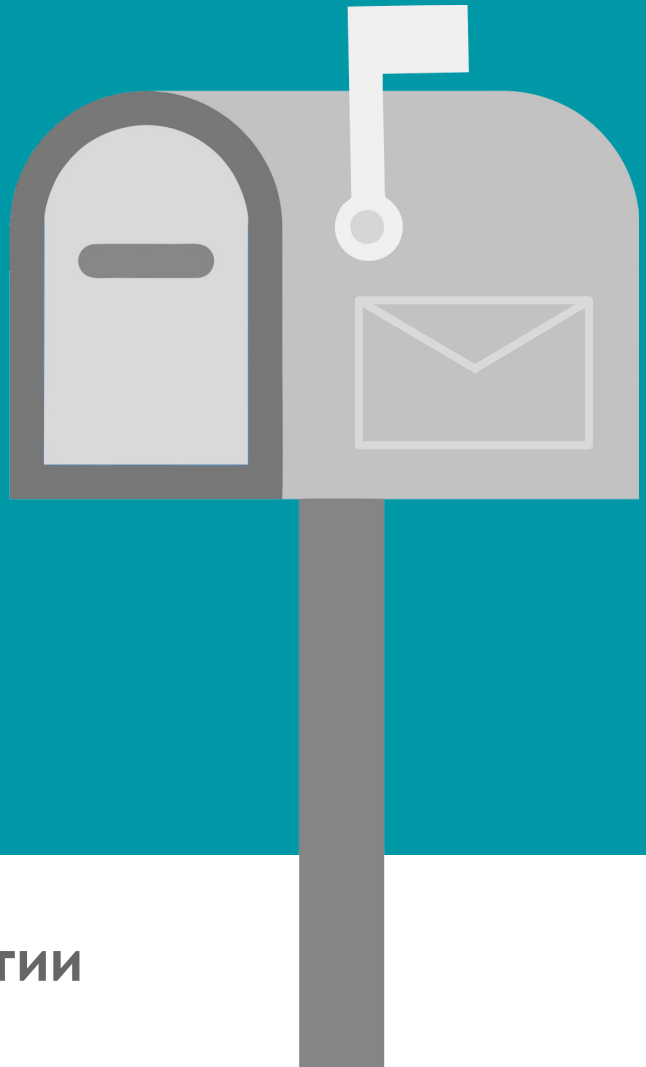
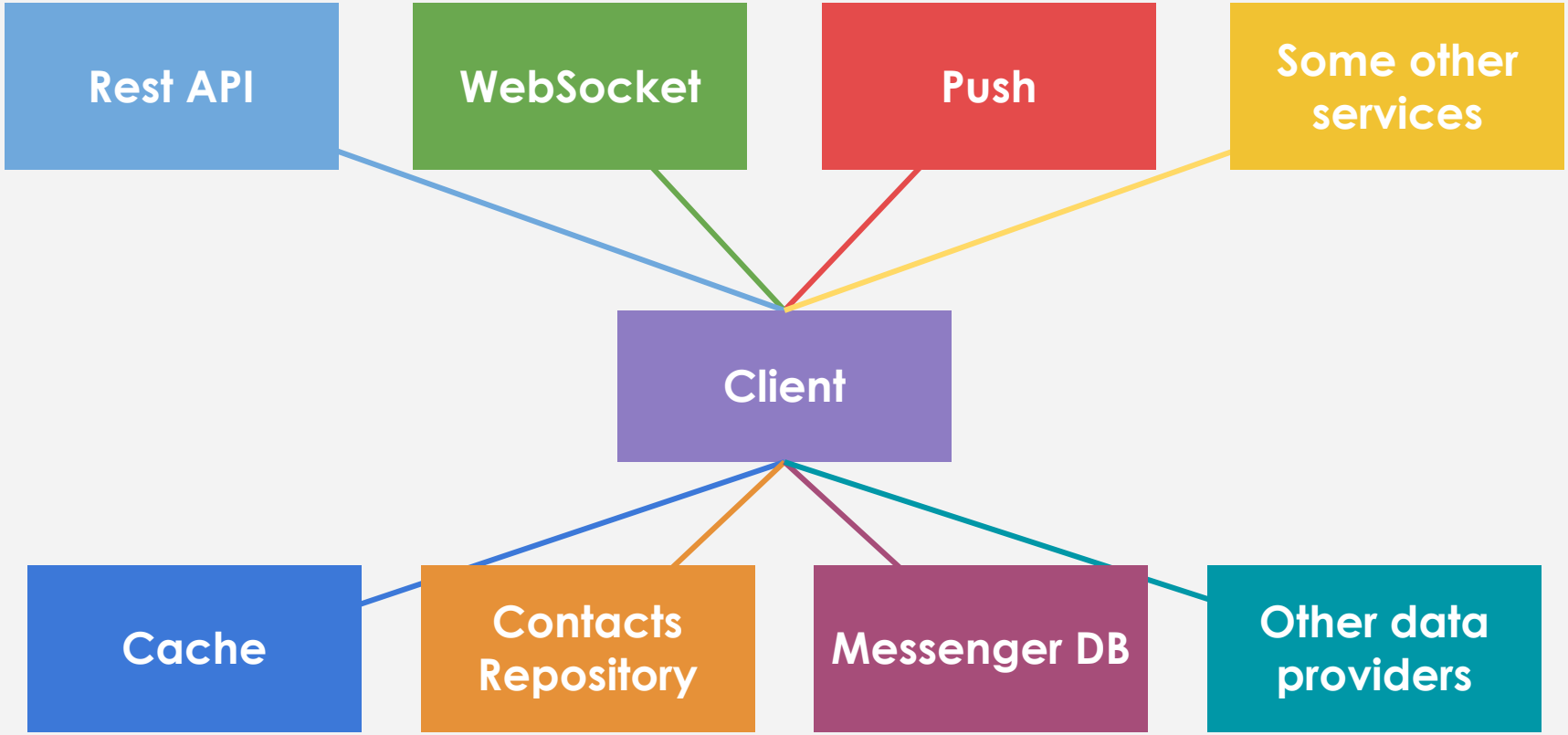
Акторные модели

