Nearby Networking with Multipeer Connectivity

Session 708

Demijan Klinc
Software Engineer

What Is Multipeer Connectivity?

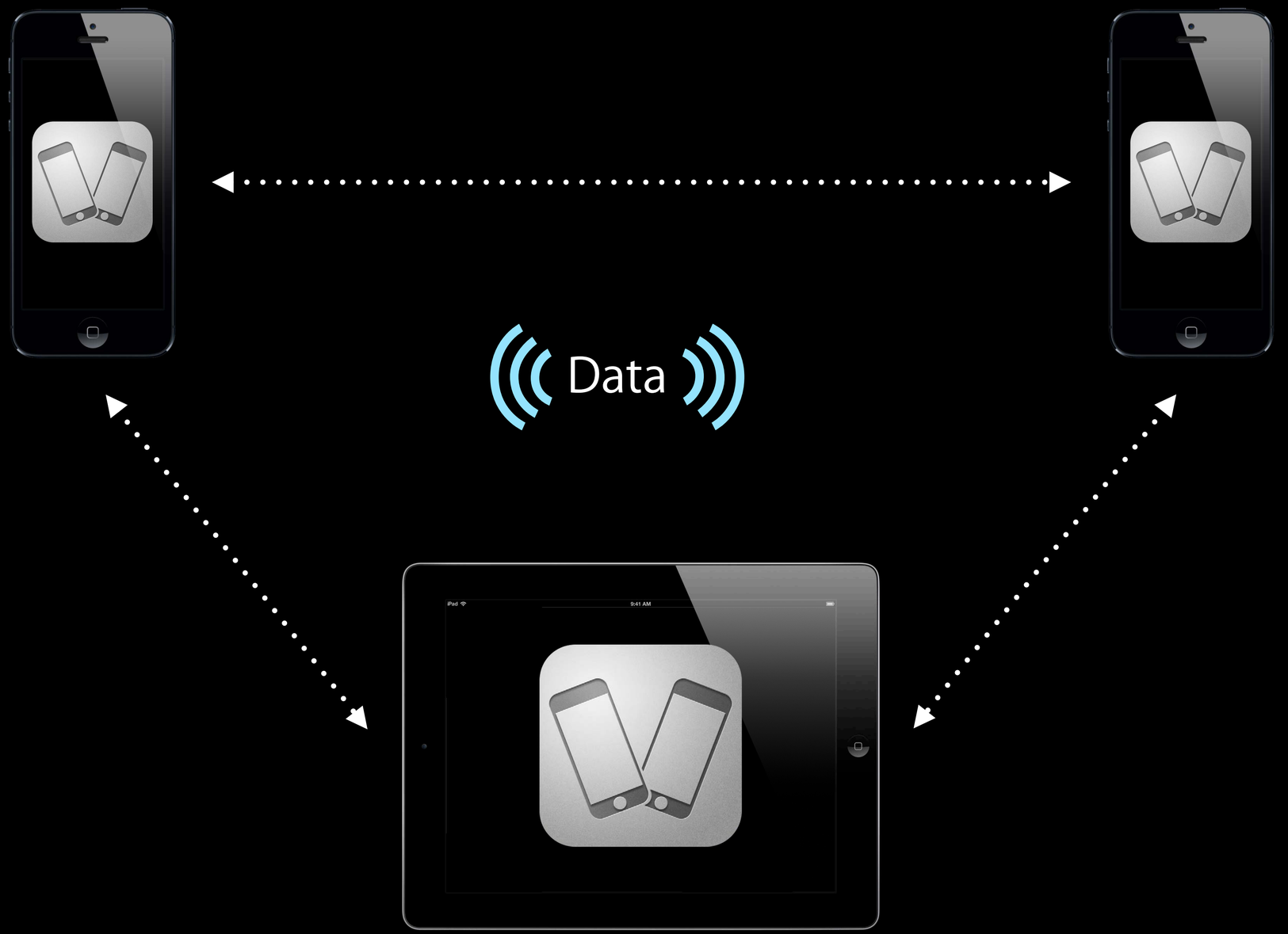
Facilitates

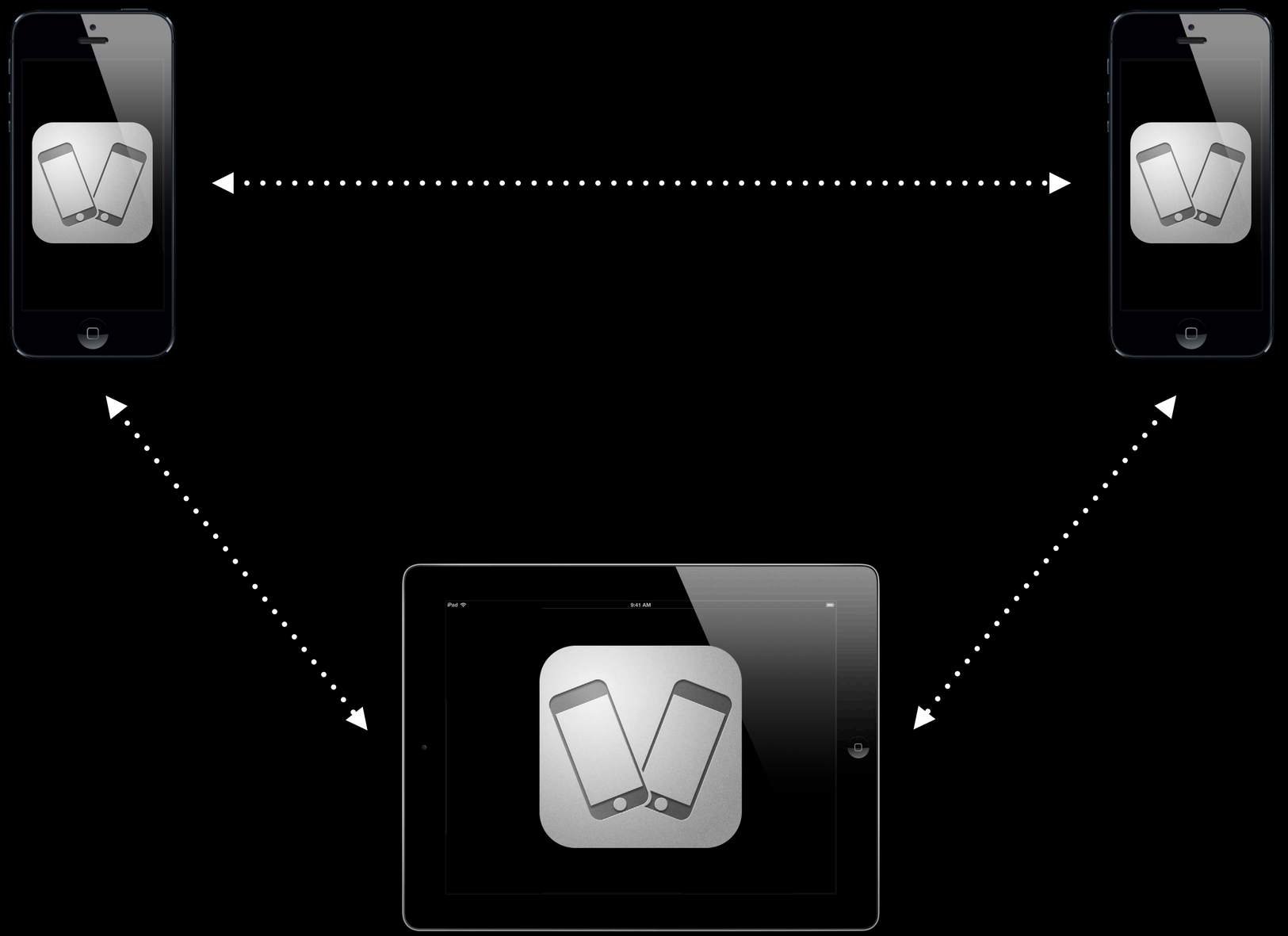
Discovery of and Communication

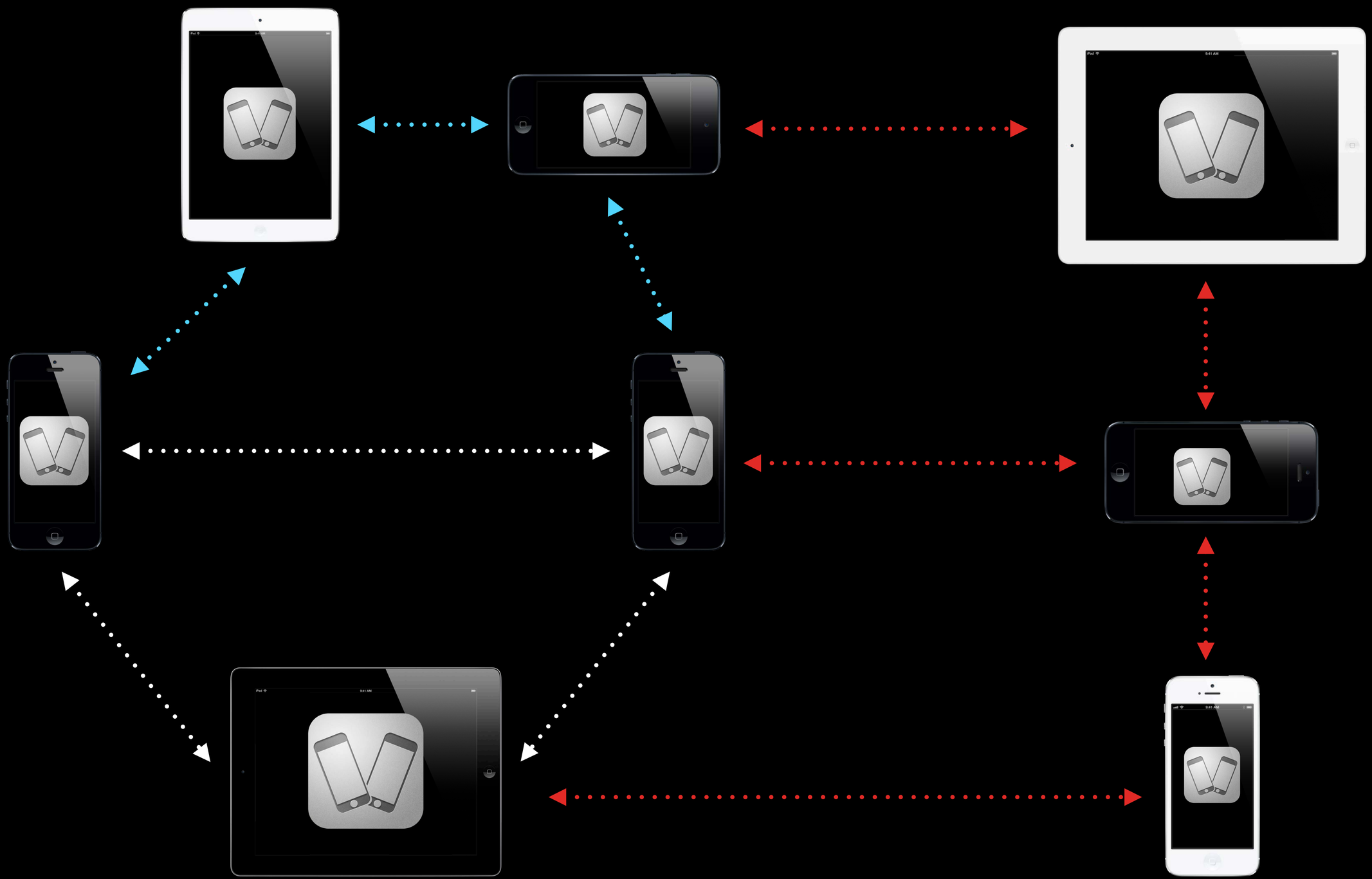
with Nearby Devices


iPad
9:41 AM

Data


Data
iPad
9:41 AM

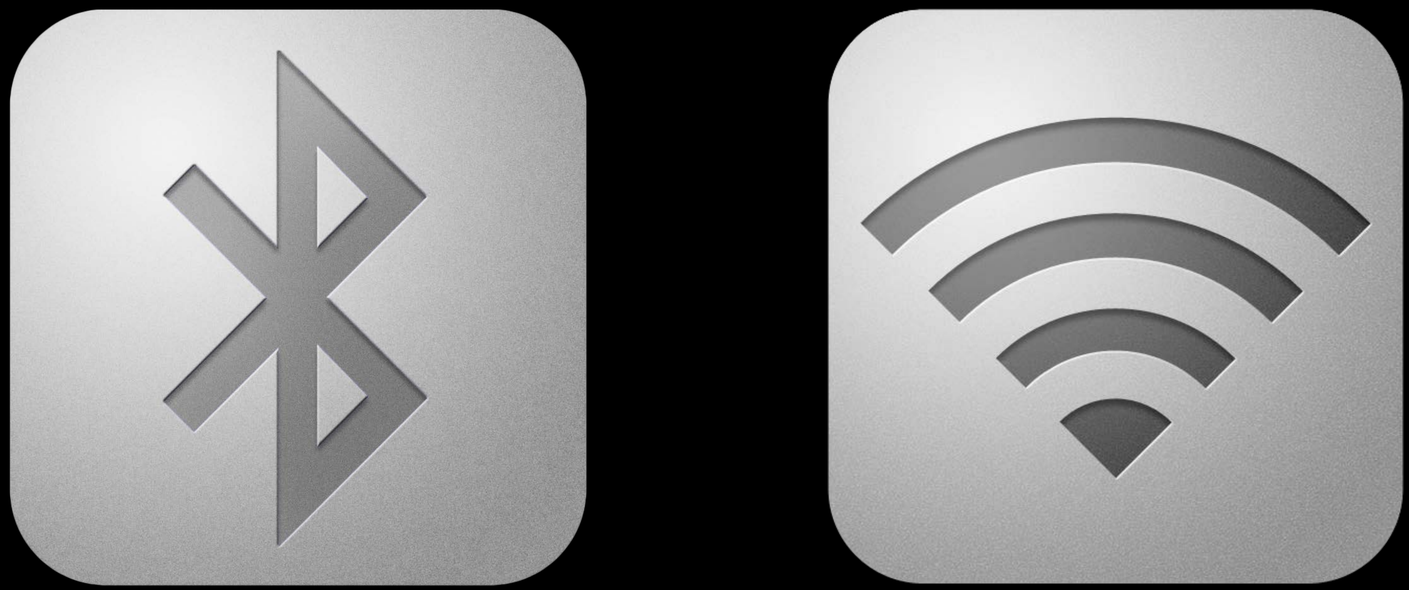

iPad
9:41 AM



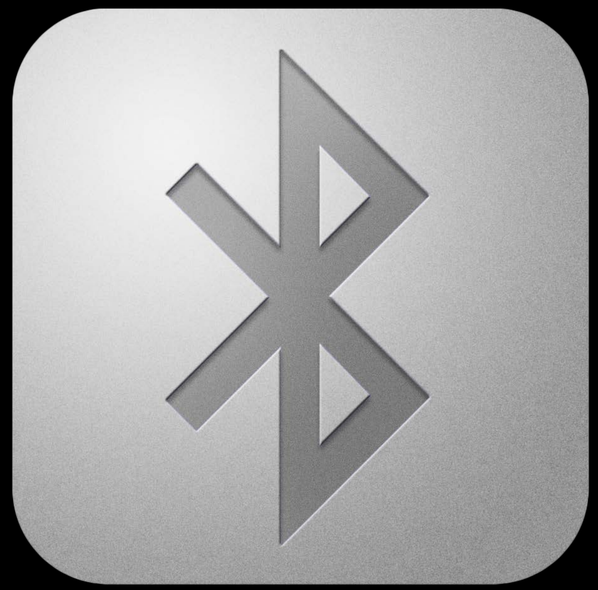
Use Cases

- Interactive tutoring
- Collaborative document/photo editing
- File sharing
- Coordination across multiple devices
- Sensor data aggregation

Wireless Technologies



Wireless Technologies



Features

- Multiple wireless technologies
- Interface selection
- Convenience discovery and invitation UI
- Message-based and stream-based data
- Authentication and encryption

Agenda

- Essentials
  - Discovery phase
  - Session phase
- Advanced
  - Programmatic discovery
  - Security

Essentials

Terminology

Nearby
Within range of supported wireless technologies

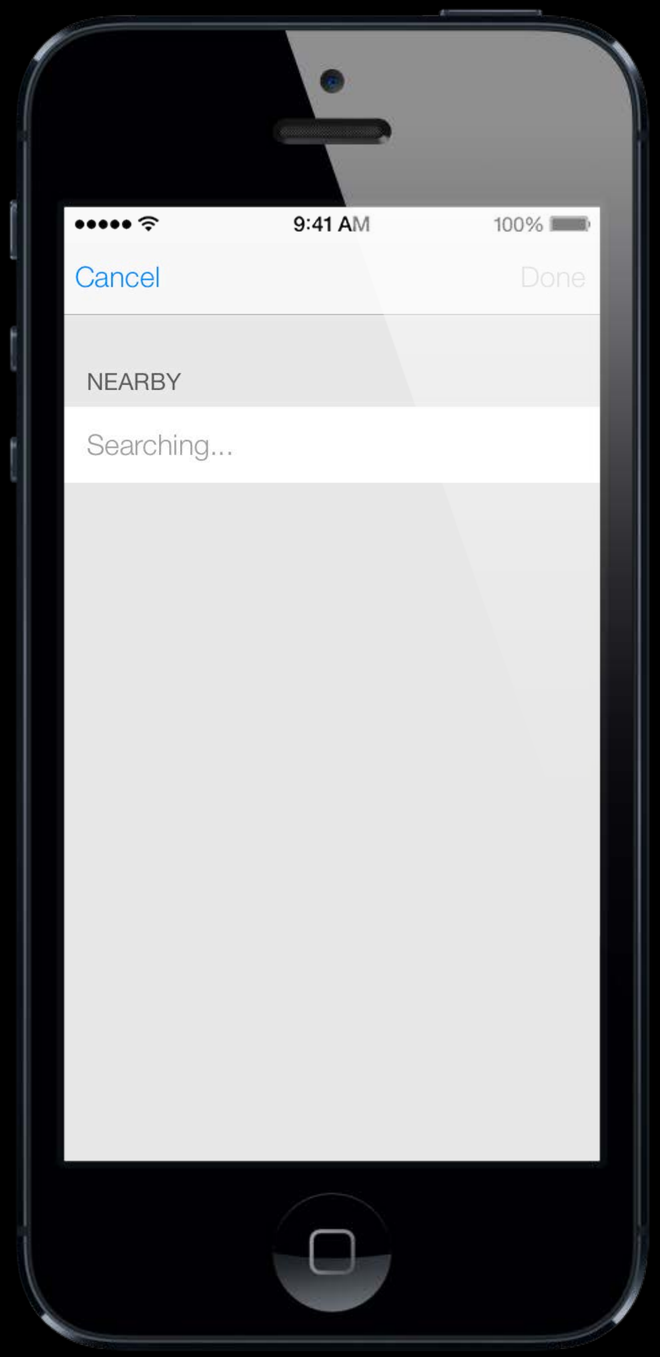
Peer
Nearby device

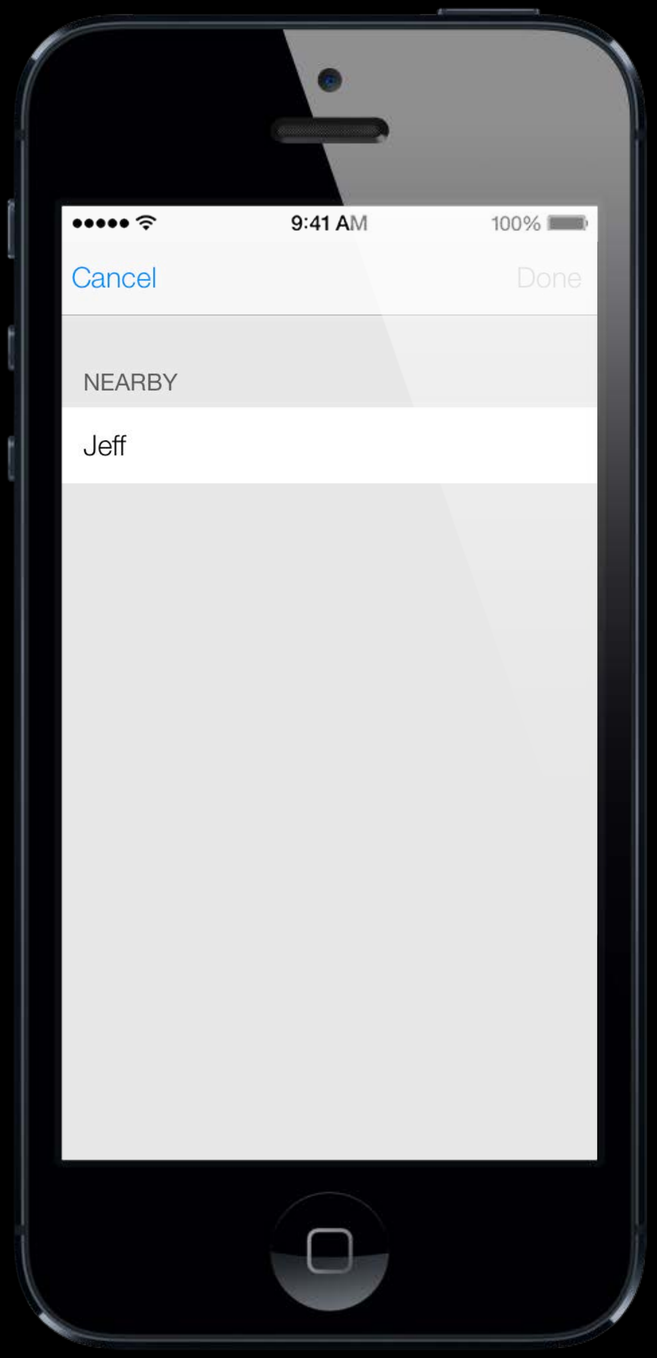
Advertiser
Device discoverable by other nearby devices

Browser
Device searching for other nearby devices

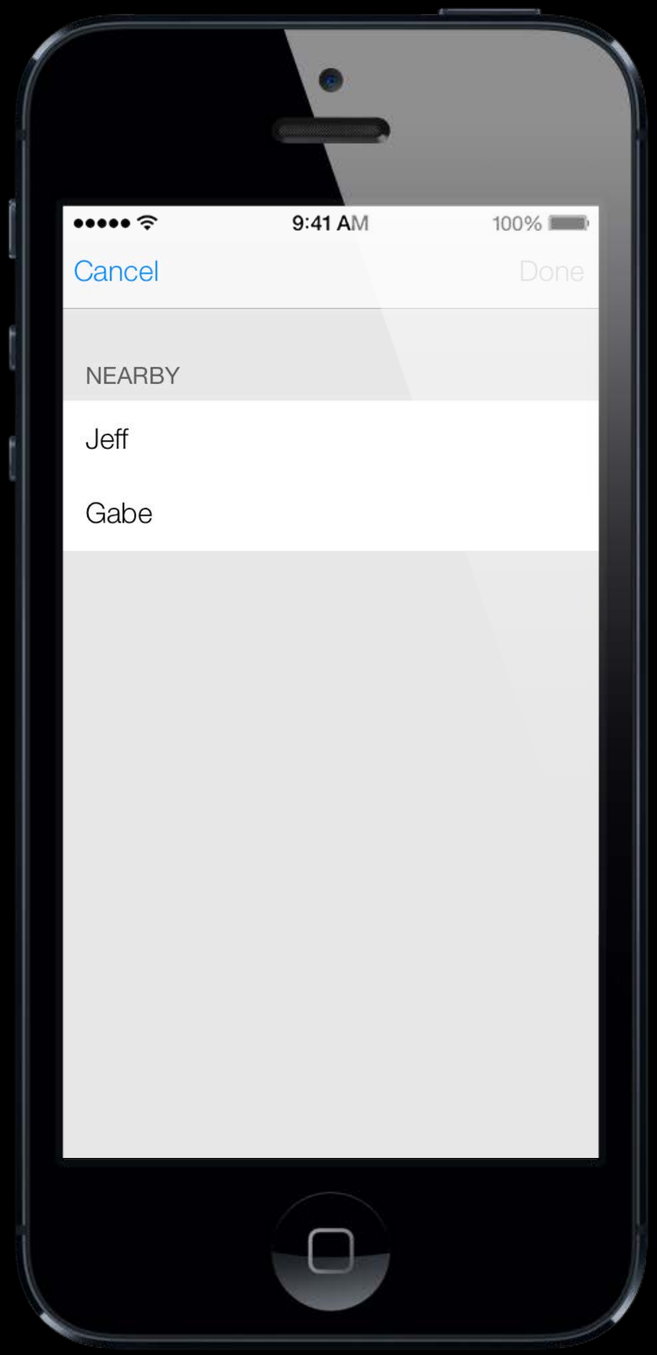
Discovery Phase


Demijan


9:41 AM
100%
Cancel
Done
NEARBY
Searching...

