J · BOSN 실시간 운영체제 ①

J · BOSN 실시간 운영체제란 무엇인가

J · BOSN 실시간 운영체제는 고성능의 멀티-태스킹(Multi-tasking)을 지원하고 매우 강력한 실시간 응용프로그램을 개발할 수 있는 바탕을 제공하여 준다. 1회에서는 J · BOSN 실시간 운영체제의 개념을 알아보고 나아가 실시간 시스템, 설계원리 등과 나노커널, 트랩, 쓰레드관리, 메모리 관리, 메시지 통신 관리 기능 등에 알아본다.

자료제공: 아이보슨시스템즈
www.jbosn.com

J · BOSN 실시간 운영체제는 고성능의 멀티-태스킹을 지원하고 강력한 실시간 응용프로그램을 개발할 수 있는 바탕을 제공하여 준다. 특히 운영체제의 기능이 모듈화 되어 있어 사용자가 개발하는 실시간 시스템에 맞게 운영체제를 재구성할 수 가 있다. J · BOSN 운영체제는 다음과 같은 기능에 최적화 되어 설계가 되었다.

- 성능
- 안정성
- 확장성
- 자율성
- 사용상 편의성

결과적으로 J · BOSN 운영체제를 기반으로 개발한 실시간 시스템에 매우 강력한 신뢰성을 제공하며

사용자의 요구에 맞는 자유로운 재구성을 보장하면서 시스템을 설계하는데 특별한 제한조건을 부과하지 않는다. 사용자는 J · BOSN 운영체제를 실시간 시스템에 이식하거나 디바이스 드라이버를 개발할 때, 최대한 자율성을 보장받는다.

구조

J · BOSN 실시간 운영체제는 모듈화를 기본 설계개념으로 하였다. 고성능의 실시간 커널(나노커널)과 다른 서비스모듈(마이크로커널, 매크로커널)이 결합되어 전체시스템이 이루어진다. 커널을 구성하는 모듈도 사용자의 요구조건에 따라 선택적으로 재조립이 될 수도 있다. 모든 모듈은 각자의 독립성을 최대한 보장하고 있으며, 각 모듈은 실시간 시스템의 서비스를 확장시키는 것이고, 모듈 중 필요한 서비스기능들만 결합하여 사용자의 요구조건에 만족하는 실시간 개발시스템을 제작할 수 있다. J · BOSN 운영체제에서 기본적으로 제공하는 모듈 등은 다음과 같다.

고성능의 실시간 나노커널

사용자의 성공적인 리얼타임 시스템의 구현을 가능하게 해주는 강력한 멀티태스킹 멀티쓰레딩 기능 지원과 서버간의 통신채널의 관리 및 운영체제에 필요한 각종 기능을 최소한으로 구현한 매우 작고 효율적인커널이다. 멀티타스킹기능은 실시간 응용프로그램들이 여러 개의 독립적인 프로세스로 실행이 되고 각 프로세스는 시스템 리소스들과 여러 개의 쓰레드들을 가지고 독립적인 일을 수행할 수 있다.

시스템 서버

시스템의 모든 리소스관리를 관리하는 서비스 모듈이다. 리소스는 하드웨어적인 리소스인 메모리를 포함해, 소프트웨어적인 리소스인 각종 커널 구조체 및 데이터를 관리한다. 특히, 시스템의 전원관리를 총책임지고 있는 서비스 모듈이다.

타임 서버

시스템의 운영에 관련된 각종 타이머와 시간에 관련된 기능을 제공하여 준다. 시간에 관련된 정보는 리얼타임 시스템에서 매우 중요한 기능을 담당하고 있으며, 시스템 전체의 시간관리 및 드라이버와 애플리케이션에서 필요한 시간관련 기능을 제공한다.

동기화 서버

쓰레드/태스크 간의 동기화 및 통신을 관리하는 부분이다. 여러 프로세스나 쓰레드들 간의 통신기능은 시스템을 구성하는데 있어서 일을 분할하여 처리하도록 여러 쓰레드로 나누어서 프로그램을 작성할 수 있게 하여준다. 이 나누어진 쓰레드들 간의 상호 협조기능은 이곳에서 제공하는 기능을 사용하여 달성할 수 있고, 시스템 전체의 통합기능을 제공할 수 있다.

