특 집

내장형 미들웨어 연구 동향

이 승 룡*

● 목 차 ●

1. 서 론

포스트 PC, 인터넷 정보가전, 이동 단말시스템과 같은 내장형 시스템이 차세대 정보통신 산업의 주력으로 부상하게 됨에 따라 내장형 시스템의 기술 확보가 중요한 이슈로 대두되고 있다. 내장형 시스템은 대규모 시스템의 일부로서 내장된 컴퓨터로, 일반적으로 센서나 액츄에이터(actuators)를 통해서 실세계와 직접 상호작용(interact)을 한다. 내장형 시스템을 개발하기 위해서는 도메인 지식, 실시간 운영체제 및 미들웨어, 최적화, 계산이론, 병행성, 컴파일러, 잘 정의된 네트워크 프로토콜, 소프트웨어 설계기법 등을 동시에 고려해야만 하는데, 지금까지 내장형 시스템 개발은 주로 도메인 지식을 가진 전문가가 내장형 소프트웨어를 개발하는 형태로 진행되어 왔다.

내장형 시스템은 도처에 산재한(ubiquitous) 컴퓨팅 플랫폼에 적용되며 자원이 제약이 심한 환경에서 안전성 신뢰성의 요구사항을 만족해야 되고 이동성, 실시간성, 융통성, 상호 운용성, 이식성 등을 지원해야 하며, 멀티미디어 서비스 제공을 요구받는다[1]. 그러나, 현재의 내장형 운영체제들은 인터넷 연결이나 플랫폼 독립적인 프로그램 다운로드 등과 같은 기능을 충분히 지원하기 어렵기 때문에, 산재한 환경에서 다양한 내장형 시스템들간의 상호 운용성, 플랫폼 독립성과 이식성을 지원할 수 있는 환경을 제공하는 내장형 미들웨어의 중요성이 부각되고 있다.

내장형 시스템의 특성상 개발자들은 충분히 필드에서 검증된 환경만을 사용하려는 보수적인 성향을 가지고 있다. 하지만 최근, 다양한 하드웨어 및 소프트웨어의 요구사항이 급변하고 있으며, 내장형 미들웨어 개발 환경도 이러한 요구사항을 잘 수용할 수 있어야 된다. 따라서, 플랫폼의 재사용성, 융통성, 재구성의 용이함을 제공하고, 개발자에게는 기존의 환경과 친숙한 개발 환경을 제공할 필요가 증대되고 있다. 더우기, 이동 내장형 시스템의 서비스가 멀티미디어로 확대됨에 따라, 이를 효과적으로 지원할 수 있는 하드웨어도 재구성(reconfigurable)이 가능하고 재 프로그램 가능한(reprogrammable) 프로세서 아키텍처로 전이되고 있다. 이러한 변화와 아울러 소프트웨어도 운영체제, 미들웨어, 응용 수준에 이르는 모든 영역에서, 재구성, 재사용, 적응성을 효과적으로 지원할 수 있는

* 경희대학교 전자정보학부 컴퓨터공학전공 교수

컴포넌트기반 소프트웨어 개발 방법론[2,3]이 주목받고 있다.

본 고에서는 향후 내장형 시스템의 미들웨어로서 중요한 역할을 담당할 자바가상머신(Java Virtual Machine: JVM), 컴포넌트에 기반한 미들웨어, 그리고 실시간 컴포넌트 설계 방법론에 대한 연구들을 소개함으로서 향후 내장형 시스템 미들웨어 개발 방향과 특징에 대하여 살펴보기로 한다.

본 고의 구성은 다음과 같다. 2장에서는 자바가상머신에 대한 연구를 실시간 지원기능, 경량화, 성능향상 관점에서 연구동향을 살펴보고, 3장에서는 경량 미들웨어 및 컴포넌트기반 미들웨어에 대하여 소개한다. 4장에서 컴포넌트기반 실시간 시스템 개발 툴셋에 대하여 언급하고, 5장에서는 실시간 컴포넌트 설계방법에 관한 연구를 소개한 다음, 마지막으로 6장에서 결론을 내린다.

2. 자바 가상머신

동적인 어플리케이션 다운로드, 크로스 플랫폼 호환성, 빠른 응답성, 비연결성, 보안 기능 제공 등의 이유로 내장형 시스템 개발자들은 자바를 사용하고 있다. 그러나 실시간 또는 자원의 제약이 심한 내장형 시스템에 응용하기 위해서 자바를 수정하는 연구가 활발히 진행중이며, 또한 자바가 가지는 장점을 유지하면서 어떻게 성능을 향상시킬 것인가에 대한 연구도 진행 중에 있다.

