



Network Apps for iPhone OS

Part 2

Quinn "The Eskimo!"
Developer Technical Support

“The difference between theory and practice is a lot greater in practice than it is in theory.”

Brian Bechtel

Practical Matters

- Asynchrony
- Debugging
- Common ~~anti-patterns~~ mistakes

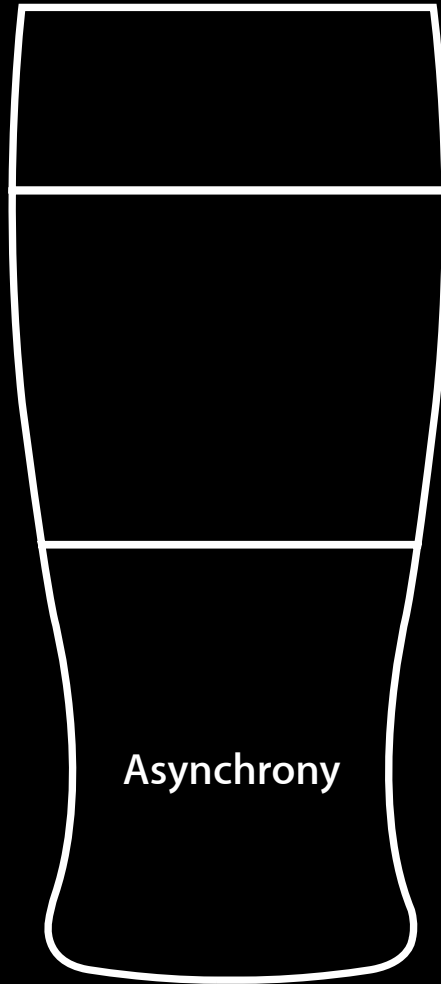


A beer glass is depicted against a black background. The glass is divided into three horizontal sections. The top section is a thick, white, rounded cap of foam. The middle section is filled with a golden-yellow liquid, representing beer. The bottom section is a narrower, tapered base, also filled with the same golden-yellow liquid. Each section contains a text label.

Common
Mistakes

Debugging

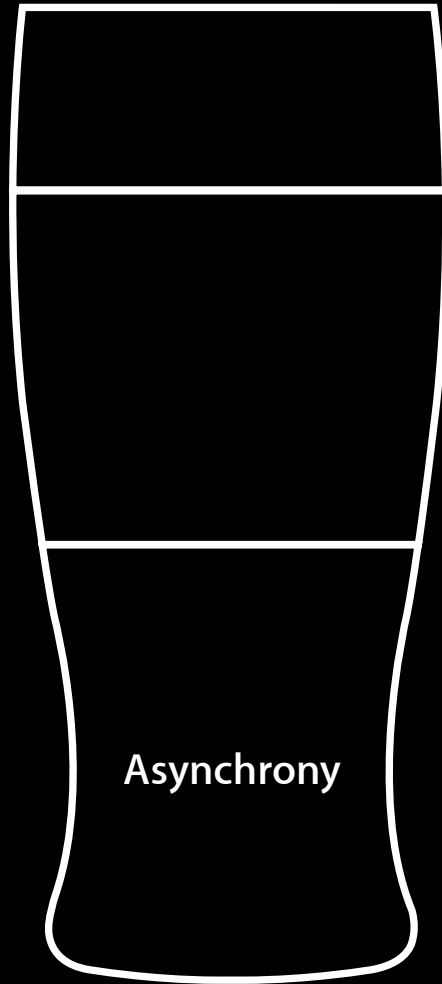
Asynchrony



Asynchrony

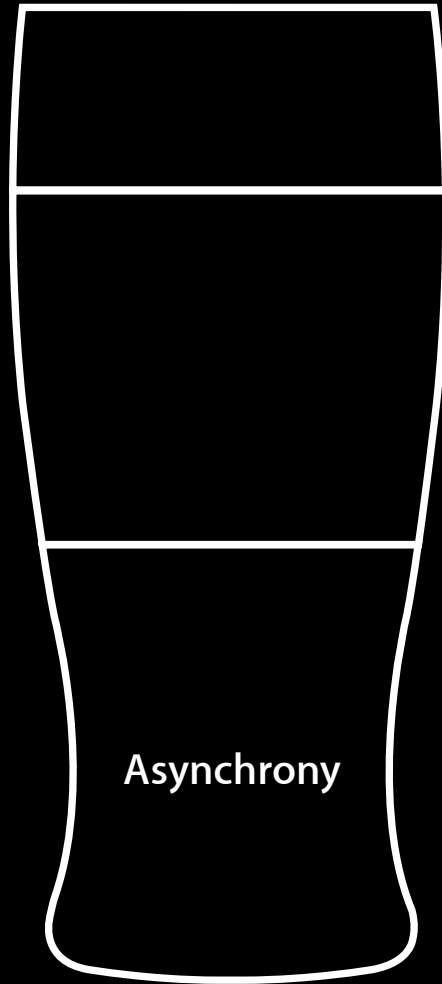
Asynchrony

- The basics
- Run loops
- State management



Asynchrony

- The basics
- Run loops
- State management



Synchronous vs Asynchronous

```
- (void)start {  
    [self runRequest1];  
    [self processResults1];  
    [self runRequest2];  
    [self processResults2];  
    [self runRequest3];  
    [self processResults3];  
}
```

```
- (void)start {  
    [self startRequest1];  
}  
- (void)request1Done {  
    [self processResults1];  
    [self startRequest2];  
}  
- (void)request2Done {  
    [self processResults2];  
    [self startRequest3];  
}  
- (void)request3Done {  
    [self processResults3];  
}
```

Synchronous vs Asynchronous

```
- (void)start {  
    [self runRequest1];  
    [self processResults1];  
    [self runRequest2];  
    [self processResults2];  
    [self runRequest3];  
    [self processResults3];  
}
```

```
- (void)start {  
    [self startRequest1];  
}  
- (void)request1Done {  
    [self processResults1];  
    [self startRequest2];  
}  
- (void)request2Done {  
    [self processResults2];  
    [self startRequest3];  
}  
- (void)request3Done {  
    [self processResults3];  
}
```

Why Asynchronous?

Synchronous + Main Thread = Death



Recognize a Watchdog Crash

Look for “ate bad food”

Exception Type: 00000020
Exception Codes: 0x8badf00d
Highlighted Thread: 0

Documentation:

TN2151 Understanding and Analyzing iPhone OS Application Crash Reports
<http://developer.apple.com/iphone/library/technotes/tn2008/tn2151.html>

See Session:

Understanding Crash Reports on iPhone OS

Watchdog and Synchronous Networking

Operation	Approximate Timeout*
Watchdog	20 seconds
DNS	30 seconds
TCP Connection	75 seconds
NSURLConnection	60 seconds

*Default timeouts; subject to change

Synchronous + Main Thread = Death

Hidden Synchronous Networking

- Utility methods

 - initWithContentsOfURL:

 - +stringWithContentsOfURL:

- DNS

 - gethostbyname

 - gethostbyaddr

 - NSHost (Mac OS X)

 - +sendSynchronousRequest:returningResponse:error:

- Synthetic synchronous

Synthetic Synchronous

- Call an asynchronous API
- Wait for it to complete
 - Typically running the run loop

```
[self startRequest1];  
while ( ! [self isFinished] ) {  
    [self wait];  
}  
[self processResults1];
```

- Helpful in some circumstances
- Not a miracle cure

Attaining Asynchrony

Threads

Threads Are Evil™

GCD

The Future™

Run loop

Recommended by nine out of ten Quinns™

iPhone OS Networking

Applications

Foundation

CFNetwork

Darwin Foundation



Attaining Asynchrony

Threads

Threads Are Evil™

GCD

The Future™

Run loop

Recommended by nine out of ten Quinns™

Why Are Threads Evil?

- Locking
- Cancellation
- Timeouts
- Bidirectional
- Resource use

But...

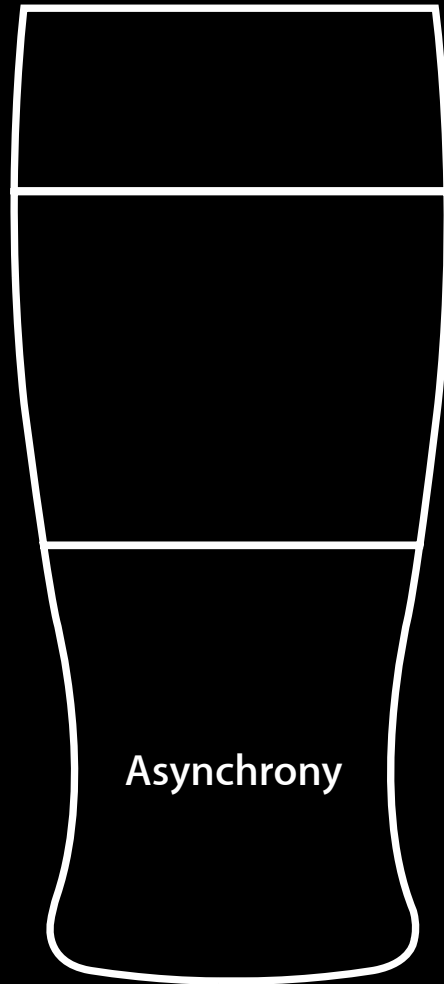
NSOperation

- Encapsulates asynchrony
- Mix and match CPU and I/O operations
 - Standard for CPU tasks
 - Concurrent for network tasks
- Concurrent operations tricky
- LinkedImageFletcher

