Linux와 TCP/IP를 이용한 분산 실시간 이동로봇 시스템 구현에 관한 연구

A Study on the Distributed Real-time Mobile Robot System using TCP/IP and Linux

김 주 민*, 김 홍 렬, 양 광 웅, 김 대 원
(Joo Min Kim, Hong Ryeol Kim, Kwang Woong Yang, and Dae Won Kim)

Abstract : An implementation scheme and some improvements are proposed to adopt public-licensed operating system, Linux and de-facto world-wide network standard, TCP/IP into the field of behavior-based autonomous mobile robots. To demonstrate the needs of scheme and the improvement, an analysis is performed on a server/client communication problem with real time Linux previously proposed, and another analysis is also performed on interactions among TCP/IP communications and the performance of Linux system using them. Implementation of behavior-based control architecture on real time Linux is proposed firstly. Revised task-scheduling schemes are proposed that can enhance the performance of server/client communication among local tasks on a Linux platform. A new method of TCP/IP packet flow handling is proposed that prioritizes TCP/IP software interrupts with aperiodic server mechanism as well. To evaluate the implementation scheme and the proposed improvements, performance enhancements are shown through some simulations.

Keywords : linux, TCP/IP, behavior, mobile robot, real time, server/client, packet flow

I. 서론

최근까지의 이동로봇 개발에 관한 연구는 대부분 특정한 용도 혹은 환경에서의 운영을 고려하여 이루어지고 있다. 이는 이동로봇이 그 운영에 있어 주위 환경에 영향을 많이 받고, 또한 어느 정도의 자율성이 요구되기 때문이다. 하지만 이러한 연구 동향은 이동 로봇의 적용에 제한을 가져오는 측면이 많기 때문에 이동 로봇의 시장 확대에 많은 어려움이 있는 실정이다.

이동로봇의 시장 확대를 위해서는 다양한 사용자의 요구 충족을 위해 다품종 소량 제작 시스템이 도입되어야 한다. 하지만 아직까지의 제조업체 특성은 다품종 소량 제작 시스템을 도입하기 위해서는 많은 투자가 선행되어야 하고 다품종 소량이라는 특성 상 제조 원가의 상승이 필연적일 수밖에 없다.

이러한 다품종 소량 제작 시스템의 문제점을 피할 수 있는 방법이 네트워크 시스템의 도입을 통한 분산화이다. 분산화를 통해 제작 업체는 자신들이 전문화된 이동로봇의 기능적 구성품을 모듈화 하여 소품종 다량 생산할 수 있고, 소비자는 이러한 모듈을 조합하여 자신이 원하는 로봇을 구성할 수 있다. 이렇게 분산화된 이동로봇 시스템을 가정용 로봇에 적용하려 한 연구가 최근에 수행되고 있다[1].

분산화의 가장 필수적인 요소는 상호 운용성이다. 서로 다른 제작업체에서 생산된 이동로봇의 기능적 모듈이 조합되어 하나의 로봇을 구성하기 위해서는 각각의 모듈 간의 상호 운용성이 확보되어야만 하고 이를 위해서는 표준화되고 개방된 방식의 인터페이스(interface)를 제공하는 모듈을 제작하여야 한다.

Linux의 개방성과 상업적인 매력은 최근에 Linux의 적용 분야를 급속히 확대시키는 역할을 하였는데, 최근에 Linux의 적용 분야로 각광 받고 있는 분야가 내장 시스템(embedded system)에의 적용이다. 특히 실시간 시스템에의 적용을 위해 실시간 Linux에 관한 연구가 꾸준히 이루어지고 있다[2].

본 논문에서는 기존의 실시간 Linux를 실시간이 요구되는 분산 이동로봇 시스템에 적용하였을 경우에 발생할 수 있는 문제점에 대해 논의하고 이를 개선할 수 있는 기법을 제안한다. 문제점 분석 대상이 되는 스케쥴링(scheduling) 전략은 정적 스케쥴링의 대표적인 방식인RMS(Rate Monotonic Scheduling)[11]를 사용 한다.

기존의 RMS는 운영 체계의 작업 단위인 태스크(task) 간의 서버/클라이언트(server/client) 통신 시에 발생하는 실질적인 우선순위 역전에 관한 고려가 이어지지 않았다. 기존의 RMS를 사용하게 되면 서비스를 요청하는 클라이언트 태스크의 우선순위에 상관없이 서비스를 실질적으로 수행하는 태스크인 서버 태스크의 우선순위에 따라 클라이언트 태스크의 실질적인 작업 완료가 지연되게 된다.

따라서 본 논문에서는 이를 개선하기 위해 클라이언트 태스크의 우선순위를 서버 태스크로 전달하여 서버 태스크의 수행 우선순위를 클라이언트 태스크 우선순위와 동일하게 만들어주는 우선순위 전달 기법을 제안하고 제안된 방식의 시간 특성을 모델링한다.

분산화의 필수적인 조건인 네트워크는 일반적으로 실시

* 책임저자(Corresponding Author)
논문접수 : 2002. 12. 30., 채택확정 : 2003. 7. 10.
김주민, 김홍렬 : 명지대학교 정보제어공학과
(goodmate@mju.ac.kr/hr.kim@carrier.co.kr)
양광웅 : 한국생산기술연구원(ygkgwg@kitech.re.kr)
김대원 : 명지대학교 정보공학과(dwkim@mju.ac.kr)

간성을 보장하기 어려운 요소를 가지고 있다. 이러한 네트워크의 실시간성을 위해 수많은 네트워크 프로토콜이 제안되었으나, 이런 실시간 네트워크의 적용 분야가 주로 산업계에 한정되어 있고 대부분의 네트워크 프로토콜이 영리단체에 의해 제정이 주도되어 네트워크 프로토콜의 표준화보다는 네트워크 프로토콜의 난립이라는 결과를 가져왔다[3]. 이동로봇의 경우, 분산화를 위해 CAN (Controller Area Network)[4]을 이용한 연구가 활발히 진행되어 왔으나 표준 네트워크 프로토콜로의 수용 여부는 아직 미지수이며, 특히 이동로봇이 일반 가정이나 산업 현장에서 사용될 때 외부의 다른 시스템 네트워크와 상호 운영상을 가지기 위해서는 표준화된 상위 계층의 네트워크 프로토콜을 지원하여야 한다.

