Ubiquitous Bus Mapping System on Mobile Phone via Web Architecture

Chakchai So-In, Sarayut Poolsanguan, Chartchai Poonriboon and Natnicha Veeramongkonleod
Department of Computer Science, Faculty of Science, Khon Kaen Uniaversity
Maung, Khon Kaen, Thailand, 40002

***Abstract*—This paper introduces a new bus mapping system, especially for mobile usage. Web architecture is purposely used so as not to limit the system into a particular organization. Once a local map is given, an automated map will be ready-to use in the mobile device to interact with Global Positioning System (GPS) and mapping service API (Bing Map) features. The paper describes the instrument and component of the key design of the system. The feasibility of the architecture has been evaluated with Window Phone 7 (HTC HD7) within the campus. A well-known optimal shortest path algorithm, Dijkstra, is used to find the local route. In addition, especially for a large number of nodes or bus-stops, a heuristic method based on Dijkstra is proposed when a particular route needs to be recalculated, and the performance evaluation results comparatively show the substantial reduction of computational time with a slight increase of path cost.**

***Keywords*—Navigation System; Ubiquitous Bus Mapping System; Window Phone 7; Global Positioning System; GPS; Mapping Service; Dijkstra; Heuristic; Web Architecture**

I. INTRODUCTION

Nowadays, a mobile and wireless communication usage' trend has been increasing, especially a smart phone (mobile phone), which is globally beyond 5 billion in number by 2011 [1]. The popularity has come across over many reasons, such as a wide variety of features – camera, sensors, and GPS (Global Positioning System), an exponential increase of processing power and memory, an improvement of wireless transmission speed (toward beyond 3G), and cost-cutting to be affordable mobile equipment.

Aside from those hardware functions/features, the smart phone offers a variety of tuned up operating system, properly used for a mobile device with distinct characteristics (battery-operated), such as Window Phone 7 (WP7), Android OS, Iphone IOS, Blackberry (RIM), and Brada (Samsung). All these provide enriched features together with plenty of easy-to-develop, system application, and marketing approach (e.g., AppStores, AndroidMarket, and WindowPhone Marketplace) similar to that of a traditional PC-based architecture.

One of the useful applications aiding people globally connected is a mapping service-based system. Google Map [2] and Bing Map [3] are well-known mapping service providers, especially with the easy-to-use API acquiring the detailed public map, initially for PC-based architecture. Recently, there is an effort to make use of them being operated in mobile devices, especially for each individual organization usage [4-9], and the bus-mapping system is one of the examples utilizing these features.

Although those mapping services provide global location with detailed government and public knowledge mapping information, local mapping knowledge information, especially for individual usage purpose, may not be easily integrated.

As a result, in this paper, we explore the possibility to utilize the mobile phone (WP7) embedded with the GPS functionality and easy-to-use mapping service API (Bing Map) together with web architecture so as to integrate the bus-mapping system onto the mobile phone. The system assuages the limitation of the only-global mapping service in that it includes a local map manually created for any organization via web architecture. Then, the individual map based on local knowledge information will be created and loaded onto the mobile phone for navigational usage.

This paper is organized as follows. In Section II, we briefly survey recent research/proposals regarding the techniques for bus mapping architecture. Then, in Section III, the overview of our architecture is discussed. Section IV provides the detailed optimization (heuristic approach) to recalculate the new route. After that we discuss the performance of our heuristic approach versus the traditional shortest path algorithm, Dijkstra, in Section V. Finally, the conclusions are drawn in Section VI.

II. RELATED WORK

Due to the advance of GPS and mapping service, the navigation system has been improved for global use. Google Map [2] and Bing Map [3] are well-known examples for mapping providers, and recently, the integration of these is ported into a mobile device for convenient usage.

Although, these providers offer globally detailed maps based on public information, the local knowledge on individual map for each organization is not easy to integrate, and so several attempts are to create the individual map for local usage in several approaches.

