DOI: 10.3745/KIPSTA.2009.16-A.4.263

홈 네트워크 환경에서의 동적 확장성과 부하분산을 위한 다중 미들웨어 브리지

김 연 우[†] · 장 현 수[††] · 송 창 환[†††] · 엄 영 익[††††]

요 약

스마트 홈 인프라를 기반으로 유비쿼터스 컴퓨팅 환경을 구축하기 위해, 여러 연구기관과 산업체에서는 홈 네트워크에 대한 다양한 연구를 진행하고 있다. 이로 인해 다수의 홈 네트워크 미들웨어들이 제안되었으며, 이는 홈 네트워크 표준화를 지연시켰고, 이러한 표준화의 지연은 이기종 미들웨어 간 상호운용성 문제를 해결할 수 있는 미들웨어 브리지를 요구하였다. 현재 두 미들웨어에 대한 상호운용성을 지원하는 일대일 브리지, 다수의 미들웨어에 대한 상호운용성을 지원하는 일대다 브리지 등 여러 미들웨어 브리지에 대한 개발과 상호운용성 지원 기법에 대한 연구가 진행되고 있다. 하지만 기존의 시스템과 기법들은 특히 스마트 홈 환경에서 요구되는 동적 확장성과 성능에 대한 고려가 부족하다. 미들웨어 브리지는 전문적인 지식이 없는 사용자를 위해 브리지의 수정 없이 쉽게 새 미들웨어를 추가할 수 있는 브리지 확장성을 제공해야 한다. 또한 브리지는 메시지 집중을 해결할 수 있는 부하 분산 기법 역시 제공해야 한다. 본 논문에서는 홈 네트워크 환경에서의 동적 확장성과 부하분산을 위한 다중 미들웨어 브리지(Multi-Middleware Bridge, MMB)를 제안한다. MMB는 분산 구조를 통해 브리지 확장성과 부하 분산 기능을 제공한다. 또한 이에 대한 구현과 평가를 통해, 상호운용성 지원 기능과 브리지 확장성, 부하분산 알고리즘의 성능을 검증한다.

키워드 : 홈 네트워크, 미들웨어, 상호운용성, 미들웨어 브리지, UPnP, Jini

A Multi-Middleware Bridge for Dynamic Extensibility and Load Balancing in Home Network Environments

Younwoo Kim[†] · Hyunsu Jang[††] · Changhwan Song[†††] · Young Ik Eom[††††]

ABSTRACT

For implementing the ubiquitous computing environments with smart home infrastructures, various research on the home network have been performed by several research institutes and companies. Due to the various home network middleware that are developed recently, the standardization of the home network middleware is being delayed and it calls for the middleware bridge which solves the interoperability problem among the heterogeneous middlewares. Now the research on the scheme for interoperability and the development of the various bridges are in progress, such as one-to-one bridge supporting interoperability between two middlewares and one-to-many bridge supporting interoperability among the multi-middlewares. However, existing systems and schemes does not consider the dynamic extensibility and performance that is particularly needed in the smart home environments. The middleware bridge should provide bridge extensibility with zero-configuration for non-expert users. It should also provide the load balancing scheme for efficient and proper traffic distribution. In this paper, we propose a Multi-Middleware Bridge(MMB) for dynamic extensibility and load balancing in home network environments. MMB provides bridge scalability and load balancing through the distributed system structure. We also verify the features such as interoperability, bridge extensibility, and the performance of the load balancing algorithm.

Keywords : HomeNetwork, Middleware, Interoperablility, Middleware Bridge, UPnP, Jini

1. 서 론

각종 유무선 통신 기술과 전자 산업의 발달로 가전기기들의 기능과 성능이 발전하고 있다. 이러한 가전 기기의 발전은 스마트 홈 인프라를 기반으로 유비쿼터스 컴퓨팅 환경을 구축시켜주는 홈 네트워크에 대한 다양한 연구를 이끌어내었다. 홈 네트워크의 보급을 위해 전문적인 지식 없이도 가

※ 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2009-(C1090-0902-0046))
† 준 회 원 : 성균관대학교 전자전기컴퓨터공학과 석사
†† 준 회 원 : 성균관대학교 전자전기컴퓨터공학과 박사과정
††† 준 회 원 : 성균관대학교 전자전기컴퓨터공학과 석사과정
†††† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2008년 9월 12일
수 정 일 : 1차 2009년 3월 5일, 2차 2009년 4월 20일, 3차 2009년 6월 3일
심사완료 : 2009년 6월 3일

전기기 간 장치 인식과 기기 간 상호작용이 가능한 홈 네트워크 미들웨어(UPnP[1], HAVi[2], Jini[3], DLNA[4])들이 제안되었다. 여러 연구기관과 산업체에서 동시에 이루어지고 있는 홈 네트워크 미들웨어에 대한 연구는 홈 네트워크 미들웨어의 표준화를 지연시키고 있다. 홈 네트워크 미들웨어의 표준화가 지연됨에 따라, 서로 간의 호환을 생각하지 않은 이기종 홈 네트워크 미들웨어들이 늘어났다. 이러한 상호운영성 문제는 홈 네트워크 서비스의 개발을 어렵게 하였다.

홈 네트워크의 상호운용성 문제에 대한 해결책으로 미들웨어 브리지들이 제안되었다. 미들웨어 브리지는 프로토콜 변경과 미들웨어의 운영 기법 간 이질성 해소를 위해 이기종 홈 네트워크 미들웨어 간의 상호운용성을 지원한다. 이러한 브리지로 Jini meets UPnP[5]등의 두 미들웨어의 메시지 매핑을 중심으로 상호운용성을 지원하는 일대일 브리지와 UMB[6]등의 홈 네트워크 통합 계층을 향한 추상화 및 구체화를 통해 여러 미들웨어들의 상호운용성을 동시에 지원하는 일대다 브리지 등이 연구되었다.

확장성과 성능은 홈 네트워크 미들웨어 브리지에서 가장 중요한 요소이다. 이미 많은 홈 네트워크 미들웨어가 존재하며 더 추가될 상황이기 때문에, 홈 네트워크 미들웨어 브리지는 기존 시스템에 별다른 변화 없이도 새 홈 네트워크 미들웨어를 지원할 수 있는 브리지 확장성을 제공해야 한다. 이 과정에서 전문 지식이 없는 사용자에 대한 배려도 필요하다. 또한 미들웨어 브리지는 여러 미들웨어를 동시에 관리하고 모든 메시지를 처리하므로 브리지의 성능에 대한 고려 역시 반드시 필요하다.

본 논문에서는 분산 구조를 통해 확장성과 부하 분산 기능을 가지는 MMB(Multi-Middleware Bridge)를 제안한다. MMB는 네트워크에 분산된 여러 미들웨어의 연결을 통해 동적인 확장성을 제공하고, 우선순위 연산 기반의 부하분산을 사용해 성능을 향상시켰다. 본 논문은 MMB의 설계를 보이고 구현과 시뮬레이션을 통해 MMB의 기능과 부하분산 알고리즘의 성능을 검증한다.

