소 특 집

상용 RTOS 적응이 가능한 장치제어계 프레임워크 개발 사례

강 경 순, 문 성, 김 형 환, 임 동 선, 최 완

한국전자통신연구원

I. 서 론

일반적으로 임베디드 시스템(Embedded System)은 컴퓨터의 하드웨어와 소프트웨어, 그리고 추가적인 주변장치 등으로 구성되어 특정 기능을 수행하도록 설계된 시스템이다. 임베디드 시스템은 범용 시스템과 비교하여 특화된 기능만을 수행하므로 단순한 하드웨어 회로 혹은 하나의 순차적 프로그램만으로 구성할 수도 있다 그러나 최근에는 디지털 TV, 비디오 폰, PDA 등과 같은 차세대 정보기기와 같이 임베디드 시스템의 적용 분야가 확대되고 있으며 실시간 처리 및 멀티태스킹과 같은 복잡한 기능이 임베디드 시스템에 요구되고 있다. 더욱이 짧아진 제품 개발주기와 다양한 응용 프로그램의 안정적인 개발이 필요하게 됨으로써 임베디드 시스템에 실시간 운영체제(Real-Time Operating System ; RTOS)를 사용하는 것은 일반적인 현상이 되었다.

실시간 운영체제는 그 개발의 주체에 따라서 크게 두 가지로 나뉠 수 있다. 대체로 상용 제품의 로열티를 회피하기 위하여 혹은 상용 제품이 지원하지 않는 하드웨어나 특수한 기능이 필요한 경우에 자체 개발되는 실시간 운영체제(In-House RTOS)와 특정 업체 혹은 그룹에서 임베디드 시스템에 어느 정도의 범용성을 갖추어 타 업체에 제공하기 위하여 개발되는 상용 실시간 운영체제(Commercial RTOS)로 구분될 수 있다. 상용 실시간 운영체제는 비용 측면에서 운용 로열티를 지급하는 것과 최초 구입 시 사용료를 일괄 지급하는 것들이 있다.

실시간 운영체제는 우선 순위에 기반한 선점형(Preemption) 스케줄링, 멀티 쓰레드(Multi-Thread) 지원 및 쓰레드간의 예측 가능한 동기화 방법을 제공해준다. 또한 우선 순위(Priority) 상속 기능, 짧고 제한된 시간의 인터럽트 전달 지연과 처리, 제한된 시간 내에서의 시스템 호출 처리 및 인터럽트 마스킹 등을 지원해 준다. 이와 같이 시스템 개발자가 부담하여야 할 작업을 실시간 운영체제가 대신해 줌으로써 생산성과 효율성, 안정성을 높이게 된다.

모든 실시간 운영체제가 동일한 기능과 성능을 제공해 주는 것은 아니므로 응용이 복잡해지고 제품을 시장에 내놓아야 할 시기 및 향후 제품 단가에 대한 고려를 할 때 어떤 실시간 운영체제를 선택할 것인가는 매우 중요한 문제가 된다. 예전에 개발된 롬 메모리에 저장되어서 순차적인 프로그래밍 방식으로 수행되는 임베디드 시스템의 기능을 확장하고 향후 유지보수 비용을 줄이기 위해 실시간 운영체제의 도입을 고려하고 있다고 가정하자. 유지보수를 편리하게 하기 위해서는 교차 개발 환경이 필요하고 성능 향상을 위해서는 멀티태스킹이 지원되는 운영체제가 필요하다. 풍부하고 다양한 개발환경을 제공해주는 상용 실시간 운영체제를 도입할 경우에는 별도의 운영체제 개발 기간 없이 이러한 문제점들을 해결할 수 있다.