하위 계층에 독립적인 상위 계층의 네트워크 프로토콜 중에 특히 TCP/IP의 경우는 인터넷의 급격한 발달과 함께 정보업계 전반의 사실 상 표준으로 인정받고 있다. 기존의 TCP/IP를 이용한 분산 시스템의 실시간성에 관한 연구는 대부분 네트워크 프로토콜의 개선을 통한 TCP/IP의 실시간성 보장에 관해 이루어 졌기 때문에 본 논문에서는 TCP/IP를 이용하였을 경우의 시스템 실시간성 보장에 관한 연구를 수행한다.

본 논문에서는 Linux 환경에서 TCP/IP 데이터를 처리해 주는 소프트웨어 인터럽트를 충분한 서비스를 제공하는 한도 내에서 제한하여 스케쥴링 전략에 의해 운영되는 태스크 들을 위한 수행 대역폭을 최대한 확보할 수 있는 기법을 제안한다.

본 논문의 2장에서는 관련된 기존의 연구 결과와 기존의 연구 결과가 갖는 문제점 및 개선점에 대해 기술하고, 3장에서는 이러한 문제점을 개선할 수 있는 기법을 제안한다. 4장에서는 분산화된 행동기반의 이동로봇을 대상으로 한 모의실험을 통해 본 논문에서 제안한 기법이 적용되었을 경우와 그렇지 않은 경우에 대해 성능 비교를 함으로써, 제안된 기법이 유용함을 입증한다. 마지막으로 5장에서 결론과 향후 연구 과제에 대해 기술한다.

II. 연구배경 및 문제점 고찰

1. 실시간 Linux

일반적인 표준 Linux는 기본적으로 POSIX 1003[5] 기반의 인터페이스를 갖는 운영체계로 알려져 있다. 하지만 실제로 POSIX의 운영체계 표준은 30 개 이상의의 개별적인 표준으로 구성되어 있고, 실제로 현재 대부분의 운영체계에서 지원하고 있는 표준 사양은 POSIX 1003.1a-OS Definition, 1003.1b-Real Time Extensions, 그리고 1003.1c-Threads 등이다[6]. 이러한 기본 표준 사양마저 모두 지원하는 운영체제는 많지 않은데, Solaris[7]가 상기의 표준을 모두 지원하는 좋은 예일 것이다.

Linux의 경우, POSIX 1003 표준 중에서 실시간 확장(real-time extension)을 위한 기본 세트(basic set)인 POSIX 1003.1b의 정밀한 타이머 분해능(timer resolution)과 실시간 처리 메시지 큐(message queue)를 지원하지 않는다[6].

표준 Linux 시스템의 소프트웨어 타이머는 주기적인 틱 인터럽트(tick interrupt)에 의해 트리거(trigger)가 이루어진다. 이러한 타이머의 트리거 주기는 10ms로 결정되어 있는데, 이러한 타이머의 트리거 주기는 실시간 시스템에 적용할 수 없을 정도의 지연(latency)을 갖게 한다. 타이머의 트리거 주기에 위한 지연을 극복하기 위해 틱 인터럽트의 주기를 높이는 것은 시스템에 심각한 오버헤드(overhead)를 발생시키므로, 비주기 인터럽트 발생기(aperiodic interrupt source)를 이용한 타이머의 트리거에 관한 연구가 수행된 바 있다[8]. 실제로 현재까지 10ms 이상의 주기적인 틱 인터럽트를 보장할 수 있는 강력한 시스템은 Compaq의 Alpha에 포팅(porting)한 Linux 정도이다.

트리거 주기는 실시간성의 시간적인 분석을 수행할 경우, 연산 시간 및 주기의 최소 시간 단위가 된다. 태스크의 연산 시간 및 주기는 트리거 주기의 배수로 표현되게 된다.

본 논문에서는 타이머 트리거에 대한 세부적인 내용은 다루지 않으며, 현재는 여러 가지 트리거 방식에 대한 연구가 진행되고 있기 때문에 시간 특성 분석의 최소 단위를 변수로 정의하여 사용한다.

실시간 운영체계에서 스케쥴링 전략은 다중 태스크 시스템에서 실시간성을 보장하는 핵심 요소이며, 지금까지 이러한 스케쥴링 전략에 관한 연구가 다수 이루어져 왔다. 그 중에서도 RMS는 대표적인 정적 스케쥴링 전략으로서, 최소 주기를 갖는 태스크가 최고의 우선순위를 갖는 스케쥴링 전략이다. RMS를 이용할 경우 태스크들의 주기는 오프라인(offline) 상에서 사전에 실시간 가능성 검사를 통해 고정적으로 결정된다[11]. 따라서 RMS는 온라인(online) 상에서 어떤 적응성도 기대할 수 없고 자원 이용율이 저하된다는 단점이 있으나, 구현이 용이하고 스케쥴링 자체에 소요되는 연산 시간이 적기 때문에 대부분의 실시간 Linux에서 기본 스케쥴링 전략으로 제공된다.

실시간 동기화의 필요성은 태스크 간의 자원 공유에 의해 발생한다. 이러한 자원 공유의 문제점을 해결하는 대표적인 기법이 우선순위 상속 기법이다[9]. 실시간성을 제공하는 대부분의 운영체계는 우선순위 상속 기법과 같은 실시간 동기화 기법을 제공한다. 이러한 실시간 동기화 기법을 통해 높은 우선순위를 갖는 태스크가 낮은 우선순위를 갖는 태스크에 의해 그 수행이 지연되거나 어떠한 태스크도 수행을 완료할 수 없는 데드락(deadlock) 현상을 방지할 수 있다.

수행 자체를 지연 시키는 것은 아니지만 태스크의 처리 작업 지연을 일으키는 또 하나의 요소는 태스크 간의 통신이다. Linux는 태스크 간의 통신을 위해 파이프라인(pipeline), 메시지 큐(message queue) 등의 여러 가지 메커니즘을 제공한다[10]. 그 중에서도 메시지 큐 방식은 Linux에서 제공하는 또 다른 태스크 간 통신 방식인 파이프라인과 비교할 때 메시지를 보내고 받는 프로세스에 모두 독립적으로 존재한다는 장점이 있으며 이것을 통해 동기화의 문제를 최소화해 준다.

일반적으로 태스크 간 서버/클라이언트(server/client) 통신의 경우가 앞에서 기술한 태스크의 처리 작업 지연을 일으키는 대표적인 경우이다.

