

Spring

Reactive Or Not Reactive




1984

 @tolkv

 @lavcraft



@jekaborisov 

@jeka1978 



 alfalab



 alfalab

Spring
Reactive
Or Not Reactive





Задача



Задача



Задача



Задача

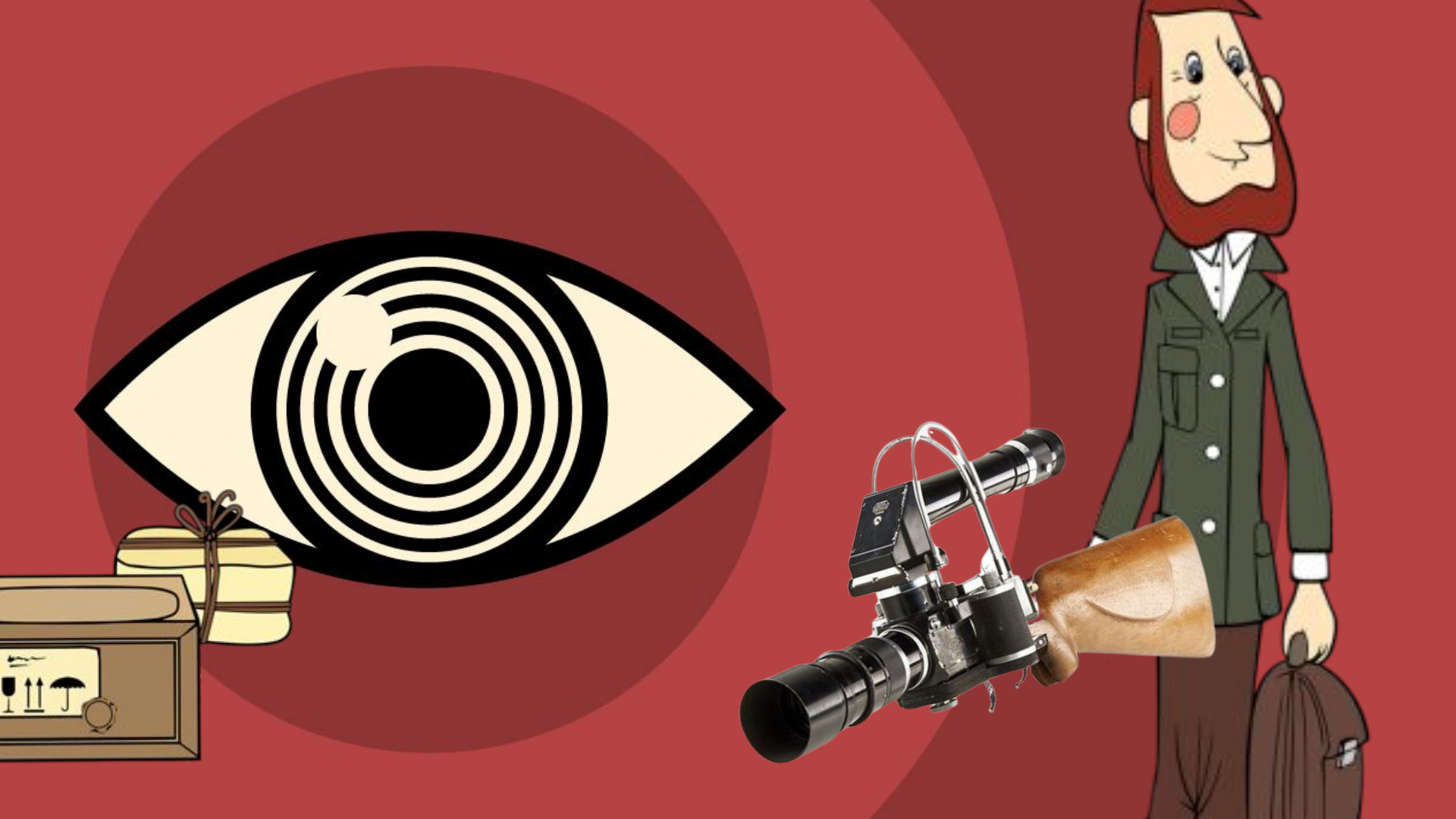


Задача



Имя: Папа
Фамилия: Папа мальчика
Должность: Академик





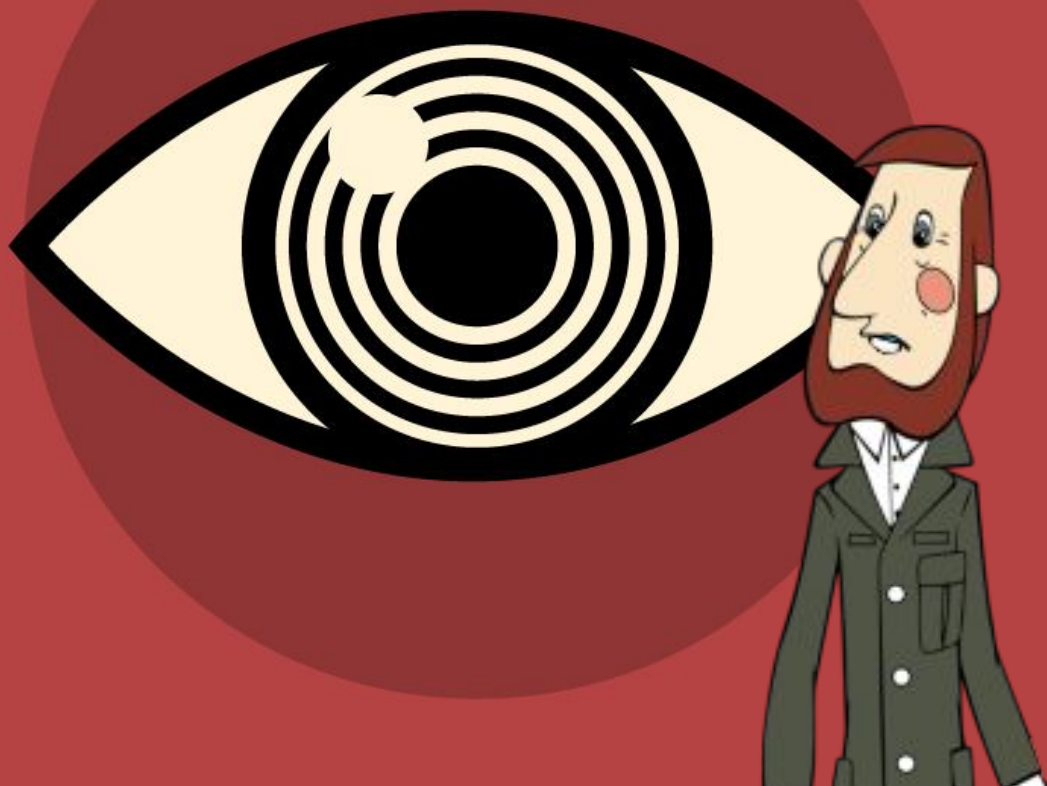


Задача

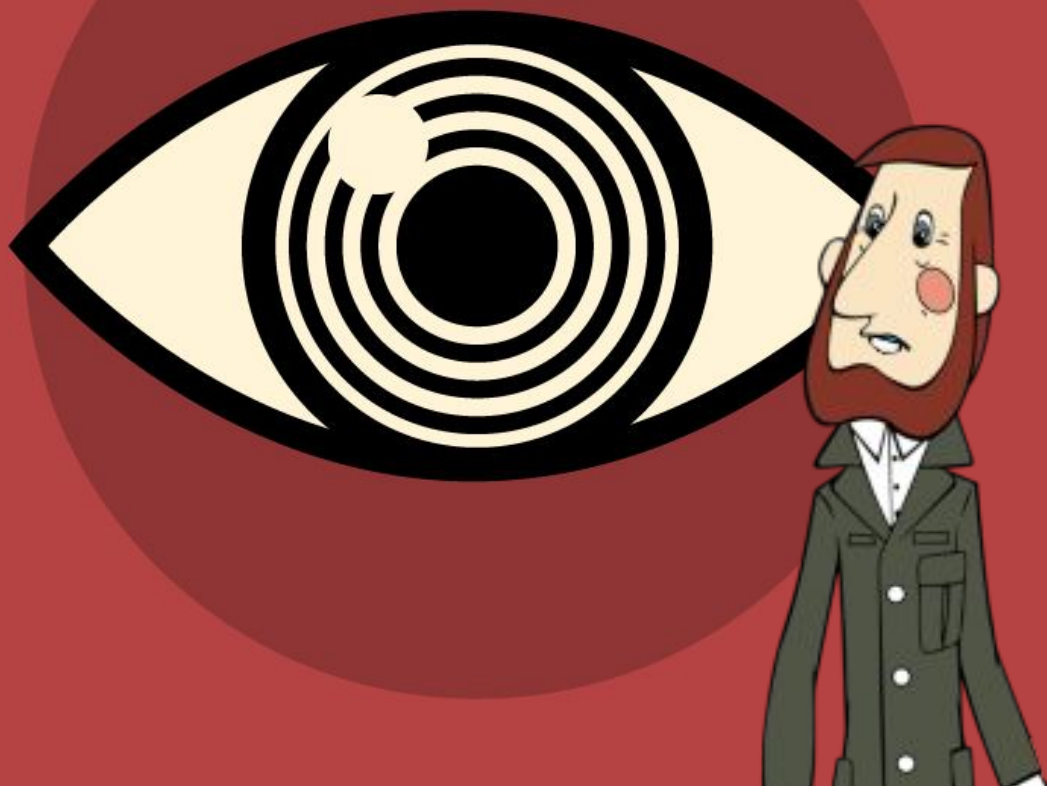


Имя: Папа
Фамилия: Папа мальчика
Должность: Академик









Архитектура проекта «Big Brother»



Перехватчик писем,
отсылает все письма на распознавание

Расшифровщик писем,
Выясняет чье письмо и отправляет
в Правоохранительные органы



Правоохранительные органы,
принимают решение об аресте

ЗАПРЕТИТЬ! УЖЕСТОЧИТЬ!



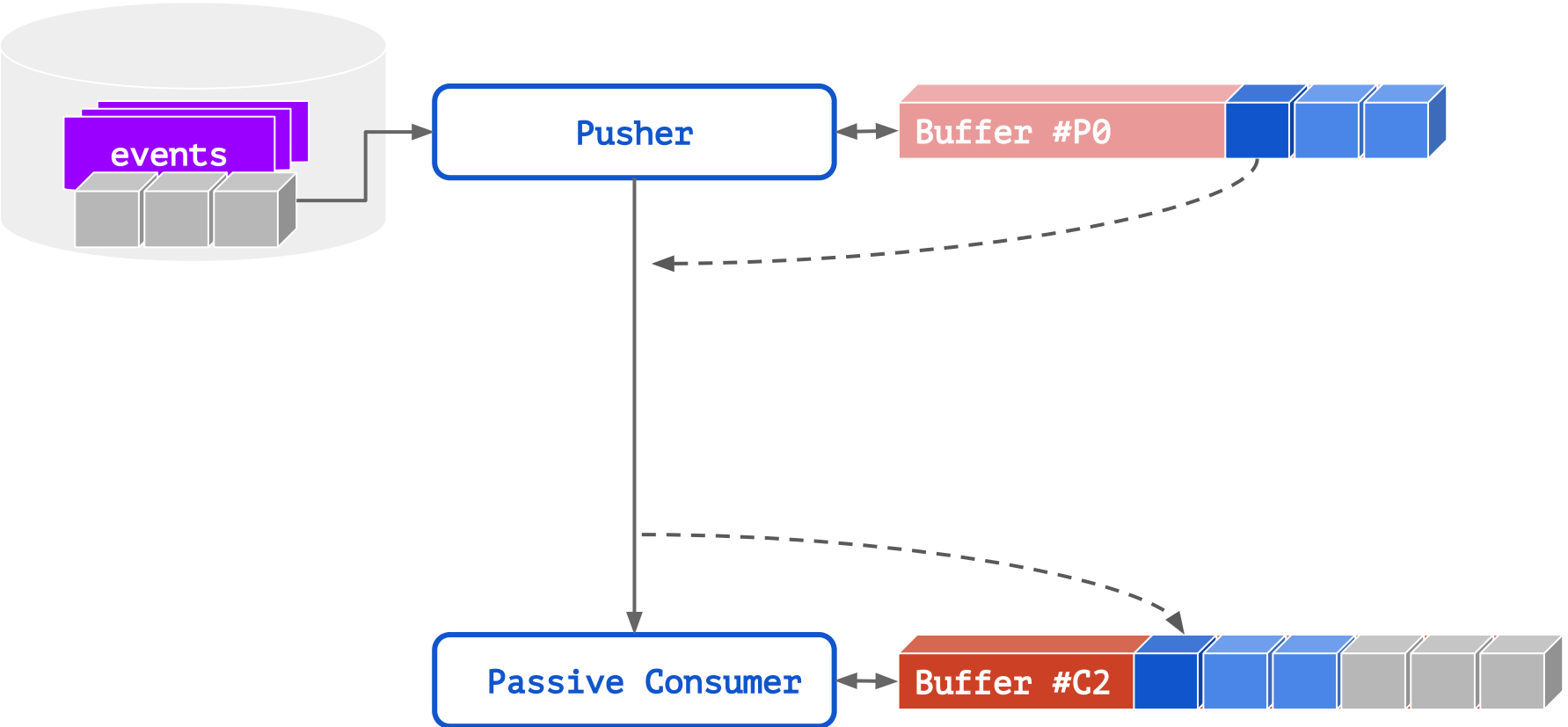
НЕ ПУЩАТЬ!