I/O 시스템 서버

J · BOSN 운영체제는 여러 특성의 디바이스를 접근하는데 동일한 소프트웨어 인터페이스를 제공하고 있다. 사용자는 제공된 인터페이스만을 사용하여 모든 디바이스 드라이버에 접근하여 원하는 서비스를 제공받을 수 있다. 또한 리부팅 없이 드라이버의 탈착이 가능하고, 다른 모듈과의 의존성을 최소화하였다.

파일 시스템 서버

실시간 기능과 MS 윈도우와의 호환성을 유지한 매우 빠른 파일시스템이다. 현재는 FAT파일시스템, ROM파일시스템, RAM 파일시스템 등을 지원하고 있다. 그러나 실제 파일 시스템은 사용자의 시스템을 재구성할 때, 시스템의 플랫폼 영역에 포함이 되어 운영이 될 것이다. 따라서, 파일 서버는 단지 전체적인 파일 관리의 도구를 제공한다. FAT, ROM, RAM 파일 시스템은 라이브러리 형태로 제공이 되고 있다.

윈도우 시스템 서버

다양하고 효율적인 사용자 인터페이스와 그래픽을 원하는 시스템에서는 이 서버를 운영하여 기본적이고 강력한 윈도우 개념을 구현할 수 있으며, 사용자가 요구하는 강력한 GUI를 구현할 수 있다.

네트워크 서버

네트워크를 원하는 시스템에서는 이 서버를 운영하여 네트워크에서 필요한 기본적인 프로토콜 및 소켓인터페이스 개념을 구현할 수 있다.

I/O 드라이버

드라이버의 디자인 모델은 특별한 제한조건을 부과하지 않고 사용자와 디바이스의 특성을 고려하여 자유롭게 작성을 할 수 있도록 되어 있다. 스트림 디바이스 드라이버, 웨이브 디바이스 드라이버, 블록 디바이스 드라이버, 터치 디바이스 드라이버 등 다양한 드라이버 모델은 표준적으로 제공된다.

보드-서프트 패키지

J · BOSN 운영체제를 개발시스템에 이식을 시키는 부분으로서, 모든 시스템에 대한 동일한 소프트웨어 인터페이스를 제공하여 이식성을 극대화한다. 하드웨어의 초기화와 인터럽트의 관리와 발생, 하드웨어 클락 및 시간 관리, 메모리의 크기와 위치 관리 및 다른 하드웨어 관리를 포함하고 있다.

부트 코드

부팅코드는 최초로 타깃 시스템에 들어가서 크로스 개발툴(Cross Development Tools)에 의해서 개발된 시스템을 타깃 시스템에 장착하여 부팅을 할 수 있는 기능을 제공한다.

실시간 운영체제인 J · BOSN은 모든 기능이 모듈화 되어 있고, 각 커널모듈은 독립적인 쓰레드로서 움직이고 있다. 최소한의 기능을 나노커널(NanoKernel)에 구현을 하고 운영체제가 갖추어야 할 필수적인 기능들은 서비스요청을 받아 서비스를 해주는 서버의 개념으로 작성된 모듈로 이루어져 있다. 따라서 운영체제의 기능은 완벽하게 서로 분리되어 있고, 또한 서로 의존적이지 않다. 이와 같은 구조에서는 필요한 기능만을 사용자가 골라서 자신에게 맞는 최소, 최적의 실시간 시스템을 구현할 수 있도록 하여 주는 장점이 있다.

J · BOSN 운영체제

소개

실시간 운영체제는 멀티태스킹과 통신기능의 조화를 바탕으로 제작되도록 구성된다. 멀티태스킹을 지원하면 시스템 자원과 여러 개의 쓰레드를 가진 태스크로 실시간 시스템이 구성되는 것이 가능하다. 통신기능은 여러 개의 태스크나 쓰레드 간의 동기화나 통신기능을 사용하는데 사용하며, 주어진 작업을 하는데 상호협력관계를 구축할 수 있는 기능을 제공한다. J · BOSN 운영체제는 멀티태스킹과 매우 다양한 통신기능을 제공하고 있다.

