모듈기반 퍼스널 로봇을 위한 미들웨어 구조

Middleware Structure for Module-based Personal Robot

윤 건*, 김 형 육, 김 홍 석, 박 홍 성
(Gun Yoon, Hyoung Yuk Kim, Hong Seok Kim, and Hong Seong Park)

Abstract : This paper proposes a middleware structure for the module-based personal robot, which can run on heterogeneous network interfaces and provides users easy interface-method regardless of underlying heterogeneous interfaces and convenient exchange of modules. The proposed middleware is divided into three layers of a streaming layer (SL), a network adaptation layer (NAL) and a network interface layer (NIL). The streaming layer manages application transactions using middleware services and provides user a uniform interfaces to the proposed middleware. The network adaptation layer manages a message-routing and provides naming service and it is a core of the proposed middleware. And the network interfaces layer manages dependent parts of heterogeneous network interfaces such as IEEE1394, USB, Ethernet, and CAN (Control Area Network). This paper implements the proposed middleware structure, where 3 types of interfaces of IEEE1394, USB and Ethernet are used, and measures response times among those interfaces.

Keywords : personal robot, middleware, heterogeneous, module

I. 서론

최근 퍼스널 로봇이라 불리는 로봇에 대한 많은 연구가 이루어지고 있다. 퍼스널 로봇은 행동, 지능, 언어등에서 인간과 유사한 특징을 지니며, 주로 가정에서 사용되기 때문에 가전 제품의 특성도 지닌다[1]. 예를 들어 퍼스널 로봇은 자체 비젼 기능과 위치 인식 기능을 이용하여 집안을 이동하며 청소하고, 또는 안전 및 보안 시스템을 점검하거나 관리하게 된다. 이와 더불어 집안 내부의 전자식 가전 기기들을 적절하게 제어 함으로써 불필요한 에너지가 사용되는 것을 예방할 수도 있다. 이와 같이 퍼스널 로봇은 인공지능, 로봇 시스템, 가전기기 및 가정 보안 시스템등 여러 기술이 적용되고 있다. 따라서 퍼스널 로봇은 다양한 기술을 지원할 수 있는 구조가 되어야 한다. 다시 말하자면 새로운 기술을 개발하여 로봇에 적용하거나 로봇에 기능을 추가할 때 퍼스널 로봇은 기존 시스템에 큰 혼돈 없이 쉽게 확장 및 적용할 수 있는 구조를 갖추고있어야 한다.

지금까지 이러한 로봇의 확장성과 교환성을 지원할 수 있는 로봇의 구조에 대하여 많은 연구가 이루어 지고 있다[2-6]. 이들 연구에서 로봇은 모듈이라 불리는 하드웨어 컴포넌트 단위로 구성되어 있다. 이와같은 모듈 기반 로봇 시스템은 모듈간 통신을 통해서 동작하는 일종의 분산 시스템이다. 따라서 로봇의 성능은 모듈뿐 아니라 모듈간 통신, 즉 모듈간 인터페이스에 따라 로봇 성능에 영향을 받는다. 그렇지만 모듈 기반 로봇에서 모듈간 인터페이스에 대한 연구는 거의 이루어지고 있지 않다. [6]에서는 이동 로봇을 위한 모듈간 통신과 이를 지원하기 위한 개방형 구조에 대하여 제안하고 있다. 모듈간 인터페이스로는 VME와 Ethernet이 사용되었으나 이들 인터페이스들은 특정 모듈간에 지정되어 사용된다. 즉 이러한 방법은 모듈들간에 인터페이스가 항상 특정한 인터페이스로 고정된 것으로 모듈간 사용될 인터페이스 종류에 제약을 주게 된다. 이와 같은 제약사항은 특정 모듈만 사용해야 한다는 의미이며, 여러 벤더에서 자신의 모듈 특성에 맞는 다른인터페이스를 개발하더라도 사용할 수 없게 된다. 따라서 이러한 제약사항은 퍼스널 로봇의 확산을 막게 되는 요건이 된다. 실제로 퍼스널 로봇에 사용 가능한 인터페이스에는 IEEE1394[7], USB[8], CAN[9], Ethernet, WLAN, Bluetooth [10]등과 같이 다양하고 많은 종류가 존재한다. 이와 같은 네트워크들은 전송 속도, 대역폭등 특징이 종류별로 각각 다르다. 따라서 모듈 기반의 퍼스널 로봇에서 로봇의 성능을 충분히 뒷받침하기 위해서는 모듈간 인터페이스에 대한 연구가 필요하다.

퍼스널 로봇에 사용되는 기술중 영상 및 음성에 대한 기술은 다루는 데이터 용량이 크며 지속적으로 발생하기 때문에 충분한 전송 속도와 대역폭을 필요로 한다. 반면에 단순한 메시지 전송은 데이터 용량이 크지 않기 때문에 상대적으로 적은 대역폭의 네트워크로도 사용가능하다. 영어등의 학습용 퍼스널 로봇과 청소용 퍼스널 로봇, 엔터테이먼트용 퍼스널 로봇은 응용이 다르므로 사용되는 모듈도 각각 다르다. 이러한 모듈들을 일일이 모두 개발하는 것은 불가능하기 때문에 범용 모듈을 구입해서 개발하는 것이 좋은 방법이다. 다시 말하자면 네트워크 인터페이스들은 모듈에 따라 통신에 필요한 대역폭, 링크 속도, 패킷 크기, 사용용도 등에 따라 선택되어야 한다. 따라서 퍼스널 로봇 모듈들이 사용하는 인터페이스들은 서로 다를 수 있다. 이것은 퍼스널 로봇 벤더들이 퍼스널 로봇의 모듈간 호환성 및 광범위한 사용성을 지원하기 위해 하나 이상의 인터페이스에 기반한 퍼스널 로봇을 개발하여야 한다는 것을 의미하기도 한다. 그러나 개발자가 인터페이스의 종류마다 특별한 어플리케이션을 개발하는것은 낭비적 요소가 많다. 그 이유는 네트워크 종류마다 프로

* 책임저자(Corresponding Author)
논문접수 : 2003. 6. 17., 채택확정 : 2004. 1. 8.
윤건, 김형육, 박홍성 : 강원대학교 전기전자정보통신공학부
(yoongun@control.kangwon.ac.kr/petrus@control.kangwon.ac.kr/hspark@cc.kangwon.ac.kr)
김홍석 : 한국생산기술연구원 허브-로봇센터(hskim@kitech.re.kr)
※ 본 논문은 2003년도 강원대학교 BK21 사업에 의하여 지원 되었음.

토콜, 사용방법등이 모두 다르고 이를 개발자가 모두 이해하고 개발하는데에는 많은 시간과 노력이 들기 때문이다. 따라서 하나 이상의 개방형 인터페이스를 지원하기 위해서는 효과적인 미들웨어 구조가 필요하다.

현재 사용되고 있는 미들웨어로는 RMI[11], CORBA[12], COM or DCOM[13]과 같은 종류가 있다. 이들 대부분의 미들웨어들은 원격의 분산 객체에 접근하거나 통신하기 위해 TCP/IP[14]를 기반으로 동작한다. 그러나 TCP/IP를 USB, IEEE1394, CAN과 같은 네트워크에 적용하는 것은 현재로서는 일반적인 방법이 아니다. 비록 CORBA의 ESIOP가 TCP/IP기반이 아닌 다른 프로토콜을 기반으로 하여 동작할 수 있지만 동시에 여러 이종 네트워크를 지원할 수 없다. 결과적으로 기존의 모든 미들웨어는 퍼스널 로봇의 요구사양인 개방형 이종 인터페이스를 만족시키지 못하고 있다.

따라서 개방형 이종 인터페이스를 지원하는 모듈기반의 퍼스널 로봇을 위한 새로운 미들웨어 구조에 대한 연구가 필요하다. 본 논문은 퍼스널 로봇을 위한 미들웨어 구조를 제안하고 IEEE1394, Ethernet, USB를 사용하여 미들웨어를 구현하고, 그 동작을 검증한다. 본 논문에서 제안하는 미들웨어는 스트리밍 계층 (Streaming Layer, SL), 네트워크 적응 계층 (Network Adaptation Layer, NAL), 네트워크 인터페이스 계층 (Network Interface Layer, NIL)의 3계층으로 나뉘어 진다. 스트리밍 계층은 미들웨어 서비스를 이용하여 어플리케이션 트랜잭션을 관리하고 미들웨어 사용자에게 제안된 미들웨어로의 일정한 인터페이스를 제공한다. 네트워크 적응 계층은 메시지 라우팅, 네이밍 서비스와 같이 종류가 다른 여러 네트워크를 수용하기 위한 서비스를 제공하는 계층으로서 미들웨어의 핵심 기능을 수행한다. 그리고 네트워크 인터페이스 계층은 IEEE1394, USB, Ethernet 그리고 CAN과 같은 이종 인터페이스들의 네트워크 의존적인 부분을 관리하고 수행한다. 미들웨어 사용자는 종류가 다른 여러 네트워크로 구성된 네트워크 환경에서 제안된 미들웨어를 사용함으로써 어떤 모듈이 어떤 네트워크로 연결되어 있는지, 그리고 네트워크로 데이터를 전송하기 위해서는 어떤 방법을 사용해야 하는지에 대하여 관여하지 않고 원하는 응용들을 쉽게 개발할 수 있다.

