□ 기술해설 □

엄격한 실시간 운영체제 커널의 현황 연구

한국과학기술원 박재현* · 이흥규**

목 차

1. 개 요

하드 리얼 타임 시스템은 그 시스템의 정확성(correctness)이 계산의 논리적 결과의 정확성(correctness) 뿐만 아니라, 결과가 나오는 시간에 의존하는 시스템이다[1]. 이러한 시스템의 예로는 Command and Control System, 공정 제어 시스템, 비행 제어 시스템, 우주 왕복선 비행 시스템, 그리고 미래의 우주 정거장, SDI와 같은 우주에 기반을 둔 방어 시스템을 들 수 있겠다[1]. 이러한 시스템들은 정확한 동작을 하기 위해서 여러가지 특성을 지녀야 한다. 이러한 특성들로는 Timing Constraints, Resource Constraints, Precedence Constraints, Concurrency Constraints, Communication Requirements, Placement Constraints, 그리고 Criticalness를 들 수 있는데, 이러한 특성을 지키기 위해 하드 리얼타임 시스템은 빠른 속도와 예측 가능성(Predictability), 신뢰성 그리고 적응성을 지녀야 한다[1].

본 고에서는 이러한 하드 리얼 타임 시스템에 있어서 위에서 말하는 신뢰성과 예측 가능성, 빠른 응답 속도 그리고 적응성 제공을 위한 기반이 되는 메커니즘을 제공하기 위한 하드 리얼 타임 커널 아키텍쳐에 대해서 알아본다.

이를 위해 기존의 상용으로 사용중인 하드 리얼타임 시스템들과 현재 연구가 진행중인 시스템을 중심으로 기존의 시스템을 설계시 고려한 점들을 살피고, 이들이 간과한 점들을 지적하고 이 점들을 해결하는 방안에 대해 살펴본다.

2. 상용 시스템들

2.1 VRTX (Versatile Real-Time Executive)

이 시스템은 현재 상용으로 사용되고 있는 대표적인 실시간 처리 시스템중 하나이다 [7]. 이는 스케쥴링과 동기화 메커니즘 측면에서 기존의 타임 슬라이싱 운영체제의 커널과 유사하며, 제공하는 기능의 수준에서 그리고 기억장치의 관리 측면과 입출력 관리 측면에서 다른점들을 지니는 작고 빠른 우선순위 스케쥴링을 제공하는 커널로 구성되어 있다.

이 시스템은 이벤트들에 대해서 명시된 유한한 시간내에 응답을 보이며, 멀티 태스킹의 기능을 제공한다. 이 시스템은 기본적으로 충분한 계산

*준회원
**종신회원

파워를 가정한다. 이 시스템은 Rate monotonic scheduling에 적합하다.

2.1.1 제공하는 기능들

커널은 다음과 같은 기능들을 제공한다.

· Task Services : Preemptive Priority Based Scheduling을 기반으로 다음과 같은 시스템 서비스를 제공한다. 즉 Create, Delete, Suspend, Resume, Change-Priority, Turn Time-slicing On/Off Preemption On/Off, priority-and-status의 보고하는 기능을 제공한다.

· Interprocess Communication : Mailbox와 Queue에 기반을 둔 프로세스간의 통신 메커니즘을 제공한다. 이때 특기할 점은 Queue는 우선순위에 의해 관리된다는 점이며, Mailbox는 다수 대 다수의 통신을 허용한다는 점이다.

· Memory Management : 정해진 크기의 블록을 관리하며, 예측가능성의 증대를 위해 가상기억장치를 지원하지 않는다. 또한 가용한 블록을 선택하기 위해서 최적인 것을 선택하기(Best fit)를 사용한다.

· Link and Location Independence : 여러 Component vector table(예; VRTX Configuration Table, VRTX Component Table)을 이용하여 각 부분의 기능들에 접근을 하고, 잘 정의된 프로토콜을 사용하여 제어와 매개 변수를 전한다.

· Real-Time Clock Independence

· Character Device Independence : Device와 VRTX Buffer간에 실행중인 태스크의 Rescheduling없이 한 문자를 이동시킬 수 있는 시스템 콜을 제공한다.

· Language and Specialized Device Independence : VRTX의 Overhead 없이 Application specific interrupt handlers를 호출할 수 있다. 또한 커널의 변경 없이 "user exit"와 같은 것을 행할 수 있도록 3개의 Hook을 제공한다.

