Improving Power Efficiency with App Nap

Session 209

Tony Parker
Software Engineer, Cocoa Frameworks

What is App Nap?

How App Nap Works

App Nap API

What is App Nap?

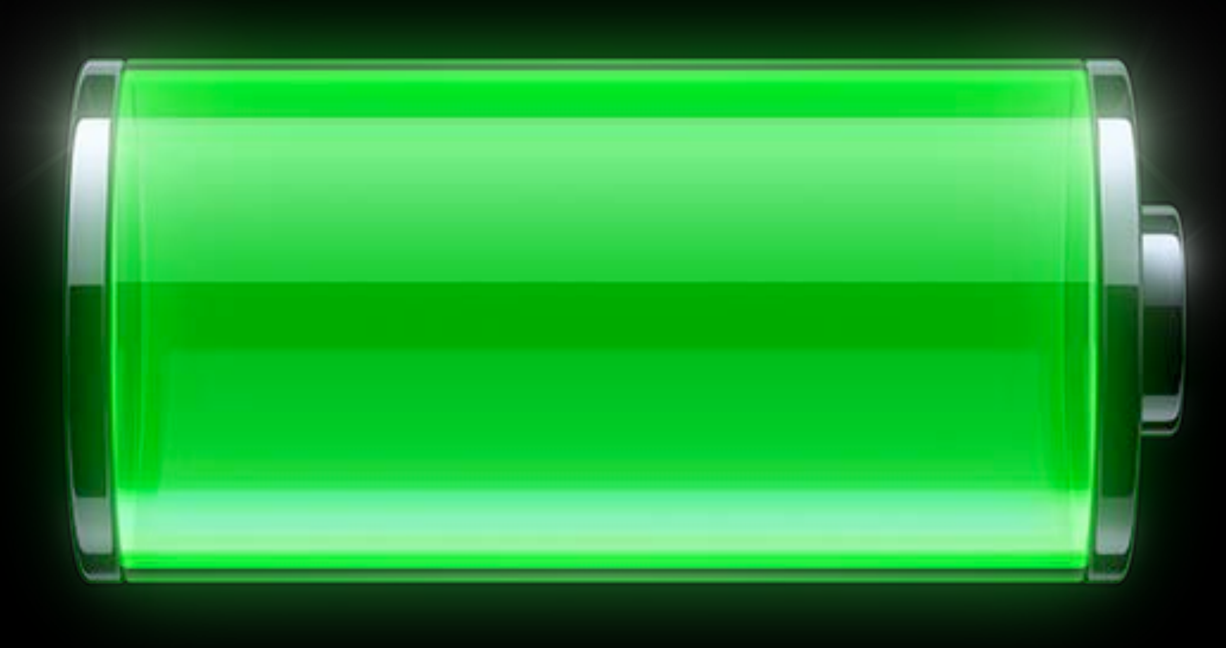
In a World Where…

- Users expect long battery life and high-performance apps

In a World Where…

- Users expect long battery life and high-performance apps
- All apps have about equal access to limited resources
    - CPU time
    - Disk I/O
    - Energy

App Nap focuses system resources on the most important user work.



60
5
10
15
20
25
30
35
40
45
50
55

Battery Life

Responsiveness

App Nap Heuristics

Understanding what’s important

App Nap Heuristics

Understanding what’s important

- Foreground vs. background application

App Nap Heuristics

Understanding what's important

- Foreground vs. background application
- Application type

App Nap Heuristics

Understanding what's important

- Foreground vs. background application
- Application type
- Visibility

App Nap Heuristics

Understanding what's important

- Foreground vs. background application
- Application type
- Visibility
- Drawing activity

App Nap Heuristics

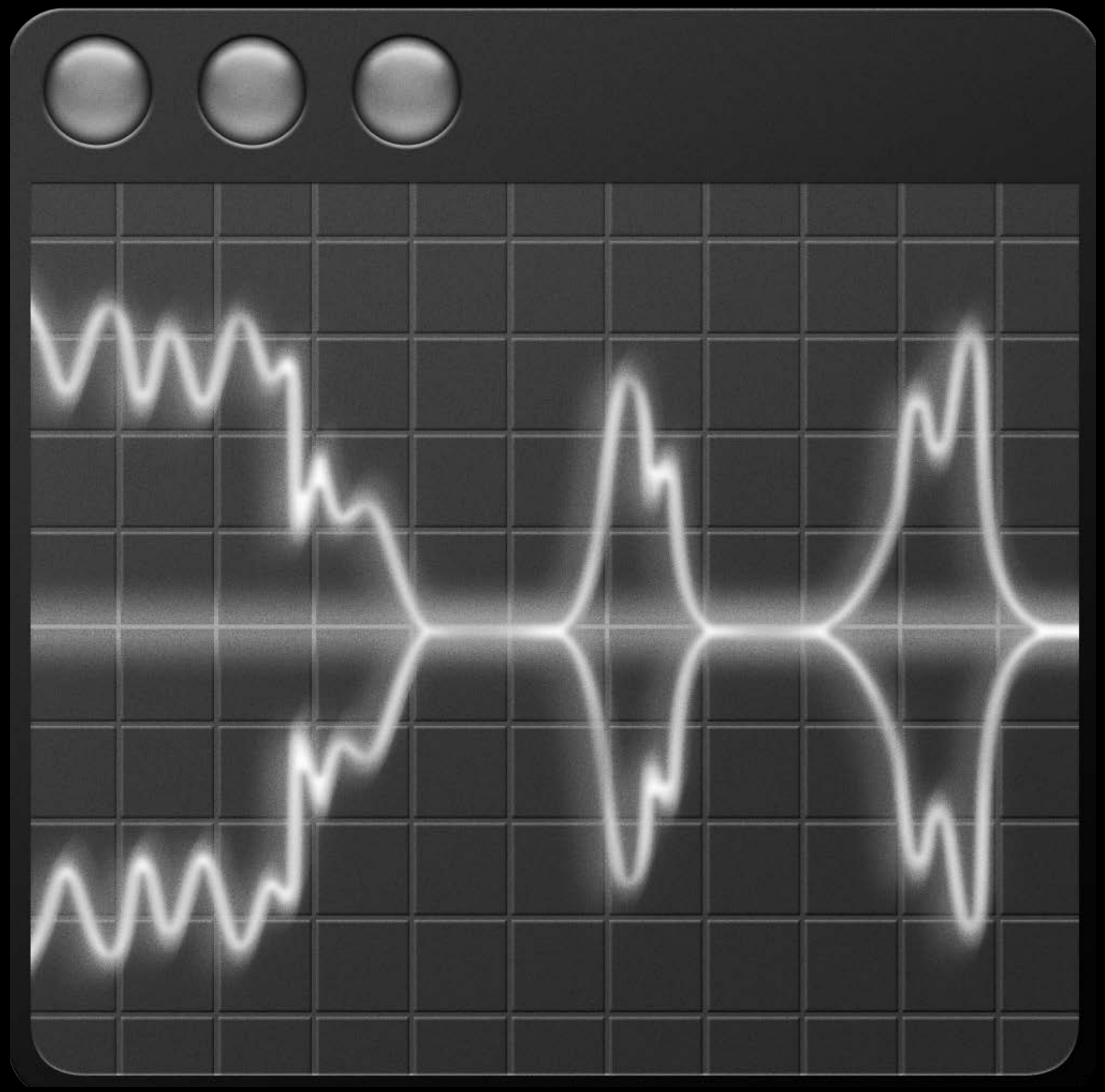
Understanding what's important

- Foreground vs. background application
- Application type
- Visibility
- Drawing activity
- Audio playback

App Nap Heuristics

Understanding what's important

- Foreground vs. background application
- Application type
- Visibility
- Drawing activity
- Audio playback
- Event processing



App Nap Heuristics

Understanding what’s important

- Foreground vs. background application
- Application type
- Visibility
- Drawing activity
- Audio playback
- Event processing
- Use of existing IOKit power assertion API

App Nap Heuristics

Understanding what's important

- Foreground vs. background application
- Application type
- Visibility
- Drawing activity
- Audio playback
- Event processing
- Use of existing IOKit power assertion API
- Use of new App Nap API

Demo

App Nap

How App Nap Works

What’s a Watt?

What’s a Watt?

- Power
  - Rate at which energy is consumed
  - Measured in watts (W)

What’s a Watt?

- Power
  - Rate at which energy is consumed
  - Measured in watts (W)
- Energy
  - Stored potential to do work
  - Measured in watt-hours (Wh)

What’s a Watt?

- 50 watt-hour battery
- 7 hour battery life

What’s a Watt?

- 50 watt-hour battery
- 7 hour battery life

$$\frac{\text{50 watt-hours}}{\text{7 hours}}$$

What's a Watt?

- 50 watt-hour battery
- 7 hour battery life

$$\frac{\text{50 watt-hours}}{\text{7 hours}} \approx \text{7.1 watts}$$

What's a Watt?

- 50 watt-hour battery
- 7 hour battery life

$$\frac{50 \text{ watt-hours}}{7 \text{ hours}} \approx 7.1 \text{ watts}$$

- Screen
- GPU
- Network
- Storage
- Memory
- CPU

What’s a Watt?

- 50 watt-hour battery
- 7 hour battery life

$$\frac{50 \text{ watt-hours}}{7 \text{ hours}} \approx 7.1 \text{ watts}$$

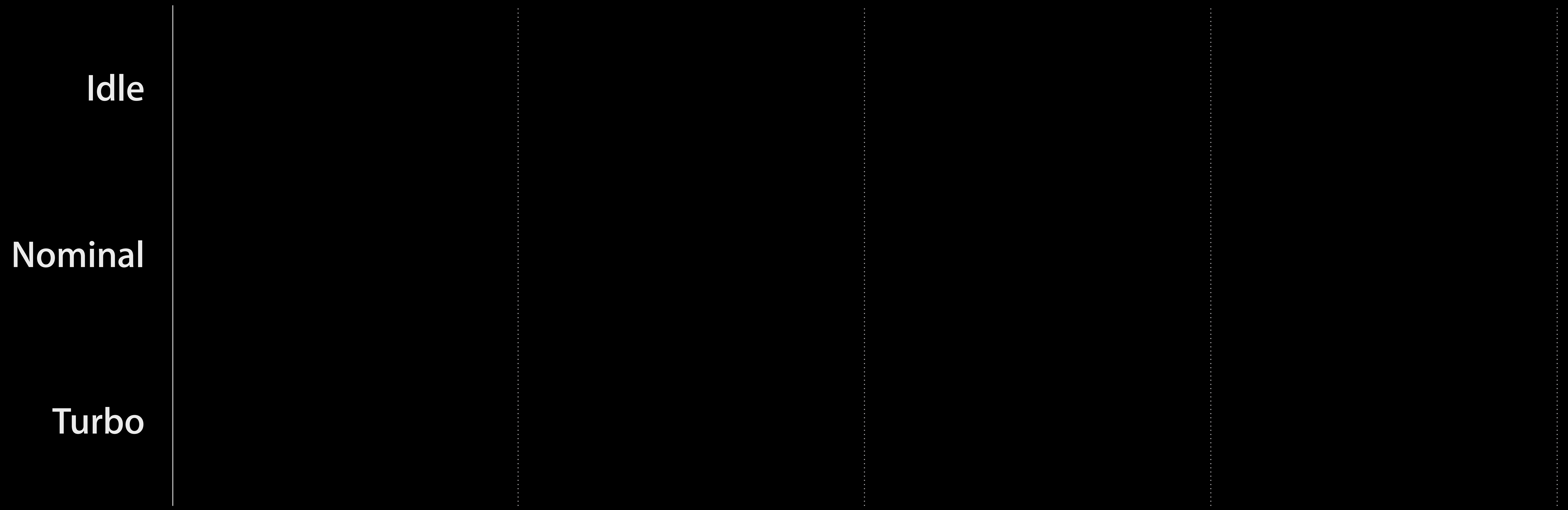
Screen
GPU
Network
Storage
Memory
CPU

CPU Power Usage

What can modern chips do?

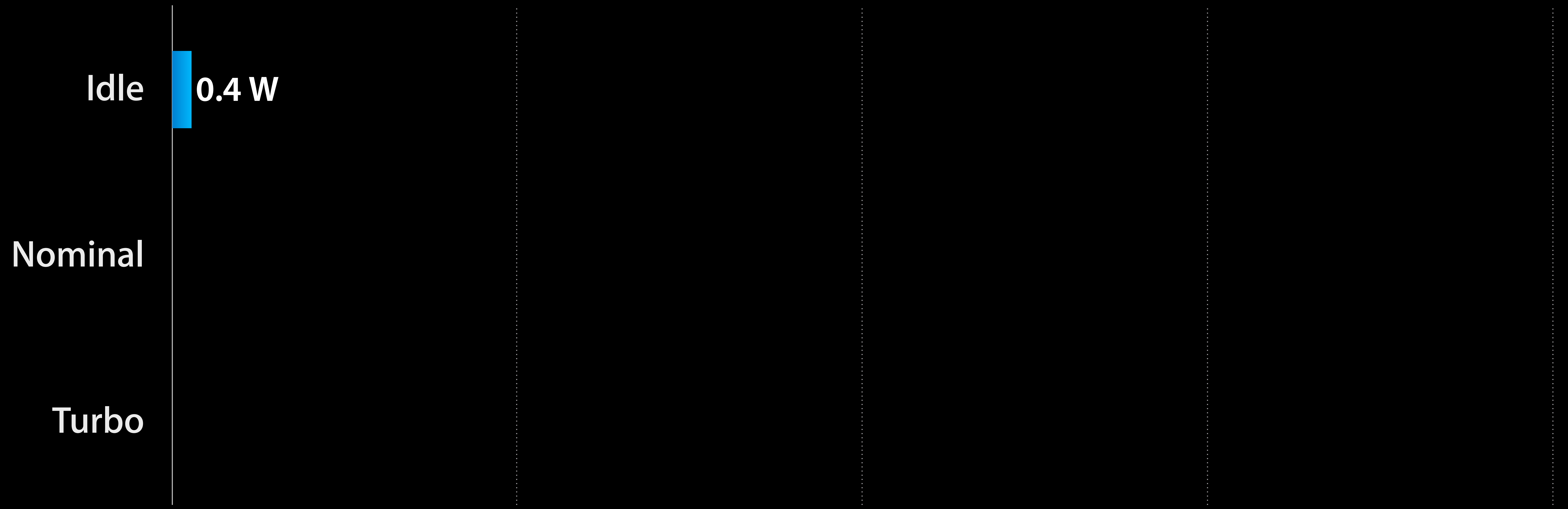
CPU Power Usage

What can modern chips do?



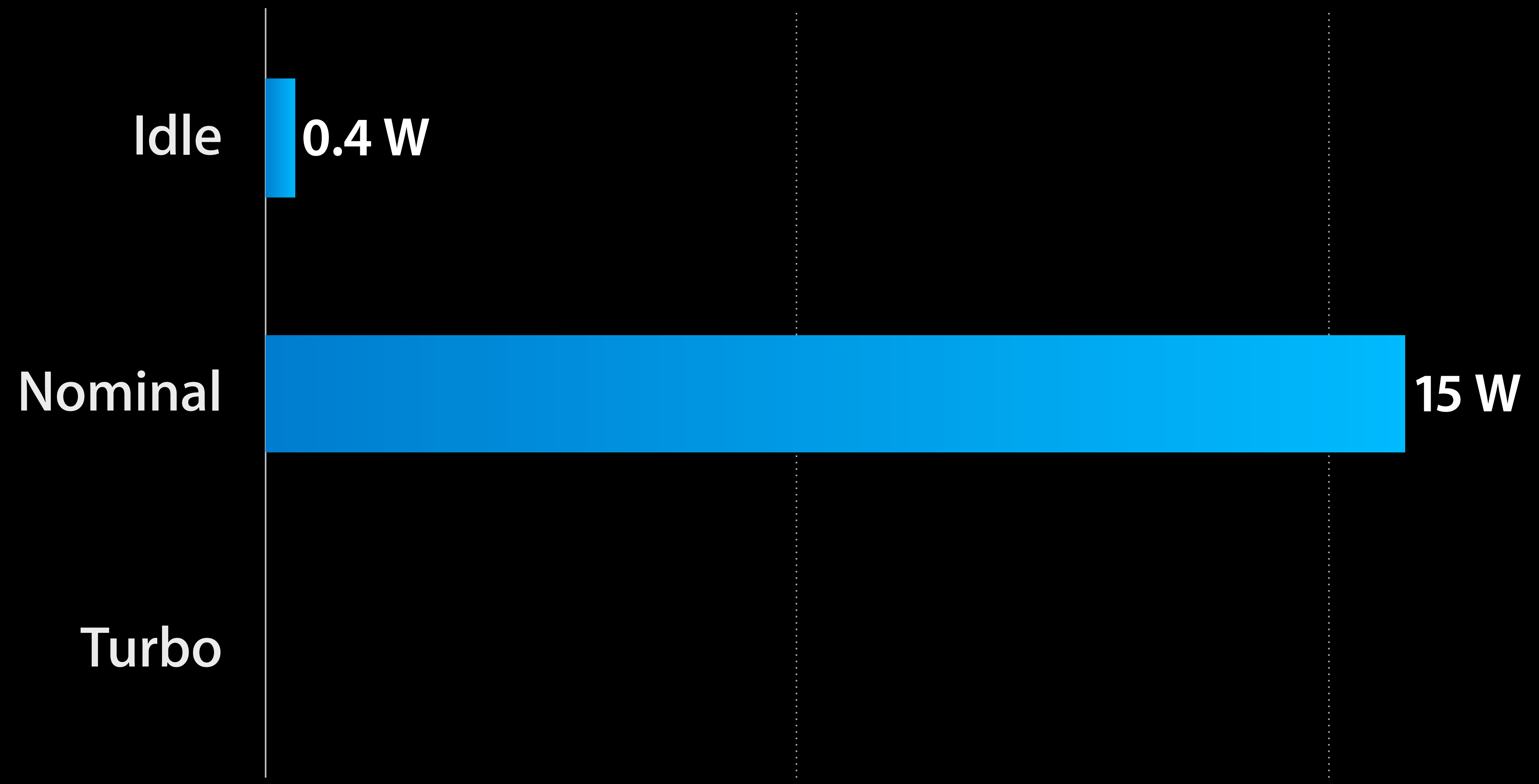
CPU Power Usage

What can modern chips do?



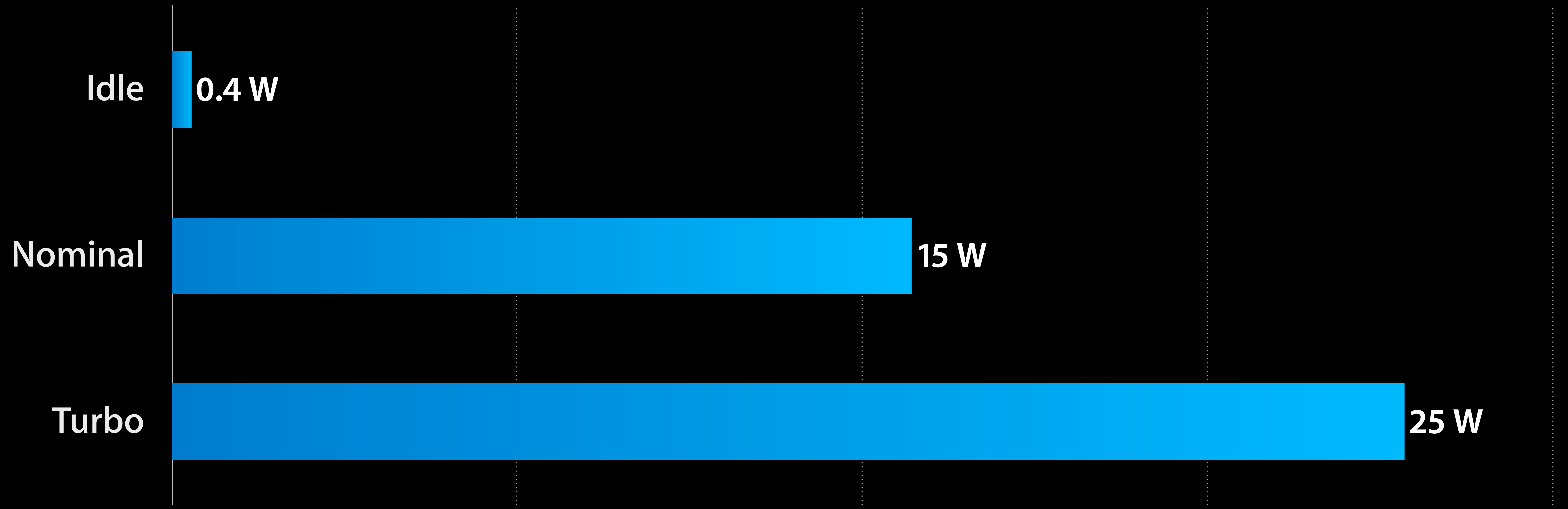
CPU Power Usage

What can modern chips do?



