韓國컴퓨터情報學會 論文誌
第16卷 第9號, 2011. 9.

2011-16-9-4-1

스마트 이동객체의 App 기반 능동형 재해경보서비스 시스템

한 기 태*

Active Disaster Alerting Service System based on App of Smart Moving Object

Gi-Tae Han*

요 약

기존의 위치기반 경보서비스는 서버에서 각 이동객체의 위치를 확인하는 방법을 사용함으로 인해 서버의 부하문제가 대두 되었다. 본 논문에서는 스마트 이동객체에 앱(App)을 탑재하여 필요시에만 자신의 위치정보를 가지고 서버에 접근하는 방식의 ADAS(Active Disaster Alert Service)라는 새로운 알고리듬을 제안하고, 사용자 시각화를 고려한 실시간 재해경보서비스 시스템을 구현하였다. 제안하는 방법은 App의 구동시 GPS 모듈 혹은 이동통신망에 의해 획득한 이동객체의 현재 위치좌표와 ID를 서버로 전송하면, 서버에서는 이동객체의 현재 위치좌표와 재해정보데이터베이스(Disaster Information Database: DIDB)에 등록된 재해지역 좌표들을 비교하여 현재 위치로부터 가장 가까운 거리에 존재하는 n개의 NDI(Near Disaster Information)를 얻어 클라이언트로 전송한다. App에서는 이동객체의 현재 위치좌표와 서버에서 보내온 NDI의 재해좌표들을 실시간으로 비교하여 경보수준을 위험, 주의, 안정으로 분류한 서비스를 진행하고, 구동 중에 주기적으로 서버의 DIDB에 접근하여 최신의 NDI를 얻는다. 그러므로 제안한 방법은 이동객체의 모든 이력을 서버에서 관리하는 기존 경보서비스에 비하여 서버부하 문제를 획기적으로 줄일 수 있는 방법이 될 것이다.

▸ Keyword: 위치기반, 앱, 능동형재해경보서비스, 위성항법장치, 재해정보데이터베이스, 근거리재해정보

Abstract

Previous alerting service based on LBS was caused severe overload problem of server by using the method to confirm the location of each moving object on server. In this paper, by loading an App on smart moving object, we proposed a novel algorithm named ADAS(Active Disaster Alert

• 제1저자 : 한기태 • 교신저자 : 한기태
• 투고일 : 2011. 07. 07, 심사일 : 2011. 07. 18, 게재확정일 : 2011. 07. 25.
* 경원대학교 인터랙티브미디어학과 교수(Dept. of Interactive Media, Kyungwon University)
※ 이 연구는 2011년 경원대학교 지원에 의한 결과임.

Service) for accessing to the server site with oneself location information as needed and implemented the disaster alerting service system with visualization for user. In the proposed method, running App access to the server periodically with the present location coordinate gained from GPS module or network module and the ID of moving object. Then, the server compare the present location coordinate of moving object and the coordinates of disasters registered in DIDB and transmit the n NDIs existed in near distance orderly from the coordinate of present moving object to the client. The App compares the coordinate of present location for moving object and the coordinates of NDI is transmitted from server by real time and executes the service with classifying levels of alert into three steps such as danger, carefulness and safety. And new NDIs are gained by accessing DIDB on Server periodically during running App. Therefore, this will be become a novel method for reducing fundamentally the server overload problem in comparison with previous alerting service that the career of moving object is managed on server.

▸ Keyword: LBS, App, ADAS, GPS, DIDB, NDI

Ⅰ. 서론

이동객체의 위치를 기반으로 하는 서비스는 정보산업사회의 많은 분야에서 이미 활용되고 있거나 새로운 서비스 모델들의 개발을 시도하고 있다. 위치기반 경보서비스는 이동통신 네트워크상에서 단말기 사용자의 위치를 모니터링하여 특정한 지역이나 설정된 영역에 진입, 존재, 퇴거 등의 이벤트가 발생될 경우 사용자에게 SMS 등을 통하여 알리거나, 사용자에 의해 미리 정의된 특정 서비스를 제공하는 서비스이다[1].

위치기반 서비스의 형태에는 위치기반 광고서비스, 재난·재해경보서비스, 물류관재서비스, L-Commerce, 교통정보서비스 등을 들 수 있는데, 최근에는 스마트 이동객체의 등장으로 새로운 위치기반 관련 서비스의 모델들이 개발되고 있다. 위치기반 경보서비스를 제공하기위해서는 이동객체에 대한 지속적인 위치획득이 필요한데, 위치기반 경보 서비스의 경우에는 대상 이동객체의 수가 증가하면 증가할수록 서버 시스템의 성능은 심각하게 저하될 수 있다. 예를 들어 대상 이동체가 500만개이고 위치획득시간이 30초라면, 하루 동안의 총 위치획득회수는 144억(5,000,000 * 2880)번으로 실제의 이동통신 시스템 환경에서는 적용하기 어려운 수치라고 할 수 있다. 그러므로 기존 연구에서는 이동객체의 이동패턴을 파악하여 동적으로 위치획득시간 간격을 조절함으로써 동일한 경보서비스 수준을 유지하면서도 이동객체에 대한 위치획득 횟수를 줄이기 위한 연구를 다각적으로 진행해왔다[1,2]. 그동안 서버 상에서 위치획득의 횟수를 줄일 수 있는 다양한 방법들이 연구되었지만, 이러한 알고리듬의 모듈들이 여전히 서버시스템에서 구동되기 때문에 이동객체의 증가와 정확성 유지를 위한 모니터링 주기가 짧아진다면 여전히 시스템 부하 문제는 남아 있게 된다.

제안한 방법에서는 위치기반 재해경보서비스 App을 개발하여 스마트 이동객체에 탑재함으로써 지속적인 실시간 위치확인 기능을 서버에 두지 않고 스마트 이동객체에서 능동적으로 객체의 이동상황에 따른 정확한 재해경보서비스를 실시간으로 서비스하는 방법을 제안하고 구현한다. 이 방법은 스마트 이동객체에 탑재된 App을 구동하여 이동객체가 능동적으로 자신의 위치를 획득하며, App의 구동시점과 일정시간의 경과나 일정거리의 이동시점에서만 서버로의 접근을 시도하고, 서버의 DIDB로부터 얻은 이동객체의 위치로부터 가장 가까운 거리에 있는 재해경보인 NDI를 App의 멤버변수에 저장하여, 일정기간 혹은 일정거리 이동 동안 App이 독자적으로 스마트 이동객체의 이동에 따른 재해경보서비스를 수행한다.

