분산 멀티미디어 서버를 위한 IPC 구조 연구

°*신 동원, *정석용, *최 경희, **정기현
*아주대학교 정보 및 컴퓨터 공학부, **아주대학교 전기 전자 공학부

A Study of IPC Framework for Distributed MOD System

°*Dongwon Shin, *Suk-Yong Jung, *Kyunghee Choi, **Gihyun Jung
*Div. of I&CE Ajou Univ. *Div. of E&EE Ajou Univ.

요 약

본 논문에서는 분산 멀티미디어 서버를 위한 IPC Framework을 제안한다. 본 논문의 기반으로 사용되는 Scalable MOD System(SMOD)은 멀티미디어를 위한 연성 실시간 시스템(soft real-time system)으로서, 미디어를 분산 배치하여 실시간으로 통신 처리하는 분산 시스템(Distributed System)이다. SMOD는 높은 성능의 MOD Service를 위하여, 독자적인 System Model을 개발하였으며, SMOD의 구성요소로서 단위 처리기 PE(Processing Element)와, PE를 위한 실시간 커널 RT-Mike를 개발하였다. 이러한 구성요소들은, DPRAM, FIFO등의 하드웨어적인 통신장치와 Bus와 LAN, ICN, ATM을 기반으로 통신을 하게된다. RT-Mike는 이러한 다양한 통신 매체와 응용에 적합한 IPC를 효과적으로 제공하여야 한다.

본 논문에서는, 성능, 확장성, 실시간성 측면에서 IPC 요구 사항을 고찰하여, 분산 멀티미디어 서버를 위한 IPC Framework을 제안하였다.

1. 서론

높은 성능의 MOD Service를 위하여, 미디어를 Striping하여 분산 저장하는 독자적인 분산 멀티미디어 서버 SMOD(Scalable MOD System)를 개발중이다. 이 SMOD는 ICN(Inter-Connection Network), SE(Switching Element), PE(Processing Element)등의 하드웨어로 구성되며, 이를 위한 실시간 커널 RT-Mike도 개발되었다.

MOD Server, PE, ATM I/F등 SMOD의 구성요소들은 DPRAM, FIFO등의 하드웨어적인 통신장치와 Bus와 LAN, ICN, ATM을 기반으로 통신을 하게된다. RT-Mike는 이러한 다양한 통신 매체와 응용에 적합한 IPC를 효과적으로 제공하여야 하며, 이를 위해 구조화된 IPC Framework을 제안한다.

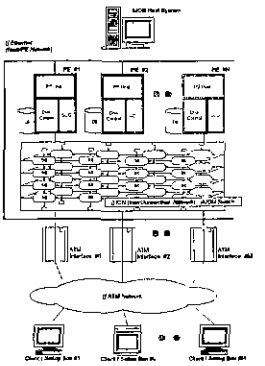
본 논문에서는 성능, 확장성, 실시간성의 세 가지 측면을 고려하여 IPC Framework을 제안한다. 실시간성의 측면에서 실시간 동기화(real-time synchronization)와 실시간 통신(real-time communication)을 지원하여 동기화와 IPC시에 발생하는 우선순위 역전현상을 최소화한다. 또한 인터럽트 관리자(interrupt manager)와 인터럽트를 고려한 우선순위 모델을 통해, 인터럽트에 관련된 실시간성 보장 및 성능문제를 해결하고자 하였다. 성능측면에서 Single Address Space OS의 특징을 살려, copy의 횟수와 protection overhead를 최소화하였으며, 기반 통신매체의 특성을 적극적으로 이용하였다. 확장성의 측면에서는, RT-Mike의 확장성 concept을 일관되게 따라 적절한 모듈화 설계가 이루어졌다. 이를 통해 커널의 경량화와 높은 확장성이 기대된다.

2. 기반 연구 고찰

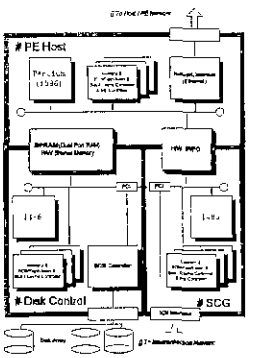
2.1 SMOD의 System 모델

[그림 1]에서 보듯, SMOD는, 통신을 기반으로 다수 PE간의 Continuous Media Striping기법을 적용한 분산 구조로서, 부하 분산과 확장성의 장점을 목표로 한다.[1]

SMOD는 PE(Processing Element)를 기본으로 한다. PE는 분산된 구조를 띤 SMOD의 기본 단위이며, 독립 운용 가능한 소규모 MOD Server라고 볼 수 있다. PE는 내부적으로 분리된 구조이며, 각 부분에 실시간 시스템이 탑재되어 독립적으로 운영된다. Pentium processor가 탑재된 System Processor Module(PE Host)은 디스크 I/O 스케쥴링 및 서비스 채널에 대해 관리하며, intel 486 processor가 탑재된 I/O Processor Module(Disk Control, SCG)은 고속의 디스크 입출력 및 ATM 스위치 접속부와의 접속 및 전송을 책임진다. PE의 구조는 [그림 2]와 같다.



