2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

자바를 이용한 실시간 CORBA 이벤트 서비스의 구현

구태완, 강석태, 정연진, 이광모
한림대학교 컴퓨터공학과
e-mail : {taewani, stkang, yjjung, kmlee}@cie.hallym.ac.kr

The Implementation of Real-Time CORBA Event Services Using Java

Tae-Wan Gu, Seok-Tae Kang, Yeon-Jin Jung, Kwang-Mo Lee
Dept. of Computer Engineering, Hallym University

요 약

대표적인 분산 객체 환경을 위한 프레임 웍으로 1990 년대 초 OMG(Object Management Group)에서 발표한 CORBA(Common Object Request Broker Architecture)를 들 수 있다. 이것은 분산, 이기종 환경에서 객체기반 응용 프로그램들의 재사용성(reusability), 이식성, 상호 운용성(interoperability)을 위한 공통 프레임 웍이다. 하지만 표준 CORBA 는 실시간 응용 프로그램 지원을 위한 한계점을 내재하고 있으므로 이를 그대로 적용하기에는 적합하지 않다. 때문에 RT-SIG(Real-Time Interest Special Group)에서는 실시간 어플리케이션을 지원하기위한 일련의 작업이 이루어졌고, 그 산물로는 TAO(The Ace Orb)와 NraD(US Navy Research and Development), Iona 사의 Orbix ORB, Lockheed Martin 사에서 개발한 CORBA 등이 있다. 하지만 표준 CORBA 명세서에 따르면 COS(CORBA Object Service)중의 하나인 이벤트 서비스는 실시간 응용프로그램을 지원하기 위해 필요한 실시간 디스패칭과 스케줄링의 보장, 중앙 집중화된 이벤트 필터링과 상관성을 위한 명시, 주기적 처리 지원 등의 기능이 결여되어 있기 때문에 이를 지원하는 실시간 이벤트 서비스가 요구된다. 이러한 요구에 부합하기 위해 필요한 요구사항을 자바 API 형태로 구현하고 향후 성능에 대한 개선 방향을 제시한다.

1. 서론

CORBA(Common Object Request Broker Architecture)는 클라이언트나 서버 객체의 동작 환경에 관계 없이 클라이언트가 구현 객체에서 제공하는 메소드를 자유롭게 요청해서 사용할 수 있는 분산 환경을 제공해준다. 특히 CORBA 이벤트 서비스는 객체간의 비동기적 통신에서 유연한 모델을 제공하는 장점을 가지고 있음에도 불구하고 실시간 응용 프로그램을 지원하기에는 부족한 부분이 많다. 그렇기 때문에 1996 년 RT-SIG (Real-Time Special Interest Group)에서는 CORBA 백서를 발표하였는데 여기에는 표준 CORBA 를 실시간 CORBA 로 확장 개발할 때 필요한 기술 및 개념을 정의하고 있다. 이 백서에 따르면, 실시간 이벤트 서비스는 다양한 구성 요소들로 이루어져 있어 실시간 응용 프로그램을 지원하기에 적합한 구조를 제공하고 있다.

2. 관련연구

2.1 표준 CORBA 이벤트 서비스

표준 CORBA 호출 모델이 가지고 있는 한계점들을 완화시키기 위해 설계된 것이 CORBA 객체 서비스(CORBA Object Service, COS)중의 하나인 이벤트 서비스인데 표준 COS 이벤트 서비스 명세서는 공급자(supplier)와 소비자(consumer)의 쌍(pair)으로 정의하고 있다. 여기서 소비자란 이벤트 공급자와 공급자로부터 전달된 이벤트와 이벤트 공급자를 연

결(bind)하기위한 CORBA 객체이고, 공급자는 소비자를 위한 이벤트 생성과 이벤트 소비자를 이어주는 역할을 하는 CORBA 객체이다[5]. 그리고 비동기적 메시지 전송을 지원하는데 있어 한 개 이상의 공급자가 한 개 또는 그 이상의 소비자에게 메시지를 전송하는 것을 허용한다.

