Spring
Reactive Or Not Reactive

...

1984

@tolkv

@lavcraft

@jekaborisov

@jeka1978

ЦИАН

alfalab

РАЗБОР ПОЛЁТОВ

#RAZBORPOLETOV

ПОДКАСТ

ДОБРЫЙ ДЕНЬ ДОБРЫЙ ВЕЧЕР ДОБРОЕ УТРО ДОБРОЙ НОЧИ

NAYA TECHNOLOGIES

alfalab

Spring Reactive Or Not Reactive

Задача

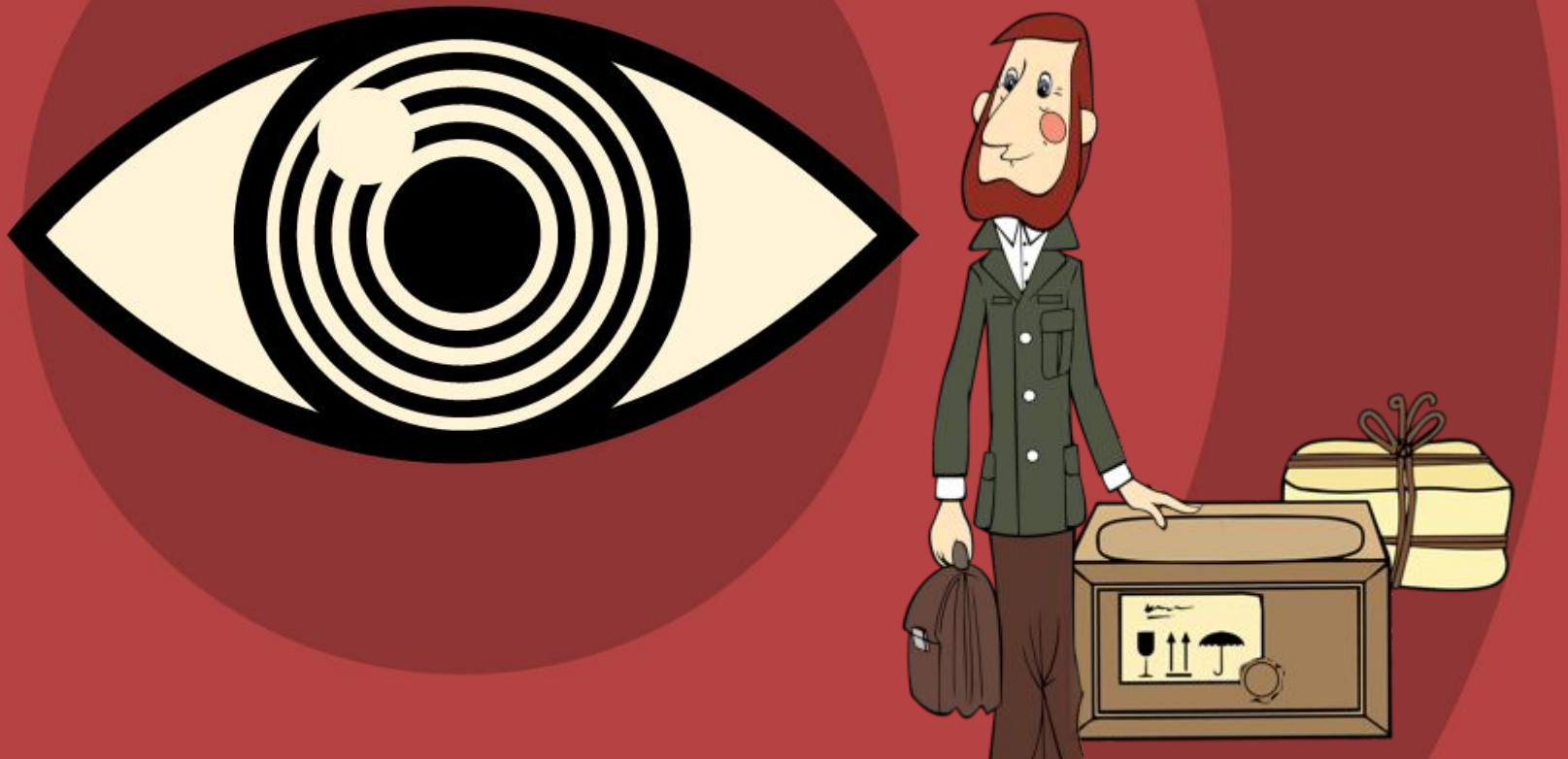
Задача

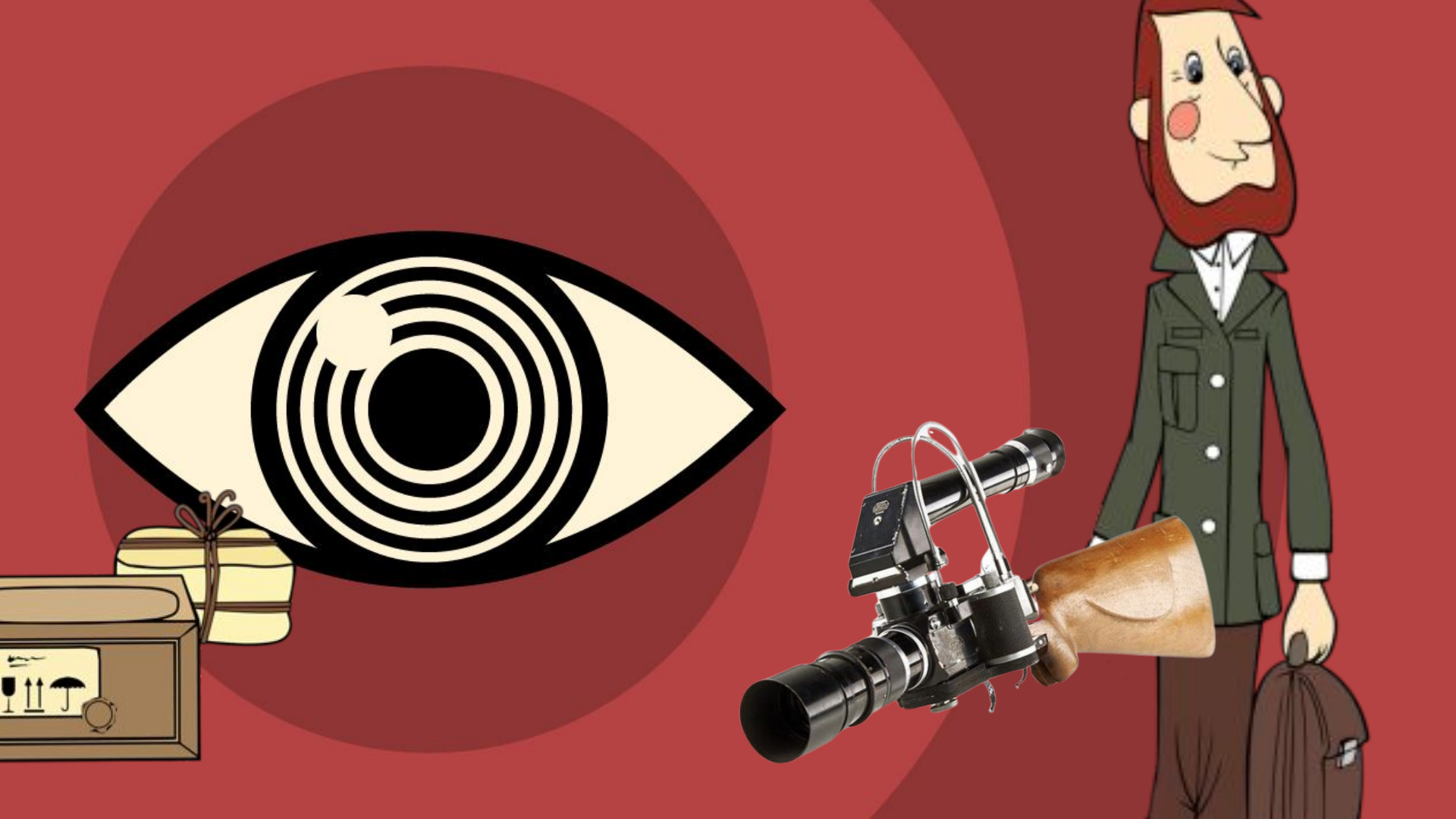
Задача

Задача

Имя: Папа
Фамилия: Папа мальчика
Должность: Академик

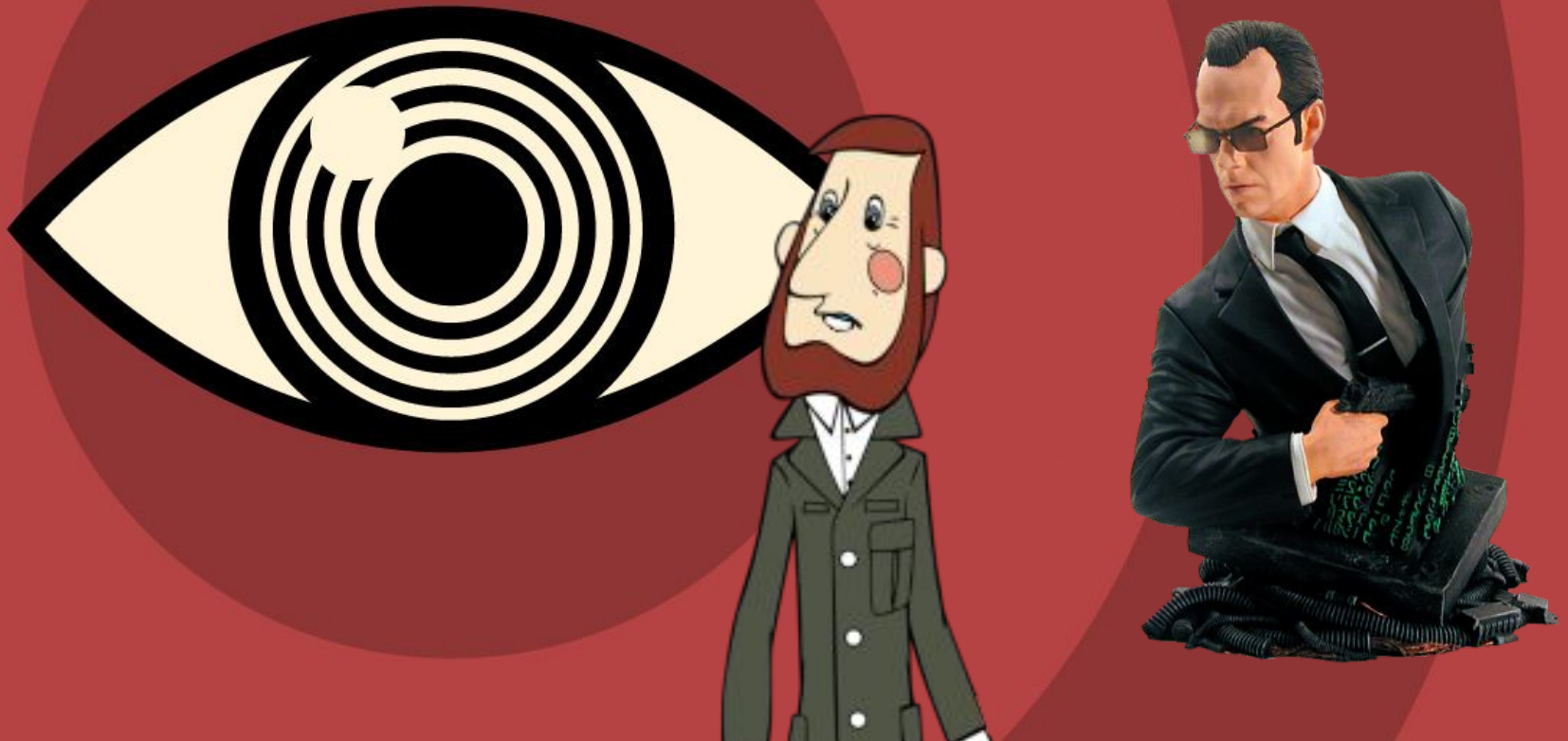
Задача

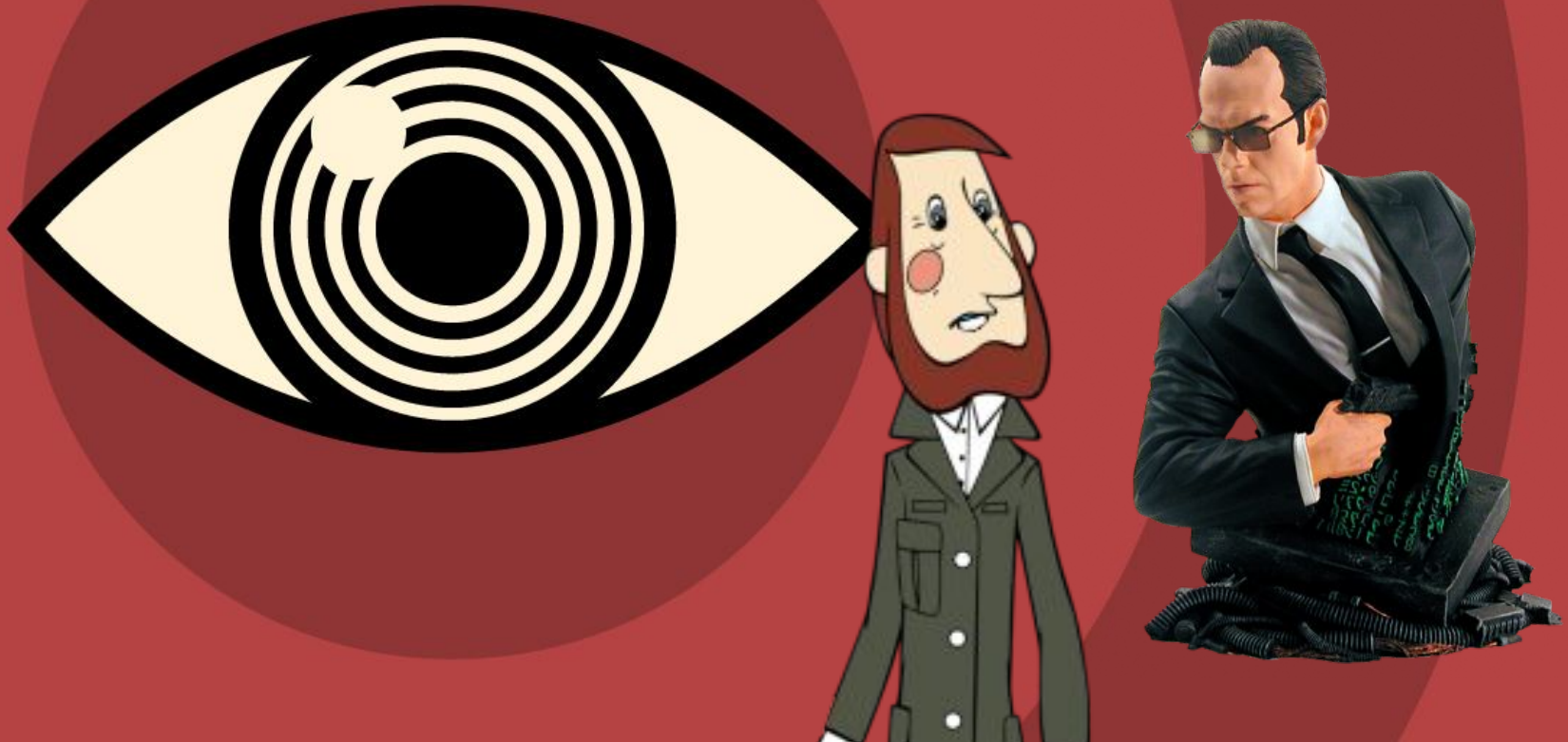


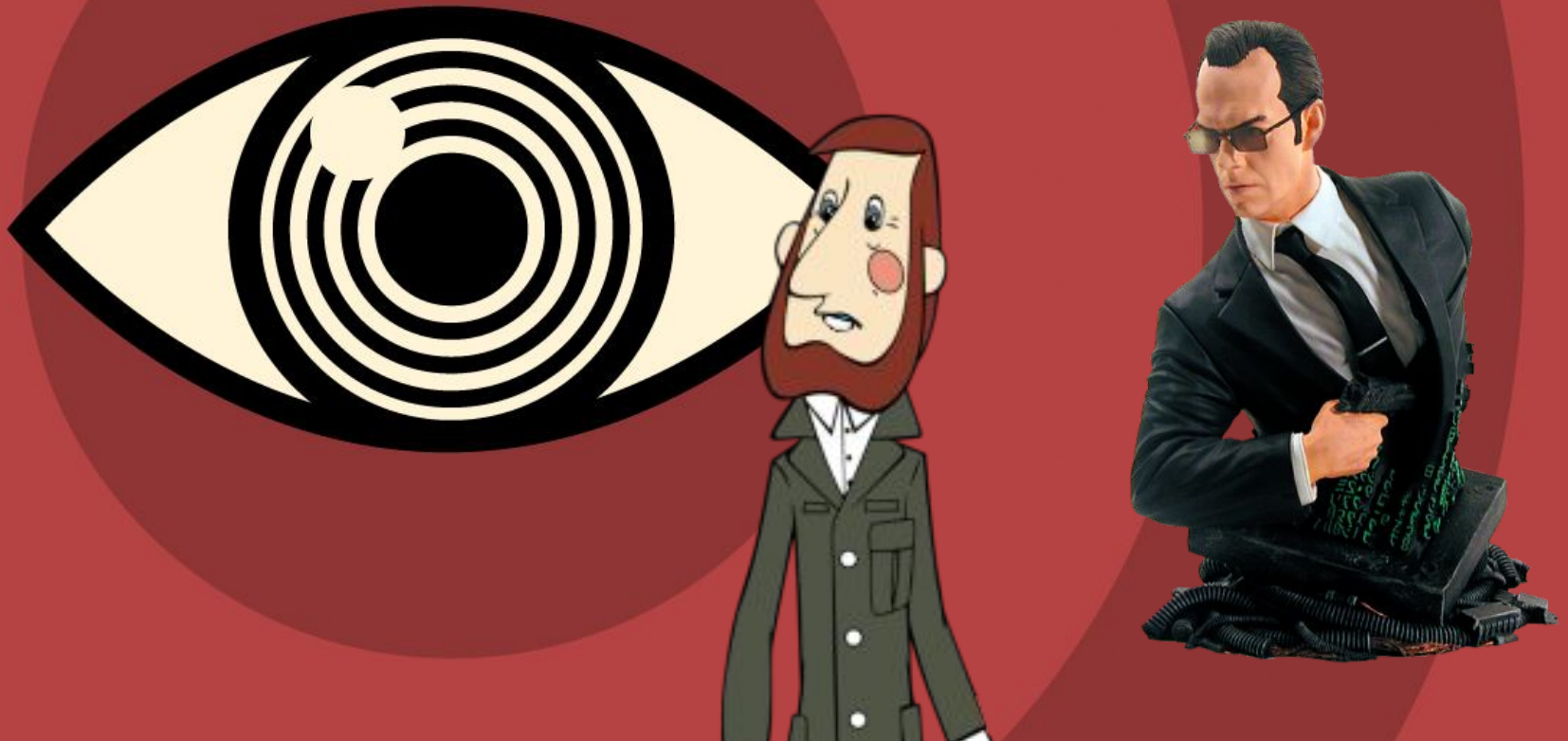


Имя: Папа
Фамилия: Папа мальчика
Должность: Академик

Задача







Архитектура проекта «Big Brother»

Перехватчик писем,
отсылает все письма на распознание

Расшифровщик писем,
Выясняет чьё письмо о отсылает
в Правоохранительные органы

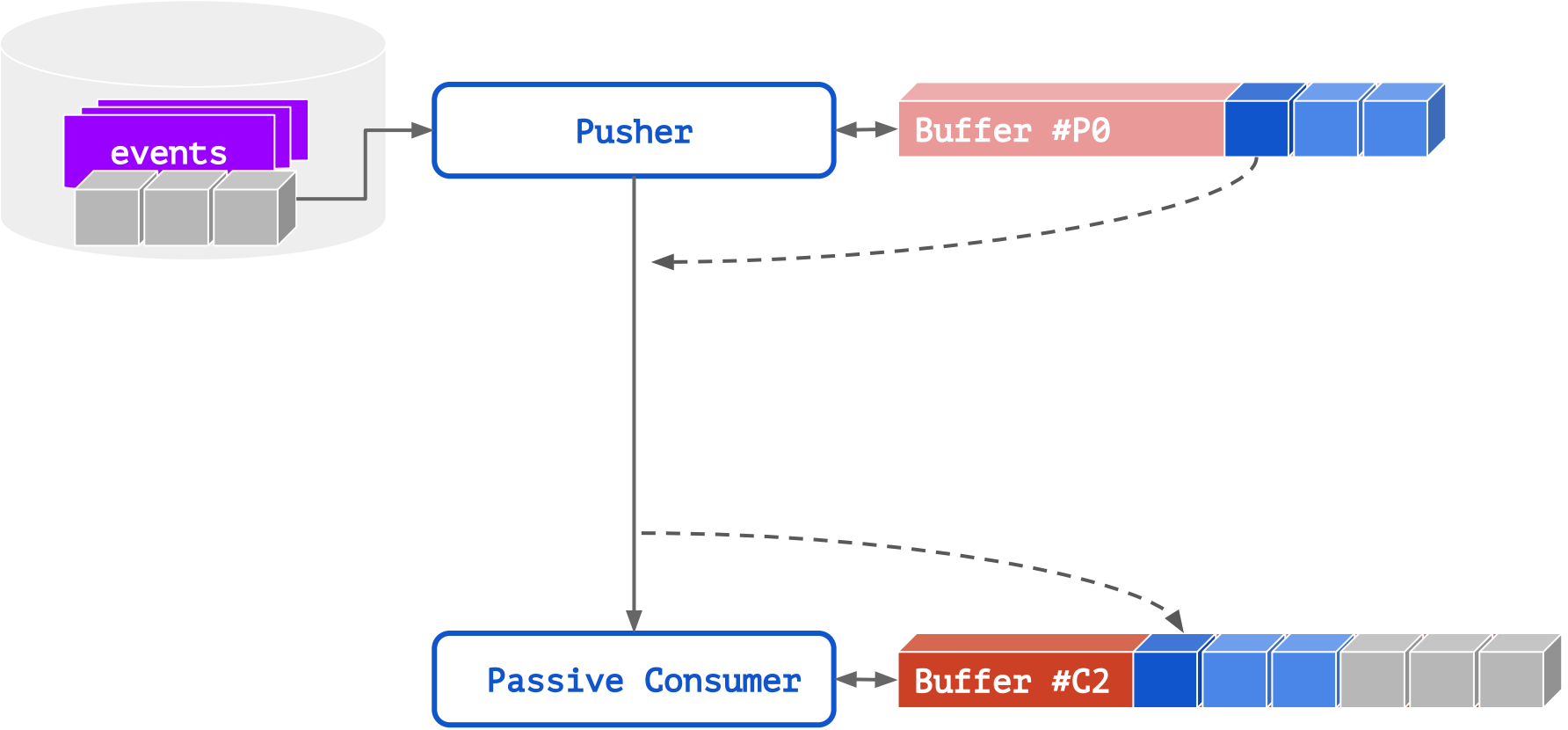
Правоохранительные органы,
принимают решение об аресте


ЗАПРЕТИТЬ! УЖЕСТОЧИТЬ!
НЕ ПУШАТЬ!

