유비쿼터스 컴퓨팅을 위한 통합 소프트웨어 구조

한양대학교 김태형** · 한국산업기술대학교 전광일**
단국대학교 최종무* · 서울대학교 홍성수**

1. 서 론

유비쿼터스 컴퓨팅은 1980년대 후반 미국에서 창안된 이후에 미래지향적 기술로 인식되어 지속적으로 연구되어 온 분야이다. 근자에 들어 많은 연구자들은 유비쿼터스 컴퓨팅이 웹과 초고속 인터넷처럼 산업과 사회전반에 막대한 영향을 미치게 될 것이라고 예측하고 있다. 이에 따라 국내외에서 유비쿼터스 컴퓨팅이 폭발적인 주목을 받고 있으며 현재 활발히 연구가 진행되고 있다. 그러나 아직은 그 연구결과가 초기 단계이고, 많은 면에서 다소 모호한 개념을 포괄하고 있어 유비쿼터스 컴퓨팅의 요소 기술들을 구체적으로 이해하는 것이 중요하다.

본 고에서는 유비쿼터스 컴퓨팅 시스템을 극도로 분산화된 컴퓨팅 플랫폼으로 인식하였을 때, 이를 지원하는 통합 소프트웨어 기반구조(integrated software infrastructure) 기술에 대해서 고찰하고자 한다. 유비쿼터스 컴퓨팅을 지원하는 많은 기술들로는 센서 네트워크, 사용자 인터페이스 기술, 인식과 보안 기술 등이 있지만, 통합 소프트웨어 기반구조 역시 매우 중요한 기술이다. 이를 구체적으로 서술하기 위해서 먼저 간단히 유비쿼터스 컴퓨팅을 정의하고, 유비쿼터스 컴퓨팅과 유사한 다른 컴퓨팅 패러다임과의 차이를 살펴본다. 계속해서, 대표적인 세가지 통합 소프트웨어 기반구조를 소개한다. 각각의 소프트웨어 기반구조에 대해서는 컴퓨팅 모델, 요소 기술, 소프트웨어 구조, 적용 사례에 대해서 차례로 기술한다. 아울러 결론적으로 향후 유비쿼터스 컴퓨팅을 위한 통합 소프트웨어 구조가 어떠한 방향으로 발전하게 될지에 대해서 전망한다.

* 정 회 원
** 종신회원

1.1 유비쿼터스 컴퓨팅의 정의

유비쿼터스(ubiquitous)는 '어느 곳에나 존재한다'라는 의미를 가지고 있는 라틴어 단어이다. 유비쿼터스 컴퓨팅이라는 표현에서는 장소나 시간에 구애 받지 않고, 생활 속에서 자연스럽고 편리하게 컴퓨터를 사용할 수 있는 환경을 의미한다. 즉 컴퓨터가 도처에 편재하여 센싱과 트래킹을 통해 장소나 시간에 따라 그 내용이 변화하는(context aware) 특화된 정보서비스를 받을 수 있음을 의미하는 것이다.

유비쿼터스 컴퓨팅은 1988년에 미국 제록스사 Palo Alto 연구소의 Mark Weiser에 의해 처음으로 제창되었다. 컴퓨팅의 진화 과정에서 보면, 이는 메인 프레임 기반 컴퓨팅, PC 기반 컴퓨팅에 이어서 제3세대 컴퓨팅 환경에 해당한다. 지금까지의 컴퓨팅에서는 온라인(가상) 공간 개념이 주축을 이루어 실재를 온라인 공간에 옮기는 것이 주된 목적이었으나, 유비쿼터스 컴퓨팅에서는 반대로 모든 실재에 컴퓨팅을 심는 것을 주목적으로 한다. 이를 위해서는 사물과 환경에 극소형의 컴퓨터를 심어서(embedding) 이들을 지능화시켜야 한다. 그 결과로 사물의 일부가 된 컴퓨터들은 주변 공간의 문맥(context)을 인식할 수 있고, 지리적으로 떨어진 곳에서도 사람들이 대상 사물과 그 주변 환경의 변화를 지각하거나 추적할 수 있도록 해준다. Mark Weiser는 다음과 같은 4가지의 판단기준에 따라 유비쿼터스 컴퓨팅을 정의하였다[1].

(1) 네트워크에 연결되지 않으면 유비쿼터스 컴퓨팅이 아니다.
(2) 사용자 인터페이스는 calm technology로 구성되어 눈에 띄지 않아야 한다.
(3) 현실 세계의 어디서나 컴퓨터의 사용이 가능해야 한다.

(4) 사용자 상황(장소, ID, 장치, 시간, 온도, 명암, 날씨 등)에 따라 서비스가 변해야 한다.

한편 유비쿼터스 컴퓨팅의 의미를 보다 포괄적으로 해석하는 경향도 있다. 대표적인 예가 노무라 연구소의 유비쿼터스 네트워크인데, 이는 유무선 네트워크를 통해 인터넷에 접근할 수 있는 오늘날의 정보가전 기반의 컴퓨팅 환경을 의미한다. 이런 관점에서 본다면 유비쿼터스 컴퓨팅은 비약적 기술의 진보라기 보다는 점진적 기술의 발전으로 인식될 수 있다.

1.2 유사 기술과의 차이점

유비쿼터스 컴퓨팅과 유사한 개념으로 종종 사용되는 컴퓨팅 패러다임들이 있다. 첫째가 nomadic 컴퓨팅이다. Nomadic 컴퓨팅이란 사용자가 장소나 기기에 구애 받지 않고 자신만의 정보 환경을 구축할 수 있으며, 이를 사용하며 일관된 방식으로 정보를 제공받을 수 있는 환경을 의미한다. 둘째가 증강 현실(augmented reality)이다. 이것은 다양한 장소와 기기에 센서가 부착되어 있어, 센서가 장착된 대상의 움직임을 파악할 수 있고, 그 대상에게 서비스를 제공해 줄 수 있는 환경을 의미한다. 대표적인 예로 GPS를 통한 교통 정보 시스템을 들 수 있다. AT&T에서는 이를 "sentient computing"이라는 이름으로 부르고 있다. 셋째로 착용 컴퓨팅(wearable computing)이다. 이것은 우리의 의복처럼 컴퓨터를 착용하고 다니는 환경을 말한다. 넷째가 pervasive 컴퓨팅이다. 순수하게 도처에 산재하는 컴퓨팅의 요건을 만족시키면 유비쿼터스 컴퓨팅이라 할 수 있으나, 이에 더 나아가 사용자가 전혀 컴퓨팅을 인식하지 않아도 되는 상태에 도달하게 되면 이를 pervasive 컴퓨팅이라고 한다.

