# 웹기반 대중교통 안내시스템 설계 및 구현

## (Design and Implementation of a Web-based Public Transportation Guidance System)

배 수 강 † 이 승 룡 †† 최 대 순 ††† 정 태 충 †† 승 현 우 ††††
(Sookang Bae) (Seungyong Lee) (Daesoon Choi) (Taechoong Jung) (Hyunwoo Seung)

**요 약** 본 논문에서는 웹(World Wide Web)에서 사용자가 손쉽고 편리하게 이용할 수 있는 멀티미디어 대중교통 안내시스템 개발 경험을 소개한다. 개발된 시스템은 클라이언트와 서버 시스템, 경로탐색 시스템, 교통정보 저장 시스템, 노선 및 정류장 관리 시스템으로 구성되어 있다. 클라이언트에서 작동되는 사용자 인터페이스는 직관적으로 이해가 쉽고, 사용이 편리하며 인터액티브한 멀티미디어 대중 교통안내 서비스를 제공한다. 서버 시스템은 교통정보 수집 시스템으로부터 입력되는 데이타와, 경로탐색 시스템, 교통정보 저장 시스템과 연동되어 클라이언트의 요구사항을 처리하고 그 결과를 사용자에게 돌려준다. 수정된 A* 알고리즘을 이용하는 경로탐색 시스템은 최적경로를 탐색하며, 교통정보 저장 시스템은 현재 교통상황, 정류장, 노선, 지도 등의 정보를 저장한다. 노선 및 정류장 관리시스템은 시스템 관리자가 노선 또는 정류장 관리를 서버 화면의 지도상에서 효율적으로 수행할 수 있는 도구이다. 본 논문에서 다루는 대중교통 안내시스템은 Java로 구현하였기 때문에 확장과 이식이 용이하며, 시스템 유지보수 비용이 적게 드는 장점을 가지고 있다. 그리고, 웹 브라우저가 동작되는 환경에서는 어디서나 쉽게 접근이 가능하며 향후 구축될 Intelligent Transportation Systems(ITS)의 한 모듈로써 바로 작동될 수 있을 뿐만 아니라, 현재 인터넷상에서 제공되는 다양한 서비스와도 연동이 가능하다.

***Abstract*** This paper introduces our experience for developing a public transportation guidance system, which facilitates the World-Wide Web(WWW) to provide users with easier access and use. The proposed system is composed of four subsystems: client/server system, path search system, traffic data storage system, and traffic raw-data management system. The user interface in clients utilizes Java to furnish users with multimedia data accessibility and interactivity. The server processes clients' requests based on the traffic data coming from remote sensing devices and interacts with the path search system and traffic data storage system to provide users with the results. The path search system, which uses a modified A* algorithm, produces optimal solutions based on dynamic traffic data. The traffic data storage system stores the current traffic information together with the geographical information about the bus/subway routes. The traffic raw-data management system is a graphical user interface which enables the system manager to handle the traffic information easily on the map in the terminal screen. The system has considerable benefits such as portability, scalability, and flexibility since it is implemented using Java. Also, it can be extended to an integrated Intelligent Transportation Systems(ITS) which includes a variety of information on the Internet as well as traffic information.

## 1. 서 론

· 본 연구는 한국건설기술연구원의 '96건설교통기술연구개발사업 KICT#96-0147에서 지원 받았음.

† 학생회원 : 경희대학교 전자계산공학과
bsk@oslab.kyunghee.ac.kr
†† 종신회원 : 경희대학교 전자계산공학과 교수
sylee@oslab.kyunghee ac kr
tcchung@nms kyunghee ac kr
††† 비 회 원 한국건설기술연구원 연구원
dschoi@kict re kr
†††† 정 회 원 : 서울여자대학교 컴퓨터학과 교수
hwseung@cs swu.ac.kr
논문접수 : 1998년 4월 29일
심사완료 : 1999년 3월 30일

최근 우리나라에서도 자동차 숫자가 폭발적으로 증가하여 1997년에 드디어 천만대를 돌파하였다. 이러한 차량 증가의 대부분은 개인승용차이며 이와 같은 현상이 대도시 교통난을 더욱 어렵게 하고 있음은 주지의 사실이다. 따라서 각국에서는 천문학적인 비용이 소용되는 도로, 철도, 지하철을 새로 건설하는 대신에 기존의 대중교통 시스템의 서비스 및 운영관리 체계에 첨단 컴퓨터, 전자, 제어, 통신 등의 기술을 적용하여 사용자에게는 시간과 장소에 구애됨이 없이 다양하고, 편리하며, 신속 정확한 대중교통 정보서비스를 제공하는 첨단 대중교통 시스템(APTS: Advanced Public Transportation Systems)의 개발을 서두르고 있다[1][2][3][4].

APTS는 대중교통 시스템을 더 효율적이고 신뢰할 수 있게 만들어 교통흐름을 원활하게 해주고, 체증을 피할 수 있는 대체경로를 제시해주며, 대중교통 이용자들에게는 다양하고 생생한 현장 교통정보 서비스를 제공해 줄 수 있다. 이는 결과적으로 교통인구를 대중교통 수단으로 유도하여 도시 교통난을 가중시키고 있는 개인 승용차의 이용을 억제함으로써 에너지의 낭비를 막고, 시간을 절약할 수 있어 환경 보호와 경제적인 측면에서 사회에 많은 기여를 할 수 있다.

APTS는 ITS를 구성하는 시스템의 일부이기 때문에 ITS의 아키텍처에 영향을 받게 되며, 그것의 물리적 구조는 일반적으로 교통정보 수집 시스템, 중앙 관제시스템, 통신시스템, 단말 시스템으로 구성되어 있다(그림1). 교통정보 수집시스템은 비콘 또는 각종 차량 검지기들로부터 도로교통, 기상, 사고 정보 등을 수집하는 체계로 구성되어 있다. 중앙관제 시스템은 사용자의 요구를 처리하고 전체 시스템을 제어하며, GIS(Geographical Information Systems), GPS (Global Positioning System), AVL(Automatic Vehicle Location) 시스템과 연동된다[5][6]. 통신 시스템은 유무선망을 사용하여 교통정보 수집 시스템으로부터 획득된 데이타를 중앙 관제시스템에 전송해주며, 그 처리 결과를 다시 사용자에게 전달하는 역할을 한다. 단말 시스템은 사용자 또는 시스템 관리자가 접하는 최종 인터페이스이다.

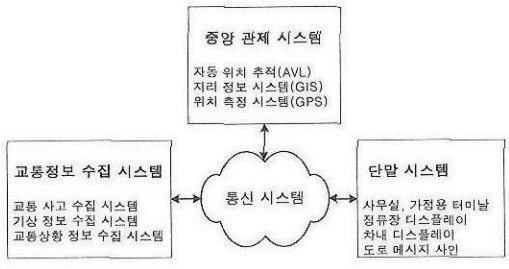



그림 1 일반적인 APTS의 물리적 구조

본 논문에서는 APTS의 일부로써 사용자가 손쉽고 편리하게 이용할 수 있는 웹기반 대중교통 안내시스템 개발 경험을 소개한다. 개발된 시스템은 현장의 실시간 교통정보와, 현재의 교통상황을 고려하여 출발지에서 목적지까지의 최적경로를 제공해주고 효율적이고 편리한 대중교통 노선관리 서비스도 제공해준다. 시스템의 물리적 구조는 크게 클라이언트와 서버시스템, 경로탐색 시스템, 교통정보 저장 및 관리시스템으로 구성되며, 이들은 인터넷에 연결되어 있다(그림 2).

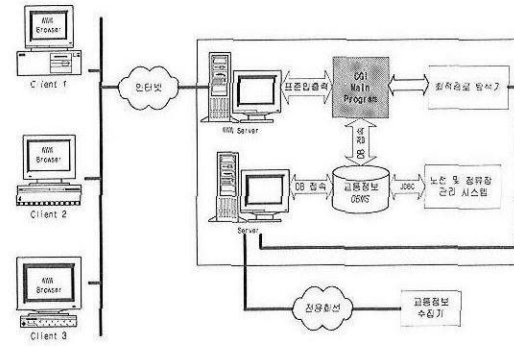



그림 2 제안한 시스템 구조

클라이언트는 서버로부터 HTML(Hypertext Markup Language)과 Java 애플릿을 다운로드 받아 사용자의 요구사항을 입출력하는 인터페이스를 제공하며, 플랫폼에 관계없이 Java를 지원할 수 있는 Netscape나 Explore와 같은 웹 브라우저로 구성되어 있다. 서버는 교통 정보 수집기, 교통 정보 저장시스템, 경로 탐색기를 연동하여 사용자 요구사항을 처리하며, 웹 서버는 CGI(Common Gateway Interface)와 JDBC(Java Database Connectivity)를 이용하여 외부 실행 모듈과 연결되어 있다. 클라이언트와 서버간의 통신은 TCP/IP 위에서 동작하는 웹의 기본 프로토콜인 HTTP (Hypertext Transfer Protocol)를 통하여 이루어진다. 모듈간의 동작 순서는 그림 3에 나타난 바와 같이 클라이언트가 인터넷을 통하여 서버에게 탐색을 요구하면 서버는 CGI를 통해서 저장시스템으로 교통정보를 요청하게 되고, 이 정보를 바탕으로 경로탐색 시스템은 실시간 탐색을 수행하여 그 결과를 클라이언트에게 돌려주게 된다(그림 3).