CPU Power Usage

What can modern chips do?



Extending Battery Life

Three key rules

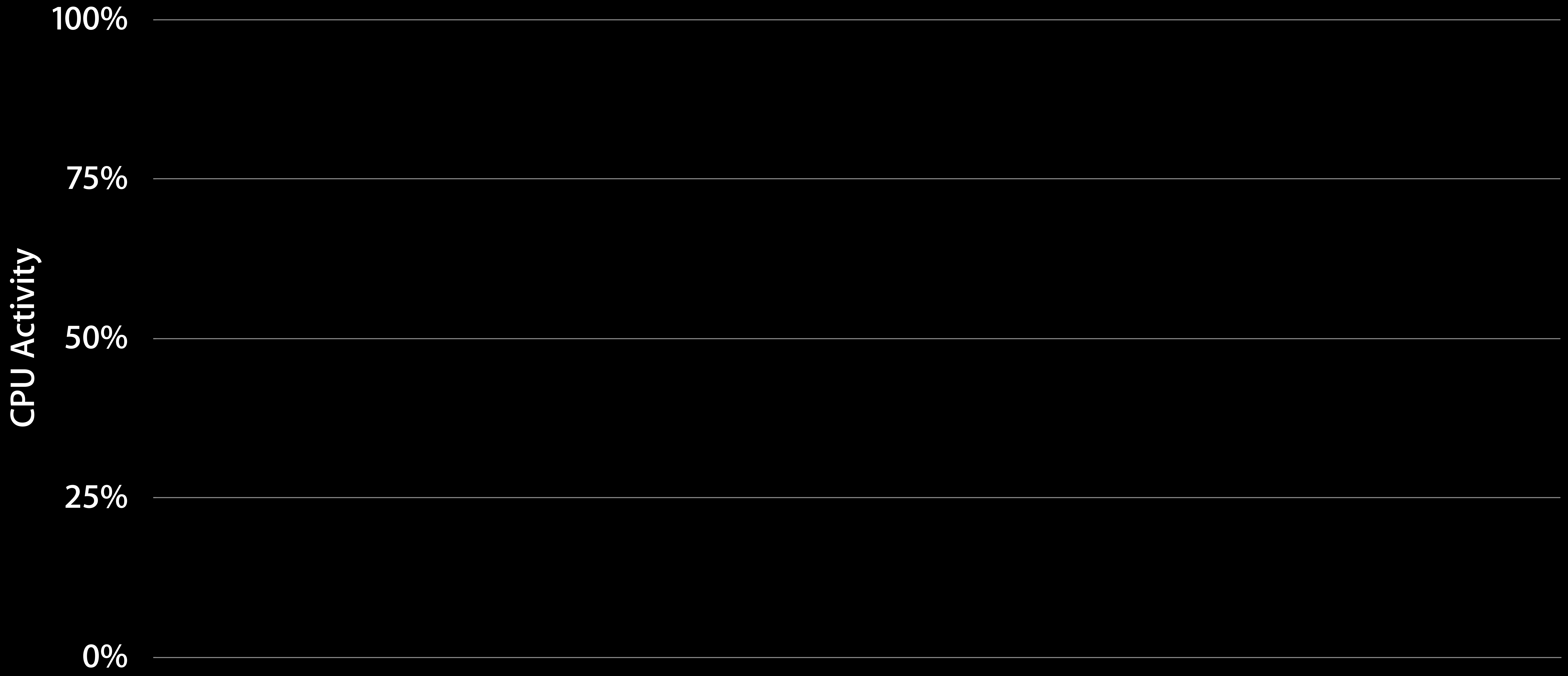
- Stay idle as long as possible
- Avoid unnecessary work
- Return to idle as quickly as possible

