지능형 서비스 로봇을 위한 유비쿼터스 환경과 제어구조

Control Architecture and Ubiquitous Environment for Intelligent Service Robots

지능형 로봇 특집

이승익 (S.I. Lee) 로봇소프트웨어아키텍처연구팀 선임연구원
서범수 (B.S. Seo) 로봇소프트웨어아키텍처연구팀 선임연구원
김중배 (J.B. Kim) 로봇소프트웨어아키텍처연구팀 팀장

목 차

본 고에서는 지능형 서비스 로봇에 적용되고 있는 기존의 로봇 제어구조에 대한 기술 동향을 살펴보고, 가정환경에서 로봇이 적절한 정보를 획득하고, 이를 환경 내의 다른 장치나 가전도구와 연동할 수 있도록 하기 위한 환경 구축 방안과 로봇 내부의 제어구조 및 다양한 서비스를 제공하기 위한 서버와의 통신 문제 등에 대하여 살펴본다. 로봇이 가정 내에서 다양한 서비스를 제공하기 위해서는 가정 환경과의 적절한 상호작용은 필수적이다. 이를 위해서는 가정의 다양한 기기들과의 적절한 정보 교환이 필요하고, 이러한 정보를 효율적으로 이용할 수 있는 환경의 구축이 필수적이다. 또한, 로봇 내부적으로도 다양한 서비스를 제공하기 위한 적절한 제어구조가 필요하며, 경우에 따라서는 로봇 외부의 서버와의 협업을 통한 질높은 서비스의 제공이 지능형 서비스 로봇의 성공에 필수적인 조건이라 할 수 있다. 본 고에서 제안된 로봇 제어구조는 2006년 10월부터 시작된 URC 로봇 시범사업에 적용되어 현재 시범운용중에 있다.

Ⅰ. 서론

URC[1]는 “언제 어디서나, 나와 함께 하며, 나에게 필요한 서비스를 제공하는 로봇”으로 간단하게 정의할 수 있다. 일견 단순해 보이는 이 정의는 많은 내용을 함축하고 있지만 가장 중요한 것은 URC 로봇이 유선 혹은 무선 통신이 가능한 각종 외부 디지털 디바이스와 상호 연동할 수 있어야 한다는 것이다. 이를 통하여 환경 인식이나 음성 인식 등과 같이 로봇이 수행해야 할 기능을 외부 디바이스에 분담시킴으로써 로봇의 하드웨어 구성을 단순화시켜 제작 원가를 절감하고, 날씨나 교통 정보와 같이 로봇 단독으로는 획득할 수 없는 정보 등을 이용하여 보다 다양한 서비스를 할 수 있다는 점이다. 즉, URC 로봇을 기존 서비스 로봇보다 향상된 서비스를 지원하면서도 더 싸게 만들 수 있게 된다.

가정용 서비스 로봇은 인간에게 각종의 편리한 서비스를 제공하기 위한 로봇이다. 이 로봇이 제공하는 기능으로는 청소 서비스, 홈 모니터링, 교육, 날씨예보와 같은 서비스를 제공한다. 기존의 독립형 로봇과는 다르게 유비쿼터스 환경에서의 서비스 로봇은 네트워크 및 통신기능을 최대한 활용하여 기존의 독립형 로봇으로는 제공할 수 없었던 각종 서비스를 제공할 수 있다는 장점이 있다.

이러한 가정용 서비스 로봇을 설계하는 데 있어서는 많은 도전적 과제들이 우리 앞에 대기하고 있다. 우선은, 로봇이 가정 내에서 다양한 서비스를 제공하기 위해서는 가정 환경과의 적절한 상호작용은 필수적이다. 이를 위해서는 가정의 다양한 기기들과의 적절한 정보 교환이 필요하고, 이러한 정보를 효율적으로 이용할 수 있는 환경의 구축이 필수적이다. 또한, 로봇 내부적으로도 다양한 서비스를 제공하기 위한 적절한 제어구조가 필요하며, 경우에 따라서는 로봇 외부의 서버와의 협업을 통한 질 높은 서비스의 제공이 지능형 서비스 로봇의 성공에 필수적인 조건이라 할 수 있다.

본 고에서는 지능형 서비스 로봇이 가정환경에서 로봇이 적절한 정보를 획득하고, 이를 환경 내의 다른 장치나 가전도구와 연동할 수 있도록 하기 위한 환경과 로봇 내부의 제어구조 및 다양한 서비스를 제공하기 위한 서버와의 통신 문제 등에 대하여 살펴본다.

Ⅱ. 유비쿼터스 환경 구축

전통적인 자율 로봇 분야에서는 로봇과 로봇을 둘러싼 환경을 서로 독립적인 것으로 간주한다. 로봇을 둘러싼 환경은 정해지지 않았으며 단순히 관측 가능한 개체로 여김으로써 로봇이 가진 불확실한 센서나 불완전한 액추에이터를 통해 환경에 적응한다.

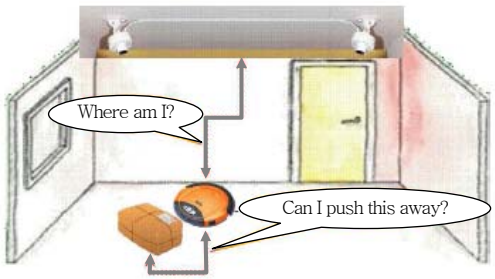
본 논문에서는 로봇과 환경을 하나의 동일한 시스템으로 간주하고 이들이 센서나 액추에이터, 응용 시스템, 액티브 태그 시스템, 혹은 전통적인 모바일 로봇과 같은 형태로 분산되어 존재함을 가정하고, 이러한 물리적 디바이스들이 서로 통신하고 협력함으로써 사람에게 정보를 제공하거나 특정 작업을 수행하는 유기적 네트워크인 PEIS-Ecology를 제안한다.