Новый взгляд на старый подход



Владимир Теблоев, Сбербанк-Технологии

“Толстый” клиент



Парадигмы программирования

ООП

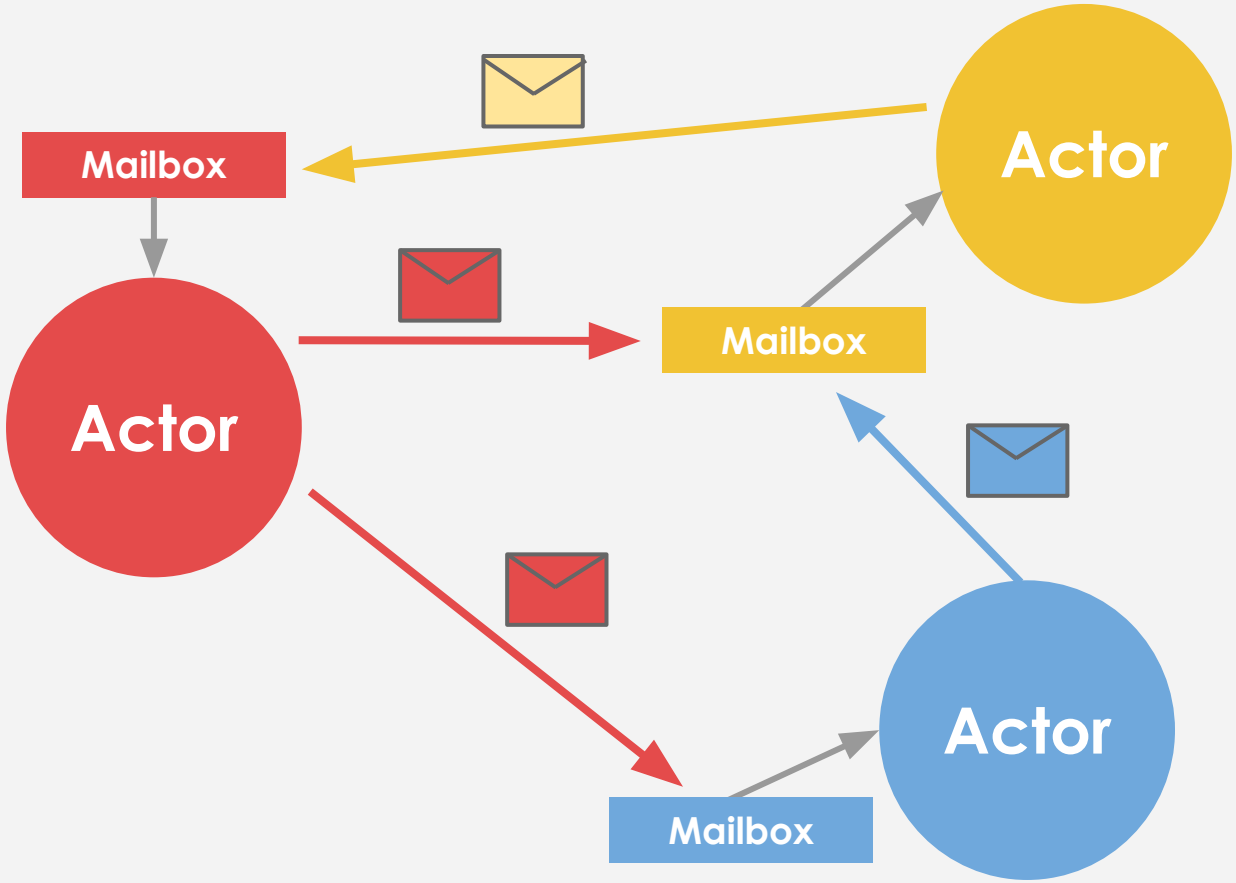
ФП

АМ

Концепция акторной модели

- Все есть актор. Даже вы ¯_(ツ)_/¯
- Фундаментальный инструмент
- Универсальные примитивы расчёта

Акторная система



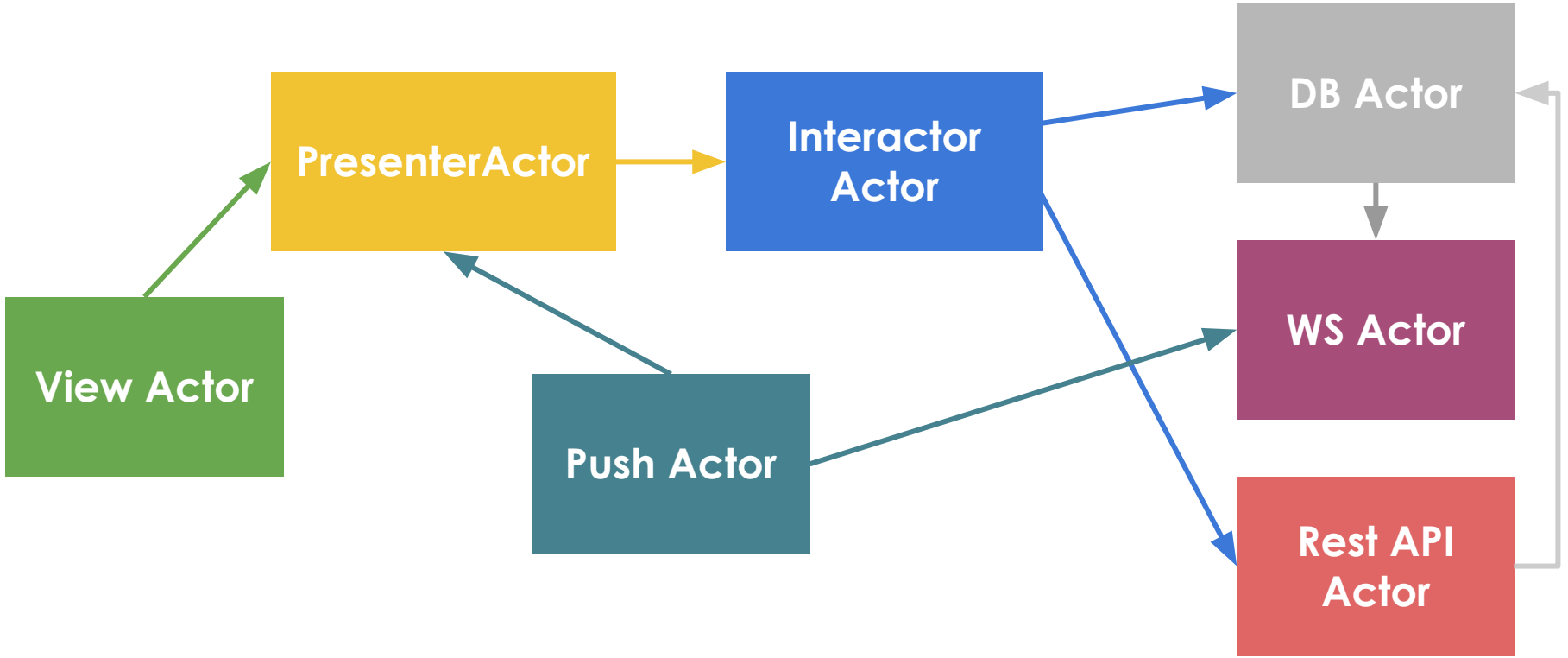
Простая реализация актора

```
fun someActor() = actor<Any>(CommonPool) {
    for (msg in channel) {
        when (msg) {
            is String -> println(msg)
            is Int -> someFun(msg)
        }
    }
}
```

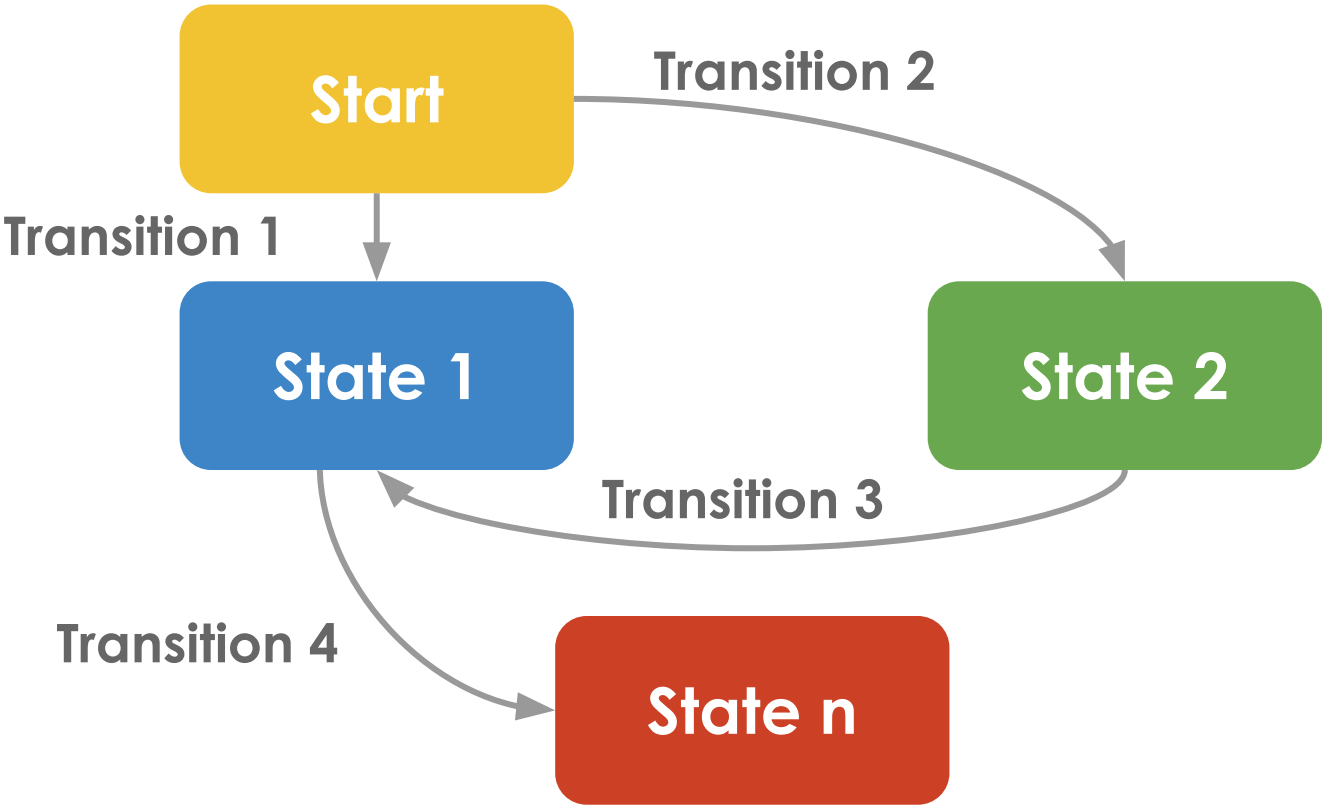
Области применения

- Асинхронное взаимодействие
- Машины состояний
- Полнодуплексные системы
- Свой реактивный фреймворк