본 논문에서는 클라이언트인 이동객체가 독자적으로 자신의 위치를 획득할 수 있고 스스로 경보서비스의 시기를 결정할 수 있는 상태를 능동형(activity)이라고 기술하고, 클라이언트 자체적으로 자신의 위치를 획득할 수 없고 스스로 경보서비스의 시기를 결정할 수 없어 서버가 관여하는 상태를 수동형(passivity)으로 기술한다. 2장에서는 관련 연구로 기존의 수동형 경보서비스 모델들(PDAS: Passive Disaster Alerting Service)에 대하여 살펴보고, 3장에서는 본 논문에서 제안하는 App 기반 능동형 재해경보서비스 모델(ADAS: Active Disaster Alerting Service)을 살펴본다. 4장에서는 제안한 ADAS 모델이 기존의 PDAS 모델의 어떤 문제점을 해

결 할 수 있는 지를 살펴보고, 마지막으로 연구의 결론 및 향후 연구방향을 제시한다.

II. 관련 연구

이동객체는 이동하면서 시간이 지남에 따라 위치나 모양이 바뀌는 객체를 의미하며[3], 이 이동객체의 정보를 효과적으로 파악하고 알리는 방법을 다루는 영역을 위치기반 서비스라고 한다. 위치기반 서비스를 위해서는 위치관리 작업이 중요하며, 이동통신 분야에서는 위치를 계산하는 방법과 시스템의 오버헤드를 최소화하면서 효율적으로 위치를 관리하기위한 페이징과 위치보고에 관한 연구가 주요한 분야를 차지하고 있다[4]. 페이징은 통신망이 이동객체의 위치를 찾는 과정을 말하고 위치보고는 페이징을 용이하게 하기위해 이동객체가 자신의 위치를 통신망에게 보고하는 것을 말한다[2]. 기존의 위치기반 시스템은 서버 측에 위치기반 서비스 서버와 위치획득 서버를 두고 있는데, 이와 같은 시스템 구조는 서비스대상 이동객체의 수가 증가하면 증가 할수록 부하증가로 인하여 시스템 성능이 현저히 저하되어 위치를 획득하고 저장하기까지의 시간이 계속 증가할 수 있다는 문제점을 안고 있다. 그러므로 어느 정도의 시간간격을 가지고 이동객체의 위치를 획득하느냐에 따라서 시스템의 성능이 좌우되기 때문에 위치획득에 대한 다양한 모델들이 그동안 연구되어 왔다.

지금까지 연구된 위치획득 모델들은 이동객체 데이터베이스와 이동통신망 사이의 통신부하를 최소화하기 위해 이동객체의 이동패턴을 파악하고 이동객체의 위치획득 간격을 동적으로 조절하여 불필요한 위치정보를 획득하지 않도록 함으로써 위치획득 횟수를 줄이도록 하는 방법들이 대부분을 차지하고 있다[1].

1. 경보서비스를 위한 기존의 위치획득 방법

이동통신망에서의 위치관리 작업은 페이징과 위치보고로 이루어지며, 이들 간에는 trade-off 관계가 존재하고, 효율적인 위치관리를 위해서는 사용자의 이동패턴을 고려하여 설계되어야 한다. 위치보고 방법에는 시간간격을 정하여 그 시간 간격이 지나면 다시 위치를 보고하는 time-based location update[5], 이동거리 threshold를 정해놓고 위치보고를 한 LA(location area)[2]에서 정해진 거리만큼 이동이 되었을 때 위치를 보고하는 distance-based location update[6], LA로의 이동회수 threshold를 정해놓고 위치보고를 한 후 정해진 회수만큼의 다른 LA로 이동 되었을 때 위치보고를 하는 movement-based location update[7] 등이 있다.

지금까지 연구된 경보서비스를 위한 위치획득방법들의 대표적인 것으로는 정적 위치획득모델, MATT 위치획득모델, 이동거리비율 위치획득모델과 이들을 조합한 모델들이 소개되었다[1,2,8].

정적 위치획득모델은 모든 이동객체들에게 위치획득시간 간격을 일정하게 두어 위치를 획득하는 방법으로 획득시간 간격이 커지면 위치서버의 부하는 줄어들지만 신뢰도가 감소하고 획득시간 간격이 작아지면 신뢰도는 증가하지만 위치서버의 부하가 증가하는 문제점을 갖고 있어 이동객체의 수가 많은 서비스에는 적합하지 않는 모델로 인식되고 있다. MATT(Minimum Alert Triggering Time) 위치획득모델은 모바일 사용자의 최대 이동속도와 현재위치에서 가장 가까운 경보지역까지의 거리를 이용하여 MATT를 계산하고, 모바일 사용자가 최소한 MATT 동안은 가장 가까운 경보지역에 진입하지 않을 수 있다는 전제하에 위치획득시간 간격을 동적으로 조절하도록 하는 방법이다.

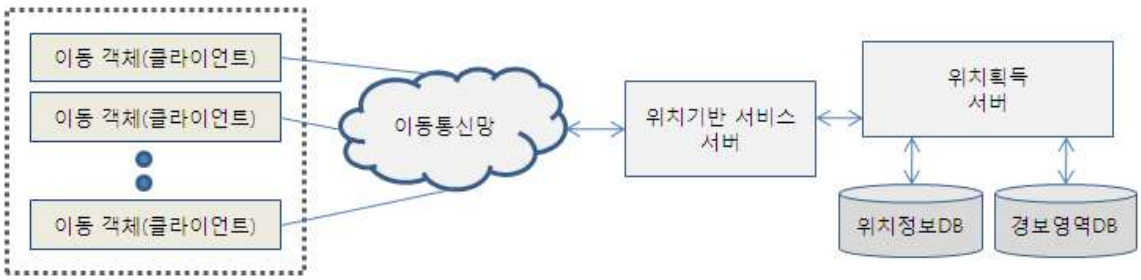

그림 1. 서버상의 위치획득 기반 수동형 재해경보서비스(PDAS) 시스템
Fig. 1. Passive Disaster Alerting Service(PDAS) System based on Location Acquisition on Server

표 4. 기존 서버 위치획득 모델(PDAS)의 문제점
Table 1. Problems of Location Acquisition Model(PDAS) on Server

모 델	문 제 점
정적위치획득	획득시간 간격이 커지면 신뢰도가 감소하고 획득시간 간격이 작아지면 위치서버의 부하가 증가
MATT	모바일 사용자의 최대 이동속도가 큰 상황에서 실제 이동속도가 작을 경우에 적합지 않음
이동거리비율 위치획득	최대 이동속도를 고려하여 위치획득 시간 간격을 계산했을 때 보다 불필요한 위치획득 회수가 발생
하이브리드 위치획득모델	위치획득 조정계수를 구하기 위한 다양한 실험을 통하여 $weight$ 값을 얻을 수 있음

이 MATT는 모바일 사용자의 최대 이동속도를 이용하기 때문에 최대 이동속도보다 매우 느린 속도로 오랫동안 이동하는 경우에는 불필요한 위치획득을 더 자주 하게 되는 문제점을 가지고 있다.

