Design of Vehicle Location Tracking System using Mobile Interface

Ji Moon Chung*, Sung Choi**, Keun Ho Ryu***

Contents

Key Words:Vehicle Tracking System, Mobile Interface, Moving Object, Real-time Position Tracking

Abstract

Recent development in wireless computing and GPS technology cause the active development in the application system of location information in real-time environment such as transportation vehicle management, air traffic control and location based system. Especially, study about vehicle location tracking system, which monitors the vehicle's position in a control center, is appeared to be a representative application system. However, the current vehicle location tracking system can not provide vehicle position information that is not stored in a database at a specific time to users. We designed a vehicle location tracking system that could track vehicle location using mobile interface such as PDA. The proposed system consist of a vehicle location retrieving server and a mobile interface. It is provide not only the moving vehicle's current location but also the position at a past and future time which is not stored in database for users.

* Ji Moon Chung(Namseoul University, jmchung@nsu.ac.kr)

** Sung Choi(Namseoul University, sstar@nsu.ac.kr)

*** Keun Ho Ryu(Chungbuk National University, khryu@dblab.chungbuk.ac.kr)

※ This research is performed to the support of Namseoul University.

I. Introduction

As the wireless networking technology and mobile device development, traditional information systems are able to move to wireless communication environment. And development of the wireless environment and GPS technology cause to development energetically application system of location information in real-time environment such as transportation vehicle management, air traffic control and location based service systems. Especially vehicle location tracking system(VLTS), which monitors the vehicle's position in a control center in real-time, is the representative research in this area. But traditional VLTSs are able to search only location of vehicle in online-network, use traditional commercial DMBS. So those systems can not support uncertain future and past location information of moving objects

In this paper, we implemented vehicle location tracking system which can retrieve vehicle location in real-time using mobile interface on the wireless networking. Implemented system consists of two parts: one is VLTS server, the other is mobile interface. VLTS server stores and manages the location information of vehicle, and supports retrieved results of vehicle location information to mobile interface. Mobile interface requests vehicle location information to server and prints retrieval results supported from server. For implementing server system for vehicle location tracking, we will define modeling of vehicle location information. And we will present architecture of VLTS, functions of each component in the presented system and algorithms for each component. Especially we define the method for estimation of past and future location using linear interpolation based on historical location information saved in database. For mobile interface, we design and implement the server connecter, transmitter-receiver. Using these module, the implemented system is able to retrieve vehicle location in real-time in wireless environment. Implemented system is able to estimate past and future location information not saved in database, and process query related vehicle location saved in database using mobile interface.

The proposed system is able to retrieve vehicle location such as following query.

'Retrieve location information for whole or specified time interval of all vehicles'

'Retrieve location information for whole or specified time interval of specified vehicle'

'Retrieve location information for specified time stamp of specified vehicle'.

And through the implementation of the system, we will show that the system work properly like above queries.

The outline of this paper is as follows. In chapter 2, we will show the previously

developed VLTSs and related works. And the architecture of proposed VLTS and algorithm will be presented and define modeling for location information moving vehicle in chapter 3. In chapter 4, we will show the implementation environment of the proposed system, and will be conclude in chapter 5.

Ⅱ. Related works

There are several VLTSs such as Commercial Vehicle Operations (CVO) [1, 4], Advanced Public Transport System (APTS) and vehicle management and control system of EuroBus[2,5,11]. CVO efficiently manages cargo/cargo vehicles in order to reduction of logistic cost, prevention of accident and promotion of emergency measure. This system consists of two part: Freight and Fleet Management (FFM), Hazardous Material Monitoring (HMM). FFM traces cargo/cargo vehicles and forwards variety of information to the drivers to prevent empty cargo and to figure out the optimal interval time between vehicles. HMM administers vehicles with hazardous cargo and then traces their location to make sure they do not enter prohibited or dangerous route. In case of emergencies, such as accidents, HMM cooperate with other traffic control facilities to deal with the incidents. APTS supports information related vehicle location, monitoring vehicle, vehicle management to user (passenger or transport company). In the vehicle management and control system of EuroBus, the whole information related to bus equipped with receiver sends to the central center and central center supports related information as well as location about bus to the company and bus stop.

But these systems using conventional commercial DBMS have some problems since they deal with continuously moving vehicle. Generally if continuous moving objects are managed by a conventional database, it is impossible to store all position information changed over time in the database. Therefore a time period of regular rate is determined and location information of moving vehicles are discretely stored in the system for every time period. However in this case, DBMS cannot support information if user requires information which is not saved in DBMS. For example, when we store the location information of the express transport cargo which moves to seoul from busan, we will decide start time and time-intervals for storing the location of vehicle such as the 5miniutes intervals from 8:00AM. As a result, the vehicle location stored in DMBS will be location at 8:05AM, 8:10AM, 8:15AM. But if user requires the vehicle location at 8:06AM, 8:13AM, the system cannot support properly query result. To solve these problems, research on VLTS using moving object database (MODB) in which vehicle

defines as moving point object is actively in progress[12, 13, 14]. Moving objects are classified into moving point and moving region which continuously changes over time. Moving point is the object that changes its location over time such as human, animal, vehicle, airplane, ships. Moving region is the object that changes its shape as well as location over time such as administrative districts in the country, the region under the influence of typhoon, status of cancer cells [6-9].

There are several researches on vehicle managing systems such as DOMINO [10, 15-18], CHOROCHRONOS [19-23], Battlefield Analysis [24-26].

Prototype of DOMINO project is focused on predicting future location of moving objects based on present location of the object, speed, and direction. However this prototype deals with only uncertain future location of vehicle and it does not store history information of uncertain past movement of moving objects. So this prototype cannot support uncertain past location information.

