What’s New with Multitasking

Keep app content fresh and interesting

Session 204

David Chan
iOS Software Engineering

Today

- Multitasking in iOS 6
  - Changes in iOS 7
- New Multitasking APIs
- Running in the background
  - Data Protection
  - Battery life
  - Cellular data usage

Multitasking in iOS 6

- Background Task Completion
- Background Audio
- Location Services
  - Region Monitoring
  - Significant Location Changes
  - Continuous Location Monitoring
- VoIP
- Newsstand

Changes to Existing Multitasking

- Background Task Completion
- App Switcher
- Location Services
- Newsstand

Background Task Changes

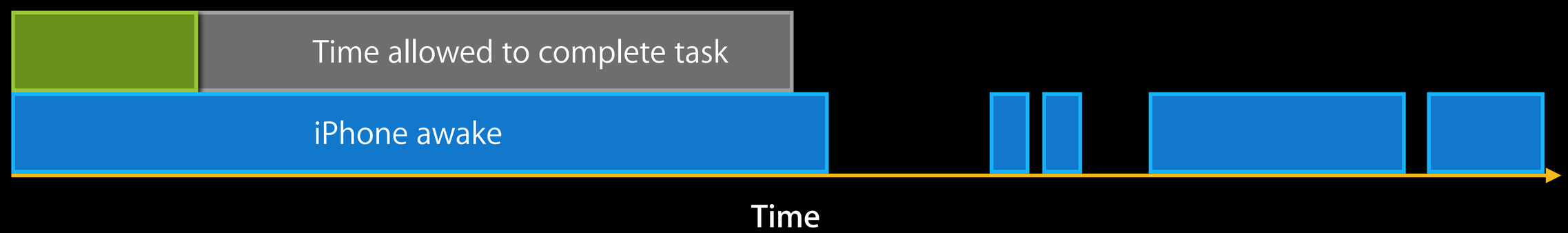
In iOS 6


```
task = [app beginBackgroundTaskWithExpirationHandler:^{
    task = UIBackgroundTaskInvalid;
};
// ...
[app endBackgroundTask:task];
```

- Used for
  - Encoding video
  - Uploads or downloads
  - Completing database operations

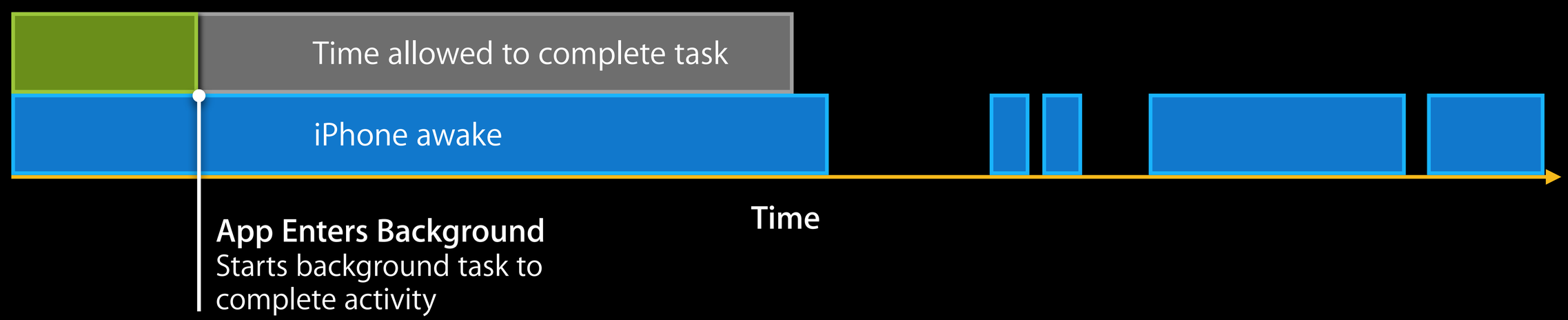
Background Task Changes

In iOS 6



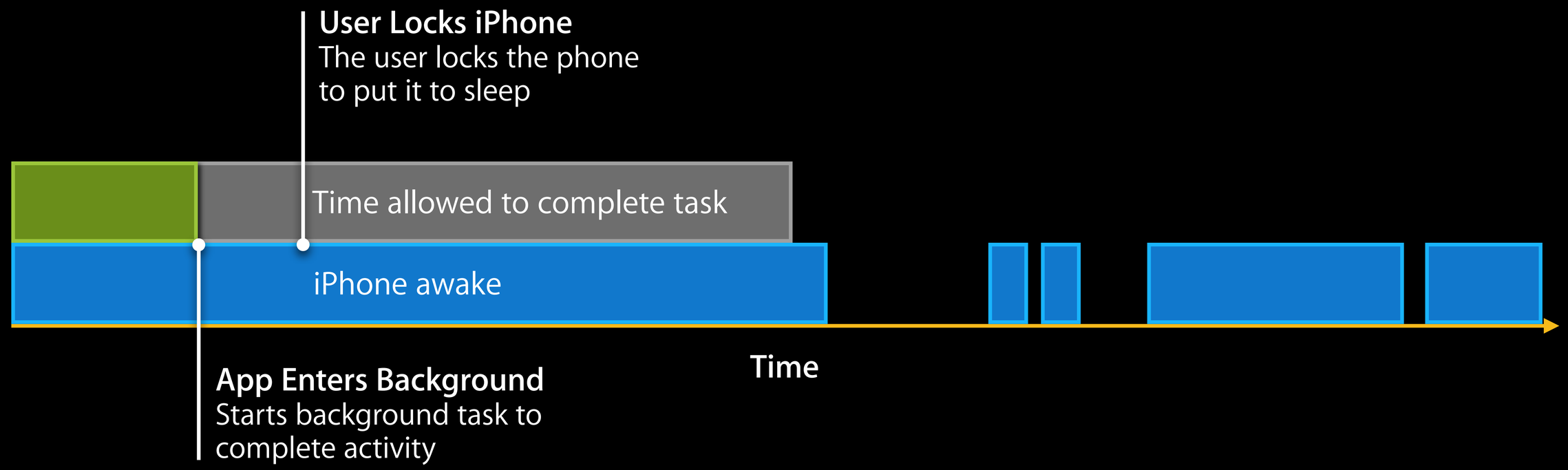
Background Task Changes

In iOS 6



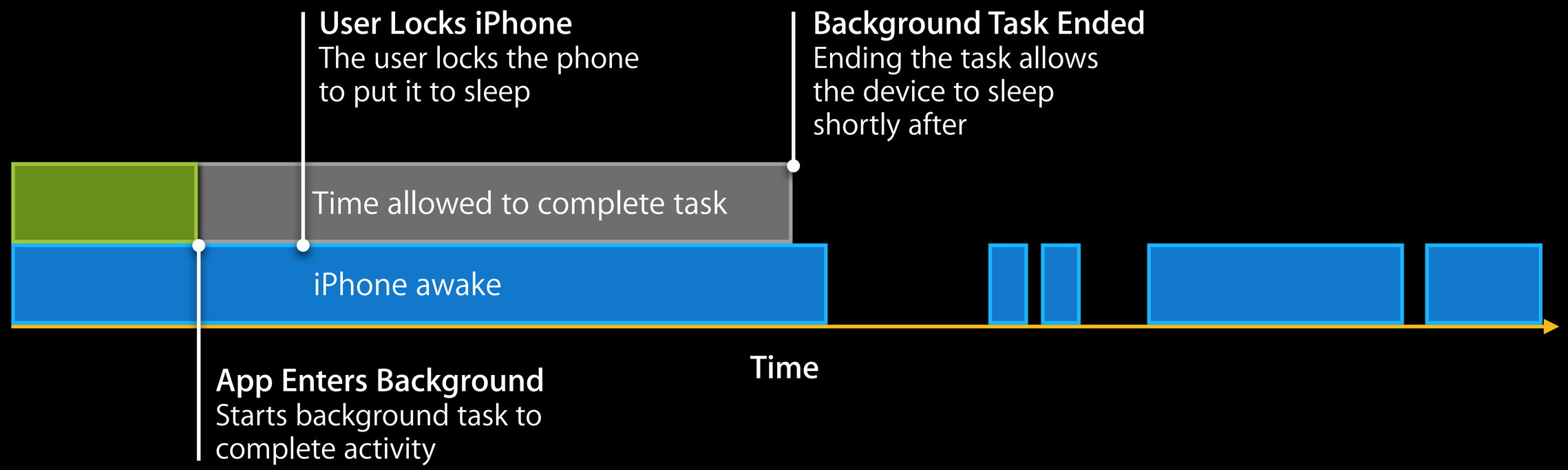
Background Task Changes

In iOS 6



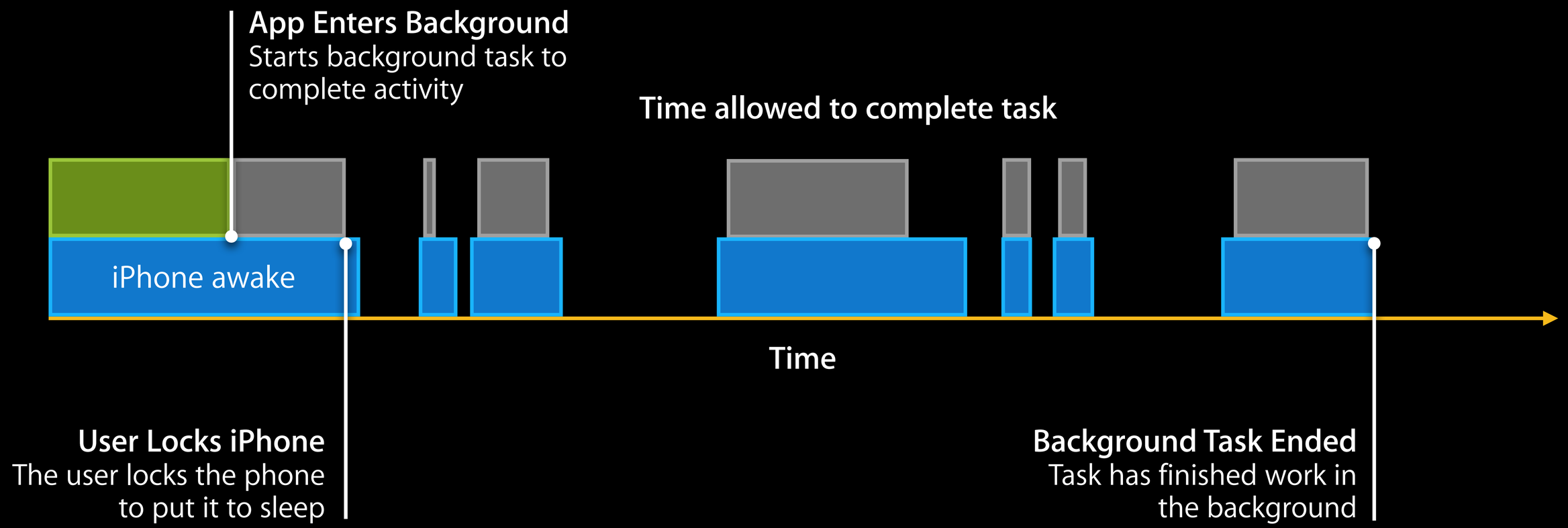
Background Task Changes

In iOS 6



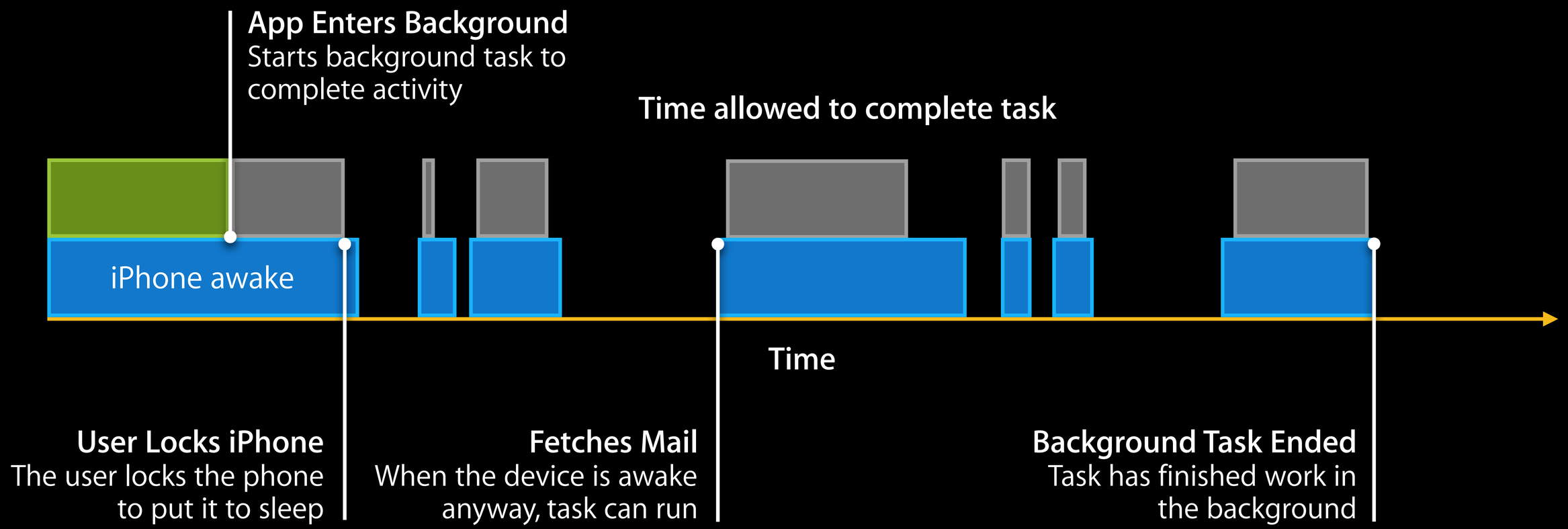
Background Task Changes

In iOS 7



Background Task Changes

In iOS 7



Background Task Changes

Handling iOS 6 vs iOS 7

- If you were using background tasks for network transfers, use `NSURLSession` instead
- Switch between old and new mechanisms like this

```
if ([NSURLSession class]) {
   // Create a background session and enqueue transfers
}
else {
  // Start a background task and transfer directly
}
```

Background Task Changes

In iOS 7

- Apps will no longer keep the device awake
- Apps will still get several minutes of runtime
- Just not guaranteed to be contiguous

App Switcher

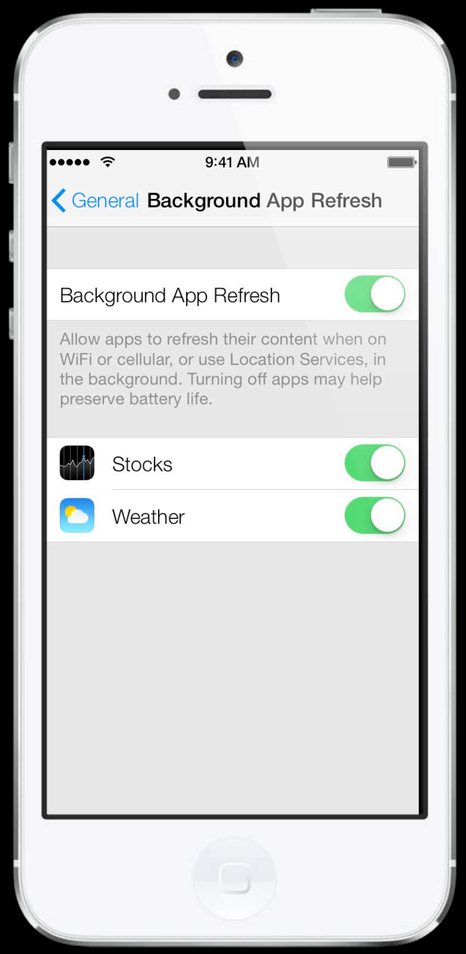
Just click home twice

- New UI prominently features app snapshots
- Make sure your app looks good after the user leaves and comes back
  - State Restoration
- Swipe up to remove apps
  - Stop running in the background
- Updating snapshots in the background

Location Services

Changes from iOS 6

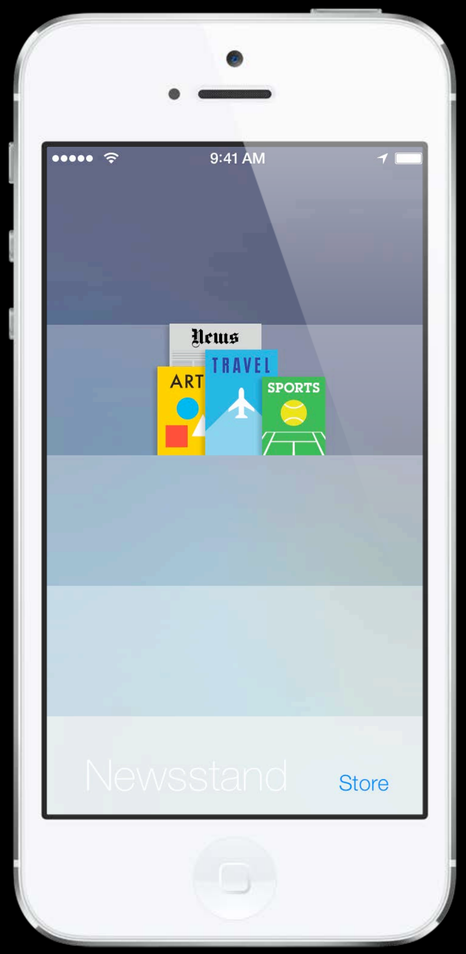
- Background activity configurable from Settings
- Respects App Switcher
  - Won’t launch in the background if user removed app from App Switcher



Newsstand

Changes from iOS 6

- Background activity configurable from Settings
- Respects App Switcher
  - Won't launch in the background if user removed app from App Switcher
- Stick with Newsstand API



New Multitasking APIs

New Multitasking APIs

- Background Fetch

New Multitasking APIs

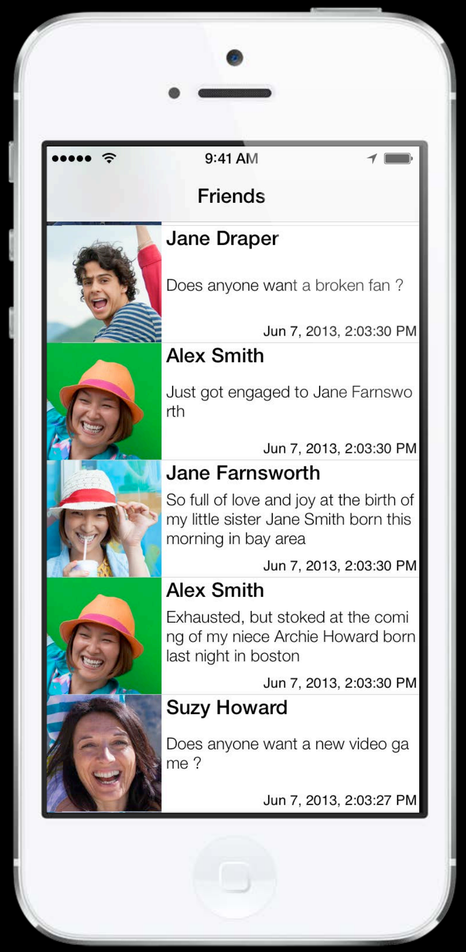
- Background Fetch
- Remote Notifications

