# 커뮤니티 컴퓨팅을 위한 에이전트 기반 지능형 미들웨어†

성균관대학교 한승욱 · 송성근 · 윤희용

## 1. 서 론

유비쿼터스 컴퓨팅 기술은 사용자 중심의 서비스를 제공하기 위해 센서 및 RF-ID를 활용한 자동 환경 인식 및 사물 인지와 네트워크 인프라를 구축하기 위한 네트워킹 기술, 그리고 서비스를 제공하는 플랫폼 재구성, 분산처리 등에 초점이 맞춰져 왔다. 그러나 유비쿼터스 환경에서 사용되는 고도 분산 환경의 애플리케이션은 고도화, 다양화, 분산화 되어 주변에 산재해 있는 디바이스를 활용해 서비스를 제공하는 것뿐만이 아니라 사용자에게 최적화된 서비스를 제공하기 위한 지능적인 요소까지 갖추어야 한다.

이와 같은 요구사항에 따라 에이전트를 사용하여 사용자에게 지능적인 서비스를 제공하고, 더 나아가 에이전트와 서비스 및 디바이스, 상황정보간의 연관관계를 통하여 커뮤니티를 생성하고 생성된 커뮤니티 간의 연관관계를 통하여 사용자에게 최적화된 서비스를 제공하기 위한 방법들이 제시되고 있다. 즉 각각의 최적화된 서비스 보다는 서비스를 제공하기 위한 모든 자원들이 협업하여 서비스를 제공함으로써 좀 더 지능화된 서비스를 제공하는 기술이 매우 중요하게 요구 되고 있는 실정이다.

본고에서는 **커뮤니티 컴퓨팅의 기반 기술이 되는 분산처리 미들웨어와 에이전트 플랫폼, 그리고 디스커버리 미들웨어에 대하여 알아보고, 이를 바탕으로 최종적으로 커뮤니티 컴퓨팅을 위한 에이전트 기반 지능형 미들웨어 기술의 발전 방향을 제시**하려고 한다.

## 2. 미들웨어와 에이전트 플랫폼

유비쿼터스 환경을 위한 미들웨어 시스템의 중요한 요구사항은 이기종 환경에서 실행되는 애플리케이션 간의 데이터 교환을 기반으로 상황 변화에 따라 최적의 서비스를 제공하는 것이다. 이러한 요구사항을 만족시키기 위해서는 지능적인 작업을 수행하기 위한 에이전트와 에이전트의 실행환경인 에이전트 플랫폼, 그리고 에이전트 플랫폼 자체가 상황에 능동적으로 적응할 수 있도록 하는 리플렉티브하고 어댑티브 한 요소를 가지고 있어야 한다. 이에 더하여, 효율적이고 최적화된 서비스를 지원하기 위해 사용자와 디바이스, 그리고 그룹간의 커뮤니티 컴퓨팅을 지원하는 것이 밑받침 되어야 한다.

### 2.1 분산 컴퓨팅 미들웨어

인간 친화적인 유비쿼터스 환경을 구성하기 위해서는 내장형 프로세서 기술과 유무선 통신을 위한 네트워킹 기술, 능동적 센서와 이를 연결해 주는 센서 네트워킹 기술 및 사물 인식을 위한 RF-ID기술과 내장형 실시간 운영체제 기술이 요구된다. 그리고 이와 같이 다양한 기술을 사용하여 수집한 정보를 효율적으로 교환하고 공유하기 위한 미들웨어 기술이 필수로 요구된다.

미들웨어란 운영체제와 응용 프로그램 사이에 존재하는 소프트웨어 계층으로 사용자에게 하부의 하드웨어나, 운영체제, 네트워크에 상관없이 분산 컴퓨팅 환경에서 원격 프로시져 호출(RPC), 메시징과 같은 서비스를 제공해 주는 소프트웨어로 정의할 수 있다. 본 장에서는 미들웨어의 기능에 따라 메시지 기반 미들웨어와 객체지향 미들웨어, 그리고 리플렉티브 미들웨어에 대하여 살펴보도록 한다.

#### 2.1.1 메시지 기반 미들웨어

미들웨어의 가장 기본적인 구조인 메시지 기반 미들웨어(Message Oriented Middleware-MOM)는 응용간의 비동기 통신을 지원하며 전송하는 데이터에 대하여 특정 양식으로 제한하지 않으며, 애플리케이션 내부 구조체를 사용하여 데이터를 송 · 수신하는 방법으로 구성되어 있다.

서비스 제공자와 서비스 수신자간에 데이터 송 · 수신을 위해 중간 매개체 역할을 하는 메시지 큐를 사용하며 메시지큐는 메시지 큐 브로커 또는 메시징 서버에서 큐 관리자를 통하여 관리된다.

† 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크원천기반기술개발사업의 지원에 의한 것임.

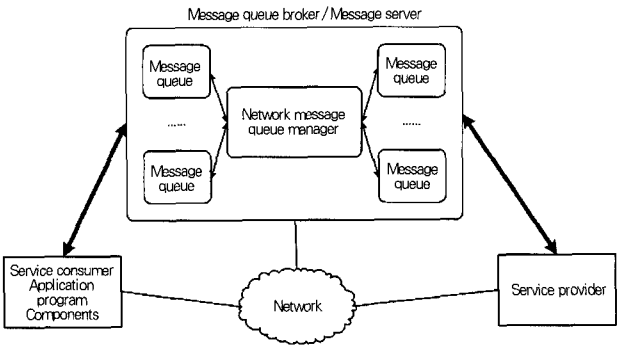


그림 1 메시지 기반 미들웨어 구조

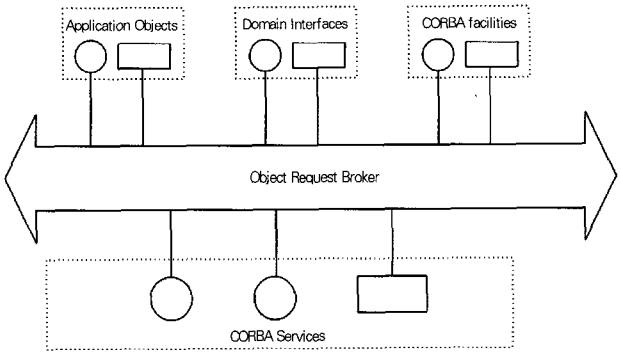


그림 2 객체 지향 미들웨어의 구조 (CORBA)[2]

이와 같이 중간 매개체 역할을 하는 큐를 사용함으로써 서비스 제공자와 서비스를 제공받는 수신자가 동일한 시간에 사용 가능해야할 필요가 없다. 또한 고장감내, 트랜잭션 및 부하분산의 추가 기능을 위하여 서비스 제공자와 수신자의 구조를 크게 변경하지 않아도 되는 장점을 가지고 있는 반면에 비동기식 메시지 처리와 응용에 따라 다른 메시지 구조를 사용하기 때문에 동기식 RPC를 사용하는 객체지향 미들웨어 보다 응용 개발자에게 투명성을 제공하지 못한다는 단점을 가지고 있다[1].