예를 들어, 우유팩을 잡고자 하는 로봇의 예를 들어보자. PEIS-Ecology에서는 로봇이 우유팩의 정보(크기, 색상 등)를 얻기 위해 반드시 카메라를 이용할 필요가 없다. 대신, 우유팩 자체가 자신의 정보를 IC 태그 안에 가지고 있다면 이를 로봇에게 전달함으로써 이를 대신할 수 있다.

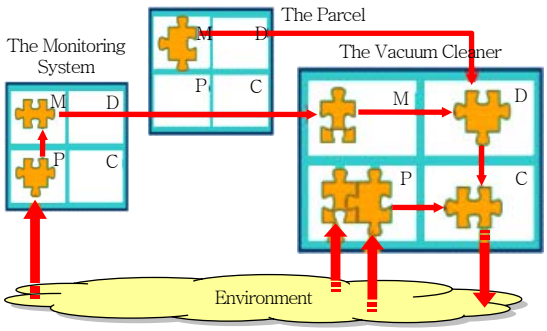
PEIS는 가정이나 사무실에 존재하는 임베디드 시스템을 지칭한다. PEIS 컴포넌트는 PEIS에 존재하는 단위 컴포넌트로써 입력과 출력 포트를 가진 센서나 액추에이터를 가지고 상호 연결될 수 있다. 응용 프로그램 역시 하나의 컴포넌트로써 존재가 가능하다. 예를 들면, 센서 PEIS 컴포넌트로부터 정해진 포트를 통해 정보를 받아 상위 수준의 경로 계획이나 의사 결정을 수행할 수 있는 숙고형 PEIS 컴포넌트가 존재할 수 있으며, 이 숙고형 PEIS는 의사결정에 필요한 외부 세계 모델링 정보를 모델링 PEIS 컴포넌트로부터 입력 받아 실제 액추에이터를 가진 제어 PEIS에게 물리적 모터를 움직이도록 일

련의 명령을 내린다. 이와 같은 모델링(M), 인식(P), 제어(C), 숙고(D) PEIS 컴포넌트 외에도 사람과 로봇이 상호 작용하기 위한 컴포넌트나 GUI와 같은 컴포넌트 역시 PEIS가 될 수 있다.

(그림 1)은 RFID를 가진 박스와 천정에 웹 캠을 가진 모니터링 시스템, 그리고 자율 청소 기능을 가진 청소 로봇으로 구성된 PEIS-Ecology의 예이다. (그림 2)는 (그림 1)의 예를 컴포넌트간의 상호 연결 관계를 고려하여 표현한 것이다. 박스의 경우 자신에 대한 정보만을 가지고 있는 M PEIS만으로 구성되어 있고, 웹 캠을 가진 모니터링 시스템은 방에 대한 정보를 가진 M PEIS와 웹 캠으로부터 화상 정보를 받아 들이는 P PEIS로 구성된다. 청소 로봇의 경우 4가지 PEIS 컴포넌트를 모두 가지고 있다. 로봇은 자신의 위치 정보를 자신이 가진 오도미터리 컴포넌트로부터 얻을 수도 있으며 모니터링 시스템에서 얻을 수도 있다. 또한, 청소 중 만난 박스를 밀어야 할지 말아야 할지를 판단하기 위한 정보를 박스로부터 직접 수집함으로써 청소를 수행한다.



(그림 1) 단순한 PEIS-Ecology의 예



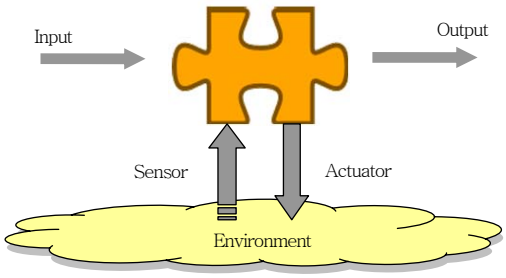
(그림 2) PEIS 컴포넌트의 연결 관계

◎ PEIS-Ecology의 구성요소

PEIS-Ecology를 구현하기 위해서는 각 컴포넌트의 모델과 이들 간의 통신 방법, 그리고 이를 위한 미들웨어 등의 기능들이 필요하다. 또한 분산되어 있는 다양한 PEIS 컴포넌트들을 정적 혹은 동적인 방법으로 상호 연결시키기 위한 기술들이 필요하다.

• PEIS 컴포넌트 모델

PEIS 컴포넌트는 입력과 출력을 가지며, 센서 입력과 액추에이터 출력을 가지는 일종의 논리적 컴포넌트이다. PEIS 컴포넌트는 앞서 소개한 P, M, C, D 기능 블록뿐 아니라 특정 기능의 응용이나 전체 시스템 자체도 하나의 컴포넌트가 될 수 있다. (그림 3)은 이러한 PEIS 컴포넌트 모델의 개념을 나타낸다.



(그림 3) PEIS 컴포넌트 모델

• 통신 모델

PEIS-Ecology에서는 한 기능을 제공하는 다양한 컴포넌트들이 수시로 ecology에 들어오거나 나갈 수 있다. 즉, (그림 3)에서 모니터링 시스템이 오류를 일으켰다면 청소기는 자신의 위치 정보를 자신이 가진 오도미터리 컴포넌트로부터 얻어 계속 청소를 수행할 수 있다. 이러한, 동적 환경을 위해서 PEIS-Ecology는 피어투피어(peer-to-peer)에 기반한 분산 튜플스페이스(tuple-space)[2],[3] 아키텍처를 사용한다. 상호 교환되는 튜플은 다음과 같은 데이터 형태를 가진다.

<piesID, compID, key, val0. ⋯, valN>

peisID는 PEIS의 ID이며, compID는 해당 PEIS에 존재하는 컴포넌트 ID이다. Key는 해당 컴포넌트가 생성하는 데이터에 대한 문자열 형태의 key이며 이에 따른 값들이 전송된다. 실제 구현에서는 각 컴포넌트들은 자신이 관심 있는 데이터에 대해서 key를 이용하여 PEIS 내에 특정 컴포넌트 혹은 모든 컴포넌트에게 등록하여 데이터 생성과 함께 전송받는다.

