

Power, Performance, and Diagnostics

What's new in GCD and XPC

Session 716

Daniel Steffen

Darwin Runtime Engineer

Overview

Background

Quality of Service Classes

New QoS and GCD API

Propagation of QoS and Execution Context

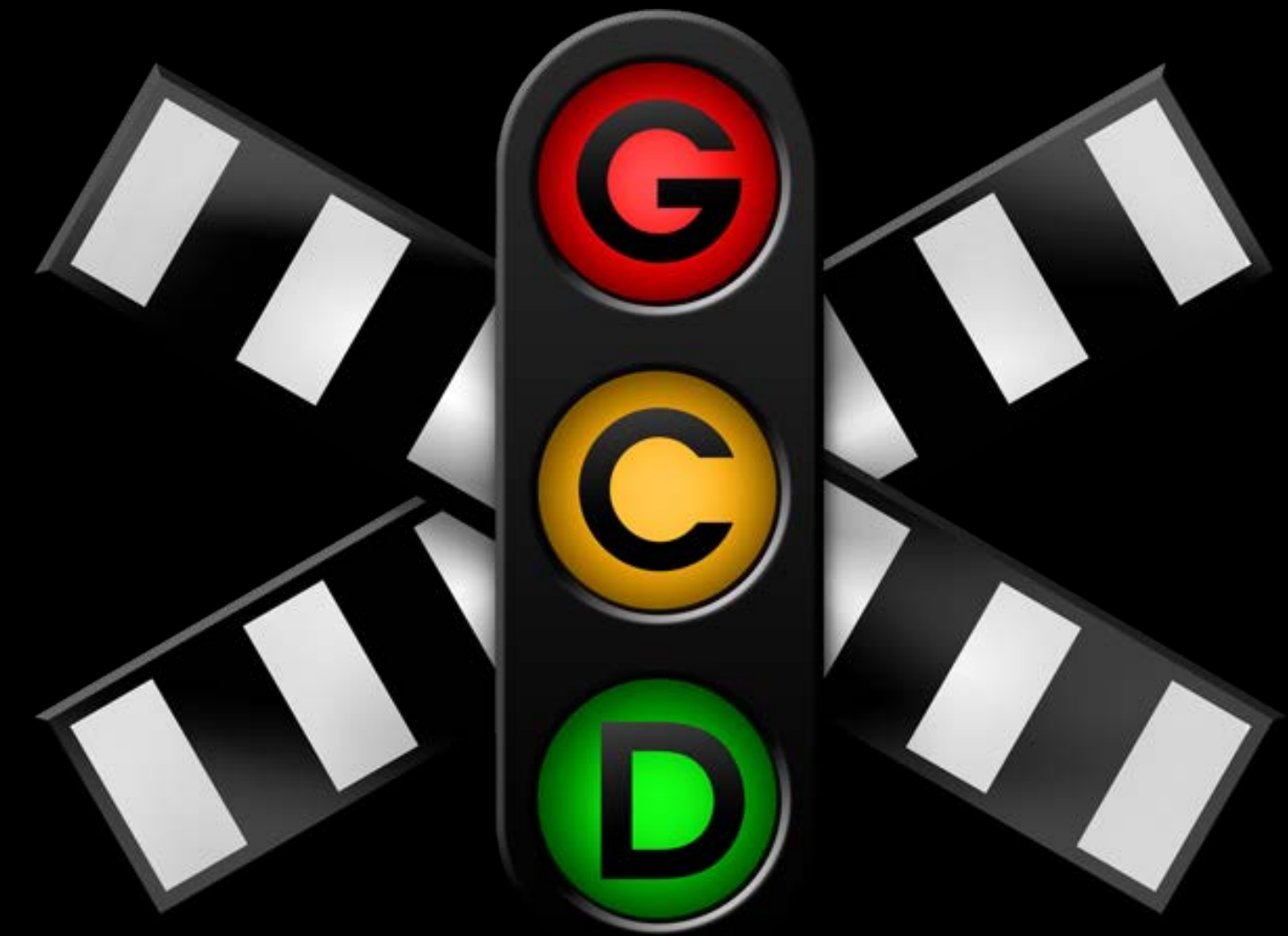
Diagnostics and Queue Debugging

Grand Central Dispatch

Asynchronous execution

Concurrent execution

Synchronization

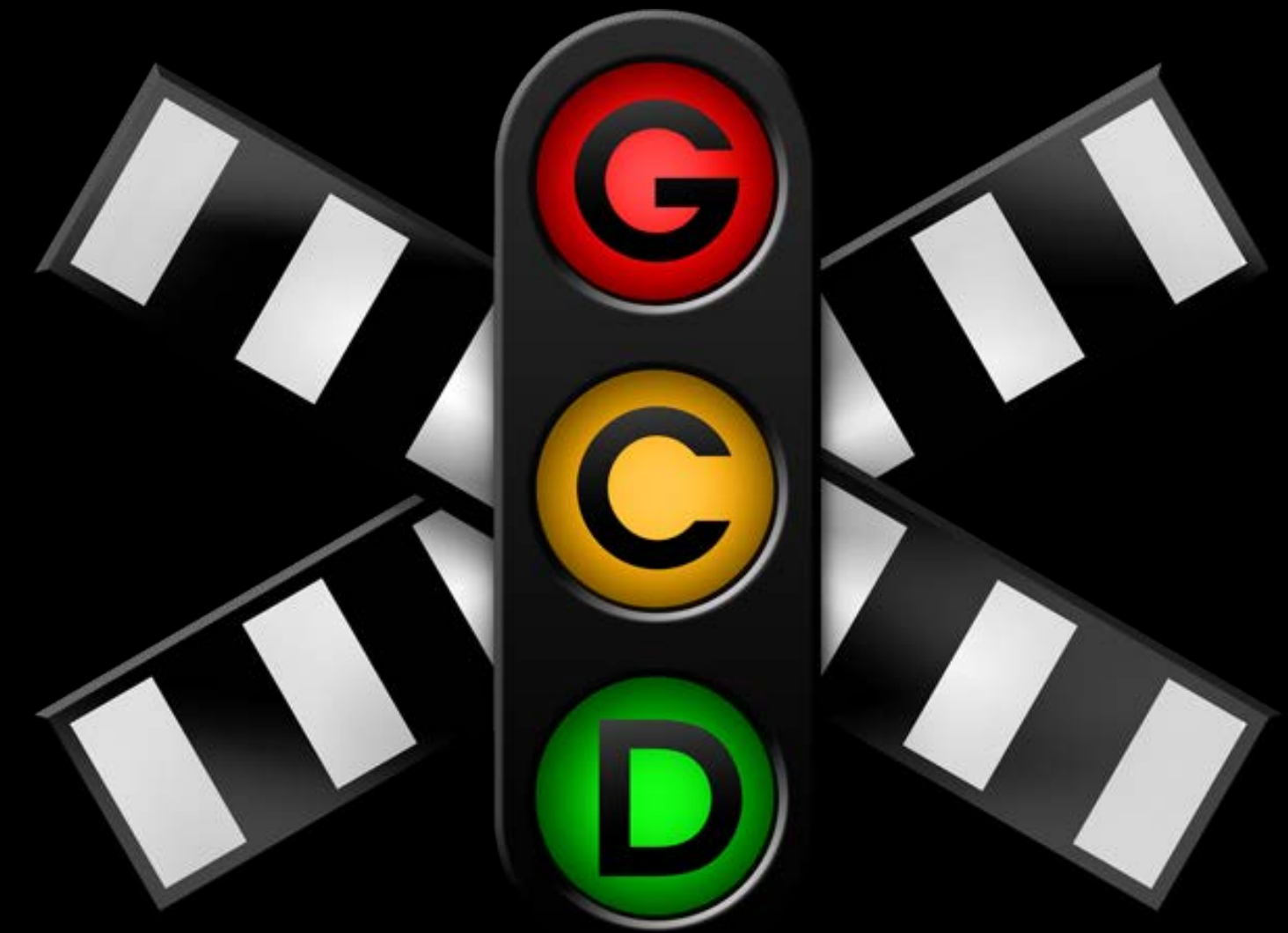


Asynchronous Execution

Asynchronous Execution

GCD

Run code in separate environment
in same process

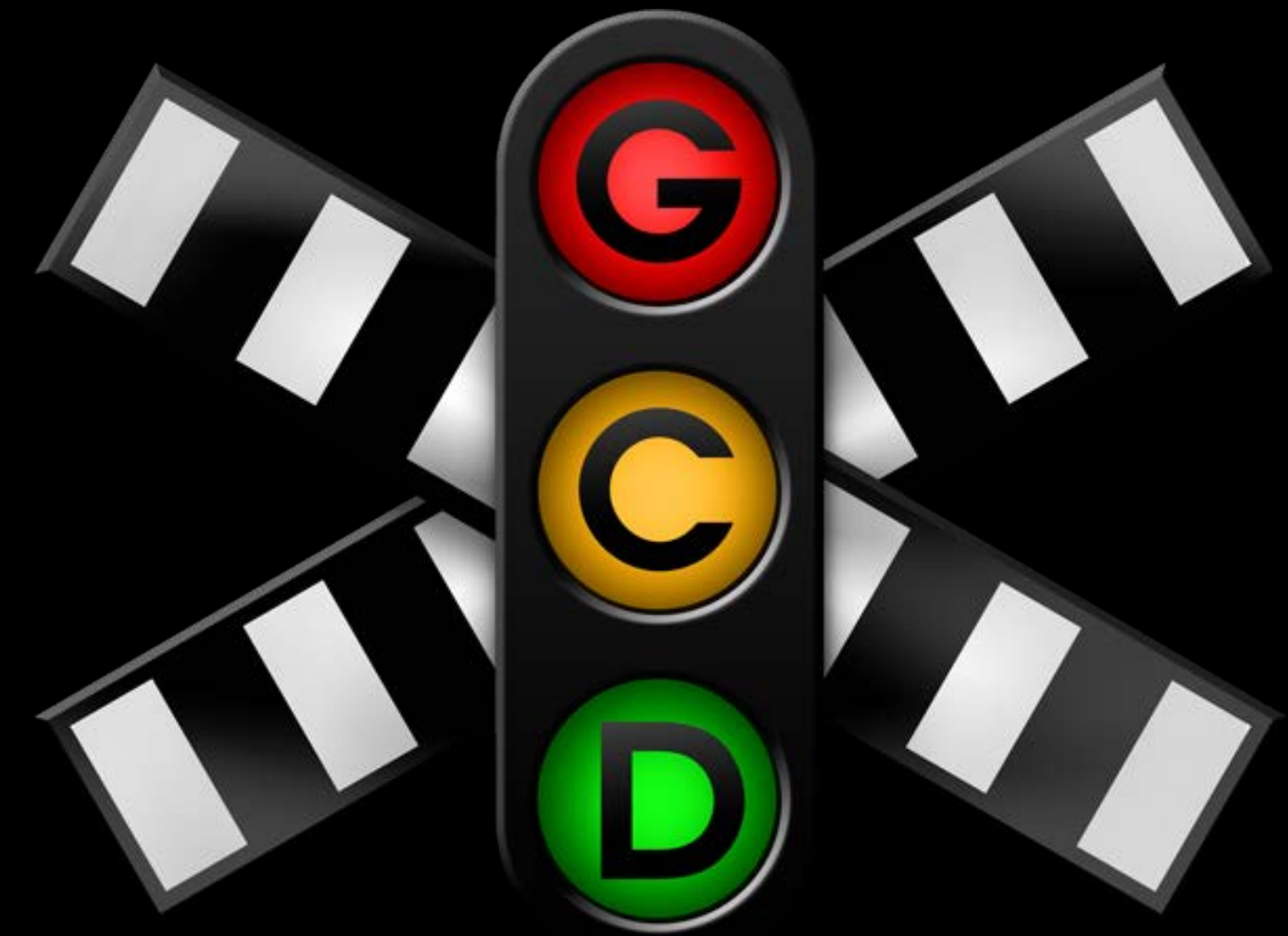


Asynchronous Execution

GCD

Run code in separate environment
in same process

- Avoid interfering with current thread
- Different priority level
- Coordination between multiple clients



Asynchronous Execution

XPC

Run code in separate process



Asynchronous Execution

XPC

Run code in separate process

- Avoid interfering with current process
- Different privilege level
- Coordination between multiple clients



Previously...

On developer.apple.com

-
- [Introducing Blocks and Grand Central Dispatch on iPhone](#) WWDC 2010

 - [Simplifying iPhone App Development with Grand Central Dispatch](#) WWDC 2010

 - [Introducing XPC](#) WWDC 2011

 - [Blocks and Grand Central Dispatch in Practice](#) WWDC 2011

 - [Mastering Grand Central Dispatch](#) WWDC 2011

 - [Asynchronous Design Patterns with Blocks, GCD, and XPC](#) WWDC 2012

 - [Efficient Design with XPC](#) WWDC 2013
-

The Big Picture

Primary Goal

Provide best user experience



Primary Goal

Provide best user experience

What is important to user?



Primary Goal

Provide best user experience

What is important to user?

- Frontmost app
- Responsive user interface



Responsive User Interface



Responsive User Interface

Ensure resource availability for

- Main thread of frontmost app
 - UI event handling
 - UI drawing
- OS User Interface infrastructure



Responsive User Interface

Other work should execute

- Off main thread
- Independently
- At lower priority



Priorities

Priorities

Resolving resource contention



Priorities

Resolving resource contention

Under contention

- High priorities win



Priorities

Resolving resource contention

Under contention

- High priorities win

No contention

- Low priorities have no restriction



Scheduling Priority

Kernel scheduler

- High priorities get CPU first
- Low priorities
 - No restriction if no contention
 - May not run during UI activity

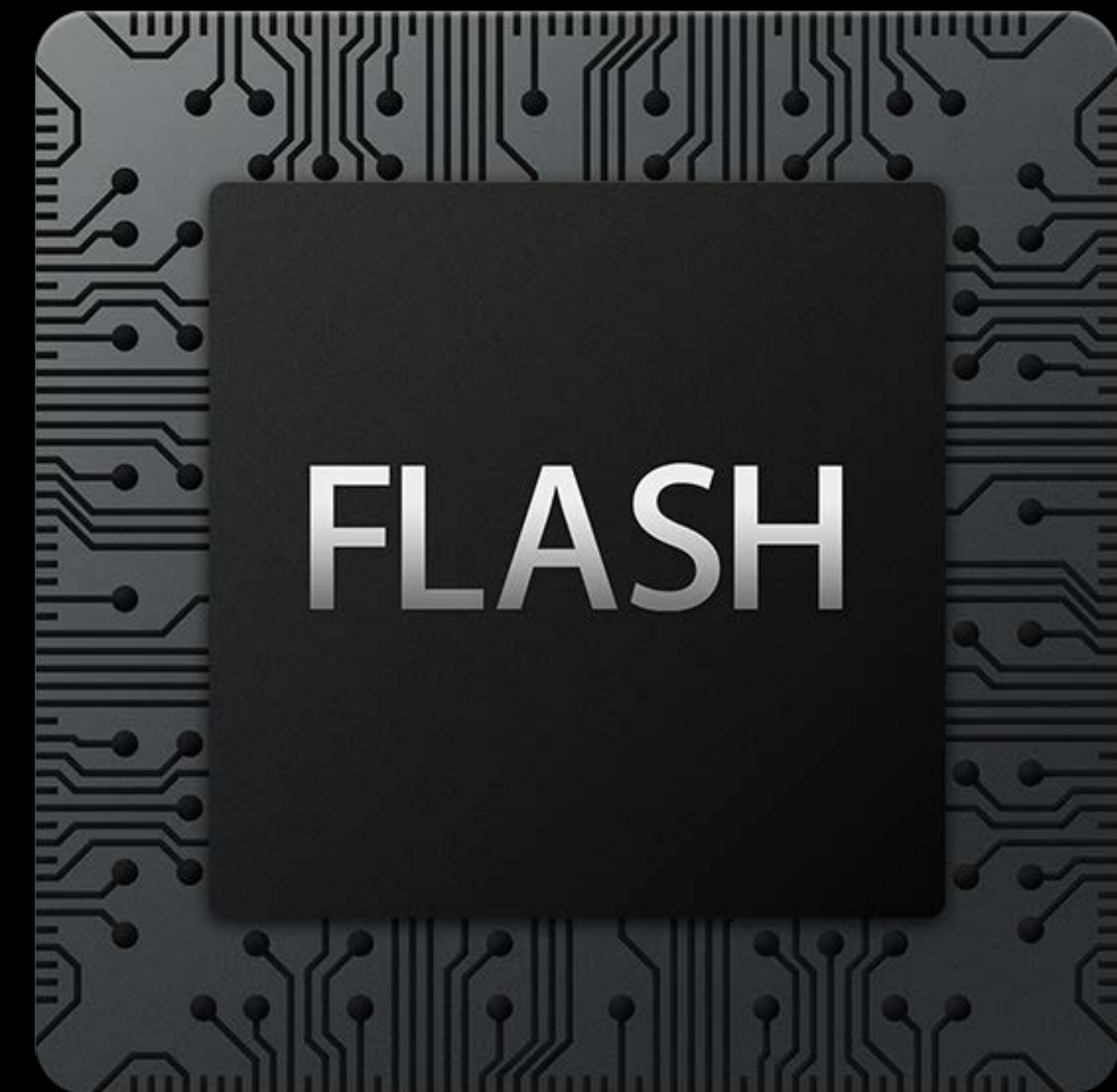


I/O Priority

Background queue

Low priority I/O

- No restriction if no high priority I/O present
- Otherwise deprioritized



Priorities

Many other resource controls

Complex configuration

No unified approach

No clear expression of intent



Quality of Service Classes

Quality of Service Classes

NEW

Communicate developer intent

Explicit classification of work

Single abstract parameter

- Move away from dictating specific configuration values



Quality of Service Classes

Effects

CPU scheduling priority

I/O priority

Timer coalescing

CPU throughput vs. efficiency

More...

