Core OS

#WWDC14

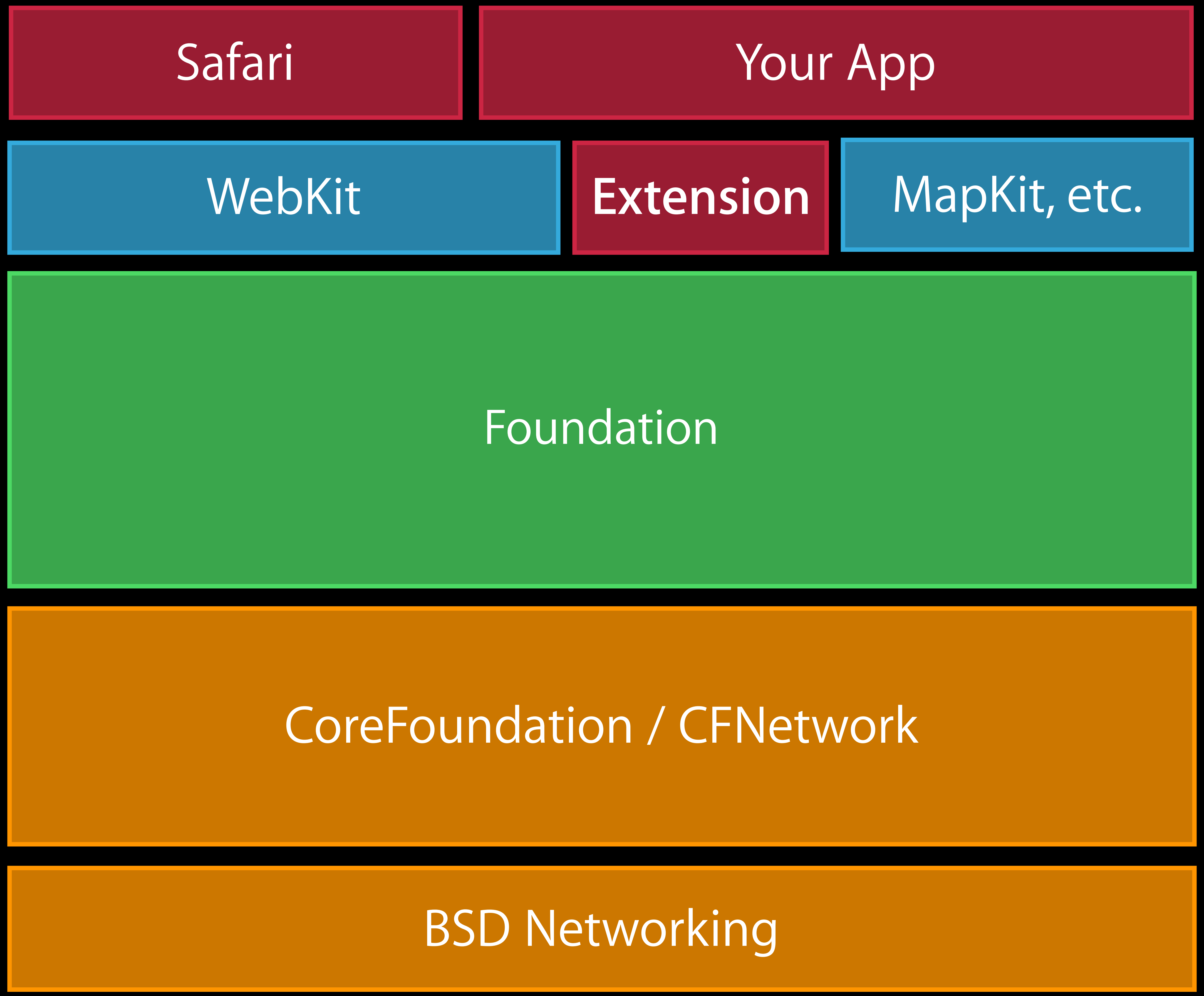
What's New in Foundation Networking

Session 707
Steve Algernon
Senior Wrangler

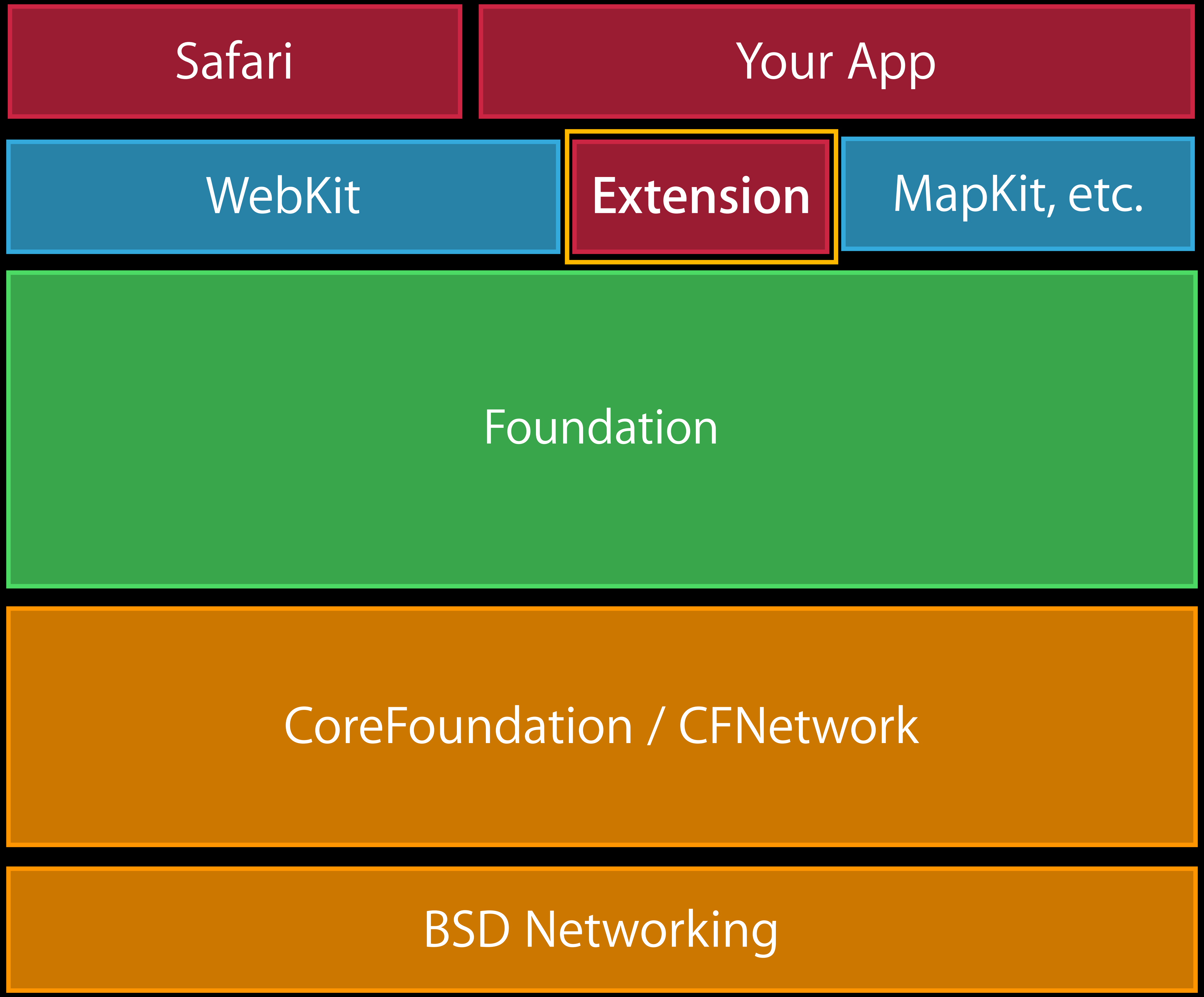
Introduction

Foundation Networking provides high-level, secure communication APIs and provides the basis for iOS and Mac OS X application networking.

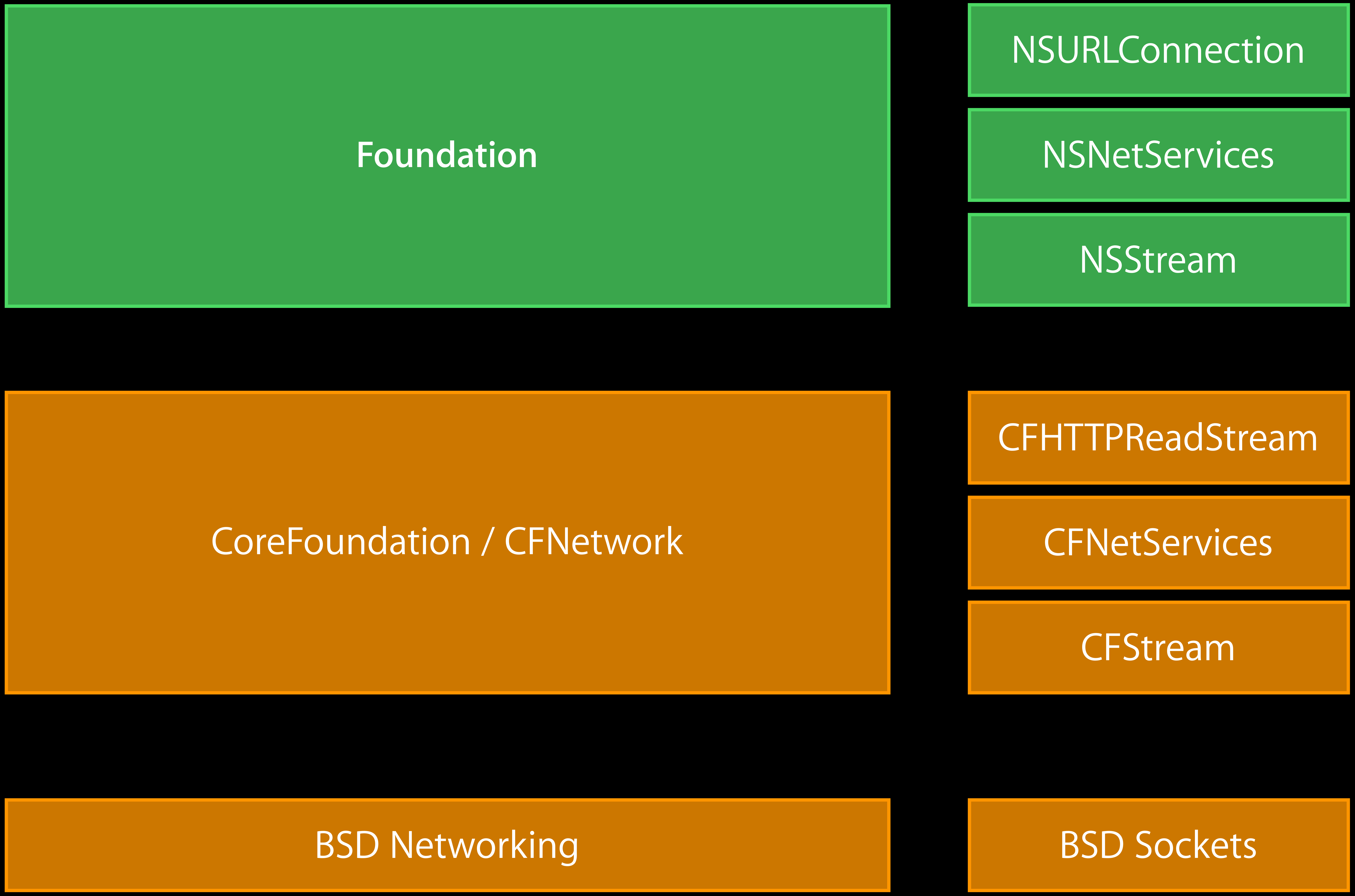
Foundation Networking



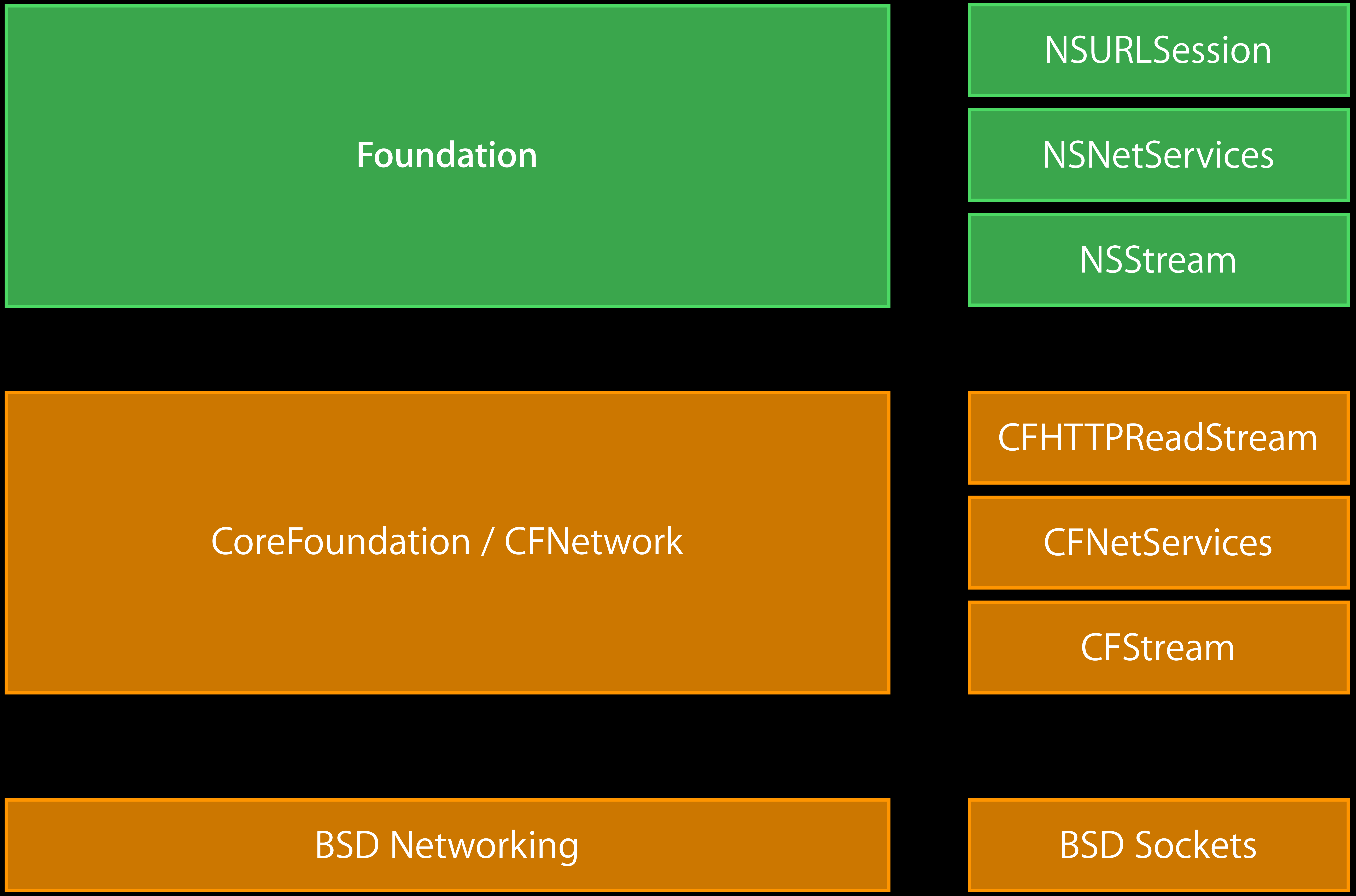
Foundation Networking



Foundation Networking



Foundation Networking



What You Will Learn

What You Will Learn

New API

- NSNetServies
- NSStream
- NSURLSession

What You Will Learn

New API

- NSNetServies
- NSStream
- NSURLSession

Review of NSURLSession

What You Will Learn

New API

- NSNetServies
- NSStream
- NSURLSession

Review of NSURLSession

New protocol support

What You Will Learn

New API

- NSNetServies
- NSStream
- NSURLSession

Review of NSURLSession

New protocol support

Background sessions and extensions—Best practices

Foundation Networking

New API

NEW

NSNetServices

```
@property BOOL includesPeerToPeer NS_AVAILABLE(10_10, 7_0);
```

Foundation Networking

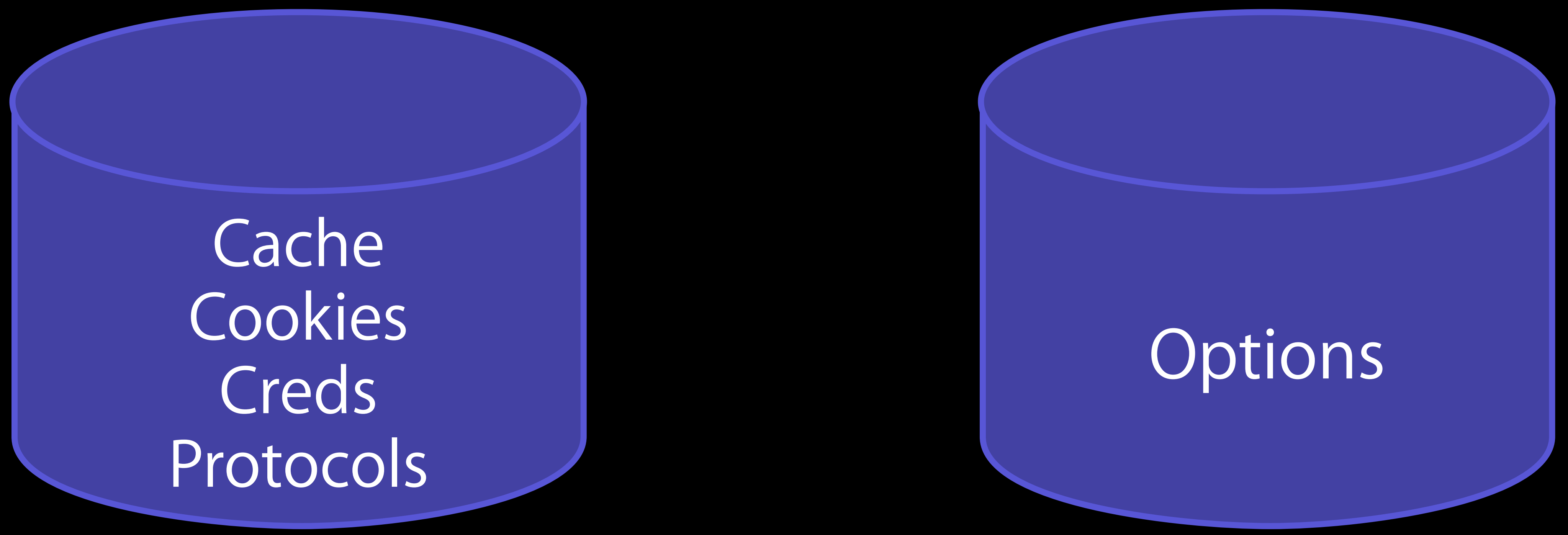
New API

NSStream

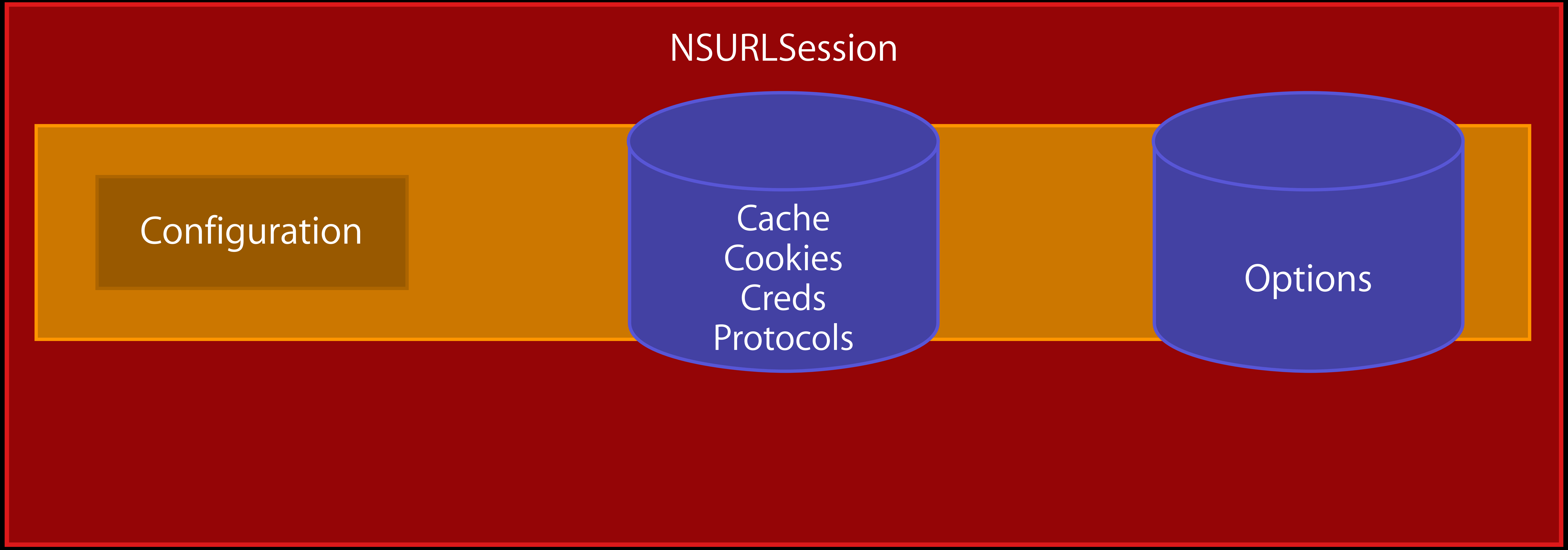
```
+[NSStream getStreamsToHostWithName:(NSString *)host
                              port:(NSInteger) port
                       inputStream:(NSInputStream **) sin
                      outputStream:(NSOutputStream *) sout];


+[NSStream getBoundStreamsWithBufferSize:(NSUInteger)bufferSize
                             inputStream:(NSInputStream **) sin
                            outputStream:(NSOutputStream **) sout];
```

