

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

URLSession

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

WWDC 2013

What's New in Foundation Networking

WWDC 2014

URLSession and Connectivity

URLSession with defaultSessionConfiguration fetches or fails

Lack of connectivity causes URLSessionTasks to immediately fail with errors

- NSErrorNotConnectedToInternet
- NSErrorCannotConnectToHost

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

NEW

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 `var waitsForConnectivity: Bool`

- Not necessary for background URLSession (does this automatically)

URLSession Adaptable Connectivity API

Insufficient connectivity notification

NEW

Notification that a URLSessionTask is waiting for connectivity before starting

Opportunity to alter app behavior or indicate status

New URLSessionTaskDelegate method

```
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 `waitForConnectivity`

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

- NSErrorConnectionLost
- NSErrorTimedOut

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

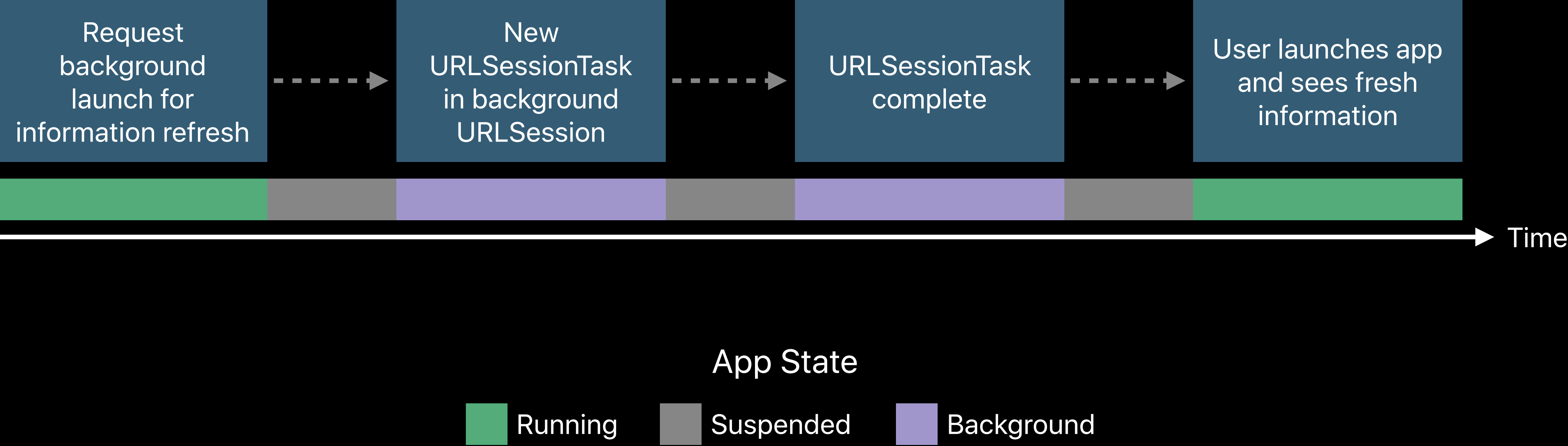
What's New with Multitasking

WWDC 2013

Keeping Your Watch App Up to Date

WWDC 2016

Background App Refresh in Action



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

