RUPI 서버 소프트웨어 프레임워크

김록원 | 이강우 | 서영호 | 김형선
한국전자통신연구원

요 약

본고에서는 RUPI 규격이 포함하고 있는 여러 명세들 중에서 RUPI 서버에서 고려해야 되는 4가지 명세들 - (1)다수 이종 로봇 클라이언트가 RUPI서버에 접속하였을 때 고가용성을 보장하는 'RUPI 서버 가용성' , (2)원격에서 로봇 클라이언트의 상태를 진단하고 복구하는 '로봇 클라이언트 관리' , (3)로봇 클라이언트에서 수행되기 어려운 응용 컴포넌트를 로봇 클라이언트를 대신하여 적재 및 관리하고 원격에서 실행하는 '로봇 클라이언트 응용 관리' 그리고 (4)상황인지 기반의 원격 로봇 응용 개발 및 운용을 위한 '원격 로봇 응용 프레임워크' - 각각의 기술 규격에 대한 기본 개념을 소개한다. 또한 이들 각각의 기술들이 RUPI 서버 프레임워크에서 구현되기 위하여 만족해야 할 기술적 요구 사항들과 핵심 요소 기술들을 도출한다.

I. 소 개

차세대 성장 동력 사업의 일환으로 최근 수년간 정부는 로봇 산업 육성을 위하여 다양한 로봇 관련된 연구들을 활발히 진행해왔다.

전통적인 로봇 연구 분야인 로봇 플랫폼, 네비게이션, 인공지능, 음성 및 비전 인식 등은 물론이고 IT분야에서 발전된 기존 콘텐트나 소프트웨어 시스템과 로봇을 연동하기 위한 많은 연구들이 시도되었다[8][9]. Ubiquitous Robotic Companion (URC)라 명명된 네트워크 기반의 지능형 서비스 로봇 개발 사업도 이러한 시도의 하나로 시작되었다[7][10].

URC 로봇 개념에 대한 사업성을 평가 하기 위하여 2005년부터2007년 사이 두 차례에 걸쳐서 그간 개발된 URC 로봇을 가정이나 공공 장소에서 일정기간 동안 실제로 운용하는 시범 사업을 실시하였다[7]. 시범 사업의 목적은 로봇 사용자의 만족도를 평가하여 부족한 부분을 찾고 이를 개선하기 위해 필요한 기술을 보완하는 것이다. 그 결과로 URC 서비스에 대한 소비자 만족수준 향상을 위해서 지능화된 다양한 로봇 콘텐트 및 서비스를 발굴 그리고 대규모의 사용자를 지원할 수 있는 개방형 URC 시스템 개발의 필요성이 제기되었다. 웹의 경우 디스플레이, 키보드, 마우스로 구성된 디펙토(de factor) 단말 장치와 HTML과 HTTP라는 콘텐트 기술 및 전송 표준이 있기 때문에 누구나 쉽게 웹 콘텐트를 작성할 수 있다. 사용자 수준을 만족할 수 있는 다양한 로봇 콘텐트나 응용을 제공하기 위해서는 웹의 경우처럼 누구나 손쉽게 로봇 콘텐트나 응용을 작성할 수 있어야 한다. 이를 위해서 가장 시급한 기술적 요구는 표준화된 콘텐트 작성 그리고 콘텐트나 응용을 실행 할 수 있는 표준 실행 환경의 제공이다.

Robot Unified Platform Initiative(RUPI) [1]는 다양한 서비스 사업자와 로봇들이 쉽게 연동될 수 있게 하고, 로봇 및 서비스 개발 투자 비용을 최소화하기 위해 시작된 표준화된 로봇 소프트웨어 프레임워크 규격이다. RUPI는 기존의 다양한 소프트웨어 기술을 로봇에 적합한 형태로 최적화, 계층화 및 모듈화한 로봇 소프트웨어에 관한 기술 그리고 로

봇과 연동되는 외부 소프트웨어에 대한 구조 규격을 명세한 것이다. 좀더 기술적으로 정의하면 RUPI는 로봇 소프트웨어 플랫폼간의 상호 호환성, 다양한 통신 및 정보기기와의 상호 운용성, 이종 통신망과의 상호 접속성을 갖는 지능형 로봇의 소프트웨어 표준플랫폼 및 제반 표준규격이다.

2006년부터 시작된 RUPI규격은 현재2007년 12월 배포된 2.0.X 버전 상태에 있다. 앞으로 RUPI를 통해서 지능형 로봇과 네트워크 및 정보기기간의 융합 서비스가 가능하게 되고 다양한 로봇 서비스 및 콘텐트 산업의 신규 시장 창출을 기대할 수 있을 것이다.

RUPI는 크게 RUPI서버, 통신프로토콜, RUPI 로봇 클라이언트, 로봇 콘텐트, 응용 컴포넌트 그리고 통합 개발 환경 등의 세부 분야들로 구성되어 있다. 본고에서는 이들 중에서 RUPI서버에 관련된 내용을 살펴본다. RUPI 서버의 역할은 RUPI 규격에 맞춰 작성된 로봇 응용이나 콘텐트를 로봇 클라이언트가 요청하면 이에 응답하는 것이다. 이를 위해 RUPI서버는 다수의 다양한 로봇 클라이언트를 식별 관리할 수 있어야 하고 로봇 클라이언트가 요청한 콘텐트나 응용 실행 과정에서 로봇 클라이언트와 상호 작용할 수 있게 하는 클라이언트 인터페이스를 미리 알고 다양한 로봇 클라이언트들의 인터페이스 차이에 대해서 동적으로 대처할 수 있어야 한다.