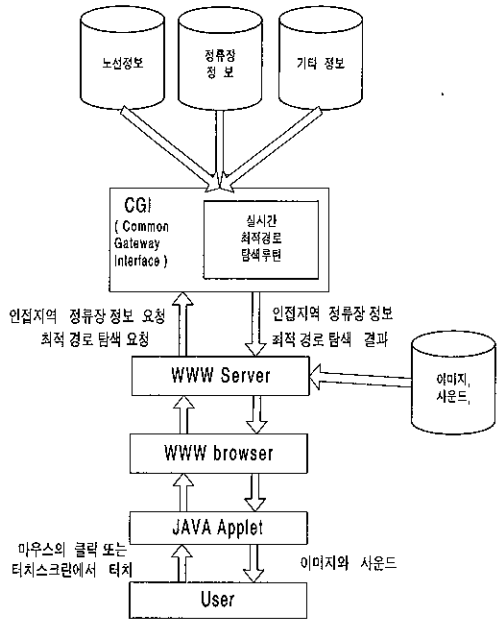



그림 3 시스템의 각 모듈간 데이타 흐름도

경로탐색은 A* 알고리즘을 수정하여 사용하는데, 이는 최적경로 탐색 시 환승조건, 탐색노선 수와 같은 현재의 교통상황을 반영한다. 교통정보 저장 시스템은 현재의 교통상황, 서울시 지하철 및 버스 정류장, 노선, 그리고 편집된 지도정보를 저장한다. 노선 및 정류장 관리 시스템은 노선이나 정류장의 등록, 수정, 변경 및 삭제 시 관리자가 서버 화면상의 지도에서 용이하게 관리할 수 있는 환경을 제공한다. 교통정보 수집기로부터 입력되는 동적인 실시간 트래픽 raw-data는 본 논문의 연구범위 상 가상 트래픽 raw-data로 대체 사용하는데 이는 제안된 시스템의 트래픽 생성기로부터 발생된다. 이 가상 데이타는 서울시 지방경찰청과 기업의 협조를 얻어 루프 디텍터, 교통 영상 검지기, CCTV, 각종 센서, 버스에 부착된 비콘과 GPS 장치로부터 입력되는 동적인 실제 교통 raw-data를 바탕으로 하고, 노선별, 요일별, 시간대별, 계절별, 공사현황, 사고발생 분포, 교통량, 노선폭, 노면상태, 환승정보, 정류장 대기시간 등을 고려하여 실제 현장의 교통상황과 유사한 형태로 발생된다. 제안된 시스템은 객체지향 언어이자 플랫폼인 Java를 사용하였기 때문에 확장과 타 시스템과의 연동이 용이하며, 단기간 내에 시스템을 개발할 수 있고, 유지보수 비용이 적게 드는 장점을 가지고 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 제안된 시스템과 유사한 다른 시스템에 대한 특성을 살펴보고, 3 장에서는 클라이언트와 서버의 역할을 설명한다. 4 장에서는 경로 탐색 시스템에 대해 기술한 다음, 5 장에서는 교통정보 저장시스템과 노선 및 정류장 관리시스템에 대하여 논의하고, 6 장에서 결론을 맺는다.

## 2. 관련연구

본 논문에서 제안된 시스템과 관련된 연구로는 국내의 경우 도로교통 안전협회, 서울시, 서울시 지방경찰청에서 운영중인 수도권 대중교통 정보시스템과 지하철과 시내·외 버스 노선안내가 있다[7][30][31]. 그러나 이들 시스템 구조는 확장성, 이기종 시스템과의 연동에 대한 고려가 미진하며 사용자가 수도권 지역에 대한 지리적인 정보를 어느 정도 파악하고 있어야 이용이 가능하다.

Infobank[8]에서는 서울시 종로 일대를 대상으로 버스 도착정보 제공 시범서비스를 수행하였고, 과천지역을 대상으로 버스 노선안내와 환승정보를 제공하고 있다. 이는 비콘과 GPS를 버스에 부착하여 정류장에서 기다리는 승객들에게 버스 도착시간과 위치를 예고해준다. 그러나 이 시스템은 제한된 서비스를 제공하며, 경로탐색 모듈이 없고, Java로 시스템을 구축하지 않은 점이 본 논문에서 제안된 시스템과 다르다.

한국항공대학[9]에서 개발한 교통정보 안내시스템은 대중교통 안내 시스템이 아닌 차량항법시스템(Car Navigation System)이다. 이는 GIS와 GPS를 이용하여 자동차의 주행중 또는 일반 사무실이나 가정에서 미리 주행 경로를 확보하기 위한 수단으로써 이용될 수 있다. 그러나 시스템 구현에 사용된 언어가 C++이기 때문에 특정 플랫폼에 대해서만 이용 가능하다는 제약이 있다.

한편, 국외의 관련연구로는 미국의 FTA(Federal Transit Administration)에서 ITS 사업의 일환으로서 진행되는 APTS 프로그램이 있다. 이 프로그램의 목적은 첫째, 도로에서 사용자에 대한 서비스 질을 향상시키고 둘째, 시스템의 생산성과 작업의 만족도를 증진시키는 것이며, 셋째는 전체 공동 사회의 목적을 위해 대중교통 시스템을 강화시키고, 넷째 APTS 혁신 기술과 관련된 전문지식을 확대하는 것이다. APTS 서비스를 위해 제안된 주요 기술들은 차량관리(Fleet Management), 승객 정보(Traveler Information), 전자 요금 지불(EFP: Electronic Fare Payment)이며[10], 이러한 시스템의 예로는 신시내티의 SORTA(Southwest Ohio Regional Transit Authority)[11], 호놀루루의 The-BUS[12], 워싱톤 대학의 BusView[13] 등이 있다.

유럽 연합은 전체 유럽의 대중교통 시스템을 단일 시

스템으로 묶는 EUROBUS 프로젝트를 수행하고 있다[14][15]. EUROBUS는 중부 유럽과 동부 유럽의 7개 국가 (불가리아, 체코, 헝가리, 폴란드, 루마니아, 슬로바키아, 슬로베니아)의 버스를 중심으로 한 대중교통 체계를 통합하기 위한 대규모 프로젝트이다. EUROBUS는 차량 배차 및 제어 시스템(VSCS: Vehicle Scheduling and Control Systems), 첨단 여행자 정보 시스템(ATIS: Advanced Traveller Information System), 노선 관리 시스템(NMS: Node Management Systems), 대중교통 요금 관리 (Public Transport and Car Debiting Systems) 시스템 등으로 구성되어 있다. EUROBUS 프로젝트는 지역적으로 광범위하고 버스, 전차, 전철 등 이질적인 대중교통 수단을 자동적, 효율적으로 관리할 수 있고 양질의 교통정보 서비스를 사용자에게 전달할 수 있는 통합 시스템 구성을 목표로 하고 있다. 이를 위하여 프로젝트 기능분야를 교통 정책 입안을 위한 대중교통 전략 수립, 대중교통 전술 수립과 대중교통 운영 분야, 향후 계획을 위한 통계 분야의 4단계로 나누고 이들에 대하여 각각 마케팅, 교통 수단, 승객 정보, 장비 관리, 요금 징수, 인원 관리, 계정 부분의 7가지 상세 분야에 걸쳐 개발을 추진하고 있다[4]. EUROBUS 이외도 독일의 COMFORT(Cooperative Management for Urban and Regional Transport)[16], 영국 버밍검의 QUARTET(QUadrilateral Advanced Research on Telematics for Environment and Transport)[17], 런던의 LLAMD(London-Lyon-Amsterdam-Munich-Dublin)[16], 프랑스의 Minitel[18], RATP(Regie Autonome des Transports Parisiens)[19], SYMPHONIE [16]등의 프로젝트가 진행중이다.

일본은 1994년 VERTIS(VEhicle Road and Traffic Intelligence Society)를 구성하였다[20]. 이들의 목표는 교통사고의 감소, 교통 혼잡의 제거, 연료 사용의 감소, 환경 오염 방지 등이며, 제공하는 서비스는 여행, 도로 교통, 운전 정보, 주차 관리, 운송 효율제고, 교통 수요 관리, 교통 제어 및 관리, 전자 통행료 징수, 긴급상황 관리, 재난 방지, 차량 및 운전자 상태 경고, 제어와 충돌 방지 등이다.

