분산시스템에서 다중 이벤트 처리를 위한 분해/통합 서비스 객체 설계

*김형준 · *이경복 · *이재완

*군산대학교 정보통신공학과 분산처리연구실

A Design of Decomposition/Integration Service Object for Processing of Multiple Events in Distributed System

*Hyung-Jun Kim · *Kyung-Bok Lee · *Jae-Wan Lee

*Dept. of Telecommunication Eng., Kunsan National Univ.

E-mail : artemis@Knusun2.Kunsan.ac.kr

요 약

네트워크의 발달과 멀티미디어 기술의 발달함에 따라 사용자들의 서비스 요구는 더욱 복잡화되고 있으며, 이벤트 구조도 점차 다중화되고 복잡 해지고 있다. 따라서 다중 구조의 이벤트를 효율적으로 검색할 수 있는 연구가 필수적이다. 본 연구에서는 CORBA의 비동기적인 이벤트 서비스를 기반으로 다중 구조의 검색을 처리하는 서비스 구조를 제안한다. 다중 구조의 이벤트처리를 위해 기존의 이벤트 서비스에 다중 이벤트를 단일 이벤트로 나누어 수행하고 이를 통합하여 결과를 제공하는 분해/통합 서비스 객체를 추가하여 기존의 이벤트 서비스를 확장하였다. 본 연구는 CORBA를 기반으로 하는 서비스와 연결하여 다중 구조 검색 등에 활용할 수 있다.

Ⅰ. 서 론

분산 기술에 대한 필요성과 관심의 증가로 인해 여러 가지 오브젝트를 기반으로 한 분산 미들웨어들이 출현하고 있다. 이들 중 대표적인 것이 OMG(Object Management Group)에서 제안한 CORBA(Common Object Request Broker Architecture)이다. CORBA는 객체들을 실행하는데 있어 특정한 플랙폼과 기술들에 상관없이 객체들을 통합 운용할 수 있는 기본 구조를 제공한다[1,2,3].

분산 응용 프로그램들은 이벤트 위주의 수행 모델을 사용하여 비동기적인 통신을 통해 데이터를 전달한다. 분산 객체 컴퓨팅의 표준으로 제안된 CORBA 모델은 하나의 클라이언트와 서버가 동기적으로 연결되는 형태로서, 표준 CORBA 통신 모델에서는 비동기적이나 다중 응용 프로그램간의 멀티 캐스팅(multicasting) 등의 기능을 지원하지 않는다. 따라서 OMG에서는 다중 응용 프로그램간의 비동기적인 통신을 지원할 수 있도록 CORBA 이벤트 서비스(Event Service)를 제안하였다. 이벤트 서비스는 표준 CORBA를 확장하기 위한 CORBA 객체 서비스중 하나이다[1][2][3].

소비자가 요구하는 다중 이벤트가 여러개의 검색어로서 구성된 다중 이벤트일 경우, 이벤트들을 처리할 수 있는 작은 단위의 단일 이벤트로 분해하여 수행하고 이를 통합하여 서비스하는 기법이 중요하다.

본 논문에서는 다중 이벤트들을 처리할 수 있는 작은 단위의 단일 이벤트로 나누고, 각 이벤트는 시스템 내에 유일한 식별자를 부여하여 관리하며, 이 식별자는 공급자로부터 반환된 정보를 통합할 때 사용되게 된다. 다중 이벤트들은 이벤트 서비스(Event Service)를 통해 공급자와 비동기적으로 통신하게 된다. 공급자로부터 반환된 정보들을 소비자가 원하는 정보로 추출하기위해 공급자의 이벤트들을 필터링하기위해 필터링 서비스를 사용한다.

Ⅱ. CORBA 이벤트 서비스

CORBA는 OMG(Object Management Group)에 의해서 정의된 분산 객체 컴퓨팅 미들웨어 표준이다. 유동적이고 재사용이 가능한 서비스들과 응용 프로그램의 개발을 지원하기 위해 설계되었다. 이벤트 서비스는 비동기적인 메시지나 이벤트들을 보내고 받는 결합도가 낮은 객체들을 제공한다.

CORBA 이벤트 서비스에서는 이벤트 채널

(Event Channel)을 통하여 공급자(Supplier)와 소비자(Consumer)을 연결하고, 비동기적으로 통신할 수 있도록 중재하는 역할을 하는 객체이다. 공급자와 소비자 사이에 사건 통신을 초기화하는 방법에는 두가지 방법이 있다. 이 두가지 방법은 Pull 모델과 Push 모델이다[3,4,6].

Push 모델은 사건들의 공급자가 소비자에게 사건 데이터의 전송을 초기화 시킨다. Pull 모델은 사건들의 소비자가 공급자에게 사건 데이터를 요구할 수 있게한다.

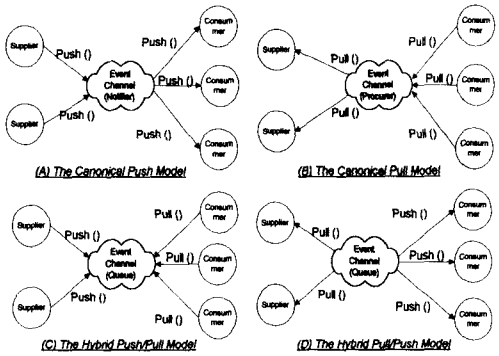

그림 1. 이벤트 서비스 통신모델

OMG COS Events Service 기본구조는 그림 1과 같이 구성요소 협동의 4가지 일반 모델을 정의하고 있다[3][5].