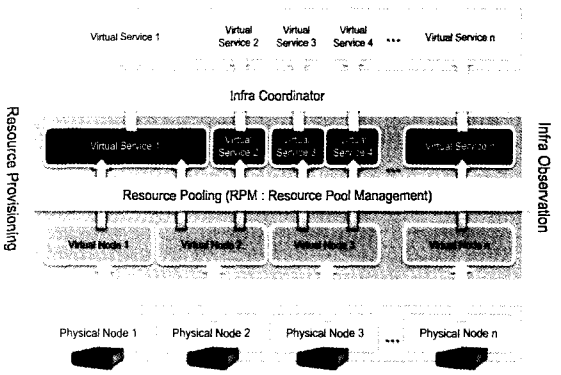
RUPI 서버 관련 규격들은 이러한 기능을 수행하는 데 필요한 기술을 크게 4가지로 나누어서 명세하고 있다. 2장에서는 서버에 관련된 4가지의 RUPI 규격들에 대하여 소개하고 3장에서는 RUPI 서버를 명세를 만족하는 시스템을 구현하는데 필요한 요소 기술들에 대해서 기술하고 4장에서는 RUPI 서버 규격의 기술적 가치를 언급하고자 한다.

II. RUPI 서버 규격

RUPI 서버 규격은 네트워크로 연결된 다수의 로봇 클라이언트의 인식하고 관리하며, 로봇 클라이언트가 요청한 로봇 콘텐트 및 응용과 같은 지능형 서비스를 제공하는 네트워크 기반 서비스 소프트웨어 프레임워크 구현에 필요한 기술을 명세한 것이다.

1. RUPI 서버 가용성

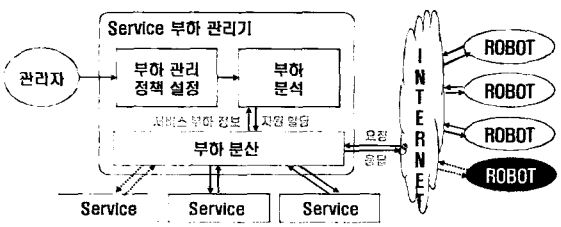
서버 가용성에 관한 RUPI 2.0.2-C11.1명세[2]는 로봇 클라이언트들이 요청하는 서비스들의 품질을 높이고 RUPI서버의 안정적인 동작을 위해 요구되는 기능들을 정의하고 있다. 서비스의 빠른 응답과 고가용성을 보장하기 위해서는 서버의 가상화 기술과 능동적인 부하 관리 기술이 필요하다.



(그림 1) RUPI 서버의 가상화

가상화 기술은 논리적RUPI 서버를 구성하는데 포함된 모든 물리적 자원들 즉, 응용 서버, 소프트웨어 컴포넌트, 네트워크 및 스토리지 등을 동적으로 할당하고 회수 가능하도록 하기 위하여 서버를 구성하는 최하위 물리적 자원들과 로봇 클라이언트에게 제공하는 최상위 서비스들 사이의 추상화 개념을 이용하여 계층화한 것이다.

(그림 1)은 RUPI 서버의 가상화 구조를 표현한 것으로 최하위 물리적인 자원들이 가상화를 통해서 최상위에서 서비스로 보여진다. 가상화된 서비스 즉, 로봇 응용이나 콘텐트가 로봇 클라이언트에게 보여지는 서비스가 된다. 로봇 클라이언트에게는RUPI 서버가 다양한 서비스를 제공하는 하



(그림 2) 부하관리 시스템 개념도

나의 서버로 보이지만 사실 여러 대의 물리적인 자원과 서버들로 구성된다.

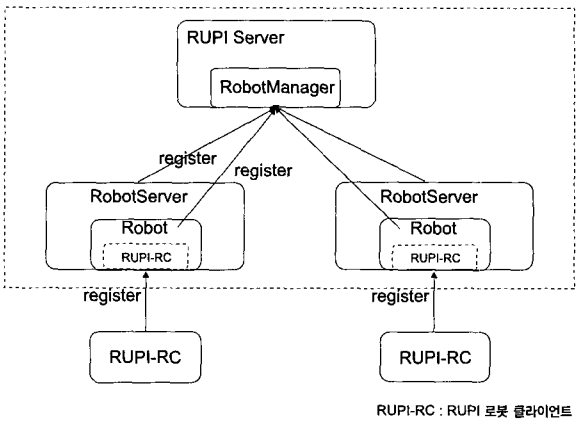
RUPI서버가 동적으로 자원을 할당하고 회수하려는 이유는 능동적인 서비스의 부하관리를 하기 위해서다. 능동적인 부하 관리 기술은 로봇 클라이언트로부터 요청되는 서비스들 중에 RUPI 서버에 부담되는 부하를 감시하여 부하 관리 정책에 따라 능동적으로 서비스에 필요한 자원을 재할당 분배하는 기능을 수행한다.

