
Яндекс

Яндекс

Оптимизация работы приложения на медленной сети

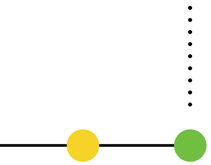
Ася Свириденко,
руководитель iOS-группы Яндекс.Почты, Санкт-Петербург

Содержание

Обнаружение проблемы

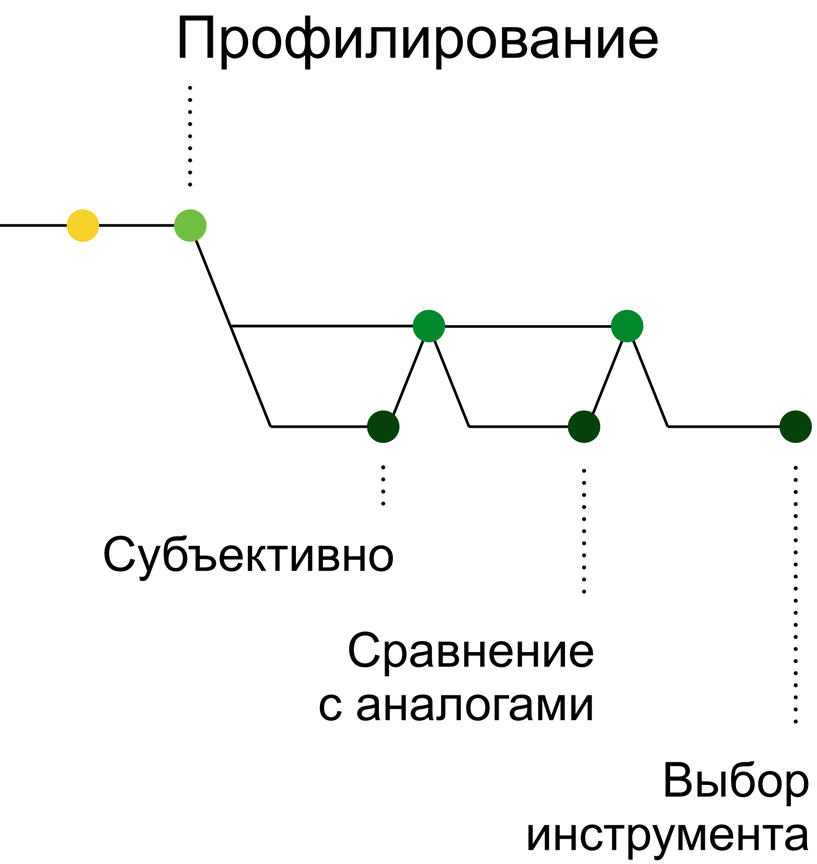
Timeline

Профилирование



Цель профилирования в этой задаче — определить, что именно мы будем оптимизировать

Timeline


Профилирование
Субъективно
Сравнение
с аналогами
Выбор
инструмента

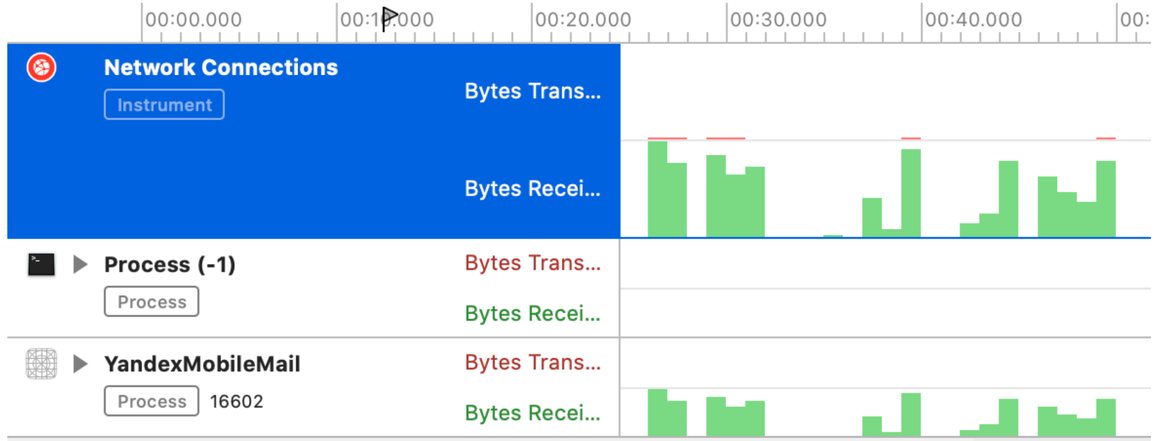
~~Network Profiler~~ Charles

Что можно получить

- хост, порт, протокол, интерфейс, адрес
- сколько всего пакетов отправлено/получено
- как растёт трафик

Что нельзя получить

- raw–packet data
- скорость выполнения запроса
- какие запросы выполняются медленнее других

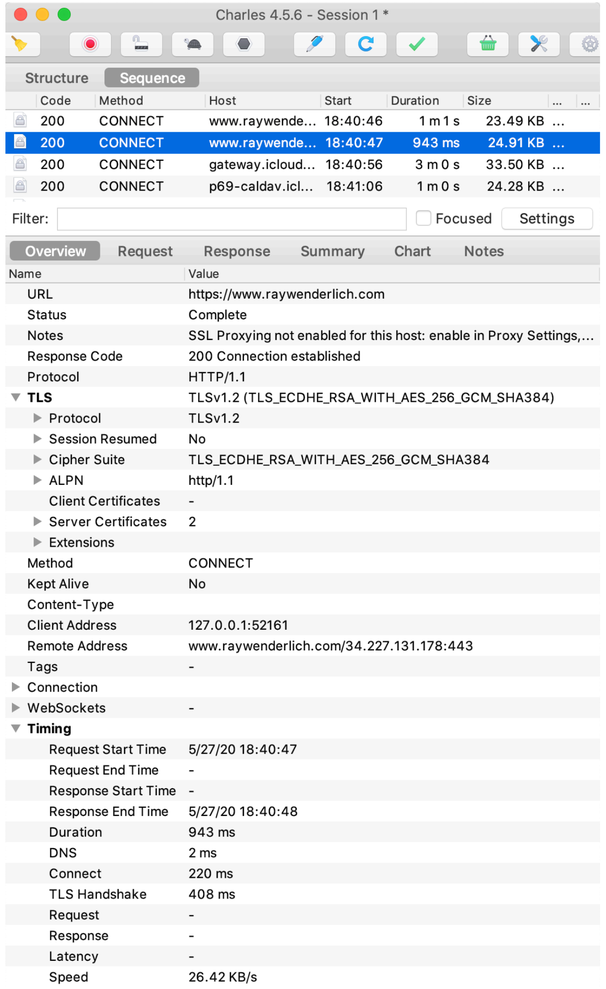


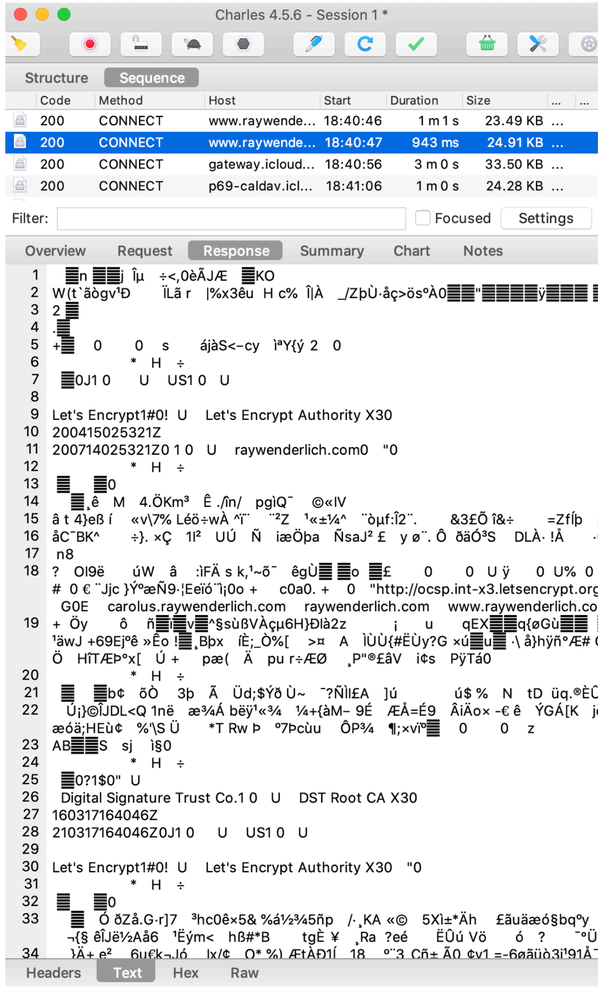
Netw...	Packets In	Bytes In	Packets Out	Bytes Out	Out-of-order	Min. Rou...	Average...
Wi-Fi	31	35.29 KiB	25	3.35 KiB	0 Bytes	19.69 ms	29.78 ms
Wi-Fi	1	52 Bytes	1	766 Bytes	0 Bytes	19.69 ms	85.50 ms
Wi-Fi	1,249	1.73 MiB	610	31.70 KiB	84.64 KiB	15.78 ms	22.53 ms
Wi-Fi	932	1.29 MiB	299	16.32 KiB	81.91 KiB	15.78 ms	18.56 ms
Wi-Fi	1,009	1.40 MiB	374	18.99 KiB	0 Bytes	15.78 ms	21.31 ms
Wi-Fi	873	1.21 MiB	560	30.02 KiB	135.16 KiB	15.78 ms	22.53 ms
Wi-Fi	989	1.37 MiB	473	24.59 KiB	68.26 KiB	15.78 ms	28.06 ms
Wi-Fi	27	36.87 KiB	4	932 Bytes	0 Bytes	15.28 ms	27.03 ms
Wi-Fi	14	17.69 KiB	4	156 Bytes	0 Bytes	15.28 ms	24.66 ms
Wi-Fi	544	768.20 KiB	92	5.86 KiB	55.22 KiB	15.28 ms	28.22 ms
Wi-Fi	130	181.35 KiB	81	4.78 KiB	0 Bytes	15.28 ms	18.19 ms
Wi-Fi	1,244	1.72 MiB	610	31.66 KiB	79.18 KiB	15.28 ms	25.56 ms
Wi-Fi	192	269.93 KiB	113	5.69 KiB	0 Bytes	15.28 ms	19.91 ms

Charles

Что можно получить

- данные HTTPS сессии
- raw–packet data
- троттлинг
- command–line tools
- и ещё много всякого