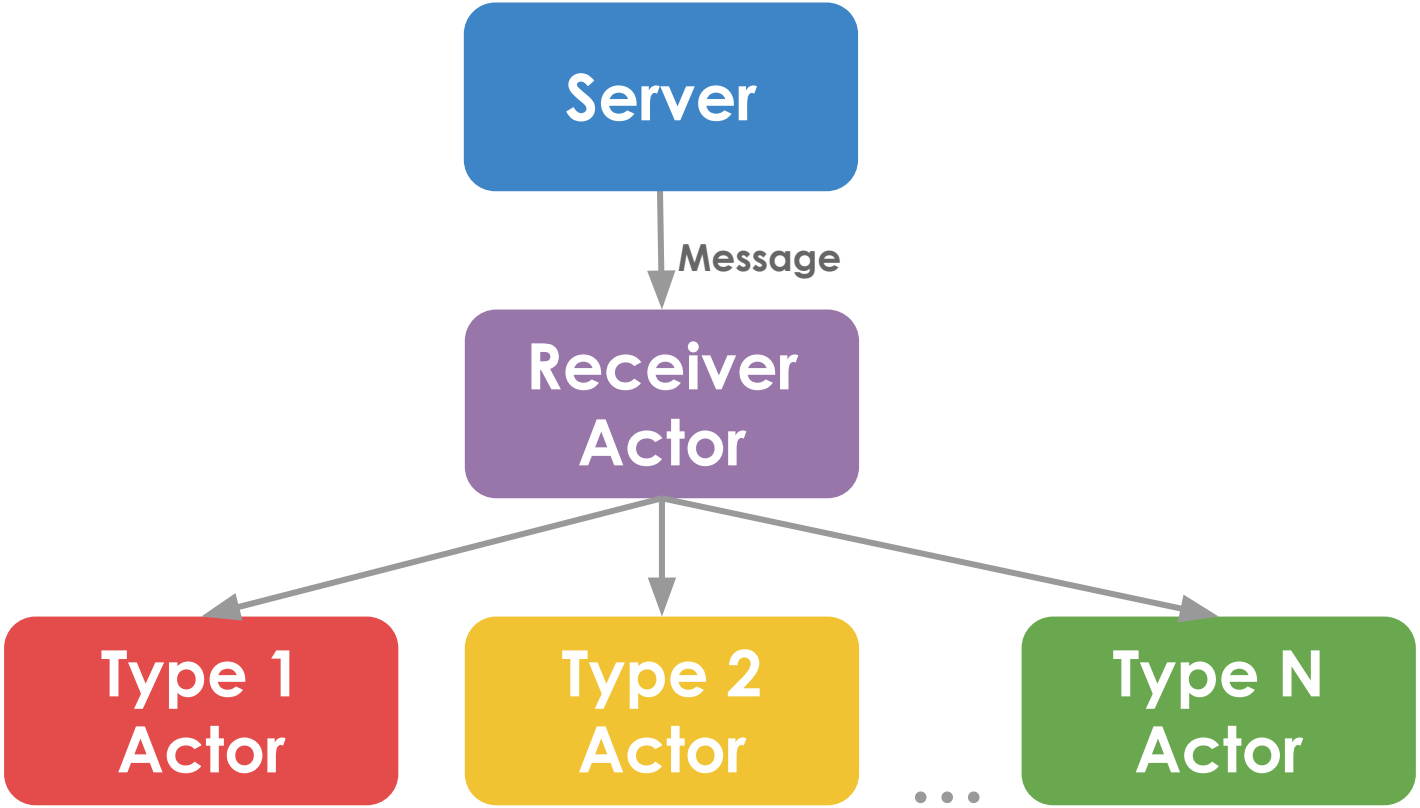
Типичная ситуация



Машина состояний



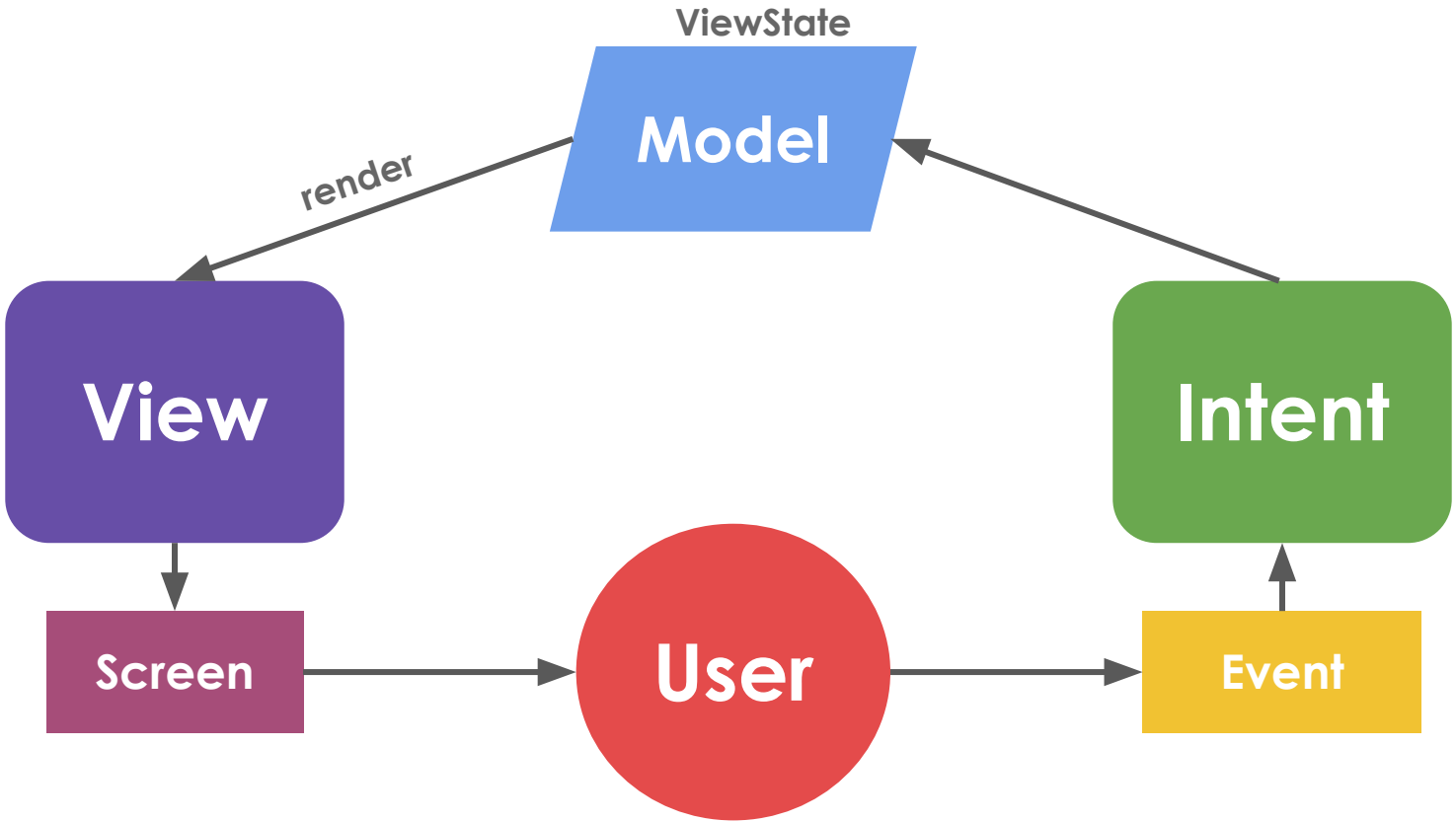
Полнодуплексная система



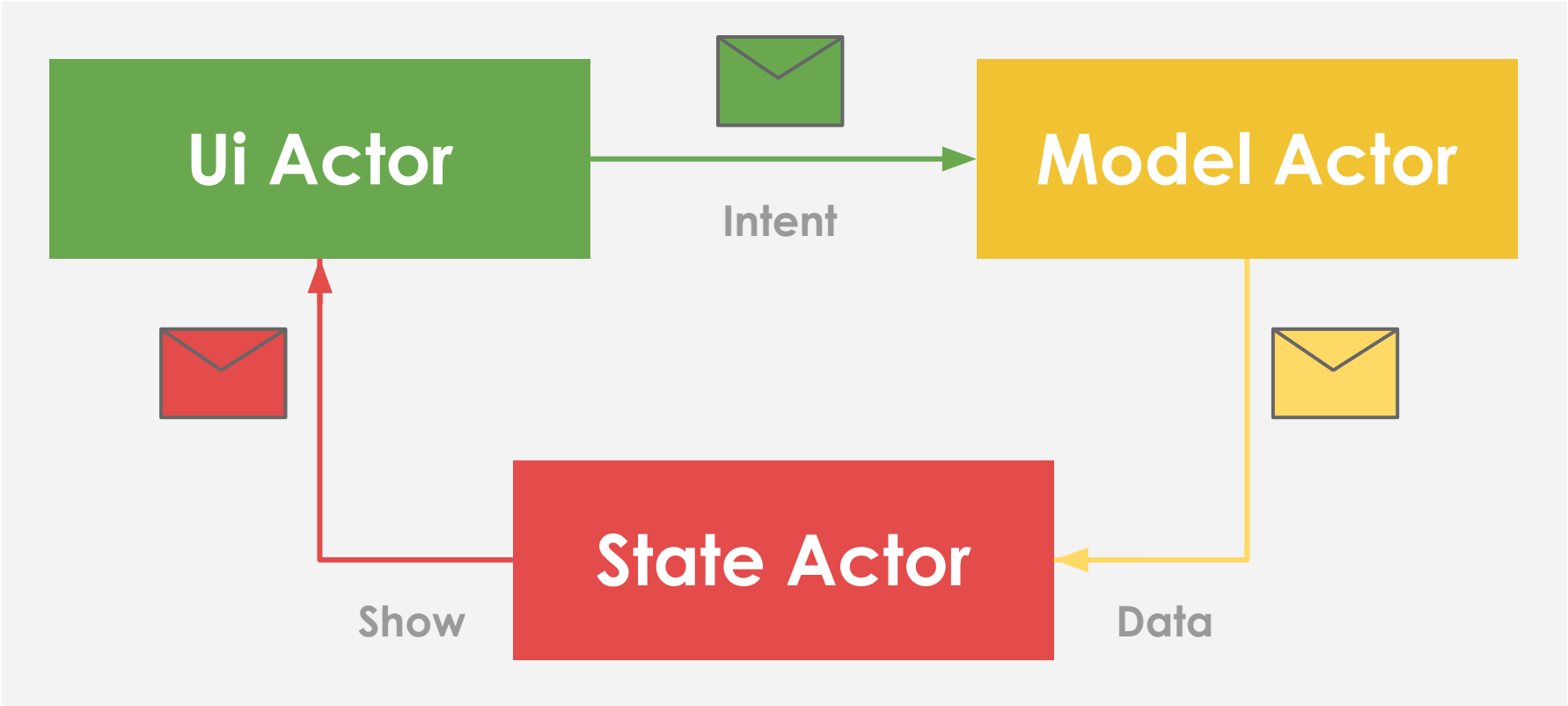
Зачем нужна акторная модель на android?

- Никаких синхронизаций и блокировок
- Простота в реализации отдельно взятых акторов