• PEIS 구성

동일 기능을 가진 다양한 컴포넌트들이 PEIS-Ecology에는 존재한다. 따라서, 이러한 컴포넌트들을 연결하여 주어진 작업을 수행한다. 컴포넌트들을 연결하는 방법은 정적인 방법과 동적인 방법이 가능하다[4],[5]. 이를 위해서 각 컴포넌트들이 제공하는 기능과 매개변수 등을 표현하는 방법이 요구되며, 주어진 작업과 현재 ecology의 환경에 따라 이들을 연결시킬 수 있는 방법에 대한 연구가 필요하다.

• PEIS 커널

PEIS 컴포넌트의 요청을 받아 다른 PEIS 컴포넌트에게 튜플 데이터 형태로 전달하거나 원하는 데이터를 수신하는 작업들을 수행하는 일종의 미들웨어 기능을 수행하는 것이 PEIS 커널이다[6]. PEIS 커널은 PEIS-Ecology에 존재하는 컴포넌트에 대한 탐색과 메시지 라우팅을 수행하며, P2P에 기반한 데이터 전송을 담당한다. (그림 4)는 이러한 PEIS 커널의 계층별 기능을 보여 준다.

PeisKernel API	get/set tuple, subscribe/unsubscribe register/cancel callback
Service layer	store/retrieve/propagate tuples subscriptions, callback hooks, ...
Tuple-space layer	create/send/receive/route packages topology management, time sync, ...
Communication layer	TCP, UDP, serial links, ... network discovery, ...

(그림 4) PEIS 커널의 계층도

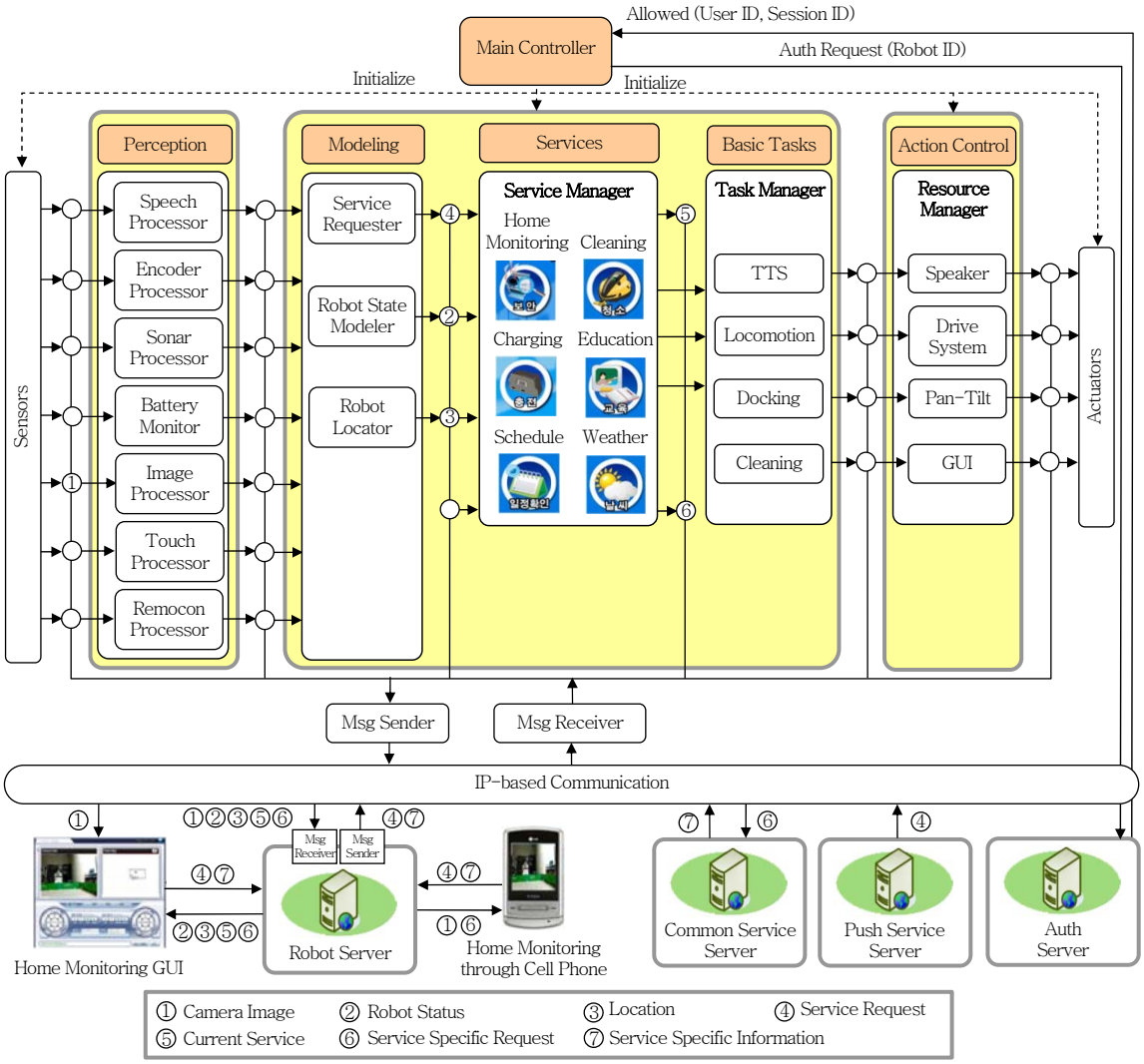
Ⅲ. 홈 서비스 로봇 제어구조

지능형 가정용 서비스 로봇을 위한 제어구조는 (그림 5)와 같다. 그림에서 보이는 여러 서브 시스템이 존재하며, 그 중에서 로봇 내부, 로봇 서버, 그리고 이들간의 통신을 위한 프로토콜이 가장 중요한 서브 시스템으로 여겨진다.