New Multitasking APIs

- Background Fetch
- Remote Notifications
- Background Transfer Service

Background Fetch

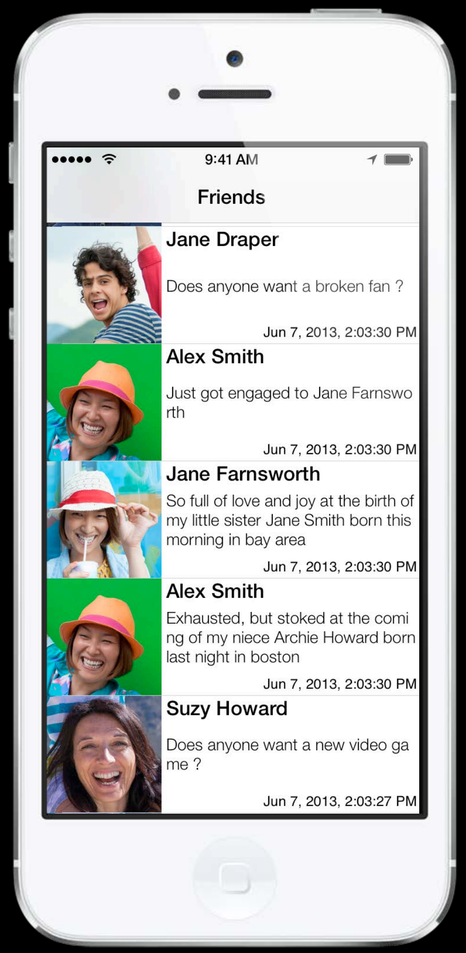
Motivation



Background Fetch

Motivation

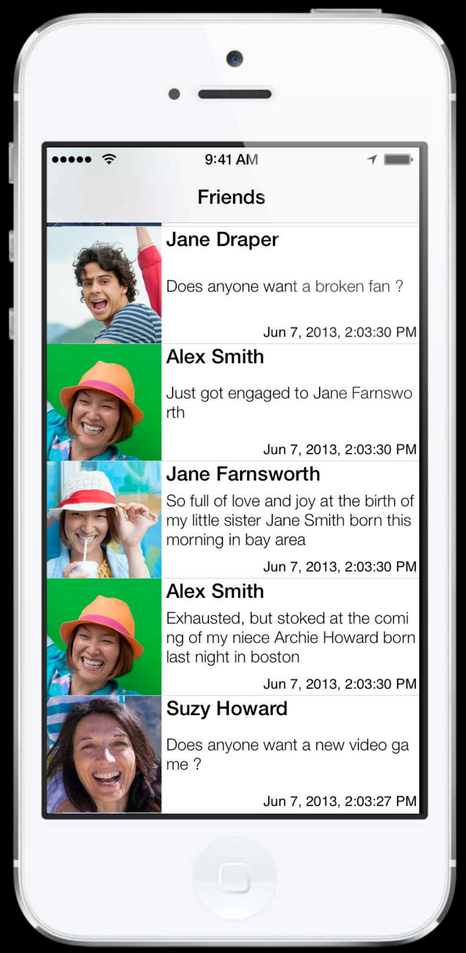
- Let's say you have the next great social networking app



Background Fetch

Motivation

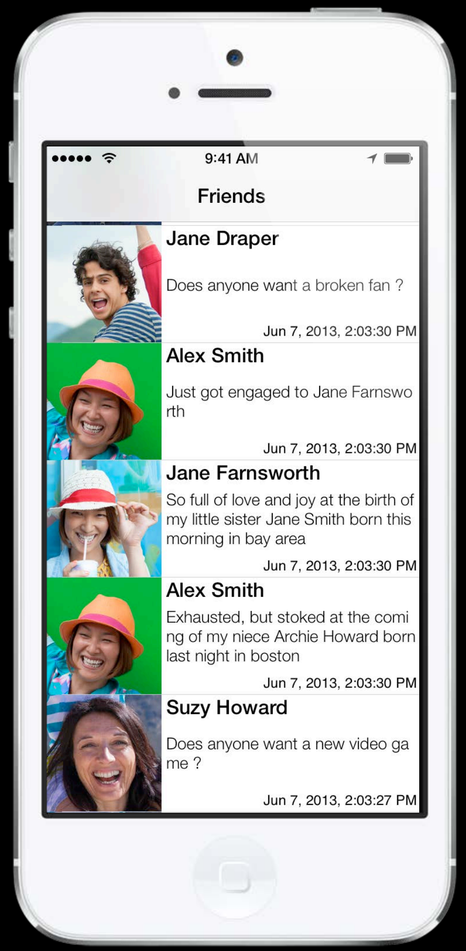
- Let's say you have the next great social networking app
- When your app becomes frontmost, you refresh your feed



Background Fetch

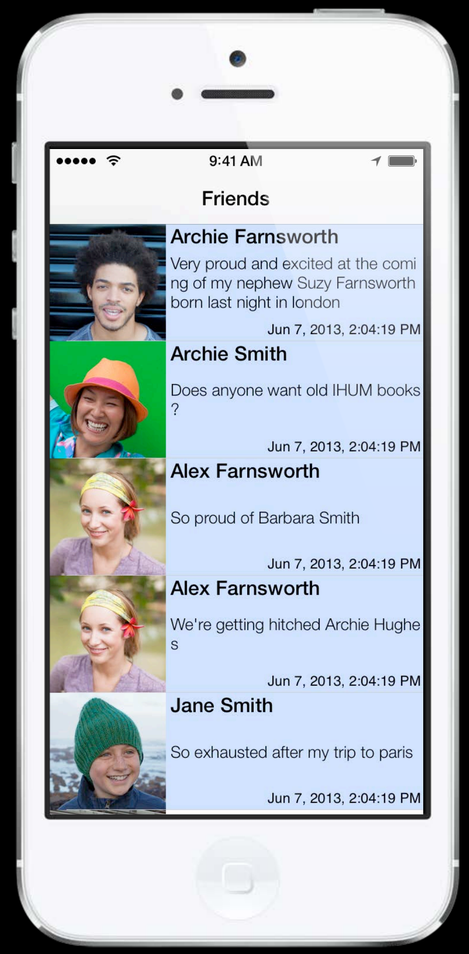
Motivation

- Let's say you have the next great social networking app
- When your app becomes frontmost, you refresh your feed
- Every time your users return to your app, they have to wait for new and interesting stuff



Background Fetch

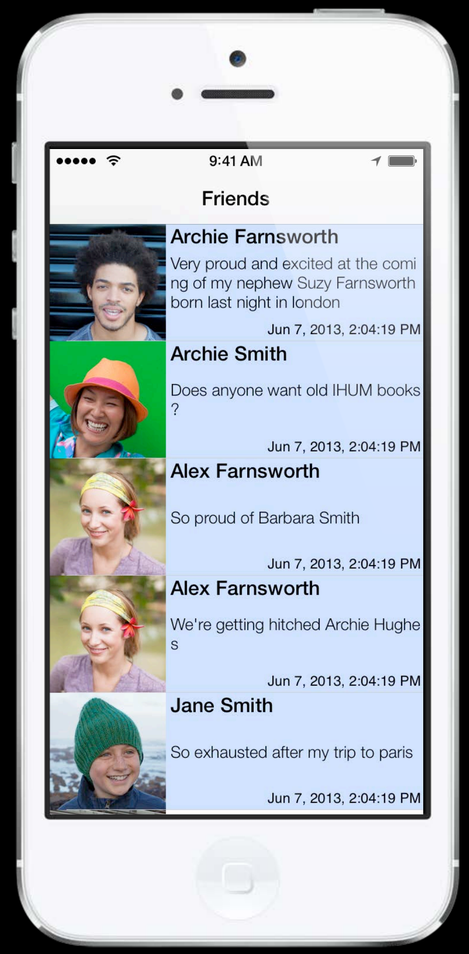
Motivation



Background Fetch

Motivation

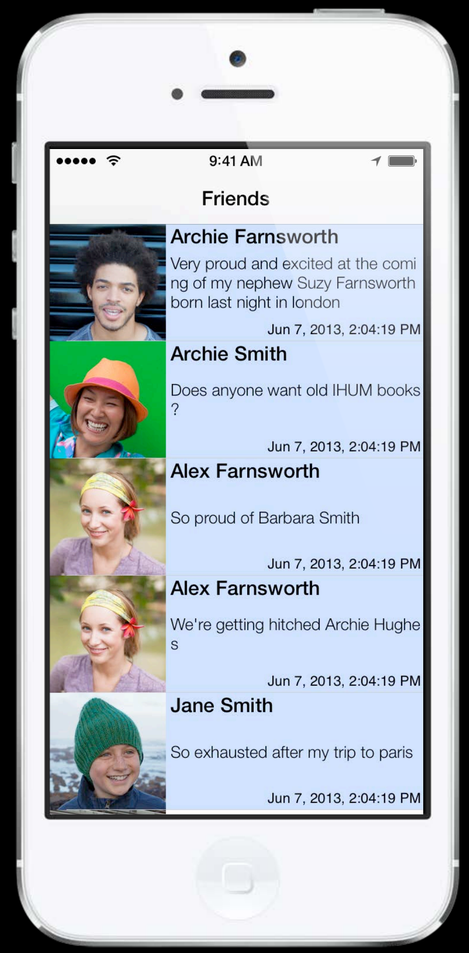
- Now, you can update your content before the user returns to your app



Background Fetch

Motivation

- Now, you can update your content before the user returns to your app
- So that the new and interesting content waits for your users to see



Background Fetch

UIKit API

Background Fetch

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	fetch
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

Background Fetch

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	fetch
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Enable fetching

```
[app setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum]
```

Background Fetch

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	fetch
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Enable fetching

```
[app setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum]
```

3. Launched into background

Background Fetch

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	fetch
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Enable fetching

```
[app setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum]
```

3. Launched into background

Background Fetch

UIKit API

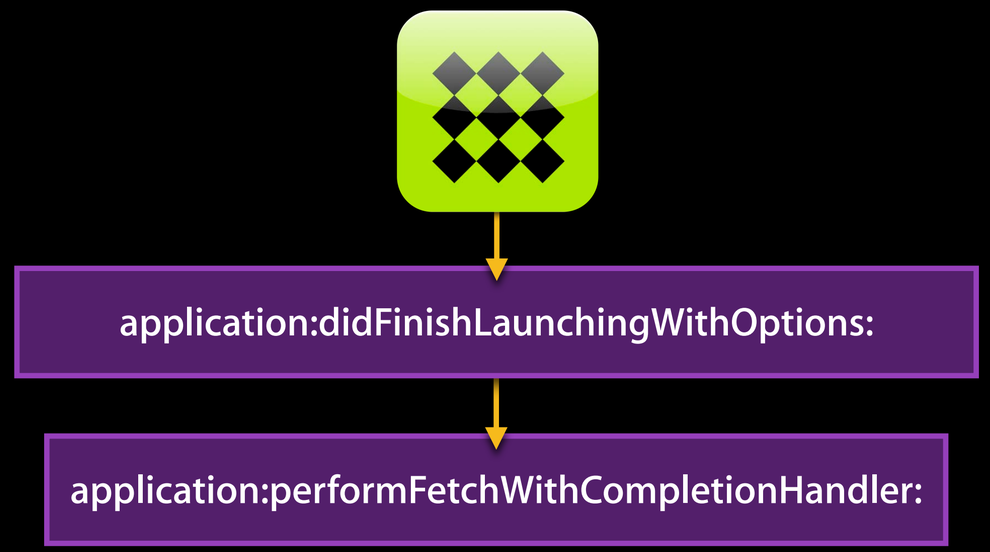
1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	fetch
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Enable fetching

```
[app setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum]
```

3. Launched into background



Background Fetch

UIKit API

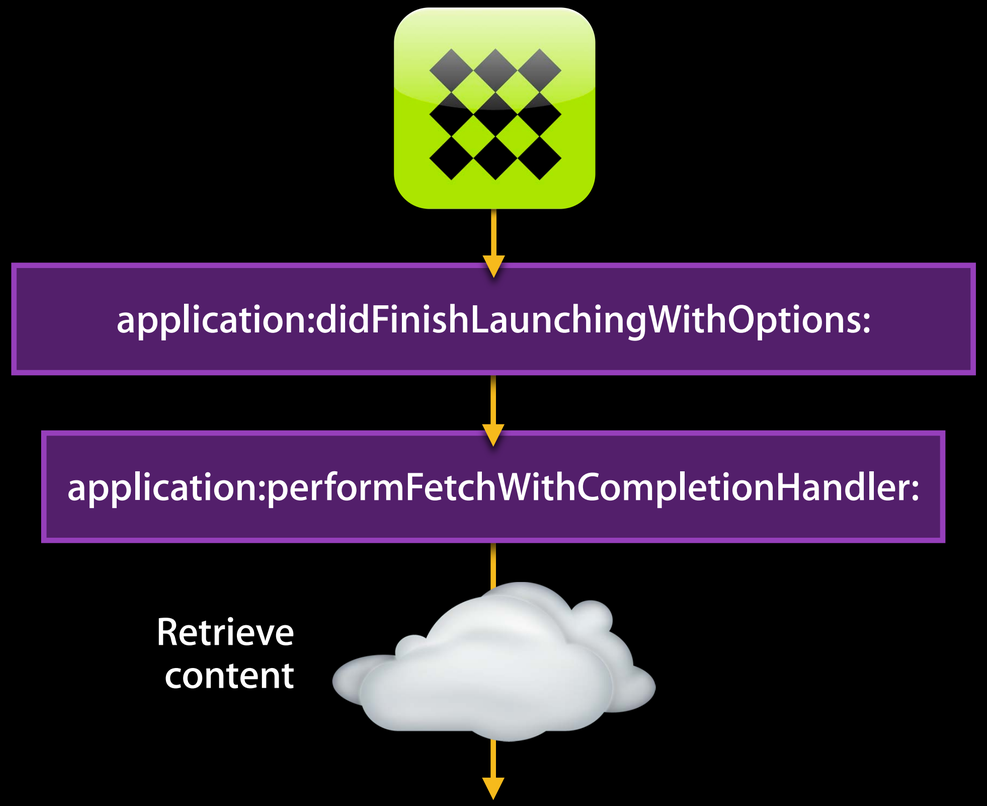
1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	fetch
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Enable fetching

```
[app setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum]
```

3. Launched into background



Background Fetch

UIKit API

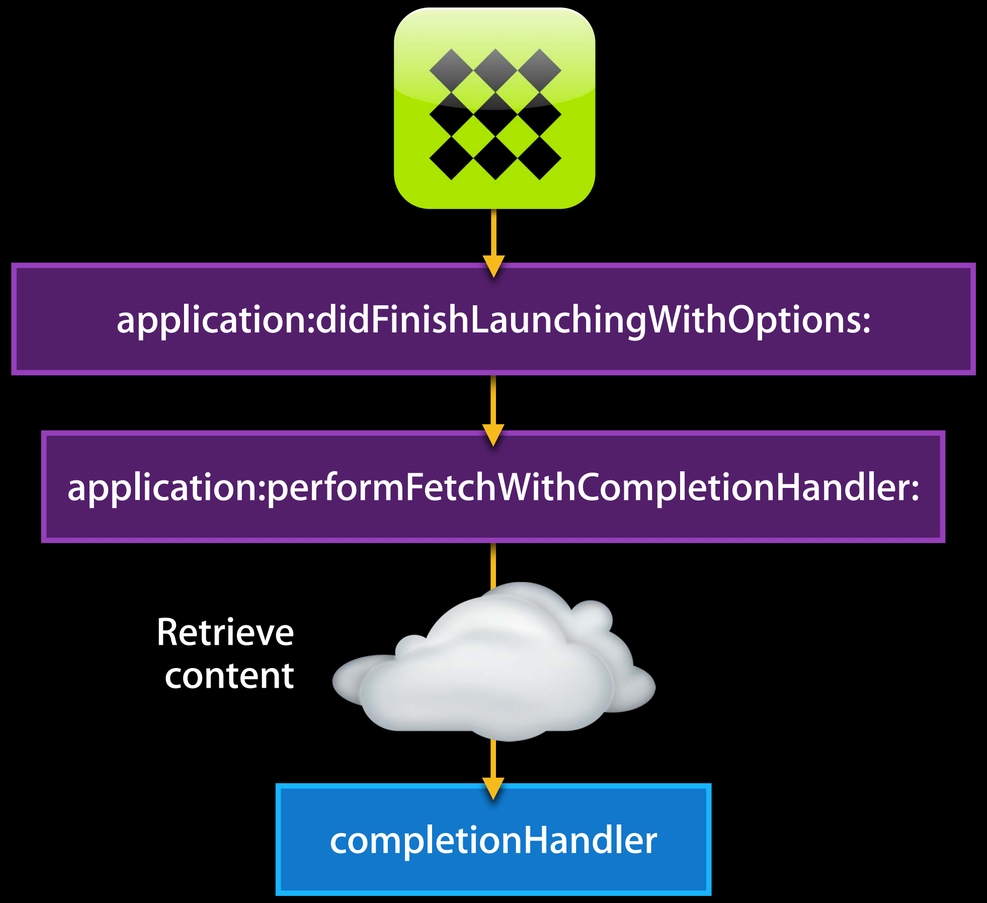
1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	fetch
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Enable fetching

```
[app setMinimumBackgroundFetchInterval:
UIApplicationBackgroundFetchIntervalMinimum]
```

