MadHoc

A Mobile Ad-hoc Network

Team Members: Cole Cummings, Ethan Niemeyer, Cody Lougee
Advisor/Client: George Amariucai

Project Goals

- Build a messaging application (app) for mobile devices.
- App works in the absence of standard wireless network infrastructure.
- App will function by maintaining an ad-hoc network.
- An ad-hoc secret-key establishment protocol will allow secure communication between the users.

Design requirements

Functional

- Operate without access to cell towers, wifi routers, internet access, etc.
- Send communication securely if possible
- Users communicate via joined groups
- Create and maintain an ad-hoc network

Non-functional

- Reasonable response time
- Adhere to Apples' app store standards

Operating environment

- Primarily outdoors in scenarios such as
  - Diaster areas
  - Protests
  - Large gatherings

Intended users and uses

- Fire and rescue workers could use it to communicate if cell towers or power is knocked out in an area after a natural disaster
- Protestors have used similar apps to plan protests in Iraq and Hong Kong when the government cut off internet access

Testing

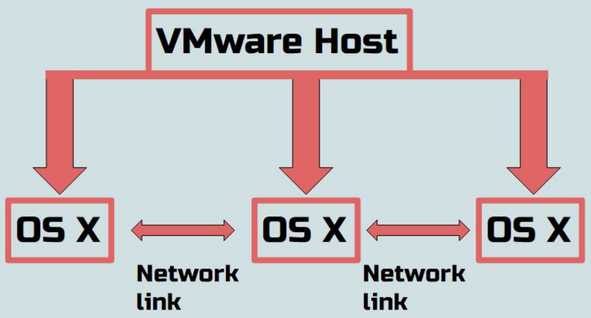
Environment

- Xcode iPad/iPhone simulators in VMware
- iPad devices physically distributed

Strategy

- Simulate heavy loads virtually
- Test how the network responds to sub-networks leaving and joining
- Test if all nodes can talk to each other
- Check if encryption is working properly

Virtual testing



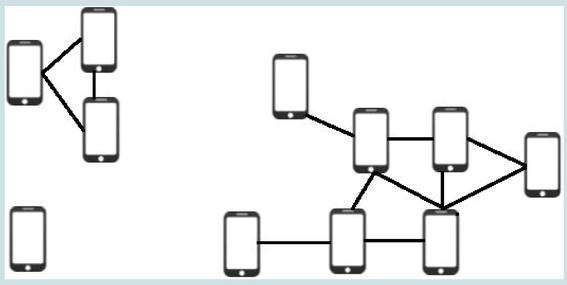
Simulation

A simple ad-hoc network can be simulated for testing purposes. The IDE Xcode can simulate a single iOS device. In order to simulate multiple iOS devices, multiple instances of OSX networked together need to be ran. One instance of OSX acts as a middle man that the other two instances talk through.

Technologies used

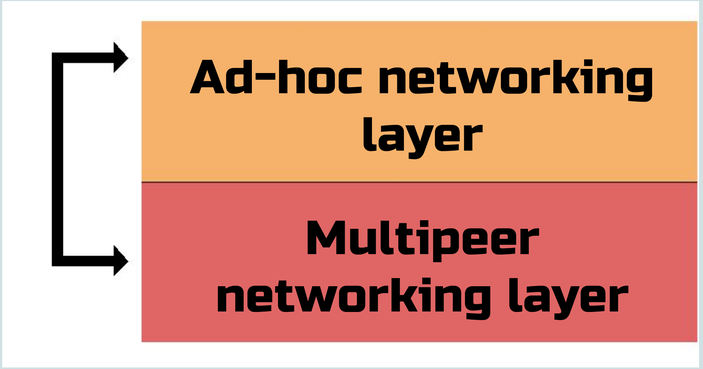
Apple iPad - Apples' tablet computer used to run and test the app on.
Xcode - Apples' IDE used for developing the app.
iOS - The operating system that runs on the iPad.
OSX - Apple desktop operating system used to run Xcode.
Swift 2.0 - Programming language used to write the app.
Multipeer Connectivity Library - Swift code library that handles peer to peer connections.
VMware - Virtual machine software used to simulate multiple mobile devices as well as the network conditions.

Design approach



Ad-hoc network

In an ad-hoc network each connected device participates in network routing. Devices are connected to one another rather than a central router.



Network layering

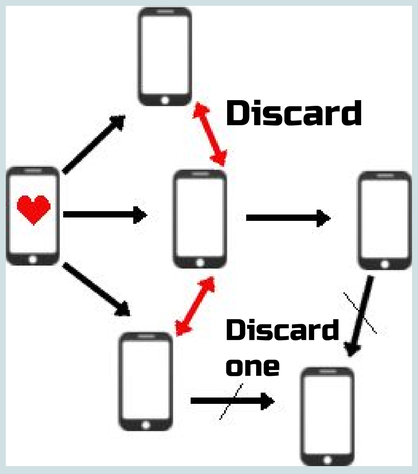
The networking logic is divided into two layers that talk with each other. Our ad-hoc layer and Apples' Multipeer framework layer.

Multipeer layer

Swift code library that looks for devices within range and creates a connection with them. It also sends data and receives data from connected peers. This data is passed to the ad-hoc layer where further processing happens.

Ad-hoc layer

Manages route finding, device discovery, and encryption. Devices in the network and routes to them are found by using a heartbeat, which is explained below. Data is passed from one device to another until it reaches its destination. While Multipeer handles sending data to the next device, the ad-hoc layer decides what the next network hop is.



Heartbeat

Each device sends out a special heartbeat broadcast message. The message is sent out periodically. Since ad-hoc networks are dynamically changing, the heartbeat acts as a way to keep track of what devices are currently in the network. The heartbeat contains the route it has taken thus far. The device stores the route to the heartbeat's originator and re-broadcasts the heartbeat. If the heartbeat has already been seen, discard it.

Encryption

Once enough devices join the network, a secret key between two devices can be generated. The key is generated by looking at known routes cointaining the two devices. The routes that are unique to both devices can be used for randomness to generate the key.

Routes
1 3 4 7 11
2 6 5
1 2 5 8
4 7 5 2 9

Algorithm to find unique paths and generate key

Key between two devices