Multipeer Networking

Rob Stearn

History
Overview
Code
!
Future

History


Chooser
AppleShare
Select a file server:
Macintosh mcbx
OK
User Name:
mcbx
AppleTalk
Active
Inactive

Overview

Multipeer networking

sends data *from one device to many*

Multipeer networking

sends data *from one device to many*

requires *no server infrastructure*

Multipeer networking

sends data *from one device to many*

requires *no server infrastructure*

sends over *Bluetooth, *Ad-Hoc or Infrastructure WiFi*

Multipeer networking

sends data *from one device to many*

requires *no server infrastructure*

sends over *Bluetooth, *Ad-Hoc or Infrastructure WiFi*

bridges across interfaces

Consider 3 iOS devices...

Consider 3 iOS devices...

Consider 3 iOS devices...

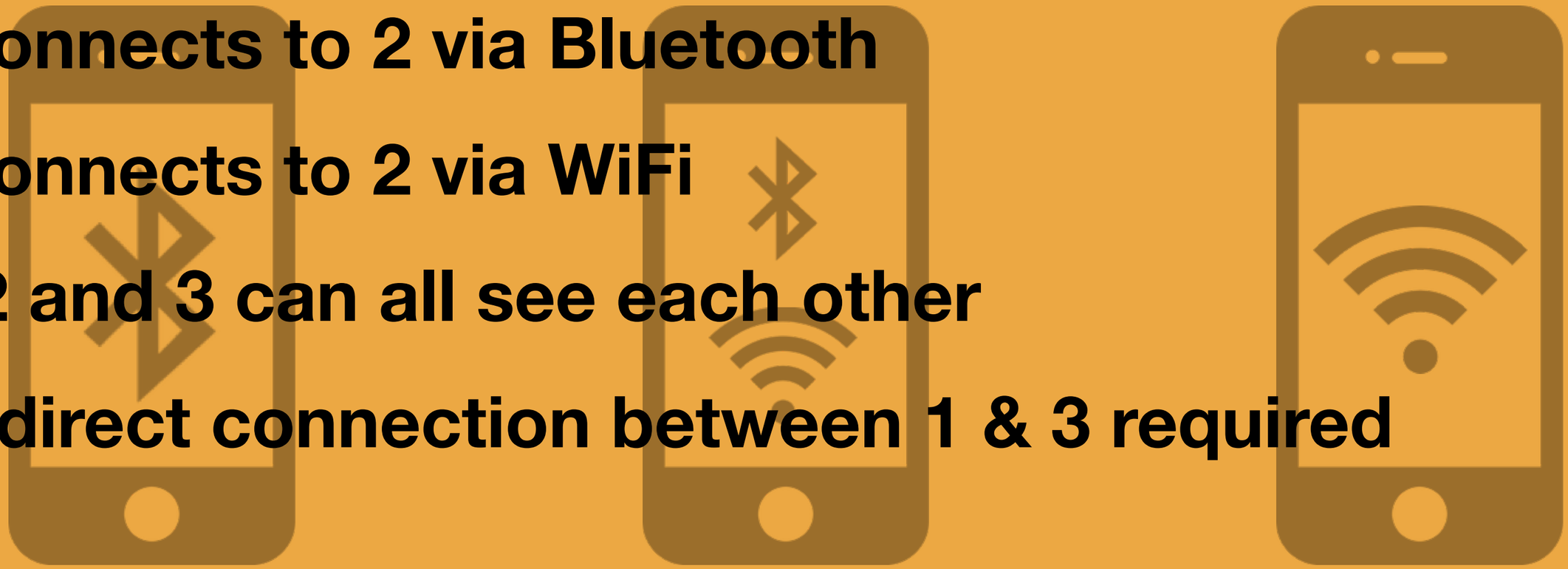
Consider 3 iOS devices...

1 connects to 2 via Bluetooth

3 connects to 2 via WiFi

1, 2 and 3 can all see each other

no direct connection between 1 & 3 required



Multipeer networking

sends data *from one device to many*

requires *no server infrastructure*

sends over *Bluetooth, *Ad-Hoc or Infrastructure WiFi*

bridges across interfaces

can send data as *Message, File or Stream*

Multipeer networking

sends data *from one device to many*

requires *no server infrastructure*

sends over *Bluetooth, *Ad-Hoc or Infrastructure WiFi*

bridges across interfaces

can send data as *Message, File or Stream*

can ensure *order and delivery of data*

Multipeer networking

sends data *from one device to many*

requires *no server infrastructure*

sends over *Bluetooth, *Ad-Hoc or Infrastructure WiFi*

bridges across interfaces

can send data as *Message, File or Stream*

can ensure *order and delivery of data*

API requires iOS7+

Discovery

Discovery

Zero-configuration networking over IP

Discovery

Zero-configuration networking over IP

Every peer acts as a DNS for itself

Discovery

Zero-configuration networking over IP

Every peer acts as a DNS for itself

Uses URLs with with the .local domain

Discovery

Zero-configuration networking over IP

Every peer acts as a DNS for itself

Uses URLs with with the .local domain

Uses multicast DNS to discover services

Discovery

Zero-configuration networking over IP

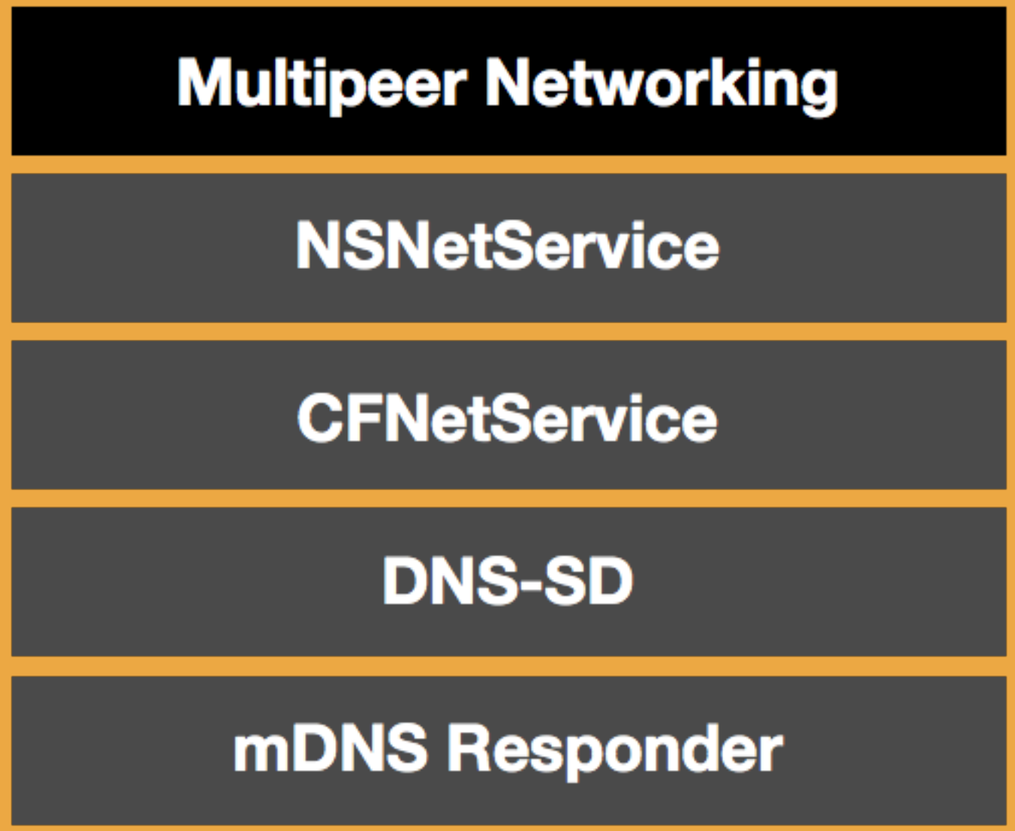
Every peer acts as a DNS for itself

Uses URLs with with the .local domain

Uses multicast DNS to discover services

Peers self-assign an IP in the `169.254.xxx.xxx` range

Discovery APIs



Code

3 Objects:
Session
Browser
Advertiser

Session, Browser and Advertiser

Session, Browser and Advertiser

Session, Browser and Advertiser

Session, Browser and Advertiser

Session, Browser and Advertiser

Session, Browser and Advertiser

kMCSessionMaximumNumberOfPeers

kMCSessionMaximumNumberOfPeers

Bluetooth: 8

kMCSessionMaximumNumberOfPeers

Bluetooth: 8

Wi-fi: 8?