본 논문의 구성은 다음과 같다. 제2장에서는 홈 네트워크 미들웨어와 미들웨어 간 상호운용성 지원을 위한 미들웨어 브리지에 대한 연구를 소개한다. 제3장에서는 본 논문에서 제안하는 MMB의 설계와 구현을 보인다. 제4장에서는 다른 브리지와의 비교 및 성능 평가를 통해 MMB를 검증하고, 마지막으로 제5장에서는 결론과 향후 연구 과제에 대해 말한다.

2. 관련 연구

2.1 홈 네트워크 미들웨어

홈 네트워크는 가정 내에서 사용하는 각종 장비들의 연결을 뜻한다. 홈 네트워크 미들웨어는 홈 네트워크를 쉽고 편리하게 구축할 수 있도록 도와준다. 이러한 홈 네트워크 미들웨어로 UPnP, HAVi, Jini등이 존재한다. UPnP는 마이크로소프트, Intel, compaq, 필립스, 소니 등 백여 개의 기업이 참여한 PC 중심의 가전기기 제어 소프트웨어 표준이다. UPnP는 TCP/IP, HTTP, GENA, SOAP, SSDP등 널리 인정받은 표준을 이용하고 있다. 그 예로 DLNA[4], LnCP등 새롭게 연구 되고 있는 새로운 홈 네트워크 미들웨어 및 브리지 기술 등은 UPnP를 중심으로 구성 되었다. HAVi는 AV(Audio Video)시장의 주요 업체인 소니, 톰슨, 필립스, 도시바, 샤프, 히타치 등 8개 기업들이 참여한 디지털 오디오/비디오 간 상호운용성 지원을 위한 단체이다. HAVi는 IEEE1394 인터페이스 및 프로토콜을 이용한 AV 스트리밍 정보의 전송에 특화되어 있다. Jini는 썬 마이크로시스템즈사를 중심으로 제안된 자바 기반의 홈 네트워크 자원 공유 플랫폼이다. Jini는 기존의 홈 네트워크 기술 중 자원 탐색에 가장 효과적이며 기술적으로 가장 우수하다. 하지만 JVM지원이 반드시 필요하다는 단점이 있다. LonWorks는 전력선을 활용한 저속 통신 기반의 홈 네트워크 미들웨어다. LonWorks의 응용 표준에 가전 기기 표준을 추가하기 위해 여러 작업이 진행 중이다.

이렇듯 많은 수의 홈 네트워크 미들웨어가 존재함에 따라, 홈 네트워크 장비 설계의 어려움과 장비 간 통신에서의 이질성 문제 등, 홈 네트워크 확산에 걸림돌이 되는 상호운용성 문제들이 발생하였다.

2.2 홈 네트워크 미들웨어 브리지

홈 네트워크 환경에서의 상호운용성 문제에 관한 두 가지 관점이 있다. 첫째는 장비 간 물리적 연결에 대한 관점이다. 하지만 이는 IPv6가 통신 네트워크의 표준이 되어가는 현재의 상황에서 비교적 중요성이 낮다. 둘째는 프로토콜 및 통신 기법의 차이에 대한 관점이다. 두 장비가 네트워크를 통해 연결되어 서로 통신을 할 수 있는 상황이라 하더라도, 만약 두 장비가 각기 다른 홈 네트워크 미들웨어를 사용하고 있으면 두 장비는 서로에 대한 상호작용을 할 수 없다. 두 홈 네트워크 미들웨어 간의 프로토콜과 통신 기법의 차이에 의해 장치 인식 및 서비스 호출이 불가능하기 때문이다.

이러한 홈 네트워크 미들웨어 간 차이를 극복하기 위해 홈 네트워크 미들웨어 브리지가 연구되고 있다. 홈 네트워크 미들웨어 브리지는 프로토콜 변환 및 보조적인 데이터 운영을 통해 이기종 홈 네트워크 미들웨어 간 통신을 가능케 하는 시스템이다. 홈 네트워크 미들웨어 브리지로, 두 홈 네트워크 미들웨어에 대한 분석을 통해 두 홈 네트워크 미들웨어를 연결하는 일대일 브리지[5, 7], 홈 네트워크 미들웨어를 서브 미들웨어로 설정하고 그것들을 통합하는 계층을 새로 제공하여 여러 홈 네트워크 미들웨어를 동시에 통합하는 일대다 브리지[8, 9, 10, 11]등이 존재한다.

이미 많은 수의 홈 네트워크 미들웨어가 존재하고 있으며, 향후 더 추가될 것이다. 그러므로 복수의 홈 네트워크 미들웨어를 지원할 수 있는 일대다 브리지에 대한 연구가 필요하다. OSGi(Open Service Gateway initiative) Alliance는 1999년 3월에 설립된 비영리 단체로 유비쿼터스 환경에서 상호운용성을 지원하기 위한 홈 게이트웨이 기술이다. OSGi는 자바 기반의 서비스 플랫폼으로, 독립적인 자바 가

상머신 상에서 동적인 컴포넌트 모델인 번들의 동작을 관리한다. OSGi는 이러한 풍부한 운용 가능성을 주목받아, 원래의 적용 분야인 홈 게이트웨이를 넘어 데스크톱 운영 플랫폼, 텔레매틱스, 엔터프라이즈, 자동차 등에서 연구되고 있다[8]. UMB는 각각의 홈 네트워크 미들웨어를 담당하는 UMB-Adaptor의 장비 추상화를 통해 모든 장비를 일괄 관리하여 상호운용성을 지원한다. 또한 UMB는 새로운 홈 네트워크 미들웨어의 출현 시 새로운 UMB-Adaptor의 추가와 추상화를 위한 정보를 갱신하는 것만으로 확장성을 제공할 수 있다는 장점이 있다[6]. HOMI는 각 홈 네트워크 미들웨어와 연결되는 에이전트와 이를 관리하는 에이전트 매니저를 통해 장비와 통신하고, 여기서 갱신되는 자료를 중앙에서 관리함으로써 상호운용성을 지원한다. 특히 HOMI는 두 정보와 HOMIL(HOMI Language)를 이용하여 홈오토메이션을 제공한다[11].

2.3 홈 네트워크 미들웨어의 발전방향

홈 네트워크 미들웨어는 스마트 홈 환경에서 발생할 수 있는 산업적 요구사항들을 수용할 수 있도록 발전하고 있다. 먼저 홈 네트워크를 구성하는 네트워크 프로토콜이나 토폴로지 구성에 대한 이슈가 있다[12, 13, 14, 15, 16]. 그리고 이동 단말기를 통한 응용 서비스 제공, 실내와 실외를 아우르는 홈 네트워크 서비스 등 수요자의 사용 방법에 대한 이슈가 있다[17, 18, 19, 20, 21]. 특히 각기 다른 장비들을 제어하는 하나의 지배적 미들웨어만이 존재하는 홈 네트워크 환경을 가정하기는 어려운 만큼, 여러 미들웨어가 공동의 지식을 기반으로 복합적인 상호운용성을 지원할 수 있는 방법에 대한 고려가 필요하다[22, 23].

