논문 2006-43SC-2-3

UPnP 기반 로봇 미들웨어

(UPnP based Robot Middleware)

안 상 철*, 이 정 우**, 김 형 곤*

(Sang Chul Ahn, Jung-Woo Lee, and Hyoung-Gon Kim)

요 약

본 논문은 최근 Device용 미들웨어로 부각되고 있는 UPnP를 로봇 미들웨어로 사용하는 것을 제안한다. 본 논문에서는 UPnP의 장점을 보이기 위해서 현재 몇몇 로봇개발에 사용된 TAO CORBA와 비교분석을 하였다. 또한, 하나의 sample 로봇 아키텍처를 잡고, 여기에 UPnP를 로봇 미들웨어로 사용하는 것을 실험하여, 소프트웨어 아키텍처, message mapping, realtime 이슈, 우선순위, 네트웍, 성능, 메모리 용량, 그리고 소프트웨어 배치(deployment) 등의 관점에서 UPnP가 로봇 미들웨어로 사용될 수 있음을 보였다.

Abstract

This paper proposes to use the UPnP as a middleware for robots. It describes the advantages of the UPnP by comparing it with the TAO CORBA that was used in a few robot development projects. We select a sample robot architecture, and examine the possible use of the UPnP as a robot middleware. This paper shows how the UPnP architecture can be applied to building a robot in the view of software architecture, message mapping, realtime, priority, network selection, performance, memory footprint, and deployment issues.

Keywords : Robot middleware, UPnP, CORBA, TAO, RTAI

I. 서 론

근래에 들어 연구자들이 로봇에 점점 더 많은 기능을 포함하고 또한 로봇을 보다 지능적으로 만들려고 노력하면서, 로봇 시스템은 점점 복잡해지고 있다. 이러한 경향은 인간형 로봇을 추구하는 연구 프로젝트인 HRP(Humanoid Robotics Project) 같은데서 잘 볼 수 있다[1]. 이렇게 복잡한 로봇은 주행, 센싱, 조작, 지능의 하위 시스템들을 구성하고 있는 하드웨어, 소프트웨어 component들 간의 많은 상호작용과 작업조정 등이 필요하다. 따라서, 예전의 간단한 기능을 하는 로봇을 개발할 때의 방식과는 다르게 이렇게 복잡한 로봇의 개발에는 시스템적인 개발방식이 필요하다.

시스템적인 개발에는 미들웨어가 필요하게 되는데 요즘의 로봇 개발에는 미들웨어가 필수적으로 사용되고 있다. 앞서 언급한 HRP 프로젝트에서는 OpenHRP라는 플랫폼을 개발하였는데 여기에 CORBA[2]를 미들웨어로 사용하였다. 유럽에서 진행된 OROCOS(Open RObot COntrol Systems) 프로젝트는 로봇의 소프트웨어 component들을 분산시키고, 공유하고, 다시 사용하기 위한 소프트웨어 프레임워크를 개발하였는데, 여기서도 CORBA를 기반으로 하고 있다[3]. Miro는 mobile 로봇을 위한 또 다른 미들웨어로 역시 유럽에서 개발되었다[4]. 이는 TAO CORBA[5]를 기반으로 개발되었고, 일반적인 로봇 제어 기능들을 제공하기 위한 class들을 제공한다. 이와 같이 많은 로봇 연구자들은 미들웨어를 기반으로 소프트웨어 프레임워크를 구축하였고, 그 대부분이 CORBA를 기반으로 하고 있다.

* 정회원, 한국과학기술연구원 영상미디어연구센터
(Imaging Media Research Center, KIST)
** 학생회원, 한국과학기술연구원 영상미디어연구센터, 한양대학교 전자통신컴퓨터공학과
(IMRC, KIST, Depr. of Electronics and Computer Engineering, Hanyang University,)
※ 본 연구는 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었음.
접수일자: 2005년8월1일, 수정완료일: 2006년3월13일

CORBA는 분산 컴퓨팅 환경을 위한 미들웨어 아키텍처(architecture)이다. 이는 다양한 응용 프로그램이 네트웍 상에서 공동 작업을 할 수 있게 만드는 공개적이고, 개발업체에 무관한 규정을 가지고 있다. 한편, 최근에는 인터넷이 널리 보급되면서, 컴퓨팅 환경이 dynamic 해지고 있다. 여기서 dynamic 하다는 것은 컴퓨팅 장치(device)가 네트웍에 dynamic하게 연결되었다 분리되었다 할 수 있다는 의미이다. 따라서, 미들웨어도 이를 수용하는 방향으로 발전하고 있다. Jini[6]나 .Net[7], UPnP[8] 등이 이러한 'dynamic' 분산 컴퓨팅 환경을 위한 미들웨어이다. 그림 1은 미들웨어의 computing problem space를 구분한 그림으로, 이 그림에 표시된 미들웨어 중 일부는 사용되지 않고 있지만 분산환경에서 dynamic 분산환경으로 변하면서 미들웨어가 어떻게 변하고 있는지를 잘 보여주고 있다.

