RWTH AACHEN UNIVERSITY

iPhone Application Programming
Networking

Jonathan Diehl
Media Computing Group
RWTH Aachen University

WS 2013/2014
http://hci.rwth-aachen.de/iphone

Networking

Game Center

Multipeer Connectivity

AirDrop

Push Notifications

URL Loading System

Bonjour

Socket Networking

AsyncNetwork

Game Center

- Leaderboards, Achievements, Challenges
- Matchmaking
  - Real-time: all players are connected simultaneously and can exchange data
  - Turn-based: players connect sequentially, data is exchanged as needed
  - Self-hosted: Game Center provides the players, you provide the networking

Game Center Checklist

- Set up Game Center in the developer program
  - Requires explicit app ID and provisioning profile (ask Jan-Peter)
- Enable Game Center in the list of capabilities of your app
- Add GameKit framework
- Authenticate the local player
- Implement Game Center features

Game Center Authentication

```
GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];
localPlayer.authenticateHandler = ^(UIViewController *viewController, NSError *error)
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];
    if (viewController != nil)
    {
        // show the authentication dialog by presenting viewController
    }
    else if (localPlayer.isAuthenticated)
    {
        // store localPlayer as the authenticated player
    }
    else
    {
        // disable Game Center
    }
};
```

Real-Time Matchmaking

- Create `GKMatchRequest`
- Find players via `GKMatchmaker` or `GKMatchmakerViewController`
  - Accept direct invites by implementing the inviteHandler of `GKMatchmaker`
  - Optionally find nearby players
  - Optionally find players for a hosted match
- Exchange Data
  - Send data to other players
  - Optionally pick a player to act as server

Turn-Based Matchmaking

- Create `GKMatchRequest`
- Find players via `GKTurnBasedMatch` or `GKTurnBasedMatchmakerViewController`
  - Or load ongoing matches via `GKTurnBasedMatch`
- Determine current player
- Load and update match data
  - Match data is stored with the match and distributed to all players
- Advance the match to the next player

Documentation

- Game Center Overview
- Game Center Programming Guide
- GameKit Framework Reference

Multipeer Connectivity

- Connect to and exchange data with nearby devices
- Uses Wifi, peer-to-peer Wifi, or Bluetooth

Multipeer Connectivity Overview

- Create an `MCPeerID` for your device
- Create an `MCSession`
- Create an `MCNearbyServiceAdvertiser` or `MCAdvertiserAssistant` to advertise your device
- Create an `MCNearbyServiceBrowser` or `MCBrowserViewController` to find and connect to advertised devices
- Exchange data via the `MCSession` object

Multipeer Connectivity Example

```
// set up peer id
self.peerId = [[MCPeerID alloc] initWithDisplayName:[[UIDevice currentDevice] name]];

// set up session
self.session = [[MCSession alloc] initWithPeer:self.peerId];
self.session.delegate = self;

// set up advertising service
self.advertiser = [[MCAdvertiserAssistant alloc] initWithServiceType:MyServiceName
  discoveryInfo:nil session:self.session];
self.advertiser.delegate = self;
[self.advertiser start];

// set up peer browser
self.browserViewController = [[MCBrowserViewController alloc]
  initWithServiceType:MyServiceName session:self.session];
self.browserViewController.delegate = self;

[self presentViewController:self.browserViewController animated:YES completion:NULL];
```

Multipeer Connectivity Example (cont.)

```
// asynchronously send data to all peers
if (![self.session sendData:data
  toPeers:self.session.connectedPeers
  withMode:MCSessionSendDataReliable
  error:&error]) {
    // handle error
}

// open output stream to a specific peer
self.stream = [self.session
  startStreamWithName:MyStreamName toPeer:peer
  error:&error];
if (!self.stream) {
    // handle error
}
self.stream.delegate = self;
```

```
// receive data from peer
- (void)session:(MCSession *)session
  didReceiveData:(NSData *)data fromPeer:
  (MCPeerID *)peerID {
    // ...
}

// receive input stream from peer
- (void)session:(MCSession *)session
  didReceiveStream:(NSInputStream *)stream
  withName:(NSString *)streamName fromPeer:
  (MCPeerID *)peerID {
    self.inputStream = stream;
    self.inputStream.delegate = self;
}
```