Session

Browser

Advertiser

Sending data

Session / MCPeerID

```
MCPeerID *peerID =
[[MCPeerID alloc]
initWithDisplayName:@"cocoad
elica"];
```

Session / Initialize

```
self.session = [[MCSession alloc]
initWithPeer:peerID securityIdentity:nil*
encryptionPreference:
MCEncryptionRequired**];
```

Session / Initialize

```
self.session = [[MCSession alloc]
initWithPeer:peerID securityIdentity:nil*
encryptionPreference:
MCEncryptionRequired**];
```

* **optional Array with** `SecIdentityRef` **and** `SecCertificateRef` **items**

Session / Initialize

```
self.session = [[MCSession alloc]
initWithPeer:peerID securityIdentity:nil*
encryptionPreference:
MCEncryptionRequired**];
```

* **optional Array with** `SecIdentityRef` **and** `SecCertificateRef` **items**

** **choose from** `MCEncryptionRequired`, `MCEncryptionOptional` **or** `MCEncryptionNone`

Session / Delegate

```
didChangeState:
didReceiveCertificate:
didReceiveData:
didStartReceivingResourceWithName:
didFinishReceivingResourceWithName:
didReceiveStream:
```

Session / Details

Foreground operation only

Session / Details

Foreground operation only

Disconnects on breakpoints

Session / Details

Foreground operation only

Disconnects on breakpoints

Good candidate for a singleton*

Browser / Initialize

```
MCBrowserViewController *bvc =
[[MCBrowserViewController alloc]
initWithServiceType:(NSString
*)serviceType session:(MCSession
*)session];
```

Browser / Initialize

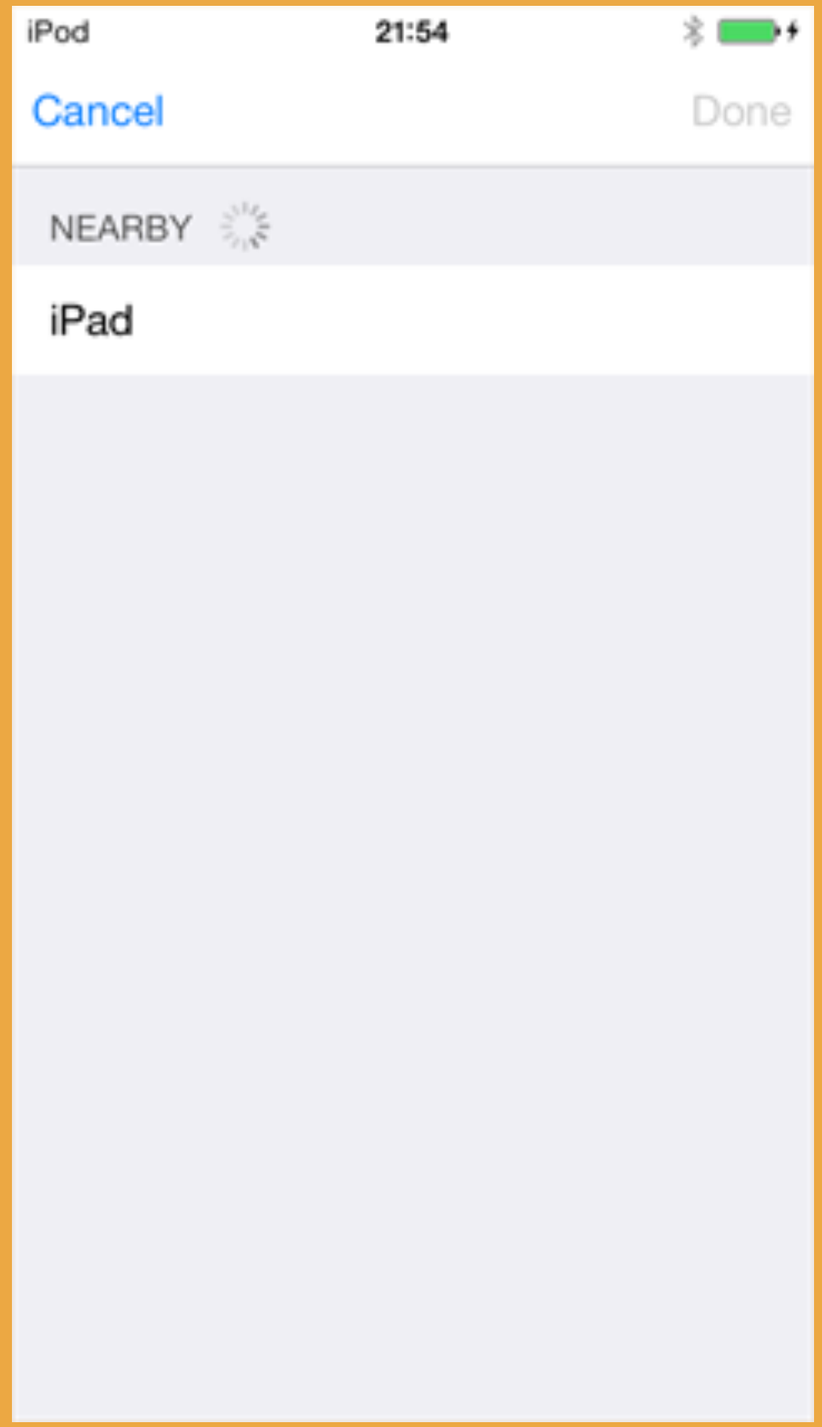
```
MCBrowserViewController *bvc =
[[MCBrowserViewController alloc]
initWithServiceType:(NSString
*)serviceType session:(MCSession
*)session];
```

Use a 1-15 character string for service type

Check IANA domain naming conventions

Browser

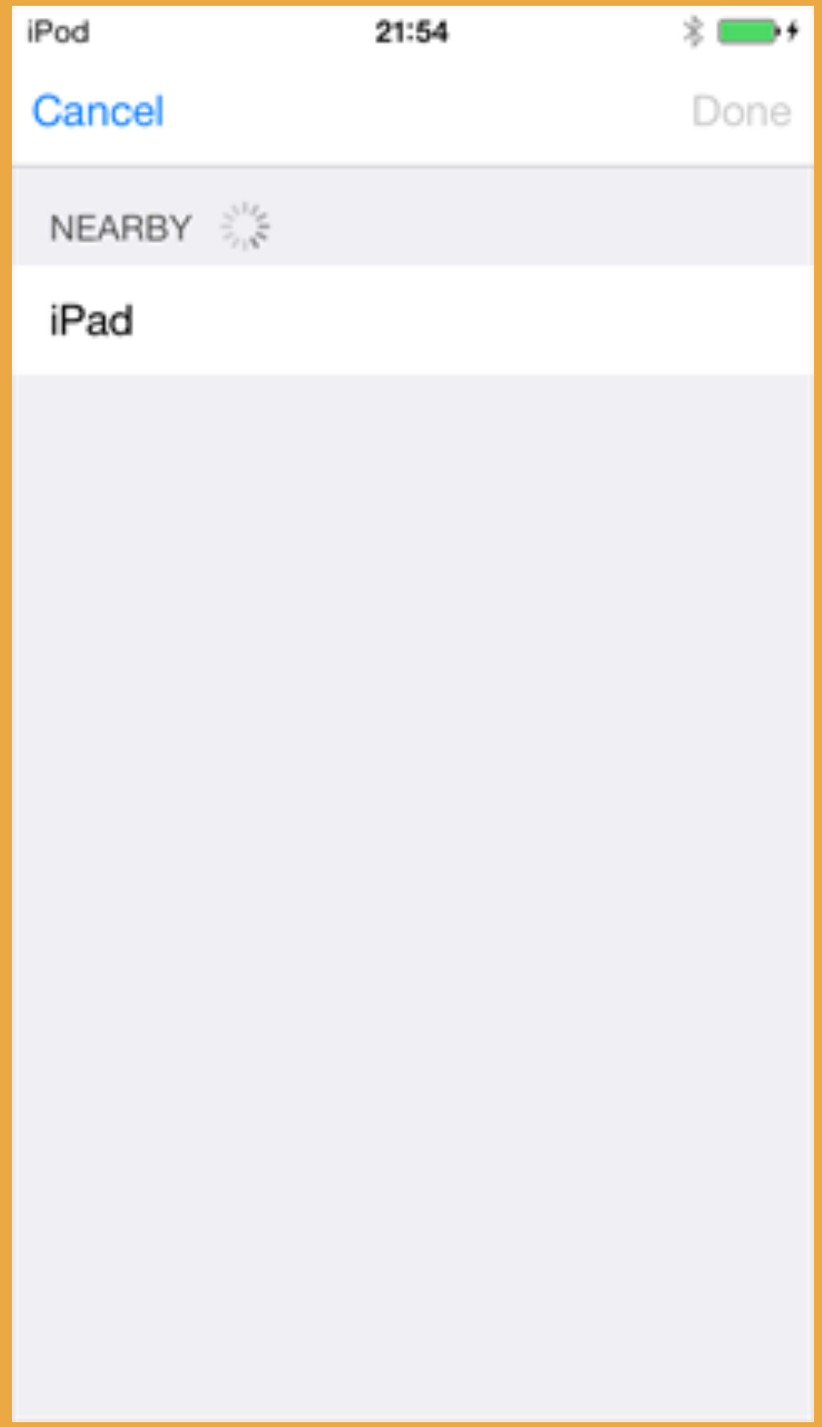
"Easy mode"