상용 실시간 운영체제를 도입할 경우에는 이미 개발된 응용 프로그램들을 최소한의 수정으로 새로운 환경 하에서 운영이 가능한지를 점검하여야 한다. 또한 커널 소스가 함께 제공되는지, 만약 제공되지 않는다면 사용자의 요구사항이 도입될

상용 실시간 운영체제의 소스를 수정하지 않고도 수용될 수 있는지를 고려하여야 한다. 만약 운영체제 소스를 수정해야 한다면 수정에 필요한 별도의 비용도 고려해야 한다. 그리고 POSIX와 같은 표준을 준수하고 있는지를 검사하여야 한다. 운용 로열티 지불이나 새롭게 추가되는 특수한 장치를 해당 실시간 운영체제가 지원하지 않는 경우에는 향후 다른 운영체제 혹은 자체 개발한 운영체제로 쉽게 천이가 가능한 구조인지를 확인하여야 한다. 이와 같은 고려사항을 충족시키며 기존의 자체 개발 시스템에 새로운 상용 실시간 운영체제를 적응(Adaptation)하기 위한 구조가 다음에 제시하는 ALFA(Abstracted Layer for Fast Adaptation) 프레임워크(Framework)이다.

II. ALFA 프레임워크

새롭게 도입될 실시간 운영체제와 응용 프로그램 사이에 별도의 계층을 두어 응용 프로그램이 상용 실시간 운영체제 제품들의 고유한 API (Application Programming Interface)를 사용하지 않고 하부 실시간 운영체제가 제공하는 API에 대한 종속성을 없애주는 API인 유니버설 (Universal) API만을 사용하도록 하면 기 개발된 응용 프로그램들에게 운영체제로부터의 독립성을 확보할 수 있게 해준다. 즉, 하부 운영체제가 바뀌어도 응용 프로그램은 변경하지 않아도 된다.

운영체제와 응용 프로그램 사이의 별도의 계층은 하부 커널의 API를 추상화한 유니버설 API들을 제공하는 서버들로 구성되는 하나의 프레임워크가 되며 이를 이용해서 기존 응용 프로그램들을 수정하지 않거나 최소한의 수정만으로 쉽고 빠르게 새로운 운영체제를 도입할 수 있다. 이러한 의미에서 상기한 별도의 계층을 ALFA (Abstracted Layer for Fast Adaptation)라고 부른다.

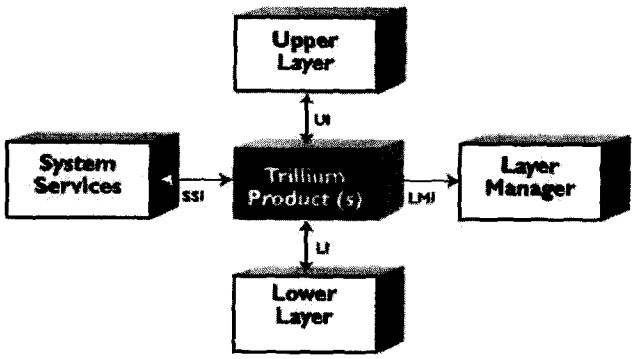
ALFA 구조를 갖는 장치 제어계 프레임워크를 개발하기 위하여 교차 개발 환경을 제공하는 상용 실시간 커널이 ALFA의 하부에 위치하도록 구성된 장치 제어계 공통 개발 플랫폼을 개발하였다. 상용 실시간 커널의 API를 ALFA의 API로 추상화함으로써 HANbit ACE ATM 교환기 장치 제어계에 적용되는 실시간 운영체제인 mSROS(micro Scalable Real-Time Operating System)가 생성된다.

하부 커널의 종류에 관계없이 상위의 응용 프로그램들을 손쉽게 이식할 수 있도록 하기 위한 목적으로 만들어진 프레임워크는 다른 시스템에서도 그 예를 찾을 수 있다. 각종 프로토콜 스택을 판매하는 Trillium사와 Netplane사는 다양한 수행 환경에 자사의 소프트웨어 제품들을 최소한의 변경 혹은 변경 없이 이식할 수 있도록 하기 위하여 각각 TAPA(Trillium Advanced Portability Architecture)[3]와 LEAP(Layered Environment for Accelerated Portability)[4]라는 프레임워크를 개발하였다. 또한 IpInfusion사의 ZebOS Advanced Routing Suite[5]는 Network Service Module(NSM)을 통해 라우팅 스위칭 소프트웨어에서 확장이 가능한 모듈화 되고 플랫폼 독립적인 ZebOS 구조를 채택함으로써 Linux, VxWorks, OSE, Solaris, BSD 등 여러 운영체제를 지원하도록 하고 있다. 아래에서는 TAPA와 LEAP에 대해 알아본다.