1.3 소프트웨어 기반구조의 유형들

기존의 컴퓨팅 패러다임의 관점에서 볼 때, 유비쿼터스 컴퓨팅은 완전히 새로운 기술로의 진보라기 보다는 발전선 상의 기술로 인식될 수 있다. 이런 측면에서 유비쿼터스 컴퓨팅 시스템을 본다면, 이는 간단히 분산 컴퓨팅 시스템으로 이해될 수 있다. 그러나 그 형태적인 측면에서 세 가지 중요한 특성을 갖는다. 이는 (1) 극도의 분산화, (2) 엄청난 수의 컴퓨팅 노드 존재, (3) 동적인 ad hoc 네트워크 이용이다.

극도의 분산화란 많은 정보가 생성되고, 저장되고, 처리되는 것이 지역적으로 발생한다는 것을 의미한다. 물론 유비쿼터스 컴퓨팅 시스템에도 중앙집중적 데이터베이스가 존재하지만, 문맥 인식 컴퓨팅(context-aware computing)이나 지역 기반 서비스(location based service)를 제공하기 위해서는 지역적 정보가 분산적으로 수집되고 축적되게 된다. 이를 지원할 수 있는 명명 기술(naming service), 위치 인식 기술 등과 함께 분산 데이터 베이스 기술, 수집된 정보를 가지고 지능적 의사 결정을 내릴 수 있는 학습(learning), 계획(planning) 등의 인공지능 기술도 필요하다.

또한 유비쿼터스 컴퓨팅 시스템 상에 엄청난 수의 센서 노드, 컴퓨팅 노드, 디바이스 노드가 존재하게 될 것이라는 점은 누구나 쉽게 예측할 수 있다. 이들을 효율적으로 관리하고 유지하는 기술도 필수적인 기술이다.

아울러 유비쿼터스 컴퓨팅은 동적으로 변화하는 무선 네트워크인 센서 네트워크를 필수요소로 사용하기 때문에 이에 대한 소프트웨어 구조적 대응이 필요하다. 따라서 유비쿼터스 컴퓨팅 시스템은 동적 라우팅, P-to-P(peer to peer) 컴퓨팅 등을 지원할 수 있는 구조를 갖추어야 한다.

본 고에서는 이런 요구사항을 가지고 있는 유비쿼터스 컴퓨팅 시스템을 위한 통합 소프트웨어 기반 구조에 대해서 살펴 본다. 이를 위해 먼저 웹과 분산 객체에 기반을 둔 소프트웨어 구조에 대해서 살펴보고, 계속해서 센서 네트워크를 지원하는 소프트웨어 구조에 대해서 살펴본다.

2. 유비쿼터스 통합 소프트웨어 구조

소프트웨어 구조에 대한 연구는 모듈 내부의 프로그래밍을 위한 소규모 프로그래밍(programming-in-the-small) 활동이 모듈들 간의 상호연결을 위한 대규모 프로그래밍(programming-in-the-large) 활동과 직접적 관련이 없다는 사실을 지적한 DeRemer와 Kron에 의해 처음 시작되었다[2]. 일반적으로 소프트웨어 구조 연구에서는 ADL(Architectural Description Language)이라는 새로운 형태의 언어를 사용하여 소프트웨어 시스템을 구성(configuration)하는데 편리한 방법을 제공하는 것을 목적으로 한다. 이러한 소프트웨어 구조에서는 분산된 객체들 간의 상호운용성(interoperability)을 보장해 주기 위한 소프트웨

어 버스와 같은 기반구조가 필수적으로 사용된다. 예를 들면, CORBA(Common Object Request Broker Architecture)의 ORB(Object Request Broker)[3, 4]가 이와 같은 소프트웨어 버스의 역할에 해당한다.

최근 유비쿼터스 컴퓨팅을 가능하게 하는 많은 하드웨어 기술상의 발전과 통신 환경의 진보는 이에 상응하는 소프트웨어 기반 구조의 발전을 요구하고 있다. 모듈 내부에 관한 프로그래밍과 소프트웨어 버스를 매개체로 한 모듈간의 상호연결성을 위한 대규모 프로그래밍 만으로는 이 상황을 정확하게 반영할 수 없다고 판단되기 때문이다. 즉, 통상적인 대규모 분산 프로그래밍에서는 본질적으로 동등한 관계에 있는 컴퓨터 상호간의 분산된 컴퓨팅을 그 목적으로 한다. 그러나 유비쿼터스 컴퓨팅 환경에서는 내장형 마이크로 프로세서 또는 콘트롤러가 내장된 초소형 기기들이 대량으로 컴퓨팅 환경에 편입되므로 일반적인 분산 프로그래밍 환경과는 차이가 있다. 따라서 통상적인 대규모 프로그래밍에서 진일보한 새로운 소프트웨어 기반구조가 필요하며 이것을 DeRemer와 Kron의 용어 선정방식으로 설명한다면 대규모-다수-프로그래밍(programming-in-the-large-and-many)의 새로운 프로그래밍 환경이 될 것이다.

본 절(節)에서는 유비쿼터스 컴퓨팅을 위한 소프트웨어 기반구조를 살펴보기 위해 현재 많은 연구기관에서 진행중인 관련 프로젝트에서 기본적으로 생각하고 있는 컴퓨팅 모델을 설명하고, 이러한 모델을 구현할 수 있는 최신 소프트웨어 구축 요소 기술을 알아본 후, 이들의 통합구조와 이러한 구조가 적용될 수 있는 컴퓨팅 시나리오를 설명한다.

2.1 컴퓨팅 모델

Kleinrock은 유비쿼터스 컴퓨팅의 환경은 장소, 이동여부, 보유하고 있는 컴퓨팅 기기, 통신기기 및 통신 대역폭의 다양성과 같은 물리적 요소로부터 독립적인 컴퓨팅이 가능한 모델이 되어야 한다고 하였으며 이것을 Nomadic computing [5] 이라고 명명하였다.