Staying Idle: a Case Study

Visiting apple.com in Safari

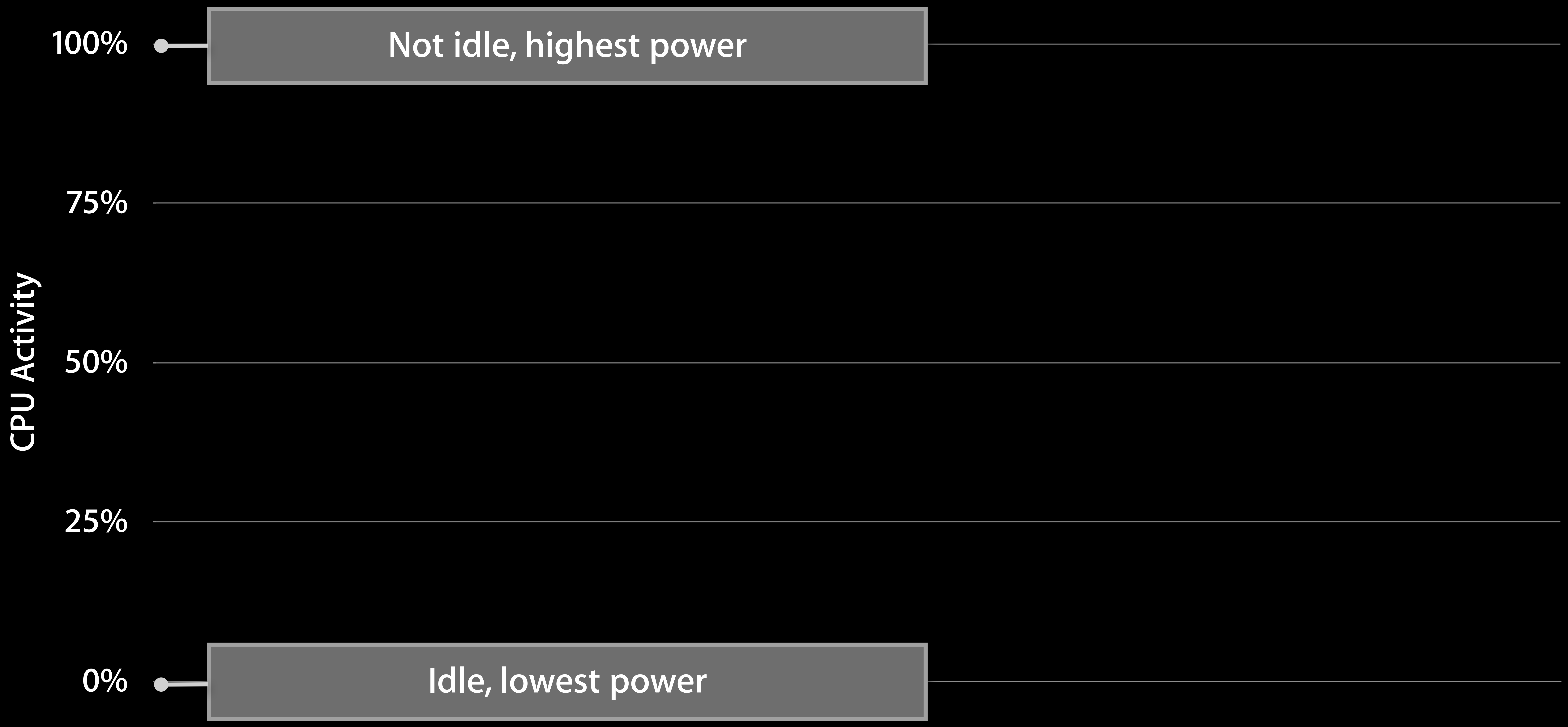
Staying Idle: a Case Study

Visiting apple.com in Safari



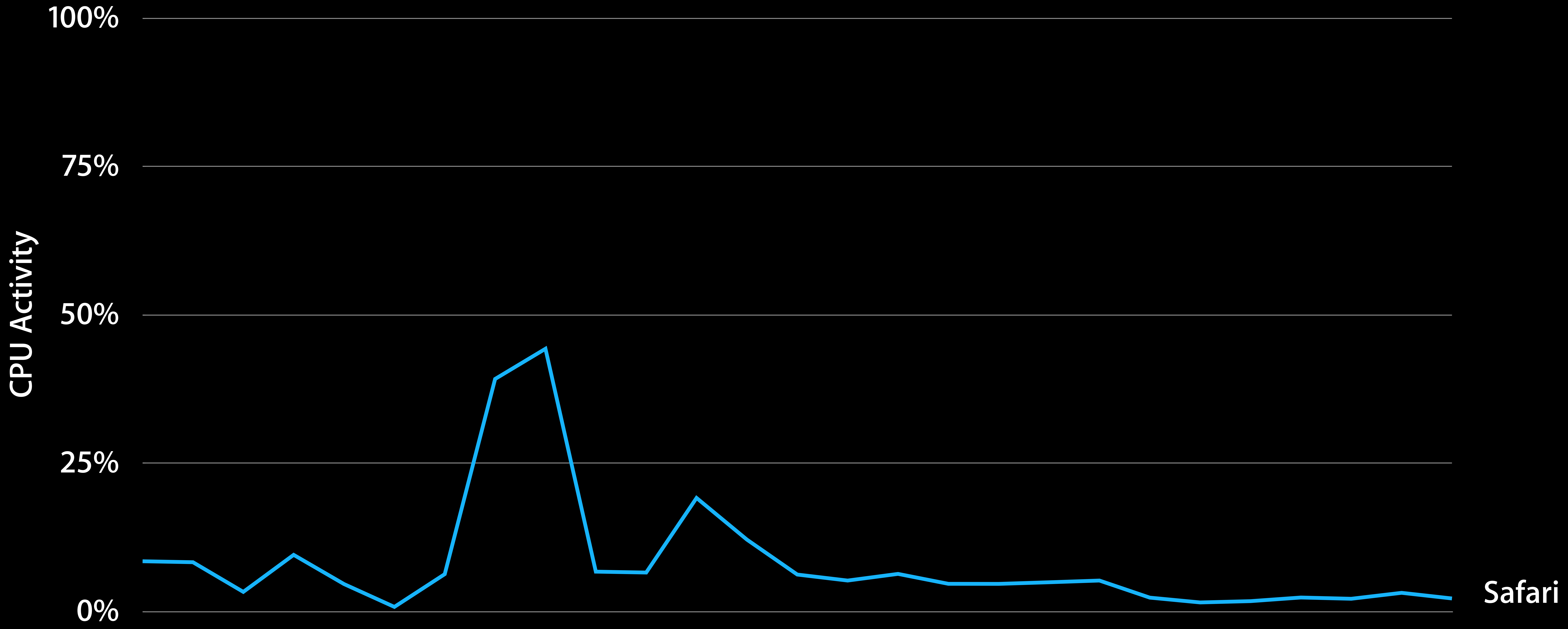
Staying Idle: a Case Study

Visiting apple.com in Safari



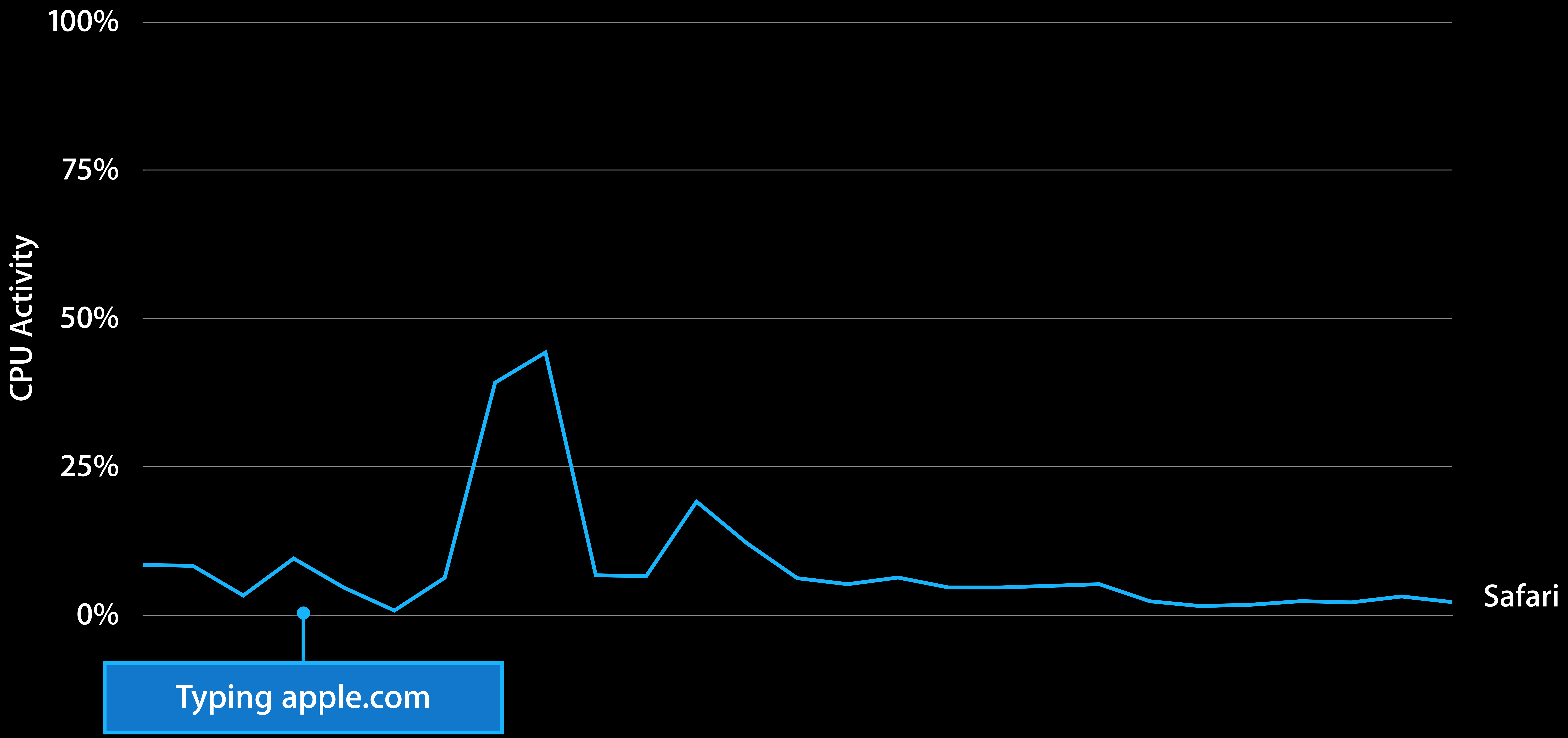
Staying Idle: a Case Study

Visiting apple.com in Safari



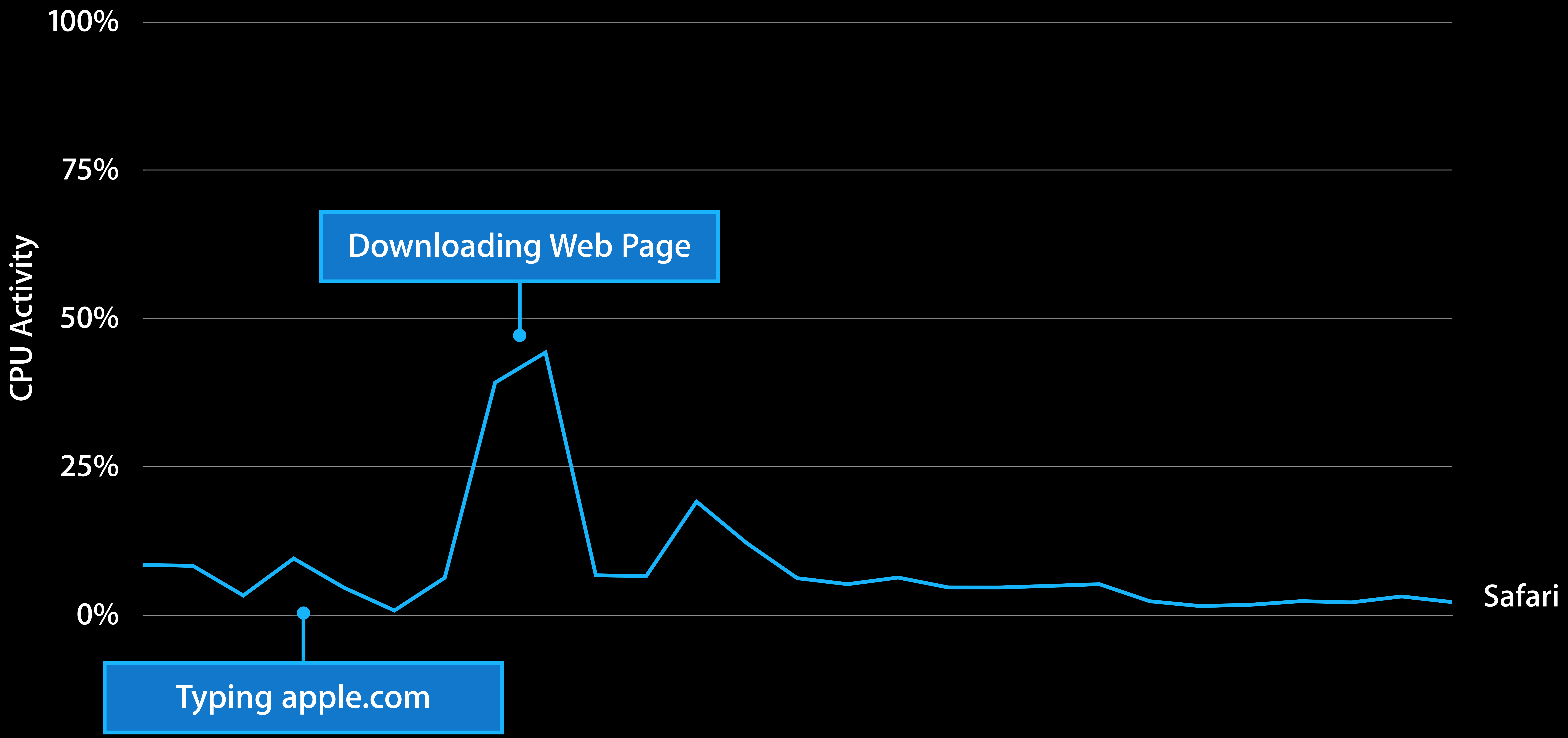
Staying Idle: a Case Study

Visiting apple.com in Safari



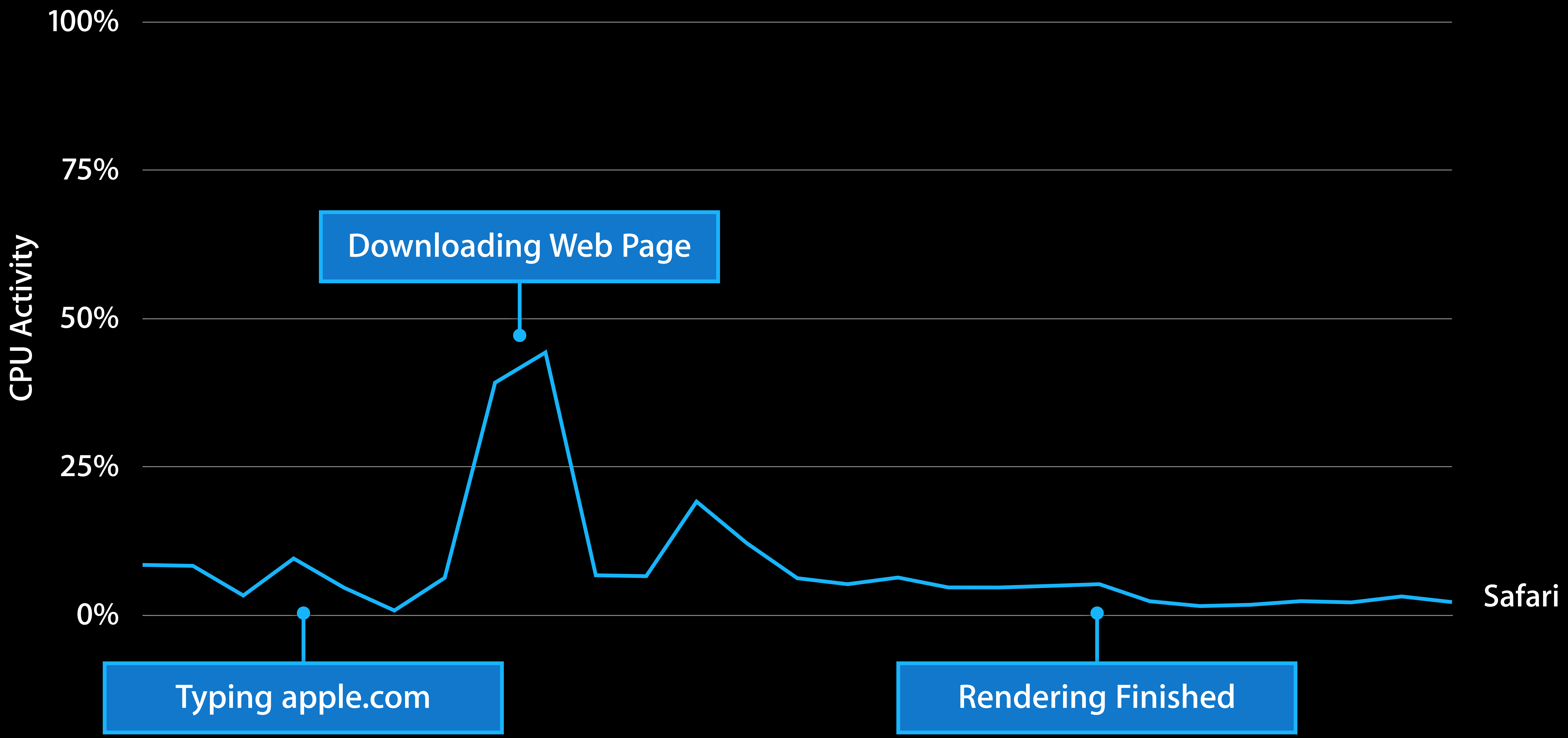
Staying Idle: a Case Study

Visiting apple.com in Safari



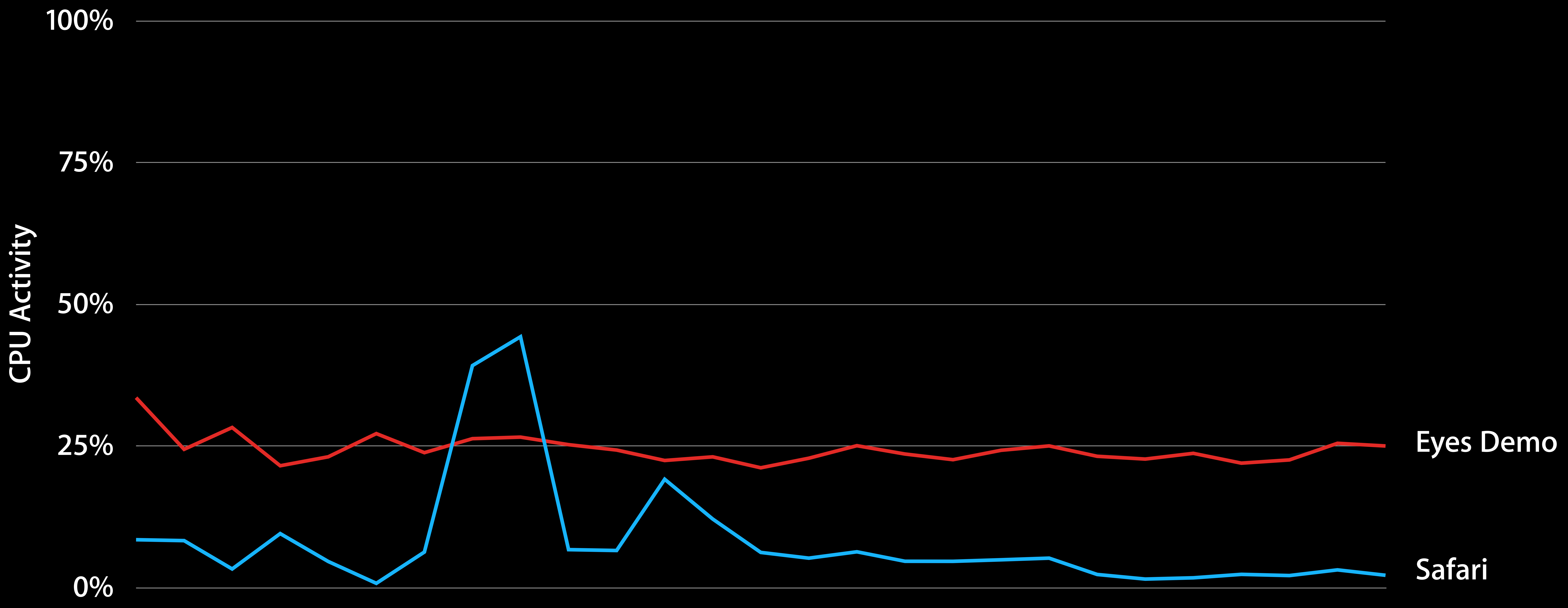
Staying Idle: a Case Study

Visiting apple.com in Safari



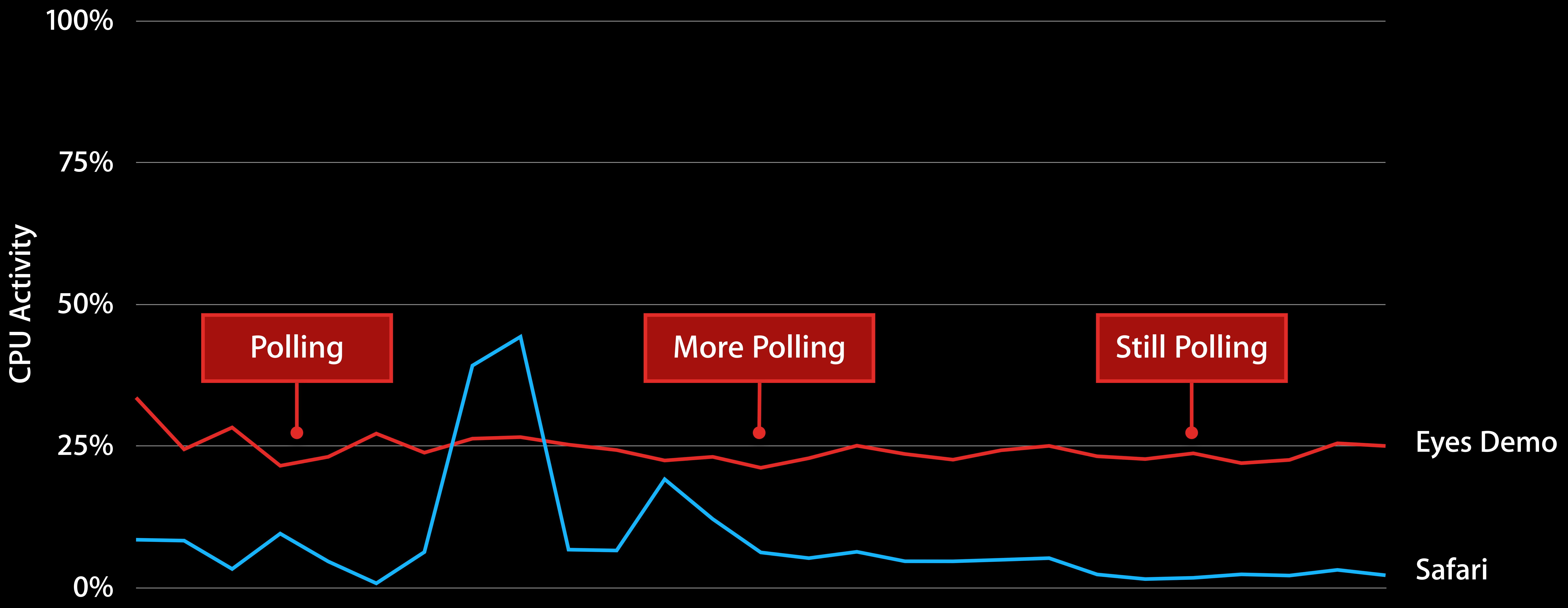
Staying Idle: a Case Study

Visiting apple.com in Safari



Staying Idle: a Case Study

Visiting apple.com in Safari



Exiting Idle

When there is work to do

Exiting Idle

When there is work to do

- Network activity

Exiting Idle

When there is work to do

- Network activity
- Mouse or keyboard input

Exiting Idle

When there is work to do

- Network activity
- Mouse or keyboard input
- Disk I/O

Exiting Idle

When there is work to do

- Network activity
- Mouse or keyboard input
- Disk I/O
- Timers

Exiting Idle

When there is work to do

- Network activity
- Mouse or keyboard input
- Disk I/O
- Timers

API with Timers

Everything with a relative or absolute deadline

API with Timers

Everything with a relative or absolute deadline

```
NSTimer, CFRunLoopTimerRef, DISPATCH_SOURCE_TYPE_TIMER
```

API with Timers

Everything with a relative or absolute deadline

```
NSTimer, CFRunLoopTimerRef, DISPATCH_SOURCE_TYPE_TIMER
```

```
sleep()
```

API with Timers

Everything with a relative or absolute deadline

`NSTimer, CFRunLoopTimerRef, DISPATCH_SOURCE_TYPE_TIMER`

`sleep()`

`pthread_cond_timedwait(), semaphore_timedwait()`

API with Timers

Everything with a relative or absolute deadline

```
NSTimer, CFRunLoopTimerRef, DISPATCH_SOURCE_TYPE_TIMER

sleep()

pthread_cond_timedwait(), semaphore_timedwait()

-[performSelector:withObject:afterDelay:], -[NSRunLoop runUntilDate:]
```

