MadHoc

Mobile Ad Hoc Networking

The Team

Ethan Niemeyer

Cole Cummings

Cody Lougee

Advisor/Client: George Amariucai

Project Description

- Mobile device chat application
- Works without typical network infrastructure
- Achieved by creating an ad-hoc network
- Encrypts messages

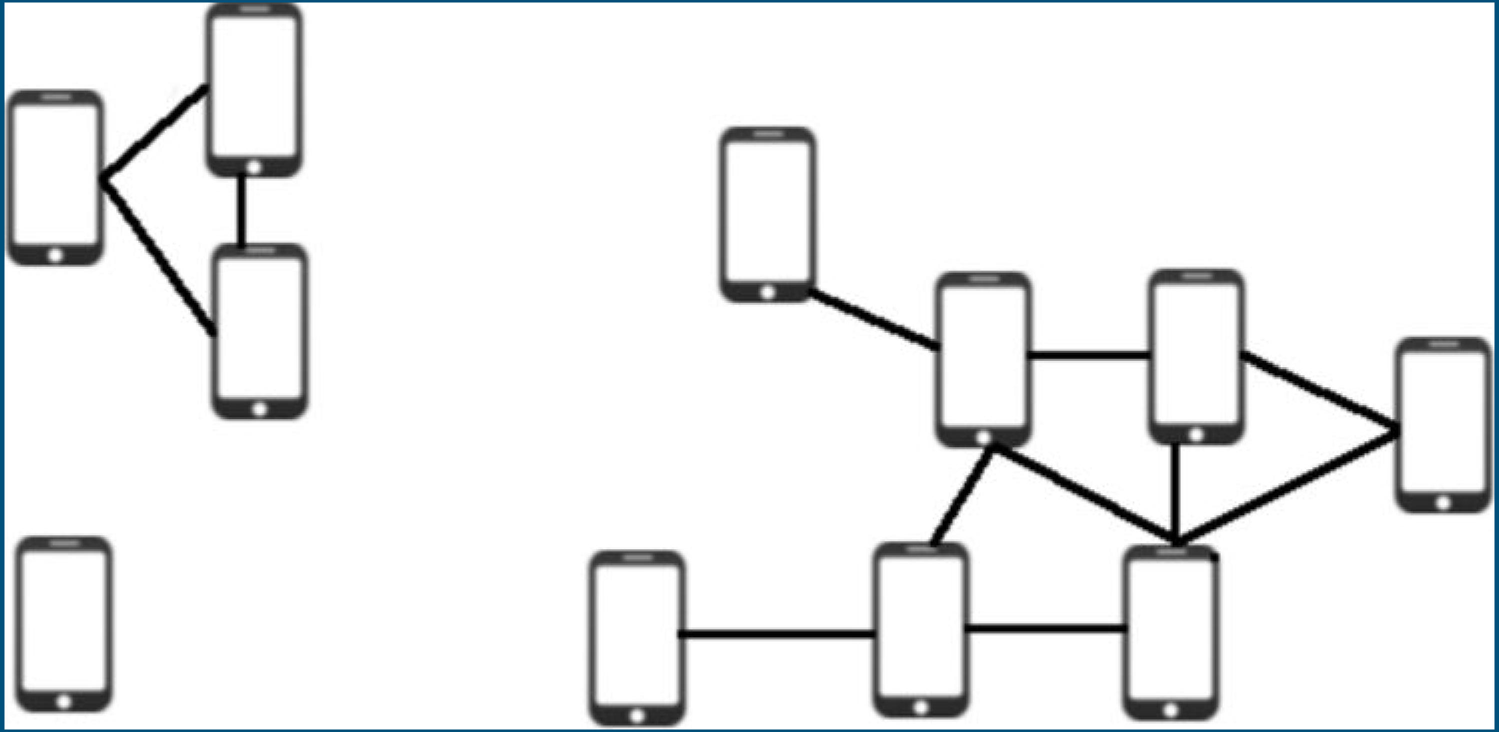
Ad Hoc

- No infrastructure
- Dynamically changing
- All nodes have equal responsibility/privileges on the network

Node

- Phones
- Each connects to those around it
- Propagate messages further

Example of an Ad Hoc Network



Early First semester

- Focused on researching existing projects
- Brushing up on Android
- Studying Ad Hoc networking examples

Existing App - Fire Chat

- Created by OpenGarden
- Wrote their own library for wireless communication on top of wifi and bluetooth
- Their library is not open sourced
- Does not support any form of encryption right now