Quality of Service Classes

Effects

CPU scheduling priority

I/O priority

Timer coalescing

CPU throughput vs. efficiency

More...

Configuration values tuned for each platform/device

QoS Classes



User-Interactive



User-Initiated



Utility



Background

User-Interactive

UI thread

Directly involved in

- Event handling
- UI drawing

Small fraction of total work



User-Initiated

Asynchronous to UI

Directly UI-initiated

User waiting for immediate results

Required to continue user interaction



Utility

Long-running with user-visible progress

- Computation, I/O, networking

Ongoing data feed to UI

Getting ready for next UI request

Energy efficient



Background

User is unaware work is occurring

Prefetching

Deferrable

Maintenance



Choosing QoS Class

Choosing QoS Class



User Interactive

Is this work actively involved in updating the UI?

Choosing QoS Class



User Interactive

Is this work actively involved in updating the UI?



User Initiated

Is this work required to continue user interaction?

Choosing QoS Class



User Interactive

Is this work actively involved in updating the UI?



User Initiated

Is this work required to continue user interaction?



Utility

Is the user aware of the progress of this work?

Choosing QoS Class



User Interactive

Is this work actively involved in updating the UI?



User Initiated

Is this work required to continue user interaction?



Utility

Is the user aware of the progress of this work?



Background

Can this work be deferred to start at a better time?

Choosing QoS Class

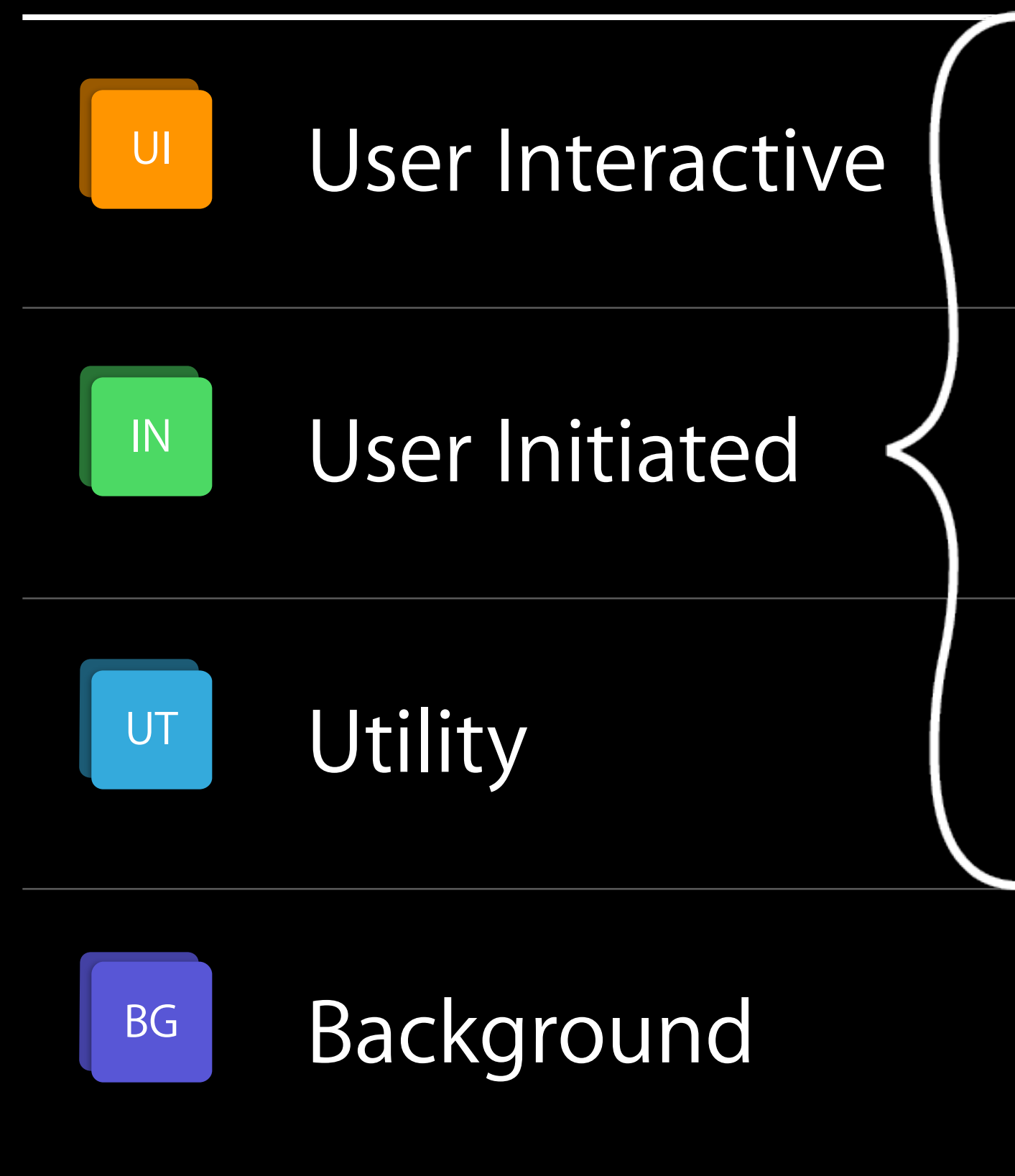
 User Interactive

 User Initiated

 Utility

 Background

Choosing QoS Class



Choosing QoS Class



User Interactive

Is it okay for User Interactive work to happen before my work?



User Initiated

Is it okay for this work to compete with other User Initiated work?



Utility

Is it okay for my work to take precedence over Utility work?



Background

Recap

Responsive User Interface

- Asynchronous execution at correct priority
- Had no unified way to express intent

Quality of Service Classes

- Explicit abstract classification of work
- Questions for choosing QoS

QoS Class API

QoS Class

Can be specified on

Threads

Dispatch Queues

Dispatch Blocks

NSOperationQueue/NSOperation

Processes

QoS Class Constants

`sys/qos.h`

UI `QOS_CLASS_USER_INTERACTIVE`

IN `QOS_CLASS_USER_INITIATED`

UT `QOS_CLASS_UTILITY`

BG `QOS_CLASS_BACKGROUND`

`Foundation.h`

`NSQualityOfServiceUserInteractive`

`NSQualityOfServiceUserInitiated`

`NSQualityOfServiceUtility`

`NSQualityOfServiceBackground`

QoS Class Constants

sys/qos.h

UI	QOS_CLASS_USER_INTERACTIVE
IN	QOS_CLASS_USER_INITIATED
DF	QOS_CLASS_DEFAULT
UT	QOS_CLASS_UTILITY
BG	QOS_CLASS_BACKGROUND
	QOS_CLASS_UNSPECIFIED

Foundation.h

NSQualityOfServiceUserInteractive
NSQualityOfServiceUserInitiated
NSQualityOfServiceUtility
NSQualityOfServiceBackground

Special QoS Class Values

QOS_CLASS_DEFAULT

- No specific QoS information was available
- Ordered between UI and non-UI QoS
- Thread and global queue default
- Not intended as a work classification



User Interactive



User Initiated



Default



Utility



Background

Special QoS Class Values

`QOS_CLASS_UNSPECIFIED`

- No QoS specification at given level
- QoS should be inferred from work origin
- Returned after legacy API QoS opt-out

QoS Relative Priority

Relative position within a QoS Class band

QoS Relative Priority

Relative position within a QoS Class band

UI

IN

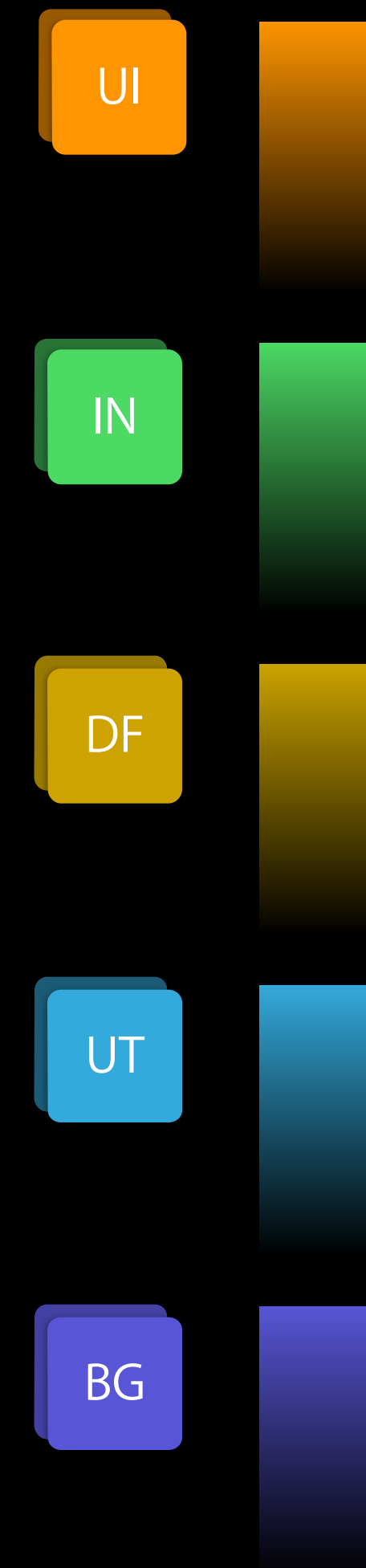
DF

UT

BG

QoS Relative Priority

Relative position within a QoS Class band



QoS Relative Priority

Relative position within a QoS Class band
Lower than default



QoS Relative Priority

Relative position within a QoS Class band

Lower than default

Intended for unusual situations

- Interdependent work within same QoS class with differing priority
- Producer/Consumer scenarios



Thread QoS API

Thread QoS Getters

QoS Class of current thread

```
qos = qos_class_self();
```

Initial QoS Class of main thread

```
qos = qos_class_main();
```

Thread QoS Getters

QoS Class of current thread

```
qos = qos_class_self();
```

Initial QoS Class of main thread

```
qos = qos_class_main();
```

Thread QoS Getters

QoS Class of current thread

```
qos = qos_class_self();
```

Initial QoS Class of main thread

```
qos = qos_class_main();
```

Process Type	Main QoS
App	User-Interactive
XPC Service	Default

GCD QoS API

Global Queues

Global Queue

Main






High priority concurrent

Default priority concurrent

Low priority concurrent

Background priority concurrent

Global Queues

	QoS Class	Global Queue
	User Interactive	Main
	User Initiated	High priority concurrent
	Default	Default priority concurrent
	Utility	Low priority concurrent
	Background	Background priority concurrent

Global Queues with QoS

Get a global concurrent queue with QoS Class

```
queue = dispatch_get_global_queue(QOS_CLASS_UTILITY, 0);
```

Get QoS Class of a queue

```
qos = dispatch_queue_get_qos_class(queue, &relative);
```

Global Queues with QoS

Get a global concurrent queue with QoS Class

```
queue = dispatch_get_global_queue(QOS_CLASS_UTILITY, 0);
```

Get QoS Class of a queue

```
qos = dispatch_queue_get_qos_class(queue, &relative);
```

Global Queues with QoS

Get a global concurrent queue with QoS Class

```
queue = dispatch_get_global_queue(QOS_CLASS_UTILITY, 0);
```

Get QoS Class of a queue

```
qos = dispatch_queue_get_qos_class(queue, &relative);
```


Queue QoS API

Get QoS queue attribute:

```
qos_attr = dispatch_queue_attr_make_with_qos_class(  
                                                    attr, QOS_CLASS_UTILITY, 0);  
  
queue = dispatch_queue_create("com.my.utility", qos_attr);
```