3. Launched into background



Background Fetch

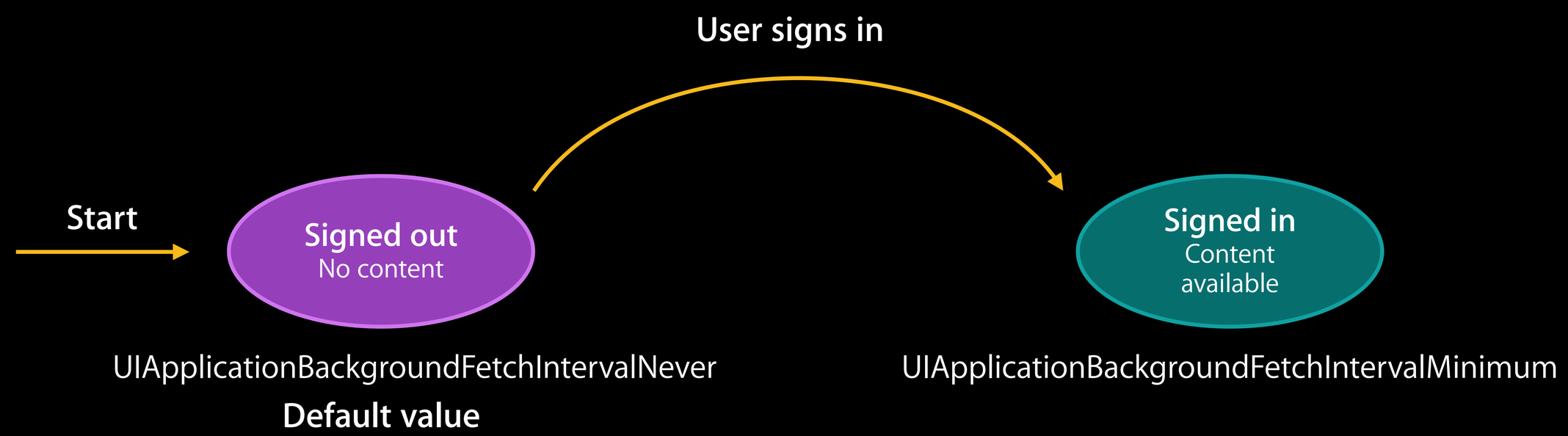
Minimum Background Fetch Interval

UIApplicationBackgroundFetchIntervalNever

Default value

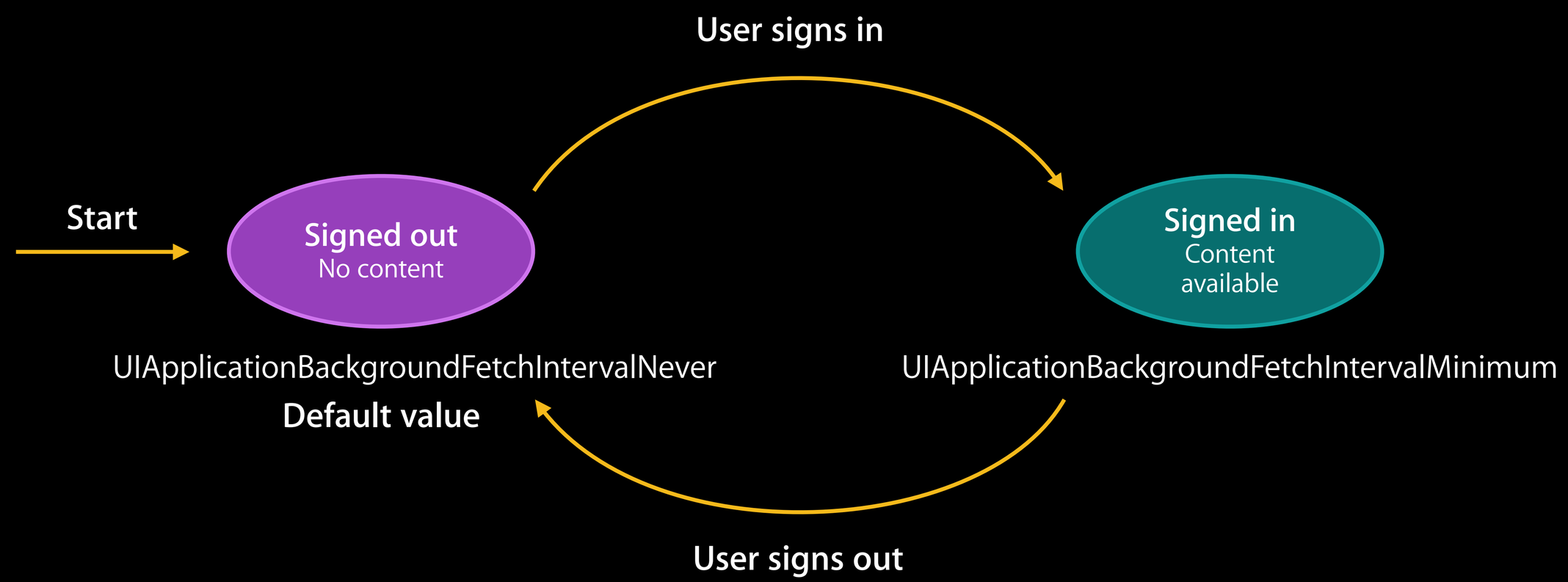
Background Fetch

Minimum Background Fetch Interval



Background Fetch

Minimum Background Fetch Interval



Background Fetch

Minimum Background Fetch Interval Custom Value

App Activity
After user activity or
system-initiated fetch

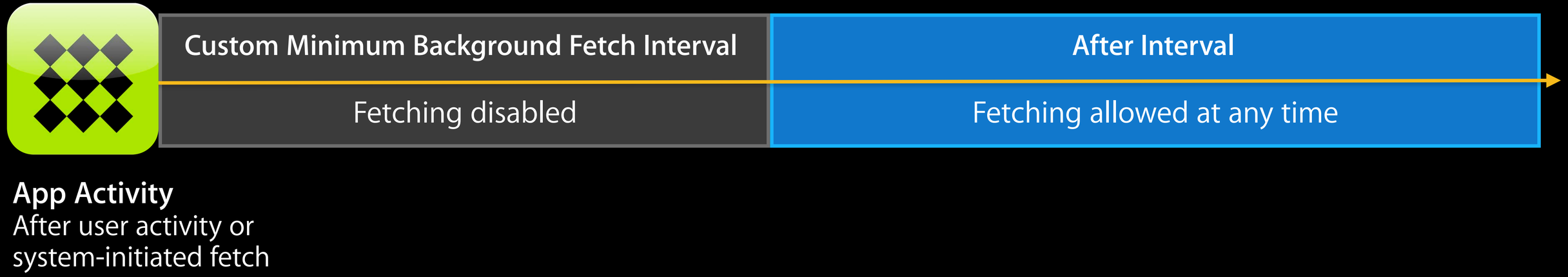
Background Fetch

Minimum Background Fetch Interval Custom Value

App Activity
After user activity or
system-initiated fetch

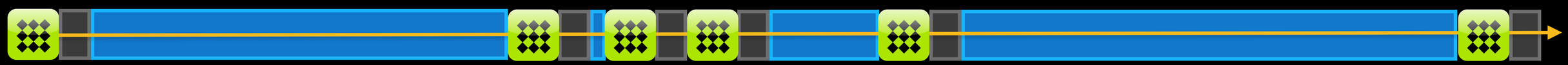
Background Fetch

Minimum Background Fetch Interval Custom Value



Background Fetch

Minimum Background Fetch Interval Custom Value



Demo

Brittany Hughes
iOS Software Engineer—SpringBoard

Background Fetch

Demo

- Make sure to pass completion handler all the way through
- Call completion handler with proper status

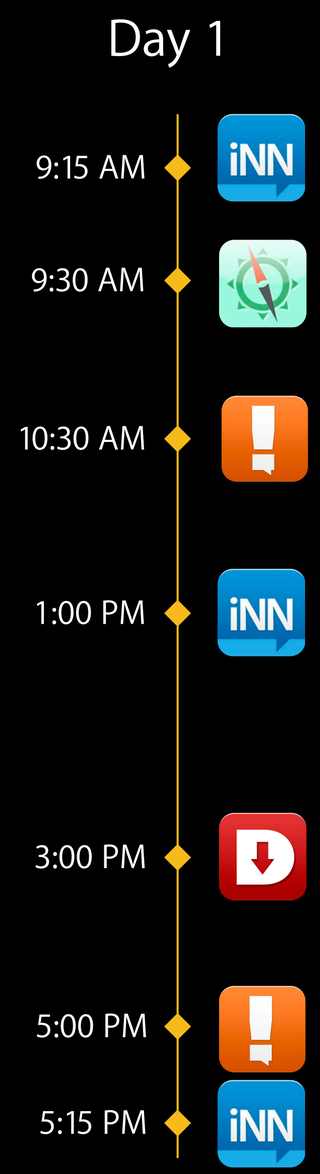
Background Fetch

Design

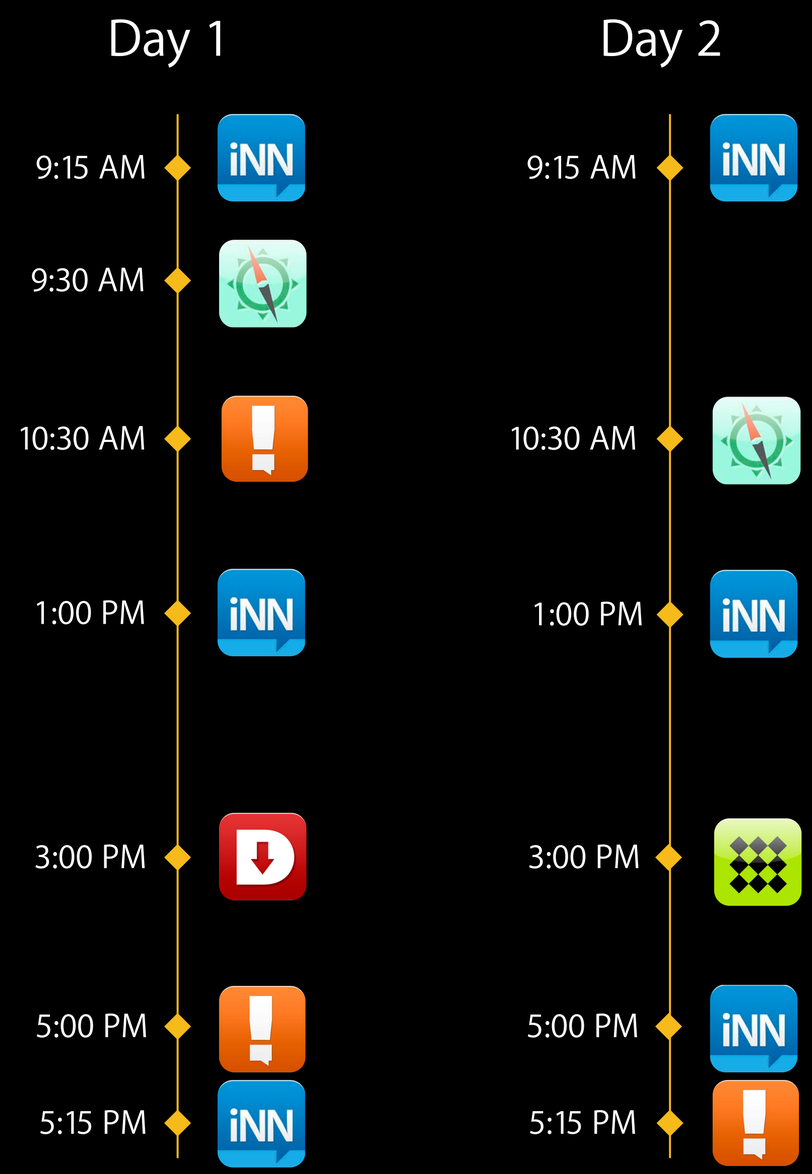
- System-scheduled fetch
  - Coalesced across applications
- Adapts to actual usage patterns on device
- Sensitive to energy and data usage
- Indifferent to actual app running state