이동거리비율 위치획득모델은 모바일 사용자가 최근에 이동한 거리에 비례하여 위치획득시간 간격을 동적으로 조절함으로서 이동속도 변동 환경에도 적응할 수 있도록 하는 방법이다. 그러나 이 방법은 적절한 위치획득시간 간격을 조절하기 위한 파라미터의 설정이 매우 어렵고 버퍼영역을 사용함으로써 불필요한 위치획득 횟수가 증가하는 문제점을 가지고 있다.

또한 이러한 방법들의 장점을 결합한 하이브리드 위치획득 모델[1]이 등장하였는데, 이 방법은 $weight$ 값에 따라 최근이동거리 비율을 높일 것이냐 아니면 낮출 것이냐를 결정하게 되는데, 적정한 $weight$ 값을 얻기가 쉽지 않아 다양한 실험을 통하여 적정한 $weight$ 값을 얻어야 만하는 어려움을 안고 있다.

2. 기존방법의 문제점 분석

위치기반 서비스는 이동통신망을 이용하여 이동객체의 위치를 실시간으로 파악하여 이를 활용하는 다양한 응용시스템 및 서비스를 통칭하는데, 기존의 재해경보서비스는 서버 상에 존재하는 모바일 이동객체의 이력을 관리하는 데이터베이스로부터 서비스 대상 모바일 이동개체의 위치를 획득하면서 이동객체가 서버에 설정되어 있는 재해경보영역으로 접근할 때 이동객체에게 경보를 보내는 구조로 되어 있다. 이는 이동객체의 수가 증가하면 객체의 위치획득에 대한 서버의 부담이 증가한다.

또한 시간적 주기를 가지고 이동객체의 위치를 획득하는 환경에서 이동객체의 속도가 증가하면 경보서비스를 놓칠 위험이 있고 이동객체의 속도가 감소하면 불필요한 위치획득으로 인한 시스템 자원을 낭비하는 문제가 제기된다. 그림 1은 기존에 사용되고 있는 서버상의 위치획득 기반 PDAS 시스템 구조도이다[1].

앞에서 언급한 기존 서버 위치획득 모델들의 문제점들을 정리하면 표 1과 같다. 표 1에서 살펴본 기존의 모든 PDAS 모델들은 서버에서 클라이언트의 위치를 획득하는 방법을 사용하기 때문에 위치획득 서버의 부하를 줄이기 위한 근본적인 해결방법이 되지 못하고 있다.

III. App기반 ADAS 모델

본 논문에서는 최근 스마트 이동객체의 확산과 사용 환경을 고려하여 효과적인 실시간 재해경보서비스를 구현하기위하여 스마트 이동객체 상에 위치기반 경보서비스 App을 탑재하여 기존 위치기반 서비스 방법의 문제점들을 개선하고자 한다. 본 논문에서 제안한 방법은 App의 구동 시점에 서버의 DIDB에 접속하여 스마트 이동객체의 위치와 근접한 위치의 재해정보인 NDI를 얻고, 클라이언트에서는 스마트 이동객체의 이동에 따라 현재 이동객체의 위치좌표와 NDI를 비교하여 그 결과를 가지고 재해경보서비스를 수행한다.

이 방법에서는 App의 구동 중에 스마트 이동객체의 이동거리와 최근 서버접속 경과시간을 고려하여 클라이언트가 갖고 있는 NDI를 서버의 최신 재해정보로 갱신할 때만 서버에 접근하도록 함으로써 기존 위치기반 경보서비스의 서버 부하 문제를 해결하고, 객체의 이동상황을 실시간으로 반영한 재해경보서비스 시스템을 구현하고 있다. 즉, 스마트 이동

객체 사용자가 스마트 위치기반 경보서비스 App을 구동하면 인터넷(WIFI 혹은 3G 망)을 통하여 서버의 DIDB로 접근하여 현재 위치좌표에서 가장 가까운 영역 순으로 n개의 NDI (위치좌표, 재해종류, 위험반경, 주의반경, 상황개요 등)를 얻어서 App의 멤버변수에 저장하고, App은 멤버변수에 저장된 NDI를 가지고 이동객체의 이동 중의 자신의 위치좌표와 주기적으로 비교하여 자신의 위치가 안정지역, 주위지역, 위험지역 중의 어디에 해당되는지를 맵 상에 시각적으로 표시하는 경보 서비스를 수행한다.

제안한 방법은 이동객체의 위치획득 및 재해경보서비스를 클라이언트 측에서 독자적으로 수행하게 하고, 새로운 재해정보 NDI를 얻기 위해 사용자 설정의 threshold(일정시간과 일정거리)를 만족할 때만 서버로 접속을 시도하도록 하고 있다. 즉, App은 NDI의 정보와 GPS 혹은 네트워크 모듈로부터 얻은 현재 이동객체의 위치정보 및 타이머를 활용하여 경보서비스를 진행하다가 설정된 threshold(객체의 이동거리, 서버접근의 주기)를 넘어설 때 서버에 접근하여 새로운 NDI를 갱신한다.

본 논문에서 스마트 이동객체의 위치기반 재해경보서비스 시스템을 구축하기 위해 서버 측에 전국 규모의 모든 재해가 수록된 DIDB와 접속한 이동객체의 현재상황을 기록하는 SDBMO(State Database of Moving Object)를 구축하고, 클라이언트의 App에는 자기 자신에게 적용할 재해정보인 NDI를 멤버변수에 가지고 있으며, 스마트 이동객체에 탑재되는 재해경보서비스 App은

1) 현재 이동객체의 위치를 얻는 위치획득 모듈,
2) 현재 이동객체위치와 NDI의 재해위치를 비교하여 서비스를 수행하는 재해경보서비스 모듈,
3) 설정된 threshold(일정시간 혹은 일정거리)를 초과할 경우에 새로운 NDI를 갱신하는 NDI갱신 모듈을 포함하며, 또한 서버 측 시스템에는 클라이언트의 요청이 있을 때마다 DIDB로부터 이동객체의 위치에서 가장 가까운 거리에 있는 n개의 재해정보인 NDI를 검색하여 XML형식으로 결과를 클라이언트에 보내주는 재해정보검색 모듈과 이동객체의 상황을 서버에서 관리할 수 있는 DB인 SDBMO에 이동객체의 정보를 등록하거나 갱신하는 모듈들을 둔다.