(그림 5)에서 상위부분에 나타나 있는 것이 로봇의 내부 시스템의 구조이다. 로봇은 사람과 상호작용하며, 사람이 원하는 서비스를 제공해주는 개체이다. 이를 구성하는 내부 시스템은 인식(perception), 모델링(modeling), 서비스, 기본 태스크, 그리고 행동 제어(action control) 모듈 등으로 구성된다. 인식모듈은 외부와의 상호작용을 통하여 일차적인 정보를 획득하는 기능을 수행한다. 이를 통하여 로봇은 내부적으로 필요한 정보를 추출한다. 예를 들어, 입력된 영상으로부터 사람의 눈 부분을 추출한다든가 하는 기능을 수행하게 된다. 모델링 모듈은 로봇과 환경의 상태를 모델링하는 기능을 수행한다. 이는 인식모듈에서 올라온 정보를 바탕으로 하지만, 보다 장시간 유지되는 정보를 대상으로 한다. 예를 들면, 사용자의 현재 위치를 모델링할 때, 인식 모듈을 통하여 사용자의 위치가 입력되는 경우도 있지만, 카메라의 각도 등의 문제로 인하여 순간적으로 사용자의 위치가 인식모듈을 통하여 입력이 되지 않을 수도 있다. 하지만, 그렇다고 하더라도 방금 전까지 있었던 사용자가 사라진 것이 아니라 잠시 시야에서 벗어난 것이므로 모델링 모듈에서는 이를 감안하여 사용자의 현재 위치를 추적한다.

행동 제어 모듈에서는 로봇이 지니고 있는 액추에이터 모듈에 실행 명령을 전달한다. 이 실행 명령은 로봇이 지니고 있는 서비스와 이 서비스가 이용하고 있는 기본 태스크로부터 행동 제어 모듈에 전달된다.

이 로봇은 다양한 종류의 서버와 협업을 하게 된다. 로봇 서버(robot server)는 사용자 정보 및 서비스를 관리한다. 공통 서비스 서버(common service server)는 이 서버에 연결된 모든 로봇을 대상으로



(그림 5) 로봇 내부 제어구조

로봇이 요청하는 서비스의 콘텐트를 전달하는 역할을 수행한다. 푸시 서비스 서버(push service server)는 사용자의 일정관리 서비스에서 지정된 시간에 알림 기능을 수행할 때, 지정된 시간을 체크하여 해당 시간이 되면 이를 로봇에게 알린다. 로봇은 이 알림을 받게 되면, 해당 서비스를 실행하여 사용자에게 주어진 시간의 일정을 알리게 된다. 인증서버(auth server)는 로봇이 로봇 서버 및 서비스 서버들에 접근하는 데 필요한 인증을 담당하게 된다.

로봇과 서버간에 정보를 주고 받기 위해서는 이에 적절한 전송 방송이 고안되어야 한다. 이러한 정보에는 동작 제어 메시지, 서비스 제어 메시지, 비디오 메시지, 로봇 상태 메시지, 그리고 콘텐트 패킷 메시지 등이 있다. 본 고에서는 이를 만족시키기 위하여 설계된 프로토콜을 제안한다.

1. 가정용 서비스 로봇

가. 개요

본 고에서 대상으로 하는 로봇은 가정용 서비스 로봇이다. 기존의 청소로봇이 단순히 청소만을 수행하는데 비해 본 논문에서 제안하고 있는 서비스 로

(그림 6) 가정용 서비스 로봇

봇은 인터넷이라는 통신매체와 결합하여 청소서비스뿐만 아니라 홈 모니터링, 뉴스, 날씨, 교육, 사진찍기, 영상메시지 등 다양한 정보 서비스를 제공할 수 있다. (그림 6)은 가정용 서비스 로봇을 보여준다.

로봇을 제어하기 위한 다양한 제어 아키텍처에 대한 모델이 제시되어 있지만, 정보 콘텐트 서비스를 포함한 제어 아키텍처에 대한 모델은 많지 않다. 본 논문에서 제안하고 있는 로봇 제어 아키텍처는 기존의 제어 아키텍처에 정보 콘텐트를 서비스하기 위한 서비스 제어 아키텍처를 포함한다.

이러한 정보 콘텐트 서비스를 위해서는 로봇의 이동과 관련된 로봇의 모션제어뿐만 아니라 정보 콘텐트 서비스를 제공함에 있어 다양한 입력매체로부터의 사용자 명령과 복합적으로 실행 요청되는 서비스들에 대한 조정, 로봇 자원의 충돌 방지 및 효율적 분배 등이 요구된다.

나. 로봇 내부 제어구조

로봇서브시스템은 청소와 같은 로봇 자체의 서비스뿐만 아니라 외부의 콘텐츠 서버로 접속하여 다양한 서비스를 제공할 수 있도록 설계되었으며, 다음과 같은 모듈로 구성된다.

