AndroDMTP

An Android implementation of the OpenDMTP protocol

Codella Tommaso
Matr. 739376, (tommaso.codella@gmail.com)
Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milan, IT

Report for the master course of Embedded Systems
Reviser: PhD. Patrick Bellasi (bellasi@elet.polimi.it)

Received: september, 2011

Abstract

In this report will be introduced AndroDMTP an implementation of the OpenDMTP protocol for Android.
OpenDMTP is a protocol and a framework designed for mobile devices that allows bi-directional communication between server and devices over network.
OpenDMTP was developed in J2ME and C programming language; the second one implementation could be yet useful in small embedded devices but the J2ME implementation could seem dated because Java Micro Edition is no longer used by newest mobile devices such as iOS, Android or WP7 devices.
The goal of this project was to re-implement the necessary classes of the J2ME implementation of OpenDMTP in order to realize an Android Service that provides basic GPS monitoring service over DMTP.

1 Overview

In this report will be introduced the main goals behind the developing of the Android implementation of OpenDMTP.
In section 2 will be analyzed the OpenDMTP's main features and its operation.
Will be focus on the features that make this protocol perfect for small mobile devices.
In the third section will be analyzed the Android localization system and its differences with the GPS support of Java Micro Edition (J2ME), also will be shown the simplifications introduced by the Location Manager of the Android OS.
In section 4 AndroDMTP will be explained; first will be analyzed the structure of the J2ME version of DMTP and then the structure of AndroDMTP.
The last section is dedicated to the final evaluations on the project and will be explained some possible future developments.

2 OpenDMTP Protocol

The "Open Device Monitoring and Tracking Protocol", also known as OpenDMTP, is a lightweight protocol and framework created to allow a bi-directional communication between a server and some devices.
It is particularly geared to the distribution of Location Based Information such as GPS data or also temperature or other data that could be collected from a remote-monitoring devices.
This protocol is designed to fit to the limited resources of the small devices on which it can run it also has a small footprint in terms of bandwith consumption so it can be used on devices with a limited band per month.
Another fact that made OpenDMTP a perfect choice for mobile devices is that all precaution taken to fit it to mobile devices makes it a low energy consumption protocol.
The only way to understand how this protocol is so well designed for embedded and mobile devices is to analyze its main features and key points so, in the next few section OpenDMTP will be introduced focalizing attention on the points that has made it different, and in some cases best, from other standard protocol.
During this overview on the OpenDMTP features the reader has to keep in mind that openDMTP was ideated about five years ago by GEOTelematic Solution, Inc. and at that times the mobile devices world was very different from the actual : the iOS and the Android OS will born only a year or two later and the pdas were very limited in terms of hardware and bandwith. The limitation of band was not only a speed limit of the GSM/CDMA module but also a limitation in terms of data send because mobile operators, at that time, did very expensive plans that allow you to send only small amounts of data.
Nowadays with the concept of mobile internet and connection everywhere, mobile devices had a more fast connection (in some cases faster than the home ones) with data plans that permit to send very large amount of information with a small cost.
In this optic the features that will be illustrated in the next sections sometimes could seems not a real need of new mobile devices but remember that these features remain interesting for embedded environment and for special pur-

pose applications like automotive sector.

2.1 Target event generation and network efficiency

OpenDMTP protocol is designed to save bandwidth and to do this it not send copious amount of useless data.
This protocol can be set up to send GPS data only when is necessary e.g., when the device is in motion or when the device has travelled a minimum distance.
In this manner the bandwidth is used to transmit only useful data and, for example, if the device is stopped the protocol does not send any data to the server.
For the reasons above described OpenDMTP is optimized to save bandwidth during the transmission even if is a bi-directional transmission.
To obtain a small footprint another solution taken is to use ASCII packet instead of XML packet: XML is extensible but it add an overhead to sent information that not allow to obtain network efficiency in addition, some really small mobile devices not have the resources to elaborate XML data.

2.2 Small footprint

Mobile devices could have limited resources so the protocol is optimized to have minimum impact on the device.
In addition to this it is designed to allow to send configurable messages not only in the content of the message but also in the encoding; the reason of this is that sometimes some type of data can be sent better if encoded in ASCII format instead of binary format.

