

# Modernizing GCD Usage

## How to stay on core

Session 706

Daniel Chimene, Core Darwin  
Daniel A. Steffen, Core Darwin  
Pierre Habouzit, Core Darwin



`dispatch_async`

`dispatch_queue_create`

`DispatchQueue.concurrentPerform`

`dispatch_after`

`DispatchQueue.async`

`dispatch_sync`

`DispatchQueue.sync`

`dispatch_activate`

`DispatchSource.activate`

`dispatch_source_create`

`DispatchSource.setEventHandler`

`dispatch_once`

`dispatch_apply`

`DispatchWorkItem.notify`



🍏 A4

🍏 A5

🍏 A6

🍏 A7

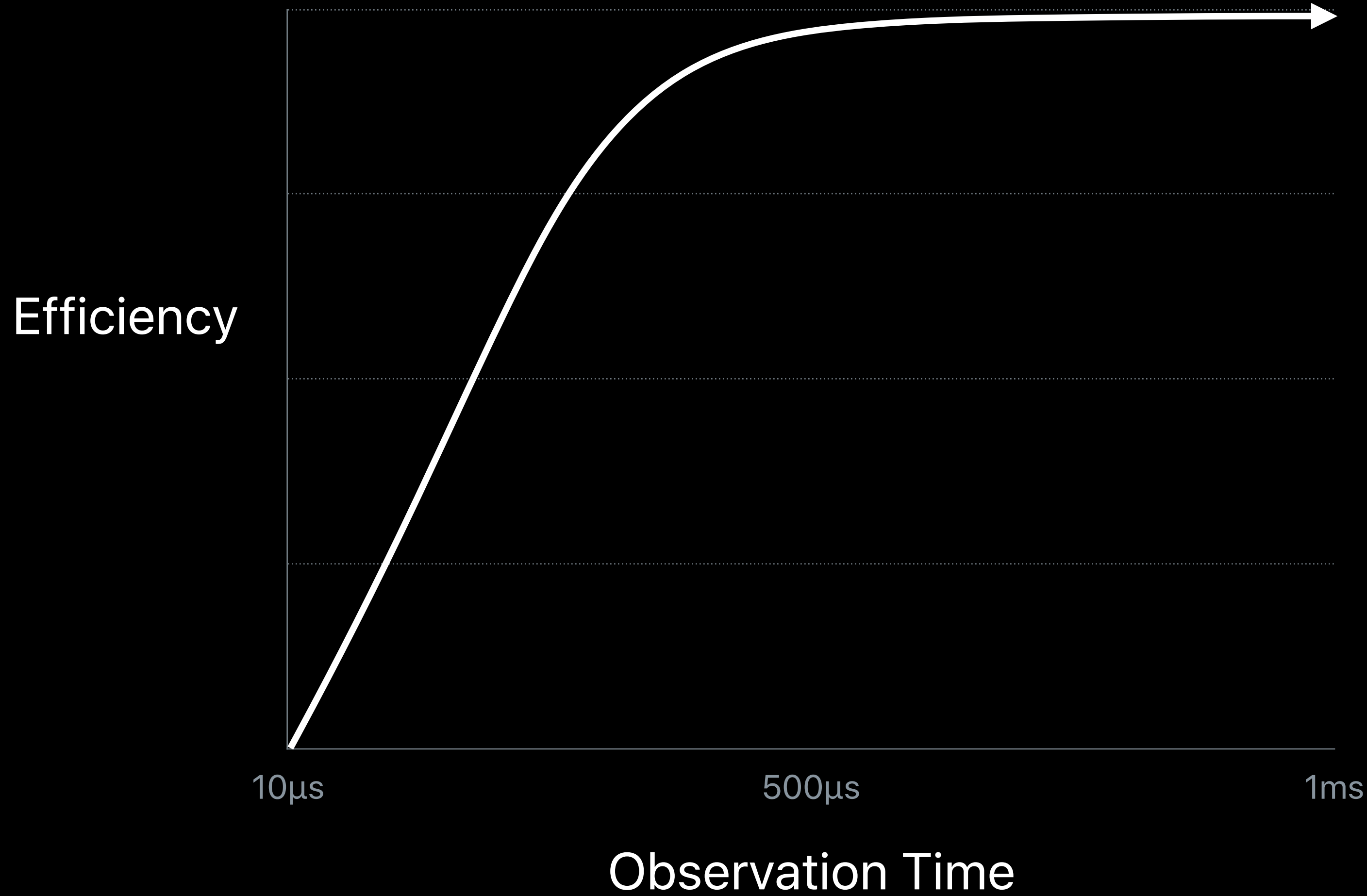
🍏 A8

🍏 A9

🍏 A10

# Efficiency Through Observation

Going off core during an operation reduces efficiency



**1.3x**

faster after combining queue hierarchies

# Parallelism and concurrency

Parallelism and concurrency  
Using GCD for concurrency



Parallelism and concurrency

Using GCD for concurrency

Unified Queue Identity

Parallelism and concurrency

Using GCD for concurrency

Unified Queue Identity

Finding problem spots

# **Parallelism**

Simultaneous execution of closely related computations

# **Concurrency**

Composition of independently executed tasks

**Parallelism**

# Parallelism

Simultaneous execution of closely related computations



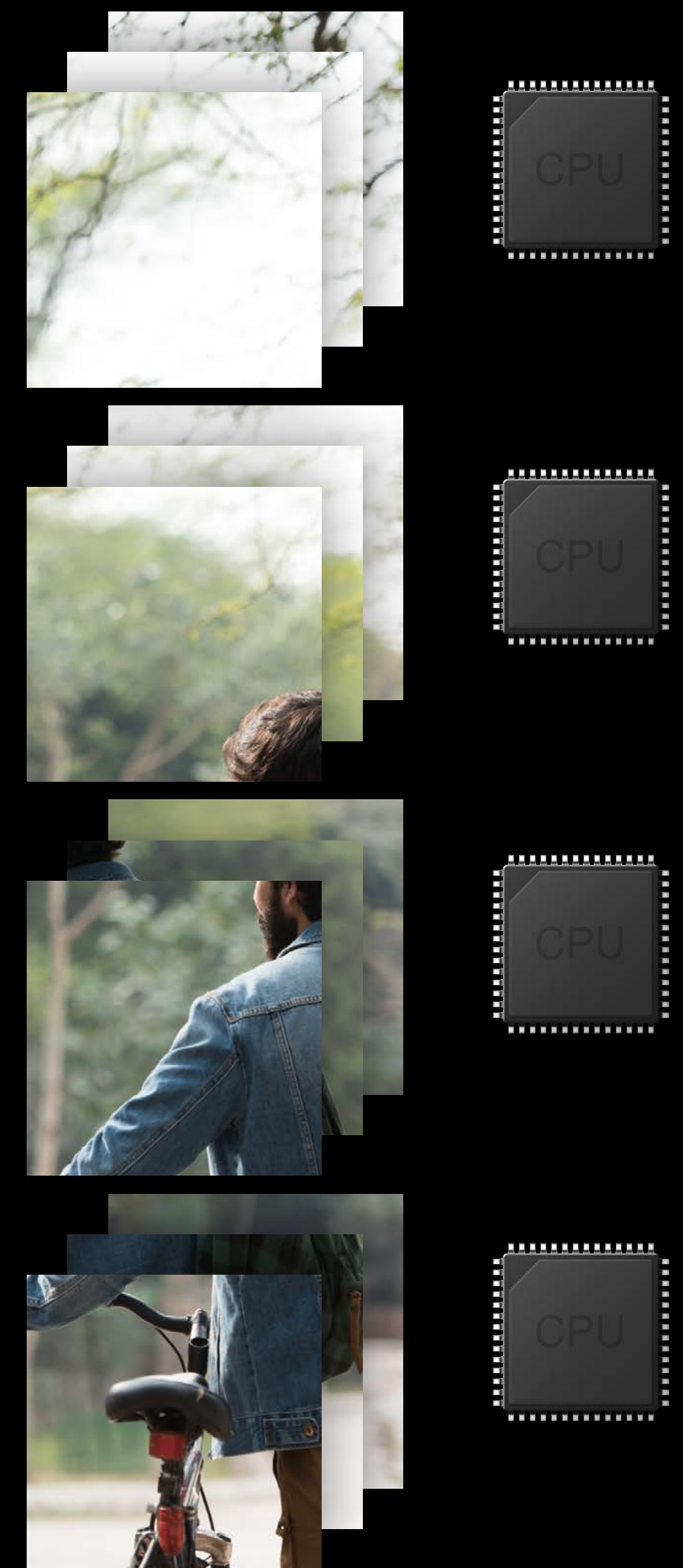
# Parallelism

Simultaneous execution of closely related computations



# Parallelism

Simultaneous execution of closely related computations



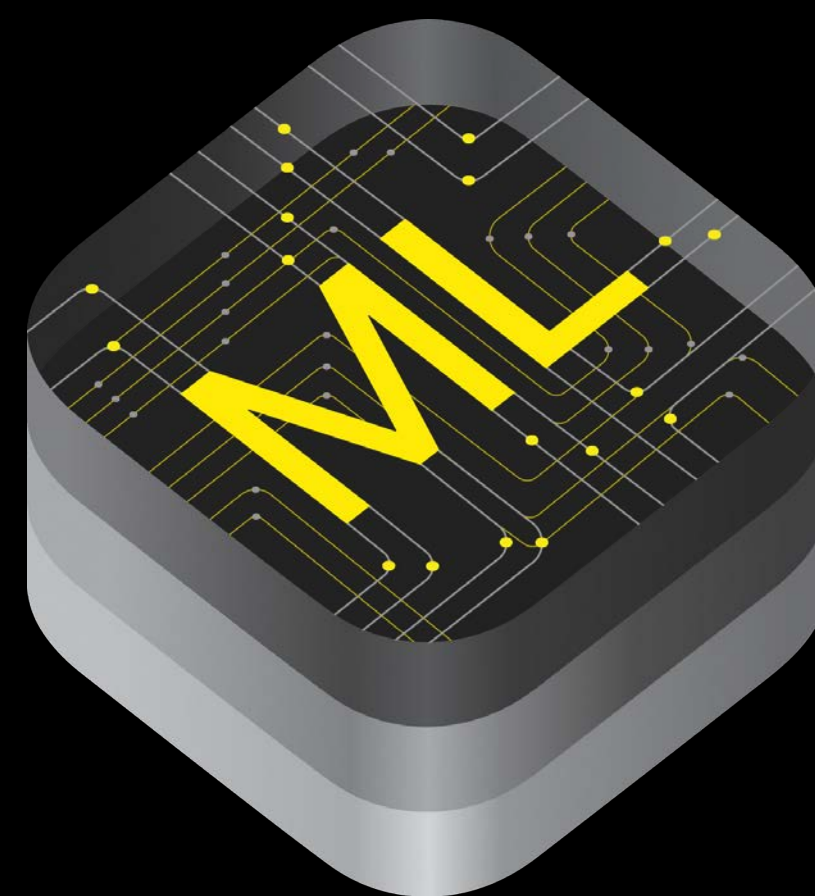
# Take Advantage of System Frameworks



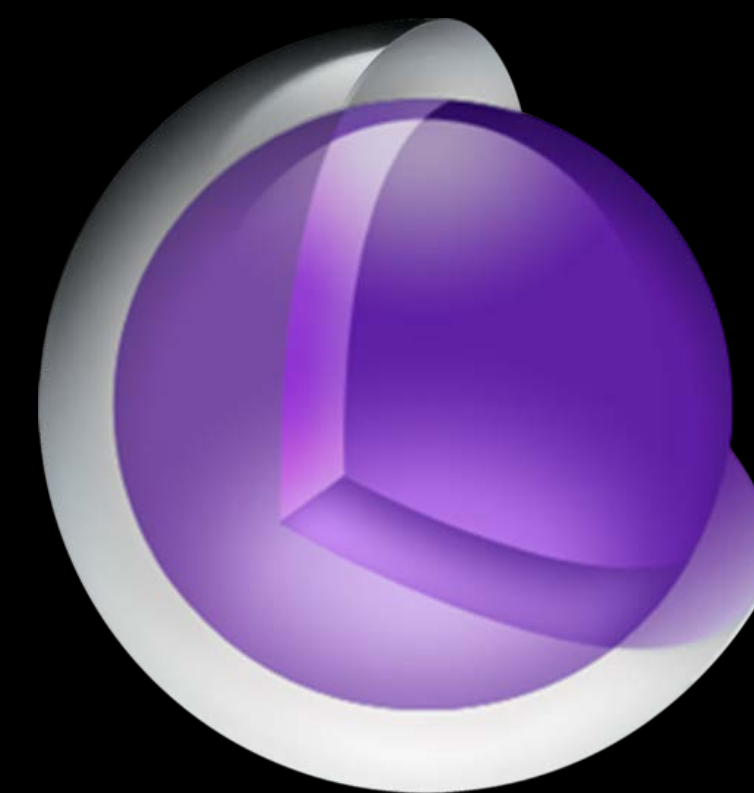
Accelerate



Metal 2



Core ML



Core Animation



# Parallelism with GCD

Express explicit parallelism with `DispatchQueue.concurrentPerform`

Parallel for-loop—calling thread participates in the computation

More efficient than many asyncs to a concurrent queue

```
DispatchQueue.concurrentPerform(1000) { i in /* iteration i */ }
```

# Parallelism with GCD

Express explicit parallelism with `DispatchQueue.concurrentPerform`

Parallel for-loop—calling thread participates in the computation

More efficient than many asyncs to a concurrent queue

```
DispatchQueue.concurrentPerform(1000) { i in /* iteration i */ }
```

# Parallelism with GCD

NEW

Express explicit parallelism with `DispatchQueue.concurrentPerform`

Parallel for-loop—calling thread participates in the computation

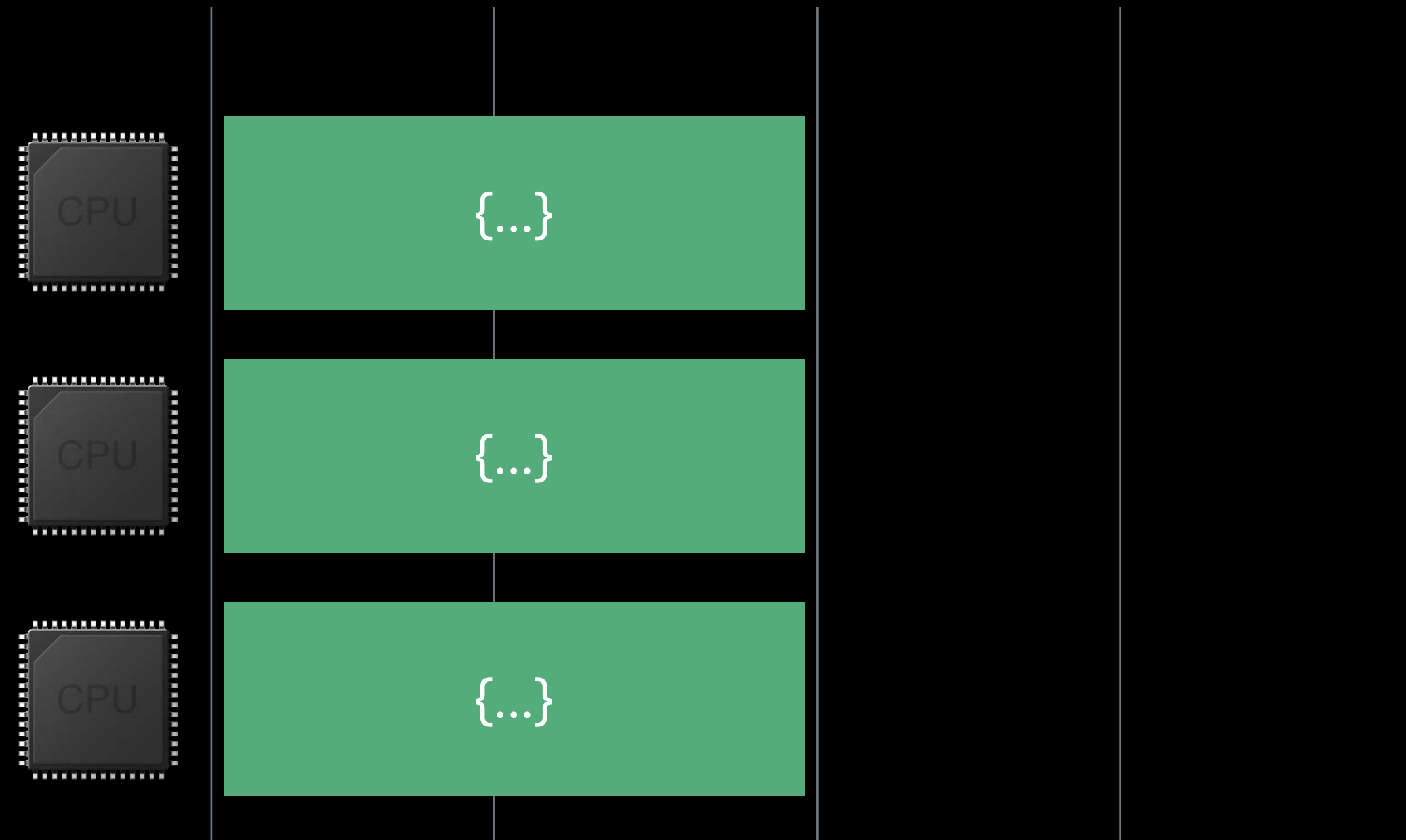
More efficient than many asyncs to a concurrent queue

```
DispatchQueue.concurrentPerform(1000) { i in /* iteration i */ }  
  
dispatch_apply(DISPATCH_APPLY_AUTO, 1000, ^(size_t i){ /* iteration i */ })
```

`DISPATCH_APPLY_AUTO` deploys back to macOS 10.9, iOS 7.0

# Dynamic Resource Availability

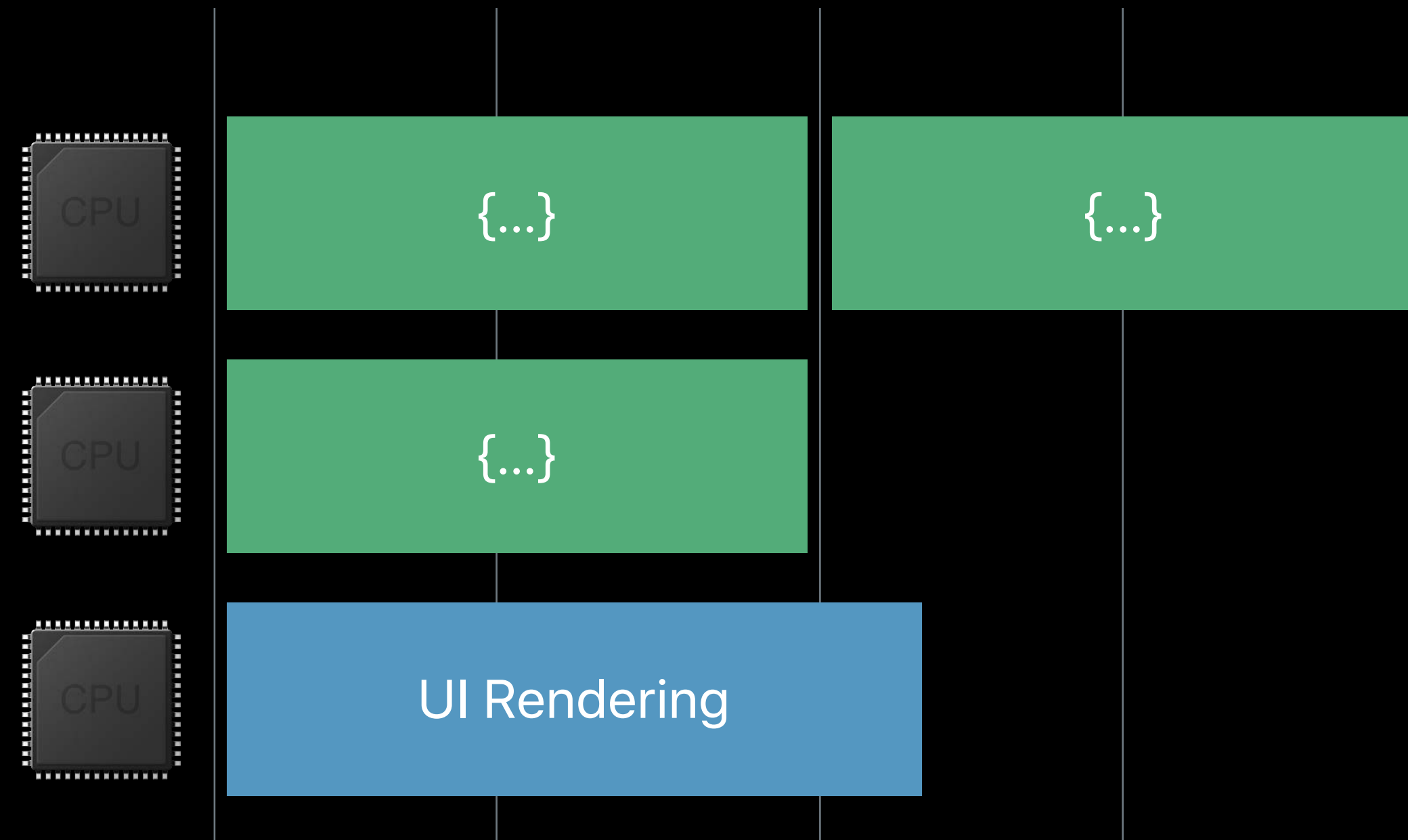
Choosing an iteration count



```
DispatchQueue.concurrentPerform(3) { i in /* iteration i */ }
```

# Dynamic Resource Availability

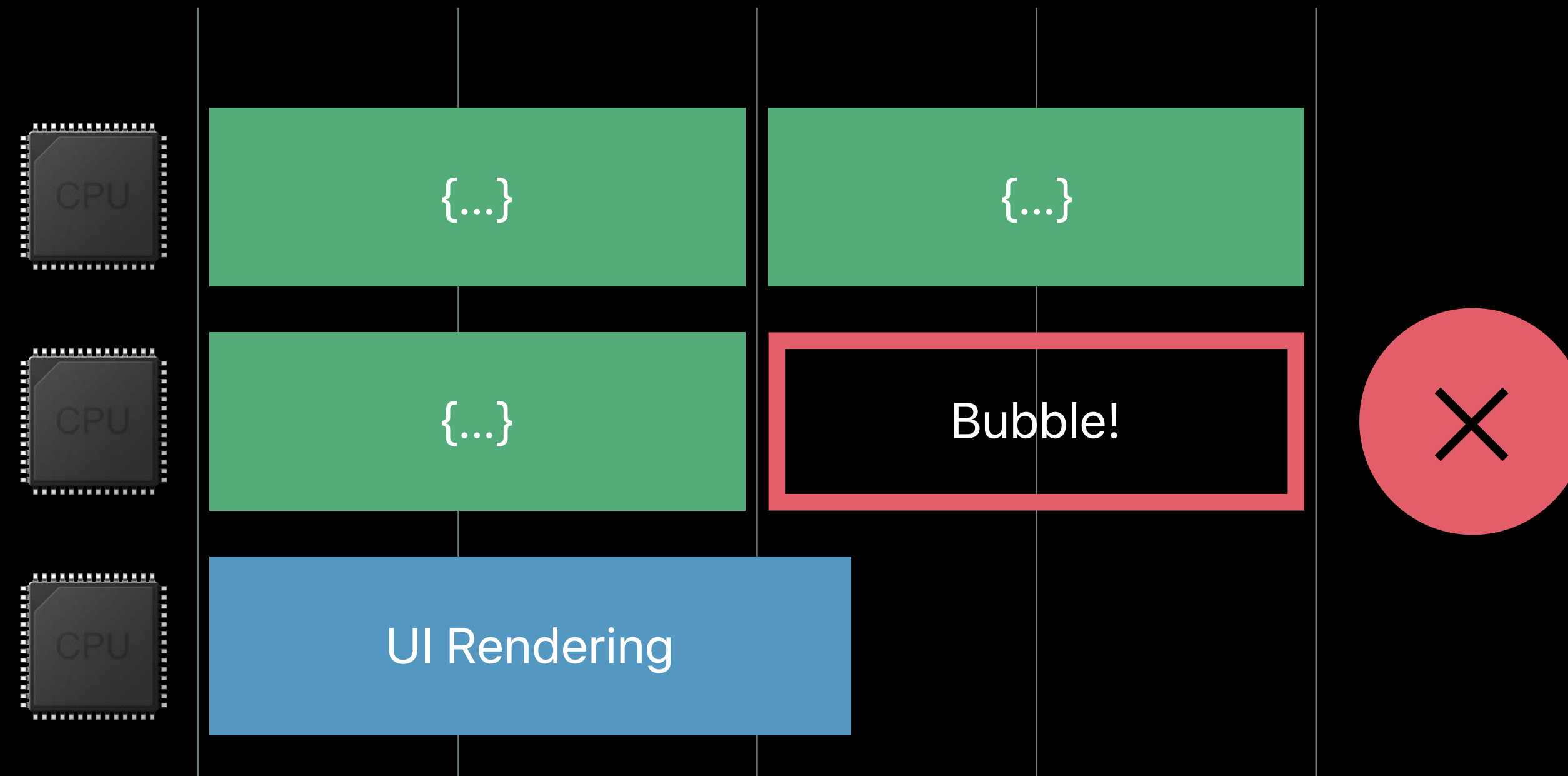
Choosing an iteration count



```
DispatchQueue.concurrentPerform(3) { i in /* iteration i */ }
```

# Dynamic Resource Availability

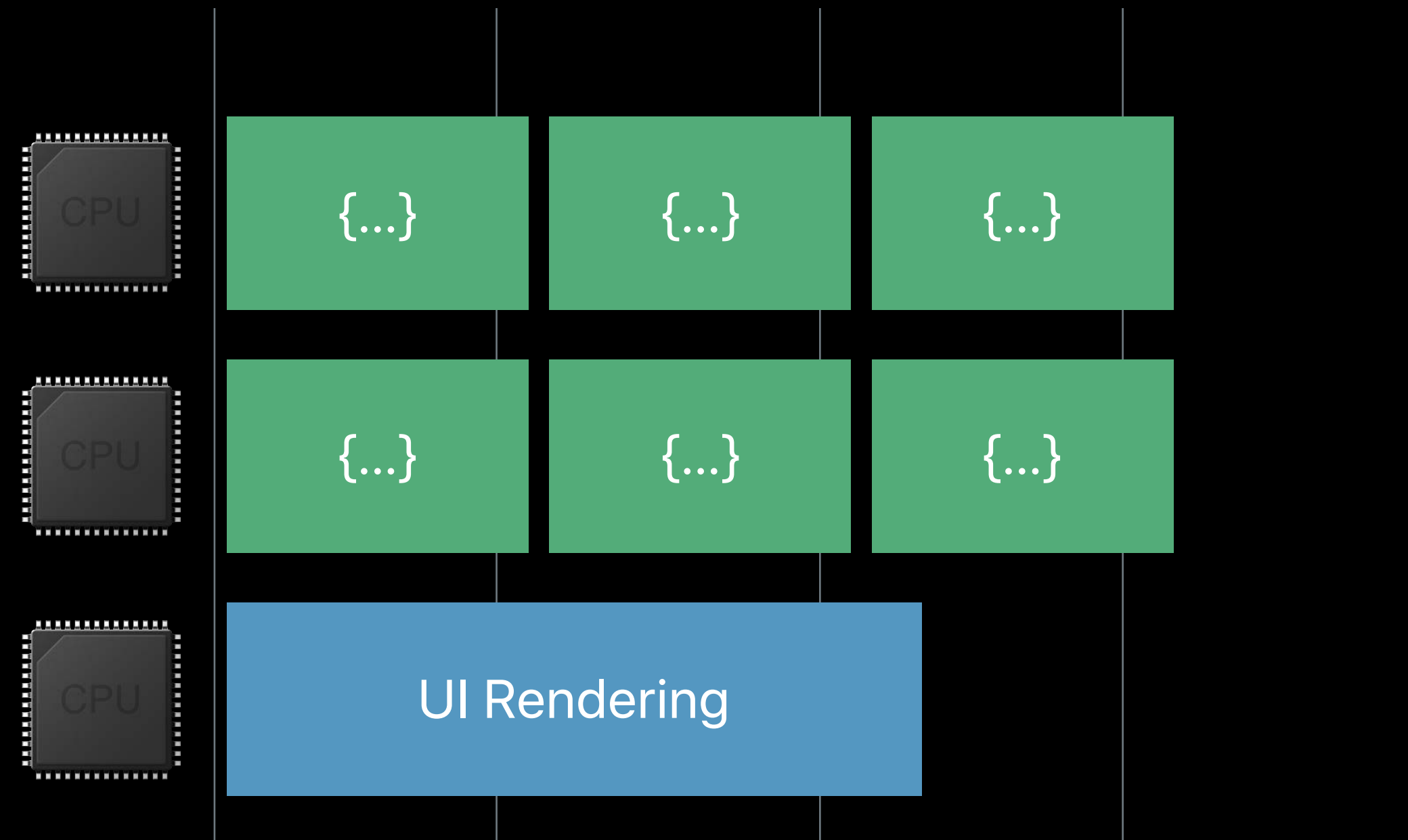
Choosing an iteration count



```
DispatchQueue.concurrentPerform(3) { i in /* iteration i */ }
```

# Dynamic Resource Availability

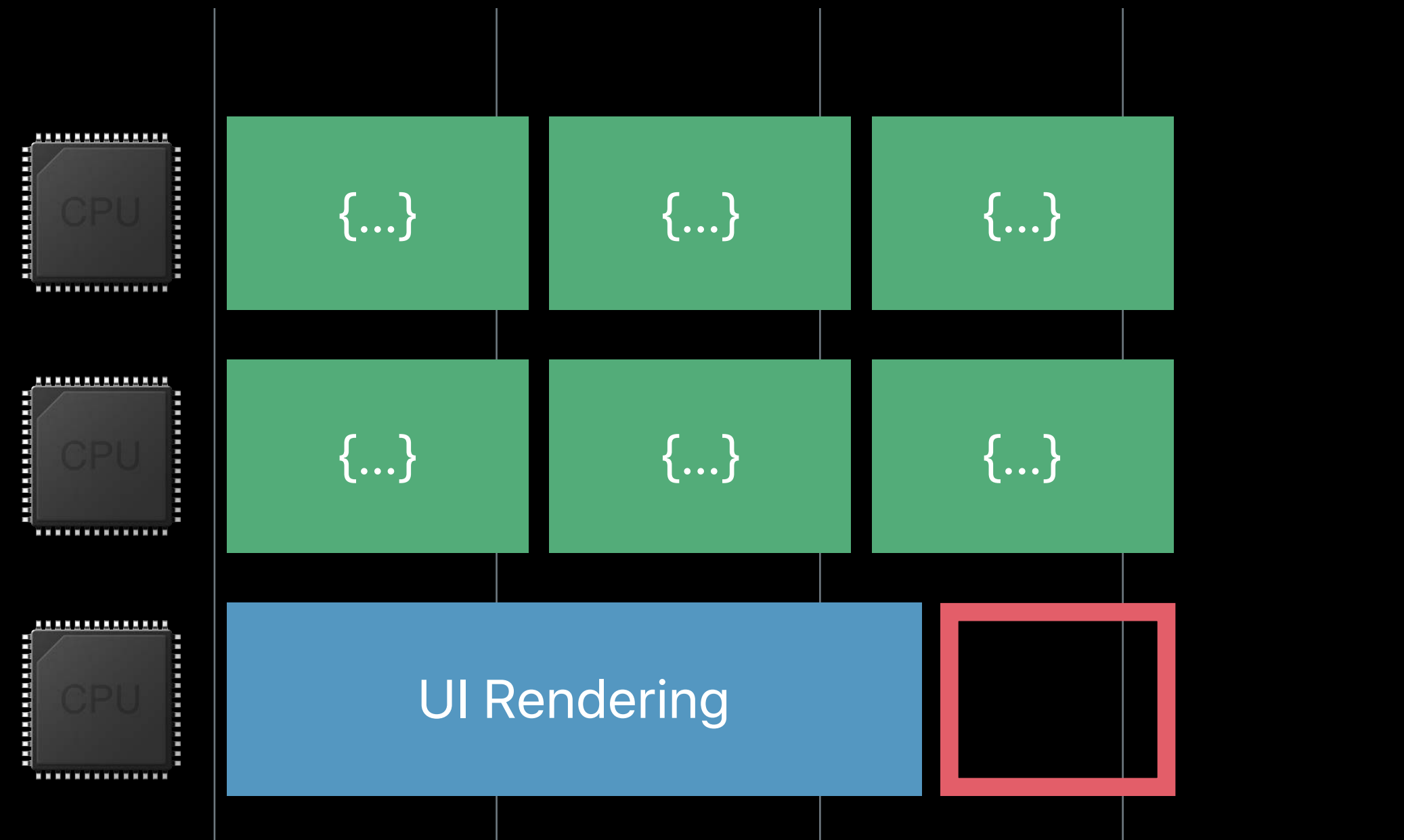
Choosing an iteration count



```
DispatchQueue.concurrentPerform(6) { i in /* iteration i */ }
```

# Dynamic Resource Availability

Choosing an iteration count

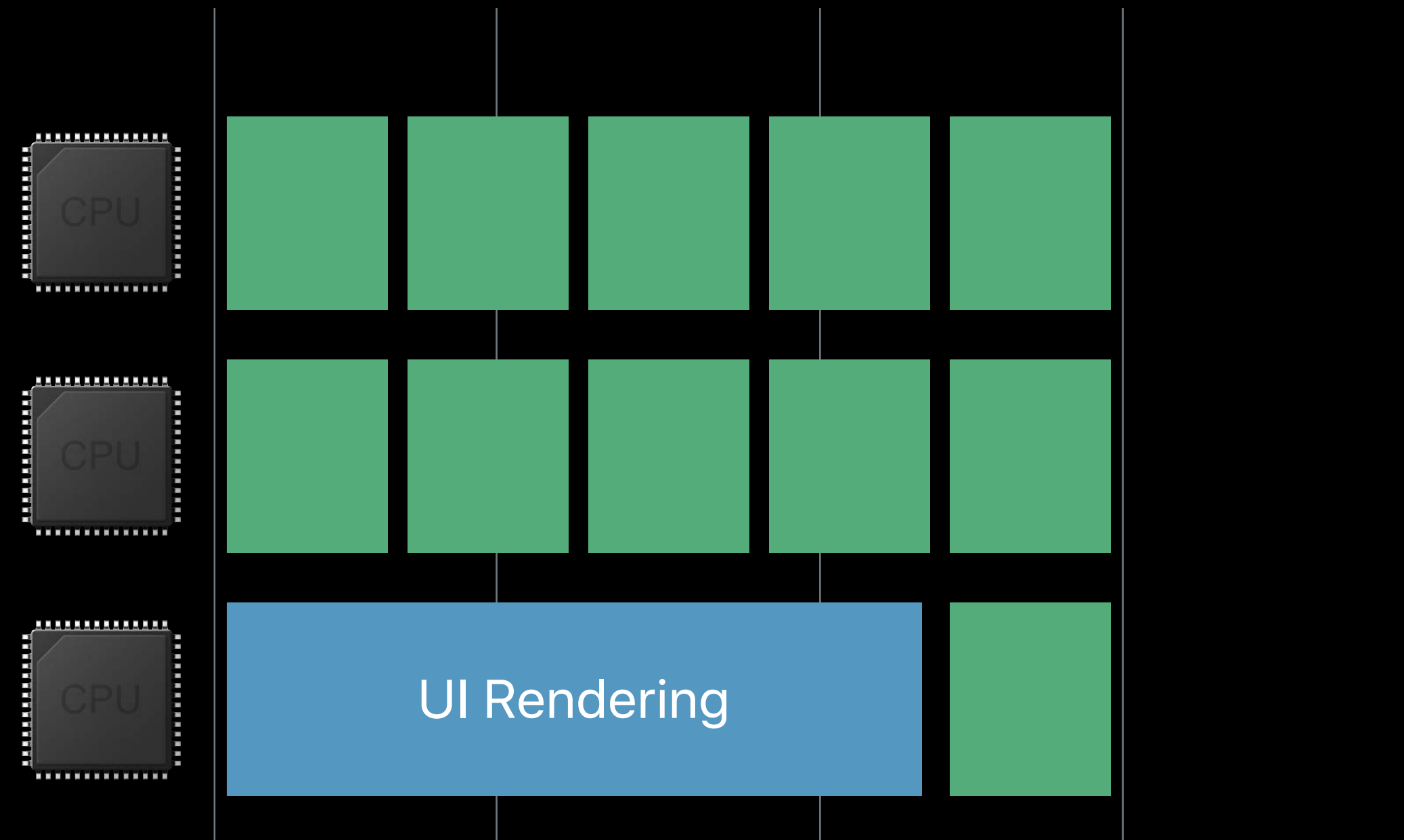


```
DispatchQueue.concurrentPerform(6) { i in /* iteration i */ }
```



# Dynamic Resource Availability

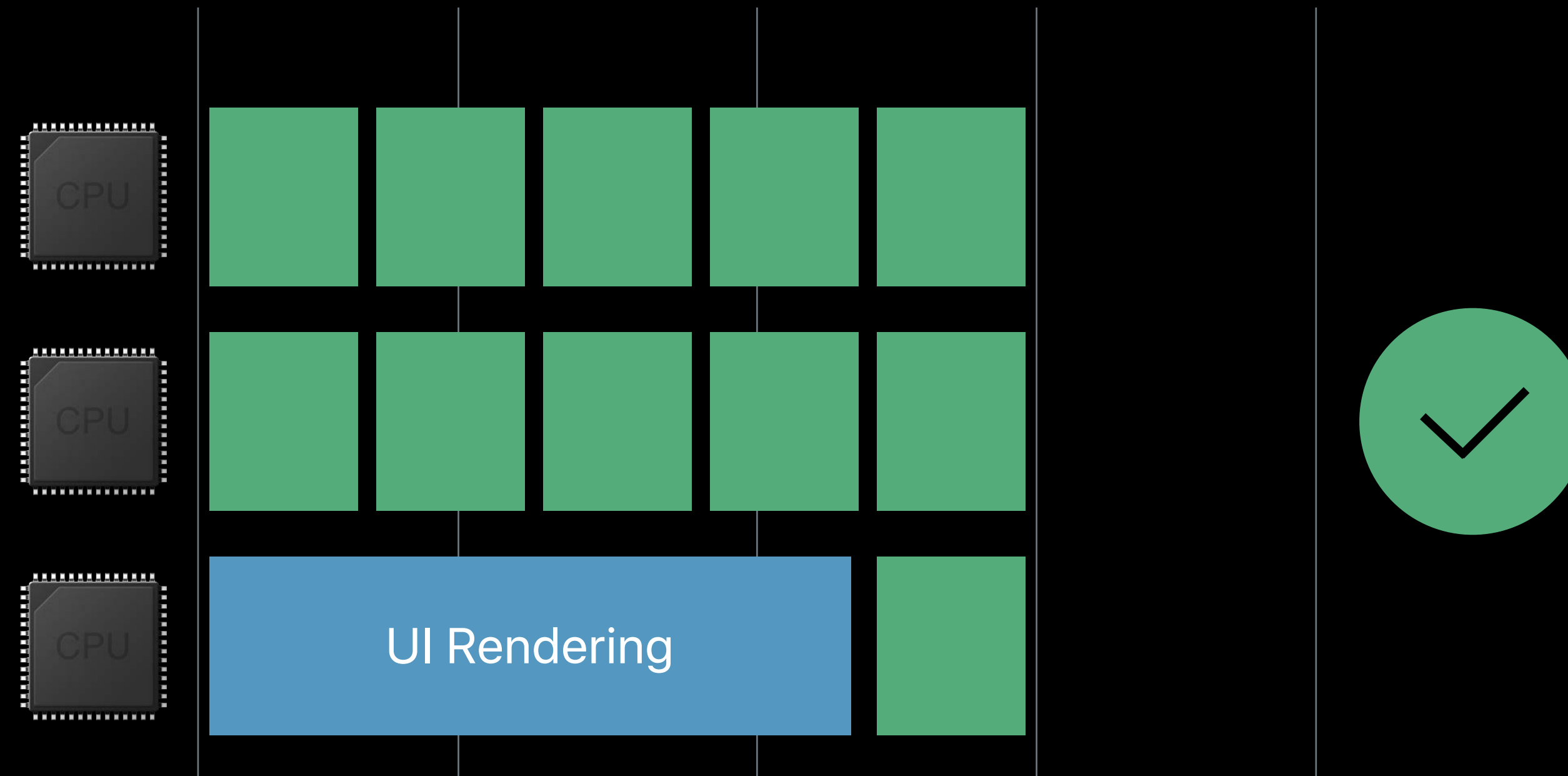
Choosing an iteration count



```
DispatchQueue.concurrentPerform(11) { i in /* iteration i */ }
```

# Dynamic Resource Availability

Choosing an iteration count



```
DispatchQueue.concurrentPerform(11) { i in /* iteration i */ }
```

# Dynamic Resource Availability

Choosing an iteration count



```
DispatchQueue.concurrentPerform(1000) { i in /* iteration i */ }
```

# Parallelism

Leverage system frameworks

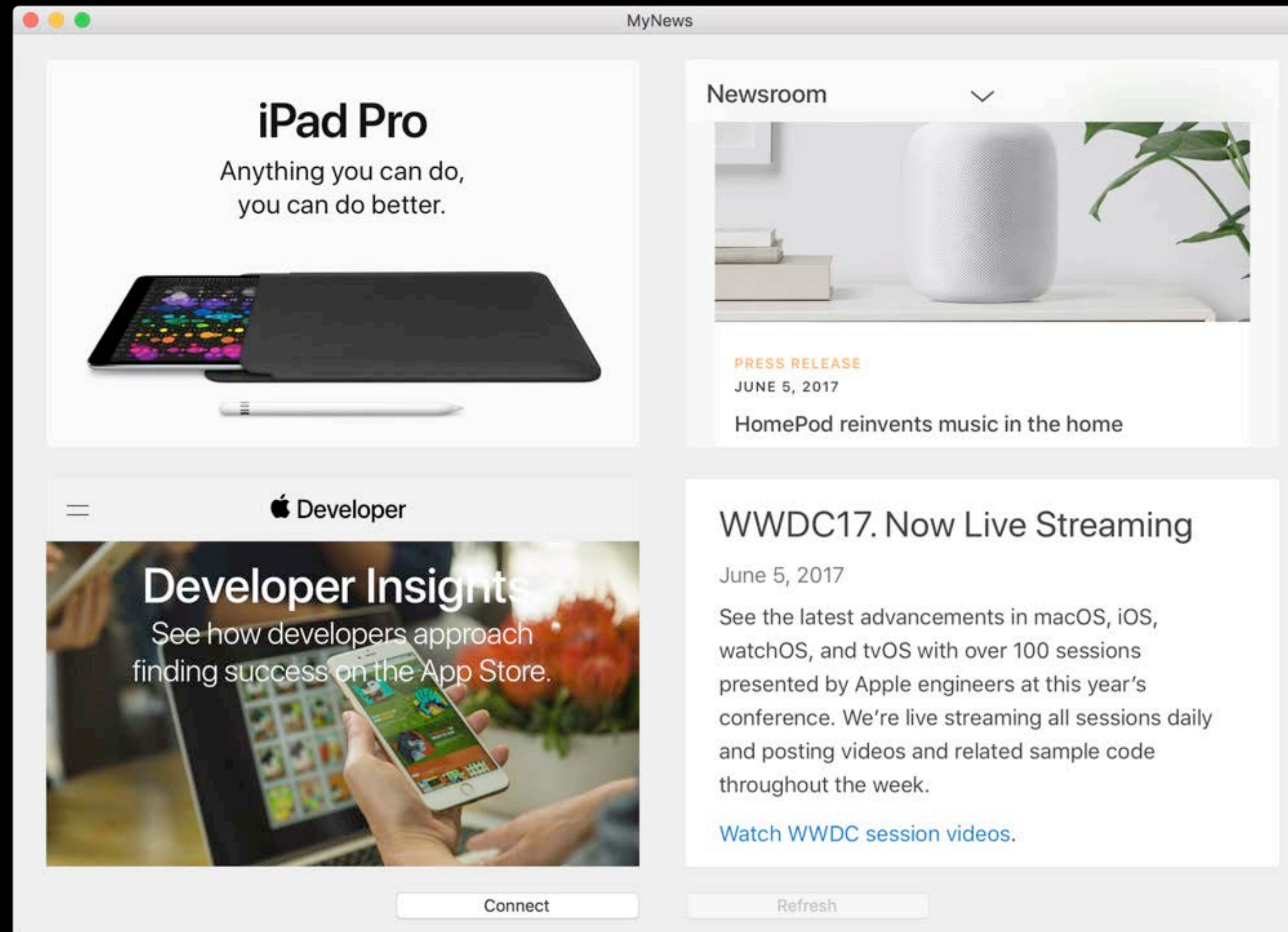
Use `DispatchQueue.concurrentPerform`

Consider dynamic availability

# Concurrency

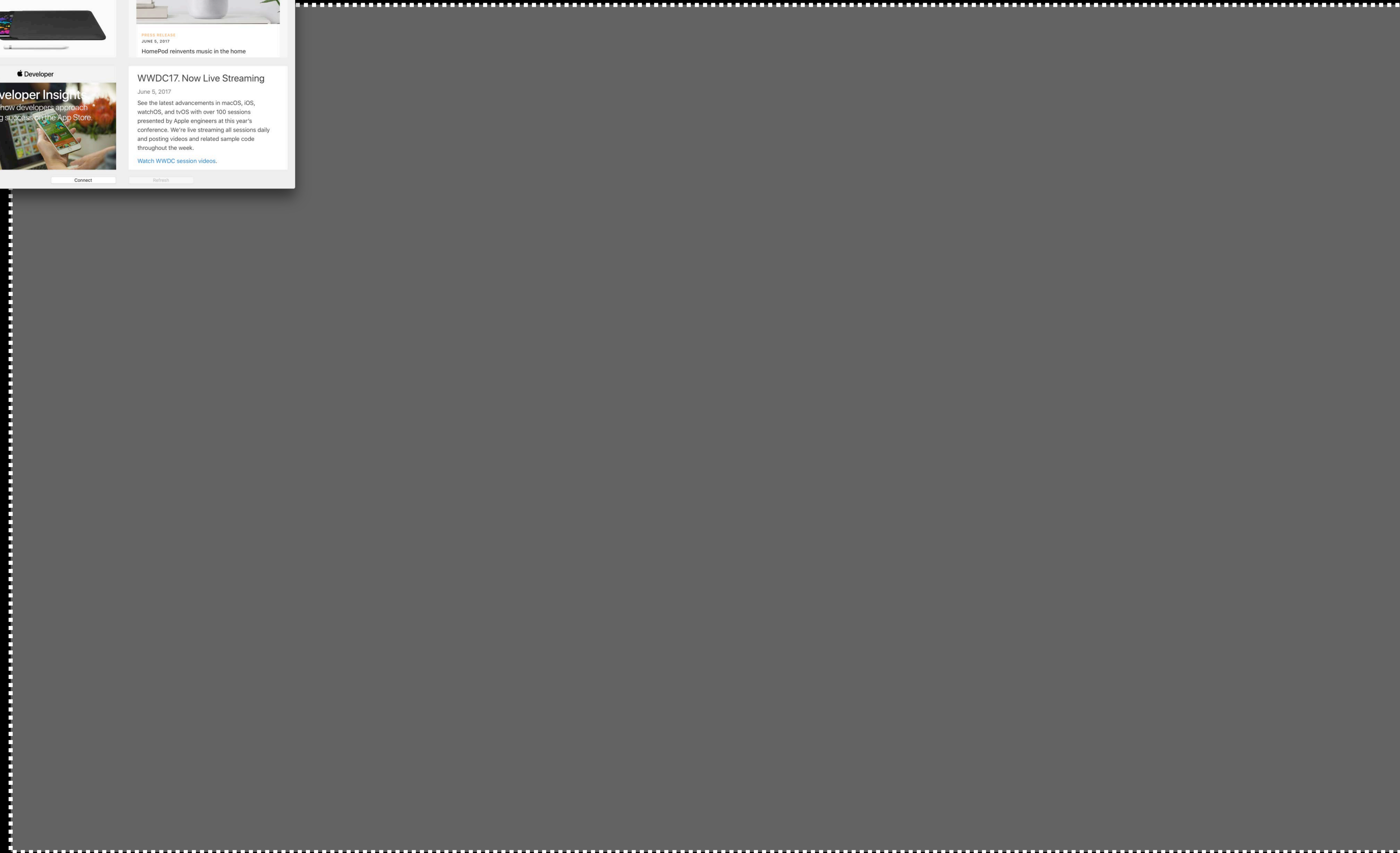
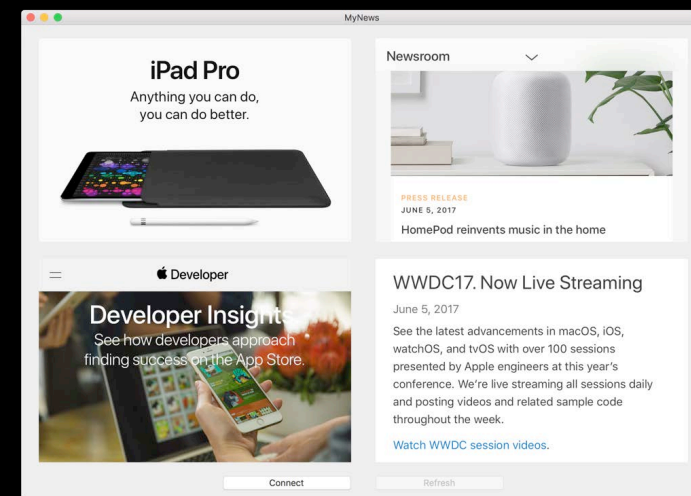
# Concurrency

Composition of independently executed tasks



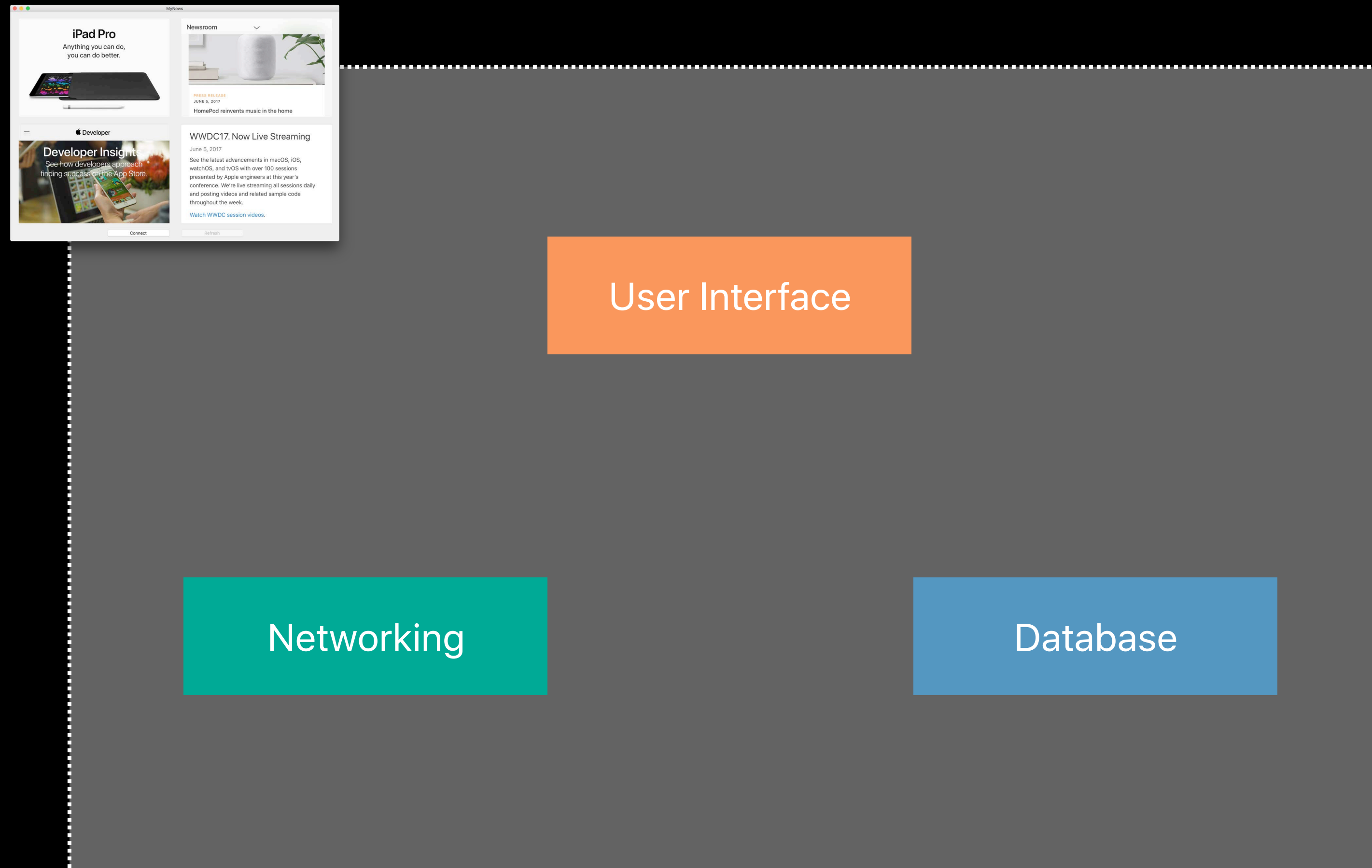
# Concurrency

Composition of independently executed tasks



# Concurrency

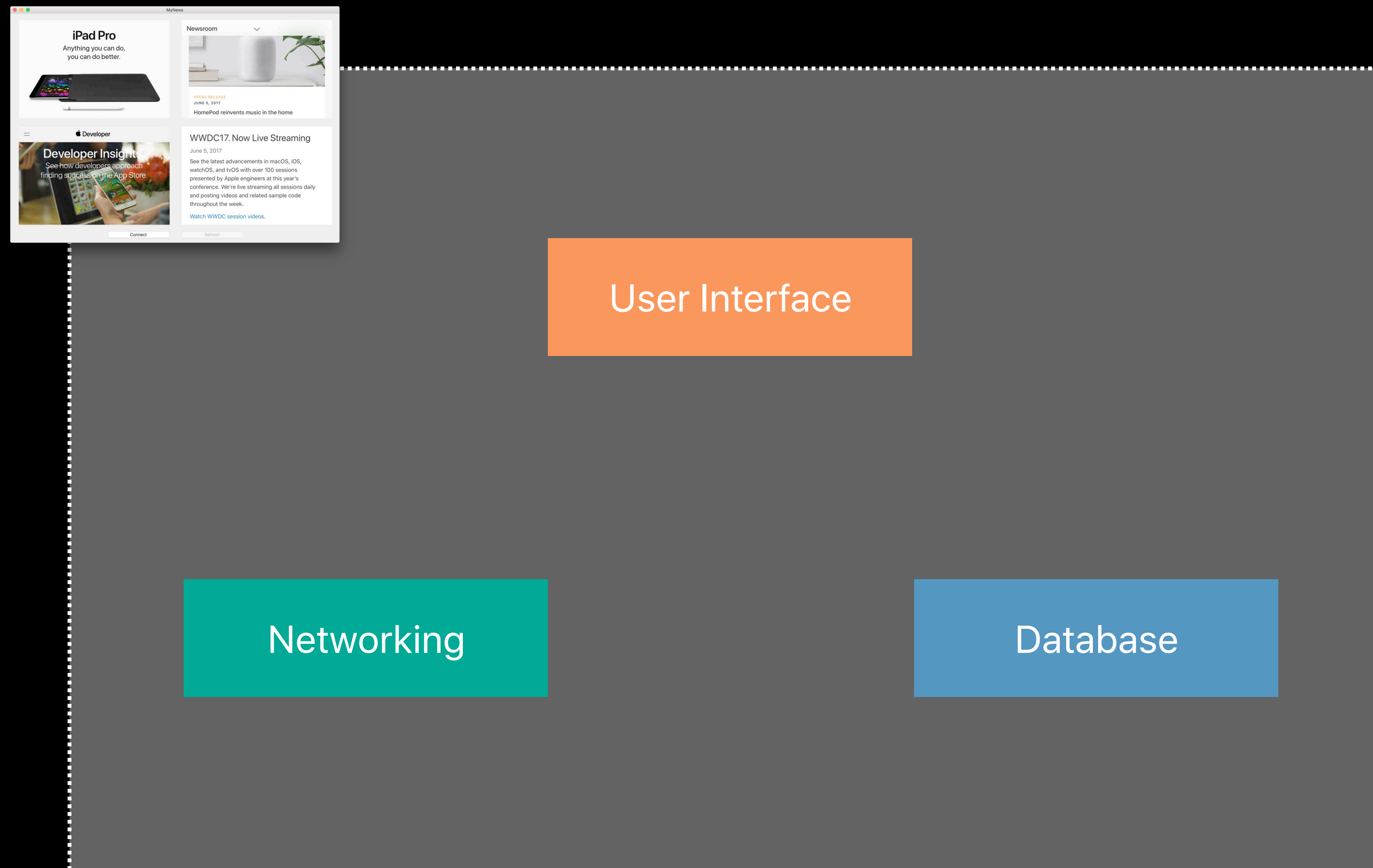
Composition of independently executed tasks