## 3. 클라이언트/서버 시스템

### 3.1 클라이언트 시스템

본 논문에서 제안된 시스템은 물리적으로 클라이언트와 서버 시스템으로 구성되어 있다. 클라이언트에서는 Java의 윈도우 시스템인 AWT(Abstract Windows Toolkit) 클래스[21][22]를 이용하여 인터액티브한 사

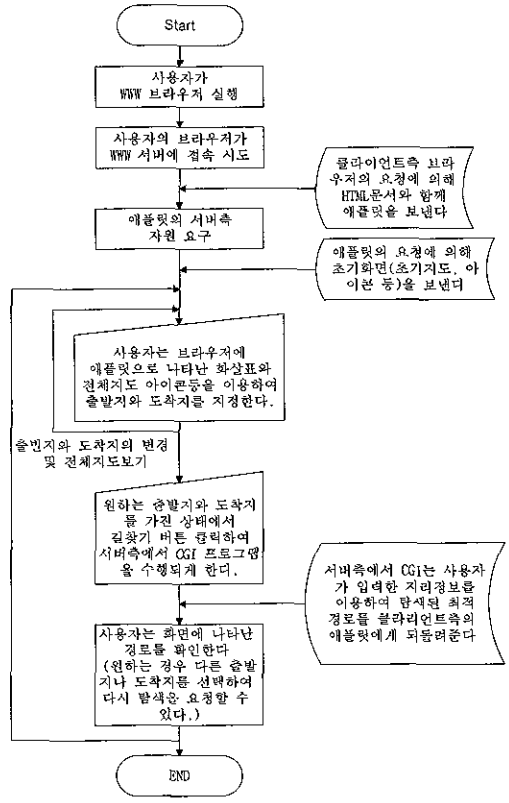



그림 4 사용자 인터페이스 시나리오

용자 인터페이스(예를 들면 메뉴, 버튼, 다이얼로그 박스 등)를 제공해주며, 지도를 표현하거나 이를 스크롤 해주는 작업과 최적경로의 탐색결과를 출력한다. 그리고 사용자가 원하는 지점의 현재 교통상황에 대한 정보를 동영상 또는 음성으로 제공해 준다.

사용자 인터페이스는 직관적으로 이해가 쉽고 단순한 조작으로 원하는 결과를 얻을 수 있도록 설계하였다. 그림 4는 제안된 시스템의 사용자 인터페이스 작동 과정을 보여 준다.

그림 5는 클라이언트에서 제공하는 '길찾기' 버튼을 눌러 선택한 출발지에서 도착지까지의 최적 경로를 탐색한 결과를 보여주는 화면이다.

사용자 인터페이스의 각 주요 부분에 대한 설명은 그림 6과 같다.

최적의 경로는 화면 지도상에 적색 선과 화면 우측의 글상자에 문장으로 나타나며, 환승역을 아이콘화 하여간단한 그림으로 보여준다. 그리고 버스의 경우 최단거리보다는 최소시간이 노선 선택 시 더 중요한 요소이므

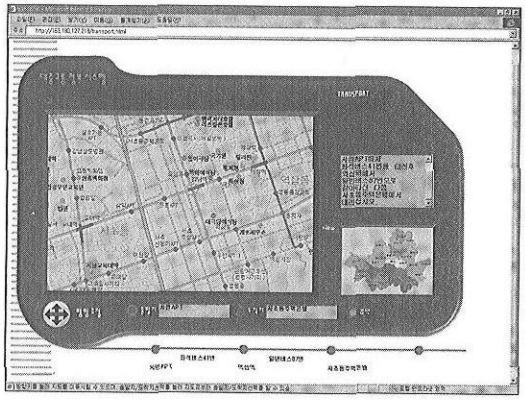


그림 5 최적경로 탐색 결과화면

로 전용버스 차선과 도로의 정체상황, 정류장 대기시간, 환승 소요시간, 배차시간, 운행시간 등을 고려한 가상데이타를 경로탐색 시 파라미터로 반영하였다.

### 3.2 서버 시스템

서버시스템에서는 지도, 정류장 및 노선 정보들을 이용하여 경로탐색을 수행하고, 클라이언트의 요구사항을 처리한다. 그림 7에서 CGI 프로그램의 메인 모듈은 클

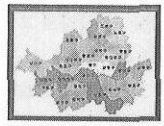
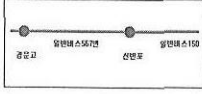
| | |
|---|---|
| [최적 경로는...] 경문고에서 일반버스567번을 타신후 산반포에서 일반버스150-5번으로 갈아타신 다음 서초경찰서에서 내리십시오. | 안내문을 문자를 통하여 사용자에게 제시하는 부분이다. 사용자가 지도상에서 정확한 출발지와 도착지를 선택할 때 이용된다. |
|  | 서울 전역을 각 구별로 단순하게 나타내주어 사용자가 원하는 지역으로 쉽게 이동할 수 있다. |
|  | 사용자가 마우스 또는 좁은 포인팅 지역 내에서도 충분히 지도를 스크롤 할 수 있는 기능 제공 |
|  | 사용자가 출발지를 선택할 때 사용되며, 선택된 출발지를 화면에 문자로서 보여준다. |
|  | 사용자가 목적지를 선택할 때 사용되며, 출발지의 경우와 유사한 기능을 가지고 있다. |
|  | 출발지와 도착지를 선택한 후 경로를 탐색할 때 사용되는 버튼이다. |
|  | 탐색된 경로를 환승하는 정류장, 이용할 대중교통수단의 이름과 번호를 그림으로 제시 해준다. |

그림 6 컴포넌트별 사용자 인터페이스

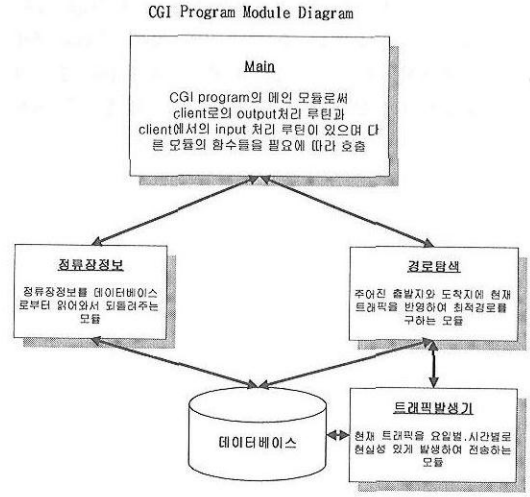



그림 7 CGI 모듈의 전체 구조

라이언트 측의 Java 애플릿에서 보낸 파라미터를 이용하여 사용자의 요구에 해당하는 각 서브 모듈을 호출하게 되는데, 메인 모듈이 호출할 수 있는 서브 모듈로는 정류장 정보 모듈과 경로 탐색 모듈이 있다. 각각에 대한 입력 모드와, 입력 파라미터, 출력 값은 표 1과 같다.

표 1 모드에 따른 입출력 값

| 구분 / 모듈 | 입력 파라미터 | | | | | 출력 값 |
|---|---|---|---|---|---|---|
| | 모드 | 파라미터 1 | 파라미터 2 | 파라미터 3 | 파라미터 4 | |
| 정류장 정보 | getvn | X 좌표값 | Y 좌표값 | X 범위 | Y 범위 | 선택된 지점에서 범위내에 존재하는 모든 정류장 이름 |
| 경로 탐색 | search | 출발지 정류장 이름 | 도착지 정류장 이름 | 없음 | 없음 | 출발지에서 도착지까지 정류장 좌표값 |

클라이언트 측에서 정류장 정보를 요구한 경우는 데이타베이스에 저장되어 있는 정류장 정보를 읽어와서 CGI 메인 프로그램으로 결과 값을 반환한다. 경로 탐색을 요구한 경우에 경로 탐색 모듈은 데이타베이스에서 시간과 거리에 따른 해당 구간의 정보를 읽어오며, 또한 경로 탐색은 현재 어떤 구간의 교통량 현황에 대한 정보를 트래픽 생생기로부터 얻어와 수행된 다음 클라이언트로 결과 값을 보낸다.

클라이언트와 서버 사이의 데이타 교환방법은 소켓을 이용할 수 있는데 이는 Java 애플릿에서 직접 CGI 프로그램을 호출하여 그 결과를 Java 애플릿에서 받아서

처리하는 방식이다. 이 경우 Java 애플릿과 CGI 프로그램이 일대일로 동작하므로 안정적이다. 그러나 Java 애플릿에서 CGI 프로그램을 호출 시 매번 CGI 프로그램이 실행되었다가 결과를 반환하고 종료되기 때문에 오버헤드가 크다. 따라서, 본 논문에서 제안된 시스템에서는 웹 서버를 자바 웹 서버로 하여 서블렛(servlet)으로 CGI를 구성하였다. 이경우 일단 최초의 서블렛이 초기화되면 이후의 사용자에 대해서는 쓰레드로 수행된다. 따라서 초기화에 대한 오버헤드가 없고, 사용자 수가 증가하더라도 시스템 성능이 비례적으로 감소하지 않게 된다. 본 연구 결과로는 서블렛이 C 언어를 사용하는 경우보다 보다 단일 사용자 환경에서 2회 이상 수행 시 실행속도 면에서 평균 약 20%의 성능향상을 보여주었다.

## 4. 경로탐색 시스템

경로탐색 문제는 학문적 분야뿐 아니라 실생활 분야에서도 많은 사람들의 연구 대상이 되어 왔다. 최적의 경로를 보다 빠른 시간 내에 제시하기 위해 그 동안 많은 알고리즘들이 적용되었으며 그 중에서도 A*알고리즘[23]은 다른 어떤 알고리즘 보다 많은 문제에 적용되어 왔다. 그러나 A*알고리즘이 우수하지만 실제 문제 상에서는 노드로 대변되는 데이타가 무수히 많아 이를 위해서는 데이타를 충분히 수용할 수 있는 메모리가 요구될 뿐만 아니라 최적의 경로 이외의 다른 대체 경로를 제시하지 못한다는 단점을 갖고 있다.