In the COROCHRONOS project, research on data modeling of moving vehicle, index, and vehicle management system which manages the location and trajectory of vehicle was performed. Especially in this project, they presented application scenario applied in transport management system based on GPS and multimedia system. With the result of this project, they cannot support information related to future location because they take into consideration past location data of vehicle. And they didn't make a prototype like DOMINO until now.

The prototype of the battlefield analysis defines moving units and tank as moving vehicle which have property of moving point object. This prototype is able to predict the motion of moving units and tank for decision making in the simulation battle field. But this prototype cannot properly process data supplied in real-time, therefore it cannot apply in mobile environment.

In this paper, we propose the VLTS capable of predicting past and future location of moving vehicle which is not saved in DBMS. Proposed VLTS is able to process for query related to vehicle location using mobile interface.

Ⅲ. Design of VLTS using mobile interface

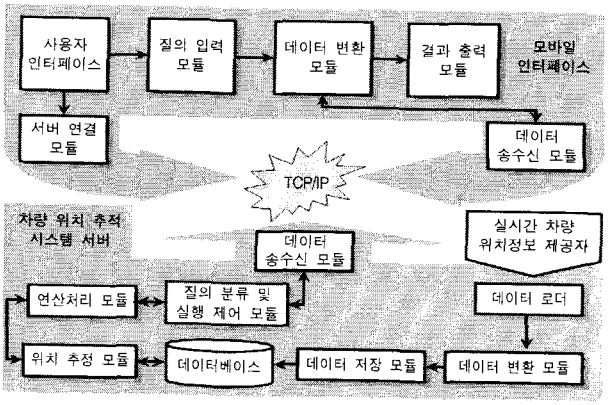
In this chapter, we will present the architecture of real-time vehicle monitoring system and the function of each module. And then we will define modeling for location information of moving vehicle. For the data modeling, we will define moving point object data and operations and present historical and current information

of moving vehicle saved in DBMS. Finally we will present the DB schema used in the proposed system and algorithms for each module.

3.1 System architecture and functions

The proposed system consists of VLTS server and mobile interface such as figure 1. VLTS server stores and manages the location information of vehicle, and supports retrieved results of vehicle location information to mobile interface. VLTS server is composed of data loader, data conversion and storing module, data transceiver/receiver, query classification and execution control module, operation processing module, location prediction module and DMBS. Mobile interface requests vehicle location information to server and shows retrieval results supported from server. Mobile interface is composed of user interface, server connection module, query input module, data conversion module, displaying result module, data transceiver/receiver.

그림 1. 모바일 인터페이스를 이용한 차량 위치 추적 시스템 구성도



The functions of each module in the VLTS server are as following. Data Loader (DL) receives and collects the location information of a moving vehicle. The collected data are pipelined through Data Conversion Module (DCM) and Storage Module (SM) to be recorded in the database. The position information of the vehicle will be fed from an external Data Transmitter (DT) at regular intervals. Query Classification and Execution Control Module (QCECM) calls and controls Operation Processing Module (OPM) to process the vehicle position query requests received from Mobile Interface (MI) on wireless network. When MI requests past or future position information not stored in the database, Location Prediction Module (LPM) processes the query request. LPM will generate a mathematical function of vehicle's position over time to calculate and predict the position of the vehicle. The function uses a past position deduction function and a future position prediction function.

3.2 Location information model of moving vehicle

In this paper, a moving vehicle is defined as a mobile point object for information management purpose; moving vehicle is a mobile object that changes only position as time changes. Moving vehicle *MV* has time, space, and general properties, and is

expressed as . $MV = \langle T_A, S_A, G_A \rangle$. MV 's time property is defined as $TA = \langle \upsilon t_s, \upsilon t_e \rangle$ where υt_e is the starting time and υt_e is the ending time. υt_s and υt_e are elements of the valid time set, S_{VT}. The valid time set is a subset of the real time set and each element in the set follows the order of $S_{VT} = \{t_0, t_1, t_2 \cdots, t_k, \cdots t_{now}\}$. The elements are defined as $t_k = t_{k-1} + 1$, $t_k = t_0 + k$, $k \geq 0$ integer. t_{now} is a time constant indicating the current time. The domain of the valid time is of linear, discrete and absolute time, and the database has a constant period of valid time. $S_A = \langle x y \rangle$, $x, y \in R$, space property, R is a real number

〈Table 1〉 Operators for processing location information

Operator	Function
SobjSTraj	Retrieval of trajectory information of an arbitrary vehicle over specific time period
FobjFTraj	Retrieval of trajectory information of all vehicles over whole time period
TrajDistance	Calculation of the straight distance between inquired vehicles over the trajectory
TrajNearest	Retrieval of the closest trajectory from the trajectory of a an arbitrary vehicle in motion
PositionAtTime	Retrieval of the position information of a moving vehicle over a specific time period
MDistance	Retrieval of the distance between two arbitrary vehicles

The database of the mobile vehicle location information is organized as a set of mobile vehicles. The historical record set of the database composed of S_{MV} becomes $m\upsilon_i$. Each historical record set of $H_{MV} = \{Hm\upsilon_i\}^n_{i=0}$ belonging to $m\upsilon_{i_k} = \langle T_A(m\upsilon_{i_k}), S_A(m\upsilon_{i_k}), G_A(m\upsilon_{i_k}) \rangle$. $m\upsilon_{i_k}$ means $m\upsilon_i$ kth historical information of $T_A(m\upsilon_{i_k})$, $m\upsilon_i$, kth time property, $S_A(m\upsilon_{i_k})$, kth space property, and $G_A(m\upsilon_{i_k})$, kth general property.

For information management purpose, the following operations are defined; general properties of mobile vehicles are not defined here since the functions of existing commercial DBMS will be used without modification. For location related operators, SobjSTraj, FobjFTraj, TrajDistance, TrajNearest, PositionAtTime, MDistance are defined as suggested in the Table 1.

The operators defined in Table 1 are the actual implementations of the operators for mobile point objects suggested by Erwig[6.7], implemented in this paper to enable retrieval of vehicle position information.