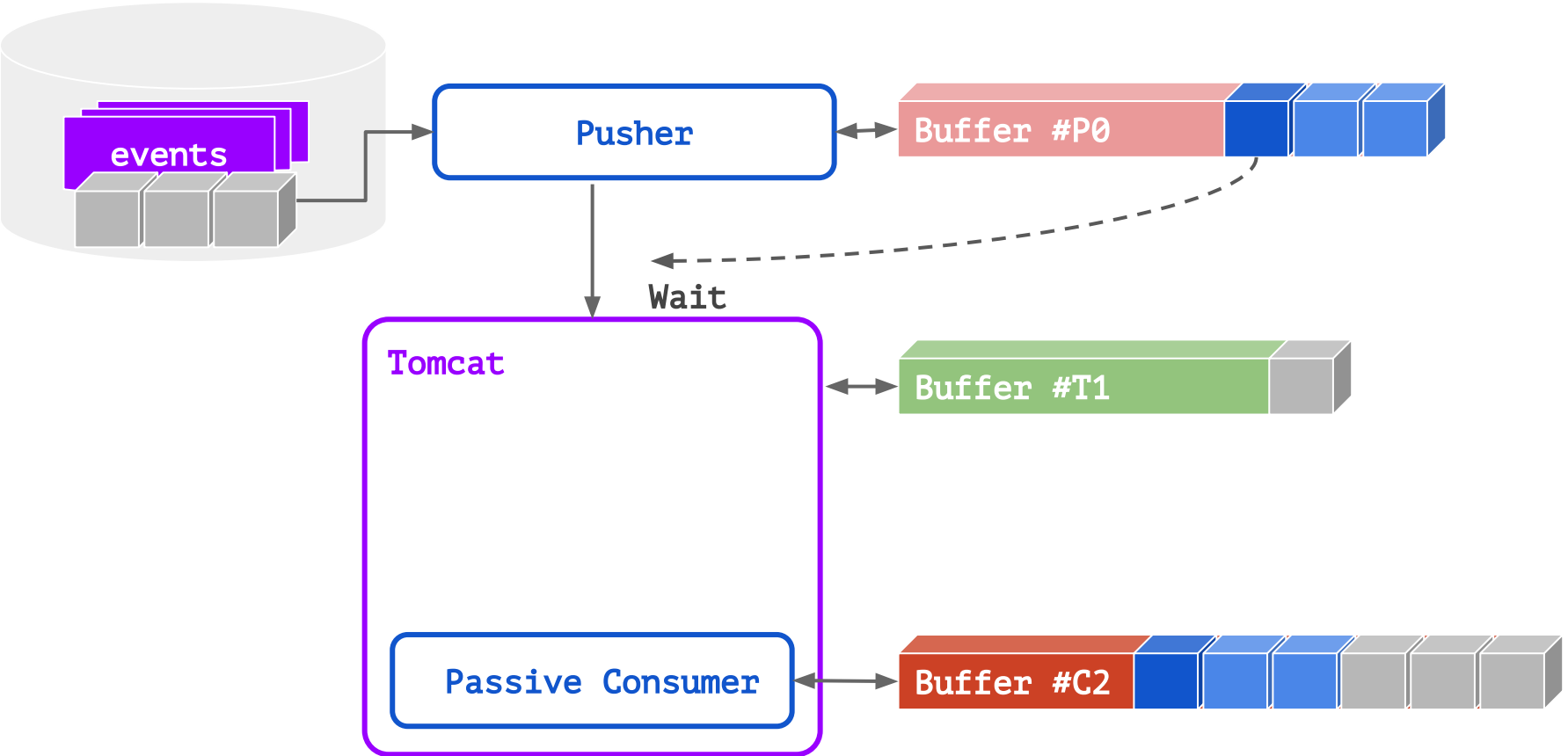
Demo

Наивная имплементация #0

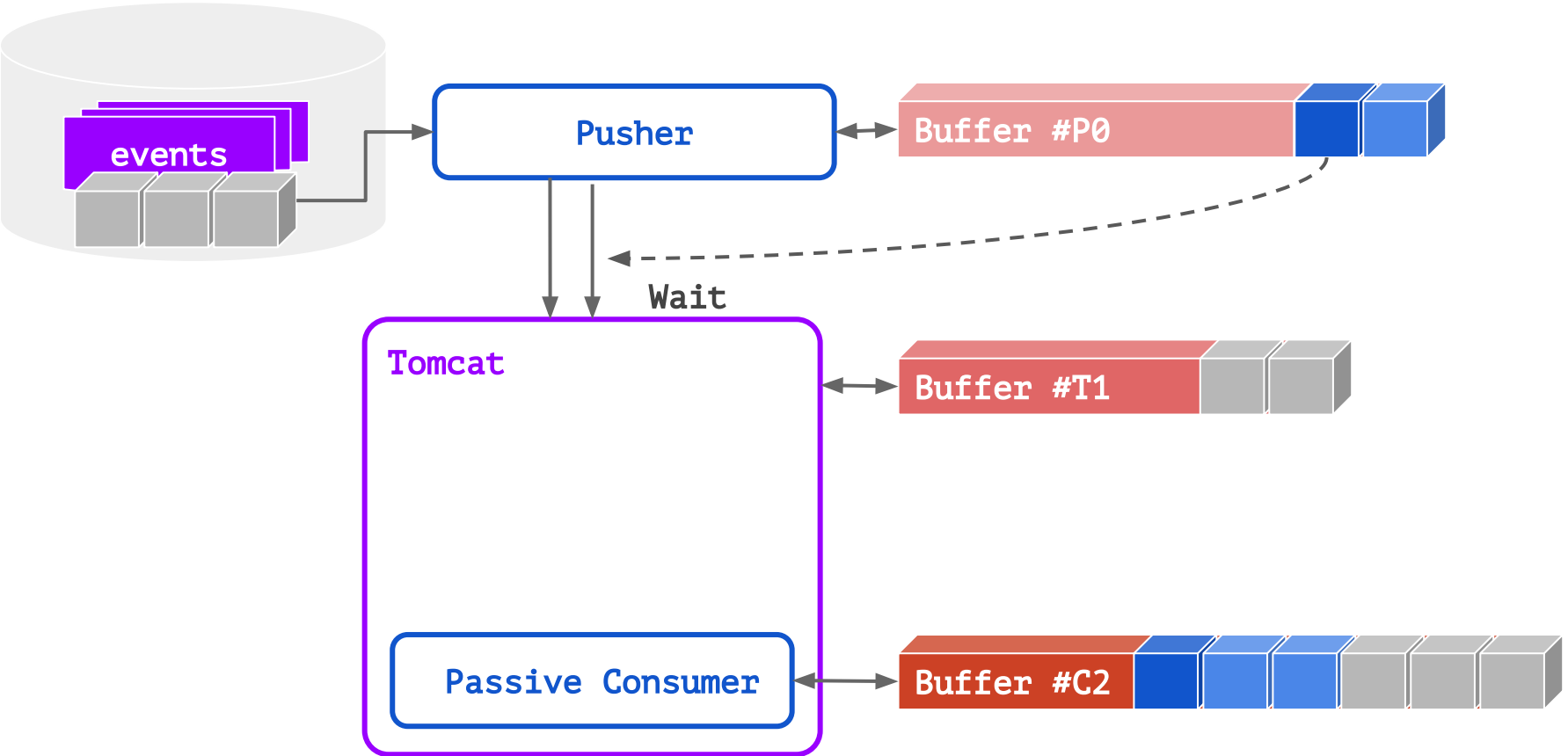
Push and Pull semantics related to consumer



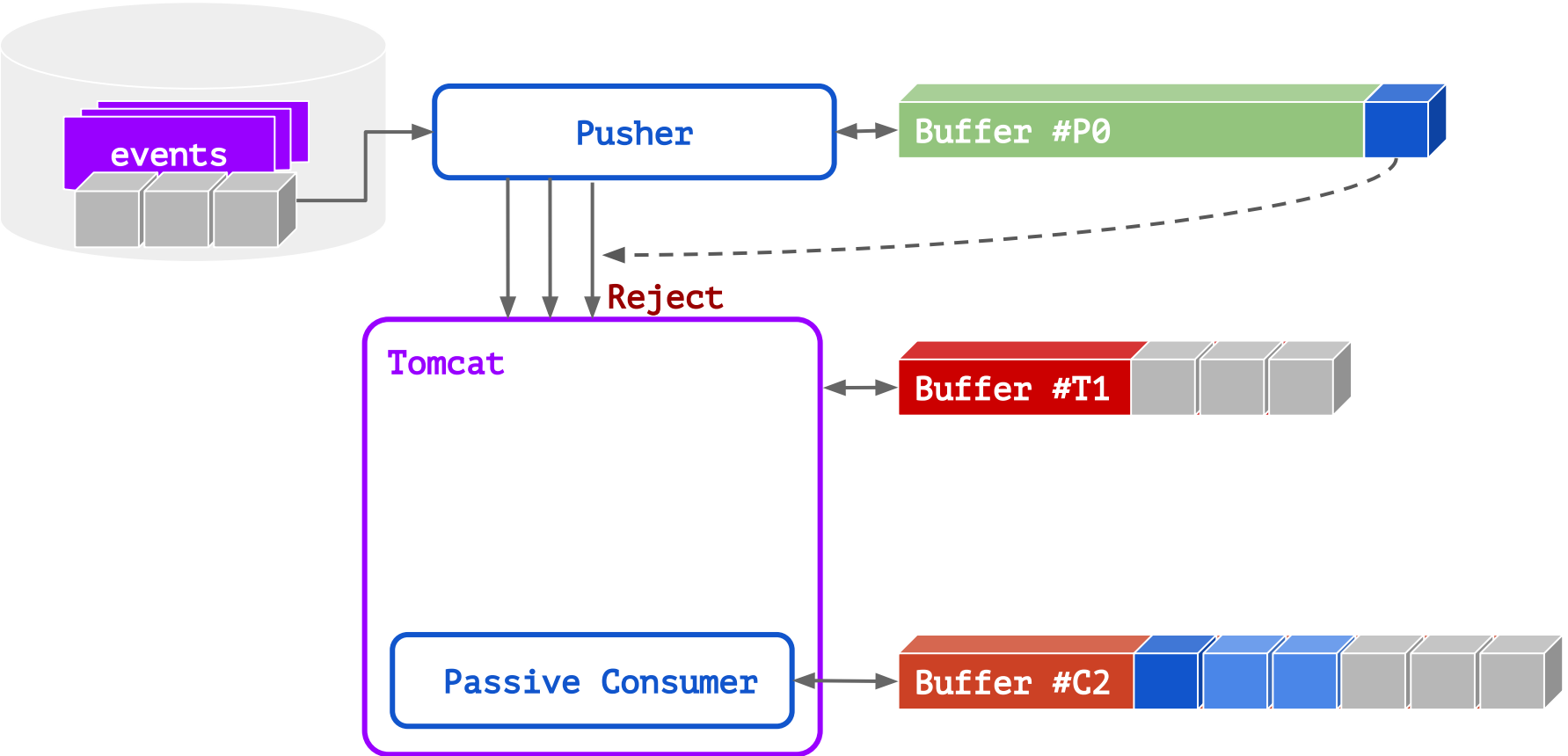
Push and Pull semantics related to consumer



Push and Pull semantics related to consumer



Push and Pull semantics related to consumer





Демо

Наивная имплементация #1







О пропускной способности



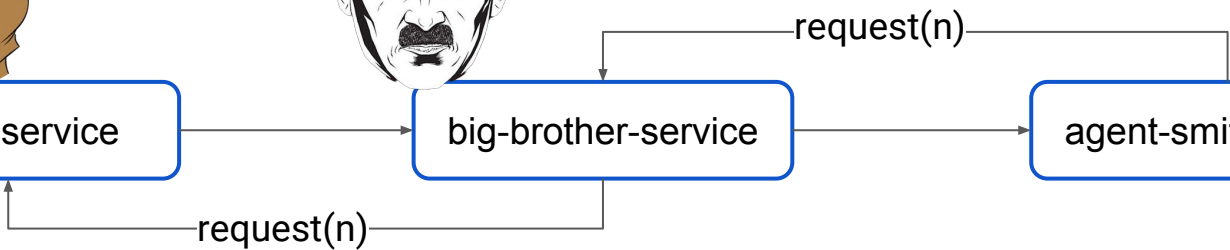
pechkin-service



big-brother-service



agent-smith-service



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



pechkin-service



big-brother-service



agent-smith-service

request(n)

request(n)

Buffer #P0

1t

∞

Buffer #P1

8t

8rps

Buffer #P2

2t

2rps

A man with dark hair and glasses, wearing a blue shirt, is looking intently at a computer monitor. The monitor displays a close-up of a person's face. In the foreground, a vintage-style keyboard with yellow keys is visible. The scene is dimly lit, suggesting a control room or office environment.

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

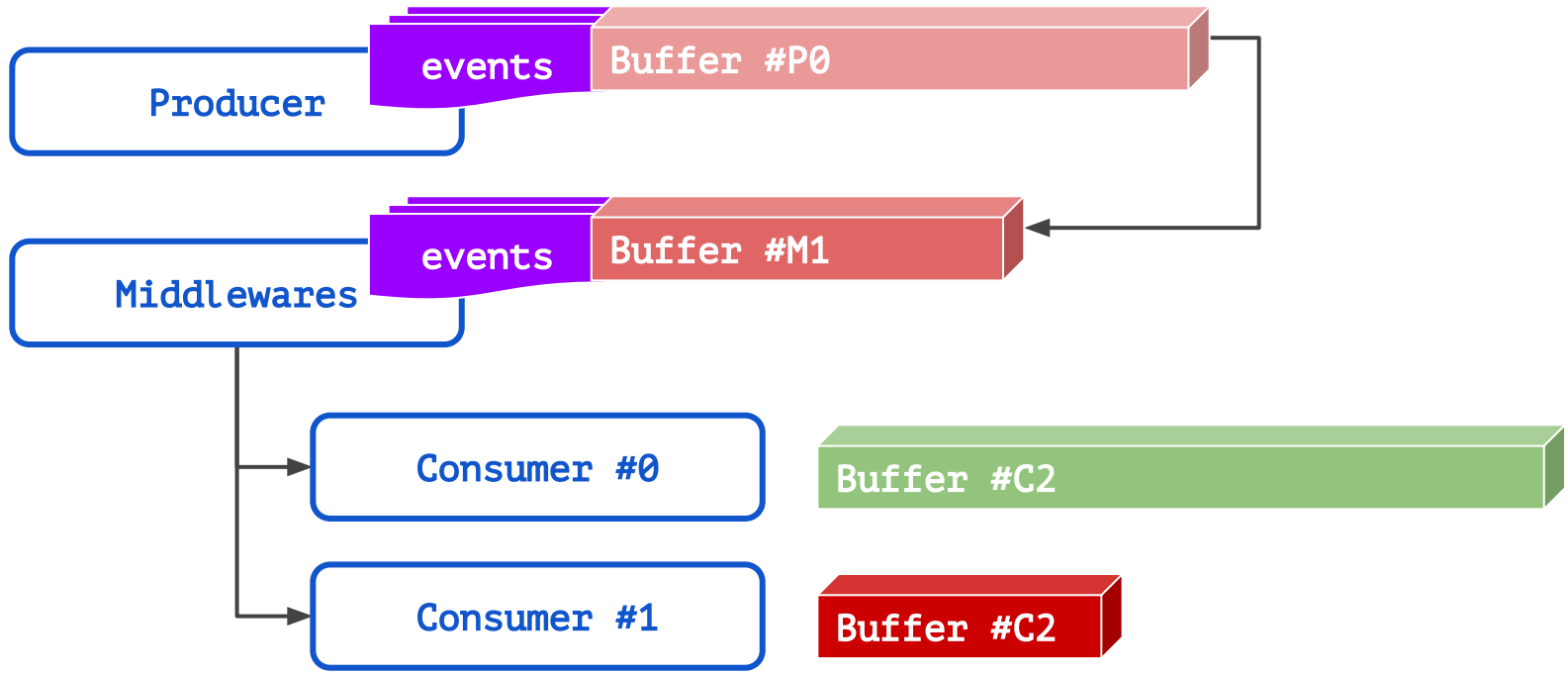


Нет проблемы

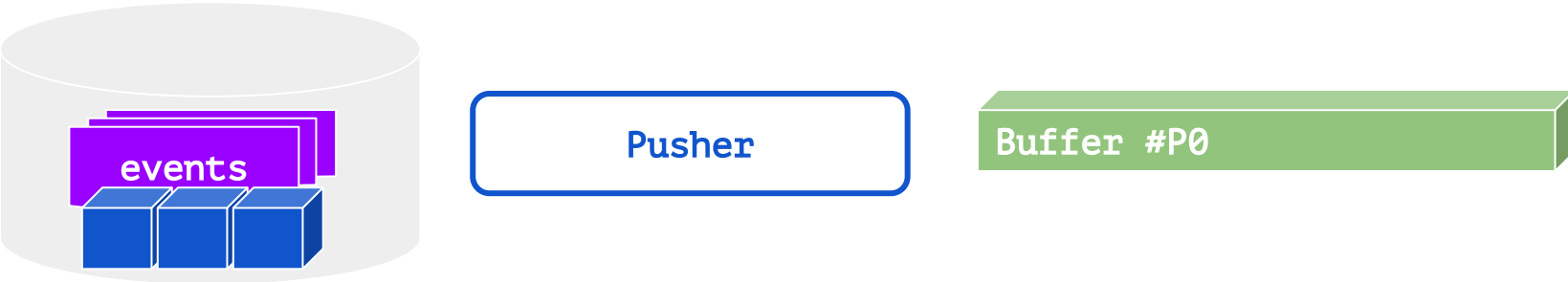
В продюсер текут данные



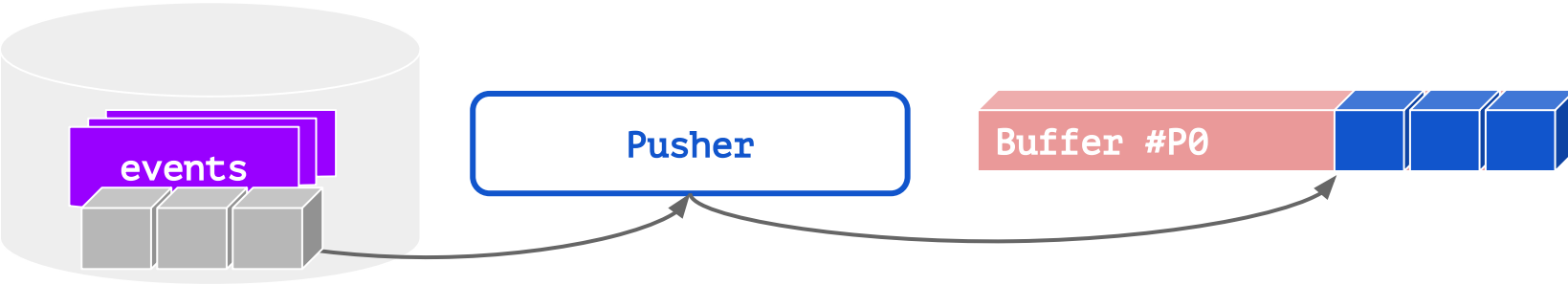
Всё что не успеваем
отослать, суем в
очередь, когда нет
места в очереди,
дропаем, то что
менее важно



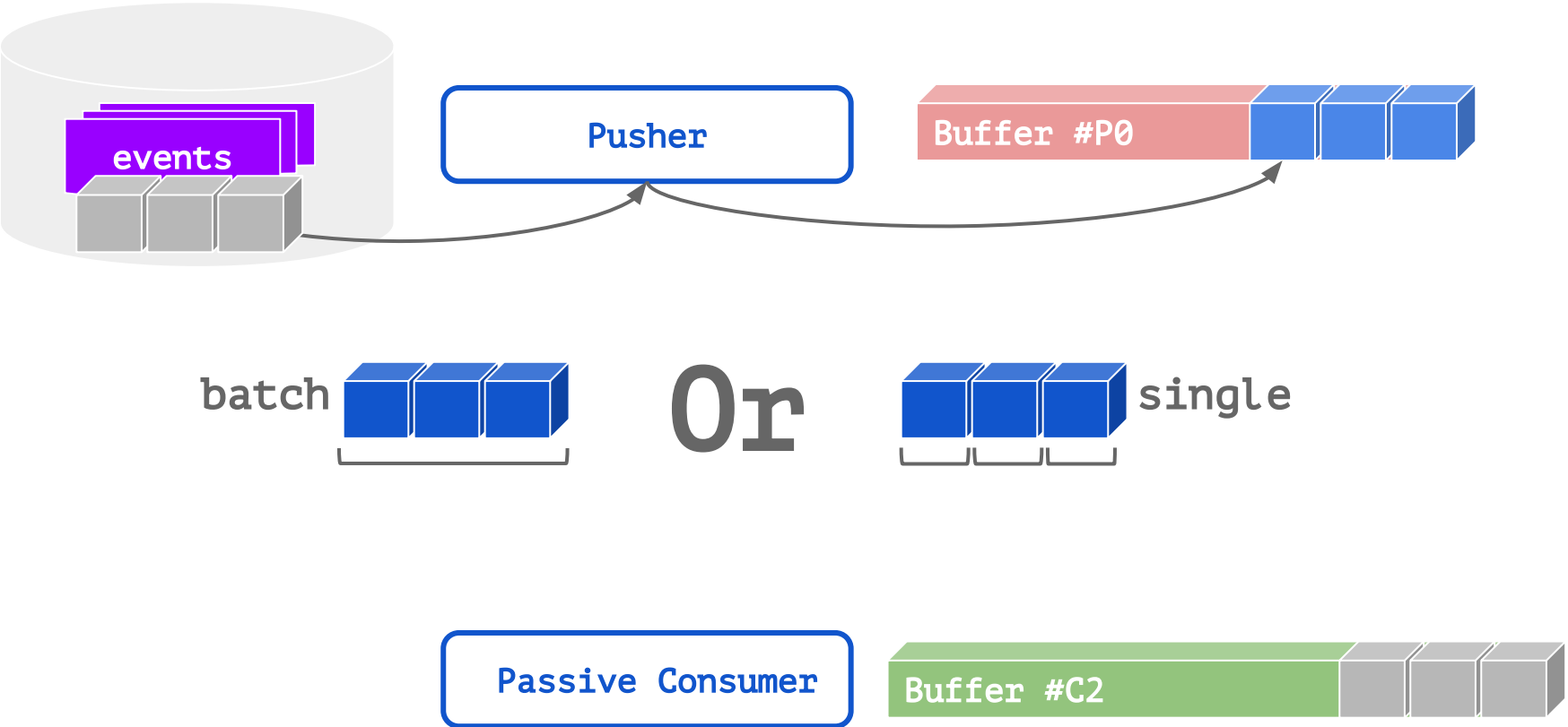
Push and Pull semantics related to consumer