- 리모컨, 로봇 터치스크린, 전용 클라이언트, 통지서버를 통한 이벤트 등 다양한 매체를 통해 입력되는 사용자 명령에 대한 처리를 하는 서비스 요청자(service requester)
- 서비스의 실행과 우선순위 조정 등을 처리하는 서비스 관리자(service manager)
- 로봇의 자원관리 및 모션제어를 위한 행동 제어 모듈(action control)
- 공통서비스와의 연계를 위한 공통서비스 연계 모듈
- 일정 알림과 같이 통지서버를 통해 발생되는 통지서비스 요청처리를 위한 푸시서비스 연계모듈

로봇서브시스템의 실행과정은 다음과 같다. 먼저 로봇이 구동과정을 마치면 인증서버를 통하여 인증을 받고 특화서비스를 위해 로봇제어서버로 접속하여 인증된 세션정보를 전달한다. 로봇제어서버와의 접속 후에는 로봇제어서버와의 통신상태를 유지하기 위해 로봇의 센서정보 등을 주기적으로 주고 받는다. 로봇사용자는 로봇의 터치스크린, 리모컨을 이용하여 서비스를 요청하면 로봇서브시스템의 명령해석기를 통해 서비스 관리/조정자로 전달된다. 서비스 관리/조정자는 서비스 객체를 생성하고 실행한다. 이때 이미 서비스중이라면 우선순위를 기반으로 실행순서를 결정한다. 실행되는 서비스가 특화서비스인 경우 서비스 객체를 통해 로봇제어서버로 요구되는 데이터 및 서비스를 요청한다. 로봇제어서버는 로봇으로부터 요청된 데이터 및 서비스를 처리한 후 로봇서브시스템으로 전송한다. 로봇서브시스템은 로봇제어서버로부터 받은 콘텐트 정보를 이용하여 로봇의 터치스크린, 스피커 등으로 해당 콘텐트를 출력하고 로봇 구동 모듈을 통해서 로봇의 모션을 제어한다. 공통서비스의 경우는 특화서비스와는 달리 콘텐트 제공자에 의해 배포되는 서비스로써 로봇서브시스템과는 별도의 프로세스로 동작된다. 서비스 관리/조정자에 의해 공통서비스가 요청되면 서비스 관리/조정자에 의해 관리되는 서비스 객체를 통해 별도의 프로세스로 실행된다. 특화 서비스와 마찬가지로 이미 서비스중이라면 우선순위를 기반으로 실행순서가 조정된다. 좀 더 자세한 처리과정은 별도의 절을 통해서 논하기로 한다.

서비스의 실행요청은 로봇사용자의 명시적 요청뿐만 아니라 일정 알림 등과 같이 이벤트에 의해서 발생되기도 한다. 우리는 이를 통지서비스라 하며 로봇서브시스템의 푸시서비스 연계모듈에서 처리한다. 통지서비스의 대상은 뉴스, 날씨, 일정 알림, 모

닝콜과 같은 공통서비스를 기반으로 하고 있으며, 통지서버에 의해 요청되는 메시지에 따라 선택적으로 실행된다. 통지서비스에 대한 처리과정은 통지서버에 의해 이벤트가 발생되면 로봇서브시스템의 푸시서비스 연계모듈로 전송된다. 푸시서비스 연계모듈은 통지서비스에 대한 요청 메시지를 해석하여 서비스 관리/조정자로 통지서비스의 종류와 파라미터를 전달한다. 이후의 과정은 공통서비스의 실행과정과 동일하다.

로봇서브시스템은 서비스에 대한 요청처리뿐만 아니라 자원에 대한 효율적 분배를 제공하는데, 이러한 처리를 위해 리소스 관리자를 두고 있다. 특히 카메라 영상에 대한 효율적 활용을 위해서 영상 분배기를 두고 있다. 카메라 영상을 이용하는 서비스는 사용자 얼굴인식, 사진 찍기, 영상메시지, 홈 모니터링 등 다양하다. 네토로에 적용된 서비스를 예로 들면 PC 홈 모니터링, 휴대폰 홈 모니터링, 사진 찍기, 영상메시지가 있다. 특히 홈 모니터링서비스와 같이 복수의 사용자에 의해 단일의 로봇 카메라 영상정보를 공유할 필요가 있는데 로봇서브시스템에서는 리소스 관리자에 가상의 로봇 카메라를 두어 단일의 카메라 영상정보를 공유하도록 하였다. 로봇에 탑재된 단일의 카메라로부터의 입력되는 영상정보를 가상의 로봇 카메라를 통해 PC 홈 모니터링 및 휴대폰 홈 모니터링서비스에서 동시에 이용할 수 있다.

본 논문에서 제안하는 로봇 제어구조와 같이 로봇의 모션제어뿐만 아니라 정보콘텐츠를 제공하는 로봇제어시스템은 다양한 매체를 통한 사용자명령과 실행되는 서비스 순서 및 우선순위 조정 그리고 다양한 서비스 제작방법에 따른 실행 지원을 고려해야 한다. 본 논문의 로봇서브시스템도 다음과 같은 사항들을 주로 고려하여 설계하였다.

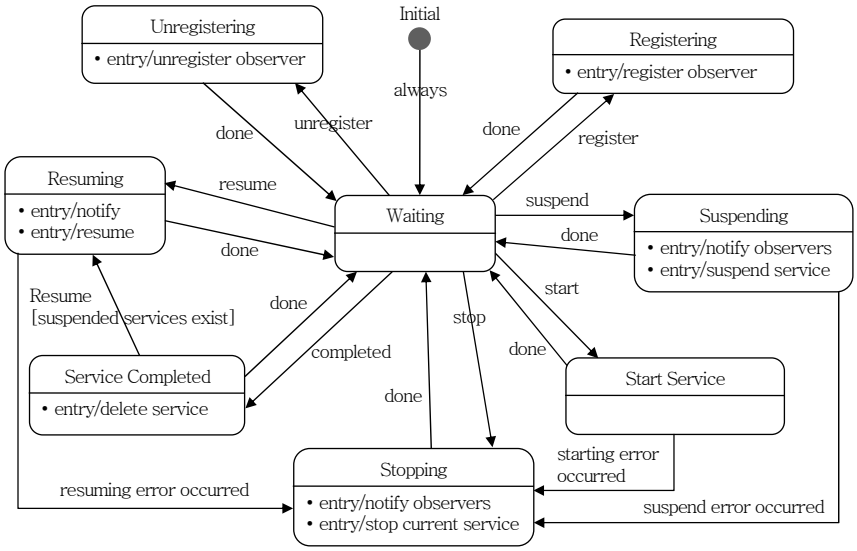
- 다양한 매체를 통해 입력되는 사용자 명령처리
- 실행되는 서비스 순서 및 우선순위 조정
- 다양한 서비스제작 방법의 지원
- 로봇 자원의 충돌 방지 및 효율적 분배