(그림 2)는 (그림 1)의 Infra Coordinator의 기능을 좀더 구조적으로 표현한 개념도이다. RUPI 서버에서 제공하는 여러 서비스들 중에서 특정 서비스에 대한 서버의 작업부하가 증가하면 부하분산 정책에 의해서 작업부하가 적은 가상 서버의 수를 줄이고 작업부하가 증가하는 서비스를 위한 가상 서버를 많이 할당 함으로써 부하의 균형을 맞추게 된다.

2. 클라이언트 관리

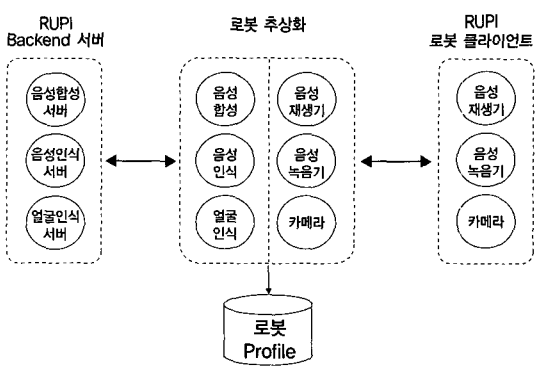
RUPI서버에 연결되는 로봇 클라이언트들의 종류는 다양하기 때문에 RUPI 서버가 여러 종류의 다수 로봇들을 관리하기 위해서는 각각의 로봇들이 어떤 기능이 있고 어떻게 관리에 필요한 정보를 가져올 수 있는지에 대해서 미리 알고 있어야 한다. 이를 위해서 로봇 클라이언트가 자기 자신에 대해서 서버에 미리 알려주어야 하는데 이를 프로파일링이라고 한다. RUPI 서버는 로봇 클라이언트가 등록한 프로파일 정보를 이용하여 다양한 종류의 로봇 클라이언트를 식별하고 임의의 로봇 클라이언트의 이벤트 획득 및 구동을 위한 인터페이스 그리고 이들을 관리하고 모니터링 하는데 필요한 인터페이스 정보를 얻을 수 있다. 이러한 정보를 바탕으로 RUPI 서버는 로봇을 관리 할 수 있게 된다.

RUPI 2.0.2-C12.1은 로봇 클라이언트 관리에 관한 규격을 명세하고 있다[3]. 먼저 RUPI 서버가 종류가 다른 다수의 로봇 클라이언트를 관리하는 체계에 대해서 살펴보자. 그림 3처럼 로봇 클라이언트를 관리하기 위해서는 RobotManager와 RobotServer 두 종류의 컴포넌트들이 필요하다. RUPI 서버는 RobotManager를 통해서 로봇 클라이언트들을 관리한다. 반드시 하나의 RobotManager가 RUPI서버에 존재해야 한다. RobotManager는 등록된 다수의 RobotServer들에 대한 참조를 가지고 있다. 임의의 RobotServer는 로봇 클라이언트의 종류별로 존재한다. 같은 RobotServer에 등록된 로봇 클라이언트들은 같은 종류이며 RobotServer의 역할은 로봇 클라이언트들의 접속 상태와 프로파일을 관리하는 것이다.



(그림 3) RUPI 로봇 클라이언트 관리 구조

로봇 클라이언트에 대한 프로파일은 RUPI 서버에 의해서 확장될 수 있다. 프로파일의 확장은 로봇 클라이언트의 기능 확장을 의미한다. 로봇 클라이언트에서 수행되는 응용 또는 콘텐트들은 로봇 클라이언트의 기능적 한계에 의존적이기 때문에 로봇 클라이언트 기능이 다양할수록 수행 가능한 응용과 콘텐트의 종류들도 증가한다. 그런데 RUPI 서버의 도움으로 저기능의 로봇 클라이언트들도 자신의 제약을 어느 정도 극복 할 수 있다. (그림 4)의 로봇 클라이언트는 음성 합성/인식 및 얼굴 인식 기능이 없다. 하지만 스피커, 마이크, 카메라 등의 장치들을 가지고 있는 경우 RUPI 서버



(그림 4) 로봇 클라이언트의 기능 확장

〈표 1〉 관리 객체의 설명

분류 항목	관리 노드	절대 경로(Root 기준)	설 명
단말 관리	DMAcc	./DMAcc/〈X〉/	단말이 관리 서버와 접속을 위해 설정하는 인증 정보 노드
	DevInfo	./DevInfo/	단말 기본 정보 노드
	DevDetail	./DevDetail/	단말 상세 정보 노드
	AddDevDetail	./RUPI/AddDevDetail	단말의 추가적 상세 정보 노드 (향후 DevDetail의 Ext에 포함되어 질 수 있음)
진단&모니터링 관리	Evt	./RUPI//Evt	이벤트 관리 노드
	Performance	./RUPI//Performance	단말의 성능분석 노드
제어 관리	Control	./RUPI//Control	단말을 제어하기 위한 노드
소프트웨어 배포 관리	Fumo	./RUPI//Fumo	단말의 펌웨어를 업데이트하기 위한 노드
	Scomo	./RUPI//Scomo	단말의 소프트웨어 컴포넌트 관리 (업데이트/설치/제거)하기 위한 노드

는 기존 프로파일 정보를 음성 합성/인식 그리고 얼굴인식이 가능하도록 로봇 클라이언트의 프로파일을 확장을 할 수 있다.

