Simplify Networking with Bonjour

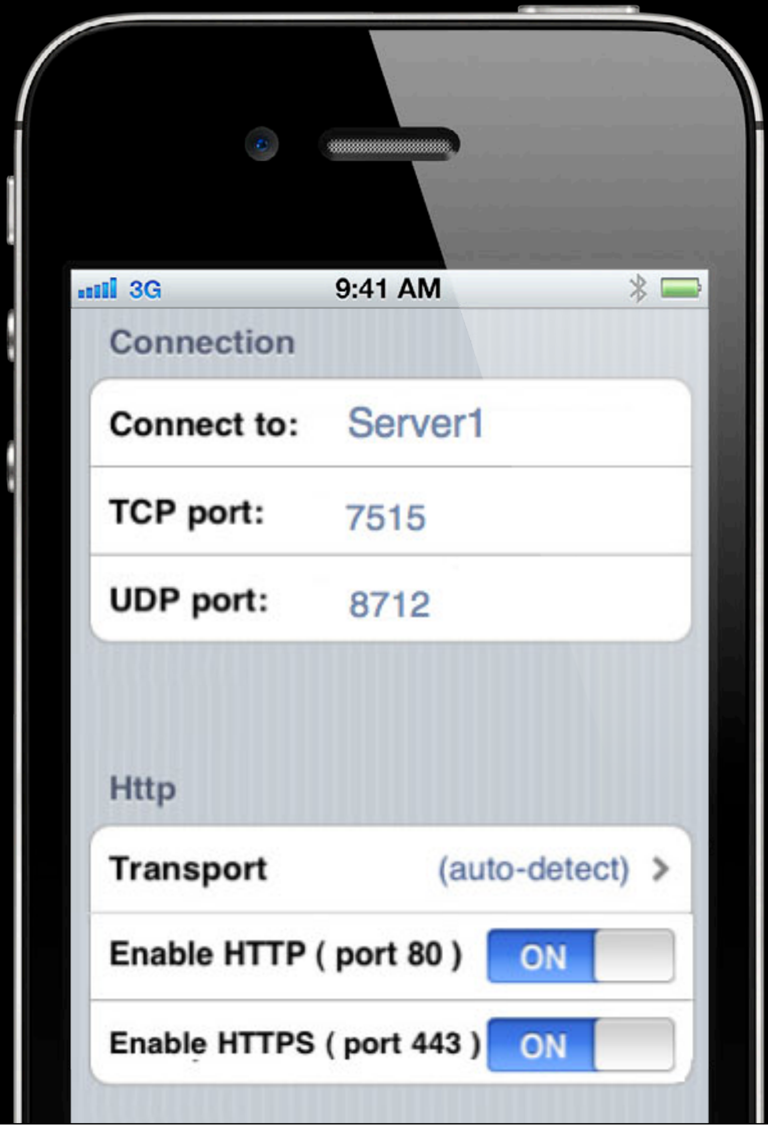
Session 707

Dr Stuart Cheshire
Bonjour Architect

Rory McGuire
Senior Packet Wrangler

Introduction

- Bonjour Overview
  - Improve your user experience
  - Ecosystem
  - Technology
  - Three operations
- Demo
- Tactical and practical
  - Coding explained
  - Tips and reminders



Connection

Connect to: Server1

TCP port: 7515

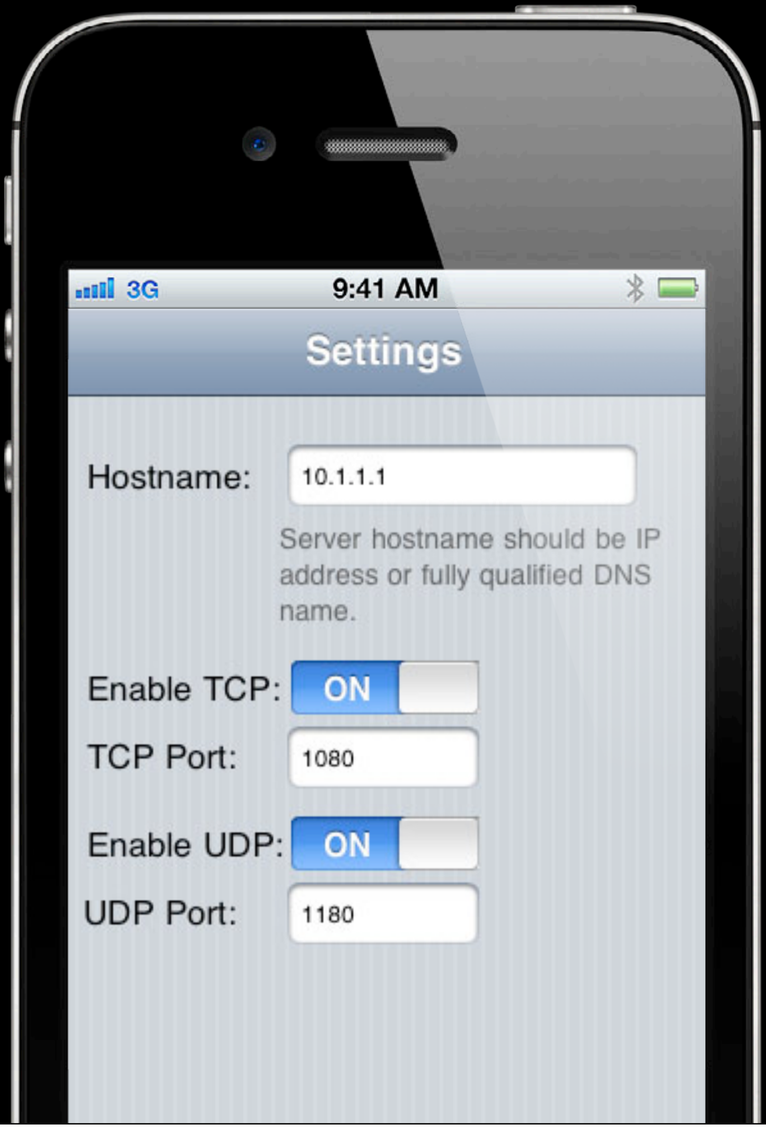
UDP port: 8712

Http

Transport (auto-detect) >

Enable HTTP (port 80) ON

Enable HTTPS (port 443) ON



Settings

Hostname: 10.1.1.1

Server hostname should be IP address or fully qualified DNS name.

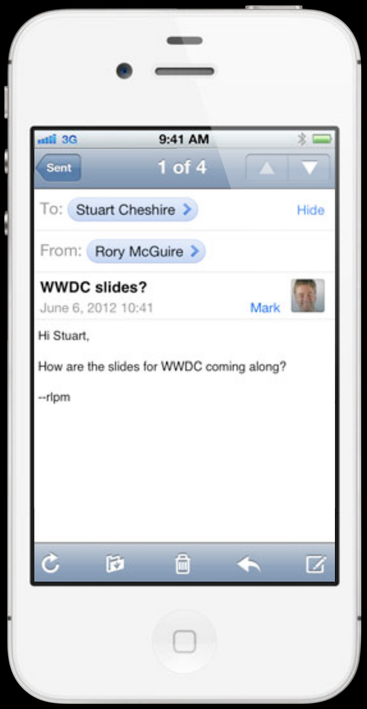
Enable TCP: ON

TCP Port: 1080

Enable UDP: ON

UDP Port: 1180

Improve Your User Experience



To: Stuart Cheshire

From: Rory McGuire

WWDC slides?

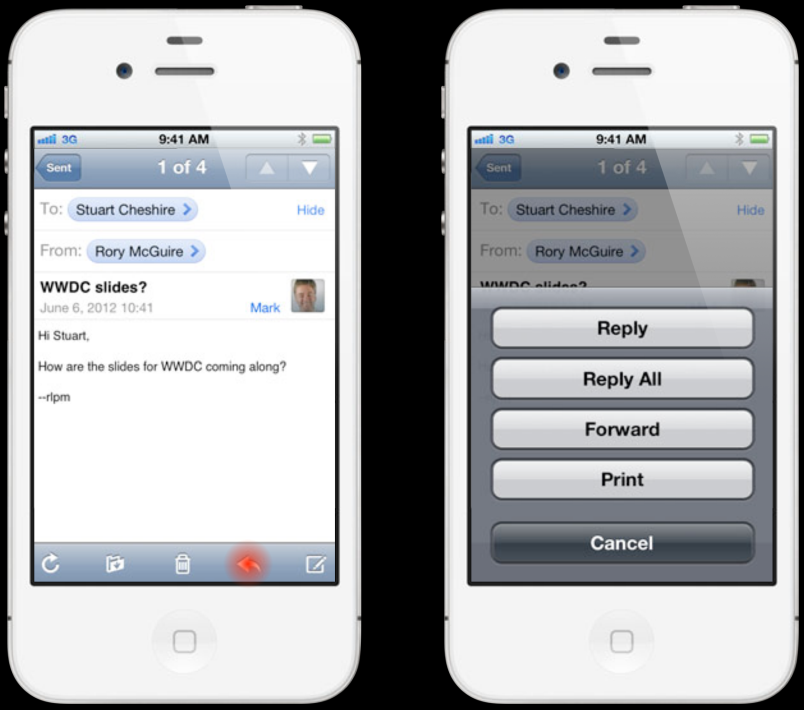
June 6, 2012 10:41

Hi Stuart,

How are the slides for WWDC coming along?

--rlpm

Improve Your User Experience



To: Stuart Cheshire

From: Rory McGuire

WWDC slides?