실시간 운영체제에서 또 다른 기능중의 하나는 하드웨어 인터럽트의 관리이다. 하드웨어 인터럽트는 외부의 이벤트(서비스 요구)를 시스템에게 알려주는 중요한 수단이기 때문이다. 가능한 매우 빠른 인터럽트 응답을 얻는 리얼타임성을 높이기 위해, J · BOSN 운영체제는 인터럽트 차단(Disable) 구간을 최대한 줄여 언제나 인터럽틀 받을 수 있는 대기상태를 유지하고 있다.

실시간 시스템

실시간이란 무슨 뜻인가? 실시간 시스템이란 용어는 정확한 시간 내에 내부나 외부의 이벤트(서비스 요청)에 대하여 요구되는 정확한 응답을 하여 주어야 하는 시스템을 일컷는 말이다. 실시간성이란 두 가지로 나눌 수 있다. 첫째 소프트실시간(Soft Real-Time)성은 가끔씩 정확한 시간에 정확한 대응

을 하지 못하여도 시스템의 동작에 큰 손실을 가져오지 않는 것을 말한다. 둘째 하드 실시간(Hard Real-Time)은 정확한 대응이 없으며 시스템에 치명적인 오류가 발생하는 것을 말한다.

오늘날의 실시간 시스템은 다양한 요구사항을 만족시켜주어야 한다. 하나의 일을 하는 시스템은 보통 하나의 목적만을 위하여 동작을 하여 독립적인 시스템 형태로 구현이 가능하다. 하지만 요즘의 실시간 시스템은 하드 실시간과 소프트 실시간이 공존하는 경향이 뚜렷하고 두기능이 동시에 설계되어도 원하는 실시간 시스템의 목적에 충분히 기능을 발휘하기를 원한다.

문제점

실시간 시스템에서는 실행되는 프로세스들에게 각자가 가지는 중요도에 따라서 프로세서를 분배하는 방법이 매우 중요하다. 즉, 매우 중요한 이벤트가 발생하였을 때, 그 이벤트에 대응하는 프로세스를 최대한 빨리 실행할 수 있게 해야 한다. 그러나 실시간 운영체제가 없이 구현한 실시간성 요구 시스템은 이런 문제에 정확하고 신속하게 대응할 방법에 상당한 제약을 받는다.

예를 들면, 소프트실시간성을 요구하는 단순한 실시간시스템은 어플리케이션이 직접 프로세서를 제어하는 작은 로직을 가지고 있는 것이 대부분이다. 이런 종류의 구현은 태스크가 실행되고 있는지를 지속적으로 검사하는 제어 루프를 기반으로 구성이 된다. 이런 구현 방법은 실시간성을 떨어뜨리는 다양한 문제점들을 안고 있다.

첫째로 긴급한 이벤트를 발생 즉시 감지하지 못하고 또한 이를 수행하여주기 위하여 대응하는 루틴을 수행할 수 있게 만드는데도 시간이 많이 걸려서 느린 응답속도의 문제를 야기한다.

둘째로 CPU를 할당하는 부분이 응용프로그램에 들어 있기 때문에 CPU를 할당받아서 실행을 할 수 있는 기회가 다른 태스크에 매우 의존적이게 된다. 이것은 하나의 응용프로그램에서 오류가 발생하면 그 오류가 전 시스템에 전파되는 결과를 초래하게 된다. 결론적으로 응용프로그램을 수정하는 것이 굉장히 어렵게 된다.

셋째로 시스템의 자원을 관리하는 부분이 애플리케이션에 직접 영향을 받아서 통합적인 관리가 이루어지지 않아 시스템 자원의 낭비를 야기한다. 이것은 실시간 시스템의 성능에 치명적인 약점이다.

넷째로 실행할 태스크의 숫자가 많아지면 태스크가 실제로 실행되고 있는지를 검사하는데 들이는 시간이 점점 늘어나게 된다. 이 시간은 주어진 작업을 진행하는 속도를 떨어뜨리게 된다.