위에서 언급한 이러한 두 가지 문제점을 해결하기 위해 본 논문에서는 최적경로 1개만을 찾아주는 A* 알고리즘을 변형하여 최적경로 여러 개를 나열해 주는 알고리즘으로 변형하고, 시간적 효율성을 위해 휴리스틱 값으로 사용될 예측 값을 동적으로 구하는 방식을 도입하였다.

### 4.1 경로탐색 알고리즘

본 논문에서 경로탐색은 대중교통 노선 망에서 출발지에서 목적지까지 최적의 경로를 빠른 시간 내에 cost value의 순서로 여러 개의 대체 경로를 제시하는 것을 목표로 한다.

최적경로탐색 문제는 다음과 같은 유형으로 분류될 수 있다[24]. a) 그래프에서 두 노드들 사이의 최적경로를 구하는 문제, b) 그래프에서의 모든 노드들의 쌍에 대한 최적경로를 구하는 문제, c) 최적경로뿐만 아니고, 두 번째, 세 번째 등의 대체 최적경로를 구하는 문제, d) 특정 노드를 경유하였을 때의 최적경로를 구하는 문제이다. 최적경로탐색문제에 있어서, 경험적 탐색방법을 사용하면 문제를 풀기 위해 요구되는 시간이 지수 함수에 비례하지 않고, 그보다 적은 시간 내에 세일즈맨의 방문 문제와 같이 어려운 문제를 효율적으로 풀 수 있다[25][26].

본 논문에서 알고리즘 구현기준은 한 개 이상의 경로를 탐색할 수 있는 $K$-th최적경로 문제이며, 단일소스(single source) 유형의 일대일(one-to-one) 문제이다. 또한 대중교통수단의 환승조건이 고려되기 때문에 본 논문에서는 A* 알고리즘을 변형하여 구현하였다.

### 4.2 수정된 A* 알고리즘

A* 알고리즘은 최적경로 하나만을 탐색하기 위하여 동적 프로그래밍 기법을 사용하지만 본 논문에서 제안한 수정된 알고리즘에서는 탐색노선 수와 환승 패턴을 만족시켜야 한다는 제약조건을 이용한 조건부 가지치기(conditional prunning) 기법을 사용하며, 환승 문제를 고려하기 위한 환승비용 개념을 알고리즘에 추가되었다. 또한 탐색기준을 시간으로 선택했을 때의 예측 값에 대한 효율을 높이기 위하여 DB에서의 시간/거리에 대한 평균값의 적절한 가중치를 두도록 하였다. A* 알고리즘과 수정된 A* 알고리즘과의 차이점은 그림 8과 같다.

| | |
|---|---|
| A* 알고리즘 | Best First search + under estimation + dynamic programming |
| 수정된 A* 알고리즘 | Best First search + proper estimation + conditional prunning + management of changing vehicles |

그림 8 A* 와 수정된 A* 알고리즘과의 비교

### 4.3 휴리스틱 함수

제안된 시스템에 사용하는 휴리스틱 함수는 탐색조건이 시간과 거리의 두 개 요소로 구성되어 있으며, 탐색조건이 거리일 경우 예측값으로는 Euclidean Distance-(UD)를 사용한다[27][28]. 탐색조건이 시간일 때 문제가 되는 것은 UD를 얼마만큼의 시간비용으로 환산해 주어야 하는 것인데, 본 논문에서는 DB에서의 시간/거리에 대한 평균값인 TD에 대한 가중치 R을 곱해서 사용하도록 하였다. 즉, 모든 도로의 평균 주행 속도를 기준으로 남은 거리에 대해 얼마만큼의 최소한 시간이 걸릴 것인가를 반영할 것인가에 대한 비율 R을 도입한다. 따라서 본 논문에서 고려되는 비용의 기준은 두가지 즉, 시간, 거리가 있고, 고려사항으로는 갈아타는 비용이 주어져 있다 비용의 기준과 갈아타는 비용은 사용자가 입력할 수 있다. 그러한 상황에서 $f^*(n)$, $g(n)$, $h^*(n)$는 다음 그림 8과 같이 계산된다. 여기서 노드 n을 지나서

| | 시 간 | 거 리 |
|---|---|---|
| g(n) | Σ시간(분) + 환승비용 *갈아타는 횟수 | Σ거리(km) + 환승비용 *갈아타는 횟수 |
| $h^*$ | UD * TD * R | UD |

(UD : Euclidian Distance
TD : DB 에서의 평균 cost/distance
R : (0 ~ 1) 사이의 실수 )

그림 9 시간비용과 거리비용의 차이

목적지까지 갈 때 걸리는 최대 예측비용 $f^*(n)$은 노드 n까지 올 때 이미 치른 비용 g(n)과 노드 n에서 목적지 노드까지 치루는 최소 비용 예측치 $h^*(n)$의 합이다.

그림 9에서 보는 바와 같이 시간을 비용의 기준으로 계산하는 경우에 g(n)은 출발 노드에서 노드 n까지의 구간별 걸린 시간에 갈아탈 때의 비용을 시간(분)으로 계산한 값을 더한 후에 이에 갈아타는 횟수를 곱해서 얻어진다. 평가함수인 $h^*$는 현재 위치와 목적지 간의 ED와 가중치로 사용되는 R값은 탐색시간과 해의 질에 상반관계가 있으므로 사용자에게 입력을 받도록 하였다. R값이 1에 가까울수록 탐색시간은 적게 걸리나 해가 정확하지 못하고, R값이 0에 가까울수록 해는 정확해지나 탐색시간은 많이 걸린다.

또한 거리를 비용의 기준으로 계산하는 경우의 g(n)은 소스 노드에서 노드 n까지의 구간별 거리에 갈아탈 때의 비용을 시간(분)으로 계산한 값을 더한 후에 갈아타는 횟수를 곱해서 얻어진다. 이 때의 평가함수인 h*(n) 역시 현재 위치와 목적지 간의 ED를 사용한다.

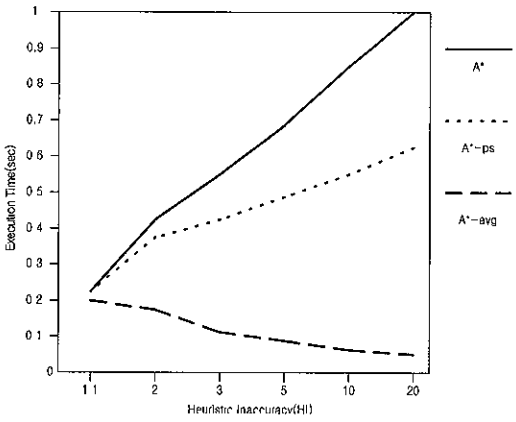



그림 10 알고리즘들의 평균수행시간

본 논문에서 제안하는 또 다른 아이디어는 path-sensitive heuristic이다. 시내 도로의 비용(cost)값이 균일하지 않으므로 $h^*(n)$을 구할 때 n이 위치한 주변 비용 값들을 반영하여 $h^*(n)$을 구하자는 개념이다[29]. 그 개념을 활용한 알고리즘 A*-PS와 평균적인 비용을 활용한 알고리즘 A*-avg를 비교한다. 휴리스틱의 부정확도 즉, 실제와 얼마나 차이가 발생하는가의 정도에 대해 각 알고리즘의 평균 수행 시간 및 상대 오차를 구해 보면 위의 그래프와 같다.

그림 10과 11에서 보듯이 A*-ps는 A*-avg에 비해 수행시간이 많이 드는 대신(A* 보다는 적게 걸리지만) 오차가 작고 A*-avg는 그 반대이다. 따라서 사용환경에 따라 각 알고리즘이 선택될 수 있다.

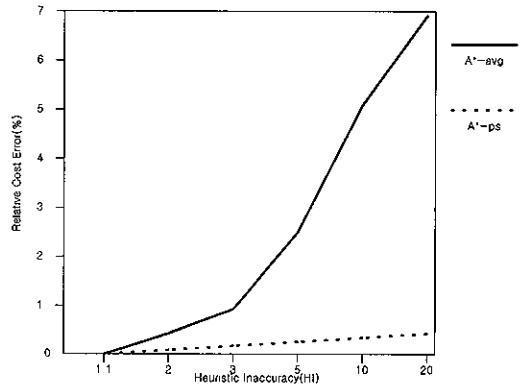



그림 11 알고리즘들의 찾은 경로에 대한 상대오차

### 4.4 휴리스틱 함수의 시뮬레이션 및 고찰

본 논문에서는 두 가지 시뮬레이션을 수행하였는데, 하나는 R의 영향이고 또 다른 하나는 휴리스틱 함수 $h^*(n)$을 path sensitive 할 때의 효과이다. 먼저 R의 영향을 살펴본다. 탐색기준으로 시간을 선택했을 때 UD에 대한 환산비용으로 TD를 계산할 때 적절한 R값의 선택을 위하여, R값에 따른 평균 확장 노드 수 및 탐색시간, 평균 상대오차를 알아본다. 탐색에 사용한 휴리스틱은 두 노드 사이의 직선거리인 ED에 DB의 평균 시간/거리를 곱한 값이며 시뮬레이션 상황은 다음과 같다.