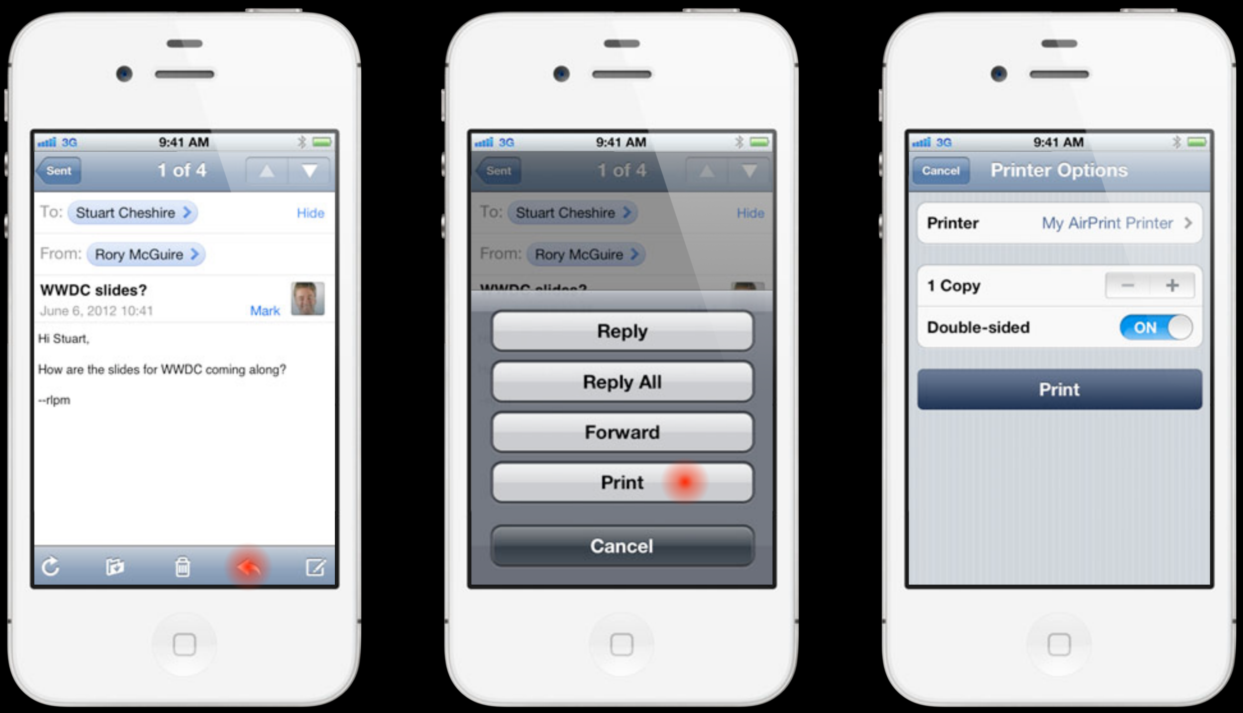
June 6, 2012 10:41

Hi Stuart,

How are the slides for WWDC coming along?

--rlpm

Improve Your User Experience



To: Stuart Cheshire

From: Rory McGuire

WWDC slides?

June 6, 2012 10:41

Hi Stuart,

How are the slides for WWDC coming along?

--rlpm

Printer: My AirPrint Printer

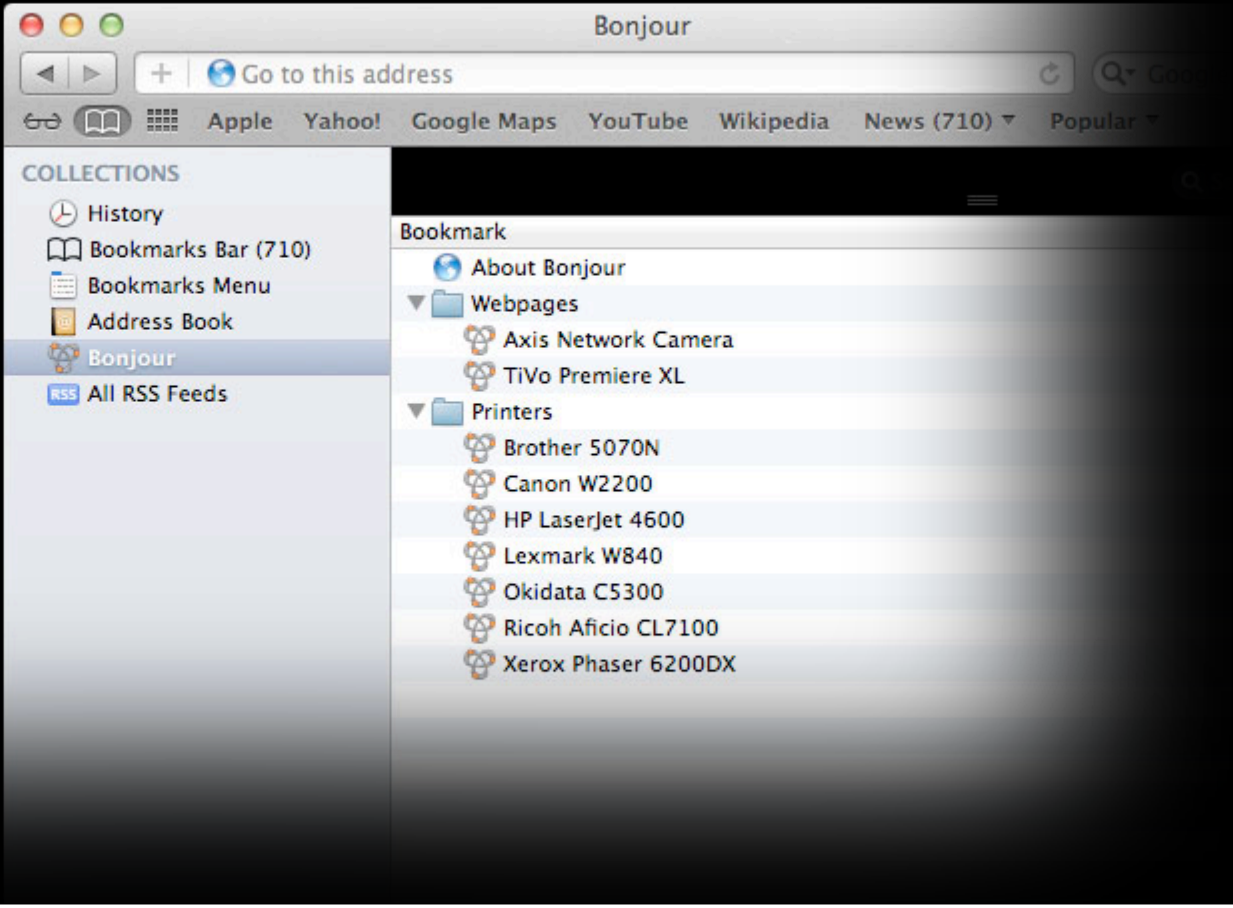
1 Copy

Double-sided: ON

AXIS
COMMUNICATIONS

Bonjour in Safari

Press Command-Option-B to show COLLECTIONS

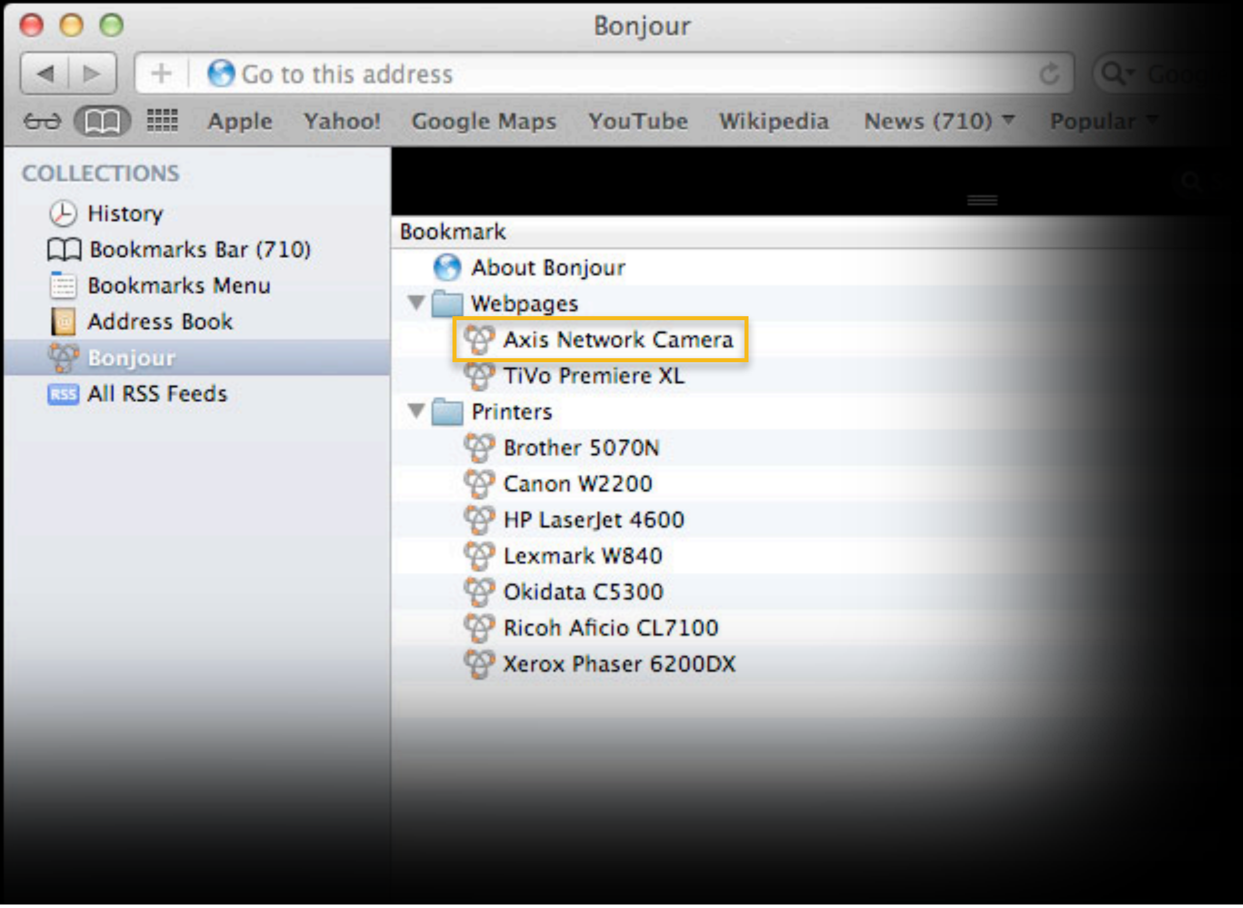


Bookmark

- About Bonjour
- Webpages
  - Axis Network Camera
  - TiVo Premiere XL
- Printers
  - Brother 5070N
  - Canon W2200
  - HP LaserJet 4600
  - Lexmark W840
  - Okidata C5300
  - Ricoh Aficio CL7100
  - Xerox Phaser 6200DX

Bonjour in Safari

Press Command-Option-B to show COLLECTIONS



COLLECTIONS

- History
- Bookmarks Bar (710)
- Bookmarks Menu
- Address Book
- Bonjour
- All RSS Feeds

Bookmark

- About Bonjour
- Webpages
  - Axis Network Camera
  - TiVo Premiere XL
- Printers
  - Brother 5070N
  - Canon W2200
  - HP LaserJet 4600
  - Lexmark W840
  - Okidata C5300
  - Ricoh Aficio CL7100
  - Xerox Phaser 6200DX

