CarrierPigeon Messenger

Application Authors

Abdullah Al-Syed (aza2105)
Chris D'Angelo (cd2665)
Ifeoma Okereke (iro2103)
Riley Spahn (rbs2152)

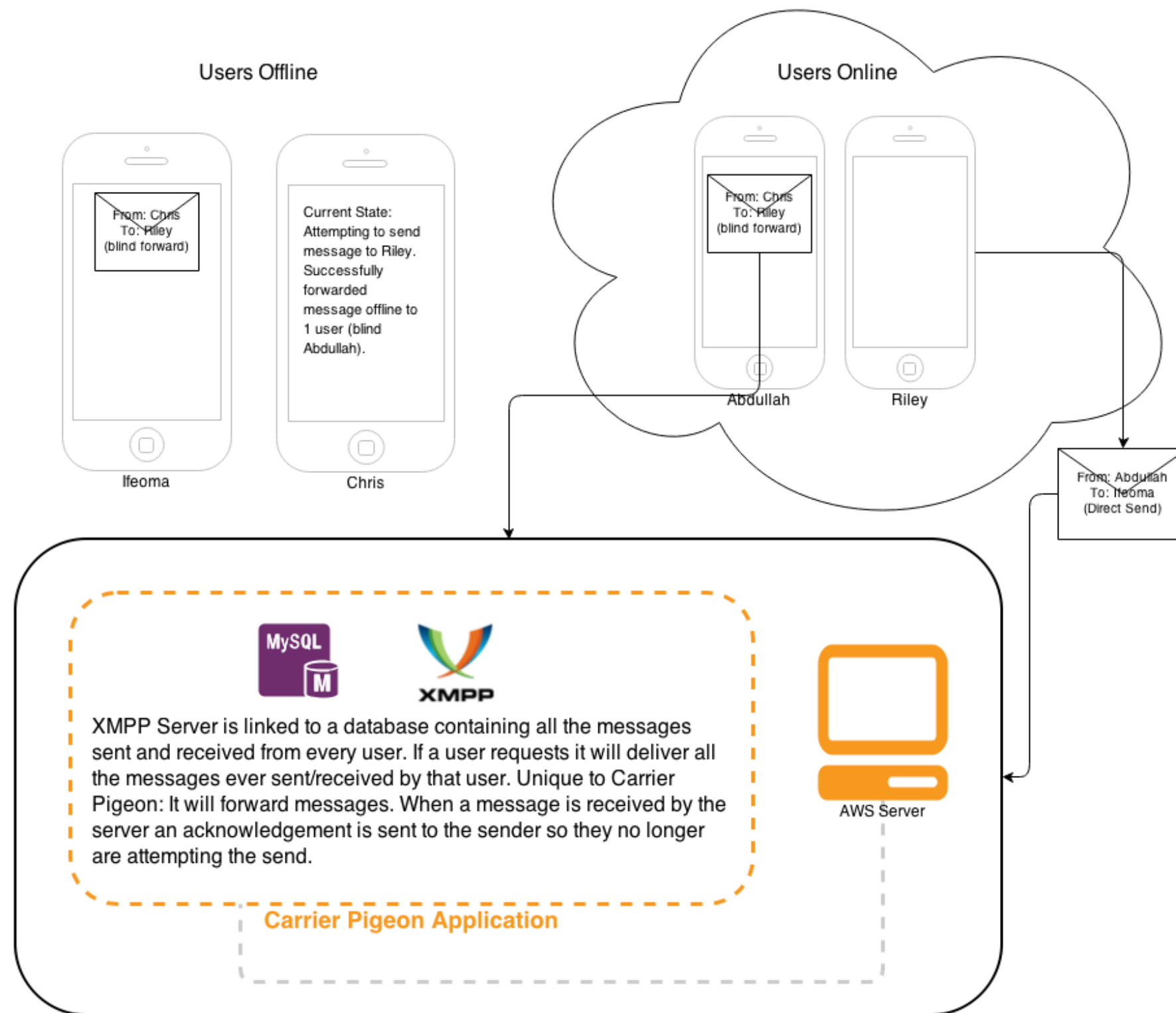May 5, 2014

Introduction to CarrierPigeon

Carrier Pigeon is an iOS app that allows you to send messages to your friends while online and offline.