이 시스템은 Motorla 68 Family, Intel 80×86, National serise 32000, Air Force Standard Architecture Mil-Std 1750A에 대해 가용하다.

2.2 Real-Time Unix

유닉스를 리얼 타임 시스템으로 수정하여 Open Real-Time System을 제공하고자 하는 시도는 많이 있어 왔다. 이러한 시도는 유닉스 시스템이 사용자 인터페이스나 통신 메커니즘 그리고 화일 시스템 측면에서 높은 수준의 서비스들을 제공한다는 장점을 가지고 있고, C programming 환경으로 유닉스 시스템 콜, 표준 입출력, 디바이스 드라이버들을 제공한다는 점 때문에 있어 왔다. 이는 일반 목적으로 사용하기 좋게 처리율(throughput)과 반응시간(Response Time)에 중점이 맞추어 제작된 유닉스를 실시간 처리에 적합하게 만드는 방향으로 행하여 왔다.

이러한 시도의 표준인 Real-time Posix는 priority based scheduler, multiple threads로 특징 지울 수 있는데, 이를 기존의 Unix와 비교하여 보면 다음과 같다.

	UNIX	Real-Time POSIX
Scheduling	time-sharing with fairness	preemptive fixed-priority
Time	interval time	absolute and relative time
Threads	single	multiple
Synchronous IO	blocked until data queued	transferred physically
Asynchronous IO		polled/signaled
RT file		preallocated, contiguous
Shared memory	(mmap, shmopen)	
Memory locking	(plm/unplm)	
Signal	unreliably registered	queued
IPC others	(pipe, socket)	semaphore, message

2.2.1 LynxOS

LynxOS는 다음에 중점을 두어 만들어졌다.

· 예측 불가능한 큰 지연 : 실시간 처리 응용 프로그램은 주기적 또는 비주기적 이벤트에 제한(bounded)된 시간 내에 반응을 보여야 하고, 응답시간(Response Time)은 하드웨어나 컴파일러의 효율성 뿐만 아니라, 각 이벤트에 대한 인터럽트와 태스크의 응답에 종속되어 있다.

· 다양한 외부 장치에 대한 효율적 인터페이스 : 최소한의 추가부담(Overhead)으로 고급언어로 작성되기 쉽도록 되어야 한다.

· Kernel Preemption : 낮은 우선순위를 지닌

태스크가 시스템 콜을 수행 중에 있다면 높은 우선순위를 지닌 태스크라도 preemption을 못한다. 이러한 우선순위 역전 현상(Priority Inversion)을 해결하고자 커널 Preemption을 행한다.

다음 표 2.1에서 우리는 LynxOS와 Unix의 차이점을 알 수 있다.

표 2.1 LynxOS와 Unix의 비교 분석

LynxOS	Unix
User Provide Absolute Priority	"nice" as guide line
Kernel is preemptive (Brief Critical Section, Counting Semaphore)	Non-preemptive kernel
2 Asynchronous software traps for signal	1 Asynchronous software traps for signal
Interrupt handler perform essential only, such as buffer critical data or handshaking with i/o device	All interrupt handler run at higher priority than regular process
Wide variety of device (I/O board, A/D converter, Servo motor controller)	
Unified I/O System (Group of sensor as file)	
Device driver can be dynamic configured	
Contiguous files, By-pass cache through DMA	Cache
Nonvolatile Disk Cache	Exhaustive check periodic disk synchronization
ROM based System(O S, Application, File system)	

2.2.2 REAL/IX

REAL/IX는 Fully Preemptive Scheduling을 가장 특징으로 하는데, 이는 현재 커널에서 수행중인 하위의 우선순위를 지닌 수행을 커널의 데이타 구조가 일관성을 지닐 때까지만 유지하고, 상위의 태스크를 수행하도록 한다. 이때 커널의 데이타 구조의 일관성의 제어를 위해서 Kernel Level Semaphore와 Spin Lock를 사용한다. REAL/IX는 다음과 같은 특성을 지니고 있다.

