Многопоточность в iOS

Специфика и лучшие практики

Сбербанк

Владимир Озеров

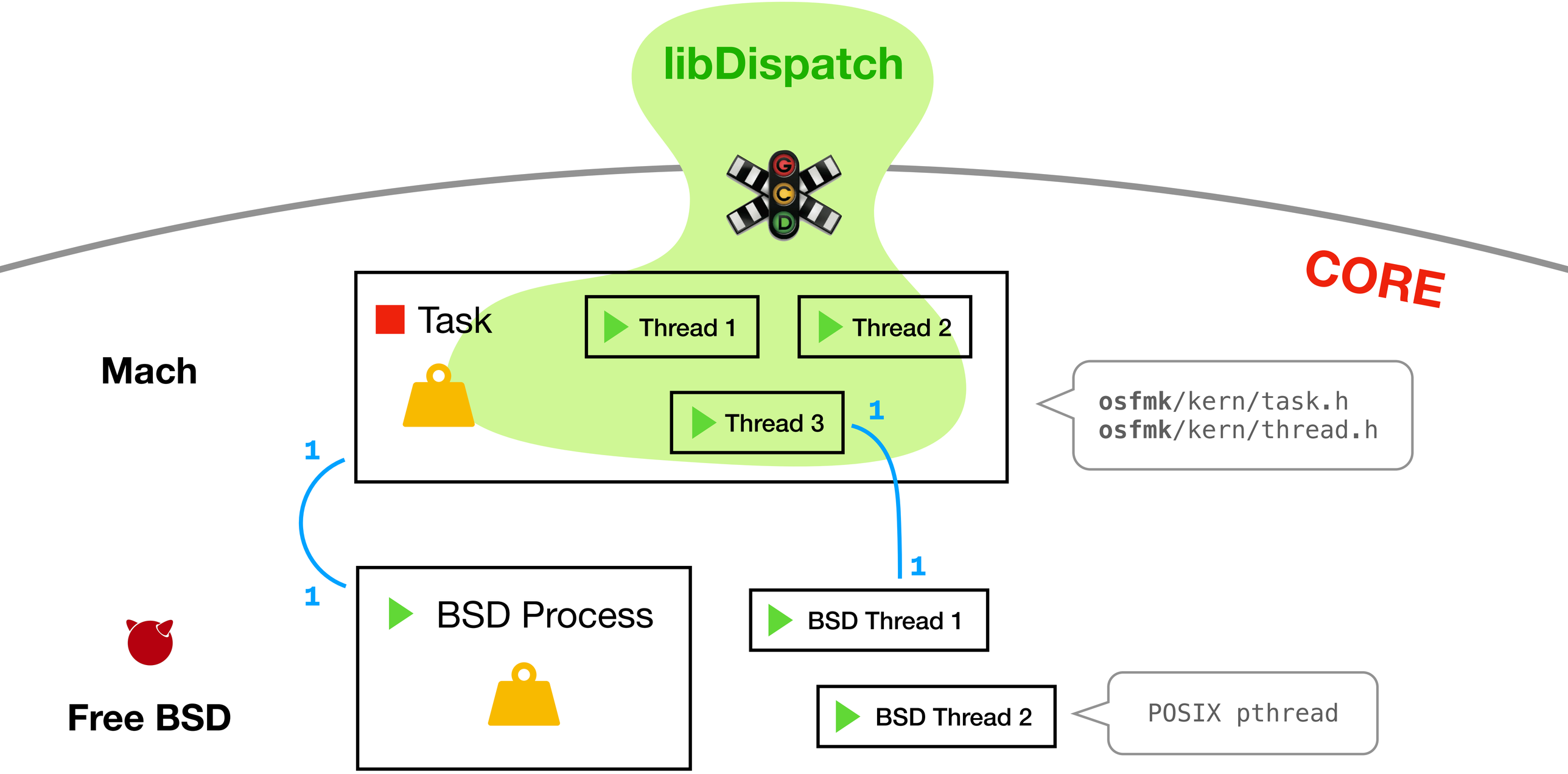
О чем я НЕ расскажу

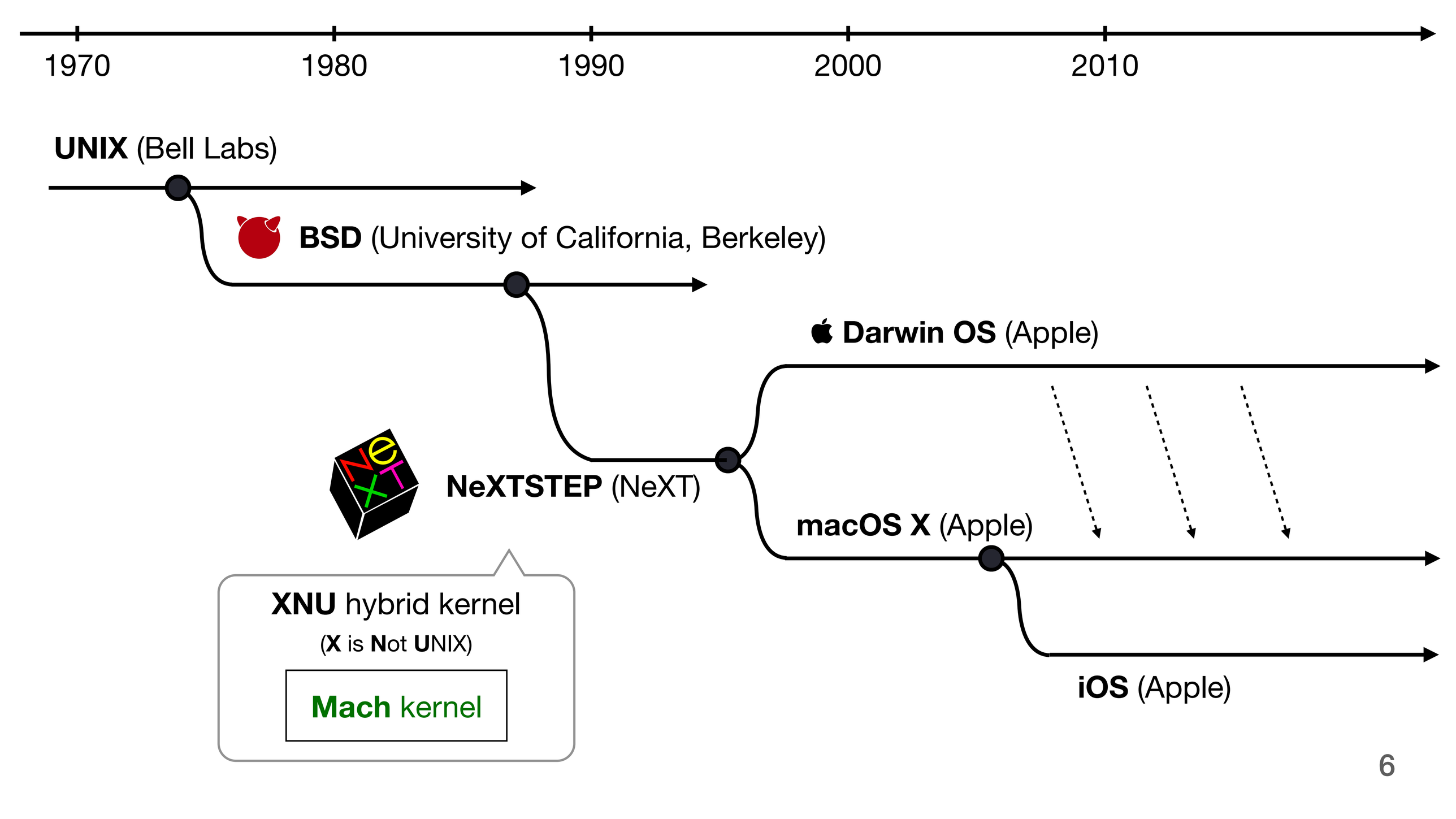
- Различные способы создания потоков в iOS
- Как пользоваться GCD
- Классические задачи и инструменты синхронизации
- Асинхронность на уровне языка программирования (Async/Await)

О чем это выступление?