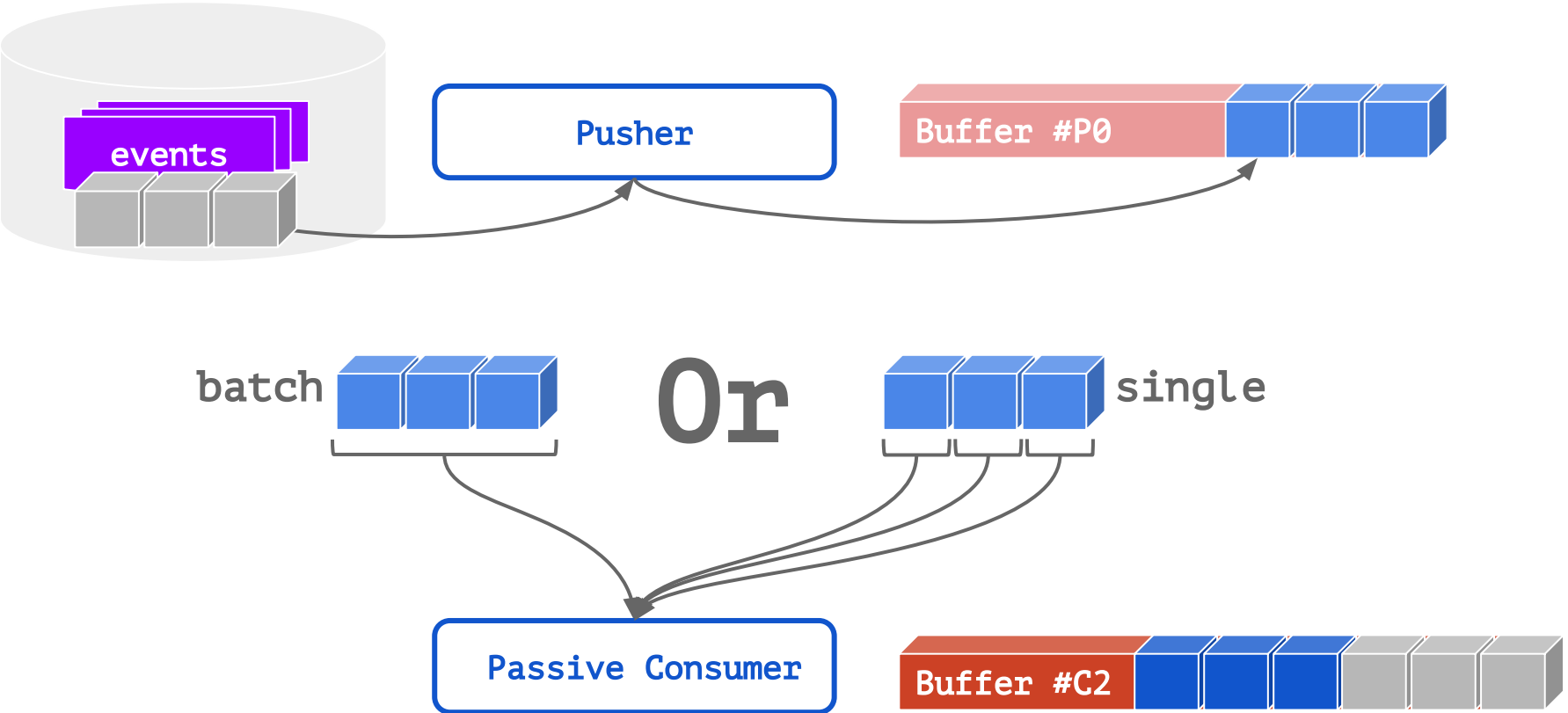
Push and Pull semantics related to consumer



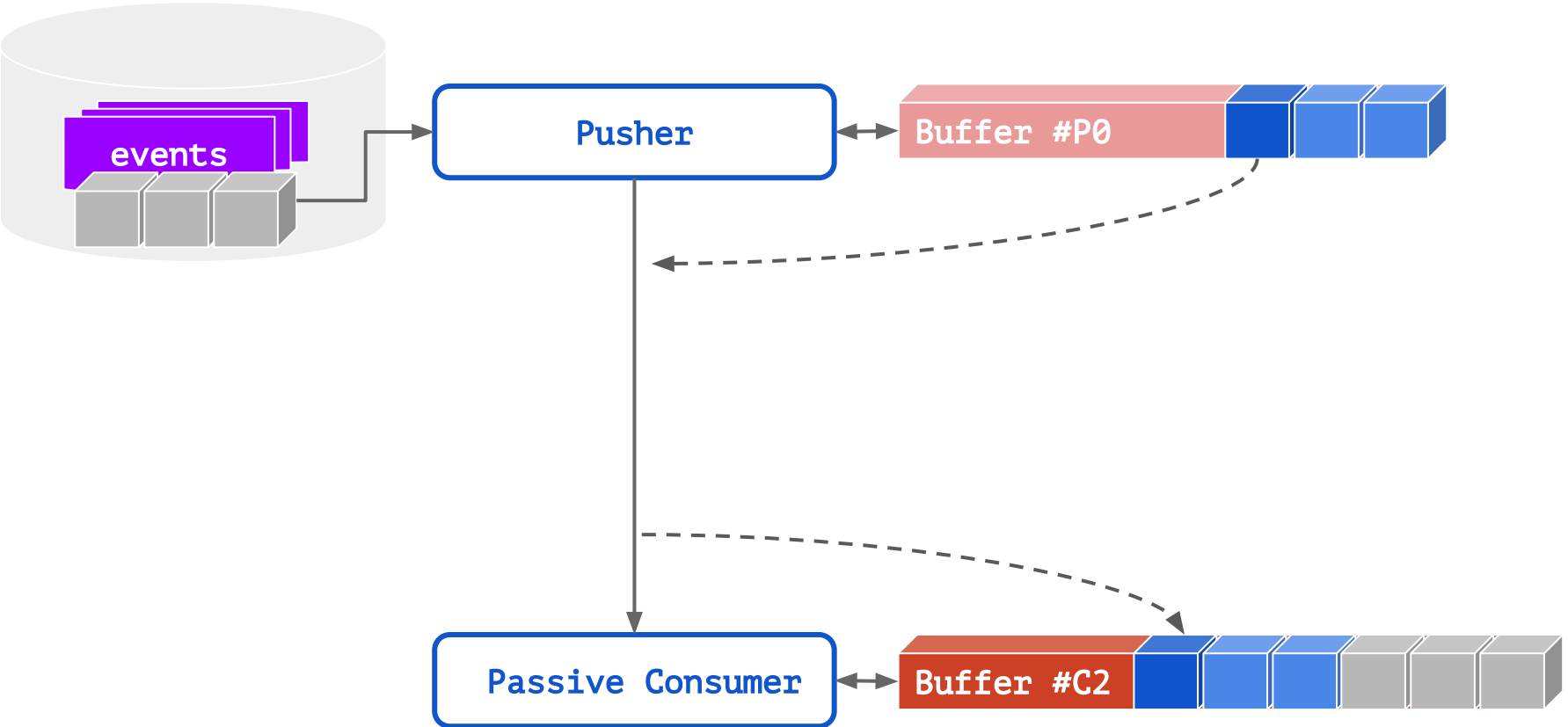
Push and Pull semantics related to consumer



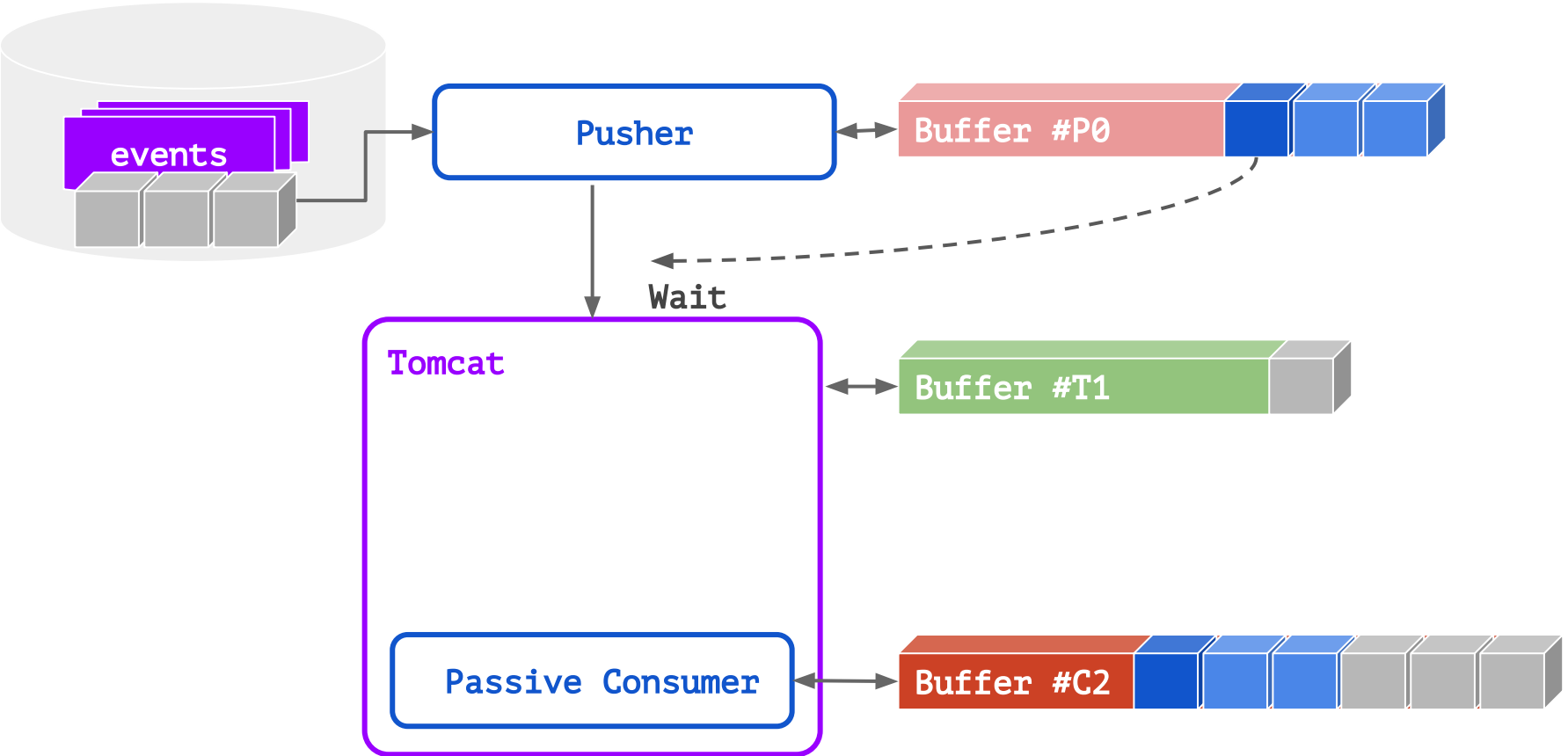
Push and Pull semantics related to consumer



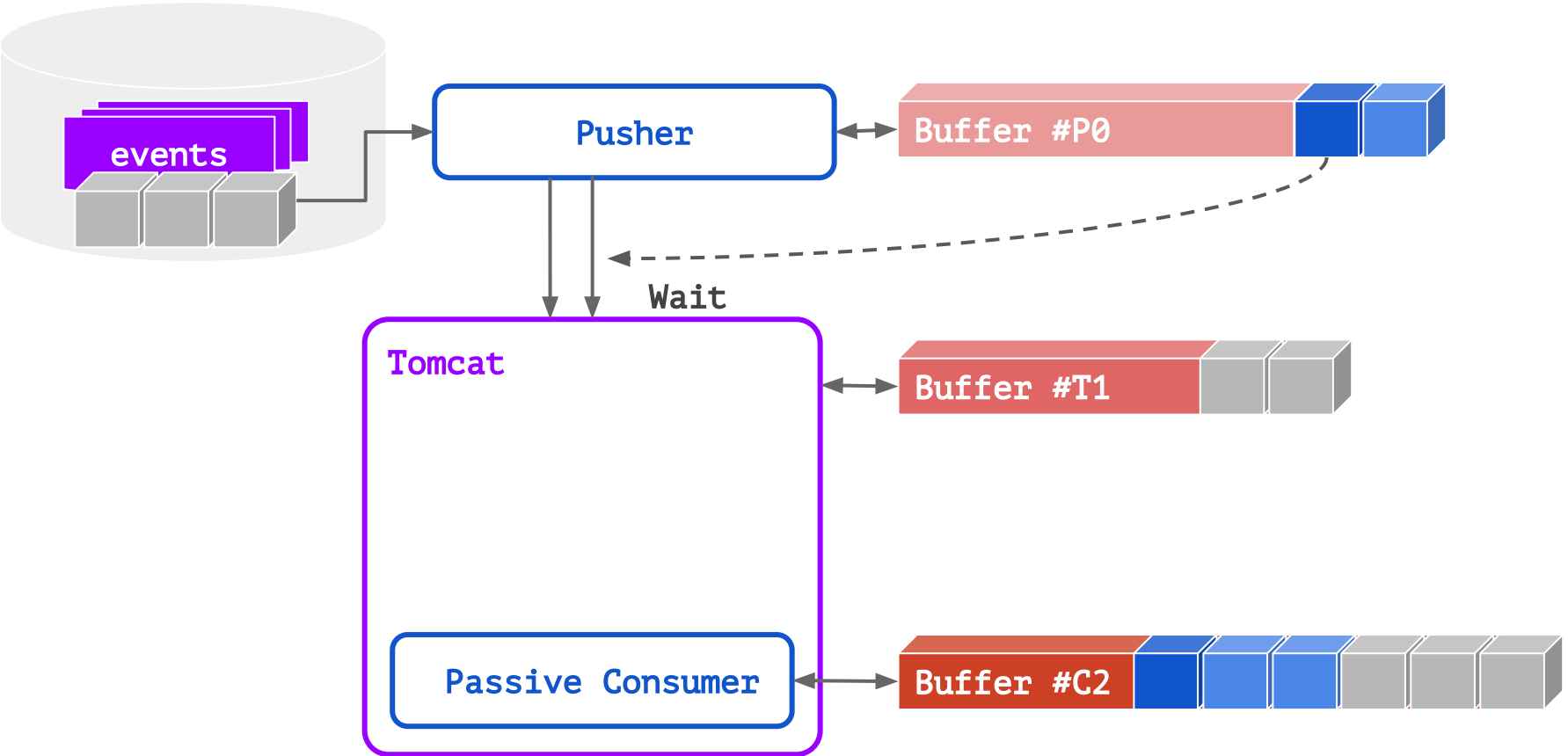
Push and Pull semantics related to consumer



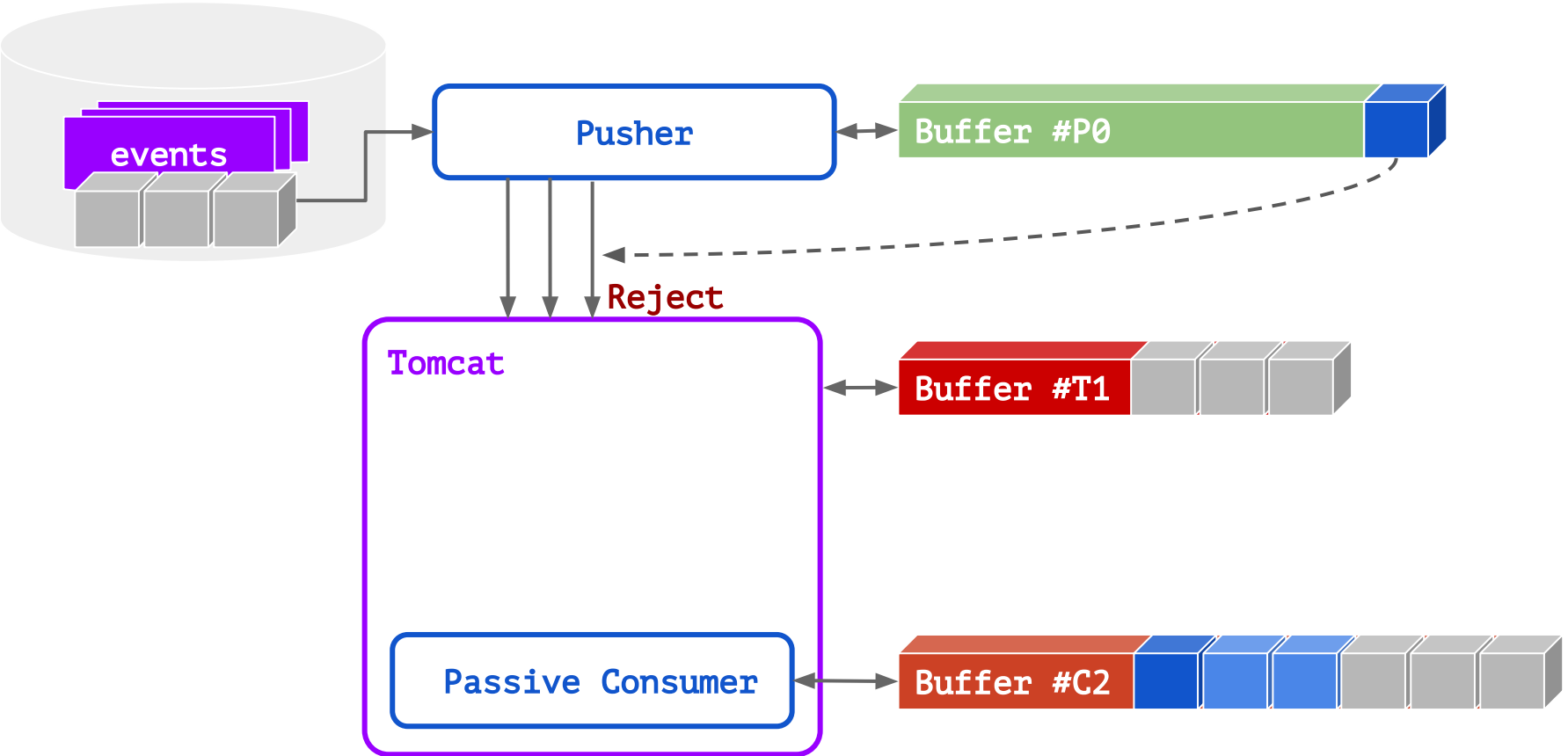
Push and Pull semantics related to consumer



