퍼지 및 지능시스템학회 논문지 2005, Vol. 15, No. 4, pp. 479-485

지능형 에이전트 기반의 3-Tier 컨텍스트 인식 처리 서버/클라이언트 구조

The Structure of 3-Tirer Context-awareness Processing Server/Client based Intelligent Agents

윤효근* · 이상용**

Hyo-Gun Yun · Sang-Yong Lee

* 공주대학교 컴퓨터공학과

** 공주대학교 컴퓨터공학부

요 약

현재 컴퓨팅 기술은 유비쿼터스 컴퓨팅 환경에서 컨텍스트 인식을 위한 지능화된 시스템 구조를 요구하고 있다. 컨텍스트를 인식하기 위한 지능화된 시스템은 에이전트를 기반으로 하고 있으며, 사용자를 인식하기 위한 센서의 정보와 서비스를 지원하기 위한 프레임을 필요로 한다.

따라서 본 논문에서는 각 센서와 서비스들을 동적으로 연결하고, 각종 컨텍스트를 안정적으로 지원할 수 있는 3-Tier 컨텍스트 인식 처리 서버/클라이언트 구조를 제안한다. 제안하는 시스템의 구조는 사용자의 상황 정보를 인식하는 클라이언트 계층과 인식된 상황정보를 처리하기 위한 응용처리 에이전트 서버(APAS) 계층, 이 두 계층을 관리하기 위한 관리서버(Management Server)로 구성된다. 또한 이 구조에서는 정교한 서비스를 제공하기 위하여 사용자의 정보는 동적 프로파일로 구성되어 있다.

Abstract

Recently, computing technology requires intelligent system structures for context-awareness in ubiquitous computing environment. An intelligent system for context-awareness is based on agents, and need sensor information to recognize users and frames to support service.

Therefore, this paper proposes the structure of 3-tier context-awareness processing server/client that can connect dynamically with each sensor and service, and support various context stably. The structure of a proposal system is composed of a client class that recognizes uses' context information, a server class that processes realized context information by an application processing agent, and a management server that manages these two classes. Also, in this structure users information is composed of dynamic profile to support exquisite service.

Key words : 지능형 에이전트, 컨텍스트 인식 구조, 유비쿼터스 컴퓨팅, 서버/클라이언트 구조

1. 서 론

유비쿼터스 컴퓨팅(Ubiquitios Computing)의 배경은 휴대용 정보 통신 기기와 가전 제품 등이 인간 생활의 편리성과 다양성을 위해 하나의 네트워크로 연결되면서 연구되기 시작하였다. 유비쿼터스 환경에서 사용자는 언제, 어디서나 인터넷 서비스 및 생활 정보 서비스 등을 필요에 따라 지원받을 수 있다[1][2]. 이러한 지능적인 서비스 처리를 위해서는 사용자 주변의 센서로부터 사용자의 상태 및 상황(Context) 정보를 인식하는 컨텍스트 인식 과정이 매우 중요하다. 그리고 이때 생성된 상황 정보(Context information)는 컨텍스트 인식 및 처리시스템에 의해 분석되어 사용자에게 적합한 개별화된 서비스를 제공하는데 이용된다[3].

컨텍스트 인식을 위한 대표적인 시스템으로는 2-Tier 서버/클라이언트 구조가 있으며, 유비쿼터스 컴퓨팅 환경에서 센서와 서비스의 동적인 연결을 위해 사용되고 있다. 이 시스템은 동적인 컨텍스트 인식을 위해 다양한 컨텍스트 정보를 입력받고 있다. 하지만 컨텍스트 정보에 대한 표현력이 부족하며, 사용자의 관심을 예측하고 해당 시점에 맞는 서비스를 지능적으로 판단하고 제공하는 능력이 부족하다. 그리고 수많은 센서와 응용 서비스의 빈번한 추가, 삭제, 변경으로 인해 별도로 존재하는 리소스의 관리가 어려우며, 환경 변화에 따라 시스템 간의 네트워크 과부하가 발생할 수 있다[3][4].

이러한 문제점을 해결하기 위해, 본 논문에서는 지능형 에이전트 기반의 3-Tier 컨텍스트 인식 처리 서버/클라이언트 구조를 제안한다. 제안하는 시스템의 구조는 기존 2-Tier 서버/클라이언트 구조의 문제점을 해결하기 위해 센서와 서비스의 동적 연결성 및 분산 관리 기능을 지능형 에이전트에게 부여함으로써, 각종 컨텍스트 정보가 급속히 증가할 경우에도 안정된 서비스를 제공할 수 있도록 하였다. 또한 지능형 에이전트는 컨텍스트에 따른 차별화된 서비스를 제공하기 위하여 동적인 컴퓨팅 환경에서 서비스에 대한 우선 순위를 부여하였고, 인식된 컨텍스트 정보의 체계적인 표현도 지원하도록 하였다. 그럼으로써 사용자의 감정이나 의도를 파악해

접수일자 : 2005년 3월 11일
완료일자 : 2005년 5월 11일

지능적이고 개인화된 서비스를 제공하고자 한다.

2. 관련 연구

2.1 지능형 에이전트

유비쿼터스 컴퓨팅 환경에서 지능형 에이전트는 주어진 환경을 인식하고, 필요한 목적을 성취하기 위하여 적절한 행위를 선택하는 능력이 요구된다. MIT의 Maes는 이러한 지능형 에이전트를 동적이고 복잡한 환경에서 일련의 목적을 만족시키는 시스템으로 정의하였으며, 센서를 통해서 사용자의 환경을 파악하고 적합한 행동을 수행한다고 하였다[5].