Browser

"Easy mode"

boxed solution

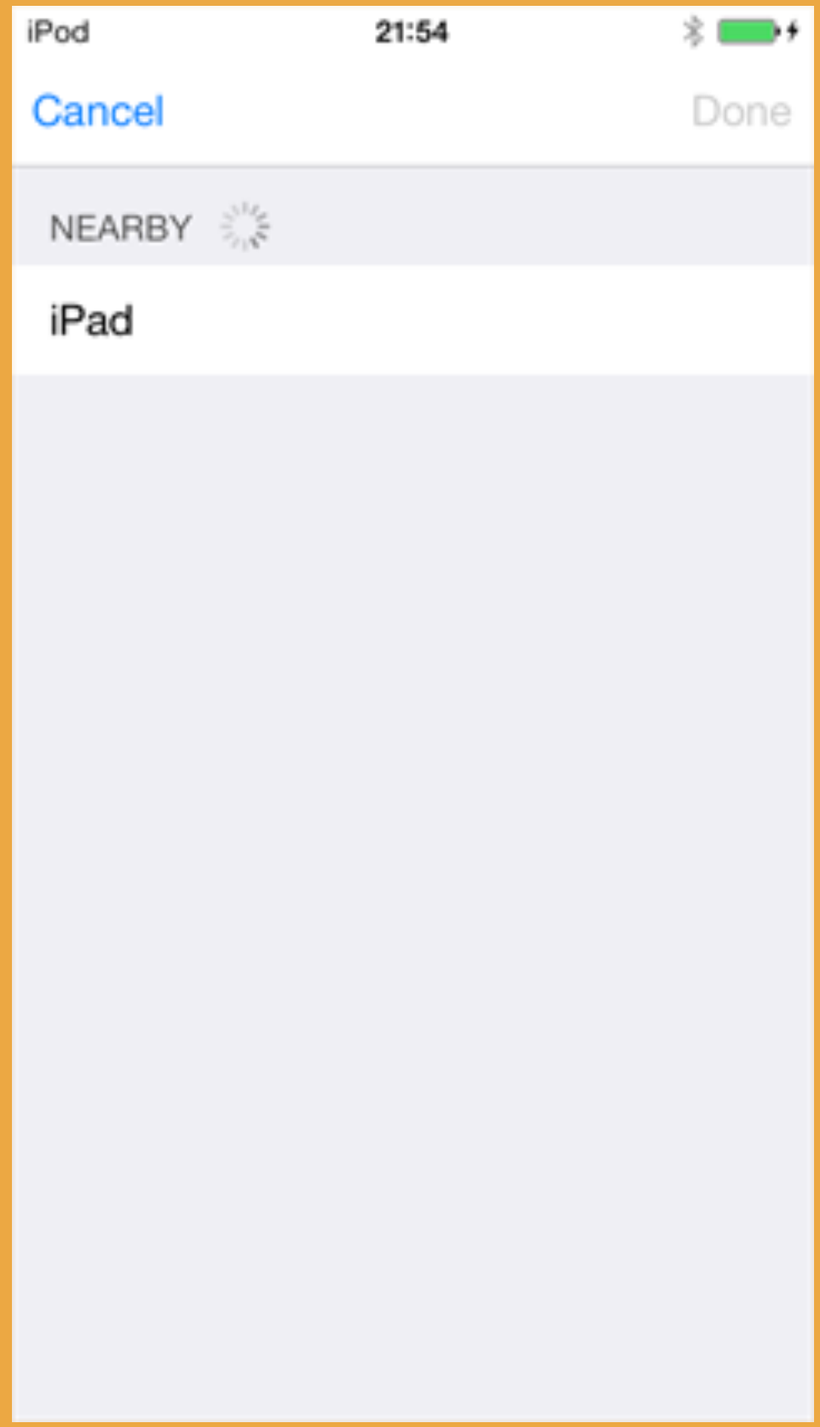


Browser

"Easy mode"

boxed solution

minimal styling



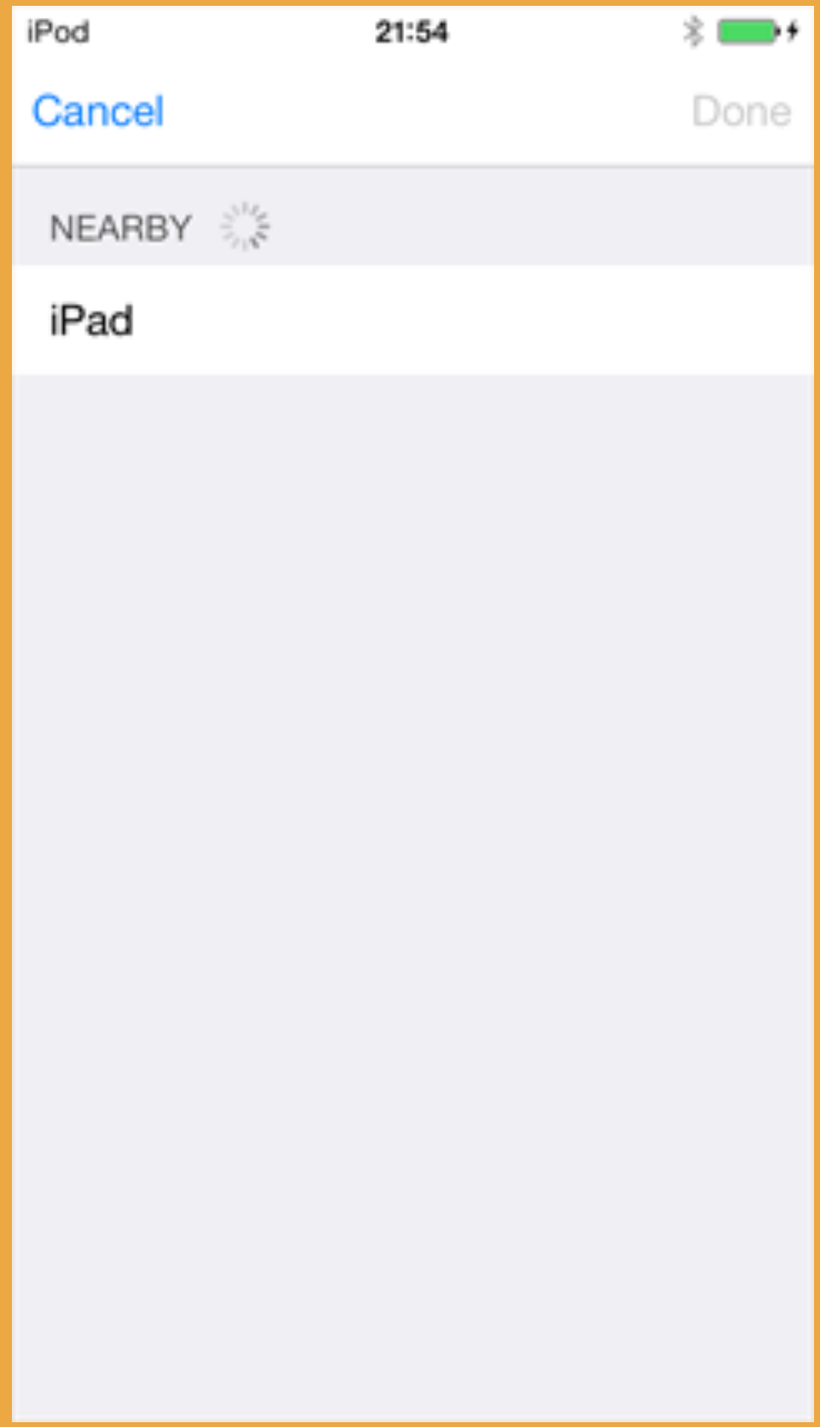
Browser

"Easy mode"

boxed solution

minimal styling

awesome!