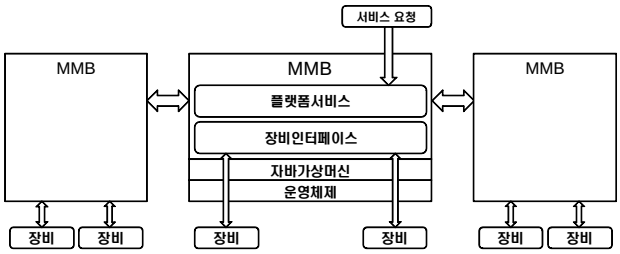
이에 따라 다양한 가전 기기들을 복합적으로 관리할 수 있는 미들웨어 또는 프레임 워크들이 요구되고 있다. 원격 장비 제어, 동적 오류 진단 및 분산 제어를 바탕으로 한 성능 관리 등을 이동에이전트를 통해 제공하는 프레임워크나, 네트워크 보안, 프록시 서비스 및 사용자의 요구 서비스 예측, 홈 오토메이션 등 지능형 서비스를 제공할 수 있는 플랫폼 등에 대한 연구들이 진행되고 있다[24, 25, 26, 27].

3. MMB 설계 및 구현

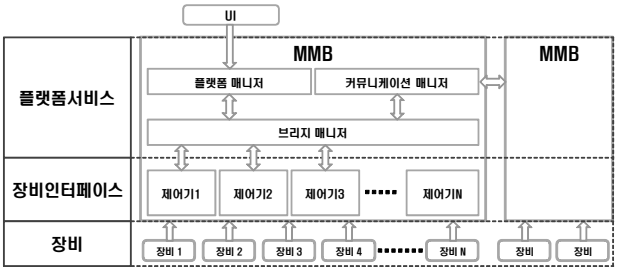
3.1 시스템 아키텍쳐

MMB는 네트워크 내의 각 머신들에 분산되어 동작한다. 각 분산된 브리지들은 동등한 관계를 가진다. 각 브리지는 브리지 인식 과정을 통해 추가적인 설정 없이도 자동으로 연결되며 장비 확장성과 부하 분산을 위해 협업한다. 다음 (그림 1)은 MMB의 전체 구조를 보인다.

독립된 하나의 MMB는 운영체제, 자바가상머신과 주요 컴포넌트인 플랫폼 서비스, 장비 인터페이스로 구성된다. 운영체제는 메모리, CPU 등의 하드웨어 자원을 관리하여 컴퓨터 시스템을 유지한다. 자바가상머신은 MMB가 하드웨어에 독립적으로 동작할 수 있도록 플랫폼 독립성을 지원한다.



(그림 1) MMB의 전체 구조



(그림 2) MMB의 시스템 아키텍쳐

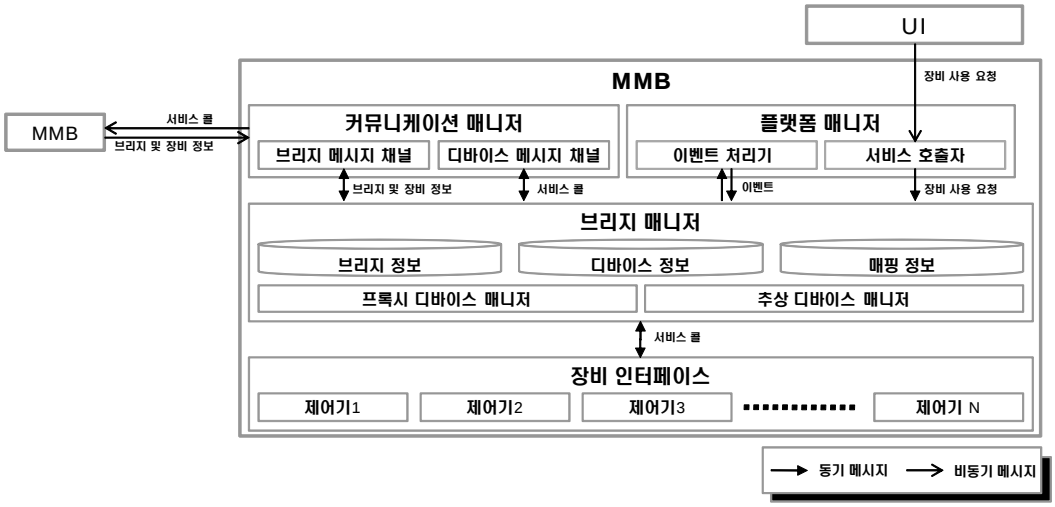
주요 컴포넌트인 플랫폼서비스와 장비 인터페이스는 운영체제와 자바가상머신의 지원을 받아 동작한다. 플랫폼 서비스는 서비스 요청을 받아 자신의 장비 인터페이스를 통해, 또는 다른 브리지의 장비 인터페이스를 통해 처리한다. 이를 위해 플랫폼 서비스는 다른 브리지를 탐색하고, 다른 브리지와 장비에 관한 정보를 서로 교환하고 보관한다. 장비 인터페이스는 각 홈 네트워크 미들웨어의 프로토콜과 운영 기법을 사용하여 장비의 탐색, 정보 전달, 서비스 호출, 상태 처리 등의 연산을 통해 서브 미들웨어의 홈 네트워크 장비와의 통신을 수행한다. (그림 2)는 MMB의 시스템 아키텍쳐를 보인다.

MMB의 플랫폼 서비스는 플랫폼 매니저, 브리지 매니저, 커뮤니케이션 매니저로 구성되어 서비스 요청과 타 브리지와의 통신을 담당한다. 플랫폼 매니저는 사용자의 서비스 호출과 이벤트 처리 등 플랫폼에 소속된 장비들의 제어를 담당한다. 브리지 매니저는 플랫폼 매니저가 서브 미들웨어와 무관하게 이기종 장비들을 제어할 수 있도록 브리지 서비스를 제공한다. 타 브리지를 통해 제어해야할 경우, 커뮤니케이션 매니저를 이용해 메시지를 전달한다. 커뮤니케이션 매니저는 타 브리지와 타 브리지에 소속된 장비들과의 메시지 통신을 제공한다. 이를 통해 브리지 탐색, 소속 장비 변경을 통한 부하 분산, 정보 전달, 서비스 호출 등이 가능하다.

장비 인터페이스는 홈 네트워크 장비들과의 통신을 담당한다. 브리지 매니저는 각 서브 미들웨어를 담당하는 장비 인터페이스내의 제어기를 통해 홈 네트워크 장비들과 통신할 수 있다.

3.2 플랫폼 서비스

플랫폼 서비스는 장비 인터페이스 상단에 위치한다. 플랫폼 서비스는 UI를 통해 서비스 콜을 받고, 장비 인터페이스



(그림 3) 플랫폼 서비스

를 통해 서비스 콜을 수행하거나 이벤트를 처리한다. (그림 3)은 플랫폼 서비스의 세부 구조를 보인다.

