

CS 456 - Assignment 1 (Spring 2009)

Handheld wardriving

May 5, 2009

1 Goal

Wardriving is the act of searching for wireless networks by a person in a moving vehicle, using a portable computer or PDA.

In this assignment, you will design and implement an application to perform cellular wardriving on a BlackBerry device. That is, you will be searching for and logging cell phone towers in the neighborhood of your device. This will give you a good sense of link-level wireless connectivity. You will also learn how to determine a mobile's geographical location.

2 Assignment

Your application must locate GSM cell towers and record the ID along with the current geographical location and time.

Specifically, the application, once downloaded to the device and started, will wake up periodically, where the period is a programmable parameter and is initially set to 3 minutes. After waking up, the application will retrieve the cell tower ID that the BlackBerry is currently associated with (if any). You may assume that the GSM radio is already enabled. If the BlackBerry is within cellular coverage and can retrieve a cell tower ID, then the application should write a record to a text file on the device with the ID of the cell tower. If not, it should write a record to the same text file indicating that it is out of coverage.

Each record corresponding to a successful determination of a cell tower ID has **eight** fields: the timestamp (seconds), a fixed string indicating that the entry is a 'GSM' entry, latitude (degrees), longitude (degrees), the absolute radio frequency channel number, the base station identity code, the cell tower ID, and the signal strength. Following is an example of a cell tower detected while the user standing outside of the Davis Center at 11 am EST on April 23, 2009.

```
1240498824: GSM, 43.474, -80.542, 131, 13, 35503, -87
```

Each record corresponding to an unsuccessful determination of a cell tower ID has **four** fields: the timestamp (seconds), a fixed string indicating that the entry is an out-of-coverage or 'OOC' entry, latitude (degrees), longitude (degrees). Following is an example of an 'out-of-coverage' event at 11 am EST on April 23, 2009.

```
1240498824: OOC, 43.474, -80.542
```

The timestamp may be obtained using the standard *System.currentTimeMillis()* Java command.

The application should *also* have a user interface that satisfies the following minimum requirements:

- Contains a text field that indicates the *current* cell tower ID or ‘none’ if the device is not associated with any cell tower.
- A text field that indicates the time that last scan occurred.
- A text field that contains a list of the 10 most recent cell tower IDs (skipping over out-of-coverage periods)
- A numeric input field that allows the user to specify the sample interval in seconds.
- An ‘OK’ button or menu item, which will update the sleep interval.
- A ‘Start/Stop’ button or menu item, which enables and disables the application.
- A text field that contains a ten digit number. This field will uniquely identify your trace file after uploading. It may be randomly generated or specified by the user.
- An ‘Upload’ button or menu item, which will transfer the trace file to:

`http://blow.cs.uwaterloo.ca/cgi-bin/cs456_a1_submit.py`

The data **must** be uploaded over the WiFi interface only, so be sure to specify *interface=wifi* when establishing an **HTTP** connection. The GSM interface should be used only to passively collect cell tower IDs. Errors should be reported to the user with a dialog box or error screen.

- After uploading a file, the user should be prompted to keep or delete the trace file.

The `cs456_a1_submit.py` script has been configured to accept the following parameters in the form of an HTTP POST:

```
uid
trace
```

The *uid* is the ten digit number that uniquely identifies your upload and *trace* is the trace data you are uploading. After successfully uploading a trace file, you may view a map of cell tower IDs in your browser by invoking the following URL:

`http://blow.cs.uwaterloo.ca/cgi-bin/wardrive_map.py?view=<ten digit random id>`

2.1 Assumptions

You may make the following assumptions when implementing your assignment:

- The cellular radio is enabled. You therefore do not need to call `Radio.activateWAFs(..)` (a function that requires code signing) since the *WAF_3GPP* is assumed to be enabled.
- The */SDCard/* file system root is always available. Therefore creating a new `FileConnection` to a file on the SDCard will always succeed.
- The default permissions on the BlackBerry are all set to ‘Allow’. You therefore do not need to prompt the user for permissions to perform certain tasks. You may set the default permissions on installed applications by going to *Options* -> *Advanced Options* -> *Applications* then select *Edit Default Permissions* from the menu and set everything to ‘Allow’.

2.2 Bonus

The application may optionally implement WiFi Wardriving by recording the WiFi access points that the BlackBerry associates with throughout the day. The application must periodically enable the WiFi radio (if it is off), and attempt to connect to a neighbouring WiFi AP. If the BlackBerry associates with an AP within 60 seconds, then a record should be written to the same file used to record cell tower IDs. If the WiFi radio was originally off, then the application should disable it after recording an AP or the 60 seconds period ends.

Each WiFi record has **seven** fields: a fixed string indicating that the entry is a 'WiFi' entry, the timestamp (seconds), latitude (degrees), longitude (degrees), the WiFi BSSID, the data rate, and the signal strength. Following is an example of a WiFi AP detected while the user standing outside of the Davis Center at 11 am EST on April 23, 2009.

```
1240498824: WiFi, 43.474, -80.542, 00:23:12:55:57:3a, 11, -91
```

3 What to turn in

In addition to uploading the traces to the assignment server, you also should turn in:

- Complete source code, project file, workspace file, and resources using the unix *submit* utility available in the undergraduate computing environment.
- A sample trace (*trace.txt*) output in text format using the *submit* utility.
- A paper document that describes your application, specifies the ten digit random number used to submit your trace, indicates what you have completed (or not completed), problems encountered, and any other comments that you wish to make.

Please limit the document to **three** pages, with 1 inch margins, and no less than 12pt font. The document may also include screenshots of your application captured using the screen recording tool pre-installed on each BlackBerry.

The unix *submit* utility has been configured to accept:

```
*.java  
*.png  
*.jdp  
*.jdw  
trace.txt
```

Therefore all files must be contained within a **single directory**. To submit all of the files in the current directory, use the command: *submit cs456 a1 .*

4 Grading scheme

This assignment will be evaluated as follows:

Basic functionality	Application runs and writes data to file in the correct format.	40%
Communication	Data is successfully uploaded and errors correctly reported.	40%
Wardriving	Trace file contains at least 5 unique cell tower IDs over at least 4 hours.	20%
Bonus	A maximum of 15% will be added for successfully completing of the bonus component.	15%

5 Question, comments, or problems

Please post general questions or comments to the 'A1' discussion group on UW ACE.

Remember: You must use the WiFi interface on the BlackBerry to transmit data. The GSM interface should be used only to passively collect cell tower data.