Browser

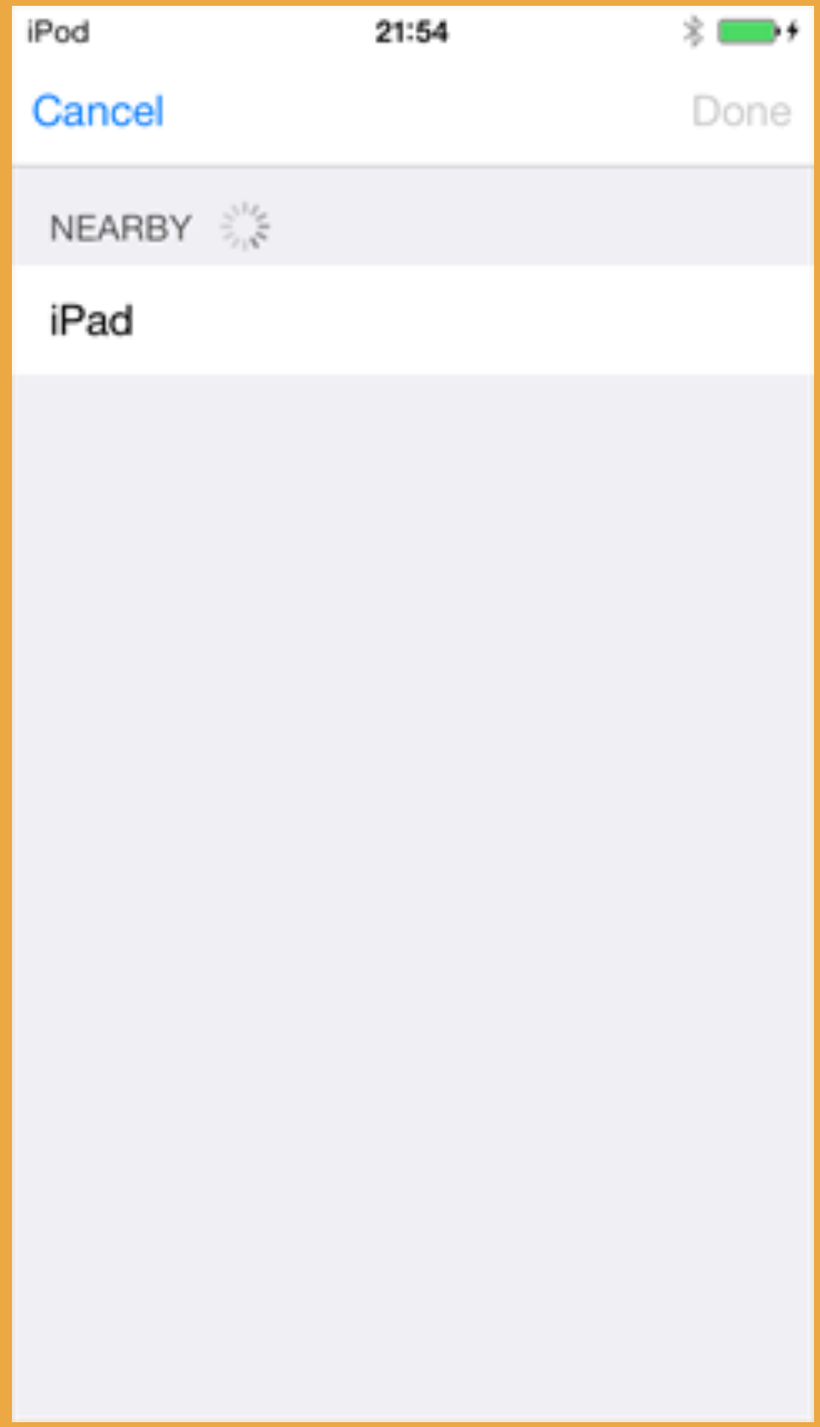
"Easy mode"

boxed solution

minimal styling

awesome!

a bit dull



Browser / Delegate

`MCBrowserViewControllerDelegate`

browserViewController:shouldPresentNearbyPeer:withDiscoveryInfo:

browserViewControllerDidFinish:

browserViewControllerWasCancelled:

Browser / Details

"Hard Mode"

Build a custom UI

Use `MCNearbyServiceBrowser` **for data and callbacks**

Browser / 'Hard Mode'

`MCNearbyServiceBrowserDelegate`

didNotStartBrowsingForPeers:

foundPeer:withDiscoveryInfo:

lostPeer:

Advertiser / Initialize

```
[[MCAdvertiserAssistant alloc]
initWithServiceType:service
discoveryInfo:discoveryDict
session:mySession];
```

Advertiser / Initialize

```
[[MCAdvertiserAssistant alloc]
initWithServiceType:service
discoveryInfo:discoveryDict
session:mySession];
```

`discoveryInfo` **is your opportunity to add additional info and context about your user.**

Advertiser / discoveryInfo

NSDictionary

Keys and Values must be of type NSString

Max size per key/value pair of 256 bytes

Advertiser / Details

"Hard Mode"

Build a custom UI for invitations

Use `MCNearbyServiceAdvertiser` **advertising & callbacks**

Advertiser / 'Hard Mode'

```
MCNearbyServiceAdvertiserDelegate

advertiser: didNotStartAdvertisingPeer:

advertiser:
didReceiveInvitationFromPeer:
withContext: invitationHandler:
```

Sending data

Messages

Resources

Streams

Messages

Data with known bounds, serialized into `NSData` objects, sent atomically

```
sendData: toPeers: withMode: error:
```

```
session: didReceiveData: fromPeer:
```

Messages can be sent as either:

`MCSessionSendDataReliable` **which guarantees delivery and order**

Messages can be sent as either:

`MCSessionSendDataReliable` **which guarantees delivery and order**

`MCSessionSendDataUnreliable` **which does not**

Messages can be sent as either:

`MCSessionSendDataReliable` **which guarantees delivery and order**

`MCSessionSendDataUnreliable` **which does not**

Analogy to TCP/UDP

Resources

Text files, File URLs or Web URLs

```
sendResourceAtURL: withName: toPeer:
withCompletionHandler:
```

Callbacks to start and finish transfer, uses `NSProgress`

Streams

Unbounded data, uses NSStreams

`startStreamWithName: toPeer: error:`
returns an `NSOutputStream`

`session: didReceiveStream: withName: fromPeer:` **gives an** `NSInputStream`

Streams

For both input and output streams:

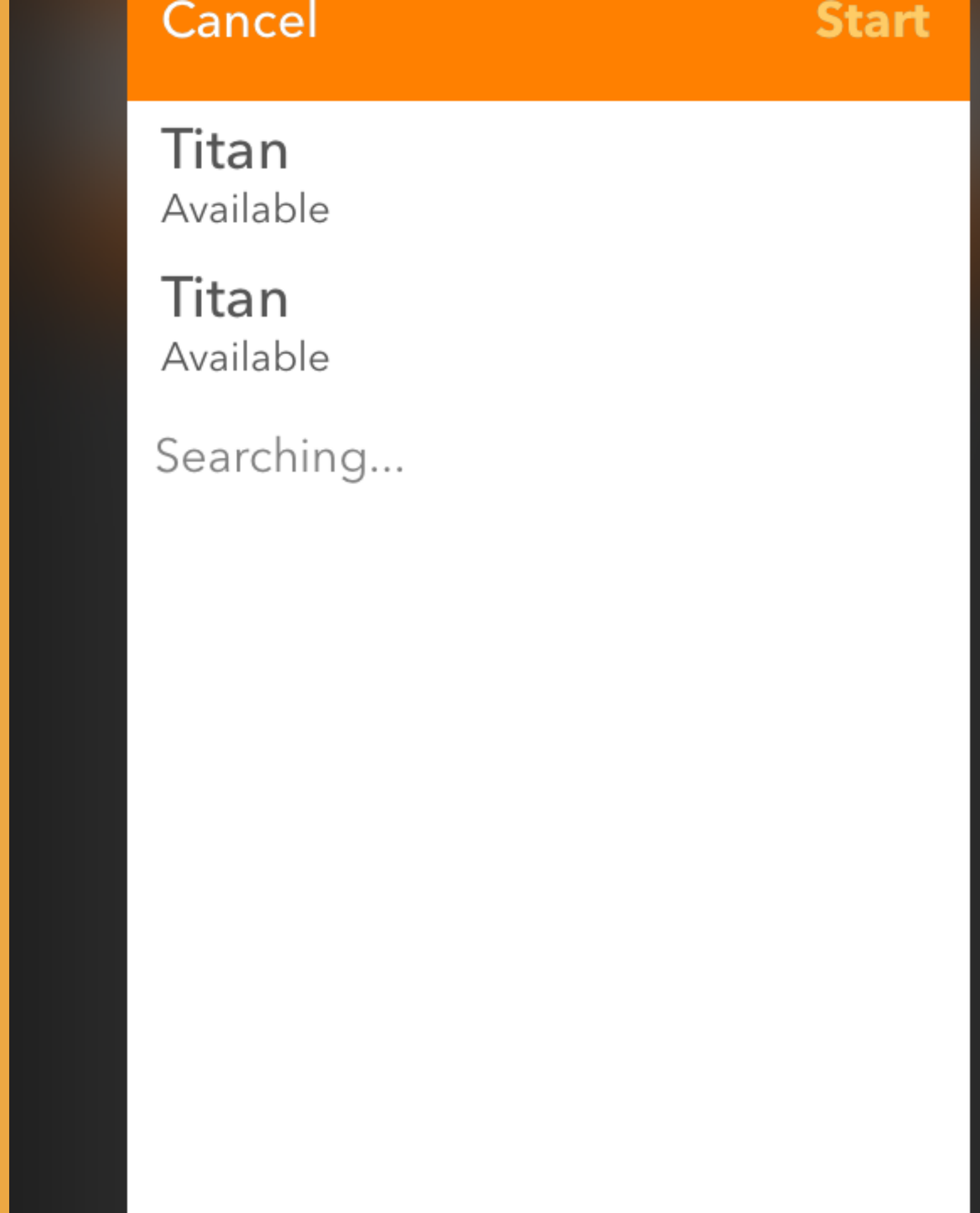
> Add stream to a run loop

> Open the stream

> Set a delegate and respond to `stream:handleEvent:` **callback**

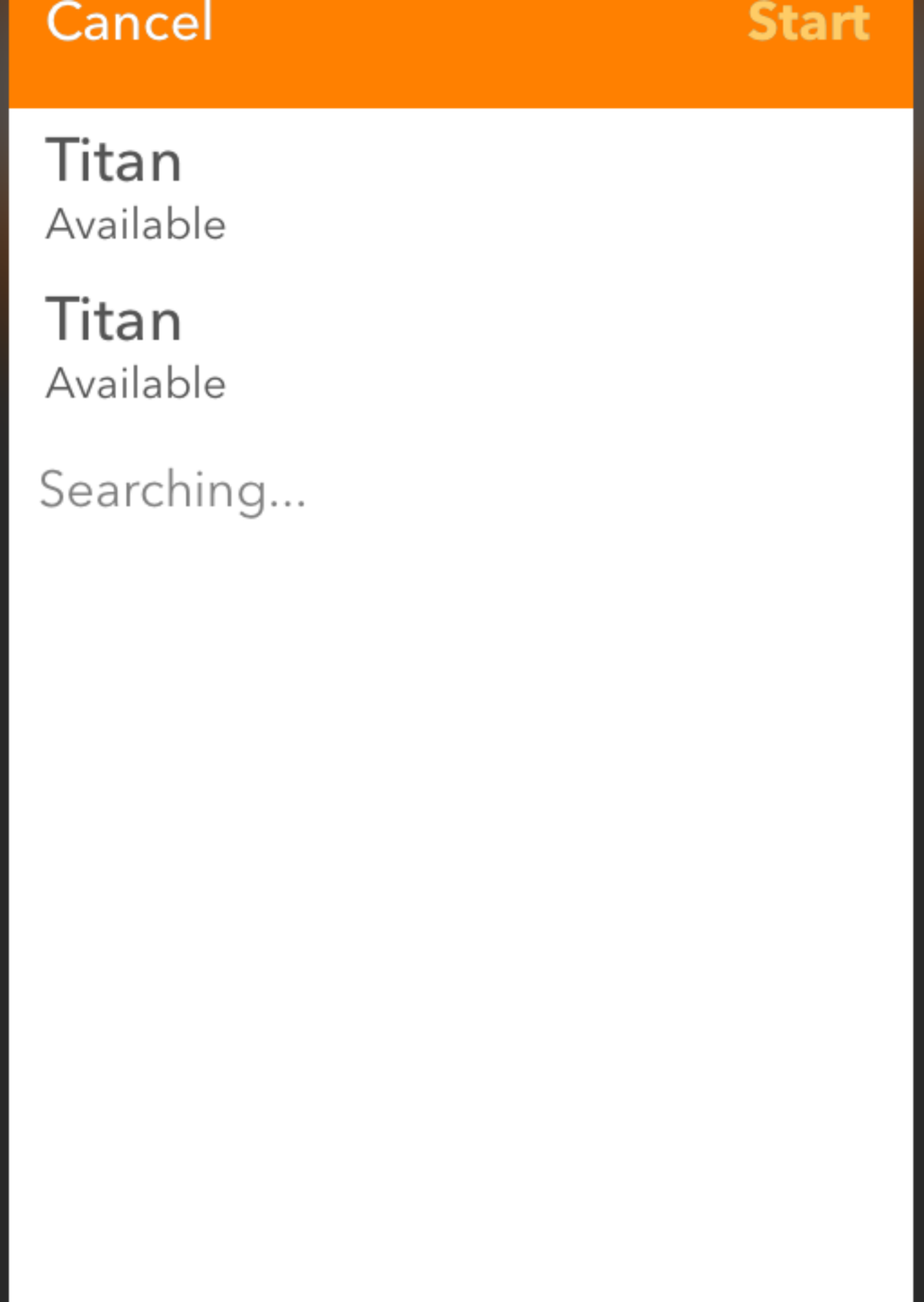

AppleACPIPlatformExport/PC1000/AppleACPIPCI/SATA@1F,2/AppleIntelPchSeriesAHCI/PR1000/IOAHCIDevice@0/Apple
driver/SAMSUNG SSD 830 Series Media/IOGUIDPa
Return Address
backtrace:
com.apple.CoreStorage(274.7)[5A5DC088-8553-3485-B461-B7B95A7ED734]0xffffff7fa221e000->0xffffff7fa22d5fff
dependency: com.apple.iokit.IOStorageFamily(1.8)[5BA4CD36-E96D-3A9E-ADFF-A863B8D63BC7]0xffffff7fa15c5000
Sat Aug 25 00:48:52 PDT 2012; root:xnu-2050.18.24~1/RELEASE_X86_64
MacBookPro8,1 (Mac-942459F5819B171B)

Duplicates in the browser



Duplicates in the browser

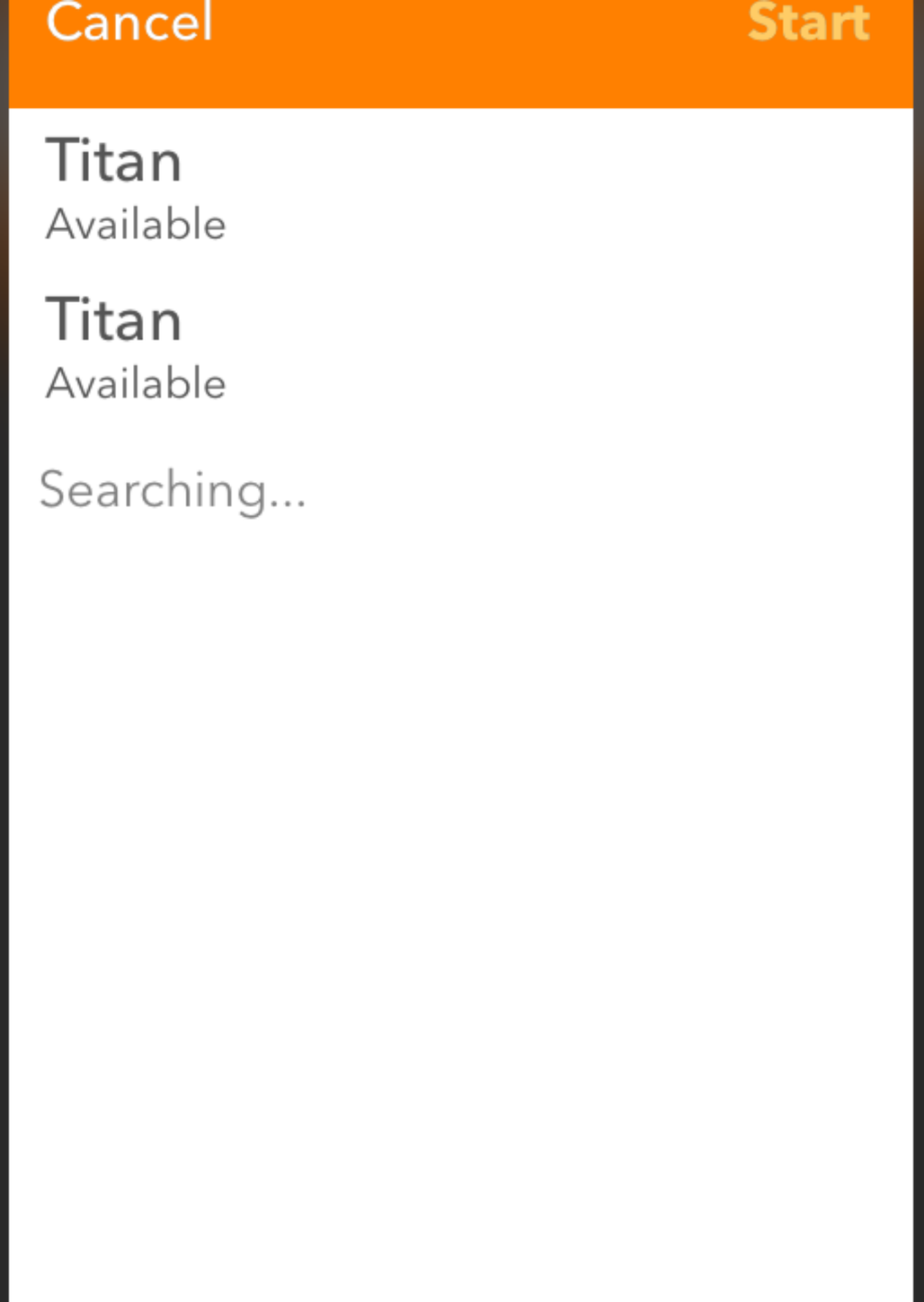
Issue: `MCPeerID` **created each time** `MCSession` **is instantiated**