- Легкость тестирования
- Не ограничивает выбор архитектуры
- Производительность

Обычный MVI



MVI на ~~стероидах~~ акторах

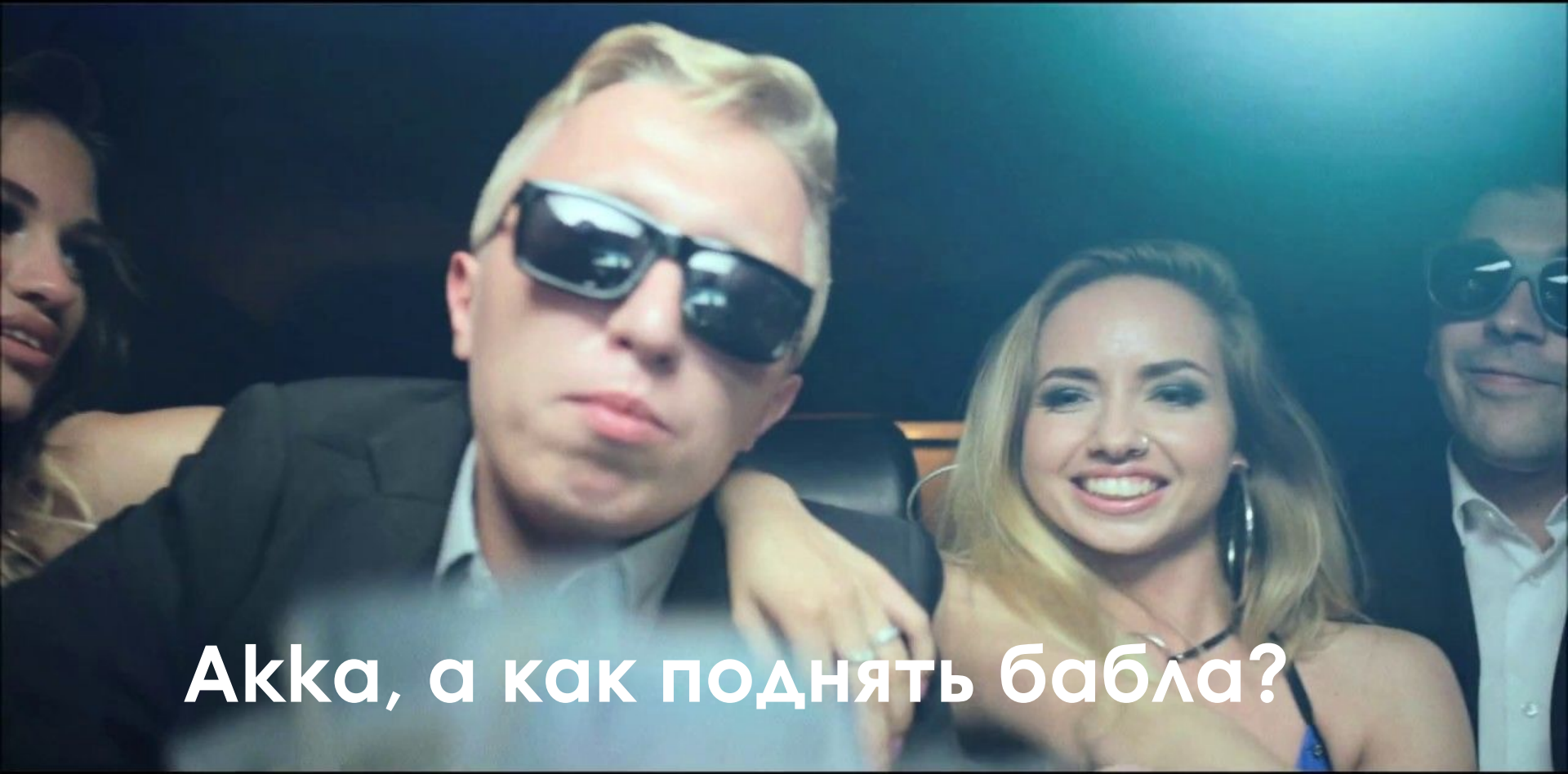


Минусы

- Совершенно иная парадигма разработки (нет)
- Трудоемкость рефакторинга
- Нет библиотек под android