다음장에서는 제안된 미들웨어 구조에 대하여 설명하고, 3장에서는 미들웨어를 이용하여 메시지를 전송하는 과정과 그에 따른 계층별 동작에 대하여 설명한다. 4장에서는 제안된 미들웨어를 구현하고 그 성능을 분석하며, 마지막으로 5장에서는 결론을 맺는다.

II. 모듈 기반 퍼스널 로봇을 위한 미들웨어

퍼스널 로봇의 모듈을 공급하는 업체는 모듈이 사용하는 데이터 종류와 그 크기를 고려하여 모듈에 사용될 인터페이스를 선택한다. 예를 들어 영상 및 음성 데이터를 처리하는 모듈은 다루는 데이터 용량이 크고 주기적으로 생성되기 때문에 통신 대역폭이 크고, 링크 속도가 빠른 IEEE1394가 적합하다. 반면 이동 모듈은 영상 모듈과 달리 사용하는 데이터 크기가 비젼 데이터에 비해 그 크기가 작아서 IEEE1394보다

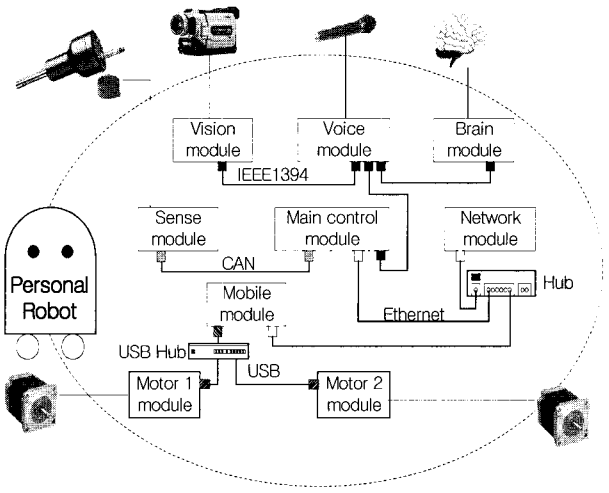


그림 1. 모듈기반 퍼스널 로봇.
Fig. 1. A Module-based Personal Robot.

저렴하면서 링크속도가 IEEE1394보다 비교적 느린 USB를 사용할 수도 있다. 이러한 구성의 예가 그림 1에 나타나 있다. 로봇의 성능면에서 볼 때 가능하면 실시간 성능을 제공하는 IEEE1394, 혹은 CAN을 사용하는 것이 바람직하지만 만약 해당 모듈에서 IEEE1394, 또는 CAN 인터페이스를 제공하지 않으면 다른 인터페이스를 사용하여야 한다. 이와 같이 퍼스널 로봇에서 모듈간 인터페이스는 하나 이상의 종류가 사용될 수 있기 때문에 언제든지 다른 종류의 인터페이스와 연결할 수 있어야 한다. 따라서 모듈 기반 퍼스널 로봇을 위한 미들웨어는 인터페이스의 종류에 상관 없이 모든 인터페이스를 지원할 수 있는 개방형 구조를 갖추고 있어야 한다.

본 논문에서 제안하는 모듈 기반의 퍼스널 로봇을 위한 미들웨어의 구조는 그림 2와 같다. 제안된 미들웨어는 응용에 미들웨어의 서비스를 제공하는 스트리밍 계층 (Streaming, SL), 종류가 다른 여러 네트워크를 수용하는 네트워크 적응 계층 (Network Adaptation Layer, NAL), 네트워크 의존적인 기능을 담당하는 네트워크 인터페이스 계층 (Network Interface Layer, NIL)의 세 계층으로 구성되어 있다.

스트리밍 계층은 미들웨어의 서비스를 사용하는 어플리케이션에 관한 요소로 이루어져 있다. 주요 기능은 네트워크 적응 계층을 이용하여 어플리케이션이 네트워크의 종류에 상관없이 원격 모듈에 존재하는 어플리케이션에 접근하기위한 메커니즘을 제공하는 것이다. 다시 말하자면, 원격 모듈에 존재하는 변수를 쉽게 읽거나 쓰기 또는 원격 함수를 호출하는 등의 서비스를 스트리밍 계층에서 제공하며, 이와 관련된 어플리케이션 관리, 트랜잭션 관리등도 병행하여 실행한다.

네트워크 적응 계층은 네트워크 인터페이스 계층에 추가된 여러 종류 네트워크 컴포넌트들을 통합하는 계층으로서 실질적인 미들웨어의 핵심 기능을 담당하는 계층이다. 이 계층은 메시지 라우팅, 이종 네트워크간 모듈 어드레싱, 네이밍 서비스 등의 기능을 수행하며, 네트워크 인터페이스 계층의 여러 네트워크 컴포넌트들을 통합한다.

네트워크 인터페이스 계층은 여러 종류의 네트워크 컴포넌

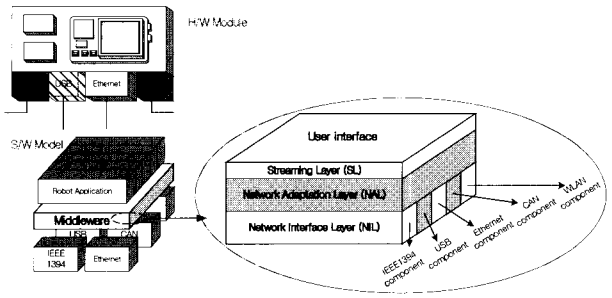


그림 2. 미들웨어 구조.
Fig. 2. Middleware Structure.

트가 존재하는 계층으로서 네트워크 종류에 따라 하드웨어, 소프트웨어에 의존적인 부분을 수행하는 컴포넌트로 구성되어 있다. 이러한 네트워크 인터페이스 계층은 미들웨어에서 새로운 네트워크가 추가 되거나 삭제될 때 일일이 미들웨어를 수정하는 일이 없도록 하며, 각 네트워크의 의존적인 부분을 담당 및 관리할 수 있는 구조를 갖춘다.

그림 2와 같은 미들웨어 구조에서는 사용자에게 필요한 네트워크를 그 종류에 상관 없이 미들웨어에 쉽게 추가할 수 있고, 응용 개발자는 네트워크상의 원격 또는 로컬에 존재하는 객체에 접근할 때 네트워크의 종류에 무관하게 미들웨어를 통해 손쉽게 응용 프로그램을 설계 및 구현할 수 있다.

본 논문에서 제안하는 미들웨어 구조는 개방형 인터페이스를 지원하기에 매우 적합한 구조이며, 상세 구조는 그림 3과 같다.

1. 스트리밍 계층(Streaming Layer, SL)

스트리밍 계층의 자세한 구조가 그림 4에 나타나 있다. 스트리밍 계층은 미들웨어의 서비스를 사용하는 어플리케이션에 관한 요소로 이루어져 있다. 주요 기능은 어플리케이션이 네트워크의 종류에 상관 없이 로컬 모듈 또는 원격 모듈에 존재하는 어플리케이션에 접근하기위한 방법을 제공하는 것이다. 여러 로봇 어플리케이션이 미들웨어를 사용할 수 있도록 스트리밍 계층은 여러 로봇 어플리케이션을 관리하고 등록 API 를 어플리케이션에 제공함으로써 로봇 어플리케이션이 스트리밍 계층에 쉽게 등록할 수 있도록 한다. 로봇 어플리케이션이 스트리밍 계층에 등록한 이후에는 언제든지 미들에어의 서비스를 받을 수 있다. 그 예로 변수 읽기 서비스는 원격 또는 로컬 모듈에 존재하는 어플리케이션의 변수를 읽을 때 변수 읽기 서비스를 요구한 스트리밍 계층 측에서는 원격 또는 로컬에 요구를 하고 요구에 대한 응답을 기다려야 한다. 만약 변수 읽기에 대한 요구가 동시에 여러 로봇 어플리케이션으로부터 스트리밍 계층으로 들어왔다면 요구에 대한 응답은 요구 개수만큼 이루어 질것이다. 그러나 이렇게 응답을 기다리는 트랜잭션이 많으면 응답을 수신했을 때 어느 요구에 대한 응답인지 알 수 없게 된다. 이러한 오류를 막기 위해 스트리밍 계층에서는 각 요구에 대한 트랜잭션을 관리한다.