플랫폼 서비스는 크게 플랫폼 매니저와 브리지 매니저, 커뮤니케이션 매니저로 구성된다. 플랫폼 매니저는 서비스 호출자와 이벤트 처리기를 통해 장비 사용 요청을 다룬다. 서비스 호출자는 UI를 통해 사용자로부터 받은 서비스 콜을 브리지 매니저에게 전달한다. 이벤트 처리기는 브리지매니저로부터 각 장비의 상태 변화 메시지를 받아, 미리 입력된 조건에 충족하면 서비스 콜을 발생시킨다.

3.2.1 MMB 초기화

MMB의 플랫폼 서비스는 먼저 네트워크 내의 다른 브리지를 탐색하고 연결하는 작업이 필요하다. 이러한 작업들은 자동적으로 수행되기 때문에, 사용자의 추가적인 조작이 필요하지 않다.

MMB가 기동되면, 먼저 MMB는 플랫폼 매니저를 제어하는 데몬 스레드를 생성한다. 그리고 파일로부터 플랫폼 정보 및 정책들을 읽는다. 이후 플랫폼 서비스를 제공하기 위한 커뮤니티 매니저, 브리지 매니저를 초기화하며 네트워크 내에 있는 브리지를 탐색한다. UDP를 통해 MMB 탐색 메시지를 브로드캐스트 한다. 네트워크 내에 존재하던 기존의 MMB가 탐색 메시지에 응답할 경우, 이 MMB를 통해 이미 구성되어있는 브리지와 장비에 관한 정보를 수신한다. 그리고 자신에 대한 정보를 기존의 MMB에게 알린다.

3.2.2 장치 추상화

브리지 초기화 과정이 종료되면, 플랫폼 서비스와 연결된 장비 인터페이스가 동작을 시작한다. 장비 인터페이스에 소속된 각각의 제어기는 홈 네트워크 미들웨어를 구현한다. 그리하여 각 홈 네트워크 미들웨어를 이용하는 장비들과 통신한다. 이 통신을 바탕으로 네트워크 내의 장비들을 탐색하고, 장비의 정보를 바탕으로 장비를 추상화시켜 브리지 매니저에게 보낸다. 브리지 매니저는 추상화된 장비를 보관한다. 장치 추상화 기능은 MMB가 동작하는 동안 계속 수행된다. 세부적인 장비 추상화 과정은 장비 인터페이스의 제어기들이 담당한다.

3.2.3 서비스 콜 처리

서비스 콜 처리 기능은 플랫폼 매니저의 메시지 처리에 의해 진행된다. 플랫폼 매니저의 서비스 호출자는 UI 또는 이벤트 처리기로부터 서비스 콜을 수신한다. UI는 사용자의 입력에 의해, 이벤트 처리기는 장비의 상태 변경 체크를 통해 서비스 콜을 발생시킨다. 서비스 콜을 수신한 플랫폼 매니저는 서비스 콜을 브리지 매니저에게 전달한다. 브리지 매니저는 보관된 브리지와 장비에 관한 데이터를 바탕으로 서비스 콜을 내부에서 수행할지, 외부 브리지를 통해 수행할지를 결정한다. 만약 로컬의 장비 인터페이스에서 제어할 수 있는 미들웨어와 장비일 경우, 내부의 장치인터페이스에서 서비스 콜을 수행한다. 그 외의 경우, 즉 내부에서 수행할 수 없을 경우, 프록시 디바이스 매니저는 커뮤니케이션 매니저를 이용해 외부 브리지를 통한 서비스 콜을 수행한다.

3.2.4 브리지 협업 및 부하 분산

장비 인터페이스의 제어기가 지원하지 않는 미들웨어나 장비에 대한 서비스 콜이 있을시, MMB는 네트워크로 연결된 다른 브리지를 이용하여 서비스 콜을 수행한다. 브리지 간 협업은 프록시 디바이스 매니저와 커뮤니케이션 매니저를 통해 이루어진다.

프록시 디바이스 매니저는 먼저 서비스콜의 목표 미들웨어와 목표 장비를 다룰 수 있는 브리지를 선택한다. 그리고 커뮤니케이션 매니저는 선택한 브리지에게 서비스 콜을 보낸다. 만약 목표 미들웨어와 목표 장비를 다룰 수 있는 브리지를 찾을 수 없다면 오류를 발생시킨다. 이때 서비스콜을 전달할 목표 브리지 선택은 굉장히 중요하다. 부하분산에 대한 고려 없이 무작위로 브리지를 선택하면, 지원하는 미들웨어와 장비의 종류가 많은 브리지에 부하가 집중될 수

```
Algorithm proxy_device_manager.recalculate
{
        a load of local bridge = a load of local bridge / 2
        if a load of local bridge < notified value*0.5 or a load of local bridge < notified value*1.5 then
                notify a load of local bridge
                notified value = a load of local bridge
        end if
}
```

(그림 4) 부하 재계산 알고리즘

```
Algorithm proxy_device_manager.select_bridge
Input : target_middleware, target_device
Output : target_bridge
{
        for bridge = every information of bridge in repository_manager
                if bridge can support target_middleware and target_device then
                        possibility of selection of bridge = 1/(a load of bridge^2)
                        the sum of possibility += possibility of selection of bridge
                else if
                        possibility of selection of bridge = 0
                end if
        end for
        rv = random( 0~ 1.0)
        for bridge = every information of bridge in repository_manager
                rv -= possibility of selection of bridge / the sum of possibility
                if rv < 0 then return bridge
        end for
        return null
}
```

(그림 5) 브리지 선택 알고리즘

있다. (그림 4)는 MMB의 부하 재계산 알고리즘이며 (그림 5)는 MMB의 브리지 선택 알고리즘을 보인다.

부하분산 알고리즘은 브리지 선택 알고리즘과 부하 재계산 알고리즘을 통해 이루어진다. 브리지 선택 알고리즘은 자신이 알고 있는 각 브리지의 알려진 여러 브리지의 처리량을 계산하여 각 브리지의 우선순위를 구한다. 그리고 각 브리지의 우선순위를 통해 각 브리지의 선택률을 계산한다. 부하 재계산 알고리즘은 브리지 선택에 있어 최근 데이터를 중시하도록 처리량 값을 재조정하며, 현재의 처리량 값과 외부에 알려진 자신의 처리량의 차이가 50% 이상일 경우 새로운 값을 외부에 알린다. 본 부하분산 알고리즘에 의해 브리지 선택률은 아래 수식 (1)과 (2)의 공식을 통해 확률을 계산하게 된다.

$$V_i = \frac{1}{\left\{\sum_t^{\infty} R_i^t \times \left(\frac{1}{2}\right)^t\right\}^2} \cdots (1)$$

$$P_a = \frac{V_a}{\sum V_i} \cdots (2)$$

R_i^t = 브리지i의 시간 t당 처리량
V_i = 브리지i의 우선순위
P_a = 브리지a의 선택율

수식 (1)은 각 브리지별 우선순위를 계산하는 공식이다. 먼저 해당 브리지의 시간별 부하량을 더한다. 이때 부하량은 시간에 따라 시간이 지날수록 비중은 절반으로 떨어지게 된다. 이렇게 계산된 해당 브리지의 부하량 합이 제곱된 값의 역이 해당 브리지의 우선순위가 되게 된다. 수식 (1)에 따라 각 브리지는 부하가 많으면 많을수록 우선순위가 낮아진다.

