#WWDC19

Advances in Networking

Part 2

Eric Kinnear, Internet Technologies
Tommy Pauly, Internet Technologies
Stuart Cheshire, Internet Technologies

Part 2

Bonjour

Building Framing Protocols

Collecting Metrics

Best Practices and Status Updates

URLSession
Network.framework

Bonjour

Eric Kinnear, Internet Technologies

Bonjour Support

iOS

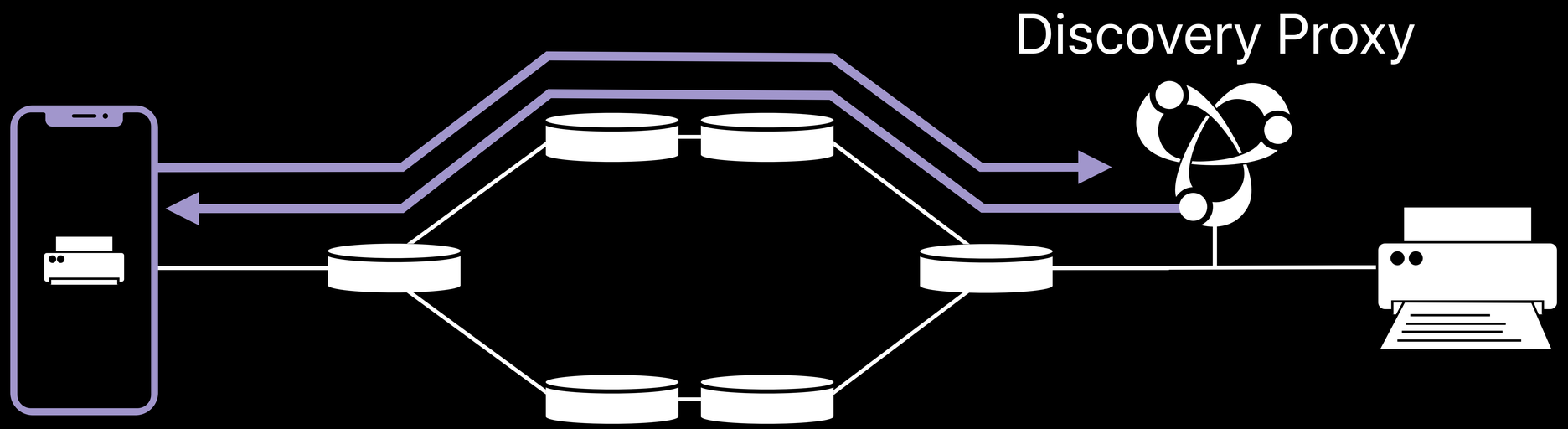
macOS

watchOS

tvOS

Wide-Area Service Discovery

NEW



https://github.com/IETF-Hackathon/mDNSResponder

Wide-Area Service Discovery

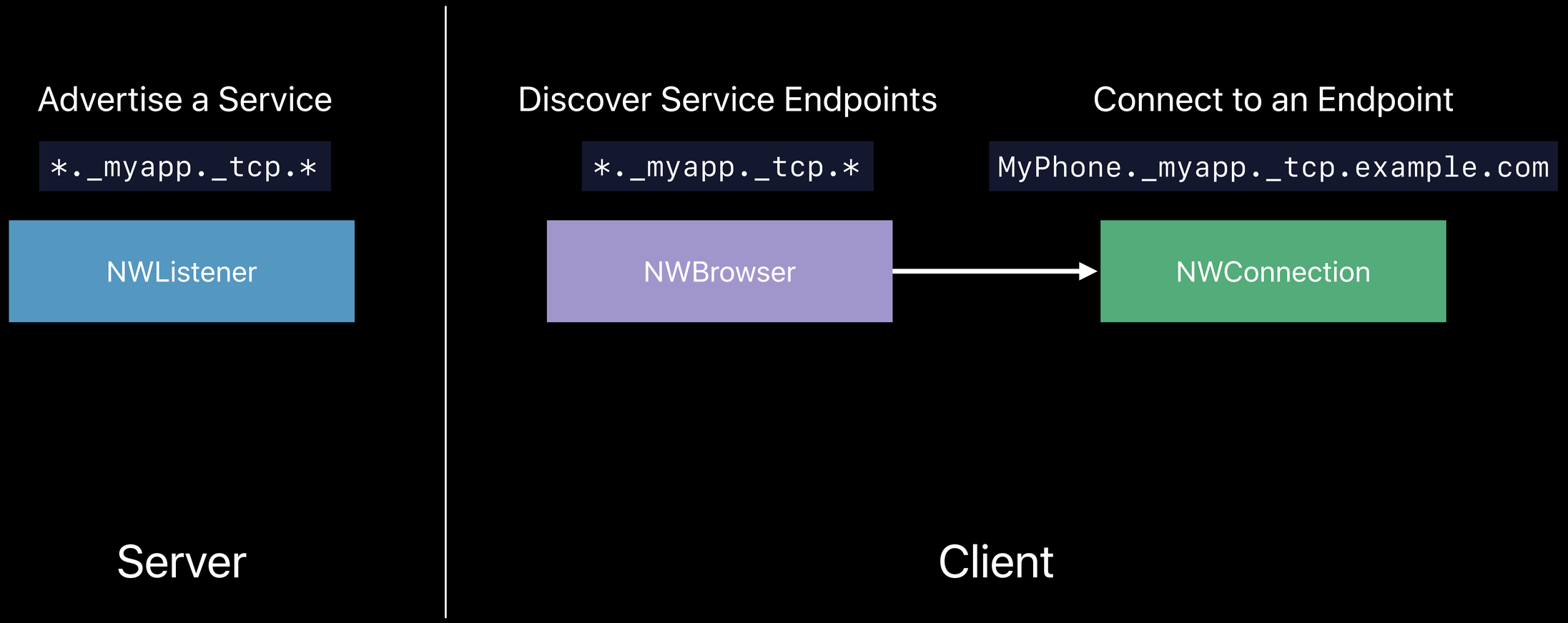
What this means for your app

When browsing, specify `nil` for domain

Specifying `"local"` will explicitly prevent any non-local discovery

Bonjour in Network.framework

NEW



Service Discovery in Network.framework

NEW

Browse for Bonjour service types

Optimized for Swift

Optionally access TXT record details

```
// Create a browser for "_myapp._tcp" on all domains
let browser = NWBrowser(for: .bonjour(type: "_myapp._tcp", domain: nil),
                        using: NWParameters())

browser.browseResultsChangedHandler = {...}

// Start browsing
browser.start(queue: myQueue)
```

```
browser.browseResultsChangedHandler = { _, changes in
    for change in changes {
        switch change {
        case .added(let browseResult):
            print("Added \(browseResult.endpoint)")
        case .removed(let browseResult):
            print("Removed \(browseResult.endpoint)")
        case .changed(_, let browseResult, let flags):
            if flags.contains(.interfaceAdded) {
                print("\(browseResult.endpoint) added interfaces")
            }
            if flags.contains(.interfaceRemoved) {
                print("\(browseResult.endpoint) removed interfaces")
            }
        default:
            print("No change")
        }
    }
}
```

```
browser.browseResultsChangedHandler = { browseResults, _ in
    // Up-to-date and complete list of browse results
    for browseResult in browseResults {
        print("Discovered \(browseResult.endpoint) over \(browseResult.interfaces)")
    }
}
```

