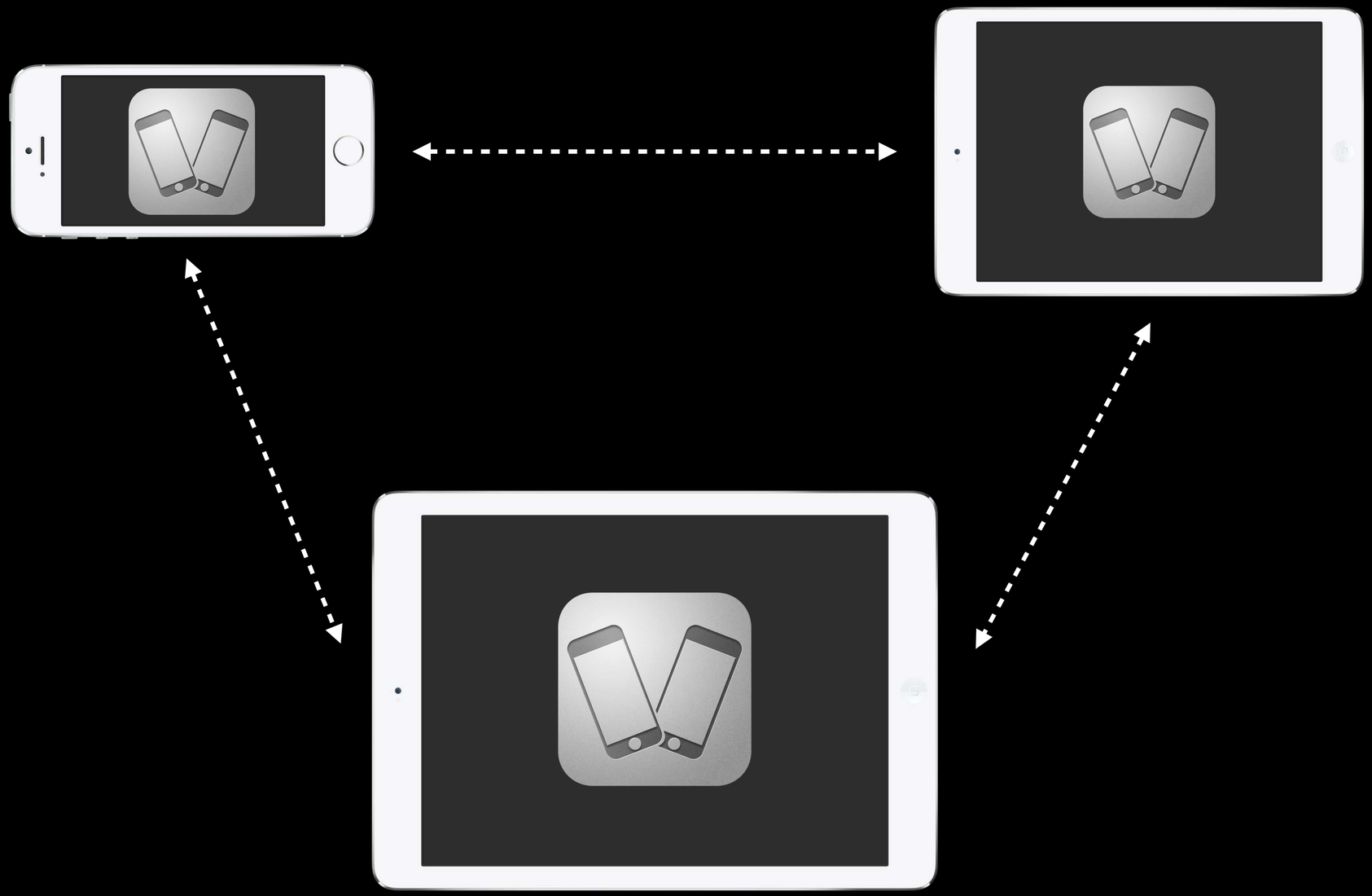
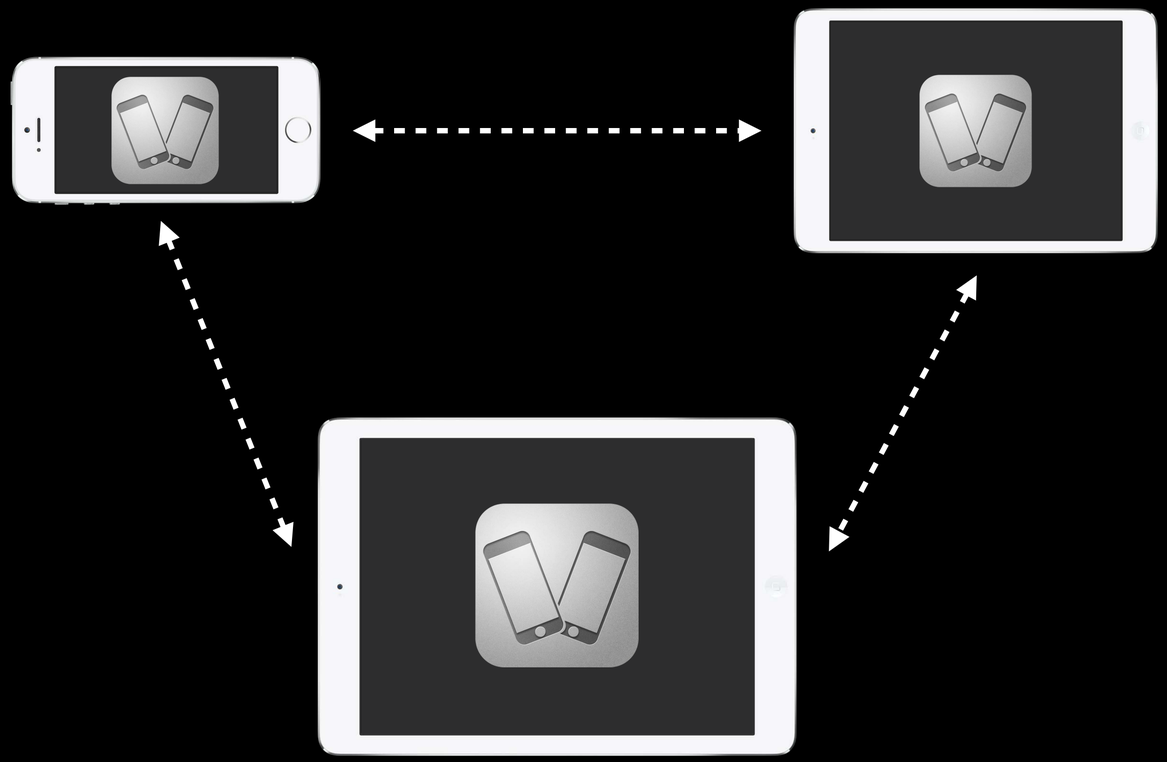
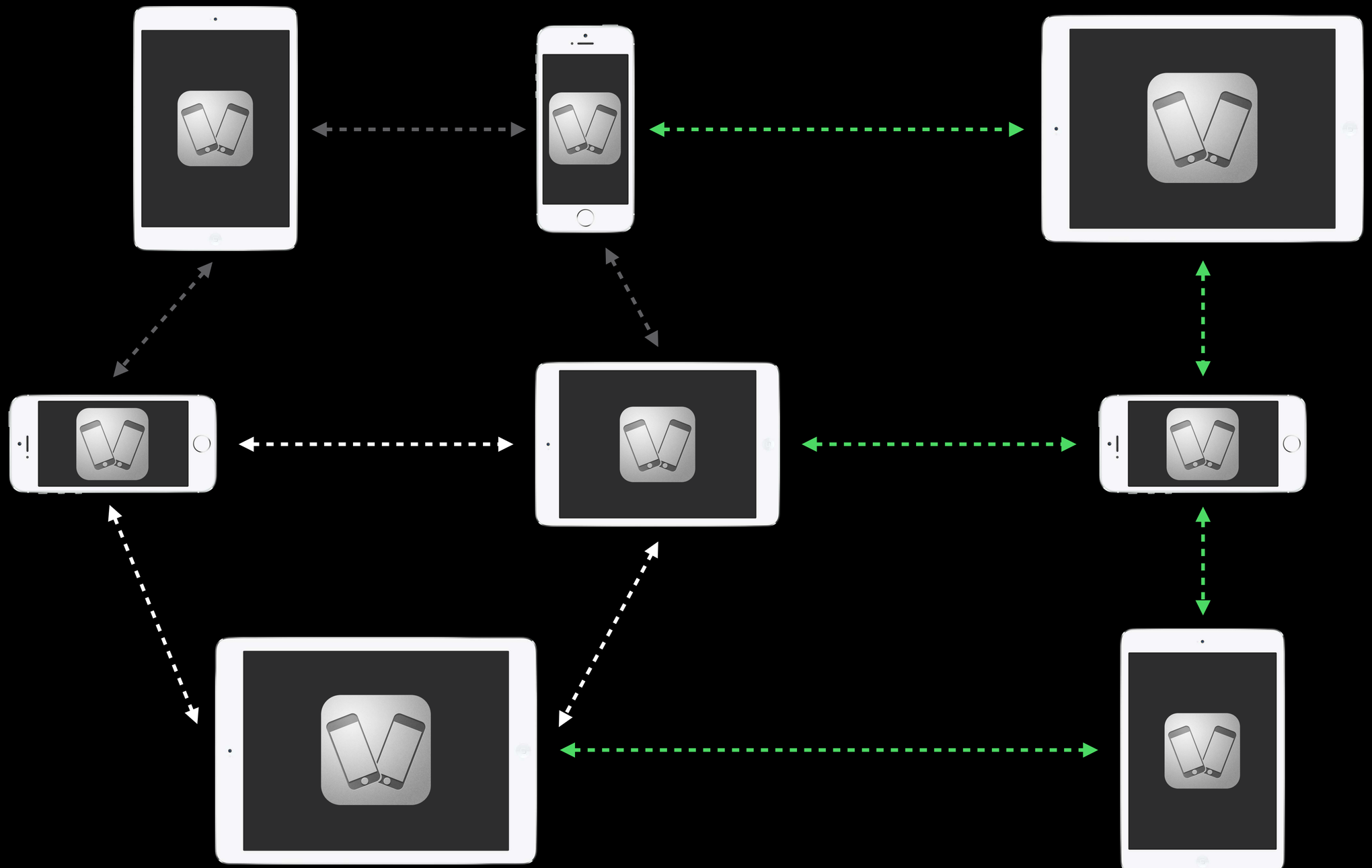
Cross Platform Nearby Networking

Session 709
Demijan Klinc
Software Engineer








HD

fx



TD




PRO

You
Compose







DREAM LEAGUE
SOCCER







FTS 14













HD

TOUCH

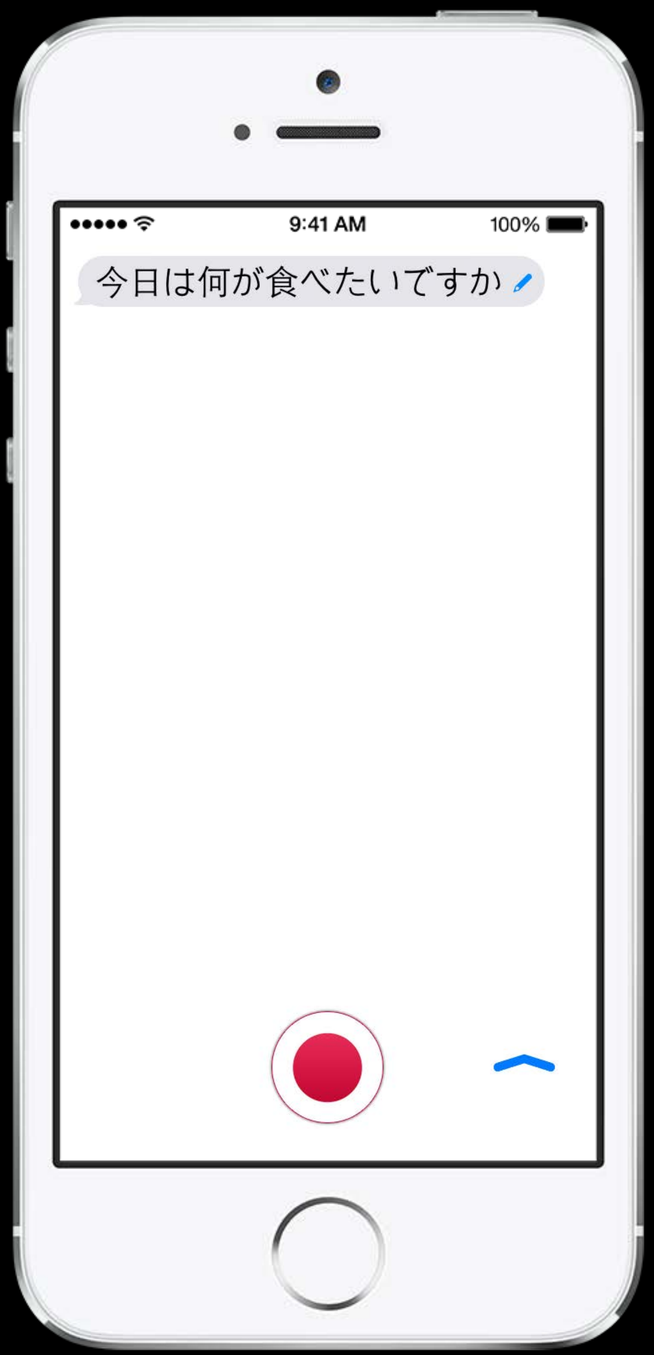
CARDNET

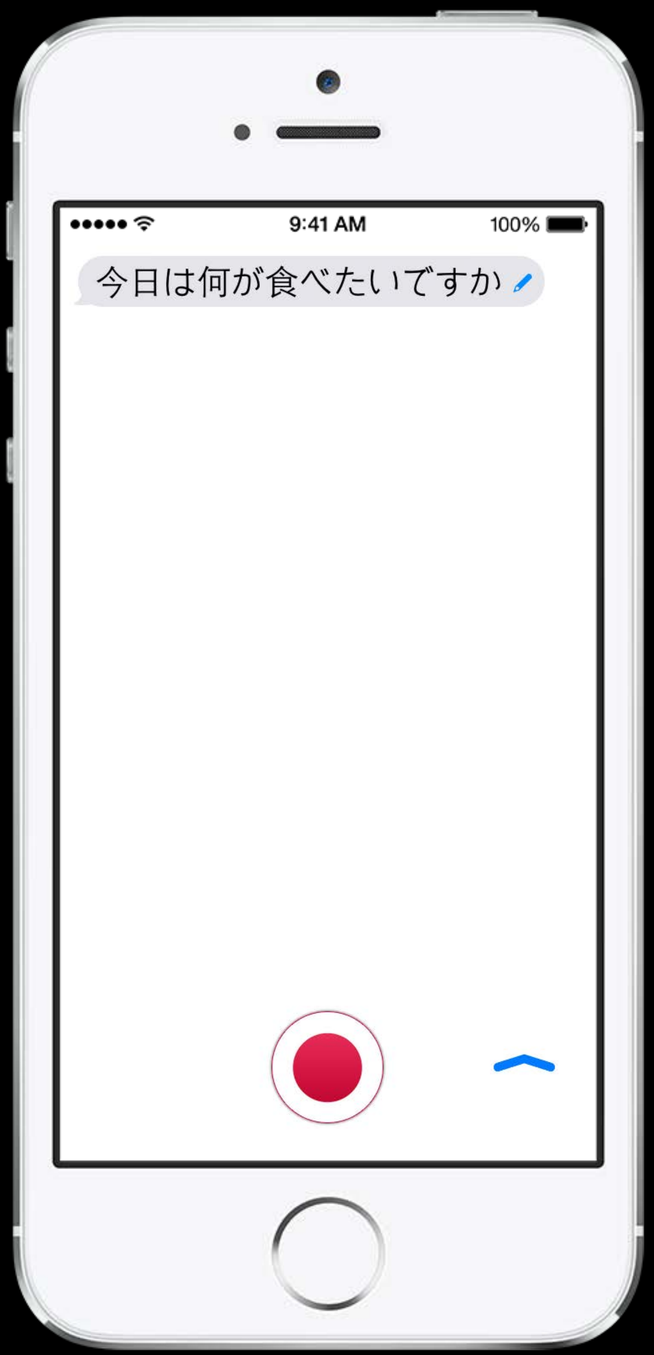
Score!