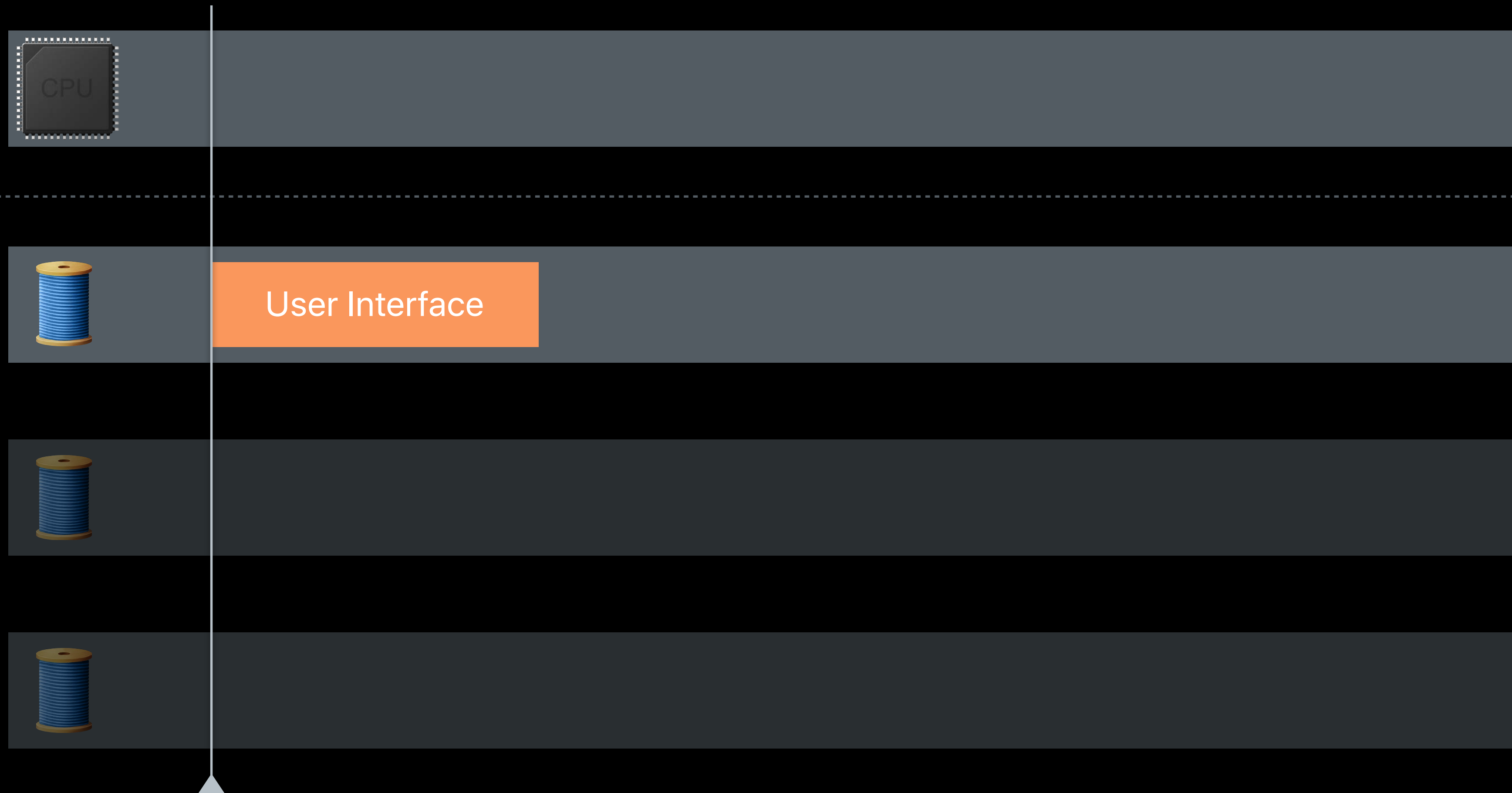
# Concurrency

Composition of independently executed tasks



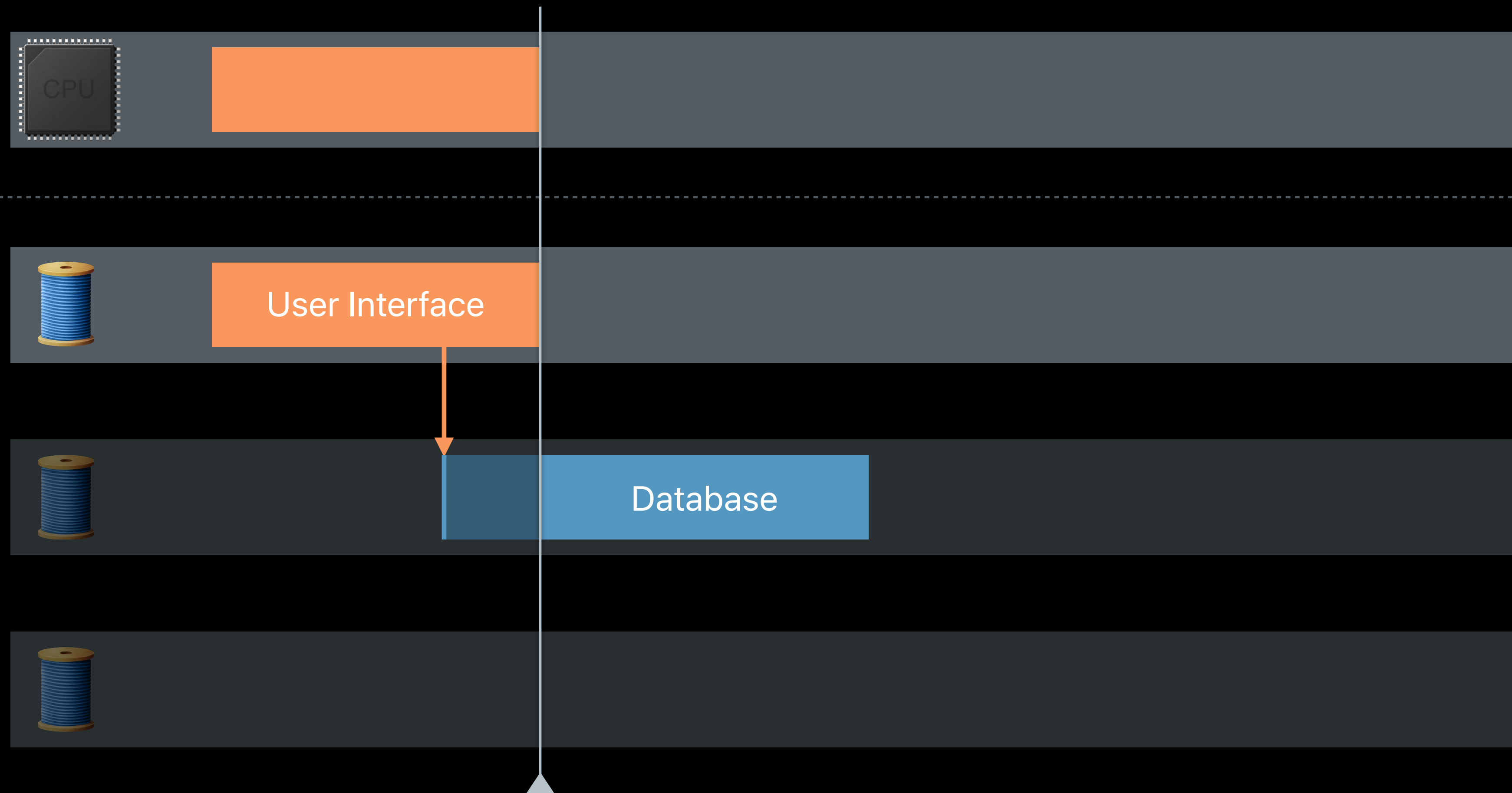
# Concurrency

Composition of independently executed tasks



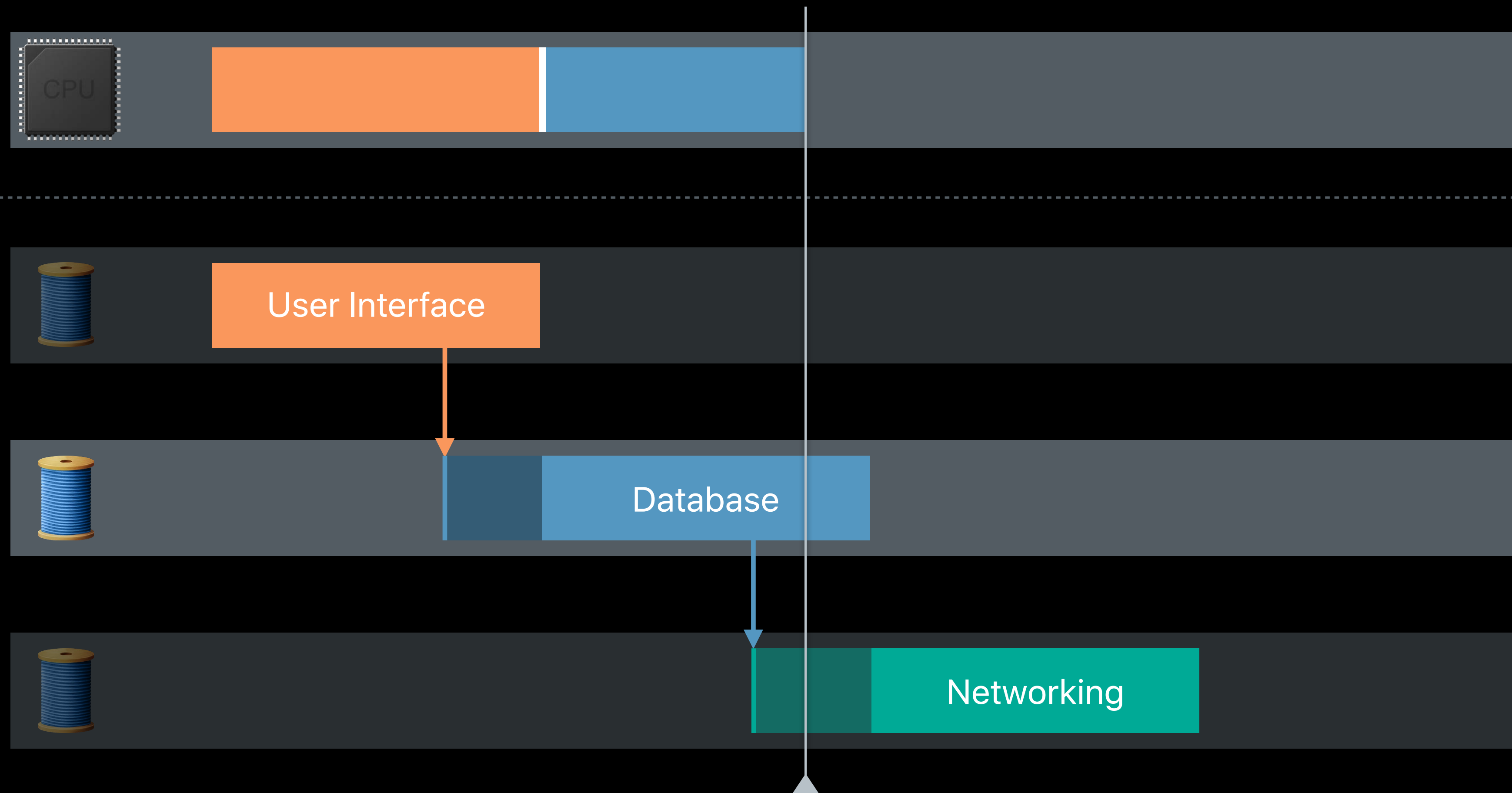
# Concurrency

Composition of independently executed tasks



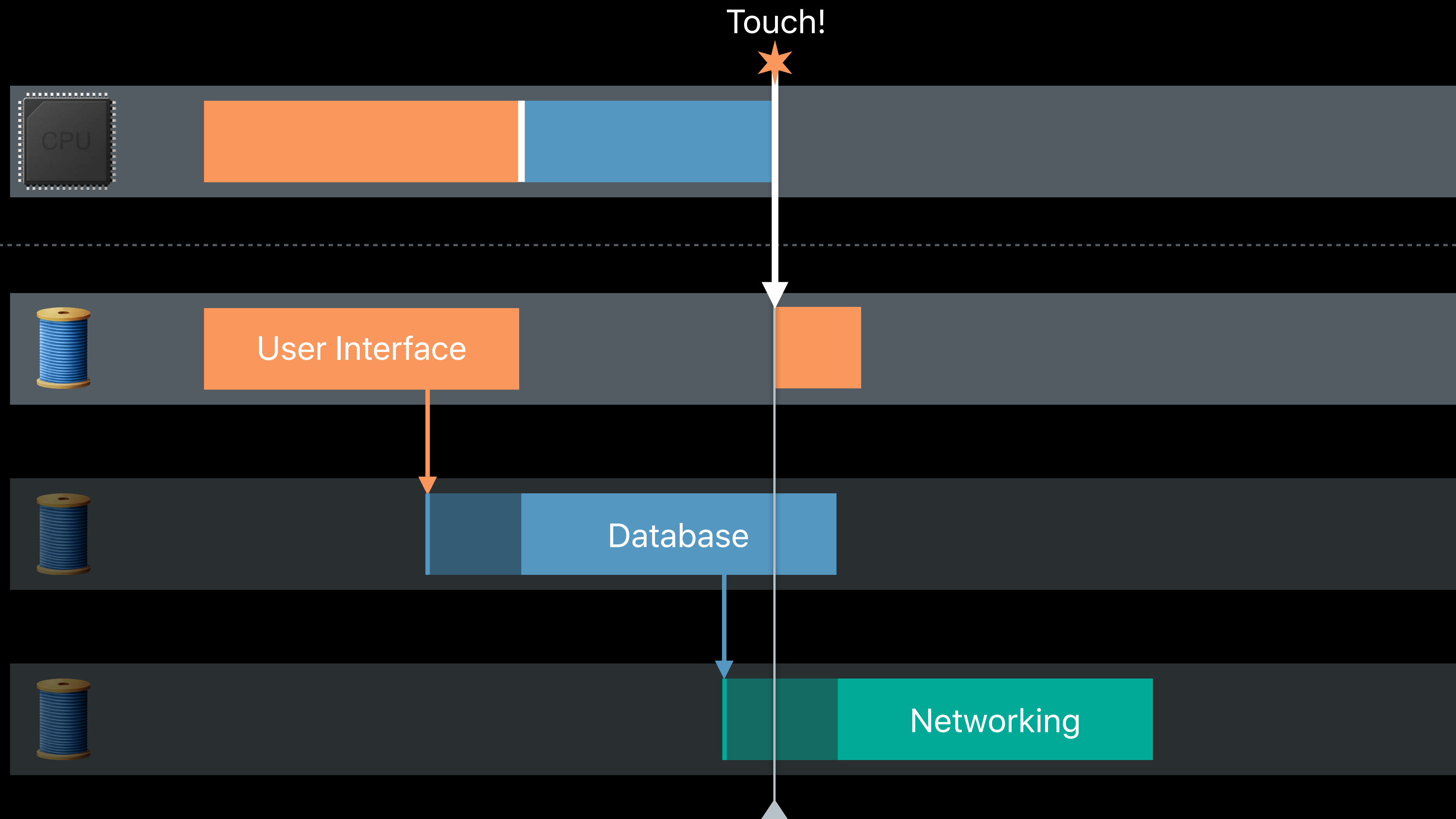
# Concurrency

Composition of independently executed tasks



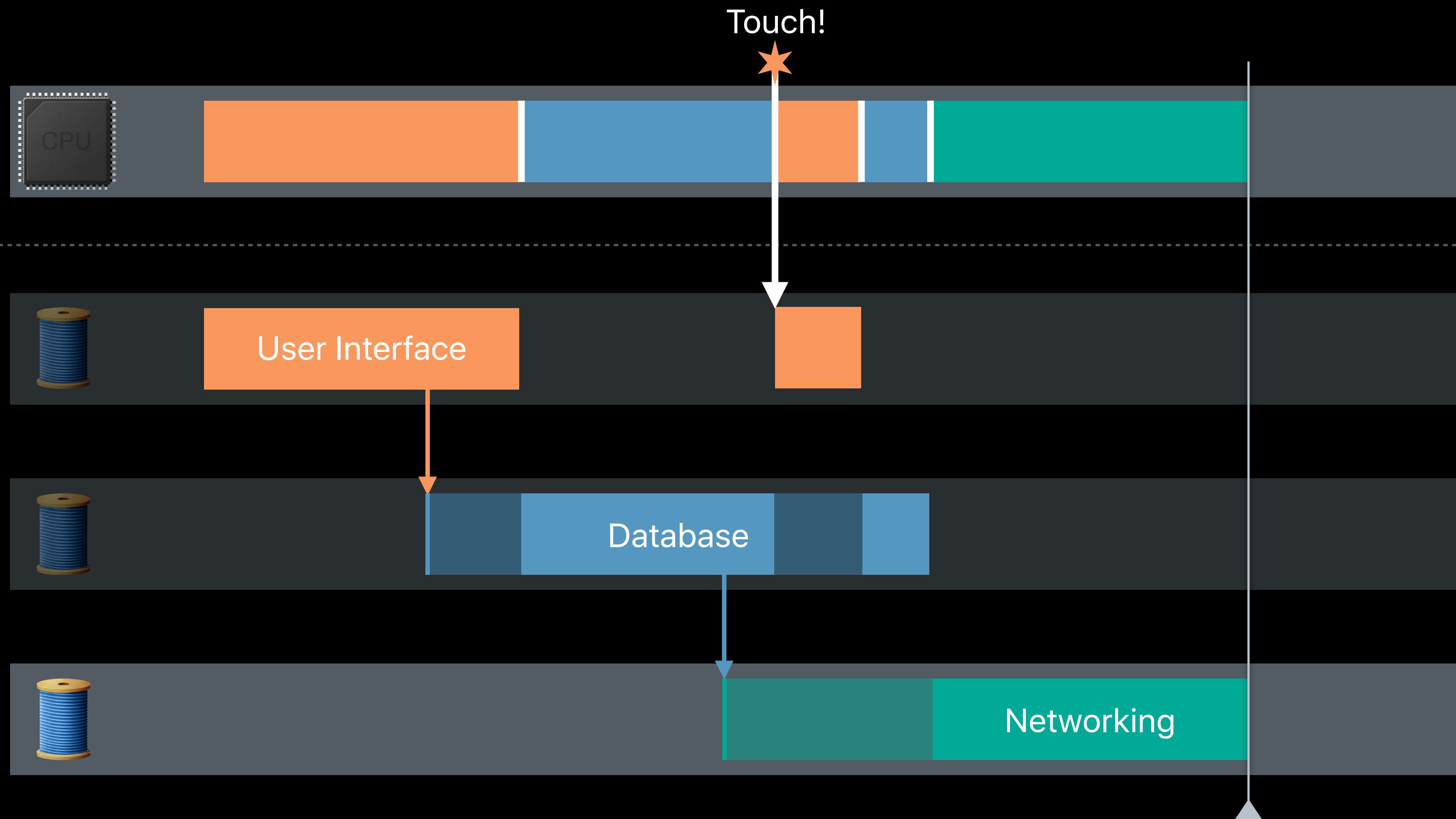
# Concurrency

Composition of independently executed tasks



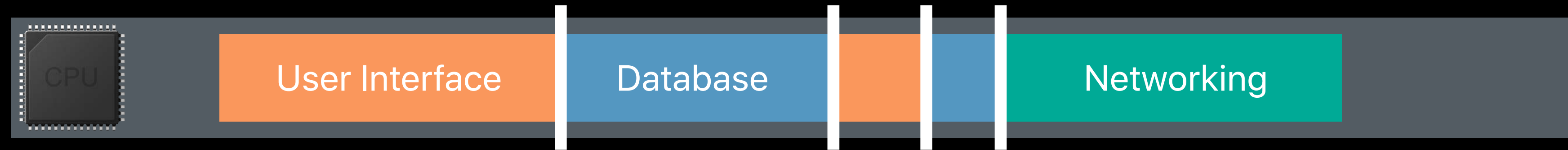
# Concurrency

Composition of independently executed tasks



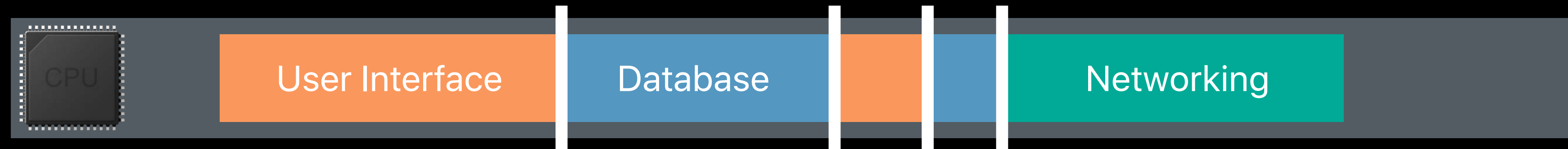
# Concurrency

## Context switching



# Concurrency

## Context switching

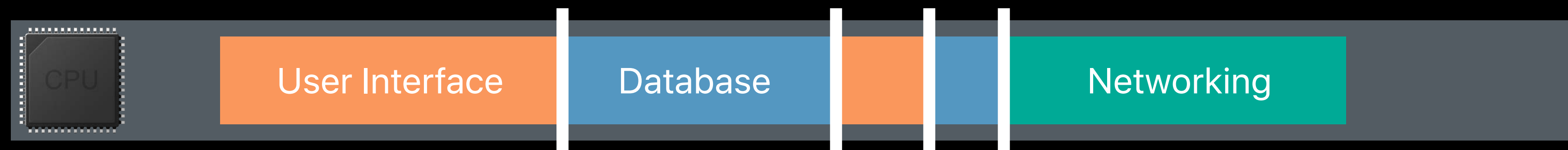




# Context Switching

The power of concurrency

The OS can choose a new thread at any time

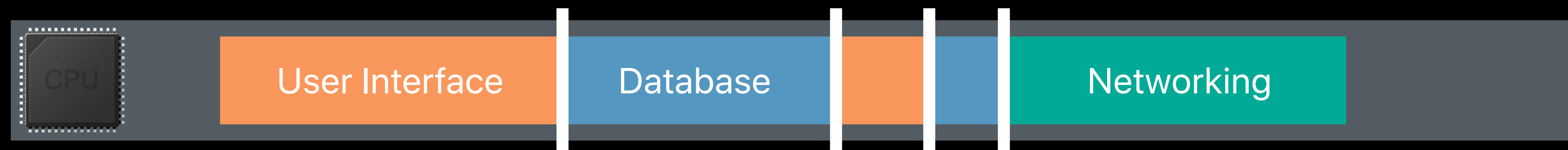


# Context Switching

The power of concurrency

The OS can choose a new thread at any time

- A higher priority thread needs the CPU

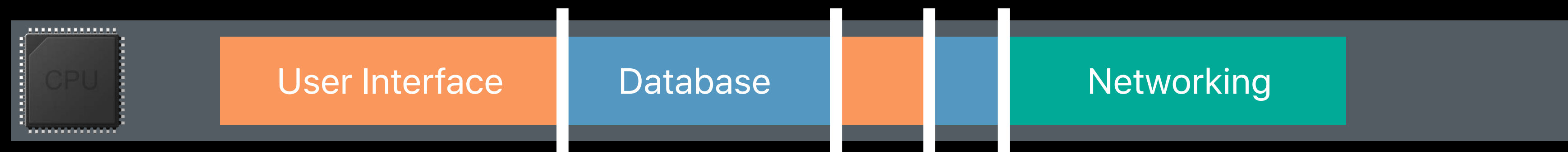


# Context Switching

The power of concurrency

The OS can choose a new thread at any time

- A higher priority thread needs the CPU
- A thread finishes its current work

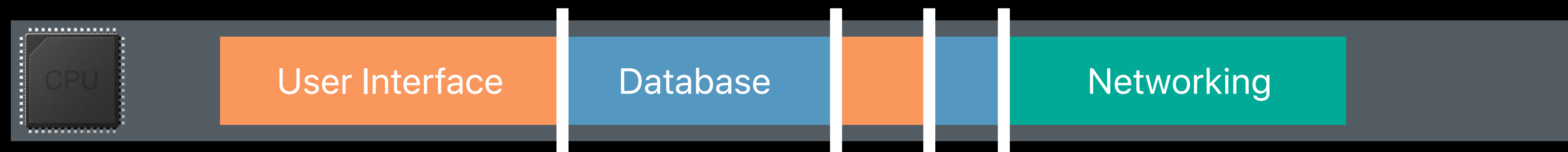


# Context Switching

The power of concurrency

The OS can choose a new thread at any time

- A higher priority thread needs the CPU
- A thread finishes its current work
- Waiting to acquire a resource

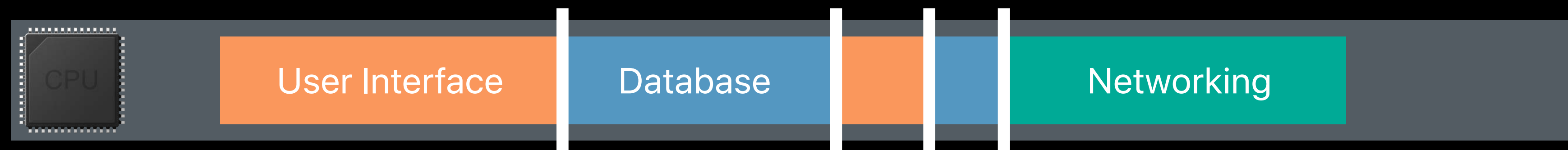


# Context Switching

The power of concurrency

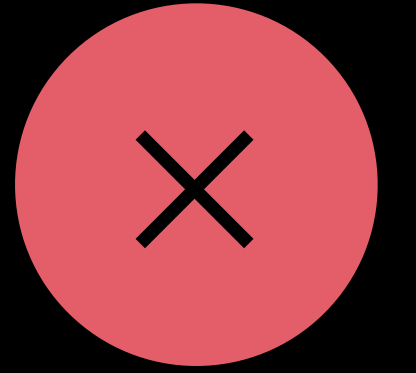
The OS can choose a new thread at any time

- A higher priority thread needs the CPU
- A thread finishes its current work
- Waiting to acquire a resource
- Waiting for an asynchronous request to complete



# Excessive Context Switching

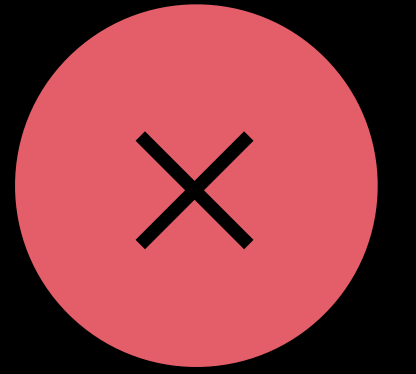
Too much of a good thing



Repeatedly bouncing between contexts can become expensive

# Excessive Context Switching

Too much of a good thing

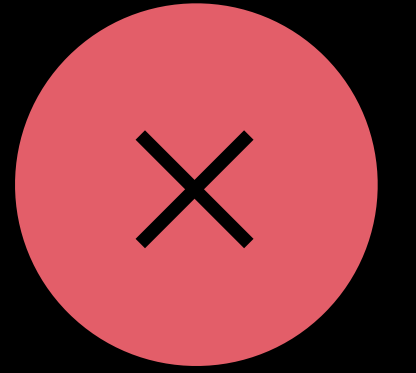


Repeatedly bouncing between contexts can become expensive



# Excessive Context Switching

Too much of a good thing



Repeatedly bouncing between contexts can become expensive

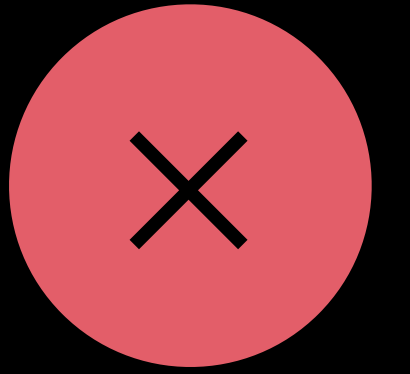
- CPU runs less efficiently





# Excessive Context Switching

Too much of a good thing



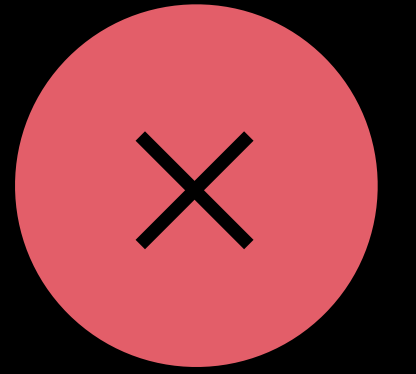
Repeatedly bouncing between contexts can become expensive

- CPU runs less efficiently



# Excessive Context Switching

Too much of a good thing



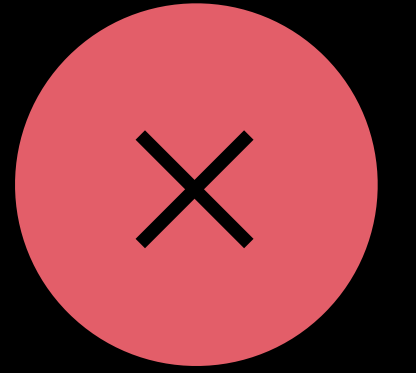
Repeatedly bouncing between contexts can become expensive

- CPU runs less efficiently
- There may be others ahead in line for CPU access



# Excessive Context Switching

Too much of a good thing



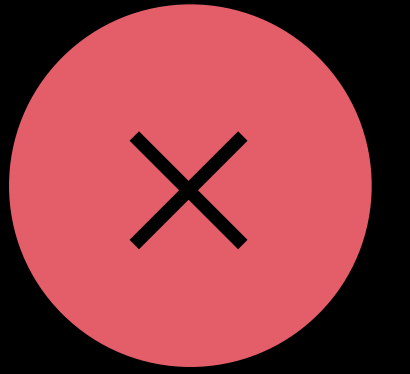
Repeatedly bouncing between contexts can become expensive

- CPU runs less efficiently
- There may be others ahead in line for CPU access



# Excessive Context Switching

Too much of a good thing



Repeatedly bouncing between contexts can become expensive

- CPU runs less efficiently
- There may be others ahead in line for CPU access



# Excessive Context Switching

Too much of a good thing

Repeatedly waiting for exclusive access to contended resources

Repeatedly switching between independent operations

Repeatedly bouncing an operation between threads

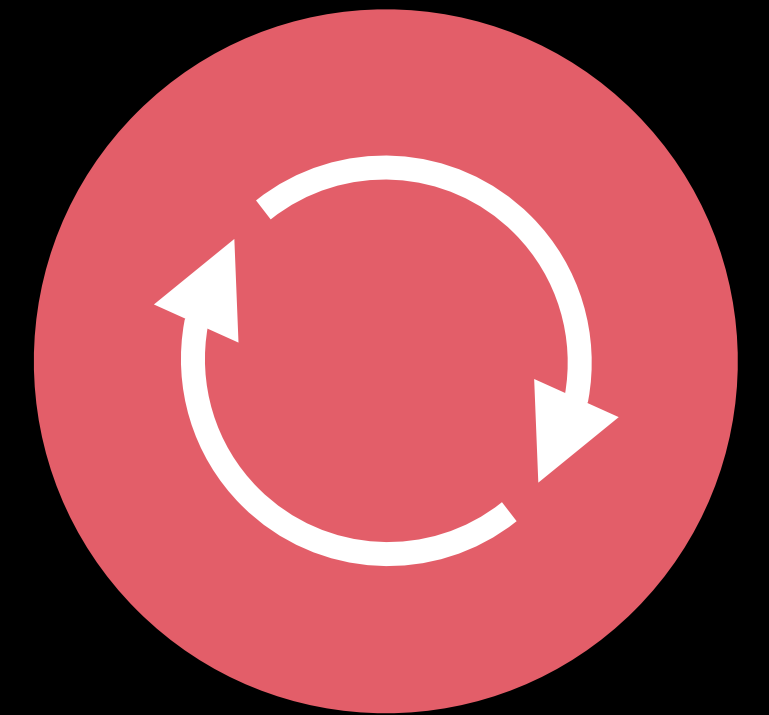
# Excessive Context Switching

Too much of a good thing

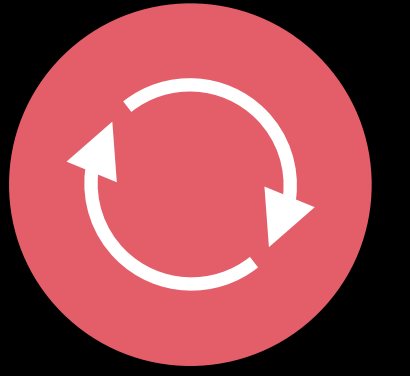
Repeatedly waiting for exclusive access to contended resources

Repeatedly switching between independent operations

Repeatedly bouncing an operation between threads



# Too much of a good thing

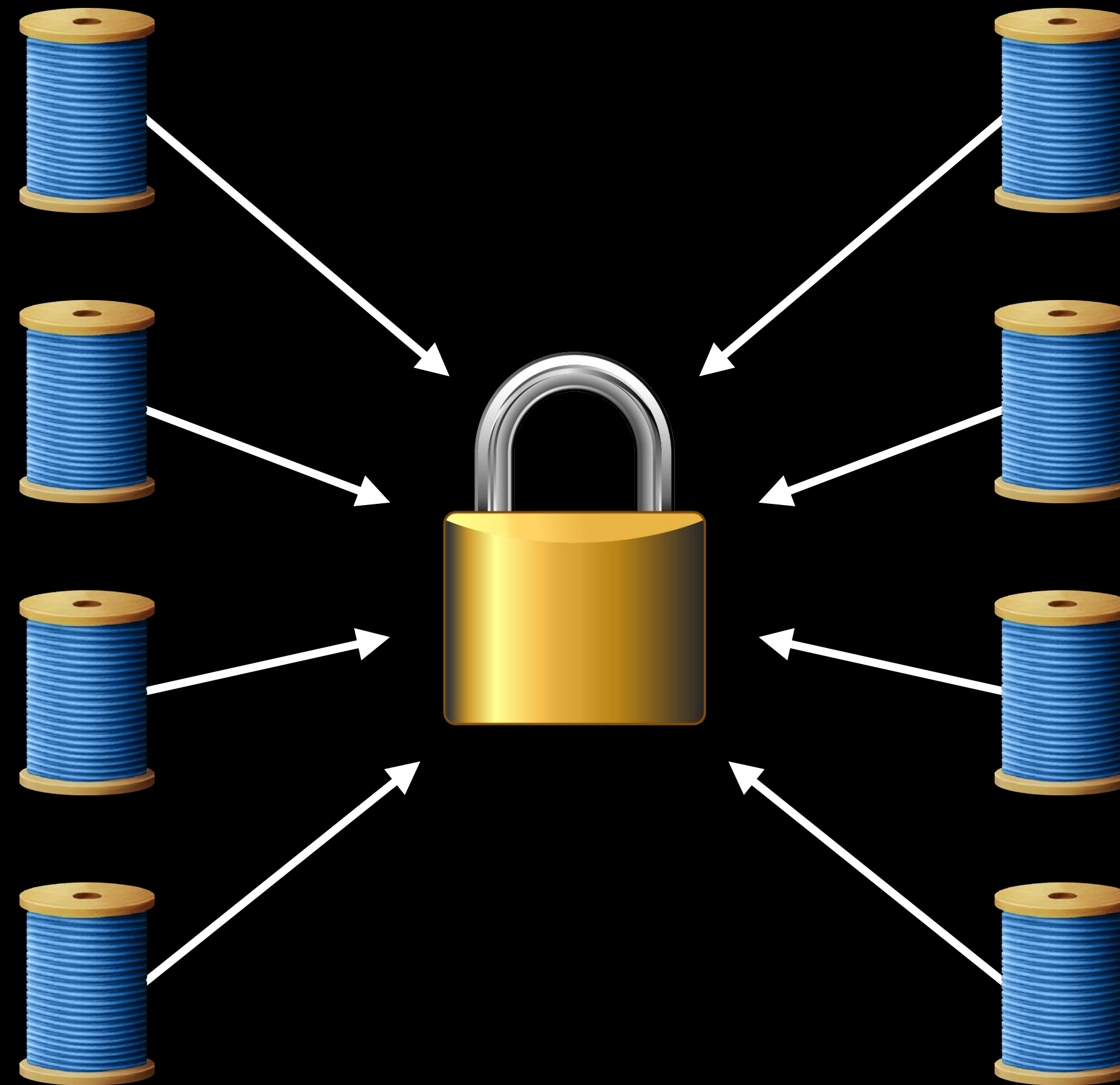


Repeatedly waiting for exclusive access to contended resources

Repeatedly switching between independent operations

Repeatedly bouncing an operation between threads

# Lock Contention





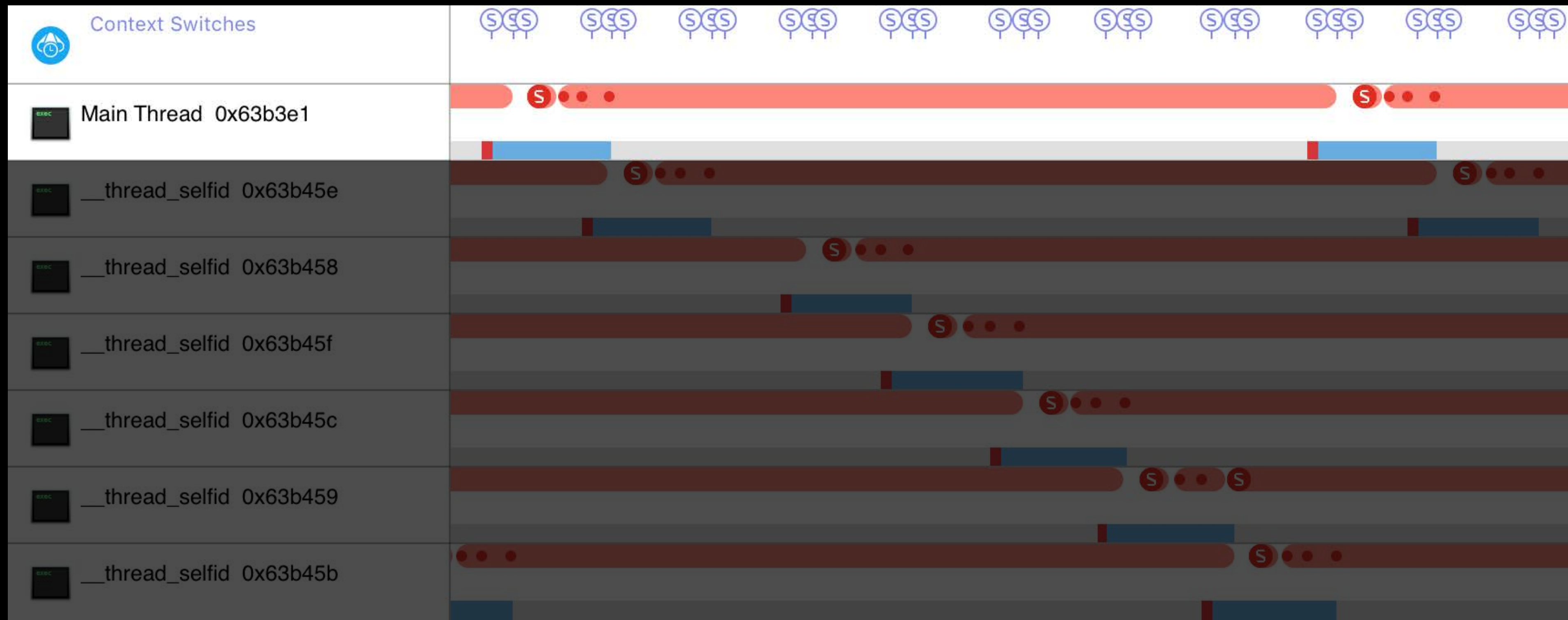
# Lock Contention

## Visualization in Instruments



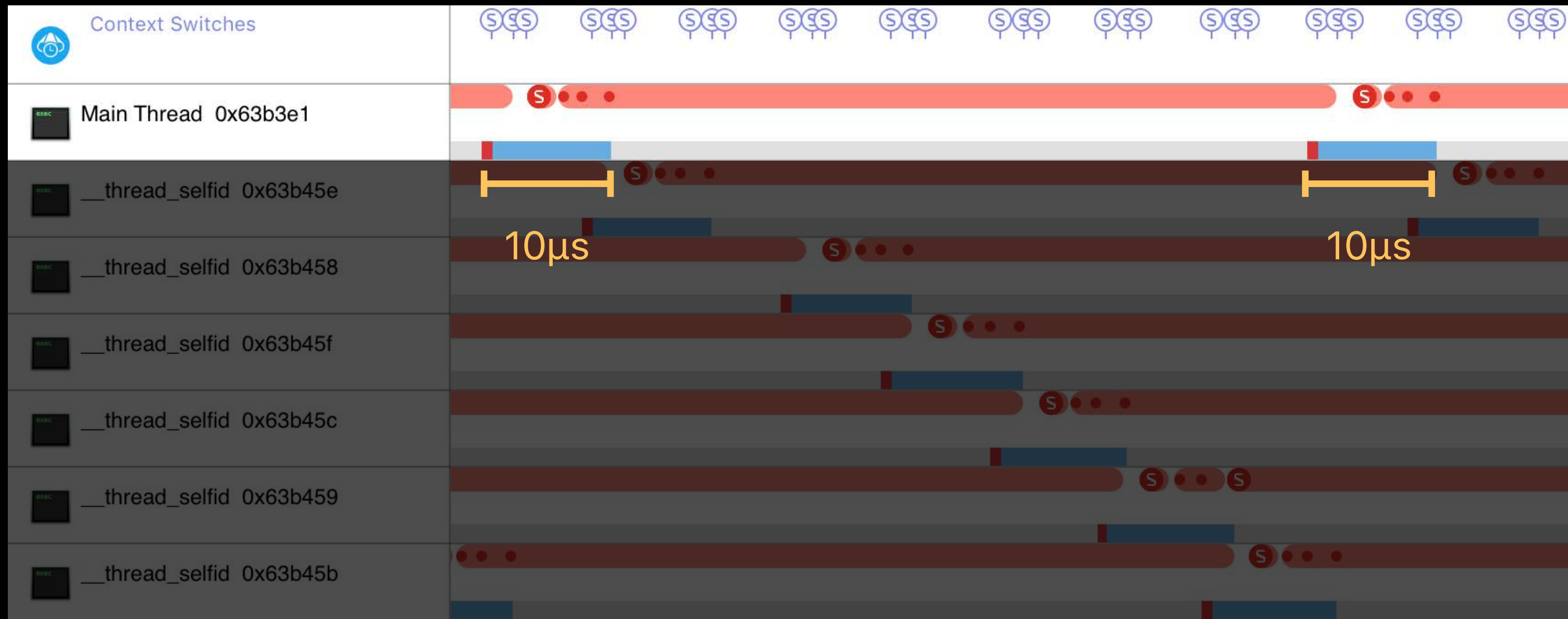
# Lock Contention

## Visualization in Instruments



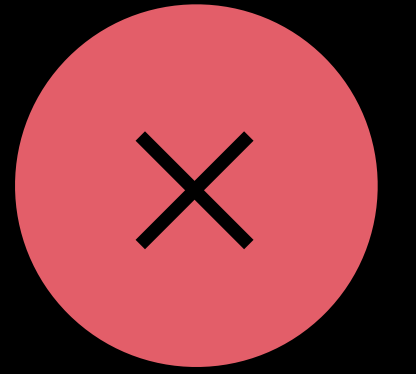
# Lock Contention

## Visualization in Instruments

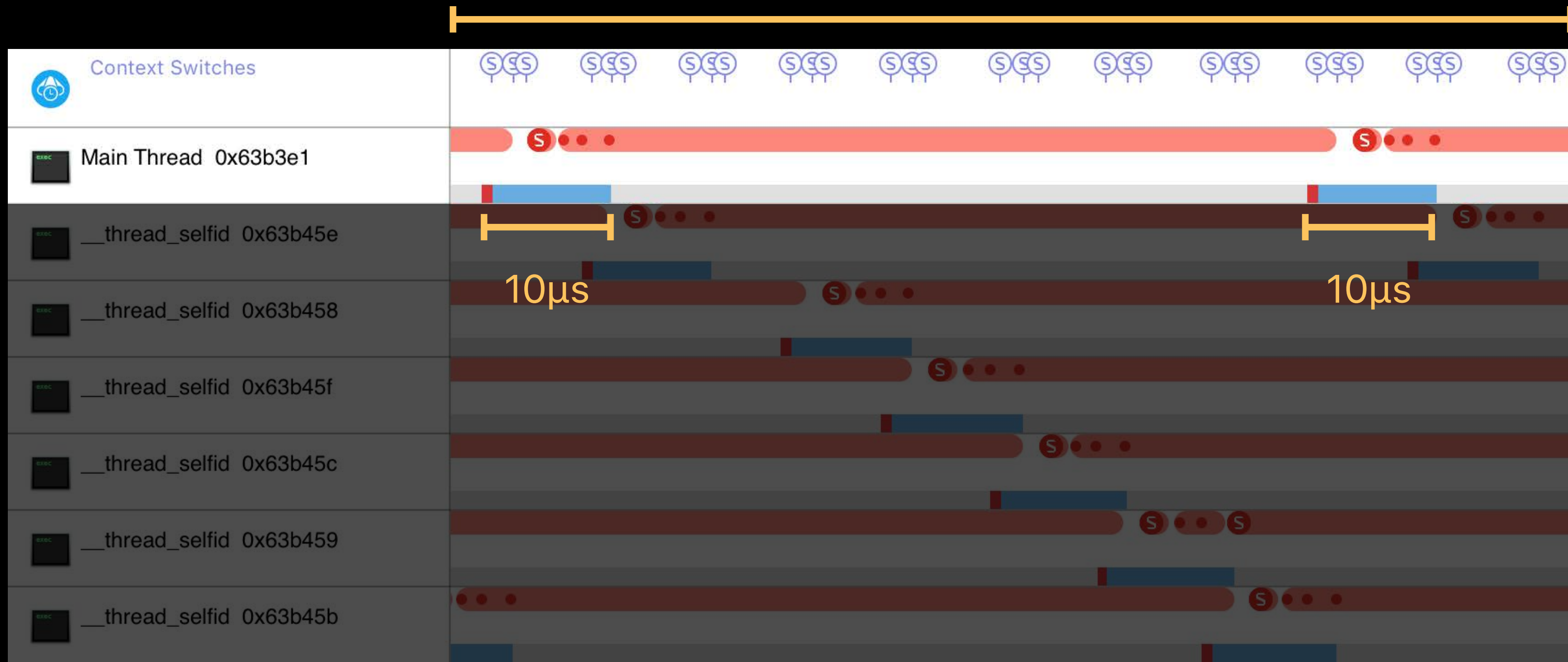


# Lock Contention

## Visualization in Instruments



Frequent context-switching



# Lock Contention

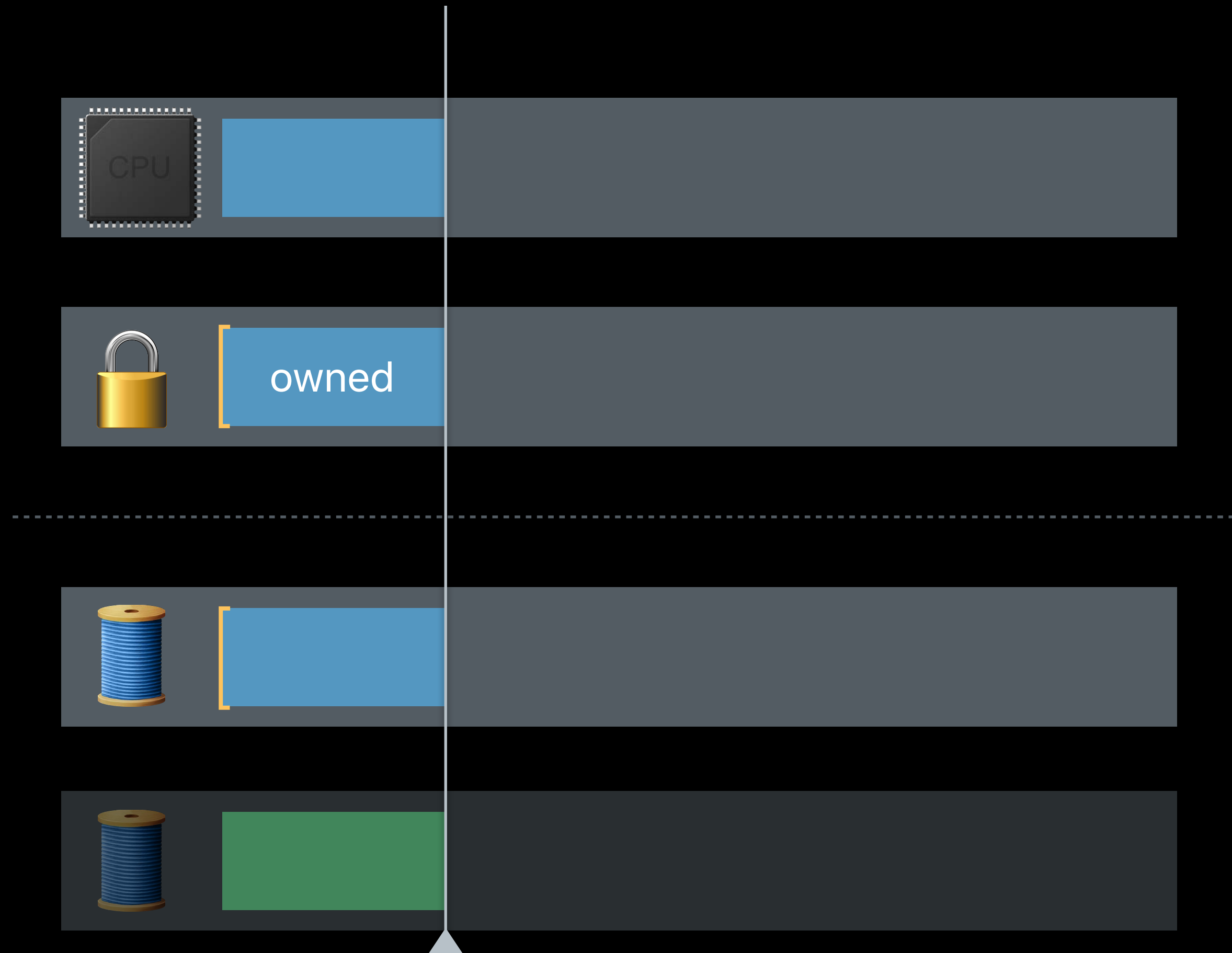


# Lock Contention



# Lock Contention

Fair locks



# Lock Contention

Fair locks





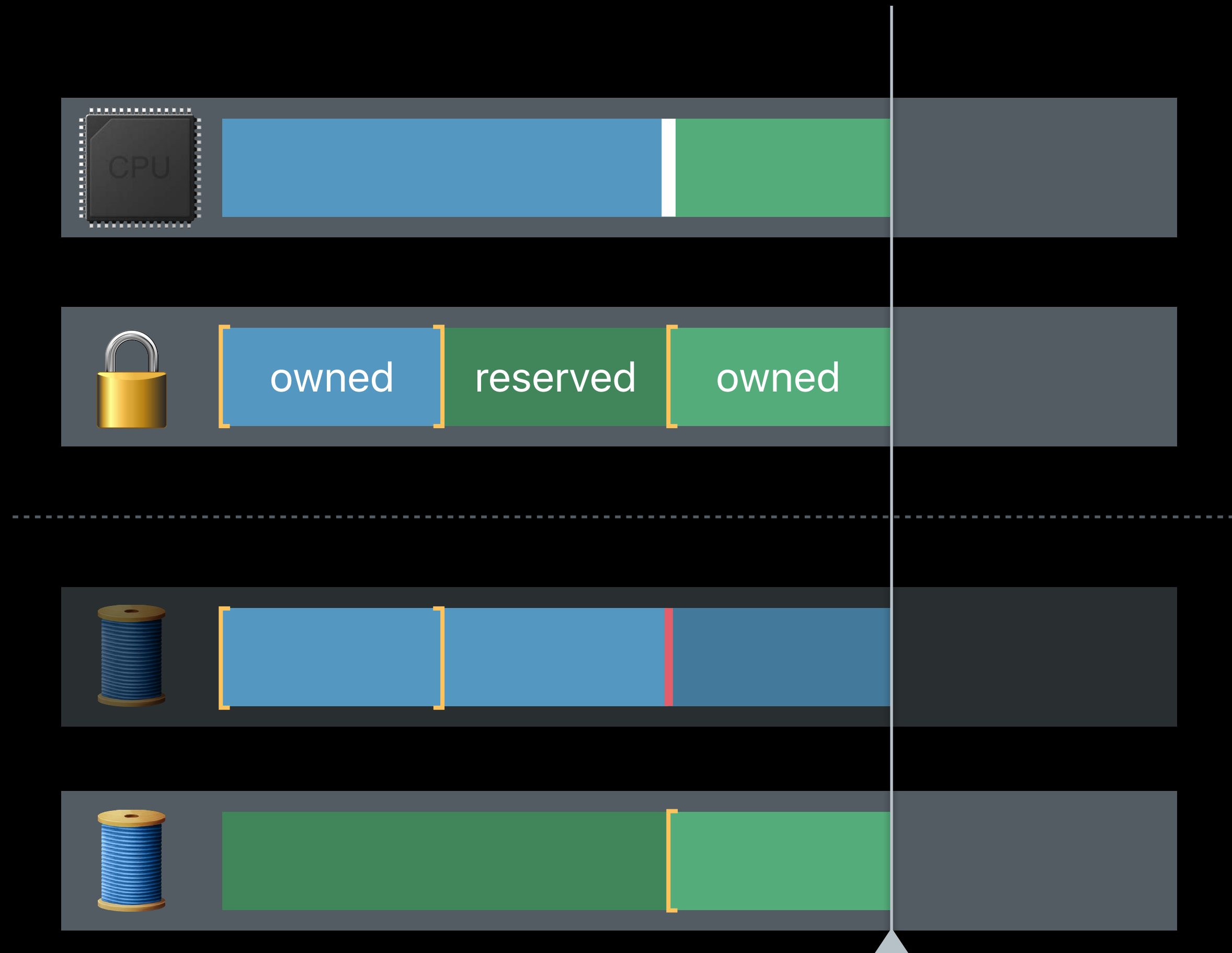
# Lock Contention

Fair locks



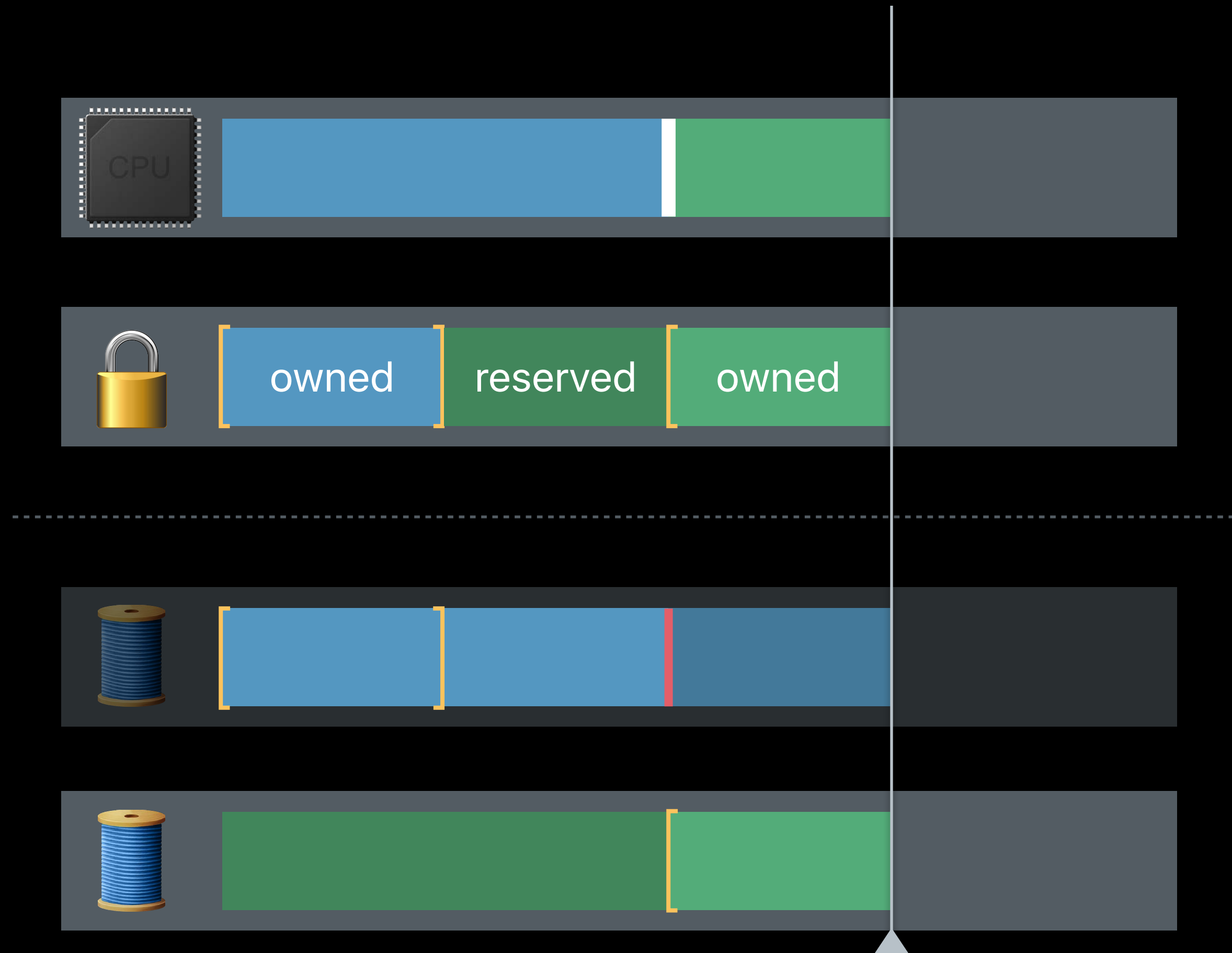
# Lock Contention

Fair locks



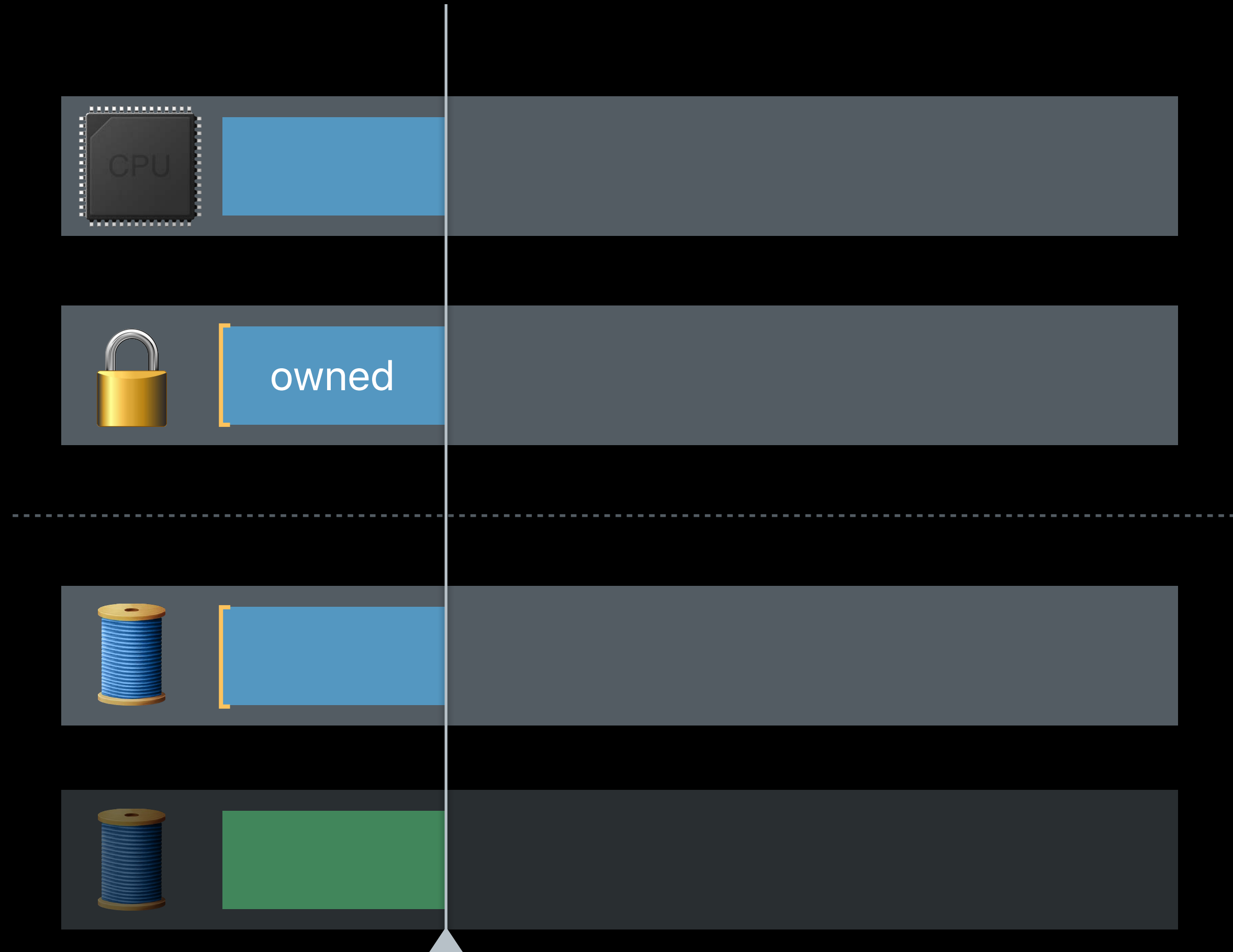
# Lock Contention

Fair locks



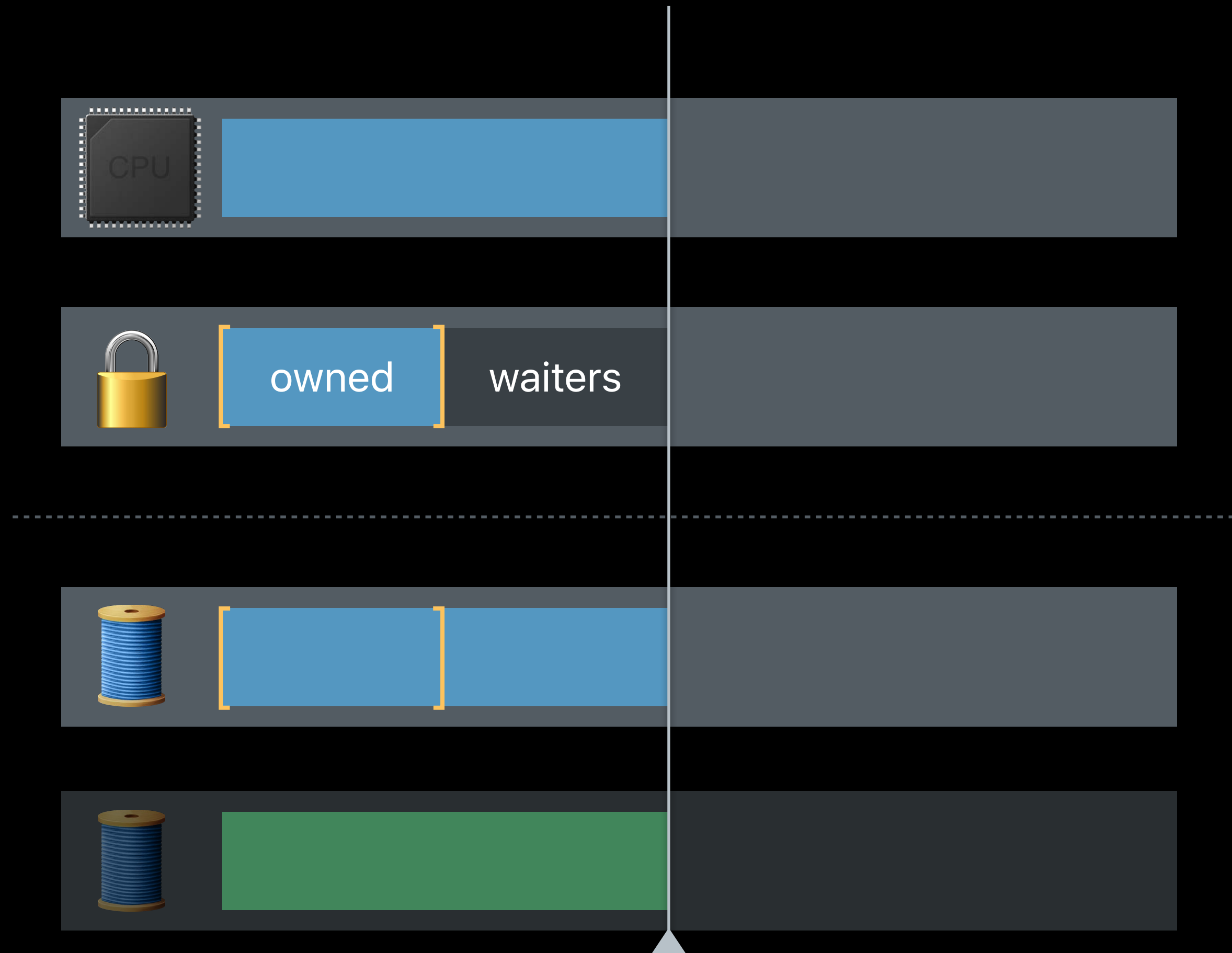
# Lock Contention

Unfair locks



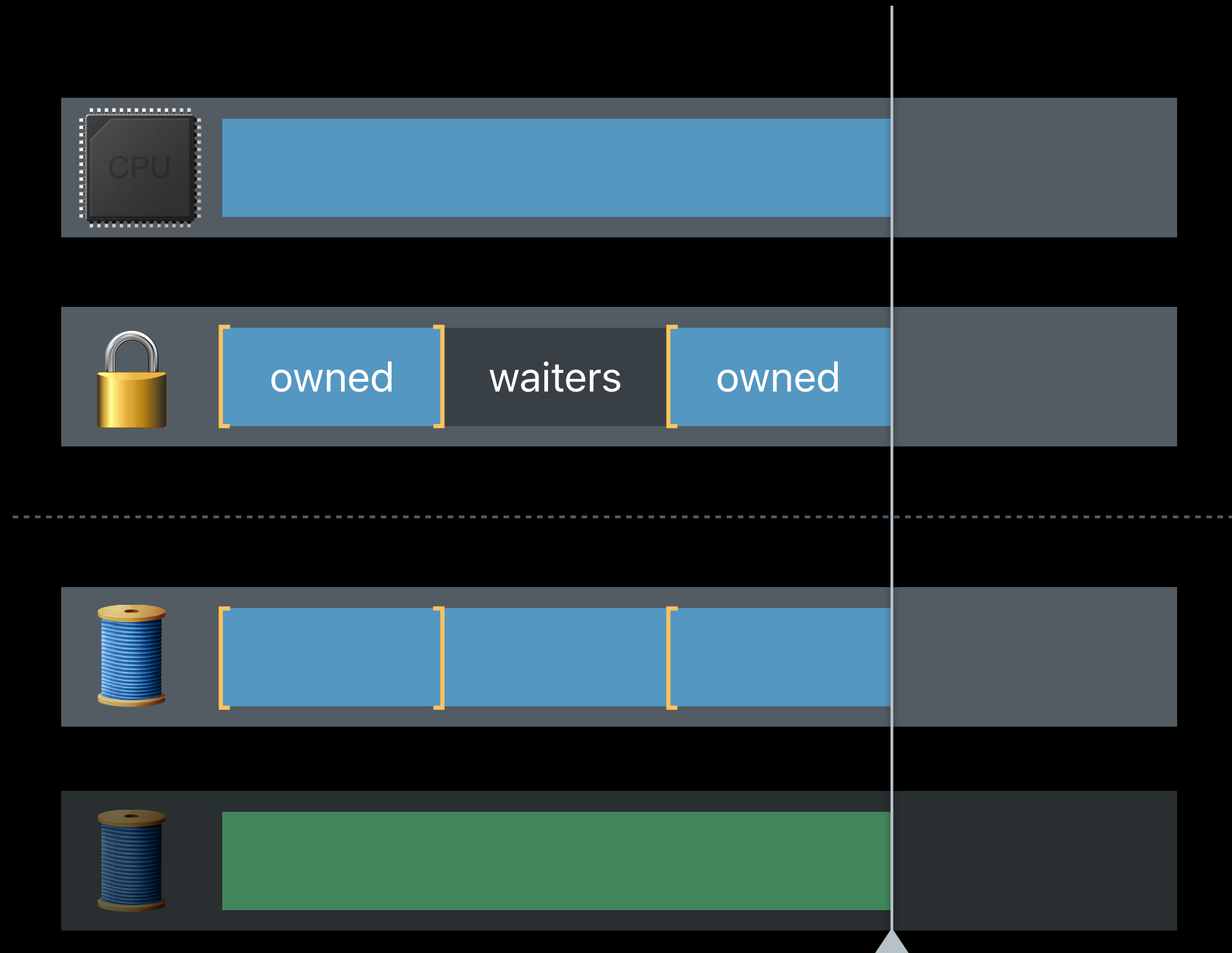
# Lock Contention

Unfair locks



# Lock Contention

Unfair locks



# Lock Contention

Use the right lock for the job

	Unfair	Fair
Available types	<code>os_unfair_lock</code>	<code>pthread_mutex_t</code> , <code>NSLock</code> <code>DispatchQueue.sync</code>
Contended lock re-acquisition	Can steal the lock	Context switches to next waiter
Subject to waiter starvation	Yes	No