3.3 DB schema

Data storage schema is based on the relational database structure, and the data will be stored, classified into three relations, FreetObject, FleetCurrent, and FleetHistory. FleetObject records general properties of moving vehicles.

〈Table 2〉 FleetObject relation

CarId	Name	Type
356583455	fleet1	truck
356583466	fleet2	container

CarID of FleetObject relation in Table 2 is an object identifier and key value. Other general properties such as names of vehicles, and vehicle types can be recorded in FleetObject relation as well. FleetCurrent keeps track of current time, and time and space properties of moving vehicles.

〈Table 3〉 FleetCurrent relation

CarId	Vs	Ve	X_Vs	Y_Vs	X_Ve	Y_Ve
356583455	2002-03-01 -08-20-00	now	202053.00	444755.69	null	null
356583466	2002-03-01 -09-00-00	now	201153.42	445201.23	null	null

FleetCurrent relation in Table 3 has the same structure as FleetHistory relation in Table 4. The most up-to-date records of the positional coordinates of a vehicle will be recorded in FleetCurrent and all the other records will be stored in FleetHistory. FleetHistory relation archives all the past historical records of the location information. Table 4 is an example of FleetHistory relation, keeping track of vehicle position at 5 minutes interval.

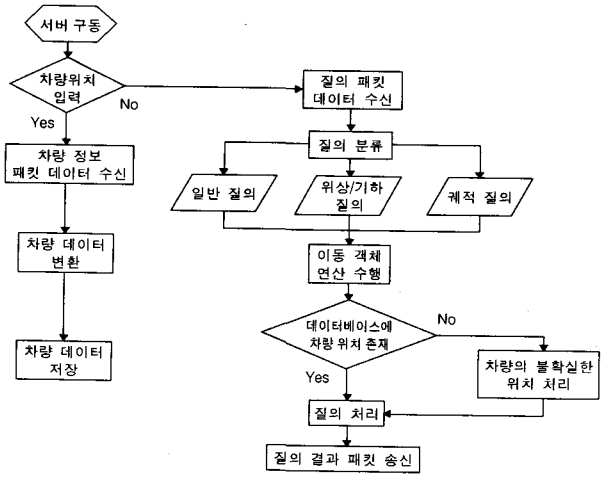
〈Table 4〉 FleetHistory relation

CarId	Vs	Ve	X_Vs	Y_Vs	X_Ve	Y_Ve
356583455	2002-03-01 -07-50-00	2002-03-01 -07-55-00	200998.11	445124.01	201287.75	445238.44
356583455	2002-03-01 -07-55-00	2002-03-01 -08-00-00	201287.75	445238.44	201566.67	445345.72
356583455	2002-03-01 -08-00-00	2002-03-01 -08-05-00	201566.67	445345.72	201809.84	445410.08
356583455	2002-03-01 -08-05-00	2002-03-01 -08-10-00	201809.84	445410.08	201888.51	445188.38
356583455	2002-03-01 -08-10-00	2002-03-01 -08-15-00	201888.51	445188.38	201974.33	444973.82
356583455	2002-03-01 -08-15-00	2002-03-01 -08-20-00	201974.33	444973.82	202053.00	444755.69
356583466	2002-03-01 -08-50-00	2002-03-01 -08-55-00	201005.24	444345.34	201083.56	444980.21
356583466	2002-03-01 -08-55-00	2002-03-01 -09-00-00	201083.56	444980.21	201153.42	445201.23

3.4 VLTS server

The operation procedure of VLTS server is as suggested in Figure 2. For system communication, VLTS server has to be fully operational and in standby mode for connection. When vehicle position related information is received from an external data provider, it will go through Data Receiver and Storage Module to be stored in the database. When a query is requested from Mobile Interface, a proper query processing module will be invoked to handle the query request and return the query result.

Figure 2. VLTS serverThe operation process of the VLTS server



3.4.1 Operation processing module

Operation Processing Module calls proper execution modules for query requests. Executable operators are as suggested in Table 1 in section 3.2. The database used for the operation algorithm is based on the FleetObject, FleetCurrent, FleetHistory schema suggested in section 3.3.

Figure 3. Operation Processing Algorithm

```
Algorithm SobjSTraj(name, t)
입력 => name : 임의의 이동 차량 mv의 이름, t : 검색하고 싶
        은 질의 시점
출력 => x : t 시점의 좌표 값, y : t 시점의 좌표 값
Begin
   FleetObject에서 입력된 name을 갖는 차량의 CarId를 검색
   If (검색 결과가 null이 아니면) Then
      current_time ← FleetCurrent에서 CarId를 갖는 차
                     량의 time 속성 검색
      If (time = current_time) Then
        x ← FleetCurrent에서 CarId를 갖는 차량의 x 속성
            검색
        y ← FleetCurrent에서 CarId를 갖는 차량의 y 속
            성 검색
      Else If (time < current_time) Then
          x ← FleetHistory에서 CarId와 time 속성을 갖는
튜플의값 x 검색
          y ← FleetHistory에서 CarId와 time 속성을 갖
는 튜플의값 y 검색
          If (검색 결과가 null 이면) Then
              past_location(CarId, t)모듈 호출한 후, x, y값 구함
                //past_location : 과거 시점의 불확실한 위치
                추정 모듈
          Else future_location(CarId, t)모듈 호출한 후, x, y
          값 구함
                // future_location : 미래 시점의 불확실한 위
                치 추정 모듈
   return x and y // 이동 차량의 t 시점의, x, y 좌표 값 반환
End
```

Figure 3 is an implementation algorithm of SobjTraj that tracks the position of a moving vehicle at a specific time point. The algorithm evaluates the past, present and future position values of the target vehicle existing in FleetObject, by comparing the input time values. In each Operation Processing Module, Location Prediction Module is called to deduce the result for the past and the future position queries.