Adapts to User Activity

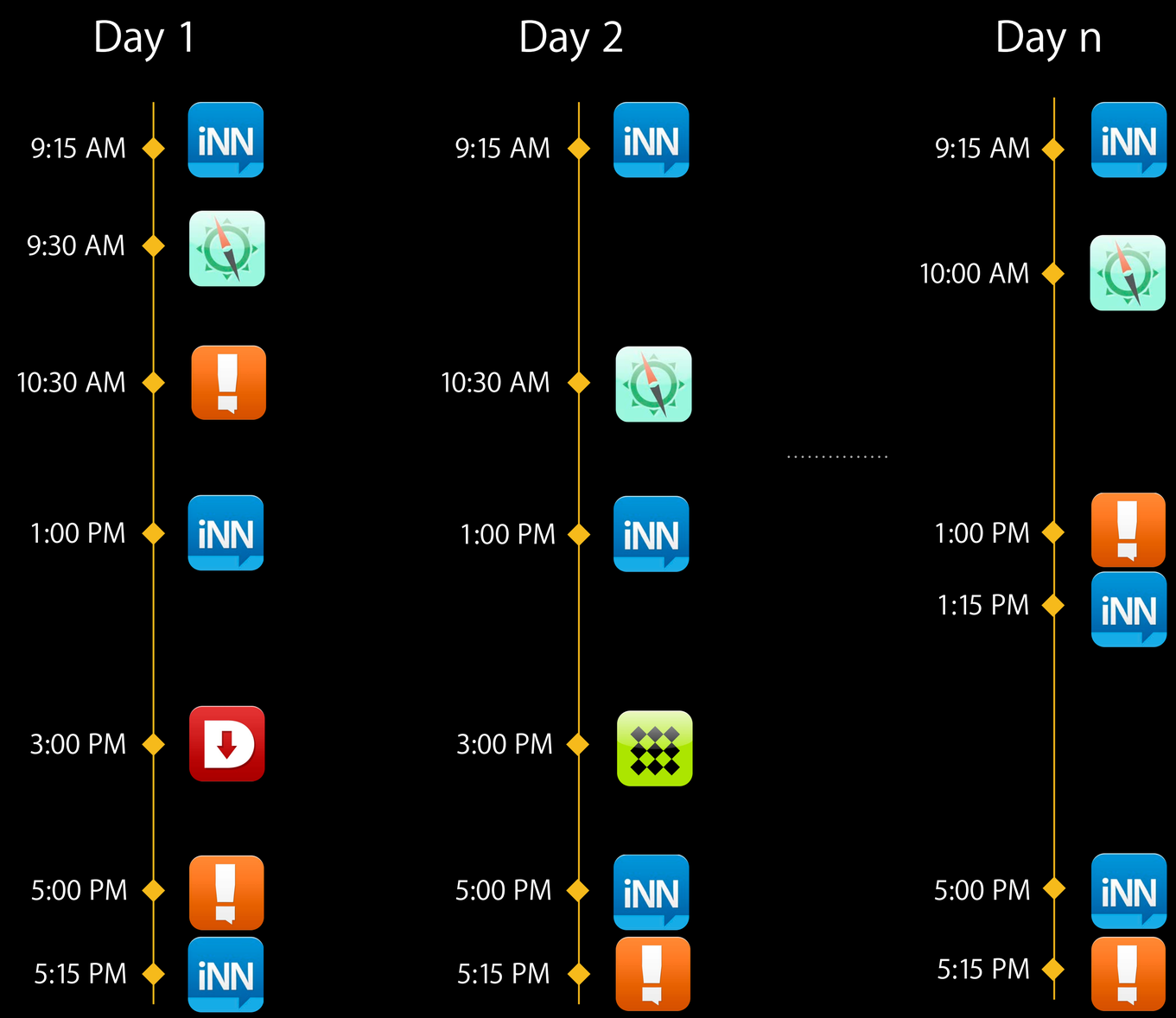
Adapts to User Activity



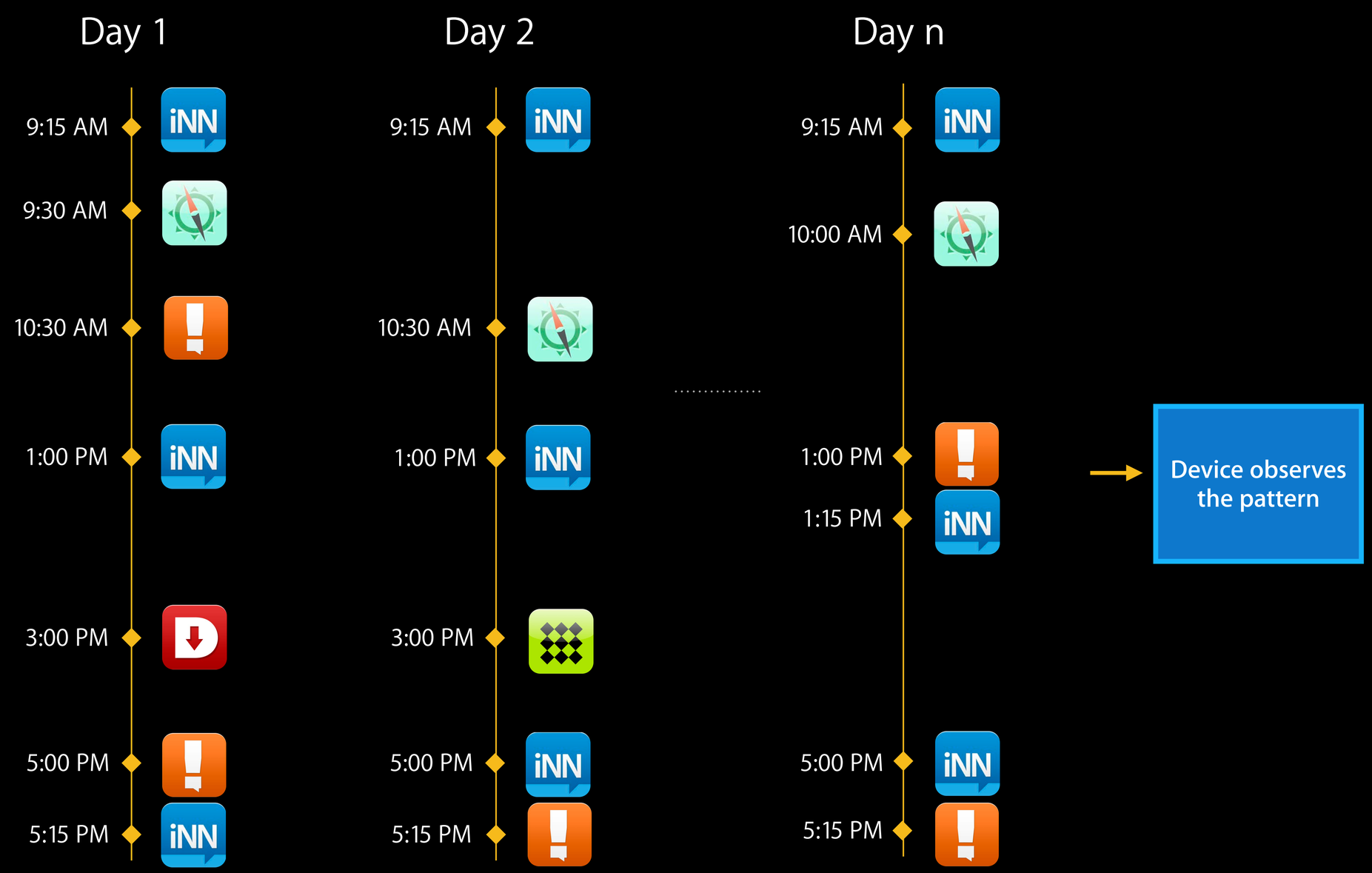
Adapts to User Activity



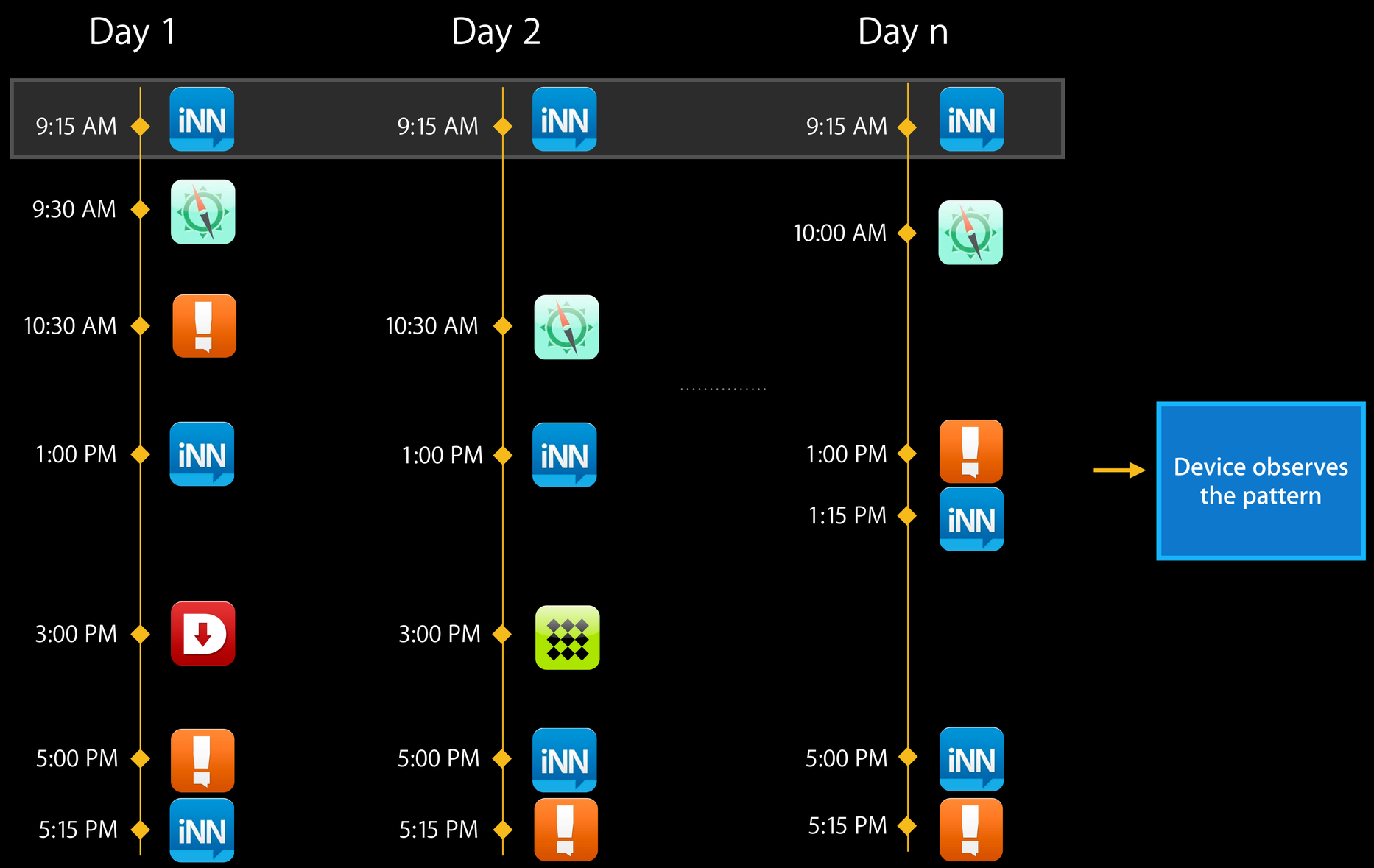
Adapts to User Activity



Adapts to User Activity

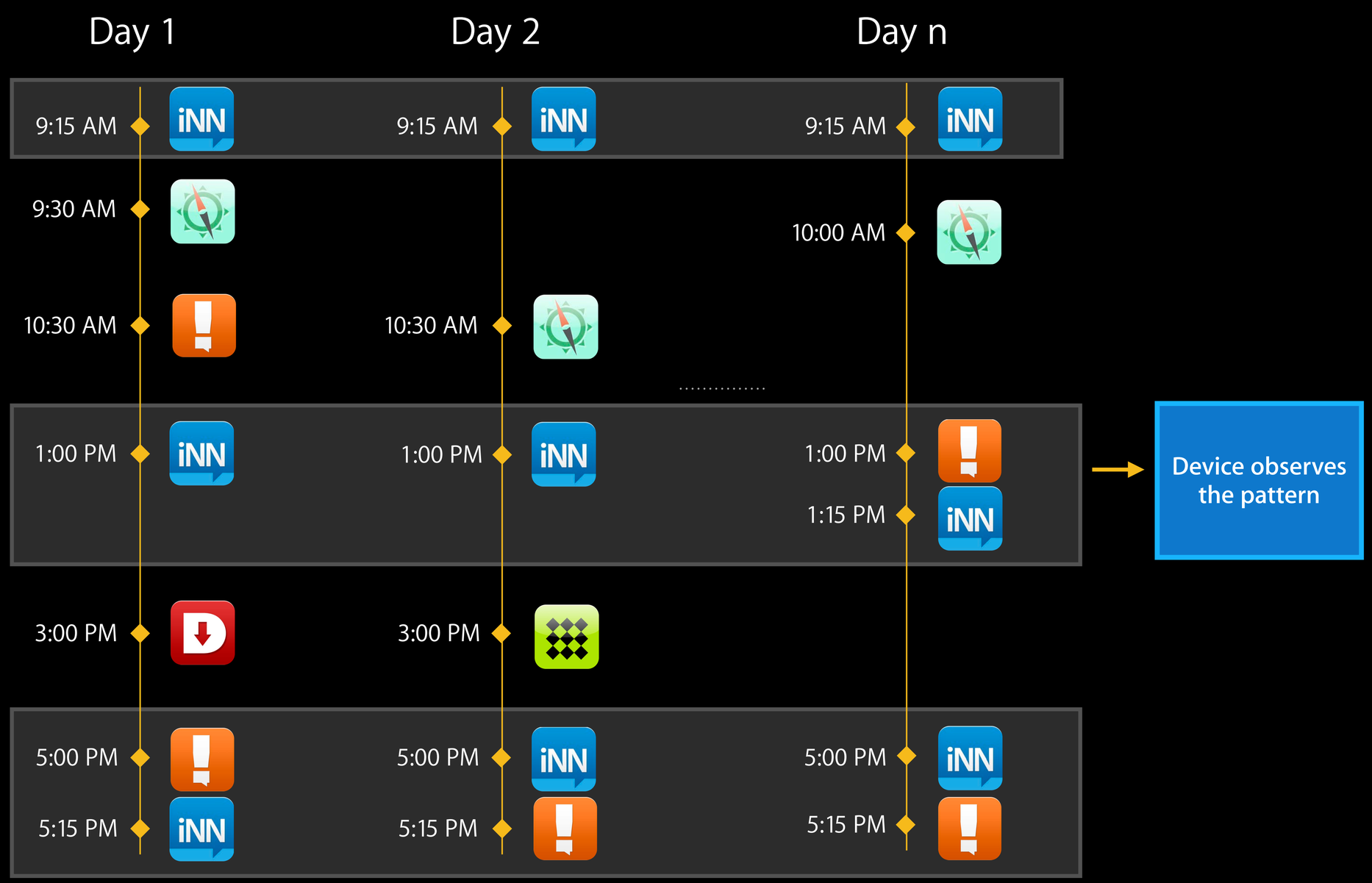


Adapts to User Activity

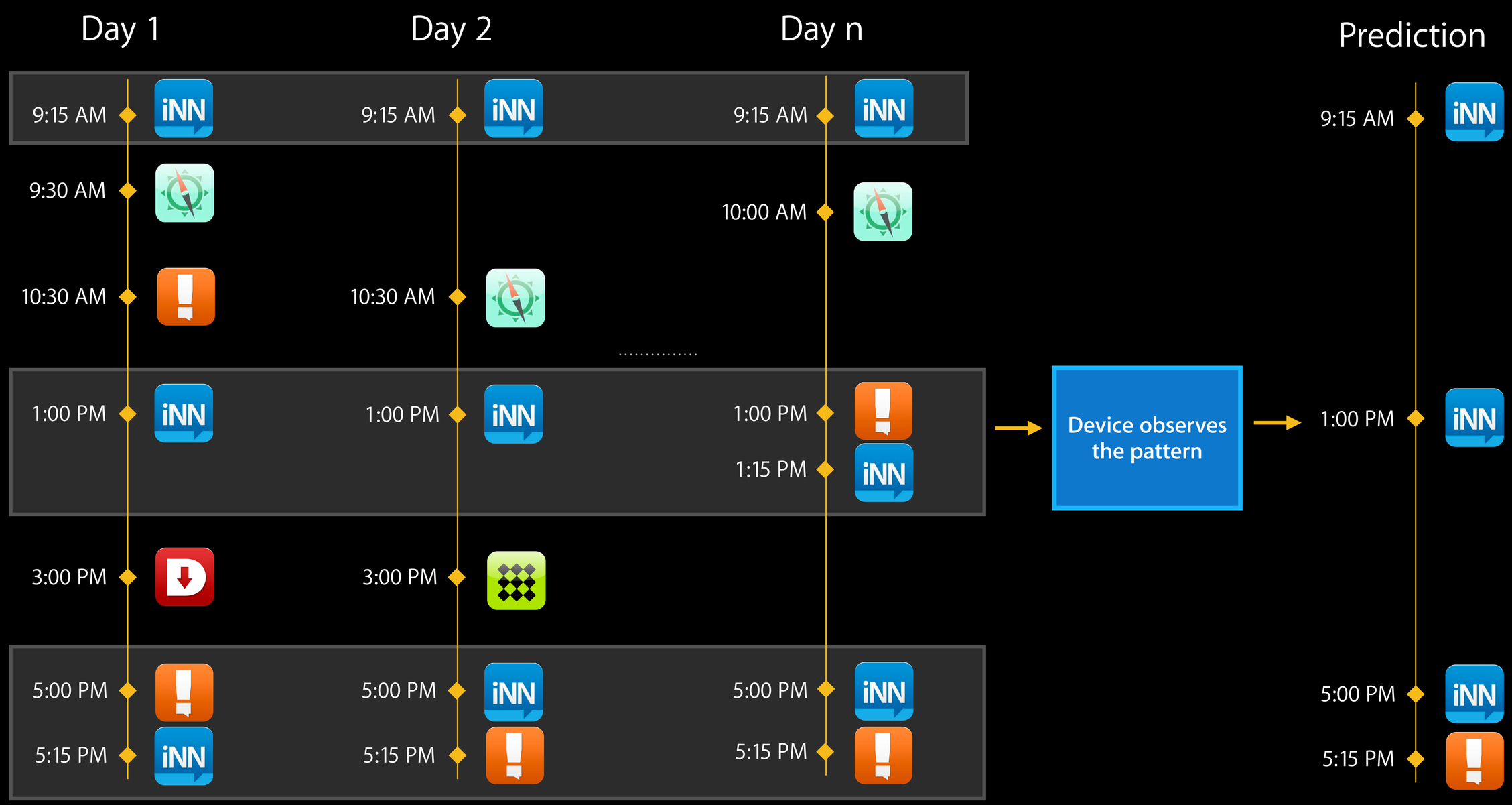


Adapts to User Activity

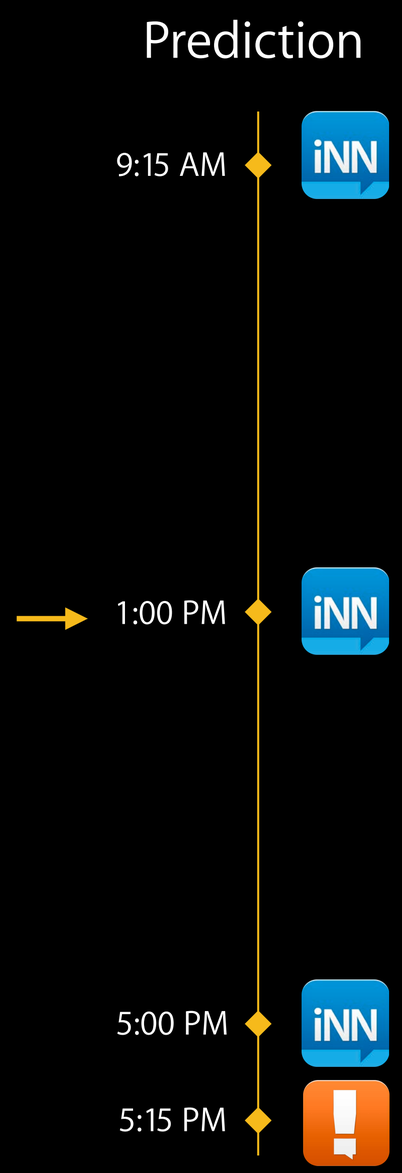
Adapts to User Activity



Adapts to User Activity



Adapts to User Activity



Learns patterns based on device usage

Coalesces fetches across apps

Avoids frequent fetching during periods of inactivity

Background Fetch

Examples

- Useful for many kinds of applications and features
  - Social network feeds
  - News and entertainment
  - Blog aggregators
  - Weather
  - Finance
- Use in conjunction with Background Transfers to automatically update
  - Photo sharing
  - Video sharing

New Multitasking APIs

- Background Fetch
- Remote Notifications
- Background Transfer Service

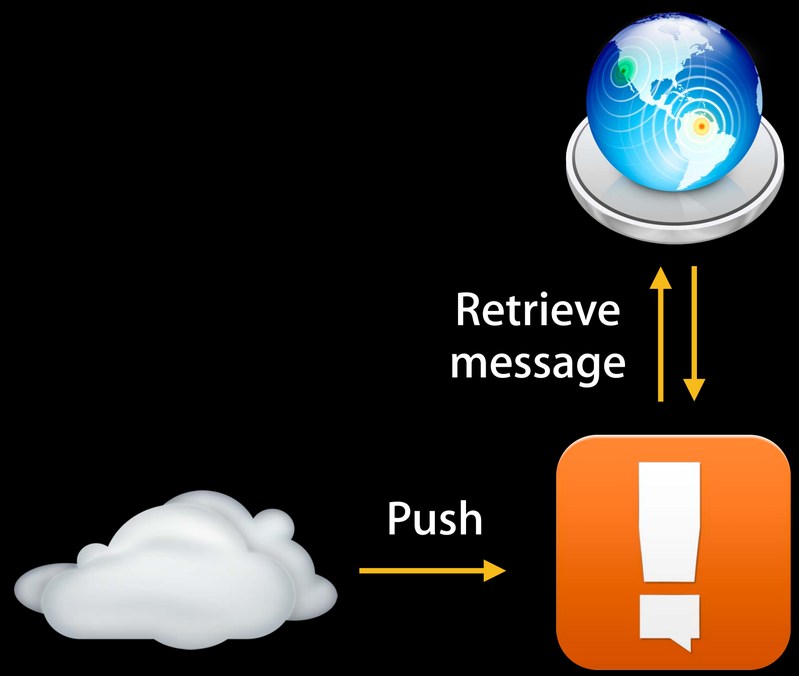
Remote Notifications in iOS 6

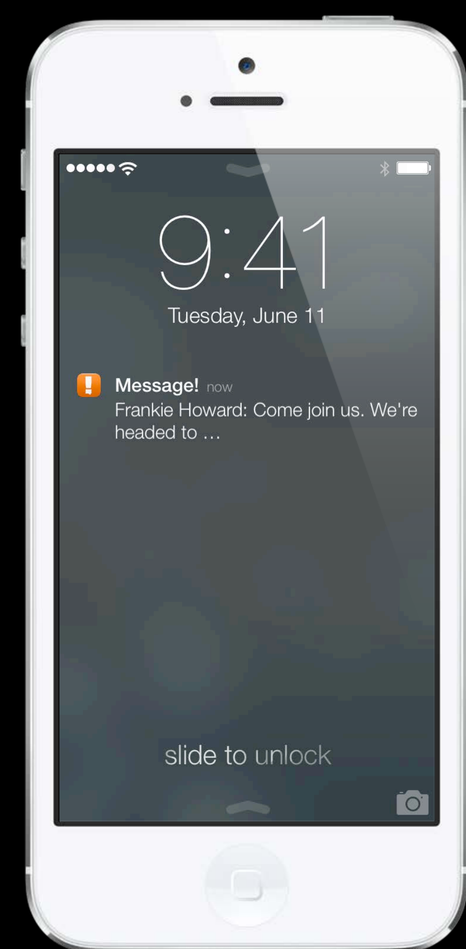
Users must wait for your app to catch up


```
aps {
  alert: {...}
}
```

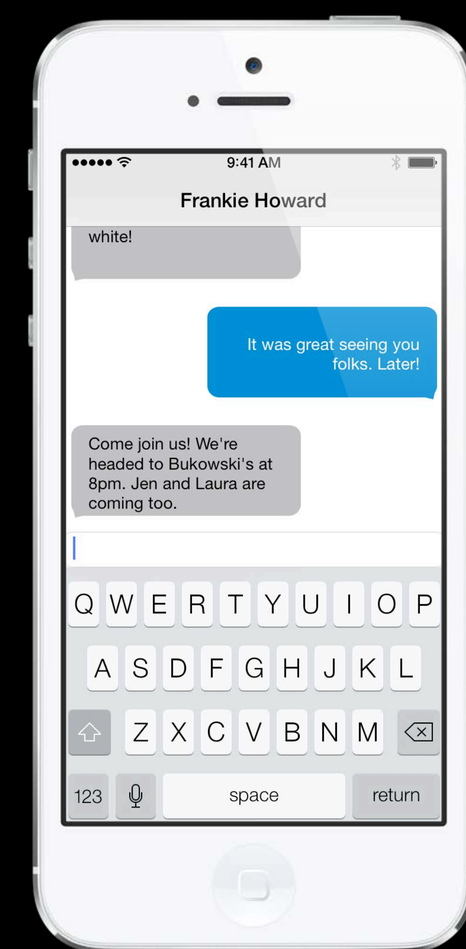
Remote Notifications in iOS 7

Can be delivered in the background to the app





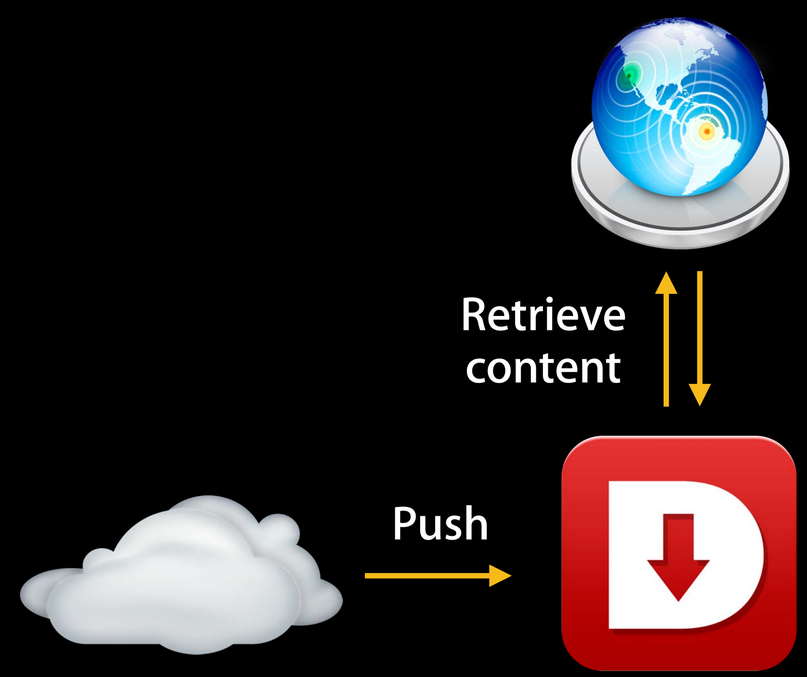
Slide to view




```
aps {
  content-available: 1
  alert: {...}
}
```