다. 다양한 서비스 요청 처리 방법

로봇으로의 서비스 요청은 로봇의 터치스크린, 리모컨, 외부의 전용클라이언트를 통한 서비스 실행요청 그리고 통지서버에 의한 이벤트 발생에 의해서 이루어진다. 이처럼 다양한 경로를 통한 서비스 요청을 처리하기 위해 로봇서브시스템은 사용자명령해석기를 두고 있다. 로봇의 터치스크린, 리모컨에 의한 명령은 사용자명령해석기로 바로 전달되고 다시 서비스 관리/조정자로 서비스 요청이 전달된다. 통지서버에 의해서 발생되는 이벤트는 이벤트를 수신하는 푸시서비스 연계모듈을 통해 해석되고 서비스 관리/조정자로 서비스 요청이 전달된다. 외부의 전용클라이언트를 통한 서비스 실행요청은 로봇제어서버를 통해 로봇의 통신관리자로 전송되고 다시 사용자명령해석기로 전달된다. 사용자명령해석기는 요청된 명령이 로봇 내부인지 로봇제어서버 또는 콘텐츠서버로부터 요청된 원격명령인지 또는 통지서버에 의해서 요청된 이벤트인지를 해석하여 서비스 관리/조정자로 요청된 서비스 종류와 서비스 요청자를 파라미터로 전달하고 서비스 관리/조정자는 서비스 요청에 대한 실행 결과를 파라미터로 전달받은 서비스 요청자에게 전송한다.

라. 서비스 조정 방법

로봇사용자는 필요에 따라 복수의 서비스를 동시에 실행하거나 특정 서비스를 우선하여 실행해주기를 원할 것이다. 예를 들어, 로봇이 청소를 하면서 뉴스나 날씨와 같은 정보를 알려주거나, 약속된 일정을 정해진 시간에 알려주기를 원할 것이다. 그리고 배터리 충전과 같은 충전서비스는 서비스에 관계없이 다른 서비스보다 우선하여 실행되어야 한다. 이처럼 상황에 따라 복잡하게 선택되고 실행되는 서비스들의 우선순위와 실행순서를 조정하여야 할 필요성이 요구된다. 로봇서브시스템에서는 각각의 서비스 특성에 따라 제공되는 서비스들에 대한 우선순위 테이블을 만들고 요청된 서비스들에 대해서 우선순위가 가장 높은 서비스가 먼저 실행되도록 하고



(그림 7) 서비스 조정자 상태도

있다. 그리고 복수의 서비스 요청에 대해서는 실행 서비스 테이블을 만들어 등록하고 우선순위에 따라 실행되도록 하고 있다. (그림 7)은 서비스 조정자에 대한 상태도이다.

마. 다양한 서비스 제작 방법 지원

앞서 설명한 로봇서브시스템은 특화서비스뿐만 아니라 공통서비스와 같이 로봇서브시스템에 특화되어 개발되지 않은 서비스를 지원하도록 설계되어 있으며 특화서비스는 로봇서브시스템 사용자 명령 해석기를 통해 처리되며 공통서비스의 경우는 공통서비스 연계모듈과 푸시서비스 연계모듈이 담당한다. 공통서비스로 분류되는 서비스들은 시스템에 특화되어 개발되어 있지 않기 때문에 로봇서브시스템과는 별도의 프로세스로 실행된다. 이런 경우 특화서비스와의 서비스 실행순서 및 우선순위를 조정할 필요가 요구되는데, 로봇서브시스템은 이를 위해 서비스를 실행하고 관리하기 위한 가상의 서비스 객체를 생성하여 관리하도록 하고 있다. 공통서비스에 대한 실행제어를 위해 로봇서브시스템은 공통서비스에 대응되는 가상의 서비스 객체를 이용한다.

또한 별도의 프로세스로 동작되는 공통서비스가 로봇의 모션을 제어하기 위해서는 로봇의 자원을 이용하여야 하는데 이때 특화서비스와의 자원충돌이 발생하게 된다. 이를 피하기 위해 로봇서브시스템은 공통서비스 연계모듈을 통해 약속된 프로토콜로 공통서비스와 통신하면서 로봇구동모듈의 리소스관리자를 통해 로봇의 자원을 공유하도록 하고 있다.

통지서비스와 같이 이벤트에 의해서 발생되는 서비스를 지원하기 위해 로봇서브시스템은 푸시서비스 연계모듈을 두어 통지서버와 약속된 프로토콜로 통신하면서 요청된 이벤트를 처리하도록 하였다. 통지서버에 의해 발생되는 이벤트들은 공통서비스로 분류되는 뉴스, 날씨, 일정 알림, 모닝콜에 해당된다. 공통서비스의 실행과정은 다음과 같다. 공통서비스가 별도의 프로세스로 실행되기 때문에 공통 서비스서버로 접속하고 데이터를 전송 받기 위해서는 세션정보를 필요로 한다. 이러한 세션정보는 공통서비스가 실행되면서 로봇서브시스템의 공통서비스 연계모듈과 접속하여 약속된 메시지로 세션정보를 전달 받게 된다. 공통서비스에서는 이 세션정보를 이용하여 공통서비스서버를 통해 콘텐트 데이터를 얻게 된다. 통지서비스의 경우는 통지서버로부터 이벤트에 의해 실행된다는 점을 제외하면 공통서비스와 그 처리과정이 동일하다.