Hidden Threads

- NSOperation
 - Even concurrent operations
- GCD
 - performSelectorInBackground:withObject:

Asynchrony



- The basics
- **Run loops**
- State management

Run Loop Factoids

- One run loop per thread
- Event dispatch mechanism
- Event sources
 - Each with associated callback
- Run loop must be run explicitly
 - Monitors event sources
 - Calls callbacks
- UIKit runs main thread's run loop

Run Loops

Main Thread

Run Loop

Secondary Thread

Run Loop

Secondary Thread

Run Loop

...and so on

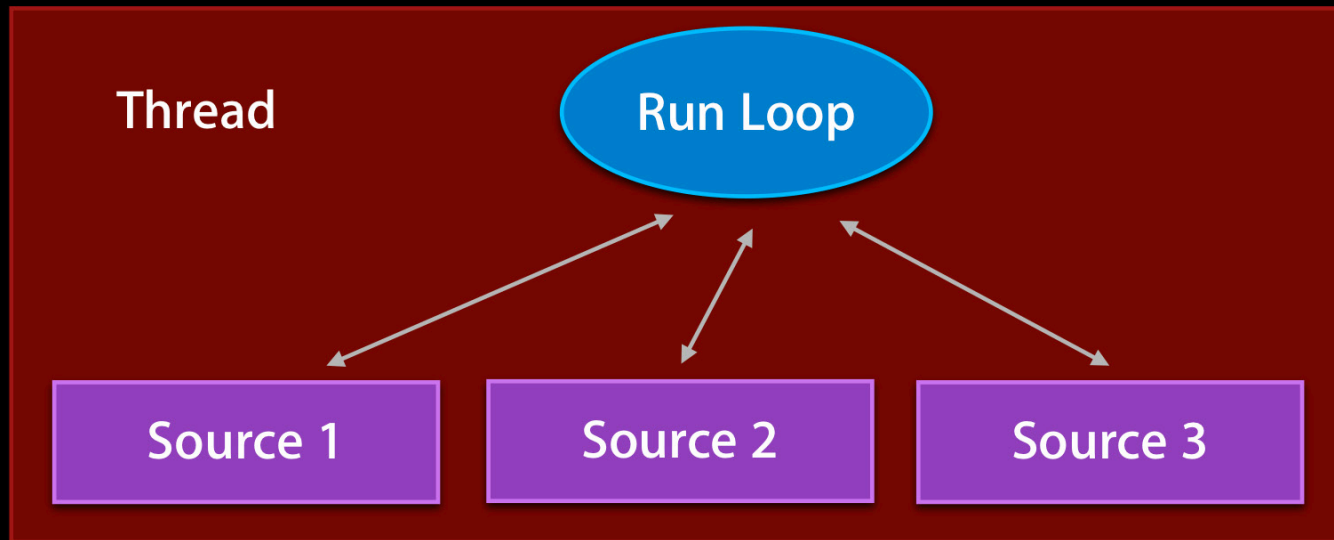
Run Loops

Main Thread

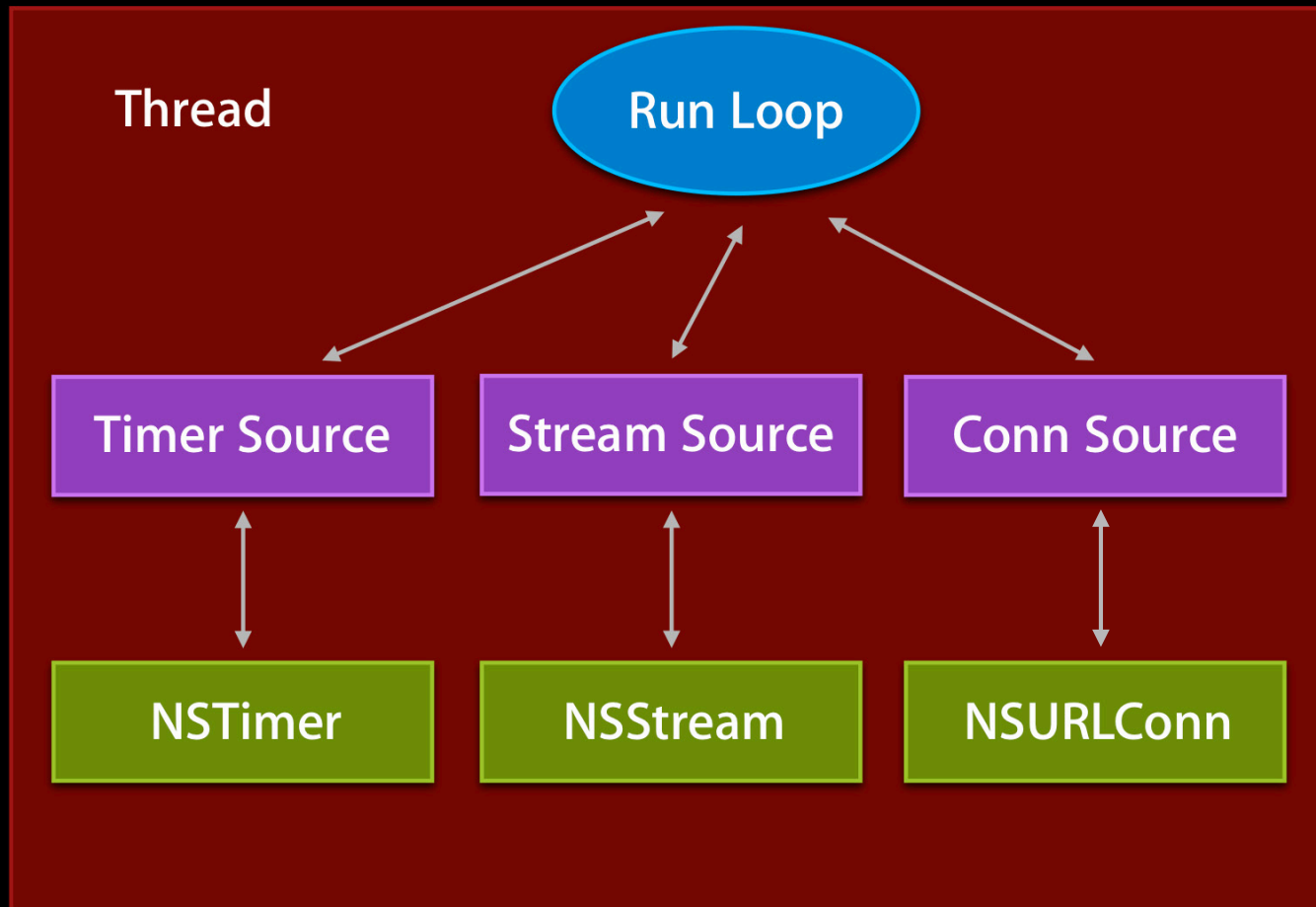
Run Loop

A diagram illustrating the components of a run loop. It features a dark red rectangular background. On the left side of this background, the text "Main Thread" is written in white. On the right side, there is a blue oval with a white border, containing the text "Run Loop" in white.

Run Loops



Run Loops



Explicit Scheduling

```
NSInputStream * stream;  
  
[netService getInputStream:&stream outputStream:NULL];  
[stream scheduleInRunLoop:[NSRunLoop currentRunLoop]  
    forMode:NSDefaultRunLoopMode];  
[stream setDelegate:self];  
[stream open];
```

```
-(void)stream:(NSStream *)stream handleEvent:(NSStreamEvent)e  
{  
}
```


Implicit Scheduling

```
NSURLConnection * conn;  
  
conn = [NSURLConnection connectionWithRequest:req  
      delegate:self];
```

```
- (void)connection:(NSURLConnection *)conn  
  didReceiveResponse:(NSURLResponse *)resp  
{  
}
```