Bonjour Ecosystem

Devices and platforms

Bonjour Ecosystem

Devices and platforms

Bonjour Ecosystem

Devices and platforms

Bonjour Ecosystem

Devices and platforms

Bonjour Ecosystem

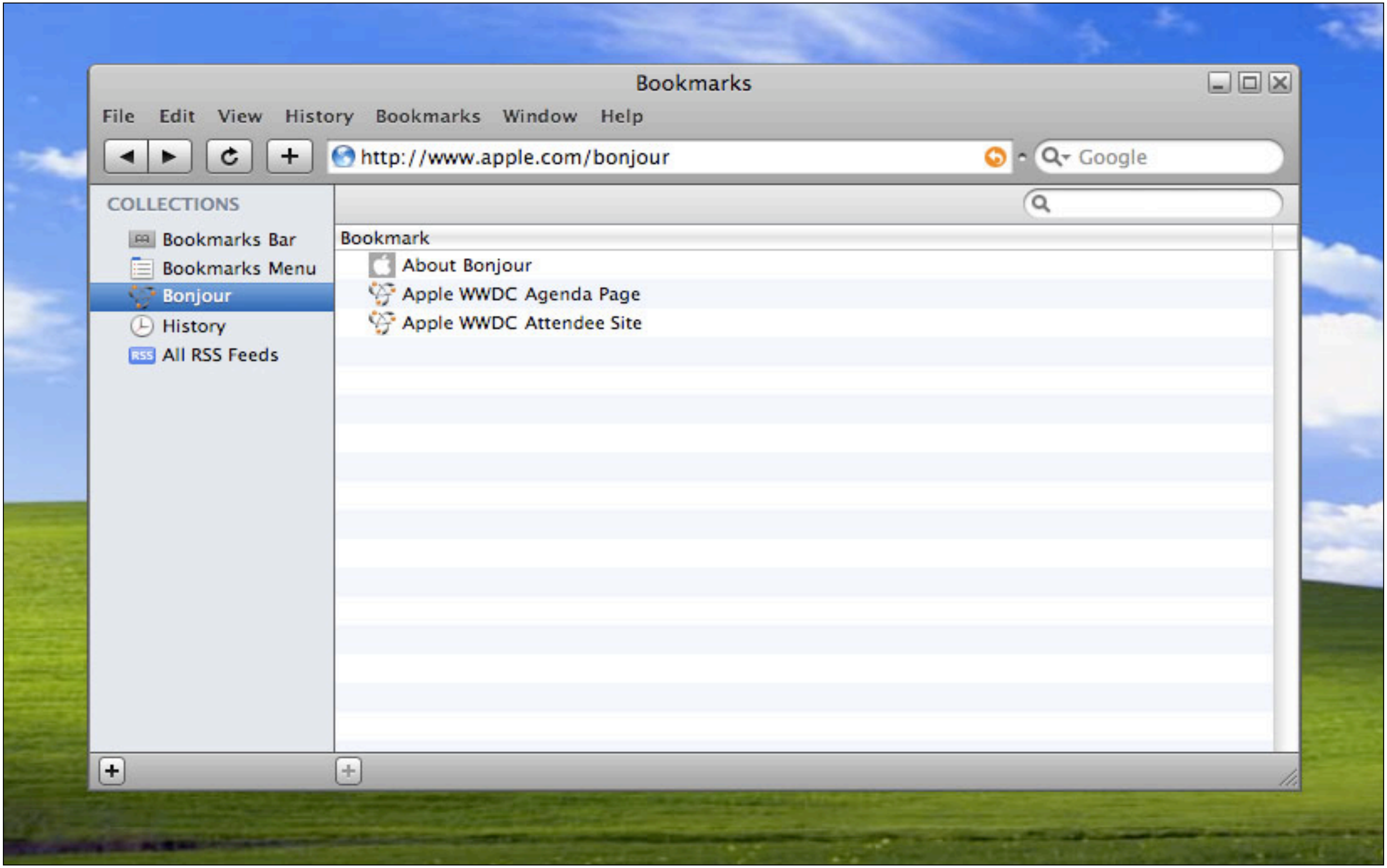
Devices and platforms

Bonjour Ecosystem

Devices and platforms

Bonjour Ecosystem

Devices and platforms

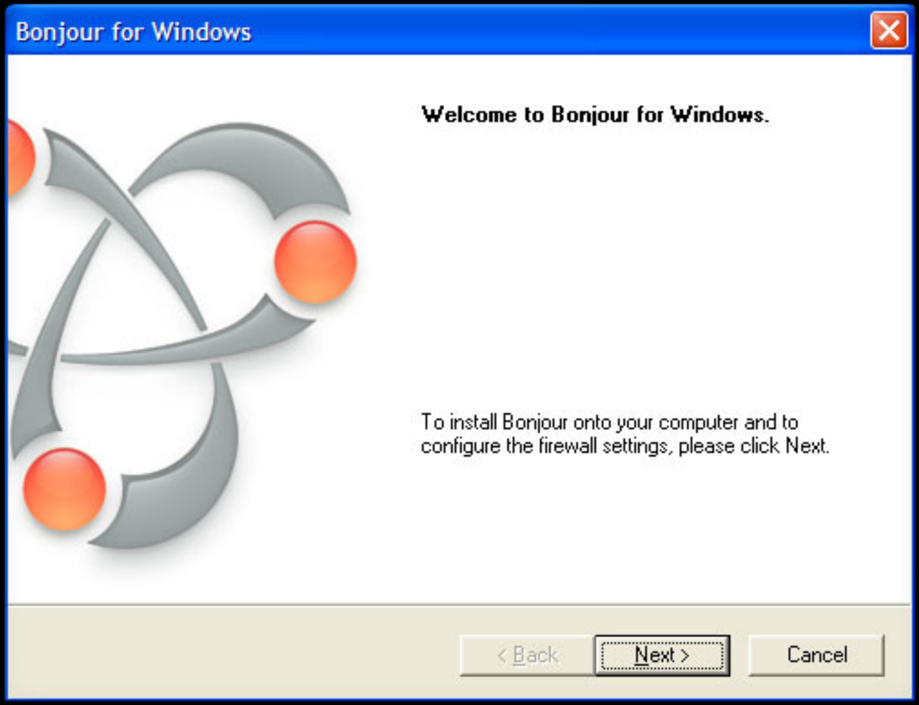


http://www.apple.com/bonjour

Bookmark
About Bonjour
Apple WWDC Agenda Page
Apple WWDC Attendee Site

Bundle Bonjour for Windows with Your App

No license fee



Welcome to Bonjour for Windows.

To install Bonjour onto your computer and to configure the firewall settings, please click Next.

Technology

- Link-local addressing
    - IPv4 (RFC 3927)
    - IPv6 (RFC 2462)
- Multicast DNS
    - http://www.multicastdns.org
- DNS Service Discovery
    - Link-local and wide-area
    - http://www.dns-sd.org