서비스 제공자가 복잡한 연산을 수행하고 요구에 따른 응답 속도가 늦을 경우에 주로 사용되며 대표적으로 IBM WebSphere의 MQ와 SUN Java Message Service(JMS), Microsoft사의 MSMQ, TIBCO 소프트웨어의 TIBCO Rendezvous 등이 있다.

### 2.1.2 객체 지향 미들웨어와 이벤트 서비스

이기종 플랫폼의 응용간의 자원 공유와 제어를 수행하는 시스템 형태인 RPC에서 발전된 모델인 객체지향 미들웨어는 ORB(Object Request Broker)를 기반으로 분산 객체 또는 컴포넌트 모델에서 가장 중요한 공통된 객체간 통신 서비스로서 분산된 객체들 간의 연결성을 만들어 주는 핵심적인 객체 미들웨어 기술이다.

객체지향 프로그래밍의 개념을 분산 환경에 접목시켜 객체간의 동기화 통신을 지원하며, 원격 객체간의 메소드 호출과 이를 지원하기 위한 객체 네이밍, 트랜잭션, 고장 감내, 트레이딩, 라이프 사이클과 같이 다양한 서비스를 지원함으로써 분산 객체간의 최적화된 호출을 지원한다. 또한, 비동기식 통신을 추가적으로 지원하며 이벤트 서비스와 공지 서비스를 지원함으로써 다양한 분산 모델을 수용하기에 용이하다.

대표적인 객체지향 미들웨어로는 CORBA 표준을 따르는 VisiBroker, ORBacus, TAO등이 있으며, SUN사의 Java RMI/JINI, 그리고 Microsoft사의 COM/DCOM등이 있다.

객체 지향 미들웨어에서 비동기 데이터 전송 및 객체의 데이터 및 상태 변화를 다른 객체에게 효율적으로 통보하기 위한 이벤트 서비스를 지원한다.

이벤트 서비스에서는 클라이언트와 서버를 공급자와 소비자로 부르는데, 공급자는 이벤트를 생성하며 소비자는 이벤트 서비스 서버를 통하여 이벤트를 받는다. 공급자와 소비자는 이벤트 채널에 연결되어 데이터를 전송받기 때문에 공급자가 소비자의 데이터(IP 주소, 수신포트)를 알아야 필요가 없다. 이벤트 채널은 이벤트 서버의 중재자와 같은 역할을 하여 공급자와 소비자의 등록과 해지, 데이터 전송 및 에러처리와 같은 역할을 수행한다.

이벤트 서비스는 데이터 전송 모델에 따라 Push 모델과 Pull 모델과 같이 두 가지 모델을 지원하며 그림 3, 4와 같다.

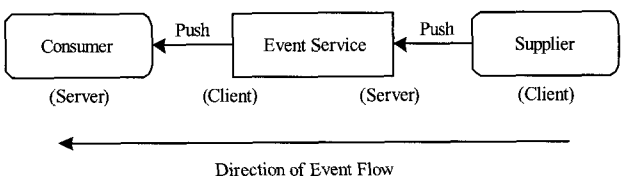


그림 3 Push 모델

Push 모델은 공급자에 이벤트가 생성되면 소비자의 받는 함수 Push()를 호출함으로써 이벤트를 전달하는 모델이다. 즉, 이벤트가 생성되는 대로 바로 소비자에게 보내지는 모델이다.

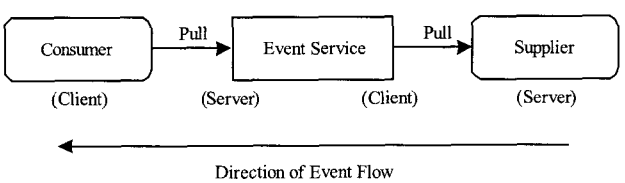


그림 4 Pull 모델

반면 Pull 모델은 공급자에 이벤트가 생성되면 공급자는 이벤트를 큐에 저장해 놓고 직접 보내지는 않고 소비자가 수시로 공급자의 보내는 함수 Pull()을 호출하여 큐를 체크함으로써 새로 생성된 이벤트가 있는지를

확인하고 이벤트를 가져오는 모델이다[3].

Push, Pull 모델은총 4가지의 경우 Push-Push, Push-Pull, Pull-Push, Pull-Pull로 사용할 수 있다.

#### 2.1.3 리플렉티브(Reflective) 미들웨어

기존 미들웨어 기술은 새로운 컴퓨팅 환경을 위한 분산 응용의 개발을 수월하게 하지만, 새로운 환경의 동적인 특성들을 처리하는데 필요한 지원을 하지 못한다. 미래의 응용은 다양한 환경 변화에 적응할 수 있고 PDA, 센서, 컴퓨터 등 다양한 장비에서 실행될 수 있는 미들웨어가 요구됨에 따라 리플렉티브 미들웨어가 제안되었다. 미래의 애플리케이션들을 지원하는 여러 가지 대안들이 있겠지만, 그 중 리플렉티브 미들웨어가 유연하고 적응적인 시스템 및 애플리케이션 개발과 다양한 동적인 환경을 다루는데 효과적인 대안이다.

