International Journal of Control and Automation
Vol.8, No.11 (2015), pp.93-104
http://dx.doi.org/10.14257/ijca.2015.8.11.10

Implementation of a Bus Travel Path Recommendation System

Won Joo Lee[1], Jaegeol Yim[2*] and Youngjoon Kim[2]

[1]*Dept. of Computer Science, Inha Technical College, Incheon, Korea*
[2]*Dept. of Computer Engineering, Dongguk University at Gyeongju, Gyeongbuk, Korea*
[*]*yim@dongguk.ac.kr*

Abstract

Buses are the only public transportation available to citizens in cities that do not belong to a metropolitan area. The interval between two consecutive bus departure times is pretty long in these cities. Therefore, a bus information system (BIS) that provides a variety of useful information to users is desperately needed in these cities. One type of useful information that BISs provide is how to get from the starting bus stop to the destination bus stop. In this paper, we analyze one of the existing algorithms that finds efficient bus travel paths and point out its shortcomings. Then, we design and implement an algorithm that mitigates the shortcomings. Implementation details of the algorithm are also discussed.

Keywords: *Smartphone, Shortest Path, Bus Information System, Intelligent Transportation System, Sensors*

1. Introduction

A bus information system (BIS) provides a variety of useful information, such as an efficient bus path that brings the user from the starting bus stop to the destination bus stop, the current positions of buses that are approaching the designated bus stop, points of interest around the designated bus stop, and so on.

Among the information types that BISs provide, an efficient bus path that gets the user to the place the user wants to go is the most valuable information to a traveler who is visiting a strange town. In this paper, we survey algorithms that find efficient bus paths, and design a new algorithm that mitigates the shortcomings of the existing algorithms.

Then, we developed a bus travel path recommendation system implementing the algorithm on real bus data for a small city. The recommendation system is a client/server system. The client is a mobile app, whereas the server is a personal desktop computer. The client accepts a starting bus stop and a destination bus stop as user input. If the inputs are valid bus stops, then the app sends them to the server. The server receives the user input, creates a distance matrix, finds the shortest path, finds the bus routes coinciding with the shortest path, adjusts the bus routes, and sends the adjusted bus routes to the client. The client receives the bus routes and displays them on the user interface.

2. Related Works

The bus information system has been a hot research topic for a lot of researchers. Cheng. [1] conducted a survey to gauge levels of satisfaction in bus passengers within five months of a dynamic bus information system's completion and initial assessment tests. Megalingam [2] Proposed a system that could track the current position of buses, the dynamic arrival and departure times, and inform passengers via display boards at the

* Corresponding author

ISSN: 2005-4297 IJCA

terminus or through an app installed on a smartphone. The proposed system consisted of a transmitter module, a receiver module, and a smartphone. The transmitter module is fitted on the bus, and the receiver module at the bus terminus.

Zhou [3] proposed implementing a crowd-participation bus arrival time prediction system utilizing cellular signals. Independent of any bus company, the system bridges the gap between users querying about the bus's arrival time and users willing to share information, offering them real-time bus information. A querying user sends the bus stop and route of interest to the server. A sharing user sends a cell tower sequence to the server. The server then matches the sequence of cell towers to the bus route and predicts the bus's arrival time.

Bojan [4] proposed an intelligent transportation system consisting of three components: the server system, the monitoring system, and the display system. A sensor system receives data from a global positioning system (GPS) and from near field communication (NFC), temperature, and humidity sensors. The monitoring system extracts meaningful information from the raw data collected from the sensor system and provides it to the bus driver. The display system shows bus and travel-related information to commuters at the bus stop.

Adachi [5] proposed a wireless sensor network with which the bus information system can provide the present bus location and estimated bus arrival times to users. Bus nodes, bus stop nodes, router nodes, and concentrators are parts of the network.

Xu [6] surveyed GPS, remote sensing (RS), and geographic information system (GIS) techniques and proposed an idea to utilize them all in order to represent the real-time status of each bus and bus arrival time on maps.

John [7] introduced a smart public transport system consisting of bus modules equipped with a GPS receiver, digital speedometer, telecommunications modem and other server modules, bus stop modules and client apps. This system provides information to users about the current locations of the buses approaching the stop.