1. 시스템 구성도

제안한 방법의 시스템구성을 살펴보면 서버 측에 DIDB와 SDBMO를 두고, 관리자는 재해발생시점에 DIDB에 해당 재해정보를 삽입(입력)한 후 게시설정을 ON하며, 재해발생해제시점에 정보게시설정을 OFF로 설정한다. 또한 SDBMO에는 재해경보서비스를 요청한 클라이언트 이동객체의 ID(전화번호 등), 위치좌표, 최근 서버접속시간 및 서비스 ON/OFF 상태를 기록한다. 클라이언트 이동객체는 서버 DIDB로부터 자신의 위치좌표와 비교하여 근거리에 위치한 n개의 NDI를 App의 멤버변수에 가지고 있으면서, 일정시간의 경과 혹은 일정거리의 이동 동안 APP의 멤버변수에 저장된 NDI를 참조하여 경보서비스를 수행한다.

이동객체에 탑재된 App은 GPS 모듈 혹은 네트워크로부터 이동객체의 현재위치를 실시간으로 획득하며, 현재 이동객체의 위치좌표와 NDI가 가지고 있는 재해발생지점의 위치를 비교하여 거리를 계산하고, 계산한 거리와 NDI에 있는 주의반경 혹은 위험반경을 비교하여 경보서비스를 실시하게 된다.

그림 2. App 기반 ADAS 시스템
Fig. 2. Active Disaster Alerting Service System based on App

표 5. DIDB(재해정보 데이터베이스) 구조도
Table. 2. Layout of DIDB(Disaster Information Database)

일련번호 (PK)	재해 종류	재해위치좌표	재해 상황	위험 반경	주의 반경	발생일시	해지일시	개시 설정
재해번호	유형	경도/위도/고도	설명	km	km	일자 : 시각	일자 : 시각	on/off

(PK: Primary Key)

표 6. SDBMO(이동객체 상태 데이터베이스)의 구조도
Table 3. Layout of SDBMO(State Database of Moving Object)

객체 ID (PK)	현재위치좌표	최근 서버 접속 일시	노출재해 (FK)	노출상태
ID (예: 전화번호)	경도/위도/고도	일자 : 시각	재해번호	1:주의 2:위험

(PK: Primary Key, FK: Foreign Key)

표 7. NDI(근접 재해정보) 구조도
Table 4. Layout of NDI(Near Disaster Information)

순위	종류	재해위치좌표	재해상황	주위반경	위험반경	서버접속시각
번호	유형	경도/위도/고도	설명	km	km	일쟈시각

그리고 서버의 DIDB로부터 NDI를 읽어온 후 일정시간이 지나거나 일정거리를 이동하였을 경우에는 다시 서버에 접속하여 DIDB로부터 새로운 NDI를 갱신하는 과정을 수행한다. 또한 서버의 DIDB로부터 정보를 읽기 위하여 접속한 클라이언트의 이동객체 id와 위치좌표, 접속시간, 서비스상태에 관한 정보들을 서버의 SDBMO에 기록하여 재해경보서비스 시스템에 접속한 클라이언트 이동객체들에 대한 상황정보가 통합 관리되도록 시스템을 설계한다. 즉, 재해경보시스템에 접속한 클라이언트 이동객체에 대한 상황정보를 SDBMO에 수록함으로써 효율적인 재해경보 서비스제공 및 긴급구조와 같은 물리적인 지원을 위한 판단 자료로 이용할 수 있게 설계하고 있다.

그림 2는 App 기반 능동형 재해경보서비스의 시스템 구성도이며, 표 2와 3은 각각 DIDB와 SDBMO의 구조도이다.

또한 클라이언트 이동객체에 탑재된 App은 현재위치정보를 가지고 서버에 접속하여 DIDB부터 현재 이동객체의 위치와 가장 근접한 위치에 있는 n개의 NDI를 획득하여 클라이언트로 전송하며, App은 실시간으로 이 NDI를 참조하여 자체 적으로 경보서비스를 수행한다. 표 4는 NDI 정보의 구조를 보이고 있으며, 본 시스템에서는 NDI를 App의 멤버변수에 저장하고 있다.

2. 제안한 ADAS 알고리듬

여기서는 스마트 이동객체 상에 탑재된 App이 DIDB로부터 얻은 NDI를 이용하여 실시간 재해경보서비스를 수행할 수 있는 능동형 재해경보서비스(ADAS) 알고리듬을 제안한다. 본 알고리듬에서 사용한 약어를 정리하면 표 5와 같다.

표 8. 제안한 ADAS 방법에서 사용된 약어
Table 5. A key to an abbreviation represented in proposed method.

약어	약어풀이	의미
PCMO	Present Coordinate of Moving Object	이동객체의 현재 위치좌표
CDL	Coordinate of Disaster Location	재해위치의 좌표
DIDB	Disaster Information Database	재해정보 데이터베이스
SDBMO	State Database of Moving Object	이동객체 데이터베이스
NDI	Near Disaster Information	근거리 재해정보

DIGT	Disaster Information Gain Time	재해정보 획득 시각
AST	Alert Service Time	경보서비스 시각
ASC	Alert Service Coordinate	경보서비스 시행 위치좌표
PTAS	Pass Time after Alert Service	경보서비스 경과시간
PTIG	Pass Time after Disaster Information Gain	재해정보획득 경과시간
R_ASC_PCMO	Range between ASC and PCMO	ASC와 PCMO간 거리
NUC	NDI Update Coordinate	NDI 갱신 시점의 이동객체 위치좌표
R_NUC_PCMO	Range between NUC and PCMO	NUC와 PCMO간의 거리
threshold_1 (TCUDI)	threshold_1 (Time Cycle for Updating Disaster Information)	문턱값_1 (재해정보갱신을 위한 시간주기)
threshold_2 (MDCUDI)	threshold_2 (Moving Distance Cycle for Updating Disaster Information)	문턱값_2 (재해정보갱신을 위한 이동거리주기)
threshold_3 (TCAS)	threshold_3 (Time Cycle for Alerting Service)	문턱값_3 (경보서비스를 위한 시간주기)
threshold_4 (MDCAS)	threshold_4 (Moving Distance Cycle for Alerting Service)	문턱값_4 (경보서비스를 위한 이동거리주기)

[ADAS 알고리듬]

① Start App

② Acquisition PCMO from GPS or network moduleloop_1

③ Acquisition n NDIs in results to compare PCMO with DIDB of server

④ Insert or update information of present moving object on SDBMO

⑤ Store n NDIs to member variables of App in the client loop_2

⑥ Calculate a distance(d) between the CDL of each NDI and the PCMO using the expression[9,10]

⑦ Perform a novel alert service to compare the distance d in ⑥ with the range of caution and danger

⑧ Store AST and ASC. loop_3

⑨ Compute PTAS(PTAS <- Present Time - AST)

⑩ Compute PTIG (PTIG <- Present Time - DIGT)

⑪ Compute R_ASC_PCMO (R_ASC_PCMO <- |ASC - PCMO|)

⑫ Compute R_NUC_PCMO (R_NUC_PCMO <- |NUC - PCMO|)

⑬ IF PTIG >= threshold_1 OR R_NUC_PCMO >= threshold_2
server_connecting_flag = 1;

⑭ IF server_connecting_flag ≠ 1
IF PTAS < threshold_3 AND R_ASC_PCMO < threshold_4
goto loop_3;
ELSE
Acquisition PCMO from the GPS or network module.
goto loop_2;
ELSE goto loop_1;

ADAS 알고리듬에서 보는 바와 같이 스마트폰에서 구동되는 App은 최초 start시(①)와 서비스 중에 주기적으로 이동객체의 현재위치좌표를 가지고(②) 서버의 DIDB에 접속하여 PCMO와 가장 가까운 최신의 NDI를 획득(③)하며, 동시에 이동객체의 정보를 SDBMO에 등록 혹은 갱신(④)한다.