Dispatch Block Objects

Dispatch Block Objects

Configure properties of individual units of work on a queue

Dispatch Block Objects

Configure properties of individual units of work on a queue

Address individual workunits for

- Wait for completion
- Completion notification
- Cancellation

Dispatch Block Objects

Configure properties of individual units of work on a queue

Address individual workunits for

- Wait for completion
- Completion notification
- Cancellation

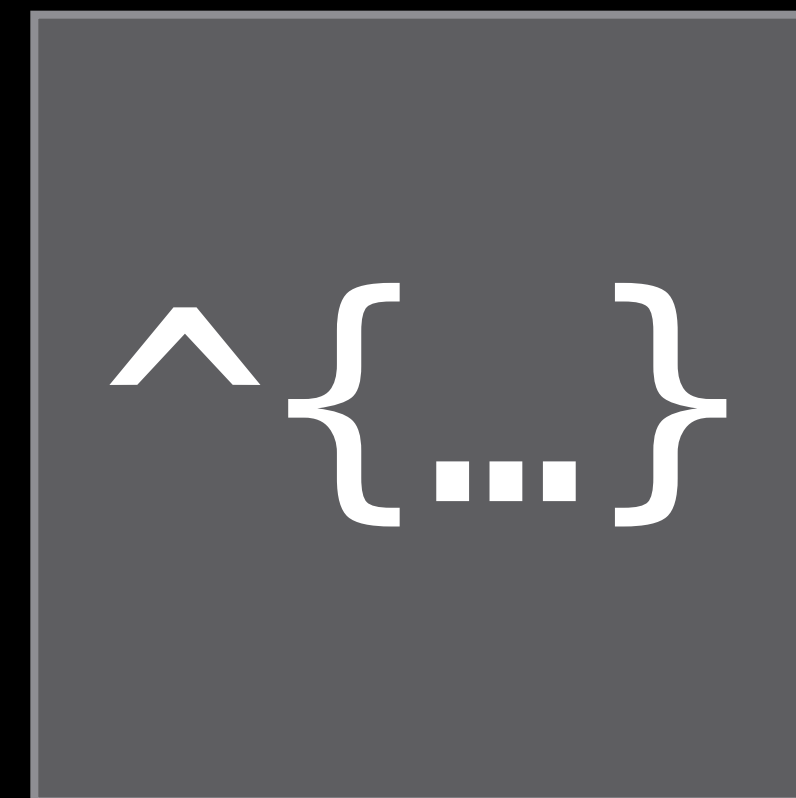
Integrate with existing API

Dispatch Block Objects

Wrapper Block

Created from an existing GCD Block

- `dispatch_block_t`



Dispatch Block Objects

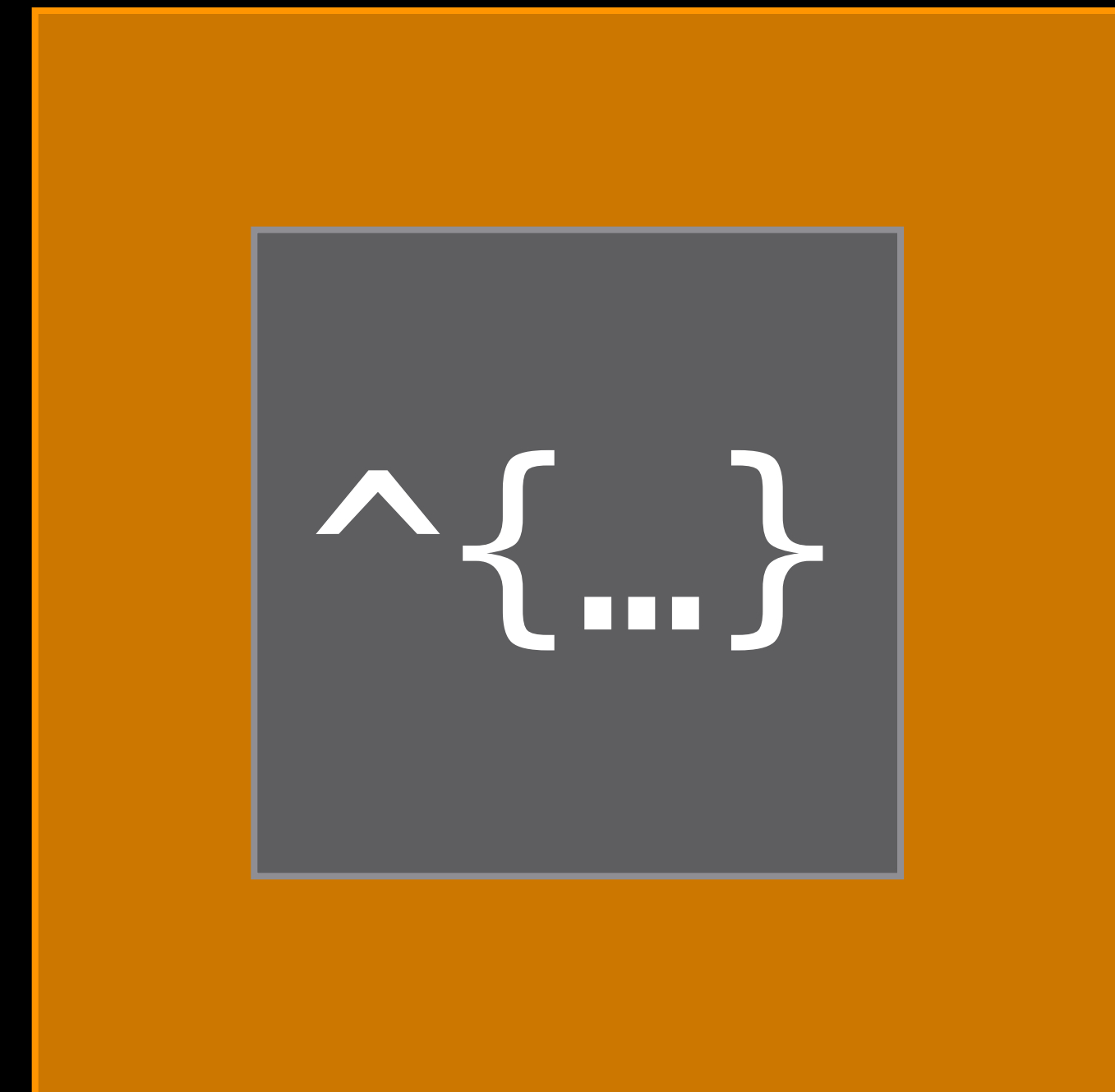
Wrapper Block

Created from an existing GCD Block

- `dispatch_block_t`

Additional configuration

- QoS Class
- Flags



Dispatch Block Objects

Wrapper Block

Created from an existing GCD Block

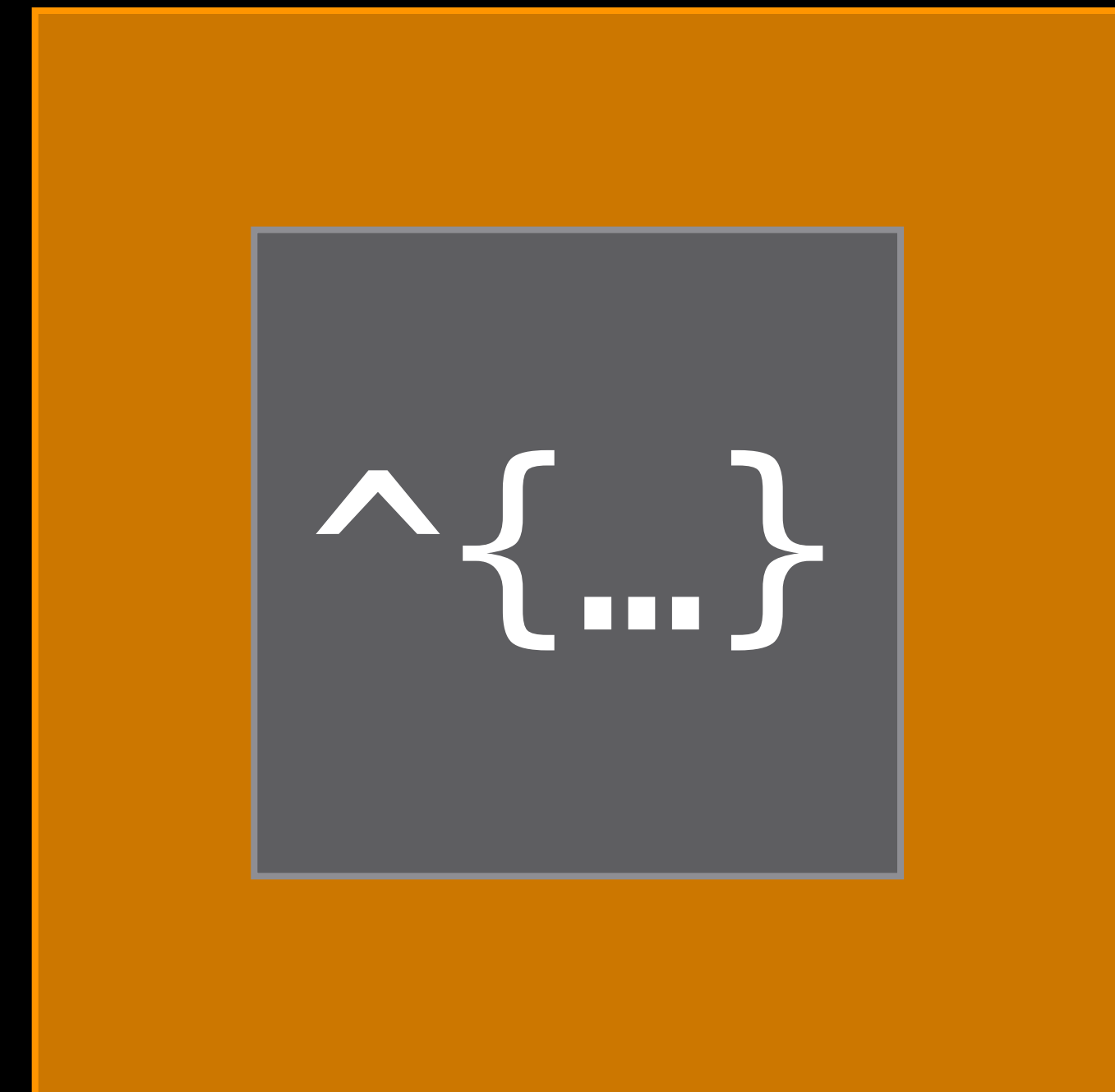
- `dispatch_block_t`

Additional configuration

- QoS Class
- Flags

Heap object

- `Block_release()`



GCD Block API

```
dispatch_block_t block;

block = dispatch_block_create(0, ^{
    NSLog(@"Hello World!");
});

dispatch_async(queue, block);

// Do some work

dispatch_wait(block, DISPATCH_TIME_FOREVER);

Block_release(block);
```


GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(0, ^{  
    NSLog(@"Hello World!");  
});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
dispatch_wait(block, DISPATCH_TIME_FOREVER);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(0, ^{  
    NSLog(@"Hello World!");  
});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
dispatch_wait(block, DISPATCH_TIME_FOREVER);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(0, ^{  
    NSLog(@"Hello World!");  
});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
dispatch_wait(block, DISPATCH_TIME_FOREVER);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(0, ^{  
    NSLog(@"Hello World!");  
});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
dispatch_wait(block, DISPATCH_TIME_FOREVER);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(0, ^{  
    NSLog(@"Hello World!");  
});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
dispatch_wait(block, DISPATCH_TIME_FOREVER);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;  
  
block = dispatch_block_create(0, ^{  
    NSLog(@"Hello World!");  
});  
  
dispatch_async(queue, block);  
  
// Do some work  
  
dispatch_wait(block, DISPATCH_TIME_FOREVER);  
  
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create_with_qos_class(  
    0, QOS_CLASS_UTILITY, -8, ^{...});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
// Change your mind
```

```
dispatch_cancel(block);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create_with_qos_class(  
    0, QOS_CLASS_UTILITY, -8, ^{...});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
// Change your mind
```

```
dispatch_cancel(block);
```

```
Block_release(block);
```


GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create_with_qos_class(  
    0, QOS_CLASS_UTILITY, -8, ^{...});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
// Change your mind
```

```
dispatch_cancel(block);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create_with_qos_class(  
    0, QOS_CLASS_UTILITY, -8, ^{...});
```

```
dispatch_async(queue, block);
```

```
// Do some work  
// Change your mind
```

```
dispatch_cancel(block);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create_with_qos_class(  
    0, QOS_CLASS_UTILITY, -8, ^{...});
```

```
dispatch_async(queue, block);
```

```
// Do some work
```

```
// Change your mind
```

```
dispatch_cancel(block);
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(DISPATCH_BLOCK_DETACHED, ^{  
    // Clean caches  
});
```

```
dispatch_async(queue, block);
```

```
dispatch_notify(block, dispatch_get_main_queue(), ^{  
    // Cleanup complete  
});
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(DISPATCH_BLOCK_DETACHED, ^{  
    // Clean caches  
});
```

```
dispatch_async(queue, block);
```

```
dispatch_notify(block, dispatch_get_main_queue(), ^{  
    // Cleanup complete  
});
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(DISPATCH_BLOCK_DETACHED, ^{  
    // Clean caches  
});
```

```
dispatch_async(queue, block);
```

```
dispatch_notify(block, dispatch_get_main_queue(), ^{  
    // Cleanup complete  
});
```

```
Block_release(block);
```

GCD Block API

```
dispatch_block_t block;
```

```
block = dispatch_block_create(DISPATCH_BLOCK_DETACHED, ^{  
    // Clean caches  
});
```

```
dispatch_async(queue, block);
```

```
dispatch_notify(block, dispatch_get_main_queue(), ^{  
    // Cleanup complete  
});
```

```
Block_release(block);
```

Interaction of QoS Specifications

Interaction of Multiple QoS Specifications

Asynchronous Blocks

Default to QoS class of queue

- Or inherited from immediate global target queue

Interaction of Multiple QoS Specifications

Asynchronous Blocks

Default to QoS class of queue

- Or inherited from immediate global target queue

If neither are specified

- Use Block QoS class
- Or QoS inferred from submitting thread

Interaction of Multiple QoS Specifications

Inferred QoS

QoS captured at the time of block submission

- User Interactive translated to User Initiated

Interaction of Multiple QoS Specifications

Inferred QoS

QoS captured at the time of block submission

- User Interactive translated to User Initiated

Intended for use on queues

- Without specific identity or single purpose
- Mediating between many different clients

Interaction of Multiple QoS Specifications

Synchronous Blocks

Default to QoS class of Block

- Or current thread
- Will only raise QoS

Interaction of Multiple QoS Specifications

Explicit control

DISPATCH_BLOCK_INHERIT_QOS_CLASS

- Prefer queue/thread QoS

DISPATCH_BLOCK_ENFORCE_QOS_CLASS

- Prefer Block QoS
- Only if higher than queue/thread QoS

Interaction of Multiple QoS Specifications

Explicit control

DISPATCH_BLOCK_INHERIT_QOS_CLASS

- Prefer queue/thread QoS

DISPATCH_BLOCK_ENFORCE_QOS_CLASS

- Prefer Block QoS
- Only if higher than queue/thread QoS

Priority Inversions

Priority Inversion

Progress of high-priority work depends on

- Results of low-priority work
- Resource held by low-priority work



Priority Inversion

Progress of high-priority work depends on

- Results of low-priority work
- Resource held by low-priority work

High-priority threads are

- Blocking
- Spinning/polling

Waiting for low-priority threads



Priority Inversion

Synchronous

High QoS thread waiting on lower QoS work

Priority Inversion

Synchronous

High QoS thread waiting on lower QoS work

System will attempt to automatically resolve inversion for

- `dispatch_sync()` and `dispatch_wait()` of blocks on serial queues
- `pthread_mutex_lock()`

QoS of work is raised for the duration of the wait

Priority Inversion

Asynchronous

High QoS Block submitted to serial queue

- Created with lower QoS
- Containing Blocks with lower QoS

Priority Inversion

Asynchronous

High QoS Block submitted to serial queue

- Created with lower QoS
- Containing Blocks with lower QoS

System will attempt to automatically resolve inversion

QoS of queue is raised until high QoS Block is reached

Avoiding Priority Inversions

Decouple shared data as much as possible

- Use finer grained synchronization
- Move work outside of lock/serial queue

Avoiding Priority Inversions

Decouple shared data as much as possible

- Use finer grained synchronization
- Move work outside of lock/serial queue

Prefer asynchronous execution over synchronous waiting

Avoiding Priority Inversions

Decouple shared data as much as possible

- Use finer grained synchronization
- Move work outside of lock/serial queue

Prefer asynchronous execution over synchronous waiting

Avoid spinning/polling

- Look out for timer-based “synchronization”

Recap

QoS Class constants

QoS relative priority

Thread and queue QoS API

Dispatch Block API

Interaction of multiple QoS specifications

Priority Inversions

Propagation of Execution Context

Execution Context

Thread-local attributes maintained by system

- Activity ID
- Properties of current IPC request
 - Originator
 - Importance
 - More...