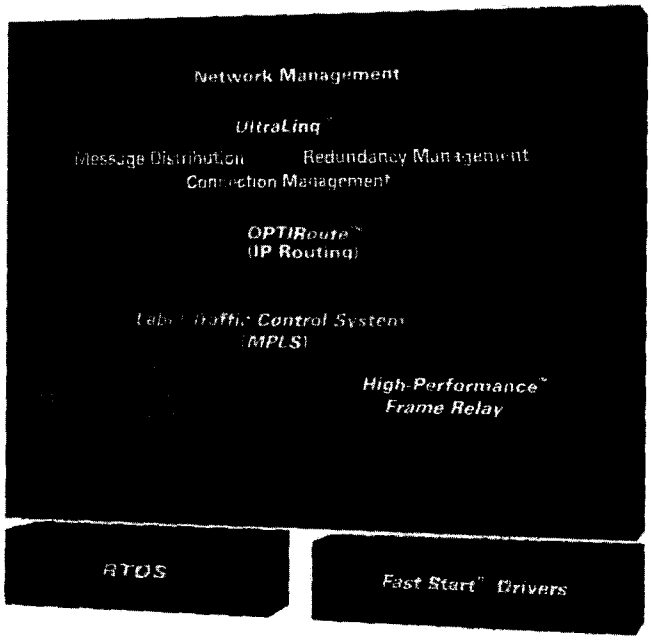
1. TAPA

Trillium사의 제품들은 공통적으로 구조 및 프로그래밍 표준을 공유하고 있다. 이러한 표준을 실현한 것이 TAPA이며 이것을 통해 새로 개발되는 제품들은 설계 및 구현 그리고 시험의 과정에서 최소한의 변경만으로 순차적으로 추가될 수 있다. TAPA는 Trillium 소프트웨어와 타겟(Target) 시스템간의 모든 상호작용을 제어하는 인터페이스 집합을 정의한다. <그림 1>은 TAPA의 구조를 나타낸다.

각 Trillium 소프트웨어들은 단일 태스크처럼 직접 함수 호출을 통해서 또는 연동하는 태스크



〈그림 1〉 TAPA 구조도



〈그림 2〉 LEAP 구조도

들 사이의 메시지에 의해서 하나 또는 그 이상의 서비스 사용자나 공급자들에게 연결될 수 있다. 모든 Trillium 소프트웨어 제품들은 동일한 시스템 서비스 인터페이스를 사용한다.

2. LEAP

Netplane사의 LEAP 구조는 타겟의 동작 환경에 공통 인터페이스를 제공하고 운영체제와의 통합(Integration), 플랫폼 서비스 그리고 하드웨어 드라이버들과의 통합을 제공함으로써 다른 Netplane 서브 시스템들을 사용하는 사용자에게 요구되는 통합 노력의 최소화를 목적으로 한다. 따라서 다양한 Netplane 서브 시스템들의 재사용성을 증진시키기 위해 개체들에 대한 인터페이스들을 표준화하였다. 〈그림 2〉는 LEAP의 구조를 나타낸다. LEAP은 제품별로 특화된 기본 코드가 공통 기본 코드(Common Base Code)로부터 분리되고 모든 기본 코드가 하부의 타겟 환경으로부터 분리되도록 서비스들을 분할(Segregates and Partitions)한다.