3.4.2 Location prediction module

Location Prediction Module evaluates and returns the positional coordinates at a specific time point, not recorded in the database. Linear interpolation is used in the module to deduce the past and the future location. When the time point the database is updated with is $\{t_1\}_{i=0}^{now}$, for an arbitrary past time point that satisfies $t_i < t_p < t_{i+1}$, a mathematical function t_p is generated with the coordinates of (t_i, x_{ti}) and (t_i, x_{ti+1}), and another function $x(t_p)$ is generated with the coordinates of $x(t_p)$ and $y(t_p)$. Then the location value at the past time point is deduced from the functions and $x(t_p)$ and $y(t_p)$.

Figure 4. Past Position Deduction Algorithm

Algorithm past_location(CarId, t_p)
입력 => *CarId* : 임의의 이동 차량 *mv*의 식별자, t_p : 과거의 특정 시점
출력 => x_{t_p} : 시점 t_p시 *mv*의 *x* 좌표 값, y_{t_p} : 시점 t_p시 *mv*의 *y*좌표 값
Begin
FleetObject에서 입력된 *CarId*를 갖는 차량 검색
If (검색 결과가 null이 아니면) Then
FleetHistory에서 $t_i < t_p < t_{i+1}$의 조건을 만족하는 위치 좌표 쌍
(t_i, x_{t_i}), (t_i, y_{t_i})와 $(t_{i+1}, x_{t_{i+1}})$, $(t_{i+1}, y_{t_{i+1}})$을 검색
If (검색 결과가 null이 아니면) Then

$$x_{t_p} \leftarrow \frac{x_{t_{i+1}} - x_{t_i}}{t_{i+1} - t_i}(t_p - t_i) + x_{t_i}$$

$$y_{t_p} \leftarrow \frac{y_{t_{i+1}} - y_{t_i}}{t_{i+1} - t_i}(t_p - t_i) + y_{t_i}$$

Else x_{t_p} ← 오류 값, y_{t_p} ← 오류 값
Return x_{t_p} and y_{t_p} // 과거의 t_p 시점의 (x, y) 좌표 값 반환
End

Figure 5. Future Position Prediction Algorithm

Algorithm future_location(CarId, t_f)
입력 => *CarId* : 임의의 이동 차량 *mv*의 식별자, t_f : 미래의 특정 시점
출력 => x_{t_f} : 시점 t_f 시 *mv*의 *x* 좌표 값, y_{t_f} : 시점 t_f 시 *mv*의 *y*좌표 값
Begin
FleetObject에서 입력된 *CarId*를 가지는 차량 검색
If (검색 결과가 null이 아니면) Then
FleetHistory에서 $(t_{now-1}, x_{t_{now-1}})$, $(t_{now-1}, y_{t_{now-1}})$와 $(t_{now}, x_{t_{now}})$, $(t_{now}, y_{t_{now}})$ 좌표 검색
If (검색 결과가 null이 아니면) Then

$$x_{t_f} \leftarrow \frac{x_{t_{now}} - x_{t_{now-1}}}{t_{now} - t_{now-1}}(t_f - t_{now}) + x_{t_{now}}$$

$$y_{t_f} \leftarrow \frac{y_{t_{now}} - y_{t_{now-1}}}{t_{now} - t_{now-1}}(t_f - t_{now}) + y_{t_{now}}$$

Else x_{t_f} ← 오류 값, y_{t_f} ← 오류 값
Return x_{t_f} and y_{t_f} // 미래의 t_f 시점의 (x, y)좌표 값 반환
End

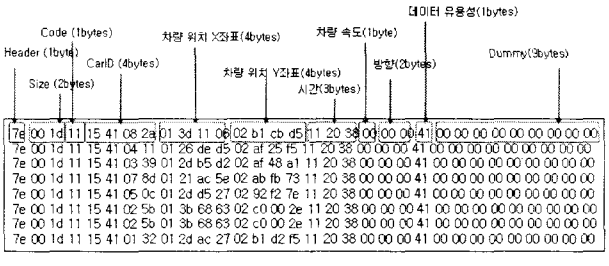
The algorithm of the operation past_location accepts CarID and an arbitrary past time point of a mobile object as input. These input values will be used to search for the vehicle with CarID in FleetObject relation. If such a vehicle exists, the location coordinates in the closest past record in FleetHistory relation, and the location coordinates in the next record after the closest past record will be retrieved. The result obtained from the operation with these coordinate values is returned.

For the future time point t_f that satisfies the conditions $t_{now} < t_f$, a mathematical function is obtained from the coordinates pair of $(t_{now-1}, x_{t_{now-1}})$ and $(t_{now}, x_{t_{now}})$. Using the function $(t_{now-1}, y_{t_{now-1}})$ and $(t_{now}, y_{t_{now}})$, the future position value is predicted.

3.4.3 Real-time location information receiver and storing module

The location information of a moving vehicle transmitted from a real-time position provider consists of data packets with the structure shown in Table 5. The data packet structure, containing only the location information to be stored in the database, designed in this paper, is as shown in Figure 6.

Figure 6. Real-time vehicle position data packet



Real-time displacement information of a moving vehicle with the structure shown in Figure 6 is transmitted to Data Loader Module at regular time intervals. Data Loader stores the transmitted location information, pipelined through Data Conversion Module and Storage Module.

In Figure 7, data_translator converts part of data packets reAlgorithm data_translatorceived from Data Loader, that is to be actually stored in the database, into corresponding data format. Then, the converted data get stored in the database through *data_store* module. *Data_store* module inserts a new tuple in FleetHistory relation. If a data tuple with *data_CarID, X, Y, Time store* exists in FleetCurrent relation, *X, Y, Time* property will be renewed. Otherwise, a new tuple with values of *CarID, X, Y, Time* will be inserted.

3.5 Mobile interface

The operation process of Mobile Interface follows the order shown in Figure 8.