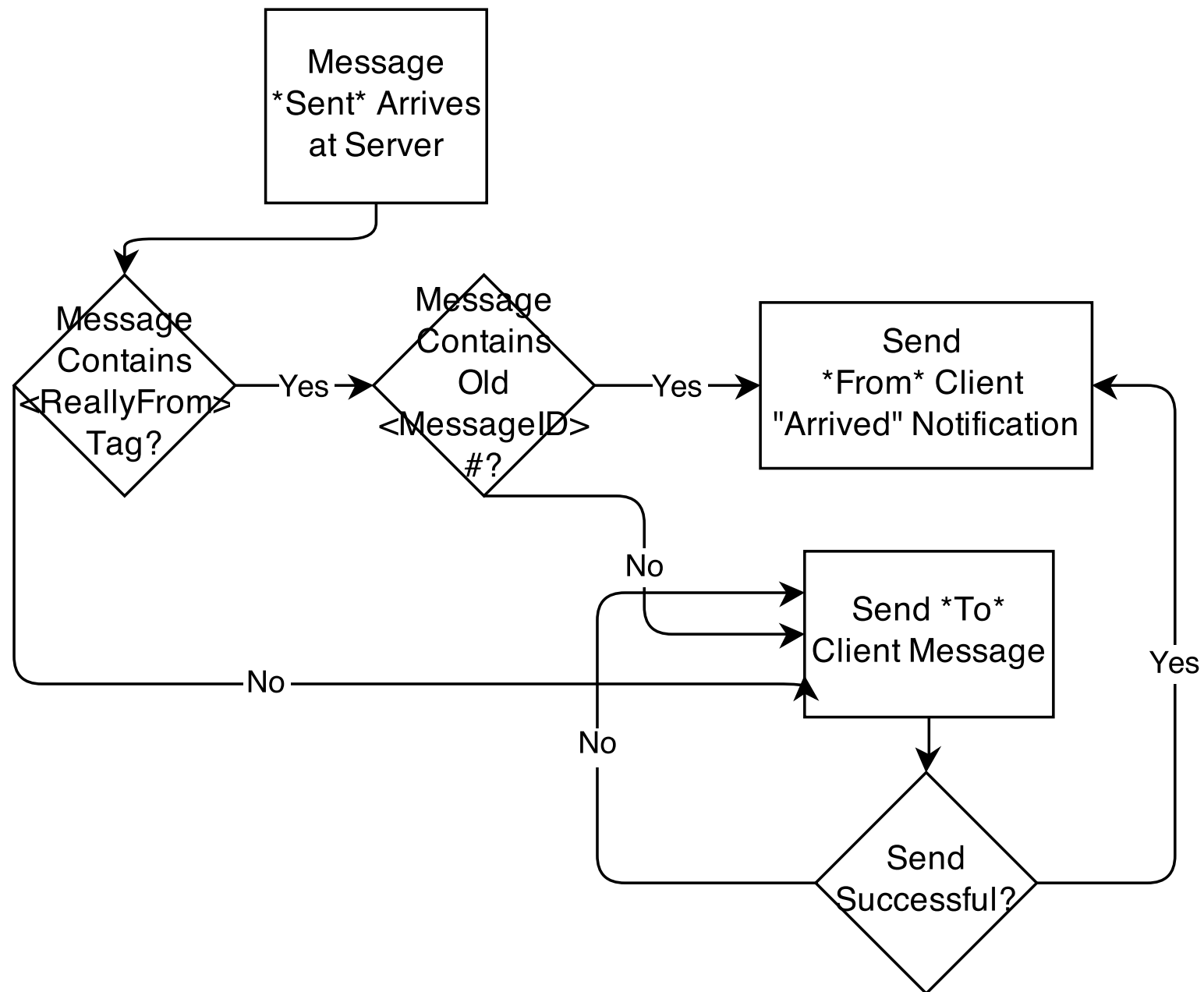
Use Case

Say you're in the subway and you want to send your mom a message that you'll be late. Your message can piggy back on other users who access the internet sooner. These "Carrier Pigeons" will bring your message to your mom faster than you can because you don't have the internet and they do!