본 논문에서는 분산 컴퓨팅 환경을 위해 제안된 UPnP(Universal Plug and Play)를 로봇용 미들웨어로 사용하는 것을 제안하고자 한다. 이는 앞으로 로봇 시스템 자체도 복잡해지면서 dynamic 분산 컴퓨팅 환경으로 갈 것으로 예상되기 때문이다. 미래에는 로봇의 모든 component가 모듈화되고 사람 몸이 신경으로 연결되어 있는 것과 같이 네트웍으로 연결될 것으로 예상된다. 이러한 접근방법은 이미 여러 프로젝트에서 채택되고 있다. 그리고, 미래에는 로봇의 모듈들이 원하면 언제나 분리되거나 연결될 수 있을 것인데, 그렇게 되면 로봇 시스템 자체가 하나의 dynamic 분산 컴퓨팅 환경이 되는 것이다. 따라서, 이에 적합한 미들웨어가 필요하게 된다.

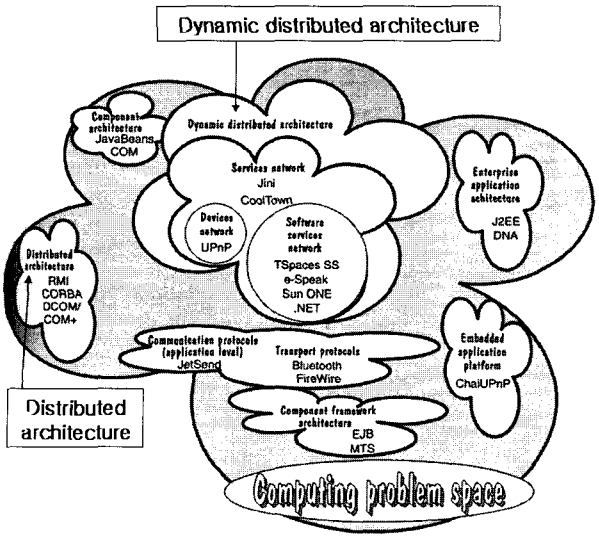


그림 1. 컴퓨팅 환경과 미들웨어
Fig. 1. Computing problem space of middlewares[9].

또 다른 관점으로 보면, 미래의 로봇은 지금 TV나 냉장고와 같이 하나의 가전기기가 될 것이다. 그렇게 되면 로봇은 지금 노트북처럼 집이나 사무실의 네트웍 환경에 dynamic하게 연결되었다 떨어지게 될 것이다. 즉, 로봇은 외부 dynamic 분산환경에 대응하여 dynamic하게 통신을 할 수 있어야 하고, 미래 로봇 미들웨어는 이를 지원하여야 한다. 따라서, dynamic 분산환경을 배경으로 탄생한 UPnP는 미래 로봇 시스템을 위한 미들웨어로서 하나의 후보가 될 수 있는 것이다.

II. TAO CORBA와 UPnP

CORBA는 Common Object Request Broker Architecture의 약자로 Object Management Group (OMG)라는 단체에서 제안한 분산 컴퓨팅 환경을 위한 언어와 플랫폼에 독립적인 미들웨어 아키텍처이다. 이는 이름에 조금 언급되는 것과 같이 객체(object) 지향 프로그래밍 모델을 지원한다. CORBA 아키텍처의 기본 엔진은 ORB라고 불리는 object request broker이다. 이 ORB는 object 버스라고도 불리는데, 클라이언트와 서버의 사이에서 서버측의 object를 찾고, method invocation을 전달하고, 결과를 클라이언트로 반환하는 일련의 처리과정을 책임진다. CORBA는 object간의 인터페이스를 정의하기 위해서 Interface Definition Language(IDL)이라는 것을 사용한다. CORBA는 IDL을 여러 언어로 변환하는 mapping을 제공하여, 언어와 플랫폼에 독립적인 특성을 유지한다. 클라이언트와 서버 프로그램은 stub과 skeleton이라는 proxy들을 사용하여 ORB와 통신하는데, stub과 skeleton은 IDL을 컴파일하여 인터페이스를 정한다. 클라이언트는 ORB의 naming 서비스를 통하여 서버 객체를 찾고 적절한 서비스를 호출한다. 이를 위해서 클라이언트와 서버 객체는 사전에 서비스 정보를 가지고 연결이 되어야 한다.

TAO는 The ACE ORB의 약자로 open 소스, 고성능, 실시간 CORBA ORB 시스템을 표방하고 있다. 이는 분산 실시간 embedded 시스템을 위해 개발된 것이다[5]. TAO는 CORBA에 ACE[10]를 결합하고, 실시간 I/O sub 시스템과 high speed 네트웍 인터페이스를 추가하여 확장한 것이다. 이는 응용 프로그램에서 통신과 메모리 리소스를 관리할 수 있도록 한다. 또한, QoS 속성을 제공하기 위해서 run-time 스케쥴러를 사용하고 있다. TAO는 실시간 OS의 많은 서비스를 사용한다.