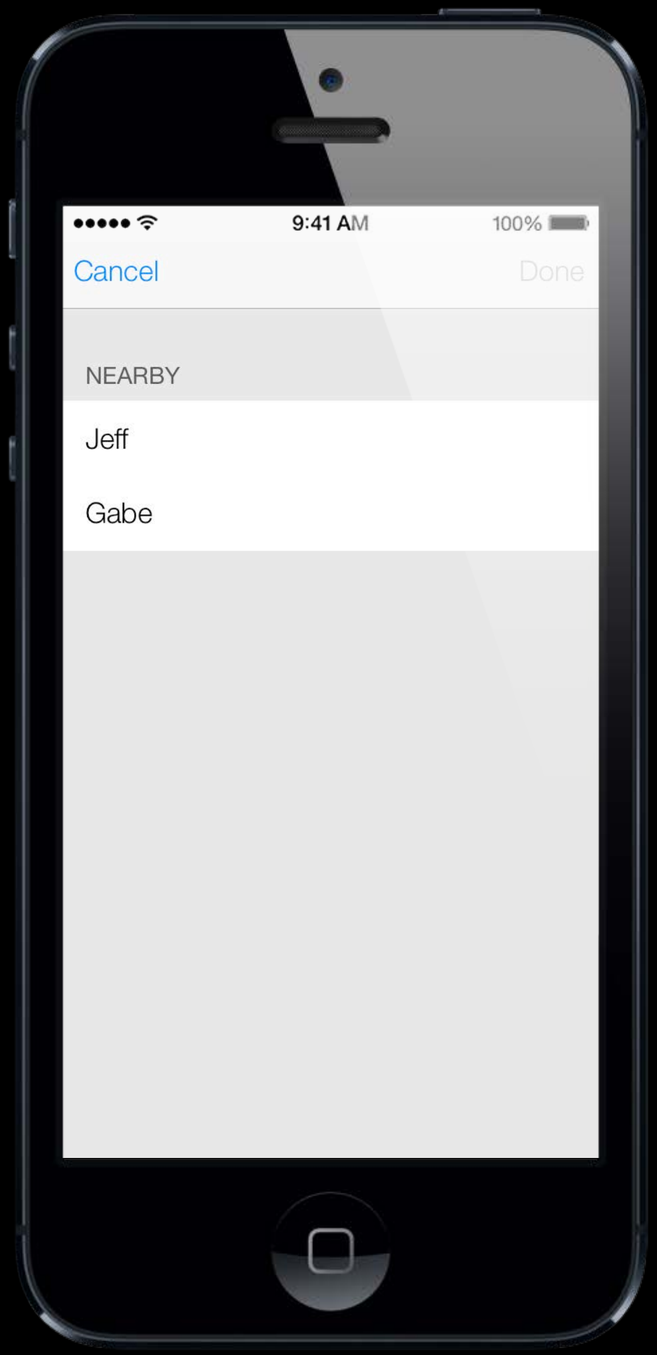

9:41 AM
100%
Cancel
Done
NEARBY
Jeff


Jeff


9:41 AM
100%
Cancel
Done
NEARBY
Jeff
Gabe

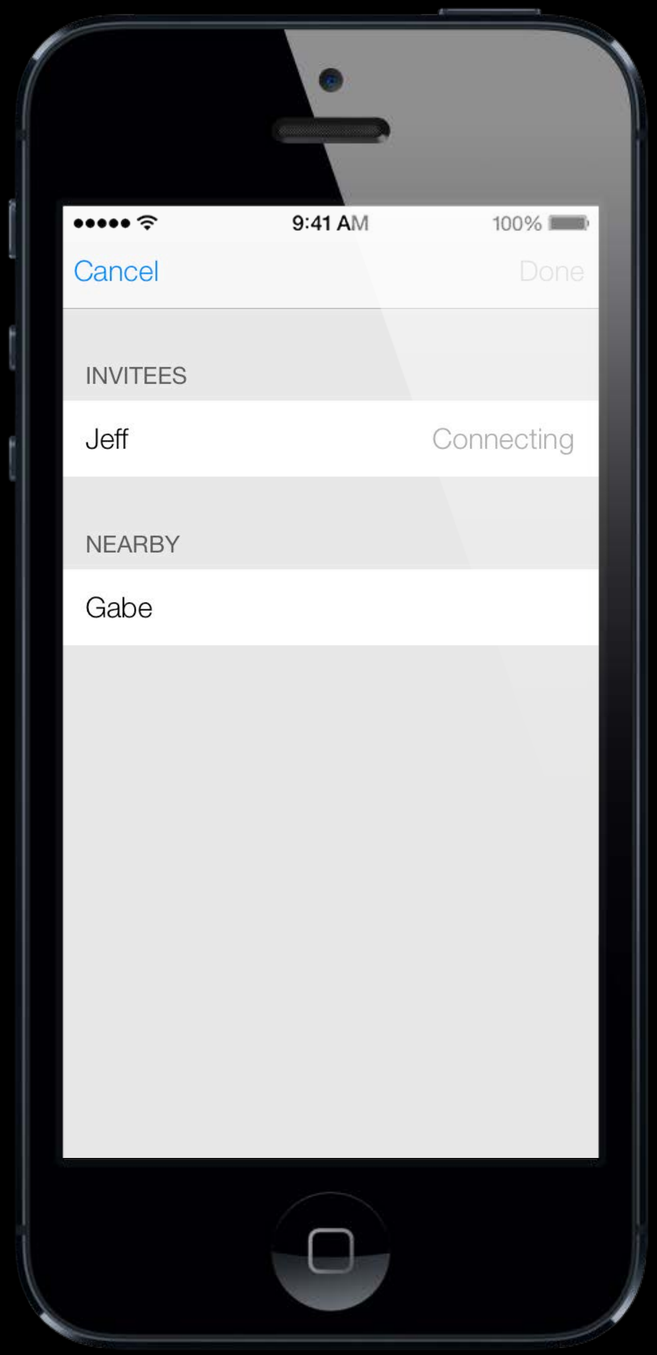

Jeff


Gabe


9:41 AM
100%
Cancel
Done
NEARBY
Jeff
Gabe


Jeff


Gabe

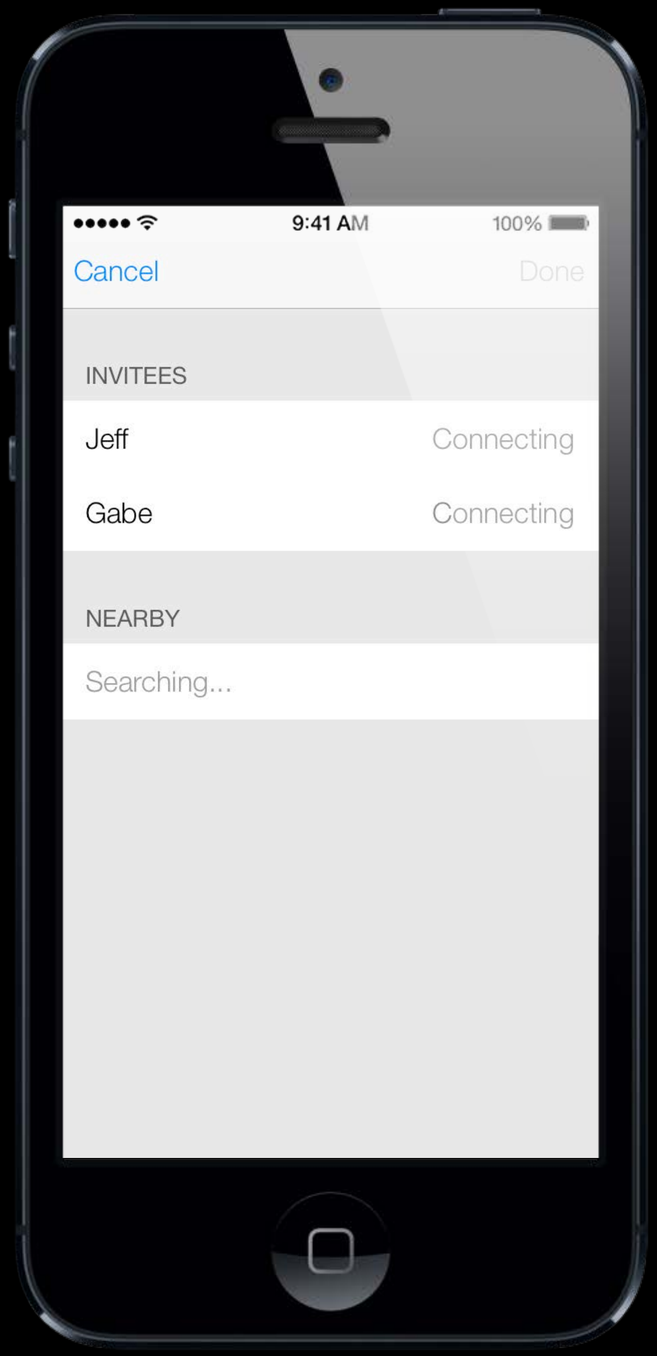

9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
NEARBY
Gabe


Jeff


Gabe


9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
NEARBY
Gabe
Invitation
Jeff
Gabe

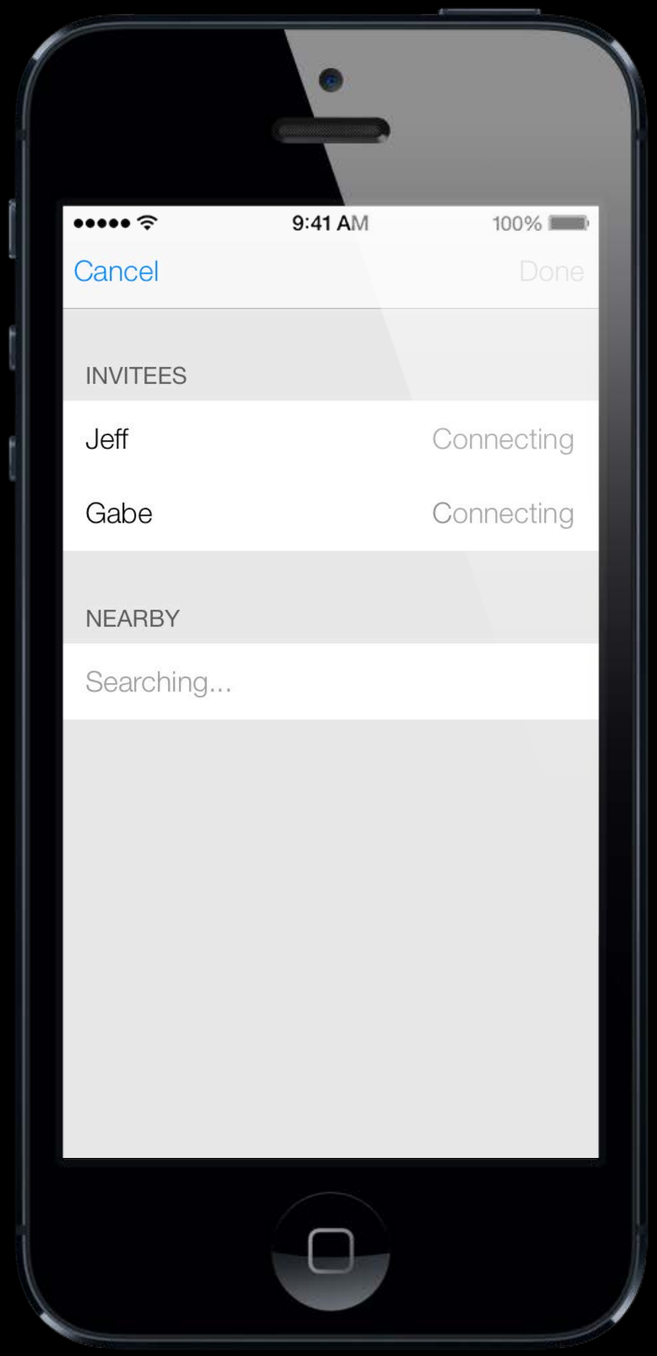

9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
NEARBY
Gabe
Invitation
Jeff
Gabe

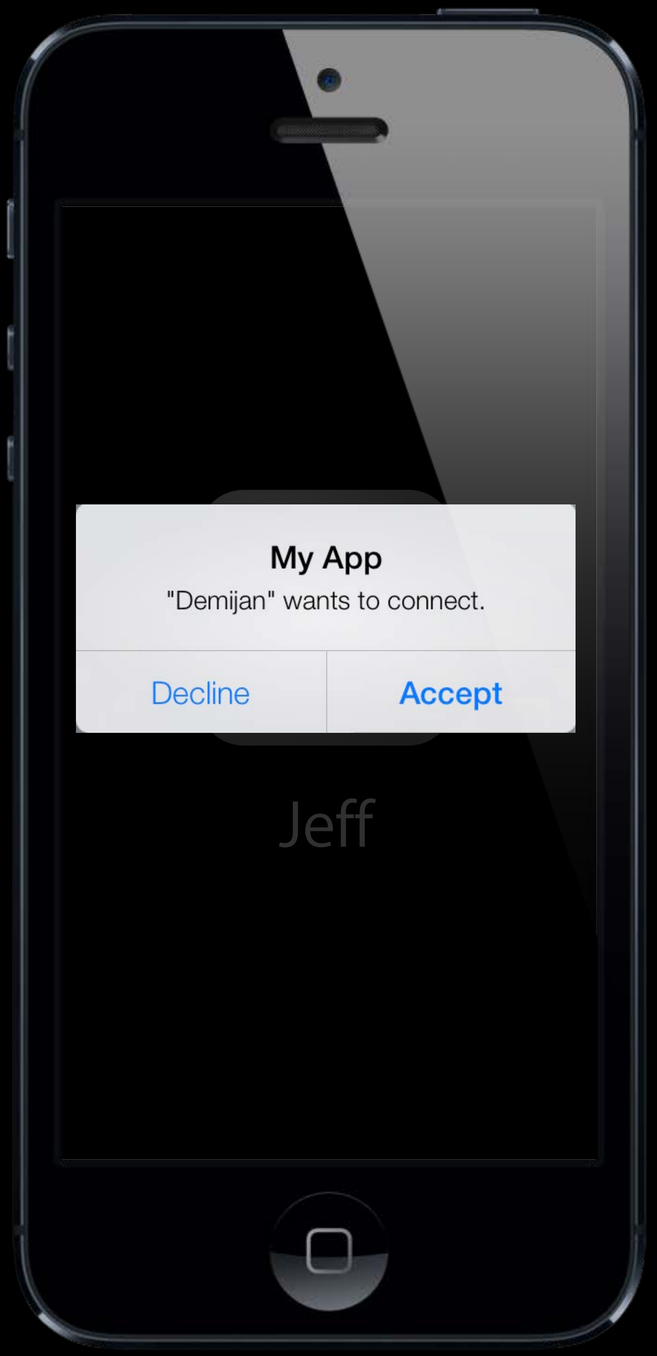

9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
Gabe
Connecting
NEARBY
Searching...


Jeff

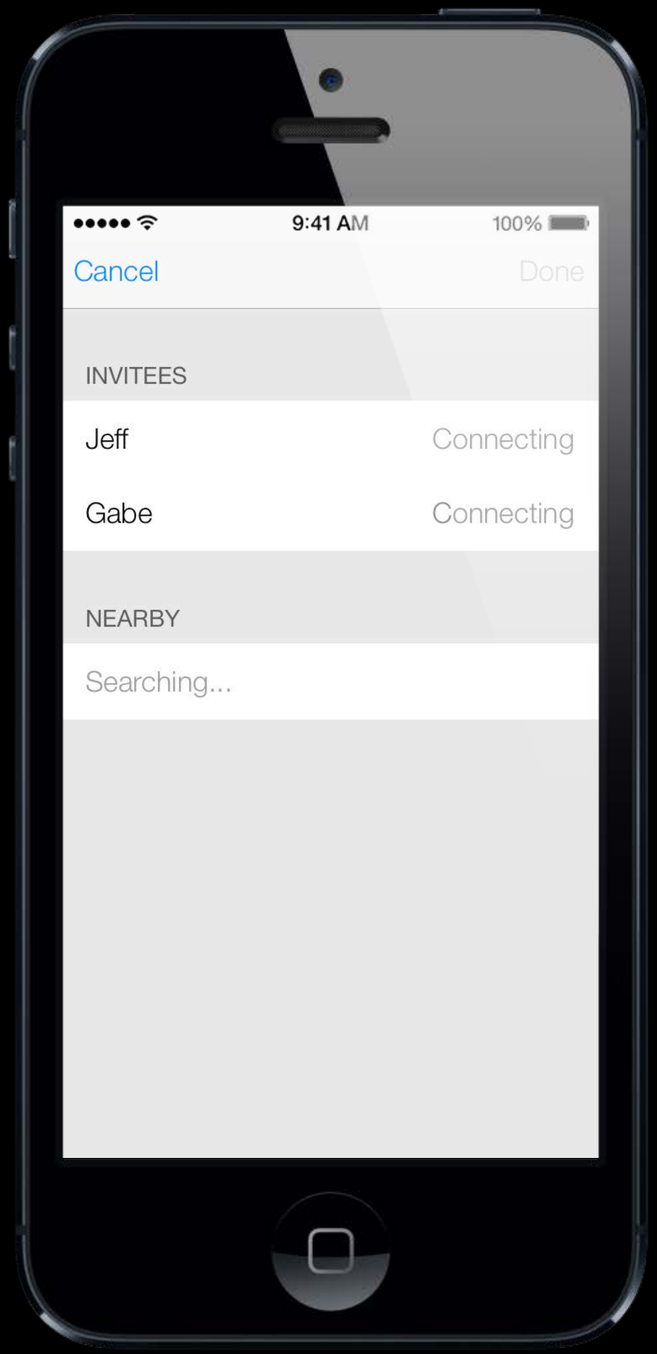

Gabe

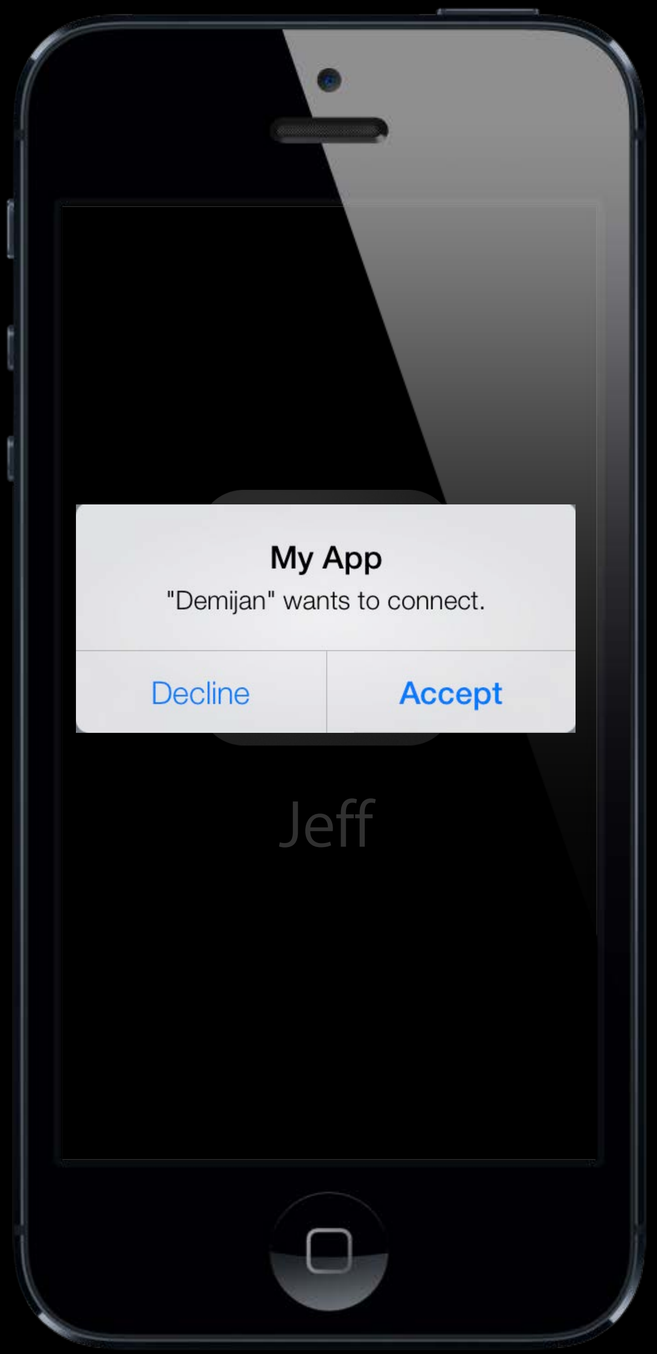

9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
Gabe
Connecting
NEARBY
Searching...
Jeff
Invitation
Gabe

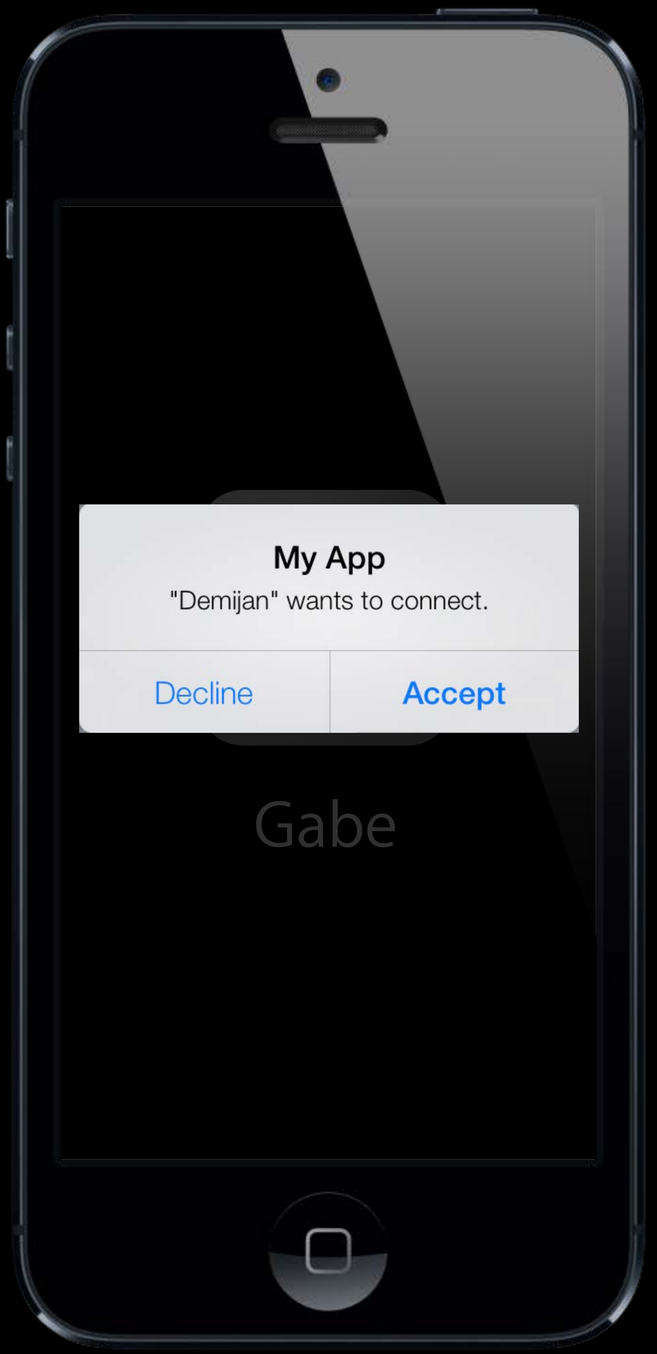

9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
Gabe
Connecting
NEARBY
Searching...