2.1 자바 내장형 실시간 시스템

실시간 내장형 시스템 구축에 자바가 사용되는 이유는 동적 클래스 로딩, 쓰레기 수집기, 멀티쓰레딩 등의 기능을 제공하기 때문이다. 반면에, 느린 JIT(Just-In-Time) 컴파일러의 런타임 번역이나 바이트 코드 해석, 쓰레드의 행위가 JVM이나 운영체제에 의존해 있기 때문에 실시간 태스크를 다루는데 적절하지 못하고, 현재 자바가상머신 쓰레기 수집기가 실시간 시스템을 지원하기가 어렵다는 이유로 자바가 실시간 내장형 시스템에 바로 적용하기가 적합하지 못하다. 이러한 문제를 해결하기 위하여 미국 National Institute of Standards and Technology(NIST)의 관리를 받고 있는 자바 플랫폼의 실시간 확장에 관한 요구사항(Requirements for Real-time Extensions for the Java Platform) 그룹(http://www.nist.gov/rt-java)은 표준 실시간 자바 확장에 대한 요구사항(Real-Time Java: RTJ)을 정의하고 있으며, 다음과 같은 기초 요구사항 문서를 출간하였다.

첫째, 자바프로그래밍 환경에서 디바이스 드라이브를 구현하거나, 물리적 메모리에 직접 접근할 필요가 있거나 인터럽트를 처리하기를 원할 때 사용자가 하드웨어에 직접 접근할 수 있는 기능을 추가한다. 둘째, Rate Monotonic(RM) 스케줄링 또는 Earliest Deadline First (EDF) 알고리즘과 같은 선점형 우선순위 기반 스케줄링 정책을 지원해야 한다. 셋째, 우선순위 역전현상과 같은 동기화 문제를 해결하기 위하여 우선순위계승(priority inheritance) 또는 우선순위 상한(priority ceiling) 프로토콜을 지원해야하며, 큐잉 정책을 보장해줄 수 없는 자바의 모니터 기능을 개선해야 한다. 넷째, 비동기 이벤트 처리 기능을 지원해야하며 다섯째, 자원 어카운팅(accounting), 자원 회수(reclamation), 자원 할당, 자원 협상과 같은 자원관리 기능을 제공해야하고, 마지막으로, 실시간 시스템에 적합한 예를 들면 점진적(incremental) 쓰레기 수집기를 지원해야 한다.

이는 API 기반의 확장을 지원하고 있으며, NIST 문서를 기반으로 하고 있는 자바 내장형 실시간 시스템에 대한 접근방법은 크게 실시간 API v0.2 (RTJ-02)[4], Real-Time Specification for Java, v0.8.1 (RTJ-08)[5], Real-Time Core Extension for the Java Platform(RT Core)[6], 실시간 JavaO 사양(RT JavaO)[7]으로 구성된다. NIST에서 문서를 만들기 전에 소개되었던 해결책 중에 하나는 Real-Time Java

Threads(RTJT)[8]이며, 다른 하나는 Portable Executive for Reliable Control(PERC)[9]으로 이것은 실시간 시스템에 대한 추상을 제공하는 실시간 패키지와 하드웨어를 접근하기 위해 하위 레벨 추상을 제공하는 내장 패키지로 구성되어 있다. 그리고 CSP algebra, Occam2 언어, Transputer 마이크로프로세서를 기반으로 하는 Communication Threads for Java (CTJ)[10]가 있다. 다른 유형의 접근방법은 자바가 상머신을 운영체제에 통합하는 것으로 General Virtual Machine(GVM)[11]에 의해서 이루어졌다. 자바에 성능을 향상시키는 또 다른 방법으로 JVM을 마이크로프로세서에 통합하는 것으로 Rockwell Collins의 JEM-1 마이크로프로세서(picoJava-I)가 있다[12]. (그림 1)은 각 접근방법의 특징들을 비교한 것이다. 여기에서 "A"는 해당 이슈를 상세하게 설명하였음을 의미하며,"M"은 부분적으로, " - "는 전혀 설명하지 않았음을 의미한다.

2.2 실시간 자바 전문가 그룹(Real-Time Java Expert Group: RTJEG)

1998년 실시간 자바 플랫폼을 개발하고 있던 NewMonics 사와 NIST, IBM, Sun이 하나로 합쳐져 실시간 요구사항 워킹 그룹이 만들어졌다. 이 그룹이 "자바 플랫폼의 실시간 확장에 관한 요구사항" 라는 문서를 발간하는 과정에서 Sun은 1998년 말에 자바의 새로운 요구사항들을 검토하여 스펙을 만드는 기구인 Java Community Process(JCP)를 만들었으며, 워킹 그룹의 연구를 토대로 IBM이 작성하여 제출한 Java Specification Request(JSR)를 채택하였다(JSR-000001). 워킹 그룹의 리더인 Greg Bollella는 실시간 자바 스펙(Real-Time Specification for Java: RTSJ)을 정의하는 그룹과 컨설턴트 그룹으로 구성된 RTJEG[13]을 결성하였으며, 이 그룹의 주요 구성원은 IBM, Cyberonics, Aonix/Ada Core Technologies, Microware Systems Corporation, QNX System Software Lab, Sun Microsystems, Rockwell-Collins/aJile, Nortel Networks 등이다. 이 구룹에서는 실시간성을 지원하기 위해 고려되어야 할 것들을 다음과 같이 정의하였다.