Real–time трафик

Receiving

17.6 KB/s
Per Second

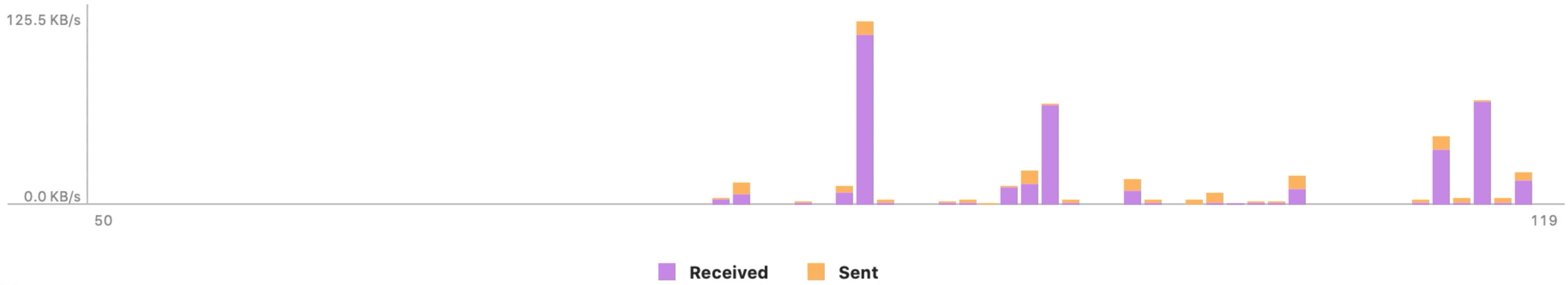
0.7 MB
Total

Sending

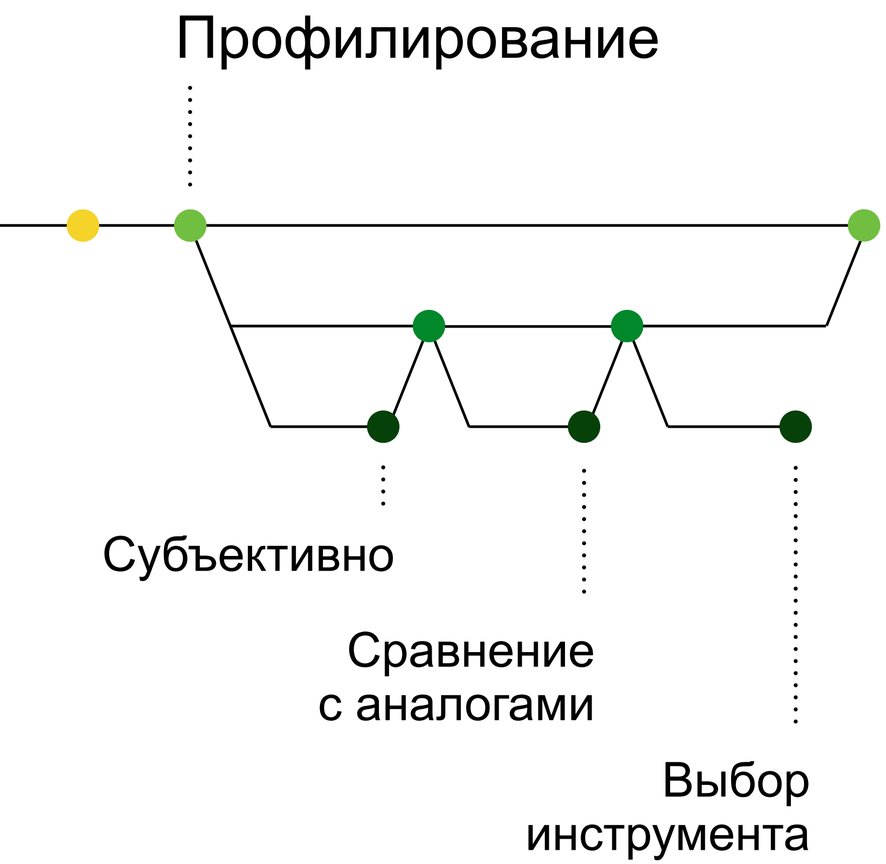
7.4 KB/s
Per Second

0.2 MB
Total

Receiving and Sending Rates

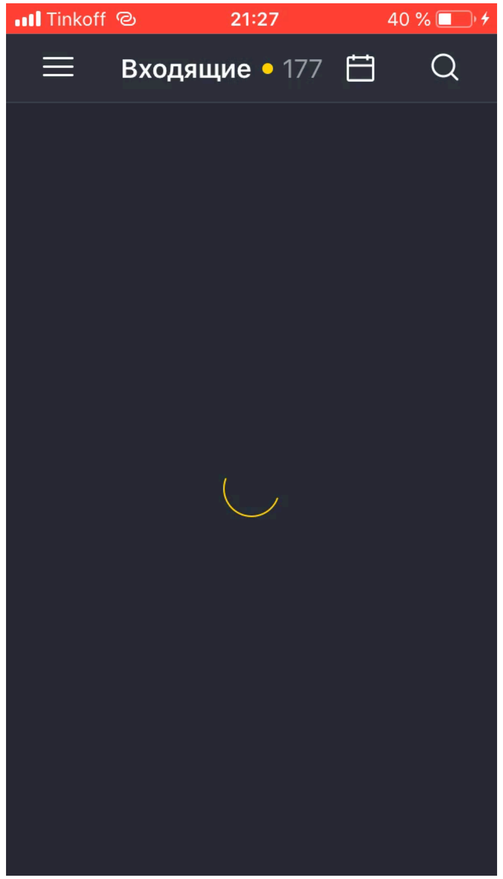


Timeline

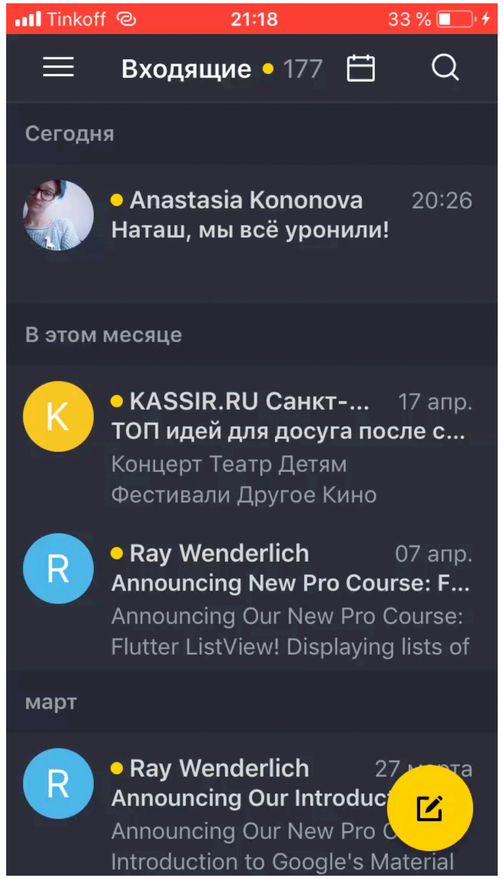

Профилирование
Субъективно
Сравнение
с аналогами
Выбор
инструмента

Какие проблемы увидели

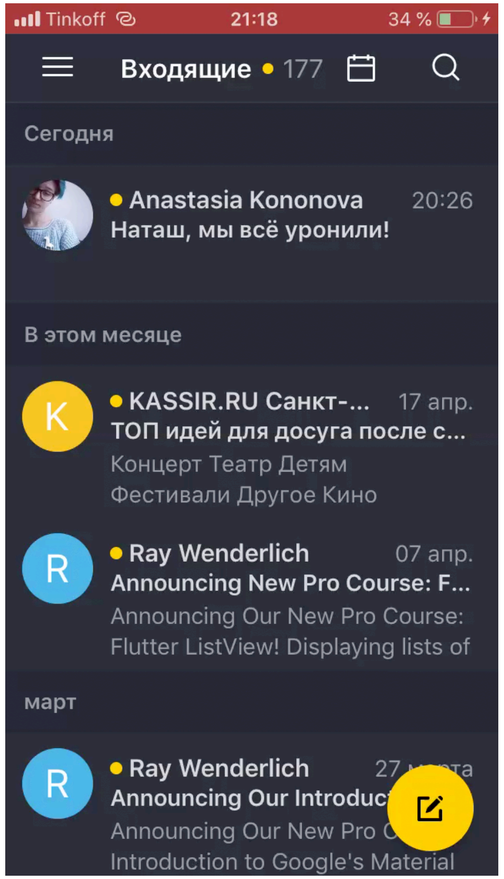
Загрузка папки ~18s



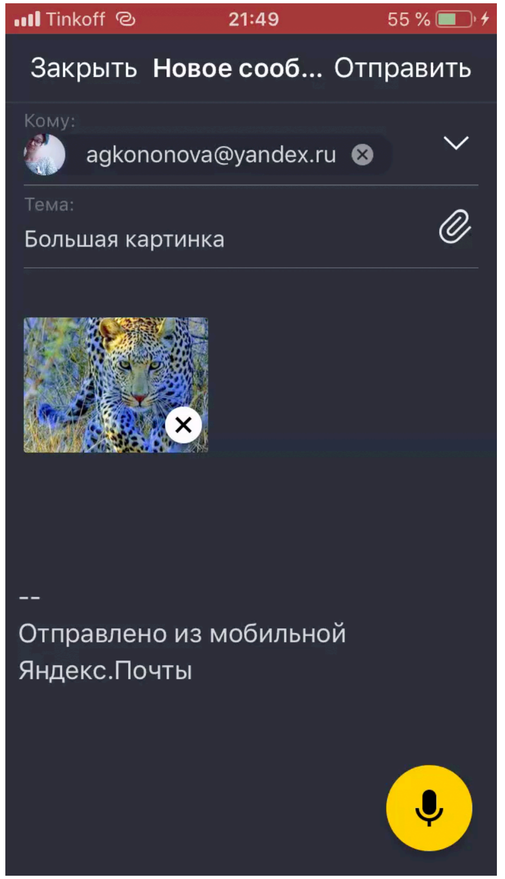
Pull to Refresh ~17s



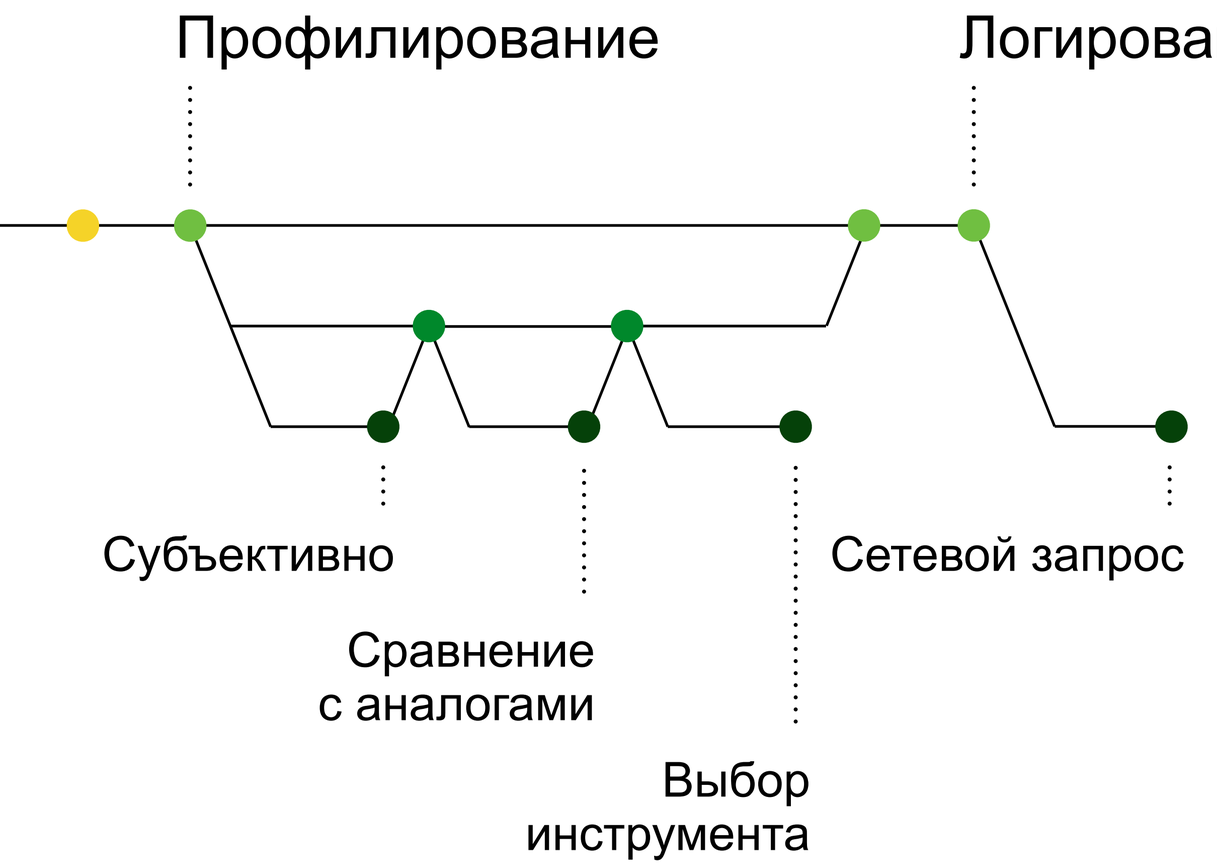
Load More ~20s



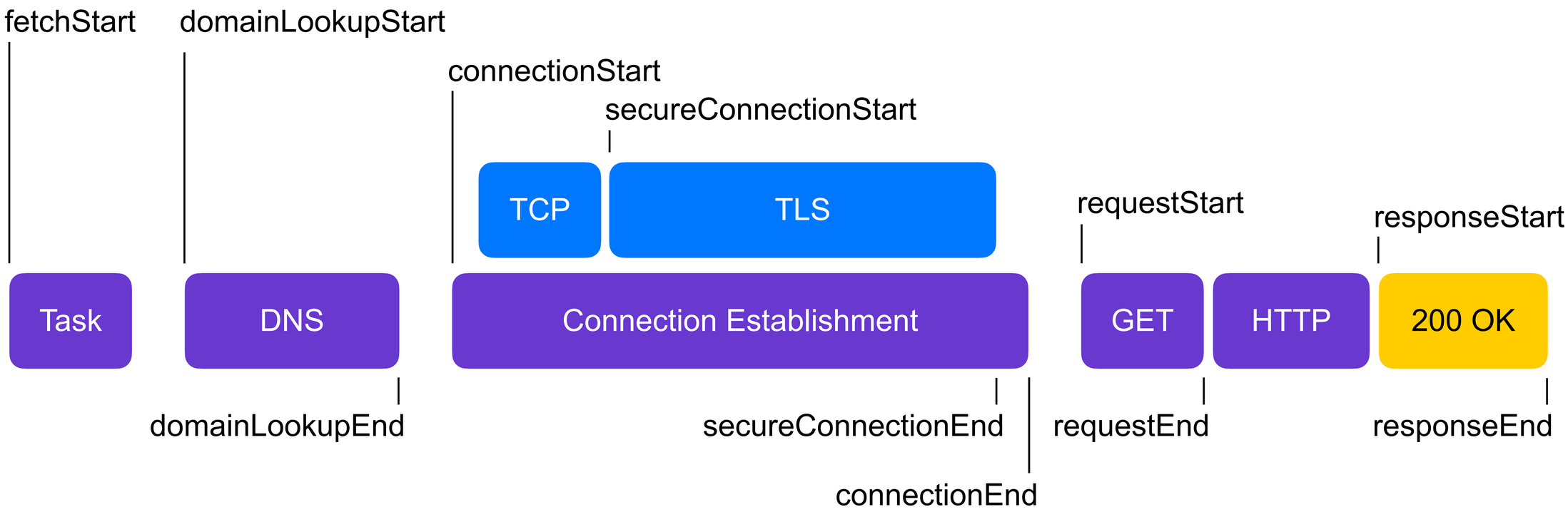
Отправка письма ~27s



Timeline


Профилирование
Логирование
Субъективно
Сравнение
с аналогами
Выбор
инструмента
Сетевой запрос

URLSessionTaskTransactionMetrics



Логирование сетевого запроса

```
let config = URLSessionConfiguration.default
let session = URLSession(configuration: config, delegate: self,
delegateQueue: nil)


// MARK :- URLSessionTaskDelegate
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                didFinishCollecting metrics:
URLSessionTaskMetrics) {
    for metric in metrics.transactionMetrics {
        print(metric)
    }
}
```

Логирование сетевого запроса

```
Fetch Start: 1588272794932
Domain Lookup Start: 1588272795014
Domain Lookup End: 1588272795037
Connect Start: 1588272795040
Secure Connection Start: 1588272795314
Secure Connection End: 1588272795597
Connect End: 1588272795597
Request Start: 1588272795591
Request End: 1588272795592
Response Start: 1588272796552
Response End: 1588272795592
Protocol Name: http/1.1
Proxy Connection: false
Reused Connection: false
Fetch Type: networkLoad
Request Header Bytes: 435
Request Body Bytes: 72
Response Header Bytes: 213
Response Body Bytes: 571
TLS Protocol Version: 0x0303
Cellular: false
Expensive: false
Constrained: false
Multipath: false
```

Логирование сетевого запроса

```
Fetch Start: 1588272794932
Domain Lookup Start: 1588272795014
Domain Lookup End: 1588272795037
Connect Start: 1588272795040
Secure Connection Start: 1588272795314
Secure Connection End: 1588272795597
Connect End: 1588272795597
Request Start: 1588272795591
Request End: 1588272795592
Response Start: 1588272796552
Response End: 1588272795592
Protocol Name: http/1.1
Proxy Connection: false
Reused Connection: false
Fetch Type: networkLoad
Request Header Bytes: 435
Request Body Bytes: 72
Response Header Bytes: 213
Response Body Bytes: 571
TLS Protocol Version: 0x0303
Cellular: false
Expensive: false
Constrained: false
Multipath: false
```

Логирование сетевого запроса

Fetch Start: 1588272794932
Domain Lookup Start: 1588272795014
Domain Lookup End: 1588272795037
Connect Start: 1588272795040
Secure Connection Start: 1588272795314
Secure Connection End: 1588272795597
Connect End: 1588272795597
Request Start: 1588272795591
Request End: 1588272795592
Response Start: 1588272796552
Response End: 1588272795592
Protocol Name: http/1.1
Proxy Connection: false
Reused Connection: false
Fetch Type: networkLoad
Request Header Bytes: 435
Request Body Bytes: 72
Response Header Bytes: 213
Response Body Bytes: 571
TLS Protocol Version: 0x0303
Cellular: false
Expensive: false
Constrained: false
Multipath: false

Логирование сетевого запроса

Fetch Start: 1588272794932

Domain Lookup Start: 1588272795014

Domain Lookup End: 1588272795037

Connect Start: 1588272795040

Secure Connection Start: 1588272795314

Secure Connection End: 1588272795597

Connect End: 1588272795597

Request Start: 1588272795591

Request End: 1588272795592

Response Start: 1588272796552

Response End: 1588272795592

Protocol Name: http/1.1

Proxy Connection: false

Reused Connection: false

Fetch Type: networkLoad

Request Header Bytes: 435

Request Body Bytes: 72

Response Header Bytes: 213

Response Body Bytes: 571

TLS Protocol Version: 0x0303

Cellular: false

Expensive: false

Constrained: false

Multipath: false

Логирование сетевого запроса

Fetch Start: 1588272794932

Domain Lookup Start: 1588272795014

Domain Lookup End: 1588272795037

Connect Start: 1588272795040

Secure Connection Start: 1588272795314

Secure Connection End: 1588272795597

Connect End: 1588272795597

Request Start: 1588272795591

Request End: 1588272795592

Response Start: 1588272796552

Response End: 1588272795592

Protocol Name: http/1.1

Proxy Connection: false

Reused Connection: false

Fetch Type: networkLoad

Request Header Bytes: 435

Request Body Bytes: 72

Response Header Bytes: 213

Response Body Bytes: 571

TLS Protocol Version: 0x0303

Cellular: false

Expensive: false

Constrained: false

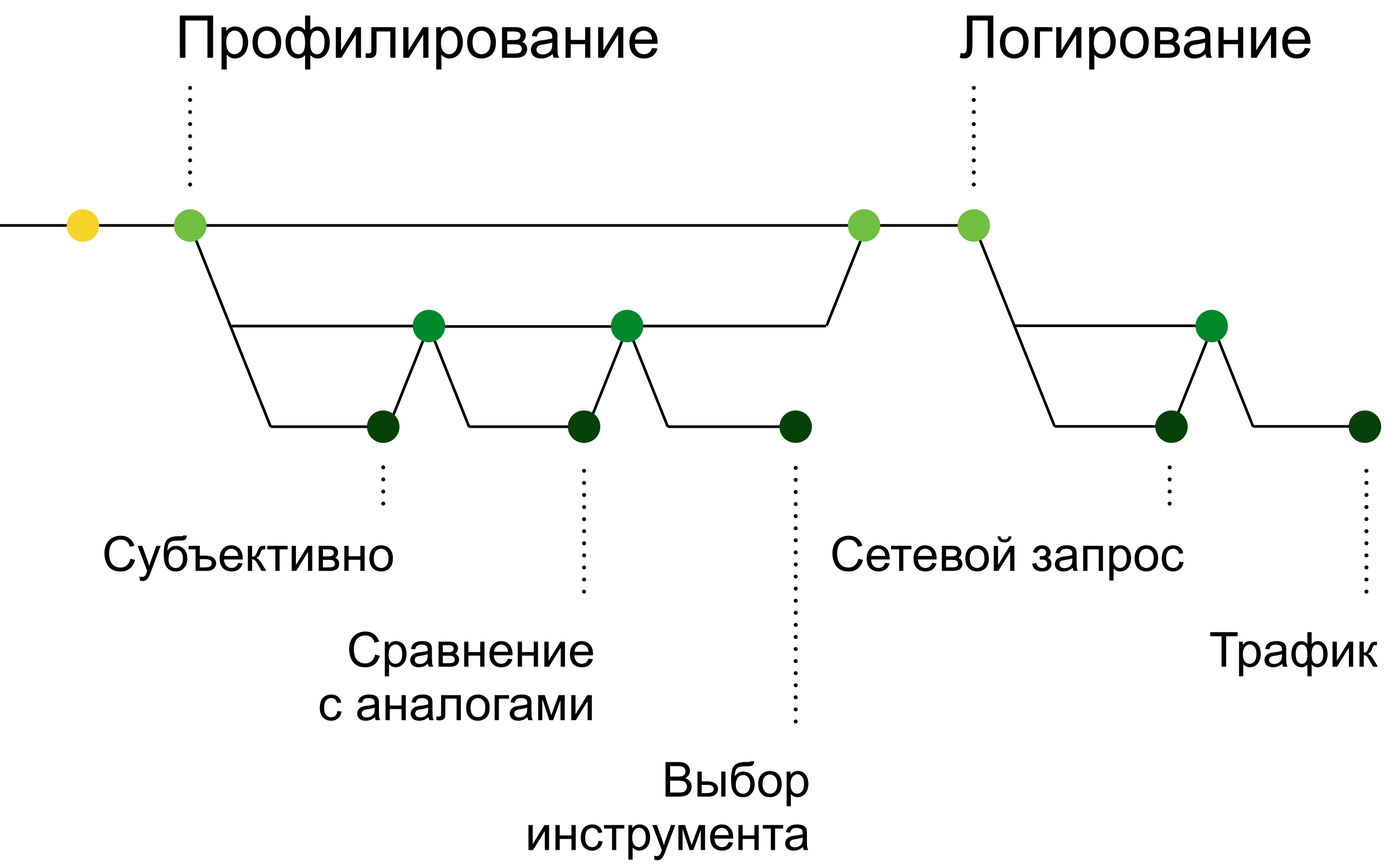
Multipath: false

Логирование сетевого запроса

```
Fetch Start: 1588272794932
Domain Lookup Start: 1588272795014
Domain Lookup End: 1588272795037
Connect Start: 1588272795040
Secure Connection Start: 1588272795314
Secure Connection End: 1588272795597
Connect End: 1588272795597
Request Start: 1588272795591
Request End: 1588272795592
Response Start: 1588272796552
Response End: 1588272795592
Protocol Name: http/1.1
```

```
Proxy Connection: false
Reused Connection: false
Fetch Type: networkLoad
Request Header Bytes: 435
Request Body Bytes: 72
Response Header Bytes: 213
Response Body Bytes: 571
TLS Protocol Version: 0x0303
Cellular: false
Expensive: false
Constrained: false
Multipath: false
```