Bonjour Three Operations

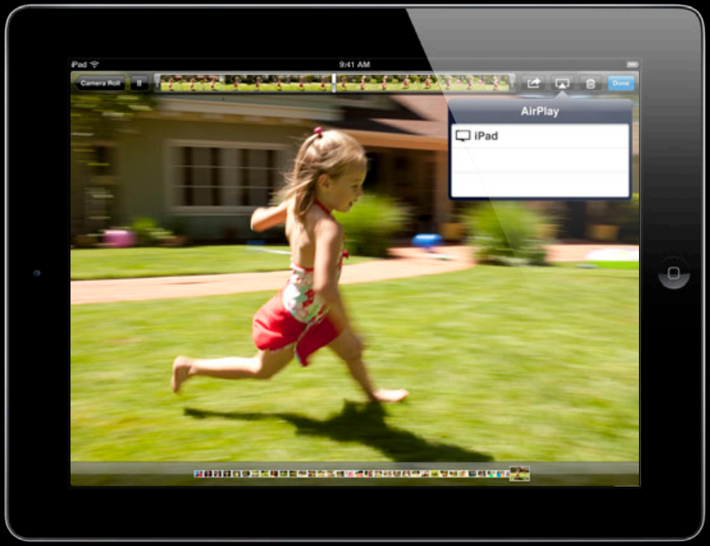
Register

Bonjour Three Operations

Register

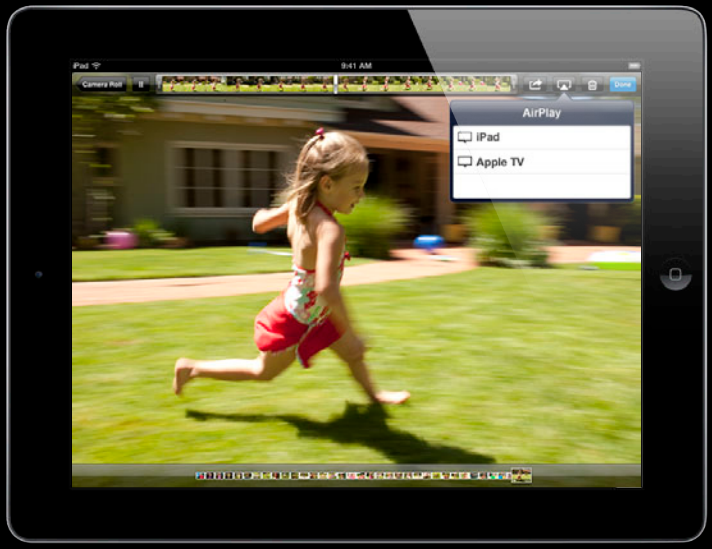
Bonjour Three Operations

Browse



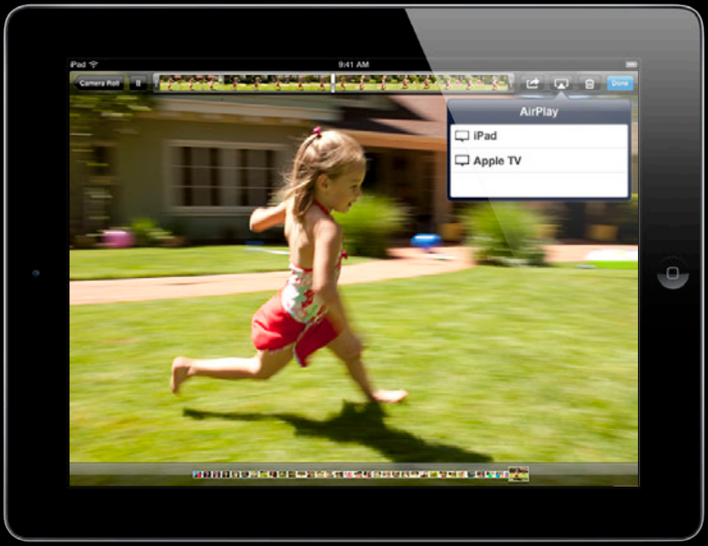
Bonjour Three Operations

Browse



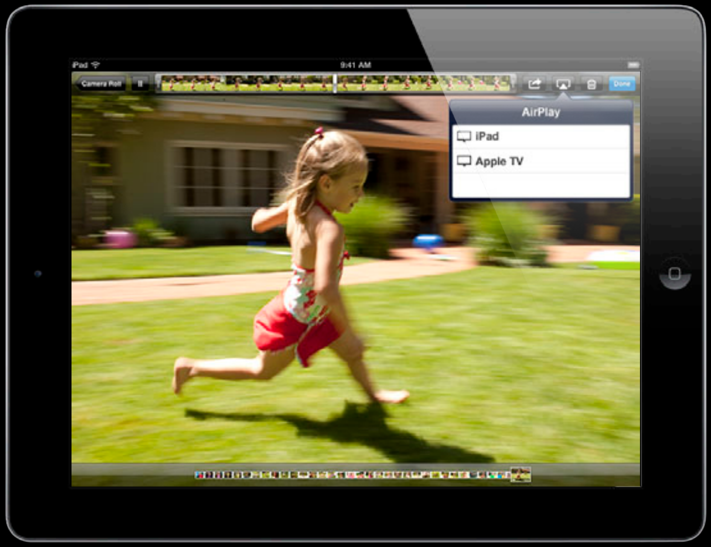
Bonjour Three Operations

Resolve



Bonjour Three Operations

Resolve



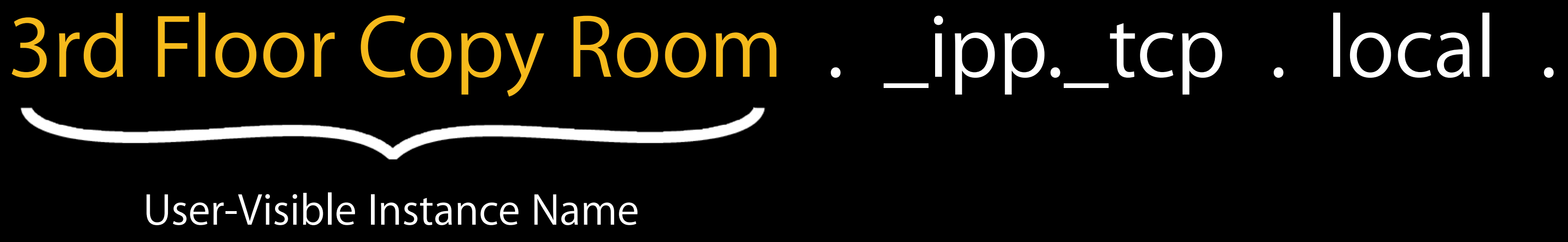
Components of Service Name

3rd Floor Copy Room._ipp._tcp.local.

Components of Service Name

3rd Floor Copy Room . _ipp._tcp . local .

Components of Service Name



3rd Floor Copy Room . _ipp._tcp . local .

User-Visible Instance Name

Components of Service Name

3rd Floor Copy Room . _ipp._tcp . local .

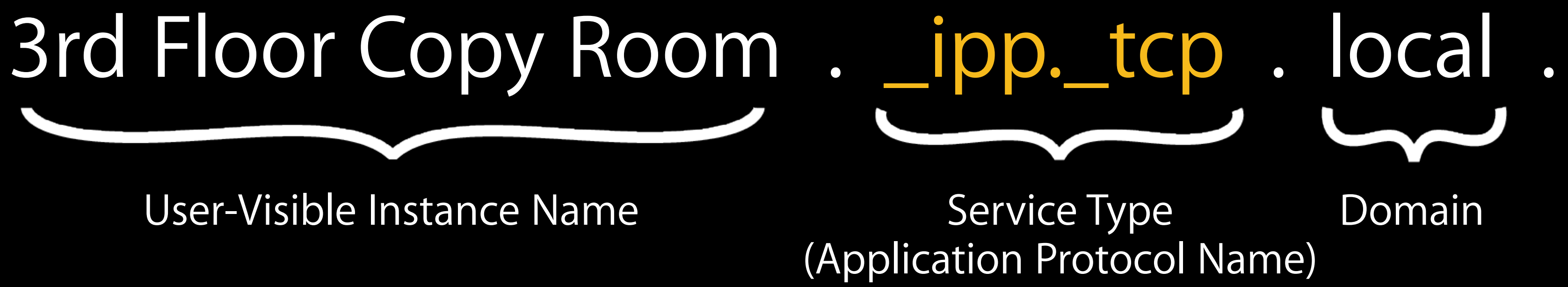
Service Type
(Application Protocol Name)

Components of Service Name

3rd Floor Copy Room . _ipp._tcp . local .

Domain

Components of Service Name



3rd Floor Copy Room . _ipp._tcp . local .

User-Visible Instance Name

Service Type
(Application Protocol Name)

Domain

Service Types

- Unique identifier string for every different service type
    - Maximum 15 characters
    - US-ASCII, letters, digits and hyphens
- Identifier string signifies
    - What the service does
    - How it does it — i.e. what on-the-wire protocol it uses

Service Types

IANA manages registry of unique service type strings

- RFC 6335 “IANA Procedures for the Management of the Service Name and Transport Protocol Port Number Registry”
- IANA list of assigned service type strings
  - http://www.iana.org/assignments/service-names-port-numbers
- Applying for your own is easy (and free)
  - http://www.iana.org/form/ports-services
- Before shipping, register your unique service type
  - So you don’t accidentally pick “daap”
    for “Don’s Action Adventure Playground”

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
    - Retransmission. How much? How often?

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
    - Retransmission. How much? How often?
- Efficiency

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
    - Retransmission. How much? How often?
- Efficiency
    - Known Answer lists

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
    - Retransmission. How much? How often?
- Efficiency
    - Known Answer lists
- Disconnect/Reconnect

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
  - Retransmission. How much? How often?
- Efficiency
  - Known Answer lists
- Disconnect/Reconnect
  - Monitor network configuration changes

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
  - Retransmission. How much? How often?
- Efficiency
  - Known Answer lists
- Disconnect/Reconnect
  - Monitor network configuration changes
  - Switch Wi-Fi base station

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
  - Retransmission. How much? How often?
- Efficiency
  - Known Answer lists
- Disconnect/Reconnect
  - Monitor network configuration changes
  - Switch Wi-Fi base station
- Sleep/Wake

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
    - Retransmission. How much? How often?
- Efficiency
    - Known Answer lists
- Disconnect/Reconnect
    - Monitor network configuration changes
    - Switch Wi-Fi base station
- Sleep/Wake
    - Send Goodbye Packets on sleep, Reannounce on wake

Why Not Invent Your Own Discovery Protocol?

After all, how hard could it be?

- Packet loss
    - Retransmission. How much? How often?
- Efficiency
    - Known Answer lists
- Disconnect/Reconnect
    - Monitor network configuration changes
    - Switch Wi-Fi base station
- Sleep/Wake
    - Send Goodbye Packets on sleep, Reannounce on wake
    - … or register with Sleep Proxy if present

Demo

Dr Stuart Cheshire
Bonjour Architect

Cross-Link Proxies

ARUBA® networks

Aerohive™ NETWORKS

XIRRUS® High Performance Wireless Networks

Why Not Invent Your Own Discovery Protocol?

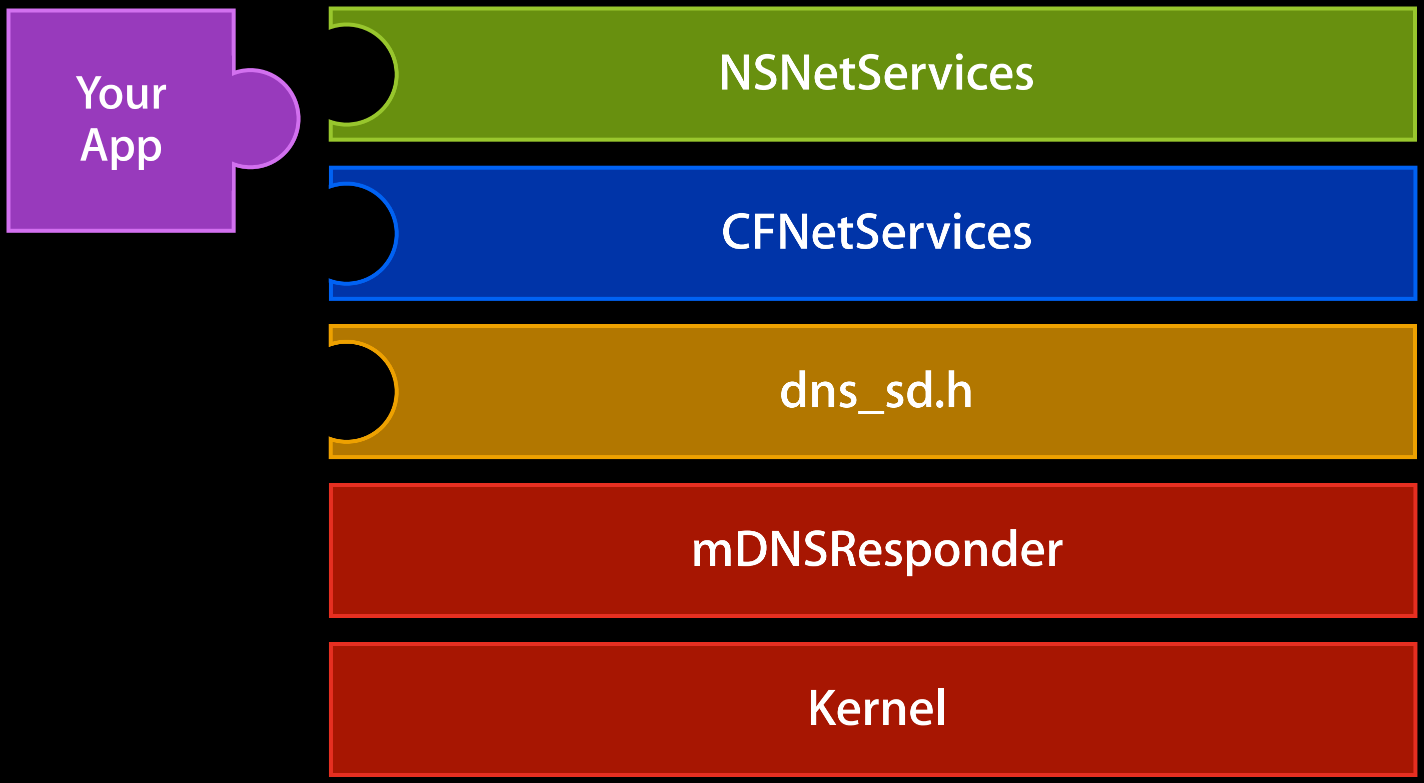
After all, how hard could it be?