다섯째로 이런 환경에서는 응용프로그램간의 의존성이 매우 중요하여져서 여러 명의 프로그래머가 서로 협조하여 실시간 시스템개발을 개발하는 것이 거의 불가능하게 된다. 또한 작성된 시스템을 다른 하드웨어에 이식을 하는 것도 매우 어려운 일이 된다. 결론적으로 실시간 시스템의 소프트웨어의 개발이 매우 어렵고 이식성도 떨어지게 된다.

J · BOSN 실시간 운영체제의 해결책

J · BOSN RTOS는 임베디드 마이크로프로세서에 실시간 시스템을 구현하기 위해 설계되었고, 기능이 모듈화 되어 있는 고성능의 실시간 운영체제이다. 거의 모든 부분이 C로 작성이 되어서 이식성이 매우 뛰어나고 누구나 쉽게 완벽한 멀티태스킹 시스템을 구현할 수 있는 환경을 제공하여야 한다. 사용자가 사용하기 쉽고, 기능들 중 필요한 것만 선택하여 원하는 최적의 실시간 시스템을 구현할 수 있다. 운영체제 기능의 인터페이스도 대부분이 C 라이브러리 형태로 구성이 되어 있기 때문에 J · BOSN 운영체제를 기반으로 한 실시간 응용프로그램은 이 라이브러리와 링크를 하여 사용하게 된다. 실제 기본적인 기능을 구현된 J · BOSN 운용체제의 나노커널의 최소 크기는 약 8KB 정도이다.

결론은 J · BOSN 실시간운영체제는 위의 문제를 깨끗이 해결하였다. 프로세스간의 독립성을 확실히 보장하고, 시스

템의 자원의 관리를 통합적으로 관리를 하며, CPU의 할당문제를 응용프로그램에서 담당하지 않고 커널이 직접 제어를 하기 때문에 보다 높은 우선순위를 가지는 프로세스는 곧바로 수행을 시작할 수 있다. 높은 우선순위를 가지고 있는 프로세스의 실행이 멈춘 후에 곧바로 다시 수행이 되게 된다. 그러므로 가장 늦은 응답속도는 현재의 프로세스를 멈추고 다른 프로세스를 수행하는데 걸리는 시간이 된다. 이 시간은 하드웨어의 성능과 매우 밀접한 관계를 가지고 있지만, J · BOSN 운영체제는 이 시간이 극히 짧게 설계가 되어 매우 빠르고 일정한 응답속도를 제공한다.

또한 응용프로그램의 변경이나 시스템의 변경, 그리고 새로운 응용프로그램의 추가도 시스템의 응답속도에 아무런 영향을 미치지 않는다. 프로세스 수행을 관리하는 것 외에도 J · BOSN 운영체제는 프로세스간의 통신이나, 동기화, 시간관리 그리고 메모리 관리 등의 기능 등 모든 시스템자원의 관리기능을 제공하여 준다. 소프트웨어의 개발관점에서 보면, 프로세스간의 의존성을 아주 작게 줄이고 표준화를 통한 모듈화 성격을 최대한 높여서 여러 명의 개발자가 서로 협력하여 작업을 하는데 아무런 문제가 발생하지 않는다. J · BOSN 운영체제는 하드웨어에 독립적인 실시간 환경을 제공하여 주어진 하드웨어를 이해하려는 노력을 하는 대신에 실시간 응용프로그램을 개발하는데 집중할 수 있게 해주고, 작성된 프로그램이 수정 없이 다양한 시스템에서 동작할 수 있게 해준다.

결론적으로 J · BOSN 운영체제는 저렴한 개발 비용과 짧은 개발 기간 안에 성공적인 실시간시스템을 쉽게 개발할 수 있게 도와주며 개발된 시스템을 다른 시스템으로 이식하는 것이 쉬워서, 새로운 시스템을 개발하기 위한 투자를 최소한으로 줄여준다.

설계원리

실시간 운영체제인 J · BOSN은 모든 기능이 모듈화 되어 있고 각 모듈은 독립적인 쓰레드로서 움직이고 있다. 최소한의 기능을 나노커널에 구현을 하고 운영체제가 갖추어야 할 필수적인 모든 기능은 모듈화되어 다른 쓰레드들로부터 서비스 요청을 받아 서비스를 해주는 서버의 개념으로 작성이 되었다. 따라서 운영체제의 기능들은 완벽하게 서로 분리되어 있고, 또한 서로 의존적이지 않다. 이와 같은 구조에서는 필요한 기능만을 사용자가 골라서 자신에게 맞는 최소, 최적의 시스템을 구현할 수 있도록 하여 준다.

