CSC714 Project Report 3

Android WIFI Social Network

Team:
Phil Marquis - ppmarqui
Sushmita Lokala - slokala

Project URL : http://www4.ncsu.edu/~ppmarqui/csc714/

Proposed Project:

We have used a Google Android phone (the HTC G1) to implement a scheme for sharing local wifi signal strength among other Android users.

A user's phone is able to publish the signal strength of wifi networks in the area to a centralized server. Other Android users are able to view this information if a better wifi location is desired. Wifi data are be presented on a Google Map, to indicate other users' wifi statuses.

Outline:

For the sake of simplicity, we have chosen a linear execution path:

1. Get wifi signal strength of the current location.
2. Read the current location in terms of longitude and latitude using the phone's GPS hardware.
3. Read friends' wifi signal/location history from disk.
4. Send updated wifi/location data on the server.
5. Retrieve friends' history from the server.
6. Save this information on the disk to display it the next time the application runs. This is helpful when the user is in a place without internet access and cannot connect to the server. The most recently updated set of friends' data will be displayed, so that a wifi connection can be located nearby.
7. Display the user history data on a Google Map, with a unique color for each user. The map will display the user's name, the name of the wifi network with the strongest signal, and the signal strength on a scale from 0 (weakest) to 10 (strongest).

Detailed Description:

Step 1:
The android SDK provides a WifiManager class that allows our application to scan for all wifi networks in our current location. From the Results retrieved by the ScanResult object of this class, we store all available networks and their signal strengths in a data container.

Our data container classes consist of several layers of detail to facilitate the handling of a user's wifi history. The WifiNetwork class is very much like the ScanResult class, containing a wifi network's SSID and signal level. The WifiData class encapsulates several WifiNetwork objects along with a location at which these networks were assessed. Finally, the WifiUserHistory object provides a means for tracking the locations that a user has visited by maintaining a collection of WifiData objects. All of these classes implement the Serializable interface, so that they can be serialized for the purposes of reading/writing from/to disk and for server communication.

Step 2:
We procure the longitude and latitude readings from the G1's GPS hardware using the Location class. The method getLastKnownLocation() returns the coordinates of the phone at the last successful GPS retrieval.

Unfortunately this GPS functionality does not perform as well as expected. The phone struggles to obtain GPS coordinates, so the user's location may be slightly out of date. In addition, the GPS function is very slow and maybe be a significant burden on our application's run time. That is, pausing while the phone determines its location may result in unacceptable delays in execution.

If the phone were connected to a cellular network, it would most likely be able to utilize Assisted GPS technology. That is, the phone uses the GPS system in conjunction with a cellular provider's towers to determine its location far more quickly.

Step 3:
The application will read a serialized array of WifiUserHistory objects from the phone's file system. This object is the most recent set of friends' data that was obtained from the server. These data are available for display even if an internet connection is unavailable.

In our current implementation, the code assumes that this object already exists on the file system. The first time the application is run, the code to save some dummy data to disk must be manually invoked (by uncommenting it in Main.java).

Step 4:

After we have obtained our coordinates and the current status of any wifi networks at the present location, we update the user's WifiUserHistory and send it to the server. The server finds the user's name in its cache and updates the data.

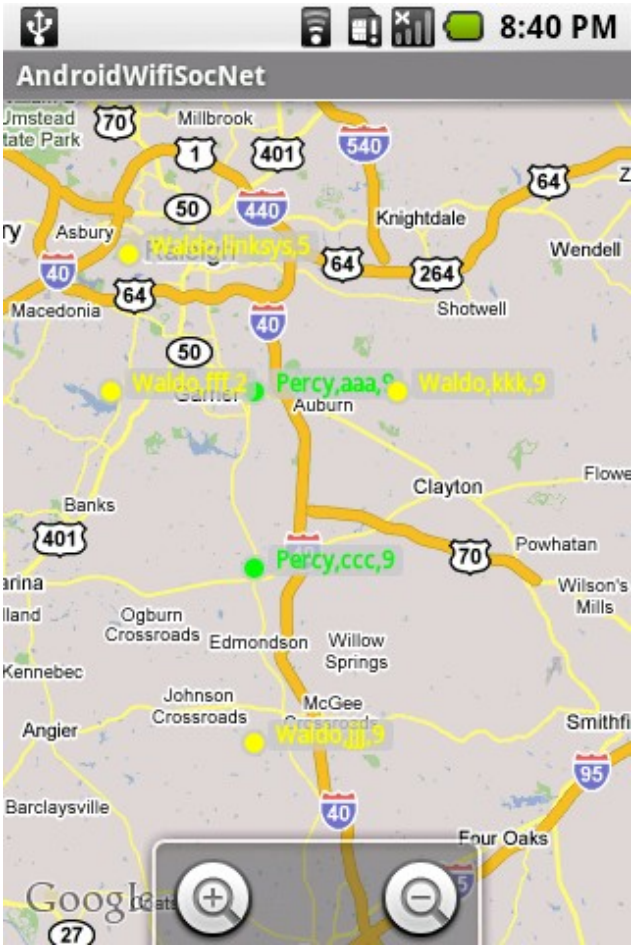
Step 5:
The phone sends a request to the server for the friends' data. The response is a UDP packet that contains all of the WifiUserHistroy objects on the server.

