특 집

실시간 시스템을 위한 미들웨어

김 정 국*

● 목 차 ●

1. 서 론

"실시간"이라는 말처럼 다양한 의미를 갖고 사용되는 IT 관련 용어도 드물 것이다. 때로는 단순히 빠르거나 온라인이라는 뜻으로, 때로는 데드라인에 대한 보장성 컴퓨팅의 의미로, 때론 시스템 설계 시에 시간조건(timing constraints)을 줄 수 있는 것으로, 때론 멀티 태스킹의 의미로도 사용된다.

이들 중 실시간 처리를 연구하는 학계에서는 "보장성과 예측성"을 가지는 컴퓨팅 패러다임과 기법을 연구하는 것이 보통이지만, 산업계에서는 대부분 이벤트에 대한 처리지연이 최소화되는 멀티 태스킹을 지원하는 소규모 운영체제 기반의 응용 개발에 의미와 초점을 맞추고 있다고 생각된다. 이중 "어떤 것이 올바른 용어의 사용이고 어떤 것이 그르다"라고 하는 것을 떠나서, 본 기고는 일반적으로 학계가 지향하는 목표, 즉, 시간 조건에 대해 보장성과 예측성이 주어지거나(경성 실시간 시스템), 시간 조건의 위반에 대해 대응적 처리를 취할 수 있는 실시간 시스템(연성 실시간 시스템)에 초점을 맞추어, 좀더 굳건하고, 편리한 실시간 시스템 구축을 위한 실시간 운영체제, 실시간 미들웨어, 실시간 응용의 세 부분에 대한 각각의 역할, 기법, 기능적 요구 사항을 점검해 보고자 한다.

이러한 점검에는 실시간 컴퓨팅이라는 목표와 함께, 근자에 들어서 많은 실시간 컴퓨팅 시스템의 구축에 요구되고 있는 "분산 컴퓨팅"과 "객체 지향 설계"의 접목에 대해서도 언급하고자 한다. 이와 아울러 새로운 실시간 패러다임으로 부상하고 있는 분산 실시간 객체 모델인 TMO[1]와 이의 수행을 위한 분산 미들웨어 엔진[4,5,7]에 대해서도 소개한다. 그리고 결언으로서, 아직 까지는 멀고 어렵게 느껴지는 실시간 보장성 컴퓨팅으로 가기 위해 해결되어야 할 문제들을 열거해 본다.

2. 실시간 운영체제와 미들웨어의 역할

본 기고에서 초점을 맞추고자 하는 실시간 시스템의 특성은 다음과 같이 요약될 수 있다.

- 주기적 태스크 및 산발적(sporadic) 태스크에 대한 시간 조건의 적용
- 시간 조건에 대한 보장성(timeliness guaranteed) 컴퓨팅
- 예측성(Predictability)

* 한국외국어대학교 컴퓨터공학과 교수

위에서 시간 조건이라 함은 구동 시간, 완료시간 등으로 볼 수 있다. 이러한 실시간 컴퓨팅의 목표로 가기 위해 중요한 사실은 운영체제, 미들웨어, 응용 시스템의 세 가지 계층구조의 잘 정의된 상호 협조적 역할이 반드시 있어야 한다는 점이다. 흔히 동일한 프로그램을 일반 운영체제에서 소위 실시간 운영체제라고 지칭되는 새로운 플랫폼으로 이식하면 자동적으로 실시간 수행이 되는 것으로 오해되는 부분이 있는 것이 사실인데, 실제 위와 같은 완벽한 의미의 실시간 시스템의 구축은 현 세계적 기술 수준으로도 매우 어려운 일이고, 특히 어떤 특정 실시간 운영체제의 사용이나 실시간 미들웨어 또는 언어의 사용만으로 해결되는 일이 절대 아닌 것이다. 그렇다면 경성 실시간 시스템을 위한 각 계층구조의 역할과 기능은 무엇인가? 각 응용 시스템의 형태에 따라 이러한 요구 사항은 달라 질 수 있으므로 일반적인 요구사항을 열거하기로 한다.

2.1 운영체제에 대한 요구사항

실시간 응용을 개발하기 위해 실시간 운영체제에 요구되는 일반적 사항은 다음과 같다.