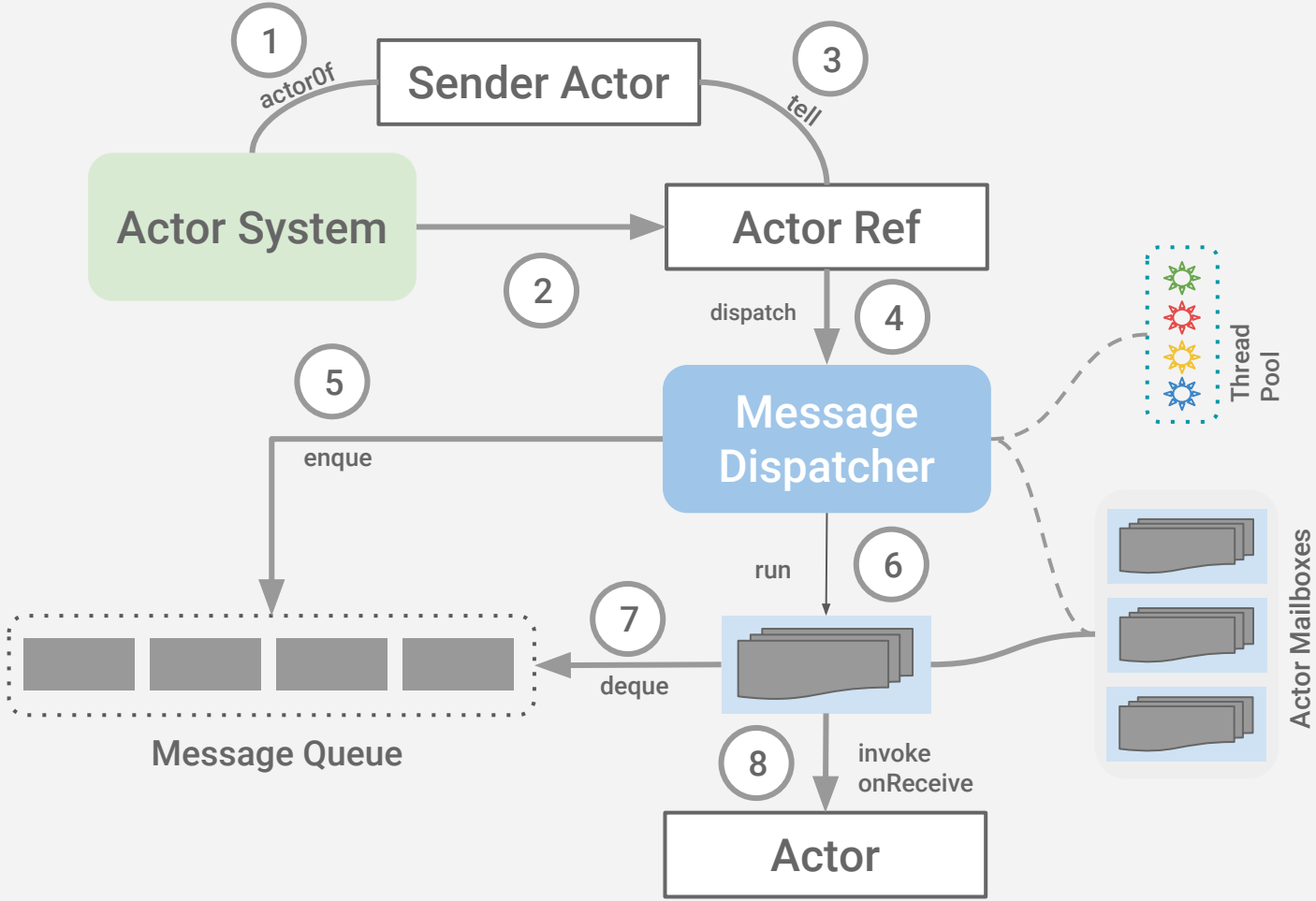
Реализации акторной модели

- Akka
- Kotlin coroutines
- Quasar
- etc


Akka, а как поднять бабла?

Akka

- Реализация акторной системы
- Машина состояний
- Реактивные потоки данных
- Шина данных
- etc

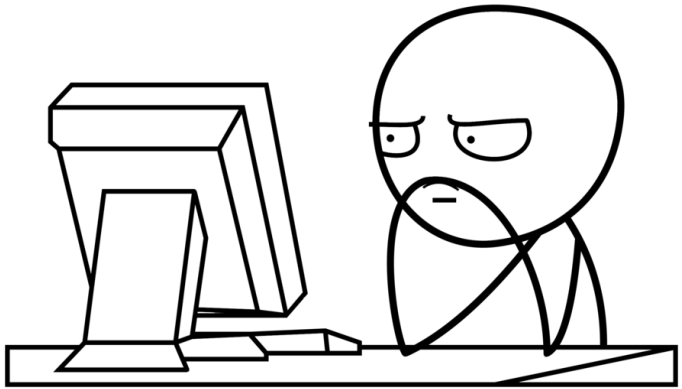

1
actor0f
Sender Actor
3
tell
Actor System
Actor Ref
2
dispatch
4
5
Message Dispatcher
Thread Pool
enque
run
6
7
deque
Actor Mailboxes
Message Queue
8
invoke
onReceive
Actor

Запуск актора на Akka

```
ActorSystem system = ActorSystem.create("System");
ActorRef actorRef = system.actorOf(
        Props.create(UiActor.class, MainActivity.this));
actorRef.tell("message", ActorRef.noSender());
```