LEAP의 운영체제 프리미티브 계층은 LEAP 구조를 특정 운영체제에 매핑 시켜줌으로써 LEAP의 운영체제 서비스 계층을 위한 API로서 동작하게 된다. 운영체제 프리미티브들은 이식 파일들(Porting Files)를 통해 특정한 운영체제에 매핑(Mapping) 된다. 이것은 타겟 서비스들과 드라이버 인터페이스 콤포넌트들이 타겟 환경에서 서브 시스템들을 분리시키는 것과 같은 개념이다.

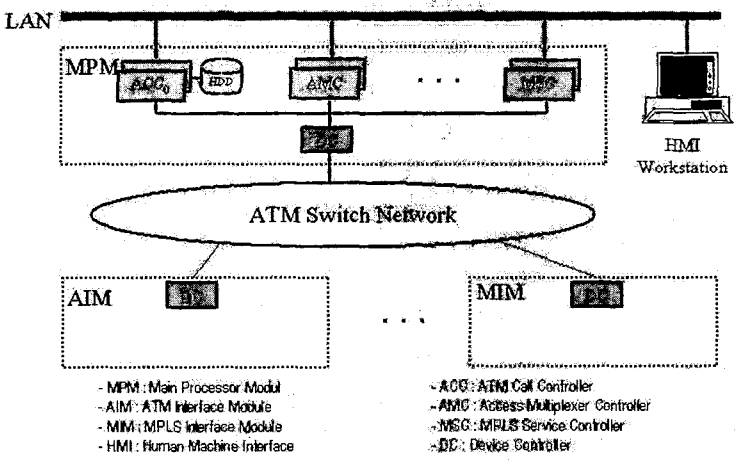
Ⅲ. mSROS 적용 사례

1. mSROS 구조

HANbit ACE ATM 교환기 시스템은 〈그림 3〉과 같이 주 제어계(Main Control System)와 장치 제어계(Device Control System)로 구성되는 2계층 제어계 구조를 갖는다. 주 제어계는 교환 시스템 전체의 운용 및 유지보수와 가입자들의 각종 서비스들에 대한 처리와 트래픽 제어(Traffic Control) 등을 담당하며, 장치 제어계에서는 ATM 단말기 등과 같은 서비스 가입자들을 직접 수용하고 ATM 셀(Cell)들의 분해 조립 기능과 ATM 시그널 처리 기능들과 같은 연결 제어를 담당하게 된다. 분산된 시스템들 사이의 동기 및 통신을 위하여 IPC[11]를 사용하며 이 기능은 시스템의 처리 용량 및 성능과 밀접한 관계를 갖게 된다.

ATM 교환기의 최초 모델에서 장치 제어계는 Motorola사의 MPC68X로 같은 계열이기는 하지만 동일한 CPU를 적용하지 않았고 서브 시스템들을 위한 하드웨어 보드들도 서로 다르게 구성되었다. 이로 인하여 서브 시스템별로 응용 프로그램을 위해 제공되는 모니터 프로그램도 각 하드웨어 보드별로 개발되었다. 이러한 장치 제어계의 구성은 아래와 같은 문제점들을 내포하고 있다. 모니터 기반이기 때문에 서비스의 추가나 오류 수정이 어렵고 ROM(Real Only Memory)으로 제작되어 응용 프로그램을 교체하기 위해서는 교환 시스템의 운용을 장시간 중단하여야 하는 문제점이 있다. 이는 연속 서비스를 제공해야 하는 교환 시스템의 신뢰성에 중요한 결점으로 작용한다. 마지막으로 멀티태스킹이 지원되지 않고 프로시져(Procedure) 방식의 순차적인 수행만 지원하므로 성능 향상에 한계가 있다.