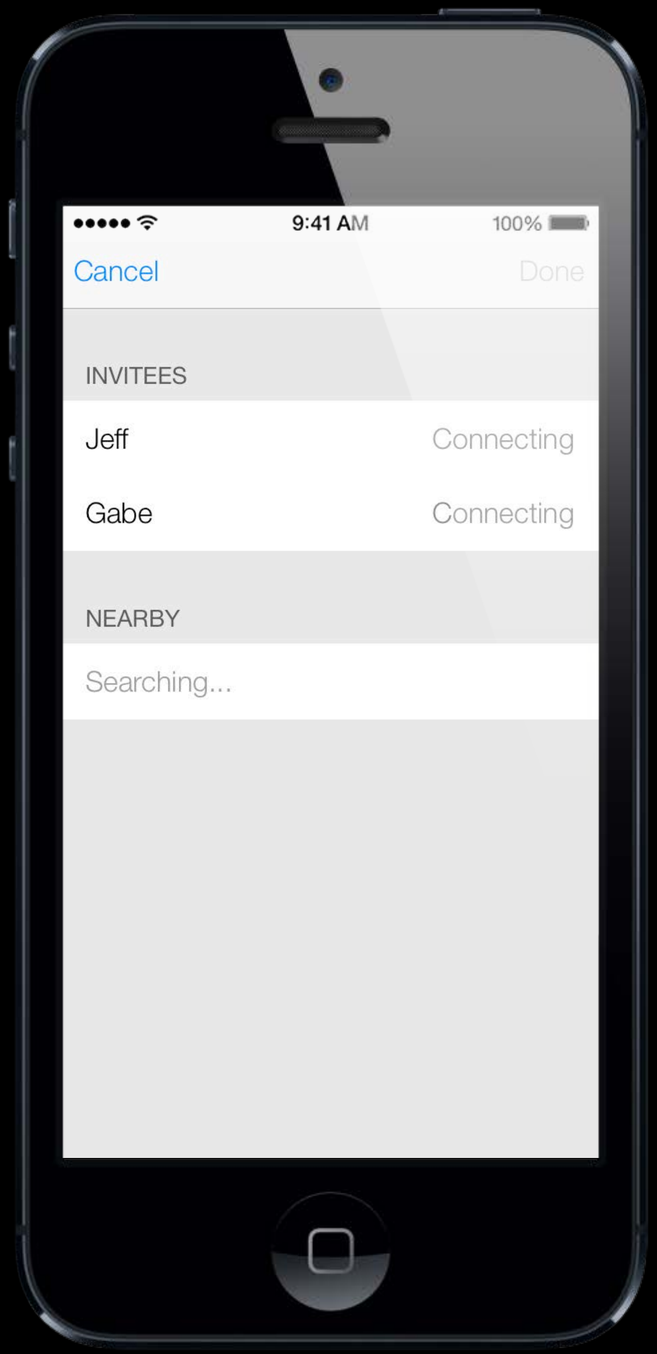

My App
"Demijan" wants to connect.
Decline
Accept
Jeff

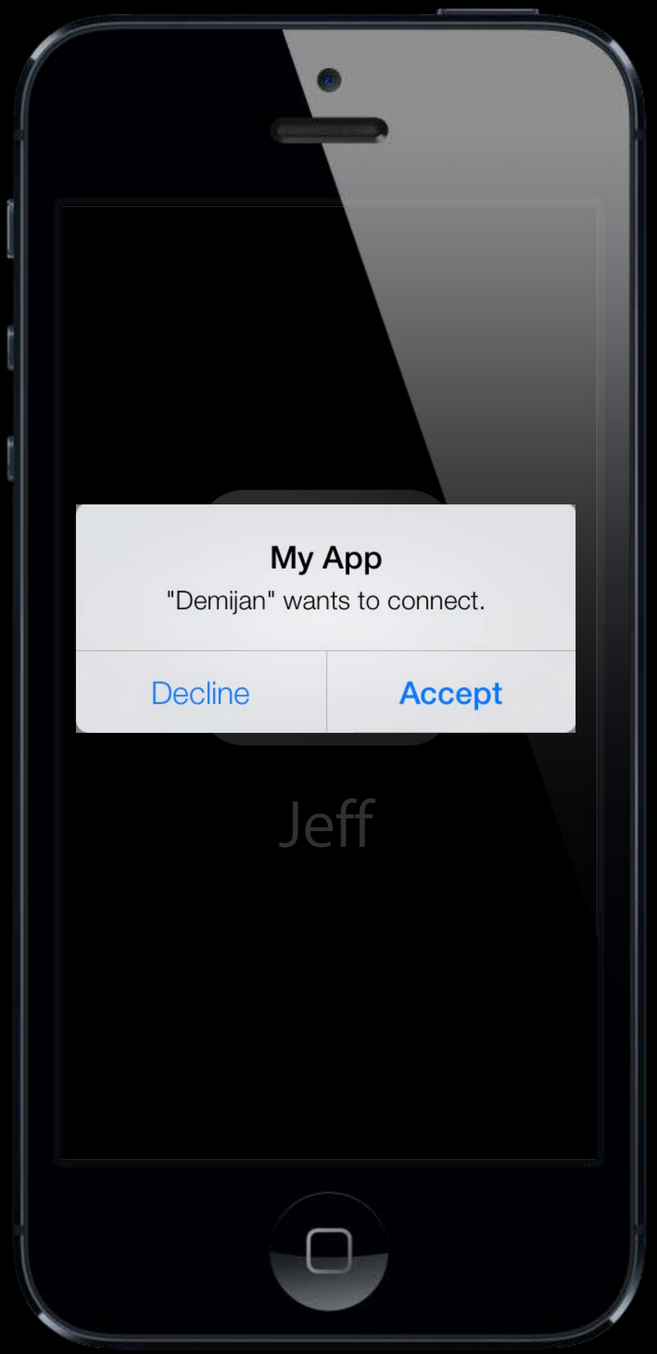

Gabe

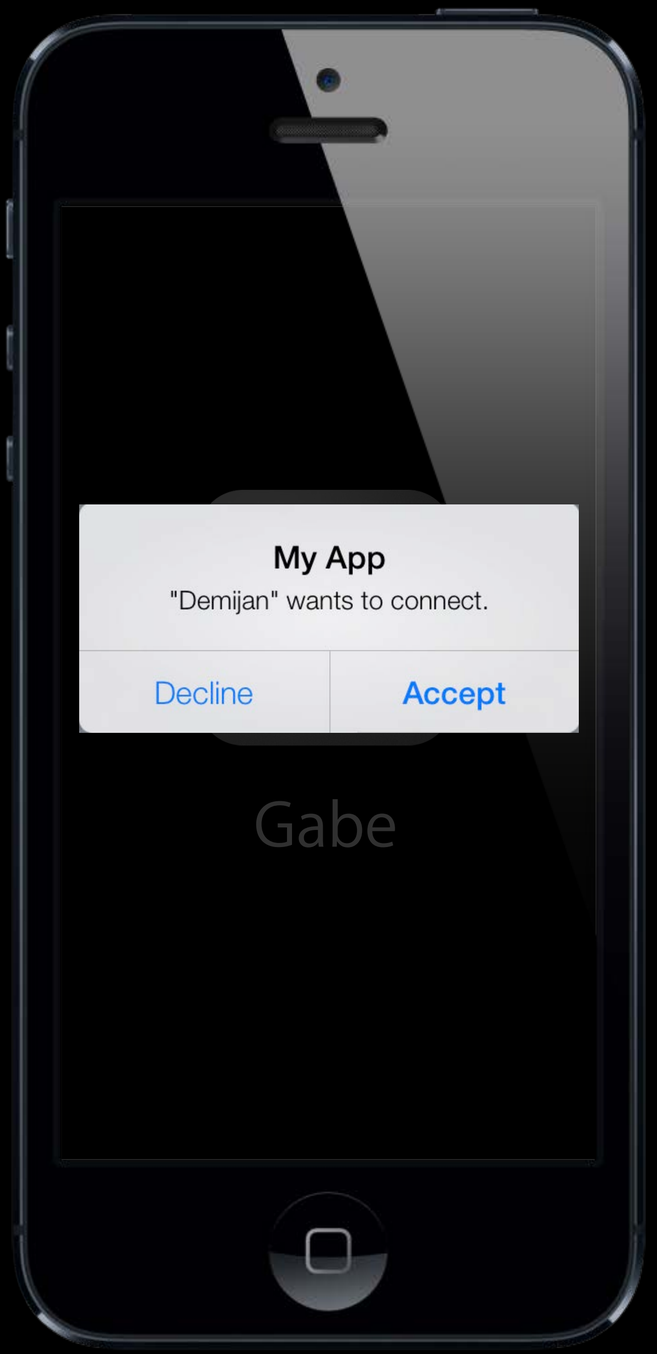

9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
Gabe
Connecting
NEARBY
Searching...

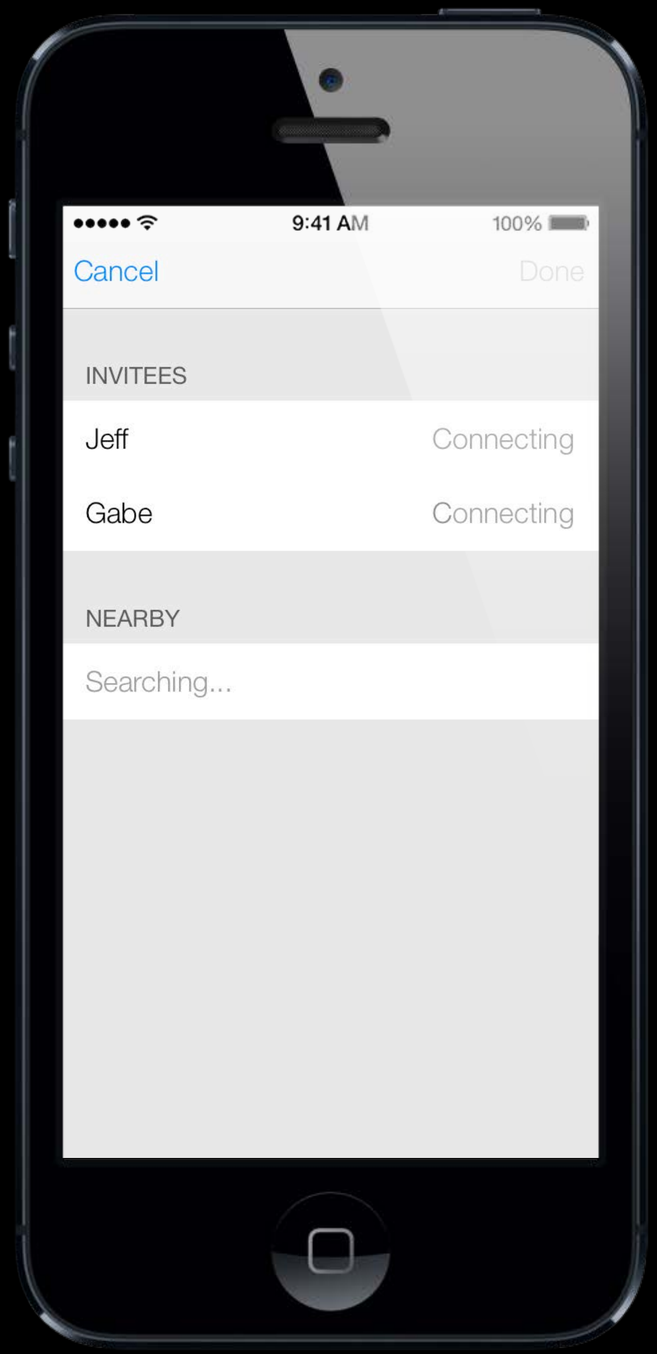

My App
"Demijan" wants to connect.
Decline
Accept
Jeff

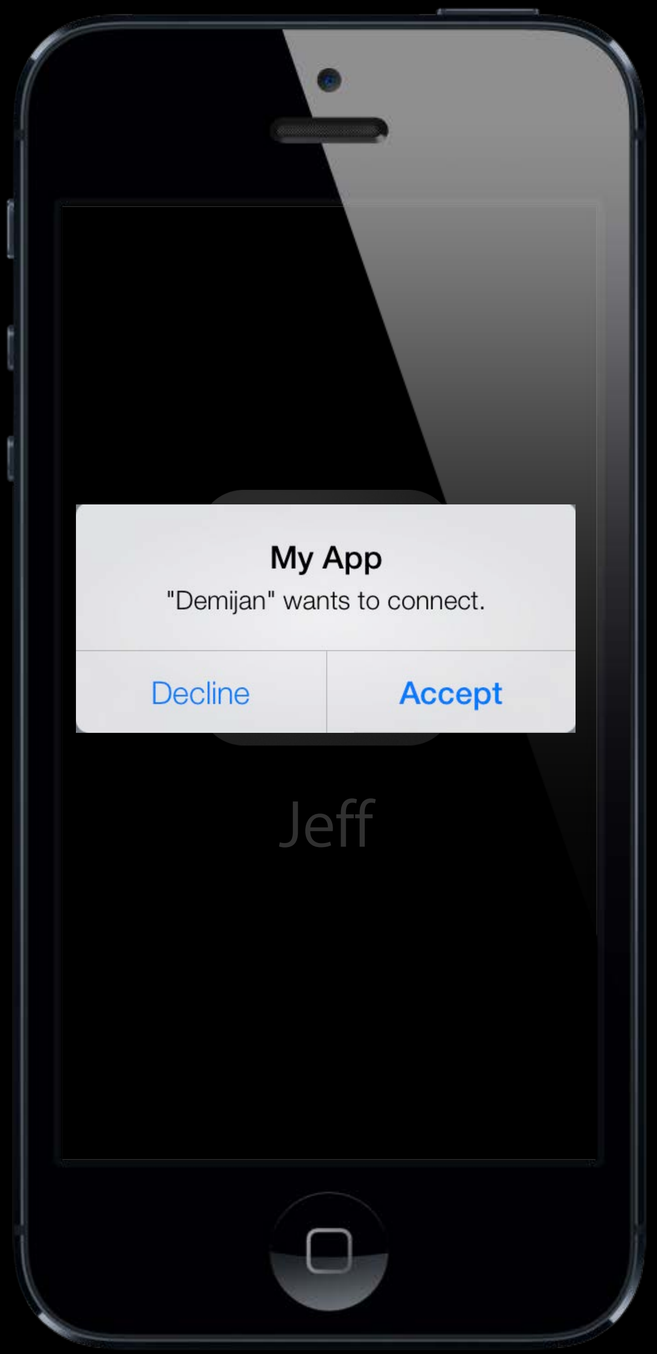

My App
"Demijan" wants to connect.
Decline
Accept
Gabe

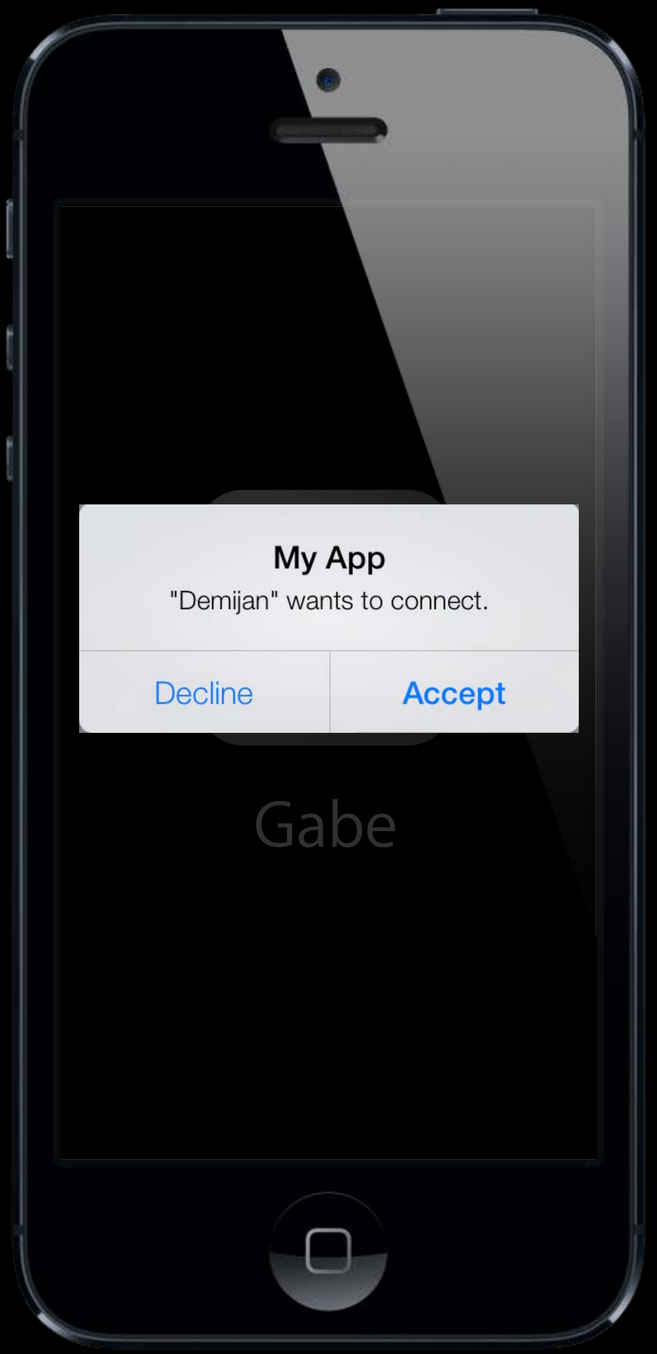

9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
Gabe
Connecting
NEARBY
Searching...

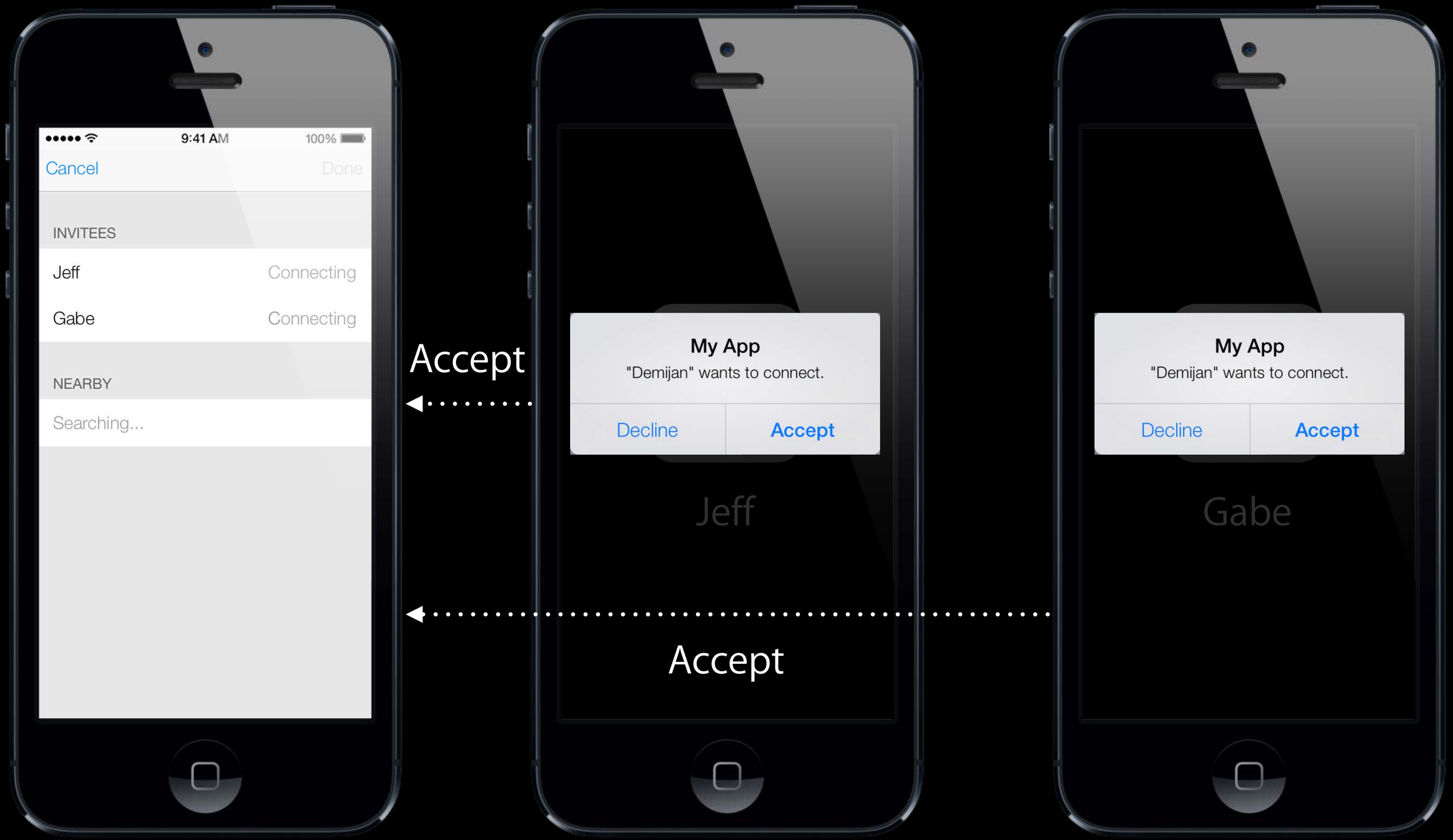

My App
"Demijan" wants to connect.
Decline
Accept
Jeff

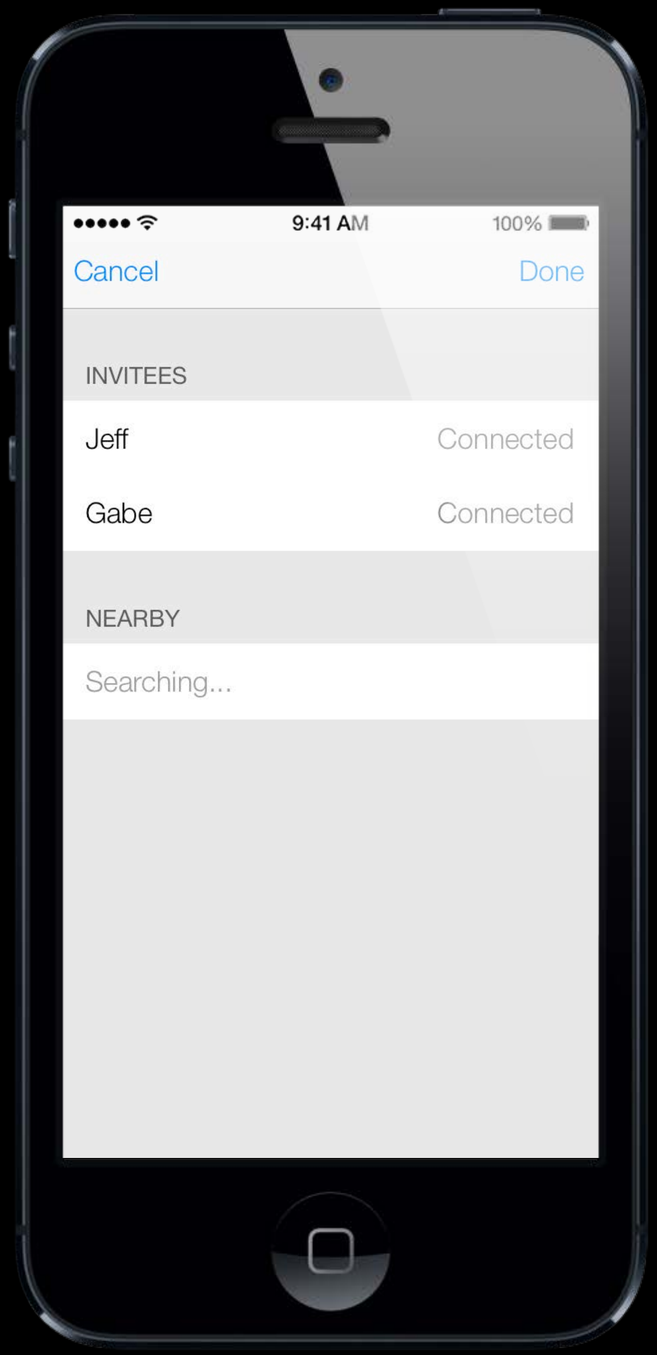

My App
"Demijan" wants to connect.
Decline
Accept
Gabe


9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
Gabe
Connecting
NEARBY
Searching...


My App
"Demijan" wants to connect.
Decline
Accept
Jeff


My App
"Demijan" wants to connect.
Decline
Accept
Gabe


9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connecting
Gabe
Connecting
NEARBY
Searching...
Accept
My App
"Demijan" wants to connect.
Decline
Accept
Jeff
My App
"Demijan" wants to connect.
Decline
Accept
Gabe
Accept


9:41 AM
100%
Cancel
Done
INVITEES
Jeff
Connected
Gabe
Connected
NEARBY
Searching...