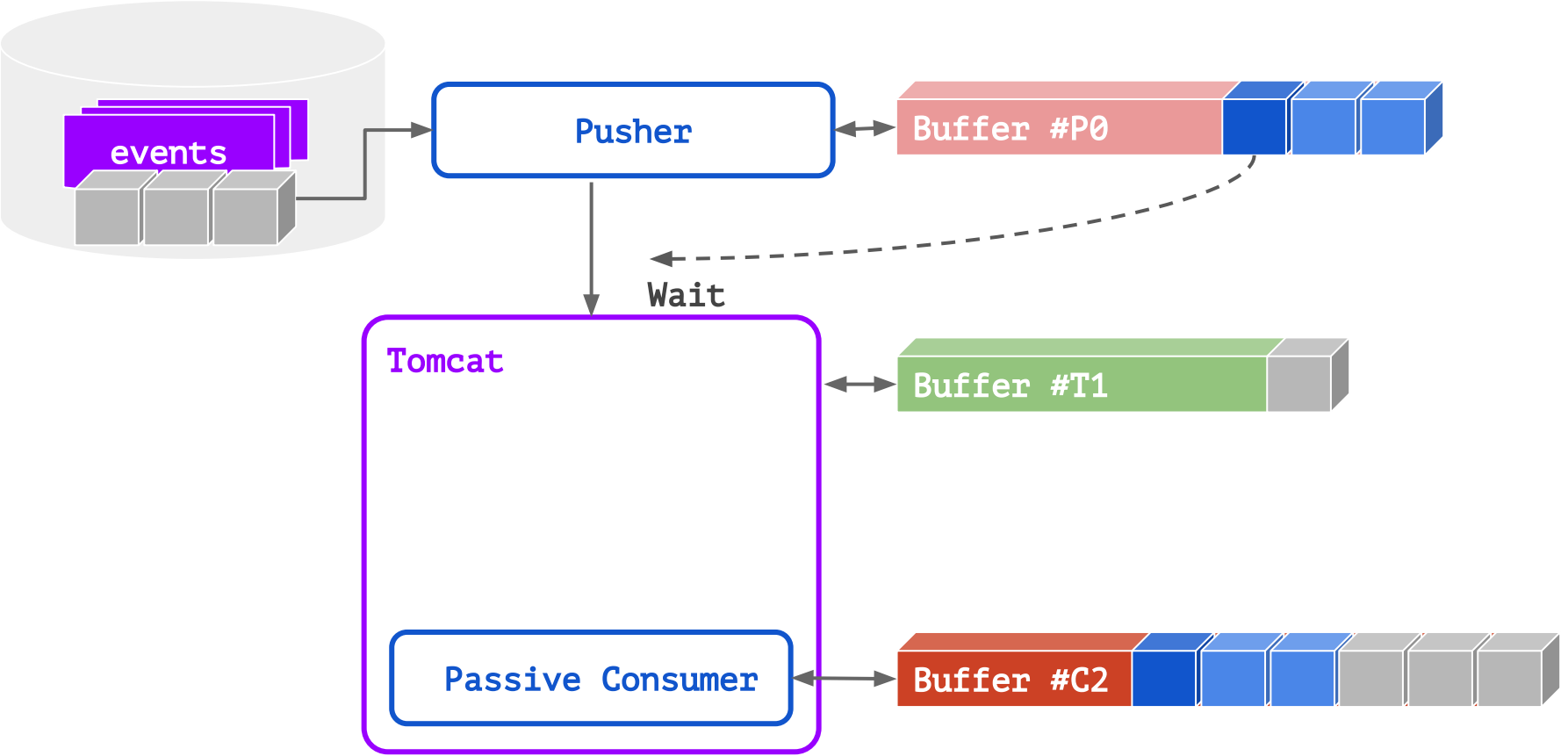
Demo

Наивная имплементация #0

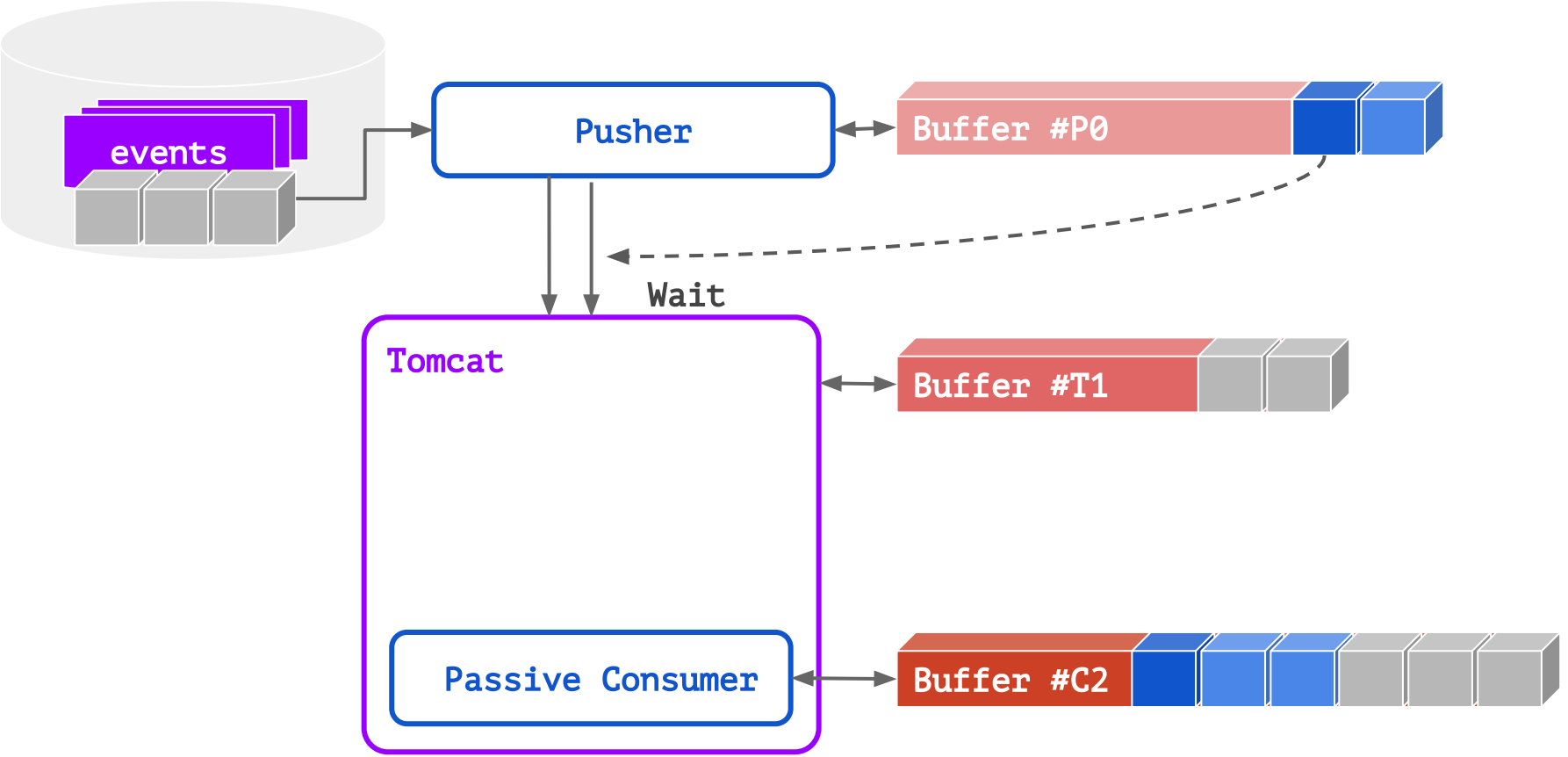
Push and Pull semantics related to consumer



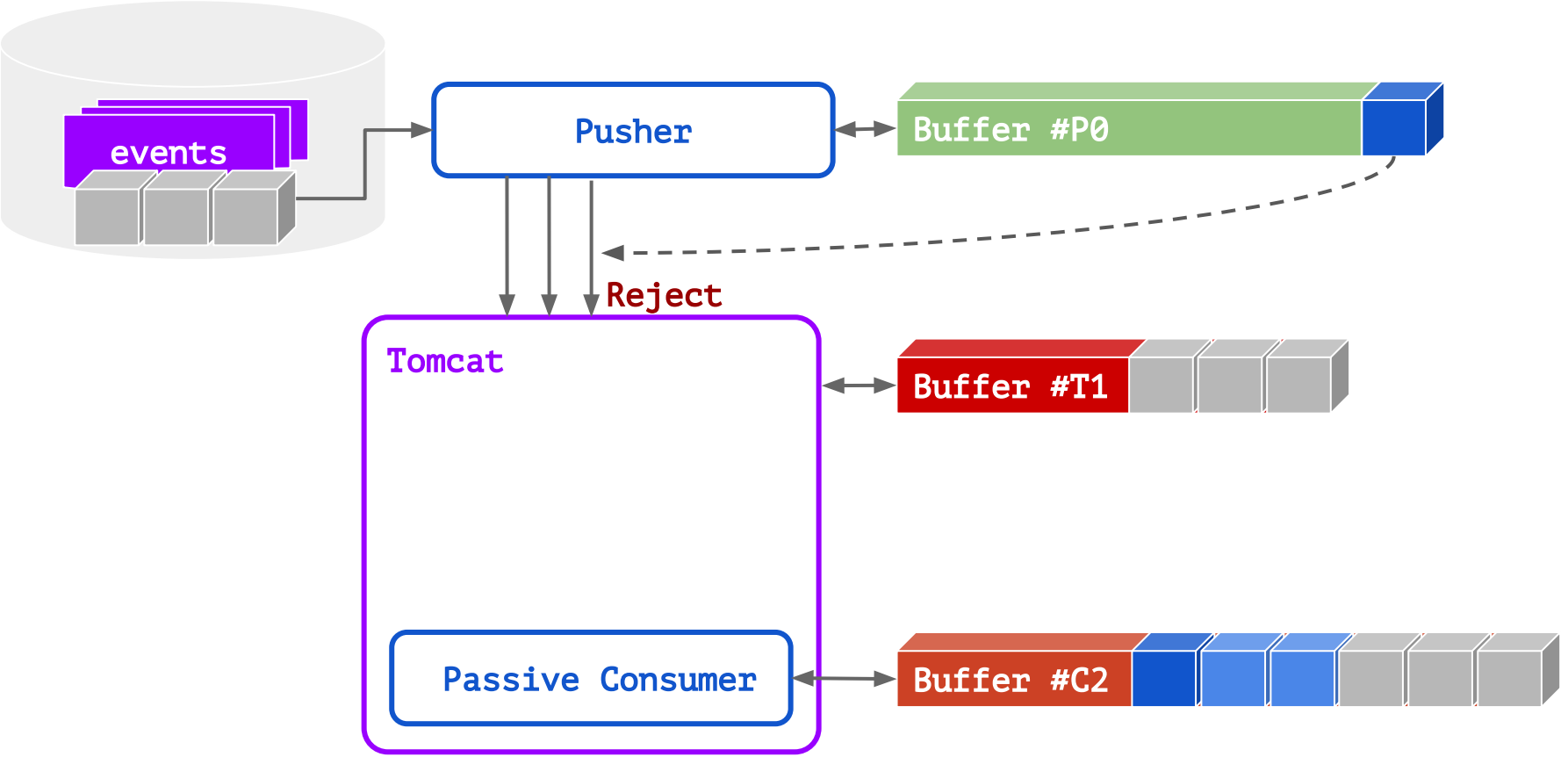
Push and Pull semantics related to consumer



Push and Pull semantics related to consumer

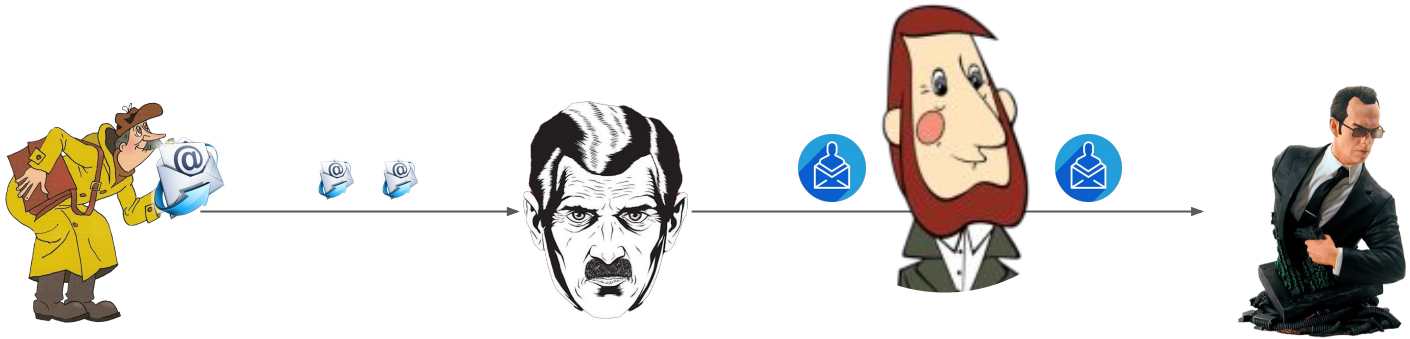


Push and Pull semantics related to consumer



Demo

Наивная имплементация #1



О пропускной способности

pechkin-service

big-brother-service

agent-smith-service

request(n)

request(n)

Buffer #P0

1t

∞

Buffer #P1

8t

8rps

Buffer #P2

2t

2rps

pechkin-service

big-brother-service

agent-smith-service

request(n)

request(n)

Buffer #P0

1t

∞

Buffer #P1

8t

8rps

Buffer #P2

2t

2rps

request(n)

pechkin-service

big-brother-service

agent-smith-service

request(n)

Buffer #P0

1t

∞

Buffer #P1

8t

8rps

Buffer #P2

2t

2rps

Задача

Оптимизировать количество запросов под ресурсы в цепочке «Backpressure»

Demo

Manual backpressure

pechkin-service

big-brother-service

agent-smith-service

request(n)

request(n)

Buffer #P0

1t

∞

Buffer #P1

8t

8rps

Buffer #P2

2t

2rps

Варианты получать данные

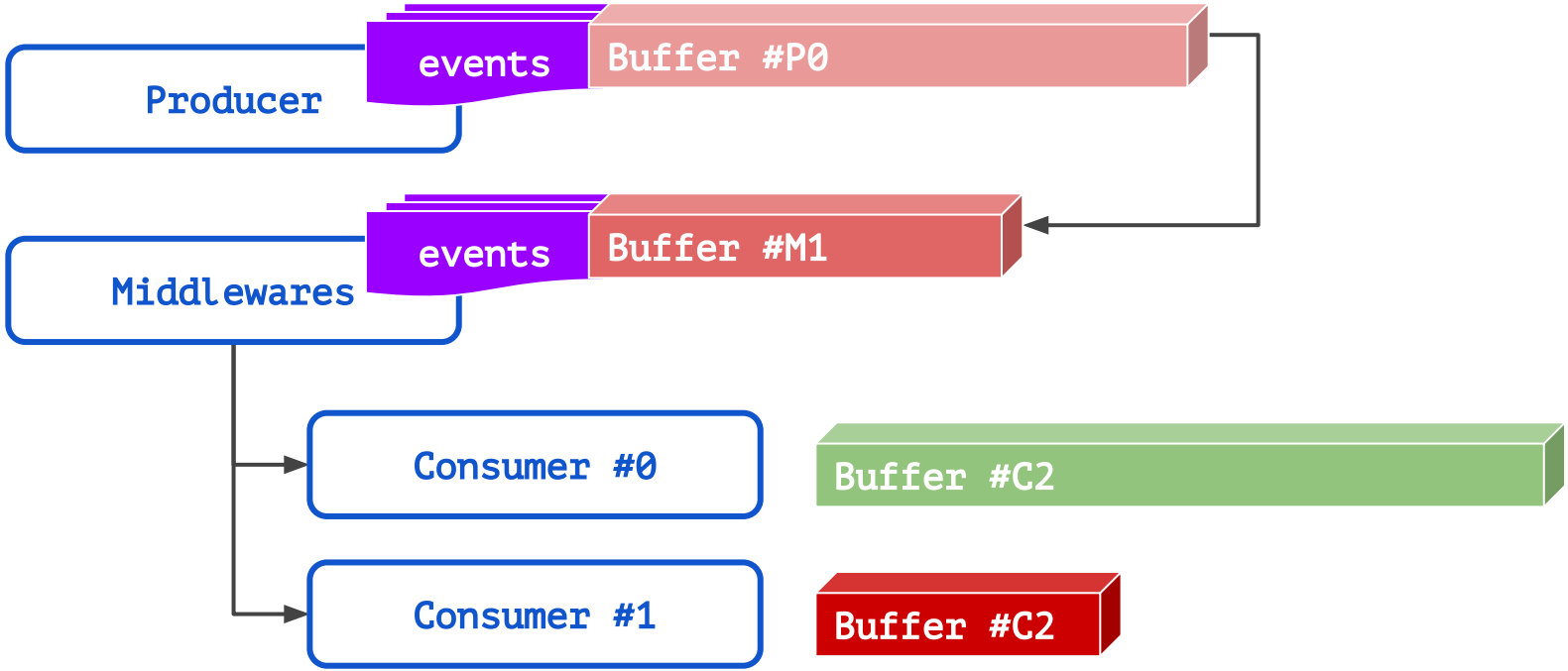
Что должен делать producer когда consumer не справляется

Продюсер читает данные с нужной скоростью

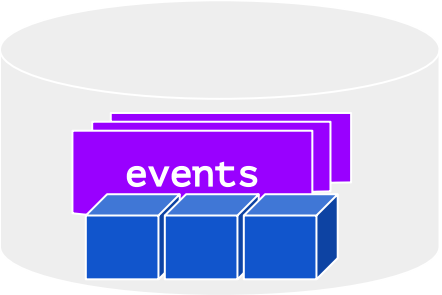
Нет проблемы

В продюсер текут данные

Всё что не успеваем отослать, суем в очередь, когда нет места в очереди, дропаем, то что менее важно


Producer
events
Buffer #P0
Middlewares
events
Buffer #M1
Consumer #0
Buffer #C2
Consumer #1
Buffer #C2

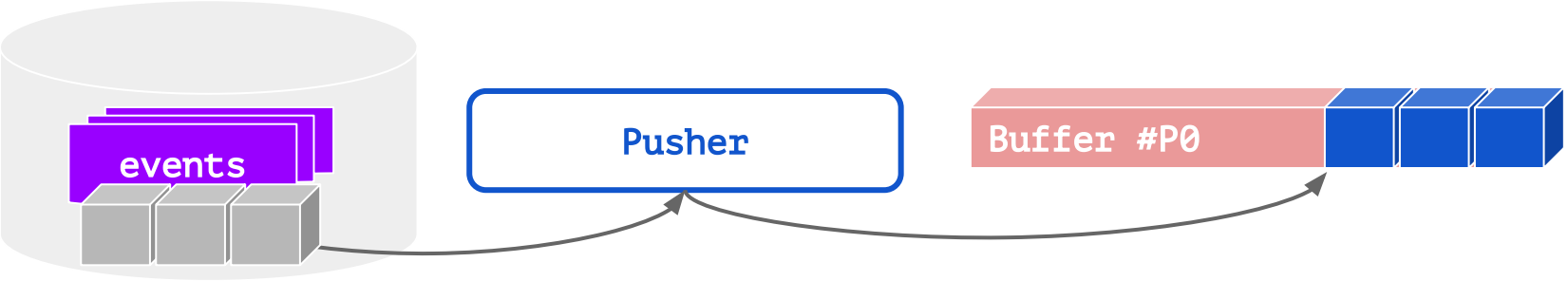
Push and Pull semantics related to consumer



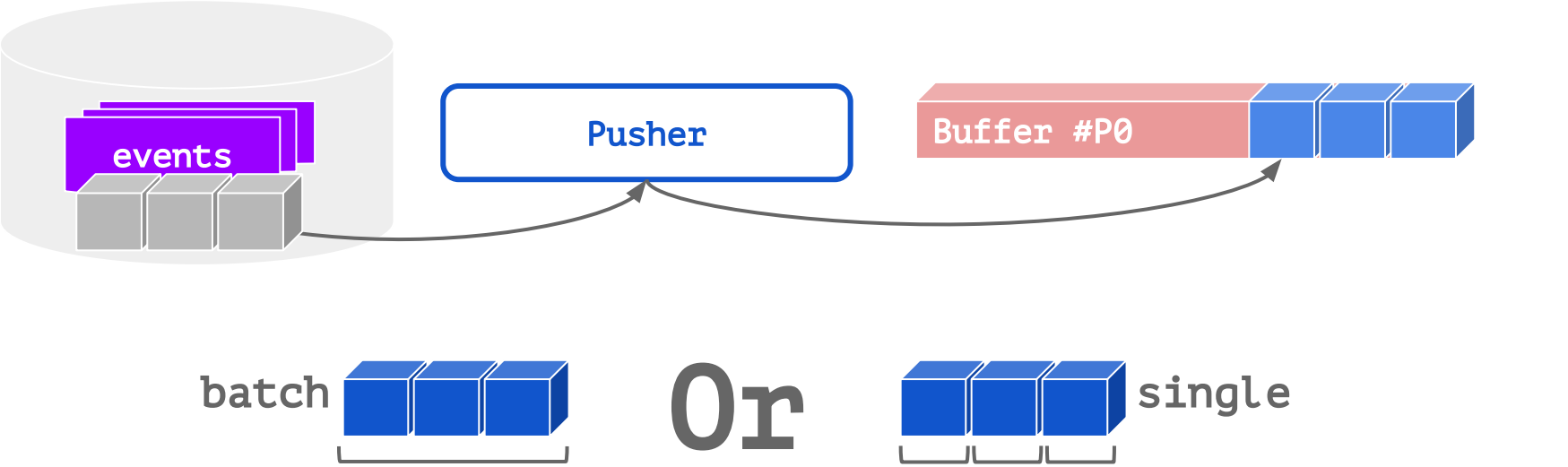
Pusher

Buffer #P0

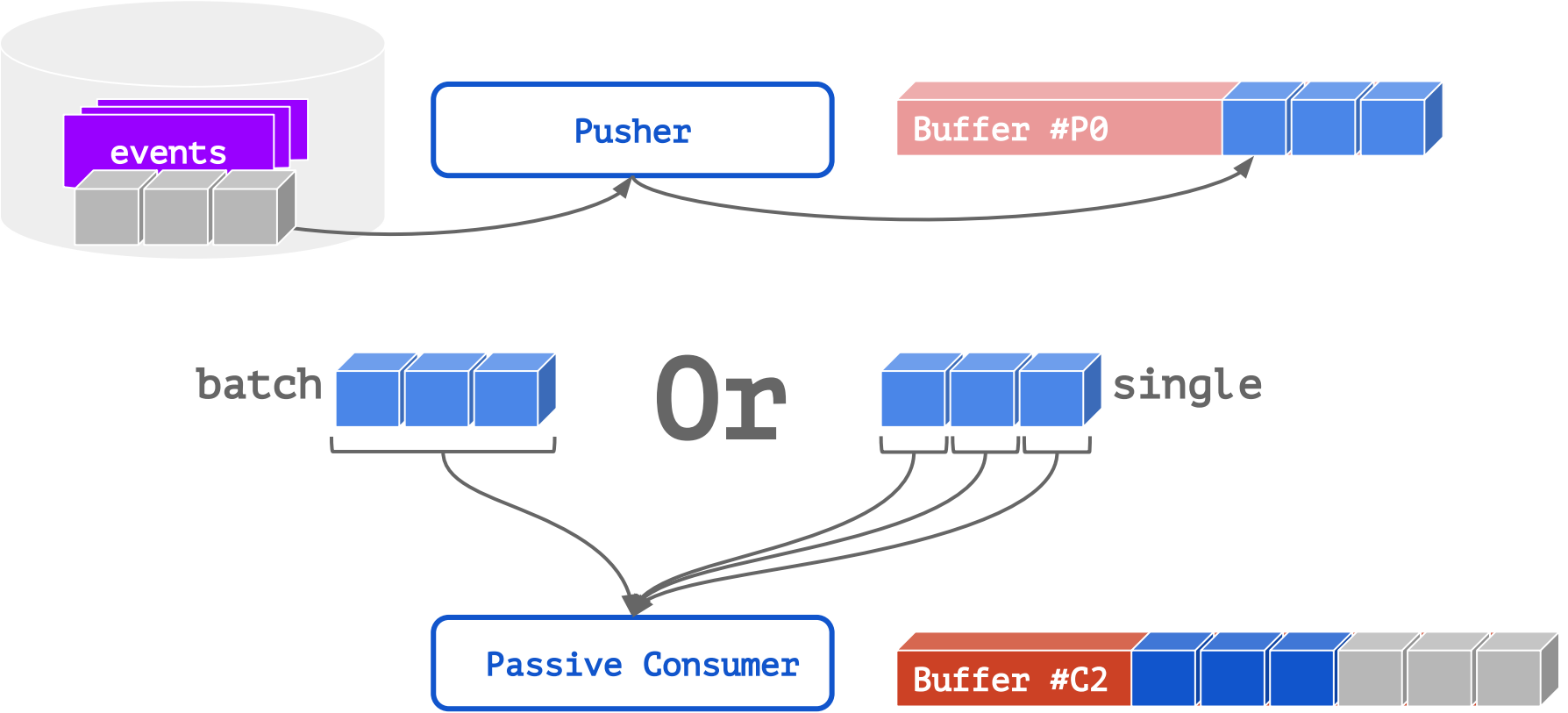
Push and Pull semantics related to consumer