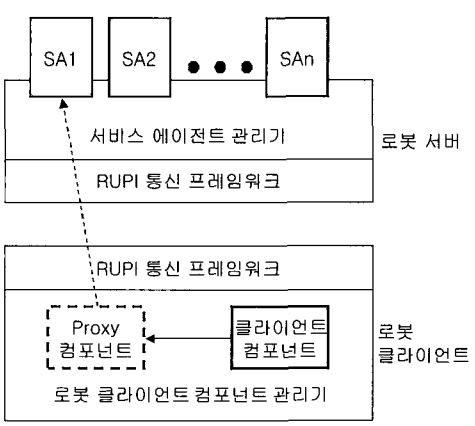
확장된 프로파일을 RUPI 서버로부터 다시 얻어와서 로봇 클라이언트의 응용을 작성할 때 확장된 기능을 이용하여 더 다양한 응용을 작성할 수 있게 된다. 이러한 프로파일 확장은 단순히 음성 합성과 같은 단일 기능이 아니라 더 고차원적인 기능으로도 확대할 수 있기 때문에 아주 단순한 기능만을 가지는 로봇 클라이언트에서도 다양하고 복잡한 응용이 구동 될 수 있다.

RobotServer의 다른 주요한 역할은 로봇 클라이언트에 대한 접속 관리, 진단 그리고 모니터링 하는 것이다. 장치에 대한 진단 및 관리적인 측면만을 고려한다면 로봇 클라이언트를 핸드폰과 같은 이동 단말로도 간주 할 수가 있다.

그러므로 이동 장치 관련 표준화 기관인 Open Mobile Alliance (OMA)에서 제안된 단말 관리 표준인 Device Management(DM) [6]의 관리 객체 정의 방법을 이용하여 로봇 클라이언트 관리를 정의한다. 〈표 1〉은 최상위 관리 객체 각각의 의미를 표현한 것이다. 최상위 관리 객체 경로에 포함된 하위 객체에 대한 자세한 내용은 RUPI 2.0.2-C12.1 명세[3]와 OMA-DM 표준 문서[13]를 참조하기 바란다.

3. 로봇 클라이언트 응용 관리

앞 절에서 프로파일링을 통하여 로봇 클라이언트의 기능이 RUPI 서버의 도움으로 확장되는 것을 설명하였다. 좀더 정확히 표현하면 실제 로봇 클라이언트의 물리적 기능이 확장된 것이 아니라 서버가 확장된 기능을 대신 수행하는 것이다. 서버가 특정 기능을 수행한다는 것은 서버가 다양한 응용 서비스를 제공하고 이를 로봇 클라이언트가 원격으로 사용할 수 있는 기능을 갖추고 있기 때문에 가능한 것이다. 이처럼 로봇에서 수행되기 어려운 응용 서비스를 로봇 클라이언트를 대신하여 적재, 관리하고 로봇 클라이언트에서 원격으로 사용될 수 있도록 하는 기능 및 기타 관련 기능을 '로봇 클라이언트 응용 관리' 라 부른다. 용어적인 혼란을 방지하기 위해서 RUPI 2.0.2-C13 명세[4]에서 사용하는 응용이라는 용어의 의미는 로봇 태스크와 비슷한 개념이 아니라 단순히 하나의 기능을 수행하는 서비스라 생각하면 된다. RUPI 서버에서 제공되는 서비스 들의 예로는 음성 인식/합성, 영상 인식 또는 날씨/뉴스와 같은 콘텐트 서비스들이 있다.



(그림 5) 로봇 클라이언트 응용관리 개념 구조

(그림 5)는 로봇 클라이언트 응용 관리의 개념적 구조를 나타낸다. 참고로 RUPI 서버와 로봇 클라이언트 간의 연결 및 상호동작은 RUPI통신 프레임워크 규격을 따른다[11][12]. RUPI 통신 프레임워크는 원격 객체 호출 방식 프로토콜로 로봇 클라이언트가 서버에서 제공하는 응용 객체에 대한 참조를 얻고, 그 객체가 마치 클라이언트 자신의 객체인 것처럼 함수 호출할 수 있는 기능을 제공한다[12]. RUPI서버가 로봇 클라이언트에게 원격 호출 가능한 응용을 제공하기 위해서 먼저RUPI 통신 프레임워크를 사용하여 원격 응용 서비스의 API들을 구현한 뒤에 이를 패키징한Server Service Agent (SSA)를 RUPI 서버에 설치 해야 한다. 또한 클라이언

트에서는 SSA를 마치 자신의 내부 컴포넌트처럼 보여지게 하는 SSA에 대한Proxy컴포넌트가 존재해야 한다.