서버로부터 획득한 NDI와 획득시간 DIGT를 클라이언트 상의 App 프로그램에서 기억하고(⑤) 있다가 PCMO와 n개의 NDI 재해위치 좌표(CDL)간의 거리(d)를 계산한다[9,10]. 계산결과인 di=1,..,n와 NDI에 포함된 각 재해의 주의반경과 위험반경을 비교하여 현재 이동객체를 소지한 자신의 위치를 안전지역, 주의지역, 위험지역 중의 하나로 분류하여 맵

상에 재해지점으로부터의 거리와 함께 표시하여 경보서비스를 수행(⑦)한다.

다음 서비스의 시점을 계산하기 위해서는 재해경보서비스 시각(AST)과 재해경보서비스 위치좌표(ASC)를 기억(⑧)하고 있다가 재해경보서비스 경과시간(PTAS) 계산(⑨)과 경보서비스 위치좌표(ASC)와 이동객체의 현재좌표(PCMO) 간 거리(R_ASC_PCMO)를 계산(⑪)하는데 사용한다.

또한 현재시각과 최근 재해정보획득시점(DIGT)으로 부터의 경과시간(PTIG)을 계산(⑩)하고, 스마트폰 App의 NDI 갱신 시점에서의 이동객체의 위치좌표(NUC)와 이동객체의 현재 위치좌표(PCMO)간의 거리(R_NUC_PCMO)를 계산(⑫)한다.

만약 최근 재해정보획득시점(DIGT)으로부터의 경과시간(PTIG)이 threshold_1(TCUDI) 를 초과하거나 NDI 갱신 시점의 이동객체 위치좌표와 이동객체 현재위치좌표(PCMO) 간의 거리 (R_NUC_PCMO)가 threshold_2(MDCUDI) 를 초과할 경우에는 서버에 접속하여 새로운 재해정보를 획득하기 위한 flag를 1로 세팅(server_connecting_flag = 1) 한다.

또한 flag가 1이 아니면서 재해경보서비스 경과시간(PTAS)이 threshold_3(TCAS)에 미달되고 동시에 재해경보서비스 위치좌표(ASC)와 이동객체의 현재 위치좌표(PCMO) 간의 거리(R_ASC_PCMO)가 threshold_4(MDCAS)에 역시 미달될 때에는 서비스 대기를 위하여 loop_3인 재해경보서비스 경과시간(PTAS) 계산(⑨) 위치부터 반복을 수행한다.

반면 flag가 1이 아니면서 재해경보서비스 경과시간(PTAS)이 threshold_3를 초과하거나 혹은 재해경보서비스 위치좌표(ASC)와 이동객체의 현재 위치좌표(PCMO)간의 거리(R_ASC_PCMO)가 threshold_4를 초과할 경우에는 GPS 모듈로부터 PCMO를 다시 획득한 후 재해 경보서비스 수행을 위하여 loop_2의 위치인 현재 이동객체의 위치좌표(PCMO)와 NDI에 등록된 n개의 등록된 재해 위치좌표(CDL) 간의 거리(d)를 계산(⑥)하는 위치부터 반복을 수행한다.

마지막으로 flag가 1인 경우에는 서버의 DIDB로부터 최신의 재해정보를 얻기 위하여(③) loop_1으로 이동한다.

이동객체의 현재위치좌표를 얻는 방법에는 GPS PROVIDER를 이용하는 방법과 NETWORK PROVIDER를 이용하는 방법이 있는데 제안한 방법에서는 둘 중에서 가용한 것을 시스템이 알아서 선택하는 방식을 이용하였다.

제안한 방법에서는 안드로이드 환경에서 이동객체가 서버에 접속을 시도하기 위하여 표 6과 같이 HttpPost 객체와 HttpClient 객체 및 ResponseHandler 객체를 사용하였다.

HttpClient 객체의 execute 메서드를 실행하면 String 형태의 result가 전송된다.

표 10. 서버 접속 시도 메서드(sendData)의 간략화 구조
Table 6. Simplified structure of the method(sendData) for accessing the server.

```
private String sendData(
  String lati, String longi, String pnum){
  //Http Post 방식으로 서버에 전송하기 위한 객체생성
   HttpPost request
   = makeHttpPost(위도,경도, pnum, url);
  //안드로이드 웹서비스용 HttpClient컴퍼넌트 생성
   HttpClient client
   =  new DefaultHttpClient();
  //HttpResponse객체 생성과정에 숨겨진 Handler 생성
   ResponseHandler<String> reshandler
   = new BasicResponseHandler() ;
  //String 템플릿형 reshandler를 넘김
   result = client.execute( request, reshandler );
   return result; //String 형태 전달
}
```

본 논문에서 구현한 재해경보 서비스 시스템에서는 n을 3으로 설정하였으며, 서버 DIDB로부터 검색된 3개의 NDI가 클라이언트인 이동개체를 향하여 XML형식으로 전송되며 그 형식은 표 7과 같다.