Documentation

- Multipeer Connectivity Framework Reference

AirDrop

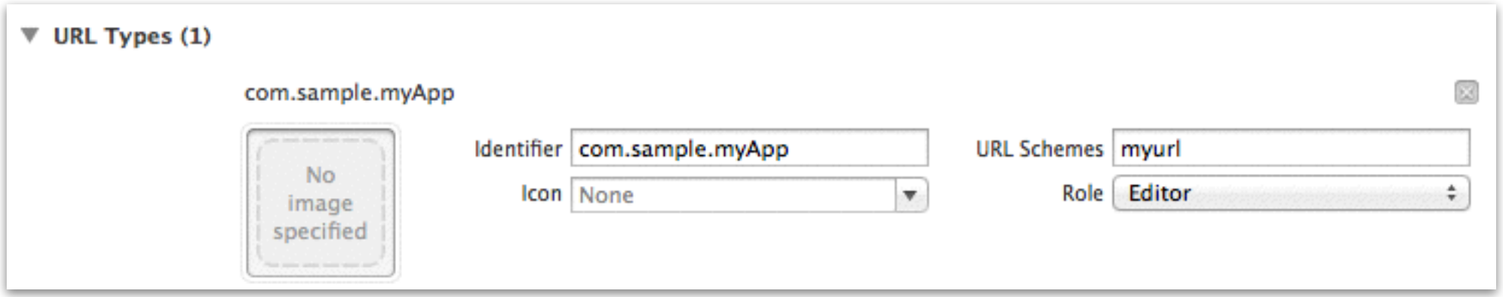
- Exchange data with nearby devices
- Uses Wifi, peer-to-peer Wifi, or Bluetooth

AirDrop Checklist

- Only supports transfer between iOS 7 devices
  - Enable AirDrop on the receiving device from the control center
- Create and configure `UIActivityViewController`
- Attach your content to be shared
  - Supports `NSString` and `UIImage`
  - Optionally use `UIActivityItemProvider` or `UIActivityItemSource`
- To accept shared content, register a custom UTI or URL scheme

AirDrop Example

```
// configure and display the activity view controller
// self.myContent implements the UIActivityItemSource protocol
UIActivityViewController *activityViewController = [[UIActivityViewController alloc]
  initWithActivityItems:@[self.myContent] applicationActivities:nil];

[self presentViewController:activityViewController animated:YES completion:nil];


// Handle opening URLs in the AppDelegate
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
  sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
    if (url) {
        // ...
    }
}
```

