Wireless Mesh Networking App

December 2015 - project 13

Group Members

Cole Cummings
Cody Lougee
Ethan Niemeyer
Holden Rehg

Advisor

George Amariucai

Client

George Amariucai

Table of Contents

Purpose of the Project

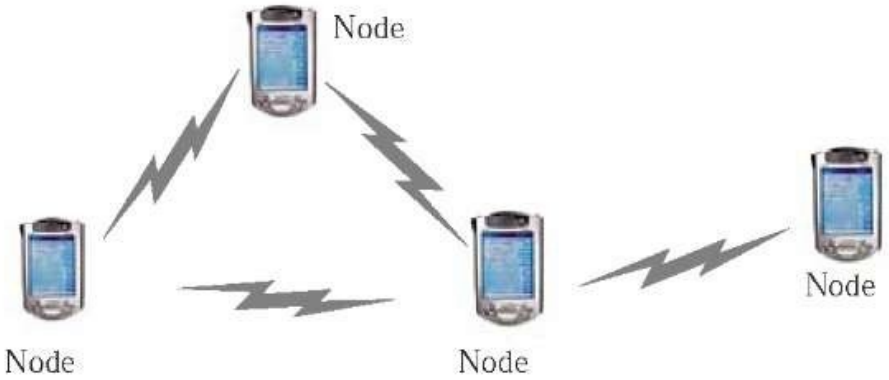
The purpose of the project is to build a messaging application for mobile devices that supports text, voice and video communications in the absence of a standard wireless network. The goal is to allow secure and encrypted communication between specific users. We believe this could be used in times when traditional networks fail, for instance natural disasters may destroy infrastructure like cell towers but responders need to be able to communicate. Some other applications have also been used at times when the government restricted access to traditional internet. Firechat was used in Iraq and Hong Kong protests but the communications are not encrypted.

Deliverables

An app for iOS that can securely communicate between mobile devices with the app while using the Dynamic Source Routing (DSR) protocol in a mesh network. Using this communication, the app will be able to transmit text, audio, images, and video.

Background

Ad-hoc or mesh networks are communication networks between multiple devices that transmit data between each other rather than a specific access point. This is illustrated in the picture below.



The way that this network will operate is by using the DSR protocol. DSR is intended to be used with mesh networks and works by building routing tables at each node. In this way, nodes will know the path to send a packet rather than determining the next step at each node. Additionally, the encryption will be based on the public key methods that are commonly used today.

Requirements

The messaging app must function by maintaining an ad-hoc network based on the Dynamic Source Routing (DSR) protocol. A specialized ad-hoc secret-key establishment protocol must also be implemented within the app to facilitate secure communications between the users.

Proposals/Solutions

Create the app for Apple devices based off of their provided Multi-peer libraries.

- Strengths
  - Only one standard of operating system to design for
  - Works with iPads
- Weaknesses
  - Only one type of phone usable
  - Requires the use of apple iOS
  - Learn Swift

Create the app for Android devices

- Strengths
  - More accessibility
  - AndroidSDK easily accessible
- Weaknesses
  - Requires user to root their device
  - Requires different rooting procedures/code for different devices
  - Requires design for multiple different versions of android

Validation and Acceptance Test

The app will be considered valid and complete when it meets the criteria listed in the requirements section. That is that the app will allow mobile devices to communicate text, audio, images, and video in an ad-hoc network without any outside communication through the internet or mobile networks. Additionally, the app should be able to encrypt the communication so that anyone listening in on the network will not be able to decrypt communications when encryption is enabled.

Technical Approach

1. Broadcast signal to available devices
2. Receive call back for possible routing
3. Maintain list of devices
4. Forward data from sender to node(s) to recipient
5. If someone drops reroute
6. Maintain security using a public key and private key approach

Process Details

In the current stage of production, the goal is to first establish a code base for our routing protocol. Off of that, we will then create an app that uses our routing code to send text messages over a mesh network. Once we

have basic communication working, we will add encryption. The last steps will be to add audio, video, and image exchange as well as just polishing the app.

Test Plan

In our early stages of testing, we will test the application by trying to establish an ad hoc network of simulated iPhones. Once we are able to acquire some Apple devices, we will test our application by placing the app on supported devices and have one device communicate with another that is out of its wifi range through an intermediate device.

We also plan to implement unit testing for each module in the mobile application, integration tests between the presentation and data layers, and code coverage analysis. We plan to test over 80% of the code written in the application before a final release.

All testing tools that we plan to use are built into the IOS framework.

Project Schedule

At this current point in time, our group was recently set back by our discovery that creating the app on android is not the best course of action.

- May 2015 – A mobile application with basic functionality using DSR and an adhoc network.
- Fall 2015 - Create Security protocols and implement data management .
- December 2015 - Finished product and app store release.

Market/Literature Survey

We researched other applications that use mesh networking on Android devices. The two most popular ones that we found were Serval and Firechat. We were able to look through the code for them and found a lot of restrictions for making the application in Android.

Firechat for Android claims to use mesh networking to allow devices that are close by to communicate but what we found is that it uses BlueTooth, which relies on a master/slave architecture and slaves cannot connect to one another and a master can only have up to 7 slaves. If any devices have an internet connection they try to be the master and then communicate with servers over the internet to allow more than 7 nearby people to chat, which isn't a true mesh network. The company that makes Firechat claims they have created a mesh networking protocol that even allows Android and iOS devices to form mesh networks together, but they have not open sourced the project yet and so we are not sure how they did this and they may still be using some server communication.