pop

iTranslate Voice


9:41 AM
100%
今日は何が食べたいですか

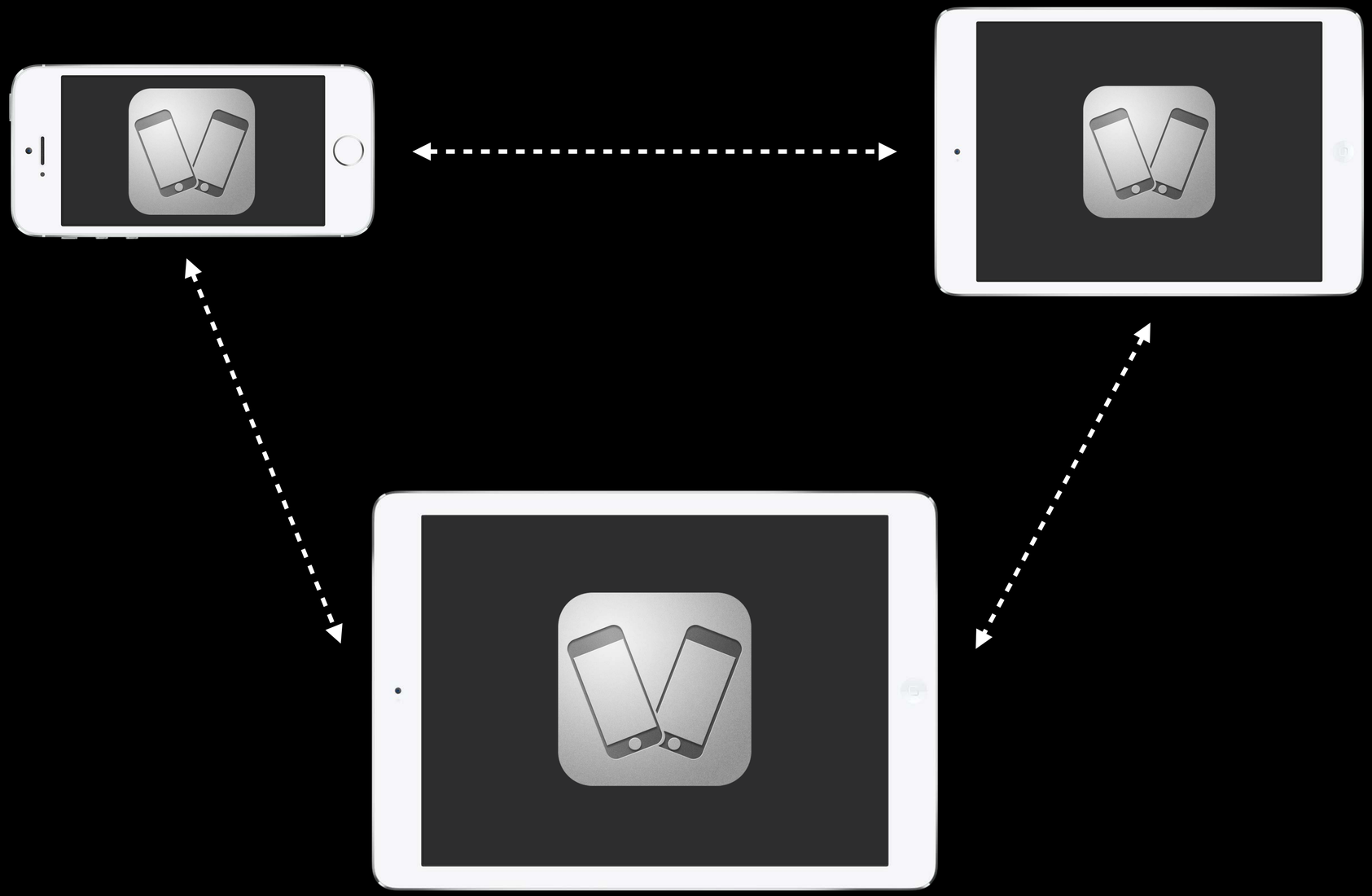

9:41 AM
100%
今日は何が食べたいですか

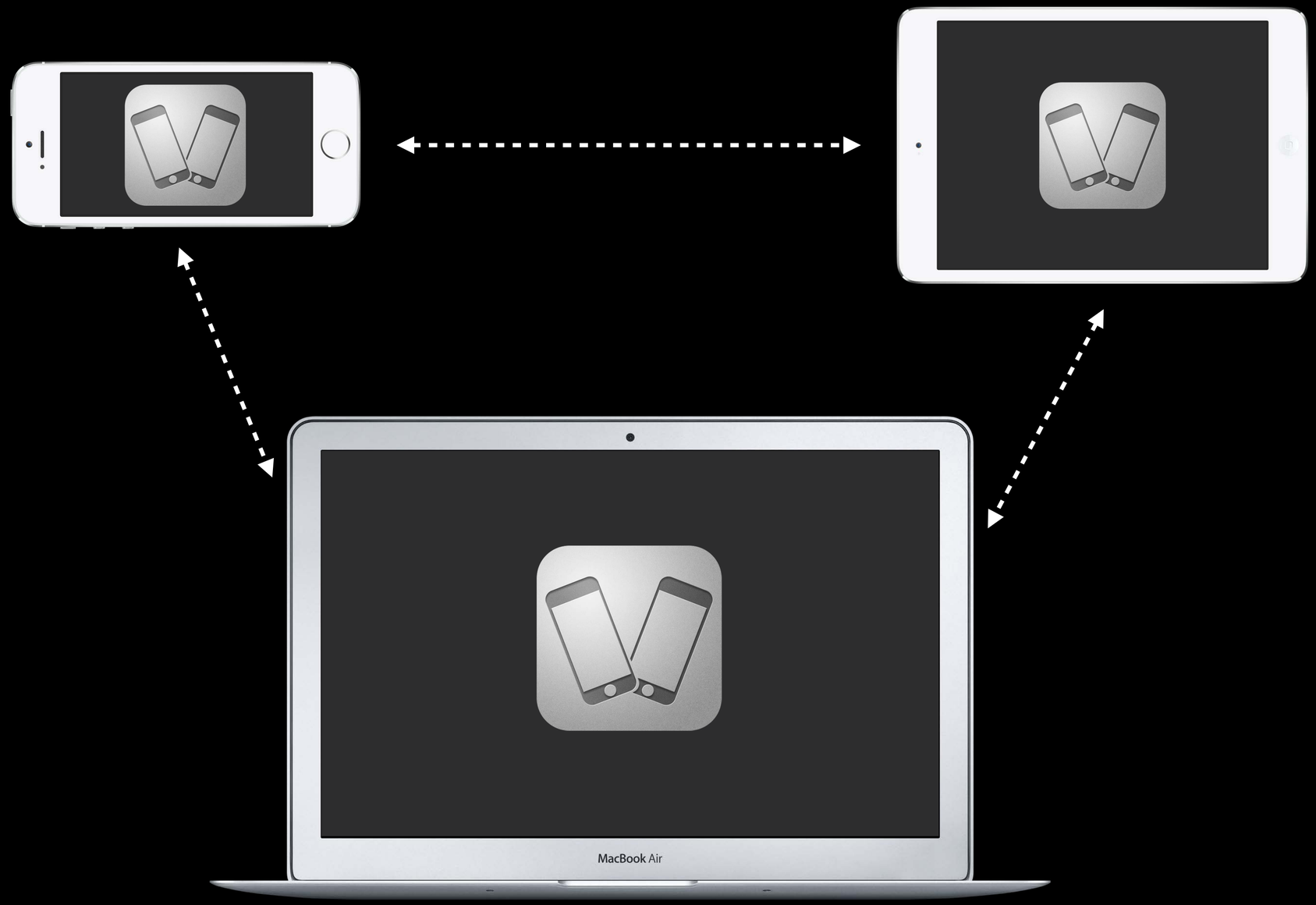

9:41 AM
100%
今日は何が食べたいですか

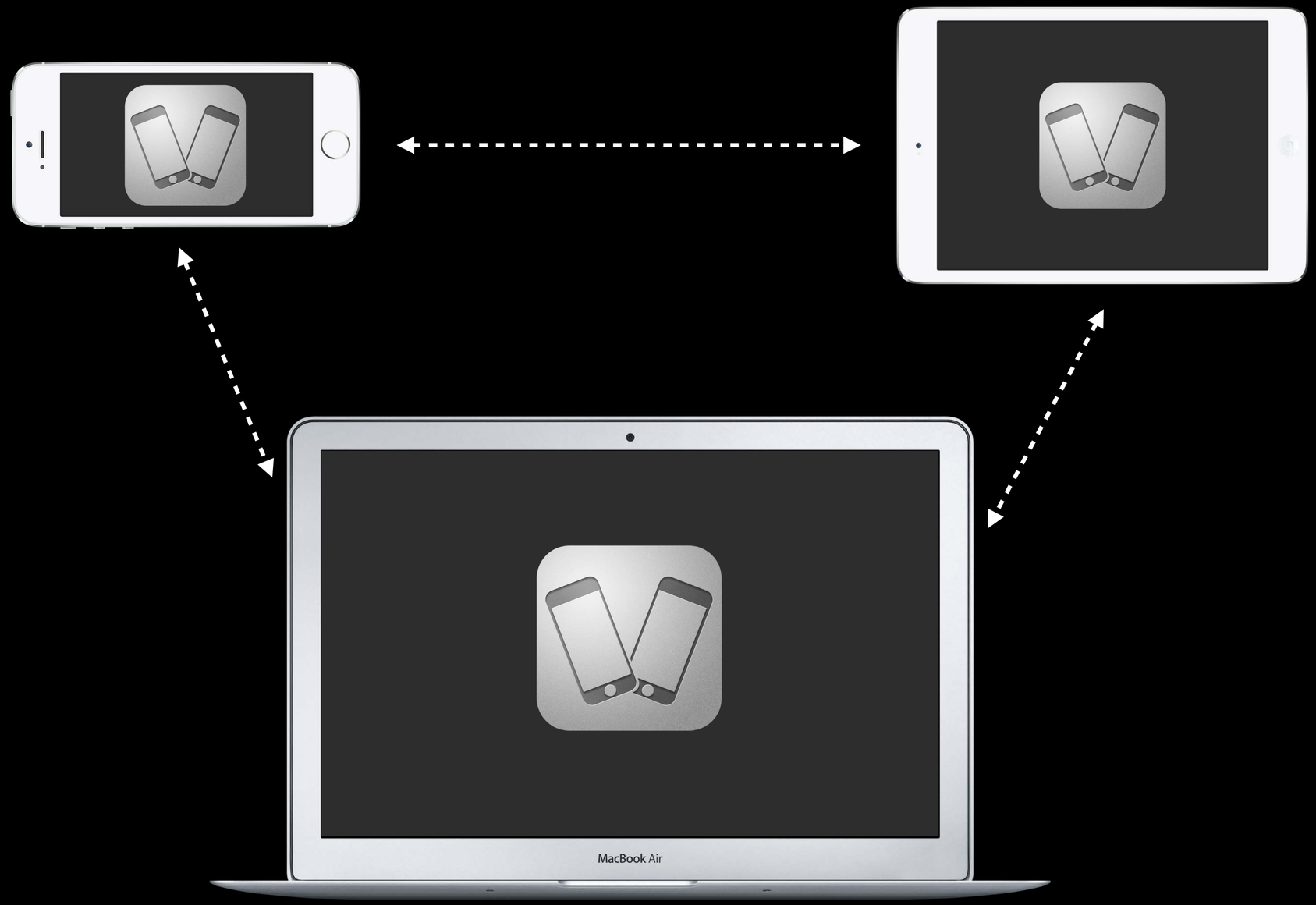

9:41 AM
100%
What do you want to eat today

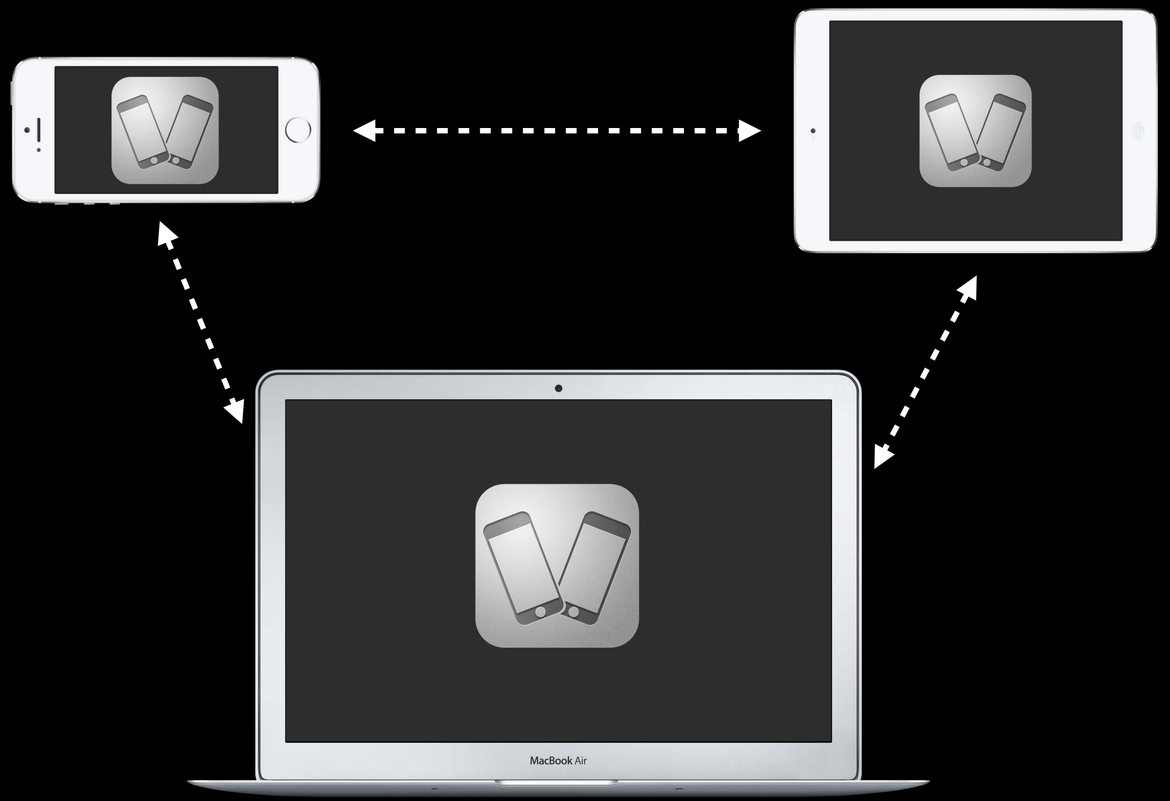
Metronome Touch

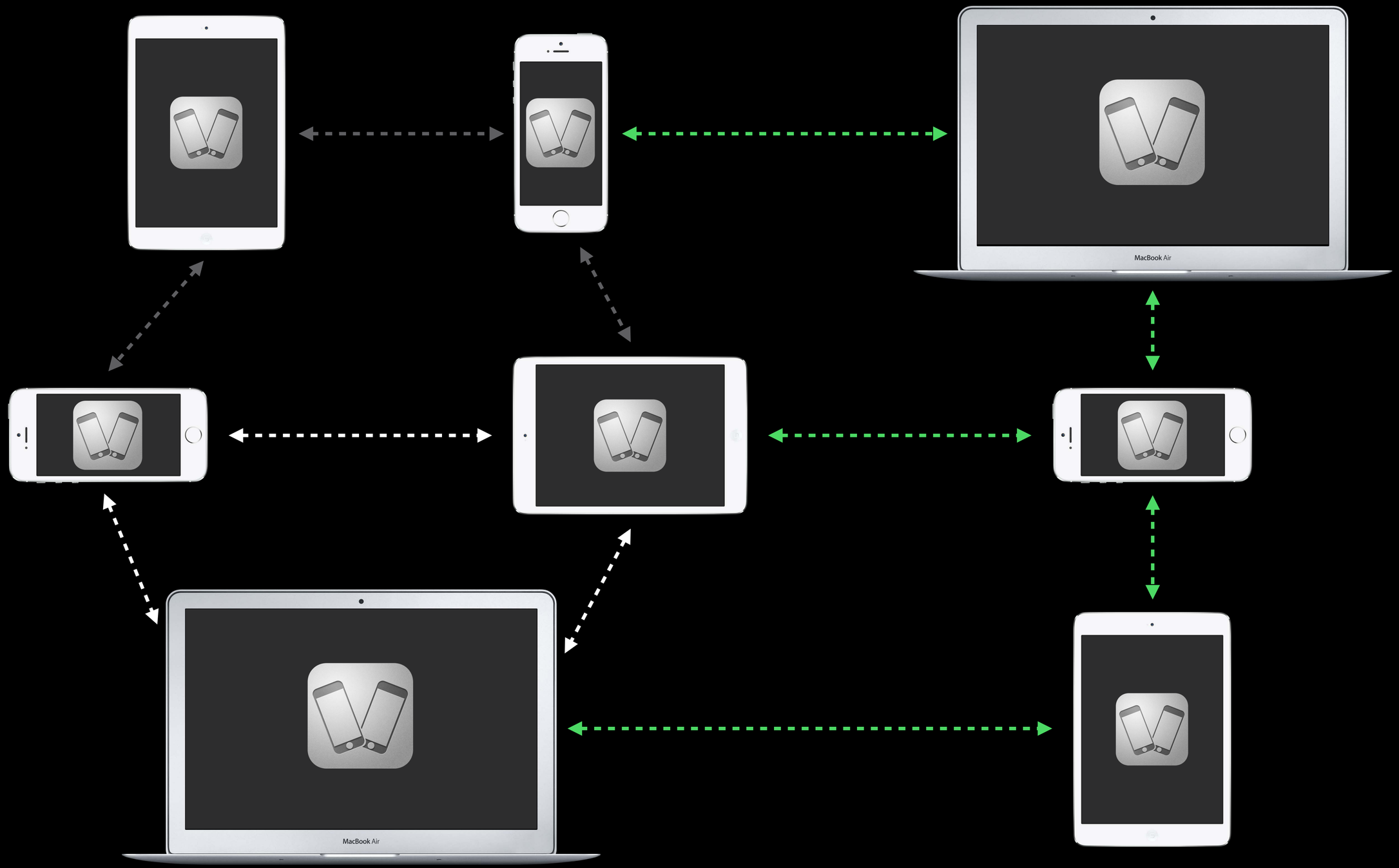
Firechat




MacBook Air


MacBook Air


MacBook Air


MacBook Air
MacBook Air

Agenda

Basics

Multipeer Connectivity on OS X

Best Practices

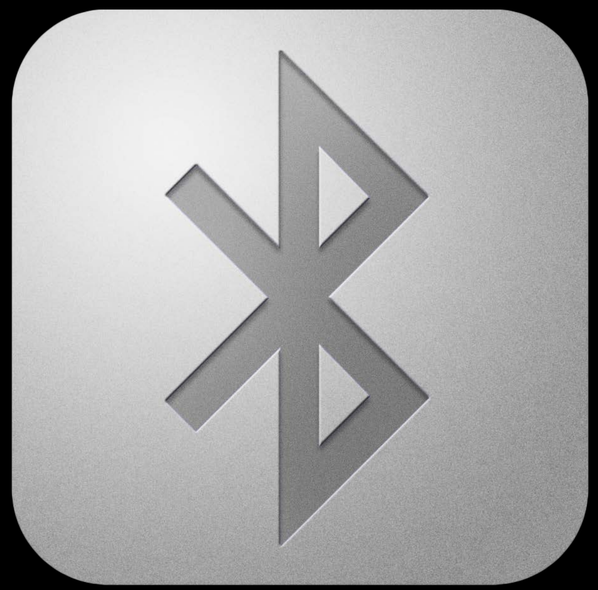
Custom Discovery

Authentication

Basics

Wireless Technologies

iOS



Wireless Technologies

OSX

Devices with Lightning connector

Devices with Lightning connector

Starts with 2012 Macs

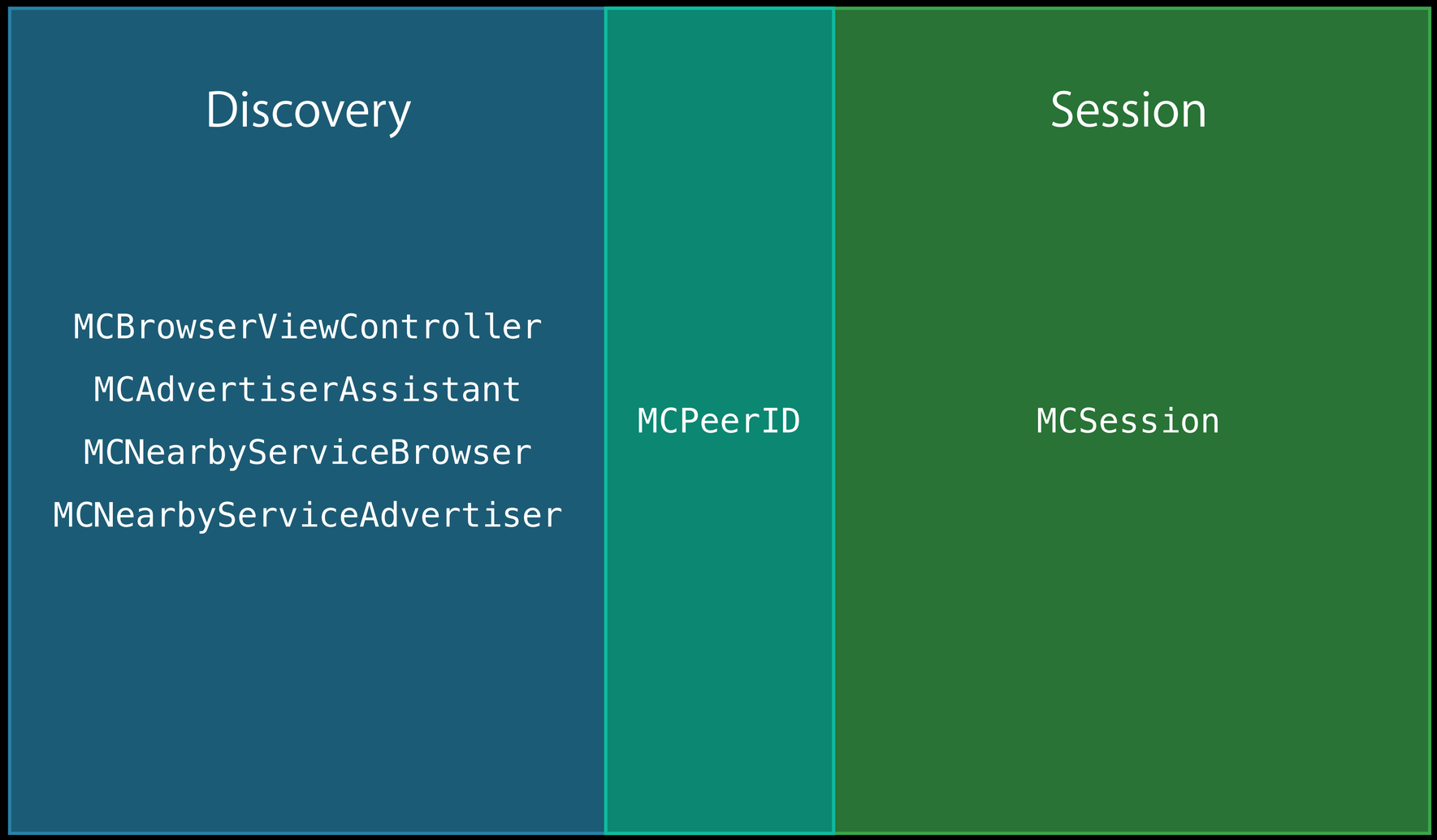
Terminology

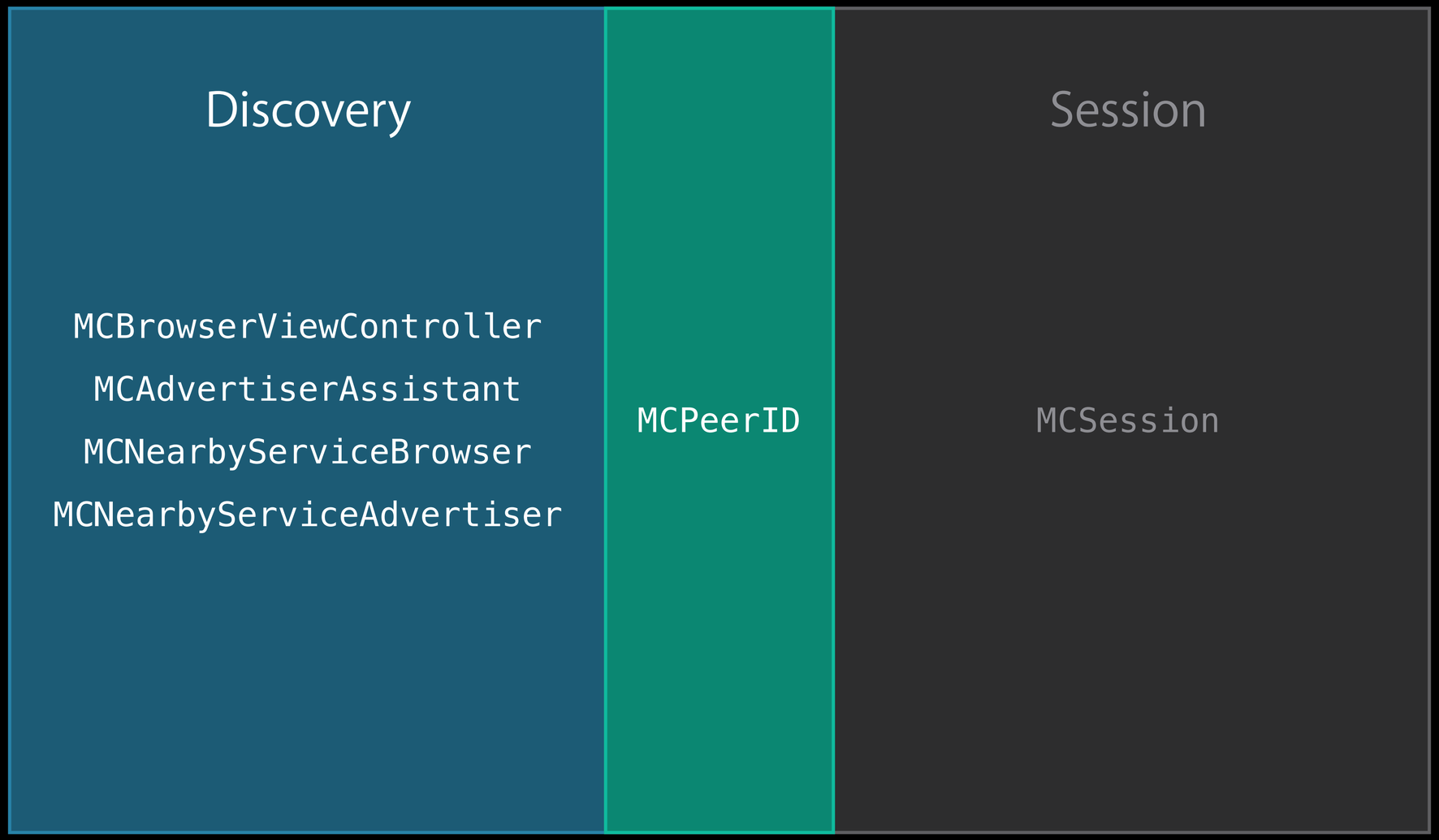
Nearby
Within range of supported wireless technologies