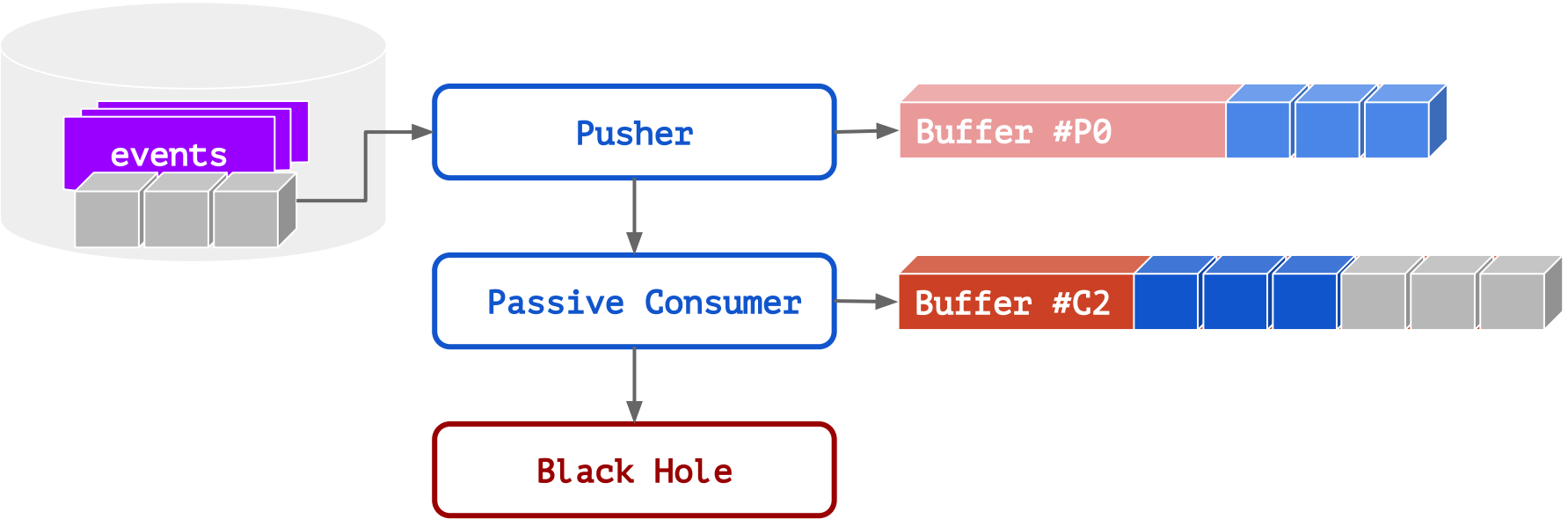
Push and Pull semantics related to consumer



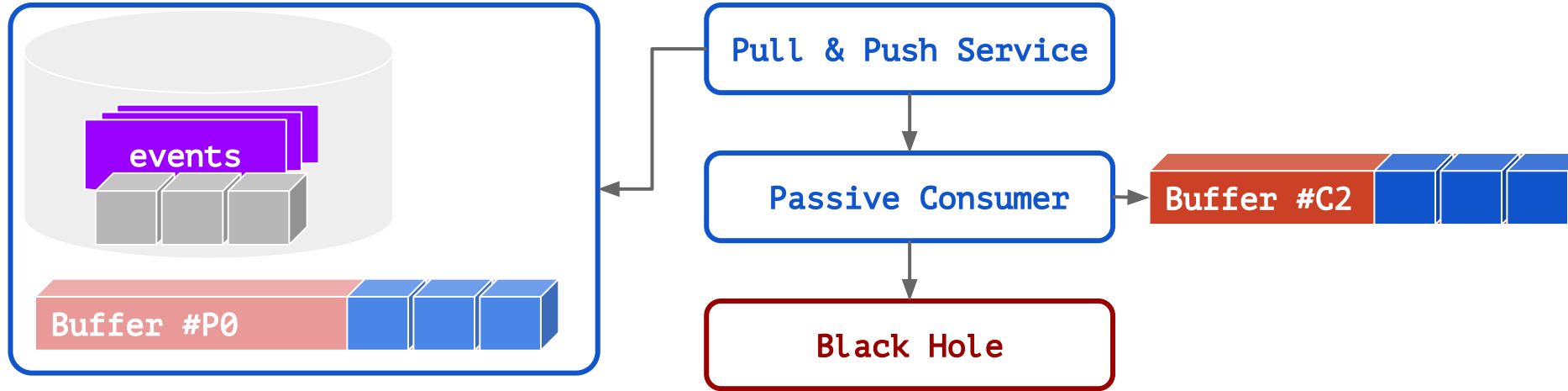
Push and Pull semantics related to consumer



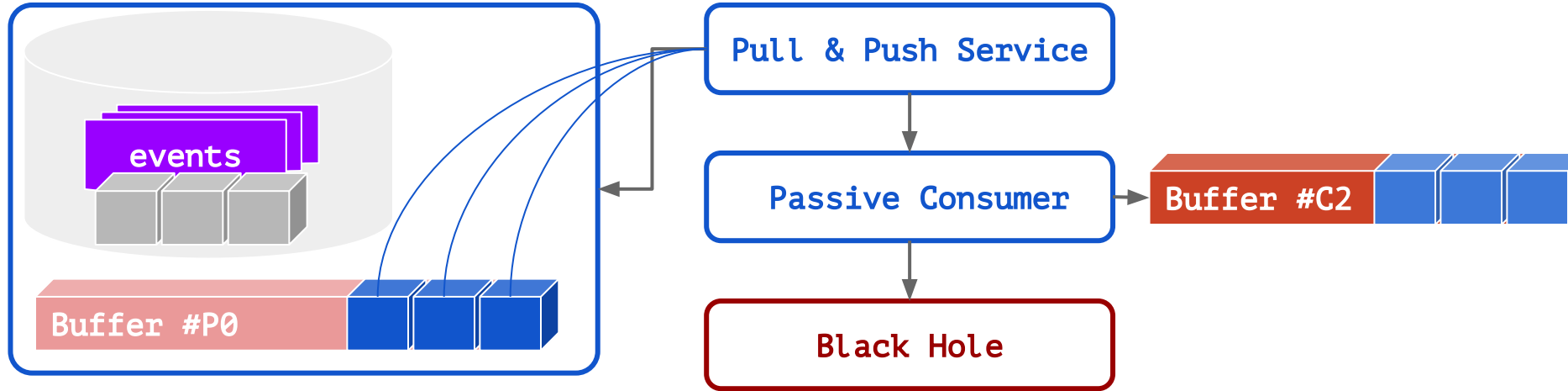
Push and Pull semantics related to consumer



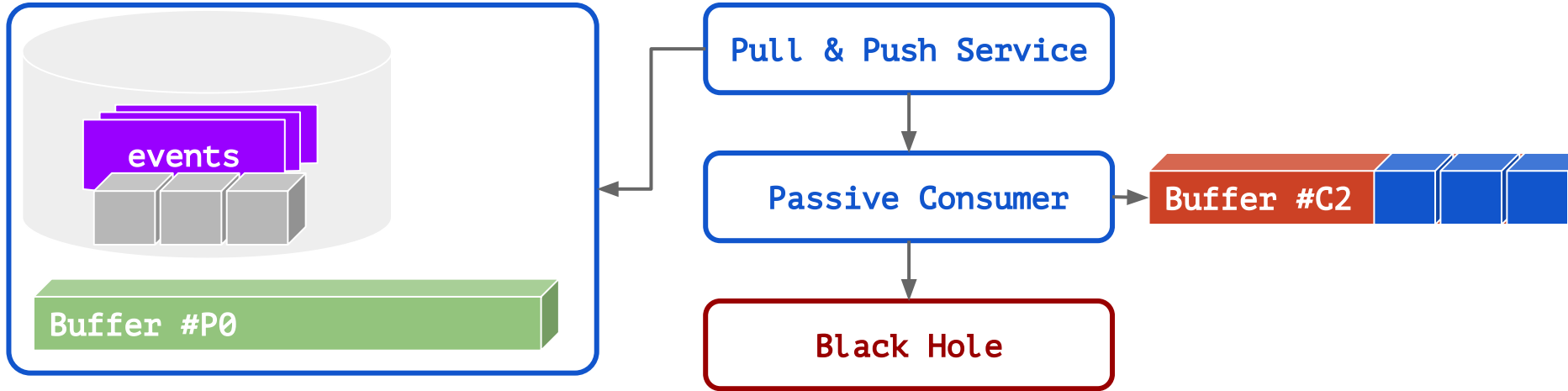
Push and **Pull** semantics related to consumer



Push and **Pull** semantics related to consumer



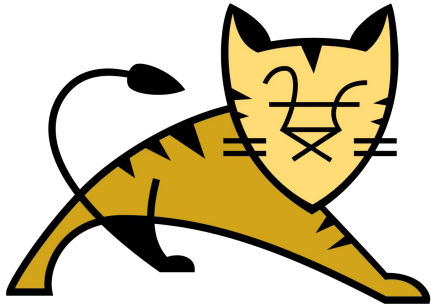
Push and **Pull** semantics related to consumer



Webflux



Spring Webflux & Project Reactor



jetty://



undertow



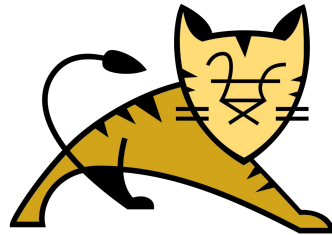
Netty

Spring Webflux & Project Reactor

Spring MVC (Servlet 3.1+)

or

Spring Webflux (default)



jetty://



undertow



Netty

Подключаем webflux

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

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

WebClient is a new RestTemplate

Было

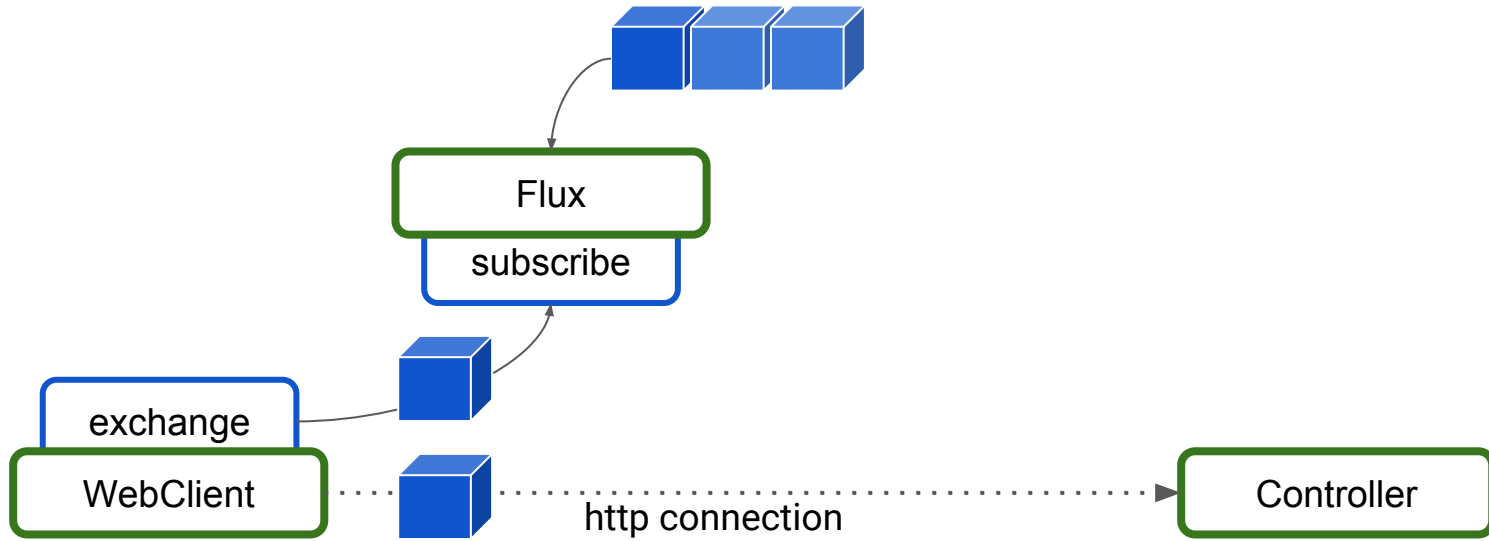
`RestTemplateBuilder` → `RestTemplate`

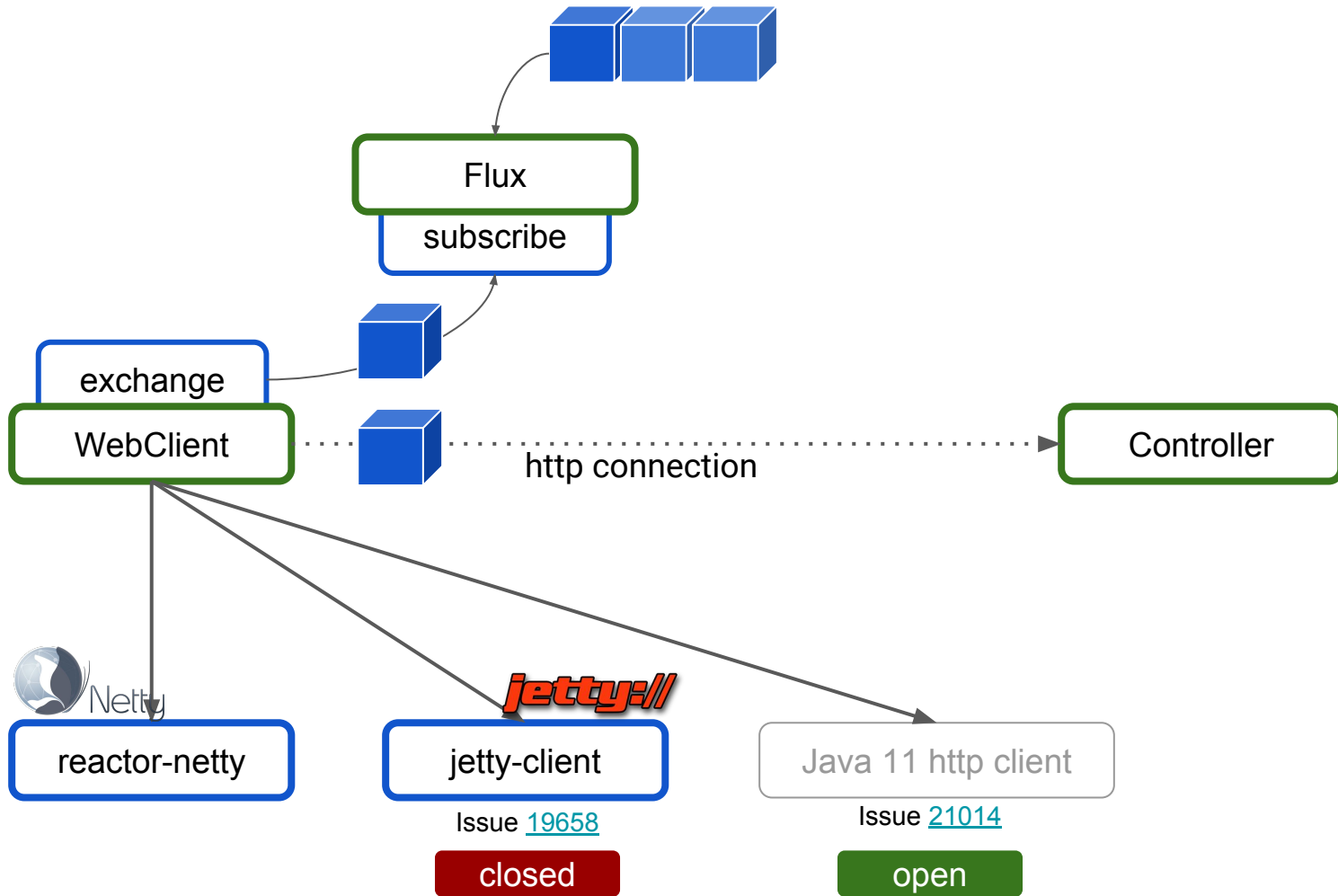
`RestTemplate` — передаём статический текст в `body`

