동적인 네트워크 환경을 지원하는 에이전트 플랫폼의 설계와 구현

이철희° 윤현상 이은석
성균관대학교 소프트웨어공학 연구실
chee@skku.edu° {wizehack, eslee}@ece.skku.ac.kr

Design and Implementation of Agent Platform for Dynamic Network Environment

Cheolhui Lee° Hyunsang Youn Eunsuk Lee
Sungkyunkwan Univ. Software Engineering LAB.

요 약

최근 네트워크 환경이 점차 복잡해지고 동적으로 변화함에 따라서 이러한 에드혹 네트워크 환경을 지원하기 위한 에이전트 플랫폼에 관한 연구가 진행되고 있다. 그러나 기존 에드혹 네트워크 환경을 지원하는 에이전트 플랫폼은 플랫폼의 경량화에는 성공하였지만 FIPA에서 제안하는 에이전트 플랫폼의 표준안을 만족시키는 다양한 기능을 제공하지 못한다. 이것은 다른 에이전트 기반 시스템들과의 상호 운용성을 떨어트린다. 본 논문에서는 FIPA에서 제안하는 기능을 만족시키면서, 보다 빠른 서비스 제공 및 경량화된 에이전트 플랫폼을 설계 및 구현하였다. 본 논문에서 제안하는 에이전트 플랫폼은 FIPA 표준안을 만족시키기 때문에 에드혹 네트워크 환경뿐만 아니라 범용적인 네트워크 환경에서도 동작 가능하고, 플랫폼의 경량화된 특성으로 인하여 PDA와 같은 모바일 기기에도 이식하여 사용할 수 있다. 본 논문에서는 에드혹 네트워크 환경을 지원하는 에이전트 플랫폼을 설계 및 구현하였으며 동적 및 정적 네트워크 환경의 효과적인 적용 및 에이전트 플랫폼이 구동 불가능 단말기에서의 구동여부를 평가하여 그 유효성을 검증하였다.

1. 서 론

최근 네트워크 환경이 점차 복잡해지고 동적으로 변화함에 따라서, 이를 이용하는 컴퓨팅 환경 역시 변화하고 있다[1]. 또한 유비쿼터스 컴퓨팅 환경에서 동작하는 지능형 시스템을 개발하기 위해 에이전트 기반 소프트웨어를 지원하기 위한 개발 및 운용 플랫폼에 관한 연구가 활발히 진행되고 있다[2].

FIPA 에서는 에이전트 기반 시스템 개발 및 에이전트 간 상호 운영을 위해 에이전트 플랫폼에 관한 표준안을 제시하고 있다. 본 표준안에 따르면 에이전트 플랫폼은 에이전트간 커뮤니케이션을 지원하고, 에이전트들의 관리 및 서비스 디스커버리를 제공한다.[2] 그러나 일반적으로 에이전트 플랫폼의 경량화가 중요시 되는 에드혹 네트워크 환경에서는 에이전트 플랫폼이 FIPA에서 제안하는 표준안을 만족시키는데 많은 문제점을 가지고 있다. 이러한 에이전트 플랫폼의 경량화 문제점으로 인하여 Petteri Alahuhta, Henri Lothman, Heli Helaakoski는 그들의 연구에서 무선 네트워크 환경에서 사용되는 모바일 단말기에서 사용 가능한 에이전트 플랫폼의 아키텍처를 제안하였다.[5] 그러나 Petteri Alahuhta 외 2명이 제안하는 Apricot 에이전트 플랫폼은 HTTP/Socket 기반의 에이전트 서비스를 제공하여서 에드혹 네트워크에 적합하지 못하다. 그리고 에이전트 관리 및 서비스 디스커버리 기능을 제공하는데 있어서 FIPA의 표준안을 따르지 않았으며 이것은 기존에 존재하는 에이전트 플랫폼들간의 상호 운용성을 떨어트리게 된다. 그러나 에드혹 네트워크 환경에서의 컴퓨팅 서비스를 제공하면서, FIPA에서 제안하는 모든 기능을 만족시키는 에이전트 플랫폼을 개발하는 것은 에이전트 기반 시스템의 개발 및 운용 환경을 제공하는데 있어서 매우 어려운 문제이다.