즉, 사람은 일하고 쉬면서 통신기기가 내장된 기본적인 휴대용 컴퓨팅 기기를 가지고 끊임없이 장소를 이동하게 되며 이때 만나게 되는 장소, 이동, 컴퓨팅, 통신 환경의 여러 변화에 따라 적절한 서비스를 제공해 줄 수 있는 컴퓨팅 모델이 필요하다는 것이다. 유비쿼터스 컴퓨팅 시스템은 이와 같은 상황에서 편리한 형태로 사용자의 현재 상황(context)에 가장 적절한 응용을 제공해 줄 수 있어야 한다.

유비쿼터스 컴퓨팅을 위한 소프트웨어 기반 구조를 제시하기 위해 여러가지 노력들이 시도되고 있지만 현 시점에서 가장 실용성 높은 것이 미국 Hewlett Packard사 Internet and Mobile System Lab.에서 진행중인 Cooltown 프로젝트[6, 7, 8, 9]이다. 이것은 위에서 언급한 Kleinrock의 nomadic computing 개념을 구현하기 위해 이미 현실에서 광범위하게 사용되고 있는 웹 기술에 초점을 맞춘 웹기반 유비쿼터스 컴퓨팅 기반구조라고 볼 수 있다. 유비쿼터스 컴퓨팅이 단순히 일회적인 기술적인 혁신의 형태가 아니라 진화의 과정임을 이해한다면 이것은 매우 자연스럽고 적합한 연구 방향이라고 판단된다. Cooltown 프로젝트에서 컴퓨팅 환경은 웹(정확하게는, 웹 표현 또는 web presence)으로 통합되는 형태로 제공된다. 또한 유비쿼터스 컴퓨팅이 궁극적으로 실세계와 가상세계와의 원활한 통합이라고 생각할 때 이러한 웹 표현은 실세계와 가상세계의 교량역할을 하는 중요한 매개체로 사용된다. 여기서는 이러한 모델의 전체 구성을 설명하고 이러한 구성의 기본적인 요소로 사용되는 세 가지 개체의 의미에 대해 살펴본다.

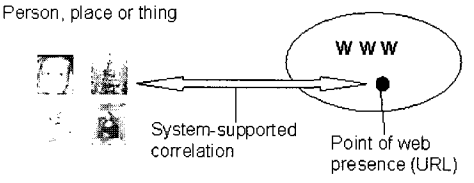


그림 1 웹으로 통합된 사람, 장소, 사물의 관계

2.1.1 웹기반 유비쿼터스 컴퓨팅 모델의 구성

Cooltown 프로젝트에서 컴퓨팅 모델은 그림 1과 같이 사람, 장소, 사물이라고 하는 중요한 세 가지 개체(entities)로 구성된다. 즉, 사람들은 고기능 컴퓨팅 기기를 휴대하고 무선 네트워크로 거의 언제나 연결되어 있다. 지난 10여 년 동안 끊임없이 급속히 팽창된 월드와이드 웹은 분산된 컴퓨팅 환경의 근간을 이루고 있다. 여기서 사람이라고 하는 개체가 장소로 이동하여 특정 사물에 관하여 동적으로 변화하는 정보를 제공하도록 함으로써 실세계와 가상세계가 통

합된 유비쿼터스 컴퓨팅을 실현하려는 것이 웹기반 유비쿼터스 컴퓨팅 모델이다.

2.1.2 첫 번째 개체 - 사람

"사람"이 유비쿼터스 컴퓨팅 모델의 중심이 되어야 할 것은 Mark Weiser가 그의 초기 논문[1]에서 이미 명확하게 지적한 바 있다. 즉, 사람이 컴퓨터가 있는 곳으로 가서 그 컴퓨터를 이용해서 어떠한 목적을 달성하는 것이 아니고 그 사람의 생활을 중심으로 여러 컴퓨터가 존재하여 사람이 크게 의식하지 않으면서 환경과 사람간의 상호작용을 통한 컴퓨팅이 이루어져야 한다는 것이다. 사람과 환경이 서로 상호작용을 하려면 사람에게 부착된 active badge와 같은 기기를 통해 자신의 존재를 자신이 처해 있는 환경에게 알려줄 수 있어야 한다. 또한, 사람이 (정확하게는 자신이 휴대하고 있는 컴퓨팅 기기가) 장소를 인지하기 위해 특정 지역마다 설치되어 있는 beacon에서 발신되는 장소에 관한 정보를 수신할 수 있는 taggy device를 갖게 된다. 사람이 이동함에 따라 컴퓨팅 context라고 생각할 수 있는 주변 환경(즉, 컴퓨팅 기기의 능력, 장소 등)은 지속적으로 변하게 된다. 변화하는 상황에서 해당 지역에 관한 특별한 정보는 이미 웹 표현의 형태로 존재하며 이것을 PDA와 같은 휴대용 기기의 브라우저를 통해 전달 받을 수 있도록 함으로써 문맥인식 컴퓨팅(context-aware computing)을 가능하게 한다.

2.1.3 두 번째 개체 - 장소

"장소"는 물리적인 경계를 갖고 있는 특정 지역을 의미한다. 따라서 강의실, 식당, 서점 등은 장소가 될 수 있지만 가상채팅 공간이나 온라인 서점 등은 이 모델에서 의미하는 장소가 될 수 없다. 이러한 물리적인 장소는 이미 존재 목적을 갖고 있으므로 이에 대한 정보는 이미 웹 표현으로 가상 공간에 존재하게 된다. 이 장소를 방문하는 사람은 이 장소에 대한 주소를 자신이 휴대한 taggy device를 통해 부여받게 됨으로써 관련 정보를 현장에서 얻을 수 있다. 즉, 어떤 사람이 특정 장소를 물리적으로 방문하였을 때 해당 장소에 관한 정보를 가상 공간에서 가져옴으로써 가상 공간과 실세계를 통합할 수 있다는 것이다. 물론, 이를 위해서는 모든 장소에 주소가 부여되어야 한다. 이름(naming)에 대한 일반적인 문제를 해결하기 위해 URI(Universal Resource Identifier)[10] 또는 URN(Universal Resource Name)[11]에 대한 연구가 있었지만 이와 같은 표준화가 실현되지 않았더라도 일단은 간단한 변환 작업을 통해 해당하는 문서의 URL을 얻을 수 있다.

2.1.4 세 번째 개체-사물

"사물"은 PDA, 라디오, 프린터와 같은 디지털 기기들 뿐 아니라 책, 그림과 같은 일반 물품도 포함한다. 이러한 사물들은 사람 또는 장소와 연관되어 존재한다. 즉, 사람은 특정한 사물을 휴대하거나, 단순히 소유하고 있거나, 사용하고 관리한다. 또한 어떤 특정한 장소에 속하여 그 장소의 목적에 맞는 서비스를 제공하는 역할을 할 수도 있다.

