Advances in Networking

Part 2

Session 709

Jeffrey Twu, Apple CFNetwork Engineer
Jeff Jenkins, Apple CFNetwork Engineer
Stuart Cheshire, Apple DEST
Advances in Networking
Part 2

URLSession Adaptable Connectivity API
URLSessionTask Scheduling API
URLSession enhancements
Best practices
Ongoing developments
URLSession
Adaptable Connectivity API
Introduction

NSURLSession

Easy-to-use API for networking

• Emphasis on URL loading

Supported on all Apple platforms

Replacement for deprecated NSURLConnection API

What’s New in Foundation Networking

<table>
<thead>
<tr>
<th>WWDC 2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Foundation Networking</td>
</tr>
<tr>
<td>WWDC 2014</td>
</tr>
</tbody>
</table>
URLSession and Connectivity

URLSession with defaultSessionConfiguration fetches or fails

Lack of connectivity causes URLSessionTasks to immediately fail with errors
• NSURLErrorNotConnectedToInternet
• NSURLErrorCannotConnectToHost

Background URLSession has built-in support for monitoring connectivity
Unsatisfactory Connectivity

Examples

No Ethernet cable, not connected to Wi-Fi network, no cellular signal

Device in Airplane Mode

Only cellular connectivity, but allowsCellularAccess prohibits cellular

VPN not connected
Current Solutions

Each app must manually retry URLSessionTasks once connectivity is satisfactory. When is that?

- Monitor conditions with SCNetworkReachability API
- Polling/manual retry

Current mechanisms cannot guarantee connection establishment will succeed.
Wouldn’t it be easier to say...

“Please fetch me this resource when the network is available.”
URLSession Adaptable Connectivity API
Built-in connectivity monitoring

Indicates URLSession should monitor network conditions and wait to start tasks

Begins network load once connectivity is satisfactory instead of delivering errors

• No longer a need to monitor connectivity and manually retry requests

New URLSessionConfiguration property

```swift
var waitsForConnectivity: Bool
```

• Not necessary for background URLSession (does this automatically)
URLSession Adaptable Connectivity API

Insufficient connectivity notification

Notification that a URLSessionTask is waiting for connectivity before starting

Opportunity to alter app behavior or indicate status

New URLSessionTaskDelegate method

```swift
urlSession(_:taskIsWaitingForConnectivity:)
```

- Optional—not required to take advantage of adaptable connectivity functionality
- Called at most one time for each URLSessionTask
URLSession Adaptable Connectivity API
When to enable it

No downside—if connectivity is available, tasks will start right away

General recommendation

Always enable `waitsForConnectivity`

Exception

Requests that must be completed immediately, or not at all for example, “Fill or Kill” stock trading transaction
**URLSession Adaptable Connectivity API**

What to expect

Create and resume URLSessionTask

If insufficient connectivity

`urlSession(_:taskIsWaitingForConnectivity:)` called (if implemented)

URLSession waits until connectivity is satisfactory

Existing URLSessionDelegate methods/completion handler called, just as before
// URLSession Adaptable Connectivity API
// Example 1: Enabling adaptive connectivity

let config = URLSessionConfiguration.default
config.waitsForConnectivity = true

let session = URLSession(configuration: config)
let url = URL(string: "https://www.example.com/")!

let task = session.dataTask(with: url) { (data: Data?, response: URLResponse?, error: Error?) in
    ...
}

task.resume()
Maintain Robustness to Failures

Adaptable connectivity applies to establishing new connections.

Network and server problems can still occur once connected, causing failures.

- NSURLLErrorConnectionLost
- NSURLLErrorTimedOut

Application-specific logic must determine resolution.

- Refer to Technical Q&A QA1941 on Apple Developer website
  Handling “The network connection was lost” Errors
Recap

Polling for network connectivity is prone to problems

Avoid retrying URLSessionTasks due to lack of network connectivity

Let URLSession do the work

• Monitors network conditions for you
• Begins loading your URLSessionTask once connectivity is satisfactory
URLSessionTask Scheduling API

Jeff Jenkins, Apple CFNetwork Engineer
Introduction
Background URLSession

Uploads and downloads continue while your app is not running

System monitors conditions (for example, network, battery, etc.) for you

App launched
• When delegate response is required
• When tasks are complete
Background App Refresh

What is it?