- 사용자 정의 우선순위 : 실시간 처리 우선순위를 지정할 수 있는 서비스를 제공한다.
- Fully preemptive kernel
- 보장된 인터럽트 응답 : 시간적인 측면과 기능적 측면에서 결정적(Deterministic)으로 움직임
- 빠르고 신뢰성 있는 프로세스간의 통신
- 빠른 속도의 데이터 모음 : 디스크에 데이터를 최적으로 저장하는 수단을 제공한다. 이는 내용을 저장하는 파일을 위해 물리적으로 연속적인 장소를 미리 할당하는 방법과 버퍼링을 사용자가 제어할 수 있는 수단을 제공함으로써 이루어 진다.
- 입출력 시스템 : Custom I/O Driver와 비동기적인 입출력 Driver를 기존의 시스템과 쉽게 결합할 수 있는 도구를 제공한다.
- 시스템 자원에 대한 사용자 제어 : Memory Locking, Device의 Locking을 허용한다.

3. 연구 시스템들

3.1 Spring Kernel

다양한 태스크에 동적으로 대처할 수 있는 실시간 처리 시스템을 만들기 위해서, 실시간 처리 시스템은 예측 가능성(Predictability)과 유연성(Flexibility)을 지녀야 한다.

이러한 시스템을 만들기 위해서, 즉 수행시간에 동적으로 데드라인를 보장하게(On-Line Dynamic Guarantees of Deadline) 하기 위해서, Segmentation 개념을 도입하고, 스케쥴링 알고리즘과 커널의 설계 부분을 접합(Integration)하는 접근방법을 사용한다.

이러한 시스템은 많은 수의 태스크들이 하드리얼타임 시스템이 사용되는 동안 계속해서 그 요구 사항이 바뀌어, 실시간 처리의 조건을 정적으로 만족시키는 모든 가능한 스케쥴을 미리 할당하는 것이 불가능하다는 점에 관심을 두고 있다

3.1.1 동 기(Motivation)

기존의 시스템들은 대개 앞서 살펴본 VRTX와 같이 실 시간 처리를 직접적으로(Explicitly) 해결하고자 하는 접근이 아니었음에 착안하여, 다음과 같은 특성을 지녀야 한다고 생각하였다.

예측 가능성을 제공하는 운영체제. 단기적으로

그리고 장기적으로 높은 수준의 적응성, 높은 수준의 협력(Cooperation)을 하는 다중처리기 상에서의 물리적인 분산(Distribution)을 시키는 것, 실시간 처리 그리고 신뢰성과 커다란 시스템 때문에 기인하는 결합된(Integrated) 해결책, 인공 지능 분야로의 접합, 그리고 복잡한 응용 문제를 다룰 수 있는 능력을 지녀야 된다고 생각하였다.

이러한 능력을 지니기 위해서 다음의 4가지 구성요소들을 개발을 시도했다.

- Dynamic, Distributed, On-line Scheduling Algorithm
- Network으로 연결된 다중 처리기에 적합한 스프링 커널
- 커널과 스케쥴링 알고리즘을 직접 받쳐주는 다중 처리기 노드
- 실시간 처리 영역에서 개발시 사용되는 시뮬레이터

3.1.2 시스템 모델

이 시스템에 있어서 기본적으로 가정하고 있는 시스템의 구조는 물리적으로 분산되어 있는 네트워크로 연결된 다중 처리기 시스템으로 구성되어 있고, 이들 각각의 다중처리기 시스템은 적어도 하나의 응용 프로세서와 하나 이상의 시스템 프로세서 그리고 입출력 서브 시스템으로 구성되어 있다. 모든 프로세서는 68020이고, VME 버스로부터 접근 가능한 로컬 메모리를 지니고 있으며, 메모리는 글로벌 메모리 영역을 구성할 수 있다. 또한 시스템 프로세서들은 스케쥴링 알고리즘이나 운영체제의 오버헤드를 책임져서, 응용 태스크들이 속도면에서나 예측 가능성 측면에서 침해 받는 것을 막는다.

3.1.3 Segmentation

Segmentation은 시스템의 자원을 응용 프로그램에서 요구되는 시스템의 특성을 고려하여 특정 크기의 단위로 나누는 과정을 의미한다.이러한 과정을 통해서 우리는 전 자원이 고정된 크기로 나누어져 있다고 가정할 수 있으며, 이는 실시간 처리 시스템의 복잡도를 낮추고, 시스템의 동작을 좀더 예측 가능하게 하며, On-Line Scheduling을 행하는 기능을 강화 한다. (이러한 세그먼트된 자원을 스케쥴링시 고려해서 자원이 있을 때만 수행이 가능하도록 함으로써 자원 사용의 충돌을 사전에 막을 수 있도록 한다. 이는 자원을 기다리며 이러한 형태로 우리는 예측 불가능한 지연을 막는다.)