API with Timers

Everything with a relative or absolute deadline

`NSTimer, CFRunLoopTimerRef, DISPATCH_SOURCE_TYPE_TIMER`

`sleep()`

`pthread_cond_timedwait(), semaphore_timedwait()`

`-[performSelector:withObject:afterDelay:], -[NSRunLoop runUntilDate:]`

… and many more

Extending Battery Life

Reducing the impact of timers

Extending Battery Life

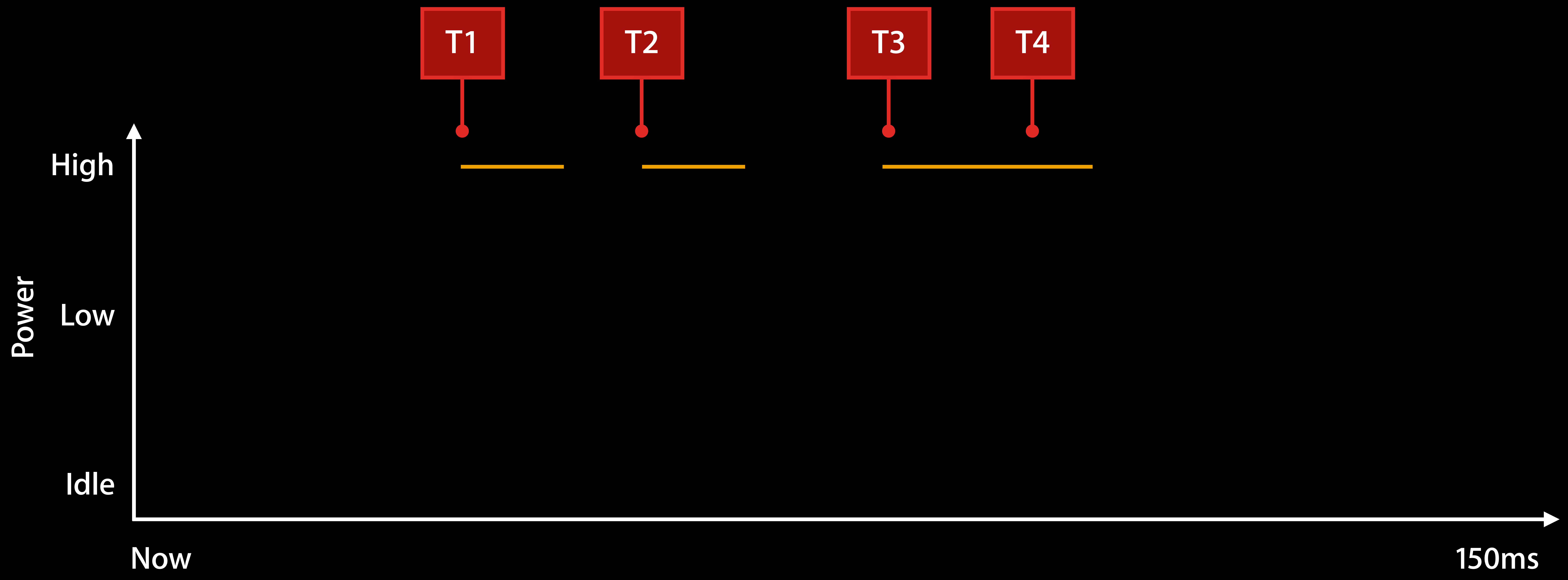
Reducing the impact of timers

- Timer Coalescing
- Timer Rate Limiting

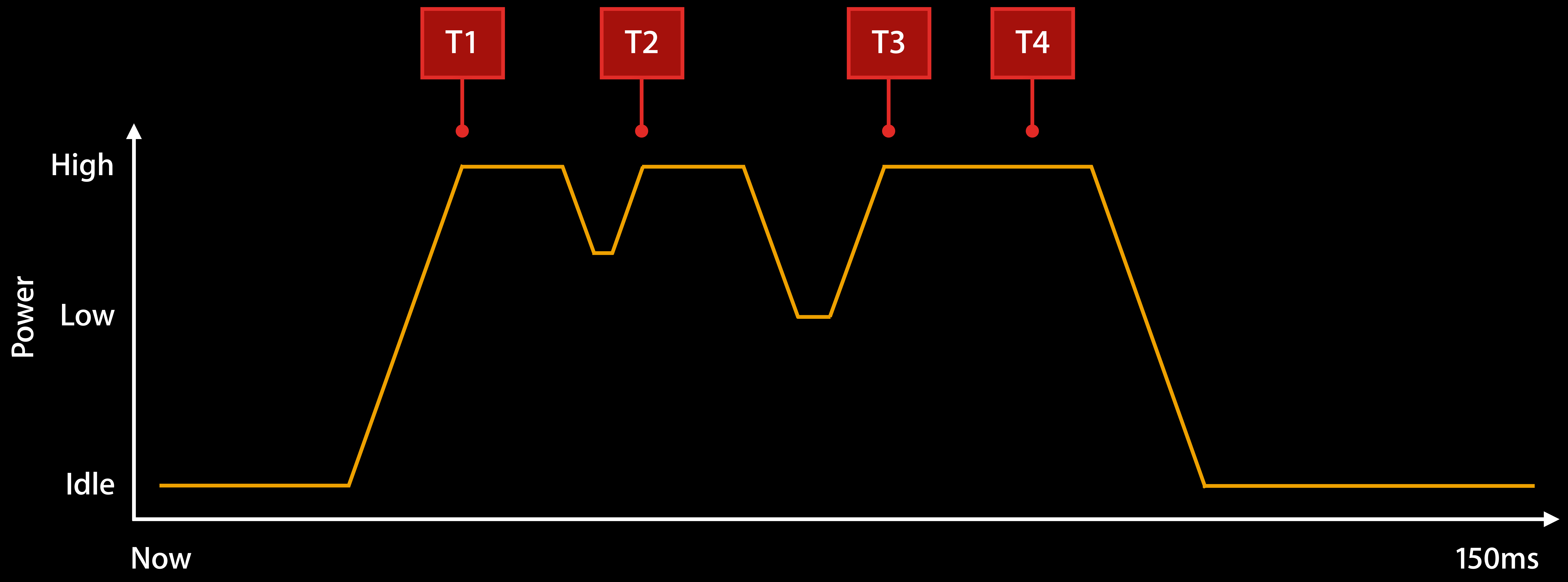
Timer Coalescing

Timer Coalescing

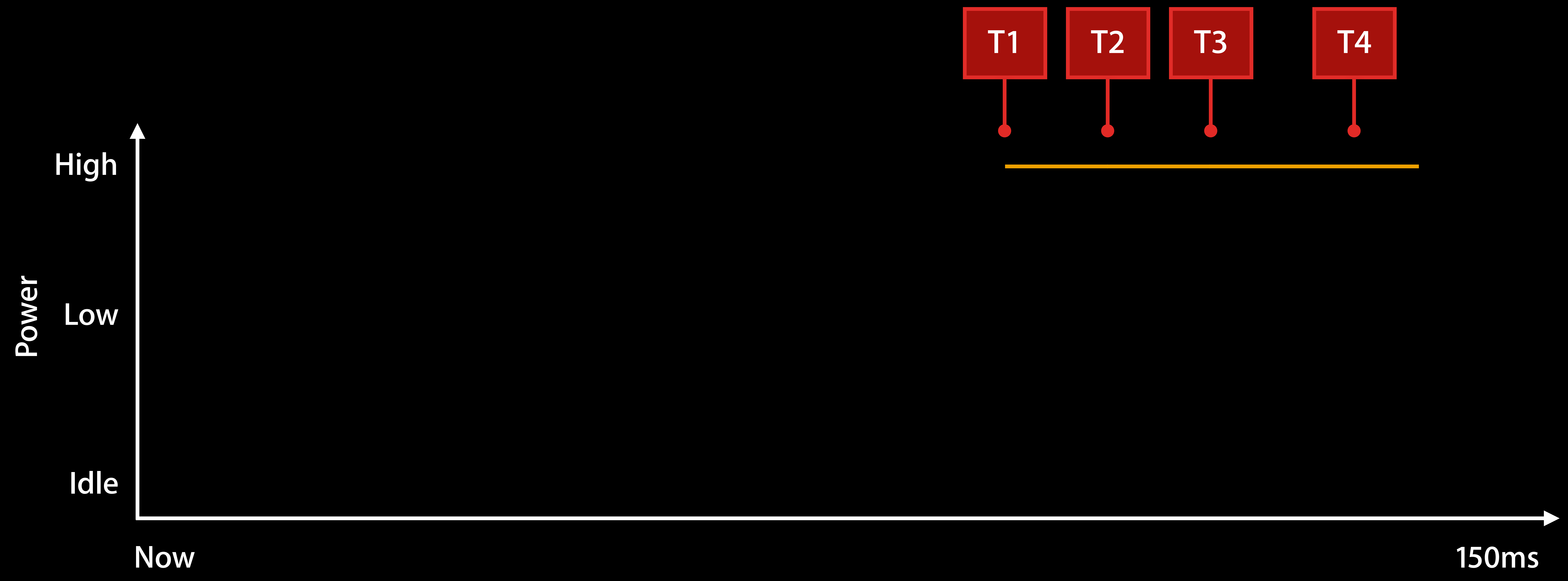
Timer Coalescing



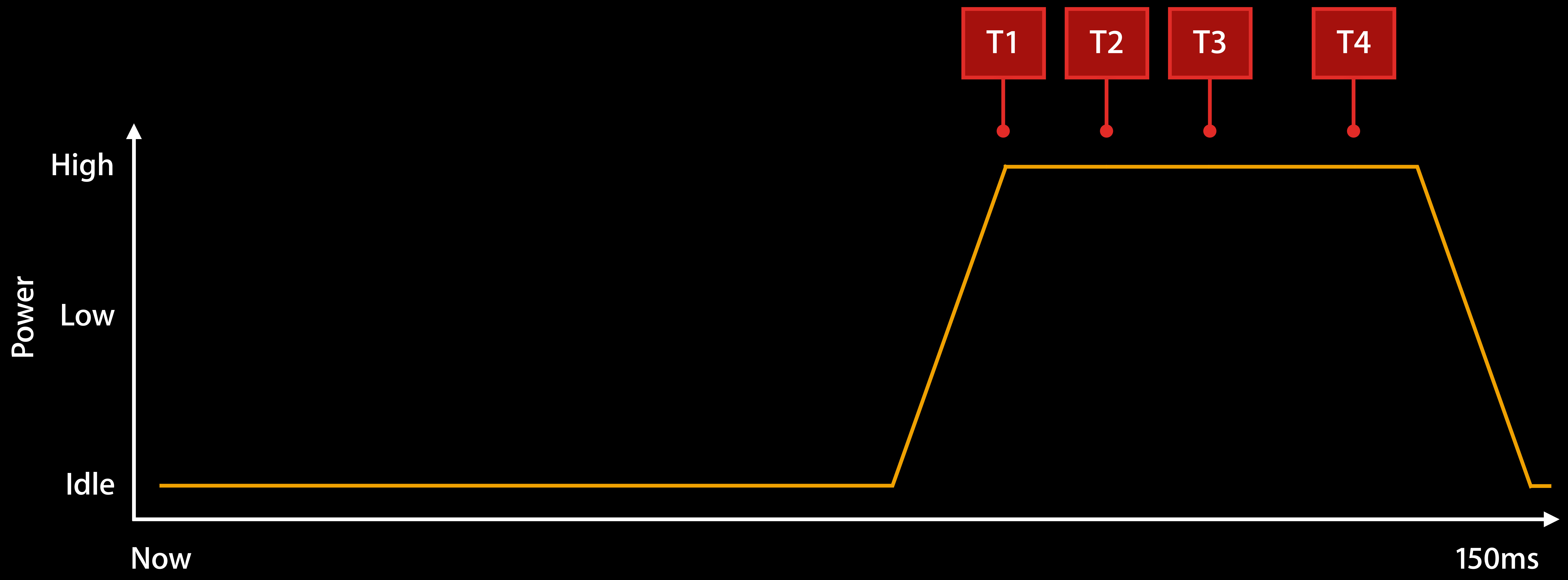
Timer Coalescing



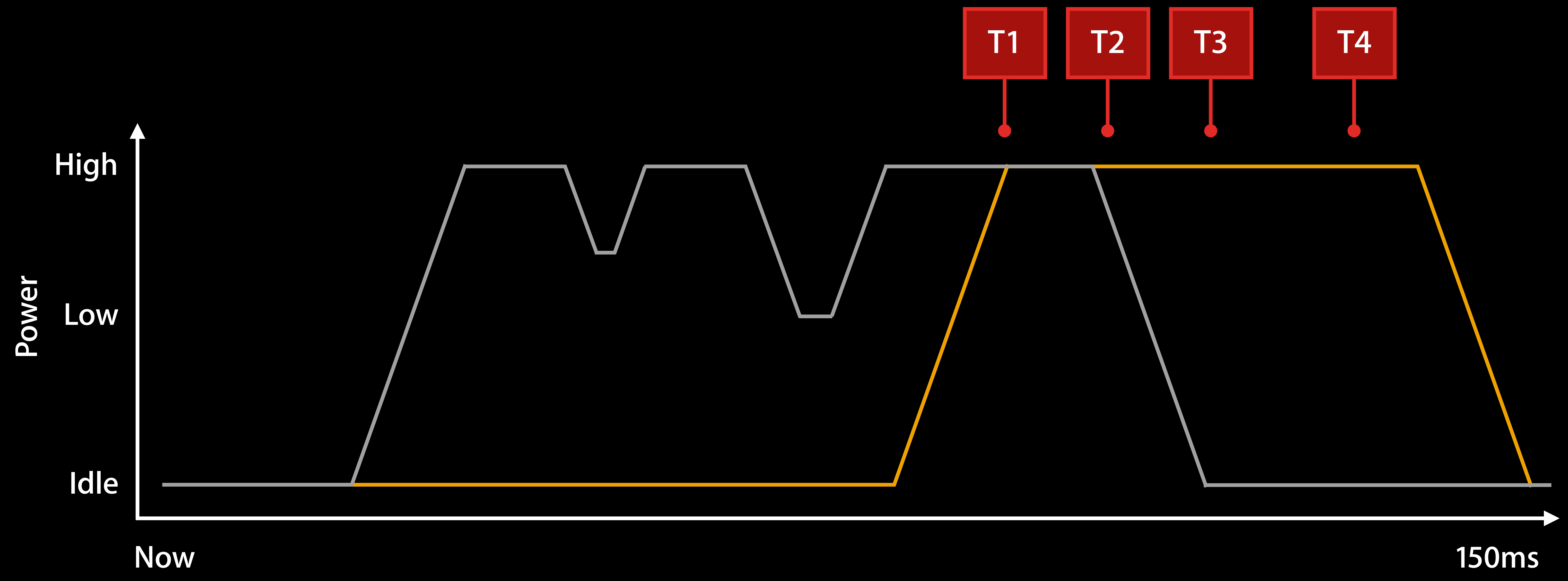
Timer Coalescing



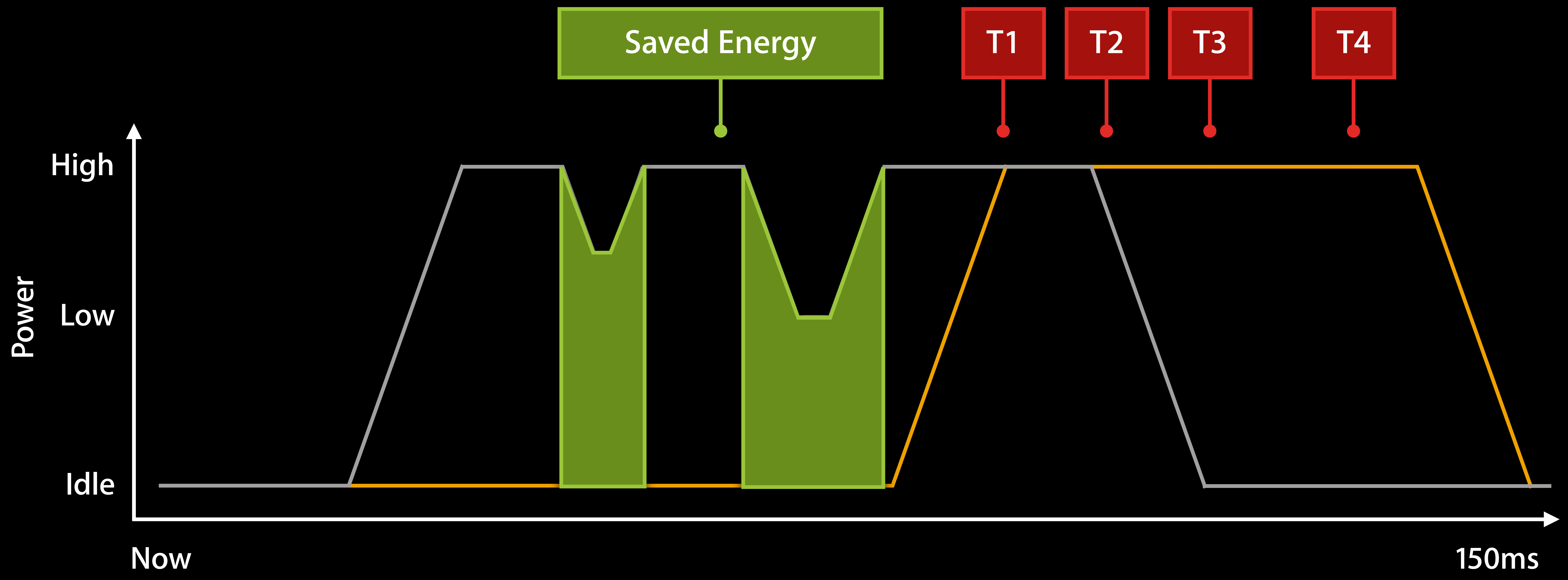
Timer Coalescing



Timer Coalescing



Timer Coalescing



Second Order Effects

After the timer fires

- CPU usage

Second Order Effects

After the timer fires

- CPU usage
- Screen

Second Order Effects

After the timer fires

- CPU usage
- Screen
- GPU usage

Second Order Effects

After the timer fires

- CPU usage
- Screen
- GPU usage
- Network

Second Order Effects

After the timer fires

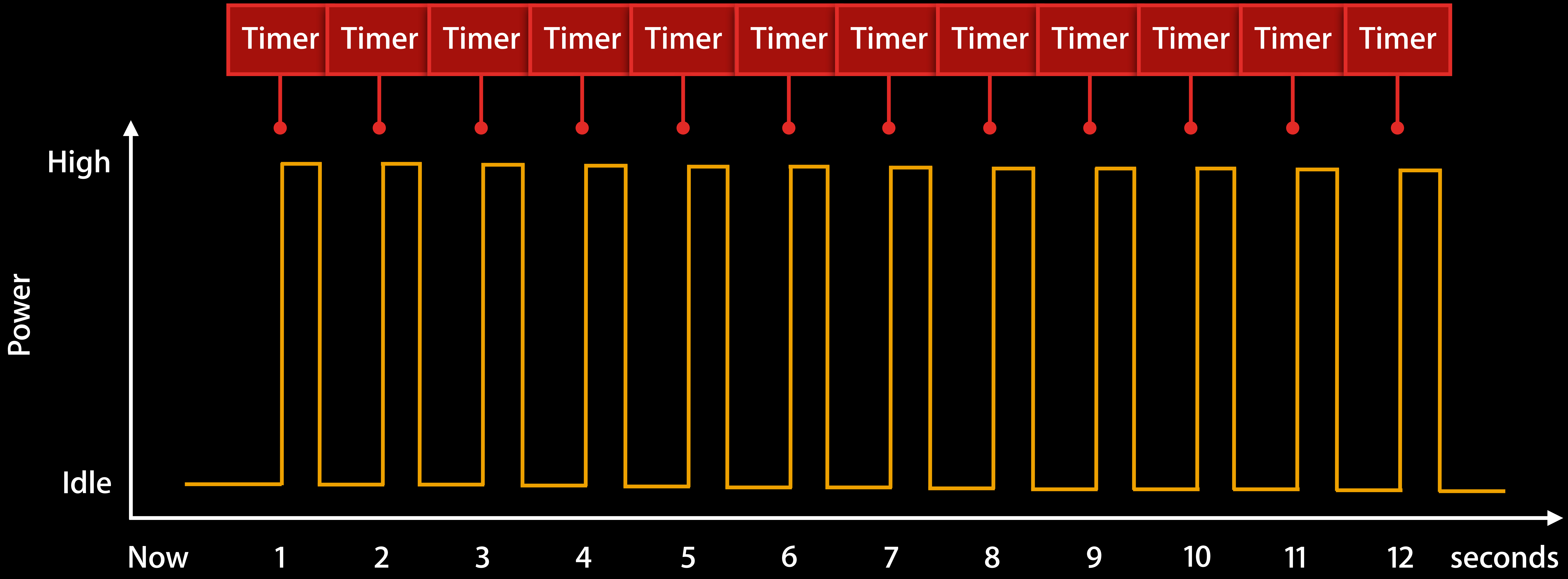
- CPU usage
- Screen
- GPU usage
- Network
- Storage