2.3 Extensible

This protocol, as will see in the usage scenario section, could be used in very different application so it is important to have a very flexible protocol in order to adapt it to the specific application and device on which it have to run.

3 Android Localization System

Android has introduced localization inside the OS so it is simple to interact with localization devices such as GPS or GSM.
This innovation has simplified the implementation of localization oriented application.
In the past with J2ME it was impossible to interact directly with the GPS and it was necessary to have java driver to interact with these devices.
For this reason in the J2ME implementation of OpenDMTP there was some classes that was designed to interact with specific GPS antennas.
This feature does not allow to compile a single version of the protocol that could work on every device so if you want to run OpenDMTP on a not supported device or antenna you had to implement a new Java class that interacts with the desired equipment.
In order to simplify application development and to make applications available for any device independently from GPS antenna inside it Android OS contains localization support inside the system.
In this way the programmer can ignore what type of GPS or other localization systems the user has because is the operating system that collects data from the GPS and sends it to the application.
For this reason during the developing of AndroDMTP all classes used to manage the GPS antenna were deleted because in Android only the OS can manage the GPS.

3.1 Using LocationManager

In order to obtain user's location, programmer have to invoke the LocationManager.
LocationManager is the class responsible to manage the location updates send from localization sensors.
In an Android device, the tracking sensors could be: GPS, Wi-Fi or GSM.
To respond to location updates LocationManager needs a LocationListener in which the programmer could define the actions to take when an update arrives.
As said before Android allows to get location from more provider for this reason when LocationManager is instantiated it is possible to define from which provider the manager can accept updates and in the listener is possible to recognize from which sensor the update came so the application could react in various way depending from the provider that has generated the update.
In the Android Developer's Guide there are all information that a programmer needs if he or she wants to develop an application that uses user's location.
The only really interesting thing that could be useful to understand the LocationManager behavior is the flow for obtaining the user location.

- Start application.
- Sometime later, start listening for updates from desired location providers.
- Maintain a "current best estimate" of location by filtering out new, but less accurate fixes.
- Stop listening for location updates.
- Take advantage of the last best location estimate.

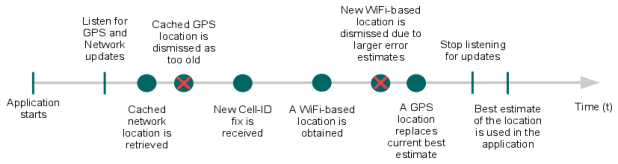

Figure 1: Flow for obtaining user location.

In this flow you can see that the third point is to maintain the "current best location"; In the Android Developer's Guide there is an implementation of an algorithm that evaluate the "current best location" every time a location update arrives.
For this reason all methods in OpendDMTP for Java that evaluate if the new location is better than the previous were deleted because this check in AndroDMTP will be done in the LocationListener every time a new location is detected.

4 AndroDMTP

In this section will be shown how AndroDMTP was builded.
In a first moment AndroDMTP was ideated as a simple wrapper of the J2ME version of OpenDMTP but after analyzing its source code seemed very difficult and useless because most of the code that implements abstract classes or the interfaces was developed for specific devices such as motorola treo.
For these reasons was decided that the best approach was to identify which classes can be maintained.
In this optic was decided to remove all classes that implements code for specific devices or classes that used J2ME specific code.

4.1 Features

AndroDMTP is designed as an Android Service invokable from external application using bindings.
This first version of DMTP protocol for Android allows only simple GPS tracking and communication with a DMTP server.
The code is prepared to connect other modules such temperature detection module, odometric module and others.
AndroDMTP is bundled with a configuration application that allows to configure standard parameters of the protocol.
In addition to this with the configuration application is possible to test AndroDMTP because the application allows to configure a connection to a DMTP server and to record GPS tracking data with duplex communication with the server.