Silent Remote Notifications in iOS 7

Delivered in the background




```
aps {
  content-available: 1
}
```

Remote Notifications

UIKit API

Remote Notifications

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	remote-notification
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

Remote Notifications

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	remote-notification
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Set content-available flag in push notification

```
aps {
  content-available: 1
  alert: {...}
}
```

Remote Notifications

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	remote-notification
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Set content-available flag in push notification

```
aps {
  content-available: 1
  alert: {...}
}
```

3. Launched into background

Remote Notifications

UIKit API

1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	remote-notification
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Set content-available flag in push notification

```
aps {
  content-available: 1
  alert: {...}
}
```

3. Launched into background

Remote Notifications

UIKit API

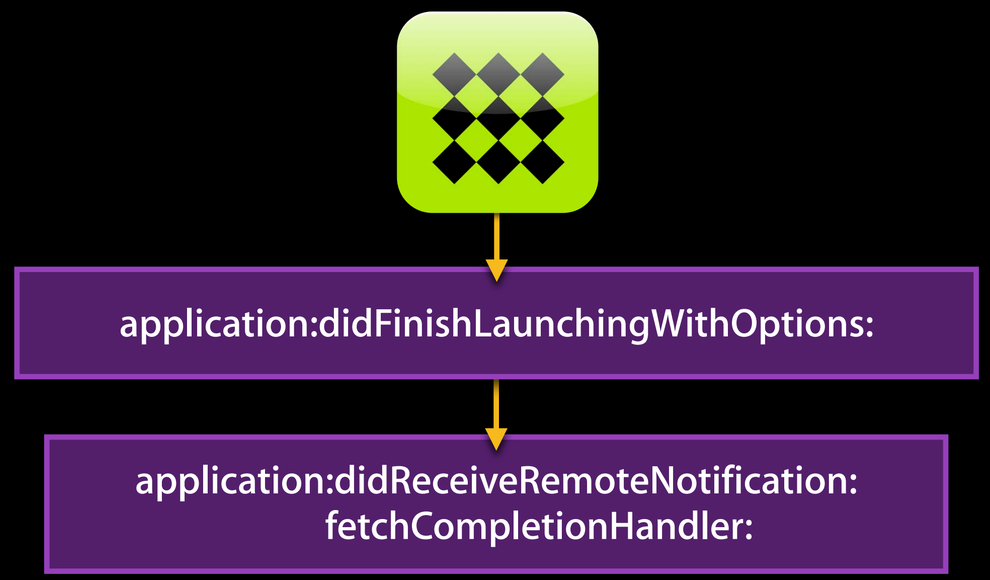
1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	remote-notification
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Set content-available flag in push notification

```
aps {
  content-available: 1
  alert: {...}
}
```

3. Launched into background



Remote Notifications

UIKit API

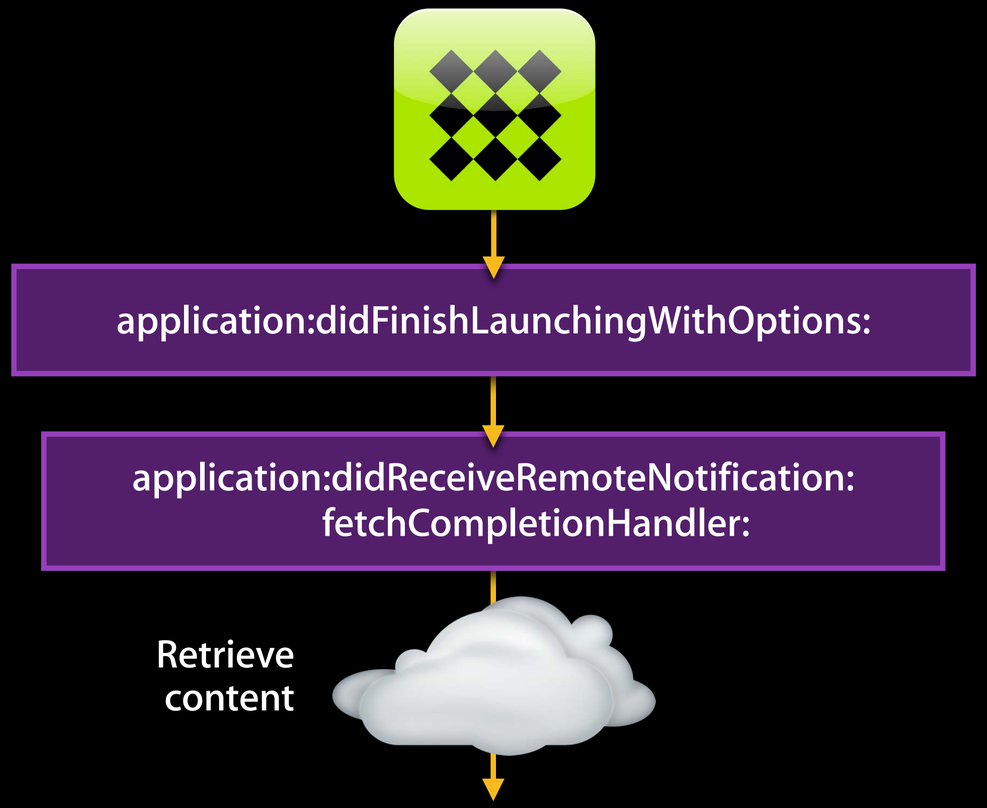
1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	remote-notification
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Set content-available flag in push notification

```
aps {
  content-available: 1
  alert: {...}
}
```

3. Launched into background



Remote Notifications

UIKit API

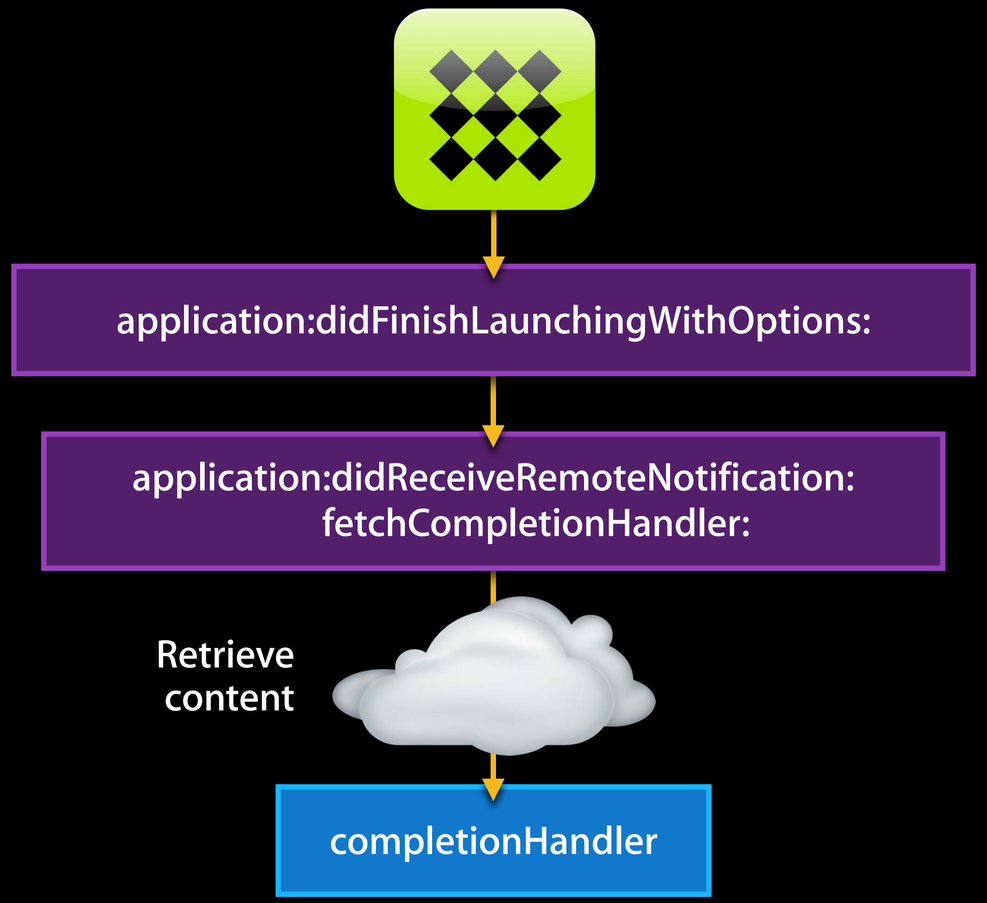
1. Info.plist

LSRequiresIPhoneOS	YES
UIBackgroundModes	remote-notification
UIMainStoryboardFile	...
UIRequiredDeviceCapabilities	{...}
UIStatusBarTintParameters	{...}
UISupportedInterfaceOrientation	{...}

2. Set content-available flag in push notification

```
aps {
  content-available: 1
  alert: {...}
}
```

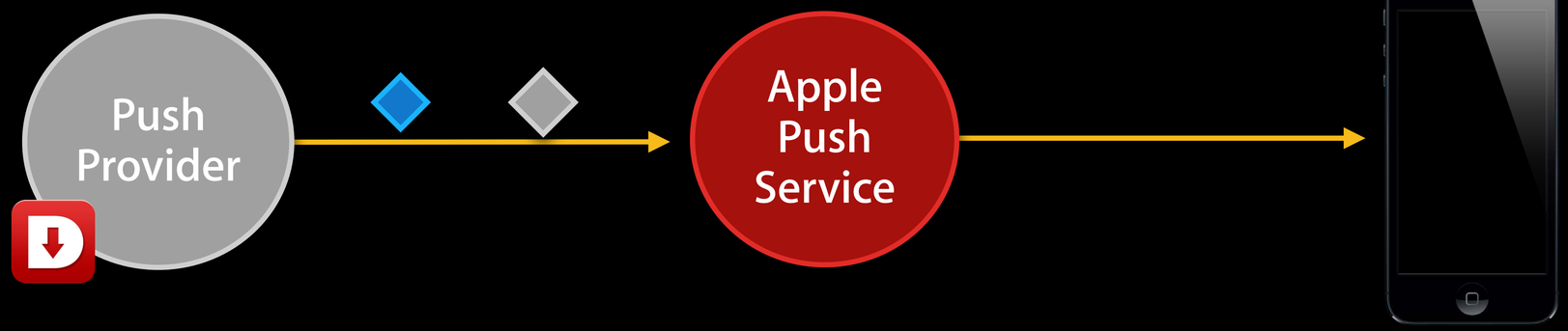
3. Launched into background



Silent Remote Notifications in iOS 7

Silent pushes are rate limited

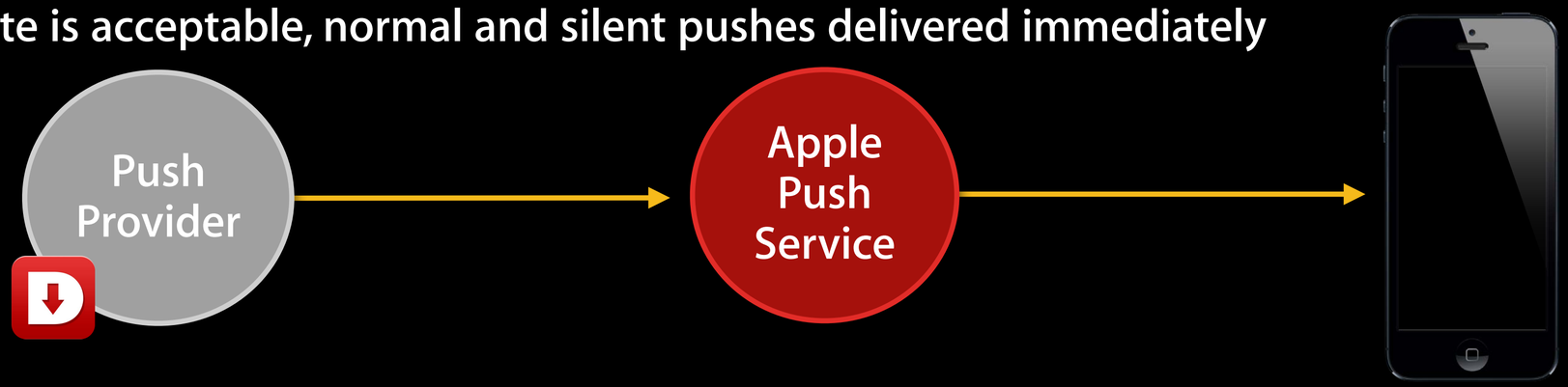
When push rate is acceptable, normal and silent pushes delivered immediately



Silent Remote Notifications in iOS 7

Silent pushes are rate limited

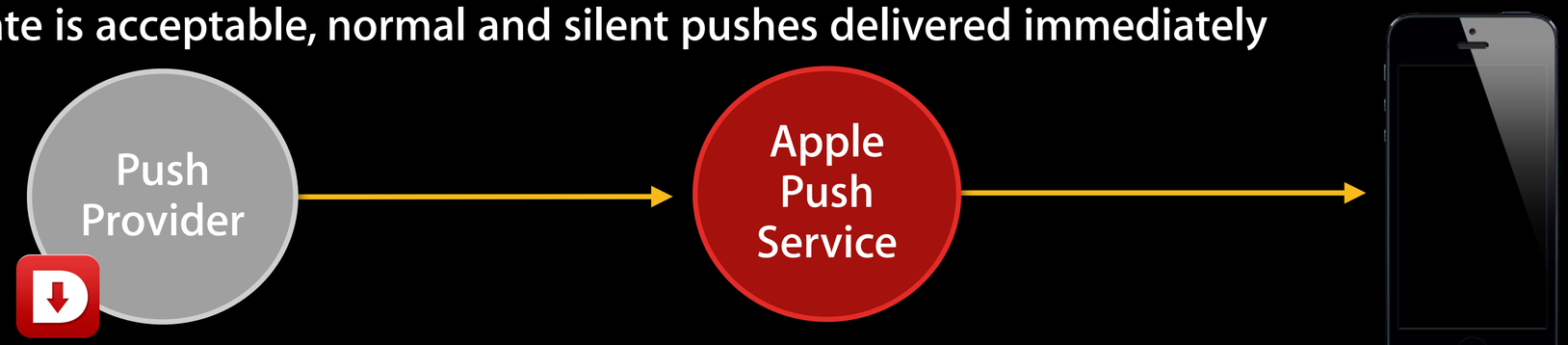
When push rate is acceptable, normal and silent pushes delivered immediately



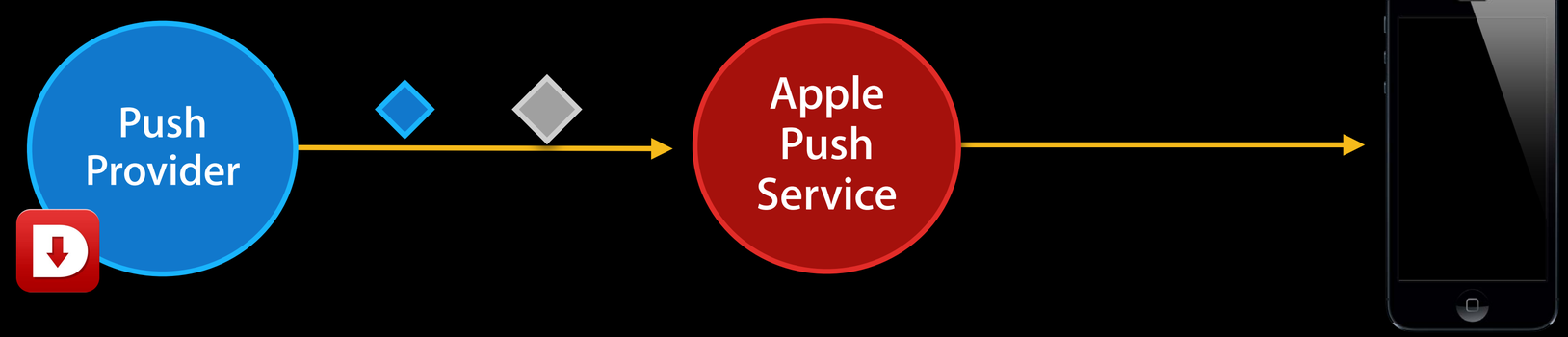
Silent Remote Notifications in iOS 7

Silent pushes are rate limited

When push rate is acceptable, normal and silent pushes delivered immediately



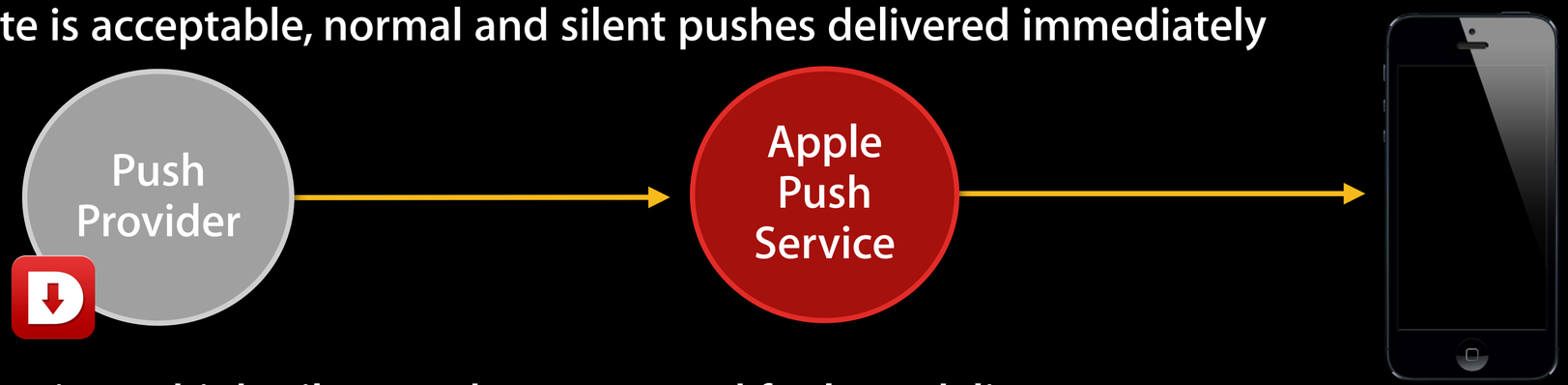
When push rate is too high, silent pushes are stored for later delivery