2. 로봇과 서버의 통신

가. 개요

원격에서 제어 서버를 이용하여 네트워크 기반 정보 서비스 로봇을 제어하기 위해서는 동작 제어 메시지, 서비스 제어 메시지, 영상 메시지, 로봇 상태 메시지, 콘텐트 패킷 메시지 등 다양한 종류의 메시지를 혼용하여 사용하여야 한다. 본 논문에서는 이러한 메시지들의 송수신 방법을 응용 프로그램에게 투명하게 제공하기 위해 로봇과 제어 서버 양쪽에 메시지의 송수신을 담당하는 프로토콜 프록시를 두었다. 즉, 로봇 내부의 프로그램 및 제어 서버의 제어 프로그램은 마치 자신의 로컬에 있는 API를 호출하듯이 프로토콜 프록시의 API를 호출하면 프로토콜 프록시는 API에 해당하는 메시지를 만들어서 원격지에 있는 상대방의 프로토콜 프록시에게 전달한다. 요청을 전달 받은 프로토콜 프록시는 로봇이나 제어 서버의 자원에 접근하여 요청된 명령을 수행하고, 그 수행된 결과를 응답 메시지를 만들어 요청을 보낸 프로토콜 프록시에게 전송한다. 응답 메시지를 받은 프로토콜 프록시는 API를 호출한 프로그램에게 응답 메시지를 이용하여 호출 결과를 만들어서 리턴한다((그림 8) 참조).

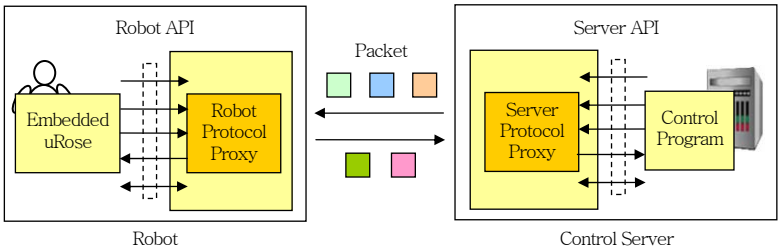
양쪽의 프로토콜 프록시 사이에서 통신을 하는데 있어서, CORBA[7]나 RMI 등과 같은 기존 request/response 모델 기반의 분산 클라이언트/서버 미들웨어를 사용하기에는 부족한 면이 있다. 왜냐하면, 네트워크 기반 정보 서비스 로봇에서는 다양한 형태의 서비스를 능동적으로 제공하기 위해서 request/response 방식뿐만 아니라 다음과 같이 다양한 형태의 통신 모델을 지원해야 하지만, 기존 미들웨어에서는 이러한 모든 통신 모델을 유연하게 지원하지 못한다.

- Oneway Upload: 사용자로부터의 명령 및 환경으로부터 획득한 센싱 정보들을 제어 서버로 전송
- Oneway Download: 로봇의 장치들을 피드백 없이 구동하기 위해 서버에서 단순 명령을 전송
- Request/Response: 서버에서 로봇에게 요청 메시지를 보내고 응답을 받는 통신 모델
- 역방향 Request/Response: 로봇에서 서버에게 필요한 데이터를 요청하기 위해 요청을 보내고 응답을 받는 통신 모델

또한, 기존의 request/response 모델의 미들웨어를 이용할 때에는 기계적 움직임과 같이 응답속도가 오래 걸리는 메시지의 경우, 요청을 보낸 제어서버가 동작의 실행결과를 로봇으로부터 돌려받을 때까지 대기하여야 하므로 다른 장치를 동시에 제어할 수 없게 되어 로봇 제어가 자연스럽지 않게 보일 수 있다. 이를 위해서 메시지마다 스레드를 생성하여 처리하게 하거나 여러 개의 커넥션을 이용하여 동시에 여러 장치를 제어하도록 할 수도 있지만, 이 경우 여러 스레드나 커넥션들에서 공통으로 사용되는 자원에 대해 이전 동작이 완료되지 않은 상황에서 새로운 동작 제어 메시지를 처리하려고 할 수 있어서 로봇 동작의 순차성을 해치게 되어 로봇을 비정상적으로 동작시킬 수 있다.

나. 순차성과 병행성을 지원하기 위한 프로토콜

이를 위해서 본 시스템의 프로토콜 프록시에서는



(그림 8) 프로토콜 구조

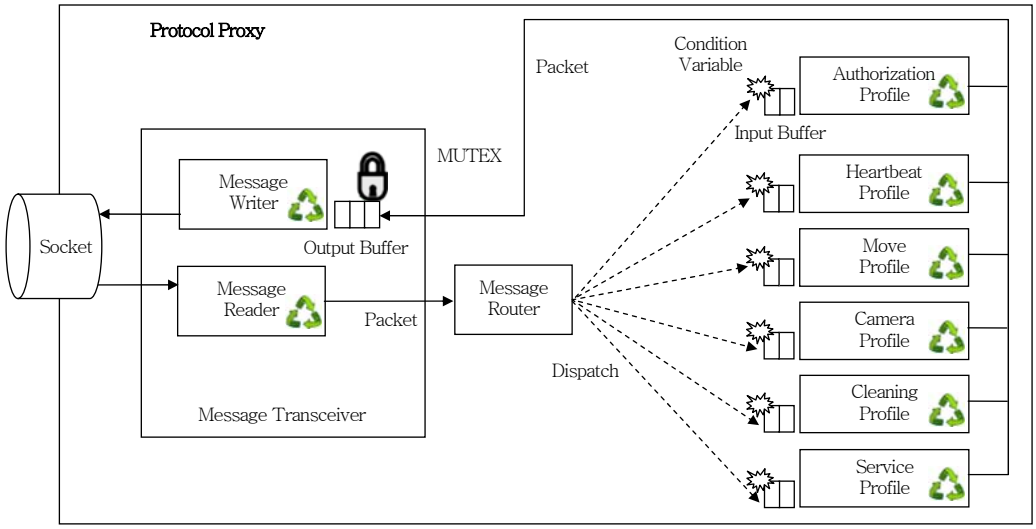
로봇 내부의 같은 자원을 사용하는 API들을 프로파일이라는 그룹으로 묶고, 각 프로파일마다 API 메시지를 처리하기 위한 스레드를 두고 있다((그림 9) 참조).