다음에서 우리는 여러가지 자원의 Segmentation하는 것을 볼 수 있을 것이다.

- Memory Segmentation : 가상 기억장치를 사용하지 않음으로써 앞에서 설명한 바와 같은 실행 시간에 있어서 지연이 발생하는 Page Fault와 같은 일이 일어나지 않도록 하고, 메모리는 여러 크기의 고정된 크기의 블록으로 나눈다.
- I/O Segmentation : DMA의 경우를 보면 cycle steal method를 사용하는 것이 아니라 Preallocation을 이용한 Memory time-slice method를 사용하는 것을 들 수 있다.
- Task Segmentation : 분산된 실시간 운영체제 같은 시스템은 각각의 전담한 기능을 하는 모듈로 구성되어 있고, 이들 모듈은 CPU, 기억장치, 시스템 서비스, IPC와 같은 요구되는 자원에 기반을 두고 나누어(Segment)질 수 있다. 이때 나눔의 단위는 Thread의 멈춤(Block)에 의존하여, 멈춤이 있을 때 마다 나누어 지도록 한다. 이렇게 함으로써 CPU의 사용에서 임의의 지연이 생기는 것을 줄이는 방향으로 이끌어 갈 수 있다.

3.1.4 Scheduling Algorithms

위와 같은 형태로 자원을 나눈 후, 이를 Dispatcher, Local Scheduler, Global (Distributed) Scheduler, System의 장기적인 변화에 대처하기 위한 Meta-level Scheduler를 사용하여 스케쥴링 한다. 이때 태스크들은 (최악의 경우 수행 시간, 필요한 자원) 쌍으로 특징지워질 수 있고, 이때 자원들은 세그먼트를 단위로 할당되고, 스케쥴링에 의해서 자원 사용의 충돌을 회피하도록 하여, 자원의 충돌로 인한 비결정적인 지연을 막는다. 이러한 동적인 하드 리얼타임 스케쥴링 알고리즘은 NP-hard이기 때문에 Heuristic Al-

gorithm을 사용하여 결정한다.

그 밖의 나머지 커널이 제공하는 기능들은 IPC의 time-out기능을 제외하고 동일하다.

3.2 ARTS

ARTS는 실시간처리에 있어서 프로세스 모델 대신에 객체 지향 모델을 운영 체제의 기본 구성 모델로 제안한 시스템이다. 이렇게 객체 지향 모델을 사용함으로써 우리가 얻을 수 있는 것은 다음과 같다.

· 유지보수성과 코드의 재사용 가능성을 증대시킨다.

· 프로세스 기반 모델에서는 직접 표현되지 않는 시간 특성(Timing Characteristics)과 실시간 처리 시스템에 있어서 통신 구조를 직접적으로 표현하게 함으로써, 이를 우선순위 역전(Priority Inversion)을 피하는 효과적인 스케쥴링을 하는데 사용할 수 있다.

3.2.1 실시간 객체 지향 시스템

객체 지향 시스템은 실시간 시스템에 적합한 특성이 있고, 그렇지 않은 특성이 있다. 실시간 처리 시스템을 하나의 스레드를 지닌 객체의 집합으로 모델링 한다. 이 때 하나의 객체에 대한 서비스 요구간에 우선순위 역전 현상이 일어난다. 이러한 것을 대개의 경우처럼 FIFO로 처리하면 문제가 발생하는데, 즉 높은 우선순위를 지닌 요청이 낮은 우선순위를 지닌 요청을 제한없는 시간동안(Unbounded) 기다리는 일이 발생할 수 있다. 이는 Time Encapsulation의 부족을 초래한다. 즉 한 객체가 데드라인을 넘기면 같은 자원에 의존하고 있는 다른 객체들도 영향을 받아 데드라인을 넘기는 일이 발생한다.

좋은점으로는 객체간의 통신 구조와 시간적 특성이 기술하는 언어 수준에서 뿐만아니라 시스템 수준에서도 담겨져 있는데 이는 시스템에서 시스템 자원의 효과적인 스케쥴링에 유용하다. 또한 통신 서비스는 우선순위의 역전 현상을 막는데 유용하다.