리플렉티브 미들웨어는 애플리케이션의 시작 시간에 요구되는 컴포넌트를 동적으로 구성한다. 리플렉티브 미들웨어의 인터페이스는 변하지 않으며 기존 미들웨어를 바탕으로 개발된 응용에 의해 이용될 수 있다. 게다가 시스템과 응용은 미들웨어의 내부 구조를 검사하기 위해 메타 인터페이스를 이용할 수 있고, 필요에 따라 환경 변화에 적응하기 위해 미들웨어의 내부 구조를 재구성할 수 있다. 리플렉티브 미들웨어는 재구성 방법으로 시스템이 상황에 맞는 최적화된 성능을 발휘할 수 있도록 네트워크 프로토콜, 보안 정책, 인코딩 알고리즘 등 다양한 메커니즘들을 선택할 수 있다. 이러한 특성들로 인해 모바일 장치에서 동작하는 분산 멀티미디어 환경에서 멀티미디어 데이터를 처리하기 위해 서비스들에 따라 네트워크 프로토콜을 변경하거나 서비스의 질을 조정하는 것과 같이 동적 환경에서의 멀티미디어 응용, 그룹 통신과 실시간 내장형 플랫폼과 모바일 컴퓨팅 환경에서 미들웨어를 사용할 때의 한계점을 줄일 수 있다.

일반적인 관점에서 리플렉티브 미들웨어는 미들웨어 시스템의 검사와 적응을 지원하기 위한 연관된 자기 표현(self-representation)의 활용으로 언급된다. 여기서 자기 표현이란 관리되고 조작이 가능한 미들웨어 내부 구조의 표현이다. 표현의 변화는 미들웨어 내부 구조 자체의 변화에 적용되며, 미들웨어 내부 구조의 변화는 표현 변화에 적용된다[4].

전통적인 미들웨어는 특정 응용이 필요로 하는 모든 기능들을 포함하였다. 그러나 대부분의 응용은 실행 시간에단지 일부 기능만을 사용할 뿐이다. 리플렉티브 미들웨어는 단단한 구조의 블랙박스처럼 구성된 이러한 기존 미들웨어들과는 달리 상호 협업이 가능한 컴포넌트들의 그룹으로 구성된다. 이러한 특성으로 인해 리플렉티브 미들웨어는 기존의 미들웨어와 상호 작용이 가능하고 모바일 환경에 적합한 최소형 미들웨어 엔진 구성이 가능하다. 이러한 리플렉티브 미들웨어는 주로 대학기관에서 연구 중에 있으며, OpenORB(Lancaster Univ.)[5], 2K Project (UIUC)[6] 등이 있다.

### 2.2 디스커버리 미들웨어

유비쿼터스 환경에서 서비스 디스커버리 기술은 사용자가 원하는 서비스 및 자원을 보다 효율적으로 찾아서 이용하게 함으로써 사용자에게 최적의 편의성을 제공하는 기술로 홈 네트워킹 서비스 디스커버리와 그리고 유비쿼터스 환경의 서비스 디스커버리 미들웨어로 구분 지을 수 있다.

#### 2.2.1 홈 네트워킹 서비스 디스커버리 미들웨어

홈 환경에서 서비스 및 디바이스들을 효율적으로 관리하기 위한 서비스 디스커버리 미들웨어로는 Jini, UPnP, SLP, Salutation, HAVi, 그리고 BluetoothSDP 등이 있다. 이 미들웨어들은 자체적으로 고유의 프로토콜을 사용하여 효율적으로 동작하지만 미들웨어간의 상호운용성을 지원하지 못하고 이로 인하여 각각의 서비스 디스커버리 미들웨어들은 특정한 도메인에서만 동작하는 한계성을 지니게 되었다.

이와 같은 제약사항을 해결하기 위해 제안된 이기종 서비스 디스커버리 미들웨어 사이의 상호운용성을 지원하는 시스템은 1) 일대일 프로토콜 브리지, 2) 일대다 프로토콜 변환, 3) 일반적인 통합 미들웨어, 4) 서비스 레벨의 프록시와 같이 네가지로 분류할 수 있다.

일대일 프로토콜 브리지 방식으로 구분되는 시스템은 필립스사의 Jini-HAVi 브리지, 소니와 선, 선의 SLP-Jini 브리지[7] 등이 있다. 일대다 프로토콜 변환 방식[8]은 서비스를 요청하는 클라이언트 역할을 하는 미들웨어의 프로토콜을 일반적인 프로토콜로 변환하고 이를 다시 서버 역할을 하는 미들웨어의 프로토콜로 변환한다.

또 다른 방식으로 일반적인 통합 미들웨어 방식[9]을 들 수 있다. 이는 기존의 미들웨어들을 수용하는 것을 가능하게 하는 방식으로 이 방식의 가장 큰 특징은 새로운 미들웨어가 등장하였을 때 단순하게 프로토콜 변환을 위한 모듈을 추가함으로써 프레임워크(framework)안에 참여가 가능하다는 점이다.

마지막으로 프록시를 기반으로 하는 방식이 있는데 이 방식의 대표적인 예로 Jini/UPnP 프레임워크[7], [10]를 들 수 있다. 이 방식에서는 실제 서비스와 관련이 있는 가상의 서비스를 만들어 이를 통해 Jini나 UPnP 자체의 수정 없이 상호간의 서비스를 이용할 수 있다는 장점이 있다. 아래의 표1은 상호운용성을 지원하기 위해 제안된 시스템들을 각각의 특징에 따라 분류한