- 시스템 호출에 대한 시간 보장: 미들웨어나 응용이 호출하는 시스템 및 라이브러리 호출에 대해 시간 보장이 이루어져야 한다. 물론 커널이 모든 자원에 대한 복수 개의 큐잉(Queueing) 시스템으로 구성되고, 태스크의 개수, 자원 점유 등에 따라 모든 상황이 변화하기 때문에 일반적인 형태의 보장은 거의 불가능한 것이 사실이다. 다만 특수한 응용의 경우, 정해진 최다 태스크 개수, 정해진 점유 자원 종류라는 전제 조건이 있을 때는 이러한 보장이 가능해 질 것이다.
- 실시간 자원관리의 제공: 흔히 운영체제의 실시간 화에 있어 프로세서 스케줄링의 기법에만 치중되는 경우가 있는데 프로세서와 마찬가지로 다른 공유 자원에 대해서도 반드시 실시간 관리가 병행되어야 할 것이다. 즉 운영체제 자체적으로 시간 조건에 대한 데드라인 기반 프로세서 및 자원관리 스케줄링 등이 함께 제공되는 것이 바람직하다. 현재 특수 실시간 운영체제가 아닌 범용 운영체제나 그의 실시간 버전은 흔히 일반 태스크에 대한 우선 배려 정책으로서 상대적으로 높은 고정 우선 순위의 실시간 태스크만을 제공하는 것이 보통이다. 이러한 스케줄링 정책은 범용 환경에서의 실시간 태스크의 우선 수행 기능 제공의 개념만을 제공한다고 보아야 할 것이다.
- 네트워크 또는 분산 환경을 위한 실시간 통신의 제공: 가격 대비 성능, 신뢰도를 고려하여 분산 시스템을 활용한 실시간 시스템의 구축이 점점 증가하고 있는 추세이므로, 운영체제와 네트워크 구성 요소의 협력 하에 실시간 통신이 제공되는 것이 바람직하다.

2.2 미들웨어의 역할

실시간 시스템의 구현은 실시간 운영체제와 미들웨어 등으로만 해결되는 것이 아니다. 즉 앞서 언급한 실시간 운영체제의 기본적 기능 위에서 프로그래머가 해야 할 일은, 각각의 태스크의 구동 시간 조건, 자원 점유 형태를 분석하고 이에 따른 태스크의 시간적 수행 행태(behaviour)를 결정하여 이를 설계 시부터 반영하여야 한다. 이를 쉽게 이해하려면 다음과 같은 극단적인 경우를 생각해 볼 수 있다. 즉, 자원 점유에 있어 CPU만 경쟁 상태에 있다고 가정하고, 여러 개의 주기적 태스크와 및 제한적인 산발적 태스크 및 각각에 대한 시간제한 조건이 주어졌을 때, 가장 확실하고 궁극적인 실시간 시스템의 구성 방법은 사용자에 의한 사전 스케줄링, 즉 정적 스케줄링이다.

제한적 자원 상황에서 고도의 신뢰성을 요구하는 경성 실시간 시스템의 경우, 이렇게 사용자에

의한 정적 스케줄링이 사용되는 경우가 있다. 또한 실시간 통신이 보장되는 분산 컴퓨팅의 경우는 극단적으로 태스크 하나마다 프로세서 노드 하나를 할당하는 경우도 있다. 이러한 예가 사용자가 대부분의 실시간 문제를 해결하는 극단적인 경우라면, 미들웨어의 역할은 주어진 운영체제의 기능상에서 사용자가 좀더 편리하게 실시간 시스템의 수행 행태를 분석하여 설계할 수 있는 패러다임과 도구를 제공하는 것이라 할 수 있다. 이를 기능별로 구분해 보면 다음과 같다.

2.2.1 다양한 형태의 실시간 스케줄링과 실시간 태스크 인터페이스의 제공

일반적으로 고정 우선순위 정책의 실시간 태스크만을 제공하는 범용 운영체제의 상위에서는 실시간 미들웨어가 EDF(Earliest Deadline First), LLF (Least Laxity First) 등의 좀더 다양한 스케줄링 기법들을 "태스크 감시자"라는 형태를 통해 제공할 수 있다. 커널에 내장되는 형태보다 정밀성은 감소하지만 근래 운영체제에서 제공되는 선점형 high resolution timer와 고정 우선 순위 실시간 태스크의 우선 순위 제어를 통해 제공할 수 있다. 즉 사용자는 미들웨어 도구를 이용하여 주기적 태스크, 산발적 태스크 및 그들의 시간 조건을 명시할 수 있고 미들웨어는 이들을 시간의 흐름에 따라 운영체제의 기본 우선 순위 기능을 이용하여 간접적으로 스케줄링해 주는 역할을 할 수 있다.