Step 6:
If we have successfully received updated data from the server, we will save it to the disk for future use. This is done by serializing the aforementioned array of WifiUserHitory objects.

Step 7:
The Android SDK provides a MapView class to display a Google Map. We have a zoom Controller on the map to zoom in/out on the map. We have defined a class called the WifiOverlay that stores the wifi history of all the friends in a HashMap. This class extends the Overlay class in the SDK and overrides the draw method to display ovals and text (for the user name and signal information) in all the places recorded for that particular friend. To make out display more user friendly, we have assigned different color codes to each user's overlay.

Results

A screen shot of our application:



Milestones

Task	Assigned	Deadline
Install Eclipse+SDK on Linux	Phil	April 1
Install Eclipse+SDK on Windows	Sushmita	April 1
Research WifiManager and obtain wifi data, displayed to screen. Obtain current Location Geopoint from GPS server.	Sushmita	April 1
Evaluate WAMF and Google Latitude	Phil	April 1
Create wifi data container classes and serialize locally/remotely	Phil	April 8
Display name, location, and wifi data via Google Maps overlay	Sushmita	April 8
Phase 1: Store wifi data + geographical data in a local data structure (on phone)	Phil	April 8
Phase 1: Display data as Google Maps overlay	Sushmita	April 8
Phase 2: Post wifi data structure to server	Phil	April 15
Phase 2: Retrieve wifi data structure from server and display locally	Sushmita	April 15
Prepare presentation & final report	Phil+Sushmita	April 21

Future Work

The application in its current form represents a proof-of-concept for the wifi social networking concept. As a result, some of the features are somewhat crude and should be improved in the future.

Much of the code is inflexible and is hardwired for lack of a better solution, given the time constraint of the project schedule. The user must hard-code the IP of the server inside the code, so an interface for specifying a server URL is needed. In addition, the server does not authenticate the Android user and all of the wifi data on the server is public.

Therefore, the server side should be improved. Because this is an Android project, the Googe Apps interface would be an ideal solution. The user could log into their Google user account, which would provide the desired authentication. The server side would also restrict access to those who are acquainted with another user, perhaps by utilizing Gmail contacts and coordinating with the G1's contact list. Such a system would also eliminate the problem of data storage: on our application's server side, all of the wifi history data is simply stored in memory. A more robust solution would save the data to disk or inside an SQL database.

Our application's user interface could be improved by the addition of a menuing system for user options. Along these lines, an interface for manually recording the wifi status could be devised, rather than simply updating data automatically each time the application is opened. Finally, the user will most likely desire additional information about the wifi networks at a particular location. Rather than simply the signal strength of the “best” network, other data could be shown as well: all of the available networks and their respective encryption status. Perhaps Google Maps text bubbles would conveniently show such information. The existing framework of the application can be extended to include these features.

References & Credits

1.http://www.howtoforge.com/installing-google-android-sdk1.0-on-ubuntu8.04 desktophttp://www.google.com/mobile/default/latitude.html

2.http://developer.android.com/reference/packages.html

3.http://moss.csc.ncsu.edu/~mueller/g1/

4.http://blogoscoped.com/archive/2007-11-19-n27.html

5.http://mobiforge.com/developing/story/using-google-maps-android