RFID Based Navigation for Tourists

Sibani P. Senji and Nethra Priyadarshini J.

***Abstract*—The main problem that tourists face in India is related to navigation .since they are not able to communicate properly because of varied languages present in India .They do not use the public transport system too. The system suggested in this paper is to guide such visitors from other country and also people from other states in the country.**

RFID-based public transport ticketing systems rely on widespread networks of RFID readers that locate the user within the transport network in real time to be able to verify whether he can travel at that time with the ticket he holds. This paper presents a system that uses that same RFID-based location information to give the user navigation indications depending on his current location, provided that the user has indicated beforehand the places he intends to visit. This system is designed to be cost effective.

***Index Terms*—KIOSK, navigation, RFID, SOAP**

I. Introduction

The architecture of common public transport RFID-ticketing systems can be seen in Fig. 1. Such ticketing systems usually consist in a network of RFID readers placed at the entrance to each transport, be it underground, tram, bus or train. The users carry tickets that are passive RFID cards and can be loaded with trip details .The user uses his card at the entrance to validate his trip.

The system verifies if RFID car is charged to the required amount.

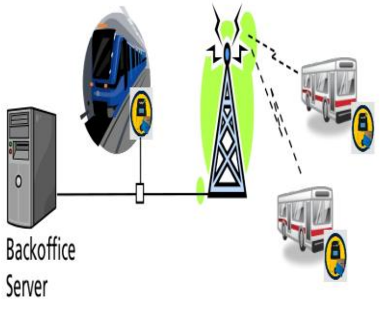

Fig. 1. Architecture

Since each user of the PTN must carry an RFID ticket and validate it each time he enters a transport, the system knows where the carrier of a certain card is at a certain moment. So, a public transport system with RFID based ticketing inherently can provide the real-time localisation of a card in the PTN. This paper describes a personal navigation systems that exploits this real time geographic localisation to navigate the user in the public transport network [1]. Kiosk is the interactive computer used for checking validity of RFID card and helps user in selecting locations he wants to visit by providing relevant details. It has been used in many counties because with integration of technology kiosk can perform a wide range of functions [2].

II. System Architecture

The system is designed to provide the service described is depicted in Fig. 2. Although the navigator is an enhanced service on top of the PTN with RFID ticketing, an open system architecture that enables different necessary services to be provided by different providers is used and, ultimately, even the navigation service can be operated independently of the PTN. Besides information on the bus schedules, it is also necessary to have a geographic information system. Only geographically indexed points of are supported destinations, so it is necessary to have the geographic coordinates of the PTN stations [3]-[5].

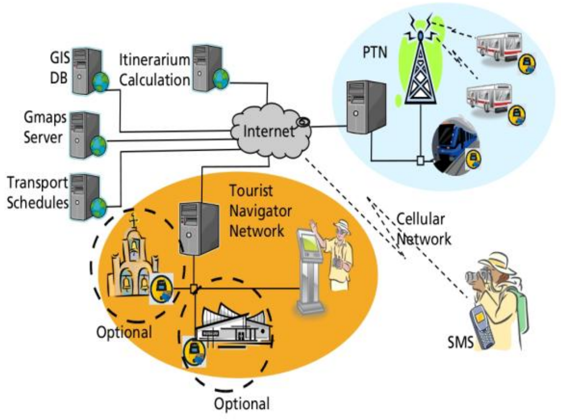

Fig. 2. Service provided

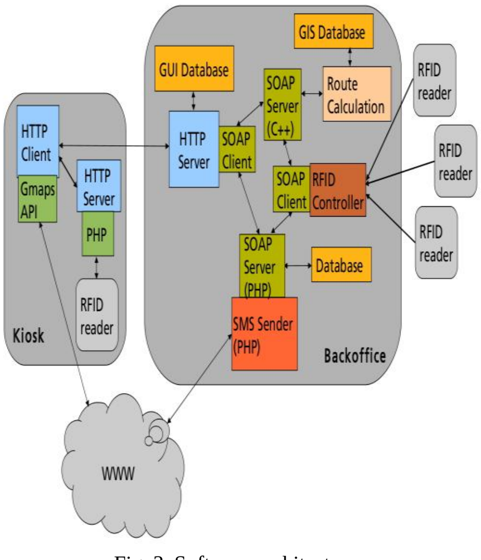

Fig. 3. Software architecture

Manuscript received July 1, 2013; revised September 2, 2013.