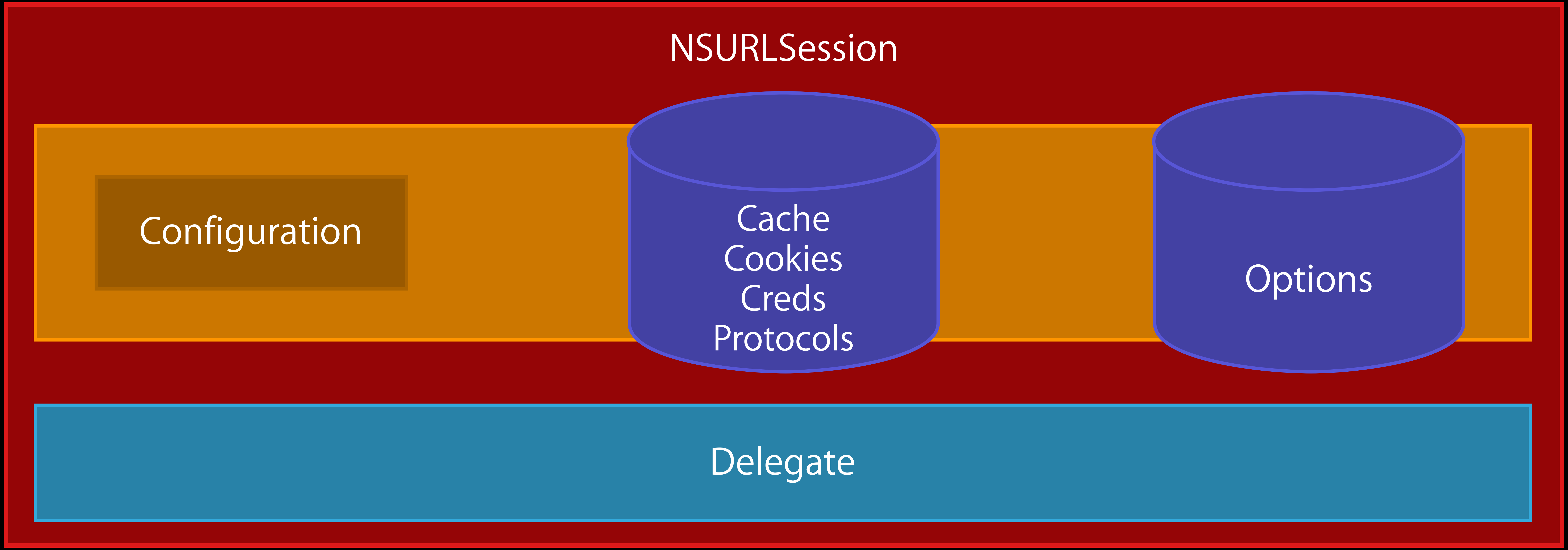
NSURLSession



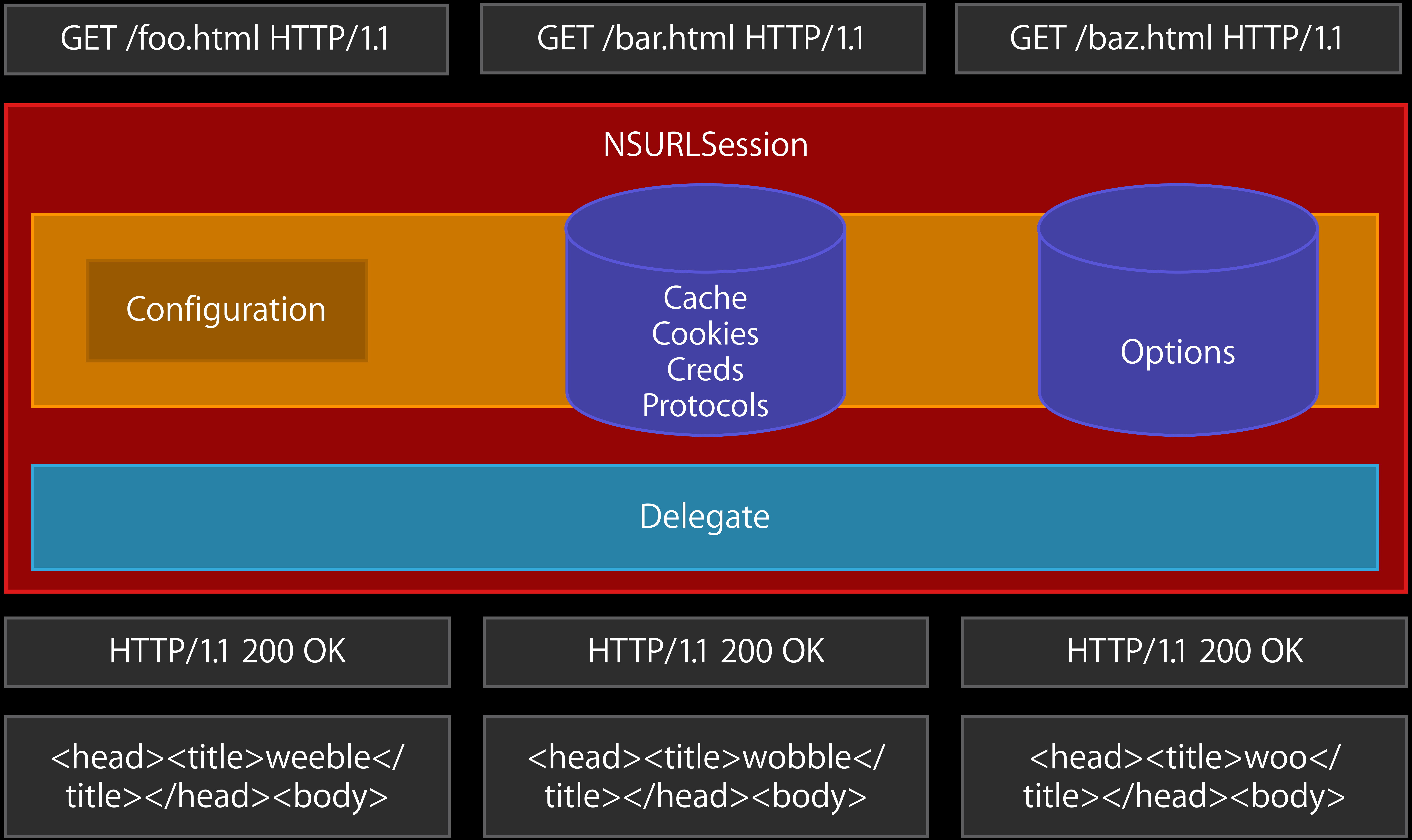
NSURLSession



NSURLSession



NSURLSession



NSURLSession

Concepts

NSURLSession

Concepts

Session and configuration

NSURLSession

Concepts

Session and configuration

Session tasks

NSURLSession

Concepts

Session and configuration

Session tasks

Session delegate

NSURLSession

Concepts

Session and configuration

Session tasks

Session delegate

Credentials, credential storage

NSURLSession

Concepts

Session and configuration

Session tasks

Session delegate

Credentials, credential storage

Cookies, cookie storage

NSURLSession

Concepts

Session and configuration

Session tasks

Session delegate

Credentials, credential storage

Cookies, cookie storage

Protocols

NSURLSession

Concepts

Session and configuration

Session tasks

Session delegate

Credentials, credential storage

Cookies, cookie storage

Protocols

URL cache

NSURLSession

Configuration

NSURLSession

Configuration

NSURLSessionConfiguration object

NSURLSession
Configuration

NSURLSessionConfiguration object

Properties that affect transfers

- TLS levels
- Cellular usage
- Network service type
- Cookie policies
- Cache policies
- Storage objects
- Request & resource timeouts

NSURLSession

Configuration

NSURLSession

Configuration

```
+ [NSURLSessionConfiguration defaultSessionConfiguration]
```

Best place to start for customization

Modifications only affect **this** configuration object

NSURLSession

Configuration

```
+ [NSURLSessionConfiguration defaultSessionConfiguration]
```

Best place to start for customization

Modifications only affect **this** configuration object

```
+ [NSURLSessionConfiguration ephemeralSessionConfiguration]
```

Does not persist cache, credentials or cookies

NSURLSession

Configuration

```
+ [NSURLSessionConfiguration defaultSessionConfiguration]
```

Best place to start for customization

Modifications only affect **this** configuration object

```
+ [NSURLSessionConfiguration ephemeralSessionConfiguration]
```

Does not persist cache, credentials or cookies

```
+ [NSURLSessionConfiguration backgroundSessionConfigurationWithIdentifier:]
```

Create or reassociate with a background session

NSURLSession

Creation

NSURLSession

Creation

```
+ [NSURLSession sharedSession]
```

For delegate-less, simple asynchronous requests

NSURLSession

Creation

`+ [NSURLSession sharedSession]`

For delegate-less, simple asynchronous requests

`+ [NSURLSession sessionWithConfiguration:]`

Custom configuration, but no delegate

Great to use with ephemeral configuration

NSURLSession

Creation

`+ [NSURLSession sharedSession]`

For delegate-less, simple asynchronous requests

`+ [NSURLSession sessionWithConfiguration:]`

Custom configuration, but no delegate

Great to use with ephemeral configuration

`+ [NSURLSession sessionWithConfiguration:delegate:delegateQueue:]`

Maximum flexibility through delegates

Required interface for background sessions

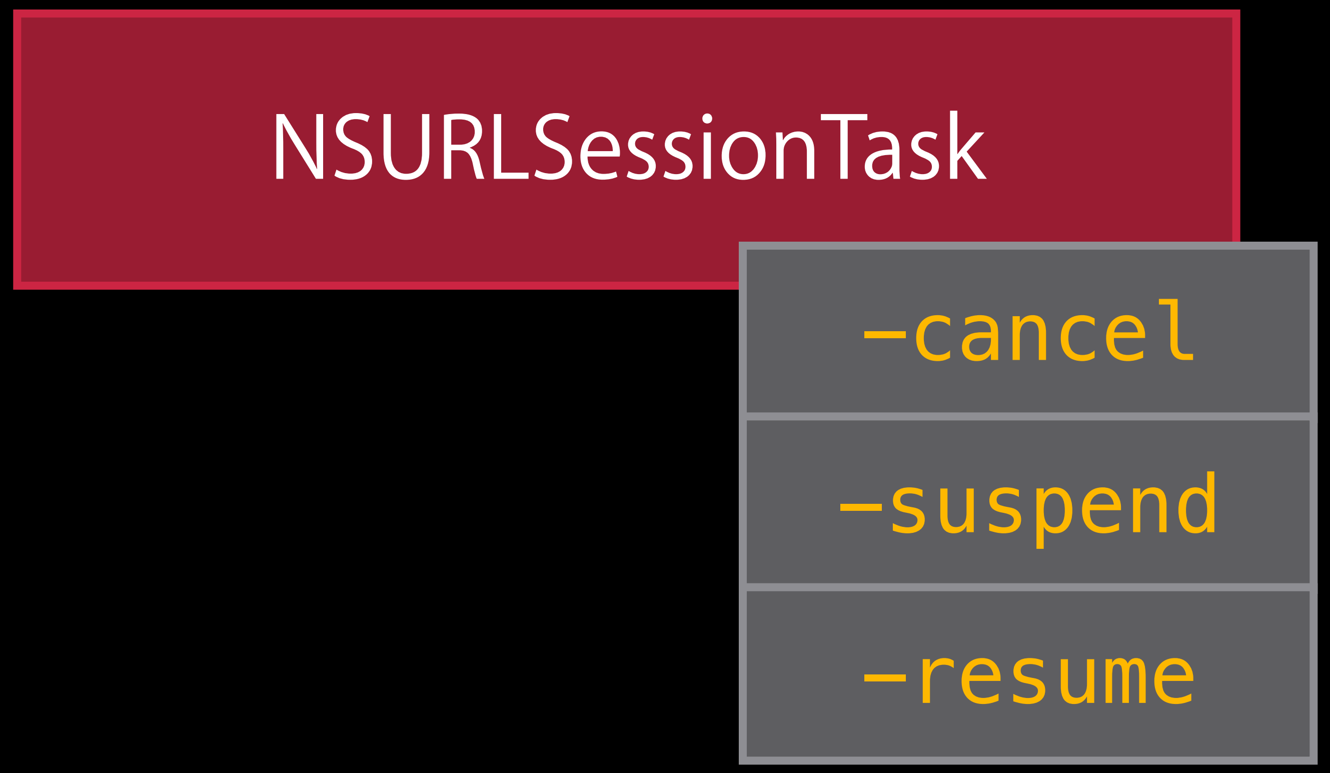
`delegateQueue` may be concurrent

NSURLSession

Tasks

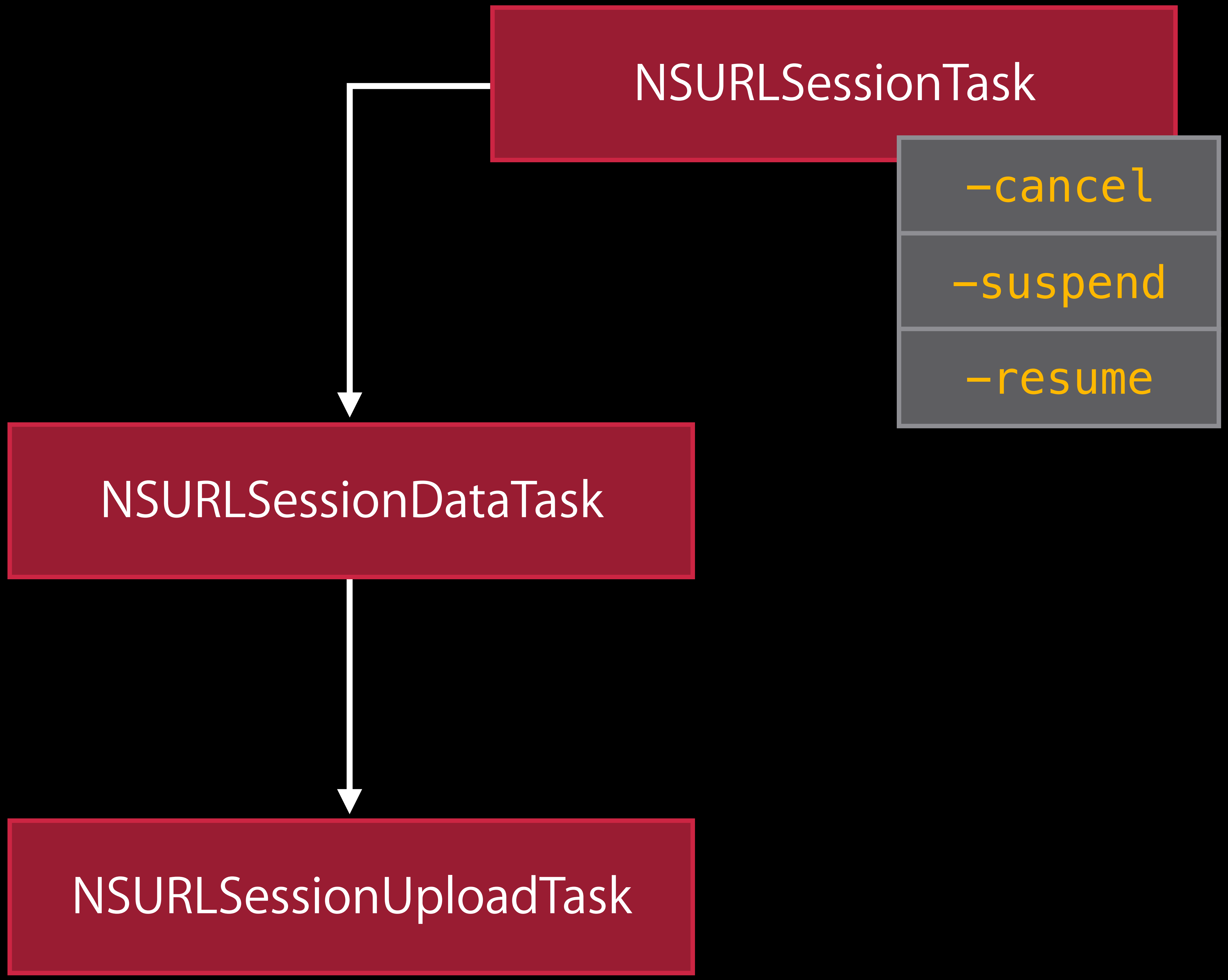
NSURLSession

Tasks



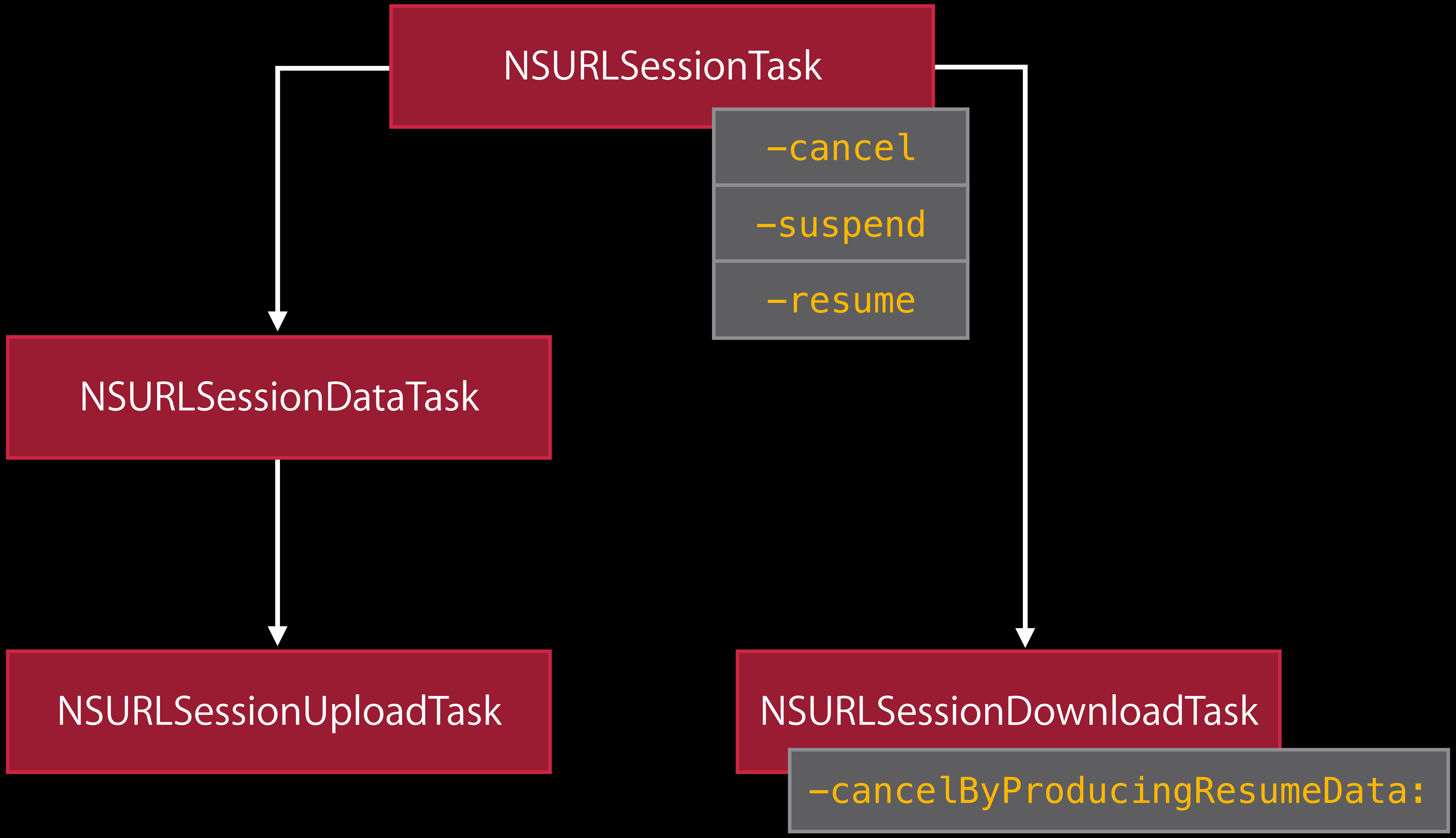
NSURLSession

Tasks



NSURLSession

Tasks



NSURLSessionTask

Creation

NSURLSessionTask

Creation

Data task

```
-dataTaskWithURL:
-dataTaskWithRequest:
```

NSURLSessionTask

Creation

Data task

```
-dataTaskWithURL:
-dataTaskWithRequest:
```

Upload task

```
-uploadTaskWithRequest:fromFile:
-uploadTaskWithRequest:fromData:
-uploadTaskWithStreamedRequest:
```

NSURLSessionTask

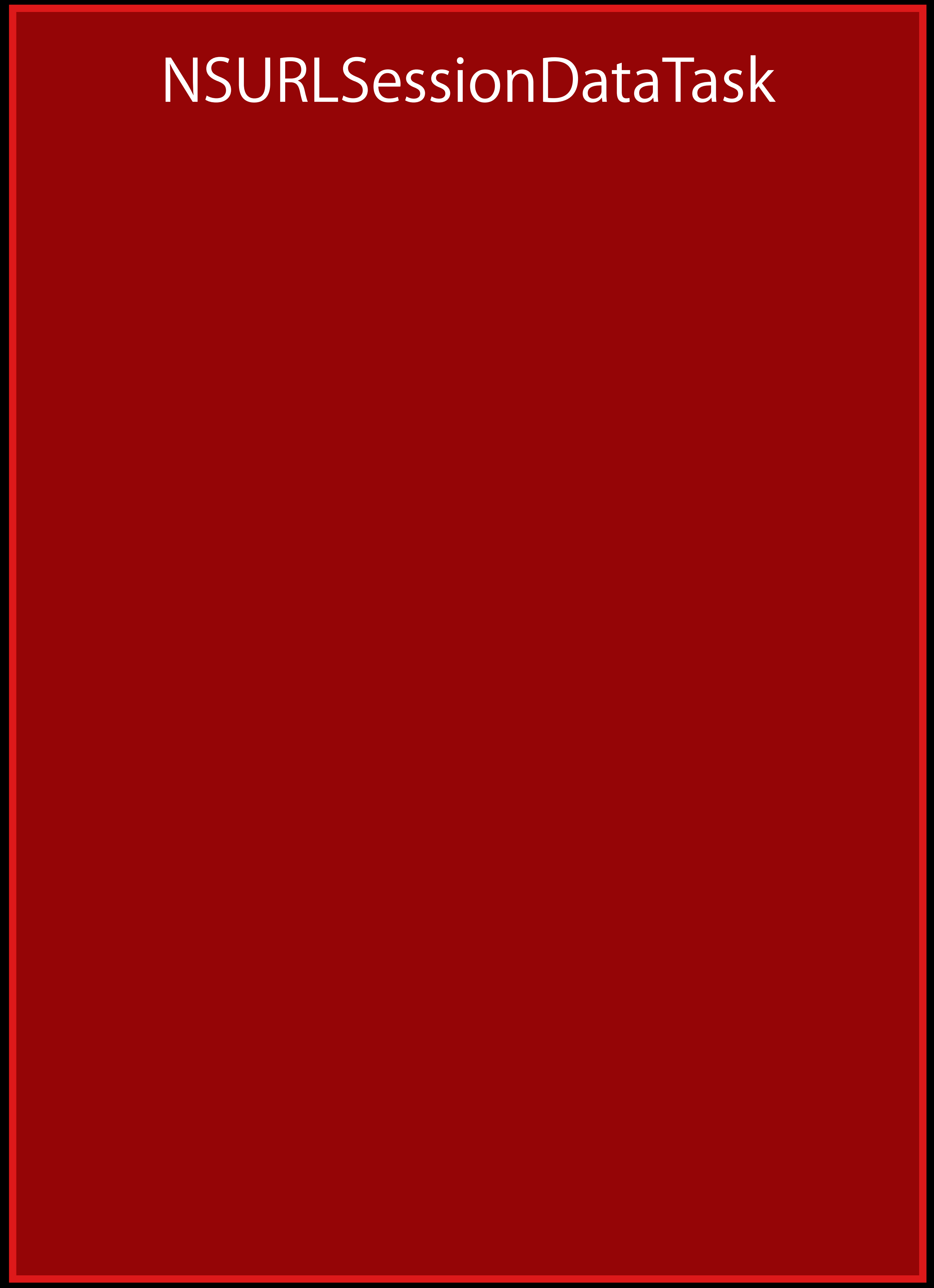
Creation

Download task

```
-downloadTaskWithURL:
-downloadTaskWithRequest:
-downloadTaskWithResumeData:
```