Need data from network to present fresh information to user
- Stock prices, flight status
- News, social network feed
- Weather forecast

Applies to apps and watchOS complications

<table>
<thead>
<tr>
<th>What’s New with Multitasking</th>
<th>WWDC 2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keeping Your Watch App Up to Date</td>
<td>WWDC 2016</td>
</tr>
</tbody>
</table>
Background App Refresh in Action

- Request background launch for information refresh
- New URLSessionTask in background URLSession
- URLSessionTask complete
- User launches app and sees fresh information
Room for Improvement

Extra background launch just to create a URLSessionTask to fetch future data
• Extra launch impacts battery life

Context may change between request creation and start of networking
• Stale request wastes network data

System lacks information about your task to know the best time to schedule it
Indicate the desired start time of a URLSessionTask

New URLSessionTask property `var earliestBeginDate: Date?`

- Guaranteed that task will not begin networking earlier than this
- Only applicable to background URLSession

```
resume()   earliestBeginDate   timeoutInterval
Waiting    Eligible          Timeout
```

NEW
Background App Refresh in Action

Original workflow

- Request background launch for information refresh
- New URLSessionTask in background URLSession
- URLSessionTask complete
- User launches app and sees fresh information

App State
- Running
- Suspended
- Background
Background App Refresh in Action

Improved workflow

User launches app and sees fresh information

New URLSessionTask with earliestBeginDate

URLSessionTask complete

User launches app and sees fresh information

App State:
- Running
- Suspended
- Background
Opportunity to alter future request when system is ready to begin networking

New URLSessionTaskDelegate method

```swift
urlSession(_:task:willBeginDelayedRequest:request:completionHandler:)
```

- Only called for tasks with `earliestBeginDate` set
- Background URLSession only
- Optional—not required to take advantage of URLSessionTask scheduling
- Completion handler—proceed, change request (URL and headers), or cancel
Background App Refresh in Action

Advanced workflow

New URLSessionTask with earliestBeginDate

Change URLSessionTask before starting network load?

URLSessionTask complete

User launches app and sees fresh information

Time

App State

- Running
- Suspended
- Background
URLSessionTask Scheduling API

Indicate estimated transfer size of each URLSessionTask

Allows better background task scheduling by the system

Two new URLSessionTask properties:

```swift
var countOfBytesClientExpectsToSend: Int64
var countOfBytesClientExpectsToReceive: Int64
```

Provide “best guess” (approximate upper bound)

or

```swift
NSURLESessionTransferSizeUnknown
```
// URLSessionTask Scheduling API
// Example 1: Scheduling a background task to start no earlier than 2 hours in the future

let config = URLSessionConfiguration.background(withIdentifier: "...")
let session = URLSession(configuration: config, delegate: ..., delegateQueue: ...)

var request = URLRequest(url: URL(string: "https://www.example.com/"))!
request.addValue("...", forHTTPHeaderField: "...")

let task = session.downloadTask(with: request)

// Indicate desired scheduling
task.earliestBeginDate = Date(timeIntervalSinceNow: 2 * 60 * 60)

// Request is small (no body, one added header) and response is ~2 KiB
task.countOfBytesClientExpectsToSend = 80
task.countOfBytesClientExpectsToReceive = 2048

task.resume()}
// URLSessionTask Scheduling API
// Example 2: Altering HTTP request headers to avoid a stale request

func urlSession(_ session: URLSession, task: URLSessionTask, willBeginDelayedRequest request: URLRequest, completionHandler: @escaping (URLSession.DelayedRequestDisposition, URLRequest?) -> Void) {

    var updatedRequest = request
    updatedRequest.addValue("...", forHTTPHeaderField: "...")

    completionHandler(.useNewRequest, updatedRequest)
}

Recap

Background URLSession allows apps to upload and download when not running

New URLSessionTask scheduling API gives you control
• Delay tasks to when you need them for the freshest information
• Opportunity to alter tasks before network load begins to avoid stale requests

Help us deliver the best customer experience
• Specify expected byte counts for every URLSessionTask
URLSession Enhancements

Stuart Cheshire, Apple DEST
URLSession Enhancements

ProgressReporting
Brotli compression
Public Suffix List updates
URLSessionStreamTask and authenticating proxies
URLSessionTask Progress Tracking
Old API for progress calculation

Extra work for URLSession API clients

Need Key-value Observing setup for

- countOfBytesExpectedToReceive,
- countOfBytesReceived
- countOfBytesExpectedToSend,
- countOfBytesSent

Not always available

- countOfBytesExpectedToReceive
- countOfBytesExpectedToSend
URLSessionTask Adopts ProgressReporting
Improved API for progress calculation