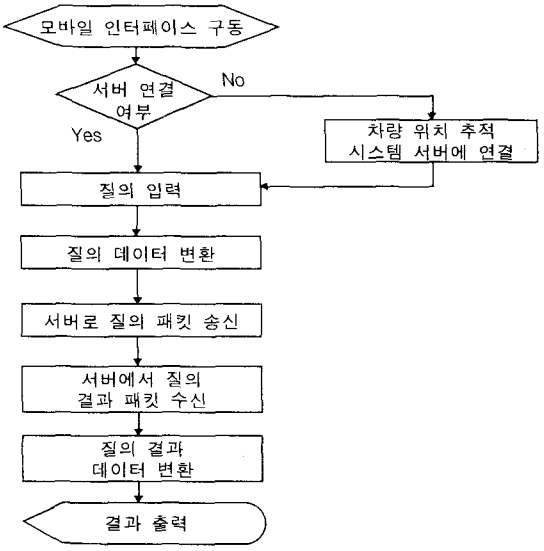
In order for the implemented VLTS to function, socket communication between Mobile Interface and the server has to be enabled first. With the server in operation, when executed, Server Connection Module of Mobile Interface connects to the server in standby mode, enabling TCP/IP socket communication. Thereafter, input query data, entered by the user using Query Data

Figure 7. conversion and storage algorithm of location information

```
Algorithm data_translator ( packet)
입력 => packet : 버퍼 형태로 구성된 패킷 데이터 배열
Begin
   CarId ← packet 배열 중 CarId 부분인 4번째 값을 정수로 변환
   x ← packet 배열 중 x 좌표 값을 나타내는 5번째 값을 실수로 변환
   y ← packet 배열 중 y 좌표 값을 나타내는 6번째 값을 실수로 변환
   t ← packet 배열 중 t 값을 나타내는 7번째 값을 문자형으로 변환
   data_store( CarId, t, x, y) 모듈을 호출하여 실행
End
Algorithm data_store( CarId, t, x, y)
입력 => CarId : 차량 식별자, t : 시간 값, x : x좌표 값, y : y좌표 값
Begin
   FleetHistory에 CarId, t, x, y를 가지는 새로운 튜플 삽입
   FleetCurrent에서 CarId를 가지는 객체 검색
      If (검색 결과가 null이 아니면) Then
         CarId를 가지는 튜플의 t, x, y 속성 값을 갱신
      Else
         FleetCurrent에 CarId, t, x, y를 가지는 새로운 튜플 삽입
End
```

Input Module get transmitted to the server, after conversion into data packets in Data Conversion Module,. The query result processed in the server is received in data packets from Data Receiver Module of Mobile Interface. Data Conversion Module converts the received result into a displayable format of Output Module Screen and the user can access the query result in text format.

Figure 8. Operation process of Mobile Interface



3.5.1 Server connection module

Server Connection Module is the first part of Mobile Interface to be executed to transmit a query request to the VLTS server.

Figure 9. Server Connection Algorithm

```
Algorithm ServerConnect()
Begin
 If (이 Null이 아니면) Then
  화면에 접속 에러 메시지를 표시한다.
 Else
  함수를 호출하여 소켓을 생성한다.
  함수를 사용하여 서버에 연결한다.
 if (서버 연결에 실패한 경우) Then
  함수를 호출하여 소켓 생성을 해제시킨다.
  접속 실패 메시지를 화면에 표시한다.
End
```

Figure 9 shows the server connection algorithm; when Server Module is in operation, server connection algorithm connects to the server using socket connection, and generates error messages when the connection attempt fails or a connection is already established. Because Mobile Interface is developed with Microsoft Embedded Visual C++ 4.0, it is implemented with Microsoft Foundation Class as a base. The socket related functions in the server connection algorithm are implemented with MFC supplied socket classes as well. When socket class is not Null, an error occurs because a reconnection is attempted although a socket is already open. When a connection is established with the server, create() function is called to create a socket, and connect() function with parameters, IP address, and Port Number connects to the server. When connection fails, close() function is called to release the socket, and an error message is displayed.

3.5.2 Data transceiver / Receiver

Data Transceiver Module consists of Query Transmitter and Result Receiver. Figure 10 shows transmitter algorithm that sends string format data from Data Conversion Module to the server, and receiver algorithm that transfers the query result received from the server to Display Module.

Figure 10. Data Transceiver Algorithm

```
Algorithm Packet
입력 => : 패킷 형태로 변환된 데이터
Begin
if (질의 전송 일 경우) Then
  if (Server에 연결되지 않았을 경우) Then
    접속 에러 메시지 표시
  Else
   함수 호출하여 서버에 질의 패킷을 전송한다.
Else (질의 결과 수신 일 경우)
 함수를 호출하여 서버로부터 결과 패킷을 전송 받음
End
```

Data Transceiver Module accepts as input values, the data packets converted from the query data in Data Conversion Module and the result data received from the server in data packets. Then the current packet is classified into either query transmission packet or result reception packet. For query transmission, server connection is checked to rule out the errors of data transmission without an established connection. If a connection is established, Send() function is called to send the data packets to the server. For query result reception, Receive() function is called to receive the packets.

3.5.3 Data conversion module

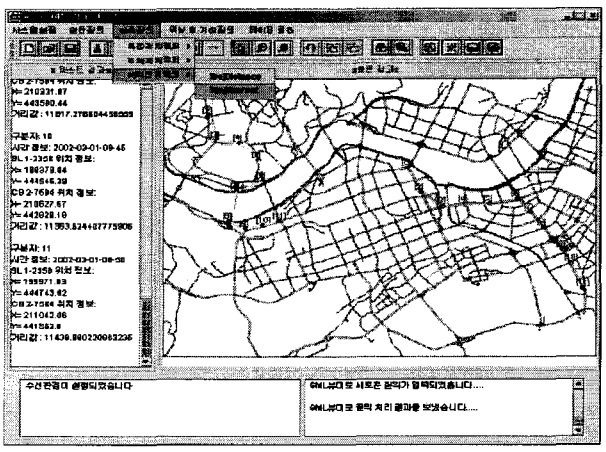
Data Conversion Module converts the query data entered from the user interface of Mobile Interface into packets, to be submitted to the VLTS server. Data Conversion Module also converts the result data packets received from the server into a format that could be displayed on the user screen. Figure 11 shows an algorithm that converts the data passed from the data input module into a format that is suitable for packet transmission and another algorithm that converts the data packets received from the server into a displayable format of user interfaces.