- Кратко: потоки в iOS
- Lazy loading в фоне
- RunLoop и блокировка главного потока
- Минимизация рисков в многопоточном коде
- Динамическая приоритезация потоков

Кратко о потоках в iOS


libDispatch
CORE
Mach
Task
Thread 1
Thread 2
Thread 3
1
1
1
1
osfmk/kern/task.h
osfmk/kern/thread.h
BSD Process
BSD Thread 1
BSD Thread 2
POSIX pthread
Free BSD


1970
1980
1990
2000
2010
UNIX (Bell Labs)
BSD (University of California, Berkeley)
Darwin OS (Apple)
NeXTSTEP (NeXT)
macOS X (Apple)
iOS (Apple)
XNU hybrid kernel
(X is Not UNIX)
Mach kernel

Address Space

Task / Thread

Port

IPC

Mach

Carnegie
Mellon
University

OSFMK

Open
Software
Foundation

XNU

NeXT
X is Not Unix

I/O Kit

Darwin

Utilities

Drivers

Shells

Apache

X11

Free BSD OS

University Of California, Berkeley

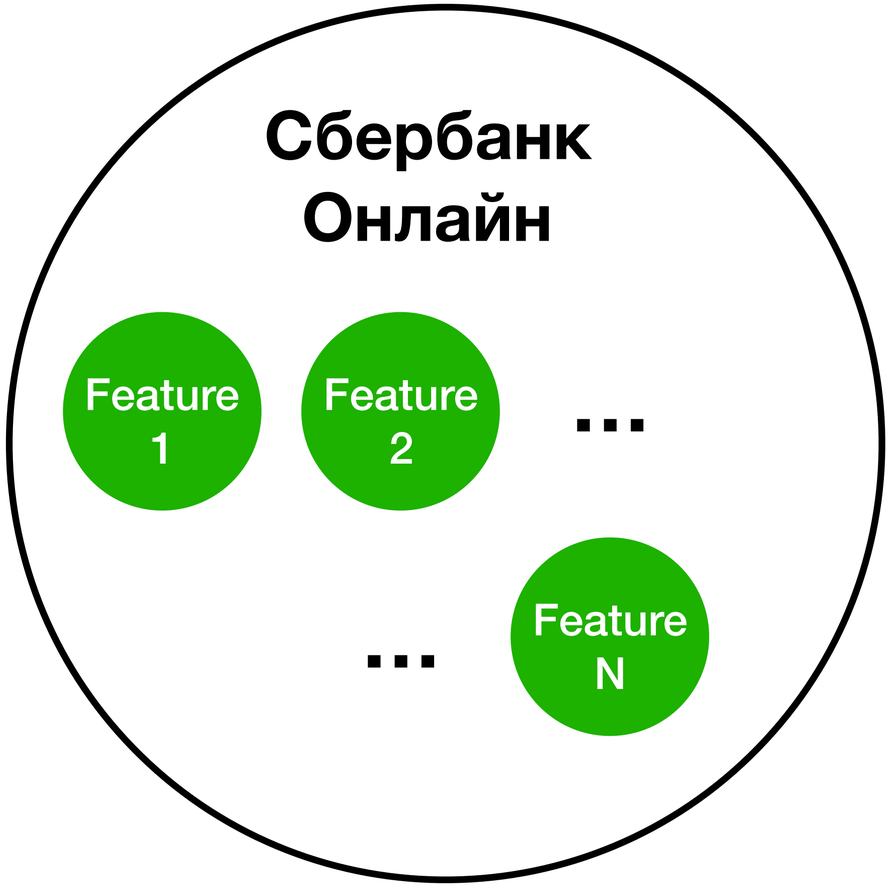
POSIX API

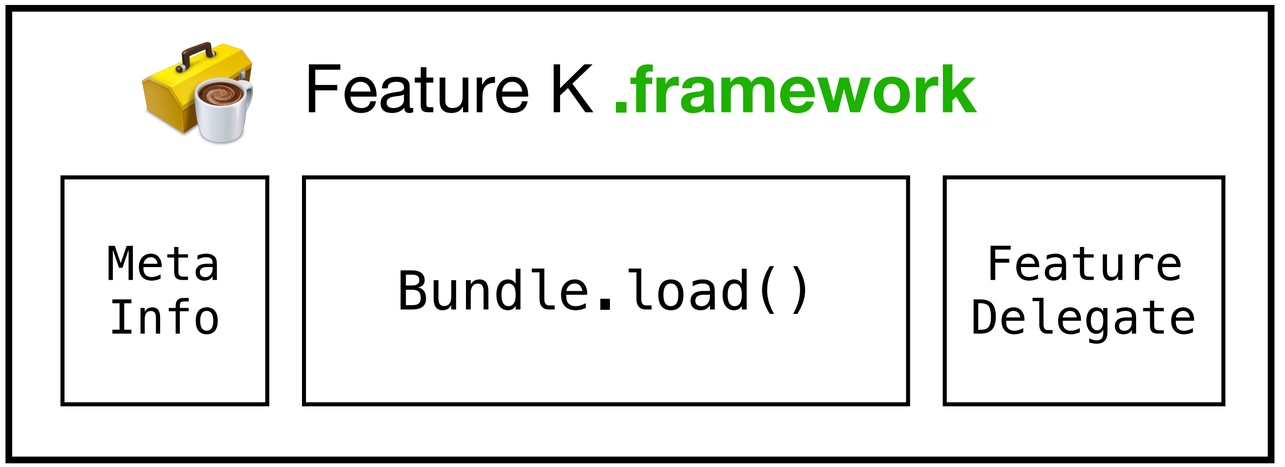
```
Thread 1 name:  com.apple.uikit.eventfetch-thread
Thread 1:
0   libsystem_kernel.dylib      0x00000001a5a34c04 mach_msg_trap + 8
1   libsystem_kernel.dylib      0x00000001a5a34020 mach_msg + 76
2   CoreFoundation              0x00000001a5be6aa8 __CFRunLoopServiceMachPort + 220
3   CoreFoundation              0x00000001a5be1940 __CFRunLoopRun + 1428
4   CoreFoundation              0x00000001a5be1084 CFRunLoopRunSpecific + 480
5   Foundation                  0x00000001a5f263d0 -[NSRunLoop+ 33744 (NSRunLoop) runMode:beforeDate:] + 232
6   Foundation                  0x00000001a5f262a8 -[NSRunLoop+ 33448 (NSRunLoop) runUntilDate:] + 92
7   UIKitCore                   0x00000001a9ded170 -[UIEventFetcher threadMain] + 156
8   Foundation                  0x00000001a5f25034 -[NSThread main] + 40
9   Foundation                  0x00000001a6060a8c __NSThread__start__ + 852
10  libsystem_pthread.dylib     0x00000001a5975d50 _pthread_start + 128
11  libsystem_pthread.dylib     0x00000001a597dc88 thread_start + 8
```