Kim [8] introduced a security-augmenting scheme for a bus information system. Lin. [9] proposed using a genetic algorithm to find the shortest driving time with diverse scenarios of real traffic conditions and varying vehicle speeds. Wang and Zhang [10] proposed a two-step approach to vehicle detection for intelligent transportation systems.

3. The Existing Algorithm

Bhattarakosol [11] proposed a bus information system that can provide passengers with not only bus routes, but also valuable information on how to travel, the position of the buses, and bus travel-time approximations.

In order to provide travelling information, they find the bus routes that pass through both the start and destination bus stops. If no bus route passes through both start and destination stops, then for each of bus stop A that is on any of the bus routes that pass through the start stop, they find the bus routes that pass through both A and the destination stop.

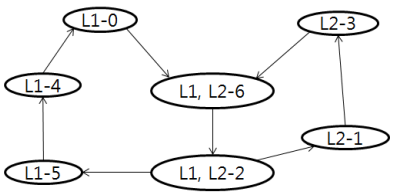


Figure 1. An Example Graph Representing Two Bus Routes, L1 and L2

For example, suppose a person wants to travel from L1-5 to L2-3 in Figure 1, where L stands for the bus route. In this example, L1-0 represents the bus stop with an ID of 0, and bus route 1 passes through the bus stop. "L1, L2-6" represent bus stop 6, which is on both

bus routes 1 and 2. The bus route L1/L2 is the only one that passes through bus stop 5/3. This implies that no bus route passes through both bus stops 5 and 3. Bhattarakosol investigated bus stops 4, 0, and 6 (in that order) to see if L2 passes through any of these bus stops. On learning that bus route L2 passes through bus stop 6, they conclude that the person should take the bus route L1 and transfer to bus route L2 at bus stop 6.

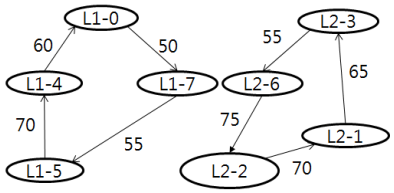


Figure 2. Another Sample Graph representing Two Bus Routes. The Bus-Searching Algorithm does not Know how to Travel from L1-5 to L2-3, even if L1-7 is Close to L2-6

4. Our Algorithm

The existing algorithm fails to find a path from L1-5 to L2-3, even though L1-7 is close to L2-6 if the bus routes are configured as shown in Figure 2. The arcs are labeled by the numbers representing bus travel times. For example, it takes 70 seconds for a bus to run from L1-5 to L1-4. In this example, bus routes L1 and L2 are disconnected. Therefore, we propose to connect close bus stops. For example, if it takes about 200 seconds to walk from L1-7 to L2-6, then we can represent the graph by the adjacent matrix shown in Figure 3.

Then, we find the shortest path from the start vertex to the destination vertex. For example, if the starting stop is L1-0 and the destination stop is L2-2, then we will find the following path: L1-0 -> L1-7 -> L2-6 -> L2 -2. Once we have the shortest path at hand, we find the intersection, "common bus routes", of the "start bus routes" and the "destination bus routes" where "start/destination bus routes" is a set of all bus routes passing through the start/destination stops. If "common bus routes" is not empty, we select the route where the distance from the start to the destination is the shortest. If this distance is not much greater than the distance of the shortest path, then we return this common bus route; otherwise, we return the shortest path.

	L 1-0	L 1-4	L 1-5	L 1-7	L 2-1	L 2-2	L 2-3	L 2-6
L 1-0	0	∞	∞	50	∞	∞	∞	∞
L 1-4	60	0	∞	∞	∞	∞	∞	∞
L 1-5	∞	70	0	∞	∞	∞	∞	∞
L 1-7	∞	∞	55	0	∞	∞	∞	200
L 2-1	∞	∞	∞	∞	0	∞	65	∞
L 2-2	∞	∞	∞	∞	70	0	∞	∞
L 2-3	∞	∞	∞	∞	∞	∞	0	55
L 2-6	∞	∞	∞	∞	∞	75	∞	0

Figure 3. The Adjacent Matrix Representing Figure 2

5. Implementation of the System

We implement a system called a bus travel path recommendation system (BTPR) that accepts a pair of bus stops (starting and destination) and returns the shortest path between them. The BTPR is a part of the bus information system described in Figure 4. There are two kinds of users: requesters and uploaders. An uploader is a bus passenger or a driver in a running bus that periodically sends his current location to the server. A requester is a user who is waiting for a bus. Our BPTR provides the shortest path to the requester. In addition to the shortest path, the BIS provides other information, such as the current locations of the buses approaching a designated bus stop, points of interest around a bus stop, the route of a given bus, and so on.