Making It Explicit

```
NSURLConnection * conn;

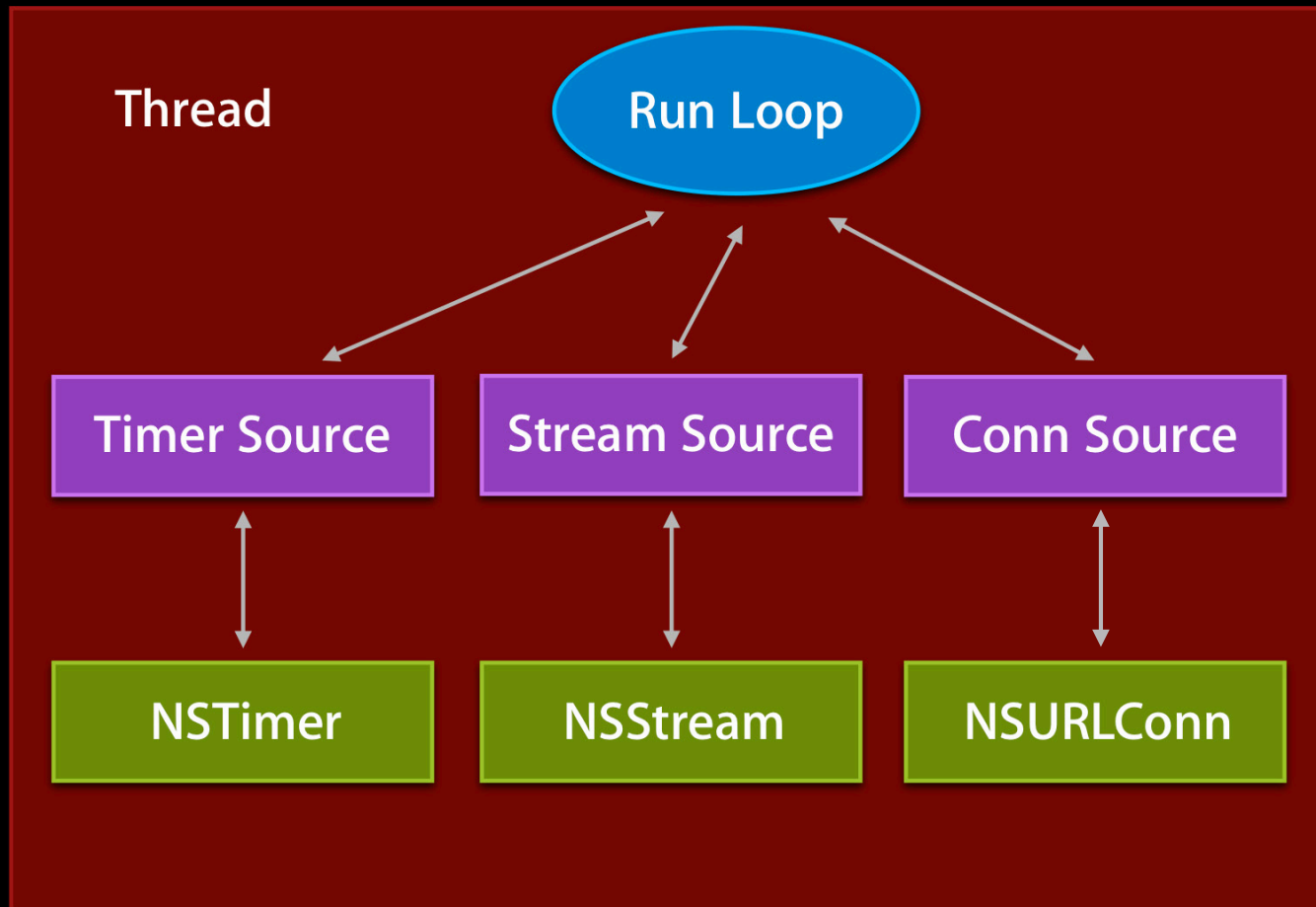
conn = [[NSURLConnection alloc] initWithRequest:req
        delegate:self
        startImmediately:NO];
[conn scheduleInRunLoop:[NSRunLoop currentRunLoop]
        forMode:NSDefaultRunLoopMode];
[conn start];
```

```
- (void)connection:(NSURLConnection *)conn
    didReceiveResponse:(NSURLResponse *)resp
{
}
```

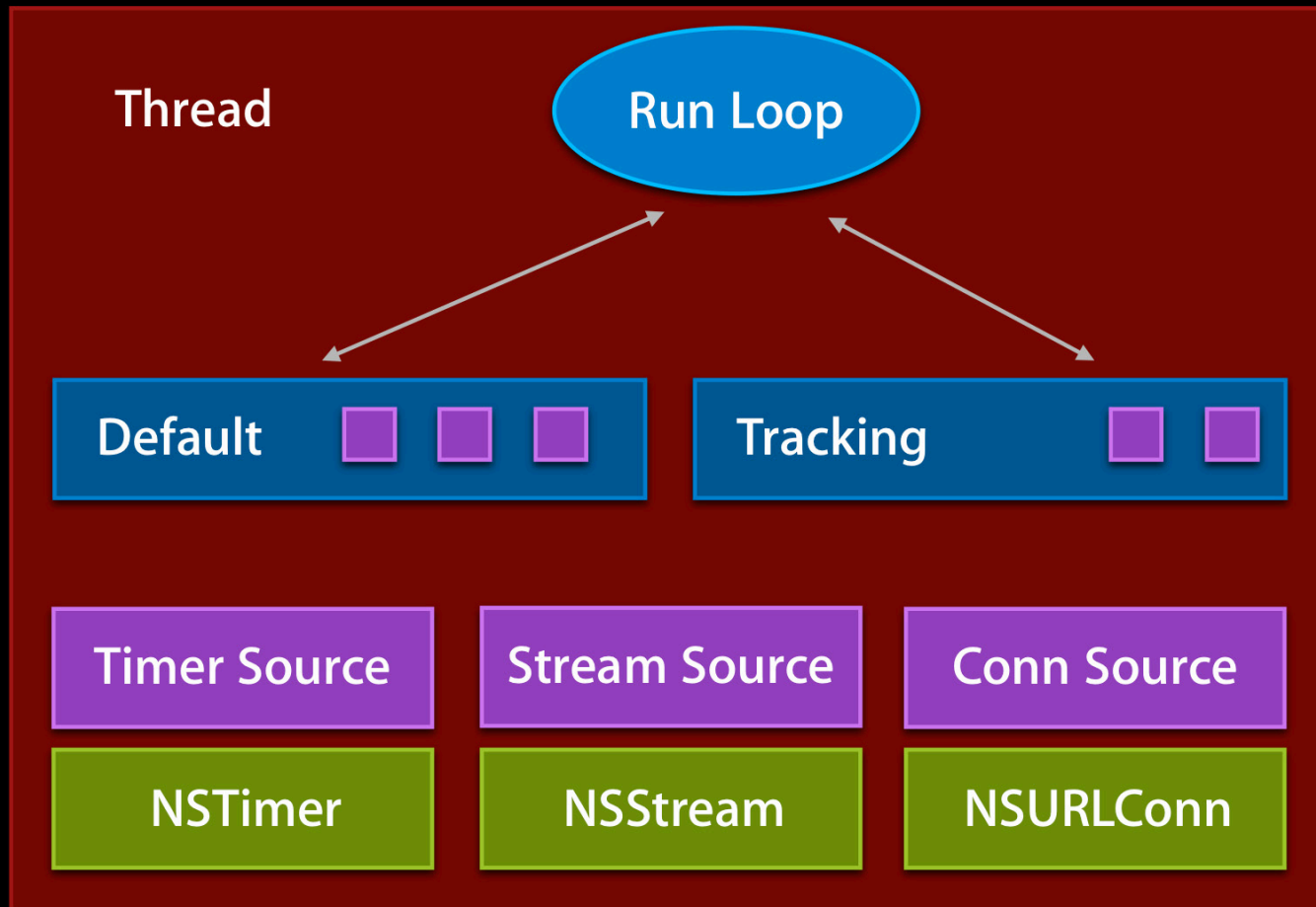
Run Loop Modes

- Event sources added in a mode
- Run loop runs in a mode
 - Monitors event sources in that mode
 - Other event sources ignored