TransSendData() function combines with data and converts the data transferred from each Data Input Module into data packets. data is a query type to classify each queries. Again, separators are added between each data because these data have to be separated at the server. TransReceiveData() function converts the query results in data packet form transferred from the server into a displayable format of the mobile interface screen. The server-transmitted data get separated at separators, and get displayed on the screen with data identifier added.

Figure 11. Data Conversion Algorithm

```
Algorithm TransSendData(Tname, CarId, Ts, Te)
입력 => Tname : 테이블 명, CarId : 차량 번호
        Ts : 유효 시작 시간, Te : 유효 종료 시간
Begin
    (String) Packet변수에 Tname값과 구분자 할당
    (String) Packet변수에 CarId값과 구분자 추가 할당
    (String) Packet변수에 Ts값과 구분자 및 Te값과 구분
             자 추가할당
    (String) Packet변수에 질의 타입인 Type값을 할당
    Packet(Packet) 함수를 호출하여 변환된 데이터 전송
end
Algorithm TransRecieveData
입력 =>Packet : 서버에서 전송된 질의 결과 패킷 데이터
Begin
 for (Packet의 길이)
  if (Packet이 구분자인 경우)
    ListBox에 출력하지 않고 통과
  Else
    ListBox에 식별자를 붙여 출력.
End
```

Figure 12. Result of TrajNearest operation

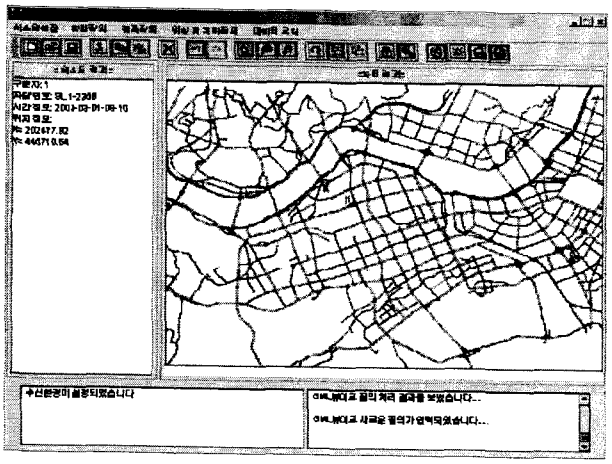


Ⅳ. Implementation

VLTS is implemented with Pocket PC and JDK 1.3, and Microsoft SQL server 2000 is used for DBMS. Mobile Interface is implemented with a PDA that uses Microsoft Windows CE 2002 as an OS. Pocket PC based applications can be implemented with such tools as Microsoft Embedded Visual Basic and Microsoft Embedded Visual C++ 4.0. Mobile Interface implemented in this paper is developed with EVC 4.0. Mobile Interfaces were tested on PDAs with Pocket PC OS and Desktop Pocket PC emulators. The VLTS server was implemented with

that could be operated platform-independently. Tests of the server module were done on Windows 2000 using JDK 1.3 environment.

Figure 13. Result of PositionAtTime operation



4.1 Processing operation of VLTS server

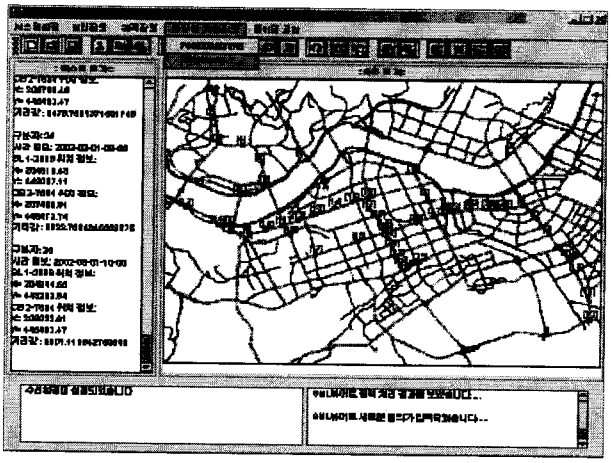
In this section, instances of TrajNearest, PositionAtTime, and MDistance operations are presented, with query examples. VLTS server has a separate program that could accept query request on its own, apart from Mobile Interface. The server program can request all the query types Mobile Interface can. Figure 12 shows an example of TrajNearest operation result that retrieves "the trajectory of the closest vehicle from the trajectory of an arbitrary moving vehicle." The query example used in this case is "retrieve the trajectories of the vehicles that are closest from the vehicle, SL 1?2358, between 8:00 am, March, 1, 2002 and 08:50AM March, 1, 2002."

From the user interface of the VLTS server program, table name, vehicle identification number, starting and ending time to be queried are entered. The entered query data are passed into Operation Processing Module via Query Classification Module. Operation Processing Module searches through the related location data in the database. The search result will be displayed in text and graphics format on the user interface as shown in Figure 12.

PositionAtTime retrieves the location information of a moving vehicle at a specific time point and the example here shows performing of the query, "retrieve the location information of SL?2358 at 9:10 am, March, 1, 2002. The example is

illustrated in Figure 13.