4.2 Usage Scenario

The usage scenario of AndroDMTP could be very interesting because it can be used in all application in where DMTP was used such as: truck tracking or equipment's monitoring.
In this release can be only done motion tracking of truck, people or everything which can move.
This function is the same provided by OpenDMTP for Java but in difference AndroDMTP have not additional modules implemented but is fully prepared to interact with them.
As said this implementation of the protocol can be used not only to monitoring trucks or vehicles but also people this is possible thanks to the fact that Android is commonly used as smartphone OS.
An interesting use of this protocol is to implement a people tracking system that allows to create an analysis of the people flow in a city.
AndroDMTP could be used to get information about pedestrian traffic in a city or better in a touristic city.
Take as example the city of Verona, a touristic city that attracts almost three millions tourists every year.
The City of Verona could implements an application that track the position of the mobile phone of the user and send the data to a server using DMTP; the server could process the data of every users and provide to every client pedestrian traffic situation in the downtown.
Also the application could analize queues at the entrance of major attractions so the tourist could decide which monument to visit based on the information provided by the flow of tourists in the city.
The interaction between client's device and the server could be done using DMTP because this protocol provides a small footprint in terms of bandwidth used. This feature is very interesting for a tourist: nowadays mobile internet tariffs are expensive abroad and sometimes are related to the amounts of data downloaded or uploaded so the usage of a protocol that grants low data interaction with the server is the best way to realize an application like that.
Other applicative scenarios could be possible but the one presented should be an interesting example of using of this protocol.
This type of application could work only if a large amounts of people install the application on their devices, for this reason and for others that will be discussed in the last section it could be interesting an implementation of DMTP for iOS.

4.3 The Configuration Application

AndroDMTP Configuration Application is the application bundled with AndroDMTP Service that permits to the user to configure basic settings of DMTP protocol is organized in three tabs: Status, GPS, Server.

Status Tab The status tab is divided in two main sections: in the first one is possible to start, pause or stop the DMTP's main loop.
The second section displays gps current data, this section is active only if the protocol is started otherwise all labels displays: "no data".

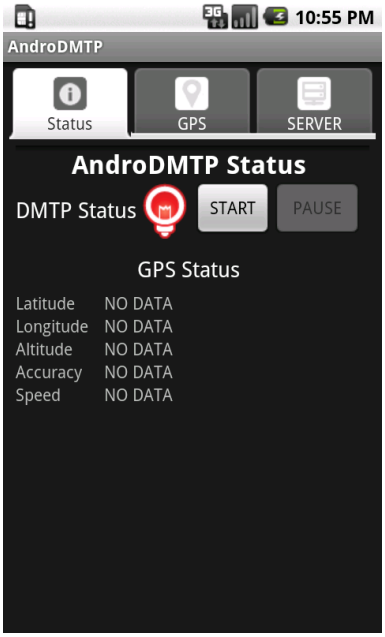

Figure 2: Status tab screenshot.

GPS Tab In this tab user can sets how AndroDMTP have to treat GPS data and the conditions to send or not data to the server.
In the following list will be shown which parameters could be set and what they means.

- GPS rate: indicates the time that must elapse between one detection and the other before considering the new data as such
- GPS accuracy: indicates the minimum accuracy accepted by the protocol to consider the received data as valid
- GPS min speed: indicates the minimum speed that the device must have in order to send received data to the server. AndroDMT, in order to consume less bandwidth, can send data to the server only if the device is in motion so it not send multiple useless data when the user is stopped
- GPS motion start type: this parameter can be set to motion speed or motion position and indicates which method the protocol can use to understand if the device is in motion or not. If set to motion speed, the device is in motion if the device's speed is more than the speed defined in "GPS motion start speed" field otherwise, the device is in motion if it has travelled a distance greater that the one indicated by the field: "GPS motion start meter".
- GPS motion in motion, GPS motion stop and GPS motion dormant: these fields are actually not used by this implementation of AndroDMTP but they could be useful in a future when motion module will be available.

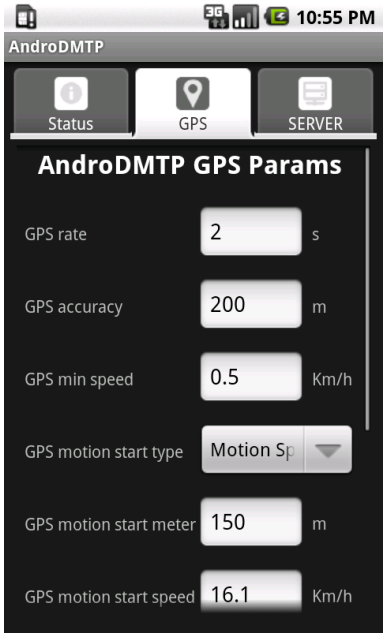

Figure 3: GPS Tab screenshot.