Tic-Tac-Toe

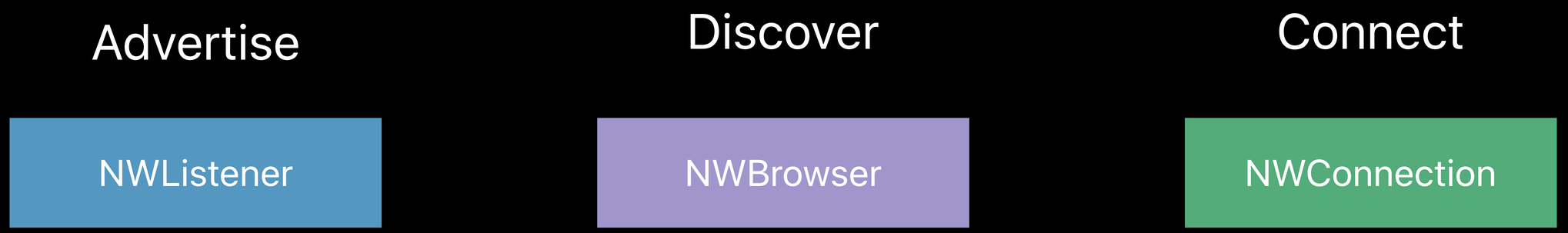
Advertise games to nearby players

Browse for available games

Use browse results to connect to game

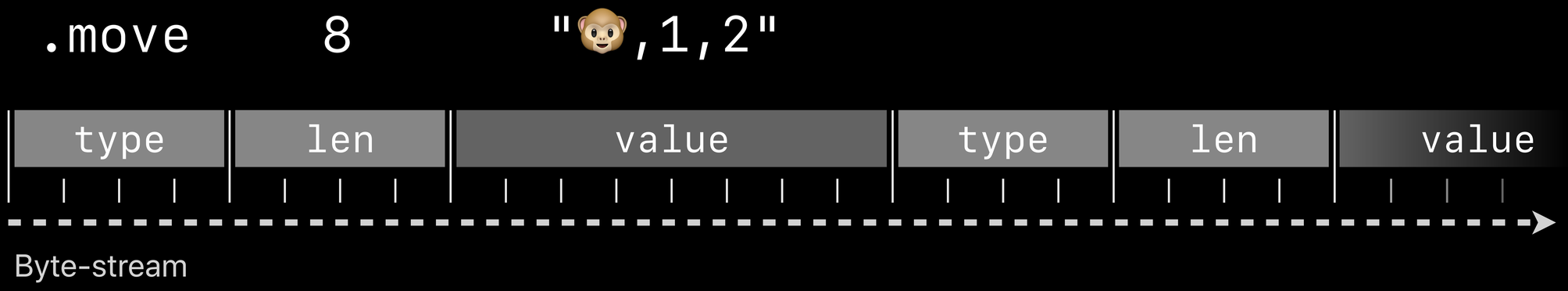
Demo

Tic-Tac-Toe

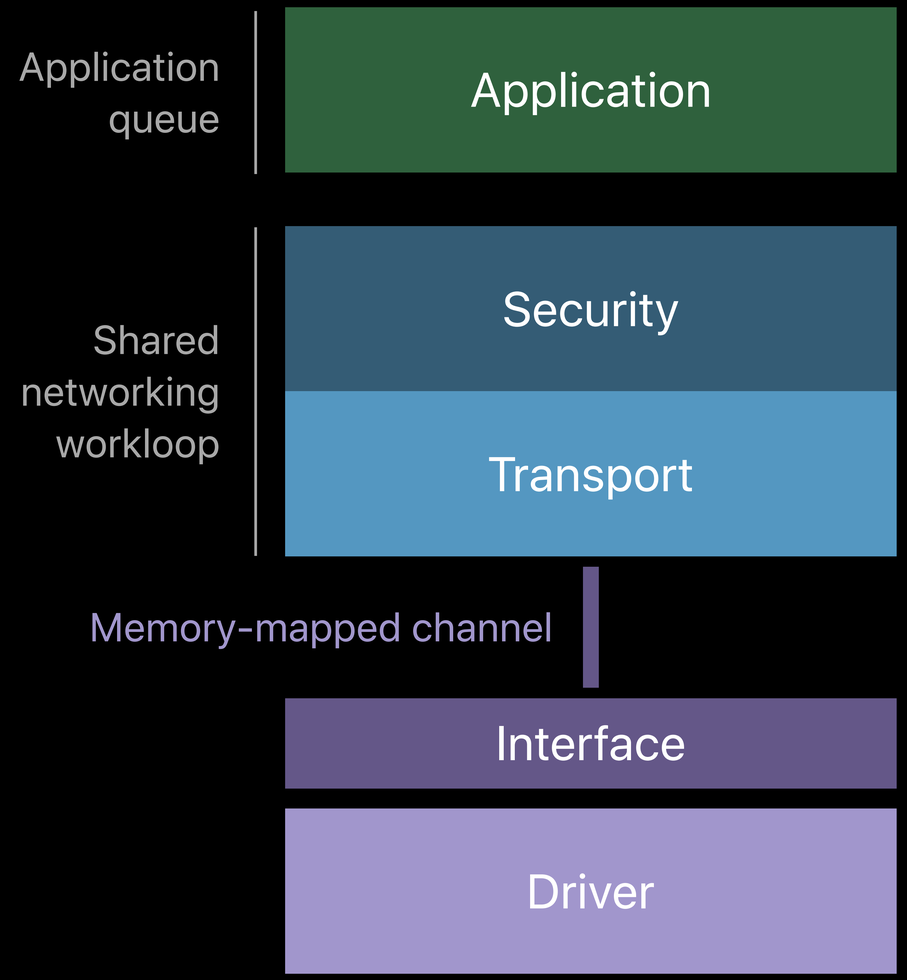

Advertise
Discover
Connect
NWListener
NWBrowser
NWConnection

Building Framing Protocols