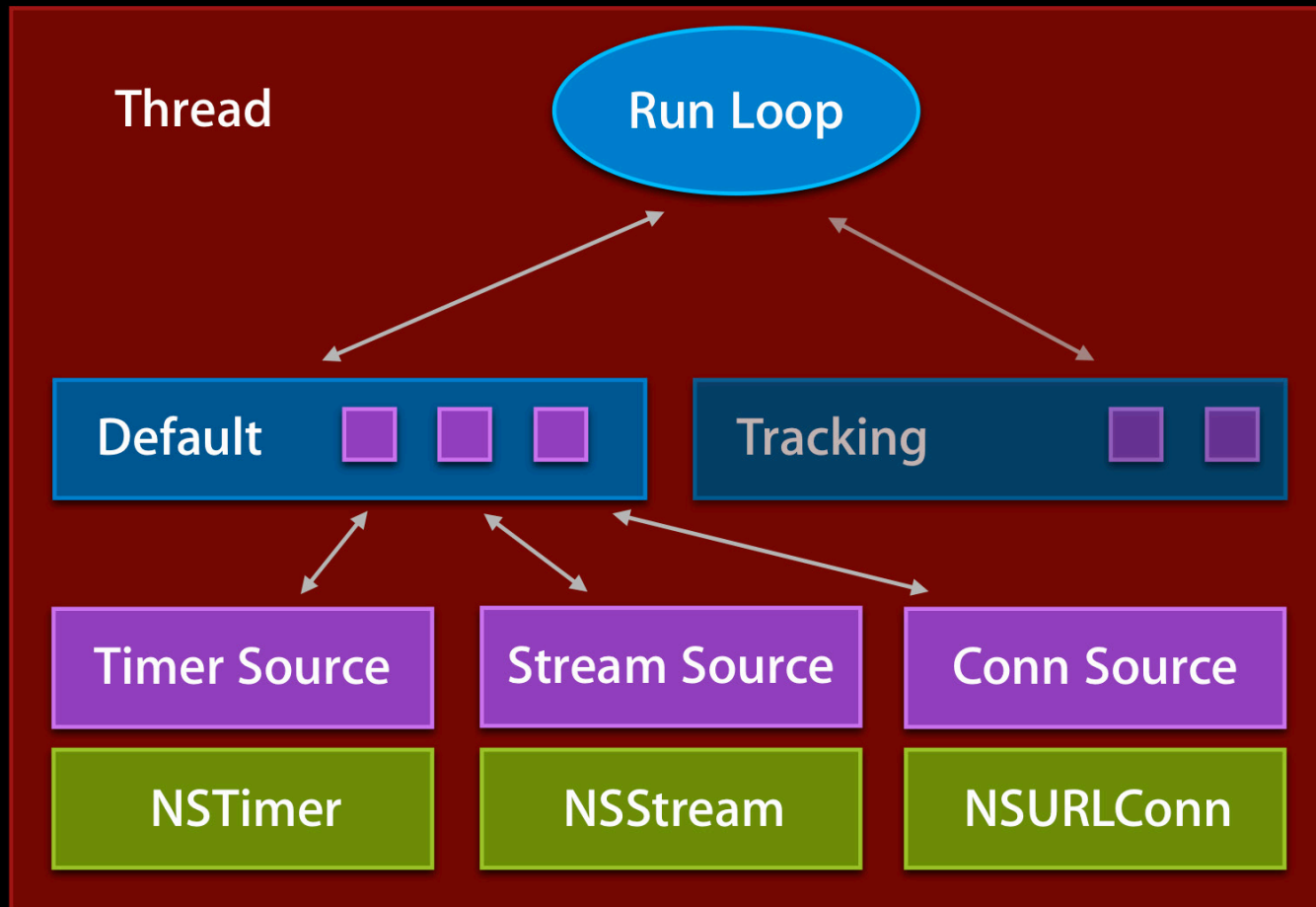
Run Loop Modes



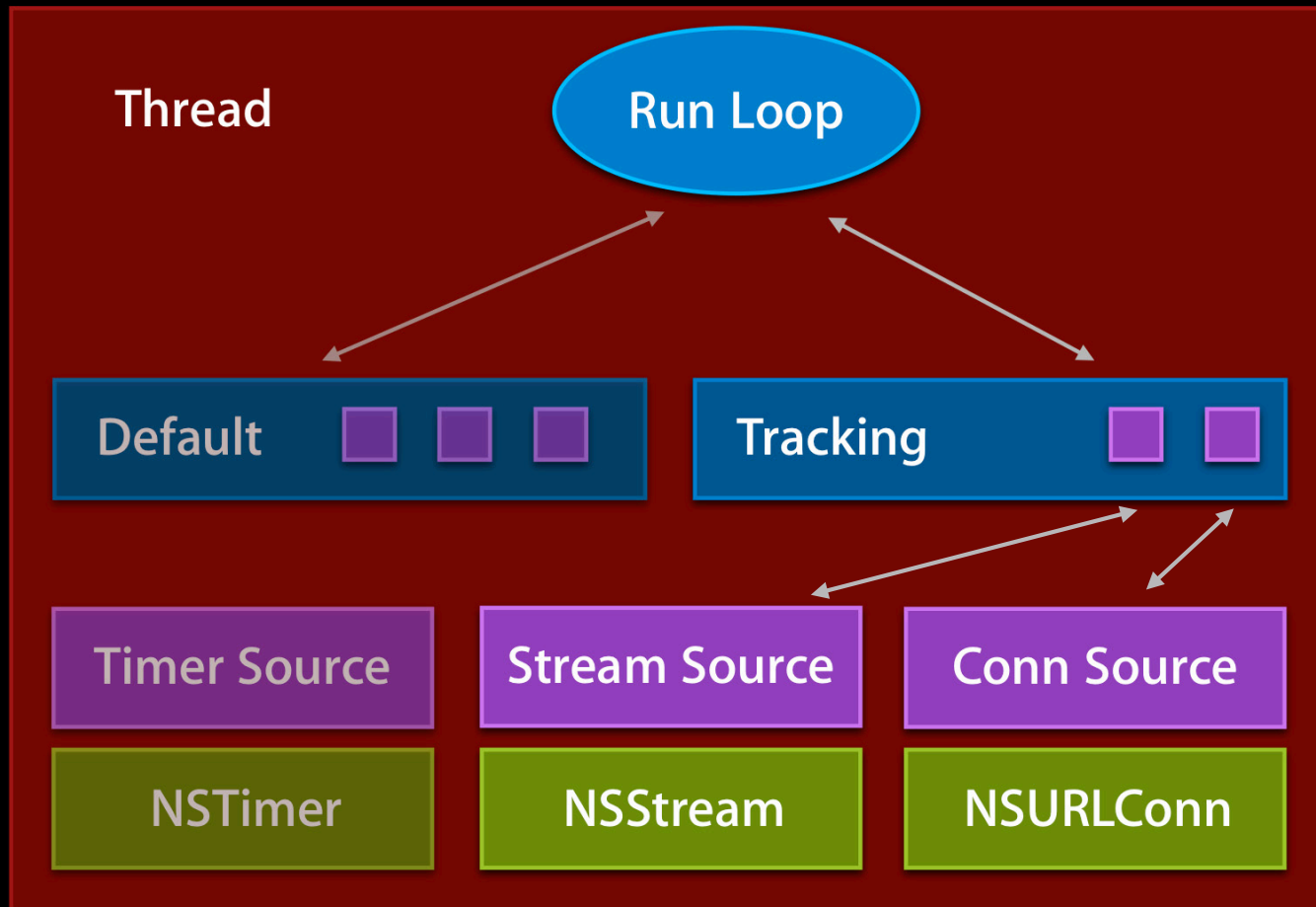
Run Loop Modes



Run Loop Modes



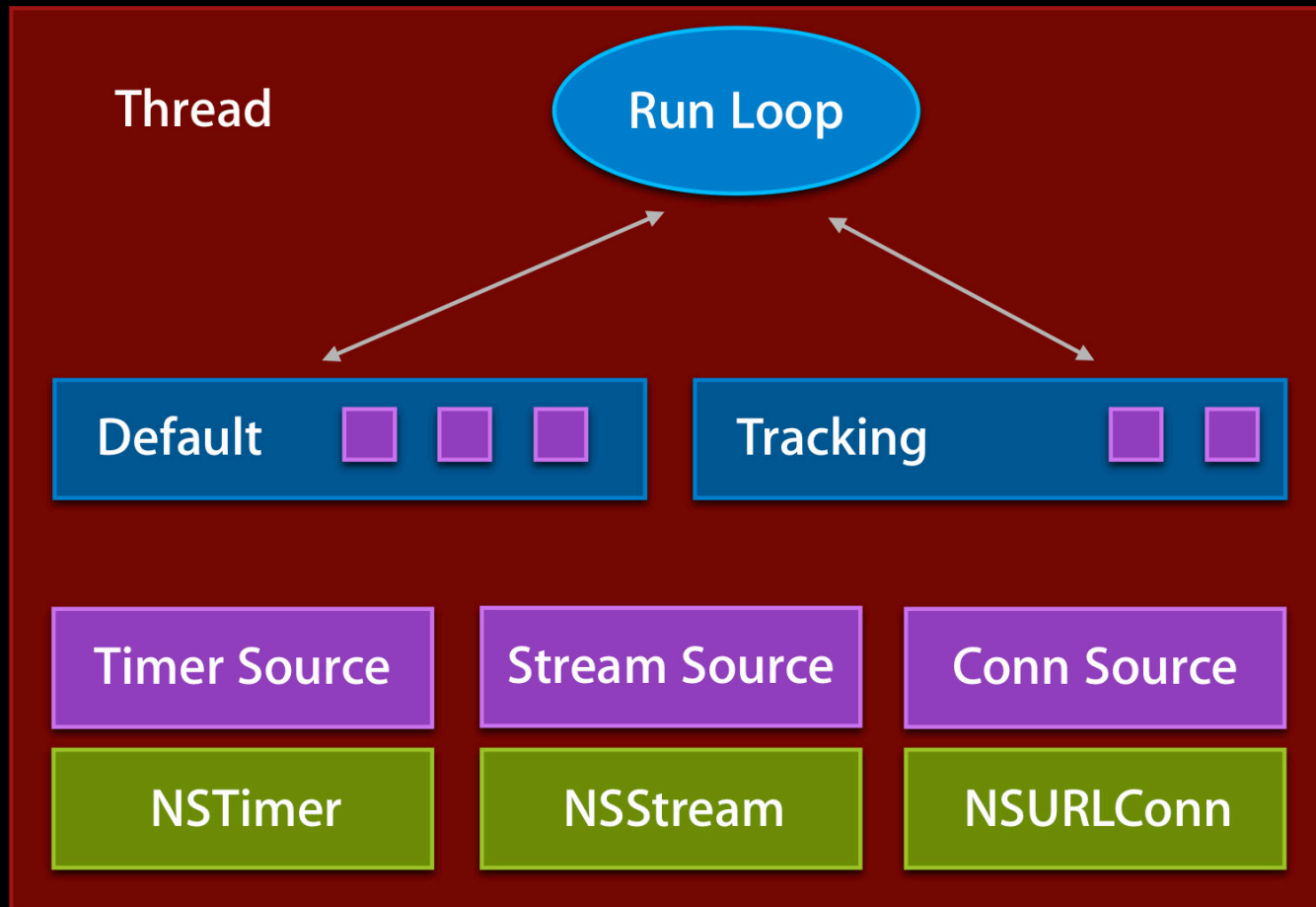
Run Loop Modes



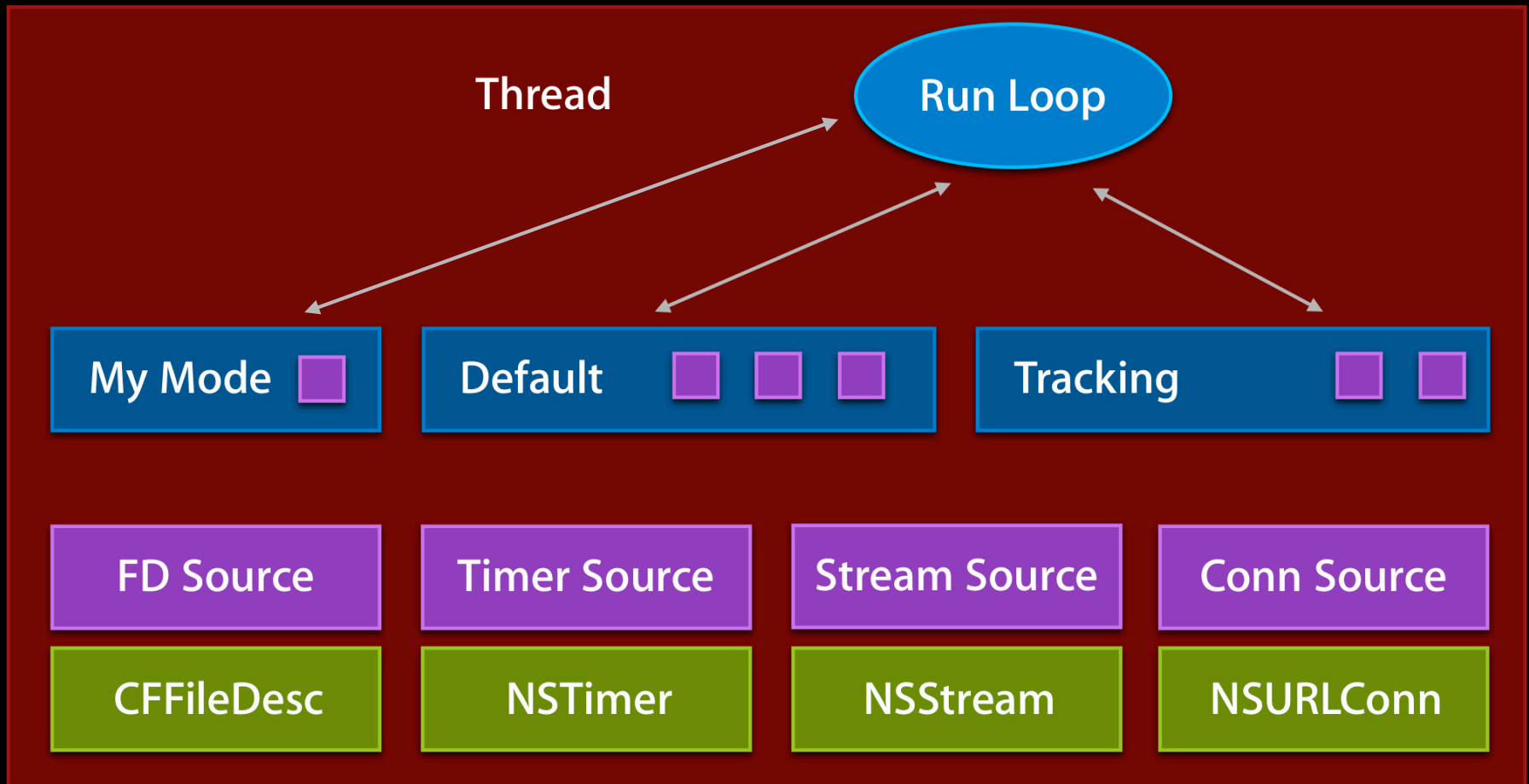
Why Run Loop Modes?

- It's all about recursion
- Synthetic synchronous
 - To run async APIs sync
 - Schedule in custom mode
 - Run in custom mode
- UI tracking

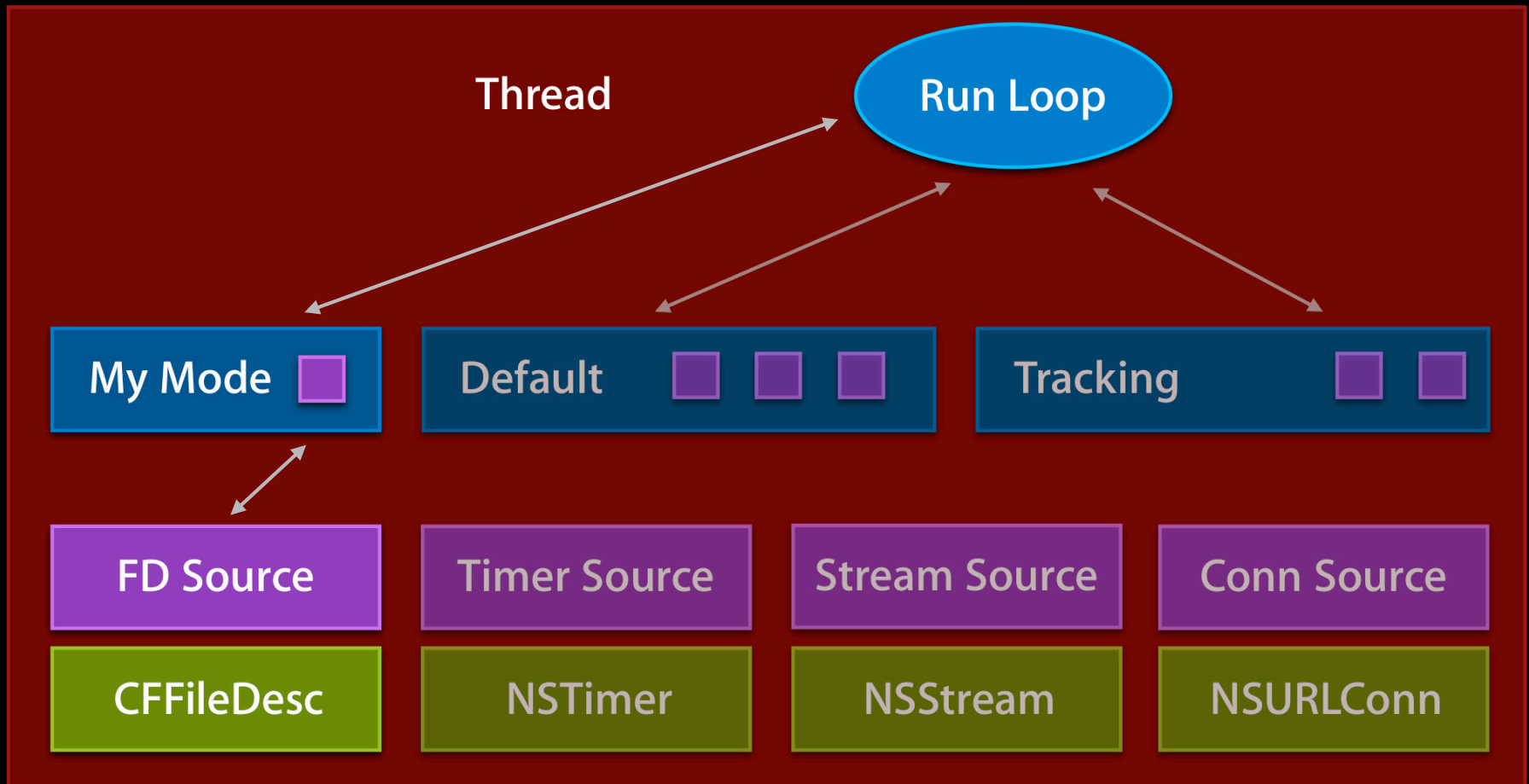
Run Loop Modes



Run Loop Modes



Run Loop Modes



UI Tracking

- Tracking scroll view
- Specific form of synthetic synchronous
 - Needs touch events, but not others
 - `UITrackingRunLoopMode`
- Common modes
 - `NSRunLoopCommonModes`
 - Meta mode when scheduling
 - Default, tracking mode, and so on
- Context issues

Run Loop Tips

- No create or destroy
- Invalidate your sources
- Avoid cross thread scheduling
- No recursion in default mode on main thread
- Serialization implies latency
 - Single secondary networking thread
- Beware hidden threads