Implements ProgressReporting protocol

```swift
class URLSessionTask : NSObject, NSCopying, ProgressReporting

public var progress: Progress { get }
```
URLSessionTask Adopts ProgressReporting

Improved API for progress calculation

URLSessionTask overall work completed

```swift
var fractionCompleted: Double in [0.0, 1.0]
```

General and more specific progress description

```swift
var localizedDescription: String!
var localizedAdditionalDescription: String!
```

Can attach Progress object to a UIProgressView or NSProgressIndicator

Progress of multiple tasks by using a parent progress object

Key-value observing and Cocoa bindings
URLSessionTask Adopts ProgressReporting

URLSessionTask and Progress object state management

Progress state management methods change URLSessionTask state

URLSessionTask

- func resume()
- func suspend()
- func cancel()

Progress

- func resume()
- func pause()
- func cancel()
URLSession Brotli Support

RFC 7932 “Brothli Compressed Data Format”

Content-Encoding: br

Faster URL loads
• Median 15% improvement in compressed sizes
  versus gzip for text-based assets (HTML, JS, CSS, ...)

Requires HTTPS (TLS)
URLSession Public Suffix List
Effective top level domain list

Public Suffix List
• https://publicsuffix.org

Heuristic to determine administrative boundaries
• “apple.com” is one organization
• “com.au” is many organizations
URLSession Public Suffix List Updates

Effective top level domain list

URLSession can now receive updates over the air

Update can be pushed biweekly (or even more frequently) depending on the number of TLDs added to the list

Better security for users against cookie attacks

• URLSession APIs
• HTTPCookieStorage
URLSessionStreamTask

Allows for direct TCP/IP connection to a host and port

Optional secure handshaking (STARTTLS)

Ability to convert to legacy NSInputStream/NSOutputStream

For new code we recommend using native URLSessionStreamTask APIs

Navigation of authenticating HTTPS proxies
URLSession Enhancements

ProgressReporting

Brotli compression

Public Suffix List updates

URLSessionStreamTask and authenticating proxies
Networking Best Practices

Don’t use BSD sockets

Don’t embed networking libraries

Do use Apple’s APIs to get benefits of future improvements
  • Wi-Fi Assist
  • Power efficiency
  • Discretionary/background work

Do use connect-by-name APIs
Networking Best Practices

URLSession timers

```swift
var timeoutIntervalForResource: TimeInterval
Fires if entire resource not received in time
```

```swift
var timeoutIntervalForRequest: TimeInterval
Once started, fires if no forward progress being made
```
Networking Best Practices

URLSession usage

Generally one URLSession per app

Multiple concurrent URLSessionTasks can share single URLSession

Clean up any dynamic URLSession objects that you create
  • finishTasksAndInvalidate
  • invalidateAndCancel
Networking Best Practices
Convenience methods and delegate callbacks

Delegate callbacks
• Intermediate progress reported to the delegate object

Convenience methods
• Final outcome reported to completionHandler

Don’t use both on the same URLSession
• If using completionHandler, no delegate callbacks delivered
• Two exceptions: taskIsWaitingForConnectivity
didReceiveAuthenticationChallenge
## URLSession Best Practices

Impact of URLSessionConfiguration and loading control properties

<table>
<thead>
<tr>
<th>Default and Ephemeral Configuration + waitsForConnectivity</th>
<th>Background Configuration</th>
<th>Background Configuration + discretionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-process</td>
<td>Out-of-process</td>
<td>Out-of-process</td>
</tr>
<tr>
<td>No retry</td>
<td>Automatic retry until timeoutIntervalForResource</td>
<td>Automatic retry until timeoutIntervalForResource</td>
</tr>
<tr>
<td>Delegate + convenience</td>
<td>Delegate only</td>
<td>Delegate only</td>
</tr>
<tr>
<td>Tasks start immediately If fails, will call taskIsWaitingForConnectivity and automatically retry as necessary</td>
<td>Tasks will consider connectivity, power, etc.</td>
<td>Scheduled for optimal system performance</td>
</tr>
</tbody>
</table>

**Application loading requirements**

- **More urgent**
- **Less urgent**
Ongoing Developments
TLS 1.3
Transport Layer Security

Update to TLS 1.2

TLS 1.3 standard expected to be finalized by the end of this year

Apple is participating

Draft implementation available for testing now
QUIC
Quick UDP internet connections