Execution Context

Automatic propagation

Execution Context

Automatic propagation

Propagated across threads

- GCD
- NSOperationQueue
- Foundation

Execution Context

Automatic propagation

Propagated across threads

- GCD
- NSOperationQueue
- Foundation

Propagated across processes

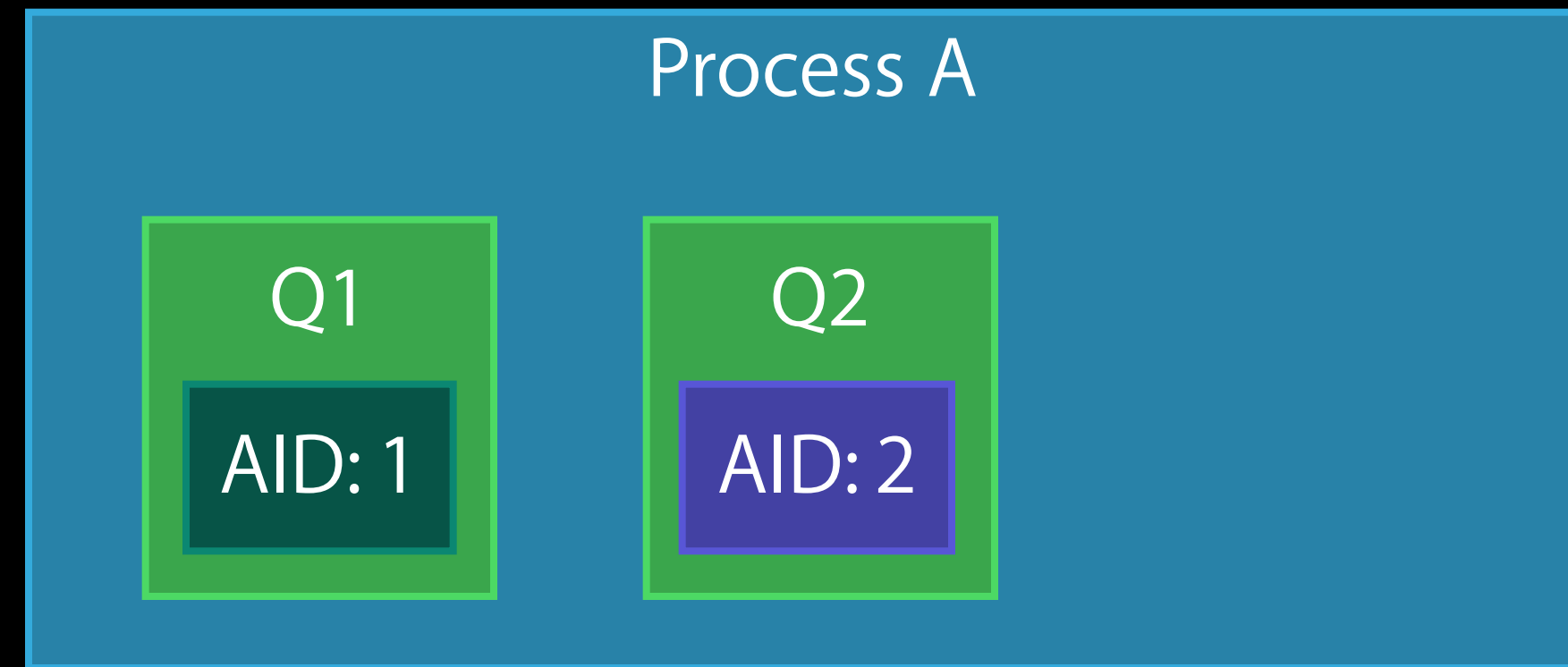
- XPC
- MIG
- CFMachPort

Automatic Propagation

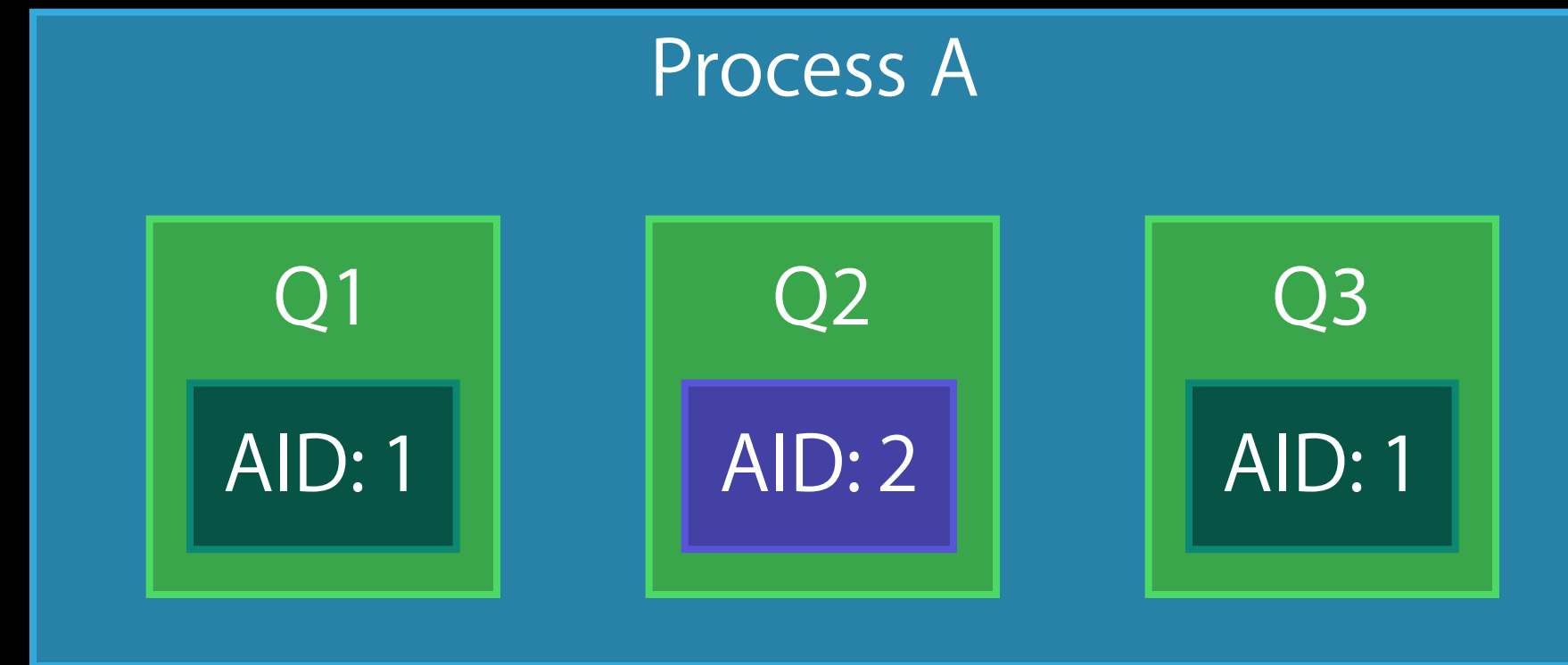


Process A

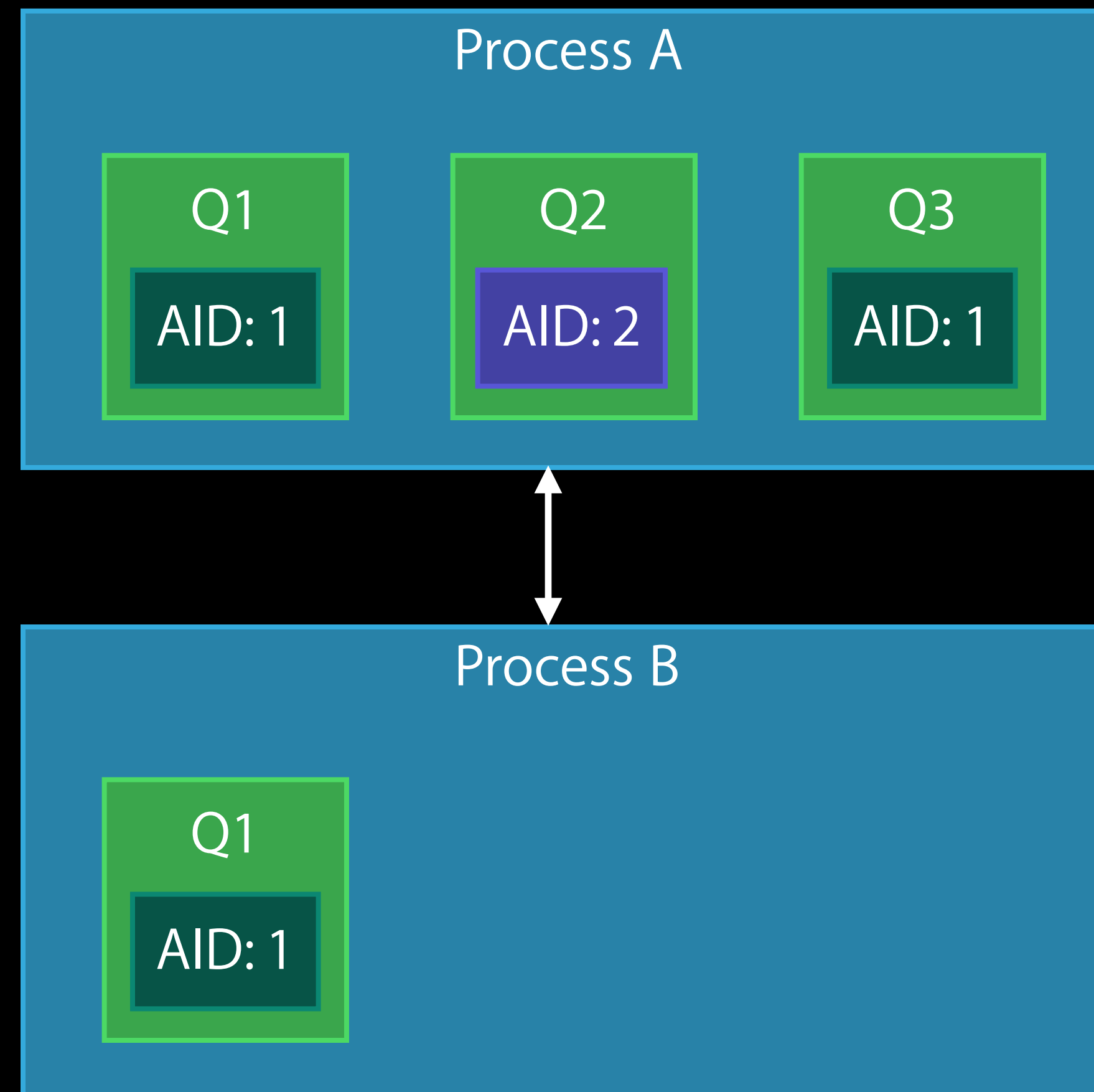
Automatic Propagation



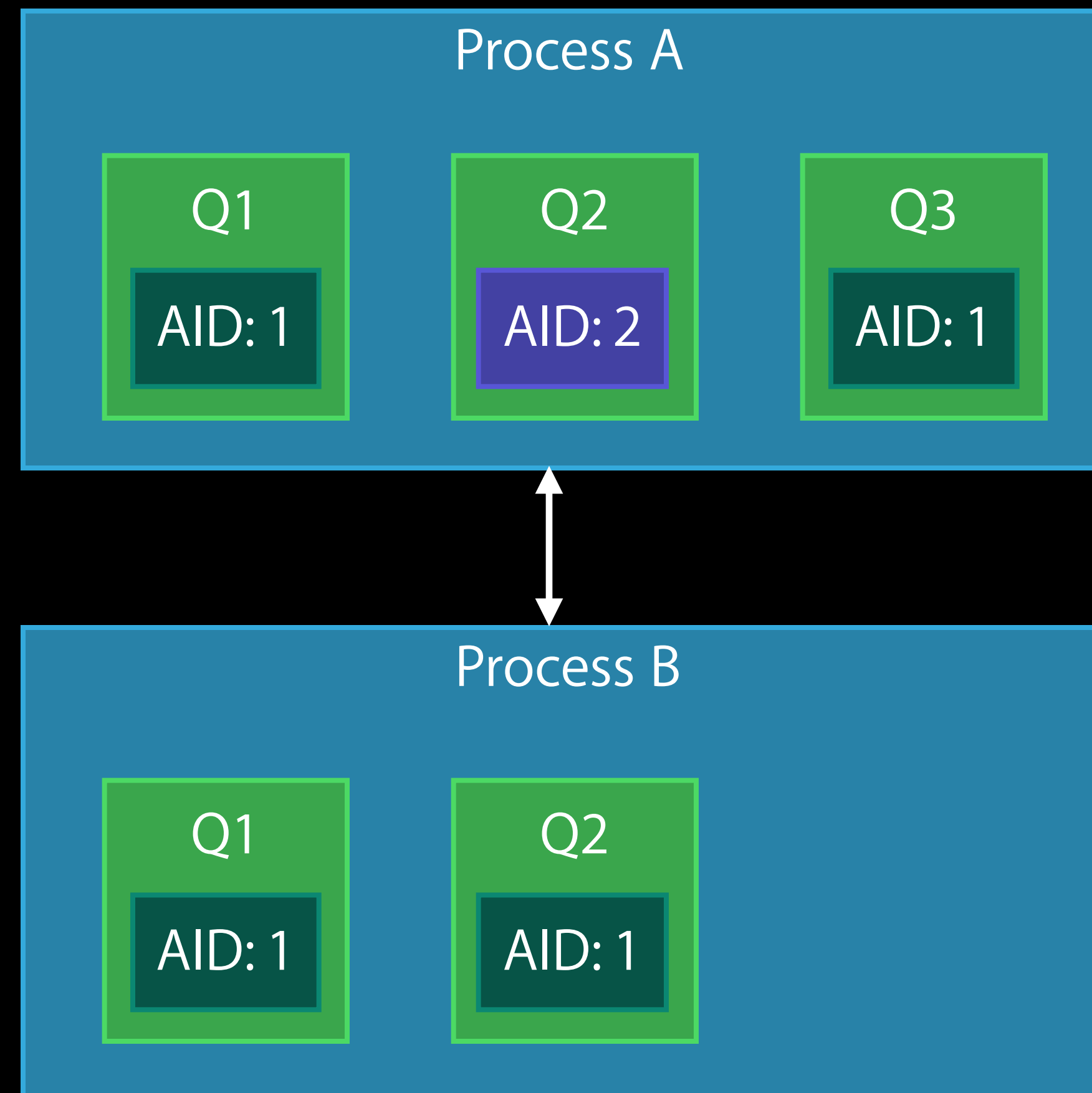
Automatic Propagation



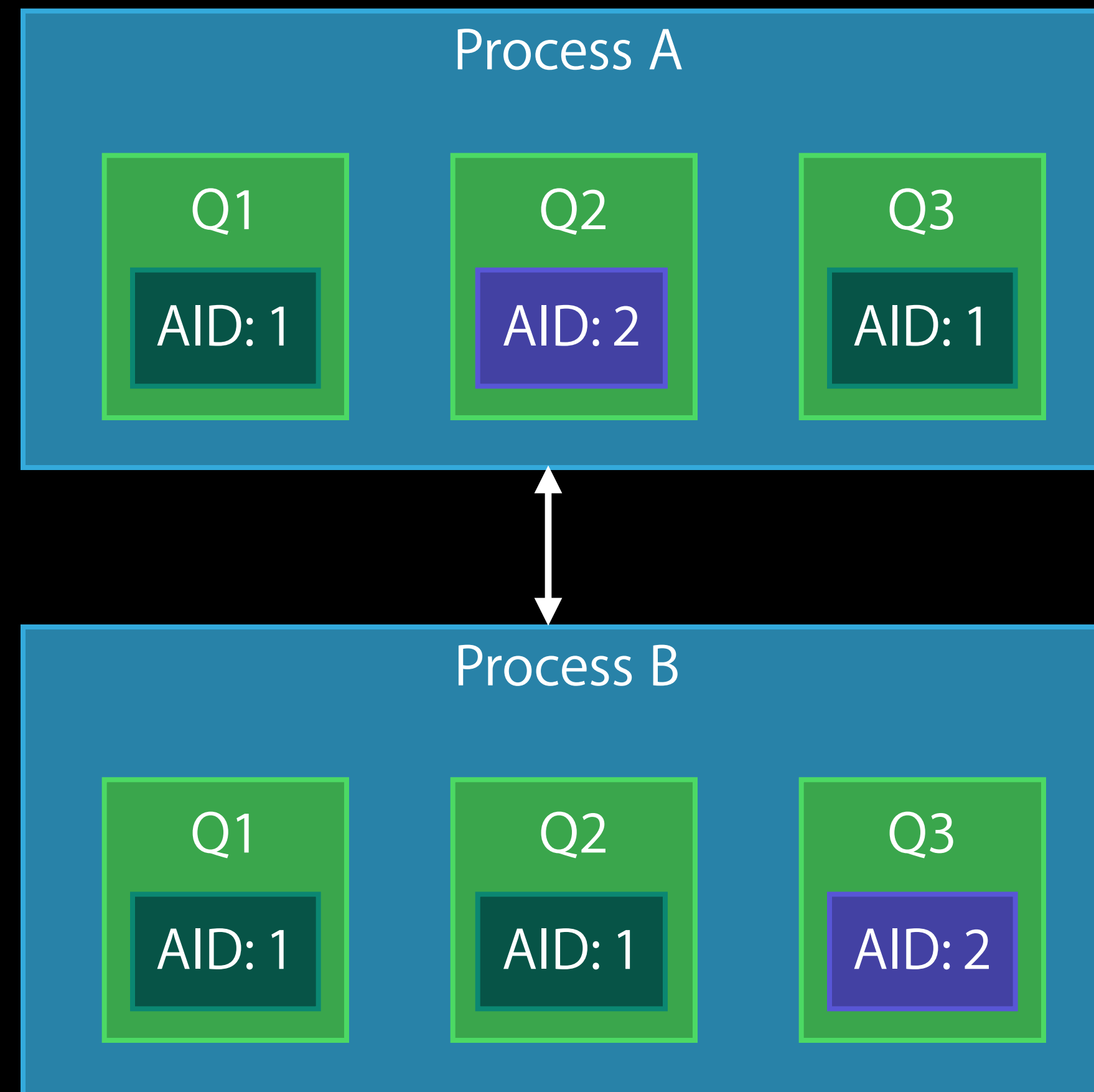
Automatic Propagation



Automatic Propagation



Automatic Propagation



Propagation Control

Prevent propagation

DISPATCH_BLOCK_DETACHED

Work disassociated from principal activity

- Asynchronous, long-running cleanup

Propagation Control

Prevent propagation

DISPATCH_BLOCK_DETACHED

Work disassociated from principal activity

- Asynchronous, long-running cleanup

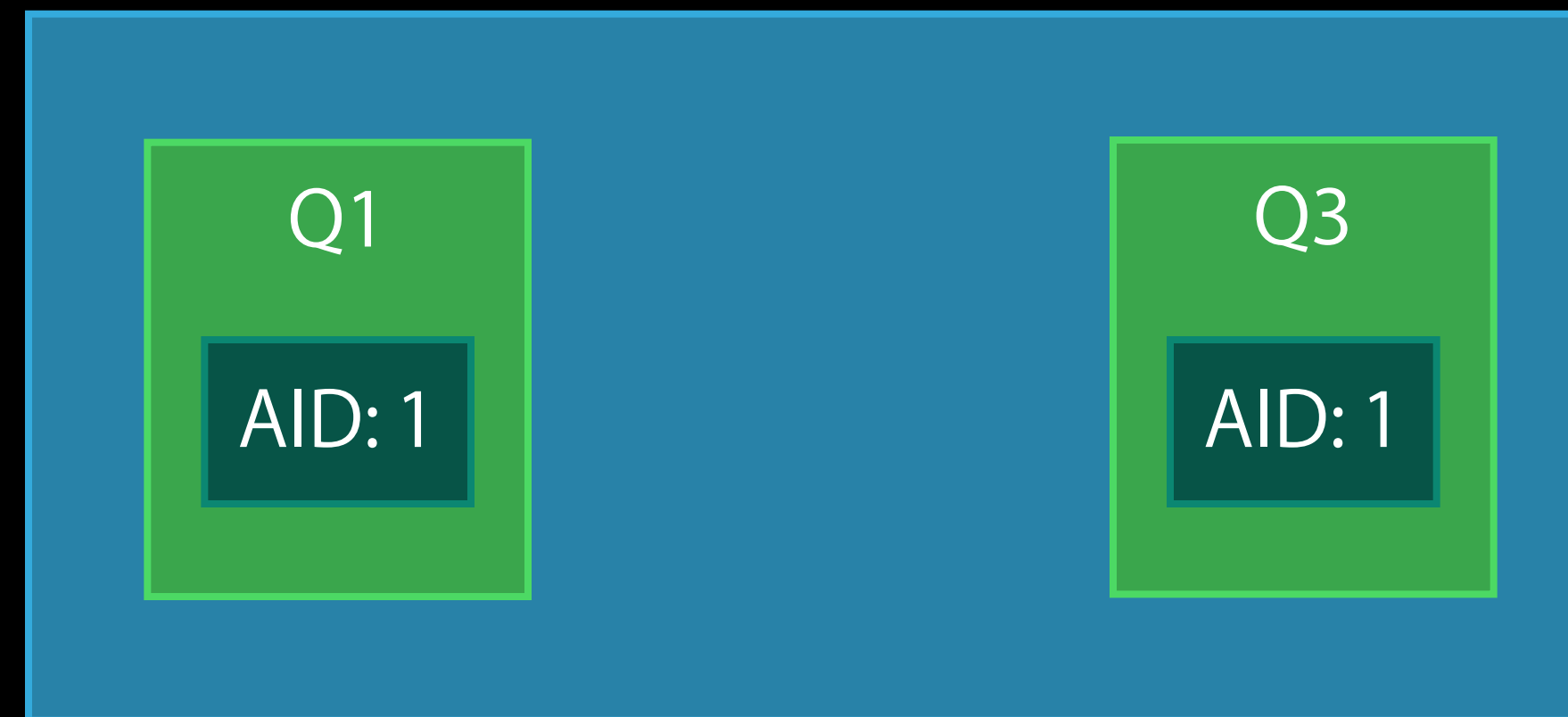
Detached by default

- Dispatch source handlers
- `dispatch_after()`

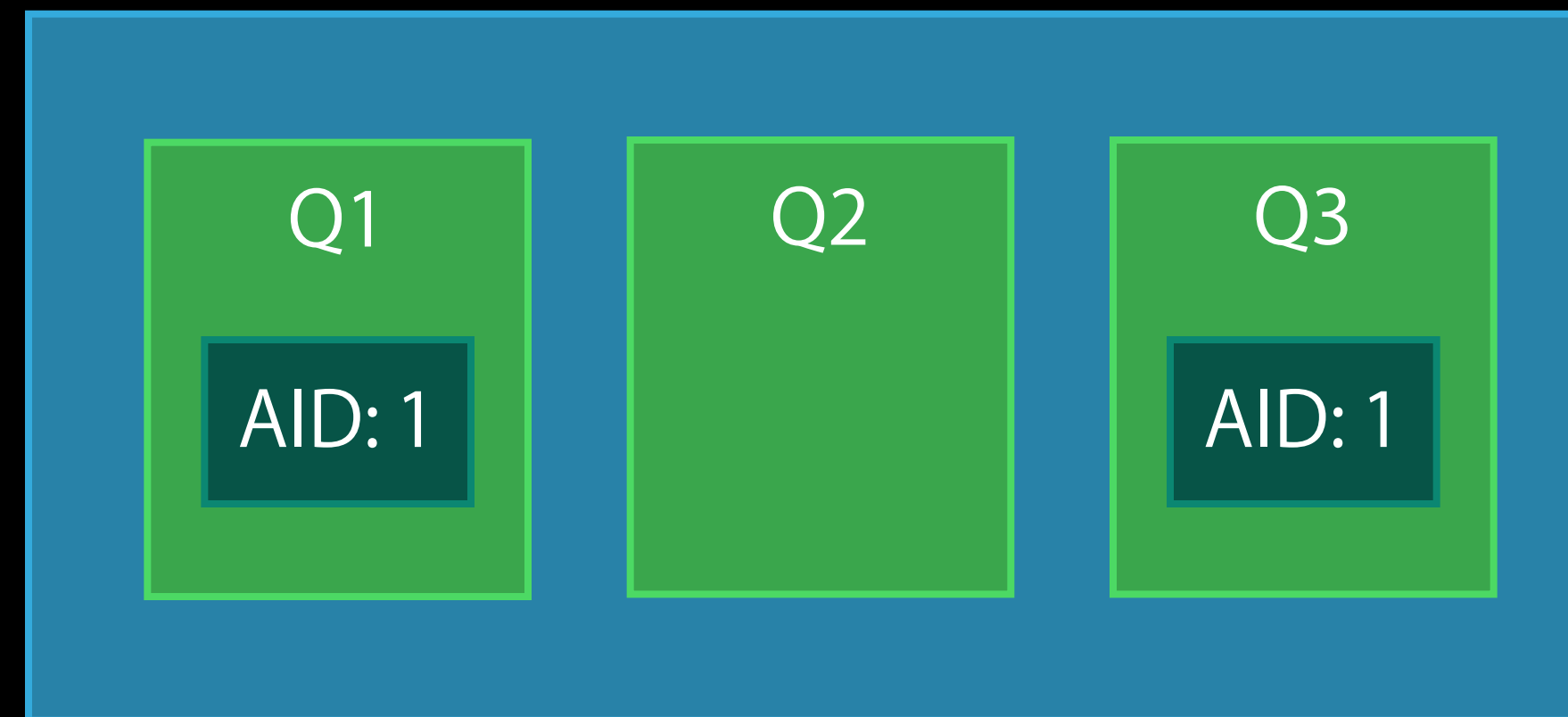
Detached Block



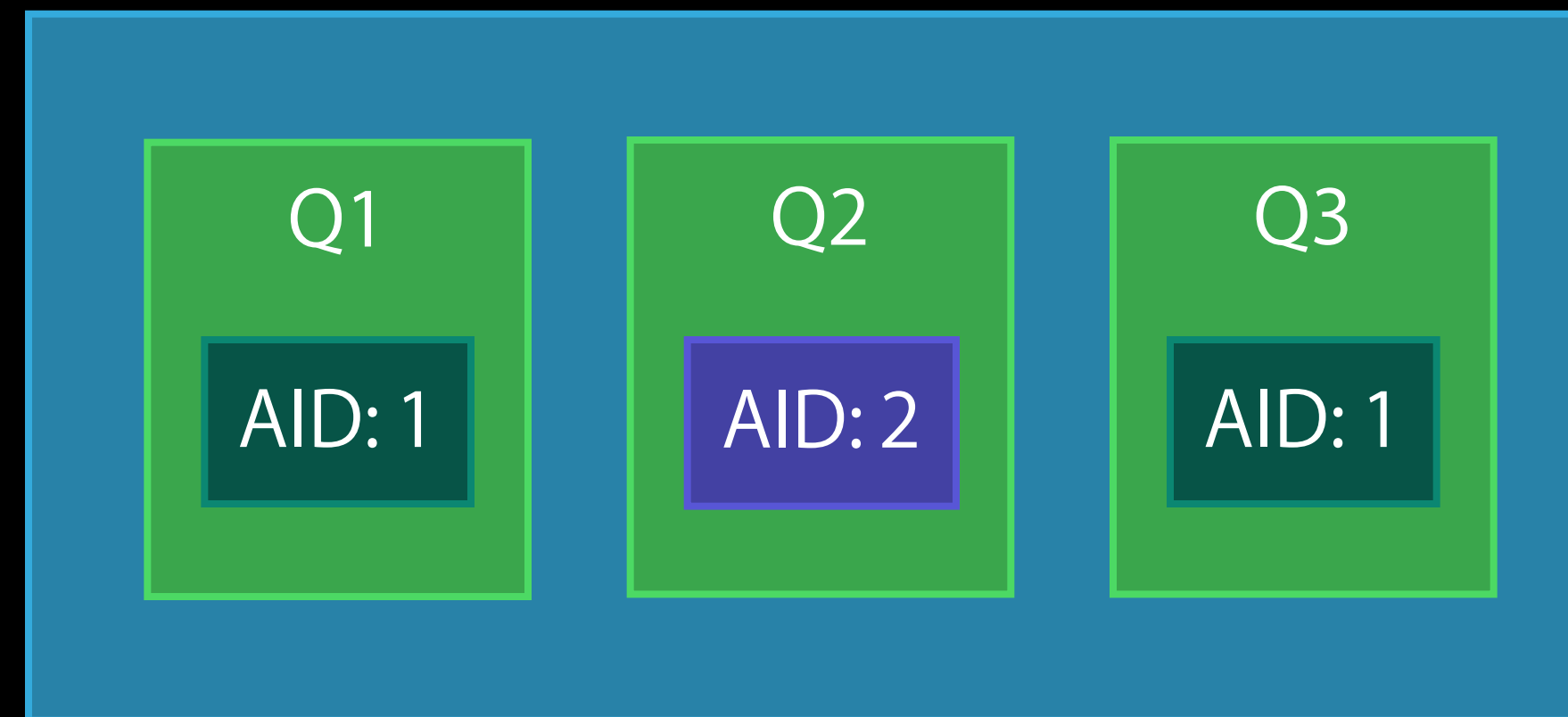
Detached Block



Detached Block



Detached Block



Propagation Control

Manual propagation

DISPATCH_BLOCK_ASSIGN_CURRENT

- Assigns current QoS Class and Execution Context

Propagation Control

Manual propagation

DISPATCH_BLOCK_ASSIGN_CURRENT

- Assigns current QoS Class and Execution Context

Store Block for later execution

- Direct call on manually created pthread
- Submission to dispatch queue

XPC

Propagation

XPC connections automatically propagate

- QoS Class
- Execution Context

XPC

Propagation

XPC connections automatically propagate

- QoS Class
- Execution Context

Capture of current state on sending thread

XPC

Propagation

XPC connections automatically propagate

- QoS Class
- Execution Context

Capture of current state on sending thread

XPC handlers prefer propagated QoS over queue QoS

XPC Service

Importance boosting

Initially clamped to Background QoS

Clamp removed during IPC with UI



XPC Service

Importance boosting