Figure 14. Result of MdISTANCE OPERATION JAVA

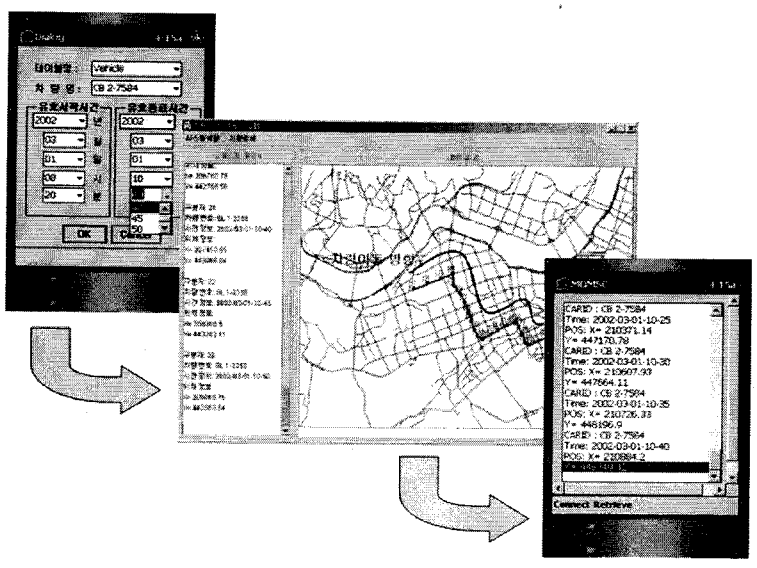


Using the query input interface the same way as with TrajNearest operation, table name, vehicle identification number and specific time are entered. The server program performs operations with the input data and the result will be diplayed on the user interface. The vehicle identification number, queried time period, and location coordinates will be diplayed in text format, and the location will be graphically displayed on a road map.

Finally, Figure 14 shows the outcome of the MDistance operation evaluating the distance between arbitrary two vehicles in motion. This example performs the query "retrieve the distance between the vehicle SL 1-2358 and CB 2-7584 from 8:00 am, March, 1, 2002 and 10:00 am, March, 1, 2002"

The input data for the query are table name, vechile identification numbers to be queried, and starting and ending time. The query results with the input data are shown in Figure 14. The text format results display the time information, location information and displacement data between the vehicles, The locations of the vehicles will be graphically displayed on the map as well.

Figure 15. Retrieval of the position of a specific vehicle over specific time period



4.2 Vehicle location retrieval of mobile interface

In order to perform a vehicle location retrieval from Mobile Interface, the VLTS server has to be run first. Then, Mobile Interface Is run and connects to the server. Query data are entered through the query menu and submitted to the VLTS server. The query result Is displayed in the list box of the client in text format. In this paper, an example of performing query function for " the location of all vehicles over the entire time period" is shown. Figure 15 shows the procedure of performing the query function for the location of a specific vehicle in a specific time period. The example performs the query "track the location of CB 2?3584 between 8:20am, March 1, 2002 and 10:40 am , March, 1, 2002"

When query function is selected from the mobile query menu of Mobile Interface, and table name, vehicle name, starting time and ending time for the search are entered, query data are submitted. The VLTS server processes the query request and displays the result in text and graphics format and sends the result to the client. The mobile interface receives and displays the search result in the list box in text format.

Figure 16. Performing location queries of all vehicles over the entire time period

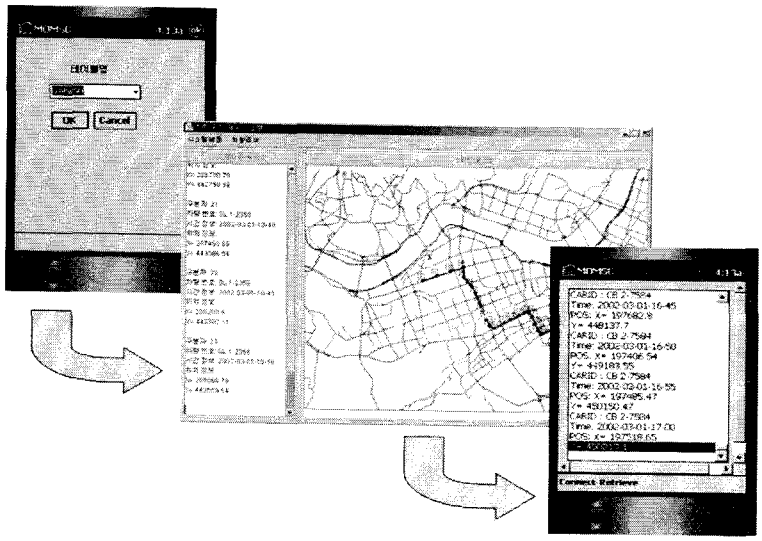


Figure 16 shows the procedure of performing the query function of all vehicles over the entire time period. The example performs the query 'track the locations of all vehicles existing in the database'.

The server processes the queries when the type of query is selected and the name of the table is entered on Mobile Interface, in the same way as in the query for the location of a specific vehicle in a specific

time period. The VLTS server will display the query result in text format and graphical trajectory format, and Mobile Interfaces will display the results in text format only.

V. Conclusion

Existing vehicle positioning systems fail to present ways to estimate uncertain past and future position of a moving vehicle not stored in the database. Also, because existing systems mainly provide services on wired network, mobile clients on wireless network can't access the real-time search functions of the tracking systems. For these reasons, a new real-time vehicle positioning system utilizing wireless mobile interfaces such as PDA has been designed and proposed in this paper. The proposed system consists of Vehicle Location Tracking System server and Mobile Interface. The VLTS server stores, keeps track of, and manages the past and the future location information of vehicles and provides Mobile Interface search results of vehicle position queries. Mobile Interface requests the server for real-time position information of vehicles, and displays the search results.