End-to-End Transport Protocol, like TCP

Started as Google experiment

Now an IETF Working Group

Specification is making rapid progress, but still far from complete

Apple is participating
15 years
Bonjour Launch
Mac OS X 10.2
Jaguar

2002

Wide-area (Unicast)
Bonjour
Mac OS X 10.4
Tiger

2004

Bonjour for
Windows,
Linux,
Java

2005

Avahi (GPL)
for Linux

2006

Bonjour Sleep Proxy
Mac OS X 10.5
Leopard

2008

Back to My Mac
Mac OS X 10.6
Snow Leopard

2010

Windows 10 adds native
Bonjour (DNS-SD) APIs

2014

Bonjour in Android 4.1
"Jelly Bean"
(API Level 16)

2015

2016

2017
# Input Settings

<table>
<thead>
<tr>
<th>Input</th>
<th>BD/DVD</th>
<th>SAT/CATV</th>
<th>GAME</th>
<th>VIDEO1</th>
<th>VIDEO2</th>
<th>TV</th>
<th>SA-CD/CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icon</td>
<td>BD/DVD</td>
<td>SAT/CATV</td>
<td>GAME</td>
<td>VIDEO1</td>
<td>VIDEO2</td>
<td>TV</td>
<td>SA-CD/CD</td>
</tr>
<tr>
<td>Name</td>
<td>BD/DVD</td>
<td>SAT/CATV</td>
<td>GAME</td>
<td>VIDEO1</td>
<td>VIDEO2</td>
<td>TV</td>
<td>SA-CD/CD</td>
</tr>
<tr>
<td>Watch/Listen</td>
<td>Watch</td>
<td>Watch</td>
<td>Watch</td>
<td>Listen</td>
<td>Watch</td>
<td>Watch</td>
<td>Watch</td>
</tr>
<tr>
<td>Show/Hide</td>
<td>Hide</td>
<td>Hide</td>
<td>Hide</td>
<td>Hide</td>
<td>Hide</td>
<td>Hide</td>
<td>Hide</td>
</tr>
<tr>
<td>OPT/COAX</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Input Mode</td>
<td>Auto</td>
<td>Auto</td>
<td>Auto</td>
<td>Auto</td>
<td>Auto</td>
<td>Auto</td>
<td>Auto</td>
</tr>
</tbody>
</table>

Copyright 2014 Sony Corporation
SUNNY BOY 6.0

Device status
Ok

Current power
287 W

Yield:
Today: 35.37 kWh
Yesterday: 0.000 kWh
Total: 7.4519 kWh

Feed-in management
The active power limitation is deactivated.

Current power
Day Month Year Total
May 29, 2017

PV power
Bonjour
Continued development

IETF DNS Service Discovery (DNSSD) Working Group
• Enhancements for enterprise and mesh networks

For app developers
• No change to APIs
• Remember that browse results might not be “local”

For device makers
• Remember to support IPv6 link-local addressing
Summary
Part 1

Explicit Congestion Notification
• Supported in clients and server—the stage is set for network adoption

Continue testing your apps on a NAT64 network
• Update your servers to native IPv6

User-space networking

NEHotspotConfiguration, NEDNSProxyProvider

Multipath protocols for multipath devices
Summary
Part 2

URLSession enhancements
• `waitsForConnectivity`
• `ProgressReporting`
• `Public Suffix List`

• `earliestBeginDate`
• `Brotli compression`
• `URLSessionStreamTask`

Best Practices

Ongoing developments—TLS 1.3, QUIC, Bonjour
More Information

Part 1

Part 2
<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your Apps and Evolving Network Security Standards</td>
<td>WWDC 2017</td>
<td></td>
</tr>
<tr>
<td>Privacy and Your Apps</td>
<td>WWDC 2017</td>
<td></td>
</tr>
<tr>
<td>Advances in HTTP Live Streaming</td>
<td>WWDC 2017</td>
<td></td>
</tr>
<tr>
<td>What's New in HomeKit</td>
<td>WWDC 2017</td>
<td></td>
</tr>
<tr>
<td>What's New in Safari View Controller</td>
<td>Executive Ballroom</td>
<td>Thursday 10:00AM</td>
</tr>
<tr>
<td>What's New in Device Configuration, Deployment, and Management</td>
<td>Grand Ballroom B</td>
<td>Thursday 1:50PM</td>
</tr>
<tr>
<td>Labs</td>
<td>Technology Lab</td>
<td>Time</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Networking Lab</td>
<td>Technology Lab D</td>
<td>Thu 9:00AM-11:00AM</td>
</tr>
<tr>
<td>Networking Lab</td>
<td>Technology Lab J</td>
<td>Fri 1:50PM-3:50PM</td>
</tr>
</tbody>
</table>