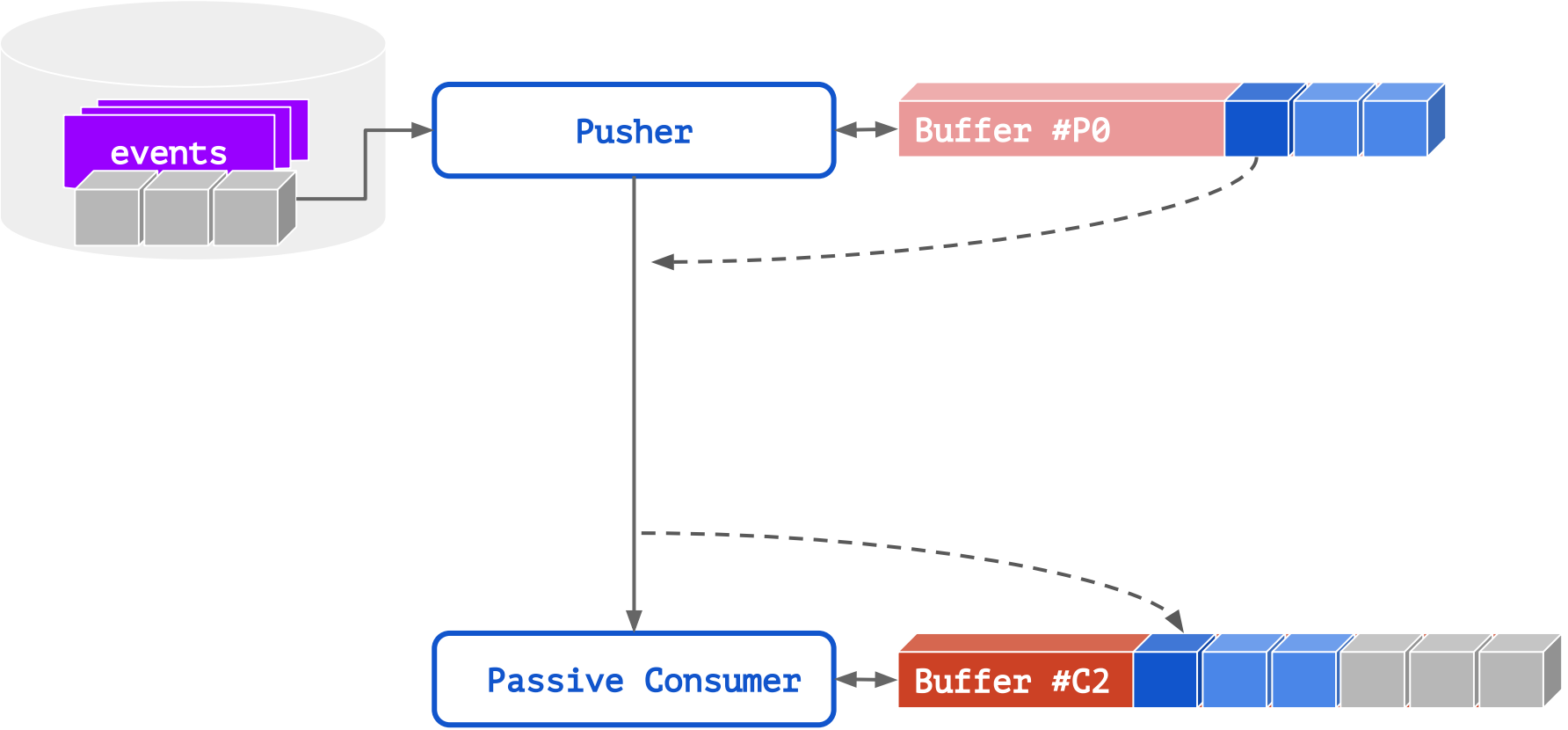
Push and Pull semantics related to consumer



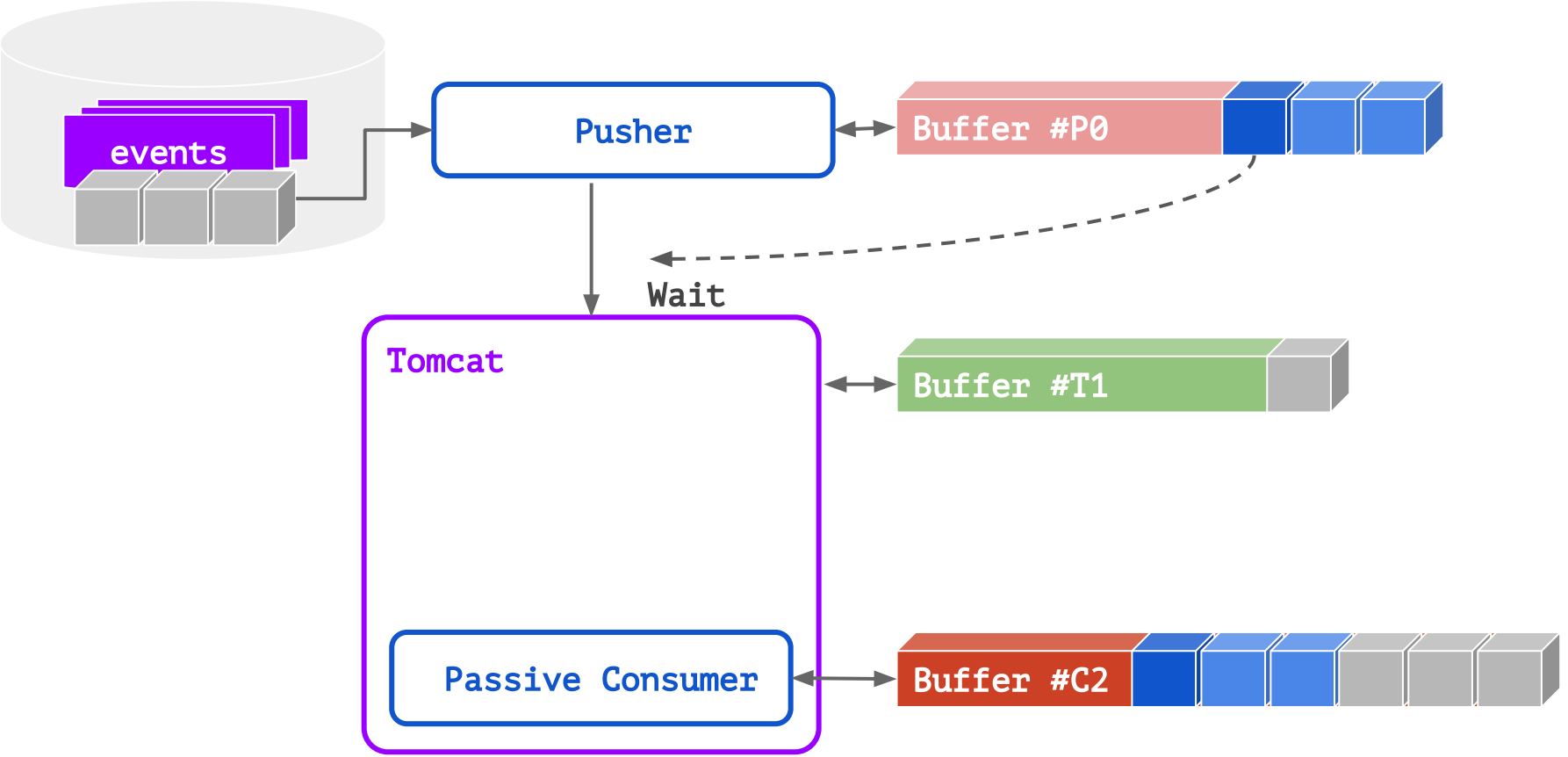
Push and Pull semantics related to consumer



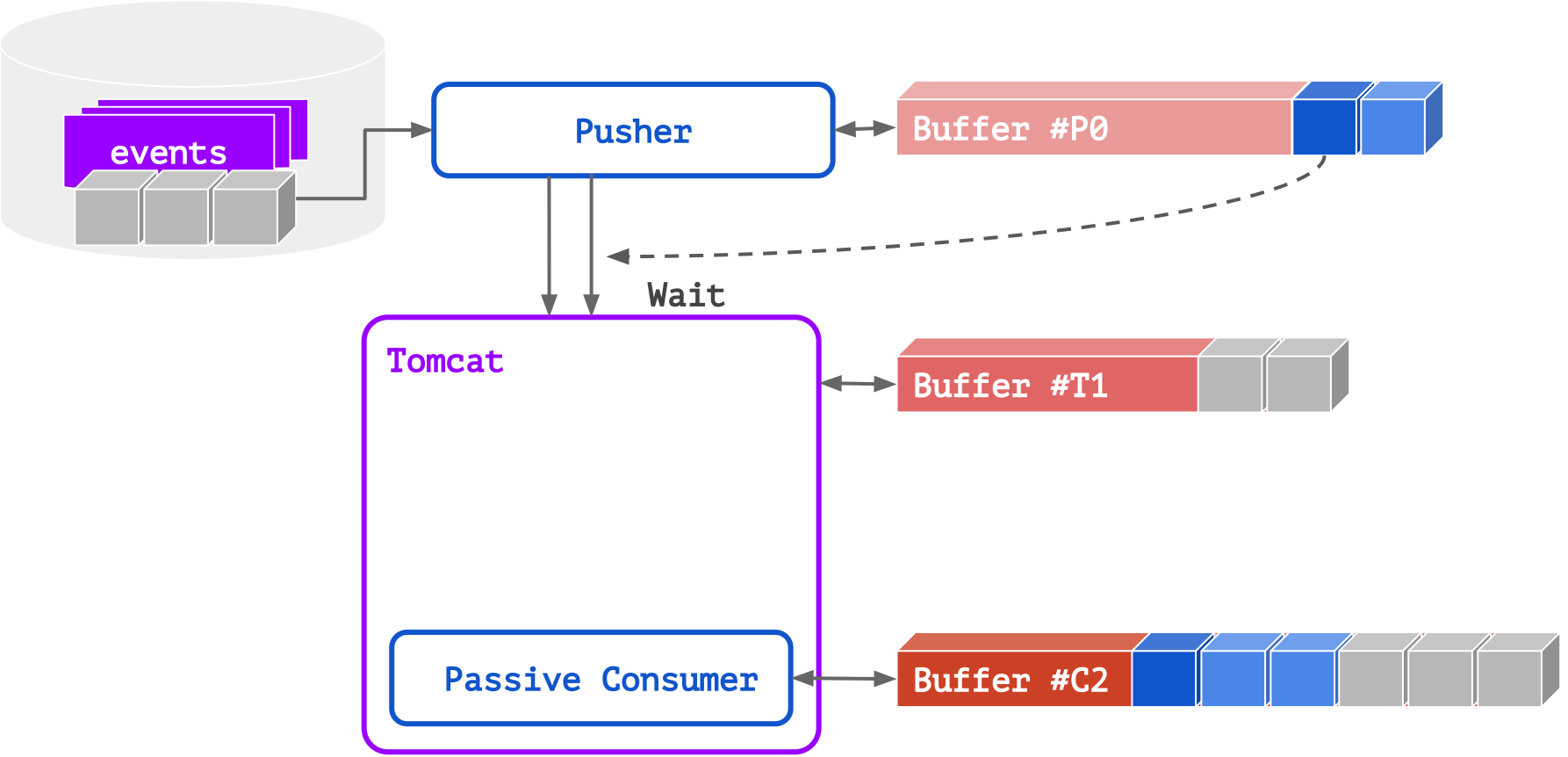
Push and Pull semantics related to consumer



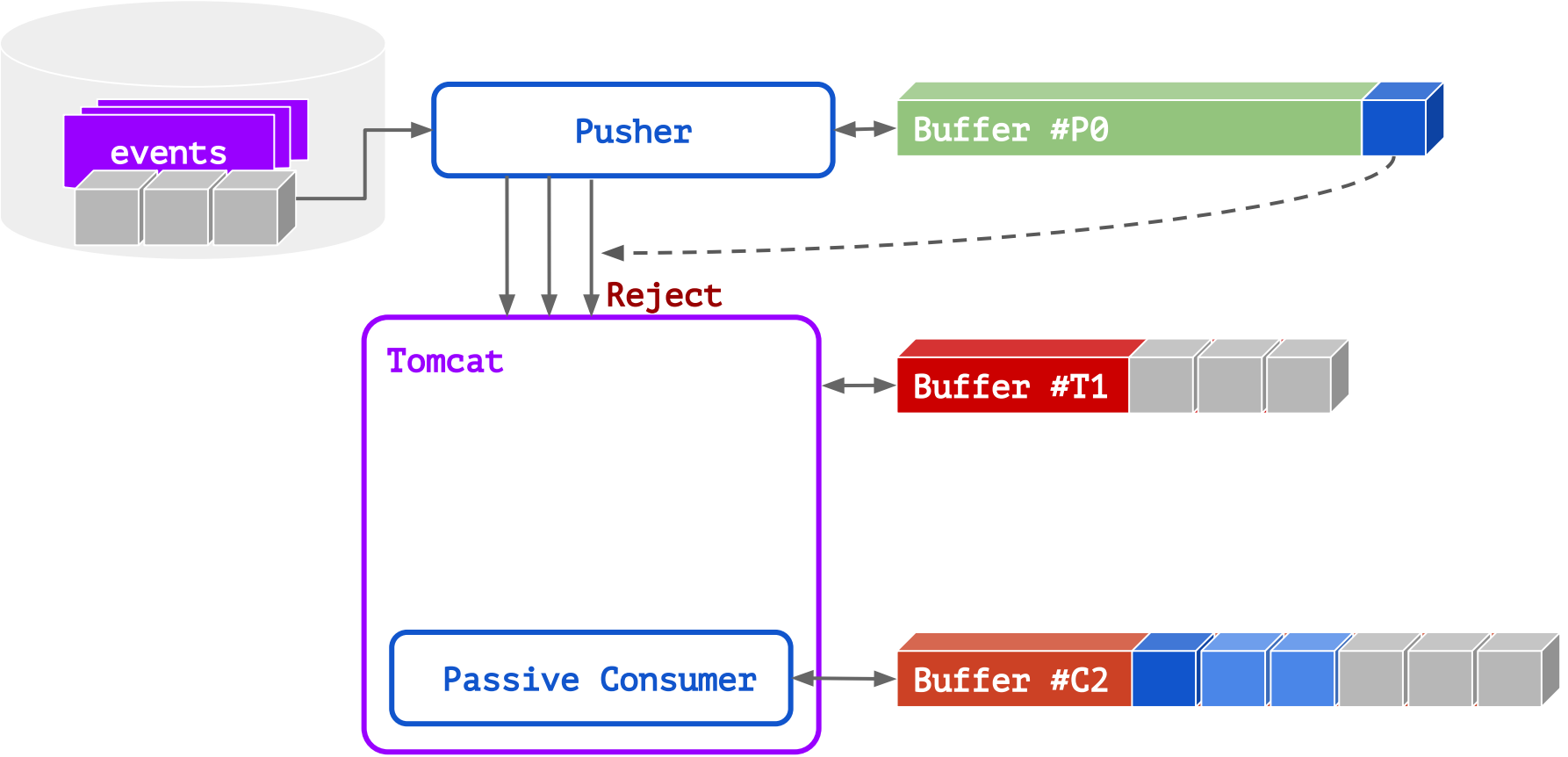
Push and Pull semantics related to consumer



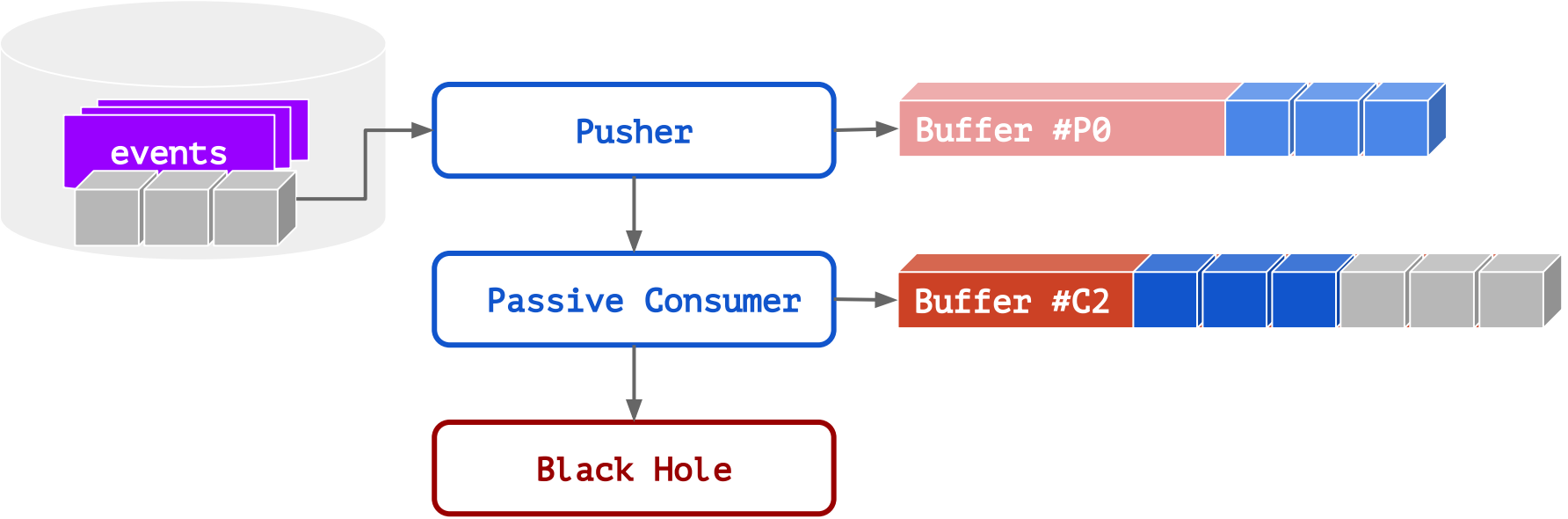
Push and Pull semantics related to consumer



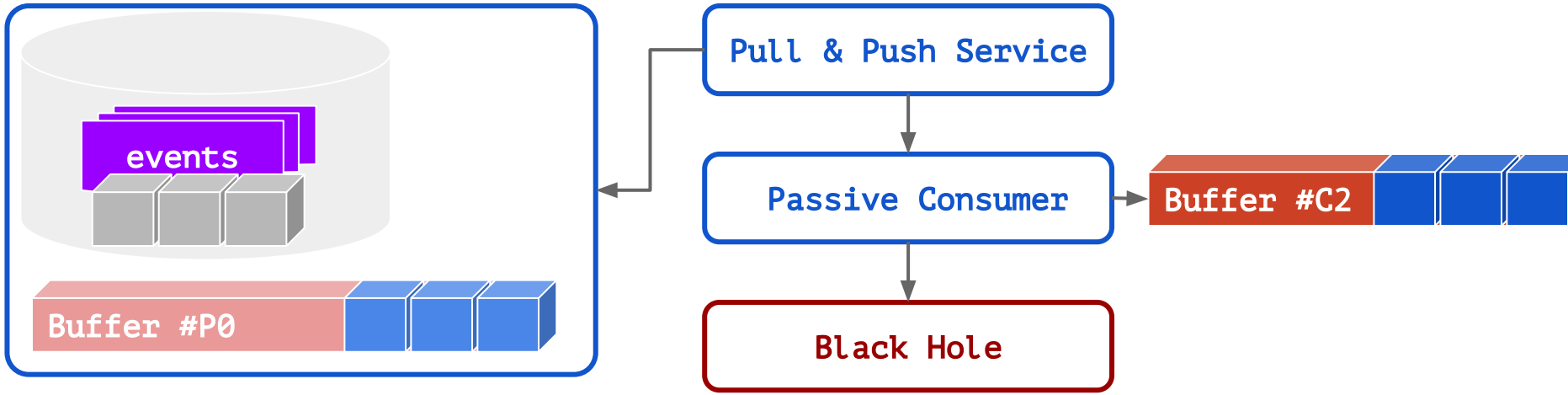
Push and Pull semantics related to consumer



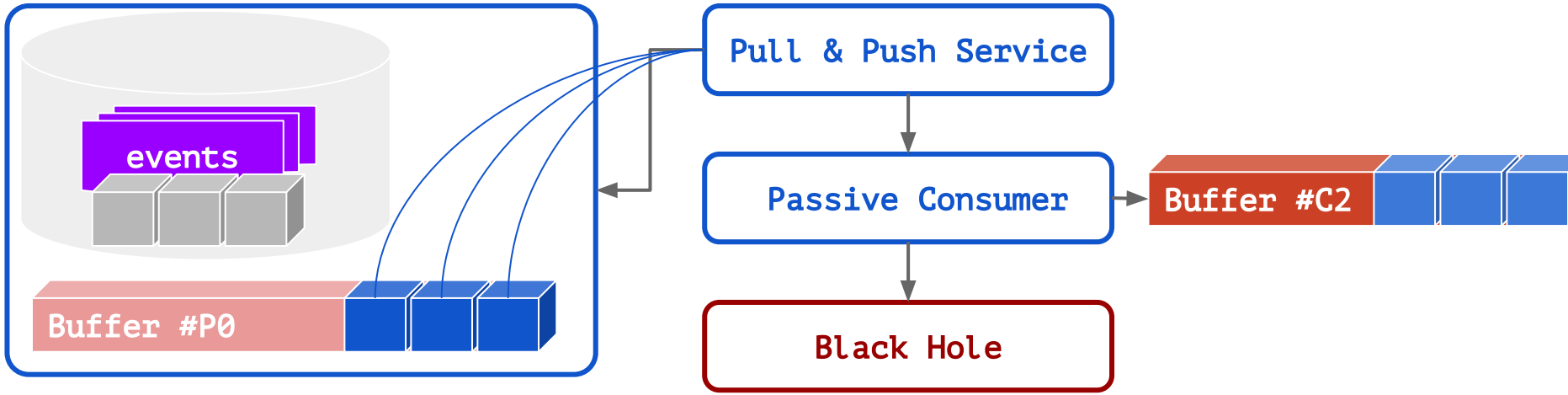
Push and Pull semantics related to consumer



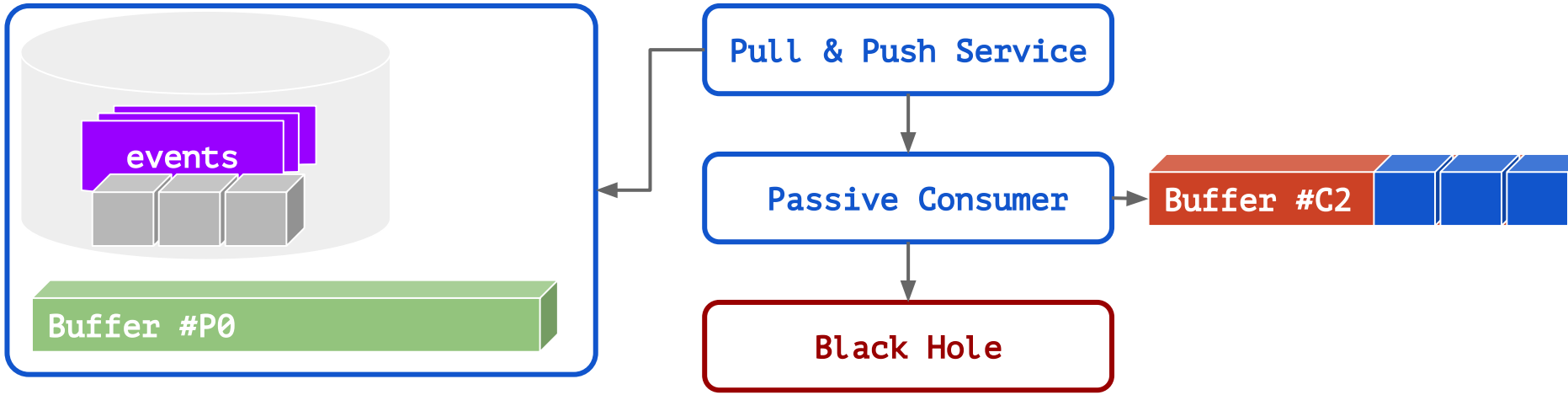
Push and **Pull** semantics related to consumer



Push and **Pull** semantics related to consumer



Push and **Pull** semantics related to consumer

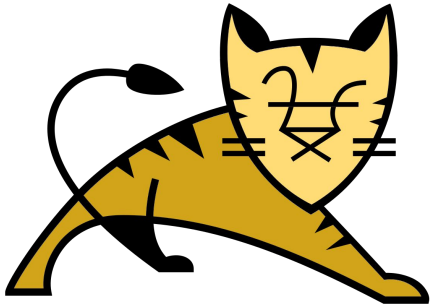


Webflux


Жизнь программиста
Все с чистого листа.
Все заново. В этот раз я сделаю
все как положено


Позже...
Я ведь сделал все
правильно?..