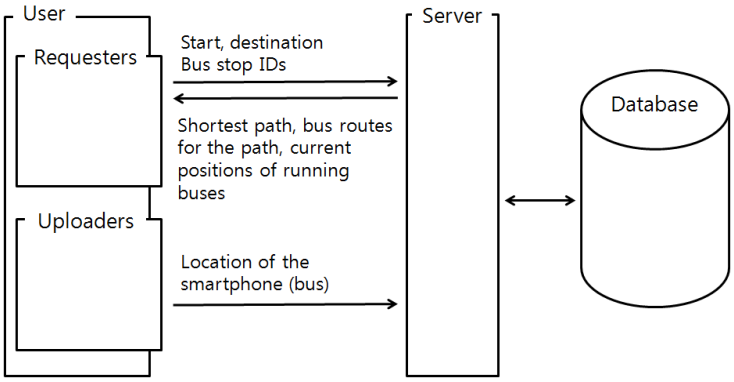


Figure 4. A Conceptual Model of the BIS

The BIS server consists of the recommender, bus location provider, data collector, and estimator components. The recommender finds the shortest path and the bus routes coinciding with the path, and sends them to the client. The location provider retrieves the current location of a given bus from the Trace Table and sends it to the client. The data collector receives bus locations from uploaders, saves the received data at the database, and updates the current location of the bus recorded in the Trace Table. The estimator calculates the time needed for a bus to travel between two bus stops. So, the [i, j] entry of the distance matrix represents the time currently needed for a bus to run from bus stop i to bus stop j. This paper focuses on the recommender and how to implement it.

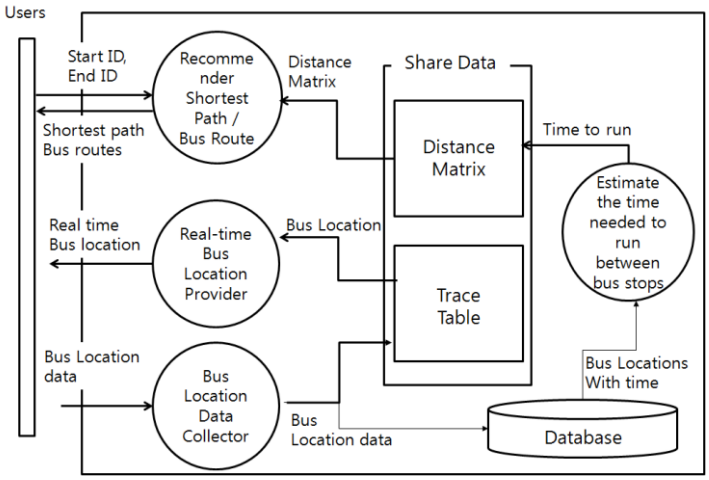


Figure 5. A Conceptual Model of the BIS Server

Given a distance matrix, the first step the recommender does is to insert the time needed for a person to travel on foot between two bus stops, where the distance between them is less than 100 meters. The second step for the recommender is to find the shortest path with the Dijkstra algorithm. Our Dijkstra algorithm terminates as soon as it finds the shortest path from the starting bus stop to the destination bus stop.

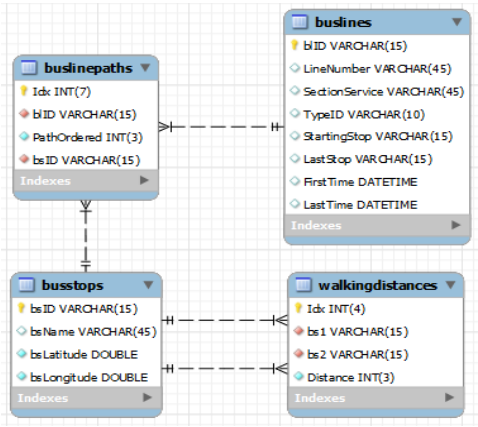


Figure 6. Part of the Database

With the shortest path, s_1, s_2,.. s_n, the recommender finds the common bus routes that go through both s_i and s_{i+1} for all i from 1 to n-1. By investigating the common bus routes, the recommender finds a sequence of bus routes where the number of transfers is fewest. Finally, it specifies the sections where the user has to walk.