첫째 쓰레드 스케줄링으로, 실시간 스케줄링을 위하여 스케줄링 객체는 힙 영역에 생성되어 쓰레기 수집기의 영향을 받는 RealtimeThread와 힙 영역의 데이터를 접근할 수는 없지만 쓰레기 수집기의 영향을 받지 않는 NoHeapRealtimeThread를 사용한

Feature	Hw Access	Task Scheduling		Synchronization Communication		Event Handling		Resource Management		Garbage Collection	
Solution		Priorities	Deadlines	Java monitors	Task Thread	Internal	External	Budgets	Negotiation	Preemptible	Bounded
RTJ-02	-	A	-	-	-	A	A	A	-	A	-
RTJ-08	A	A	A	-	A	A	A	A	-	A	A
RT Core	A	A	A	-	A	A	A	A	-	A	-
RT JavaO	M	A	-	M	M	M	M	-	-	A	-
PERC	A	A	M	-	-	A	M	A	A	A	M
CTJ	A	A	-	-	-	-	-	-	-	-	-
RTJT	-	A	A	-	-	-	-	M	-	M	-
GVM	M	A	-	-	-	-	-	-	-	-	-
picoJava-I	A	-	-	-	-	-	-	-	-	A	-

(그림 1) 자바 내장형 실시간 시스템에 대한 접근방법 비교

다. 스케줄링을 하기 위한 방법은 우선순위에 기반한 스케줄링 기법을 제공해야 하며, 이를 위해서는 기본적으로 최소한 28개의 유일한 우선순위를 갖는 고정된 우선순위 선점형 디스패쳐를 제공해야 한다.

둘째, 메모리 관리로써 실시간성을 지원하기 위하여 메모리는 쓰레기 수집기의 예측할 수 없는 지연 문제를 해결할 수 있는 메카니즘을 가져야 하며, 쓰레기 수집기의 영향을 받지 않기 위해 메모리 영역을 분리한다. 이러한 메모리 영역은 scoped 메모리로써 이는 범위(scope)에 의해 정의되는 라이프타임(lifetime)을 갖는 객체의 클래스를 처리하기 위한 메카니즘을 제공한다. 그리고, 물리적 메모리로써, 이는 메모리에 빠르게 접근하는 것과 같은 중요한 특징을 갖는 특정한 물리적 메모리에 객체가 생성되는 것을 허용한다. 마지막으로, 임모탈(immortal) 메모리는 한번 할당받아서 응용이 종료될 때까지 존재하는 객체를 가지고 있는 메모리 영역을 표현한다.

셋째, 동기화로써, 동기화를 위해서는 우선순위를 사용한다. 높은 우선순위를 갖는 쓰레드는 실행하기 위해 대기하고 있는 쓰레드 중에서 스케줄러가 선택한 쓰레드가 된다. 우선순위를 사용하기 위해서 대기 큐가 필요하며 이 큐는 FIFO방식으로 하여, 쓰레드의 우선순위가 같은 경우가 발생했을 경우 처리가 용이하도록 한다. 또한, 우선순위 계층 모니터 제어 정책을 구현하여야 한다.

넷째, 비동기 이벤트 핸들링으로, 비동기 이벤트 핸들링을 위해서는 AsyncEvent와 AsyncEventHandler가 필요하다. AsyncEvent는 어떤 이벤트가 발생할 경우 핸들러들을 unblocking하고, 핸들러들의 셋을 이벤트와 결합시킨다. AsyncEventHandler는 쓰레드와 비슷하게 작동을 하며, 이벤트 발생시 결합된 핸들러들을 스케줄 한다.

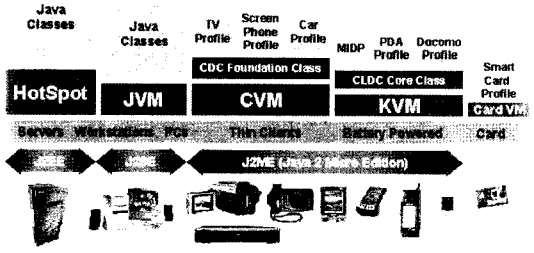
마지막으로, 물리적 메모리 접근으로 RTSJ는 프로그래머들이 직접 물리적 메모리에 접근할 수 있는 클래스를 정의하였다. RawMemoryAccess는 프로그래머가 물리적 주소의 범위와 byte, short, int, long, float, double로 물리적 메모리에 접근할 수 있는 객체를 생성할 수 있도록 정의하였다.