로봇 내부 프로그램이나 제어서버의 제어 프로그램도 여러 개의 스레드로 구성되어 있는데, 이런 스레드로부터 프로토콜 프록시의 여러 API가 동시에 호출될 수도 있다. 이때 요청을 보낸 쪽의 프로토콜 프록시는 이전에 보낸 요청에 대한 응답이 오지 않은 상태에서도 새로운 요청을 상대방의 프로토콜 프록시에 보내어 명령을 수행하도록 하여 로봇의 여러 장치를 동시에 제어할 수 있다. 또한 응답을 필요로 하지 않는 이벤트나 단방향 명령의 경우에는 다른 API의 실행중 여부와 상관없이 프로토콜 프록시의 API를 호출하여 이벤트 메시지나 명령 메시지를 상대방에게 전송한다.

프로토콜 프록시는 네트워크를 통해 들어온 메시지를 메시지 라우터를 통해 해당 프로파일로 배포하며, 프로파일 스레드는 조건변수를 통해 새로운 메시지의 전달 사실을 통보 받고 휴면상태에서 깨어나 자신에게 전달된 메시지를 처리한다. 프로파일은 자신이 수행할 메시지가 온 경우 로봇이나 제어 서버의 자원에 접근하여 요청된 명령을 수행하는데, 각 프로파일은 자신만의 독립적인 스레드를 갖고 있어서 다른 프로파일과 병행하여 동작하기 때문에 서로 충돌되지 않는 자원에 대해서 동시에 제어가 가능하게 되어 병행성을 제공할 수 있다. 반면에, 동일한 프로파일에 배포된 메시지는 프로파일의 큐에 넣어지고, 해당 프로파일 스레드는 더 이상 처리할 메시지가 없을 때까지 자신의 큐에 있는 메시지를 순차적으로 추출하여 해당 명령을 실행하여 명령의 순차성을 보장하고 있다.

Ⅳ. 결론

본 고에서는 유비쿼터스 환경 하에서 다양한 서비스를 제공할 수 있는 서비스 로봇을 위한 환경 구축 방안 및 로봇 내부의 제어구조에 대하여 살펴보았다. 지능형 로봇을 위한 유비쿼터스 환경을 구축하게 되면, 그 동안 로봇이 직면하고 있는 많은 문제를 보다 쉽게 해결할 수 있다는 장점이 있다. 예를 들면, 물체를 인식할 때 기존의 방법은 주로 로봇의 비전을 이용하여 물체 영상에 대한 영상처리를 수행하고 특징을 추출하여 해당 물체가 어떤 물체인지를 알아내는 방법을 사용하였다. 하지만, 이러한 방법은 인식률이나 속도 면에서 만족스러운 결과를 내지



(그림 9) 프로토콜 프록시 구조

못하고 있는 실정이다. 유비쿼터스 환경은 로봇과 주변 객체와의 상호 통신을 통하여 정보를 주고 받음으로 인하여 이러한 인식과정 없이 보다 손쉽게 주변을 인지할 수 있는 방법을 제공할 수 있다. 아직까지는 본 고에서 제안된 유비쿼터스 환경구축방안과 로봇이 유기적으로 결합되어 있지 않지만, 향후 이 둘 간의 결합을 통하여 로봇이 보다 많은 환경정보를 이용할 수 있을 것으로 기대된다.

또한, 본 고에서 제안한 로봇 제어 구조는 2006년 10월부터 전국 가정을 대상으로 수행중인 URC 로봇 시범사업 중의 하나인 한울 로보틱스의 네토로 로봇에 적용되어 운용되고 있다. 현재, 약 50대의 네토로 로봇이 전국 가정에서 실사용자를 대상으로 운용중에 있으며, 현재 약 50% 정도의 로봇 이용률을 보이고 있다. 시범사업을 통하여 제기된 주된 문제는 로봇의 크기가 청소용 로봇으로는 다소 크다는 문제점이 제기되었으나, 이는 로봇의 제어구조와 무관한 문제이다. 다만, 가정 환경이 가구별로 상이한 조건이므로 각 가정에서의 환경과의 불일치에 따른 소프트웨어적인 튜닝이 필요하다.

약 어 정 리

CORBA	Common Object Request Broker Architecture
ERSP	Evolution Robotics Software Platform
MIRO	MIddleware for RObot
ORCA	Open Robot Controller Architecture
OROCOS	Open RObot COntrol System
PEIS	Physically Embedded Intelligence System
RMI	Remote Method Invocation
TAO	The Ace Orb
URC	Ubiquitous Robotic Companion

참 고 문 헌

[1] S.R. Oh, "Network-based Intelligent Service Robot: Ubiquitous Robotic Companion," *Communications of the Korea Information Science Society*, Vol.23, No.2, Feb. 2005.

[2] A. Tanenbum and M. Steen, Distributed Systems - Principal and Paradigms, Prentice Hall, 2002.

[3] D. Gelernter, "Generative Communication in Linda," *ACM Transaction on Programming Languages and Systems*, Vol.7, No.1, 1985, pp.80-112.

[4] R. Lundh, L. Karlsson, and A. Saffiotti, "Can Emil Help Pippi?," *Proc. of the ICRA-05 Workshop on Cooperative Robotics*, Barcelona, ES, Apr. 2005.

[5] R. Lundh, L. Karlsson, and A. Saffiotti, "Plan-based Configuration of a Group of Robots," *In Proc. of the European Conf. on AI*, Riva del Garda, IT, 2006, pp.683-687.

[6] A. Saffiotti and M. Broxvall, "Peis Ecologies: Ambient Intelligence Meets Autonomous Robotics," *Proc. of the sOc-EUSAI Conf. on Smart Objects and Ambient Intelligence*, Grenoble, FR, Oct. 2005.

[7] "Common Object Request Broker Architecture: Core Specification Version 3.0.3," Object Management Group, Mar. 2004.