로컬 어플리케이션이 원격의 어플리케이션에 접근하기 위해서는 어플리케이션의 요구가 네트워크를 통해 전송되어야 한다. 그러나 어플리케이션 레벨에서 사용하는 메시지를 그대로 네트워크로 전송하는 것은 수신측에서 수신한 데이터를

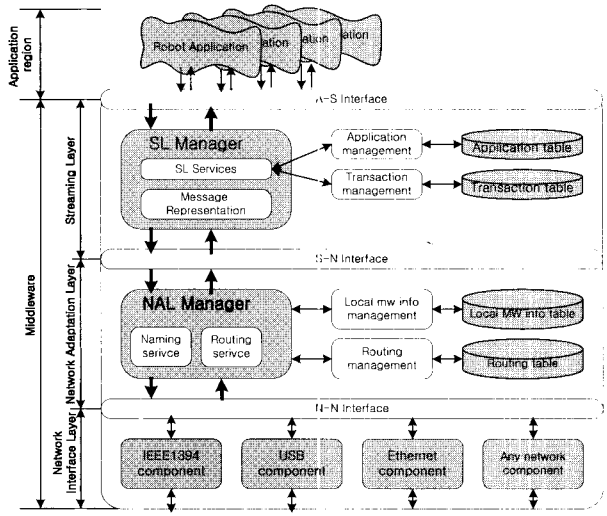


그림 3. 미들웨어 상세 구조.
Fig. 3. Detailed Middleware Structure.

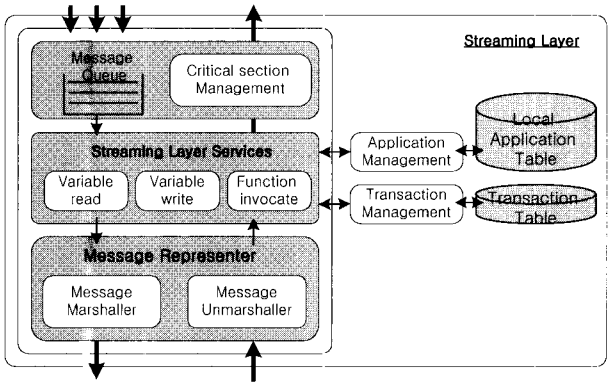


그림 4. 스트리밍 계층.
Fig. 4. Streaming Layer.

처리하는데 매우 애매한 상황을 초래할 수 있다. 예를 들어 원격의 모듈 A에 존재하는 변수 'b'에 대하여 그 값을 읽어오는 요구를 할 경우 "read(A.b)"의 명령어가 전송되는데, 원격의 모듈 A에서는 수신한 데이터를 일반 문자열로 처리할지 아니면 정상적인 요구인 "read(A.b)"로 처리할지 알지 못한다. 또한 어플리케이션 레벨의 메시지를 그대로 네트워크로 전송할 경우 필요 없는 공백문자, 구분자 같은 오버헤드가 늘어날 수 있다. 이 같은 이유로 어플리케이션에서 요구한 데이터는 네트워크로 전송하기에 알맞은 형태로 메시지의 재표현 과정을 거쳐야 한다. 메시지 재표현 과정에는 그림 4와 같이 마샬링과 언마샬링이 있다. 마샬링이란 어플리케이션이 요구한 서비스는 네트워크로 전송하기위해 적당한 형태로 재표현하는 과정을 말하며 언마샬링은 그 반대로 네트워크로 수신된 데이터를 다시 원래 어플리케이션 레벨의 메시지로 복원하는 과정을 말한다. 모듈 기반 퍼스널 로봇과 같은 분산 시스템 환경에서 분산 어플리케이션간 상호 운용성을 지원하기 위해서는 이러한 메시지 재 표현과정이 사전에 표준화되어야 한다. 예를 들어 통신 채널상의 전송측과 수신측의 미들웨어는 데이터의 크기나 포맷에대해 일치되어

있어야 한다. 그렇지 않으면 수신측에서는 수신된 데이터를 복원 및 이해할 수 없기 때문이다. 메시지 변환 과정에 대한 표현이 (1)에 나타나 있다.

$T(x)$: Translate Raw Message into the Serialized Message using Marshaller of Message Representer

$T^{-1}(x)$: Translate Serialized Message into the Raw Message using Unmarshaller of Message Representer

$$RawMsg \xrightarrow{T(RawMsg)} Serialized\ Msg \qquad (1)$$
$$\xrightarrow{T^{-1}(Serialized\ Msg)} RawMsg$$

명령 read(A.b)와 문자 스트링 메시지 "read(A.b)"가 1과 같은 변환 과정을 거칠때 *SerializedMSG*의 간단한 표현이 표 1에 나타나 있다. 수신 측에서는 패킷 타입으로 수신한 메시지가 명령인지 문자 스트링인지 구분하게 되고 명령일 경우에는 일반 변수인지 레지스터 값인지 그리고 스택의 값인지 구별하여 해당하는 변수 값을 읽는다. 문자 스트링일 경우에는 일반 문자 스트링으로 처리한다.

이와 같이 스트리밍 계층은 실제로 미들웨어의 최상위에 존재하는 계층으로서 로봇 응용 프로그램과 직접 인터페이싱 하는 계층이다. 스트리밍 계층에 등록한 응용 프로그램에 미들웨어의 서비스를 제공하고 응용들이 미들웨어 서비스를 사용하는데 필요한 어플리케이션 동기화와 트랜잭션, 그리고 네트워크로 전송하기에 알맞게 데이터를 재 표현하는 과정을 수행한다.

2. 네트워크 적응 계층 (Network Adaptation Layer, NAL)

네트워크 적응 계층은 모듈 기반 퍼스널 로봇에 개방형 이종 인터페이스를 지원하기 위한 핵심 기능을 제공하는 계층으로서 모듈간 안전하고 효율적으로 데이터를 전송하기 위해 여러 종류의 네트워크들을 투명하게 관리하여야 한다. 여기에서 투명성이란 네트워크 적응 계층을 통해 네트워크의 종류에 상관없이 네트워크들을 사용할 수 있다는 뜻이다.

이러한 네트워크 적응 계층의 주요 기능은 종류가 다른 네트워크들을 하나의 가상 네트워크로 통합하여, 네트워크 사용자가 단일 네트워크처럼 사용할 수 있도록 하고, 개방형 구조를 갖춤으로써 동적으로 이종 네트워크들을 지원할 수 있도록 하는 것이다.

여러 종류의 네트워크를 통합하는 과정은 이종 네트워크간 어드레싱, 메시지 라우팅하는 방법을 포함한다. 모듈 기반 퍼스널 로봇에서 어드레싱 방법이란 종류가 다른 여러 네트워크로 구축된 네트워크 환경에서 모듈간 데이터 통신을 하기위해 필요한 주소 체계를 의미한다. 그 예가 그림 6에 나타나 있다. 예를 들어 IEEE1394에서는 노드 ID를 사용하여 모듈간 어드레싱을 하고 있으며 Ethernet에서는 IP를 사용하여 어드레싱하고 있다. 그림 6과 같이 IEEE1394와 Ethernet을 동시에 혼용하여 사용한다면 양 끝단의 디스플레이 모듈과 센서 모듈간 직접적인 데이터 전송은 불가능하다. 센서 모듈에서 센서들의 값을 측정하여 그 값이 디스플레이 모듈에 전송하고 디스플레이 모듈에서 수신한 값을 출력한다고 가정하면 센서 모듈의 데이터가 디스플레이 모듈에 성공적으로 전송되어야 한다. 그러나 Ethernet에서는 패킷을 전송하기위해 어드레싱 방법으로 IP를 사용하고 IEEE1394에서는 노드 ID를 사용한다. 따라서 브레인 모듈에서는 수신한 패킷을 다시 노드 ID를 이용하여 패킷을 디스플레이 모듈로 전달하여야 한다. 이와 같이 데이터를 전송하고 또다시 수신측에서 최종 목적지까지 데이터를 전달하기위해서는 센서 모듈에서 브레인 모듈에 대한 디스플레이 모듈의 IEEE1394 버스상의 노드ID를 알고 있어야 한다.