UPnP(Universal Plug and Play)는 홈 네트워크 등과 같은 소규모 네트워크 내에서 지능화된 기기들을 peer-to-peer 방식으로 연결하기 위한 아키텍처이다. 이는 원래 Microsoft에서 개발되었지만 지금은 UPnP 포럼에서 표준을 관리하고 있다. 2005년 11월 현재 Microsoft, Intel, Dell, HP 등과 같은 약 760개의 기업이 UPnP Forum의 회원으로 활동 중이며, 22개 회원사로 이루어진 UPnP Steering Committee가 UPnP Forum을 주도하고 있다. UPnP는 TCP/IP, SSDP, SOAP, GENA, HTTP 및 XML 등과 같은 인터넷 표준과 기술을 기반으로 하고 있으며, 서비스 기반의 프로그래밍 모델을 제공하고 있다.

UPnP 아키텍처의 주요 컴포넌트들은 Control Point, Device, 그리고 Service이다. UPnP Device는 서비스를 제공하는 본체이며, Control Point는 서비스 요청자이다. 그리고 서비스는 Device에 의해서 구현된 기능적 단위가 된다. UPnP Device는 여러 서비스들을 포함할 수 있다. 그림 2는 이들 컴포넌트들의 관계를 표현하고 있다.

UPnP 네트워킹은 Addressing, Discovery, Description, Control, Eventing, Presentation의 6단계를 통해서 이루어지게 된다. Addressing 단계에서는, Device가 네트워크에 연결되면 Auto-IP를 통해서 자동으로 주소를 얻게 된다. Discovery 단계에서는 SSDP를 통해 Device가 네트워크상에 자신을 알리거나 또는 Control Point가 관심있는 Device를 찾게 된다. Description 단계에서는 Control Point가 찾은 Device의 Description을 요청해서 가져오게 된다. 이때 Control Point는 받은 Device Description을 보고 서비스들에 대한 Service Description도 가져오게 된다. Control 단계에서는 Control Point가 Device와 그에 대한 Service의 Description을 가지고, SOAP를 통해서 원하는 Action을 요청하게 된다. Eventing 단계는 Control Point가 Device의 상태 변화를 Event로 줄

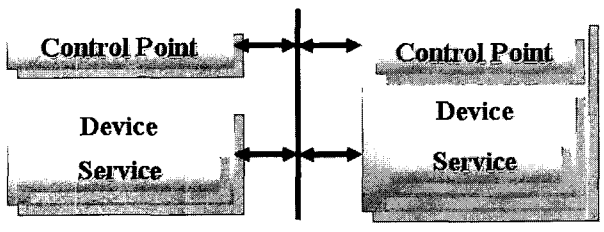


그림 2. UPnP 주요 컴포넌트들과 그 관계
Fig. 2. Main components in UPnP architecture.

것을 요청하게 된다. 이러한 단계를 표 1에서 나타내고 있다.

UPnP에서는 이러한 모든 동작이 자동으로 이루어지기 때문에 zero-configuration 특성을 가진다고 일컬어진다. 또한, 이러한 특성 때문에 UPnP에서는 Device나 Control Point가 네트웍에 dynamic하게 들어왔다 나가는 것이 자연스럽게 지원되는 것이다. 표 2에서는 TAO와 UPnP의 특성을 비교하였다. UPnP가 TAO와 다른 첫 번째 특성은 앞서 언급한대로 dynamic 배치(deployment)가 가능하다는 점이다. 두 번째는 그 scope과 프로그래밍 모델이다. TAO는 object를 대상으로 하는 반면에 UPnP는 서비스를 대상으로 한다. 여기서 서비스는 내부에서의 처리 프로세스를 밖으로 드러

표 1. UPnP 네트워킹의 단계
Table 1. UPnP networking steps.

Steps	Description
Addressing	Control point and device get addresses
Discovery	Control point finds interesting device
Description	Control point learns about device capabilities
Control	Control point invokes actions on device
Eventing	Control point listens to state changes of device
Presentation	Control point controls device or views device status using HTML UI

표 2. TAO와 UPnP의 비교
Table 2. Comparison of TAO and UPnP.

분류	TAO	UPnP
Management	Object	Control point, Device, Service
Deployment	Static	Dynamic
Programming model	Object oriented	Service oriented
Scheduling	O	X
Network	ATM, VME, compactPCI, IPv6	Ethernet, IEEE1394, USB, HomeRF
Protocol	GIOP, IIOP	TCP, UDP, IP, HTTP, SSDP, SOAP
OS	LynxOS, VxWorks, QNX, Linux, OS9	Windows, Linux, PocketPC
Interface	Stub, Skeleton	XML documents
Memory Footprint	Big	Small

내지 않으면서 인터페이스와 결과만을 제공하는 단위로 일반적으로 object의 scope 보다는 크다. 또한, UPnP는 TAO보다 작은 크기로 구현될 수 있어서 작은 memory footprint를 가지는 점이 작은 메모리를 가지는 embedded 시스템에서 잘 활용될 수 있다.