본 논문에서는 상기의 문제점을 해결하기 위하여 FIPA의 표준안을 기초로 무선랜등의 프로토콜을 적용하여 보다 효율적인 에드혹 네트워크 기반 에이전트 플랫폼을 구축할 수 있도록 하는 방법을 제안하였다. 이것을 통해서 에이전트 기반 시스템 개발자는 FIPA의 표준안은 만족시키면서 에드혹 네트워크 환경에서 동작하는 경량화된 에이전트 플랫폼을 사용할 수 있다. 본 논문에서 제안한 에이전트 플랫폼의 주요 기능 및 특성은 다음과 같다.

1) 동적 및 정적 네트워크 환경을 효과적으로 적용
2) 에이전트 플랫폼의 경량화 및 네트워크 안정도 향상
3) 에이전트 플랫폼이 구동되기 어려운 작은 단말기와의 연동

본 논문에서 제안한 시스템의 유효성을 평가하기 위해 프로토 타입을 제작하여 기존 플랫폼과 비교하여 성능을 측정하였다. 다중 경로를 이용하는 블루투스를 통한 에이전트 서비스보다 직접 연결이 되는 무선랜을 이용한 경우의 에이전트 서비스가 빠른 응답속도를 보이는 결과가 도출된다. 그리고 에이전트 서비스가 불가능한 단말기까지의 서비스 제공 역시 가능함이 증명되었다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 모바일 환경을 지원하는 에이전트 시스템에 대해 간략히 소개한다. 3장에서는 제안하는 시스템에 대해 설명하며, 4장에서는 제안시스템의 몇 가지 성능에 대한 테스트를 기술한다. 그리고 마지막으로 5장에서 결론을 정리한다.

2. 관련연구

Java Agent DEvelopment Framework (JADE)[3]는 자바로 구현된 소프트웨어 프레임워크이다. JADE의 목적은 FIPA 표준에 맞는 미들웨어를 통해 멀티에이전트 시스템 구현을 단순화하는 것에 있다. JADE를 모바일 장치에서 사용할 수 있도록 개발된 것이 JADE의 애드온(add-on)인 JADE-Lightweight Extensible Agent Platform(JADE-LEAP)이다. JADE-LEAP은 FIPA 에이전트가 자바나, Microsoft .Net 프레임워크가 동작하는 경량 장치에서 실행할 수 있다. 모바일 장치에서부터 기업용 서버까지 이형적인 네트워크를 통해서도 사용될 수 있는 멀티 에이전트 시스템이다. LEAP은 어떠한 수정도 없이 충분한 자원과 프로세싱 파워를 제공하는 모바일 장치에서 동작할 수 있도록 개발되었다. LEAP은 TCP/IP GSM, IEEE 802.11 Wireless LAN과 같은 무선 네트워크를 활용한다.

JADE-LEAP은 가장 널리 사용되는 모바일 에이전트 시스템이지만, TCP/IP에 접근할 수 없는 곳에서는 활용할 수 없으며, J2SE기반의 메인 컨테이너가 반드시 동작하고 있어야 하는 한계점을 갖고 있다. 즉, 애드혹 환경에서 동작할 수 있도록 구현되어 있지 않다.

Apricot Agent Platform[5]는 핀란드 Oulu 대학교에서 제안한 에이전트 플랫폼으로 HTTP/Socket을 이용하여 모바일 장치에서 사용 가능한 에이전트 플랫폼이다. Apricot은 모바일 장치가 인터넷에 연결이 가능 할 때 인터넷을 통한 에이전트 플랫폼을 구축한다. 이는 장거리 에이전트 플랫폼에서 효과를 발휘한다.

무선을 사용하긴 하나 HTTP/Socket을 프로토콜로 사용하기 위해서는 인터넷에 접근이 가능해야 한다는 제약이 있다. 인터넷을 위한 고정된 장치가 반드시 존재해야 되는 것은 유비쿼터스 환경에서 큰 마이너스 요인이며 최근 주목 받는 에드혹 네트워크에서의 에이전트 플랫폼과는 거리가 멀다.