다시 말하자면 디스플레이 모듈로 패킷을 전달하기위해 브레인 모듈에대한 디스플레이 모듈의 IEEE1394노드 ID를 패킷 헤더에 삽입하여야 한다는 것이다. 이러한 과정은 매우 번거럽고 일일이 모든 모듈에 대한 상대적 주소를 모든 모듈이 알고 있어야 함을 의미한다. 따라서 새로운 모듈간 어드레싱 방법이 필요한데 본 논문에서는 이를 모듈 어드레싱이라고 한다.

모듈 어드레싱은 네트워크 적응 계층에서 이루어진다. 네트워크 적응 계층에서는 퍼스널 로봇 내부 내트웍에서 고유한 모듈 ID를 생성한다. 이때 사용하는 방법은 모듈의 이름을 Hash하여 얻어 내며 모듈의 이름은 로봇 내부에서 고유

표 1. 메시지 재표현기에서의 메시지 정의.
Table 1. Message define in the Message Respresenter.

메시지	패킷 타입	변수 타입	변수이름
read(A.b)	VAR_READ	one of variable, register, stack	b
"read(A.b)"	STRING	read(A.b)	

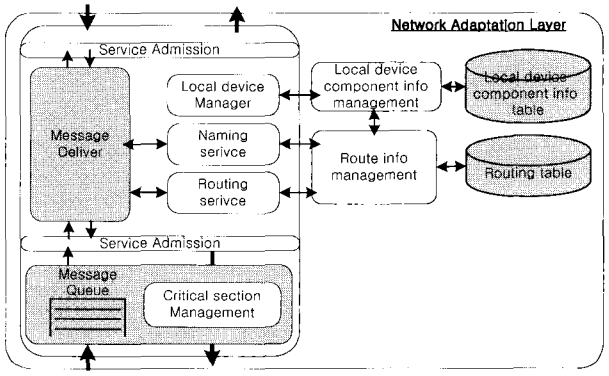


그림 5. 네트워크 적응 계층.
Fig. 5. Network Adaptation Layer.

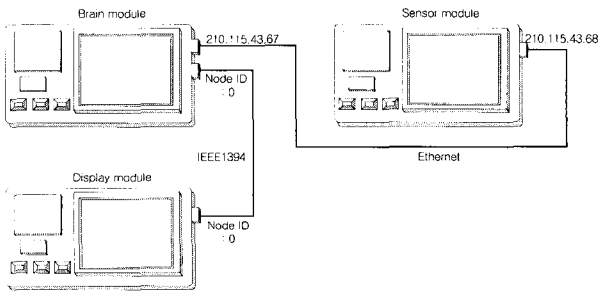


그림 6. 모듈 어드레싱.
Fig. 6. Module Addressing.

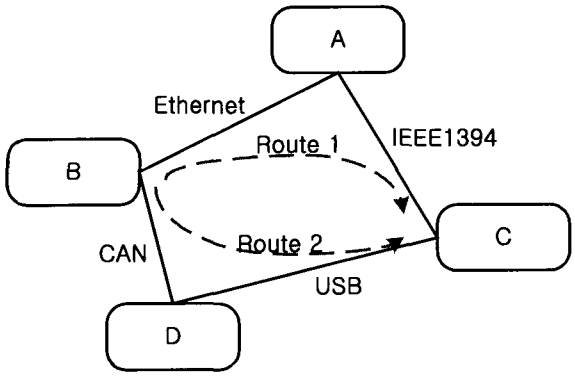


그림 7. 이종 네트워크 환경에서의 다중 경로.
Fig. 7. Multiple routes in the heterogeneous network Environments.

한 이름을 갖는다고 가정한다. 이렇게 생성된 모듈 ID를 기반으로 네트워크 적응 계층은 원격의 모듈과 데이터 통신을 할 수 있도록 한다. 이해를 돕기 위해 TCP/IP와 본 논문에서 제안한 모듈 어드레싱과 비교 한다면, 모듈 ID는 TCP/IP의 IP와 같은 역할을 하며 스트리밍 계층의 서비스를 사용하는 어플리케이션 ID는 TCP/IP의 Port와 같은 역할을 한다.

네트워크 적응 계층에서는 이러한 모듈 어드레싱 방법을 구체화하여 사용할 수 있도록 하기위해 메시지 라우팅 방법을 정의한다. 그림 6에서 설명한 바와 같이 종류가 다른 여러 네트워크를 사용하는 네트워크에서 두 가지 이상의 네트워크를 통해 메시지를 전달할 경우가 생긴다.

그림 6과 같은 경우는 간단한 경우이지만 3가지 이상의 네트워크가 사용되고 여러 종류의 네트워크 연결이 루프(loop)를 이룬다면 그림 7과 같이 B모듈에서 C모듈로 데이터를 전송할 경우 경로 1과 경로 2가 생겨 다중 경로가 형성된다.

다중 경로가 나타나면 어느 경로가 최적의 경로인가를 찾아내는 과정이 필요하게 된다. 예를 들어 그림 7의 2번 경로는 1번 경로보다 느리다. 그 이유는 CAN(1Mbps), USB (12Mbps)로 구성된 1번 경로보다 Ethernet(100Mbps), IEEE1394 (400Mbps)로 구성된 2번 경로가 빠르기 때문이다. 따라서 데이터 전송에 있어서 1번 경로가 2번 경로보다 속도면 에서 유리하다고 할 수 있다. 이와 같이 모듈 기반 퍼스널 로봇 내부에 사용될 네트워크에서는 패킷을 올바로, 그리고 최적의 경로로 전송하기위해 라우팅 서비스가 필요하다. 패킷을 올바른 경로로 전달하기 위해서 최종 목적지 경로에 대한 정보는 라우팅 테이블로 관리된다. 기존에 사용되는 라우팅 알고리즘은 TCP/IP기반의 알고리즘이다. 그러나 본 논문에서 제안하는 미들웨어에서는 모듈 ID와 링크 스피드, 그리고 사용 가능한 대역폭을 기반으로 라우팅을 해야한다. 따라서 기존의 라우팅 방법을 사용하지 않는 새로운 라우팅 방법을 사용한다. 라우팅 알고리즘은 Link State Protocol의 OSPI[14]에서 사용하는 알고리즘과 유사하며 브로드캐스팅하는 라우팅 정보에는 모듈 이름, 모듈 ID, 네트워크 인터페이스 계층 정보 홉간 거리, 비용과 같은 정보가 포함되며, 네트워크 인터페이스 계층의 정보는 디바이스 이름, 디바이스가 사용하는 네트워크 종류, 네트워크 종류에 따른 어드레싱 방법, 실제 네트워크의 어드레스로 구성되어 있다.

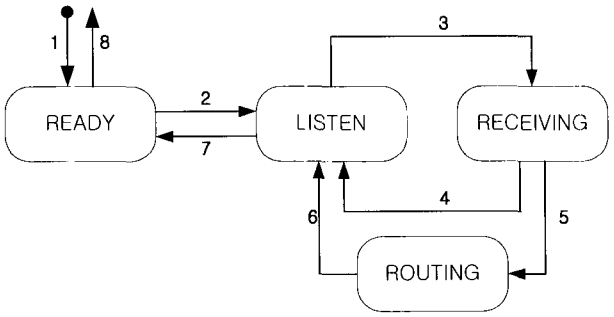


그림 8. 메시지 라우팅 상태 천이도.
Fig. 8. State diagram for message routing.

네트워크 적응 계층에서는 이와 같은 라우팅 정보를 네트워크상의 모든 모듈에 브로드캐스트하고 이 정보를 바탕으로 퍼스널 로봇 내부의 각 모듈은 각자의 라우팅 테이블을 작성한다. 라우팅 테이블은 모듈 ID를 기반으로 하여 네트워크상에 있는 모듈이 현재 자신의 모듈과 어떤 네트워크로 연결되어 있는지, 그리고 네트워크 종류에 따른 어드레싱 방법을 포함하고 있다. 또한 특정 모듈로 데이터를 직접 전송할 수 없을 경우 최종 목적지로의 다음 홉에 관한 정보가 유지관리된다.