Architecture

Basic Design

Users Offline

Users Online

From: Chris
To: Riley
(blind forward)

Current State:
Attempting to send
message to Riley.
Successfully
forwarded
message offline to
1 user (blind
Abdullah).

From: Chris
To: Riley
(blind forward)

Abdullah

Riley

Ifeoma

Chris

From: Abdullah
To: Ifeoma
(Direct Send)

MySQL
M

XMPP

XMPP Server is linked to a database containing all the messages
sent and received from every user. If a user requests it will deliver all
the messages ever sent/received by that user. Unique to Carrier
Pigeon: It will forward messages. When a message is received by the
server an acknowledgement is sent to the sender so they no longer
are attempting the send.

AWS Server

Carrier Pigeon Application

Server Incoming Message Paths

```
                    ┌─────────────────┐
                    │    Message      │
                    │ *Sent* Arrives  │
                    │   at Server     │
                    └────────┬────────┘
                             │
        ┌────────────┐            ◇ Message
        │  Message   │───Yes──→  ◇ Contains  ───Yes──→  ┌──────────────────┐
        │  Contains  │           ◇ Old                  │      Send        │
        │<ReallyFrom>│           ◇ <MessageID>          │  *From* Client   │
        │   Tag?     │              #?                  │"Arrived" Notification │
        └─────┬──────┘                │                 └──────────────────┘
              │                       No
              │                        │        ┌──────────────────┐         Yes
              │                        └──────→ │  Send *To*       │ ←───────
              No                                │  Client Message  │
              │                        No       └────────┬─────────┘
              └────────────────────────────────┘         │
                                                          ▼
                                                   ◇  Send
                                                   ◇ Successful? ◇
```

Client Incoming Message Paths

Message
Arrives at Client

Message
Contains
<ReallyFrom>
Tag?