Existing App - Serval

- Created for Australian outback and areas where the population is too sparse to have cell tower infrastructure
- Intended to be used by rescue personnel and law enforcement in emergencies
- Also some everyday use by people

Original prototype - Android

- Group had some previous experience in Android
- We started basic prototypes in Android
- Some early time was spent looking into similar apps that already exist

Android

- Apps written in Java
- Similar Applications exist on Android
- Android’s implementation of Peer to Peer
- One way spoke network
- True Ad Hoc disabled at the firmware level

IOS

- Apps written in Swift
- Multipeer Connectivity library
- Multipeer provided basic ad hoc functionality
- Connects nearby nodes
- Sends data between nodes

IOS - Week 10, First semester

- Most promising of possible solutions
- However
    - No experience with Swift, little experience with mobile apps
    - Apps must be developed on Xcode which only runs on OSX
    - One member had an Apple computer, no one had Apple mobile devices
    - Multipeer isn’t perfect

Second Semester

- Holden transferred Universities
- Swift update broke boilerplate code and removed some functions completely
- Decided to use the newer Swift because some Multipeer bugs were fixed

Design

- Had to make significant changes over the semester
- DSR Protocol
- Node to Node Communication
- Node IDs
- Groups

Nodes

- Users specify a username
- Nodes need a unique ID - identified by their ID
  - Advertising ID
  - Vendor ID
- Couldn’t use Advertising ID
- Usernames shown as chosen name and 4 digits of Vendor ID

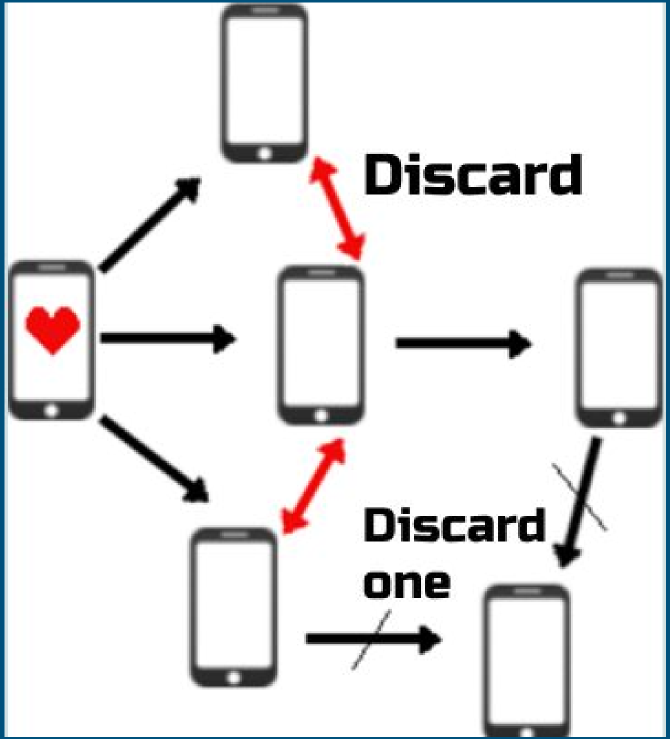
DSR

- Networking protocol initially used
- Can be used for ad hoc networks
- Written with the assumption nodes are aware of other nodes already
- We needed a way to update in close to real time when a node joined or dropped the network
- Heartbeat is our solution

Heartbeat - Week 7, second semester

- Broadcast message with route information
- Contains list of nodes passed through thus far
- Nodes add their ID and rebroadcast
- Discard heartbeats already seen
- Store paths in an array
- Updated by the heartbeat