Jeff


Gabe

Demijan

Jeff

Gabe

Demijan

Jeff

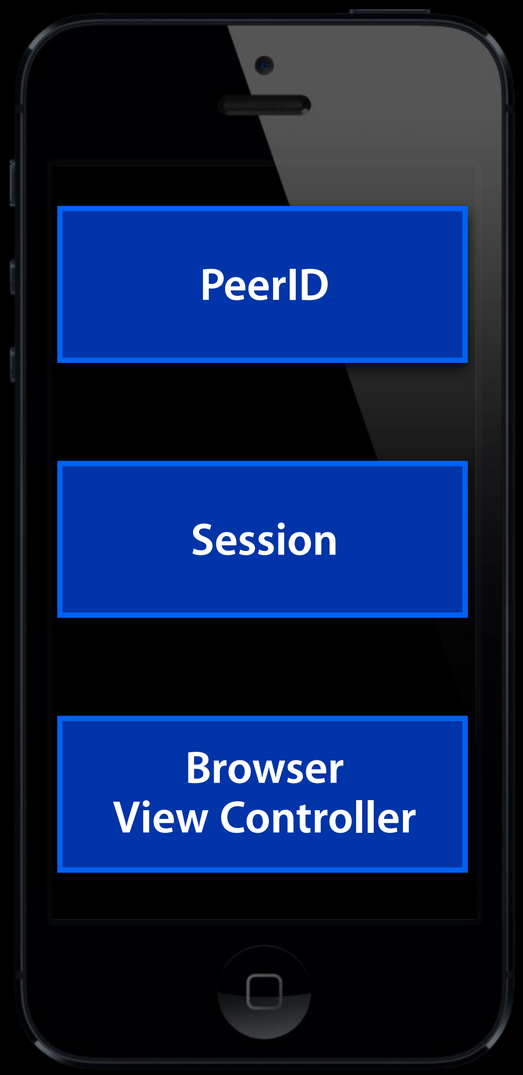
Gabe

Jeff

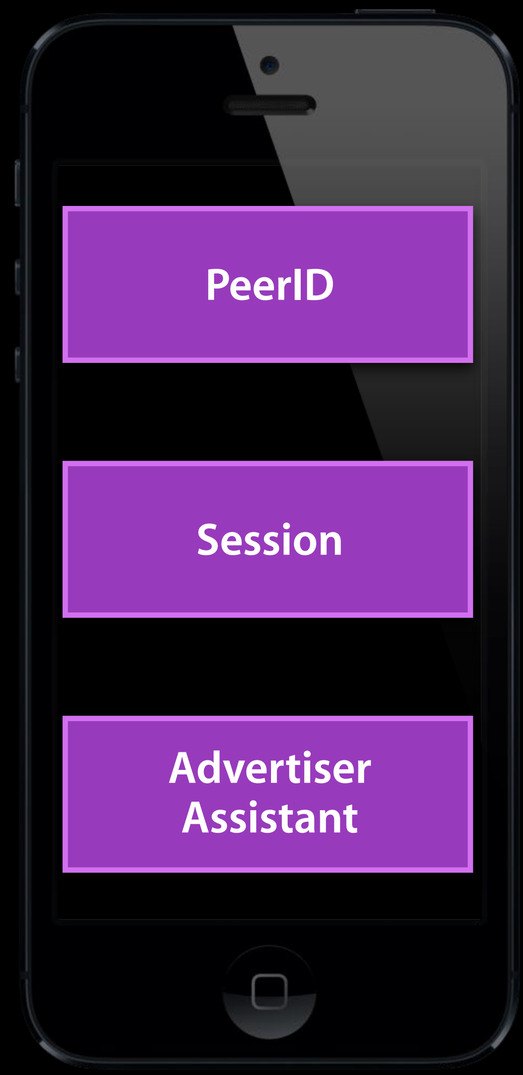
Data

Demijan

Gabe

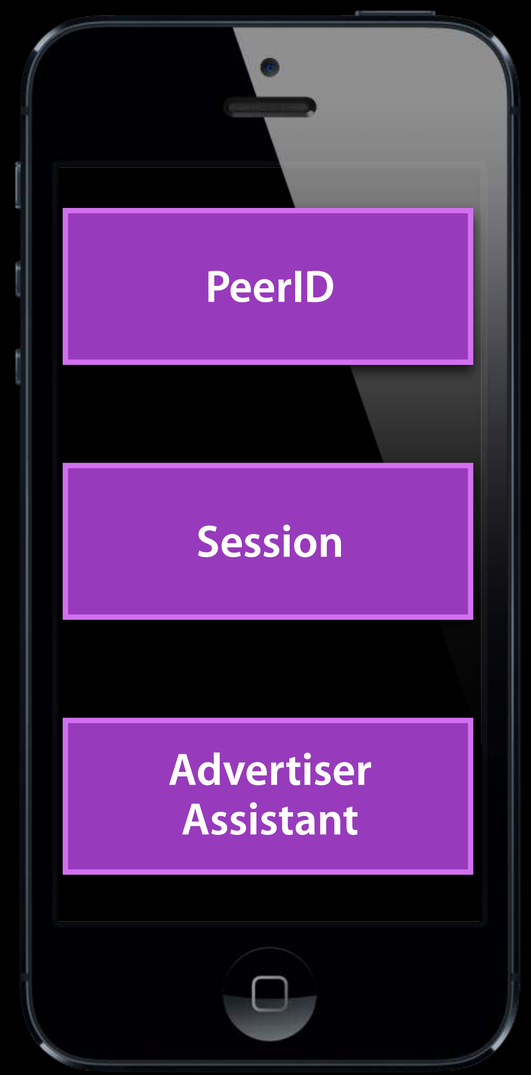


Browser



Advertiser

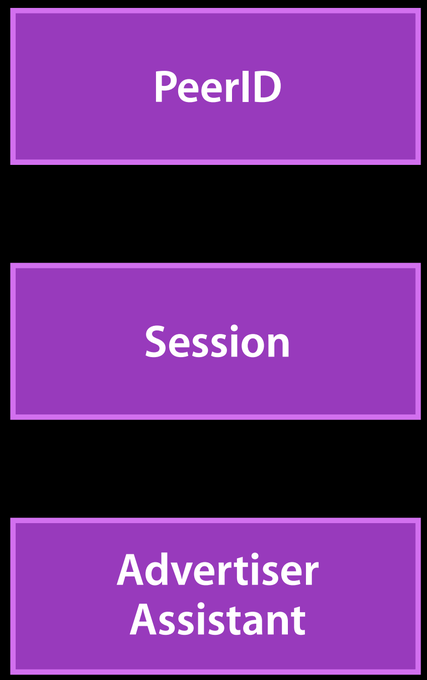
Advertiser Setup



Advertiser

Tasks

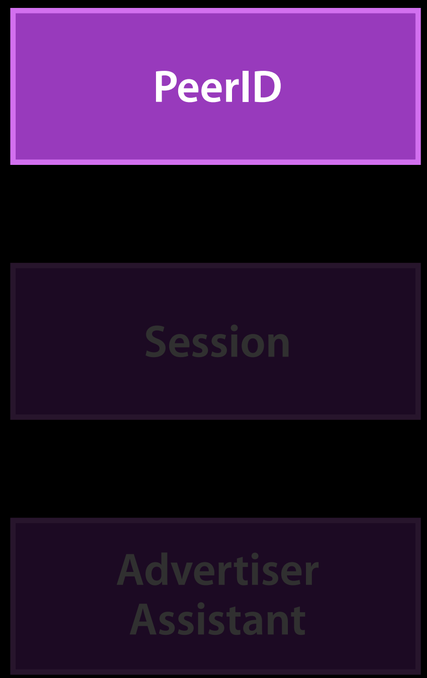
- Make device discoverable
- Present invitations to the user
- Handle user response
- Connect peer to session



MCPeerID

Identify yourself

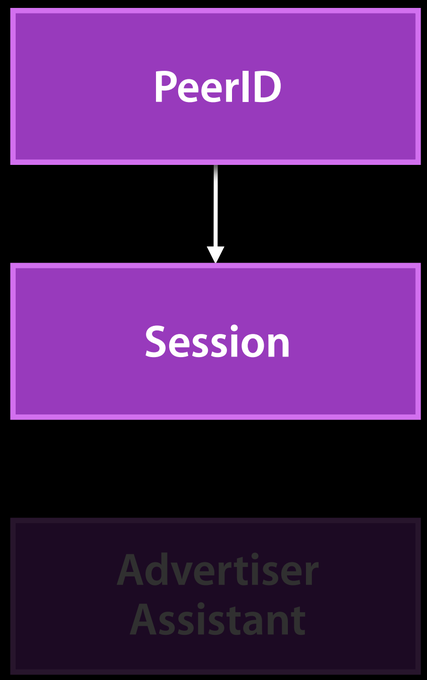
```
// initialize local peer
MCPeerID *myPeerID = [[MCPeerID alloc]
                              initWithName:@”Jeff”];
```



MCSession

Define a session

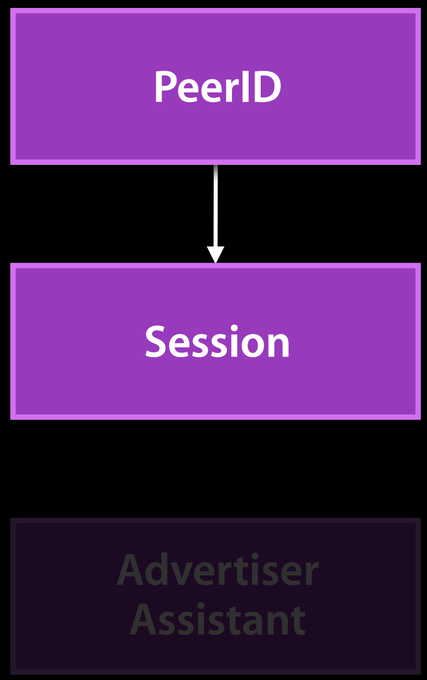
```
// initialize
MCSession *session = [[MCSession alloc]
                              initWithPeer:myPeerID];

// set the delegate
session.delegate = self;
```



MCSession

Define a session

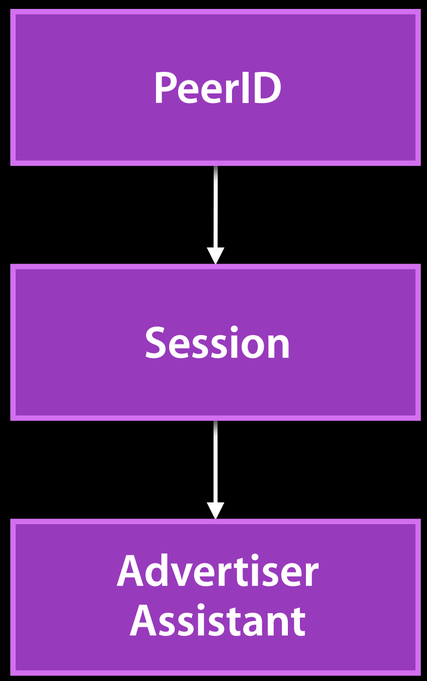
```
// initialize
MCSession *session = [[MCSession alloc]
                              initWithPeer:myPeerID];

// set the delegate
session.delegate = self;
```



MCAdvertiserAssistant

Setup

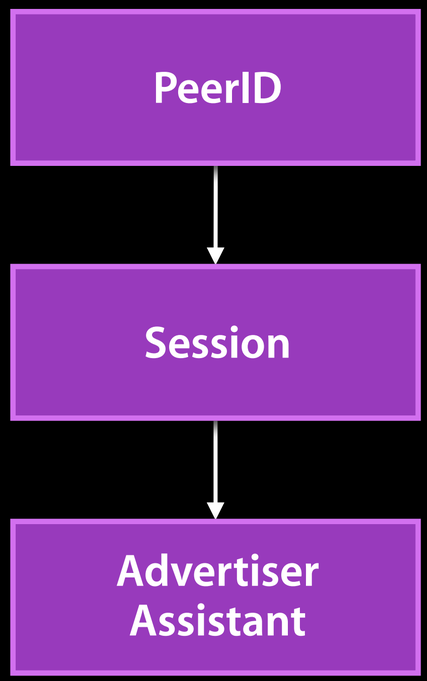
```
// initialize
MCAdvertiserAssistant *assistant =
     [[MCAdvertiserAssistant alloc]
                initWithServiceType:type
                      discoveryInfo:nil
                            session:session];


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

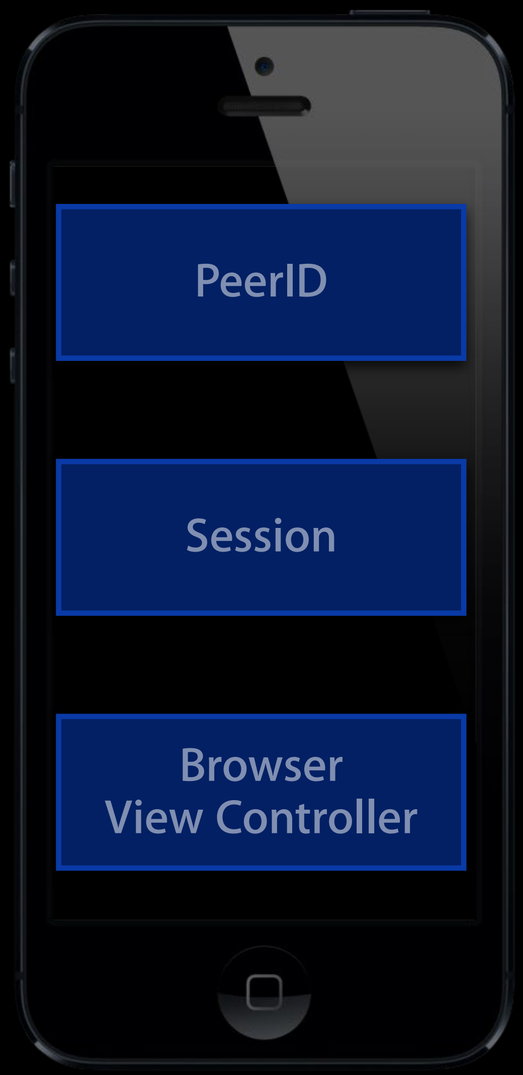


MCAdvertiserAssistant

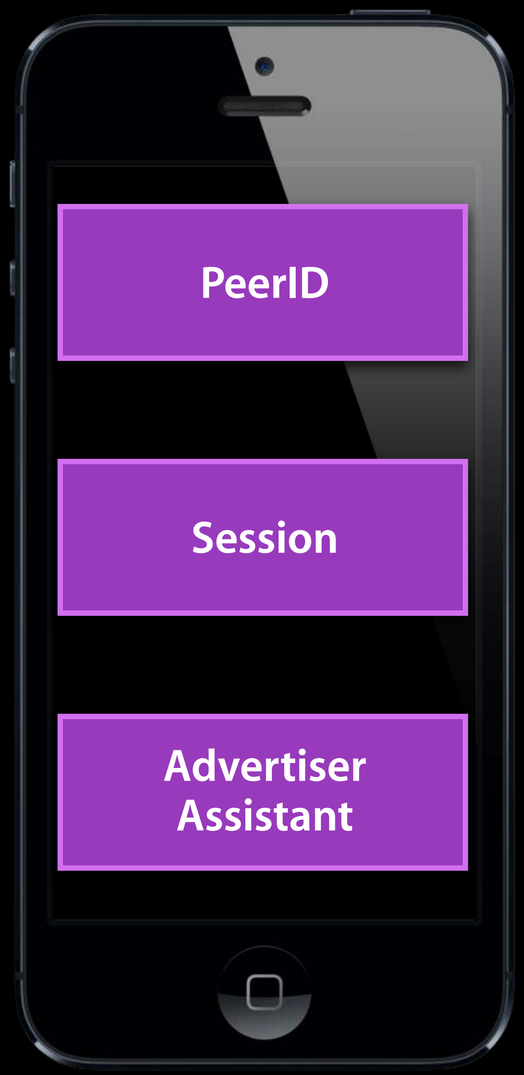
Setup

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


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



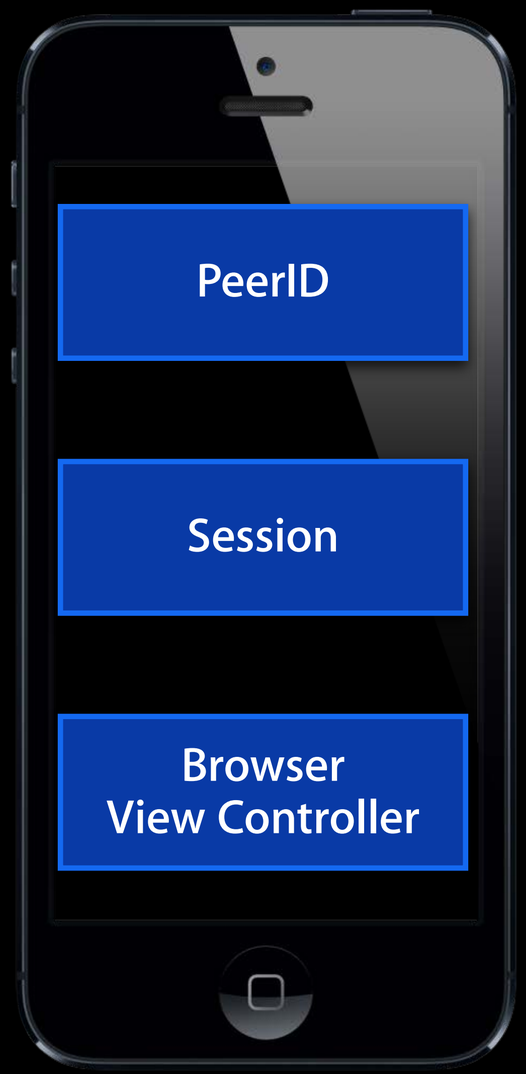


Browser



Advertiser

Browser Setup



Browser

Tasks

- Present nearby peers
- Send invites
- Handle invite responses
- Connect peer to session

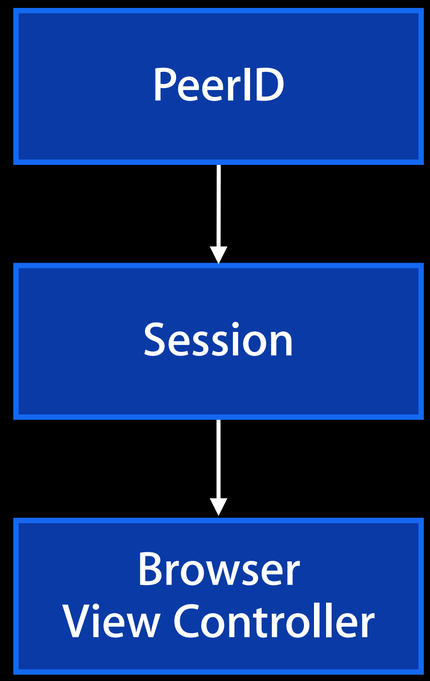
PeerID

Session

Browser
View Controller

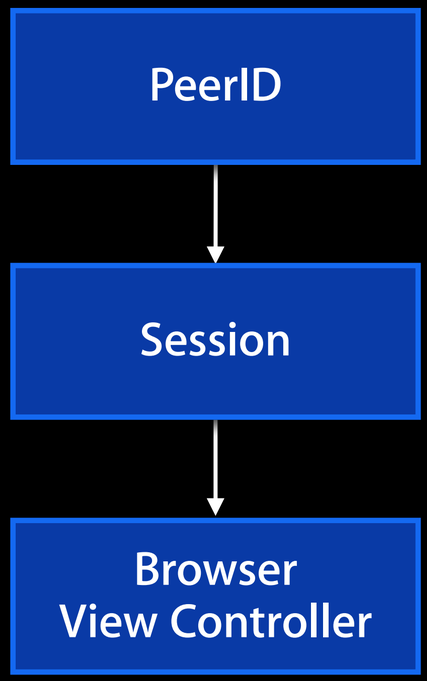
MCBrowserViewController

```
// initialize
MCBrowserViewController *browserVC =
  [[MCBrowserViewController] alloc]
            initWithServiceType:serviceType
                        session:session];

// set delegate, present
browserVC.delegate = self;
[self presentViewController:browserVC
                   animated:YES
                 completion:nil];
```



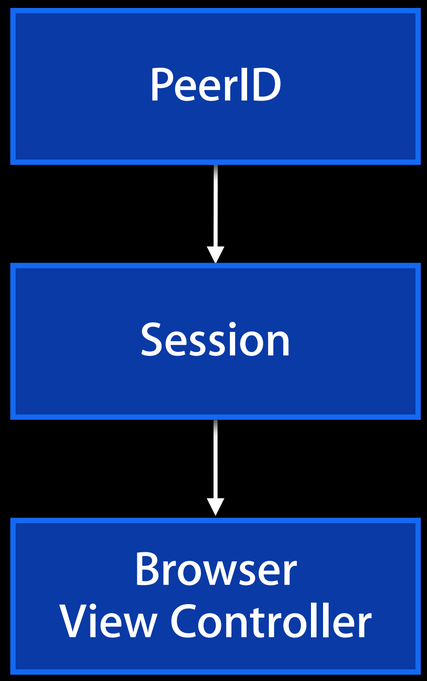
MCBrowserViewController

```
// initialize
MCBrowserViewController *browserVC =
  [[MCBrowserViewController] alloc]
            initWithServiceType:serviceType
                        session:session];

// set delegate, present
browserVC.delegate = self;
[self presentViewController:browserVC
                   animated:YES
                 completion:nil];
```



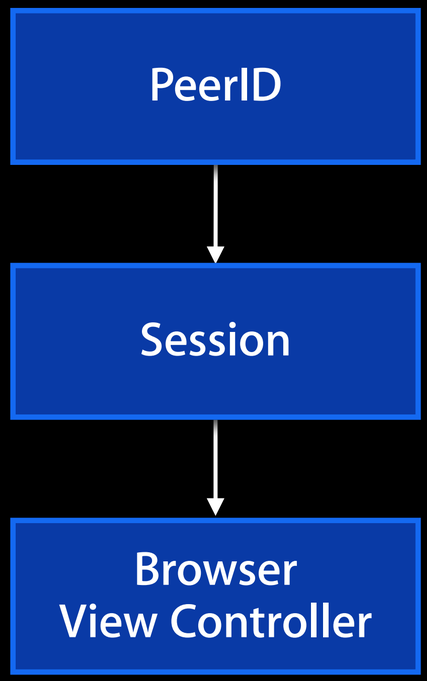
MCBrowserViewController

```
// initialize
MCBrowserViewController *browserVC =
  [[MCBrowserViewController] alloc]
            initWithServiceType:serviceType
                        session:session];