수식 (2)는 그렇게 계산된 브리지별 우선순위를 바탕으로, 한 서비스 콜의 브리지 선택률을 계산하는 수식이다. 각 서비스 콜은 호출하려는 목적 장치에 따라 거쳐 갈 수 있는 브리지 후보들이 다르다. 각 서비스 콜은 서비스 콜을 전달받을 브리지 후보들의 우선순위의 비중에 따라 브리지 선택률을 계산한다. 즉 서비스 콜이 브리지 a로 전달될 확률은, 브리지 a의 우선순위를 전달 가능한 모든 브리지의 우선순위의 합으로 나눈 값이 된다. 수식 (2)에 따라 상대 우선순위가 높은 브리지가 서비스 콜을 전달받을 확률이 높아지게 된다. 이렇듯 본 알고리즘은 확률에 기반을 두기 때문에, 난수 발생기의 난수 영역이 고른 분포를 보이지 않을 경우, 특정 브리지에 과도하게 메시지 콜이 전달되어 많은 부하를 발생시키는 문제가 발생할 수도 있다.

3.3 장비 인터페이스

장비 인터페이스는 플랫폼 서비스와 홈 네트워크 미들웨어의 장비를 연결한다. MMB의 플랫폼 서비스는 장비 인터페이스를 통해 홈 네트워크 장비에게 서비스 콜을 전달한다.

장비 인터페이스는 각각의 홈 네트워크 미들웨어를 구현한 제어기로 구성된다. 이 제어기의 종류가 한 브리지가 다를 수 있는 홈 네트워크 미들웨어의 종류를 결정한다. 제어

기는 여러 홈 네트워크 디바이스를 찾는다. 그리고 제어기는 장비의 정보를 바탕으로 추상 장비 정보를 생성하여 추상 디바이스 매니저에게 전송한다. 추상 디바이스 매니저는 이렇게 전송된 추상 장비 정보를 저장하고, 이 자료를 참조하여 서비스 콜을 수행한다. 그러므로 장비 인터페이스는 추상 디바이스 매니저와 연결되어 추상 장비 정보를 생성해 전달할 수 있는 제어기만 구현하면 되며, 이 외의 제약은 없다.

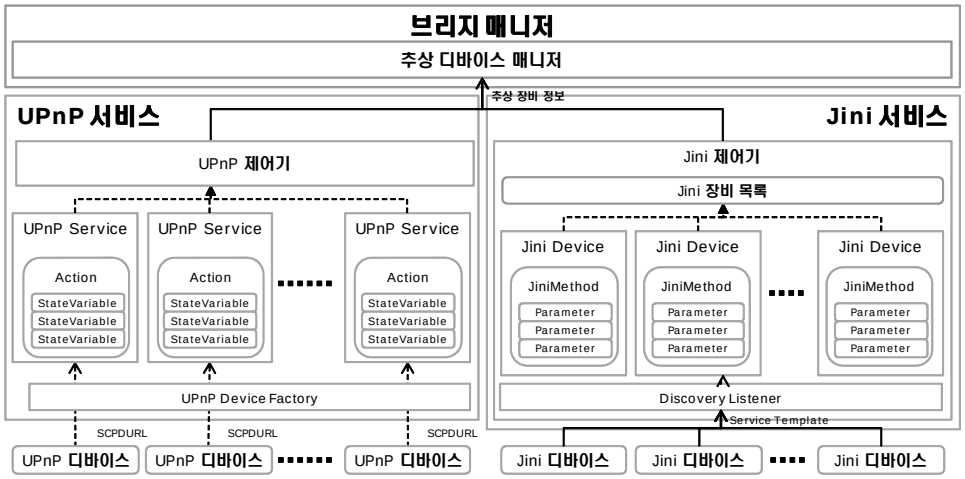
3.4 구현

MMB의 상호운용성 지원 기능을 보이기 위해, UPnP 및 Jini를 제어할 수 있는 두 제어기를 가진 MMB와 Jini로 구현된 시계인 JiniClock장비와 UPnP로 구현된 오디오인 AudioMediaRenderer를 구현하였다. (그림 6)은 구현된 두 제어기를 보인다.

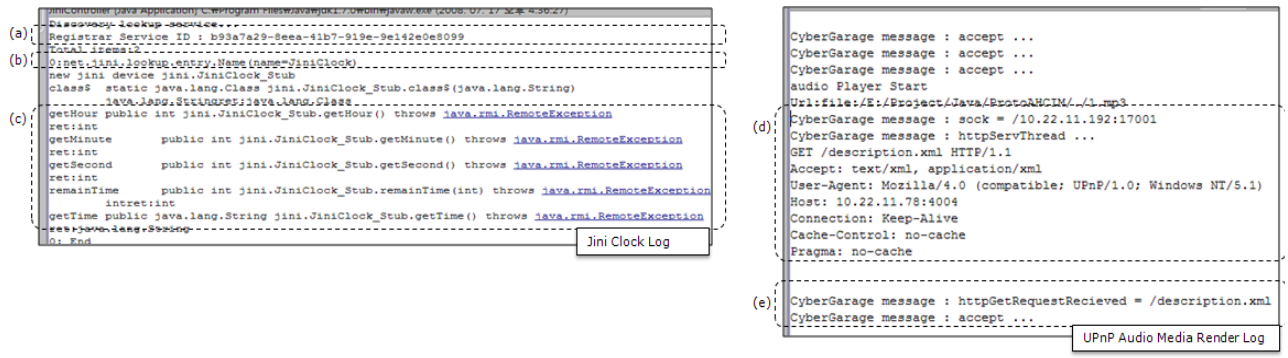
UPnP 제어기는 CyberLink for java와 Apache Xerces를 사용하여 구현하였다. UPnP 제어기는 UPnP의 컨트롤 포인트로 동작한다. UPnP 제어기는 SSDP(Simple Service Discovery Protocol) 메시지를 Cyberlink for Java의 리스너 처리를 통해 수신하여, UPnP 추상 장비 생성 과정을 수행한다. 이를 통해 먼저 UPnP 장비와 서비스에 관한 XML 정보를 받은 UPnP 장비 생성기는 UPnP의 Service, Action, Parameter, StateVariable을 추상 장비, 추상 서비스, 추상 파라미터로 추상화한다. UPnP 장비 생성기는 추상화된 정보를 조합하여 하나의 추상 장비를 생성한 후, 추상 장비를 추상 디바이스 매니저에 등록한다. 추상 디바이스 매니저는 등록된 추상 장비로 UPnP 장비를 제어할 수 있다.

