Design of an Android Real-Time Bus Location Provider

Gyeyoung Lee, Jaegeol Yim

Department of Computer Engineering, Dongguk University, Gyeongju, Gyeongbuk, 780-714, Korea
{lky, yim}@dongguk.ac.kr

Abstract: An intelligent transportation system (ITS) is an application that provides dynamic traffic information to users so that the users can make safer and smarter use of transport networks. Examples of information type provided by an ITS include: bus arrival information, route information, bus service information, allocation information, route arrival information, station arrival information, front and rear car interval, transit time, and so on. In order to identify vehicles and detect locations and speeds of vehicles, we have to install a lot of special equipments in an ITS. For example, road side equipments (RSE), automatic vehicle identification (AVI) systems, automatic vehicle locations (AVL) systems, trunked radio systems (TRS) are needed to be installed in an ITS. Consequently, the cost to implement and maintain an ITS is enormous. This paper introduces a design of an Android App system that provides bus location information. The structure of our system contains three components: service request client, location up-loader, and a server. Design detail of the system is discussed in this paper.
[Hassanein M. K., M. A. A. Abdrabbo, A. A. Farag, S.M. Abolmaaty and A. A. Khalil. **Application of Geographic Information Systems to produce descriptive maps for Poultry Farms in Egypt.** *Life Sci J* 2014;11(7):619-625]. (ISSN:1097-8135). http://www.lifesciencesite.com. 86

Keywords: Bus arrival time; Smartphone; App; Intelligent Transportation System

1. Introduction

We have proposed a collaborative bus arrival time estimation method in our earlier works (Yim 2014). Bus information system (BIS) provides real-time bus service information for bus users to save their valuable time. The bus arrival time service is a key service to improve public transport attractiveness by providing users with real-time bus arrival information which can help them to arrange their bus travel schedule intelligently (Zhu, Dong, Huang, Pang and Du 2012).

Existing practical BISs use special equipments installed on buses and/or bus stops. For example, global positioning system-based automatic vehicle location (AVL) systems have been adopted by many transit agencies for tracking their vehicles and predicting travel time in real time (Gong, Liu and Zhang 2013).

We introduced our design of BIS that does not require any special equipment installed on buses or bus stops. Our system is a client and server system. There are two kinds of clients: GPS up-loaders and bus arrival time requesters. An up-loader is a smartphone app running on a bus passenger's smartphone. Up-loaders continuously read GPS value and the current time and send them to the server. With up-loaded GPS values, the server determines which bus is running on which bus route and where it is. A bus arrival time requester is a smartphone app and is supposed to be run by a user who is waiting for a bus. This paper discusses design detail of the system.

2. Related Works

The main idea of our system is based on the following facts: 1) Almost everybody has a smartphone. 2) A smartphone is equipped with very accurate GPS receiver. 3) Human being is a social animal. Therefore, bus passengers can collect their location information with their smartphones and would be willing to upload their location information to the server in order to help others who are waiting for buses (Yim 2014).

Collecting bus location information with GPS devices is not new. Padmanaban, et al collected the travel time data using GPS units carried by observers travelling in buses (Padmanaban, Divakar, Vanajakshi and Subramanian 2010). However, they do not collect GPS values from passengers in real-time.

The bus arrival time is primary information to most city transport travelers. Excessively long waiting time at bus stops often discourages the travelers and makes them reluctant to take buses. Zhou et al presented a bus arrival time prediction system based on bus passengers' participatory sensing (Zhou, Zheng and Li 2013). The basic idea of our system design is very similar to theirs. However, there is big difference in detail. For example, the up-loaders of their system use commodity mobile phones as well as various build-in sensors to sense and report the lightweight cellular signals and the surrounding environment to a backend server whereas our up-loaders use their ordinary smartphones. As another example, their up-loaders upload regularly such as every one second whereas our up-loaders detect bus stop status and only upload when the bus stops.

Real-time bus location information is one of the services provided by an Intelligent Transportation system (ITS). Most of the ITSs require on-board devices and radio on vehicles especially buses, pavement sensors, CCTVs, AVI (automatic vehicle identification)/AVL (automatic vehicle location) readers, and so on.