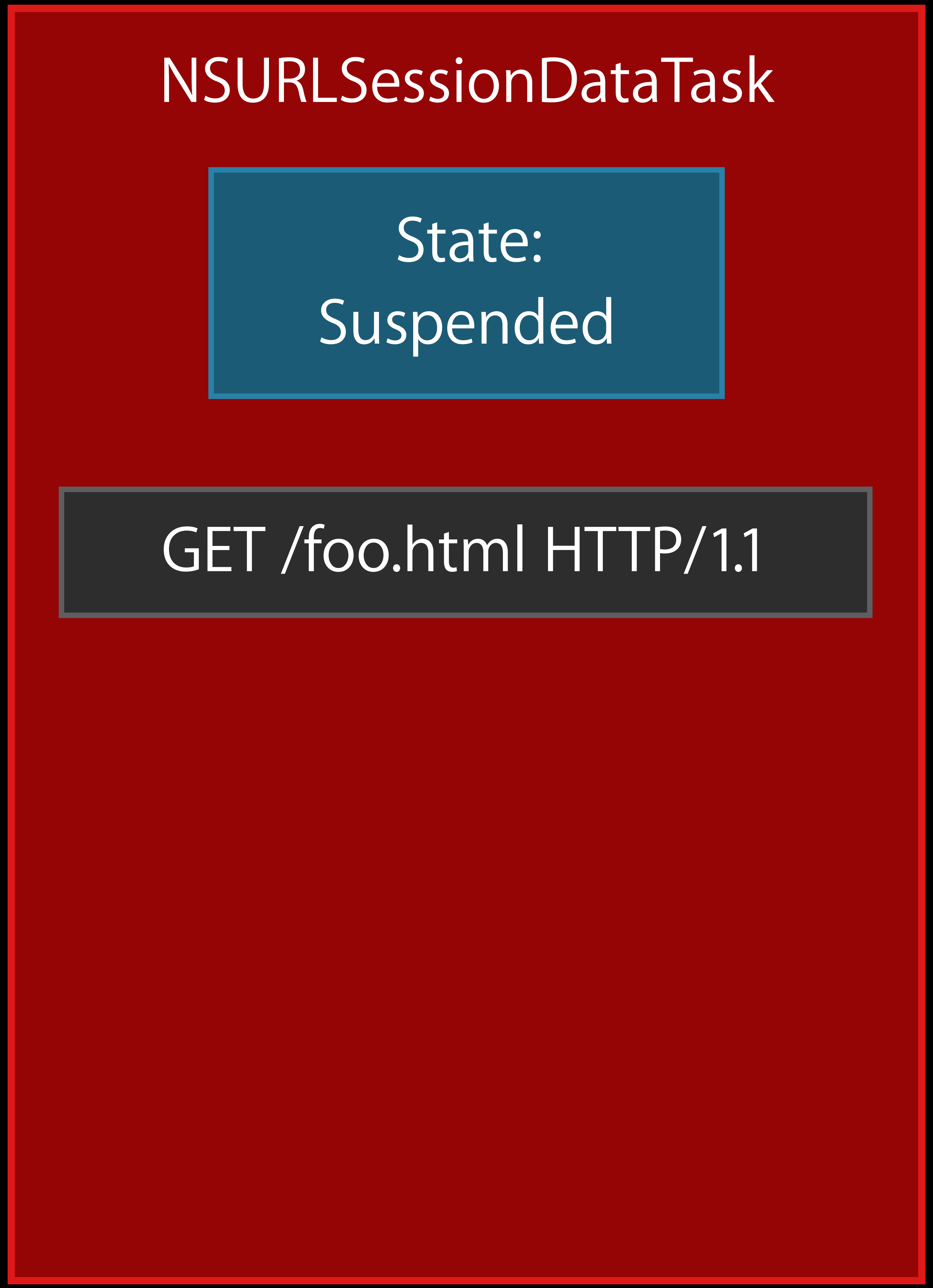
NSURLSession

Data task



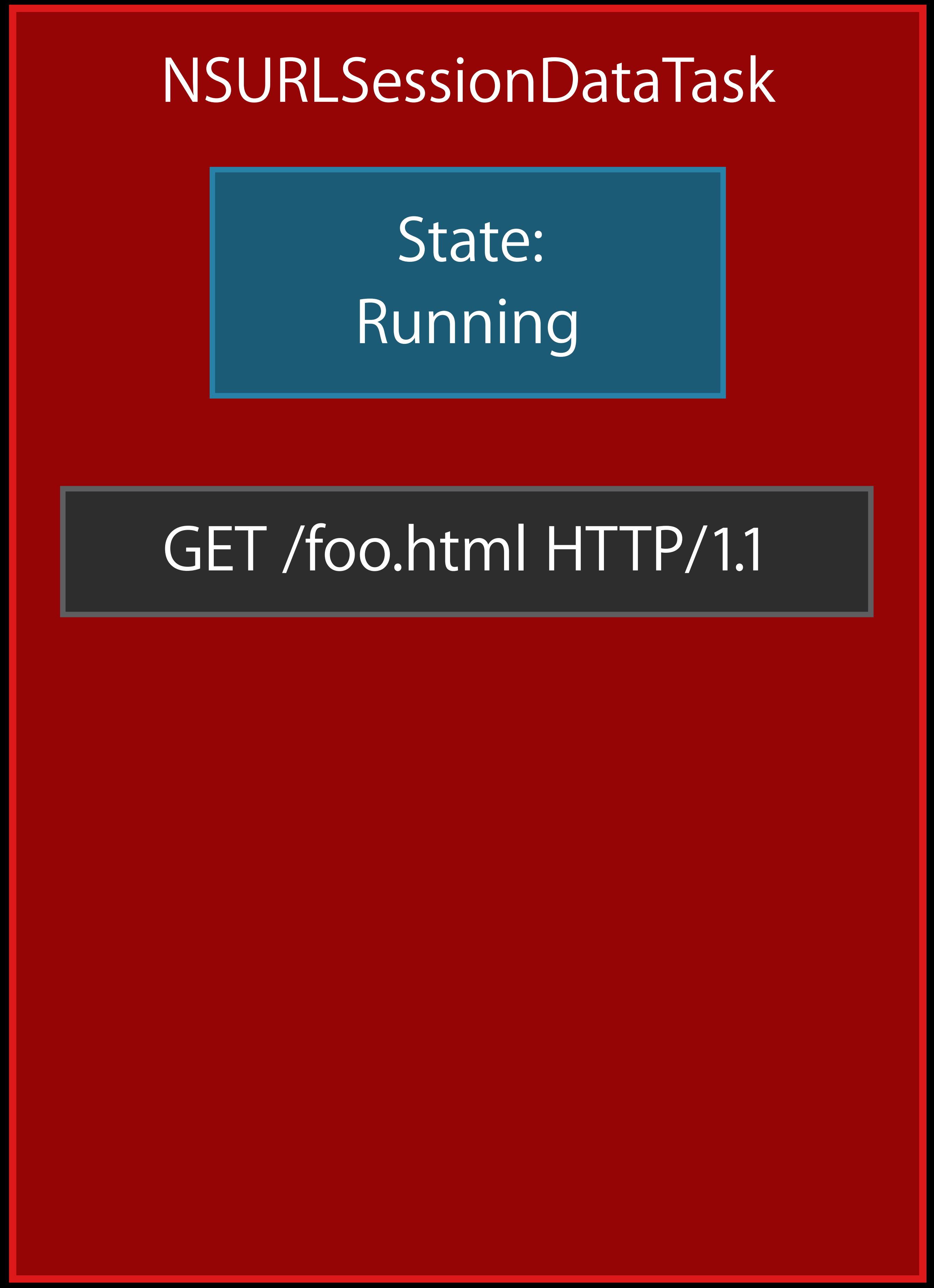
NSURLSession

Data task



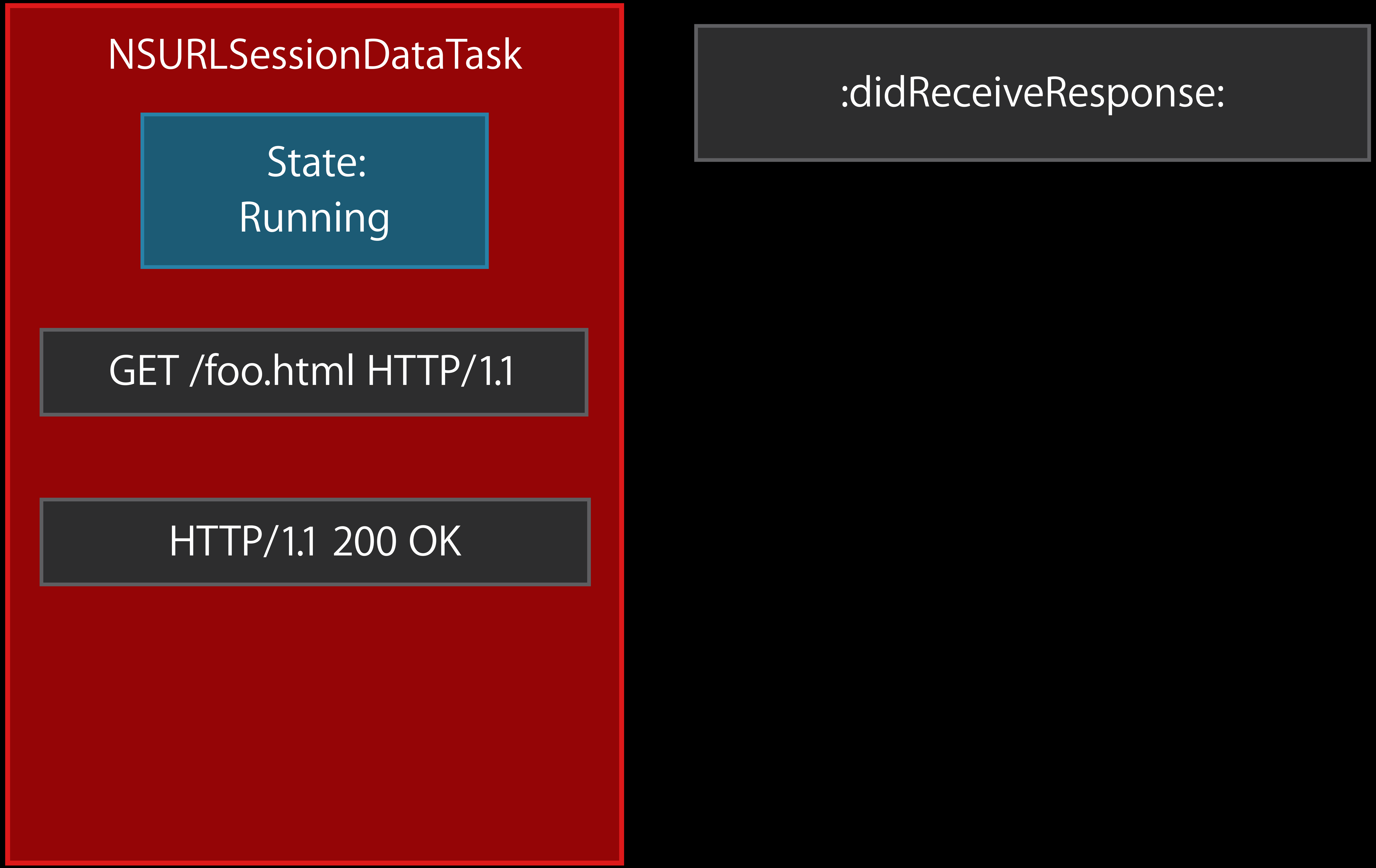
NSURLSession

Data task



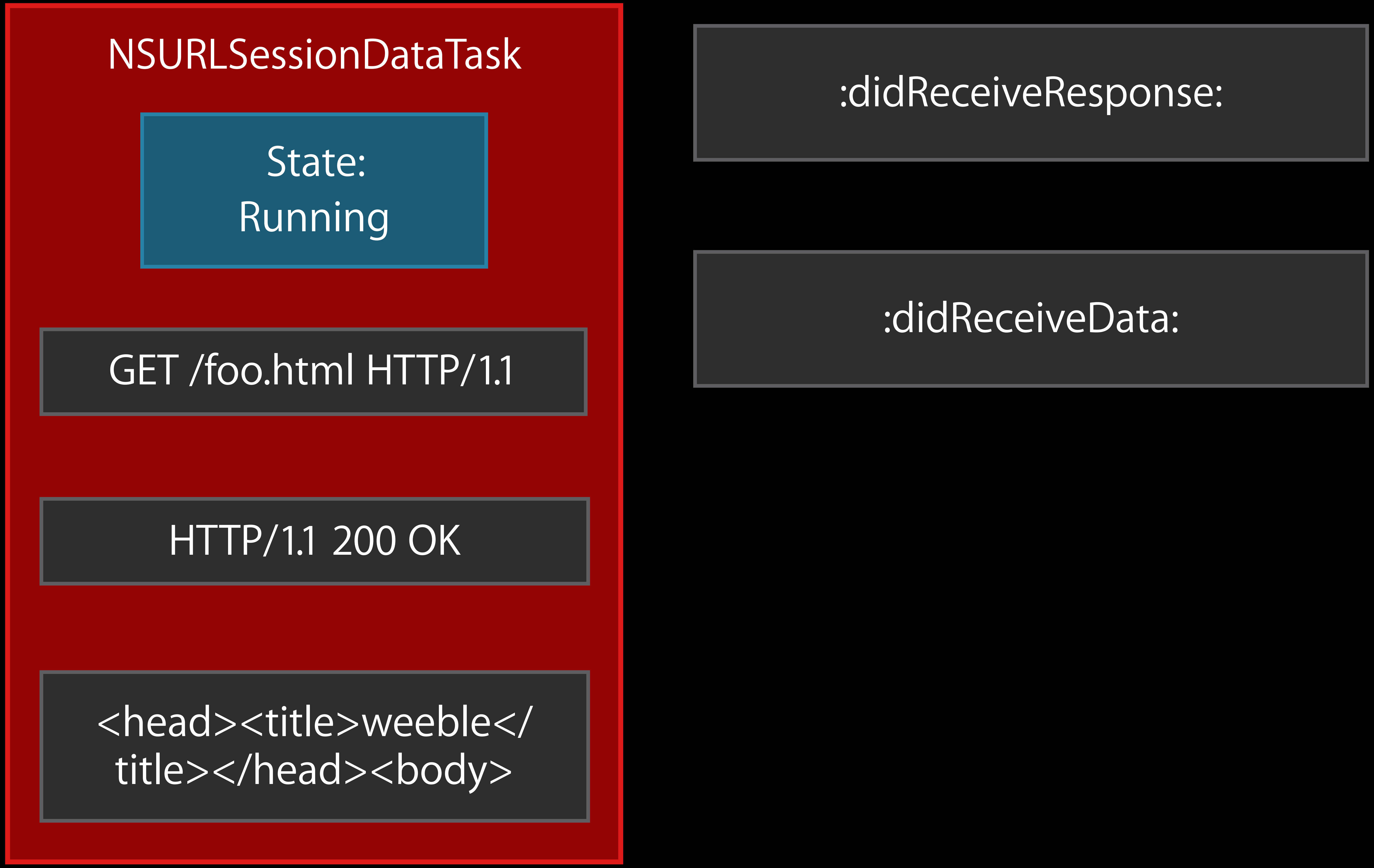
NSURLSession

Data task



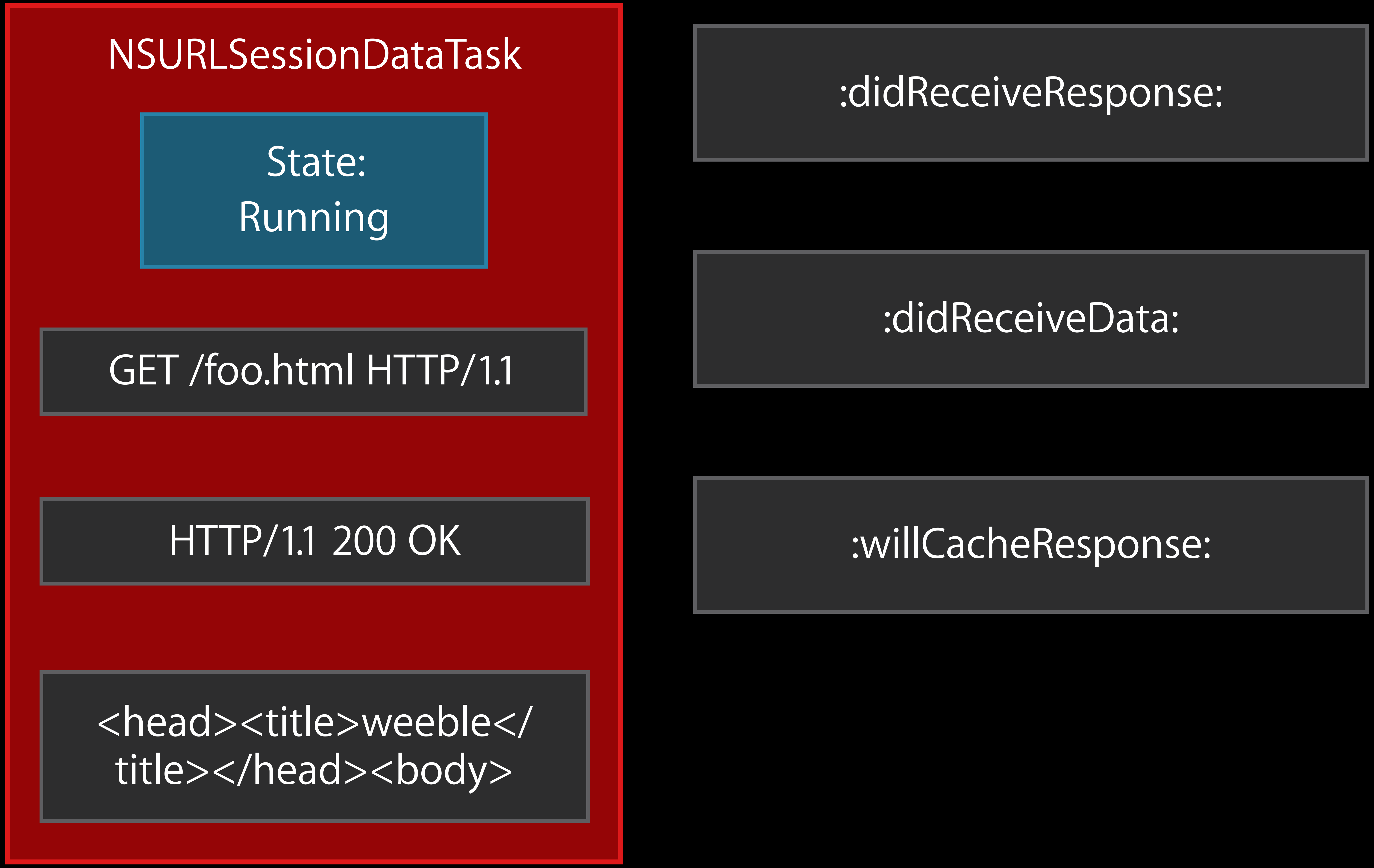
NSURLSession

Data task



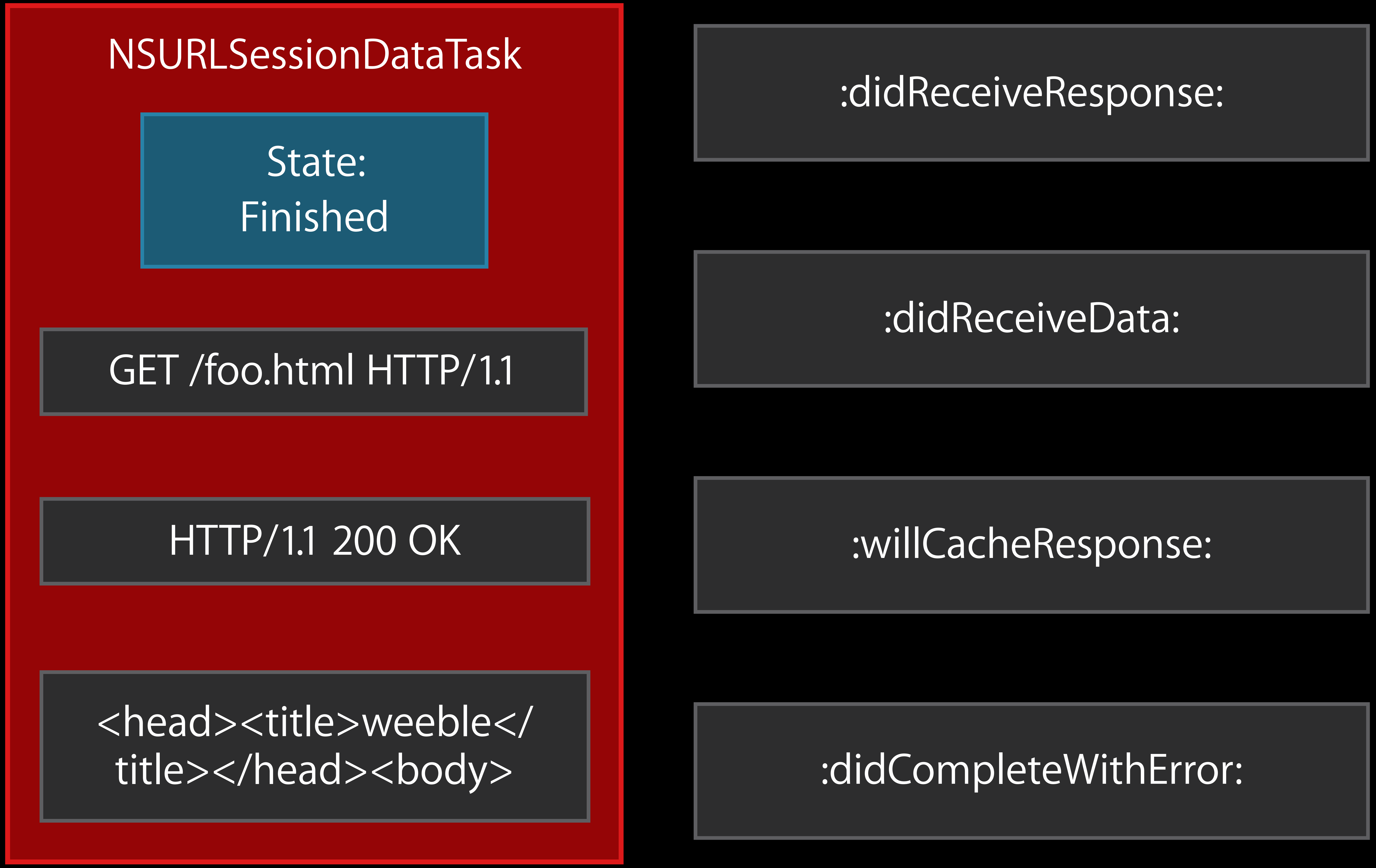
NSURLSession

Data task



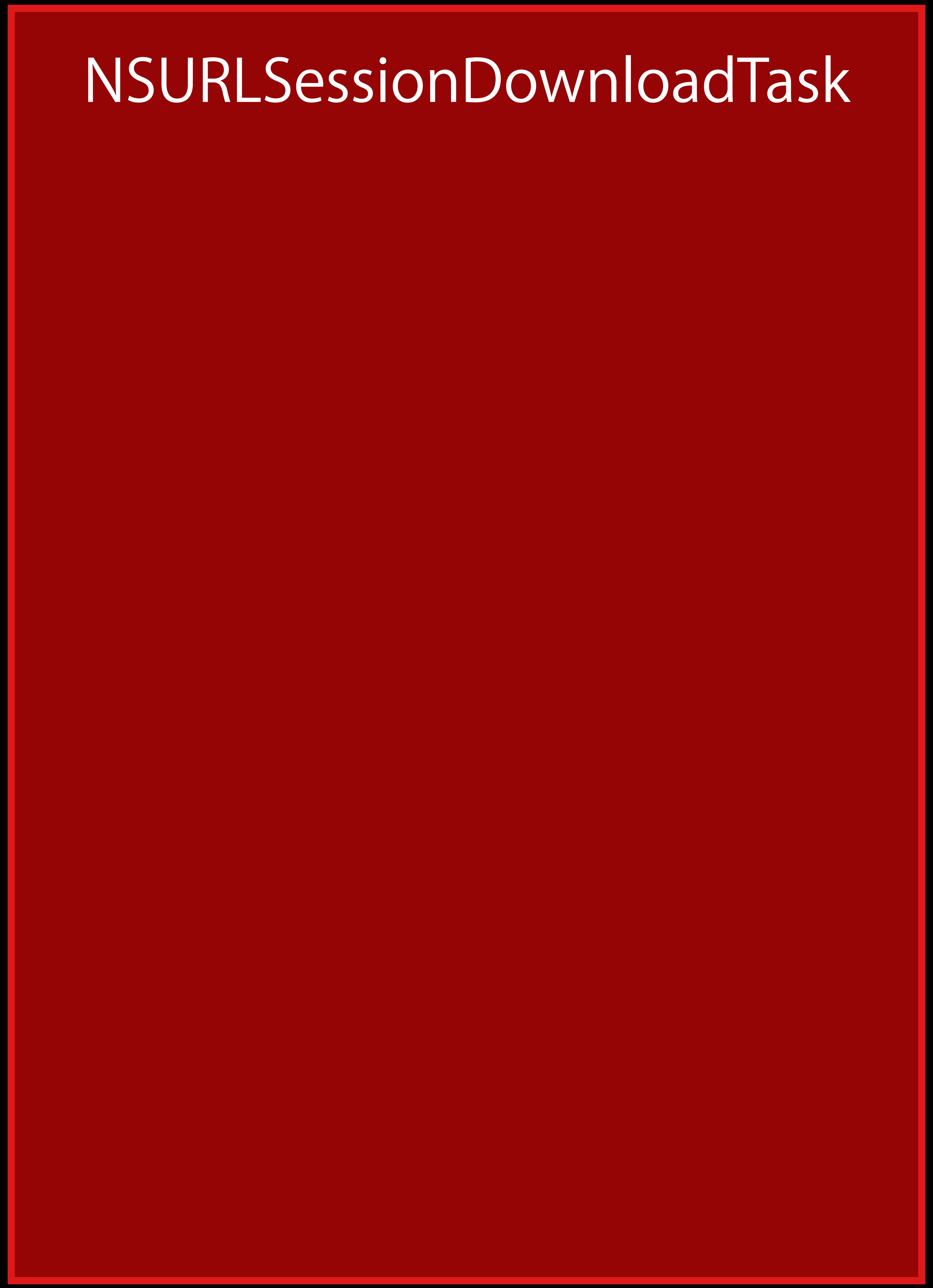
NSURLSession

Data task



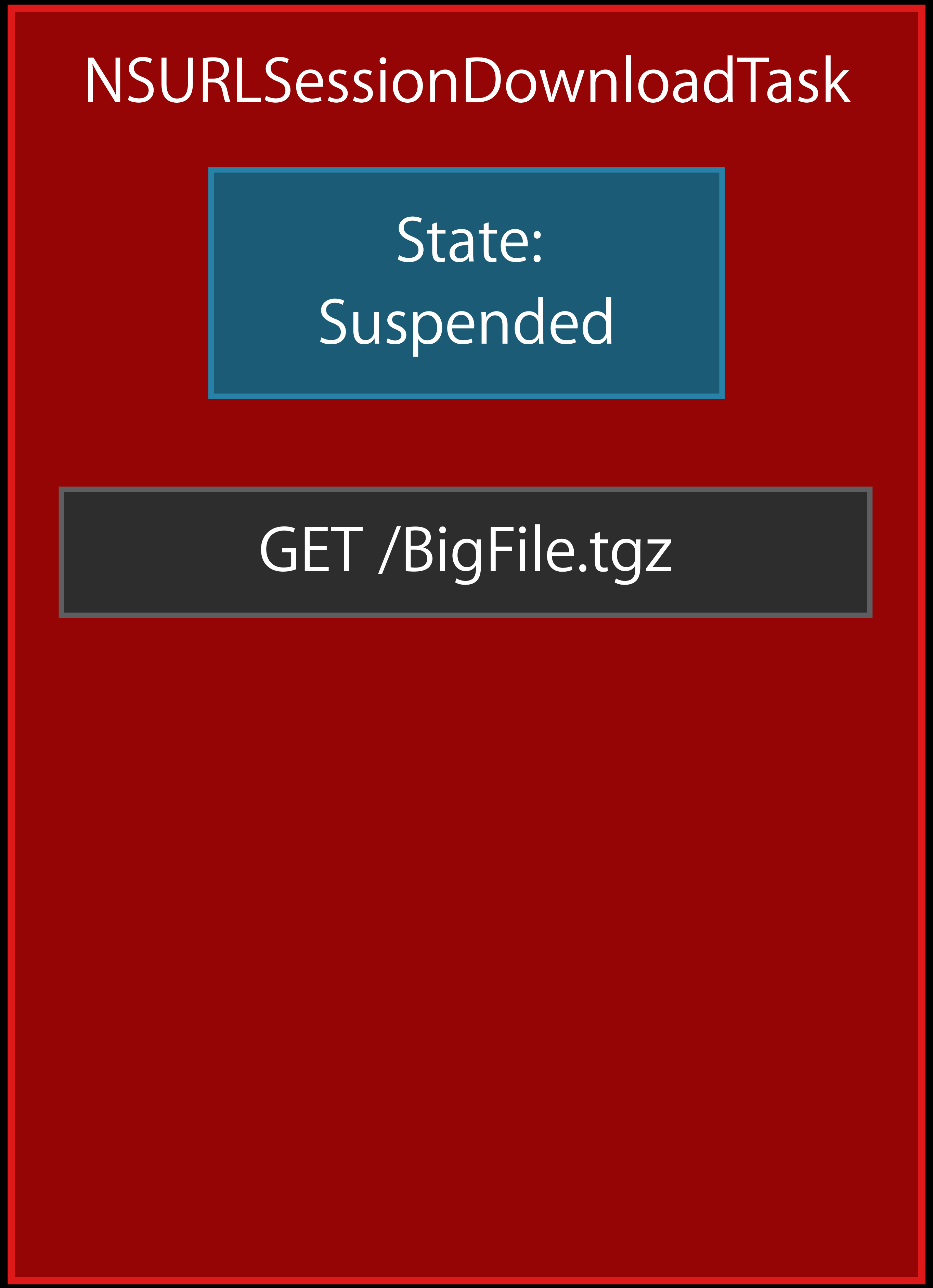
NSURLSession

Download task



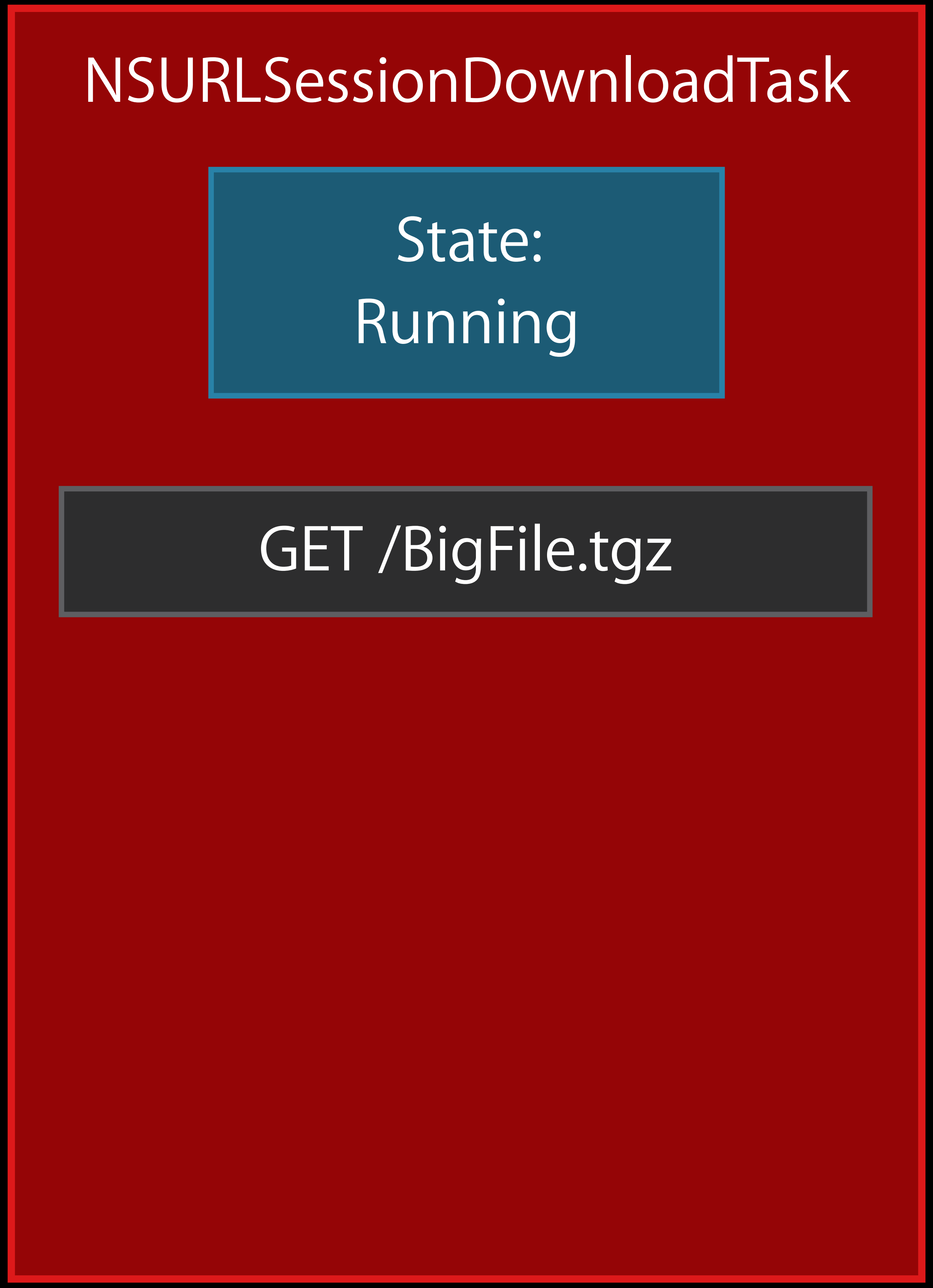
NSURLSession

Download task



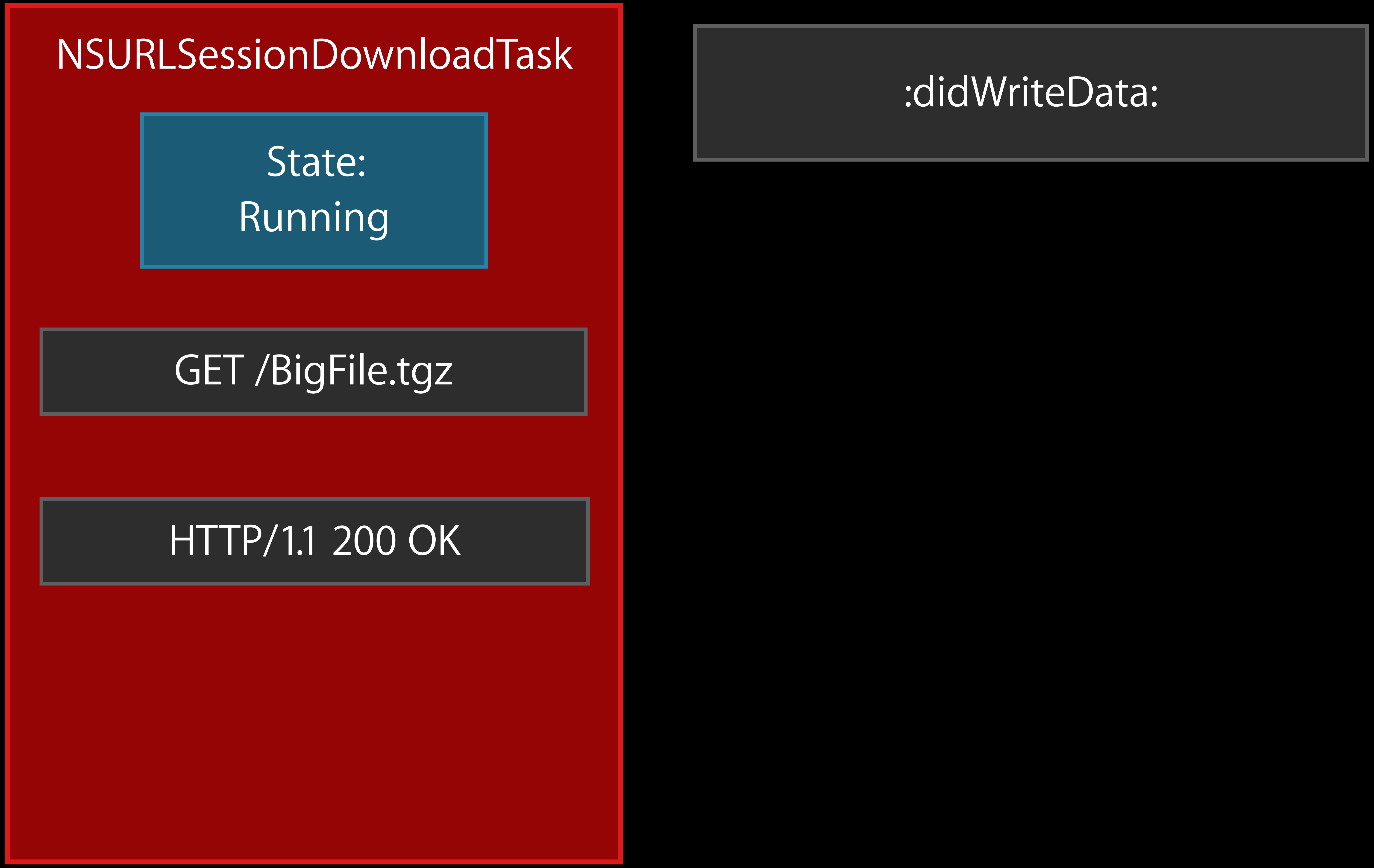
NSURLSession

Download task



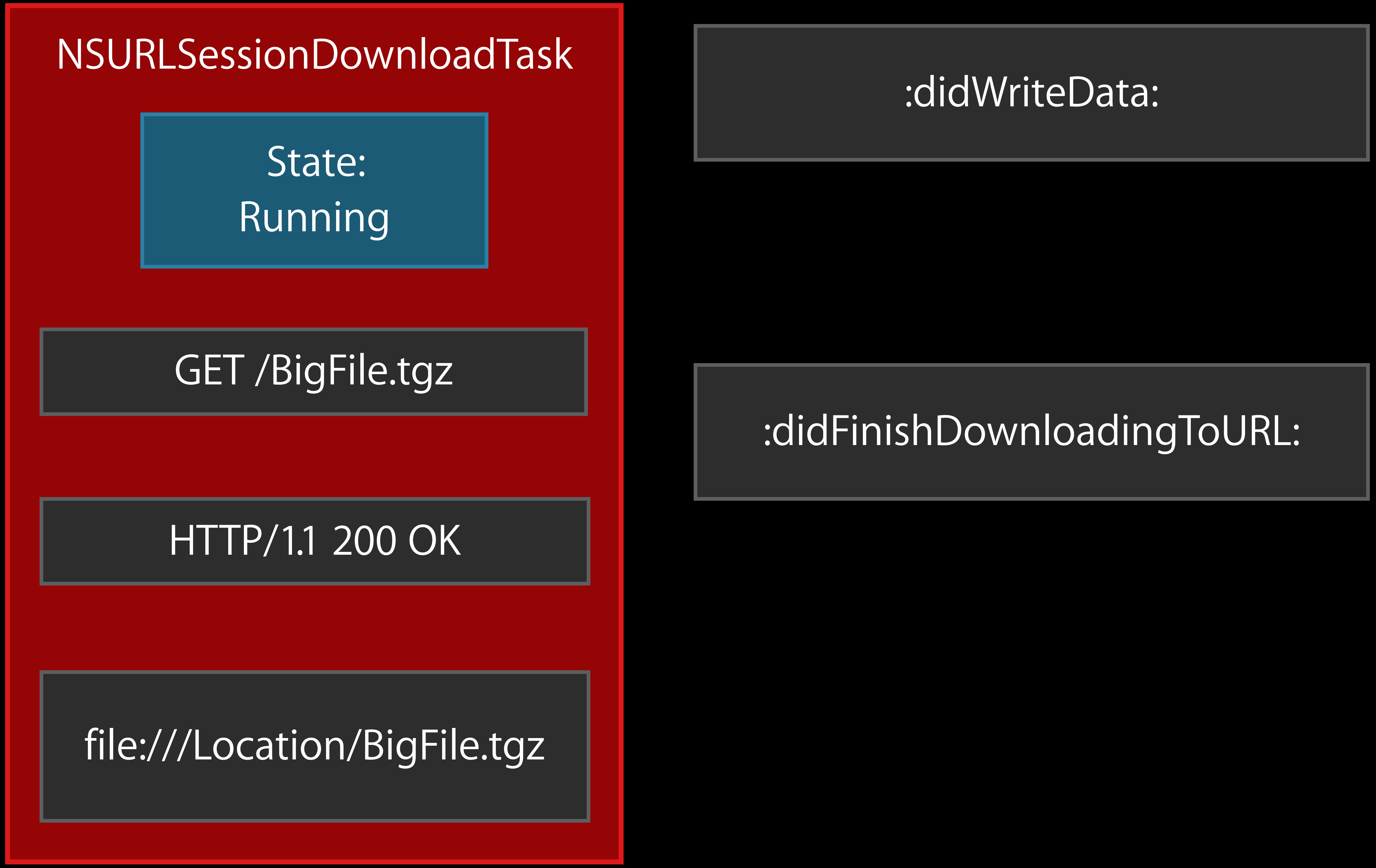
NSURLSession

Download task



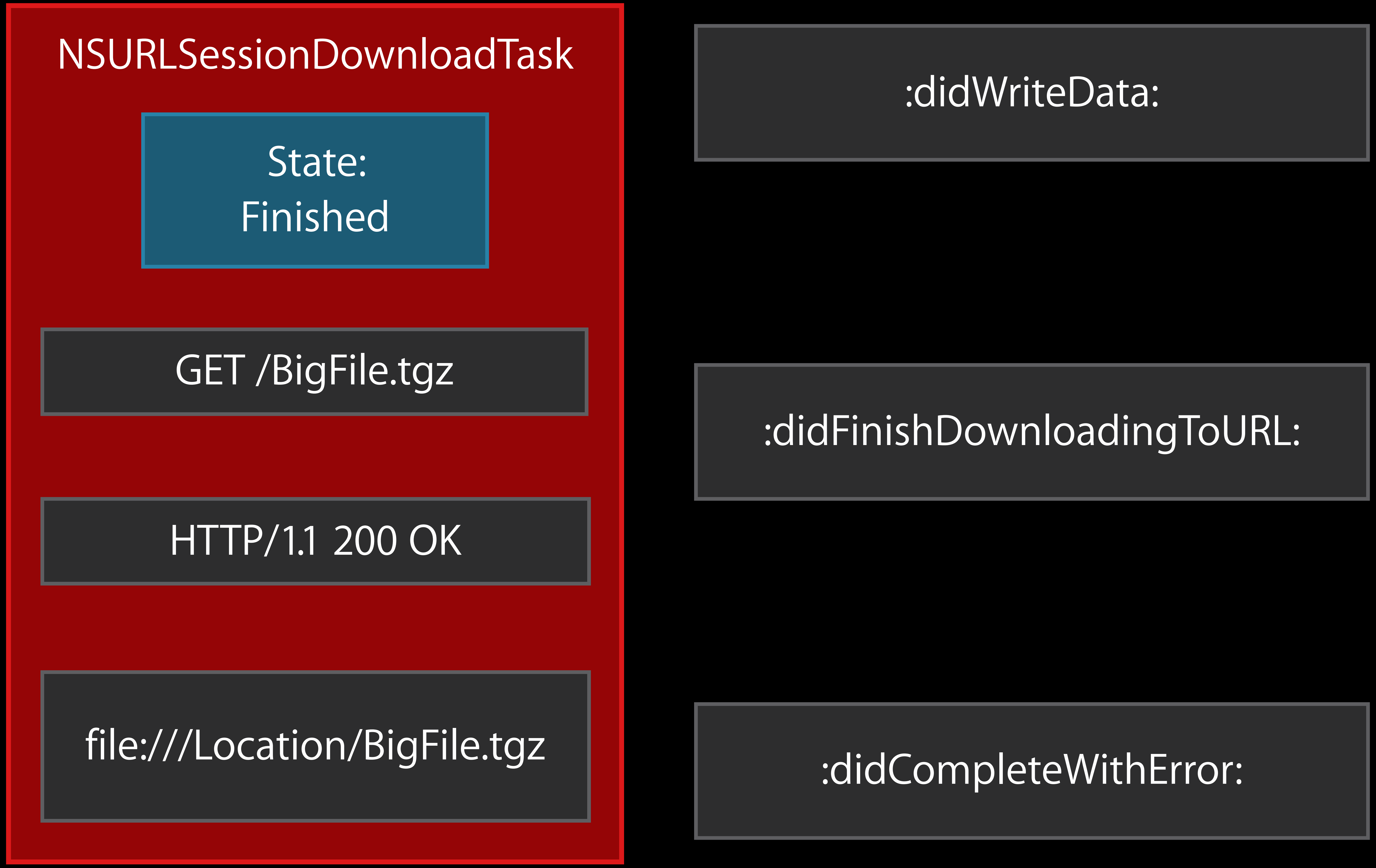
NSURLSession

Download task



NSURLSession

Download task



NSURLSessionTask

Creation

NSURLSessionAsynchronousConvenience

Tasks may be canceled

Ignores session delegate—except for authentication challenges

Cannot be used with background sessions

`-dataTaskWithURL:completionHandler:`

`-downloadTaskWithURL:completionHandler:`

`-uploadTaskWithRequest:fromFile:completionHandler:`

NSURLSessionTask

Creation

```
NSURL *myPrivateURL = [NSURL URLWithString:@“http://example.com/secret”];
NSURLSessionConfiguration* myConfiguration =
             [NSURLSessionConfiguration ephemeralSessionConfiguration];
NSURLSession* mySession =
             [NSURLSession sessionWithConfiguration:myConfiguration];
NSURLSessionTask* myTask = [mySession dataTaskWithURL:mySecretURL
                                completionHandler:^(NSData* data,
                                NSURLResponse* response, NSError* error) {
    [self gotSecret:data];
}];
[myTask resume]
```

