Design and Implementation of an Internet Public Transportation Guidance System*

Sukang Bae†, Sungyoung Lee†, Dae-Soon Choi††, Hyonwoo Seung†††, Tae-Choong Chung†

† Department of Computer Engineering, Kyunghee University, Korea
{sylee@oslab.kyunghee.ac.kr}
†† Korea Institute of Construction Technology, Korea
††† Department of Computer Science, Seoul Women's University, Korea

contract KICT#96-0147

Summary

This paper introduces the design and implementation of a public transporta-tion guidance prototype system, which enables the WWW (World-Wide Web) to provide users with easier access and use. The proposed system is composed of three subsystems: client/server interface, knowledge based path search system and traffic data storage system. The user interface utilizes Java language to furnish users with multimedia data accessibility and interactiveness. The path search system produces optimal solutions based on dynamic traffic data, while previous search systems are limited to using static traffic data such as bus/subway route information. The storage system is designed to give the search system more efficient access to traffic information. The constituent sub-systems are inter-connected on the WWW using CGI (Common Gateway Interface). A client requests the server to search a path. The server asks the search system through CGI to get a result from the database, and returns it to the client. The system can be extended to an integrated navigation system which includes a variety of information from the Internet in addition to traffic information.

INTRODUCTION

The increase in personal mobility in metropolitan areas is notorious cause of traffic congestion, which in turn causes environmental pollution, wasted time and road accidents. Fortunately, current infor-mation technology and mobile communi-cations can help to alleviate some of these problems, though without directing people to public transportation, those problems can not be solved ultimately. More countries are getting interested in APTS (Advanced Public Transportation Systems) as a part of an ITS(Intelligent Transportation System) which provides users with a variety of convenient and accurate public traffic information services and the public transport industry with economical and efficient traffic manage-ment techniques.

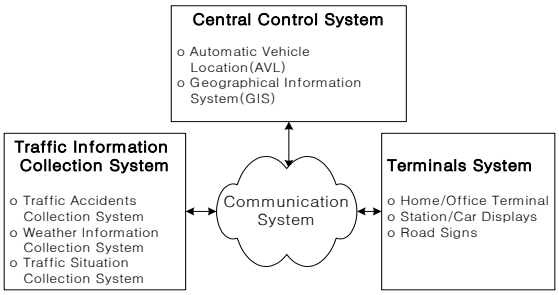
An APTS consists of four subsystems: Traffic Information Collection System, Central Control System, Communication system and Terminal System. The traffic information collection system provides weather information and road condition or accident information collected from beacons and car detectors. The central control system controls the whole system, processes users' requests and commands the AVL (Automatic Vehicle Location) system which in turn utilizes GIS (Geographical Information System) and GPS(Global Positioning System). The communication system transmits data collected from the information collection system through the

* This work has been partially supported by

network(wire or wireless) to the central control system, and returns the processed results to the users. The terminal system is the interface through which users or the system manager communicate with the system.

This paper introduces our experience for developing a public transportation guidance system as a prototype of APTS which enables the WWW to provide users with easier access and use. The proposed system aims to provide users with accurate and live on-the-spot traffic information, and optimal paths from the places where they are to the destinations based on the current traffic situation.

The system is composed of four subsystems: clients and the server connected together through the Internet, the path search system, the traffic data storage system and the traffic raw-data management system. While existing traffic



[Figure 1] General Structure of APTS

guidance systems provide traffic infor-mation fragmentarily in texts or images, the client utilizes a Web browser such as Netscape or MS-Explorer as its user interface, and makes the best of the Java programming language to provide state-of-the-art user interfacing techniques (3)(4)(5) (7)(13). On the Web, through a graphical browser, users can activate a hyperlink, read multimedia traffic infor-mation, or down-load a file within a single mouse click. In the path search system, a modified A* algorithm with a conditional pruning technique is used to consider various search conditions such as the number of routes to search or transfers, instead of adopting the dynamic pro-gramming technique which is generally used to search a shortest path. The traffic data storage system maintains the current traffic situations, stores stations/ routes and their geographical data, and provides routing information for the path search. The traffic raw-data management system is an easy-to-use graphical user interface which enables the system manager to register, modify, update or delete stations/routes through visual maps on the screen. Each module of the system is connected using the CGI and JDBC(Java Database Connec-tivity) (2)(11).