표 11. 검색결과에 대하여 서버에서 클라이언트로 전송되는 XML형식
Table 7. XLM format for transmitting the retrieval result from server to client

```
out.println("<disInfo>");          //재해정보의시작
for(int t=0; t<3; t++){
  disInfoEntity dis =disinfo.getdisrecord(aa[t]);
                                   //NDI객체 생성
  double dbLati
       =Double.valueOf(dis.getLati());
  double dbLongi
       =Double.valueOf(dis.getLongi());
  out.println("<dis"+t+">");     //NDI 정보시작
  out.println("<snum>"+dis.getSnum()+"</snum>");
            //NDI 재해번호
  out.println("<disCode>"+dis.getDisCode()+"
              </disCode>"); //NDI 재해유형코드
  out.println("<Lati>"+dbLati+"</Lati>");
             //NDI 위도
  out.println("<Longi>"+dbLongi+"</Longi>");
            //NDI 경도
  out.println("<e_Radi>"+dis.getE_radi()+"</e_Radi>");
            //NDI위험반경
  out.println("<w_Radi>"+dis.getW_radi()+"</w_Radi>");
             //NDI주의반경
  out.println("</dis"+t+">");    //NDI 정보 끝
}
  out.println("</disInfo>");     //재해정보의 끝
```

3. App기반 ADAS 실행 결과와 자동 시각화 방법

여기서는 제안한 방법을 가지고 구현한 재해경보서비스 시스템의 결과를 살펴본다. 이 시스템은 윈도우 7과 안드로이드 환경에서 구현되었으며, DBMS로는 mySQL과 개발 툴로는 안드로이드 SDK ver 2.2을 사용하였고, 스마트폰인 갤럭시 S와 구글 폰에 포팅하여 실험을 실시하였다.

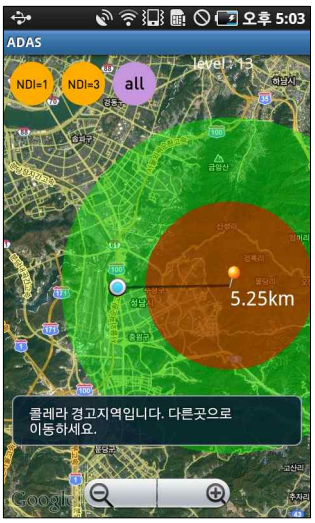

그림 3. 표준 재해경보 서비스
Fig. 3. stand of disaster alert service

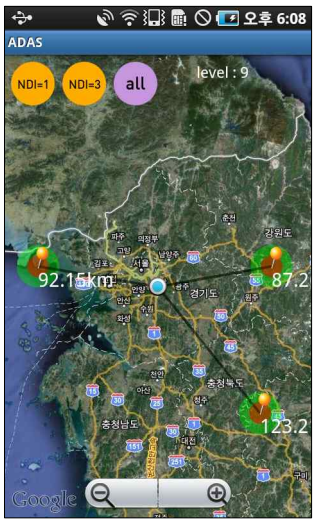

그림 4. NDIn=3 재해경보 서비스
Fig. 4. disaster alert service of NDIn=3

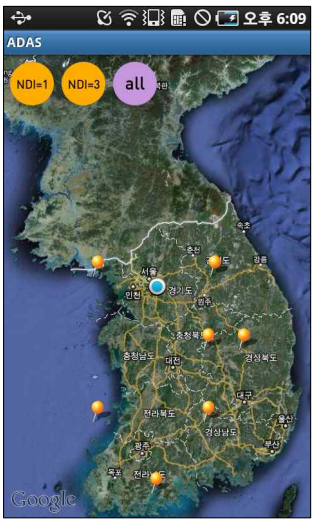

그림 5. 전국 재해지역서비스
Fig.5. countrywide disaster alert service

App의 서비스유형은 App이 구동될 때 기본적으로 터치버튼 NDI=1이 선택되며, 이것은 이동객체 자신을 중심으로 가장 가까이에 있는 재해위치와 거리를 비교하여 그림 3과 같이 현재 상태의 경보수준과 거리를 표시하고 동시에 voice와 text로 경보서비스를 내보낸다. 또한 터치버튼 NDI=3을 선택하면 그림 4처럼 현재의 이동객체를 중심으로 3개의 NDI와 현재 이동객체간의 거리를 하나의 화면에 나타낼 수 있으며, 터치버튼 all을 선택하면 현재 이동객체와 현재 경보발령 중인 재해지역 전체를 한눈에 볼 수 있는 그림 5와 같은 기능을 서비스할 수 있다.

이동객체의 위치와 재해지역 위치 간의 거리에 따라 화면의 zoom을 자동으로 조정하는 자동시각화 재해경보서비스를 구현하기 위해서는 이동객체와 재해위치와의 거리계산을 통해 zoom-in-out에 사용할 zoom level을 구해야한다.

이것은 실행화면의 size와 현 이동객체 위치와 재해지역 위치좌표 간의 거리를 계산하여 level을 도출하고, 이 level을 map을 drawing하는 함수를 호출할 때 매개변수로 기술한다.

안드로이드 기반 스마트폰에서 자신의 객체위치와 재해지역간의 거리에 맞추어서 가장 적합한 화면을 표시하기 위하여 zoom level을 바꾸어가며 반복적으로 실험하여 얻은 거리에 따른 zoom level을 표 8에 나타내었다. NDI=1인 표준 재해경보서비스는 가장 가까운 재해지역과의 거리를 기준으로 zoom level을 적용하고, NDI=3인 경우에는 3개중에서 가장 먼 곳의 재해지역과의 거리를 가지고 zoom level을 적용하였다.

표 8에서 보는 바와 같이 street view가 satellite view 보다 좀 더 상세한 영역까지 화면에 표시할 수 있다.

본 실험에서는 기본적으로 재해발생위치로부터 위험반경을 5km, 주의반경을 10km로 설정하여 실험(그림 3~7)을 하였고, zoom-in의 한계 level을 도출하기 위한 실험에서는 위험반경과 주의반경을 각각 0.2km와 0.5km로 축소하여 실험(그림 8~11)을 하였다. 실험결과에서 위험반경은 적색으로 주의반경은 녹색으로 나타내고 있다.

표 8. 자동 시각화를 위한 zoom level
Table 8. zoom level for automatic visualization

level	19	18	17	16	15	14	13
거리 (km)	0.15	0.3	0.6	1.5	2.5	5	10
적용가능	street view			street and satellite view			

level	12	11	10	9	8	7	비고
거리 (km)	20	40	80	160	320	640	
적용가능	street and satellite view						

2.5km 이내인 15를 적용하고 있다.