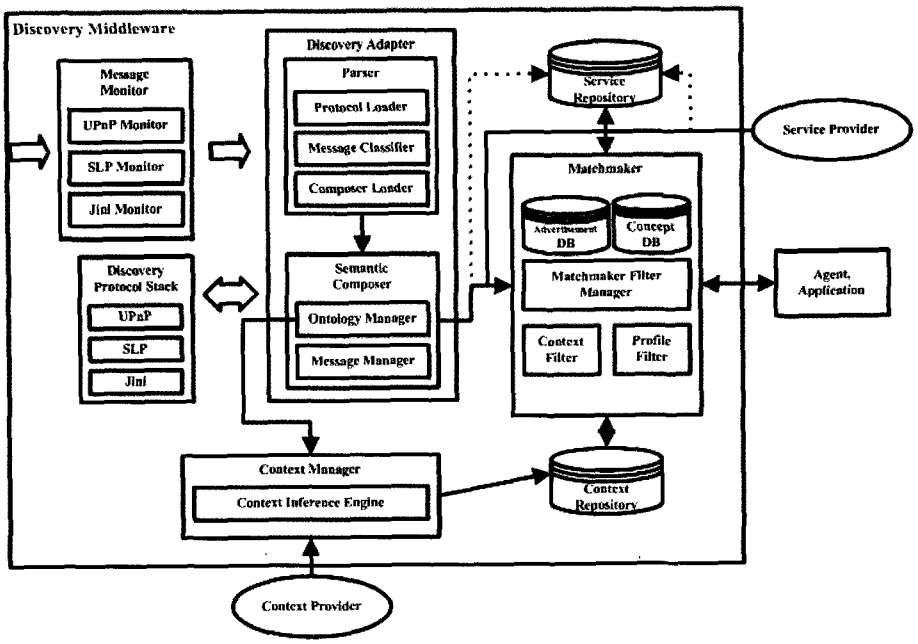


그림 5 유비쿼터스 환경의 서비스 디스커버리 미들웨어

표 1 상호운용성을 지원하는 방식 비교

| | | 프로토콜 대 프로토콜 변환 | 통합 미들웨어 | 프록시 기반 |
|---|---|---|---|---|
| 상호 운용성 지원 | | 낮음 | 중간 | 높음 |
| 기존의 미들웨어 변화에 대한 적응 | | 아니오 | 아니오 | 예 |
| 새로운 미들웨어에 대한 적응 | | 아니오 | 아니오 | 아니오 |
| 상호운용성에 따른 시스템 부하 | | 높음 | 높음 | 중간 |
| 개발 비용 | 초기 | 매우 높음 | 매우 높음 | 낮음 |
| | 새로운 서비스 타입 | 낮음 | 낮음 | 높음 |
| | 기존 미들웨어 변화 | 높음 | 높음 | 매우 낮음 |
| | 새로운 미들웨어 추가 | 매우 높음 | 매우 높음 | 높음 |

것이다.

이에 더하여 다양한 미들웨어를 묶어 정보가전기기들을 호환성과 홈 네트워크에 부가가치 서비스를 제공하기 위해서 OSGi(Open Service Gateway initiative)에 관한 연구가 활발히 진행되고 있으며, 국내에서 제안된 미들웨어에는 HWW와 LG의 LnCP가 있다. 그러나 현재 대부분의 미들웨어는 특정 응용영역에서 사용되고 있으므로, 미들웨어를 이용하는 장치뿐 아니라 통신 프로토콜에도 독립성을 지원해야 하는 진정한 의미의 미들웨어로 인정받기가 어렵다. 또한, 각 미들웨어들은 독자적인 기술을 이용함으로써 서로간의 호환이 어렵기 때문에 미들웨어간의 표준화 경쟁이 계속 되고 있다. 따라서 각 사용 목적에 따라 해당 응용 영역에 알맞은 미들웨어를 선택하여야 한다.

### 2.2.2 유비쿼터스 환경의 서비스 디스커버리 미들웨어

유비쿼터스 환경의 서비스 디스커버리 미들웨어는 홈 네트워킹 디스커버리 미들웨어를 수용하며, 애드혹 네트워킹 디스커버리 알고리즘과 지능형 서비스를 지원하기 위한 에이전트 기반의 디스커버리를 수용하여 발전되었다. 유비쿼터스 서비스 디스커버리 미들웨어는 상호운용성을 지원하기 위한 모듈 뿐만 아니라 사용자나 서비스 제공자의 다양한 컨텍스트 정보를 이용하기 위한 모듈(상황인지)과 온톨로지로 기술된 데이터와 일반적인 서비스 저장소에 존재하는 데이터를 이용하여 사용자의 요구에 보다 더 적합한 결과를 도출하기 위한 추론 엔진을 지원하여야 한다.

그림 5의 Message Monitor 모듈은 에이전트 플랫폼에서 하나의 에이전트로 동작하며 이기종 디스커버리 미들웨어에서 전송되는 메시지를 수신하는 역할을 담당한다. 여기서 수신된 메시지들은 Discovery Adapter 모듈로 전송 되어 parser에서 적절한 값들로 분석되고 Semantic Composer에서 에이전트 플랫폼 내에서 사용되는 DADs (Df Agent Descriptions)로 변환된다. 이러한 정보는 Matchmaker 에이전트로 전송되어 서비스 저장소 역할을 하는 에이전트 플랫폼의 DF에 저장되게 된다. 이때 메시지 모니터 모듈에 의해 들어온 메시지는 이기종 디스커버리 미들웨어에서 들어온 메시지이므로 해당 미들웨어의 이름이 설명서 앞에 표시되어

저장 된다. Context Manager는 Context Provider가 제공하는 컨텍스트 정보를 적절한 추론과정을 거쳐 Context Repository에 저장하는 역할을 담당하며 Matchmaker 에이전트는 User 에이전트로부터 요청이 들어왔을 때 DF에 저장된 서비스와 Context Repository에 저장된 컨텍스트 정보를 사용하여 최적의 결과 값을 도출하는 역할을 한다.

### 2.2.3 서비스 디스커버리를 위한 온톨로지

온톨로지는 "공유하기 위한 개념들의 개념화를 형식적이고, 명백하게 설명해 놓은 명세서"로 정의할 수 있다. 여기서 개념화는 어떤 현상에 대해 관련 있는 개념들을 식별하여 그 현상을 추상화한 모델로 설명하는 것이며, 명백하게 표현한다는 것은 사용된 개념의 종류와 개념들이 추상화된 모델과 갖는 관계들, 그리고 그 개념들의 사용에 있어서 주어지는 제한점들을 명백하게 정의한다는 것을 의미한다.

현재의 디스커버리 메커니즘은 서비스 사용 요구자, 제공되는 서비스 그리고 주변 정보 등의 의미를 전혀 고려하지 않는 단순한 키워드 기반의 서비스 추출로써 사용자를 충족시키지 못하며, 키워드 사이의 상관관계를 고려하지 않기 때문에 의미적인 관계를 다루는 것이 불가능하다. 온톨로지를 기반으로 하는 서비스 디스커버리 메커니즘은 이러한 문제점들의 해결을 목적으로 하며 대표적인 구조는 그림 6과 같다.