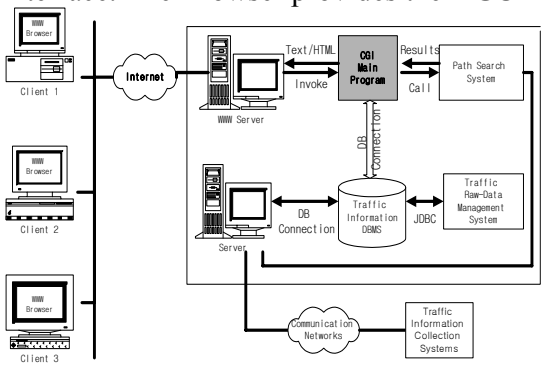
Since the system uses Java, which is object-oriented and platform-independent, it is easy to integrate with systems on the Internet running on heterogeneous machines as well as the other commercially available traffic information systems. It is also easy to alter the system within a relatively short period of time in case the user requirements change, and, consequently, easy to keep the system maintenance cost relatively low. While previous systems use exclusive terminals furnished at bus/subway stations or public institutions, the proposed system can be used on PCs, NCs(Network Computers) or Internet TVs at home or offices since it is mounted on the Web. Furthermore, the system can be directly plugged into ITS in the future.

The remainder of this paper is organized as follows. In Section 2, the system archi-tecture is described. Section 3 discusses the client part and server part of the system. The path search system and the traffic data storage system are presented in Section 4 and 5, respectively. In Section 6, the traffic raw-data management system is discussed. Finally, our conclusion and plans for future work appear in Section 7.

SYSTEM ARCHITECTURE

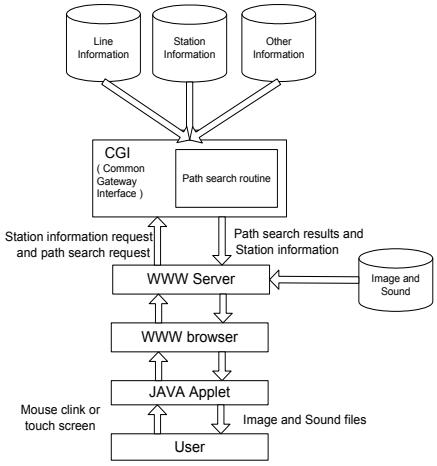
The physical structure of the proposed system is shown in [Figure 2]. It consists of clients and the server, the path search system, the traffic data storage system and the traffic data management system.

The client has a Web browser such as Netscape or MS-Explorer as its user interface. The Browser provides the GUI



[Figure 2] System Architecture

environment which enables users to see Web pages and to access other pages with a click of a mouse button. A Web page can have any multimedia traffic information such as formatted or unformatted text, images, sounds and videos included in it. The browser loads such pages and displays them. The traffic information contained on the pages is formatted in HTML(Hypertext Markup Language) and Java Applet/Script. The CGI is used in order to interconnect the client to the server where the path search system and the traffic data storage system reside. The client and the server commu-nicate with each other using a common protocol, HTTP(Hypertext Transfer Protocol) over the TCP/IP. The client requests the server to search a path. Then, the server asks the path search system through CGI to get a result from the traffic database, and returns it to the client. [Figure 3] shows how the mechanism works.



[Figure 3] Data Flow Diagram of the System

The real-time traffic data gathered from the traffic data collection system should be used for the path search system to find optimal paths. However, the virtual data produced by the traffic data generator have been used in the proposed system for practical reasons.