2.1.5 통합 모델

사실상 전통적인 웹 환경은 컴퓨터에 저장된 컴퓨터 중심의 환경이라고 볼 수 있다. 특정한 정보에 접근하려면 사람은 반드시 그 내용이 저장된 컴퓨터 상의 주소를 통해서만 가능하다. 따라서, 실세계와 가상세계 간에는 명확하게 분리되어 존재한다. 그런데 유비쿼터스 컴퓨팅에서는 사람이 상황을 인식하고 필요한 지시를 내릴 필요가 없는 환경을 구축하려고 하는 것이다. 컴퓨팅 환경을 구성하는 중요한 세 개의 개체는 위에서 살펴본 사람, 장소, 사물이며 이들이 또한 실세계를 구성하고 있다. 그런데 사람은 이동성이 있으므로 컴퓨팅 환경을 구성하는 개체들은 항상 변화한다. 이렇게 변화하지만 상황에 적합한 정보를 언제 어디서나 얻기를 원한다. 그림 1과 같은 컴퓨팅 모델은 실세계와 가상세계 간의 통합한 것이다. 즉, 사람, 장소, 사물은 실세계에서 존재하는 개체이지만 그 각각에 대한 가상 공간에서의 개체가 존재하며 이것을 해당 개체의 웹 표현(web presence)이라고 볼 수 있다. 이렇게 통합된 모델에서는 특정한 장소(예: 박물관)에 가서 그곳에 관한 정보(예:전시물 내역, 전시실 구조 등)를 얻거나 그곳에 위치한 특정 사물(예:소장품)에 관한 정보를 PDA와 같은 휴대용 기기를 통해 전달 받을 수 있는 서비스를 제공할 수 있다.

2.2 요소 기술

유비쿼터스 컴퓨팅 소프트웨어 구조를 위한 요소 기술은 크게 분산객체 기술, 웹 기술 및 미들웨어 기술로 분류될 수 있다. 웹 기반 유비쿼터스 컴퓨팅을

위한 소프트웨어 모델은 하나의 거대한 유비쿼터스 객체 시스템으로 볼 수 있다. 그러나 일반적인 분산 시스템과 유비쿼터스 컴퓨팅 시스템은 두 가지 기본적인 차이가 있다. 바로 이 차이가 기존의 기술을 단순히 확장하여 그대로 유비쿼터스 컴퓨팅에 적용할 수 없도록 만든다.

첫째로는 앞에서 지적한 바와 같이 유비쿼터스 컴퓨팅은 기존의 분산 시스템보다 훨씬 더 높은 극도의 분산화를 포함하고 있다는 사실에 따른 차이이다. 분산 시스템에서는 컴퓨팅 능력에 있어서는 차이가 있지만 근본적으로는 대등한 컴퓨터와 컴퓨터 간의 분산된 서비스 처리에 관한 기술을 다루고 있다. 하지만 유비쿼터스 컴퓨팅을 위해서는 컴퓨팅 능력과 메모리, 디스플레이, 통신능력 등의 자원 측면에서 근본적으로 차이가 있는 초소형 기기들이 다수 포함되어 있으므로 분산 시스템 기술을 단순 확장하여 사용하기 어렵다.

둘째로는 유비쿼터스 컴퓨팅의 편재성(ubiquity)과 분산 시스템의 분산(distribution)의 의미에 관한 본질적인 차이이다. 먼저, 편재성의 예로 전화교환 시스템을 생각해 볼 수 있다. 전화를 연결하는 것에 관한한 우리는 서비스 제공자와 전화 기간의 연결방식에 있어서 아무런 차이가 없다는 것을 잘 알고 있다. 반면에 텔레비전 방송 기술의 경우는 이와 다르다. 즉, 지역별로 NTSC, PAL, SECAM 등의 복수의 표준이 존재하고 있으므로 단말기기와 방송 시스템 간의 투명한 연결은 불가능하다. 방식이 서로 같지 않은 데도 함께 사용하기를 원하면 별도의 컨버터가 필요하다. 이것이 바로 분산 시스템에서 발생하는 상황이다. 이렇게 서로 같지 않은 이기종 간에 분산된 객체들이 원활하게 상호작용을 하기 위해 OMG의 CORBA ORB[3, 4], Microsoft의 DCOM[12], Sun Microsystems의 Java RMI[13] 등 여러 표준화된 분산객체용 미들웨어가 필요하며 이러한 방향으로 분산응용 시스템이 연구되어 왔다. 이것을 편재성의 수준까지 올리려면 이렇게 다양한 복수의 표준이 있는 상태에서도 상호운용성이 보장되어야 함으로 유비쿼터스 컴퓨팅을 위한 소프트웨어 기반구조로 이들 상호간의 또 다른 게이트웨이 서비스가 필요하다.

물론 편재성을 감안한 상호운용성을 위해 하나의 전세계적인 분산객체 미들웨어 구조를 표준화 하려는 노력을 할 수 있다. 실제로 프랑스의 INRIA에서 주관하는 ObjectWeb 컨소시엄이 1999년에 시작되어 지금도 여전히 WWW, Enterprise Java Bean Server, J2EE 및 CORBA-compliant 응용 서버에서 효율적으로 통합될 수 있는 오픈소스 분산객체용 미들웨어에 대한 연구가 활발히 진행되고 있는 상황이다[14]. 하지만 서로 다른 방향으로 발전되어 분산객체 미들웨어의 종류와 이와 관련된 기업들 간의 역학관계를 고려해 볼 때 새로운 또 하나의 표준이 만들고 이것으로 기존의 시스템을 대체하려는 실질적인 표준으로 만들 것을 기대하는 것은 위험한 시도가 될 것이다. 따라서 이러한 여러 미들웨어 기술이 공존하는 상황을 받아들이면서 유비쿼터스 컴퓨팅 환경을 구축하는 것이 보다 현실적인 접근이 될 수 있다. 오히려 유비쿼터스 컴퓨팅 환경은 이러한 다양성에 바탕을 두고 있으며 이러한 이유로 하나의 혁신적인 기술의 문제가 아니라 점근적으로 발전하는 evolving technology로 인식하는 것이 더 바람직하기 때문이다.