# Lock Ownership





# Lock Ownership

Ownership helps resolve priority inversion

- High priority waiter
- Low priority owner



# Lock Ownership

Single Owner

Serial queues

`DispatchWorkItem.wait`

`os_unfair_lock`

`pthread_mutex, NSLock`

# Lock Ownership

Single Owner

No Owner

Serial queues

`dispatch_semaphore`

`DispatchWorkItem.wait`

`dispatch_group`

`os_unfair_lock`

`pthread_cond, NSCondition`

`pthread_mutex, NSLock`

Queue suspension

# Lock Ownership

Single Owner

No Owner

Multiple Owners

Serial queues

`dispatch_semaphore`

Private concurrent queues

`DispatchWorkItem.wait`

`dispatch_group`

`pthread_rwlock`

`os_unfair_lock`

`pthread_cond, NSCondition`

`pthread_mutex, NSLock`

Queue suspension

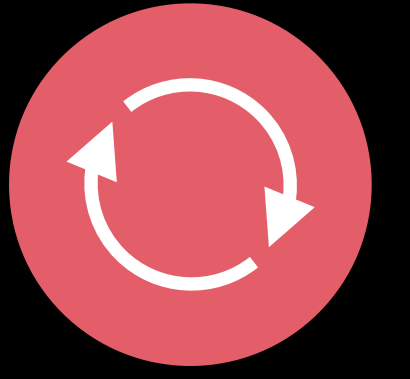
# Optimizing Lock Contention

Inefficient behaviors are often emergent properties

Visualize your app's behavior with Instruments

Use the right lock for the job

# Too Much of a Good Thing



Repeatedly waiting for exclusive access to contended resources

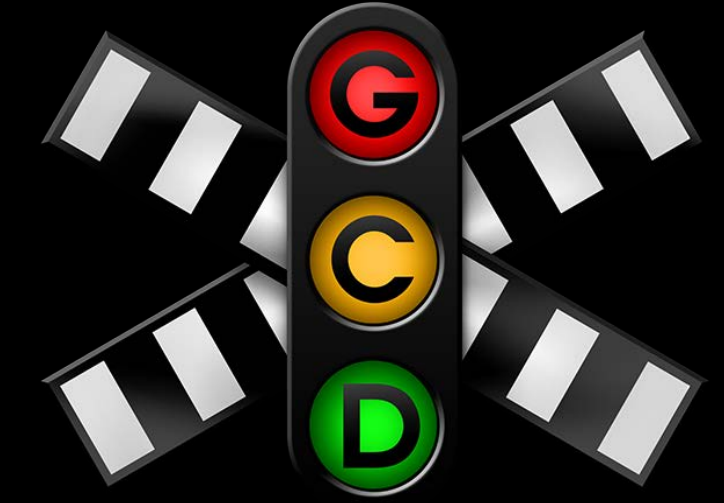
Repeatedly switching between independent operations

Repeatedly bouncing an operation between threads

# Using GCD for Concurrency

Daniel A. Steffen, Core Darwin

# Grand Central Dispatch



---

Simplifying iPhone App Development with Grand Central Dispatch

WWDC 2010

---

Asynchronous Design Patterns with Blocks, GCD, and XPC

WWDC 2012

---

Power, Performance, and Diagnostics: What's new in GCD and XPC

WWDC 2014

---

Building Responsive and Efficient Apps with GCD

WWDC 2015

---

Concurrent Programming with GCD in Swift 3

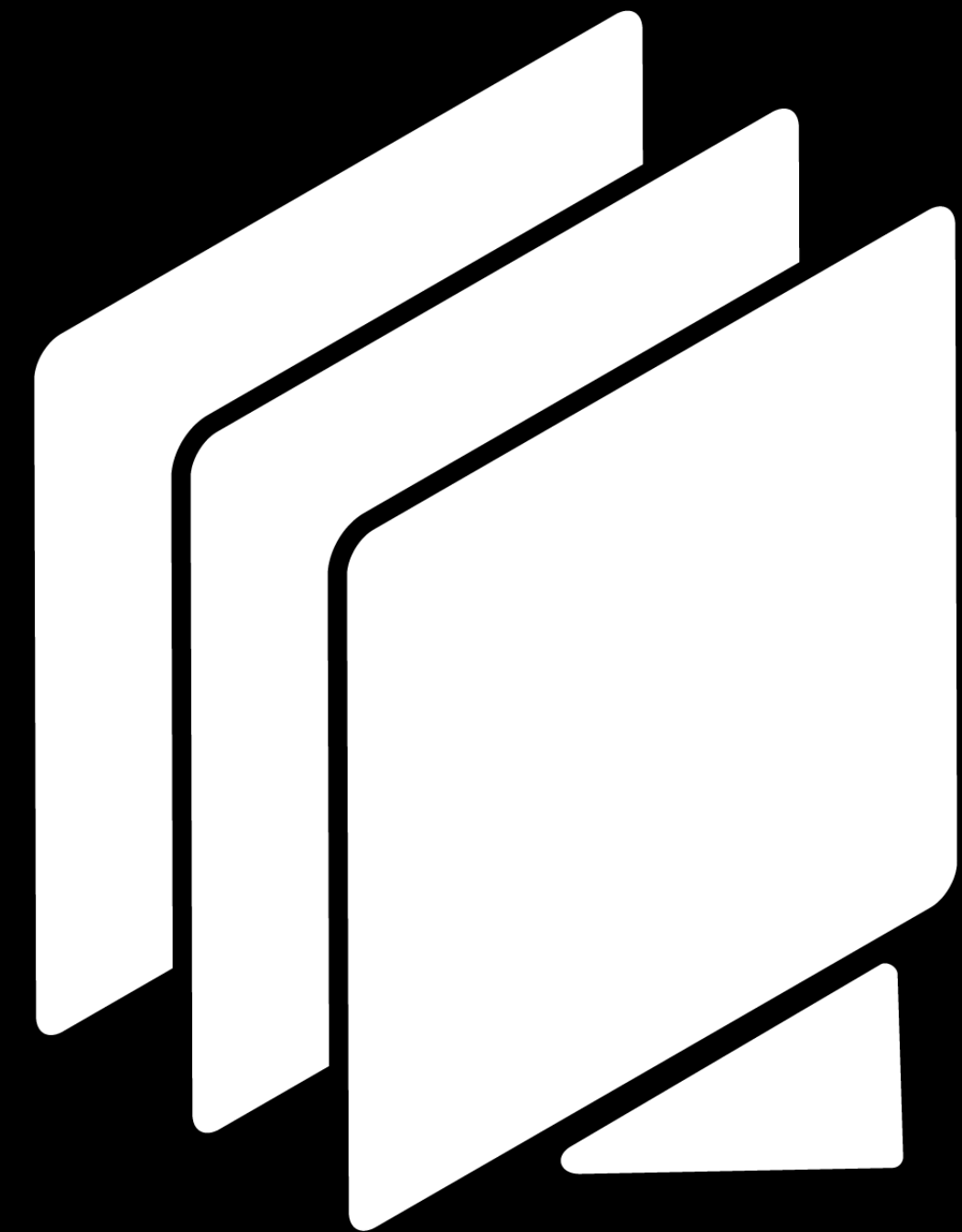
WWDC 2016

---



# Serial Dispatch Queue

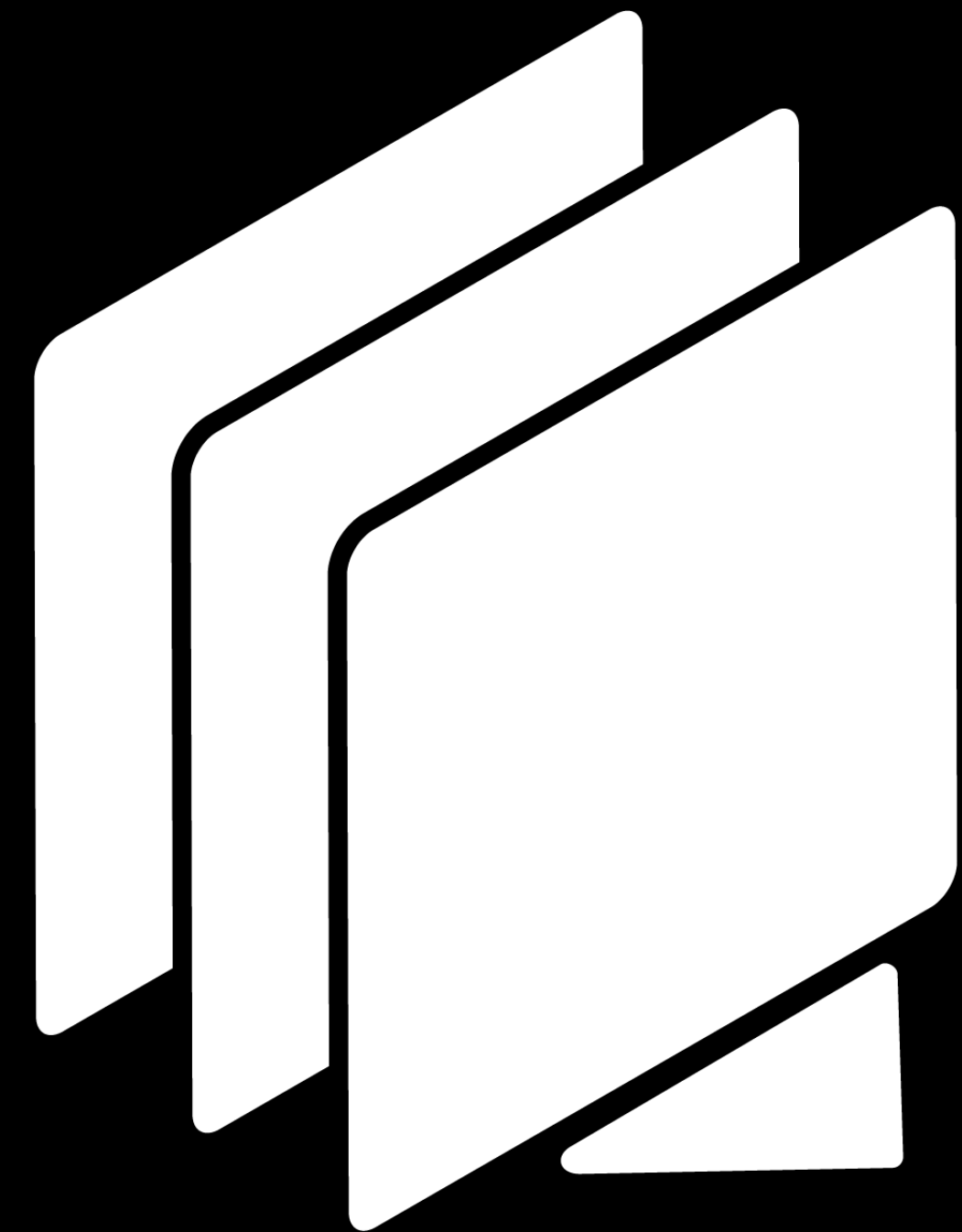
Fundamental GCD primitive



# Serial Dispatch Queue

Fundamental GCD primitive

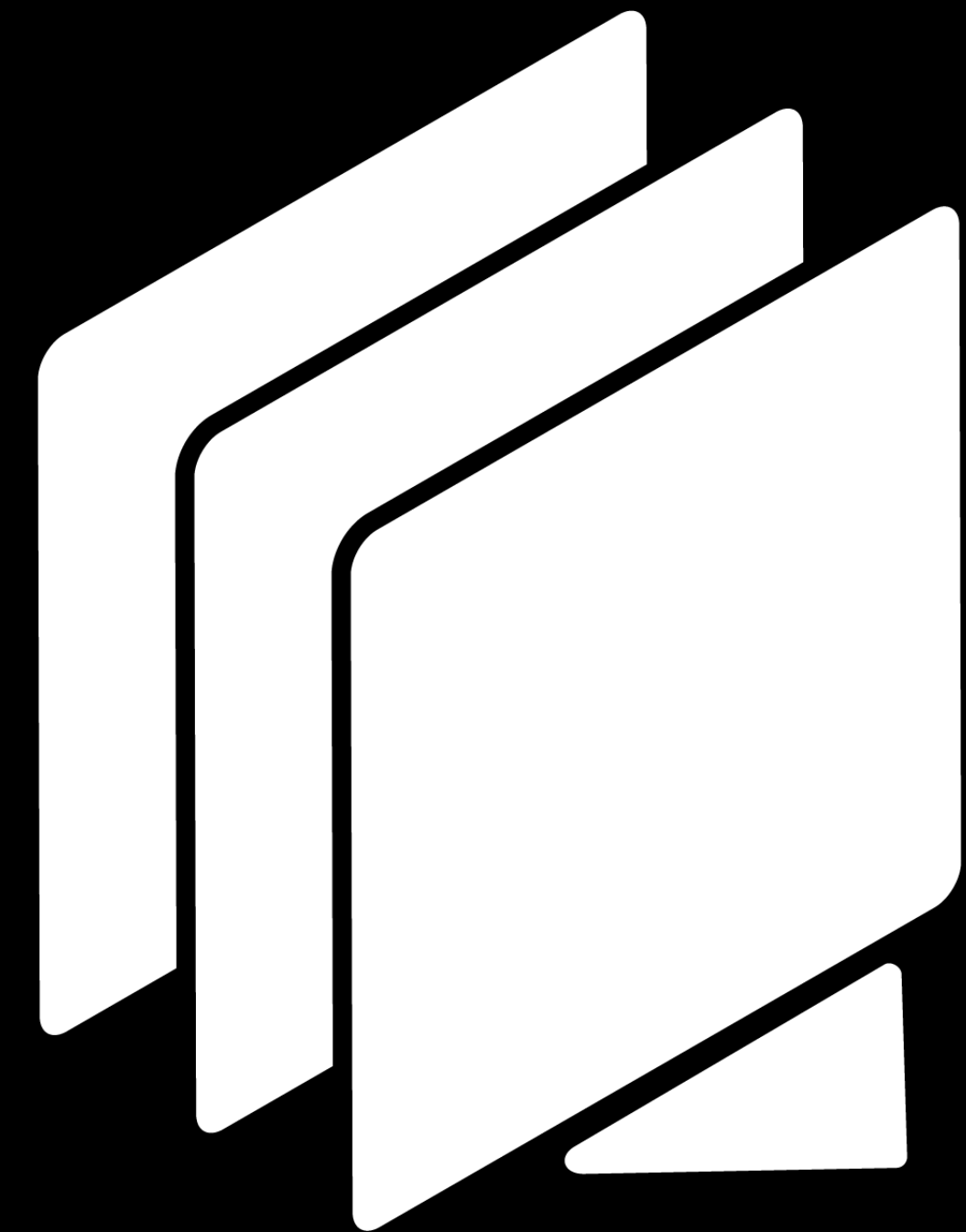
- Mutual exclusion
- FIFO ordered



# Serial Dispatch Queue

Fundamental GCD primitive

- Mutual exclusion
- FIFO ordered
- Concurrent atomic enqueue
- Single dequeuer



# Serial Dispatch Queue

```
let queue = DispatchQueue(label: "com.example.queue")
queue.async { /* 1 */ }
queue.async { /* 2 */ }
queue.sync  { /* 3 */ }
```

# Serial Dispatch Queue

queue

```
let queue = DispatchQueue(label: "com.example.queue")
```

```
queue.async { /* 1 */ }
```

```
queue.async { /* 2 */ }
```

```
queue.sync { /* 3 */ }
```

# Serial Dispatch Queue

1 2

queue



```
queue.async { 1 }
```



```
queue.async { 2 }
```

```
let queue = DispatchQueue(label: "com.example.queue")
```

```
queue.async { /* 1 */ }
```

```
queue.async { /* 2 */ }
```

```
queue.sync { /* 3 */ }
```

# Serial Dispatch Queue

1 2 3

queue

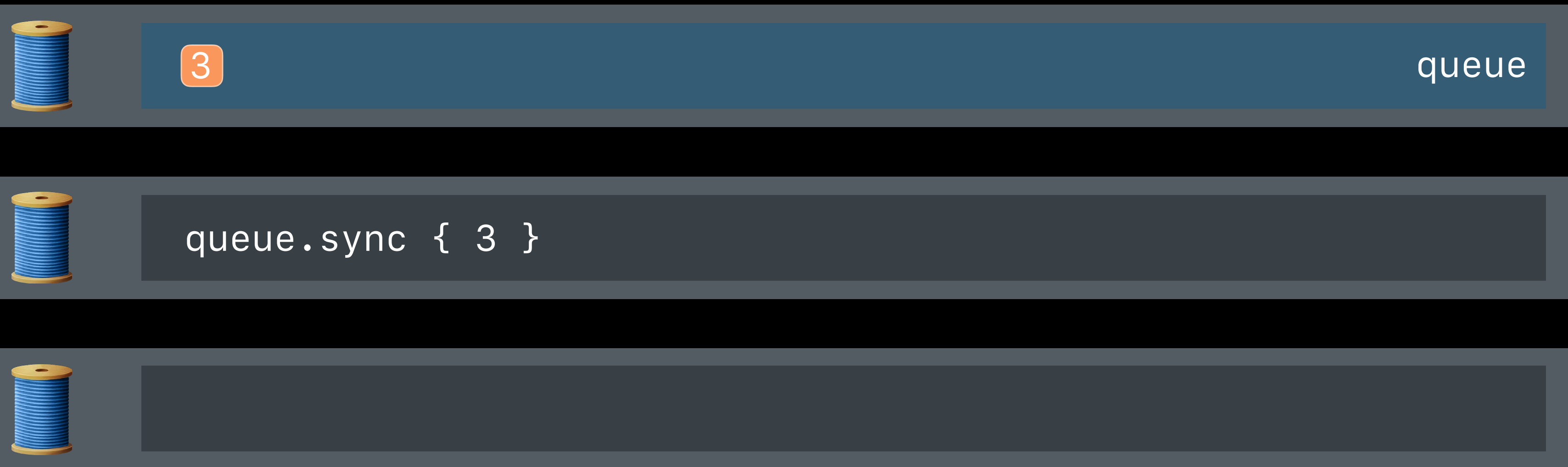


```
queue.sync { 3 }
```



```
let queue = DispatchQueue(label: "com.example.queue")
queue.async { /* 1 */ }
queue.async { /* 2 */ }
queue.sync { /* 3 */ }
```

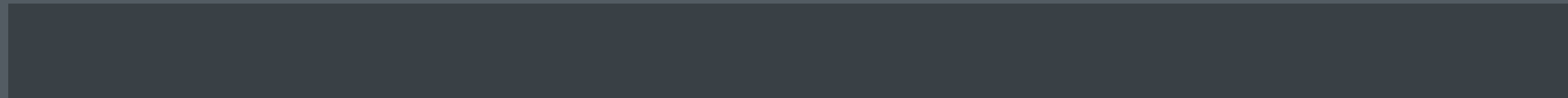
# Serial Dispatch Queue



```
let queue = DispatchQueue(label: "com.example.queue")
queue.async { /* 1 */ }
queue.async { /* 2 */ }
queue.sync { /* 3 */ }
```

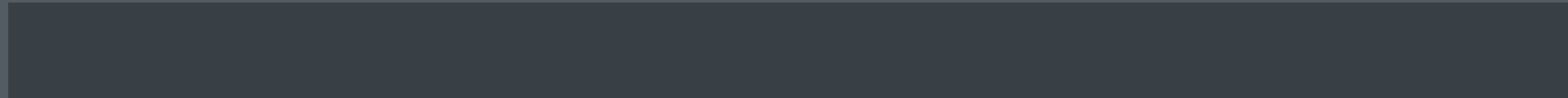


# Serial Dispatch Queue



`queue.sync { 3 }`

`queue`



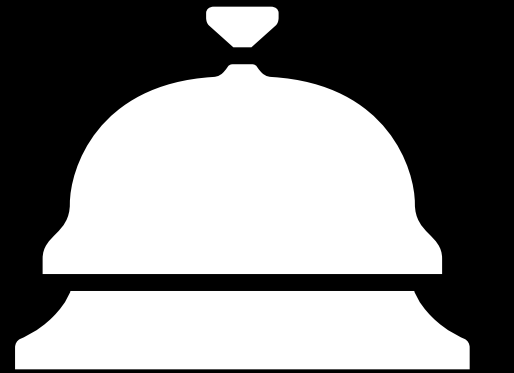
```
let queue = DispatchQueue(label: "com.example.queue")
```

```
queue.async { /* 1 */ }
```

```
queue.async { /* 2 */ }
```

```
queue.sync { /* 3 */ }
```

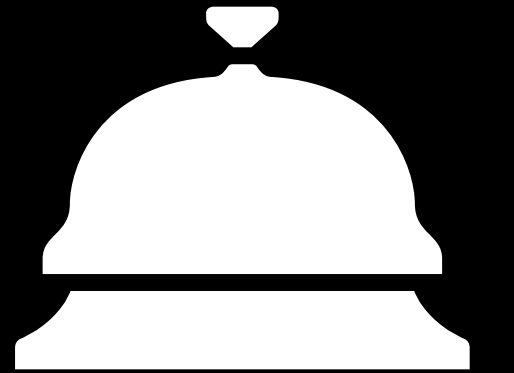
# Dispatch Source



Event monitoring primitive

```
let source = DispatchSource.makeReadSource(fileDescriptor: fd, queue: queue)
source.setEventHandler { read(fd) }
source.setCancelHandler { close(fd) }
source.activate()
```

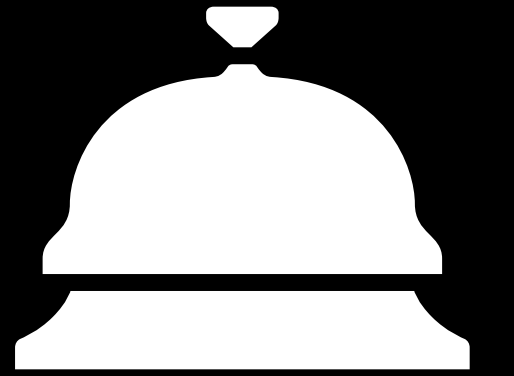
# Dispatch Source



Event monitoring primitive

```
let source = DispatchSource.makeReadSource(fileDescriptor: fd, queue: queue)
source.setEventHandler { read(fd) }
source.setCancelHandler { close(fd) }
source.activate()
```

# Dispatch Source

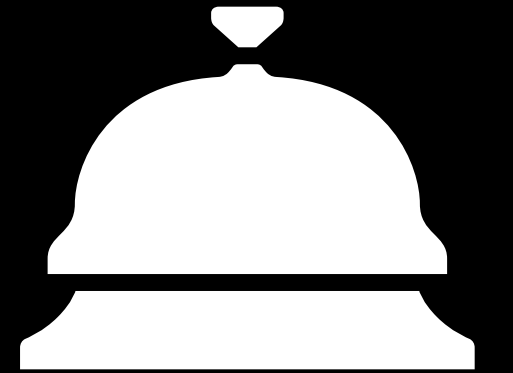


Event monitoring primitive

- Event handler executes on target queue

```
let source = DispatchSource.makeReadSource(fileDescriptor: fd, queue: queue)
source.setEventHandler { read(fd) }
source.setCancelHandler { close(fd) }
source.activate()
```

# Dispatch Source

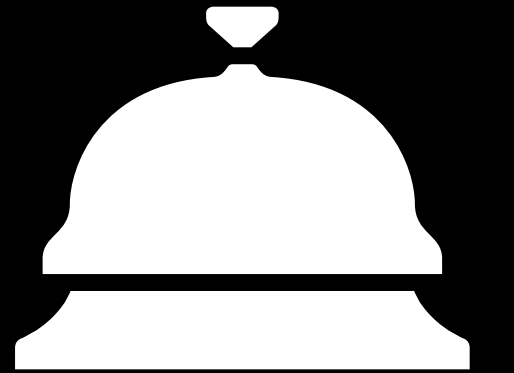


## Event monitoring primitive

- Event handler executes on target queue
- Invalidation pattern with explicit cancellation

```
let source = DispatchSource.makeReadSource(fileDescriptor: fd, queue: queue)
source.setEventHandler { read(fd) }
source.setCancelHandler { close(fd) }
source.activate()
```

# Dispatch Source



## Event monitoring primitive

- Event handler executes on target queue
- Invalidation pattern with explicit cancellation
- Initial setup followed by activate

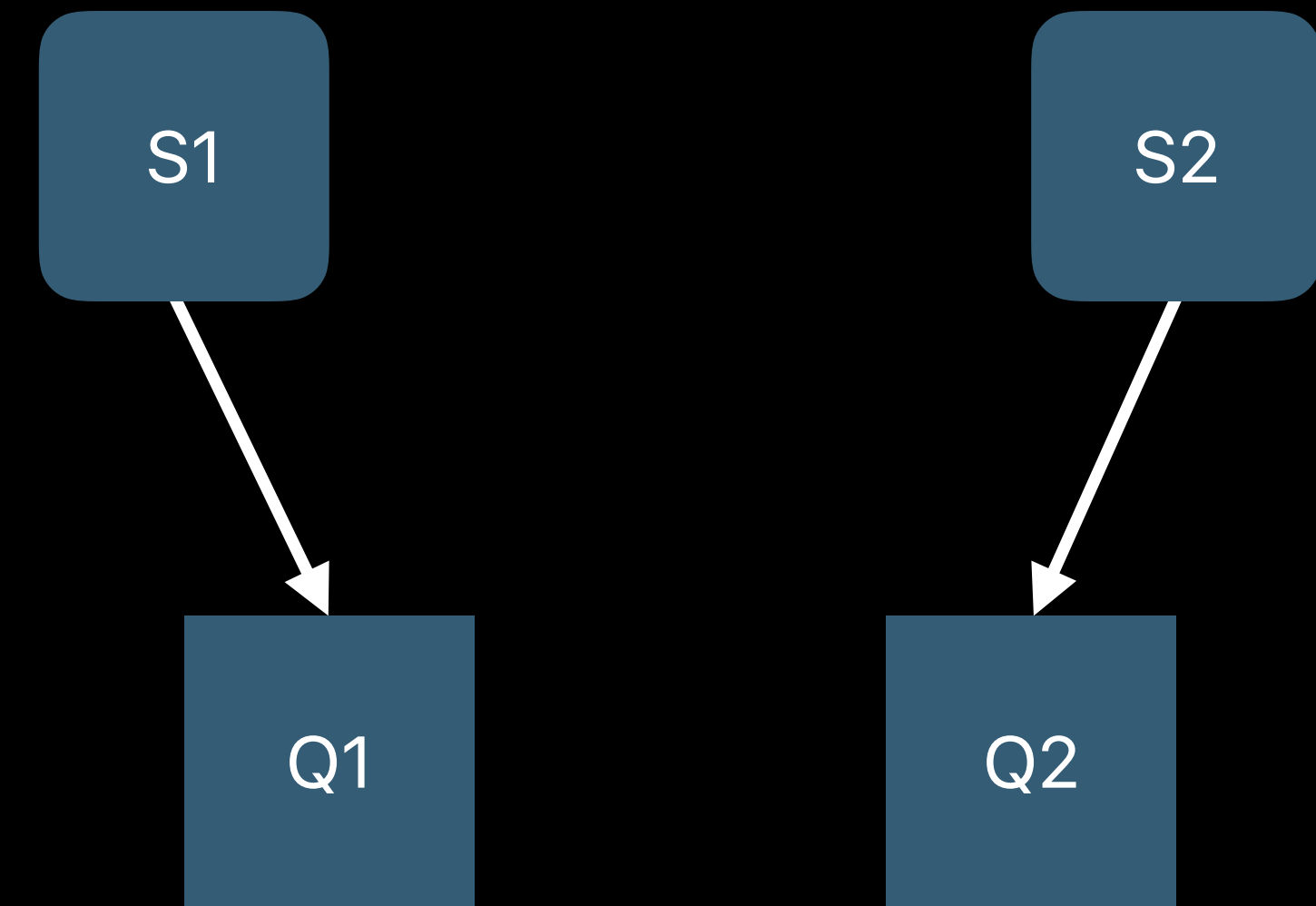
```
let source = DispatchSource.makeReadSource(fileDescriptor: fd, queue: queue)
source.setEventHandler { read(fd) }
source.setCancelHandler { close(fd) }
source.activate()
```

# Target Queue Hierarchy

Serial queues and sources can form a tree

# Target Queue Hierarchy

Serial queues and sources can form a tree

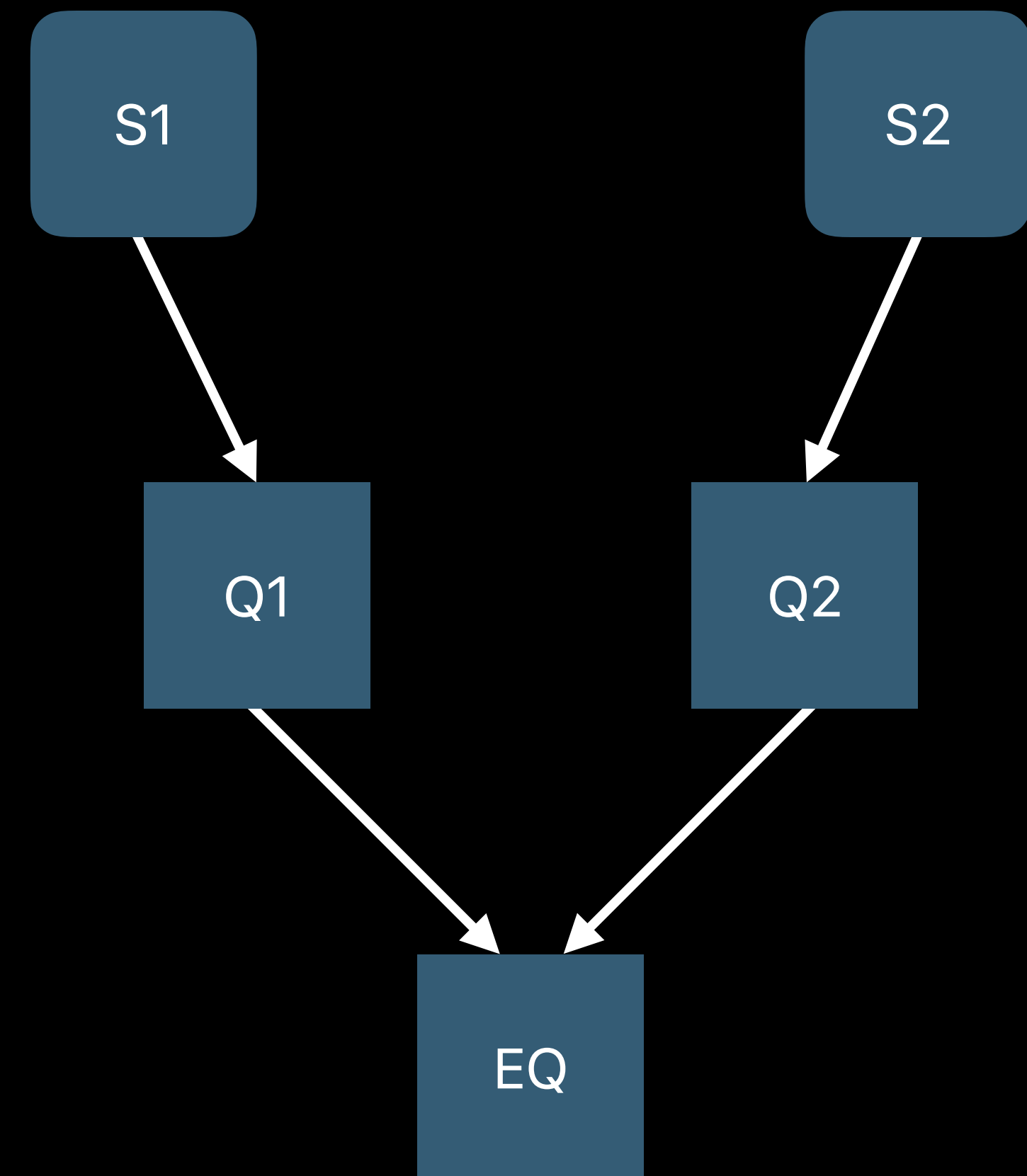




# Target Queue Hierarchy

Serial queues and sources can form a tree

```
let Q1 = DispatchQueue(label: "Q1", target: EQ )  
let Q2 = DispatchQueue(label: "Q2", target: EQ )
```



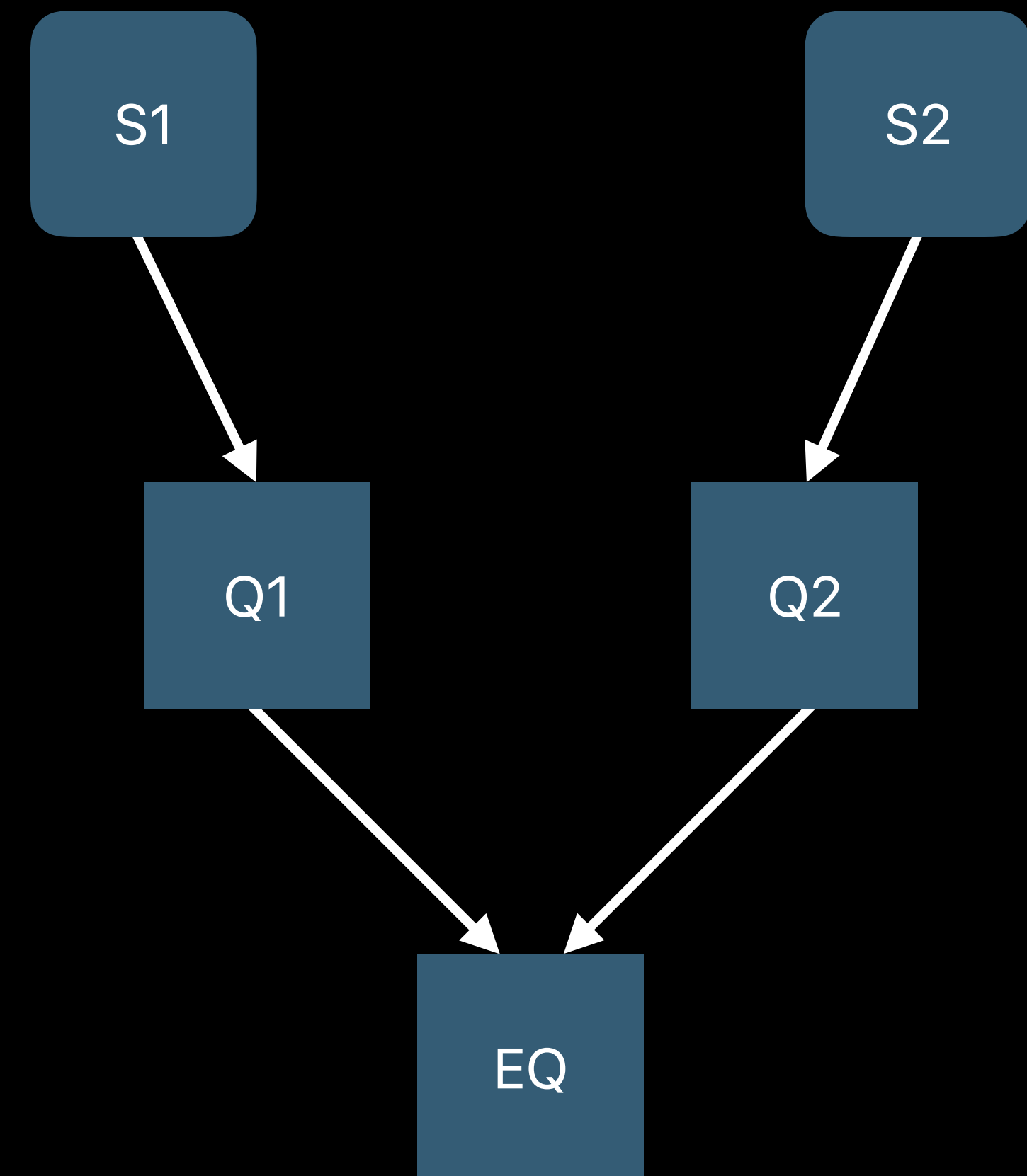
# Target Queue Hierarchy

Serial queues and sources can form a tree

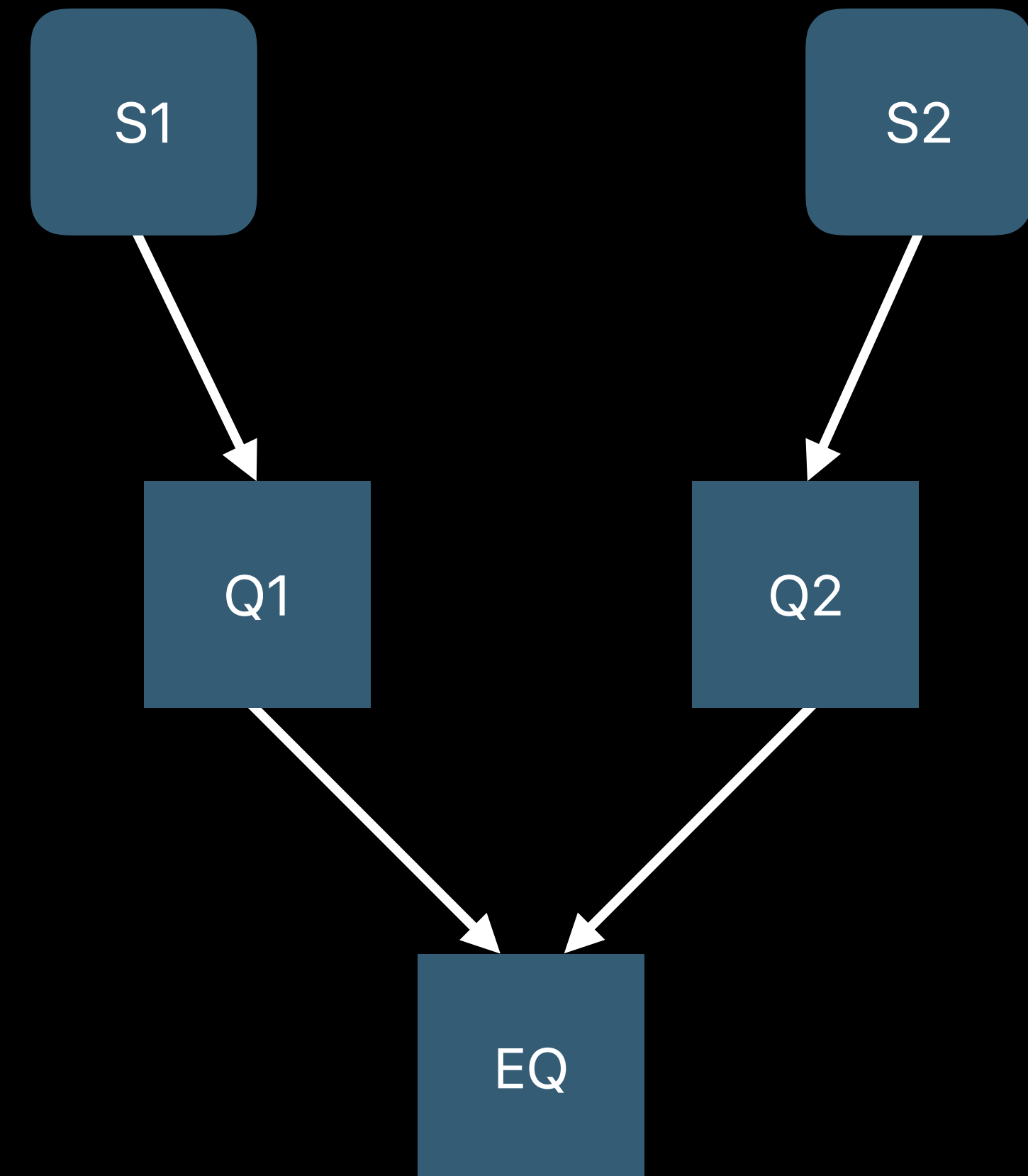
Shared single mutual exclusion context

Independent individual queue order

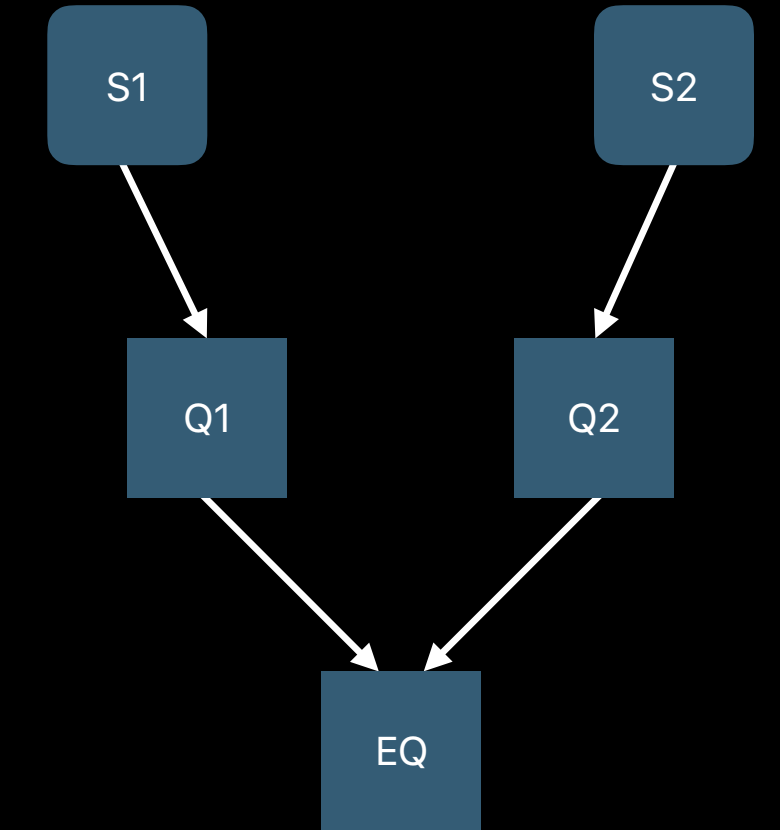
```
let Q1 = DispatchQueue(label: "Q1", target: EQ )  
let Q2 = DispatchQueue(label: "Q2", target: EQ )
```