첫째 노선망 그래프는 실제 노선정보를 입력하여 노드가 300개인 그래프를 생성하였다. 노선망 그래프에서 이웃하는 두 노드사이의 비용은 cost(n,m) = $h^*(n,m)$ * TD * R로 정하였다. 여기서 $h^*(n,m)$은 n, m 사이의 ED이며 TD는 노선망인 정류장 DB의 평균 시간/거리

값을 사용하였다. R은 0 에서 1 사이의 실수로 TD에 대한 가중치로 사용하였다. 노선망 그래프에서 임의의 현재위치와 목적지를 선정하여 입력파라미터인 R값을 변경하면서 200회씩 시행하였다. 이 때 출력 파라미터는 확장된 노드의 개수, 수행에 걸린 시간, 탐색된 경로의 비용에 대한 상대오차 E 등이 있다. 이 때 E 값은 다음 식으로서 계산할 수 있고, 이에 따른 시뮬레이션의 결과는 그림 12, 그림 13에 나타나 있다.

$$E = \frac{\text{탐색된경로의비용} - \text{최적경로의비용}}{\text{최적경로의비용}} * 100(\%)$$

시간거리의 평균값 TD에 대한 가중치 R을 1로 주었을 경우, 탐색 트리의 가지치기가 $h^*(n,m)$만을 사용할 때보다 많아지므로 그림 12와 같이 탐색의 수행시간은 줄어든다. 반면에, 표준 $A^*$와의 탐색 비용을 비교하면, 가중치 R을 높게 준 경우 차이가 많음을 그림 13에서 볼 수 있다. 그러나 그림 13과 같이 표준 $A^*$가 지도상의 거리만을 고려한 것이므로 대중교통 노선의 탐색에서 최적이라고 볼 수 없다. 따라서 R 값의 경험적 설정을 통해 최적값을 찾아내면 된다.

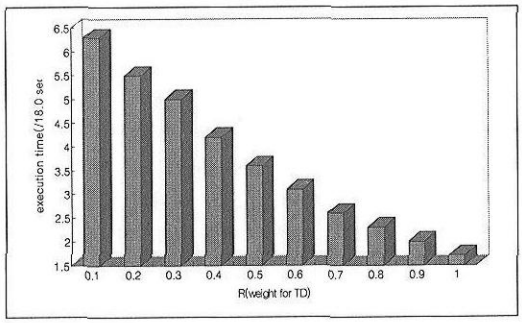



그림 12 R의 변화에 따른 평균 수행시간

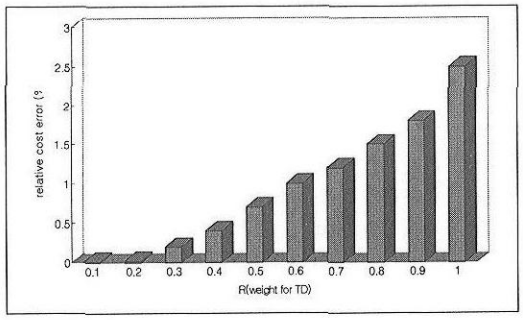



그림 13 R의 변화에 따른 평균 상대오차

### 4.5 경로탐색기 구현

개발된 경로탐색 알고리즘은 Java로 구현하였다. 객체지향언어인 Java로 구현하면 여러 플랫폼에서 수행이 가능할 뿐만 아니라 새로운 기능을 추가하거나 경로탐색 알고리즘을 다른 곳에 적용하기가 용이하다. 표 2는 Java로 포팅된 최적 경로 탐색 알고리즘의 각 클래스를 나타내며, 주요 클래스의 설명은 다음과 같다.

- Astar 클래스: 개선된 $A^*$ 최적 경로 탐색 알고리즘을 구현하는 데 사용된다.
- Database 클래스: 노선정보와 정류장 정보를 읽어들여 정보를 리스트 형식으로 구성한다. 각 파일로부터 노선 이름, 노선의 종류, 노선에 속한 정류장, 정류장 이름, 정류장의 좌표, 정류장의 종류, 정류장과 정류장 사이의 트래픽을 읽어들인다. 읽어들인 정보는 Astar 클래스에서 최적 경로 탐색을 할 때에 쓰인다.
- Graph 클래스: Database 클래스에서 리스트 형식으로 구성한 정보들을 Astar 클래스에서 사용할 수 있도록 그래프 형식의 데이타로 작성한다.

표 2 경로탐색기의 각 클래스

| 클래스 | 설 명 |
|---|---|
| DEFINE | 탐색중 사용하는 모든 상수를 정의 |
| BILIST | 탐색중 사용하는 큐 |
| LINELIST | Node의 노선정보를 저장 |
| LIST | 그래프의 Arc structure |
| RESULT | 탐색 결과를 저장 |
| VERTEX | 그래프의 노드 structure |
| Hangul | ASCII코드를 바이트로 변환(한글 변환에 사용) |
| GRAPHDATA | 읽어 들인 데이타를 그래프 데이타 형식으로 변환하기 위해 사용되는 클래스 |
| LINE | |
| NODE | |
| POINT | 좌표를 저장하기 위한 클래스 |
| LINEDATA | Database클래스에서 사용하는 클래스 |
| LINEHEADER | |
| NODEDATA | |
| NODEHEADER | |
| Astar | 최적 경로 탐색 |
| Database | 각 노선과 정거장의 정보를 읽어 리스트로 구성 |
| Graph | 데이타를 읽어 그래프를 생성 |
| Main | 질의를 받아 분석하고 탐색 |
| Util | 그래프 데이타를 생성할 때 사용하는 함수들의 모음 |

- Util 클래스: Database 클래스와 Graph 클래스에서 정보를 리스트나 그래프 형식으로 구성할 때 필요한 함수들을 가지고 있다. 예를 들면, 리스트 형식의 변수를 초기화하는 함수, 메모리에서 삭제하는 함수, 리스트에 아이템을 추가하는 함수 등을 정의해 놓은 클래스이다.
- Main 클래스: 실제 교통정보 시스템의 애플릿에서 이 클래스를 호출한다. Main 클래스는 애플릿에서 전송된 질의를 분석하고 분석된 질의에 따라 결과값을 생성하여 애플릿에게 돌려준다.

그림 14는 Java로 포팅된 경로탐색 알고리즘의 클래스들 간의 관계를 나타낸 것이다. 애플릿에서 경로 탐색 시스템에 질의를 전달하면 Main 클래스는 질의를 분석하고 필요한 데이타를 Graph 클래스에 요청하게 된다. Graph 클래스는 요청된 데이타를 찾기 위해 Database 클래스에서 생성한 정보 리스트를 Graph 클래스에 돌려준다. Graph 클래스는 리스트에서 필요한 정보를 찾아 그래프 형식의 데이타로 만들어 Main 클래스에게 돌려주고, Main 클래스는 Graph 클래스에서 받아온 그래프 데이타에서 Astar 클래스를 사용하여 최적 경로를 탐색하여 그 결과를 애플릿에게 돌려주게 된다.

Java로 포팅된 최적 경로 탐색 알고리즘은 특정 클래스만을 수정하여 컴파일함으로써 향후 개발될 교통정보 GIS와의 연동이 가능하다. 또한 데이타 베이스를 사용하는 교통 트래픽 생성기와의 연동에서도 Java를 사용하면 데이타베이스의 종류에 관계없이 정보를 읽고 쓰는 것이 용이하다.

한편, Java의 실행 속도를 향상시키기 위해 JIT, HOT-SPOT, Quick Instruction 구조를 사용하여 프로

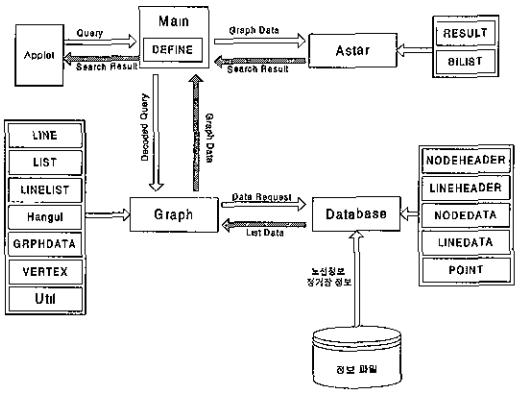



그림 14 최적경로탐색 시스템의 클래스 관계

그램의 실행속도의 약 7배 정도 향상시켰다. 그러나 이러한 방법도 C나 어셈블리 언어에 비하면 실행속도가 느리기 때문에 실행속도 향상을 위한 궁극적인 해결책은 Java 칩을 사용하는 것이다. 소프트웨어로 구현된 컴파일과 인터프리터를 하드웨어로 실행하거나 클래스 파일을 기계어로 취급하여 직접 실행한다면 상당한 수준의 속도 향상을 기대할 수 있을 것이다.

## 5. 교통정보 저장 및 노선/정류장 관리 시스템

### 5.1 교통정보 저장 시스템

교통정보 저장시스템을 개발하기 위한 접근 방법으로는 파일시스템, 관계형 데이타베이스 시스템, 객체지향 데이타베이스 관리기법이 있다. 제안된 시스템에서는 일반적으로 많이 사용되고 있는 오라클 데이타베이스를 이용하여 노선 및 정류장 정보 저장시스템을 개발하였다. 정류장 및 노선과 같은 정적 raw-data 취득은 서울시의 일부 구간인 강남지역 대중교통 노선 및 정류장 정보를 각 버스회사로부터 또는 해당 관계기관으로부터 수집하였다. 또한 데이타의 취득 후에는 해당 대중교통 수단의 정류장 및 노선 정보를 지도 이미지 파일에 표기하며, 각 정류장 및 노선의 절대좌표를 얻어내어 이를 파일 형태로 저장하는 작업을 수행하였다.