지능형 에이전트에 의한 컨텍스트 기반 정보 서비스와 일반적인 인터넷 정보 서비스와는 많은 차이점을 가지고 있다. 첫째로 사용자는 노골적으로 서비스 및 그 내용을 요구하지 않는다는 것이다. 둘째로 사용자는 자신에 대한 정보를 몰라도, 적절한 서비스를 받을 수 있을 것이라는 기대감을 갖고 있다. 셋째로 사용자는 불필요한 서비스를 제거한 자신에 맞는 맞춤형 서비스를 제공받기를 원하고 있다는 것이다.

유비쿼터스 컴퓨팅 환경에서 지능형 에이전트의 연구는 사용자 컨텍스트를 자동으로 인식하기 위한 컨텍스트 구조 연구와 사용자의 요구(Need) 파악 방법, 지능적인 서비스 지원 방법에 대한 연구가 필요하다.

현재 지능형 에이전트의 표준화를 다루는 기관은 FIPA(Foundation for Intelligent Physical Agent)[6]와 JCP(Java Communtiy Process)로, 에이전트간의 원활한 통신과 작업의 효율성을 위해 FIPA-OS, JADE(Java Agent DEvelopment framework), ZEUS, AAP(April Agent Platform), JAS(Java Agent Service)등과 같은 표준안을 설정해 놓았다. 그리고 FIPA에서는 컨텍스트 인식 기반 서비스를 위한 지능형 에이전트를 장치기반(Device-oriented) 에이전트와 네트워크 기반(Network- oriented) 에이전트로 정의하였다. 장치 기반 에이전트는 사용자에 대한 컨텍스트를 생성하고 관리하며, 주위 환경에 대한 적응적 인터페이스 제공 및 여러 사용자들에게 서비스나 장치들을 연결시켜 협업이 가능하게 한다. 네트워크 기반 에이전트는 사용자와 센서를 비롯한 여러 장치들간의 중재 역할을 수행한다.

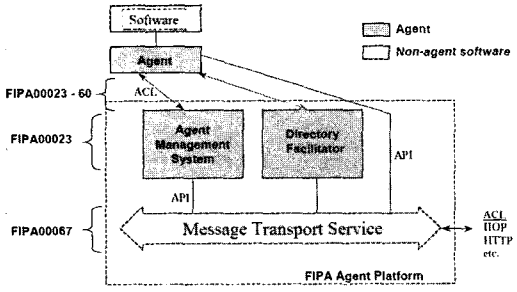

그림 1 FIPA 에이전트 플랫폼
Fig. 1 FIPA Agent Platform

[그림 1]은 FIPA-OS의 에이전트 프레임워크를 나타낸 것으로 에이전트 플랫폼(AP:Agent Platform)을 시스템의 기본단위로 하고 있다. AMS(Agnet Management System)는 에이전트의 등록, 동적 구동, 그리고 종료와 같은 라이프 사이클을 관리하며, DF(Directory Facilitator)는 각 에이전트의 능력을 평가하고 필요한 에이전트 정보를 제공한다.

2.2 컨텍스트 인식 기술

컨텍스트는 사용자가 처한 환경에서 사용자의 현재 위치, 행동 및 작업 등 사용자에 대한 정보 값과 그 정보들의 변화를 의미하며, 컨텍스트 인식은 사용자의 환경으로부터 상황정보를 얻어내는 기술을 말한다. 그리고 컨텍스트 인식을 위한 정보 구조는 응용 분야마다 다양하며 사용자 ID, 위치, 시간, 온도, 심리적 요소 등을 주로 이용하고 있다[7][8].

컨텍스트 인식 시스템은 주변 환경을 감지하여 컨텍스트를 파악하고, 컨텍스트 정보에 따라 적절한 서비스를 제공하는 것과 컨텍스트에 맞추어 시스템의 실행 조건이나 주변 환경을 스스로 변경하는 것으로 나뉜다. 또한, 여러 종류의 컨텍스트 인식 시스템이 갖는 공통적 특징을 기반으로 하는 분류 방법들이 있다. 이 분류 방법들은 사용자의 컨텍스트와 관련된 서비스 자원들을 사용자가 선택하도록 제공하는 서비스 선택, 컨텍스트의 변화에 맞춰 적절한 서비스를 자동으로 실행시키는 서비스 자동 실행, 다양한 디스플레이어 장치를 이용하여 정보를 나타내는 정보 디스플레이어, 정보를 대상물에 직접 증강시키는 정보 증강 등으로 분류되며 대부분의 컨텍스트 인식 시스템들을 이 범주로 구분할 수 있다.

유비쿼터스 컴퓨팅 분야의 컨텍스트 인식 기술에 대한 중요성이 증가함에 따라 컨텍스트 인식 시스템과 응용 서비스에 대한 개발이 활발하게 진행되고 있다[8][9][10]. 현재 사용되고 있는 컨텍스트 인식 모형으로는 GATEH의 Context Toolkit[8]과 Couder와 Kermaree[9]의 컨텍스트 인식을 처리하는 일반적인 구조와 컨텍스트 객체를 표현하는 모델(Context Object Model)이 있다. 또한 컨텍스트의 효율적인 관리와 사용자에게 맞는 응용 서비스를 지능적으로 제공하는 ubi-UCAM 모델[10] 등이 있다.