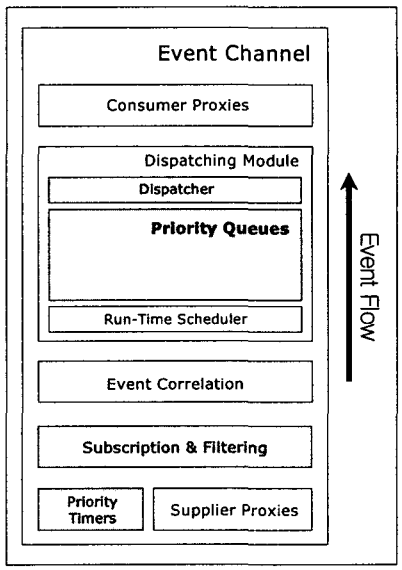
그리고 표준 CORBA 이벤트 서비스의 구성요소로는 공급자, 소비자, 이벤트 채널로 구성되는데 공급자와 소비자간의 연결을 담당하는 모듈이 바로 이벤트 채널이다. 이렇게 소비자와 공급자를 이어주는 매개체 역할을 하는 이벤트 채널은 표준 CORBA 이벤트 서비스의 핵심이다[3]. 즉, 공급자는 반드시 소비자에게 이벤트를 밀어 보내기 위해 이벤트 채널을 사용해야 한다는 말이 된다. 그러나, 표준 CORBA 이벤트 서비스에서 정의하는 이벤트 채널은 실시간 응용프로그램을 지원하기 위해 필요한 실시간 이벤트 디스패칭(dispatching)과 스케줄링의 보장, 중앙 집중화된 이벤트 필터링과 상관성을 위한 명시, 주기적 처리 지원 등의 기능이 결여 되어 있어 이를 지원할 수 있는 실시간 이벤트 서비스가 요구된다.

2.2 실시간 CORBA 이벤트 서비스

CORBA 이벤트 서비스는 이벤트에 기반한 객체 상호간의 통신을 제공하는 유연성있는 모델이지만 항공 시스템이나 멀티미디어 스트리밍 서비스 혹은 네트워크 관리와 같은 실시간 응용프로그램을 지원하기에는 부족한 부분이 많다[2]. 결국 분산 환경에서 요구되는 실시간 이벤트 서비스를 지원하기 위한 다양한 방법이 제기되었는데 대표적인 예로 TAO(The Ace ORB)의 이벤트 서비스와 NraD(US Navy Research and Development Laboratories), 그리고 RTESC(Real-Time Event Service Center)[2] 등이 있다. 하지만
본 논문의 주된 구현 관점은 실시간 이벤트 서비스의 이벤트 채널 모듈에 초점이 맞추어져 있으며 그 구성요소는 다음과 같다[1].

- 이벤트 채널(Event Channel)
- 소비자 프록시 모듈
- 공급자 프록시 모듈
- 등록(subscription) 및 필터링(filtering)
- 우선순위(priority) 타이머
- 이벤트 상관성
- 디스패칭(dispatching)의 구성요소 및 기능

그러나 실시간 이벤트 서비스 모델에서 이벤트 채널과 공급자/소비자 프록시 모듈은 기존의 CORBA 이벤트 서비스에서 제공하는 기능과 동일하게 수행된다. 그러나 소비자가 단지 공급자로부터 불필요한 이벤트를 전달받지 않기 위한 이벤트 등록(subscription)과 필터링(filtering), 채널에 등록된 모든 타이머를 관리하는 특별한 목적의 우선순위 타이머 프록시, 그리고 디스패칭 모듈에서 런타임 스케줄링 서비스의 성능 향상을 위한 쓰레드 풀을 이용한 디스패처(dispatcher) 부분은 QoS(Quality Of Service)를 지원하기 위한 실시간 CORBA 이벤트 서비스를 위해 제안된 구조이다. 본 논문에서는 이같은 사항을 중심으로 구현하며 전반적인 실시간 이벤트 서비스 구조는 다음과 같고 세부사항은 TAO 의 이벤트 모델 구조를 따른다.

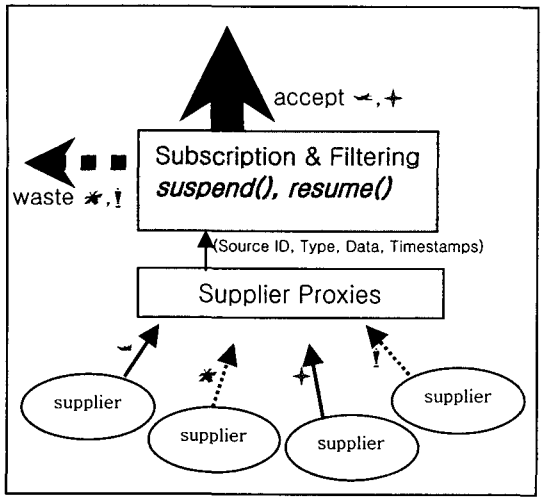


<실시간 이벤트 채널의 구조>

3. 등록(subscription)과 필터링(filtering)