한편, 화면상에 서울시의 지도를 보여주고 그곳에서 직접 출발지와 목적지를 선택하는 형태로 설계하기 위해 상용으로 판매되는 대동여지도를 변경하여 사용하였다. 대동여지도는 서울시 전체지도를 $x$축 7800, $y$축 6800의 해상도로 표현한다. 지도가 하나의 파일로 구성하기에는 무리가 있기 때문에 이를 221개의 600 * 400의 크기로 이루어진 단위지도들로 구성하였다. 이 각각의 파일은 두 자리의 숫자로 구성된 파일명을 가지며, 웹 브라우저에서 기본적으로 디스플레이 시킬 수 있는 GIF형태의 그래픽 파일로 구성된다.

또한, 본 논문은 마우스 또는 터치스크린 등의 포인팅 디바이스를 이용하여 실제의 지도 위에서 버스정류장 또는 지하철역들을 선택하도록 되어있기 때문에 이들 버스 정류장과 지하철역의 좌표들이 필요하게 된다. 그러므로 이들 지점들의 좌표를 구하여 파일 형태로 구성하였다. 이 때 이 지점들의 좌표는 7800 * 6800의 서울시 전체 지도에서 좌측 최상단을 (0,0)으로 설정한 좌표계를 사용하여 저장한다.

버스정류장 또는 지하철역의 좌표를 저장할 때 단순하게 정류장의 좌표만을 택하여 이들 간의 연결상태를 그려준다면 원래 지도상의 도로의 형태를 충실히 재생해 줄 수 없는 문제가 발생할 수 있다. 이런 점을 해결

하기 위해서 두 지점을 연결해주는 지도상의 도로가 직선형태가 아닐 경우에는 이를 재생해 주기 위해서 그 도로의 꺾기는 지점에 대한 좌표를 추가적으로 취득하여 이를 저장하게 된다. 위의 과정에서 취득한 좌표를 가지고 {일련번호, 노선번호, 지도이름, 지도상의 상대 $x$ 좌표, 지도상의 상대 $y$좌표, 노선번호, 추가링크정보}의 형태로 정류장 정보를 입력하며, 추가정보는 현재 정류장과 다음 정류장 사이의 꺾기는 지점에 대한 정보를 표시하고, 노선의 기점부터 종점까지 정류장을 순서대로 입력하여 노선정보를 나타낸다.

본 논문에서 제안된 시스템에서는 서울시의 강남지구를 대상으로 일반버스 및 좌석버스의 데이타를 확보하고 이를 확대 편집하였다. 서울시 전역으로 확대 편집하는데 있어서 기존의 지도에는 표시되지 않은 도로 등의 모습도 실제 지리정보와 같도록 함께 편집하였다. 특히 버스인 경우에는 이러한 작업을 해주어야 한다. 실제 도로상에서도 도로의 좌측면에서의 정류장과 우측면에서의 정류장의 위치가 다른 경우가 많고 또한 사거리나 아파트단지 같은 경우에는 같은 이름을 가진 정류장이 여러 곳에 산재한 경우가 많다. 따라서 실제 정밀한 교통정보를 위해서는 노선마다 실제 승하차 기점을 노선정보에 반영하여야 한다.

### 5.2 노선/정류장 관리 시스템

현실적인 대중교통 안내 서비스를 제공하기 위해서는 정류장 및 노선에 관한 정적인 raw-data의 데이타베이스 구축이 중요하다. 이러한 자료가 제대로 구축되어야 버스 및 지하철 노선과 도로를 네트워크 모델로 보고 차량 및 교통인구, 소요시간 등의 교통량을 파라미터로 하여 정확한 교통정보를 제공할 수 있기 때문이다. 그러나 지하철과 달리 버스사업은 민간기업에서 운영하기 때문에 경영수지에 따라 노선이 수시로 바뀌기도 하고 없어지기도 하며 생성되기도 한다. 따라서, 이러한 정적 raw-data의 변화를 각 버스 회사에서 손쉽게 브라우저상에서 바로 반영하여 관리하는 것이 필요하다.

본 논문에서 제안된 시스템에서는 정적 raw-data 관리시스템을 Oracle 서버와 연동되는 자바 어플리케이션으로 개발하여 플랫폼에 독립적으로 모든 입력, 수정, 삭제 등의 작업을 단말 화면의 실제 교통지도 위에서 수행하였다. 이는 변동된 노선 및 정류장 데이타를 관리자가 직접 눈으로 확인 할 수 있으며 지도상의 실제 좌표를 알 필요 없이 단말 화면상에 보이는 위치만으로 작업할 수 있어 정류장 및 노선 raw-data의 구축, 유지관리에 드는 많은 경비를 절감할 수 있다.

개발된 시스템을 위한 정적 raw-data는 크게 지도, 노선, 정류장 데이타로 이루어져 있다. 지도 데이타는 상용 지도를 본 시스템에서 사용하기 알맞게 편집을 하였으며, 노선 정보는 버스조합, 버스 노선도 등을 참고로 하여 만들었고, 정류장 정보는 노선 정보에 포함된 모든 정류장을 토대로 하여 구축하였다.

#### 5.2.1 DB 구조

DB는 그림 15와 같은 테이블들을 가지고 있는데, DB 관리를 위한 입력기는 노선(LINE테이블)을 기준으로 하여 노선의 입력, 수정, 삭제와 노선을 구성하고 있는 각 정류장(STATIONS 테이블)에 대한 입력, 수정, 삭제를 주축으로 한다. 각 노선과 정류장은 타입별로 구분되어 있으며, 타입에는 일반버스, 좌석버스, 지하철, 도보 등이 타입테이블에 저장된다. 한 노선에 대한 정류장이 모두 입력되면, 노선을 기준으로 하여 노선이 지나가는 정류장을 별도의 테이블(LINE_STATION)에 자동 입력되게 하고, 정류장을 기준으로 정류장을 지나가는 노선들에 대한 정보도 별도의 테이블(PASS_LINE)에 자동 입력되게 한다.

STATIONS 테이블은 정류장에 대한 정보를 포함하고 있으며, LINE 테이블은 노선에 대한 정보를 포함한다. 즉, STATIONS는 정류장을 기술하기 위한 자료로써 정류장의 고유 ID, 이름, 전체 지도에서의 절대 좌표, 그리고 정류장이 일반버스용인지 또는 좌석버스용인지 등의 구분을 위하여 사용된다. 그리고 LINE은 노선을 기술하기 위한 자료로써 노선의 고유 ID, 노선을 구분하기 위한 이름(예: 일반버스 1번), 노선 종류(일반버스, 좌석버스, 지하철 등)에 대한 ID 등이 사용된다.

각 정류장을 기준으로 같은 이름의 정류장이지만 반대방향의 정류장이라든지, 정류장이 두 세 군데로 나누어진 경우는 서로 다른 테이블(SUBSTATION)에 별도로 관리하도록 하였다. 그리고 서울시 전역에 대한 지도의 비트맵 이미지에 대한 정보는 지도 테이블에 저장된다. 본 NEXT_STATION은 시스템 운영 시 인근 정류장에 대한 정보를 추출하는 데 시간을 단축시키기 위하여 LINE 테이블이 생성될 때 자동으로 생성되도록 하였다. LINE_STATION은 하나의 노선 운행 시 경유하는 모든 정류장에 대한 정보를 저장할 수 있도록 하였고, PASS_LINE 또한 시스템 운영 시 반응 시간을 줄이기 위하여 하나의 정류장을 경유하는 모든 노선에 대한 정보를 저장할 수 있도록 하였다.

이러한 시스템은 웹 브라우저를 사용하여 가시적인 면을 보이고, 모든 데이타의 입력, 수정, 삭제를 실제 눈에 보이는 지도상에서 수행함으로써 관리자의 편리함을 도모하게 된다.

5.2.2 사용 시나리오

노선 및 정류장 관리시스템을 실행하면 프로그램은 사용자에게 인증을 요구한 다음, 서버로 접속하여 현재 업데이트된 정보가 있는지 여부를 확인하고, 있다면 지역에 저장된 데이타를 업데이트시키는 작업을 하게 된다. 이러한 일련의 초기화작업이 끝난 다음 제어권은 관리자에게 넘어가게 된다. 그림 16은 노선 및 정류장 DB 관리 시스템 초기화면 화면을 보여준다.

본 프로그램에서 기능의 선택은 메뉴를 통해 사용할 수 있으며 그림 17에 나타나는 것과 같은 항목과 기능을 가지고 있다.