ANBD(Agent Network for Bluetooth Devices)[6]는 블루투스 프로토콜을 사용하여 개인 모바일 장치에서 에이전트 서비스를 제공하는 시스템이다. ANBD는 모바일 장치와 고정된 장치로 구성되며 모바일 장치는 J2ME가 동작하는 PDA, 스마트폰, 노트북, 랩탑 컴퓨터 등의 블루투스가 가능한 장치이며, 고정된 장치는 PC같은 데이터 베이스와 에이전트 서버가 동작하며 모바일 장치들이 블루투스로 접근할 수 있는 장치이다. 모바일 장치가 고정된 장치에 접근하여 서비스를 요청할 때, 고정된 장치는 해당 모바일 장치를 위한 에이전트를 생성하여 관리하고, 모바일 장치에서는 해당 에이전트와의 사용자간의 인터페이스만 수행한다.

ANBD 시스템이 비록 TCP/IP를 지원하지 않는 환경에서 에이전트 서비스를 제공할 수 있지만 JADE-LEAP과 마찬가지로 고정된 기반 시설이 존재하는 곳에서만 동작할 수 있다는 문제점이 있다. 또한 이 시스템에서는 사용자를 위한 에이전트를 고정된 기반시설에서 생성하게 되는데, 이러한 방식에서는 사용할 수 있는 에이전트 서비스 종류에 제약이 따르게 된다.

3. 제안 시스템

본 논문에서 우리는 다음과 같은 시스템을 제안한다. 첫째로, 고정된 기반 시설이 없는 곳에서도 여러 개의 모바일 장치간의 에이전트 서비스를 제공한다. 둘째로, 동적으로 AMS, DF 서비스 제공자를 생성하며, 이를 변경할 수 있다. 셋째로, 서비스를 제공할 때 보다 효율적인 프로토콜의 선택으로 서비스 반응 속도를 증가시킨다. 넷째로, 서비스 제공자가 정상, 비정상적으로 네트워크를 이탈할 때 모두, 다른 장치가 차기 서비스 제공자가 되어 서비스를 유지할 수 있다. 다섯째로, 에이전트 플랫폼이 동작하기 힘든 장치에도 에이전트 서비스를 제공한다.

앞서 밝힌 듯이 기존의 에이전트 시스템은 고정된 기반 시설(애드혹 네트워크의 베이스 스테이션, 무선 AP등)이 있는 곳에서만 사용할 수 있거나 또는 단일 프로토콜로 인한 서비스 제공에 한계점을 갖고 있다. 위에 언급한 특성을 통해 앞서 제기된 고정된 기반 시설이 있는 곳이나 단일 프로토콜로 제공되는 서비스의 한계를 해결할 수 있으며, 노드들의 네트워크 참여와 이탈이 빈번한 애드혹 네트워크 상에서도 화이트, 옐로우 페이지 서비스를 제공하며 이 서비스를 제공하는 네트워크 내에서 발생할 수 있는 서비스 중단이라는 문제점을 해결할 수 있다. 또한 에이전트 플랫폼이 동작할 수 없는 장치도 플랫폼이 동작하는 장치와 통신이 가능할 때 자신의 에이전트를 이용하여 에이전트 서비스를 제공받을 수 있다.

3.1 시스템 구성

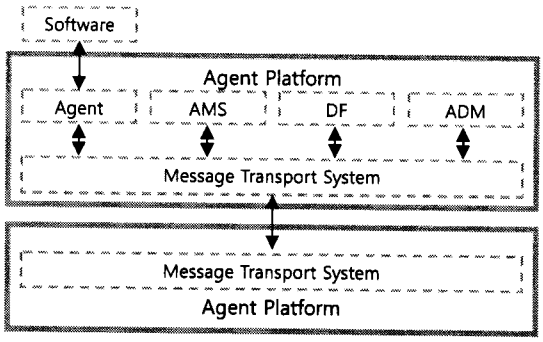

그림 1. 시스템 구성도

본 논문에서 제안하는 시스템은 (그림 1)과 같이 구성된다. 플랫폼 내부를 FIPA의 표준에 맞추어 구성하되, ADM(AMS DF Manager)이라는 요소를 추가하고 AMS와 DF는 에드혹 네트워크에 적합하게 변경하였다. 각 에이전트 플랫폼의 Message Transportation System을 통해서 ADM은 다른 플랫폼의 Agent, AMS, DF, ADM등과 통신할 수 있다. 추가된 요소인 ADM은 (그림 2)와 같이 ADC(AMS DF Contoller), TM(Transport Manager) 그리고 PC(Protocol Converter)로 구성된다. 변경된 AMS, MTS(Message Transport System) 와 각각의 모듈에 대한 설명은 다음과 같다.