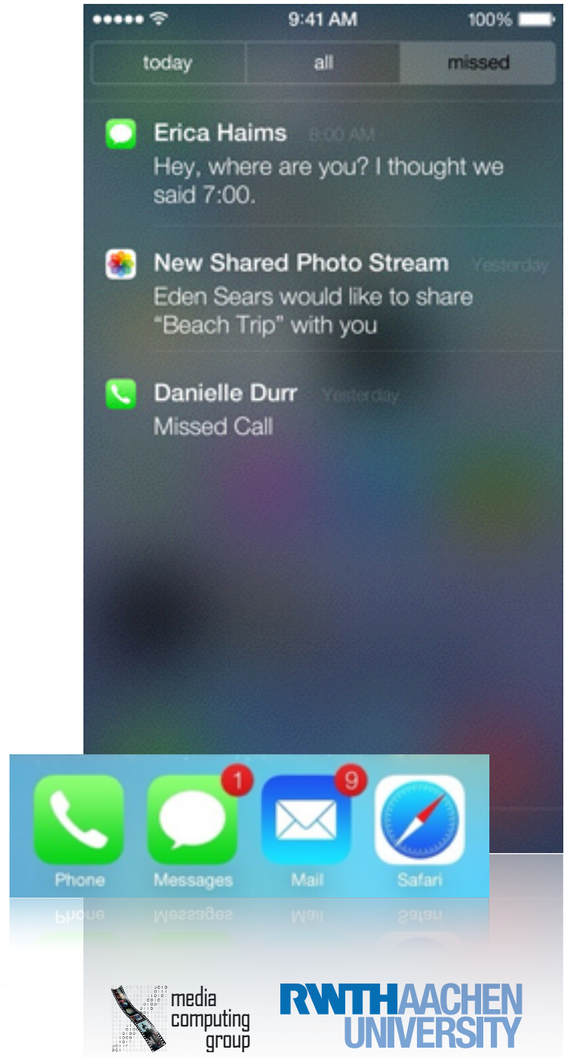


Documentation

- AirDrop Example Code

Local and Push Notifications

- Notify the user (and your app) about an external event
- Notifications are visualized in multiple ways:
  - Badge with a number
  - Alert
  - Sound
  - Notification Center
- Not used to transmit data!



Local Notifications

- Schedule a local notification
  - Create and configure `UILocalNotification`
  - Tell `UIApplication` to schedule the location notification
- Receive a local notification
  - Implement `UIAppDelegate`
  - Application is running:
    `application:didReceiveLocalNotification:`
  - Application is not running:
    `application:didFinishLaunchingWithOptions:`

Scheduling Local Notifications Example

```
UILocalNotification *notification = [UILocalNotification new];
notification.fireDate = [[NSDate alloc] initWithTimeIntervalSinceNow:10.];
notification.alertBody = @"Answer me";
notification.soundName = UILocalNotificationDefaultSoundName;
notification.applicationIconBadgeNumber = 1;
notification.userInfo = self.customInfo;

[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Receiving Local Notifications Example

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
  (NSDictionary *)launchOptions
{
    UILocalNotification *notification =
        launchOptions[UIApplicationLaunchOptionsLocalNotificationKey];
    if (notification) {
        [self handleNotification:notification];
    }
    return YES;
}

- (void)application:(UIApplication *)application didReceiveLocalNotification:
  (UILocalNotification *)notification;
{
    [self handleNotification:notification];
}
```

Remote Notifications

- Register for Remote Notifications
  - Tell `UIApplication` to register for remote notifications
  - Implement `UIApplicationDelegate` and send device token to your server
- Receiving Remote Notifications
  - Similar to local notifications (replace `Local` with `Remote`)

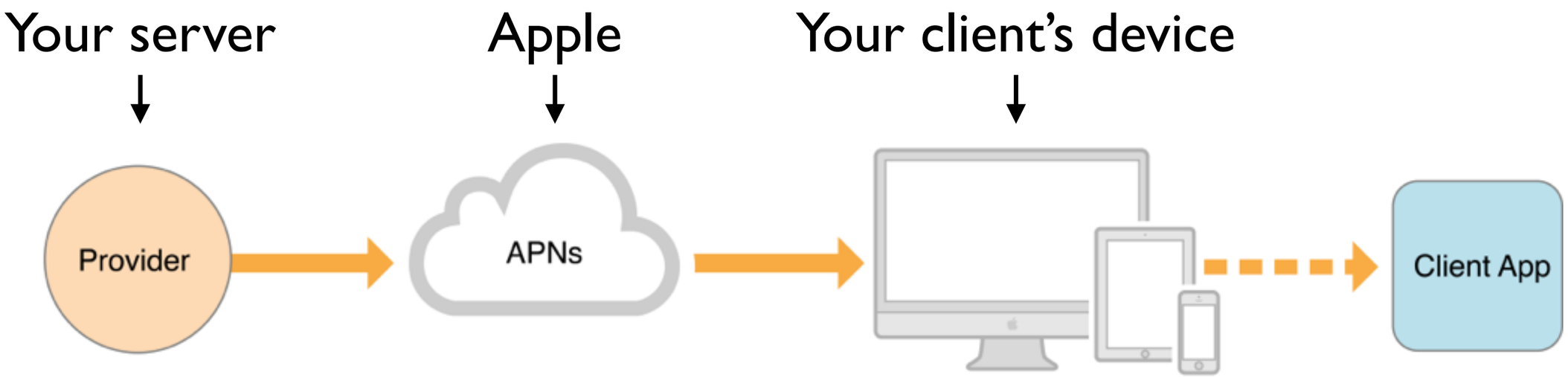
Registering for Remote Notifications Example