As an example, let us consider Daejeon city ITS [5]. The structure of the system is shown in Figure 1. In the figure, TRS stands for trunked radio system. A TRS is a complex type of computer-controlled two-way radio system that allows sharing of relatively few radio frequency channels among a large group of users. Instead of assigning, for example, a radio channel to one particular organization at a time, users are instead assigned to a logical grouping, a "talkgroup". When any user in that group wishes to converse with another user in the talkgroup, a vacant radio channel is found automatically by the system and the conversation takes place on that channel [6].

RSE in the figure stands for road side equipment (RSE). An RSE contains an automatic vehicle identification (AVI) and an automatic vehicle location (AVL) system. For example, an RSE can be based on an RFID (radio-frequency identification) reader. RFID is the wireless non-contact use of radio-frequency electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects. Some tags are powered by and read at short ranges (a few meters) via magnetic fields (electromagnetic induction), and then act as a passive transponder to emit microwaves or UHF (ultra high frequency) radio waves. Others use a local power source such as a battery, and may operate at hundreds of meters. Unlike a barcode, the tag does not necessarily need to be within line of sight of the reader, and may be embedded in the tracked object [7].

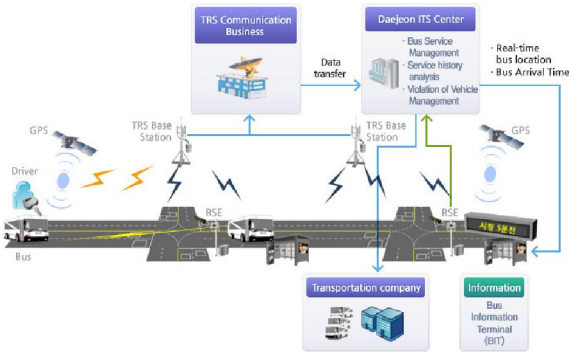

Figure 1. Structure of Daejeon BIS

In the figure, every bus is attached with an RFID tag. An RFID reader in an RSE reads RFID tags and identifies the bus that arrives the spot where the RSE is located. RSEs send collected data to a TRS base station in real-time. TRS base stations, in turn, transmit collected data to Daejeon ITS center.

Using the data, ITS center provides the following information to citizens: bus arrival information, route information, bus service information, allocation information, route arrival information, station arrival information, front and rear car interval, transit time, and so on [5].

3. Design of Our BIS

The main components of our BIS are Up-loader, requester and the server as shown in Figure 1. An up-loader is a smartphone app that reads GPS values regularly, detects if the bus stops, and sends the current time and GPS value including latitude and longitude to the server if the bus stops. An up-loader is supposed to be run on a smartphone of a bus passenger.

The server receives GPS values from up-loaders and saves them in PassengersTable in the database. The attributes on PassengersTable include UserID, Time, Latitude, and Longitude. There is BusRoutesTable in the database. The attributes on BusRoutesTable include BusRouteID, StartStop, EndStop, and BusCompany. There is BusStopsTable in the database. The attributes on BusStopsTable include BusStopID, BusRouteID, Latitude, and Longitude. With the sequence of GPS values from an up-loader, the server can recognize the bus route on which the bus, in which the passenger is riding, is running. We call this bus route a *bus running route*. Note, we have one *bus running route* per up-loader.

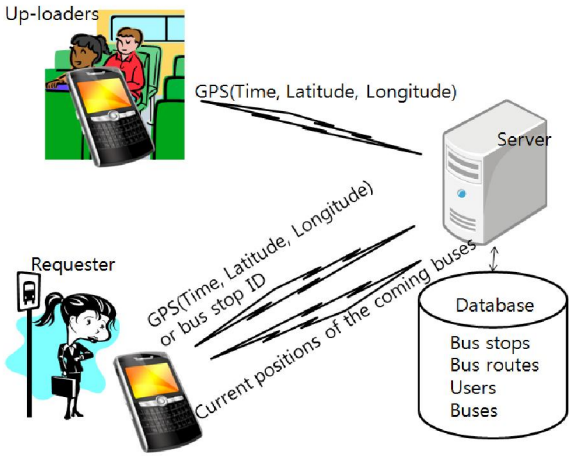

Figure 2. Main components of our BIS [8]

A requester is a smartphone app and is supposed to be run by a user who is waiting for a bus. The user can be at a bus stop or at some other places. In the former case, the requester reads the current GPS value and send them to the server. In the latter case, the requester sends the bus stop ID (or name) to the

server. Then, the server retrieves all the *bus routes* that contain this bus stop. We call these bus routes "*related bus routes*". The server returns the current positions of the passengers in the buses running on these "*related bus routes*".