†This work was supported in part by the grant from Research and Academic Affairs Promotion Fund, Faculty of Science, Khon Kaen University, Fiscal year 2010 (RAAPF)

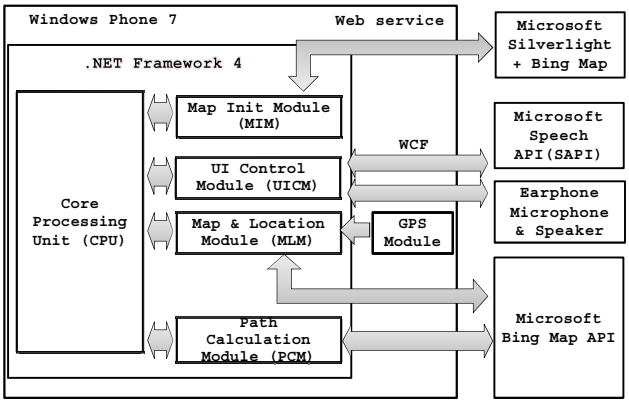


Fig. 1. Overall System Architecture

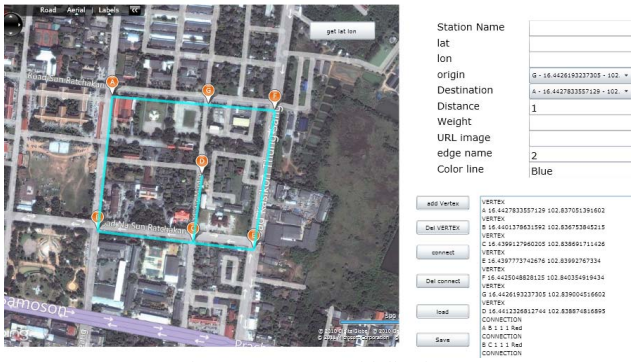


Fig. 2. Route-Map Initialization
[http://202.28.94.51/users/chakchai/TestPage.html]

For example, C. Yu and X. Xie [4] applied MapX for navigational use within Wuhan University (China). W. Guangchao et al., [5] used SuperMap Objects to construct the navigational system for Henan Polytechnic University (China).

R. Workman et al., [6], Y. Yang and J. Zheng [7], and The University College London (UCL) [8] made use of Google Map API to build a local-map for each individual organization, i.e., Xinjiang University and UCL.

Instead of utilizing well-known map providers, R. Jacob et al., [9] constructed the web navigational system, the so-called OpenStreetMap, for bus navigation using GIS (Geographic Information System) information and local knowledge within National University of Ireland Maynooth.

Notice that for all map systems described above, each has to individually create its own map using a specific local knowledge, and so the system cannot be re-used for other organizations, possibly due to the incompatibility.

Aside from mapping service provider perspectives, an algorithm to find the path or route is usually distinctive depending on the characteristic of the individual organization. However, a well-known technique, Dijkstra [10], is mostly used with some modifications to tune up for each organization requirement, such as A. M. Haziq et al., [11] utilized Dijkstra to figure out the shortest path for a local route in LRT systems (Malaysia). A* [10] algorithm can be also used. For instance, the bus finder system, the so-called Modified BUSTRAP, was developed by F. Siddiqui and A. Pandey [12]. This system is used for trip-planning purpose.

Consider a mapping service for mobile usage. Recently, there have been several attempts to port/integrate this service for mobile users with a variety of mobile operating systems, and so some concerns need to be brought up, such as the use of limited power consumption when only battery-operated.

For example, P. Deb et al., [13] and S. Vivas and G. Riano [14] developed Java ME-based navigational systems using Google Map API for mobile usage. Similarly, H. Yi [15] applied this API for mobile users in Bogota (Colombia). Notice that the power consumption is not explicitly in their concerns. In addition, the use of Bing Map, one of the biggest mapping providers, is not completely investigated and integrated into the mobile device.

From various techniques discussed above, noticing that there are advantages, but some useful techniques aren't considered probably depending upon the technology inadequacy. As a result, with the advance of current mobile phone technology, we have proposed a new system to collaborate necessity features to make a ubiquitous local mapping service via web architecture so as to be used (reused) for any organization.

We have also discussed the design choices of the system and the path-finding algorithm in this paper. The system has been implemented with enriched features into a HTC HD7 phone (Window Phone 7). In addition, the system has been tested by the bus system within the University (Khon Kaen University).