UPnP는 또한 DLNA(Digital Living Network Alliance)[11]에도 채택이 되었는데, DLNA는 CE (Consumer Electronics), Mobile 및 PC 업체들이 홈 네트워크 상에 있는 디지털 TV, Set-top box, Mobile phone, PDA, PC등 모든 디지털 기기들의 상호호환성 문제를 해소하기 위한 컨소시엄이다. UPnP는 여기에서 장치 검색 및 제어 표준 프로토콜의 기반이 되고 있다. 따라서, UPnP는 앞으로 가정이나 사무실에서 많이 사용되는 미들웨어로 채택될 가능성이 크다고 하겠다.

III. 로봇 미들웨어로서의 UPnP 검토 및 실험

본 연구에서는 UPnP가 로봇 미들웨어로 사용되기에 적합한가를 여러 가지 측면에서 검토하고 하고자 한다. 로봇 미들웨어는 여러 가지 다른 타입의 로봇에 적용이 가능하도록 일반적이어야 하지만, 본 연구에서는 여러 로봇 타입을 포괄할 수 있는 약간 일반화된 샘플 로봇 아키텍처를 정하고, 여기에 UPnP를 적용하는 것을 검토한다. 본 연구에서 정한 샘플 로봇은 하드웨어적으로는 몇 장의 SBC(single board computer)로 이루어져 있고, 각각은 네트웍으로 연결되어 있는 것으로 하였다. 그림 3(a)는 이러한 하드웨어 구성도를 나타내고 있다. 이들 SBC에는 Linux가 돌고 있고, RTAI[12]와 같은 실시간 OS를 채택하여 사용한다. 각 SBC는 주행, 조작, 비전(vision), 음성인식, 동작계획(planning) 등의 기능을 맡고 있다. 소프트웨어 구조는 기본적으로 소프트웨어 컴포넌트(component) 들로 이루어져 있다. 하나의 소프트웨어 컴포넌트는 내부적으로 간단하거나 복잡한 기능을 구현하고 있고, 다른 컴포넌트들에게 인터페이스(interface)를 제공하여 그 기능을 사용할 수 있도록 한다. 이러한 소프트웨어 컴포넌트들을 조합하면 'planning'이나 'navigation'과 같은 복잡한 기능을 구현할 수 있게 된다. 그림 3(b)는 이러한 소프트웨어 구성을 나타내고 있는데, 여기서 각 블록(block)은 소프트웨어 컴포넌트를 나타낸다. 소프트웨어 컴포넌트들은 여러 개가 하나의 SBC에 위치할 수 있다. 또한, 어떤 컴포넌트들은 하드웨어와 밀접한 연관을 가지는 것들도 있다. 센서와 액츄에이터(actuator)에 관련된 소프트웨어 컴포넌트들이 그러한 예이다. 이러한 컴포넌트들은 실시간 제한(constraint)을 가질 수 있다.

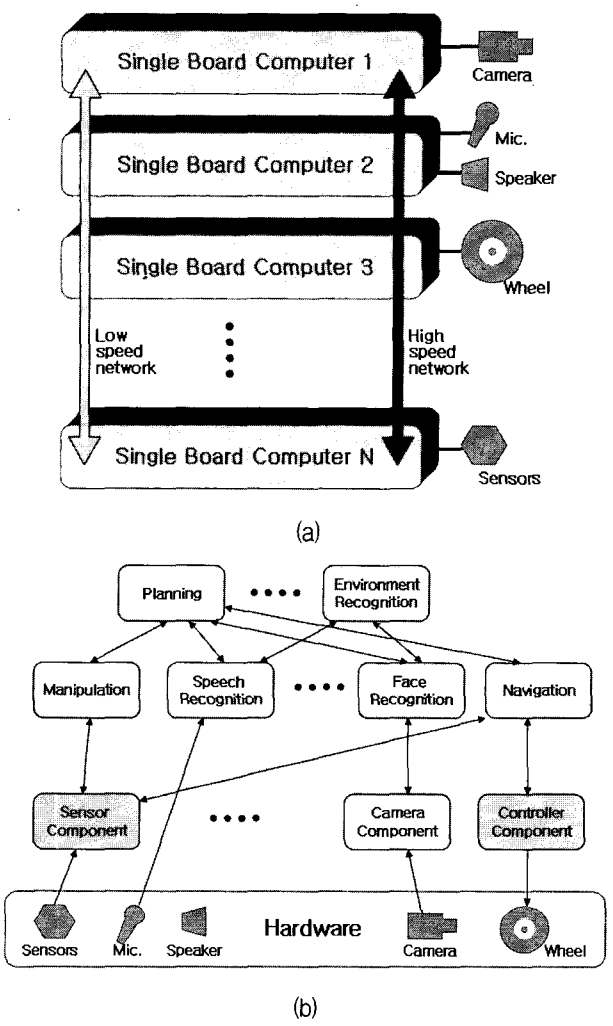


그림 3. 샘플 로봇의 (a) 하드웨어 아키텍처 (b) 소프트웨어 아키텍처

Fig. 3. (a) Hardware architecture and (b) software architecture of the sample robot.