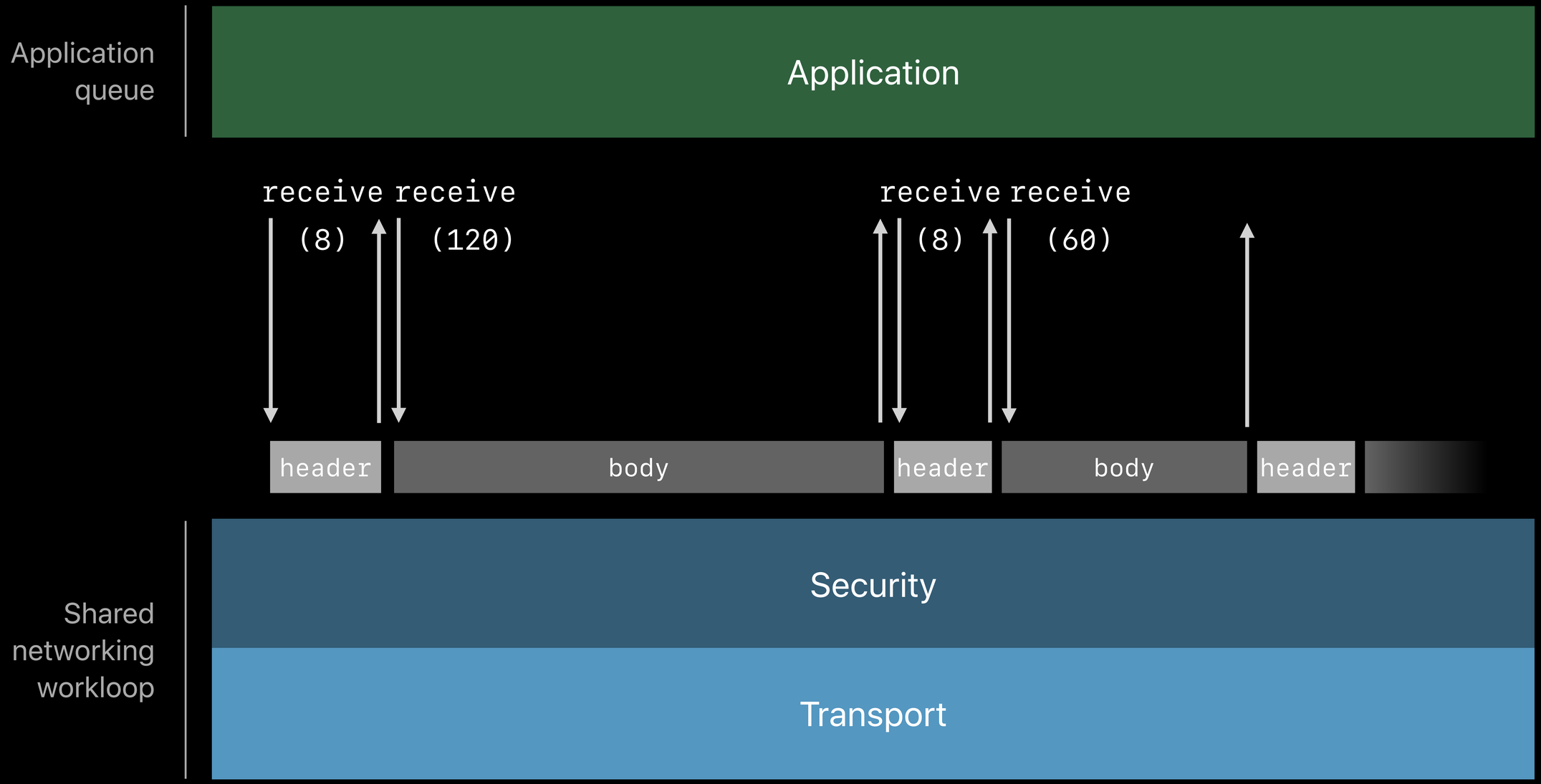
Tommy Pauly, Internet Technologies


.move 8 "🐵,1,2"
type
len
value
type
len
value
Byte-stream

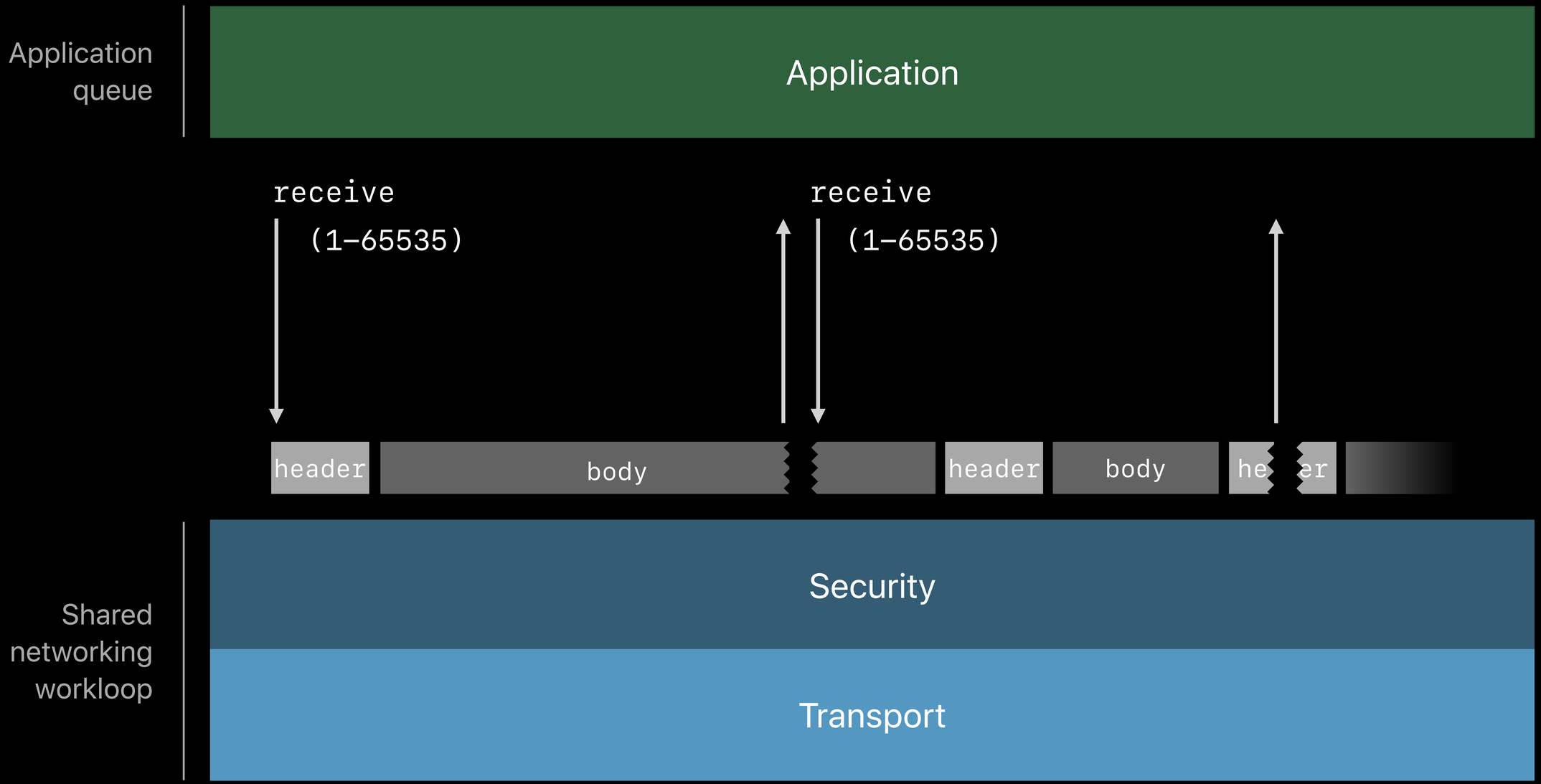
Messages, not byte-streams

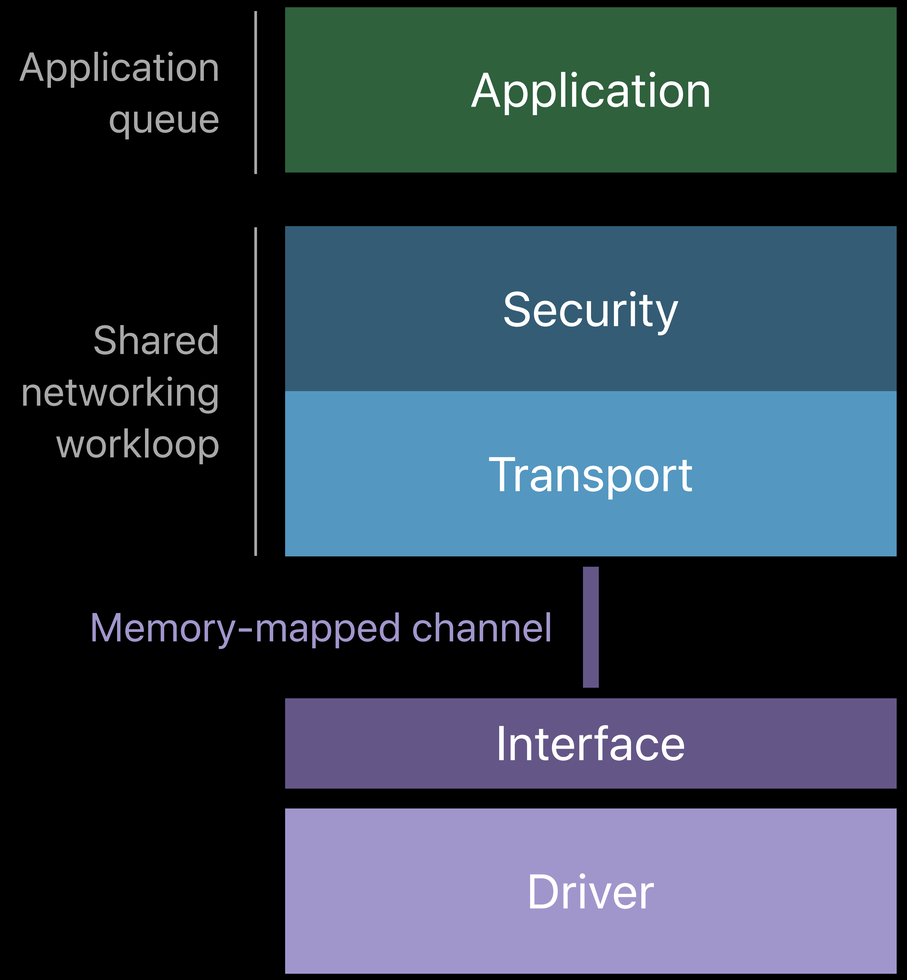

Application queue
Application
Shared networking workloop
Security
Transport
Memory-mapped channel
Interface
Driver