결과적으로 유비쿼터스 컴퓨팅을 위한 요소 기술은, HTML 또는 HTML 4.0 스펙에 따른 마크업 언어를 기본적인 프레젠테이션 도구로, HTTP와 같은 인터넷 전송 프로토콜을 배포(dissemination) 도구로, Java와 같은 플랫폼 독립적인 언어를 응용 서비스 프로그램의 이동성(mobility)을 보장할 수 도구로 사용하면서, CORBA ORB, Java RMI, Microsoft DCOM 등의 분산객체용 미들웨어를 기반구조로 인정하고 이들 상호간의 컨버터, 어댑터 또는 브릿지 역할을 담당하는 별도의 소프트웨어 계층으로 하는 다층 구조로 진화하는 형태를 갖게 될 것이다. 분산객체용 미들웨어는 기존의 서버용 또는 데스크탑용 PC와 같은 완벽한 컴퓨팅 환경을 갖춘 경우에 대한 구현뿐 아니라 디스플레이, 메모리, 연산능력, 전력 등 다양한 자원 제한적 상황에서도 작동할 수 있는 형태로 resource-constrained middleware 기술에 대한 연구가 많이 필요하다.

그 외에 정보의 전세계적인 편재성과 분산성으로 인한 보안의 문제가 중요한 요소 기술로 기반 소프트웨어에 포함되어야 하며 위치기반 및 문맥인식용 응용 프로그램 제작을 위한 각종 저작도구들이 공급되어야 한다. 현재와 같은 웹 환경을 그대로 사용하면서 전문적인 웹 프로그래머가 유비쿼터스 컴퓨팅 응용 프로그램을 제작하는 것은 대단히 복잡한 작업이 될 수 밖에 없기 때문이다. 사람과 장소와 사물이 하나로 통합되고 이에 따른 가상 공간에서의 웹 표현이 존재하며 응용 서비스를 주어진 환경에 맞게 동작할

수 있도록 작성하려면 이와 같은 기능들이 통합된 소프트웨어 기반구조가 유비쿼터스 컴퓨팅을 위해 제공되어야 할 것이다.

2.3 통합 소프트웨어 구조

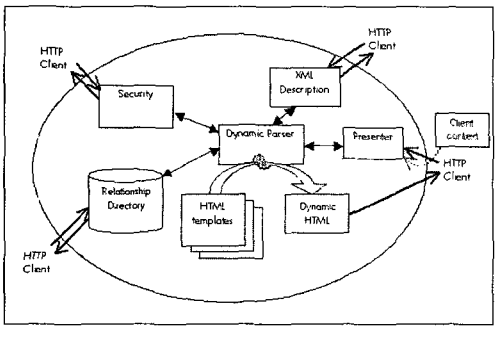


그림 2 웹 기반 유비쿼터스 컴퓨팅 통합 소프트웨어 구조

그림 2는 일반적인 분산객체 미들웨어와 이들 간의 게이트웨이 구조 위에 존재하는 실세계 개체들에 대한 웹 표현 처리를 위한 통합 소프트웨어 구조이다. 이것은 2.1절에서 언급한 컴퓨팅 모델을 구현한 하나의 형태이다. Dynamic Parser는 개체에 관한 웹 표현을 동적으로 HTML 또는 XML 문서로 생성하는 역할을 한다. 동적 웹 문서 생성 작업시 응용 서비스의 보안수준을 고려한다. Dynamic Parser의 동작 과정은 다음과 같다.

(1) 클라이언트 브라우저가 Presenter에게 HTTP 요청을 보낸다.

(2) Presenter는 요청에 해당하는 웹 문서에 대한 템플릿을 가져온다.

(3) Dynamic Parser가 이 템플릿을 처리하여 결과를 생성해낸다.

(4) 웹 문서가 클라이언트 브라우저에 보내진다.

Relationship Directory는 여러 개체들에 대한 웹 표현들 간의 관계를 기록하고 관리한다. 즉, 하나의 개체에 대한 URL을 사용자가 몰라도 각 개체가 갖고 있는 고유코드를 무선 랜, 블루투스, RF 등의 방식으로 가져온 후 해당 개체에 대한 실제 URL을 바인딩 해줌으로써 실세계 개체들이 가상 공간의 내용을 참조할 수 있도록 해준다. URL 키에 대한 참조와 효율적인 인덱싱을 위해 이 디렉토리 서비스는 관계형 데이터베이스의 형태를 갖게 된다.

어떤 개체가 다른 개체에 대한 가공되지 않은 형태의 웹 표현 정보를 알 필요가 있을 경우를 위해 XML Description이 존재한다. XML 표현에서는 필요한 기초 정보가 모두 포함되어 있으므로 클라이언트 디바이스의 특성에 맞는 특별한 태그를 갖는 HTML 또는 WML 문서를 생성할 수 있게 된다. 컨텐츠 적응성 (contents adaptation) 문제는 다양한 기기들이 존재하는 유비쿼터스 컴퓨팅 환경에서 필수적으로 필요한 기능이다.

이러한 기반구조에 여러가지 종류의 보안 체크 포인트가 존재할 수 있다. 웹 기반 유비쿼터스 컴퓨팅 환경에 있는 사용자들은 다양한 역할로 분류될 수 있기 때문이다. 관리자 역할의 사용자는 각종 HTML/XML 템플릿을 만들고 보안정책을 결정하며 개체정보 저장소를 관리한다. 일반 사용자라면 공개된 웹 표현을 가져와서 자신의 목적에 맞게 사용할 수 있다. 서로 다른 역할은 서로 다른 수준의 보안을 전제로 해야만 가능하다. 이를 위해 보안 컴포넌트가 통합 구조에 포함되어 있다.

2.4 적용 시나리오

위와 같이 사람, 장소, 사물이 웹 표현으로 통합된 모델에서 다음과 같은 유비쿼터스 컴퓨팅 시나리오를 적용해 볼 수 있다.