Peer
Nearby device

Advertiser
Device discoverable by other nearby devices

Browser
Device searching for other nearby devices

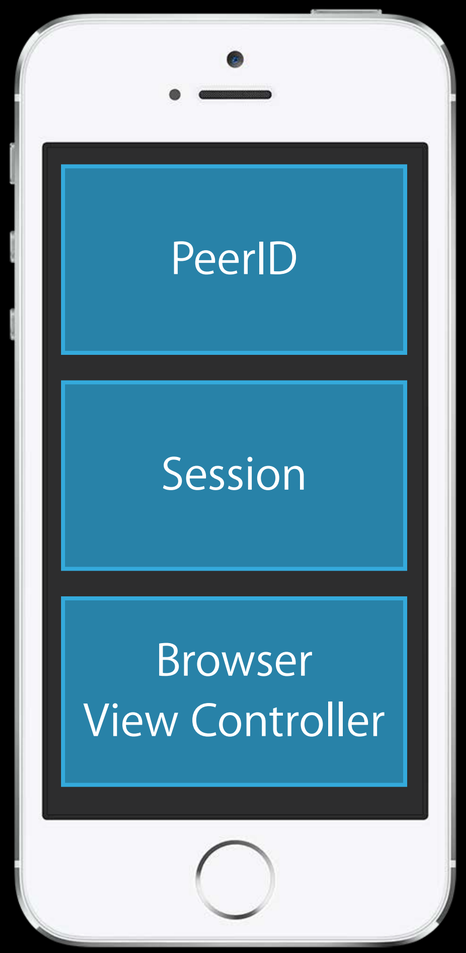

Discovery
MCBrowserViewController
MCAdvertiserAssistant
MCNearbyServiceBrowser
MCNearbyServiceAdvertiser
MCPeerID
Session
MCSession


Discovery
MCBrowserViewController
MCAdvertiserAssistant
MCNearbyServiceBrowser
MCNearbyServiceAdvertiser
MCPeerID
Session
MCSession

Discovery

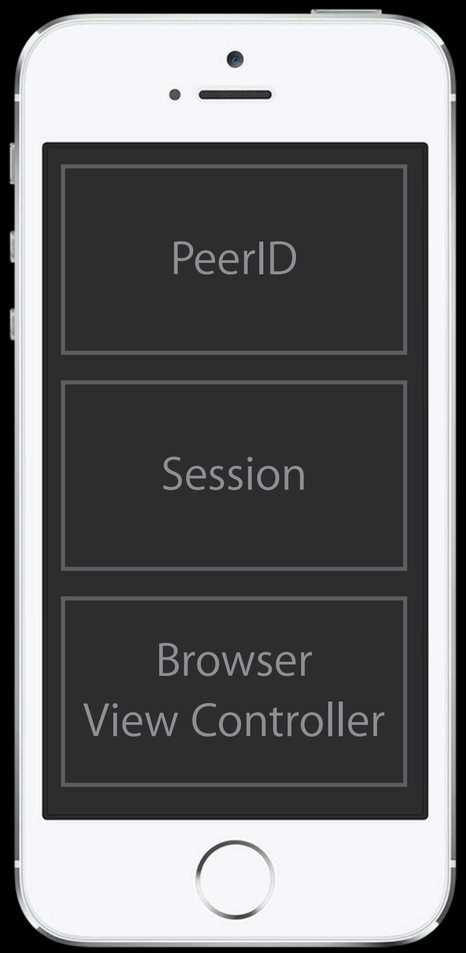
UI-based

```
MCPeerID
MCBrowserViewController
MCAdvertiserAssistant
```



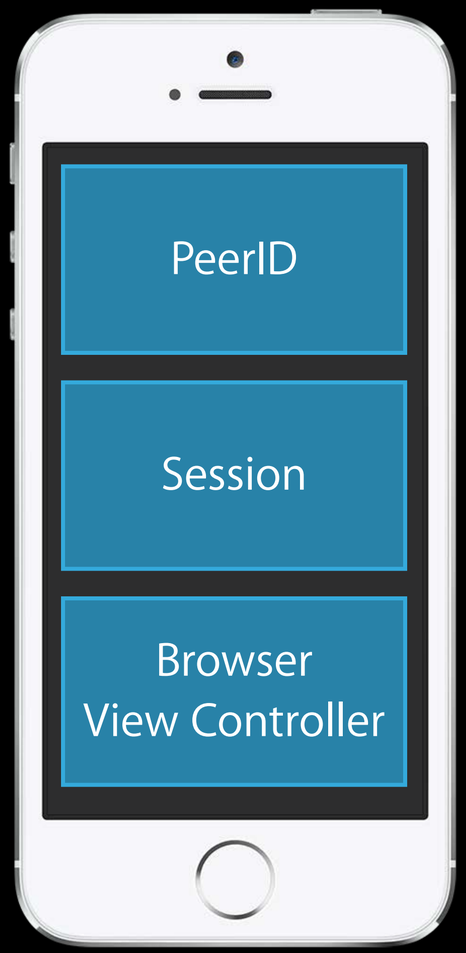
Browser

Advertiser

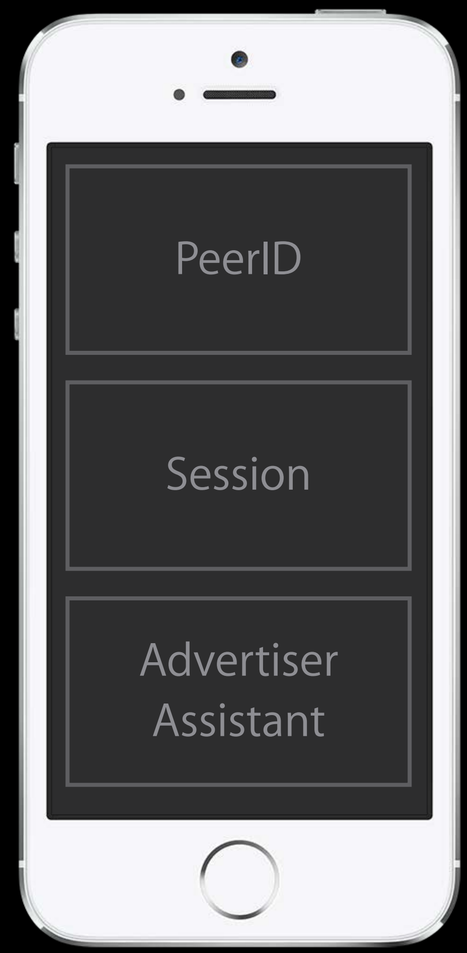


Browser

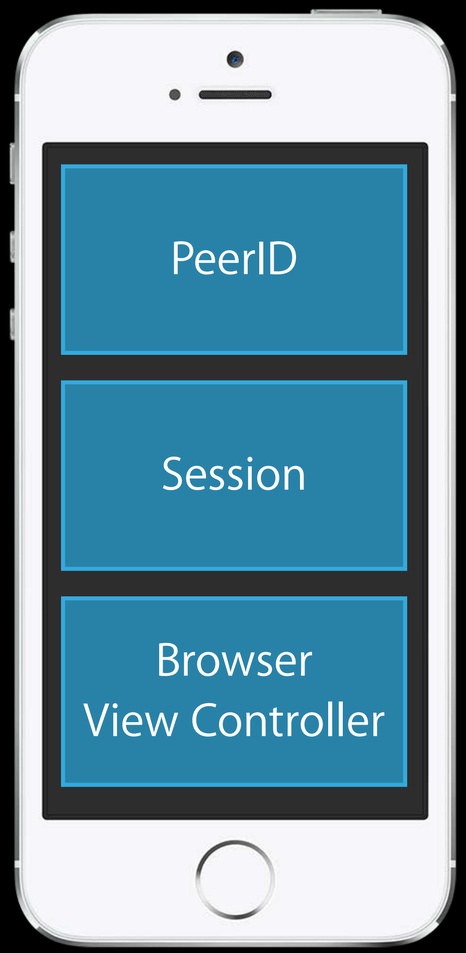
Advertiser



Browser



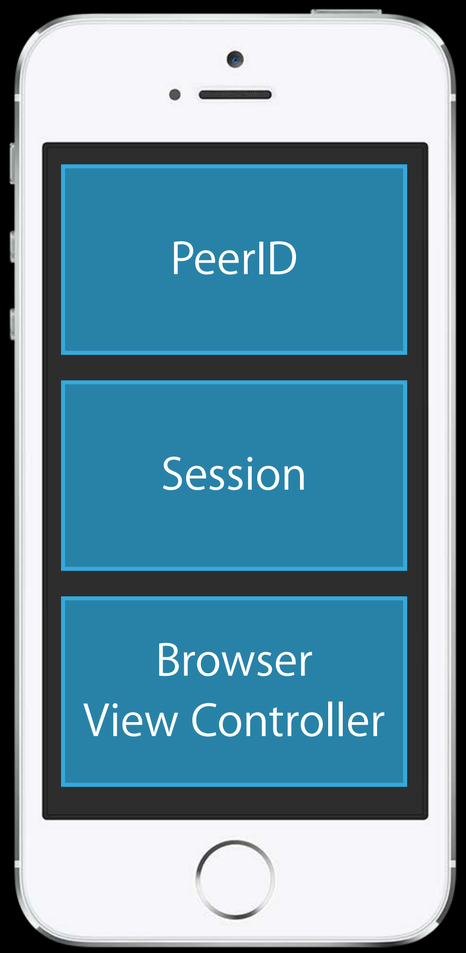
Advertiser



Browser

Advertiser

…peer:didChangeState:…



Browser

…peer:didChangeState:…

Advertiser

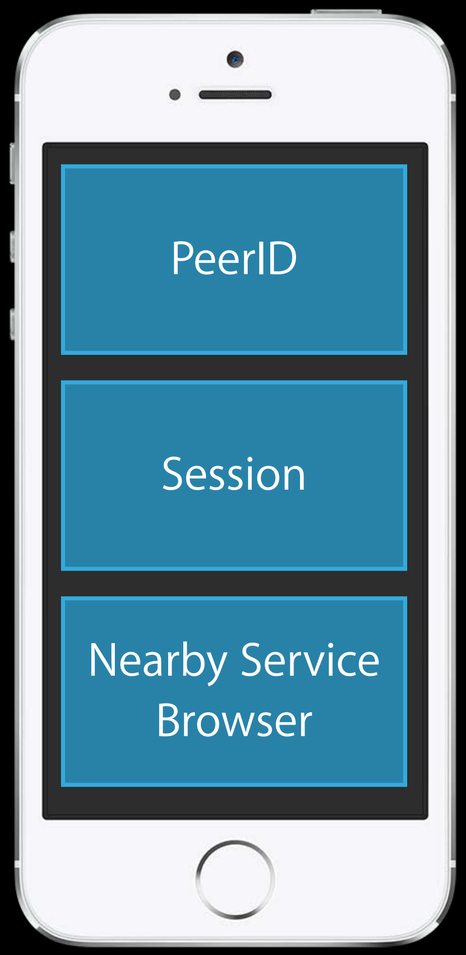
Discovery

UI-based

```
MCPeerID
MCBrowserViewController
MCAdvertiserAssistant
```

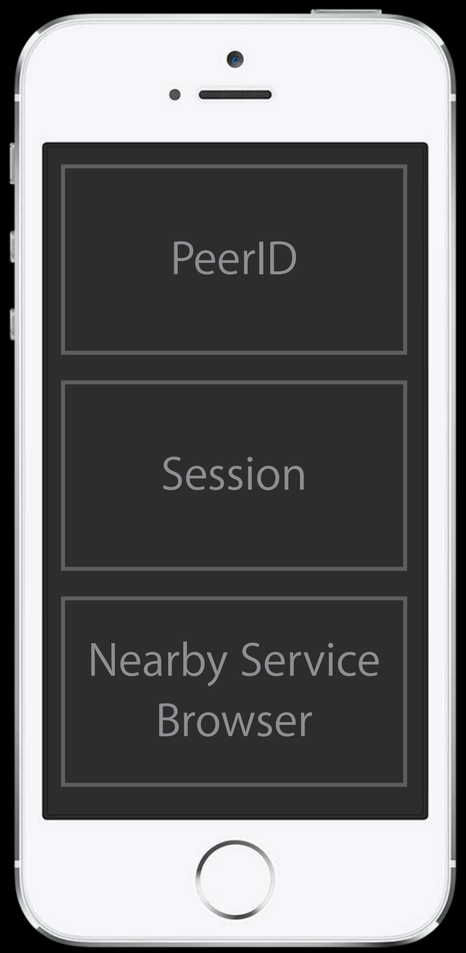
Programmatic

```
MCPeerID
MCNearbyServiceBrowser
MCNearbyServiceAdvertiser
```