모든 기능 자체가 서버 모듈로 구성이 되어 있기 때문에 개개의 서비스 서버를 여러 개의 쓰레드들이 동시에 접근을 하더라도 다른 운영체제에서 발생하는 쓰레드간의 충돌이 전혀 발생하지 않아 문제를 매우 효과적으로 해결을 할 수 있다. 특히 다른 운영체제에서 디바이스 드라이버를 작성하여 보신 분들께서는 느끼고 계시겠지만, 여러 개의 프로세스가 동시에 서비스를 이용하려 할 때나 또는 서비스를 받고 있는 중간에 인터럽트 발생으로 인한 디바이스 드라이버의 이상 동작 등을 많이 경험했을 것이다.

또한 많은 프로세스가 동시에 드라이버에 접근하기 때문에 자원의 사용문제가 발생하고, 특히 변수들의 효과적인 관리가 문제가 된다. 이러한 문제를 해결하려면 디바이스 드라이버의 서비스를 이용하려는 프로세스의 접근을 제한하기 위해 O/S에서 제공하는 동기화 서비스를 또 받아야 하고, 인터럽트의 방해를 차단하기 위하여 인터럽트를 차단하고 시스템자원을 접근하는 프로세스 숫자만큼 할당을 하거나 또는 공통으로 사용하는 변수를 사용하려면 동기화서비스를 받아야 한다. 이런 모든 것이 시스템의 성능을 저하시킨다.

반면에 J · BOSN은 앞의 문제가 전혀 발생하지 않도록 쓰레드는 단지 서비스를 요청만 하고 자신이 직접 서비스를 받으려고 디바이스 드라이버를 제어하지 않는다. 또한 디바이스 드라이버는 여러 쓰레드가 동시에 자신을 제어하지 않고 자신이 선택적으로 서비스를 결정하므로 서비스를 이용할 때 충돌이나 인터럽트를 제한하는 경우는 발생하지 않게 된다. 부가적으로 드라이버에 문제가 발생하여도 이것이 서비스를

요구한 쓰레드들의 동작이상으로만 제한되어, 발생한 에러가 전파되지 않는다.

이 서버 모듈 설계 개념의 장점은 J · BOSN 운영체제를 사용하는 설계자가 자신이 원하는 기능을 추가하고 싶을 때, 운영체제에 관한 완벽한 이해가 없고 약간의 애플리케이션을 작성하는 능력만 있더라도 자신만을 위한 기능을 추가할 수 있게 만든다. 디바이스 드라이버 또한 서비스 서버형태로 작성이 되므로 운영체제에서 아무런 제약을 주지 않아 하드웨어의 제어만을 숙지하면 곧바로 완벽한 드라이버를 작성할 수 있다. 이러한 장점은 드라이버 작성에 많은 어려움을 겪는 개발자에게 시스템 설계의 신세계를 열어 줄 것이다. 누구나 J · BOSN 운영체제를 이용하면 거의 제약조건이 없이 설계자의 의도가 최대한 반영이 되면서 안정적이고 완벽한 시스템을 구축할 수 있다.

실시간 운영체제에서는 작성된 프로세스가 예정된 시간 내에 작업을 수행하여 원하는 시간 내에 서비스를 제공할 수 있는 능력이 중요하다. 이러한 목표를 이루기 위해서는 인터럽트의 레이턴시를 최소한으로 줄여야하고 인터럽트와 프로세스의 우선순위 역전(Priority Inversion) 현상을 막아야 하고 또한 프로세스의 우선순위에 맞게 실시간 운영체제가 높은 우선순위 프로세스의 서비스 요구를 빠른 시간 내에 제공해 주어야 한다. 이러한 조건을 만족하여 주기 위해서 J · BOSN 운영체제는 매우 효율적으로 설계가 되어 있다.