이러한 방법으로 만들어진 라우팅 테이블을 이용하여 네트워크 적응 계층에서 패킷을 전달할 때 목적지 모듈이 사용하는 네트워크가 어떤 네트워크인지, 실제 네트워크가 사용하는 주소는 무엇인지 알 수 있게되며, 최종 목적지가 아닌 다음 홉에 패킷이 전달되어 라우팅 과정이 필요할 경우 네트워크 적응 계층에서 수신한 패킷의 헤더에 담겨 있는 모듈 ID를 라우팅 테이블 에서 참조하여 최종 목적지 또는 다음 홉으로 패킷을 라우팅할 수 있게 된다. 예를 들어 그림 7에서 모듈 B 가 C로 1번 경로를 이용하여 데이터를 전송한다고 가정하면 모듈 B는 네트워크 적응 계층의 헤더의 최종 목적지 필드에 모듈 C의 ID를 삽입한다. 그런 후 완성된 패킷을 모듈 A로 전송하고 모듈 A는 수신된 패킷의 최종 목적지가 C이므로 자신의 라우팅 테이블을 참조하여 패킷을 모듈 C로 라우팅 한다.

이상과 같이 메시지를 라우팅하는 방법은 모듈 ID를 라우팅 테이블과 매핑 시킴으로써 패킷 전달시 헤더에 순차적으로 필요한 각 네트워크의 어드레스를 감소시키며, 결과적으로는 패킷의 오버헤드를 줄임으로써 효과적인 데이터 통신을 할 수 있게 한다.

라우팅 과정은 항상 수신 대기 상태에서 시작한다. 네트워크 적응 계층에 패킷이 도착한 이후에 패킷의 헤더를 분석하고 라우팅이 필요한 패킷일 경우에만 라우팅 서비스를 한다. 따라서 네트워크 적응 계층의 라우팅 서비스를 위한 상태 천이도는 그림 8과 같이 나타낼 수 있다. 또한 그림 8의 상태 천이에 대한 설명이 표 2에 있다.

네트워크 적응 계층의 수신 대기 상태는 미들웨어의 수신 대기 상태와 같으며 각 상태에 대한 정의는 다음과 같다. READY 상태는 미들웨어 초기화 성공 상태이며 LISTEN 상태는 수신 대기 상태이다. RECEIVING 상태는 메시지 수신

표 2. 상태 천이표.
Table 2. State Transition Table.

	Current State	Trigger	New State
1	…	네트워크 디바이스초기화 성공	READY
2	READY	네이밍 서비스 및 라우팅 서비스 초기화 성공	LISTEN
3	LISTEN	메시지 수신 시작	RECEIVING
4	RECEIVING	메시지 수신 직후	LISTEN
5	RECEIVING	라우팅 서비스가 필요한 메시지 수신 직후	ROUTING
6	ROUTING	메시지에 대한 라우팅 서비스가 끝난 직후	LISTEN
7	LISTEN	수신 중지 명령	READY
8	READY	미들웨어 서비스 종료 명령	…

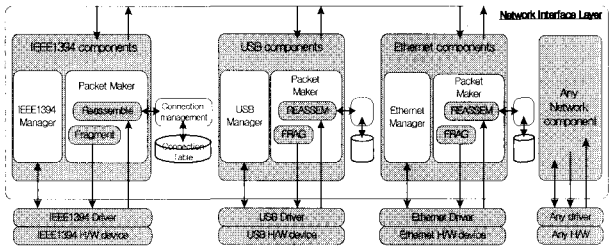


그림 9. 네트워크 인터페이스 계층.
Fig. 9. Network Interface Layer.

상태이고 ROUTING 상태는 메시지 라우팅 상태이다.

네트워크 적응 계층은 라우팅 서비스이외에도 스트리밍 계층으로부터 전달 받은 메시지에 대한 최종 목적지를 찾기 위한 네이밍 서비스를 지원한다. 이때 사용되는 네이밍 서비스는 어플리케이션에서 최종 목적지를 이름을 사용하여 명시하는 방법으로서 분산 어플리케이션 개발자가 목적 객체에 접근하기 위해 ID가 아닌 이름으로 접근하는 것을 가능하게 한다.

일반적으로 개발자가 개발을 하는데 있어서 어떤 객체에 대 한 ID를 이용하여 프로그램 하는 것 보다 객체에 대한 이름을 사용하는 것이 숫자에서 오는 오류를 예방할 수 있다. 이와 같이 네이밍 서비스를 제공함으로써 개발자에게 프로그램 능률을 향상시키는 장점을 가지고 있다. 이러한 네이밍 서비스를 이용하여 라우팅 테이블을 검색하게 되며, 검색이 성공하면 라우팅 정보에 따라 패킷을 전송한다.

요컨대 네트워크 적응 계층은 종류가 다른 여러 네트워크들을 통합하여 가상의 통합된 네트워크 환경으로 만들어 미들웨어가 어떤 종류의 네트워크든 지원 가능하도록 한다. 또한 네트워크 적응 계층의 개방형 구조를 지원하는 인테페이스 관리와 네이밍 서비스, 그리고 라우팅 서비스는 능동적으로 이종 네트워크를 지원할 수 있도록 한다.

3. 네트워크 인터페이스 계층 (Network Interface Layer, NIL)

네트워크 인터페이스 계층의 자세한 구조가 그림 9에 설명되어 있다. 네트워크 인터페이스 계층의 주요 기능은 종류가 다른 여러 네트워크의 컴포넌트들을 수용하며, 각 네트워크에 의존적이거나 특징적인 부분을 네트워크 컴포넌트를 이용하여 관리하는 것이다.

각 네트워크의 의존적인 부분은 각 컴포넌트 내부에 존재하는 관리자가 처리를 한다. 패킷 전송과 수신에 있어서 각 네트워크마다 전송 가능한 패킷의 최대 크기가 다르기 때문에 각 컴포넌트의 Packet Maker가 패킷을 분할하거나 재조합하는 기능을 수행하여야 한다. 이와 같이 네트워크 인터페이스 계층이 네트워크별 컴포넌트로 구성된 이유는 각 네트워크가 사용하는 표준이 다르기 때문이다. 표준이 다르면 하드웨어 및 소프트웨어적인 명세가 다르게 되고, 이로 인해 종류가 다른 네트워크를 서로 연결할 경우 특별한 소프트웨어나 추가적인 하드웨어 없이는 통신을 할 수 없게 된다. 예를 들어IEEE1394는 IEEE Std 1394-1995[7]의 표준을 따르며, Ethernet은 IEEE 802.3의 표준을 따른다. 또한 USB는 Universal Serial Bus Specification[8]을 따른다.

이와 같이 네트워크 인터페이스 계층은 본 논문에서 제안하는 이종 네트워크 인터페이스를 지원하기위해 네트워크 종류별 의존적인 특성을 담당하는 컴포넌트로 구성되어 있으며, 이러한 구성은 네트워크 적응 계층이 이종 네트워크들을 통합하기위한 근본적인 메커니즘을 가능하게 한다.

III. 데이터 전송 절차

3장에서는 각 모듈에서 동작하는 로봇 어플리케이션이 미들웨어를 사용하여 이종 네트워크 환경에 존재하는 원격의 모듈로 메시지를 전송하는 과정을 상세히 설명한다.

로봇 어플리케이션은 스트리밍 계층에서 제공하는 서비스를 이용하여 원격 또는 로컬에 대한 서비스를 요청한다. 스트리밍 계층에서는 각 요구에 대하여 메시지를 생성하고 네트워크로 전송하기 알맞은 타입으로 메시지를 재 표현한다. 이와 관련된 내용은 2장의 1절에 설명되어 있다. 스트리밍 계층에서 생성된 스트림화 된 메시지는 네트워크 적응 계층에서 네이밍 서비스를 이용하여 메시지의 목적지를 찾게 된다. 목적지를 찾은 이후에는 네트워크 적응 계층의 헤더를 삽입하여 네트워크 적응 계층의 패킷을 만들고 네트워크 인터페이스 계층에서 등록한 API를 이용하여 패킷을 전송한다. 그러나 네이밍 서비스를 이용하여 찾은 메시지의 목적지가 원격이 아니면 로컬 서비스를 한다. 로컬 서비스는 로컬의 어플리케이션간 메시지를 주고 받을 때 또는 요구를 할때 사용하며 네트워크 적응 계층에서 메시지를 네트워크를 통해 밖으로 보내지 않고 바로 스트리밍 계층으로 올림으로써 이루어진다. 스트리밍 계층으로 전달된 메시지는 서비스를 해석하고 최종 목적 어플리케이션으로 전달하기위해 어플리케이션 관리를 이용하여 어플리케이션을 찾고 메시지, 또는 데이터를 전달한다.