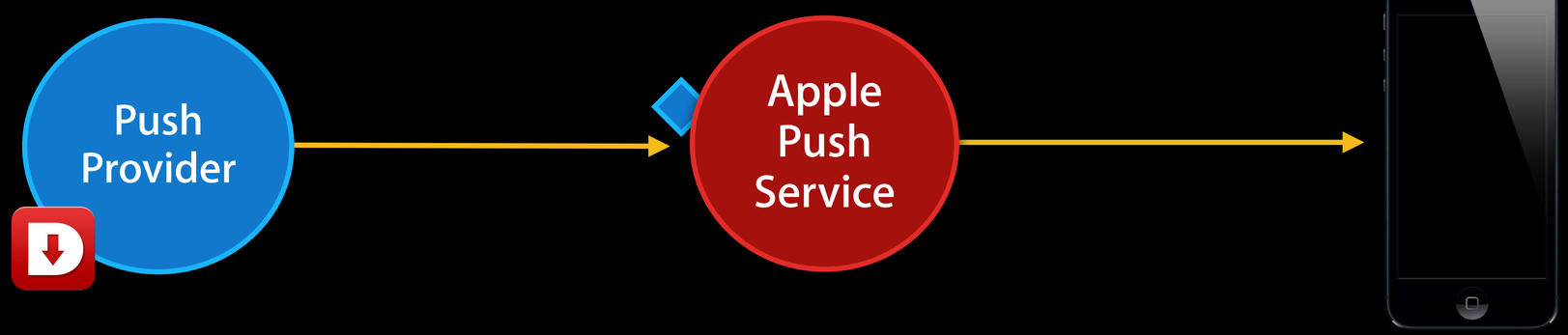
Silent Remote Notifications in iOS 7

Silent pushes are rate limited

When push rate is acceptable, normal and silent pushes delivered immediately



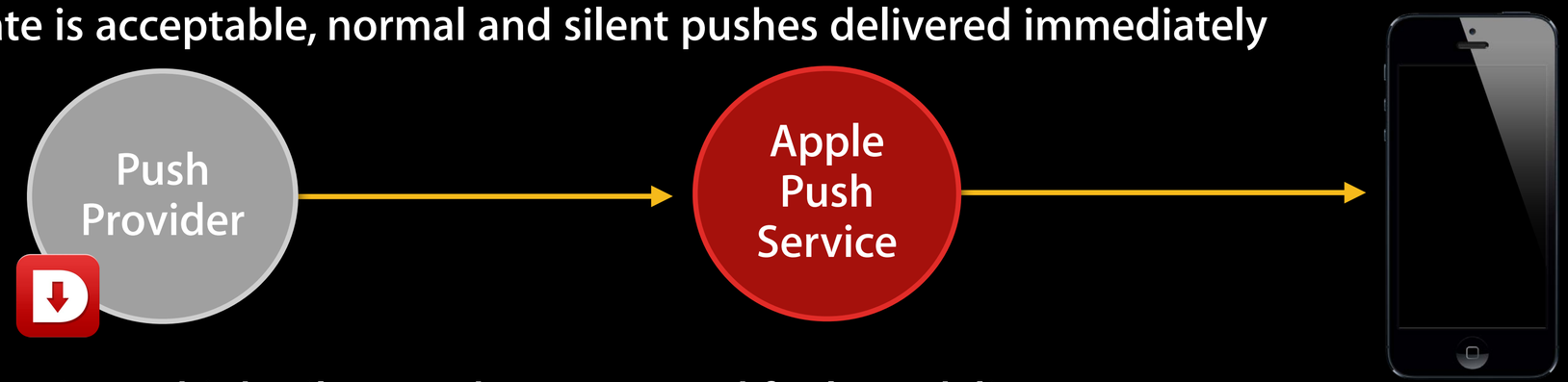
When push rate is too high, silent pushes are stored for later delivery



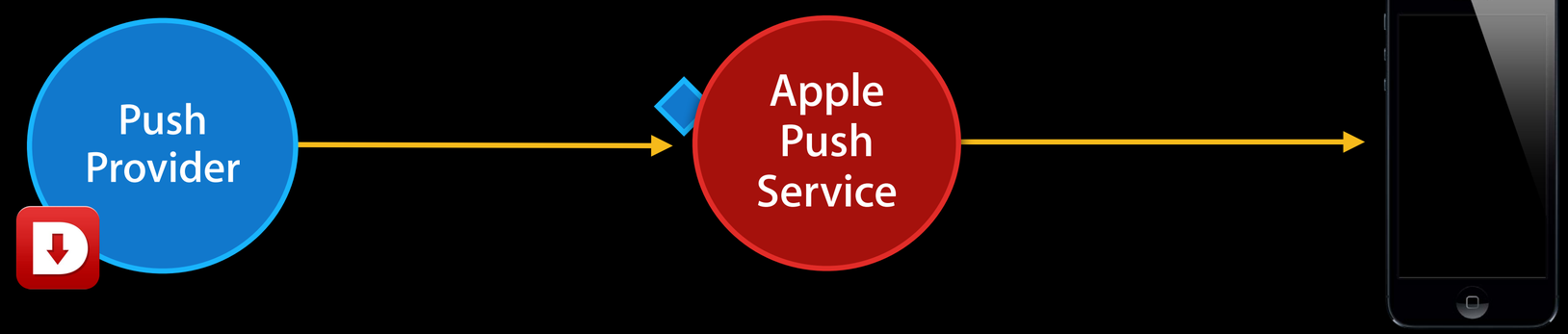
Silent Remote Notifications in iOS 7

Silent pushes are rate limited

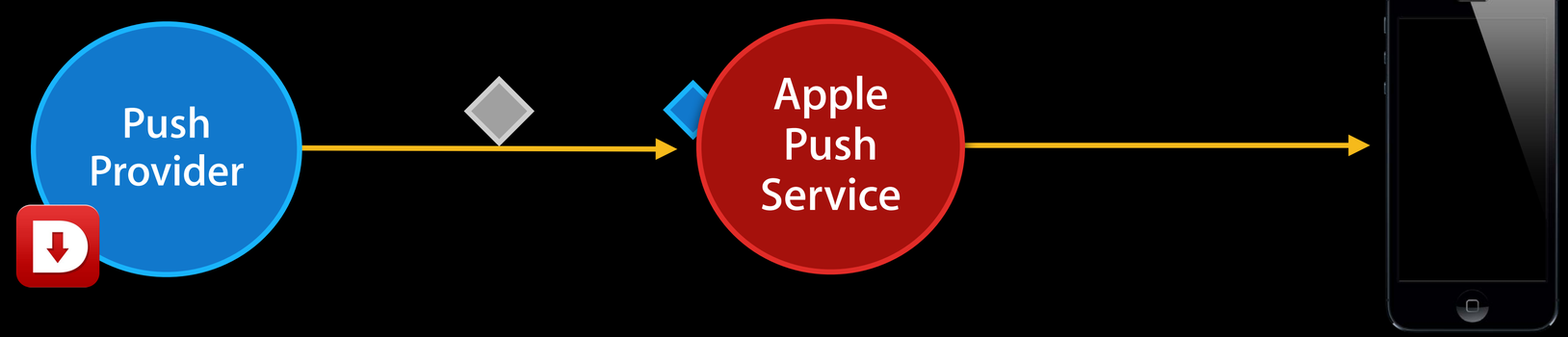
When push rate is acceptable, normal and silent pushes delivered immediately



When push rate is too high, silent pushes are stored for later delivery



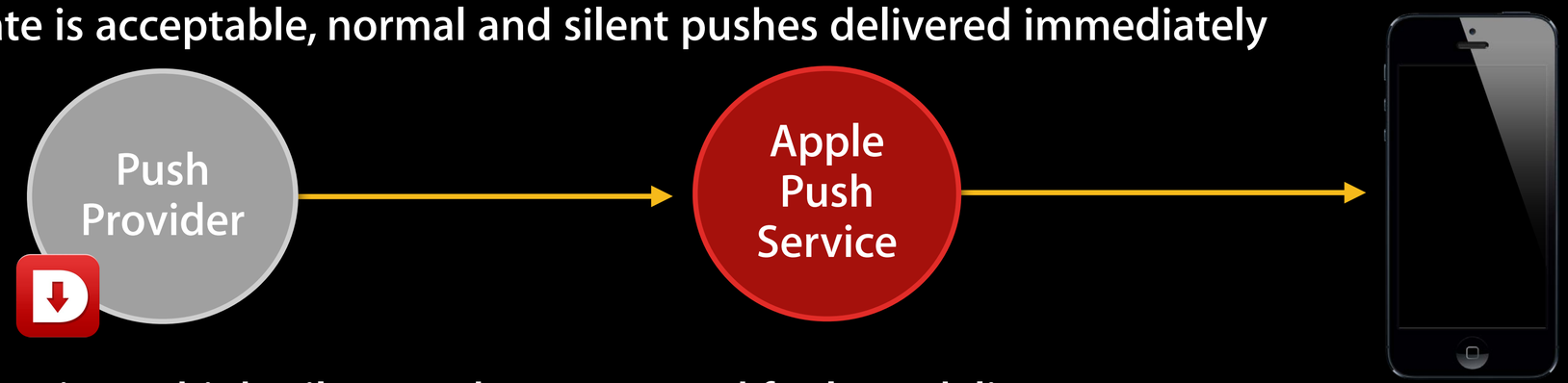
Stored pushes are delivered along with normal pushes and keep-alive responses



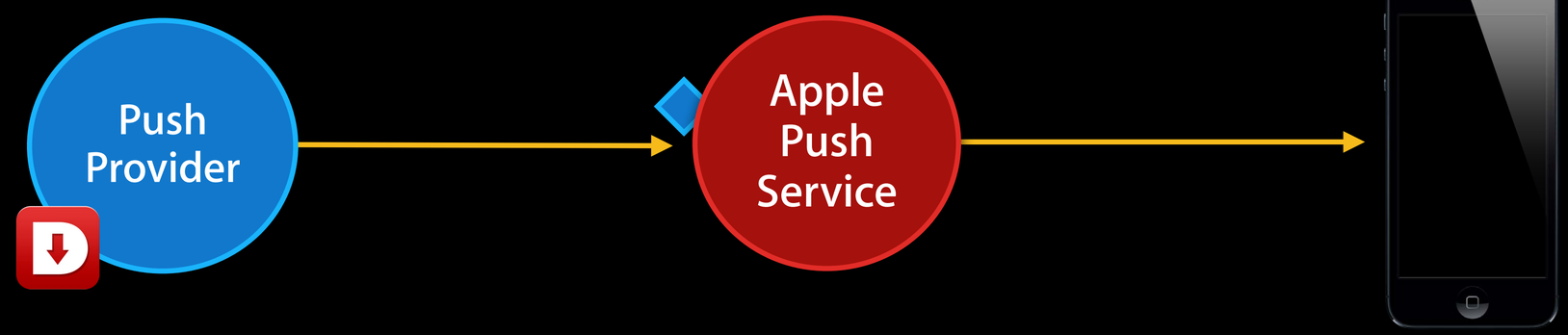
Silent Remote Notifications in iOS 7

Silent pushes are rate limited

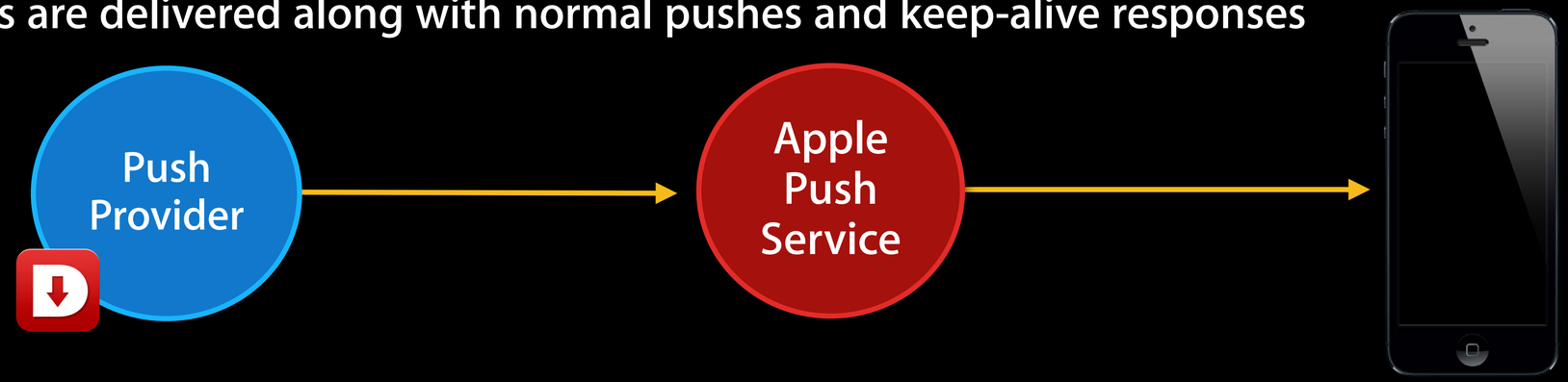
When push rate is acceptable, normal and silent pushes delivered immediately



When push rate is too high, silent pushes are stored for later delivery



Stored pushes are delivered along with normal pushes and keep-alive responses



Remote Notifications

Examples

- Can be used for many applications and features
  - Instant messaging
  - Picture messaging
  - Video messaging
  - Email
- Silent remote notifications can be useful for
  - Episodic content—TV shows, podcasts
  - Read these stories later
  - Purchase syncing
  - File syncing

Remote Notifications

Example: Auto-download in a TV app

Remote Notifications

Example: Auto-download in a TV app

- User asks for new episodes of a show to be downloaded when available

Remote Notifications

Example: Auto-download in a TV app

- User asks for new episodes of a show to be downloaded when available
- When episode is available, silent push is sent to the device

Remote Notifications

Example: Auto-download in a TV app

- User asks for new episodes of a show to be downloaded when available
- When episode is available, silent push is sent to the device
- App wakes up, checks for any newly available episodes

Remote Notifications

Example: Auto-download in a TV app

- User asks for new episodes of a show to be downloaded when available
- When episode is available, silent push is sent to the device
- App wakes up, checks for any newly available episodes
- Enqueues them in the Background Transfer service

Remote Notifications

Example: Auto-download in a TV app

- User asks for new episodes of a show to be downloaded when available
- When episode is available, silent push is sent to the device
- App wakes up, checks for any newly available episodes
- Enqueues them in the Background Transfer service
- When completed, app wakes up and updates UI

Remote Notifications

Example: Auto-download in a TV app

- User asks for new episodes of a show to be downloaded when available
- When episode is available, silent push is sent to the device
- App wakes up, checks for any newly available episodes
- Enqueues them in the Background Transfer service
- When completed, app wakes up and updates UI
- Sends local notification to notify user when episode is ready to watch

Remote Notifications

Example: File syncing app

Remote Notifications

Example: File syncing app

- User tags a few files as favorites to be updated all the time

Remote Notifications

Example: File syncing app

- User tags a few files as favorites to be updated all the time
- Whenever those file change, the service sends a silent push

Remote Notifications

Example: File syncing app

- User tags a few files as favorites to be updated all the time
- Whenever those file change, the service sends a silent push
  - Because silent pushes are rate limited already

Remote Notifications

Example: File syncing app

- User tags a few files as favorites to be updated all the time
- Whenever those file change, the service sends a silent push
  - Because silent pushes are rate limited already
- App wakes up, checks for any newly available file updates

Remote Notifications

Example: File syncing app

- User tags a few files as favorites to be updated all the time
- Whenever those file change, the service sends a silent push
  - Because silent pushes are rate limited already
- App wakes up, checks for any newly available file updates
- Enqueues file diffs into the Background Transfer Service

Remote Notifications

Example: File syncing app

- User tags a few files as favorites to be updated all the time
- Whenever those file change, the service sends a silent push
  - Because silent pushes are rate limited already
- App wakes up, checks for any newly available file updates
- Enqueues file diffs into the Background Transfer Service
- When completed, app wakes up and updates UI

Remote Notifications

Summary

- Receive push notifications in the background
- Silent pushes are rate limited—a handful per hour

New Multitasking APIs

	Background Fetch	Remote Notifications
Content Importance	Interesting, but not critical	Immediate
Content Availability	Very frequent	Infrequent or sporadic
Examples	News Social networking feeds Photo sharing	Instant messaging Synced content Read this later

New Multitasking APIs

- Background Fetch
- Remote Notifications
- Background Transfer Service

Background Transfer Service

Motivation

- In iOS 6, apps can transfer files while in the foreground or for a few minutes when the app returns to the background
- Limited arbitrarily by time
- Couldn’t effectively auto-download content or upload large assets

Background Transfer Service

Motivation

- Downloads and uploads managed by iOS
- Continue even after app exits
- Not restricted by time
- Enqueue at any time—from foreground or background
- App woken up to handle authentication, errors, or completion

Background Transfer Service

CFNetwork and UIKit API

Background Transfer Service

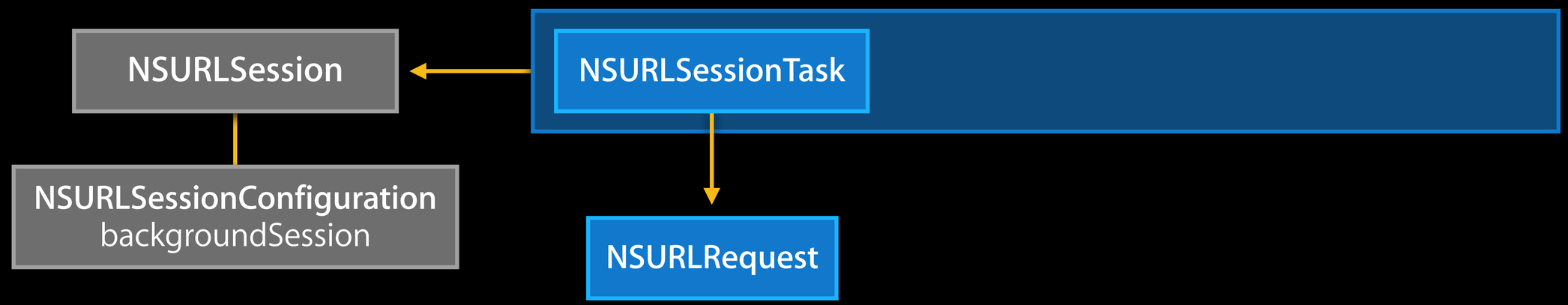
CFNetwork and UIKit API

1. Create a background NSURLSession and add download or upload tasks

Background Transfer Service

CFNetwork and UIKit API

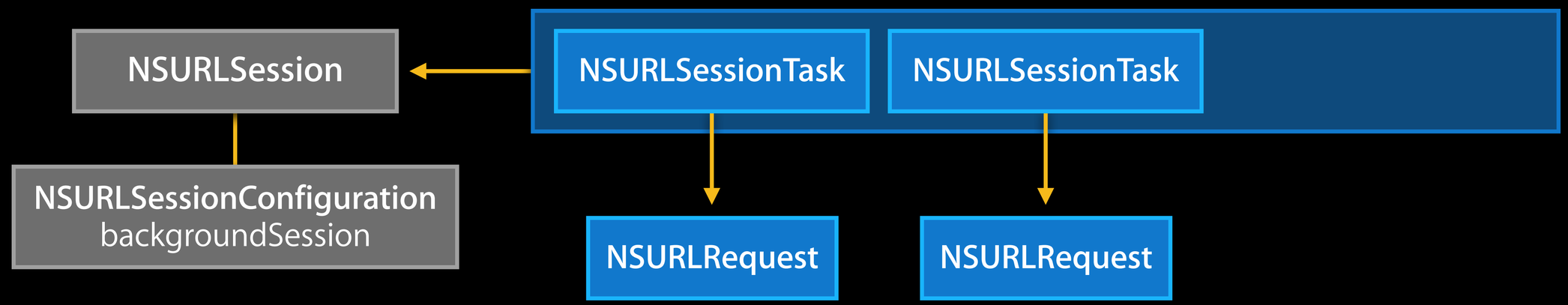
1. Create a background NSURLSession and add download or upload tasks