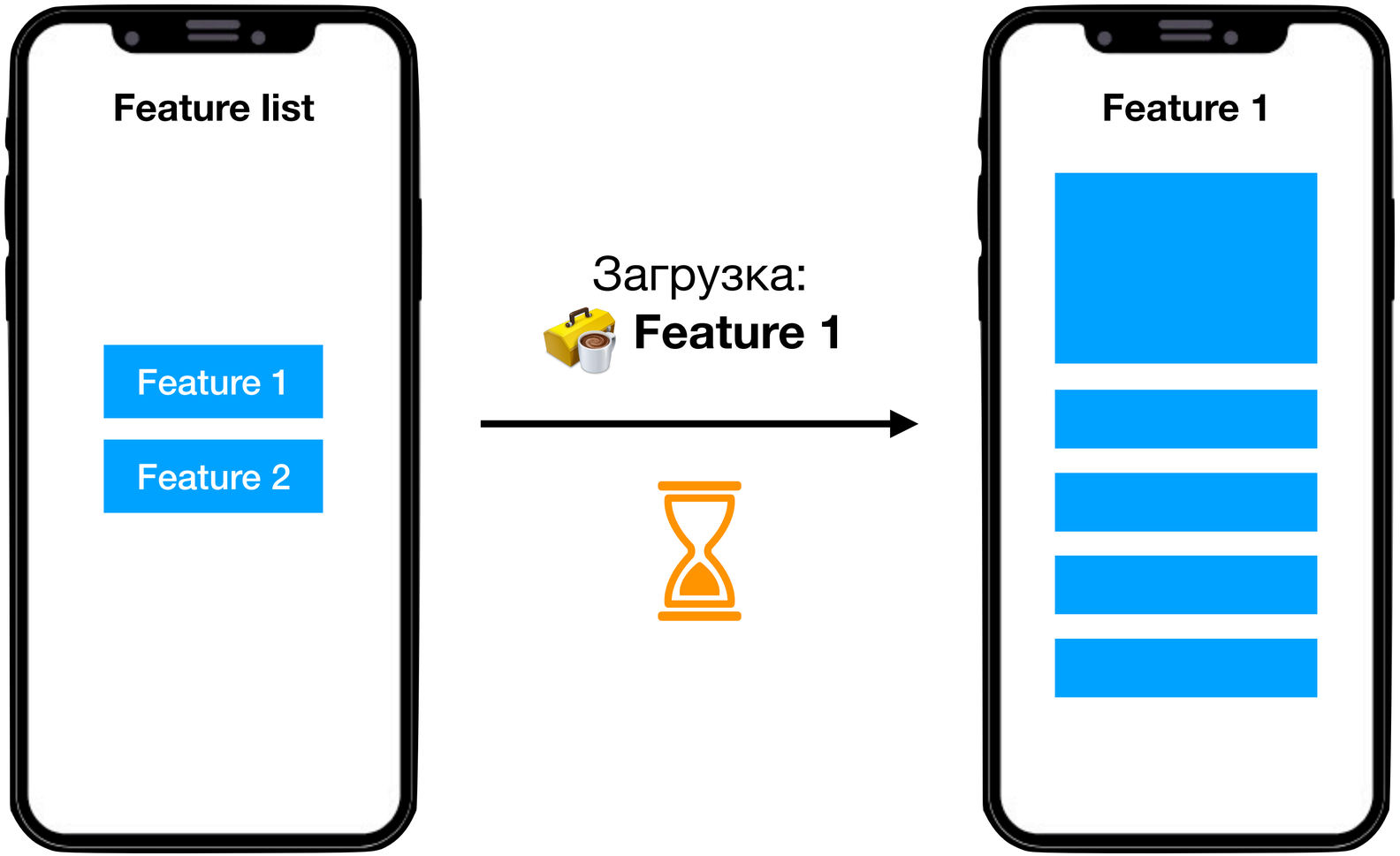
IPC

Message
+
Port

Lazy loading в фоне


Сбербанк
Онлайн
Feature
1
Feature
2
...
...
Feature
N

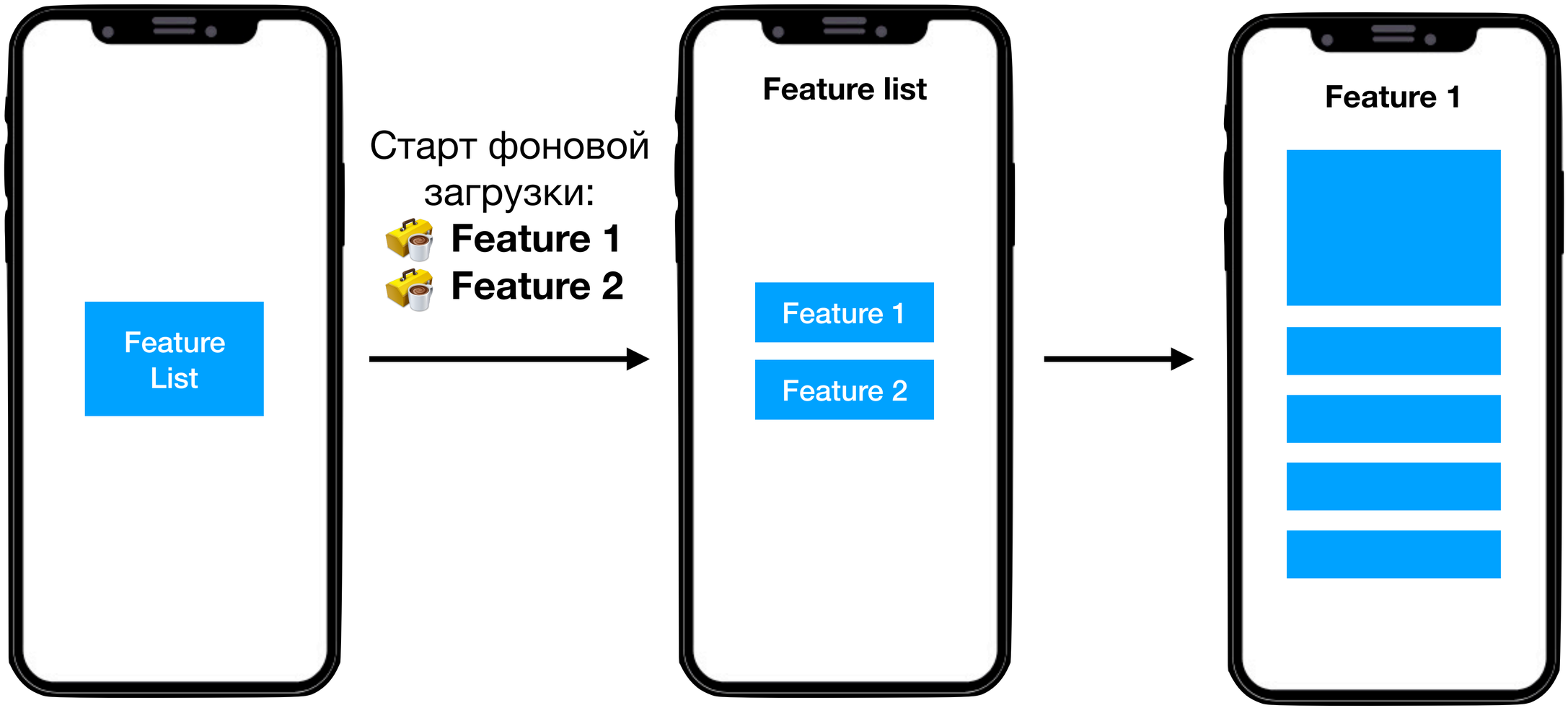

Feature K .framework
Meta
Info
Bundle.load()
Feature
Delegate

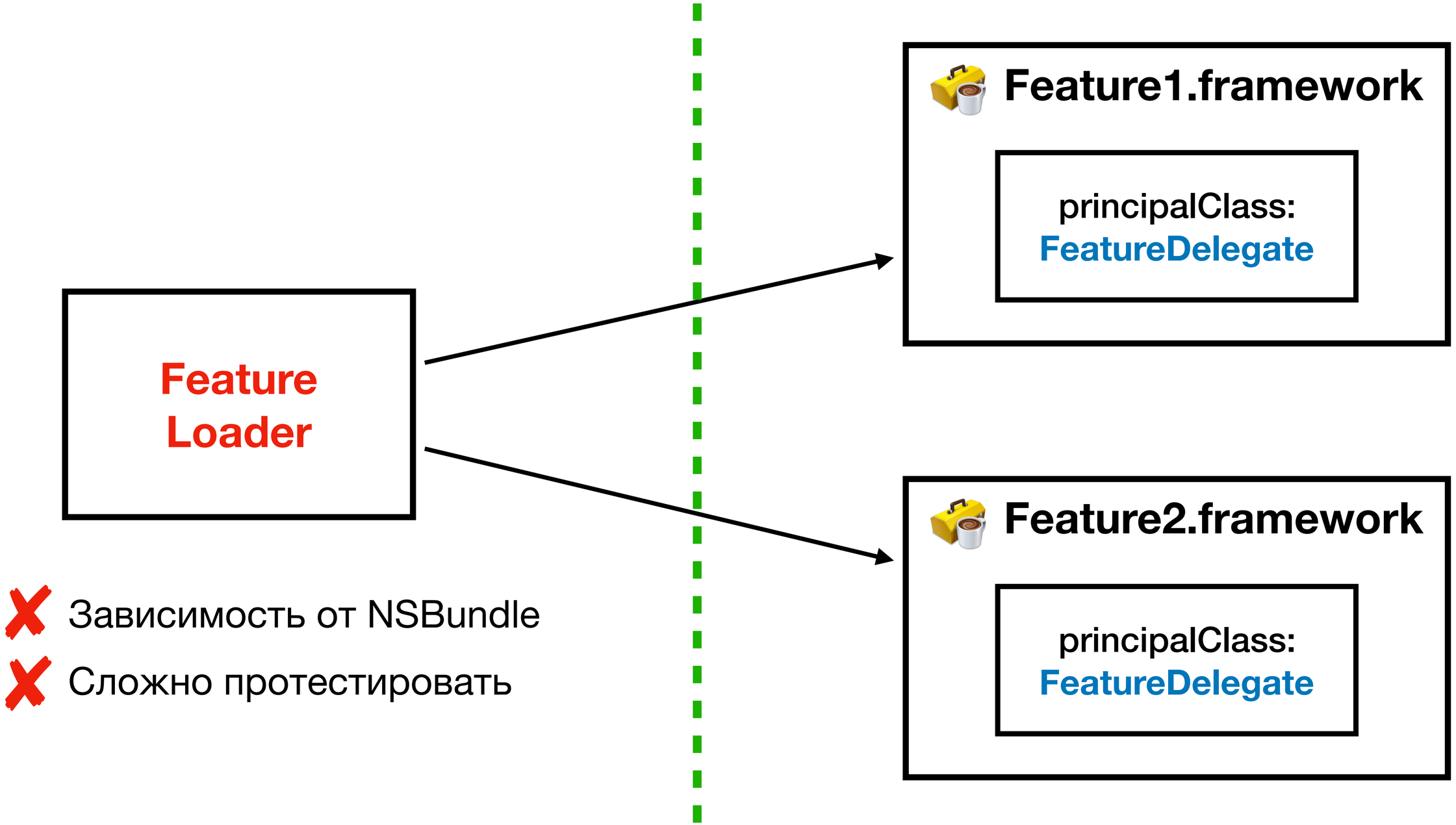

Feature list
Feature 1
Feature 2
Загрузка:
Feature 1
Feature 1

... / TheApplication.app / Frameworks / Feature1.framework

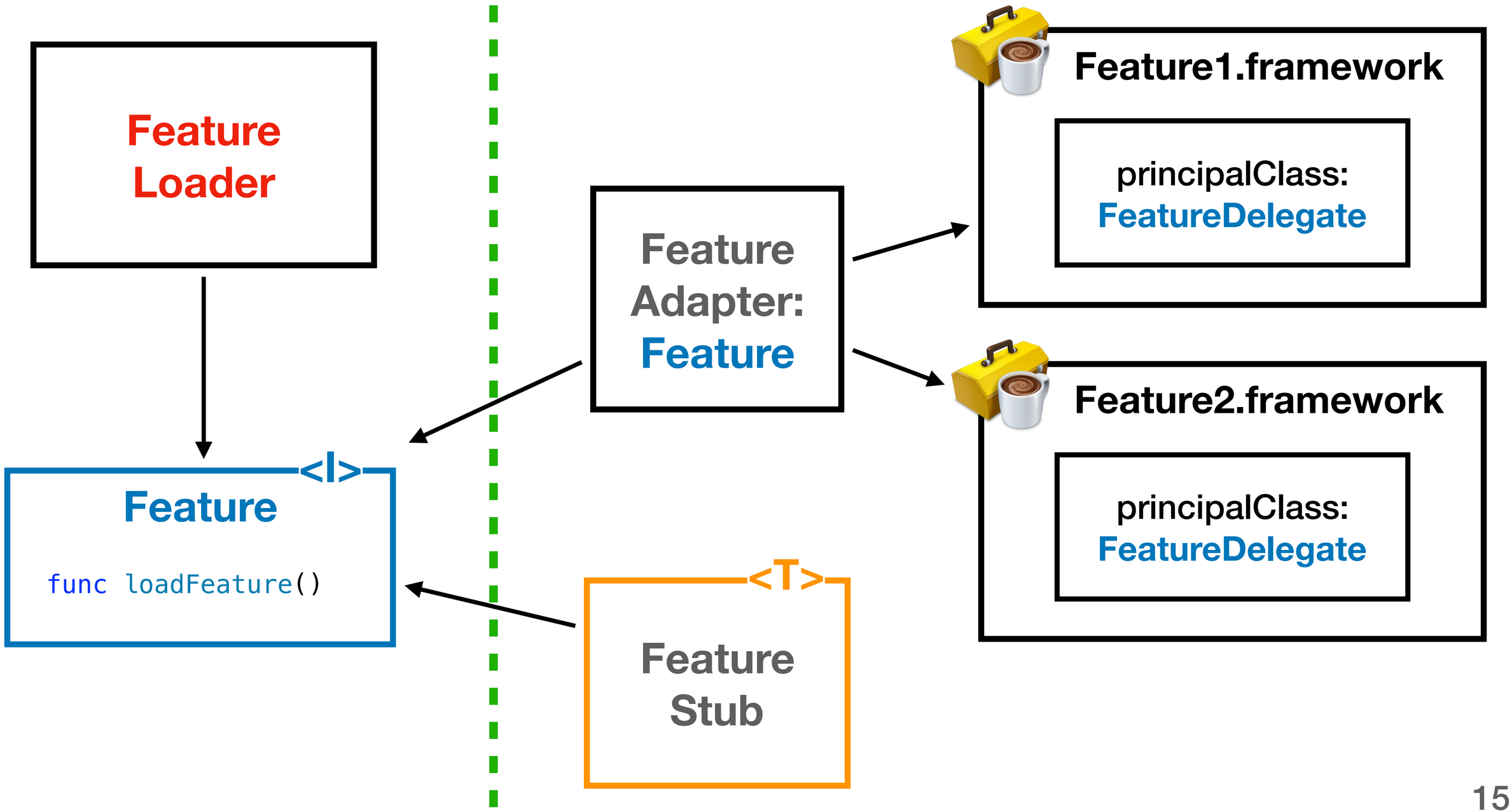
```
0  func loadFeature(path: String) -> UIViewController {
1
2      let featureBundle = Bundle(path: path)
3      featureBundle?.load()
4
5      let featureEndpoint = featureBundle?.principalClass as? FeatureDelegate
6      featureEndpoint?.featureDidFinishLoading()
7