로컬 서비스가 아닌 원격 서비스의 경우에 네트워크 적응 계층의 패킷을 전달 받은 네트워크 인터페이스 계층의 네트워크 컴포넌트는 전달 받은 패킷의 크기를 자신이 담당하는 네트워크의 Maximum Transfer Unit (MTU)과 비교하여 패킷을

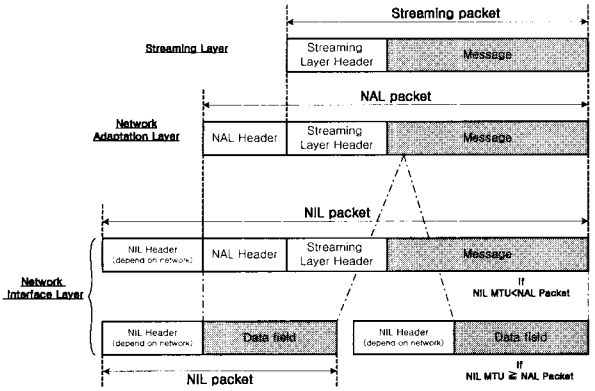


그림 10. 계층별 패킷타이제이션.
Fig. 10. Packetization of each layer.

분할하여 전송할 것인지 하나의 패킷으로 전송할 것인지 판단한다. 그 예로, IEEE1394 버스 상에 있는 노드로 메시지를 전송하기 위해서는 IEEE1394에서 사용하는 패킷 포맷으로 만들어 메시지를 전송하여야 하며, IEEE1394의 비동기 전송 모드에서 최대 패킷 크기는 2048바이트이다. 그리고 USB 버스 상에 있는 슬레이브 노드로 메시지를 전송하기 위해서는 USB에서 사용하는 패킷 포맷으로 만들어 메시지를 전송하여야 하며, 전송 가능한 최대 패킷 크기는 64바이트이다. 이것은 네트워크 적응 계층의 256바이트 크기의 메시지가 전송되어야 할 경우 IEEE1394로 전송한다면 하나의 패킷으로 메시지 전송이 가능하지만 USB로 전송 할 경우 약 5개의 패킷으로 분할되어야 한다는 것을 의미한다. 이러한 패킷의 생성 과정이 그림 10에 나타나 있다.

그림 11은 미들웨어 계층별로 원격 서비스와 로컬 서비스의 두 경우 메시지 전송에 대하여 설명하고 있다. 제안된 미들웨어는 스트리밍 계층을 통해 어플리케이션의 요구를 최초로 받아들인다. 스트리밍 계층은 어플리케이션의 요구에 대한 메시지 재 표현 과정을 거치기 전에 트랜잭션 관리를 하기위한 작업을 한다. 트랜잭션을 등록한 어플리케이션 요구에 대하여서만 마샬링을 통한 메시지 재 표현과정을 거치고 네트워크 적응 계층으로 메시지가 전달된다.

메시지는 네이밍 서비스를 통해 메시지의 목적지 레퍼런스, 즉 모듈 ID를 찾아내고 해당하는 목적지의 위치를 모듈 어드레싱을 통해 알아낸다. 이 과정에서 라우팅 테이블을 사용하게 되며 최종 목적지가 정해진 메시지는 네트워크 인터페이스 계층의 해당하는 네트워크에서 사용되는 패킷타입으로 다시 만들어 진다. 메시지의 크기가 해당하는 네트워크의 MTU보다 클 경우 분할되어 전송되며 그렇지 않을 경우 하나의 패킷으로 만들어져 전송된다. 수신의 과정은 전송의 역과정을 거치며, 라우팅이 필요한 메시지 일 경우 네트워크 적응 계층에서 라우팅하는 과정을 거치게 된다.

다음 장에서는 제안된 미들웨어의 구현과 그 성능을 측정함으로써 실제 퍼스널 로봇에 적용 가능한지를 검증한다.

IV. 구현 및 성능 측정

4장에서는 모듈 기반 퍼스널 로봇에서 개방형 이종 인터페이스를 지원하기 위해 제안된 미들웨어의 구현에 대해

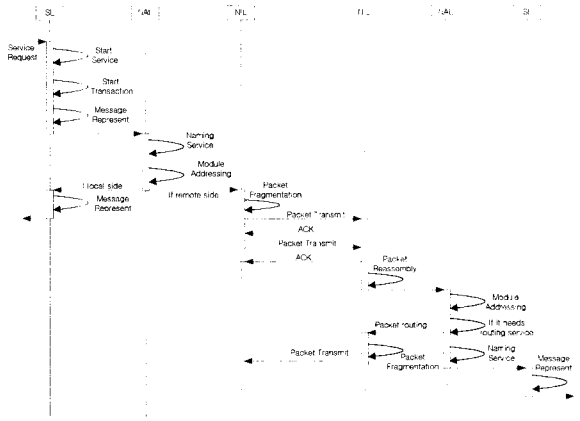


그림 11. 메시지 전송.
Fig. 11. Message transmission.

표 3. 모듈별 하드웨어 구성.
Table 3. Hardware configuration.

모듈 이름	모듈 ID	인터페이스	CPU
Brain	7112	Ethernet	Pentium 3 733MHz
Base	4437	Ethernet, IEEE1394	Pentium 3 733MHz
Arm	3208	IEEE1394, USB	Pentium 3 733MHz
Elbow	3571	USB ver1.1	80c196 20MHz
Hand	5376	USB ver1.1	80c196 20MHz

설명하고, 그 성능을 측정함으로써 퍼스널 로봇에 적용 가능한지 살펴본다.

1. 구현 환경

본 논문에서 제안한 미들웨어 구조가 서로 다른 종류의 네트워크로 구성된 이종 네트워크 분산 환경에서 적합하며, 동작 가능한지를 검증하기위해 IEEE1394, USB, Ethernet의 3가지 네트워크를 사용하였다. 사용된 테스트 모듈은 실제 로봇이 없는 관계로 산업용 보드를 이용하여 가상의 로봇 모듈을 설정하였다. 테스트에는 Brain 혹은 주 제어 모듈, Base 모듈, Arm 모듈, Elbow 모듈, Hand 모듈의 5개 모듈이 사용되었다. Brain 혹은 주 제어 모듈, Base 모듈, Arm 모듈은 Ethernet과 IEEE1394, USB가 장착된 산업용 컴퓨터를 사용하였으며 Elbow 모듈과 Hand 모듈은 USB 인터페이스를 장착하고 80c196컨트롤러를 사용하는 USB개발 킷을 사용하였다. 각 모듈에 대한 사양이 표 3에 나타나 있다.

그림 12는 실제 사용된 테스트 모듈들의 구성을 보여주고 있다. Arm 모듈에는 USB포트가 하나로 두개의 USB 개발 킷을 연결하기위해 USB Hub를 사용하였다. 각 모듈의 동작은 텔넷을 사용하여 모니터링하였다. 제안된 미들웨어 및 로봇 어플리케이션은 C를 이용하여 구현하였으며, Brain, Base, Arm 모듈의 OS는 구현상의 편의를 위해 공개 소스가 제공되는 Wow Linux 7.1(커널 버전 2.4.2)을 사용하였다. USB 인터페이

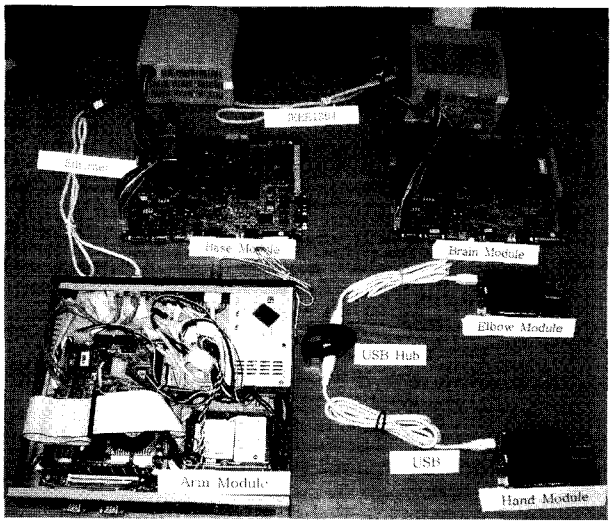


그림 12. 하드웨어 환경.
Fig. 12. Hardware Environment.

표 4. 미들웨어 테스트 목적.
Table 4. Aim of testing the proposed middleware.

구분	목적	구성
Case1	이종 네트워크 환경하에서 종단간 메시지 전송여부	Brain 모듈에서 Hand 모듈로 데이터 전송
Case2	이종 네트워크 환경하에서 네트워크별 응답시간	IEEE1394를 사용하는 모듈간 응답시간
		Ethernet을 사용하는 모듈간 응답시간
		USB를 사용하는 모듈간 응답시간
		IEEE1394, Ethernet을 사용하는 모듈간 응답시간
		IEEE1394, USB를 사용하는 모듈간 응답시간
		Ethernet, IEEE1394, USB를 사용하는 모듈간 응답시간
	사용된 메시지 크기	각 계층의 헤더를 포함한 메시지 크기 <64byte

스를 사용하는 Elbow, Hand 모듈의 미들웨어와 응용 프로그램도 C를 이용하여 구현하였다.