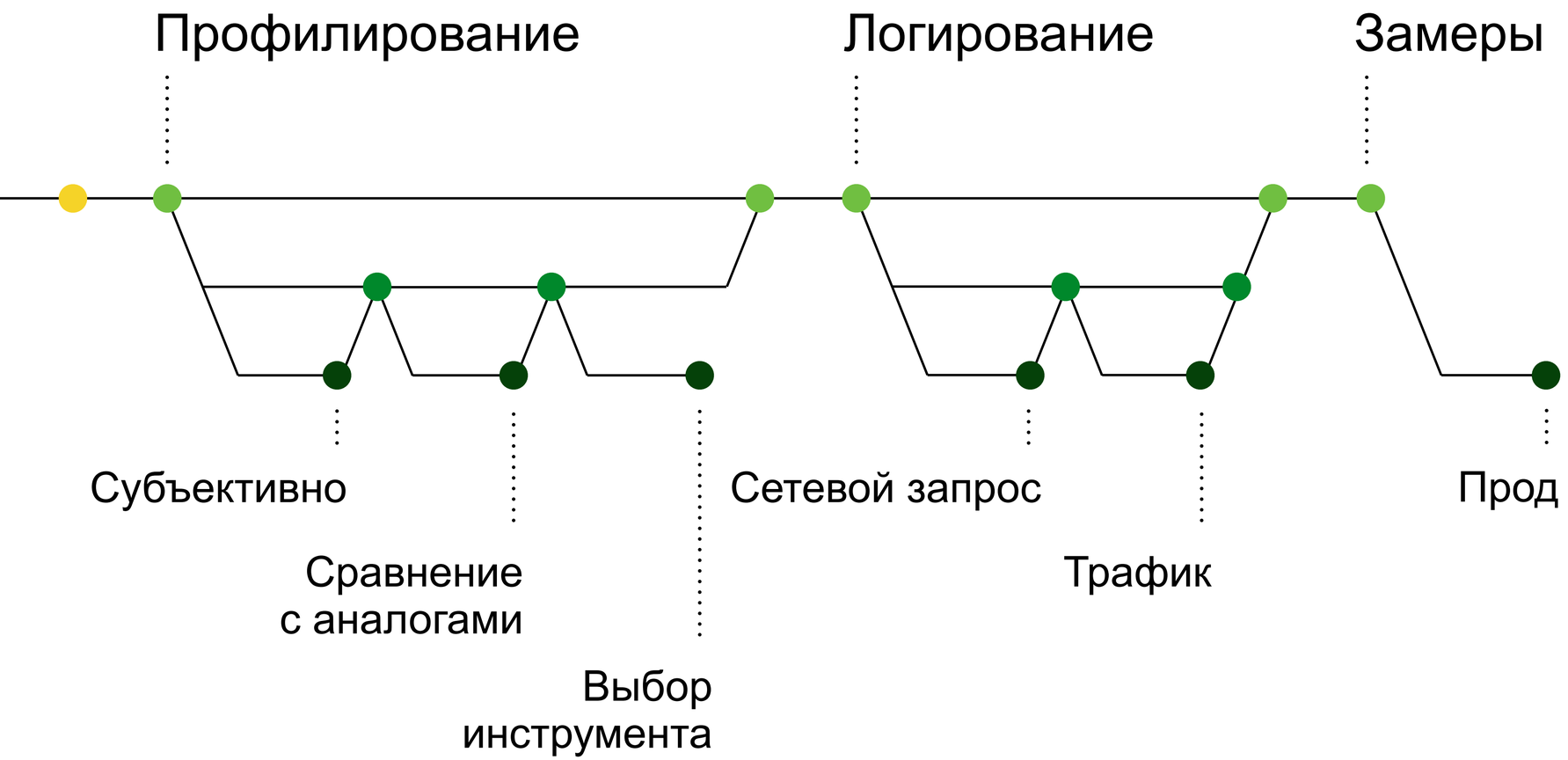
Timeline


Профилирование
Логирование
Субъективно
Сравнение
с аналогами
Выбор
инструмента
Сетевой запрос
Трафик

Логирование трафика

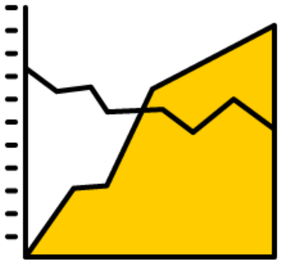
```
// MARK :- URLSessionTaskDelegate
func urlSession(_ session: URLSession, task: URLSessionTask,
                didFinishCollecting metrics:
URLSessionTaskMetrics) {
    for metric in metrics.transactionMetrics {
        if #available(iOS 13.0, *) {
            print(metric.countOfRequestBodyBytesSent)
            print(metric.countOfResponseBodyBytesReceived)
        } else {
            print(task.countOfBytesSent)
            print(task.countOfBytesReceived)
        }
    }
}
```

Timeline


Профилирование
Логирование
Замеры
Субъективно
Сравнение
с аналогами
Выбор
инструмента
Сетевой запрос
Трафик
Прод

Реальные пользователи

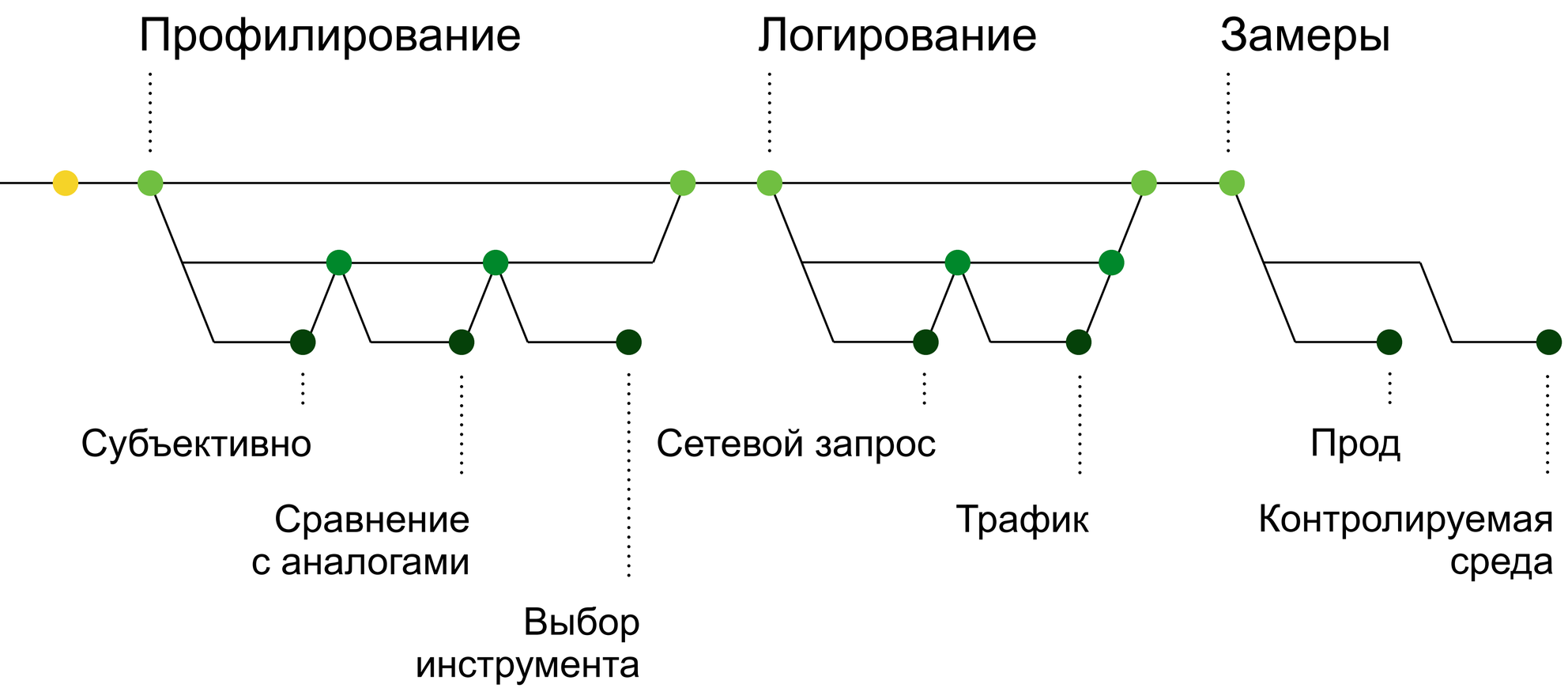
Реальные данные



Большой разброс данных

Тяжело измерять трафик

Timeline


Профилирование
Логирование
Замеры
Субъективно
Сравнение
с аналогами
Выбор
инструмента
Сетевой запрос
Трафик
Прод
Контролируемая
среда

Контролируемая среда

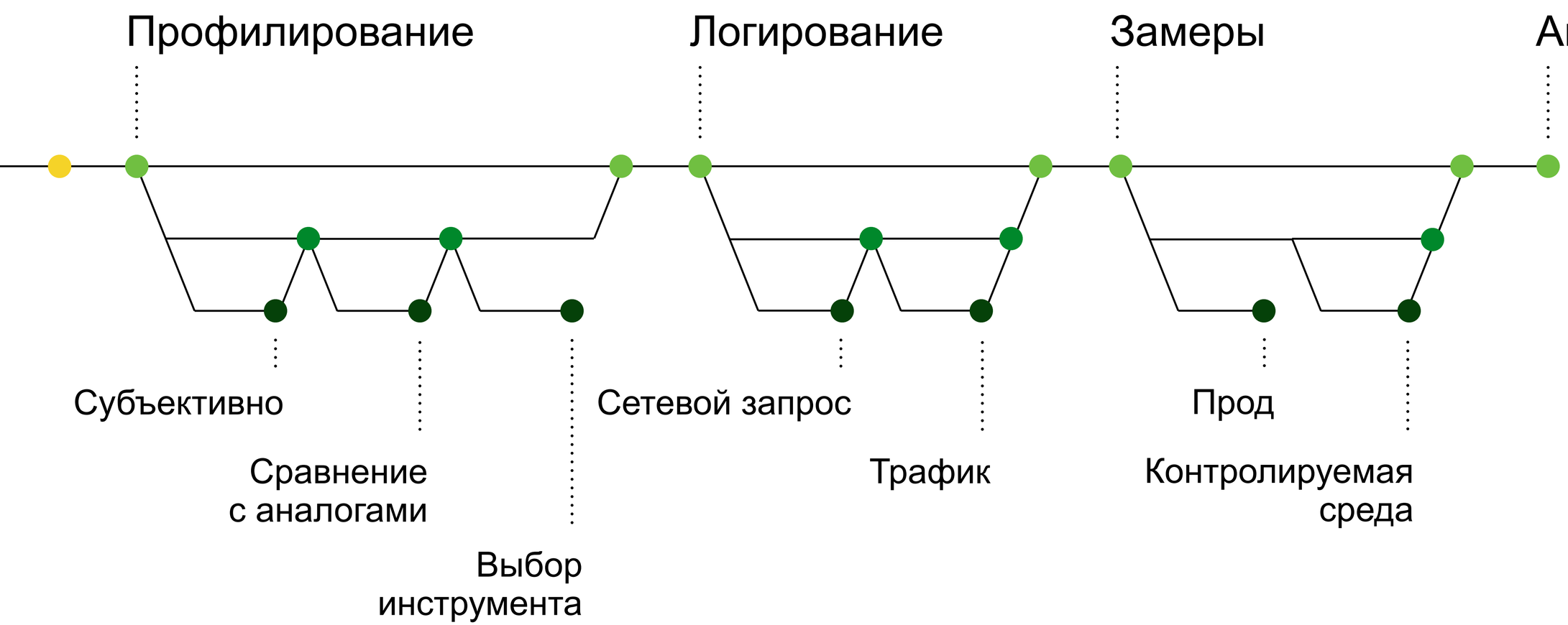
Только необходимые замеры

Быстрый сбор данных

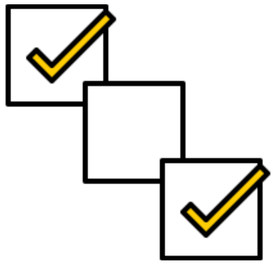
Нужны автотесты

¯_(ツ)_/¯

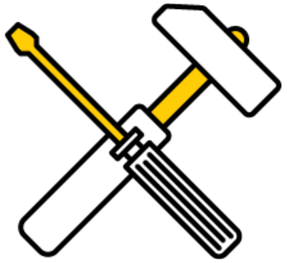
Timeline


Профилирование
Логирование
Замеры
Автотесты
Субъективно
Сравнение
с аналогами
Выбор
инструмента
Сетевой запрос
Трафик
Прод
Контролируемая
среда

Автотесты + контролируемая среда



TDD ❤️



Автотесты
на базе XCTest

Эмуляция
плохой сети

Запуск на CI

XProxy

```
export class CommonConfigurations {
    static edge(toxy: Toxy): Toxy {
        return toxy.all('/*')
            .poison(Poisons.latency(null, 10, 50))
            .poison(Poisons.bandWidth(1000, 100))
    }

    static random500(toxy: Toxy, percent_str: string): Toxy {
        let percent = Utils.parseIntUnsafe(percent_str);
        return toxy.all('/*')
                .poison(Poisons.responseWithDelay(1000))
                .poison(Poisons.responseError500())
                .withRule(Rules.probability(percent));
    }
}
```

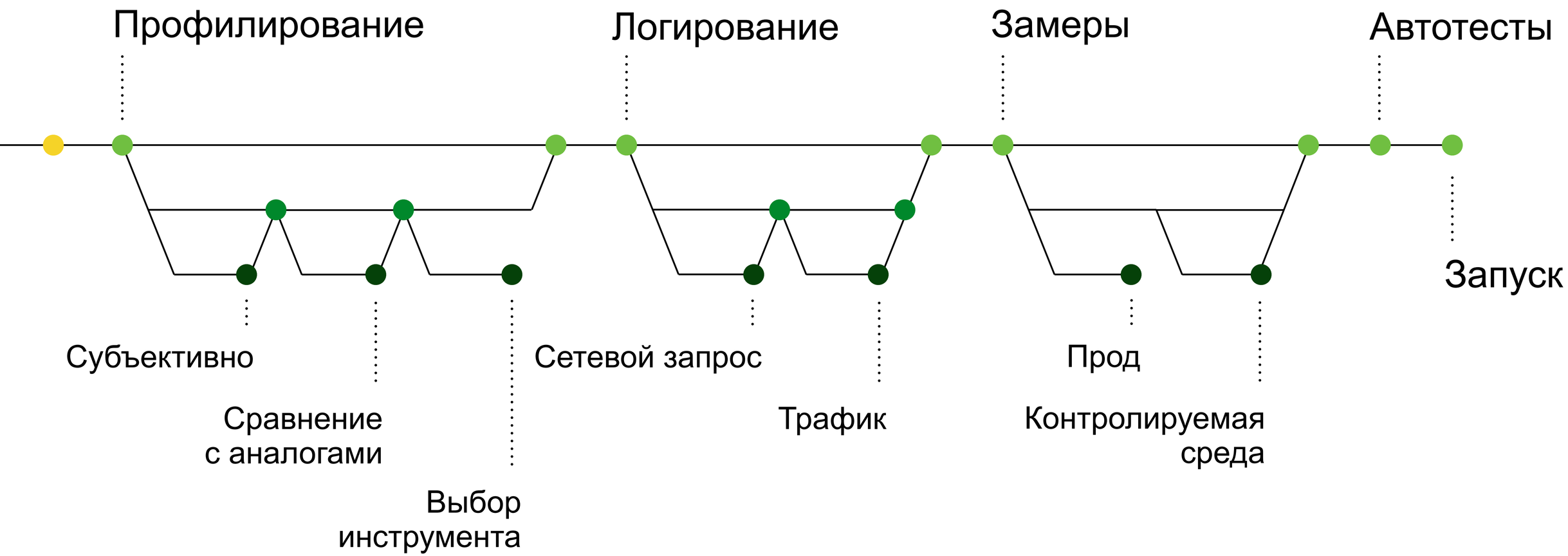
Autotest + XProxy

```
public class XProxyTestCase: XCTestCase {
    private static let proxyHost = "https://xproxy.test.ru"

    var configuration: XProxyConfiguration {
       return .edge
    }

    public var launchArguments: [String] {
        let path = “\(XProxyTestCase.proxyHost)/c/
                    \(self.configuration)/mobapi”
        return [CommandLineArguments.baseURLKey, path]
    }
}
```