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
  (NSDictionary *)launchOptions
{
    // register for remote notifications
    [application registerForRemoteNotificationTypes:(UIRemoteNotificationTypeBadge |
        UIRemoteNotificationTypeSound)];

    return YES;
}

- (void)application:(UIApplication *)app
  didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)devToken {
    // send the device token to your server
}
```

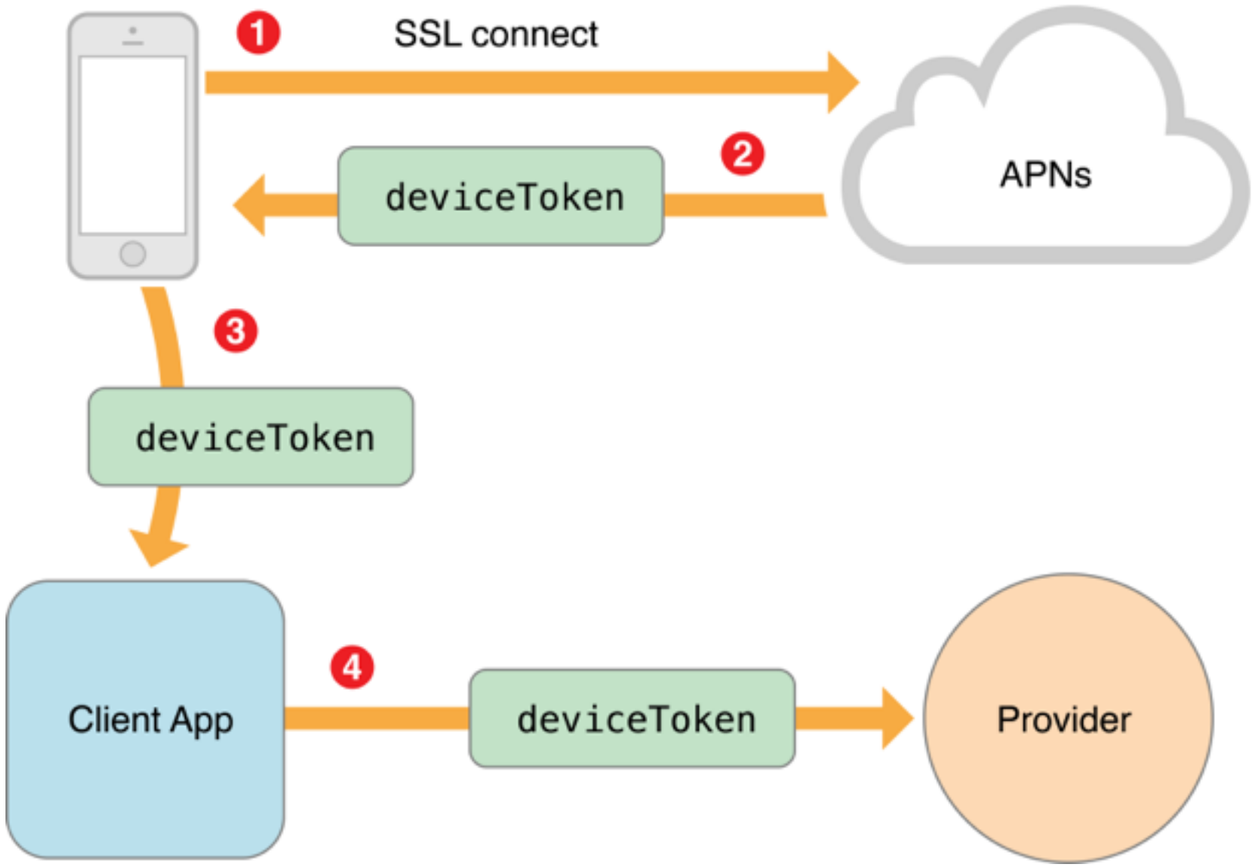
Receiving Remote Notifications Example

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
  (NSDictionary *)launchOptions
{
    NSDictionary *userInfo =
        launchOptions[UIApplicationLaunchOptionsRemoteNotificationKey];

    if (userInfo) {
        [self handleRemoteNotification:userInfo];
    }

    return YES;
}

- (void)application:(UIApplication *)application didReceiveRemoteNotification:
  (NSDictionary *)userInfo;
{
    [self handleRemoteNotification:userInfo];
}
```