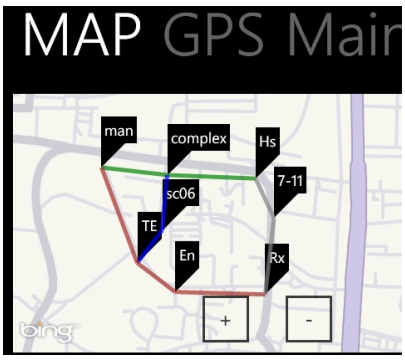
III. System Architecture

To integrate necessity components for the bus mapping system to be used ubiquitously, five main modules are constructed under a Microsoft .NET framework over Window Phone 7 (WP7) platform: Core Processing Unit (CPU), Map Init Module (MIM), Map and Location Module (MLM), User Interface Control Module (UICM), and Path Calculation Module (PCM) as shown in Fig.1.

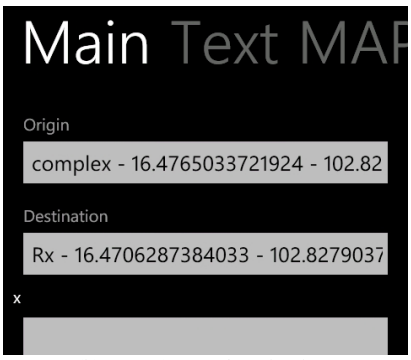
1) CPU: This module performs as the main function for computational purpose as well as cooperation with other modules to make the overall system running properly.

2) MIM: This module functions as the interaction module via web architecture. The initial detailed route-map is loaded directly from text file or via web URL, such as *http://202.28.94.51/users/chakchai/kkumap*. Note that the manual route-map was created using MS Silverlight and Bing Map API as shown in Fig. 2.

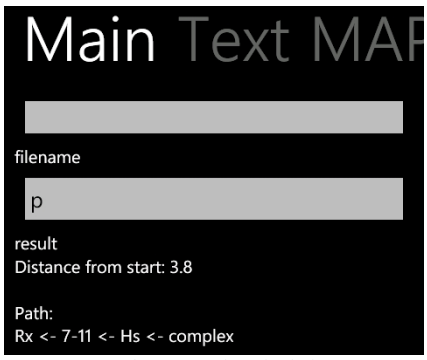
Consider the route-map on web. It functions as an initial map creator, properly used for an individual organization providing local knowledge, such as a specific bus-route. The administrator can manually point out the bus-stop information into the map, and then specific routes (source-destination) can be linked to create an organizational route-map. The GPS position, latitude/longitude, will be automatically generated accordingly.



(a) Map loaded from Web



(b) Input (Source/Destination)



(c) Shortest Path Calculation

Fig 3. Bus-Mapping System on Mobile Phone

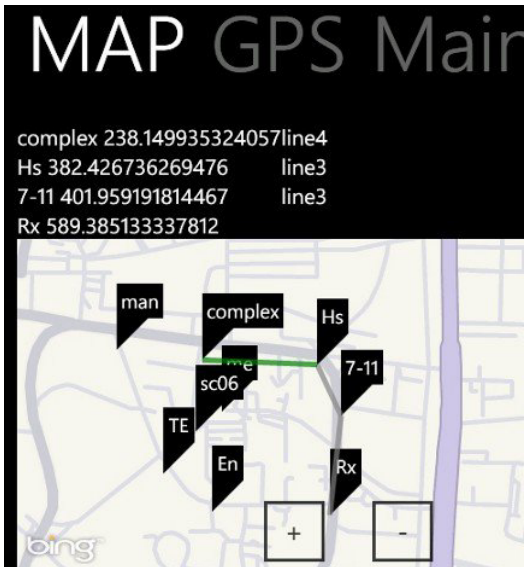


Fig. 4. Shortest Path Map

TABLE I
DIJKSTRA PSEUDOCODE

```
Node/Path Initialization (vertex v in Graph)
WHILE Q in set of all nodes IS NOT empty
     u := vertex in Q with smallest distance[]
     IF distance[u] = infinity THEN
          break
     ENDIF
     Remove u from Q
     FOR EACH neighbor v of u
          alt := distance[u] +
distance_between(u, v)
          IF alt < distance[v] THEN
               distance[v] := alt
               previous[v] := u
               Reorder v in Q
          ENDIF
     ENDFOR
ENDWHILE
```

In addition, the system offers different bus routes toward user-defined edges as the different edge numbers with individual color selection as an option; the color will show once the route-map is loaded into WP7. Notice that the administrator can provide extra information such as a traffic condition to provide an additional metric, *weight*, aiding the shortest path calculation process by not just taking the distance as the only main metric.