로봇 클라이언트 응용이 올바른 동작을 하기 위하여 RUPI 서버는 원격 호출되는 응용의 시작, 중지, 종료, 설치 그리고 삭제 등을 포함하는 생명주기 관리기능을 제공해야 한다. 아울러 동시에 다수의 로봇 클라이언트에 의한 SSA사용이 가능하도록 동시 SSA검색/호출 기능을 제공해야 하며 로봇 클라이언트 오류에 의한 자원 고갈 현상을 막기 위해서 응용에 대한 자동 세션 소멸 기능도 갖추어야 한다. 보다 자세한 클라이언트 응용관리 부분의 역할과 기능은 RUPI 2.0.2-C13 명세[4]를 참조하기 바란다.

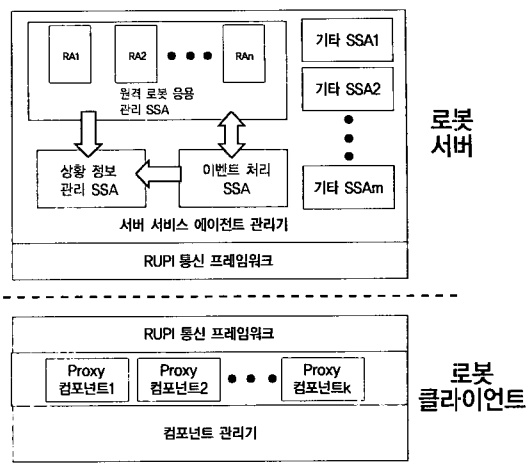
4. 원격 로봇 응용 수행을 위한 SSA

원격 로봇 응용이란 용어는 앞 절에서 설명한 클라이언트 응용처럼 원격에서 실행 된다는 점에서는 차이가 없고 의미적으로 비슷하기 때문에 혼돈되기 쉽다. 하지만 원격 로봇 응용의 경우 하나의 원격 응용 전체가 완전히 RUPI서버에서 실행되는 것에 비해 로봇 클라이언트 응용은 클라이언트가 자신의 기능인 것처럼 인식하고 사용하는 단일 서비스를 의미한다.

다시 말해 원격 로봇 응용은 로봇 클라이언트가 원격 로봇 응용 실행을 RUPI서버에 요청하면 로봇 클라이언트는 RUPI 서버로부터 원격 로봇 응용에 실행에 필요한 제어 명령을 받기 때문에 제어의 주체가 RUPI 서버가 된다. 반면에 클라이언트 응용은 로봇이 자신의 응용을 스스로 제어하면서 필요할 때 클라이언트 응용을 이용하기 때문에 제어의 주체가 로봇 클라이언트이다. 단순히 클라이언트 응용을 단일 서비스로 생각하고 원격 로봇 응용을 서버 기반 어플리케이션과 같은 개념으로 생각하면 된다.

(그림 6)은 RUPI 서버에서 원격 로봇 응용을 수행할 경우 필요한SSA들과 이들의 관계를 표현하고 있다. 앞 절의 클라이언트 응용처럼 SSA란 용어를 사용하는 이유는 RUPI 서버에서 제공하고 원격에서 호출할 수 있는 콘텐트나 응용에 대해서 SSA라고 부를 수 있기 때문이다.

(그림 6)에 나타난 이벤트 처리, 응용 관리 그리고 상황정보 관리 등의 3가지 SSA들은 클라이언트 응용처럼 단일 응용 서비스일 구현에는 필요 없지만 제어 구조가 포함되는 원격 로봇 응용을 구현 할 때 반드시 필요한 모듈들이다.



(그림 6) RUPI 원격 로봇 응용 수행환경

먼저, 이벤트 처리 SSA는 로봇 클라이언트 또는 다양한 물리 공간에 설치된 센서들로부터 이벤트(음성, 영상, 온도/습도, 일정 등의 정보)를 입력 받아 이들을 필터링하거나 조합하여 상황정보 관리 SSA나 응용 관리 SSA에 전달한다. 로봇 클라이언트로부터 발생되는 이벤트 정보는 RUPI통신 프레임워크를 통해서 입력 받기 때문에 로봇 클라이언트에는 이벤트 처리 SSA에 대한 Proxy 컴포넌트가 있어야 한다.

두 번째, 상황정보 관리 SSA는 이벤트 처리 SSA로부터 입력된 데이터를 이용하여 위치, 사용자, 로봇 클라이언트 그리고 서비스에 대한 상황 정보들을 유지하고 관리한다. 원격 로봇 응용은 상황 변화에 따라 로봇이 행위가 계속 바뀌기 때문에 상황 정보 변화를 감지할 수 있는 기능은 필수적이다. 또한 사용자 정보를 이용한 개인화 서비스를 제공할 수도 있다.

예를 들어 한 가정에서 온도 조절과 정보 콘텐트 제공에 대한 원격 로봇 응용이 수행된다고 할 때 사용자가 선호하는 온도와 콘텐트를 사용자 정보로부터 얻어서 지능적인 능동 서비스를 제공할 수 있다.

세 번째, 원격 로봇 응용 관리 SSA는 RUPI 서버에서 수행되는 원격 로봇 응용들을 수행, 관리 및 제어하는 역할을 한다. 현재 어떤 원격 로봇 응용들이 수행 중이고, 각 원격 로봇 응용의 소유자와 원격 로봇 응용의 생명주기를 관리한다. 또한 서로 충돌되거나 영향을 주는 원격 로봇 응용들의 실행 순서를 조절하는 조정자 역할도 한다. 사용자는 원격

로봇 응용 관리 SSA를 통해서 임의의 로봇 응용을 새롭게 실행 시킬 수 있으며, 실행 중이던 로봇 응용을 종료 시킬 수도 있다.