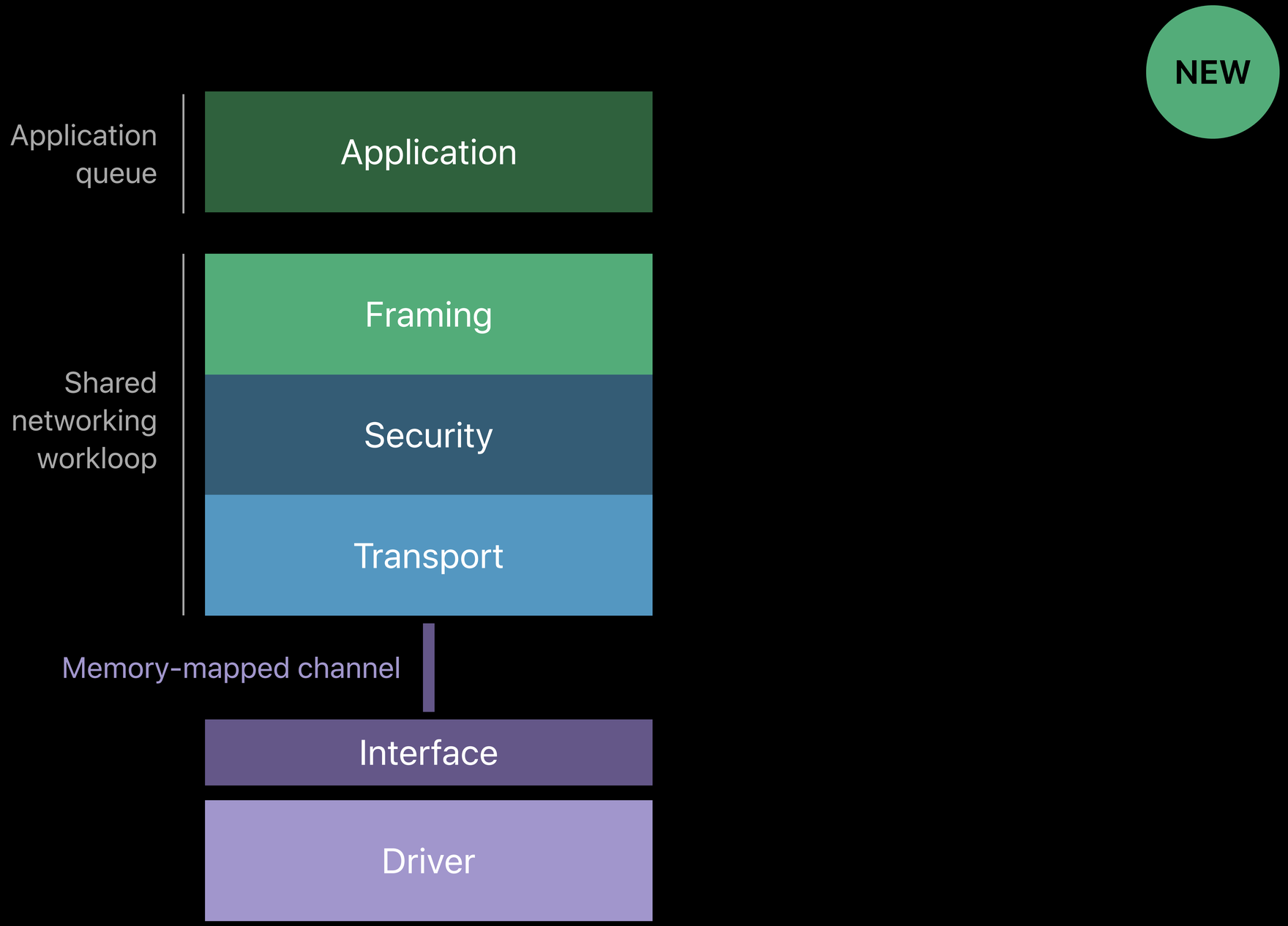
Parsing Messages



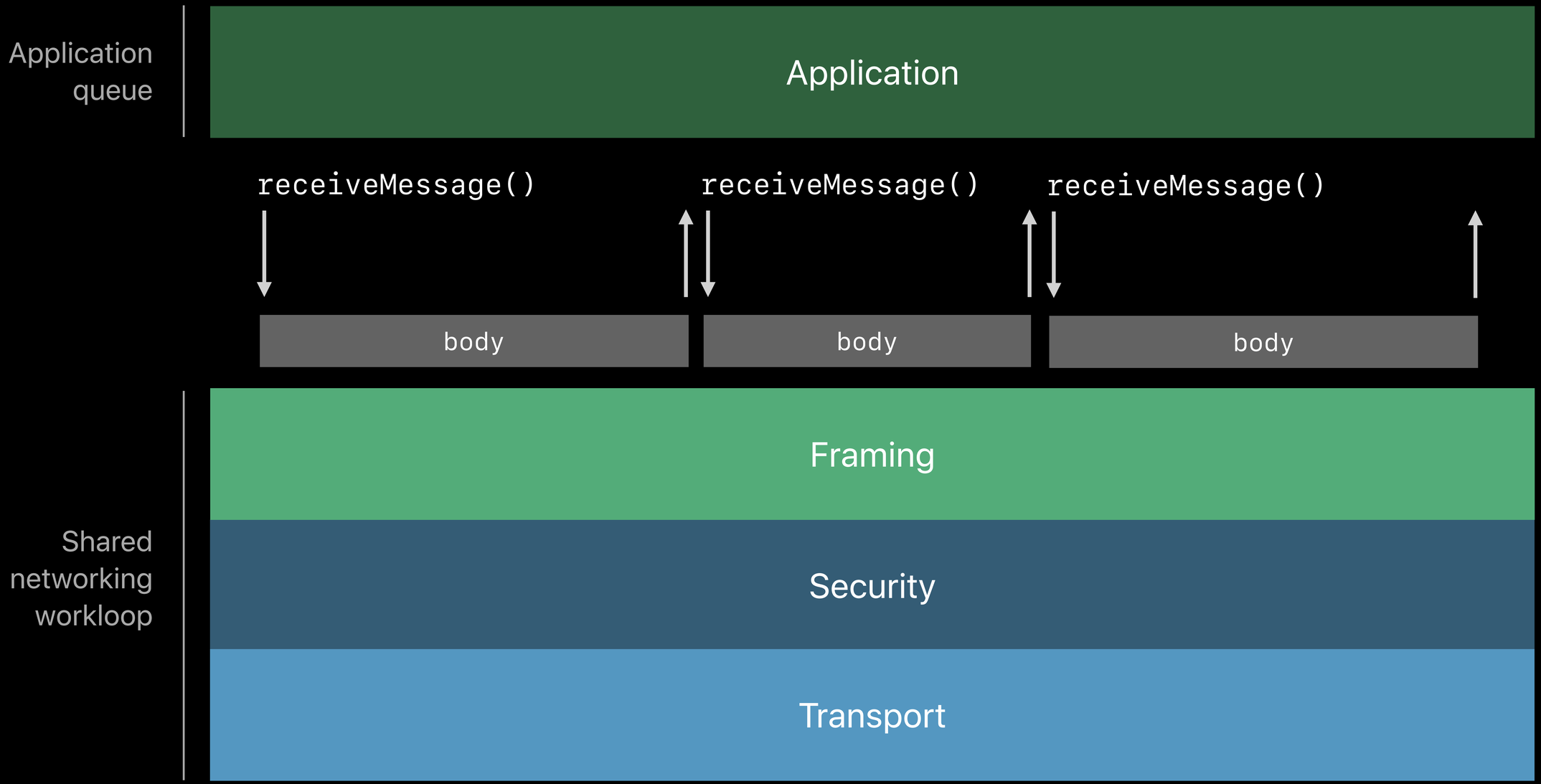
Parsing Messages




Application queue
Application
Shared networking workloop
Security
Transport
Memory-mapped channel
Interface
Driver


NEW
Application queue
Application
Shared networking workloop
Framing
Security
Transport
Memory-mapped channel
Interface
Driver

Message Framing



Framing protocols
encapsulate or **encode**
application messages

1. Implement a reusable framing protocol

2. Add the framing protocol to a connection

Implementing Framing Protocols

Define a class that implements `NWProtocolFramerImplementation`

- Encapsulate messages in `handleOutput()`
- Parse bytes into messages in `handleInput()`