The system also supports user-defined description for each bus station. The image for a specific station can be stored for the purpose of easy-to-remember stopping point when loaded into the mobile phone.

3) MLM: This module functions as a navigational system over window phone 7. After the initial route-map has been properly set up, MLM integrates a bundled GPS module to the mapping system in order to update the current location during the journey. In addition, each passing-by station will be updated to inform the traveler not to miss out the key location.

4) UICM: This module performs a voice recognition function via Microsoft Speech API (MSAPI) as a conversion from text to speech of the guidance system, i.e., turning left/right or going straight including key location when passing by. Window Communication Foundation (WCF) is also used to communicate between a client (mobile device) and a MS SAPI server. Finally, a voice direction will be forwarded to earphone or speaker.

5) PCM: The main function of this module is to find the shortest path from the source toward the destination. A well-known algorithm, Dijkstra [10], is primarily used for that purpose. Table I shows the pseudocode of this algorithm. However, especially for path recalculation, a heuristic approach is applied to reduce a computational time which will be explained in details in next Section (Section IV).

Note that Figs. 3 and 4 show an overview of user interface for the bus-mapping system on WP7. After each individual map is generated toward Web featured by MS Silverlight, this map will be loaded onto WP7 either by text file or from a given URL.

In this example (Fig. 4), there are four routes: (Line 1 – Brown) *man*↔*TE*↔*En*↔*Rx*; (Line 2 – Blue) *TE*↔*sc06*↔*complex*; (Line 3 – Gray) *H2*↔*7-11*↔*Rx*; (Line 4 – Green) *man*↔*complex*↔*Hs* in which each node is an actual bus-stop within the University. In this case, the origin is at *complex* and *Rx* for the final destination. Since there is no direct route toward the destination, the system will calculate the shortest path through two routes (buses): taking the bus from *complex*→*Hs* (Line 4) and then *Hs*→*7-11*→*Rx* (Line 3).

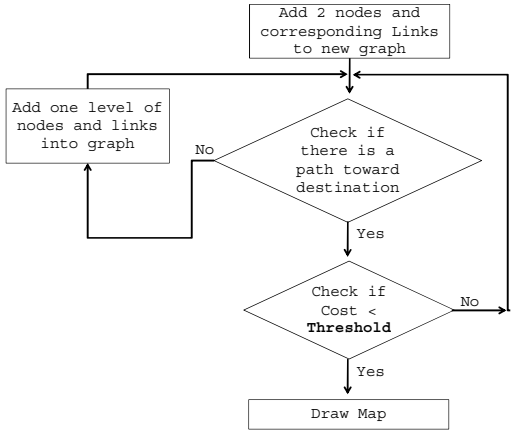


Fig. 5. Flowchart: Heuristic Approach based on Dijkstra

TABLE II
PSEUDOCODE: HEURISTIC APPROACH BASED ON DIJKSTRA

```
Crate new graph G
Add 2 nodes and links into G
WHILE (NotFound Path (Dijkstra))
      IF a new node k not in set of N THEN
         Add k in N
         Add links l connecting k into G
      ENDIF
END WHILE
```

IV. PATH FINDING ALGORITHM

As discussed in the previous section, the Dijkstra algorithm is applied to figure out the shortest path toward the destination. However, especially for path recalculation when a large number of nodes are involved, the computational time will be also large. As a result, we have applied a heuristic method by using Greedy approach [10] to reduce a number of considered nodes, and so the recalculation time will be substantially reduced.

Fig. 5 and Table II show a flowchart and a pseudocode of the heuristic approach. In general when the link is broken, instead of recalculating for a whole graph using Dijkstra, we only consider a subset of nodes which is closer/closest to the broken link with the assumption that a possible shortest path should be close to that as well.

With the use of list implementation, from the nodes that link is broken, we increase a group of nodes in each level, and then Dijkstra is performed only with nodes in that group. Noting that it's obvious that the probable path may not be the shortest one; however, the computational time will be reduced substantially. The results have been shown in Section V.

Notice that in Fig.5, the accuracy of the heuristic approach can be tuned up according to the threshold. The proper setup should be close to Dijkstra with the computational time trade-off. Noting that based on our evaluation (Section V), only 2 levels can bring the path cost close to that in all nodes scenario.