ATM 교환 시스템의 장치 제어계 개발을 위하여 교차 개발 환경을 지원하는 상용 실시간 운영체제의 도입을 통하여 순차적인 수행을 제공하는 모니터를 기반으로 했던 응용 프로그램들은 멀티태스킹이 지원되는 운영체제를 통해서 성능을 향상시키고 유지 보수에 드는 비용도 절감할



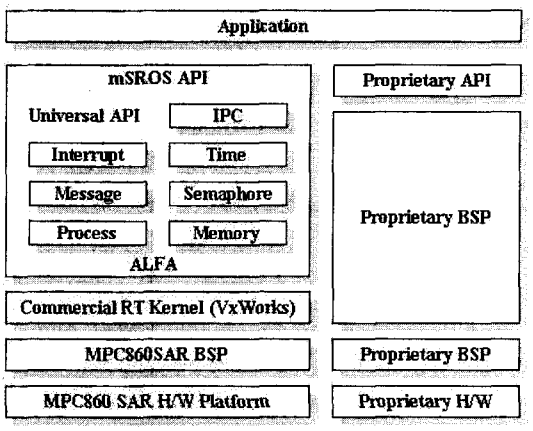
〈그림 3〉 HANbit ACE2000 ATM 교환시스템

수 있었다. 교환 시스템에서 요구하는 특수한 기능들과 제약 사항을 수용하기 위해서는 도입될 운영체제에 대해 수정이 발생할 수 있으며 이로 인하여 개발 기간이 지연된다. 그러므로 도입될 운영체제는 운영체제에 대한 수정 없이 필요한 기능을 추가하기에 적합한 마이크로 커널(Micro Kernel) 구조이어야 한다[2]. 이러한 고려 사항에 적합한 상용 실시간 운영체제로 WindRiver사[6]의 VxWorks 커널과 Tornado 교차 개발 환경을 선택하였다.

ALFA 구조를 갖는 프레임워크와 상용 실시간 운영체제인 VxWorks 커널로 구성된 mSROS를 개발하는 것은 단일화된 하드웨어 및 운영체제라는 개념을 기본으로 하므로 장치 제어계를 구성하는 모든 하드웨어 보드들의 공통 부분을 추출하는 작업이 선행되었다. 추출된 하드웨어의 공통 부분에는 Motorola PowerPC 계열의 MPC860SAR 칩을 적용하였다. 이 하드웨어 공통 부분을 코어 보드로 하여 각 응용별로 필요한 부분을 추가하는 방법으로 여러 종류의 장치 제어계용 하드웨어 보드를 제작하였다. mSROS는 하드웨어의 공통 부부인 코어 보드에 적용되며 응용에 맞게 별도의 하드웨어 모듈들을 코어 보드에 추가하여 최종적인 하드웨어 보드가 만들어지고 추가된 하드웨어 모듈들의 제어에 필요한 소프트웨어들은 mSROS의 한 구성으로 수용되는데 〈그림 4〉의 Proprietary 부분이 이것에 해당된다.

〈그림 4〉는 mSROS의 구조를 나타낸다. mSROS는 다양한 개발도구를 포함하는 상용 개발 환경과 연동이 가능한 상용 실시간 커널을 기본으로 하고 교환기 응용을 위해 요구되는 고성능 고신뢰 IPC 통신을 위한 전용 IPC 프로토콜 처리 기능, 다양한 실시간 관련 요구사항을 처리하기 위한 시간관리 기능, 프로세스 제어 기능, 동기 및 통신 기능, 메모리 기능, 인터럽트 처리기의 등록 및 취소 기능들로 구성된 가상 기계를 갖고 응용 프로그래머들에게 가상 기계에 기반한 인터페이스 API를 제공한다.

장치 제어계는 공통 하드웨어와는 별도로 가입



〈그림 4〉 mSROS 구조

자의 특성 및 각 응용에 따라 각기 자신들만의 하드웨어(Proprietary H/W)를 구성할 수 있는데 mSROS는 이 부분을 수용하기 위한 인터페이스를 제공하고 있어서 각 장치 제어기별 응용 프로그램 개발자들은 자신들만의 요구사항에 따른 부분들(Proprietary BSP, Proprietary Device Driver, Proprietary API)을 추가함으로써 하부 커널을 추상화한 가상기계가 제공하는 인터페이스를 손쉽게 확장할 수 있도록 하였다.