NSURLSessionTask

Creation

```
NSURL *myPrivateURL = [NSURL URLWithString:@“http://example.com/secret”];
NSURLSessionConfiguration* myConfiguration =
             [NSURLSessionConfiguration ephemeralSessionConfiguration];
NSURLSession* mySession =
             [NSURLSession sessionWithConfiguration:myConfiguration];
NSURLSessionTask* myTask = [mySession dataTaskWithURL:mySecretURL
                                completionHandler:^(NSData* data,
                                NSURLResponse* response, NSError* error) {
    [self gotSecret:data];
}];
[myTask resume]
```

NSURLSessionTask

Creation

```
NSURL *myPrivateURL = [NSURL URLWithString:@“http://example.com/secret”];
NSURLSessionConfiguration* myConfiguration =
             [NSURLSessionConfiguration ephemeralSessionConfiguration];
NSURLSession* mySession =
             [NSURLSession sessionWithConfiguration:myConfiguration];
NSURLSessionTask* myTask = [mySession dataTaskWithURL:mySecretURL
                                 completionHandler:^(NSData* data,
                                 NSURLResponse* response, NSError* error) {
    [self gotSecret:data];
}];
[myTask resume]
```

NSURLSessionTask

Creation

```
NSURL *myPrivateURL = [NSURL URLWithString:@“http://example.com/secret”];
NSURLSessionConfiguration* myConfiguration =
             [NSURLSessionConfiguration ephemeralSessionConfiguration];
NSURLSession* mySession =
             [NSURLSession sessionWithConfiguration:myConfiguration];
NSURLSessionTask* myTask = [mySession dataTaskWithURL:mySecretURL
                                completionHandler:^(NSData* data,
                                NSURLResponse* response, NSError* error) {
   [self gotSecret:data];
}];
[myTask resume]
```

NSURLSessionTask

Creation

```
NSURL *myPrivateURL = [NSURL URLWithString:@“http://example.com/secret”];
NSURLSessionConfiguration* myConfiguration =
             [NSURLSessionConfiguration ephemeralSessionConfiguration];
NSURLSession* mySession =
             [NSURLSession sessionWithConfiguration:myConfiguration];
NSURLSessionTask* myTask = [mySession dataTaskWithURL:mySecretURL
                                completionHandler:^(NSData* data,
                                NSURLResponse* response, NSError* error) {
   [self gotSecret:data];
}];
[myTask resume]
```

NSURLSessionTask

Creation

```
NSURL *myPrivateURL = [NSURL URLWithString:@“http://example.com/secret”];
NSURLSessionConfiguration* myConfiguration =
             [NSURLSessionConfiguration ephemeralSessionConfiguration];
NSURLSession* mySession =
             [NSURLSession sessionWithConfiguration:myConfiguration];
NSURLSessionTask* myTask = [mySession dataTaskWithURL:mySecretURL
                                completionHandler:^(NSData* data,
                                NSURLResponse* response, NSError* error) {
    [self gotSecret:data];
}];
[myTask resume]
```

NSURLSessionTask

Creation

```
NSURL *myPrivateURL = [NSURL URLWithString:@“http://example.com/secret”];
NSURLSessionConfiguration* myConfiguration =
             [NSURLSessionConfiguration ephemeralSessionConfiguration];
NSURLSession* mySession =
             [NSURLSession sessionWithConfiguration:myConfiguration];
NSURLSessionTask* myTask = [mySession dataTaskWithURL:mySecretURL
                                completionHandler:^(NSData* data,
                                NSURLResponse* response, NSError* error) {
   [self gotSecret:data];
}];
[myTask resume]
```

NSURLSession

Delegate

NSURLSession

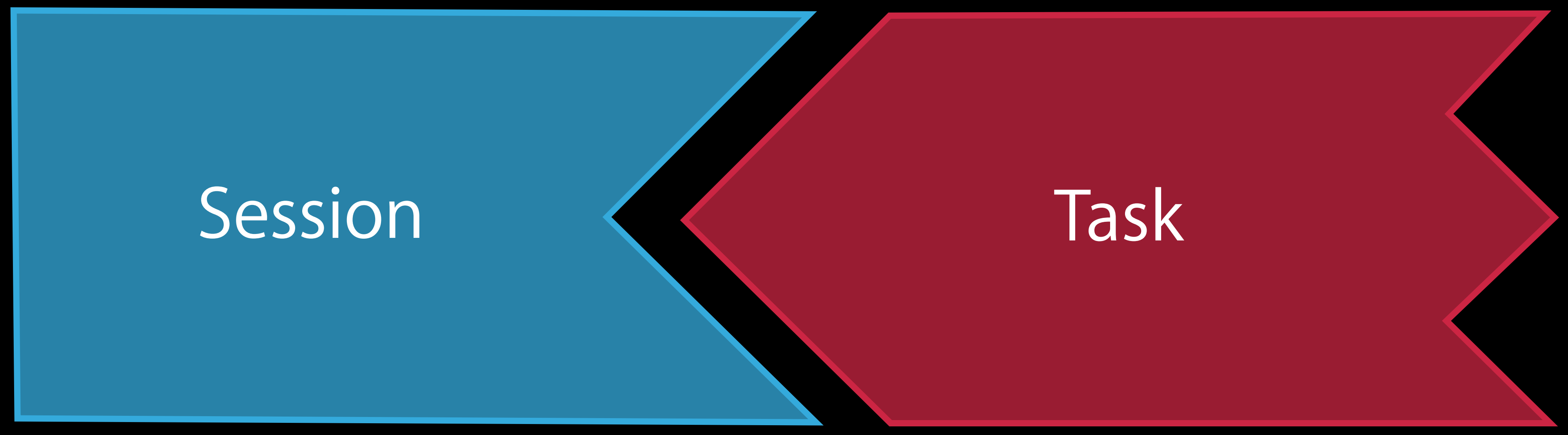
Delegate

`NSURLSessionDelegate`

- Session-related delegate messages
- Connection authentication handling
- Session invalidation/errors

`NSURLSessionTaskDelegate`

- Extends NSURLSessionDelegate
- Request authentication handling
- Task completion/errors

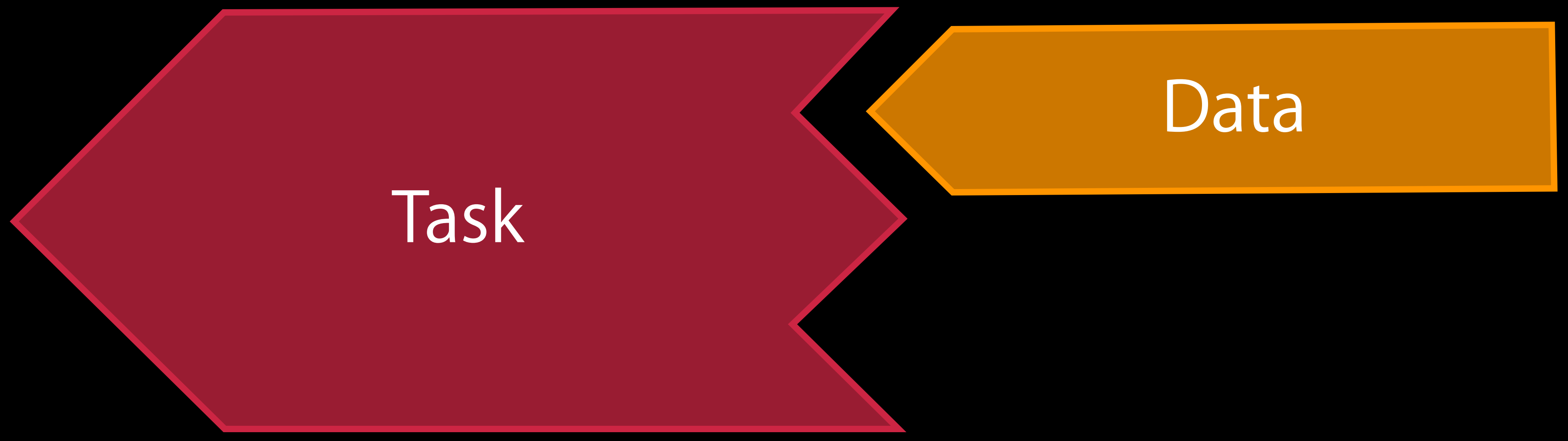


NSURLSession

Delegate

`NSURLSessionDataDelegate`

- Extends NSURLSessionTask protocol
- Delivers bytes as they are transferred
- :didReceiveResponse: disposition



NSURLSession

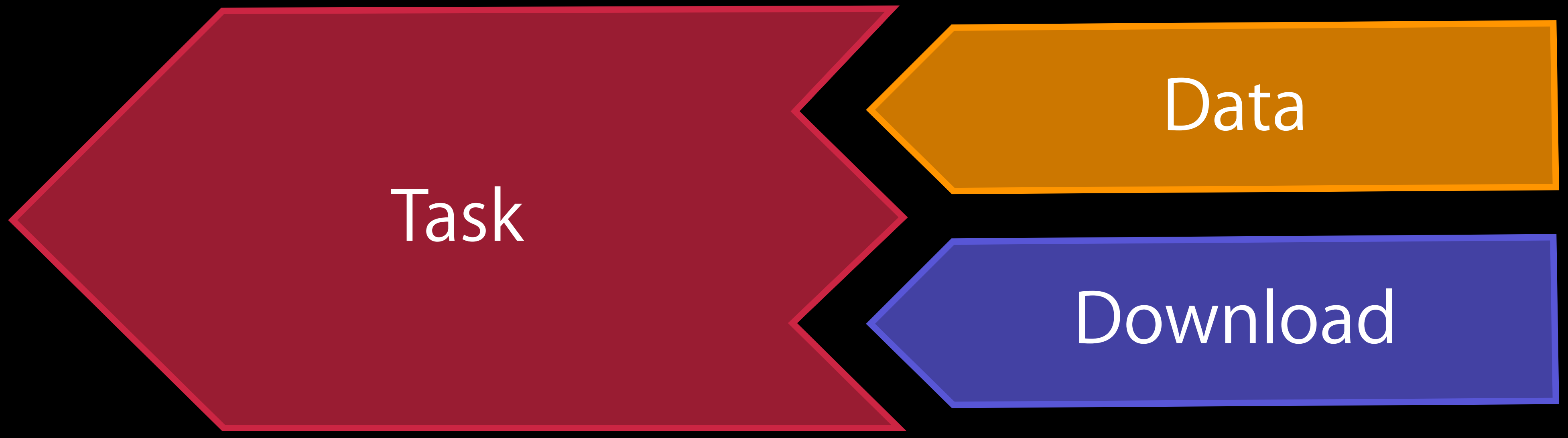
Delegate

`NSURLSessionDataDelegate`

- Extends NSURLSessionTask protocol
- Delivers bytes as they are transferred
- :didReceiveResponse: disposition

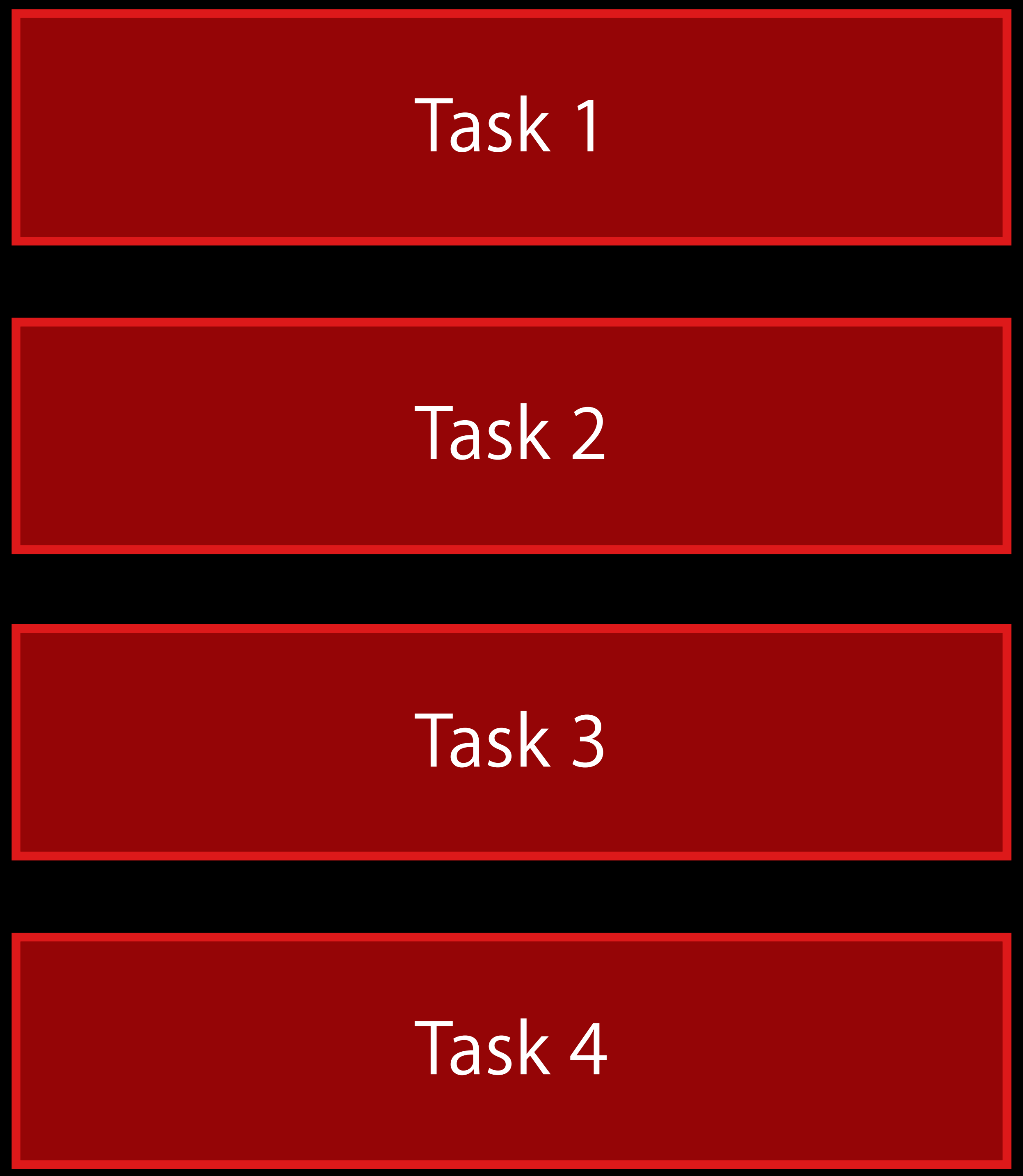
`NSURLSessionDownloadDelegate`

- Extends NSURLSessionTask protocol
- Delivers progress during a transfer
- Provides a local file URL of the transferred resource



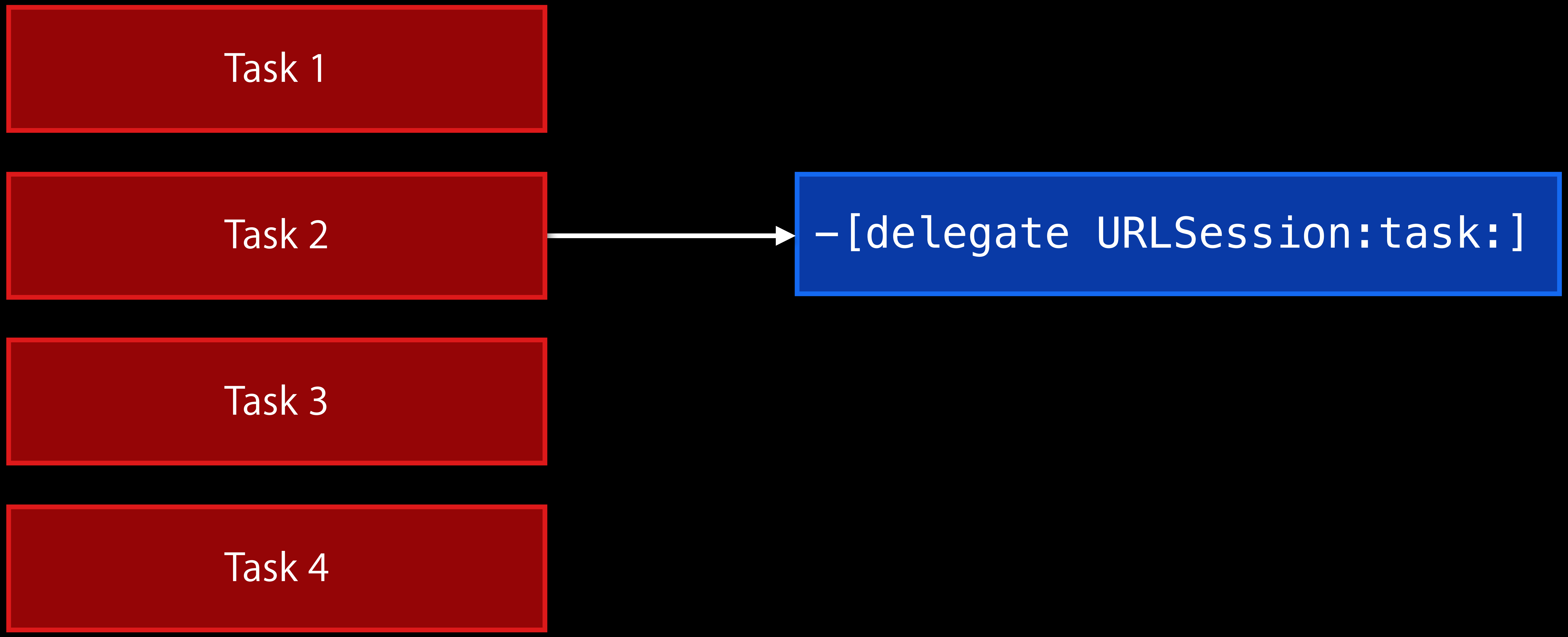
NSURLSession

Delegate queue–Serialized



NSURLSession

Delegate queue—Serialized



NSURLSession

Delegate queue–Serialized

Task 1
Task 2
Task 3
Task 4

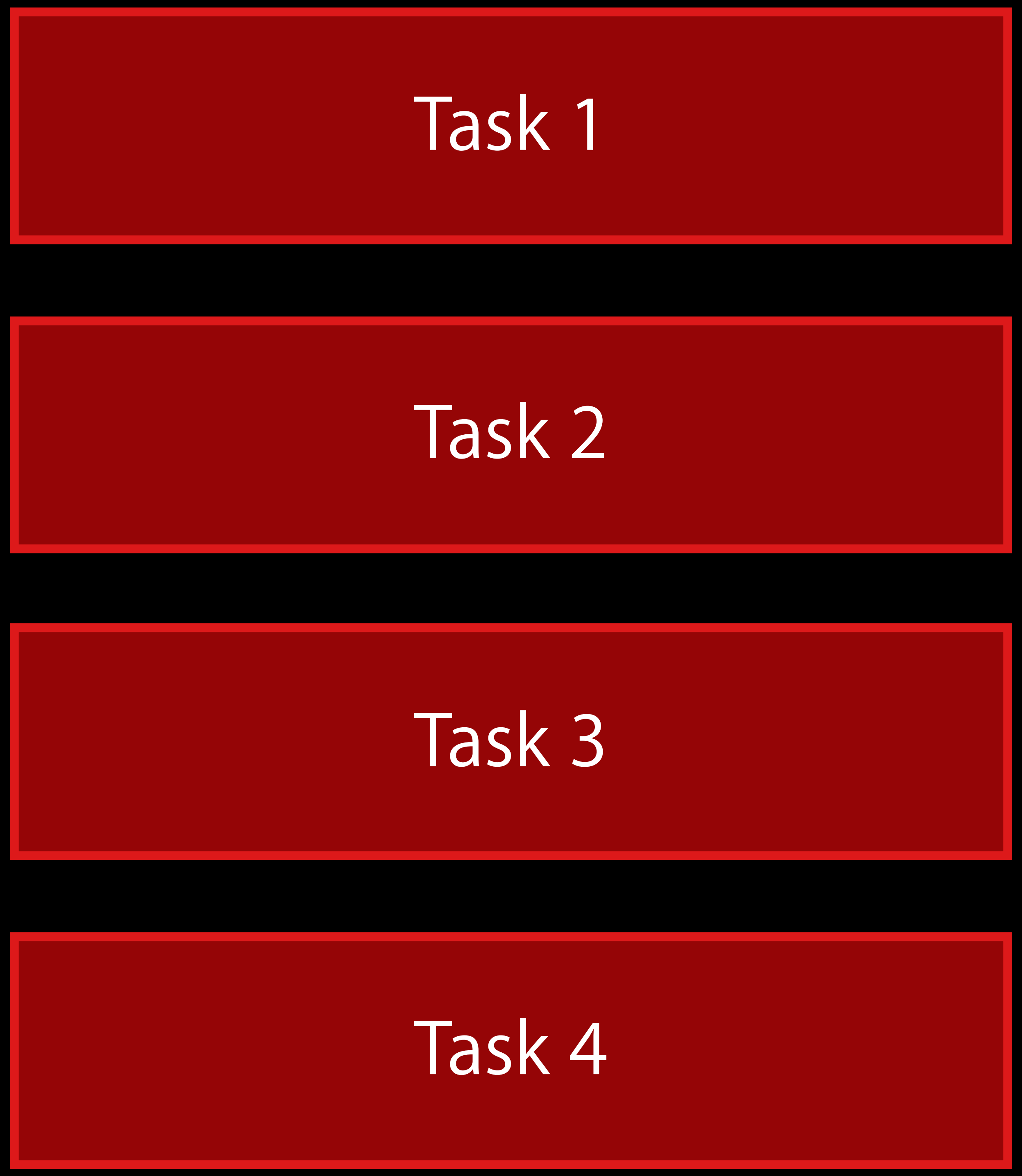
NSURLSession

Delegate queue–Concurrency

Task 1
Task 2
Task 3
Task 4

NSURLSession

Delegate queue–Concurrency



NSURLSession

New API

NEW

NSURLSession

New API

NEW

Storage objects

- NSURLSessionTaskAdditions category
- Provides asynchronous storage access

NSURLSession

New API

NEW

Storage objects

- NSURLSessionTaskAdditions category
- Provides asynchronous storage access

NSHTTPCookieStorage

```
– storeCookies:forTask:
– getCookiesForTask:completionHandler:
```

NSURLSession

New API

NEW

Storage objects

- NSURLSessionTaskAdditions category
- Provides asynchronous storage access

NSHTTPCookieStorage

```
– storeCookies:forTask:
– getCookiesForTask:completionHandler:
```

NSURLCredentialStorage

```
– getCredential:forProtectionSpace:task:completionHandler:
```

NSURLSession

New API

NEW

Storage objects

- NSURLSessionTaskAdditions category
- Provides asynchronous storage access

NSHTTPCookieStorage

```
– storeCookies:forTask:
– getCookiesForTask:completionHandler:
```

NSURLCredentialStorage

```
– getCredential:forProtectionSpace:task:completionHandler:
```

NSURLCache

```
– getCachedResponseForDataTask:completionHandler:
```

New Protocol Support

Scott Marshall
Software Engineer

Faster HTTP—SPDY

What is SPDY?