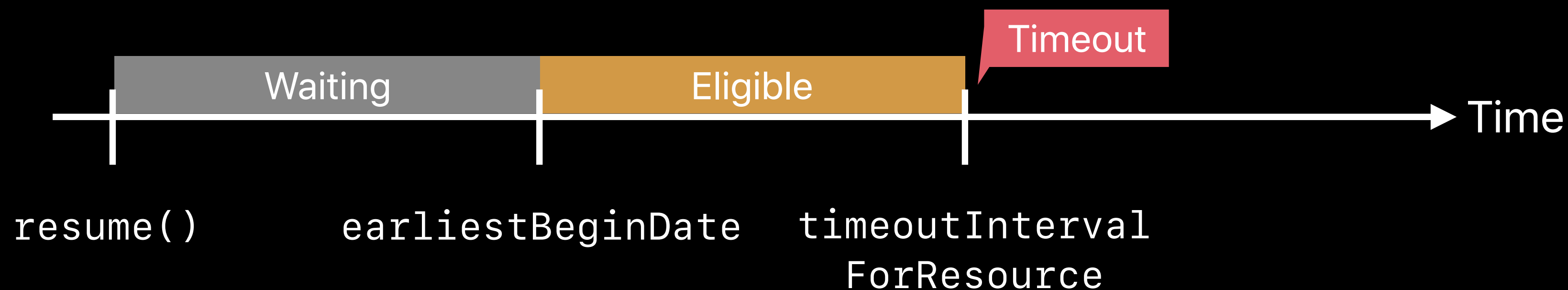
URLSessionTask Scheduling API

NEW

Indicate the desired start time of a URLSessionTask

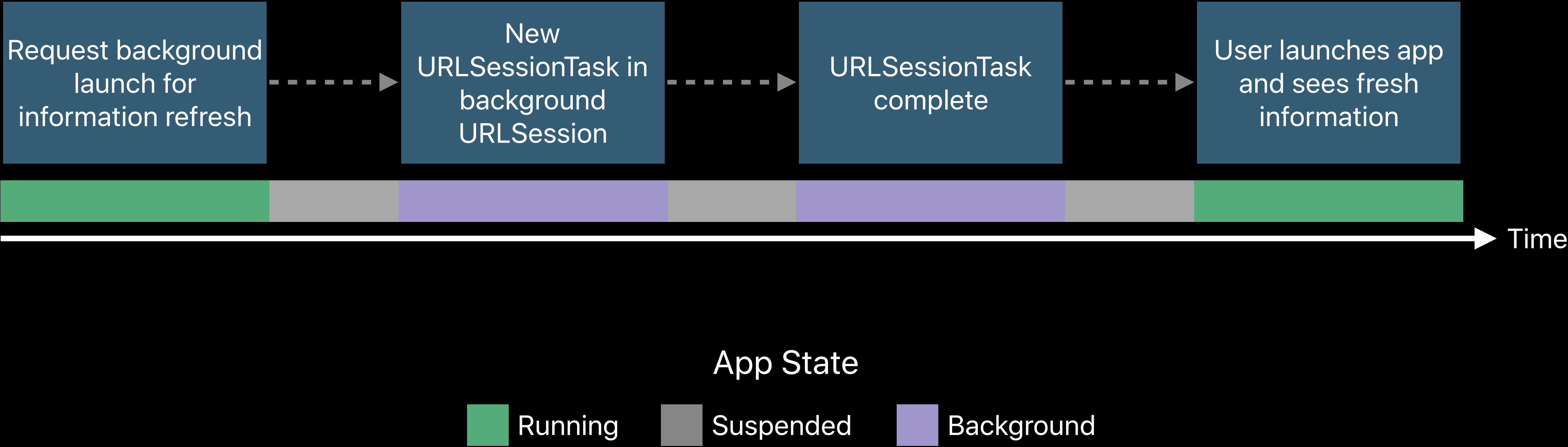
New URLSessionTask property `var earliestBeginDate: Date?`

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



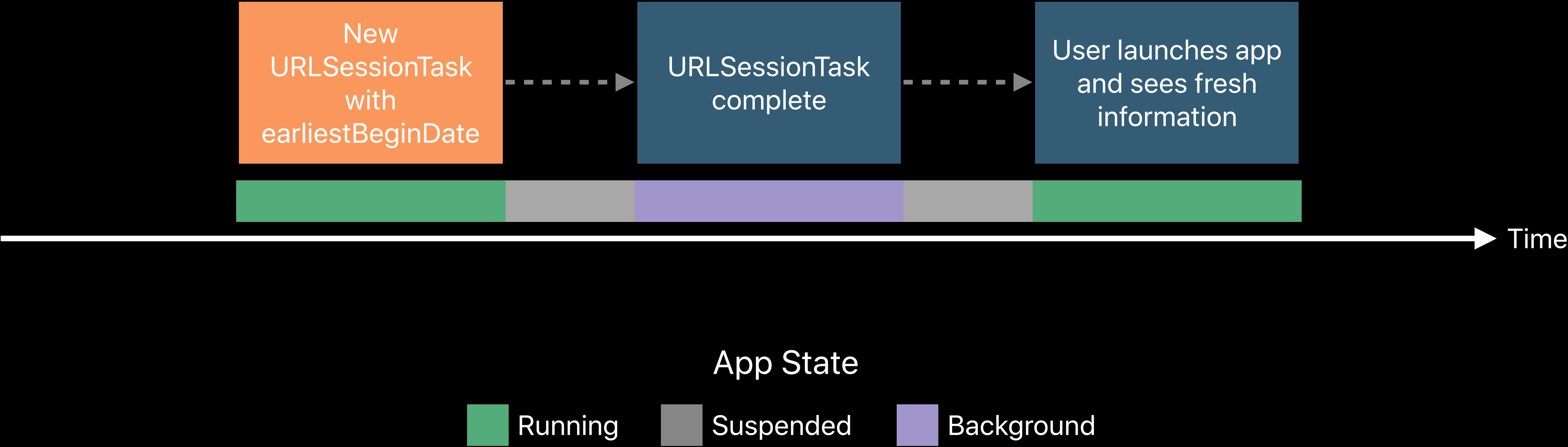
Background App Refresh in Action

Original workflow



Background App Refresh in Action

Improved workflow



URLSessionTask Scheduling API

NEW

Opportunity to alter future request when system is ready to begin networking

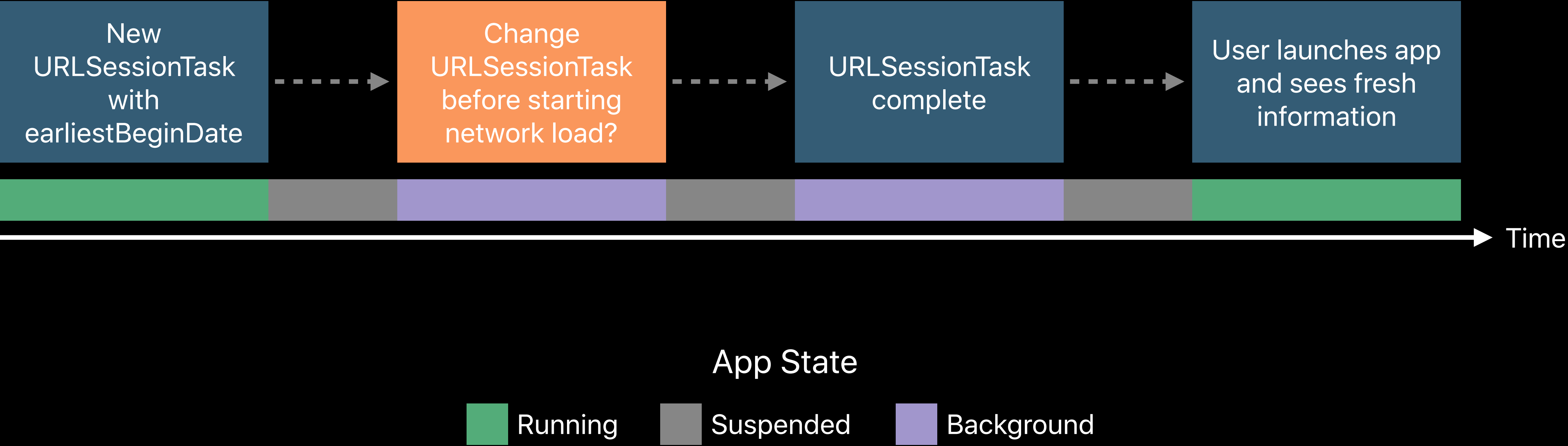
New URLSessionTaskDelegate method