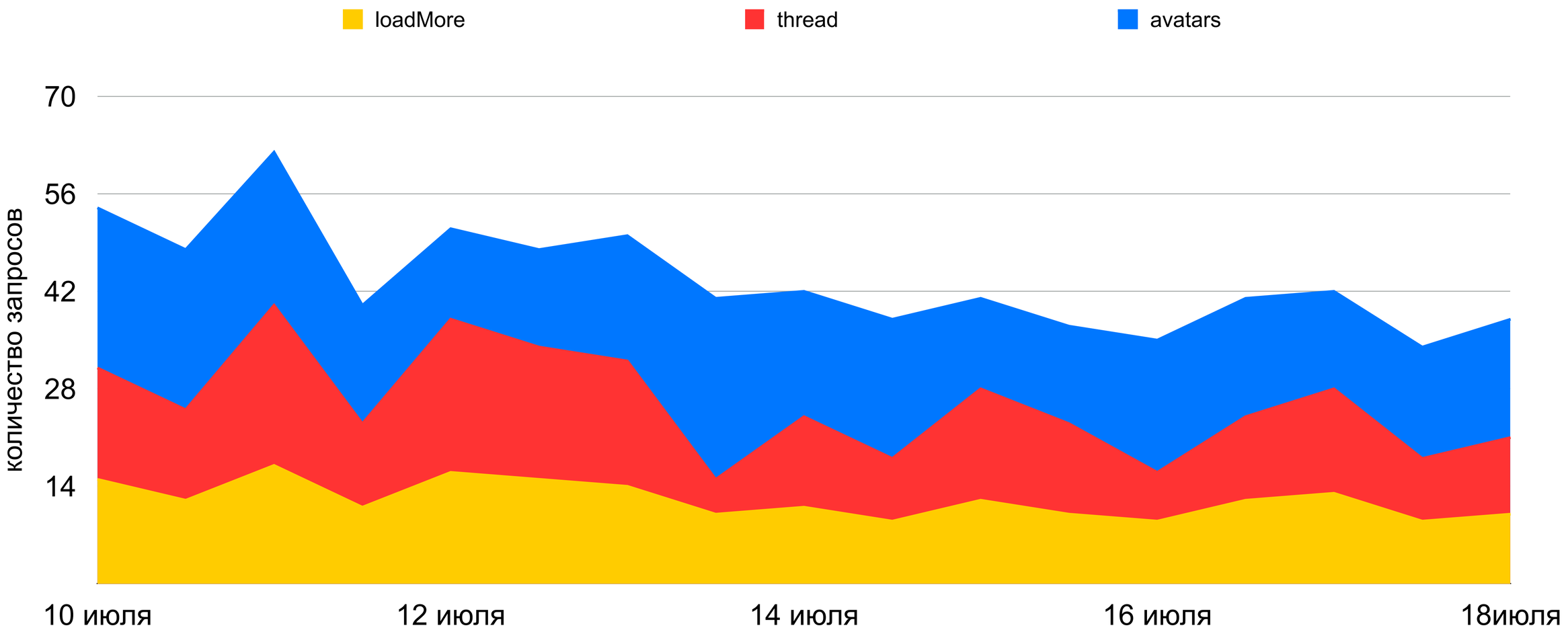
Autotest + XProxy

```
public final class PTRMessagesTest: XProxyTestCase {
    public override var launchArguments: [String] {
        let useCase = [CommandLineArguments.networkMetricsEventName,
                       UseCases.pullRoRefresh.rawValue]
        return super.launchArguments + useCase
    }

    func testPullToRefreshNetworkMetrics() {
        let plan = PredefinedTestSteps
                    .login(user: UsersPool.testUser1)
                    .then { $0.pullToRefresh }
        ActionsRunner.performPlan(plan)
    }
}
```

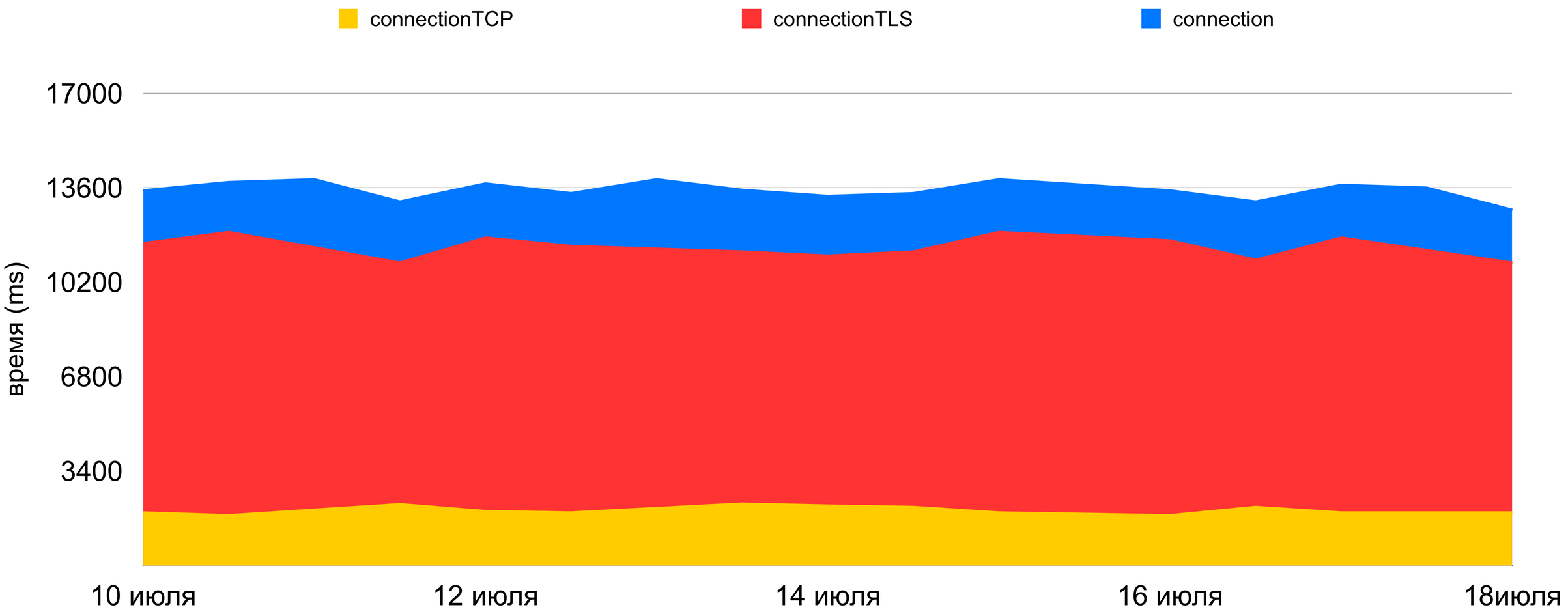
Timeline


Профилирование
Логирование
Замеры
Автотесты
Запуск
Субъективно
Сравнение
с аналогами
Выбор
инструмента
Сетевой запрос
Трафик
Прод
Контролируемая
среда

Запуск: запросы



Запуск: connection

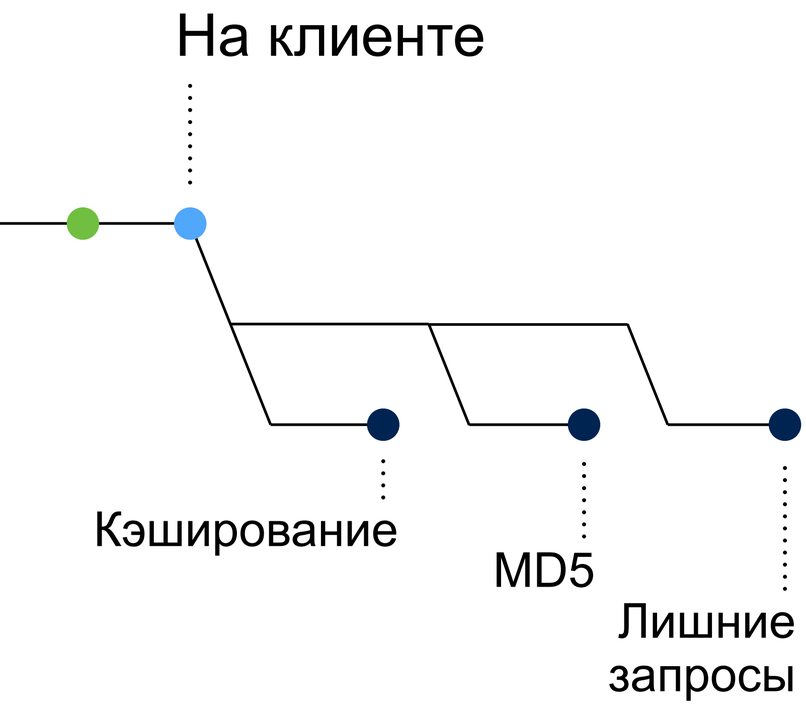


Проблемы, которые нашли

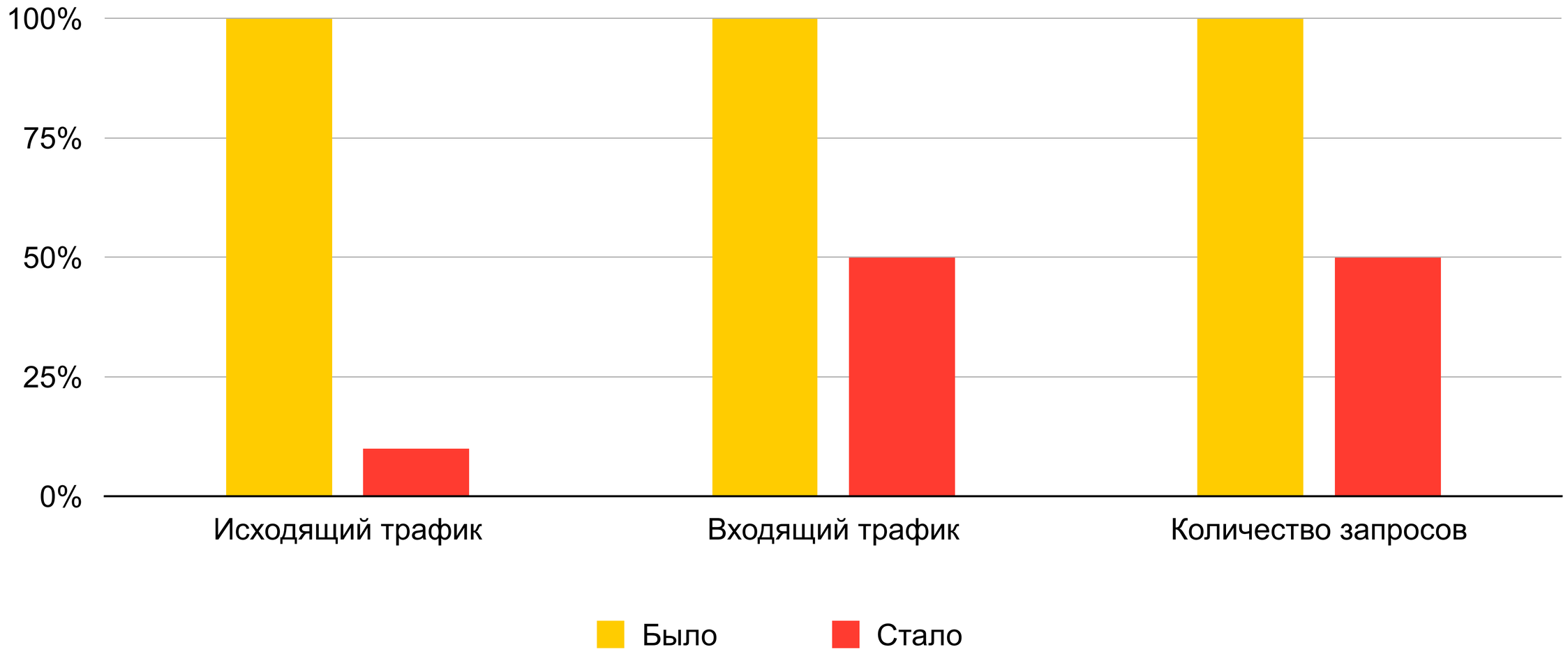
- Делаем лишние запросы «тонких» тредов
- Не сжимаем аттачи при отправлении письма
- Загружаем аватарки по одной
- На каждый запрос долгая установка соединения

Оптимизация запросов и трафика

Timeline



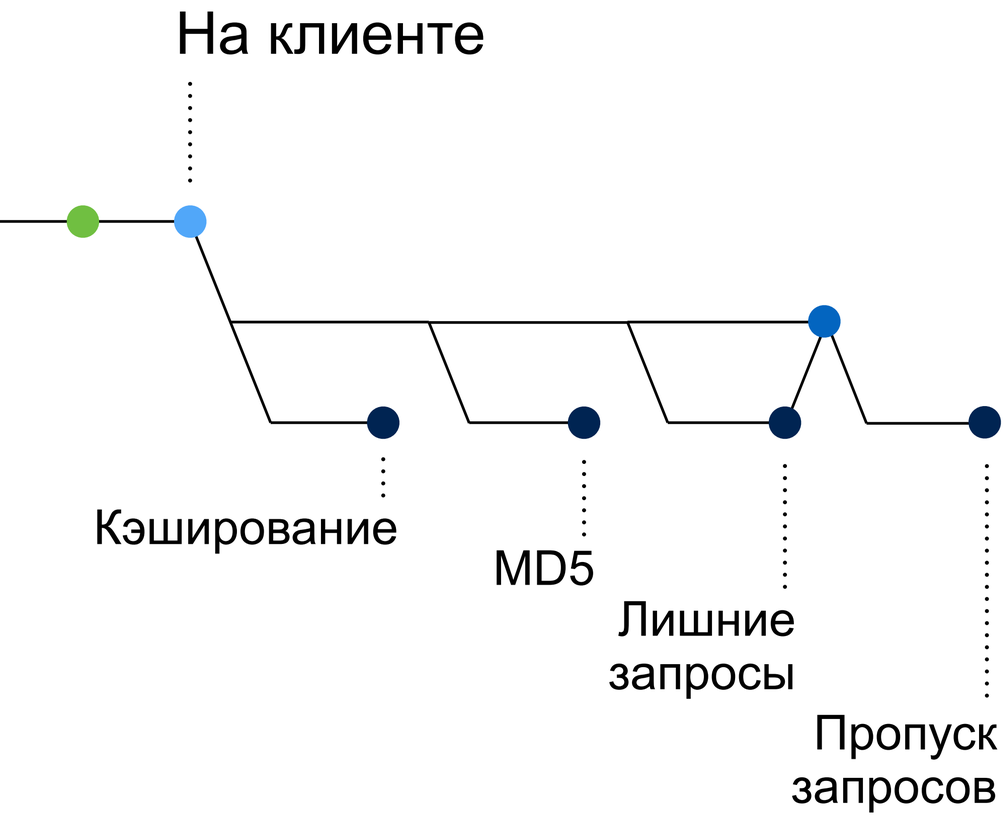
Что дало удаление лишних запросов



Проблемы, которые нашли

- ~~Делаем лишние запросы для «тонких» тредов~~
- Не сжимаем аттачи при отправлении письма
- Загружаем аватарки по одной
- На каждый запрос долгая установка соединения