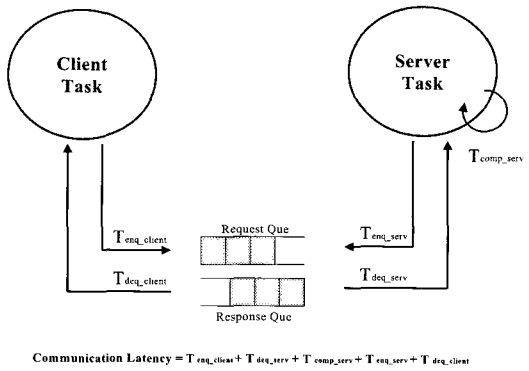

그림 1. 다중 태스크 환경의 서버/클라이언트 통신처리시간 지연요소 모델.

Fig. 1. The server/client communication latency model in multitask systems.

Linux와 같은 다중 태스크 환경에서 서버/클라이언트 통신을 사용할 경우의 처리시간 지연 요소(communication latency) 모델은 그림 1과 같다.

그림에서 보는 바와 같이 처리시간 지연 요소는 클라이언트 태스크가 메시지 큐에 요청을 삽입하는 데 소요되는 시간(Tenq_client), 서버가 메시지 큐로 전달된 요청을 인출하는데 소요되는 시간(Tdeq_serv), 전달된 요청에 따른 작업을 수행하는데 소요되는 시간(Tcomp_serv), 처리된 응답을 다시 메시지 큐에 삽입하는데 소요되는 시간(Tenq_serv), 그리고 최종적으로 클라이언트가 전달된 응답을 인출하는데 소요되는 시간(Tdeq_client)으로 구성된다.

그림 1과 같은 지연 시간은 클라이언트 태스크의 우선순위가 낮고 서버 태스크의 우선순위가 높을 때 극대화된다. 우선순위 방식의 선점형 멀티 태스크 환경에서의 그림 1의 지연 시간 요소는 그림 2와 같이 발생한다. 그림 2는 RMS 전략을 사용하여 선점형 스케쥴링을 했을 때의 경우이다.

그림 2에서 서버 태스크는 3가지 태스크 중 수행 주기가 가장 길므로 RMS 전략에 의하면 가장 낮은 우선순위를 갖게 되고, 이로 인해 요청 인출 시간과 요청 작업 처리시간이 지연 되어 전체 서버/클라이언트 통신의 지연은 증가하게 된다. 반면에 가장 짧은 수행 주기를 갖는 클라이언트 태스크는 서버 태스크의 응답 지연으로 인해 수행은 주기적으로 지속된다 할지라도 실질적인 작업은 지연되게 되어 실질적인 우선순위 역전이 발생한다.

우선순위를 설정한 서버/클라이언트 통신에 관한 연구로는 분산 객체 환경 프레임워크(framework)인 CORBA (Common Object Request Broker Architecture)의 실시간 시스템 적용에 관한 연구를 통해 수행된 바 있다[12].

본 논문에서는 CORBA의 경우와 유사하게 태스크 간의 메시지 큐 통신에 있어서 높은 우선순위를 갖는 클라이언트 태스크가 낮은 우선순위를 갖는 서버 태스크로 전달(hand-off)되어 메시지 처리에 의한 지연 시간을 최소화하는 기법을 제안한다.

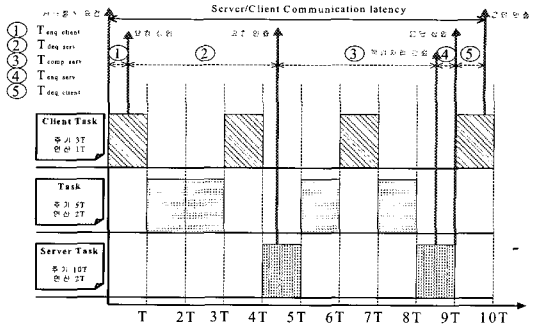

그림 2. RMS 전략에 의한 통신처리시간 지연의 발생.

Fig. 2. Occurrence of the communication latency with RMS strategy.

2. 실시간성을 보장하는 TCP/IP

개방형 프로토콜 스택(open protocol stack)의 하나인 TCP/IP는 원래 범용적인 목적으로 개발되었기 때문에 실시간 성에 대한 보장을 하지 못하고 있고, 이러한 문제점을 해결하기 위해 다양한 연구가 이루어지고 있다.

RTP/RTCP에[13] 관한 연구는 이러한 연구 중 하나이며, 또한 TCP/IP에서 네트워크 계층별 지연시간의 수학적 모델링과 이를 바탕으로 주어진 메시지들의 최악의 응답시간을 통한 실시간성 보장 여부를 분석한 연구가 이루어진 바 있다[14].

위와 같은 기존 연구가 갖는 문제점은 통신 프로토콜이 탑재되고 운영되는 시스템과의 상호 운영에 있어 서로의 실시간성에 미치는 영향에 대한 고려가 이루어지지 않았다는 점이다.

그림 3은 표준 Linux 커널(kernel)에서 TCP/IP 패킷 경로에 대한 개념도이다[15]. Linux 환경에서 TCP/IP 패킷은 NIC(Network Interface Card)의 하드웨어 인터럽트(interrupt)를 통해 바이트 단위의 복사, 혹은 DMA (Direct Memory Access)를 통해 메모리 상에 저장된 후 소프트넷(Softnet)으로 알려진 입력 큐(que)에 저장된다. 입력 큐에 저장된 패킷은 소프트웨어 인터럽트에 의해 큐에서 인출되어 통신 3계층인 IP 계층으로 전달되어 태스크의 핸들링을 기다린다. 이 때 인출 없이 대기할 수 있는 큐의 길이는 최대 300개로, 이 큐가 최대 크기가 되면 이후에 수신되는 메시지는 손실되게 되며, 다시 메시지 수신이 가능해지는 것은 입력 큐의 크기가 0이 된 이후이다. 따라서 소프트웨어 인터럽트 처리는 큐의 길이가 300이 되기 전에 이루어져야 한다. 물론 TCP 프로토콜의 경우 3계층에서의 전송 확실성을 보장하지만, 큐 크기의 제한으로 인한 메시지의 손실과 재 전송은 실시간을 방해하는 중대한 요소가 될 수 있다.

뿐만 아니라 이러한 소프트웨어 인터럽트가 자주 일어나는 경우 TCP/IP 통신과 무관한 태스크의 수행 시간 지연의 심각한 요인이 되기도 한다. Linux 환경에서 이러한 소프트웨어 인터럽트 처리는 비주기적으로 발생하고, 특히 이러한 비주기적 태스크의 수행은 오프라인 상에서 수행 주기가 결정되는 RMS와 같은 정적 스케쥴링 전략의 실시간성 보장에 장해가 된다[19].