```
urlSession(_:task:willBeginDelayedRequest: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



URLSessionTask Scheduling API

NEW

Indicate estimated transfer size of each URLSessionTask

Allows better background task scheduling by the system

Two new URLSessionTask properties:

```
var countOfBytesClientExpectsToSend: Int64
```

```
var countOfBytesClientExpectsToReceive: Int64
```

Provide “best guess” (approximate upper bound)

or `NSURLSessionTransferSizeUnknown`

```
// 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

<code>countOfBytesExpectedToReceive,</code>	<code>countOfBytesReceived</code>
<code>countOfBytesExpectedToSend,</code>	<code>countOfBytesSent</code>

Not always available

<code>countOfBytesExpectedToReceive</code>
<code>countOfBytesExpectedToSend</code>

URLSessionTask Adopts ProgressReporting

Improved API for progress calculation

NEW

Implements ProgressReporting protocol

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

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

```
public var progress: Progress { get }
```

URLSessionTask Adopts ProgressReporting

Improved API for progress calculation

URLSessionTask overall work completed

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

General and more specific progress description

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

Can attach Progress object to a UIProgressView or NSProgressIndicator

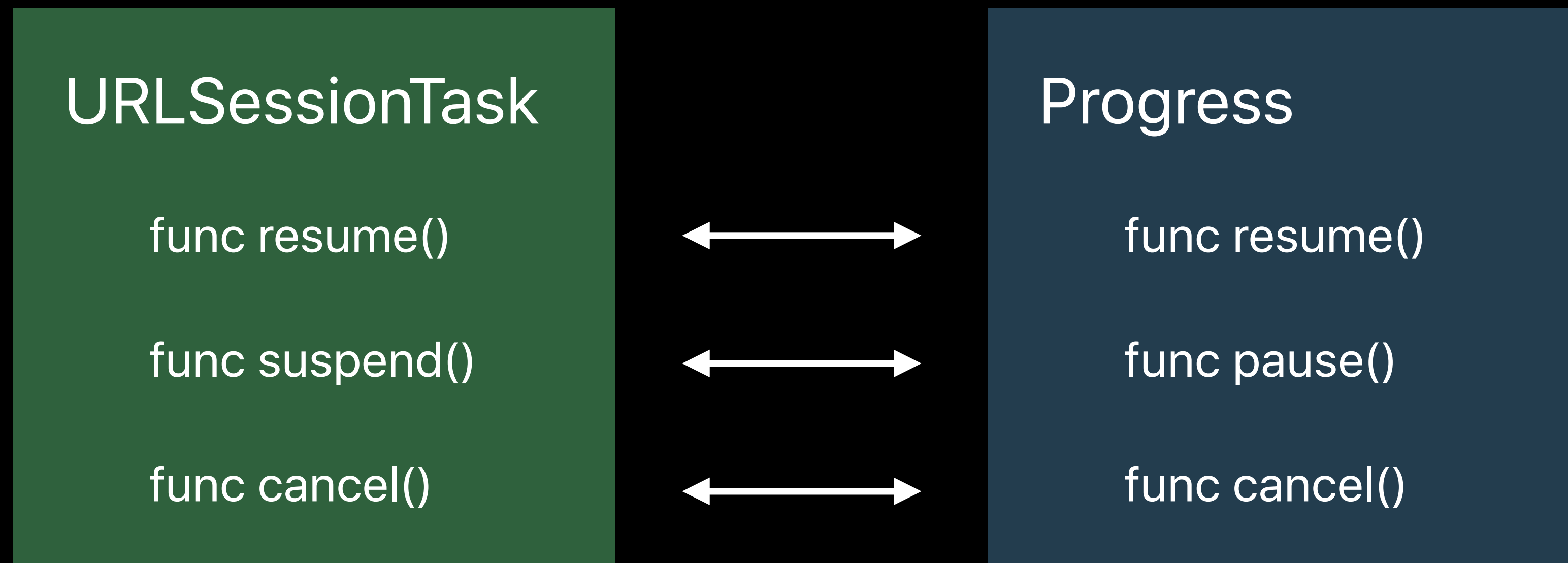
Progress of multiple tasks by using a parent progress object

Key-value observing and Cocoa bindings

URLSessionTask Adopts ProgressReporting

URLSessionTask and Progress object state management

Progress state management methods change URLSessionTask state



URLSession Brotli Support

NEW

RFC 7932 “Brotli 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

NEW

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

NSURLSessionTask

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 URLSessionTask APIs

Navigation of authenticating HTTPS proxies

NEW

URLSession Enhancements

ProgressReporting

Brotli compression

Public Suffix List updates

URLSessionStreamTask and authenticating proxies

Networking Best Practices

Tips to remember

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

```
var timeoutIntervalForResource: TimeInterval
```

Fires if entire resource not received in time

```
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

Default and Ephemeral Configuration + waitsForConnectivity	Background Configuration	Background Configuration + discretionary
In-process	Out-of-process	Out-of-process
No retry	Automatic retry until timeoutIntervalForResource	Automatic retry until timeoutIntervalForResource
Delegate + convenience	Delegate only	Delegate only
Tasks start immediately If fails, will call tasksWaitingForConnectivity and automatically retry as necessary	Tasks will consider connectivity, power, etc.	Scheduled for optimal system performance
More urgent	Less urgent	
Application loading requirements		