[그림 1. SMOD의 구조]



[그림 2. PE의 구조]

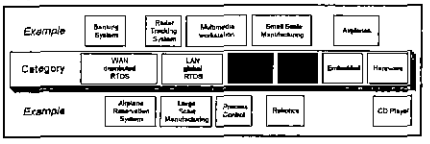
2.2 Real-Time Mike

Real-Time Mike는 SMOD를 위해 개발된 커널로써 PE와 ATM I/F에 탑재된다 RT-Mike는 exokernel과 같은 library extensible한 2세대 마이크로 커널로서, 운영체제의 많은 부분들이 선택적으로 적용 가능한 라이브러리 운영체제이다. 라이브러리 운영체제는 기존 운영체제에서 제공하는 자원 관리의 기능과 더불어, 기존 운영체제에서는 커널이 제공하던 스케쥴링 기능, 시스템 서비스 기능을 제공한다. 즉, 태스크, 메모리, 스토리지, 네트워크, 인터럽트, IPC에 대한 관리 및 서비스 기능을 제공한다.

3. IPC Framework의 필요성 및 요구사항

3.1 IPC Framework의 필요성

SMOD의 구조에 필요한 Real-Time OS를, [그림 3]과 같은 RTOS category[2]에서 표시해 보았다



[그림 3. RTOS Category]

이와 같이 통신을 기반으로 하는 대용량 처리 시스템은 Bus나 LAN을 기반으로 하는 분산 시스템이라 할 수 있겠다. 회색으로 표시된 부분이, SMOD가 속한 RTOS Category이다. 이와 같은 분산환경에서는 시스템의 구성에 따라서 다양한 IPC가 필요하게 된다. SMOD에서 통신 기반에 따라 구체적으로 IPC를 구분하면 다음과 같다. [그림 1]을 참조한다.

① 한 PE의 같은 processor 내(같은 프로세서 상)에서 수행되는 태스크 사이의 통신 : Local Memory 이용

② 한 PE의 다른 processor 사이(Bus로 연결된 프로세서들)에서 수행되는 태스크 사이의 통신

③ LAN(Ethernet)을 기반으로 연결된 시스템 상에서 수행되는 태스크 사이의 통신

④ WAN(ATM)을 기반으로 연결된 시스템 상에서 수행되는 태스크 사이의 통신

이처럼 다양한 통신 매체와 응용, 상황에 적합한 다양한 IPC가 필요하다. 이를 효과적으로 제공하기 위하여, 구조화된 IPC Framework이 필요하며, 이장의 나머지 절들에서 요구사항을 살펴한다.

3.2 성능(Performance) 측면의 요구사항

미디어를 분산 배치한 아주대 MOD System의 특성상, IPC가 매우 빈번하게 일어나기 때문에, IPC의 성능이 차지하는 비중이 크다. 다음은 IPC의 성능 향상을 위한 고려사항과 제안이다.

① Local IPC의 성능은 context switching, 보호영역(protection domain)의 전환 등에 의해 결정된다. RT-Mike는 Single Address Space OS[3]로서, context switching overhead가 적으며, 시스템 콜이 function call로 이루어지므로, 보호영역 전환 overhead가 없다. 때문에 IPC Framework설계시 local IPC의 성능개선은 고려하지 않는다.