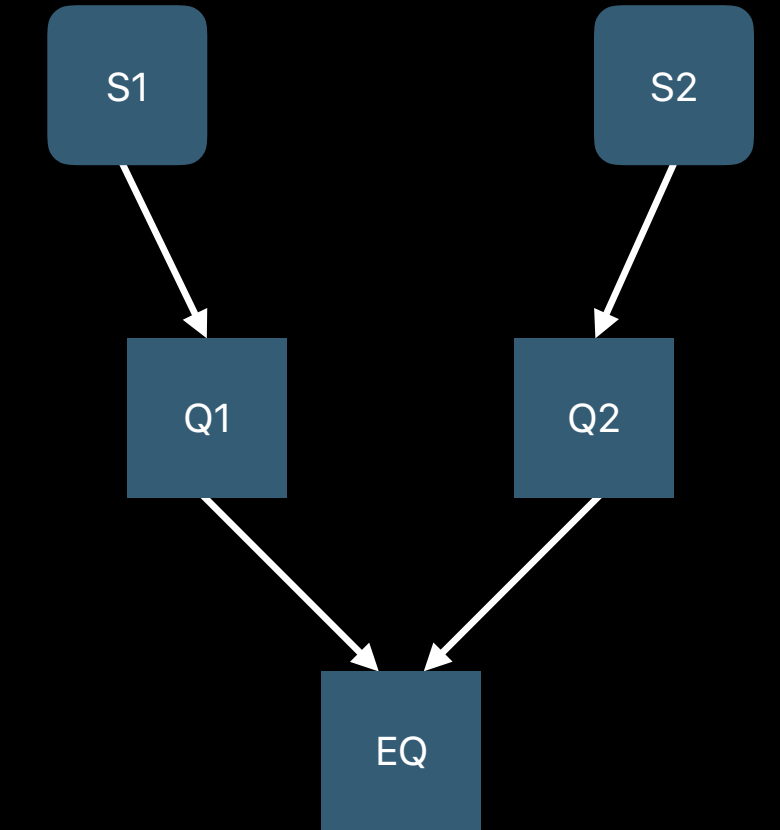
# Target Queue Hierarchy



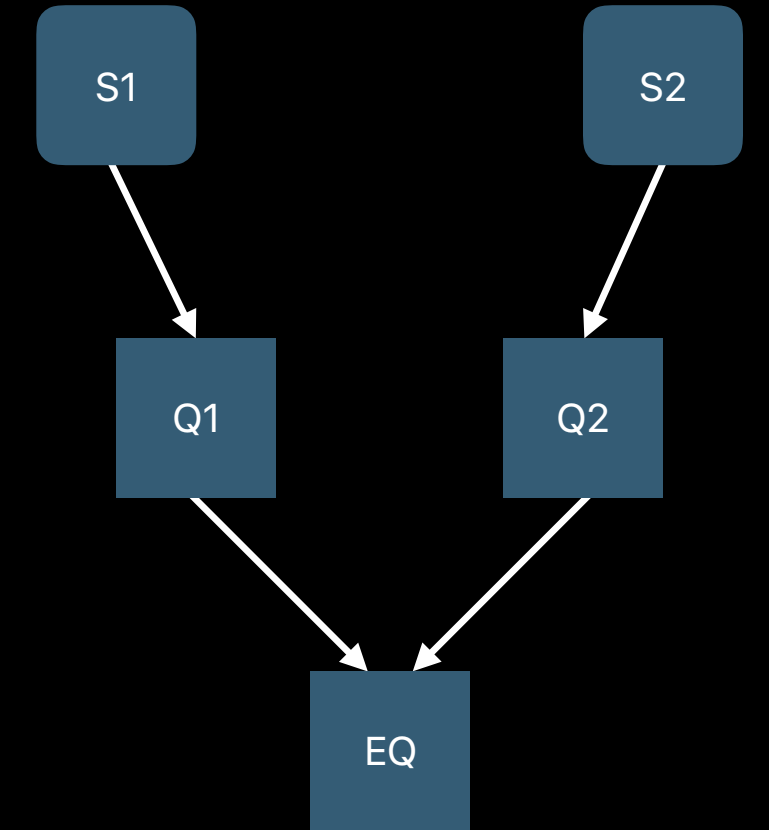
# Target Queue Hierarchy



# Target Queue Hierarchy



# Target Queue Hierarchy



# Quality of Service

Abstract notion of priority

Provides explicit classification of your work

Affects various execution properties

User Interactive

User Initiated

Utility

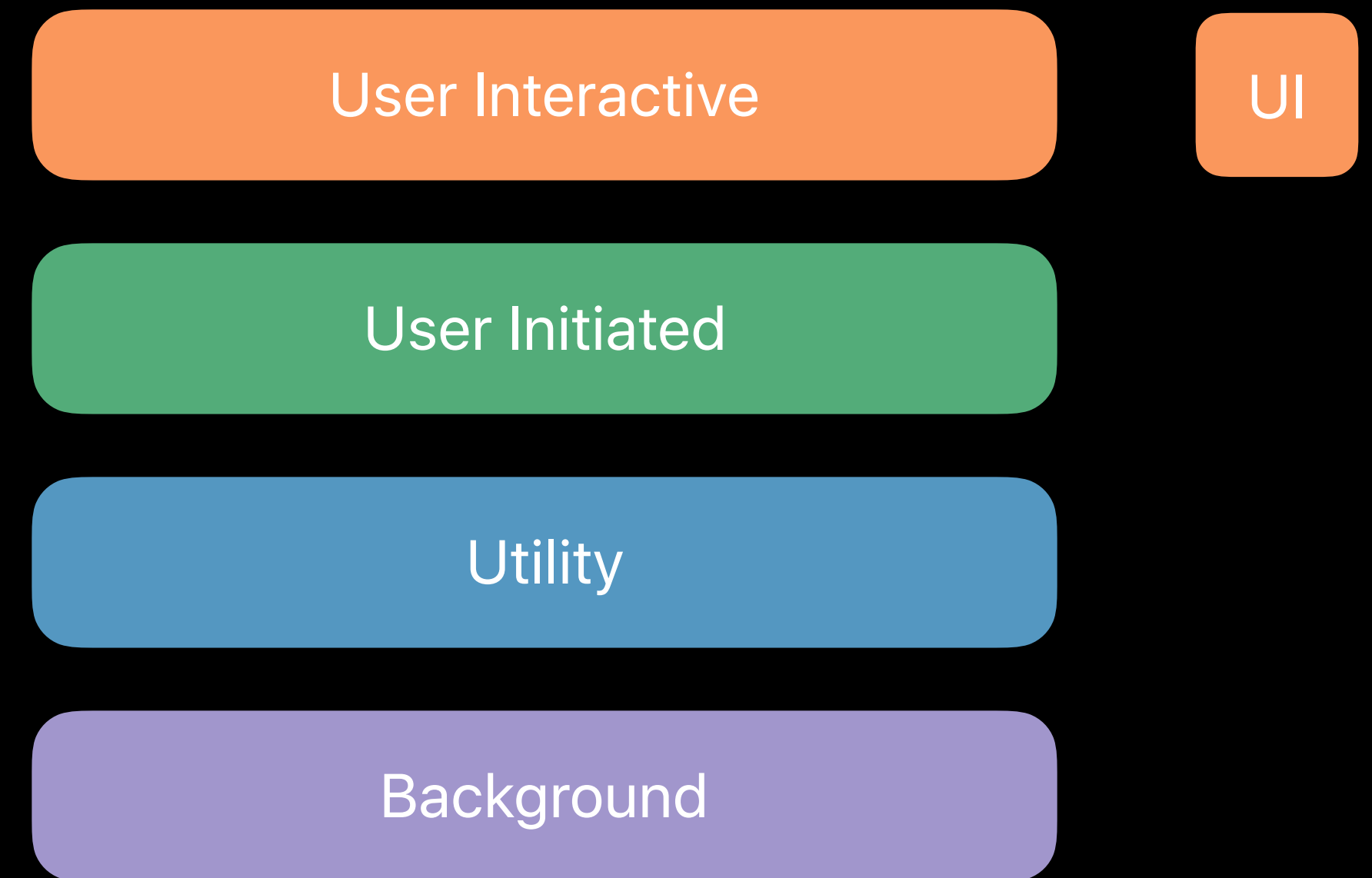
Background

# Quality of Service

Abstract notion of priority

Provides explicit classification of your work

Affects various execution properties



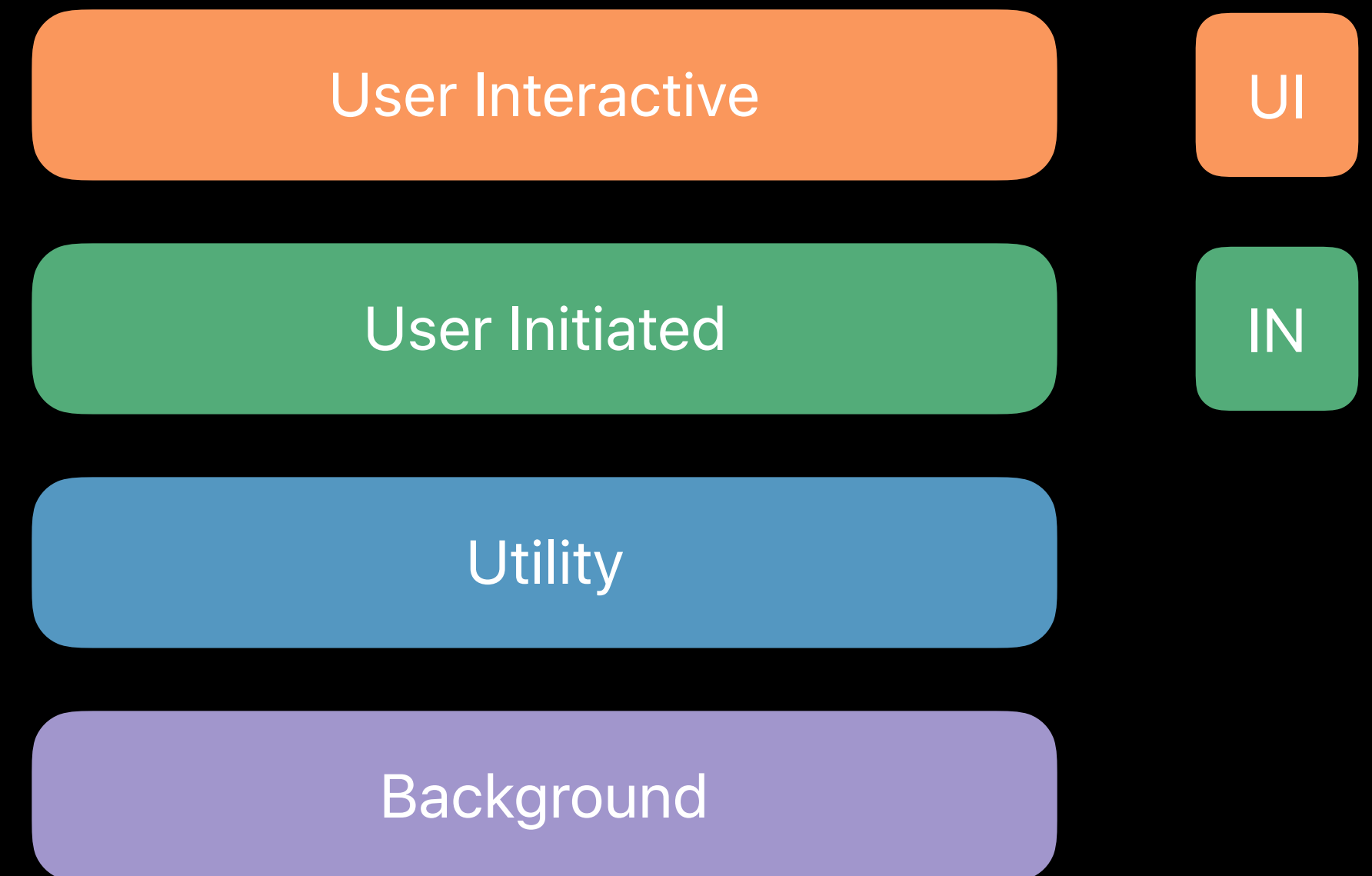


# Quality of Service

Abstract notion of priority

Provides explicit classification of your work

Affects various execution properties

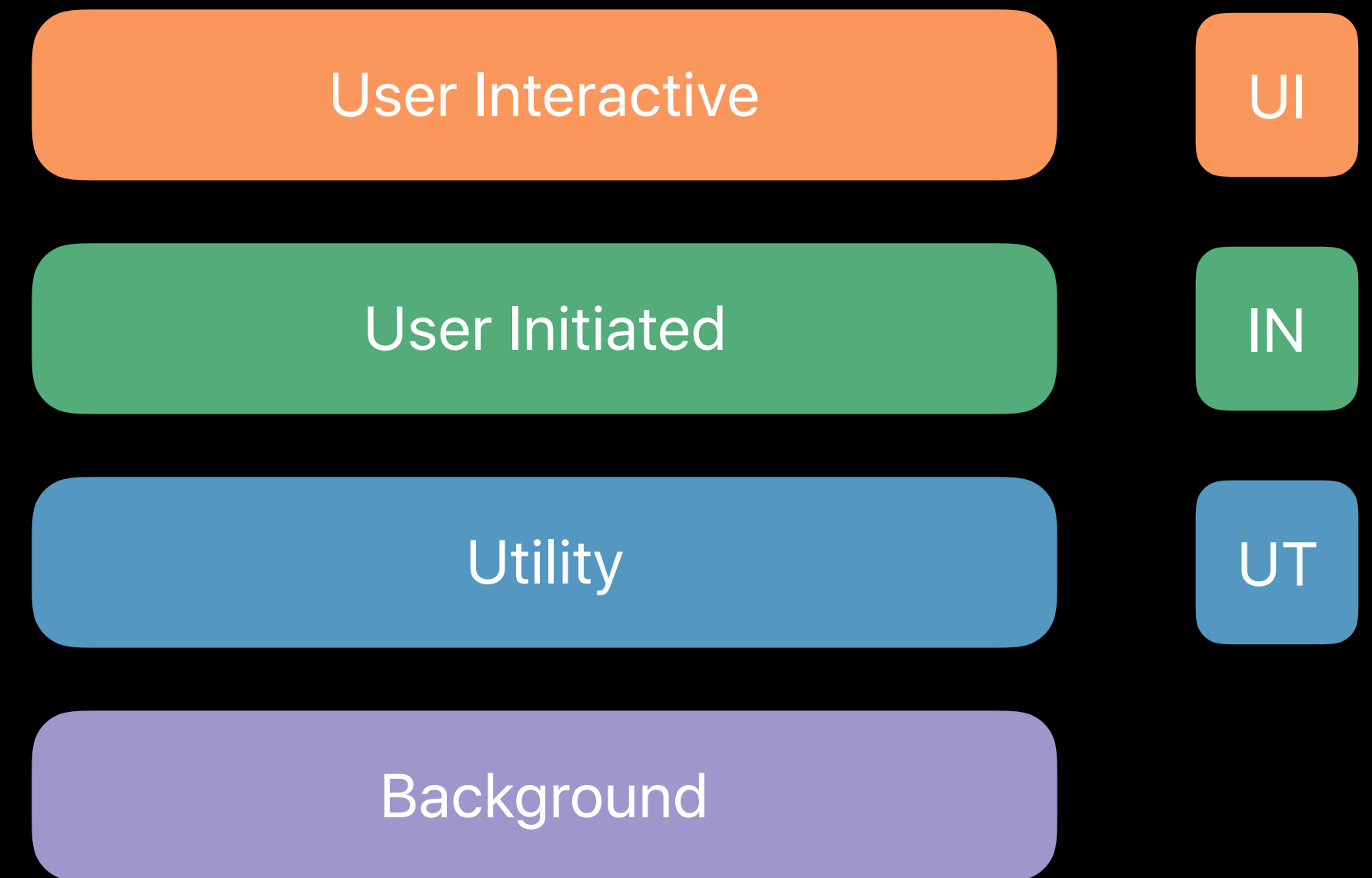


# Quality of Service

Abstract notion of priority

Provides explicit classification of your work

Affects various execution properties

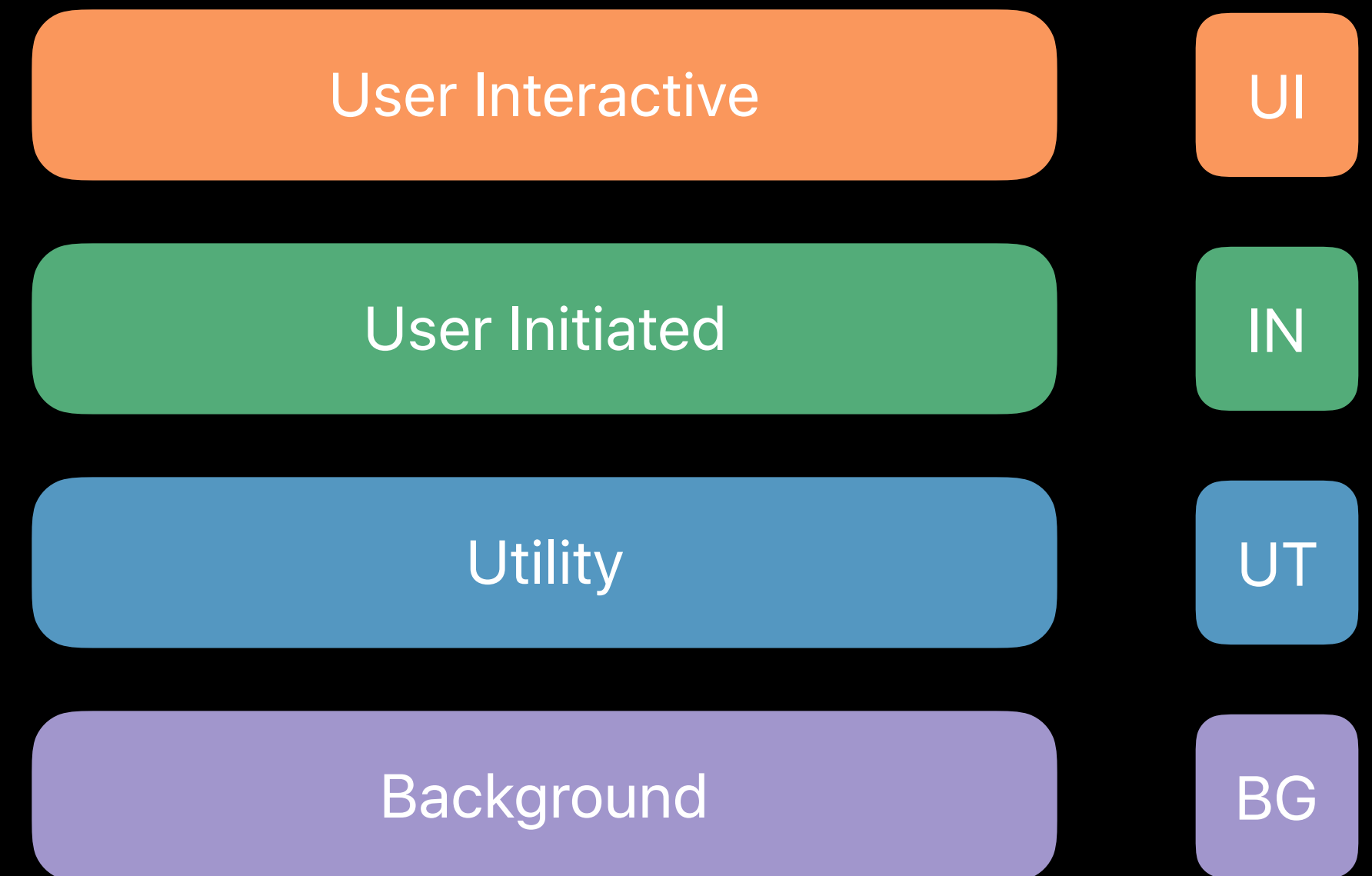


# Quality of Service

Abstract notion of priority

Provides explicit classification of your work

Affects various execution properties



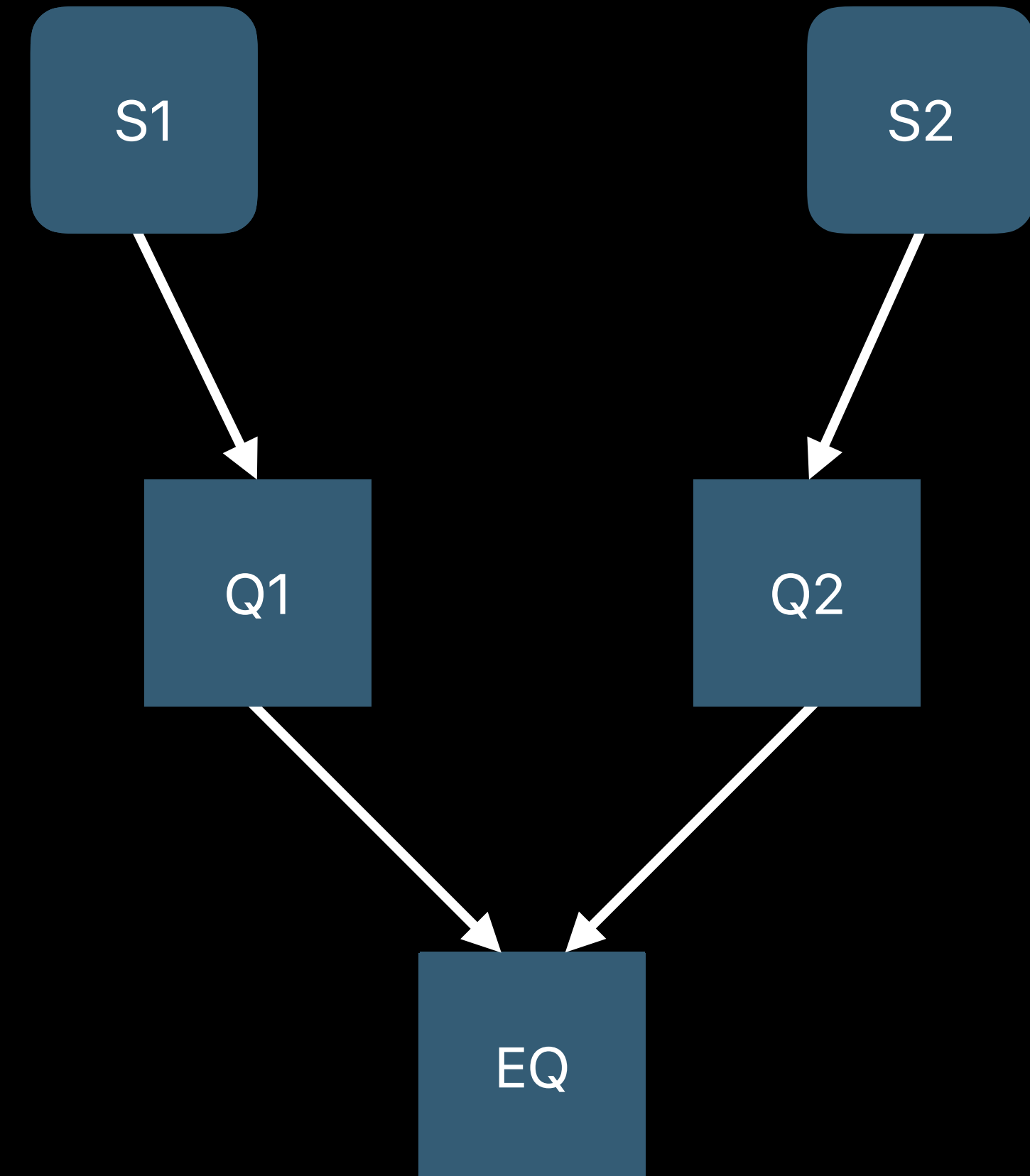
# QoS and Target Queue Hierarchy

UI

IN

UT

BG



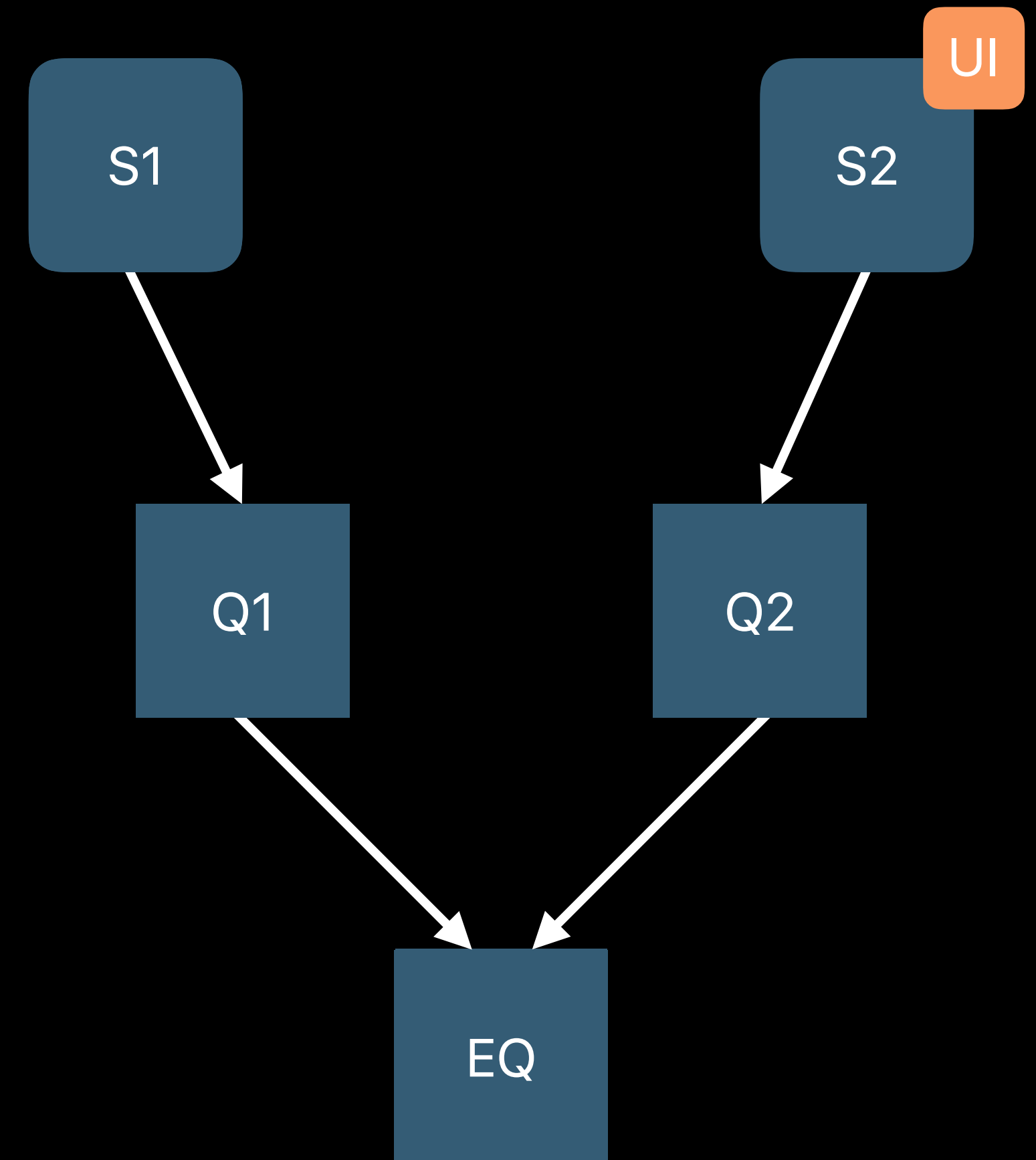
# QoS and Target Queue Hierarchy

UI

IN

UT

BG



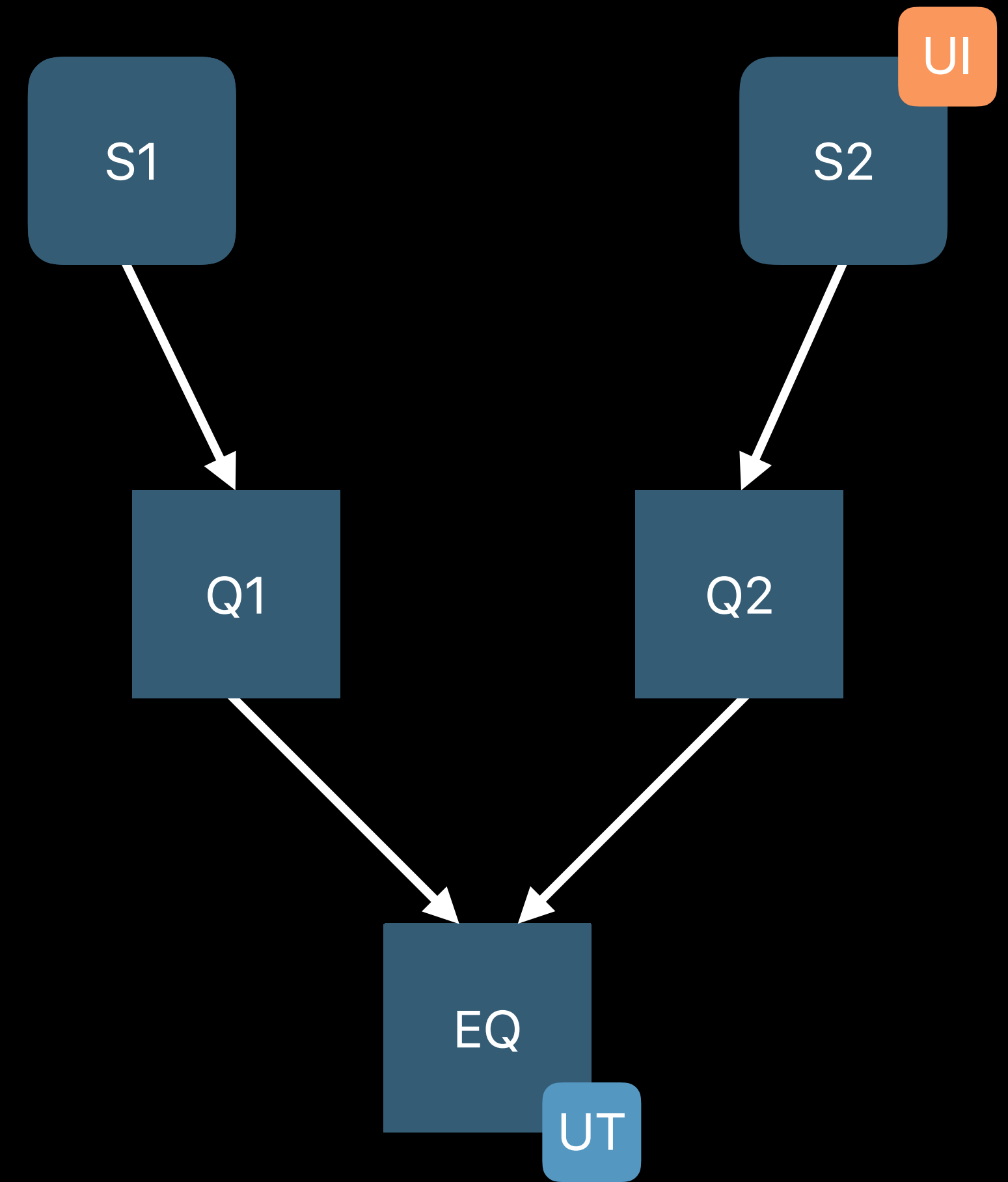
# QoS and Target Queue Hierarchy

UI

IN

UT

BG



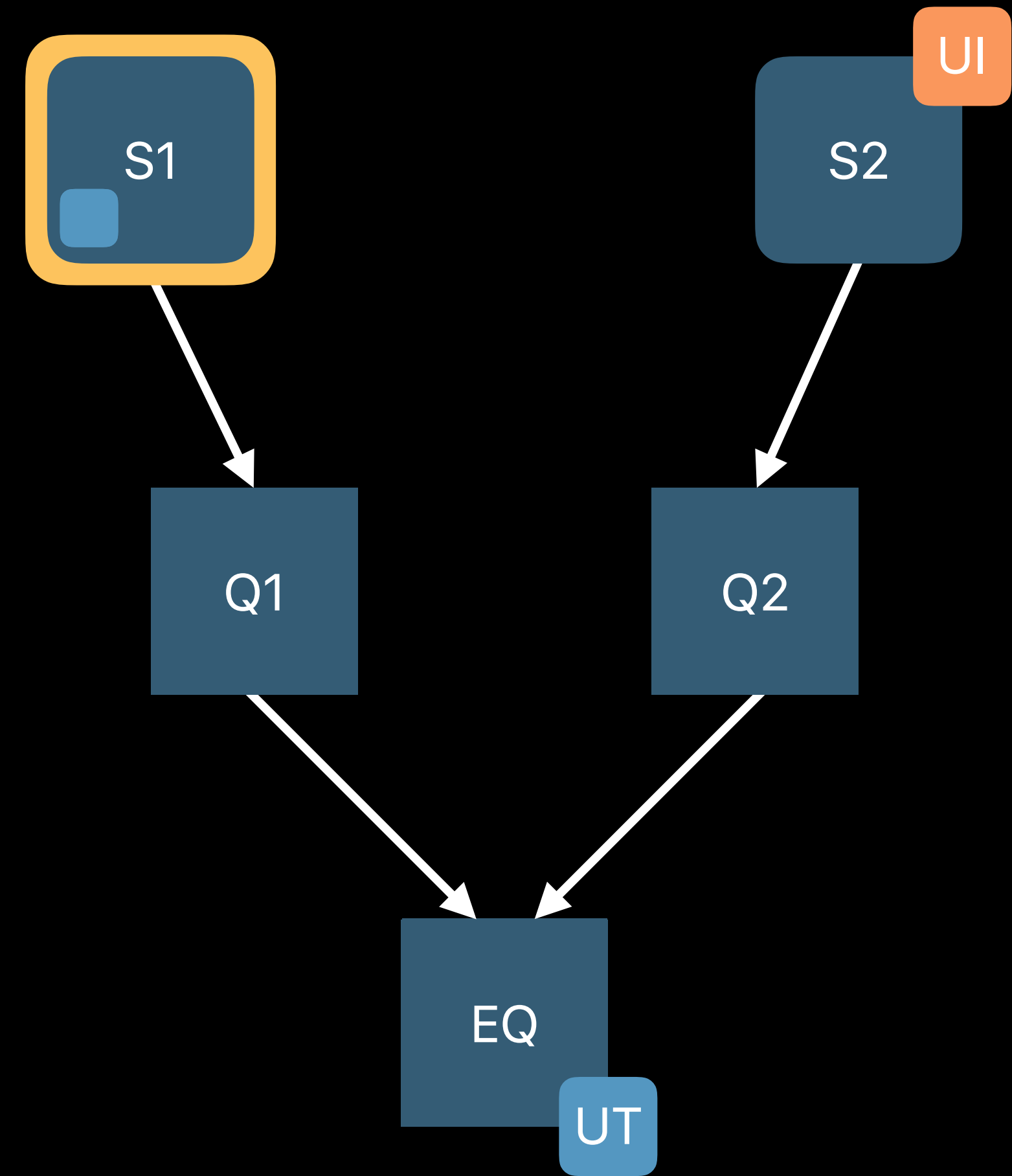
# QoS and Target Queue Hierarchy

UI

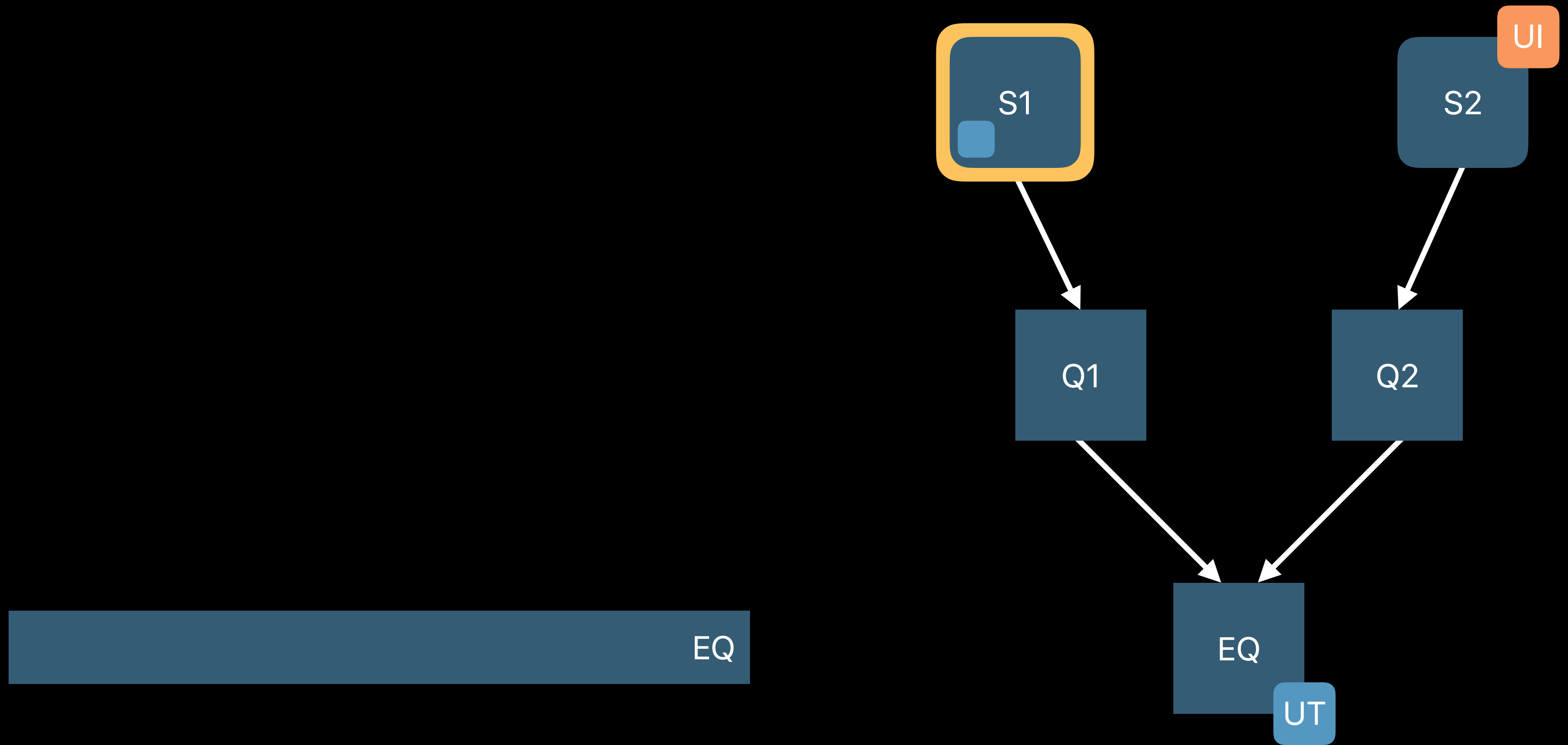
IN

UT

BG

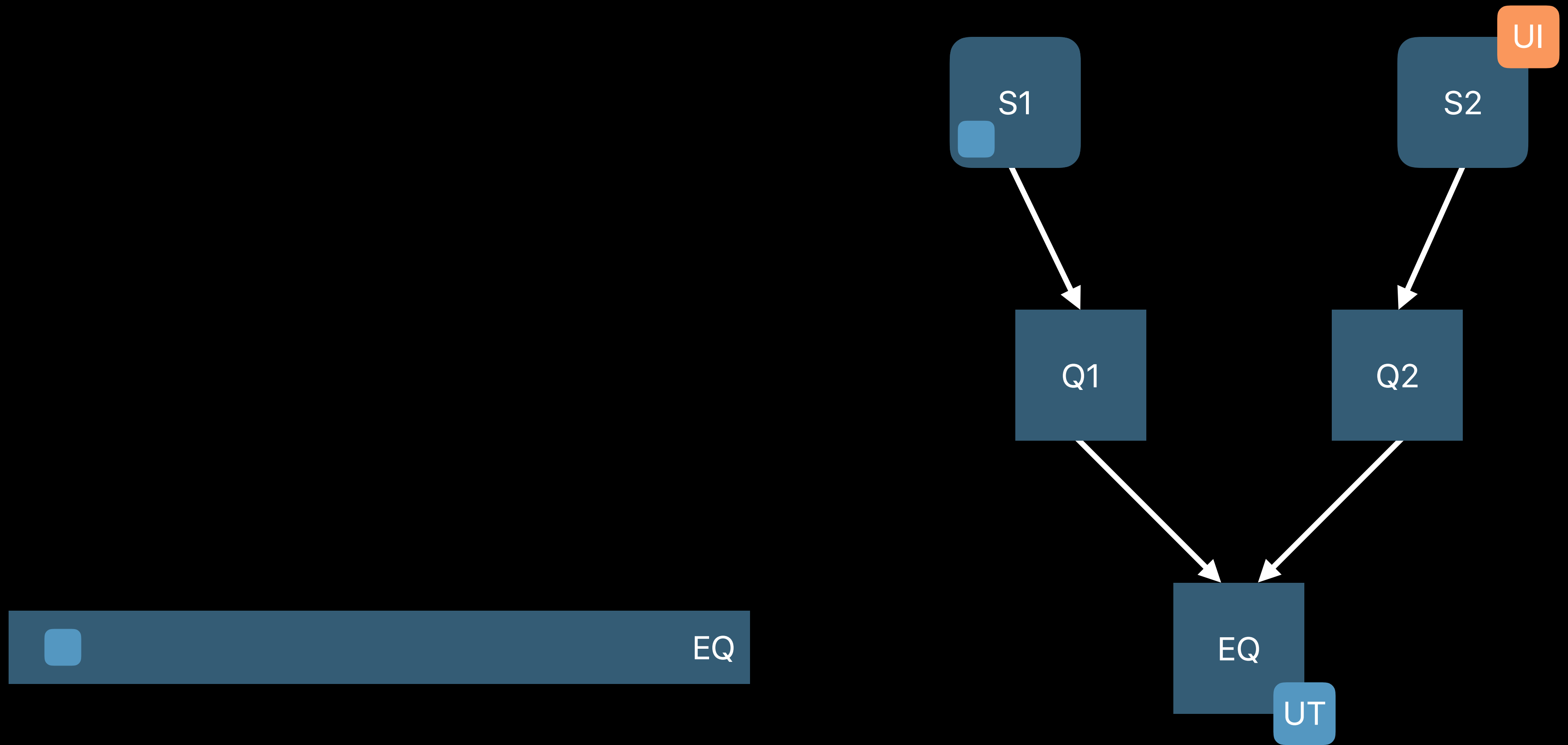


# QoS and Target Queue Hierarchy

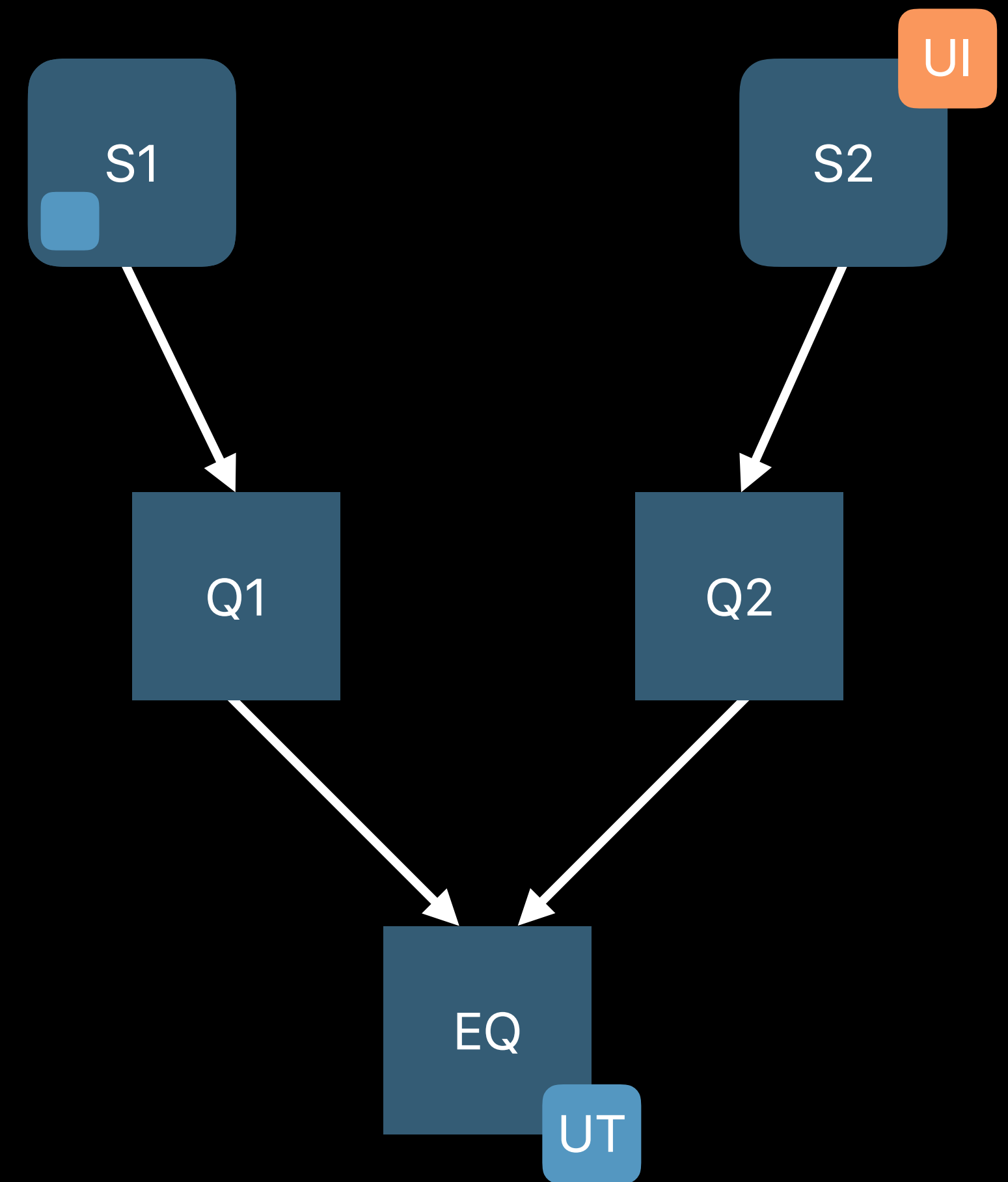
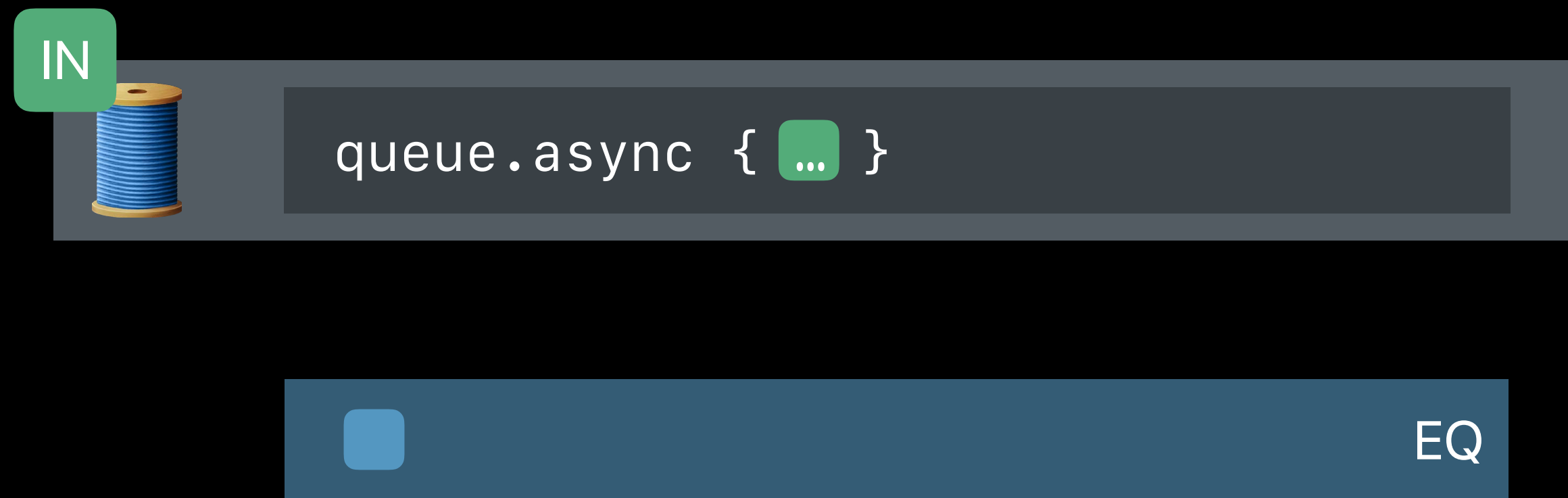




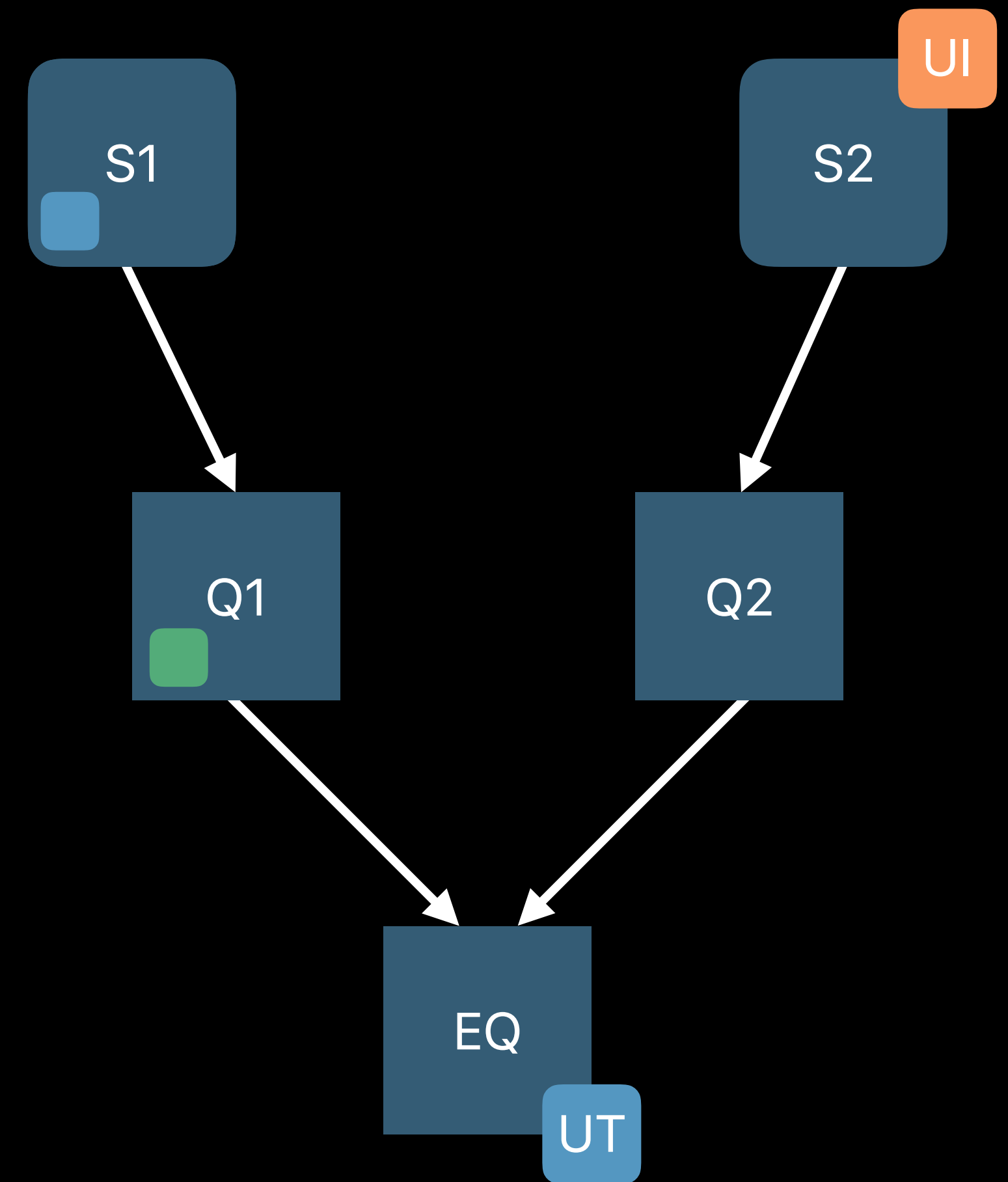
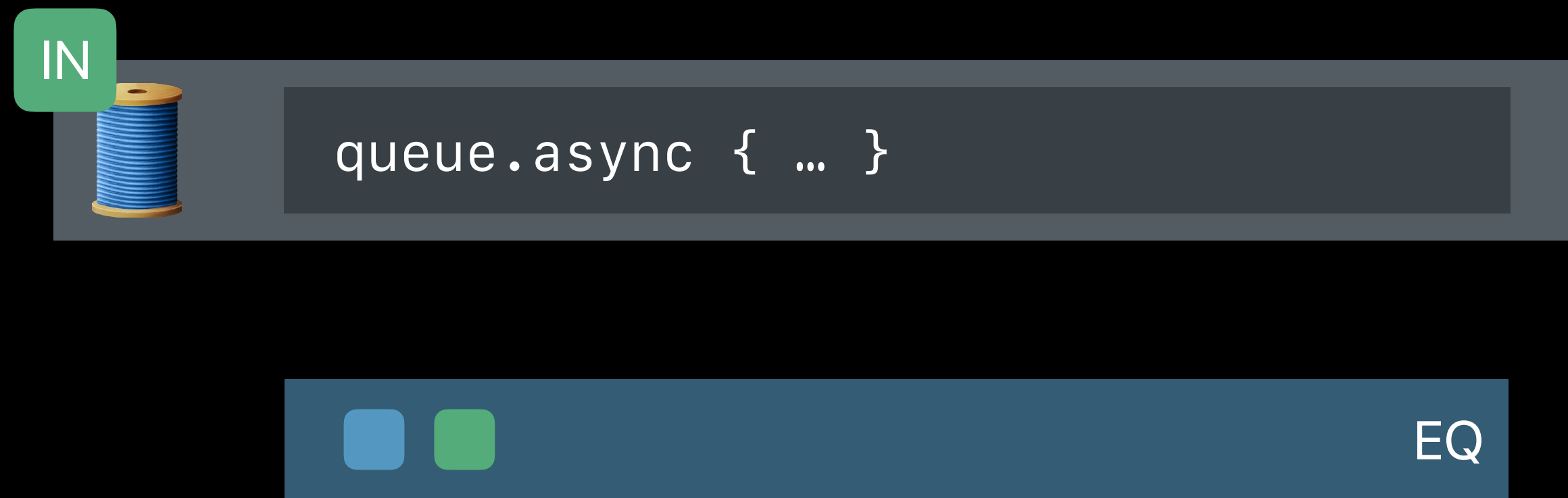
# QoS and Target Queue Hierarchy



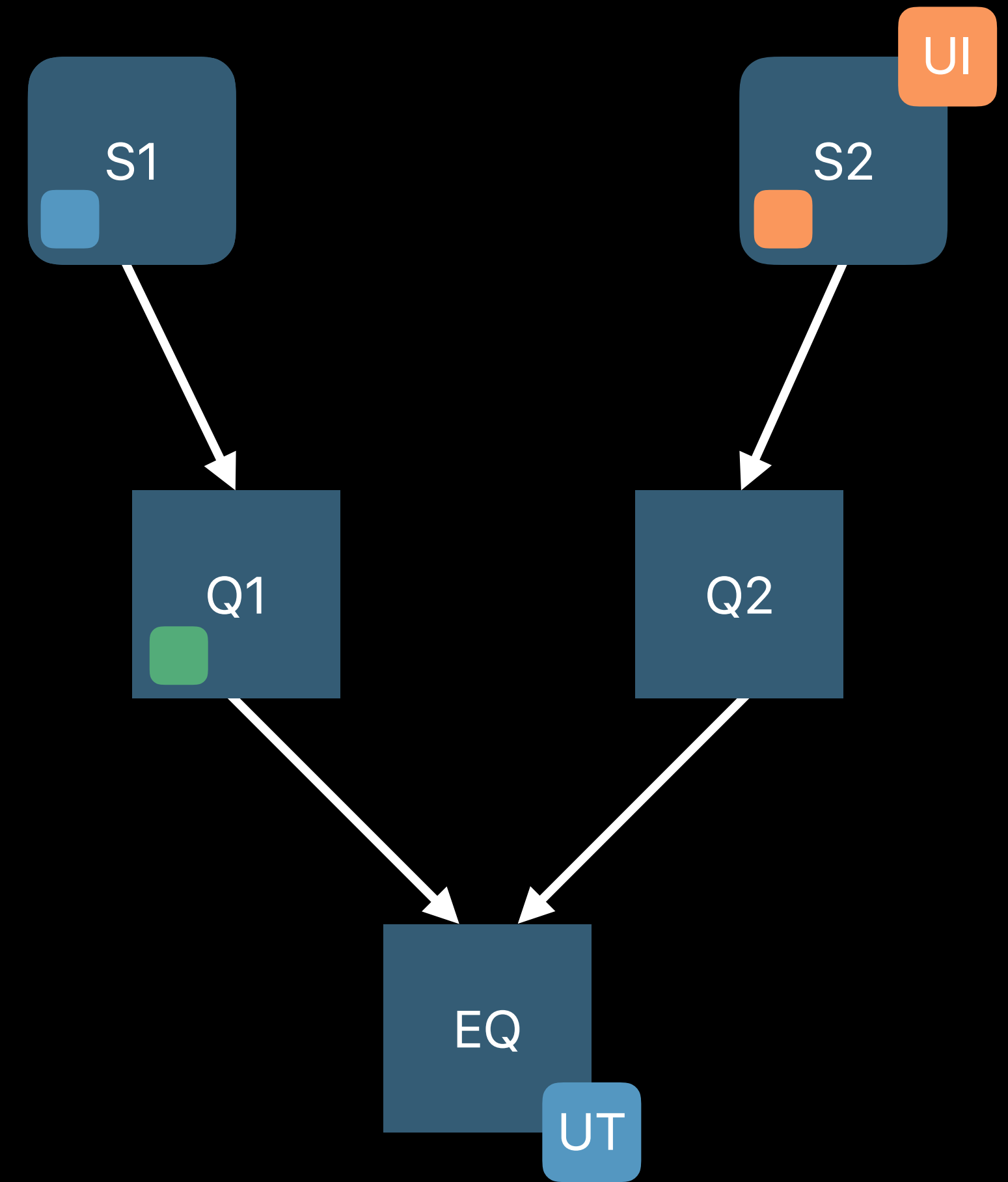
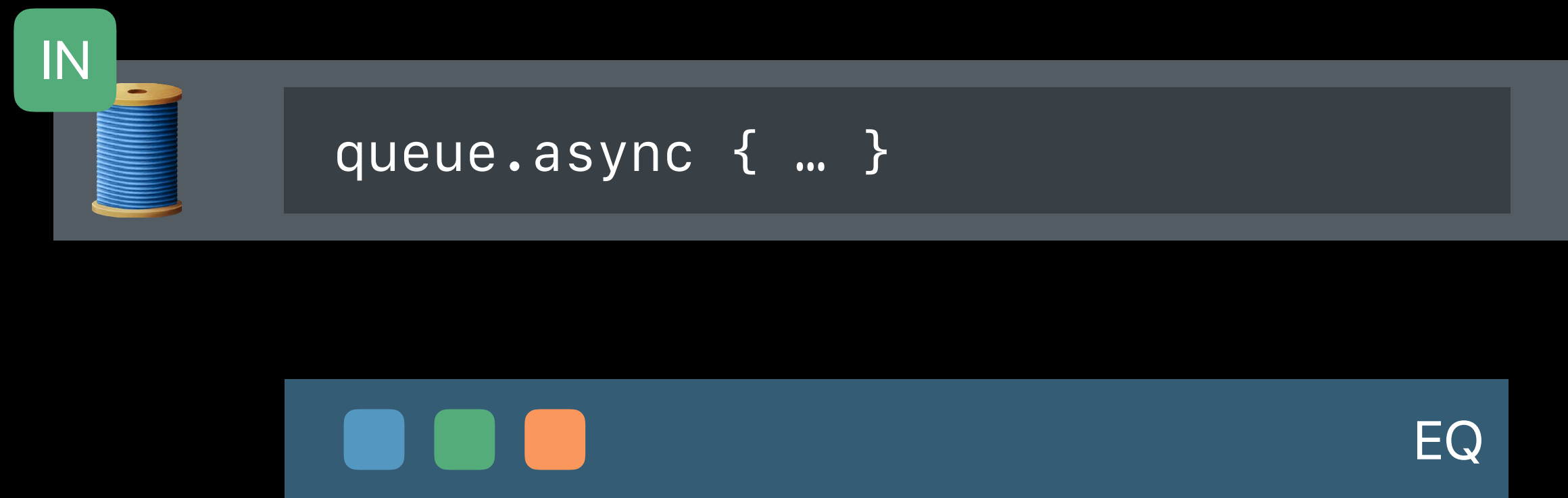
# QoS and Target Queue Hierarchy



# QoS and Target Queue Hierarchy

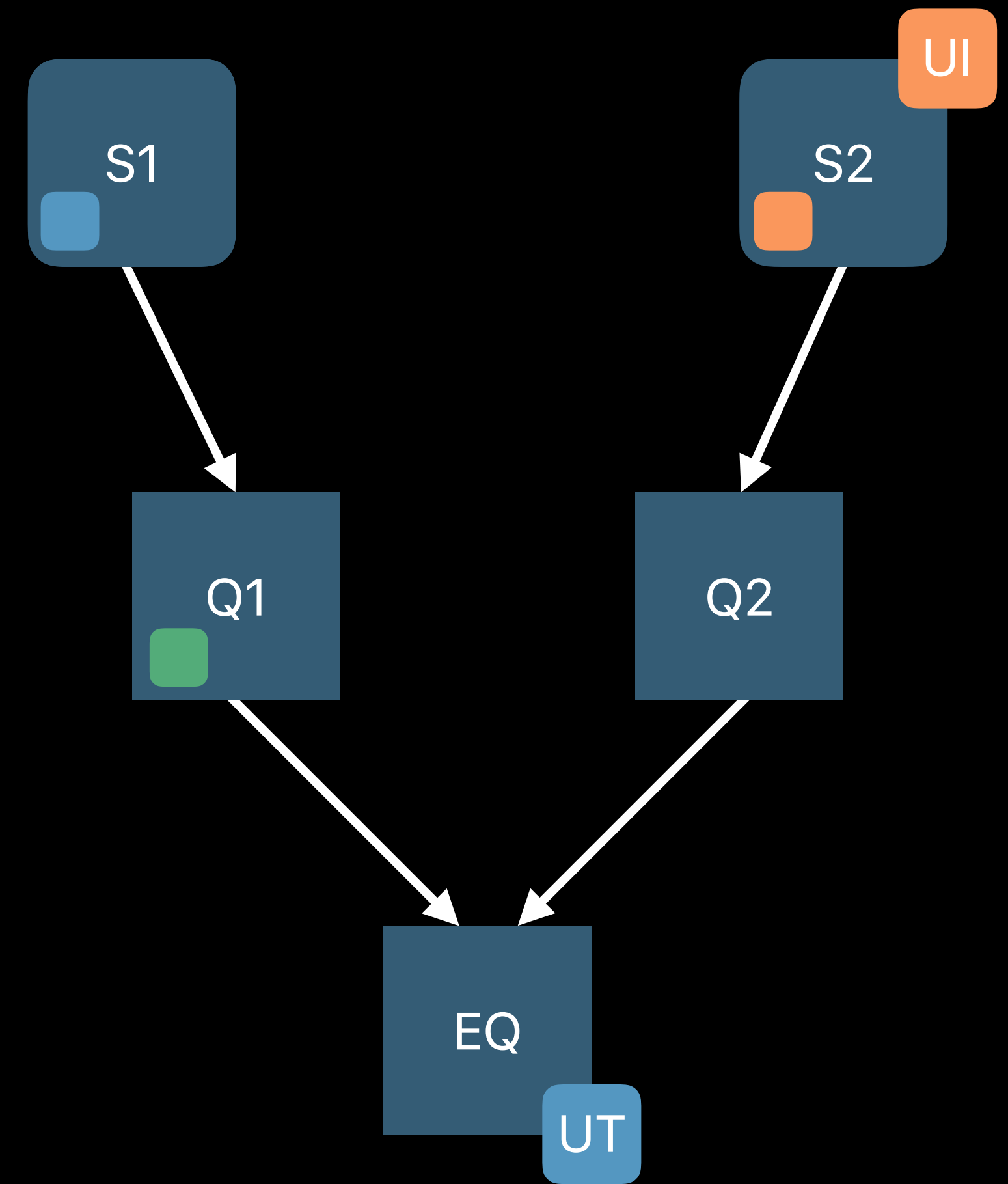


# QoS and Target Queue Hierarchy

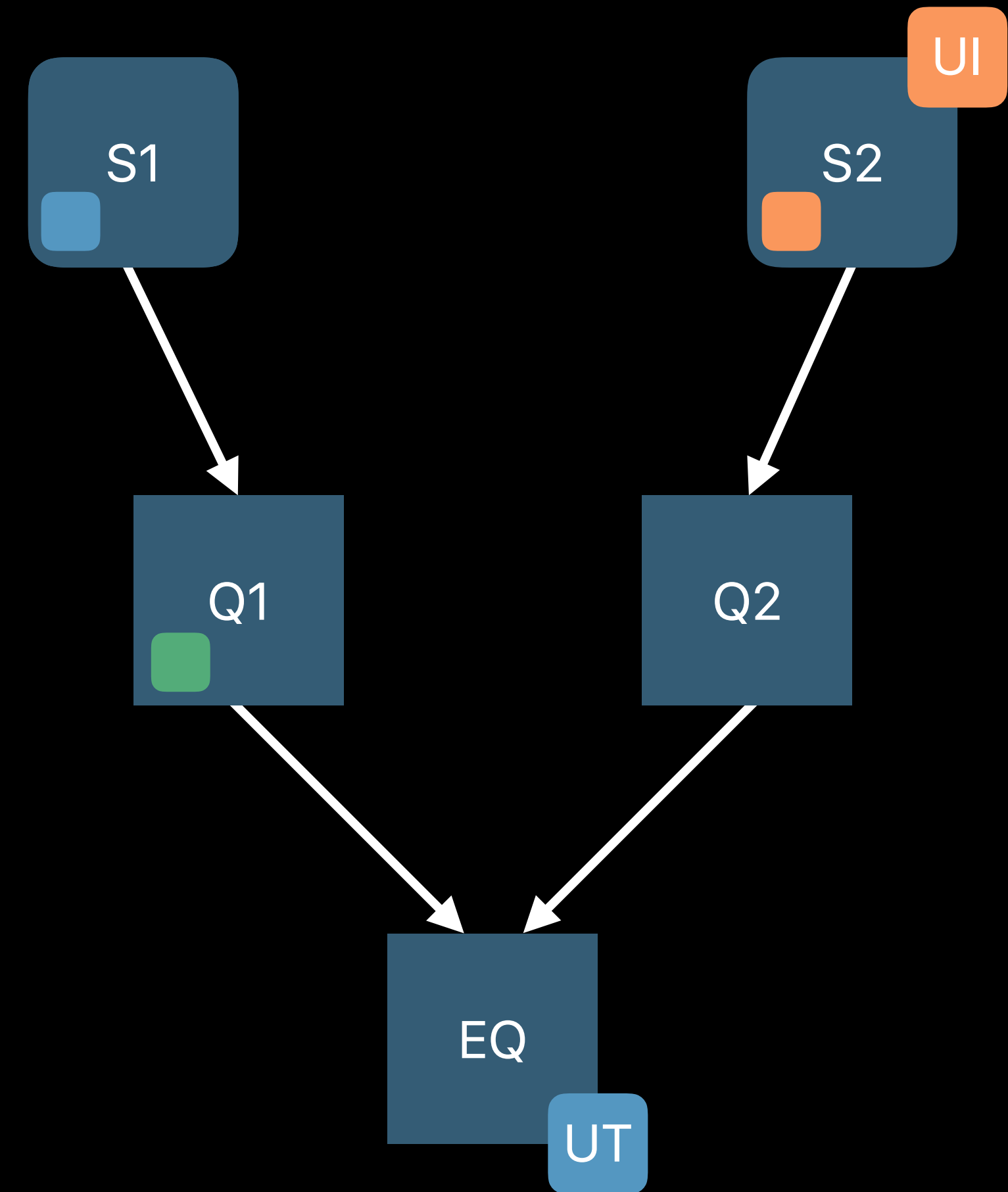
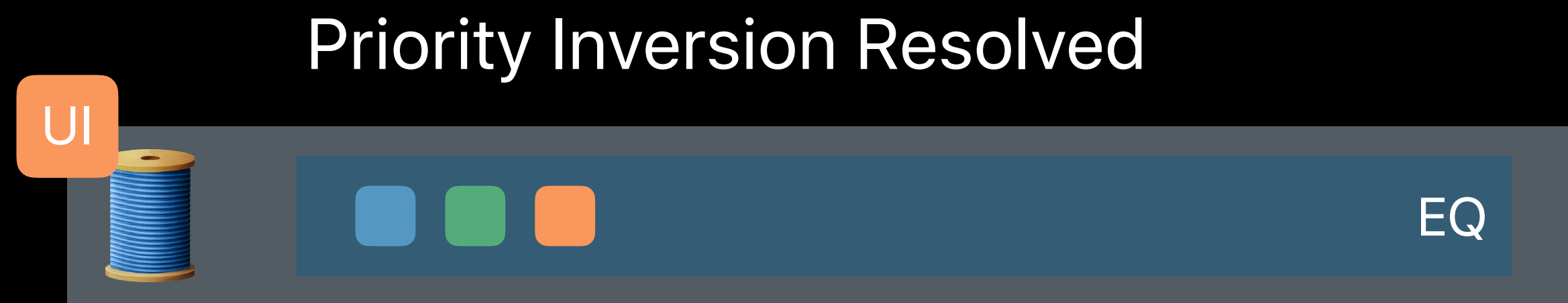