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

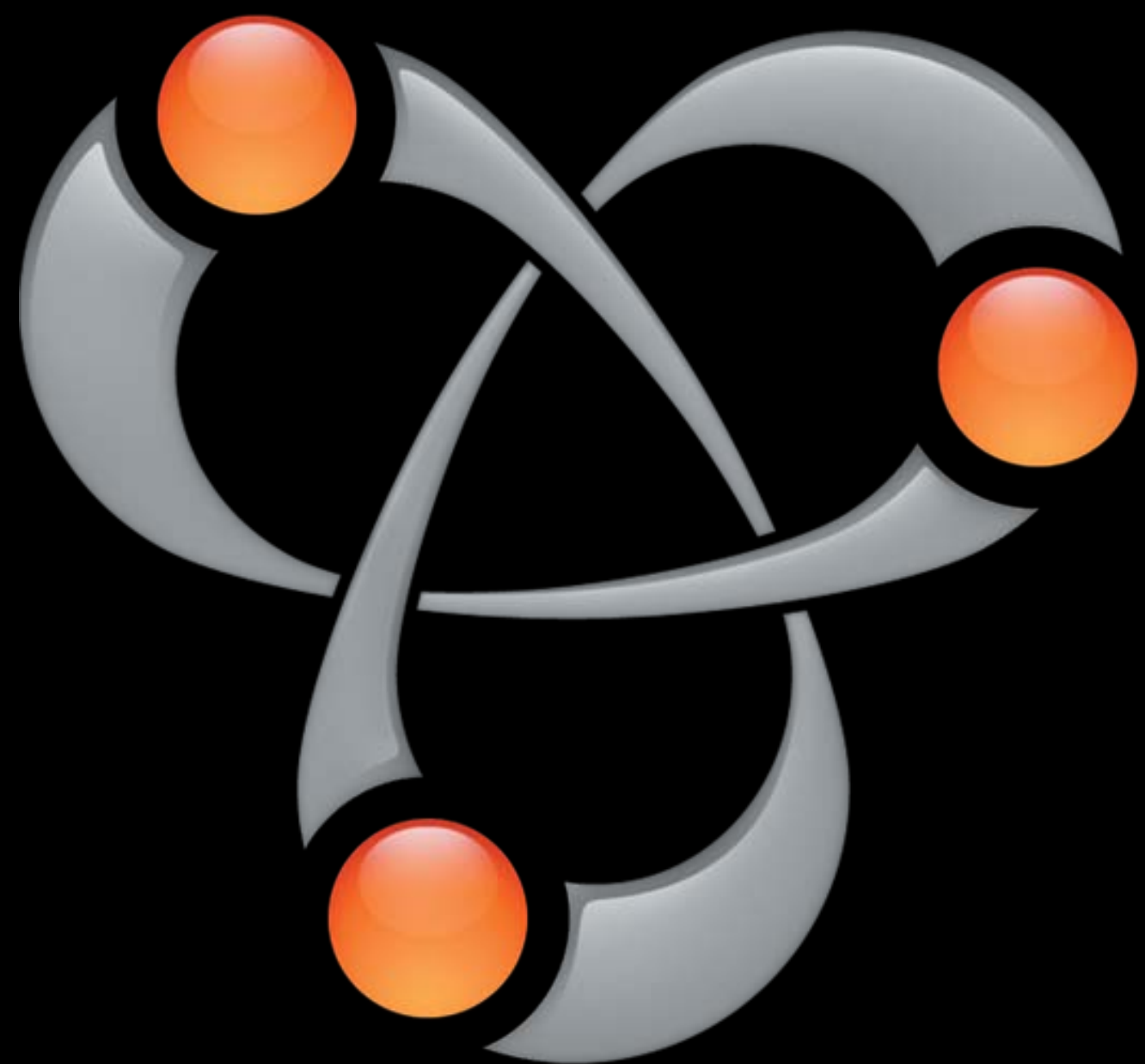
End-to-End Transport Protocol, like TCP

Started as Google experiment

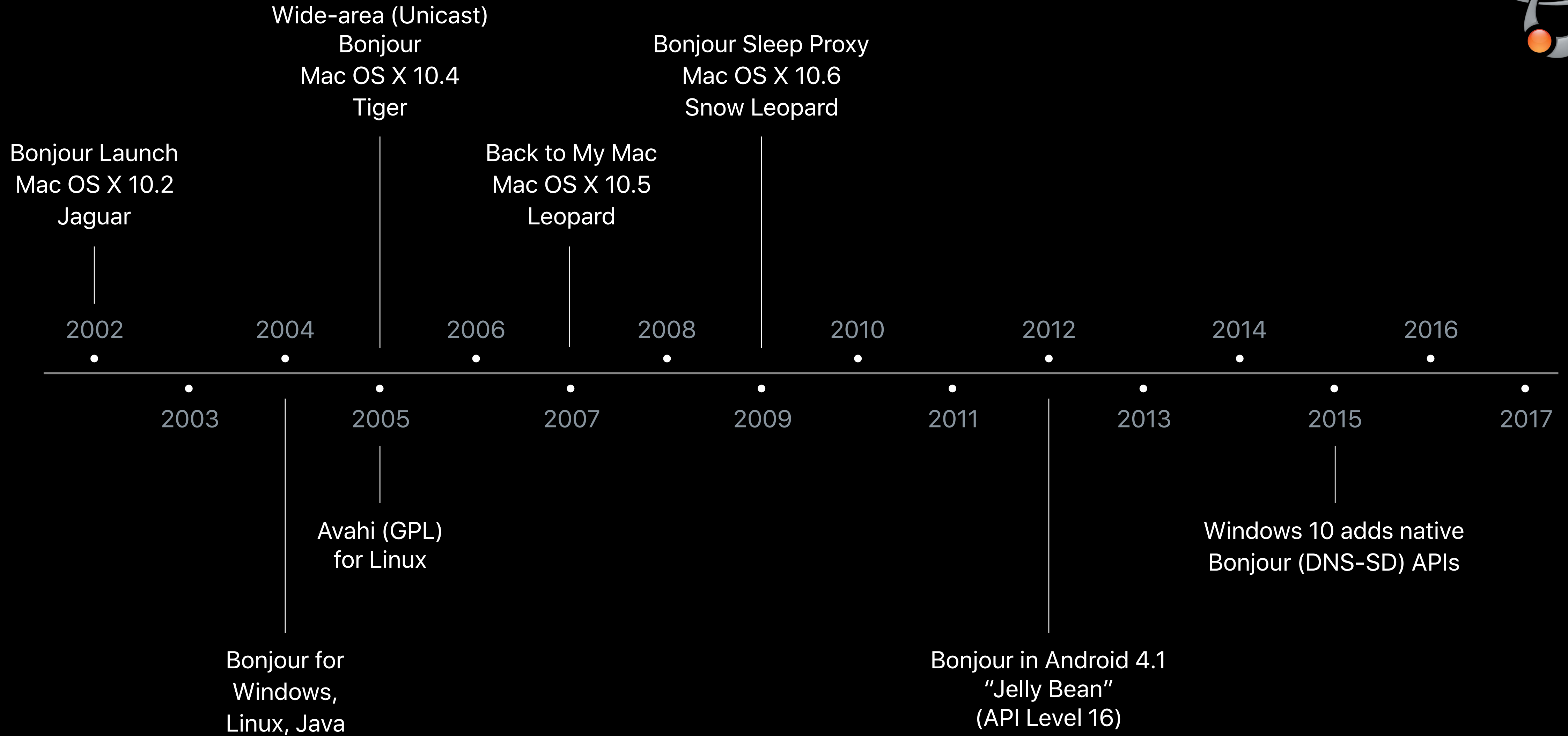
Now an IETF Working Group

Specification is making rapid progress, but still far from complete

Apple is participating



15 years



chrome.cast.local

Bonjour Apple iCloud Yahoo Bing Google Wikipedia Facebook Twitter LinkedIn The Weather Channel Yelp TripAdvisor