Second Order Effects

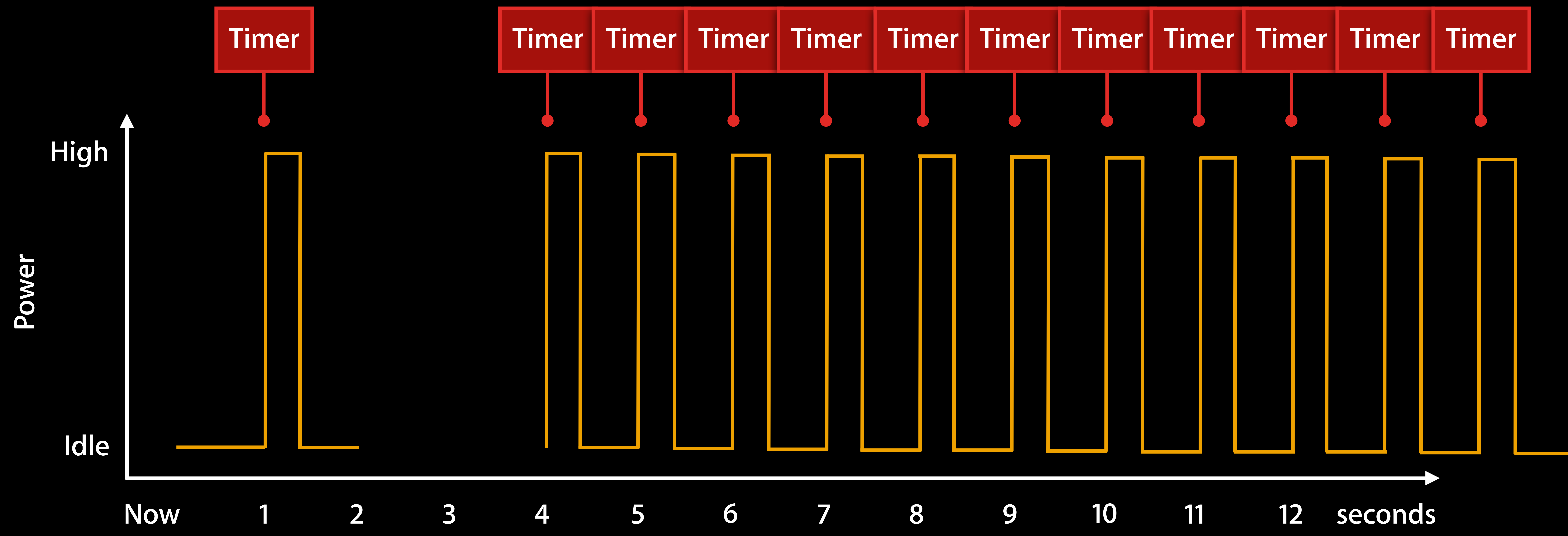
After the timer fires

- CPU usage
- Screen
- GPU usage
- Network
- Storage
- Memory

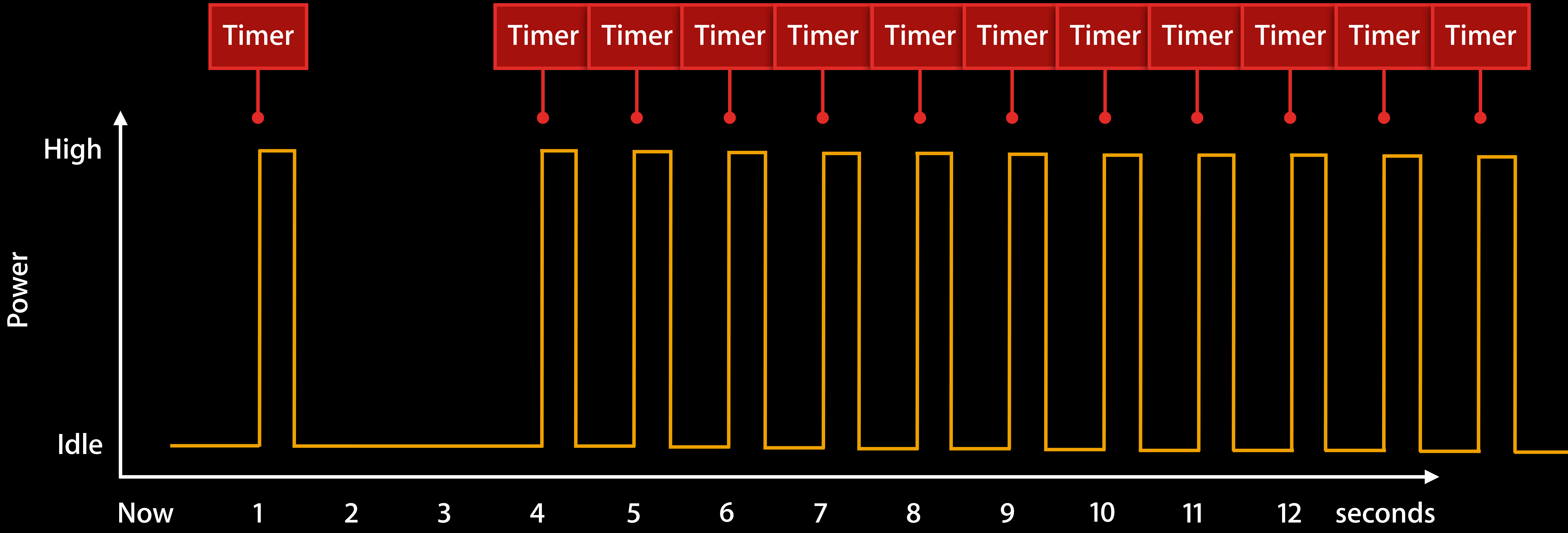
Timer Rate Limiting



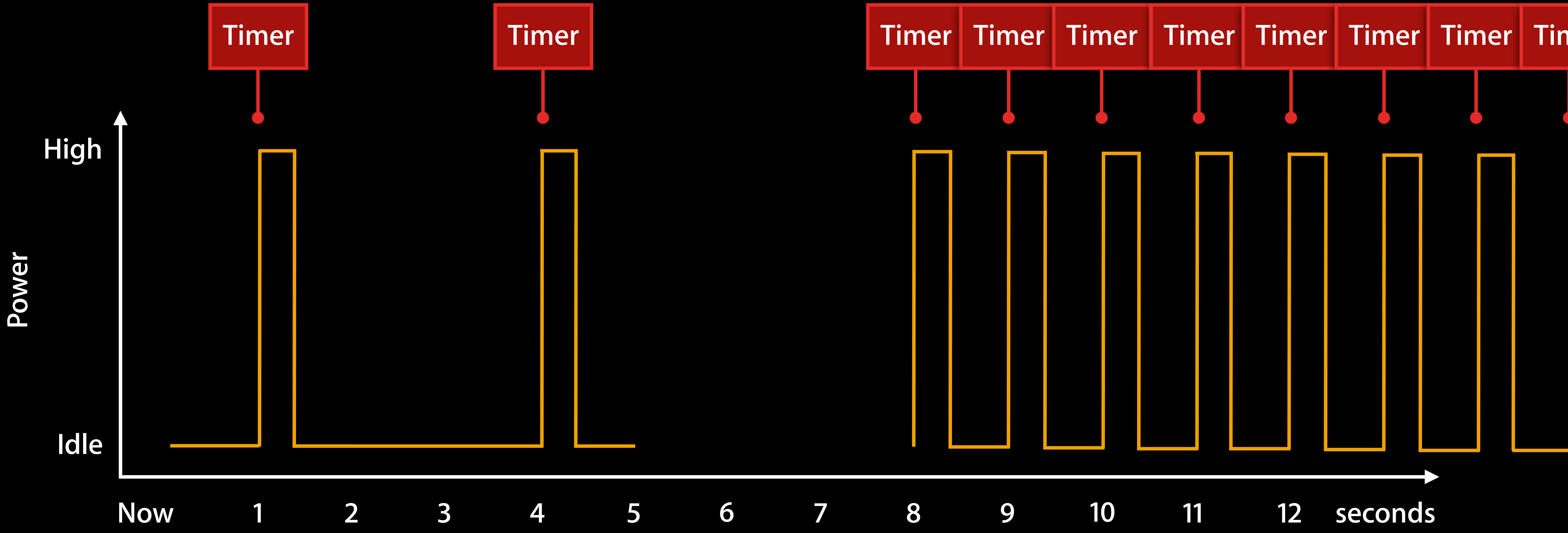
Timer Rate Limiting



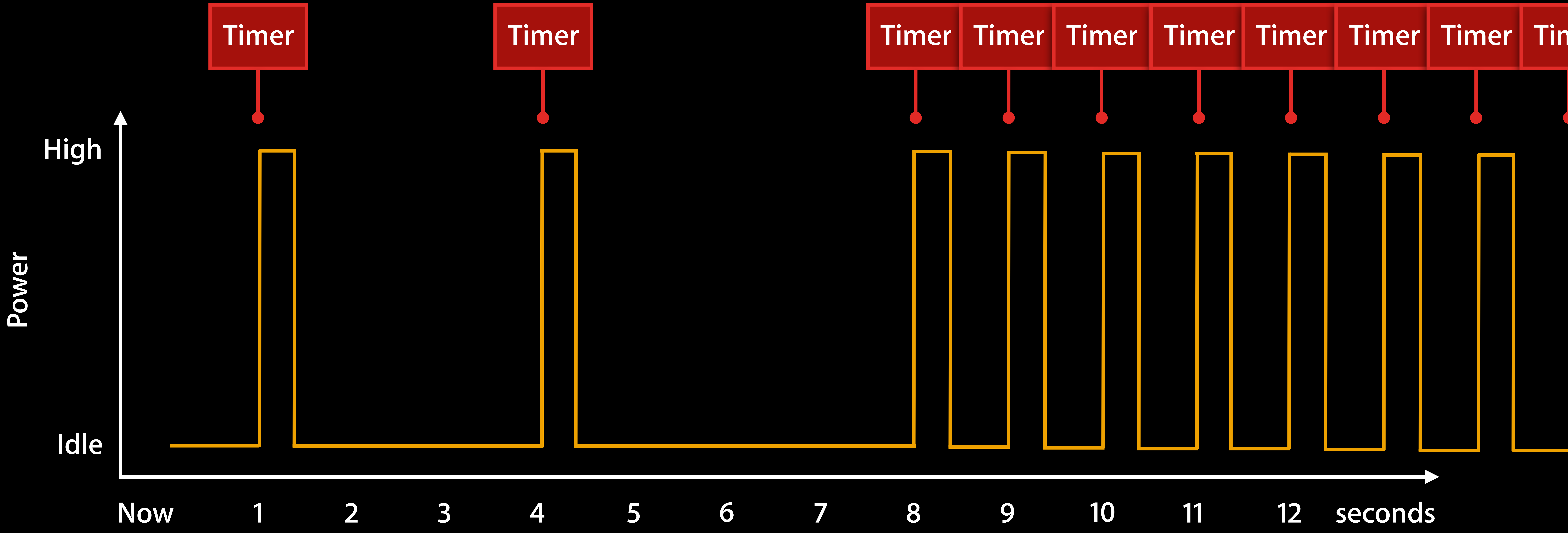
Timer Rate Limiting



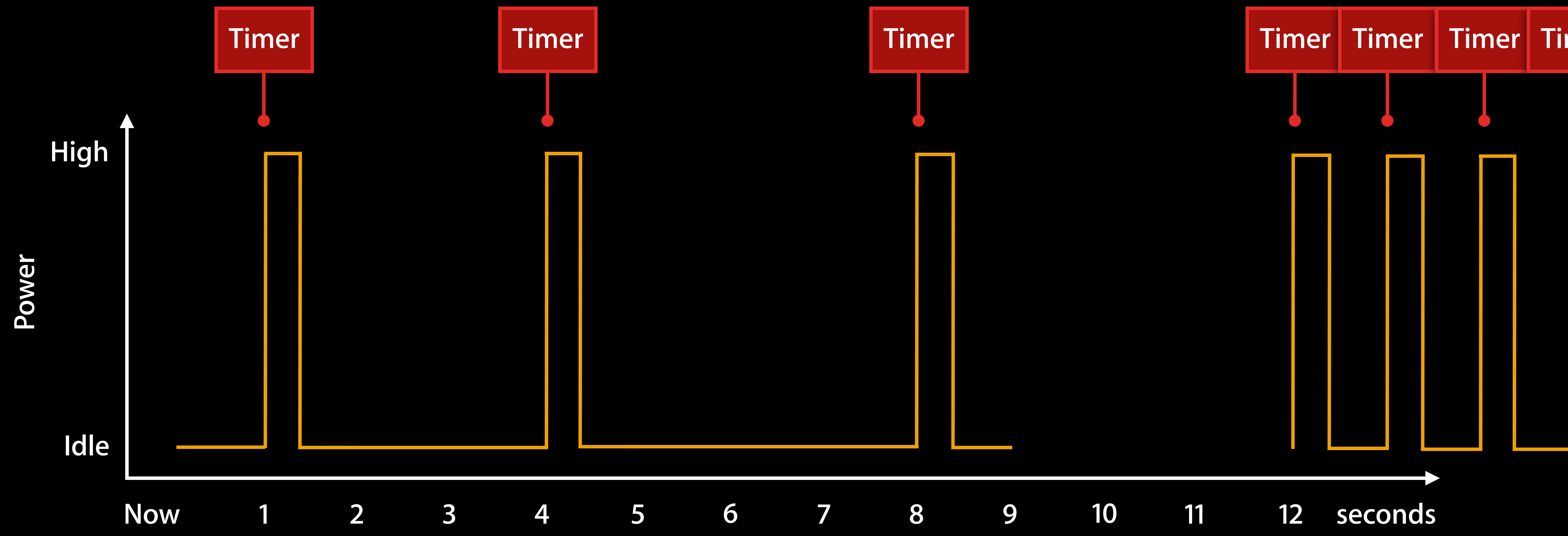
Timer Rate Limiting



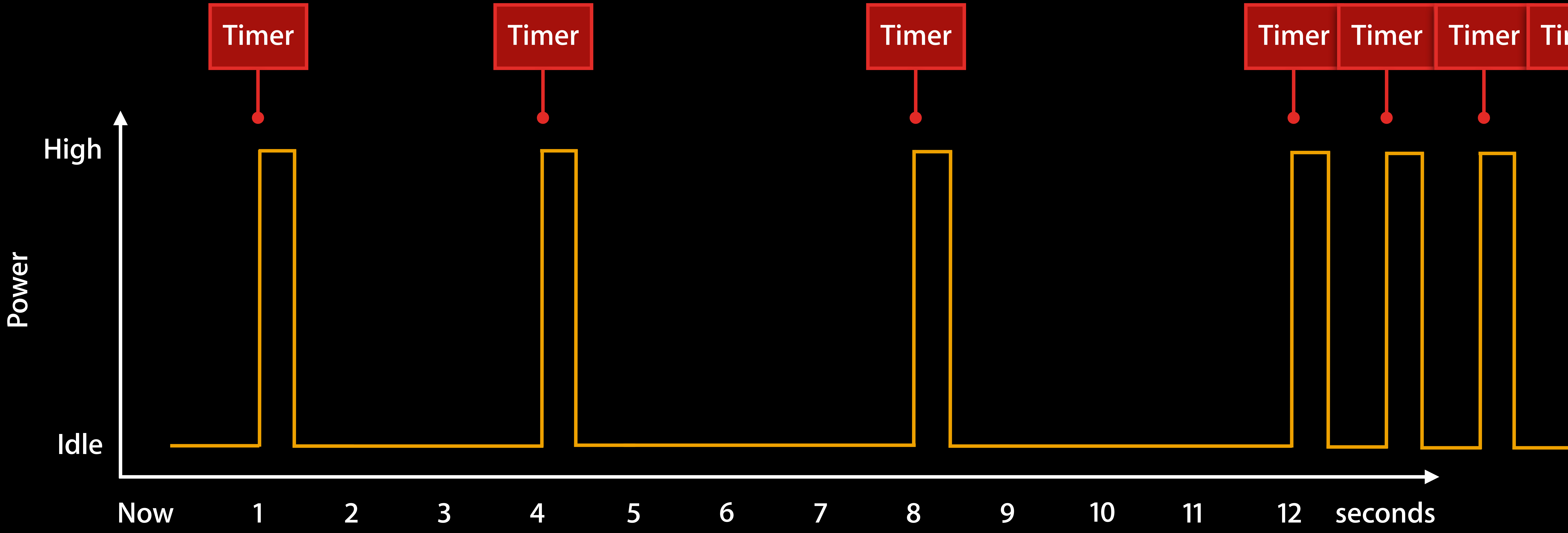
Timer Rate Limiting



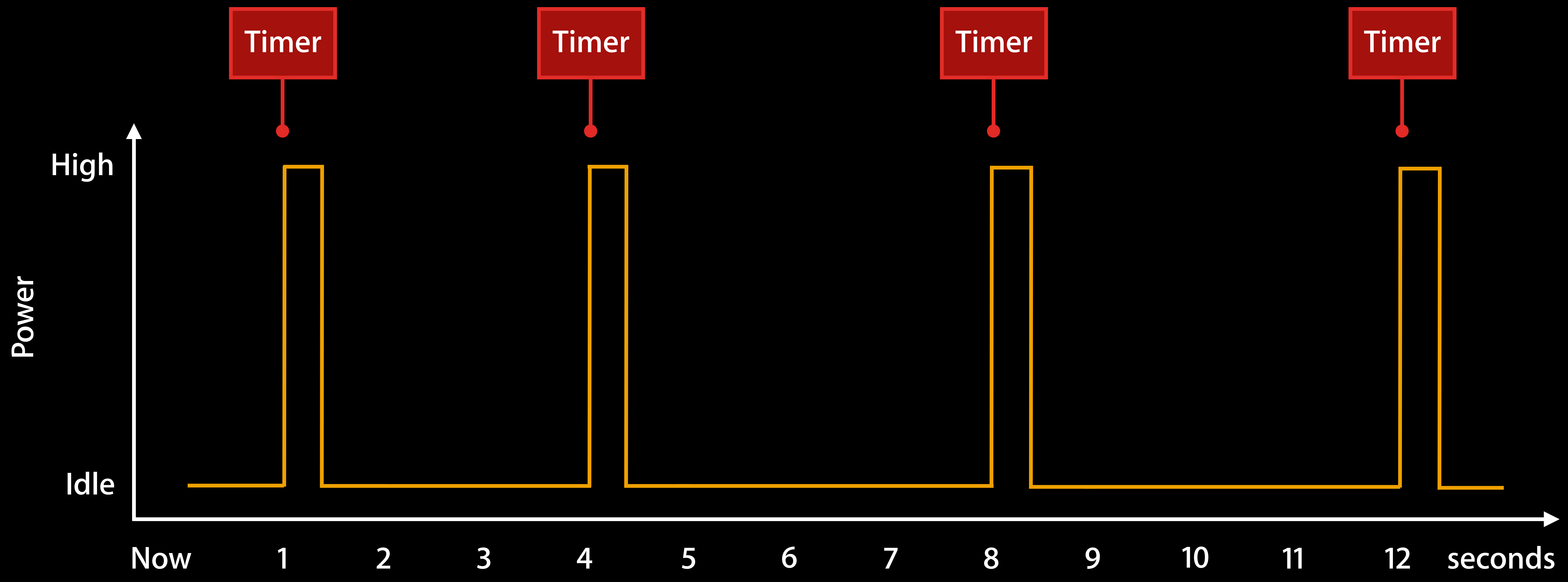
Timer Rate Limiting



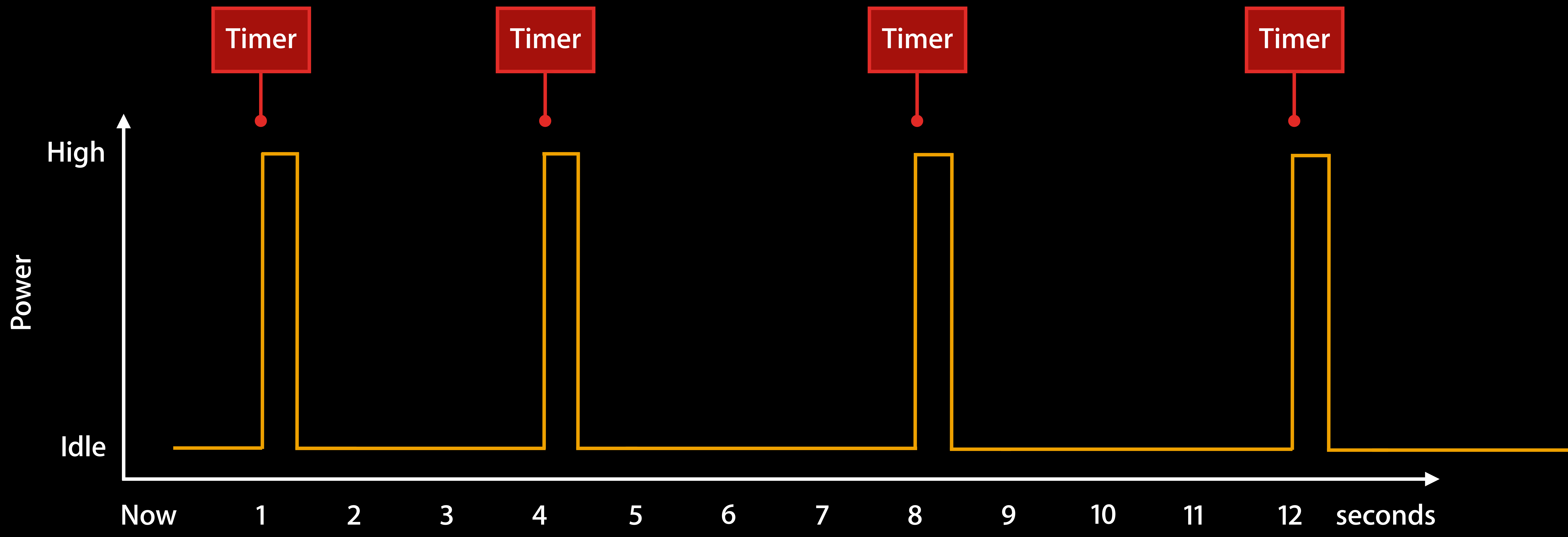
Timer Rate Limiting



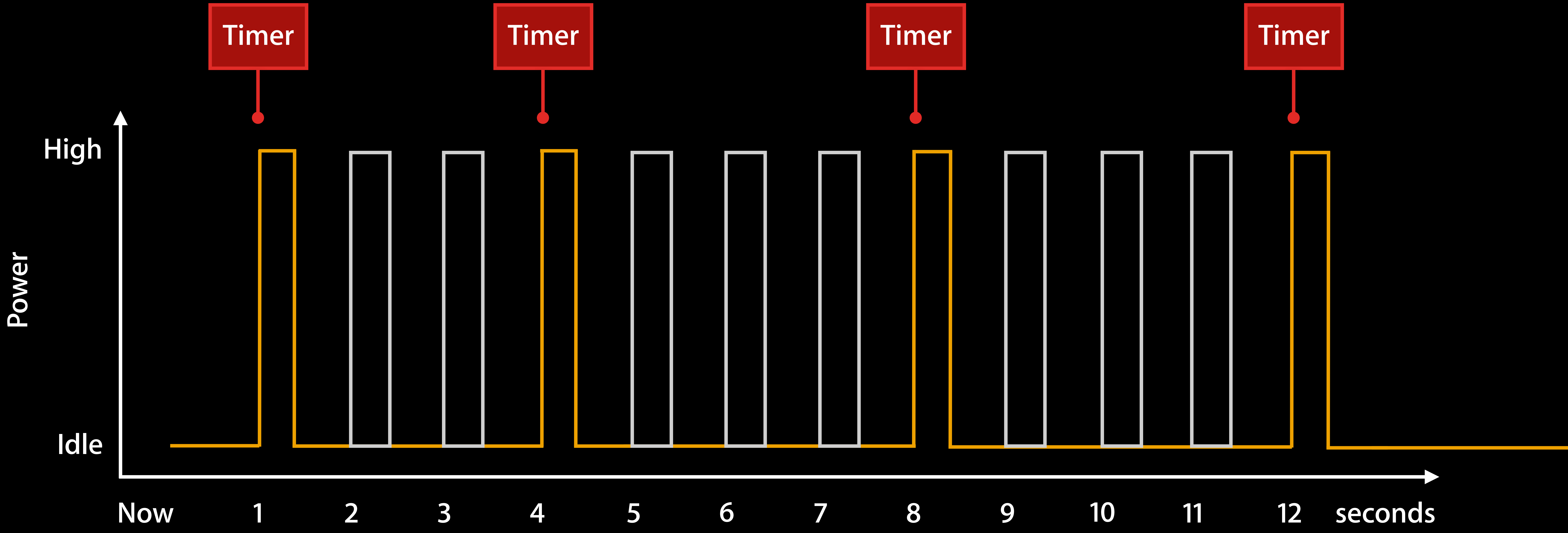
Timer Rate Limiting



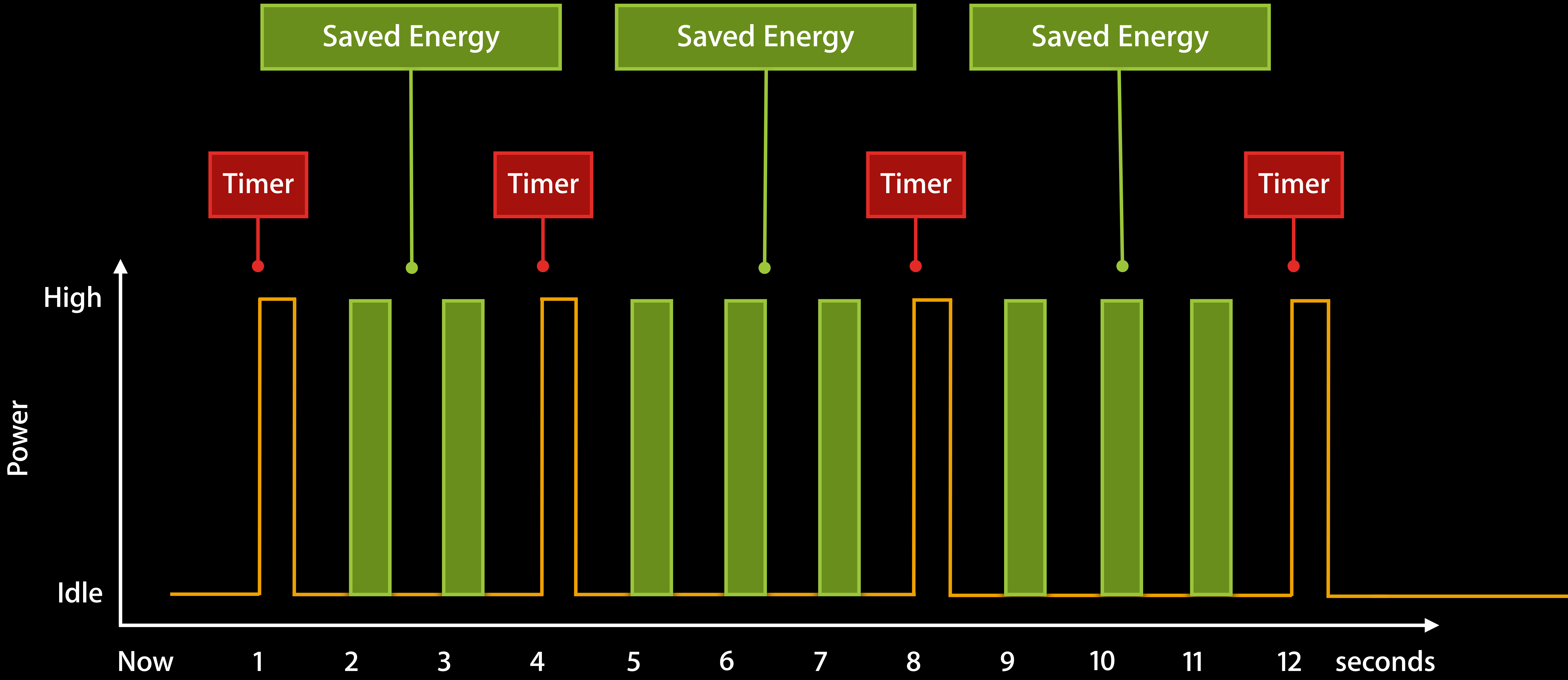
Timer Rate Limiting



Timer Rate Limiting



Timer Rate Limiting



Coalescing and Rate Limiting

- Coalescing delay is on order of 100 ms
  - About the same as delay due to normal system load
  - Undetectable to user

Coalescing and Rate Limiting

- Coalescing delay is on order of 100 ms
  - About the same as delay due to normal system load
  - Undetectable to user
- Rate limiting delay is on order of seconds

Coalescing and Rate Limiting

- Coalescing delay is on order of 100 ms
    - About the same as delay due to normal system load
    - Undetectable to user
- Rate limiting delay is on order of seconds
- Timers do not fire early

Coalescing and Rate Limiting

- Coalescing delay is on order of 100 ms
  - About the same as delay due to normal system load
  - Undetectable to user
- Rate limiting delay is on order of seconds
- Timers do not fire early
- Exact delays depend on heuristics

Coalescing and Rate Limiting

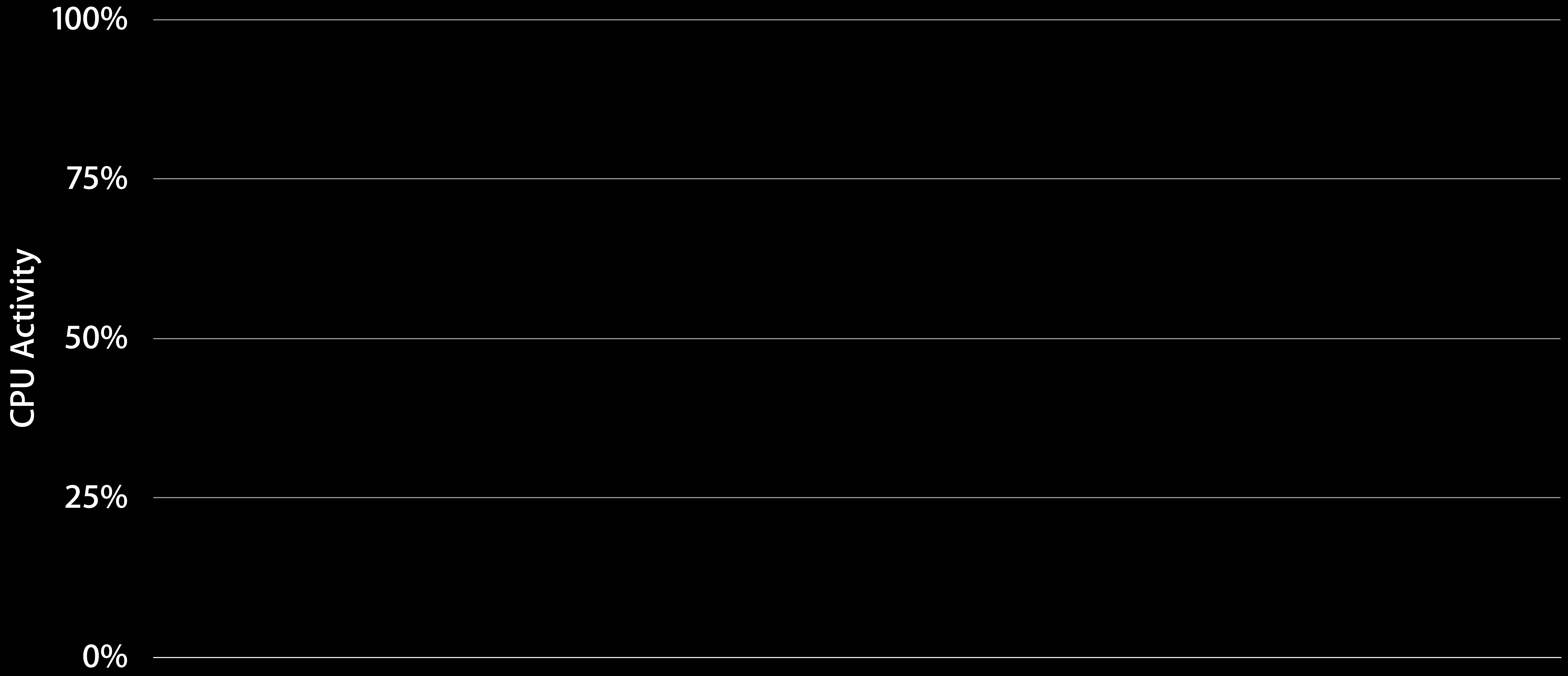
- Coalescing delay is on order of 100 ms
  - About the same as delay due to normal system load
  - Undetectable to user
- Rate limiting delay is on order of seconds
- Timers do not fire early
- Exact delays depend on heuristics
- Configurable