그러나 실시간 운영체제나 실시간 미들웨어 상의 시스템 설계자가 오해하지 말아야 할 부분은, 부적합한 시간 조건에 대해서도 운영체제나 미들웨어에 의한 자동적인 실시간 수행이 가능한 것이 아니라는 점이다. 즉 운영체제나 미들웨어 도구의 책임은 프로그래머로 하여금 시간적 분배나 실시간 스케줄링을 설계할 편리한 기본적 도구를 제공하는데 그치는 것이고, 그 나머지 책임은 설계자에 있다는 점이다. 예를 들면, "이벤트 발생 후 n 초의 데드라인을 가지고 이를 처리하는 태스크의 경우 이를 정의하고 시간 조건대로 수행시키는 것은 운영체제나 미들웨어의 책임이지만, 설계자는 n초 보다 짧은 구간에 동일한 이벤트를 발생시키지 않도록 시스템을 설계 하여야한다"는 것이다. 즉 이때 n초의 데드라인은 처리의 우선 순위 결정의 의미와 함께, 최대한의 처리 지연에 대한 선언적 의미도 함께 갖는다는 점을 인식해야 한다.

위의 스케줄링과 함께 미들웨어는 실시간 시스템 프로그래머가 주기적 태스크, 산발적 태스크들과 그의 시간적 수행 형태를 쉽게 정의할 수 있는 인터페이스를 제공해야 하겠다. 특히 실시간 태스크의 구현에 있어 범용 운영체제 상의 다중 프로세스 및 쓰레드 모델을 모두 사용할 수 있도록 제공되어야 한다.

2.2.2 객체 지향 기법과 실시간 컴퓨팅의 결합

구성 컴포넌트와 그 네트워크의 변경에 의한 모델의 수정이 자주 이루어지는 실시간 제어 응용 분야를 고려할 때, 유지 보수의 효율을 위해 객체 지향 설계와 실시간 컴퓨팅의 결합이 필요하다. 이를 다른 말로 기술하면, 주기적 또는 산발적 실시간 쓰레드의 객체 멤버화라 할 수 있다. 즉 자율적으로 시간 조건에 의해 동작하는 멤버 쓰레드를 가진 새로운 개념의 객체의 정의로 병행 실시간 컴퓨팅에 객체 지향 기법을 좀 더 밀착시킬 수 있다는 점이다. 이렇게 동적인 멤버 쓰레드를 가지는 객체 모델을 일반적으로 "실시간 객체 모델"이라 칭하게 되었고 90년대 초반부터 국내외에서 모델 및 그 수행 플랫폼에 대한 연구가 활발히 진행되고 있다.[1,2,4,5,7] 이중 대표적인 모델로는 TMO(Time-triggered Message-Triggered Object) 모델이 있는데 이에 관하여는 3장에서 기술하기로 한다.

2.2.3 분산 또는 병렬 컴퓨팅 환경의 제공

실시간 응용의 특성상, 분산 제어 또는 컴퓨팅이

다중-프로세서 환경보다 매우 효율적인 경우가 있다. 즉, 작업 분산화에 의한 로드 분산 및 신뢰도의 향상, 통신이 빈번하지 않은 독립적 모듈의 병렬 컴퓨팅에 의한 빠른 제어, 등을 고려할 경우, 분산 컴퓨팅이 그 해로서 제시되는 경우가 많다. 이러한 환경의 제공을 위해서 분산 미들웨어는 네트워크에 투명한 분산 IPC를 제공하여야 한다. 이러한 분산 IPC에 대한 기능적 요구 사항은 다음과 같다.

첫째, 시스템의 노드 재구성에 대해 소프트웨어의 수정이 거의 필요 없는 환경을 제공하여야 한다. 둘째, 분산 IPC를 위한 통신 채널에 대해 유니캐스팅/멀티캐스팅/그룹캐스팅/브로드캐스팅 등의 다양한 바인딩을 제공하는 것이 통신 효율성을 위해 좋다. 단순한 채널 방식의 분산 IPC의 차원을 넘어서 "분산 공유 메모리"나 "분산 공유 객체"가 제공될 수 도 있다. 단 이러한 도구의 사용은 사용자의 편의성에 중점을 둔 것인 만큼, 통신 오버헤드가 증가하는 것을 막을 수는 없다.