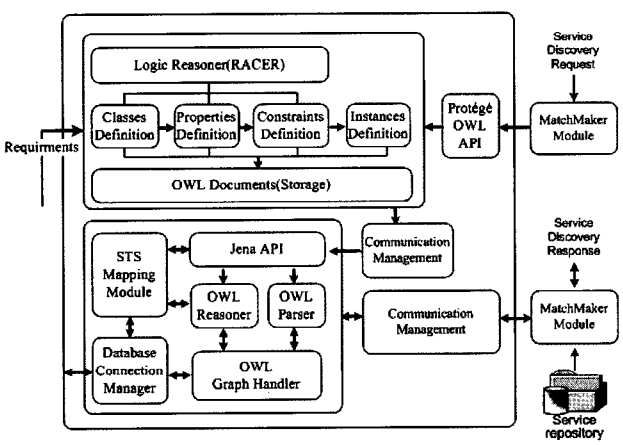


그림 6 서비스 디스커버리를 위한 온톨로지

온톨로지 기반 서비스 디스커버리에서는, 해당 메커니즘에 참여하는 모든 요소들, 심지어 디스커버리 프로세스까지도 의미 상호운용을 위한 공유가 가능한 온톨로지 사용에 동의한다. 이런 기본적인 전제를 통한 특정 도메인 기반 서비스 주체와 객체 그리고 주변 환경들에 대한 공통의 이해는 양질의 서비스 탐색 결과를 제공한다. 이러한 의미 상호운용은 요구사항 필터링에 의해 그 필요성을 설명할 수 있다. 디스커버리 프로세스의 가장 중요한 목적 중의 하나는 특정 작업을 수행하고자 하는 임의의 사용자에 의해 구체적으로 정의된 서비스의 인스턴스를 찾아서 해당 사용자와 연결시키는 작업이다.

특정한 목적에 맞는 온톨로지를 개발하는 것은 사용자의 목적을 정확하게 파악하는 것과 동시에 도메인 전문가, 온톨로지 개발자의 협력적인 작업이 필요하다. 미국과 유럽의 온톨로지 개발과 기술에 대한 연구는 이미 90년대 후반부터 본격화됐으며 현재 빠른 속도로 발전하고 있다. 국내에서는 한국전자통신연구원(ETRI)이 정보통신부의 지원을 받아 개발한 비주얼 온톨로지 편집기인 ezOWL이 만들어졌다. esOWL은 시맨틱 웹 콘텐츠 구축을 위한 도구로써 뿐만 아니라 온톨로지를 이용해 지식을 만들고자 하는 모든 분야에 활용될 수 있다.

## 2.3 에이전트 플랫폼

에이전트란 사용자의 관점에서는 시스템의 사용자를 대신해서 사용자가 원하는 작업을 자동적으로 수행해 주는 소프트웨어를 지칭한다. 즉, 이는 특정 목적에 대해 사용자를 대신하여 작업을 수행하는 자율적 프로세스이며, 독자적으로 존재하지 않고 어떤 환경의 일부이거나 그 안의 일부로서 동작하게 된다. 에이전트의 특징은 자율성, 지능성, 이동성, 사교성 등을 들 수 있으며, 이러한 기본적인 특성 외에도 에이전트가 가지는 성질은 환경 변화에 대하여 반응할 수 있는 반응성, 잘못된 정보를 주고받지 않도록 하는 정직성, 그리고 에이전트의 활동을 합리적인 방법으로 목적을 달성할 수 있도록 하는 합리성 등이 있다[11].

표 2 JADE와 Aglets의 비교 분석

| 분류 | JADE | Aglets |
|---|---|---|
| 개발기관 | Tilab | IBM, Tokyo |
| 목적 | 멀티 에이전트 개발 툴 제공 | 모바일 에이전트 모델 및 자바 API 제공 |
| 특징 | ▪ FIPA 기반의 에이전트 시스템 모델<br>▪ 에이전트간의 협업 지원 분산 프레임워크 제공<br>▪ AMS(Agent Management System)을 통한 효과적인 에이전트 관리<br>▪ 에이전트 Ontology 제공<br>▪ 에이전트 코드와 상태 이주 실행 지원<br>▪ XML과 RDF를 포함한 메시지 컨텐츠 생성 및 관리 지원 | ▪ 모바일 환경에 적합하도록 구성<br>▪ 사용자 클래스와 인터페이스를 지원하는 에이전트 개발 툴킷 제공<br>▪ Proxy 사용(자가복제 기능)<br>▪ 프록시를 사용한 보안 기능<br>▪ 자바 이벤트 위임처리 모델 기반<br>▪ 자바 객체를 호스트간에 이동할 수 있도록 구성 |
| 지원가능 프로토콜 | Java-RMI, JICP(Jade proprietary protocol), HTTP,IIOP, | Agent Transfer Technology (ATP) |

에이전트는 크게 멀티 에이전트와 모바일 에이전트로 구분된다. 멀티에이전트는 하나의 에이전트로 해결하지 못하는 복잡한 작업을 처리하기 위하여 다양한 작업을 할 수 있도록 구성된 여러 에이전트간의 협업을 통해 작업을 수행하도록 하는 구조이며, 모바일 에이전트는 사용자의 목적에 따라 작업을 수행하기 위해 프로그램 자체가 네트워크를 통해 돌아다니며 수행되는 구조이다. 이러한 특징으로 인해모바일 에이전트는 이동 컴퓨팅 기술과 같은 무선 네트워크 환경에서 널리 사용되고 있다.

이러한 에이전트의 특징을 효율적으로 지원하기 위해서 에이전트 플랫폼을 사용하는데, 대표적인 멀티 에이전트 플랫폼과 모바일 에이전트 플랫폼은 JADE[12]와 Aglets[13] 이 있으며 다음과 같은 특징을 가지고 있다.

멀티에이전트 표준인 FIPA에서 제안하고 있는 멀티 에이전트 플랫폼의 구조는 그림 7과 같다[14].

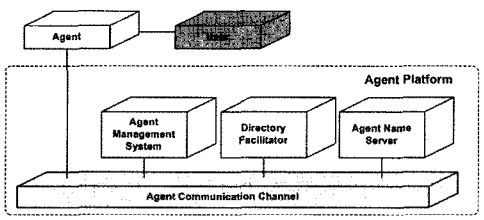


그림 7 멀티 에이전트 플랫폼 구조

멀티에이전트는 에이전트간의 통신을 위하여 ACL (Agent Communication Language) 또는 KQML을 사용하여 정보를 교환한다. ACC(Agent Communication Channel)는 에이전트 플랫폼 내부의 에이전트간의 메시지 전송과 플랫폼간의 통신을 지원한다. ANS(Agent Name Server)는 에이전트의 이름과 전송을 위한 주소 정보를 각 에이전트에게 제공한다. DF(Directory Facilitator)는 플랫폼 내의 에이전트들이 제공하는 능력이나 서비스에 대한 정보를 각 에이전트에게 제공한다. 즉, 서비스 디스커버리에 해당하는 역할을 수행한다. AMS(Agent Management System)는 플랫폼 내의 에이전트의 등록 및 제거, 일시 정지와 회복 등의 전반적인 에이전트 생명주기에 대한 관리를 수행한다. 플랫폼 영역 밖의 에이전트는 하나의 도메인의 기본적인 수행자를 나타내는데, 일종의 응용 에이전트의 개념이다.