Sending Remote Notifications



Device Token



Remote Notification Payload

- JSON (max. 256 bytes)
- `aps` defines notification type
  - alert message, badge count, sound
- Additional keys can be provided as needed
- Payload is available via the `userInfo` dictionary

```
{
    "aps" : {
        "alert" : "You got your emails.",
        "badge" : 9,
        "sound" : "bingbong.aiff"
    },
    "acme1" : "bar",
    "acme2" : 42
}
```

Implementing the Server

- Many packages available
  - Ruby: https://github.com/jpoz/APNS
  - NodeJS: https://github.com/argon/node-apn
  - PHP: https://code.google.com/p/apns-php
  - Python: https://github.com/djacobs/PyAPNs
  - …

URL Loading System

- Download content from a web server
  - Supports background download
- Communicate with a web service

URL Session

- Access content via HTTP
- Session Types:
  - Default: store content on the disk
  - Ephemeral: store content to memory
  - Background: process task in the background (when app is closed)
- Session Tasks:
  - Data tasks: exchange NSData objects with a web server (not available for background session type)
  - Download/Upload tasks: download or upload large files

Data Task Example

```
// create the data task to download an image
NSURLSession *session = [NSURLSession sharedSession];
NSURLSessionDataTask *task = [session dataTaskWithURL:self.imageURL
    completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

    // update the UI on the main thread
    [[NSOperationQueue mainQueue] addOperationWithBlock:^{
        self.imageView.image = [UIImage imageWithData:data];
    }];
}];

// start the data task
[task resume];
```

Background Download Task Example

```
// AppDelegate
- (void)createSession {
    if (!self.session) {
        NSURLSessionConfiguration *config = [NSURLSessionConfiguration backgroundSessionConfiguration:Id];
        _session = [NSURLSession sessionWithConfiguration:config delegate:self delegateQueue:nil];
    }
}

- (void)start {
    // create and start the download task
    [self createSession];
    [[self.session downloadTaskWithURL:DownloadURL] resume];
}

- (void)application:(UIApplication *)application handleEventsForBackgroundURLSession:(NSString *)identifier
  completionHandler:(void (^)())completionHandler {
    // recreate the session if necessary and store the completion handler
    [self createSession];
    self.completionHandler = completionHandler;
}

- (void)URLSession:(NSURLSession *)session downloadTask:(NSURLSessionDownloadTask *)downloadTask
  didFinishDownloadingToURL:(NSURL *)location {
    // update the UI, then create a new snapshot by calling the completion handler
    if (self.completionHandler) self.completionHandler();
}
```