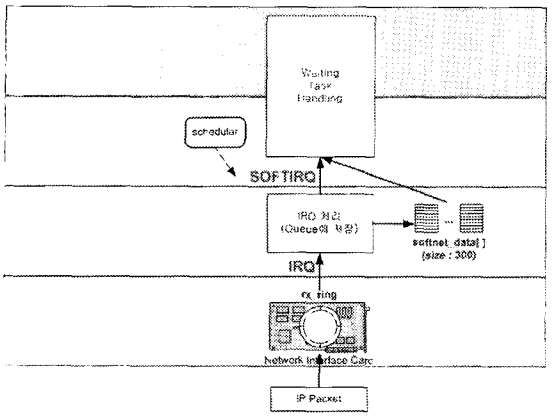

그림 3. Linux 커널에서의 패킷 경로.
Fig. 3. The packet path in Linux kernel.

따라서 본 논문에서는 정적 스케쥴링 전략의 사용 시에도 비주기적인 태스크의 발생에 실시간성을 보장할 수 있는 대표적인 방식인 비주기 서버(aperiodic server) 방식[19]을 이용하여 TCP/IP 데이터의 손실 없이 RMS를 따르는 태스크의 실시간성을 최대한 보장할 수 있는 기법을 제안한다.

3. 이동로봇의 제어프레임워크

80년대 중반, Brooks는 행동단위로서 로봇을 제어하는 행동언어(behavior language)에 대한 개념을 발표하였고, 이후 Connel과 Maes에 의해 행동언어는 로봇 언어에 새로운 패러다임을 열었다. 로봇제어에 대한 전통적인 접근 방법이 수직적이라면 행동언어는 수평적인 제어구조를 따르고 있다[16][17]. 행동언어는 행동 간의 독립성이 보장되는 최소 단위로 구성되며, 이는 병렬적이며 동시적인 처리가 가능하게 하는 기반이 된다. 이러한 행동기반언어를 사용함으로써 로봇은 새로운 지능을 가지게 되었고, 이동로봇의 제어 프레임워크에 대한 다양한 접근과 연구가 진행되어 왔다.

그러나 대부분의 기본 연구는 추상적인 제어 프레임워크의 구성에만 치우쳐 있고 실질적인 구현 시에 발생하는 문제점의 분석 및 해결 방법에 관한 연구는 거의 이루어지지 않았다.

따라서 본 논문에서는 이러한 행동 기반 제어 프레임워크를 Linux와 TCP/IP 기반의 분산 환경에 적용하는 방안을 제안한다. 본 논문에서를 예시를 들어 사용할 행동 기반의 제어 구조는 그림 4와 같다.

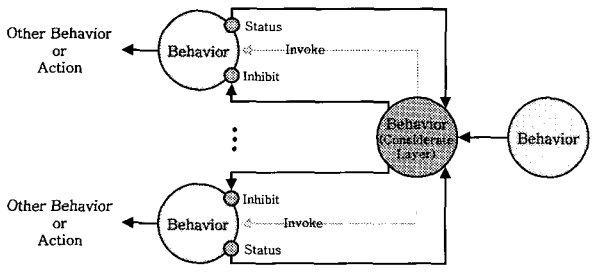

그림 4. 이동로봇의 제어 구조.
Fig. 4. A control architecture of a mobile robot.

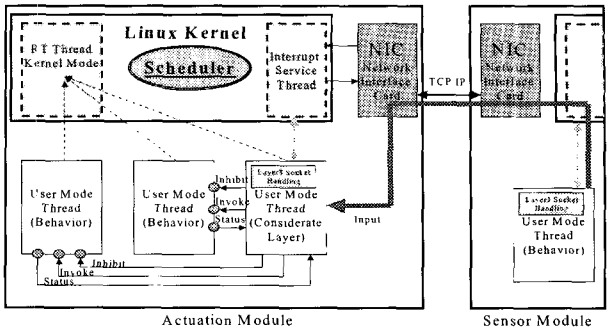

그림 5. Linux와 TCP/IP를 이용하여 분산화된 제어 프레임워크.
Fig. 5. The distributed control framework using Linux and TCP/IP.

그림 4에서 이동 로봇 전체의 제어 정책을 수립하는 행동 단위(considerate layer)는 주위 환경 상태를 제공하는 행동 단위로부터 입력을 받아 주위 환경 상태에 적합한 제어용 행동 단위를 기동해(invoke) 주고 그렇지 않은 행동 단위를 중지(inhibit) 시키는 역할을 수행한다.

III. Linux와 TCP/IP를 이용한 분산 실시간 이동로봇 시스템 구현

1. 분산화된 이동로봇 제어프레임워크의 구현

이동로봇 제어 프레임워크의 분산화는 행동 단위의 분산화를 의미한다. 본 논문에서는 네트워크로 분산화된 각각의 행동 단위를 Linux가 제공하는 태스크 단위인 쓰레드(thread)로 구현하는 것을 제안한다. 일반적으로 쓰레드 기반의 다중 태스크 환경이 프로세스(process) 기반의 다중 태스크 환경에 비해 콘텍스트 스위칭 시간(context switching time)을 최소화 할 수 있는 이유로 인해 실시간 시스템에 더 적합한 것으로 알려져 있다[15].

분산화된 이동로봇 행동 단위인 쓰레드는 실시간 처리를 위해 커널 모드 쓰레드로 매핑(mapping)되어 실행되며 이러한 매핑은 스케쥴러에 의해 이루어진다. 이동 로봇 제어 프레임워크의 모든 행동 단위는 그림 5에서와 같이 사용자 모드 쓰레드로 구현되며 스케줄러의 우선순위 선정 방식에 의해 실시간성을 제공하는 커널 모드로 매핑되는 방식을 사용한다. 즉 행동 단위 쓰레드는 커널 모드에서만 동작하게 된다. 이러한 방식은 기존의 Solaris[7]의 실시간 태스크 처리 방식과 동일하다.

본 논문에서는 분산화된 쓰레드 간의 통신 프로토콜로서 TCP/IP를 제안한다. TCP/IP를 사용함으로써 하위 2계층 통신에 독립적인 프로토콜 스택을 구성할 수 있으며, 컴퓨터 등의 범용적인 장비와의 손쉬운 통신 기능을 제공할 수 있다.