Spring Webflux & Project Reactor

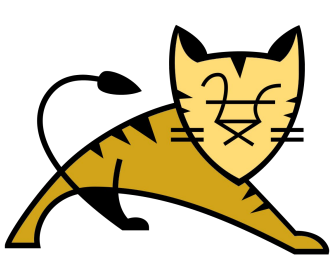


Spring Webflux & Project Reactor

Spring MVC (Servlet 3.1+)

or

Spring Webflux (default)



Подключаем webflux

Было **'org.springframework.boot:spring-boot-starter-web'**

Стало **'org.springframework.boot:spring-boot-starter-webflux'**

WebClient is a new RestTemplate

Было

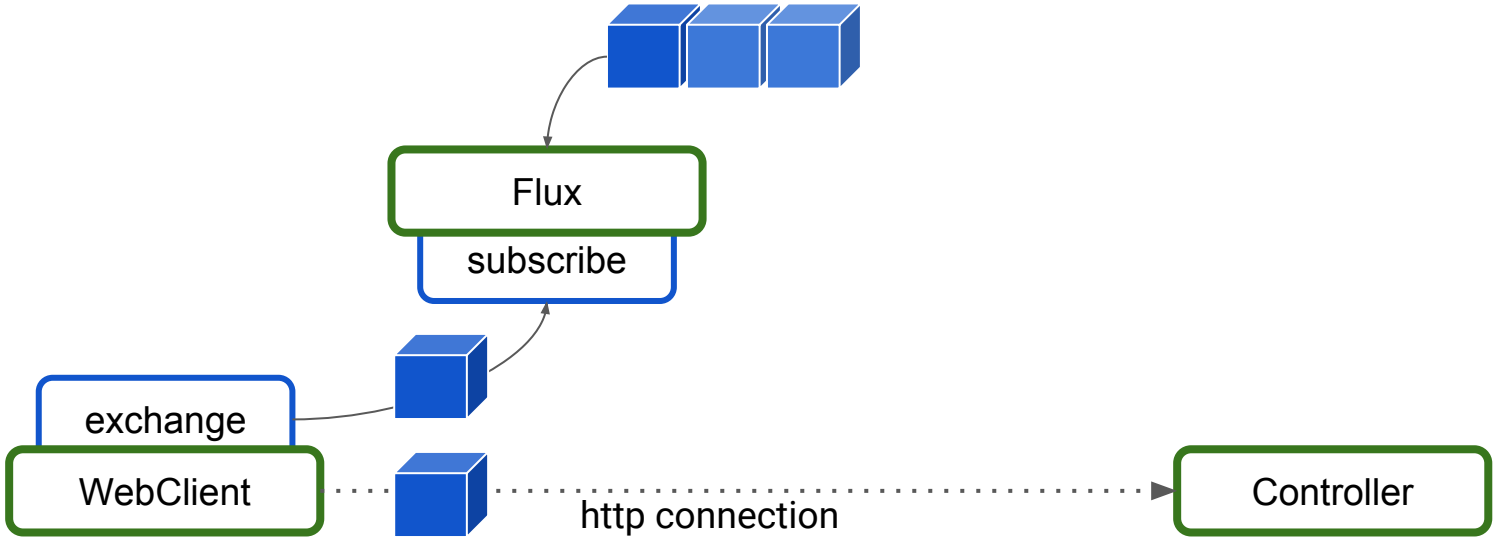
`RestTemplateBuilder → RestTemplate`

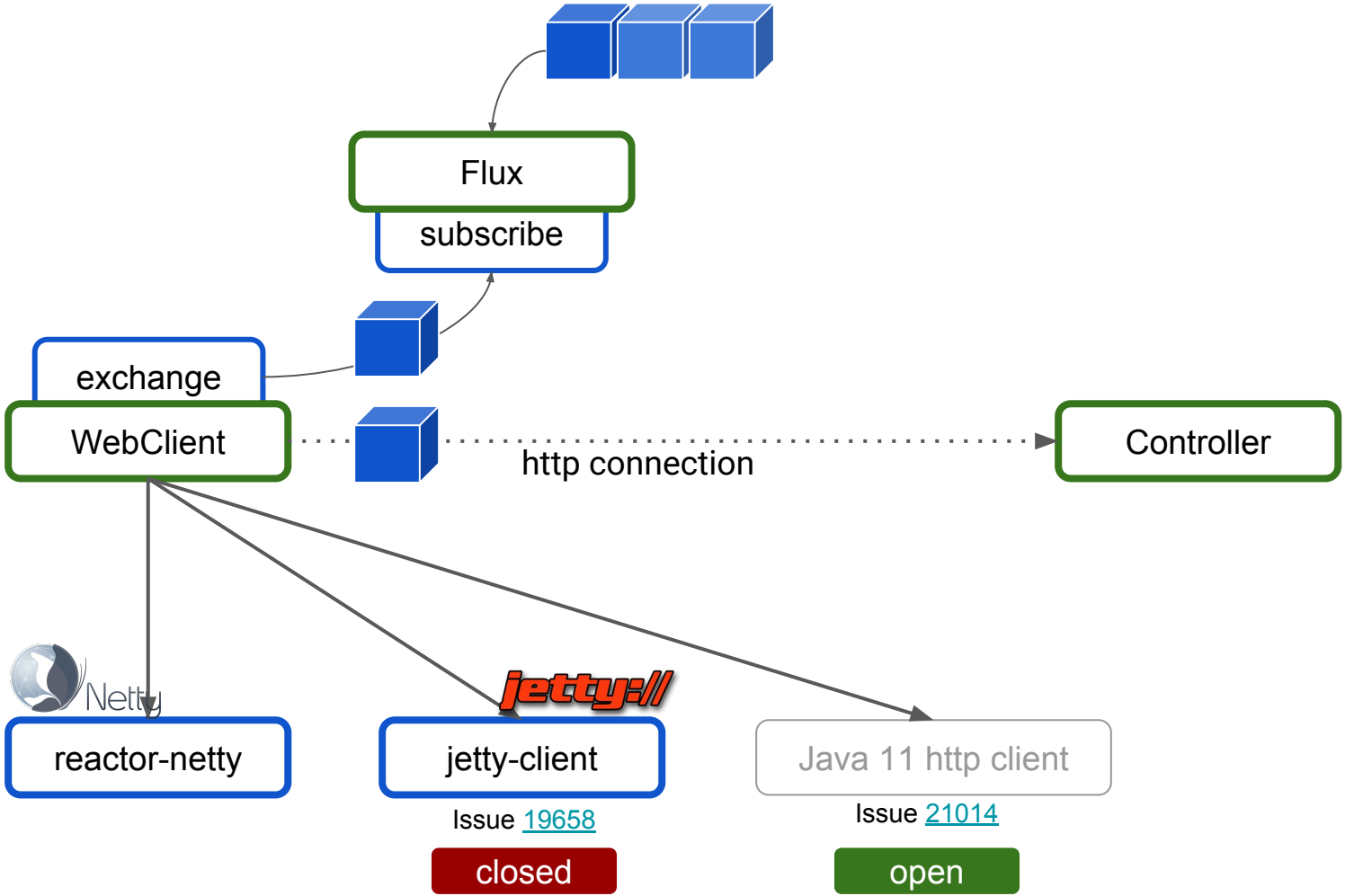
`RestTemplate —` передаём статический текс в body

Стало

`WebClient.Builder → WebClient`

`WebClient —` можем передать последовательность **`Flux`** в качестве body


Flux
subscribe
exchange
WebClient
http connection
Controller


Flux
subscribe
exchange
WebClient
http connection
Controller
Netty
reactor-netty
jetty://
jetty-client
Issue 19658
closed
Java 11 http client
Issue 21014
open

Letter generator

```
Flux.<Letter>generate(
    sink -> sink.next(randomLetter())
);
```

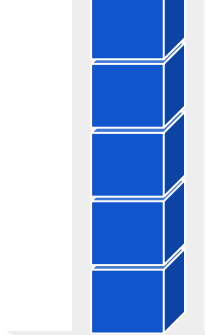
Demo

Webflux

WebFlux #1

```
public void processLetter(@RequestBody Flux<Letter> f)
```

400 BAD_REQUEST "Request body is missing

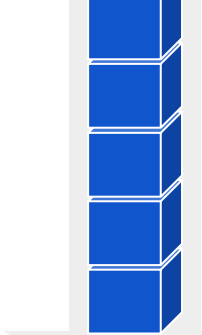


Close connection

LetterController

```
void processLetter(Flux<Letter> f)
  f.doOnNext()
    .doOnNext()
    .log()
    .subscribe()

    return Mono.empty() / nothing / anything
```

400 BAD_REQUEST "Request body is missing



Close connection

LetterController

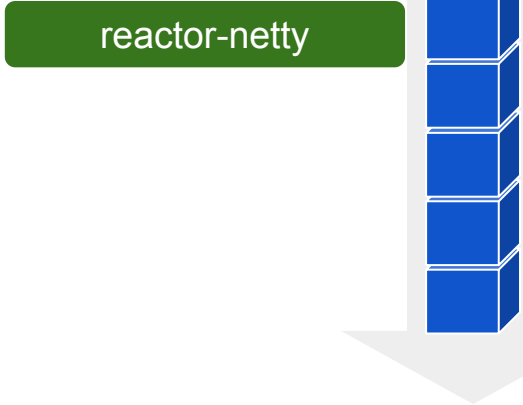
```
void processLetter(Flux<Letter> f)
  f.doOnNext()
    .doOnNext()
    .log()
    .subscribe()

    return Mono.empty() / nothing / anything
```

Для начала вычитывания

400 BAD_REQUEST "Request body is missing

Flux пайплайн жизненный цикл

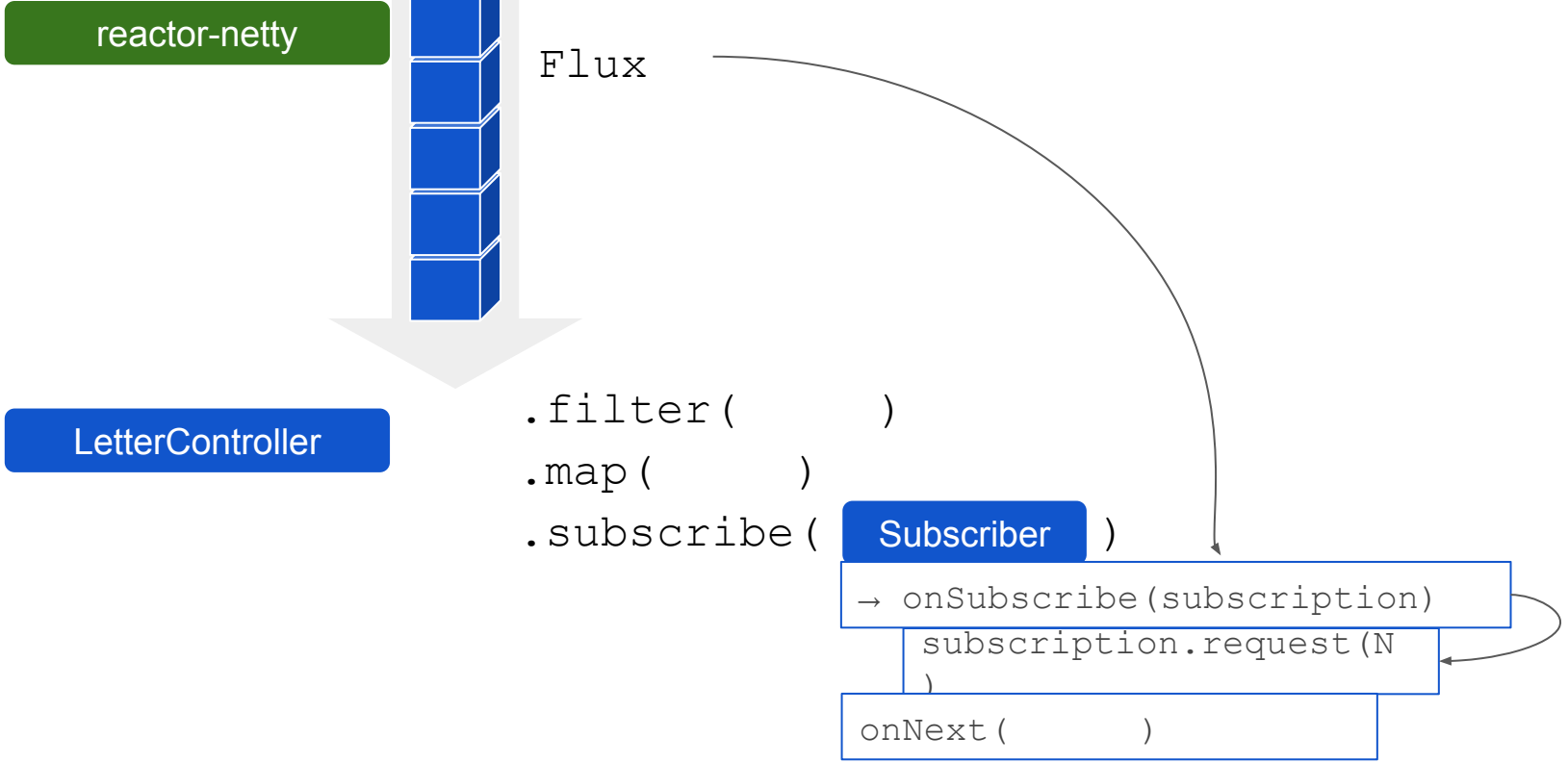
1. Assembly
   Как `new Builder().prop().build()`
   Только `Flux.create().subscribe(`**`subscriber`**`)`
2. В **`subscriber.onSubscribe()`** передаётся подписка – subscription
3. Через subscription выполняется первый запрос на ивенты или отписка

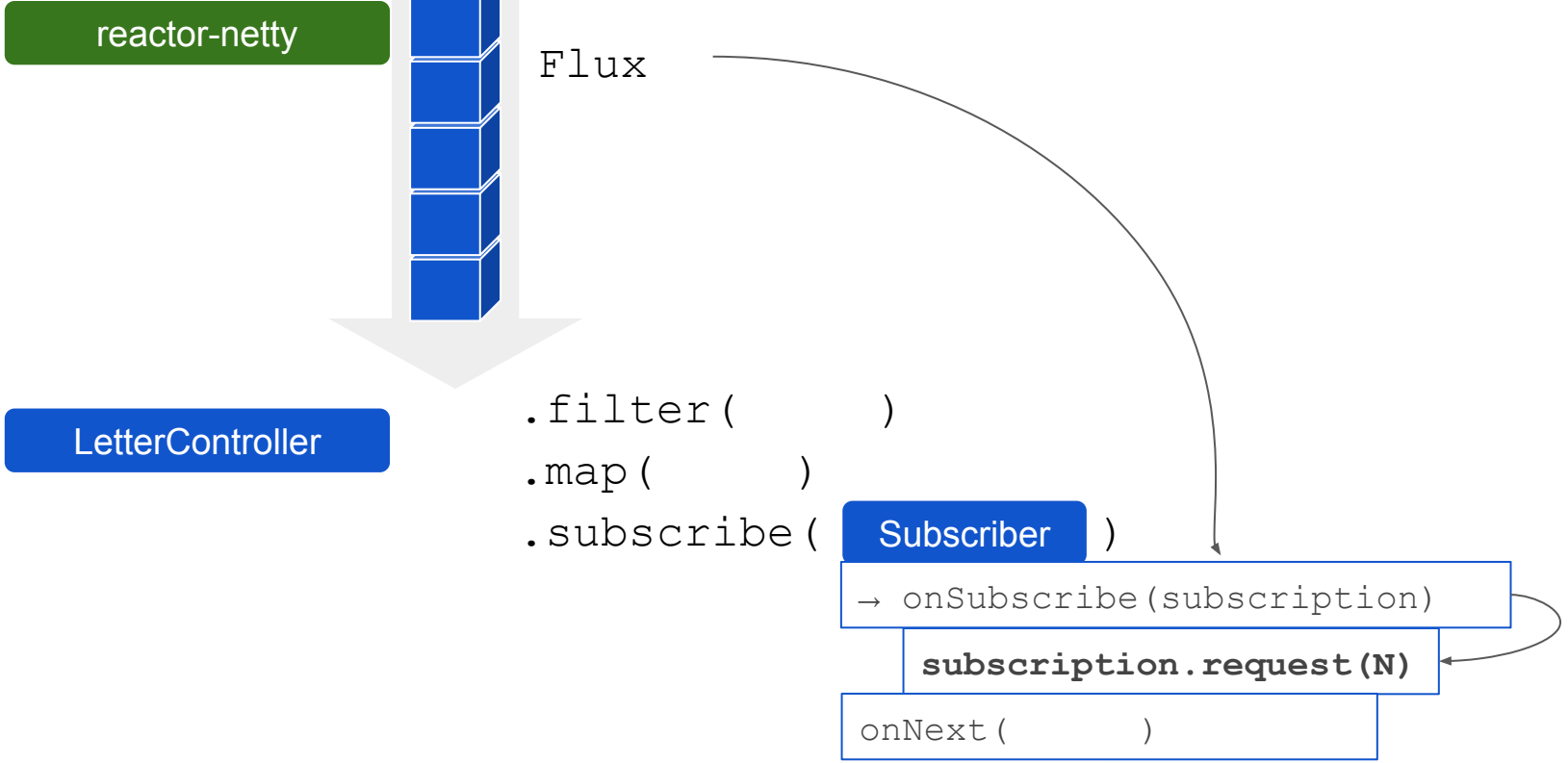


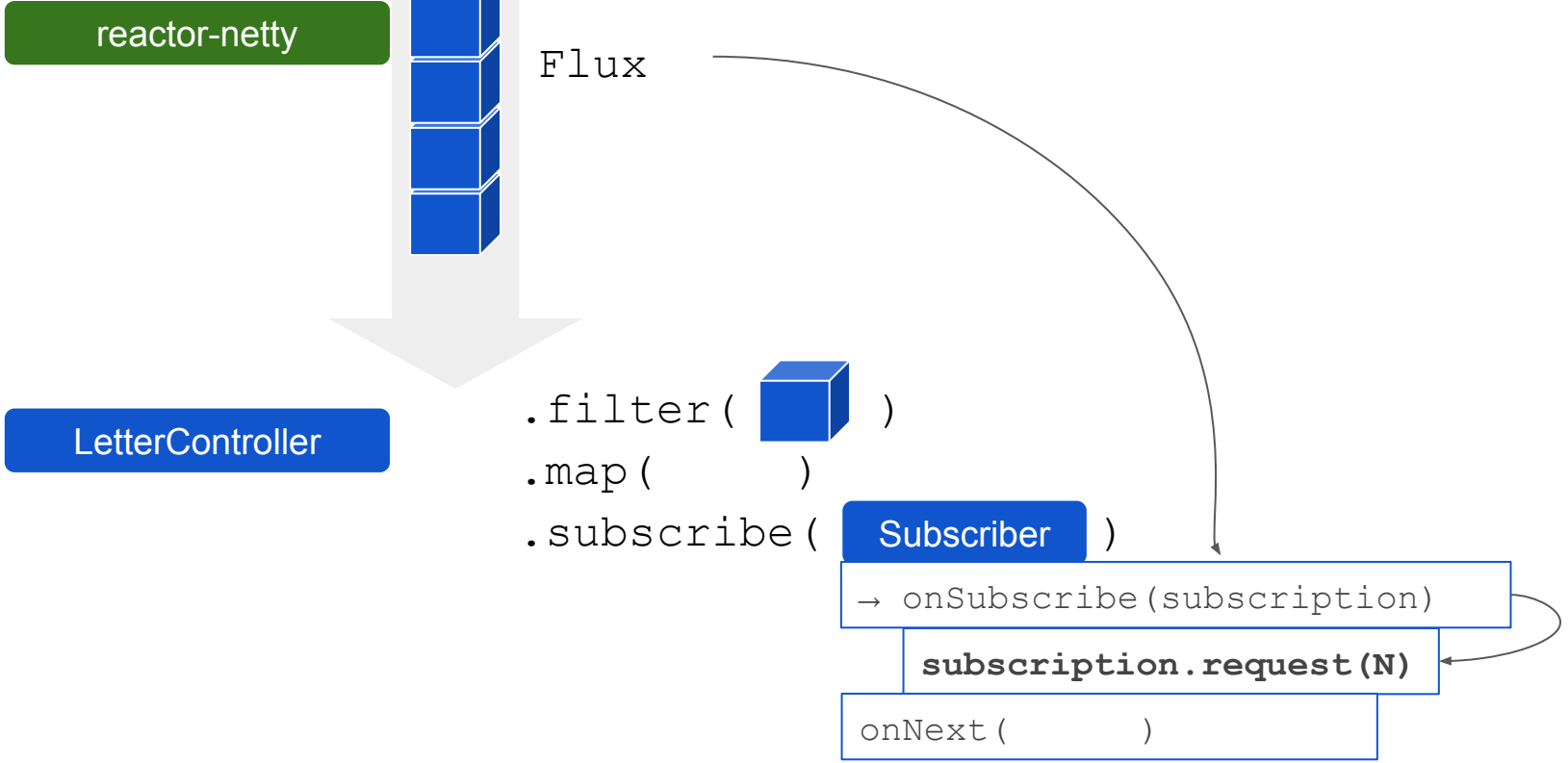
Flux

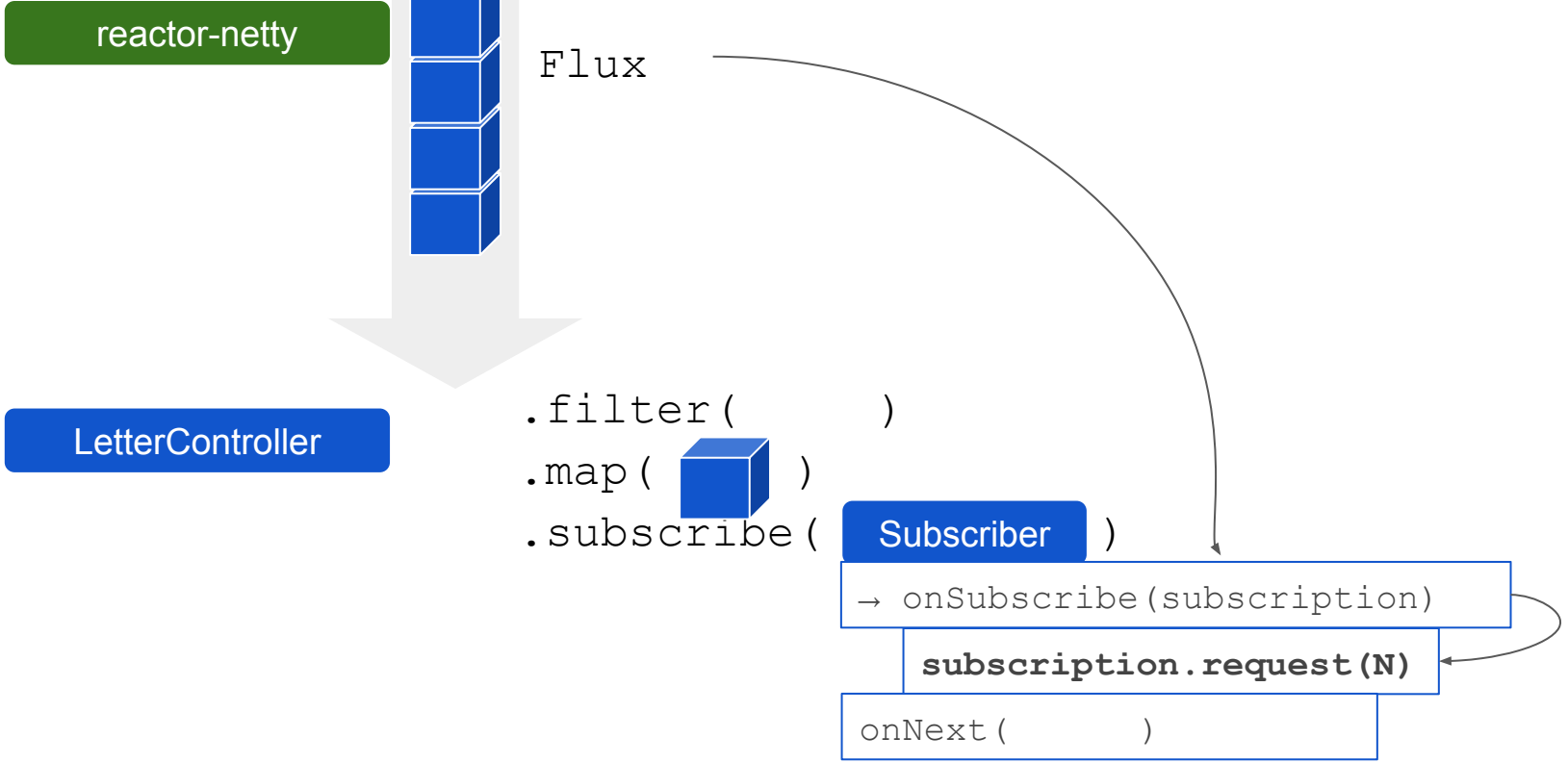
LetterController

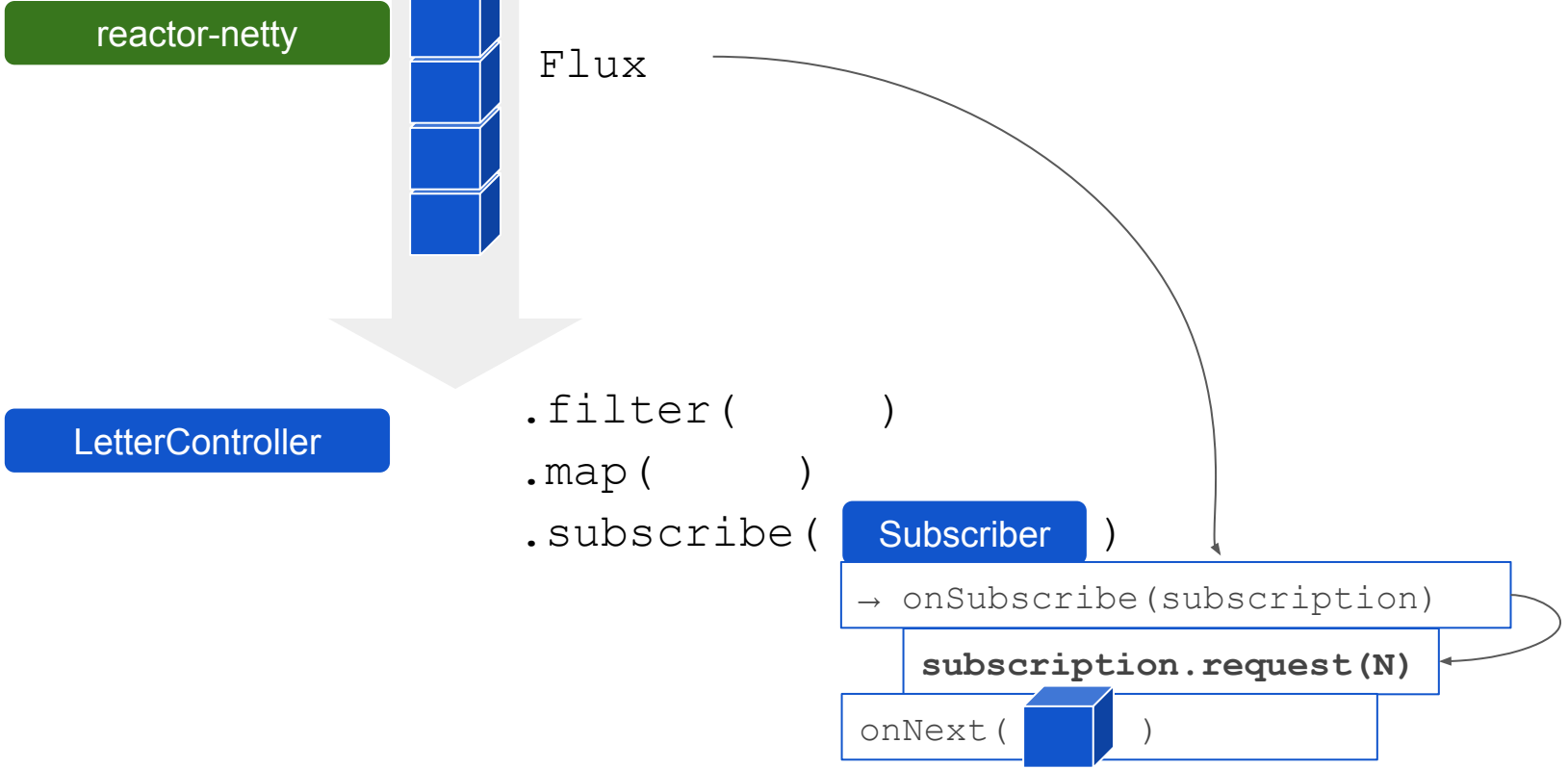
subscribe(Subscriber)