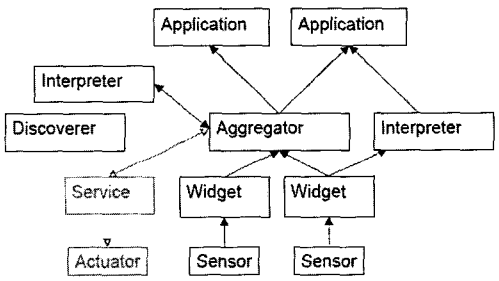

그림 2 Context ToolKit(GATEH)
Fig. 2 Context ToolKit(GATEH)

[그림 2]는 GATEH의 Context Toolkit으로 이 구조는 센서와 응용서비스 사이의 종속성 문제를 해결하기 위해 센서와 서비스 사이에 컨텍스트를 관리하는 중간 매개체를 사용하고 있는 것이 특징이다. 먼저 Widget는 센서로부터 받은 실제 세부 컨텍스트를 신뢰성이 있는 것으로 분리해 Aggregator 또는 또 다른 Context Components에 보내는 역할을 한다. Aggregator는 Context Server개념으로 Widget에 온 정보를 승인하고, 모든 동작(속성, 횟수, 서비스)들을 집합으로 유지하며, 리소스를 발견하면 추가하거나, 자동적으로 확장한다. Interpreter는 컨텍스트를 다른 곳으로 전환할 때 신뢰성이 있게 해석해 주며, 입·출력을 할 경우 새로운 컨텍스트 정보를 유지한다. 하지만 이 구조는 응용 서비스가 컨텍스트를 사용하기 위해서는 중간 매개체를 새로 작성하거나 기존의 중간 매개체와 복잡하게 연결해야 하는 문제점이 발생한다.

2.3 2-Tire 서버/클라이언트 구조

Couder와 Kermarrec는 상황 인식을 위한 일반적인 구조와 상황 객체를 표현하는 모델로 IRISA/INRIA를 제안하였다[9]. 이 구조는 2-Tier 서버/클라이언트 구조로 구성되어 있으며, [그림 3]에서 보는 것과 같다. 클라이언트는 세 개의 계층으로 이루어져 있으며, 하위 계층인 탐지(Detection)/통지(Notification)계층으로 시스템과 네트워크를 모니터링하여 상황을 탐지하며 상위 계층으로 통지해 준다. 중간 계층은 적응적 계층(Adaptive Layer)으로 상황 정보를 관리하고 선택하여 서버에 저장한다. 마지막으로 상위계층은 응용(Application)으로 구성되어 있다. 서버에서는 정보 서버 역할을 수행하여 상황정보를 저장하고 관리하며 클라이언트 요청에 응답한다.

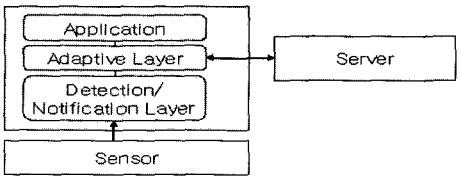

그림 3 2-Tier 서버/클라이언트 구조
Fig. 3 2-Tier Server/Client structure

기존의 2-Tier 서버/클라이언트 구조는 유비쿼터스 환경의 특성상 많은 양의 컨텍스트가 동시 다발적으로 발생했을 때, 컨텍스트 정보에 대한 표현력이 부족하며, 사용자의 상황에 적합한 지능적인 서비스를 제공하는 것이 곤란하다. 그리고 수많은 센서와 응용 서비스를 위한 별도의 리소스 관리가 어려우며, 컨텍스트 인식 환경의 변화에 따라 시스템 간의 네트워크 과부하가 발생할 수 있다.

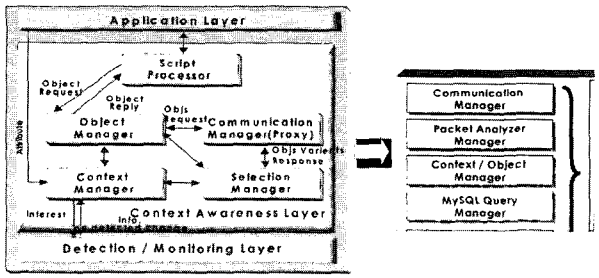

그림 4 상황인식 처리를 위한 미들웨어
Fig. 4 A middleware for context-awareness process

[그림 4]은 한국전자통신연구원의 임베디드 S/W기술센터에서 Couder와 Kermarrec가 제안한 IRISA/INRIA구조의 미들웨어 모델을 참조하여 설계한 것으로 상황 정의 스크립트 처리기를 부가적으로 추가하여 범용성을 높였다[11].

3. 3-Tier 컨텍스트 인식 서버/클라이언트

본 논문은 유비쿼터스 환경에서 사용자의 컨테스트에 대하여 자율성과 상황에 따른 효과적인 반응성, 스스로의 학습과 협력 및 이동 능력을 고려하여 지능형 에이전트를 기반으로한 사용자의 컨텍스트 인식 및 처리 시스템을 제안한다. 본 논문에서 제안한 시스템의 구조는 사용자의 컨텍스트를 효과적으로 인식하고 관리하기 위해 세 개의 영역으로 구분되며, 지능형 에이전트를 기반으로 구성된다.