2.3 Sun 사의 J2ME 플랫폼 및 KVM

현재 자원이 제약된 소형의 이동단말기 시스템을 위한 자바플랫폼은 Sun 사에서 Java 2 Micro Edition(J2ME) Connected, Limited Device Configuration(CLDC)/MIDP[14, 15]와 CLDC/PDAP가 있다. CLDC/MIDP는 휴대전화기를 위한 것이고, CLDC/PDAP는 PDA를 위한 자바 플랫폼이다. 퍼스널 자바와 내장형 자바는 J2ME 플랫폼이 등장하기 이전부터 내장형 시장을 목표로 한 자바 플랫폼이었다. 인터넷 셋탑박스와 같은 나름대로의 성과도 있었지만, pJava와 eJava는 시장에서 주목받을 만한 결과를 출시하지 못했다. 그것은 기본적으로 Sun 사가 하드웨어 설계 및 제조와 관련해서는 내세울 만한 강점이 없었기 때문이었다. 즉, 하드웨어 업체들의 적극적인 지지 없이는 자바의 내장형 시장 진출은 어려운 과제였다. 그리고, 수많은 소비자/내장형 디바이스들에 공통적으로 적용할 수 있는 플랫폼을 개발한다는 것도 내부적인 한계가 있었다. 다양한 하드웨어와 운영체제, 전혀 호환성 없는 유저 인터페이스까지 이 모든 차이점을 극복하면서 기존의 J2SE 기반의 자바와의 호환성까지 유지할 수 있는 솔루션은 불가능하다는 현상을 인식하기 시작한 것이다. 그래서 등장한 개념이 컨피규레이션과 프로파일로의 플랫폼 분할이다. 컨피규레이션이란 자바가상머신과 코어 API들에 대한 명세를 의미하고, 프로파일은 그 상위의 클래스 라이브러리, 즉 표준 API 집합에 대한 명세를 의미한다. 이러한 개념적인 분할이 필요한 이유는 메모리와 CPU 등의 크기와 성능이라는 측면에서의 요구사항이 동일한 디바이스들의 집합을 하나로 묶어 컨피규레이션을 정의하고, 이러한 컨피규레이션을 바탕으로

각 디바이스들의 기능 혹은 버티컬 시장의 요구사항에 맞춰 프로파일을 정의함으로써 플랫폼의 통일성과 다양성을 동시에 만족시킬 수 있기 때문이다.

경량 내장형 자바가상머신 분야에서는 Sun 사가 2000년 J2ME을 개발하면서 이의 핵심 컴포넌트인 KVM을 선보였다(http://java.Sun.com/ products/cldc) [16]. KVM은 40에서 80 킬로바이트의 메모리 영역에서 구현되며 이식성, 모듈성을 지원한다. KVM은 128 킬로바이트 미만의 메모리를 갖는 16/32 bit RISC/CISC 마이크로프로세서에 적합하며, 셀룰러 폰, 호출기, PDA, POS 단말기 등에 적용된다. KVM은 가상머신, 최소한의 자바 클래스 라이브러리, 자바 어플리케이션이 동작할 수 있는 여분의 힙(heap) 영역을 갖는다. 또한 경량화를 위하여, 플로팅 포인트를 지원하지 않고, 클래스 인스턴스의 finalization을 지원하지 않으며, JNI를 지원하지 않는다. 그리고, 사용자 정의 클래스 로더가 없으며, 리플랙션(reflection) 기능이 없고, 쓰레드 그룹이나 데몬 쓰레드를 위한 지원 기능이 없고, 약 참조(weak references)가 없으며, 에러 처리기능을 제한함으로써 경량화를 지원한다. (그림 2)는 자바 2 플랫폼을 보여주며, 여기서 CDC는 Connected Device Configuration, J2EE는 Java 2 Enterprise Edition, J2SE는 Java 2 Standard Edition을 일컫는다.



(그림 2) 내장형 시스템을 위한 자바 2 플랫폼

2.4 Java 가상머신 성능

자바 가상머신의 성능 향상을 위해서는 잘 설계된 소프트웨어, 최적화 컴파일러, 아키텍처의 지원, 그리고 효율적인 런타임 라이브러리 등이 잘 조화를 이루어야 한다. 자바 가상머신의 성능 향상 연구에 대한 소개는[17,18]에서 잘 보여주는데 이는 크게 컴파일 개선(compilation improvement) 런 타임(runtime improvement) 개선, 하드웨어적인 접근 방법으로 자바프로세서로 구분된다. 런 타임 개선으로서는 컴파일 개선에는 JIT 컴파일러, 직접 컴파일러(direct compiler), 바이트코드 대 소스 변환기(bytecode-to-source translator)의 성능을 개선해야 된다. 런 타임 개선으로는 바이트코드 최적화, 병행 및 분산 수행, HotSpot와 같은 동적 컴파일(dynamic compilation)이 있다. 그리고 자바가상머신의 성능을 개선시키는 기법으로는 쓰레드 동기화, RMI (Remote Method Invocation), 고성능 수치계산, 클래스로더, 객체 할당기, 그리고 쓰레기 처리기의 성능을 개선 시켜야 한다.

3. 경량 및 컴포넌트기반 미들웨어

학술용 경량 미들웨어에 대한 연구로 North Carolina 주립대학의 Balay[19]는 경량 미들웨어 구조인 (Lightweight Software Bus: LSB)를 제안하였다. 이는 일반적인 미들웨어와는 달리 내장형 및 이동 환경을 고려한 경량 미들웨어는 간단한 API를 제공해야되고 제한된 자원에서 낮은 부하를 유지하도록 해야 된다는 것이다. LSB는 사용자에게 효율적이며 간단한 인터페이스를 제공하고, 모듈에 메시지와 GUI를 모두 고려하며, 외부 프로세스와의 통신을 응용에서 처리하지 않는 기능을 가지고 있다. LSB는 메시지 인터페이스, 소켓 인터페이스, 다이렉트 채널 인터페이스로 구성되어 있다. 메시지 인터페이스는 메시지 전송을 위한 메시지 타입과 핸들링을 위한 콜백 함수로서 멀티캐스트로 전

송한다. 소켓 인터페이스는 클라이언트의 각 프로세스간의 통신을 위한 함수 등록 등을 담당하며, 직접 채널 인터페이스는 하나의 클라이언트가 다른 클라이언트와 연결이 성립된 후에 통신을 담당하는 인터페이스를 제공한다.