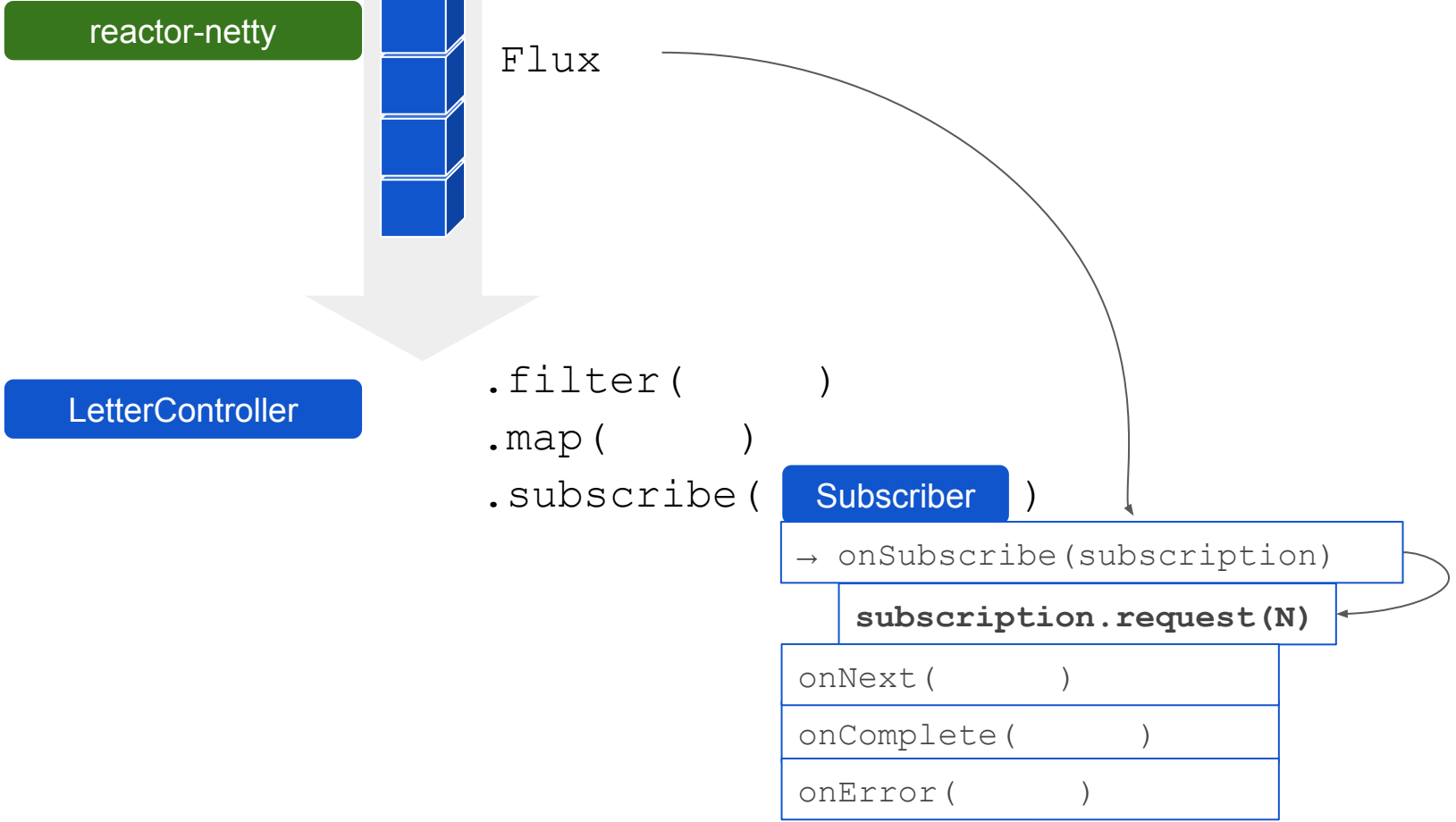

reactor-netty
Flux
LetterController
.filter( )
.map( )
.subscribe(Subscriber)
→ onSubscribe(subscription)
subscription.request(N
)
onNext( )


reactor-netty
Flux
LetterController
.filter( )
.map( )
.subscribe(Subscriber)
→ onSubscribe(subscription)
subscription.request(N)
onNext( )


reactor-netty
Flux
LetterController
.filter( )
.map( )
.subscribe(Subscriber)
→ onSubscribe(subscription)
subscription.request(N)
onNext( )


reactor-netty
Flux
LetterController
.filter( )
.map( )
.subscribe(Subscriber)
→ onSubscribe(subscription)
subscription.request(N)
onNext( )

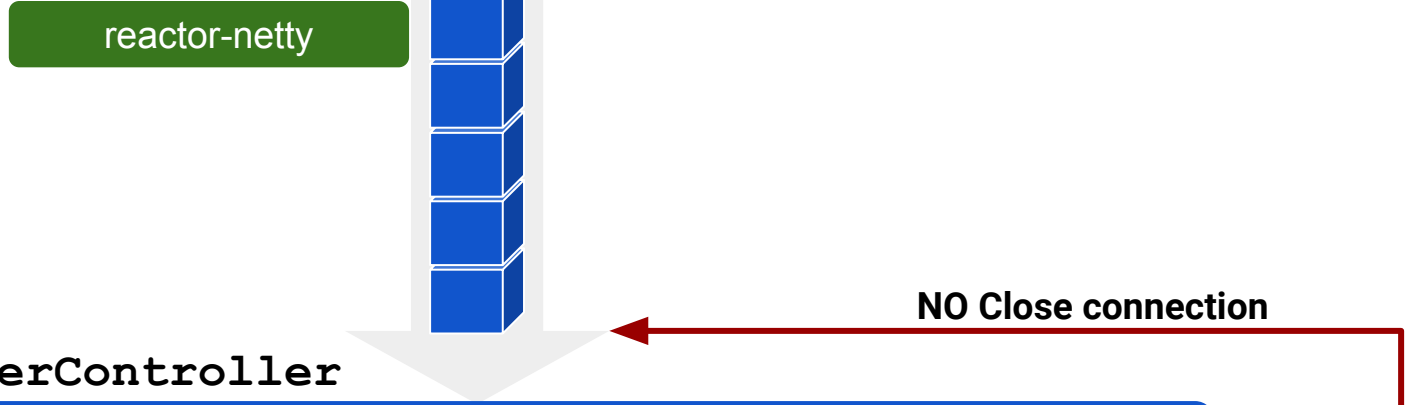

reactor-netty
Flux
LetterController
.filter( )
.map( )
.subscribe(Subscriber)
→ onSubscribe(subscription)
subscription.request(N)
onNext( )


reactor-netty
Flux
LetterController
.filter( )
.map( )
.subscribe(Subscriber)
→ onSubscribe(subscription)
subscription.request(N)
onNext( )
onComplete( )
onError( )

WebFlux #1

```
public Mono<Void> processLetter(@RequestBody Flux<Letter> f)
```

```
Mono.empty()

Mono.never()
```



LetterController

```
Mon<Void> processLetter(Flux<Letter> f)
  f.doOnNext()
    .doOnNext()
    .log()
    .subscribe()

  return Mono.never()
```

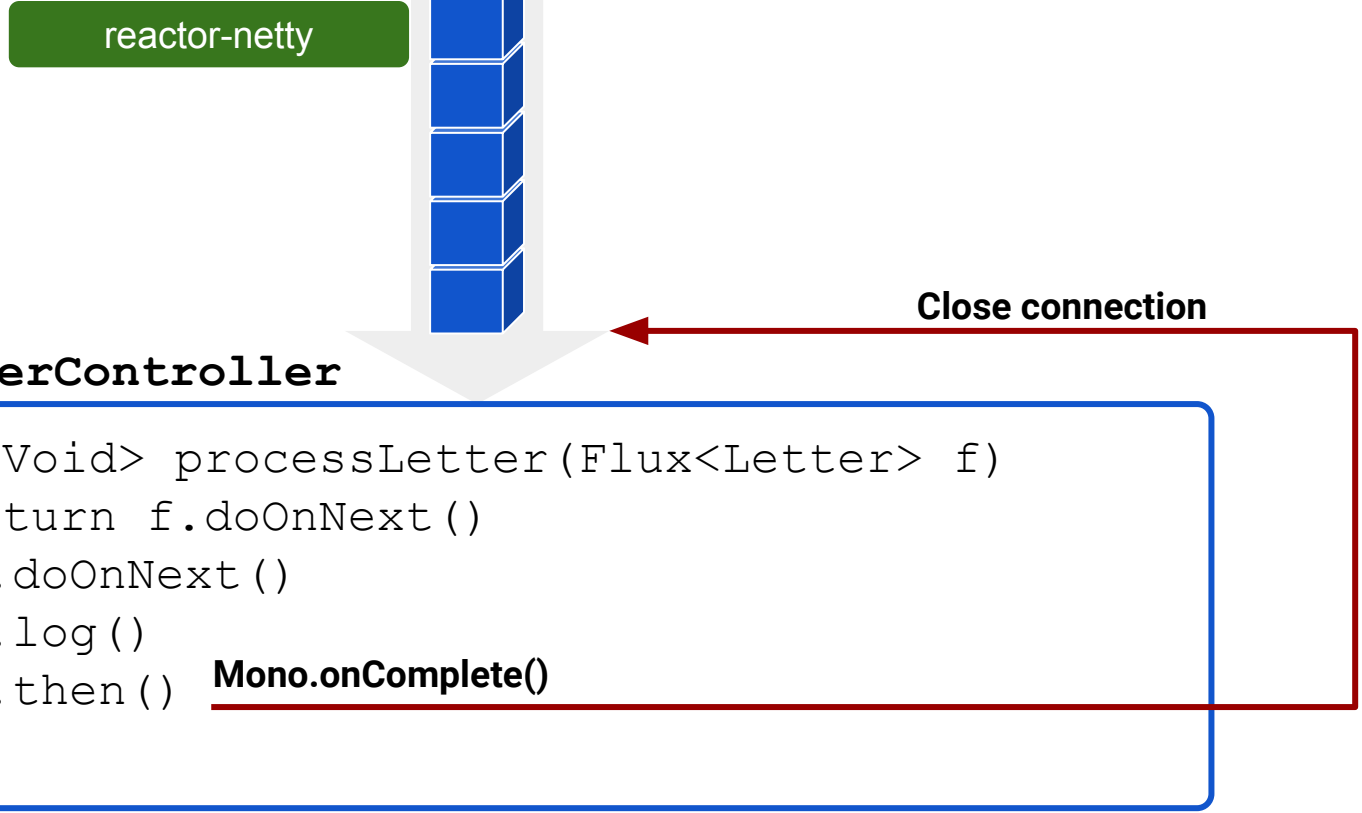
Немного о HTTP

request

GET / HTTP 1/1
HOST: dfsdfg

\n\r

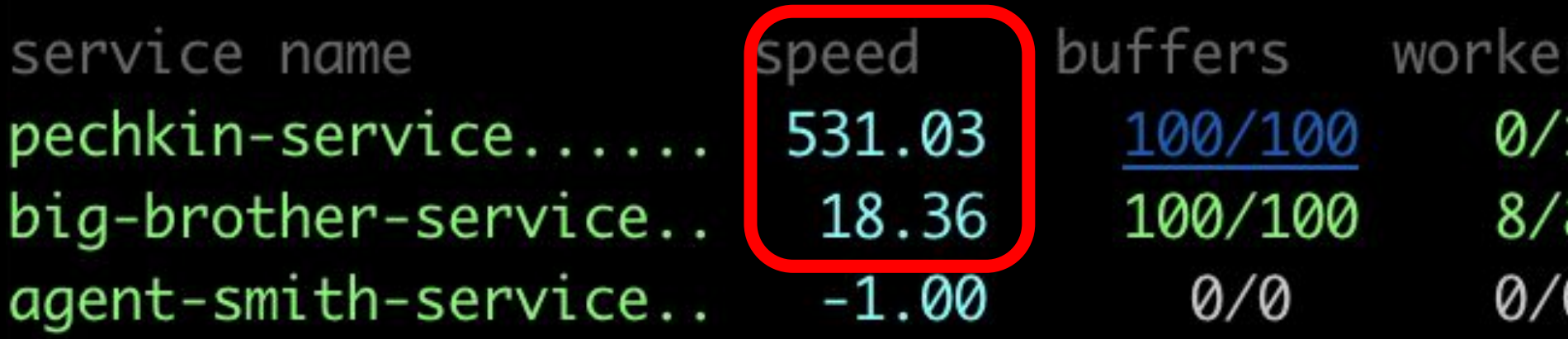
response


reactor-netty
Close connection
LetterController
Mon<Void> processLetter(Flux<Letter> f)
return f.doOnNext()
.doOnNext()
.log()
.then()
Mono.onComplete()

Demo

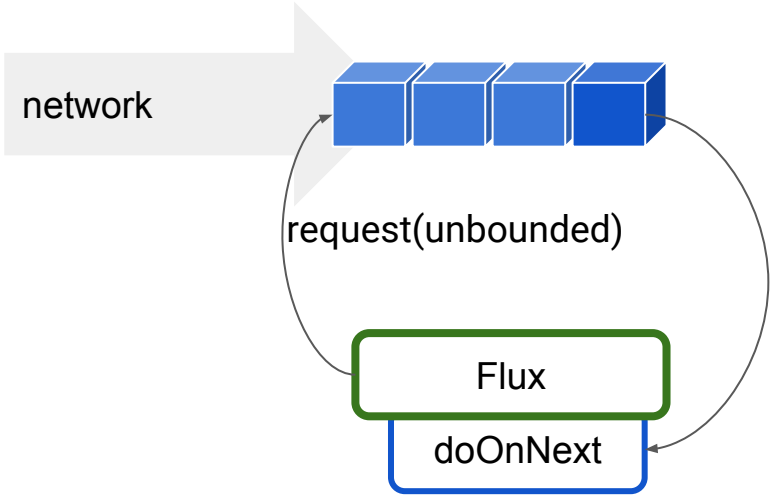
Webflux #2

Чувак, где мои письма

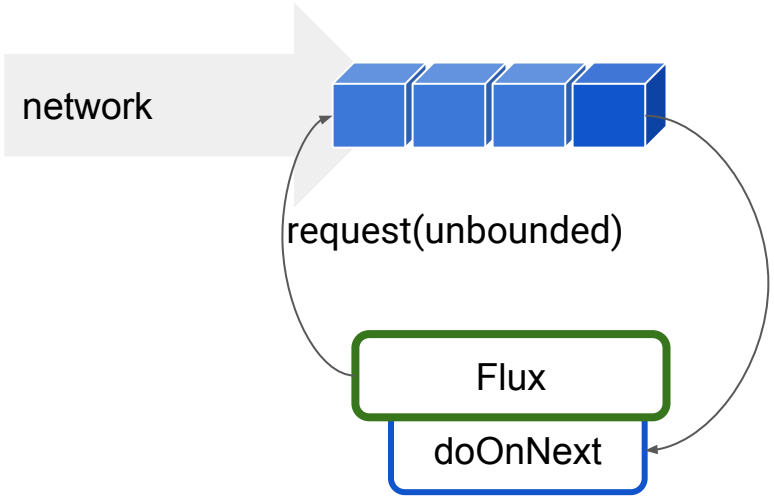


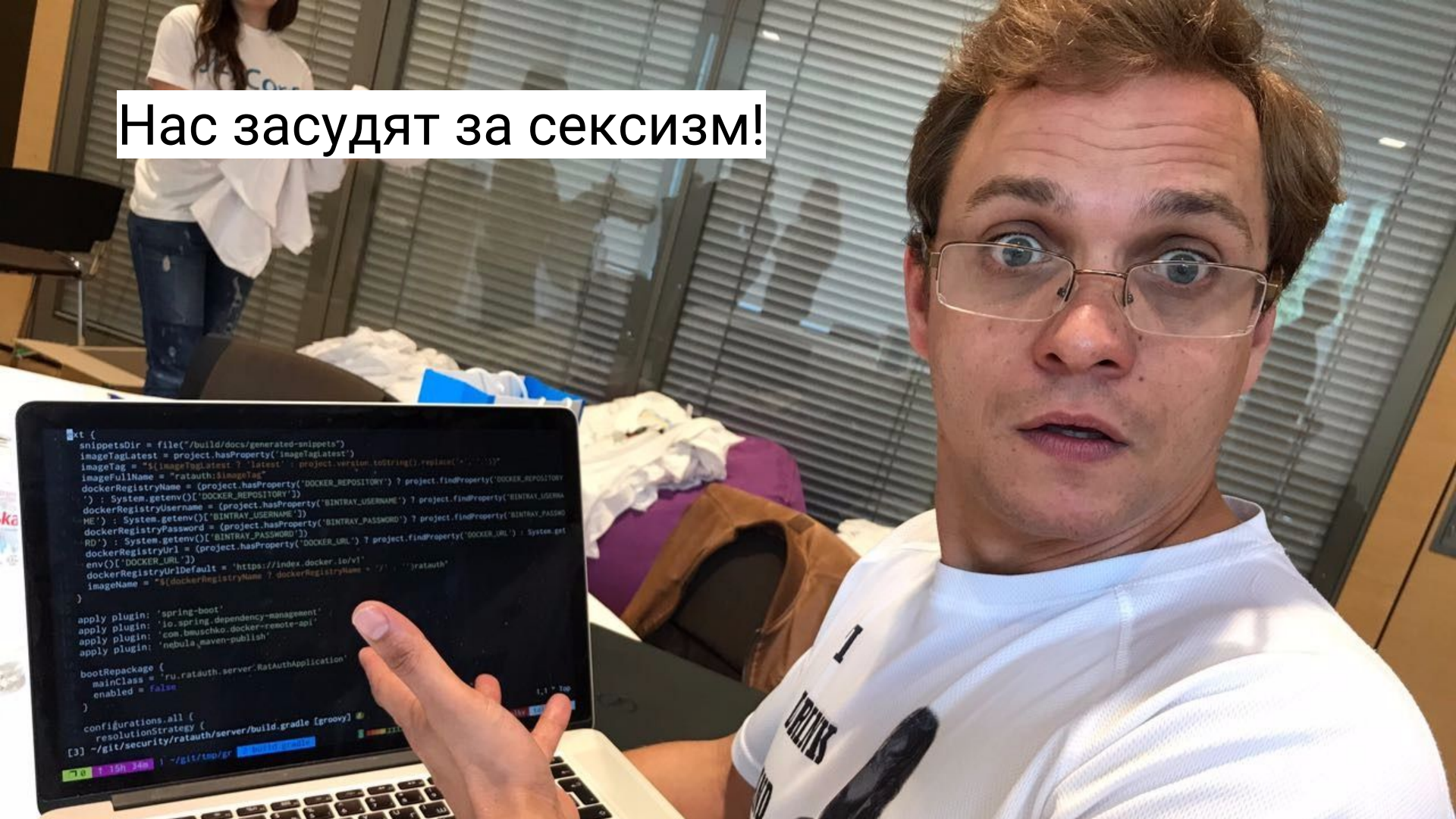
У нас же есть Backpressure из коробки!

Чувак, где мои письма



Чувак, где мои письма – в буферах!

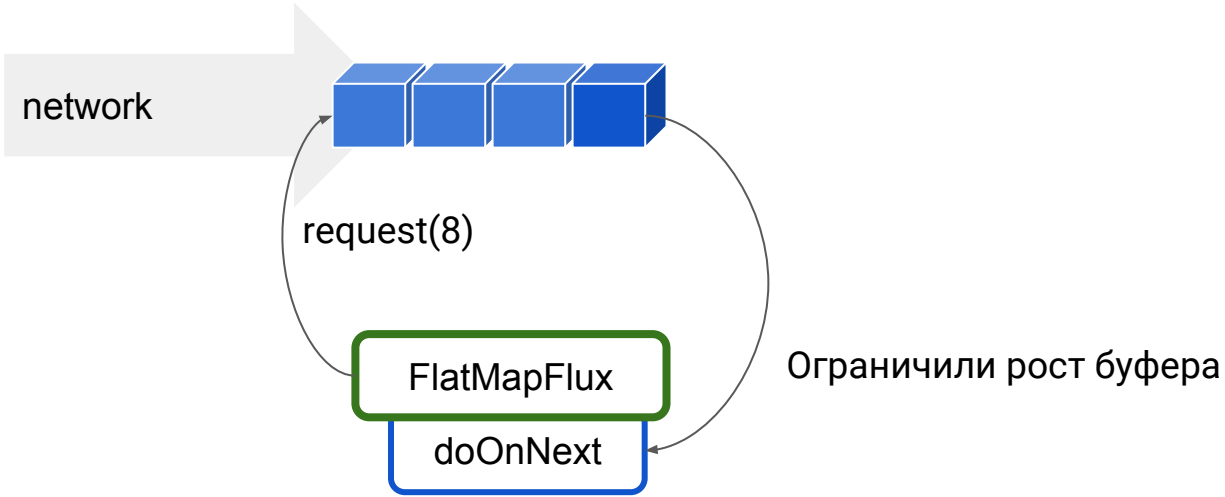



Нас засудят за сексизм!

Нас засудят за сексизм!

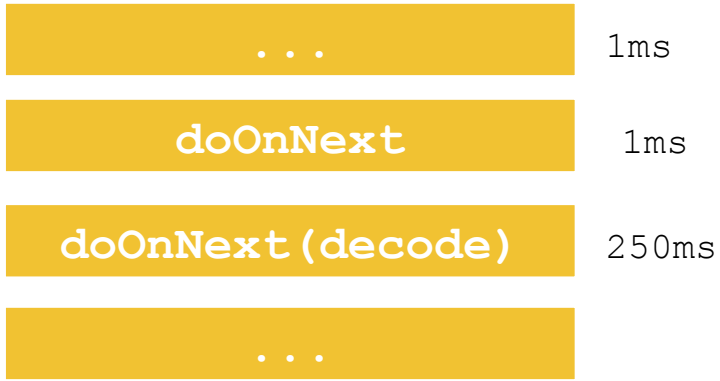
Чувак, где мои письма – в буферах!

Чувак/**чувиха**, где мои письма – в буферах!



Чувак, почему так медленно?