Jini 제어기는 Sun사의 Jini 라이브러리를 사용해 구현하였다. Jini 제어기는 Jini 룩업 서비스의 접근을 통해 Jini장비에 관한 정보를 가져온다. 이후 Jini Discovery Listener 이벤트를 통해 Service Template을 가져온 후 Service Template의 인터페이스를 자바의 Reflect 기능을 이용해 분석하여 Jini장비의 서비스를 분석하고, 메소드, 인자의 데이터 타입, 결과 인자의 데이터 타입 등을 받아 추상 장비를 생성한다. 생성된 추상 장비는 추상 디바이스 매니저에 등록된다.

(그림 7)은 Jini제어기가 Jini장비인 JavaClock을 등록하여 출력된 로그와 이벤트 처리로 UPnP제어기에 의해 호출되어 동작하기 시작한 AudioMediaRenderer를 보인다.



(그림 6) UPnP 및 Jini 제어기



(a) Jini Service ID
(b) 발견된 장치
(c) 발견된 장치의 서비스

(d) UPnP 라이브러리 초기화
(e) UPnP 장비 탐색 시작

(그림 7) 실행 결과

4. MMB 평가 및 검증

4.1 브리지 간 구조 비교

본 절에서는 본 논문에서 제안하는 MMB와 홈 네트워크 미들웨어 간 상호운용성을 지원하는 다른 브리지 시스템들을 비교한다. 다음 (그림 8)은 평가 대상인 MMB와 다른 브리지들의 구조에 대한 비교이다.

(a) OSGi : OSGi는 홈 게이트웨이로 자바 기반의 서비스 플랫폼이다. OSGi는 다양한 번들 프로그램을 동작시킬 수 있다. 또한 모듈 간 동적 관계와 의존을 효과적으로 관리함으로써, 동적인 컴포넌트 모델을 제공한다[8]. 하지만 OSGi는 여러 자바 컴포넌트를 동작시킬 수 있는 플랫폼만 지원할 뿐, 여러 브리지의 관계나 협업에 대한 고수준의 서비스를 제공하지 않는다. 이는 개발자에게 자율성을 부여하지만, 동시에 중복 서비스 제거, 부하 분산 등 상호 운용성 제공을 위한 많은 부담들을 개발자에게 안겨준다.

(b) UMB(Universal Middleware Bridge) : UMB는 브리지의 중심을 이루는 UMB Core와 각각의 홈 네트워크 미들웨어와 연결되는 UMB Adaptor의 연결을 통해 홈 네트워크 미들웨어 간 상호운용성을 지원한다. 이러한 구조를 통해 UMB는 유연한 브리지 확장성을 제공한다. 새로운 홈 네트워크 미들웨어가 나타나면, 새 미들웨어를 제어하는 UMB Adaptor를 추가해 바로 지원하는 홈 네트워크 미들웨어의 수를 늘릴 수 있다. 하지만 UMB에는 부하 분산에 대한 고려가 없다.

(c) HOMI(HOmenetwork Middleware for Interoperability) : HOMI는 각 홈 네트워크 미들웨어와 연결되는 에이전트를 두고, 이 에이전트를 HOMI 서버의 에이전트 매니저로 관리한다. 그리고 HOMI 서버는 장비 정보와 장비 상태, 상황 정보 등 여러 정보를 한 곳에서 관리함으로써 이 세 정보를 이용한 홈 오토메이션 처리에 높은 효율을 보인다. 하지만 모든 정보가 하나의 저장소에 하나의 형태로 저장되는 만큼, 새롭게 추가된 홈 네트워크 미들웨어의 구조에 따라, HOMI 서버의 저장소 구조가 변해야 하는 단점이 있다. 이는 브리지 확장성에 좋지 않다.

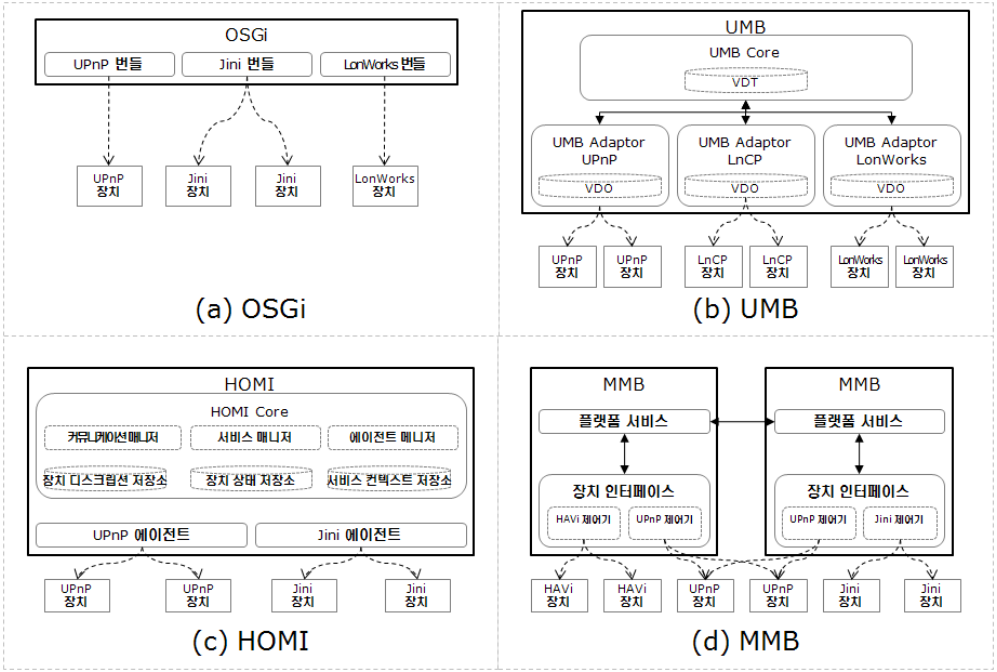
(d) MMB(Multi-Middleware Bridge) : MMB는 먼저 각각의 홈 네트워크 미들웨어와 연결되는 장비 인터페이스를 통해 직접적인 브리지 기능을 수행한다. 그리고 플랫폼 서비스를 통해 네트워크 내의 다른 브리지와 협업함으로써 부하 분산과 확장성을 제공한다. MMB는 새로운 제어기를 가진 새 브리지를 인식함으로써 추가된 새 브리지를 통해 지원 가능한 미들웨어의 종류를 늘린다. 또한 서비스 콜 전달에 대한 부하 분산을 통해 성능을 향상시킨다.