정류장 등록할 경우 필요한 자료는 세 가지로 생각할 수 있다. 정류장의 이름, 정류장의 위치, 정류장의 타입이 그것이다. 정류장 등록과정은 위의 두 가지를 입력

| 테이블 이름 | 필드 이름 | 데이타 타입 | 내 용 |
|---|---|---|---|
| STATIONS (한 정류장에 대한 정보) | STSTION_ID | Numeric(5), Identity | 정류장 ID |
| | NAME | Char(50) | 정류장 이름 |
| | X | Int | X 좌표 |
| | Y | Int | Y 좌표 |
| | TYPE_ID | Numeric(2) | 정류장 종류 ID |
| | MEMO | Text | 기타 정보 |
| LINE (한 노선에 대한 정보) | LINE_ID | Numeric(4), Identity | 노선 ID |
| | LINE_NAME | Char(50) | 노선 이름 |
| | TYPE_ID | Numeric(2) | 노선 종류 ID |
| | COMMENT | Text | 기타 정보 |
| SUB STATIONS (동일한 이름의 주변 정류장) | STATION_ID | Numeric(5) | 정류장 ID |
| | SUB_ STATION_ID | Numeric(5) | 동일한 이름의 주변 정류장 |
| | SUB_X | Int | X 좌표 |
| | SUB_Y | Int | Y 좌표 |
| | COMMENT | Text | 기타 정보 |
| NEXT_ STATION (정류장의 이웃한 정류장들) | STATION_ID | Numeric(5) | 정류장 ID |
| | NEXT_ STATION_ID | Numeric(5) | 정류장 ID |
| | COST | Float | 비용 |
| | TYPE_ID | Numeric(2) | 노선 종류 ID |
| LINE_ STATION (한 노선이 지나가는 정류장들에 대한 정보) | STATION_ID | Numeric(5) | 정류장 ID |
| | LINE_ID | Numeric(4) | 노선 ID |
| | LINE_INDEX | Int | 노선에서 인덱스 |
| | LINE_MEMO | Text | 기타 정보 |
| PASS_LINE (정류장을 지나는 노선에 대한 정보) | STATION_ID | Numeric(5) | 정류장 ID |
| | LINE_ID | Numeric(4) | 노선 ID |
| | PASS_MEMO | Text | 기타 정보 |

그림 15 데이타베이스의 테이블 구조

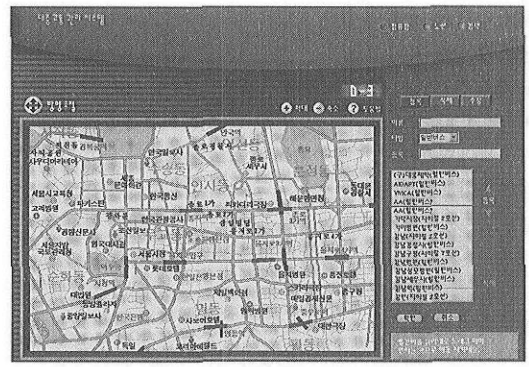


그림 16 노선 및 정류장 DB 관리 시스템

받고 이를 최종적으로 확인을 받아 정류장 정보에 추가하게 된다. 만약 추가할 정류장의 정확한 위치를 모를 경우 각 좌표 값을 빈칸으로 두고 지도상에서 직접 선

| 주메뉴 | 서브 메뉴 | 기 능 |
|---|---|---|
| 파일 | 열기 | 로컬에 임시 저장된 작업 파일을 연다. |
| | 저장 | 서버와 연결이 되지 않는다면 작업한 내용을 로컬에 임시로 저장 |
| | 서버로 보냄 | 서버측으로 작업한 데이타를 보냄 |
| | 작업 마침 | 프로그램의 사용을 끝냄 |
| 정류장 관리 | 정류장 등록 | 정류장의 위치와 이름을 등록 |
| | 정류장 수정 | 이미 존재하는 정류장의 위치와 이름에 대한 재설정 |
| | 정류장 삭제 | 이미 존재하는 정류장의 위치와 이름을 삭제 |
| 노선 관리 | 노선 등록 | 노선의 이름과 경로 등을 등록 |
| | 노선 수정 | 이미 존재하는 노선의 이름과 경로에 대한 재설정 |
| | 노선 삭제 | 이미 존재하는 노선의 이름과 경로를 삭제 |
| 정보 검색 | 노선 검색 | 지정된 노선이름으로 노선에 대한 정보를 얻어 옴 |
| | 정류장 검색 | 지정된 정류장이름으로 정류장의 위치에 대한 정보를 얻어 옴 |
| | 경로 검색 | 지정된 경로상을 통과하는 모든 대중교통의 노선정보를 얻어 옴 |
| 인쇄 | 노선 정보 | 현재 화면의 노선 정보를 프린터로 출력 |
| | 정류장 정보 | 현재 화면의 정류장 정보를 프린터로 출력 |
| 도움말 | | 본 관리툴을 사용하는데 도움말을 제공 |

그림 17 노선 및 정류장 DB 관리 시스템 메뉴

택을 해주면 이를 좌표 값으로 바꿔주므로 관리자가 숫자로 된 정류장의 위치를 몰라도 상관없다. 이상의 진행 과정은 그림 18과 같다.

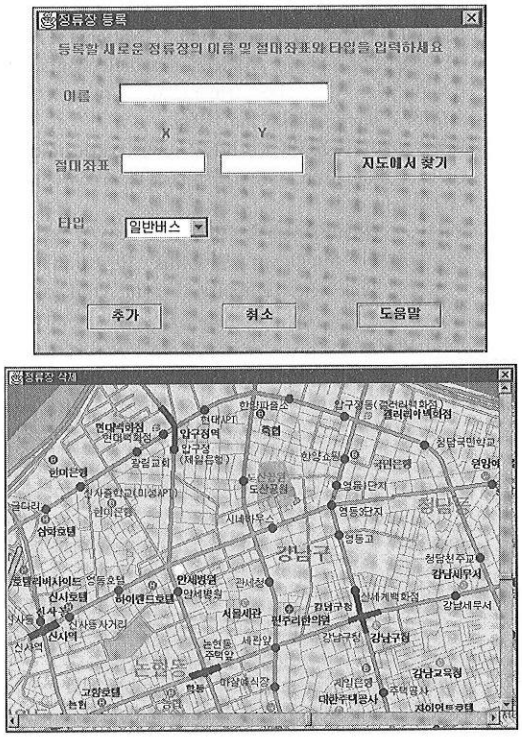



그림 18 정류장 등록화면과 지도에서 위치 찾기 화면

정류장 삭제 시 관리자는 전체 지도에서 원하는 정류장이 위치한 지역을 선택을 하고, 관리 프로그램은 해당 지역 내의 인접 정류장이름을 데이타베이스로부터 가져와서 보여주게 된다. 이 중 사용자로부터 삭제하고자 하는 정류장을 선택받아 정류장을 삭제하게 된다. 삭제를 선택하게 되면 프로그램의 내부적인 루틴에 의해서 일단 모든 노선을 조사하여 만약 삭제하고자 하는 정류장을 지나는 노선경로가 존재한다면 이를 관리자에게 알리고 정류장 삭제 작업을 취소하게 된다. 따라서 정류장을 삭제하고자 할 경우에는 노선 경로를 확인하는 절차가 필요하게 된다. 이는 만약 실수로 노선 정보를 관리자가 삭제할 수도 있기 때문이다.

노선 등록과정에 필요한 데이타는 크게 세 가지로 나눌 수 있다. 첫째 노선이름, 둘째 등록하고자 하는 노선을 사용하는 대중교통 수단의 종류이며, 마지막으로 모든 정류장 이름에서 실제 노선을 순서대로 가져다가 이를 적용시키는 것이다. 노선을 삭제하기 위해서는 현존하는 모든 노선을 제시하고 이 중에서 관리자가 선택하여 삭제하는 것이 바람직할 것이다.

노선 검색은 관리자가 하나의 노선명을 주어졌을 경우 해당 노선명을 가진 것이 어떤 경로로 도착지까지 연결이 되어 있는가에 대한 연결도를 보여준다. 따라서 관리자는 시각적으로 노선의 구성을 알 수 있으며, 이를 수정하는데 용이하게 사용되어질 수 있게 된다. 노선 검색을 하는 과정은 일단 하나의 노선을 선택하게 되면 그림 19와 같은 결과를 볼 수가 있다.