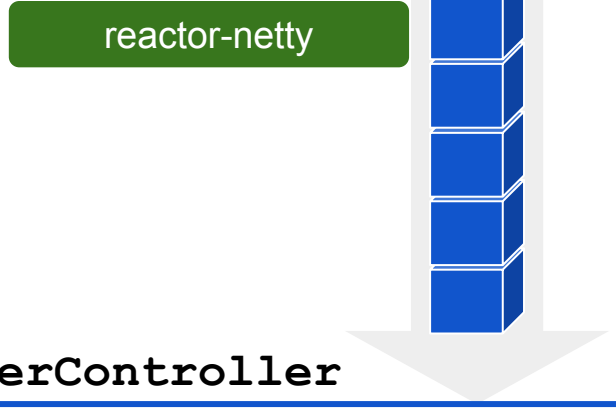
1000rps vs 2rps



Чувак, где мои письма

Demo

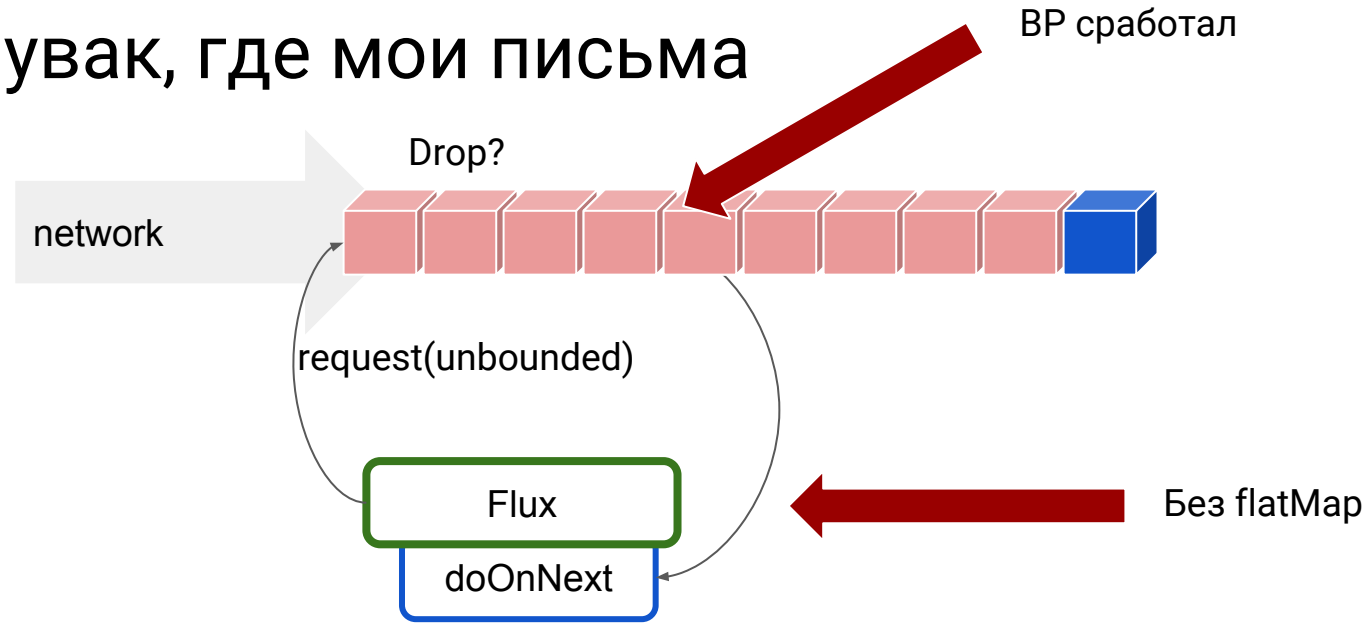
Webflux #3



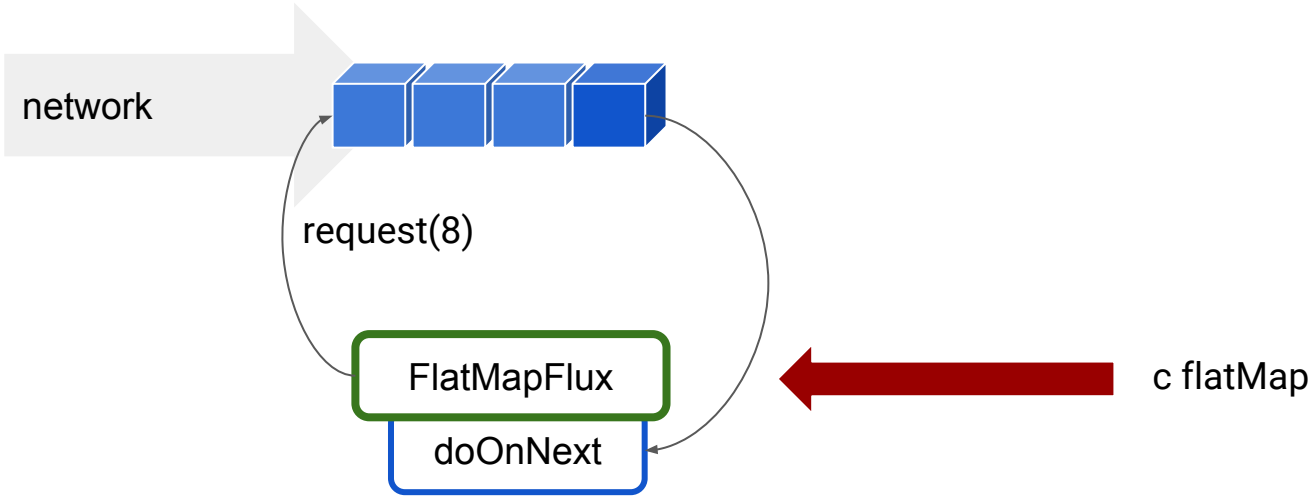
LetterController

```
Mon<Void> processLetter(Flux<Letter> f)
  return f.doOnNext()
    .flatMap(
      letter -> Mono.fromCallable(() -> decode(letter))
        .subscribeOn(fromExecutor(threadPool)),
      threadPool.getMaximumPoolSize())
    .log()
    .then()
```

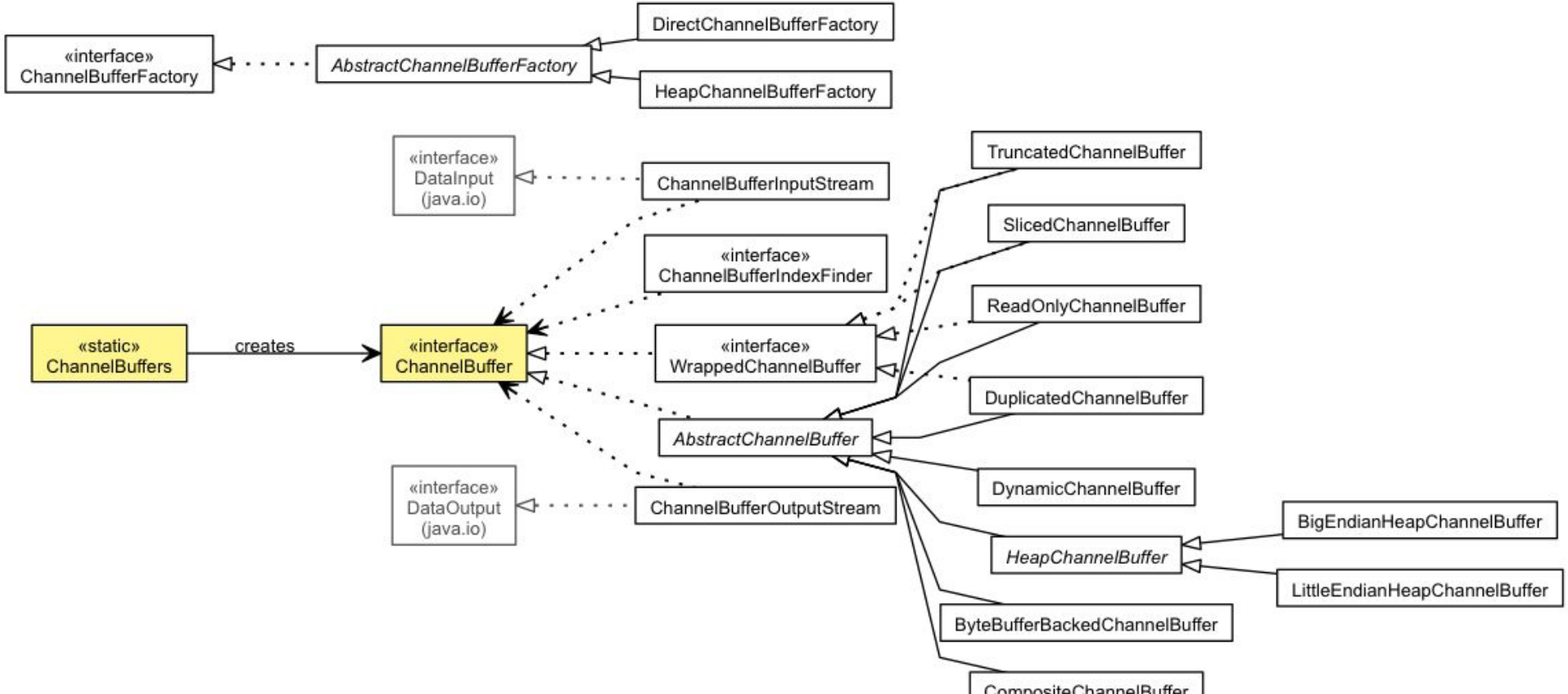
Чувак, где мои письма



Чувак, где мои письма

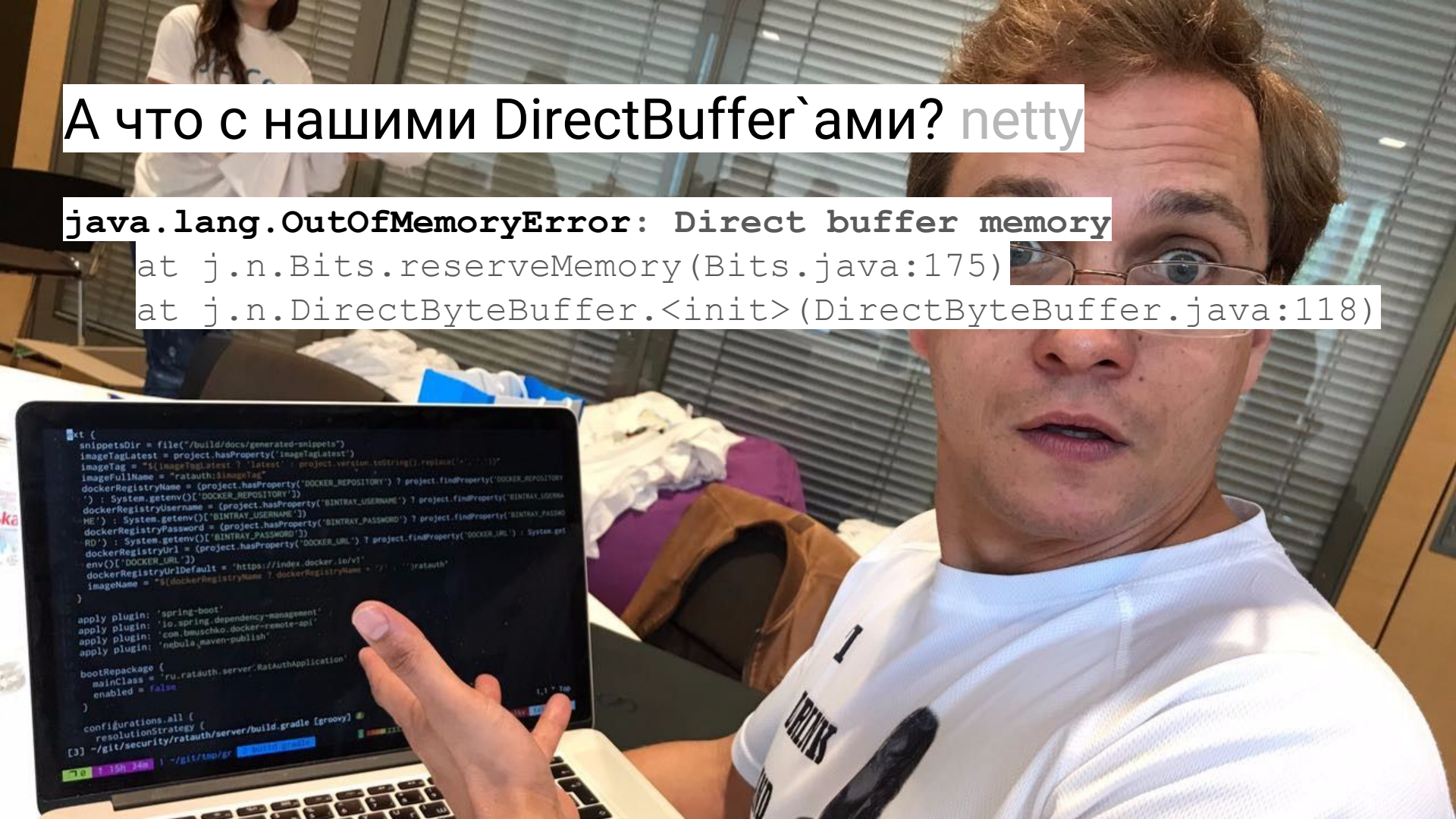


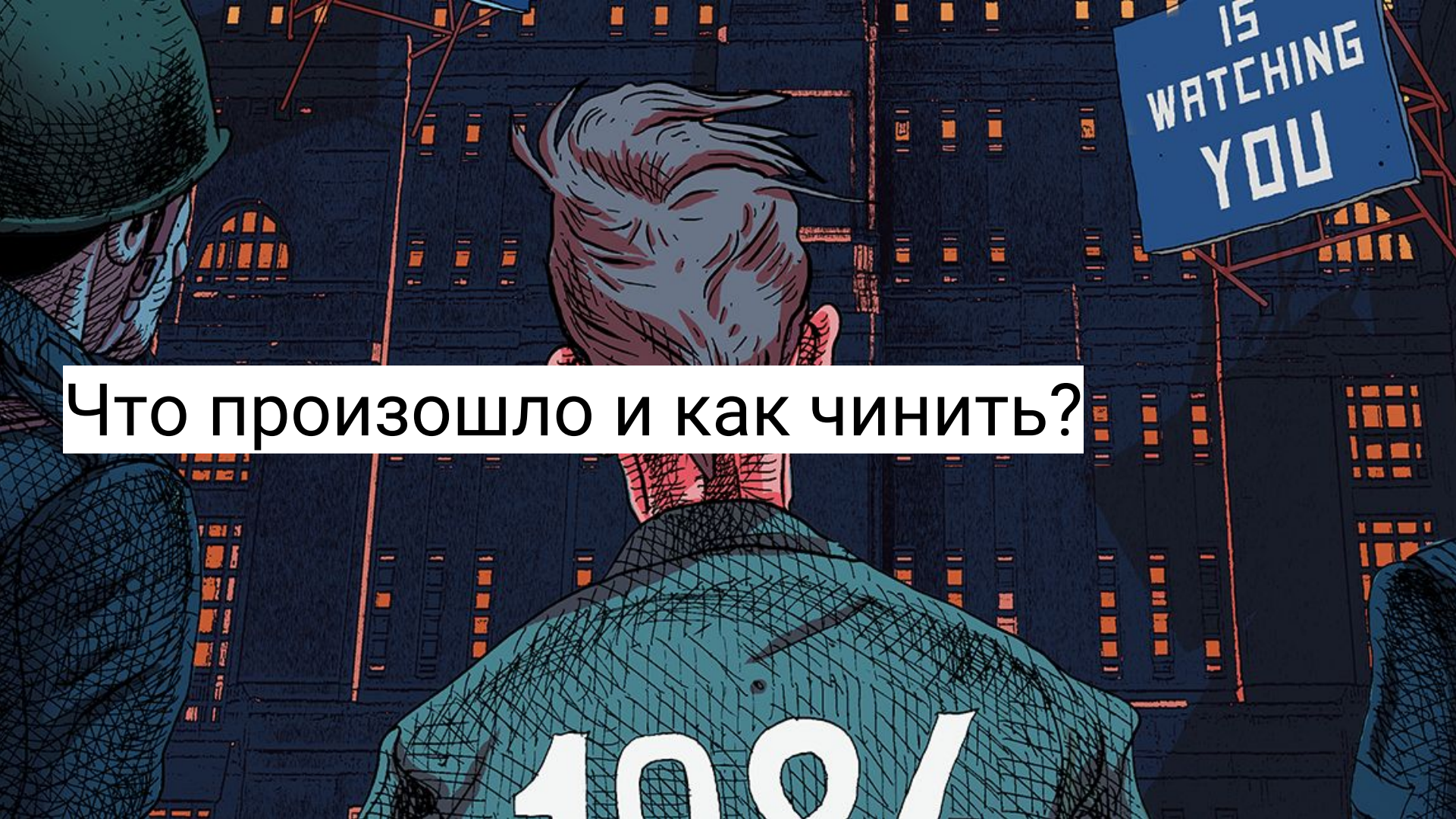
А что с нашими DirectBuffer`ами? netty



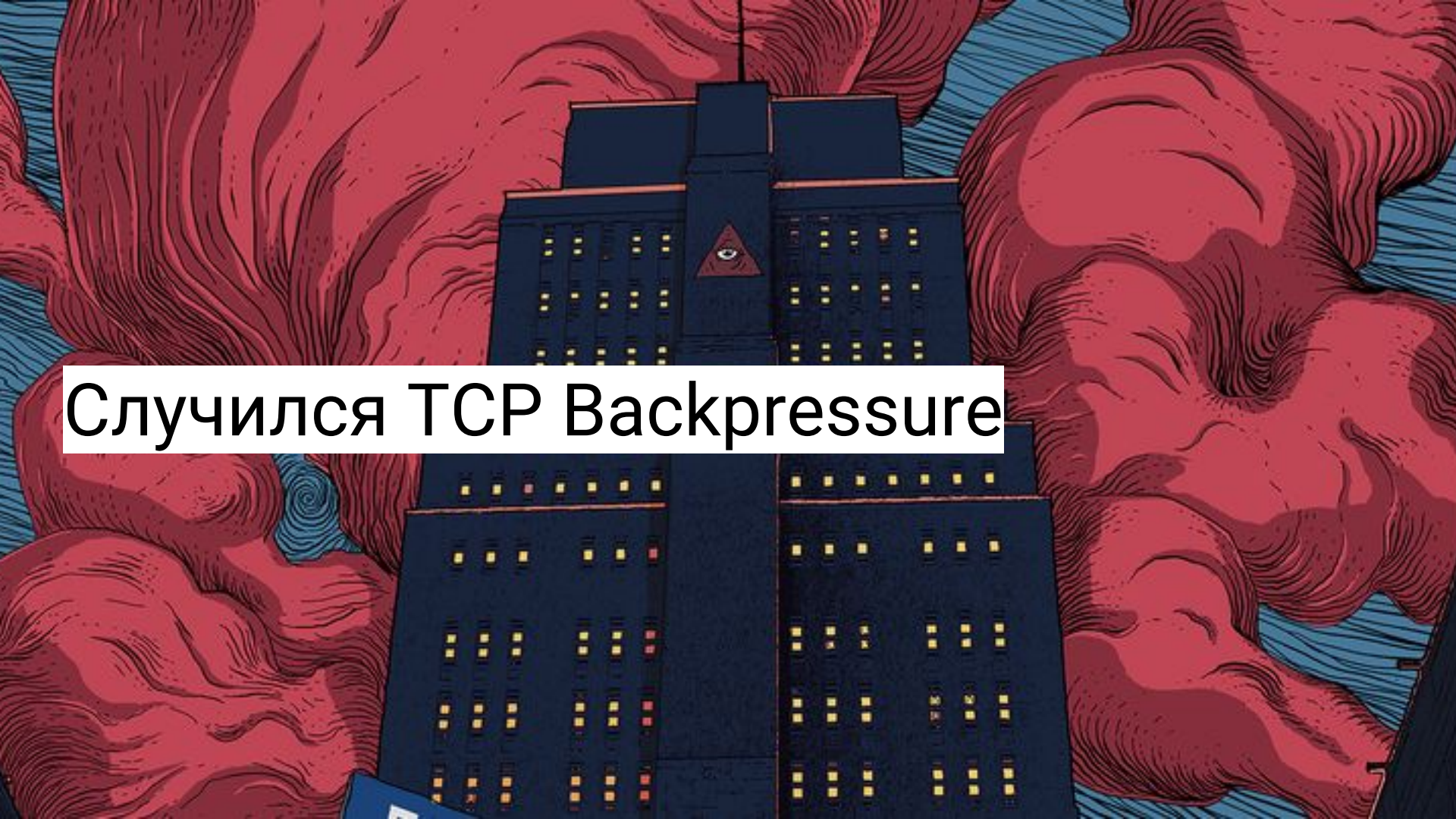
А что с нашими DirectBuffer`ами? netty

```
java.lang.OutOfMemoryError: Direct buffer memory
    at j.n.Bits.reserveMemory(Bits.java:175)
    at j.n.DirectByteBuffer.<init>(DirectByteBuffer.java:118)
```





Что произошло и как чинить?


Случился TCP Backpressure

1
1
Service A
WebFlux
TCP
WebFlux
Service B
request(10)
onNext(1..10)
bytes
request(1)
onNext(1)
2
acknowledge
request(10)
3
Enqueued
Long Running Job
4
Wait acknowledgement
request(1)
acknowledge
request(1)
Timeline


Как то так, через хитро
закрученную жопу
оно и работает

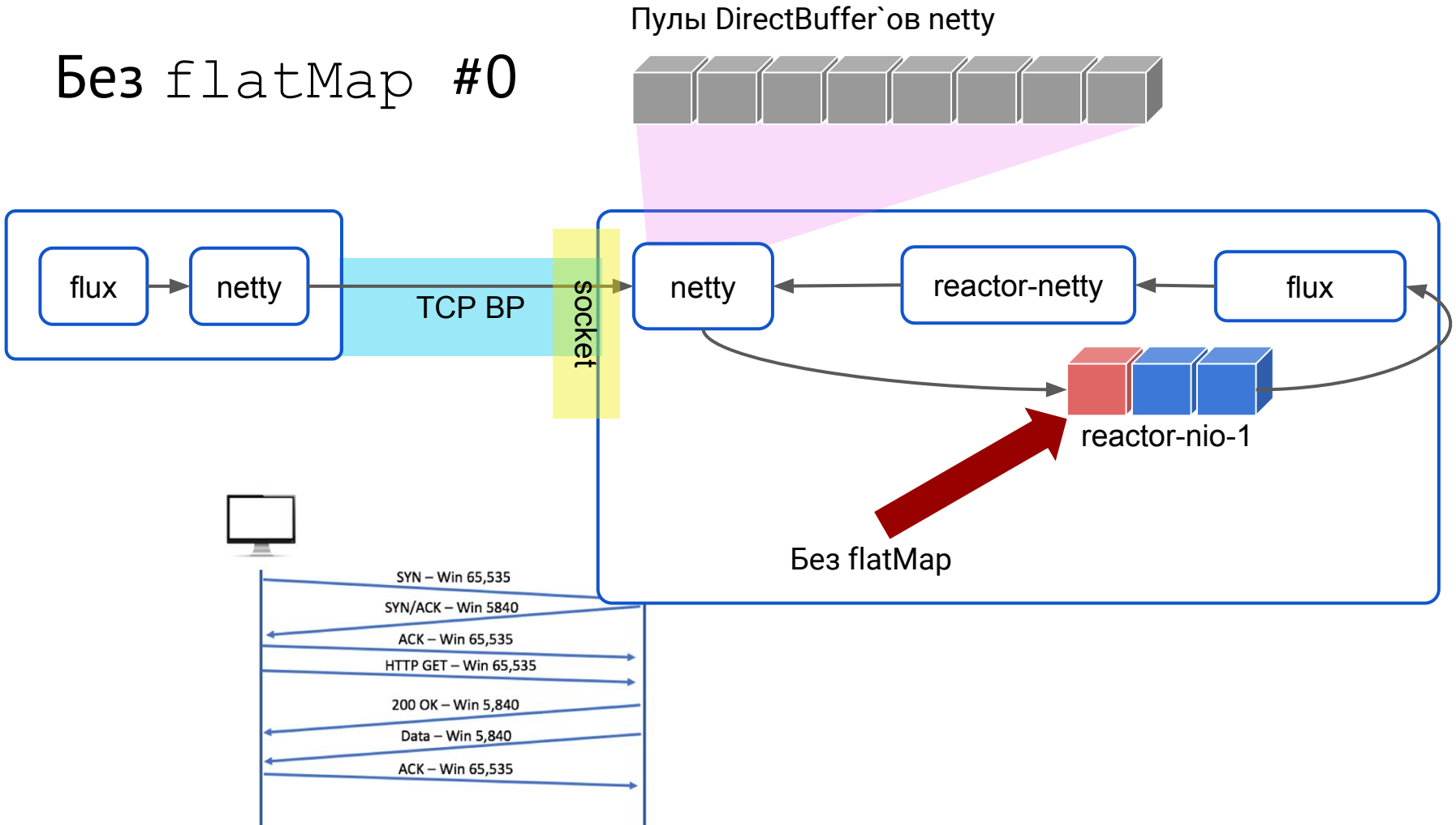
Почувствуйте разницу

```
[letter-4] onNext
[letter-4] onNext
[letter-2] onNext
[letter-1] onNext
[letter-4] onNext
[letter-3] onNext
```