- Packet loss
  - Retransmission. How much? How often?
- Efficiency
  - Known Answer lists
- Disconnect/Reconnect
  - Monitor network configuration changes
  - Switch Wi-Fi base station
- Sleep/Wake
  - Send Goodbye Packets on sleep, Reannounce on wake
  - … or register with Sleep Proxy if present
- Cross-subnet proxying

Bonjour — Tactical and Practical

Rory McGuire
Senior Packet Wrangler

Bonjour API Architecture



Your App

NSNetServices

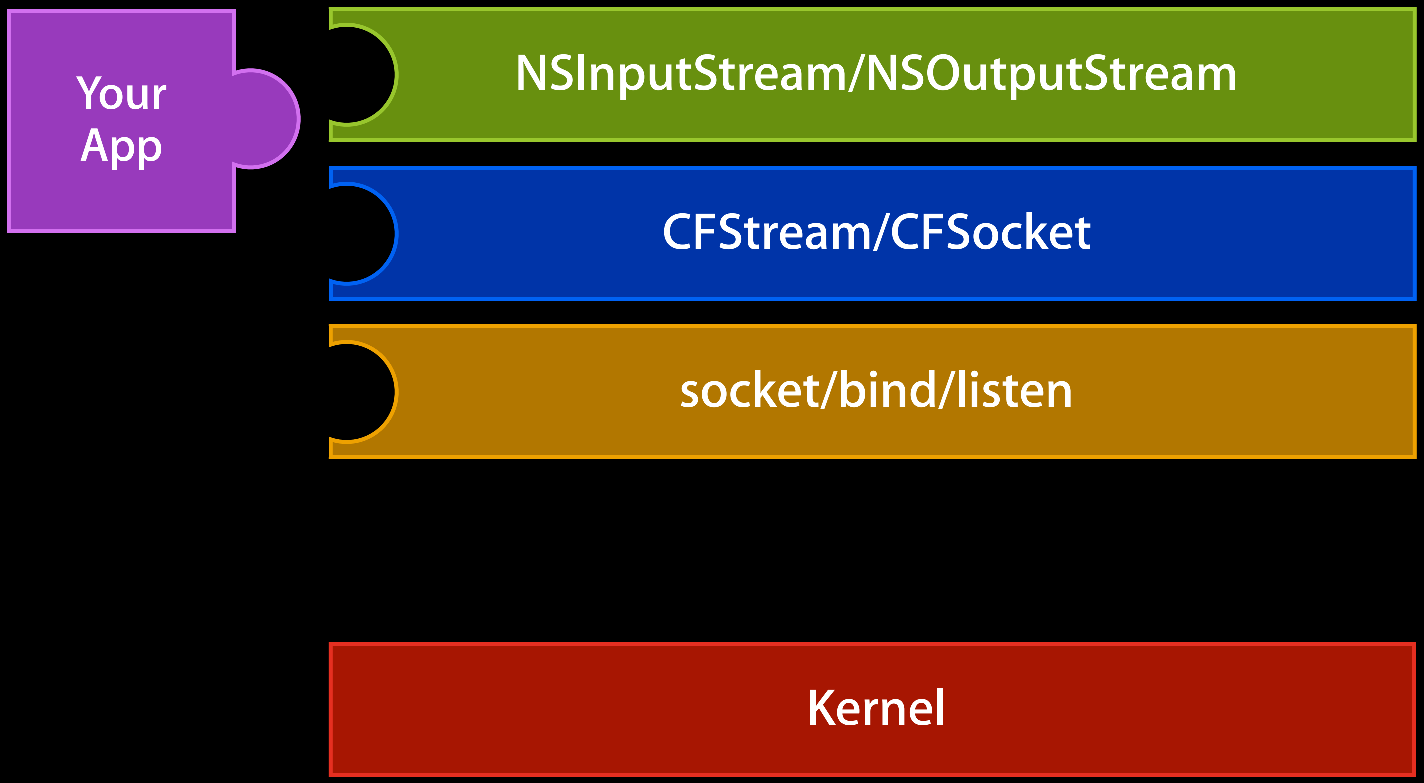
CFNetServices

dns_sd.h

mDNSResponder

Kernel

TCP API Architecture



Your App

NSInputStream/NSOutputStream

CFStream/CFSocket

socket/bind/listen

Kernel

Server Flow: Register

`socket/bind/listen`

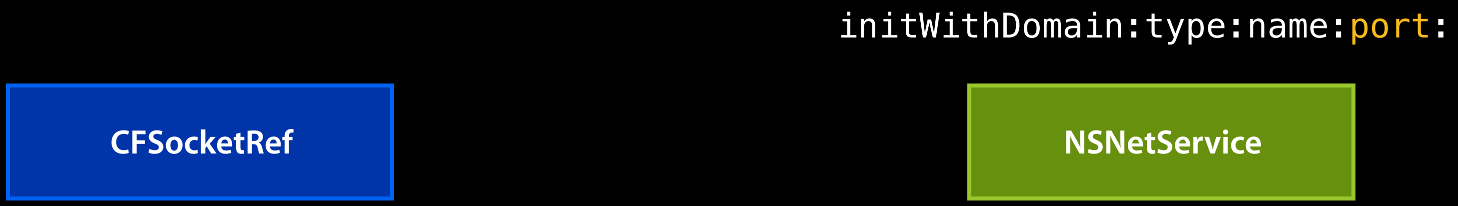
int fd

Server Flow: Register

`CFSocketCreateWithNative`

CFSocketRef

Server Flow: Register

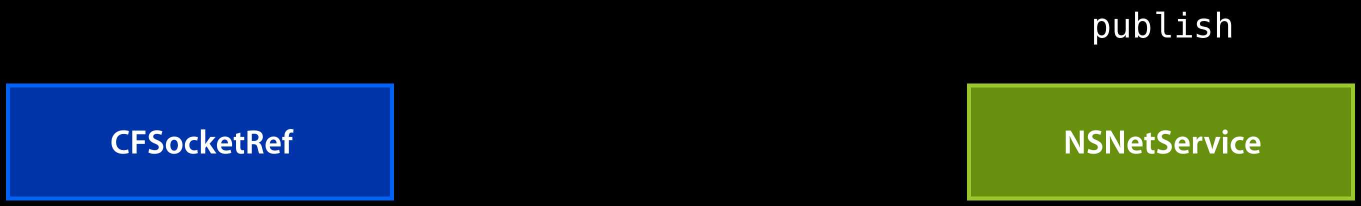


`initWithDomain:type:name:port:`

CFSocketRef

NSNetService

Server Flow: Register



CFSocketRef

publish

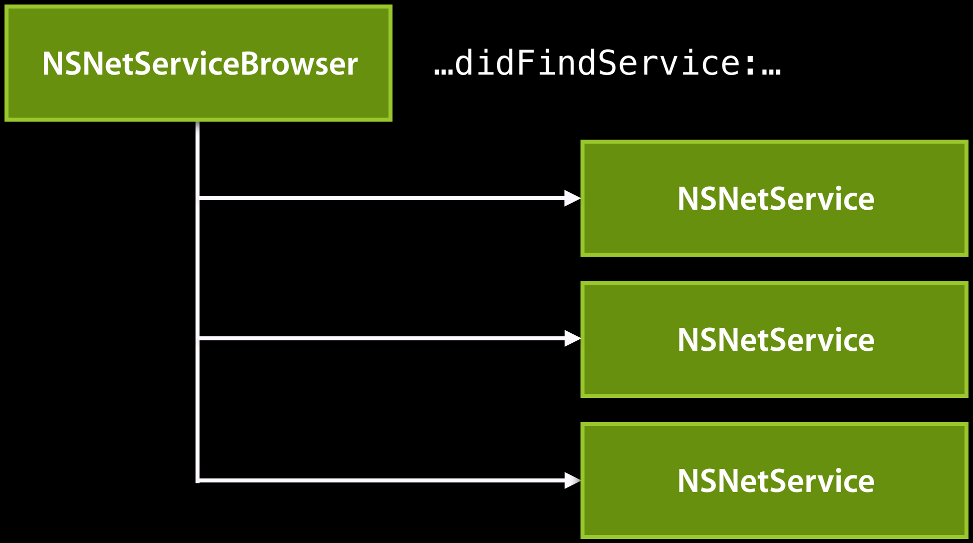
NSNetService

Client Flow: Browse

```
searchForServicesOfType:inDomain:
```

NSNetServiceBrowser

Client Flow: Browse



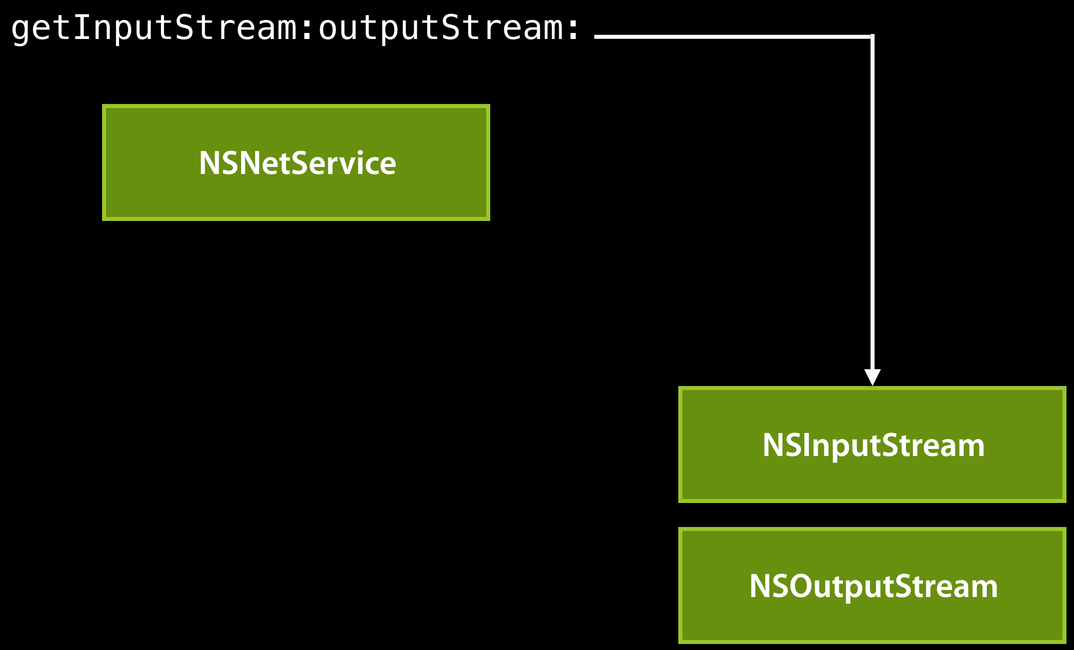
Client Flow: User Selects a Service Instance