Initially clamped to Background QoS

Clamp removed during IPC with UI

Boost lifetime automatically handled by XPC

- Until reply is sent
- While using message



XPC Service

Importance boosting



Initially clamped to Background QoS

Clamp removed during IPC with UI

Boost lifetime automatically handled by XPC

- Until reply is sent
- While using message
- While asynchronous work submitted from handler context is ongoing
 - Ensure unrelated work is detached

Recap

Execution Context attributes

Automatic propagation of Execution Context and QoS

Manual propagation control

XPC propagation and importance boosting

Diagnostics and Queue Debugging

Xcode 6 CPU Report

Xcode 6 Queue Debugging

Activity Tracing

Diagnostics and Queue Debugging

Xcode 6 CPU Report

Xcode 6 Queue Debugging

Activity Tracing

MyApplication
PID 7632, Paused

- CPU** 7%
- Memory** 12.7 MB
- Energy Impact** Low
- Disk** Zero KB/s
- Network** Zero KB/s

Thread 1
Queue: com.apple.main-thread (serial)

- 0 mach_msg_trap
- 11 NSApplicationMain
- 12 main
- 13 start

Thread 2
Queue: com.apple.libdispatch-manager (serial)

Thread 5
Queue: com.apple.root.utility-qos (concurrent)

- 0 __semwait_signal
- 2 usleep
- 3 __45-[AppDelegate applicationDidFini...
- 4 _dispatch_call_block_and_release
- 9 start_wqthread

Enqueued from com.apple.main-thread (Thr...

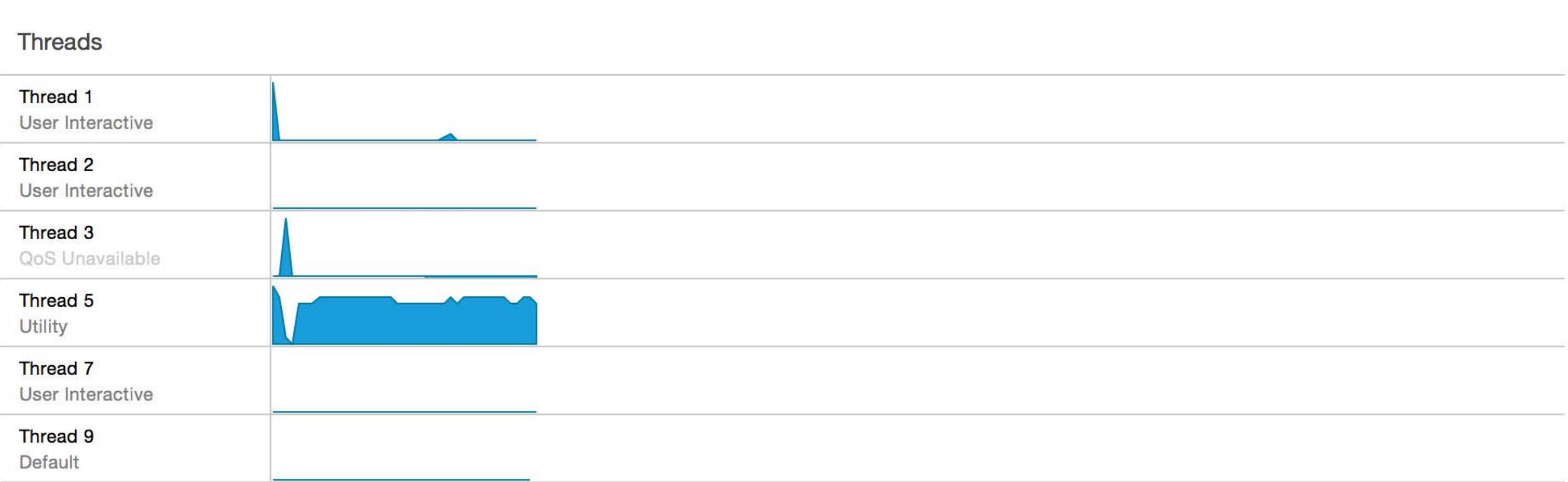
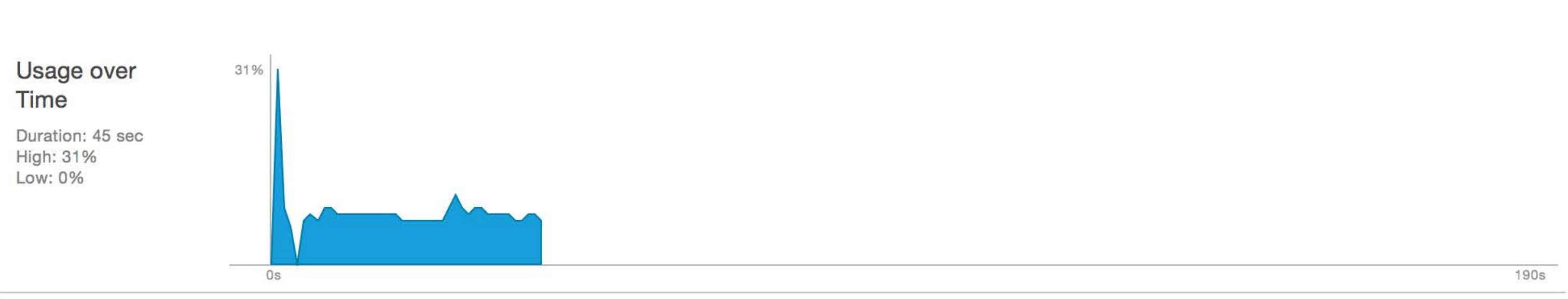
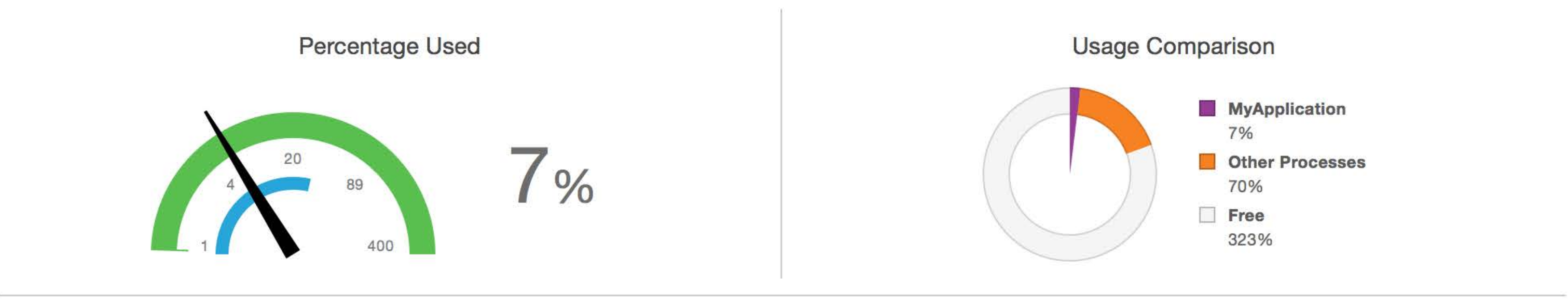
- 0 _dispatch_async_f_slow
- 1 -[AppDelegate applicationDidFinishLa...
- 2 __CFNOTIFICATIONCENTER_IS_CALL...
- 18 NSApplicationMain
- 19 main
- 20 start

Thread 7

Thread 9

Thread 10

CPU Report



(lldb)

All Output

MyApplication
PID 7632, Paused

- CPU** 7%
- Memory** 12.7 MB
- Energy Impact** Low
- Disk** Zero KB/s
- Network** Zero KB/s

Thread 1
Queue: com.apple.main-thread (serial)

- 0 mach_msg_trap
- 11 NSApplicationMain
- 12 main
- 13 start

Thread 2
Queue: com.apple.libdispatch-manager (serial)

Thread 5
Queue: com.apple.root.utility-qos (concurrent)

- 0 __semwait_signal
- 2 usleep
- 3 __45-[AppDelegate applicationDidFini...
- 4 _dispatch_call_block_and_release
- 9 start_wqthread