4.2. 부하분산 알고리즘 성능 평가

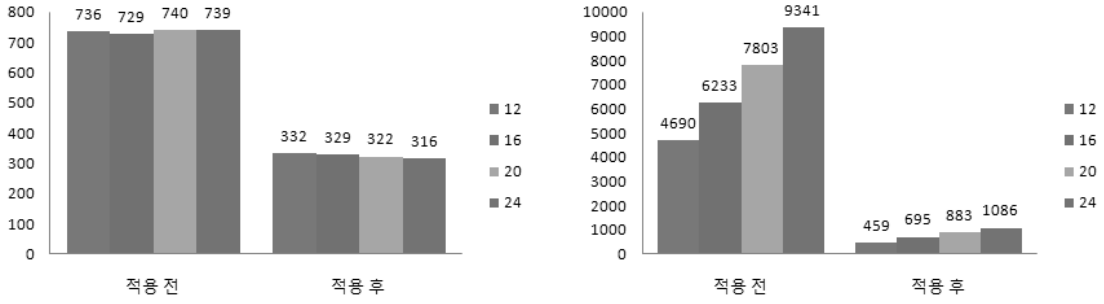
MMB의 부하분산 알고리즘은 다른 브리지를 통해 처리되는 외부 서비스 콜이란 부하를 브리지 별로 최대한 균등하게 분산하는 것이 목적이다. 또한 이 과정에서 발생하는 정보 교환 패킷의 오버헤드는 최소화 되어야 한다.

4.2.1 평가 환경 구축

각각 가상의 미들웨어를 사용하는 가상 장비를 자바로 구현하였다. 또한 이 가상 장비들을 제어할 수 있는 가상의 브리지를 구현하였다. 가상의 장비와 미들웨어를 에뮬레이션 함으로써 실제 미들웨어를 사용했을 때 미들웨어와 장비



(그림 8) 상호운용성 지원을 위한 네 브리지의 구조도



(a) 부하분산 알고리즘 적용에 따른 브리지별 부하량 분산

(b) 부하분산을 위한 오버헤드 트래픽

(그림 9) 제안된 알고리즘을 적용에 따른 시뮬레이션 결과

의 차이에 의해 발생할 수 있는 성능 차이를 제거하였다.

가상 장비의 사용 미들웨어 설정과 브리지의 미들웨어 지원 능력 설정 등은 각 시도 때마다 달라진다. 즉 시도 때마다 각 브리지는 다른 환경을 접하게 되고, 이후 부하분산 알고리즘을 통해 환경에 맞는 브리지별 전달 우선순위를 변경함으로써 부하를 분산한다.

평가를 위해 3, 4, 5, 6개의 미들웨어, 12, 16, 20, 24개의 브리지와 브리지당 50개의 홈 네트워크 장비가 바인딩된 환경을 각각 설정하였다. 이렇게 총 16가지 경우에 대해 각각 100회씩, 총 1600번의 시뮬레이션을 시도하였다.

4.2.2 알고리즘 성능 평가

본 논문은 얼마나 부하량이 분산되었는가, 부하량 분산을 위한 정보교환 패킷의 양을 얼마나 최소시켰는가, 두 가지 기준으로 부하분산 알고리즘을 평가하였다.

상기 환경을 통해 얻어진 시뮬레이션 통계를 홈 네트워크 환경의 규모에 따라 정리하였다. 즉 12개의 브리지와 600개의 장비가 있는 환경, 16개의 브리지와 800개의 장비가 있는 환경, 20개의 브리지와 1000개의 장비가 있는 환경, 24개의 브리지와 1200개의 장비가 있는 네 가지 환경으로 정리하였다. 다음 (그림 9)는 평가 통계 결과를 보인다.

먼저 (그림 9) (a)는 부하분산 알고리즘이 적용되기 전과 후의 브리지별 부하량 분산의 변화를 보인다. 두 경우 모두 부하량, 즉 서비스 콜 처리량의 총 합은 같다. 하지만 알고리즘 적용 전과 적용 후의 부하 분산은 다르다. 부하분산 알고리즘이 적용됨에 따라 각 브리지의 서비스 콜이 다른 브리지의 가용능력에 따라 가능한한 균일하게 분포되었고, 각 브리지간 서비스 콜 규모가 비슷해졌고 이에 따라 분산(Variance) 수치가 작아졌다.

(그림 9) (b)는 알고리즘 동작을 위해 요구되는 정보 교환 패킷, 즉 오버헤드 트래픽의 규모를 보인다. 정보 교환 패킷은 각 브리지가 부하 전달을 위한 브리지 우선순위를 설정하기 위해 필요하다. 따라서 즉 다른 장비들의 정보가 없고 우선순위 설정이 정확하지 않은 '안정화 전' 상태에서는 우선순위 재계산이 빈번하게 일어나고, 따라서 정보 교환 패킷의 규모가 최대화 된다. 또한 각 브리지의 정보 교환 패킷 규모는 네트워크내 연결된 다른 브리지의 개수와 비례한다. 중앙 서버가 없이 스타토폴로지로 브리지가 구축된 까닭에 각 브리지는 자신과 통신하는 브리지의 개수만큼 패킷을 보내야 하기 때문이다. 하지만 이러한 오버헤드 트래픽의 규모는 정보가 충분히 교환되어 우선순위 설정이 유효값을 찾음으로써, 평균 10.94%로 감소한다.

5. 결 론

홈 네트워크 미들웨어 표준의 난립과 미들웨어 간의 이질성은 반드시 해결해야할 문제이다. 브리지 시스템은 미들웨어 간 상호운용성 지원을 통해 이러한 문제를 해결한다. 이 경우, 여러 미들웨어를 동시에 다루는 브리지 시스템의 특성상 동적 확장성과 성능에 대한 고려는 매우 중요하다.

본 논문은 홈 네트워크의 상호운용성 지원을 위한 다중 미들웨어 브리지(MMB)를 제안하였다. MMB는 네트워크 내의 여러 브리지가 메시지 통신을 통해 협업하는 분산구조를 갖는다. 각 브리지는 홈 네트워크 장비와 직접 통신하는 장비 인터페이스와 이를 이용해 다른 브리지와 협업하는 플랫폼 서비스로 나뉜다. MMB는 플랫폼 서비스와 장비 인터페이스의 분리를 통해 기존 시스템을 유지하면서 새로운 홈 네트워크 미들웨어를 추가 할 수 있는 동적 확장성을 제공하며, 각 브리지의 서비스 호출 회수를 바탕으로 한 한 부하 분산 알고리즘을 바탕으로 성능 향상을 꾀하였다. 또한 본 논문은 구현과 평가를 통해 MMB의 상호운용성 지원 능력을 검증하였으며, 부하분산 알고리즘의 성능 향상 효과를 보였다.

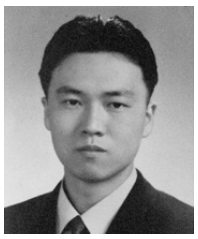
향후 연구로 다른 홈 네트워크 미들웨어에 대한 분석을 통한 장비 인터페이스의 지원 미들웨어 확장과 각 미들웨어의 세부 기법까지 고려된 향상된 부하분산 알고리즘, 그리고 상황인지 시스템, 이벤트 처리 시스템 등 홈 네트워크 서비스를 위한 응용 기법에 대한 연구 등을 진행할 것이다.