UPnP를 로봇 미들웨어로 사용하기 위해서는 몇 가지 고려되어야 하는 사항들이 있는데, 본 연구에서는 위에서 설명한 샘플 로봇을 대상으로 고려해 본다.

소프트웨어 아키텍처 - 위에서 말한 소프트웨어 아키텍처를 보면 각 소프트웨어 컴포넌트들이 세세한 object 수준으로 나뉘어져 있다기 보다는 보다 큰 기능 위주로 나뉘어 있는 것을 알 수 있다. 이는 로봇에 복잡한 기능을 포함시키려고 할 때, 일반적으로 여러 다른 그룹이 각각의 컴포넌트를 만들게 되는데, 컴포넌트들을 object 단위로 분할하기 보다는 조금 더 큰 수준에서 기능을 나누어 구현하는 경우가 많기 때문이다. 따라서, 이러한 소프트웨어 아키텍처에는 UPnP의 SOA(Service Oriented Architecture)가 적합하다고 할 수 있다. 이때

각 소프트웨어 컴포넌트들은 UPnP의 Device나 Control Point에 대응될 수 있고, 컴포넌트의 인터페이스는 UPnP Device의 Service에 의해 제공될 수 있다.

메시지(Message) - UPnP의 Device와 Control Point는 service invocation과 response를 포함하는 메시지를 전달하여 통신을 한다. 이러한 메시지는 Simple 메시지(지연시간 조건이 별로 없는 작은 메시지), 실시간(Realtime) 메시지(시간 제한조건이 있는 작은 메시지), 메시지 stream(일정 주기를 가진 메시지 sequence), Isochronous(지속적인 media type의 데이터), Burst(대용량 데이터)의 5가지로 분류해 볼 수 있다. Simple 메시지와 실시간 메시지는 UPnP의 네트워킹 동작 중에 Control 단계에 전송될 수 있다. (실시간 메시지에 대한 것은 다음에 다시 설명) 메시지 stream은 UPnP의 네트워킹 동작 중에 Eventing을 이용해 전송이 가능하다. 만일 Eventing에서 시간 변수를 상태(state) 변수로 잡으면 이 상태 변수를 구독(subscribe)하는 Control Point에 주기적인 메시지 전송이 가능할 것이다. 만일 Isochronous와 Burst에서 처럼 전송할 데이터가 많으면 UPnP AV 방식을 사용할 수 있다. UPnP AV는 UPnP를 사용하여 디지털 audio와 video를 전송하기 위한 아키텍처이다. 그림 4는 UPnP AV 아키텍처를 보여주고 있다. UPnP AV에서는 Media server와 Media renderer의 두 가지 타입의 device가 정의되어 있다. Media server는 미디어를 저장하고 있는 device이고, Media renderer는 미디어를 rendering 해주는 device이다. UPnP AV Control Point는 Media server와 Media renderer를 네트웍 상에서 찾고, UPnP action을 통해서 Media server를 Media renderer와 연결시켜 준다. 일단 연결이 되면 두 개의 device는

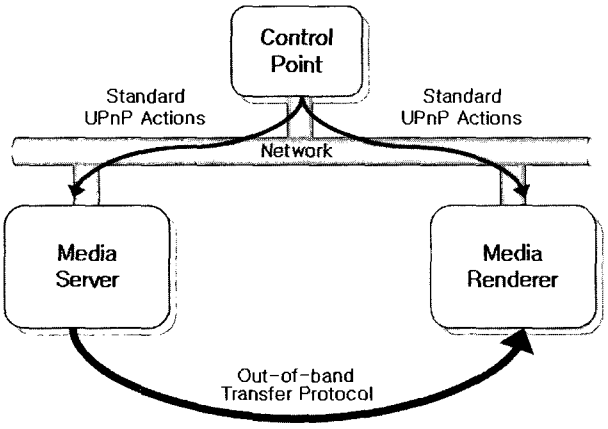


그림 4. UPnP AV 아키텍처
Fig. 4. UPnP AV architecture.

표 3. 메시지 타입과 UPnP action과의 mapping
Table 3. Mapping between message types and UPnP actions.

메시지 타입	설명	예	UPnP action
Simple message	지연시간 조건이 별로 없는 작은 메시지	일반적인 service invocation	Control
Realtime message	시간 제한조건이 있는 작은 메시지	제어 신호 invocation	Control*
Message stream	일정 주기를 가진 메시지 sequence	주기적인 센서 데이터	Eventing
Isochronous	지속적인 media type의 데이터	카메라 데이터, 음성 데이터	UPnP A/V style action
Burst	대용량 데이터	지도	UPnP A/V style action

UPnP AV Control Point의 개입없이 media stream을 전송한다. 이와 같은 방식으로 Isochronous와 Burst 데이터는 데이터 source(Media server)에서 데이터를 사용하는 측(Media renderer)으로 전송될 수 있을 것이다. 추가적으로 이 경우에 그림 3(a)에 표시된 high speed 네트웍을 사용할 수도 있을 것이다. 표 3은 지금까지 설명한 메시지 대응 관계를 나타내고 있다.