# QoS and Target Queue Hierarchy

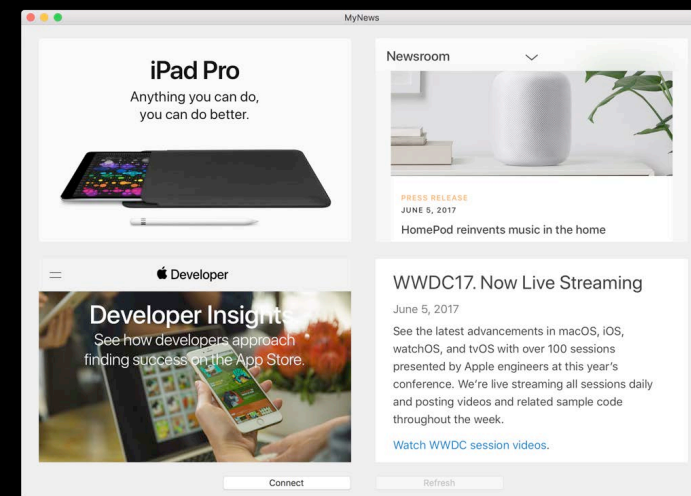
Priority Inversion



# QoS and Target Queue Hierarchy



# Granularity of Concurrency

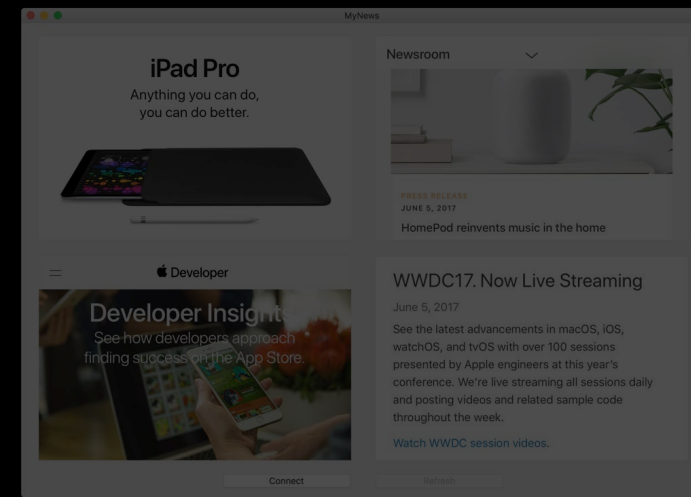


User Interface

Networking

Database





# Networking

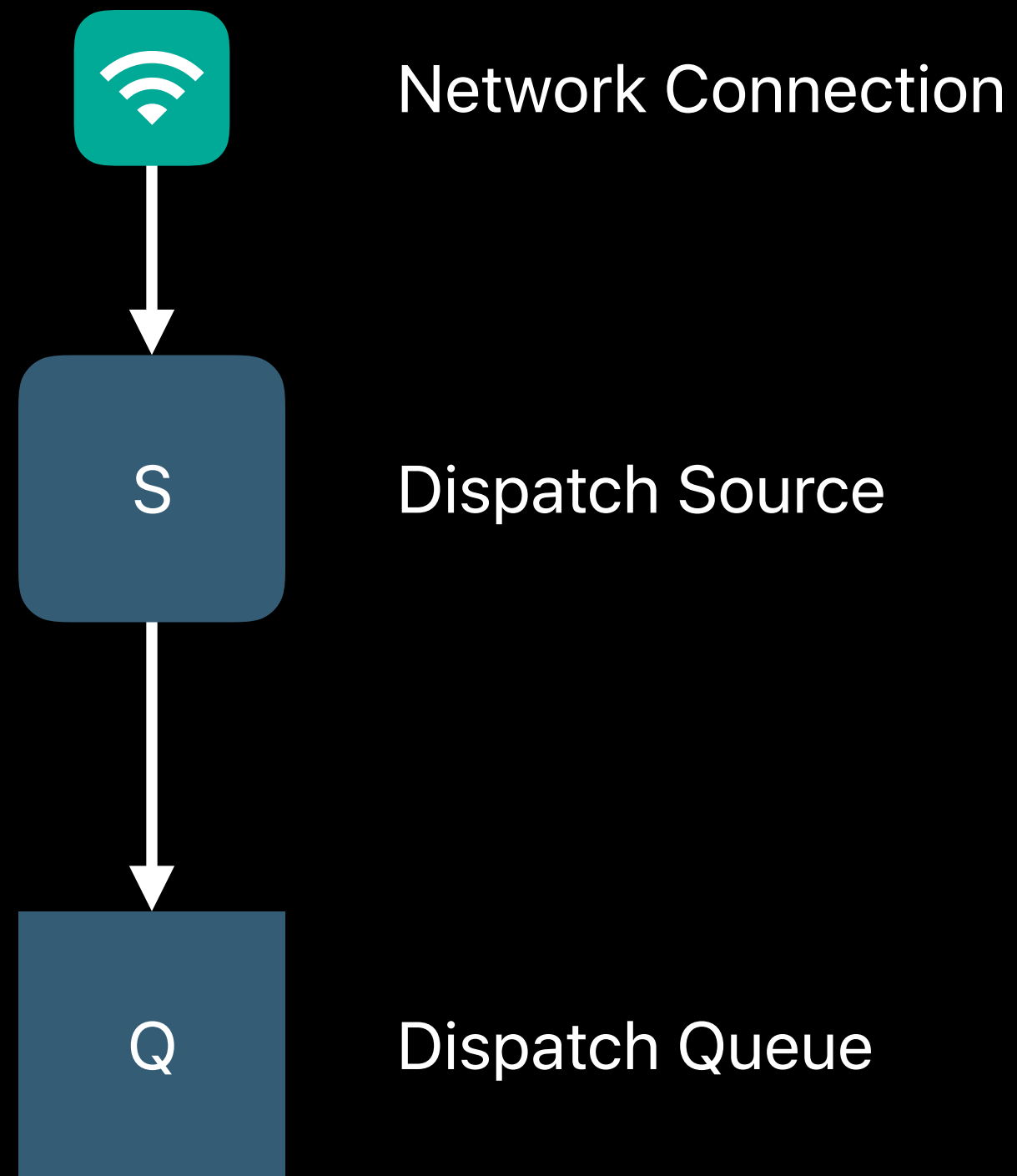
# Event Monitoring Setup

# Event Monitoring Setup

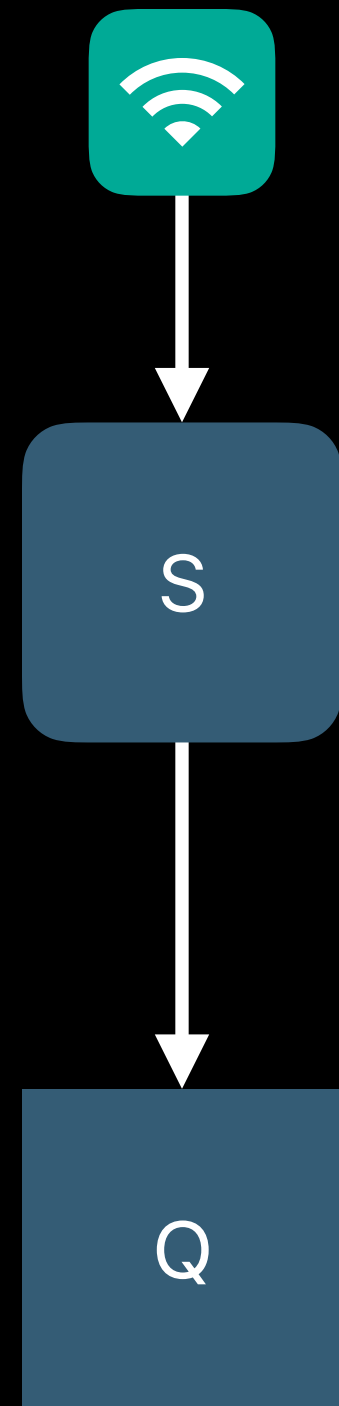


Network Connection

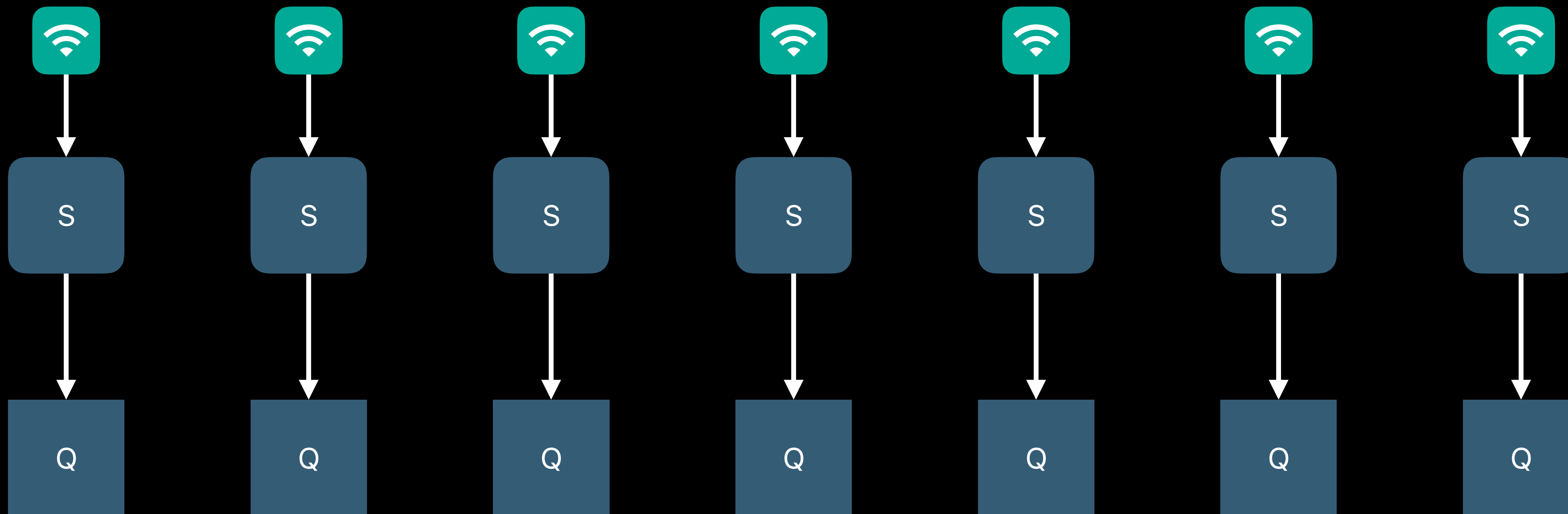
# Event Monitoring Setup



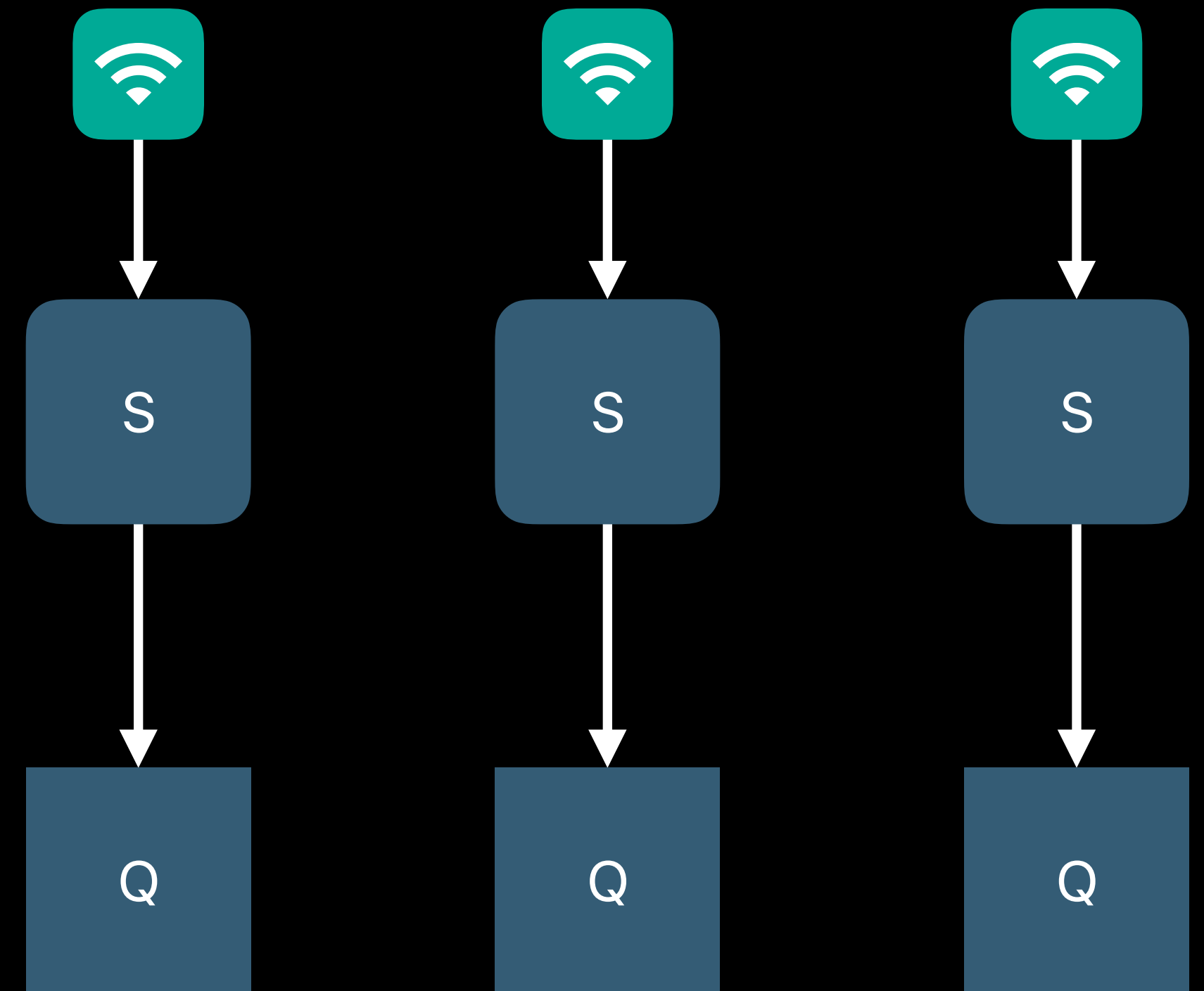
# Event Monitoring Setup



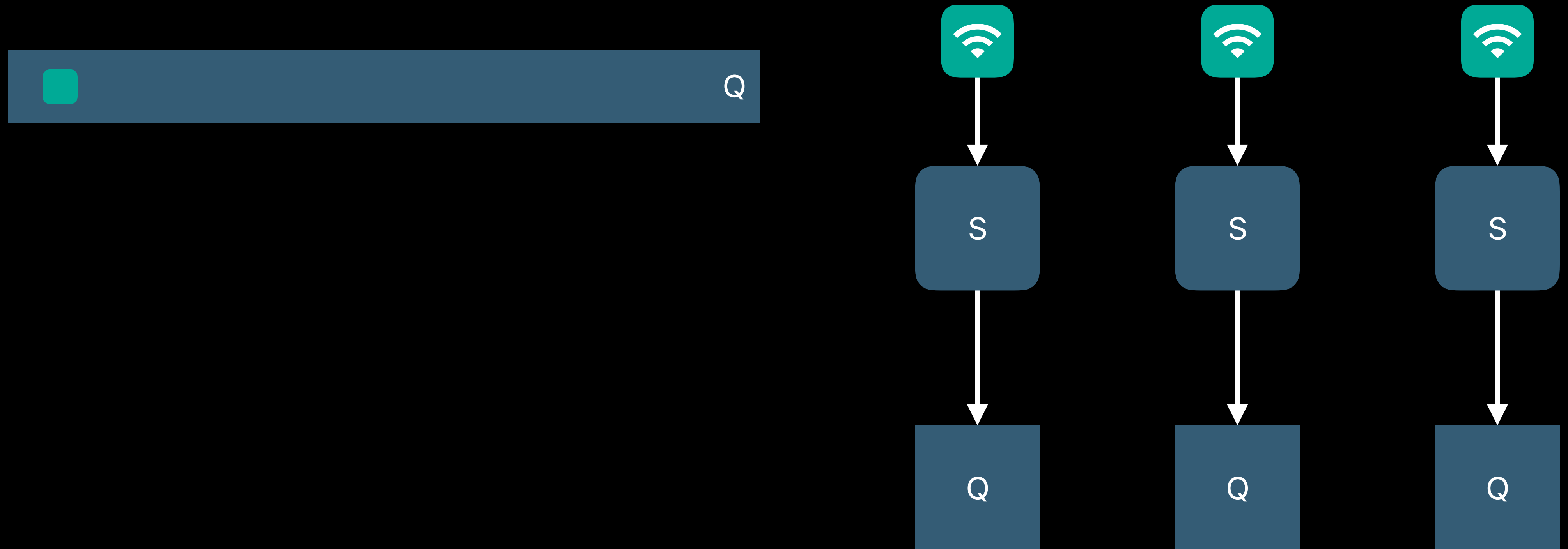
# Event Monitoring Setup



# Event Handling on Many Independent Queues

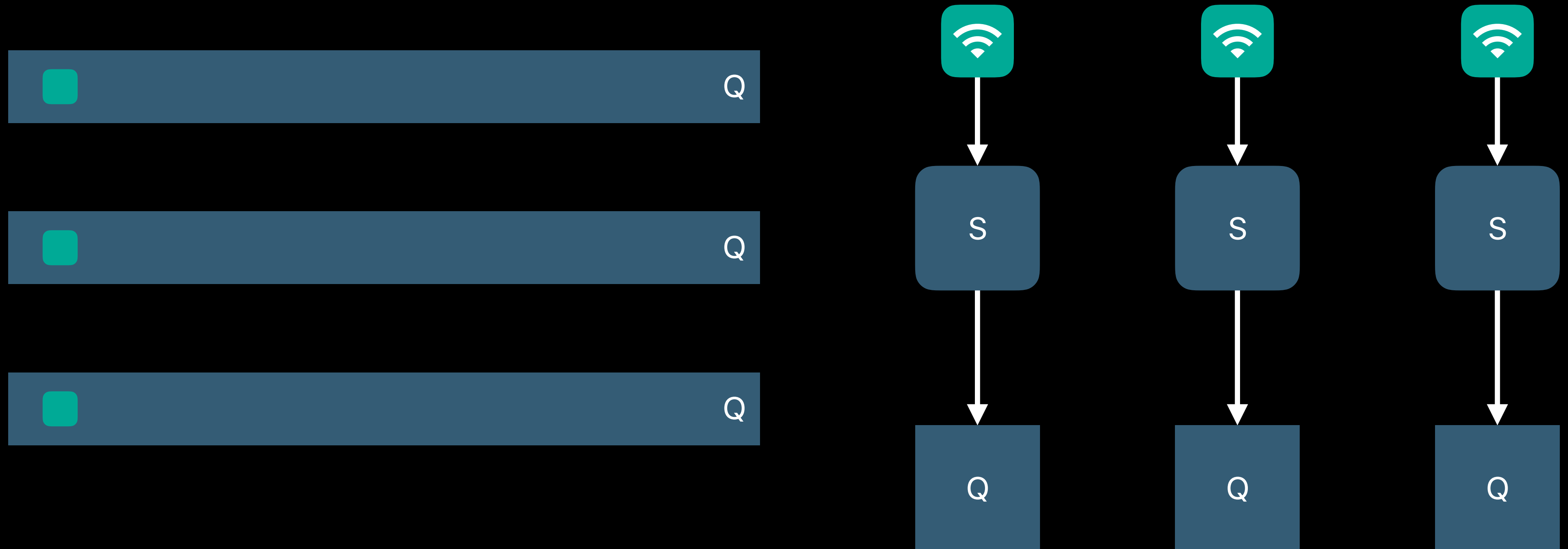


# Event Handling on Many Independent Queues

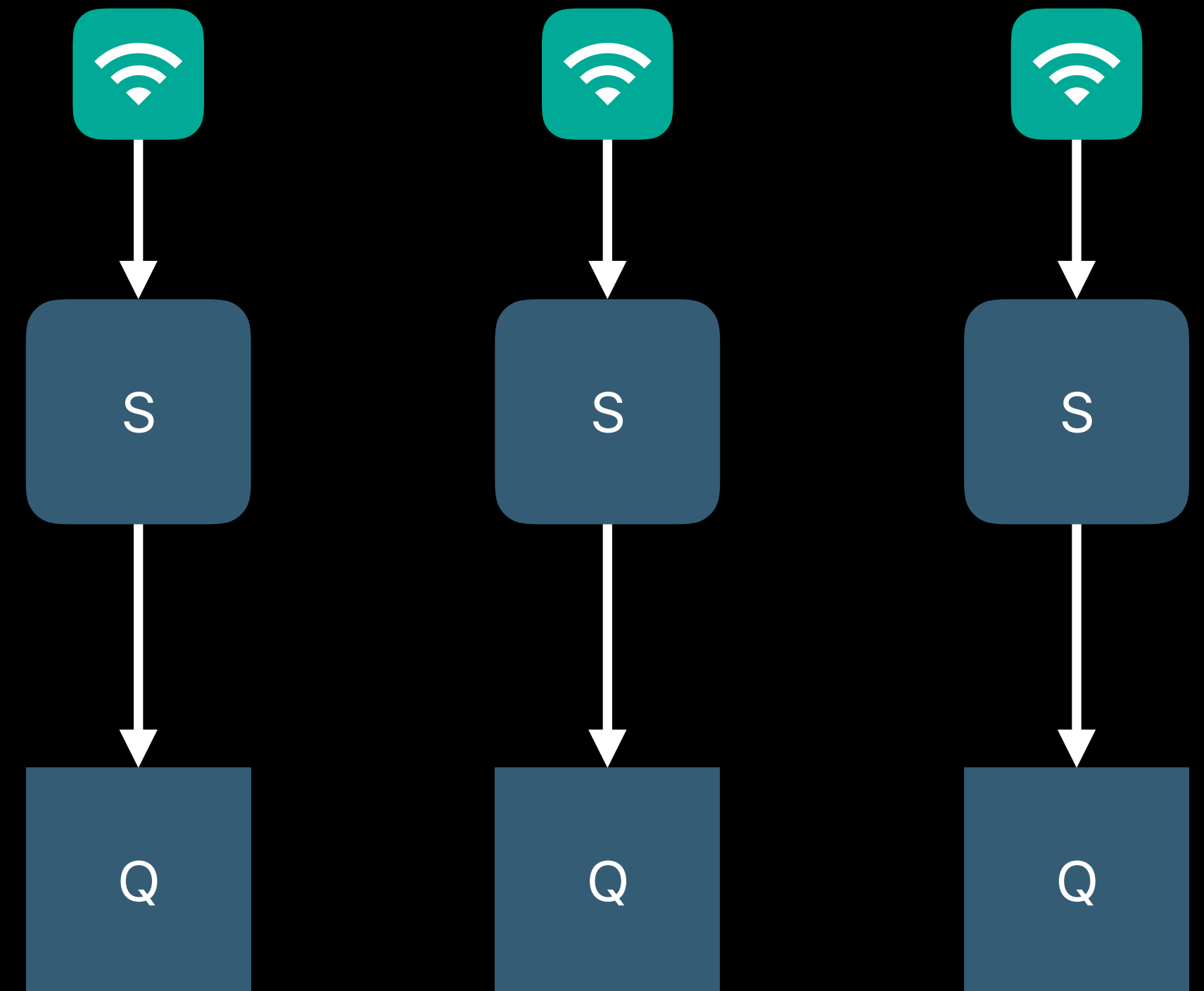
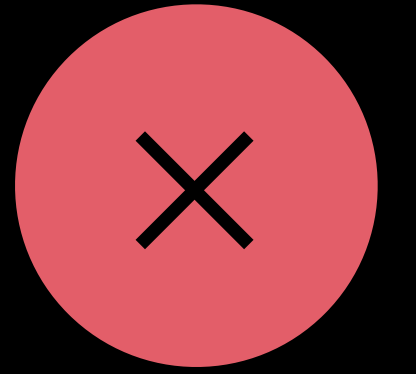




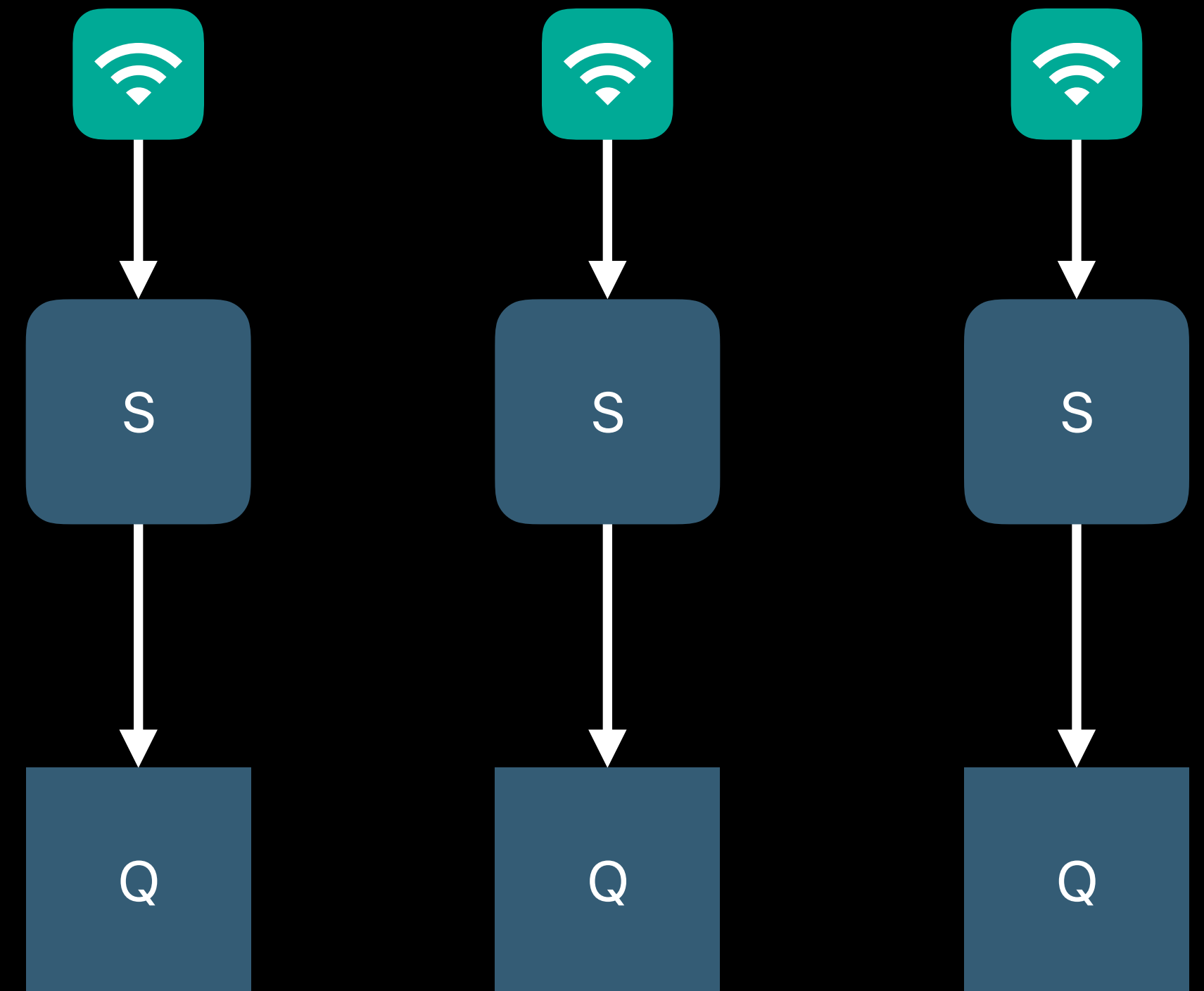
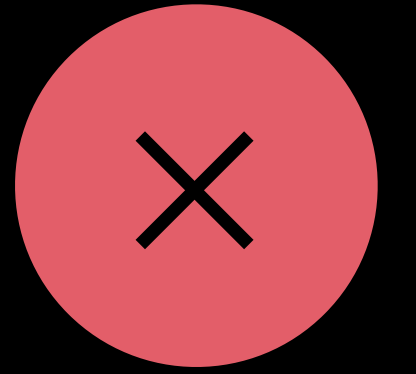
# Event Handling on Many Independent Queues



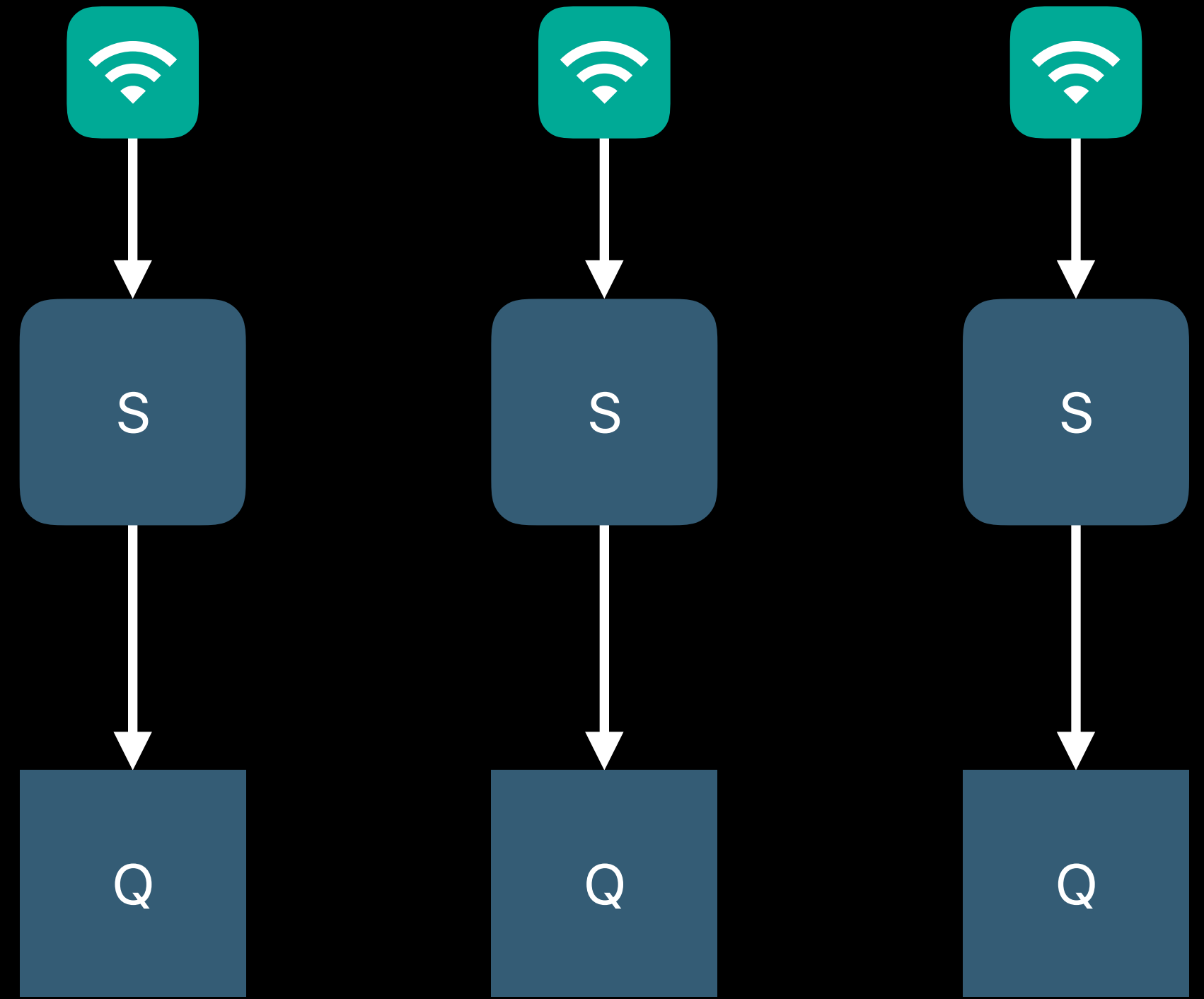
# Event Handling on Many Independent Queues



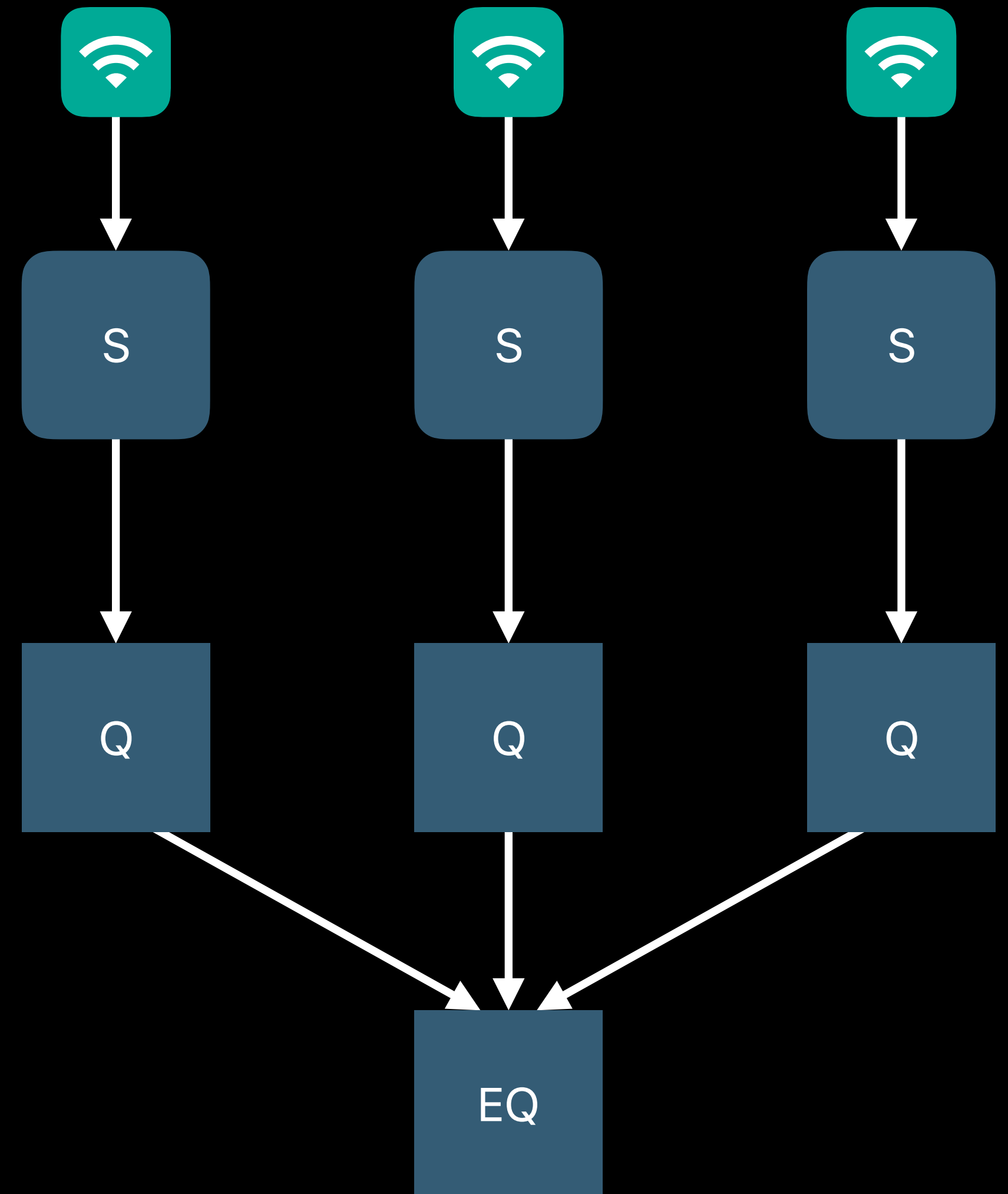
# Event Handling on Many Independent Queues



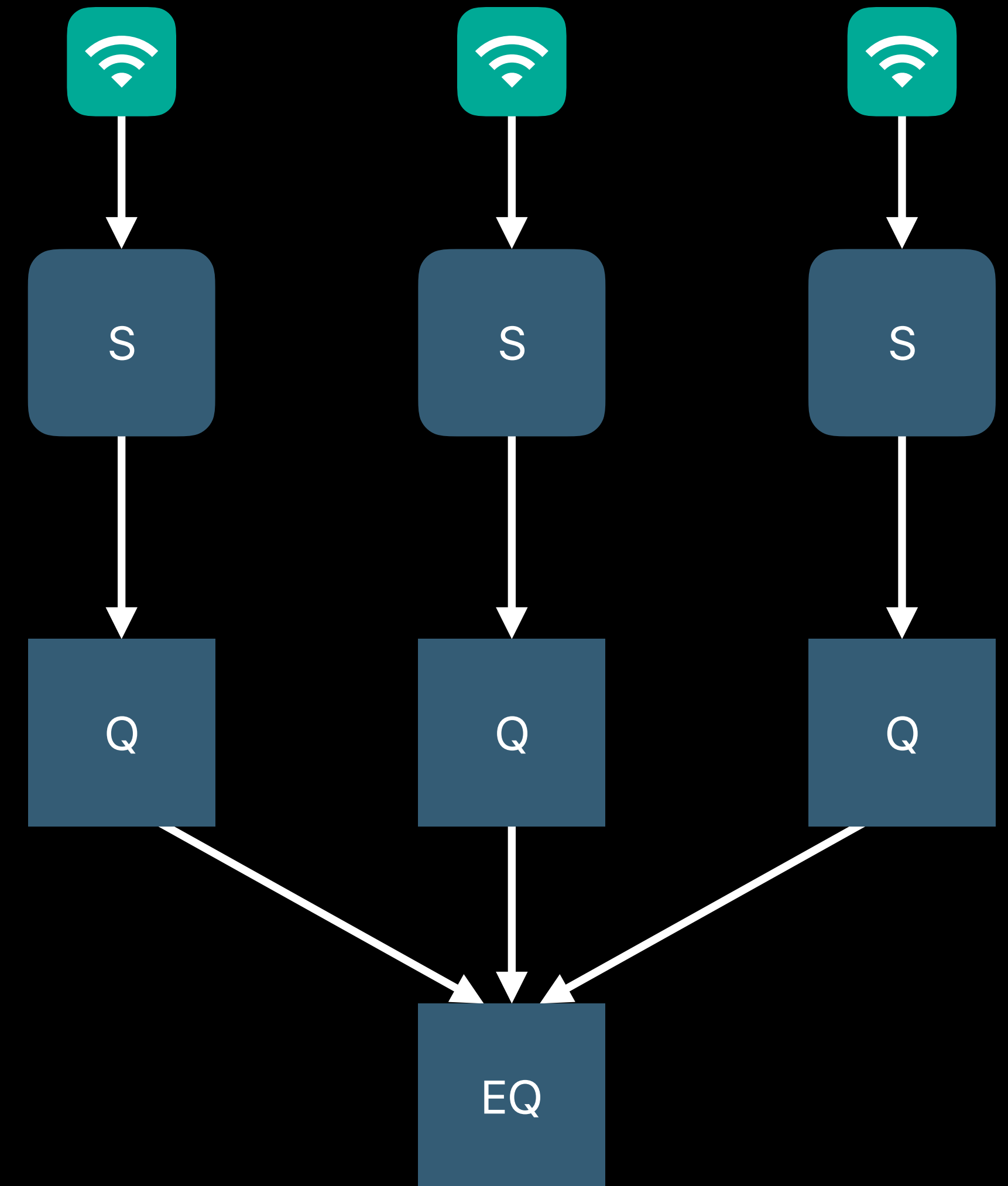
# Single Mutual Exclusion Context



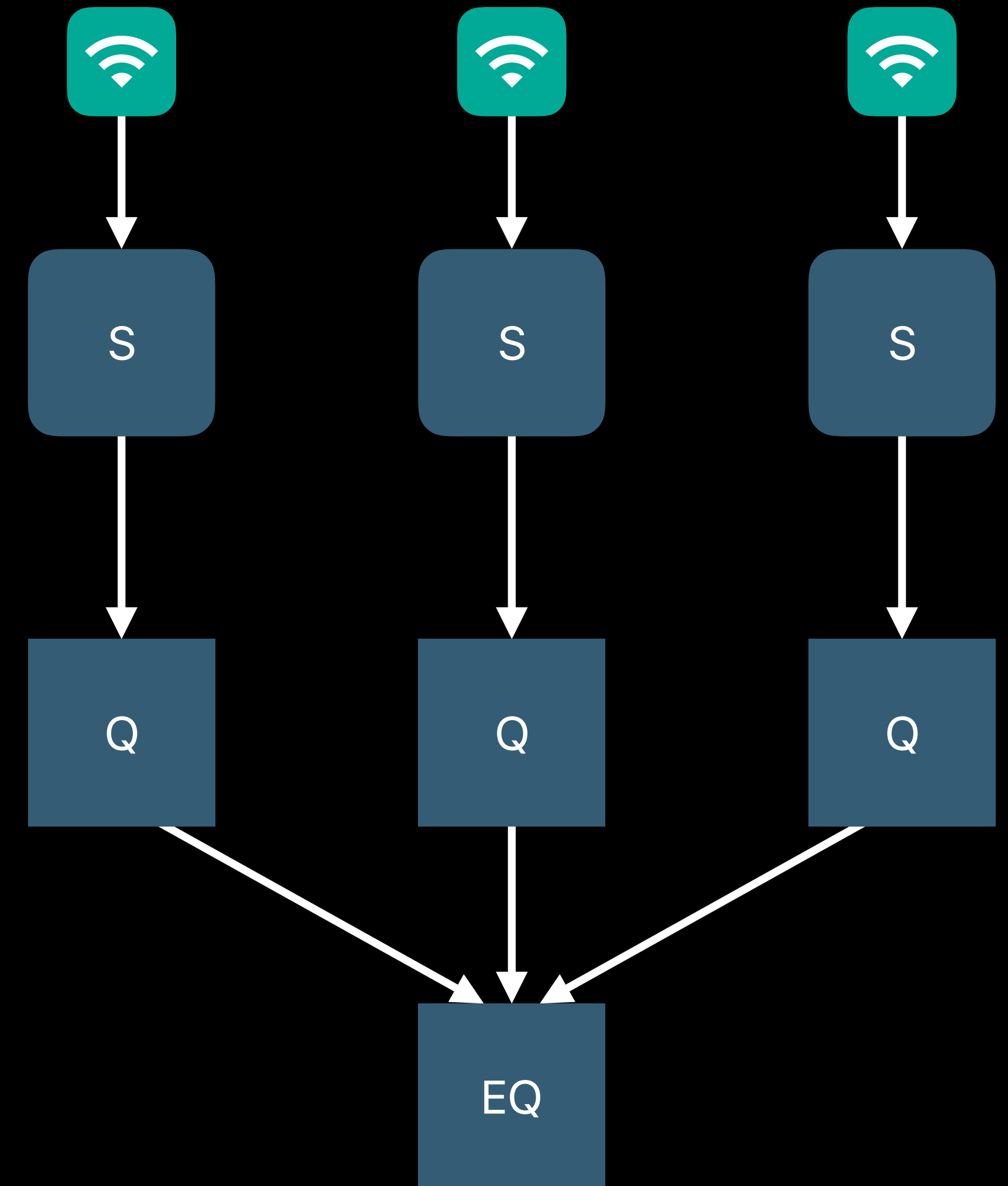
# Single Mutual Exclusion Context



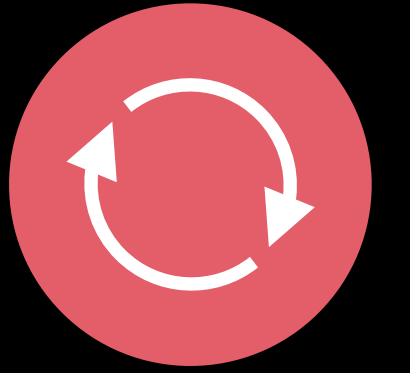
# Single Mutual Exclusion Context



# Single Mutual Exclusion Context



# Too Much of a Good Thing



Repeatedly waiting for exclusive access to contended resources

Repeatedly switching between independent operations

Repeatedly bouncing an operation between threads

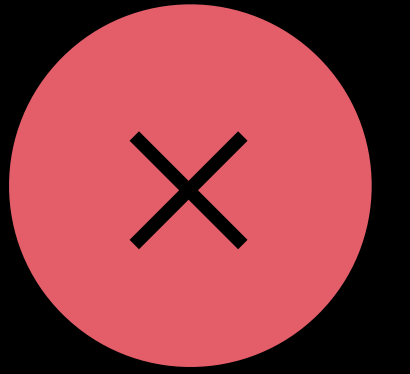


# Avoid Unbounded Concurrency

Repeatedly switching between independent operations

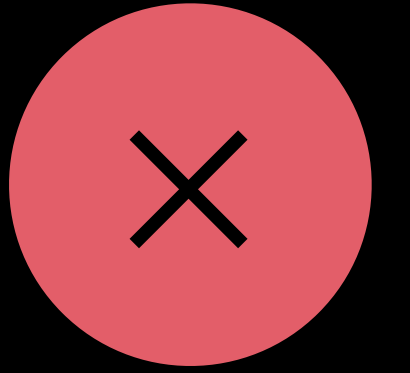
# Avoid Unbounded Concurrency

Repeatedly switching between independent operations



# Avoid Unbounded Concurrency

Repeatedly switching between independent operations

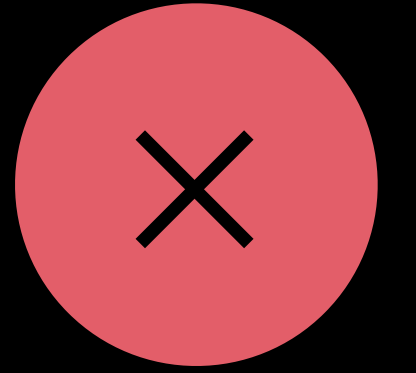


Many queues becoming active at once

- Independent per-client sources
- Independent per-object queues

# Avoid Unbounded Concurrency

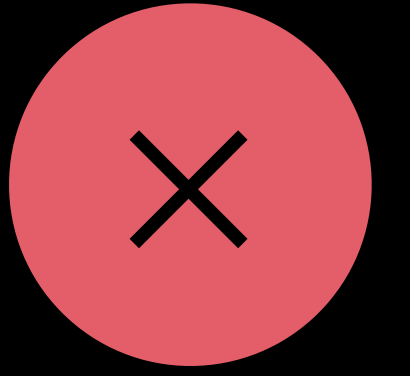
Repeatedly switching between independent operations



Many workitems submitted to global concurrent queue

# Avoid Unbounded Concurrency

Repeatedly switching between independent operations

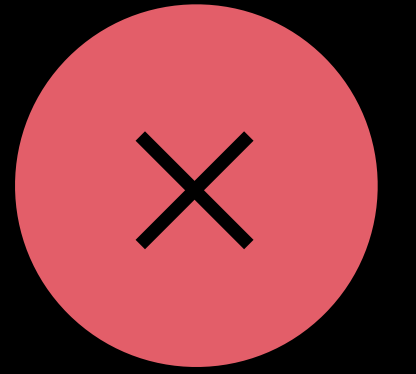


Many workitems submitted to global concurrent queue

- If workitems block, more threads will be created
- May lead to thread explosion

# Avoid Unbounded Concurrency

Repeatedly switching between independent operations

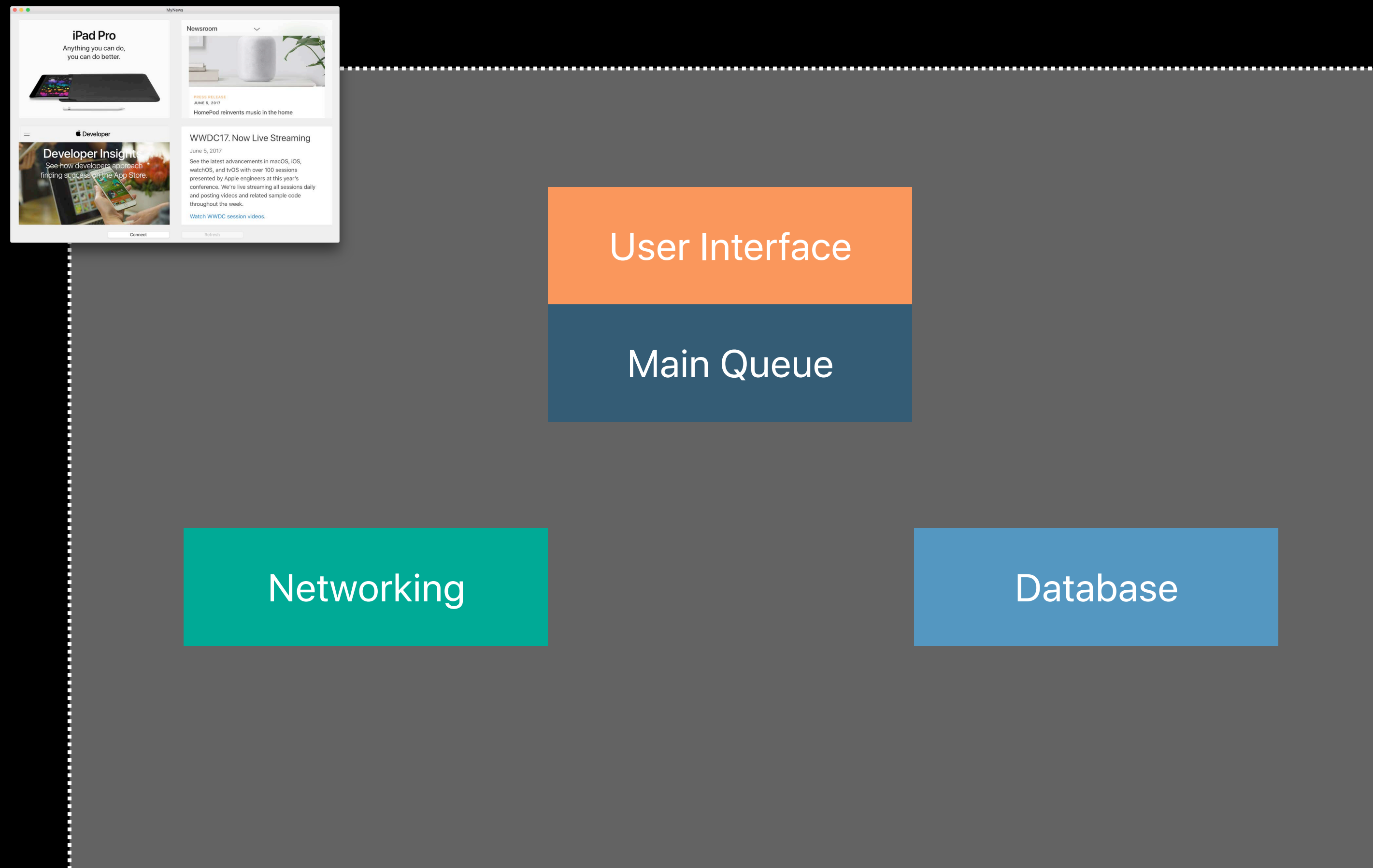


Many workitems submitted to global concurrent queue

- If workitems block, more threads will be created
- May lead to thread explosion