8      let viewController = featureEndpoint?.createViewController()
9      return viewController
10 }
```

Контракт для всех фичей


Feature List
Старт фоновой загрузки:
Feature 1
Feature 2
Feature list
Feature 1
Feature 2
Feature 1



- ✗ Зависимость от NSBundle
- ✗ Сложно протестировать

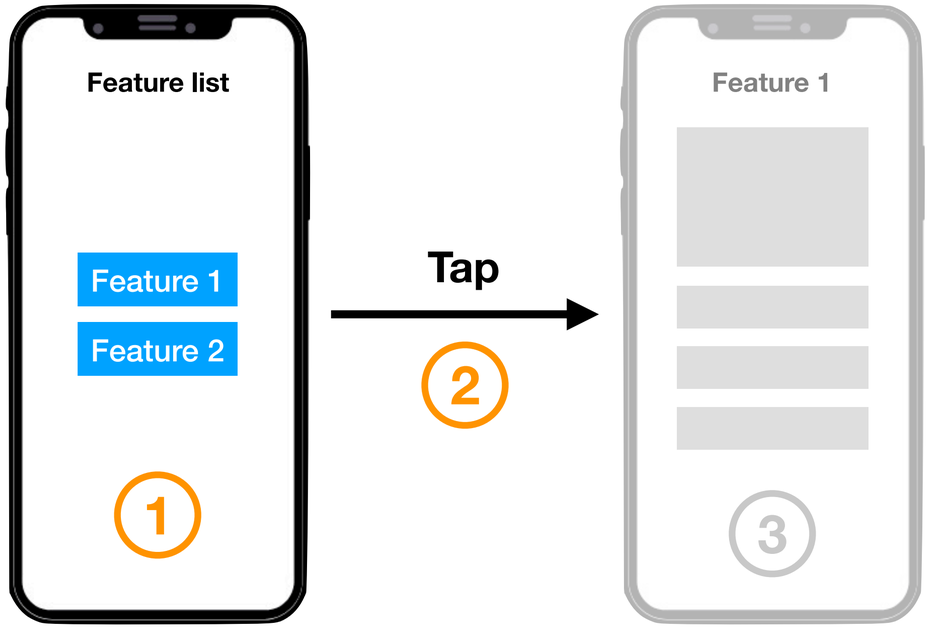

Feature
Loader
Feature
Adapter:
Feature
<I>
Feature
func loadFeature()
<T>
Feature
Stub
Feature1.framework
principalClass:
FeatureDelegate
Feature2.framework
principalClass:
FeatureDelegate

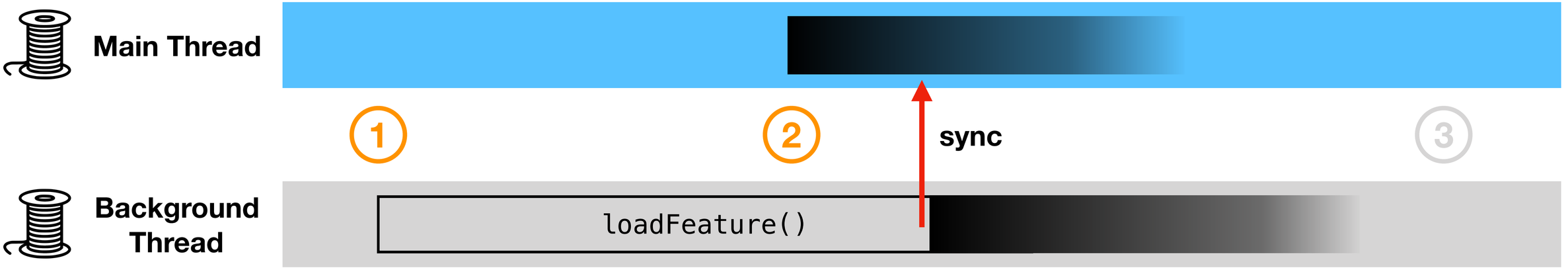
Принцип единой ответственности

«Многопоточные архитектуры достаточно сложны, чтобы их можно было рассматривать как причину изменения сами по себе, а следовательно, **они должны отделяться от основного кода**».