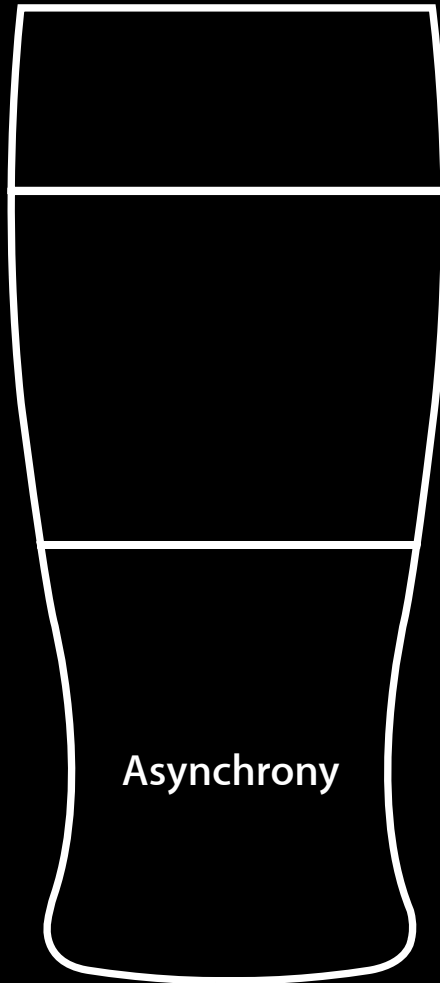
Beware Hidden Threads

```
...  
[self performSelectorInBackground:@selector(doStuff) withObject:nil];  
...
```

```
- (void)doStuff  
{  
    ... do stuff ...  
    (void) [NSTimer scheduledTimerWithTimeInterval:1.0  
        target:self  
        selector:@selector(doMoreStuff:)  
        userInfo:nil  
        repeats:NO  
    ];  
}
```

Asynchrony

- The basics
- Run loops
- State management



State Management, Unsolicited



State Management, Solicited



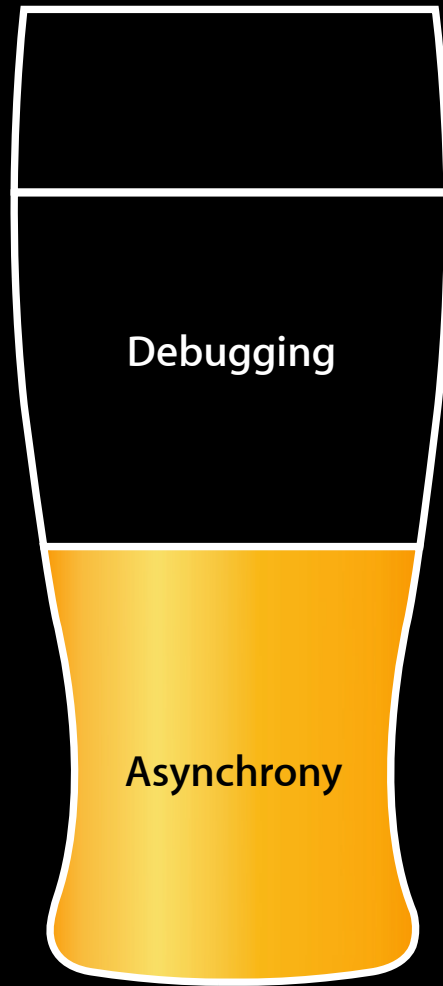
State Management Tips

- Asynchrony implies state management
- Don't fear it, plan for it!
- Hide irrelevant states
- Model notifications

Asynchrony



- The basics
- Run loops
- State management



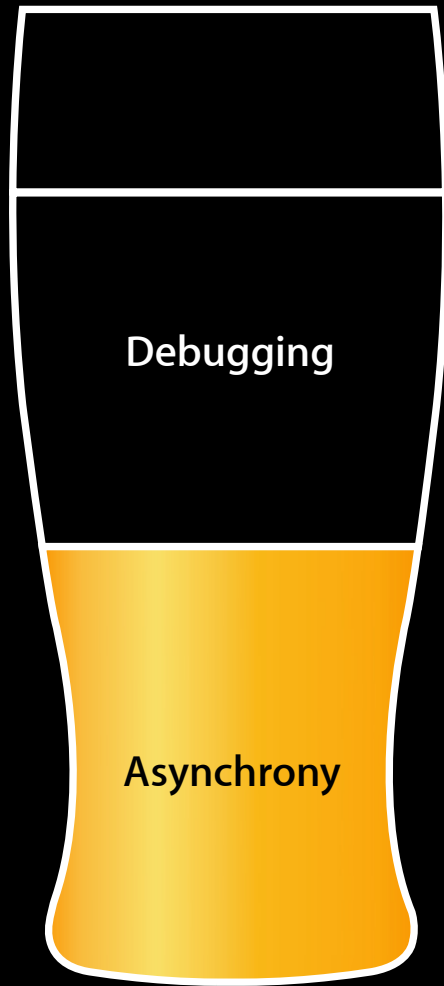


Don't Write Bugs

Don't Write Bugs

- Design
- Compiler warnings
- Static analyzer
- Asserts
- Zombies

Debugging



- Logging
- Packet trace
- Simulator

Debugging



- Logging
- Packet trace
- Simulator



Travel in Time & Space

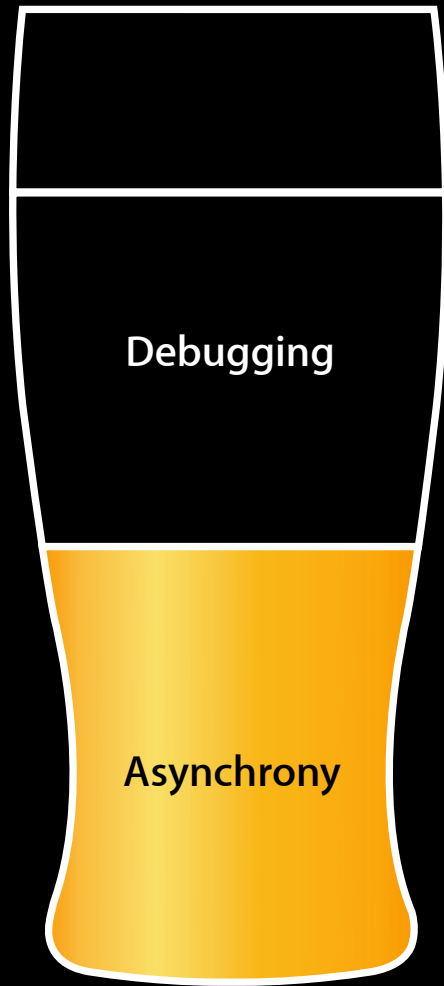
- Time
 - Non-deterministic
 - Real time
 - Replay time
- Space
 - In-the-field debugging



Logging Done Right

- Present but disabled
- User accessible
- Persistent
 - Limit your disk usage
- Easy to retrieve
- Apple System Log (Mac OS X)

Debugging



- Logging
- Packet trace
- Simulator

Packet Trace Benefits

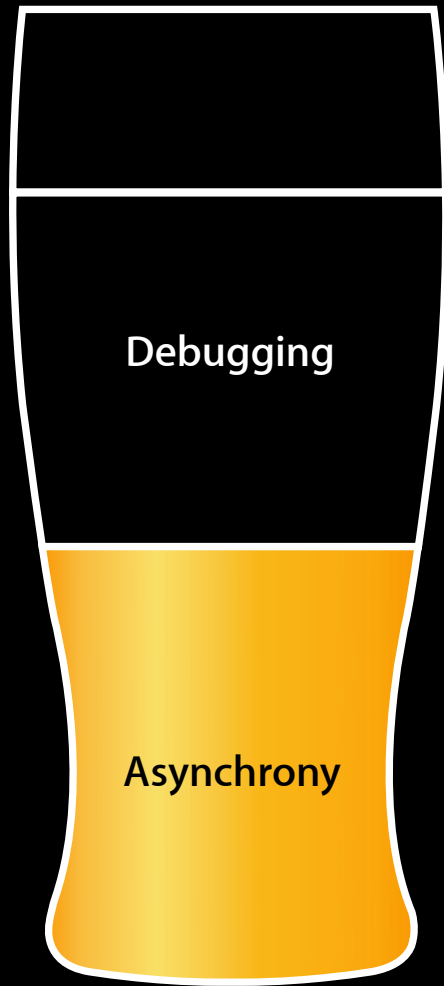
- Divide and conquer
- Comparison
- Verification
 - “Leaks” for the network
- Turn off TLS

Documentation

QA1176 Getting a Packet Trace

<http://developer.apple.com/mac/library/qa/qa2001/qa1176.html>

Debugging



- Logging
- Packet trace
- **Simulator**

The Simulator

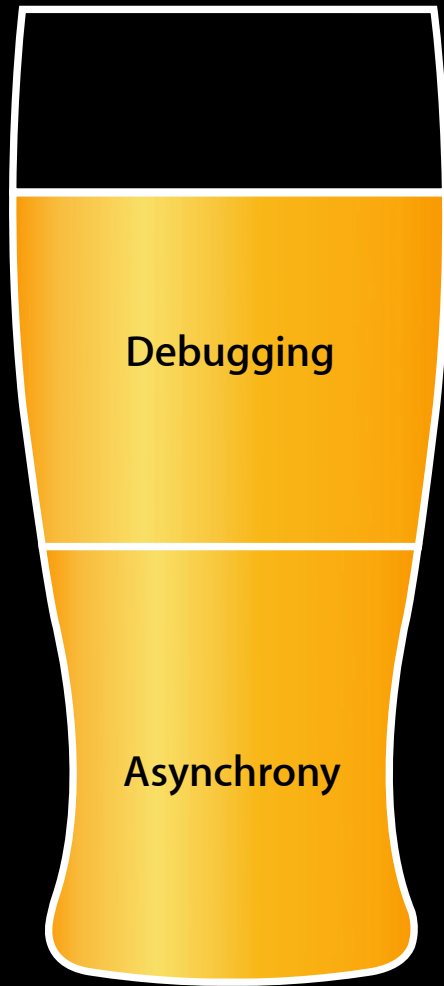
- Improved with 3.0
- Mac OS X debugging tools
 - DTrace!
- Uses the Mac OS X kernel
 - Obvious limitations