그림 4에서 예를 든 제어 프레임워크의 구성을 센서의 기능을 수행하는 모듈(sensor module)과 동작 기능을 수행하는 모듈(actuation module)로 분산화 하고, Linux와 TCP/IP를 이용하여 구성한 제어 프레임워크는 그림 5와 같다. 그림 5에서 동작 기능을 수행하는 모듈 내부 쓰레드 간의 통신에

는 메시지 큐를 사용한다.

그림 5와 같은 방식으로 이동 로봇 시스템을 구현을 하여 실시간 동작을 보장하기 위해서는 II 장에서 기술 했던 바와 같이 2 가지 문제점에 대한 개선이 이루어 져야 한다.

첫째, 동작 기능 내부의 제어 정책을 수립하는 행동 단위가 제어 정책을 변경할 경우, 즉 특정 행동 단위를 중지 시키고 다른 행동 단위를 기동시킬 경우에, 발생할 수 있는 전환 시간(transition time)을 최소화하기 위하여 중지 혹은 기동을 요청 받은 행동 단위는 가급적 빠른 응답 시간을 가져야 한다. 전환 시간이 큰 경우에는 전체 미션(mission) 완수 시간이 비례하여 증가하게 된다.

둘째, 센서 모듈로부터의 주위 환경 정보 전송의 유실을 막고, 통신으로 인한 동작 쓰레드의 지연을 최소화하기 위해서는 먼저 Linux의 기본 패킷 경로를 감안하여 패킷의 유실이 일어나지 않음을 보장하여야 한다. 아울러 전송 받은 패킷을 3계층으로 전송하는 경우 발생하는 소프트웨어 인터럽트에 의한 다른 쓰레드의 동작 지연이 최소화 될 수 있도록 소프트웨어 인터럽트의 스케쥴링 기법이 개선되어야 한다.

2. 우선순위 메시지 큐 통신

II장에서 기술 했던 바와 같이, 서버/클라이언트와 같이 요청/응답이 수행되는 통신의 경우에는 높은 우선순위의 클라이언트 태스크가 낮은 우선순위의 서버 태스크의 응답 지연으로 인해 수행은 주기적으로 지속된다 할지라도 실질적인 작업 지연이 발생하게 된다.

따라서 본 논문에서는 높은 우선순위의 태스크가 낮은 우선순위의 태스크에 메시지 큐를 이용하여 작업 처리 요청을 하는 경우, 높은 우선순위를 갖는 클라이언트 태스크의 우선순위가 낮은 우선순위를 갖는 서버 태스크로 전달되어 메시지 처리에 의한 지연 시간을 최소화하는 기법을 제안한다.

그림6에서 보는 바와 같이 우선순위 전달기법이 사용되었을 경우에는 우선순위가 가장 낮은 서버 태스크가 우선순위가 가장 높은 태스크인 클라이언트 태스크의 우선순위

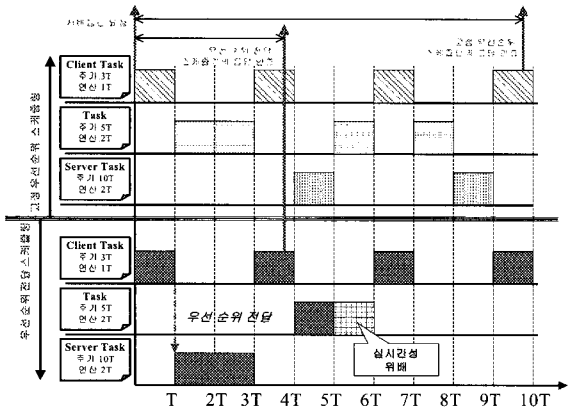

그림 6. 우선순위 전달기법을 사용한 경우의 지연 시간 변경.

Fig. 6. Change of the communication latency using priority hand-off protocol.

를 전달받아 수행되므로 중간 우선순위를 갖는 태스크에 의해 선점되지 않고 수행된다. 그림 6에서 주기 및 연산 시간을 표현하는 변수 T는 Linux의 타이머 트리거 주기를 나타낸다.

고정적인 우선순위를 갖는 RMS 전략에 있어 식 (1)의 최악 응답 시간이 주어질 경우 식 (2)의 조건을 만족하는 태스크는 실시간성을 만족하는 것으로 판단한다[11].

$$W_{task}(t) = \sum_{j=1}^{task} C_j \lceil t/T_j \rceil \tag{1}$$

여기서, $W_{task}(t)$는 n개의 태스크가 존재할 경우 우선순위가 가장 높은 태스크 1과 우선순위가 가장 낮은 태스크 n 사이의 우선순위를 갖는 특정 태스크의 최악 응답 시간이며, C_j는 태스크 j의 연산 소요 시간, T_j는 태스크 j의 주기, 그리고 t는 시간을 나타낸다.

$$S_{task} = \{kT_j \mid j = 1,.....,task; k = 1,.....\lfloor T_{task}/T_j \rfloor\} \text{ 에서,}$$
$$Min(t \in S_{task}) W_{task}(t)/t \le 1 \tag{2}$$

여기서, S_{task} 는 임의의 번째 태스크의 스케쥴링 포인트(scheduling point)이다. 그림 6의 고정 우선순위를 갖는 스케쥴링 방법의 경우에 중간 우선순위를 갖는 태스크의 실시간 보장성을 위의 (1),(2)를 이용하여 검증하면 다음과 같다.

중간 우선순위를 갖는 태스크 보다 높은 우선순위를 갖는 태스크는 클라이언트 태스크 밖에 없으므로 S_{task}={3T, 5T}가 되고, 다음의 조건식이 참이 되면 실시간성을 보장할 수 있게 된다.

$C_{\text{클라이언트 태스크}} + C_{\text{중간 우선순위의 태스크}} < 3T$

$: T + 2T < 3T$ 혹은,

$2C_{\text{클라이언트 태스크}} + C_{\text{중간 우선순위의 태스크}} < 5T$

$: 2T + 2T < 5T$

위의 조건은 참이므로 고정 우선순위 방식의 스케쥴링 방법을 사용하면, 그림 6의 위에서 보는 바와 같이 위의 어떠한 조건에서도 실시간성을 만족함을 확인할 수 있다.