We record bus stop information in the database consisting of four tables: bus stops, buslines, bus line paths and walking distances. In the bus stops table, we record information on all the bus stops in the city. Information on all bus routes in the city is recorded in the bus lines table. For each bus route, we record in the bus line paths table all bus stops the bus route goes through. Referring to this bus line paths table, we initialize the distance matrix and figure out the direction of a running bus. In the walking distances table, we record all pairs of bus stops (bs1, bs2) where the distance between bs1 and bs2 is less than 100 meters.

The process for the client is described in Figure 7. When the app is executed, it creates an ArrayList object and initializes it with all bus stop names in the city. Then, it creates a HashMap object and initializes it with (bus stop ID, bus stop name) pairs for all bus stop names. It creates another HashMap object consisting of (bus route ID, bus route name) pairs for all bus routes.

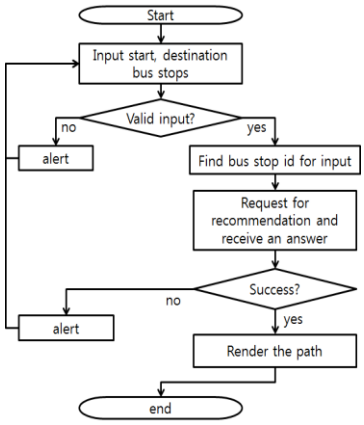


Figure 7. The Process of the Client

The app accepts names of the starting and destination bus stops. Using the ArrayList of bus stop names, it automatically completes the user input. Then, using the (bus stop ID, bus stop name) HashMap, it finds the bus stop IDs corresponding to the input, and sends the IDs to the server. Then, it receives a message representing the shortest path from the starting bus stop to the destination stop written in JSON from the server. An example message shown in Figure 8 represents the path: go to bus stop 421861 from 421860 on foot, then take the bus on bus route 1520126. The bus stops at 421861, 421902, and so on. Then, get off the bus at 421937. Go to bus stop 421936 on foot. Then, take the bus on bus route 1520002 and get off the bus at bus stop 421875, which is the destination bus stop. The values 1 and 2 of "means" in a message represent "by bus" and "on foot," respectively.


```
[
  - {
        p1: "421860",
        p2: "421861",
        means: "2",
        line: "null",
      - path: [
            "421860",
            "421861"
        ]
    },
  - {
        p1: "421861",
        p2: "421937",
        means: "1",
        line: "1520126",
      - path: [
            "421861",
            "421902",
            "421822",
            "421900",
            "421878",
            "421976",
            "421937"
        ]
    },
  - {
        p1: "421937",
        p2: "421936",
        means: "2",
        line: "null",
      - path: [
            "421937",
            "421936"
        ]
    },
  - {
        p1: "421936",
        p2: "421875",
        means: "1",
        line: "1520002",
      - path: [
            "421936",
            "421854",
            "421981",
            "421824",
            "422401",
            "421949",
            "421928",
            "421897",
            "421819",
            "421875"
        ]
    }
]
```


Figure 8. An Example JSON Message

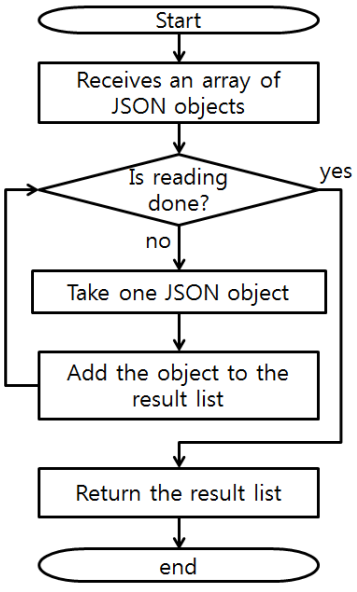


Figure 9. The Process of Parsing a JSON Message

The process of parsing a JSON message is described in Figure 9. It parses the JSON message into a Java object, jsonDatas, using the JSONValue.parse method and creates an ArrayList object, resultPaths. It repeats the following process while there is a JSON object in jsonDatas. It takes out one JSON object from jsonDatas, reads the values in the object, creates an element and assigns the values to the element, and appends the element to the resultPaths ArrayList. This process returns resultPaths when jsonDatas is empty.