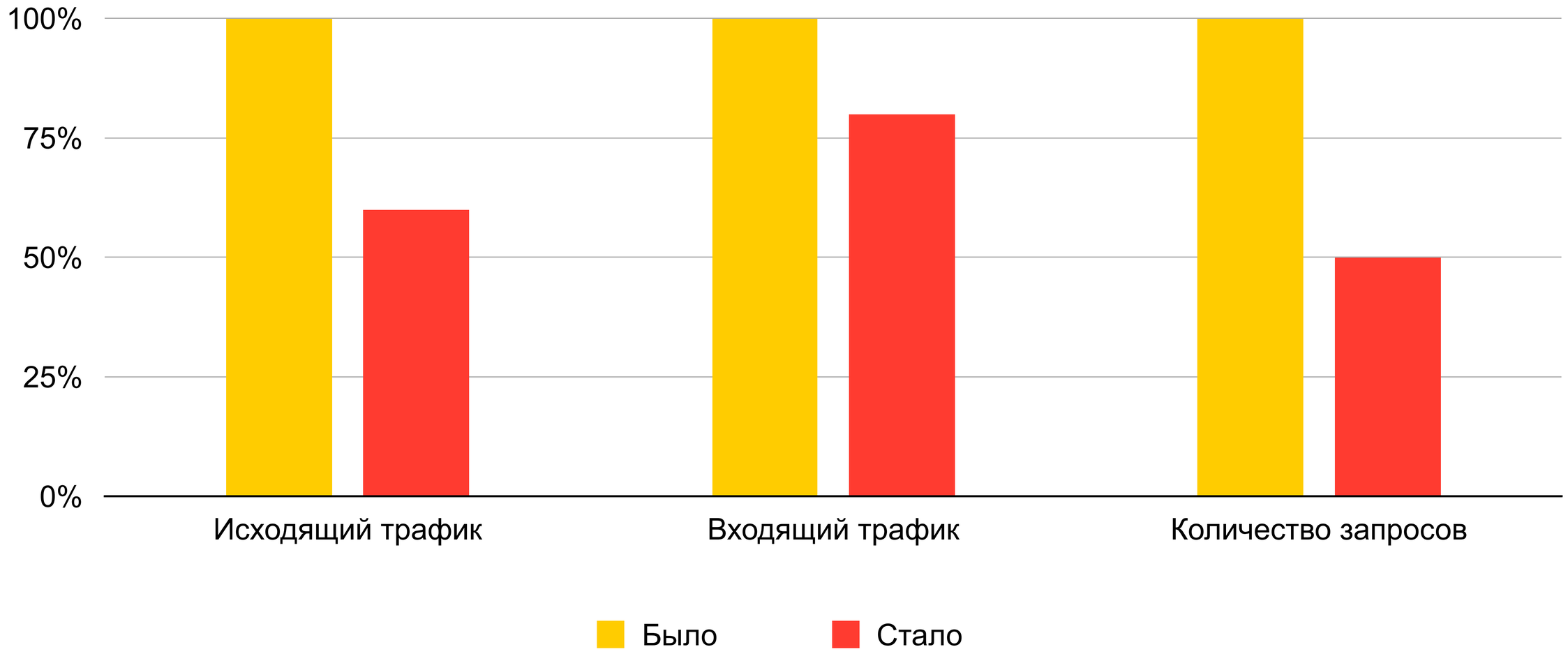
Timeline


На клиенте
Кэширование
MD5
Лишние
запросы
Пропуск
запросов

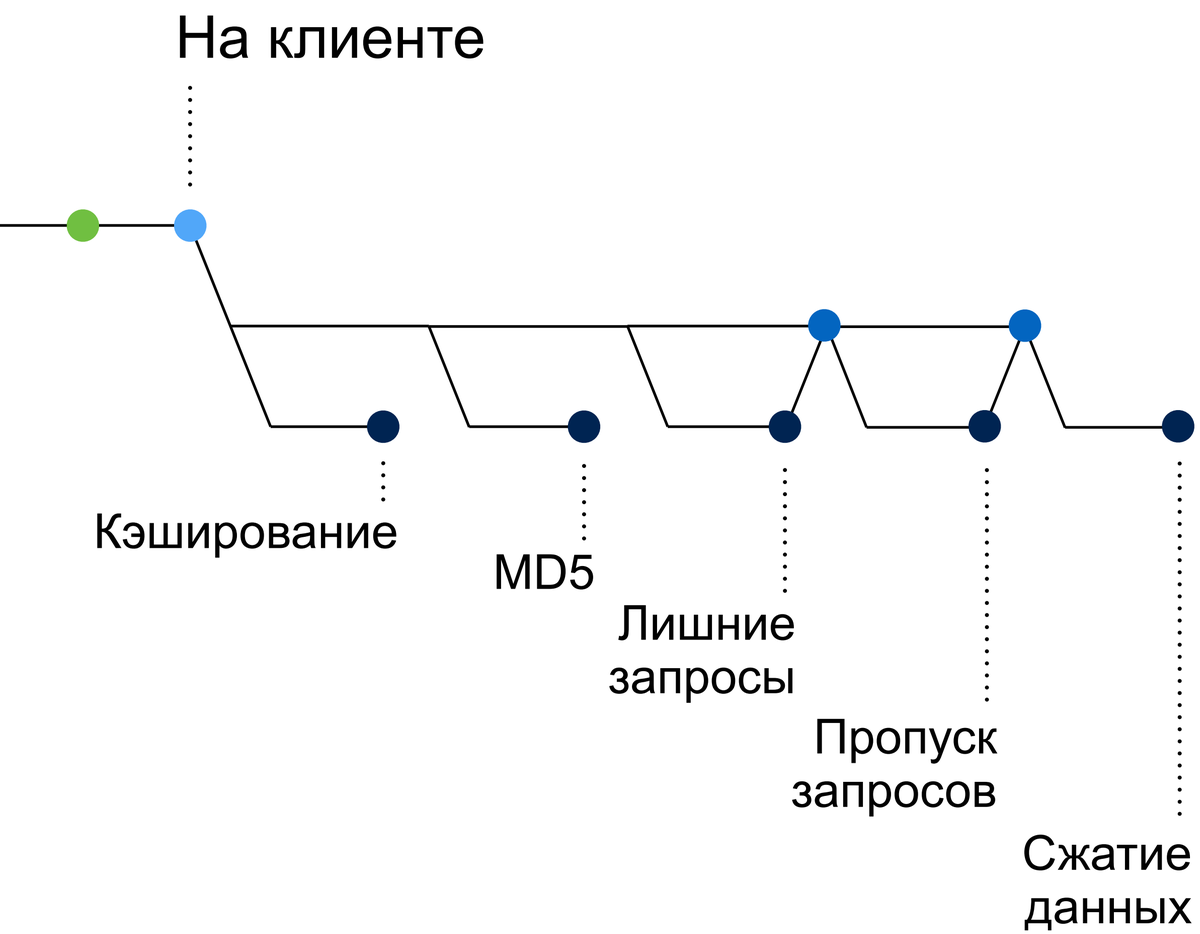
Как определить тип сети

```
static var current: NetworkType {
#if TEST
    return CommandLineArgumentsParser.useBadNetworkConnection ? .edge : .undefined
#endif
    if self.reachabilityManager.isReachableViaWiFi {
        return .wifi
    }
    guard let technology = CTTelephonyNetworkInfo()
                          .serviceCurrentRadioAccessTechnology?.values.first else {
        return .undefined
    }
    switch currentTechnology {
    case CTRadioAccessTechnologyEdge:
        return .edge
    case CTRadioAccessTechnologyHSDPA,
         CTRadioAccessTechnologyHSUPA:
        return .highSpeed
    case CTRadioAccessTechnologyLTE:
        return .lte
    default:
        return .undefined
    }
}
```

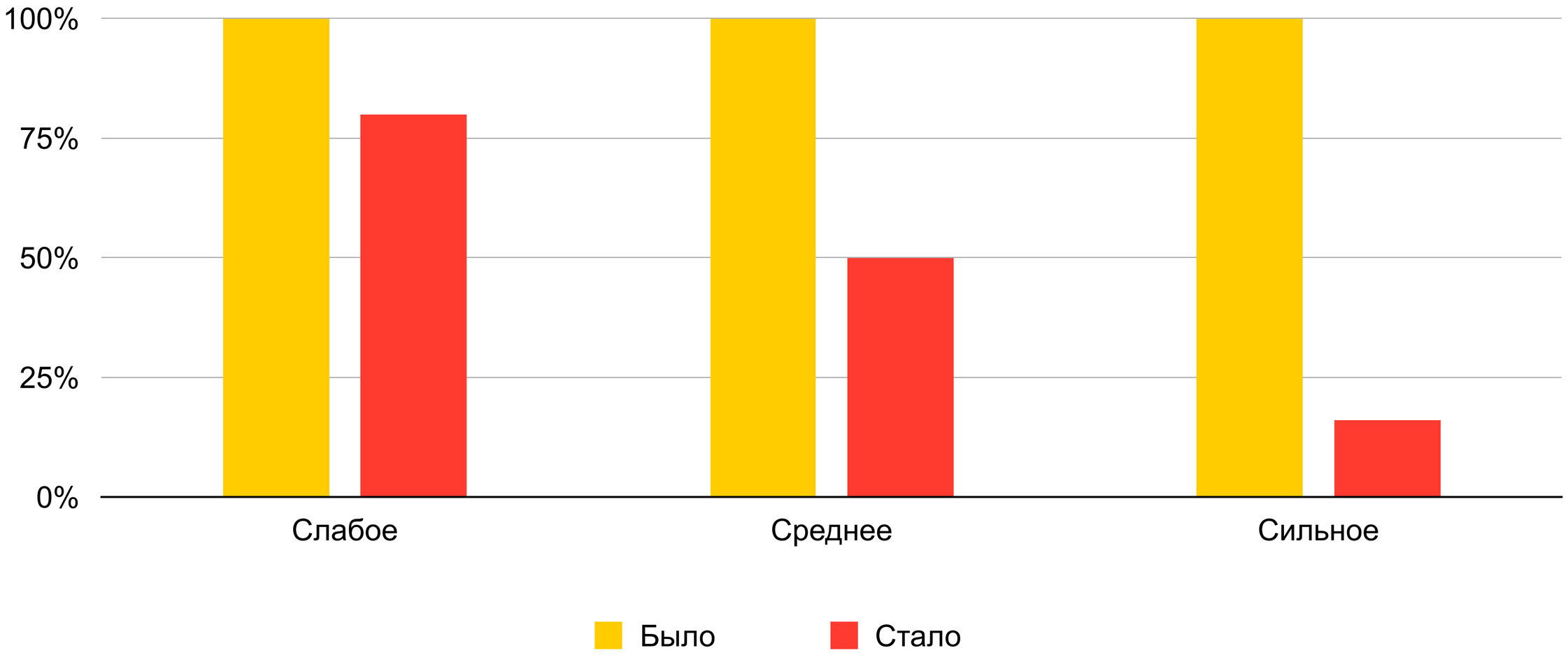
Что дал пропуск запросов



Timeline


На клиенте
Кэширование
MD5
Лишние
запросы
Пропуск
запросов
Сжатие
данных

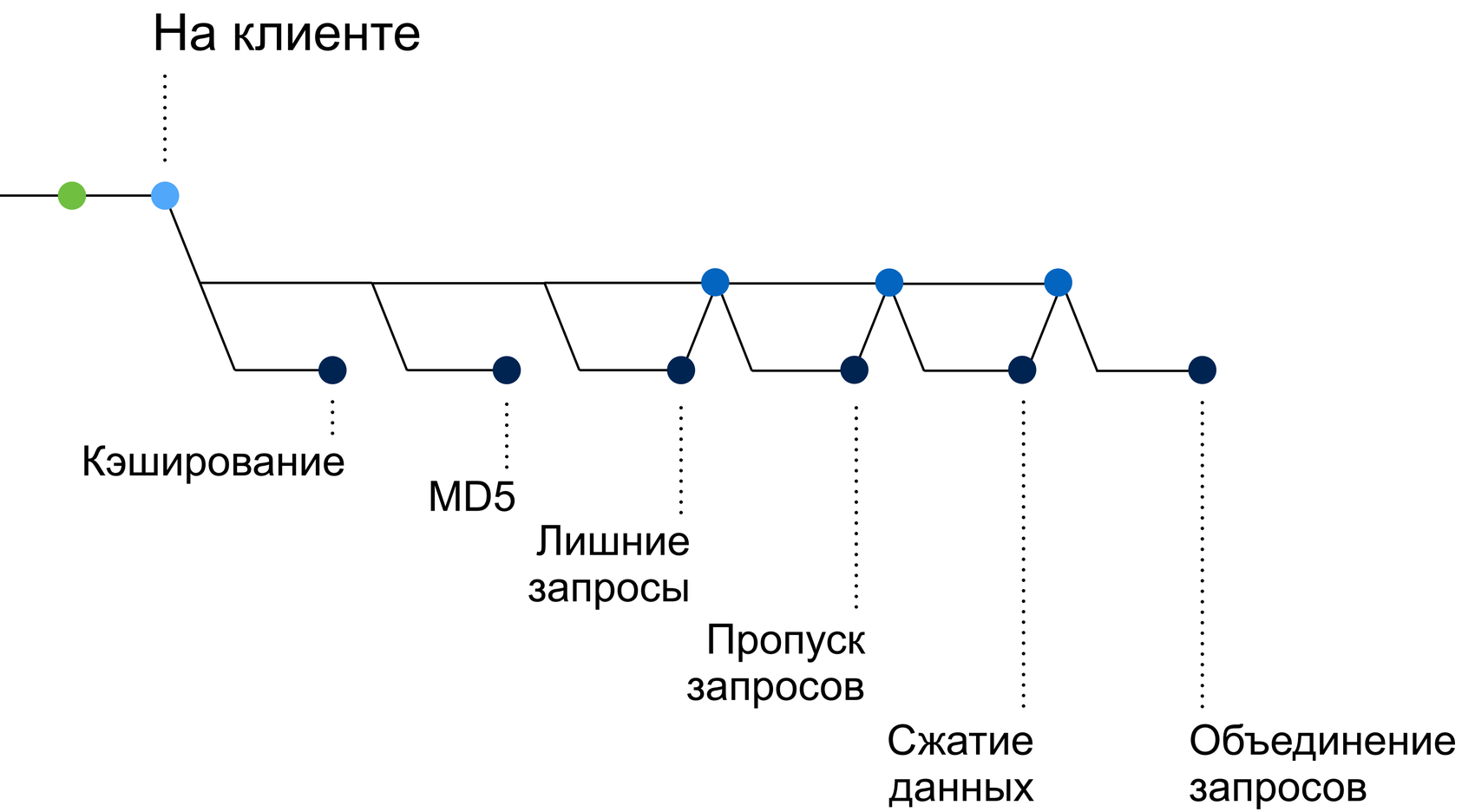
Что дало сжатие аттачей



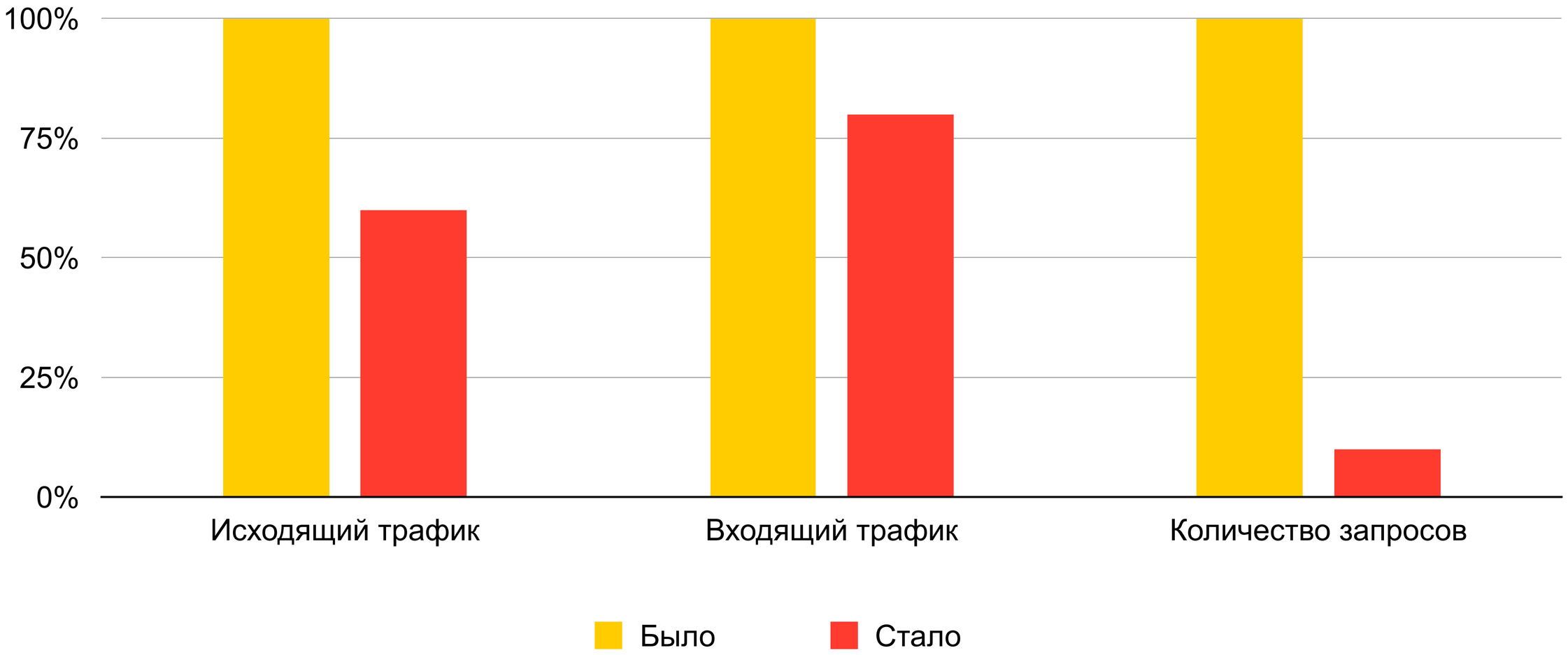
Проблемы, которые нашли

- ~~Делаем лишние запросы для «тонких» тредов~~
- ~~Не сжимаем аттачи при отправлении письма~~
- Загружаем аватарки по одной
- На каждый запрос долгая установка соединения