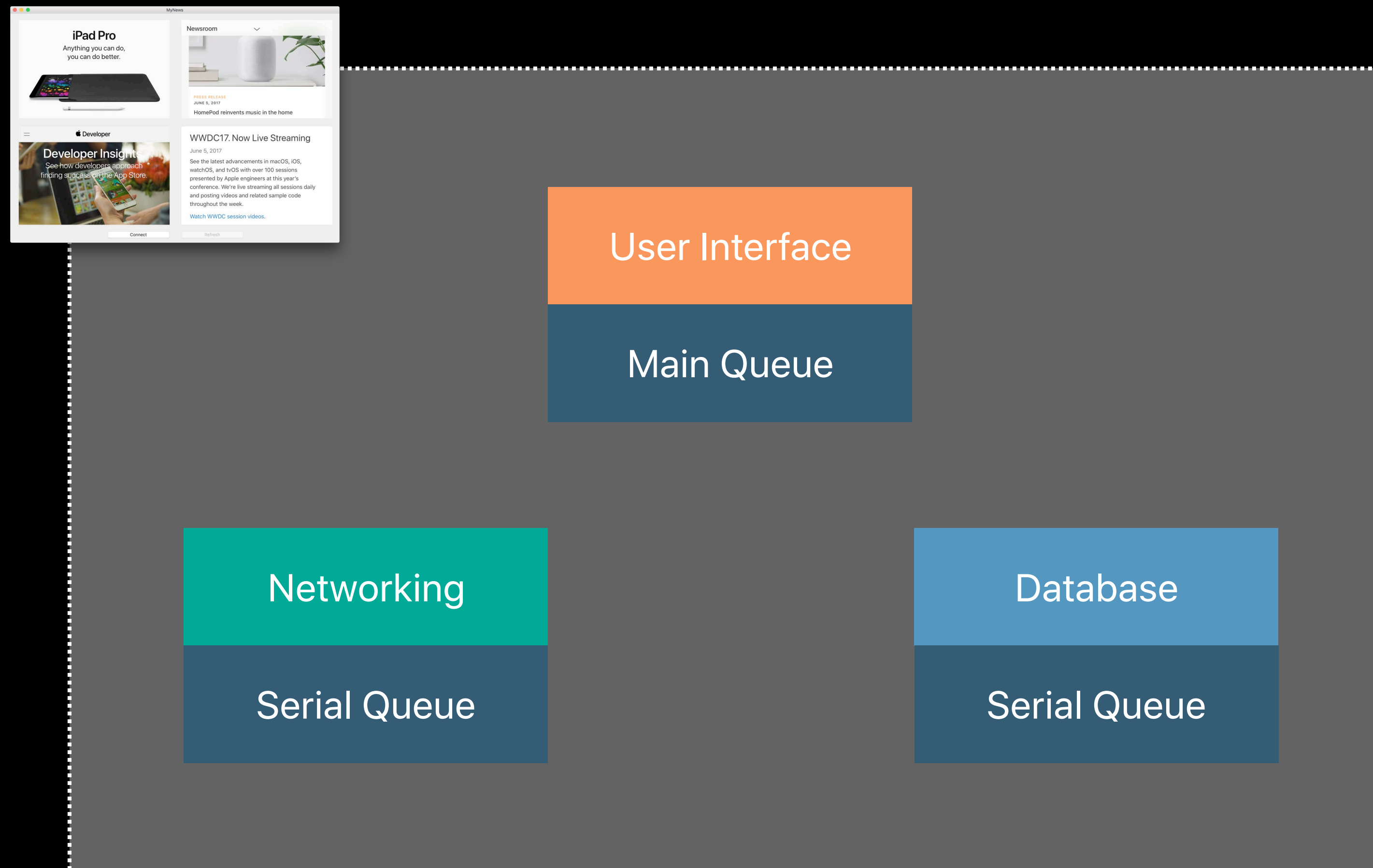
# One Queue per Subsystem

# One Queue per Subsystem

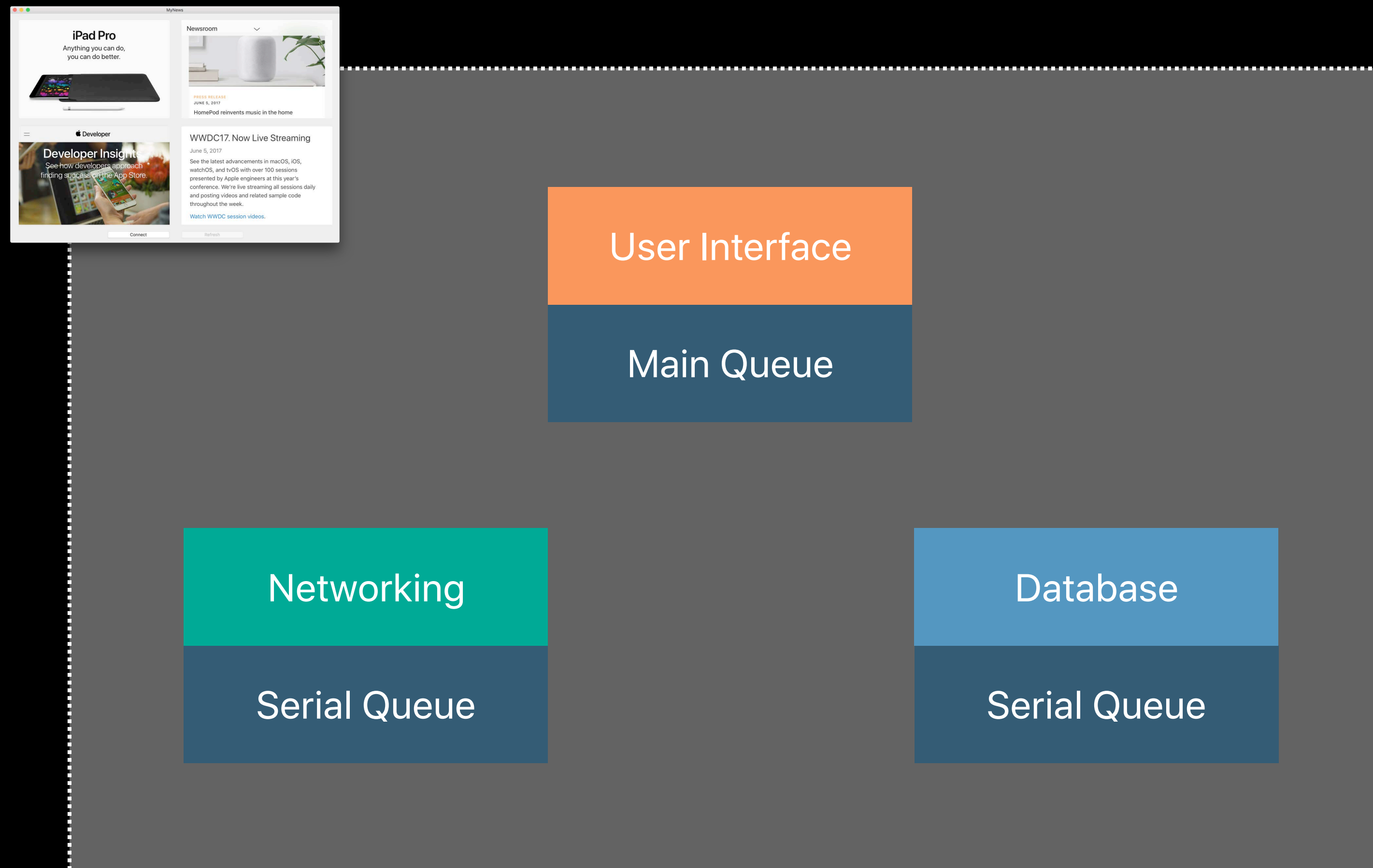




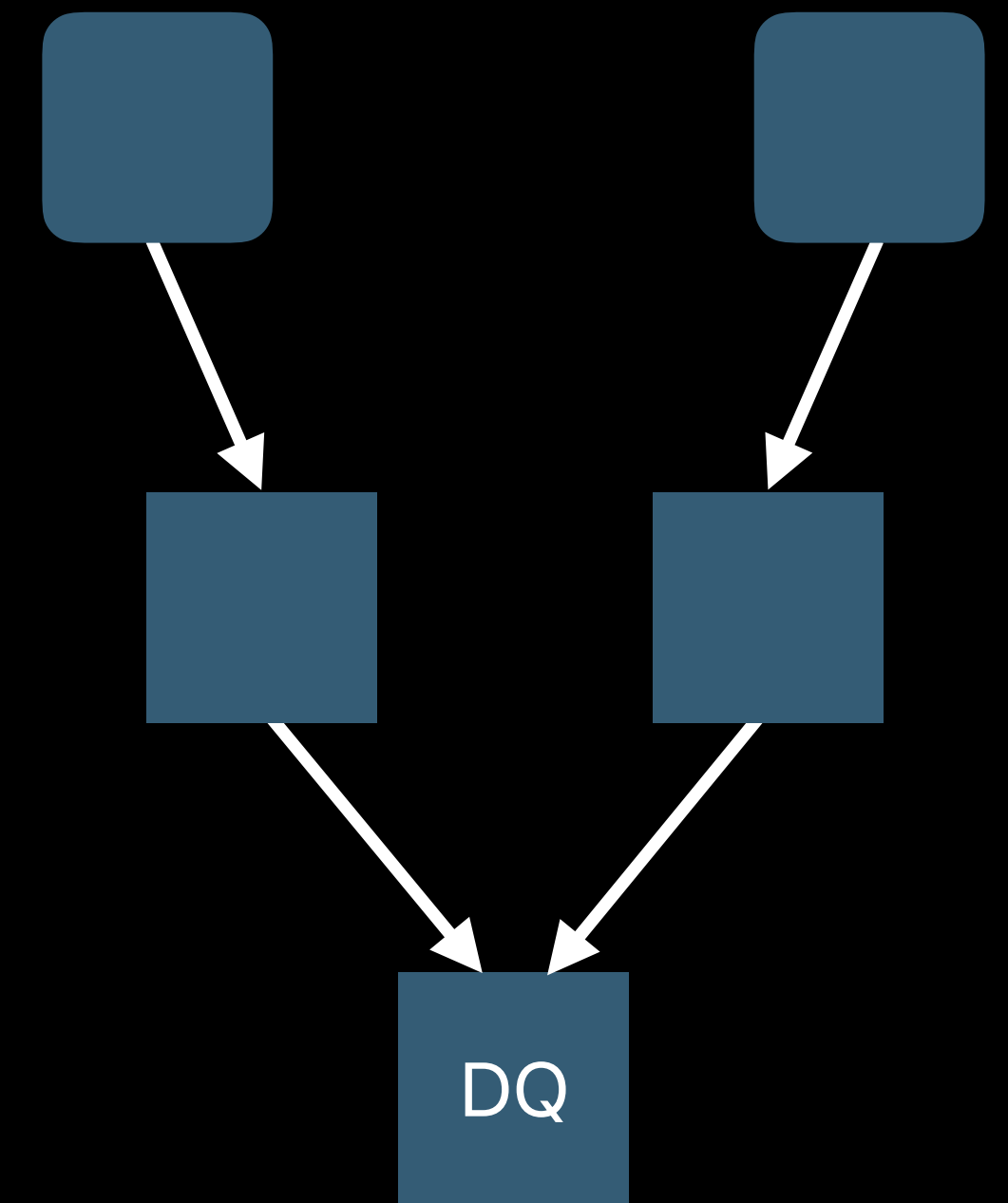
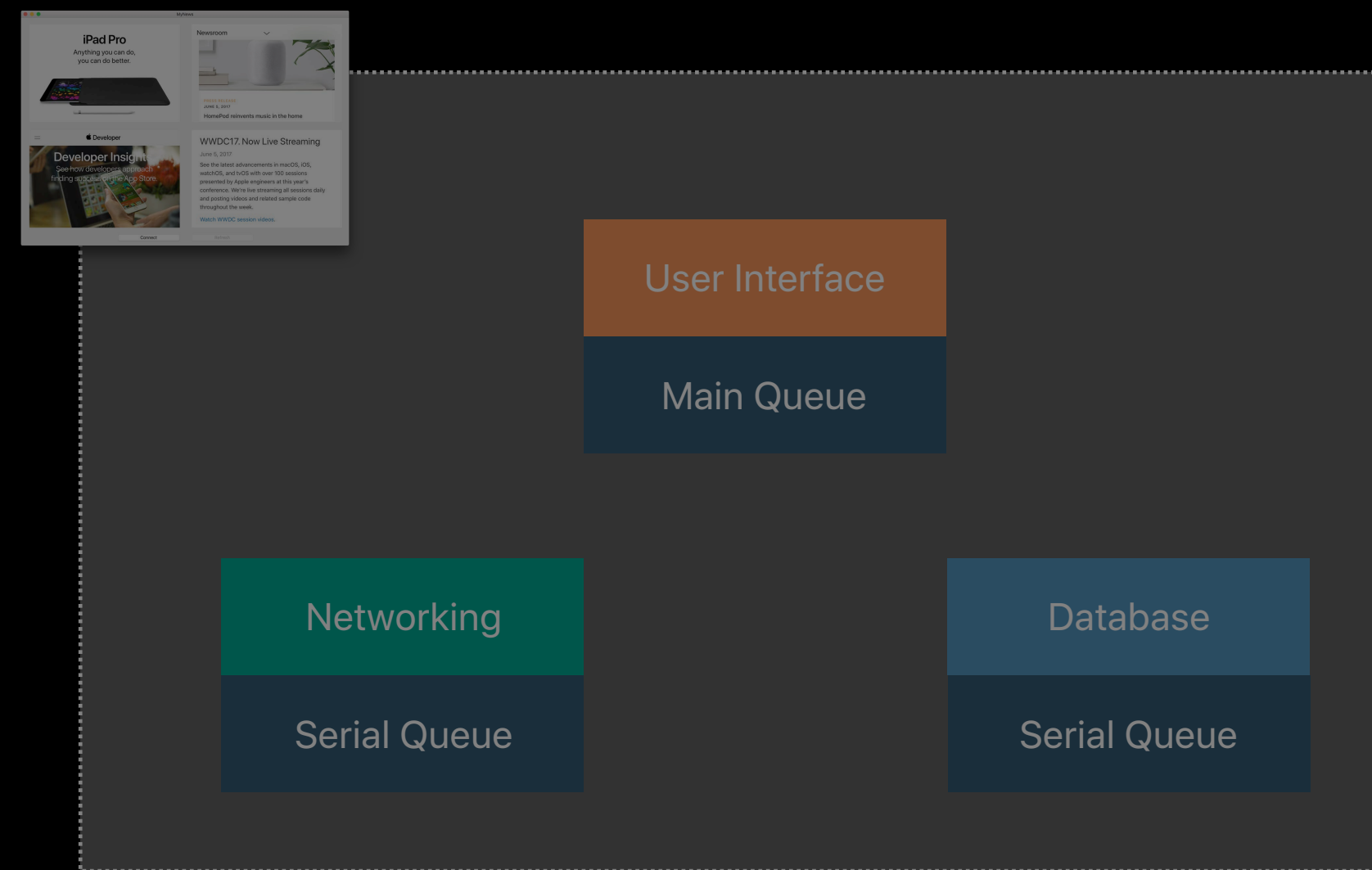
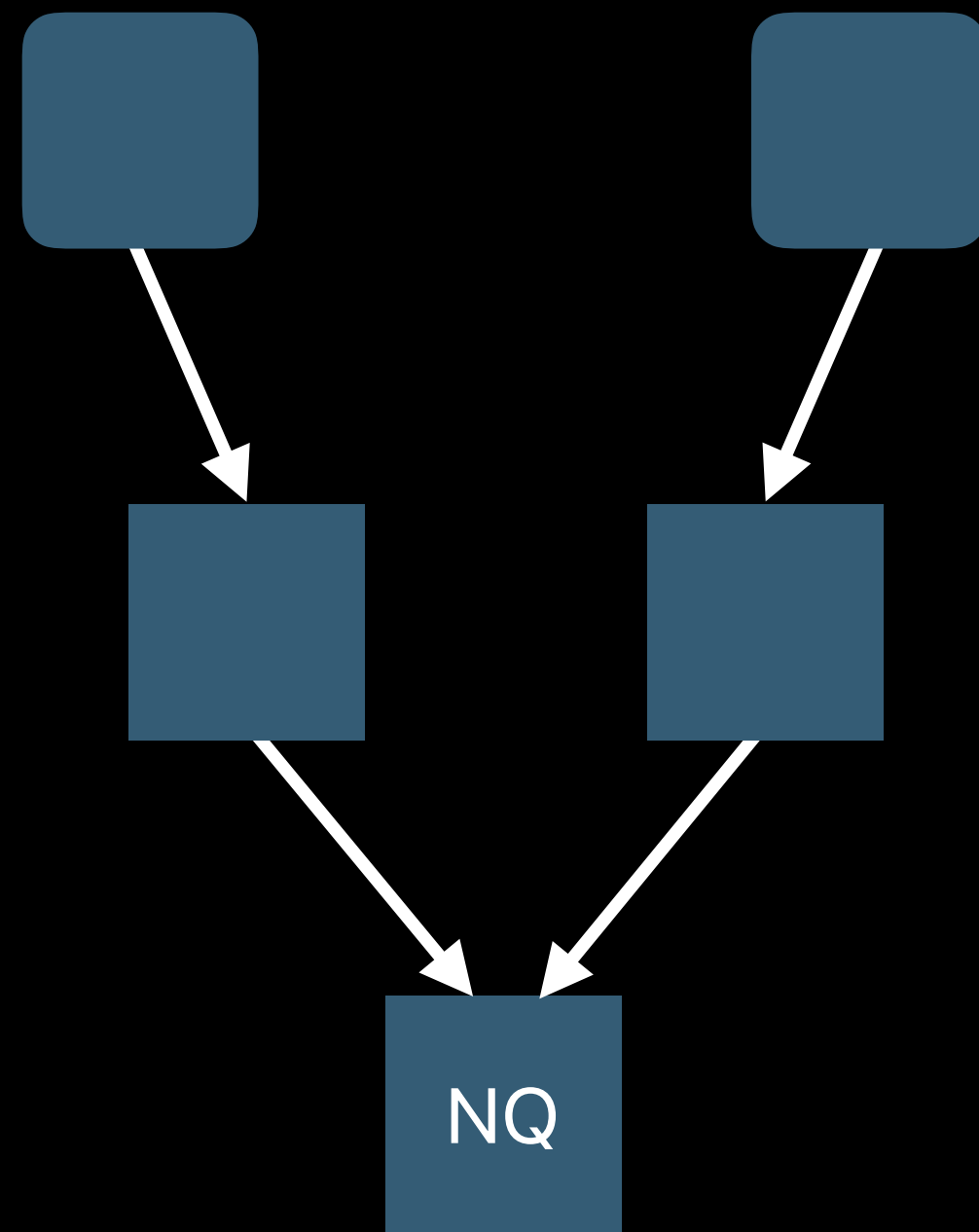
# One Queue per Subsystem



# One Queue Hierarchy per Subsystem



# One Queue Hierarchy per Subsystem



# Good Granularity of Concurrency

Fixed number of serial queue hierarchies

# Good Granularity of Concurrency



Fixed number of serial queue hierarchies

# Good Granularity of Concurrency



Fixed number of serial queue hierarchies

Coarse workitem granularity between hierarchies

# Good Granularity of Concurrency



Fixed number of serial queue hierarchies

Coarse workitem granularity between hierarchies



# Good Granularity of Concurrency



Fixed number of serial queue hierarchies

Coarse workitem granularity between hierarchies

Finer workitem granularity inside a hierarchy





# Using GCD for Concurrency

Organize queues and sources into serial queue hierarchies

Use a fixed number of serial queue hierarchies

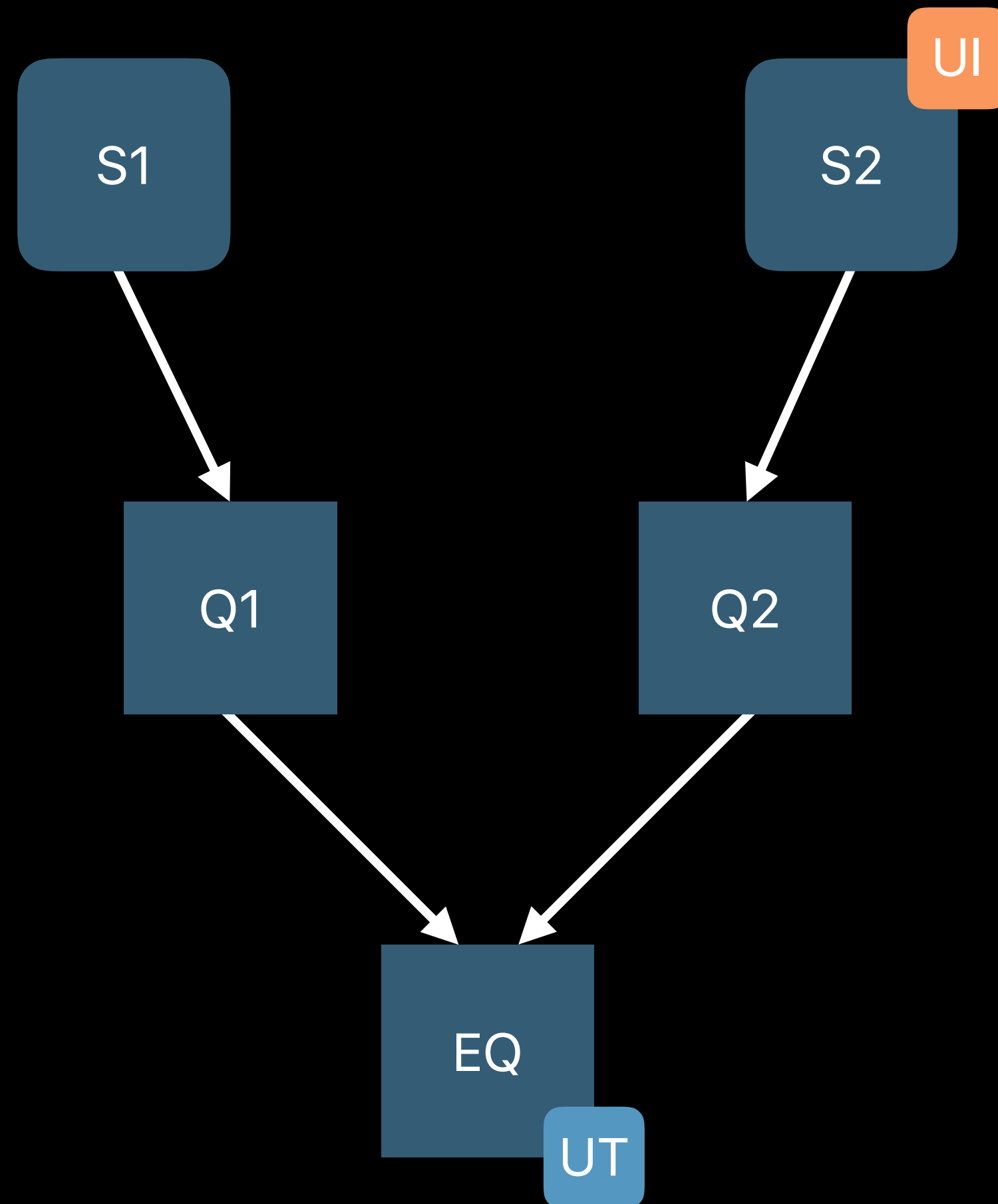
Size your workitems appropriately

# Introducing Unified Queue Identity

Pierre Habouzit, Core Darwin

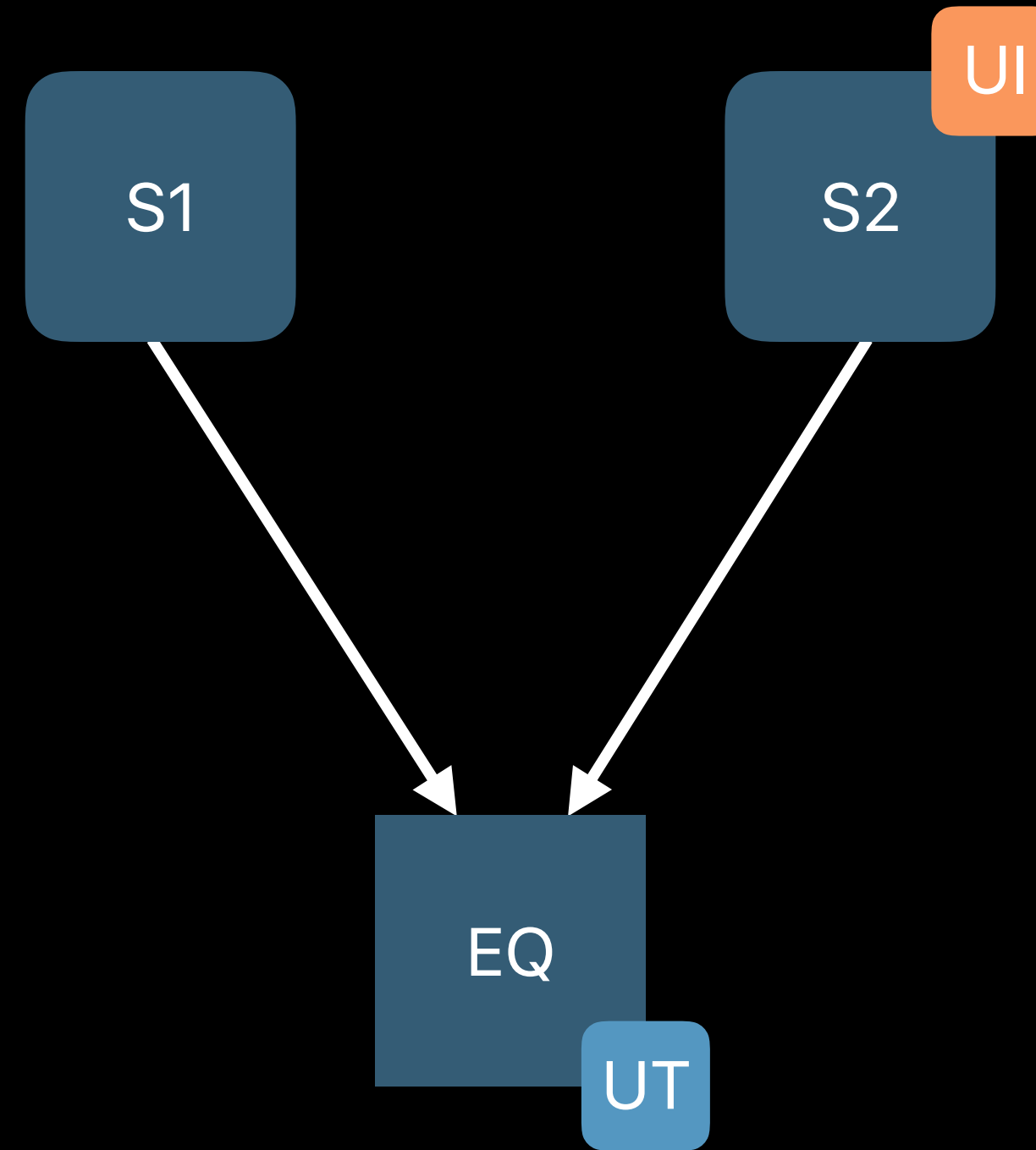
# Mutual Exclusion Context

Deep dive



# Mutual Exclusion Context

Deep dive



# Unified Queue Identity

Kernel

Application



EQ

```
let EQ = DispatchQueue(label: "com.example.exclusion-context")
```

# Unified Queue Identity

Asynchronous workitems

Kernel

Application

EQ

```
EQ.async { ... }
```

# Unified Queue Identity

Asynchronous workitems

Kernel

Application



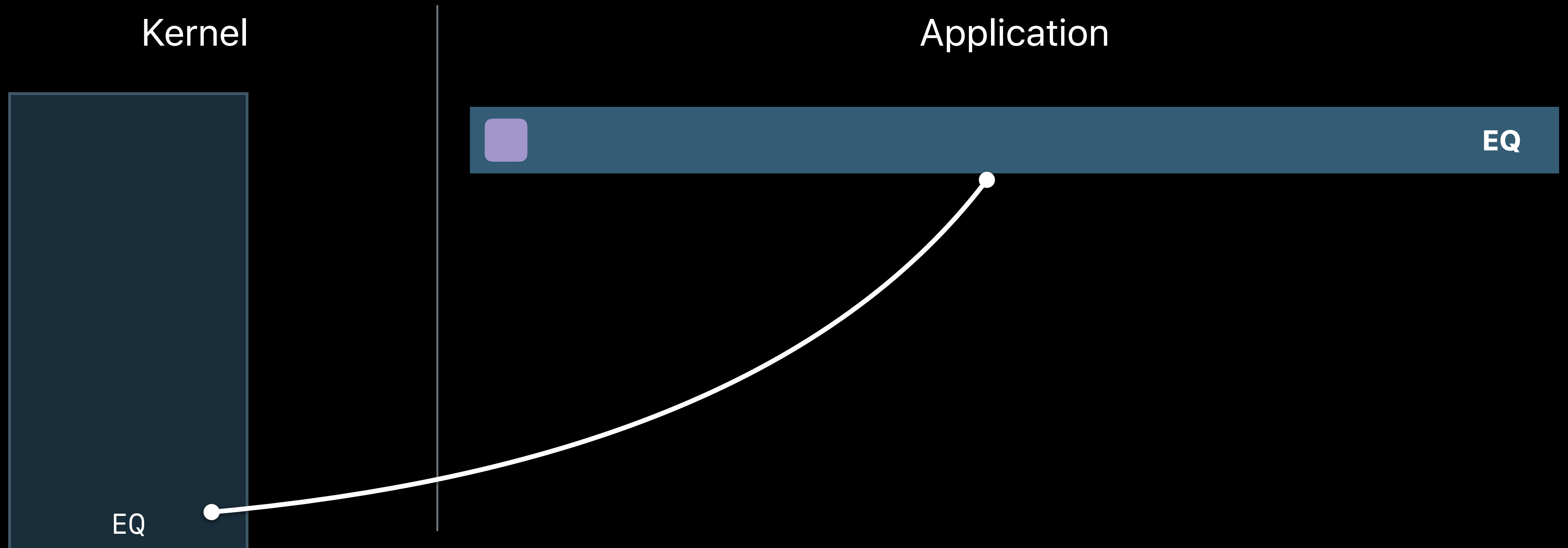
EQ

```
EQ.async { ... }
```

# Unified Queue Identity

Asynchronous workitems

NEW

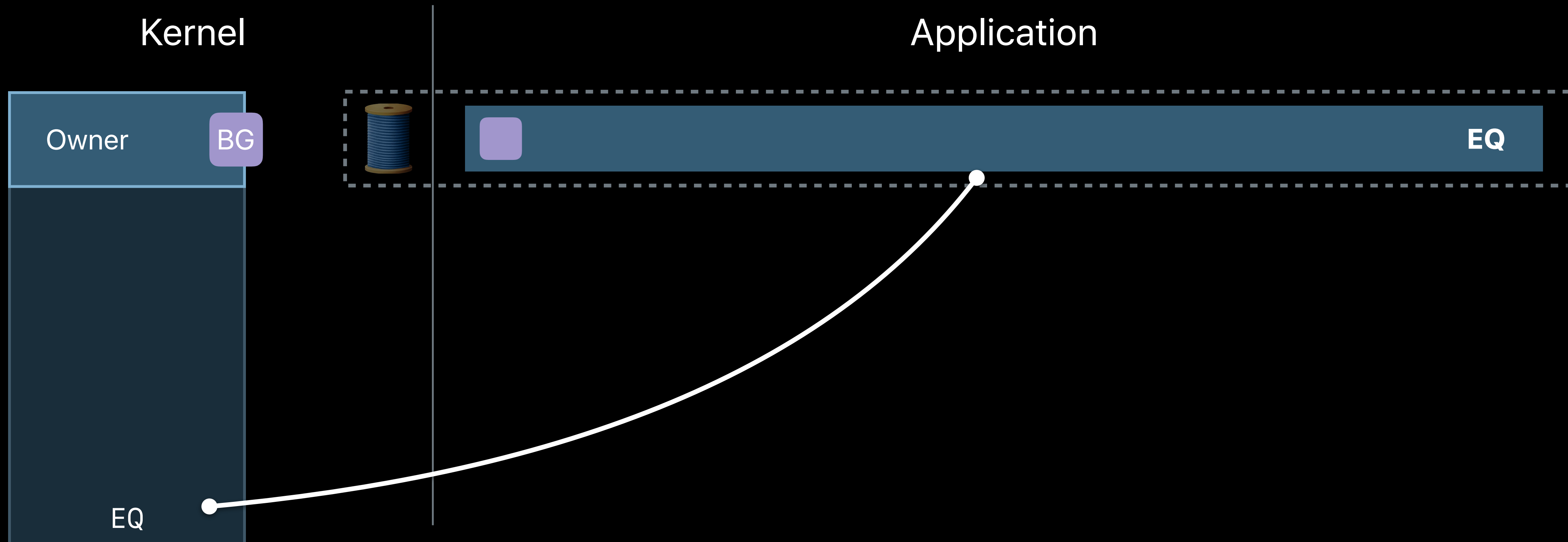


```
EQ.async { ... }
```



# Unified Queue Identity

Asynchronous workitems

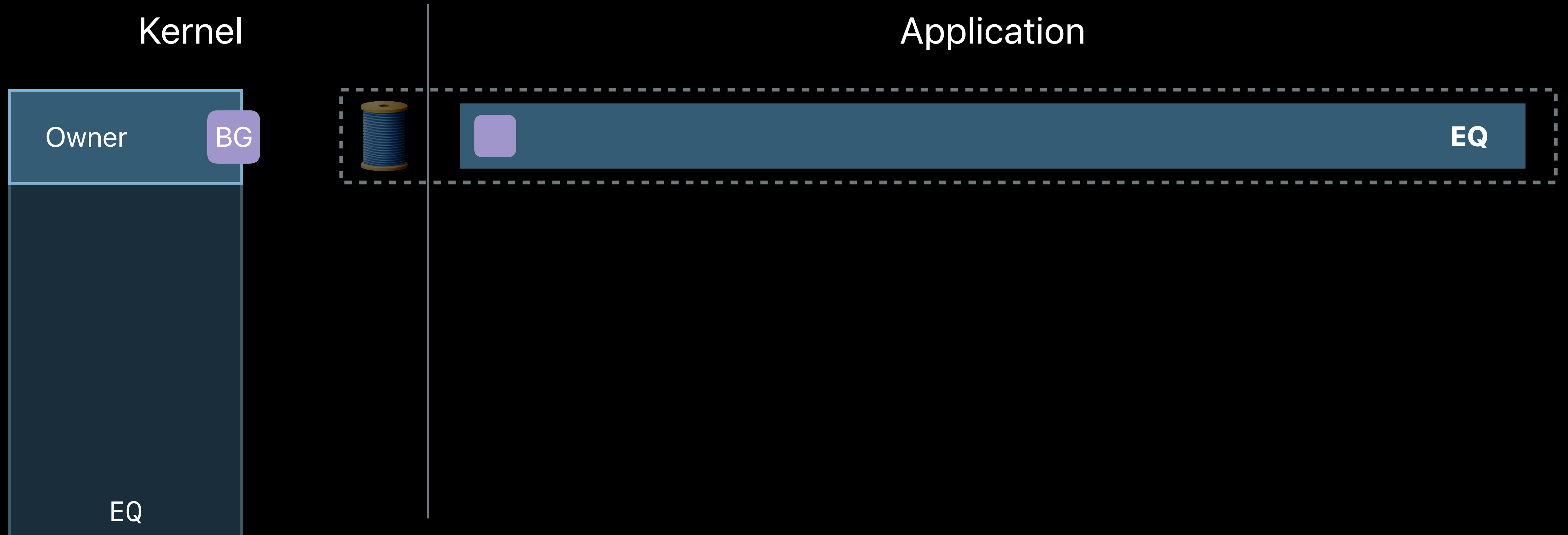


```
EQ.async { ... }
```

# Unified Queue Identity

Asynchronous workitems

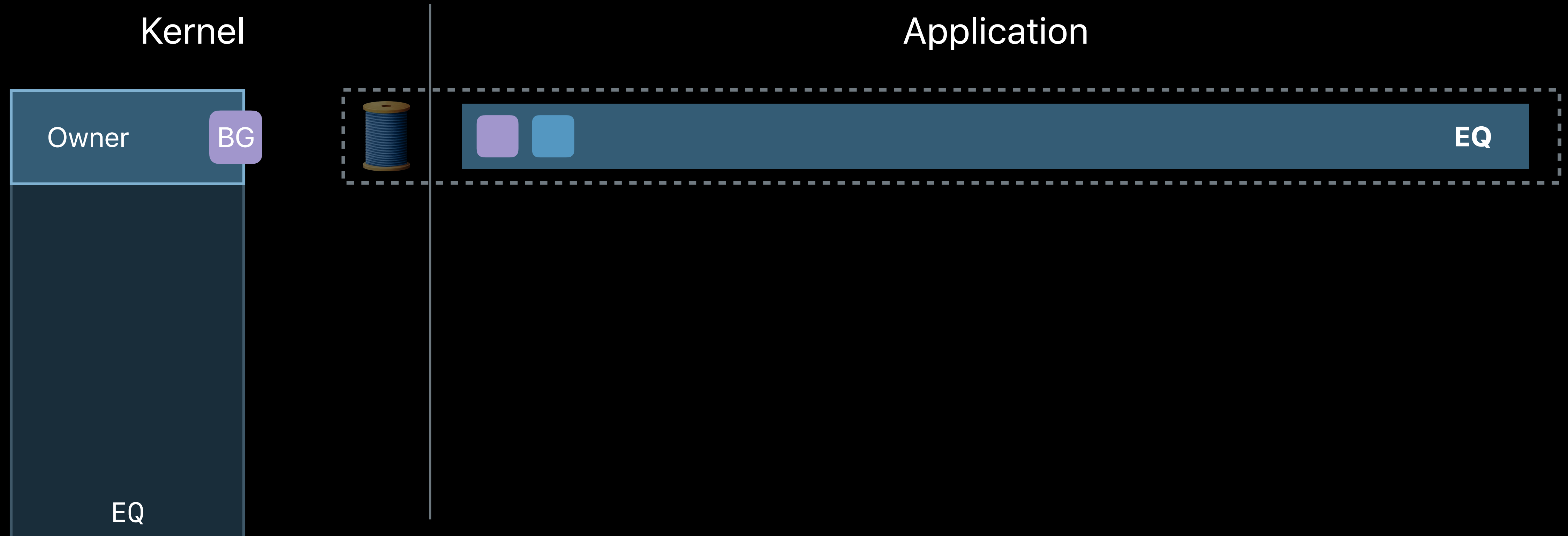
NEW



# Unified Queue Identity

Asynchronous workitems

NEW

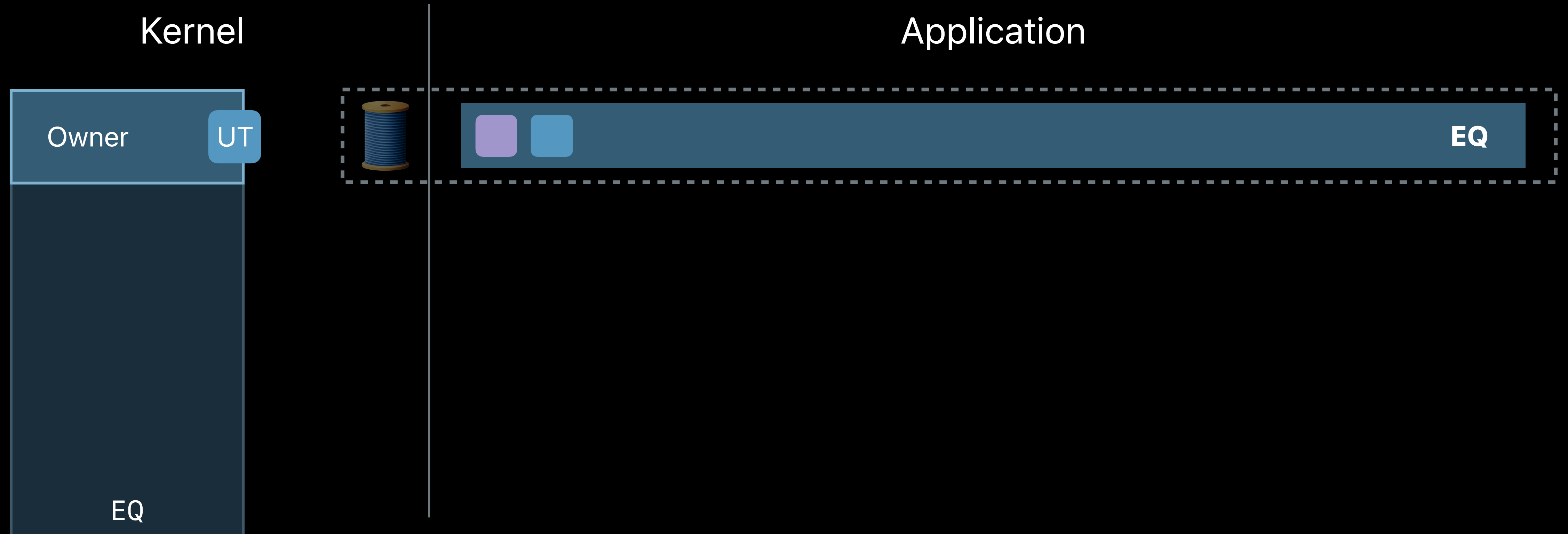


```
EQ.async { ... }
```

# Unified Queue Identity

Asynchronous workitems

NEW

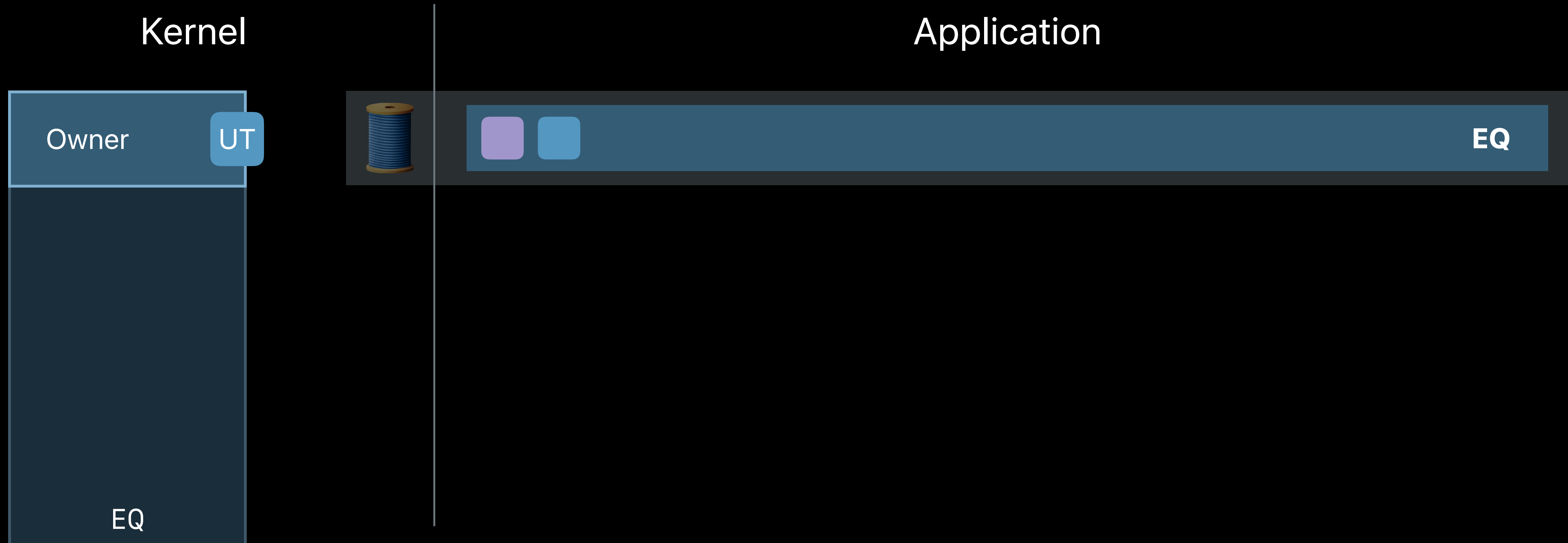


```
EQ.async { ... }
```

# Unified Queue Identity

Asynchronous workitems

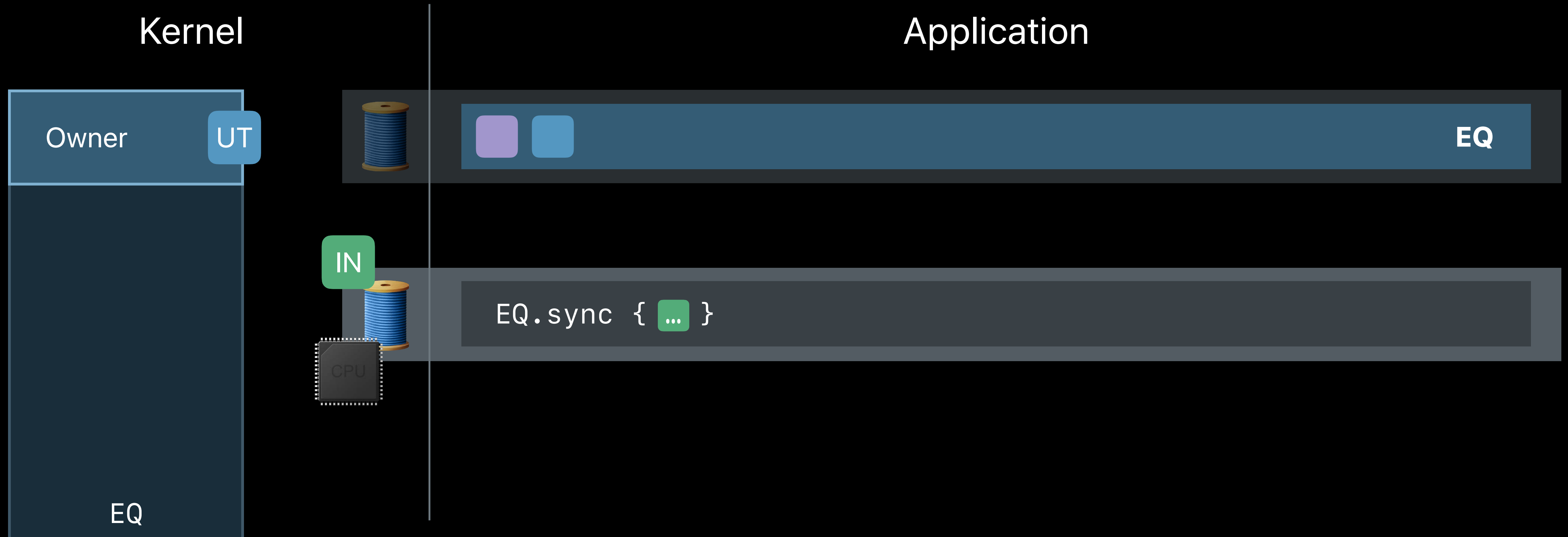
NEW



# Unified Queue Identity

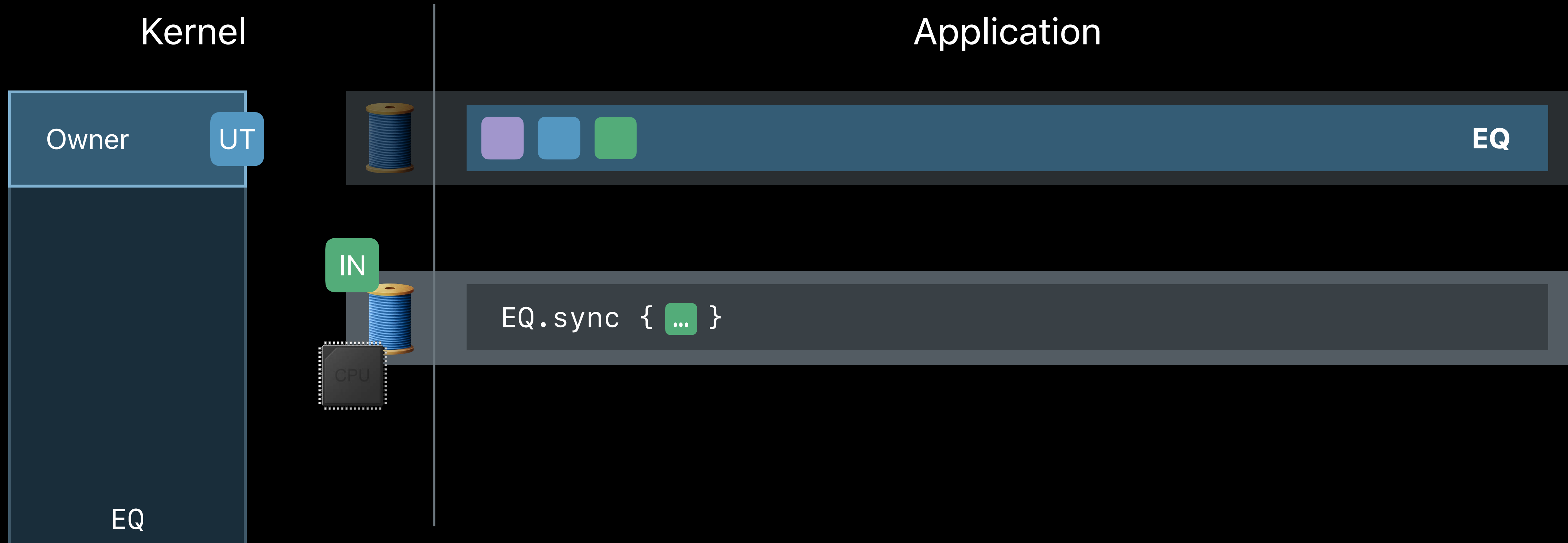
Synchronous workitems

NEW



# Unified Queue Identity

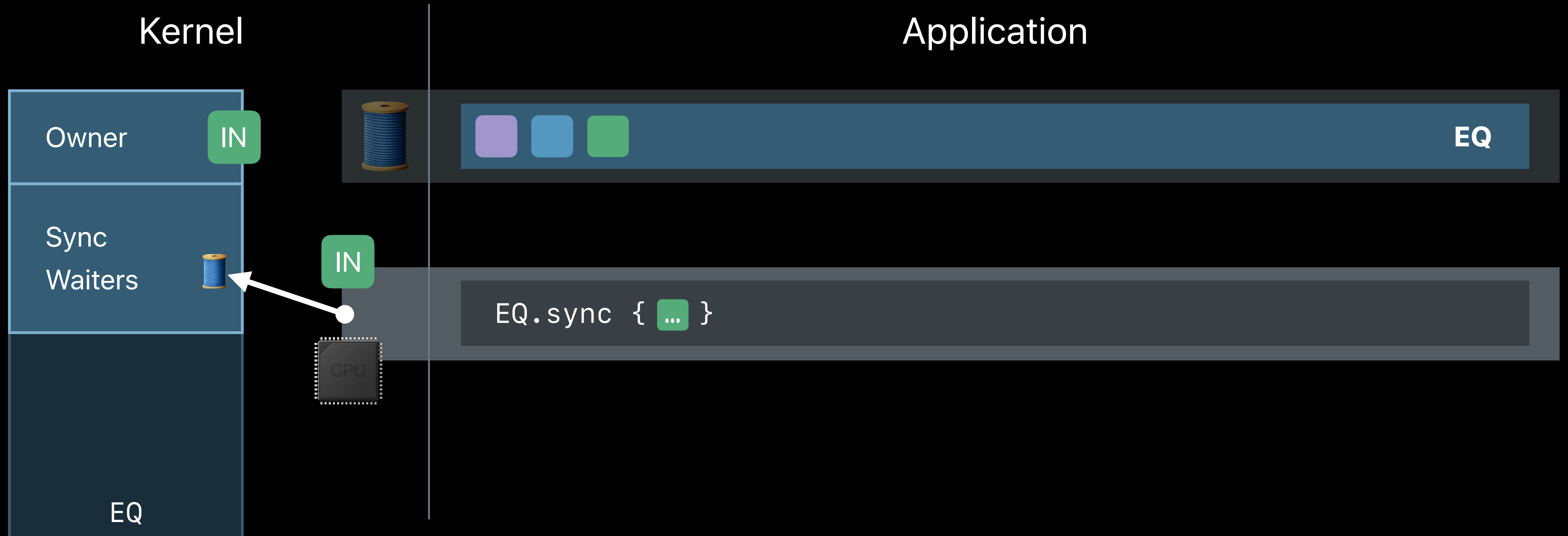
Synchronous workitems



# Unified Queue Identity

Synchronous workitems

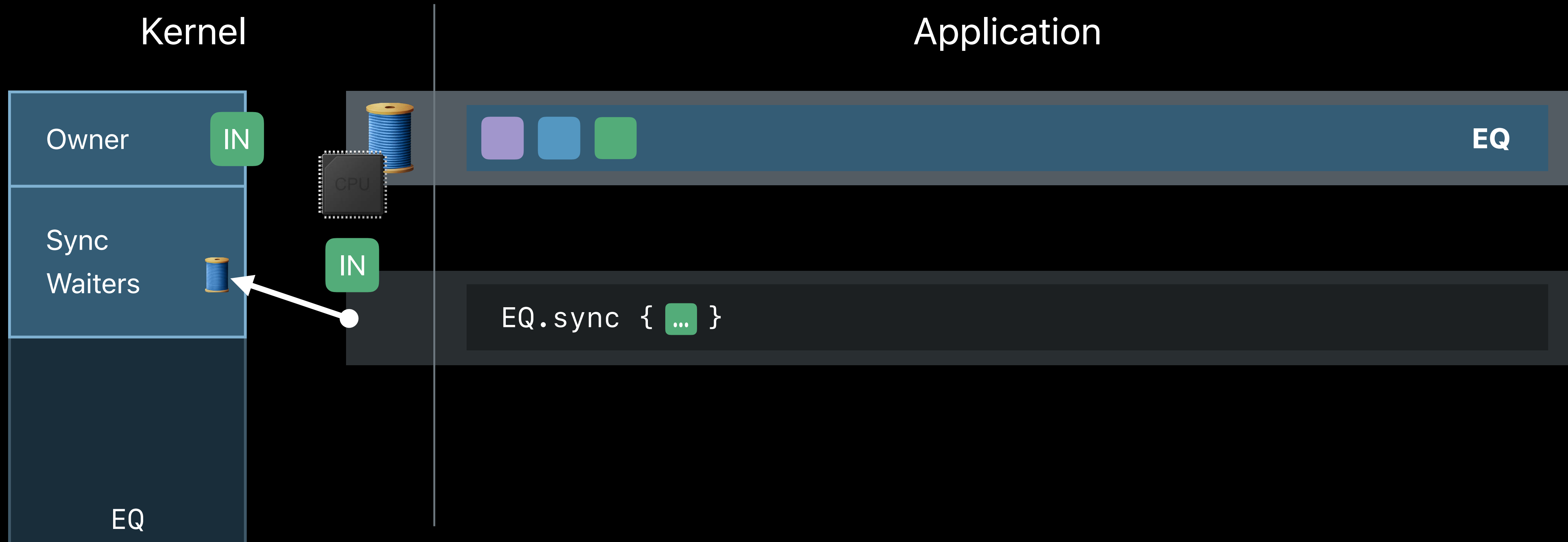
NEW





# Unified Queue Identity

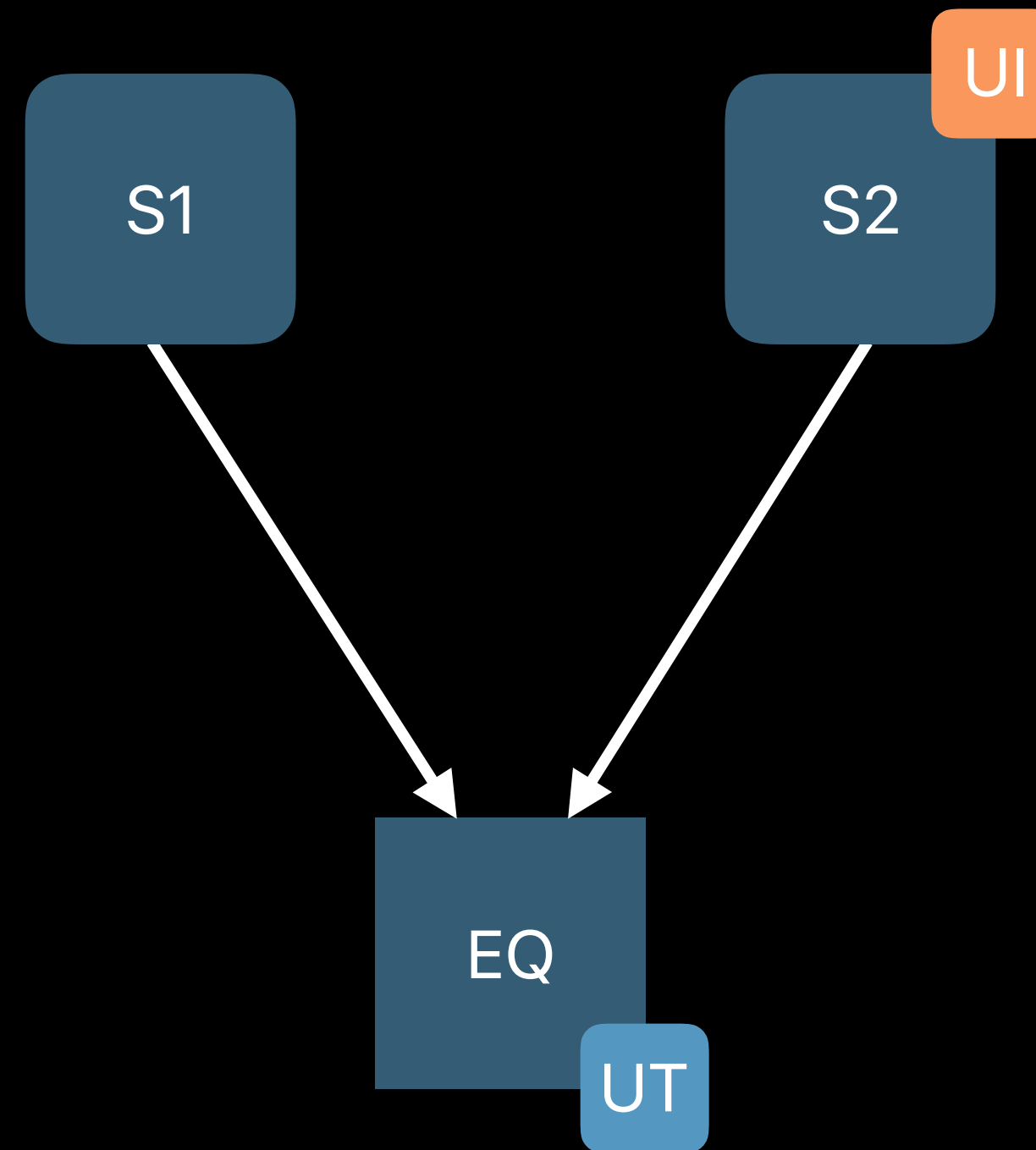
Synchronous workitems



# One Identity to Find Them All

... and in the kernel bind them

NEW



# One Identity to Find Them All

... and in the kernel bind them

NEW

Kernel

Application

UT

EQ

S1

```
let S1 = DispatchSource.makeReadSource(  
    fileDescriptor: fd, queue: EQ)  
S1.setEventHandler { ... }  
S1.activate()
```

# One Identity to Find Them All

... and in the kernel bind them

NEW

Kernel

Application

UT

EQ

S1

```
let S1 = DispatchSource.makeReadSource(  
    fileDescriptor: fd, queue: EQ)  
S1.setEventHandler { ... }  
S1.activate()
```

# One Identity to Find Them All

... and in the kernel bind them

NEW

Kernel

Application

UT

EQ

UT

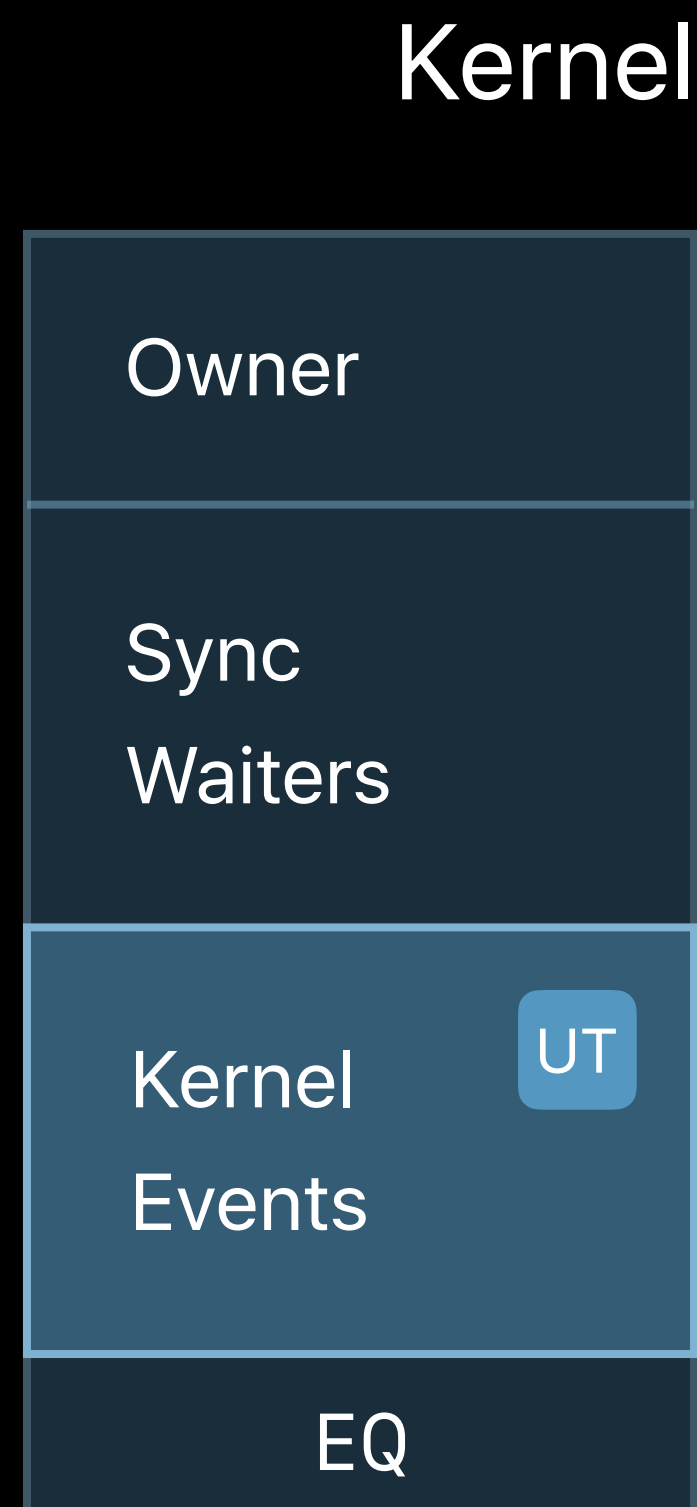
S1

```
let S1 = DispatchSource.makeReadSource(  
    fileDescriptor: fd, queue: EQ)  
S1.setEventHandler { ... }  
S1.activate()
```