V. PERFORMANCE EVALUATION

In this section, we performed the evaluation process so as to justify the feasibility of a well-known path-finding algorithm, Dijkstra, to our system. We only focused on the recalculation process. Then, we evaluated the performance of our heuristic approach when only considering a smaller group of nodes once the path was available. Finally, Window Phone 7 (HTC HD7) was integrated with the optimized algorithm to confirm the feasibility.

A. Simulation and Experimental Setup

We evaluated our system into two main scenarios to test the performance and the feasibility as follows.

1) To show the performance of the two path-recalculation algorithms, we first scale the nodes from 100, 1,000 and 4,000, respectively to perform the test. The cost of each link is randomly chosen between 1 and 20,000. We used PC with a standard configuration over Window 7 Service Pack 1 operating system (32 bits): CPU Intel(R) Core 2 Quad 2.66 GHz (6 MB L2 Cache), 2048×2 MB DDR-SDAM, 1 TB Disk. In addition, the simulator is written in C# (MS .NET 2010 framework).

Second, in each set, we ran the Dijkstra algorithm to find out the shortest path. Notice that the links among nodes toward the destination are randomly chosen in which each node has at least 4 edges inter-connected with randomly chosen 0-2 edges as an extra. After that we randomly cut the link and performed the reconstruction process to find another route toward the destination. We ran 10 tests for each set and calculated a mean and standard deviation as our main metrics. Again, we only consider the path-recalculation process for performance comparison purpose.

Third, to evaluate the accuracy of the algorithm as a trade-off with the complexity, we simulated up to 2-level tests in that the first level only includes the necessity nodes that are enough to reconstruct the first found path toward the destination. Thus, the second level will include more nodes which are enough to find at least 2 possible paths, and so the probability to find the least path-cost is increased, and so on.

2) To illustrate the real performance, we evaluated the system response time for our heuristic approach. The experiment is over Window Phone 7 (HTC-HD7).

B. Simulation and Experimental Results

The first scenario has showed the average and standard deviation of the computational time as well as the path cost of two algorithms as shown in Table III and IV.

Table III shows that comparing to a traditional Dijkstra algorithm, the optimized version' computational time has substantially decreased, especially when a number of nodes are large, i.e., with 10,000 nodes, the time reduction is over 100% (from approximately 240 to 120 second), and surprisingly the path-cost incurs the same (Table IV).

Consider 4,000 nodes scenario. The computational time was reduced by a factor of 100% with the increase of path-cost slightly over 2.68%.

However, when considering less number of nodes, i.e., 100 nodes scenario, the computational time is reduced in a smaller factor, i.e., over 4% with the increase of path-

cost over 0.5%. That result has implied that the optimized one will be efficiently used for a large number of nodes scenario.

The reason we considered the 2nd level test is because when performing only the first level test, the result is under our justification. For example, we did perform the 1st level test for 4,000 nodes which the result showed the reduction of computational time is around 100% but the percentage of the path-cost increases over 13%.

TABLE III
COMPUTATIONAL TIME (SECOND): DIJKSTRA VS. OPTIMIZED DIJKSTRA

	Dijkstra		Optimized Dijkstra	
#Nodes	Mean	Std.	Mean	Std.
100	0.0248	0.0032	0.0239	0.0017
1,000	2.4571	0.4090	1.2118	0.0848
4,000	34.2282	1.5501	17.1576	0.9520
10,000	241.2470	13.0597	122.7435	1.8492

TABLE IV
PATH COST (UNIT): DIJKSTRA VS. OPTIMIZED DIJKSTRA

	Dijkstra		Optimized Dijkstra	
#Nodes	Mean	Std.	Mean	Std.
100	3,985.50	2,235.57	4,004.80	2,202.74
1,000	4,047.10	2,345.55	4,047.10	2,345.55
4,000	4,803.90	3,377.07	4,930.30	3,431.89
10,000	4,588.83	2,437.48	4,588.83	2,437.48

We also performed the 3rd level test for 4,000 nodes scenario, and the results show that the accuracy is 100% with the increase of computational time. As a result, based on our extensive evaluation, the implementation beyond the 2nd level may not necessarily be used.

The second scenario shows the average and standard deviation of the computational time as well as the path cost in 1,000 nodes scenario when running on WP7. The computational time results in a mean of 7.66 second with a standard deviation of 0.48 second for the optimized Dijkstra comparing to 14.46 and 0.95 seconds, respectively for the traditional one. The path cost after recalculation is the same. In addition, consider the results from Table III, the computational time on WP7 increases due to the hardware efficacy. However, with less than 100 nodes, the computational time is below few seconds.