실시간성(Realtime) - UPnP 프로토콜에서는 시간 제한조건을 맞추기 위한 방법이 없기 때문에 실시간 메시지를 UPnP로 전달하려고 할 때는 문제가 생길 수 있다. 하지만, 이러한 문제는 소프트웨어 구조를 큰 기능별로 나누는 coarse grained SOA를 사용하게 되면 많은 부분이 해결이 가능하다. 즉, 소프트웨어 컴포넌트의 기능을 큰 기능별로 나누면 대부분의 시간 제한이 있는 실시간 처리는 하나의 컴포넌트 안에서 해결하도록 할 수 있기 때문에 굳이 UPnP를 실시간 메시지 전송이 사용할 필요가 없게 된다. 예를 들어 실시간 처리가 필요한 부분으로 모터 제어 신호가 있는데, 우리가 모터 제어 컴포넌트를 만들어 여기서 긴급제어(emergency control)을 포함한 low level 제어를 모두 책임지도록 하면, 실시간 메시지의 필요성은 현저히 줄게 된다. 또한 UPnP 서비스 내부적으로 실시간 서비스를 구현하는 것도 생각해 볼 수 있는데, 이를 위해서는 실시간 OS의 기능을 이용하여 실시간 UPnP 서비스를 만들 수 있다. UPnP는 Linux 위에 구현될 수 있지만, Linux는 실시간

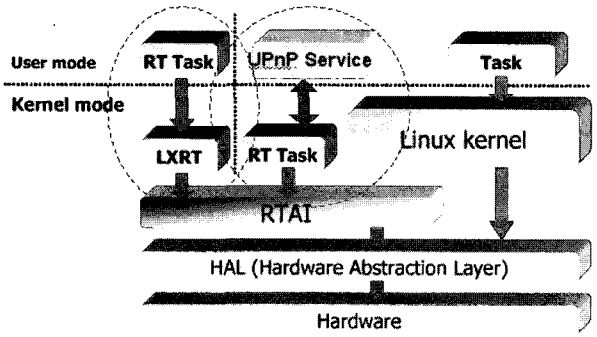


그림 5. 실시간 UPnP 서비스 구현을 위한 2가지 방법
Fig. 5. Two implementation methods for realtime UPnP service.

처리를 보장하지 못한다. 따라서 Linux와 같이 사용하는 실시간 OS의 기능을 이용하는 방법이 필요하다. 만일 우리가 RTAI[12]와 같은 실시간 OS를 사용한다면 이를 활용하는 방법은 2가지가 있다. 그림 5는 이를 나타내고 있다. 하나는 RTAI의 LXRT 모듈을 활용하는 것이다. 이는 RTAI의 기능을 이용하여 user mode에서 실시간 task를 돌릴 수 있도록 해주는 모듈이다. 하지만, 이 모듈이 모든 경우에 정확한 시간을 보장해주지는 못한다. 그림 5의 왼쪽 동그라미 표시가 이를 나타낸다. 두 번째 방법은 실시간 task의 실시간 처리 부분을 RTAI task로 구현하고 이를 user mode의 UPnP 부분과 연동하게 만드는 방법이다. 이렇게 하면 외부에서는 이 컴포넌트를 UPnP 컴포넌트로 인식하게 되고, 내부적으로는 실시간 처리가 가능하게 된다. 그림 5의 오른쪽 동그라미가 이를 나타낸다.

우선순위(priority) - 실시간성과 밀접히 연관되어 있는 것으로 우선순위가 있다. UPnP가 로봇 미들웨어로 사용되기 위해서는, 로봇이라는 특성상 전달되는 제어 메시지 간의 우선순위가 필요하게 되며, 우선순위가 높은 중요한 메시지가 덜 중요한 메시지보다 먼저 전달되도록 해야 한다. 예를 들어 주행하라는 명령과 앞에 사람이 있으니 멈추라는 명령이 있으면 나중 것이 더 우선되어야 할 것이다. 그러나 UPnP에서는 이렇게 전달되는 메시지의 우선순위를 주고, 또 이를 관리하는 부분이 없다. 그러므로 UPnP 내에 메시지의 우선순위의 부여와 관리 기능이 구현될 필요가 있다.