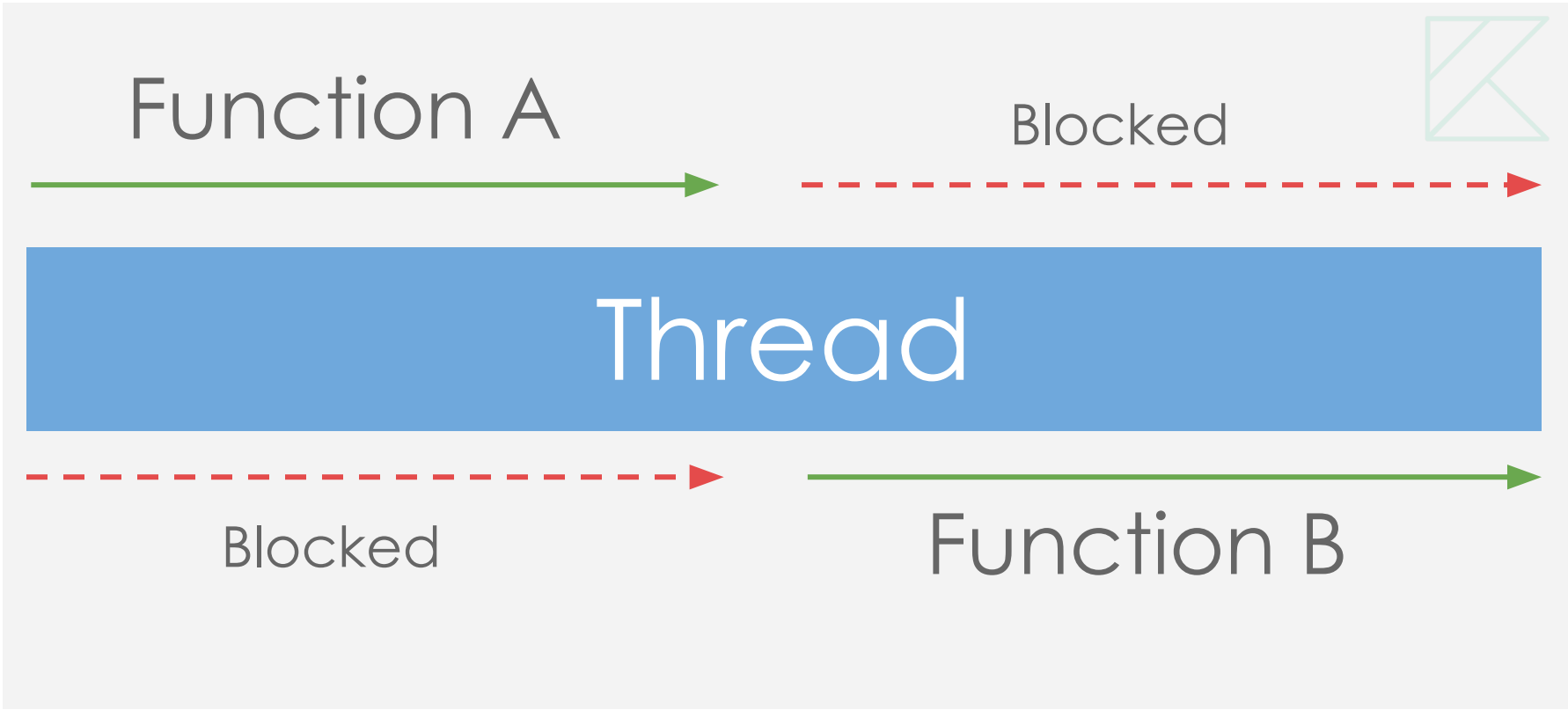
Минусы Akka

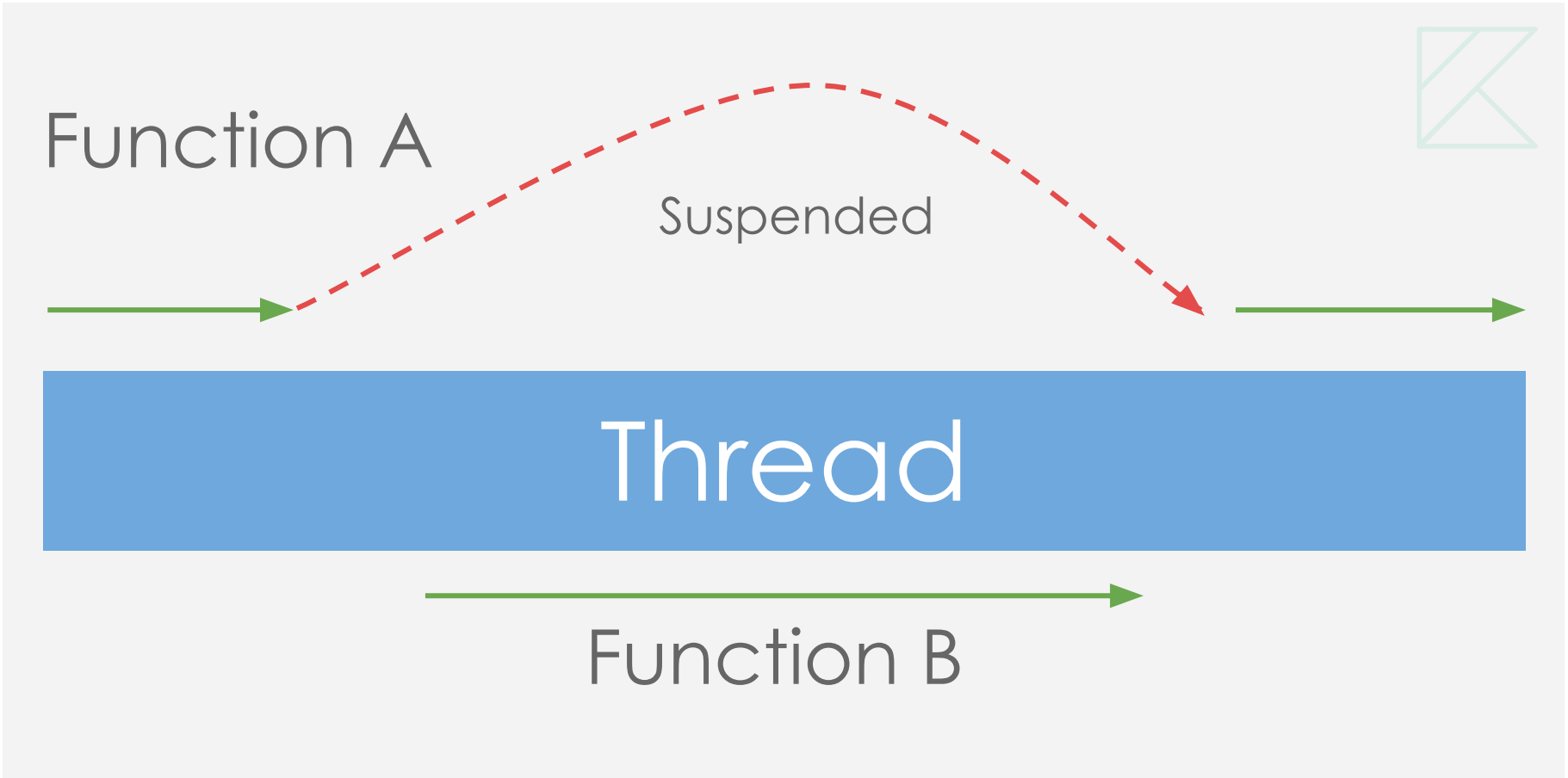
- Размер (привет, MultiDex!)
- Избыточность для android
- Написана на scala
- minSdk 24



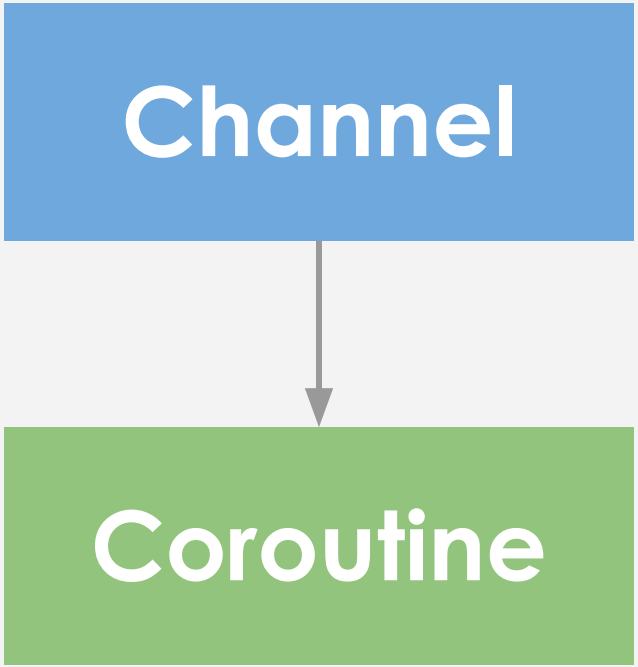
Kotlin coroutines

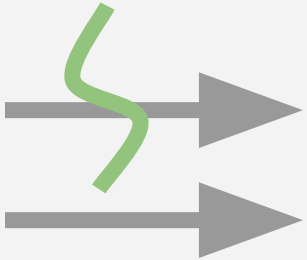
- Легковесные потоки
- Поощряют стиль без общего состояния
- Работа с помощью приостановки потоков
- Различные реализации


Function A
Blocked
Thread
Blocked
Function B


Function A
Suspended
Thread
Function B

Coroutines Actor





SendChannel<E>

suspend fun send (E)

fun offer (E) : Boolean

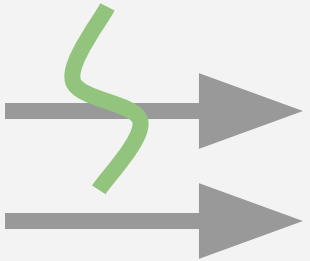
fun close (Throwable)

Channel <E>

suspend fun receive () : E

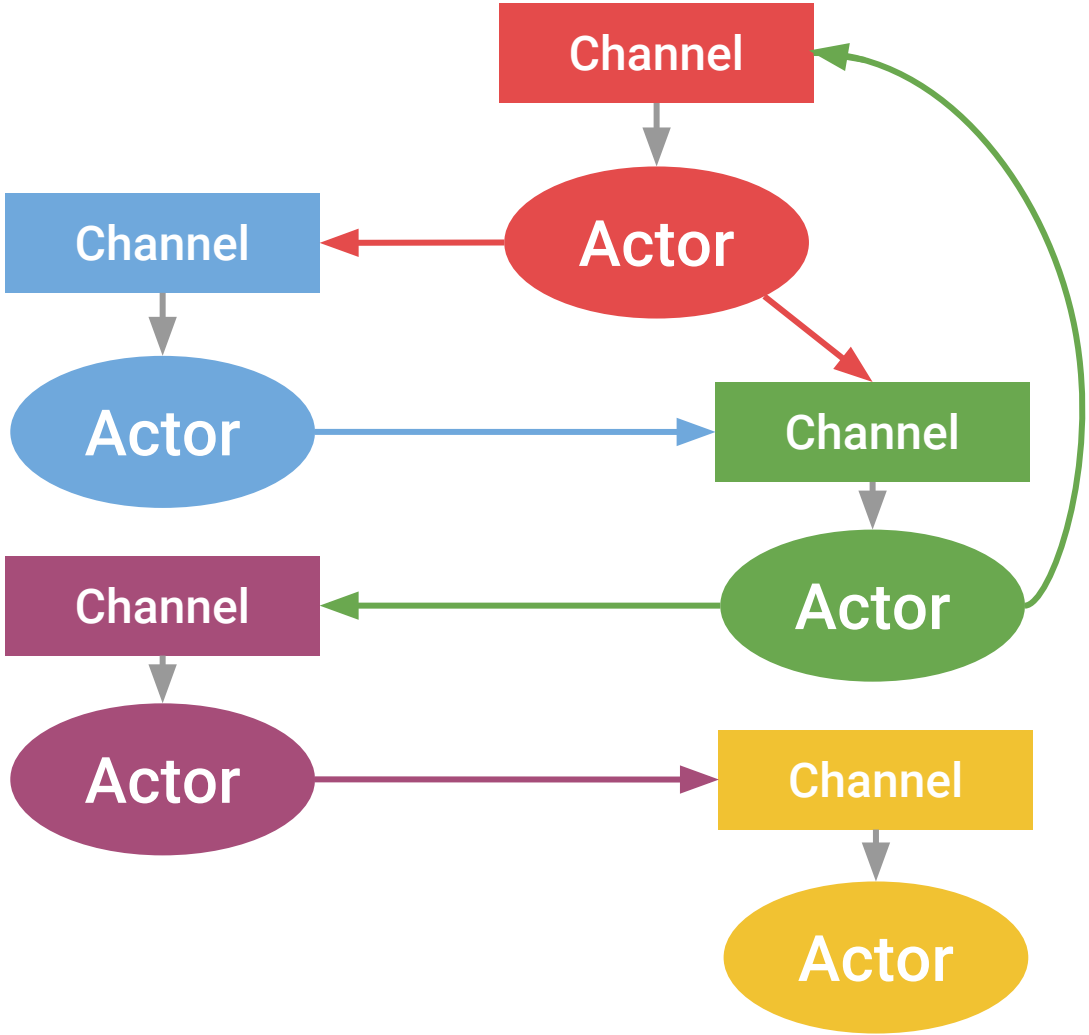
fun poll () : E?