시스템 이미지를 최초 플래쉬(Flash) 메모리에 저장하도록 하여 부팅 시에는 자신의 이미지로 부팅하도록 하였다. 그리고 시스템 이미지가 변경된 경우에는 새로운 이미지를 플래쉬 메모리에 저장하는 방법을 통해 운용 중에 시스템 이미지를 변경할 수 있도록 하였다. 이러한 DC 로딩(Device Controller Loading) 기능으로 인하여 이전의 ROM 교체 방식에 비해 보다 나은 시스템 서비스의 연속성을 보장해 주도록 하였다.

2. mSROS의 ALFA 구조

mSROS ALFA는 장치 제어계 응용 프로그램들의 필요에 따라 다양한 교환기 서비스 기능들을 구현할 수 있는 API를 제공하는 IPC 서버, Time 서버, Message 서버, Semaphore 서버, Process 서버, Memory Server, Interrupt 서버로 구성된다. 이중에서 교환기 응용에

특화된 IPC 서버와 Time 서버를 개발하였으며 나머지 서버들은 호환성 제공을 위해 하부에 도입될 상용 실시간 커널에서 제공되는 API를 추상화하였다. 이러한 작업을 통해 동작 환경에 영향 없이 하부 커널을 변경하거나 다른 상용 실시간 운영체제 커널로의 교체 혹은 자체 개발한 커널로의 교체가 손쉽게 이루어질 수 있는 구조를 갖도록 하였다.

프레임워크의 하부에는 다양한 커널이 사용될 수 있다. 임베디드 운영체제의 표준화가 진행되고는 있으나 일반적으로 커널들은 서로 다른 API를 제공한다. 응용 프로그램들은 하부 커널별로 제공하는 API에 독립적이어야 하므로 유니버설 API를 통해 응용 프로그램이 하부에 위치하는 운영체제의 API 변화에 독립적일 수 있도록 한다.

3. 서버와 상용 실시간 운영체제 커널과의 인터페이스

VxWorks는 커널 소스는 제공하지 않지만 응용에 맞는 확장을 위해 필요한 부분을 제공해 준다. mSROS에서의 프로세스 제어 정보(Process Control Block ; PCB)는 기존 VxWorks의 태스크 제어 정보(Tack Control Block ; TCB)를 기본으로 IPC 기능을 위한 부분이 확장되었다. VxWorks의 TCB는 이러한 확장이 가능하도록 24바이트의 영역을 TCB에 예약해 두고 있다.

IPC 서버의 구현을 위하여 8바이트의 영역을 사용하고 있다. 4바이트는 IPC의 시그널 송수신 및 프로토콜 처리 기능을 위한 IPC PCB를 위해 할당하고 나머지는 mSROS에서의 프로세스가 갖는 인스턴스 값(Process ID)을 저장하기 위해 할당된다.

Ⅳ. 결 론

HANbit ACE 교환기 시스템은 주 제어계와 장치 제어계로 구성되는 제어계 구조를 갖는다. 장치 제어계 초기 개발 모델에서 나타난 각각 상이한 하드웨어 구조에서 서브 시스템별로 모니터 프로그램을 개발함으로써 나타나는 비효율성 및 성능 저하를 줄이고자 교차 개발 환경을 지원하는 상용 실시간 운영체제의 도입을 고려하게 되었다. 상기한 문제점들을 해결하기 위해 재구성이 손쉬운 마이크로 커널 기반의 클라이언트-서버 구조를 갖는 VxWorks 커널과 교차 개발 환경인 Tornado를 장치 제어계용 운영체제로 도입하였다. ALFA 구조를 갖는 프레임워크를 개발함으로써 커널에 대한 수정 없이 개발 기간을 단축시킬 수 있었으며 기존 응용 프로그램들은 ALFA 구조를 갖는 프레임워크에서 제공되는 API 만을 사용함으로써 하부 커널이 대체되더라도 API의 일관성을 유지할 수 있다.