Timeline


На клиенте
Кэширование
MD5
Лишние запросы
Пропуск запросов
Сжатие данных
Объединение запросов

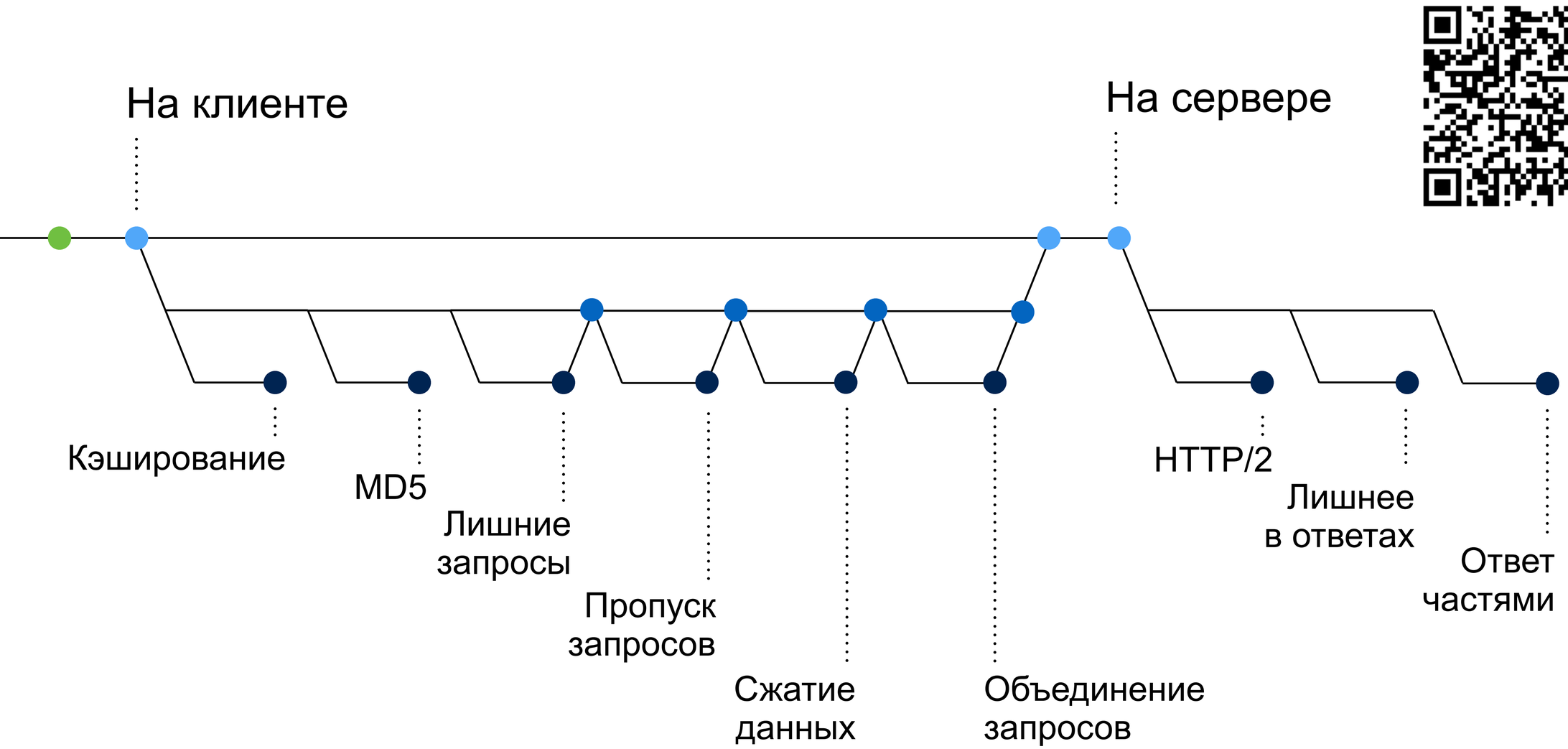
Что дало объединение запросов



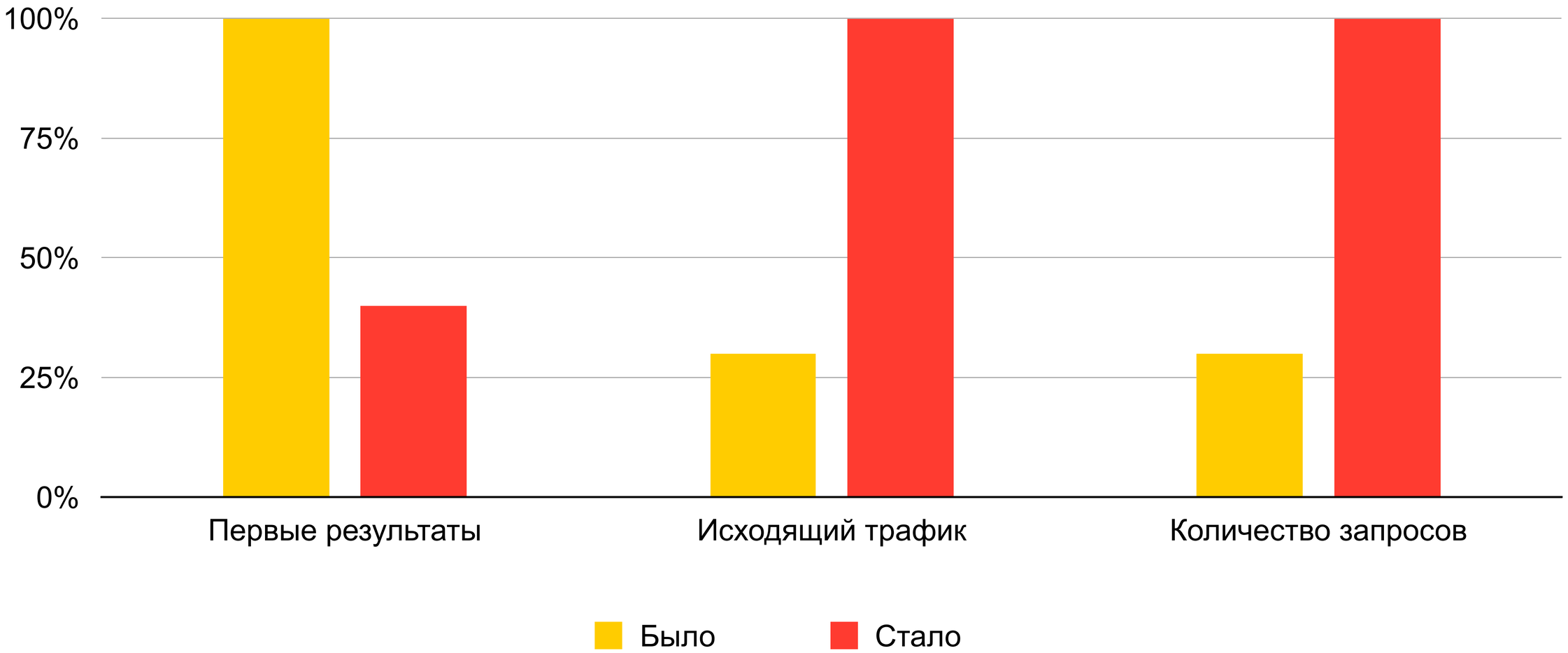
Проблемы, которые нашли

- ~~Делаем лишние запросы для «тонких» тредов~~
- ~~Не сжимаем аттачи при отправлении письма~~
- ~~Загружаем аватарки по одной~~
- На каждый запрос долгая установка соединения

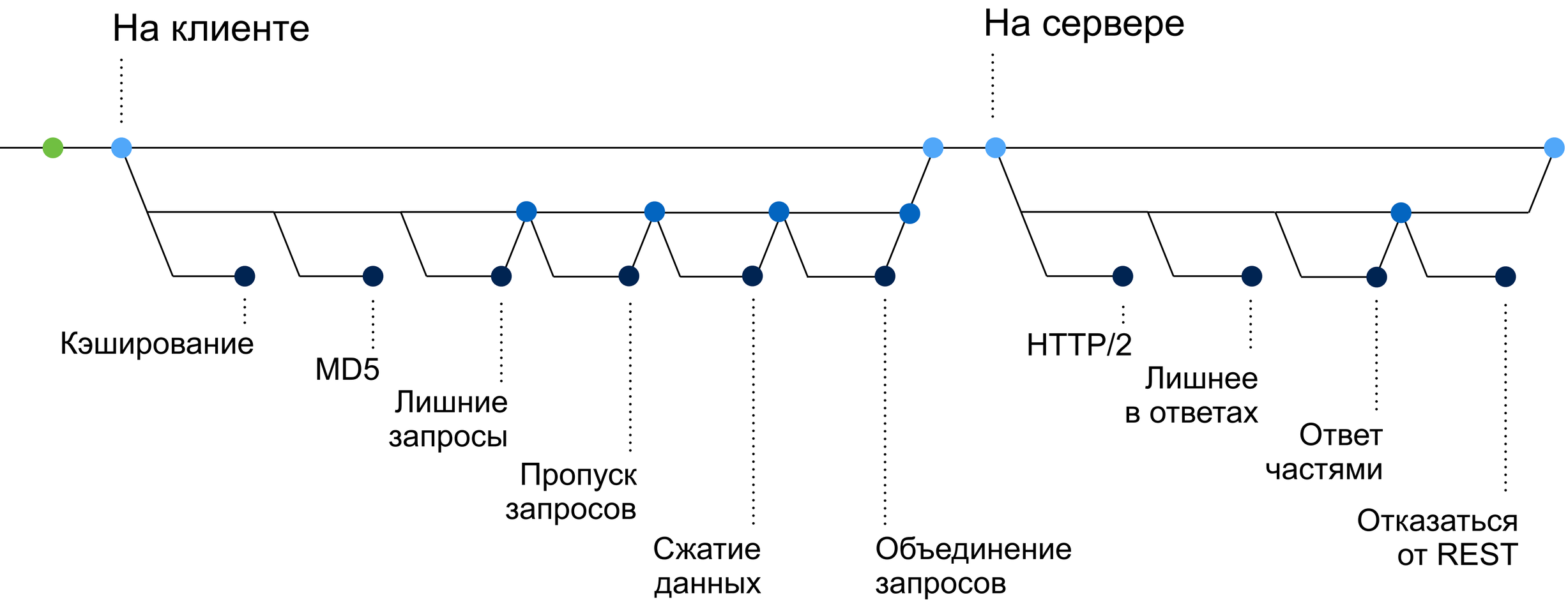
Timeline


На клиенте
На сервере
Кэширование
MD5
Лишние запросы
Пропуск запросов
Сжатие данных
Объединение запросов
HTTP/2
Лишнее в ответах
Ответ частями

Что дал ответ частями

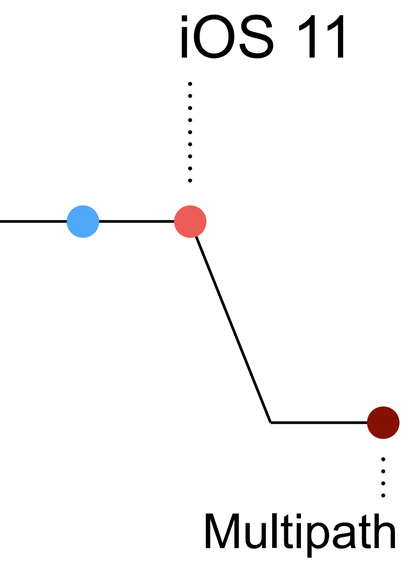


Timeline


На клиенте
На сервере
Кэширование
MD5
Лишние запросы
Пропуск запросов
Сжатие данных
Объединение запросов
HTTP/2
Лишнее в ответах
Ответ частями
Отказаться от REST

Что может предложить iOS

Timeline



Multipath

Автоматически передаёт данные по сотовой сети
при недостаточном уровне сигнала Wi-Fi

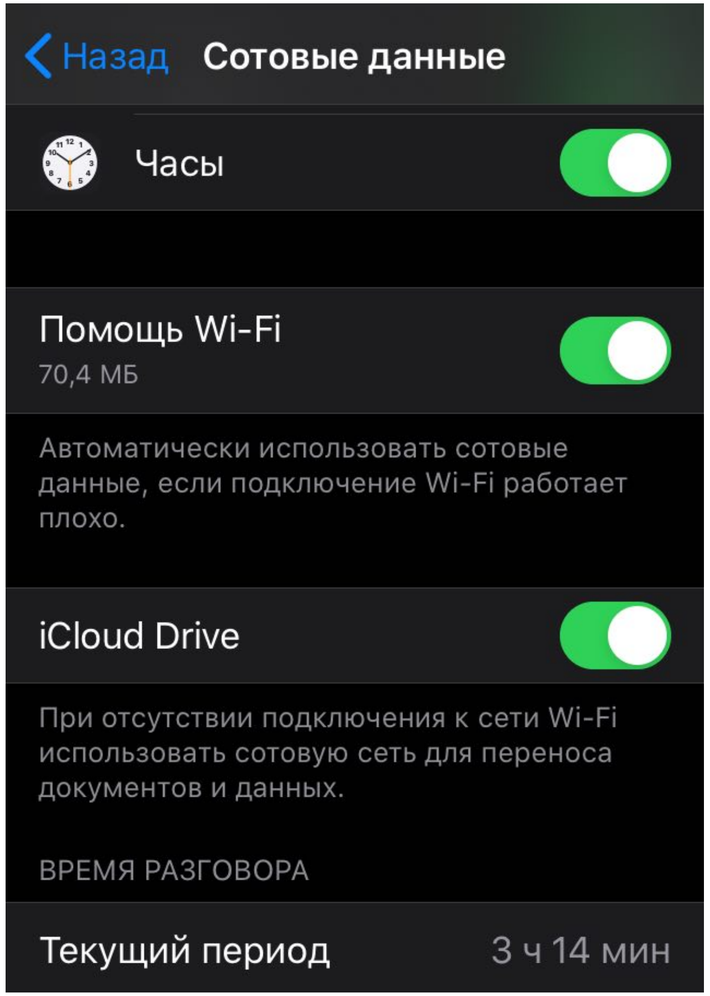
Xcode Capabilities -> Multipath Entitlement

```
let configuration = URLSessionConfiguration.default
configuration.multipathServiceType = .none
// .handover – переключаться между Wi-Fi и cellular
// .interactive – будет выбран lowest-latency интерфейс
// .aggregate – объединяет возможности остальных, рекомендуется
для разработки/тестирования
```

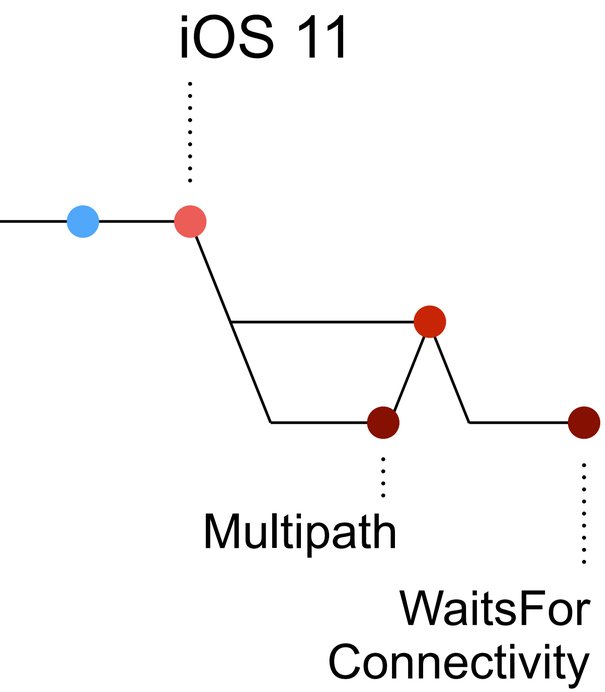
Multipath + WiFi Assist

User

- Сотовая связь не используется, если приложение в бэкграунде
- Wi-Fi Assist ограничивает количество данных, которые можно слать через сотовую сеть
- Для разработки можно отключить Wi-Fi Assist в Developer Settings



Timeline



WaitsForConnectivity

Зачем нужен

- позволяет дождаться, пока появится сеть
- эффективнее, чем дёргать Reachability