CLIENT/SERVER

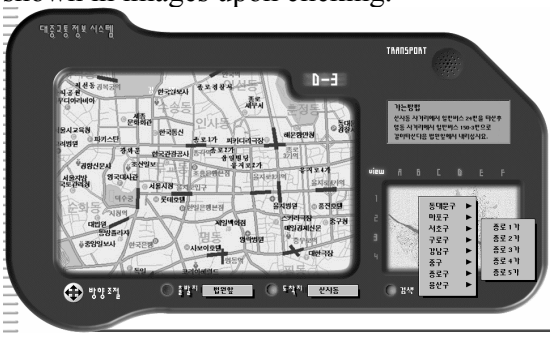
CLIENT

The client uses AWT(Abstract Window Toolkit) classes, the Windows-version of Java, to provide menus, buttons, dialog boxes, to represent traffic maps, to select the source/destination and to produce the output of search results.

For the services of the system to be more realistic, the actual running time from station to station must be reflected by considering road conditions, exact locations of stations and bus operation records, etc. The exact locations of bus stations can not be adequately shown on the map. Even if, for example, two stations have same name on the map, they may not have same locations. The actual distance between the two stations could be very long. There could be up to eight stations with same name on an intersection. Since the time or cost to transfer may be quite different according to the actual locations, such information must be reflected. The shortest time path is more important than the shortest distance path when the user chooses a bus/subway route,

since traffic congestion is the most critical problem in the metropolitan area. Therefore, actual running time based on bus operation records was taken into account in the system.

The client displays the result in text and graphics on the browser screen as shown in [Figure 4] after the user selects the source/destination by clicking on the "Find Path" button. The red bold line shows an optimal path from source to destination, and at the same time, the path is shown in the text box at the bottom of the screen. The transfer points are iconized and can be shown in images upon clicking.

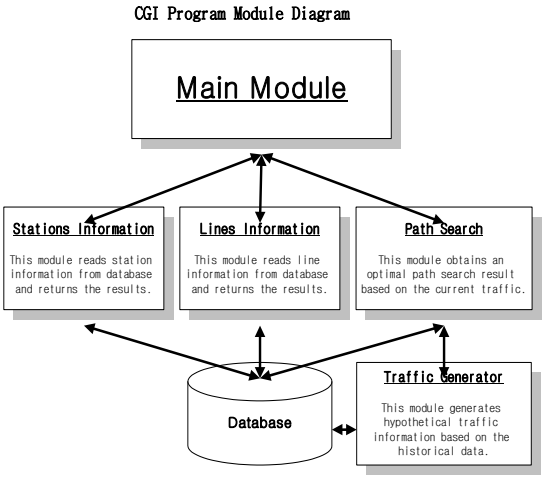


[Figure 4] Search Result displayed on Client Screen

SERVER

The CGI is used in the proposed public traffic information system in order to connect the client to the server. Following is the normal process how CGI works in the system.

1) The user calls a CGI program to issue a search request on the Web browser.
2) The Web browser contacts the Web server asking for permission to run the CGI program.
3) The Web server checks if the requester is allowed access to the CGI program.
4) The CGI program executed.
5) The resulting paths produced by the CGI program are returned in HTML format to the Web browser.
6) The Web browser displays the CGI output.



[Figure 5] Structure of the Server-side CGI module

Java Applets were used in the system in order to make the traffic information appear dynamic and live on the browser screen since data are no longer restricted to a page-by-page display with Java. For the system performance, we let the client-side Java Applet take care of the user interface only, and the server-side CGI program take over the other routines such as the path search system. The structure of the server-side CGI module is diagrammed in [Figure 5].

PATH SEARCH SYSTEM

The path problem has been a hot research topic for decades not only in academic fields but in real life applications. Many algorithms have been developed to find optimal solutions. Among them, A* algorithm has received more attention than any other algorithms. Although A* algorithm has proved to be successful, it quickly runs out of space even for problem instances of moderate size when searching for optimal solutions, since it requires exponential space. Furthermore, it produces the best solution only. It does not produce the alternate solutions sometimes needed in many applications.