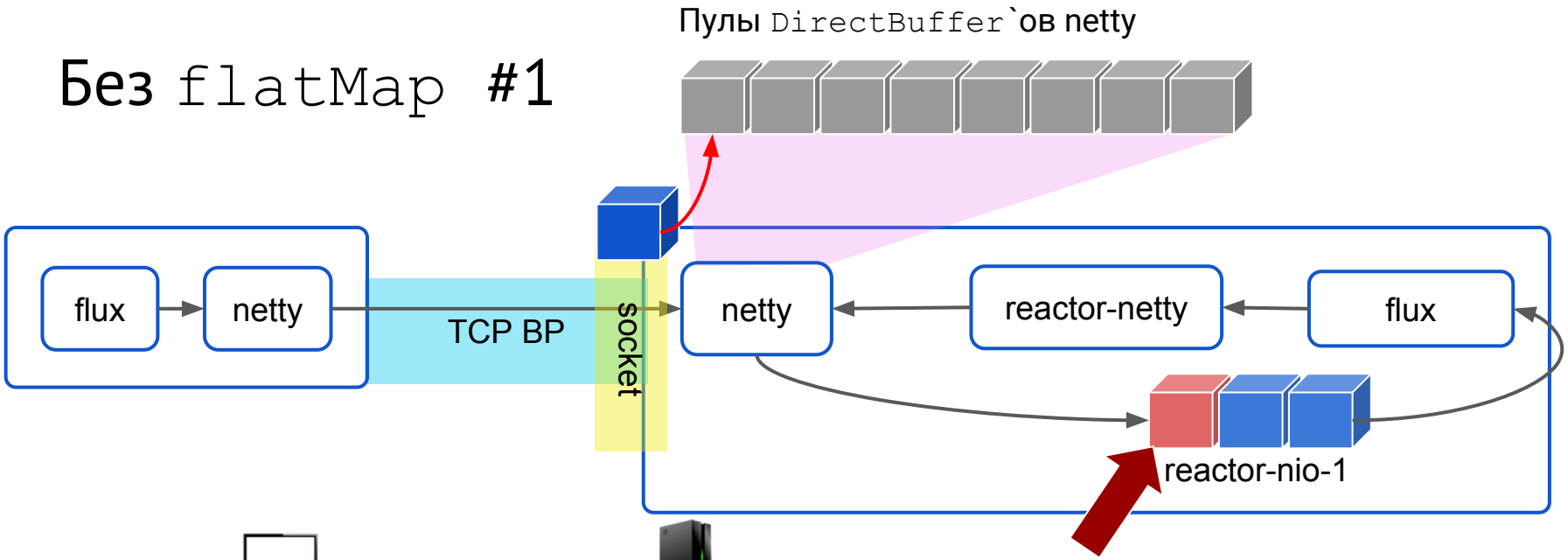
vs

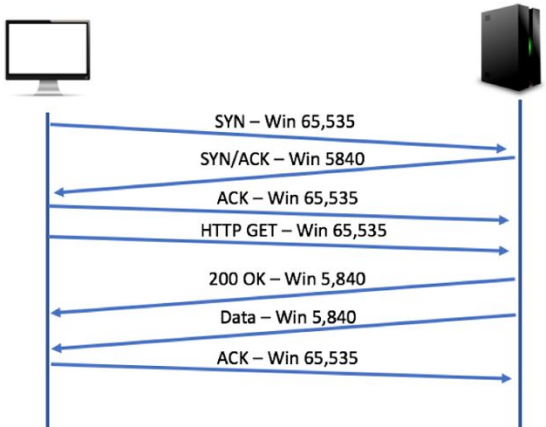
```
[reactor-http-nio-3] onNext
[reactor-http-nio-3] onNext
[reactor-http-nio-3] onNext
[reactor-http-nio-3] onNext
[reactor-http-nio-3] onNext
[reactor-http-nio-3] onNext
```

Без `flatMap` #0



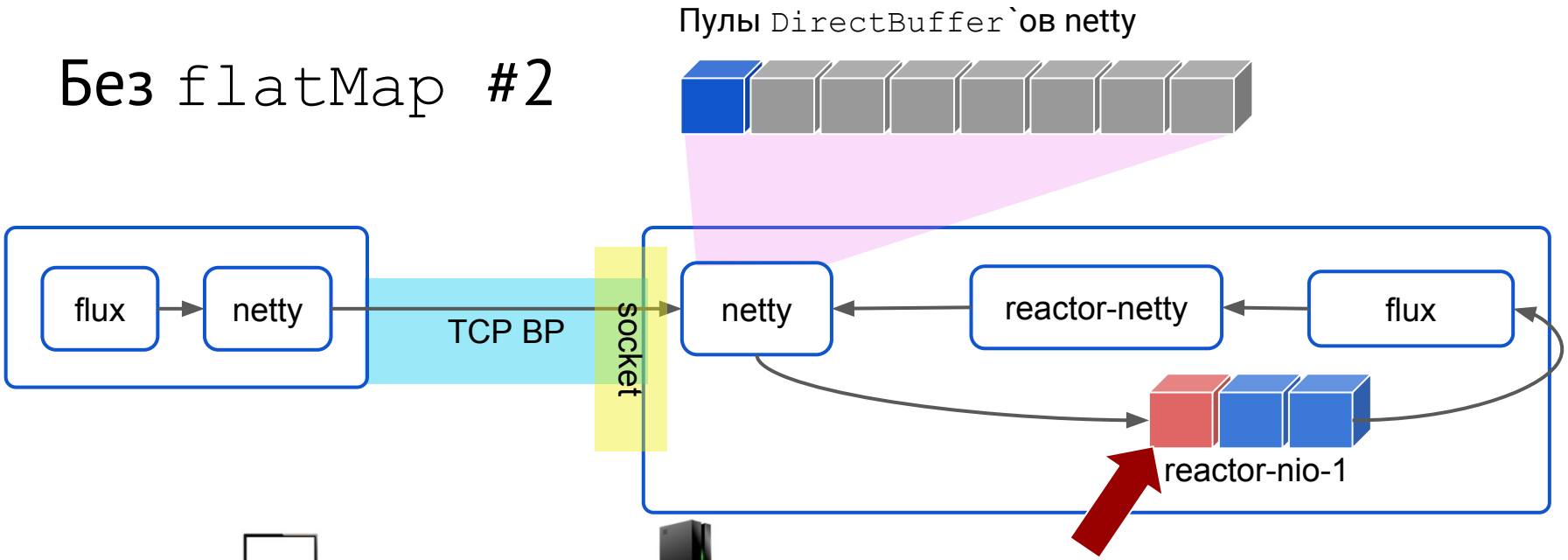
Без `flatMap` #1

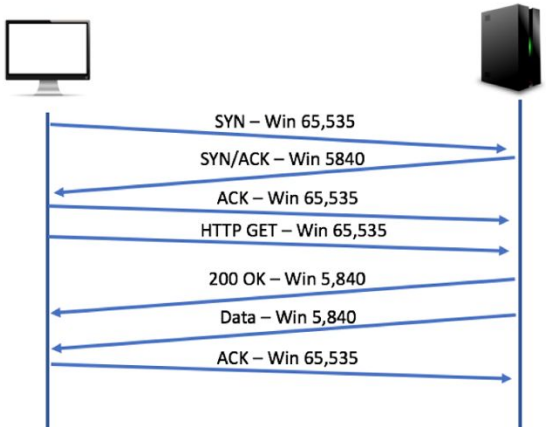




1. Копирует данные из системного сокета в `DirectBuffer`

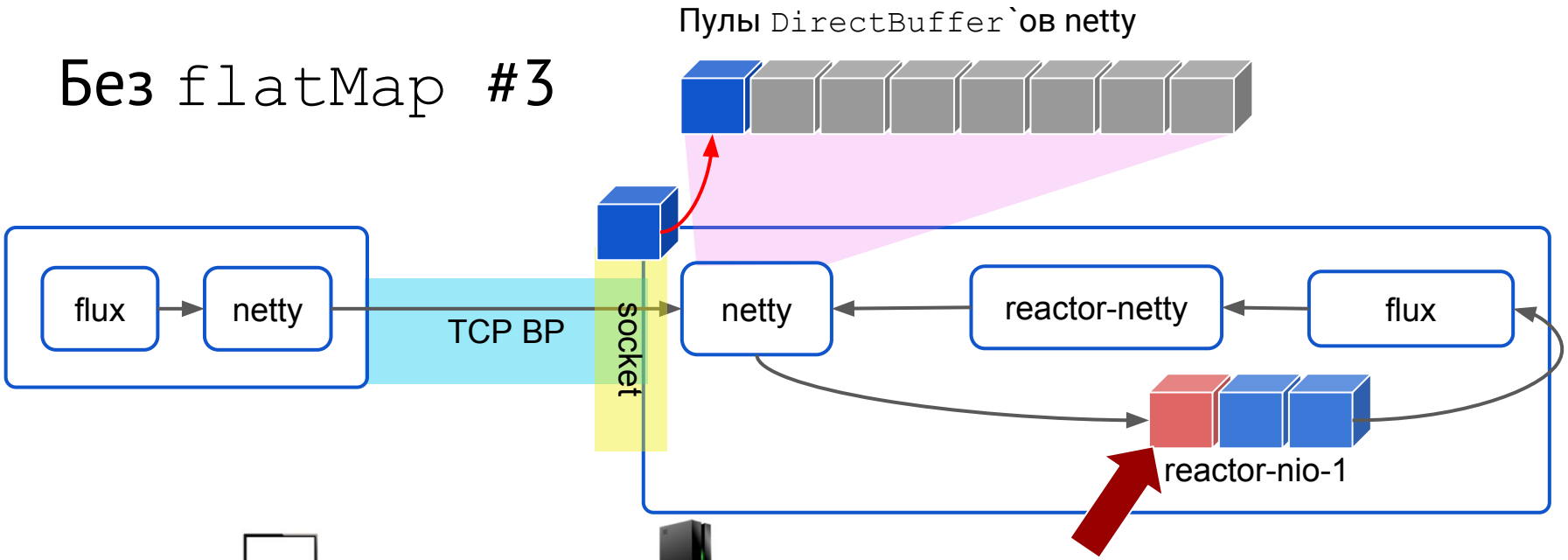
Без `flatMap` #2

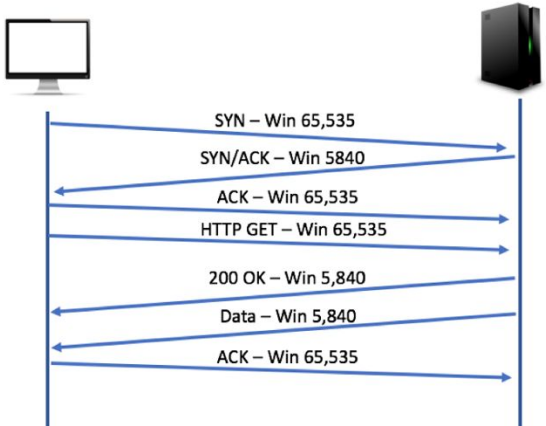




1. Копирует данные из системного сокета в `DirectBuffer`
2. Вызывает наш медленный метод `decoder.decode` `LetterController.java`

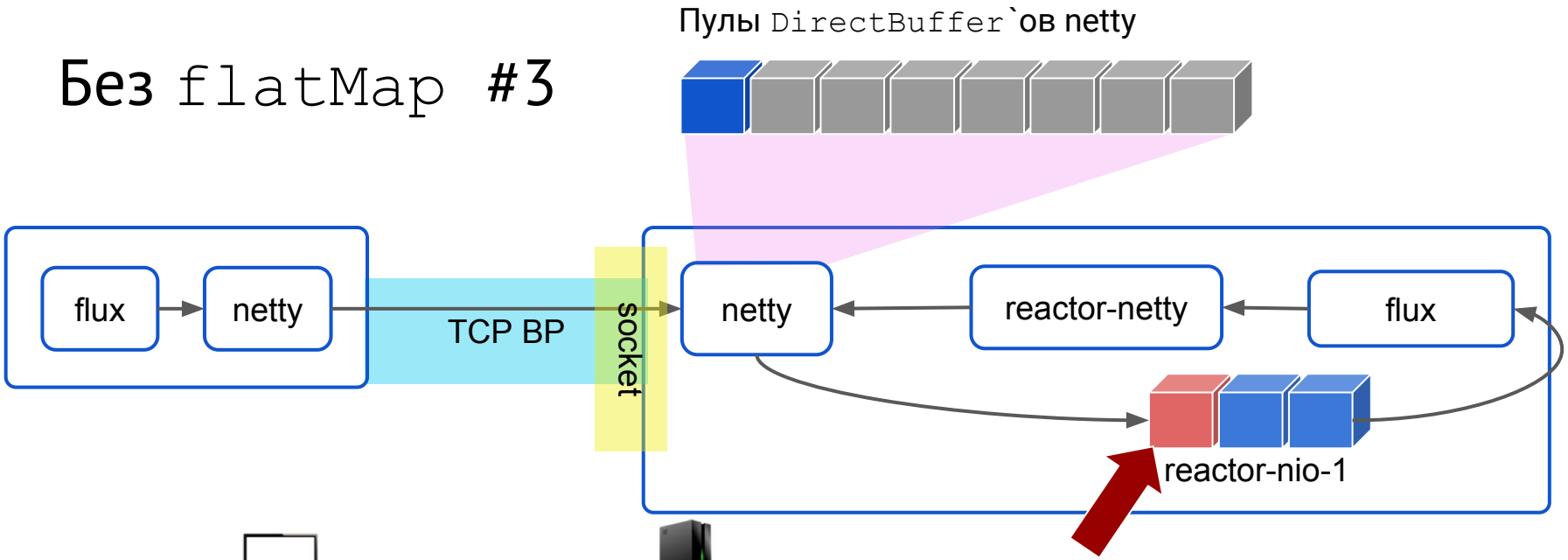
Без `flatMap` #3

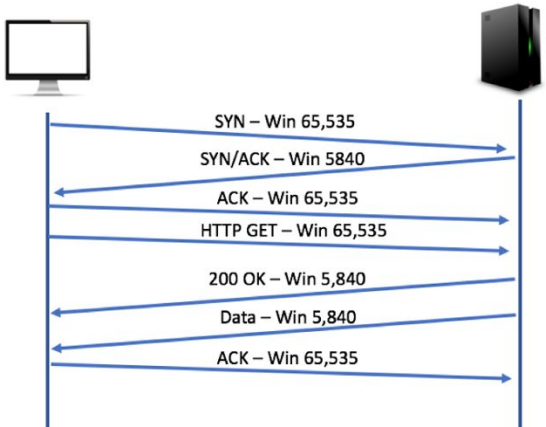




1. Копирует данные из системного сокета в `DirectBuffer`
2. Вызывает наш медленный метод `decoder.decode` `LetterController.java`
3. Копирует данные из системного сокета в `DirectBuffer`

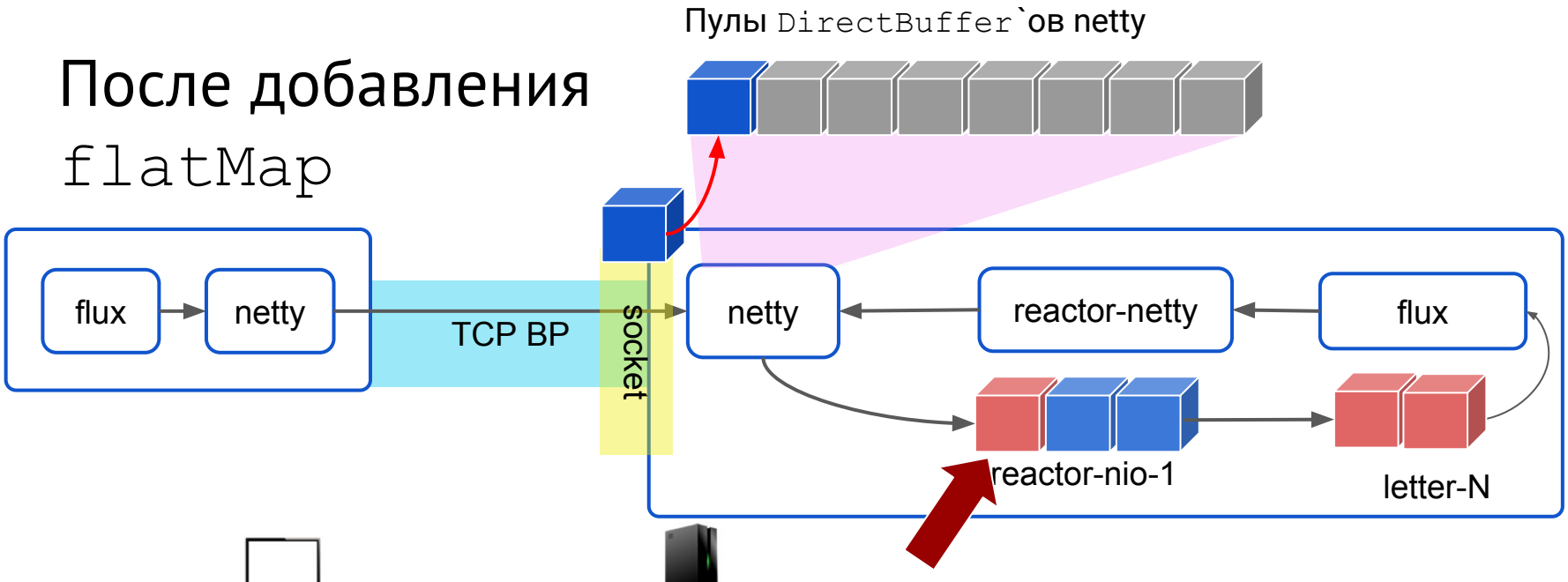
Без `flatMap` #3

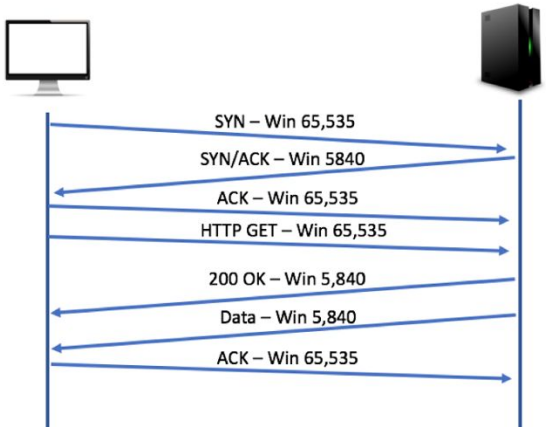




1. Копирует данные из системного сокета в `DirectBuffer`
2. Вызывает наш медленный метод `decoder.decode` `LetterController.java`
3. Копирует данные из системного сокета в `DirectBuffer`
4. `...` и так повторяется

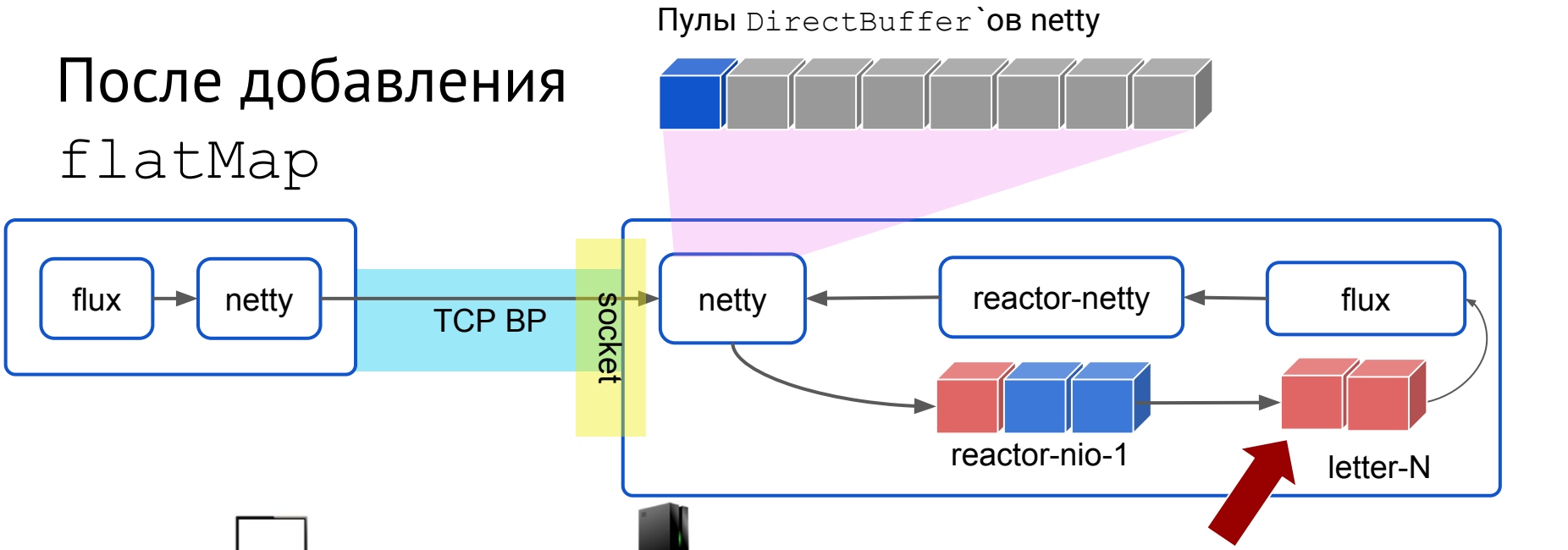
После добавления `flatMap`

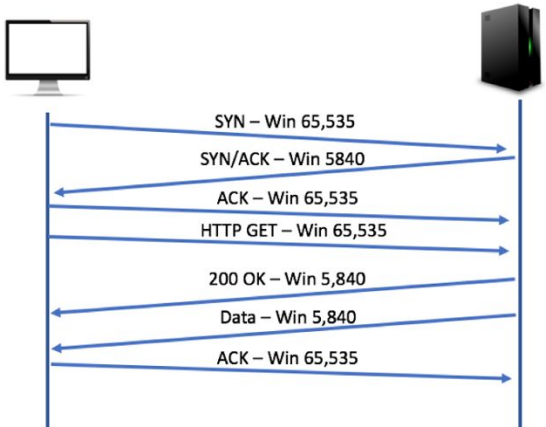




1. Копирует данные из системного сокета в **[1ms]** `DirectBuffer`

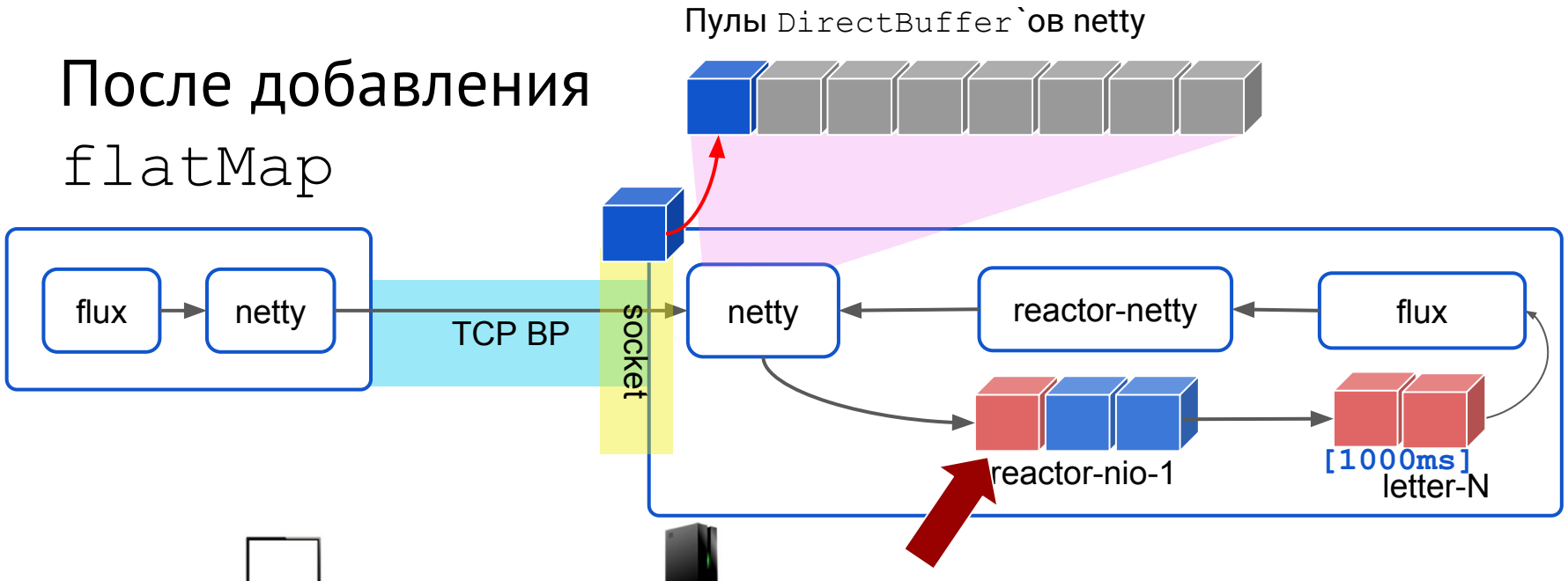
После добавления `flatMap`

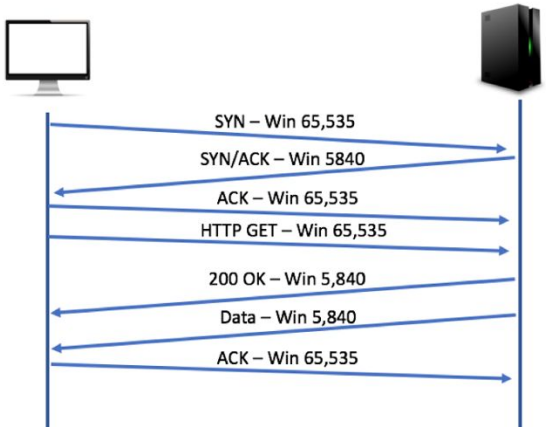




1. Вызывает наш медленный метод **[1000ms]**
   `decoder.decode` `LetterController.java`