한편, CORBA나 Java RMI는 기존의 표준 클라이언트/서버 응용에는 잘 적용할 수 있으나, 최근 멀티미디어, 실시간, 이동성과 같은 다양하고 새로운 요구사항을 동시에 만족시키기에는 적절치 못하다는 인식이 확산되면서 차세대 미들웨어에 대한 연구가 진행중이다. 특히 이들은 새로운 요구사항에 대한 적응과 기술적 진화에 대하여 원칙 없이 ad hoc한 방법으로 접근함으로써 응용 프로그램과 시스템을 복잡하게 만든다. 따라서, 미들웨어 아키텍쳐 설계 시 부터 융통성과 적응성을 염두에 둔 리플랙티브(reflective)하고 컴포넌트에 기반한 차세대 미들웨어에 대한 연구가 진행중이다. 차세대 미들웨어는 공개된 엔지니어링 미들웨어 플랫폼을 정의함으로서 이러한 요구사항을 만족시킬 수 있는데 이는 런타임 재구성, 하부 컴포넌트에 대하여 적응할 수 있고 그들을 검수 할 수 있어야 한다.

리플랙션은 프로그램에 대한 접근과 추론 그리고 변경을 용이하게 하여 준다. 이러한 특징은 미들웨어의 내부 행위(behavior)를 검사할 수 있고, 미들웨어의 구현을 모니터링하기 위해서 추가적인 행위를 삽입할 수 도 있다. 뿐만 아니라, 커뮤니케이션 스트리밍의 대역폭 요구사항을 감소시키기 위하여 필터 객체를 삽입함으로서 내부 미들웨어의 행위에 적응할 수 있게 된다. 컴포넌트는, 독립적인 배치(deployment)나 구성의 단위로서, 이는 소프트웨어의 재사용이나, 빠른 구성, 적응성, 수정과 분배를 효과적으로 수행 할 수 있다. 이러한 컴포넌트들은 IP나 IP 멀티캐스팅과 같은 하위수준의 통신 프로토콜, 필터나 디스페쳐, 버퍼와 같은 엔드 시스템 컴포넌트, 그리고 스케줄링 정책, QoS 관리 컴포넌트와 같은 관리 컴포넌트들을 포함한다. 한편 차세대 미들웨어에서 고려할 사항으로는 메타 공간(meta-space)으로서, 모든 컴포넌트들은 컴포넌

분야	기술 특성
자바 내장형 실시간시스템	Accessing Hardware, Scheduling (RMA, EDF), Synchronization, Asynchronous Event Handling, Resource Management, Dynamic Error Handling
실시간 자바 전문가그룹	No Garbage Collection, No Dynamic Initialization and Resolution, Stack Allocation, Partitioning Memory and CPU Time
경량자바 가상머신	-No Floating Point Support, No Support for Finalization of Class Instances, No Support for the JNI, No User-Defined Class Loader, No Reflection Features, No Support for the Thread Groups or Daemon Threads, No Weak References, Limitation on Error Handling
자바 가상머신 성능개선	- Compilation Improvement JIT(Just-In-Time), Direct Compiler, Bytecode-to-Source Translator, - Runtime Improvement Bytecode Optimization, Dynamic Compilation, Executing Java in Parallel Improve JVM Thread Synchronization, Remote Method Invocation, High Performance, Numeric Computation, Garbage Collection - Hardware Processor (Java Processor)
경량미들웨어	- 내장 및 이동 환경을 고려한 간단한 API 제공 - 제한된 자원에서 낮은 부하를 유지.
컴포넌트기반미들웨어	- 프로토콜 프레임워크 기반 적응형 멀티미디어 - 스트림 인터페이스, 명시적 바인딩, 제 삼자 바인딩, QoS 지원 - 예측 가능한 실시간 QoS 지원

(그림 3) 현재 개발중인 미들웨어의 특성

트들에 대한 하부 인프라스트럭쳐에 적응하고 검사를 지원할 수 있도록 메타 공간과 협력해야된다. 이와같은 메타 모델은 4가지로 나눌 수 있는데 첫째, 객체의 오퍼레이션이나 흐름과 같은 객체 인터페이스를 다루는 인캡슐레이션 메타모델 둘째, 바인딩과 같은 복합 객체의 컨피규레이션을 나타내는 구성 (composition) 메타모델 셋째, 인터페이스에서 상호 작동하는 프로세스를 구체화하는 환경 메타모델 그리고 마지막으로 쓰레드나, 메모리 등의 자원을 나타내는 자원 메타 모델로 구성된다. 또한 이러한 리플랙티브 미들웨어가 가져야할 추가적인 성질로는, 미들웨어, 운영체제, 네트워크등에 seamless한 접근이 가능해야 하고, 기존의 이러한 레가시(legacy) 시스템을 잘 활용해야되며, QoS 관리 기능을 지원해야하고, 이들 미들웨어로 인한 하부시스템 성능에 최소한의 영향을 주어야 하며, 개방성이나 확장성, 그리고 하부 미들웨어와의 통합이 용이해야 한다.