우선 인터럽트의 레이턴시를 줄이기 위해 인터럽트는 받아들이는 부분인 나노커널의 'Trap' 부분은 단지 인터럽트를 해당하는 드라이버에 전달만 해주고 직접처리를 하지 않으므로 곧바로 다른 인터럽트를 받을 수 있는 상태로 전환이 된다. 이러한 설계는 인터럽트 레이턴시를 최대한으로 줄일 수 있고 인터럽트의 발생과 처리부분이 분리되어 인터럽트 처리의 역전현상을 운영체제 설계 원리 자체에서 제거하였다.

인터럽트의 처리가 디바이스 드라이버의 우선순위에서 결정이 되므로 디바이스 드라이버의 우선순위가 곧 인터럽트의 우선순위로 설정할 수도 있다. 이것은 마치 인터럽트에도 우선순위를 부여하는 것과 동일한 효과를 보게 된다. 우선순위가 낮은 인터럽트는 높은 우선순위를 갖는 인터럽트보다 절대로 먼저 처리될 수 없는 이유이다. 또한 디바이스 드라이버는 J · BOSN의 서비스 서버들 보다는 우선순위가 낮으나 어플리케이션 보다 높게 설정되는 것이 일반적이다.

실시간 운영체제에서 우선순위가 높은 프로세서가 서비스를 요청할 경우 보다 낮은 우선순위 프로세스의 서비스 요청은 무시되고 곧바로 높은 우선순위 프로세스의 서비스를 제공해 주어야 할 경우도 있다. 이것은 전적으로 드라이버의 특성과 시스템설계의 목적에 맞게 제작이 되어야 하는데, J · BOSN 운영체제는 디바이스 드라이버가 스스로 서비스를 결정할 수 있기 때문에 이러한 요구사항에 맞게 드라이버가 모든 것을 결정할 수 있다.

예를 들면, 사운드 드라이버는 소리가 끊김 없이 재생을 하여 주어야 한다. 그러나 소리 데이터를 저장장치에서 읽어 온다고 가정을 해보자. 저장장치 드라이버 또한 실시간 재생을 위하여 매우 빠르게 데이터를 제공해 주어야 한다. 여기서는 두개의 프로세스를 가정해 보자. 하나는 저장장치에서 데이터를 읽어서 사운드 드라이버에 데이터를 제공하여 소리를 재생하도록 하는 것이다. 또 다른 하나는 저장장치에서 데이터를 읽어서 화면에 데이터를 나타내 주는 프로세스이다. 첫 번째는 실시간성을 요구하고 두 번째 것은 실시간성을 요구하지 않는 프로세스이다.

동시에 두개의 프로세스가 저장장치에서 데이터를 읽으려 한다면, 당연히 실시간 프로세스인 첫 번째 프로세스가 먼저 데이터를 서비스 받아 가져가야 한다. 그리고 남는 시간에 비실시간 프로세스에 서비스를 해주어야 한다. 이것은 누가 먼저 서비스를 요구 했는가는 중요하지 않고 서비스를 요구한 프로세스의 우선순위만을 고려하여 서비스를 제공하여야 하는 것이다. 그러나 저장장치로부터 데이터를 읽어 들이는 동작은 소리를 재생하는 동작보다 중요도에서 훨씬 떨어지므로 설계자들은 사운드 드라이버를 저장장치 드라이버보다 높은 우선순위를 설정하게 된다.

하지만, 만일 높은 우선순위의 프로세스가 저장장치로부터 데이터를 서비스를 요청하였고, 동시에 보다 낮은 우선순위의 프로세스가 사운드 재생을 요청하였다면 어떤 결과가 나오게 될까? 당연히 사운드소리가 먼저 나오고 저장장치 드라이버는 사운드 드라이버보다 나중에 서비스를 제공하여 줄 것이다. 이것은 높은 우선순위 프로세스가 낮은 우선순위 프로세스 보다 늦게 서비스를 제공받는 우선순위 역전현상이 발생하게 된다.

이러한 경우를 방지하기위하여 J · BOSN 드라이버의 우선순위를 현재 서비스를 요청한 프로세스의 우선순위에 따라 변할 수 있도록 하는 옵션을 제공하고 있어 우선순위 역전현상을 최대한 줄일 수 있도록 설계가 되었다.