Background Transfer Service

CFNetwork and UIKit API

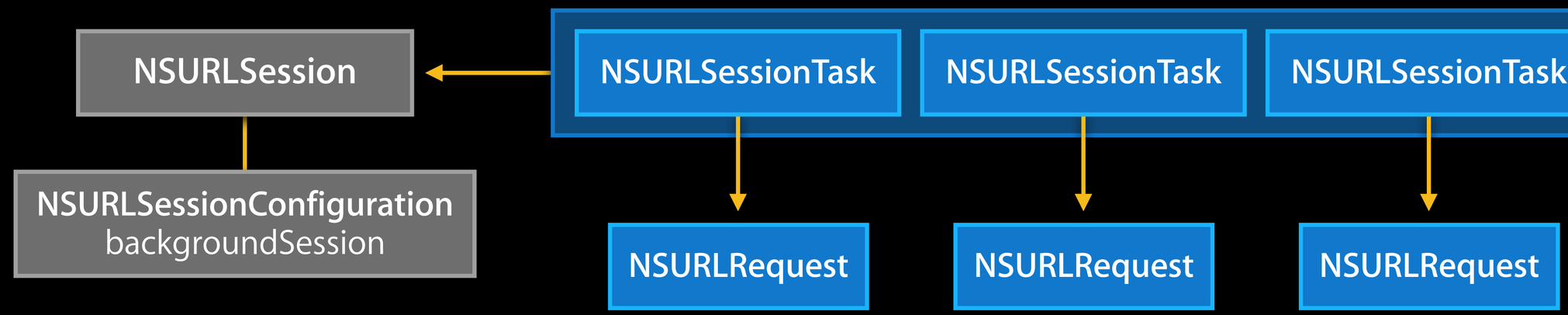
1. Create a background NSURLSession and add download or upload tasks



Background Transfer Service

CFNetwork and UIKit API

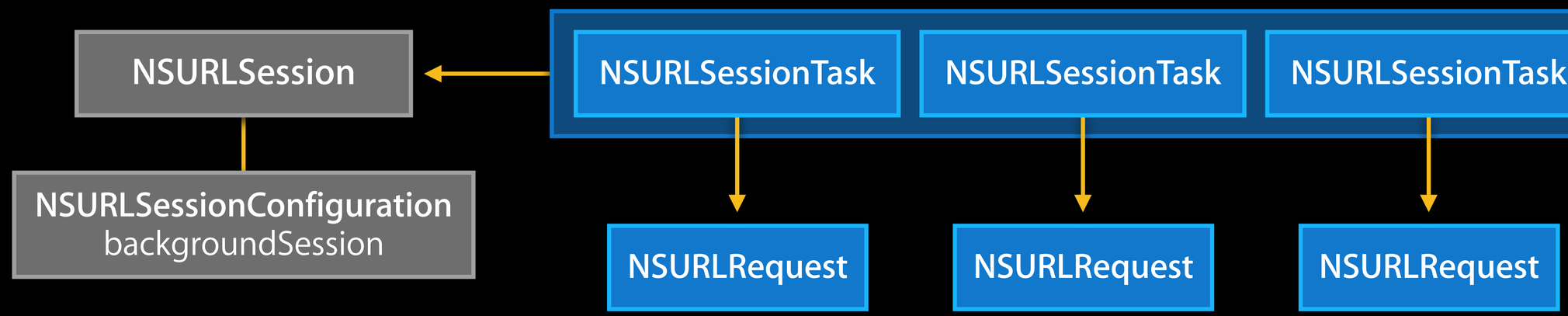
1. Create a background NSURLSession and add download or upload tasks



Background Transfer Service

CFNetwork and UIKit API

1. Create a background NSURLSession and add download or upload tasks

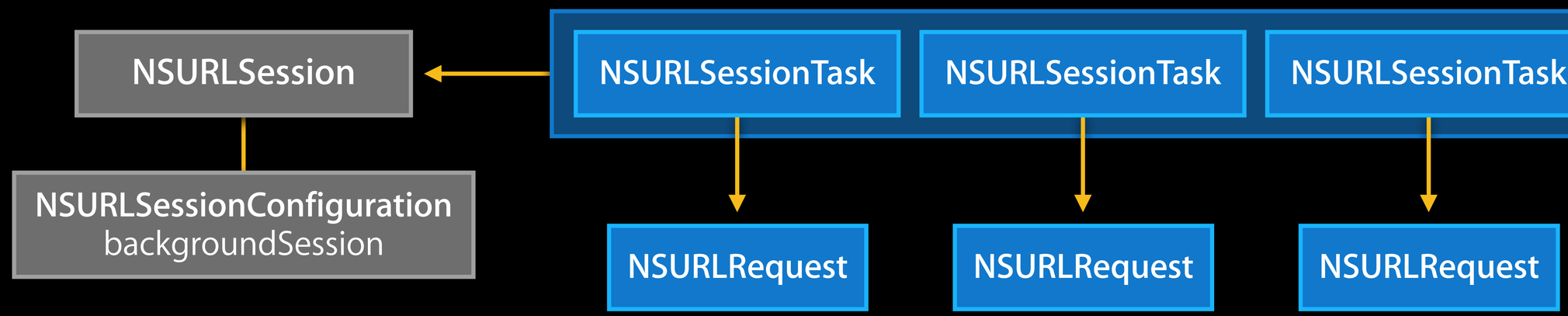


2. Launched into background

Background Transfer Service

CFNetwork and UIKit API

1. Create a background NSURLSession and add download or upload tasks

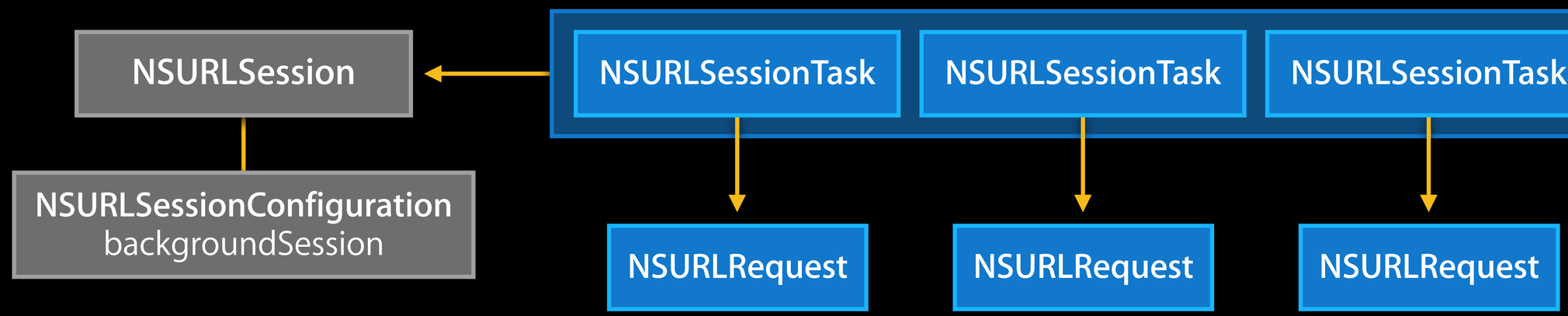


2. Launched into background

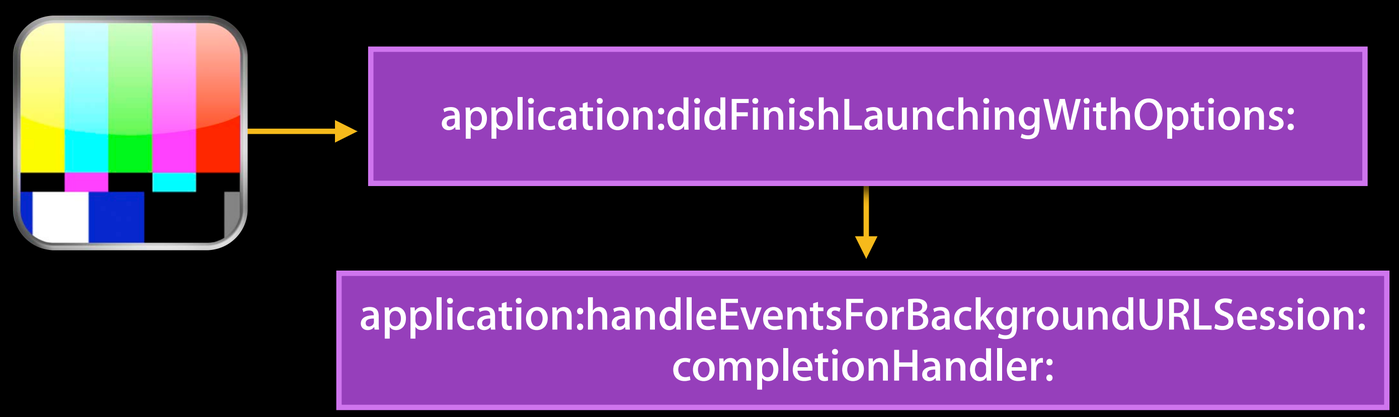
Background Transfer Service

CFNetwork and UIKit API

1. Create a background NSURLSession and add download or upload tasks



2. Launched into background

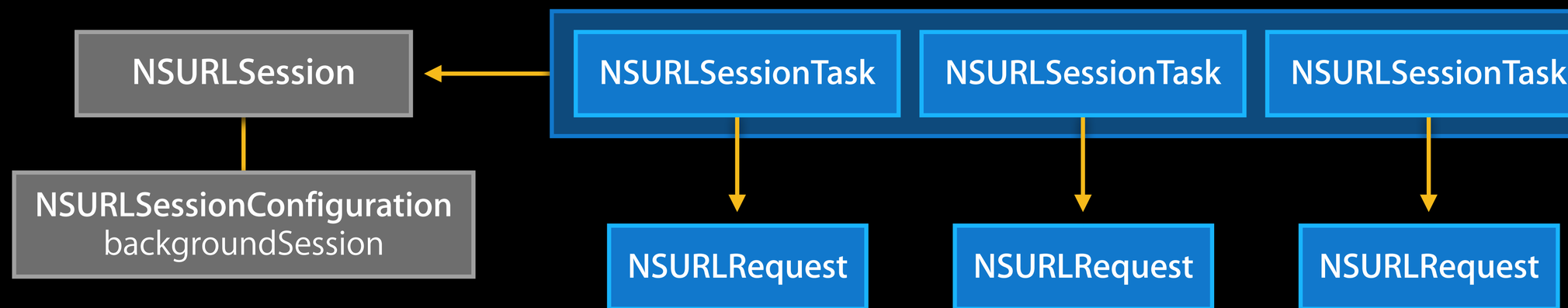


Handle new content

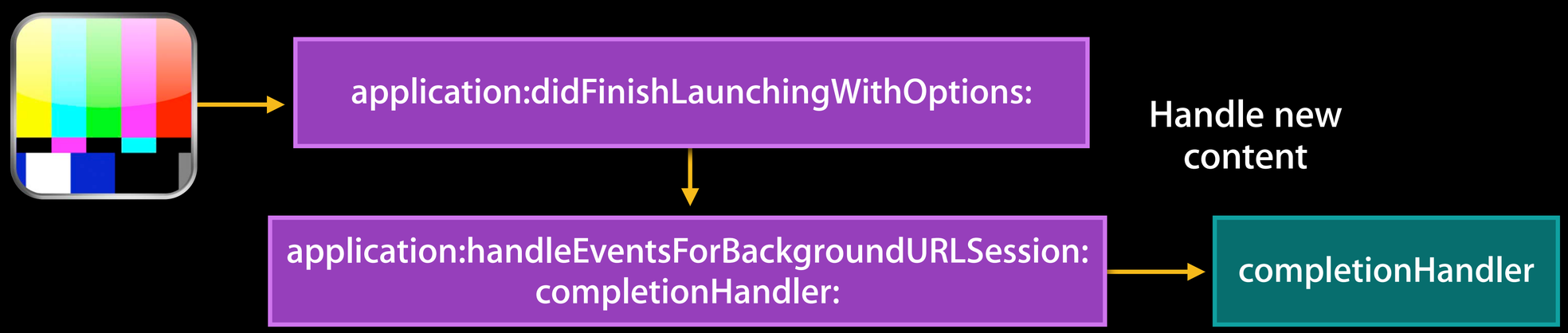
Background Transfer Service

CFNetwork and UIKit API

1. Create a background NSURLSession and add download or upload tasks



2. Launched into background



Background Transfer Service

Discretionary transfers help preserve energy and data

- Discretionary transfers are power-managed and will only go over WiFi
- From the background, transfers are always discretionary
- From the foreground, transfers can optionally request discretionary

Background Transfer Service

Examples

- Useful for uploading photos or videos
- Use in combination with other Multitasking modes
  - Region Monitoring and Significant Location Changes
  - Background Fetch and Remote Notifications
- Helps keep your app up to date by downloading in the background
  - Shared photos and videos
  - Purchased books or other content
  - TV shows
  - Podcasts
  - Game content

New Multitasking APIs

- Background Fetch
- Remote Notifications
- Background Transfer Service

Considerations

- Limited time to run in background
- Background completion tasks initiated from the background
- Seamless background experience
- Data Protection and Keychain
- Efficient battery life usage
- Cellular data usage
- Removed from the App Switcher
- Background App Refresh Settings

Limited Time in Background

Apps given limited time to update content and UI

- Given less than a minute to finish update
- Fetched in parallel with other apps
  - Optimize launch and fetching paths for CPU usage
  - Use Time Profiler in Instruments
- Important to complete as soon as possible

Background Task Completion Changes

From background activity

- When woken for
  - Background Fetch
  - Remote Notifications
  - Background Transfer
- Background tasks are only given seconds rather than minutes of run time

Seamless Background Experience

Snapshots and state restoration

- Snapshot and state restoration saved after calling completion handler
- Configure view hierarchy to hide passwords, etc.
- Use State Restoration to make sure app seamlessly transitions from snapshot to live

Data Protection

Always use data protection when handling sensitive user data

	NSFileProtection	Keychain
High	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
Security level	NSFileProtectionCompleteUnlessOpen	
	NSFileProtectionCompleteUntilFirstUserAuthentication	kSecAttrAccessibleAfterFirstUnlock
Low	NSFileProtectionNone	kSecAttrAccessibleAlways

Data Protection

Always use data protection when handling sensitive user data

	NSFileProtection	Keychain
High	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
Security level	NSFileProtectionCompleteUnlessOpen	
	NSFileProtectionCompleteUntilFirstUserAuthentication	kSecAttrAccessibleAfterFirstUnlock
Low	NSFileProtectionNone	kSecAttrAccessibleAlways

Data Protection

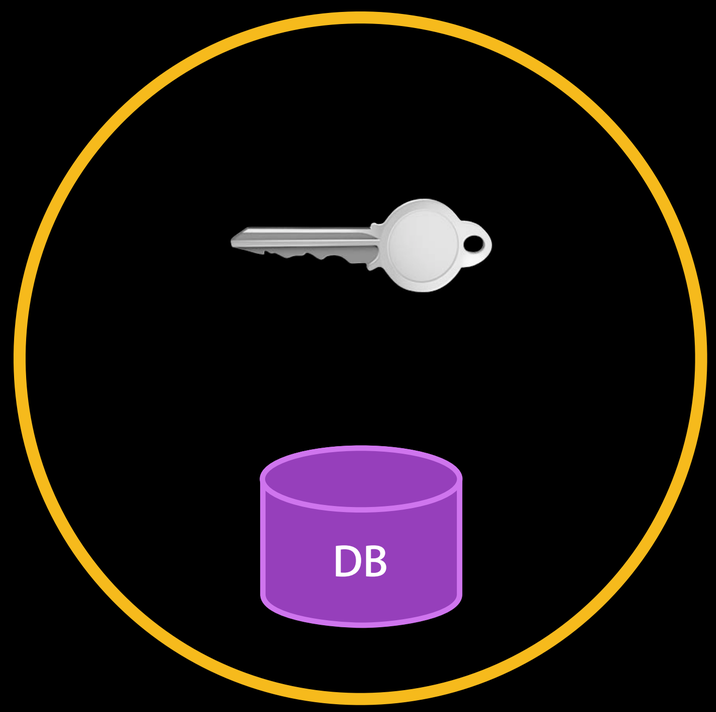
Always use data protection when handling sensitive user data

Security level: High → Low

NSData	sqlite3
NSDataWritingFileProtectionComplete	SQLITE_OPEN_FILEPROTECTION_COMPLETE
NSDataWritingFileProtectionCompleteUnlessOpen	SQLITE_OPEN_FILEPROTECTION_COMPLETEUNLESSOPEN
NSDataWritingFileProtectionCompleteUntilFirstUserAuthentication	SQLITE_OPEN_FILEPROTECTION_COMPLETEUNTILFIRSTUSERAUTHENTICATION
NSDataWritingFileProtectionNone	SQLITE_OPEN_FILEPROTECTION_NONE

Data Protection

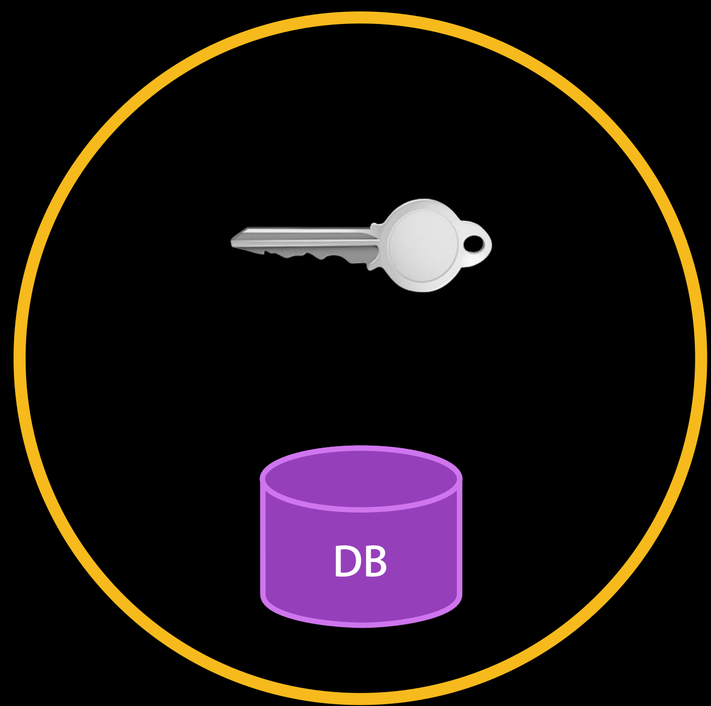
How to handle data while running in the background