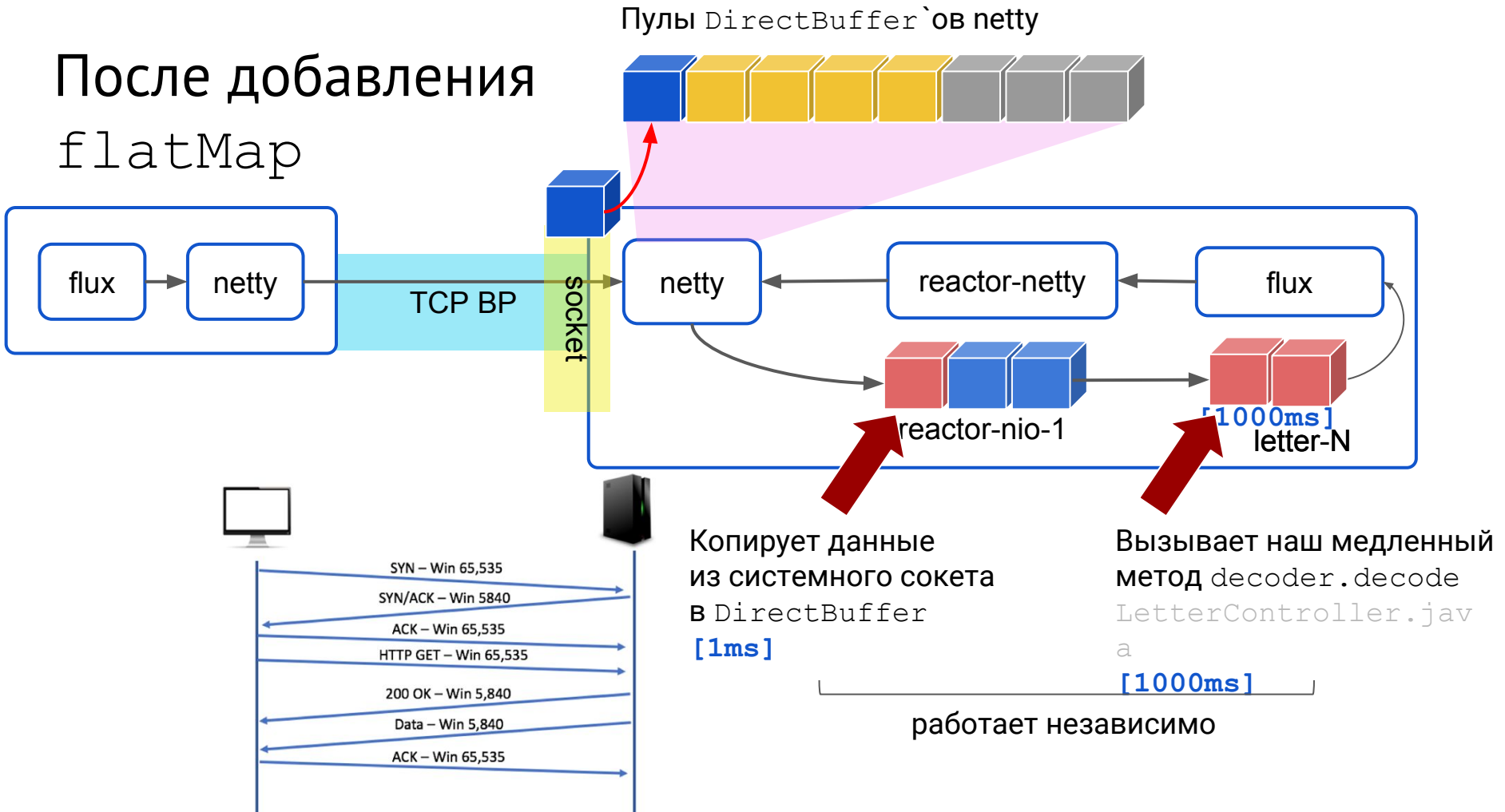
После добавления `flatMap`



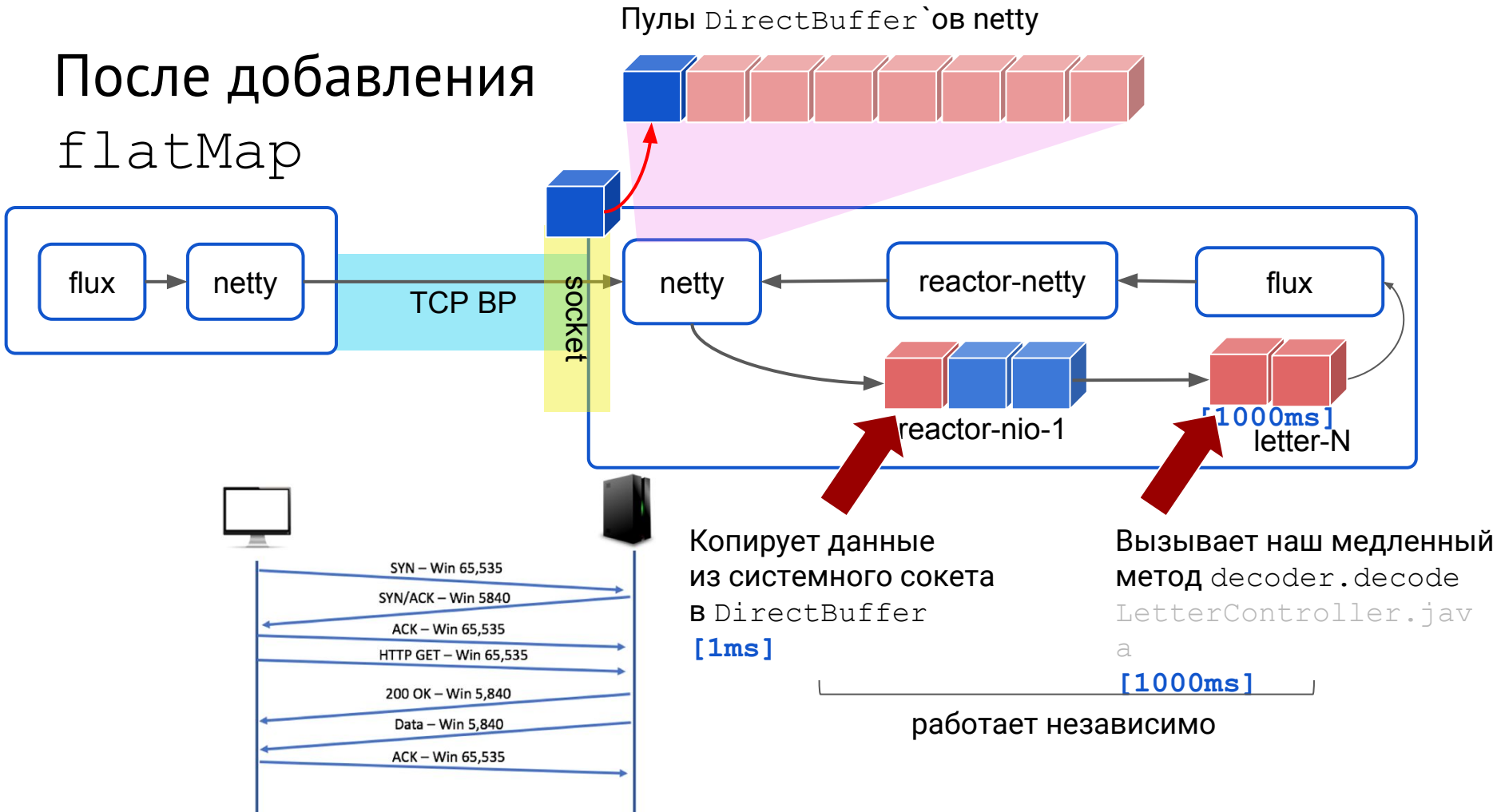


1. Копирует данные из системного сокета в `[1ms]` `DirectBuffer`

После добавления `flatMap`

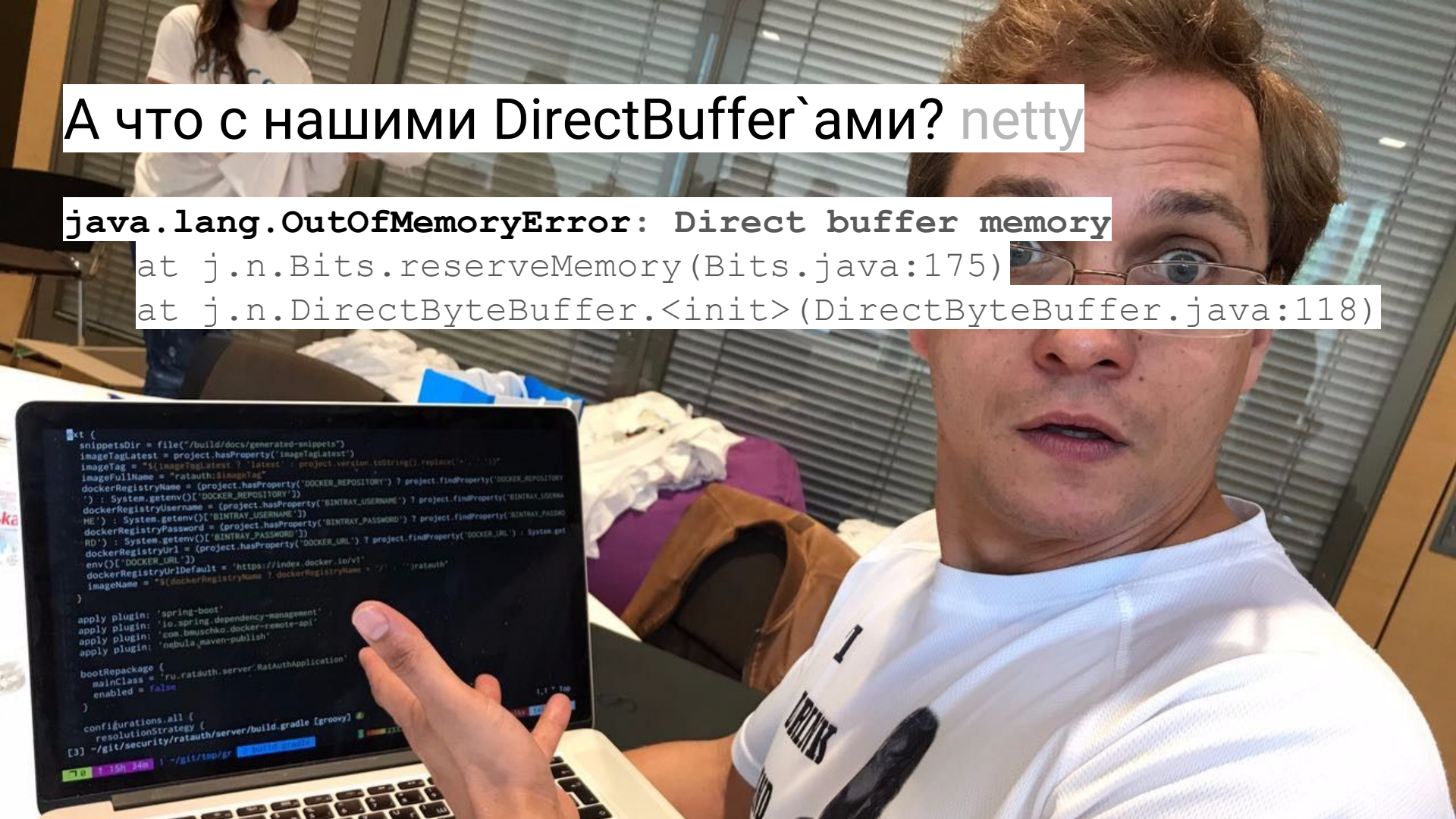


После добавления `flatMap`



А что с нашими DirectBuffer`ами? netty

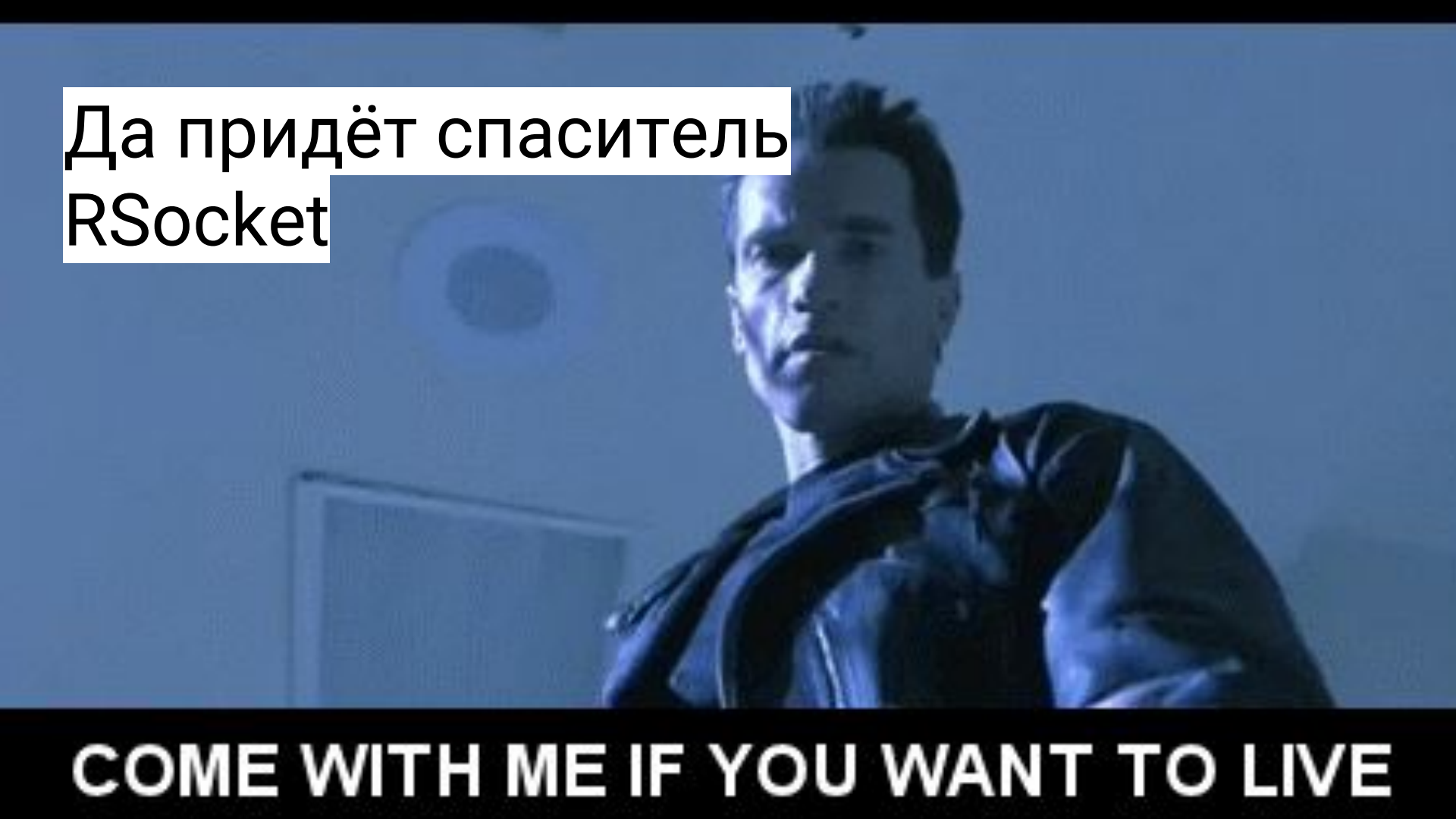
```
java.lang.OutOfMemoryError: Direct buffer memory
    at j.n.Bits.reserveMemory(Bits.java:175)
    at j.n.DirectByteBuffer.<init>(DirectByteBuffer.java:118)
```



У нас же есть Backpressure из коробки!

У нас же есть Backpressure из коробки!

Хочу Backpressure


Да придёт спаситель
RSocket
COME WITH ME IF YOU WANT TO LIVE

Кто такое RSocket

см https://jpoint.ru/talks/2xelfl2d9mgyweeeq60wiw/

Demo

Webflux #3

RSocket идеальное решение?

Можно было бы его продать так же как webflux :)

Но

1. Версия у него 0.11
2. Пока делали доклад нашли 2 бага. Спасибо Олегу, смогли обойти их
3. Привычные вещи не всегда делаются просто
4. И это только только то что мы успели найти ...

Выводы ложные

1. К чёрту велосипеды
2. К чёрту webflux
3. Да здравствует RSocket

Выводы ложные

1. К чёрту велосипеды
2. К чёрту webflux
3. Да здравствует RSocket

Сияющие технологии

1. В каких то задачах webflux будет хорош
2. В каких то задачах RSocket будет плох
3. Но своё решение всегда как минимум заставляет задуматься

Сияющие технологии

1. В каких то задачах webflux будет хорош
2. В каких то задачах RSocket будет плох
3. Но своё решение всегда как минимум заставляет задуматься

Выводы

1. Не всё то Reactive что с интегрировано с Project Reactor
    a. Поддержка Backpressure на основе TCP BP имеет сайд эффекты
    b. Control flow между сервисами для реактивного пайплайна так же важен
2. Webflux не подходит для балансировки скорости между сервисами
3. С RSocket честный backpressure, пока сырой, но перспективный
4. 

Настоящие выводы

Не стоит внедрять технологии на основе авторитета:

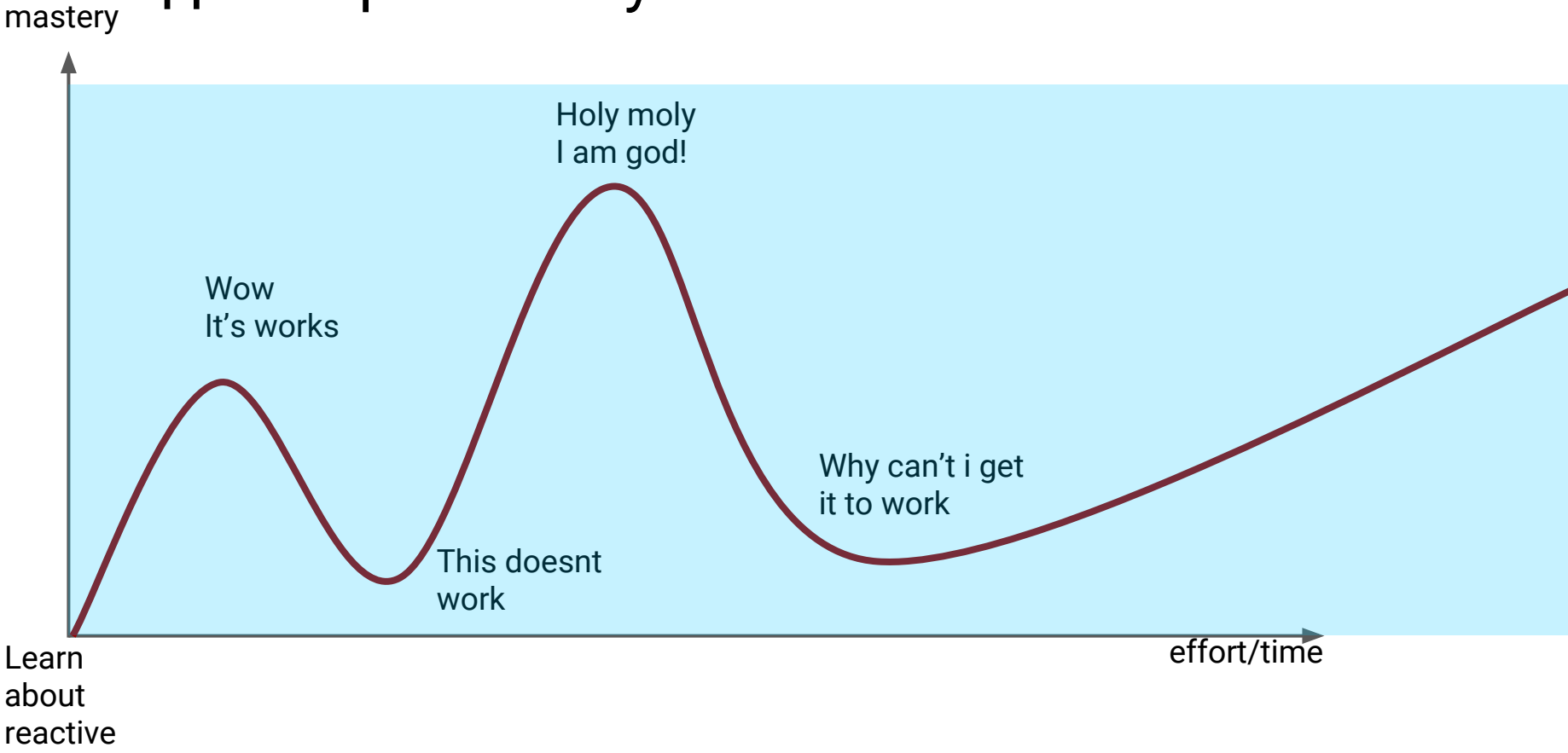
1. спикера
2. компании
3. Тренда

Webflux хорош когда нужно сэкономить коннекшены и отправить пачку данных

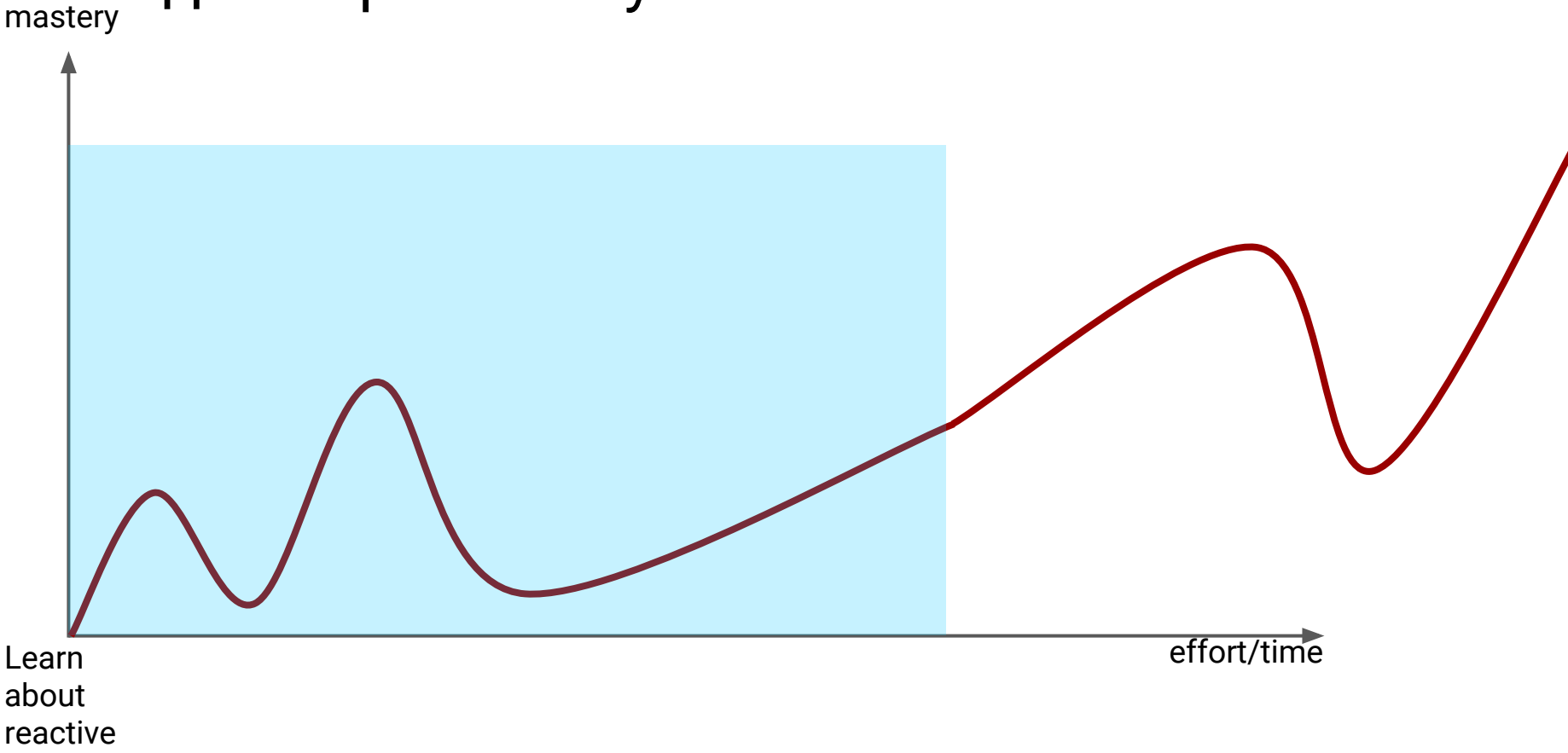
RSocket хорош во всех наших кейсах

Свой велосипед всегда поняитнее

Выводы – кривая обучения reactive



Выводы – кривая обучения reactive



Вопросы

@tolkv

@lavcraft

@jekaborisov

@jeka1978

Вопросы

ЦИАН

РАЗБОР ПОЛЁТОВ
#RAZBORPOLETOV
ПОДКАСТ
Добрый день добрый вечер
Доброе утро доброй ночи

alfalab

NAYA
TECHNOLOGIES

alfalab

Ссылки

Код с демо

1. https://github.com/lavcraft/spring-react-or-not-react-jpoint2019

Spring Webflux Streaming

1. Json streaming
2. Jackson Smile

Spring WebClient

1. WebClient Java 11 HttpClient Support
2. Webflux and RSocket backpressure