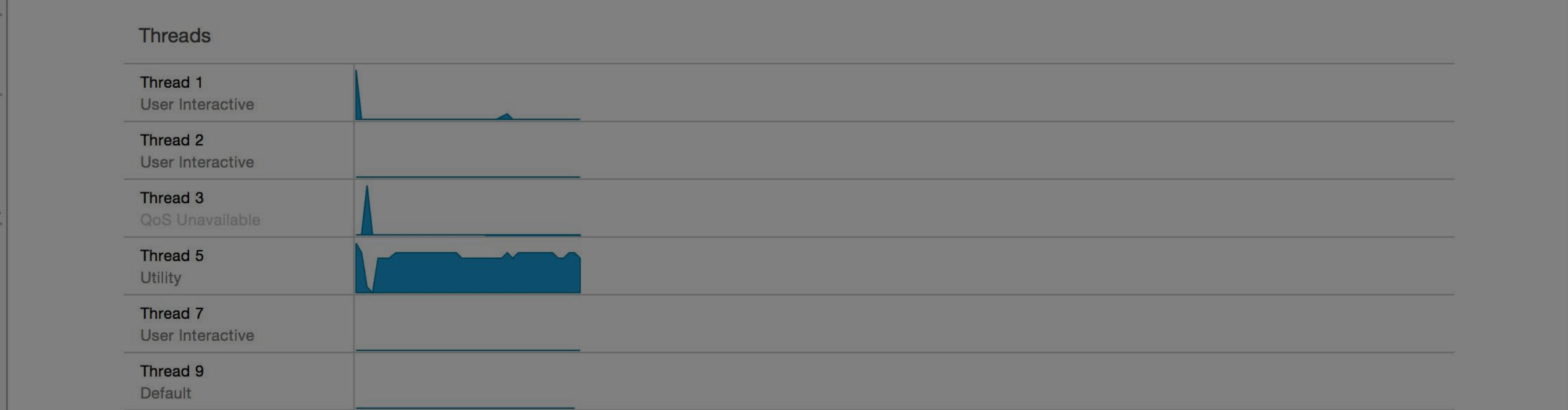
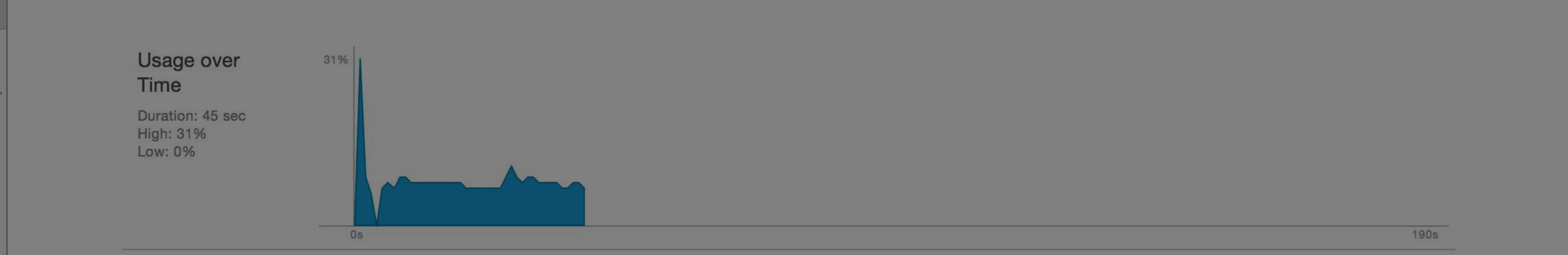
Enqueued from com.apple.main-thread (Thr...)

- 0 _dispatch_async_f_slow
- 1 -[AppDelegate applicationDidFinishLa...
- 2 __CFNOTIFICATIONCENTER_IS_CALL...
- 18 NSApplicationMain
- 19 main
- 20 start

Thread 7

Thread 9

Thread 10



MyApplication > Thread 5 > 0 __semwait_signal

(lldb)

All Output

MyApplication > My Mac 64-bit | Running MyApplication : MyApplication

MyApplication
PID 7632, Paused

- CPU** 7%
- Memory** 12.7 MB
- Energy Impact** Low
- Disk** Zero KB/s
- Network** Zero KB/s

Thread 1
Queue: com.apple.main-thread (serial)

- 0 mach_msg_trap
- 11 NSApplicationMain
- 12 main
- 13 start

Thread 2
Queue: com.apple.libdispatch-manager (serial)

Thread 5
Queue: com.apple.root.utility-qos (concurrent)

- 0 __semwait_signal
- 2 usleep
- 3 __45-[AppDelegate applicationDidFini...
- 4 _dispatch_call_block_and_release
- 9 start_wqthread

Enqueued from com.apple.main-thread (Thr...

- 0 _dispatch_async_f_slow
- 1 -[AppDelegate applicationDidFinishLa...
- 2 __CFNOTIFICATIONCENTER_IS_CALL...
- 18 NSApplicationMain
- 19 main
- 20 start

Thread 7

Thread 9

Thread 10

CPU Report

Percentage Used

7%

Usage Comparison

- MyApplication 7%
- Other Processes 70%
- Free 323%

Usage over Time

Duration: 45 sec
High: 31%
Low: 0%

Threads

Thread	Category	Usage
Thread 1	User Interactive	Low usage
Thread 2	User Interactive	Low usage
Thread 3	QoS Unavailable	Low usage
Thread 5	Utility	High usage (up to 31%)
Thread 7	User Interactive	Low usage
Thread 9	Default	Low usage

MyApplication > Thread 5 > 0 __semwait_signal

(lldb)

All Output

MyApplication
PID 7632, Paused

- CPU** 7%
- Memory** 12.7 MB
- Energy Impact** Low
- Disk** Zero KB/s
- Network** Zero KB/s

Thread 1
Queue: com.apple.main-thread (serial)

- 0 mach_msg_trap
- 11 NSApplicationMain
- 12 main
- 13 start

Thread 2
Queue: com.apple.libdispatch-manager (serial)

Thread 5
Queue: com.apple.root.utility-qos (concurrent)

- 0 __semwait_signal
- 2 usleep
- 3 __45-[AppDelegate applicationDidFini...
- 4 _dispatch_call_block_and_release
- 9 start_wqthread

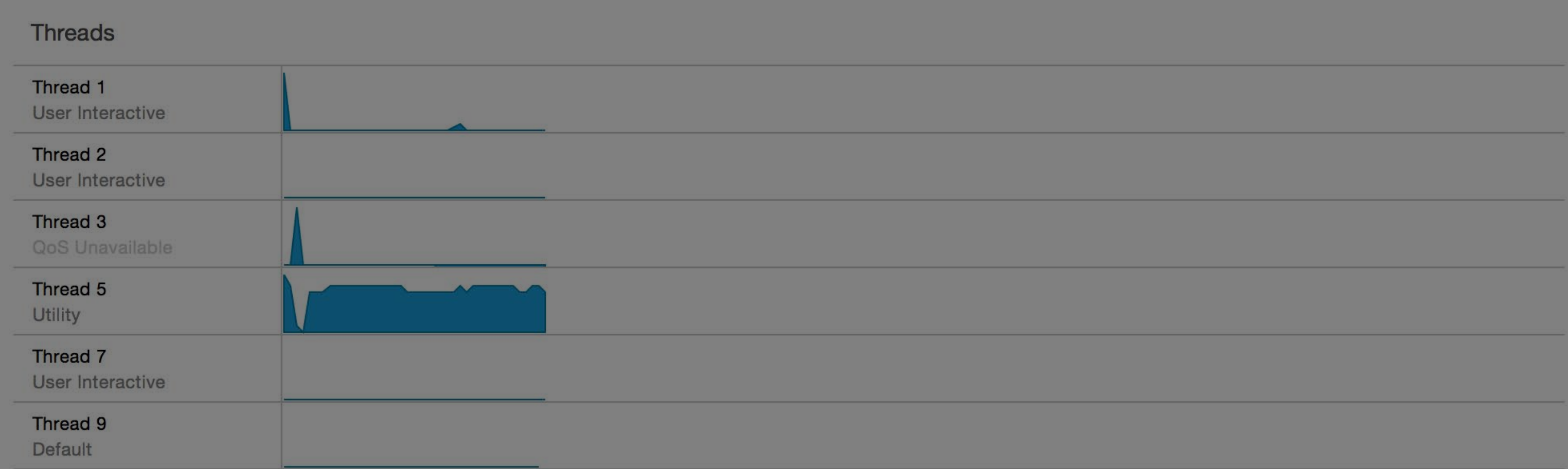
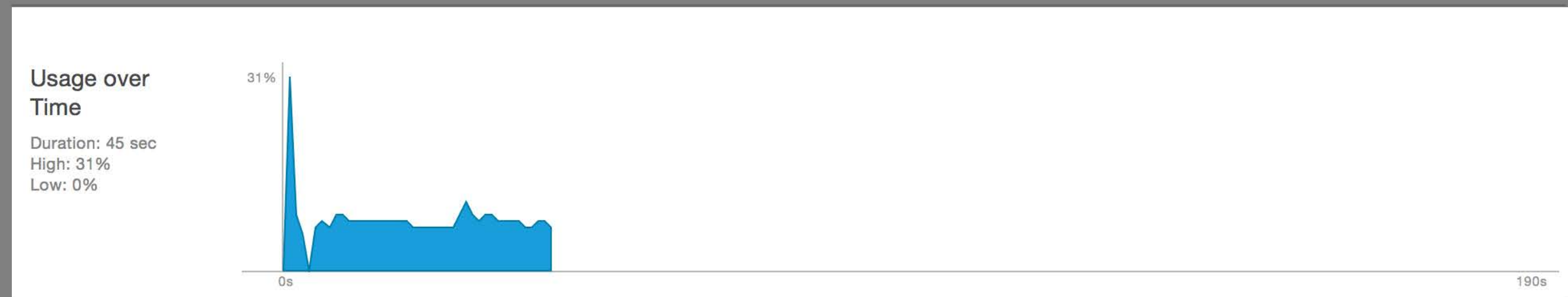
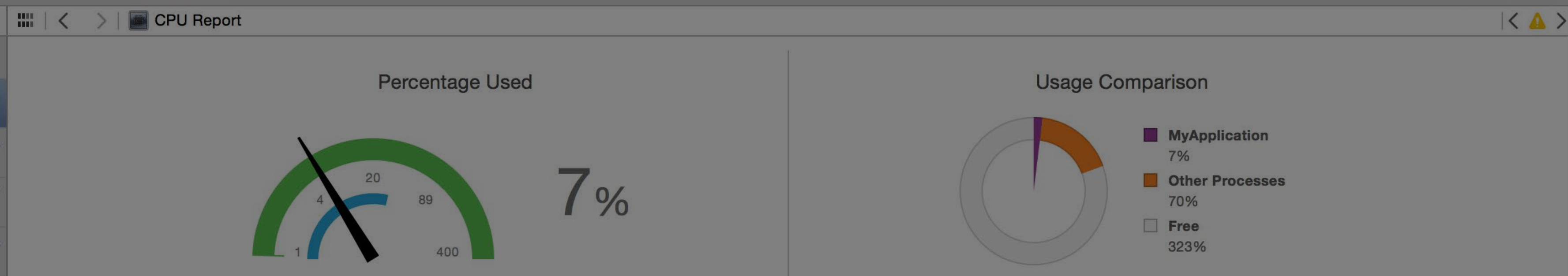
Enqueued from com.apple.main-thread (Thr...)

- 0 _dispatch_async_f_slow
- 1 -[AppDelegate applicationDidFinishLa...
- 2 __CFNOTIFICATIONCENTER_IS_CALL...
- 18 NSApplicationMain
- 19 main
- 20 start

Thread 7

Thread 9

Thread 10



(lldb)

All Output

MyApplication
PID 7632, Paused

- CPU** 7%
- Memory** 12.7 MB
- Energy Impact** Low
- Disk** Zero KB/s
- Network** Zero KB/s

Thread 1
Queue: com.apple.main-thread (serial)

- 0 mach_msg_trap
- 11 NSApplicationMain
- 12 main
- 13 start

Thread 2
Queue: com.apple.libdispatch-manager (serial)

Thread 5
Queue: com.apple.root.utility-qos (concurrent)

- 0 __semwait_signal
- 2 usleep
- 3 __45-[AppDelegate applicationDidFini...
- 4 _dispatch_call_block_and_release
- 9 start_wqthread

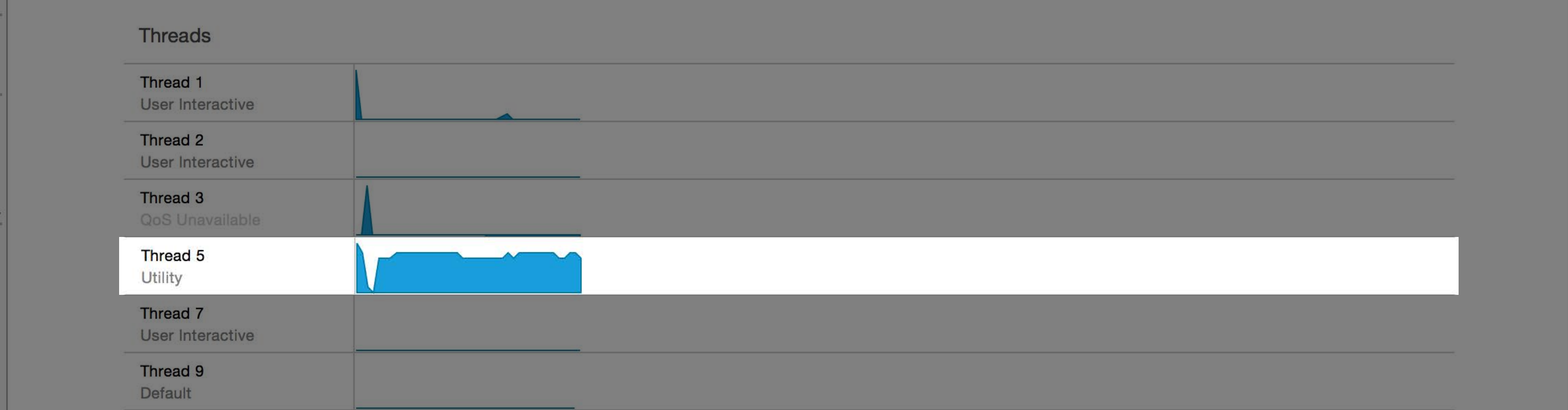
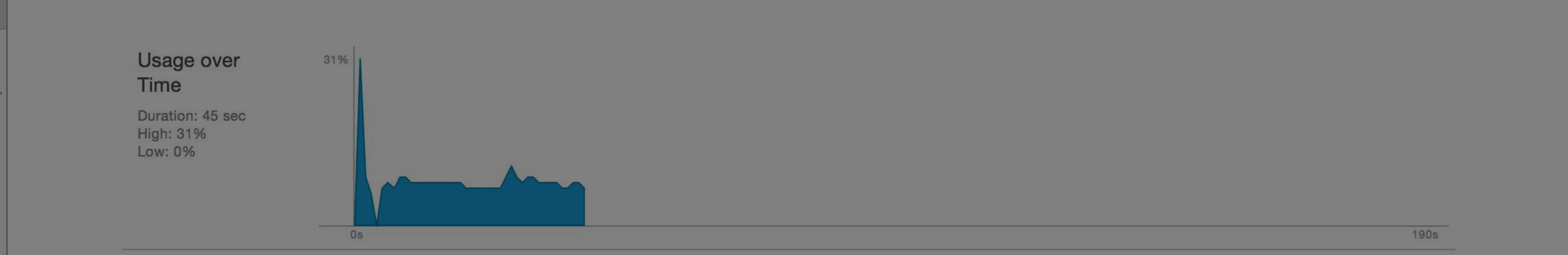
Enqueued from com.apple.main-thread (Thr...)