# One Identity to Find Them All

... and in the kernel bind them

NEW



Application

UT

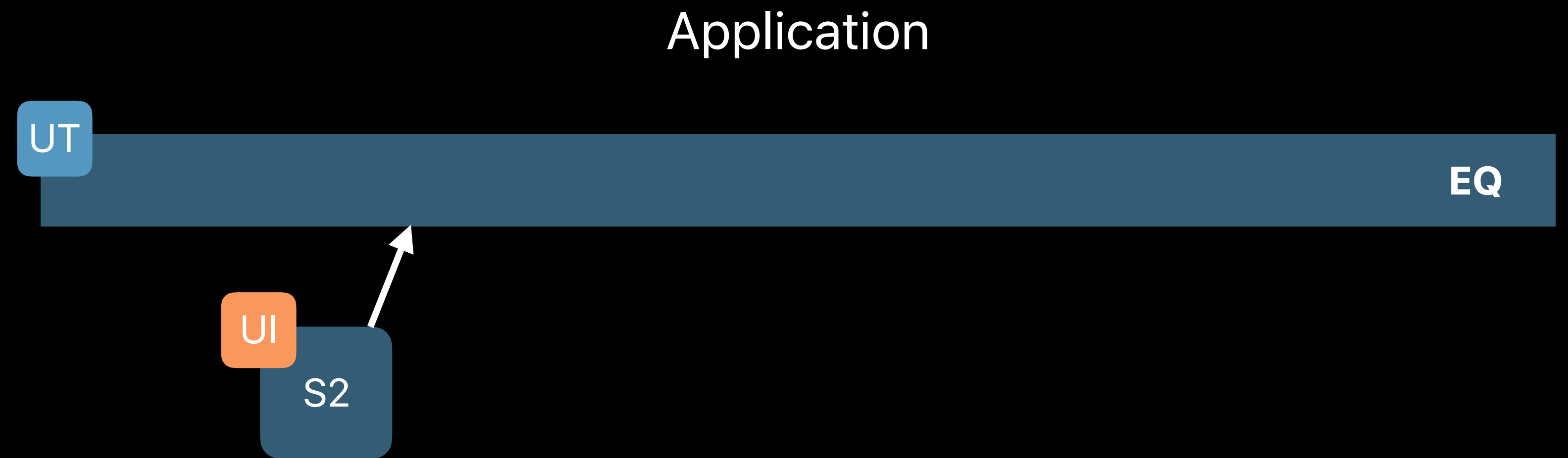
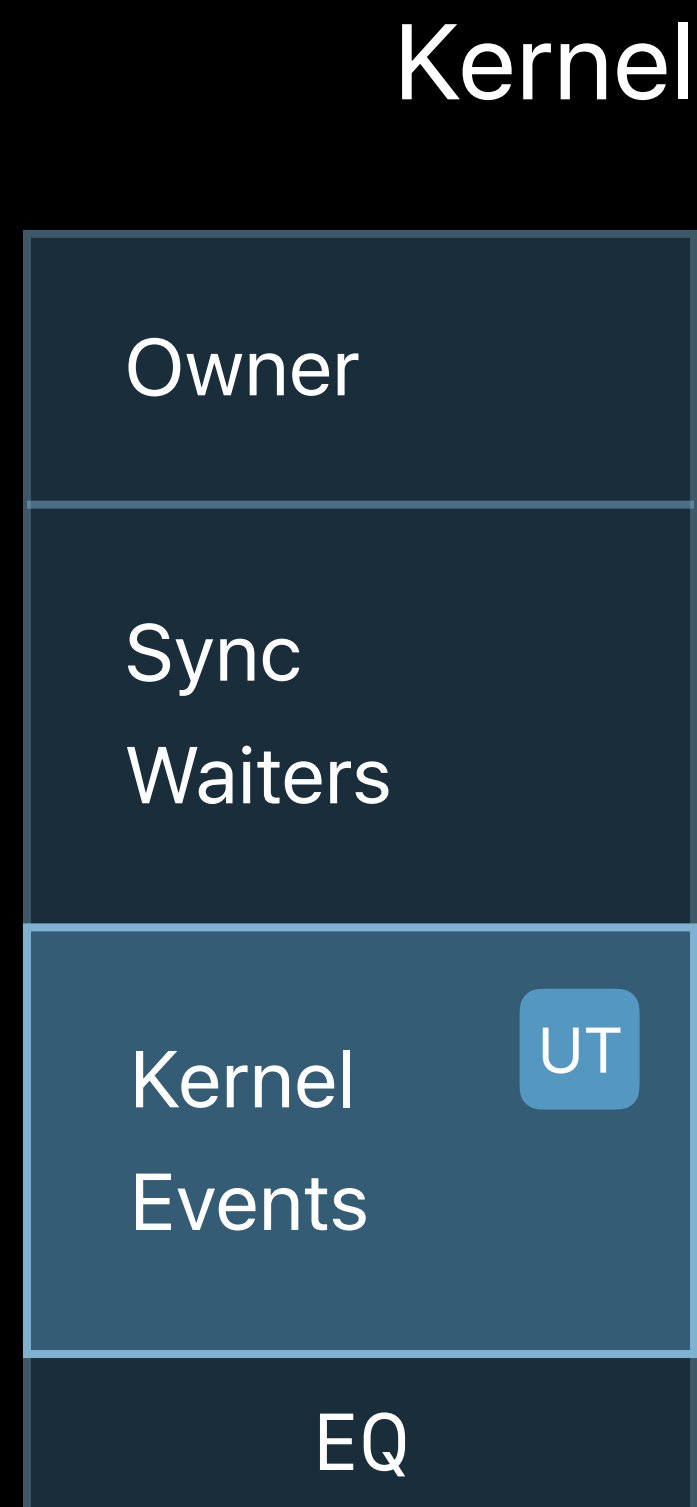
EQ

```
let S1 = DispatchSource.makeReadSource(  
    fileDescriptor: fd, queue: EQ)  
S1.setEventHandler { ... }  
S1.activate()
```

# One Identity to Find Them All

... and in the kernel bind them

NEW

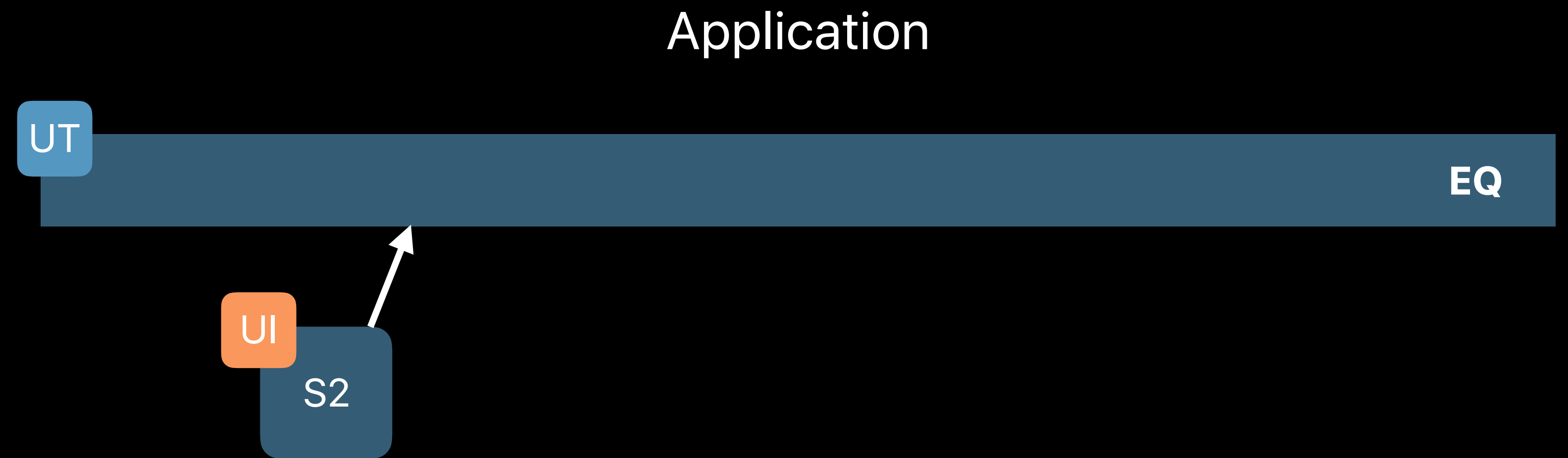
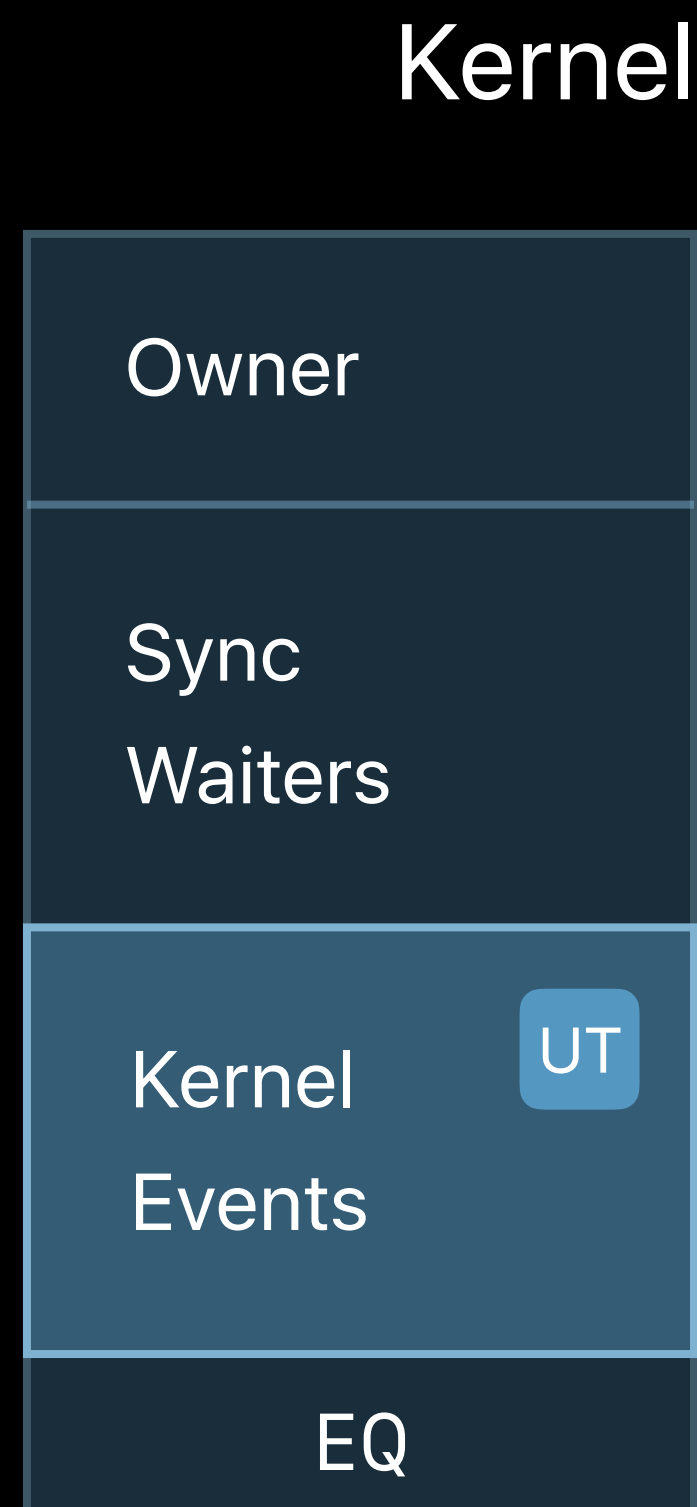


```
let S2 = DispatchSource.makeReadSource(  
    fileDescriptor: fd, queue: EQ)  
S2.setEventHandler(qos: .UserInteractive) { ... }  
S2.activate()
```

# One Identity to Find Them All

... and in the kernel bind them

NEW



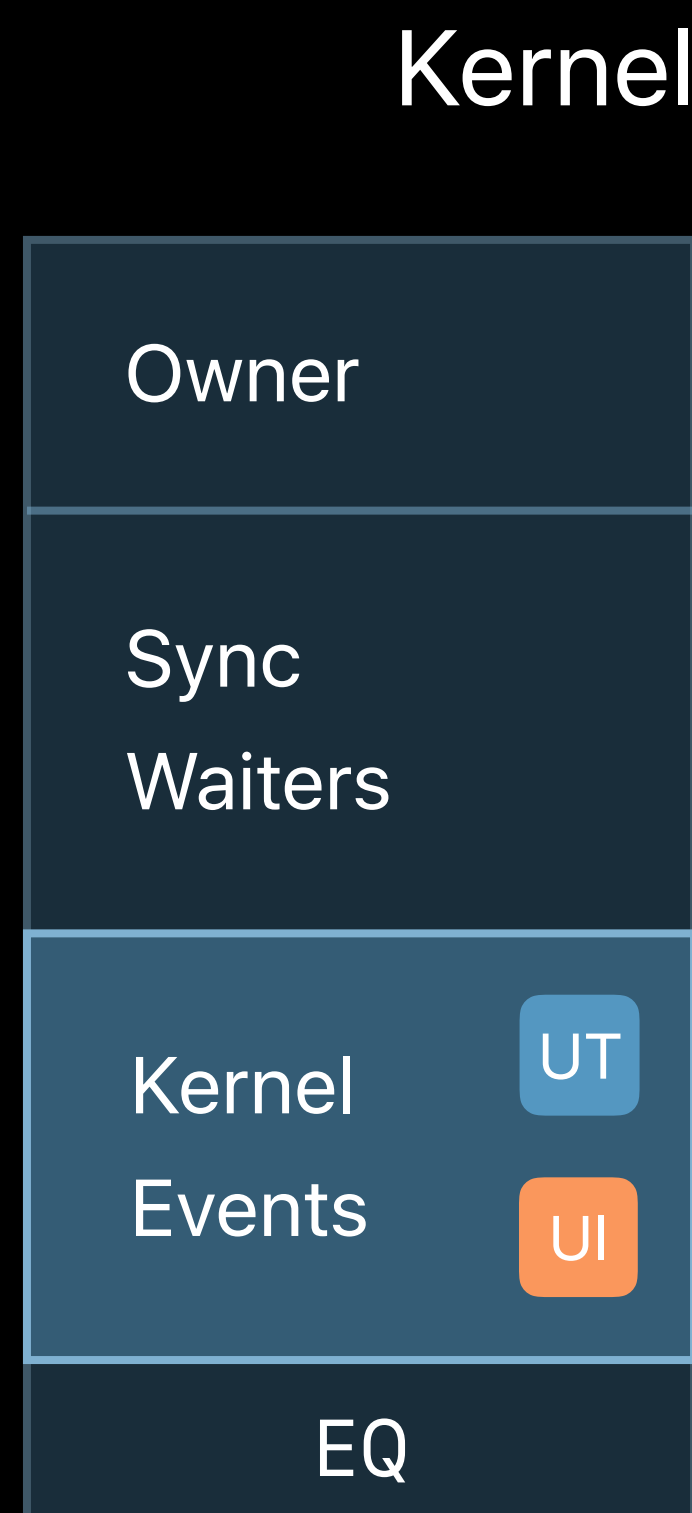
```
let S2 = DispatchSource.makeReadSource(  
    fileDescriptor: fd, queue: EQ)  
S2.setEventHandler(qos: .UserInteractive) { ... }  
S2.activate()
```



# One Identity to Find Them All

... and in the kernel bind them

NEW



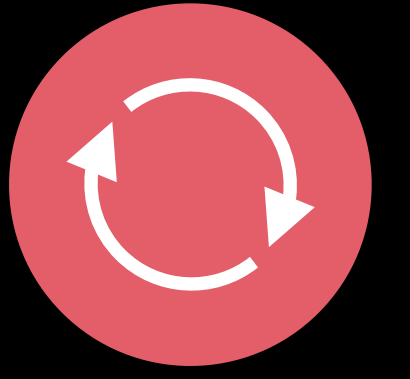
Application

UT

EQ

```
let S2 = DispatchSource.makeReadSource(  
    fileDescriptor: fd, queue: EQ)  
S2.setEventHandler(qos: .UserInteractive) { ... }  
S2.activate()
```

# Too Much of a Good Thing



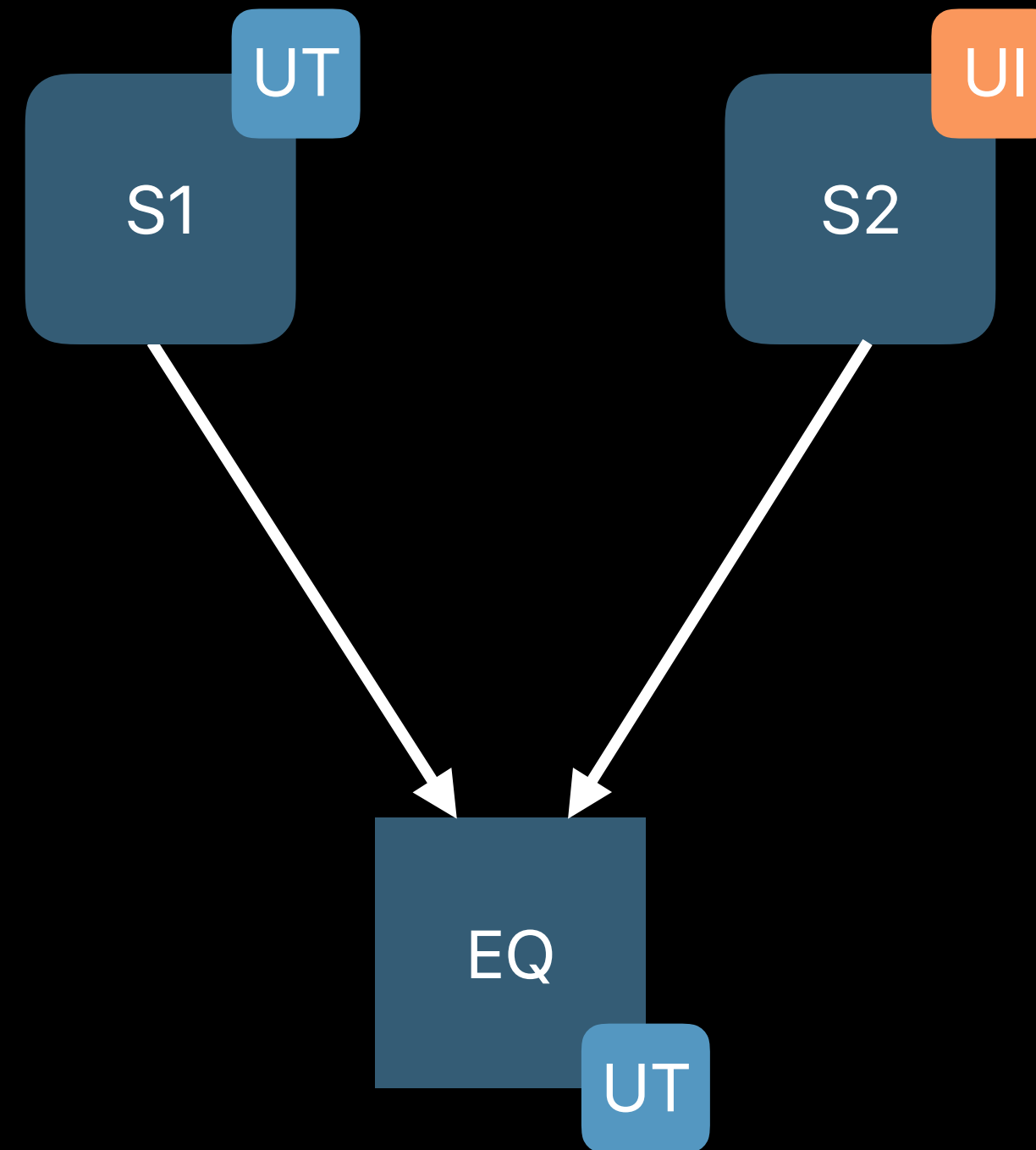
Repeatedly waiting for exclusive access to contended resources

Repeatedly switching between independent operations

Repeatedly bouncing an operation between threads

# Without Unified Identity

In macOS Sierra and iOS 10



# Without Unified Identity

In macOS Sierra and iOS 10



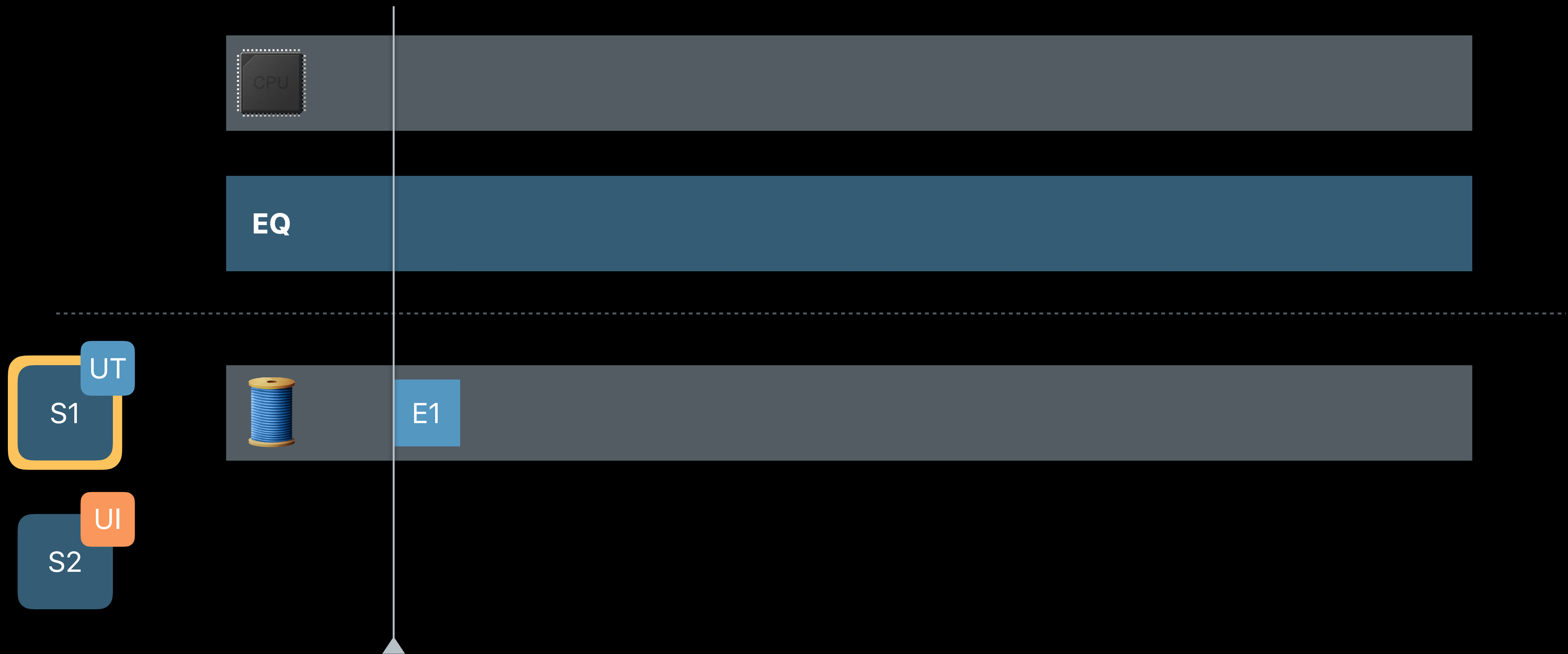
# Without Unified Identity

In macOS Sierra and iOS 10



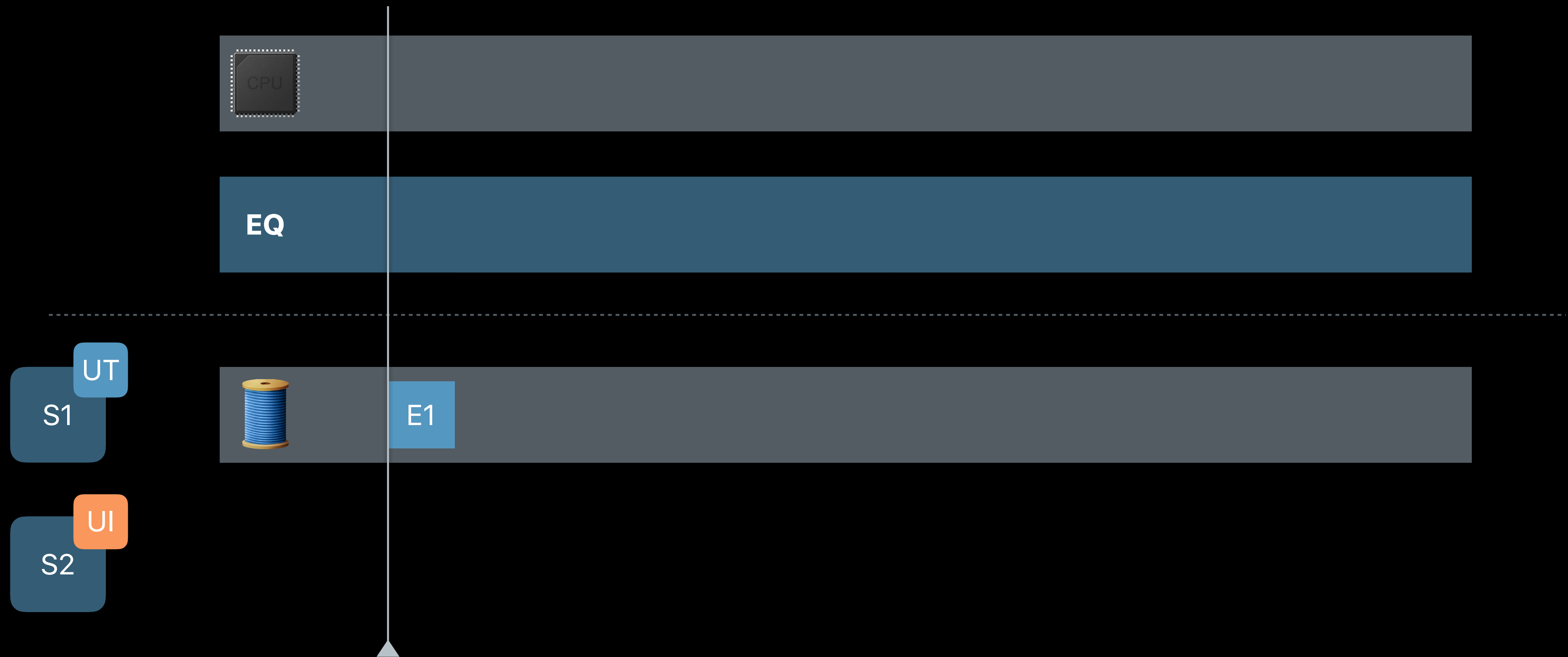
# Without Unified Identity

In macOS Sierra and iOS 10



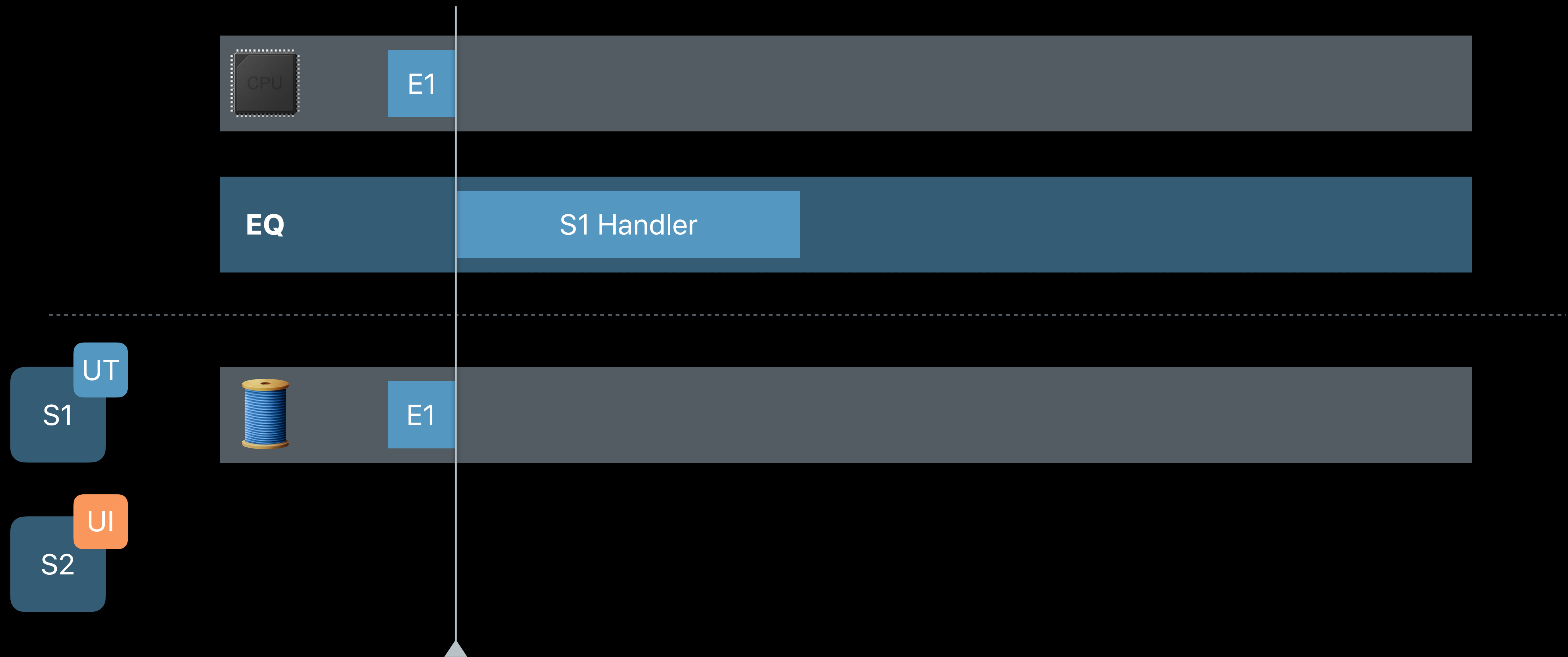
# Without Unified Identity

In macOS Sierra and iOS 10



# Without Unified Identity

In macOS Sierra and iOS 10





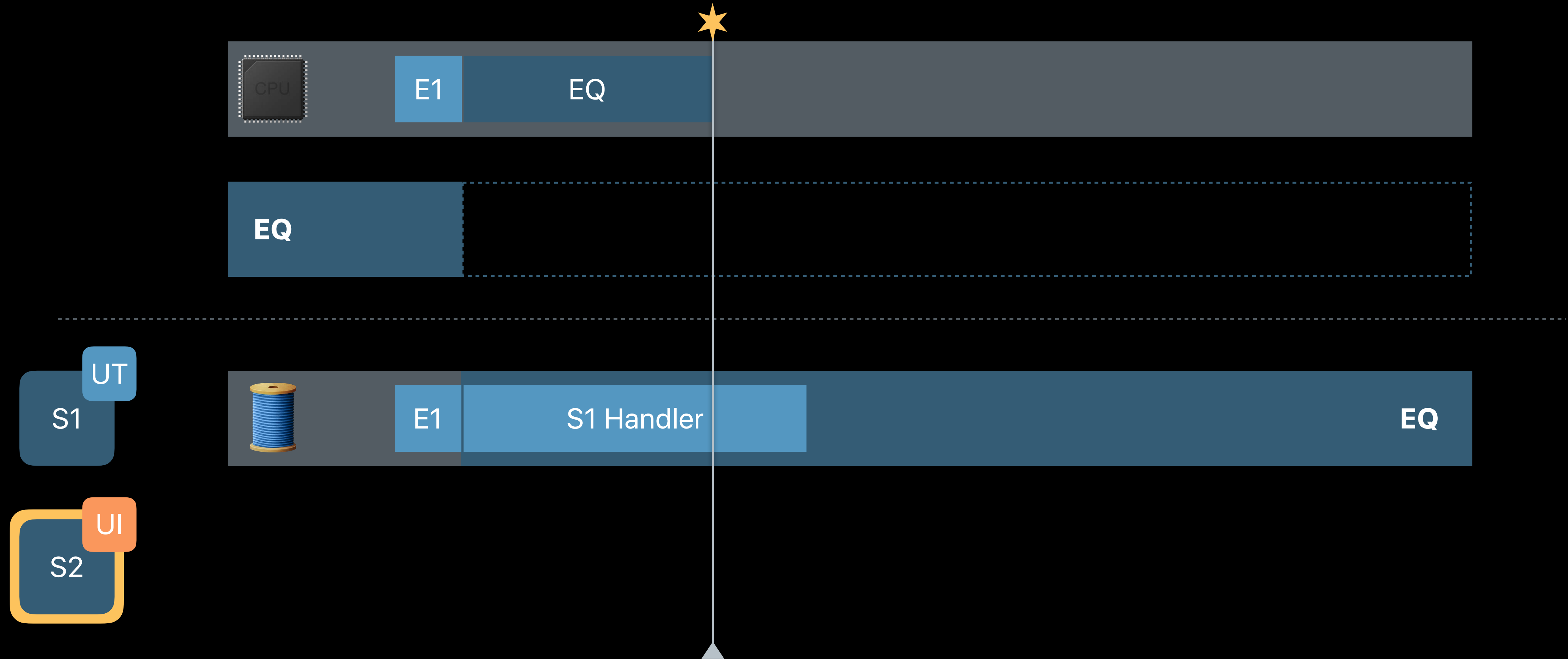
# Without Unified Identity

In macOS Sierra and iOS 10



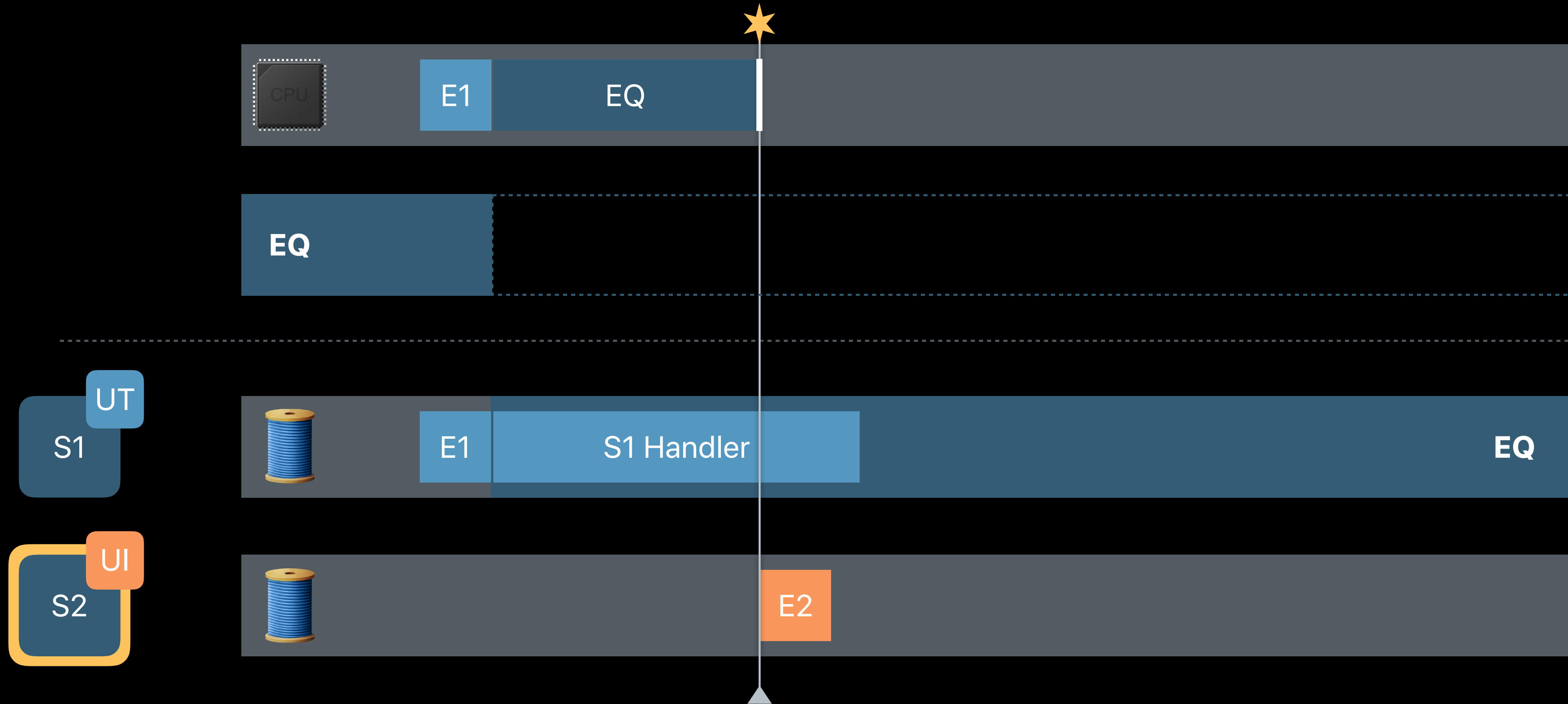
# Without Unified Identity

In macOS Sierra and iOS 10



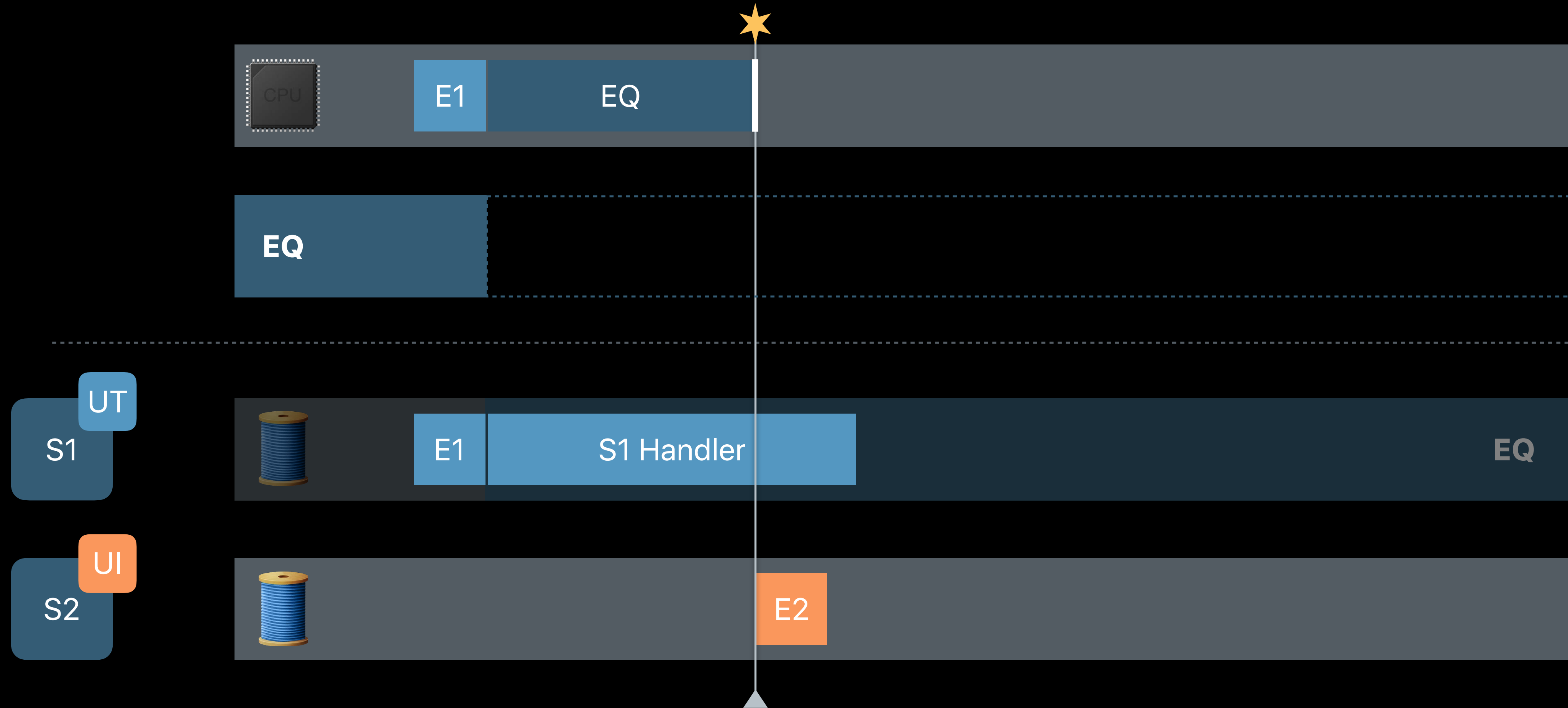
# Without Unified Identity

In macOS Sierra and iOS 10



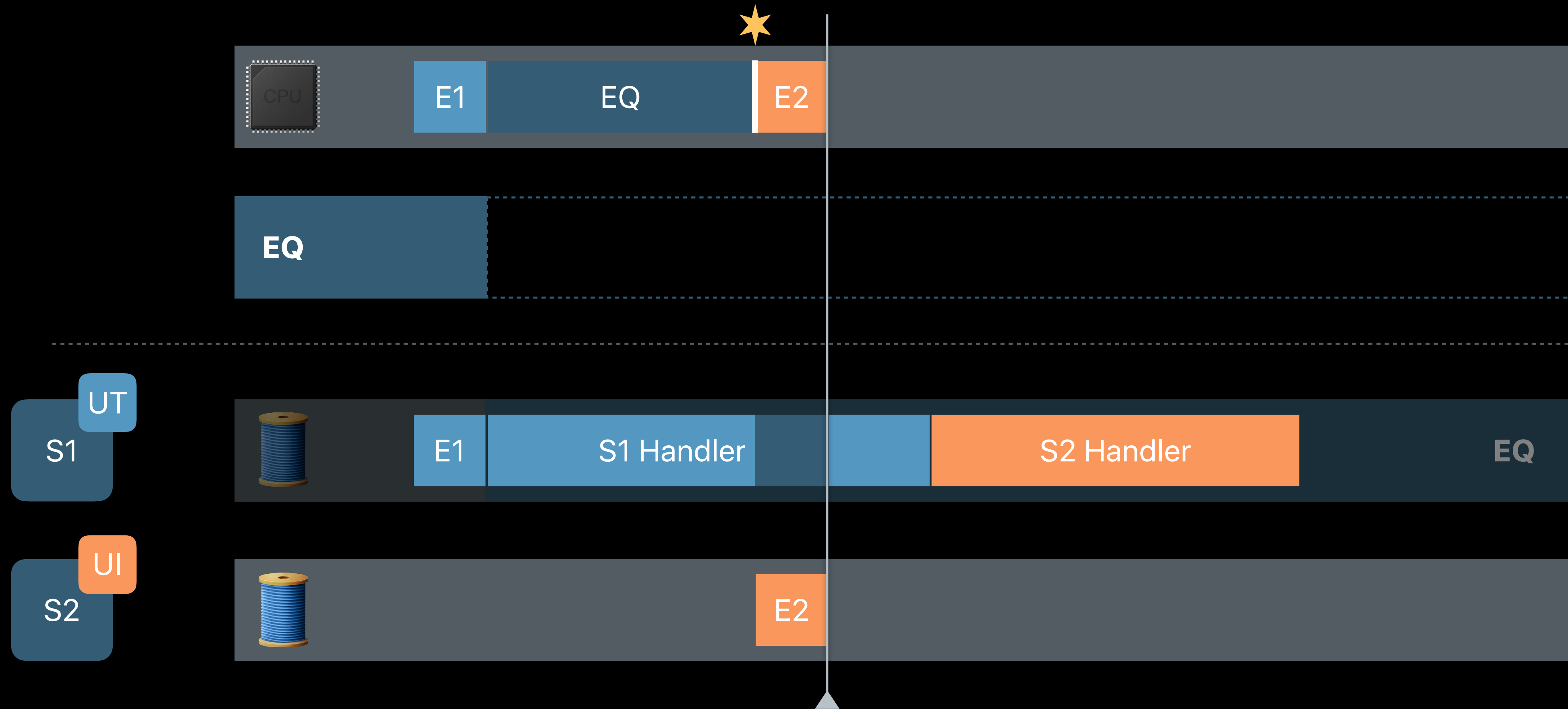
# Without Unified Identity

In macOS Sierra and iOS 10



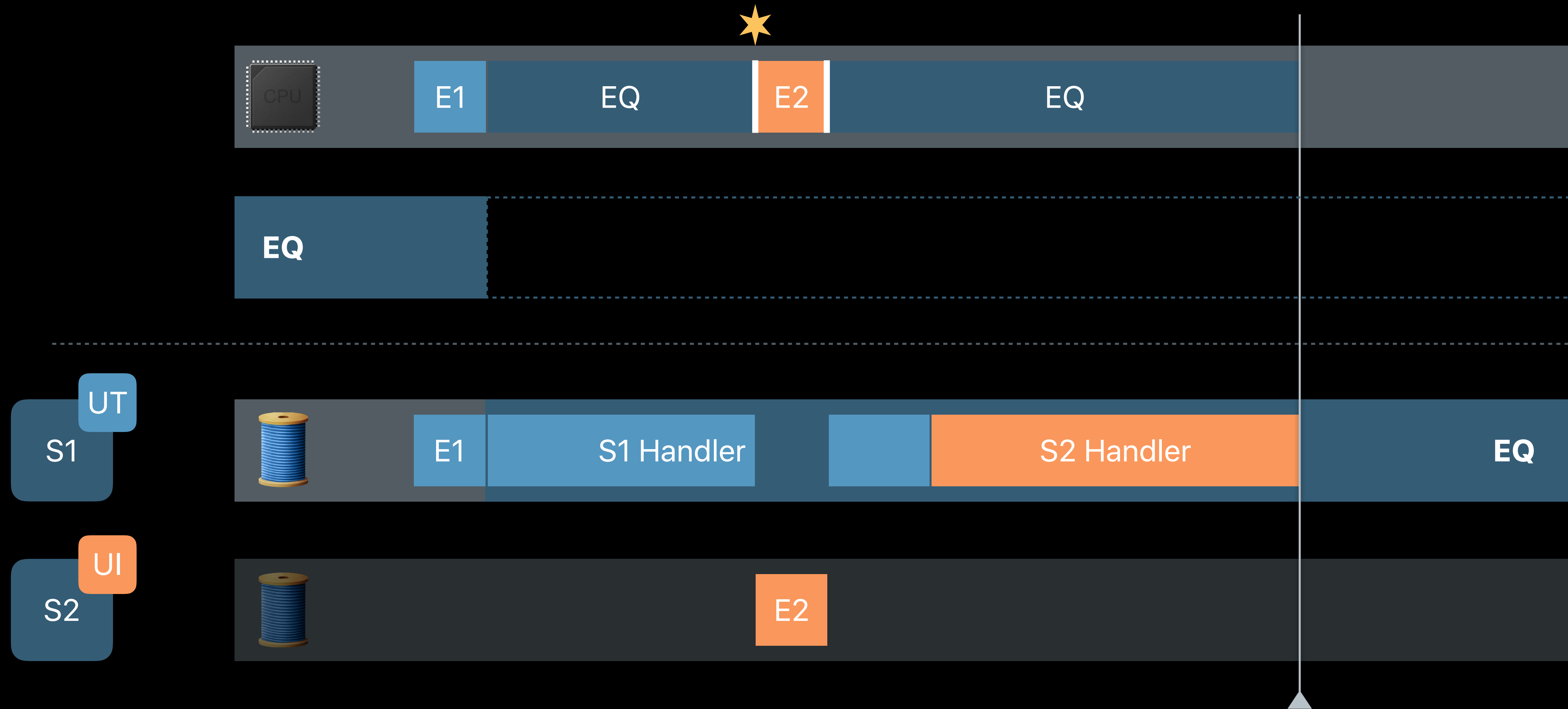
# Without Unified Identity

In macOS Sierra and iOS 10



# Without Unified Identity

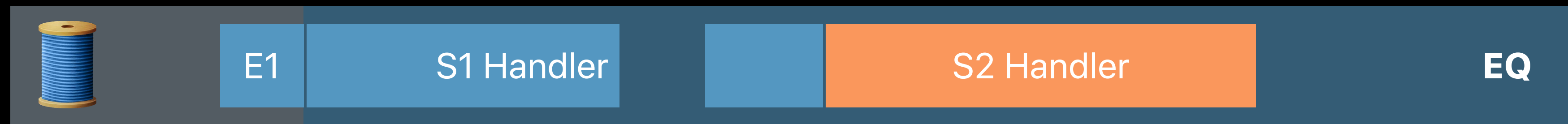
In macOS Sierra and iOS 10



# Leveraging Ownership and Unified Identity

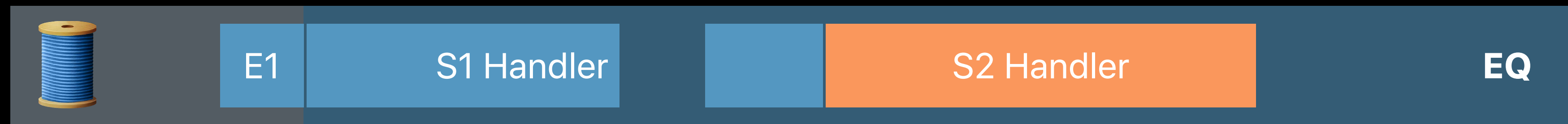


# Leveraging Ownership and Unified Identity



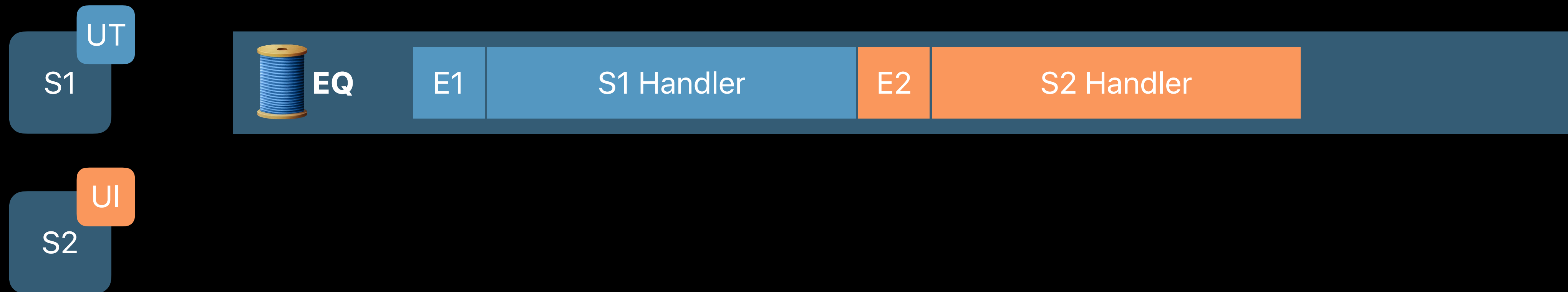


# Leveraging Ownership and Unified Identity



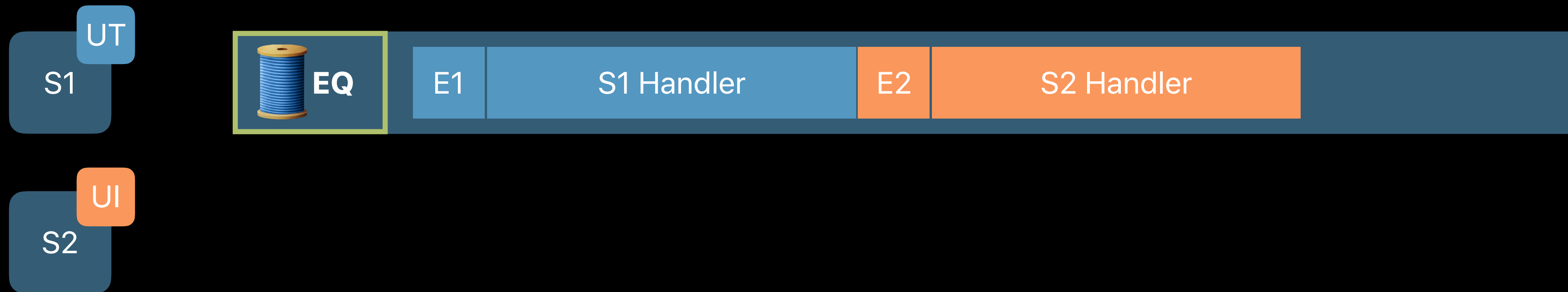
# Leveraging Ownership and Unified Identity

NEW



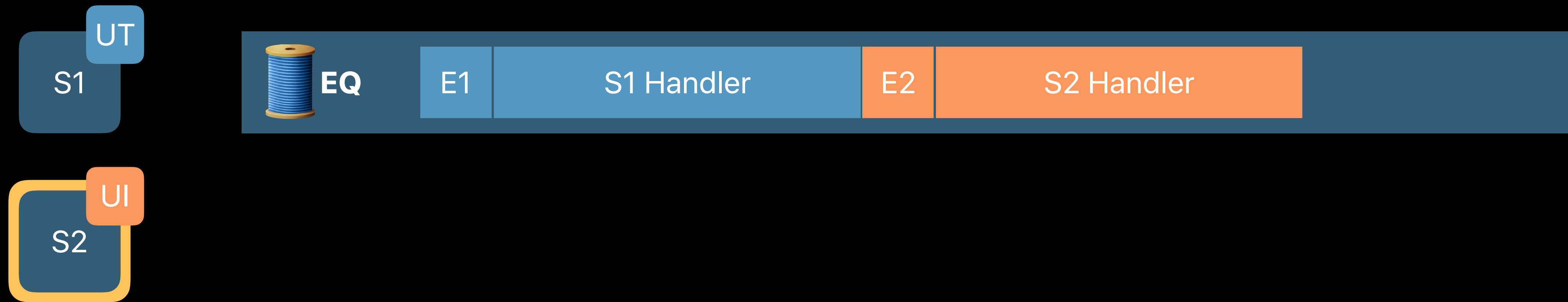
# Leveraging Ownership and Unified Identity

NEW



# Leveraging Ownership and Unified Identity

NEW

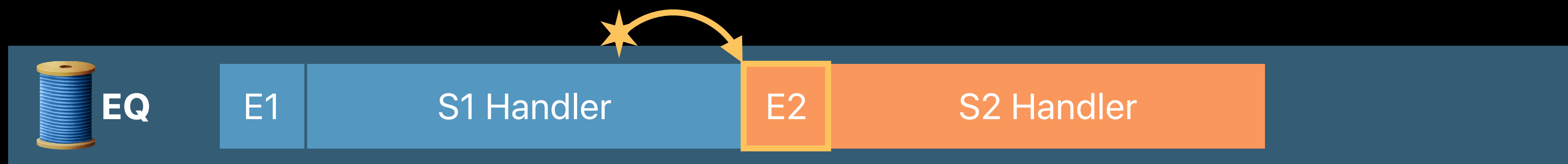
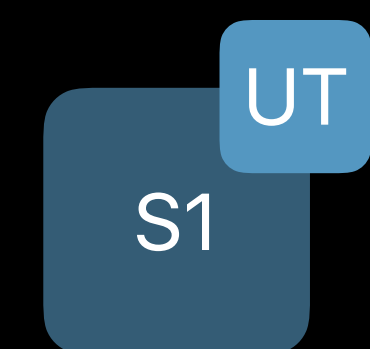


# Leveraging Ownership and Unified Identity

NEW



# Leveraging Ownership and Unified Identity



The runtime uses every possible hint  
to optimize behavior

# Modernizing Existing Code



# Modernizing Existing Code

No dispatch object mutation after activation

Protect your target queue hierarchy

# No Mutation Past Activation

Set the properties of inactive objects before activation

- Source handlers
- Target queues

# No Mutation Past Activation

Set the properties of inactive objects before activation

- Source handlers
- Target queues

```
let mySource = DispatchSource.makeReadSource(fileDescriptor: fd, queue: myQueue)
```

# No Mutation Past Activation

Set the properties of inactive objects before activation

- Source handlers
- Target queues

```
let mySource = DispatchSource.makeReadSource(fileDescriptor: fd, queue: myQueue)
```

```
mySource.setEventHandler(qos: .userInteractive) { ... }
```

```
mySource.setCancelHandler { close(fd) }
```

# No Mutation Past Activation



Set the properties of inactive objects before activation

- Source handlers
- Target queues

```
let mySource = DispatchSource.makeReadSource(fileDescriptor: fd, queue: myQueue)
```

```
mySource.setEventHandler(qos: .userInteractive) { ... }
```

```
mySource.setCancelHandler { close(fd) }
```

```
mySource.activate()
```

# No Mutation Past Activation



Set the properties of inactive objects before activation

- Source handlers
- Target queues

```
let mySource = DispatchSource.makeReadSource(fileDescriptor: fd, queue: myQueue)
```

```
mySource.setEventHandler(qos: .userInteractive) { ... }
```

```
mySource.setCancelHandler { close(fd) }
```

```
mySource.activate()
```

```
mySource.setTarget(queue: otherQueue)
```

# Effects of Queue Graph Mutation

Priority and ownership snapshots can become stale

- Defeats priority inversion avoidance
- Defeats direct handoff optimization
- Defeats event delivery optimization

# Effects of Queue Graph Mutation

Priority and ownership snapshots can become stale

- Defeats priority inversion avoidance
- Defeats direct handoff optimization
- Defeats event delivery optimization