셋째는 실시간 보장성 통신의 제공이다. 분산 컴퓨팅에 의해 실시간 태스크들의 프로세서 자원 점유율 보장이라는 측면은 해결되지만 분산 노드, 태스크, 객체간의 분산 통신 지연에 의한 불확실성 문제가 대두된다. 이 문제를 위한 접근 방법은, 플랫폼 차원에서 통신 자원의 실시간 관리를 제공하는 실시간 통신 기법을 제공하는 것과, 실질적으로 이러한 실시간 통신 기법이 확보되기 어려울 경우 응용 차원에서 소프트웨어 단위 모듈의 구성과 노드 배치를 분산 IPC가 최소화 되도록 하는 것이다.

분산 컴퓨팅을 위한 미들웨어는 CORBA, RT-CORBA[3]와 같은 대표적인 것이 있지만, 이들의 목적은 객체 기반 범용 분산 서비스의 제공, 이질적(heterogeneous) 환경의 합동작업 등을 지향하고 있는 경우이고, 위에 언급한 바와 같이, 범용 운영체제가 제공하지 않는 다양한 실시간 스케줄링, 주기적/산발적 실시간 태스크와 객체의 결합을 위한 새로운 패러다임 등을 제공하는 실시간 시스템을 위한 미들웨어라고 볼 수는 없다고 하겠다. 즉 이러한 측면에서 볼 때, 표준성이 있는 분산 시스템용 미들웨어는 위에 언급한 실시간 시스템을 위한 요구 사항을 반영함으로써 본격적 실시간 미들웨어로 진일보 될 수 있다고 생각된다.

2.2.4 동기화 도구

실시간 시스템은 실시간 프로세스/쓰레드의 형태나 객체내의 멤버 쓰레드의 형태로 병행 프로그램으로 구성되는 것이 보통이므로 이들 사이의 잘 정의된 동기화 도구는 필수적이다. 실시간 프로세스 및 쓰레드 관련 인터페이스가 미들웨어에서 제공되는 경우, 이들을 위한 동기화 도구도 그 세만틱에 맞는 것으로 미들웨어에서 제공되는 것이 바람직하다. 대표적인 동기화 도구로 상호배제(Mutual Exclusion)와 Block/Wakeup 도구등이 제공되어야 하는데 이들은 프로세스/쓰레드에 대한 것이거나, 객체내의 멤버 쓰레드를 대상으로 하여야 한다. 이 때 이러한 도구들은 실시간 시스템에 있어서 대기로 인한 필연적인 실행시간의 불확실성을 가져오게 되는데 시스템 설계자는 최악의 경우를 분석하여 설계에 반영하여야 하고, 동기화 도구 자체는 우선순위 역전 최소화를 위한 프로토콜을 제공해야한다.

2.2.5 기타 도구들

그 외 미들웨어로서 제공될 수 있는 도구들은 다음과 같다.

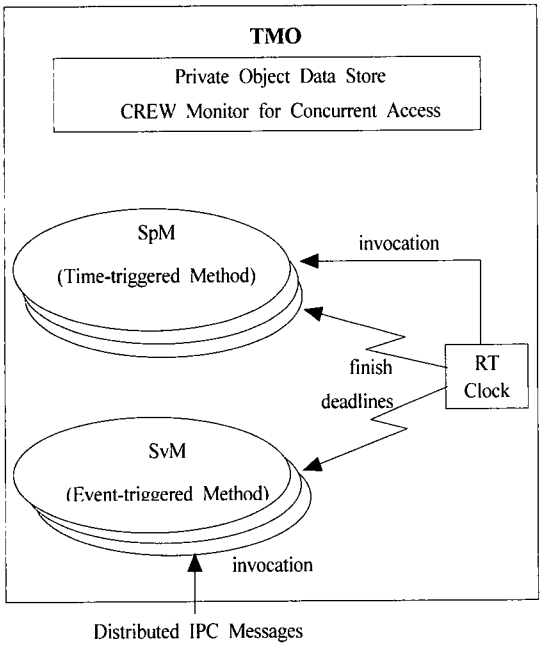
- 소프트웨어 공학적 도구
- UML과 같은 표준 형식의 설계 문서에 병행성, 시간 조건 등을 명세할 수 있는 도구
- 시간 조건의 시뮬레이션에 의한 검증 시스템 및 실행 감시(Monitoring) 도구

