COMMUNITY AWARENESS SYSTEM FOR ANDROID DEVICES

Design Document (VERSION 2.0)

Group : DEC1618 (dec1618@iastate.edu)

Clients : Dr Daji Qiao, Dr. George Amariucai

Advisors: Dr. Daji Qiao, Dr. George Amariucai

Team Members/Role:

Jason Wong: Team Leader
Erik Fetter: Communication Leader
Matt Gerst: Team Webmaster
Shikhar Vats: Key Concept Holder
Brad Anson: Key Concept Holder
Adit Kushare: Key Concept Holder

Contents

1 Introduction

1.1 Project statement

The National Security Agency has expressed an interest in developing a community awareness system for Android devices. In result, Dr. Daji Qiao and Dr. George Amariucai have been assigned to research and create a working prototype of such a system through our senior design project. The purpose of this project is to provide a squad of US soldiers the ability to create a secure network using their Android devices. Each soldier would carry an Android device and have the ability to view the approximate location of their squad members and query for information from these devices. The information to be queried includes the device's battery level, GPS coordinates, accelerometer reading, and 5 second audio/video clips from the target device. The query must return the information within 10 seconds.

1.2 Purpose

This networking component of this technology could potentially be used for military purposes. For example, a platoon of soldiers could be deployed in the field with their devices all networked together. The end goal is to have the network provide a medium of transmission for vital information such as troop movement and location. Additionally, this project allows for more interconnectivity among people in remote areas. Often times remote underdeveloped areas don't have the infrastructure to support wireless communication. The high cost of deploying routers and towers and the low mobility often makes this technology not applicable/feasible. However with mesh networking, we are now able to bring a flexible network that is not only mobile but also a cost effective solution to wireless networking between devices.

The data collection portion of this project will be fed into a Biometric Engine being developed by a separate development team. The data to be collected so far includes accelerometer information, GPS location, 5 second video and audio clips, etc.

The Biometric Engine will use the data collected in the app to verify the identity of the person using the phone and their surroundings. For example, audio could be processed to indicate possible problems such as gunfire, unknown voices, or used to help locate lost soldiers.

1.3 Goals

Project Goals:

1. Successful proof of concept: The main goal is to showcase a working proof of concept to the NSA and have our project move closer to being deployed to real US soldiers.
2. Soldier safety: The main purpose of our project is to give troops a way to share vital information with their fellow troops such as relative location, audio and video snippets, and other information that can be used to calculate the safety of a soldiers well-being.
3. Cheaper communications: A possible implication for our project is to allow communications in remote areas or third world countries where there is a lack of infrastructure that supports communications. Cell towers, telephone lines, and other infrastructure have high costs and having the ability to create wireless networks of communications can be a cheaper alternative.
4. Publish our application to the Google PlayStore: Another possible outcome of our project is to publish our application to the Google Play Store for the general public to use.

Learning Goals:

1. Learn Android: Few of our team members are new or inexperienced in Android development and this project would give us a good way to learn Android.
2. Learn Network Principles: Many of us are not familiar with basic networking principles that will most likely be needed to complete our project such as IPv4,

TCP, UDP, routing protocols, etc. Learning these networking principles are aligned with our interests and will help us in developing a quality product.

3. Learn Team Dynamics: All of us are aspiring to be successful engineers and we recognize that learning to work in a team is an important skill.
4. Learn Agile Development: Our goal is to approach this project with an Agile development lifecycle. Many of us have learned the basic principles of Agile during our coursework but have never applied it to a long term project such as this.

2 Deliverables

We will deliver an Android application which can create and maintain an ad-hoc mesh network with other Android devices as well as have the ability to collect and distribute various sensor data around the created network. For our project, we will also create a visualization portion that will show the network topology, users in the network, and sensor data collected from those users.

The ultimate goal is to integrate our network application to allow other Android applications to utilize the mesh networking capability. In result, along with our application we will provide an API in which other applications can connect to and utilize our application's networking capability.