Стало

`WebClient.Builder` → `WebClient`

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





Letter generator

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



Demo

Webflux

HER
IG YOU

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() ← Для начала вычитывания

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

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

reactor-netty



Flux



LetterController

subscribe (Subscriber)

Subscriber

reactor-netty



Flux



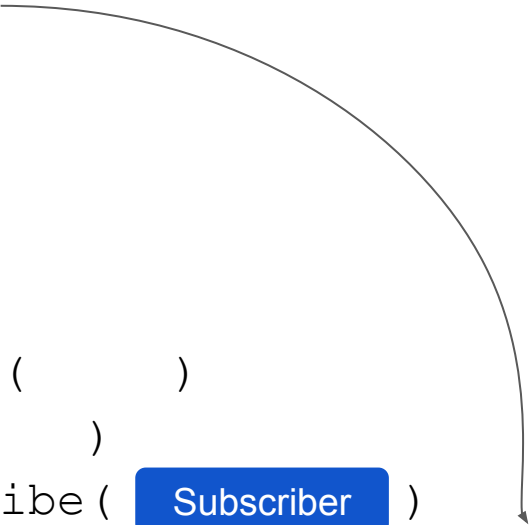
LetterController

```
.filter( )  
.map( )  
.subscribe( Subscriber )
```

→ onSubscribe(subscription)

subscription.request(N)

onNext()



reactor-netty



Flux



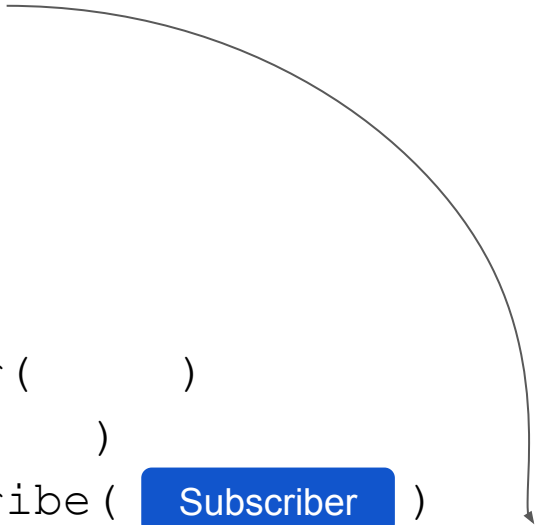
LetterController

```
.filter( )  
.map( )  
.subscribe( Subscriber )
```

→ onSubscribe(subscription)

subscription.request(N)

onNext()




reactor-netty



Flux



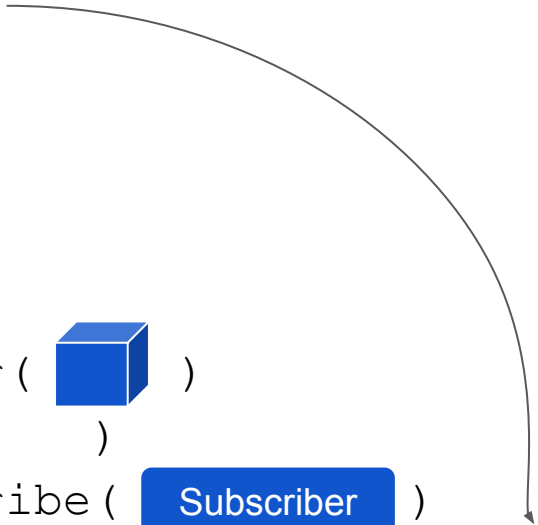
LetterController

```
.filter(  )  
.map( )  
.subscribe( Subscriber )
```

→ onSubscribe(subscription)

subscription.request(N)

onNext()




reactor-netty



Flux



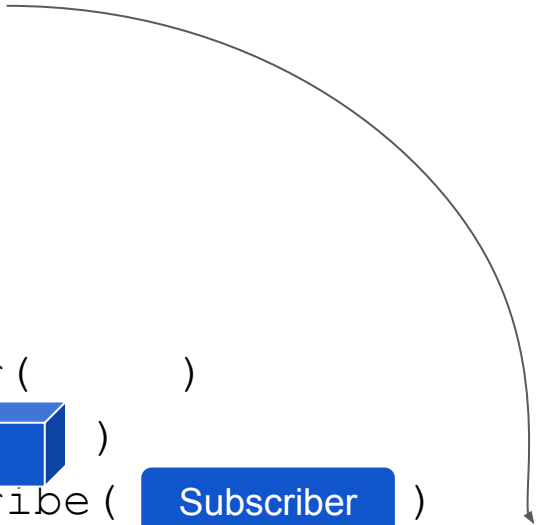
LetterController

```
.filter( )
.map(  )
.subscribe( Subscriber )
```

→ onSubscribe(subscription)

subscription.request(N)

onNext()



reactor-netty



Flux



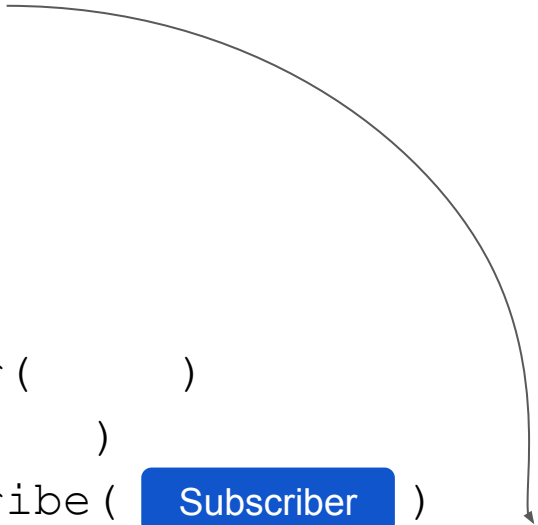
LetterController

```
.filter( )
.map( )
.subscribe( Subscriber )
```

→ onSubscribe(subscription)

subscription.request(N)

onNext()



reactor-netty



Flux



LetterController

```
.filter( )  
.map( )  
.subscribe( Subscriber )
```

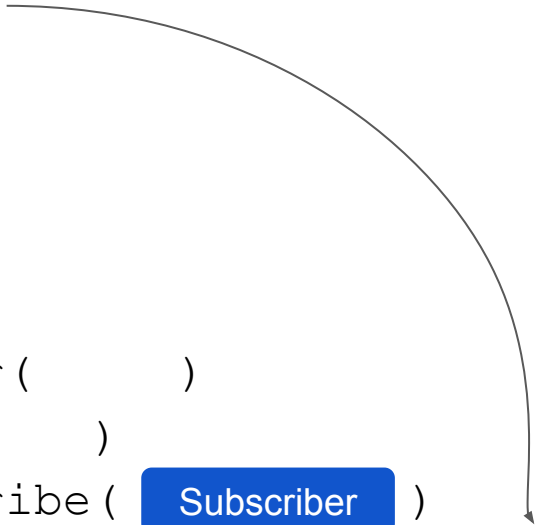
→ onSubscribe(subscription)

subscription.request(N)

onNext()

onComplete()

onError()



WebFlux #1

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



`Mono.empty()`

`Mono.never()`



reactor-netty



NO Close connection

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

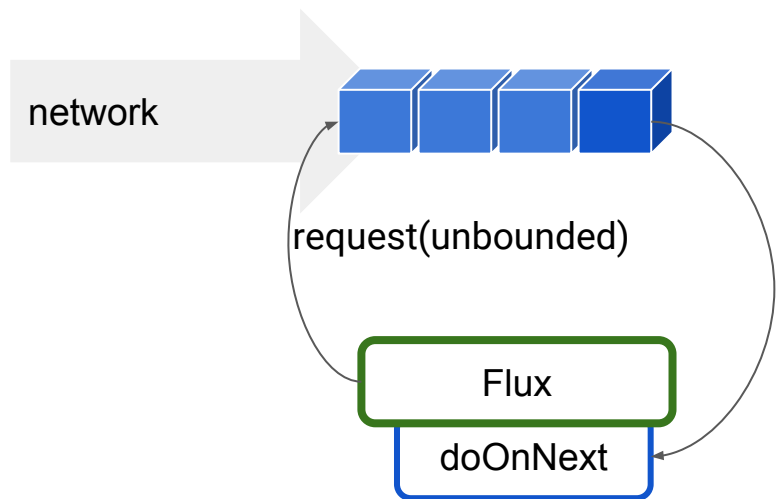
Чувак, где мои письма

service name	speed	buffers	worke
pechkin-service.....	531.03	<u>100/100</u>	0/
big-brother-service..	18.36	100/100	8/
agent-smith-service..	-1.00	0/0	0/0

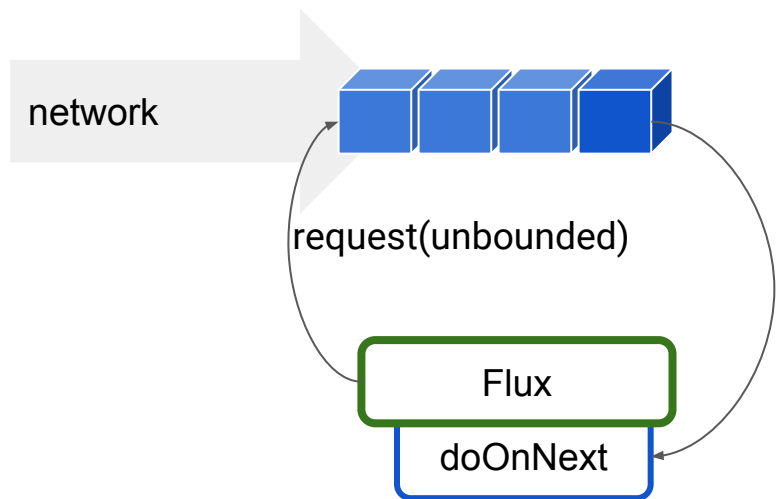
У нас же есть Backpressure из коробки!



Чувак, где мои письма



Чувак, где мои письма – в буферах!



Нас засудят за сексизм!

```
xt {
  snippetsDir = file("/build/docs/generated-snippets")
  imageTagLatest = project.hasProperty('imageTagLatest')
  imageTag = "${imageTagLatest ? 'latest' : project.version.toString().replaceAll('.', '')}"
  imageFullName = "ratauth:${imageTag}"
  dockerRegistryName = (project.hasProperty('DOCKER_REPOSITORY') ? project.findProperty('DOCKER_REPOSITORY')
  ) : System.getenv()['DOCKER_REPOSITORY']
  dockerRegistryUsername = (project.hasProperty('BINTRAY_USERNAME') ? project.findProperty('BINTRAY_USERNAME')
  ME') : System.getenv()['BINTRAY_USERNAME']
  dockerRegistryPassword = (project.hasProperty('BINTRAY_PASSWORD') ? project.findProperty('BINTRAY_PASSWORD')
  RD') : System.getenv()['BINTRAY_PASSWORD']
  dockerRegistryUrl = (project.hasProperty('DOCKER_URL') ? project.findProperty('DOCKER_URL') : System.get
  env()['DOCKER_URL'])
  dockerRegistryUrlDefault = 'https://index.docker.io/v1/'
  imageName = "${dockerRegistryName}/${dockerRegistryName}/${imageTag}"
}

apply plugin: 'spring-boot'
apply plugin: 'io.spring.dependency-management'
apply plugin: 'com.bmuschko.docker-remote-api'
apply plugin: 'nebula_maven-publish'

bootRepackage {
  mainClass = 'ru.ratauth.server.RatauthApplication'
  enabled = false
}

configurations.all {
  resolutionStrategy {
    [3] ~/git/security/ratauth/server/build.gradle [groovy]
  }
}
```

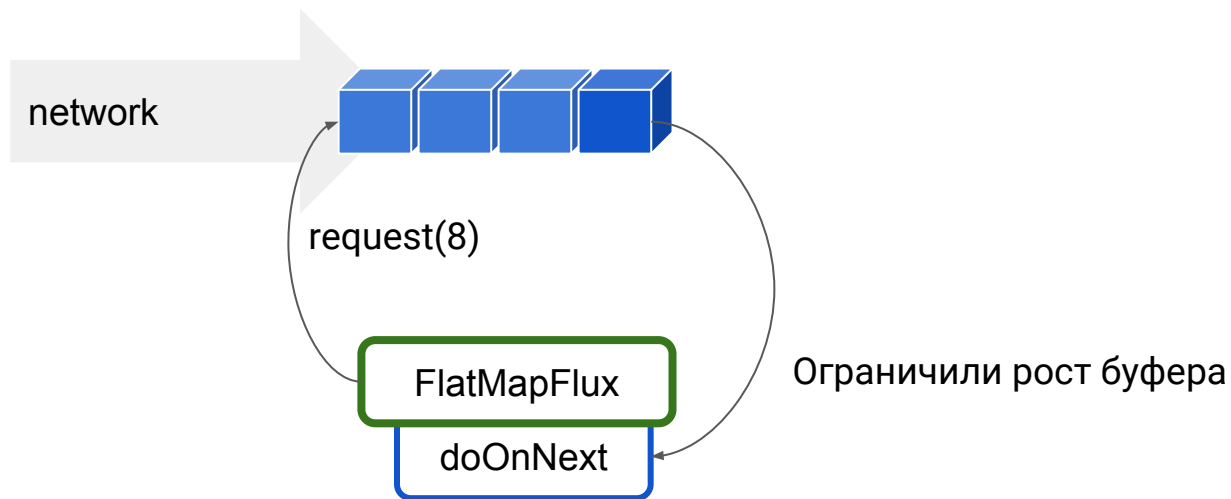
Нас засудят за сексизм!

Чувак, где мои письма – в буферах!

network

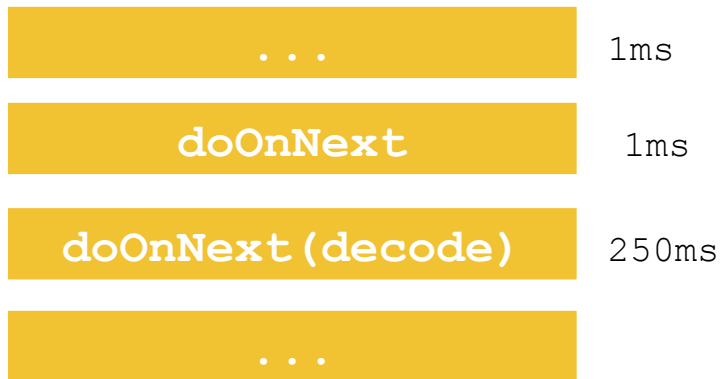


Чувак/чувиха, где мои письма – в буферах!



Чувак, почему так медленно?

1000rps vs 2rps



Чувак, где мои письма

...

`doOnNext`

`doOnNext (decode)`

`doOnNext (decode)`

`doOnNext (decode)`

...