이벤트 모델에 대한 계층적 표현이나 상황 정보 모델에 대한 보다 자세한 내용은 RUPI 2.0.2-C21.1 명세[5]를 참조하기 바란다.

III. RUPI 소프트웨어 프레임워크

2장에서는 RUPI 서버 명세들 4가지 각각에 대한 개념을 간략히 설명을 하였다. 본 장에서는 RUPI 규격에 따라 로봇 서버를 구현할 때 요구되는 기술들에 대해서 설명한다.

RUPI를 지원하는 서버 소프트웨어 프레임워크구현의 목적은 상이한 로봇 클라이언트 플랫폼이 RUPI서버와 연동되어 다양한 로봇 서비스들을 수행하도록 지원하는 네트워크 기반 로봇 표준 환경을 제공하는 것이다.

여기서 로봇 표준 환경은 RUPI규격을 구현한 소프트웨어는 물론 표준화된 상호 연동, 콘텐트와 서비스 정의를 위한 핵심 도구 그리고 원격 로봇 응용 설계 및 실행 환경 등의 기능과 도구들도 포함한다. 다음은 시스템 구현 시 고려되어야 할 로봇 표준 환경에 대한 기술적 요구 사항들을 RUPI 서버 규격 항목에 따라 나열한 것이다.

- **로봇 클라이언트 관리 기술**
  - 표준화된 클라이언트 관리를 위한 로봇 클라이언트 관리 표준 규격 제공
  - 로봇 클라이언트 연결 및 프로파일 관리, 인증 및 접근제어를 위한 로봇 클라이언트 관리 기능
  - 로봇 클라이언트 서비스 에이전트 생명 주기 관리, 동적 적재, 동시성 제어 및 충돌 방지를 위한 로봇 클라이언트 서비스 에이전트 관리기
  - 로봇 클라이언트 상태 및 성능 진단, 원격 진단 및 복구를 위한 로봇 클라이언트 원격 모니터링 기술 제공

- **로봇 클라이언트 응용관리 기술**
  - 서비스 프로파일 정의를 위한 프로파일 정의 언어 및 규격 제공
  - 서버 서비스 에이전트의 동적 적재 및 생명 주기 관리를 위한 서버 서비스 에이전트 관리기
  - 서버 서비스 에이전트 디스커버리, Caching, Load Balancing을 위한 로봇 클라이언트용 Proxy 컴포넌트 기술

- **RUPI 서버 가용성 기술**
  - RUPI 서버의 고가용성 보장을 위한 능동형 시스템 운영 솔루션 제공
  - 서비스 및 데이터에 대한 장애 감지 및 복구를 위한 RUPI 서버 이중화 기술
  - 동적 서비스 할당 및 자원 할당을 위한 능동적 부하 관리 기술
  - RUPI 서버 자동관리 및 고가용성을 위한 최적화

- **RUPI 서버 기반 원격 로봇 응용 프레임워크 기술**
  - 원격 로봇에서 수행될 응용들의 등록 저장 및 응용들의 생명주기 관리를 제공
  - 센서 등 환경 정보로부터 상황인지 및 로봇 응용 구동을 위해서 발생되는 이벤트 처리 및 전달을 위한 이벤트 서버 제공
  - 서버 기반 로봇 응용 언어 (SRAL) 컴파일러 및 관련 지원 도구 제공
  - 상황정보 연동 Coordination 정책 기반의 로봇 응용간 Coordination 제공
  - 상황정보 구성 및 활용을 위한 RUPI 서버 기반 상황정보 관리 기술
  - 유비쿼터스 로봇 공간의 상황정보 융합 및 상황 변화 통보를 위한 유비쿼터스 로봇 공간 구성 기술

〈표 2〉에서는 앞에서 나열된 기술적 요구 사항을 만족시키기 위한 요소 기술들을 구체적으로 도출한 것이다. 나열된 핵심 기술들을 지원하는 소프트웨어나 장치들이 다 갖추어 진다면 RUPI 서버 규격을 만족하는 로봇 표준 환경을 제공한다고 할 수 있다. 하지만 로봇 표준 환경을 제공하기 위하여 반드시 〈표 2〉와 같은 기술들을 구성해야 하는 것은 아니며 이것은 기술 구성의 한 예시 일뿐이다.