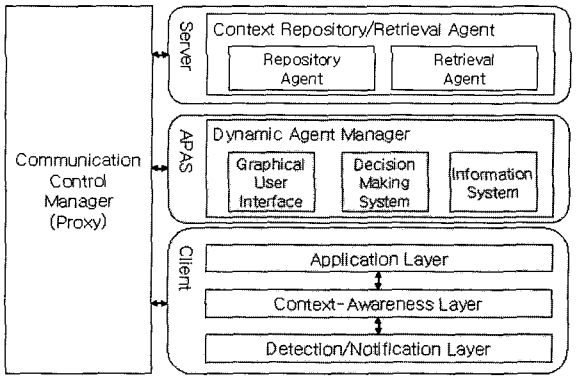

그림 5 3-Tier 컨텍스트 인식 서버/클라이언트 구조
Fig. 5 Structure of 3-Tire Context-awareness Server/Client

각 영역은 [그림 5]에서 보는 것과 같이 사용자의 컨텍스트를 인식하고 관리하기 위한 클라이언트(Client) 영역과 이를 바탕으로 사용자에게 필요한 에이전트를 동적으로 생성하고 관리하기 위한 응용 처리 에이전트 서버(APAS: Aplication Processing Agents Server) 영역, 클라이언트와 응용처리 에이전트 서버를 관리하기 위한 서버(Server) 영역으로 구성된다. 그리고 각 영역 간의 통신은 통신 제어 관리자(Communication Control Manager)를 이용한다.

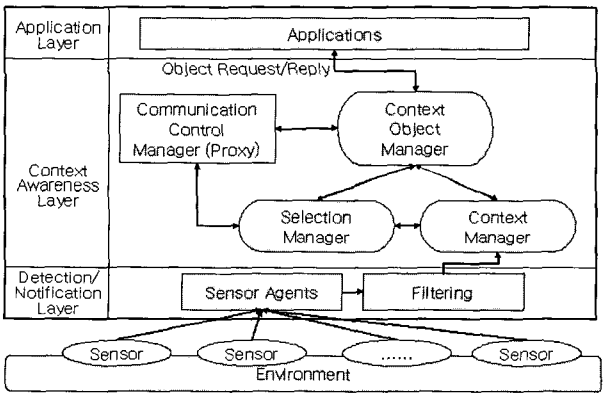

그림 6 클라이언트 구조
Fig. 6 Client structure

[그림 6]을 보면, 클라이언트 구조는 탐색 계층(Detection/Notification Layer), 컨텍스트 인식 계층(Context Awareness Layer), 응용 계층(Application Layer)과 같이 3가지 계층으로 이루어진다. 먼저 최하위층인 탐색 계층은 센서 에이전트들(Sensor Agents)을 통해 사용자와 사용자가 위치해 있는 장소와 환경에 대한 변화 정보 즉, 컨텍스트 정보를 감지한다. 감지된 각종 컨텍스트 정보는 협력적 필터링(collaborative filtering) 과정을 거쳐 컨텍스트 관리자(Context Manager)에 전달되며, 필터링된 컨텍스트 정보는 서버에 있는 컨텍스트 저장 에이전트(Context Repository Agent)에 의해 저장된다. 저장된 컨텍스트 정보는 이전의 컨텍스트 정보와 비교하여 일치되는 컨텍스트 정보를 가지고 있는지 확인하여 컨텍스트 관리자에게 통보한다. 협력적 필터링은 "사용자의 행동 패턴은 일정한 규칙을 가진다" 라는 연관 규칙에 피어슨 상관계수와 표준편차를 적용하였다.

클라이언트의 중간 계층인 컨텍스트 인식 계층은 컨텍스

트 관리자(Context Manager), 선택 관리자(Selection Manager), 컨텍스트 객체 관리자(Context Object Manager)로 구성되어 있으며, 각 영역과의 통신을 위해 통신 제어 관리자(Communication Control Manager)로 구성된다. 컨텍스트 관리자는 필터링된 사용자의 취향, 기기의 성능, 현재 위치 등을 포함하고 있는 주어진 환경에 대한 컨텍스트 정보를 관리하며, 해시 테이블(Hash Table)형식을 이용하여 컨텍스트를 통합하고 저장한다. 해시 테이블은 키(key)와 값(value)으로 하며, 컨텍스트 조건과 서비스 모듈간 1:1 관계와 n:1관계성을 제공하여 정보를 효과적으로 관리한다. 또한 컨텍스트 정보가 지능형 에이전트에 의해 사용될 수 있도록 5WIH의 통일된 형태로 컨텍스트를 정형화한다. 5WIH의 형식은 누가(Who), 언제(When), 어디서(Where), 무엇을(What), 어떻게(How), 왜(Why)를 나타내는 정보로써, 사용자 신원, 사용자 위치, 시간 정보 등과 연관된 정보를 제공하거나 특정 시점에 발생되는 사건을 저장했다가 다시 재생시키는 등의 형태를 나타낸다.

컨텍스트 객체 관리자는 응용 프로그램에서 사용되고 있는 컨텍스트 객체에 포함된 모든 정보를 위한 데이터 구조를 관리한다. 또한 응용 계층(Application Layer)로부터 특정한 객체를 요구할 경우, 선택 관리자로 요구(Request) 정보를 보낸다. 그리고 응용 프로그램이 객체를 참조할 때, 서버로부터 정보를 검색하거나, 응용 처리 에이전트 서버(Application Processing Agents Server)를 이용해서 적합한 컨텍스트 정보를 응답(Reply)하는 역할을 수행한다. 선택 관리자(Selection Manager)는 응용계층(Application Layer)에서 특정한 객체를 요구할 경우, 객체 리스트의 속성 값과 컨텍스트 관리자의 현재 상황 명세를 사용하여 가장 유사한 상황정보를 찾아낸다.