Message
Arrived

No

Yes
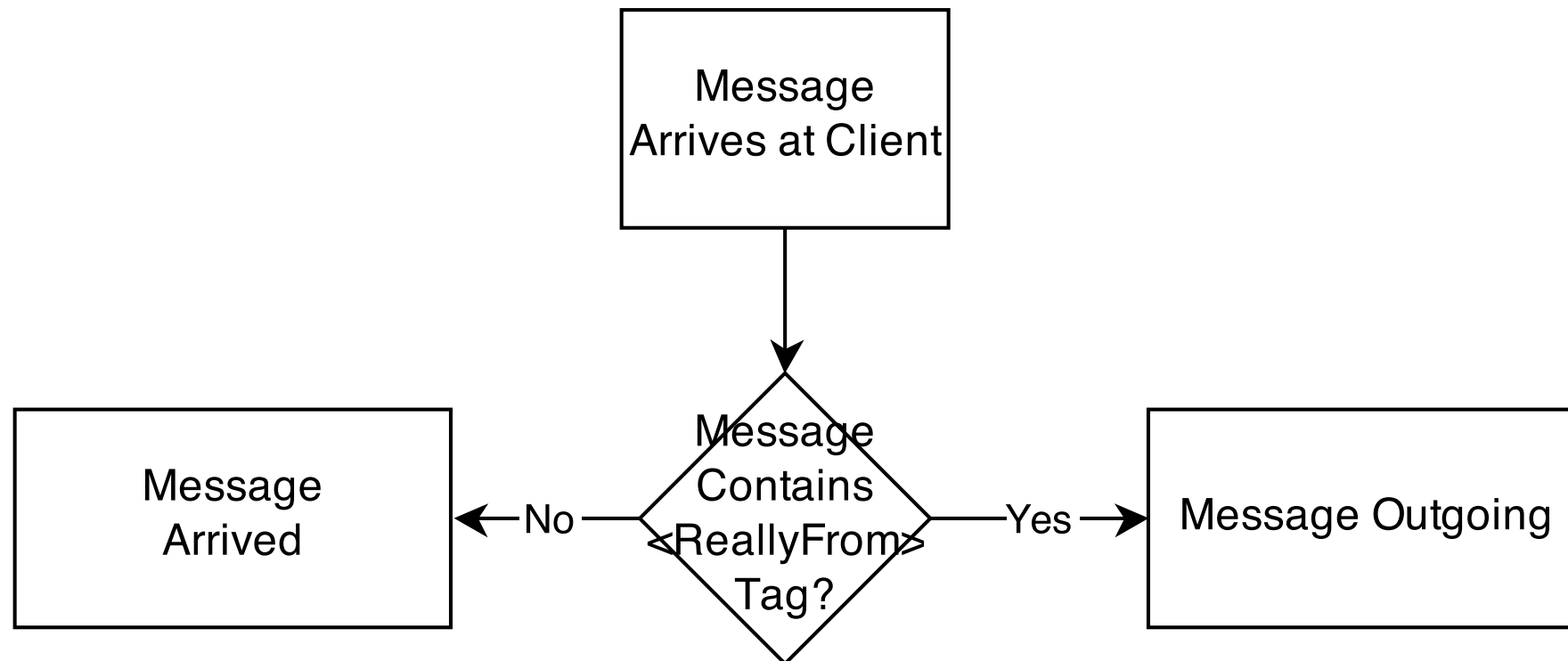
Message Outgoing

Client Outgoing Message Paths

```
                              ┌──────────────┐
                              │   Message    │
                              │   Outgoing   │
                              └──────┬───────┘
                                     │
                                     ▼
          No                   ◇ Online? ◇ ──Yes──► ┌──────────┐
    ┌────────────────────────                       │ Message  │
    │                                                │ Sending  │
    │         No                                     └────┬─────┘
    │   ┌─────────────────────                             │
    │   │                                                  ▼
    │   │        No                              ◇  Send  ◇ ──Yes──► ┌──────────┐
    │   │ ┌──────────────────────────────────────  Success? ◇         │ Message  │
    │   │ │                                                            │   Sent   │
    │   ▼ ▼                                                            └────┬─────┘
    │  ◇ Peers ◇ ──Yes──► ┌──────────┐                                      │ Message
    │  ◇Available?◇        │ Message  │                                      │ Arrive Receipt
    │   └──▲──┘            │ Relaying │                                      │ Arrives
    │      │               └────┬─────┘                                      ▼
    │      │         No          │                                    ┌──────────┐
    │      │ ┌──────────────     ▼                          ┌────────►│ Message  │
    │      │ │            ◇  Relay  ◇ ──Yes──► ┌──────────┐  │         │ Arrived  │
    │      │ │            ◇ Success? ◇          │ Message  │  │         └────┬─────┘
    │      │ └──                                │ Relaying │  │              │ Message
    │      │                                    │RelayCount++│ │              │ Read Receipt