Optionally store metadata in `NWProtocolFramer.Message`

```swift
// Define a framing protocol
class MyProtocol: NWProtocolFramerImplementation {

    static let definition = NWProtocolFramer.Definition(implementation: MyProtocol.self)

    required init(framer: NWProtocolFramer.Instance) { }
    func start(framer: NWProtocolFramer.Instance) -> NWProtocolFramer.StartResult {
        return .ready
    }
    func wakeup(framer: NWProtocolFramer.Instance) { }
    func stop(framer: NWProtocolFramer.Instance) -> Bool { return true }
    func cleanup(framer: NWProtocolFramer.Instance) { }

    func handleOutput(...)
    func handleInput(...)
    ...
}
```

```
// Header-prefixed output handler
func handleOutput(framer: NWProtocolFramer.Instance,
                  message: NWProtocolFramer.Message,
                  messageLength: Int,
                  isComplete: Bool) {
    // Initialize and write the header
    var header: MyHeader = MyHeader(...)
    framer.writeOutput(data: header.serializedData)

    // Write the body
    do {
        try framer.writeOutputNoCopy(length: messageLength)
    } catch let error {
        print("Error writing output: \(error)")
    }
}
```

```
func handleInput(framer: NWProtocolFramer.Instance) -> Int {
    while true {
        // Read out the header
        var header: MyHeader
        let headerSize = MyHeader.serializedLength
        let parsed = framer.parseInput(minimumIncompleteLength: headerSize,
                                       maximumLength: headerSize) { (buffer, isComplete) -> Int in
            // Try to parse header, and move the cursor forward by headerSize
            return headerSize
        }
        // If we couldn't read, return and wait for more bytes
        if !parsed { return headerSize }

        // Create a message object
        let message = NWProtocolFramer.Message(instance: framer)

        // Deliver the body and message to application
        if !framer.deliverInputNoCopy(length: length, message: message, isComplete: true) {
            return 0 // Waiting to deliver message, stop looping
        }
    }
}
```

Demo

Using Framing Protocols

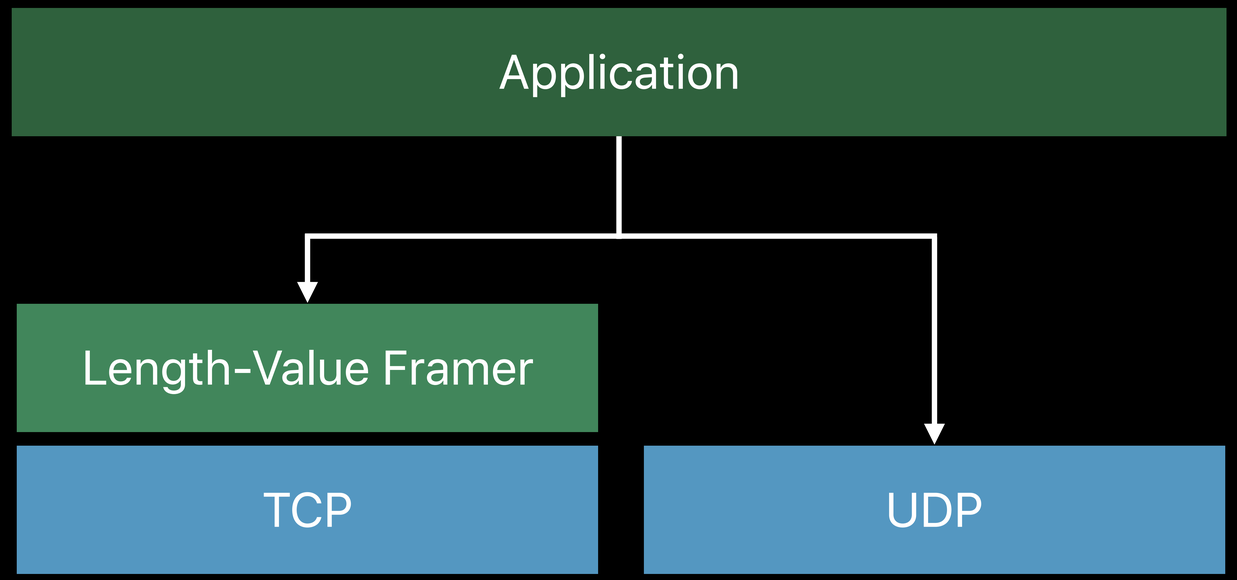
Add one or more framers to your `NWProtocolStack`

```
let framerOptions = NWProtocolFramer.Options(definition: MyProtocol.definition)

// Add the framer to NWParameters above TLS
let parameters = NWParameters.tls
parameters.defaultProtocolStack.applicationProtocols.insert(framerOptions, at: 0)
```

Add `NWProtocolWebSocket.Options()` in the same way

Using Framing Protocols

Use framing protocols to write common code across TCP and UDP transports



Using Framing Protocols

Send message values

```
let message = NWProtocolFramer.Message(definition: MyProtocol.definition)
message["key"] = “value"

let sendContext = NWConnection.ContentContext(identifier: "custom", metadata: [message])
connection.send(content: data, contentContext: sendContext,
                isComplete: true, completion: .contentProcessed { error in
    // Check for errors
})
```

Using Framing Protocols

Receive message values

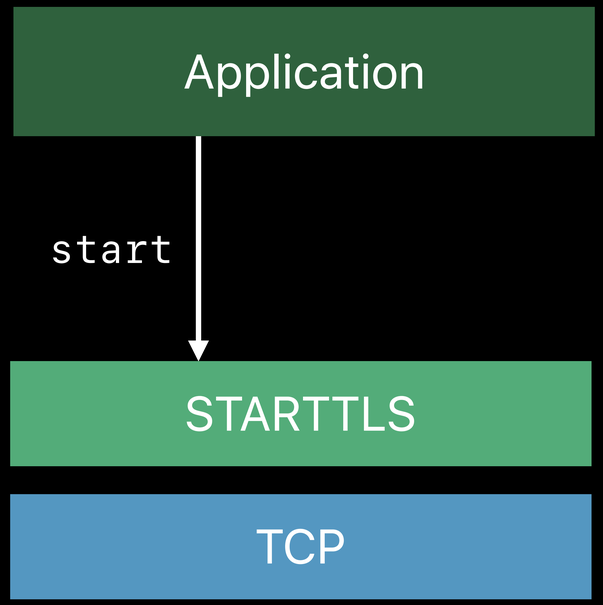
```
connection.receiveMessage { (content, context, isComplete, error) in
    if let context = context,
        let message = context.protocolMetadata(definition: MyProtocol.definition) as?
            NWProtocolFramer.Message,
        let value = message["key"] {
        // Use value
    }
}
```

Demo

Dynamic Protocol Stacks

Use `.willMarkReady` to implement a protocol handshake

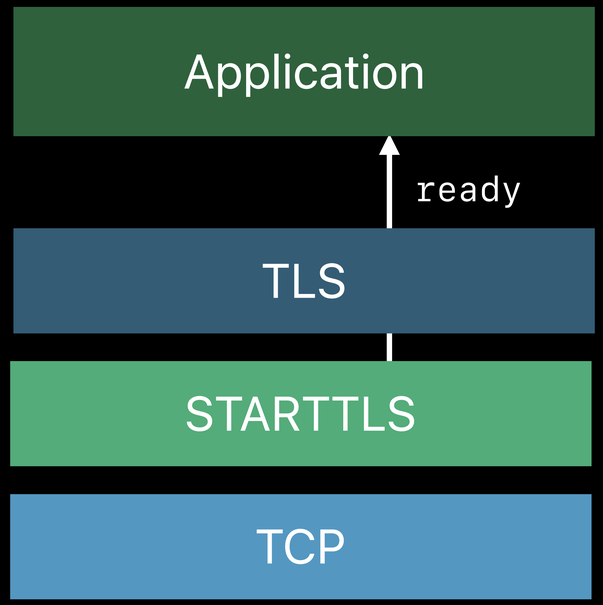
Call `prependApplicationProtocol()` to dynamically build a protocol stack