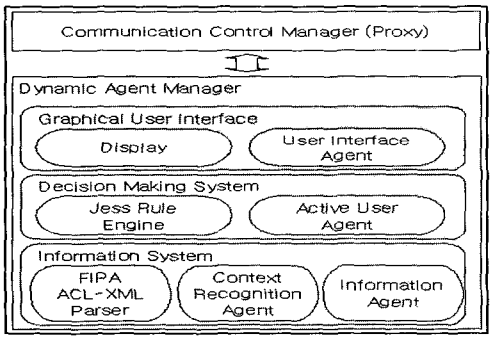

그림 7 응용처리 에이전트 서버(APAS) 구조
Fig. 7 APAS structure

응용 계층(Application Layer)은 하부의 컨텍스트 인식 계층과 독립적으로 동작하면서 컨텍스트 인식 API를 통해 상황인식 처리 기술에 기반을 둔 다양한 응용 프로그램을 개발하기 위한 기능을 수행한다. 응용 처리 에이전트 서버(APAS)는 [그림 7]에서 보는 것과 같이 동적 에이전트 관리자와 통신 제어 관리자로 구성된다. 동적 에이전트 관리자(Dynamic Agent Manager)는 각 에이전트들간에 상호교류를 통해 초기 컨텍스트들을 결합하여 보다 신뢰성이 높은 사용자 상황정보를 파악하도록 하였다. 또한 동적 에이전트 관리자에서는 적절한 컨텍스트 조건을 스스로 변경하도록 하여 작업의 효율성을 높였으며, 각 에이전트는 공통의 업무를 분배해서 처리하거나 각기 다른 일을 맡아서 처리한 후 그 결과를 분석하여 문제를 해결할 수 있도록 하였다.

동적 에이전트 관리자는 사용자 인터페이스 에이전트(User Interface Agent), 활성화 사용자 에이전트(Active User Agent), 정보 에이전트(Information Agent)로 구성된다. 사용자 인터페이스 에이전트는 정형화된 컨텍스트를 통해 각 사용자를 식별한다. 그리고 사용자 식별이후의 행위에 대한 변화는 모든 센서를 이용한 메모리 낭비보다는 로컬 탐지기(Display)를 사용함으로써 메모리의 관리 및 환경 변화에 대하여 에이전트가 능동적으로 대처하도록 하였다. 활성화 사용자 에이전트는 클라이언트에서 전달된 사용자의 컨텍스트 변경 내역(history)과 JESS Rule Engine을 이용하여 학습하고, 적절한 컨텍스트 조건을 스스로 변경함으로써 사용자 중심의 서비스를 제공하도록 하였다. 또한 제공된 서비스와 컨텍스트 정보들은 서버에 있는 지식 데이터 베이스(Knowledge Database)에 저장하도록 하였다. 정보 에이전트는 각각의 에이전트를 통해 현재 상황 정보들을 받고, 구체적인 상황정보를 통해서 대응하는 서비스를 제공한다. [표 1]는 정보 에이전트를 위해 XML로 표현된 사용자 컨텍스트 정보의 예이다.

표 1 사용자 컨텍스트 예제
Table 1 An example of Users' Context

User A	**User B**
<profile> <location> <x>58</x> <y>38</y> </location> <interests> <sport>Soccer </sport> <music>Classical </music> <book>History </book> </interests> </profile>	<profile> <location> <x>58</x> <y>34</y> </location> <interests> <sport>Swimming </sport> <music>Pop </music> </interests> </profile>
P1: /profile/interests [sport = 'Swimming'] P2: /profile/interests [count(*) >= 3]	

서버영역은 [그림 8]에서 보는 것과 같이 저장에이전트(Repository Agent)와 검색에이전트(Retrieval Agent)를 포함시킨 컨텍스트 저장/검색 에이전트(Context Repository/Retrieval Agent)구성된다.

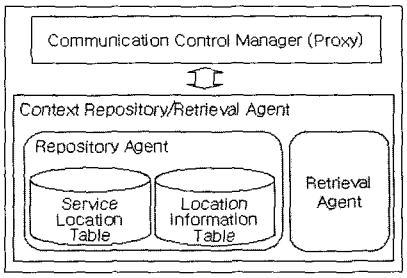

그림 8 서버 구조
Fig. 8 Server Structure

저장 에이전트는 컨텍스트에 대한 명세 및 각 명세에 따른 객체 내용을 DB에 저장하며, 검색에이전트는 저장된 컨텍스트의 명세 및 객체 내용을 검색하는 역할을 수행한다. 저장 에이전트는 네임 캐시(Name Cache)를 이용하여 디스

커버리를 위한 절차를 줄이고 서비스 목록들의 일관성을 유지하도록 하였다. 또한 검색 에이전트(Repository Agent)를 통해 서비스를 요청받고 해당 서비스를 제공하는 역할과 함께 서비스 목록이 저장되는 SLT(Service Location Table)과 LIT(Location Information Table)를 관리하는 기능을 담당한다.