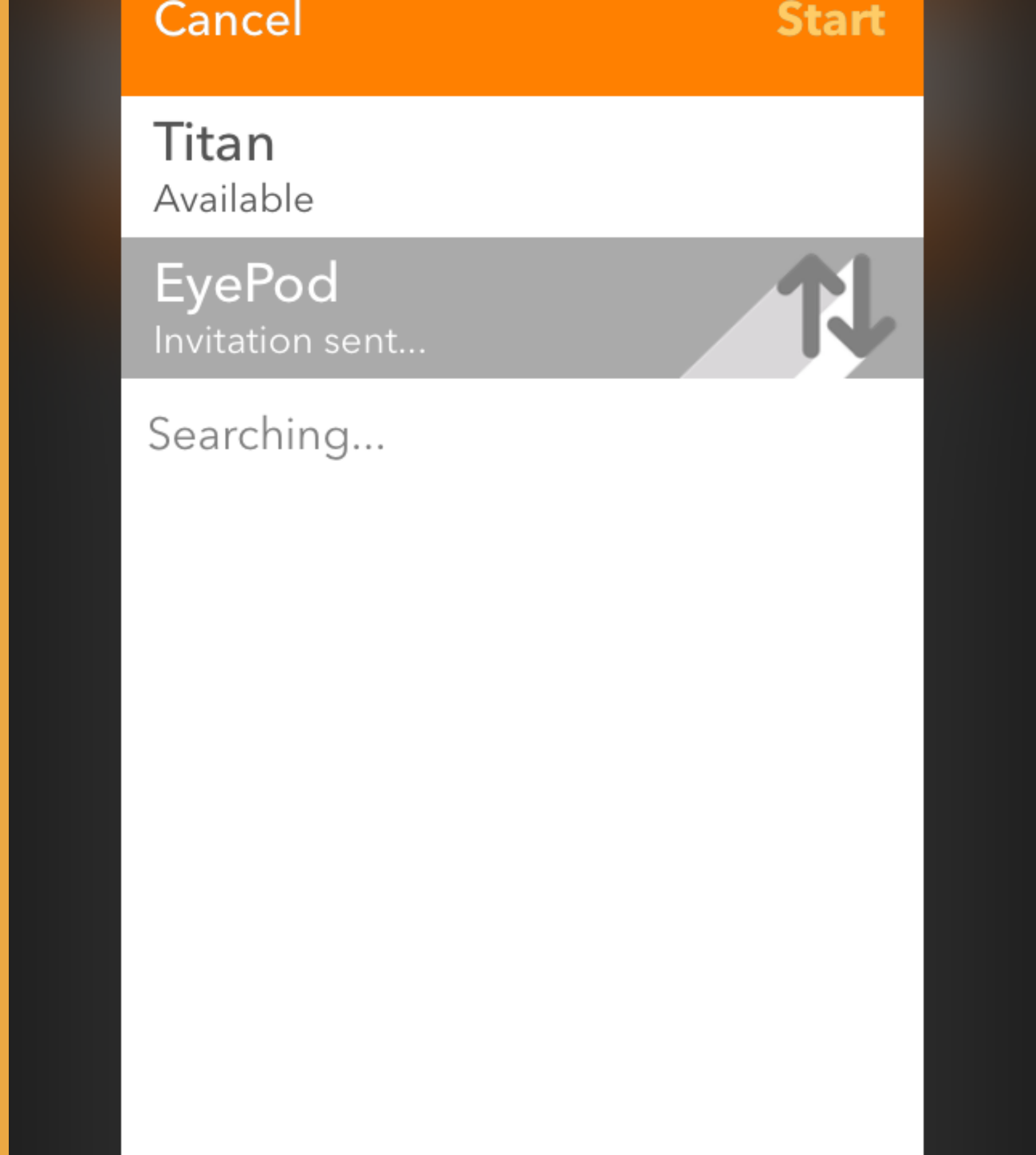
Duplicates in the browser

Issue: `MCPeerID` **created each time** `MCSession` **is instantiated**

Solution: Each peer should serialize its own `MCPeerID` **with** `NSKeyedArchiver`**, save it to disk and reuse for each new session.**