// set delegate, present
browserVC.delegate = self;
[self presentViewController:browserVC
                   animated:YES
                 completion:nil];
```

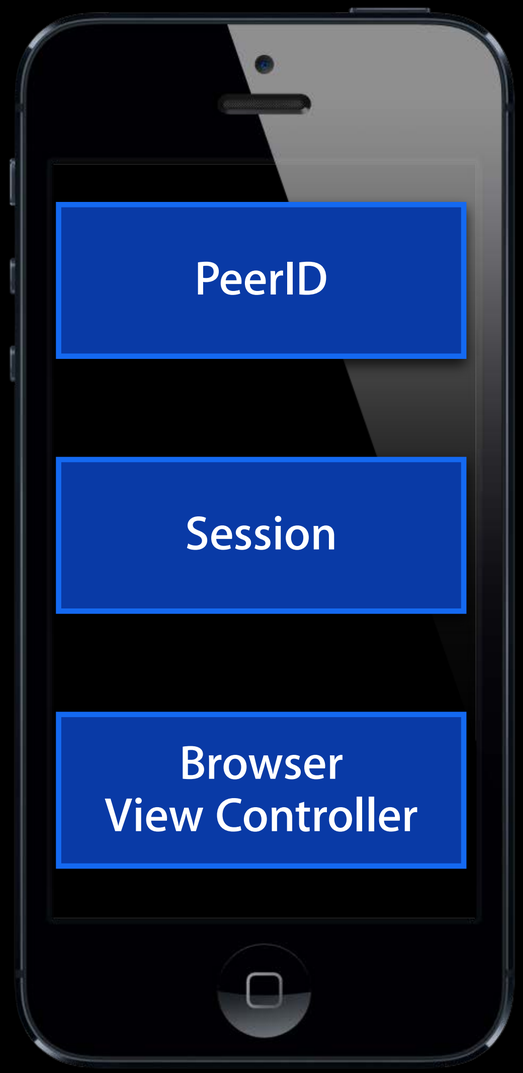


MCBrowserViewControllerDelegate

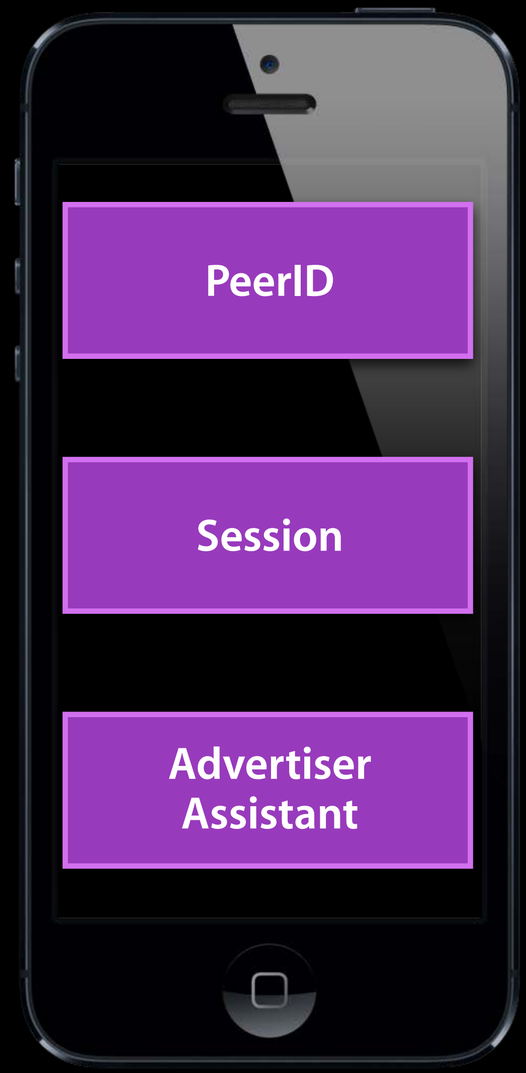
```
// done button tapped
- (void)browserViewControllerDidFinish:
    (MCBrowserViewController *)browserVC
{
    [browserVC dismissViewControllerAnimated:YES
                                  completion:nil];
}
```



Discovery Phase Class Overview

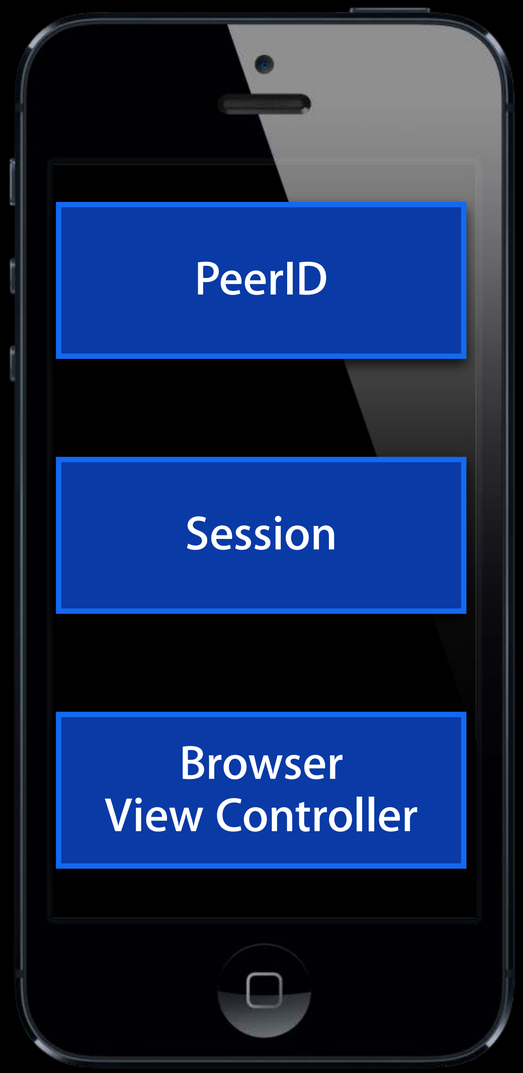


Browser

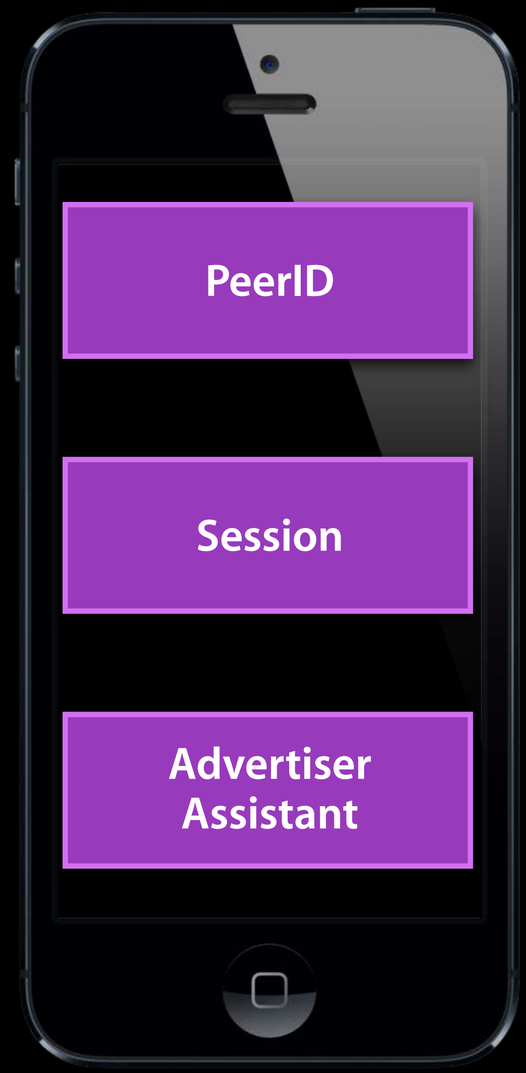


Advertiser

Discovery Phase Class Overview

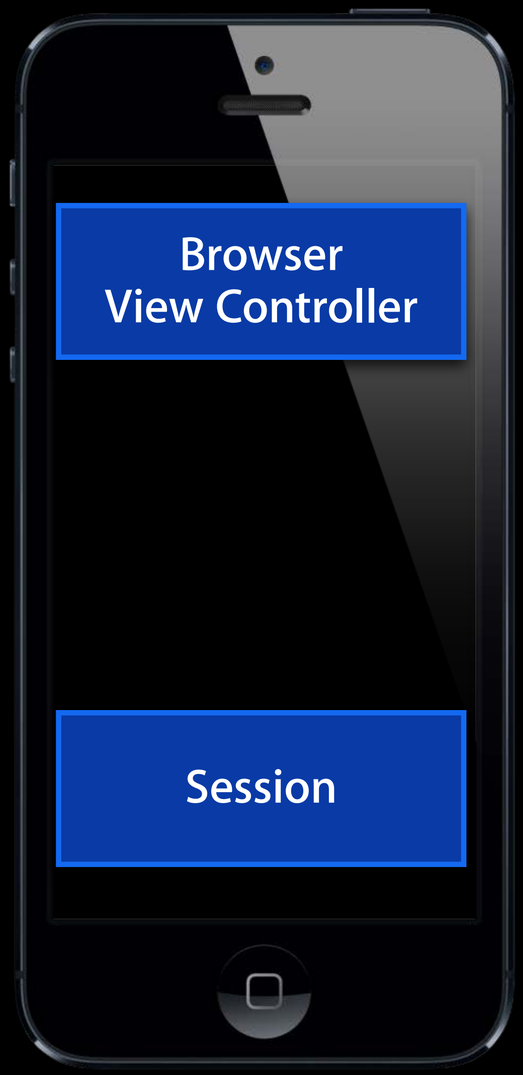


Browser

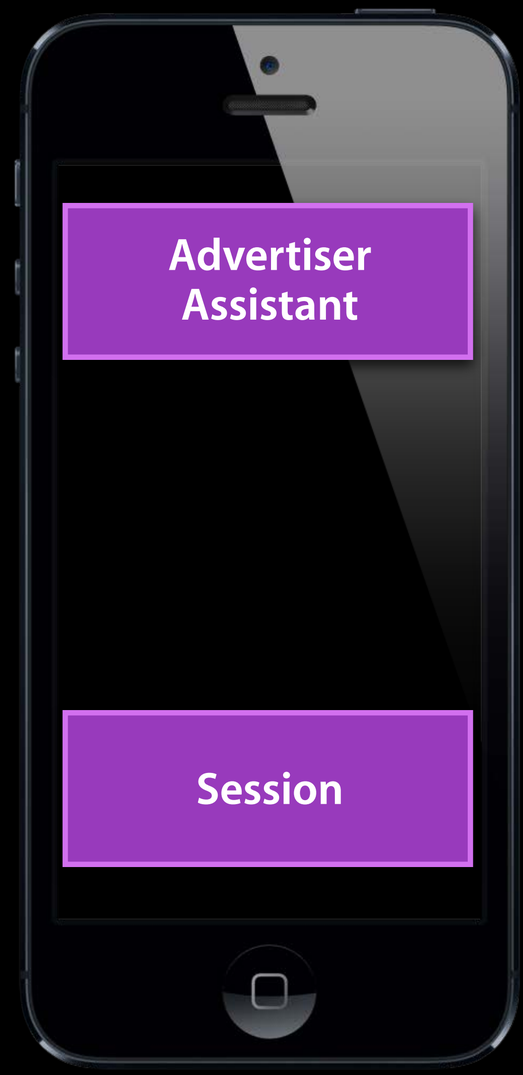


Advertiser

Getting Connected

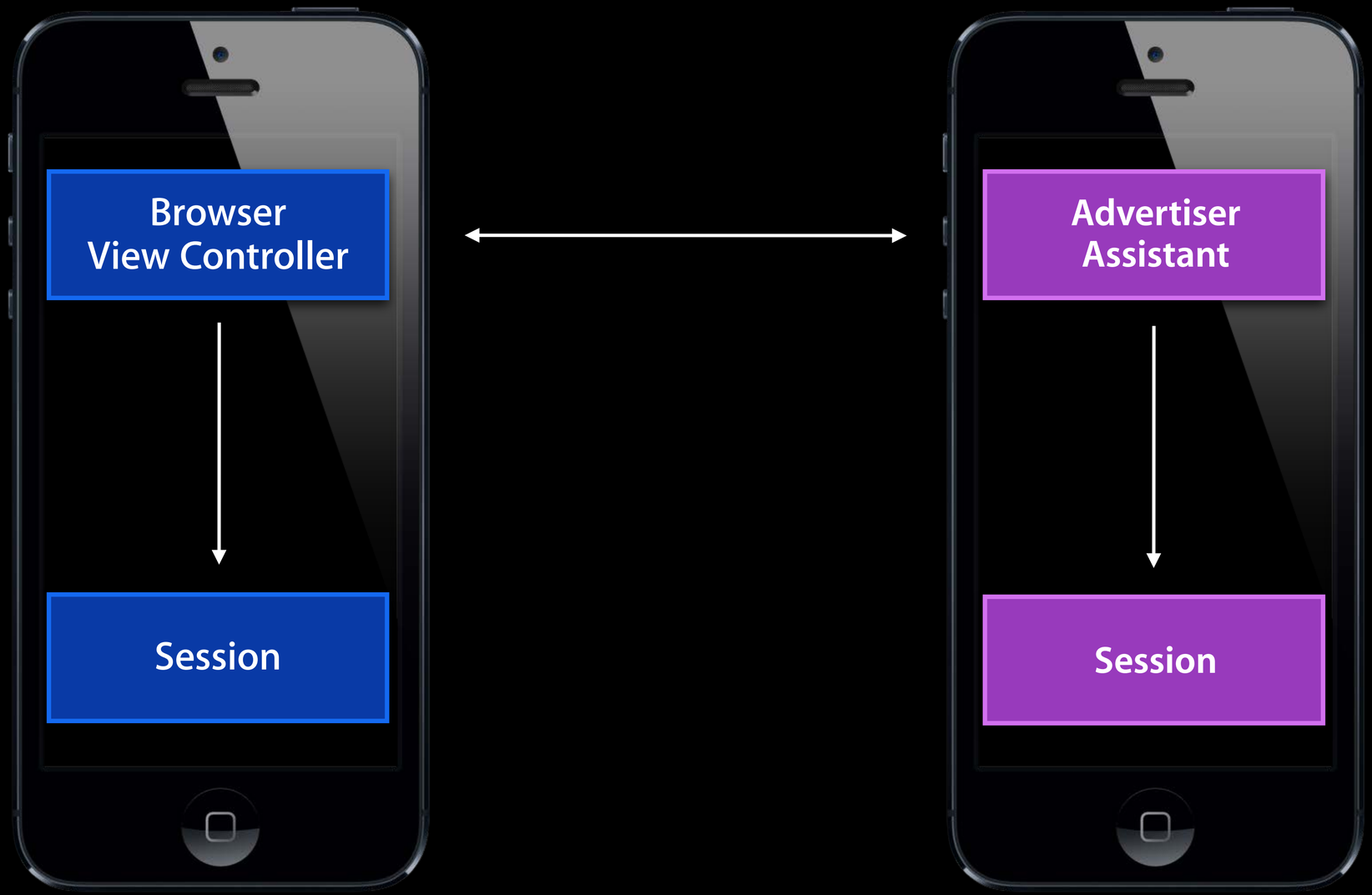


Browser

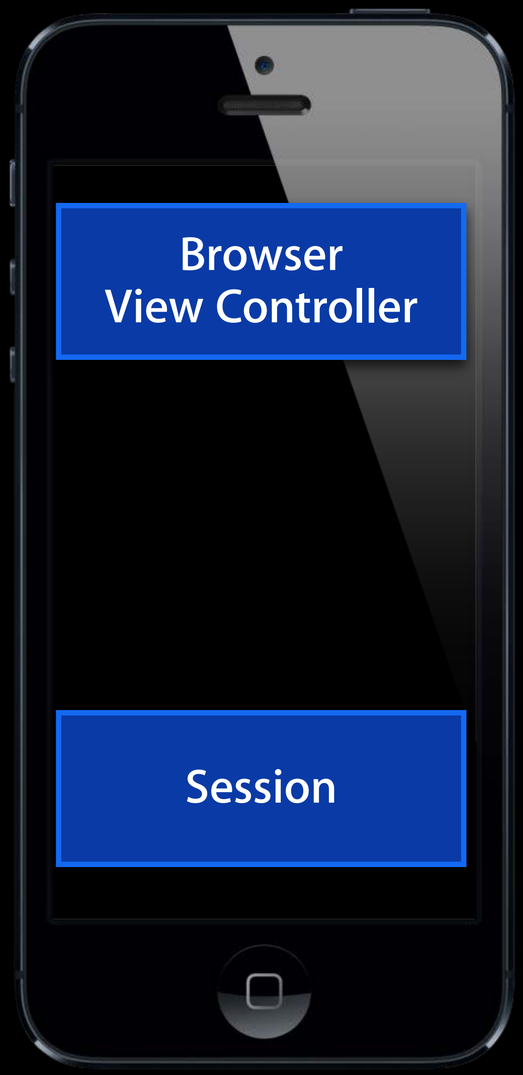


Advertiser

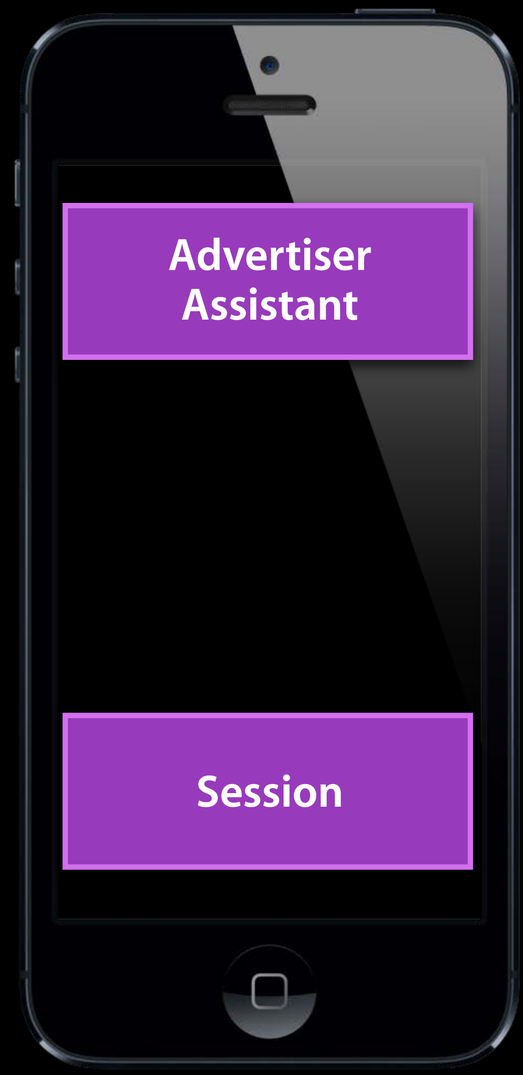
Getting Connected



Getting Connected

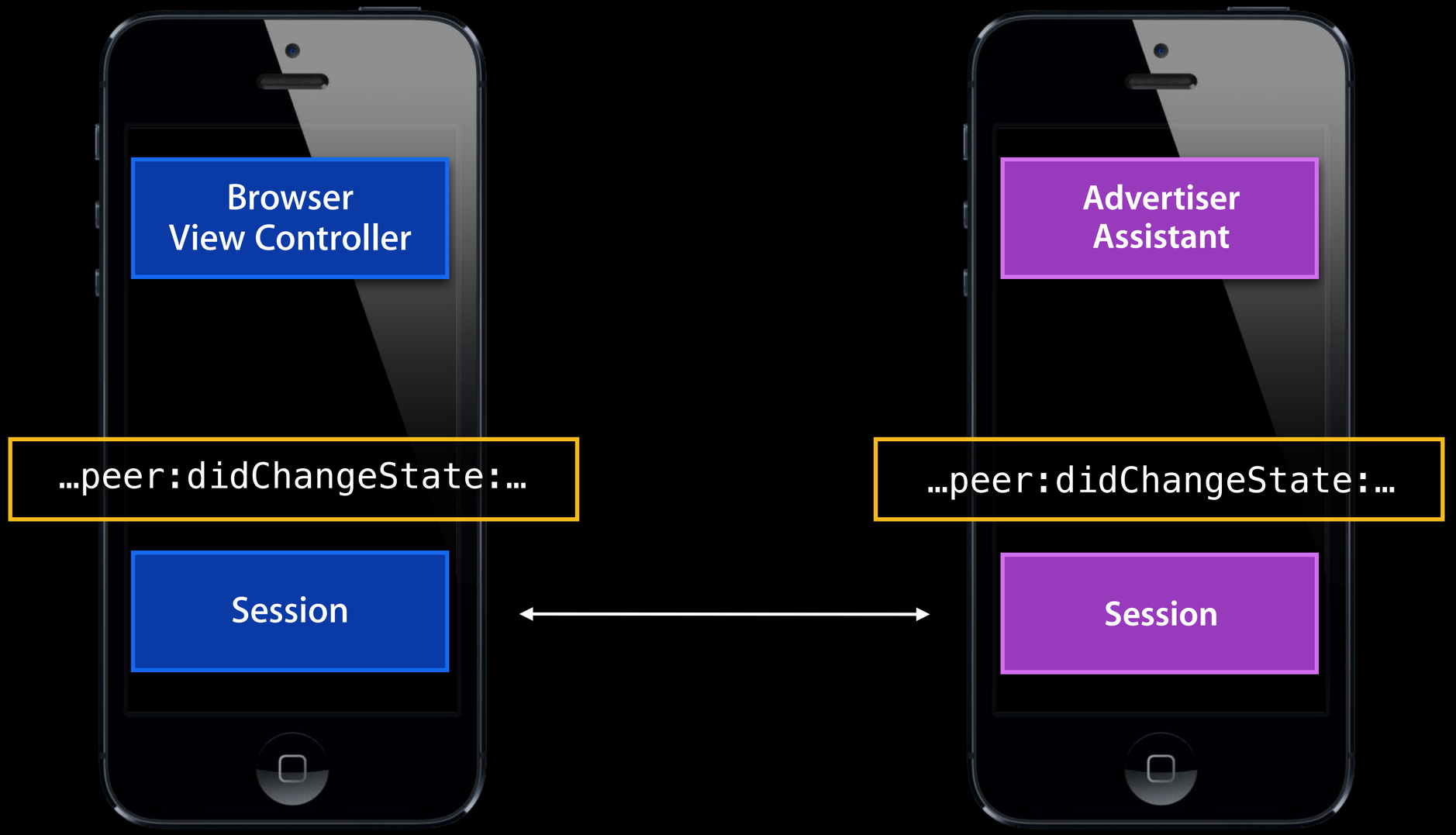


Browser



Advertiser

Getting Connected



Session Entry

- Peers connect to a session