Network 선택 - 실시간 서비스와 같은 빠른 메시지 전송과 대용량 데이터는 고속 네트워을 필요로 한다. 만일 그림 3(a)에 나타낸 샘플 로봇처럼 로봇 하드웨어에서 고속 네트워이 지원되는 경우에, UPnP가 서비스 자체나 사용자가 메시지나 데이터 타입에 맞게 네트워

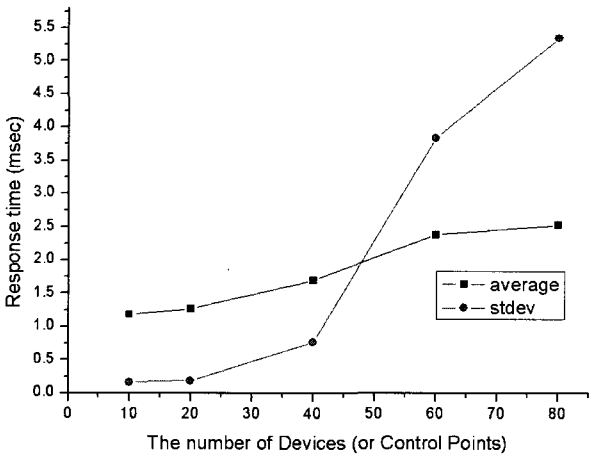


그림 6. Device 개수에 따른 response time의 변화
Fig. 6. Change of response time according to the number of Devices.

을 선택할 수 있는 기능을 제공하면 좋을 것이다. 만일 이러한 기능이 제공되면 앞서 분류한 메시지 타입들 중 Ishochronous, Burst, 실시간 메시지 등이 IEEE1394와 같은 고속 네트웍을 사용하게 될 것이다.

성능(performance) - UPnP의 Device는 하드웨어적인 device와 일대일 대응되는 것은 아니다. 하나의 SBC에 여러 개의 logical UPnP Device가 존재할 수 있는 것이다. 이러한 Device와 Control Point의 개수가 증가하게 되면 시스템의 성능은 감소하게 될 텐데, 이러한 성능 감소가 예측 가능하다면 시스템을 효율적으로 구성할 수 있을 것이다. 본 연구에서는 UPnP의 성능을 측정하고 성능의 변화 양상을 보기 위하여 UPnP Device와 Control Point들을 구현하여 response time을 측정하였다. UPnP Device와 Control Point 구현은 Intel의 UPnP SDK를 사용하였고, Device와 Control Point의 개수는 10개부터 80개 까지 증가시키면서 실험을 하였다. UPnP Device와 Control Point는 Pentium 4, 1.7GHz CPU에 256MB 메모리를 가진 하나의 SBC에 위치시켰다. OS는 Redhat 9.0이었고, 커널은 2.4.26으로 upgrade시켰다. 그림 6은 실험의 결과를 보여주고 있다. 그림에서 볼 수 있듯이 Device의 개수가 증가하면서 평균 response time이 증가하기는 했지만 3msec 이내로 많이 변화하지 않았다. 표준편차는 60-80개 사이에서 조금 급격한 변화를 보여주었다. 하지만, response time은 그리 크지 않았고, 이 그래프로부터 어느 정도 예측이 가능한 것으로 판단된다.

Device와 Control Point의 결합 - 한 소프트웨어 컴포넌트는 다른 여러 컴포넌트들을 사용하여 기능을

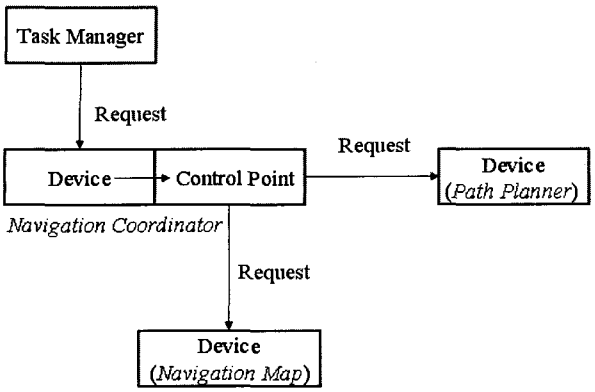


그림 7. UPnP Device와 Control Point의 결합
Fig. 7. Composition of UPnP Device and Control Point.

수행하면서, 자신 또한 다른 컴포넌트에 의해 사용될 수 있도록 구성될 수 있다. 이러한 구성을 UPnP의 Device와 Control Point의 각 한가지만으로는 표현할 수 없다. 그러므로 UPnP로 소프트웨어 컴포넌트를 표현함에 있어, 여러 Device들의 서비스를 사용하는 Control Point임과 동시에 자신 또한 서비스를 제공하는 Device로 동작할 수 있는 복합적인 형태가 필요하다. 예를 들어 주행 프로그램을 몇 개의 컴포넌트로 구성한다고 할 때 그림 7과 같은 구성이 나올 수 있을 것이다. 여기서 Navigation Coordinator 컴포넌트는 Path planner와 Navigation map 컴포넌트에 대해서는 Control Point이면서도 자기보다 상위에 있는 Task Manager에 대해서는 Device의 역할을 해야 하는 것이다.

메모리 Footprint - 표 1에서 볼 수 있듯이 UPnP의 메모리 용량(footprint)는 작다. 우리의 경험 상, 기본적인 UPnP Device와 Control Point의 크기는 100KB 이하이다. 따라서, UPnP는 다른 미들웨어에 비해서 메모리 용량에서 장점을 가지고 있다.