ATM 장치 제어계용 운영체제를 위하여 ALFA 구조를 갖는 프레임워크의 API로 교차 개발 환경을 제공하는 VxWorks 커널을 새롭게 추상화함으로써 mSROS를 개발하였다. 이러한 mSROS를 ATM 교환기의 장치 제어계에 적용함으로써 기존 2ms 간격으로 폴링 방식을 통해 동작하는 IPC에 비하여 월등히 개선된 IPC 성능을 얻을 수 있었으며 안정적이면서도 손쉬운 어플리케이션의 개발 및 디버깅의 편리성을 얻을 수 있었다. 향후 자체 커널로의 대체도 최소한의 노력과 비용으로 이루어질 것으로 예상된다.

참 고 문 헌

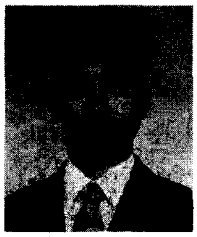
[1] Hyung Hwan Kim, Sung Ik Jun, Young Jun Cha, Ju Hyun Jo, "HLRP : A High-Performance Light-weight Real-Time Protocol", IEICE'95, pp.467-470, Aug. 1995.

[2] Randy Chow, Theodore Johnson, "Distributed Operating Systems & Algorithms", Addision Wesley, pp.4-25, 1997.

[3] http://www.trillium.coms

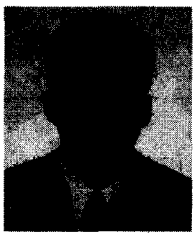
〔4〕 http://www.netplane.com
〔5〕 http://www.ipinfusion.com
〔6〕 http://www.windriver.com

저 자 소 개



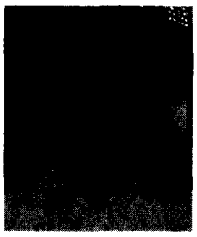
姜 景 淳

1998년 2월 원광대학교 컴퓨터공학과(학사), 2000년 2월 동국대학교 컴퓨터공학과(석사), 1999년 12월~2000년 10월 : 서울증권, 2000년 10월~현재 : 한국전자통신연구원 연구원, 〈주관심 분야 : 실시간 운영체제, 임베디드 시스템〉



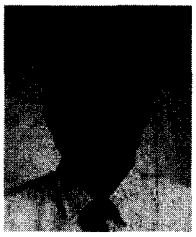
文 星

1996년 2월 한남대학교 전산학과(공학사), 1999년 2월 충남대학교 컴퓨터공학과(공학석사), 1995년 12월~1996년 10월 : 외무부 외신과, 1999년 1월~2000년 4월 : (주) 인컴아이엔씨, 2000년 5월~현재 : 한국전자통신연구원 선임연구원, 〈주관심 분야 : RTOS, 운영체제, 라우팅〉



金 亨 煥

1991년 2월 한양대학교 전자공학과(공학사), 2000년 2월 충남대학교 컴퓨터과학과(이학석사), 2001년 3월~충남대학교 컴퓨터과학과, 박사과정, 1991년 1월~현재 : 한국전자통신연구원 선임연구원, 〈주관심 분야 : 인터넷 트래픽 관리, 분산실시간처리, 내장형 실시간 시스템〉



林 東 先

1986년 2월 숭실대학교 전자계산학과 (공학사), 1996년 2월 한국과학기술원 정보및통신공학과(공학석사), 1986년 1월~현재 : 한국전자통신연구원 네트워크연구소 책임연구원, 〈주관심 분야 : 실시간시스템, 통신망(분배망), 멀티미디어서비스〉

崔 完

1981년 2월 경북대학교 전자공학과 학사, 1983년 2월 KAIST 전산학과 석사, 1983년 3월~1985년 2월 : KAIST 전산학과 조교, 1985년 3월~현재 : ETRI 네트워크 연구소 팀장, 1988년 8월 : 정보통신 기술사(전자계산기조직응용), 2000년 8월 : SPICE(S/W Process Improvement Capability dEtermination) 심사원, 〈주관심 분야 : Embedded System Software(RTOS, RTDBMS, 고성능 미들웨어), Network Architecture, 인터넷 프로토콜, S/W 공학〉