하지만 우선순위 전달기법을 사용하게 되면 그림 6의 아래에서 보는 바와 같이 식(1)을 이용하여 모델링된 최악 지연 시간을 갖는 태스크 집합이 식 (2)를 만족한다 할지라도 실제로는 서버/클라이언트 통신과 무관한 태스크의 실시간성을 위배할 수 있다는 문제점을 갖는다. 실제로 그림 6의 중간 우선순위 태스크는 고정 우선순위가 자신 보다 낮은 서버 태스크에 의해 선점됨으로써 실시간성을 위배하게 된다.

따라서 본 논문에서는 자신 보다 우선순위가 낮은 태스크에 의한 수행 선점을 고려하여 최악 응답시간을 (3)과 같이 모델링한다.

$$W_{task}(t) = \sum_{j=1}^{n} C_j \lceil t/T_j \rceil \tag{3}$$

즉, 우선순위 전달 방식의 경우에는 낮은 우선순위의 태

스크에 의해서도 선점을 당할 수 있기 때문에 모든 우선순위 태스크에 의한 선점을 고려하여 최악의 응답 시간을 구한다.

(3)을 이용하게 되면 다음의 조건식이 참일 경우에 실시간성을 보장할 수 있게 된다.

$C_{\text{클라이언트 태스크}} + C_{\text{중간 우선순위의 태스크}} < 3T$

$: T + 2T + 2T < 3T$ 혹은,

$2C_{\text{클라이언트 태스크}} + C_{\text{중간 우선순위의 태스크}} < 5T$

$: 2T + 2T + 2T < 5T$

위의 조건식은 거짓이므로 고정 우선순위 전달 방식의 스케쥴링 방법을 사용하면 위의 어떠한 조건에서도 실시간성을 만족할 수 없음을 확인할 수 있고, 그 결과가 그림 6의 아래에서 우선순위 전달 방식을 사용한 결과와 일치함을 알 수 있다.

3. 실시간 시스템에서의 TCP/IP 통신

본 논문에서는 Linux의 소프트웨어 인터럽트 우선순위를 인터럽트가 제공하는 서비스를 사용하는 쓰레드 중 가장 높은 우선순위 보다 한 단계 높은 우선순위를 갖는 LynxOS[18]와 유사한 방식을 제안한다. 제안된 방식을 이용하면 TCP/IP 패킷 처리시간 지연을 결정하는 요소인 서비스를 사용하는 쓰레드에 충분한 서비스를 제공하면서 빈번한 소프트웨어 인터럽트를 방지할 수 있다.

LynxOS의 경우에는 소프트웨어 인터럽트 서비스 쓰레드가 서비스를 제공하는 쓰레드에 비해 항상 한 단계 높은 우선순위를 갖지만, 본 논문에서 제안한 방식은 서비스 쓰레드의 우선순위보다는 높지만 얼마나 높게 설정할 지는 대기 큐의 크기에 따라 온라인 상에서 결정하여 패킷의 손실을 미연에 방지하게 된다. 이렇게 비주기적인 태스크에 대해 온라인 상에서 우선순위를 변화시키는 대표적인 방식이 비주기 서버이다[19].

본 논문에서는 기존에 제안된 비주기 서버를 응용하여 TCP/IP 패킷 처리 태스크의 우선순위 선정을 다음 (4)와 같이 수행한다.

$$T_{service\_interrupt} = (1 - Q_{current}/300)T_{serviced\_task} \quad (4)$$

여기서, $T_{service_interrupt}$는 우선순위 선정을 위해서 비주기 태스크인 소프트웨어 인터럽트의 가상 주기이며, $Q_{current}$는 현재 대기 큐의 길이, $T_{serviced_task}$는 서비스를 제공 받는 태스크의 주기를 의미한다.

식 (4)는 TCP/IP 소프트웨어 인터럽트 주기가 서비스를 제공하는 태스크의 주기에 비례하며 또한 현재 TCP/IP 대기 큐에 대기 중인 데이터의 양에 반비례함을 의미한다.

비주기 처리를 위한 서버는 항상 다음의 조건을 만족하도록 설정되어야 한다[19].

$$U_{periodic\_task} + U_{Aperiodic\_task} \le 1 \quad (5)$$

여기서, $U_{periodic_task}$는 주기적인 태스크의 이용율, $U_{aperiodic_task}$는 비주기적인 태스크의 이용율을 의미한다.

그림 7은 중간 우선순위의 태스크 2에 서비스를 제공하는 소프트웨어 인터럽트 우선순위 선정 방식의 한 예를 보여준다. 그림에서 보는 바와 같이 소프트웨어 인터럽트 처리가 요청된 이 후 큐의 크기를 검사하여 큐의 크기가 10% 정도로 아직 여유가 있는 경우에는 최상위 우선순위 태스크가 먼저 동작하고, 50% 정도 육박한 경우에는 최상위 우선순위보다 더 높은 우선순위를 갖고 소프트웨어 인터럽트가 먼저 동작하게 된다.

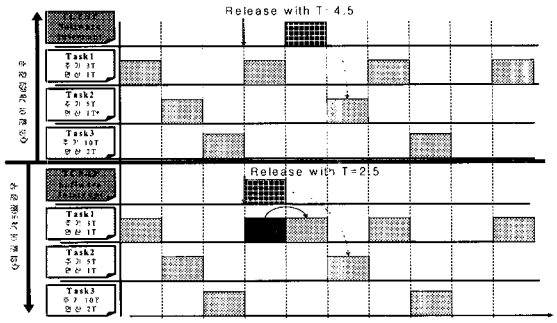

그림 7. 대기 큐의 길이에 따른 소프트웨어 인터럽트의 우선순위 선정.

Fig. 7. Determination of priority of a software interrupt based on the length of wait que.

IV. 모의실험

본 논문에서 제안하는 2가지 개선 방안에 대한 유용성을 입증하기 위해 모의실험을 수행한다. 본 논문의 모의실험에서 사용되는 태스크 집합은 기존 표준 Linux의 소프트웨어 타이머 인터럽트 주기인 10ms를 기준으로 하여 연산 수행 시간 및 수행 주기를 10ms의 배수로 표현한다.

1. 우선순위 메시지 큐 방식의 성능 평가

본 논문에서는 먼저 행동 기반 이동 로봇의 태스크 전환에 소요되는 시간을 측정하는 모의실험을 수행하여 기존의 방식에 비해 태스크 전환 시간과 비례하는 전체 미션 완수에 소요 되는 시간이 단축됨으로써 이동 로봇의 효율적인 운영이 가능함을 입증하고자 한다.

본 논문에서 모의실험을 수행하는 이동로봇의 미션은 작업 환경 내의 모든 장소에서 특정한 목표물을 찾아 수거하는 작업으로 가정한다..