Heartbeat



Node to Node Communication

- 2 Networking layers
- Ad hoc and Multipeer layers
- Work together

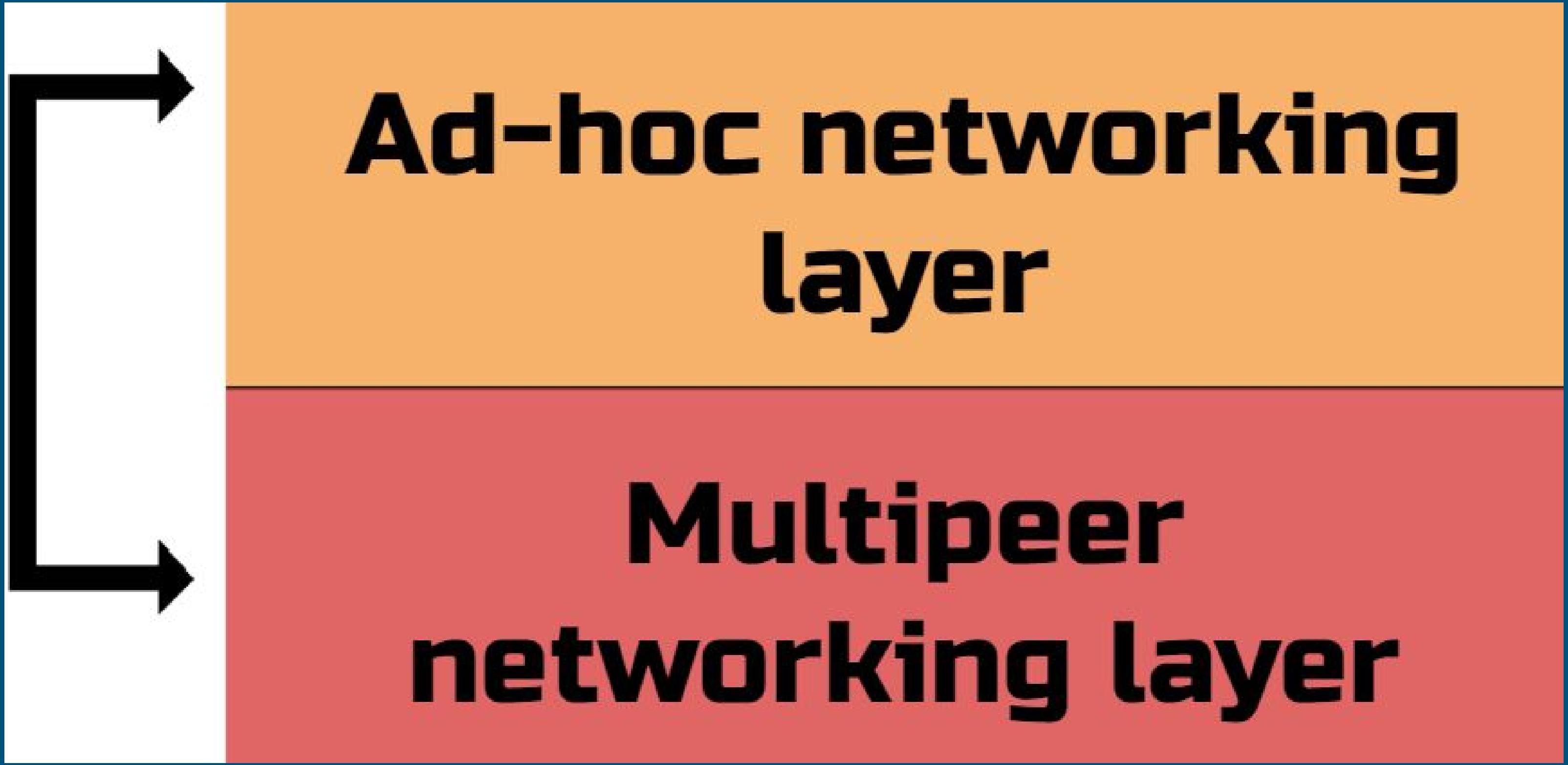
Multipeer Layer

- Swift code library that facilitates ad hoc networking
- Establishes connection with nearby nodes
- Does the actual sending of data
- Maintains list of local peers
- Passes the received data to ad hoc layer

Ad hoc Layer

- Responsible for generating heartbeat
- Decides what to do with data received
  - Propagate or discard heartbeats
  - Build routes from heartbeat
  - Determine the path for messages the node sends
  - Determine the next hop for messages
  - Display messages intended for current node
- Tells Multipeer layer who to send to next

Networking Layers



Messages

- Messages have headers to identify type
- Heartbeat messages - type 0
- Text messages - type 1
- Response messages - type 2
- Group messages - type 3
- Message handler class in ad hoc layer parses header
- Messages can be encrypted
- Nodes send confirmation of message arrival

Groups

- Initially going to have a large general chat
- Users can join a group to chat together
- In the backend single conversations and groups are implemented the same
- In the future passwords can be added to group chats to allow them to be used by groups like law enforcement

UI

- Created in Xcode’s storyboard
- Can also be programmatically generated and updated
- Many mandatory ID’s and connectors added in last Swift update and poorly documented
- Groups and users displayed in a list
- Standard chat window
- Group member who left had learned UI already