3 Design

3.1 System Specifications

3.1.1 Non-functional

- Periodic network updates should be quick without a major battery drain
- Network should be resistant to device loss by using the mesh topology

- Users should know in a reasonable amount of time if a device has left the mesh
- Remote data collection should be received in a reasonable amount of time (5-10 seconds)
- Collection of data should occur in the background, without UI interaction
- Android codebase should largely be modular and good/best practice (aside from the necessary rooting of the devices)
- The application does not need to scale very large, but it should be able to handle 10-20 devices on the network.

3.1.2 Functional

- A user should be able to find out who else is on the mesh network
- Users should be able to request information about other users on the network
- Network module should provide as many metrics, such as RSSI, if possible
- Users should be able to see a network topology, possibly organized to represent relative positions, which shows how many hops to another user
- The Application should expose both a network API usable by other applications and a sensor/inference API for the inference engine

3.2 Proposed Design/Method

This project can be divided into three distinct interconnected system layers:

- Sensor Layer: The sensor layer essentially has functions to collect the raw data from the various sensors on the android device. This information from various sensors including, but not limited to, accelerometer, gyroscope, barometric sensors, is analyzed to generate a confidence number which would be used for communication with the other android devices. The results from the sensor layer are transmitted to the network layer which then uses the information to communicate with the other devices on the network.
- Network Layer: This layer encapsulates the core functionality of the project. The android devices in our system would be interconnected using wireless ad-hoc

network mesh topology. The network layer is responsible for establishing and maintaining a connection between the android devices in the system. The higher level application layer queries information before transmitting the information, the network layer will query the sensor layer, which replies with the analyzed sensor data (see Sensor Layer). The network layer on the receiving device receives the information, and then transmits it to the higher application layer.

- Higher Level Application: This layer acts as the UI layer for the application. The flow of queries and information starts after the higher level application layer queries the network layer for information from any other device in the mesh network system.