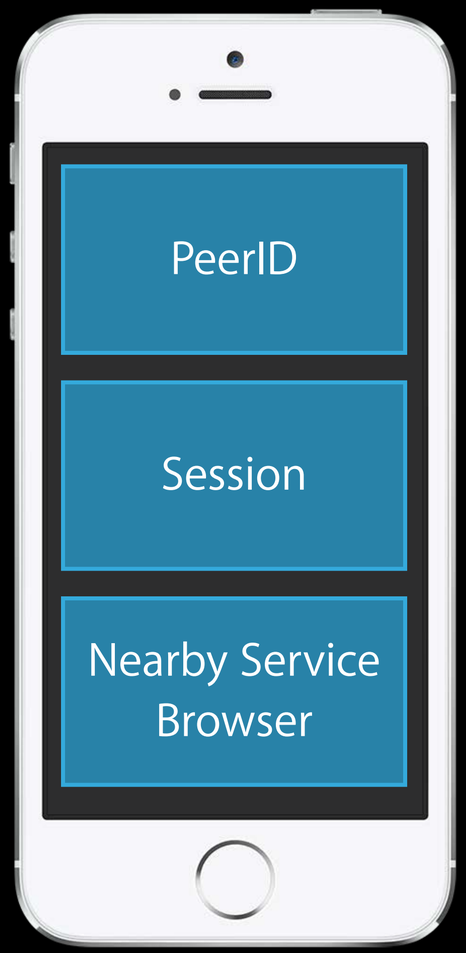
Browser

Advertiser

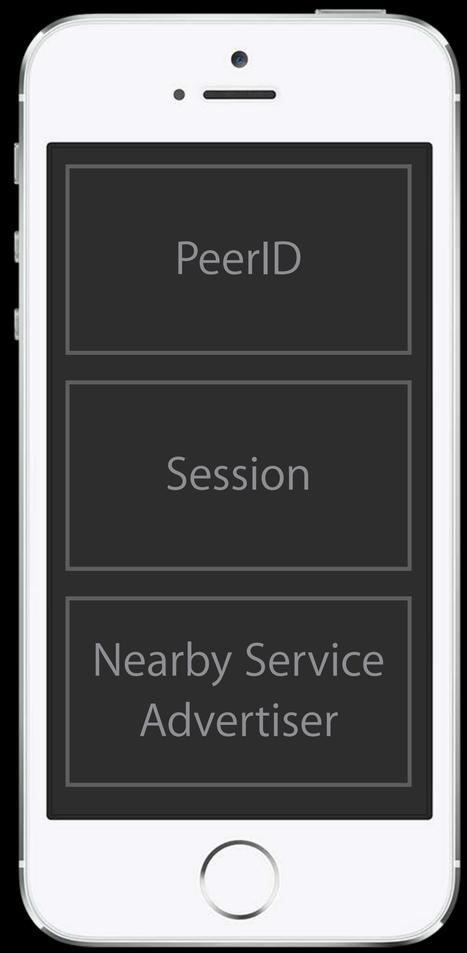


Browser

Advertiser

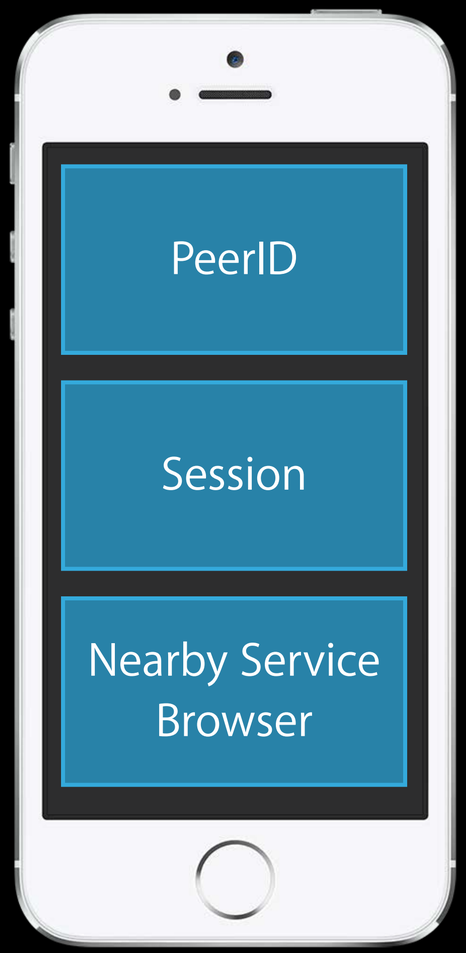


Browser



Advertiser

`startBrowsingForPeers`

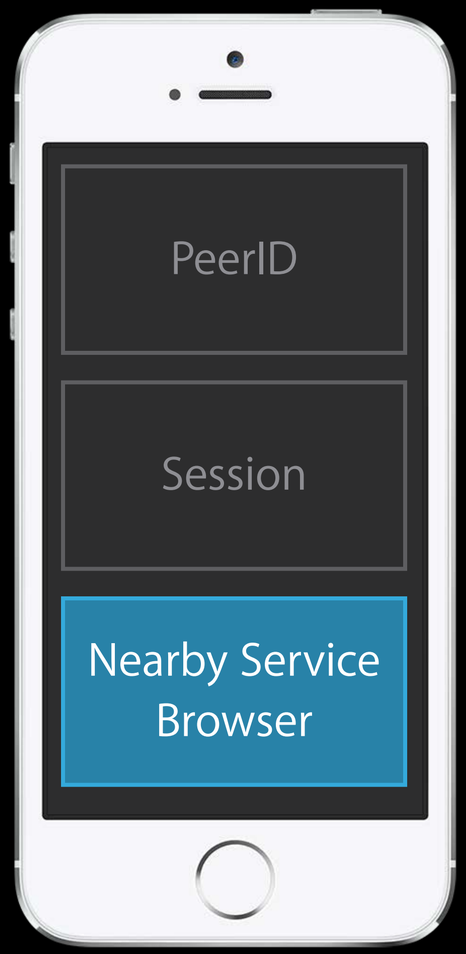


Browser

`startAdvertisingPeer`

Advertiser

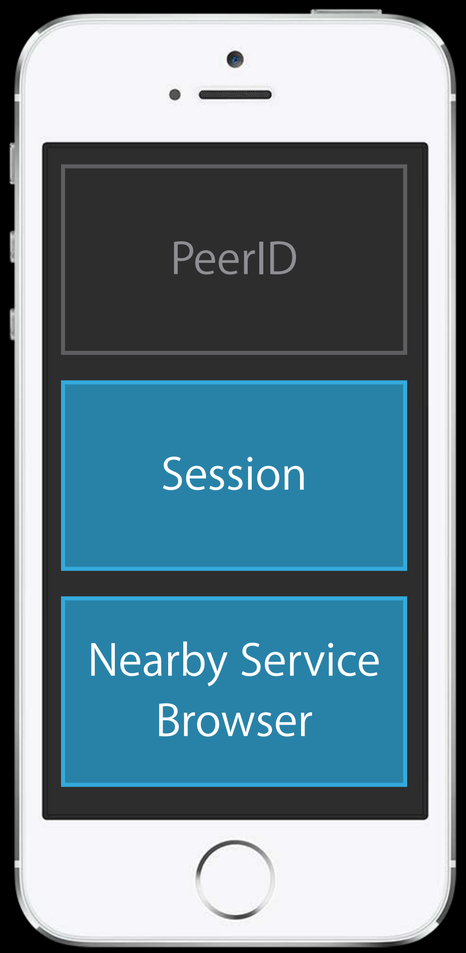
…foundPeer:…



Browser

Advertiser

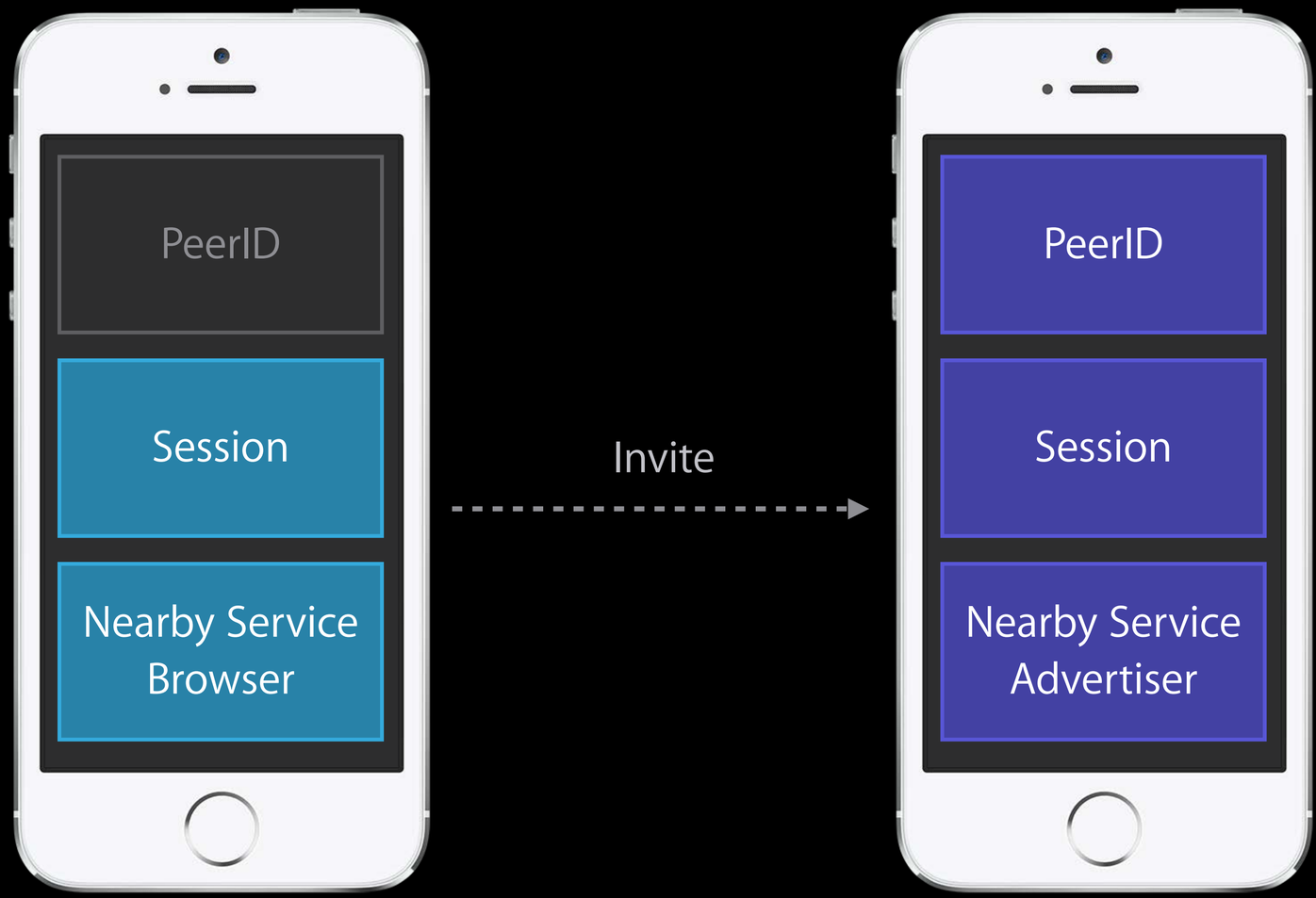
…invitePeer:…



Browser

Advertiser

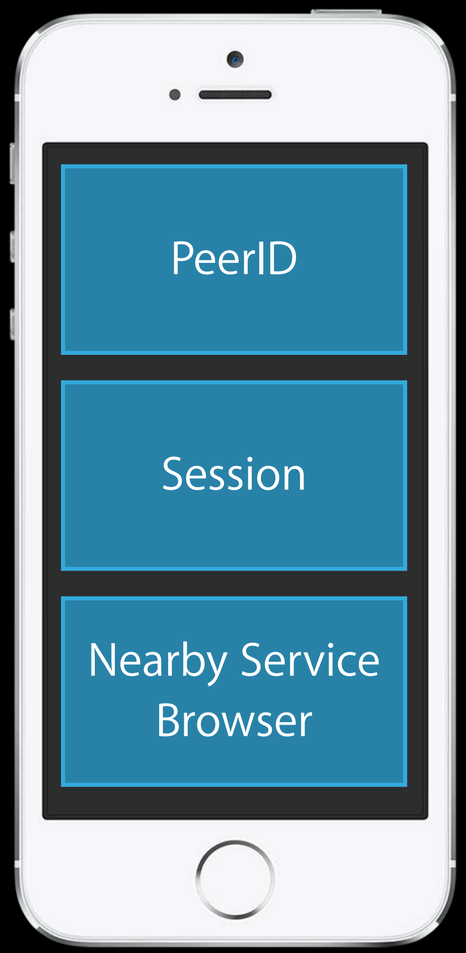
…invitePeer:…



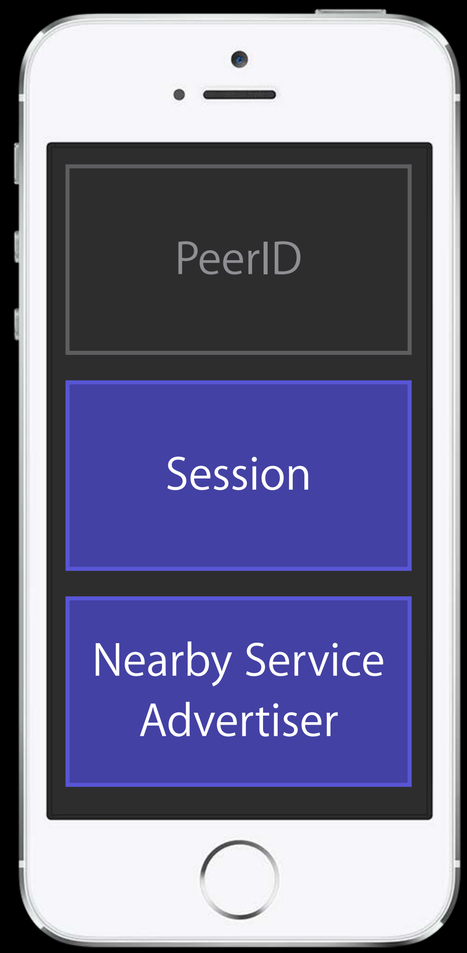
Browser

Advertiser

…didReceiveInvitationFromPeer:…

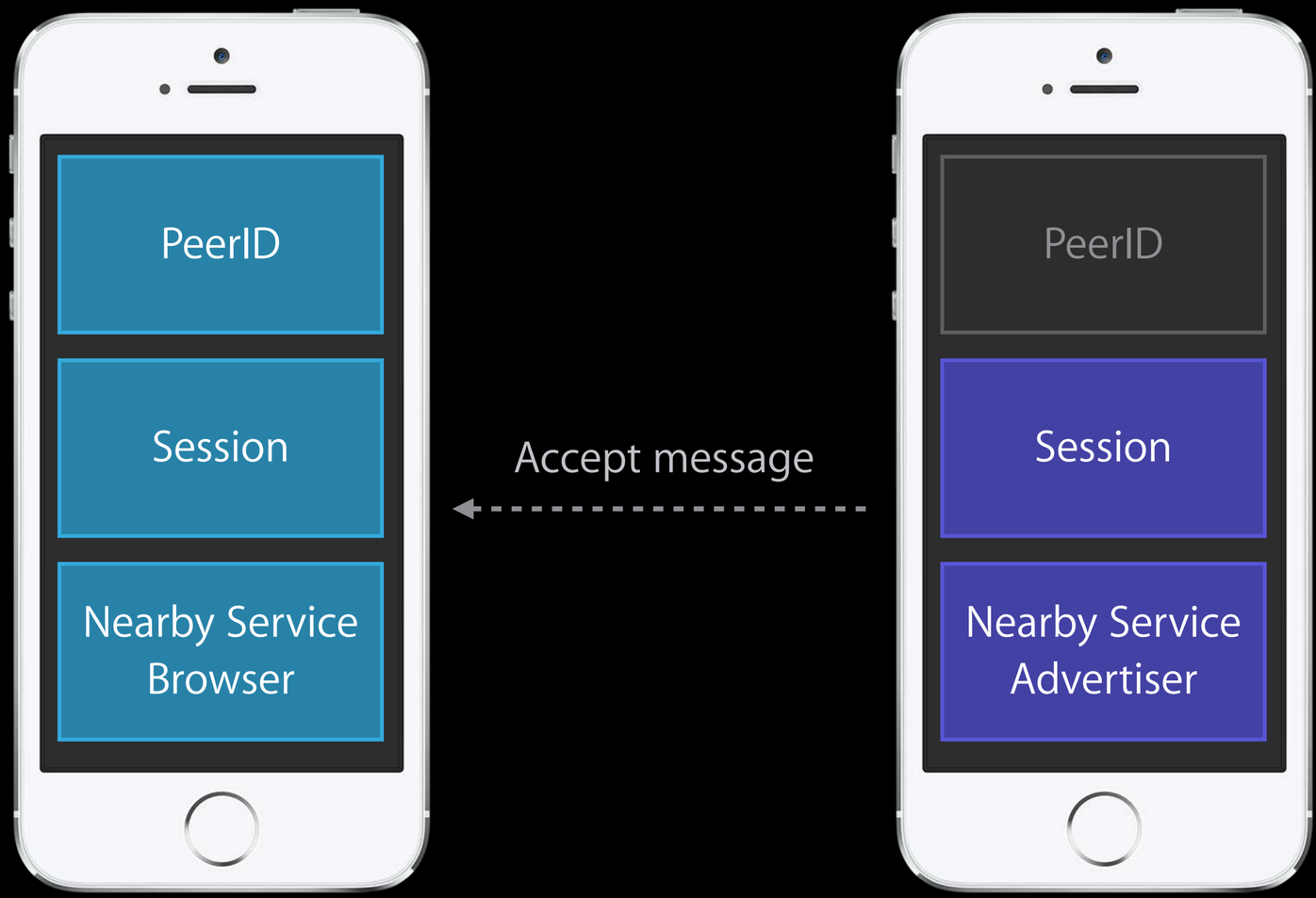


Browser



Advertiser

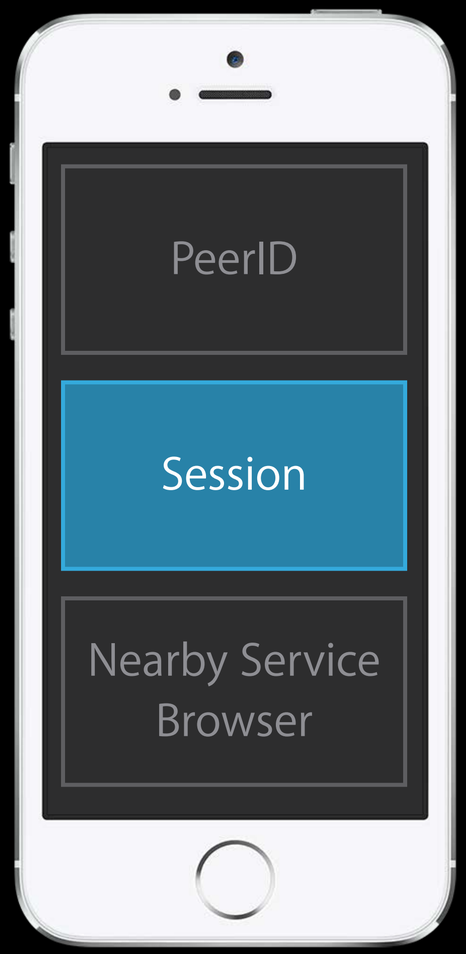
…didReceiveInvitationFromPeer:…



Browser

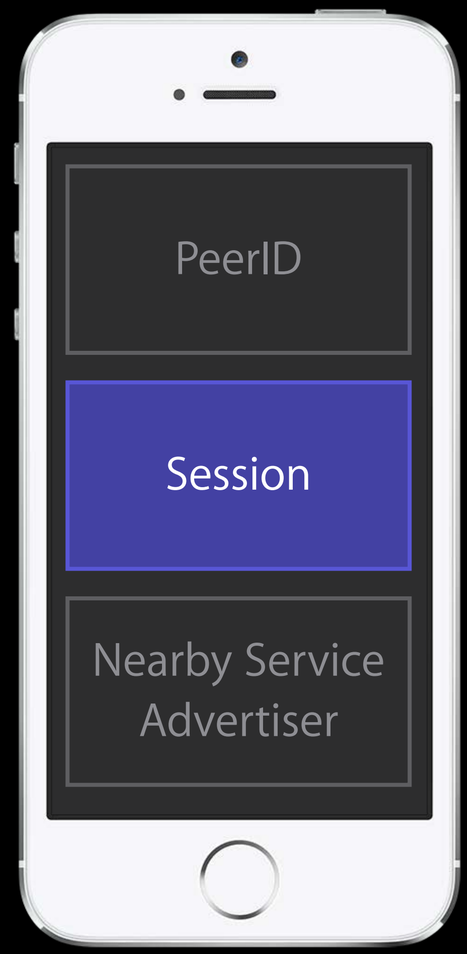
Advertiser

…peer:didChangeState:…

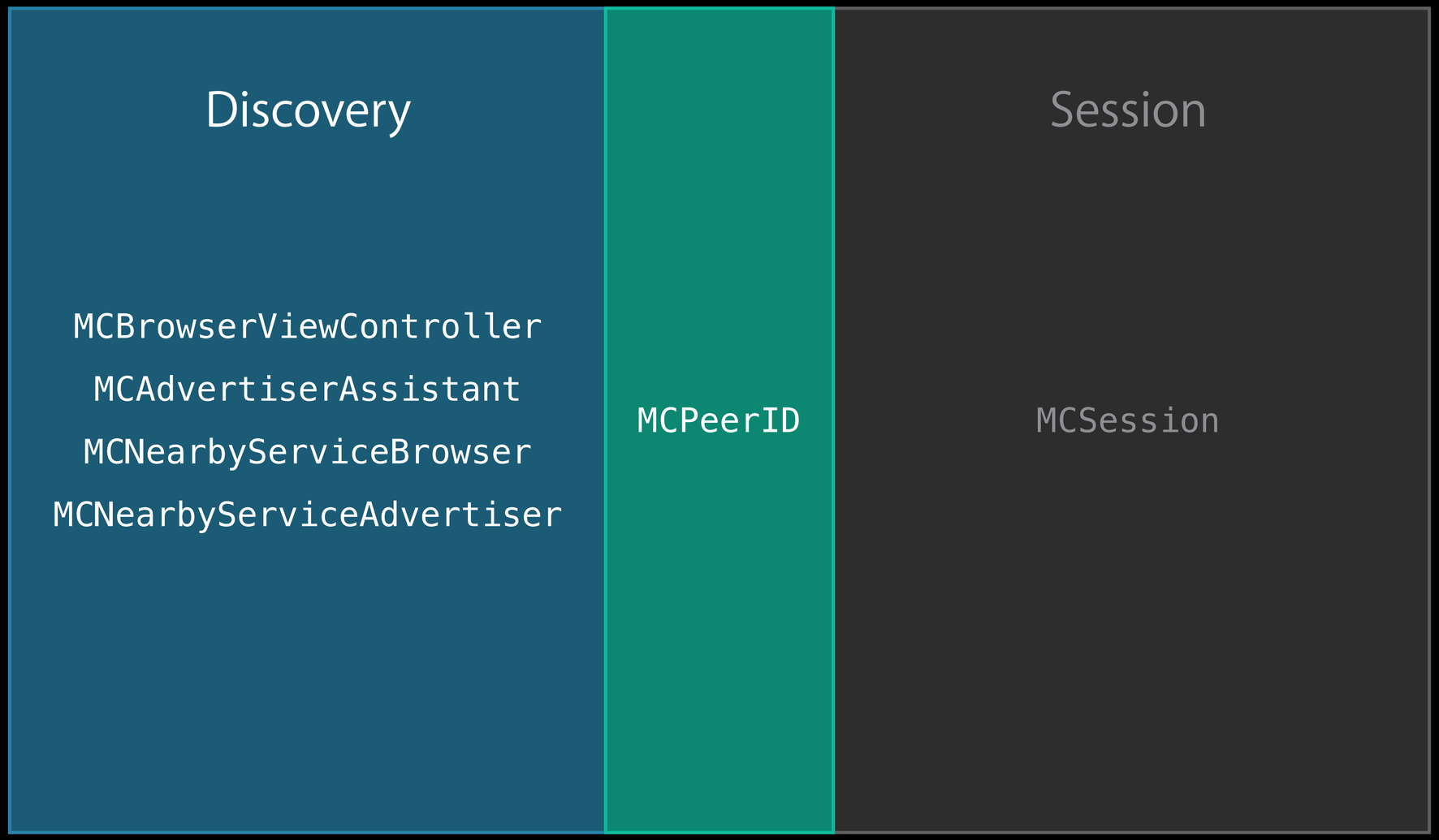


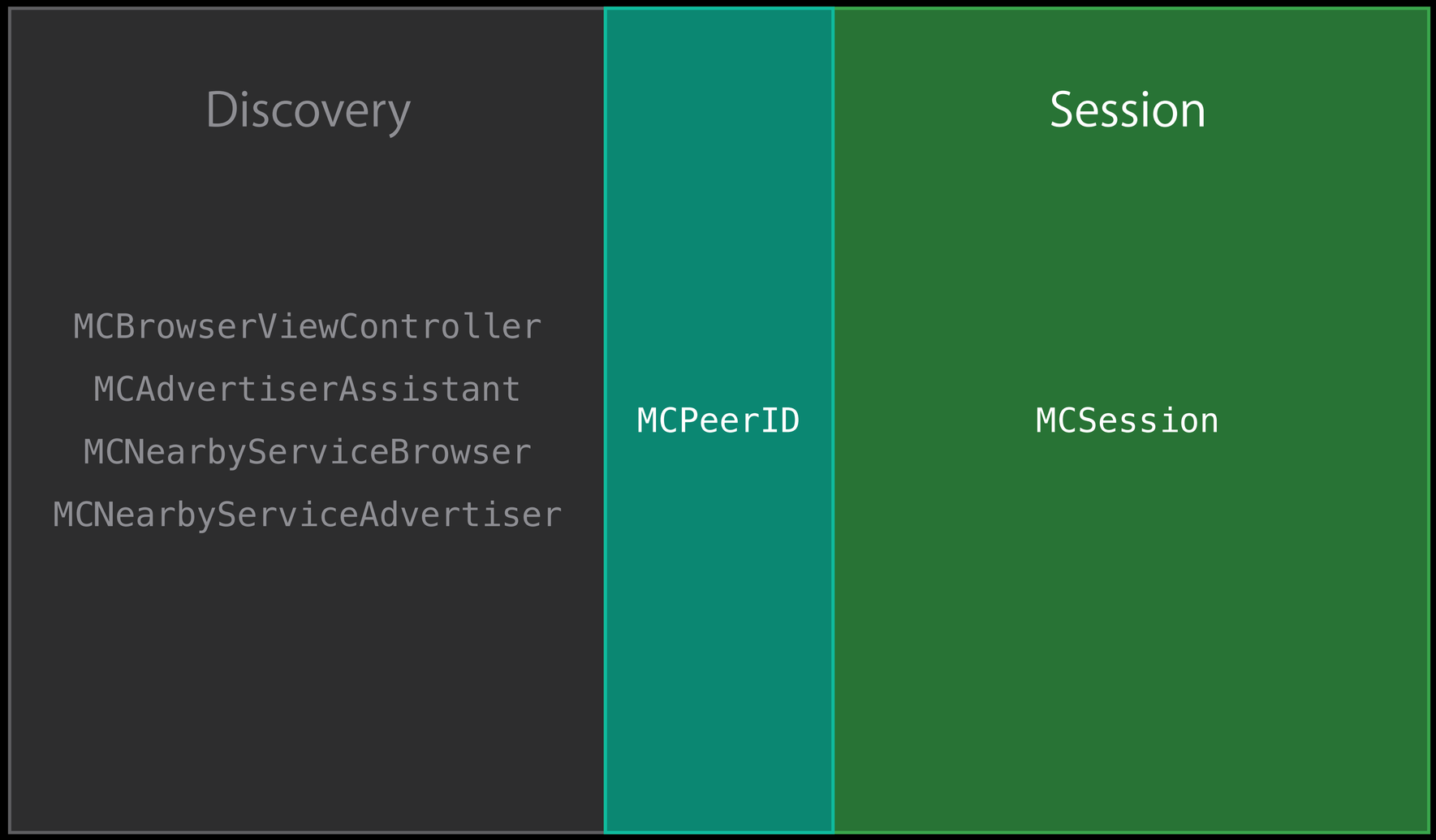
Browser

…peer:didChangeState:…



Advertiser


Discovery
MCBrowserViewController
MCAdvertiserAssistant
MCNearbyServiceBrowser
MCNearbyServiceAdvertiser
MCPeerID
Session
MCSession