The authors are with Sri Sai Ram Engineering College, India (e-mail: abisibs@gmail.com).

III. Software Architecture

The software architecture for the service is shown in Fig. 3. The use of Web Services Description Language (WSDL) with SOAP provides an open system architecture with clear, well-defined interfaces between modules. This enables the functionality of the modules to be provided by different specialised external entities and also for easy replacement or extension of each module. Furthermore, the use of a software architecture centred on web services offers easy platform portability. This preparation consists in transforming the GIS data and bus schedules into a weighted graph on which the best way for the user to visit his desired destinations can be calculated [6], [7]

IV. Positioning of Tourists

The network of RFID readers distributed throughout the PTN are transmitted to the navigator system through a SOAP module. Everytime a card is validated on a transport, the RFID controller sends the card id read and the identification of the reader where it was validated to the RFID controller in the backoffice. These calls are the actual positioning system, as they locate the user inside the PTN network. Upon reception of a card reading, the controller calls the service that calculates the next steps of the user, automatically generates the SMS text and then calls the SMS sender service [1].

V. Itenary Calculation

The total graph is divided in zones and the shortest paths between all nodes in each zone are calculated, followed by the shortest paths between the boundary nodes of all zones. Since the nodes of a PTN are fixed, these optimal paths are pre-calculated offline using the A* [2] algorithm instead of Dijskstra [2] for better performance. The destinations chosen by the user are the only nodes of the graph that arenot known in advance and the shortest paths between them and the PTN nodes must be calculated in real time.

After the desired subset of nodes to be visited is known, the shortest paths between all of them are calculated. These paths consist of one or more edges of the original graph, passing by one or more stations, with or without transport changes, so one path can consist of more than one transport. Our algorithm takes this into account and adds a fixed waiting time each time that a transport change takes place, so that paths where a single transport is taken are preferred to paths with changes of transport.

After calculating the shortest paths between the nodes to be visited, a new graph can be built containing only the nodes that the user wants to visit and the edges are the best paths between them .The problem now is to find the best way to visit all the nodes in this new graph, which is equivalent to finding the shortest spanning tree, i. e. the minimum weight tree that contains all the nodes in the graph of the nodes the user wants to visit.

Because the calculation of the shortest spanning tree was taking too long for a real-time service, a branch and bound algorithm was implemented to reduce the computation time. [8], [9], [10].

VI. A* Algorithm

It is a computer algorithm that is widely used in path finding and graph traversal, the process of plotting an efficiently traversable path between points, called nodes. A* uses a best-first search and finds a least-cost path from a given initial node to one goal node. As A* traverses the graph, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments along the way. It uses a knowledge-plus-t uses a knowledge-plus-heuristic cost function of node *x* to determine the order in which the search visits nodes in the tree. The cost function is a sum of two functions:

1) The past path-cost function, which is the known distance from the starting node to the current node *x*.
2) A future path-cost function, which is an admissible "heuristic estimate" of the distance from x to the goal

The $h(x)$ part of the $f(x)$ function must be an admissible heuristic; that is, it must not overestimate the distance to the goal. Thus, for an application like routing, the heuristic estimate might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.

Like breadth-first search, A* is complete and will always find a solution if one exists. If the heuristic function *h* is admissible, meaning that it never overestimates the actual minimal cost of reaching the goal, then A* is itself admissible (or optimal) if we do not use a closed set. If a closed set is used, then h must also be monotonic (or consistent) for A* to be optimal. This means that for any pair of adjacent nodes *x* and *y*, where $d(x, y)$ denotes the length of the edge between them, we must have:

$$h(x) <= d(x, y) + h(y)$$

This ensures that for any path *X* from the initial node to *x*:

$$L(X) + h(x) <= L(X) + d(x, y) + h(y) = L(Y) + h(y)$$

where *L* is a function that denotes the length of a path, and *Y* is the path *X* extended to include *y*. In other words, it is impossible to decrease (total distance so far + estimated remaining distance) by extending a path to include a neighbouring node. A* will behave like depth-first search among equal cost paths.