한 관람객("사람" 개체)이 박물관("장소" 개체)에 들어 온다. 이때 박물관에 관한 웹 표현과 관람객에 대한 웹 표현은 서로 동적으로 링크가 형성된다. 박물관 개체는 현재 관람객의 총수를 계산할 수 있고 방금 입장한 관람객은 전체 박물관에서 자신의 위치를 알아볼 수 있다. 관람객이 보유하고 있는 PDA 또는 핸드폰은 박물관의 beacon이 보내주는 코드를 이용하여 시스템의 Relationship Directory를 통해 박물관의 웹 표현에 해당하는 URL을 얻어올 수 있고, 이 사이트와 방문객의 PDA가 통신하여 자신의 기기 특성에 가장 잘 맞는 형태로 XML 형태의 웹 표현을 Dynamic Parser에 의해 자신의 브라우저가 표현하기에 적합한 웹 문서로 자동 변환시켜 PDA를 통해 보여준다. 이제 그 사람은 박물관 내에서 자신이 가장 관심을 갖고 있는 전시물 앞까지 이 정보를 이용해서 찾아간다. 그 전시물("사물" 개체) 자체는 디지털기기가 아니지만 그 주의에 해당 개체에 대한 고유코드를 방출하는 beacon이 있어서 이것을 수신한 관

람객의 PDA가 Relationship Directory의 도움을 다시 받아서 해당 전시물에 대한 정보가 포함된 웹 문서를 가져와서 전시물과 함께 감상한다.

3. 센서 네트워크를 위한 통합 소프트웨어 구조

3.1 컴퓨팅 모델

센서 네트워크를 위한 통합 소프트웨어는 센서기반 응용들을 개발하고, 유지하며, 센서 노드에 적용하고, 수행하는 기능을 효율적으로 지원하기 위해 필요하다. 이러한 통합 소프트웨어는 복잡한 상위 계층의 태스크로부터 서브 태스크들을 분리하고, 각 서브 태스크들을 적절한 센서 노드에 위치시키고, 각 센서 노드로부터 수집된 현상 정보를 취합하고 가공하여 관찰자(observer)가 전체적인 관점에서 현상을 분석할 수 있도록 지원할 수 있어야 한다[15].

센서 네트워크의 end-to-end 시스템의 구조는 그림 3과 같다. 시스템은 다수 개의 센서 노드로 구성된 센서 필드(또는 센서 네트워크), 수집된 현상 정보를 통합하여 서비스를 제공하는 서버, 그리고 센서 네트워크와 서버 사이에 인터넷을 통하여 데이터를 전송하는 게이트웨이로 구성된다[16]. 관찰자는 인터넷을 이용하여 서버에 접속하여 서비스를 받거나, 이동 단말기 등을 사용하여 직접 센서 노드의 정보를 이용할 수도 있다. 어떤 경우에는 센서 노드들은 서버에 연결되지 않고 센서 필드 내에서 자율적으로 정보 수집과 분석을 수행하며, 작동기(actuator)를 이용하여 현상을 제어하기도 한다.

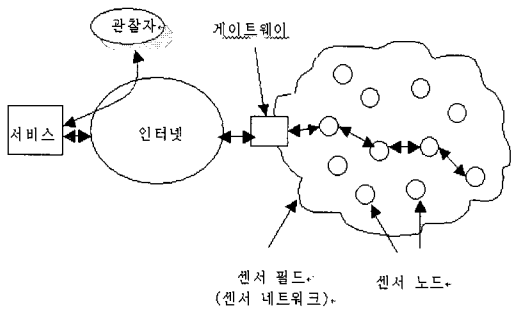


그림 3 센서 네트워크의 end-to-end 시스템 구조도

3.2 요소 기술

센서 네트워크는 전형적인 네트워크와는 다른 몇 가지 특성을 가지는데, 자가 재구성(self reconfigurable)이 가능해야 하고, 데이터 중심(data-centric)의 주소를 지원하며, 자원이 제약적이다는 것이다. 센서 네트워크를 위한 통합 소프트웨어는 이러한 특성을 지원할 수 있어야 한다.

3.2.1 자가 재구성을 위한 요소 기술

센서 네트워크는 전력의 고갈, 노드의 이동성, 노드의 결함, 그리고 환경적인 장애 등으로 인하여 형상이 동적으로 변하며, 심한 경우 네트워크가 분할되기도 하므로 자가 재구성(self reconfigurable)이 가능해야 한다. 센서 네트워크의 통신 프로토콜은 이러한 동적인 형상 변화가 있는 경우 ad hoc 네트워크를 구성하는데 저전력 요구조건과 낮은 대역폭, 오류가 많은 채널 특성, 자원의 제약 등 센서 노드의 능력을 고려하여 multi-hop ad hoc 네트워크를 구성한다[16].

센서 네트워크를 위한 통합 소프트웨어는 센서 응용들이 이러한 네트워크의 형상 변화에 관계없이 투명하게 자신들이 수행중인 태스크를 재구성하여 연속적으로 수행할 수 있도록 지원해야 한다. 이를 효과적으로 지원하기 위해서는 그룹 관리와 결함 허용 기술이 필수적이다. 그룹 관리는 목표로 하는 태스크를 성공적으로 수행하기 위해서 필요한 센서 응용들을 그룹으로 유지하며 주어진 태스크를 적절한 센서 노드에 전파하고 수집된 데이터를 통합하는 기능들을 쉽게 구현할 수 있는 프로그래밍 인터페이스를 제공해야 한다. 센서 네트워크는 전형적인 네트워크에 비하여 오류의 발생 가능성이 많은데 센서 응용에 투명하게 오류 처리 기능을 제공해야 한다. 또한 네트워크의 형상이 변한 경우에는 결함이 발생한 노드를 그룹에서 빼고, 새로운 가용한 노드를 그룹에 추가시켜 주어진 태스크가 연속해서 수행 가능하도록 지원할 수 있어야 한다.

3.2.2 데이터 중심의 주소 지원을 위한 요소 기술

센서 네트워크는 무수히 많은 센서 노드가 지역적으로 존재하고 응용의 특성으로 인해 노드 중심(node-centric)의 주소를 사용하는 것보다는 데이터

중심(data-centric) 주소를 사용한다. 즉, 인터넷과 같은 전역적인 주소보다는 attribute 기반의 주소를 사용하는데, attribute 기반의 주소 체계에서는 특정한 노드에게 쿼리를 보내기 보다는 현상의 attribute에 대한 쿼리를 이용하게 된다. 예를 들면 "센서 A에 의해서 감지된 온도"보다는 "온도가 50도가 넘는 지역은?" 또는 "A 지역의 평균 온도는?" 등과 같은 쿼리가 일반적이다[17]. 이러한 데이터 중심의 주소 체계를 지원하기 위해서 센서 네트워크를 위한 통합 소프트웨어는 새로운 형태의 명명(naming) 기술의 도입이 필요하다. 전형적인 명명 기법은 사용자가 지정한 이름을 전역적인 주소로 변환하였지만, 새로운 기법에서는 문맥(context), 속성(attribute), 위치(location) 등과 같은 여러가지의 컨텐츠 기반의 정보를 이용하여 쿼리를 처리할 수 있는 기법이 필요하다. 따라서 이를 위한 바인딩 매핑 룰의 설정, 필터링 기능의 설정, 데이터 수집(aggregation) 룰의 설정이 가능해야 한다.