NEW

SPDY protocol support is now available in NSURLSession on OS X Yosemite and iOS 8

- Available in Safari and for use in your apps
- Leveraged by other Apple frameworks (e.g., UIWebView)

Faster HTTP—SPDY

What is SPDY?

NEW

SPDY protocol support is now available in NSURLSession on OS X Yosemite and iOS 8

- Available in Safari and for use in your apps
- Leveraged by other Apple frameworks (e.g., UIWebView)

SPDY is a protocol that attempts to make the web faster

Faster HTTP—SPDY

What is SPDY?

SPDY protocol support is now available in NSURLSession on OS X Yosemite and iOS 8

- Available in Safari and for use in your apps
- Leveraged by other Apple frameworks (e.g., UIWebView)

SPDY is a protocol that attempts to make the web faster

Serves as the base for the HTTP/2.0 draft specification

Faster HTTP—SPDY

What is SPDY?

NEW

SPDY protocol support is now available in NSURLSession on OS X Yosemite and iOS 8

- Available in Safari and for use in your apps
- Leveraged by other Apple frameworks (e.g., UIWebView)

SPDY is a protocol that attempts to make the web faster

Serves as the base for the HTTP/2.0 draft specification

Allows exchange of multiple HTTP messages simultaneously (and out-of-order) over a single TCP connection

Using SPDY

Using SPDY

Available on both OS X Yosemite and iOS 8

Using SPDY

Available on both OS X Yosemite and iOS 8

SPDY/2, SPDY/3, and SPDY/3.1 are supported

Using SPDY

Available on both OS X Yosemite and iOS 8

SPDY/2, SPDY/3, and SPDY/3.1 are supported

Supported transparently by NSURLSession

Using SPDY

Available on both OS X Yosemite and iOS 8

SPDY/2, SPDY/3, and SPDY/3.1 are supported

Supported transparently by NSURLSession

No source changes needed—it just works

Using SPDY

Available on both OS X Yosemite and iOS 8

SPDY/2, SPDY/3, and SPDY/3.1 are supported

Supported transparently by NSURLSession

No source changes needed—it just works

```
NSURL *url = [NSURL URLWithString:@“https://www.example.com/"];
NSURLSessionDataTask *task = [[NSURLSession sharedSession]
  dataTaskWithURL:url
  completionHandler:^(NSData *data, NSURLResponse *response, NSError *error)
  {...}];
[task resume];
```

SPDY Benefits

Single, long-lived, TCP connection

- Mitigates latency penalty for setting up new connections
- May reduce resource requirements on your server

SPDY Benefits

Single, long-lived, TCP connection

- Mitigates latency penalty for setting up new connections
- May reduce resource requirements on your server

Request/response multiplexing: no head-of-line blocking

- Head-of-Line Blocking: when a response blocks other responses from being received
- A large response (an image) might be less important than a small response (a CSS or JS file)

SPDY Benefits

Single, long-lived, TCP connection

- Mitigates latency penalty for setting up new connections
- May reduce resource requirements on your server

Request/response multiplexing: no head-of-line blocking

- Head-of-Line Blocking: when a response blocks other responses from being received
- A large response (an image) might be less important than a small response (a CSS or JS file)

Priorities

- The order requests are issued no longer impacts the order responses are received

SPDY Benefits

Multiplexing avoids Head-of-Line Blocking

Time →

HTTP/1.1: Head-of-Line Blocking

SPDY: Multiplexing

Priority Legend

Low
Medium
High

SPDY Benefits

Multiplexing avoids Head-of-Line Blocking

Time →

HTTP/1.1: Head-of-Line Blocking *(without Pipelining)*

SPDY: Multiplexing

Priority Legend

🟦 Low
🟩 Medium
🟧 High

SPDY Benefits

Multiplexing avoids Head-of-Line Blocking

Time ⟶

HTTP/1.1: Head-of-Line Blocking *(without Pipelining)*

image.jpg

styles.css

data.xml

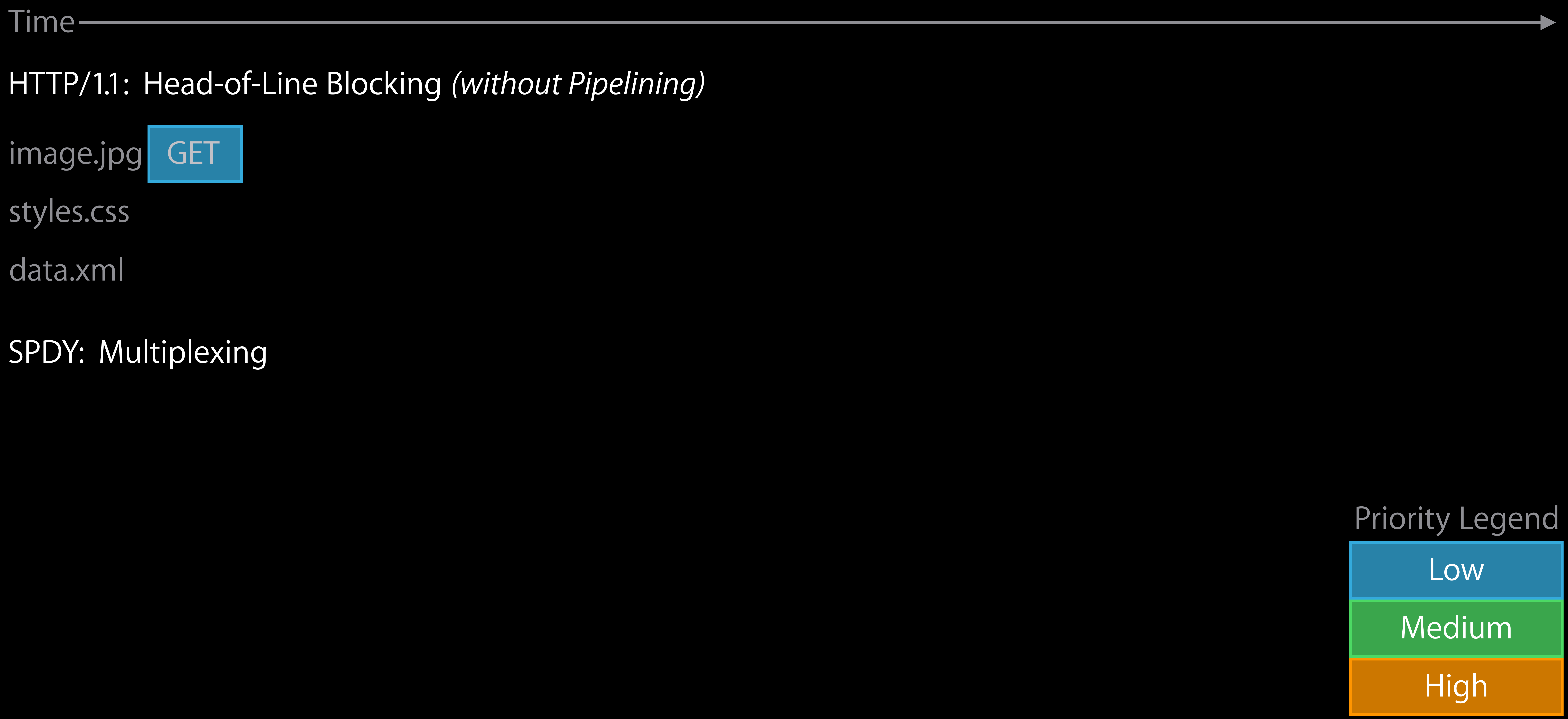
SPDY: Multiplexing

Priority Legend

🟦 Low
🟩 Medium
🟧 High

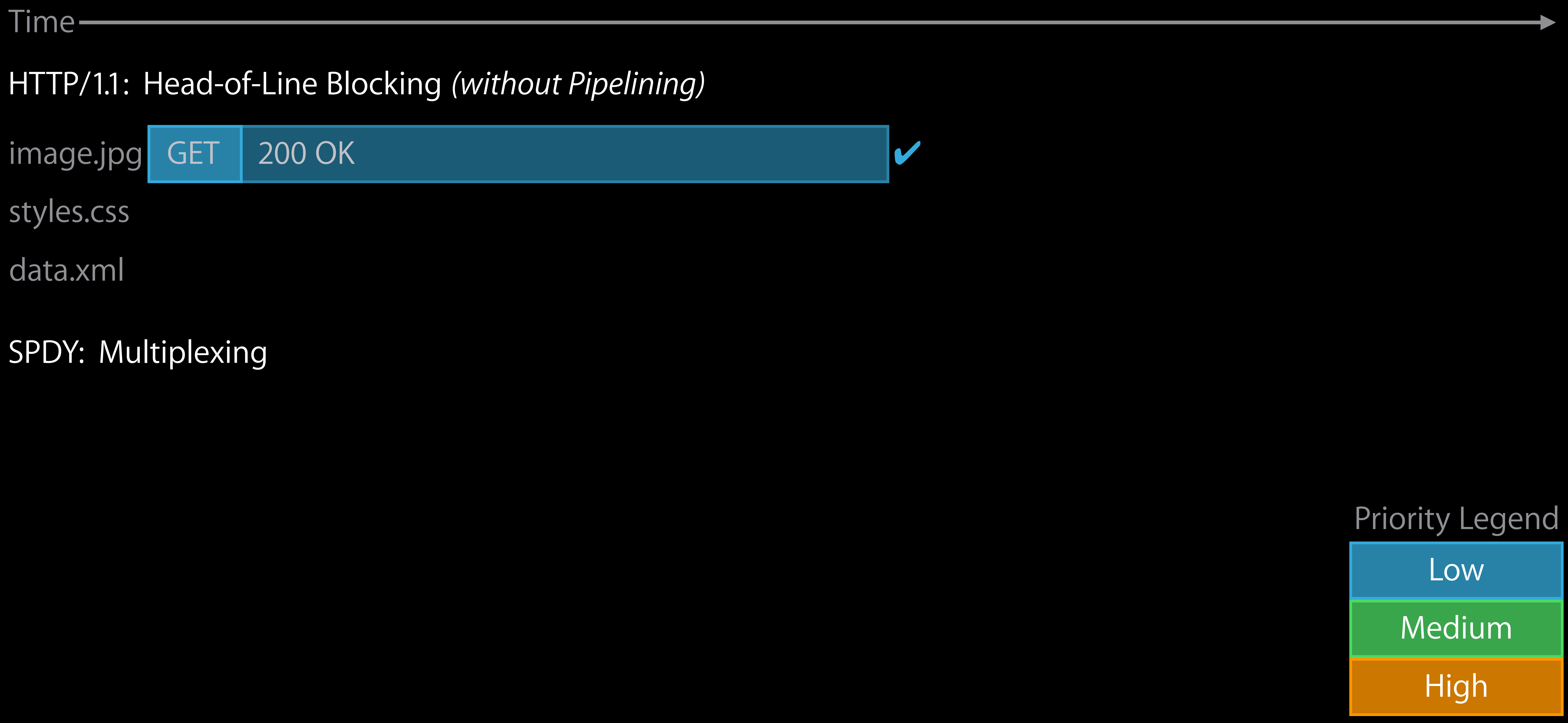
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



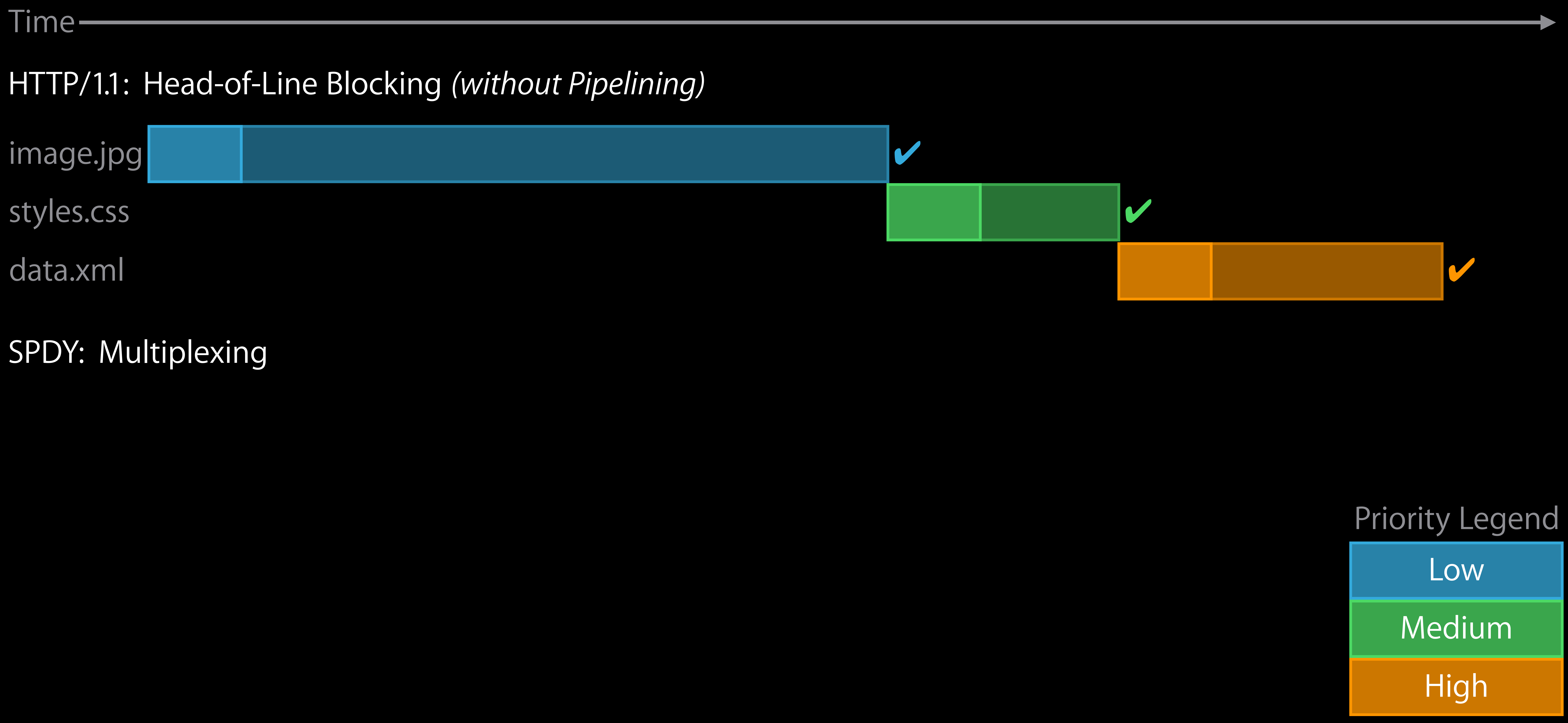
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



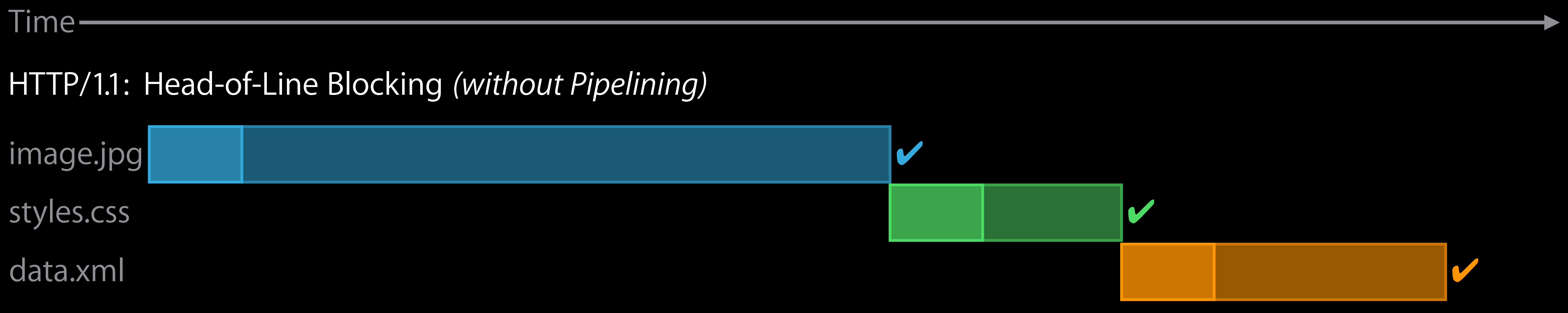
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



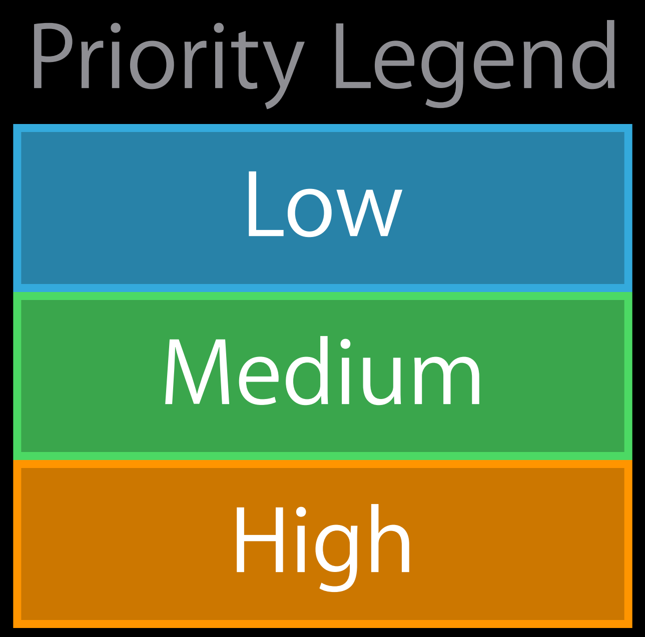
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



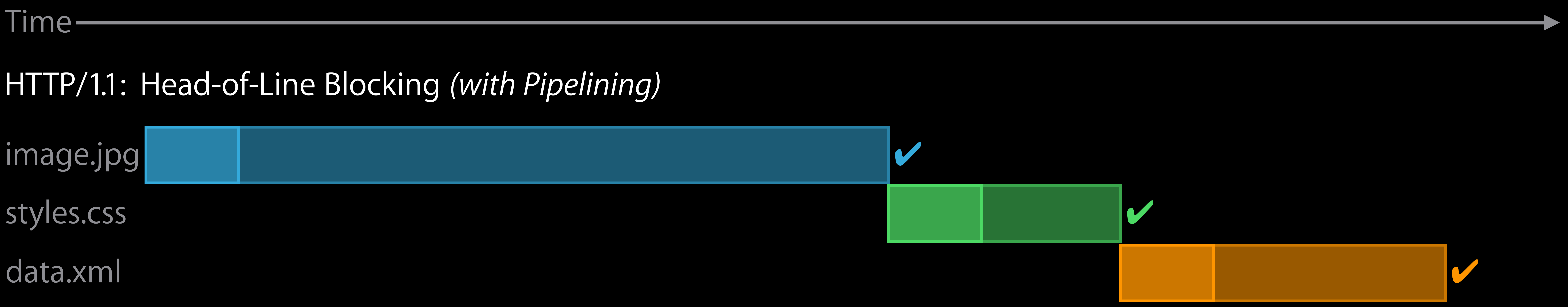
HTTP/1.1: Head-of-Line Blocking *(without Pipelining)*