The Result

Eyes demo

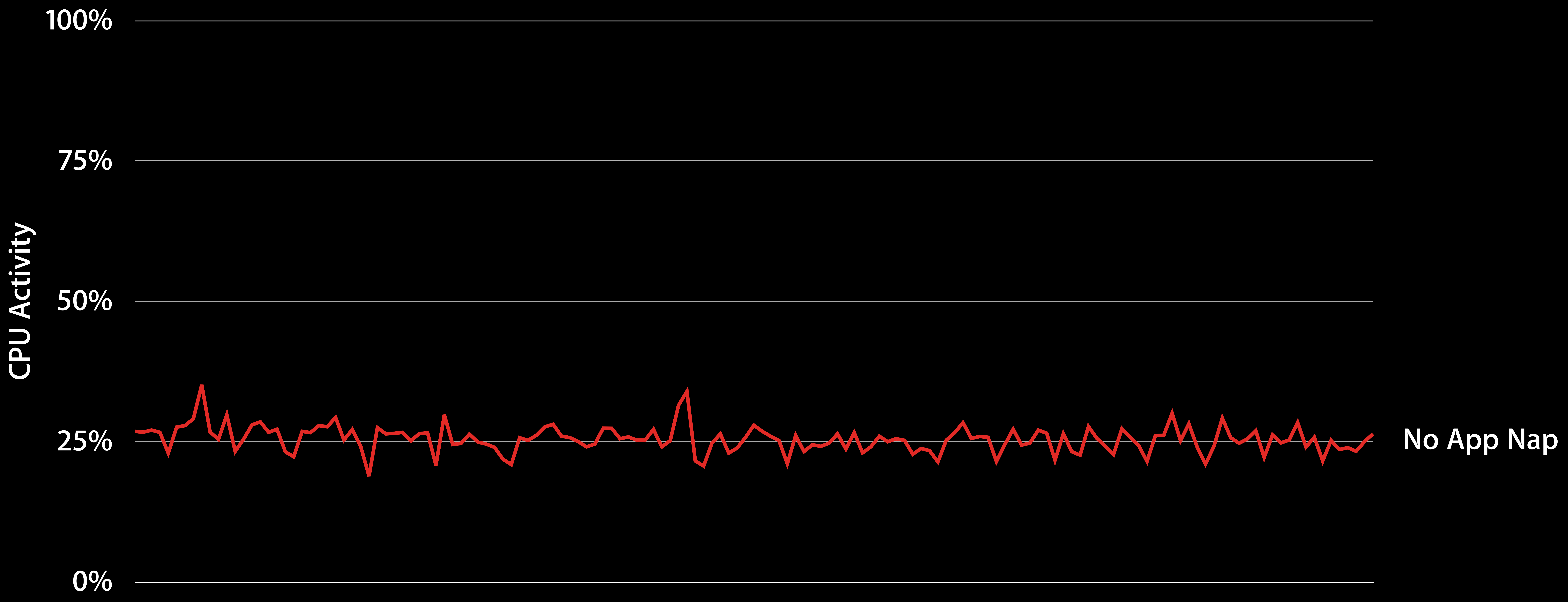
The Result

Eyes demo



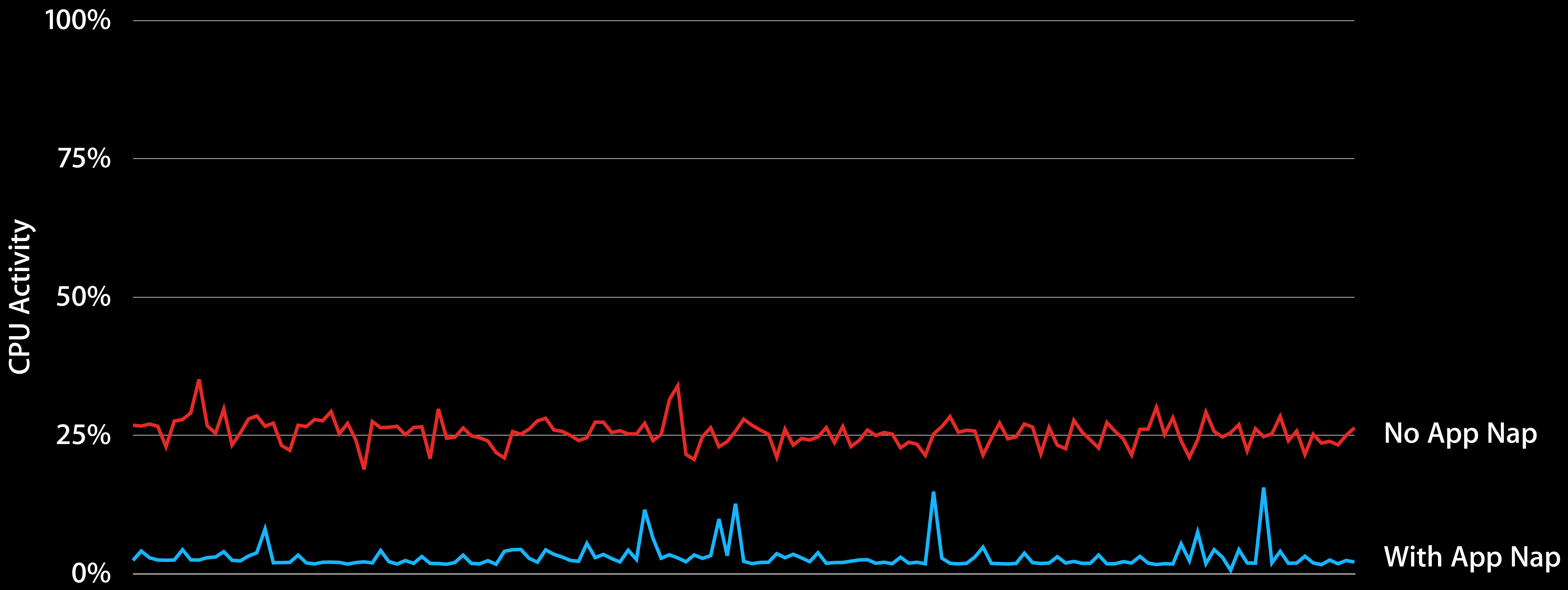
The Result

Eyes demo



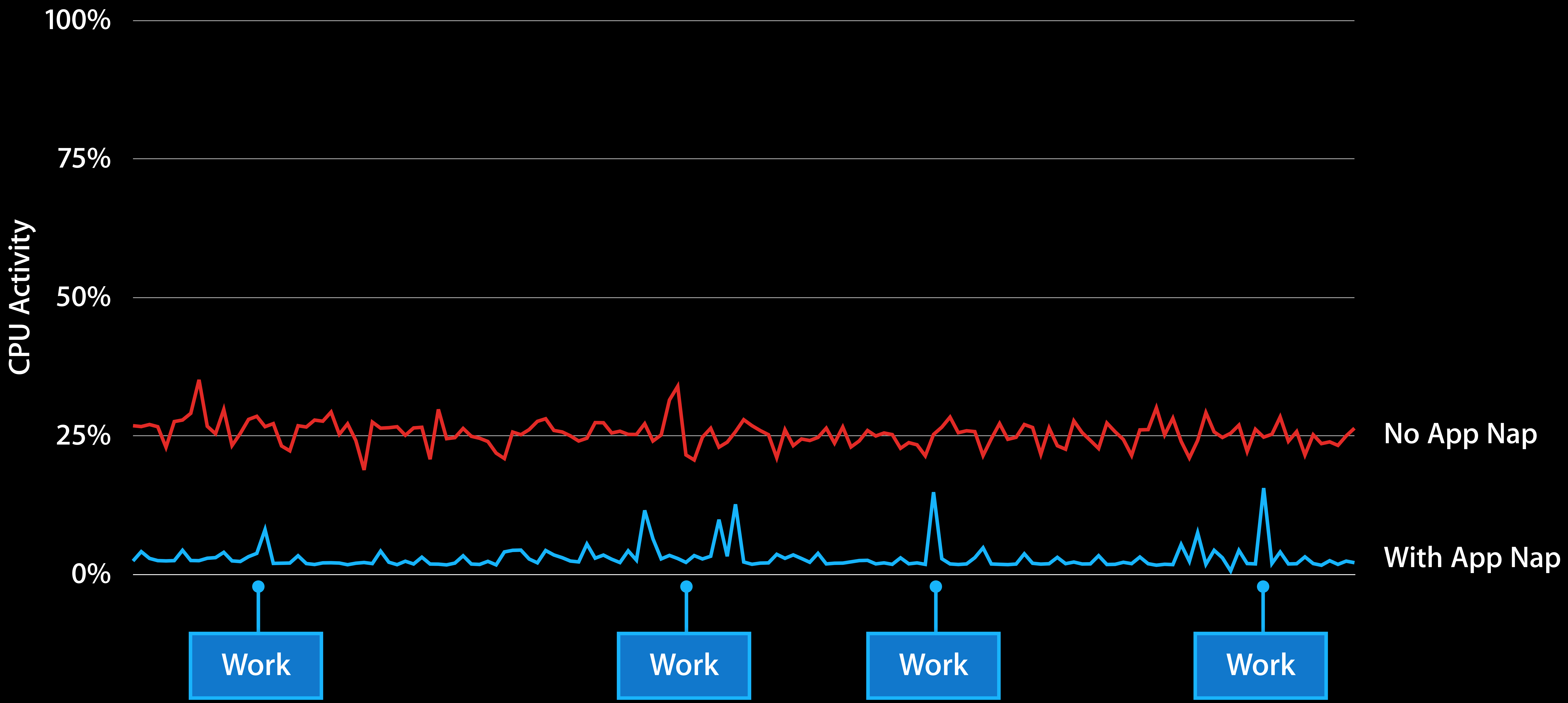
The Result

Eyes demo



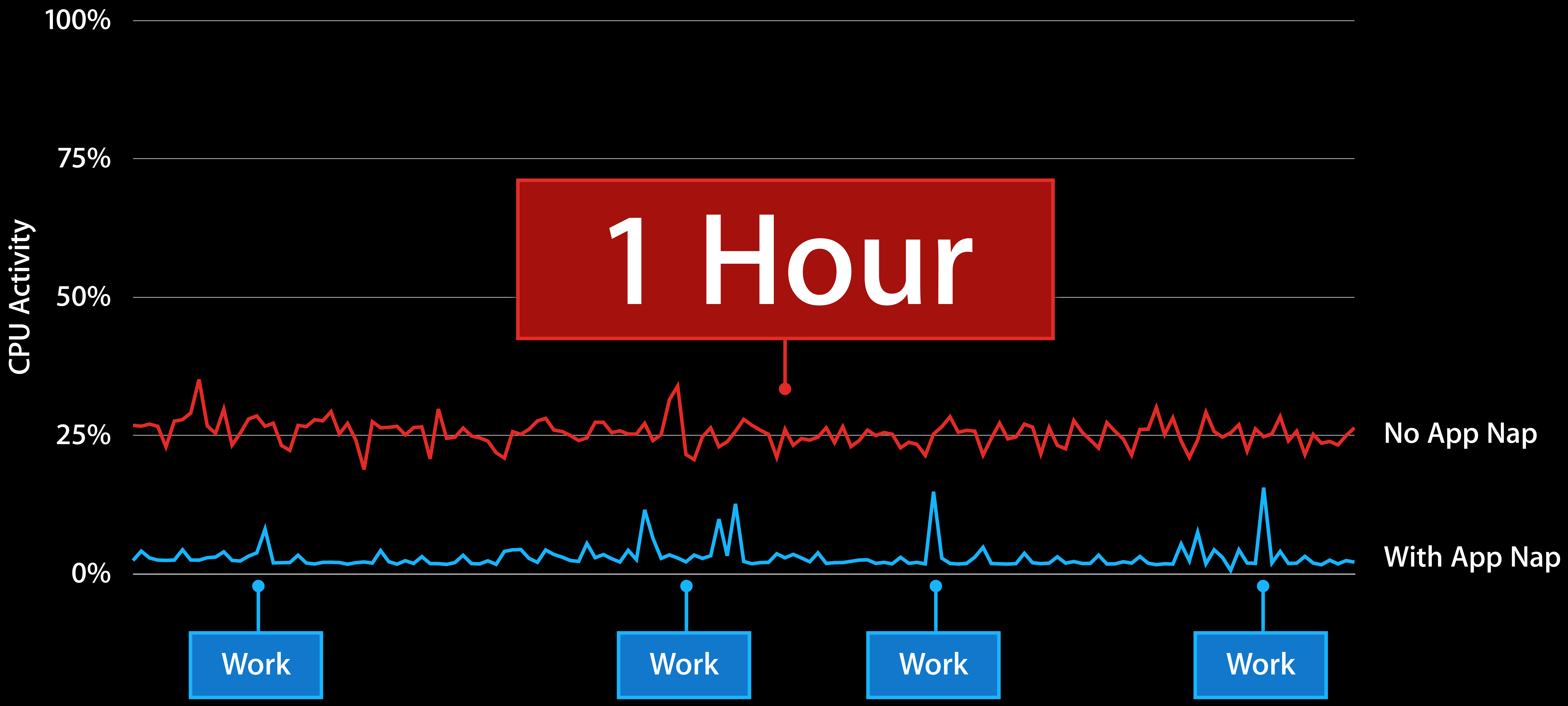
The Result

Eyes demo



The Result

Eyes demo



2012 15" MacBook Pro with Retina Display, 2.6 GHz Intel Core i7, OS X 10.9 Developer Preview

Improving the Result

Switch from timer API to event API

Improving the Result

Switch from timer API to event API

- Instead of polling key presses or mouse locations
    - Use events

Improving the Result

Switch from timer API to event API

- Instead of polling key presses or mouse locations
  - Use events
- Instead of repeatedly checking file content
  - Use FSEvents, dispatch sources, or IPC

Improving the Result

Switch from timer API to event API

- Instead of polling key presses or mouse locations
  - Use events
- Instead of repeatedly checking file content
  - Use FSEvents, dispatch sources, or IPC
- Instead of timer-based synchronization
  - Use semaphores or other locks

Improving the Result

Switch from timer API to event API

- Instead of polling key presses or mouse locations
  - Use events
- Instead of repeatedly checking file content
  - Use FSEvents, dispatch sources, or IPC
- Instead of timer-based synchronization
  - Use semaphores or other locks

Demo

Improving the Eyes application

Responsiveness

- Important work should have higher priority

Responsiveness

- Important work should have higher priority
- Apps in App Nap have lower priority
    - I/O
    - CPU

Responsiveness

- Important work should have higher priority
- Apps in App Nap have lower priority
  - I/O
  - CPU
- To improve responsiveness, improve performance

Responsiveness

- Important work should have higher priority
- Apps in App Nap have lower priority
    - I/O
    - CPU
- To improve responsiveness, improve performance

App Nap API

App Nap API

- Find out when your app is visible
- Add tolerance to timers
- Tell system about user activities

Visibility

- Find out when a window or application is occluded
    - On another space
    - Another app is in front
    - Screen saver is on

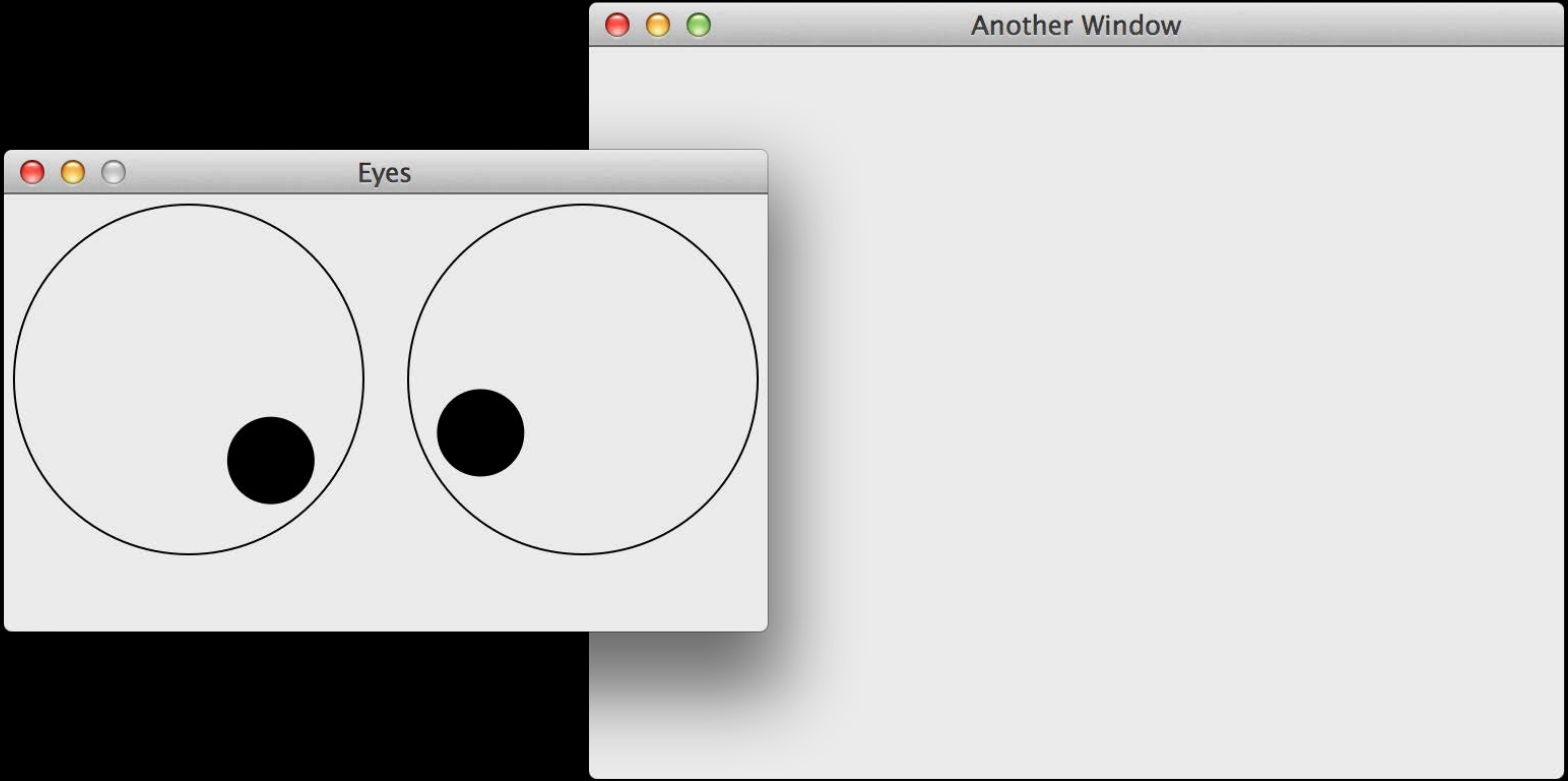
Visibility

- Find out when a window or application is occluded
    - On another space
    - Another app is in front
    - Screen saver is on
- Halt expensive work when occluded