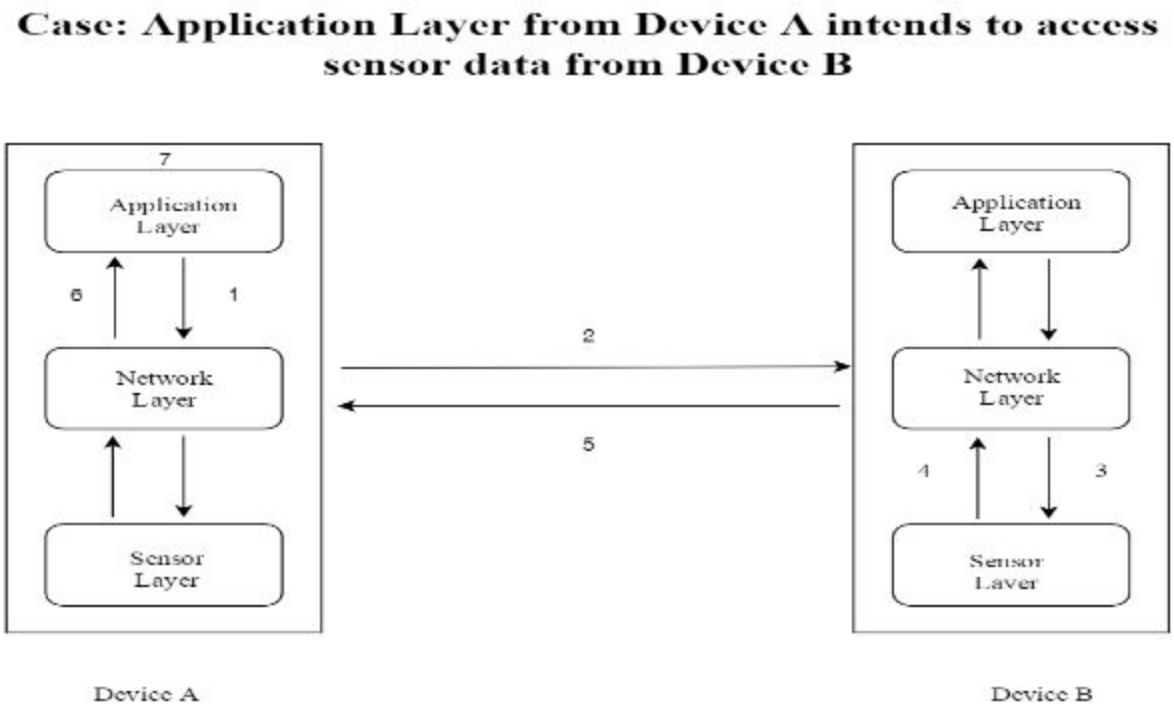


Figure 1 Flow of information between two devices on the network. Src: self

3.3 Design Analysis and Current Work

- Received Signal Strength Indication (RSSI)
    - This essentially tells the user about the proximity of any other device in vicinity. Wifi RSSI is normally available via normal Android API; however, in ad-hoc mode this is not possible. To solve this problem we looked into alternatives such as native libraries like iwlist and iwspy. Unfortunately they were not supported by our testing device (Nexus 7) hardware. To

solve the problem we decided to use Bluetooth RSSI. Even though helpful, using the Bluetooth RSSI values has its own disadvantages like:

- A shorter range than the wifi RSSI
- Different android tablets/phones may have different Bluetooth chipsets which would give different RSSI values under the same conditions. We want to avoid that.
- Bluetooth RSSI value between two "visible" android devices depends on several factors; battery level of the phone, and the orientation of the device are among them. This essentially means that at two different time instants the value for RSSI returned would be different for the devices in the same position.

Despite these drawbacks, we have decided to move forward with the Bluetooth RSSI. The reason being that for our current needs we need a way to tell the user about the distance of the other devices in the network relative to his/her position which can be achieved using our approach. We are simultaneously also looking into figuring out a way to make RSSI work with the ad-hoc wifi mode.

```
mReceiver = new BTBroadcastReceiver();
Intent intent = new Intent(this, BTBroadcastReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(this.getApplicationContext(), 0, intent, 0);
alarmManager.setExact(AlarmManager.ELAPSED_REALTIME_WAKEUP, SystemClock.elapsedRealtime() + 3000, pendingIntent);
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

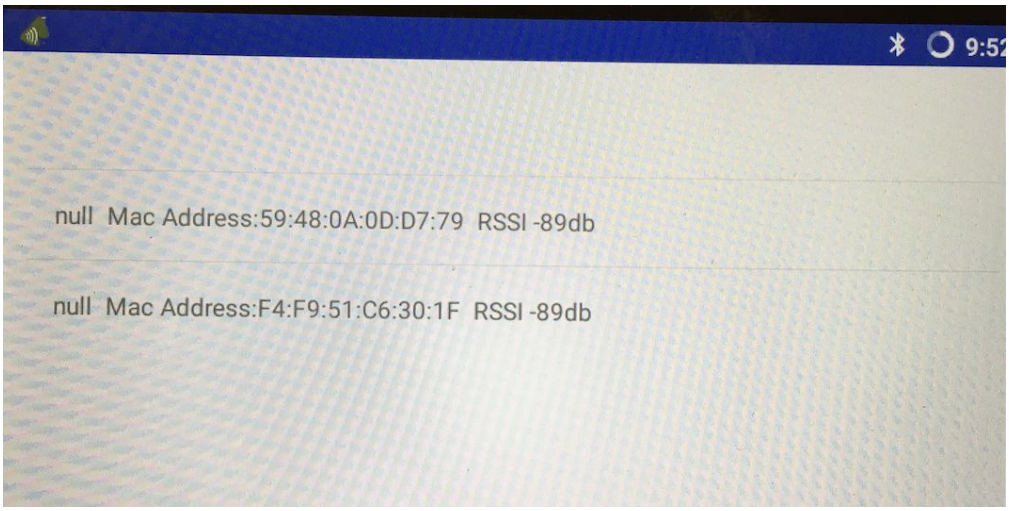
Registering the Bluetooth discovery BroadcastReceiver with the activity. src: code

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tv = (TextView) findViewById(R.id.textView);
    list_view =(ListView)findViewById(R.id.listView);
    list_view.setAdapter(BTAdapter);
    toggle = (ToggleButton)findViewById(R.id.toggleButton);
    BTAdapter = new ArrayAdapter<>(this, R.layout.activity_main, R.id.textView, list);
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if (mBluetoothAdapter == null)
        Toast.makeText(getApplicationContext(), "Device doesn't support Bluetooth", Toast.LENGTH_SHORT).show();
    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableBTooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBTooth, REQUEST_ENABLE_BT);
        Toast.makeText(getApplicationContext(), "BT enabled", Toast.LENGTH_SHORT).show();
    }
```