분산 에이전트 시스템에서는 멀티 에이전트 시스템과 분산 에이전트 시스템을 다음과 같이 구분 지을 수 있다[15].

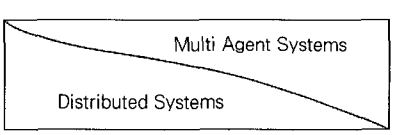


그림 8 멀티 에이전트 시스템과 분산 시스템의 관계

위의 그림과 같이 멀티 에이전트 시스템은 하부의 분산 처리 기능을 기반 (메시징, 부하분산 처리, 트랜잭션, 네이밍, 고장감내 등)으로 하여 에이전트의 특징을 갖는 프로세스를 지원하도록 한다. 즉, 멀티 에이전트 시스템은 특별한 형태의 분산 시스템으로 이야기 할 수 있다.

### 2.4 커뮤니티 컴퓨팅 미들웨어

PICO는 [16] 특정 목적을 중심으로 사용자 및 기기들을 대신해 작업을수행하는 자율 소프트웨어 개체들의 동적 커뮤니티를 구성하여 사용자에게 자율적이고 지속적인 서비스를 제공하는 것을 목표로 한다. PICO는 크게 하드웨어 기기와 센서를 나타내는 Camileuns과 그것 위에서 실행되는 소프트웨어 에이전트인 Delegents, 그리고 여러 Delegents의 집합체인 커뮤니티로 구성된다. PICO는 커뮤니티 오퍼레이션을 사용하여 Delegents, Camileun 자원간의 상관관계를 표현할 수 있도록 하였다.

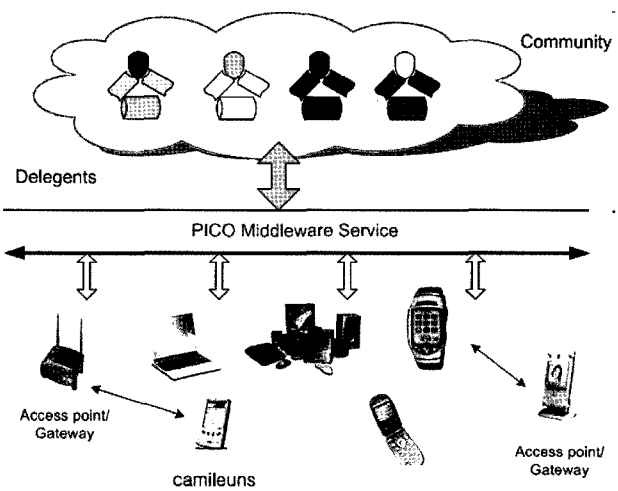


그림 9 PICO 미들웨어 구조

Delegents 간의 연관성을 통신, 협력, 협업, 커뮤니티와 같이 4가지 단계로 구분 짓고 이에 따른 특징을 정의함으로써 커뮤니티의 특징을 명확히 하고, 커뮤니티 구성원 간의 규칙을 정할 수 있도록 하였다.

표 3 PICO 에서의 상호작용 단계의 특징[16]

| | 통신 | 협력 | 협업 | 커뮤니티 |
|---|---|---|---|---|
| 상호작용 방법 | 데이터 | 데이터 | 데이터, 콘트롤 | 데이터, 콘트롤 |
| 상호작용 기간 | 짧다 | 길다 | 짧다 | 길다 |
| 상호작용 빈도 | 가끔 | 중간 | 중간 | 높다 |
| 의존성 | 없음 | 없음 | 약함 | 강하다 |
| 참여도 | 없음 | 없음 | 중간 | 높다 |
| 공유 작업 수 | 없음 | 없음 | 단독 | 다수 |

## 3. 에이전트 기반 지능형 미들웨어

유비쿼터스 환경에서는 사용자에게 적합한 서비스를 제공하기 위해 현재 상황에 대하여 상황뿐만이 아닌 환경에 대하여 스스로 인지하고, 그에 대처하기 위해 그룹 커뮤니티를 형성하고, 지식 기반 재구성을 통하여 디바이스에 관계없이 끊김없이 실시간으로 서비스를 제공해야 한다. 이러한 서비스들을 제공하기 위해 자체적으로 개발하고 있는 에이전트 기반의 지능형 미들웨어는 아래와 같다.

### 3.1 지능형 미들웨어 구조

그림 10의 통신 플랫폼 계층은 유무선 환경에서 효율적인 경량화 장치들과의 끊김없는 통신으로 상황, 위치 및 다양한 서비스들을 효율적으로 제공할 수 있도록 하며, 에이전트 플랫폼 계층은 다양한 에이전트 시스템의 장점들을 수용하여 스스로 환경에 적응하여 시스템 및 서비스의 효율성을 극대화 시킬 수 있도록 컴포넌트를 기반으로 구성되어 있다. 그림 10의 왼쪽 부분은 상황인지를 위한 것으로 지능화된 서비스를 위한 온톨로지 기반의 상황 인지 및 자가성장엔진으로 구성되어 있다. 마지막으로 에이전트 개발 툴 킷 및 상황에 대한 기반 데이터를 제공하는 컴포넌트를 작성할 때 효율적으로 상황을 표현하기 위한 상황지향 인터페이스 정의 언어(COIDL)를 지원한다.

### 3.2 통신 플랫폼 계층과 어댑터

커뮤니케이션 플랫폼은 ORB(Object Request Broker)를 기반으로 효율적인 상황인지를 지원하기 위한 이벤트 서비스와 컨텍스트 기반의 네이밍 서비스, 데이터의 일관성이나 무결성을 보장하고 오류시 회복할 수 있는 기법을 제공하는 트랜젝션 서비스, 다양한 네트워크 환경에 적응할 수 있는 동적 프로토콜 바인딩 서비스들을 제공하고, 그 외에 부하분산처리 및 보안, 고장감내 서비스 등을 제공한다. 이외에 플랫폼을 적용할 수 없는 임베디드 디바이스를 위해 커뮤니케이션 플랫폼에서는 이벤트 기반의 서비스를 제공한다. 이 서비스는 이벤트 서비스를 이용해 제작이 되었으며, 임베디드 디바이스는 어댑터를 사용하여 지능형 미들웨어의 에이전트를 활용할 수 있게 된다. 이벤트 서비스의 채널 생성과 관리는 에이전트 및 에이전트 그룹 에서 요구되는 상황 기반으로 동적으로 이루어지며, 전송되는 데이터의 상황에 맞게 프로토콜을 재결합 및 구성하여 데이터에 따라 최적화된 전송을 보장할 수 있도록 구성되어 있다.