이같은 차세대 미들웨어에 대한 연구는 다음과 같다. 첫째, 프로토콜 프레임워크에 기반한 적응형 멀티미디어 ORB인 MULTE-ORB[20-22], 둘째, 컴포넌트기반 reflective 미들웨어인 Open-ORB[23,24], FlexiNet[25], 셋째, 스트림 인터페이스, 명시적 바인딩, 제 삼자 바인딩, QoS 지원등을 이용한 멀티미디어응용을 지원할 수 있는 분산 객체기반 미들웨어 플랫폼인 Generic Object Platform Infrastructure (GOPI)[26], 넷째, 예측 가능한 실시간 QoS와 고성능 기능을 특히 고려한 TAO[27,28] 등이 있다. (그림 3)은 현재 개발중인 자바가상머신 및 미들웨어의 특성을 보여주고 있다.

4. 컴포넌트 지원 툴셋

미국 버지니아대학의 Stankovic은 컴포넌트 기반 실시간 내장형 시스템들의 개발이나, 평가 및 구현을 용이하게 할 수 있는 내장 실시간 시스템을 위한 컴포넌트 기반 운영체제를 구성하고 분석할 수 있는 Virginia Embedded Systems Tools(VEST)[29] 툴셋을 개발 중이다. 그리고 이 툴셋은 ASOS embedded 시스템 개발을 위한 것으로 컴포넌트 조합과 분석구조를 갖고 있으며, 오프라인 분석과 온라인 적응성이 있다. 이는 코드 프레그먼트나, 함수 및 객체들로 이루어진 수동적 컴포넌트를 먼저 생성한 뒤 이를 런타임 구조에 매핑 시킨다. 이때 사용하는 컴포넌트는 계층적 구조로써 컴포넌트, 서브컴포넌트, 마이크로 컴포넌트로 구성되며 컴포넌트 라이브러리에 저장한다. 이 툴셋은 컴포넌트 라이브러리, 내장형 시스템 구성, 인프라스트럭쳐 생성, 수동적 소프트웨어 컴포넌트를 능동적 런타임 구조로 매핑하는 기능을 지닌 컨피규레이션 도구, 그리고 실시간과 신뢰도 분석을 위한 분석 도구들을 가지고 있다. 이때 컨피규레이션 도구에서는 의존성 검사를 위해 factual, inter-component, aspect, general의 4가지 의존성 검사를 지원한다. 이에 반하여, MetaH[30]는 제한된 의존도 검사와 기반 실시간 운영체제를 갖는다고 가정하고 능동적 컴포넌트로 시작을 하는 것이 VEST와는 다르다. 또한 컴포넌트 기술이 보다 강력한 기능을 제공하기 위해서는 아키텍쳐 기반으로 컴포넌트의 생성, 조립 및 추출이 가능해야 하며 현재 해외에서도 소프트웨어 아키텍쳐 기술을 컴포넌트 기술 분야에 적용하려는 연구가 진행 중이다[33].

5. 실시간 컴포넌트 설계

컴포넌트 기반 소프트웨어 엔지니어링 기법은 제품의 성능을 개선하고, 적시에 시장 출시 (time-to-market) 요구사항을 잘 만족할 수 있으며, 재사용이 가능하여 개발비용을 감소시킬 수 있고, 증대되는 소프트웨어의 복잡도를 감소시킬 수 있다는 이점이 많다. 이러한 점 때문에, 아직도 많은 제약과 해결해야될 난제가 있음에도 불구하고 이

를 실시간 내장형 시스템 개발에 적용하려는 연구가 진행중이다[2,31,32]. 일반적으로 실시간 시스템에서 컴포넌트의 단위는 태스크보다 작아서는 안되며, 컴포넌트가 시간 제약을 만족할 수 있도록 설계되어야 하고, 종단간 트랜잭션 마감시간을 고려해야만 된다. 또한 실시간 시스템은 많은 독립적인 오퍼레이션을 동시에 작동 시켜야 되기 때문에 병행 처리를 수행해야만 되고, 외부 환경과 끊임없이 많은 상호작용으로 인한 부하를 효과적으로 다루기 위하여 예외처리 기능도 지원해야 된다. 그리고 프로세싱이나 메모리 요구, 예기치 못한 시간의 성질을 효율적으로 다룰 수 있는 기능도 지원되어야 한다.