〈표 2〉 RUPI 서버 핵심기술

	기술명	기술 목표	핵심 요소 기술
RUPI 서버	로봇 클라이언트 관리 기술	RUPI 서버에 접속된 다수의 로봇 클라이언트들의 연결을 안정적으로 관리하고, 원격에서 로봇 상태를 모니터링하고, 고장을 진단하여 복구 시키는 기술	· 로봇 클라이언트 관리 표준 규격 · 로봇 클라이언트 연결 관리 · 로봇 클라이언트 프로파일 관리 · 서비스 프로파일 관리 – 서비스 모델 및 프로파일 정의 언어 – 공용 서비스 프로파일 정의 및 서비스 에이전트 구현 – 프로파일 파서/impoter/export · 로봇 클라이언트 원격 진단/상태,성능점검/복구 기술
	로봇 클라이언트 응용관리 기술	다양한 고기능의 로봇 응용 서비스를 운영하고 이를 다수의 로봇 클라이언트에게 안정적으로 제공하기 위한 로봇응용 컴포넌트 관리 시스템	· 서비스 에이전트 관리 모델 · 서비스 에이전트 관리자 – 서비스 에이전트 생명 주기 관리 – 서비스 에이전트 형상 설정 정의 언어 – 서비스 에이전트 저장소?로봇 클라이언트용 Proxy 컴포넌트 – 서버 서비스 에이전트 디스커버리/Caching/Load Balancing
	서버 가용성 기술	다수의 로봇 클라이언트가 연결되어도 서비스 및 서버 기능을 안정적으로 제공할 수 있는 서버 가용성을 보장하는 기술	· URC서버와 데이터에 대한 이중화 – 데이터 동기화 기술 – 장애와 부하 감지 및 복구 기술 · 능동적 부하관리 – 부하 감시 정책 운용 기술 개발 – 동적 프로세스(자원) 할당/회수
로봇 원격 응용 프레임워크	서버 기반 원격 응용 프레임워크 기술	복수의 네트워크 기반 이종 로봇을 활용하는 서버 기반 응용을 개발하고 운영할 수 있는 프레임워크 기술	· 서버 기반 로봇 응용 관리 모델 · 서버 기반 로봇 응용 등록 저장소 · 로봇 응용 구동 기술 · 서버 기반 로봇 응용 Coordination – 상황정보 연동 Coordination 모델, 정책 언어 및 엔진 – Coordination 정책 저작 도구
	이벤트 서버 기술	센서 등 환경 정보로부터 상황인지 및 로봇 응용 구동을 위해서 발생되는 이벤트 처리 및 전달을 위한 이벤트 서버 제공	· 이벤트 모델 – 가상 이벤트 채널 · 이벤트 서버 – 이벤트 채널 및 구독자 관리 – 이벤트 전송 모듈 – 이벤트 필터링 표현 언어 및 상황정보연동 · 로봇 클라이언트 이벤트 라이브러리
	서버기반 원격로봇 응용 언어	서버에서 작동하는 응용을 용이하게 작성하는 프로그래밍 언어 및 관련 도구 개발	· 서버 기반 로봇 응용 언어(SRAL) 구문 정의 · SRAL 가상머신 – 명령어 세트 및 구현 – 코드 실행기 · SRAL 컴파일러
	유비쿼터스 로봇 공간 구성 기술	로봇, 환경 내 장치, 사용자 그리고 물리적 공간 등을 통합 추상화한 가상 공간을 구축하는 기술	· RUPI 서버 기반 상황정보 관리 – 시멘틱 네트워크 기반 지식베이스 · 유비쿼터스 공간 모델 · 유비쿼터스 상황정보관리

IV. RUPI 서버 기술의 가치

지금까지 RUPI 기반 로봇 서버에 대한 개념과 기술적 내용들을 살펴보았다. 일반적으로 로봇 클라이언트를 관리하는 측면에서 로봇 서버의 존재가치에 대하여는 대부분 공감을 하는데 반하여 네트워크 기반의 서버/클라이언트 구조가 로봇 응용에 적용되었을 때 얼마나 실효성이 있는 지에 대해서 의문을 가질 수도 있다. 현재의 로봇 기술 수준에서 사업 모델을 발굴하고 성공적인 산업화를 위하여 원격 로봇 응용 기술의 활용이 유용한 경우가 많다.

예를 들어 식당에서 다수의 로봇이 주문을 받고 음식을 배달하는 작업을 수행한다고 하자. 다수의 로봇들이 고객들로부터 주문을 받고 음식을 배달한다는 하나의 목적을 위해 협업한다고 볼 수 있다.

이러한 경우 로봇 서버가 다수의 로봇 클라이언트로부터 받은 이벤트 정보로부터 현재의 로봇 클라이언트의 상황정보를 유지하고 이를 이용하여 서빙 및 주문을 위한 적절한 제어 명령을 개별 로봇 클라이언트에게 하달하는 것이 각각의 로봇이 독립적으로 구동될 때보다 훨씬 쉽게 로봇 응용을 구현할 수 있는 방법이다. 현재의 기술 수준에서 로봇 시장을 창출하기 위해서는 서버 기반 원격 응용을 잘 활용하는 것이 비용적으로 유리하고 고객에게 운용상의 안정성을 제공할 수 있다.

원격 로봇 응용의 장점을 RUPI 서버 소프트웨어 프레임워크에서 제공하는 기술적인 측면에서 보면 크게 두 가지 표현할 수 있다.

먼저, RUPI 서버와 다양한 로봇 클라이언트가 표준화된 방법으로 상호 연동될 수 있도록 서비스 개발 표준 방법론과 로봇 표준 환경을 제공하기 때문에 서버 기반 로봇 서비스 및 응용 콘텐트가 한 번 개발된 다음에 다양한 로봇 클라이언트들에 쉽게 재사용 될 수 있다.