NSNetService

NSNetService

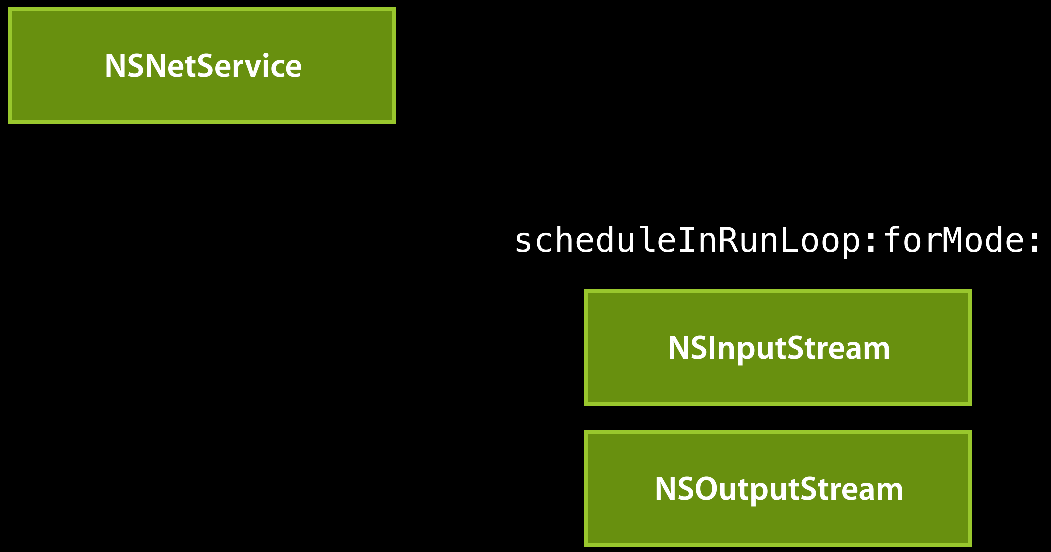
NSNetService

Client Flow: Connect



`getInputStream:outputStream:`

Client Flow: Connect



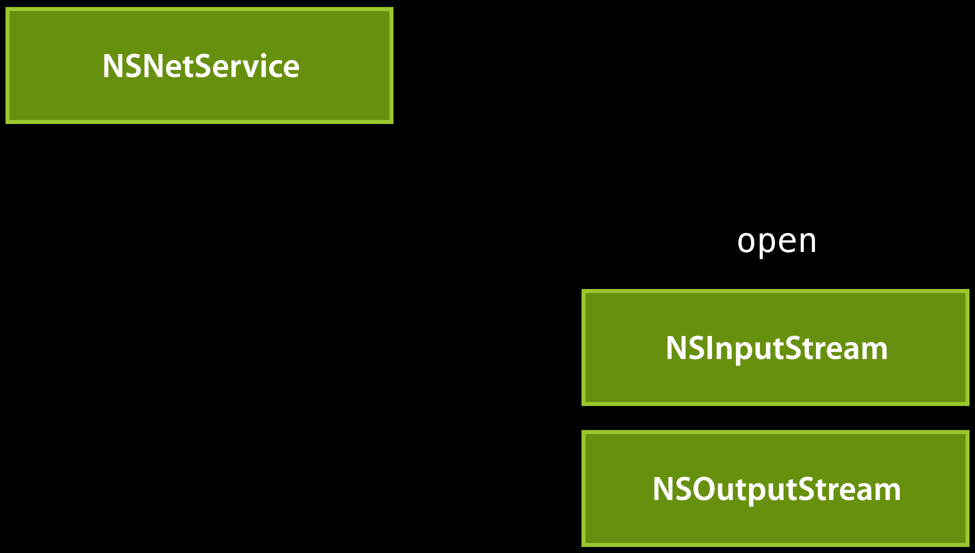
NSNetService

`scheduleInRunLoop:forMode:`

NSInputStream

NSOutputStream

Client Flow: Connect



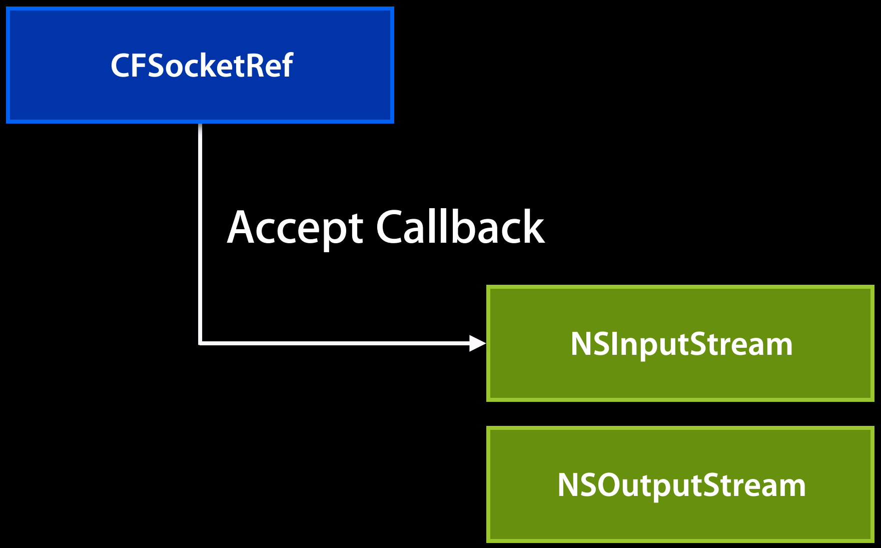
NSNetService

open

NSInputStream

NSOutputStream

Server Flow: Accept



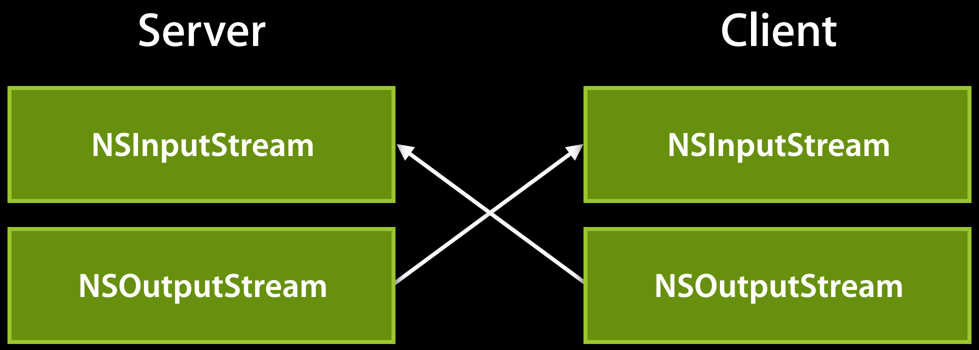
CFSocketRef

Accept Callback

NSInputStream

NSOutputStream

Connected



Server

NSInputStream

NSOutputStream

Client

NSInputStream

NSOutputStream

Server: bind/listen IPv4

```
int fd4 = socket(AF_INET, SOCK_STREAM, 0);

struct sockaddr_in sin;
memset(&sin, 0, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_len    = sizeof(sin);
sin.sin_port   = 0;

int err = bind(fd4, (const struct sockaddr *) &sin, sin.sin_len);

socklen_t addrLen = sizeof(sin);
err = getsockname(fd4, (struct sockaddr *) &sin, &addrLen);

err = listen(fd4, 5);
```

Server: bind/listen IPv4

```
int fd4 = socket(AF_INET, SOCK_STREAM, 0);

struct sockaddr_in sin;
memset(&sin, 0, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_len    = sizeof(sin);
sin.sin_port   = 0;

int err = bind(fd4, (const struct sockaddr *) &sin, sin.sin_len);

socklen_t addrLen = sizeof(sin);
err = getsockname(fd4, (struct sockaddr *) &sin, &addrLen);

err = listen(fd4, 5);
```

Server: bind/listen IPv6

```
int fd6 = socket(AF_INET6, SOCK_STREAM, 0);

int one = 1;
err = setsockopt(fd6, IPPROTO_IPV6, IPV6_V6ONLY, &one, sizeof(one));

struct sockaddr_in6 sin6;
memset(&sin6, 0, sizeof(sin6));
sin6.sin6_family = AF_INET6;
sin6.sin6_len    = sizeof(sin6);
sin6.sin6_port   = sin.sin_port;

err = bind(fd6, (const struct sockaddr *) &sin6, sin6.sin6_len);

err = listen(fd6, 5);
```

Server: Hook up IPv4 CFSocket

```
CFSocketContext    context = { 0, NULL, NULL, NULL, NULL };
CFSocketRef        sock;
CFRunLoopSourceRef rls;

sock = CFSocketCreateWithNative(NULL, fd4,
                                kCFSocketAcceptCallBack,
                                ListeningSocketCallback,
                                &context);
rls = CFSocketCreateRunLoopSource(NULL, sock, 0);
CFRunLoopAddSource(CFRunLoopGetCurrent(),
                   rls,
                   kCFRunLoopCommonModes);

CFRelease(rls);
CFRelease(sock);
```

Server: Hook up IPv4 CFSocket

```
CFSocketContext    context = { 0, NULL, NULL, NULL, NULL };
CFSocketRef        sock;
CFRunLoopSourceRef rls;

sock = CFSocketCreateWithNative(NULL, fd4,
                                kCFSocketAcceptCallBack,
                                ListeningSocketCallback,
                                &context);
rls = CFSocketCreateRunLoopSource(NULL, sock, 0);
CFRunLoopAddSource(CFRunLoopGetCurrent(),
                   rls,
                   kCFRunLoopCommonModes);

CFRelease(rls);
CFRelease(sock);
```