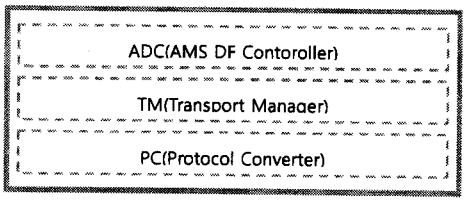

그림 2. ADM의 구성

(1) ADC : MTS의 Connector module에 의해 서비스 제공자가 네트워크를 이탈했음이 인식되면, 우선 순위에 따라 해당 장치가 서비스를 제공할 수 있도록 백업 받은 정보나 전송 받은 정보를 AMS, DF에 등록하고 동작하는 등의 작업을 진행한다.

(2) TM : 서비스 제공자 역할을 다른 장치에게 넘겨주기 위한 AMS, DF 정보 전송이나 주기적인 백업을 진행하며, 백업 받는 장치에서는 전송 받은 데이터를 보관하는 역할. 서비스 제공자나 백업 받는 장치에서만 동작한다.

(3) PC : 블루투스 또는 무선랜 중에서 적합한 프로토콜을 탐색 하고 그 결과를 바탕으로 장치간 연결될 프로토콜로 자동으로 변환시켜주는 기능을 제공한다. 배터리 상태나 메시지 전송 능력등을 고려하여 최적의 프로토콜을 설정한다.

(4) AMS : White Page Service 제공한다. 플랫폼 내에 하나의 AMS만 존재 해야하나 에드혹 네트워크의 불안정성으로 인해 백업용의 Clone AMS 존재하게 한다. 그리고 AMS 테이블에 등록되는 단말기들의 컴퓨팅 파워를 측정하고 그 결과를 토대로 컴퓨팅 파워가 강한 단말기부터 순서대로 AMS에 저장한다. AMS 테이블은 chaining 방법을 사용한 hash table을 사용한다.

(5) MTS : 현재 접속 가능한 에드혹 네트워크에 접속했을 때, 네트워크 내의 서비스 제공자를 인식하여 자신의 에이전트, 디렉토리 정보를 등록하거나 다른 장치에게 자신이 접속을 끊을 것을 알릴 때, 혹은 다른 장치의 접속이 끊어짐을 인식하는 기능과 무선랜 및 블루투스를 이용하여 메시지를 전송하는 기능을 FIPA에서 정의한 기능에 추가한다.

3.2 적합한 프로토콜 탐색 알고리즘

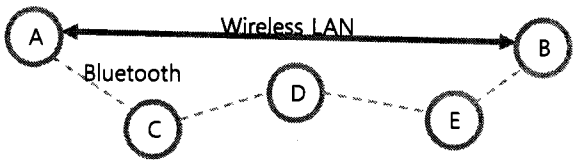

그림 3. Wireless LAN으로 직접 통신 가능한 경우

프로토콜 마다 장단점이 있지만, 네트워크 구성 속도 및 메시지 전송속도를 생각한다면 무선랜이 블루투스보다 뛰어나지만, 반면에 배터리 소모량은 블루투스가 뛰어나다. 그래서 (그림 3)처럼현재 단말기의 배터리 상태를 확인하여 무선랜의 사용가능 여부를 확인한 뒤 서비스 제공방법을 재구성한다. 그 후 접근하고자 하는 서비스 제공자의 위치를 파악하고 경로가 설정되면 서비스 제공자에 접근을 한 뒤 가용한 프로토콜을 확인한다. 서비스 제공자가 무선랜을 사용 가능한 상태이면 다중경로 대신 무선랜을 이용한 직접통신으로 프로토콜을 재설정 하고 소켓을 이용하여 연결한 뒤 메시지를 전송하는 방법을 제안한다.

```
Step1. 단말기 a에서 단말기 b로 서비스 요청
Step2. 단말기 b에서 a까지의 블루투스 경로 탐색
Step3. if(거치는 노드의 수 > 2)
Step3.1. if(단말기 b의 무선랜 동작 가능 여부 확인)
Step3.1.1. if(단말기 a의 무선랜 동작 가능 여부 확인)
Step3.1.1.1. 블루투스 경로를 통한 단말기 a의 주소 획득
Step3.1.1.2. 소켓을 통한 접속
Step3.1.1.3. 단말기 b는 소켓을 통해 단말기 a에 서비스 제공
Step3.1.2. else
Step3.1.2.1. 연결된 블루투스 경로를 통한 서비스 제공
Step3.2. else
Step3.2.1. 연결된 블루투스 경로를 통한 서비스 제공
Step4. else
Step4.1. 연결된 블루투스 경로를 통한 서비스 제공
```