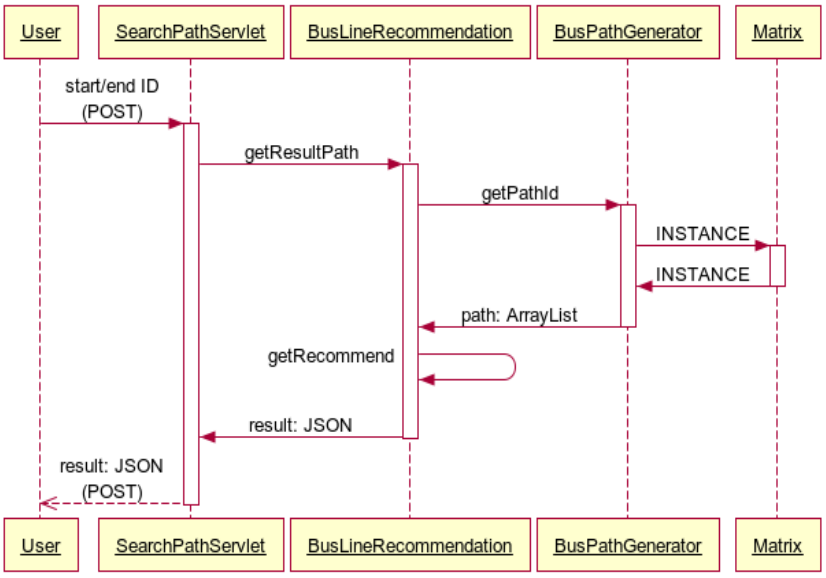


Figure 10. A Sequence Diagram Describing Interactions between Server Components

A sequence diagram describing interactions between server components is shown in Figure 10. We have the SearchPathServlet class that communicates with the clients. It receives a starting bus stop ID and a destination bus stop ID, forwards them to the BusLineRecommendation class, receives a recommendation, and forwards it to the client. The BusLineRecommendation class uses the BusPathGenerator class to obtain the shortest path from the starting to destination bus stops. Then it finds a sequence of bus routes that coincide with the shortest path and returns it to the SearchPathServlet class.

The BusPathGenerator finds the shortest path using the Dijkstra algorithm. Given the start vertex v_k, for all vertices v_j, the algorithm finds the shortest path from vk to vj. We modified the algorithm so it terminates as soon as it finds the shortest path from start to destination. The class also has a method that transforms the array representing the shortest path into an ArrayList and returns it to the caller.

The input to the Dijkstra algorithm is a distance matrix, M. An entry M[i,j] represents the time needed to reach bus stop j from bus stop i. The Matrix class creates the distance matrix reading the database with the methods defined in the BusLineSearchServiceDao class shown in Figure 11. Then, for all pairs of bus stops i and j, where the distance between bus stops i and j is less than 100 meters, it writes the time needed for a person to reach bus stop j from stop i on foot at M[i,j]. This matrix is defined as a singleton, and the class provides better methods with which other methods can access the data in the matrix. The methods to access bus information in the database are defined in the BusLineSearchServiceDao shown in Figure 11

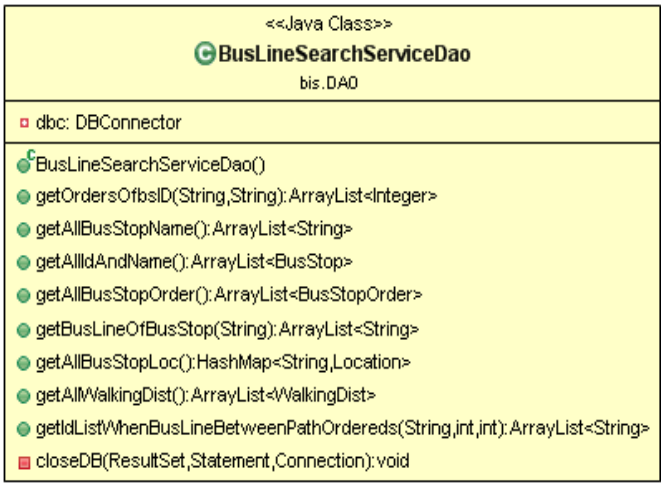


Figure 11. The Definitions of the Classes to Access Bus Information in the Database