STR-DN1060

Software Version: M29.R.0426
Destination: UC

Return to Default

Speaker

Audio

Screen

HDMI

Input

Network

Bluetooth

Zone

System

Tuner

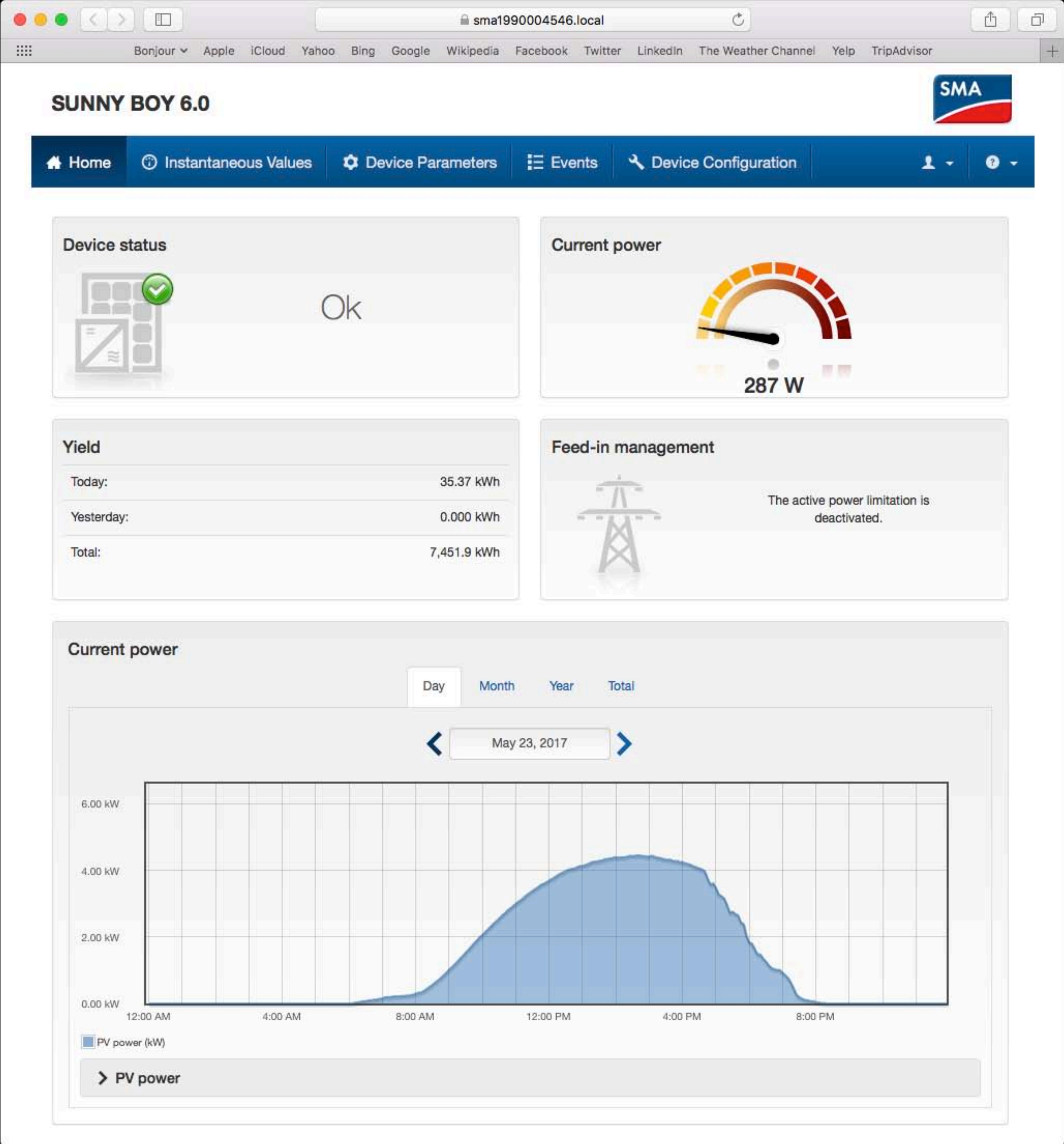
Custom Preset

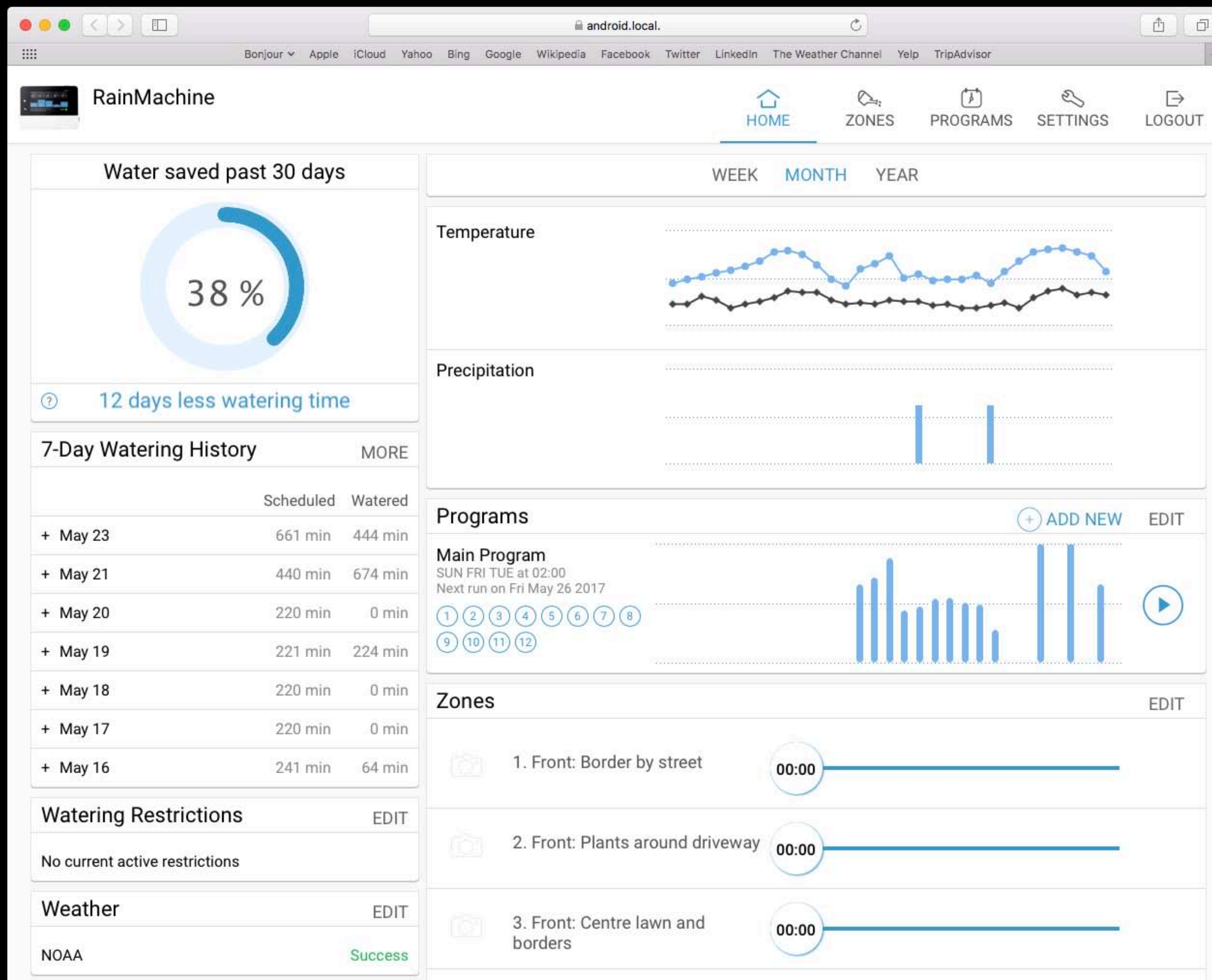
AirPlay

Input Settings

Input	BD/DVD	SAT/CATV	GAME	VIDEO1	VIDEO2	TV	SA-CD/CD
Icon							
	BD/DVD	STB	TV	CD	MOBILE	TV	CD
Name	BD/DVD	TiVo	Mac	iPod	VIDEO2	TV	AppleTV
Watch/Listen	Watch	Watch	Watch	Listen	Watch	Watch	Watch
Show/Hide	Hide	Hide	Hide	Show	Hide	Hide	Hide
OPT/COAX	None	None	None	None	None	OPT 2	COAX
COMPONENT	None	None	None	None	None		None
Input Mode		Auto		Auto		Auto	Auto

Copyright 2014 Sony Corporation





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

- `waitForConnectivity`
- `ProgressReporting`
- `Public Suffix List`
- `earliestBeginDate`
- `Brotli compression`
- `URLSessionStreamTask`

Best Practices

Ongoing developments—TLS 1.3, QUIC, Bonjour

More Information

Part 1

<https://developer.apple.com/wwdc17/707>

Part 2

<https://developer.apple.com/wwdc17/709>

Related Sessions

Your Apps and Evolving Network Security Standards	WWDC 2017	
Privacy and Your Apps	WWDC 2017	
Advances in HTTP Live Streaming	WWDC 2017	
What's New in HomeKit	WWDC 2017	
What's New in Safari View Controller	Executive Ballroom	Thursday 10:00AM
What's New in Device Configuration, Deployment, and Management	Grand Ballroom B	Thursday 1:50PM

Labs

Networking Lab	Technology Lab D	Thu 9:00AM-11:00AM
Networking Lab	Technology Lab J	Fri 1:50PM-3:50PM