표 1. 모의실험 태스크의 정의

Table 1. Definition of simulation task

태스크	주기[ms]	연산시간[ms]
위치 감지	50	10
제어 정책	60	10
자율 주행	70	15
목표물 수거	80	30

미션 수행은 그림 4와 같이 제어 정책을 수립하는 행동 단위의 제어에 의한 자율 주행 태스크와 목표물 수거 태스크의 조건적인 순차 수행으로 이루어진다. 이동 로봇은 자신의 위치 감지와 제어 정책의 수립을 항시 수행하면서, 작업 영역을 자율 주행하다가 수거하고자 하는 목표물이 감지되면 목표물을 수거한다.

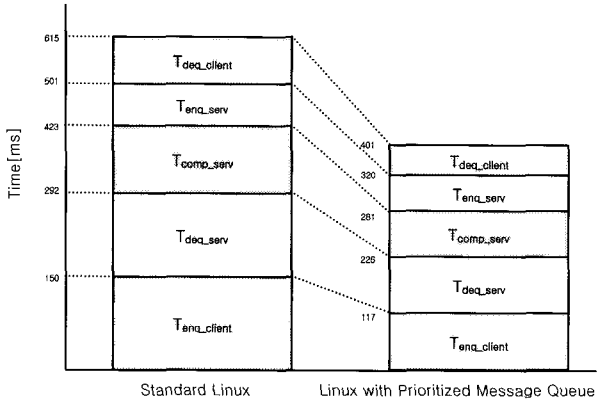

그림 8. 태스크 전환 지연시간 시험 결과.

Fig. 8. Test result of task transition delays.

제어 정책을 수립하는 태스크는 외부의 감지 센서로부터 수신되는 목표물 감지 신호를 입력 받아 목표물이 감지되면 수행되고 있는 자율 주행 태스크를 중지시키고 목표물 수거 태스크를 기동시킨다. 반대로 수거가 완료되어 목표물이 감지되지 않으면 목표물 수거 태스크를 중지시키고 자율 주행 태스크를 다시 기동 시킨다. 하나의 태스크 중지 후에는 상태(status) 신호를 통해 중지가 완료 됐음을 확인하고 다른 태스크의 기동을 수행하여 태스크 중복 수행으로 인한 미션 실패를 방지한다.

모의실험에서 사용하는 태스크의 집합은 표 1과 같이 정의한다. 표 1에서 위치 감지(localization) 태스크는 이동 로봇 자신의 현재 위치를 확인하기 위해 상시 수행하는 태스크이다.

그림 8은 표 1의 태스크 집합을 표준 Linux에서 기존의 RMS 스케쥴링 방식을 사용하였을 때와 제안된 우선순위 전달 방식을 사용하였을 때에, 상기의 미션을 수행하였을 경우에 발생하는 태스크 전환에 소요되는 지연 시간에 대한 모의실험 결과이다. 그림 8의 지연시간은 주기 10000ms 마다 태스크 전환이 이루어지는 환경을 가정할 때 10번의 태스크 전환이 이루지는 미션 수행과정에서 태스크 전환 지연 시간의 합을 나타낸다.

그림 8에서 발생하는 태스크 전환 지연시간은 미션 완수 소요시간에 영향을 미치게 된다. 태스크 전환 지연 시간은 목표물의 발견 시에 목표물 수거를 위해 발생하므로 목표물의 개수에 비례하여 증가하게 된다.

그림 8의 결과에서 우선순위 메세지 전달 방식을 사용하게 되면 표준 Linux에서 기존의 RMS만을 사용하는 것에 비해 태스크 전환 시감이 감소함으로써 이에 비례하는 미션 완수 소요 시간도 감소하게 됨을 알 수 있다.

2. 비주기 서버를 이용한 TCP/IP 소프트웨어 우선순위 선정 방식의 성능 평가

두 번째로 본 논문에서는 제안된 비주기 서버를 이용한 TCP/IP 소프트웨어 인터럽트 우선순위 선정 방식이 서비스를 제공하는 한도 내에서 최소한 수행되어 스케쥴링 전략에 따라 수행되는 다른 태스크를 위한 수행 대역폭을 최대한 확보할 수 있음을 입증 한다.

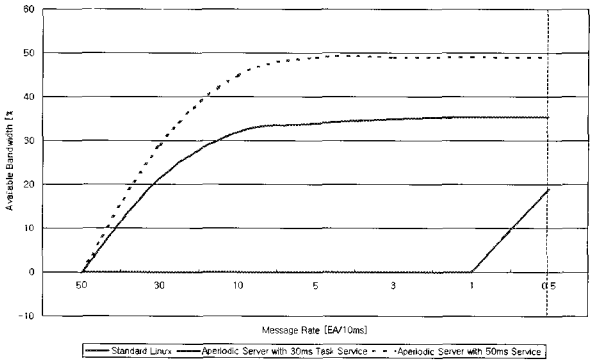

그림 9. TCP/IP 메시지 패킷 전송률에 따른 태스크 대역폭 변화 시험 결과.

Fig. 9. The variation of task bandwidth according to TCP/IP packet flow rate.

유용성을 입증하기 위해 TCP/IP 메시지 패킷이 주기적으로 전송되는 것을 가정하여 모의실험을 수행한다. 여기서, 스케쥴링 방식은 RMS 방식을 사용하여 전체 이용율이 69%를 넘게 되면 스케쥴링이 불가능 한 것으로 판단하였다[11].

그림 9는 주기적으로 수신되는 메시지 패킷의 양에 따라 기존의 표준 Linux에서 RMS를 사용했을 경우와 본 논문에서 제안한 비주기 서버를 사용한 방식에 대해 스케쥴링 가능한 대역폭(bandwidth)을 비교한 결과이다. 그림 9의 결과에서 표준 Linux에서 RMS를 사용한 경우는 TCP/IP 소프트웨어 인터럽트의 우선순위가 가장 높기 때문에 1개/10ms 이상의 주기율로 메시지 수신이 이루어지는 경우 태스크의 수행이 불가능하므로 실시간 메시지 통신에는 부적합함을 알 수 있다. 아울러 TCP/IP 서비스를 제공받는 태스크의 우선순위가 낮은 경우(50ms 주기)가 우선순위가 높은 경우(30ms 주기)에 비해 TCP/IP 소프트웨어 인터럽트를 덜 발생시킴으로써 서비스를 제공하는 태스크의 중요도에 따라 TCP/IP 소프트웨어 인터럽트의 대역폭 이용율이 변경됨을 알 수 있다.