실시간 컴포넌트 관점에서 보았을 때, 컴포넌트는 바이너리 요소이고, 독립적으로 배치할 수 있는 단위이다. 그리고, 견고한(persistance) 상태를 가지고 있지 않으며 제 삼자 구성(third-party composition)의 단위이고, 또한 시간적 요구사항을 반드시 만족 시켜야만 한다. 시간적 제약을 컴포넌트에 도입하는 것은 쉬운일은 아니다. 예를 들어 컴포넌트의 시간적 행위는 타겟 아키텍처와 메모리 구조에 의존한다. 만일 다른 제삼자 기업으로부터 획득한 컴포넌트의 목표가 자신의 그것과 다를 수가 있기 때문에 자신이 목표로 하는 타겟의 시간적 행위와는 다를 수 있다. 경성 실시간 시스템에서 최악 수행시간(Worst-Case Execution Time: WCET)은 통상적으로 시간적 요구사항을 만족시킬 수 있는지 여부를 결정하는 스케줄 분석에 사용되지만, 그것만으로는 공급자로부터 얻은 시간 정보가 옳은지를 확인할 수는 없다. 이러한 점을 극복하기 위한 가능한 해결 방법중의 하나로 바이너리 컴포넌트를 사용하기도 한다. 한편, 실시간 컴포넌트간의 정보를 교환할 때 인터페이스는 메시지 큐인 buffered와 공유메모리인 unbuffered 통신 방법을 사용한다. 통상적으로 unbuffered 통신을 사용하면 시스템의 시간적 요구사항을 손쉽게 점검할 수 있어 실시간 컴포넌트의 인터페이스는 주로 이 방법을 사용하며 다른 용어로는 포트(ports)라고도 한다.

실시간 컴포넌트를 개발하는 과정은 몇가지 단계로 나눌 수 있다. 개발은 시스템 명세단계로부터 시작되는데 이러한 시스템 명세는 최상위 수준 설계단계에 입력된다. 최상위 수준 단계에서는 시스템을 모듈별로 디컴포지션(decomposition)하고, 설계자는 컴포넌트 라이브러리를 브라우징 한 뒤 가능한 컴포넌트를 대상으로 의도하려는 시스템을 설계한다. 구체적 설계 단계에서는 어떤 컴포넌트가 적절한지를 검사하는데 이때, 실시간 또는 비실시간 컴포넌트들도 고려한다. 그런 다음 스케줄링 및 인터페이스 체크 단계에서는 선택된 컴포넌트가 시스템에 적합한지를 보여주며 이때 새로운 컴포넌트가 필요한지, 적절한 컴포넌트가 사용되었는지를 확인한다. 이러한 컴포넌트의 선택과 스케줄링은 적절한 컴포넌트를 얻을 때까지, 그리고 설계된 시스템이 정제될 때까지 반복되어 수행된다. 새로운 컴포넌트가 생성되면 이는 컴포넌트 라이브러리에 저장된다. 최종 시스템이 최초의 명세를 만족하면 컴포넌트의 시간적 행위가 설계과정에서 정의된 시간 제약조건을 만족시키는지 여부를 검증하기 위하여 타겟 플랫폼에서 검사를 수행한다.

6. 결 론

본 고에서는 이동 내장형 실시간 시스템의 미들웨어인 자바가상머신, 경량 미들웨어, 컴포넌트 기반 미들웨어, 컴포넌트기반 개발 툴셋, 그리고 실시간 컴포넌트 설계 방법론에 대한 연구들을 소개함으로서 향후 내장형 미들웨어의 개발 현황과 특징을 살펴보았다.

한편, 이동 내장형 실시간 미들웨어를 개발하기 위해서 사용자들은 경량화, 실시간 서비스, 고성능 지원, 개발의 용이함이라는 서로 상반된 특성을 동시에 요구하고 있지만, 현실은 실시간, 경량화, 성

능향상, 실시간 내장형 시스템 컴포넌트 설계방법론 등이 서로 독립적으로 연구가 진행되고 있다. 따라서, 현재의 이동 내장형 실시간 시스템 환경에서 각기 다른 단일목표 지향적으로 진행되고 있는 현재의 연구 방법과는 달리, 경량화, 시간제약, 성능이라는 특징을 통합 프레임(unified framework)내에서 사용자의 요구에 따라 이해득실을 고려하여, 최적화된 컴포넌트기반 미들웨어를 효과적으로 도출해 낼 수 있는 방법이 새로운 연구 주제로 대두되고 있다.

참고문헌

[1] Edward A. Lee, "What's Ahead for Embedded Software?", IEEE Computer, September 2000, pp. 18-26.

[2] Wolfgang Fleisch. Applying Use Cases for the Requirements Validation of Component-Based Real-Time Software. In Proceedings of 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'99), IEEE Computer Society Press, 1999.

[3] Clemens Szyperski. Component Software - Beyond Object-Oriented Programming, Addison-Wesley, 1997.

[4] Sun's Java Community Process Real-Time Expert Group, "Proposed Java Real-Time API, v0.2," Technical report, Java Software Division of Sun Microsystems, December 1998. http://www. sdct. itl.nist.gov/carnahan/real-time/Sun/api.

[5] The Real-Time for Java Expert Group, "Real-Time Specification for Java, v0.8.1," Technical report, RTJEG, September 1999. http:// www.rtj.org.

[6] J Consortium, "Draft International J Consortium Specification," Technical Report, J Consortium, September 1999. http://www.j- consortium.org/.

[7] K. H. Krause and W. Hartmann, "RT Java Proposal," Technical Report, A&D GT 1, 1999. http://www.j-consortium.org/rtjw.