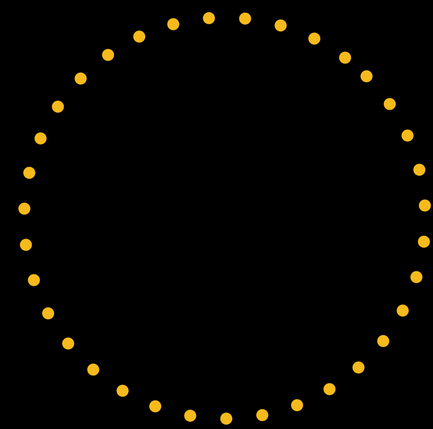
Complete Protection
Data accessible while unlocked

Data Protection

How to handle data while running in the background



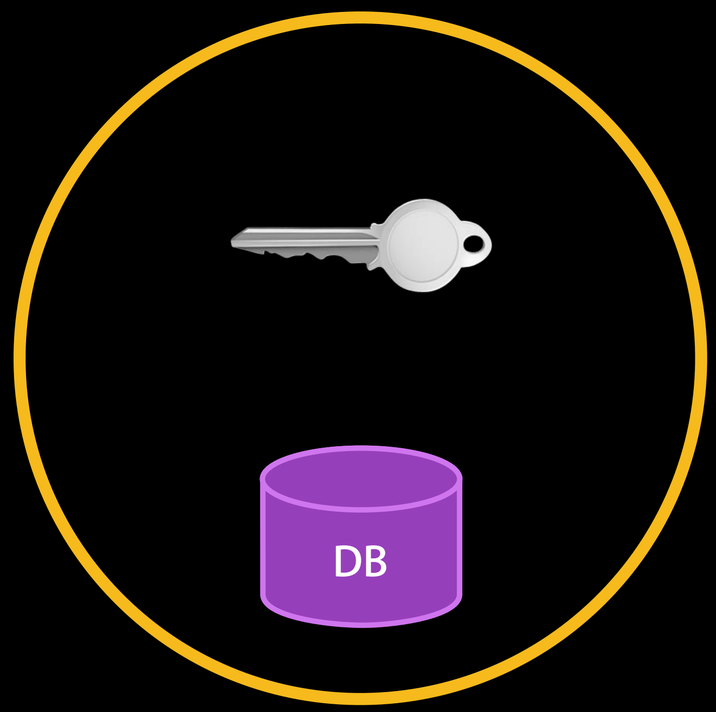
Complete Protection
Data accessible while unlocked



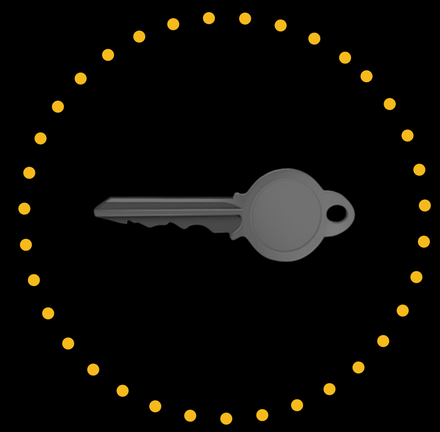
Partial Protection
Protects data from reboot until first unlock

Data Protection

How to handle data while running in the background



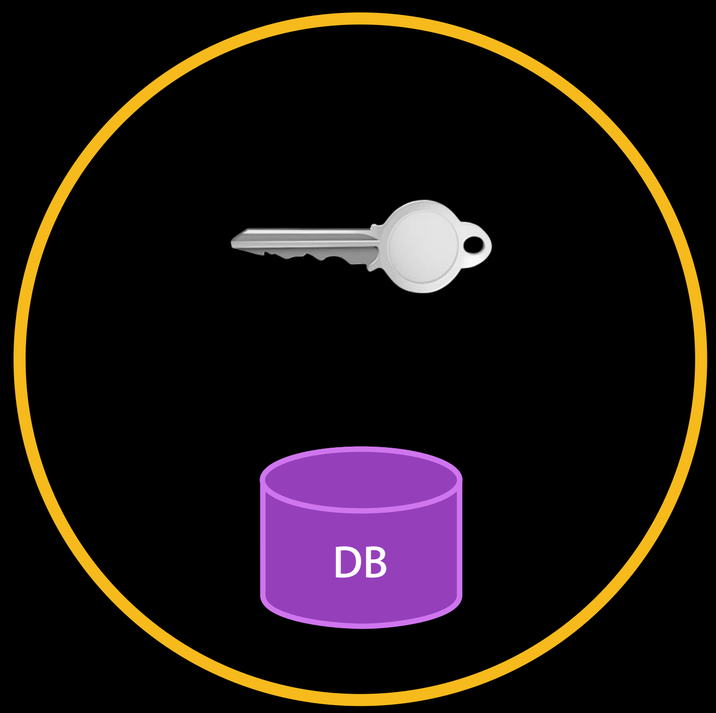
Complete Protection
Data accessible while unlocked



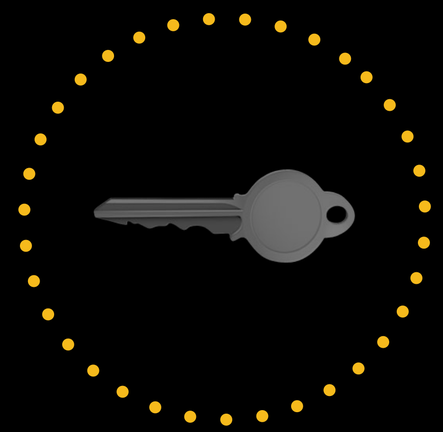
Partial Protection
Protects data from reboot until first unlock

Data Protection

How to handle data while running in the background



Complete Protection
Data accessible while unlocked

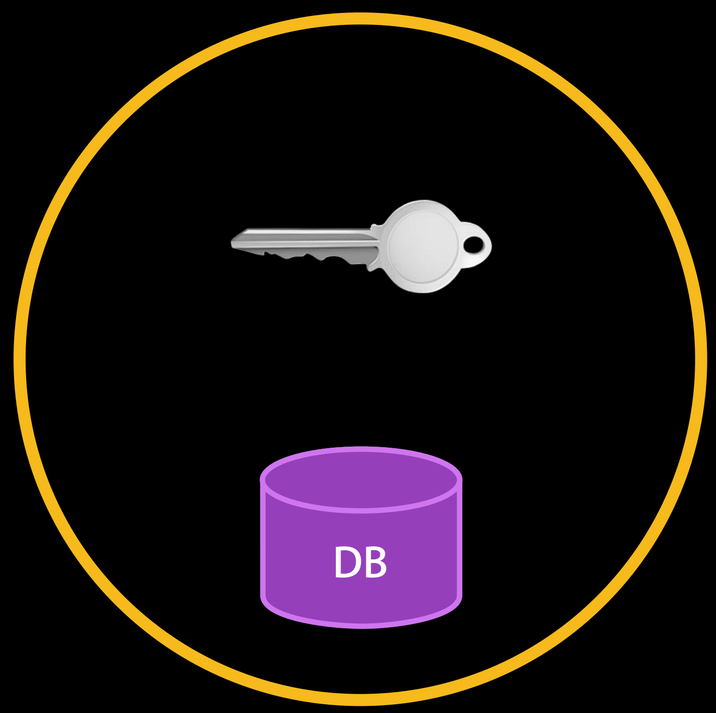


Partial Protection
Protects data from reboot until first unlock

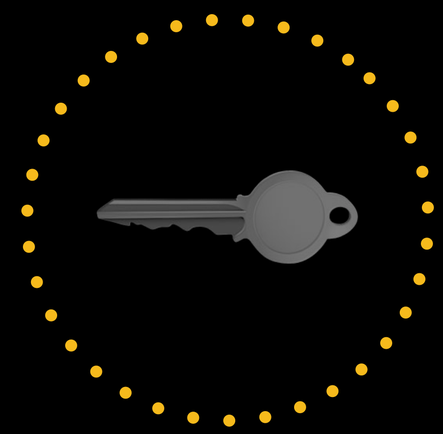
Partial Access
For example, read-only and expires within one week

Data Protection

How to handle data while running in the background



Complete Protection
Data accessible while unlocked



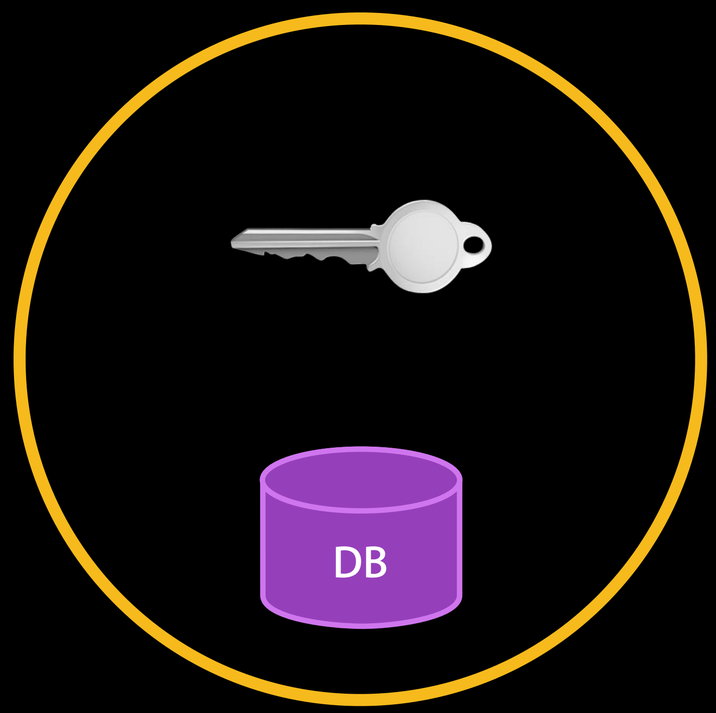
Partial Protection
Protects data from reboot until first unlock

Partial Access
For example, read-only and expires within one week

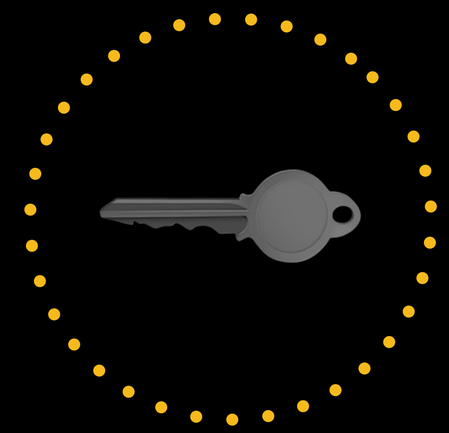
Protected While Open
Download new data which is secured when closed

Data Protection

How to handle data while running in the background

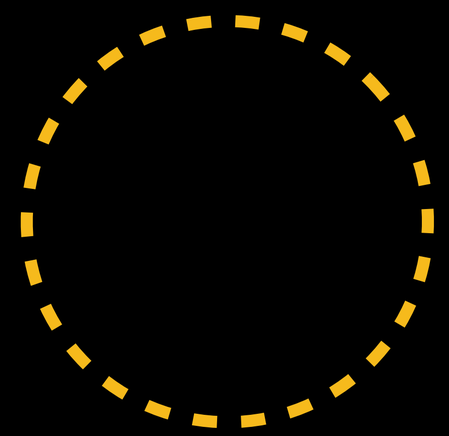


Complete Protection
Data accessible while unlocked



Partial Protection
Protects data from reboot until first unlock

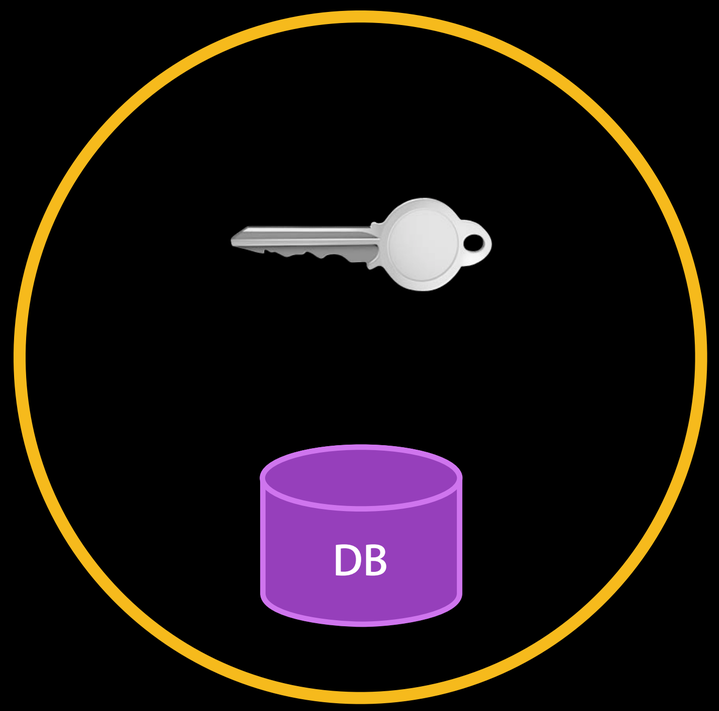
Partial Access
For example, read-only and expires within one week



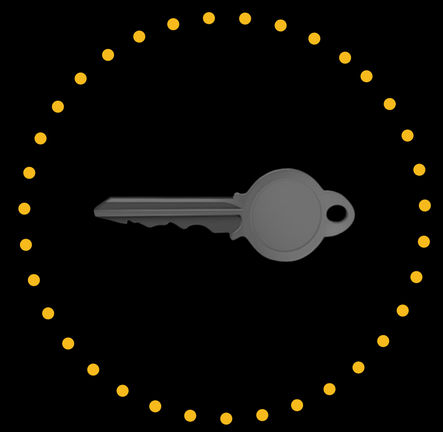
Protected While Open
Download new data which is secured when closed

Data Protection

How to handle data while running in the background

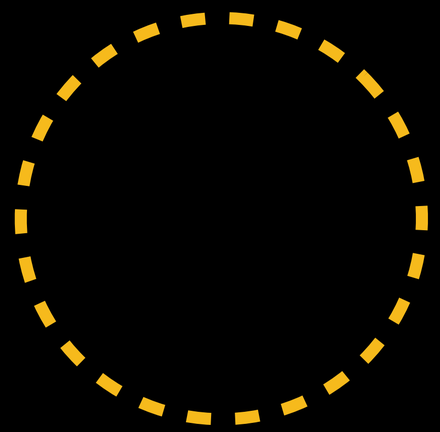


Complete Protection
Data accessible while unlocked



Partial Protection
Protects data from reboot until first unlock

Partial Access
For example, read-only and expires within one week



Protected While Open
Download new data which is secured when closed

Merge When Accessible
Merge new data into main database when appropriate

Efficient Battery Life and Cellular Data Usage

Coalesce and minimize usage

- Minimize cellular data usage
  - Prior to calling completion handler, only download what's necessary for a fresh UI—thumbnails instead of full images
  - Enqueue the rest as a background transfer
- For power efficiency, bring the radios back down as quickly as possible
  - Parallelize network activity as much as possible
  - Avoid using location, motion, or other hardware if unneeded
  - Call the completion handler when complete

App Switcher

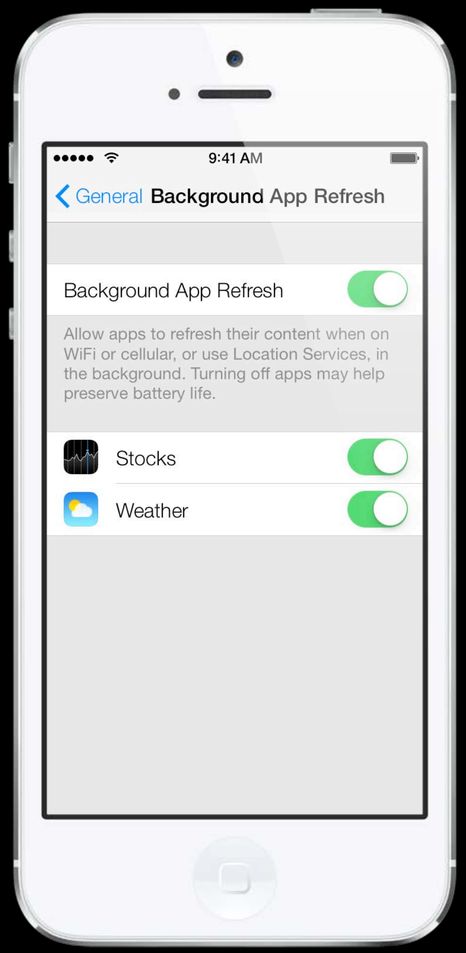
User control of background activity

- When removed from App Switcher, app will no longer run
- Until launched by the user

Background App Refresh Settings

User control of background activity

- Users can configure background activity from Settings
- Apps can be disabled individually
- API coming soon to inspect settings
- Like Newsstand and Location



Multitasking in iOS 7

Multitasking in iOS 7

Background Task Completion

Background Audio

Location Services

VoIP

Newsstand

Multitasking in iOS 7

Background Task Completion

Background Audio

Location Services

VoIP

Newsstand

Background Fetch

Multitasking in iOS 7

Background Task Completion

Background Audio

Location Services

VoIP

Newsstand

Background Fetch

Remote Notifications

Multitasking in iOS 7

Background Task Completion

Background Audio

Location Services

VoIP

Newsstand

Background Fetch

Remote Notifications

Background Transfer Service

More Information

Paul Marcos
Integration Technologies Evangelist
pmarcos@apple.com

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Documentation
iOS Application Programming Guide
http://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/

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

Related Sessions

What's New in Foundation Networking	Mission Wednesday 9:00AM
Protecting Secrets with the Keychain	Marina Wednesday 11:30AM
Energy Best Practices	Marina Thursday 10:15AM
What's New in Core Location	Presidio Thursday 11:30AM
Improving Performance and Energy Usage with Instruments	Nob Hill Thursday 11:30AM
What's New in State Restoration	Mission Thursday 3:15PM
Protecting your Users' Privacy	Mission Friday 9:00AM

Labs

Multitasking Lab	Services Lab Today 3:15PM
iOS Power Efficiency Lab	Frameworks Lab Wednesday 9:00AM
Foundation Networking Lab	Core OS Lab Wednesday 10:15AM
Push Notifications Lab	Services Lab Wednesday 11:30AM
Keychain and Data Protection Security Lab	Core OS Lab Wednesday 4:30PM
Multitasking Lab	Services Lab Thursday 9:00AM
State Restoration Lab	Frameworks Lab Thursday 4:30PM

WWDC2013