Server: Hook up IPv6 CFSocket

```
CFSocketContext    context = { 0, NULL, NULL, NULL, NULL };
CFSocketRef        sock;
CFRunLoopSourceRef rls;

sock = CFSocketCreateWithNative(NULL, fd6,
                                kCFSocketAcceptCallBack,
                                ListeningSocketCallback,
                                &context);
rls = CFSocketCreateRunLoopSource(NULL, sock, 0);
CFRunLoopAddSource(CFRunLoopGetCurrent(),
                   rls,
                   kCFRunLoopCommonModes);

CFRelease(rls);
CFRelease(sock);
```

Server: Hook up IPv6 CFSocket

```
CFSocketContext    context = { 0, NULL, NULL, NULL, NULL };
CFSocketRef        sock;
CFRunLoopSourceRef rls;

sock = CFSocketCreateWithNative(NULL, fd6,
                                kCFSocketAcceptCallBack,
                                ListeningSocketCallback,
                                &context);
rls = CFSocketCreateRunLoopSource(NULL, sock, 0);
CFRunLoopAddSource(CFRunLoopGetCurrent(),
                   rls,
                   kCFRunLoopCommonModes);

CFRelease(rls);
CFRelease(sock);
```

Server: Accept Callback

```
void ListeningSocketCallback(CFSocketRef sock,
CFSocketCallBackType type, CFDataRef address, const void *data,
void *info)
{
    int fd = * (const int *) data;
    CFReadStreamRef   readStream;
    CFWriteStreamRef  writeStream;
    NSInputStream  *  inputStream;
    NSOutputStream *  outputStream;

    CFStreamCreatePairWithSocket(NULL, fd, &readStream,
                                 &writeStream);
    inputStream  = CFBridgingRelease(readStream);
    outputStream = CFBridgingRelease(writeStream);

    [inputStream setProperty:(id)kCFBooleanTrue
      forKey:(NSString *)kCFStreamPropertyShouldCloseNativeSocket];
}
```

Server: Register

```
NSNetService *service =
 [[NSNetService alloc]
  initWithDomain:@""
            type:@"_moodring._tcp."
            name:@""
            port:ntohs(sin.sin_port)];

[service scheduleInRunLoop:[NSRunLoop currentRunLoop]
                   forMode:NSRunLoopCommonModes];

[service setDelegate:self];

[service publish];
```

Server: Register

```
NSNetService *service =
 [[NSNetService alloc]
  initWithDomain:@""
            type:@"_moodring._tcp."
            name:@""
            port:ntohs(sin.sin_port)];

[service scheduleInRunLoop:[NSRunLoop currentRunLoop]
                   forMode:NSRunLoopCommonModes];

[service setDelegate:self];

[service publish];
```

Server: Register

```
NSNetService *service =
 [[NSNetService alloc]
  initWithDomain:@""
            type:@"_moodring._tcp."
            name:@""
            port:ntohs(sin.sin_port)];

[service scheduleInRunLoop:[NSRunLoop currentRunLoop]
                   forMode:NSRunLoopCommonModes];

[service setDelegate:self];

[service publish];
```

Server: Register

```
NSNetService *service =
 [[NSNetService alloc]
  initWithDomain:@""
            type:@"_moodring._tcp."
            name:@""
            port:ntohs(sin.sin_port)];

[service scheduleInRunLoop:[NSRunLoop currentRunLoop]
                   forMode:NSRunLoopCommonModes];

[service setDelegate:self];

[service publish];
```

Server: Register

```
NSNetService *service =
 [[NSNetService alloc]
  initWithDomain:@""
            type:@"_moodring._tcp."
            name:@""
            port:ntohs(sin.sin_port)];

[service scheduleInRunLoop:[NSRunLoop currentRunLoop]
                   forMode:NSRunLoopCommonModes];

[service setDelegate:self];

[service publish];
```

Server: Register Callbacks

```
- (void)netServiceDidPublish:(NSNetService *)sender {
 NSString *name = [sender name];
 NSLog("My name is: %@", name);
}

- (void)netService:(NSNetService *)sender
  didNotPublish:(NSDictionary *)errorDict {
 NSLog("Error publishing: %@", errorDict);
}
```

Client: Browse for Services

```
NSNetServiceBrowser *browser =
 [[NSNetServiceBrowser alloc] init];
 [browser setDelegate:self];
 [browser
  searchForServicesOfType:@"_moodring._tcp."
  inDomain:@""];
```

Client: Browse for Services

```
NSNetServiceBrowser *browser =
 [[NSNetServiceBrowser alloc] init];
 [browser setDelegate:self];
 [browser
  searchForServicesOfType:@"_moodring._tcp."
  inDomain:@""];
```

Client: Browse for Services

```
NSNetServiceBrowser *browser =
 [[NSNetServiceBrowser alloc] init];
 [browser setDelegate:self];
 [browser
  searchForServicesOfType:@"_moodring._tcp."
  inDomain:@""];
```

Client: Browse Callbacks

```
- (void)netServiceBrowser:(NSNetServiceBrowser *)netServiceBrowser
  didFindService:(NSNetService *)service
  moreComing:(BOOL)moreComing {
 [self.model addObject:service];
 if (!moreComing) [self.tableView reloadData];
}
```

Client: Browse Callbacks

```
- (void)netServiceBrowser:(NSNetServiceBrowser *)netServiceBrowser
  didFindService:(NSNetService *)service
  moreComing:(BOOL)moreComing {
 [self.model addObject:service];
 if (!moreComing) [self.tableView reloadData];
}
```

Client: Browse Callbacks

```
– (void)netServiceBrowser:(NSNetServiceBrowser *)netServiceBrowser
  didRemoveService:(NSNetService *)service
  moreComing:(BOOL)moreComing {
 [self.model removeObject:service];
 if (!moreComing) [self.tableView reloadData];
}
```

Client: Browse Callbacks

```
- (void)netServiceBrowser:(NSNetServiceBrowser *)netServiceBrowser
  didRemoveService:(NSNetService *)service
  moreComing:(BOOL)moreComing {
 [self.model removeObject:service];
 if (!moreComing) [self.tableView reloadData];
}
```

Client: Resolve and Connect

```
NSNetService   * service = TheServiceTheUserSelected();
NSInputStream  * inStream;
NSOutputStream * outStream;

[service getInputStream:&inStream
           outputStream:&outStream]; // See Technical Q&A QA1546

inStream.delegate = self;
outStream.delegate = self;

[inStream scheduleInRunLoop:[NSRunLoop currentRunLoop]
                    forMode:NSDefaultRunLoopMode];
[outStream scheduleInRunLoop:[NSRunLoop currentRunLoop]
                     forMode:NSDefaultRunLoopMode];

[inStream open];
[outStream open];
```

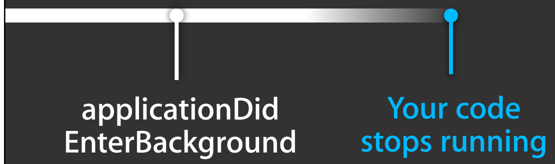
Client: Resolve and Connect

```
NSNetService   * service = TheServiceTheUserSelected();
NSInputStream  * inStream;
NSOutputStream * outStream;

[service getInputStream:&inStream
           outputStream:&outStream]; // See Technical Q&A QA1546

inStream.delegate = self;
outStream.delegate = self;

[inStream scheduleInRunLoop:[NSRunLoop currentRunLoop]
                    forMode:NSDefaultRunLoopMode];
[outStream scheduleInRunLoop:[NSRunLoop currentRunLoop]
                     forMode:NSDefaultRunLoopMode];

[inStream open];
[outStream open];
```

Tips and Reminders

Rory McGuire
Senior Packet Wrangler

Bonjour and iOS Multitasking



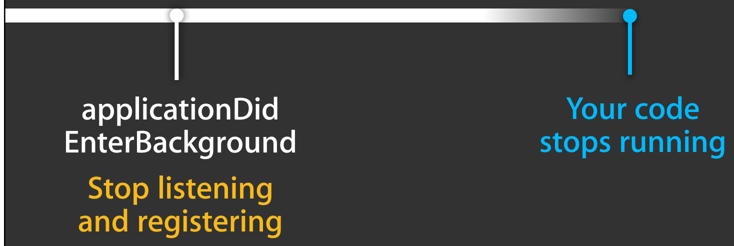
Bonjour and iOS Multitasking

applicationDid
EnterBackground

Stop listening
and registering

Your code
stops running

Bonjour and iOS Multitasking

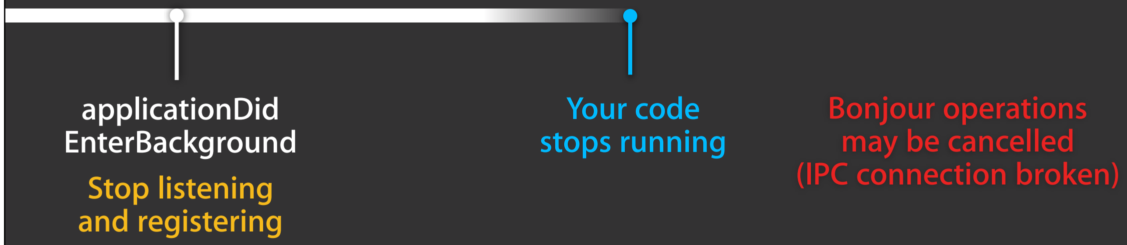


applicationDid
EnterBackground

Stop listening
and registering

Your code
stops running

Bonjour and iOS Multitasking



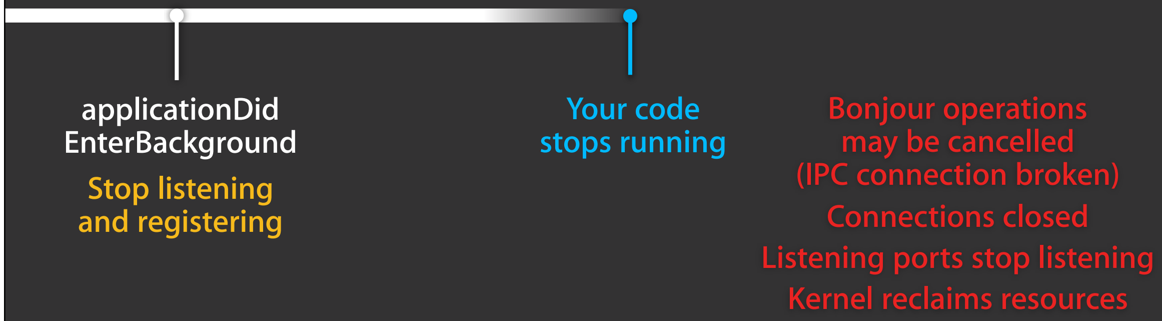
applicationDid
EnterBackground

Stop listening
and registering

Your code
stops running

Bonjour operations
may be cancelled
(IPC connection broken)