또한, 시스템 전체 차원에서 각각의 객체의 시간적 특성을 운영체제가 안다면 Time encapsulation 효과를 가지게 된다.

또한 기존의 이론적인 Schedulability 분석 기술은 어떤 간단한 모델에 기반을 두고 있는데 이는 다음과 같은 점들을 포함하지 못하고 있다.

· 어떻게 태스크들간의 구조가 구성되어 있는지

· 어디에 태스크들을 위한 데이타가 저장되어 있는지

· 어떻게 태스크들이 논리적으로 나뉘어져 있는지

· 어떻게 구현상에서 나뉘어졌는지

이러한 이론상의 장점을 실제적으로 이용하기 위해서는 이러한 모델을 프로그래밍 모델로 바꾸어야 하는데 이는 쉽지 않다.

ARTS에서는 앞서 말한 여러 요구사항을 만족시키는 프로그래밍 모델을 만든다.

3.2.2 Real-Time Object Model

운영체제의 설계에서 당연히 프로그래밍 모델이 고려되어야 한다. 프로그래밍 모델에서 표현되는 정보를 상실없이 운영체제가 이용할 수 있어야 하며, 운영체제는 프로그래밍 모델에서 표현된 것들이 잘 수행될 수 있도록 서비스들을 제공해야 한다. 또한 프로그래밍 모델은 시스템을 만드는 자의 의도를 쉽게 표현할 수 있고, 사용자가 모든 의도를 잘 표현할 수 있는 표현 가능성(Expressiveness)을 지녀야 한다.

3.2.3 실시간 객체 지향 모델 설계시 고려 해야 해야 될 점

객체 모델이 개념화(conceptualization)와 문제의 분할면에서 장점을 지니고 있으나 여러가지 적합하지 않은 면이 있음을 생각하여 다음과 같은 점들을 모델 설계시 고려했다.

· Object Granularity : 속도면을 고려하여 크기가 큰 객체를 채택하였다.

· Priority Consistency : 객체가 제공하는 Method들은 각각이 의미면에서나 시간적 특성에 따라 각각 다른 우선순위로 수행되어야 한다.

또한 이러한 우선순위는 객체들간의 상호작용시에 전파되어야 한다. 이러한 우선순위의 일관성을 위해서 ARTS에서는 스레드와 메세지의 우선순위의 Granularity를 동일하게 정했다.

· Active Object v.s. Passive Object : Active Object가 선택되었다. 이는 Active Object가 Autonomous하고 Asynchronous한 동작을 함으로써 Data Encapsulation과 Time Encapsulation에 적합하다.

· Single Thread Object v.s. Multi Thread Object : Multi Thread Object가 선택되었다. 이는 단일 스레드를 지닌 객체가 Data Encapsulation이 더 좋은 반면, 다중 스레드 객체로부터 효율성과 스케쥴 가능성을 가져온다. 즉, 우선순위의 상속으로 인해 같은 Method에 대한 서로 다른 요청를 서로 다른 우선순위로 처리가 가능하다.

3.2.4 객체간의 상호 작용시 우선순위의 상속

객체에 오는 요청들에 대해 우선순위를 주는 방법은 객체내의 스레드에 정적인 우선순위를 주고 오는 요청에 적합한 스레드를 할당하는 방법과 스레드의 우선순위를 동적으로 오는 요청에 맞추어 조정하는 방법이 있다.

동적인 방법에 있어서 상속의 방법은 요청의 우선순위가 높을 때, 스레드의 우선순위를 높이는 방법으로 이루어진다.

3.3 Chaos(Concurrent Hierachical Adaptable Object System)

Autonomous Land Vehicle Project에서 사용되기 위해 만들어진 Heterogeneous Multiprocessor Real-Time and Parallel System을 위한 운영 체제이다. 이는 시스템을 상호작용하는 객체들의 집합으로 시스템을 구성하는 데 기반을 이루는, 높은 성능의 큰 스케일을 지니는 실시간처리 소프트웨어의 기반이 되는 커널 수준의 Primitives를 제공한다. 이 커널은 객체에 기반을 둠으로써, Modularity, Reconfigurability, Maintainability를 모델로부터 얻고 있다.