CoreData

- Storing persistent data
- Known issues added in Swift update, started switching to Realm
- Halfway through second semester Apple fixed CoreData issues

Encryption

- Two nodes over time create unique paths
- Both of the nodes share these unique paths
- Locally generate an encryption key
- The two nodes can now send encrypted messages

Unique Paths

- Unique paths occur when two nodes have a list of paths between them
- The two nodes are the only ones in every single path
- Use these paths as randomness to generate key

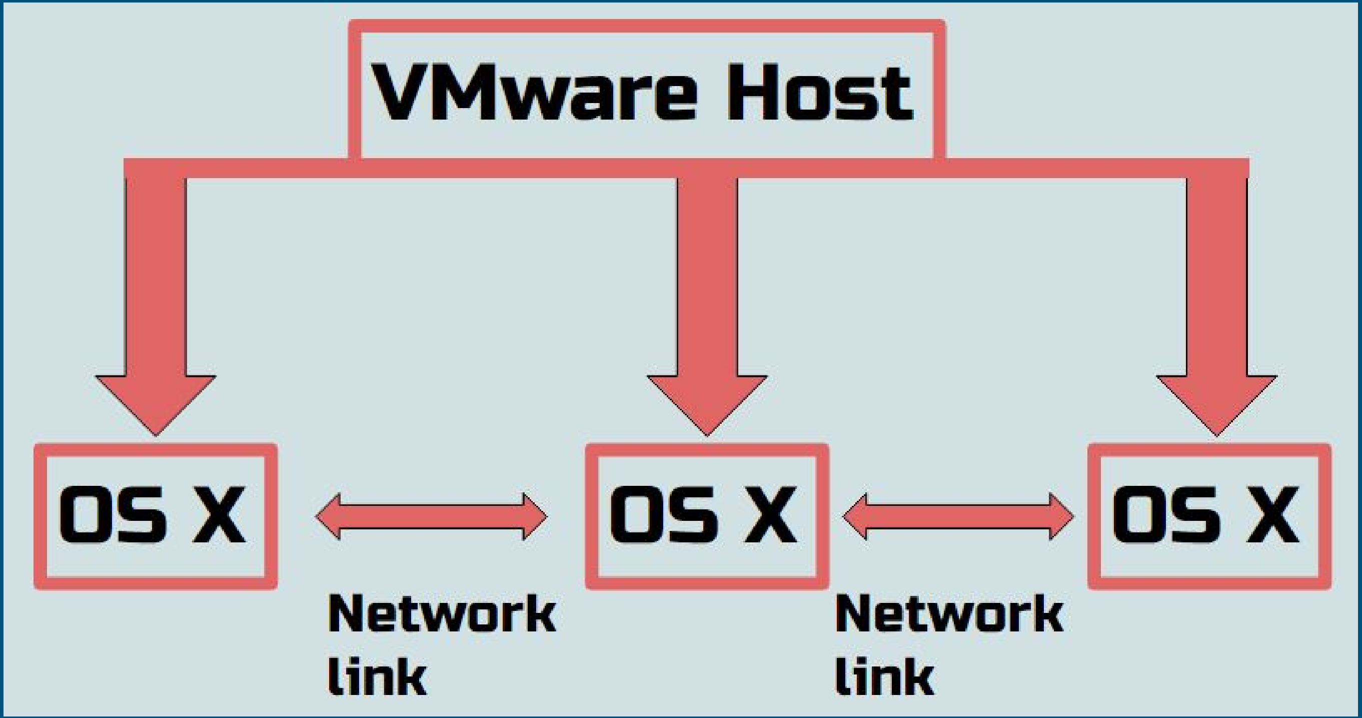
Testing

- Had to virtualize development environment due to Swift restrictions
- Made virtual machines running OSX
- Tested on iOS simulator most of the semester
    - Simulated 3 iPads and tested messaging between them

Testing

- Tested physically by spreading iPads across a distance
- Had to adjust when testing on physical iPads
  - Range on the iPads is far
- Multipeer worked well virtually, but not as well on physical hardware
  - iPads did not connect as easily on actual hardware

Virtual Network Setup



Retrospect

- Using a language that was in beta caused lots of issues
- Decided on Swift too quickly
- Apples' app development restrictions were a challenge
- Many aspects of the project were completely new to us
  - Gained experience in networking
  - More exposure to mobile development
  - Learned Swift programming language - interesting paradigm
- Had to reassess our knowledge and responsibilities when member left

Questions