Demo
Webflux #3

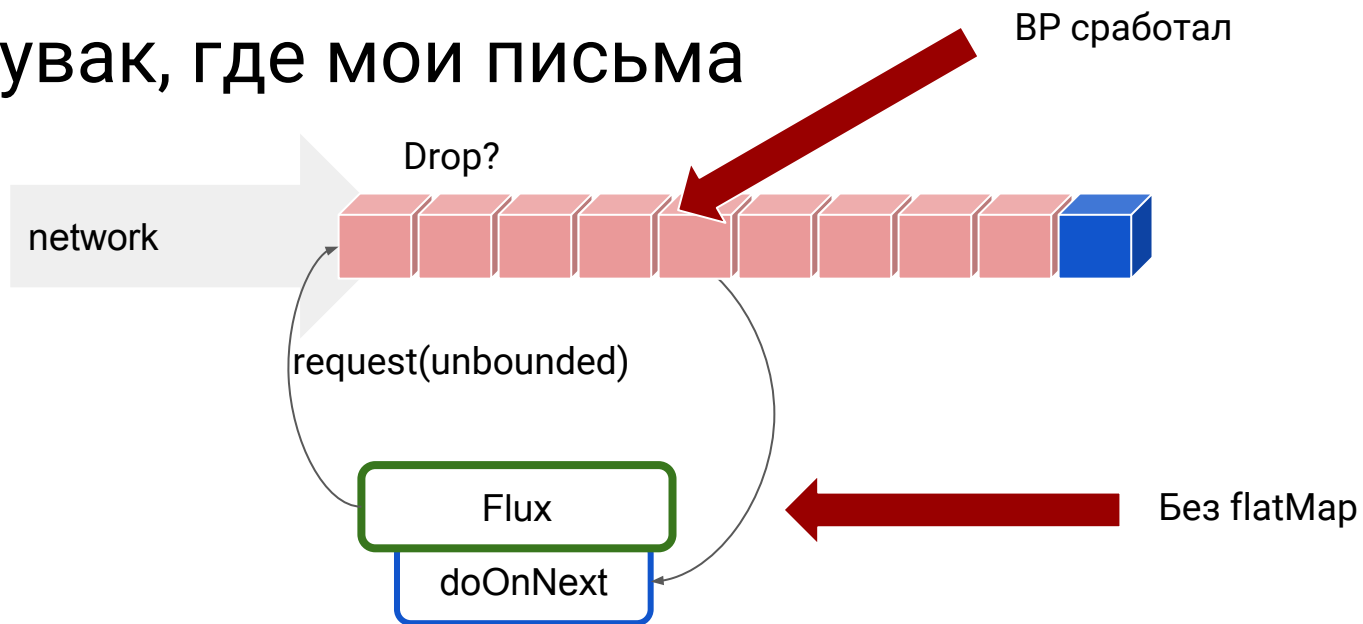
reactor-netty



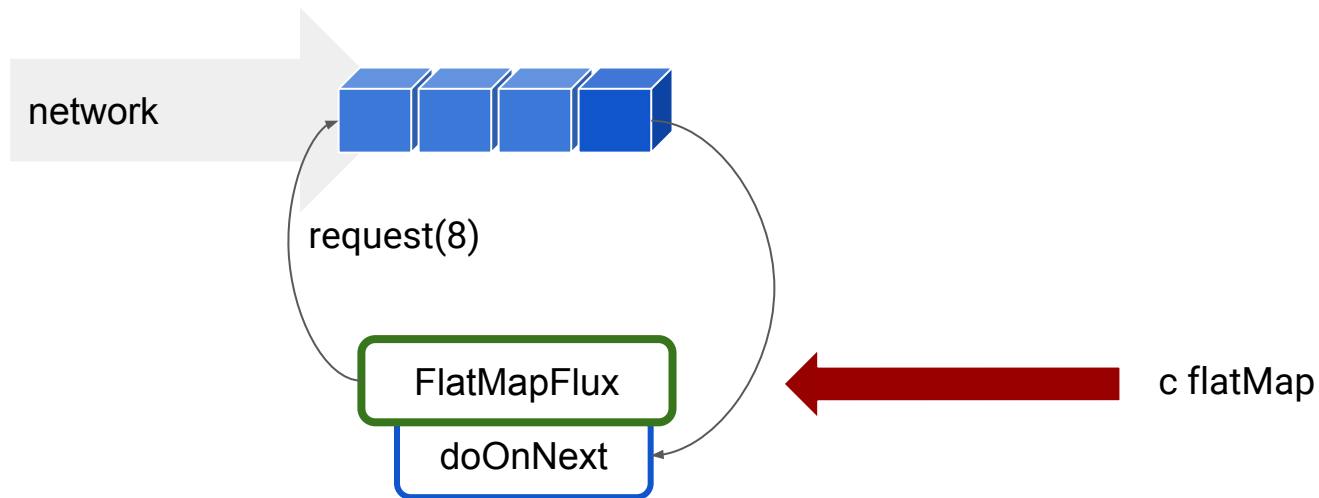
LetterController

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

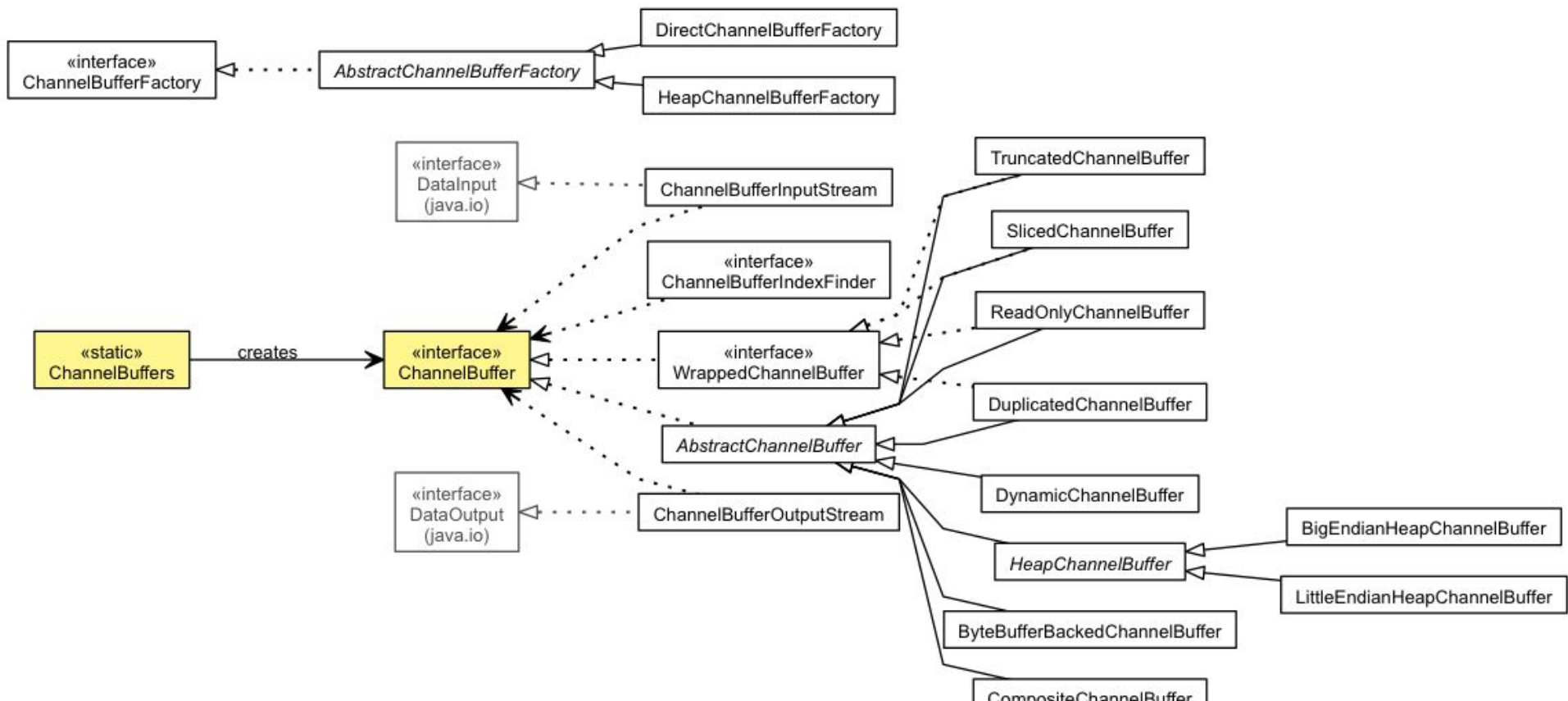
Чувак, где мои письма



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)

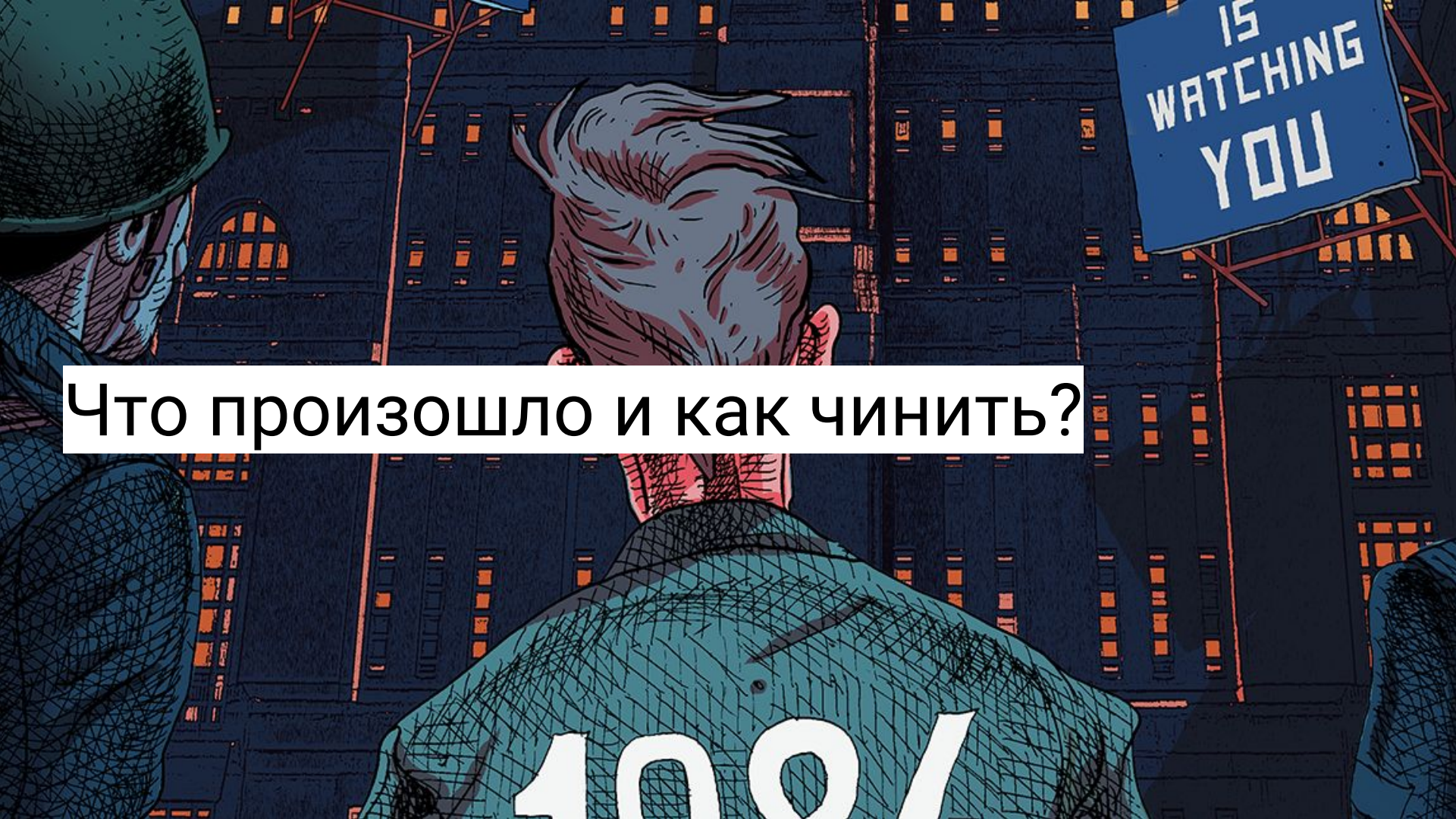
at j.n.DirectByteBuffer.<init>(DirectByteBuffer.java:118)

```
xt {
  snippetsDir = file("/build/docs/generated-snippets")
  imageTagLatest = project.hasProperty('imageTagLatest')
  imageTag = "${imageTagLatest ? 'latest' : project.version.toString().replaceAll('.', '')}"
  imageFullName = "ratauth:${imageTag}"
  dockerRegistryName = (project.hasProperty('DOCKER_REPOSITORY') ? project.findProperty('DOCKER_REPOSITORY')
  ) : System.getenv()['DOCKER_REPOSITORY']
  dockerRegistryUsername = (project.hasProperty('BINTRAY_USERNAME') ? project.findProperty('BINTRAY_USERNAME')
  ME') : System.getenv()['BINTRAY_USERNAME']
  dockerRegistryPassword = (project.hasProperty('BINTRAY_PASSWORD') ? project.findProperty('BINTRAY_PASSWORD')
  RD') : System.getenv()['BINTRAY_PASSWORD']
  dockerRegistryUrl = (project.hasProperty('DOCKER_URL') ? project.findProperty('DOCKER_URL') : System.get
  env()['DOCKER_URL'])
  dockerRegistryUrlDefault = 'https://index.docker.io/v1/'
  imageName = "${dockerRegistryName ? dockerRegistryName + '/' : ''}${imageTag}"
}

apply plugin: 'spring-boot'
apply plugin: 'io.spring.dependency-management'
apply plugin: 'com.bmuschko.docker-remote-api'
apply plugin: 'nebula_maven-publish'

bootRepackage {
  mainClass = "ru.ratauth.server.RatauthApplication"
  enabled = false
}

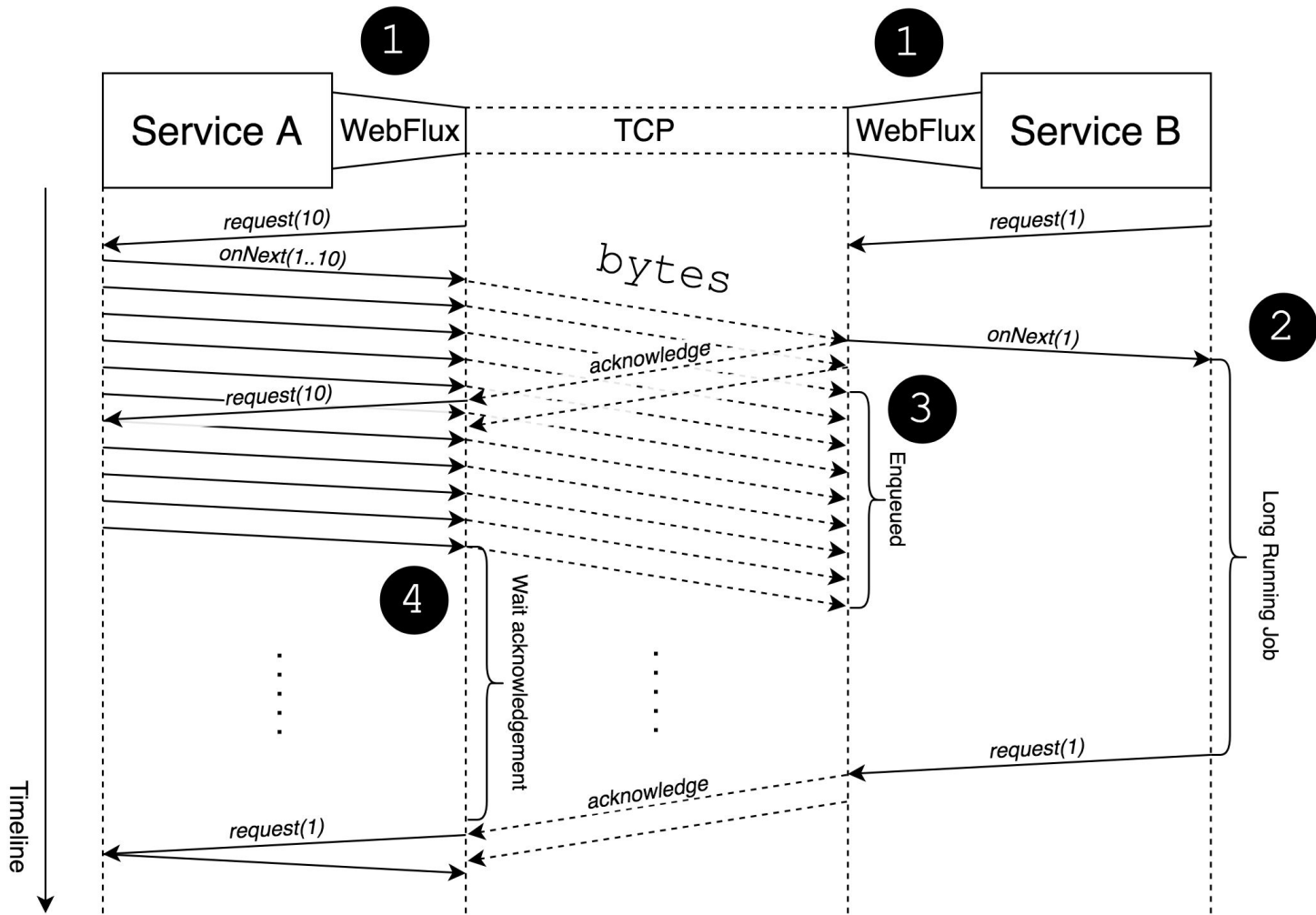
configurations.all {
  resolutionStrategy {
    [3] ~/git/security/ratauth/server/build.gradle [groovy]
```



Что произошло и как чинить?

A stylized illustration of a city at night. The sky is a deep red with swirling, wavy patterns. In the center, a tall, dark blue skyscraper stands out, featuring a prominent all-seeing eye symbol (the Eye of Providence) on its upper section. The building's windows are illuminated with small yellow and orange lights. The overall style is reminiscent of a comic book or graphic novel illustration.