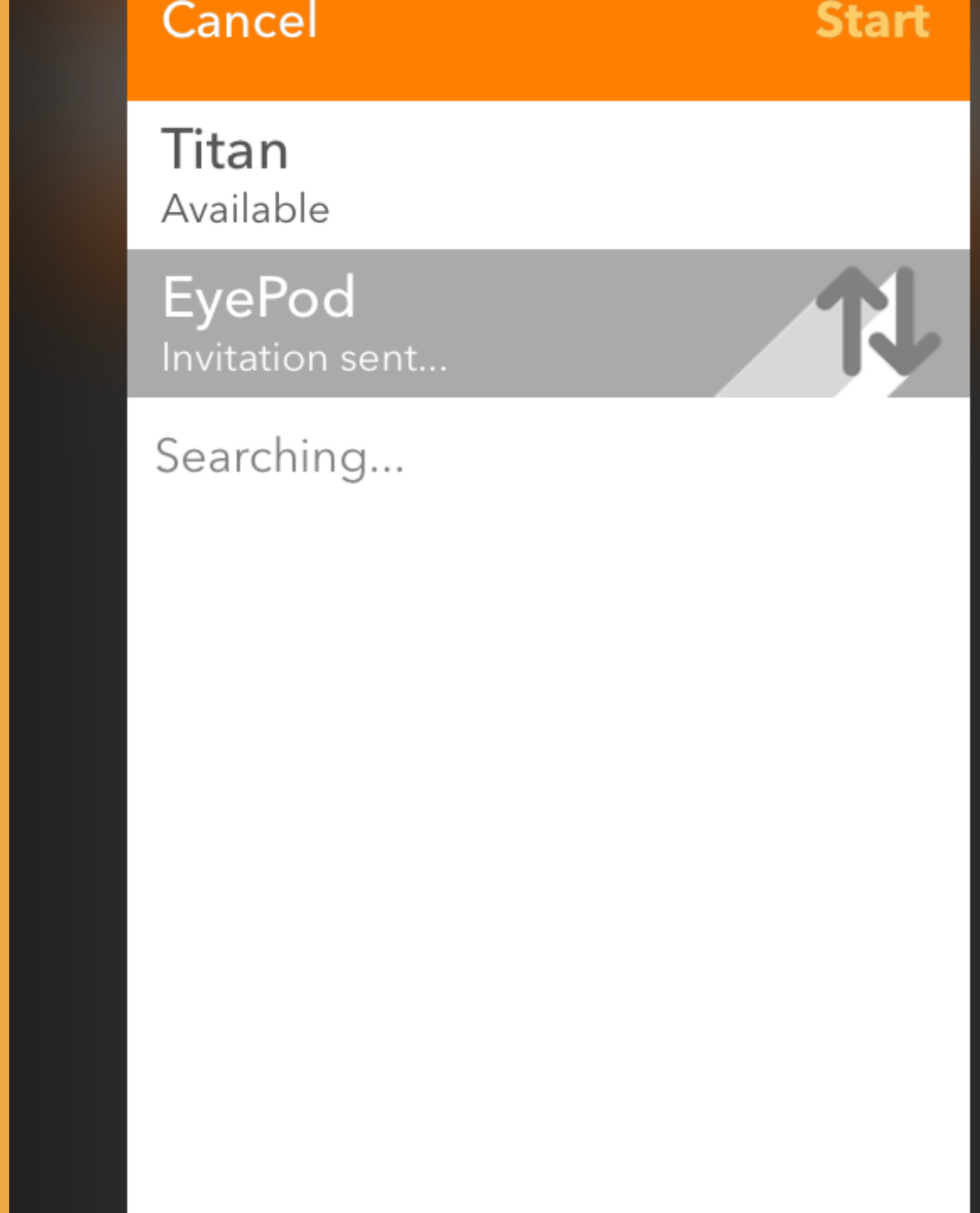


Invitation response times



Invitation response times

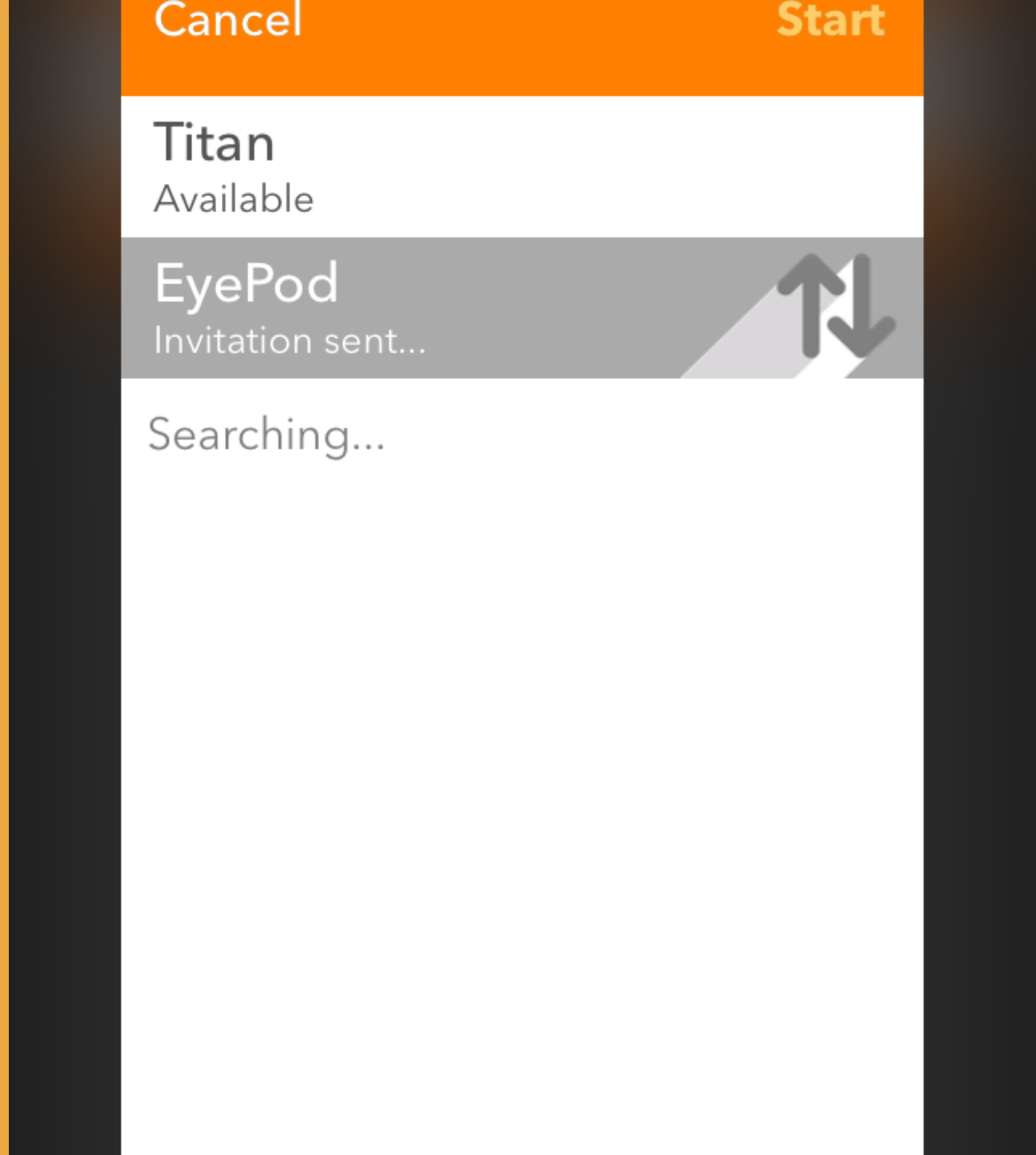
Issue: Once an invitation is accepted there's a delay before connecting



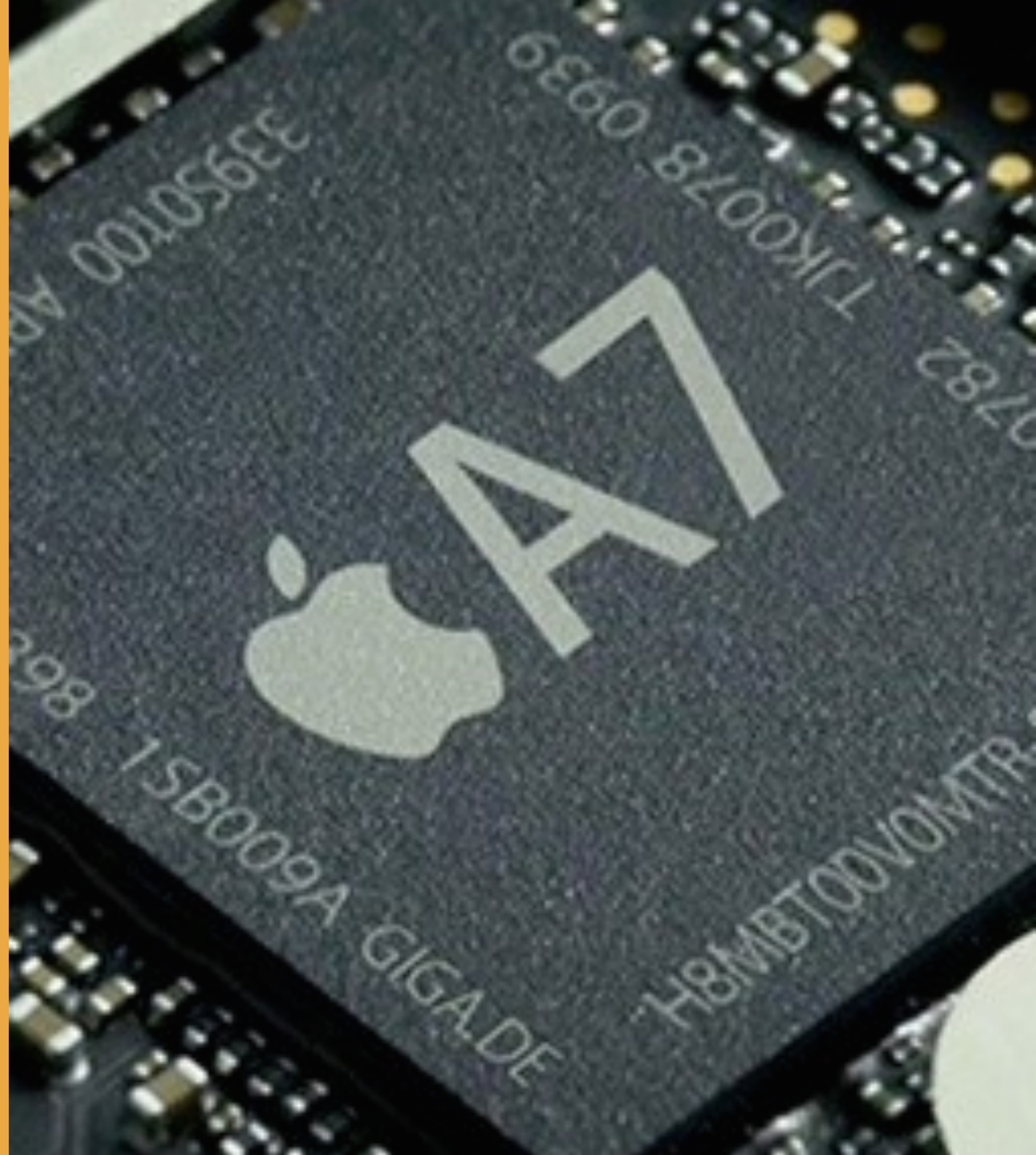
Invitation response times

Issue: Once an invitation is accepted there's a delay before connecting

Solution: Encryption is on by default which adds overhead to initial connections. Consider using `MCEncryptionNone`