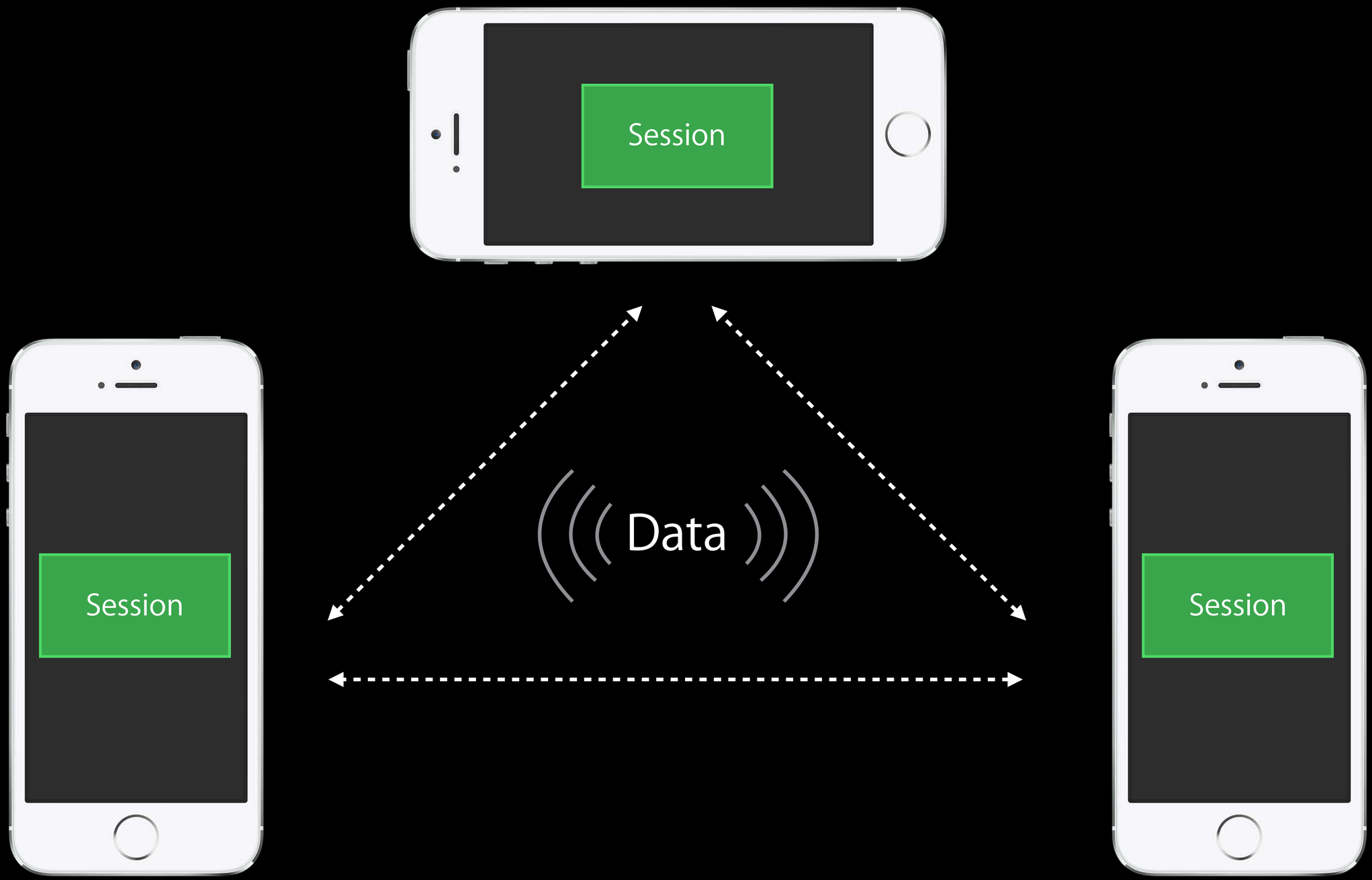

Discovery
MCBrowserViewController
MCAdvertiserAssistant
MCNearbyServiceBrowser
MCNearbyServiceAdvertiser
MCPeerID
Session
MCSession

Session

Session

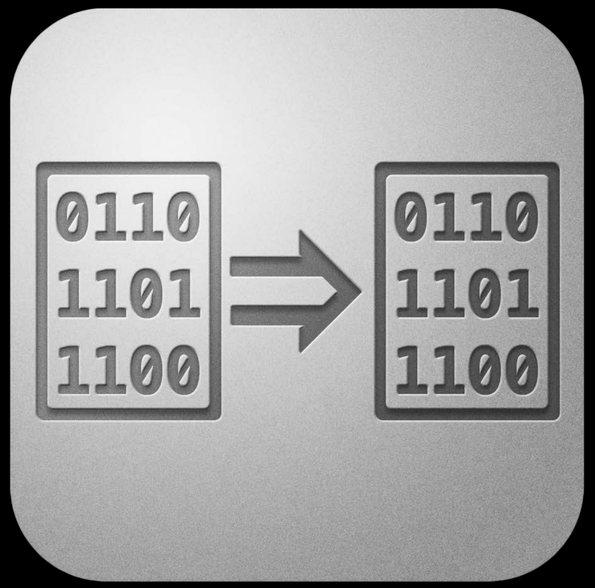
Data

Session


Session
Data
Session
Session

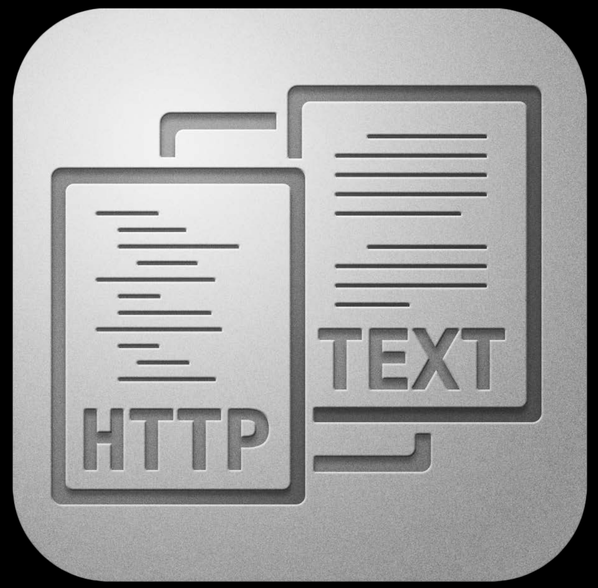
Session

`MCPeerID, MCSession`



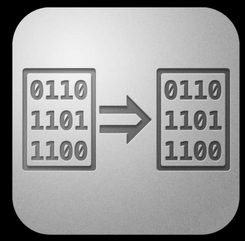
Messages

Streaming



Resources

Messages



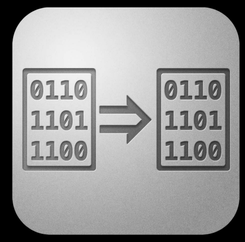
Send

```
[session sendData:data
          toPeers:peerIDs
         withMode:mode
            error:&error];
```

Receive

```
- (void)    session:(MCSession *)session
     didReceiveData:(NSData *)data
           fromPeer:(MCPeerID *)peerID;
```

Messages



Send

```
[session sendData:data
          toPeers:peerIDs
         withMode:mode
            error:&error];
```

Receive

```
- (void)    session:(MCSession *)session
     didReceiveData:(NSData *)data
           fromPeer:(MCPeerID *)peerID;
```

Streaming

Start a stream

```
NSOutputStream *outputStream =
    [session startStreamWithName:name
                          toPeer:peerID
                           error:&error];
```

Receive a stream

```
- (void)      session:(MCSession *)session
     didReceiveStream:(NSInputStream *)inputStream
             withName:(NSString *)name
             fromPeer:(MCPeerID *)peerID;
```

Streaming

Start a stream

```
NSOutputStream *outputStream =
    [session startStreamWithName:name
                          toPeer:peerID
                           error:&error];
```

Receive a stream

```
- (void)      session:(MCSession *)session
     didReceiveStream:(NSInputStream *)inputStream
             withName:(NSString *)name
             fromPeer:(MCPeerID *)peerID;
```

Resources

Send

```
NSProgress *progress =
      [session sendResourceAtURL:url
                        withName:name
                          toPeer:peerID
               completionHandler:completionHandler];
```

Resources

Start receiving resource

```
- (void)                    session:(MCSession *)session
didStartReceivingResourceWithName:(NSString *)name
                          fromPeer:(MCPeerID *)peerID
                      withProgress:(NSProgress *)progress;
```

Finish receiving resource

```
- (void)                     session:(MCSession *)session
didFinishReceivingResourceWithName:(NSString *)name
                          fromPeer:(MCPeerID *)peerID
                             atURL:(NSURL *)localURL
                         withError:(NSError *)error;
```

Resources

Start receiving resource

```
- (void)                    session:(MCSession *)session
didStartReceivingResourceWithName:(NSString *)name
                         fromPeer:(MCPeerID *)peerID
                     withProgress:(NSProgress *)progress;
```

Finish receiving resource

```
- (void)                     session:(MCSession *)session
didFinishReceivingResourceWithName:(NSString *)name
                          fromPeer:(MCPeerID *)peerID
                             atURL:(NSURL *)localURL
                         withError:(NSError *)error;
```

Summary

Discovery phase and session phase

UI-based and Programmatic discovery

Send data API—Messages, streaming, and resources

More in-depth info in last year's talk

Multipeer Connectivity on OS X


Demijan
MacBook Air


Demijan
MacBook Air


Choose 1 to 7 invitees:
Searching…
Cancel
Done
MacBook Air


Choose 1 to 7 invitees:
Searching…
Cancel
Done
MacBook Air

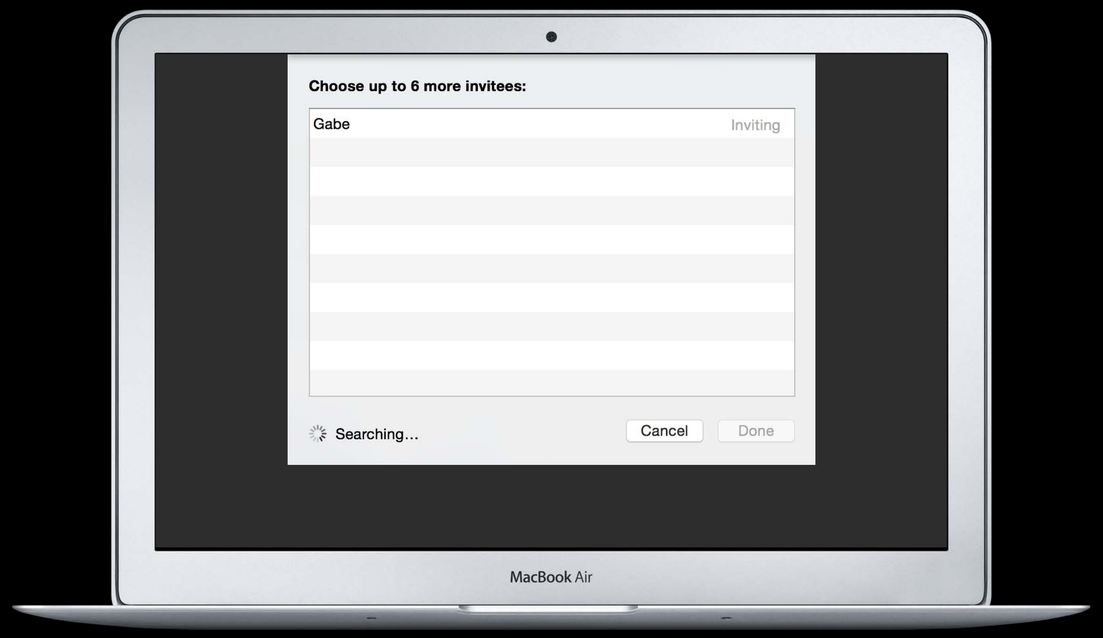

Gabe
MacBook Air


Choose 1 to 7 invitees:
Gabe
Invite
Searching…
Cancel
Done
MacBook Air

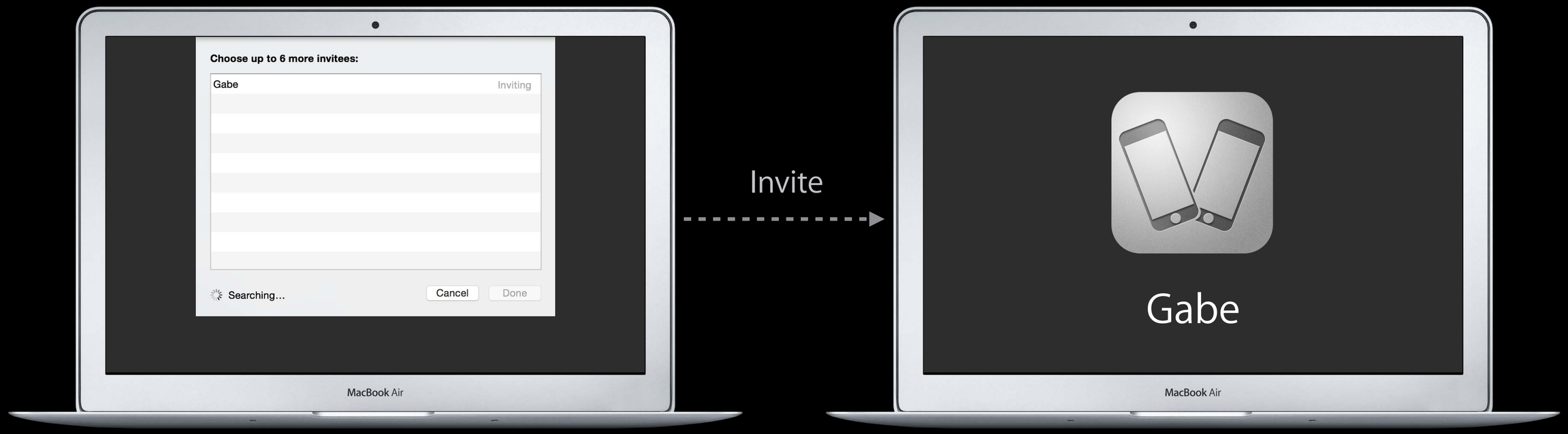

Gabe
MacBook Air

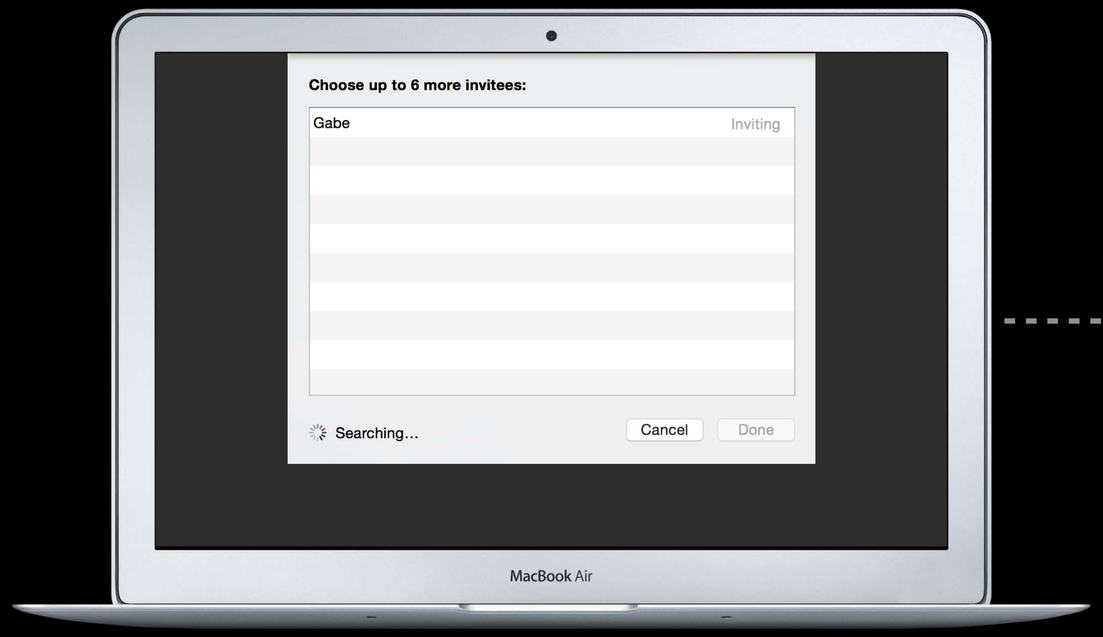

Choose 1 to 7 invitees:
Gabe
Invite
Searching…
Cancel
Done
MacBook Air


Gabe
MacBook Air


Choose up to 6 more invitees:
Gabe
Inviting
Searching…
Cancel
Done
MacBook Air

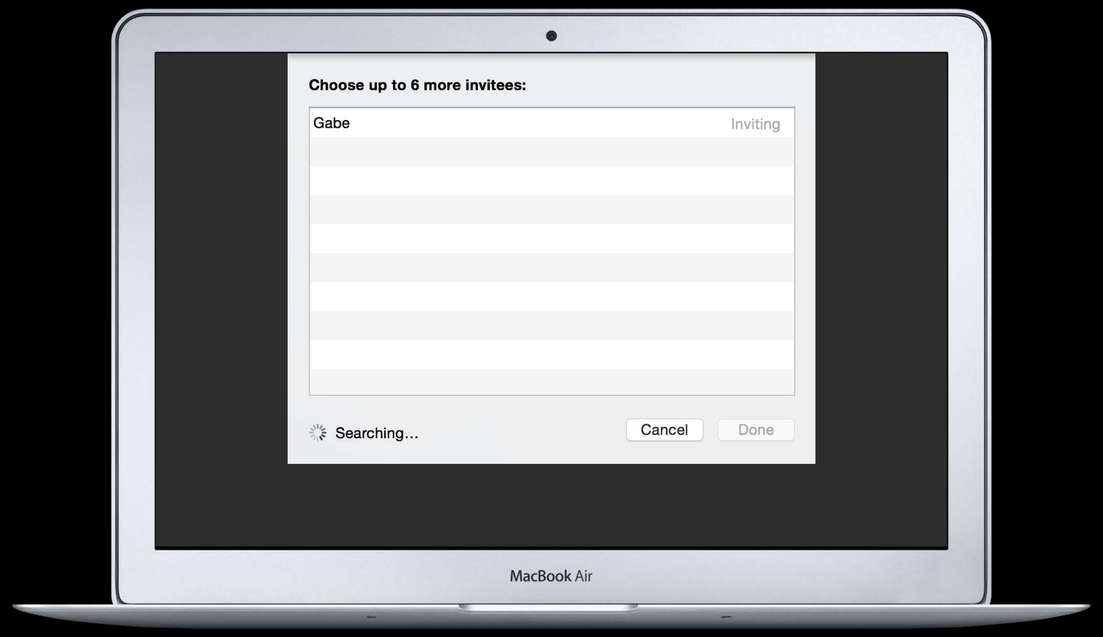

Gabe
MacBook Air


Choose up to 6 more invitees:
Gabe
Inviting
Searching…
Cancel
Done
MacBook Air
Invite
Gabe
MacBook Air

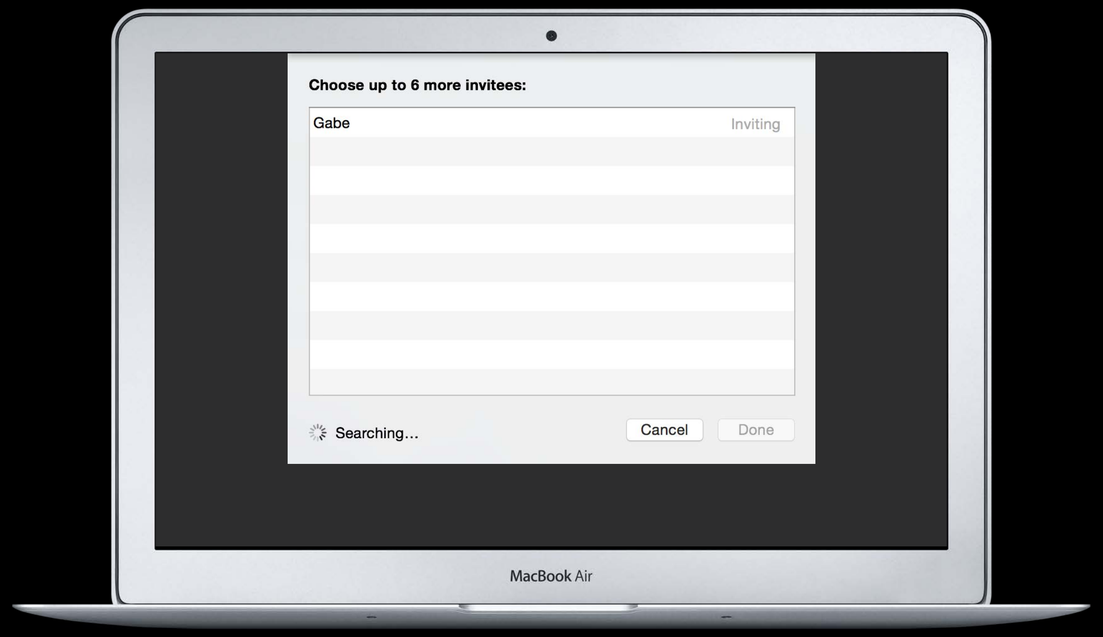

Choose up to 6 more invitees:
Gabe
Inviting
Searching…
Cancel
Done
MacBook Air

Invite

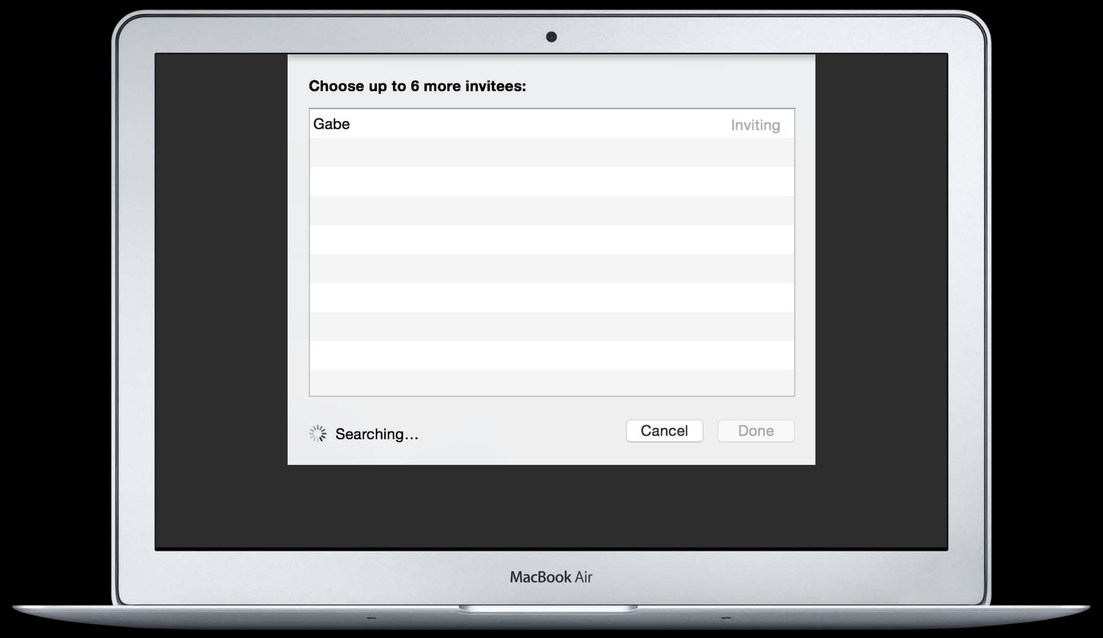

“Demijan” wants to connect.
Decline
Accept
Gabe
MacBook Air


Choose up to 6 more invitees:
Gabe
Inviting
Searching…
Cancel
Done
MacBook Air


“Demijan” wants to connect.
Decline
Accept
Gabe
MacBook Air


Choose up to 6 more invitees:
Gabe
Inviting
Searching…
Cancel
Done
MacBook Air


“Demijan” wants to connect.
Decline
Accept
Gabe
MacBook Air


Choose up to 6 more invitees:
Gabe
Inviting
Searching…
Cancel
Done
MacBook Air


Gabe
MacBook Air

Choose up to 6 more invitees:

Gabe Connecting

Searching…

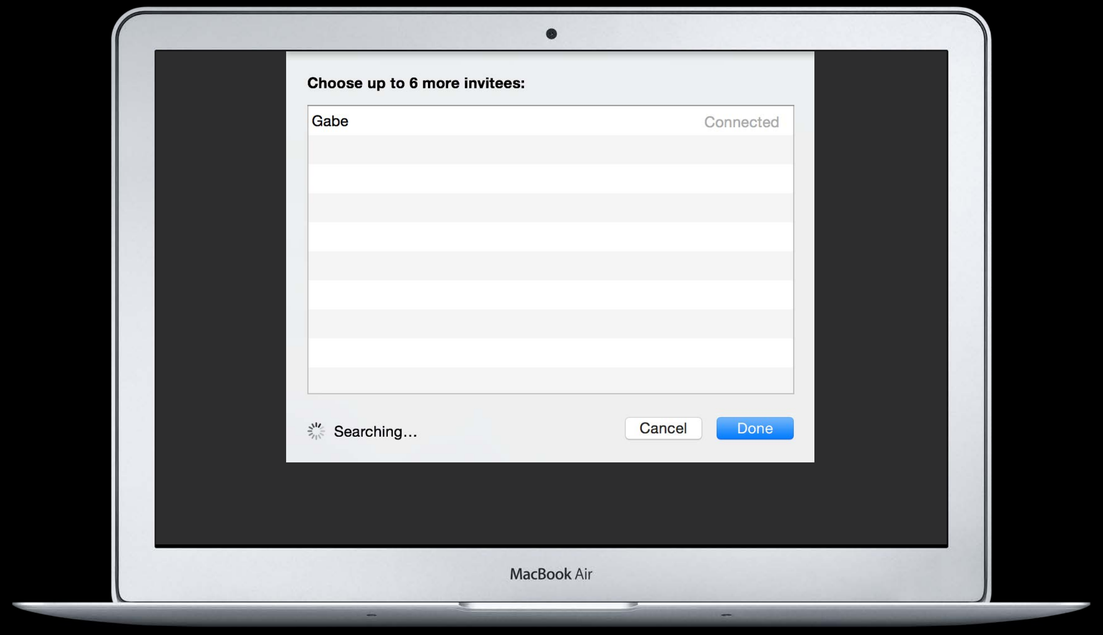
Cancel Done

MacBook Air

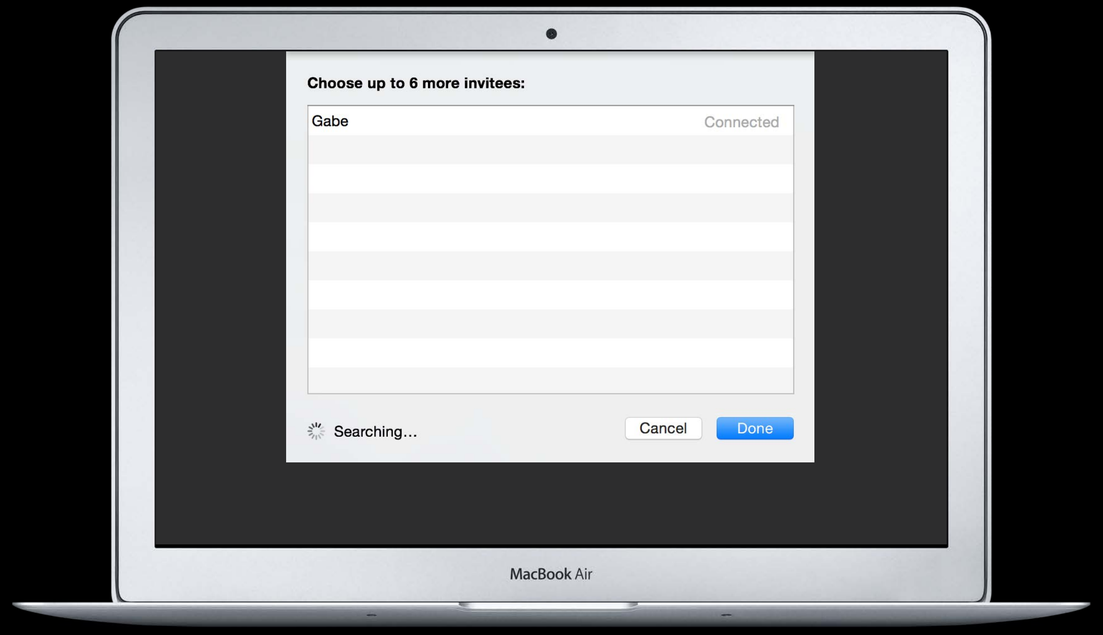
Accept

Gabe

MacBook Air


Choose up to 6 more invitees:
Gabe
Connected
Searching…
Cancel
Done
MacBook Air

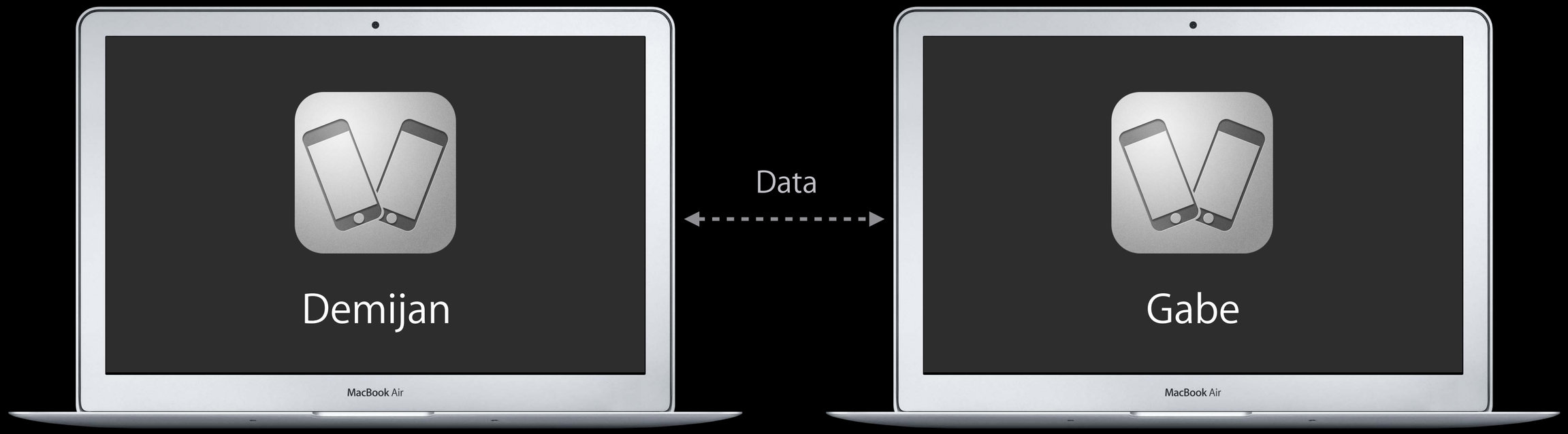

Gabe
MacBook Air


Choose up to 6 more invitees:
Gabe
Connected
Searching…
Cancel
Done
MacBook Air


Gabe
MacBook Air


Demijan
MacBook Air


Gabe
MacBook Air


Data
Demijan
MacBook Air
Gabe
MacBook Air

MCAdvertiserAssistant

OS X

```
// initialize
MCAdvertiserAssistant *assistant =
[[MCAdvertiserAssistant alloc]
        initWithServiceType:type
              discoveryInfo:nil
                    session:session];
```

MCAdvertiserAssistant

OS X

```
// initialize
MCAdvertiserAssistant *assistant =
[[MCAdvertiserAssistant alloc]
        initWithServiceType:type
              discoveryInfo:nil
                    session:session];


// start advertising
[assistant start];
```

MCBrowserViewController

OS X

Subclasses NSViewController

```
// initialize, set delegate
MCBrowserViewController *browserVC =
  [[MCBrowserViewController] alloc]
            initWithServiceType:serviceType
                        session:session];
browserVC.delegate = self;
```

MCBrowserViewController

OS X

Subclasses NSViewController

```
// initialize, set delegate
MCBrowserViewController *browserVC =
  [[MCBrowserViewController] alloc]
            initWithServiceType:serviceType
                        session:session];
browserVC.delegate = self;
```

Present modally as a sheet

```
// present
[self presentViewControllerAsSheet:browserVC];
```

MCBrowserViewController

OS X

```
// NSWindow based App
self.bvcWindow = [NSWindow
          windowWithContentViewController:self.browserViewController];
[NSApp beginSheet:self.bvcWindow
   modalForWindow:[NSApp mainWindow]
    modalDelegate:nil
   didEndSelector:NULL
      contextInfo:NULL];
```

MCBrowserViewController

OS X

```
// NSWindow based App
self.bvcWindow = [NSWindow
          windowWithContentViewController:self.browserViewController];
[NSApp beginSheet:self.bvcWindow
   modalForWindow:[NSApp mainWindow]
    modalDelegate:nil
   didEndSelector:NULL
      contextInfo:NULL];
```

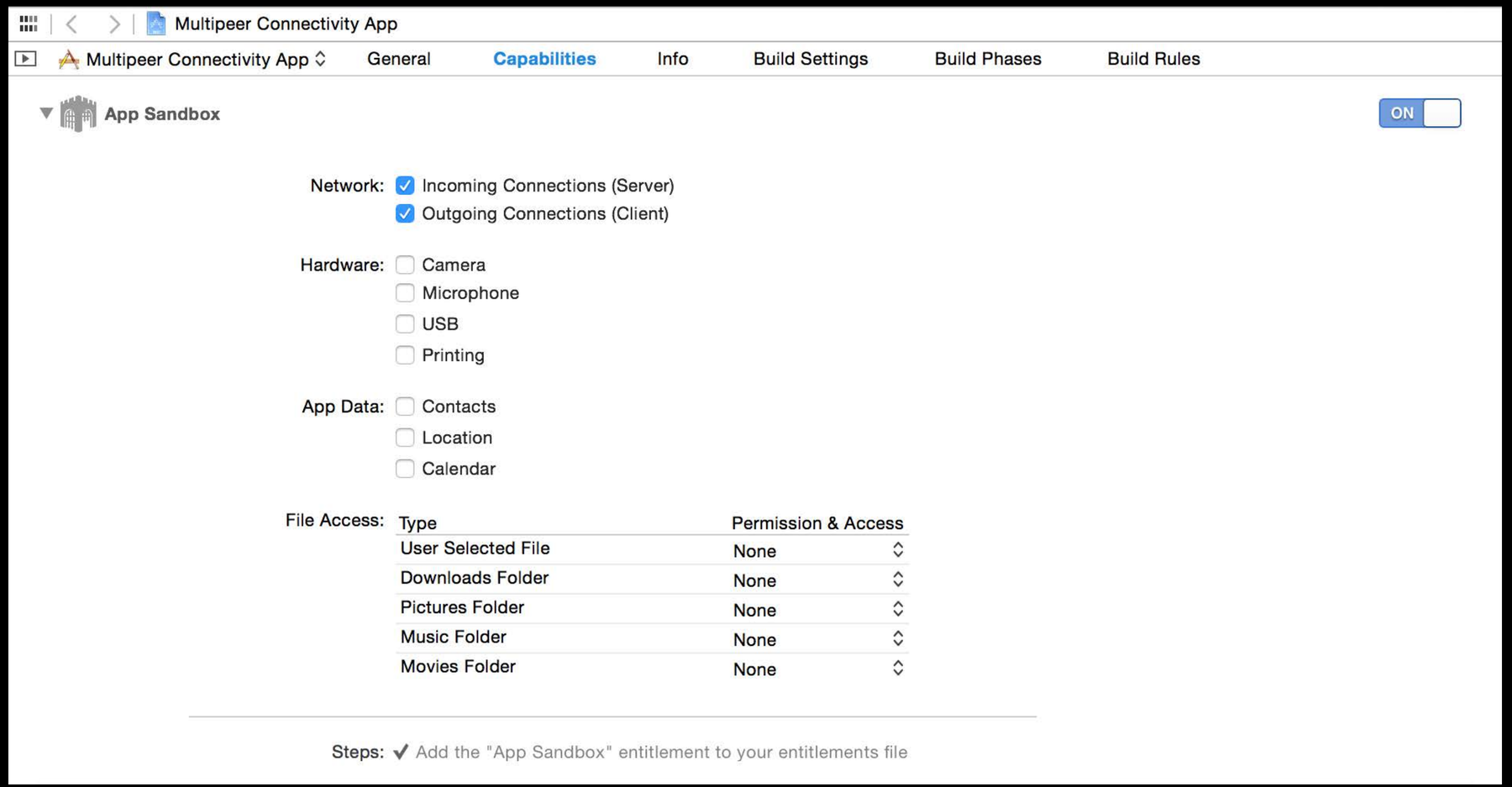
MCBrowserViewControllerDelegate

OS X

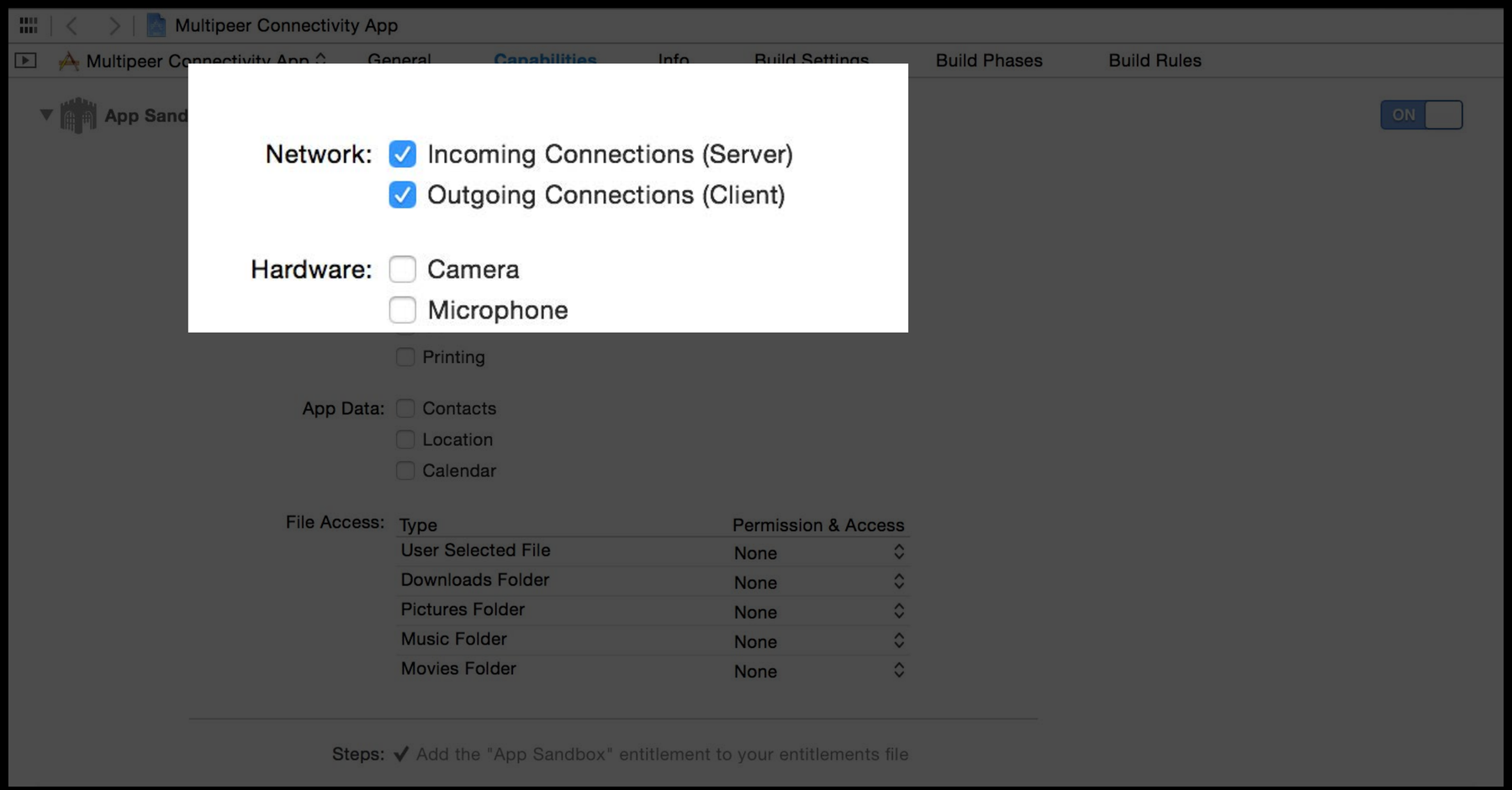
```
// done button tapped
- (void)browserViewControllerDidFinish:
   (MCBrowserViewController *)browserVC
{
   [self dismissViewController:browserVC];
}

// cancel button tapped
- (void)browserViewControllerWasCancelled:
   (MCBrowserViewController *)browserVC
{
   [self dismissViewController:browserVC];
}
```

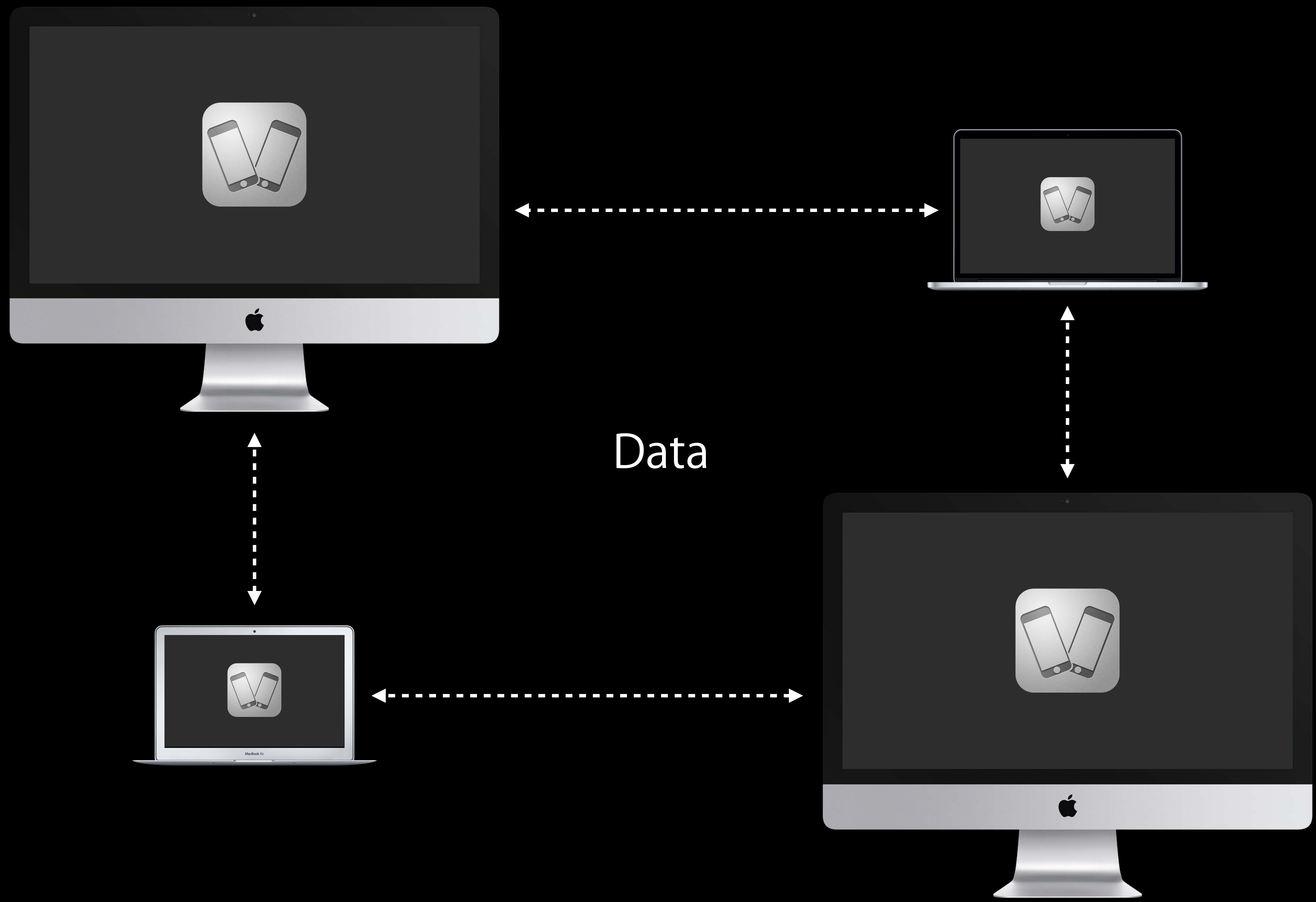
Entitlements

Entitlements



Entitlements




Data

Demo

Eric Chien
Software Engineer

Summary

API same as iOS

OS X specific presentation of the MCBrowserViewController

Networking entitlements

Best Practices


PeerID [Mac]
Advertiser
MacBook Air
PeerID
Browser


PeerID [Mac]
Advertiser
MacBook Air
PeerID
Browser
PeerID [Mac]