Случился ТСР Backpressure



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

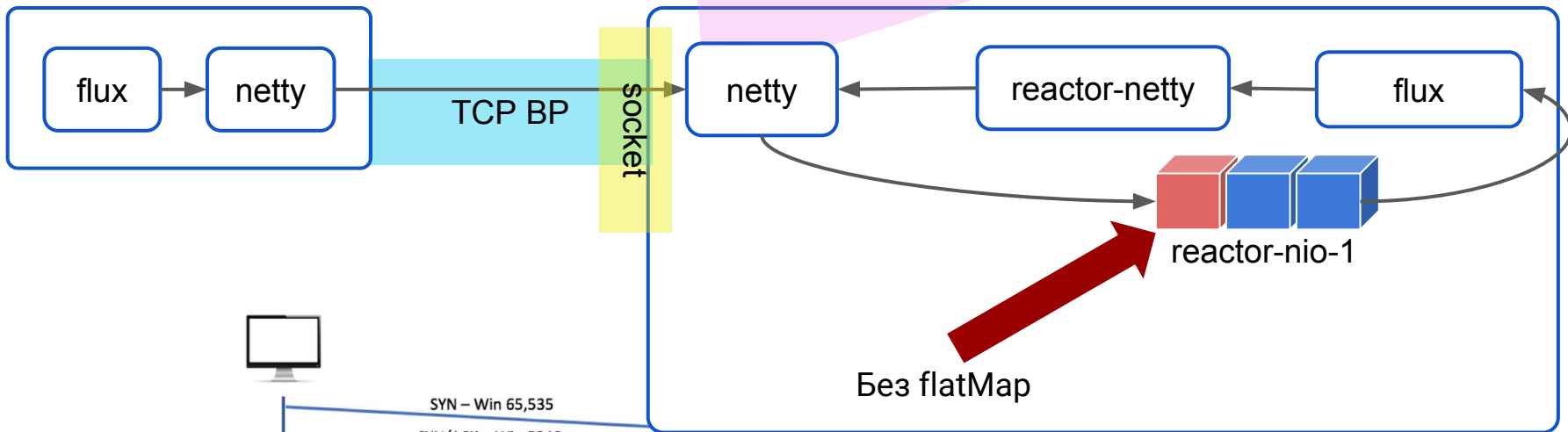
[reactor-http-nio-3] onNext

[reactor-http-nio-3] onNext

[reactor-http-nio-3] onNext

Без flatMap #0

Пулы DirectBuffer`ов netty



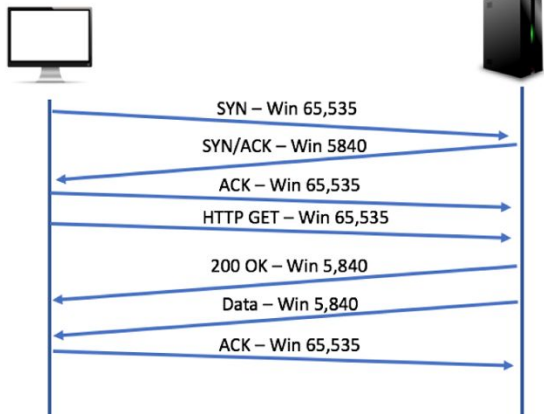
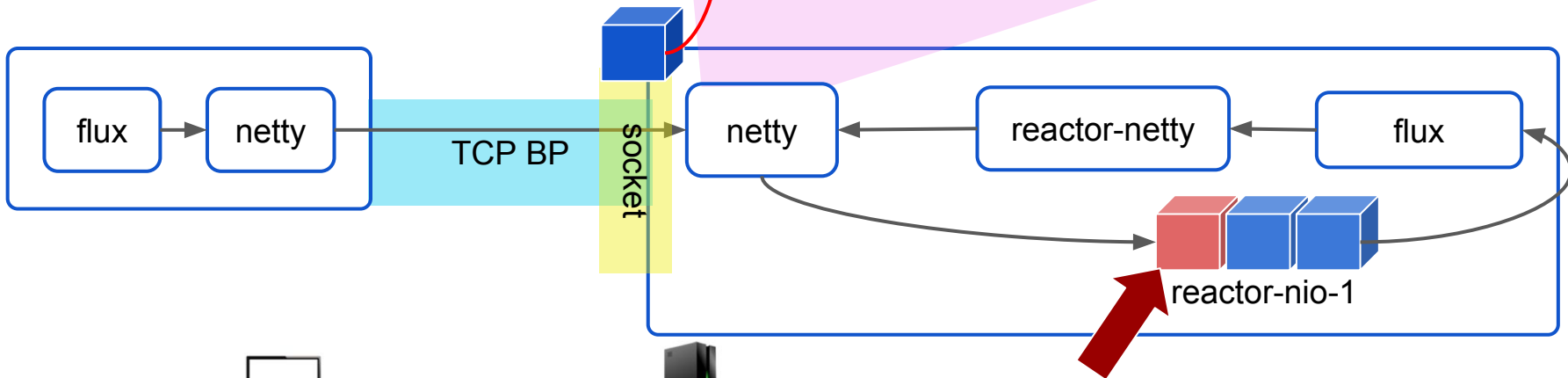
Без flatMap



SYN – Win 65,535
SYN/ACK – Win 5840
ACK – Win 65,535
HTTP GET – Win 65,535
200 OK – Win 5,840
Data – Win 5,840
ACK – Win 65,535

Без flatMap #1

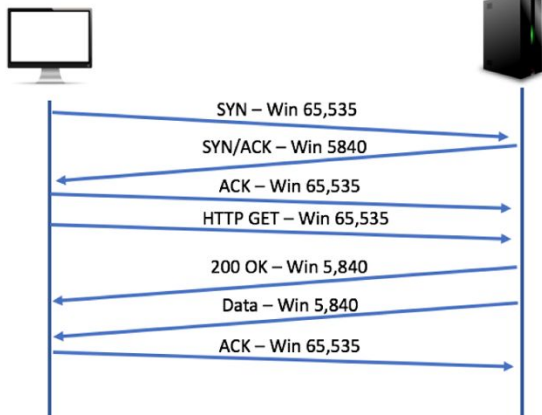
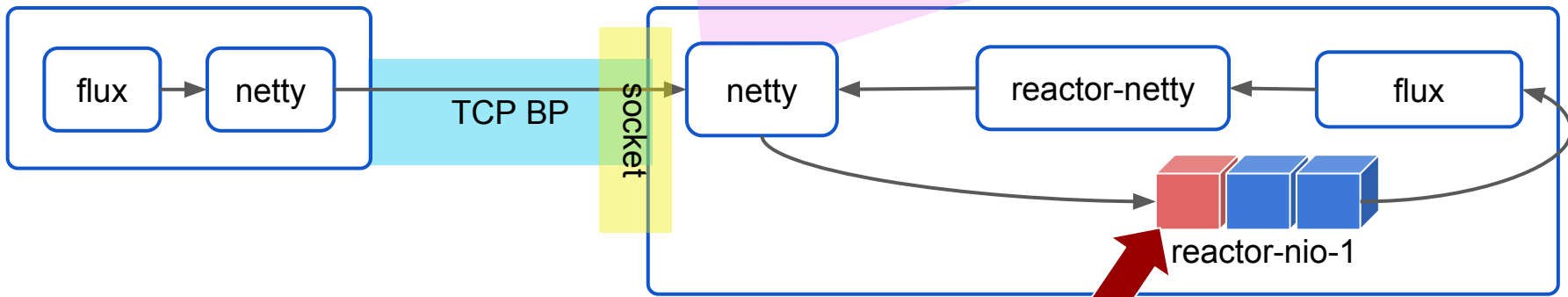
Пулы DirectBuffer`ов netty



1. Копирует данные из системного сокета в DirectBuffer

Без flatMap #2

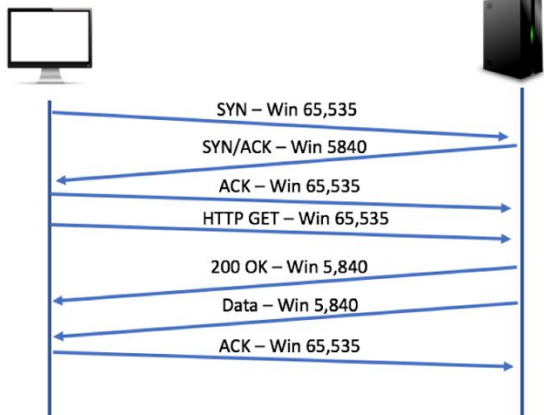
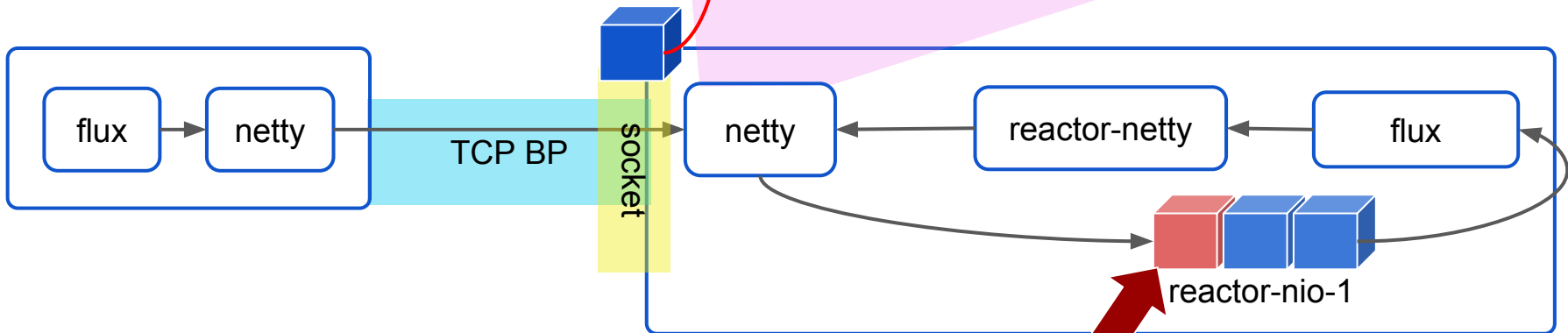
Пулы DirectBuffer`ов netty



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

Без flatMap #3

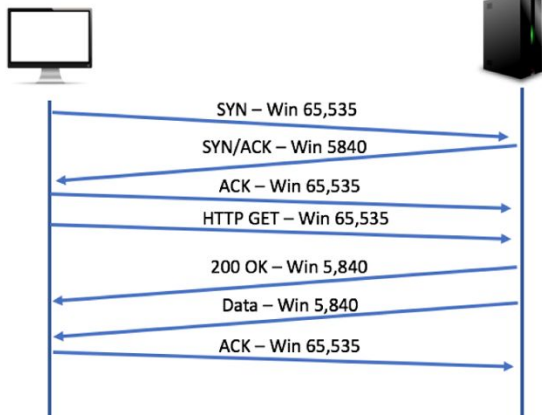
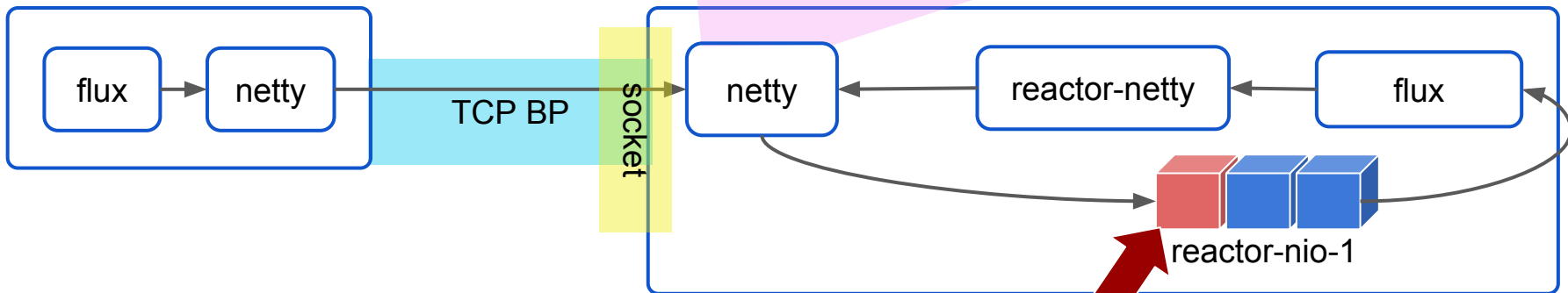
Пулы DirectBuffer`ов netty



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

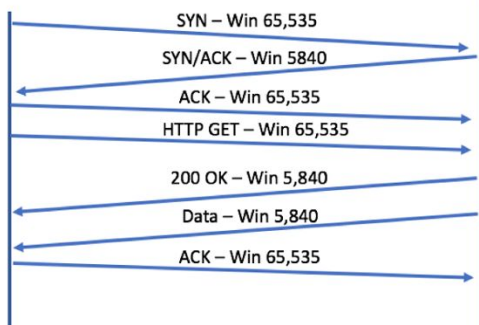
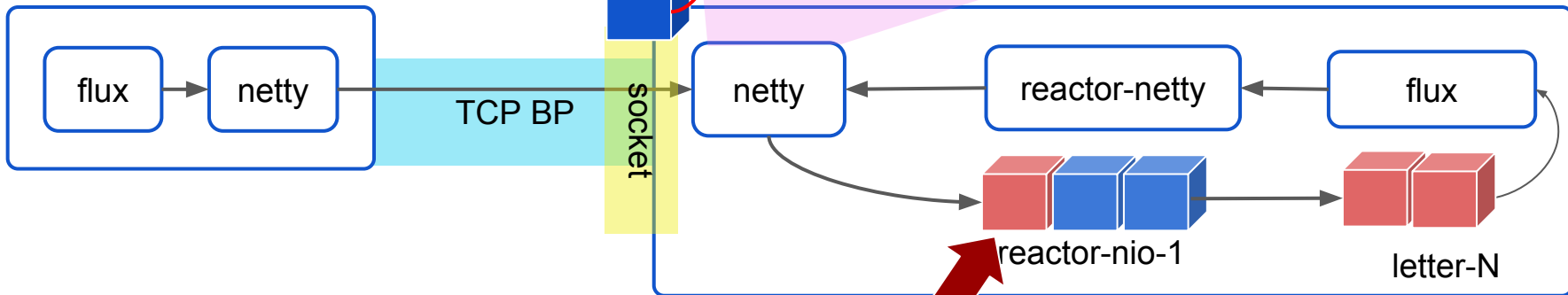
Пулы DirectBuffer`ов netty



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

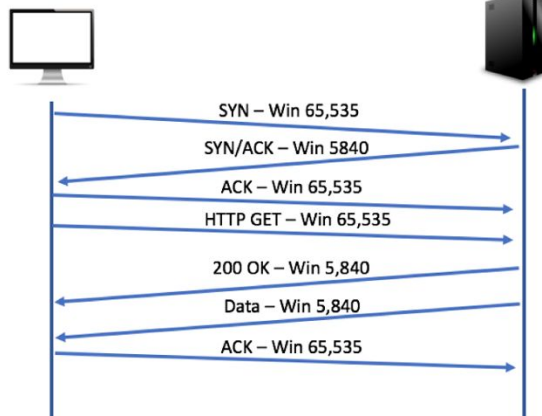
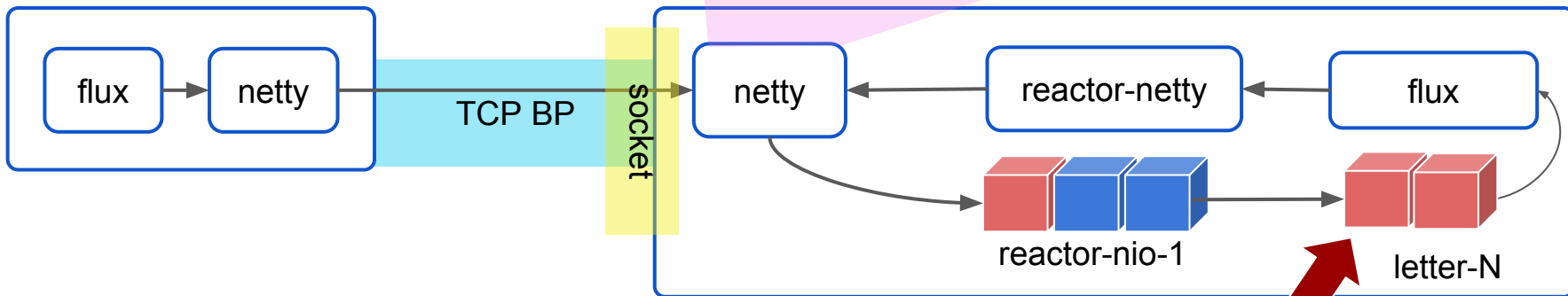
Пулы DirectBuffer`ов netty



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

После добавления flatMap

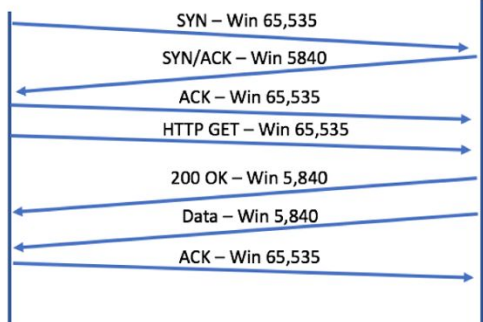
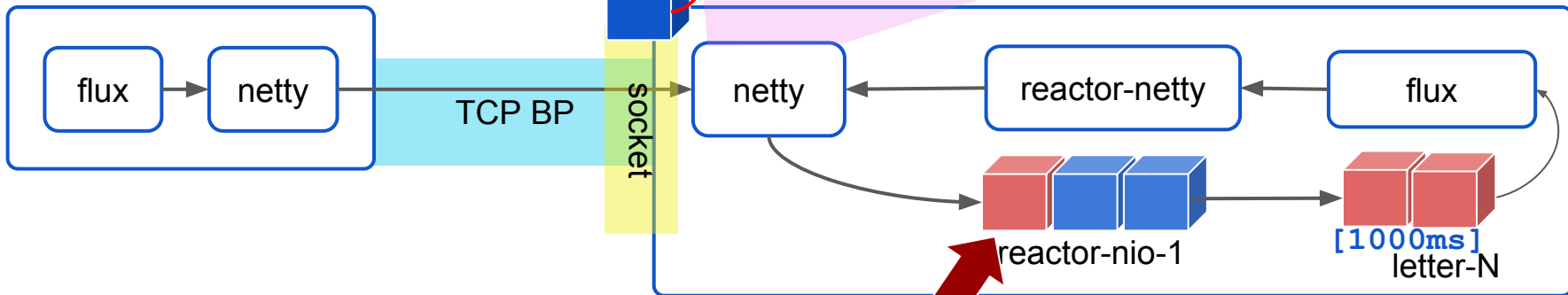
Пулы DirectBuffer`ов netty



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

После добавления flatMap

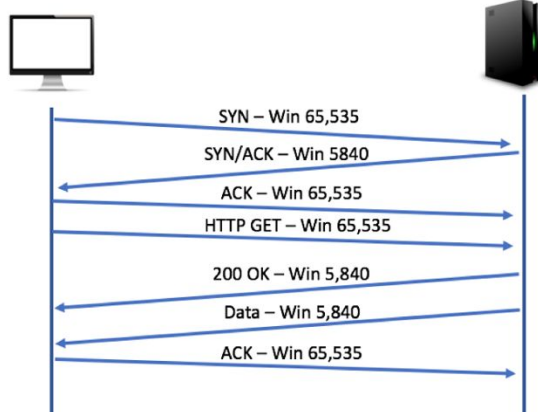
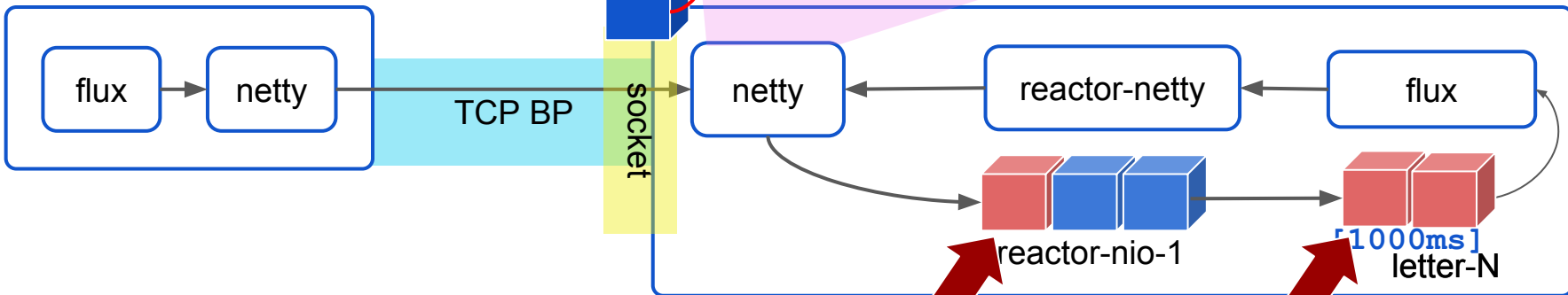
Пулы DirectBuffer`ов netty