Part of the main activity that handles the search button click event is shown in Figure 12. The most important role here is to create a GetBusLineServiceTask object and execute it if the input is valid.

```
btnSearch.setOnClickListener(new View.OnClickListener() {
public void onClick(View view) {
...
if (params.size() == 2 && mIsSearched) {
GetBusLineServiceTask task = new GetBusLineServiceTask(…ner);
task.execute(params);
} else {
return;
}
…
```

Figure 12. Part of Main Activity

Part of the GetBusLineServiceTask class is shown in Figure 13. The IP address of the server is assigned to the host variable, and the name of the servlet to which the request is sent is assigned to the path variable. The IDs of the starting bus stop and the destination bus stop are added to the paramList variable. Then we create an HttpPost object with the URI (uniform resource identifier) we created with the host and the path variables. After that, we create an UrlEncodedFormEntity object with the paramList. Then we execute the hypertext transfer protocol (HTTP) request and receive the response in an HttpResponse object. The content of the response can be read by InputStream.

```
…
private String host = "203…62";
private String path = "bis_server/SearchPathServlet";
…
paramList.add(new BasicNameValuePair("start", parameters.get(0)));
paramList.add(new BasicNameValuePair("end", parameters.get(1)));

private String requestPOST(ArrayList<String> parameters) {
…
uri = URIUtils.createURI(scheme, host, port, path, null, null);
HttpPost httpPost = new HttpPost(uri);
```



```
paramList.add(new BasicNameValuePair("start", parameters.get(0)));
…
UrlEncodedFormEntity entity = new UrlEncodedFormEntity(paramList, "UTF-8");
httpPost.setEntity(entity);
…
HttpResponse httpResponse = httpClient.execute(httpPost);
…
inputStream = httpResponse.getEntity().getContent();
if (inputStream != null)
result = convertInputStreamToString(inputStream);
```

Figure 13. Part of the Get Bus Line Service Task

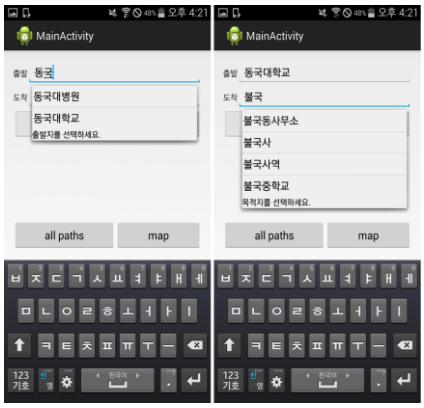


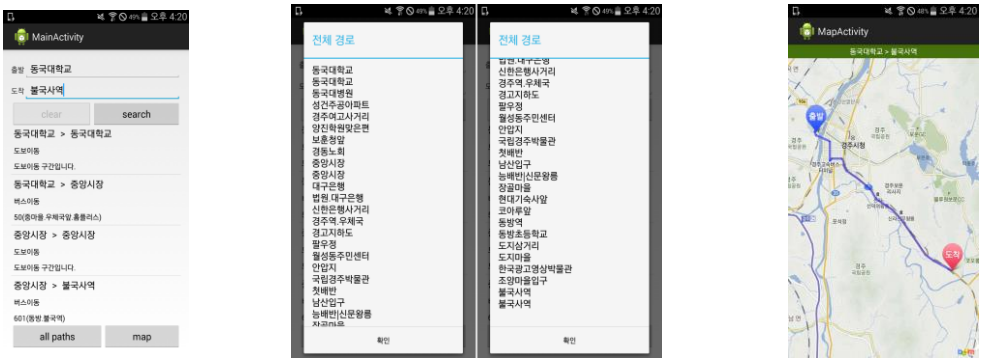
Figure 14. User Interface for Input

6. Experiments

We performed experiments in testing the system. Figure 14 shows the AutoCompleteTextView for input. After designating the start and the destination bus stops, the user clicks the search button in order to send an HTTP request to the server. When the user wants to rewrite the bus stops, he clicks the clear button in order to clear the text boxes and type again.

Figure 15(a) shows the route to reach the destination from the starting bus stop. It says that the user should walk to the starting bus stop, take bus route 50, get off the bus at the Central Mart stop, transfer to bus route 601, and get off the bus at the destination stop.