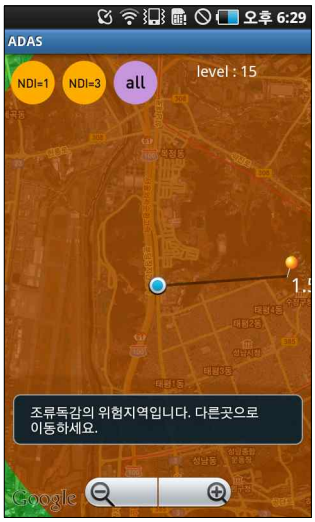

그림 6. 위험지역에서의 재해경보서비스
Fig. 6. The disaster alerting service in danger zone

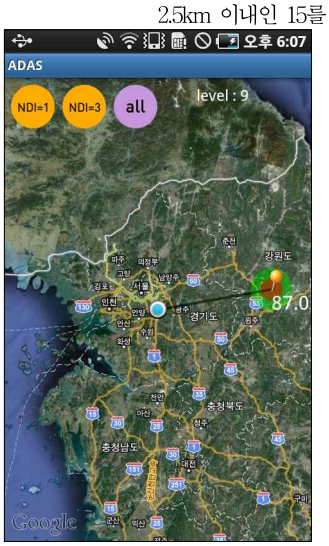

그림 7. 안정지역에서의 재해경보서비스
Fig. 7. The disaster alerting service in safety zone

그림 8. satellite view 16 level 에서의 재해경보서비스
Fig. 8. The disaster alerting service in satellite view level 16

그림 9. street view 16 level 에서의 재해경보서비스
Fig. 9. The disaster alerting service in street view level 16

그림 10. satellite view 17 level 에서의 재해경보서비스
Fig. 10. The disaster alerting service in satellite view level 17

그림 11. street view 17 level 에서의 재해경보서비스
Fig. 11. The disaster alerting service in street view level 17

그림 6은 이동객체 자신이 위험반경 내에 위치한 경우의 재해경보서비스인데, 현재 이동객체와 재해위치가 1.5km 거리로 위험반경 5km 안에 있으므로 적색으로 표시되었고, 자동시각화를 위한 zoom level은 표 8에서 보는 바와 같이

그림 7은 자신의 위치와 재해지역 중에서 가장 가까운 지역과의 거리가 아직 안전한 상태에 있음을 나타내는 서비스의 유형으로, 현재 이동객체와 가장 가까운 재해지역과의 거리가 87.0km로 주의지역 설정영역인 10km 밖에 있고 자동시각화를 위한 zoom level은 160km 이내인 level 9가 적용되었다.

또한 재해경보서비스에서 zoom-in의 한계 level을 도출하기 위하여 그림 8에서부터 11까지는 위험반경과 주의반경을 각각 0.2km와 0.5km로 축소하여 실험을 하였는데, 그림 8은 satellite view로, 그림 9는 street view로 모두 zoom level 16을 적용하여 서비스를 표시한 결과이다.

그리고 그림 10과 11은 satellite view와 street view 에서의 zoom level을 17로 적용하여 서비스를 나타낸 결과인데, satellite view에서 level 17을 적용한 결과 그림 10과 같이 깨어져 나타남을 발견할 수 있었다. 즉, satellite view에서 화면의 깨어짐 없이 표현할 수 있는 최대 zoom-in 상태는 level 16임을 알 수 있다. 표 8에서 street view는 zoom level이 최대 19까지, satellite view는 최대 16까지 표현이 가능함을 보이고 있다.

Ⅳ. PDAS와 ADAS 모델의 비교

여기서는 수동형(passive) 재해경보서비스(PDAS)의 문제점들을 살펴보고 제안한 능동형(active) 재해경보서비스(ADAS)에 대한 타당성을 제시한다. 기존의 방식인 수동형 재해경보서비스(PDAS)는 서버에서 각 클라이언트에 대한 위치획득의 모든 책임을 다 갖고 있기 때문에 서버부하 문제가 심각한 문제로 대두되었다.

하지만 제안한 능동형 재해경보서비스(ADAS)는 클라이언트에 App을 탑재하고 GPS모듈을 통하여 이동객체의 위치를 실시간으로 얻을 수 있기 때문에 위치획득 문제를 개별화한 것이어서 서버의 부하를 줄 일 수 있었다.

제안한 방법은 서버의 부하문제를 해결할 수 있는 것뿐만 아니라 App에 검색 및 시각화와 같은 다양한 기능을 구현할 수 있어 사용자 중심의 필요를 만족시켜 줄 수 있게 된다. 다음은 제안한 방법인 ADAS 모델과 기존 방법인 PDAS 모델과의 비교를 서버측에서의 단순한 부하 량 계산을 가지고 다루고자 한다. 각 대상 객체마다 재해경보서비스를 위한 위치획득 시간간격을 1분이라고 하고 전체 서비스 대상객체를 100만개라고 가정할 때, 서버 측에서 위치획득을 전담하는 PDAS 모델의 경우 하루 동안 서버에서의 위치획득수를 계산하면 다음과 같다.

1,000,000(개) × 1,440(분) = 1,440,000,000(번)

이 방법은 DIDB의 갱신주기가 30분이든, 1시간이든, 그리고 이동객체의 이동속도가 5km이든 100km이든 재해정보의 갱신주기나 이동속도에 상관없이 서버에서의 1일 위치획득수는 동일하다. 이것은 이동객체의 이동거리를 알기위해서는 서버에서 이동객체의 위치를 계산해야 하기 때문이다.

하지만, 본 논문에서 제안하는 이동객체에 App을 탑재하는 방식인 ADAS 모델인 경우에는 동일하게 1분 간격으로 재해경보서비스를 진행하더라도 클라이언트의 App의 멤버변수에 존재하는 NDI 정보를 가지고 서비스를 진행하고 서버로의 접근은 이동객체가 일정거리이상 이동하거나 서버로부터의 NDI 갱신 시간이 일정시간 이상 경과되었을 때에만 서버에 접근하도록 한다.

서비스 대상 객체가 100만개라 가정할 때, 경과시간에 따른 NDI 갱신주기를 30분과 60분으로 할 경우와 이동거리에 따른 NDI 갱신에 있어서 이동객체의 속도가 5km와 100km일 때 이동거리 주기가 각각 5km 와 10km인 경우를 적용하여 ADAS 모델의 하루 동안 서버에서의 위치획득수를 ①에서 ⑥까지 조건에 따라 계산한 결과를 표 9에서 보이고, 단순히 PDAS와 ADAS 모델의 서버 측 위치획득 수만을 가지고 두 모델에 대한 서버의 부하 량을 비교하였다.

PDAS인 기존의 위치획득 모델들은 이동객체 데이터베이스와 이동통신망 사이의 통신부하를 최소화하기 위해 이동객체의 이동패턴을 파악하고 이동객체의 위치획득 간격을 동적으로 조절하여 불필요한 위치정보를 획득하지 않도록 함으로써 위치획득 횟수를 줄이도록 하는 방법들이 대부분을 차지하고 있다. 하지만 서버에서 이동객체의 이동패턴을 파악하려면 이동객체의 위치를 파악해야만 한다.