- 0 _dispatch_async_f_slow
- 1 -[AppDelegate applicationDidFinishLa...
- 2 __CFNOTIFICATIONCENTER_IS_CALL...
- 18 NSApplicationMain
- 19 main
- 20 start

Thread 7

Thread 9

Thread 10



MyApplication > Thread 5 > 0 __semwait_signal

(lldb)

All Output

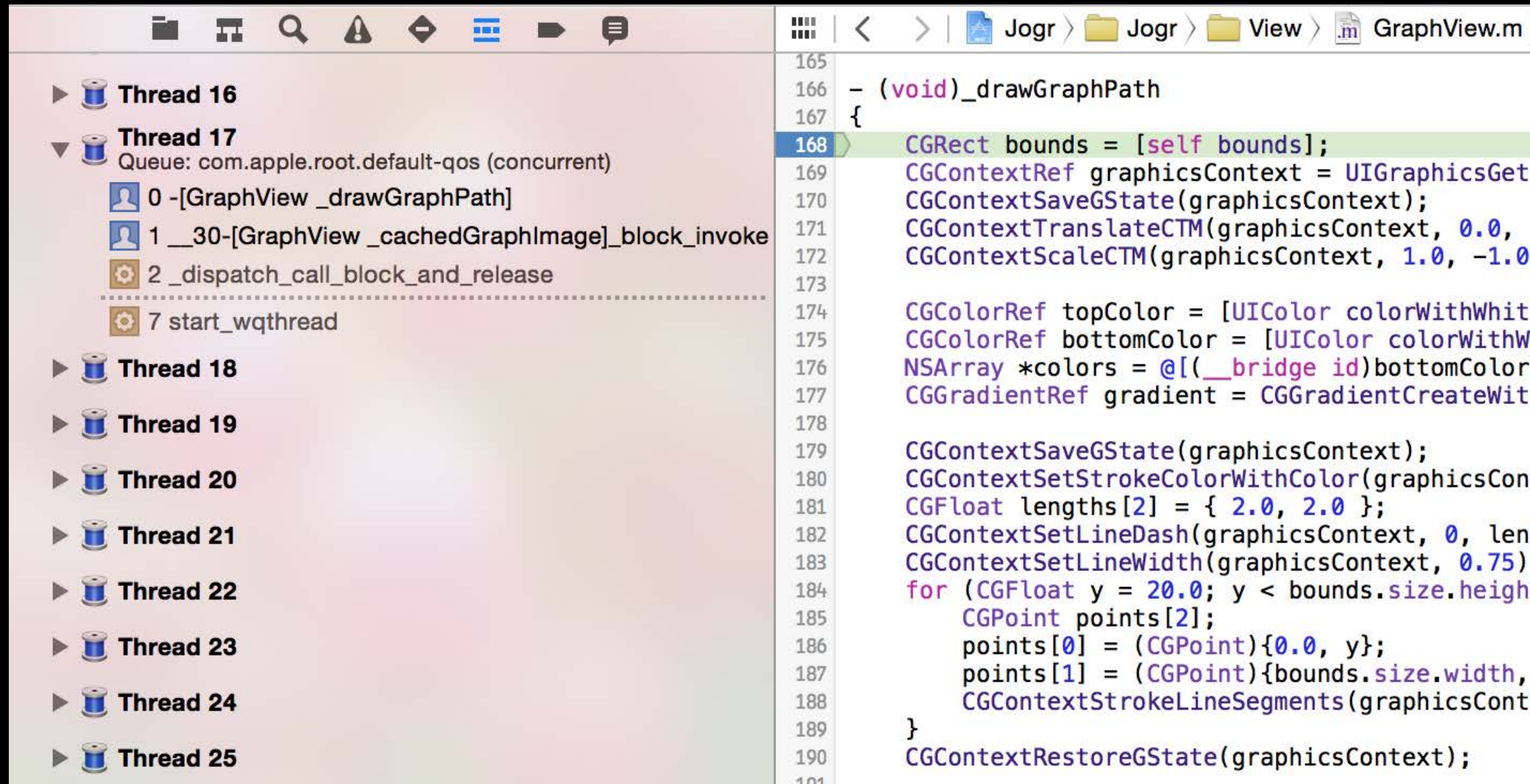
Diagnostics and Queue Debugging

Xcode 6 CPU Report

Xcode 6 Queue Debugging

Activity Tracing

Logical Backtraces



The screenshot displays the Xcode interface with a logical backtrace on the left and the source code on the right. The backtrace shows the call stack for Thread 17, which is currently executing `0 -[GraphView _drawGraphPath]`. The code on the right is from `GraphView.m` and shows the implementation of `-(void)_drawGraphPath`. The current execution point is at line 168, where `CGRect bounds = [self bounds];` is being executed.

```
165
166 - (void)_drawGraphPath
167 {
168     CGRect bounds = [self bounds];
169     CGContextRef graphicsContext = UIGraphicsGet
170     CGContextSaveGState(graphicsContext);
171     CGContextTranslateCTM(graphicsContext, 0.0,
172     CGContextScaleCTM(graphicsContext, 1.0, -1.0
173
174     CGColorRef topColor = [UIColor colorWithWhit
175     CGColorRef bottomColor = [UIColor colorWithW
176     NSArray *colors = @[(__bridge id)bottomColor
177     CGGradientRef gradient = CGGradientCreateWit
178
179     CGContextSaveGState(graphicsContext);
180     CGContextSetStrokeColorWithColor(graphicsCon
181     CGFloat lengths[2] = { 2.0, 2.0 };
182     CGContextSetLineDash(graphicsContext, 0, len
183     CGContextSetLineWidth(graphicsContext, 0.75)
184     for (CGFloat y = 20.0; y < bounds.size.height
185         CGPoint points[2];
186         points[0] = (CGPoint){0.0, y};
187         points[1] = (CGPoint){bounds.size.width,
188         CGContextStrokeLineSegments(graphicsCont
189     }
190     CGContextRestoreGState(graphicsContext);
191
```


Logical Backtraces

The screenshot displays the Xcode interface with a logical backtrace on the left and the source code on the right.

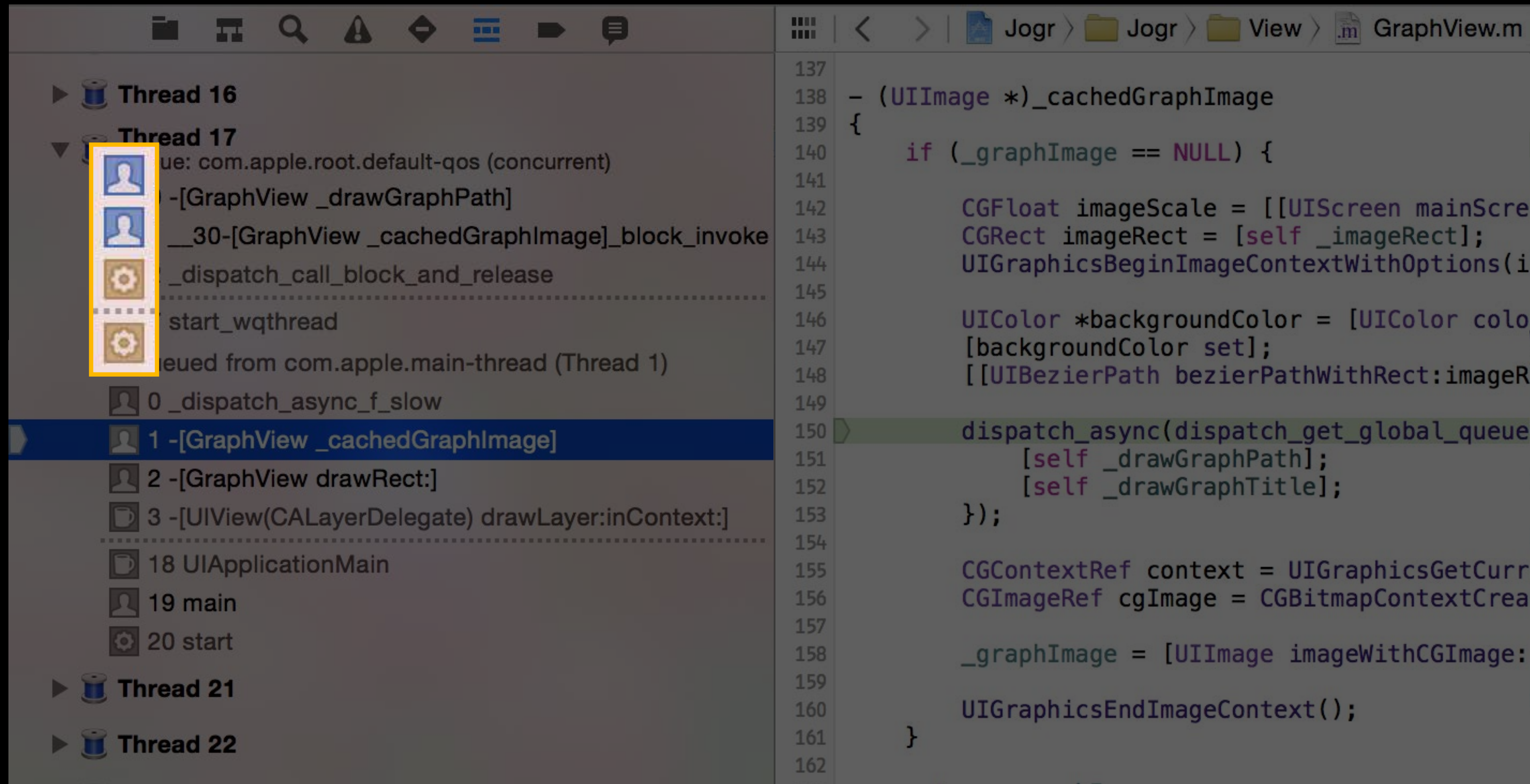
Logical Backtrace (Left Panel):

- Thread 16
- Thread 17
 - Queue: com.apple.root.default-qos (concurrent)
 - 0 -[GraphView _drawGraphPath]
 - 1 __30-[GraphView _cachedGraphImage]_block_invoke
 - 2 _dispatch_call_block_and_release
 - 7 start_wqthread
 - Enqueued from com.apple.main-thread (Thread 1)
 - 0 _dispatch_async_f_slow
 - 1 -[GraphView _cachedGraphImage]** (highlighted)
 - 2 -[GraphView drawRect:]
 - 3 -[UIView(CALayerDelegate) drawLayer:inContext:]
 - 18 UIApplicationMain
 - 19 main
 - 20 start
- Thread 21
- Thread 22

Source Code (Right Panel):

```
137  
138 - (UIImage *)_cachedGraphImage  
139 {  
140     if (_graphImage == NULL) {  
141  
142         CGFloat imageScale = [[UIScreen mainScreen] scale];  
143         CGRect imageRect = [self _imageRect];  
144         UIGraphicsBeginImageContextWithOptions(imageRect, NO, imageScale);  
145  
146         UIColor *backgroundColor = [UIColor colorWithWhite:0.0f alpha:0.0f];  
147         [backgroundColor set];  
148         [[UIBezierPath bezierPathWithRect:imageRect] fill];  
149  
150         dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0),  
151             ^{  
152                 [self _drawGraphPath];  
153                 [self _drawGraphTitle];  
154             });  
155  
156         CGContextRef context = UIGraphicsGetCurrentContext();  
157         CGImageRef cgImage = CGBitmapContextCreateFromContext(context, imageRect.size.width, imageRect.size.height, CGImageAlphaInfoPremultipliedNone);  
158         _graphImage = [UIImage imageWithCGImage:cgImage];  
159  
160         UIGraphicsEndImageContext();  
161     }  
162
```


Logical Backtraces



The screenshot displays the Xcode interface with a logical backtrace on the left and the source code for `GraphView.m` on the right.

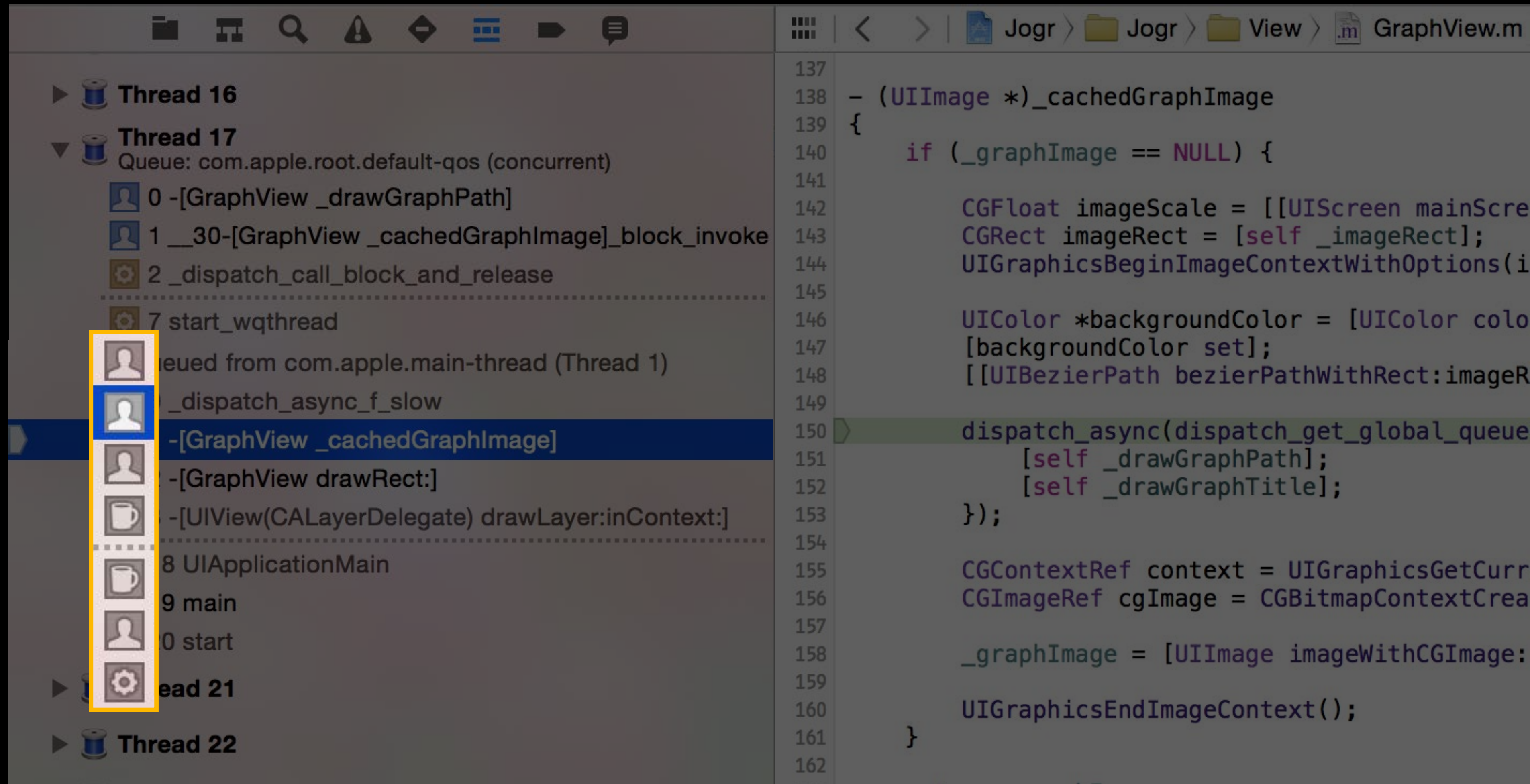
Logical Backtrace (Left Panel):

- Thread 16
- Thread 17
 - Queue: com.apple.root.default-qos (concurrent)
 - 0 -[GraphView _drawGraphPath]
 - 1 -[GraphView _cachedGraphImage]
 - 2 -[GraphView drawRect:]
 - 3 -[UIView(CALayerDelegate) drawLayer:inContext:]
 - 18 UIApplicationMain
 - 19 main
 - 20 start
- Thread 21
- Thread 22

Source Code (Right Panel):

```
137
138 - (UIImage *)_cachedGraphImage
139 {
140     if (_graphImage == NULL) {
141
142         CGFloat imageScale = [[UIScreen mainScreen] scale];
143         CGRect imageRect = [self _imageRect];
144         UIGraphicsBeginImageContextWithOptions(imageRect, NO, imageScale);
145
146         UIColor *backgroundColor = [UIColor colorWithWhite:0.0f opacity:0.0f];
147         [backgroundColor set];
148         [[UIBezierPath bezierPathWithRect:imageRect] fillWithColor:backgroundColor];
149
150         dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
151             [self _drawGraphPath];
152             [self _drawGraphTitle];
153         });
154
155         CGContextRef context = UIGraphicsGetCurrentContext();
156         CGImageRef cgImage = CGContextCreateImage(context, imageRect, imageScale);
157
158         _graphImage = [UIImage imageWithCGImage:cgImage];
159
160         UIGraphicsEndImageContext();
161     }
162
```


Logical Backtraces



The screenshot displays the Xcode interface with a logical backtrace on the left and the source code for `GraphView.m` on the right.