```
// session delegate method
- (void)session:(MCSession *)session
           peer:(MCPeerID *)peerID
 didChangeState:(MCSessionState)state;
```

- Connection successful

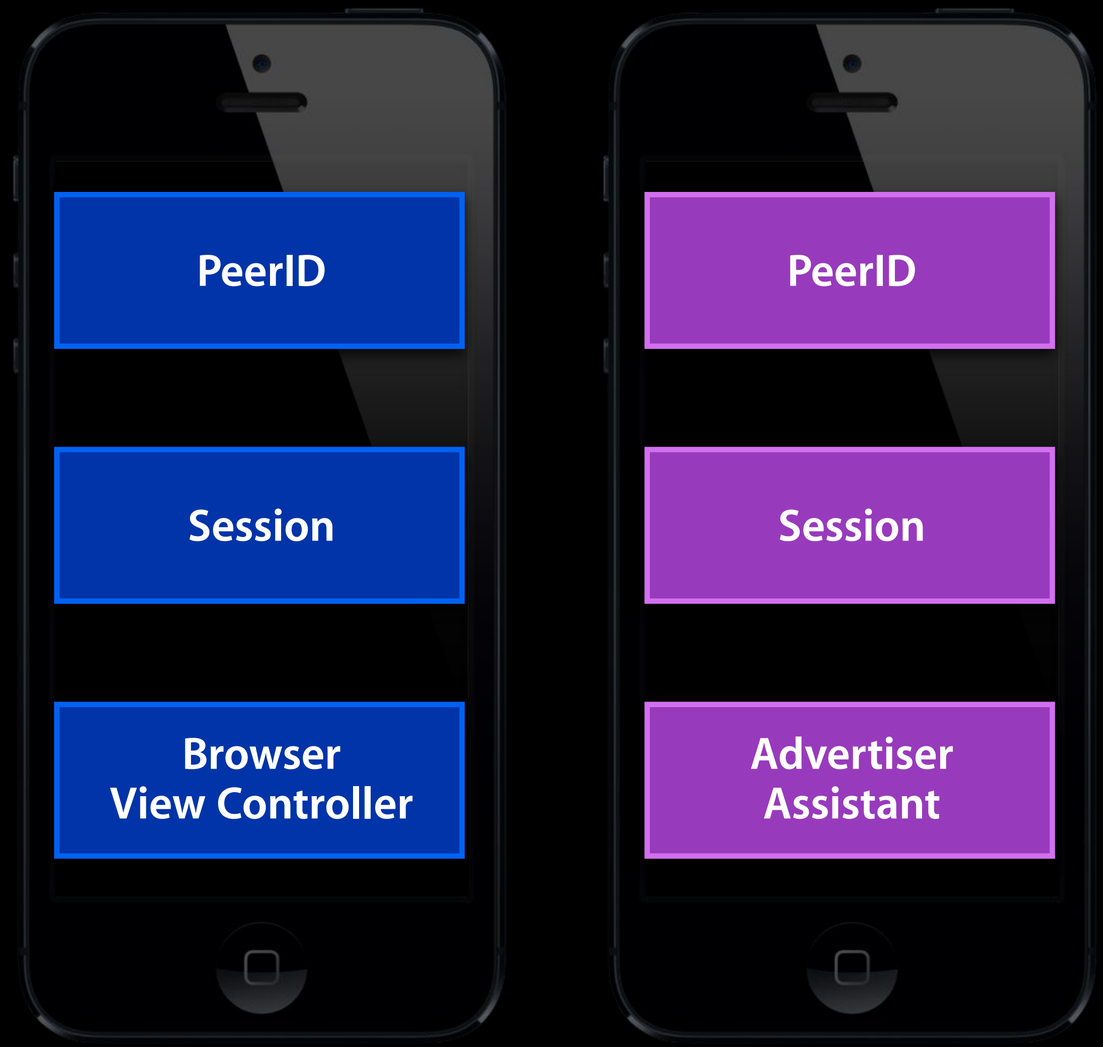
```
state == MCSessionStateConnected;
```

- Connection unsuccessful/Invitation declined

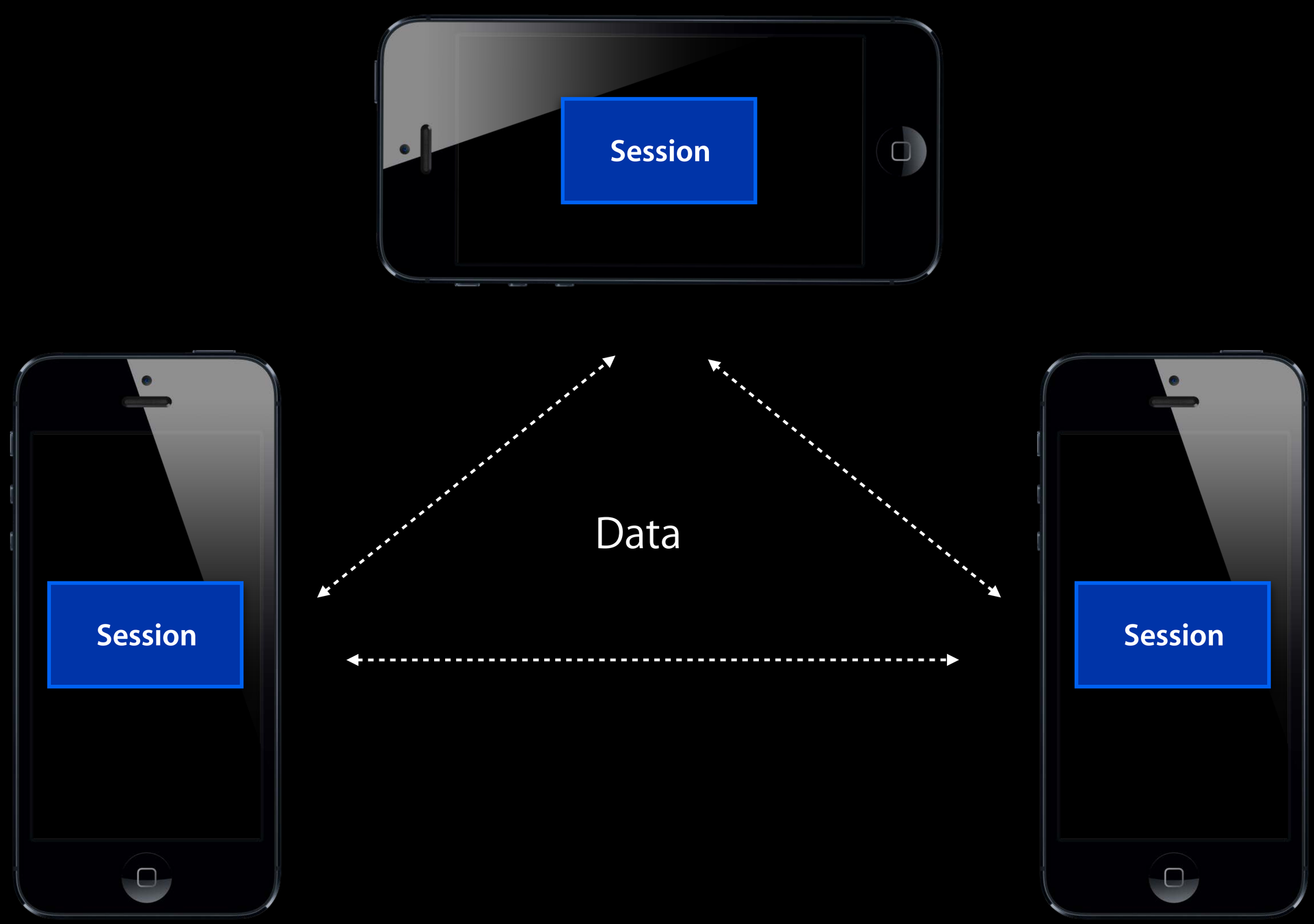
```
state == MCSessionStateNotConnected;
```

Discovery Phase Summary

- Advertiser: instantiate, start
- Browser: instantiate, present
- User-driven

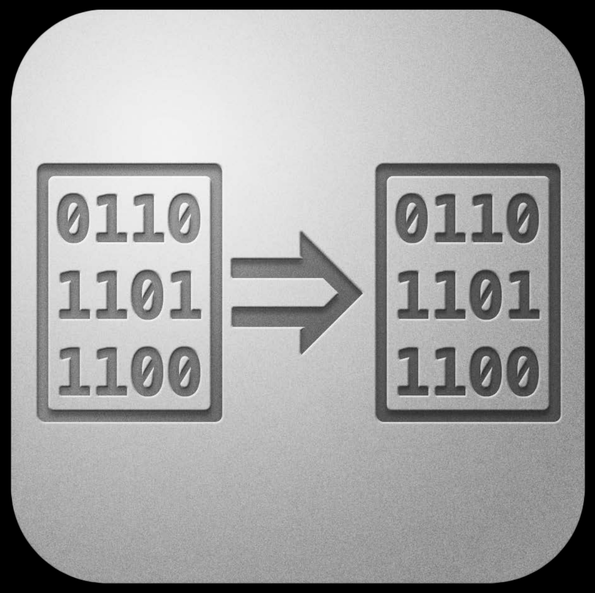


Session Phase


Session
Data
Session
Session

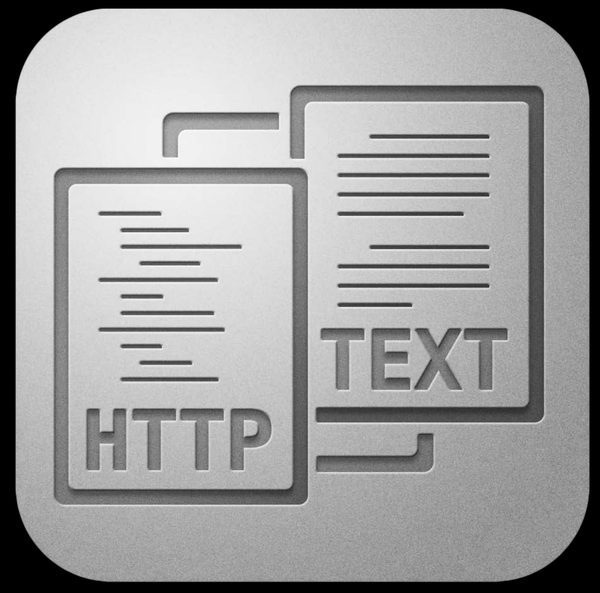

Session
Data
Session
Session

Sending Data

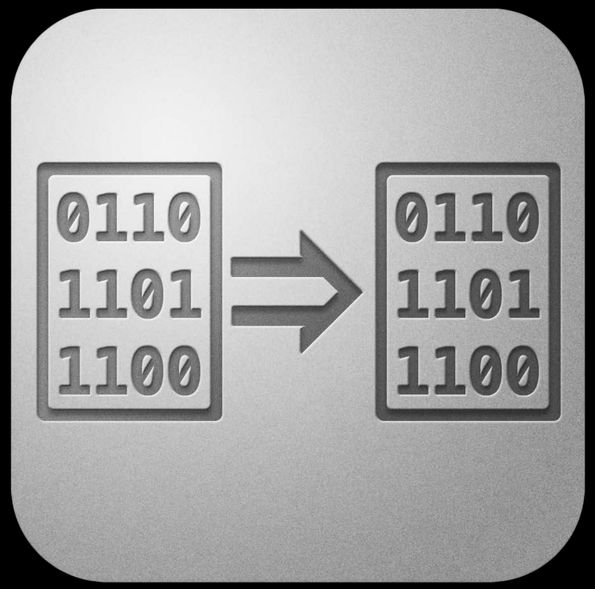


Messages

Streaming

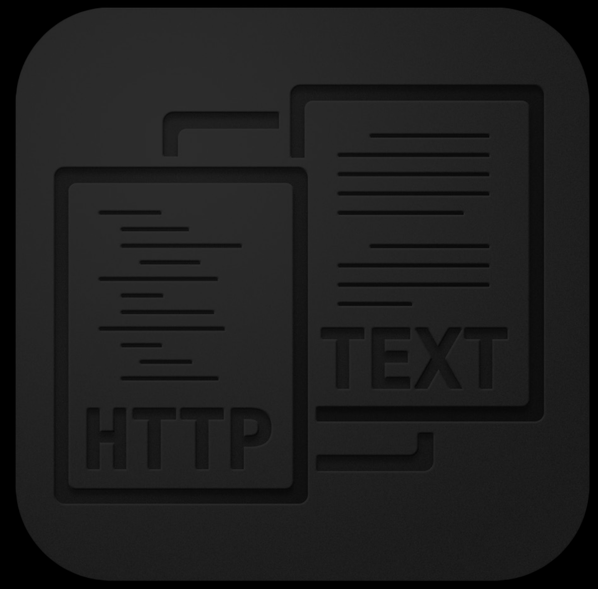


Resources



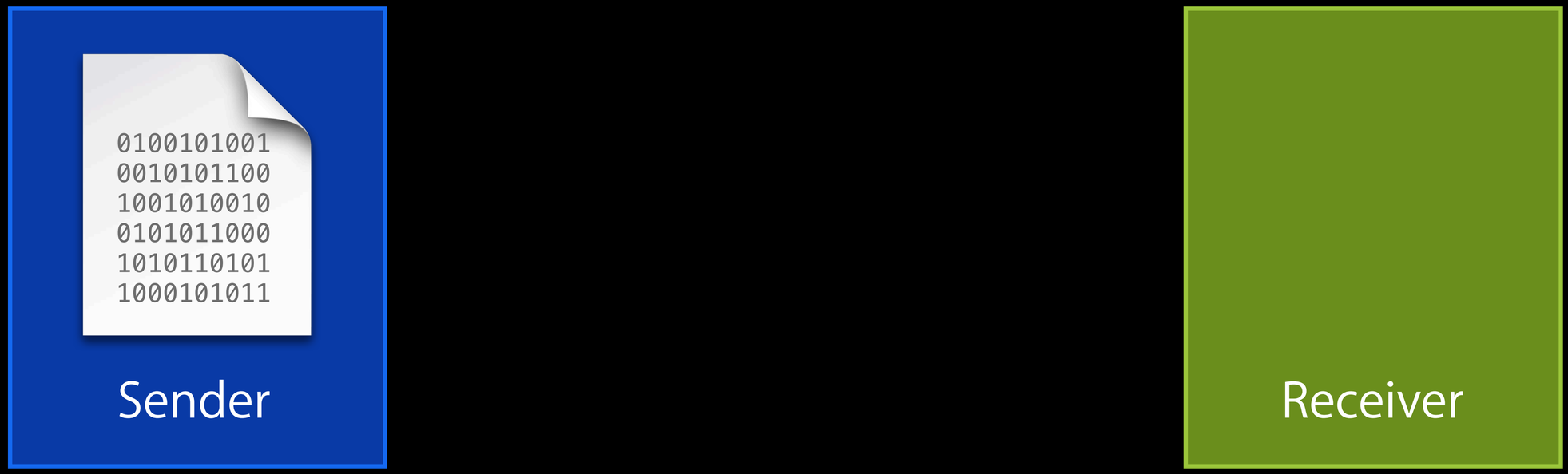
Messages

Streaming



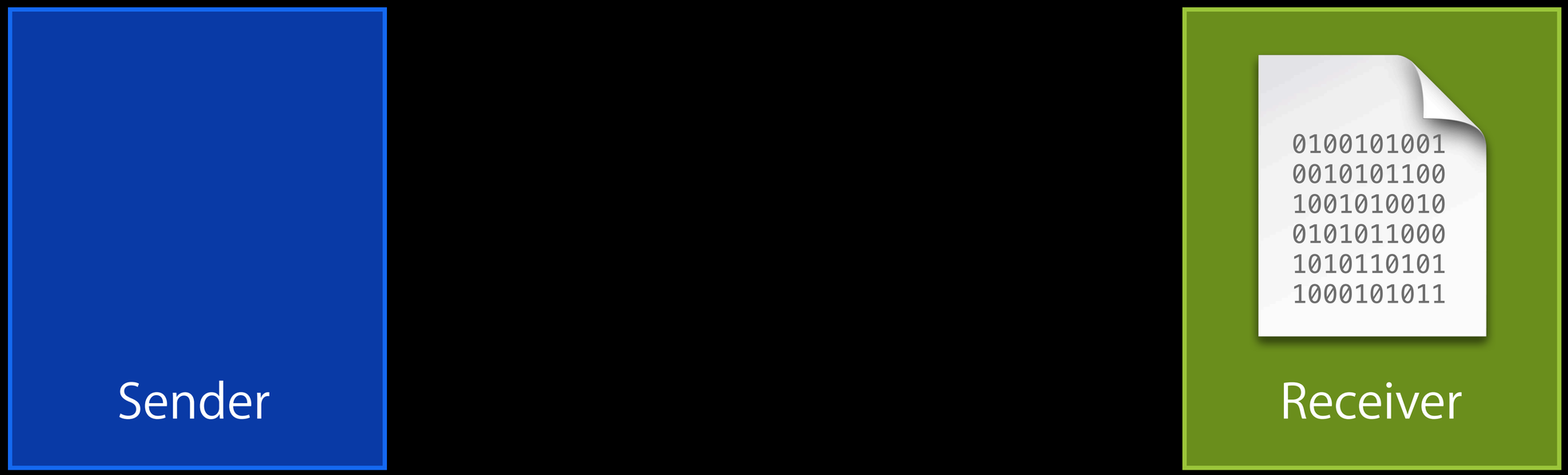
Resources

Messages



Well Defined Boundaries

Messages



Well Defined Boundaries

Messages

Reliable Mode

- Application critical data
- Retransmissions
- In order delivery

Messages

Reliable Mode

- Application critical data
- Retransmissions
- In order delivery

Unreliable Mode

- Time sensitive data
- No delivery guarantees
- No order guarantees

Messages

- Send messages

```
NSArray *peerIDs = [session connectedPeers];
[session sendData:data
          toPeers:peerIDs
         withMode:mode
            error:&error];
```

- Receive messages

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

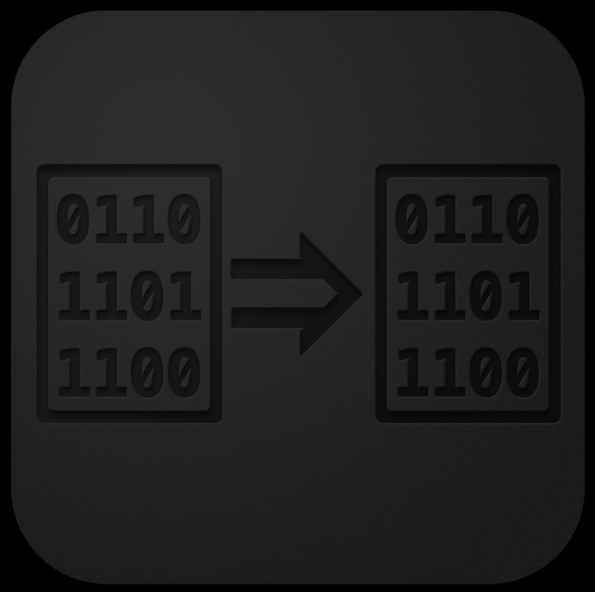
Messages

- Send messages

```
NSArray *peerIDs = [session connectedPeers];
[session sendData:data
          toPeers:peerIDs
         withMode:mode
            error:&error];
```

- Receive messages

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



Messages

Streaming

Resources

Streaming

- Start a stream

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

- Receive a stream request

```
- (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];
```

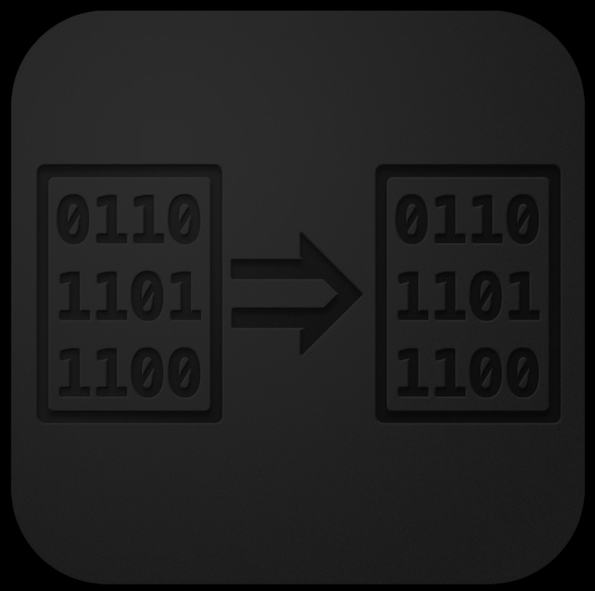
- Receive a stream request

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

Streaming

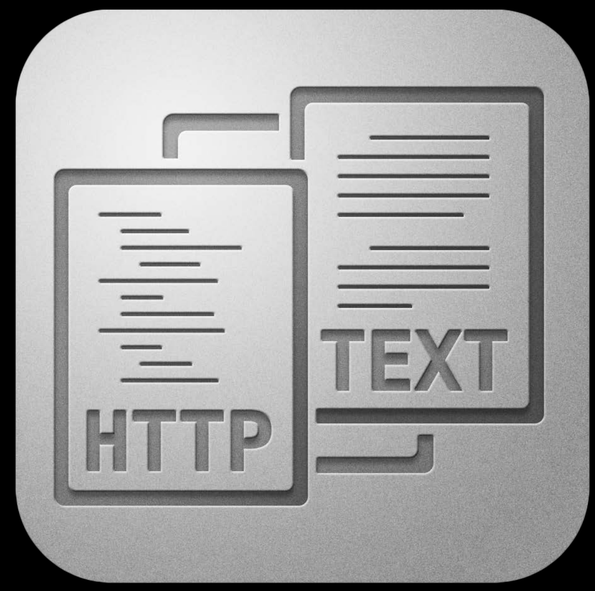
- Setup and handle input/output streams

```
// set delegate
stream.delegate = self;

// schedule in run loop
[stream scheduleInRunLoop:[NSRunLoop mainRunLoop]
                  forMode:NSDefaultRunLoopMode];

// open stream
[stream open]

// implement NSStreamDelegate callbacks
```



Messages

Streaming



Resources

Resources

- Send resource

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

- Control/Query resource transfer

```
progress.fractionCompleted
[progress cancel];
```

Resources

- Send resource

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

- Control/Query resource transfer

```
progress.fractionCompleted
[progress cancel];
```

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;
```