배치(Deployment) - UPnP는 기본적으로 dynamic 분산환경을 대상으로 개발되었기 때문에 Device와 Control Point 들이 쉽게 네트웍에 연결되었다 나갈 수 있다. 이는 소프트웨어 컴포넌트가 dynamic하게 설치되었다 제거될 수 있음을 의미한다. 또한, 나아가 SBC 자체가 로봇이 작동하는 중간에도 연결되었다 제거될 수 있음을 의미한다. 따라서, UPnP는 로봇의 부품을 교환하거나, 잘못된 부분을 고치는 데 있어 아주 좋은 장점을 가지고 있다.

Ⅳ. 토론 및 결론

본 논문에서는 UPnP를 로봇 미들웨어로 사용하는 것을 제안하였으며, UPnP를 로봇 미들웨어로 사용할 때 대두될 수 있는 소프트웨어 아키텍처, 메시지 mapping, 실시간성, 우선순위, 네트웍 선택, 성능, Device와 Control Point의 결합, 메모리 용량(footprint), 배치 등 여러 가지 요소들을 실험과 함께 검토하였다. 이러한 검토는 하나의 샘플 로봇을 대상으로 하였지만, 가능한 한 일반적인 관점에서 검토를 하였기에 이는 일반화 될 수 있다고 생각된다.

UPnP는 TAO CORBA에 비해서 많은 장점을 가지고 있다. UPnP는 Service Oriented Architecture(SOA)를 기반으로 하는데, 이는 coarse grained 프로그래밍 모델을 가지고 있으며 dynamic한 분산환경에 적합한 구조이다. 따라서, UPnP는 소프트웨어 컴포넌트들이 많이 요구되는 지능형 로봇의 개발에 적합하며 dynamic한 분산환경이 요구되는 미래 로봇 미들웨어의 중요한 후보 중에 하나이다. 비록 UPnP를 로봇 미들웨어로 사용하기 위해서는 몇 가지 점에서 풀어야 할 구현상의 문제가 있지만 UPnP는 로봇 미들웨어로 사용하기에 적합하다고 생각된다.

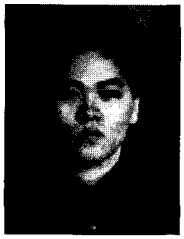
참 고 문 헌

[1] H. Hirukawa, F. Kanehiro and S. Kajita, "OpenHRP: Open Architecture Humanoid Robotics Platform," Int'l Symp. Robotics Research, 2001.
[2] CORBA, www.omg.org
[3] E.Li, D. Chen, H. I. Christensen and A. Oreback, "An Architecture for Indoor Navigation," Int'l conf. on Robotics and Automation, vol. 2, pp. 1783-1788, April, 2004.
[4] H. Utz, S. Sablatnog, S. Enderle and G. Kraetzschmar, "Miro Middleware for Mobile Robot Applications," IEEE Tr. on. Robotics and Automation, vol. 18, no. 4, pp. 493-497. August, 2002.
[5] TAO, www.cs.wustl.edu/~schmidt/TAO.html
[6] Jini, www.jini.org
[7] .net, www.microsoft.com/net
[8] UPnP, www.upnp.org
[9] S. Ilango Kumaran, Jini Technology: An Overview, Prentice Hall, p.307, 2002
[10] ACE, www.cs.wustl.edu/~schmidt/ACE.html

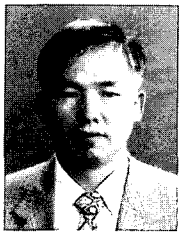
[11] DLNA, www.dlna.org
[12] RTAI, Realtime Application Interface,
www.aero.polimi.it/~rtai

저 자 소 개

안 상 철(정회원)
1988년 서울대학교 제어계측 공학과 학사 졸업.
1990년 서울대학교 제어계측 공학과 석사 졸업.
1996년 서울대학교 제어계측 공학과 박사 졸업.
1996년~1997년 미국 USC 방문연구원.
1997년~현재 한국과학기술연구원(KIST) 영상미디어연구센터 책임연구원
<주관심분야 : Mixed Reality, IBMR, Vision 기반 HCI, 가상현실, 로봇>



이 정 우(학생회원)
2005년 한양대학교 전자전기 컴퓨터공학부 학사 졸업.
2005년~현재 한양대학교 전자통신컴퓨터공학과 석사 재학
<주관심분야 : UPnP, Sensor Network, Image Processing, HCI, 로봇>



김 형 곤(정회원)
1974년 한국항공대학교 항공전자 공학과 학사 졸업.
1982년 Univ. of Kent (England) U.K. 전자공학과 석사졸업
1985년 Univ. of Kent (England) U.K. 전자공학과 박사졸업
1993년~1994년 호주 Univ. of South Australia 초청연구원.
1977년~현재 한국과학기술연구원(KIST) 책임연구원.
<주관심분야 : 영상처리용 VLSI 구조, 센서 fusion, 대화형 영상처리, 스테레오 비전 시스템, 가상공간 Interface, 영상-그래픽스 합성, MPEG-4>