In order to implement such a system, vehicle location information has been modeled, and a system architecture and algorithms to handle the information have been suggested. The search functions of the implemented systems are, location queries of all vehicles over the entire time period, location queries of all vehicles in a specific time period, location queries of a specific vehicle over the entire time period, location queries of a specific vehicle in a specific time period, and location queries of a vehicle at a specific time point. It has been established and confirmed that the suggested search functions are behaving and functioning properly. Furthermore, it was established in this paper that the inquiries of the past and the future locations of moving clients could be processed in real time in mobile environment, by demonstrating the implemented outcome of the proposed system simulated for hypothetical situations.

The proposed system is in development as a part of e-Logistics system of postal distribution. The mobile interface designed in this paper has limited capability of text output and lacks the means to produce visual output of the search results. Therefore, the future research will be directed toward the expansion of the system to implement electronic maps as search results on the output module of the mobile interface.

References

1. 교통개발연구원, 한국통신, "첨단 화물운송시스템(CVO) 기본설계", 1997.
2. 김종혁, "첨단 교통관리 시스템", 정보과학회지, 제16권 제6호, pp.5-13, 1998.
3. 김형욱, 김성수, "물류 정보서비스를 위한 통신 매체와 이를 이용한 교통 물류관제 시스템의 전반적인 이해와 적용", 한국통신 Technical Memo, 1997.
4. 안승범, "안전도향상을 위한 첨단 화물운송시스템(CVO)의 서비스와 기술", 정보과학회지, 제16권 제6호, pp.30-35, 1998.
5. 이승룡, 홍영래, 김형일, 배수강, 최대순, "첨단 대중교통 시스템", 정보과학회지, 제16권 제6호, pp.23-29, 1998.
6. M. Erwig, R. H. Guting, M. Schneider, and M. Vazirgiannis, "Abstract and Discrete Modeling of Spatio-Temporal Data Types," Chorochronos Technical Report, CH-98-14, 1998.
7. M. Erwig, R. H. Guting, M. Schneider, and M. Vazirgiannis, "Spatio-Temporal Data Types : An Approach to Modeling and Querying Moving Objects in Databases," GeoInformatica, Vol.3, No.3, pp.269-296, 1999.
8. L. Forlizzi, R. H. Guting, E. Nardelli, and M. Schneider, "A Data Model and Data Structures for Moving Objects Databases," In Proc. of the ACM SIGMOD Conf., pp.319-330, 2000.
9. R. H. Guting, and et. al, "A Foundation for Representing and Querying Moving Objects," ACM Transactions on Database Systems, Vol.25, No.1, pp.1-42, 2000.
10. O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang, "Moving Objects Databases : Issues and Solutions," Proc. of the 10th International Conference on Scientific and Statistical Database Management, SSDBM'98, Capri, Italy, pp.111-122, 1998.
11. 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.
12. 류근호, 안윤애, 이준욱, 이용준, "이동객체 데이터베이스와 위치기반 서비스의 적용", 데이터베이스연구회지, 제17권 제3호, pp.57-74, 2001.
13. 장승연, 정영진, 양은주, 안윤애, 류근호. "관계 데이터베이스를 이용한 이동 객체 처리기의 구현", 한국정보처리학회 지식 및 데이터공학연구회, pp.205-211, 2001.
14. 안윤애, 류근호, 김동호, "차량 위치 추적을 위한 이동 객체 관리 시스템의 설계", 정보처리학회논문지D, 제9-D권 제5호, 2002.
15. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Modeling and Querying Moving Objects," Proc. of the 13th International Conference on Data Engineering, ICDE'97, Birmingham, UK, Apr., 1997.
16. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Querying the Uncertain

Position of Moving Objects," Springer Verlag Lecture Notes in Computer Science number 1399, pp.310-337, 1998.

17. O. Wolfson, P. Sistla, B. Xu, J. Zhou, S. Chamberlain, N. Rishe, and Y. Yesha, "Tracking Moving Objects Using Database Technology in DOMINO," Proc. of NGITS' 99, The 4th Workshop on Next Generation Information Technologies and Systems, Zikhron-Yaakov, Israel, pp.112- 119, 1999.
18. O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez, "Cost and Imprecision in Modeling the Position of Moving Objects," Proc. of the 14th International Conference on Data Engineering, ICDE' 98, Orlando, FL, Feb., 1998.
19. S. Saltenis, C. S. Jensen, S. Leutenegger, and M. Lopez, "Indexing the Positions of Continuously Moving Objects," Proc. of the ACM SIGMOD Conference, pp.331-342, 2000.
20. D. Pfoser, Y. Theodoridis, and C. S. Jensen, "Indexing Trajectories of Moving Point Objects," Chorochronos Technical Report, CH-99-3, Oct., 1999.
21. D. Pfoser, C. S. Jensen, and Y. Theodoridis, "Novel Approaches in Query Processing for Moving Objects," Proc. of the VLDB Conference, pp.395-406, 2000.
22. D. Pfoser and C. S. Jensen, "Capturing the Uncertainty of Moving Object Representations," Proc. of Advances in Spatial Databases, 6th International Symposium, SSD' 99, pp.20-23, 1999.
23. D. Pfoser and N. Tryfona, "Fuzziness and Uncertainty in Spatiotemporal Applications," Chorochronos Technical Report, CH-00-4, Feb., 2000.
24. I. B. Oh, Y. A. Ahn, E. J. Lee, K. H. Ryu, and H. G. Kim, "Prediction of Uncertain Moving Object Location," Proc. of International Conference on East-Asian Language Processing and Internet Information Technology, EALPIIT' 02, pp.51-58, Jan., 2002.
25. S. S. Park, Y. A. Ahn, and K. H. Ryu, "Moving Objects Spatiotemporal Reasoning Model for Battlefield Analysis," Proc. of Military, Government and Aerospace Simulation part of ASTC' 01, pp.108-113 2001.
26. K. H. Ryu and Y. A. Ahn, "Application of Moving Objects and Spatiotemporal Reasoning," TimeCenter TR-58, 2001.