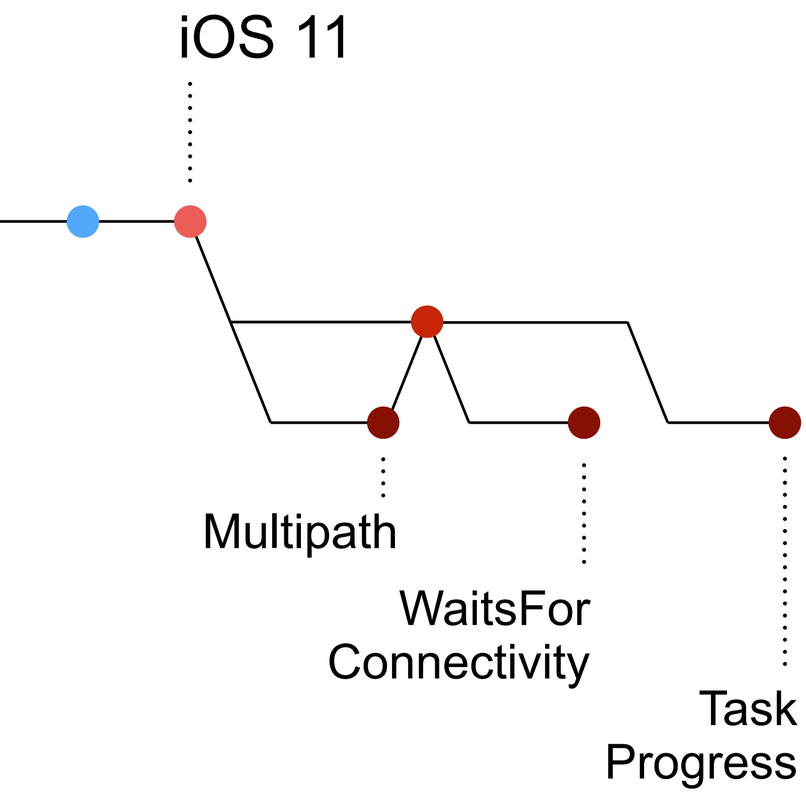
Ограничения

- игнорируется бэкграунд тасками
- ждёт только до установки соединения

WaitsForConnectivity

```swift
let configuration = URLSessionConfiguration.default
configuration.waitsForConnectivity = true
configuration.timeoutIntervalForResource = 300

let session = URLSession(configuration: configuration, delegate:
self, delegateQueue: nil)

// MARK :- NSURLSessionTaskDelegate

func urlSession(_ session: URLSession,
taskIsWaitingForConnectivity task: URLSessionTask) {
  // ждём сеть, можно обновить UI и т.п.
}
```

Timeline



URLSessionTask Progress

Нельзя ускорить — покажи, как идёт прогресс

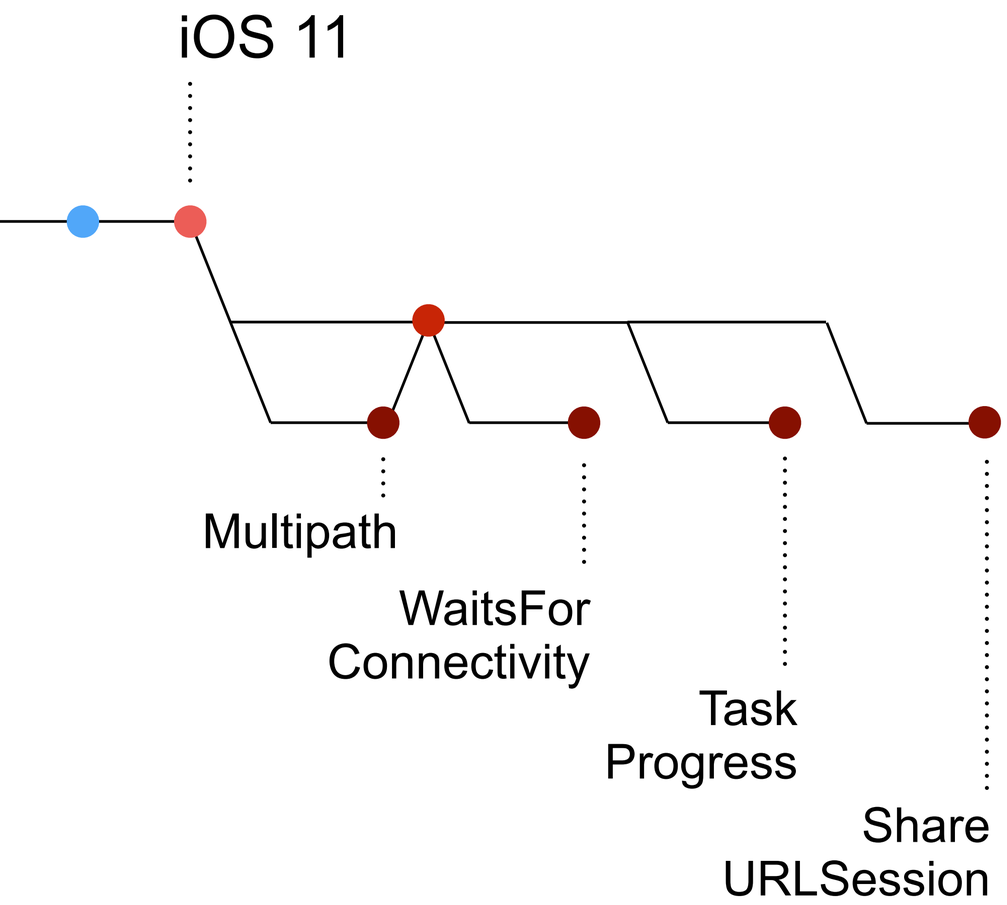
```
// MARK :- NSURLSessionTaskDelegate
func urlSession(_ session: URLSession, task: URLSessionTask,
didSendBodyData bytesSent: Int64, totalBytesSent: Int64,
totalBytesExpectedToSend: Int64) {
    let progress = Double(totalBytesSent / totalBytesExpectedToSend)
    DispatchQueue.main.async {
        self.progressView.progress = progress
    }
}

func urlSession(_ session: URLSession, downloadTask:
URLSessionDownloadTask, didWriteData bytesWritten: Int64,
totalBytesWritten: Int64, totalBytesExpectedToWrite: Int64) {
    // обновить UI
}
```

URLSessionTask Progress

Нельзя ускорить — покажи, как идёт прогресс

```
let session = URLSession(configuration: .default)
let task = session.dataTask(with: serverURL)

let observation = task.progress
                      .observe(\.fractionCompleted) { progress, _ in
    self.progressView.progress = progress.fractionCompleted
}

task.resume()
```

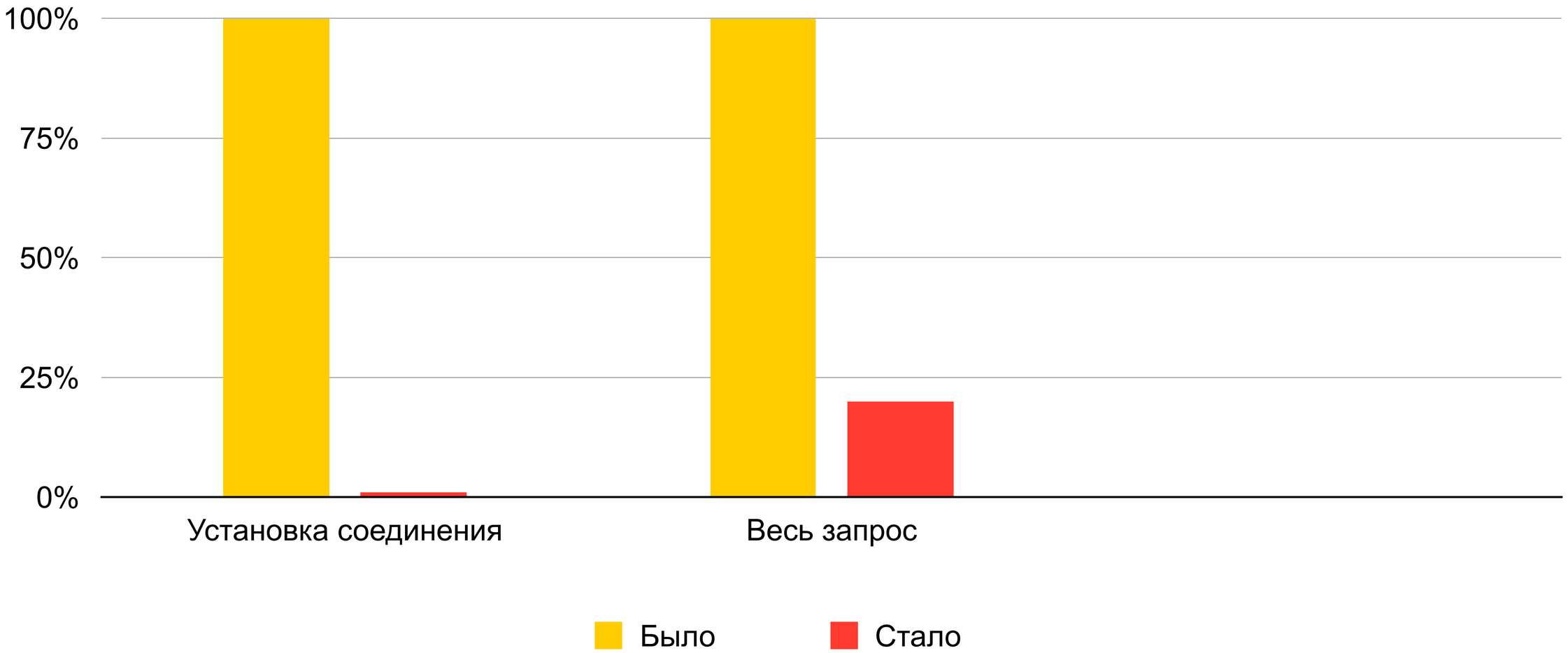
Timeline


iOS 11
Multipath
WaitsFor
Connectivity
Task
Progress
Share
URLSession

Общая URLSession

```
let session = URLSession(configuration: .default)

let webSocketTask = session.webSocketTask(with: wccServerURL)
let dataTask = session.dataTask(with: serverURL)
let uploadTask = session.uploadTask(with: request, from: data)
```

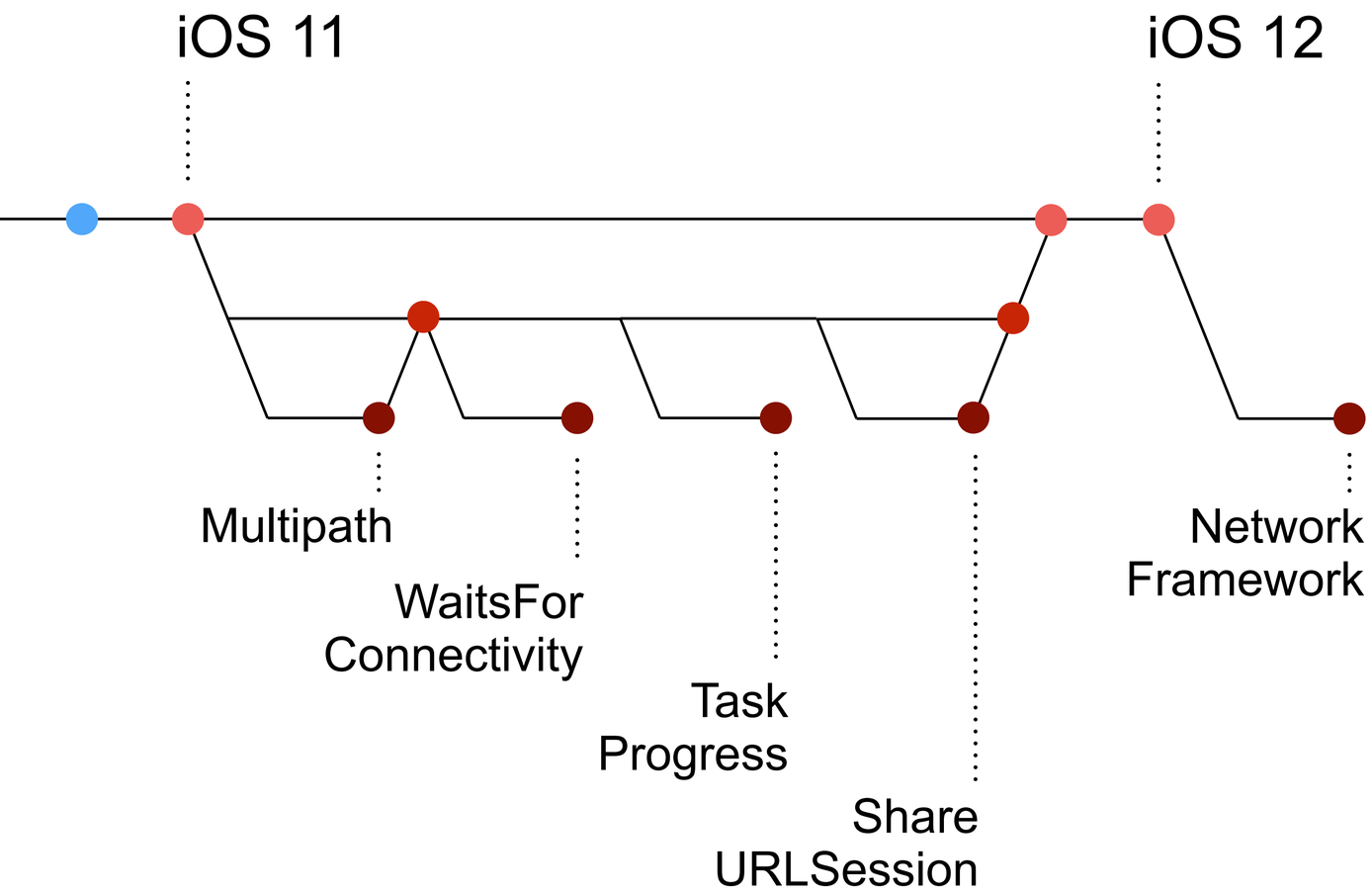
Что дало переиспользование URLSession



Проблемы, которые нашли

- ~~Делаем лишние запросы для «тонких» тредов~~
- ~~Не сжимает аттачи при отправлении письма~~
- ~~Загружаем аватарки по одной~~
- ~~На каждый запрос долгая установка соединения~~

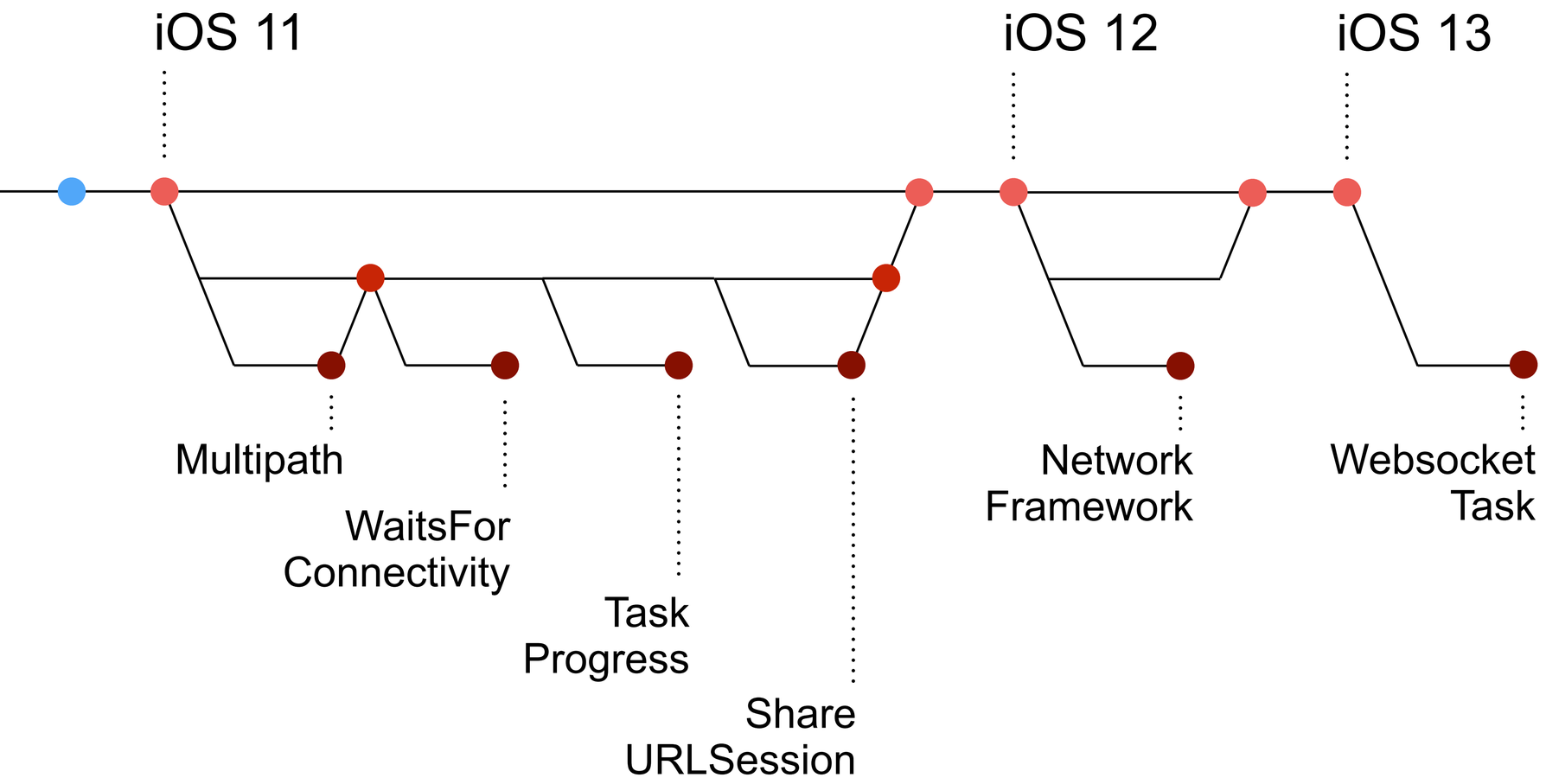
Timeline


iOS 11
iOS 12
Multipath
WaitsFor Connectivity
Task Progress
Share URLSession
Network Framework