VI. CONCLUSION AND FUTURE WORK

New bus mapping system for mobile devices to navigate travelers with GPS functionality is proposed in this paper. We utilize web architecture to ubiquitously integrate an individual local map based on local knowledge to the global map system toward Bing Map API. The system has been integrated onto a Window Phone 7 (HTC HD7).

A well-known Dijkstra algorithm, the optimal shortest path algorithm, is mainly used as the route finding algorithm. Additionally, a heuristic approach is proposed to aid the Dijkstra during the route recalculation, especially for a large number of nodes. The results show the computational time reduction with a small increase of path cost as a trade-off.

Noting that in this paper, we chose the use of Window Phone 7 for our system; however, other similar mobile phone technologies are also applicable, e.g., Android and Iphone mobile devices. In addition, due to the use of GPS, obviously, it has come with cons, i.e., problems with the indoor-signal acquiring; however, it is not our scope of this paper. We focus on outdoor mapping services.

Although the system can be integrated with a global map and ready-to-use for mobile users, the graph abstraction is only assumed to be an undirected graph for most of the bus-systems. The concern of a directed graph for a particular map is for further investigation; however, we believe only a portion of the algorithm and some consideration will be minor change as an improvement.

Finally, especially for a large number of nodes, a technique similar to cloud computing [16] can be applied in order to migrate the computational time complexity for WP7; however, this is also beyond the scope of this paper.

REFERENCES

[1] Global mobile statistics 2011. Available at "http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats"

[2] Google Maps "http://code.google.com/apis/maps/index.html"

[3] Bing Maps. Available at "http://www.microsoft.com/maps/developers/web.aspx"

[4] C. Yu and X. Xie "Campus navigation system for new students based on.net technology," *Journal of Fujian Computer*, vol. 10, pp. 121–122, 2008.

[5] W. Guangchao, D. Yinsheng, Y. Yongqiang, X. Jun, and Y. Mingbo, "Design and realization of campus navigation system for henan polytechnic university," *Journal of Geomatics*, vol. 33, pp. 38–40, 2008.

[6] R. Workman, A. Gschwender, and J. L. Chan, "Campus google map applications," In Proc. *of the NorthEast Regional Computing Program*, 2005.

[7] Y. Yang and J. Zheng, "Developing campus information service system based on googlemaps," In Proc. *of the 6th Seminar on Cartography and GIS of China*, pp. 555–562, 2008.

[8] UCL, UCL Campus Route Finder. Available at "http://crf.casa.ucl.ac.uk/"

[9] R. Jacob, J. Zheng, B. Ciepluch, P. Mooney, and A. C. Winstanley, "Campus Guidance System for International Conferences Based on OpenStreetMap," *Lecture Nodes on Computer Science*, vol. 5886, pp. 187–198, 2009.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms," 1312 pp., The MIT Press, July 2009.

[11] A. M. Haziq Lim, N. S. Wan Sazli, B. Hussin, A. A. Azlianor, S. M. Suhaizan, and K. Massila, "A Dijkstra's Mobile Web Application Engine for Generating Integrated Light Rail Transit Route," *WSEAS TRANSACTIONS on COMPUTER*, vol. 9, no. 1, January 2010.

[12] F. Siddiqui and A. Pandey, "Modified BUSTRAP: An Optimal BUS Travel Planner for Commuters using Mobile," *International Journal of Electronics Engineering*, vol. 2, no. 1, pp. 5–9, 2010.

[13] P. Deb, N. Singh, S. Kumar, and N. Rai, "OFFLINE NAVIGATION SYSTEM FOR MOBILE DEVICES," *International Journal of Software Engineering & Application*, vol.1, no.2, April 2010.

[14] S. Vivas and G. Riano, "Mobile Device Programming for Finding Optimal Path on a Fast–Bus System," In Proc. *of the IEEE Systems and Information Engineering Design Symposium*, pp. 196–201, April 2006.

[15] H. Yi, "The Study and Implementation of Mobile Digital Maps System," In Proc. *of the International Conference on Information Engineering and Computer Science*, December 2010.

[16] C. Doukas, "Mobile Healthcare Information Management utilizing Cloud Computing and Android OS," In Proc. *of International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1037-1040, 2010.