표준 CORBA 이벤트 서비스는 이벤트 채널을 공급자로부터 전달되는 이벤트들을 소비자에게 전달 해주는 역할로 규정짓고 있는데 만일 소비자가 일부 특정한 이벤트에만 관심있고 나머지에 대해서는 그렇지 않다면, 나머지 이벤트들은 모두 버려져야만 하며 이러한 메커니즘은 자원의 비효율적 사용의 원인이 되어 실시간 환경에서 요구되는 사항들을 제대로 만족시키기 어렵다. 때문에 이러한 요구사항을 만족시키기 위해 표준 CORBA 이벤트 서비스의 이벤트 채널 모델을 확장하여 실시간 이벤트 채널은 주어진 조건을 만족하는 이벤트들에 한정하여 전달을 수행하는 구조를 갖게 되는데 이것은 채널로 하여금 통신 과부하를 줄이고 효율성을 높일 일 수 있는 이점을 갖게 된다.

이벤트 등록(subscription)과 필터링(filtering) 모듈은 suspend()와 resume() 연산으로 필요한 이벤트만을 필터링하고 소비자가 필요한 이벤트들의 집합을 구성하기 위해 이벤트 등록 기능을 추가한다. 이때, 필요한 이벤트와 불필요한 이벤트의 선별은 Supplier Proxies 를 통해 들어온 이벤트들을 대상으로 하며 필요한 조건들은 공급자 모듈에서 정의되어야 하는데 그 예로는 source ID, 타입(type), 데이터(data), 타임스탬프(timestamp)가 있다.



<Subscription & Filtering 모델 >

이에 따른 이벤트 채널의 등록 및 필터링 모듈의 구조는 다음과 같다.

```
class EC_Filter
{
  private int SID, Type, Data, Timestamps;

  public boolean suspend(int condition);
  public void resume();
  public EventChannel push(String ch_name);
   .....
}
```

여기서 EC_Filter 는 suspend() 메소드를 이용하여 주어진 조건에 따라 입력된 채널을 선별하여 조건에 부합되는 이벤트만을 받아들이고 resume() 메소드를 이용하여 이벤트의 흐름을 계속해서 처리 할 수 있도록 하여 채널의 오버헤드를 줄이고 효율성을 높일 수 있도록 한다

4. 우선순위 타이머(priority timer)

표준 COS 이벤트 서비스의 이벤트 채널 모델은 Supplier Proxies 모듈을 갖고 있는데 실시간 이벤트 채널 모델에서의 Supplier Proxies 모듈은 이에 부가적으로 특별한 목적의 우선순위 타이머(priority timer)를 포함하고 있다. 이는 이벤트 채널에 등록된 모든 채널에 대하여 타이머를 관리하는 역할을 하게 되며 소비자가 각 이벤트에 대하여 timeout 을 등록하게 된다. 이때 우선 순위 타이머는 런타임 스케쥴러와 함께 작용하게 된다. 또한 우선 순위 타이머는 heap 구조 기반으로 구성된 큐(queue)형태로 구성되며 이러한 메커니즘은 메모리에 미리 할당(preallocate)되는 구조를 갖게 된다. 그래서 평균적으로 최악의 경우 시간 복잡도(time complexity)는 *O(log N)*의 복잡도를 갖게 된다[1]. 다음은 Supplier Proxies 모듈 내 우선순위 타이머를 Vector 형태로 정의하고 있다.

```
private java.util.Vector _proxyTimer;
public void setPriorityTimer() {
    _proxyTimer = new java.util.Vector();
}

public int getPriorityTimer(int timeout) {
    _proxyTimer.addElement(timeout);
    return timeout;
}
```

5. 쓰레드 풀을 이용한 디스패처(dispatcher) 구현

디스패처는 큐(queue)에서 이벤트와 소비자의 쌍을 꺼내기 위해 push() 연산을 사용한다. 이 연산을 이용해 이벤트를 소비자들에게 전달하고 이때, 이러한 일련의 동작들은 디스패처의 쓰레드 풀(thread pool)을 사용하게 되는데, 우선 런타임 스케쥴러는 event/consumer 의 쌍을 인자로 해서 enqueue() 연산을 이용해 큐에 전달하고 이것은 다시 디스패처에서 쓰레드를 이용해 큐에서 event/consumer 의 쌍을 dequeue() 연산을 이용해 제거하게 된다.