그림 4. 적합한 프로토콜 탐색 알고리즘

3.3 안정적인 서비스 제공 알고리즘

에드혹으로 이루어진 에이전트 시스템에서 AMS, DF 서비스를 제공하는 장치가 네트워크를 이탈할 경우, 나머지 장치들이 서비스를 제공받을 수 없게 된다. 따라서 이러한 상황이 발생하더라도 네트워크 내의 장치들이 서비스를 제공받을 수 있도록 기존의 정보를 유지하고 역할을 전달하는 방식이 필요하다. 서비스를 제공하던 장치가 네트워크를 이탈하는 상황은 두 가지로 정리할 수 있다. 첫 번째는 자의적으로 네트워크를 이탈하는 상황이다. 이는 사용자나 사용자의 에이전트가 네트워크 이탈을 선언한 후 이탈하는 경우이다. 두 번째는 비정상적으로 네트워크를 이탈하는 상황이다. 서비스를 제공하고 있던 장치에 오류가 발생하거나, 통신할 수 있는 거리를 벗어났을 때를 들 수 있다. 전자의 경우 네트워크를 이탈하기 전에 현재 동작하고 있는 AMS, DF정보를 다른 장치에게 전송한 후, 이탈한다. 물론 정보가 전송되는 장치는 AMS에 미리 우선순위로 정해져 있게 된다. 전자의 경우 간단한 방법으로 대처할 수 있지만, 비정상적으로 네트워크를 이탈하는 상황에는 위와 같은 방법을 적용할 수 없다. 이에 대처하기 위하여 타 장치에 주기적인 백업을 수행한다. 즉 TM은 에드혹 네트워크가 구성되어있을 때, AMS 우선순위에 따라 정보 전송이나 백업을 진행한다. 네트워크를 구성하는 장치의 수가 4이상이 될 때, AMS, DF서비스를 제공하는 장치의 TM은 AMS, DF에 등록된 정보를 다른 장치의 TM에 주기적으로 백업을 하여, 비정상적으로 이탈했을 때 백업한 장치의 ADC가 동작하여 서비스 제공자 역할을 맡아 네트워크가 정상적으로 서비스를 제공해 줄 수 있도록 한다.

두 가지 경우 모두 MTS 통해 네트워크의 변경 사실을 다른 장치들에게 알리고 이를 통해 각 장치들의 MTS 통해 정상적으로 서비스를 재개한다. 서비스 제공자는 단지 화이트 페이지 서비스와 옐로우 페이지 서비스를 제공할 뿐이므로, 서비스 제공자가 변경될 때, 이미 서로간에 에이전트 통신을 하고 있던 장치들 간의 통신에는 문제가 없다.

```
Step1. 단말기 a의 접속 종료 요청
Step2. AMS, DF에 통보
Step3. AMS와 DF에서 a의 정보 삭제
Step4. AMS에서 차순위 단말기에 AMS, DF 백업 진행
Step5. 서비스 유지
```

그림 5. 정상적인 장치 이탈시 서비스 유지 알고리즘

```
Step1. 단말기 a의 비정상적인 접속 종료
Step2. if(AMS 접근 가능)
Step2.1. if(단말기 a에 대한 접근 요청)
Step2.1.1. 단말기 a의 이탈을 인지하고 AMS, DF에 통보
Step2.1.2. AMS와 DF에서 a의 정보 삭제
Step2.1.3. AMS에서 차순위 단말기에 AMS, DF 백업 진행
Step2.1.4. 서비스 유지 및 Step2. 반복
Step2.2. else
Step2.2.1. 서비스 유지 및 Step2. 반복
                        // 단말기 a의 이탈을 인지 못함
Step3.else
Step3.1. AMS 백업용 장치 서비스 활성화
Step3.2. AMS에서 차순위 단말기에 AMS 백업 진행
Step3.3. 서비스 유지 및 Step2. 반복
```