실시간 시스템 개발자의 역할
- 기능 구조 분석을 통한 실시간 컴포넌트의 식별 (객체, 메소드, 프로세스.쓰레드 등)
- 상호 수행 관계를 고려한 시간적 특성 분석

실시간 미들웨어의 기능
- 시간적 특성을 추상화하여 정의 할 수 있는 메소드의 형태 제공
- 운영체제와 개발자의 세만틱 갭 해소
- 분산 환경에 대한 추상화된 IPC 도구
- 유지보수와 리모델링에 강한 객체 개념 적용
- 수행 시간 분석 도구
- 감시 도구, 결함 허용 도구의 제공

실시간 운영체제의 기능
- 다양한 형태의 실시간 스케줄링 제공
- 실시간 자원관리
- 실시간 IPC 및 통신

(그림 1) 실시간 시스템 계층별 역할



(그림 2) TMO의 구조

3. 분산 실시간 객체 TMO와 그 미들웨어 엔진

3.1 분산 실시간 객체 모델 TMO

범용 운영체제가 제공하는 일반적 프로세스 및 쓰레드 모델로는 주기적/산발적 실시간 쓰레드의 설정, 객체 지향 기법 및 분산 컴퓨팅 환경으로의 확장 등을 수용하기 어렵다. 이 에 따라 실시간 시스템의 설계에 적합한 RTC++나 RTO.k[1] 등의 새로운 객체 모델들이 90년대 초반에 제시되기 시작하였으며, 현재에 이르러서는 이러한 객체 모델을 분산 환경에서 실행해주는 미들웨어들이 개발되어 산업화 및 상용화의 단계에 있다. 이중 대표적인 모델로 TMO(Time-triggered Message-triggered Object : RTO.k의 개칭) 모델과 그 분산 미들웨어 플랫폼에 대해 소개하기로 한다. TMO의 특징을 요약하면 다음과 같다.

- TMO는 두 가지 형태의 멤버 쓰레드를 갖는 동적 객체이다. 쓰레드가 객체의 멤버라 함은 일반 수동적 멤버 함수와 같은 scope 규칙의 적용을 받는다는 의미이다.
- 첫 번째 형태의 멤버 쓰레드 군은 주기적 멤버 쓰레드로 주어진 주기에 의해 구동되며 매 구동 시마다 주어진 데드라인에 의해 스케줄 된다. 이를 SpM(Spontaneous Method)이라 한다.
- 두 번째 형태의 멤버 쓰레드 군은 외부 TMO로부터 전달되는 이벤트 또는 서비스 요구 메시지의 수신에 의해 구동되는 서버 형태의 쓰레드로 서비스 완료 시까지 주어진 데드라인에 의해 스케줄 된다. 이를 SvM(Service Method)이라 한다.
- 이벤트 메시지들은 분산 IPC 도구를 통해 전달되며, 따라서 클라이언트 TMO는 로컬 또는 원격(remote) TMO일 수 있다. 즉 분산 컴퓨팅 환경으로의 확장 시 모델의 수정이 필요 없다.
- 객체 내의 자료 저장소인 ODS(Object Data

Store)는 SpM과 SvM들에 의해 병행으로 접근되며 병행성 제어를 위한 동기화 도구로 CREW(Concurrent Read Exclusive Write) 모니터가 사용된다.

- TMO는 일반적 객체 지향 기법의 요구하는 상속 및 다형성을 제공한다.

TMO의 구조는 (그림 2)와 같다. TMO 모델은 기반 운영체제나 통신의 실시간 성이 보장되는 시스템 상에서는 경성 실시간 시스템을 설계할 수 있는 개념적 근거를 제공하고, 일반 운영체제상에서는 연성 실시간 시스템의 구축에 적합한 모델이라 할 수 있다. 즉 경성 실시간 시스템을 설계할 경우, SpM의 수행은 사전에 결정되는 것이며 산발적 이벤트를 데드라인 이내에 처리하는 SvM들의 수행도 설계 시에 그 최악의 경우가 파악되어 상호 간의 수행에 지장을 초래하여서는 안 된다.

