Android Walkie Talkie - First Project Report

ECE/CSC 575, Dr. Sichitiu

North Carolina State University

February 21, 2013

**Authors - Team A3:**

Andrew Davis (*ajdavis7*)

Travis Folsom (*twfolsom*)

Deenan Ravindra (*dravind2*)

Dustin Swisher (*dmswishe*)

**Table of Contents**

[Problem Statement](#h.inp5fe7le0nx) 1

[Proposed Solution](#h.79btht6neiu) 2

[Stretch Goals (Bonus)](#h.a4coug5simh2) 3

[Division of Labor](#h.akt91ybjy4a0) 4

[Tasks](#h.kgtk4dzcz48t) 4

[Assignments](#h.s3qqtbxj85ph) 5

[Dependencies and Risks](#h.ow30lrfo6fde) 5

[Project Timeline](#h.b9xnjlq90vss) 7

# Problem Statement

The problem presented is to create a software system for Android operating system-based devices to communicate in a manner similar to walkie talkies. Specifically, it is desired to emulate the push-to-talk of a walkie talkie, meanwhile allowing for protected control of who can communicate in a particular “channel.”

The following features are required:

* Users must be able to join arbitrary communication groups (in one or more of the following ways):
  + Specified via password
  + Transmitting to user-specified recipients
  + Pre-configured based on device
* In a communication group, while one user talks, all other microphones are disabled
* Must function over WiFi

The following assumptions are made:

* All users are on the same IP network
* A user can be a member of multiple groups, but only active in one at a time

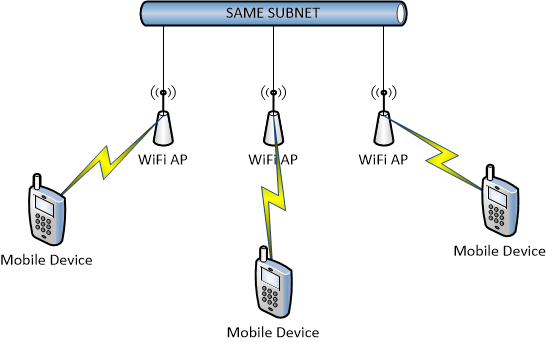
# Proposed Solution

The proposed solution aims to achieve the requirements via a decentralized WiFi infrastructure mode. Specifically, it is envisioned that the solution can operate minimally with two mobile devices and single wireless access point, with support of up to tens of mobile devices and multiple wireless access points (as long as the access points are also networked together and all are configured to use the same subnet).

A use case of the system is as follows as situated on the main NC State campus. User 1 is in Nelson Hall, while User 2 is in Riddick Hall. Each user is connected to the NC State WiFi system (and is miraculously connected to the same subnet). User 1 wants to start a walkie talkie session, so he starts his Walkie Talkie application and presses an on-screen button to scan for users. User 2’s device sees this request and responds back to User 1’s device saying it is available. User 2 shows as an option on User 1’s group selection screen, so he selects User 2 as his group member, and initiates the conversation. User 2’s device receives the initiation, and acknowledges to User 1’s device that User 2 is ready. Communication may now begin.

When a user presses the button, it broadcasts to the subnet that someone in the group wants to talk, so the other devices must remain silent. Meanwhile, the application informs the user it is negotiating the medium. Once the medium is clear, the user is free to talk. The communication ends when the user broadcasts a “done” message, or after a timeout period (see risks section).

There are of course additional use cases that can be constructed from variations of the scenario (use of a group password instead of directed communication, etc). There are also use cases for initial setup of the device, such as establishing the user’s profile and preferences. The following figure (Figure 1 on the next page) depicts the possible topology of the core solution for three devices. Of course, multiple users could be accessing the same access point, or there could be multiple access points unutilized -- as long as the devices are ultimately connected to the same subnet with sufficient performance, the system should function as expected.



**Figure 1:** Sample topology of the core system.

## Stretch Goals (Bonus)

There is also consideration for stretch goals in order to achieve bonus points. These features include development of a central server for service discovery, wide area network compatibility (for Internet/EDGE), and encryption. Out of the ten possible bonus points, the following suggestions are made:

* 1 additional point for industry-standard encryption that allows only group members in the same group to decrypt the communications
* 2 additional points for implementation of WiFi ad-hoc mode (inclusive or exclusive of above feature)
* 3 additional points for implementation of a central server on the same subnet which allows for full connection brokering (inclusive or exclusive of above features)
* 4 additional points for implementation of the solution over a WAN/Internet, using a central server, whose performance is comparable to similar VoIP solutions

# 

# 

# Division of Labor

## Tasks

There are first a couple standard software engineering tasks that must be addressed:

1. Perform market research to understand how this problem has been solved before. Determine appropriate libraries, frameworks, APIs, etc. Ensure any licensing restrictions are not violated.
2. Software configuration management and documentation: setup and manage code repository, defect tracking, in-code documentation “grooming,” general documentation.
3. Performance testing and hardware/software compatibility checks

In regards to the user interface, there are several more tasks and sub-tasks that must be accomplished:

1. User interface development
   1. Device settings/preferences and user profile
      1. Name/ID/Alias
      2. Network settings
      3. Authentication
      4. Button assignments
   2. Creation and management of groups
   3. Walkie Talkie functionality
      1. Push-to-talk network hooks
         1. Network health/status
         2. Peer discovery, group membership
      2. Codec (programmatic) interface
      3. Conversation settings
         1. Network health
         2. Muting, volume
         3. Current status
2. Application protocol design and implementation
   1. Transaction (protocol) definition
      1. Peer discovery
      2. Group membership
      3. Connection (re)negotiation
      4. Voice data codec
   2. Framework/API for user interface implementation

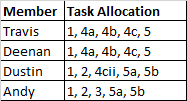
The following rough tasks are provided for the stretch goals:

1. Implement central server
   1. Architecture (tech stack)
   2. Accommodate for protocol changes
   3. Additional features
      1. Advanced grouping
      2. Encryption
2. Ad-hoc mode
   1. Device discovery
   2. Radio strength (multi-hop?)
   3. Root the phone

## Assignments

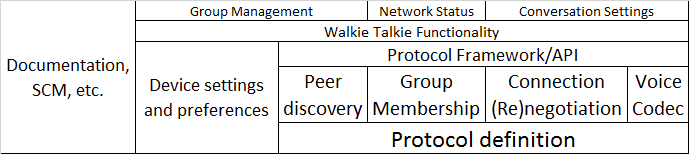
There are a couple aspects of the project which will be the shared responsibility of all team members. We all plan on helping with the requirements and design of the application’s protocol, testing activities, and any documentation relevant to each member’s code. Past that, the members have decided to divide the work in the following manner:

Travis and Deenan will focus on development of the user interface and hooks into the network protocol (tasks 1, 4a, 4b, 4c, and 5). Dustin will focus on the audio codec (and relevant protocol hooks), as well as managing the code repository (tasks 1, 2, 4cii, 5a, 5b). Andy will focus on implementation of the protocol, testing, and documentation (tasks 1, 2, 3, 5a, 5b). Despite these assignments, the team feels all should gain experience in all aspects of the project, so task responsibility will be shared and knowledge transfer performed as appropriate. A summary of the assignments is provided below:



## Dependencies and Risks

The figure below (Figure 2 on next page) depicts the broad dependencies of the system. As can be seen, much of the system depends on the protocol definition at its heart, but in reality much of the user interface can be developed separately (in parallel) from the protocol. The only true dependency would be of the user interface on the integration points with the framework/API of the protocol implementation.



**Figure 2:** Diagram of the development dependencies. Items on the top are dependent on the corresponding blocks below it.

There are several risks that this project could face given the scope of the proposed solution. First and foremost is the same risk any software development project could face: capacity. Due to unforeseen personal circumstances or computer downtime, the development teams capacity could be reduced. To mitigate this particular risk, we have tried to keep our project scope realistic, and are allowing extra time near the end for non-core functions.

In regards to the application itself, one of the risks identified is that of un-brokered groups. Consider the scenario of two mobiles creating a group literally simultaneously with the same name. In the worst case scenario, the group just will not be allowed to be created because the “other” mobile will reply with a negative acknowledgement. In the best case scenario, network latency will allow one mobile to edge out the other, thus allowing the group to be created.

Another consideration when designing the application’s communication protocol will be network quality variables like bandwidth, latency, jitter, and topology. Depending on the configuration of the networks utilized (such as in the multiple access point topology proposed earlier), these variables could be very detrimental to conversation quality. Care must be taken in order to ensure quality of service and that bandwidth capabilities are understood; one mobile might have a lower bandwidth connection and thus might not be able to handle multiple simultaneous data connections. Also, we must take into account users who may drop their connection inadvertently, and provide timeout mechanisms should any acknowledgements fail to be sent.

Risks previously identified are “known unknowns.” However, there is always the risk involved in starting a new project and analyzing the tools and information available. Particularly, there may be limitations that are discovered in hardware or software which severely hinder effective development of a feature, such as ad-hoc mode requiring root access or a convenient codec library’s licensing restrictions. As such, these risks must be taken on a case-by-case basis, and can only be anticipated as “known unknowns.” These risks are addressed in the next section’s timeline.

# 

# 

# Project Timeline

A rough project timeline is shown in the table below. The exact project milestone dates are unknown, so estimates were made. The schedule was made taking into account all “known knowns” (members’ work/school schedules), “unknown knowns” (travel for work/interviews, personal circumstances), and “wiggle room” for “unknown unknowns” (completely unforeseen circumstances).

|  |  |  |  |
| --- | --- | --- | --- |
| **Week Of** | **Description** | **Effort** | **Success Criteria** |
| 17-Feb | **First Project Report** | Discuss problem, share vision, establish development goals | Documentation produced |
| 24-Feb |  | Gain familiarity with Android SDK, get acquainted with software libraries | All run "Hello World" and know what libraries are being used |
| 3-Mar |  | Development begins | n/a |
| 10-Mar | **Milestone 1** | Development continues | Roughly functional UI |
| 17-Mar |  | Development continues | Mostly functional UI |
| 24-Mar | **Milestone 2** | Development continues | Voice data transceiving proof-of-concept |
| 31-Mar |  | Development continues | Voice data/UI integration |
| 7-Apr | **Milestone 3** | Development continues | UI/protocol refinement |
| 14-Apr |  | Development finalizes | UI/protocol refinement |
| 21-Apr | **Final Report** |  | Complete, on-time submission |