Чистый код: создание, анализ и рефакторинг — Роберт Мартин

RunLoop
и
блокировка главного потока


Feature list
Feature 1
Feature 2
1
Tap
2
Feature 1
3

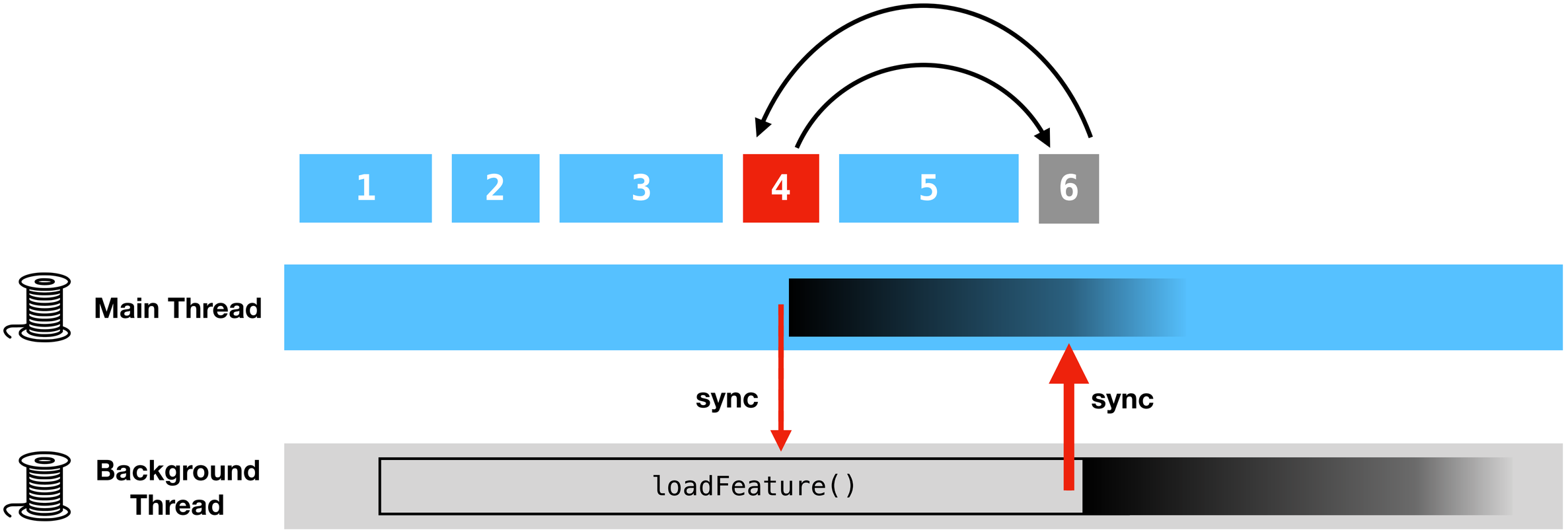

Main Thread
1
2
sync
3
Background Thread
loadFeature()

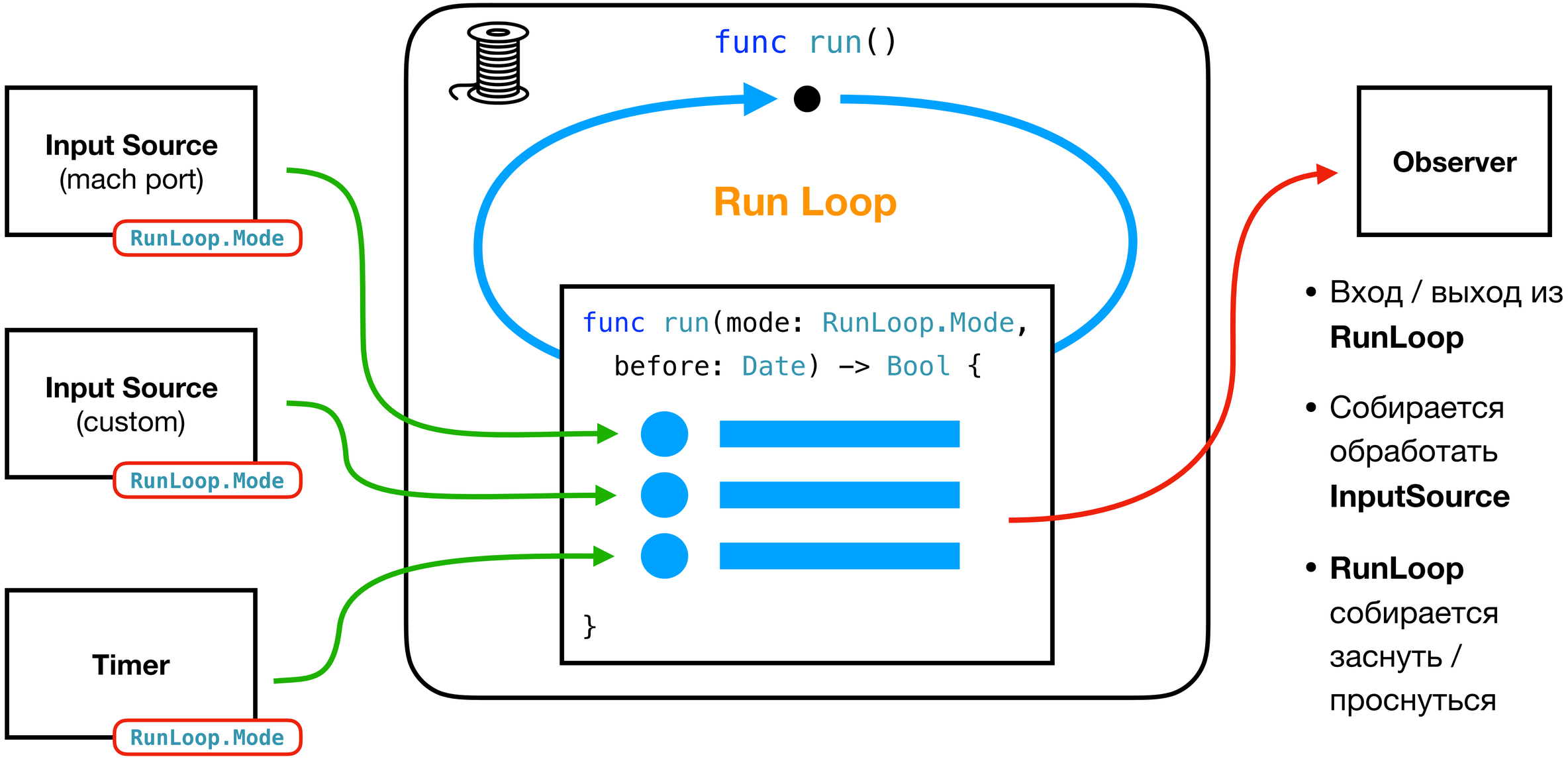
```
0  func setupCoreDataStack() {
1
2      let model = NSManagedObjectModel()
3      coordinator = NSPersistentStoreCoordinator(managedObjectModel: model)
4
5      context = NSManagedObjectContext(concurrencyType: .mainQueueConcurrencyType)
6      context?.persistentStoreCoordinator = coordinator
7      context?.mergePolicy = NSOverwriteMergePolicy
8  }
```

```
context?.performAndWait {
    ...
}
```

Знайте свой SDK

- Инструменты многопоточности
- Потоко-безопасность системных компонентов
- Специфика ~~Core Data~~


1
2
3
4
5
6
Main Thread
sync
sync
Background Thread
loadFeature()



- Вход / выход из **RunLoop**
- Собирается обработать **InputSource**
- **RunLoop** собирается заснуть / проснуться

```
func run(mode: RunLoop.Mode, before: Date) -> Bool
```