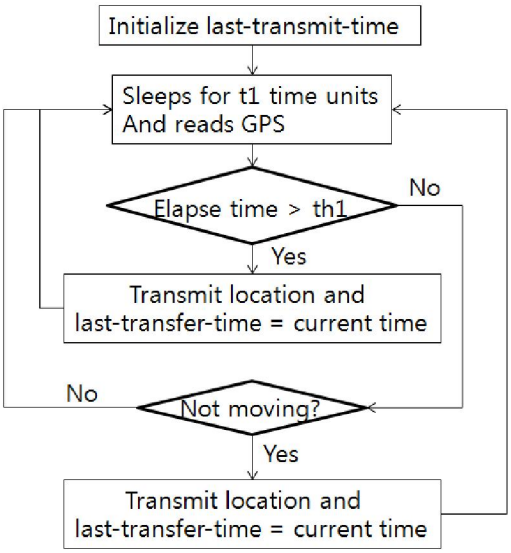

Figure 3. The process of our up-loader

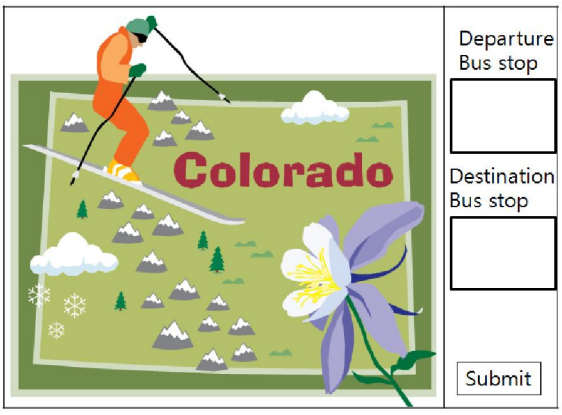

Figure 4. The layout of our requester

3.1. Design of the Up-loader

Our up-loader regularly reads GPS and transmits location value read from GPS to the server every minute or when the bus stops. To do this, our up-loader initializes last-transmit-time with the current time of the day. Our up-loader reads GPS regularly, every 500 ms for example. Therefore, our up-loader sleeps for t1 (500 for example) time units. Then, it reads GPS. If the elapsed time from the last-transmit-time is greater than a threshold (th1, one minute, for example), then it transmits location value from GPS to the server and updates last-transmit-time with the current time of the day. It repeats the process of transmitting location value from the GPS every one minute. In addition to transmitting regularly, it also transmits location value whenever the bus is not moving. This process of our up-loader is shown in Figure 3.

3.2. Design of our Requester

A scheme of the user interface of our requester is shown in Figure 4. The user interface contains two windows, one for a map and the other for user input. A user can type in the name of the departure bus stop and the destination bus stop. Then a map of the area around the departure bus stop is displayed on the map window. Then, it shows the bus routes that the user should take on the map.

3.3. Design of our Server

Our server receives location information from up-loaders and save it in the database. It also finds out the bus route on which the user of the up-loader is taking. The server also receives a message from a requester and informs the requester of the locations of the buses that the user wants to take. The processes the server performs are described in Figure 5.

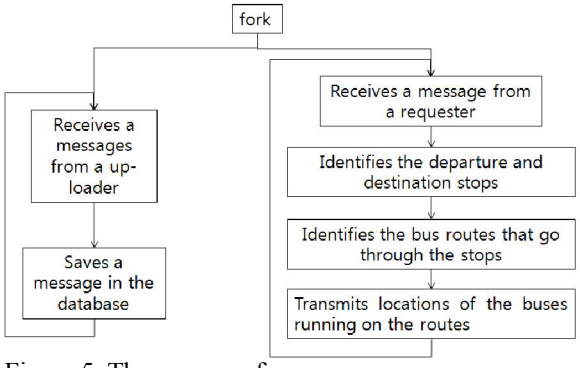

Figure 5. The process of our server

Therefore, the classes listed in Table 1 are needed to be defined in the server. DBConnector returns a database connection.