② Remote IPC의 성능은 network의 자체성능과 network device I/O, interrupt management, message copy등에 의해 결정된다. SMOD의 기반 통신매체의 특성을 적극적으로 이용하여야 한다. 특히 DPRAM과 FIFO는, SMOD의 설계 시에 IPC 성능 측면을 최우선으로 고려하여 선택된 통신 장치로서, 하드웨어적인 semaphore와 다양한 signal, interrupt를 제공하여 효율적인 통신을 가능케 한다. 실시간성 보장과 성능, 빠른 응답시간을 위해 RTPU나 특별한 Bus환경을 사용하는 등의 시스템 구축사례[2]는 여럿 존재한다. 다음으로, Network device I/O 성능 향상을 위해, 효율적인 interrupt (priority) management가 필요하다. Network device에서 발생하는 interrupt에 의해, remote IPC는 OS에게 인지되므로, 적은 overhead의 interrupt의 처리와 masking이 필요하다. 이는 궁극적으로 network device에 대한 대기시간/지연(latency)을 줄여준다. Interrupt의 관리는 성능뿐만 아니라, 뒤이어 설명할 실시간성까지 같이 고려하여 설계되어야 한다.

3.3 확장성(Extensibility) 측면의 요구사항

RT-Mike는 확장성(Extensibility)이 높은 실시간 마이크로 커널로서, exokernel과 같은 라이브러리를 이용하여 확장성(library extensible)을 이루었다. 구체적으로, RT-Mike가 적용되는 환경에 맞게 응용수준에서 라이브리리 OS를 취사 선택할 수 있다. 이러한 확장성 덕분에 커널의 경량화를 이루게 되며, 시스템 콜(system call)에 걸리는 overhead도 줄어든다. IPC Framework 역시 이러한 확장성 concept을 일관되게 따라야 한다. RT-Mike는 적용되는 기반에 따라 다양한 통신 매체 중에서 일부분만 사용하므로, 각 매체에 대한 IPC를 선택적으로 포함하여 커널의 경량화를 이룰 수 있다.

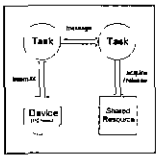
3.4 실시간성 - 우선순위 역전현상 측면의 요구사항

우선순위 역전현상은 실시간 시스템을 예측 불가능(unpredictable)하게 하고, 분석 불가능(un-analyzable)하게 한다. 이러한 우선순위 역전현상의 해결책으로 Priority inheritance protocol, Priority ceiling protocol등이 제안되었다. 우선순위 역전현상은 동기화(synchronization)와 통신(communication)에서 발생할 수 있다. 동기화는 임계영역과 같은 공유자원에 대한 것이며, 통신은 local/remote IPC를 의미한다.

4. IPC Framework의 제안

4.1 IPC 모델의 제안

이 절에서는 IPC에서의 실시간성 보장을 위한 IPC 모델을 제안한다. 스케쥴링, 동기화, IPC와 관련한 Task, Resource, Device, Message, Interrupt를 중심으로 [그림 4]와 같이 IPC 모델을 제안한다.



[그림 4. IPC Model Concept]

Task는 다른 Task로 message를 보내 통신할 수 있으며, 비동기적으로 이루어진다. 또한 Task는 Shared Resource를 독점적으로(mutual exclusion) 사용할 수 있다. 또한 IPC와 관련된 Device로부터 interrupt를 받아 이에 대한 처리를 할 수 있다. 구체적으로 Network device를 의미하며, 이에 대한 처리는 미리 등록해둔 응용레벨 인터럽트 핸들러(App-level interrupt handler)에서 이루어진다. 운영체제는 일반적으로 비동기 이벤트(asynchronous event)에 반응하기 위해 인터럽트에 의존한다.[4] 즉, 인터럽트는 remote IPC를 인지하는 기본적인 수단이다.

SMOD의 분산환경에서 IPC, Mutual Exclusion, Interrupt에 대한 다양한 경우를 표현하기 위하여, [그림 5]과 같은 구체

적인 IPC 모델을 제안한다. 구성 요소에 대한 세부적인 분류는 [그림 6]과 같다. 제안한 우선순위 모델을 이용하여 본 시스템에서 발생하는 Deadlock과 priority inversion을 쉽게 표현할 수 있다.

4.2 IPC Framework의 제안

실시간성 보장측면에서 synchronization과 communication을 같이 고려한 IPC Framework을 설계한다. 이 구조는 real-time synchronization과 real-time communication을 가능하게 하여, 실시간성을 효과적으로 보장한다. IPC Framework을 고려한 RT-Mike의 구조와 IPC Framework의 세부적인 구조를, 각각 [그림 7]과 [그림 8]에 보인다.

IPC Framework은 United Priority Management module과 RT-Synchronization module, RT-Communication module로 나누어진다.

RT-Sync module에서의 Interrupt 처리가 IPC Framework의 핵심이며, United Priority Management module에서 앞서 언급한 통합적인 우선순위 관리가 이루어진다.