Documentation

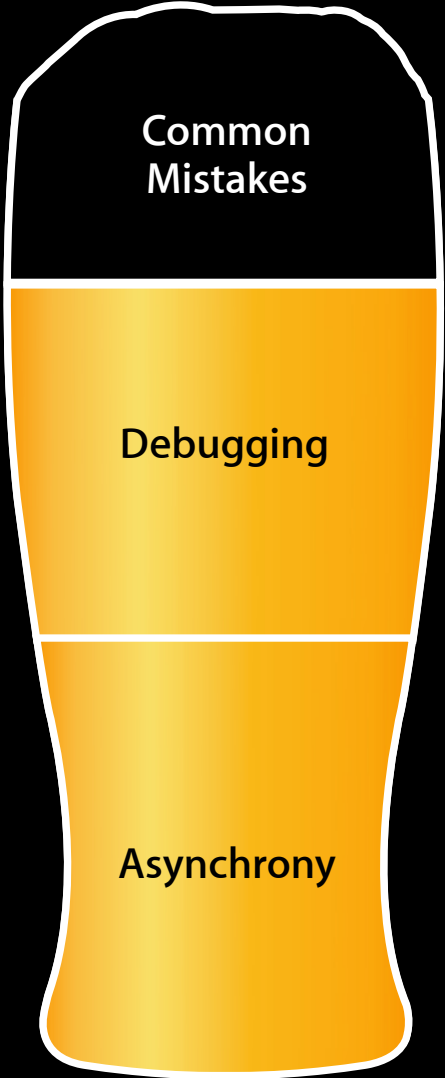
DTrace

<http://www.sun.com/bigadmin/content/dtrace/index.jsp>

Debugging



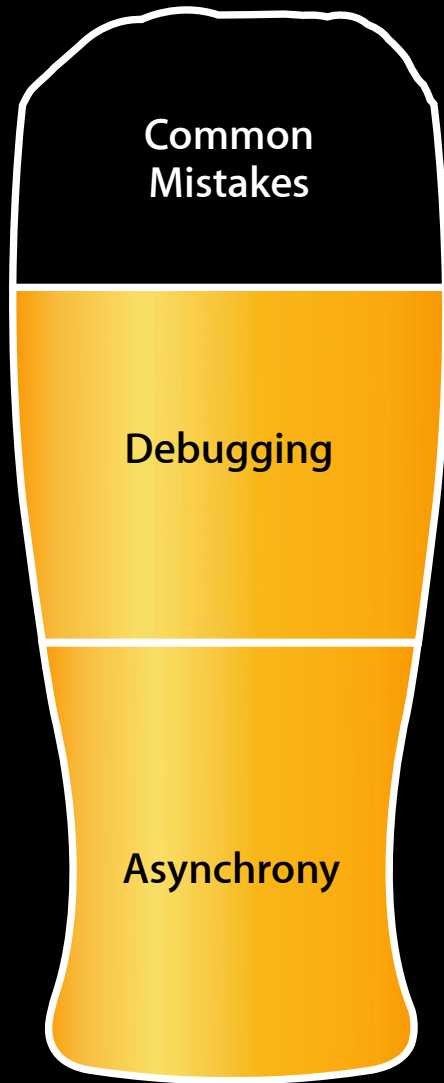
- Don't write bugs
- Logging
- Packet trace
- Simulator

A diagram of a beer glass filled with orange liquid, divided into three horizontal sections. The top section is black and contains the text 'Common Mistakes'. The middle section is orange and contains the text 'Debugging'. The bottom section is orange and contains the text 'Asynchrony'.

Common
Mistakes

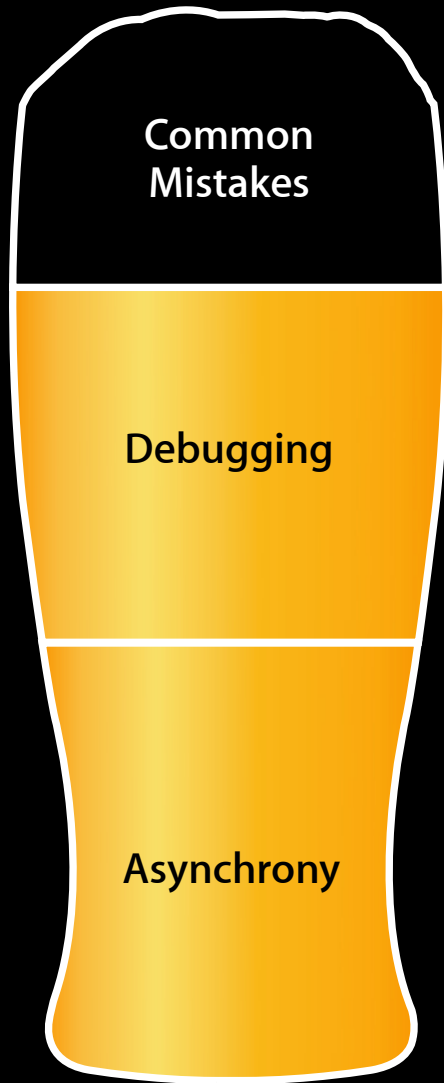
Debugging

Asynchrony



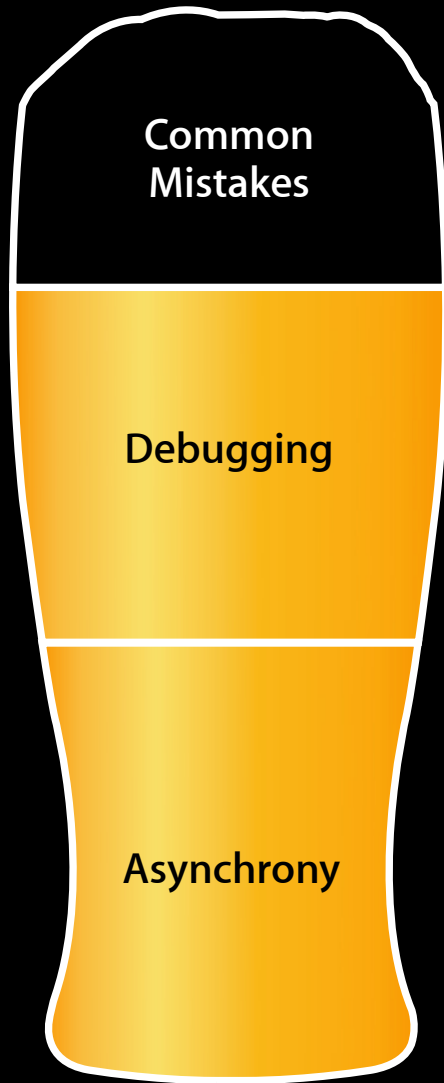
Common Mistakes

- Main thread synchronous
- Threads
- Interface lifecycle
- Reachability
- Interface type
- Timeouts



Common Mistakes

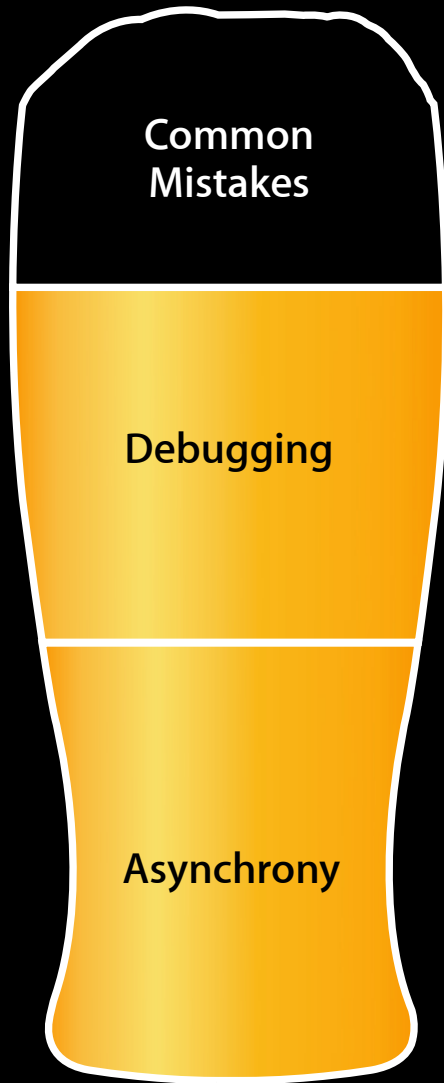
- Main thread synchronous
- Threads
- Interface lifecycle
- Reachability
- Interface type
- Timeouts



Common Mistakes

- Main thread synchronous
- **Threads**
- Interface lifecycle
- Reachability
- Interface type
- Timeouts

“Networking is hard enough without having to deal with threads as well.”