To overcome such problems, we propose a modified A* algorithm which produces more than one optimal solution and utilizes a conditional pruning technique for time

A^* algorithm	Best First search + under estimation +
modified A^* algorithm	Best First search + proper estimation + conditional pruning + management of changing vehicles

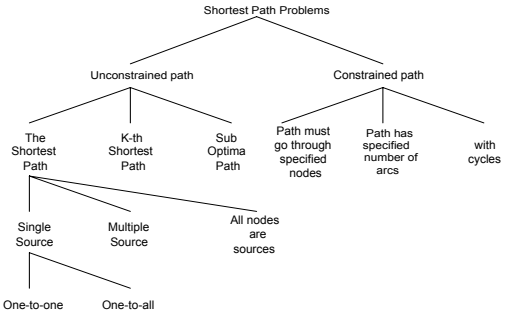
efficiency. The objective in the system is to produce more than one optimal path from source to destination in the cost value order.

Path problems can be classified into four categories: (8)

1) single source/single destination shortest paths
2) all pairs shortest paths
3) K-th shortest paths(first, second, ..)
4) shortest paths going through specified nodes

[Figure 6] shows the classification tree. The kind of algorithm applied in the proposed system falls under the single source/single destination K-th shortest path search algorithm.

Since (vehicle) transfers must be considered in the proposed system, a heuristic search technique, A^* algorithm, has been modified and applied in a way which has been proven time-efficient in certain problem domains (1)(6)(12).



[Figure 6] Classification of the optimal path search algorithms

MODIFIED A^* ALGORITHM

While A^* algorithm finds the least $g(n) + h^*(n)$ value first and uses a dynamic programming technique to search the shortest path, the modified A^* algorithm proposed in this paper uses a conditional pruning technique to consider various search conditions such as the number of routes to search or transfers. The differences are illustrated in [Figure 7].

[Figure 7] Comparison between the A^* and the modified A^* algorithm

HEURISTIC FUNCTIONS

Two heuristic functions are considered when time or distance are used as search criteria. When the search criteria is distance, the Euclidean Distance(UD) is used(10). In the case of time, UD × TD × R is used as the heuristic function, where TD stands for the average time/distance calculated from the database storing traffic cost values and R is a weight value between 0 and 1. In addition, transfer cost must be considered as well. Those search criteria and transfer costs can be given by users. [Figure 8] summarizes both heuristic functions.

	Time	Distance
g(n)	∑Time(min) + transfer cost × number of transfers	∑Distance(km) + transfer cost × number of transfers
H^*	UD×TD×R	UD

(UD : Euclidean Distance
TD : average time cost/distance in DB
R : real number between (0 ~ 1))

[Figure 8] Differences between the time cost and the distance cost

As shown in [Figure 8], when time is used as cost value, g(n) is calculated using the formula :

$\sum T + (cost(T) \times num(T))$, where $\sum T$ is summation of time cost of each edge from the start node to the n-th node (min), cost(T) is time cost of each transfer (min.), and num(T) is number of transfers. When the cost value is distance, $\sum D$ is used instead of $\sum T$. In that case, $\sum D$ is a summation of distance cost of each edge from the start node to the n-th node (km).

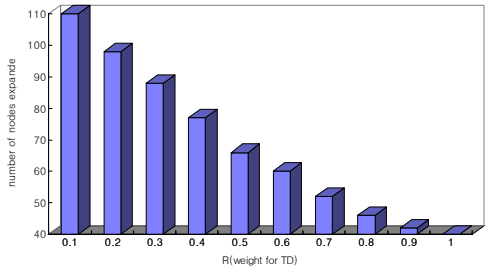
SIMULATION OF THE HEURISTIC FUNCTION

We examine the number of nodes expanded, execution time, and relative cost error for values of R(weight for TD), in case time is selected as the search criterion. For the simulation, we created a 300-node graph based on the data collected from real bus/subway routes. The cost between two adjacent nodes n and m is calculated using the following formula:

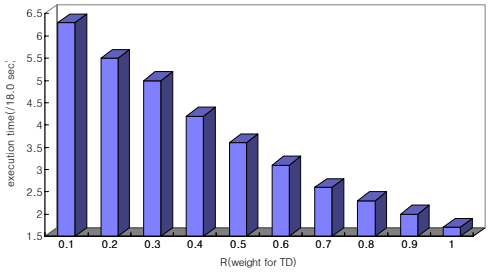
cost(n,m) = h^*(n,m) × TD × R, where h^*(n,m) is the Euclidean Distance between node n and m. The values of R are real numbers from 0 to 1. For a given value of R, we chose a pair of source/destination at random and executed 200 times. Then we averaged the number of nodes expanded, the execution time, and the relative cost error. The results are shown in [Figure 9, 10 and 11]. The relative cost error, E, was calculated as follows:

$$E = \frac{\text{cost of searched path - cost of optimal path}}{\text{cost of optimal path}} \times 100$$

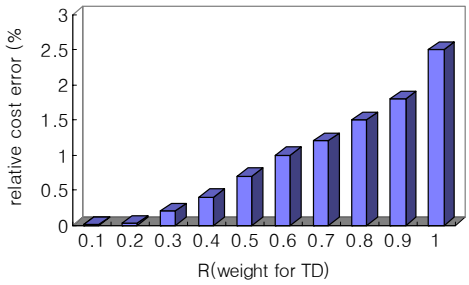
[Figure 9] illustrates a remarkable reduction in the average number of nodes expanded as R comes close to 1. Accordingly, the execution time, proportional to the number of nodes expanded, also reduces as shown in [Figure 10]. [Figure 11] shows the average relative cost error is under 3%.



[Figure 9] Average number of nodes expanded



[Figure 10] Average execution time



[Figure 11] Average relative cost error

TRAFFIC DATA STORAGE SYSTEM

The traffic data storage system maintains current traffic situations, stations/routes and their geographical data, and provides routing information for the path search. Since the user selects a bus/subway station by using a pointing device such as a mouse or touch screen on a Web browser, the coordinates of bus/subway stations are required. The raw data were obtained from bus companies or agencies concerned such as the Korea Subway Cooperation. Based on the raw data, the station/route information was stored using the Oracle database management system in image files to make a digital map by acquiring the absolute coordinates of each bus/subway station and route. The map represents Seoul in 7800×6800 resolution, as a whole. Since it is hard to manage such a wide area in a file, the map was subdivided into two hundred twenty one 600×400 resolution unit maps. Each unit map was stored in GIF format to be displayed on a Web browser.

TRAFFIC RAW-DATA MANAGEMENT SYSTEM

In order to supply more realistic public traffic information services, it is important to build accurate and up-to-data traffic raw data. By representing bus/subway stations and lines/roads in a network model and vehicles, people, time and distance as parameters to the path search algorithm based on the accurate raw data, correct

traffic information can be provided. Bus lines tend to change because they are usually operated by private companies. Such changes must be directly reflected in the path search algorithm by updating raw data on a Web browser. Therefore, the traffic raw-data management system was developed as a Java application which can be plugged into the Oracle server(9), and was designed to perform inserting, modifying and deleting operations platform-independently on a visual map. The visual management tool makes the operators work without necessarily knowing the actual coordinates of the objects, and reduces the cost of maintaining traffic raw data.

[Figure 12] shows the first screen the user sees when he/she starts the traffic raw-data management system. Once the system is executed, it asks the user for authorization. If permitted, it contacts the server and checks if there are any updated data on local sites. If any, the system reflects the updates to the central database. Then, the control is taken over to the user.

[Figure 12] Traffic Raw-Data Management System

[Figure 13] lists the menu items of the system and briefly explains their functions. When a new station is registered, three items are to be entered: name, location and type of station.

There are three kinds of station types: regular bus, seat bus and subway. Once three items are entered, the new station is appended to the station database after it is verified whether there is any ambiguity or conflict with other stations. If the location(x and y coordinate values) of the station is not known, it can be visually entered by clicking on the "Locating on Map" button and simply pointing to the position on the map.