SPDY: Multiplexing

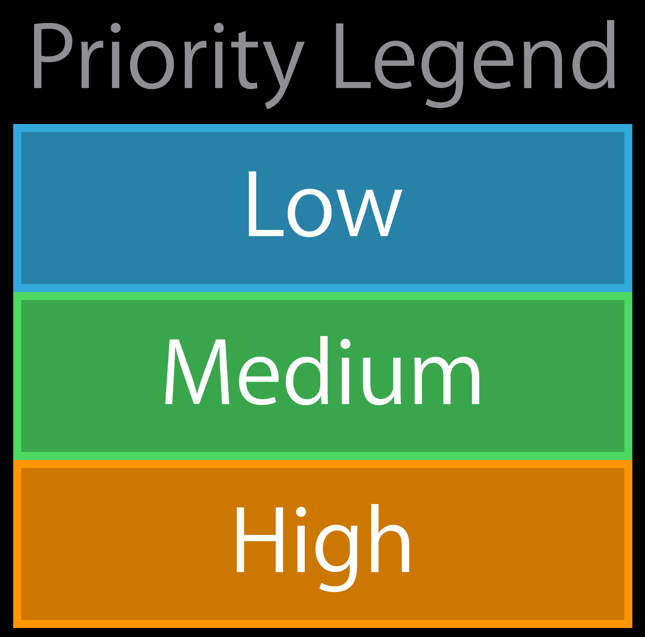


SPDY Benefits

Multiplexing avoids Head-of-Line Blocking

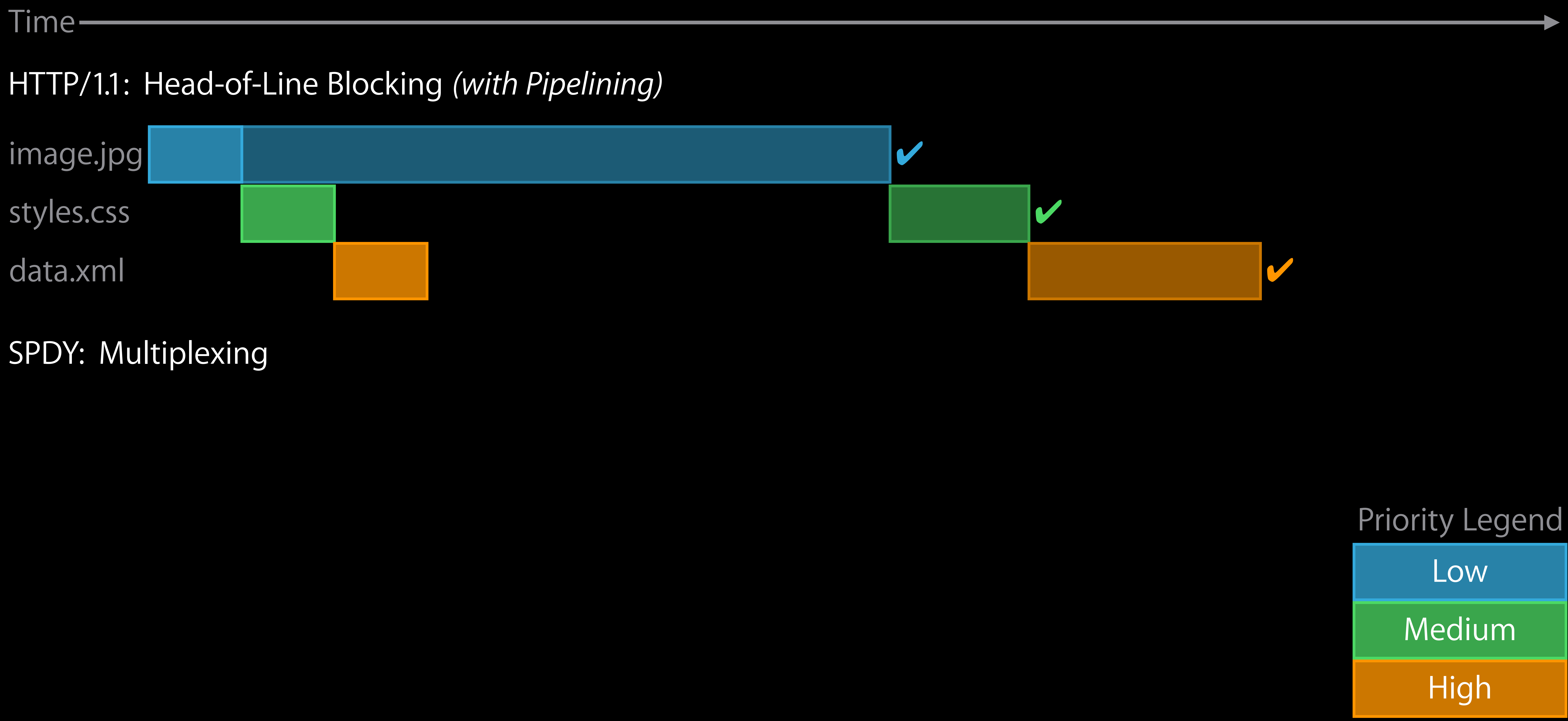


SPDY: Multiplexing



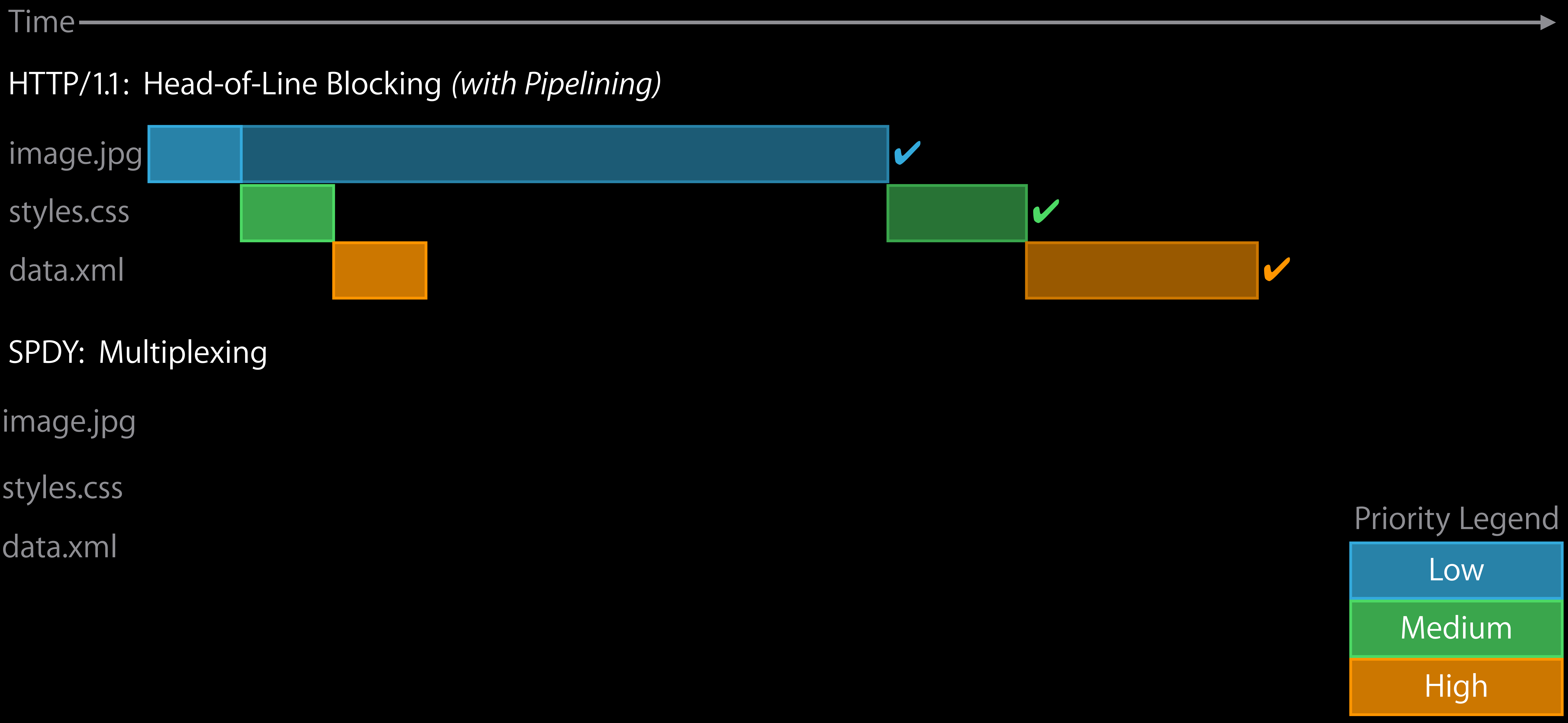
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



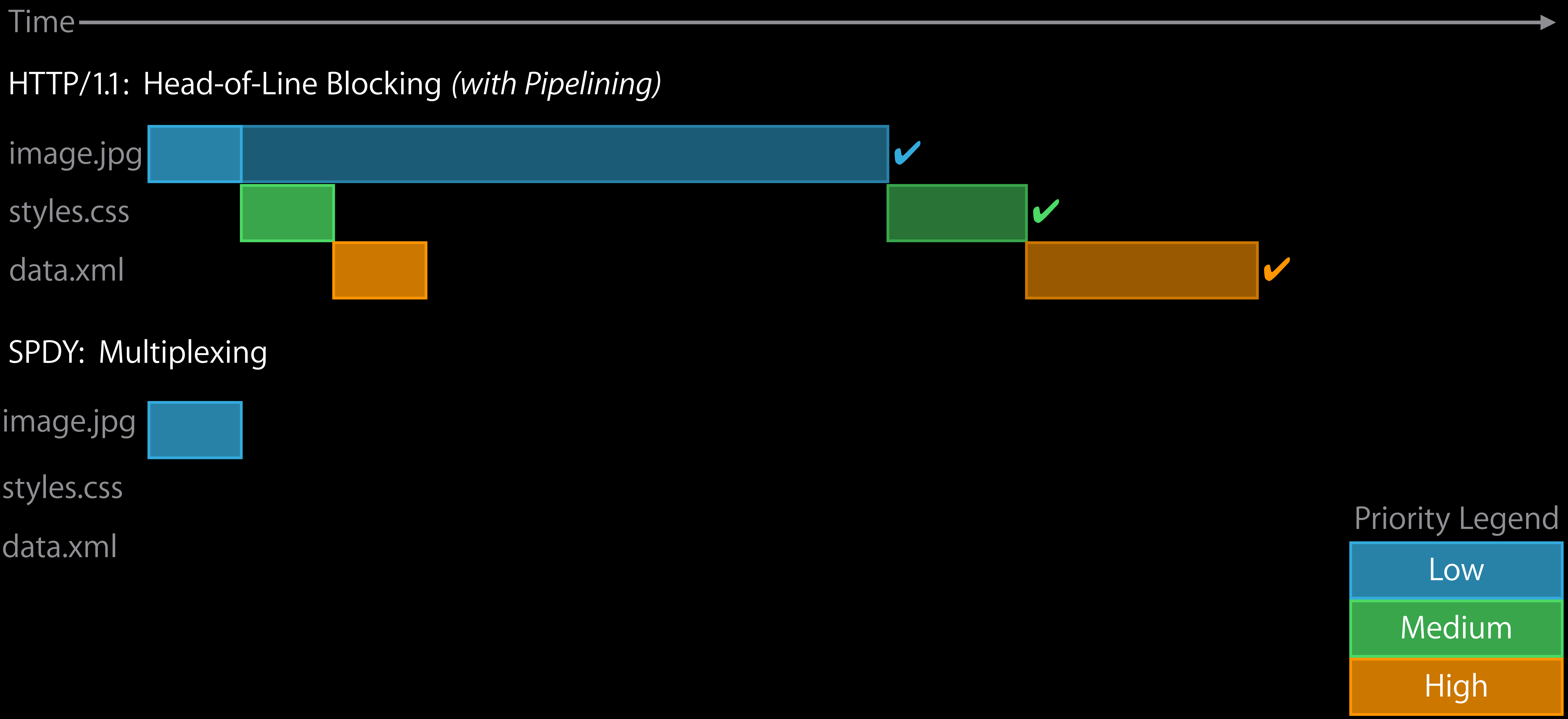
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



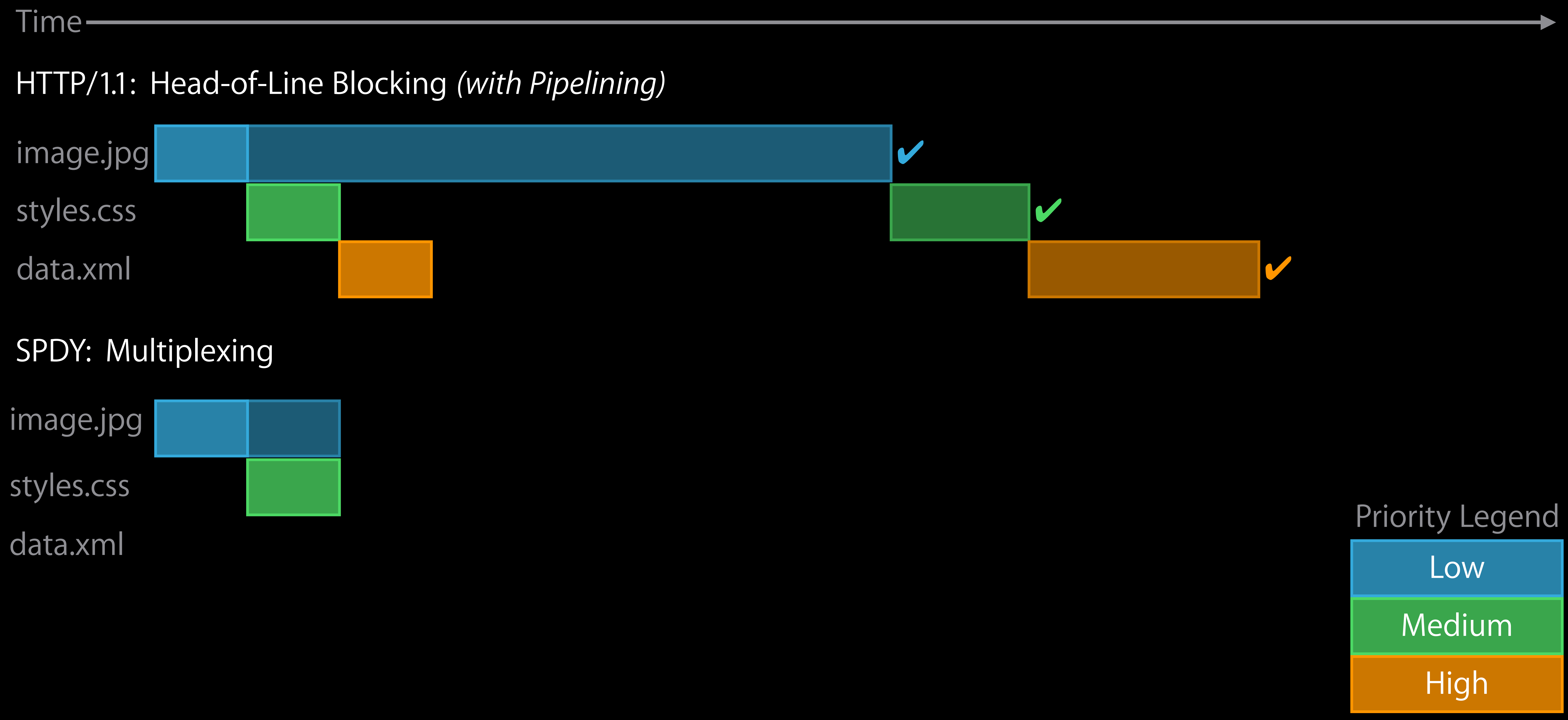
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



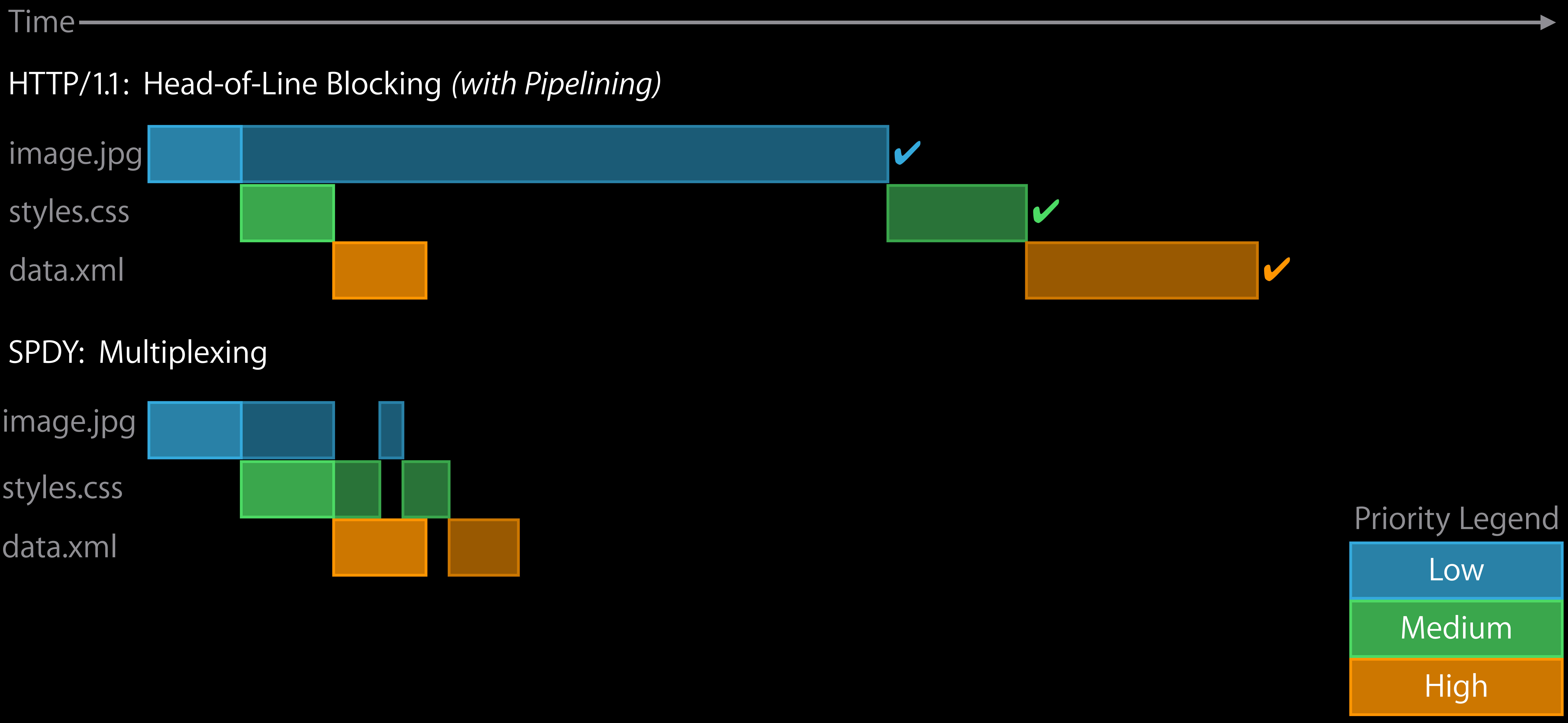
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



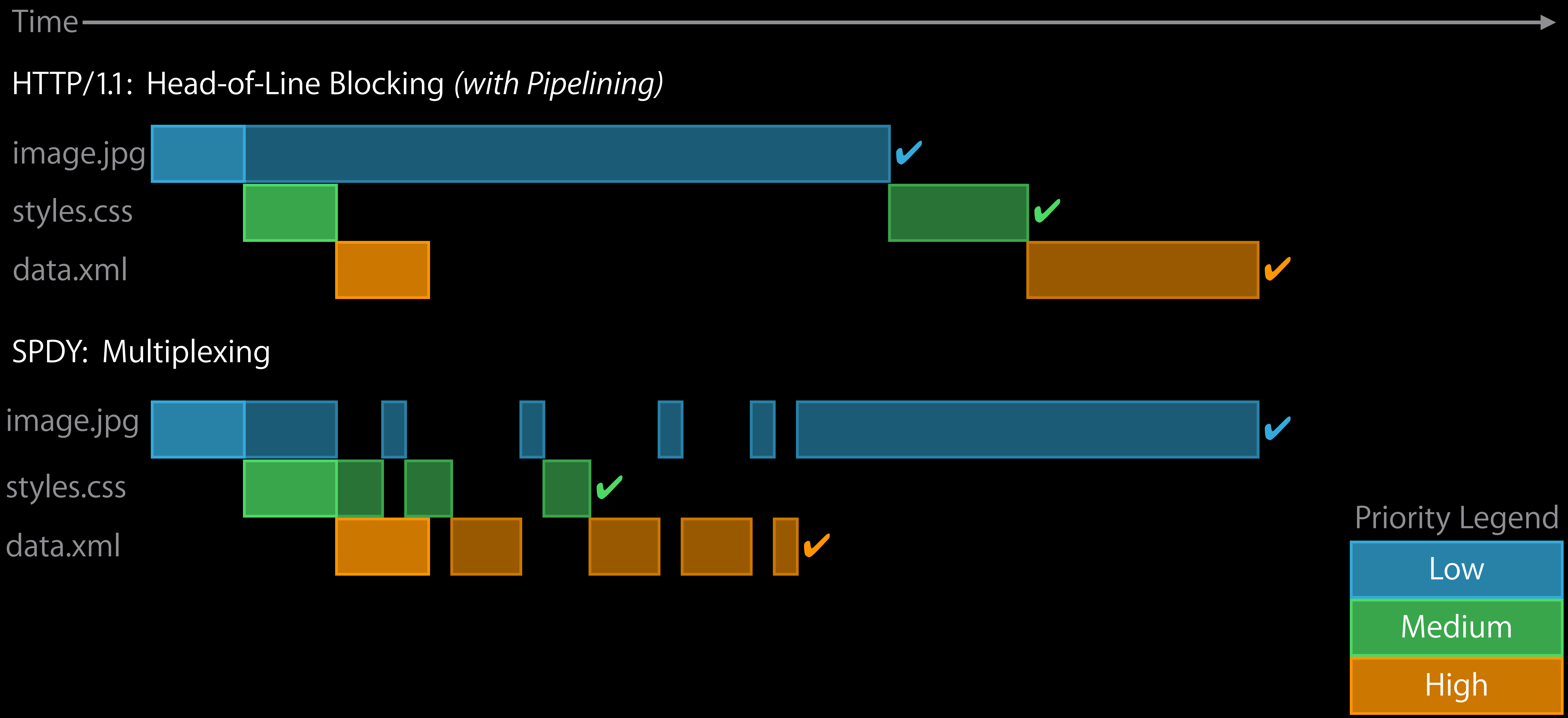
SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



SPDY Benefits

Multiplexing avoids Head-of-Line Blocking



Why Should I Adopt SPDY?

Can Give Better User Experience

- Reduced latency from long-lived connection is conducive to interactive behavior (especially over cellular)
- Our findings: SPDY is up to 25% faster than HTTP/1.1 in some cases

Why Should I Adopt SPDY?