본 논문에서 제안된 방식을 이용할 경우 그림 9와 같이 똑 같은 패킷 전송률을 처리하면서도 태스크 작업으로 할당 가능한 대역폭은 향상됨을 입증하였다.

V. 결론

본 논문에서는 행동 기반의 자율형 이동로봇에 개방형 운영체제인 Linux와 상위 계층 네트워크 프로토콜의 사실상 표준인 TCP/IP를 적용하기 위해 필요한 구현 방안과 개선법을 제안하였다. 이를 위해 기존에 제안되었던 실시간 Linux에서 서버/클라이언트 통신을 사용했을 경우에 발생할 수 있는 문제점과 TCP/IP 통신이 Linux 시스템의 실시간성에 미치는 영향에 대한 분석을 수행하였다.

구현 및 개선책으로 먼저 행동 기반의 이동로봇 제어 프레임워크를 Linux 환경을 이용하여 구현하는 방안을 제안하였으며, 이를 기반으로 하여 높은 우선순위의 클라이언트가 낮은 우선순위의 서버에게 우선순위를 전달하여 클라이

언트의 응답 특성을 향상 시키는 기법을 제안하였다. 아울러 기존에 제안된 바 있는 비주기 서버를 응용하여 전체 시스템의 성능을 최대한 유지하면서 TCP/IP 패킷 손실을 방지하는 스케쥴링 기법을 제안하였다. 제안된 기법들은 모의실험을 통해 그 성능을 기존의 방식과 비교함으로써 유용성을 입증하였다.

향후 과제로는 태스크 간의 서버/클라이언트 통신뿐만 아니라 메시지 방식의 통신 방식에 대해 태스크의 우선순위를 반영할 수 있도록 하는 기법에 관한 연구를 수행할 예정이며, 아울러 서버/클라이언트 통신과 메시지 통신의 우선순위 반영 방식을 분산 환경의 태스크로 확장하는 것에 관한 연구를 수행할 예정이다.

참고문헌

[1] H. R. Kim, D. W. Kim, H. S. Kim, and H. I. Lee, "Toward the personal robot software frame work", *ICCAS Conf.,* pp. 2307-2312, 2002

[2] L. Abeni, A. Goel, C. Krasic, J. Snow, and J. Walpole "A measurement-based analysis of the real-time perfor mance of linux", *Proceedings of the IEEE Real-Time Embedded Technology and Appli cations Symposium (RTAS),* San Jose, California, Sep tember 2002

[3] M. Santori, and K. Zech, "Fieldbus brings protocol to process control", *IEEE Spectrum,* vol. 3, iss:3, pp. 60-64, March '1996

[4] CAN Specification Version 2, Robert Bosch GmbH, 1991

[5] IEEE/ANSI Std 1003.1 : Information Technology -(PO SIX)-Part1:System Application Program Interface include 1003.1a,1003.1b,1003.1c, 1996

[6] K. M. Obenland, "The use of POSIX in real-time systems, assessing its effectiveness and perofrmance", MITRE technical paeprs, www.mitre.org, 2000

[7] J. Mauro, "The solaris process model", www. sun world. com, 1998-99

[8] H. Tokuda and T. Nakajima and P. Rao, "Real-time mach: toward a predictable real-time system", *Procee dings of USENIX Mach Workshop,* pp. 73-82,1990,

[9] L. Sha, R. Rajkumar and J. Lehoczky, "Priority inheri tance protocol: an approach to real-time synchronization", *IEEE Transaction on Computers,* vol. 39, no.9, p p . 1175-1185, 1990

[10] N. Matthew, "Beginning linux programming 2nd ddition", Wrox Press Inc, 1999

[11] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and ave rage case behavior". *IEEE Proc. on Real Time Sympo sium,* pp. 166-171, 1989

[12] D. C. Schmidt and F. Kuhns, "An overview of the real-time CORBA specification," *IEEE Computer Maga z ine, Special Issue on Object-oriented Real-time Com puting,* June 2000.

[13] A.V.T.W Group, "RTP:A transport protocol for real time applicaions", RFC 1889, 1996

[14] 배덕진, 김대원, "Ethernet기반의 인트라넷에서 네트워크의 실시간 응답 성능에 관한 연구", *Proc. of Interna tional Conf. on Control, Automation and Systems,* pp. 2907-2910, October 2001.

[15] D. P. Bovet and M Ceasti, "Linux kernel", O'Reilly, 2001

[16] Rodney A. Brooks, "A robust layered control system for a mobile robot", *IEEE Journal of Robotics and Automation,* vol. RA2, no. 1, pp. 14-23, 1986

[17] R. C. Arkin, "Motor schema based mobile robot navi gation", *Int. Journal of Robotics Research,* vol 8, pp. 92-112, 1989

[18] LynuxWorks, "LynXOSUser's Guide", www.Lynux Works. com, 2002

[19] M. Spuri and G. Buttazzo, "Scheduling aperiodic tasks in dynamic priority systems," Real-Time Systems, vol. 10, pp. 179-210, 1996

김 주 민

1973년 2월 10일생. 2001년 명지대학교 전기전자공학부 졸업. 2003년 동대학원 졸업. 2003년~현재 동대학원 박사과정 재학중. 관심분야는 실시간 분산 시스템, 퍼스널로봇 등.

김 홍 렬

1973년 11월 25일생. 1996년 명지대학교 전기공학과 졸업. 1998년 동대학원 졸업. 2001년~현재 동대학원 박사과정 재학중. 1997년~현재 (주)캐리어 기술연구소 주임연구원. 관심분야는 실시간 분산 시스템, 행위기반 로봇 프레임워크 등.

양 광 웅

1972년 6월 25일생. 1996년 인하대학교 자동화공학과 졸업. 1998년 동대학원 졸업. 2002년~현재 한국생산기술연구원 연구원. 관심분야는 실시간 컴퓨팅 시스템, 퍼스널로봇, 컴파일러, 가상머신 등.

김 대 원

1960년 2월 15일생. 1984년 서울대학교 제어계측공학과 졸업. 1986년 동대학원 졸업. 1987년~1992년 (주)대우중공업 중앙연구소 선임연구원. 1992년~현재 명지대학교 정보공학과 교수. 관심분야는 실시간 분산 시스템, 퍼스널로봇, 웹기반 응용 등.