이 시스템은 시스템의 성능을 저하시키지 않고 객체 지향 모델을 사용하려면, 여러 의미(Semantic)에 적합한 Primitive를 만들고 이를 적합하게 이용해야 된다고 주장한다. 이들 통신 Primitives는 의미면에서 그리고 성능과 신뢰성, Lift Time에서 서로 다른 특성을 지니고 있다. 이 통신 Primitive들은 ARTS의 경우와 같이 스케쥴링 정보를 지니고 있다.

4. 비교 분석

이상에서 우리는 상용으로 사용중에 있거나, 연구가 진행중인 시스템에 있어서, 설계시 고려한 점들과 그 결과 그 시스템이 담고 있는 기능에 관해서 알아 보았다.

결국 VRTX와 같은 기존의 하드 리얼타임 커널은 Rate Monotonic Scheduling Algorithm Theory[3]를 사용하는 시스템의 수행을 위한 메커니즘을 제공하는데 적합하다. 또한 VRTX는 수행시간을 측정할 수 있는 도구를 제공하며, 각 시스템 서비스의 최악의 경우의 응답시간을 보장하여 하드 리얼타임을 보장한다. 그러나 이러한 계산과정은 서로간에 자원의 공유와 동기화 같은 상호작용이 없는 경우에 한해서, 그리고 주기적인 태스크에 한해서 Off-Line 스케쥴링이 가능하다. 즉 동적인 태스크의 발생에 대해서는 대처를 못한다.

유닉스의 사용자 프로세스 간의 우선순위 역전의 시간을 줄이고자 커널의 Preemption을 가능하도록 한 것이 LynxOS와 REAL/IX이다. 그러나 이는 낮은 사용자 프로세스를 위한 인터럽트의 처리가 그보다 높은 사용자 프로세스보다 우선순위가 항상 높은 Unbounded 우선순위 역전이 발생한다.

스프링 커널의 특징은 커널과 입출력을 담당하는 프로세서를 따로두어 시스템의 백그라운드 태스크나 사용자 프로그램의 시스템 콜 사용에 의한 비결정적 지연을 막고, 세그먼테이션을 통해 자원을 할당 가능한 단위로 분할하고 이를 미리 스케쥴링 알고리즘에 의해서 할당함으로써, 자원의 충돌로 인한 예측 못하는 지연을 막아, 동적으로 휴리스틱 알고리즘으로 스케쥴링을 하며 시간(Time Scale)에 대해 결정적(Determinis-

tic)으로 움직이는 시스템을 사용자에게 제공하고자 시도했다.

ARTS는 실시간 처리에 있어서 객체 모델을 도입하였고, 객체 모델이 적합한 이유를 통신구조의 형태와 시간적 특성을 직접적으로 표현함으로써, 이를 여러 부분에서(예: 우선순위 역전을 피하게함) 유용하게 이용하려고 시도하였다. 또한 객체간에 우선순위 상속을 행함으로써 공유하는 데이타에 접근할 때 발생하는 우선순위 역전현상을 막는다. 그러나 ARTS에서는 커널내에서의 객체에 대한 고려는 없다.

Chaos에서는 객체 모델의 하드 리얼타임의 영역에의 도입을 위해서는 효율적인 통신 메커니즘이 있다고 생각했으나, 이는 [1]에서 나타나는 하드 리얼타임 시스템에 대한 잘못된 이해에 나타난 것과 같은 견해이며, 물론 현재는 객체지향 모델 도입을 위해 적절한 Engineering Solution이지만, 이는 현재의 기술에 생각이 치우친 결과인 것 같다. 그러면 다음에서 하드 리얼타임 커널에 관한 생각을 정리하자.

5. 결 론

결국 하드 리얼타임으로 처리하기 위해서는 객체 지향 모델을 지원하는 커널이 ARTS에서 말하는 것과 같은 이유, 즉 유지 보수성 면에서나 코드의 재사용성 면 그리고 시간적 특성과 통신구조를 직접적으로 표현하는 면에서 좋을 것 같다.

또한 커널은 객체에 기반을 두고 설계되어야 겠다. 그 이유는 시스템 전체가 객체들로써 구성되는 일관성을 지니고, 이러한 것은 운영체제의 구성에 있어서도, 자켜져서 통신 메커니즘을 제외한 모든 것이 메세지를 페싱하는 객체로 구성되고, 이렇게 구성된 시스템은 정적인 토폴로지를 지닌 Distributed Multiprocessor system Hardware에 분산되면 일정한(Deterministic) 시간 특성을 같는 통신 메커니즘과 이 위에서 우선순위를 메세지를 통해서 전달(Propagation)하며 우선순위의 역전을 한번으로 한정하며(Bound), 우선순위를 상속하여 전체 시스템이 결정된 우선순위에 대해서 결정적으로 일관성 있는 동작을 보일 것이다. 이러한 우선순위의 일관성은 커널의 여러 서비스까지 이어질 것이다.