    │      │                                    └────┬─────┘  │              │ Arrives
    │      │                                         │        │              ▼
    │      │         No                              ▼        │        ┌──────────┐
    │      └──────────────────────────── ◇RelayCount◇──Yes──►┌──────┐ │ Message  │
    │                                     ◇  > 5 ?  ◇         │Message│ │   Read   │
    │                                                         │Relayed│ └──────────┘
    │                                                         └───┬──┘
    └─────────────────────────────────────────────────────────────┘
                                              Message
                                              Arrive Receipt
                                              Arrives
```
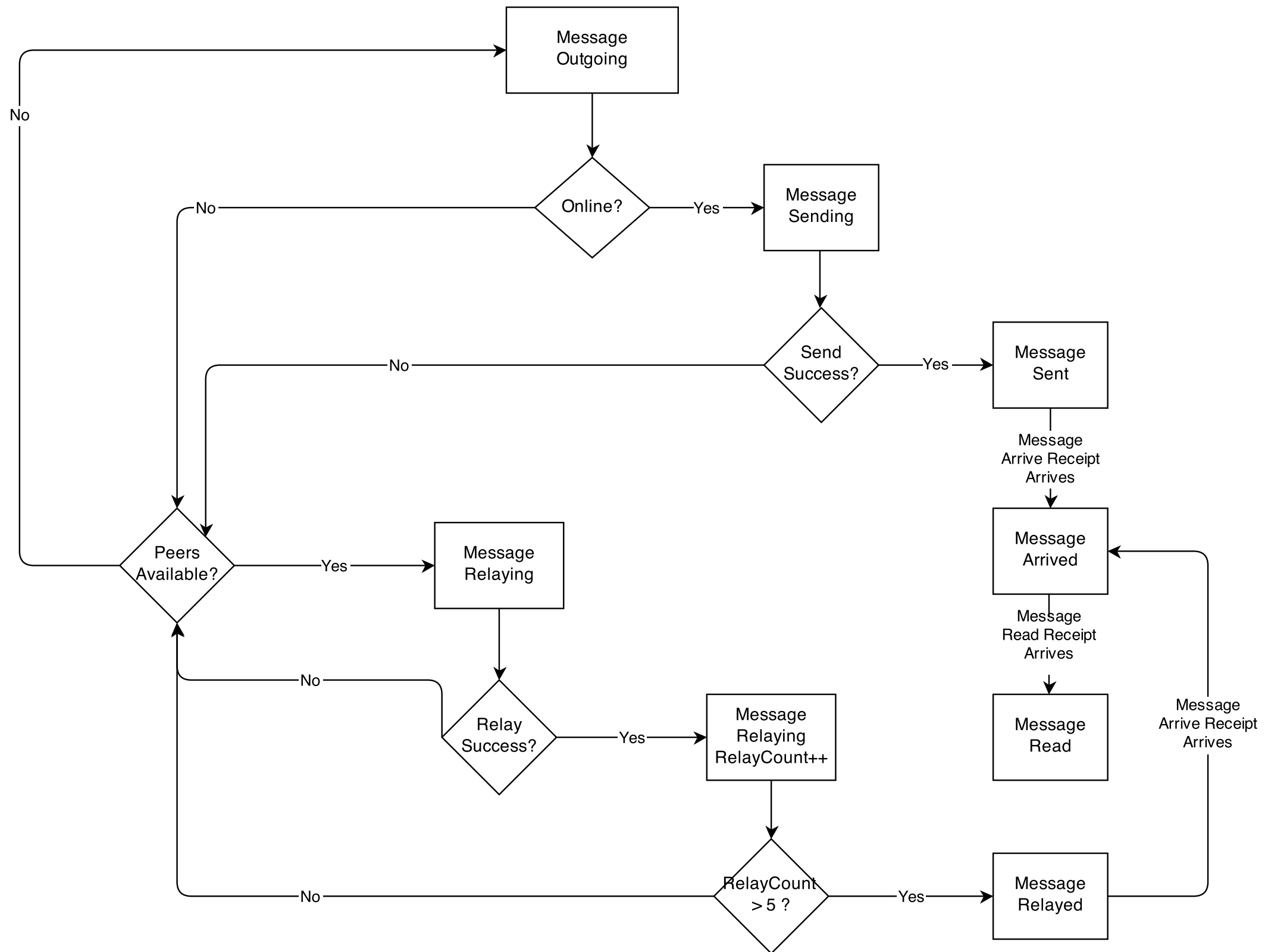
Message Forwarding

Typical XMPP Protocol Message:

```
<message type="chat" from="david" to="chris"
id="17"><body>Hello</body><active xmlns="http://
jabber.org/protocol/chatstates"></active></message>
```

Carrier Pigeon "Relayed" Message:

```
<message type="chat" from="david" to="chris" id="17"
reallyFrom="julia" reallyFromId="3"><body>Hello</
body><active xmlns="http://jabber.org/protocol/
chatstates"></active></message>
```

Beta Testing

Free Beta Testing without TestFlight? This guide is helpful:
http://jeffreysambells.com/2010/06/22/ios-wireless-app-distribution

Application Features

In Development

- Push Notifications
- Retrieving old messages from server
- iPad Version
- Message Read Receipts

Future Plans

- Intelligent use of background state
- Intelligent outbox sending
- Intelligent MultiPeer Connectivity Broadcasting
- XMPPFramework fix didDisconnect Listener
- Picture Sending
- Direct Peer to Peer sending
- End to End encryption
- Remove user-facing diagnostics views
- Bug fixes
- Much More… http://bit.ly/carrierpigeontasks