Input Source 1 — RunLoop.Mode

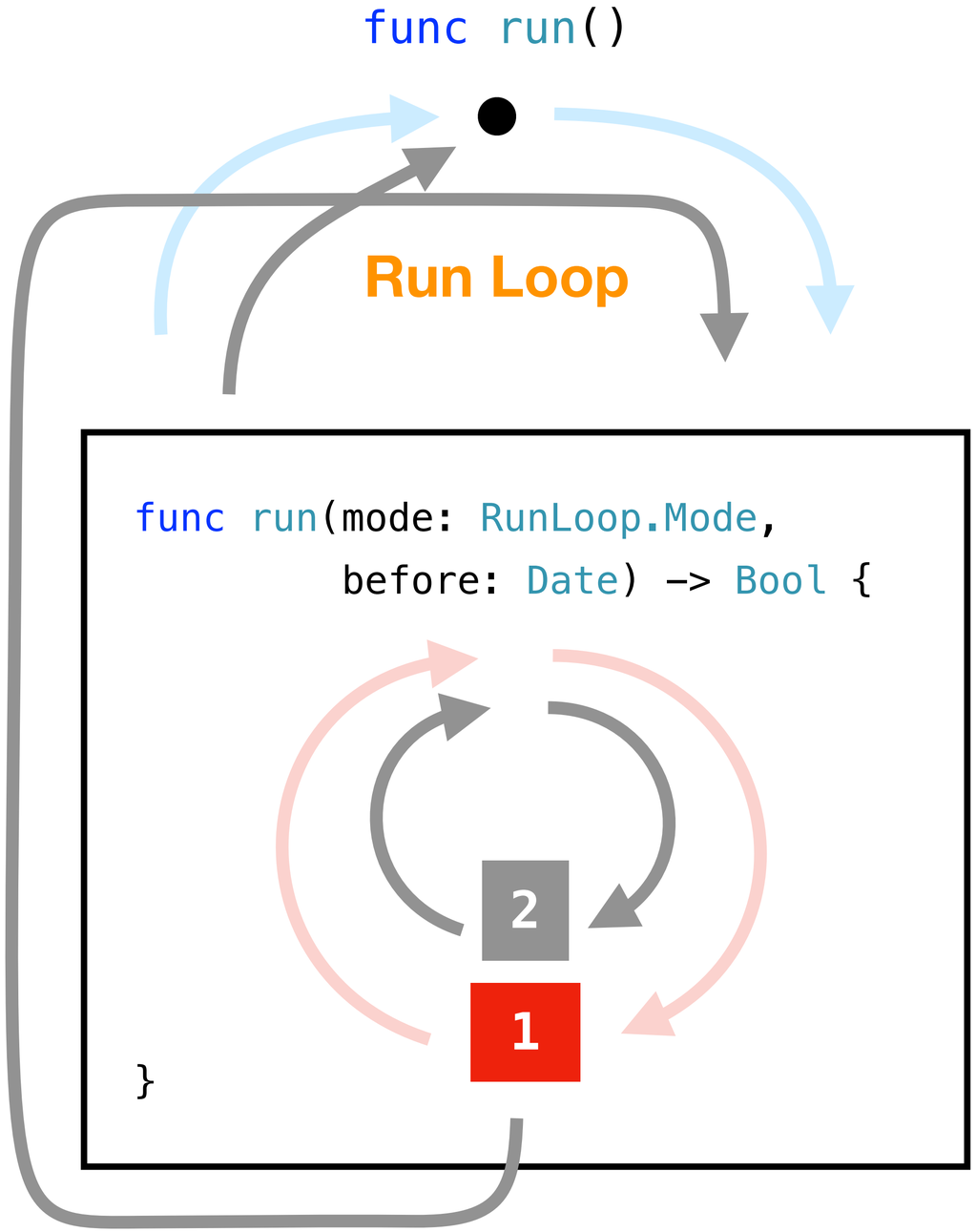
Input Source 2 — RunLoop.Mode

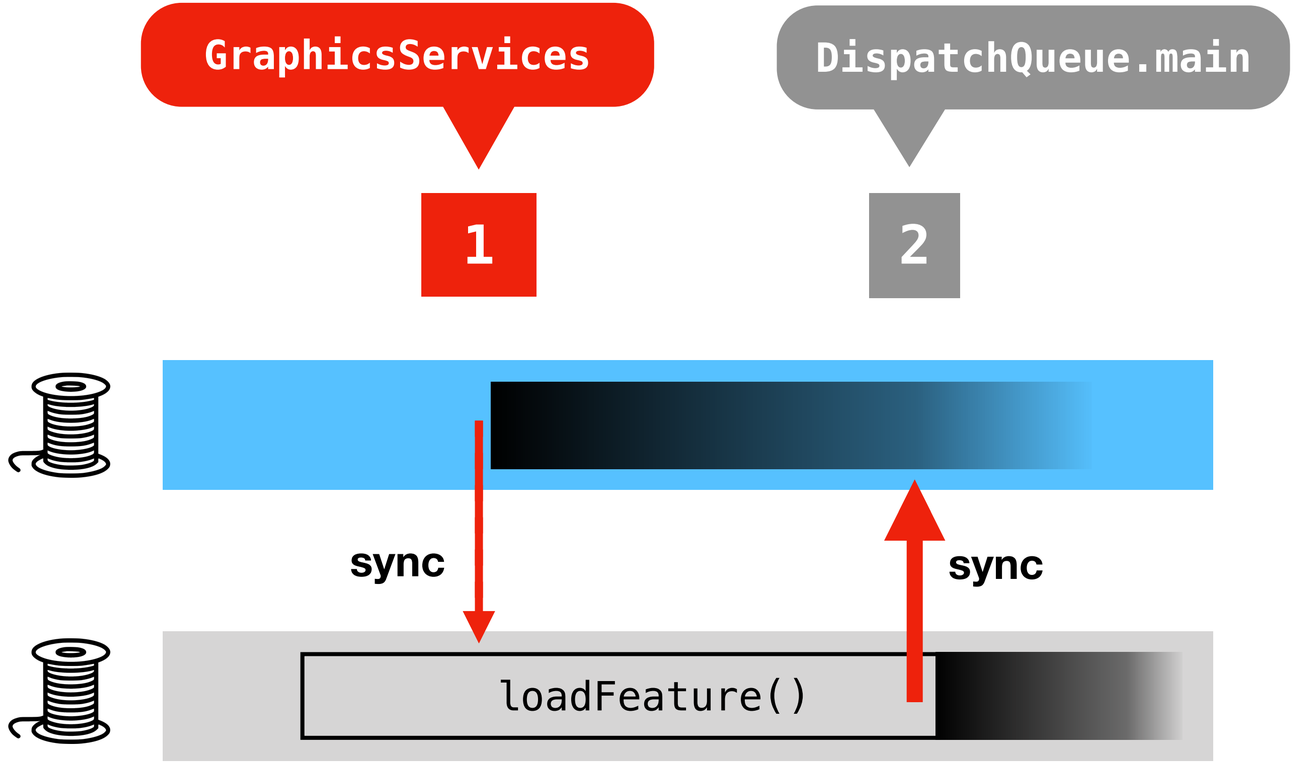
Input Source 3 — RunLoop.Mode

Input Source 4 — RunLoop.Mode

- Default
- UITracking
- GSEventReceive
- CommonModes = […]

GraphicsServices.framework






```
if !lock.try() {
    RunLoop.current.run(mode: .default,
                        before: .distantFuture)
}
```

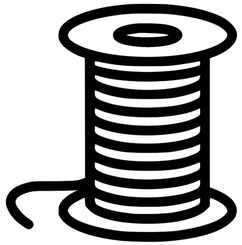
Оценка решения

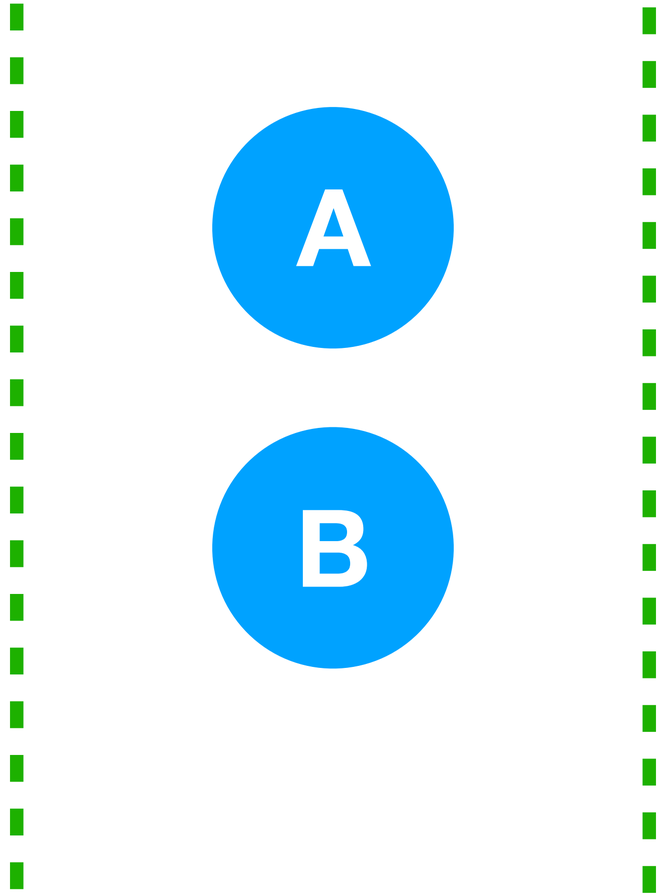
- Работает
- Мало изменений

- Сложная логика
- Непредсказуемость
- Накладываются строгие ограничения
- Создание вложенного цикла RunLoop