Bonjour and iOS Multitasking



applicationDid EnterBackground

Stop listening and registering

Your code stops running

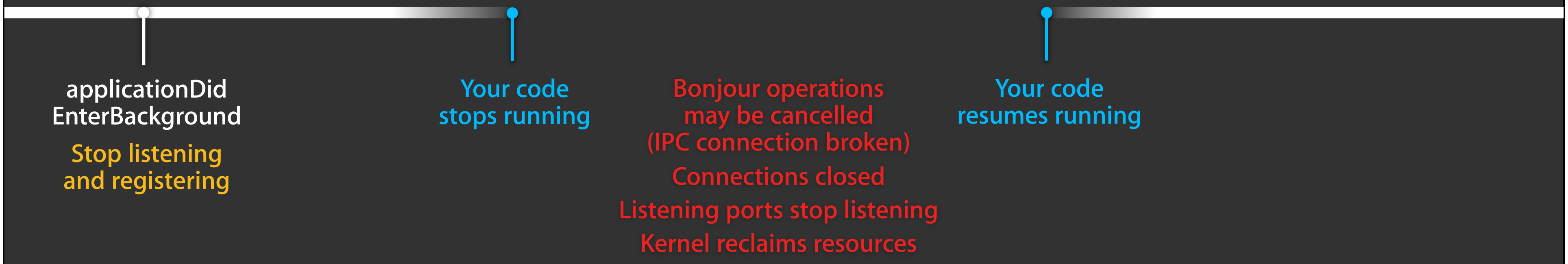
Bonjour operations may be cancelled (IPC connection broken)

Connections closed

Listening ports stop listening

Kernel reclaims resources

Bonjour and iOS Multitasking



applicationDid
EnterBackground

Stop listening
and registering

Your code
stops running

Bonjour operations
may be cancelled
(IPC connection broken)

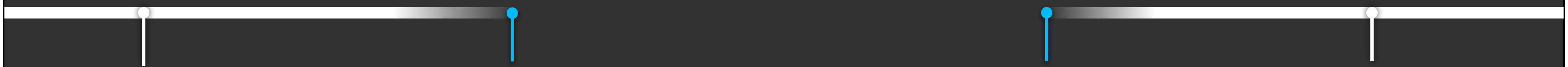
Connections closed

Listening ports stop listening

Kernel reclaims resources

Your code
resumes running

Bonjour and iOS Multitasking



applicationDid
EnterBackground

Stop listening
and registering

Your code
stops running

Bonjour operations
may be cancelled
(IPC connection broken)

Connections closed

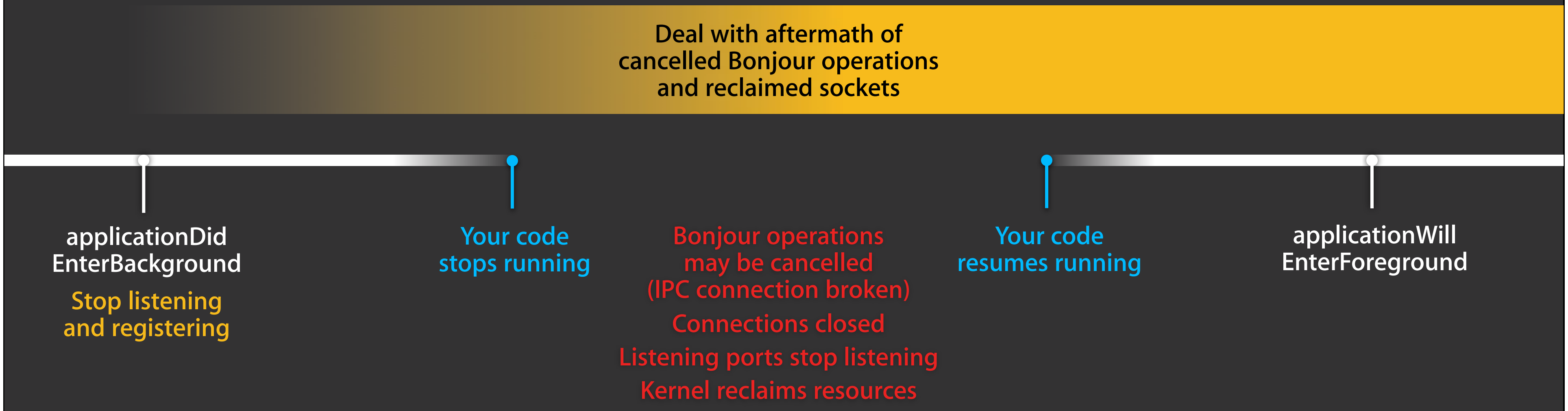
Listening ports stop listening

Kernel reclaims resources

Your code
resumes running

applicationWill
EnterForeground

Bonjour and iOS Multitasking



Deal with aftermath of cancelled Bonjour operations and reclaimed sockets

applicationDid EnterBackground

Stop listening and registering

Your code stops running

Bonjour operations may be cancelled (IPC connection broken)

Connections closed

Listening ports stop listening

Kernel reclaims resources

Your code resumes running

applicationWill EnterForeground

iOS Multitasking Aftermath

- At any time following “applicationDidEnterBackground” message, all Bonjour and other networking may become broken for your app
  - Registrations may be gone
  - Discovery may have stopped
  - Suggestion: If browse terminated by iOS, auto-dismiss your browsing UI
    - If the user wishes, they can browse again
- See Technical Note TN2277 “Networking and Multitasking”

Client: Browser Callback

```
- (void)netServiceBrowser:(NSNetServiceBrowser *)netServiceBrowser
  didNotSearch:(NSDictionary *)errorInfo {
 // Application resumed from background
 // and Bonjour browsing operation was cancelled
 // so we dismiss it now
 [self dismissBrowseUI];
}
```

Use Asynchronous Calls

- Blocking calls get your app jettisoned
  - DNS timeout is 30 seconds
  - iOS watchdog timer is 20 seconds
  - Don't do a blocking DNS call on your main thread
- Use NS API and Grand Central Dispatch

Browse Call — The Right Way

- Use live dynamic UI
  - No need for a refresh button
  - Consider your user experience
- No open-ended browsing
  - Browse when requested by user, not constantly
  - Stop browsing when not displaying browse UI
- Resolve and connect when requested by user, not to every service you find

Browse vs. Resolve

Browse vs. Resolve

Browse vs. Resolve

Leo’s Printer			
Sally’s Printer			
Jim’s Printer			
Penny’s Printer			
Paul’s Printer			
Mary’s Printer			

Browse vs. Resolve

Browse vs. Resolve

Jim’s Printer	jim.local	9100	pdl=application/postscript …

Browse and Then Resolve Everything?

Browse and Then Resolve Everything?

Don't do it!

Leo’s Printer	leo.local	9100	pdl=application/postscript …
Sally’s Printer	sally.local	9100	pdl=application/postscript …
Jim’s Printer	jim.local	9100	pdl=application/postscript …
Penny’s Printer	penny.local	9100	pdl=application/postscript …
Paul’s Printer	paul.local	9100	pdl=application/postscript …
Mary’s Printer	mary.local	9100	pdl=application/postscript …

Saving Services You Found

- Bad ideas
  - Save just the IP address
  - Save the IP address and port number
  - Save the host name and port number

Saving Services You Found

- Bad ideas
    - Save just the IP address
    - Save the IP address and port number
    - Save the host name and port number
- The right way
    - Late binding is the key
    - Service is identified by three-tuple: { Name, Type, Domain }
    - Save { Name, Type, Domain } tuple

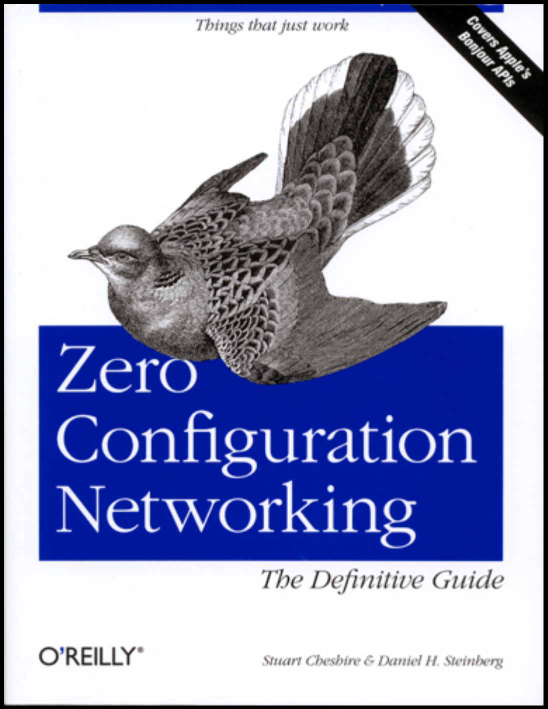
More Information

Craig Keithley
Bonjour Technology Evangelist
keithley@apple.com

Paul Danbold
Core OS Technology Evangelist
danbold@apple.com

Documentation
Bonjour Developer Web Page
WiTap and RemoteCurrency Sample Code
http://developer.apple.com/bonjour

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


Things that just work
Covers Apple's Bonjour APIs
Zero Configuration Networking
The Definitive Guide
O'REILLY®
Stuart Cheshire & Daniel H. Steinberg

Related Sessions

706 Networking Best Practices	Nob Hill Tuesday 3:15PM
712 Asynchronous Design Patterns with Blocks, GCD, and XPC	Pacific Heights Friday 9:00AM

Labs

Networking Lab	Core OS Lab A Wednesday 9:00AM
Networking Lab	Core OS Lab A Thursday 9:00AM

Summary

Networking is hard; Bonjour makes it easy and reliable

- UI should update live — no refresh button
- Use late binding — { Name, Type, Domain } tuple
- After receiving applicationDidEnterBackground, expect and handle errors
- Register your service types

WWDC2012