그림 6. 비정상적인 장치 이탈시 서비스 유지 알고리즘

3.4 에이전트 플랫폼이 동작 불가능한 단말기를 위한 XML

현재 하드웨어 성능으로는 모바일 폰에서 에이전트 플랫폼이 동작하기에 어려움이 있다. 모바일 장치에서도 에이전트 서비스를 제공받기 위해서는 플랫폼이 동작하는 장치에서의 지원이 필요하다. 우리는 플랫폼이 동작하지 않는 장치가 에이전트를 소유하기는 하되, 에이전트 프로세싱 시에는 플랫폼이 동작하는 장치로 전송하여 프로세싱하는 방식을 제안한다. 이러한 방식에서는 모바일 장치가 에이전트를 갖고 있기 때문에 기존의 시스템과 다르게 고정된 기반 시설이 에이전트 서비스를 모두 갖고 있어야 할 필요가 없으며, 에이전트가 소유자의 의도대로 동작 가능하다. 또한 이 에이전트는 필요에 따라 (에이전트가 동작한 장치가 지원하고 해당 장치에 의해 허락된다면)웹에 접근할 수도 있다. 해당 에이전트가 동작한 플랫폼에서는 동작한 결과를 XML의 형태로 생성하여 모바일 장치에 존재하는 XML 파서를 통해 프로세싱 결과를 적용한다. 이 결과는 사용자와의 대화 형태를 띨 수도 있고, 해당 모바일 장치에 설정 변경 등을 적용하는 형태일 수도 있다.

대행 서비스를 요청하는 장치가 원하는 서비스를 MTS통해 인접한 단말기에 요청하면 AMS에 자동으로 대행 서비스 장치라는 정보가 등록이 되고 이후 장치에 전송되는 메시지는 모두 XML 메시지 형태로 변경되어

제공 되어진다. 그리고 요청한 장치는 XML 파서를 통해 메시지를 분석하고 결과대로 화면에 출력하도록 한다. 사용자의 응답이 있을 때 역시 단말기는 XML 메시지를 생성하여 전송한다.

3.5 에드혹 네트워크에서의 서비스 제공자 설정

고정된 기반 시설이 존재하는 시스템에서 고정된 장치는 컴퓨팅 파워나 네트워크 대역폭의 크기가 크기 때문에, 구성된 네트워크의 화이트, 옐로우 페이지 서비스를 제공하는데 문제가 없다. 하지만 모바일 장치만으로 구성된 에드혹 네트워크에서는 이러한 서비스 제공에 하드웨어적인 한계가 존재한다. 비록 에드혹 네트워크 환경에서의 통신과 계산 양은 고정된 기반 시설의 그것에 비해 훨씬 적기는 하지만, 서비스 제공 장치는 하드웨어적으로 그 네트워크 내에서 가장 성능이 좋을수록 유리하다. 물론 이때 하드웨어의 성능은 컴퓨팅 파워가 최우선적으로 고려되어야 하지만, 모바일 장치의 한계가 존재하므로 배터리의 파워도 고려해야 한다. 따라서 최초 네트워크가 구성될 때, AMS에 등록되는 단말기들은 AMS 자체적으로 컴퓨팅 파워를 측정하고 우선순위대로 테이블에 담게 되고, 장치 중 더 컴퓨팅 파워가 큰 것이 서비스를 제공하기 시작하며, 네트워크에 새로이 접속하는 장치가 있을 때마다 AMS는 우선순위를 계속 갱신하게 된다. 그리고 갱신된 정보를 통해 서비스 관리자를 변경하거나, 차기 서비스 제공자 우선순위를 변경한다. 이때는 차기 서비스 제공자가 네트워크에 머무는 시간이나 신뢰성의 측면 등을 고려하여 서비스 제공자 권한을 변경할 것인지 아닌지를 결정해야 한다.

4. 평 가

본 논문에서 제안하는 시스템은 블루투스와 무선랜에 기반하고 있다. 블루투스를 사용하게 되면 장치 검색이 이루어진 후에 서비스 검색을 진행하게 되며, 서비스 검색을 통해 에이전트 서비스를 제공하는지의 여부를 판단할 수 있다. 이러한 서비스 검색 시에 해당 네트워크의 AMS, DF 서비스 제공자의 위치를 알려줌으로써 각각의 장치와 연결하여 서비스 검색을 하게 되는 블루투스의 단점을 해결할 수 있다. 그리고 서비스 제공 시 무선랜의 사용유무를 파악한 뒤 적절한 프로토콜을 통해서 서비스를 제공하게 되면, 블루투스로 서비스 제공할 때보다 비슷하거나 나은 성능을 나타내게 된다. 구현 언어는 자바이며, JVM은 IBM의 J9(KVM-JCL Personal Profile 1.0)[8]을 사용하였다. 블루투스 스택은 aventana[9]의 라이브러리를 사용하였다. 본 평가는 블루투스 2.0 장치 3개와 1.1 장치 2개를 사용하여 측정하였다.