ReceiveChannel<E>



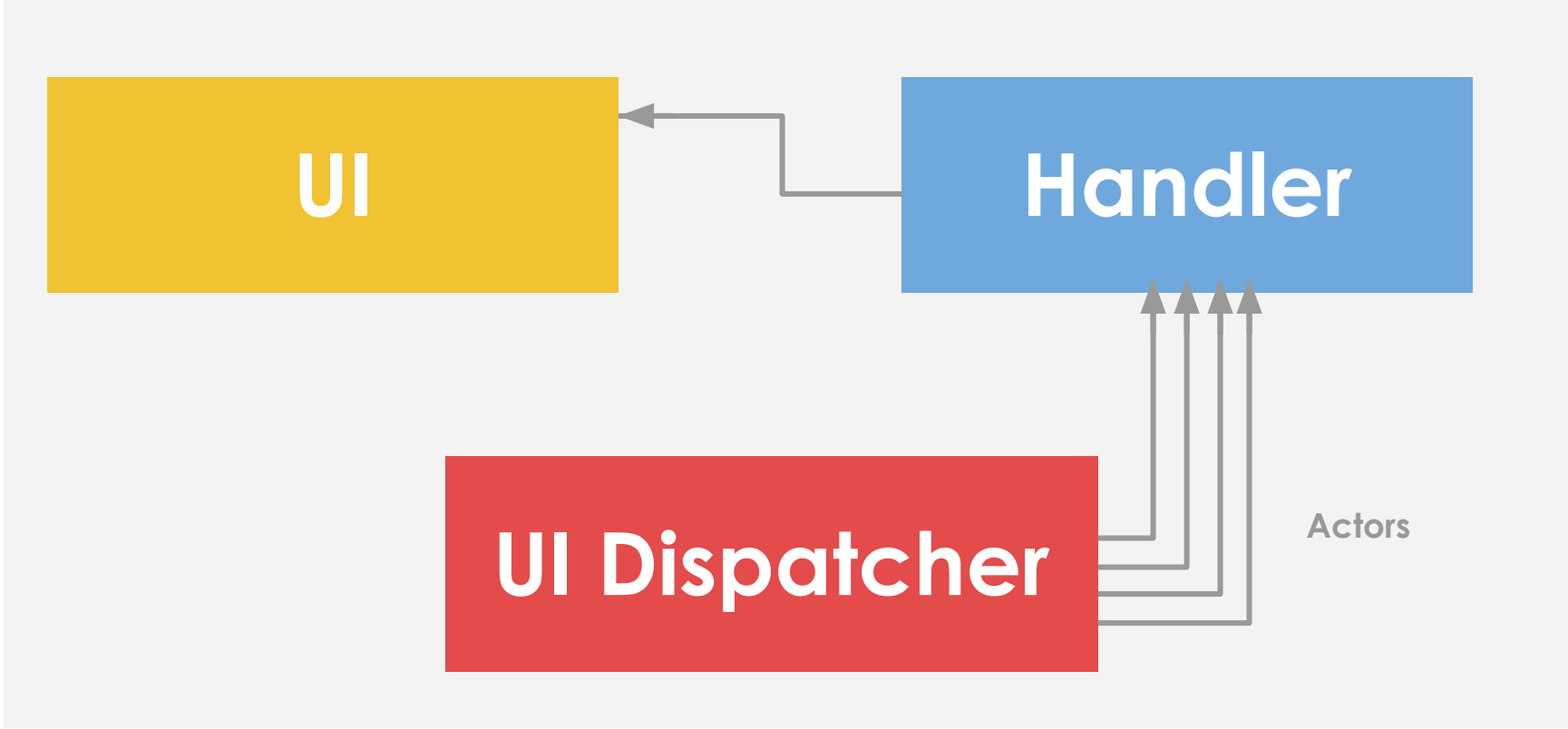
Coroutines Actors

- Именованная корутина + очередь сообщений
- Очередь используется для обмена данными
- Можно задавать определенный пул потоков

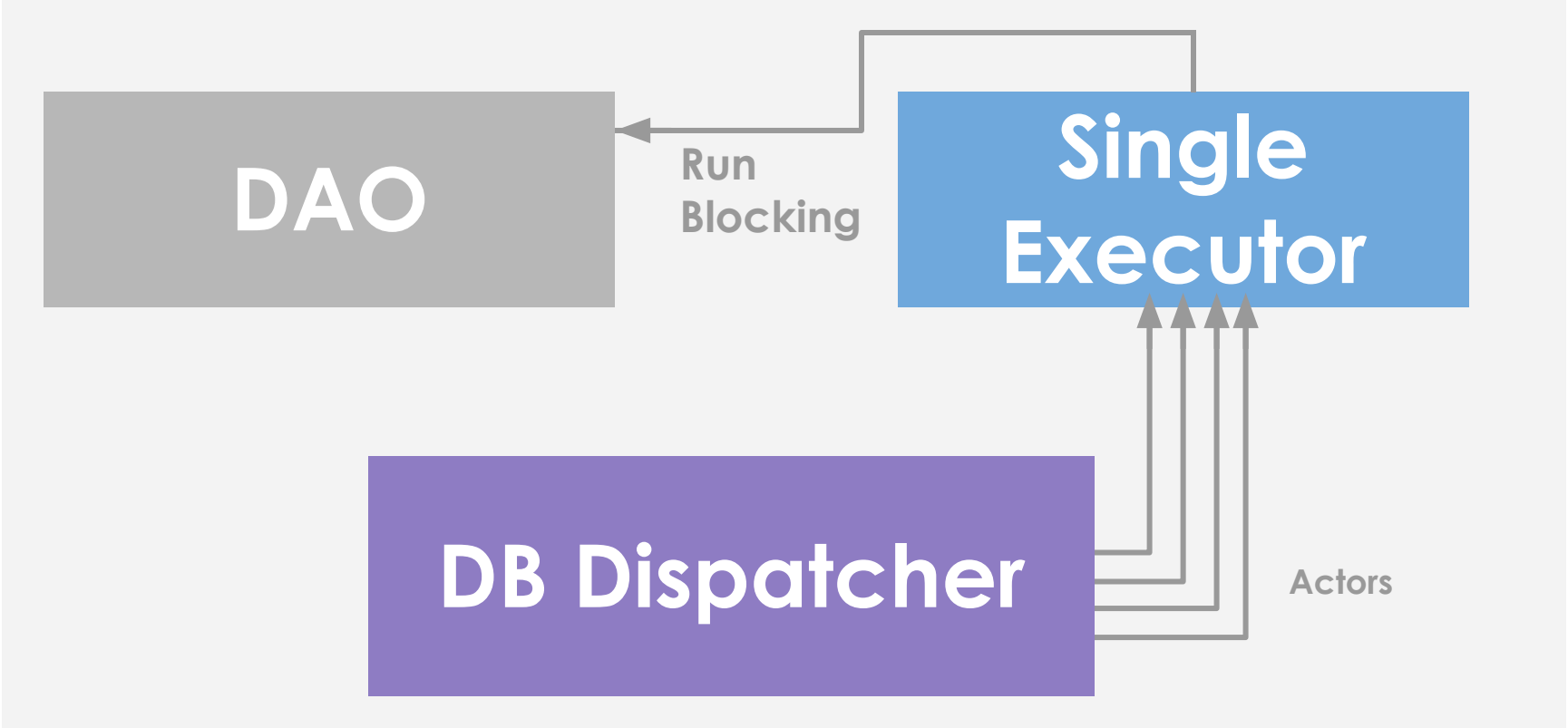

Channel
Actor
Channel
Actor
Channel
Actor
Channel
Actor
Channel
Actor

```
private fun multiplyActor() = actor<Any>(CommonPool) {
    for (msg in channel) {
        when (msg) {
            is Int -> {
                print("koroutine ")
                println(Calculator().multipleX10(msg))
            }
            else -> {
                print("Unknown")
            }
        }
    }
}
```