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

После добавления flatMap

Пулы DirectBuffer`ов netty



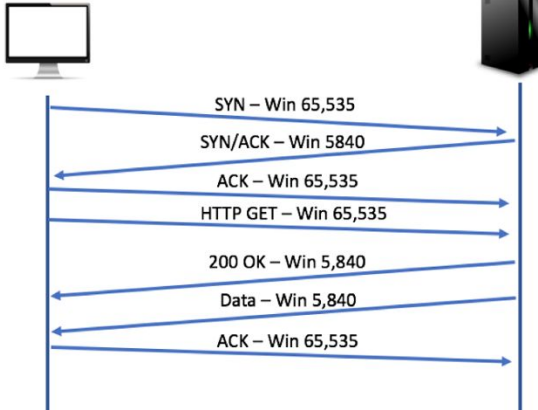
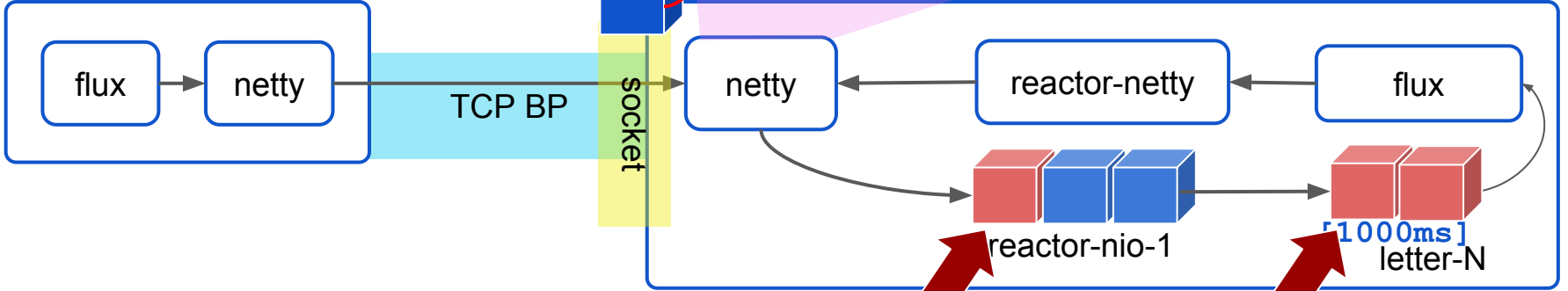
Копирует данные из системного сокета в DirectBuffer [1ms]

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

работает независимо

После добавления flatMap

Пулы DirectBuffer`ов netty



Копирует данные из системного сокета в DirectBuffer [1ms]

Вызывает наш медленный метод decoder.decode LetterController.java a [1000ms]

работает независимо

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)

```
xt {
  snippetsDir = file("/build/docs/generated-snippets")
  imageTagLatest = project.hasProperty('imageTagLatest')
  imageTag = "${imageTagLatest ? 'latest' : project.version.toString().replaceAll('.', '')}"
  imageFullName = "ratauth:${imageTag}"
  dockerRegistryName = (project.hasProperty('DOCKER_REPOSITORY') ? project.findProperty('DOCKER_REPOSITORY')
  ) : System.getenv()['DOCKER_REPOSITORY']
  dockerRegistryUsername = (project.hasProperty('BINTRAY_USERNAME') ? project.findProperty('BINTRAY_USERNAME')
  ME') : System.getenv()['BINTRAY_USERNAME']
  dockerRegistryPassword = (project.hasProperty('BINTRAY_PASSWORD') ? project.findProperty('BINTRAY_PASSWORD')
  RD') : System.getenv()['BINTRAY_PASSWORD']
  dockerRegistryUrl = (project.hasProperty('DOCKER_URL') ? project.findProperty('DOCKER_URL') : System.get
  env()['DOCKER_URL'])
  dockerRegistryUrlDefault = 'https://index.docker.io/v1/'
  imageName = "${dockerRegistryName ? dockerRegistryName + '/' : ''}${imageTag}"
}

apply plugin: 'spring-boot'
apply plugin: 'io.spring.dependency-management'
apply plugin: 'com.bmuschko.docker-remote-api'
apply plugin: 'nebula_maven-publish'

bootRepackage {
  mainClass = "ru.ratauth.server.RatauthApplication"
  enabled = false
}

configurations.all {
  resolutionStrategy {
    [3] ~/git/security/ratauth/server/build.gradle [groovy]
```

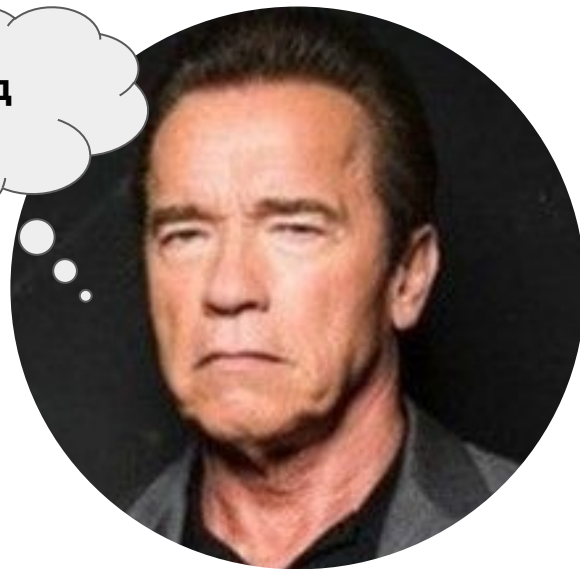
У нас же есть Backpressure из коробки!



У нас же есть Backpressure из коробки!

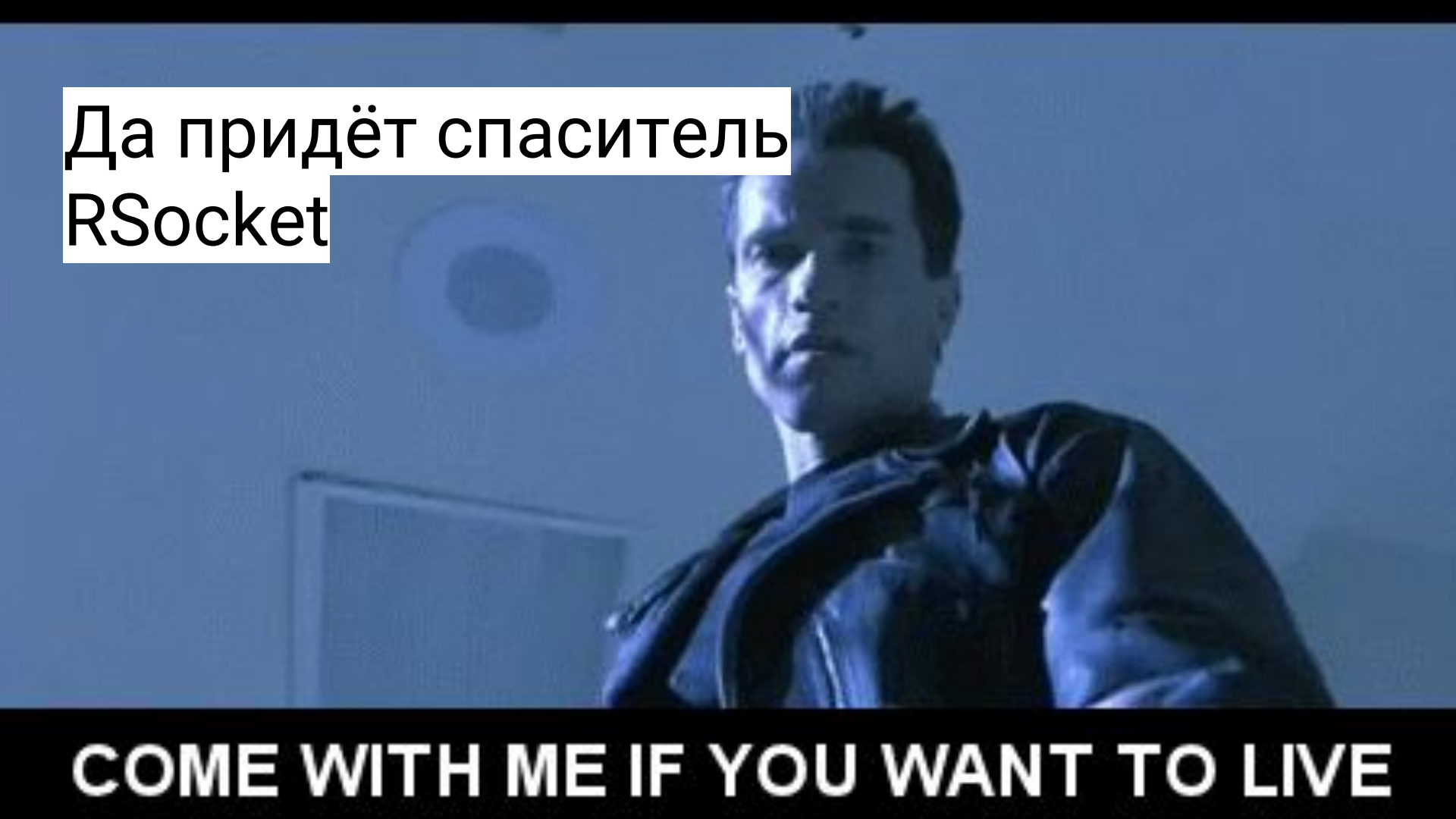


Нет, сайд эффект



Хочу Backpressure



A blue-tinted photograph of Arnold Schwarzenegger in a military-style jacket, looking off-camera with a serious expression. The background is a plain wall with a circular vent and a door frame.

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 WR имеет сайд эффекты
 - b. Control flow между сервисами для реактивного пайплайна так же важен
2. Webflux не подходит для балансировки скорости между сервисами
3. C RSocket честный backpressure, пока сырой, но перспективный
- 4.



Настоящие выводы

Не стоит внедрять технологии на основе авторитета:

1. спикера
2. компании
3. Тренда

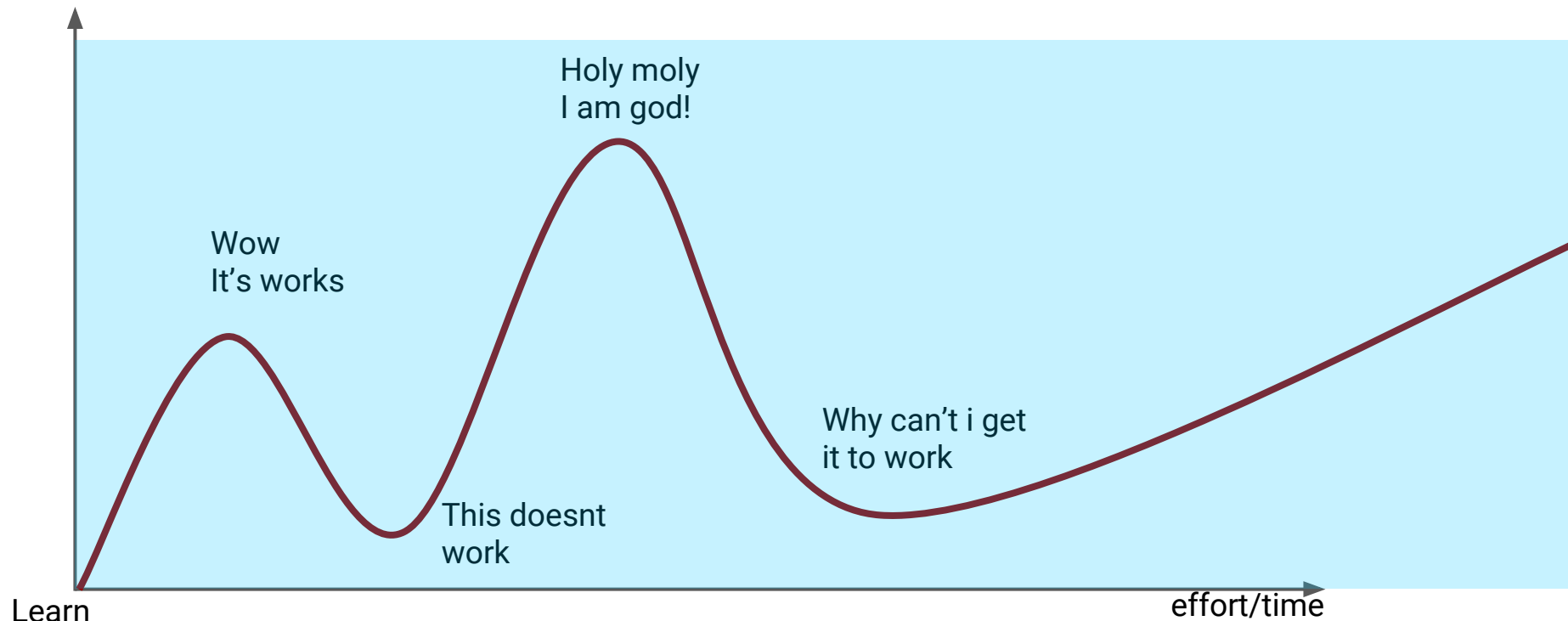
Webflux хорош когда нужно сэкономить коннекшены и отправить пачку данных

RSocket хорош во всех наших кейсах

Свой велосипед всегда понятнее

Выводы – кривая обучения reactive

mastery

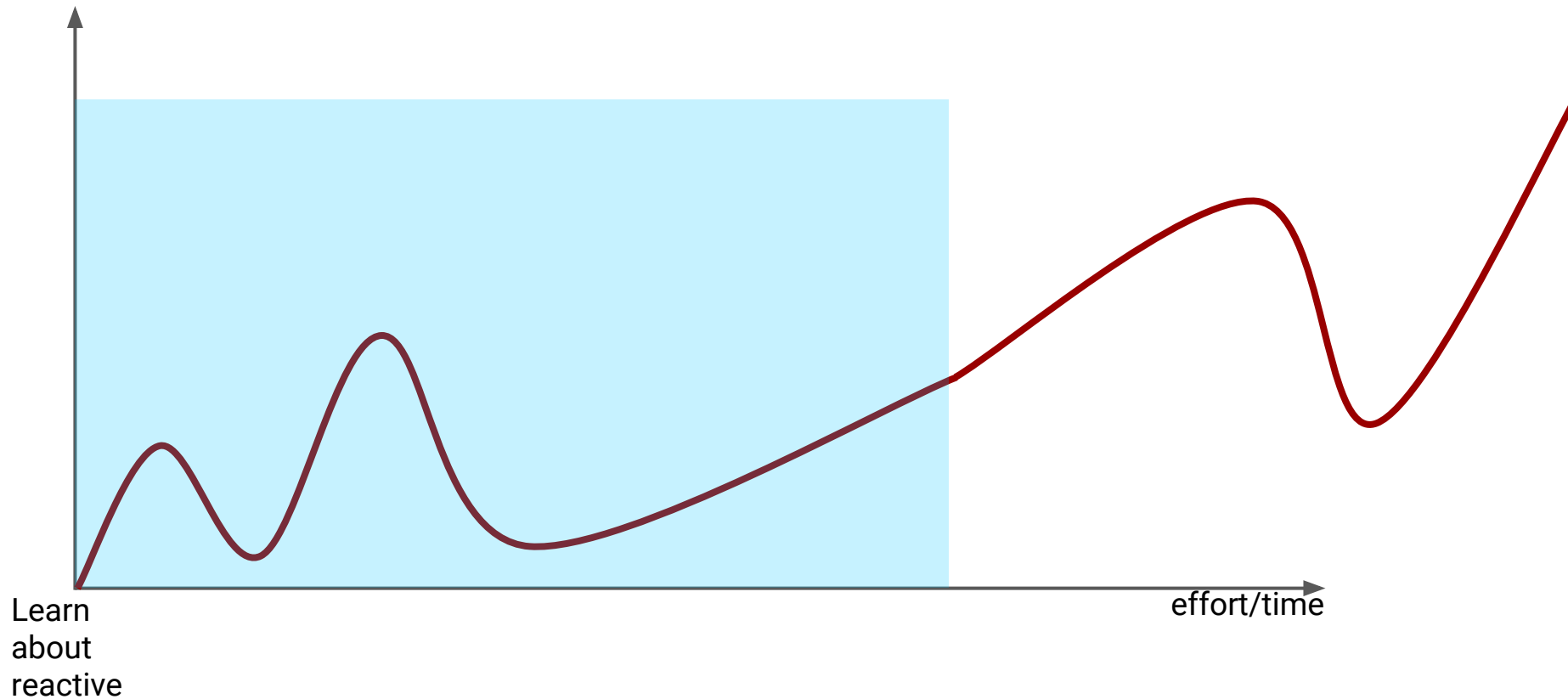


Learn
about
reactive

effort/time

Выводы – кривая обучения reactive

mastery



Вопросы



 @tolkv

 @lavcraft



@jekaborisov 

@jeka1978 

Вопросы



 alfalab



 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](#)