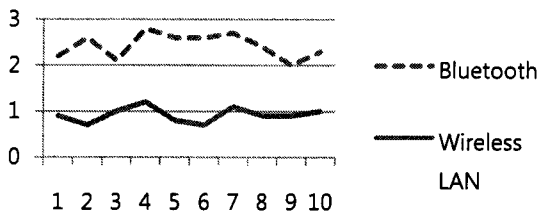

그림 7. 블루투스와 무선랜의 데이터 전송 시간 그래프

표 1. 블루투스와 무선랜의 데이터 전송 시간 표(ms)

횟 수	Bluetooth	Wireless LAN
1	2201	911
2	2633	732
3	2174	1006
4	2819	1202
5	2624	894
6	2652	787
7	2733	1127
8	2415	935
9	2364	961
10	2352	1084

평가는 100kb용량의 임시 파일을 전송하는 시간을 측정하는 방식으로 진행이 되었다. 블루투스의 경우는 데이터를 중계하는 장치를 포함한 2단계를 거쳐서 데이터가 전송이 되고, 무선랜의 경우에는 중계과정 없이 직접 통신을 하여 측정된 결과이다. 총 10회의 측정이 있었으며 결과는 최소 2배에서 최대 3.5배의 차이를 보이며 직접 통신하는 무선랜이 블루투스보다 빠른 데이터 전송을 보였다. 통상적으로 무선랜은 4~6Mbps의 속도를 보이고, 블루투스 2.0의 경우는 1~1.5Mbps의 전송 속도를 가진다고 할 때, 결과는 무선랜이 자신의 능력을 발휘하지 못했다고 할 수 있다. 이는 단말기의 무선랜이 최적화 되지 못한 것에서 이유를 찾을 수 있다. 하지만 최적화 되지 못한 상태에서도 성능의 차이를 보이는 것은 차후 최적화가 이루어지고, 더욱 많은 단계를 거치는 네트워크 환경이 조성될 때 제안한 시스템이 충분히 강점을 가질 수 있다는 것을 의미한다.

표 2. 서비스 제공자 변경 시간(ms)

구성장치 수 / 시도횟수	3	4	5
1	3532	28281	50813
2	3781	29812	51922
3	4140	23719	56312
4	4125	29328	73688
5	3922	40093	68750

(표 2)는 본 시스템에서 서비스 제공자로 동작하던 장치가 네트워크를 이탈했음을 다른 장치가 인식한 후, 우선순위상 차기 서비스 제공자로 선정된 장치가 서비스를 재개하기까지의 시간을 나타낸다. 이는 서비스 제공자의 전원이 꺼졌을 때, 다른 장치가 이를 인식하여 백업 받은 정보를 통해 동작을 시작하고 이를 네트워크 내의 다른 장치들에게 통지할 때까지로 측정되었다. 차기 서비스 제공자와 나머지 장치들이 이미 통신 하고 있는 상황이라면 서비스 제공자를 변경하는데 걸리는 시간은 아주 작다. 하지만 본 평가는 차기 서비스 제공자와 모든 나머지 장치가 새로이 연결을 진행해야 하는 최악의 상황을 가정하였다. 3개의 장치로 구성된 시스템에서는 빠른 시간 내에 전송이 가능하였지만, 네트워크를 구성하는 장치가 늘어날수록 시간이 크게 증가하였다.