표 9. PDAS모델과 ADAS모델의 서버 부하 량 비교
Table 9. the comparision of overload on server between PDAS and ADAS [이동객체 100만개 기준]

모 델	조건	서버 측 1일 위치 획득 수 (단위:백만)	부하 비율	조건설명
PDAS (기존방법) 서비스 간격:1분	기본	1,440	1	위치획득 시간간격: 1분
ADAS (제안방법) 서비스 간격:1분	①	48	1/30	NDI 갱신주기: 30분
	②	24	1/60	NDI 갱신주기: 60분
	③	24	1/60	NDI 갱신주기: <시속 5km> 이동거리: 5km
	④	480	1/3	NDI 갱신주기: <시속 100km> 이동거리: 5km

	⑤	12	1/120	NDI 갱신주기: <시속 5km> 이동거리: 10km
	⑥	240	1/6	NDI 갱신주기: <시속 100km> 이동거리: 10km

표 9를 살펴보면 서버에서 엄청난 계산 량이 요구되는 다양한 PDAS 모델들의 이동패턴에 대한 분석은 전혀 고려하지 않았고, 단지 서버에서 이동객체인 클라이언트에 대한 위치획득을 요청하는 시간간격과 재해경보서비스 주기를 동일한 1분으로 가정하여 위치획득수를 계산하였다.

ADAS도 재해경보서비스 주기를 PDAS와 동일하게 1분으로 설정하였는데, 이는 App에서 1분 간격으로 서비스를 진행하며, 클라이언트인 이동객체가 능동적으로 새로운 재해정보를 얻기 위하여 서버에 접근하는 주기를 서버 측 위치획득 수로 계상하였다.

표 9에서 ADAS는 ①에서부터 ⑥의 조건에 따라 클라이언트가 서버에 접속하는 회수를 1일 위치 획득수로 계산하여 PDAS와 비교하였다.

V. 결론 및 향후 연구 방향

ADAS 모델인 제안한 App 기반의 능동형 재해경보서비스 방법은 스마트 이동객체의 소지자가 App을 작동하였을 때 클라이언트인 현재 자신의 위치를 GPS 모듈로부터 획득하여 서버의 DIDB에 접근하며, DIDB로부터 현재 이동객체의 위치에서 가장 가까운 거리에 있는 n개의 재해정보인 NDI를 얻음과 동시에 이동객체 정보를 서버의 SDBMO에 등록 혹은 갱신한다.

또한 서버로부터 얻은 n개의 재해정보 NDI를 스마트폰에서 구동되는 App의 멤버변수인 구조체 배열에 저장하고, 주기적(시간간격 혹은 이동거리간격)으로 현재 이동객체의 자신의 위치정보와 NDI를 비교하여 재해경보수준을 위험, 주의, 안정으로 구분하여 맵 상에 현재위치에서의 거리정보와 함께 능동적으로 경보서비스를 제공하며, App의 구동 중에 주기적으로 서버의 DIDB에 접근하여 최신의 재해정보로 클라이언트의 NDI를 갱신함으로써 최신의 정보가 반영되도록 하고 있다.

표 9에서 보는 바와 같이 기존방법인 PDAS 모델은 경보서비스를 실시할 때마다 서버에서 이동객체의 위치를 획득해야만 하는 서버부하 문제가 있었지만, 제안한 ADAS 방법은 일정주기 동안은 자체적으로 클라이언트에서 NDI정보를 이용하여 능동적으로 서비스를 수행하는 방식으로 서버에 접근하는 횟수를 현저히 줄임으로써 서버부하문제를 근본적으로 해결할 수 있었다.

또한 그림 3에서부터 그림 11까지 보는 것처럼 App 상에 다양한 검색 기능(전체 재해위치 검객, 현재위치에서 가장 가까운 재해위치 검색)과 View의 시각화(현재 이동객체와 재해위치와의 거리에 따른 자동 Zoom In-Out)를 구현함으로써 사용자의 필요를 충족 할 수 있는 실시간 능동형 재해경보서비스 시스템의 프로토타입(prototype)을 개발하였다.

향후 연구방향은 이동객체의 속도와 이동거리를 계산하여 최적화된 서버의 접속시점을 찾는 것과, SDBMO의 정보를 이용한 다양한 서비스 모델을 개발하는 것이다.

참고문헌

[1] JinWoo Song, Byung-Ik Ahn, Kwang-Jo Lee, JungSuk Han, and Sung-Bong Yang, "An Improved Location Polling Algorithm for Location-Based Alert Services," Journal of KIISE : Database, Vol 37, No. 1, pp. 22-32, Feb. 2010.

[2] Kyoung-Wook Min, and Dae-Soo Cho, "Technique for Acquisition of Moving Object Location in LBS,"Journal of KIPS : D, Vol 10-D, No. 6, pp. 885-896, Oct. 2003.

[3] Ralf H. Guting, Mike H. Bohlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, Michalis Vazirgiannis, "A Foundation for Representing and Querying Moving Objects," ACM Transactions on Database Systems, Vol. 25, No. 1, pp. 1-42, Mar. 2000.

[4] Wha Sook Jeon and Dong Geun Jeong, "Design of a Paging Scheme based on User Mobility Classes for Advanced Cellular Mobile Networks," Journal of KIISE : Information and Communication, Vol. 29, No. 3, pp. 216-223, Jun. 2002.

[5] A. Bar-Noy, I. Kessler and M. Sidi, "Mobile users : To update or not to update?," Wireless Networks, Vol. 1, No. 2, pp.187-196, 1995.

[6] A. Abutaleb and V. O. K. Li, "Location update optimization in personal communication systems," Wireless Networks, Vol. 3, No. 3, pp. 205-216, 1997.

[7] I. F. Akyildiz, S. M. Ho and Y. B. Lin, "Movement-based location update and selective paging for PCS networks," IEEE/ACM Trans. Networking, Vol. 4, No. 4, pp. 629-638, Aug., 1996.

[8] Byung-Ik Ahn, Sung-Bong Yang, Heui-Chae Jin, and Jin-Yul Lee, "Location Polling Algorithm for Alerting Service Based on Location," Springer Verlag Lecture Notes in Computer Science, Vol. 3833, pp. 104-114, 2005.

[9] Changduk Suh and Gi-Tae Han, "A Direction Computation and Media Retrieval Method of Moving Object using Weighted Vector Sum," Journal of KIPS : D, Vol 15-D, No. 3, pp. 399-410, Jun. 2008.

[10] Dong-Soo Ha, Sung-June Park, "Smart-Phone based User Movement State Identification Algorithm," Journal of KSCI, Vol. 16, No.3, pp. 167-174, Mar. 2011.

저 자 소 개

한 기 태(Gi-Tae Han)

1982년 : 충남대학교 계산통계학과 졸업(학사)

1990년 : 한양대학교 대학원 전자계산학과(공학석사)

2001년 : 한양대학교 대학원 전자공학과(공학박사)

1992년-현재 : 경원대학교 인터랙티브미디어학과 교수

2009-2010년: UT at Austin, Researching professor.

관심분야 : 컴퓨터비전, 영상처리, 모바일서비스 등

E-mail: gthan@kyungwon.ac.kr