3.2.3 센서 응용 관리를 위한 요소 기술

센서 노드에 탑재된 응용 소프트웨어들을 정적인 것이 아니라 환경의 변화나, 응용 분야의 확장, 그리고 현상을 탐지하기 위한 기능의 부가 등으로 동적으로 다운로드 및 수행이 가능해야 한다. 이러한 응용 프로그램의 변경은 간단한 파라미터의 재조정으로부터 전체 프로그램을 재 작성하는 것까지 매우 다양하다. 센서 네트워크를 위한 통합 소프트웨어는 이러한 프로그램의 재구성이 용이하고, 융통성이 있으며, 효율적으로 수행될 수 있도록 지원해야 한다. J2ME (Java 2 Micro Edition), KVM(Kilobyte Virtual Machine)과 같은 소형의 장치를 위한 가상 기계에 대한 연구가 활발히 이루어져 왔다. 하지만 이러한 가상 기계는 아직까지 센서 노드에서는 제공하지 못하는 많은 양의 메모리를 요구하고 있으며 저전력에 대한 고려도 미약한 편이다. 버클리 대학에서는 이러한 점을 고려하여 Tiny OS상에서 구동되는 소형의 바이트 코드 인터프리터인 Maté 가상 기계를 개발하였다[18]. Maté는 전형적인 가상 기계의 문제점을 고려하여 디자인 되었으며, 적은 양의 메모리와 저전력을 사용하는 구조로 설계되어 있다. 하지만 Maté의 경우 코드를 단일 바이트로 구성된 24개의 명령문으로 구성된 캡슐(capsule) 단위로 크기를 제한하고 어셈블리어 형태로 코딩을 해야 하는 등 전형적인 가상 기계와는 달리 개발자에게 프로그램 환경을 제약하는 단점을 가지고 있다. 센서 노드의 기술 발전이 이루어 지면 J2ME나 KVM과 같은 전형적인 가상 기계를 센서 노드에 적용하는 것이 확장 되리라고 전망된다.

3.3 통합 소프트웨어 구조

이와 같은 센서 네트워크를 위한 요소 기술을 이용한 통합 소프트웨어의 구조를 그림 4에 나타내었다.

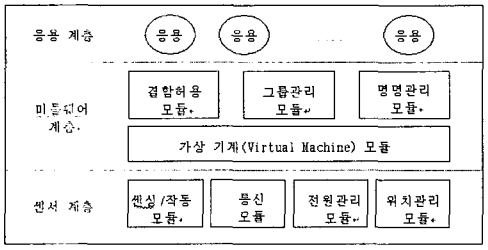


그림 4 센서 노드를 위한 통합 소프트웨어 구조

통합 소프트웨어는 미들웨어 계층에 위치하게 되는데, 하부의 센서 계층에서 제공하는 서비스를 이용하여 응용 계층의 응용 태스크들에게 투명성 있는 서비스를 제공한다. 미들웨어 계층의 결함 허용 기술, 그룹 관리 기술, 명명 관리 기술과 가상 기계 서비스를 이용하여 응용 태스크들은 네트워크의 형상에 투명한 데이터 중심의 서비스를 쉽게 제공받을 수 있다.

센서 네트워크를 위한 미들웨어는 전형적인 미들웨어와 상이한 특성을 지원한다. 따라서 센서 노드의 미들웨어와 인터넷에 연결된 서버의 미들웨어 사이의 응용간 협력이 필요한 경우에는 게이트웨이에 두 미들웨어 간에 상호 연동성을 지원하는 기능을 구현해야 한다.

4. 통합 소프트웨어 기반구조의 진화 방향

통합 소프트웨어 기반구조 중에서 핵심적인 역할을 하는 미들웨어에 관한 연구는 현재까지 3가지 형태로 각각 발전하여 왔다. 첫째는 메시지 중심의 미들웨어(MOM: Message-Oriented Middleware) 시스템이다. 이것은 클라이언트와 서버 간에 메시지 전달 방식으로 분산서비스가 제공될 수 있도록 하는 방식이다. 둘째는 프로시저 중심의 미들웨어(Procedural

Middleware)로서 80년대 초반 Sun Microsystems의 Sun RPC 또는 X/Open 컨소시엄의 DCE(Distributed Computing Environment)가 대표적이다. RPC는 현재에도 MS 윈도우스를 비롯한 대부분의 운영체제에서 지원되고 있다. 세번째는 객체 및 컴포넌트 기반 미들웨어 시스템이다. 앞서 언급한 OMG의 CORBA, Microsoft의 DCOM, Sun Microsystems의 Java RMI 및 Java Beans 등이 여기에 해당한다. 이러한 미들웨어 시스템들은 앞에서 살펴본 유비쿼터스 컴퓨팅을 위한 통합 소프트웨어 기반구조에 핵심적인 역할을 원활하게 하기 위해서는 다음과 같은 형태로 보완되어야 한다.

(1) Flexibility

위 세 가지 형태의 미들웨어는 모두 상호작용을 위해 서비스를 수행할 서버를 유일하게 식별할 수 있도록 하는 별도의 naming service를 필요로 한다. 이것은 전혀 새로운 환경으로 이동하였을 때 그 환경의 서비스를 유연하게 사용할 수 없도록 하므로 유비쿼터스 컴퓨팅에 적합하지 않다.

(2) Scalability

유비쿼터스 컴퓨팅에 참여하는 컴퓨팅 노드는 현재의 분산 컴퓨팅 보다 훨씬 더 많다. 극도로 분산화된 경우에도 상호작용이 원활할 수 있도록 미들웨어가 scalable 해야 한다.

(3) Mobility

현재의 미들웨어 시스템은 무선 이동 통신 환경에 정상적인 성능을 발휘할 수 없다. 빈번한 끊김과 좁은 대역폭에서도 안정적으로 동작할 수 있는 미들웨어가 필요하다.

(4) Resource Consciousness

유비쿼터스 컴퓨팅에서는 자원의 제약이 심한 내장형 기기들이 많이 사용된다. 기기 특성에 따라 자원 제약성을 효율적으로 관리할 수 있는 미들웨어가 필요하다.