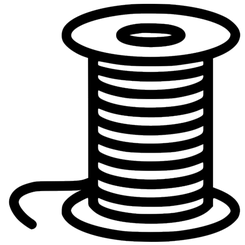
Live lock

Thread 1

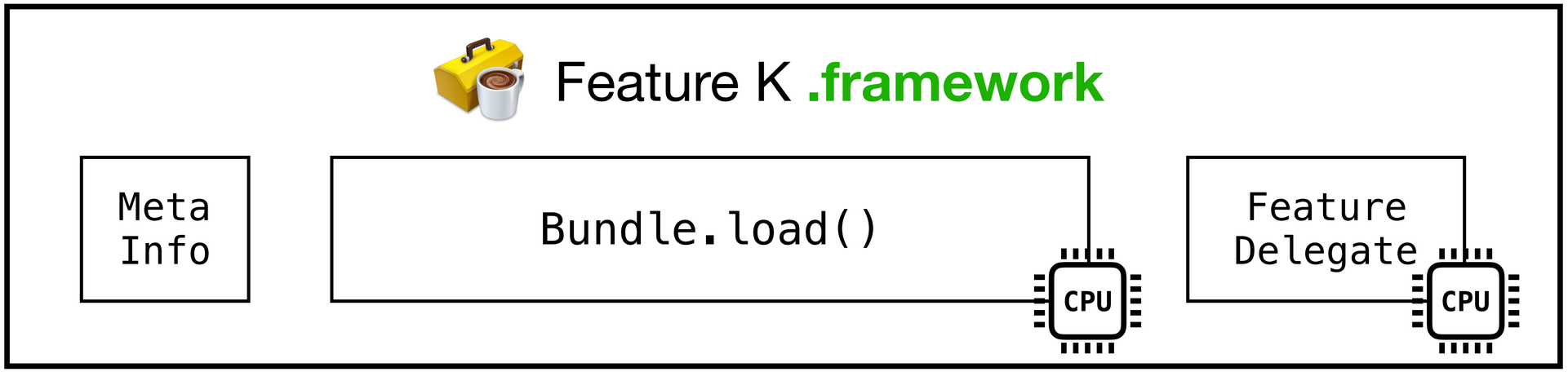




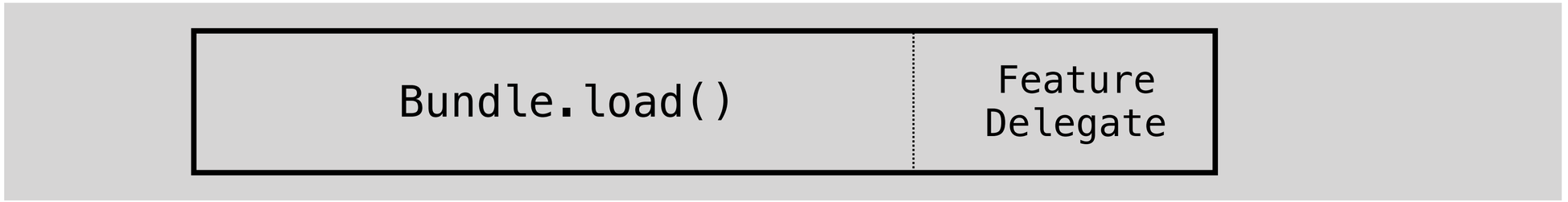
Thread 2



Минимизация рисков в многопоточном коде



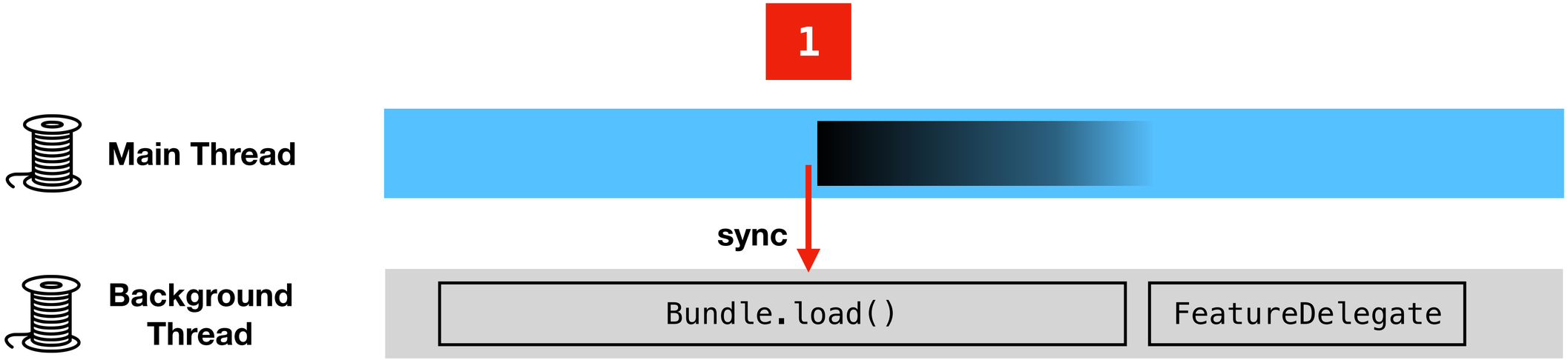
Background Thread



Загрузка динамической библиотеки

Код фичёвого фреймворка

?


1
Main Thread
sync
Background Thread
Bundle.load()
FeatureDelegate

Критические секции

- Что происходит в критической секции?
- Минимизируйте размер критической секции
- В компонентах реализуйте «серверную» блокировку

```
0  class SomeClass {
1      func lock() { ... }
2
3      func doFirstStuff() { ... }
4      func doSecondStuff() { ... }
5
6      func unlock() { ... }
7  }
```

«Клиентская» блокировка

👎

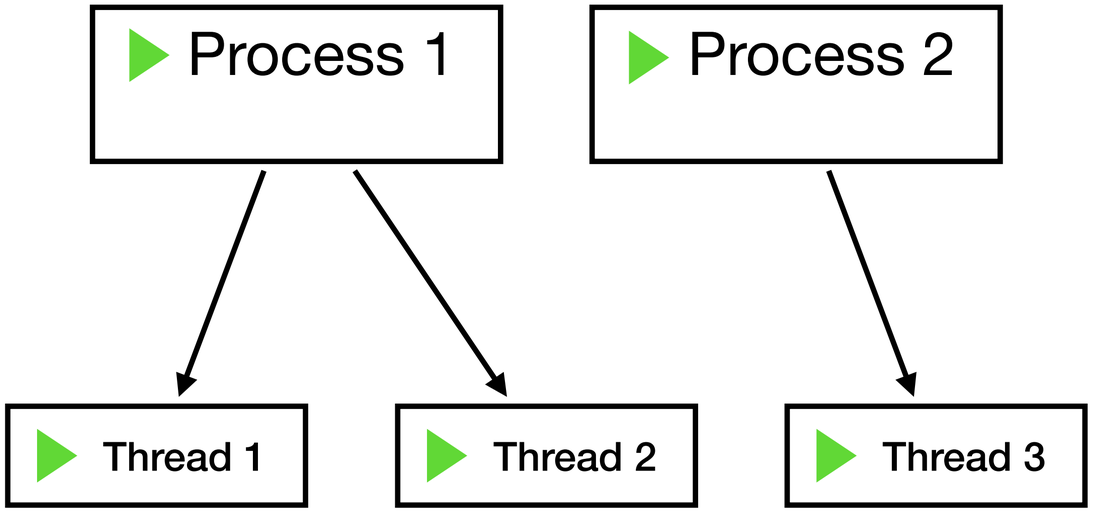
```
0  class SomeClass {
1      func doStuff() {
2          lock()
3          doFirstStuff()
4          doSecondStuff()
5          unlock()
6      }
7  }
8
9  private extension SomeClass {
10     func lock() { ... }
11
12     func doFirstStuff() { ... }
13     func doSecondStuff() { ... }
14
15     func unlock() { ... }
16 }
```

«Серверная» блокировка

👍

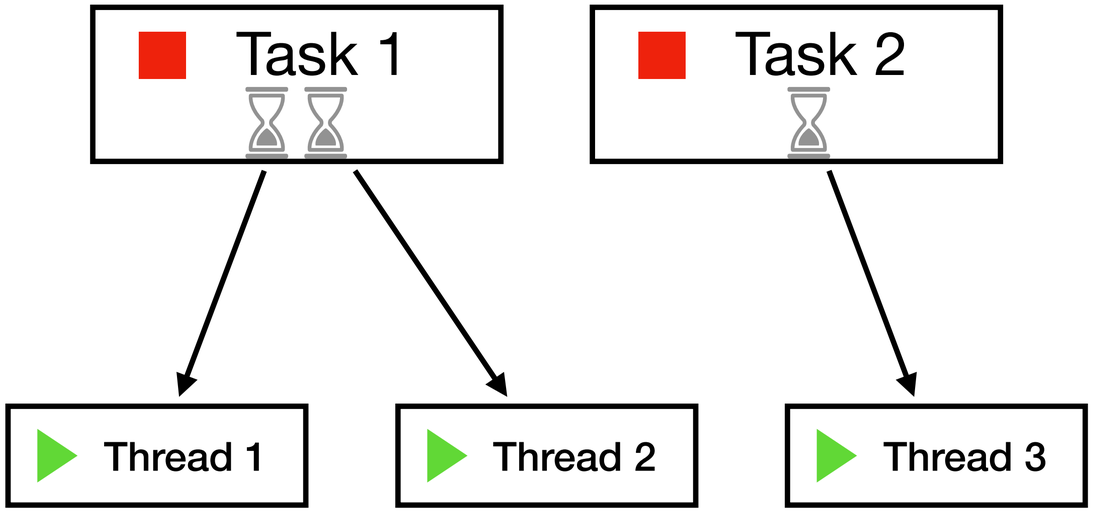
Динамическая приоритезация

Linux



«Сверху-вниз»