2. 검증 및 성능 측정

모듈간 통신이 이종 네트워크로 구성된 분산 환경 하에서 제안된 미들웨어를 통해 가능함을 보이기 위해 USB 인터페이스를 사용하는 Elbow모듈과 Hand모듈을 포함한 모든 모듈에 제안된 미들웨어를 구현 및 이식하였다. 이것은 실제로 구현된 미들웨어가 Hand 모듈과 같은 작은 컨트롤러에도 이식이 가능함을 보여주는 예로서 어떤 종류의 임베디드 시스템에도 적용 가능하다는 것을 의미한다. 그리고 구현된 미들웨어가 개방형 이종 인터페이스를 지원하는 것 이외에도 작은 컨트롤러로 구성된 모듈 시스템에 적용 가능하다는 또 다른 장점을 보여주기도 한다.

구성된 하드웨어 환경에서 라우터 모듈은 두 가지 이상의 이종 네트워크를 통해 원격 모듈의 존재를 확인 하였을 경우 자동적으로 설정되며, 그림 12에서는 Base 모듈과 Arm 모듈이 라우터 모듈로서 동작한다. 라우터 모듈은 메시지 라우팅 서비스를 수행하는데 Base 모듈의 경우 Brain모듈로부터 수신한 메시지를 Arm 모듈로 라우팅 하거나 Arm 모듈로부터 수신한 메시지를 Brain 모듈로 라우팅하게 된다. 이러한 테스트 환경을 이용한 테스트 목적과 그 구성은 표 4와 같다.

표 4에서 사용된 메시지의 크기가 스트리밍 계층, 네트워크 적응 계층, 네트워크 인터페이스 계층 그리고 각 네트워크별 물리계층의 헤더를 포함하여 64바이트 미만인 이유는 USB의 최대 패킷 크기가 64바이트로 Ethernet, IEEE1394, USB 중 가장 작기 때문이다. 만약 각 계층의 헤더를 포함하여 64바이트가 넘을 경우 USB측에서는 두개 이상의 패킷으로 분할되어 전송되어야 하기 때문에 동일한 조건에서 메시지를 전송할 수 없게 된다. Case1을 검증하기 전에 먼저 메시지가 전송되는 과정을 살펴보면 다음과 같다.

1) Brain모듈에 어플리케이션이 Hand모듈로 데이터를 전송 (64바이트 미만)
2) Brain모듈은 전송할 메시지의 목적지를 자신의 라우팅 테이블에서 검색 하여 Base모듈로 전송
3) Base모듈은 수신한 메시지의 최종 목적지가 자신이 아니기 때문에 메시지를 Arm모듈로 라우팅
4) Arm모듈은 수신한 메시지의 최종 목적지가 자신이 아니기 때문에 메시지를 Hand모듈로 라우팅
5) Hand모듈은 Arm모듈로부터 데이터를 수신

각 모듈의 미들웨어는 초기화 과정을 거치면서 자신의 라우팅 테이블을 작성한다. Arm 모듈의 라우팅 테이블이 그림 13에 나타나 있다.

라우팅 플래그는 종류가 다른 두개 이상의 네트워크를 통해 모듈이 연결되었음을 확인할 때 설정된다. Arm 모듈은 USB와 Ethernet을 통해 다른 모듈과 연결되었으므로 라우팅 플래그가 설정되었음을 볼 수 있다. Arm 모듈에서 Brain모듈로 패킷을 전송하기위해서는 Base모듈을 거쳐야 하는데 이때 Base 모듈로 전송하기 위한 네트워크 종류가 Ethernet으로 설정되어 있고, Base 모듈의 ID와 TCP 서버 포트 UDP 서버 포트가 설정되어 있는 모습을 볼 수 있다. 또한 Elbow 모듈로 전송하기위한 네트워크 종류가 USB이며 USB슬레이브 노드 ID가 설정되어 있는 것을 라우팅 테이블을 통해 알 수 있다.

Brain 모듈에 등록되어 있는 로봇 어플리케이션이 미들웨어를 통해 Hand모듈로 데이터를 전송하는 모습이 그림 14에 나타나 있다. 어플리케이션은 미들웨어의 스트리밍 계층에 등록한 후 그에 대한 승인으로 미들웨어의 정보를 저장하고 있으며, move(11, 2, 43)이라는 메시지를 Hand모듈로 전송하고 있다.

Brain모듈로부터 Base모듈로 전송된 패킷은 다시 Arm모듈로 전송되고 Arm 모듈에서는 다시 Hand모듈로 전송된다. Arm 모듈에서 Hand 모듈로 라우팅하는 모습이 그림 15에 나타나 있다. 라우팅 플래그가 설정된 Arm 모듈은 수신한

```
==========< NAL Local Information >==========
 This Module Name       = Arm
 This Module ID         = 3208
 This NAL routing Flag  = ENABLED

==========< NAL Routing Talbe >==========
 This NAL has 4 information of remote module(s)

 Remote Module Name     = Brain
 Remote Module ID       = 7112
 Connected Interface    = ETHERNET
 Interface Address      = 210.115.43.169, 8888,9999

 Remote Module Name     = Base
 Remote Module ID       = 4437
 Connected Interface    = ETHERNET
 Interface Address      = 210.115.43.169, 8888,9999

 Remote Module Name     = Elbow
 Remote Module ID       = 3571
 Connected Interface    = USB
 Interface Address      = 0

 Remote Module Name     = Hand
 Remote Module ID       = 5376
 Connected Interface    = USB
 Interface Address      = 1
==========
```

그림 13. Arm 모듈의 라우팅 테이블.
Fig. 13. Routing Table of Arm Module.

```
==========< Robot App1 Information >==========
 Middleware Name        = MMPR
 Middleware version     = 1.0
 Middleware PID         = 7952
 This Module Name       = Brain
 This Module ID         = 7112
==========

Sending 'move(11,2,43)' to Hand
==========
```

그림 14. Brain모듈에서 Hand 모듈로 메시지 전송.
Fig. 14. Message Transmission from Brain Module to Hand Module.

```
==========< NAL Local Information >==========
 This Module Name       = Arm
 This Module ID         = 3208
 This NAL routing Flag  = ENABLED
==========

 Now Pakcet Routing.. from Base to Hand
==========
```

그림 15. Arm 모듈에서 패킷 라우팅.
Fig. 15. Packet Routing at Arm Module.

패킷을 라우팅 테이블을 참조하여 Hand모듈로 라우팅하고 있다.

Hand모듈은 그림 16과 같이 Brain모듈로부터 Ethernet, IEEE 1394, USB 3가지의 네트워크를 거쳐 "move(11,2,43)"이라는 메시지를 성공적으로 수신하였다. 메시지 타입은 P2P로서 점대점 연결형 통신을 의미하며 소스 모듈의 ID는 7112로서 Brain으로부터 전송된 패킷임을 알 수 있다.

Case2에서는 Ethernet, IEEE1394, USB의 사용 여부 따라 응답시간을 측정하였다. 응답시간은 1,000,000번 반복하여 측정하였 제안된 미들웨어를 사용하여 모듈간 응답시간을 측정한 결과 그림17과 같이 평균응답시간이 1ms 이내의 값을 보였다. 으며 네트워크 별 응답시간 시간은 표 5와 같다.

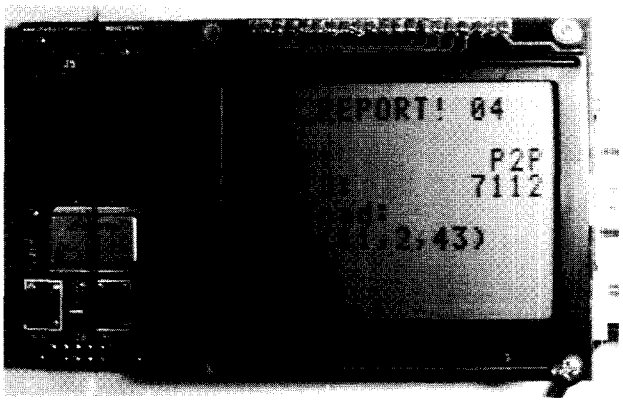


그림 16. Hand 모듈에서 메시지 수신.
Fig. 16. Message receiving at Hand Module.

표 5. 모듈간 응답시간(단위, ms).
Table 5. Response Time between Modules(Unit, ms).