3.2 TMO의 수행을 위한 분산 미들웨어 엔진

현재까지 TMO를 위한 개발된 미들웨어 엔진은 다음과 같다.

- DREAM kernel: UC Irvine의 DREAM Lab.에서 '95년 개발된 최초의 Stand-alone TMO 엔진[6]으로 실시간 스케줄링과 분산 IPC만을 제공하고 미들웨어가 아니므로 일반적 운영체제의 API나 GUI를 제공하지는 못한다. 산업용 제어 시스템의 구현에 적합하다.
- WTMOS(Windows TMO System)[4], TMOSM (TMO Supprot Middleware): '98년 외대 RTDCS Lab.과 '99년 UCI DREAM Lab.에서 각각 개발된 윈도우즈 계열 운영체제 상의 TMO 수행을 위한 분산 미들웨어이다. TMO의 분산 수행과 함께 기존 Windows 운영체제의 API나 GUI를 그대로 사용할 수 있는 연성 실시간 시스템을 위한 플랫폼이다.
- LTMOS(Linux TMO System)[5]: '00년 외대 RTDCS Lab.에서 개발된 리눅스 계열 운영체제 상의 TMO 수행을 위한 미들웨어이다.

다음은 WTMOS와 LTMOS의 기능, 특징들의 요약이다.

- C++ 기반의 TMO 프로그래밍
- 1/1000초 단위의 클럭 해상도 및 분산 클럭 동기화
- 멀티 쓰레드에 의한 SpM 및 SvM의 구현
- 분산 IPC 및 윈도우 이벤트에 의한 SvM의 구동 및 완료 데드라인 스케줄링
- 시간 조건에 의한 SpM의 구동 및 완료 데드라인 스케줄링
- 사용자 정의의 Deadline Exception Handler
- 분산 공유 객체 (Distributed Shared Object)의 제공
- Linux와 Windows 플랫폼을 함께 사용하는 heterogeneous 분산 환경의 제공

위와 같은 특징의 TMO 미들웨어는 현재로서는 연성 실시간 시스템용이라 할 수 있으며, 시간 보장과 예측성이 강조되는 경성 시스템으로의 진화는 운영체제 API의 수행시간 보장과 실시간 통신 등을 전제로 하여야 하며, 이는 아직까지는 리소스와 컴퓨팅 메소드가 극히 제한되는 특수하고 작은 시스템에서 만 가능하다고 하겠다.

3.3 실시간 미들웨어의 응용 분야

실시간 미들웨어의 응용 분야는 다음과 같이 열거할 수 있다.

- 분산 실시간 제어 및 감시[9](공장 자동화, 무기 제어, Plant 제어, Intelligent Building, Home Automation & Security 등)
- 분산 실시간 시뮬레이션[5,10] (원전, 무기 체계, 워 게임 시뮬레이션 등): 실시간 시뮬레이션이라 함은 각각의 구성 요소가 실제 상황과 같은 시간적 수행 행태(behaviour)를 보이는 시뮬레이션 시스템을 말한다. 이러한 시스템은 주로 교육과 시스템 사전 분석의 목적으로 사

용되며 여러 복합적인 기술이 필요하므로 주로 고가의 외국 시스템에 의존하고 있는 실정이다.

- 멀티미디어 서비스[7]: 연성이지만 시간 조건에 의한 멀티미디어 스트림의 처리 및 동기화 가 필요한 멀티미디어 서비스 분야에 응용이 적합하다.

위에 열거한 바와 같이 실시간 미들웨어의 응용 분야는 넓고 막대한 시장 규모를 가지고 있다. 또한 위와 같은 응용 분야에 다른 형태의 실시간 미들웨어라고 할 수 있는 실시간 MMDB(Main Memory Database)의 접목으로 그 응용 분야를 더욱 넓힐 수 있다. 그러나 설계 및 유지보수의 편이성을 고려한 새로운 패러다임에의 적응보다는 기존 운영체제의 기능만을 활용한 소규모 인원에 의한 빠른 개발에 초점을 두는 국내 산업계의 성향은 실시간 미들웨어의 보편화를 어렵게 하는 한 가지 요인이 되고 있다.