PeerID
Browser
PeerID [Mac]


PeerID
Browser
PeerID [Mac]

PeerID [Mac]

Advertiser

MacBook Air

PeerID

Browser

PeerID [Mac]


PeerID [Mac]
Advertiser
MacBook Air
PeerID
Browser
PeerID [Mac]
PeerID [Mac]

Reuse Peer Objects

Reuse Peer Objects

Serialize, store in defaults

```
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
NSData *peerIDData = [NSKeyedArchiver archivedDataWithRootObject:peerID];
[defaults setObject:peerIDData forKey:kPeerIDKey];
[defaults synchronize];
```

Reuse Peer Objects

Serialize, store in defaults

```
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
NSData *peerIDData = [NSKeyedArchiver archivedDataWithRootObject:peerID];
[defaults setObject:peerIDData forKey:kPeerIDKey];
[defaults synchronize];
```

Deserialize, retrieve from defaults

```
NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
NSData *peerIDData = [defaults dataForKey:kPeerIDKey];
MCPeerID *peerID = [NSKeyedUnarchiver unarchiveObjectWithData:peerIDData];
```

Auto-inviting

Auto-inviting

`foundPeer:`

`foundPeer:`

Auto-inviting

Use a deterministic algorithm to determine who should invite

```
- (void)browser:(MCNearbyServiceBrowser *)browser foundPeer:
(MCPeerID *)nearbyPeerID withDiscoveryInfo:(NSDictionary *)info
{
    // compare hash values
    if (myPeerID.hash > nearbyPeerID.hash)
    {
        [self.browser invitePeer:nearbyPeerID
                       toSession:session
                     withContext:nil
                         timeout:0];
    }
}
```

Discovery Info (Bonjour TXT Records)

```
// MCAdvertiserAssistant
MCAdvertiserAssistant *assistant =
[[MCAdvertiserAssistant alloc]
         initWithServiceType:type
               discoveryInfo:info
                     session:session];

// MCNearbyServiceAdvertiser
MCNearbyServiceAdvertiser *advertiser =
[[MCNearbyServiceAdvertiser alloc]
          initWithPeerID:myPeerID
             serviceType:type
           discoveryInfo:info];
```

Discovery Info (Bonjour TXT Records)

Keep discovery info small

Both keys and values must be NSString

TXT record entry—key=value

Maximum size for each TXT record entry—256 bytes

More details—http://tools.ietf.org/html/rfc6763

Custom Discovery

Discovery Phase

UI-based

```
MCPeerID
MCBrowserViewController
MCAdvertiserAssistant
```

Programmatic

```
MCPeerID
MCNearbyServiceBrowser
MCNearbyServiceAdvertiser
```

Discovery

UI-based

```
MCPeerID
MCBrowserViewController
MCAdvertiserAssistant
```

Programmatic

```
MCPeerID
MCNearbyServiceBrowser
MCNearbyServiceAdvertiser
```

Custom

```
MCPeerID
MCSession
```

Discover and Establish a Channel

Peer ID

Step 1—Serialize and exchange

Peer ID

Step 1—Serialize and exchange


```
[NSKeyedArchiver archivedDataWithRootObject:]
```

Peer ID

Step 1—Serialize and exchange


```
[NSKeyedArchiver archivedDataWithRootObject:]
```

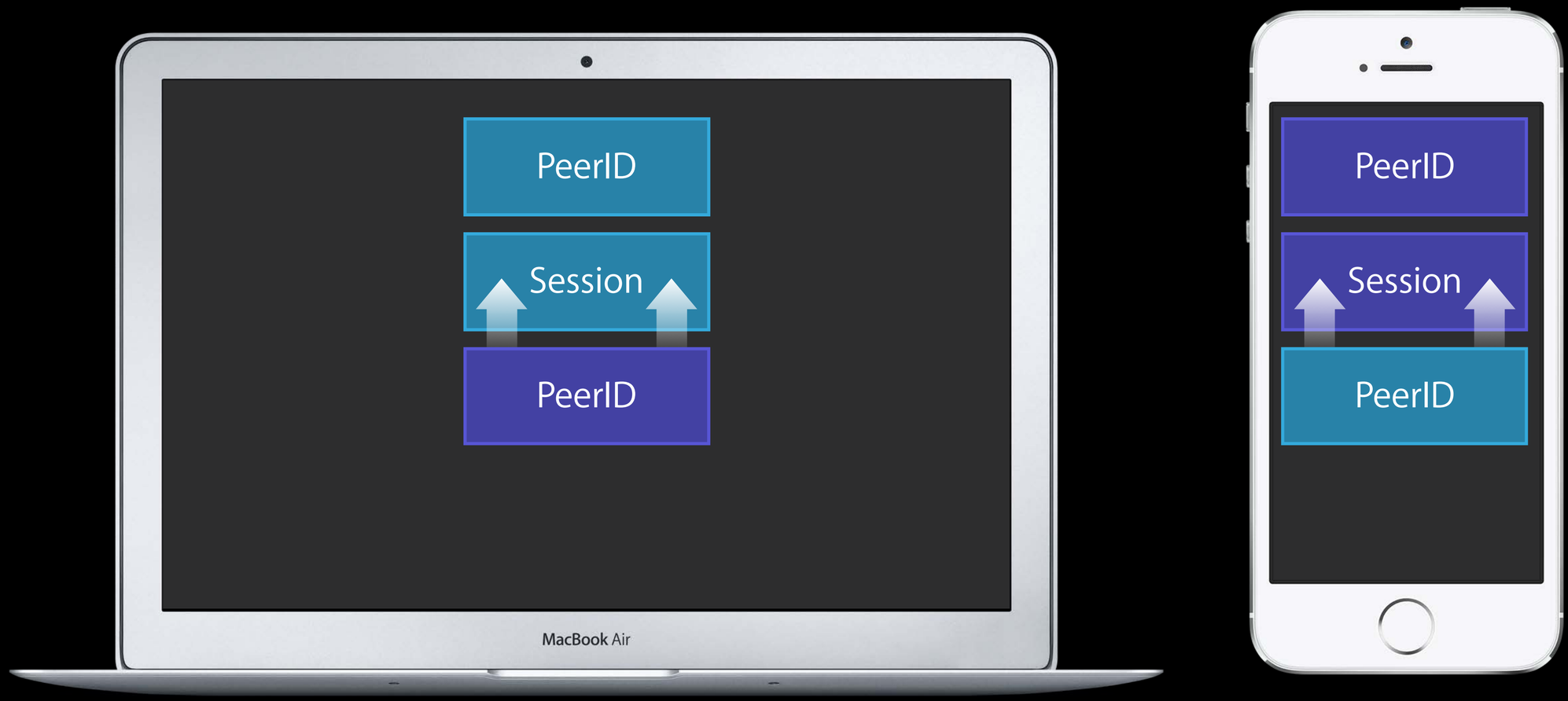
Peer ID

Step 1—Serialize and exchange


```
[NSKeyedUnarchiver unarchiveObjectWithData:]
```

Peer ID

Step 2—Create and exchange




```
[MCSession nearbyConnectionDataForPeer:withCompletionHandler:]
```

Peer ID

Step 2—Create and exchange

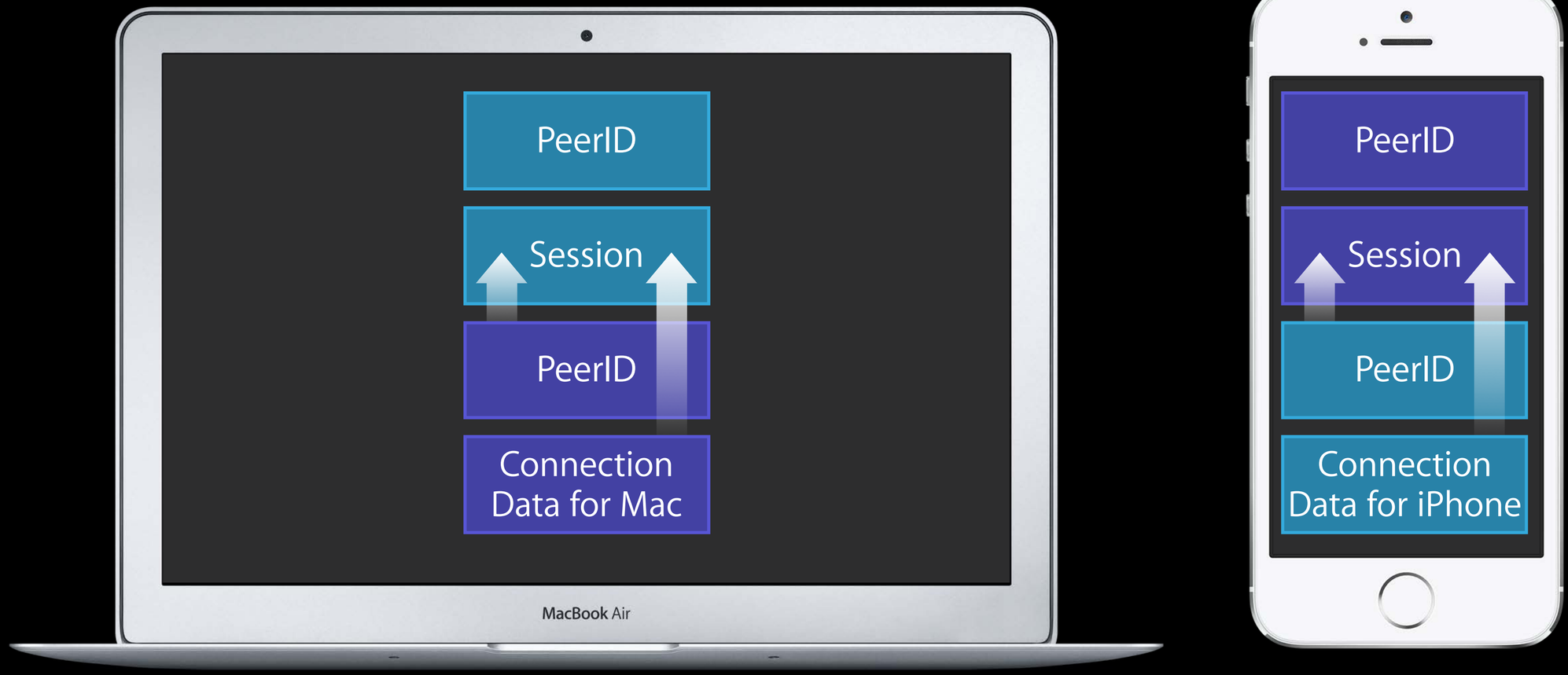

```
[MCSession nearbyConnectionDataForPeer:withCompletionHandler:]
```

Peer ID

Step 2—Create and exchange


```
[MCSession nearbyConnectionDataForPeer:withCompletionHandler:]
```

Connect Peer

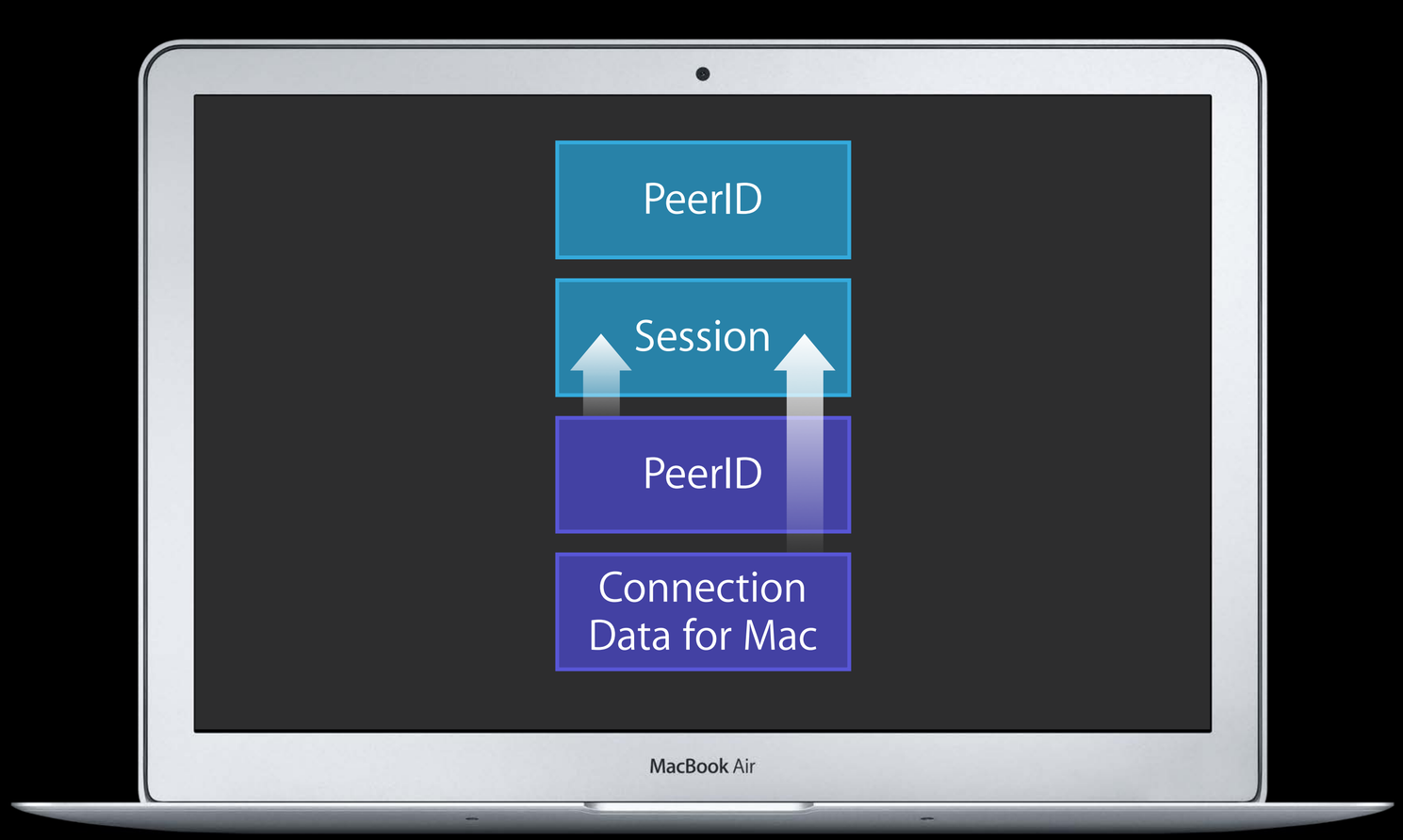


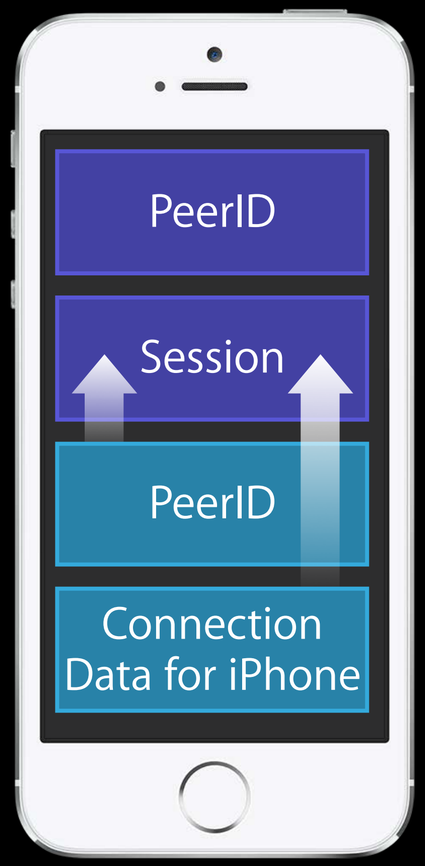

```
[MCSession connectPeer:withConnectionData:]
```

Connect Peer

`…peer:didChangeState:…`

`…peer:didChangeState:…`





`[MCSession connectPeer:withConnectionData:]`

Custom Discovery APIs

Serialize

```
NSData *peerIDData = [NSKeyedArchiver archivedDataWithRootObject:peerID];
```

Deserialize

```
MCPeerID *peerID = [NSKeyedUnarchiver unarchiveObjectWithData:peerIDData];
```

Custom Discovery APIs

Serialize

```
NSData *peerIDData = [NSKeyedArchiver archivedDataWithRootObject:peerID];
```

Deserialize

```
MCPeerID *peerID = [NSKeyedUnarchiver unarchiveObjectWithData:peerIDData];
```

Custom Discovery APIs

Serialize

```
NSData *peerIDData = [NSKeyedArchiver archivedDataWithRootObject:peerID];
```

Deserialize

```
MCPeerID *peerID = [NSKeyedUnarchiver unarchiveObjectWithData:peerIDData];
```

Custom Discovery APIs

Get nearby connection data

```
[session nearbyConnectionDataForPeer:peerID
                withCompletionHandler:^(NSData *connectionData,
                                        NSError *error) {
    // send connection data to nearby peer
}];
```

Custom Discovery APIs

Get nearby connection data

```
[session nearbyConnectionDataForPeer:peerID
                withCompletionHandler:^(NSData *connectionData,
                                        NSError *error) {
    // send connection data to nearby peer
}];
```

Custom Discovery APIs

Connect to session

```
[session connectPeer:peerID withNearbyConnectionData:data timeout:timeout];
```

Cancel connect

```
[session cancelConnectPeer:peerID];
```

Custom Discovery APIs

Connect to session

```
[session connectPeer:peerID withNearbyConnectionData:data timeout:timeout];
```

Cancel connect

```
[session cancelConnectPeer:peerID];
```

Custom Discovery APIs

Connect to session

```
[session connectPeer:peerID withNearbyConnectionData:data timeout:timeout];
```