Serval is an Android project to create a true mesh network of devices, but they found limitations with Android's current peer-to-peer service, like it only allows one device to communicate with up to one other device at a time and the user must manually accept all connections. Serval is getting around these issues by rooting the phones and changing or adding to the existing peer-to-peer functionality but doing this means that the code must be

written specifically for the hardware it is running on. Serval currently has a list of 5-8 phones it does work on and about 15 it doesn't. We want to create an app that can be used universally without rooting the phone or writing phone specific code, and it seems that is currently impossible on Android.

The SPAN project is an Android framework that was originally meant to provide ad-hoc support to Android devices for developers. Their end goal is to seamlessly provide this support without rooting a device or altering the Android kernel but at the moment they still must. They based their project off of two mesh networking technologies, project BATMAN and OLSRd.

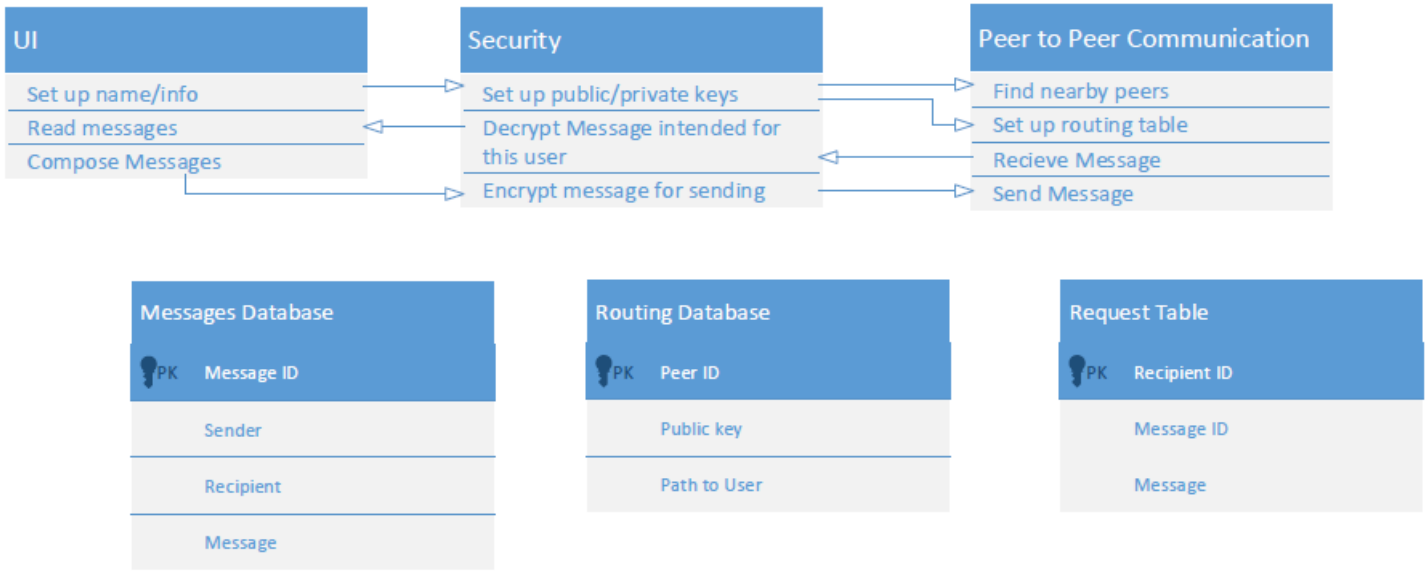
Feasibility

After extensive research we found that an ad-hoc network is impossible to produce on the current operating systems of Android without altering operating system software. This would require rooting each potential device, and writing patches for multiples types of hardware to support ad hoc networking. We moved to iOS for its Multi-peer connectivity feature. Multi-peer is an Apple created library for ad-hoc networking. It offers all the supported features needed for the application. It can send video, images, audio, files, and text securely.

Cost Considerations

With the requirement of Mac devices for both creating and testing, the purchasing of ipad or iphone devices may be required. The pricing of an Ipad is a range from $249-$699 per device. It is difficult to purchase an iPhone without a cellular contract.

Design Diagrams



In the UI the users will have to set up a name for themselves, then the Security module will generate public and private keys for them. After that it will pass them to the Peer to Peer module to find nearby peers and establish the initial routing table with information received from them. After these steps are completed the user is a member of the network. In the future they should be able to automatically connect because the set up identity step only needs to be completed once.

When the Peer to Peer module receives a message it will check if the intended recipient is the current user. If it is it will decrypt the message and display it in the appropriate conversation. If the current user is not the intended recipient it will update its routing tables with where the message came from/is headed. It will then send the message on according to its own routing tables or broadcast it if it does not have a path to the intended recipient.

When the user composes a message it will be encrypted so that it can only be read by the intended recipient. After this it will be sent to the relevant users according to the routing tables or broadcast if the intended recipient is not in the routing tables.

Conclusion

Creating an adhoc-network capable of functioning without a central router while maintaining user security is feasible and could be used in areas where there is low coverage, no coverage, or filtered coverage. It can be done using Apple's Multi-peer ad-hoc networking library. While the general availability of Android devices makes it a desirable platform to deploy the application on, it is currently infeasible to create an ad-hoc network using Android devices. This is due to the need to root devices and write specific software for each Android phone and version of Android.