Figure 15(b) shows the list of all the bus stops the user will pass through from start to destination. When the user clicks the map button, the path will be drawn on the map as shown in Figure 15(c).



(a) Search result (b) Path in detail (c) Path on the map

Figure 15. User Interface to Show the Response

7. Conclusions

This paper designed and implemented a recommendation system that finds the shortest path between a starting bus stop to a destination bus stop and a sequence of bus routes and walking segments that coincide with the shortest path. There already are websites that find a bus travel path, including transferring to another bus route on foot. However, we are not sure if they can find the shortest path. For further study, we are planning to implement the whole BIS described in Figure 4.

Acknowledgments

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2011-0006942), by the '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), and by the Dongguk University Research Fund of 2015.

References

[1] J. Cheng, C. Lai, H. Chen and C. Ou, "The service quality analysis of public transportation system using PZB model — Dynamic bus information system", in Proceedings of the 40th International Conference on Computers and Industrial Engineering, pp. 1-5, **(2010).**
[2] R. Megalingam, N. Raj, A. Soman, L. Prakash, N. Satheesh and D. Vijay, "Smart, public buses information system", in Proceedings of the International Conference on Communications and Signal Processing, pp. 1343-1347, **(2014).**
[3] P. Zhou, Y. Zheng and M. Li, "How Long to Wait? Predicting Bus Arrival Time With Mobile Phone Based Participatory Sensing", IEEE Transactions on Mobile Computing, vol. 13, no. 6, pp. 1228-1241, **(2014).**
[4] T. Bojan, U. Kumar and V. Bojan, "An internet of things based intelligent transportation system", in Proceedings of the IEEE International Conference on Vehicular Electronics and Safety, pp. 174-179, **(2014).**
[5] H. Adachi, H. Suzuki, K. Asahi, Y. Matsumoto and A. Watanabe, "Estimation of bus traveling section using wireless sensor network", in Proceedings of the Eighth International Conference on Mobile Computing and Ubiquitous Networking, pp. 120-125, **(2015).**
[6] L. Xu, S. Gao and H. Zhangm "Dynamic bus monitoring and scheduling system based on ArcGIS Server technology", in Proceedings of International Conference on Computer Science and Information Processing, pp.1438-1441, **(2012).**
[7] R. John, F. Francis, J. Neelankavil, A. Antony, A. Devassy and K. Jinesh, "Smart public transport system", in Proceedings of International Conference on Embedded Systems, **(2014).**
[8] S. Kim, "Security Augmenting Scheme for Bus Information System based on Smart Phone," IJSIA 7(3), pp. 337-346, **(2013).**
[9] C. Lin, J. Yu, J. Liu, W. Lai and C. Ho, "Genetic Algorithm for Shortest Driving Time in Intelligent Transportation Systems", IJHIT. 2(1), pp. 21-30, **(2009).**
[10] H. Wang and H. Zhang, "A Hybrid Method of Vehicle Detection based on Computer Vision for Intelligent Transportation System", IJMUE. 9(6), pp. 105-118, **(2014)**
[11] P. Bhattarakosol, P. Tiewthanom and S. Chitwiriya, "Bus Information System: A smart partner for commuters", in Proceedings of the 5th International Conference on Computer Sciences and Convergence Information Technology, pp. 12-17, **(2010)**.

Authors

Won Joo Lee, he received his B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Hanyang University(ERICA Campus), Ansan, Korea, in 1989, 1991 and 2004, respectively. Dr. Lee joined the faculty of the Department of Computer Science at Inha Technical College, Incheon, Korea, in 2008, where he has served as the Director of the Department of Computer Science. He is currently a Professor in the Department of Computer Science, Inha Technical College. He has also served as the Vice-president of The Korean Society of Computer Information and the Editor-in-Chief for the Journal of The Korean Society of Computer Information. His interests are in parallel computing, internet and mobile computing, and cloud computing.

Jaegeol Yim, he received his MSc and PhD in Computer Science from the University of Illinois at Chicago, in 1987 and 1990, respectively. He is a Professor in the Department of Computer Science at Dongguk University at Gyeongju Korea. His current research interests include Petri net theory and its applications, location-based services, AI systems, and multimedia systems. He has published more than 50 journal papers, 100 conference papers (mostly written in the Korean Language), and several undergraduate textbooks.