[8] A. Miyoshi, H. Tokuda, and T. Kitayama, "Implementation and Evaluation of Real-Time Java Threads,"" In Real-Time Systems Symposium. IEEE Computer Society, December 1997. page 166-174.

[9] K. Nilsen, ""Adding Real-Time Capabilities to Java,"" Communications of the ACM, 41(6), June 1998. page 49-56.

[10] G. Hilderink, "A new Java thread model for concurrent programming of real-time systems," Real-Time Magazine, January 1998, pp. 30-35.

[11] G. Back, P. Tullmann, L. Stoller, W. Hsieh, and J. Lepreau. Java Operating Systems: Design an implementation. Technical report, Department of Computer Science, University of Utah, August 1998, www.cs.utah.edu/projects/flux.

[12] H. McGhan and M. O'Connor, picoJava: a direct execution engine for Java bytecode, IEEE Computer, 31(2), October, 1998.

[13] G. Bollella, B. Brosgol, P. Dibble, et al., "The Real-Time Specification for Java", The Real-Time for Java Expert Group, December 1999.

[14] Java Community Process, "J2ME Connected, Limited Device Configuration," JSR-000030, May, 2000.

[15] Java Community Process, "Java 2 Platform Micro Edition (J2ME) Technology for Creating Mobile Devices," A White Paper, May, 2000.

[16] Java Community Process, "The K Virtual Machine," A White Paper, June 1999.

[17] I. H. Kazi, H.H. Chen, B. Stanley, and D. Lilja, "Techniques for Obtaining High Performance in Java Programs," ACM Computing Surveys Vol.

32, No.3, September 2000, pp. 213-240.

[18] R. Radhakrishnan, et. al, "Java Runtime Systems: Characterization and Architectural Implications," IEEE Transactions on Computers, Vol. 50, No. 2, Feb. 2001, pp. 131-146.

[19] Balay Rajini I, A, Lightweight Middleware Architecture and Evaluation of Middleware Performance, Ph.D. Thesis, Department of Computer Science, North Carolina State University, 1998.

[20] Kristensen, T., Plagemann, T, "Extending the Object Request Broker COOL with Flexible QoS Support," Technical Report UniK - Center for Technology, University of Oslo, January 1999.

[21] T. Plagemann, T., F. Eliassen, V. Goebel, T. Kristensen, H. O. Rafaelsen, "Adaptive QoS Aware Binding of Persistent Objects," in Proceedings of International Symposium on Distributed Objects and Applications (DOA'99), Edinburgh, Scotland, IEEE, September 1999.

[22] T. Plagemann, A Framework for Dynamic Protocol Configuration, Dissertation at Swiss Federal Institute of Technology, Computer Engineering and Networks Laboratory, Zurich, Switzerland, Sept. 1994.

[23] A. Andersen, G. S. Blair, G. Coulson, F. Eliassen, "A Reflective Component-Based Middleware in Python," NORUT IT, IPC8, September 1999.

[24] G. S. Blair, G. Coulson, P. Robin, M. Papathomas, "An Architecture for Next Generation Middleware," In Proceedings of Middleware '98, Springer, September 1998.

[25] R. Hayton et al., "FlexiNet Architecture Report," ANSA Phase III report, February 1999.

[26] G. Coulson, "A Configurable Multimedia Middleware Platform," IEEE Multimedia, Vol. 6, No. 1, January-March 1999.

[27] D. Schmidt, "The ADAPTIVE Communication Environment: Object-Oriented Network Program -ming Components for Developing Client/Server Applications," 11th and 12th Sun Users Group Conference, Dec. 1993-June 1994.

[28] D. Schmidt, D., D. Levine, C. Cleeland, "Architectures and Patterns for High-performance, Real-time CORBA Object Request Brokers," Advances in Computers, Academic Press, Ed., Marvin Zelkowitz, 2000.

[29] J. Stankovic, "VEST: A Toolset For Constructing and Analyzing Component-Based Operating Systems for Embedded and Real-Time Systems," University of Virginia TR CS-2000-19, July 2000.

[30] S. Vestal, "MetaH: Support for Real-Time Multi-Processor Avonics," Real-Time Systems Symposium, 1997.

[31] D. Isovic, M. Lindgerg and I. Crnkovic, "System Development with Real-Time Components", Malardalen University, Sweden, http://www.mrtc.mdh.se.

[32] D. Isovic, M. Lindgerg, "Real-Time Components", Malardalen University, Sweden, http://www.mrtc.mdh.se.

[33] R. N. Taylor, N. Medvidovic, K. M. Anderson, et al., "A component-and Message-Based Architecture Style for GUI Software", IEEE Transactions on Software Engineering, Vol. 22, No. 6, June 1996, pp. 390-406.

저자약력

이 승 룡

1978년 고려대학교 재료공학 (학사)
1987년 Illinois Institute of Technology (IIT), Chicago, Illinois, USA, 전산학 석사.
1991년 IIT, Chicago, Illinois, USA, 전산학 박사.
1991년-1993년 Governors State University, Illinois, Department of Computer Science 조교수
1993년-현재 경희대학교 전자정보학부 부교수
관심분야 : 내장형 시스템, 실시간 시스템, 미들웨어 시스템, 멀티미디어 시스템