모듈	사용된 네트워크	평균 응답시간
Base–Arm	IEEE1394	0.345
Brain–Base	Ethernet	0.420
Arm–Hand, Arm–Elbow	USB	1.000
Brain–Arm	IEEE1394, Ethernet	0.766
Base–Hand, Base-Elbow	IEEE1394, USB	1.346
Brain-Hand, Brain-Elbow	IEEE1394, Ethernet, USB	1.765

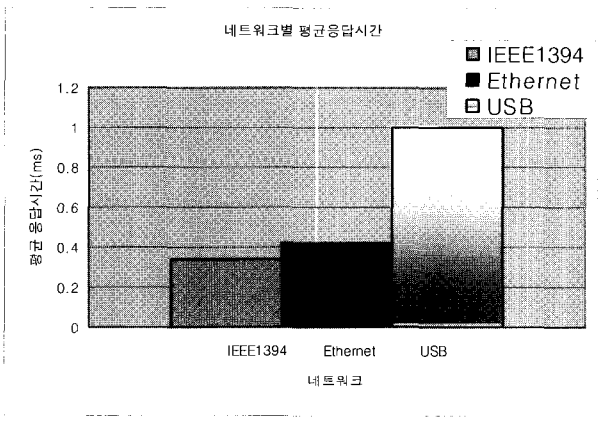


그림 17. 네트워크 별 평균 응답 시간.
Fig. 17. Average Response Time of Networks.

IEEE1394를 이용하여 전송할 경우에는 평균 0.345ms의 응답시간을 보였고, Ethernet을 이용하여 전송할 경우에는 평균 0.42ms의 응답시간을 보였다. 마지막으로 USB를 이용하여 전송할 경우에는 1ms의 응답시간 보였다.

이상과 같이 본 논문에서 제안한 미들웨어를 구현하고 그 성능을 측정하였다. 구현된 미들웨어는 IEEE1394, Ethernet, USB의 세가지 네트워크를 수용하였고, 3장에 명시된 미들웨어의 서비스를 모두 정상적으로 수행하였다. 미들웨어의 성

능분석에 있어서 평균 응답시간은 1ms 이내로 나타나 1ms의 응답시간을 요구하는 시스템에 사용가능하다는 것을 알 수 있다.

V. 결론

본 논문에서는 모듈기반 퍼스널 로봇에서 개방형 이종 인터페이스를 지원하기 위한 미들웨어 구조를 제안하고 구현하였다. 제안된 미들웨어는 어플리케이션에 미들웨어의 서비스를 제공하기위한 스트리밍 계층(Streaming Layer, SL), 종류가 다른 여러 네트워크를 수용하기 위한 네트워크 적응 계층(Network Adaptation Layer, NAL)과 네트워크 의존적인 기능을 처리하기위한 네트워크 인터페이스 계층 (Network Interface Layer, NIL)의 3 계층으로 구성되었다.

본 논문에서는 제안된 미들웨어를 구현하였으며, 모듈간 네트워크 인터페이스로는 IEEE1394, Ethernet, USB를 사용하였다. 이와 같은 이종 네트워크 환경에서, 종단 모듈간 메시지 전송이 성공적으로 이루어졌음을 보여줌으로써 제안된 미들웨어의 동작을 검증하였다. 또한 이종 네트워크 환경에서 미들웨어를 이용한 응답시간을 측정하고 분석함으로써 제안된 미들웨어가 실제 모듈기반 퍼스널 로봇에 적용 가능한지 살펴 보았다.

제안된 미들웨어는 제시된 미들웨어의 서비스를 모두 정상적으로 수행하였으며 1ms의 평균 응답시간을 요구하는 시스템에 사용가능하다는 것을 알 수 있었다. 특히 80c196을 사용한 작은 임베디드 시스템 모듈에 제안된 미들웨어를 구현 및 이식함으로써 다양한 모듈에 적용 가능하다는 것을 보였다. 이것은 제안된 미들웨어가 하드웨어와 독립적인 관계에 있다는 것을 잘 보여준다.

응용 개발자는 종류가 다른 여러 네트워크로 구성된 네트워크 환경에서 제안된 미들웨어를 사용함으로써 어떤 모듈이 어떤 네트워크로 연결되어 있는지, 그리고 네트워크로 데이터를 전송하기 위해서는 어떤 방법을 사용해야 하는지에 대하여 관여하지 않고 원하는 응용들을 쉽게 개발할 수 있다. 제안된 미들웨어는 모듈기반 퍼스널 로봇에서 종류가 다른 여러 네트워크 인터페이스를 동적으로 지원할 수 있는 구조를 갖추고 있으며 퍼스널 로봇에 필요한 교환성 및 확장성을 증가시키는데 크게 기여할 것으로 생각된다.

참고문헌

[1] T. Fukuta, R. Michlini, V. Potkonjak, S. Tzafestas, K. Valavanis, and M. Vukorbrativic, "How far away is "Artificial Man?", *IEEE Robotics & Automation Magazine.* pp. 66-73, Mar 2001.

[2] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation,* RA-2(1):14-23, 1996.

[3] T. Makelainen, J. Kaikkonen, H. Hakala, " Interfacing functional modules within mobile robots," *Intelligent Robots and Systems '91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on,* 3-5 Nov 1991.

[4] J. A. Fryer, G. T. McKee, P. S. Schenker, "Configuring robots from modules: and object oriented approach ", *Advanced Robotics, 1997. ICAR'97. Proceeding. 8th Internaltional Conference on,* 7-9 Jul 1997.

[5] H. Ishiguro, T. Kanda, K. Kimoto, T. Ishida, " A robot architecture based on situated modules" *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on.* Volume:3,199.

[6] R. Chatila, R. Ferraz de Camargo, "Open architecture design and inter-task/inter module communication for an autonomous mobile robot," *Intelligent Robots and Systems'90. 'Towards a New Frontier of Applications', Proceedings. IROS'90, IEEE International Workshop on,* 3-6 Jul 1990.

[7] IEEE standard for a High Performance Serial Bus"IEEE std 1394-1995, IEEE1394 std 1394a-2000".

[8] Universal Serial Bus Specification revision 1.1: September 23. 1998.

[9] CAN specification Part A and Part B.

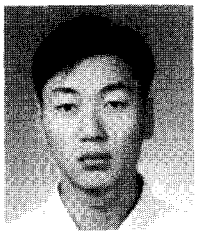
[10] Bluetooth SIG groups, Specification of the Bluetooth System, Ver1.1 Draft Oct 2000.

[11] RMI specification http://java.sun.com/products/jdk/rmi/index.html.

[12] The Common Objcet Request Broker: Architeture and Specification revision 2.3: Jun 1999.

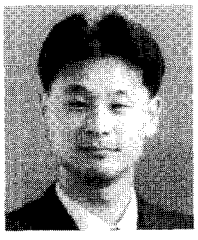
[13] COM and DCOM specification. http://www. microsoft. com/com/resources/specs.asp.

[14] William stallings, "High-speed netwroks and internets performance and quality of service" 2nd Edition. 2000.



윤 건

1977년 2월 13일생. 2001년 강원대 제어계측공학과 졸업. 동 대학원 제어계측공학과 석사(2003.8). 2003년~현재 LG산전 자동화 연구소 Advanced Platform 연구단. 관심분야는 분산 시스템, 임베디드 시스템 등.

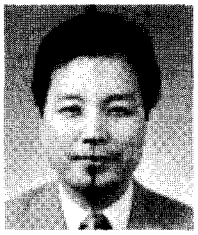


김 형 육

1973년 7월 16일생. 1999년 강원대 제어계측공학과 졸업. 동 대학원 제어계측공학과 석사(2001). 2001년~현재 동 대학원 제어계측공학과 박사과정. 관심분야는 실시간 스케줄링, 필드버스, 무선 데이 터 통신 등.

김 홍 석

1957년6월9일 생. 1980년 서울대 전기공학과 졸업. 동 대학원 제어계측공학과 석사(1983). 동 대학원 제어계측공학과 박사(1990). 1991년~현재 한국생산기술연구원 허브-로봇센터 제어지능연구팀 (수석연구원). 2001. 10~현재 산업자원부 차세대신기술개발사업: 퍼스널 로봇기반기술개발 과제 수행 중. 관심분야는 제어이론, 로보틱스, 실시간 통신 등.



박 홍 성

1961년 3월 16일생. 1983년 서울대 제어계측공학과 졸업. 동 대학원 제어계측공학과 석사 (1986). 동 대학원 제어계측공학과 박사(1992). 1992년~현재 강원대 전기전자정보통신공학부 교수. 관심분야는 실시간 네트워크, 무선 네트워크 등.