Documentation

- URL Loading System Programming Guide

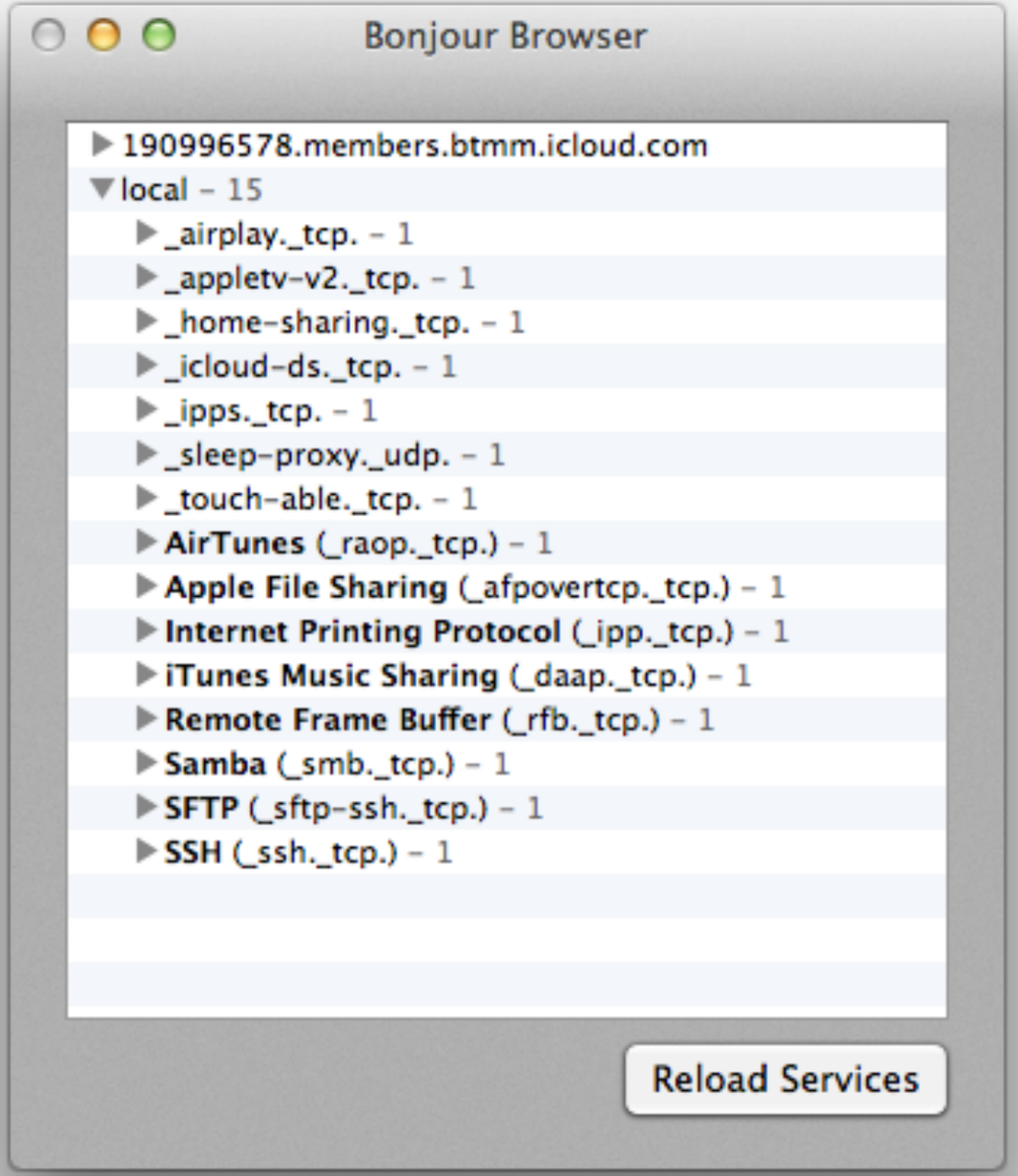
Bonjour

- Service discovery framework (zeroconf networking)
- Does not transfer data
- Also available for Unix/Linux & Windows
- Based on multicast DNS (mDNS)
  - Every client stores own DNS table
  - Service lookup over DNS request broadcast

Bonjour Service

- Publish service using `NSNetService`
  - Unique Service name
  - Service type and transport layer (“_http._tcp.”)
  - Registration domain (“local.”)
  - Port
- Discover services using `NSNetServiceBrowser`