### 3.3 에이전트 플랫폼 계층

에이전트 플랫폼은 상황정보 모니터링에 의해 수집된 상황을 필터링하고 조합하여 전달하는 상황인지 서비스

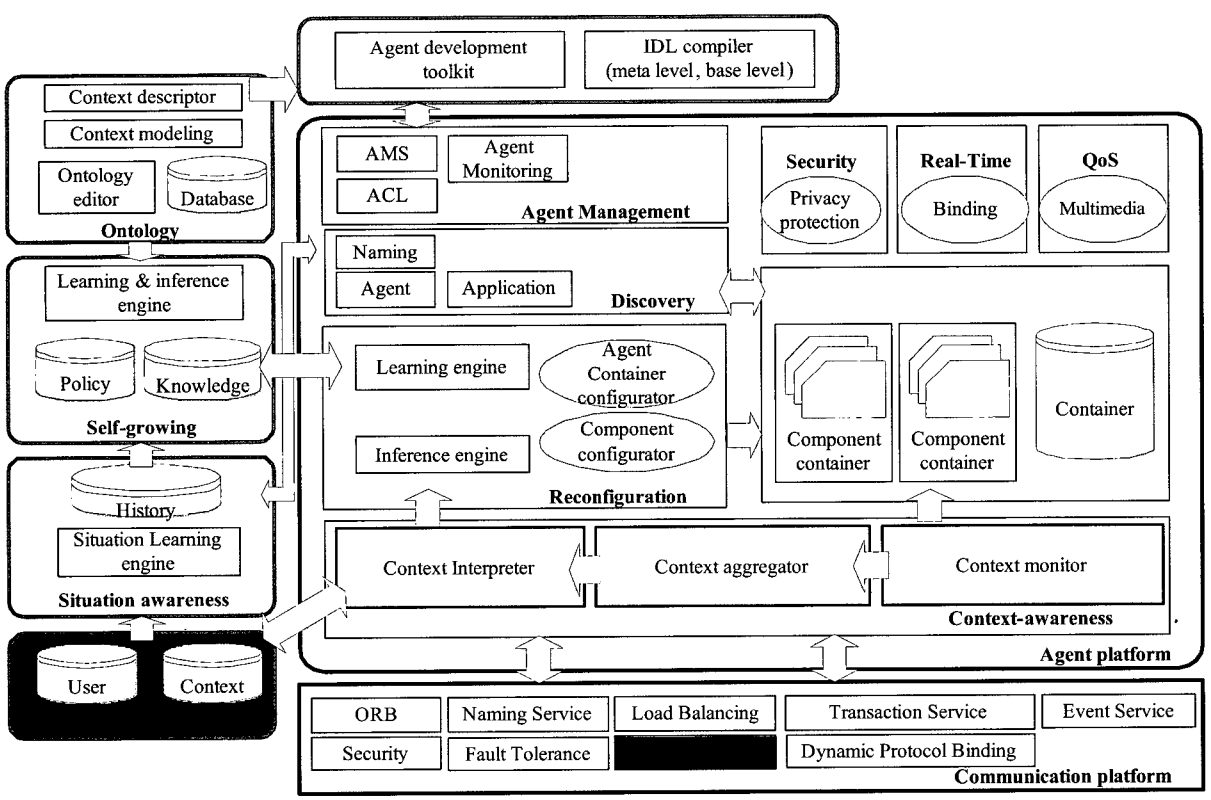


그림 10 지능형 미들웨어 구조

를 기반으로 하여 컴포넌트가 스스로 생성되고 관리, 소멸될 수 있도록 하는 컴포넌트 컨테이너 서비스를 제공한다. 또한 에이전트, 서비스 및 컴포넌트를 동적으로 검색해 스스로 재구성 하는 재구성모듈과 커뮤니티 그룹 에이전트를 자가 관리하기 위한 에이전트 관리 시스템을 제공한다. 그리고 실시간, QoS, 그리고 보안 서비스를 통하여 신뢰성 있고 최적화된 서비스를 제공한다.

에이전트 관리 시스템과 에이전트 디스커버리를 사용하여 에이전트 그룹을 동적으로 생성 및 삭제함으로써 커뮤니티 컴퓨팅을 지원하게 된다. 서비스를 요구하는 에이전트에 따라 연관된 그룹들을 동적으로 생성하고 그룹간의 연관된 상황정보 교환 및 처리를 통하여 그 상황에 최적화된 서비스를 제공한다. 이때 에이전트 및 상황의 중요성과 연관된 상황 및 장치, 서비스가 적용된다. 이와 같은 처리를 통하여 생성된 상황정보 는 사용자의 최종 확인을 통하여 적용되며, 사용자가 에이전트 결정에 대하여 만족을 느끼지 못하면, 서비스를 재조정함으로써 신뢰성을 높일 수 있다.

### 3.4 상황 인지 계층

환경으로부터 다양한 디바이스를 통해 들어온 상황정보를 수집하여 응용 또는 사용자에게 서비스나 정보를 제공해 준다. 이러한 다양한 응용 정보를 수집 및 통합하기 위하여 시스템 및 서비스의 요구사항을 수집하고 분석하여 제공한다. 이러한 상황 정보들을 여러 도메인 사이에서 공유 및 재사용을 하기 위해 상황 정보의 개념화, 구체화, 구조화 작업 및 추론 학습기능을 통하여 사용자 및 서비스에 제공하게 된다. 상황 모델링을 기반으로 한 온톨로지 및 환경정보 및 사용자 정보를 저장하는 상황 데이터베이스를 바탕으로 이력기반 환경 및 사용자 상황인지를 통하여 고수준의 정보 또는 구조화된 정보와 에이전트 그룹 상황인지를 통하여 에이전트 기반의 커뮤니티간의 효율적인 서비스 제공이 가능하도록 한다.