NetworkFramework

```
// TLS, TCP и UDP, для остального – URLSession

import Network

let connection = NWConnection(host: “example.com", port: .imaps, using: .tls)
connection.stateUpdateHandler = { (newState) in
    switch(newState) {
    case .ready:
        // Handle connection established
    case .waiting(let error):
        // Handle connection waiting for network
    case .failed(let error):
        // Handle fatal connection error
    default:
        break
   }
}
connection.start(queue: myDispatchQueue)
```

Timeline


iOS 11
iOS 12
iOS 13
Multipath
WaitsFor Connectivity
Task Progress
Share URLSession
Network Framework
Websocket Task

WebsocketTask

RFC 6455

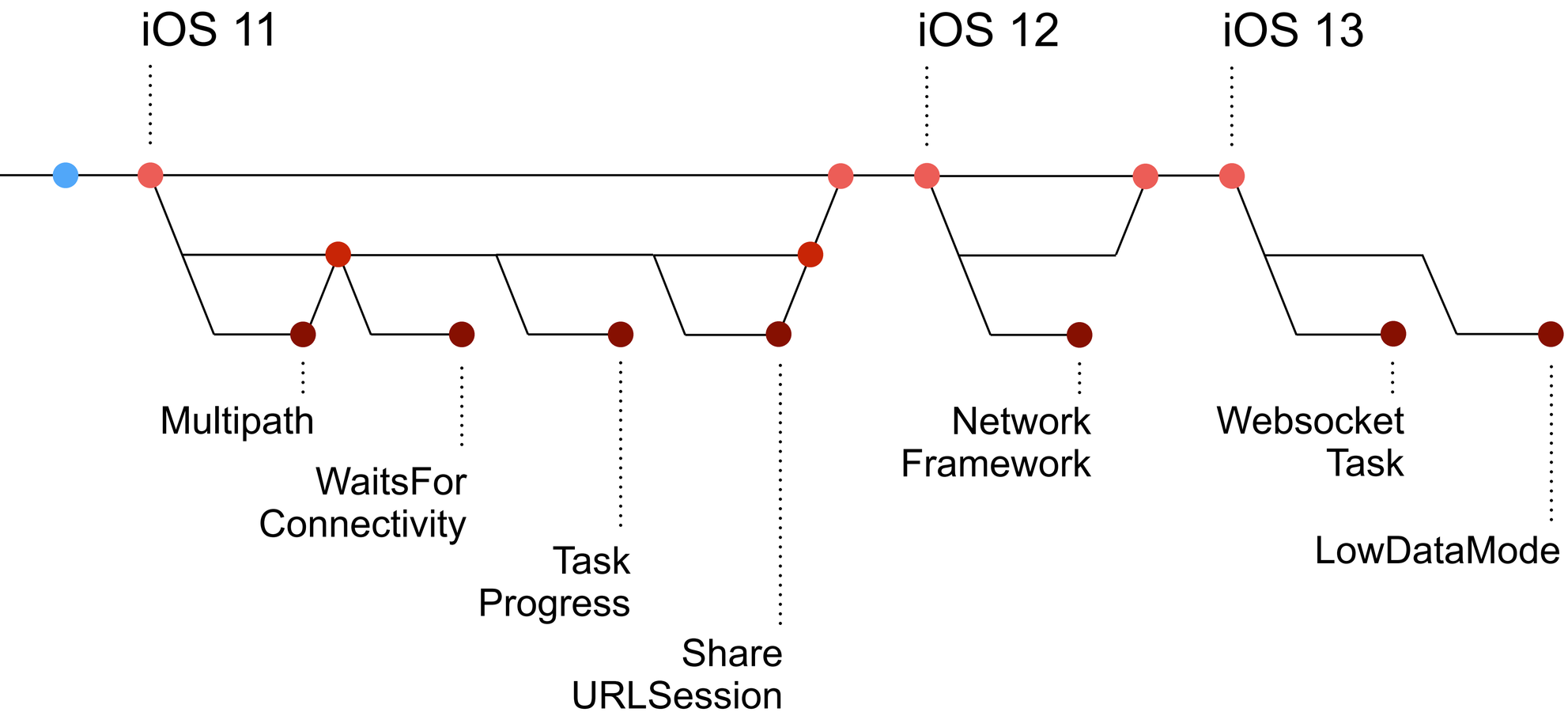
```
let session = URLSession(configuration: .default)
let webSocketTask = session.webSocketTask(with: serverURL)
webSocketTask.resume()

let message = URLSessionWebSocketTask
            .Message.string("Hello World”)

webSocketTask.send(message) { error in
    if let error = error {
        print("Unable to send message: \(error)")
    }
}
```

WebsocketTask

RFC 6455

```
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)")
        }
    }
}
```

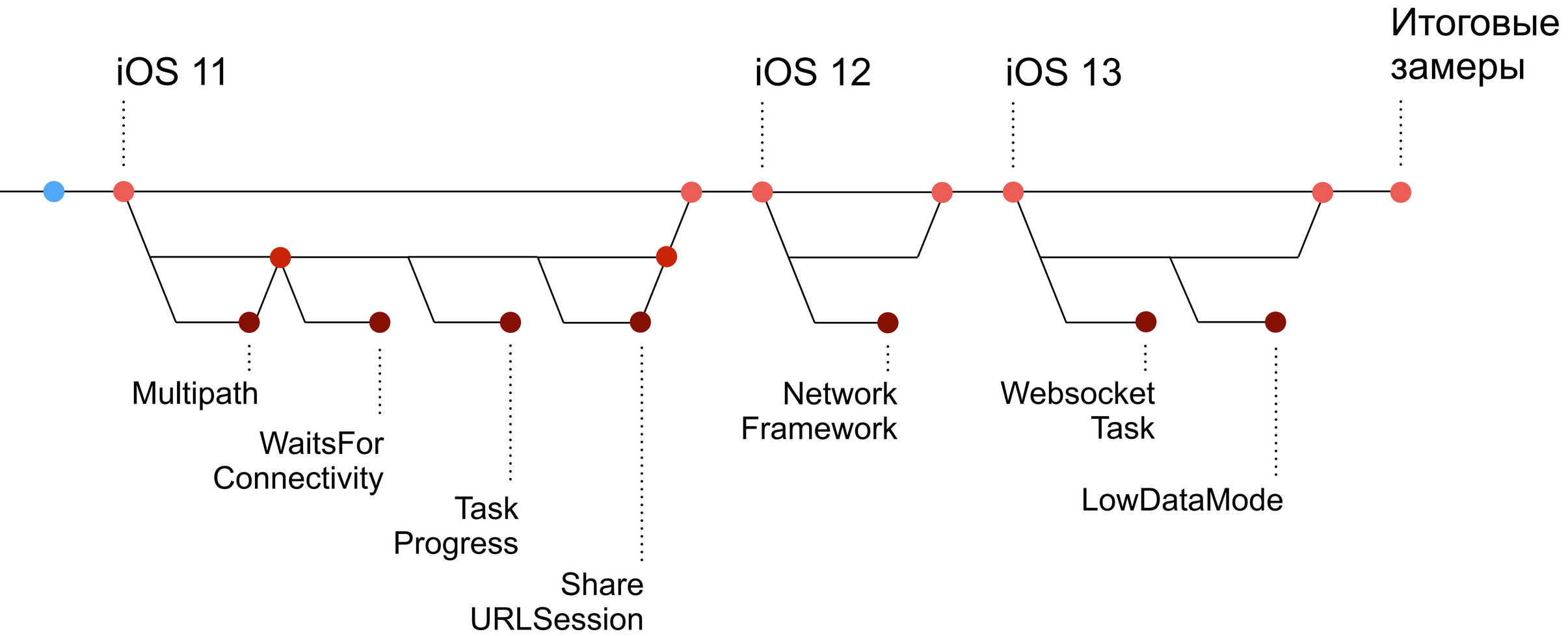
Timeline


iOS 11
iOS 12
iOS 13
Multipath
WaitsFor Connectivity
Task Progress
Share URLSession
Network Framework
Websocket Task
LowDataMode

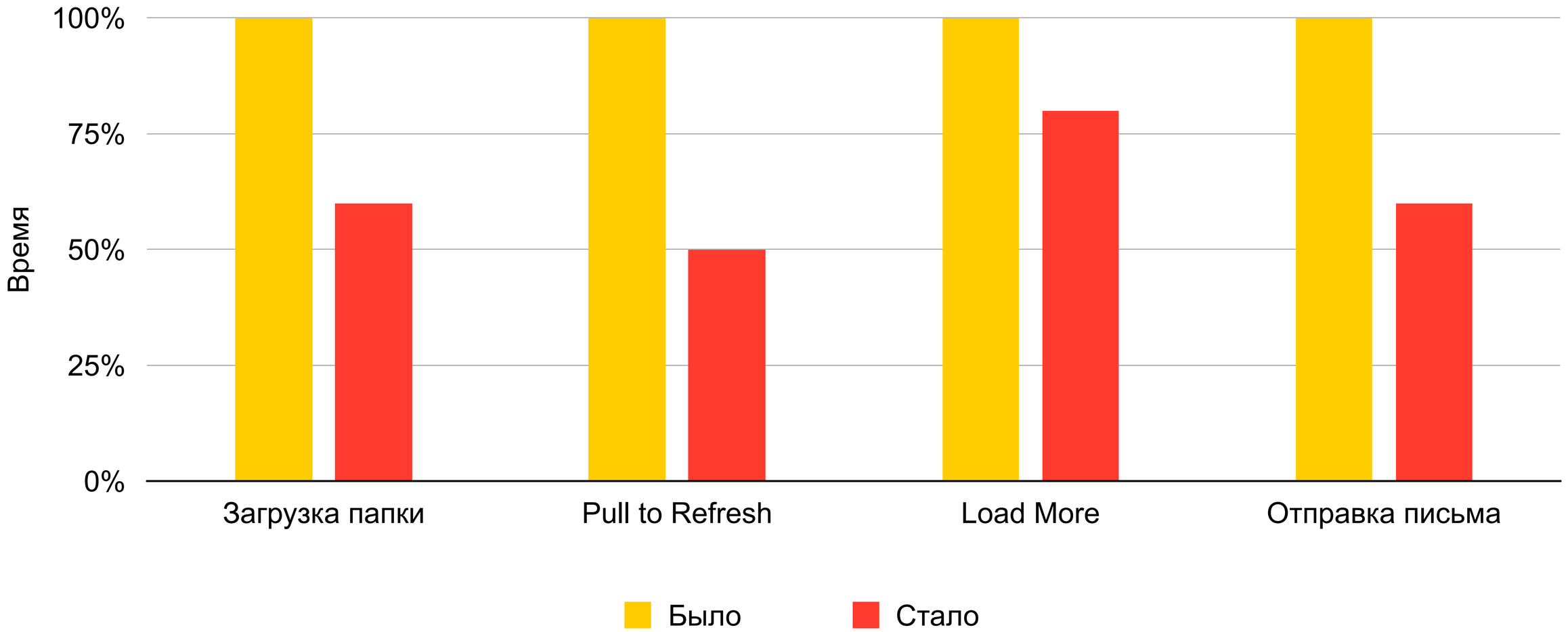
LowDataMode


```
var configuration = URLSessionConfiguration.default
configuration.allowsConstrainedNetworkAccess = false
let session = URLSession(configuration: configuration)

session.dataTask(with: request) { data, response, error in
    if let error = error {
        if let networkError = error as? URLError,
        networkError.networkUnavailableReason == .constrained {
            // можно запросить меньше данных
            // или менее тяжелые данные
        }
        return
    }
}
```

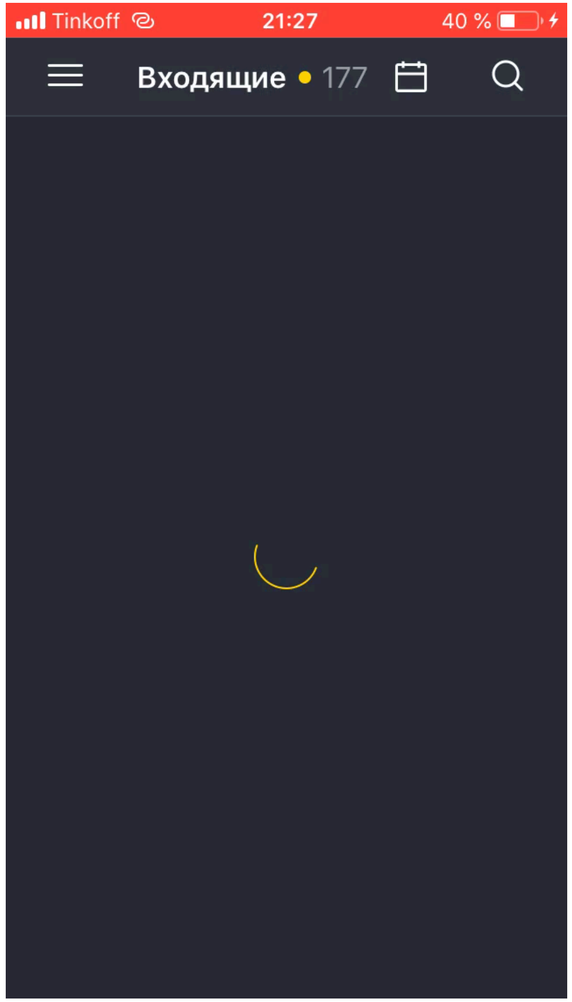
Timeline

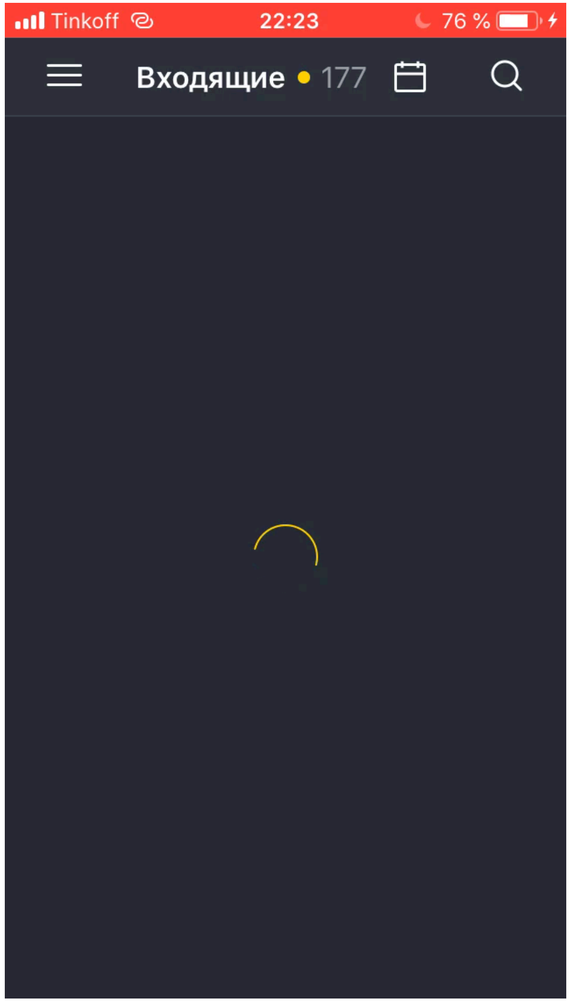

iOS 11
iOS 12
iOS 13
Итоговые замеры
Multipath
WaitsFor Connectivity
Task Progress
Share URLSession
Network Framework
Websocket Task
LowDataMode

Итоги


100%
75%
50%
25%
0%
Время
Загрузка папки
Pull to Refresh
Load More
Отправка письма
Было
Стало

Субъективное профилирование





Что нам помогло

1. Профилирование + логирование + автотесты
2. Убрали лишние запросы
3. Объединили несколько запросов в один
4. Добавили сжатие аттачей
5. Используем повторно URLSession
6. Не углублялись в дебри, чтобы получить профит

Яндекс

Спасибо

Ася Свириденко

Руководитель iOS-группы Яндекс.Почты, Санкт-Петербург

a-kononova@yandex-team.ru