5. 결 론

본 논문에서는 분산 멀티미디어 서버를 위한 IPC Framework을 제안하였다. 성능, 확장성, 실시간성을 고려하여, 멀티미디어 서비스가 원활하게 이루어지도록 설계하였다. 하지만, IPC Framework을 실제로 구현하고 실험하여, 제안을 검증하고 보완하는 연구가 이어져야 하겠다.

참고 문헌

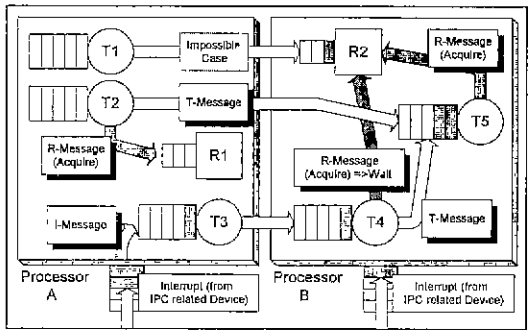
[1] 최 경희 외, "대용량 처리 시스템에서의 실시간 스케줄러 개발", 최종연구보고서, 한국전자통신연구소, 1996.

[2] D. B. Stewart, D E Schmitz and P. K. Khosla, "The Chimera I I Real-Time Operating System for Advanced Sensor-Based Control Applications", *IEEE Transaction on System, Man and Cybernetics*, Vol.22, No.6, pp.1282-1295, Nov/Dec 1992.

[3] J. Chase, M Feeley, H Levy, "Some Issues for Single Address Space Systems",

[4] Peter Kettle, Steve Statler, "Writing Device Drivers for SCO Unix, A Practical Approach", Addison Wesley, 1992.

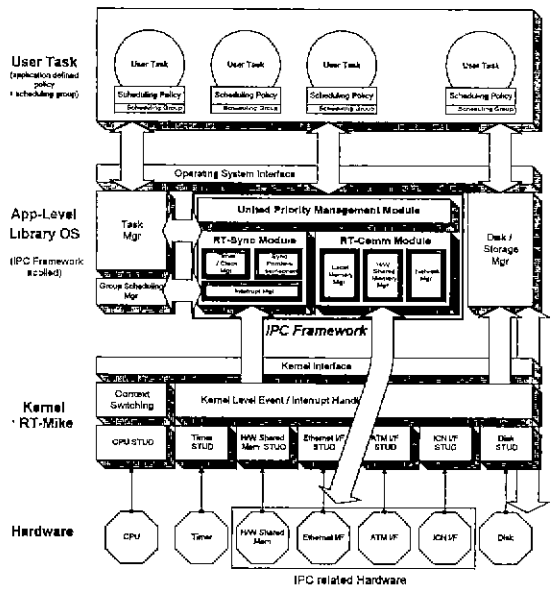
[5] T. Kitayama, T Nakajima and H. Tokuda, "RT-IPC: An IPC Extension for Real-Time Mach", *In the Proceedings of 2nd Microkernels and Other kernel Architectures*, USENIX, 1993.



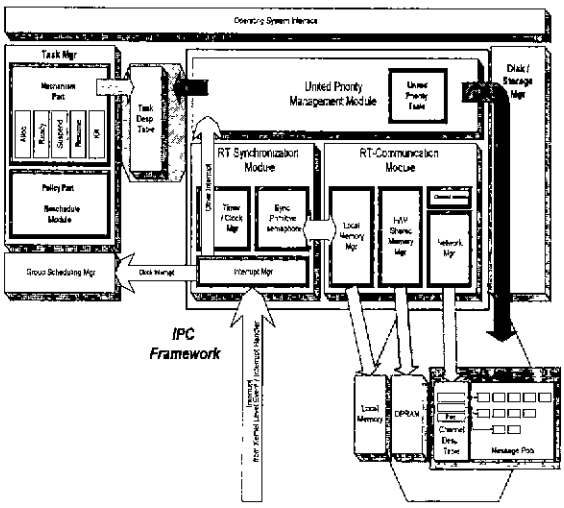
[그림 5. IPC Model]

Object	Task	Real-Time Task & Non Real-Time Task
	Resource	Shared Resource (include Critical Section)
Message	T-Message	Task Message (Task <-> Task)
	I-Message	Interrupt Message (Device -> Task)
Interrupt	R-Message	Resource Message (Task -> Resource)
	Interrupt	Hardware Interrupt from IPC related Device

[그림 6. Element of IPC Model]



[그림 7. IPC Framework을 고려한 RT-Mike의 구조]



[그림 8. IPC Framework의 구조]