효율적인 플랫폼 재구성을 위해서는 기존의 정적-정책 기반 재구성이외에 특정 서비스를 위해 정책 자체가 변경 되어야 한다. 이와 같이 정책 이력을 활용한 학습 및 추론엔진을 통하여 정책을 조합하여 새로운 정책을 생성하여 적용하는 동적-정책 기반 재구성은 플랫폼 자체가 상황 및 서비스에 좀 더 능동적으로 대처할 수 있도록 한다.

### 3.5 에이전트 툴킷과 상황지향 인터페이스 정의 언어

에이전트 개발자는 에이전트가 동작하는 환경에 대한 지식을 가지고 있어야 하고, 에이전트 표준안과 에이전트를 개발하기 위한 지원 도구들에 대하여 많은 지식을 가지고 있어야 한다. 이러한 요구사항에 따라 에이전트 설계를 위한 디자인 패턴을 제공하고, 설계단계에서 에이전트들에 대한 성능 평가 및 코딩단계에서의 자동화를 위한 에이전트 툴킷을 제공한다.

미들웨어에서 사용되는 컴포넌트를 개발하거나 특정 상황정보를 제공하기 위한 컴포넌트를 제작하기 위해서는 기존에 플랫폼에서 제공받을 수 있는 상황의 종류에 대하여 알아야 하며, 개발자는 API의 이름을 통하여 상황의 종류 및 특성을 유추하게 된다. 기존의 프로그래밍에서는 개발자들이 헝가리안 표기법을 사용하여 인터페이스 및 함수를 정의하고 사용했다. 그러나 컨텍스를 효율적으로 정의하고 사용하기 위해 헝가리안 표기법과 CORBA IDL을 사용하여 접두사, 접미사에 대해 정의함으로써 컴포넌트의 역할에 대하여 명확하게 정의하고 사용할 수 있도록 하는 COIDL을 제공한다.

## 4. 결 론

지금까지 커뮤니티 컴퓨팅의 기반이 되는 미들웨어와 관련된 다양한 기술들에 대하여 알아보았다. 유비쿼터스 환경에서 최적화된 서비스를 제공하기 위한 커뮤니티 컴퓨팅과 관련된 요소기술로 에이전트 기반의 실시간 분산처리 기술 및 상황인지, 플랫폼 재구성 및 온톨로지 기반의 서비스 디스커버리와 같은 기술들이 지속적으로 제시되고 있다. 이와 더불어 동적으로 커뮤니티를 구성하고 관리하기 위한 에이전트 그룹의 동적 조직 구성기법 및 그룹간의 연관관계 표현 및 관리 기법과 그룹 기반의 재구성기법들이 연구 개발 되어야 할 것이다.

이러한 기반 기술들이 발전하게 된다면 동적 커뮤니티 그룹을 통한 커뮤니티 컴퓨팅을 기반으로 인간 친화적 유비쿼터스 컴퓨팅 환경은 머지않아 실현 가능해 질 것이다.

## 참고문헌


[1] Daniel A. Menasce. "MOM vc. RPC: Communication Models for Distributed Applications," pp. 90~95 IEEE internet computing, March · April (2005).

[2] Object Management Group, Common Object Request Broker Architecture: Core Specification, March (2004).

[3] Object Management Group, Event Service Specification, October (2004).

[4] Geoff Coulson, "What is Reflective Middleware?" http://dsonline.computer.org/middleware.

[5] Blair, G. S., G. Coulson, et al. "The Design and Implementation of Open ORB version

2," IEEE Distributed Systems Online Journal 2(6), 2001.
[6] Fabio Kon, Fábio Costa, Roy Campbell, and Gordon Blair. "The Case for Reflective Middleware," Communications of the ACM. Vol. 45, No. 6, pp. 33-38. June, 2002.
[7] Erik Guttman, James Kempf. "Automatic Discovery of Thin Servers: SLP, Jini and the SLP-Jini Bridge," IECON, San Jose, (1999).
[8] Hiroo Ishikawa et al. "A Framework for Connecting Home Computing Middleware," Proc. of IWSAWC2002.
[9] Kyeong-Deok Moon, Younghee Lee, and Yong-Sung Son, Chae-Kyu Kim. "Universal Home Network Middleware Guaranteeing Seamless Interoperability among the Heterogeneous Home Network Middleware," IEEE Transactions on Consumer Electronics, Vol. 49, No. 3, pp.546-553, AUGUST (2003).
[10] J. Allard, V. Chinta, S. Gundala, G.G Richard. "Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability," 2003 Symposium on Applications and the Internet, Orlando, (2003).
[11] Michael Luck, Peter McBurney, Chris Preist. "Agent Technology : Enabling Next Generation Computing," AgentLink community (2003).
[12] JADE, Java Agent Development framework http://jade.cselt.it.
[13] IBM Japan Research Group "Aglets Workbench," web site: http://www.trl.ibm.com/aglets.
[14] Foundation for Intelligent Physical Agents, FIPA Agent Management Specification, SC0023J, 2002.
[15] http://dsonline.computer.org/agents.
[16] Byung Y. Sung, Behrooz Shirazi, and Mohan Kumar. "Community Computing Framework for Enhancing Internet Services," Eurasian Conference on Advances in Information and Communication Technology, Oct. 2002, Iran.

**한 승 욱**

2003. 2 강남대학교 전자계산학과(학사)
1996~2003 삼성 소프트웨어 멤버십 연구원
2003. 8~현재 성균관대학교 정보통신공학부(석사과정)
관심분야 : 시스템 소프트웨어, 분산 컴퓨팅, 미들웨어
E-mail : donny@devg.org

**송 성 근**

2000. 2 성균관대학교 전기전자컴퓨터공학부(학사)
2004. 8 성균관대학교 전기전자컴퓨터공학부(석사)
2005. 3~현재 성균관대학교교 정보통신공학부(박사과정)
관심분야 : 유비쿼터스 컴퓨팅, 시스템 보안, 분산 컴퓨팅
E-mail : kkskk103@skku.edu

**윤 희 용**

1977. 2 서울대학교 전기공학과(학사)
1979. 2 서울대학교 전기공학과(석사)
1988. 8 미국 Univ. of Massachusetts at Amherst 컴퓨터공학과(박사)
1988~1991 Univ. of North Texas, 조교수
1991~1999 Univ. of Texas at Arlington, 부교수
1999~2000 정보통신대학교 교수
2000~현재 성균관대학교 정보통신공학부 교수
관심분야 : 모바일 컴퓨팅, 시스템 소프트웨어, 분산 컴퓨팅
E-mail : youn@ece.skku.ac.kr