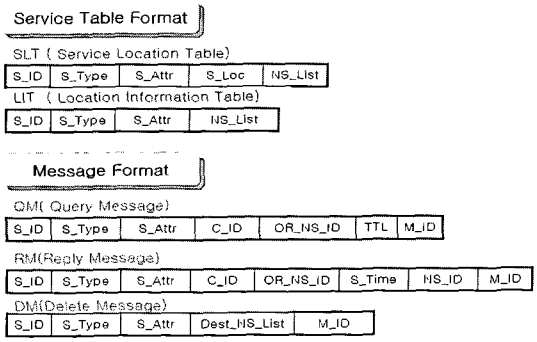

그림 9 자료 구조
Fig. 9 Data Structure

SLT과 LIT의 자료구조는 [그림 9]와 같이 구성하였고, SLT는 기존 도메인에 존재하는 서비스 네임과 동일한 서비스에 대한 네임 참조를 가지는 테이블로서, 그 생성 과정을 보면 기존 도메인에 존재하는 서비스를 먼저 SLT에 등록시키고, 동일한 서비스에 대한 네임 참조는 OR(Original Requester)가 생성시킨 QM(Query Message)을 전송하며, 그 결과에 따라 SLT에 기록하게 된다. SLT는 call back 기법을 이용해 SLT간 일관성을 유지하도록 하였다. LIT는 기존 도메인에 존재하지 않는 서비스에 대한 네임 참조를 가지는 테이블로서, 그 생성 과정을 보면 OR에서 필요한 서비스를 찾기 위해 QM을 전송하며 그 결과에 따라 LIT가 업데이트된다. LIT는 time-out 기법을 이용해 유효하지 않는 서비스 목록은 일정한 시간이 지나면 자동 삭제되도록 하였다.

검색 에이전트의 디스커버리 과정은 통신 제어 관리자를 통해 클라이언트로부터 서비스 요청을 받아 기존에 등록된 서비스인지 확인하기 위해 SLT를 검색한다. SLT에 원하는 서비스가 없으면 LIT를 검색해 다른 네임 서버에 있는 서비스임을 인식하고 인접한 네임 서버에게 전송한다.

4. 시스템 구현 및 실험

클라이언트 구현은 주어진 상황과 상황에 대한 행동을 효과적으로 처리하기 위한 상황 정의 스크립트 언어를 XML과 XPath(XML Path Language)을 이용하여 설계 및 구현하였다. 컨텍스트 객체는 센서와 장치들의 이름과 속성 값으로 이루어져 있다. 그리고 속성 값들은 컨텍스트의 조건절(condition clause)에서 사용되며, 일부는 하드웨어적인 장치에 의해서 갱신도록 하였다. 컨텍스트의 정의는 조건(condition)에 대한 상황 동작 처리를 정의함으로써, 컨텍스트에 맞는 조건절과 동작을 설정하였다. 컨텍스트의 정의 구조는 [표 2]와 같다.

표 2 컨텍스트 정의 구조
Table 2 Context definition structure

ContextDefine	context_name
Condition	(condition_expression)
Action	function_name [before/when/after time_value]or [from time_value to time_value]
Param	(param1, param2, ..., param n)

ContextDefine은 컨텍스트를 정의할 때 사용되며, Condition은 컨텍스트의 조건을 정의하고, Action은 조건절에 맞는 함수를 호출하고 Param에 의해서 매개변수가 전달된다.

표 3 시나리오 및 컨텍스트 정의 예
Tabel 3 An example of scenario and context definition

각 사용자는 선호하는 음악이 있고 특정한 방으로 이동하고 그 방에서 3초 이상 머물 경우, 1초 후에 정해진 음악을 재생한다. 예를 들어, "A" 라는 사용자가 "A1"호에 머물 경우에는 "Yesterday"라는 음악을 재생하고, "B"라는 사용자가 "A2"호에 머물 경우 "아리랑"이라는 음악을 재생한다.
/* 컨텍스트 객체와 컨텍스트 정의 */ ContextObject person(string name, string location) ContextDefine music Condition (((person.name =someone_name) And ((person.location = someone_location) during 3s)) Action PlayMusic from 1s to end Param (music_name) /*Action */ ContextInsert A into music Condition ("A", "A1") Param ("Yesterday") ContextInsert B into music/*instance_name =music */ Condition ("B", "A2") Param ("아리랑") ContextStart * in music

[표 3]은 클라이언트에 맞는 시나리오를 작성하고 컨텍스트 정의 구조에 맞춰 작성한 예이다. ContextStart 구문 사용 시 별표(*)를 사용함으로써 music의 모든 인스턴스의 조건이 체크되도록 하였다. 그리고 응용처리 에이전트 서버는 클라이언트와 서버에서 요구하는 상황정보들을 효율적으로 이용하기 위해서 UCA APAS Builder 1.0으로 구현하였으며, [표 4]는 개발 및 운용 환경이다.

표 4 UCA APAS Builder 1.0 개발 및 운용 환경
Table 4 UCA APAS Builder 1.0 development and operating environment

운영체제	Java를 지원하는 모든 시스템
개발언어	Java(J2SE 1.4)
서비스 언어	XML
라이브러리	OpenJgraph 라이브러리
개발 툴(Tool)	Jbuilder X