Table 1. A list of classes consisting the server

Class Name	Description
DBConnector	For database connection
BusInfoServiceDao	For data insertion to and retrieval from the database
FindBusRoute	To find the bus route the user is taking
SaveStopState	This is a web service that saves bus location information when the bus stops by invoking BusInfoServiceDao
SearchCurrentUserBusLine	This is a web service that executes Calculate
BusStop	A template class for bus stop
User	A template class for user information
Distance	calculates the distance between 2 point
Util	Finds the intersection set of two ArrayList

The main role of FindBusRoute class is to find the bus route a up-loader is taking with a sequence of locations the up-loader up-loaded. The process of identifying the route is described in Figure 6.

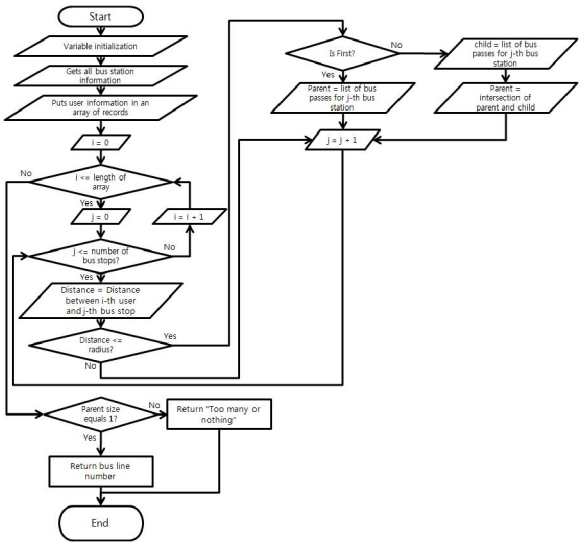

Figure 6. The process of identifying the bus route on which the up-loader is running

BusInfoServiceDao provides methods that inserts data (DeviceID, Latitude, Longitude) into the database and retrieves data from the database as shown in Table 2.

Table 2. Description of methods in BusInfoServiceDao

name	Input	Output	Description
saveState	DeviceID, Latitude, Longitude	void	Save location of a user and time in the database.
getUserInfomation	DeviceID	All data saved by the user indicated by DeviceID	List of all data saved by the user indicated by DeviceID.
getBusLineName	BusLineID	BusLineName	Returns the bus route name identified by BusLineID
getBusStationLine	BusStationID	BusLineIDs of the bus lines that stops at the bus stop indicted by BusStationID	A list of BusLineIDs of the bus lines that stops at the bus stop indicted by BusStationID
getAllBusStation	void	List of all bus stops	Returns a list of all bus stops

The process of the saveState method is shown in figure 7. It makes use of DBconnector in order to access the database, construct an insert sql sentence, and execute the sentence. After executing, it returns success or failure message to the caller in order to inform it of the execution result.

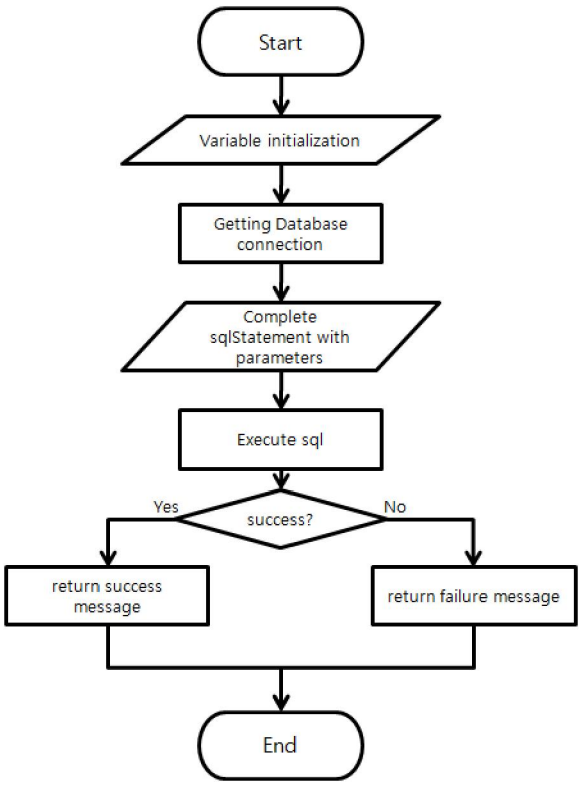

Figure 7. The process of saveState method

BusStop class and User class are template classes. Functions defined in BusStop class are decribed in Table 3.