4. 결 론

실시간 미들웨어는 운영체제에는 없는 실시간 기능을 제공하는 것이 아니다. 실시간 시스템의 개발은 운영체제의 기본적 실시간 기능의 바탕 위에서, 시스템 설계자의 기능 구조 및 시간 구조의 분석을 통한 설계 단계에서의 실시간화를 통해 이루어지는 것이고, 이때 미들웨어의 역할은 운영체제와 설계자의 세만틱 갭을 최대한 줄이는 것이다. 즉 실시간 미들웨어는 추상화된 실시간 특성 정의 도구의 제공과 함께 다음과 같은 소프트웨어 공학적인 측면의 장점을 함께 제공하여야 한다.

- 시간적 특성을 추상화하여 정의 할 수 있는 메소드의 형태 제공
- 분산 환경에 대한 추상화된 IPC 도구
- 유지보수와 리모델링에 강한 객체 개념의 결합
- 수행 시간 분석 도구, 감시 도구 및 결함 허용 도구의 제공

반면에 플랫폼 운영체제는 실시간성 향상을 실시간 스케줄링 및 자원관리 및 실시간 통신 과 IPC를 제공하여야 한다. 무기 체계 분야에서 앞서가는 일부 국가를 제외하면 세계적으로 실시간 시스템에 관한 연구의 결실은 그 응용 분야의 크기와 수요에 비해 부족한 것이 사실이다. 따라서 위에 열거한 실시간 운영체제와 미들웨어의 역할 및 기능에 대해 앞으로도 국내외적으로 많은 연구가 있어야 할 것이고, 좋은 연구의 결과가 집대성되어 특히 선진국의 전유물처럼 되어있는 고신뢰 실시간 분야에 국내 기술이 자리를 차지하기 바란다.

참고문헌

[1] K.H. Kim and H. Kopetz, "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", Proc. 18th IEEE Computer Software and Applications Conference, pp.392-402, November 1994.

[2] K.H. Kim, "Real-Time Simulation Techniques Based on the RTO.k Object Modeling", Proc. 20th IEEE Computer Software & Application Conference, August 1996.

[3] "An Overview of the Real-Time CORBA Specification", Tech. Report, Konkuk Univ. 2001. 8.

[4] J.G. Kim, M.H. Kim, B.J. Min, and D.B. Im, "A Soft Real-Time TMO Platform - WTMOS - and Implementation Techniques", Proc. 1st IEEE International Symposium on Object-oriented Real-time Distributed Computing, pp.256-264, April 1997.

[5] J.G. Kim and S.Y. Cho, "LTMOS: An Execution Engine for TMO-Based Real-time Distributed Objects", pp. 2713-2718, Proc. Of the International Conference on Parallel and Distributed

Processing Techniques and Applications, Vol 5, 2000. 6.22-29, Las Vegas.

[6] M. H. Kim and J. G. Kim, "An Environment for Real-Time Simulation Based on the TMO Model", Proc. Int. Conference on PDPTA, Jun. 1999.

[7] Kane Kim, et. al, "A Timeliness-Guaranteed Kernel Model : DREAM Kernel and Implementation Techniques", RTCSA, 1995, 10.

[8] Jung-Guk Kim, J.P. Hong, Byoung-Joon Min, Moonhae Kim, "Modeling of Multimedia Services using the TMO Model", Journal of Computer Systems Science and Engineering, 1998, 5.

[9] H.Y. Lee, B.J. Min, M.H. Kim, et. al, "A Monitoring Mechanism for the System-Level Test of Telecommunications Distributed Applications", Trans. of the Korea Information Processing Society, 1996. 3.

[10] Moon H. Kim, Jung G. Kim and Kane Kim, "Time-triggered Message-triggered Object Modeiling of a Distributed Real-time Control Application for its Real-time simulation", pp. 549-556, vol. 24, Proc. COMPSAC 2000, 2000. 10.25-27, Taiwan.

저자약력

김 정 국

1977년 서울대학교 계산통계학과 (이학사)
1979년 한국과학기술원 전산학과 (이학석사)
1986년 한국과학기술원 전산학과 (공학박사)
1983년-현재 한국외국어대학교 컴퓨터공학과 교수
관심분야 : 실시간 객체, 분산 컴퓨팅, 실시간 운영체제, 내장형 시스템
e-mail : jgkim@maincc.hufs.ac.kr