System frameworks may create sources on your behalf

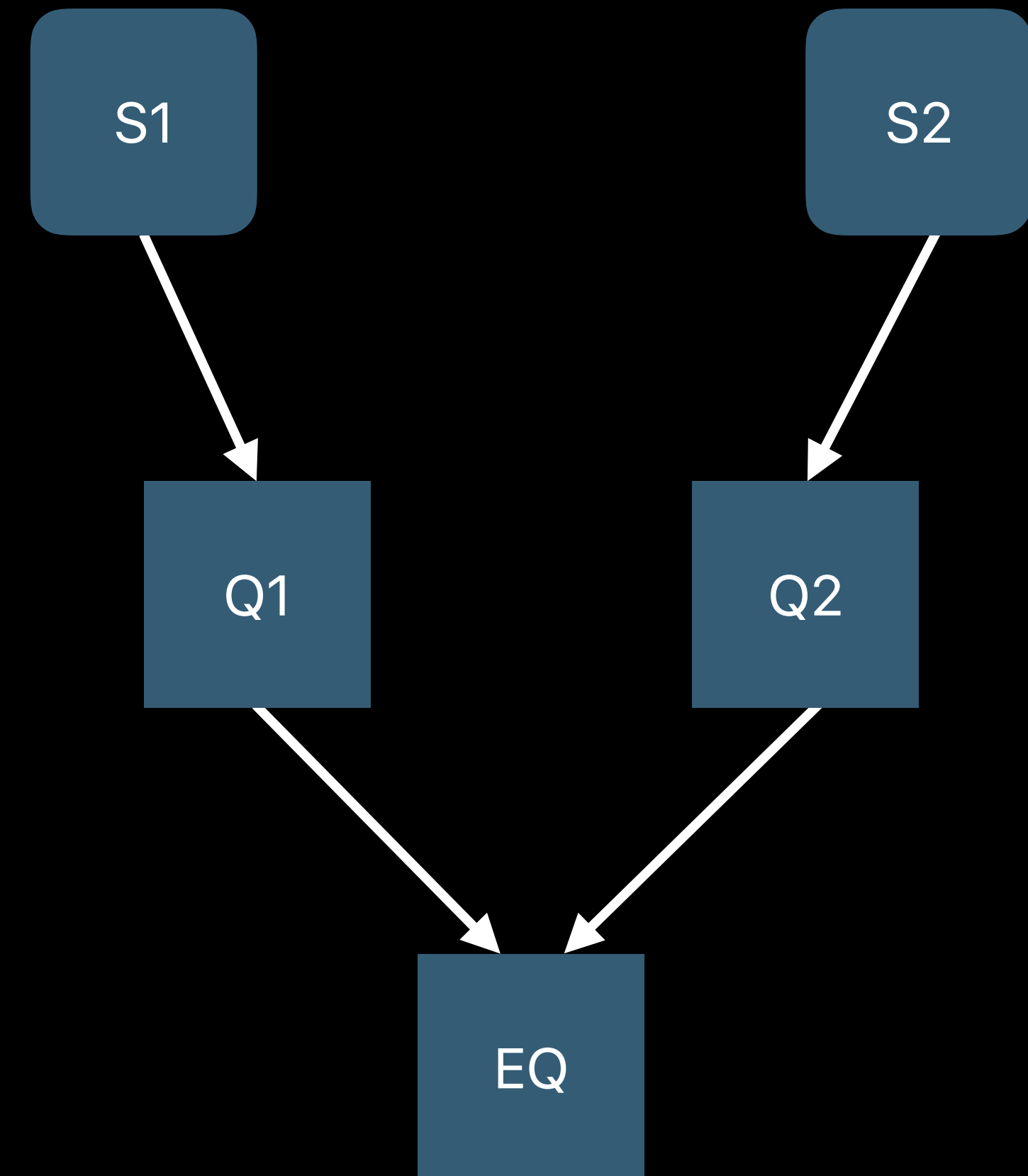
- XPC connections are like sources



# Protecting the Target Queue Hierarchy

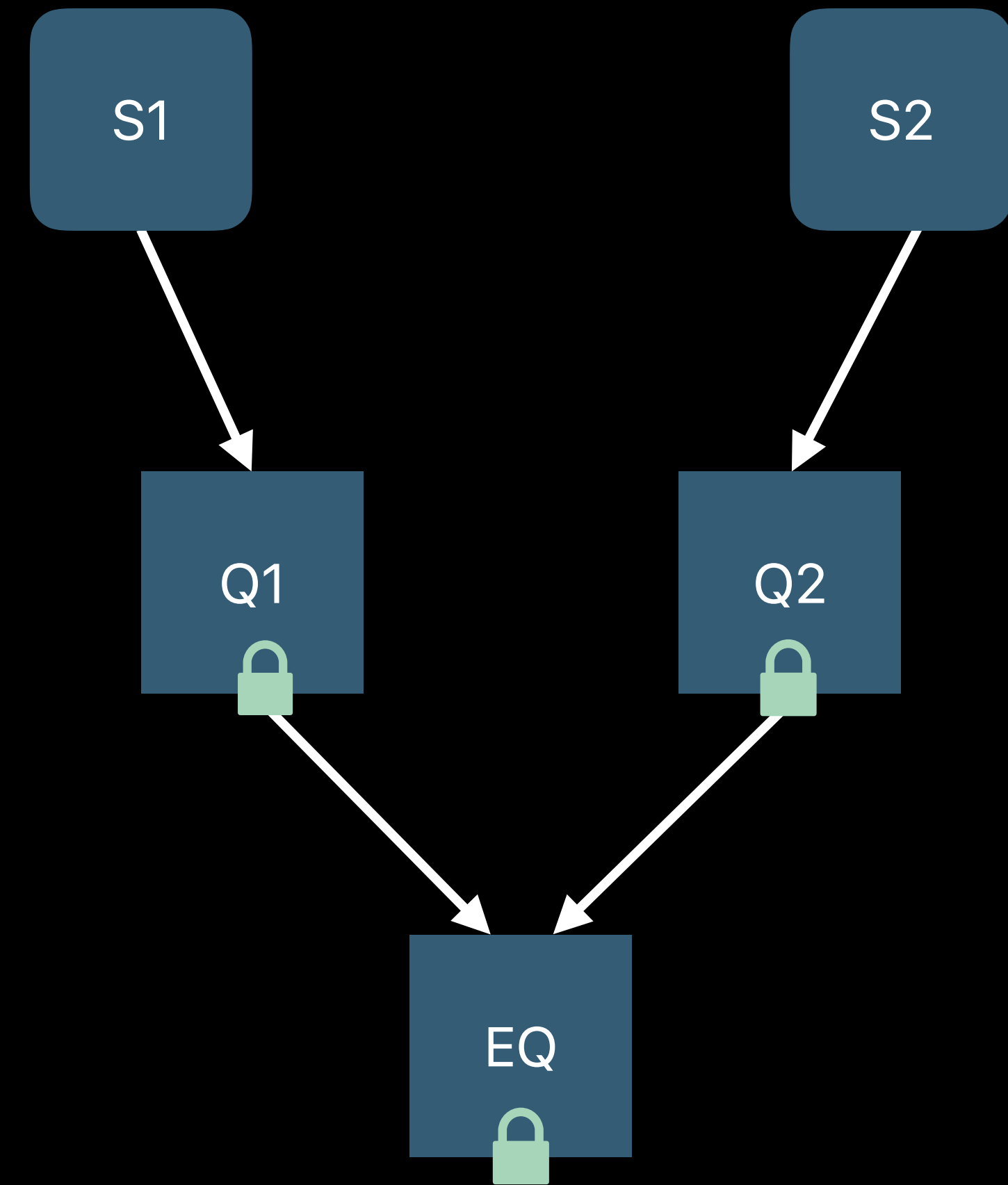
# Protecting the Target Queue Hierarchy

Build your queue hierarchy bottom to top



# Protecting the Target Queue Hierarchy

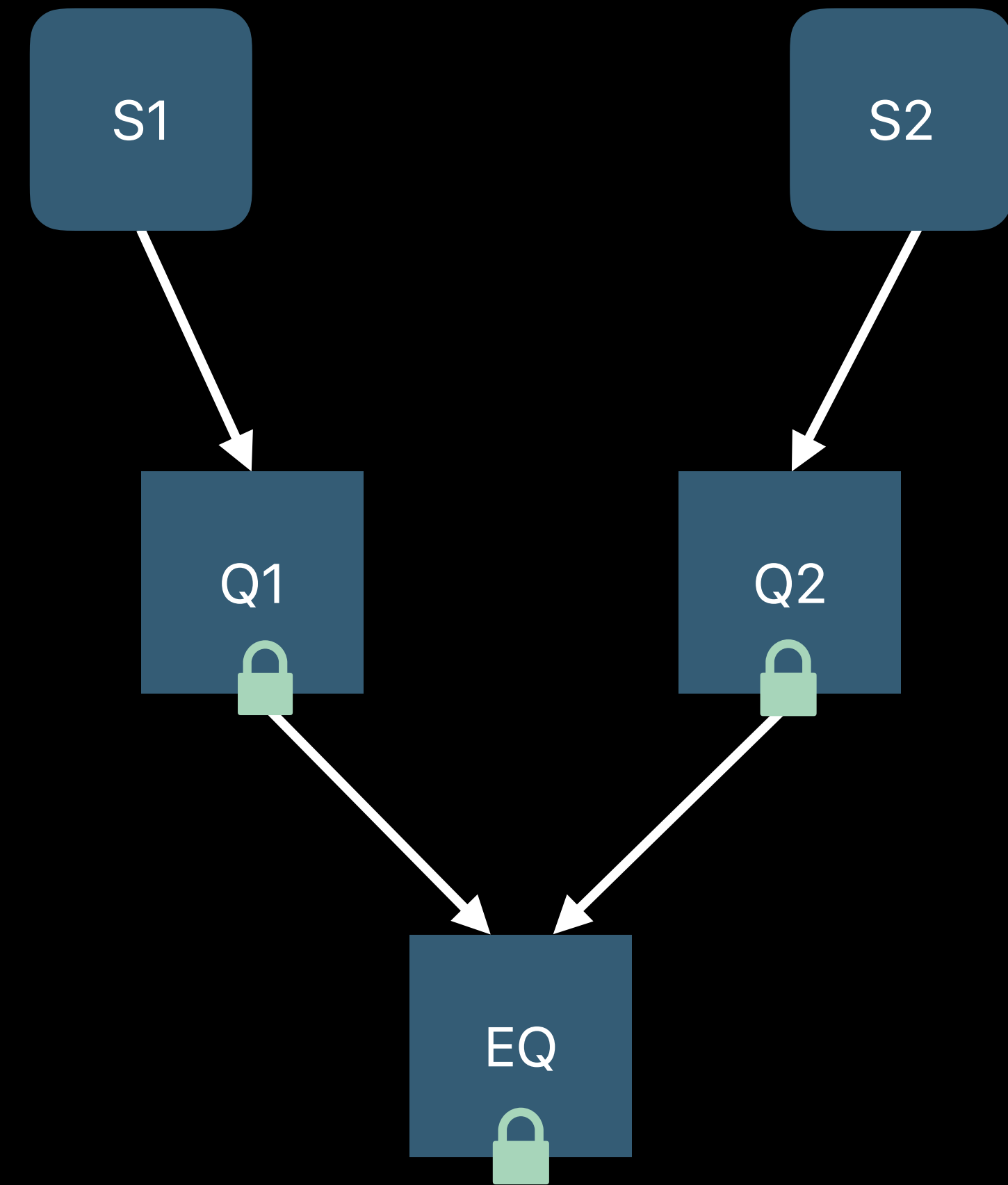
Build your queue hierarchy bottom to top



# Protecting the Target Queue Hierarchy

Build your queue hierarchy bottom to top

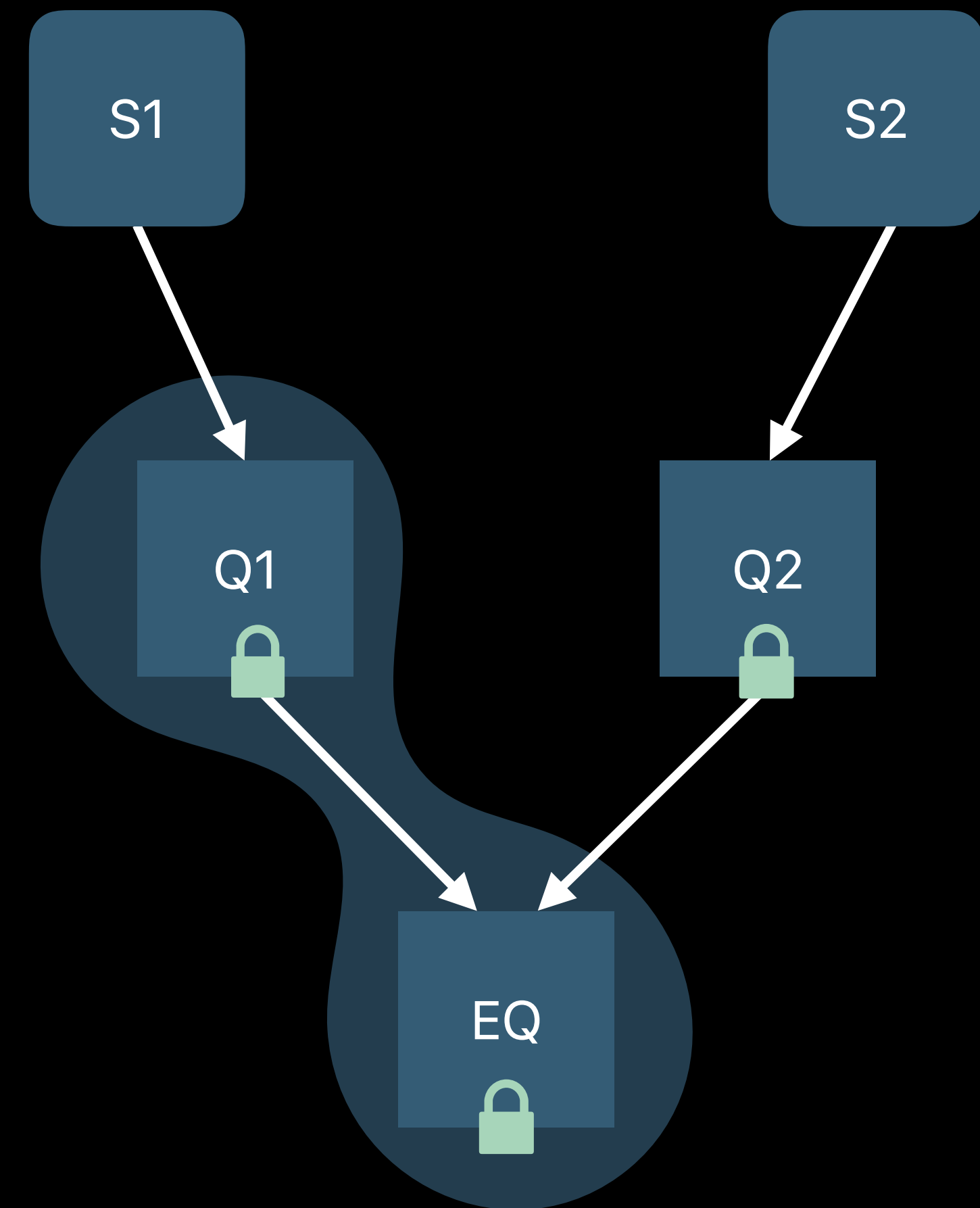
Opt into "static queue hierarchy"



# Protecting the Target Queue Hierarchy

Build your queue hierarchy bottom to top

Opt into "static queue hierarchy"

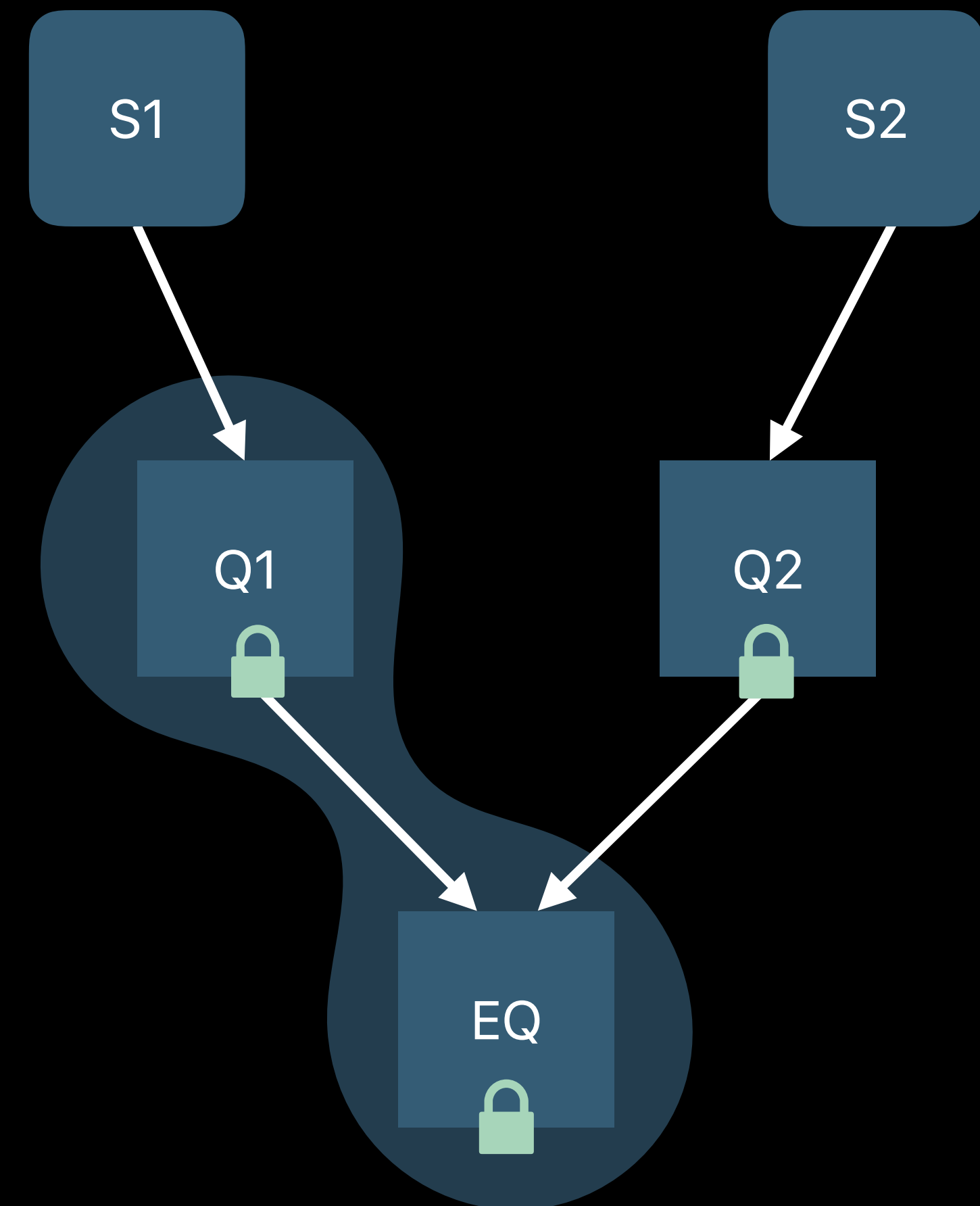


# Protecting the Target Queue Hierarchy

Build your queue hierarchy bottom to top

Opt into "static queue hierarchy"

```
Q1 = dispatch_queue_create("Q1",  
    DISPATCH_QUEUE_SERIAL)  
dispatch_set_target_queue(Q1, EQ)
```



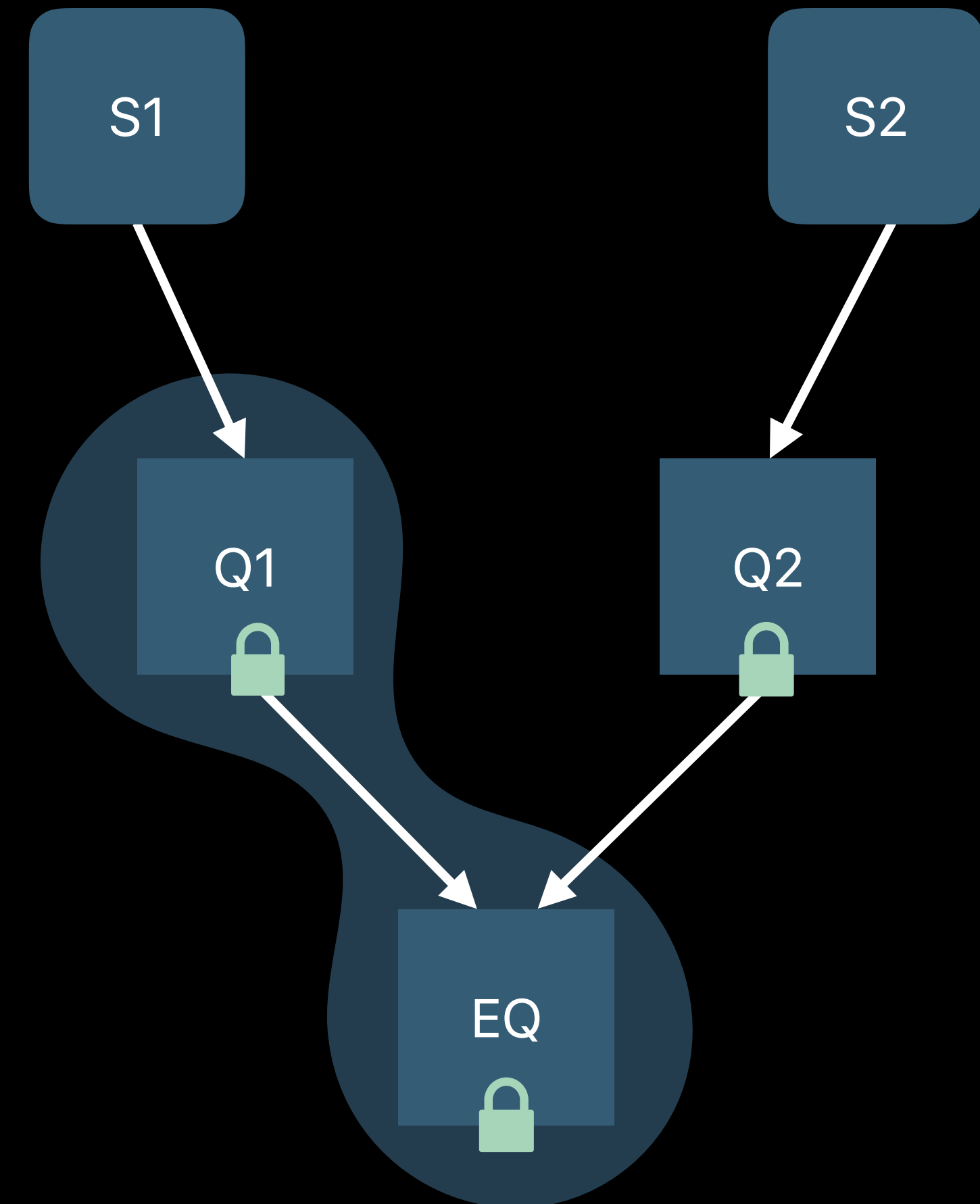
# Protecting the Target Queue Hierarchy

Build your queue hierarchy bottom to top

Opt into "static queue hierarchy"

```
Q1 = dispatch_queue_create("Q1",  
    DISPATCH_QUEUE_SERIAL)  
dispatch_set_target_queue(Q1, EQ)
```

```
Q1 = dispatch_queue_create_with_target("Q1",  
    DISPATCH_QUEUE_SERIAL, EQ)
```



***Demo***

Finding problem spots


Daniel A. Steffen, Core Darwin




MyNews

## iPad Pro

Anything you can do,  
you can do better.



### Newsroom




**PRESS RELEASE**  
JUNE 5, 2017

HomePod reinvents music in the home

Apple Developer

## Developer Insights

See how developers approach  
finding success on the App Store.



## WWDC17. Now Live Streaming

June 5, 2017

See the latest advancements in macOS, iOS, watchOS, and tvOS with over 100 sessions presented by Apple engineers at this year's conference. We're live streaming all sessions daily and posting videos and related sample code throughout the week.

[Watch WWDC session videos.](#)

Connect Refresh

```
- (void)createConnections:(int) numberOfConnections serverPort:(int)port
{
    struct sockaddr_in serverAddr = [self server];
    conns = (struct client_connection *)malloc
        (numberOfConnections * sizeof(struct client_connection));

    for (int i = 0; i < numberOfConnections; i++) {
        int sock = socket(PF_INET, SOCK_STREAM, 0);
        int ret = connect(sock, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
        assert(ret >= 0);

        int flags = fcntl(sock, F_GETFL, 0);
        fcntl(sock, F_SETFL, flags | O_NONBLOCK);

        char queue_name[1024];
        snprintf(queue_name, 1024, "com.apple.client-queue-%d", i);
        dispatch_queue_t queue = dispatch_queue_create(queue_name, DISPATCH_QUEUE_SERIAL);

        dispatch_source_t source = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, sock, 0, NULL);

        dispatch_block_t block = dispatch_block_create(DISPATCH_BLOCK_ASSIGN_CURRENT, ^{
            /* Drop the data read block start signpost */
            kdebug_signpost_start(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);

            /* Re-initialize the buffer for the connection */
```

```
- (void)createConnections:(int) numberOfConnections serverPort:(int)port
{
    struct sockaddr_in serverAddr = [self server];
    conns = (struct client_connection *)malloc
        (numberOfConnections * sizeof(struct client_connection));

    for (int i = 0; i < numberOfConnections; i++) {
        int sock = socket(PF_INET, SOCK_STREAM, 0);
        int ret = connect(sock, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
        assert(ret >= 0);

        int flags = fcntl(sock, F_GETFL, 0);
        fcntl(sock, F_SETFL, flags | O_NONBLOCK);

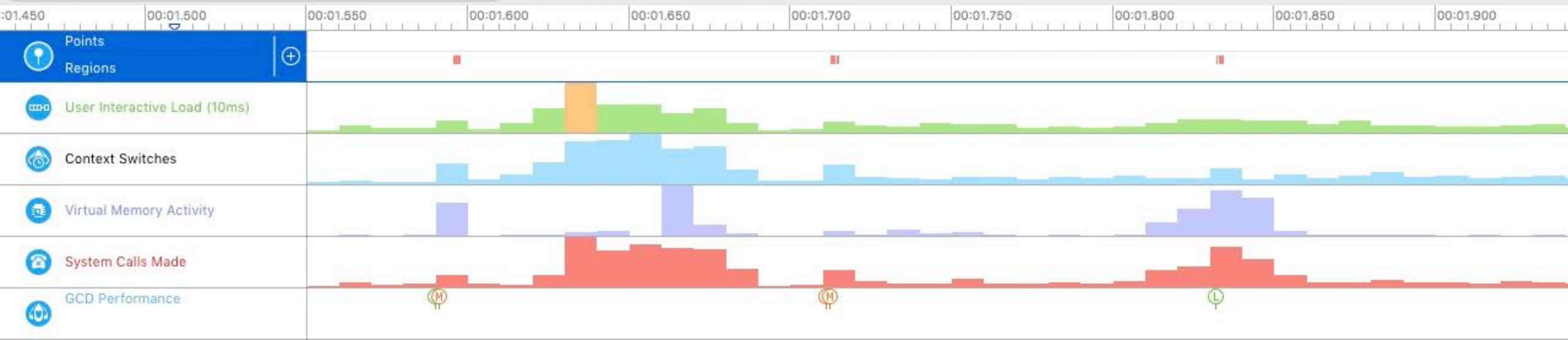
        char queue_name[1024];
        snprintf(queue_name, 1024, "com.apple.client-queue-%d", i);
        dispatch_queue_t queue = dispatch_queue_create(queue_name, DISPATCH_QUEUE_SERIAL);

        dispatch_source_t source = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, sock, 0, NULL);

        dispatch_block_t block = dispatch_block_create(DISPATCH_BLOCK_ASSIGN_CURRENT, ^{
            /* Drop the data read block start signpost */
            kdebug_signpost_start(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);

            /* Re-initialize the buffer for the connection */
```

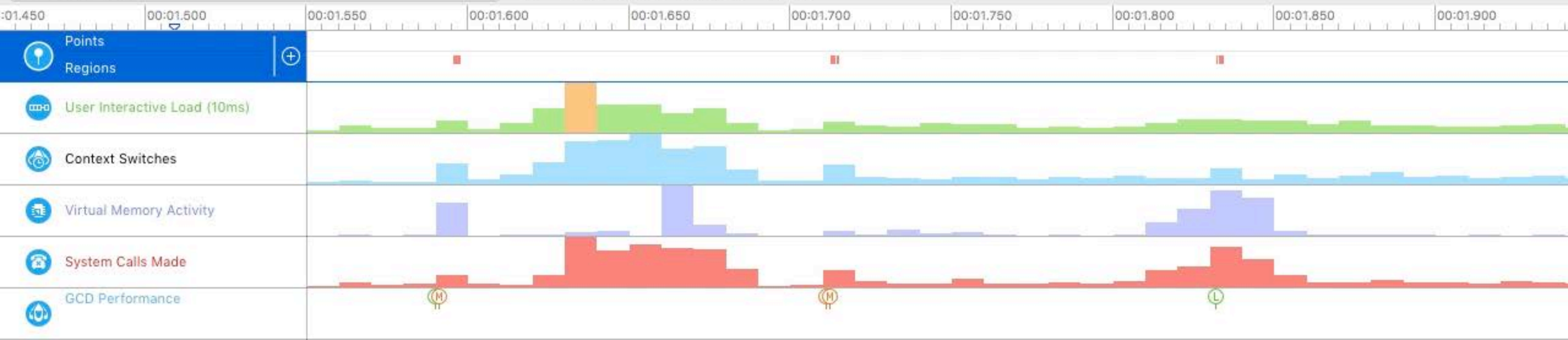
Track Filter All Instruments Threads CPUs



Points of Interest Points of Interest

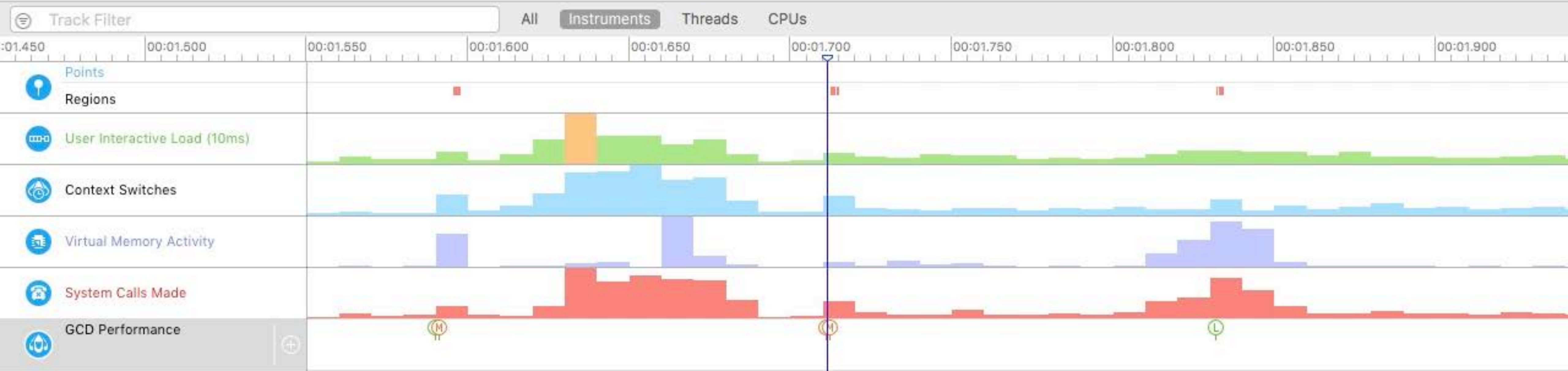
Start ^	Narrative
00:01.595.746	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:01.596.243	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:01.596.592	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.596.899	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.597.290	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.597.538	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.712.749	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.713.308	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.713.815	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.714.513	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.714.789	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.715.043	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.832.339	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.833.019	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.833.313	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)

Track Filter All Instruments Threads CPUs



Points of Interest

Start	Narrative
00:01.595.746	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:01.596.243	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:01.596.592	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.596.899	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.597.290	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.597.538	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.712.749	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.713.308	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.713.815	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.714.513	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.714.789	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.715.043	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.832.339	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.833.019	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.833.313	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)

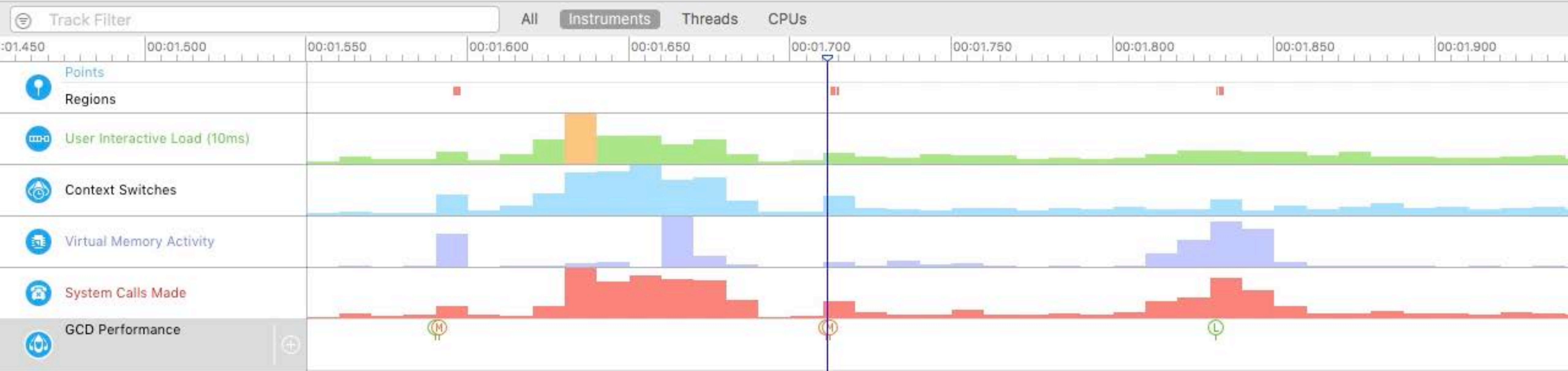


GCD Performance > GCD Performance Events > M -[ViewController createConnections:serverPort:]

Sample time ^	Dispatch Perf Event	Dispatch Queue	Dispatch Source	Old Target Queue	New Target Queue	Thread	Backtr
00:01.711.599	Mutation After Activation	0x0	0x6000001113a0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.605	Mutation After Activation	0x0	0x6000001113a0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.609	Retarget After Activation	0x6000001113a0	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.843	Mutation After Activation	0x0	0x61000010fae0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.847	Mutation After Activation	0x0	0x61000010fae0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.851	Retarget After Activation	0x61000010fae0	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.043	Mutation After Activation	0x0	0x61000010fa50	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.047	Mutation After Activation	0x0	0x61000010fa50	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.051	Retarget After Activation	0x61000010fa50	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.234	Mutation After Activation	0x0	0x61800010ed60	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.267	Mutation After Activation	0x0	0x61800010ed60	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.273	Retarget After Activation	0x61800010ed60	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.515	Mutation After Activation	0x0	0x61800010edf0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.521	Mutation After Activation	0x0	0x61800010edf0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.527	Retarget After Activation	0x61800010edf0	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb

Backtrace

- \_\_kdebug\_trace64
- \_dispatch\_queue\_set\_target\_queue
- [ViewController createConnections:serverPort:]**
- [ViewController connectButton:]
- [NSApplication(NSResponder) sendAction:to:from:]
- [NSControl sendAction:to:]
- \_26-[NSCell \_sendActionFrom:]\_block\_invoke
- [NSCell \_sendActionFrom:]
- [NSButtonCell \_sendActionFrom:]
- [NSCell trackMouse:inRect:ofView:untilMouseUp:]
- [NSButtonCell trackMouse:inRect:ofView:untilMous...
- [NSControl mouseDown:]
- [NSWindow(NSEventRouting) \_handleMouseDownE...
- [NSWindow(NSEventRouting) \_reallySendEvent:isD...
- [NSWindow(NSEventRouting) sendEvent:]
- [NSApplication(NSEvent) sendEvent:]
- [NSApplication run]



GCD Performance > GCD Performance Events > M -[ViewController createConnections:serverPort:]

Sample time ^	Dispatch Perf Event	Dispatch Queue	Dispatch Source	Old Target Queue	New Target Queue	Thread	Backtr
00:01.711.599	Mutation After Activation	0x0	0x6000001113a0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.605	Mutation After Activation	0x0	0x6000001113a0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.609	Retarget After Activation	0x6000001113a0	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.843	Mutation After Activation	0x0	0x61000010fae0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.847	Mutation After Activation	0x0	0x61000010fae0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.711.851	Retarget After Activation	0x61000010fae0	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.043	Mutation After Activation	0x0	0x61000010fa50	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.047	Mutation After Activation	0x0	0x61000010fa50	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.051	Retarget After Activation	0x61000010fa50	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.234	Mutation After Activation	0x0	0x61800010ed60	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.267	Mutation After Activation	0x0	0x61800010ed60	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.273	Retarget After Activation	0x61800010ed60	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.515	Mutation After Activation	0x0	0x61800010edf0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.521	Mutation After Activation	0x0	0x61800010edf0	0x0	0x0	Main Thread 0x314fe7	__kdeb
00:01.712.527	Retarget After Activation	0x61800010edf0	0x0	0x0	0x0	Main Thread 0x314fe7	__kdeb

Backtrace

- \_\_kdebug\_trace64
- \_dispatch\_queue\_set\_target\_queue
- [ViewController createConnections:serverPort:]**
- [ViewController connectButton:]
- [NSApplication(NSResponder) sendAction:to:from:]
- [NSControl sendAction:to:]
- \_\_26-[NSCell \_sendActionFrom:]\_block\_invoke
- [NSCell \_sendActionFrom:]
- [NSButtonCell \_sendActionFrom:]
- [NSCell trackMouse:inRect:ofView:untilMouseUp:]
- [NSButtonCell trackMouse:inRect:ofView:untilMous...
- [NSControl mouseDown:]
- [NSWindow(NSEventRouting) \_handleMouseDownE...
- [NSWindow(NSEventRouting) \_reallySendEvent:isD...
- [NSWindow(NSEventRouting) sendEvent:]
- [NSApplication(NSEvent) sendEvent:]
- [NSApplication run]

```
        if (err == 0 || (err < 0 && errno != EAGAIN && errno != EINTR)) {
            dispatch_source_cancel(source);
            break;
        }
        if (err < 0 && errno == EAGAIN) {
            break;
        }
        conns[i].index += err;
    }

    /* Add URL to global Set */
    [self processURL:conns[i].buffer];

    /* Drop the data read block end signpost */
    kdebug_signpost_end(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);
});

dispatch_activate(source);

dispatch_source_set_event_handler(source, block);
dispatch_source_set_cancel_handler(source, ^{
    close(sock);
});

dispatch_set_target_queue(source, queue);
}
}
```



```
        if (err == 0 || (err < 0 && errno != EAGAIN && errno != EINTR)) {
            dispatch_source_cancel(source);
            break;
        }
        if (err < 0 && errno == EAGAIN) {
            break;
        }
        conns[i].index += err;
    }

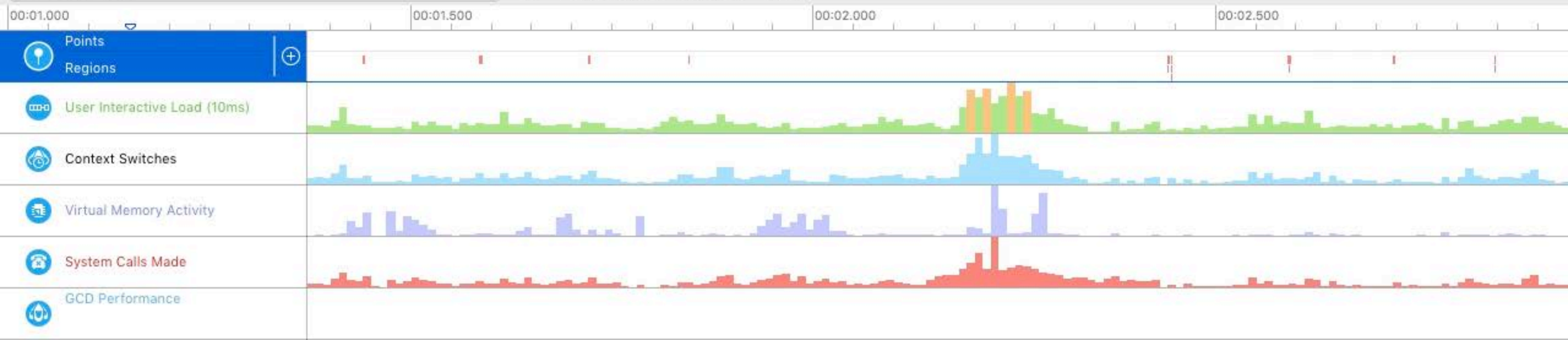
    /* Add URL to global Set */
    [self processURL:conns[i].buffer];

    /* Drop the data read block end signpost */
    kdebug_signpost_end(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);
});

dispatch_activate(source);

dispatch_source_set_event_handler(source, block);
dispatch_source_set_cancel_handler(source, ^{
    close(sock);
});

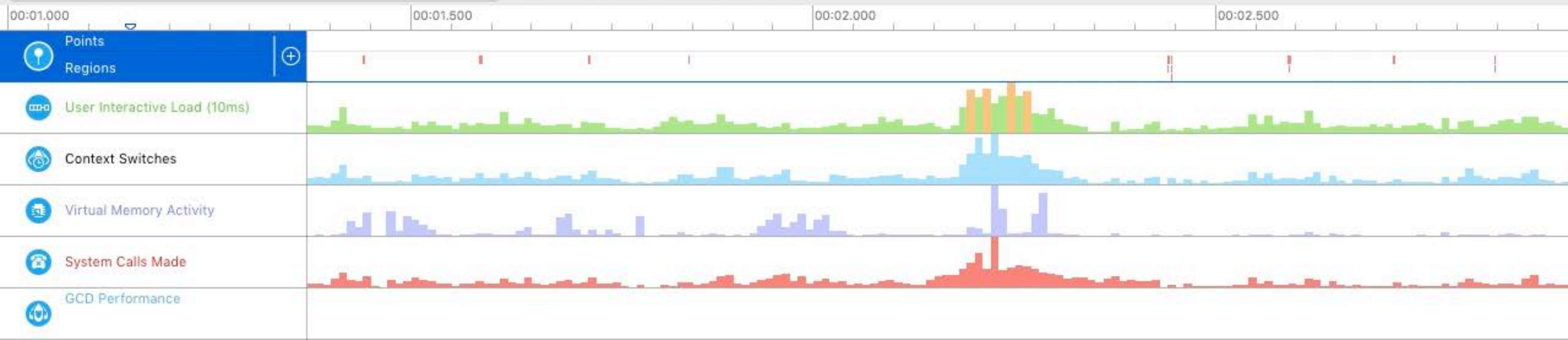
dispatch_set_target_queue(source, queue);
}
}
```



Points of Interest | Points of Interest

Start	Narrative
00:01.440.951	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.441.162	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.441.344	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.441.483	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.441.677	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.442.259	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.442.462	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.442.625	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.585.505	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.585.908	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.586.225	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.586.569	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.586.885	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)
00:01.587.129	Network Event Handler Start: (0xf 0x32 0x0 0x0), End: (0xf 0x32 0x0 0x0)
00:01.587.398	Network Event Handler Start: (0x10 0x33 0x0 0x0), End: (0x10 0x33 0x0 0x0)

Track Filter All Instruments Threads CPUs



Points of Interest Points of Interest

Start	Narrative
00:01.440.951	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.441.162	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.441.344	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.441.483	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.441.677	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.442.259	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.442.462	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.442.625	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.585.505	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.585.908	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.586.225	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.586.569	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.586.885	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)
00:01.587.129	Network Event Handler Start: (0xf 0x32 0x0 0x0), End: (0xf 0x32 0x0 0x0)
00:01.587.398	Network Event Handler Start: (0x10 0x33 0x0 0x0), End: (0x10 0x33 0x0 0x0)

```
- (void)createConnections:(int) numberOfConnections serverPort:(int)port
{
    struct sockaddr_in serverAddr = [self server];
    conns = (struct client_connection *)malloc
        (numberOfConnections * sizeof(struct client_connection));

    for (int i = 0; i < numberOfConnections; i++) {
        int sock = socket(PF_INET, SOCK_STREAM, 0);
        int ret = connect(sock, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
        assert(ret >= 0);

        int flags = fcntl(sock, F_GETFL, 0);
        fcntl(sock, F_SETFL, flags | O_NONBLOCK);

        char queue_name[1024];
        snprintf(queue_name, 1024, "com.apple.client-queue-%d", i);
        dispatch_queue_t queue = dispatch_queue_create(queue_name, DISPATCH_QUEUE_SERIAL);

        dispatch_source_t source = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, sock, 0, NULL);

        dispatch_block_t block = dispatch_block_create(DISPATCH_BLOCK_ASSIGN_CURRENT, ^{
            /* Drop the data read block start signpost */
            kdebug_signpost_start(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);

            /* Re-initialize the buffer for the connection */
```

```
- (void)createConnections:(int) numberOfConnections serverPort:(int)port
{
    struct sockaddr_in serverAddr = [self server];
    conns = (struct client_connection *)malloc
        (numberOfConnections * sizeof(struct client_connection));

    for (int i = 0; i < numberOfConnections; i++) {
        int sock = socket(PF_INET, SOCK_STREAM, 0);
        int ret = connect(sock, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
        assert(ret >= 0);

        int flags = fcntl(sock, F_GETFL, 0);
        fcntl(sock, F_SETFL, flags | O_NONBLOCK);

        char queue_name[1024];
        snprintf(queue_name, 1024, "com.apple.client-queue-%d", i);
        dispatch_queue_t queue = dispatch_queue_create(queue_name, DISPATCH_QUEUE_SERIAL);

        dispatch_source_t source = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, sock, 0, NULL);

        dispatch_block_t block = dispatch_block_create(DISPATCH_BLOCK_ASSIGN_CURRENT, ^{
            /* Drop the data read block start signpost */
            kdebug_signpost_start(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);

            /* Re-initialize the buffer for the connection */
```

Track Filter

All Instruments Threads CPUs



Points of Interest > Points of Interest

Start	Narrative
00:01.440.560	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:01.440.746	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:01.440.951	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.441.162	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.441.344	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.441.483	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.441.677	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.442.259	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.442.462	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.442.625	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.585.505	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.585.908	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.586.225	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.586.569	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.586.885	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)

**Stack**

- \_kdebug\_trace64
- \_47-[ViewController createConnections:serverPort:]...
- \_dispatch\_block\_async\_invoke
- \_dispatch\_client\_callout
- \_dispatch\_continuation\_pop
- \_dispatch\_source\_invoke
- \_dispatch\_queue\_serial\_drain
- \_dispatch\_queue\_invoke
- \_dispatch\_root\_queue\_drain\_deferred\_wlh
- \_dispatch\_workloop\_worker\_thread
- \_pthread\_wqthread
- start\_wqthread

Track Filter

All Instruments Threads CPUs

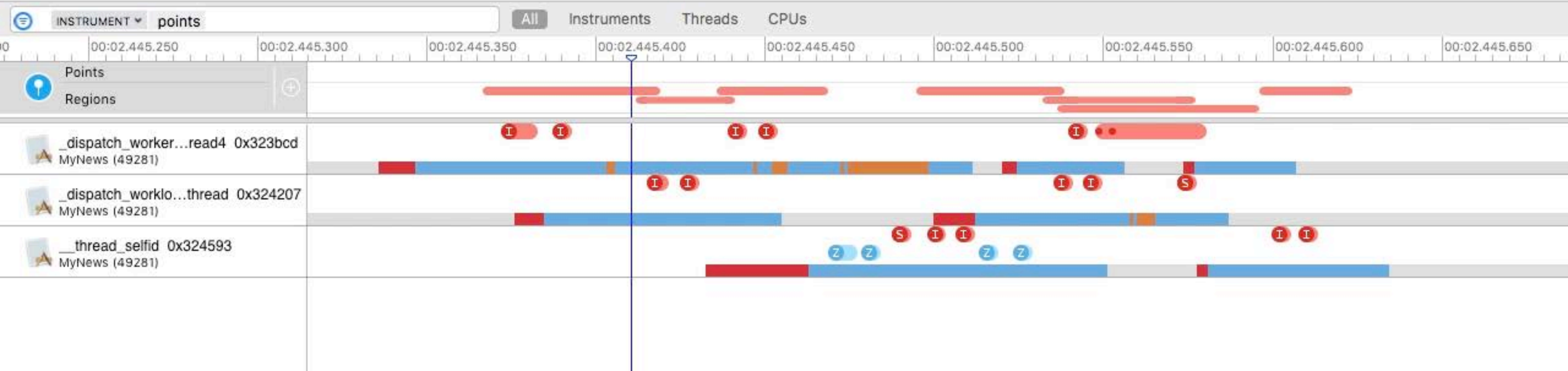


Points of Interest > Points of Interest

Start	Narrative
00:01.440.560	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:01.440.746	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:01.440.951	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.441.162	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.441.344	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.441.483	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.441.677	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.442.259	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.442.462	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.442.625	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.585.505	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.585.908	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.586.225	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.586.569	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.586.885	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)

**Stack**

- \_kdebug\_trace64
- \_47-[ViewController createConnections:serverPort:]...
- \_dispatch\_block\_async\_invoke
- \_dispatch\_client\_callout
- \_dispatch\_continuation\_pop
- \_dispatch\_source\_invoke
- \_dispatch\_queue\_serial\_drain
- \_dispatch\_queue\_invoke
- \_dispatch\_root\_queue\_drain\_deferred\_wlh
- \_dispatch\_workloop\_worker\_thread
- \_pthread\_wqthread
- start\_wqthread

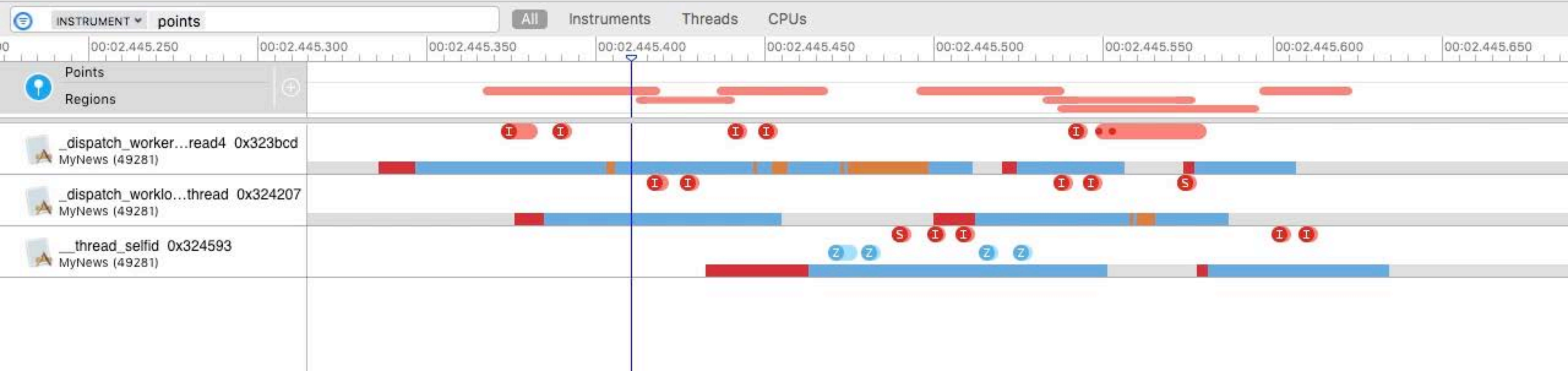


Points of Interest

Start	Narrative
00:01.845.476	Network Event Handler Start: (0x1f 0x42 0x0 0x0), End: (0x1f 0x42 0x0 0x0)
00:02.441.669	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:02.441.691	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:02.441.791	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:02.445.366	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:02.445.411	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:02.445.435	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:02.445.494	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:02.445.532	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:02.445.536	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:02.445.596	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:02.590.168	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:02.590.215	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:02.590.242	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:02.591.024	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)

No Detail





Points of Interest

Start	Narrative
00:01.845.476	Network Event Handler Start: (0x1f 0x42 0x0 0x0), End: (0x1f 0x42 0x0 0x0)
00:02.441.669	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:02.441.691	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:02.441.791	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:02.445.366	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:02.445.411	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:02.445.435	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:02.445.494	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:02.445.532	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:02.445.536	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:02.445.596	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:02.590.168	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:02.590.215	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:02.590.242	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:02.591.024	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)

```
        if (err == 0 || (err < 0 && errno != EAGAIN && errno != EINTR)) {
            dispatch_source_cancel(source);
            break;
        }
        if (err < 0 && errno == EAGAIN) {
            break;
        }
        conns[i].index += err;
    }

    /* Add URL to global Set */
    [self processURL:conns[i].buffer];

    /* Drop the data read block end signpost */
    kdebug_signpost_end(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);
});

dispatch_source_set_event_handler(source, block);
dispatch_source_set_cancel_handler(source, ^{
    close(sock);
});
dispatch_set_target_queue(source, queue);

dispatch_activate(source);
}
}
```

```
        if (err == 0 || (err < 0 && errno != EAGAIN && errno != EINTR)) {
            dispatch_source_cancel(source);
            break;
        }
        if (err < 0 && errno == EAGAIN) {
            break;
        }
        conns[i].index += err;
    }

    /* Add URL to global Set */
    [self processURL:conns[i].buffer];

    /* Drop the data read block end signpost */
    kdebug_signpost_end(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);
});

dispatch_source_set_event_handler(source, block);
dispatch_source_set_cancel_handler(source, ^{
    close(sock);
});
dispatch_set_target_queue(source, queue);

dispatch_activate(source);
}
}
```

```
- (void)createConnections:(int) numberOfConnections serverPort:(int)port
{
    struct sockaddr_in serverAddr = [self server];
    conns = (struct client_connection *)malloc
        (numberOfConnections * sizeof(struct client_connection));

    for (int i = 0; i < numberOfConnections; i++) {
        int sock = socket(PF_INET, SOCK_STREAM, 0);
        int ret = connect(sock, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
        assert(ret >= 0);

        int flags = fcntl(sock, F_GETFL, 0);
        fcntl(sock, F_SETFL, flags | O_NONBLOCK);

        char queue_name[1024];
        snprintf(queue_name, 1024, "com.apple.client-queue-%d", i);
        dispatch_queue_t queue = dispatch_queue_create(queue_name, DISPATCH_QUEUE_SERIAL);

        dispatch_source_t source = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, sock, 0, NULL);

        dispatch_block_t block = dispatch_block_create(DISPATCH_BLOCK_ASSIGN_CURRENT, ^{
            /* Drop the data read block start signpost */
            kdebug_signpost_start(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);

            /* Re-initialize the buffer for the connection */
            bzero(&conns[i].buffer, CONNECTION_BUFFER_SIZE);
```

```
- (void)createConnections:(int) numberOfConnections serverPort:(int)port
{
    struct sockaddr_in serverAddr = [self server];
    conns = (struct client_connection *)malloc
        (numberOfConnections * sizeof(struct client_connection));

    for (int i = 0; i < numberOfConnections; i++) {
        int sock = socket(PF_INET, SOCK_STREAM, 0);
        int ret = connect(sock, (struct sockaddr *) &serverAddr, sizeof(serverAddr));
        assert(ret >= 0);

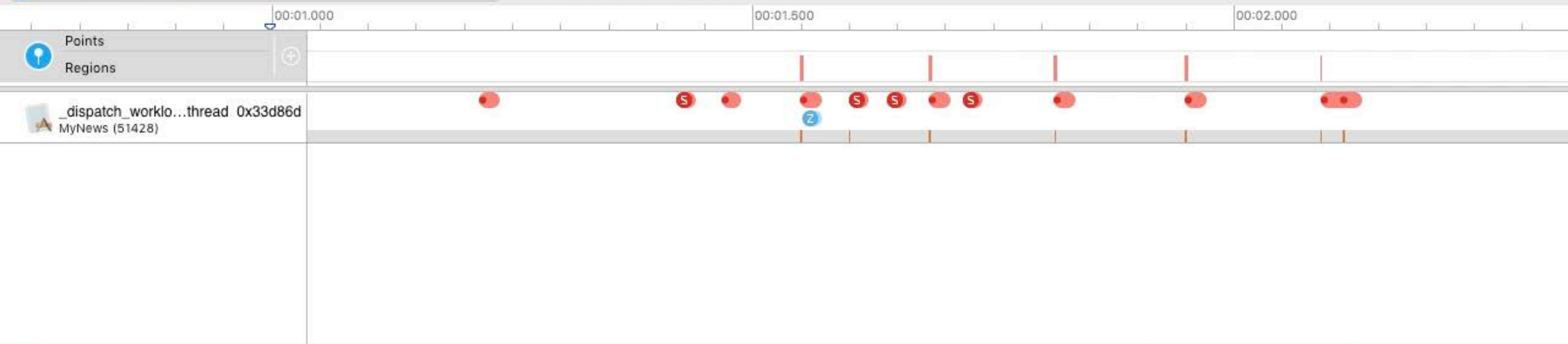
        int flags = fcntl(sock, F_GETFL, 0);
        fcntl(sock, F_SETFL, flags | O_NONBLOCK);

        char queue_name[1024];
        snprintf(queue_name, 1024, "com.apple.client-queue-%d", i);
        dispatch_queue_t queue = dispatch_queue_create(queue_name, DISPATCH_QUEUE_SERIAL);

        dispatch_source_t source = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, sock, 0, NULL);

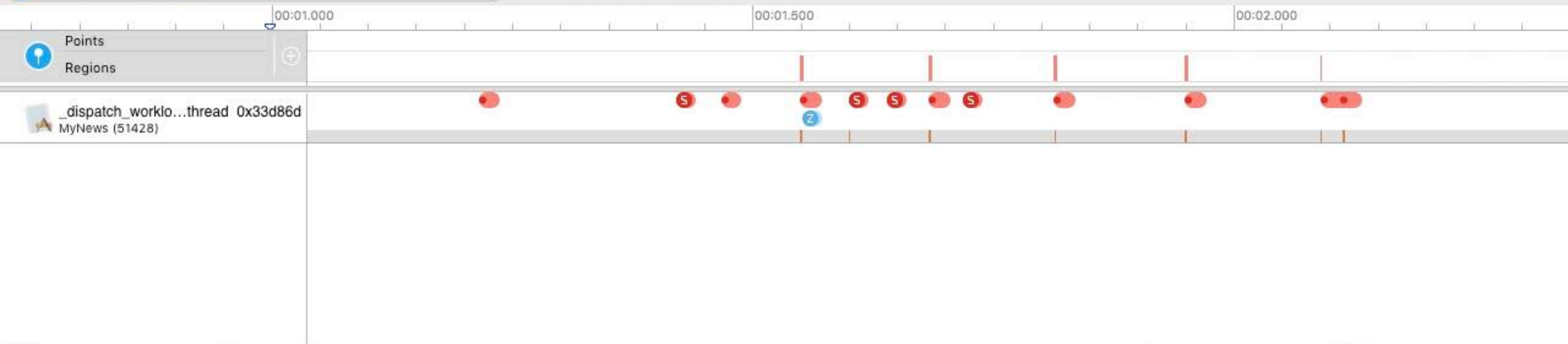
        dispatch_block_t block = dispatch_block_create(DISPATCH_BLOCK_ASSIGN_CURRENT, ^{
            /* Drop the data read block start signpost */
            kdebug_signpost_start(MYNEWS_CONN_DATA_RECV, i, sock, 0, 0);

            /* Re-initialize the buffer for the connection */
            bzero(&conns[i].buffer, CONNECTION_BUFFER_SIZE);
```



Points of Interest Points of Interest

Start ^	Narrative
00:01.549.802	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:01.550.582	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:01.551.009	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.551.540	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.551.880	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.552.190	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.682.566	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.683.244	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.683.676	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.683.992	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.684.437	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.684.797	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.685.122	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.812.895	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.813.342	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)



Points of Interest Points of Interest

Start ^	Narrative
00:01.549.802	Network Event Handler Start: (0x0 0x23 0x0 0x0), End: (0x0 0x23 0x0 0x0)
00:01.550.582	Network Event Handler Start: (0x1 0x24 0x0 0x0), End: (0x1 0x24 0x0 0x0)
00:01.551.009	Network Event Handler Start: (0x2 0x25 0x0 0x0), End: (0x2 0x25 0x0 0x0)
00:01.551.540	Network Event Handler Start: (0x3 0x26 0x0 0x0), End: (0x3 0x26 0x0 0x0)
00:01.551.880	Network Event Handler Start: (0x4 0x27 0x0 0x0), End: (0x4 0x27 0x0 0x0)
00:01.552.190	Network Event Handler Start: (0x5 0x28 0x0 0x0), End: (0x5 0x28 0x0 0x0)
00:01.682.566	Network Event Handler Start: (0x6 0x29 0x0 0x0), End: (0x6 0x29 0x0 0x0)
00:01.683.244	Network Event Handler Start: (0x7 0x2a 0x0 0x0), End: (0x7 0x2a 0x0 0x0)
00:01.683.676	Network Event Handler Start: (0x8 0x2b 0x0 0x0), End: (0x8 0x2b 0x0 0x0)
00:01.683.992	Network Event Handler Start: (0x9 0x2c 0x0 0x0), End: (0x9 0x2c 0x0 0x0)
00:01.684.437	Network Event Handler Start: (0xa 0x2d 0x0 0x0), End: (0xa 0x2d 0x0 0x0)
00:01.684.797	Network Event Handler Start: (0xb 0x2e 0x0 0x0), End: (0xb 0x2e 0x0 0x0)
00:01.685.122	Network Event Handler Start: (0xc 0x2f 0x0 0x0), End: (0xc 0x2f 0x0 0x0)
00:01.812.895	Network Event Handler Start: (0xd 0x30 0x0 0x0), End: (0xd 0x30 0x0 0x0)
00:01.813.342	Network Event Handler Start: (0xe 0x31 0x0 0x0), End: (0xe 0x31 0x0 0x0)

***Demo***

Finding problem spots

Daniel A. Steffen, Core Darwin



# Summary

Not going off-core is ever more important

Size your work appropriately

Choose good granularity of concurrency

Modernize your GCD usage

Use tools to find problem spots

# More Information

<https://developer.apple.com/wwdc17/706>

# Related Sessions

---

Introducing Core ML

WWDC 2017

---

Accelerate and Sparse Solvers

Grand Ballroom A

Thursday 10:00AM

---

Using Metal 2 for Compute

Grand Ballroom A

Thursday 4:10PM

---

Writing Energy Efficient Apps

Executive Ballroom

Friday 9:00AM

---

App Startup Time: Past, Present, and Future

Hall 2

Friday 10:00AM

---

# Labs

---

Kernel & Runtime Lab

Technology Lab D

Wed 1:50PM–4:10PM

---

Kernel & Runtime Lab

Technology Lab J

Thu 10:00AM–12:00PM

---

Performance Profiling and Runtime Analysis Tools Lab

Technology Lab K

Thu 1:00PM–4:10PM

---

Optimizing App Startup Time Lab

Technology Lab E

Fri 11:00AM–12:30PM

---