Bonjour Browser



Bonjour Example

```
// setup and start net service
self.netService = [[NSNetService alloc] initWithDomain:@"local."
  type:@"_myservice._tcp" name:@"MyService" port:8080];
[self.netService publish];

// setup and start net service browser
self.netServiceBrowser = [NSNetServiceBrowser new];
self.netServiceBrowser.delegate = self;
[self.netServiceBrowser searchForServicesOfType:@"_myservice._tcp" inDomain:@"local."];

// delegate methods
- (void)netServiceBrowser:(NSNetServiceBrowser *)browser
  didFindService:(NSNetService *)netService moreComing:(BOOL)hasMore {
    // handle the discovered service
}

- (void)netServiceBrowser:(NSNetServiceBrowser *)browser
  didRemoveService:(NSNetService *)netService moreComing:(BOOL)hasMore {
    // handle the removed service
}
```

Documentation

- Bonjour for Developers
- Bonjour Overview
- NSNetServices Programming Guide
- CocoaEcho Example

Socket Networking

- Sockets are used to establish a connection between devices
  - BSD sockets
  - CFNetwork
- Streams are used to communicate through sockets
  - CFStream
  - NSStream

Socket Networking Checklist

- Include CFNetwork Framework
- Create listening socket on server
- Create read and write streams from client to server
- Create read and write streams from server to client
- Send streaming data and respond to stream events

Server Example
Create socket

```
// create new socket
CFSocketContext socketContext = {0, (__bridge void *) self, NULL, NULL, NULL};
socket = CFSocketCreate(kCFAllocatorDefault, AF_INET,  SOCK_STREAM, 0,
  kCFSocketAcceptCallBack, &AcceptCallback, &socketContext);
if (!socket) {
  // handle error
}

// configure socket to allow wildcard address reuse
static const int yes = 1;
setsockopt(CFSocketGetNative(socket), SOL_SOCKET, SO_REUSEADDR, (const void *) &yes,
sizeof(yes));
```

Server Example

Bind socket to address and port and attach to run loop

```
// set up socket as a listening socket
struct sockaddr_in addr;
memset(&addr, 0, sizeof(addr));
addr.sin_len = sizeof(addr);
addr.sin_family = AF_INET; // TCP socket
addr.sin_port = htons(0); // automatic port
addr.sin_addr.s_addr = htonl(INADDR_ANY); // bind to all interfaces

NSData *addrData = [NSData dataWithBytes:&addr length:sizeof(addr)];
if (kCFSocketSuccess != CFSocketSetAddress(socket, (__bridge CFDataRef)addrData)) {
  // handle error
}

// attach the socket to the current run loop
CFRunLoopSourceRef source = CFSocketCreateRunLoopSource(kCFAllocatorDefault, socket,
0);
CFRunLoopAddSource(CFRunLoopGetCurrent(), source, kCFRunLoopCommonModes);
CFRelease(source);
```

Server Example
Incoming connection callback

```
static void AcceptCallback(CFSocketRef socket, CFSocketCallBackType type, CFDataRef address,
  const void *data, void *info) {
   CFSocketNativeHandle nativeSocket = *(CFSocketNativeHandle *)data;
    CFReadStreamRef readStream = NULL; CFWriteStreamRef writeStream = NULL;

    // create read and write streams
    CFStreamCreatePairWithSocket(kCFAllocatorDefault, nativeSocket, &readStream, &writeStream);
    if (!readStream || !writeStream) {
        close(nativeSocket);
        return;
    }

    // configure streams
    CFReadStreamSetProperty(readStream, kCFStreamPropertyShouldCloseNativeSocket, kCFBooleanTrue);
    CFWriteStreamSetProperty(writeStream, kCFStreamPropertyShouldCloseNativeSocket, kCFBooleanTrue);

    // create connection object
    [(__bridge MyServer *)info createConnectionWithInputStream:(__bridge NSInputStream *)readStream
      outputStream:(__bridge NSOutputStream *)writeStream];

    CFRelease(readStream);
    CFRelease(writeStream);
}
```