Table 3. A list of methods defined in BusStop

name	Input	Output	Description
getBusStopID	void	BusStopID	returns BusStopID in the class.
setBusStopID	BusStopID	void	set BusStopID with the parameter
getLatitude	void	Latitude	returns Latitude in the class
setLatitude	Latitude	void	set Latitude with the parameter
getLongitude	void	Longitude	returns Longitude of the class
setLongitude	Longitude	void	set Longitude with the parameter
getBusStopName	void	BusStopName	returns BusStopName in the class
setBusStopName	BusStopName	void	set BusStopName with the parameter

SaveStopState is a web service that takes DeviceID, Latitude and Longitude parameters and saves them in the database. The process of this web service is shown in Figure 8. This service makes use of saveState method in BusInfoServiceDao in order to access the database. If the insertion succeeds/fails then it returns a success/failure message.

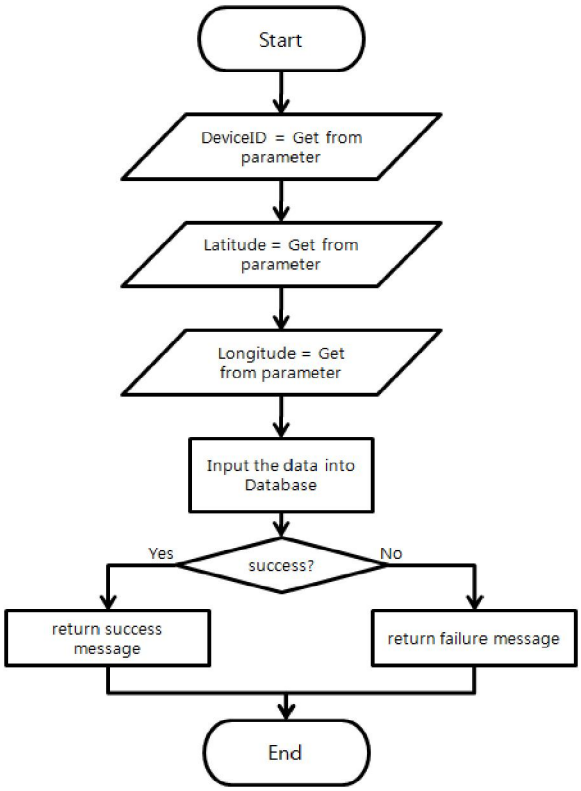

Figure 8. The process of SaveStopState

4. Implementation of Our BIS

Nowadays, most of the smartphones are equipped with very accurate GPS receivers and sensors. They also provide developers with efficient APIs. For example, Android app developers can easily read GPS/Sensor values with LocationManager/SensorManager. Our up-loader regularly collects accelerometer values and it assumes that the bus is stopped when the variance of collected accelerometer values is less than a certain threshold.

One of the most important parts of the user interface of our system is the map. Once we obtained an API_KEY, we can use Daum map as shown in Table 4 in order to display the standard map.

Table 4. A part of our implementation of mapView

```
MapView mapview;
String API_KEY = "282ec... 65";
LinearLayout linearview = (LinearLayout) findViewById(R.id.mapview);
mapview = new MapView(this);
mapview.setDaumMapApiKey(API_KEY);
mapview.setMapType(MapView.MapType.Standard);
linearview.addView(mapview);
```

Table 5. Codes to move the center of the map

```
MapPoint startpoint = MapPoint.mapPointWithGeoCoord(...);
mapview.setMapCenterPoint(startpoint, false);
```

Given a pair of (latitude, longitude) from an up-loader, we have to find the nearest bus stop. A part of the procedure to find the bus stop is shown in Table 5.