Server Tab In the server tab the user can set all information needed to connect to the DMTP server. These information are: server address (IP address or DNS name are both accepted), server port on which DMTP server is listening for incoming connections, account name and user device in order to identify which user has sent data.

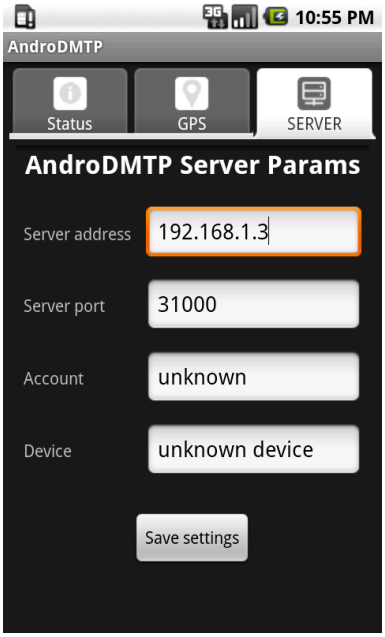

Figure 4: Server tab screenshot.

4.4 Source Code Organization

The source code of this project is organized in two main packages: one for the classes recycled from OpenDMTP and the other for the new classes created specially for Android.

OpenDMTP Classes All OpenDMTP source codes are inside the package *com.tommasocodella.androdmtp.opendmtp*. In this package is possible to find all the classes from OpenDMTP that are used in AndroDMTP.
The only classes modified are: **Protocol** in *client.base* package and **CThread** in *util* package.
These classes were modified in order to implements some functionalities: start stop and pause and to allow the use of DMTP as an Android Service.

AndroDMTP Classes All AndroDMTP classes are contained in three packages: *configurationapp*, *gps*, *services*.

configurationapp Package contains all classes that implements the configuration application described above.

- **AndroDMTPStatus** is the class that implements the Status tab.
- **GPSSettings** implements the GPS tab.
- **ServerSettings** is the class that implements Server tab.
- **ConfigurationApp** is the container class that manages the three tabs
- **CommunicationDispatcher** is the class responsible of the communication between tabs and AndroDMTP services

gps Package contains all classes that manage the interaction between AndroDMTP and Android's Location Manager.

- **AndroDMTPLocationListener** is the class responsible to listen updates from GPS and evaluate if the new data are valid or not.
- **AndroDMTPStatusLocationListener** is responsible to communicate actual valid data to the Status tab in order to display them.
- **GPSUtils** is a class with general methods that are useful in the GPS management.

services Package contains all classes that implement AndroDMTP services.

- **AndroDMTP** is the class that implements the Main Loop of DMTP protocol. In this class the protocol is initialized and the GPS listening loop is started.
- **AndroDMTPMainService** is the Android Service that manage AndroDMTP Main Loop.
- **PersistentStorage** is the class that implements a simple persistent storage for protocol parameters. In this release is not completely implemented.
- **TransportImpl** is the implementation of the DMTP transport protocol and provides all methods to send and receive data from server.

5 Conclusion and future works

In this paper was introduced AndroDMTP project: an Android application that implements the OpenDMTP protocol.
As it shown this application is only a first step in a more long path that could lead to have a more complete application with more features and well optimized for Android devices. The problem of this application is that was builded starting from OpenDMTP for Java Micro Edition.
OpendDMTP for J2ME was builded for devices with limited features and with a version of Java that does not implements many standard Java features that are now present in Android.
This implementation for Android is probably the first major update since 2006 for OpenDMTP so it's normal that, as a first release for a new platform, it could be improved.
The first improvement could be the re-writing of the classes recycled from OpenDMTP in order to have a code optimized for Android that uses latest OpenDMTP specifications.

After that the external module support could be improved and the internal storage system has to be re-designed in order to manage all parameters of the OpenDMTP protocol. A parallel project could be the implementation of a version of OpenDMTP for iPhone and other iOS devices so it could possible to build applications for end users that uses this protocol.

References

[1] Martin D. Flyn; *OpenDMTP Protocol Definition Reference Manual*. 2006

[2] Google Inc.; *Android Developer's Guide (http://developer.android.com/guide)*