Common Mistakes

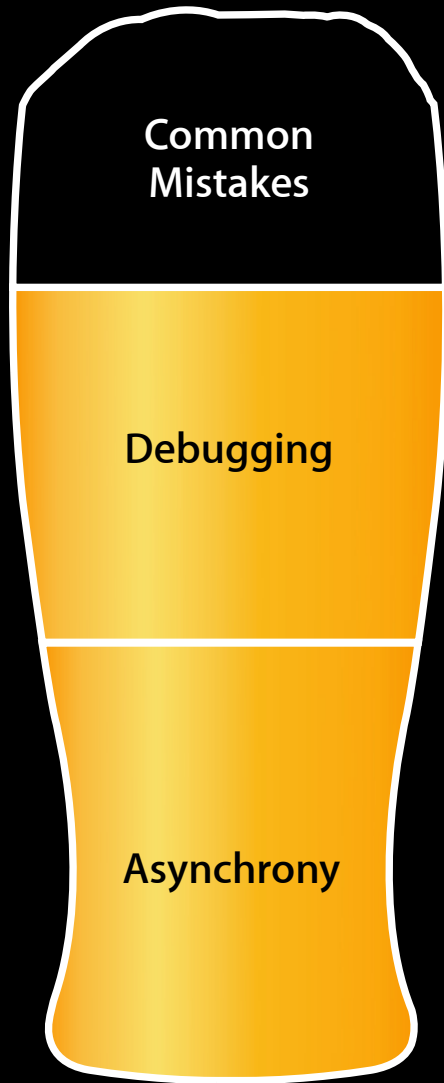
- Main thread synchronous
- Threads
- **Interface lifecycle**
- Reachability
- Interface type
- Timeouts

Interface Lifecycle

Interface	Goes Up	Goes Down
Bluetooth	Bonjour resolve	On idle
WWAN	CFSocketStream connect, other services	On idle, unless held up by other service
Wi-Fi	Complex	

Wi-Fi Interface Lifecycle

- Comes up
 - Known networks
 - “Ask to Join Networks”
 - `UIRequiresPersistentWiFi`
- Goes down
 - 30 minutes after Wi-Fi app
 - Screen lock
 - System sleep
- Captive networks



Common Mistakes

- Main thread synchronous
- Threads
- Interface lifecycle
- **Reachability**
- Interface type
- Timeouts



Preflight

Guide your UI

Interface type

Trigger retries

Interface specific connectivity

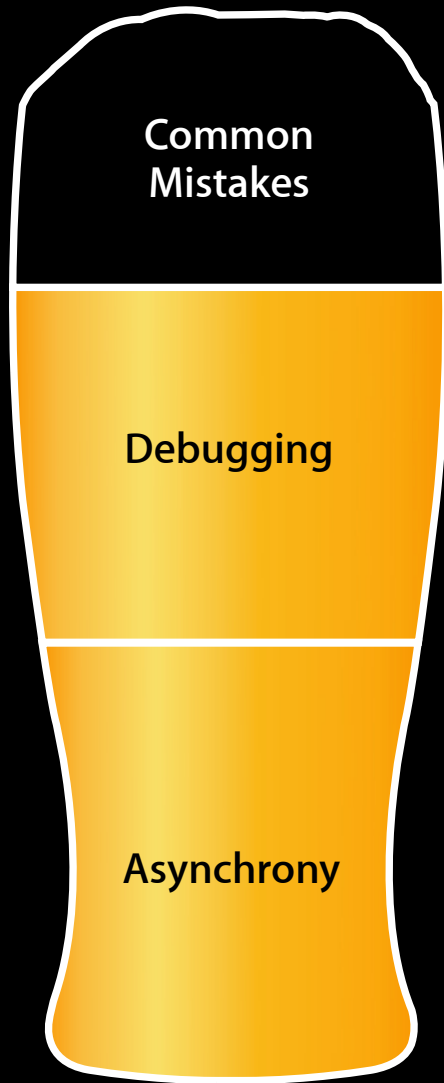
Reachability Tips

- Asynchronous
- Reachability sample
 - 2.0 or later

Sample Code

Reachability

<http://developer.apple.com/iphone/library/samplecode/Reachability>



Common Mistakes

- Main thread synchronous
- Threads
- Interface lifecycle
- Reachability
- **Interface type**
- Timeouts

“What type of interface am I on?”

J Random Developer

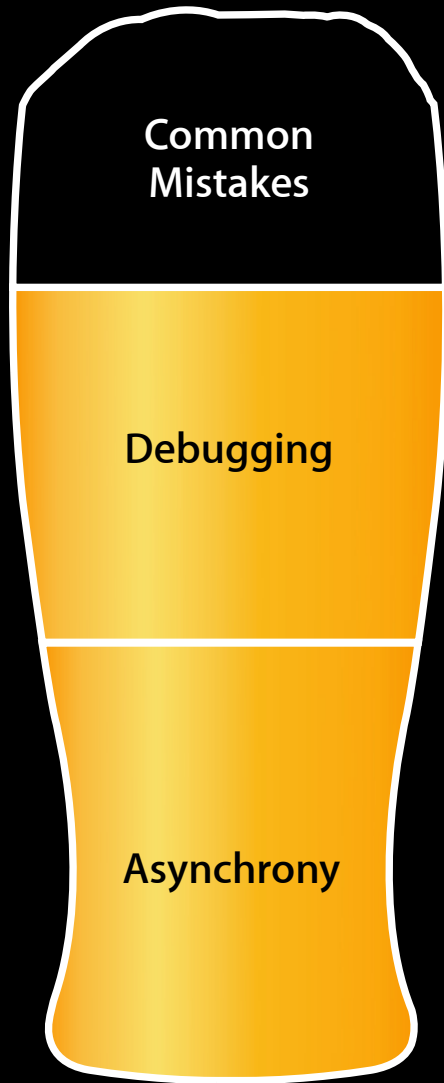
“What type of cellular am I on?”

J Random Developer

“What speed is this network?”

J Random Developer

**“If you need to know the speed,
measure it.”**

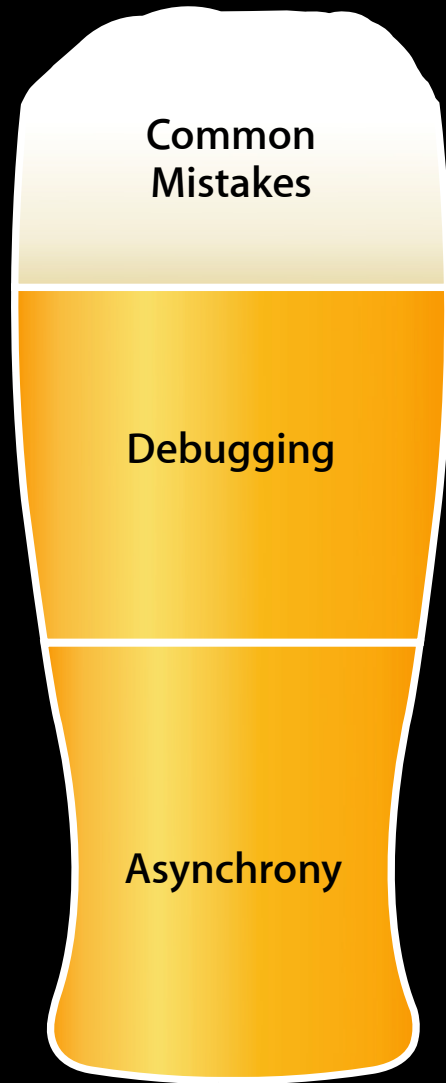


Common Mistakes

- Main thread synchronous
- Threads
- Interface lifecycle
- Reachability
- Interface type
- **Timeouts**

Timeouts

- Don't lower the timeouts
 - 60 second range
 - Critical on WWAN
- Cancellation UI for solicited operations
- Let unsolicited operations time out



Common Mistakes

- Main thread synchronous
- Threads
- Interface lifecycle
- Reachability
- Interface type
- Timeouts



Common
Mistakes

Debugging

Asynchrony

Summary

- Networking is still hard
- Good architecture is the key
- Understand asynchrony
- Plan for debugging
- Avoid the common mistakes

More Information

Quinn “The Eskimo!”

Developer Technical Support

eskimo1@apple.com

Paul Danbold

Dogsbody Evangelist

danbold@apple.com

Documentation

Networking

<http://developer.apple.com/networking/>

Apple Developer Forums

<http://devforums.apple.com>

More Information

Sample Code

SimpleNetworkStreams, SimpleURLConnections, AdvancedURLConnections, SimpleFTPSample, Reachability, WiTap, BonjourWeb

[iPhone Dev Center](#) > [iPhone Reference Library](#) > [Sample Code](#)

Sample Code

CocoaHTTPServer, CocoaEcho, UDPEcho, SimplePing

[Mac Dev Center](#) > [Mac OS X Reference Library](#) > [Sample Code](#)

Sample Code

LinkedImageFetcher

[WWDC attendee site](#)

Related Sessions

Network Apps for iPhone OS, Part 1

Pacific Heights
Wednesday 2:00PM

Core OS Networking

Pacific Heights
Tuesday 9:00AM

Simplifying Networking Using Bonjour

Nob Hill
Wednesday 10:15AM

Creating Secure Applications

Russian Hill
Tuesday 4:30PM

Understanding Crash Reports on iPhone OS

Nob Hill
Friday 9:00AM

Labs

Networking Lab

Core OS Lab B
Thursday 9:00AM

Networking Lab

Core OS Lab B
Friday 9:00AM

iPhone OS and Mac OS X Security Lab

Core OS Lab A
Thursday 2:00PM

Q&A