Demo

Both Users Online

1. Elaine (Online) messages Jerry (Online) that she's getting into the subway now.
2. Jerry (Online) responds.

"Hi Jerry. I'm about to get into the Subway"

"Hi Jerry. I'm about to get into the Subway"
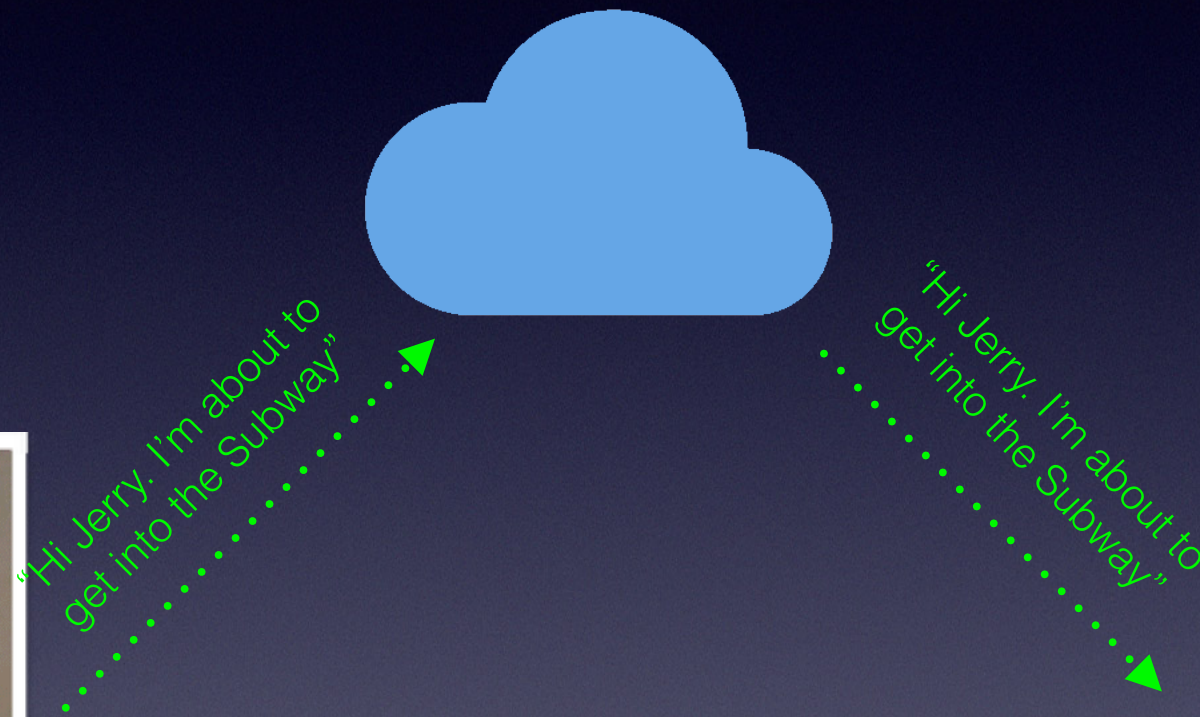
(Online)

(Online)

One User Goes Offline

1. Elaine goes into the subway where Kramer is sitting next to her.
2. Elaine (Offline, Connected P2P with Kramer) writes a message to Jerry (Online) which goes into her outbox and Kramer's outbox.
3. Kramer (Online) exits the subway before Elaine. Elaine's message is delivered to Jerry (Online) from Kramer.
4. Jerry (Online) receives the message.
5. Elaine (Online) leaves the subway. Her outbox delivers the message to the server. Server ignores it because it's been delivered.

To: Jerry "I'm Running Late"

From: Elaine "I'm Running Late"

From: Elaine "I'm Running Late"

(Offline)

(Online)

(Online)

Demo Video

bit.ly/carrierpigeondemo

Demo created April 26, 2014. Select features were finalized May 5, 2014.

CarrierPigeon

iOS Source Code: bit.ly/cpigeon

Beta Testing Distribution: bit.ly/carrierpigeonbeta

Presentation: bit.ly/carrierpigeonpresentation

Video Demo: bit.ly/carrierpigeondemo