Creating an adapter for each Bluetooth device to discover. The intent is used to enable the bluetooth on the device. Src: code

```
class BTBroadcastReceiver extends BroadcastReceiver {
    int count = 0;
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "Command Received", Toast.LENGTH_SHORT).show();
        String action = intent.getAction();
        String textView = "";
        String split[]={};
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            short rssi = intent.getShortExtra(BluetoothDevice.EXTRA_RSSI, (short) 0);
            count++;
            //textView += "" +  + "\n";
            Toast.makeText(context, "Device Found", Toast.LENGTH_SHORT).show();
            //Toast.makeText(context, textView, Toast.LENGTH_LONG).show();
            trial.add("" + device.getName() + " Mac Address: " + device.getAddress() + " RSSI " + rssi + "db");
            if (count == 0)
                BTAdapter.add("No device found");
            for (int i = 0; i < trial.size(); i++) {
                if (list.get(i).contains(device.getAddress()))
                    BTAdapter.add("Nopes");
                else {
                    BTAdapter.add(trial.get(i));
                }
```

The BroadcastReceiver class that receives the request from the activity to search for bluetooth devices and display the RSSI values. Src: code



Trial run of the application returning the name of the discovered device, the MAC address of the device and the Bluetooth RSSI value respectively. Src: code

- Interapplication Communication
    - In order to create a communication interface between the different layers such as the network and application layer we have explored possible solutions via Android's available API. We have found that Android services and BroadcastReceivers allow us to create this interface. Services and BroadcastReceivers run in the background which works perfectly for the

sensor request feature where essentially we would want the services to perform the sensor capture function hidden from the user once the user gives the command. This adds a sense of data encapsulation to the application.

- Video/Audio Capture
    - We have made progress to perform 5 second video and audio captures. The application is now able to send a request to another device to order their camera/microphone to perform a capture and send back the recorded file to the source device. Our next step is to figure out a way to perform this feature when one of the phone is in sleep mode i.e. when the phone's screen is off. For example, when a soldier has their phone in their pocket but another soldier in the mesh network wants to record 5 seconds of audio, the phone in the pocket must know when to wake up to respond to the query. We are exploring a lead on android's AndroidManager class to figure out a solution to the problem.
- TP-Link TL-MR3020
    - We ordered 3 routers to test out the ad-hoc mesh networking when offloaded to the routers. We have run basic testing procedures by using the routers along with the current ServalMesh application. The tests have been positive so far, indicating that the application is supported by the routers. Our next step is to perform rigorous testing in this regard.

4 Testing/Development

4.1 Interface Specifications

Software API

- Topology Class

    - getHopCount(MeshMember): we should be able to return the number of hops away a member is in the mesh network.
    - getDirectNeighbors(): we should be able to distinguish which members of the mesh we are directly connected to and return this list.
    - getSignalStength(MeshDirectNeigbor): we should be able to return the RSSI value of a direct neighbor.
- Sensor Class
    - captureVideoClip(): this should cause the phone to capture a 5 second video clip even if the screen was off.
    - captureAudioClip(): this should cause the phone to capture a 5 second audio clip even if the screen was off.

4.2 Hardware/Software

For the testing phase, our requirements can be categorized into hardware and software categories as follows:

Hardware:

- Tablets/Mobile phones based on Android OS and having root access
    - Nexus 7 (at least 4)
    - Htc One (at least 2)
    - Nexus 6 (at least 2)
- TP-Link TL- MR3020 routers (at least 3)

Software:

- A working prototype of the android application

For a successful testing of our product, we need to test our prototype application across as many android devices (with at least API 14) from different manufacturers as possible. The reason behind is to ensure the compatibility of our application with the other android devices. Further we need our android devices to have root access. The reason behind this is that android devices by default do not support ad-hoc networking. Since our application heavily depends on ad-hoc networking for the mesh network,

therefore this is one of the core requirement for any android device to be used in the testing phase.