Visibility

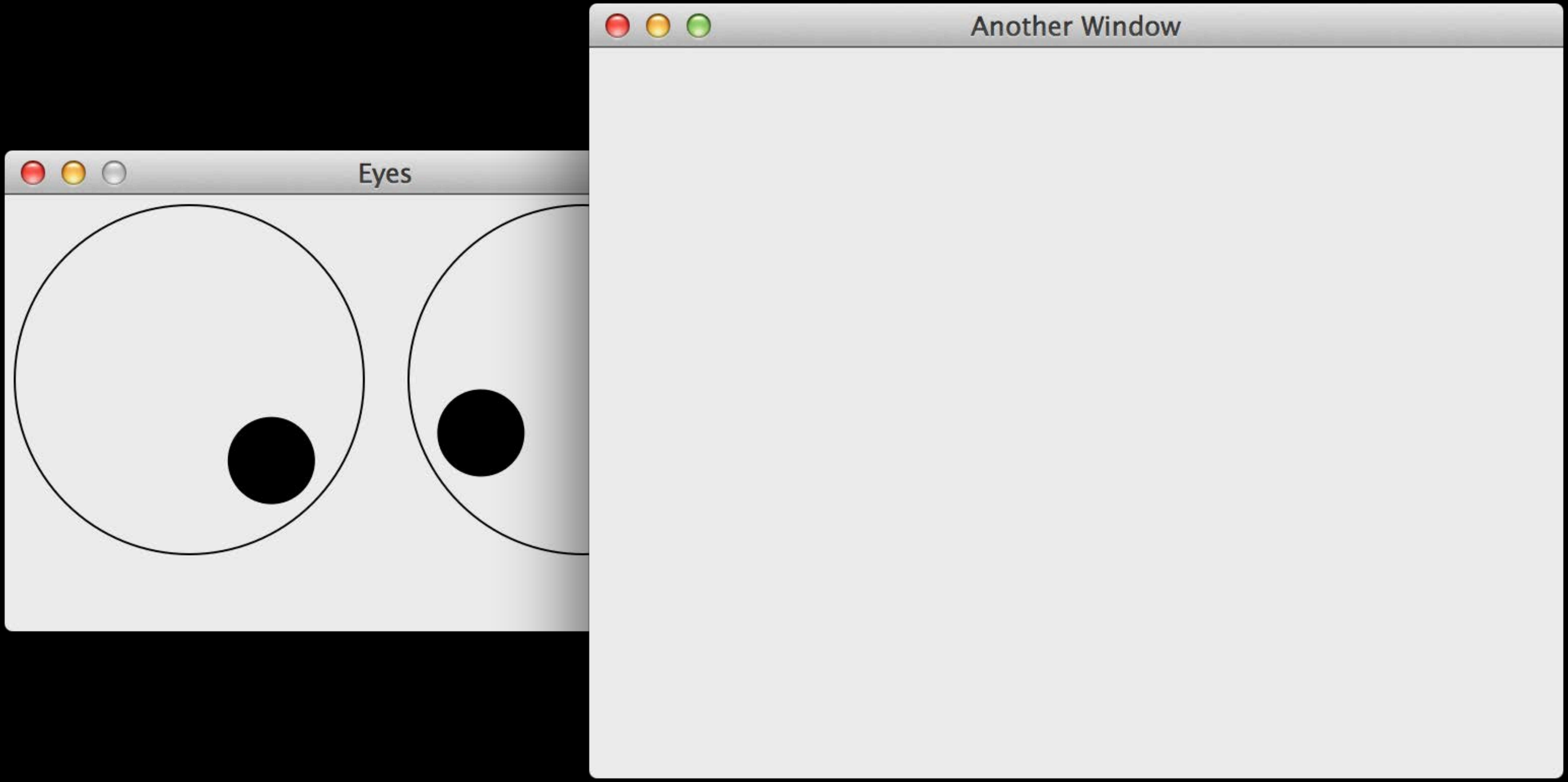
- Find out when a window or application is occluded
    - On another space
    - Another app is in front
    - Screen saver is on
- Halt expensive work when occluded
- Refresh content when becoming visible

Window Occlusion



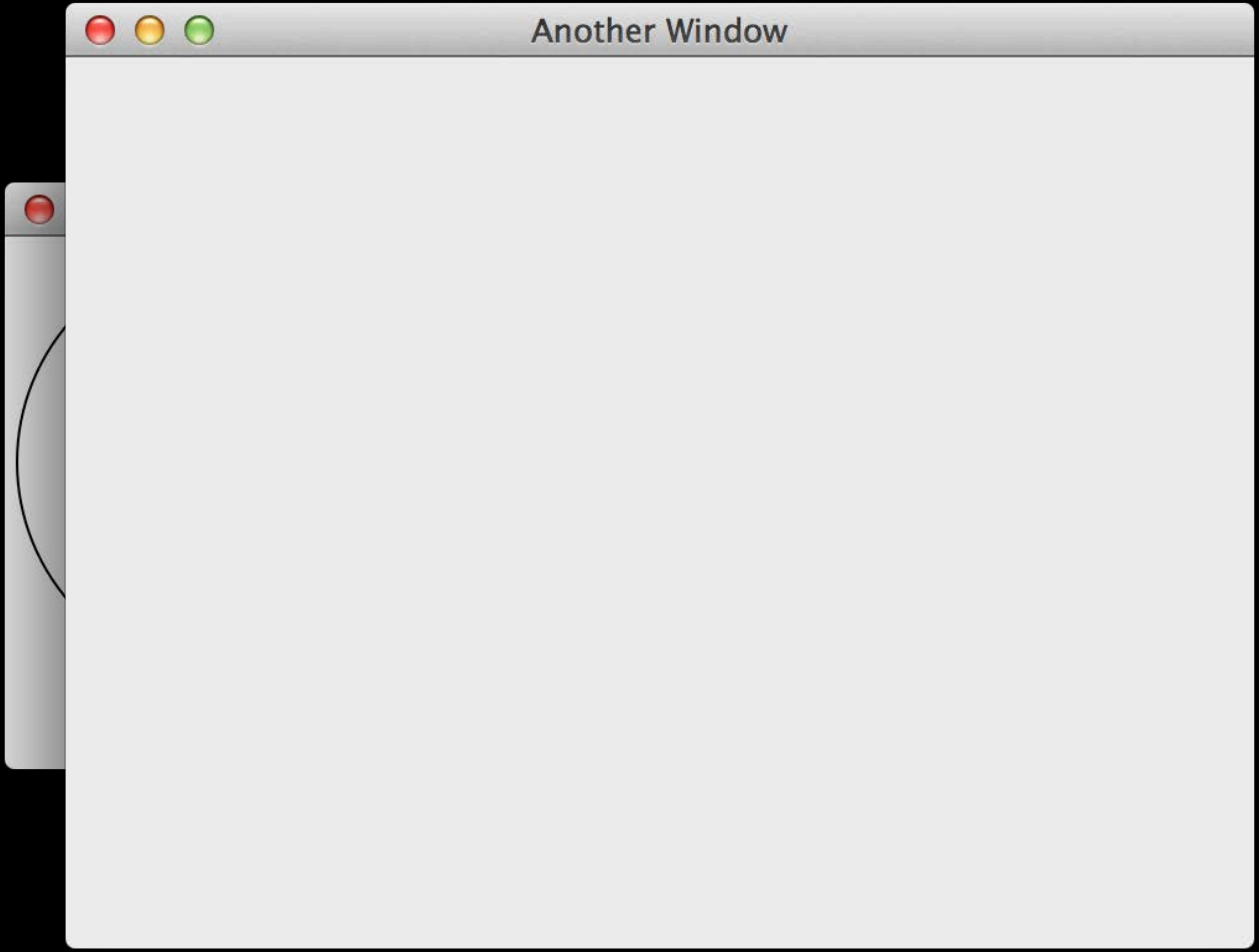
Visible

Window Occlusion



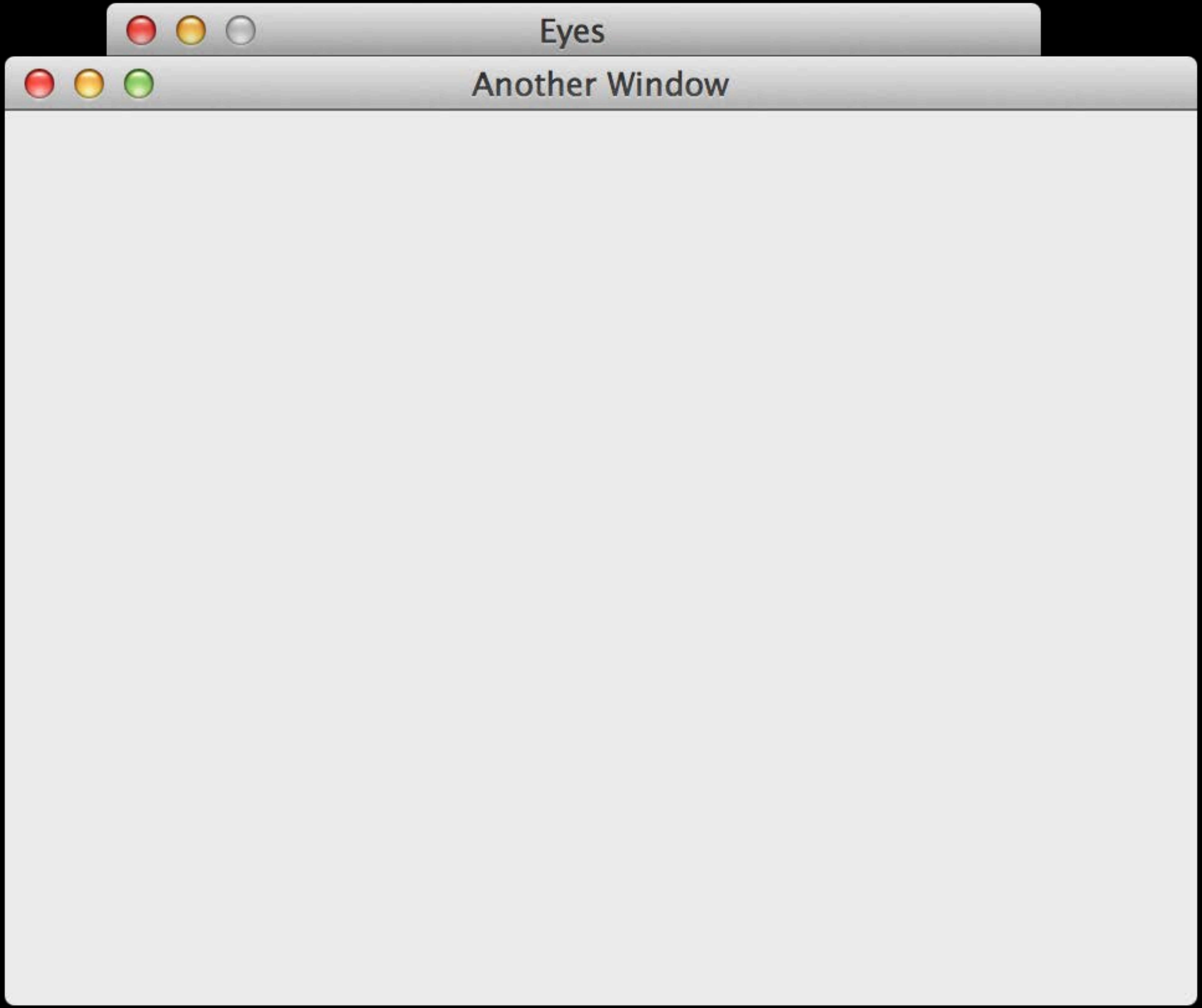
Visible

Window Occlusion



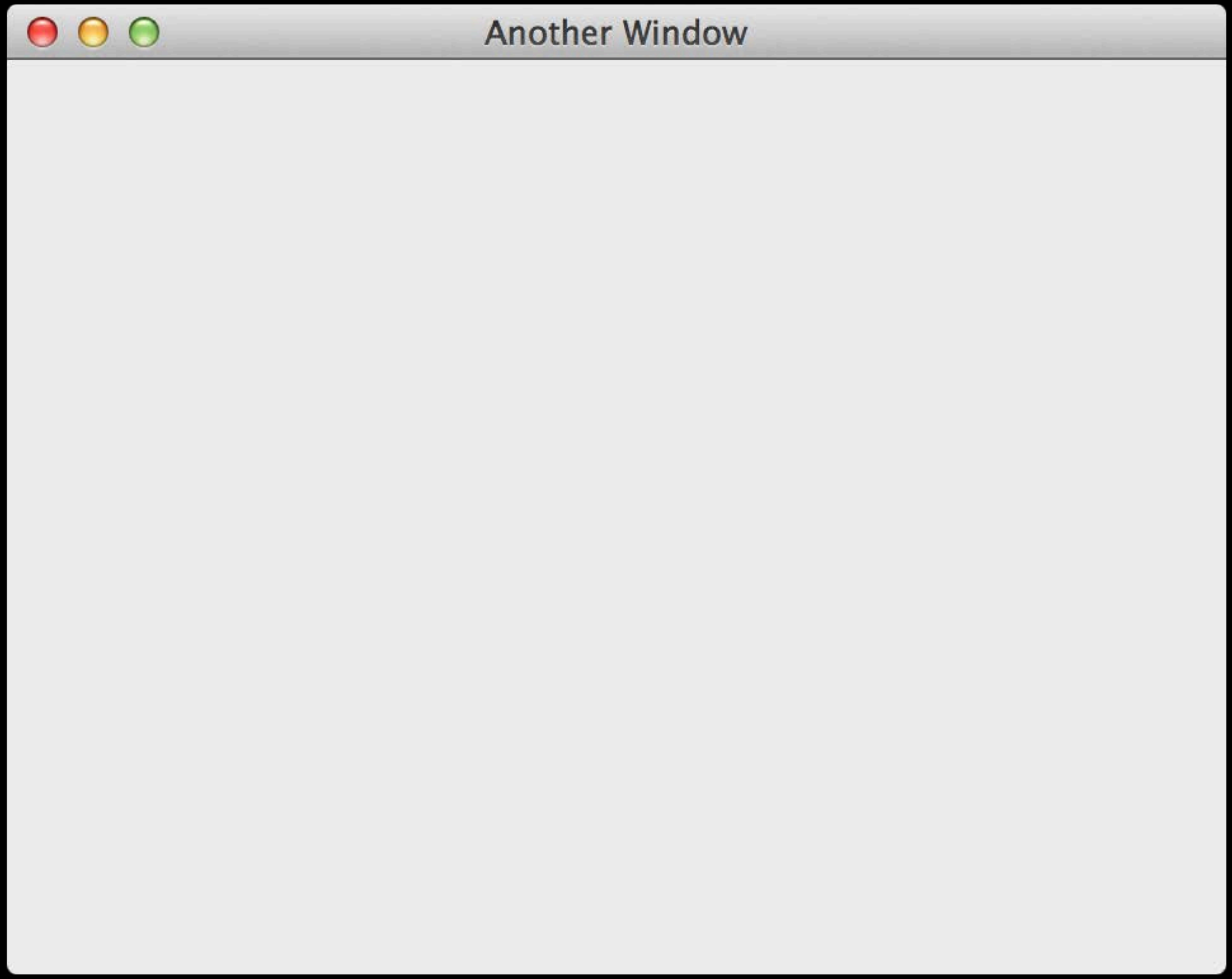
Visible

Window Occlusion



Visible

Window Occlusion

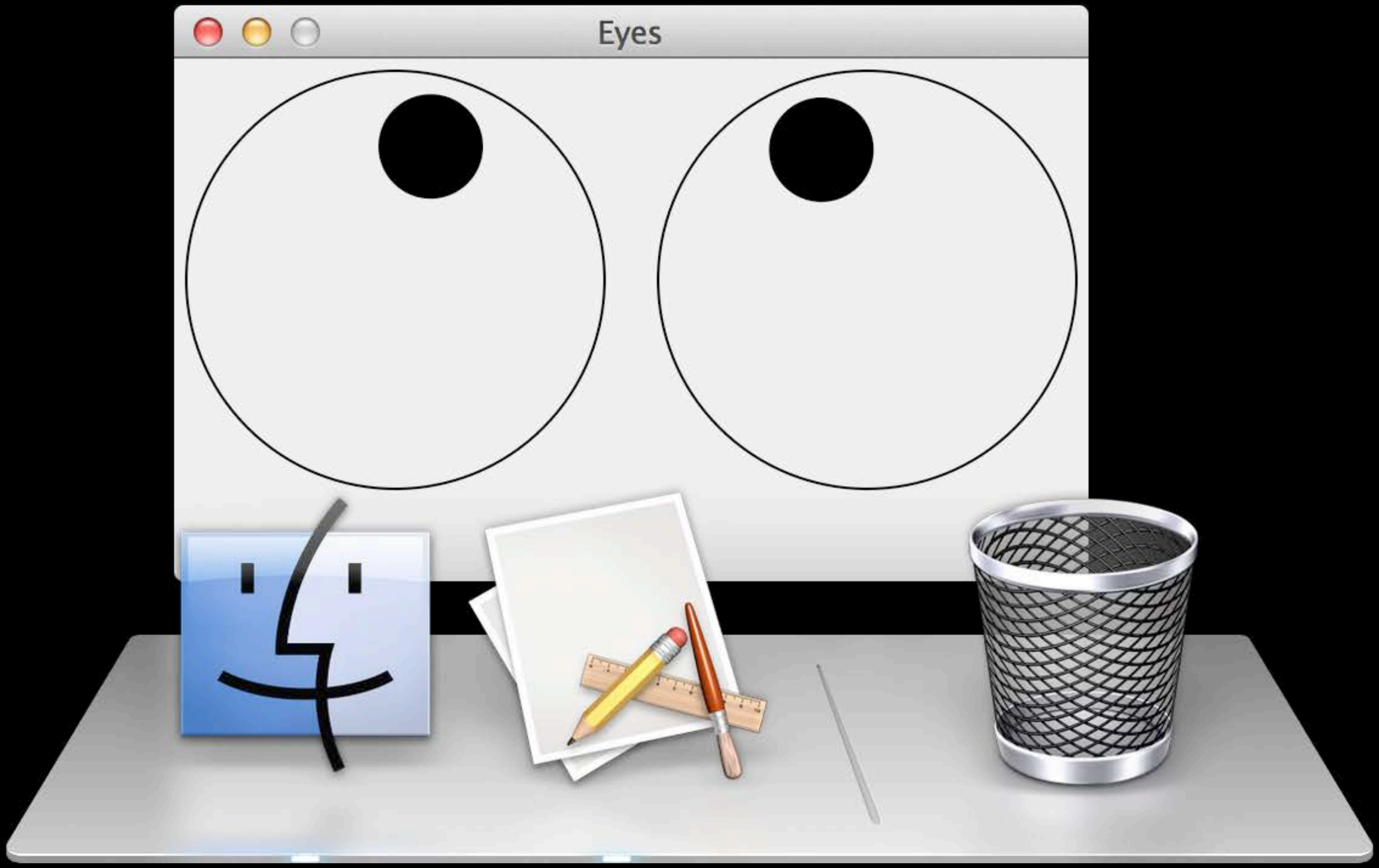


Occluded

Minimized Windows

Visible

Minimized Windows



Occluded

Application Occlusion

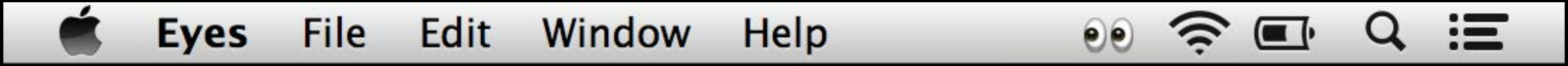
- Union of all application windows

Application Occlusion

- Union of all application windows
- Menu bar does not count
    - Except for a status item

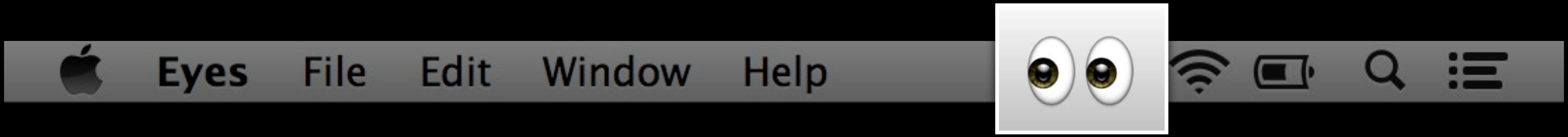
Application Occlusion

- Union of all application windows
- Menu bar does not count
    - Except for a status item



Application Occlusion

- Union of all application windows
- Menu bar does not count
  - Except for a status item



Application Occlusion

```
@protocol NSApplicationDelegate
- (void)applicationDidChangeOcclusionState:(NSNotification *)notification;
@end

@interface NSApplication
- (NSApplicationOcclusionState)occlusionState;
@end

typedef NS_OPTIONS(NSUInteger, NSApplicationOcclusionState) {
    NSApplicationOcclusionStateVisible = 1UL << 1,
}
```

Window Occlusion

```
@protocol NSWindowDelegate
- (void)windowDidChangeOcclusionState:(NSNotification *)notification;
@end

@interface NSWindow
- (NSWindowOcclusionState)occlusionState;
@end

typedef NS_OPTIONS(NSUInteger, NSWindowOcclusionState) {
    NSWindowOcclusionStateVisible = 1UL << 1,
}
```

Occlusion Example

```
@implementation EYEAppDelegate

- (void)applicationDidChangeOcclusionState:(NSNotification *)n
{
    if ([NSApp occlusionState] & NSApplicationOcclusionStateVisible) {
        // Visible
    } else {
        // Occluded
    }
}

@end
```