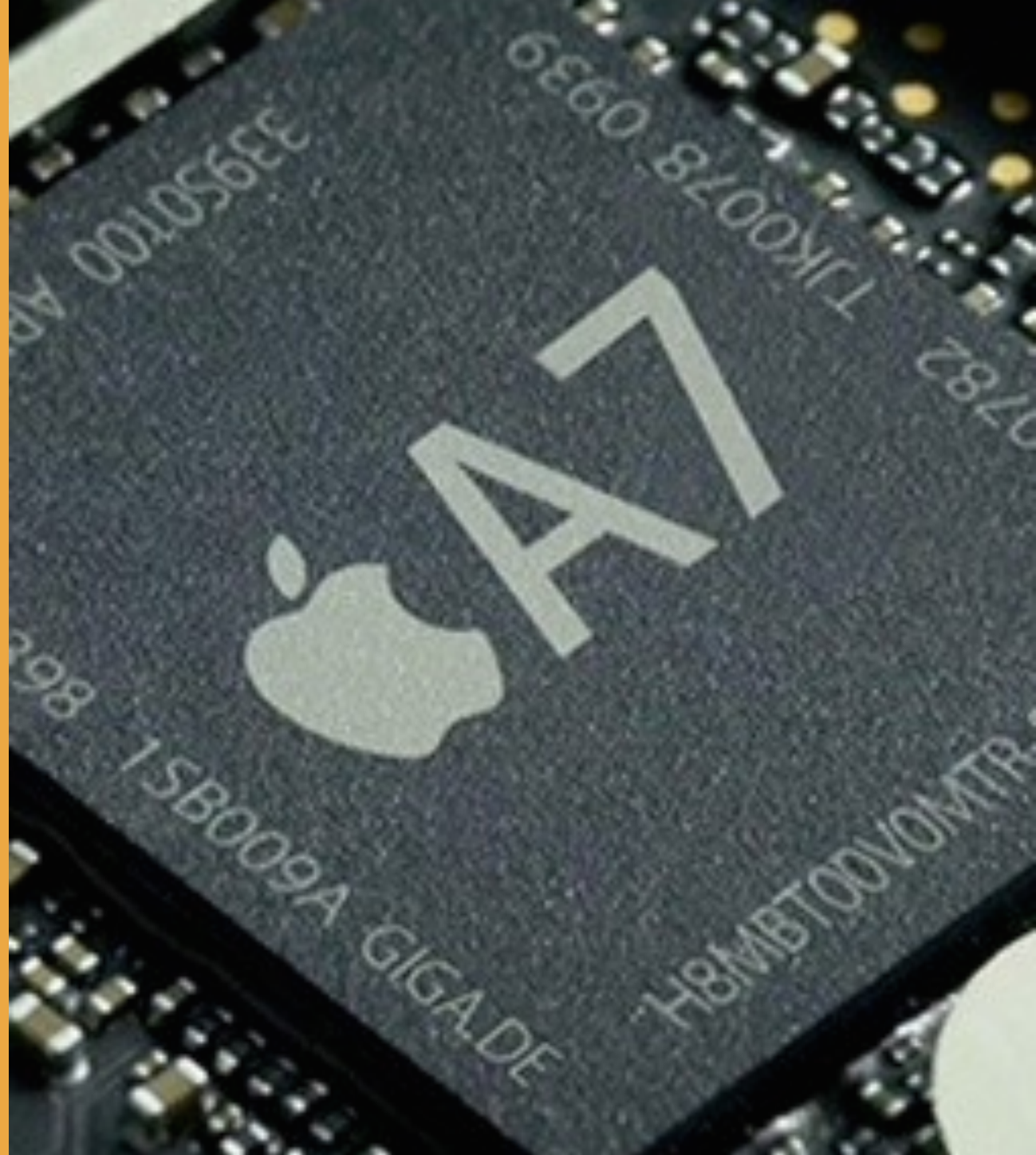


Comparing MCPeerIDs



Comparing MCPeerIDs

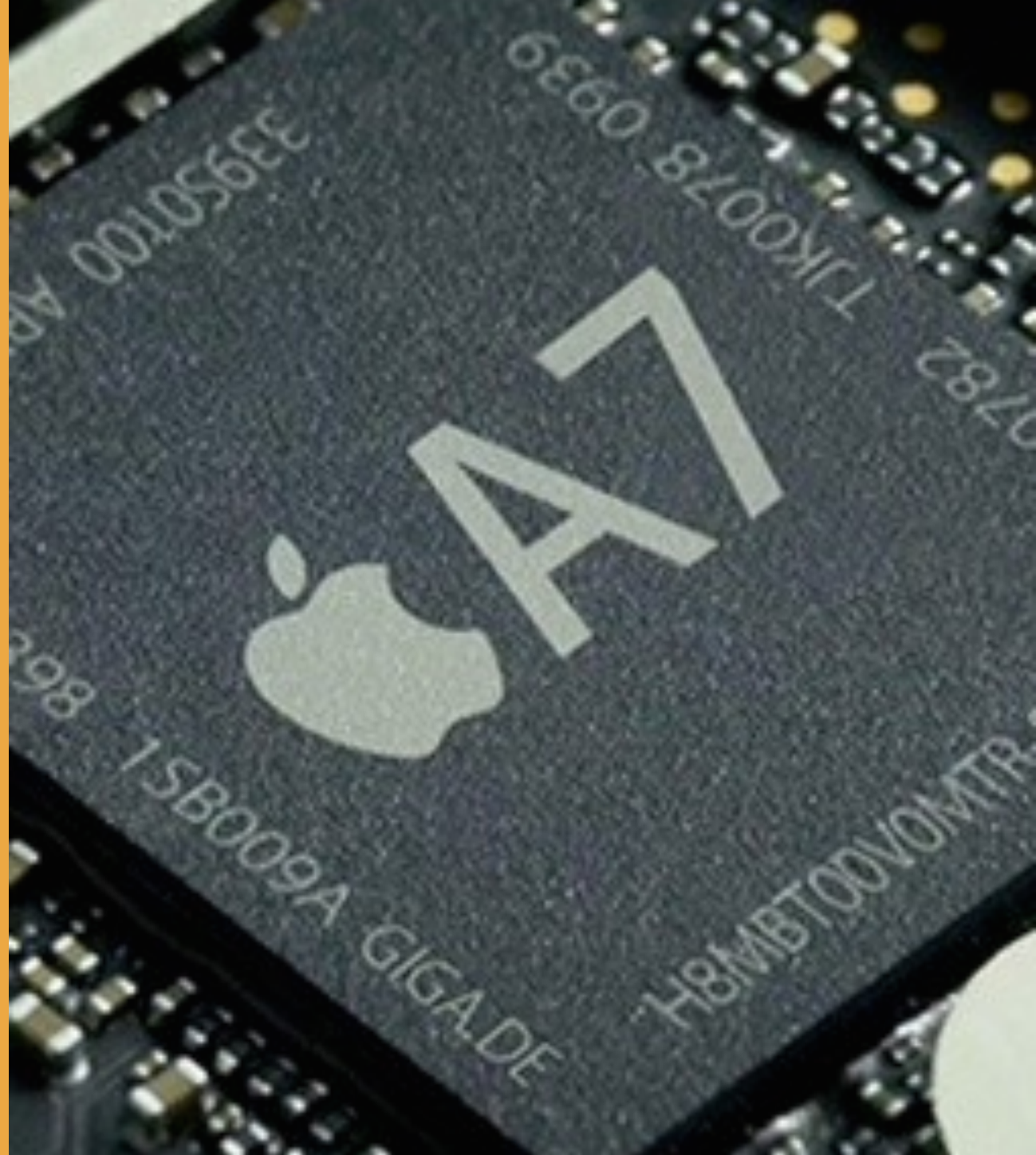
Issue: MCPeerID `hash` **method returns it's unique ID number, as an** `NSUInteger`**.**



Comparing MCPeerIDs

Issue: MCPeerID `hash` method returns it's unique ID number, as an `NSUInteger`.

Solution: Truncate the returned `hash` to 32-bit for comparison.




Demo

LocalTalk

on the app store

update in progress for iOS 8

OS X version to follow

free!


Future

Multipeer for Mac

Multipeer for Mac

Common API with iOS

Multipeer for Mac

Common API with iOS

But! Supports background operation

Multipeer for Mac

Common API with iOS

But! Supports background operation

However! Not Bluetooth

Multipeer for Mac

Common API with iOS

But! Supports background operation

However! Not Bluetooth

AirDrop rebuilt to use the new API

Multipeer for Mac

Common API with iOS

But! Supports background operation

However! Not Bluetooth

AirDrop rebuilt to use the new API

OS X <-> iOS AirDrop. *Finally*.

End

created with Deckset

http://www.decksetapp.com

icons made with Sketch

http://bohemiancoding.com/sketch/

Questions?

Rob Stearn

@cocoadelica

www.cocoadelica.co.uk

robstearn@me.com

WWDC 2013 Session 708
WWDC 2014 Session 709
IANA service type list
Bonjour Overview
Multipeer Networking Reference
NSNetServices & CFNetServices Programming Guide
MultipeerGroupChat sample code
Forums