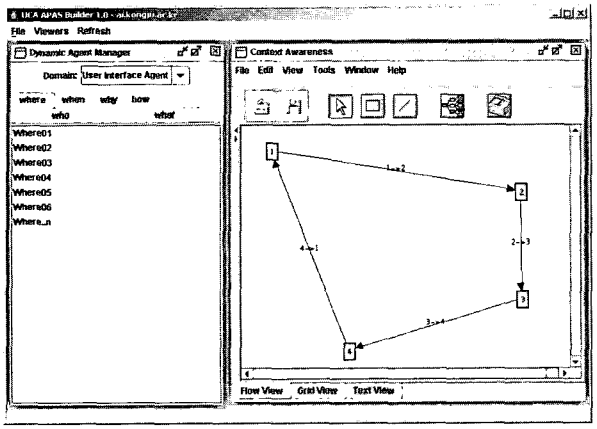

그림 10 UCA APAS Builder 1.0
Fig. 10 UCA APAS Builder 1.0

[그림 10]은 자체적으로 제작된 컨텍스트 인식을 위한 에이전트 개발 빌더이며, 각 노드는 사용자의 이동 경로 및 목적에 대한 컨텍스트 정보를 인식한 후 가중치를 부여한 것이다. 이 빌더는 지역 내에 필요한 센서를 추가함으로써, 사용자의 컨텍스트 정보를 능동적으로 수집하고 인식하는 과정으로 이루어져 있다. 사용자의 인식과 이동에 대한 초벌 컨텍스트 정보의 인식 과정을 GUI 모델로 확인이 가능하다. 또한 개발한 빌더에서는 설계한 컨텍스트 인식 모델에서 필요한 컨텍스트 정보를 임의로 생성하고, 센서가 이동함으로써 필요한 컨텍스트 정보를 인식하기 위한 에이전트의 반응을 성능 평가하고 있다.

서버의 구현은 컨텍스트 인식을 위한 원격 객체에 대한 정보와 원격 객체로부터 파생되고 추출할 수 있는 컨텍스트 인식 정보를 저장하고 검색할 수 있는 데이터베이스를 주로 한다. 이를 위하여 기존의 데이터 베이스 관리 시스템을 이용하여 구축하였다. 따라서 본 논문에서는 MS-SQL를 사용하여 컨텍스트 정보의 저장과 객체 검색 및 서버스 검색에 적용하였다. 또한 서버는 클라이언트와 APAS로부터 전달되는 패킷의 내용을 분석하고, 분석된 패킷의 내용을 수행한다. 이를 위해서는 이미 FIPA에서 규정된 표준안을 가지고 패킷을 적용한다. [표 5]은 서버에서 관리자 및 에이전트의 작업을 위해 사용된 API로 구성된 모듈을 나타낸 것이다.

표 5 서버 인터페이스 모듈
Table 5 Sever interface module

구성요소	www.trueebook.com역 할
메인 서버 모듈	서버의 시작 및 TCP/IP 소켓을 통해 통신을 하기 위한 Communication Manager 모듈
패킷 분석 모듈	Packet Analyzer Manager 부분으로 클라이언트나 APAS로부터 받은 패킷을 분석하여 해당 컨텍스트 API를 호출
컨텍스트 API	컨텍스트 API로서 DB 및 클라이언트, APAS와 상호작용을 위한 라이브러리이며 서버측 API에 해당
파일 수신 모듈	서버가 클라이언트와 APAS에게 파일을 전송하는 함수 라이브러리
기타 실행 모듈	객체의 상황에 대한 사용자 정의 실행 모듈

구현된 시스템의 평가를 위해 본 논문에서는 센서를 통해 사용자의 컨텍스트 정보를 인식하고 서비스를 제공하기까지의 네트워크 지연시간을 평균값으로 하여 기존 2-Tier 서버/클라이언트 구조와 비교 평가하였다. 실험은 10명의 사용자를 대상으로 10회 실시하였고, 실험 환경은 [표 6]에서는 보는 것과 같이 구성하였다.

표 6 실험 환경
Table 6 Test environment

Server	
OS	Microsoft Window XP pro
DBMS	SQL Server 2000
System	PentiumIV2.8, Memory 1024M
Client	
OS	Window XP home
System	PentiumIV2.0, Memory 512M

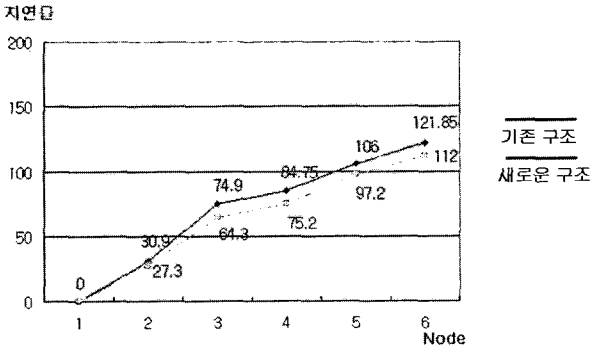

그림 11 노드당 지연 시간 비교
Fig. 11 Comparison of delay time per node

[그림 11]은 기존의 시스템과 노드 수의 증가에 따른 네트워크 지연시간을 비교 평가한 것이다. 기존의 2-Tier 서버/클라이언트 구조의 네트워크 지연 시간은 컨텍스트들의 개당 증가 수에 따라 지연 시간이 약 21ms 정도로 증가하였다. 컨텍스트 수집을 위한 시간이 500ms(센서의 감지 주기를 기반)라고 한다면, 하나의 그룹당 연결 가능한 노드의 수는 22개 이상이다. 이는 유비쿼터스 컴퓨팅 환경에서는 요구되는 조건, 즉 다양한 컨텍스트의 인식을 위한 노드의 이동을 충족시키지 못한다. 또한 네트워크의 과부하 및 리소스 관리에 대한 문제점이 발생할 수 있는 가능성을 포함하고 있다. 하지만 그래프에서 보면 제안한 시스템의 구조는 기존의 구조보다 노드당 지연시간이 평균 7%만큼 감소되는 것을 확인할 수 있다. 이는 APAS에서 수집된 노드 정보들을 공유시킴으로써, 사용자의 다음 행동을 미리 예측할 수 있었기 때문이다. 그리고 기존의 시스템보다 컨텍스트 인식과 서비스 지원이 빠른 것을 확인할 수 있었다. 또한 APAS가 클라이언트에서 전달된 사용자 컨텍스트의 조건 변경 내역(history)과 서버에 있는 컨텍스트 정보를 학습하는 과정이 네트워크에 무리를 주지 않고 있음을 확인하였다.