Client Example

```
[netService getInputStream:&_inputStream outputStream:&_outputStream];

// configure and open the input stream
self.inputStream.delegate = self;
[self.inputStream scheduleInRunLoop:[NSRunLoop currentRunLoop]
forMode:NSDefaultRunLoopMode];
[self.inputStream open];

// configure and open the output stream
self.outputStream.delegate = self;
[self.outputStream scheduleInRunLoop:[NSRunLoop currentRunLoop]
forMode:NSDefaultRunLoopMode];
[self.outputStream open];


// alternative: CFStreamCreatePairWithSocketToHost()
```

Stream Events Example

```
// NSStream delegate method
- (void)stream:(NSStream *)stream handleEvent:(NSStreamEvent)event {
    uint8_t buffer[size];
    NSInteger length;

    // there is incoming data to be collected
    if (event == NSStreamEventHasBytesAvailable) {
        length = [(NSInputStream *)stream read:(uint8_t *)buffer maxLength:size];
        if (length > 0) {
            // process this chunk of data
        }
    }

    // there is space in the buffer for more outgoing data
    else if (event == NSStreamEventHasSpaceAvailable) {
        // copy next chunk of data into the buffer
        length = [(NSOutputStream *)stream write:(uint8_t *)buffer maxLength:size];
    }
}
```

Data Chunking

- Transferred data might be bigger than the stream buffer
- Data must be sent and arrives in chunks
- Chunking strategies:
  - Transmit a special character (\0) at the end of the data
  - Transmit length of the data before the data

Documentation

- Networking Programming Topics
- Stream Programming Guide
- BSD Sockets
- CocoaEcho Example

AsyncNetwork

- Automatic server discovery and connection (Bonjour)
- Automatic Object Encoding (NSCoding)
- Simple communication (read, write, request, command byte)
- Broadcasting
- Based on GCDAsyncSocket

AsyncNetwork Client and Server

```
// Setup the server
- (void)setupServer {
    self.server = [AsyncServer new];
    self.server.serviceName = @"My Service";
    self.server.delegate = self;
    [self.server start];
}

// Client did connect
- (void)server:(AsyncServer *)theServer didConnect:(AsyncConnection *)connection {
    [self.server sendObject:@"Hello Client"];
}

// Setup client
- (void)setupClient {
    self.client = [AsyncClient new];
    self.client.delegate = self;
    [self.client start];
}

// Client did receive message
- (void)client:(AsyncClient *)theClient didReceiveCommand:(AsyncCommand)command
    object:(id)object connection:(AsyncConnection *)connection {
     NSLog(@"%@", object);
}
```

AsyncNetwork Requests

```
// send a request to the server
[AsyncRequest fireRequestWithHost:@"192.168.0.1"
                             port:10000
                          command:0
                           object:@"Hello"
                    responseBlock:^(id response, NSError *error)
{
    if (error) return;
    NSLog(@"%@", response);
}];
```

AsyncNetwork Broadcasting

```
- (void)setupBroadcaster {
    // Create and start the broadcaster
    self.broadcaster = [AsyncBroadcaster new];
    self.broadcaster.port = 10000;
    self.broadcaster.delegate = self;
    [self.broadcaster start];
}

- (void)broadcast {
    // Encode the string message
    NSString *message = @"Hello World\n";
    NSData *encodedMessage = [message dataUsingEncoding:NSUTF8StringEncoding];

    // Broadcast the encoded message
    [self.broadcaster broadcast:encodedMessage];
}

- (void)broadcaster:(AsyncBroadcaster *)theBroadcaster
      didReceiveData:(NSData *)data
            fromHost:(NSString *)host {
    // Decode the message
    NSString *message;
    message = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
    NSLog(@"%@", message);
}
```

AsyncNetwork

- https://github.com/jdiehl/async-network