Dynamic Protocol Stacks

Use `.willMarkReady` to implement a protocol handshake

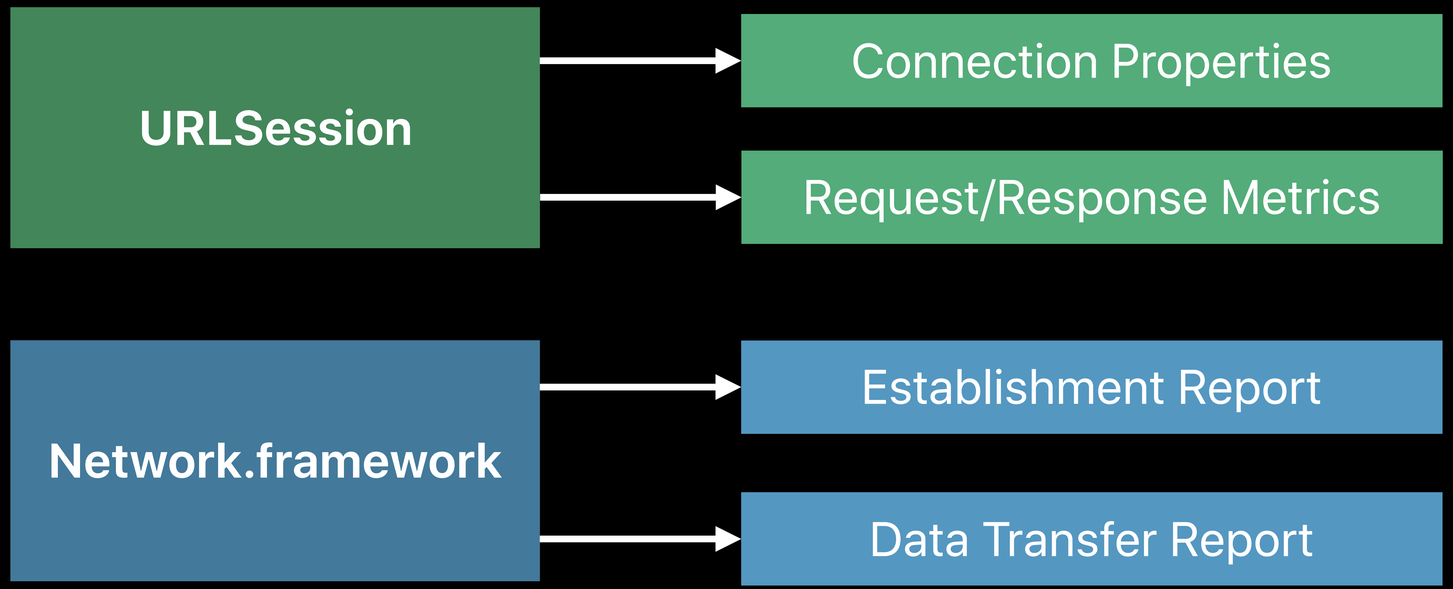
Call `prependApplicationProtocol()` to dynamically build a protocol stack



Collecting Metrics

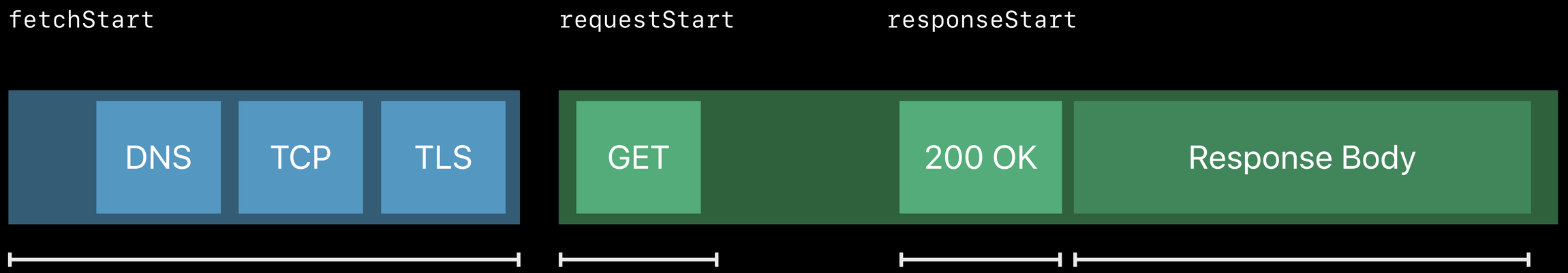
Collecting Metrics

NEW



URLSession Task Metrics

NEW

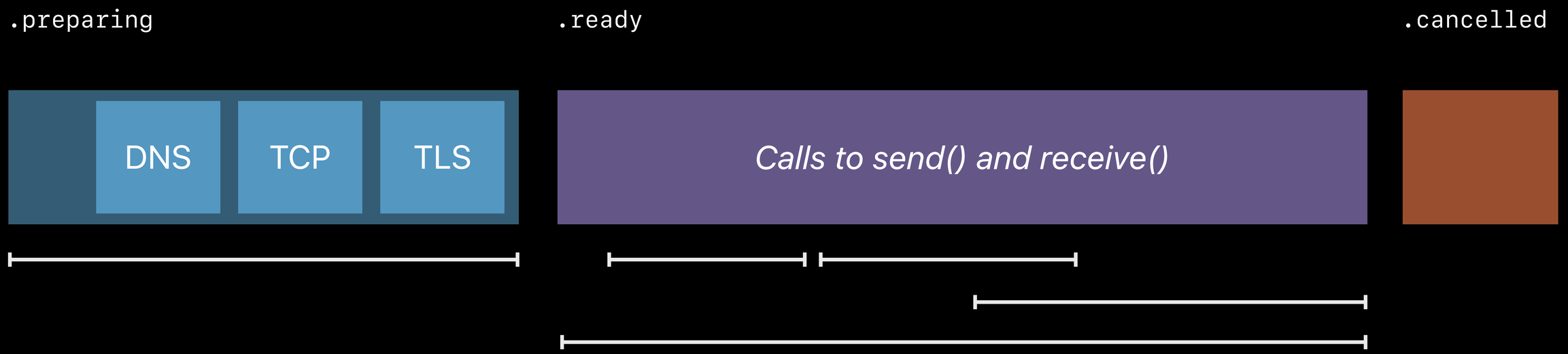


Connection Properties

Response and Request Metrics

Network.framework Metrics

NEW



Establishment Report

Data Transfer Reports

Connection Properties

URLSession

Available in the `didFinishCollectingMetrics` delegate method

```
if let transactionMetrics = metrics.transactionMetrics.last {
    // Access local and remote endpoints
    print("From \(transactionMetrics.localAddress) Port \(transactionMetrics.localPort)")
    print("To \(transactionMetrics.remoteAddress) Port \(transactionMetrics.remotePort)")

    // Check security properties
    if transactionMetrics.negotiatedTLSProtocolVersion == .TLSv13 {
        print("Used TLS 1.3")
    }

    // Inspect path properties
    print("Used constrained path: \(transactionMetrics.isConstrained)")
}
```

Establishment Report

Network.framework

Available after connection moves to the `.ready` state

DNS resolution source, latency, and address racing result

Protocol handshake duration and round-trip-time for TCP and TLS

Proxy information

```swift
// Request establishment report on a connection
connection.requestEstablishmentReport(queue: .main) { report in

    // Ensure that the report is ready
    if let report = report {
        print("Connection took \(report.duration) seconds to establish")

        for resolution in report.resolutions {
            print("Resolved \(resolution.endpointCount) endpoints from
                  \(resolution.source) in \(resolution.duration) seconds")
        }

        for handshake in report.handshakes {
            print("Handshake for \(handshake.definition) took
                  \(handshake.handshakeDuration) seconds")
        }
    }
}
```

Optimistic DNS

Optimistic DNS is now enabled by default

Improves performance for answers with short times-to-live

```
// NWConnection.EstablishmentReport.Resolution
public enum Source {
    case query
    case cache
    case expiredCache
}
```

Demo

Request and Reponse Metrics

URLSession

Access header and body bytes counts

```
if let transactionMetrics = metrics.transactionMetrics.last {
    print("Sent \(transactionMetrics.countOfRequestHeaderBytesSent) header bytes")
    print("Sent \(transactionMetrics.countOfRequestBodyBytesSent) encoded body bytes”)
    print("Sent \(transactionMetrics.countOfRequestBodyBytesBeforeEncoding) body bytes")

    print("Read \(transactionMetrics.countOfResponseHeaderBytesReceived) header bytes")
    print("Read \(transactionMetrics.countOfResponseBodyBytesReceived) encoded body bytes”)
    print(“Read \(transactionMetrics.countOfResponseBodyBytesAfterDecoding) body bytes")
}
```