J · BOSN 운영체제 설계원칙은 다음과 같은 매우 우수한 특성으로 구현이 되었다.

- Multi-Layer Kernel Structure: Modularity, Portability, Scalability
- Multi-Tasking and Flexible IPC
- Precise Timer and Priority Based Real-Time Scheduler
- Scalable Hard Real-Time
- Constant Device Management and Efficient I/O Server System
- Cost Efficient Use of Memory
- Reliable and Robust System Service
- Easy to Use
- Low Latency

지금부터는 실제 J · BOSN 운영체제의 구현에 관한 블록도를 설명하겠다.

나노커널(NanoKernel)

나노 커널에서는 하드웨어 인터페이스를 담당하고 운영체제를 구성하고 있는 각 기능서버들의 통신 연결기능과 운영체제의 최소한의 기능만을 담당하고 있다. 나노 커널은 J · BOSN 운영체제의 핵심 부분이며, Trap을 운영체제로 받아들이고 전달해 주고 각종 운영체제의 리소스 관리를 위한 도구를 라이브러리 형태로 제공을 하여 주며, 메시지 통신을 위한 기본 채널

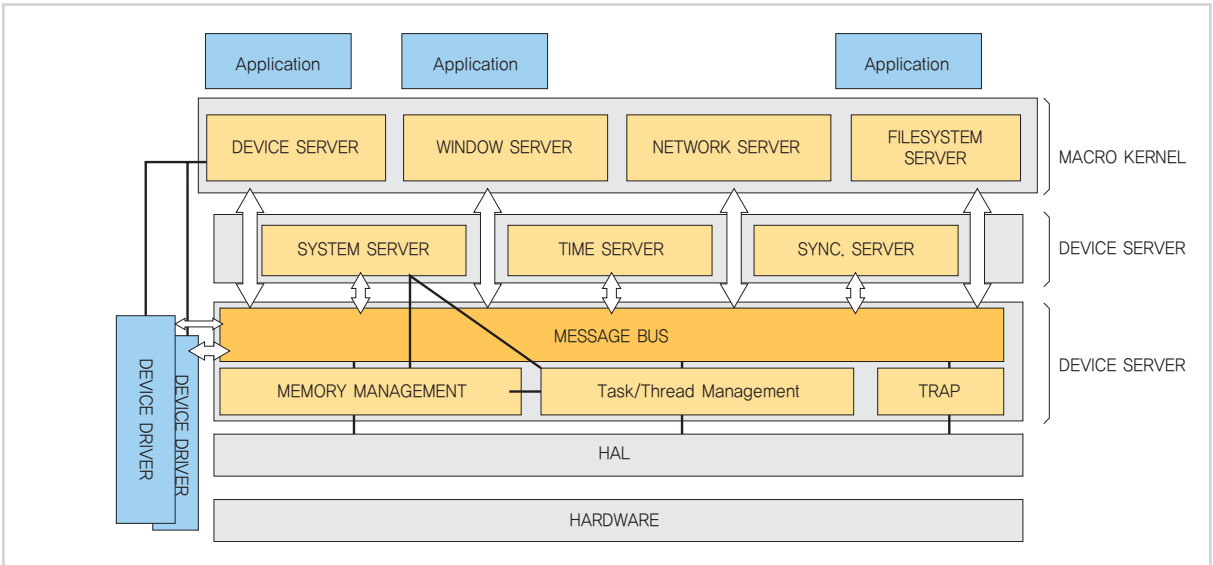

그림 1. J-BOSN 운영체제의 블록도

관리 및 전달을 위한 완벽한 기능을 구현하고 있다.

트랩(Trap)

이곳에서는 인터럽트와 시스템 콜을 받아들이는 통로이며 각각의 기능을 직접 구현하지는 않지만 필요한 운영체제 모듈에 전달하는 역할을 담당하고 있다. 결과적으로 서비스를 직접 처리하지 않기 때문에 이 모들은 언제나 인터럽트를 받아들일 준비가 되어 있어 인터럽트/프로세스의 우선순위의 역전현상과 레이턴시를 최소한으로 줄일 수 있다. 인터럽트를 이용하고 싶거나 시스템의 서버의 서비스를 이용하고 싶은 프로세스는 이곳에서 제공하는 라이브러리를 통하여 서비스를 요청할 수도 있다.