A* is admissible and considers fewer nodes than any other admissible search algorithm with the same heuristic. This is because A* uses an "optimistic" estimate of the cost of a path through every node that it considers—optimistic in that the true cost of a path through that node to the goal will be at least as great as the estimate. But, critically, as far as A* "knows", that optimistic estimate might be achievable.

While the admissibility criterion guarantees an optimal solution path, it also means that A* must examine all equally meritorious paths to find the optimal path. It is possible to speed up the search at the expense of optimality by relaxing the admissibility criterion. Oftentimes we want to bound this relaxation, so that we can guarantee that the solution path is no worse than (1 + ε) times the optimal solution path. This new guarantee is referred to as ε-admissible [10]-[11]. A* algorithm is depicted in Fig. 4.

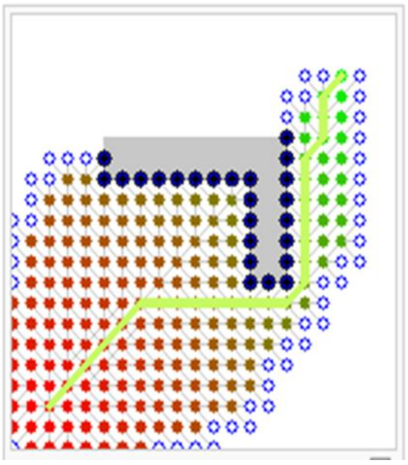
Fig. 4. A* algorithm

VII. Navigation Instructions

Each time the user passes his RFID ticket in a reader at the entrance of a transport, the answer SMS is automatically calculated by a service adjacent to route calculation. First, it is verified whether the user is where he is expected to be and the next destination to be visited is determined. If the user is where the system expects him to be, the route to the next unvisited destination is calculated. If the user has lost his way, the whole route is re-calculated with the current station as starting point and all unvisited destinies as list of destinations. Then, the user is guided as if he were not lost.

Each time, after the system knows the route to the next destination, the SMS text is automatically built, containing the necessary information to guide the user to the next ticket validation. That information is:

1) The station to get off the current transport;
2) The destinations to visit by foot next to that station, if any;
3) The station to take the next transport, if different from the one to get off;
4) The next transport to take.

With this information, the user knows what he needs to visit the places he desires and reach the next RFID reader that functions as the "positioning" device and triggers the next navigation information.

The procedures described here guarantee that the shortest paths between the nodes to be visited can be calculated online within times that are acceptable to users as real-time, guaranteeing the usability of the system. [1]

VIII. Conclusion

This system is designed to help people mainly visitors find their Way through the city .It is cost efficient method that also help in increase the usage of public transport rather than private vehicles which would lead to increased revenues for the government. It also provides comfort to the tourists and helps promote tourism.

References

[1] A. Aguiar, F. Nunes, and M. Silva, "Dirk Elias Fraunhofer Portugal Research Centre AICOS Porto, Portugal," Personal Navigator for PTN Using RFID.

[2] Interactive Kiosk –Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Interactive_kiosk

[3] B. Caulfiel and H. Reed. Cta bus tracker api - an api for thechicago transit bus system. [Online]. Available: http://ctabusapi.jottit.com/

[4] TriMet. Transit tracking service for portland, Oregon. [Online]. Available: http://developer.trimet.org/

[5] B. A. R. Transit. Transit tracking service forsan francisco,California. [Online].Available:http://bart.gov/schedules/developers/index.aspx.

[6] W3C, Web Services Description Language (WSDL). [Online]. Available: http://www.w3.org/TR/wsdl

[7] Google. Google maps api. [Online]. Available: http://code.google.com/apis/maps/.

[8] A. Drozdek, *Data Structures and Algorithms in C++*, 3rd ed. Thomson Course Technology, 2005.

[9] N. Jing, Y. W. Huang, and E. A. Rundensteiner, "Hierarchicaloptimization of optimal path finding for transportationapplications," in *Proc. of ACM Conference on Information and Knowledge Management*, 1996, pp. 261-268.

[10] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering route planning algorithms," *Algorithmics of Large and Complex Networks*, 2009.

[11] A* Algorithm-Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/A*_search_algorithm

Sibani P. Senji was born in Bangalore, India. Sibani is currently pursuing B.E in electronics and communication. Her area of interests include networking and wireless communication.

Nethra Priyadarshini J. was born in Chennai, India. Nethra is currently pursuing B.E in electronics and communication. Her area of interests include networking and communication.