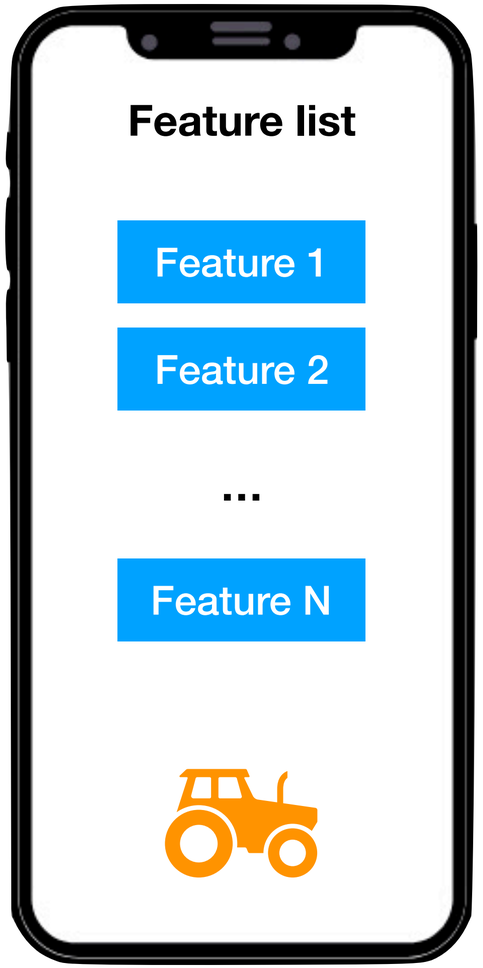
Darwin

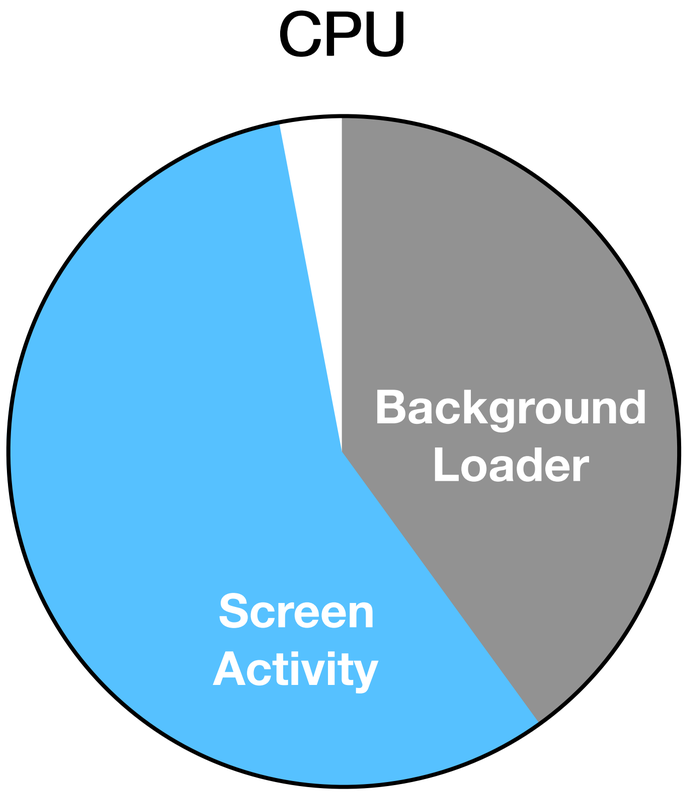


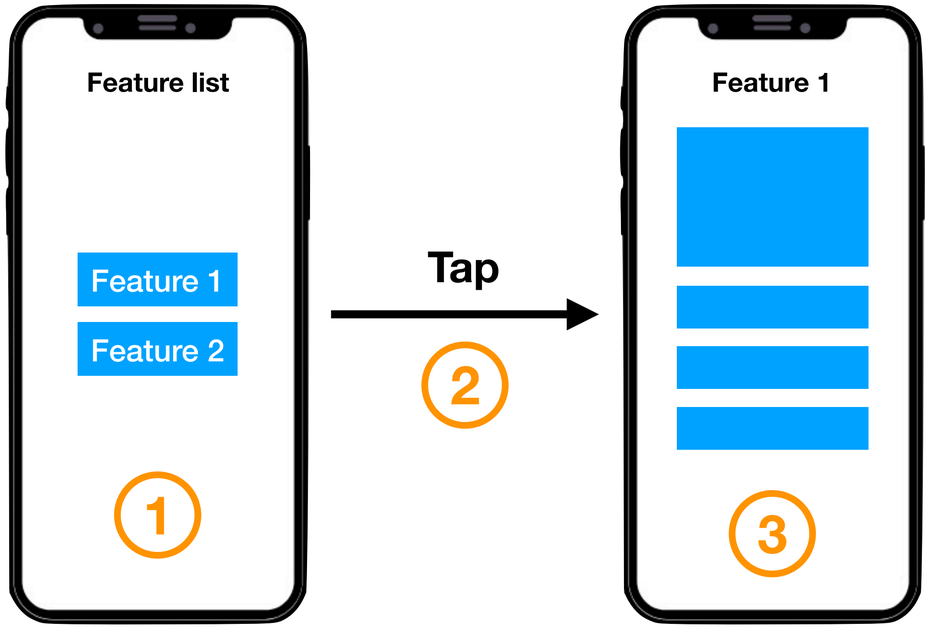
«Снизу-вверх»

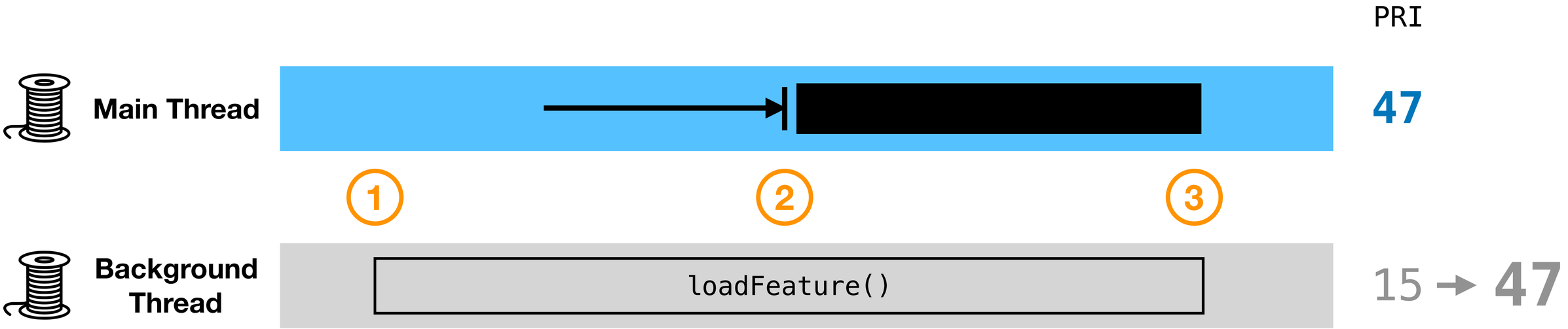
Особенности планирования в Mach

- Схема планирования - «Снизу-вверх»
- Учитываются многочисленные параметры планирования (**`osfmk/kern/thread.h`**)
- Реагирование на изменение состояния системы (**osfmk/kern/ledger.h**)


Feature list
Feature 1
Feature 2
...
Feature N


CPU
Background Loader
Screen Activity


Feature list
Feature 1
Feature 2
1
Tap
2
Feature 1
3


PRI
Main Thread
47
1
2
3
Background Thread
loadFeature()
15 → 47

Quality of Service Inference and Promotion

- Определяет **QoS** для задач в **GCD**, если имеется противоречие м/у приоритетом очереди и операции.
- Динамически изменяет приоритет очереди, в случае выявления зависимости.
- Работает по своду правил.
- Не работает, если зависимости «непрозрачные».

sysdiagnose

```
Thread 0x1cea5    DispatchQueue "com.apple.main-thread"(1)          8 samples (1-8)  priority 4  (base 4)
Thread 0x1cf47    Thread name "com.apple.uikit.eventfetch-thread"   8 samples (1-8)  priority 4  (base 4)

...

Thread 0xd835b    DispatchQueue "com.apple.main-thread"(1)          8 samples (1-8)  priority 47 (base 47)
Thread 0xd8379    Thread name "com.apple.uikit.eventfetch-thread"   8 samples (1-8)  priority 47 (base 47)
Thread 0xd8393    Thread name "com.apple.CoreMotion.MotionThread"   8 samples (1-8)  priority 47 (base 47)
Thread 0xd8394    Thread name "AVAudioSession Notify Thread"        8 samples (1-8)  priority 15 (base 15)
Thread 0xd83a2    Thread name "com.apple.NSURLConnectionLoader"     8 samples (1-8)  priority 33 (base 33)
Thread 0xd83a3    Thread name "SocketQueue.sendThreadWorker"        8 samples (1-8)  priority 31 (base 31)
```

osfmk/kern/sched.h

Многопоточность - это:

- Часть ядра ОС и системные библиотеки
- Developer SDK
- Задачи синхронизации
- Механизмы синхронизации, примитивы
- Лучшие практики

Ссылки и материалы

1. Доклад про Lazy Loading в Сбербанк Онлайн
2. Darwin-XNU GitHub
3. Grand Central Dispatch GitHub
4. The Mach Project
5. Mac OS X and iOS Internals (Jonathan Levin)
6. iOS App Reverse Engineering (Snakeninny, Hangcom)
7. Чистый код: создание, анализ и рефакторинг (Мартин Р.)
8. Apple WWDC: 2015.718, 2016.720, 2017.706
9. QoS Inference and Promotion Rules