런타임 스케쥴러는 디스패처(dispatcher)가 큐에서 event/consumer 의 쌍을 제거하기 위한 순서를 결정하고 이에 따라 쓰레드 풀의 쓰레드는 높은 우선순위를 갖는 쓰레드 순으로 이벤트 디스패칭이 이루어 진다. 디스패처는 미리 생성된 쓰레드들의 집합으로 구성되어 있다. 이때 생성된 쓰레드는 각기 서로 다른 우선순위를 가지게 되며 런타임 스케쥴러에서 enqueue 된 이벤트들을 각기 다른 우선순위의 쓰레드가 처리하여 그 효율을 높이도록 설계한다.

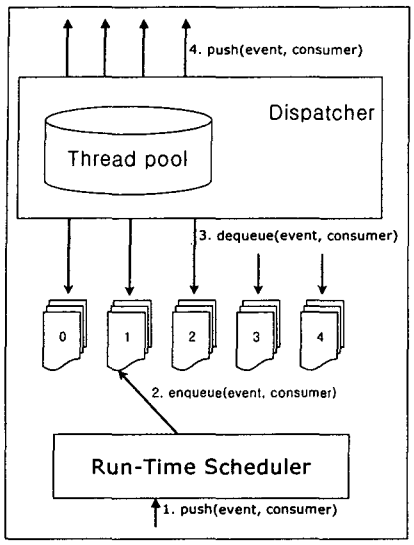
이러한 방법은 TAO 에서 제안된 RTU 디스패칭 모듈과 쓰레드 풀을 이용한 디스패칭 모듈, 그리고 EFD 디스패칭 모듈이 있는데[1] 이러한 방법들 중 쓰레드 풀을 이용하는 방법의 장점으로는 런타임 스케쥴러에 의해 큐에 들어온 이벤트들에 대하여 우선순위를 갖는 쓰레드가 각 큐를 지정하여 처리해 소비자에게 전달함으로써 쓰레드 풀이 갖는 장점을 활용하고자 함이다.

다음은 디스패처 모듈내에서 쓰레드 풀을 생성하는 모습을 보여준다

```
private java.util.Vector dispatcher;
private QueueObject qo;
qo = new QueueObject(Num);
dispatcher = new java.util.Vector();
```

그런 다음 생성된 쓰레드 풀을 이용하여 큐에서 dequeue() 메소드를 이용하여 이벤트를 consumer 에게 전달하게 된다.

```
dispatcher.dequeue(channel_name, consumer);
dispatcher.push(channel_name, consumer);
```



<쓰레드 풀을 이용한 디스패처>

6. 결론 및 향후 연구 계획

본 논문에서는 COS 의 이벤트 서비스를 확장하여 실시간의 요소를 충족시키는 이벤트 모듈을 구현하였다. 이 요소는 첫번째로 등록 및 필터링 모듈과 두번째로 Supplier Proxies 모듈내의 우선순위 타이머 모듈과 그리고 세번째로 쓰레드 풀을 이용한 디스패처 모듈의 구현인데 기존에 이미 구현된 실시간 이벤트 서비스 모델인 TAO 를 기초로 하여 자바 언어로 구현한 것이 본 논문의 핵심이다. 이것은 자바 언어가 갖는 장점과 실시간을 지원하는 이벤트 서비스를 접목함으로서 구현상의 편리함과 그에 따른 성능유지가 관건이 된다.

향후 이와 관련한 연구 계획으로는 TAO 기반의 이벤트 서비스의 수동성을 탈피한 보다 적극적인 이벤트 서비스 모듈을 구현하는 것일 수 있는데, 이를 위한 대안으로는 이벤트 supplier 가 발생시키는 이벤트를 Supplier Proxies 모듈 이전에 탐지(detection)할 수 있는 이벤트 탐지 모듈(event detection module)을 두어 보다 능동적인 이벤트 서비스를 제공함으로서 분산 환경에서 QoS 를 보증하기 위한 서비스를 구현할 수 있을 것이다[2].

그리고 디스패칭 모듈에서 각기 제안된 3 가지 사안들을 동시에 만족시킬 수 있는 인터페이스의 구현도 성능을 향상시키고 이벤트의 동적 처리가 가능한 디스패칭 모듈을 구현할 수 있을 것이다.

참고문헌

[1] T.H. Harrison, D.L. Levine and D.C. Schmidt, "The Design and Performance of a Real-Time CORBA Event Service". Proc. OOPSLA '97, 1997

[2] Guangtian Liu and Aloysius K. Mok, "An Event Service Framework for Distributed Real-Time Systems"

[3] Object Management Group, "The Common Object Request Broker:Architecture and Specification", February, 1997

[4] Distributed Object Group, "JavaORB Event Service Version. 1.1", 1999

[5] OpenORB, "OpenORB Event Service", February, 2001