원격 로봇 응용은 로봇 클라이언트 플랫폼에 크게 영향을

받지 않기 때문에 업체가 새로운 로봇 응용을 손쉽게 개발할 수 있을 뿐만 아니라 신규 업체의 로봇 시장으로의 진입 장벽을 낮출 수 있다. 이처럼 다양한 로봇 서비스 그리고 응용 개발의 확산과 촉진을 통해 결과적으로는 전체적인 로봇 시장의 활성화에 이바지할 수 있다. 두 번째로 로봇에 집중되었던 기능들을 초고속 네트워크 인프라를 이용하여 인터넷 기반의 서버에 분산시켜 시스템 비용을 최소화하며 서비스 확장성을 최대화하기 때문에 급변하는 IT 기술의 환경 변화를 수용할 수 있고 기술 융·복합에 의한 서비스 개선 및 창조가 가능하다.

V. 결 론

본고에서는 RUPI 기반 RUPI 서버 구현에 관련된 표준 명세들의 개념과 내용 그리고 시스템 구성을 위한 주요 요소 기술들에 대해서 설명하였다. 현재 RUPI 버전 2.0.X는 아직 완전하다고 할 수 없다.

아직까지 RUPI는 진행중이며 이제 기본적인 구조, 세부적인 기술 범주와 방향이 정해진 상태이다. 그러므로 현재 RUPI 규격을 따르는 시스템을 구현하고자 할 때 세부적으로 불명확한 부분과 부족한 부분 그리고 중복되거나 일관성이 없는 부분이 분명 존재할 것이다.

이러한 문제들은 앞으로 참조 모델 구현을 통한 개념 검증을 반복하면서 보다 세련되게 다듬어 질 것이다. 빠른 로봇 산업의 육성과 시장 창출을 위해서는 RUPI 규격의 참조를 통해서 많은 피드백들이 발생하고 이를 기반으로 핵심 역량을 강화할 수 있는 기술을 도출하고 집중적으로 투자하는 것이 현재 선진국에 비해 뒤쳐진 국내 로봇산업의 빠른 성장을 위한 척도가 될 것이다.

무엇보다 현재 로봇 서버 표준에 대한 기술은 RUPI 규격에서 전세계적으로 최초로 다루어 지고 있기 때문에 RUPI 서버 소프트웨어 프레임워크 기술 개발을 통해 네트워크 기반 로봇 기술에 대한 국제 특허권 확보 및 세계 표준화를 선도하기 위한 기반을 확보할 수 있을 것이다.

참 고 문 헌

[1] Robot Unified Platform Initiative, http://www.rupi.or.kr/

[2] RUPI, "URC Server' s Availability", RUPI 2.0.2-C11.1

[3] RUPI, "The Management of URC Robot Clients", RUPI 2.0.2-C12.1

[4] RUPI, "The Management of URC Robot Client Applications", RUPI 2.0.2-C13.1

[5] RUPI, "Server Service Agents for URC Remote Robot Applications", RUPI 2.0.2-C21.1

[6] Open Mobile Alliance, http://openmobilealliance.org

[7] 김현 외, "URC(Ubiquitous Robotic Companion): 네트워크 기반 서비스 로봇", 정보과학회지 제24권 제3호 2006

[8] 지능형 서비스 로봇, IT 차세대 성장동력 기획보고서, 2003

[9] 지능형 로봇 산업비전과 발전전략, 산업자원부 정보통신부, 2005

[10] H. Kim, Y.-J. Cho, and S.-R. Oh, "CAMUS: A Middleware Supporting Context-aware Services for Network-based Robots", IEEE Workshop on Advanced Robotic and its Social Impacts, 2005.

[11] RUPI, "URC Server/Client Communication Protocol (Transport Layer)", RUPI 2.0.3-R12.3

[12] RUPI, "URC Server/Client Communication Protocol (RPC Layer)", RUPI 2.0.3-R11.2

[13] OMA "OMA Device Management", http://www.openmobilealliance.org/Technical/release_program/dm_v1_2.aspx

김 록 원

1998년 충북대학교 컴퓨터공학 학사
2000년 충북대학교 전산학 석사
2000년 ~ 현재 한국전자통신연구원 U-로봇연구본부 선임연구원
관심분야 : 상황인지 미들웨어, 지식기반 시스템

서 영 호

1998년 전남대학교 컴퓨터공학 학사
2000년 광주과학기술원 정보통신학 석사
2000년 ~ 현재 한국전자통신연구원 U-로봇연구본부 선임연구원
관심분야 : Context-aware middleware, Ubiquitous Computing

이 강 우

1991년 서울대학교 겨 산통계학 학사
1993년 서울대학교 전산학 석사
2000년 서울대학교 전산학 박사
2000년 ~ 현재 한국전자통신연구원 U-로봇연구본부 선임연구원
관심분야 : 분산시스템, 상황인식 미들웨어

김 형 선

1982년 상지대학교 전자계산학과 학사
1990년 광운대학교 컴퓨터공학과 석사
2003년 대전대학교 컴퓨터공학과 박사
1985년 ~ 현재 한국전자통신연구원 U-로봇연구본부 U-로봇서버연구팀장 책임연구원
관심분야 : 분산컴퓨팅, 정보보호, 상황인식, 전자상거래, 분산데이터베이스