참 고 문 헌

[1] UPnP, http://www.upnp.org
[2] HAVi, http://www.havi.org
[3] Jini, http://www.jini.org

[4] DLNA, http://www.dlna.org
[5] Allard.J, Chinta.V, Gundala.S, and Richard.G.G, "Jini meets UPnP: an Architecture for Jini/UPnP Interoperability," Applications and the Internet, 2003.
[6] Kyeong-Deok Moon, Young-Hee Lee, Chang-Eun Lee, and Young-Sung Son, "Design of a Universal Middleware Bridge for Device Interoperability in Heterogeneous Home Network middleware," Consumer Electronics, IEEE Transactions on Vol.51, No.1, 2005.
[7] Donghee Kim, Jun Hee Park, Poltavets Yevgen, KyeongDeok Moon, and YoungHee Lee, "IEEE1394/UPnP Software Bridge," IEEE Transactions on Consumer Electronics, Vol.51, No.1, 2005.
[8] OSGi, http://www.osgi.org
[9] Eiji Tokunaga, Hiro Ishikawa, Makoto Kurahashi, Yasunobu Morimoto, and Tatsuo Nakajima, "A Framework for Connecting Home Computing Middleware," International Conference on Distributed Computing Systems Workshops'02, 2002.
[10] Kyeong-Deok Moon, Young-Hee Lee, Young-Sung Son, and 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, 2003.
[11] Min Chan Kim and Sung Jo Kim, "A Scenario-Based User-Oriented Integrated Architecture for Supporting Interoperability Among Heterogeneous Home Network Middlewares," Lecture Notes in Computer Science, International Conference on Computational Science and Its Applications, Vol.3983/2006, 2006.
[12] Adrian Friday, Nigel Davies, Nat Wallbank, Elaine Catterall, and Stephen Pink "Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments," Wireless Networks, Vol.10, No.6, 2004.
[13] Jongwoo Sung, Daeyoung Kim, Hyungjoo Song, Junghyun Kim, Seong Yong Lim, and Jin Soo Choi, "UPnP based Intelligent Multimedia Service Architecture for Digital Home Network," SEUS/WCCIA, 2006.
[14] Dong-Oh Kang, Kyuchang Kang, Sunggi Choi, and Jeunwoo Lee, "UPnP AV Architectural Multimedia System with a Home Gateway Powered by The OSGi Platform," IEEE Transactions on Consumer Electronics, Vol. 51, No. 1, 2005.
[15] Chuan-Feng Chiu, Hsu.S.J, and Sen-Ren Jan, "The design of UPnP-based home environment over peer-to-peer overlay network," Ubi-Media Computing, 2008.
[16] Chung-Sheng Li, Yueh-Min Huang, and Han-Chieh Chao, "UPnP IPv4/IPv6 Bridge for Home Networking Environment," IEEE Transactions on Consumer Electronics, Vol.54, No.4, 2008.
[17] Jun Ho Park, Myung Jin Lee, and Soon Ju Kang, "CORBA-based Distributed and Replicated Resource Repository Architecture for Hierarchically Configurable Home Network," Journal of Systems Architecture, Vol.51, No.2, 2005.
[18] Kuk-Se Kim, Chanmo Park, and Joon Lee, "Internet Home Network Electrical Appliance Control on the Internet with the UPnP Expansion," International Conference on Hybrid Information Technology, 2006.
[19] Chiu Chi-Huang, Ling Hsien-Tang, Yeh Ping-Jer, and Yuan Shyan-Ming, "An OSGi Platform Connecting Heterogeneous Smart Home Appliances from Mobile Devices," WSEAS Transactions on Computers, Vol.5, No.7, 2006.
[20] Brown.A, Kolberg.M, Bushmitch.D, Lomako.G, and Ma.M, "A SIP-based OSGi Device Communication Service for Mobile Personal Area Networks," CCNC, Vol. 1, 2006.
[21] Xie Li and Wenjun Zhang, "The Design and Implementation of Home Network System using OSGi Compliant Middleware," IEEE Transactions on Consumer Electronics, Vol.50, No.2, 2004.
[22] Miori.V, Tarrini.L, Manca.M, and Tolomei.G, "An Open Standard Solution for Domotic Interoperability," IEEE Transactions on Consumer Electronics, Vol. 52, No. 1, 2006.
[23] Yérom-David Bromberg and Valérie Issarny, "INDISS: Interoperable Discovery System for Networked Services," Lecture Notes in Computer Science, Middleware, Vol. 3790/2005, 2005.
[24 Leelaprute.P, Nakamura.M, Tsuchiya.T, Matsumoto.K, and Kikuno.T, "Describing and Verifying Integrated Services of Home Network Systems," APSEC, 2005.
[25] Haitao Zhang, Fei-Yue Wang, and Yunfeng Ai, "An OSGi and Agent based Control System Architecture for Smart Home," NSC, 2005.
[26] Choon-Gul Park, Jae-Hyoung Yoo, Seung-Hak Seok, Ju-Hee Park, and Hoen-In Lim, "Intelligent Home Network Service Management Platform Design Based on OSGi Framework," Lecture Notes in Computer Science, Management of Convergence Networks and Services, Vol. 4238/2006, 2006.
[27] Gu.T, Pung.H.K, and Zhang.D.Q, "Toward an OSGi-based Infrastructure for Context-aware Applications," IEEE Pervasive Computing, Vol.3, No.4, 2004.

김 연 우
e-mail : daroobil@ece.skku.ac.kr
2007년 성균관대학교 컴퓨터공학과(학사)
2009년 성균관대학교 전자전기컴퓨터공학과 공학석사
관심분야 : 프로그래밍 언어, 인공지능, 리눅스, 네트워크 등



장 현 수

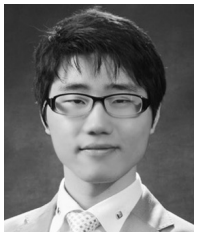
e-mail : jhs4071@ece.skku.ac.kr

2002년 성균관대학교 전자전기컴퓨터공학과 (학사)

2005년 성균관대학교 전자전기컴퓨터공학과 (공학석사)

2006년 현 재 성균관대학교 전자전기컴퓨터공학과 박사과정

관심분야 : 유비쿼터스 컴퓨팅, 이동 에이전트, HCI, 미들웨어 등



송 창 환

e-mail : eerieN@ece.skku.ac.kr

2008년 성균관대학교 컴퓨터공학과(학사)

2009년 현 재 성균관대학교 전자전기컴퓨터공학과 석사과정

관심분야 : 유비쿼터스 컴퓨팅, HCI, 이동 에이전트 등

엄 영 익

e-mail : yieom@ece.skku.ac.kr

1983년 서울대학교 계산통계학과(학사)

1985년 서울대학교 전산과학과(이학석사)

1991년 서울대학교 전산과학과(이학박사)

1996년 한국정보과학회 학회지 편집위원

2000년~2001년 Dept. of Info. and Comm. Science at UCI 방문교수

2005년~2006년 한국정보처리학회 학회지 편집위원장

1993년~현 재 성균관대학교 정보통신공학부 교수

2007년~현 재 성균관대학교 정보통신처 처장

관심분야 : 분산 컴퓨팅, 이동 컴퓨팅, 이동 에이전트, 시스템 보안, 운영체제, 내장형 시스템 등