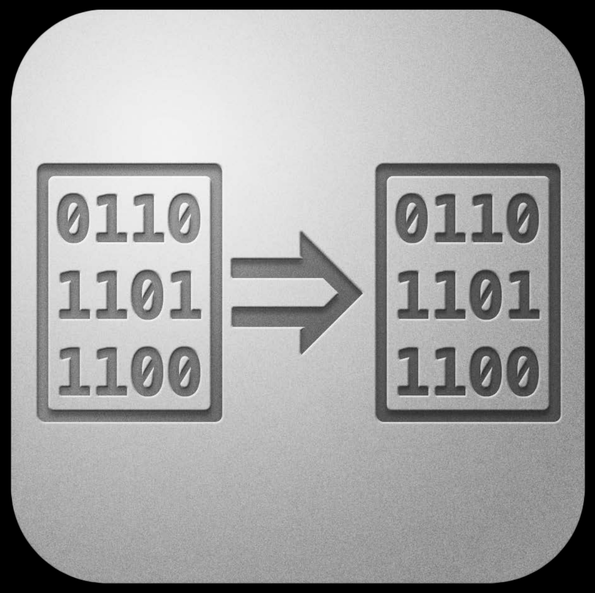
Resources

- Start receiving resource

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

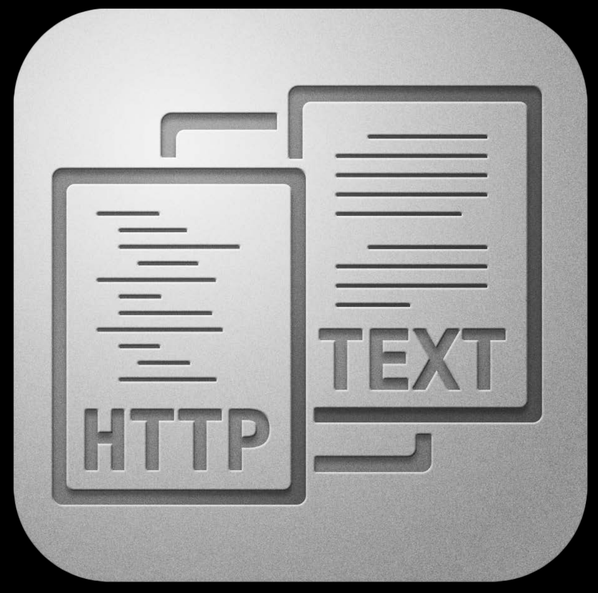
- Finish receiving resource

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



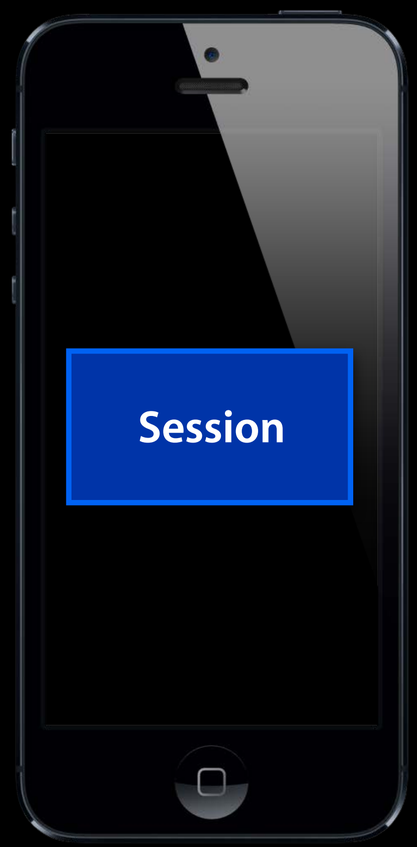
Messages

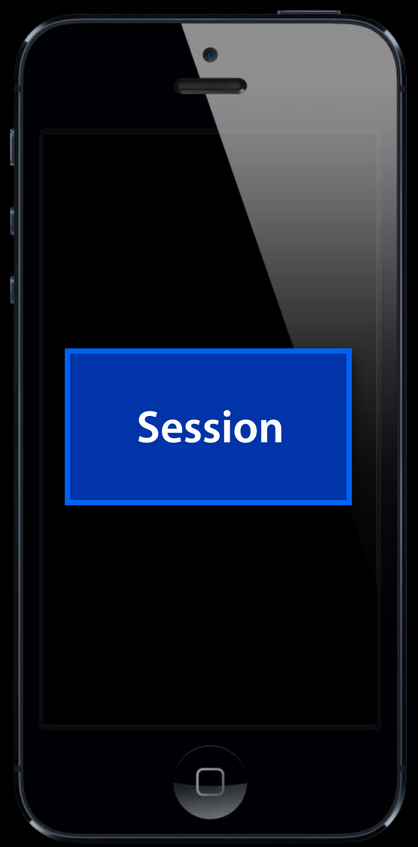
Streaming

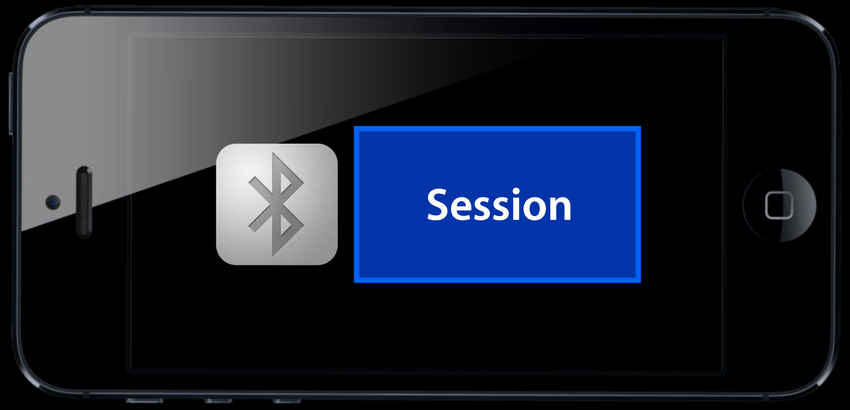


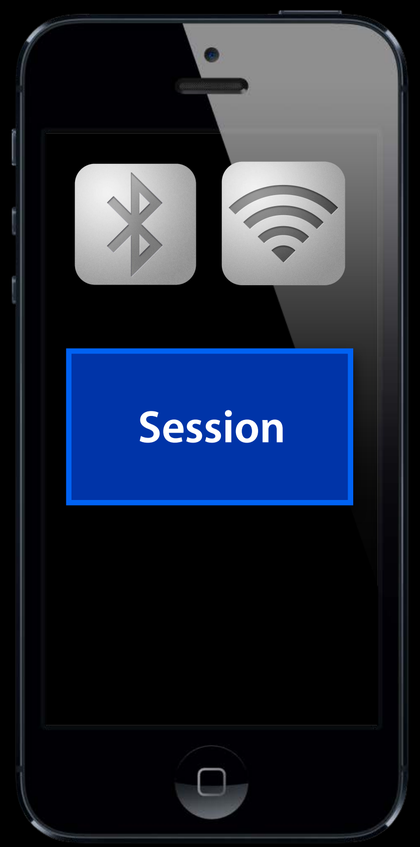
Resources

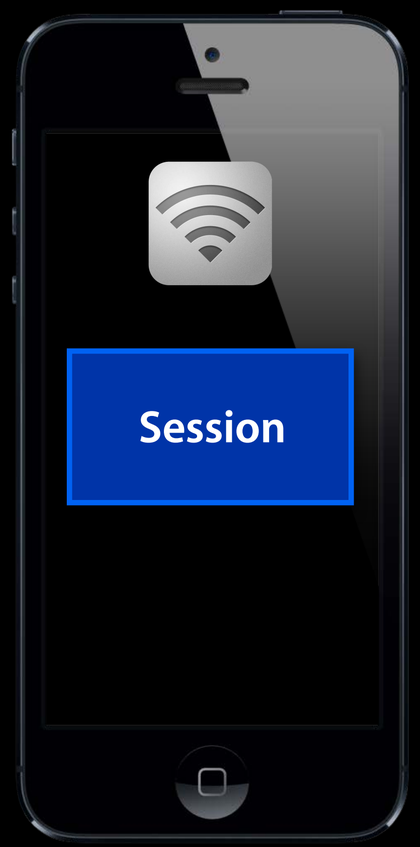

Session

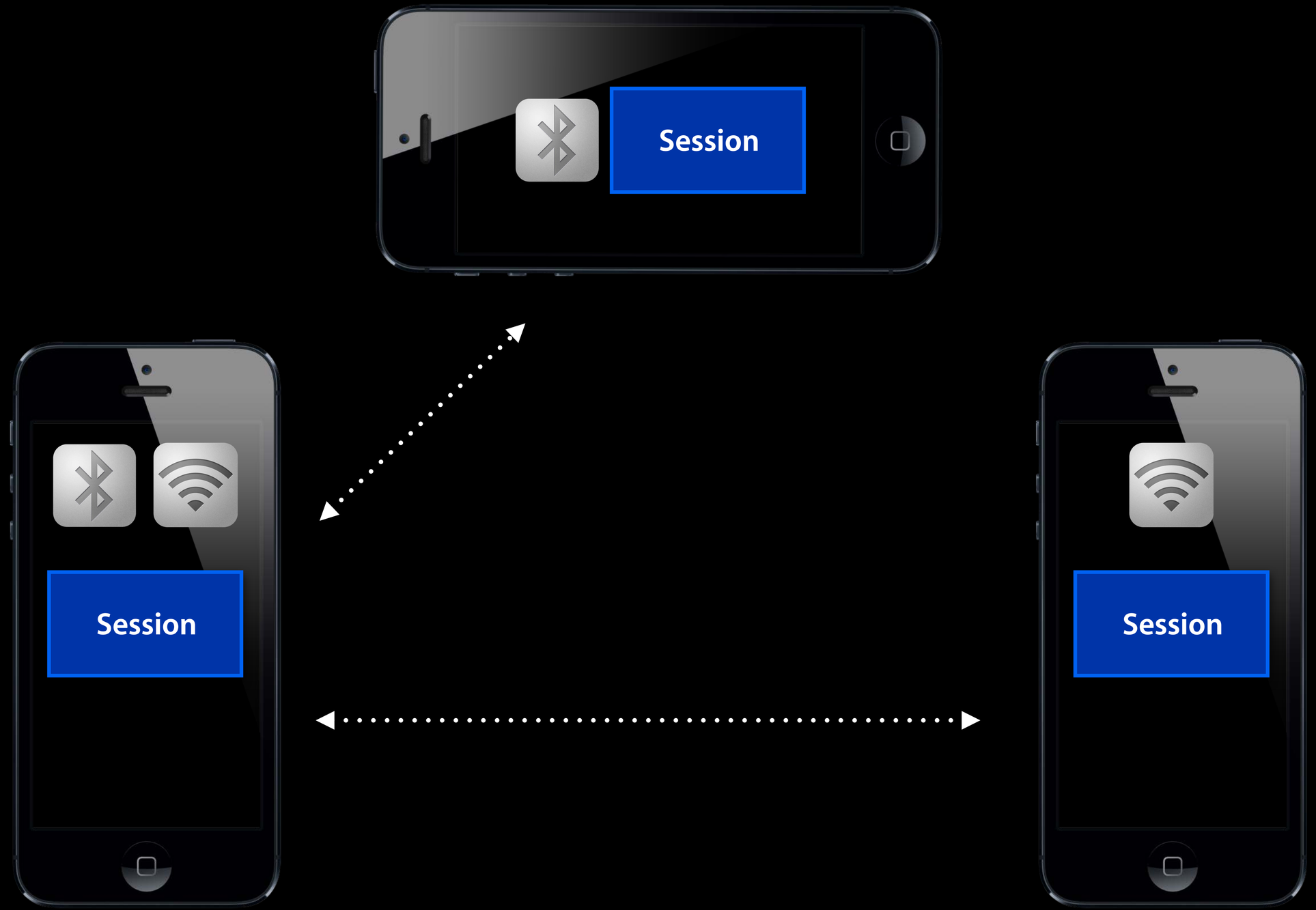

Session

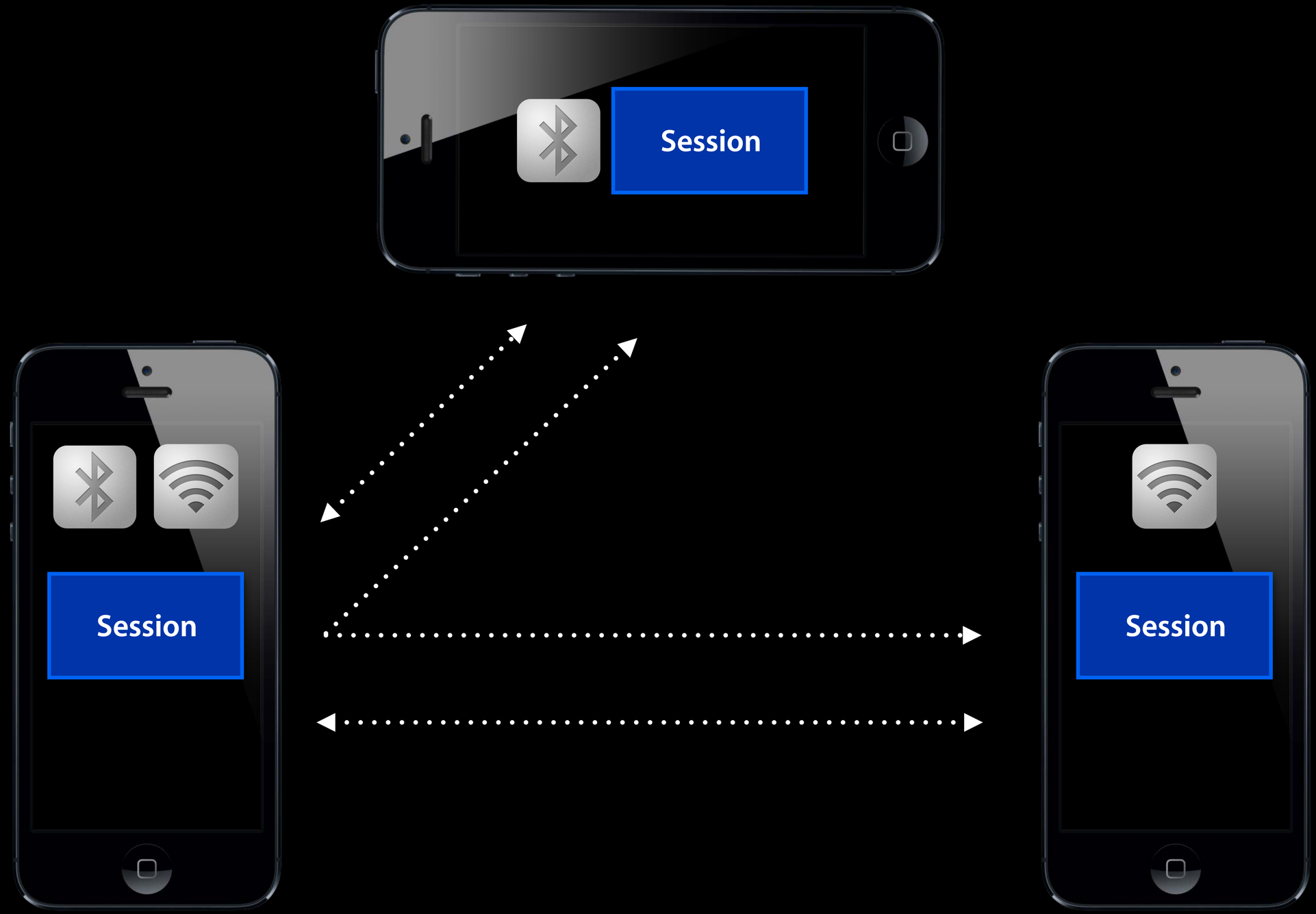

Session


Session


Session


Session


Session
Session
Session


Session
Session
Session

Disconnecting

- Self

```
[session disconnect];
```

- Others

```
- (void)session:(MCSession *)session
          peer:(MCPeerID *)peerID
 didChangeState:(MCSessionState)state;

state == MCSessionStateNotConnected
```

Disconnecting

- Self

```
[session disconnect];
```

- Others

```
- (void)session:(MCSession *)session
          peer:(MCPeerID *)peerID
 didChangeState:(MCSessionState)state;

state == MCSessionStateNotConnected
```

Session Phase Summary

- Three ways of sending data
  1. Messages (reliable and unreliable mode)
  2. Streaming
  3. Resources

…and that's it!

Advanced

Programmatic Discovery

Programmatic Discovery

- More flexibility
- Finding devices/sending invitations handled programmatically
- Build a custom UI for discovery

Programmatic Advertising

- Initialization

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

- Start advertising

```
advertiser.delegate = self;
[advertiser startAdvertisingPeer];
```

Programmatic Advertising

- Initialization

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

- Start advertising

```
advertiser.delegate = self;
[advertiser startAdvertisingPeer];
```

Programmatic Browsing

- Initialization

```
MCNearbyServiceBrowser *browser =
    [[MCNearbyServiceBrowser alloc] initWithPeer:myPeerID
                                     serviceType:serviceType];
```

- Start browsing

```
browser.delegate = self;
[browser startBrowsingForPeers];
```

Programmatic Browsing

- Initialization

```
MCNearbyServiceBrowser *browser =
    [[MCNearbyServiceBrowser alloc] initWithPeer:myPeerID
                                   serviceType:serviceType];
```

- Start browsing

```
browser.delegate = self;
[browser startBrowsingForPeers];
```


Browser
Advertiser
Advertiser


```
…foundPeer:
withDiscoveryInfo:…
```

…foundPeer:
withDiscoveryInfo:…


```
…foundPeer:
withDiscoveryInfo:…
```

…foundPeer:
withDiscoveryInfo:…

Finding/Losing Peers

- Peer found

```
- (void)      browser:(MCNearbyServiceBrowser *)browser
           foundPeer:(MCPeerID *)peerID
   withDiscoveryInfo:(NSDictionary *)info;
```

- Peer lost

```
- (void)      browser:(MCNearbyServiceBrowser *)browser
            lostPeer:(MCPeerID *)peerID;
```

Finding/Losing Peers

- Peer found

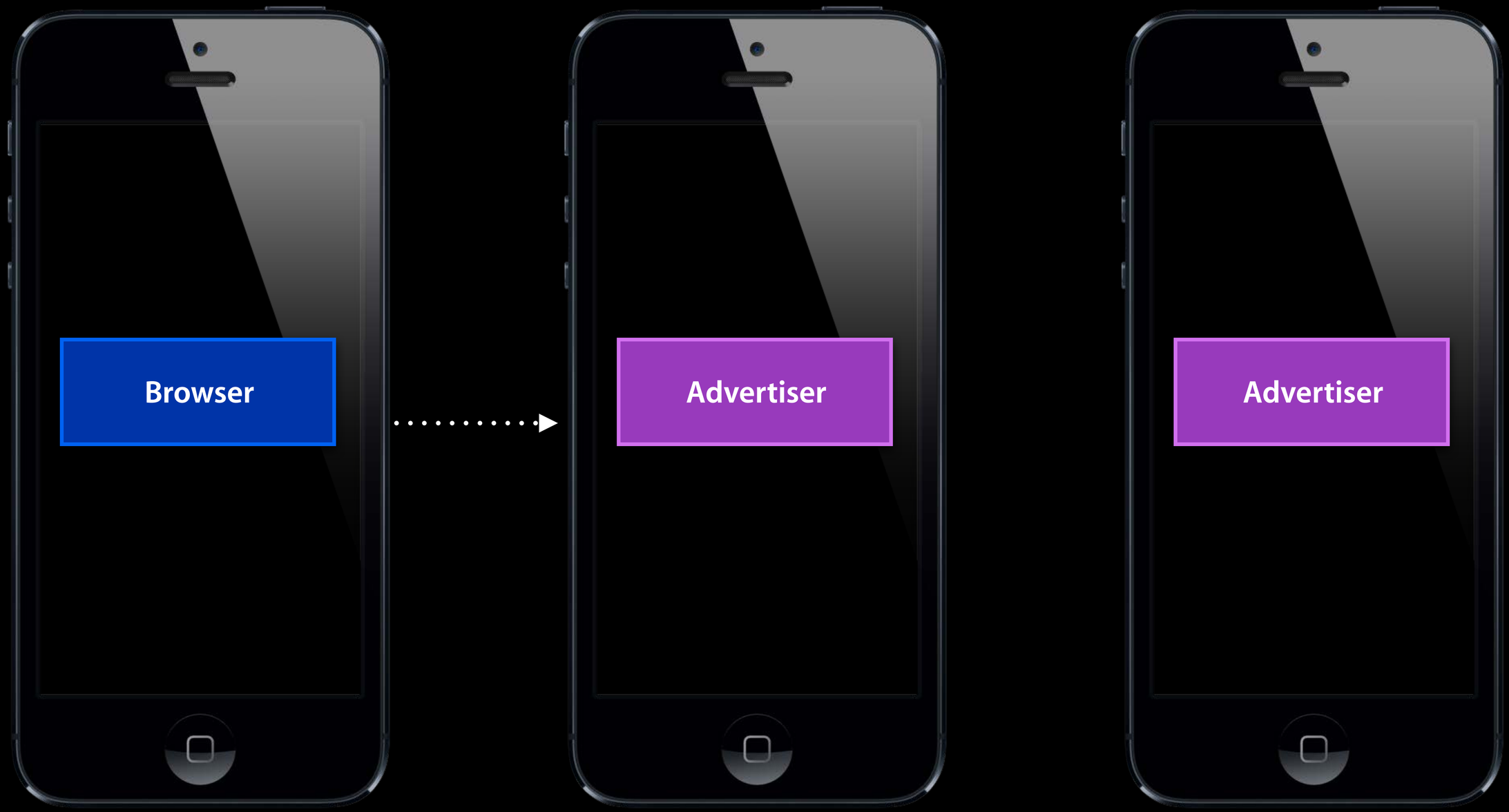
```
- (void)      browser:(MCNearbyServiceBrowser *)browser
           foundPeer:(MCPeerID *)peerID
   withDiscoveryInfo:(NSDictionary *)info;
```

- Peer lost

```
- (void)      browser:(MCNearbyServiceBrowser *)browser
           lostPeer:(MCPeerID *)peerID;
```

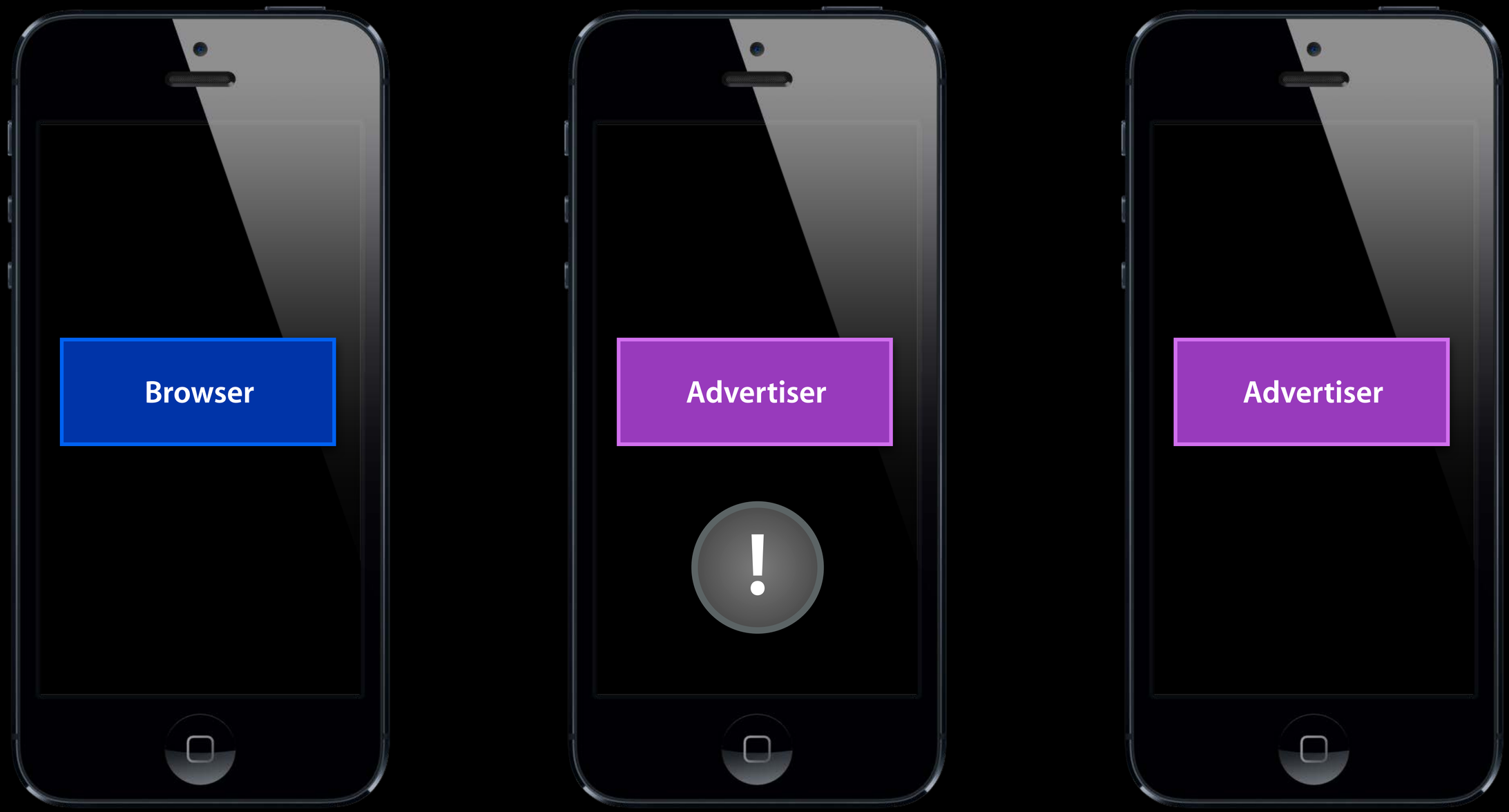

Browser
Advertiser
Advertiser


```
…invitePeer:
 toSession:…
```



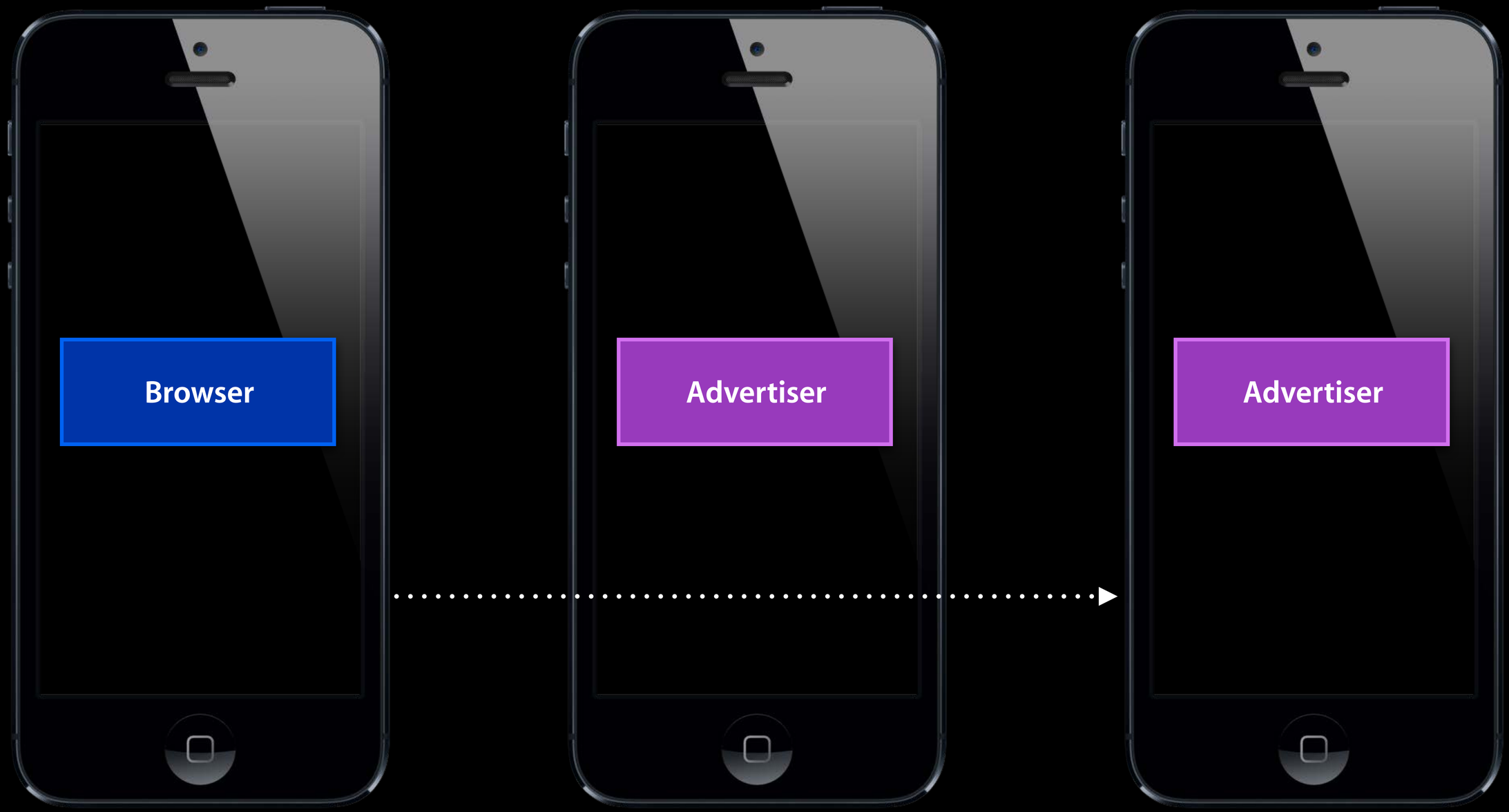

```
…invitePeer:
 toSession:…
```


```
…didReceiveInvitationFromPeer:
… invitationHandler:…
```




```
…didReceiveInvitationFromPeer:
… invitationHandler:…
```

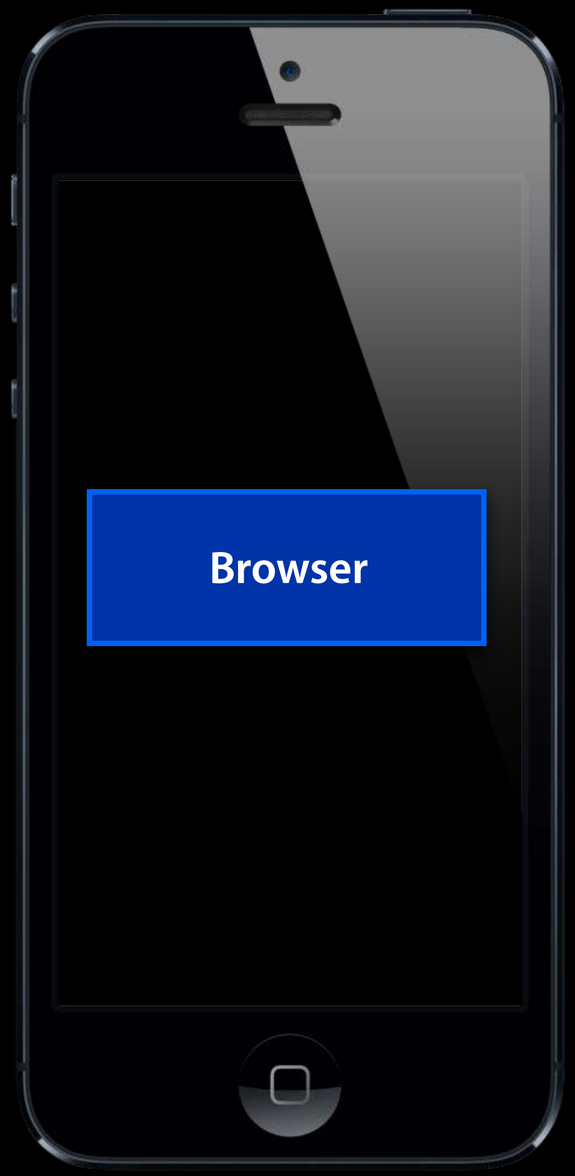

```
…invitePeer:
 toSession:…
```

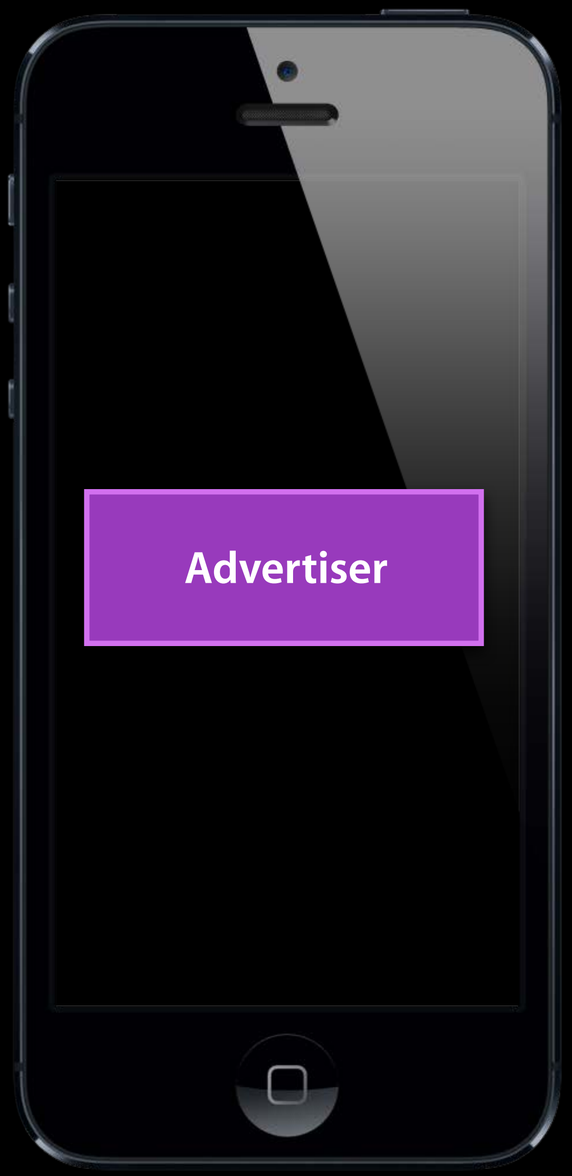



```
…invitePeer:
 toSession:…
```

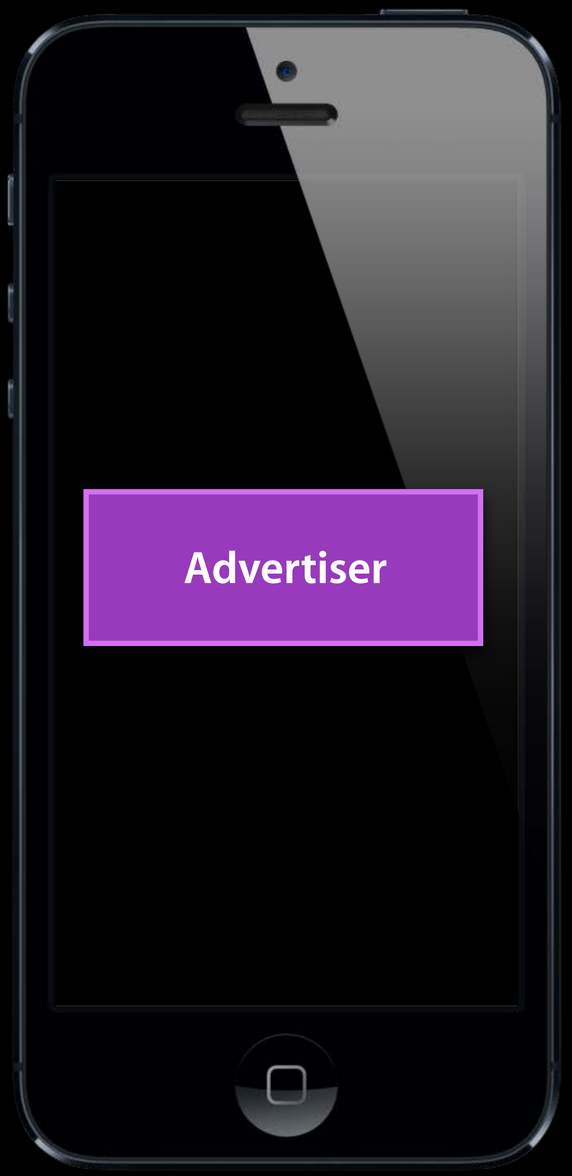

```
…didReceiveInvitationFromPeer:
… invitationHandler:…
```


```
…didReceiveInvitationFromPeer:
… invitationHandler:…
```