Can Give Better User Experience

- Reduced latency from long-lived connection is conducive to interactive behavior (especially over cellular)
- Our findings: SPDY is up to 25% faster than HTTP/1.1 in some cases

Additional Benefits

- Reduced CPU use (one SSL handshake instead of several)
- May support more clients with same server-side infrastructure

Adopting SPDY

Adopting SPDY

SPDY does require server-side support

- Client negotiates with server during TLS handshake
- Uses https:// URLs
- Existing web server software and many CDNs already support SPDY

Adopting SPDY

SPDY does require server-side support

- Client negotiates with server during TLS handshake
- Uses https:// URLs
- Existing web server software and many CDNs already support SPDY

Will not interfere with your NSURLProtocol subclasses

- Apps might have their own implementation of SPDY or other protocols

SPDY Caveats

SPDY will not always outperform HTTP/1.1

- Parallel TCP connections (used by HTTP/1.1) can be faster than SPDY’s single connection
- Performance benefits are influenced by your workload

SPDY Caveats

SPDY will not always outperform HTTP/1.1

- Parallel TCP connections (used by HTTP/1.1) can be faster than SPDY’s single connection
- Performance benefits are influenced by your workload

SPDY compression of HTTP headers is disabled

- Susceptible to CRIME vulnerability
- Disabled by many SPDY implementations

SPDY Caveats

SPDY will not always outperform HTTP/1.1

- Parallel TCP connections (used by HTTP/1.1) can be faster than SPDY's single connection
- Performance benefits are influenced by your workload

SPDY compression of HTTP headers is disabled

- Susceptible to CRIME vulnerability
- Disabled by many SPDY implementations

SPDY is not an IETF-recognized standard, but paves the way for HTTP/2.0

SPDY Best Practices

SPDY Best Practices

Issue multiple requests to let multiplexing handle your workload; you no longer need to take steps to avoid head-of-line blocking

SPDY Best Practices

Issue multiple requests to let multiplexing handle your workload; you no longer need to take steps to avoid head-of-line blocking

Consolidate server hostnames

- Hostname sharding (e.g., css.apple.com, images.apple.com) causes multiple TCP connections to open
- Using a single hostname (and port!) for all requests allows for optimal connection sharing and re-use

Background Networking

Dan Vinegrad
Software Engineer

Background Sessions

Overview

Background Sessions

Overview

Why use background sessions?

Background Sessions

Overview

Why use background sessions?

Background sessions in app extensions

Background Sessions

Overview

- Why use background sessions?
- Background sessions in app extensions
- Discretionary networking

Background Sessions

Overview

Why use background sessions?

Background sessions in app extensions

Discretionary networking

Using background sessions properly

- Handling app launches
- Data tasks
- Pitfalls and best practices

Why Use Background Sessions?

Why Use Background Sessions?

Uploads and downloads continue while app isn’t running

- App can be suspended or crash
- App will be woken up to handle auth and completion

Why Use Background Sessions?

Uploads and downloads continue while app isn't running

- App can be suspended or crash
- App will be woken up to handle auth and completion

We monitor the environment for you

Why Use Background Sessions?

Uploads and downloads continue while app isn’t running

- App can be suspended or crash
- App will be woken up to handle auth and completion

We monitor the environment for you

- Network reachability and connectivity

Why Use Background Sessions?

Uploads and downloads continue while app isn’t running

- App can be suspended or crash
- App will be woken up to handle auth and completion

We monitor the environment for you

- Network reachability and connectivity
- Automatic retry after network failures

Why Use Background Sessions?

Uploads and downloads continue while app isn’t running

- App can be suspended or crash
- App will be woken up to handle auth and completion

We monitor the environment for you

- Network reachability and connectivity
- Automatic retry after network failures
- Battery monitoring

Why Use Background Sessions?

Uploads and downloads continue while app isn't running

- App can be suspended or crash
- App will be woken up to handle auth and completion

We monitor the environment for you

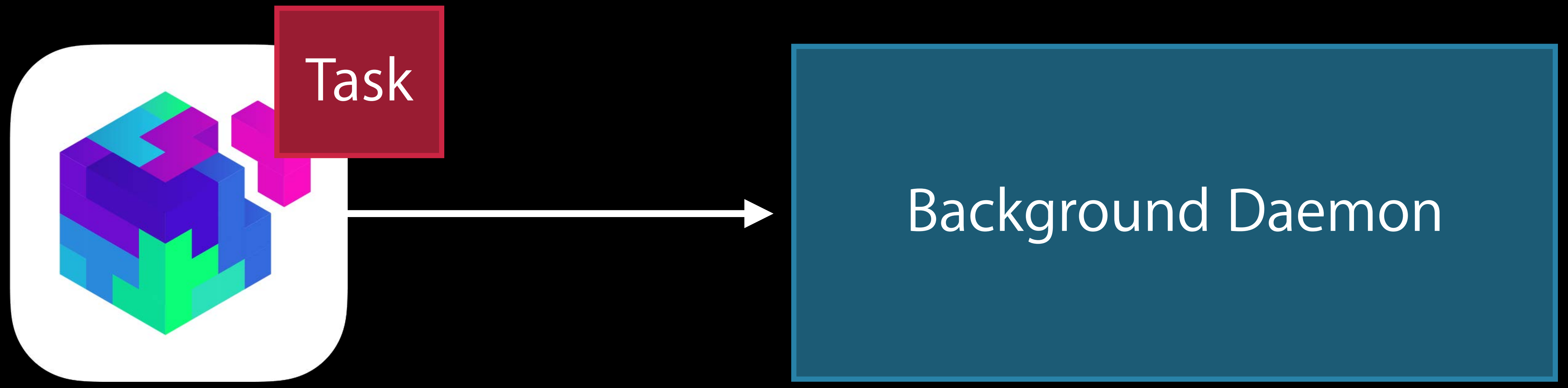
- Network reachability and connectivity
- Automatic retry after network failures
- Battery monitoring
- Bandwidth monitoring

Background Sessions in App Extensions

NEW

Extensions are short-lived processes

- In-process networking won't suffice for large uploads/downloads
- We will wake the containing app to handle events

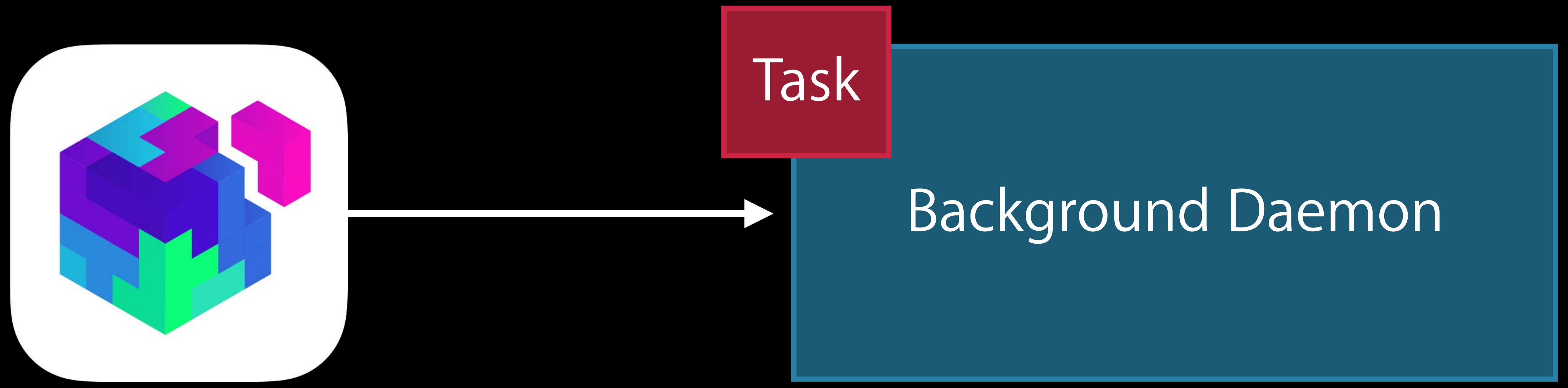


Background Sessions in App Extensions

NEW

Extensions are short-lived processes

- In-process networking won't suffice for large uploads/downloads
- We will wake the containing app to handle events

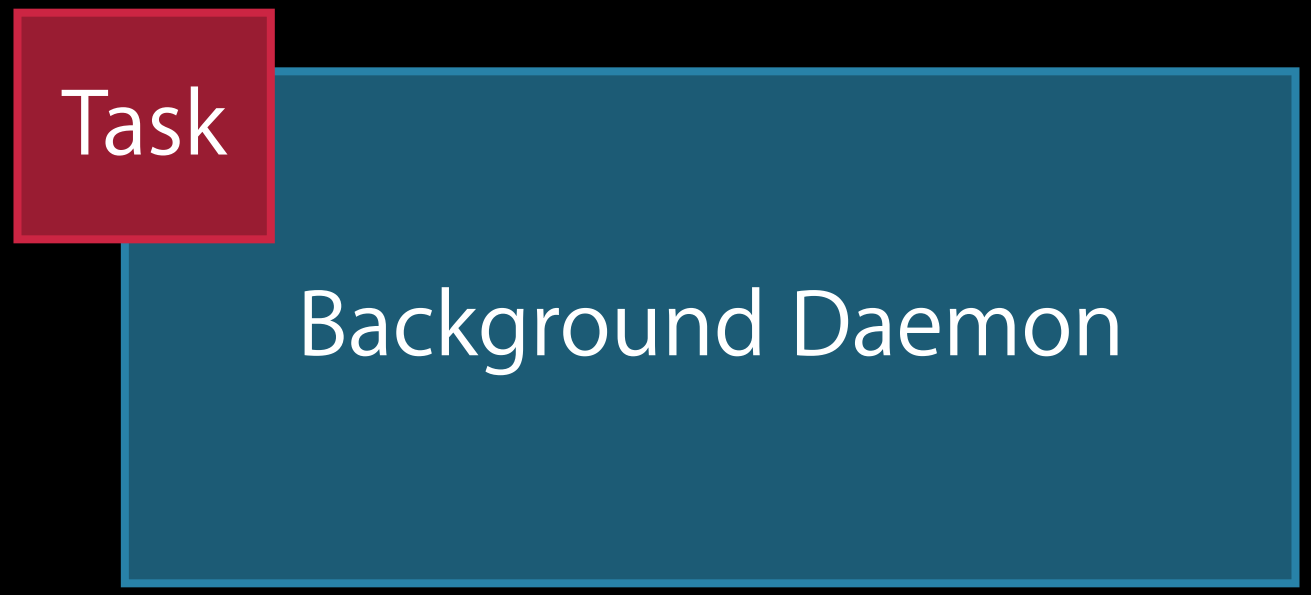


Background Sessions in App Extensions

NEW

Extensions are short-lived processes

- In-process networking won't suffice for large uploads/downloads
- We will wake the containing app to handle events

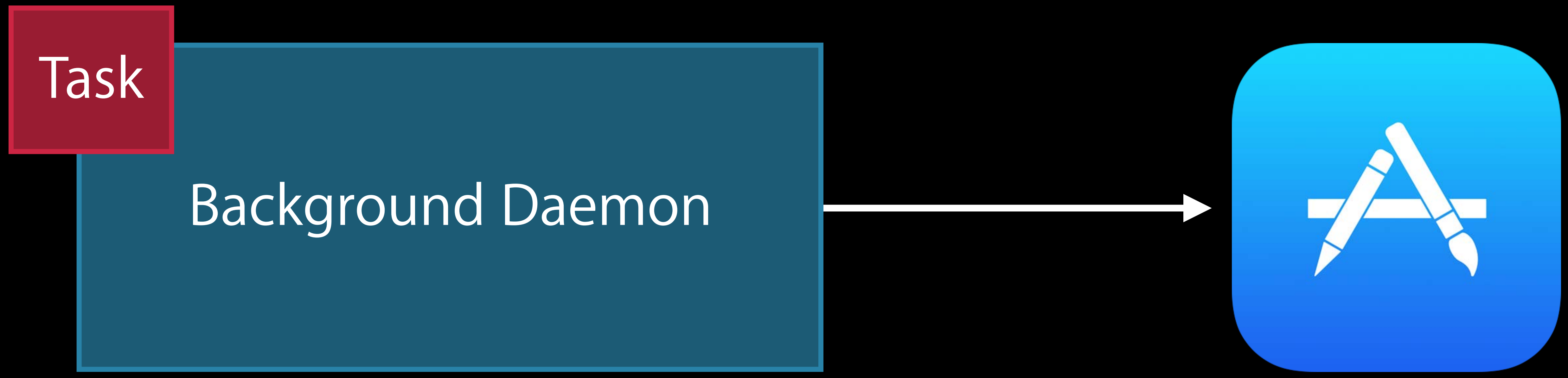


Background Sessions in App Extensions

NEW

Extensions are short-lived processes

- In-process networking won't suffice for large uploads/downloads
- We will wake the containing app to handle events



Background Sessions in App Extensions

Additional constraints

NEW

Background Sessions in App Extensions

Additional constraints

NEW

Must use a shared data container

- App and Extension won’t have access to same files

Background Sessions in App Extensions

Additional constraints

Must use a shared data container

- App and Extension won’t have access to same files

Only one process can be “connected” to the same background session at a time

- Can create a new session with a different identifier if another process is already connected

Background Sessions in App Extensions

Specifying a shared container

NEW

```
NSURLSessionConfiguration *config =
    [NSURLSessionConfiguration backgroundSessionConfigurationWithIdentifier:
                                @“com.mycompany.myapp.bgsession”];

config.sharedContainerIdentifier = @“com.mycompany.mysharedcontainer”;

NSURLSession *session = [NSURLSession sessionWithConfiguration:config
                                                       delegate:self
                                                  delegateQueue:nil];
```

Discretionary Networking

Discretionary Networking

Allows tasks to be scheduled by the system at a “good time”

- WiFi vs. cellular
- Battery considerations

Discretionary Networking

Allows tasks to be scheduled by the system at a "good time"

- WiFi vs. cellular
- Battery considerations

Takes into account how often app is launched

Discretionary Networking

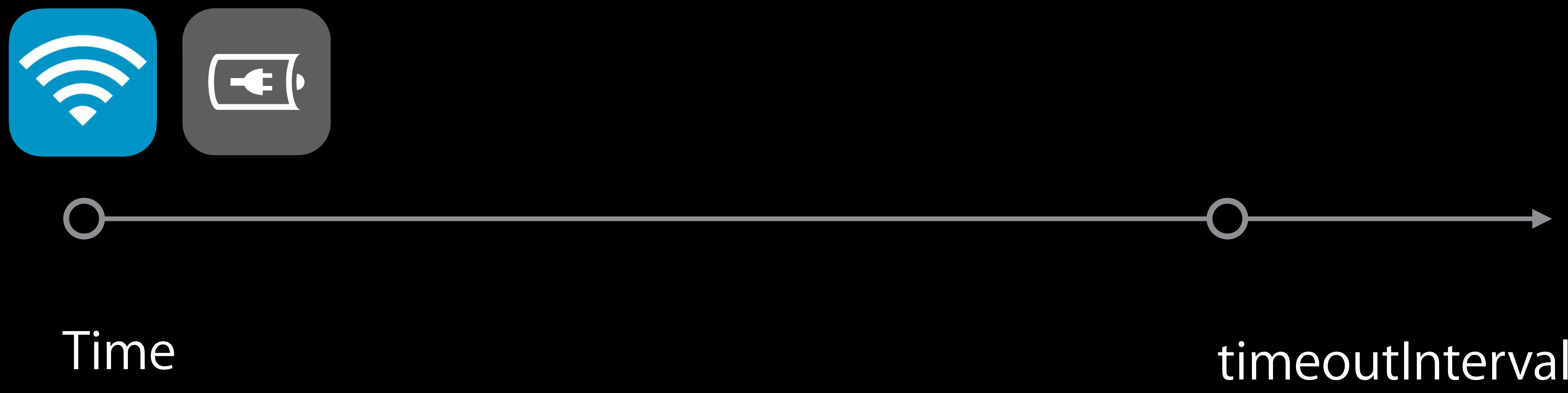
Tasks treated with more urgency as time goes on

- May restrict to WiFi and plugged in to power source at first
- Constraints will relax as timeoutIntervalForResource approaches

Discretionary Networking

Tasks treated with more urgency as time goes on

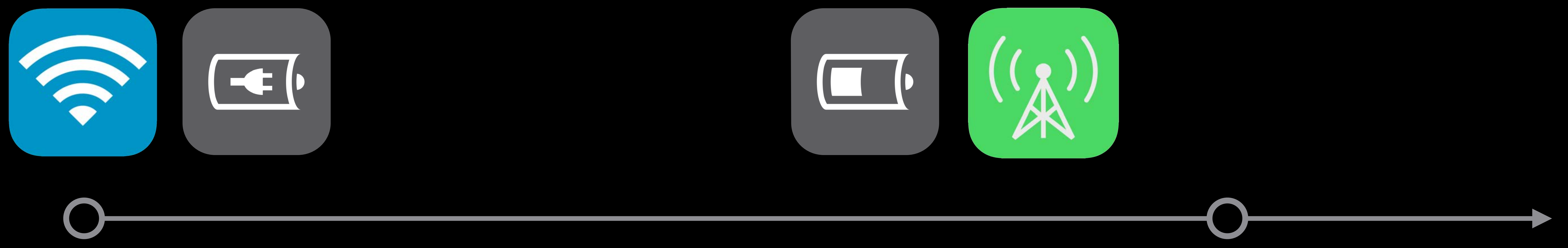
- May restrict to WiFi and plugged in to power source at first
- Constraints will relax as timeoutIntervalForResource approaches



Discretionary Networking

Tasks treated with more urgency as time goes on

- May restrict to WiFi and plugged in to power source at first
- Constraints will relax as timeoutIntervalForResource approaches



Time

timeoutIntervalForResource

Discretionary Networking

Discretionary Networking

Opt-in on NSURLSessionConfiguration:

```
config.discretionary = YES
```

Discretionary Networking

Opt-in on NSURLSessionConfiguration:

```
config.discretionary = YES
```

Non-user-initiated tasks

- Pre-fetching the next episode
- Upload syncing

Discretionary Networking

Opt-in on NSURLSessionConfiguration:

```
config.discretionary = YES
```

Non-user-initiated tasks

- Pre-fetching the next episode
- Upload syncing

Tasks created while app is running in the background are automatically discretionary

- Work performed during background fetch/push is not user-initiated
- Tasks become non-discretionary when user brings the app to the foreground

Using Background Sessions

Handling app launching

Using Background Sessions

Handling app launching

UIApplicationDelegate method:

```
–application:handleEventsForBackgroundURLSession:completionHandler:
```

Using Background Sessions

Handling app launching

UIApplicationDelegate method:

```
-application:handleEventsForBackgroundURLSession:completionHandler:
```

Reconnect to the background session

- Create background session with the provided identifier
- Receive delegate messages
- Call the completionHandler when finished handling the events

Using Background Sessions

Handling app launching

UIApplicationDelegate method:

```
–application:handleEventsForBackgroundURLSession:completionHandler:
```

Reconnect to the background session

- Create background session with the provided identifier
- Receive delegate messages
- Call the completionHandler when finished handling the events

```
–URLSessionDidFinishEventsForBackgroundURLSession:
```

Using Background Sessions

Data tasks

NEW

Using Background Sessions

Data tasks

Will only run while app is running

- Will fail with `NSURLErrorBackgroundSessionWasDisconnected` when app is suspended or exits

Using Background Sessions

Data tasks

Will only run while app is running

- Will fail with `NSURLErrorBackgroundSessionWasDisconnected` when app is suspended or exits

Can convert to download task when response is received

- Will continue after app is suspended

```
- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask
*)dataTask didReceiveResponse:(NSURLResponse *)response completionHandler:
(void (^)(NSURLSessionResponseDisposition disposition))completionHandler;
NSURLSessionResponseBecomeDownload
```

Using Background Sessions

Things to avoid

Using Background Sessions

Things to avoid

Creating one task at a time

- Tasks created in background will be discretionary
- The system will prevent your app from being launched too frequently

Using Background Sessions

Things to avoid

Creating one task at a time

- Tasks created in background will be discretionary
- The system will prevent your app from being launched too frequently

Downloading lots of small assets

- Much more efficient to download one large, zipped asset

Using Background Sessions

Things to avoid

Creating one task at a time

- Tasks created in background will be discretionary
- The system will prevent your app from being launched too frequently

Downloading lots of small assets

- Much more efficient to download one large, zipped asset

Blocking while waiting for transfers to complete

Using Background Sessions

Best practices

Using Background Sessions

Best practices

Can still use in-process networking while running in the background

- Smaller, time-sensitive assets
- Larger uploads/downloads should use background sessions

Using Background Sessions

Best practices

Can still use in-process networking while running in the background

- Smaller, time-sensitive assets
- Larger uploads/downloads should use background sessions

Support resumable downloads (Range GET requests)

Using Background Sessions

Best practices

Can still use in-process networking while running in the background

- Smaller, time-sensitive assets
- Larger uploads/downloads should use background sessions

Support resumable downloads (Range GET requests)

Handle launch events properly

- Reconnect to your background session when we launch your app
- Call the completion handler

Summary

Summary

New APIs on NSStream and NSNetService

Summary

New APIs on NSStream and NSNetService

Using NSURLSession

Summary

New APIs on NSStream and NSNetService

Using NSURLSession

New protocol support

Summary

New APIs on NSStream and NSNetService

Using NSURLSession

New protocol support

Background networking

More Information

Paul Danbold
Core OS Technologies Evangelist
danbold@apple.com

Documentation
NSURLSession Class Reference
https://developer.apple.com/library/ios/documentation/Foundation/Reference/NSURLSession_class/Introduction/Introduction.html

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

Related Sessions

● Creating Extensions for iOS and OS X, Part 1	Mission	Tuesday 2:00PM
● Creating Extensions for iOS and OS X, Part 2	Mission	Wednesday 11:30AM
● Cross Platform Nearby Networking	Nob Hill	Wednesday 9:00AM
● Fix Bugs Faster Using Activity Tracing	Russian Hill	Thursday 11:30AM
● Power, Performance, and Diagnostics: What's New in GCD and XPC	Russian Hill	Thursday 2:00PM

Labs

●	Networking Lab	Core OS Lab A	Tuesday 4:30PM
●	Networking Lab	Core OS Lab B	Wednesday 9:00AM
●	Multipeer Connectivity Lab	Core OS Lab A	Wednesday 10:15AM
●	Multipeer Connectivity Lab	Core OS Lab B	Friday 9:00AM
●	Extensions Lab	Frameworks Lab A	Tuesday 3:15PM
●	Extensions Lab	Frameworks Lab B	Thursday 2:00PM

WWDC14