(5) Dynamic Reconfigurability

미들웨어에 대한 재구성은 대부분 관리자에 의해 수동으로 행하여지고 있다. 유비쿼터스 환경에서는 수많은 기기들이 자유롭게 시스템 내부로 접속하였다가 다시 나가는 일이 빈번하므로 이러한 경우 시스템은 동적으로 재구성할 수 있는 능력이 필요하다.

우리가 앞 절에서 살펴본 통합 소프트웨어 구조는 유비쿼터스 환경에 적합한 새로운 미들웨어에 대한 고려 없이 기존의 시스템 소프트웨어 구조를 이용하여 구축하는 방법을 설명하였다. 이렇게 구축된 환경에서 다양한 응용이 실용화 된다면 이제 위에서 언급한 새로운 환경에 맞는 방향으로의 기반 소프트웨어에 대한 진화가 이루질 수 있도록 연구가 집중되어야 할 것이다.

참고문헌

[1] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," Communications of the ACM, July, 1993, 75-84.

[2] F. DeRemer and H. Kron, "Programming-in-the-large versus Programming-in-the-small," IEEE Transactions on Software Engineering, Vol. 2(2), June, 1976.

[3] The Common Object Request Broker: Architecture and Specification, Version 3.0, Object Management Group, June, 2002.

[4] Object Management Group (OMG), http://www.omg.org.

[5] L. Kleinrock, "Nomadic computing: anytime, anywhere in a disconnected world," Mobile Networks and Applications, Vol 1(4), 1997.

[6] HP Cooltown project, http://www.cooltown.com

[7] T. Kindberg and J. Barton, "A Web-Based Nomadic Computing System," HP Technical Report, HPL-2000-110, 2000.

[8] W. Chan, "Using CoolBase to Build Ubiquitous Computing Applications," HP Technical Report, HPL-2001-215, 2001.

[9] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic, "People, Places, Things: Web Presence for the Real World," HP Technical Report, HPL-2001-279, 2001.

[10] Naming and Addressing: URIs, http://www.w3.org/Addressing/

[11] The URN Working Group, http://www.ietf.org/html.charters/urn-charter.html

[12] The Distributed Component Object Model (DCOM), Microsoft Corporation http://www.microsoft.com/com/tech/DCOM.asp

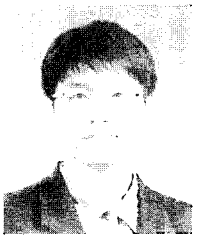
[13] Java Remote Method Invocation, http://java._sun.com/products/jdk/rmi
[14] The ObjectWeb Consortium, http://www.objectweb.com
[15] Kay Römer, Oliver Kasten, Friedmann Mattern, Middleware Challenges for Wireless Sensor Networks, http://www.inf.ethz.ch/vs/publ/papers/wsn-middleware.pdf
[16] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, A Survey on SensorNetworks, IEEE Communications Magazine, pp. 102-114, Aug. 2002.
[17] Chien-Chung Shen, Chavalit Srisathapornphat, and Chaiporn Jaikaeo, Sensor Information Networking Architecture and Applications, IEEE Personal Communications, pp. 52-59, Aug. 2001.
[18] Philip Levis, and David Culler, Mat: A Tiny Virtual Machine for Sensor Networks, In Proceedings of the Architectural Support for Programming Languages and Operating Systems, 2002.

김 태 형

1986. 2 서울대학교 컴퓨터공학과 졸업
1988. 2 서울대학교 컴퓨터공학과 석사
1996. 9 University of Maryland College Park, Dept. of Computer Science 박사
1988. 3 ~ 1990. 8 현대전자산업(주) 응용 SW 개발팀
1996. 10 ~ 1999. 3 현대정보기술(주) 정보기술연구소 책임연구원
1999. 3 ~ 2001. 2 한양대학교 전자컴퓨터공학부 전임강사
2001. 3 ~ 현재 한양대학교 전자컴퓨터공학부 조교수
관심분야 : 분산병렬컴퓨팅, 이동컴퓨팅, 내장형 시스템, 미들웨어 시스템, 소프트웨어 구조

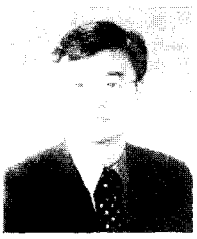
전 광 일

1986. 2 서강대학교 전자계산학과 학사
1988. 2 서울대학교 컴퓨터공학과 석사
2002. 2 서울대학교 컴퓨터공학과 박사
1988. 2 ~ 1994. 3 한국전자통신연구원 선임연구원
2001. 1 ~ 2003. 2 유비쿼스 주식회사 연구소장
2003. 3 ~ 현재 한국산업기술대학교 컴퓨터공학과 조교수
관심분야 : 운영체제, 분산시스템, 내장형시스템, 결함허용시스템



최 종 무

1988. 3 ~ 1993. 2 서울대학교 해양학과 이학사
1993. 3 ~ 1995. 2 서울대학교 컴퓨터공학과 공학석사
1995. 3 ~ 2001. 2 서울대학교 컴퓨터공학과 공학박사
2001. 3 ~ 2003. 2 (주) 유비쿼스 기술연구소 책임연구원
2003. 3 ~ 현재 단국대학교 정보컴퓨터학부 교수
관심분야 : 시스템 소프트웨어, 운영 체제, Embedded Linux, 내장형 시스템, 스토리지 시스템, 분산 시스템 등



홍 성 수

1986. 2 서울대학교 컴퓨터공학과 졸업
1988. 2 서울대학교 컴퓨터공학과 석사
1994. 12 University of Maryland at College Park, Dept. of Computer Science 박사
1988. 2 ~ 1989. 7 한국전자통신연구원 연구원
1994. 12 ~ 1995. 3 Faculty Research Associate(University of Maryland at College Park)
1995. 4 ~ 1995. 8 Silicon Graphics Inc. 연구원
1995. 9 ~ 1997. 9 서울대학교 전기컴퓨터공학부 전임강사
1997. 10 ~ 2001. 10 서울대학교 전기컴퓨터공학부 조교수
2001. 10 ~ 현재 서울대학교 전기컴퓨터공학부 부교수
관심분야 : 실시간 시스템, 실시간 운영체제, 정보가전, 실시간 시스템 설계 방법론, 멀티미디어 시스템, 소프트웨어 공학.