쓰레드관리(Thread Management)

이 모듈은 J · BOSN 운영체제의 프로세스 관리의 모든 기능을 라이브러리로 제공하고 있다. J · BOSN 운영체제에서 프로세스를 관리하는 최소한의 단위는 쓰레드이다. 이 쓰레드들이 여러 개 모여 하나의 목적을 달성하기 위한 일을 하고 있을 때 이 쓰레드의 집합을 태스크라고 하고 태스크 구조체에 통합되어 관리가 된다. 결론적으로 태스크는 최소한 한 개 이상의 쓰레드들이 모여 있는 것을 말하고 실제로 실행이 되는 객체는 아니다. 이곳은 쓰레드와 태스크의 생성과 종료 그리고 여러 가지 속성의 부여등과 같은 모든 관리를 위한 툴을 제공하고 또한 시스템 콜을 받아 직접 제어하기도 한다. 하지만 기본적으로 이곳은 J · BOSN 의 모든 리소스를 관리하는 시스템 서버의 관리를 위한 도구만을 제공하는 것을 원칙으로 하고, 이 범위를 벗어나지 않는 한도 내에서 다양한 기능을 시스템 콜로 제공을 한다.

메모리관리(Memory Management)

이 모듈은 J · BOSN 운영체제의 메모리 자원(Resource)를 관리하는 기본 도구를 제공하고 있다. 모든 운영체제에서 메모리는 매우 중요한 자원이고 운영체제 자신도 이 자원을 사용하여 수행이 되기 때문에 문제가 발생할 경우 시스템에 치명적이다. J · BOSN 운영체제는 메모리 자원을 페이지라는 단위로 나누어 관리를 하고 있다. 이 페이지는 4KB를 단위로 하여 관리가 된다. 이러한 크기로 관리를 함으로써 메모리 자원의 낭비를 상당히 줄일 수 있다. 리얼타임 운영체제는 실제로 많은 사용자가 사용하는 멀티유저 기능을 제공하지 않고, 또한 프로세스들이 많은 자원을 요구하지 않고 비교적 적은 자원만을 사용하여 시스템을 구성하기 때문에 큰 단위로 메모리를 관리하는 것은 메모리자원을 낭비하는 결과를 가져온다. 물론 사용자는 페이지의 기본사이즈(4KB)의 2의 배수 형태의 메모리를 할당받을 수 있어 보다 큰 메모리 자원도 사용할 수 있다.

블록의 관리를 위하여 메모리를 적게 차지하는 비트맵 방법을 채택하여 속도는 다소 저하 되더라도 메모리를 매우 많이 절약하는 방법을 채택하였다. 블록의 크기보다 적은 메모리를 관리하기 위하여 J · BOSN 은 변형된 Slab Allocation 알고리듬을 사용하고 있다. 사용자는 이런 방법을 시스템 서버에 메모리 자원 할당을 요구하면서 사용할 수 있다.

메시지 통신 관리(Message Bus Management)

이 모듈은 J · BOSN 운영체제에서 구현되는 모든 서버들 간의 통신과 서버에 서비스를 요구 할 때 사용할 수 있는 메시지 통신 관리에 대한 기본 도구를 제공하고 있다. 이 메시지 통신방법은 J · BOSN 운영체제의 모든 서비스 서버를 연결시켜주는 통신 채널을 확보하여 J · BOSN 운용체제 설계의 기본 근간을 이루는 중요한 모듈이다. 이 모듈은 시스템콜을 사용하여 호출이 될 수 있다. 이 모듈은 J · BOSN 운영체제에서 제일 많이 사용되는 모듈이므로 운영체제 자체의 성능에 상당한 영향을 미치게 된다. 따라서 이 모듈은 최대한 간단히 설계가 되어 있어서 사용할 때 상당한 주의를 요구한다.

앞으로 나열되는 모든 서비스 서버들은 이 메시지 통신을 통하여 상호간의 서비스를 요청하거나, 서비스 요청을 받아서 해당하는 서비스를 제공하게 된다. RealTime