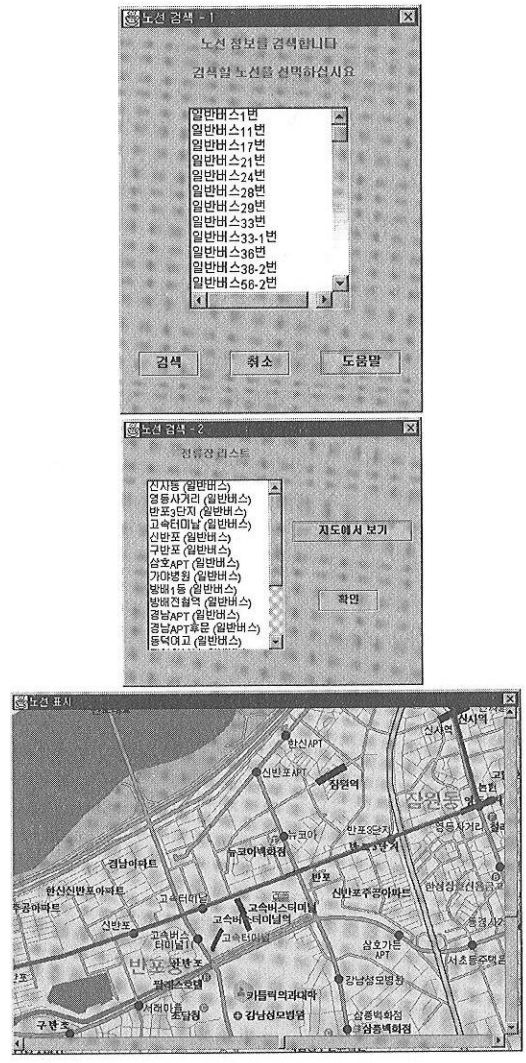



그림 19 노선 검색을 통하여 일반버스 1번의 노선을 검색한 결과

## 6. 결론 및 향후연구

대도시의 교통문제를 해결하기 위한 가장 좋은 방안은 도로나 지하철을 확장하거나 새로 건설하는 것이지만 이는 많은 비용과 오랜 시간을 요한다. 따라서 이와 같은 하드웨어적인 접근방법이 아닌 소프트웨어적인 접근방법으로 교통 문제를 해결하기 위하여 본 논문에서는 웹 기반 대중교통 안내 시스템 개발 경험을 소개하였다. 개발된 시스템은 클라이언트와 서버시스템, 수시로 변하는 현재의 교통상황을 반영하여 최적의 경로를 제공하는 탐색시스템, 그리고 교통정보 저장시스템과 노선 및 정류장 관리시스템으로 구성되어 있다. 본 논문에서 제안된 시스템은 객체지향 언어이자 플랫폼인 Java를 사용하였기 때문에 확장이 용이하고, 인터넷에서 상용으로 제공되고있는 교통 및 다른 서비스와도 연동이 용이하며, 사용자 요구가 변경되어도 단기간 내에 시스템을 개발할 수 있고, 유지보수 비용이 상대적으로 적게 드는 장점을 가지고 있다. 그리고 향후 구축될 ITS의 한 모듈로서 곧바로 연동할 수 있어 사용자들이 손쉽게 언제, 어디서나 교통정보를 이용할 수 있는 수단을 제공해줄 수 있다.

향후 연구로는 관계기관의 협조를 얻어 실제 현장의 동적인 실시간 교통 raw-data를 제안된 시스템의 교통입력데이타로 사용할 예정이며, 교통정보 GIS를 구축함으로써 보다 다양하고 정밀한 양질의 교통 지리정보와 GPS를 활용한 버스의 위치정보, 배차 스케줄 등의 서비스를 제공할 계획이다. 그리고, 시스템 성능 저하라는 제약이 있지만 확장성, 상호작용성, 이기종 간의 연동성을 위하여 구현 플랫폼으로 CORBA (Common Object Request Broker Architecture)를 사용할 예정이다.

## 참 고 문 헌


[1] D.J. Dailey, M.P. Haselkorn, "Demonstration of an Advanced Public Transportation System in the Context of an IVHS Regional Architecture", the 1st World Congress on Applications of Transport Telematics and Intelligent Vehicle-Highway Systems, Nov. 1994.

[2] EUROBUS, "Case Study on Public Transport Contribution to Solving Traffic Problems," EUROBUS Project, Deliverable 18 (Version 2.0), 1994.

[3] EUROBUS, "Definition of Main Function Areas," EUROBUS Project, Deliverable 3, 1992.

[4] Federal Transit Administration, "Advanced Public Transportation Systems: The State of the Art Update '96", U.S. Department of Transportation FTA-MA- 26-7007-96-1, Jan. 1996.

[5] E.C. Chang, K.K. Ho, P. Fei, "GIS-T Design ITS Applications", 3rd World Congress on ITS, Vol.1, pp. 522-528, Oct. 1996.

[6] Federal Transit Administration, "National Transit GIS: Data Standards, Guidelines and Recommended Practices", U.S. Department of Transportation DTRS57-95-P-80861, Jan. 1996.

[7] 교통정보센터(KORTIC), http://www.kortic.or.kr/SEOUL/st_main.html, 교통정보서비스센터, 수도권(서울시) 대중교통

[8] Infobank, 환승정보시스템, 1997

[9] 한국항공대학 Intelligent Systems Research Lab, "ATIS: The Next Generation", Int. conf. on 8th world congress on Transport Research, Antwerp, July, 1998.

[10] Federal Transit Administration, "Advanced Public Transportation Systems: The State of the Art Update '96", U.S Department of Transportation FTA-MA- 26-7007-96- 1, January 1996

[11] SORTA Homepage, http://www.sorta.com

[12] Enhanced Planning Review of the Honolulu, http://www.fta.dot.gov/library/planning/plnrpt/hon3-htm.htm, Apr. 1996.

[13] Busview:Graphical Display of Real-Time Transit Coach Locations, http://www.ivhs.washington.edu/projects/busview.html, Mar. 1997.

[14] EUROBUS, "Case Study on Public Transport Contribution to Solving Traffic Problems, "EUROBUS Project, Deliverable 18(Version 2.0), 1994.

[15] EUROBUS, "Definition of Main Function A-reas," EUROBUS Project, Deliverable 3, 1992

[16] Review and Assessment of Enroute Transit Information Systems, http://www.fta.dot.gov/library/technology/APTS/enroute.htm, Apr. 1995.

[17] QUadrilateral Advanced Research on Telematics for Environment and Transport Drive Project, http://www.transport.civil.ntua.gr/qrtplus/QuartetTestSite/Quartet/Q1.htm

[18] Minitel Homepage, http'//www.minitel.fr

[19] Advanced Public Transportation Systems, http://www.fta.dot.gov/fta/library/technology/APTS/ 333APT/333APT.HTM, Apr. 1992.

[20] VERTIS, "VERTIS.ITS Services", http://www iijnet.or.jp/vertis

[21] A. V. Hof, S.Shaio, O. Starbuck, "Hooked On Java", SUN Microsystems, 1996.

[22] David Flanagan, "Java in a Nutshell", O'Reilly, 1996.

[23] P.H. Winston, Artificial Intelligence, 1992, Addison-Weslet

[24] Judea Pearl, Heuristics: Intelligence Search Strategy

for Computer Solving, Addison-Wesley Publishing Company, pp 64-99, 1984.
[25] Patrick Henry Winston, Artificial Intelligence, Judea Pearl, Heuristics, Addison-Wesley Publishing Company, pp 81-100, 1992.
[26] R. E. Tarjan, "Fast Algorithms for Solving Path Problems", JCAM, Vol.28, No.3, pp.594-614, 1981.
[27] Hart, Nilson, and Raphael, "A Formal Basis for The Heuristic Determination of Minimum Cost Paths", IEEE Transaction on SSC, SSC-4(2), pp. 100-107, 1968.
[28] I. Pohl, "First Results on the Effect of Error in Heuristic Search", In B. Meltzer and D. Michie, editors, Machine Intelligence Vol. 5, pp. 219-236, 1970.
[29] 김명제,"그래프에서의 최단경로 탐색 알고리즘에관한 연구", 1992. 2, 경희대학교 전자계산공학과 석사논문.
[30] 올림픽대로 교통정보, http://www.seoul.npa.go.kr, 서울특별시, 서울지방경찰청.
[31] 교통정보마당, http://traffic.metro.seoul.kr, 서울시 교통관리실.

배 수 강
1996년 경희대학교 전자공학과(공학사). 1999년 경희대학교 전자계산공학과(석사). 1999년 ~ 현재 경희대학교 전자계산공학과(박사과정). 관심분야는 실시간 시스템, Garbage Collection, Java, Real-time Java, Embedded System

이 승 룡
1978년 고려대학교 재료공학과(공학사). 1986년 미국 일리노이 공대 전산과학(석사). 1991년 일리노이 공대 전산과학(박사). 1992년 ~ 1993년 Governors State University, Illinios 조교수. 1993년 ~ 현재 경희대학교 전자계산공학과 부교수. 관심분야는 실시간시스템, 실시간 고장허용시스템, 멀티미디어 시스템, 실시간 CORBA Java

최 대 순
1985년 고려대학교 금속공학과 졸업(공학사). 1988년 미국 FDU Operations Research(석사). 미국 Polytechnic Univ. 교통공학과(석사 · 박사). 1993년 ~ 1995년 TTRC at Polytechnic Univ. 연구원. 1995년 ~ 현재 한국건설기술연구원 선임연구원.

정 태 충
1980년 서울대학교 전자공학과 졸업(학사). 1982년 한국과학기술원 전산계산학과(공학석사). 1987년 한국과학기술원 전산계산학과(공학박사). 1987년 ~ 1988년 3월 KIST 시스템 공학센터 선임연구원. 1988년 3월 ~ 현재 경희대학교 전자정보학부 전산전공 교수. 관심분야는 인공지능, Timetabling, Expert system, Meta search, 인터넷응용 S/W

승 현 우
1981년 서강대학교 영문학과 졸업(학사). 1985년 미국 일리노이 공대 전산학 석사. 1991년 미국 일리노이 공대 전산학 박사. 1992년 ~ 1994년 한국통신 소프트웨어 연구소 데이터베이스 연구실장(선임연구원). 1994년 ~ 현재 서울여자대학교 컴퓨터학과 부교수. 관심분야는 GIS, 데이타베이스시스템, 객체지향 데이타베이스