Data Transfer Report

Network.framework

Retrieve byte counts, packet counts, and round-trip-times

Start and end reports to correspond to application activity

Multiple reports supported simultaneously

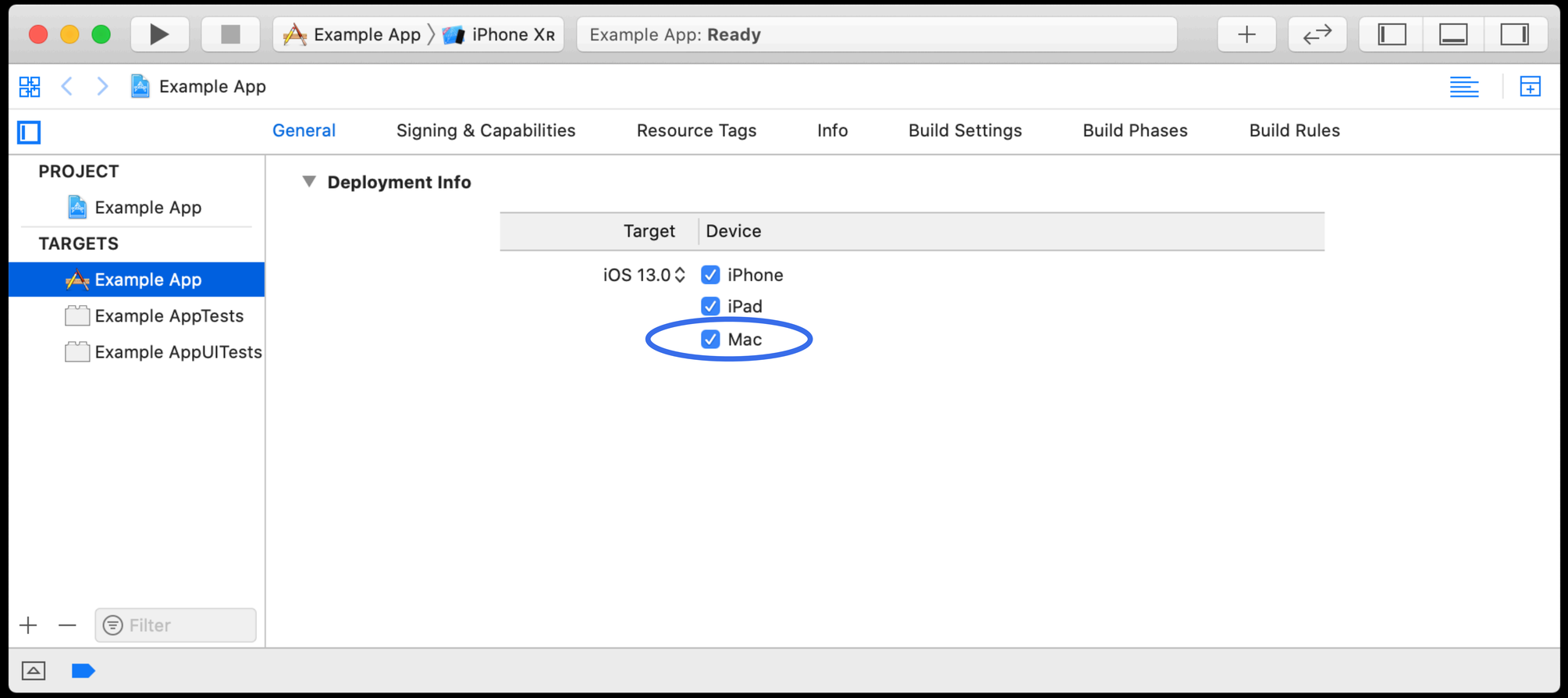
Per-path breakdown for multi-path protocols

```swift
// Start a data transfer report
let pendingReport = connection.startDataTransferReport()

// Later, collect the report
pendingReport.collect(queue: .main) { report in
    let aggregateReport = report.aggregatePathReport

    print("Sent \(aggregateReport.sentIPPacketCount) packets")
    print("Received \(aggregateReport.receivedIPPacketCount) packets")

    print("Sent \(aggregateReport.sentTransportByteCount) bytes")
    print("Received \(aggregateReport.receivedTransportByteCount) bytes")

    print("RTT is \(aggregateReport.transportSmoothedRTT) seconds")
}
```

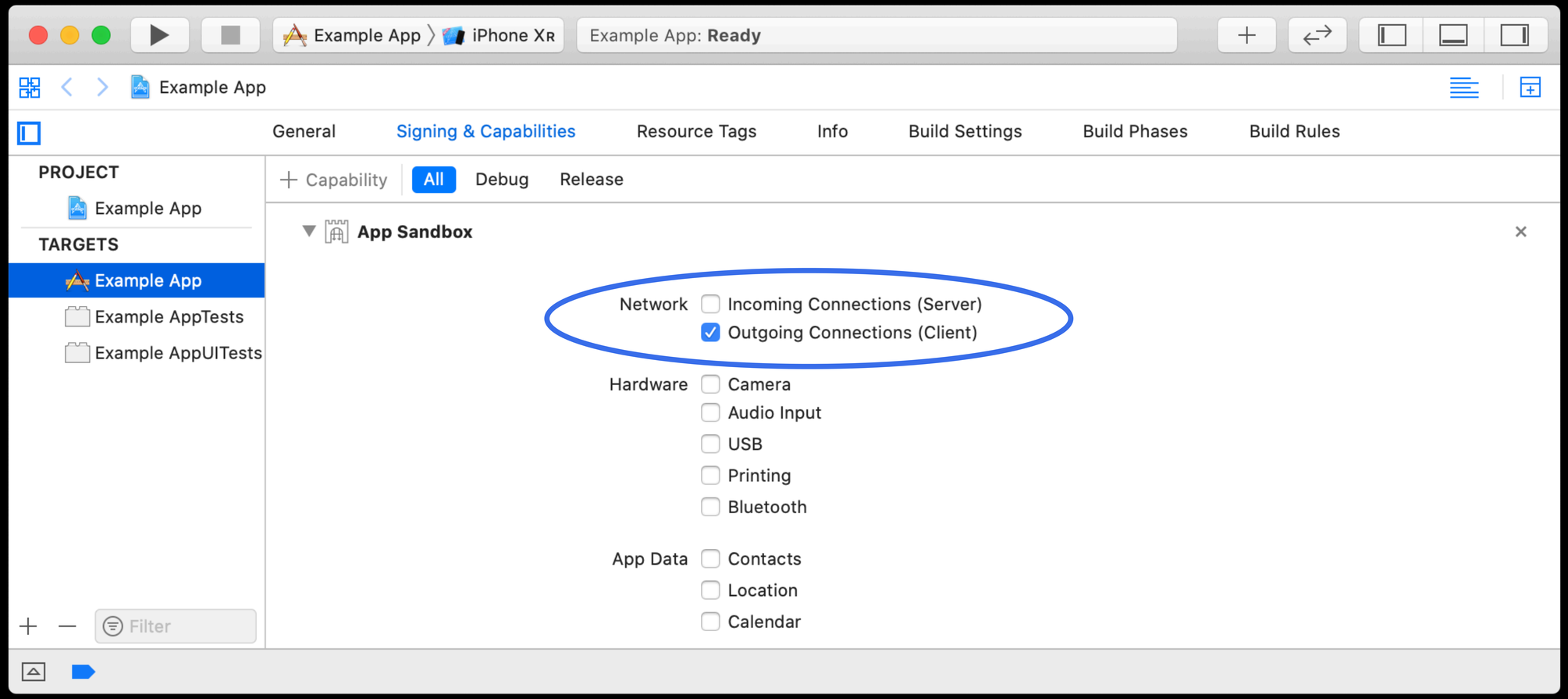
Best Practices and Status Updates

Stuart Cheshire, Internet Technologies

iPad Apps for Mac



iPad Apps for Mac



Networking on watchOS

NEW

Audio streaming apps that use `AVFoundation` can use direct networking

Use `URLSessionStreamTask` or `NWConnection` for TLS, TCP, and UDP

Traffic will either proxy through the phone, or use Wi-Fi or Cellular directly

Performance and Privacy Improvements with TLS 1.3

IETF RFC 8446, August 2018

	TLS 1.2	TLS 1.3
Connection setup performance	Two round trips	One round trip (almost always)
Security	Weak cryptographic algorithms	AEAD with Forward Secrecy
Privacy	Certificates, most handshake fields, and Server Name Indication are cleartext	Certificates and most handshake fields are encrypted Work on Encrypted SNI underway