Table 6. A part of our implementation to find the bus stop nearest to the given location

```
public void calculateArrayDistance(double latitude,double longitude){
  Distance distance = new Distance();
  ...[] arrayBusStops = mapView.getBusStops();
...
  for(int i=0;i< arrayBusStops.length;i++){
    if(arrayBusStops [i].getTag() == 0) continue;
    String temp=null;
    double P2_late = arrayBusStops [i].get... latitude;
    double P2_lon = arrayBusStops [i].get... longitude;
    double dis = distance.distance (latitude, longitude, P2_lat , P2_lon);
    temp = "Distance to" arrayBusStops[i]....() +"is"+(int)dis +"m ";
...
}
```

5. Experiments

We have tested our bus stopped status recognition program. Our test results showed that our program says "The bus is stopped" not only when the bus stops but also the bus slows down at a corner.

Table 7. Example collected GPS values

	GPS	Google Map
Stop1	35.86409402133178 129.200397409416	35.86703 129.201458
Stop2	35.862011420672324 129.19768552134303	35.860376 129.196048
Stop3	35.85794626379323 129.19756174836533	35.857677 129.197748

Figure 9 The standard map

We have tested accuracy of GPS receivers. Example test results are shown in Table 6. Considering the distance between latitude 35 degree and 36 degree is longer than 100 Km, we can imagine how big the difference between GPS value and the coordinates obtained from Google map is. Even so, our procedure to find the nearest bus stop finds the bus stop correctly because the distance between adjacent two bus stops is much longer than the GPS error.

One of the most important parts of the user interface of the client is the map. A result of executing the code in Table 4 is shown in Figure 9 and a result of Table 5 is shown in Figure 10.

Figure 10 After moving the center of the map

6. Conclusions

BIS is extremely costly because it requires special equipments installed on buses and/or bus stops. We proposed a method that relies on smartphones which are equipped with GPS receivers. Since most smartphones are equipped with GPS receivers and almost everybody carries a smartphone, our method is very economical. We also proposed a structure of bus information system (BIS) that utilizes our proposed method. Furthermore, we have shown our test results that show the proposed BIS is practical.

We discussed our design of the system in this paper. We now have to refine our procedure of detecting stopped status of a bus so that it can discriminate slowing down status of a bus.

We also have to improve accuracy of GPS values obtained from smartphones. To this end, we are planning to utilize the Kalman filter process and map matching method. After improving the accuracy of GPS values, we are planning to implement a practical BIS prototype.

Acknowledgements:

This research was supported by Basic Science Research Program through the National

Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2011-0006942) and by 'Development of Global Culture and Tourism IPTV Broadcasting Station' Project through the Industrial Infrastructure Program for Fundamental Technologies funded by the Ministry of Knowledge Economy (10037393).

Corresponding Author:
Ph.D. Yim, Jaegeol
Department of Computer Engineering
Gyeongju Campus, Dongguk Univ.
Gyeongju, Gyeongbuk, 780714, Korea
E-mail: yim@dongguk.ac.kr

References

1. Zhu T, Dong J, Huang J, Pang S, Du B. The bus arrival time service based on dynamic traffic information. 6th International Conference on Application of Information and Communication Technologies 2012; 1-6.
2. Gong J, Liu M, Zhang S. Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time GPS data. 25th Chinese Control and Decision Conference 2013; 972-976.
3. Padmanaban R, Divakar K, Vanajaksho L, Subramanian S. Development of a real-time bus arrival prediction system for Indian traffic conditions. Intelligent Transport Systems 2010; 4(3): 189-200.
4. Zhou P, Zheng Y, Li, M. How long to wait? Predicting bus arrival time with mobile phone based participatory sensing. IEEE Transactions on Mobile Computing 2013; PP(99): 1-14
5. http://traffic.daejeon.go.kr/eng/introduction/intelligentTransportSystem.do
6. http://en.wikipedia.org/wiki/Trunked_radio_system
7. http://en.wikipedia.org/wiki/Radio-frequency_identification
8. Yim, J., "Proposing a Collaborative Bus Arrival Time Estimation Method," The 2014 FTRA International Conference on Advanced Computing and Services (ACS-14) Proceedings, Jeju Island, Korea, 2014; 66-68

9/6/2012