Advances in Networking
Part 1

Josh Graessley, Internet Technologies
Guoye Zhang, Internet Technologies
Jiten Mehta, Internet Technologies
Christoph Paasch, Internet Technologies
Part 1

- Low Data Mode
- Combine in URLSession
- WebSocket
- Mobility Improvements
Part 2

Bonjour

Building Framing Protocols

Collecting Metrics

Best Practices and Status Updates
URLSession
Network.framework
Low Data Mode

Josh Graessley, Internet Technologies
Low Data Mode

User preference to minimize data usage
• Explicit signal to reduce network data use
• Per Wi-Fi and Cellular network

System policy
• Discretionary tasks deferred
• Background App Refresh disabled

Application adoption
Always save network data when there is no impact on user experience.
Application Adoption

- Reduce image quality
- Reduce pre-fetching
- Synchronize less often
- Mark tasks discretionary
- Disable auto-play
- Do not block user-initiated work
Low Data Mode APIs

**URLSession**
- Try large/prefetch with `allowsConstrainedNetworkAccess = false`
- On failure with `error.networkUnavailableReason == .constrained`
  - try Low Data Mode alternative

**Network.framework**
- Set `prohibitConstrainedPaths` on `NWParameters`
- Check `isConstrained` on `NWPath`
- Handle path updates
Constrained and Expensive

Constrained — Low Data Mode

Expensive — Cellular and Personal Hotspot

URLSession
• `allowsExpensiveNetworkAccess`

Network.framework
• Set `prohibitExpensivePaths` on `NWParameters`
• Check `isExpensive` on `NWPath`
Combine in URLSession
.debounce(0.2)

.filter { $0.count >= 3 }

.map(toSearchURL)

.sink()  

https://example.com/search/Hello%20World
What Is Combine

A declarative API for processing values over time

Introducing Combine and Advances in Foundation
Thursday, 10:00

Combine in Practice
Thursday, 2:00
Combine in URLSession

DataTaskPublisher

Single value publisher

Similar to URLSession.dataTask(with:completionHandler:)

```swift
public struct DataTaskPublisher: Publisher {
    public typealias Output = (data: Data, response: URLResponse)
    public typealias Failure = URLError
}
```
Demo
PubSocket Recap

Streamline networking code with Combine

Support retry
- Use low retry count
- Only idempotent request

Best practices for Low Data Mode
// Generalized Publisher for Adaptive URL Loading

func adaptiveLoader(regularURL: URL, lowDataURL: URL) -> AnyPublisher<Data, Error> {

    var request = URLRequest(url: regularURL)
    request.allowsConstrainedNetworkAccess = false
    return URLSession.shared.dataTaskPublisher(for: request)
        .tryCatch { error -> URLSessionDataTaskPublisher in
            guard error.networkUnavailableReason == .constrained else {
                throw error
            }
            return URLSession.shared.dataTaskPublisher(for: lowDataURL)
        }
        .tryMap { data, response -> Data in
            guard let httpResponse = response as? HTTPURLResponse,
            httpResponse.statusCode == 200 else {
                throw MyNetworkingError.invalidServerResponse
            }
            return data
        }
        .eraseToAnyPublisher()
WebSocket
WebSockets

Two-way communication over TLS/TCP connection
Works with Firewalls and CDNs
Proxy support
HTTP/1.1 Long-Polling

- GET
- 200 OK
- Body Data
- GET
- GET
- GET
- GET

- Phone
- Cloud

Message:

O'hello! Thanks for contacting WebsterChat Support. How can I help you?
WebSocket
Messaging using WebSocket

GET
101 Switching Protocols
WebSockets

Messaging using WebSockets
**WebSocket**

**URLSessionWebSocketTask**

Foundation API for WebSocket

Works with existing URLSession

```swift
// Create with URL
let task = URLSession.shared.webSocketTask(with: URL(string: "wss://websocket.example"))!
task.resume()

// Send a message
task.send(.string("Hello")) { error in /* Handle error */ }

// Receive a message
task.receive { result in /* Handle result */ }
```
Both client and server support

Receive partial or complete WebSocket messages

```swift
// Create parameters for WebSocket over TLS
let parameters = NWParameters.tls
let websocketOptions = NWProtocolWebSocket.Options()
parameters.defaultProtocolStack.applicationProtocols.insert(websocketOptions, at: 0)

// Create a connection with those parameters
let websocketConnection = NWConnection(to: endpoint, using: parameters)

// Create a listener with those parameters
let websocketListener = try NWListener(using: parameters)
```
Demo
## PubSocket+ Recap

<table>
<thead>
<tr>
<th>Server</th>
<th>NWListener</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>URLSessionWebSocketTask</td>
</tr>
<tr>
<td>Transport</td>
<td>Bidirectional WebSocket Messages</td>
</tr>
<tr>
<td>Advantage</td>
<td>Less HTTP overhead</td>
</tr>
</tbody>
</table>
Mobility Improvements

Christoph Paasch, Internet Technologies
What is Mobility?

Wi-Fi signal simulation plot courtesy of Avery Pennarun  
<https://apenwarr.ca/log/?m=201408>  <https://apenwarr.ca/beamlab/>
Mobility up to iOS 12

- **iOS 7**: Multipath TCP
- **iOS 9**: Wi-Fi Assist
- **iOS 11**: Multipath API
Mobility Improvements in iOS 13

Multipath Transports

Wi-Fi Assist

Smarter Mobility Detection

Flow Recovery

Apple Maps

Cross-Layer

FaceTime

URLSession

Network.framework

Apple Music

Safari

Multipath TCP

multipathServiceType
Wi-Fi Assist in iOS 13

- Cross-layer Mobility Detection
- Improved Flow Recovery

- URLSession
- Network.framework
- Wi-Fi
- Cellular
- Wi-Fi Assist

NEW
Wi-Fi Assist in iOS 13

Use high-level APIs like URLSession or Network.framework

Rethink SCNetworkReachability usage

Control access with allowsExpensiveNetworkAccess = false
Multipath Transports

- Responsiveness for Maps
- Fewer streaming stalls in Music
Multipath Transports

Multipath Transports for your App

• `multipathServiceType` `NSURLSessionConfiguration` and `Network.framework`

Server-side configuration

• Linux Kernel at https://multipath-tcp.org
Mobility Improvements

Mobility should not impair your Apps

Use high-level APIs

Rethink interface management

Prepare your servers and use `multipathServiceType`
Summary

Low Data Mode
Combine in URLSession
WebSocket
Mobility Improvements
# More Information

devolver.apple.com/wwdc19/712

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advances in Networking, Part 2</td>
<td>Thursday, 5:00</td>
</tr>
<tr>
<td>Network Extensions for Modern macOS</td>
<td>Friday, 9:00</td>
</tr>
<tr>
<td>Networking Lab</td>
<td>Friday, 9:00</td>
</tr>
</tbody>
</table>