표 3. 각 에이전트 플랫폼 별 사용 메모리 양

	제안 시스템	JADE-LEAP	Micro FIPA-OS
사용 메모리 양	1,231KB	3,808KB	2,357KB

(표 3)은 에이전트 플랫폼 별로 단말기에서 실행 시 사용 되는 메모리의 크기를 나타낸 표이다. 측정은 MemMaid[1]라는 프로그램을 통해서 이루어졌다. 제안 시스템과 비교대상은 JADE-LEAP[3]와 FIPA-OS[10]가 사용되었다. 결과값은 에이전트 플랫폼을 구축하고 디렉토리 서비스 등을 제공하는 에이전트의 메모리 할당량 측정값이다. 모두 JVM의 메모리 로딩까지 포함하였다. 결과는 JADE-LEAP과 FIPA-OS는 비슷한 수준이지만 제안 시스템의 경우 최대 2,577KB나 적게 측정이 되었다. 메모리를 적게 사용한다는 것은 그만큼 제한된 하드웨어 성능을 지닌 단말기에서의 사용이 용이하다는 것을 뜻한다. 이는 제안 시스템이 타 플랫폼보다 한정된 성능의 단말기로 구성되는 에드혹 네트워크에 더욱 적합하다는 의미이다.

이러한 네 가지의 실험을 통해 우리는 블루투스를 이용한 에드혹 네트워크를 기반으로 하는 에이전트 플랫폼의 구축 및 보다 효율적인 에이전트 서비스의 제공이 가능함을 확인하였다. 그리고 블루투스 자체의 데이터 전송속도는 빠르지만 최초 장치 연결 시 지연되는 시간이 단축된다면 보다 안정적인 에이전트 서비스의 유지도 기대 할 수 있다. 두 번째 실험의 경우, 브로드캐스팅을 사용한다면 변경 시간을 줄일 수 있지만 블루투스의 특성 상 차기 서비스 제공자가 피코넷 내에서 마스터로 동작해야 하는 전제가 존재한다.

[1] DinarSoft사의 MemMaid 2.0 (build 95) http://www.dinarsoft.com/memmaid/

5. 결 론

본 논문에서 우리는 블루투스 및 무선랜을 통해 구성된 애드혹 네트워크 환경에서 에이전트 관리의 당위성을 설명하고 화이트, 옐로우 페이지 서비스를 제공하는 장치의 동적인 변경을 제안하였다. 또한 이러한 상황에서 발생할 수 있는 블루투스 프로토콜로 구성된 다수의 경로를 무선랜 프로토콜로 전환하여 효율적인 서비스가 가능하게 하는 방법과 에이전트 관리 서비스 제공자의 이탈에 대처하는 방법, 에이전트 플랫폼이 동작할 수 없는 장치에 대한 에이전트 서비스를 제시하였다.

6. 참고 문헌

[1] Michael Berger and Michael Watzke, "AdHoc Proposal - Reviewed Draft for FIPA 25", Response to 1st AdHoc Call for Technology, May 2002, http://www.fipa.org/docs/input/f-in-00064.

[2] IEEE Foundation for Intelligent Physical Agents (FIPA). Agent Management Specification. http://www.fipa.org/specs/fipa00023/SC00023K.pdf

[3] Java Agent DEvelopment Framework (JADE). JADE-LEAP User Guide.

[4] Alessandro Genco, Salvatore Sorce, Giuseppe Reina, Giuseppe Santoro, "An Agent-Based Service Network for Personal Mobile Devices," IEEE Pervasive Computing, vol. 05, no. 2, pp. 54-61, Apr-Jun, 2006.
http://jade.tilab.com/doc/LEAPUserGuide.pdf

[5] Petteri Alahuhta, Henri Lothman, Heli Helaakoski, "Apricot Agent Platform for User-Friendly Mobile Service Development", University of Oulu, Linnanmaa, PL 4500, 90014 Oulun yliopisto, Finland, 2005

[6] Bryl V., Giorgini P., Fante S., "An Implemented Prototype of Bluetooth-based Multi-Agent System". In: WOA 2005: Dagli Oggetti agli Agenti. 6th AI*IA/TABOO Joint Workshop "From Objects to Agents": Simulation and Formal Analysis of Complex Systems, 14-16 November 2005.

[7] C. Carabelea and O. Boissier, "Multi-agent platforms on smart devices : Dream or reality?" in Proceedings of the Smart Objects Conference (SOC03), Grenoble, France, 2003

[8] IBM J9 KVM - Workplace Client Technology, Micro Edition http://www-306.ibm.com/software/wireless/wctme/

[9] Avetana JSR-82 implementation. http://www.avetana-gmbh.de/avetana-gmbh/produkte/jsr82.eng.xml

[10] S. Poslad, P. Buckle and R. Hadingham, "The FIPA-OS Agent Platform:Open Source for Open Standards", Manchester, UK, April 2000