When a station is to be deleted, the user selects the area on the map where the station is located, and all the adjacent stations in the area are shown from the database. Then, the user selects the station to be deleted. Care must be taken when deleting a station because it may be included in many routes. The system checks if there are line paths on which the station is situated. If any, the system informs the user and lets him/her cancel the operation.

When a new line is added, there are three items to be filled out: a line name, a means of transportation and a list of stations on the line. When a line is to be deleted, the list of existing lines is presented to the user to choose the line to be deleted. The "Line Retrieval" menu item shows the list of stations on a given line in text and optionally on the map.

Menu Item	Submenu Item	Function
File	Open	opens temporarily stored working file
	Store	when not connected to server, temporarily stores to local file
	Send to Server	sends data to server
	Exit	finishes work
Station Manage-ment	New Station	registers name, location of a new station
	Modify	modifies name, location of an existing station
	Delete	deletes a station
Line Manage-ment	New Line	registers name, path of a new line
	Modify	modifies name, location of an existing line
	Delete	deletes a line
Retrieval	Line Retrieval	shows the list of stations on a given line
	Station Retrieval	shows the location of a given line
	Path Retrieval	shows all the lines on a given path
Print	Print Line	prints the current line
	Print Station	prints the current station
Help		Assists in using the system

[Figure 13] Menu Items and their Functions in the Traffic Raw-Data

CONCLUSIONS AND FUTURE WORK

Our experience of developing a public transportation guidance system on the Internet was introduced as a means of solving the metropolitan area traffic congestion problem, not with a hardware approach like constructing new roads or subways but with a software approach. State-of-the-art techniques such as WWW, Java and CGI are utilized in the proposed system. It consists of clients and a server, a path search system finding optimal paths from source to destination, a traffic data storage system and a traffic raw-data management system. Since the system is developed using Java, which is object-oriented and platform-independent, it is easy to integrate with Internet systems running on heterogeneous machines as well as the other commercially available traffic information systems. It is also easy to alter the system within a relatively short period of time in case the user requirements change, and, consequently, easy to keep the system maintenance cost relatively low.

The current system has a performance limitation when multiple users request the service concurrently. Therefore, as a part of our future work, the Servlet provided by the Java Web server and threading will be utilized in order to reduce heavy load on the server. We are also planning to provide better quality geographic traffic information in vector images by constructing a GIS.

References

(1) A. L. Karl, H. Kaindl, "Bidirectional Best-First Search with Bounded Error: Summary of Results", IJCAI (International Joint Conference on Artificial Intelligence), 1993, pp. 217-223.

(2) Art Taylor, JDBC Developer's Resource, Informix Press, 1997

(3) A. V. Hof, S.Shaio, O. Starbuck, Hooked on Java, SUN Microsystems, 1996

(4) David Flanagan, Java in a Nutshell, O'REILLY, 1996

(5) Gray Cornell & Cay S. Horstmann, Core Java, Java Series, SunSoft, 1996

(6) J.B.H Kwa, BS : "An Admissible Bidirectional Staged Heuristic Search Algorithm", Artificial Intelligence 38(2), 1989, pp. 95-109.

(7) Lemay, Perkins, Teach yourself JAVA in 21 days, SAMS NET, 1996

(8) Judea Pearl, "Heuristics: Intelligent Search Strategy for Computer Solving", Addison-Wesley Publishing Company, 1984, pp.64-99.

(9) ORACLE, Oracle and Internet, ORACLE White Paper, 1996

(10) Hart, Nilson and Raphael, "A Formal Basis For The Heuristic Determination of Minimum Cost Paths", IEEE Transaction on SSC-

4(2), pp. 100-107.
(11) Shishir Gundavaram, CGI Programming on the World Wide Web, O'REILLY & ASSOCIATES, 1996
(12) I.Pohl, "First Result on the Effect of Error in Heuristic Search", In B. Meltzer and D. Michile, editors, Machine Intelligence 5, 1970, pp. 219-236.
(13) T. Ritchey, Java, New Rider, 1995