Cancel connect

```
[session cancelConnectPeer:peerID];
```

Summary

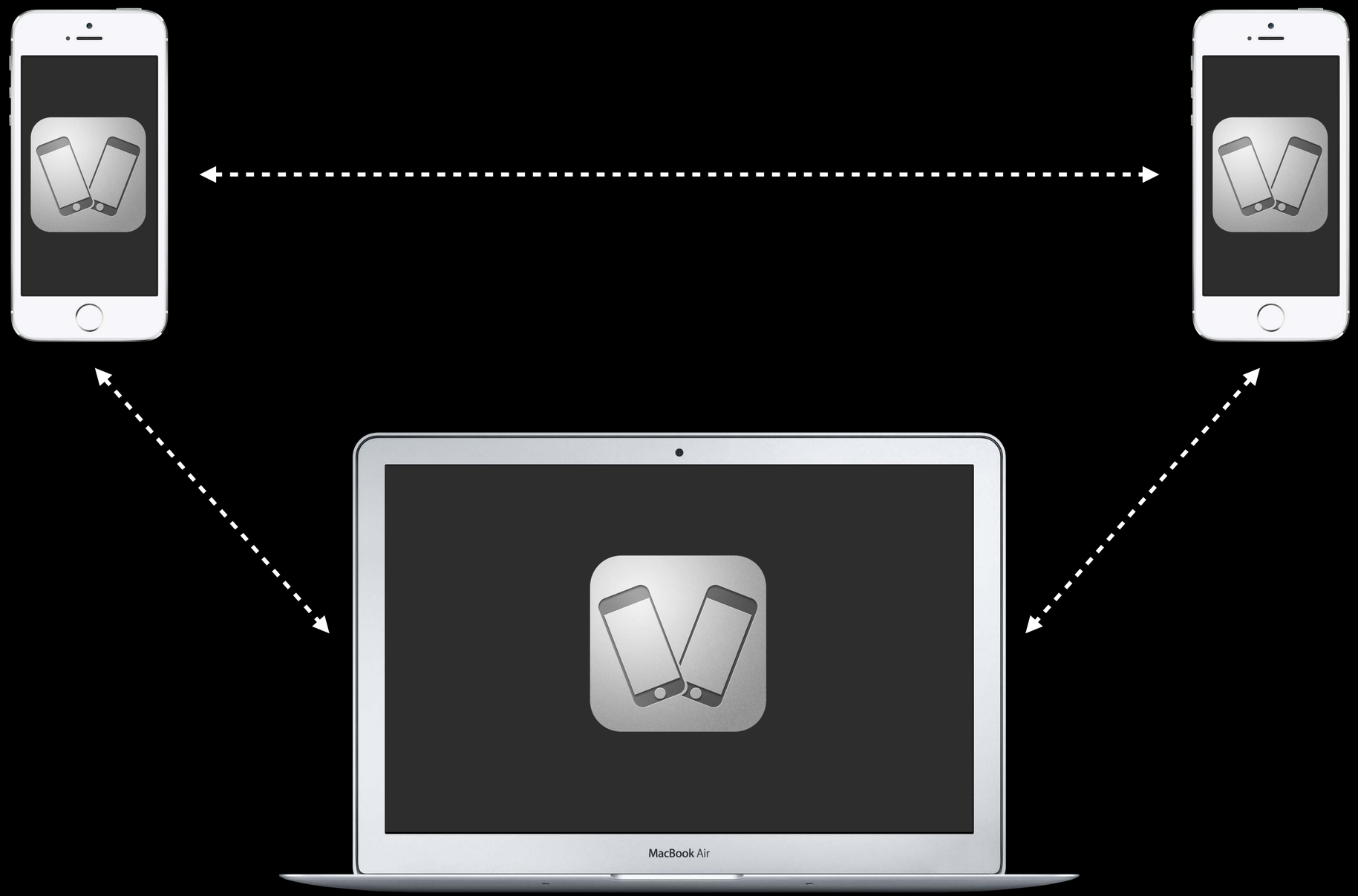
Fully customized discovery

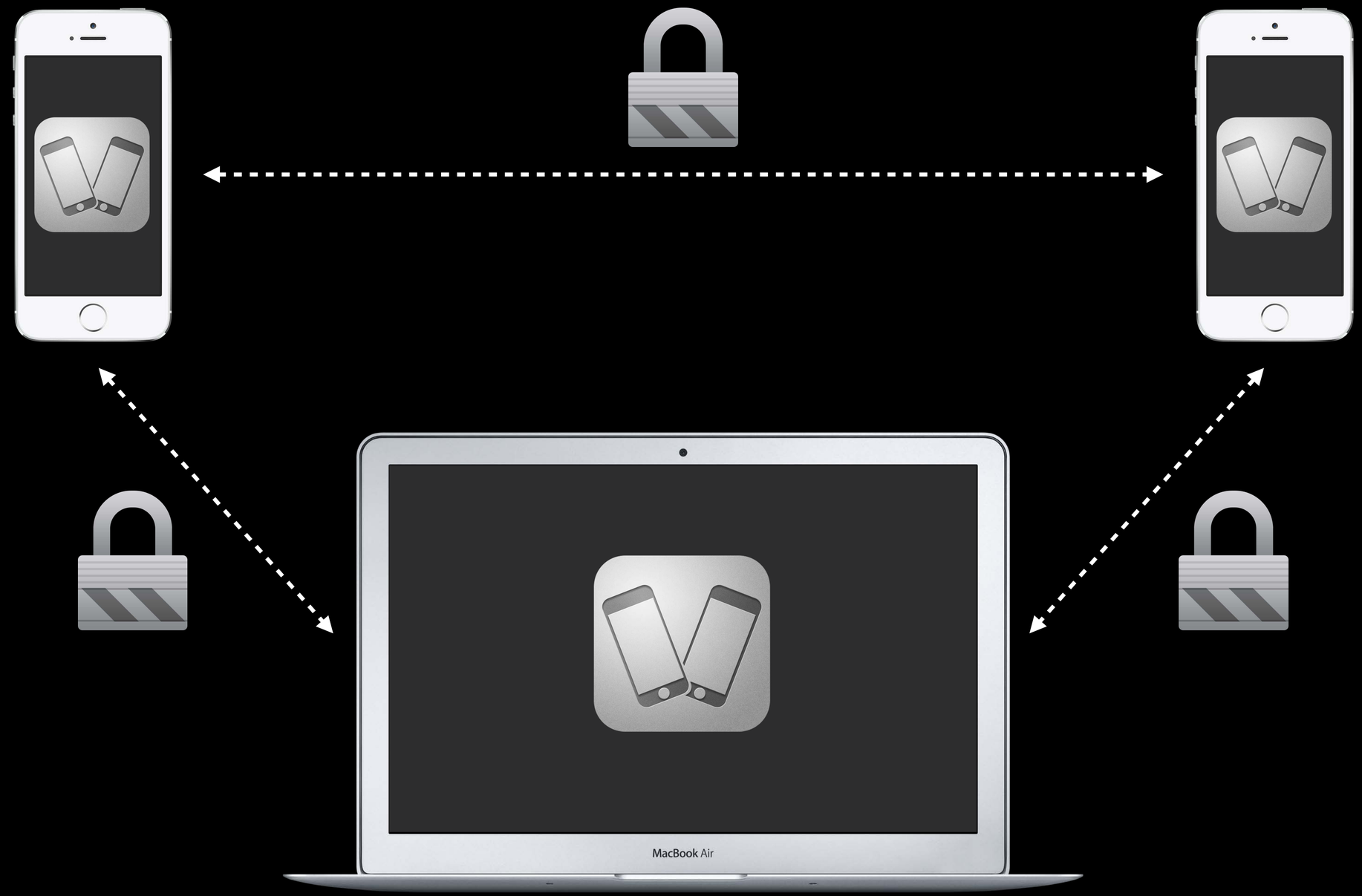
Two-step process

Exchange peerIDs

Exchange nearby connection data

Authentication


MacBook Air


MacBook Air

Encryption

Authentication

Digital Identity
(`SecIdentityRef`)

Private Key
(SecKeyRef)

Certificate
(`SecCertificateRef`)

Distributing Identities

Web server

Email attachment

Mobile Device Management server

More info—Tech QA 1745

https://developer.apple.com/library/ios/qa/qa1745/_index.html

Importing Digital Identity

```
// Get password for importing an identity.
const void *keys[] = {kSecImportExportPassphrase};
const void *values[] = {CFSTR(“password”)};

CFDictionaryRef optionsDictionary = CFDictionaryCreate(NULL, keys, values,
                                          1, NULL, NULL);

// Get identity data from PKCS#12 file.
NSData *identityData = [NSData dataWithContentsOfURL:identityDataURL];

// Import digital identity.
CFArray items = NULL;
SecPKCS12Import(CFDataRef)identityData, optionsDictionary, &items);
```

Importing Digital Identity

```
// Get password for importing an identity.
const void *keys[] = {kSecImportExportPassphrase};
const void *values[] = {CFSTR(“password”)};

CFDictionaryRef optionsDictionary = CFDictionaryCreate(NULL, keys, values,
                                    1, NULL, NULL);

// Get identity data from PKCS#12 file.
NSData *identityData = [NSData dataWithContentsOfURL:identityDataURL];

// Import digital identity.
CFArray items = NULL;
SecPKCS12Import(CFDataRef)identityData, optionsDictionary, &items);
```

Importing Digital Identity

```
// Get password for importing an identity.
const void *keys[] = {kSecImportExportPassphrase};
const void *values[] = {CFSTR(“password”)};

CFDictionaryRef optionsDictionary = CFDictionaryCreate(NULL, keys, values,
                                          1, NULL, NULL);

// Get identity data from PKCS#12 file.
NSData *identityData = [NSData dataWithContentsOfURL:identityDataURL];

// Import digital identity.
CFArray items = NULL;
SecPKCS12Import(CFDataRef)identityData, optionsDictionary, &items);
```

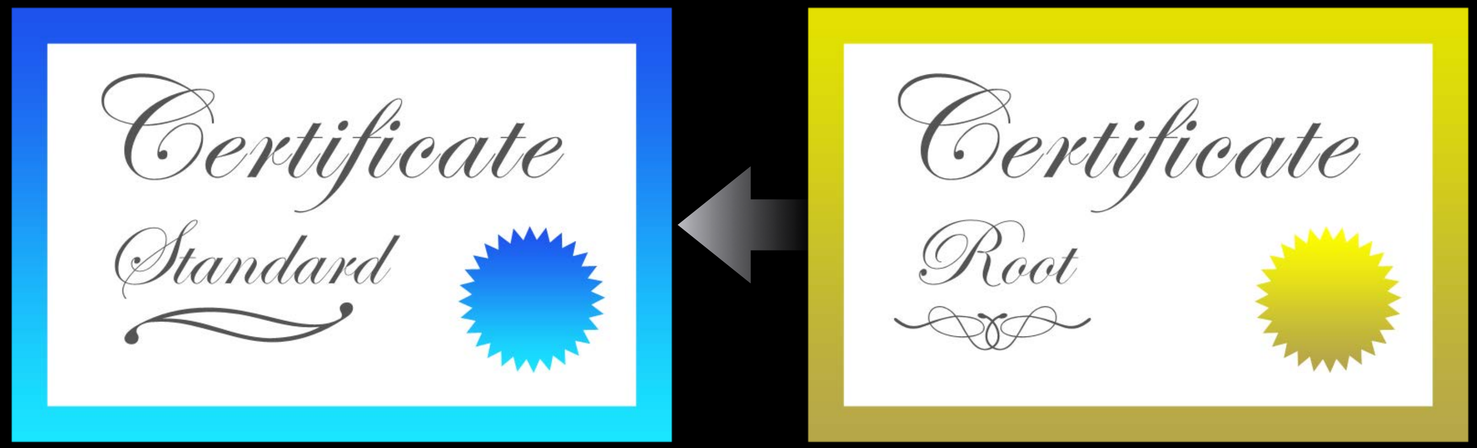
Importing Digital Identity

```
// Get password for importing an identity.
const void *keys[] = {kSecImportExportPassphrase};
const void *values[] = {CFSTR(“password”)};

CFDictionaryRef optionsDictionary = CFDictionaryCreate(NULL, keys, values,
                                          1, NULL, NULL);

// Get identity data from PKCS#12 file.
NSData *identityData = [NSData dataWithContentsOfURL:identityDataURL];

// Import digital identity.
CFArray items = NULL;
SecPKCS12Import(CFDataRef)identityData, optionsDictionary, &items);
```

Digital Identity
(SecIdentityRef)

Private Key
(SecKeyRef)

Certificate
(SecCertificateRef)


Certificate
Standard
Certificate
Root

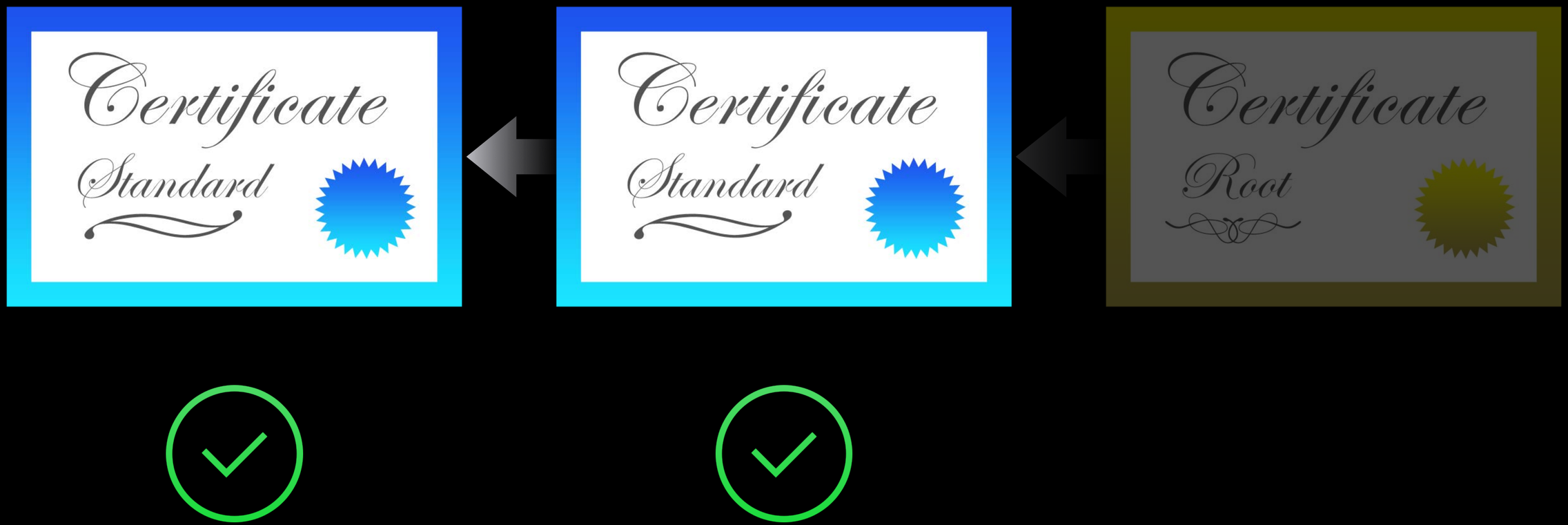

Certificate
Standard
Certificate
Standard
Certificate
Root

Evaluating Chain of Trust

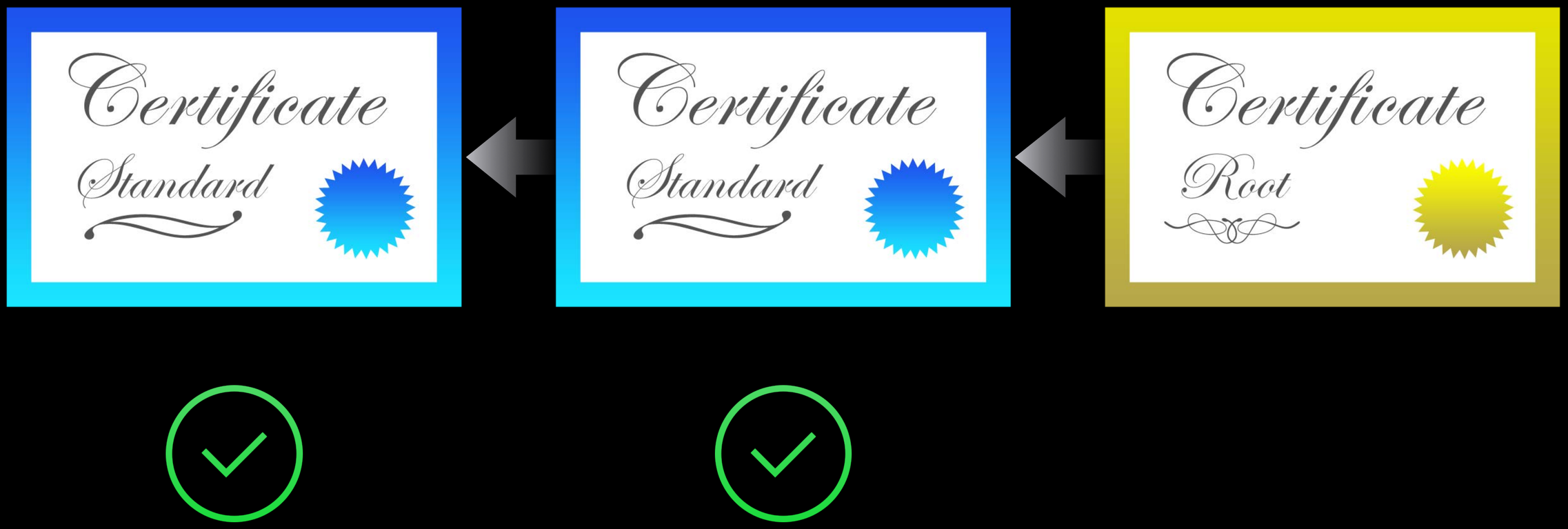
Evaluating Chain of Trust

Evaluating Chain of Trust

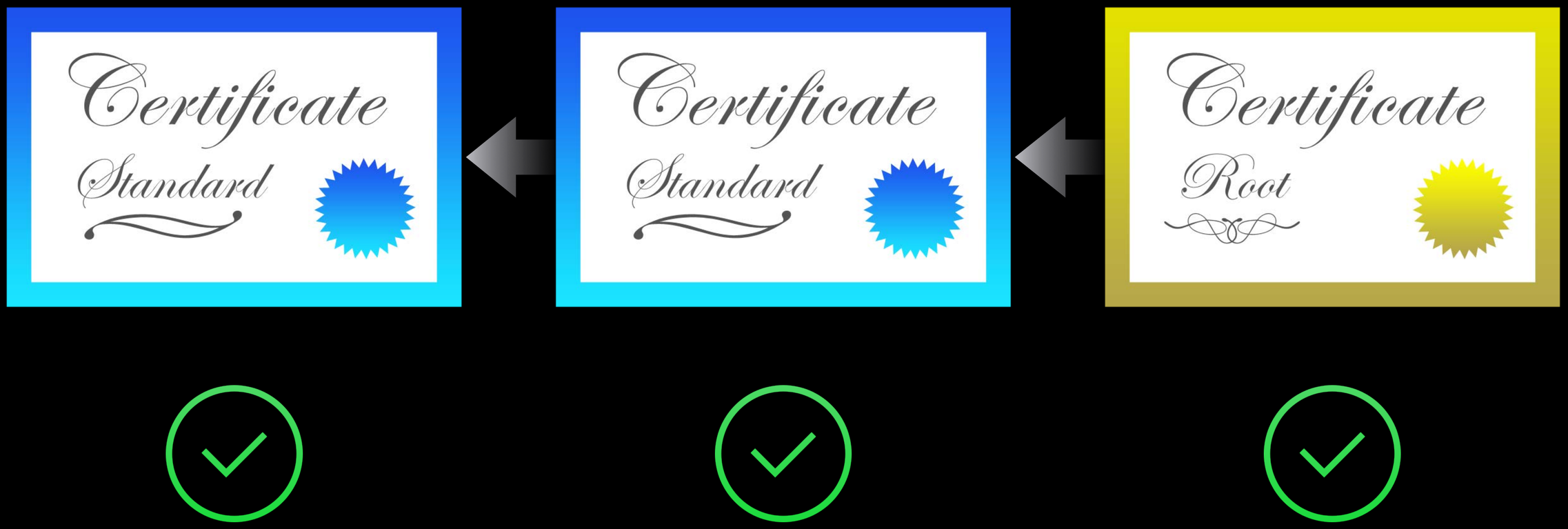
Evaluating Chain of Trust



Evaluating Chain of Trust



Evaluating Chain of Trust



Certificate Validation

Create policy object

Create trust object (using certificates and policy)

Evaluate trust object

Certificate Validation

```
NSArray *policy = [NSArray arrayWithObject:
   (__bridge id)SecPolicyCreateBasicX509()];

SecTrustRef *trust;
SecTrustCreateWithCertificates((__bridge CFArrayRef)certificate, (__bridge
CFArrayRef)policy, &trust);

SecTrustSetAnchorCertificates(trust, (__bridge CFArrayRef)[NSArray
arrayWithObject:(__bridge id)anchor]);

SecTrustResultType trustResult;
SecTrustEvaluate(trust, &trustResult);

allowConnection = (trustResult == kSecTrustResultUnspecified);
```

Certificate Validation

```
NSArray *policy = [NSArray arrayWithObject:
   (__bridge id)SecPolicyCreateBasicX509()];

SecTrustRef *trust;
SecTrustCreateWithCertificates((__bridge CFArrayRef)certificate, (__bridge
CFArrayRef)policy, &trust);

SecTrustSetAnchorCertificates(trust, (__bridge CFArrayRef)[NSArray
arrayWithObject:(__bridge id)anchor]);

SecTrustResultType trustResult;
SecTrustEvaluate(trust, &trustResult);

allowConnection = (trustResult == kSecTrustResultUnspecified);
```

Certificate Validation

```
NSArray *policy = [NSArray arrayWithObject:
   (__bridge id)SecPolicyCreateBasicX509()];

SecTrustRef *trust;
SecTrustCreateWithCertificates((__bridge CFArrayRef)certificate, (__bridge
CFArrayRef)policy, &trust);

SecTrustSetAnchorCertificates(trust, (__bridge CFArrayRef)[NSArray
arrayWithObject:(__bridge id)anchor]);

SecTrustResultType trustResult;
SecTrustEvaluate(trust, &trustResult);

allowConnection = (trustResult == kSecTrustResultUnspecified);
```

Certificate Validation

```
NSArray *policy = [NSArray arrayWithObject:
   (__bridge id)SecPolicyCreateBasicX509()];

SecTrustRef *trust;
SecTrustCreateWithCertificates((__bridge CFArrayRef)certificate, (__bridge
CFArrayRef)policy, &trust);

SecTrustSetAnchorCertificates(trust, (__bridge CFArrayRef)[NSArray
arrayWithObject:(__bridge id)anchor]);

SecTrustResultType trustResult;
SecTrustEvaluate(trust, &trustResult);

allowConnection = (trustResult == kSecTrustResultUnspecified);
```

Certificate Validation

```
NSArray *policy = [NSArray arrayWithObject:
   (__bridge id)SecPolicyCreateBasicX509()];

SecTrustRef *trust;
SecTrustCreateWithCertificates((__bridge CFArrayRef)certificate, (__bridge
CFArrayRef)policy, &trust);

SecTrustSetAnchorCertificates(trust, (__bridge CFArrayRef)[NSArray
arrayWithObject:(__bridge id)anchor]);

SecTrustResultType trustResult;
SecTrustEvaluate(trust, &trustResult);

allowConnection = (trustResult == kSecTrustResultUnspecified);
```

Authentication


```
// instantiate with security
MCSession *session = [[MCSession alloc]
                initWithPeer:myPeerID
            securityIdentity:identity
        encryptionPreference:preference];
```

Authentication


```
- (void)          session:(MCSession *)session
    didReceiveCertificate:(NSArray *)certificate
                 fromPeer:(MCPeerID *)peerID
       certificateHandler:(void(^)(BOOL accept))certificateHandler
```

Summary

Distribution and importing of digital identities

Chain of trust evaluation

More Information

Paul Danbold
Core OS Technologies Evangelist
danbold@apple.com

Documentation
Multipeer Connectivity Framework Reference
http://developer.apple.com/library/ios/documentation/MultipeerConnectivity/Reference/MultipeerConnectivityFramework

Sample Code
MultipeerGroupChat
http://developer.apple.com/library/ios/samplecode/MultipeerGroupChat

Apple Developer Forums
http://devforums.apple.com

Related Sessions

What's New in Foundation Networking	Nob Hill	Tuesday 3:15PM
StoryBoards and Controllers on OS X	Pacific Heights	Tuesday 4:30PM

Labs

• Networking Lab	Core OS Lab B	Wednesday 9:00AM
• Multipeer Connectivity Lab	Core OS Lab A	Wednesday 10:15AM
• View Controllers and Cocoa Lab	Frameworks Lab B	Thursday 11:30 AM
• Multipeer Connectivity Lab 2	Core OS Lab B	Friday 9:00AM

WWDC14