Logical Backtrace (Left Panel):

- Thread 16
- Thread 17
 - Queue: com.apple.root.default-qos (concurrent)
 - 0 -[GraphView _drawGraphPath]
 - 1 __30-[GraphView _cachedGraphImage]_block_invoke
 - 2 _dispatch_call_block_and_release
 - 7 start_wqthread
 - queued from com.apple.main-thread (Thread 1)
 - _dispatch_async_f_slow
 - [GraphView _cachedGraphImage] (highlighted)
 - [GraphView drawRect:]
 - [UIView(CALayerDelegate) drawLayer:inContext:]
 - 8 UIApplicationMain
 - 9 main
 - 10 start
- Thread 21
- Thread 22

Source Code (Right Panel):

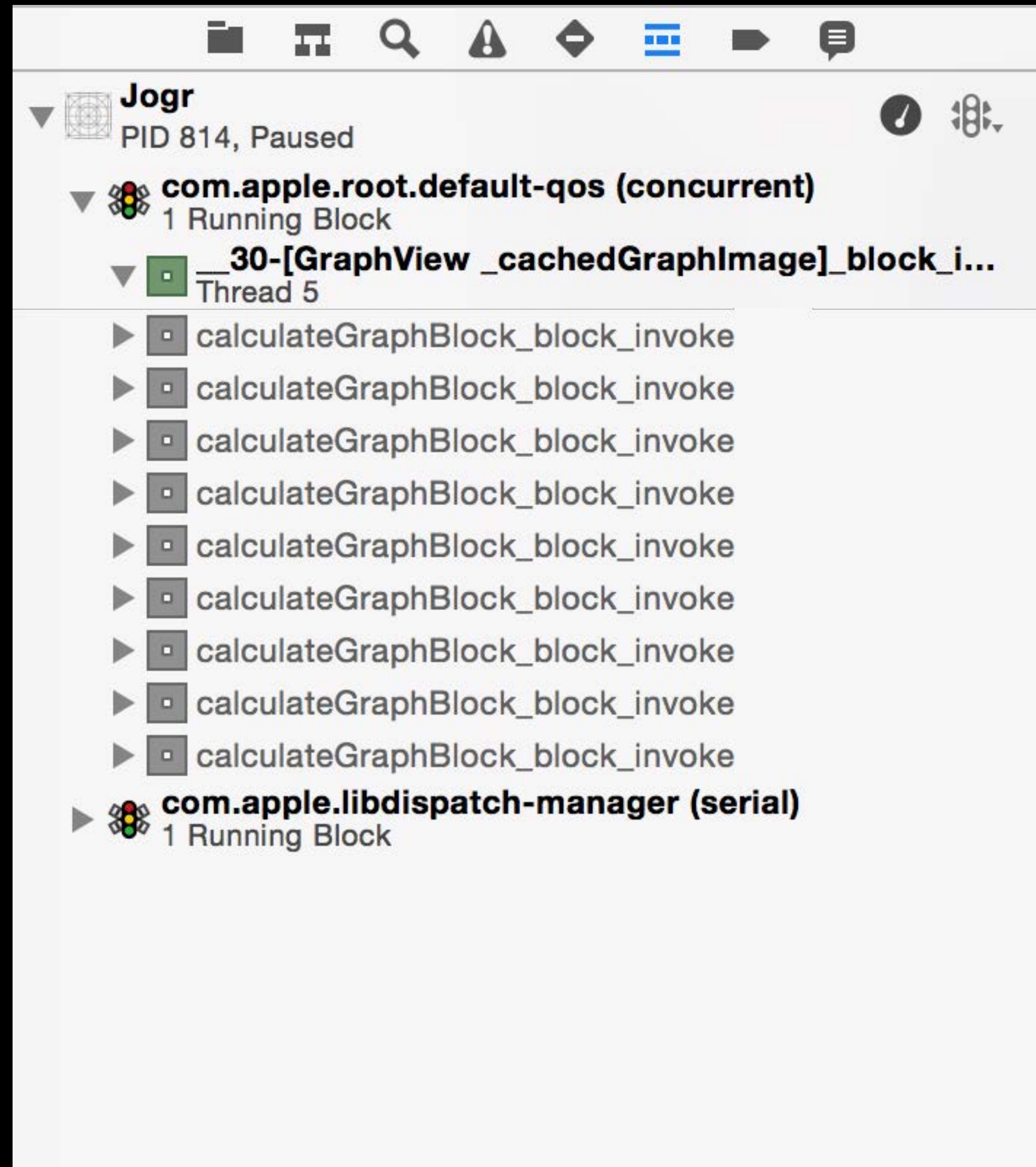
```
137  
138 - (UIImage *)_cachedGraphImage  
139 {  
140     if (_graphImage == NULL) {  
141  
142         CGFloat imageScale = [[UIScreen mainScreen] scale];  
143         CGRect imageRect = [self _imageRect];  
144         UIGraphicsBeginImageContextWithOptions(imageRect, NO, imageScale);  
145  
146         UIColor *backgroundColor = [UIColor colorWithWhite:0.0f opacity:0.5f];  
147         [backgroundColor set];  
148         [[UIBezierPath bezierPathWithRect:imageRect] fill];  
149  
150         dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{  
151             [self _drawGraphPath];  
152             [self _drawGraphTitle];  
153         });  
154  
155         CGContextRef context = UIGraphicsGetCurrentContext();  
156         CGImageRef cgImage = CGBitmapContextCreateFromContext(context, imageRect.size.width, imageRect.size.height, CGImageAlphaInfoPremultipliedNone);  
157  
158         _graphImage = [UIImage imageWithCGImage:cgImage];  
159  
160         UIGraphicsEndImageContext();  
161     }  
162
```


Pending Blocks

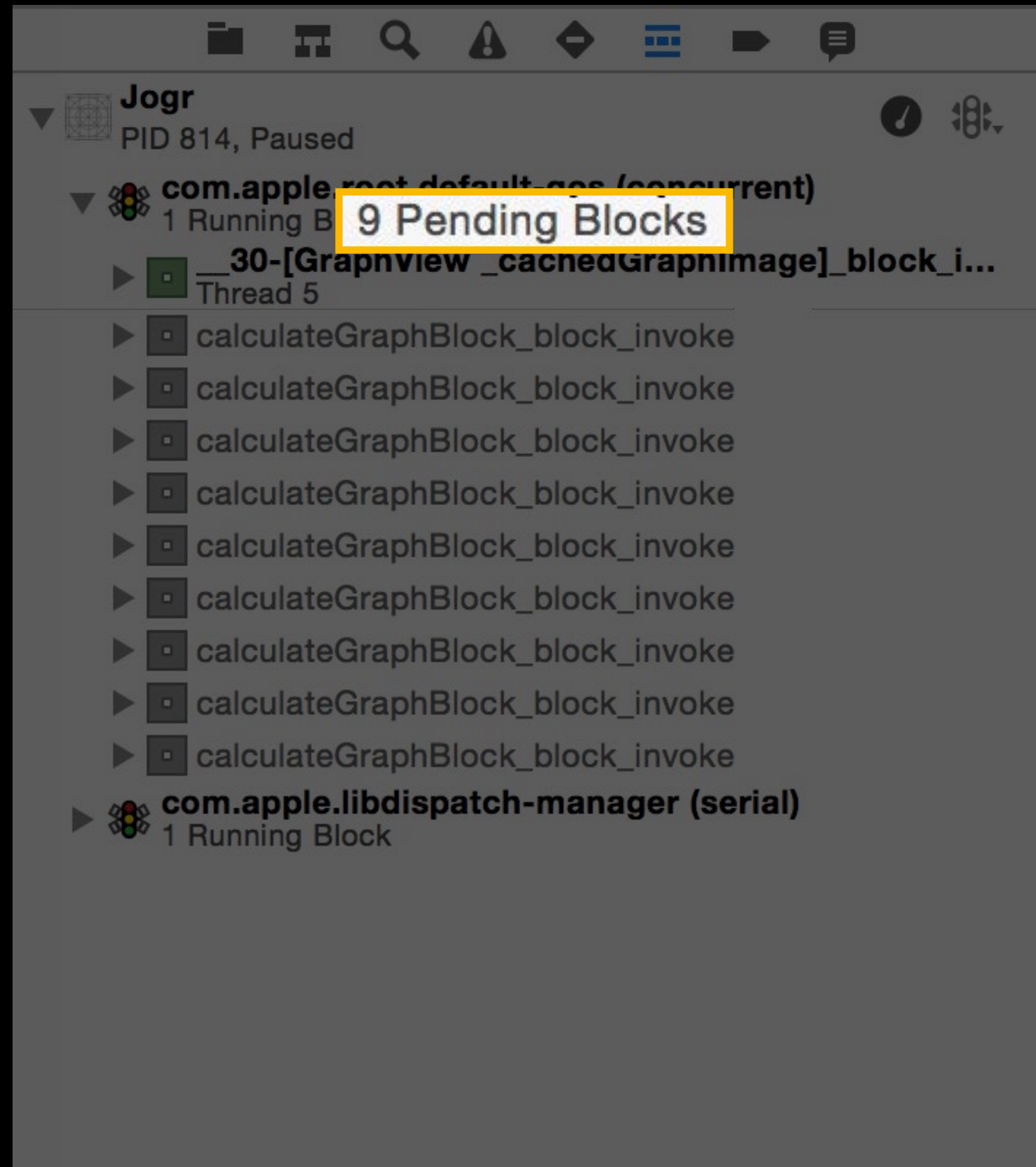
The screenshot shows the Xcode Activity Monitor interface for a process named "Jogr" (PID 814, Paused). The call stack is expanded to show the execution path of a pending block. The top of the stack is highlighted in blue.

- Jogr** (PID 814, Paused)
- com.apple.root.default-qos (concurrent)**
 - 1 Running Block
 - Thread 5**
 - 0 -[GraphView _drawGraphPath]** (highlighted)
 - 1 __30-[GraphView _cachedGraphImage]_block_i...
 - 2 _dispatch_call_block_and_release
 - 7 start_wqthread
 - Enqueued from com.apple.main-thread (Thread 1)
 - 0 _dispatch_async_f_slow
 - 1 -[GraphView _cachedGraphImage]
 - 2 -[GraphView drawRect:]
 - 3 -[UIView(CALayerDelegate) drawLayer:inContext:]
 - 18 UIApplicationMain
 - 19 main
 - 20 start
- Result cache access queue (concurrent)**
 - 1 Pending Block
- com.apple.main-thread (serial)**
 - 1 Running Block

Pending Blocks



Pending Blocks



Pending Blocks

The screenshot shows the Xcode Activity Monitor interface for a process named 'Jogr' (PID 814, Paused). Under the 'com.apple.root.default-qos (concurrent)' queue, there is one running block and nine pending blocks. The pending blocks are all of the type 'calculateGraphBlock_block_invoke'. The first pending block is expanded to show its call stack, which includes:

- 0 _dispatch_async_f_slow
- 1 -[GridView_cachedGraphImage]
- 2 -[GridView drawRect:]
- 3 -[UIView(CALayerDelegate) drawLayer:inContext:]
- 18 UIApplicationMain
- 19 main
- 20 start

Below the expanded block, there are six more pending 'calculateGraphBlock_block_invoke' blocks. At the bottom of the queue, there is one running block in the 'com.apple.libdispatch-manager (serial)' queue.

Diagnostics and Queue Debugging

Xcode 6 CPU Report

Xcode 6 Queue Debugging

Activity Tracing

Activities in Crash Reports

Crashed Thread: 0 Dispatch queue: com.apple.main-thread

Exception Type: EXC_CRASH (SIGABRT)

Exception Codes: 0x0000000000000000, 0x0000000000000000

Breadcrumb Trail (reverse chronological seconds):

6 Query directory using NSRunLoop (Activity ID: 0x0000008f00000002)

15 Query directory using NSRunLoop

```
0 com.apple.Query-Directory 0x000000001000027e0 -[QDAppDelegate query:foundResults:errors:] + 334
1 com.apple.OpenDirectory 0x00007fff9066ec88 __delegate_callback + 388
2 com.apple.CFOpenDirectory 0x00007fff84750bed _query_perform + 568
3 com.apple.CoreFoundation 0x00007fff8964e1f1 __CFRUNLOOP_IS_CALLING_OUT_TO_A_SOURCE0_PERFORM_FUNCTION__ + 17
. . .
16 com.apple.AppKit 0x00007fff83c01a9e NSApplicationMain + 1778
17 com.apple.Query-Directory 0x0000000100001a82 main + 34
18 libdyld.dylib 0x00007fff874af5c9 start + 1
```

Activity ID: 0x0000008f00000002

Activity Name: sendAction:

Activity Breadcrumb: Query directory using NSRunLoop

Activity Running Time: 6.028601 sec

Activity Failure Reason: none detected

Trace Messages (reverse chronological seconds):

```
5.866399 Query Directory 0x0000000100000000 IPC send
5.866463 Query Directory 0x00000001000027e0 aborting test due to no results
5.866529 Query Directory 0x0000000100000000 IPC send
5.866564 Query Directory 0x0000000100002243 skipping record with UID 210 (not member of 'admin')
5.866583 opendiroryd 0xffffffff00000000 IPC send
5.866596 opendiroryd 0xffffffff00021c21 request completed, delivered 1 results
. . .
5.866803 opendiroryd 0xffffffff00021640 UID: 4129, EUID: 4129, GID: 11, EGID: 11
5.866863 opendiroryd 0xffffffff00000000 IPC receive
5.866883 Query Directory 0x0000000100000000 IPC send
```


Activities in Crash Reports

Crashed Thread: 0 Dispatch queue: com.apple.main-thread

Exception Type: EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Breadcrumb Trail (reverse chronological seconds):
6 Query directory using NSRunLoop (Activity ID: 0x0000008f00000002)
15 Query directory using NSRunLoop

Thread 0 Crashed:: Dispatch queue: com.apple.main-thread

Activity ID: 0x0000008f00000002
Activity Name: sendAction:
Activity Breadcrumb: Query directory using NSRunLoop
Activity Running Time: 6.028601 sec
Activity Failure Reason: none detected

Activity Name: sendAction:
Activity Breadcrumb: Query directory using NSRunLoop
Activity Running Time: 6.028601 sec
Activity Failure Reason: none detected

Trace Messages (reverse chronological seconds):
5.866399 Query Directory 0x0000000100000000 IPC send
5.866463 Query Directory 0x00000001000027e0 aborting test due to no results
5.866529 Query Directory 0x0000000100000000 IPC send
5.866564 Query Directory 0x0000000100002243 skipping record with UID 210 (not member of 'admin')
5.866583 opendirirectoryd 0xffffffff00000000 IPC send
5.866596 opendirirectoryd 0xffffffff00021c21 request completed, delivered 1 results
• • •
5.866803 opendirirectoryd 0xffffffff00021640 UID: 4129, EUID: 4129, GID: 11, EGID: 11
5.866863 opendirirectoryd 0xffffffff00000000 IPC receive
5.866883 Query Directory 0x0000000100000000 IPC send

Activities in Crash Reports

Crashed Thread: 0 Dispatch queue: com.apple.main-thread

Exception Type: EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Breadcrumb Trail (reverse chronological seconds):
6 Query directory using NSRunLoop (Activity ID: 0x0000008f00000002)
15 Query directory using NSRunLoop

Trace Messages (reverse chronological seconds):

```
5.866399 Query Directory 0x0000000100000000 IPC send
5.866463 Query Directory 0x00000001000027e0 aborting test due to no results
5.866529 Query Directory 0x0000000100000000 IPC send
5.866564 Query Directory 0x0000000100002243 skipping record with UID 210 (not member of 'admin')
5.866583 opendirirectoryd 0xffffffff00000000 IPC send
5.866596 opendirirectoryd 0xffffffff00021c21 request completed, delivered 1 results
. . .
5.866803 opendirirectoryd 0xffffffff00021640 UID: 4129, EUID: 4129, GID: 11, EGID: 11
5.866863 opendirirectoryd 0xffffffff00000000 IPC receive
5.866883 Query Directory 0x0000000100000000 IPC send
```

Activity Running Time: 6.028601 sec
Activity Failure Reason: none detected

Trace Messages (reverse chronological seconds):

```
5.866399 Query Directory 0x0000000100000000 IPC send
5.866463 Query Directory 0x00000001000027e0 aborting test due to no results
5.866529 Query Directory 0x0000000100000000 IPC send
5.866564 Query Directory 0x0000000100002243 skipping record with UID 210 (not member of 'admin')
5.866583 opendirirectoryd 0xffffffff00000000 IPC send
5.866596 opendirirectoryd 0xffffffff00021c21 request completed, delivered 1 results
. . .
5.866803 opendirirectoryd 0xffffffff00021640 UID: 4129, EUID: 4129, GID: 11, EGID: 11
5.866863 opendirirectoryd 0xffffffff00000000 IPC receive
5.866883 Query Directory 0x0000000100000000 IPC send
```

Using lldb

```
(lldb) thread info
thread #1: tid = 0x1c93, 0x00007fff9452a37a
libsystem_kernel.dylib`__pthread_kill + 10, queue = 'com.apple.main-thread',
activity = 'sendAction:', 5 messages, stop reason = signal SIGABRT
```

```
Activity 'sendAction:', 0x4d00000002
```

```
Current Breadcrumb: Query directory using NSRunLoop
```

```
5 trace messages:
```

```
aborting test due to no results
```

```
skipping record with UID 210 (not member of 'admin')
```

```
IPC send
```

```
issued query
```

```
canceling previous query for mode: 0
```

Using lldb

```
(lldb) thread info
thread #1: tid = 0x1c93, 0x00007fff9452a37a
libsystem_kernel.dylib`__pthread_kill + 10, queue = 'com.apple.main-thread',
activity = 'sendAction:', 5 messages, stop reason = signal SIGABRT
```

```
Activity 'sendAction:', 0x4d00000002
```

```
CurrentBreadcrumb: Query directory using NSRunLoop
```

```
5 trace messages:
```

```
  aborting test due to no results
```

```
  skipping record with UID 210 (not member of 'admin')
```

```
  IPC send
```

```
  issued query
```

```
  canceling previous query for mode: 0
```

Summary

Background

Quality of Service Classes

New QoS and GCD API

Propagation of QoS and Execution Context

Diagnostics and Queue Debugging

More Information

Paul Danbold

Core OS Technologies Evangelist

danbold@apple.com

Documentation

Grand Central Dispatch (GCD) Reference

Concurrency Programming Guide

<http://developer.apple.com/>

Apple Developer Forums

<http://devforums.apple.com>

Related Sessions

-
- | | | |
|--|--------------|-------------------|
| ● Improving Your App with Instruments | Nob Hill | Tuesday 4:30PM |
| ● Writing Energy Efficient Code, Part 1 | Russian Hill | Wednesday 10:15AM |
| ● Debugging in Xcode 6 | Marina | Wednesday 10:15AM |
| ● Writing Energy Efficient Code, Part 2 | Russian Hill | Wednesday 11:30AM |
| ● Fix Bugs Faster using Activity Tracing | Russian Hill | Thursday 11:30AM |
-

 WWDC14