We have two approaches for the final deliverable product. The first one is to deliver an android application which has the root access to the devices. The requirements for which were discussed in the previous paragraph. The second approach is to use wireless routers with each device that would be responsible for the ad-hoc networking part. In addition to the routers we'll still need android devices, but now we do not need the android devices to have a root access.

4.3 Process

Serval project was always a key component of our project and was going to be underlying piece of code that enabled us to communicate between two devices directly. Some of the thought process behind the hardware choices made so far is explained in the following paragraph.

We started out with the goal of making the serval project work on non rooted android devices at the application layer. We quickly realized that this was not possible hence we rooted some Nexus 7 devices and installed Cyanogen Mod on them. With serval mesh up and running on the Nexus 7's our next step was to test serval mesh on unsupported devices. To that effect we ordered two Nexus 6's and two HTC one phones. We also ordered battery powered routers as a part of plan B in case it doesn't work on the unsupported devices. As of now we have been able get the serval mesh app working on the unsupported devices as well, with some glitches here and there. The 3 routers ordered also have been able to work great. Below we discuss some of the processes behind the software side of the project.

One of the first tasks was to create an environment in android that would be able to build the serval project. After the environment was set up we started making progress on two fronts, namely networking and sensor data collection. On the network side of things, the first goal was to integrate the batphone part of serval mesh into our android project and then progress onto other networking goals such as gauging the device

topography by acquiring the distance between two devices using RSSI (Signal strength indicator). On the sensor collection side of things the first goal was to familiarize ourselves with collecting various sensor data with the use of a button in an android application. Then we moved on to collecting this data in the background with the use of a service. From here on out we are working toward establishing an API of sorts that will make use of inter application communication to send and receive relevant sensor data making use of the mesh network. Also we are working toward creating a topological display using the signal strength data.

5 Results

So far we have been able to test a few different devices to see if Serval would work as well as purchasing standalone routers to help increase the usability for our deliverable. Of the first 4 devices we purchase we bought 2 of 2 different types. One of the types will not currently work with Serval so we will probably not be able to use these devices for our project. We have done some work in installing the necessary software to the routers so we may be able to use them and subvert the Serval issue but there are also some design drawbacks we are considering.

Currently we are testing a couple of different possible communication methods and will adjust to pursuing one when we can determine the best route for us to take. Once we complete our initial demo and project, we will be revising our code and improving upon the look and optimization of our application and libraries.

6 Conclusions

To summarize, we are feeling confident about our approach so far after taking into consideration the work done so far and the results from testing the software and the hardware ends. The best solution currently seems to use the routers to create an ad-hoc network and then use bluetooth to map the topology. This solution would help us

overcome one major concern of rooting the android devices, thus avoiding the glitches on unsupported devices. We are still learning newer and better alternatives as we progress through the project. We are confident that with continued progress and through multiple iterations we would find reasonable solutions to our concerns.

7 References

1. CyanogenMod/Servel Mesh
    - Nexus 7 Rooting Instructions: http://www.teleread.com/uncategorized/how-to-root-and-mod-a-nexus-7-2012/
    - CyanogenMod Instructions: https://wiki.cyanogenmod.org/w/Install_CM_for_grouper
    - CyanogenMod Version: cm-12-20150625-SNAPSHOT-YNG4NAO09M-grouper.zip
2. Mesh Extender
    - http://developer.servalproject.org/dokuwiki/doku.php?id=content:meshextender:prototyping_on_mr3020
    - http://www.amazon.com/TP-LINK-TL-MR3020-Wireless-Portable-Router/dp/B00634PLTW
    - http://www.ebay.com/itm/2pcs-RFD-900-MHz-Ultra-Long-Range-Radio-Telemetry-Modem-with-FTDI-Antenna-IN-US-/301579827973?hash=item46378efb05:g:9UQAAOSwqu9VGkS4
    - http://www.amazon.com/SanDisk-Cruzer-Low-Profile-Drive--SDCZ33-016G-B35/dp/B005FYNSZA/ref=sr_1_2?s=pc&ie=UTF8&qid=1454221838&sr=1-2&keywords=SanDisk+Cruzer+Fit%E2%84%A2+tiny+USB+memory+stick
3. Serval Project : http://developer.servalproject.org/