이때 I/O 처리기들에 의해서 인터럽트는 따로 처리될 수 있을 것이며, 이 인터럽트는 I/O 서버 객체에 의해서 메세지로 바뀌며, 이 메세지의 우선순위는 이 I/O와 연관된 객체에 의해서 결정될 수 있을 것이다. 이러한 시스템은 객체의 중복(Duplication)에 의해서, 하드 리얼타임 시스템에게 요구되는 특성 중의 하나인 신뢰성(Reliability)을 얻을 수 있을 것이다.

또한 객체는 문제를 논리적으로 의미있는 단위로 나누게 되므로, 해당하는 의미에 적합한 낮은 수준의 동작(Low level operation)을 정의할 수 있어서 데드라인 내에 수행을 할 수 없는 경우 그 동작을 할 수 있게 하는 기반을 제공할 수 있을 것이다.

참고문헌

[1] A. J. Stankovic, "Real-Time Computing Systems : The Next Generation," *Tutorials of Hard Real-Time Systems*, p. 14~38, 1988.

[2] S. C. Cheng and et al., "Scheduling Algorithm for Hard Real-Time Systems : A Brief Survey." *Tutorials of Hard Real-Time Systems*. p. 147~173. 1988.

[3] C. L. Liu and et al., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *JACM*, p. 46~61, Feb. 1973.

[4] R. R. Muntz and et al., "Preemptive Scheduling of Real-Time Tasks on Multiprocessor Systems," *JACM*, p. 324~328, April 1973.

[5] D. Peng and et al., "Modeling of Concurrent Task Execution in a Distributed System for Real-Time Control," *IEEE Trans on Comp*. p. 510~516. April 1987.

[6] W. Zhao and et al., "Preemptive Scheduling under Time and Resource Constraint," *IEEE Trans on Comp.*, p. 940~960. Aug. 1987.

[7] J. F. Ready, "VRTX : A Real-Time Operating System for Embedded Microprocessor Appli-

cations," *IEEE Micro,* p. 8~17, Aug. 1986.

[8] K. Schwan and et al., "Chaos : A kernel for predictable programs in dynamic real-time systems," in *Proceedings of the 7th IEEE Workshop on Real-Time Operating Systems and Software,* p. 11~19, 1990.

[9] L. D. Molesky and et al., "Preedictable real-time multiprocessor kernel-spring kernel," in *Proceedings of the 7th IEEE Workshop on Real-Time Operating Systems and Software,* p. 20~26, 1990.

[10] I. M. Singh and et al., "Unix rewritten for real-time, in Proceedings of the 7th IEEE Workshop on Real-Time Operating Systems and Software." p. 27~31, 1990.

[11] A. Pedar and et al., "Architecture optimization of aerospace computing systems," in *IEEE Trans. on Comp.,* p. 911~922, Oct. 1983.

[12] G. D. Carlow, "Architecture of the space suttle primary avionics software system," in *CACM,* p. 926~36, Sept. 1984.

[13] R. R. Razouk and et al., "Measuring operating system performance on mordern micro-processors," in *Performance 86,* p. 193~202, 1986.

[14] C. W. Mercer and et al., "The arts real-time object model," in *Proceedings of Real-Time Systems Symposium,* p. 2~10, 1990.

박 재 현

1988 중앙대학교 공과대학 전자계산학과 학사
1991 한국과학기술원 전산학과 석사
1991 ~현재 한국과학기술원 전산학과 박사
관심 분야 : 상호연결 네트워크, 실시간 운영체제, 분산/병렬 운영체제

이 홍 규

1978 서울대학교 공과대학 전자공학과 학사
1981 한국과학기술원 전산학과 석사
1984 한국과학기술원 전산학과 박사
1985 ~1986 Univ of Mishigan, U.S.A. Research Scientest
1986 ~현재 한국과학기술원 전산학과 부교수
관심 분야 : 고장감내 시스템, 엄격한 실시간 시스템