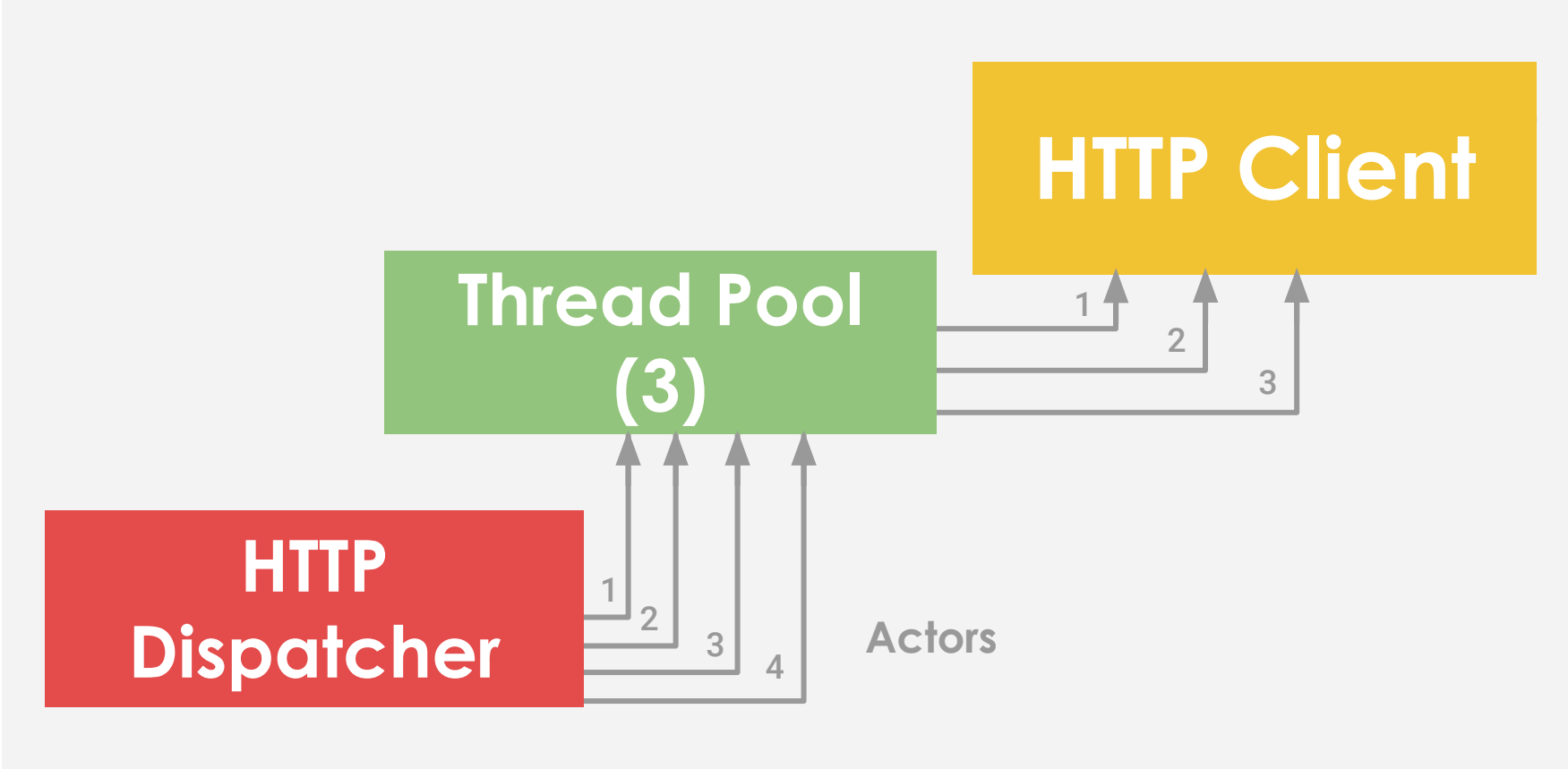
UI Dispatcher



DB Dispatcher

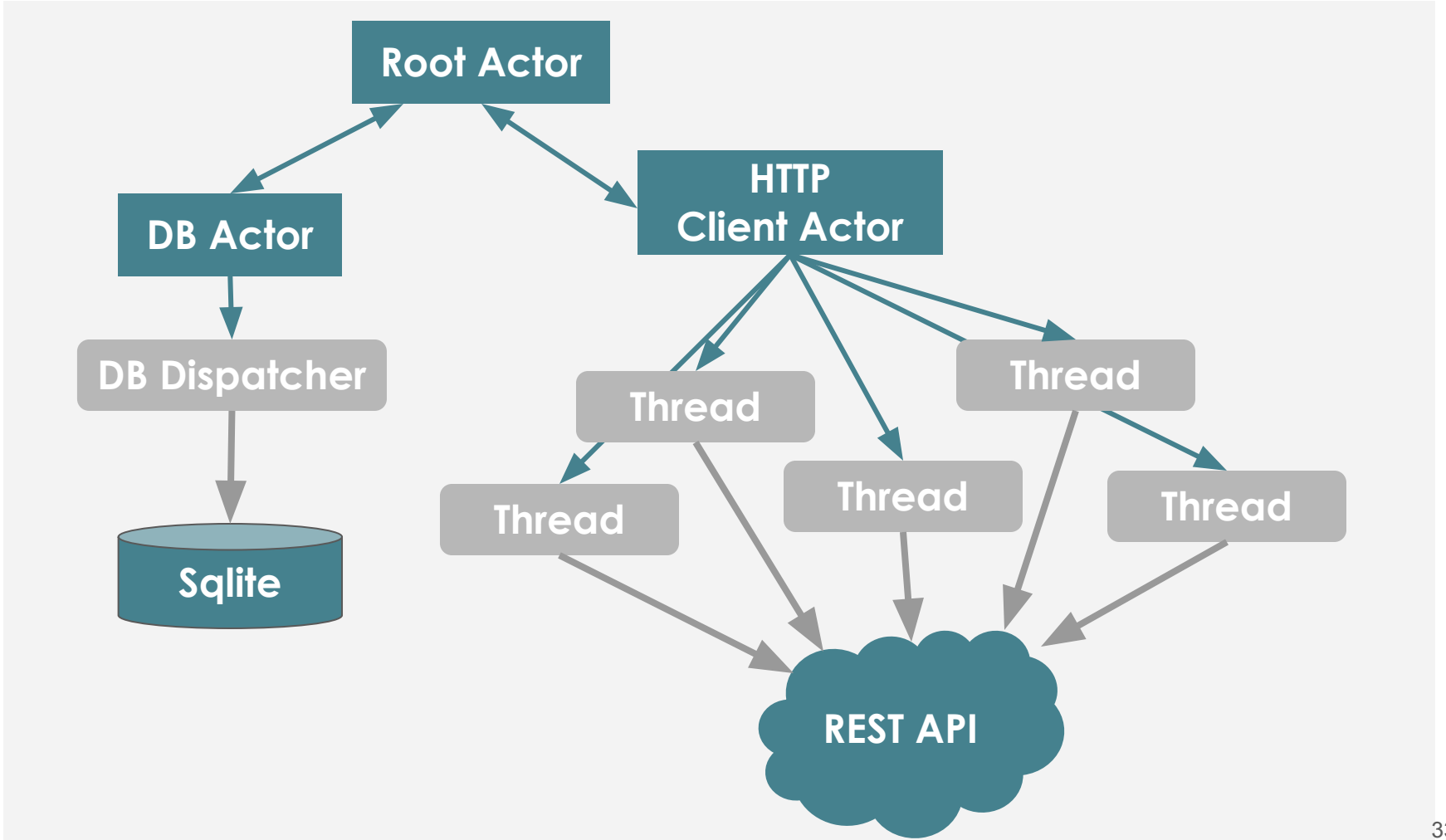


HTTP Dispatcher



Запуск актора на корунах

```
override fun setNumber(num: Int) {
    launch {
        multiplyActor().send(num)
    }
}
```


Root Actor
DB Actor
HTTP
Client Actor
DB Dispatcher
Thread
Thread
Thread
Thread
Thread
Sqlite
REST API

Тестирование акторов

```
@Test
fun actorTest() {
    val mock = mock(Calculator::class.java)
    `when`(mock.multipleX10(10)).thenReturn(100)
    val multipleActor =
ActorSystem.createActor(ActorSystem.TypeActors.MULTIPLE_ACTO
R)
    multipleActor?.sendBlocking("some string")
    verify(mock, never()).multipleX10(10)
    multipleActor?.sendBlocking(10)
    Mockito.verify(mock, times(1))
}
```

Минусы

- Нет возможности создать акторную систему
- Нужно дорабатывать вручную реализацию
- Нет инфраструктуры для тестирования

Заключение

+ Легковесные обработчики данных
+ Простота в поддержке
+ Изолированность логики
+ Нет ограничений в применении

Заключение

- Нет зарекомендованных средств
- Зависимость от реализации
- Трудность перехода

Внимание!

Спасибо за внимание!

Q&A*

* Не тестирование