6. 결론 및 향후 연구과제

유비쿼터스 환경에서 사용자에 대한 지능적인 환경 인식 및 서비스를 지원하기 위해서는 지능형 에이전트를 통한 컨텍스트 인식 기술이 매우 중요하다. 이를 위해 기존의 컨텍스트 인식 시스템 구조인 2-Tier 서버/클라이언트 구조는 동적인 컨텍스트 처리 및 리소스 관리가 힘들며, 네트워크 과부하에 대한 문제를 가지고 있다.

따라서 본 논문에서 지능형 에이전트를 기반으로 3-Tier 컨텍스트 인식 처리 서버/클라이언트 구조를 제시하였다. 제안한 시스템은 기존의 2-Tier 서버/클라이언트 구조에서 지능형 에이전트로 구성된 응용 처리 에이전트 서버(APAS)를 추가함으로써 확장성과 안정성, 효율성을 고려하였다. 그리고 동적인 환경에서도 각종 컨텍스트 정보를 주기적으로 학습함으로써 새로운 컨텍스트 인식 조건을 변경하도록 설계하였다.

이를 평가하기 위해서 실험한 결과, 사용자의 컨텍스트를 인식하고 서비스를 제공하기까지의 네트워크 지연시간이 기존의 시스템보다 평균 7% 정도 감소하였다. 또한 제안한 시스템이 각 사용자에 대한 컨텍스트 인식 시간에서도 기존의 시스템보다 빠르게 인식되었으며, 서비스 지원에서 발생할 수 있는 네트워크 과부하는 지능형 에이전트들에 의해 능동적으로 대처할 수 있는 가능성을 확인하였다.

향후, 제안한 시스템에 대한 다양한 환경에서의 성능 평가와 동적인 환경에 적합한 에이전트 구조에 대한 고려가 필요할 것이다.

참 고 문 헌

[1] Mark Weiser "The Computer for the Twenty-First Century", Scientific American, pp. 94-101 September 1991.

[2] Mark Weiser http://www.ubiq.com/hypertext/ weiser/UbiHome.html

[3] Thomas P.Moran, Paul Dourish, " Introduction to This Special Issue on Context-Aware Computing", HCI, vol.16, pp.87-96, 2001.

[4] Andreas Wennlund, "Context-aware Wearable Device for Reconfigurable Application Networks", Department of Microelectronics and Information Technology (IMIT) 2003, April 2003.

[5] J. R. Quinlan, C4.5 Programs for Machine Learning, San Mateo,CA:Morgan, Kaufaman.

[6] Foundation for Intelligent Physical Agent(FIPA) http://www.fipa.org

[7] A.K.Dey and G.D.Abowd, "Towards a Better Understanding of Context and Context-Awareness", GVU Technical Report GIT-GVU -99-22. Submitted to the 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99), June 1999.

[8] D.Salber, A.K.Dey and G.D.Abowd, "The Context Toolkit:Aiding the Development of Context-Aware Applications", In the Workshop on Software Engineering for Wearable and Pervasive Computing (Limerick Ireland), Jun 2000.

[9] P.Couder, A.M.kermarrec, "Improving Level of Service of Mobile User Using Context-Awareness", 18th IEEE Symposium on Reliable Distributed System, pp.24-33, 1999.

[10] S.Jang, W.Woo, "ubi-UCAM:A Unified Context-Aware Application Model.", LNAI(Contex03), pp.178-189, 2003.

[11] 김용기, 김영국, 심춘보, 장재우, 김정기, 박승민, "상황인식 처리를 위한 미들웨어 및 컨텍스트 서버의 설계 및 구현", 정보과학회 2004 춘계학술대회, vol 31,no 01, pp.139-141, 2004.04

저 자 소 개

윤효근(Hyo-Gun Yun)

1999년 : 한밭대학교(구 대전산업대학교) 전산과(학사)
2002년 : 공주대학교 대학원 전자계산학과 (이학석사)
2004년 : 공주대학교 대학원 컴퓨터 공학과 박사과정 수료

관심분야 : 유비쿼터스 컴퓨팅, 인공생명, 에이전트 시스템, 개인화서비스 등
e-mail : kosher@kongju.ac.kr

이상용(Sang-Yong Lee)

1984년 : 중앙대학교 전자계산학과(공학사)
1988년 : 일본동경대학대학원 총합이공학 연구과(공학석사)
1988년~1989년 : 일본 NEC 중앙연구소 연구원
1993년 : 중앙대학교 일반대학원 전자계산학과(공학박사)
1996년~1997년 : University of Central Florida 방문교수
1993년~현재 : 공주대학교 컴퓨터공학부 교수

관심분야 : 인공지능, 에이전트 시스템, 컴퓨터게임, 유비쿼터스 컴퓨팅
e-mail : sylee@kongju.ac.kr