CNCopyCurrentNetworkInfo

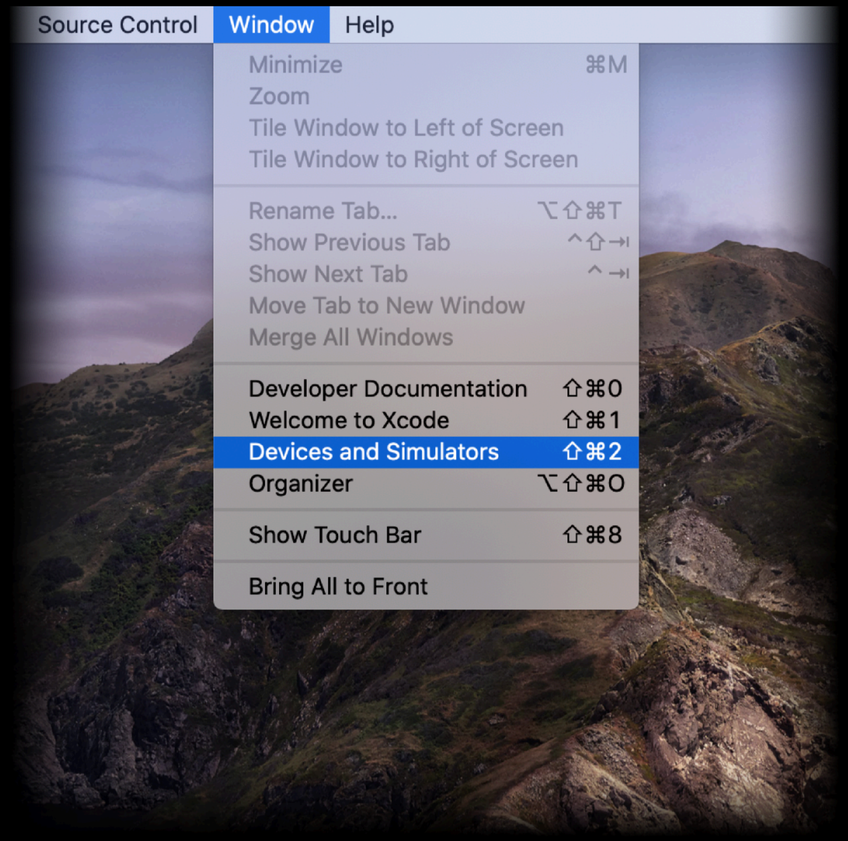
Requires Capability: `Access Wi-Fi Information`

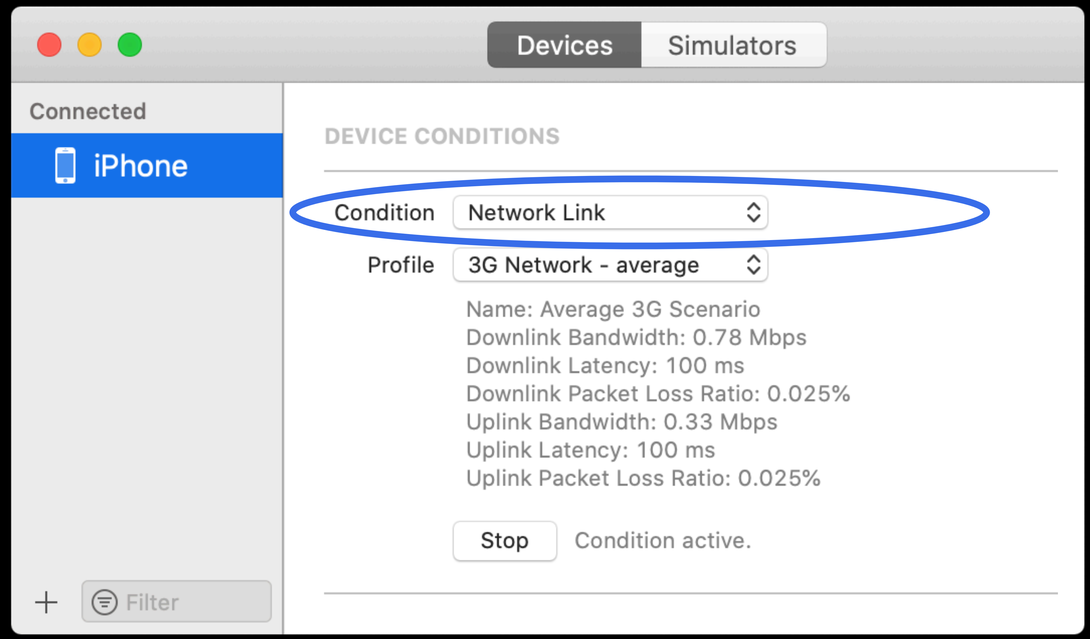
Must also meet at least one of criteria below

- Apps with permission to access location
- Currently enabled VPN app
- NEHotspotConfiguration (only Wi-Fi networks that the app configured)

Otherwise, returns `nil`

Network Link Conditioner





Alternatives to Networking Pre-Flight Checks

Pre-flight checks have inherent race conditions

We still see many apps that do this

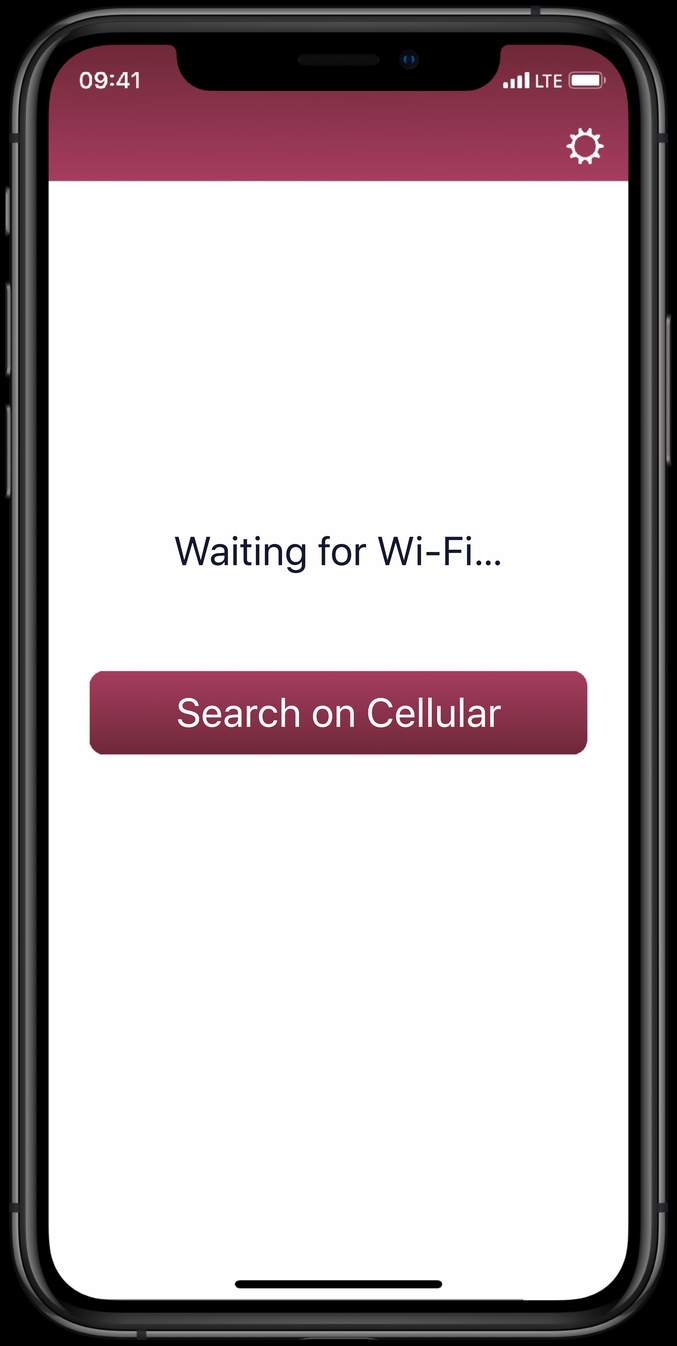
Instead of pre-flight checks, attach constraints to your operations


09:41
LTE
Get on Wi-Fi
and then
click Connect

```
allowsExpensiveNetworkAccess = false
```

```
waitsForConnectivity = true
```

```
taskIsWaitingForConnectivity
```



Deprecations

PAC files for schemes `file://` and `ftp://`

SPDY

- Replaced by HTTP/2

Secure Transport

- Does not support TLS 1.3
- Use URLSession or Network.framework instead

Summary — Part 1

Low Data Mode

Combine in URLSession

WebSocket

Mobility Improvements

Summary — Part 2

Bonjour

Building Framing Protocols

Collecting Metrics

Best Practices and Status Updates

More Information

developer.apple.com/wwdc19/713

Advances in Networking, Part 1	Thursday, 11:00
Network Extensions for Modern macOS	Friday, 9:00
Networking Lab	Friday, 9:00


WWDC19