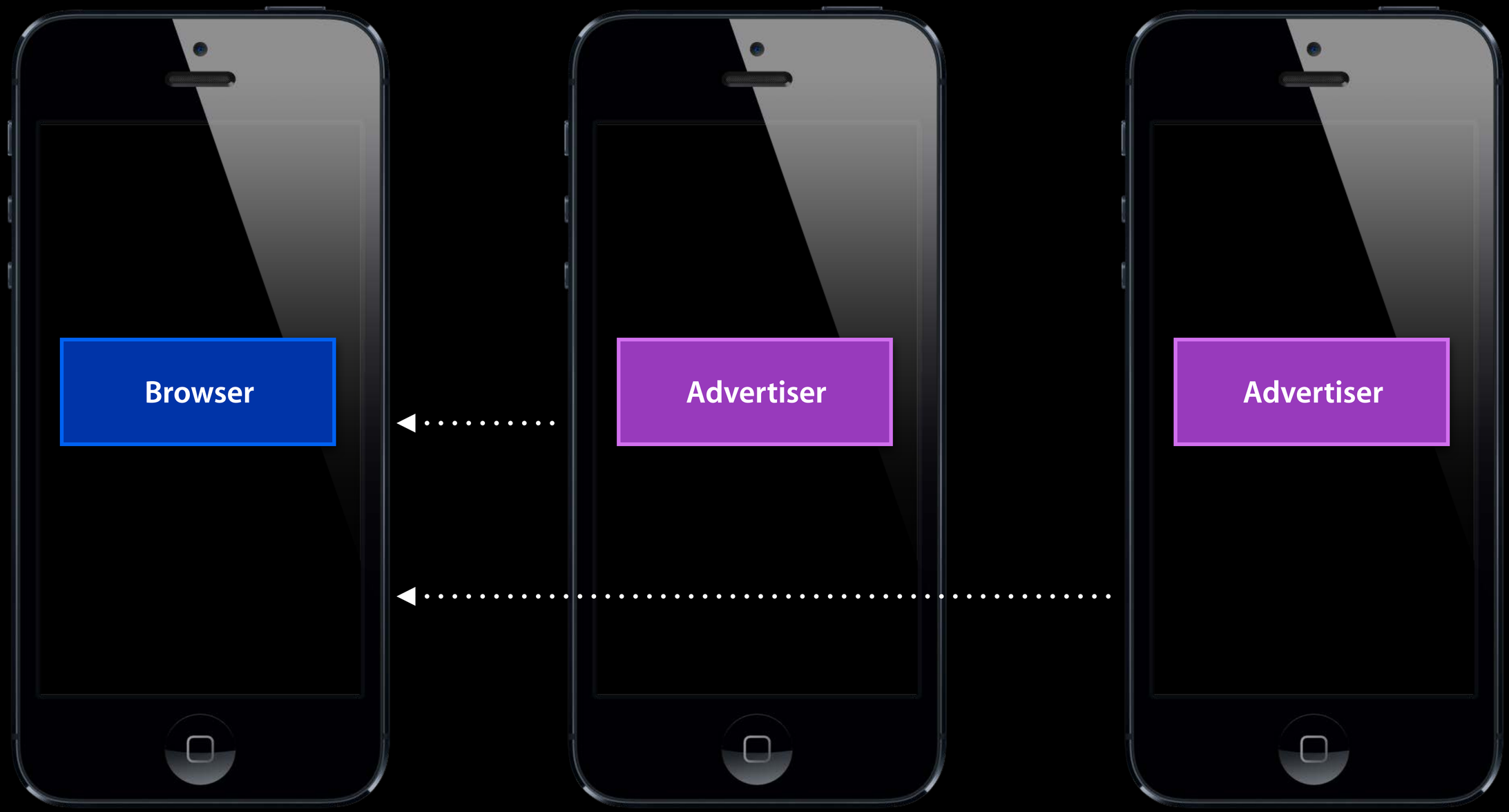





```
invitationHandler
(YES,session);
```




```
invitationHandler
(YES,session);
```


Browser
Advertiser
Advertiser
invitationHandler
(YES,session);
invitationHandler
(YES,session);

Send/Receive Invitation

- Send invitation

```
[browser invitePeer:peerID
          toSession:session
        withContext:context
            timeout:timeout];
```

- Receive invitation

```
- (void)            advertiser:(MCNearbyServiceAdvertiser *)advertiser
  didReceiveInvitationFromPeer:(MCPeerID *)peerID
                   withContext:(NSData *)context
             invitationHandler:(void(^)(BOOL accept,
                                        MCSession *session))invitationHandler;
```

Send/Receive Invitation

- Send invitation

```
[browser invitePeer:peerID
          toSession:session
        withContext:context
            timeout:timeout];
```

- Receive invitation

```
- (void)              advertiser:(MCNearbyServiceAdvertiser *)advertiser
  didReceiveInvitationFromPeer:(MCPeerID *)peerID
                   withContext:(NSData *)context
             invitationHandler:(void(^)(BOOL accept,
                                        MCSession *session))invitationHandler;
```

Advertiser

Receiving invitations

```
- (void)           advertiser:(MCNearbyServiceAdvertiser *)advertiser
 didReceiveInvitationFromPeer:(MCPeerID *)peerID
                  withContext:(NSData *)context
            invitationHandler:(void(^)(BOOL accept,
                                       MCSession *session))invitationHandler
{
}
```

Advertiser

Receiving invitations

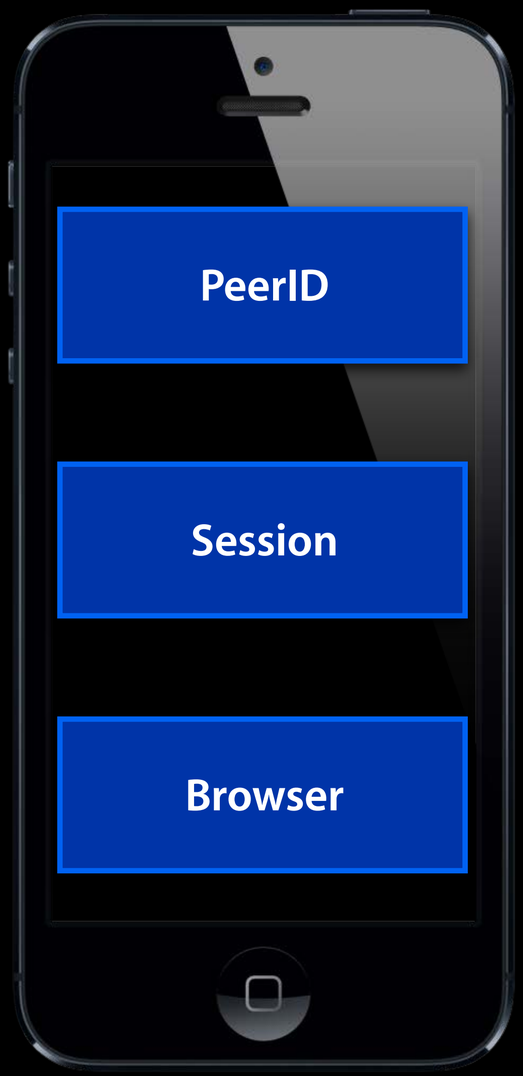
```
{
    // copy and store the invitation handler

    // ask the user
    UIAlertView *alertView = [[UIAlertView alloc]
                                  initWithTitle:title
                                        message:message
                                       delegate:self
                              cancelButtonTitle:@”Decline”
                              otherButtonTitles:@”Accept”, nil];

    // present alert view
    [alertView show];
}
```

Advertiser

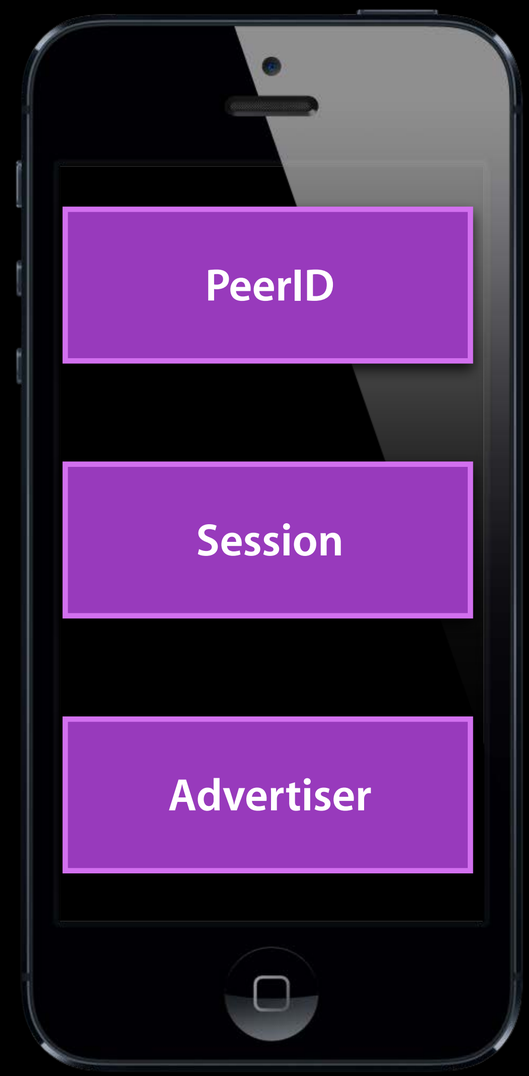
Responding to an invitation

```
- (void)      alertView:(UIAlertView *)alertView
   clickedButtonAtIndex:(NSInteger)buttonIndex
{
    // retrieve the invitationHandler

    // get user decision
    BOOL accept = (buttonIndex != alertView.cancelButtonIndex) ? YES : NO;

    // respond
    invitationHandler(accept, session);
}
```

Getting Connected

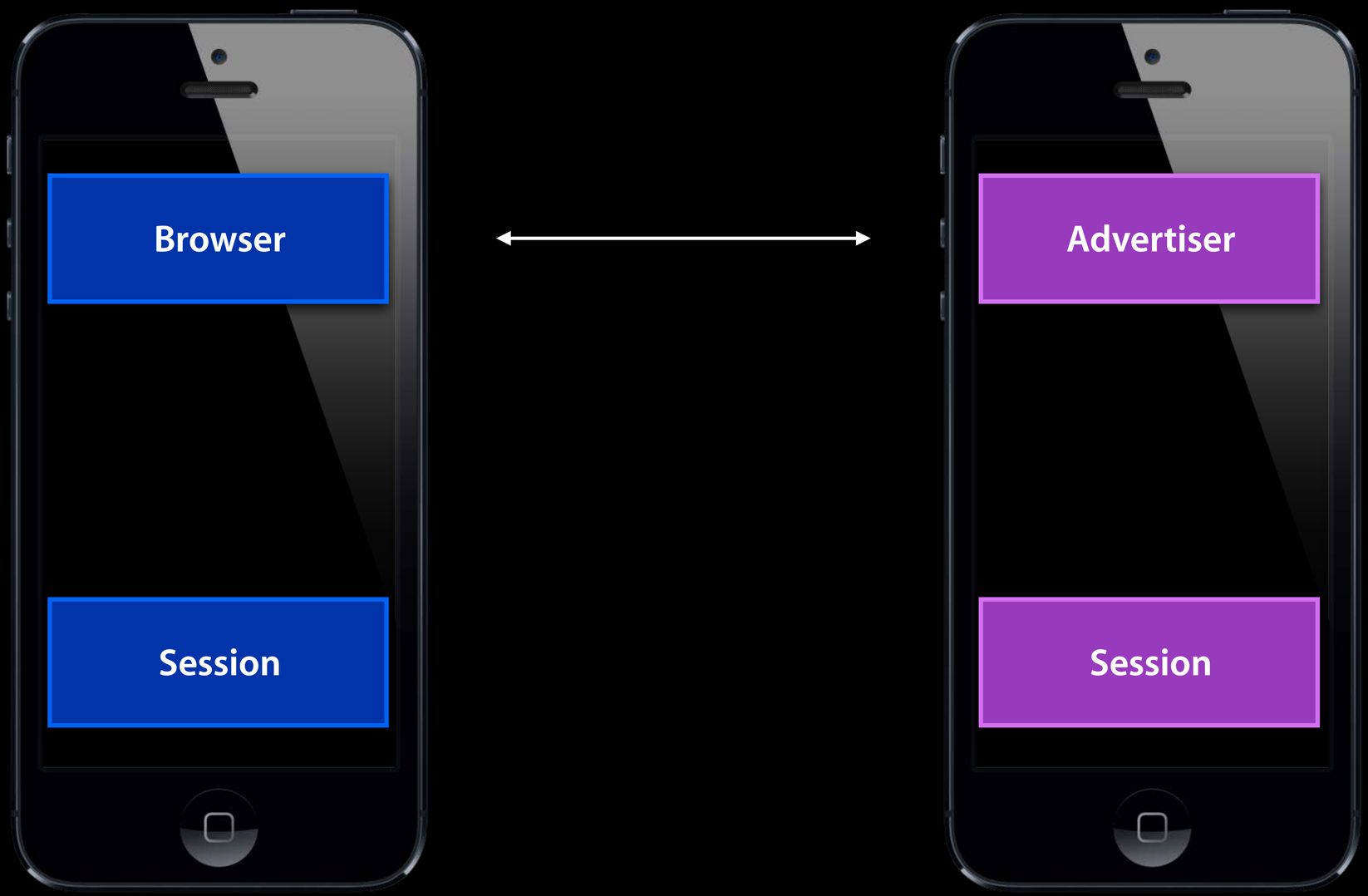


Browser



Advertiser

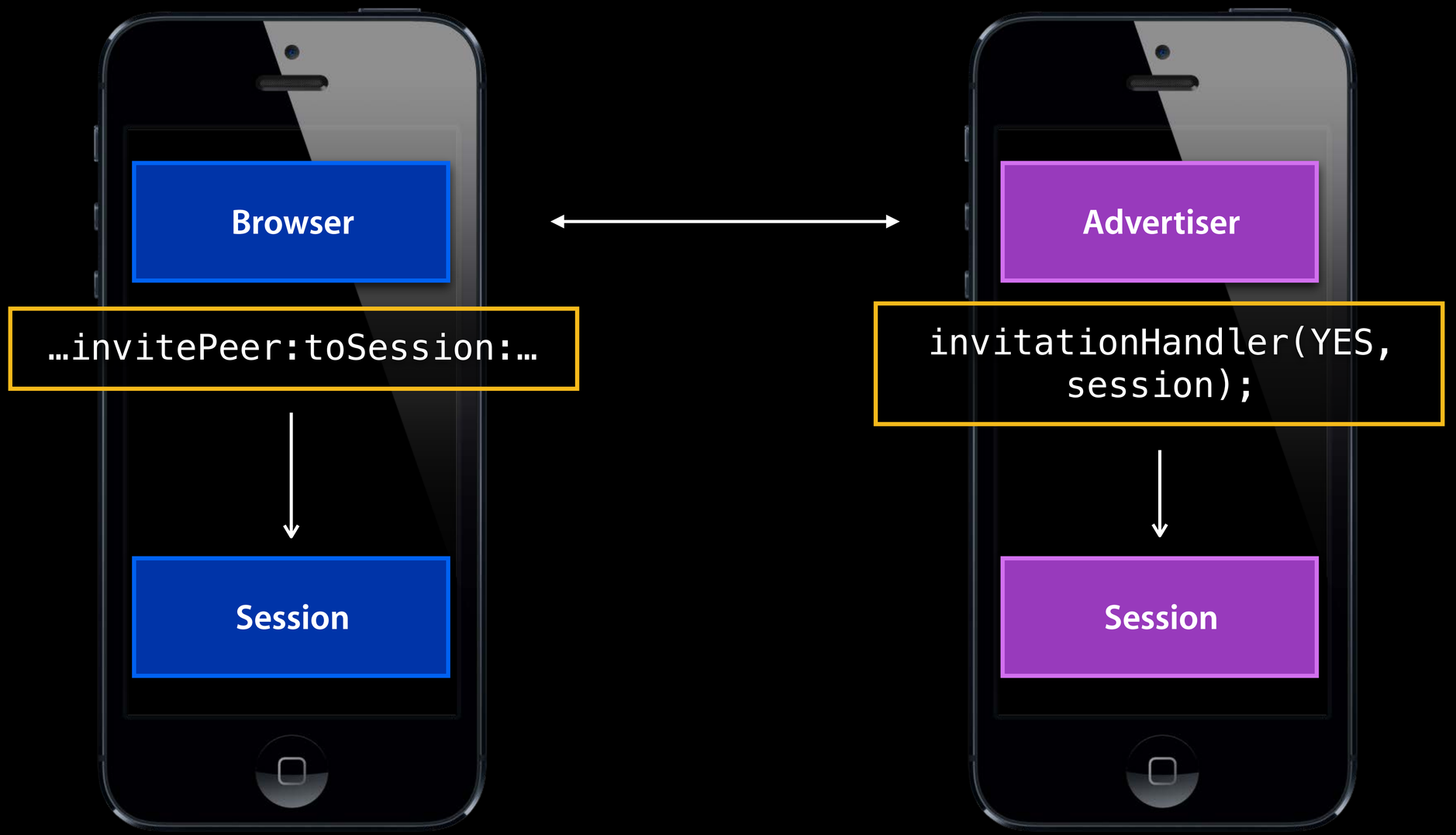
Getting Connected



Browser

Advertiser

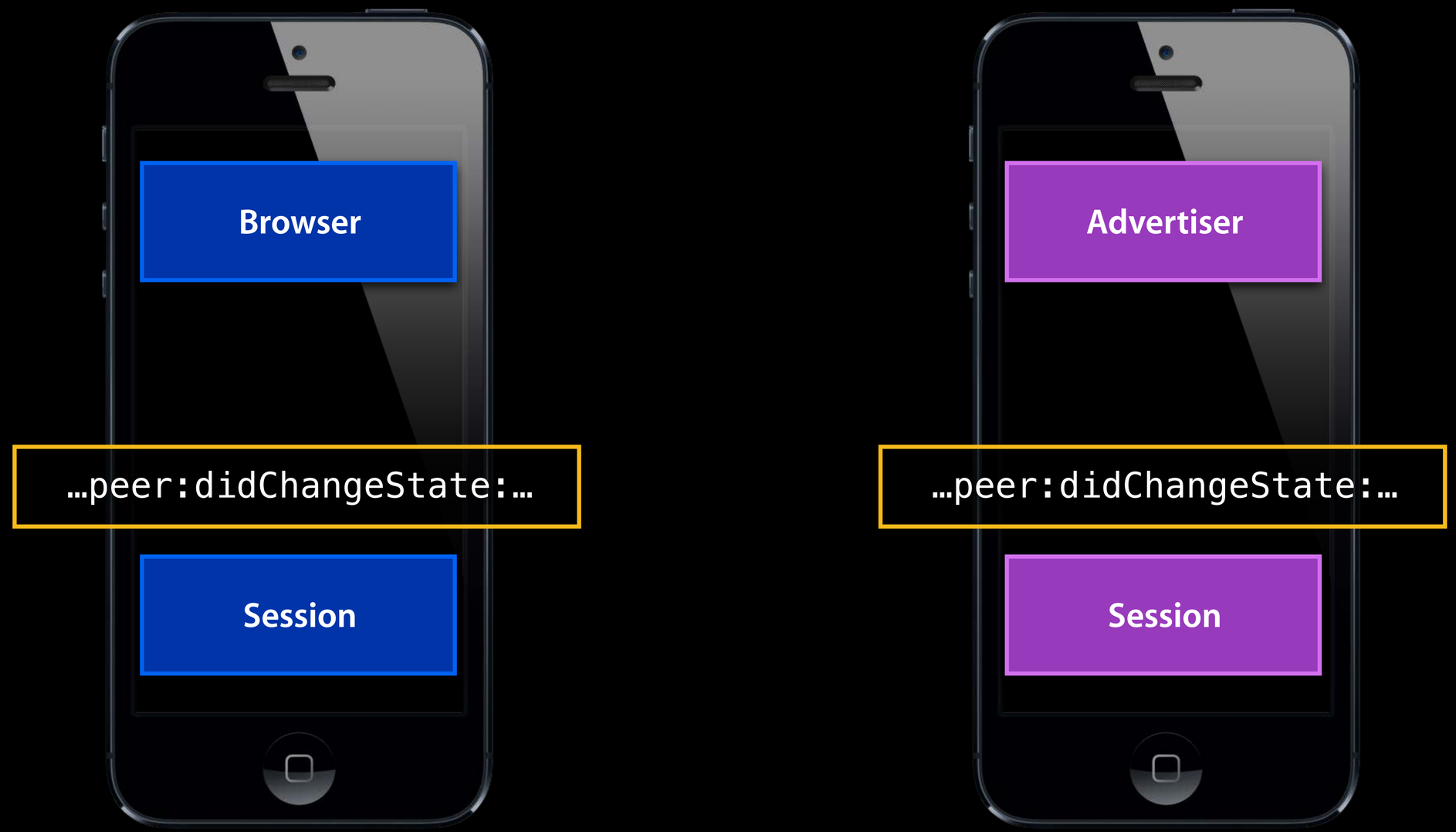
Getting Connected



Browser

Advertiser

Getting Connected



Browser

Advertiser

Security

Security

Authentication

Encryption

Security

Authentication

Encryption

Authentication


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

Authentication


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

Authentication

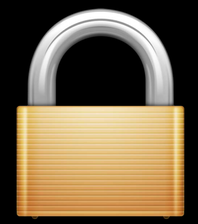

```
- (void)           session:(MCSession *)session
    didReceiveCertificate:(NSArray *)certificate
                 fromPeer:(MCPeerID *)peerID
       certificateHandler:(void(^)(BOOL accept))certificateHandler
```

Security

Authentication

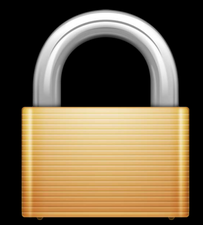
Encryption

Encryption



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

Encryption



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

MCEncryptionPreference

None

Optional

Required

Advanced

Summary

- Programmatic discovery
  - Want to build custom discovery UI? You can!
  - MCNearbyServiceAdvertiser, MCNearbyServiceBrowser
- Security
  - Authentication
  - Encryption

Some Best Practices

- Start advertising on app launch
- Stop advertising when not needed
- Stop browsing when done
- Keep discovery info short
- Keep display names short
- Send short unreliable messages for best latency

More Information

Paul Danbold
Core OS Evangelist
danbold@apple.com

Documentation
Multipeer Connectivity Framework Reference
https://developer.apple.com/library/prerelease/ios

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

Labs

Multipeer Connectivity Lab	Core OS Lab A Wednesday 11:30AM
Foundation Networking Lab	Core OS Lab B Wednesday11:30AM
Networking Lab	Core OS Lab A Thursday 9:00AM
Security Lab	Core OS Lab B Thursday 2:00PM
Multipeer Connectivity Lab	Core OS Lab B Friday 9:00AM

WWDC2013