Timer Tolerance

- Most timers do not need to be hyper-accurate
    - Default tolerance is applied to all timers

Timer Tolerance

- Most timers do not need to be hyper-accurate
  - Default tolerance is applied to all timers
- New API allows for increasing default tolerance

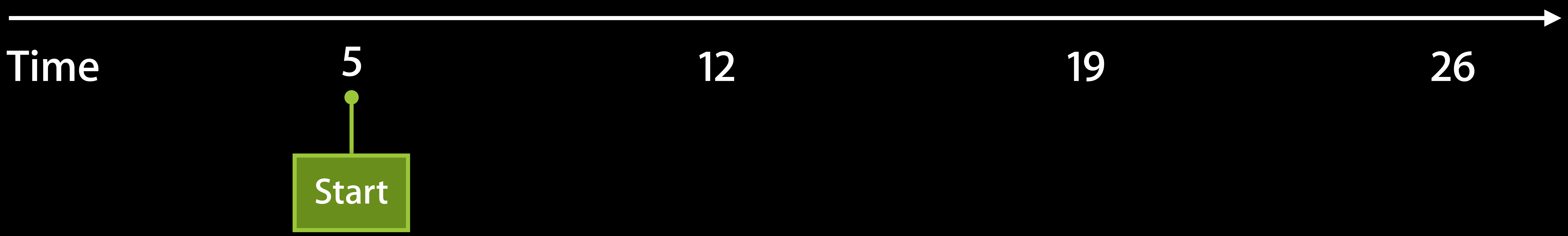
Timer Tolerance

- Most timers do not need to be hyper-accurate
    - Default tolerance is applied to all timers
- New API allows for increasing default tolerance
- System fires timer at best time in tolerance window

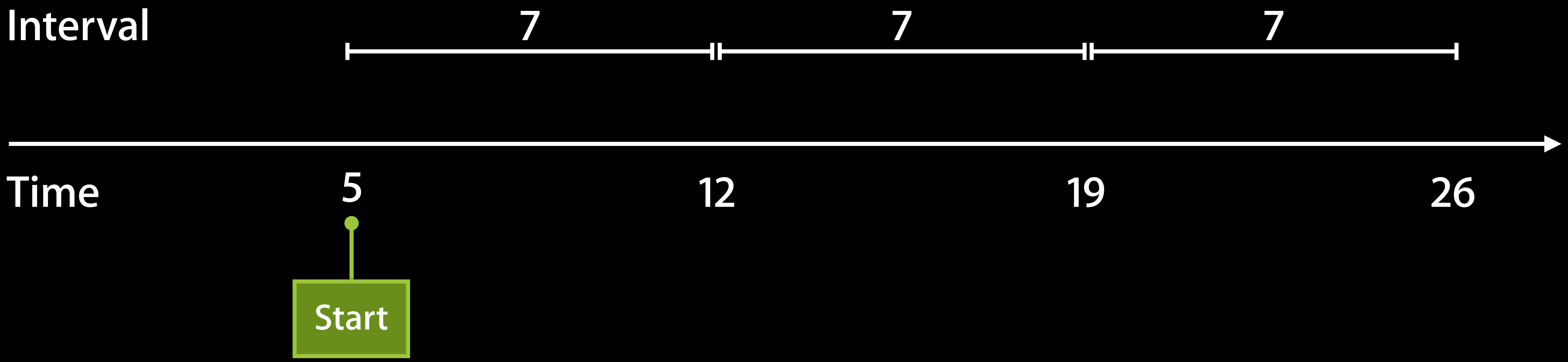
Timer Tolerance

Time
5
12
19
26

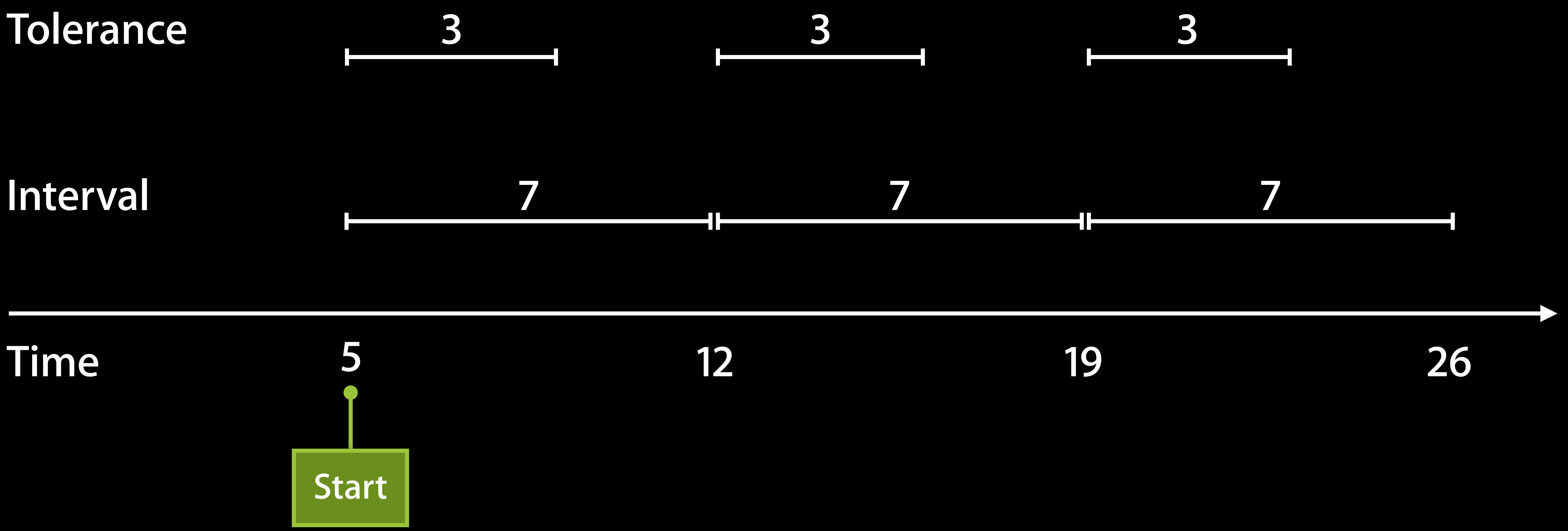
Timer Tolerance



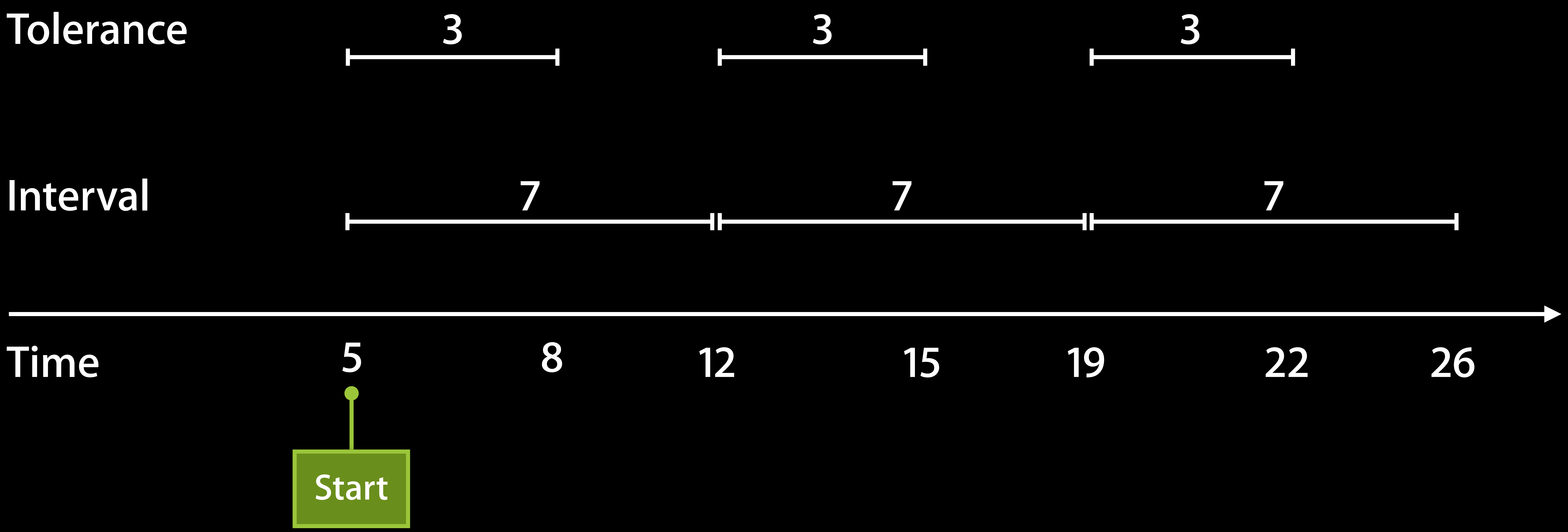
Timer Tolerance



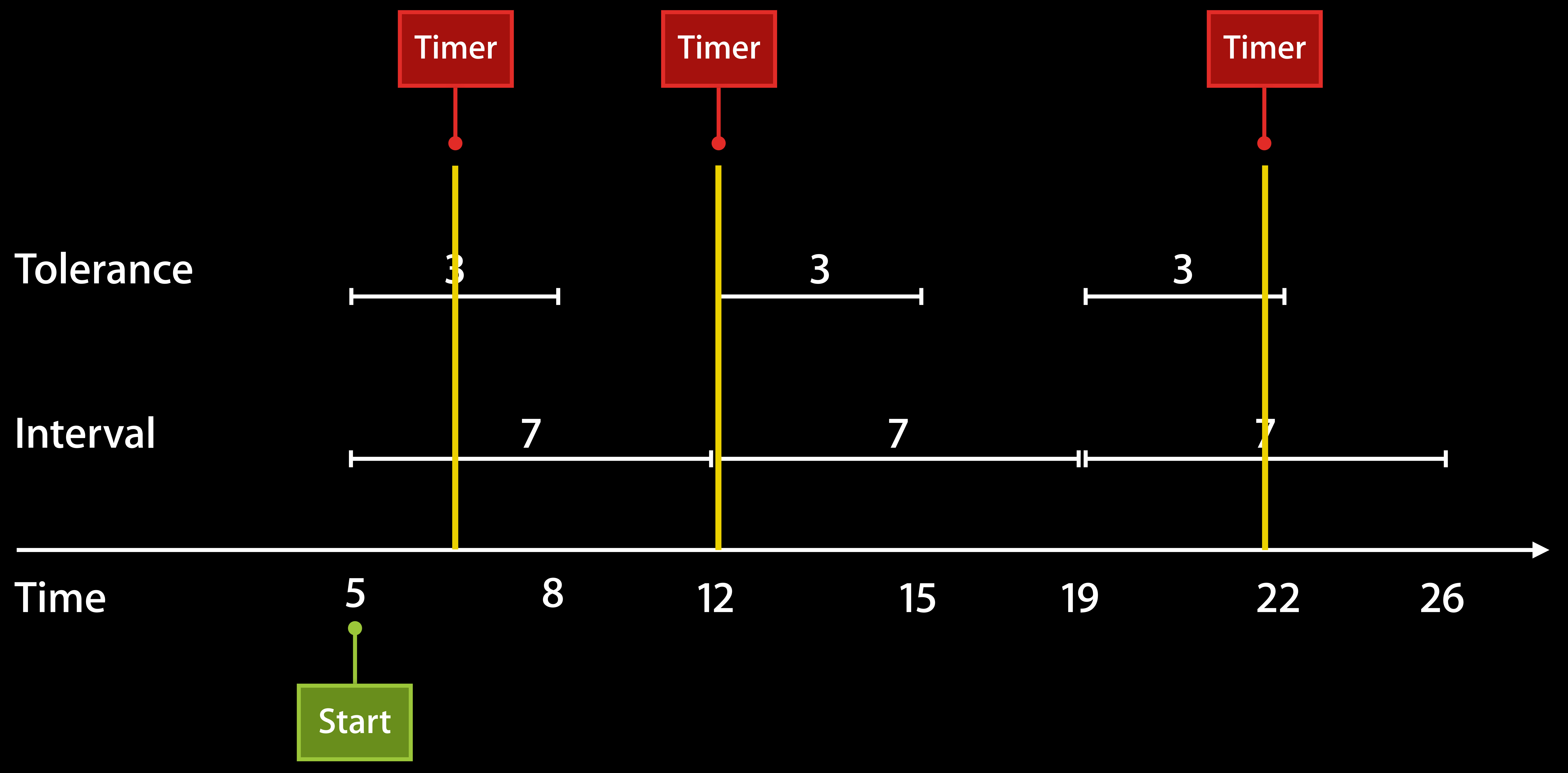
Timer Tolerance



Timer Tolerance



Timer Tolerance



NSTimer Tolerance

```
@interface NSTimer
- (void)setTolerance:(NSTimeInterval)tolerance;
- (NSTimeInterval)tolerance;
@end
```

NSTimer Tolerance

```
// Create repeating timer
NSTimer *timer = [NSTimer timerWithTimeInterval:7.0
                                          target:self
                                        selector:@selector(timerFired:)
                                        userInfo:nil
                                         repeats:YES];

// Set fire date
[timer setFireDate:[NSDate dateWithTimeIntervalSinceNow:5.0]];

// Set tolerance
[timer setTolerance:3.0];

[[NSRunLoop currentRunLoop] addTimer:timer forMode:NSRunLoopCommonModes];
```

Dispatch Timer Tolerance

```
dispatch_source_t timer;
timer = dispatch_source_create(DISPATCH_SOURCE_TYPE_TIMER,
                               0,
                               0,
                               queue);

dispatch_source_set_event_handler(timer, ^{ /* Work goes here */ });

dispatch_source_set_timer(timer,
                    dispatch_time(DISPATCH_TIME_NOW, 5 * NSEC_PER_SEC),
                    7 * NSEC_PER_SEC,
                    3 * NSEC_PER_SEC);

dispatch_resume(timer);
```

Dispatch Strict Timers

```
dispatch_source_t timer;
timer = dispatch_source_create(DISPATCH_SOURCE_TYPE_TIMER,
                               0,
                               DISPATCH_TIMER_STRICT,
                               queue);

dispatch_source_set_event_handler(timer, ^{ /* Work goes here */ });

dispatch_source_set_timer(timer,
                    dispatch_time(DISPATCH_TIME_NOW, 5 * NSEC_PER_SEC),
                    7 * NSEC_PER_SEC,
                    700 * NSEC_PER_MSEC);

dispatch_resume(timer);
```

Timer Tolerance

- Suggested tolerance is at least 10% of interval
  - Exact value will be application specific

Timer Tolerance

- Suggested tolerance is at least 10% of interval
    - Exact value will be application specific
- Tolerance used regardless of App Nap

Timer Tolerance

- Suggested tolerance is at least 10% of interval
    - Exact value will be application specific
- Tolerance used regardless of App Nap
- Strict timers are rare
    - Disables timer rate limiting
    - You should still specify a tolerance

Timer Tolerance

- Suggested tolerance is at least 10% of interval
  - Exact value will be application specific
- Tolerance used regardless of App Nap
- Strict timers are rare
  - Disables timer rate limiting
  - You should still specify a tolerance
- Critical mass effect

User Activities

- Improves accuracy of App Nap heuristics

User Activities

- Improves accuracy of App Nap heuristics
- Use for long-running or asynchronous work

User Activities

- Improves accuracy of App Nap heuristics
- Use for long-running or asynchronous work
- Cocoa API to prevent idle system sleep

User Activities

- Improves accuracy of App Nap heuristics
- Use for long-running or asynchronous work
- Cocoa API to prevent idle system sleep
- Includes automatic and sudden termination

User Activities

```
@interface NSProcessInfo

- (void)performActivityWithOptions:(NSActivityOptions)options
                            reason:(NSString *)reason
                             block:(void (^)())block;




@end
```

User Activities

```
@interface NSProcessInfo

- (void)performActivityWithOptions:(NSActivityOptions)options
                            reason:(NSString *)reason
                             block:(void (^)())block;

- (id)beginActivityWithOptions:(NSActivityOptions)options
                        reason:(NSString *)reason;

- (void)endActivity:(id)activity;

@end
```

User Activities

NSActivityOptions

- Exporting, recording, processing

```
NSActivityUserInitiated
NSActivityUserInitiatedAllowingIdleSystemSleep
```

User Activities

NSActivityOptions

- Exporting, recording, processing

```
NSActivityUserInitiated
NSActivityUserInitiatedAllowingIdleSystemSleep
```

- Maintenance

```
NSActivityBackground
```

User Activities

NSActivityOptions

- Exporting, recording, processing

```
NSActivityUserInitiated
NSActivityUserInitiatedAllowingIdleSystemSleep
```

- Maintenance

```
NSActivityBackground
```

- Latency sensitive

```
NSActivityUserInitiated | NSActivityLatencyCritical
```

User Activities

NSActivityOptions

- Idle system sleep

```
NSActivityIdleDisplaySleepDisabled
NSActivityIdleSystemSleepDisabled
```

User Activities

NSActivityOptions

- Idle system sleep

```
NSActivityIdleDisplaySleepDisabled
NSActivityIdleSystemSleepDisabled
```

- Sudden termination

```
NSActivitySuddenTerminationDisabled
```

User Activities

NSActivityOptions

- Idle system sleep

```
NSActivityIdleDisplaySleepDisabled
NSActivityIdleSystemSleepDisabled
```

- Sudden termination

```
NSActivitySuddenTerminationDisabled
```

- Automatic termination

```
NSActivityAutomaticTerminationDisabled
```

User Activities

```
NSOperationQueue *queue = ...;

id token = [[NSProcessInfo processInfo]
              beginActivityWithOptions:NSActivityUserInitiated
                                reason:@"Batch processing files"];

[queue addOperationWithBlock:^{
    // Do work here

    [[NSProcessInfo processInfo] endActivity:token];
}];
```

Choosing the Right Activity

- Applications can have multiple concurrent activities
    - NSActivityBackground for maintenance work
    - NSActivityUserInitiated when user takes action

Choosing the Right Activity

- Applications can have multiple concurrent activities
    - NSActivityBackground for maintenance work
    - NSActivityUserInitiated when user takes action
- Avoid rapidly starting and ending activities

Choosing the Right Activity

- Applications can have multiple concurrent activities
    - NSActivityBackground for maintenance work
    - NSActivityUserInitiated when user takes action
- Avoid rapidly starting and ending activities
- Idle system sleep assertions should be used with care
    - Don’t prevent idle sleep forever
    - Verify power assertions are dropped

Choosing the Right Activity

Verify power assertions

Choosing the Right Activity

Verify power assertions

```
$ pmset -g assertions
```

Choosing the Right Activity

Verify power assertions

```
$ pmset -g assertions
Assertion status system-wide:
   BackgroundTask                0
   PreventUserIdleDisplaySleep   0
   PreventSystemSleep            0
   PreventDiskIdle               0
   PreventUserIdleSystemSleep    1
   ExternalMedia                 0
   UserIsActive                  0
   ApplePushServiceTask          0
Listed by owning process:
   pid 1963(Eyes): [0x0000000100000196] 00:03:36 PreventUserIdleSystemSleep
named: "Keeping the computer awake"
```

Demo

Adopting App Nap API

Summary

Summary

- Software has a huge impact on energy efficiency

Summary

- Software has a huge impact on energy efficiency
- To extend battery life
    - Stay idle as long as possible
    - Avoid unnecessary work
    - Race back to idle

Summary

- Software has a huge impact on energy efficiency
- To extend battery life
  - Stay idle as long as possible
  - Avoid unnecessary work
  - Race back to idle
- Avoiding timers allows a longer idle time
  - Instead, use event based API
  - If you must use timers, add tolerance

Summary

- Software has a huge impact on energy efficiency
- To extend battery life
  - Stay idle as long as possible
  - Avoid unnecessary work
  - Race back to idle
- Avoiding timers allows a longer idle time
  - Instead, use event based API
  - If you must use timers, add tolerance
- Use activity API to inform system of important user work

Related Sessions

Maximizing Battery Life on OS X	Mission Tuesday 11:30AM	🟩
Building Efficient OS X Apps	Nob Hill Tuesday 4:30PM	🟩
Power and Performance: Optimizing Your Website for Great Battery Life and Responsive Scrolling	Russian Hill Wednesday 9:00AM	🟪
Energy Best Practices	Marina Thursday 10:15AM	🟩

Labs

Cocoa Lab	Frameworks Lab A Wednesday 11:30AM
Cocoa Lab	Frameworks Lab A Thursday 9:00AM
Cocoa Lab	Frameworks Lab A Friday 9:00AM

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Apple Developer Forums
http://devforums.apple.com

WWDC2013