가. 표준 Push 모델 : 표준 Push 모델(그림 1(A))은 이벤트의 공급자가 소비자들에게 이벤트 데이터 전송를 초기화 시킨다. 이벤트 채널은 통보자(Notifier)의 역할을 한다. 능동적인 공급자들은 이벤트 채널에 등록된 수동적인 소비자들에게 데이터를 보내기(Push)위해 이벤트 채널을 사용한다.

나. 표준 Pull 모델 : 표준 Pull 모델(그림 1(B))은 소비자가 공급자로부터 사건들을 요구할 수 있다. 이벤트 채널이 소비자 대신에 이벤트들을 획득하기 때문에, 이벤트 채널은 획득자(Procurer)의 역할을 한다. 능동적인 소비자는 이벤트 채널들을 경유하여 수동적인 공급자로부터 데이터를 정확하게 받을(Pull) 수 있다.

다. 혼합형 Push/Pull 모델 : Push/Pull 모델(그림 1(C))은 소비자가 공급자에의해 이벤트 채널에 저장된 이벤트을 요구하는 혼합형이다. 이벤트 채널은 큐(Queue)의 역할을 한다. 능동적인 소비자는 이벤트 채널을 경하여 공급자에의해 위탁된 데이터를 받을(Pull) 수 있다.

라. 혼합형 Pull/Push 모델 : Pull/Push 모델(그림 1(D))은 이벤트 채널이 공급자들로부터 이벤트들을 받고(Pull) 소비자들에게 이벤트들을 보낸다(Push). 이벤트 채널은 대리자(intelligent agent)의 역할을 한다. 능동적인 이벤트 채널은 수동적인 공급자들로부터 데이터를 받고(Pull) 수동적인 소비자들에게 데이터를 보낸다(Push).

이벤트 서비스의 통신모델은 이벤트의 전달 연산자와 인자의 형에따라 일반 모델(general model)과 형정의 모델(typed model)로 구분된다. 일반 모델의 경우에는 모든 통신이 이벤트 데이터를 위한 하나의 인자를 가진 일반 push()와 pull() 연산자에 의해서 수행된다. 형정의 모델은 경우에는 통신이 OMG IDL에 정의된 인터페이스의 연산자에 의해서 이루어진다[3,4,6].

III. 다중 이벤트 처리를 위한 CORBA 환경에서의 이벤트 서비스

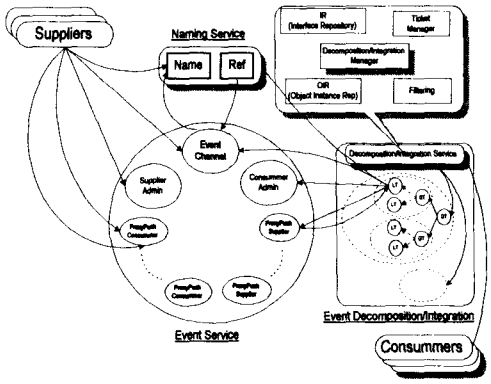

그림 2. 이벤트 서비스 구조

본 시스템은 구조는 그림 2와 같이 소비자와 생산자 사이에 이벤트들을 전달하기 위한 이벤트 서비스와 소비자로부터 요구된 다중 구조의 이벤트들을 검색할 수 있는 작은 단위인 단일 이벤트로 나누어 처리할 수 있는 이벤트 분해/통합기(Event Decomposition/Integration) 서비스 객체로 구성된다.

분해된 이벤트는 이벤트 서비스의 장점인 비동기적 통신을 이용하여 공급자에게 전송하여 단일 이벤트들을 비동기적으로 처리한다.

3.1. 이벤트 서비스 구조

3.1.1. 이벤트 서비스 설계 원칙

분산 시스템에서 이벤트 서비스는 다음과 같은 원칙을 만족한다.

가. 이벤트는 분산된 환경에서 발생한다.
나. 이벤트 서비스들은 이벤트의 다중 소비자

와 다중 이벤트 공급자가 존재할 수 있다.

다. 소비자는 공급자로부터 이벤트들을 요구하거나 또는 수신할 수 있다.

라. 이벤트들의 소비자와 생산자는 표준 OMG IDL 인터페이스를 지원한다.

마. 공급자는 단 한번에 모든 소비자에게 이벤트 데이터를 통신 하기위해 단일 표준 요구를 발생시킨다.

바. 공급자는 소비자의 식별자를 알지 못하고도 이벤트을 생성할 수 있다. 반대로 소비자 또한 공급자의 식별자를 알지 못하고도 이벤트을 수신할 수 있다.

3.1.2. 소비자, 공급자, 이벤트 채널 연결 기법

이벤트 채널(Event Channel) 인터페이스는 3가지 관리 연산을 정의한다. 소비자를 추가 하기위한 ConsumerAdmin 오브젝트를 반환하는 연산, 공급자를 추가하기 위한 SupplierAdmin 오브젝트를 반환하는 연산, 채널을 해제(Destroy)하기 위한 연산으로 구성된다.

일반적으로 소비자와 생산자가 이벤트 채널에 연결하는 절차는 그림 3과 같은 단계들을 요구하게된다.

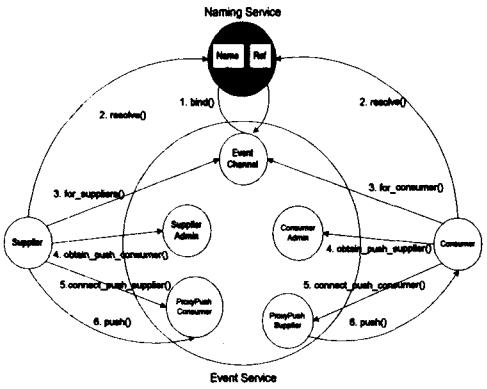

그림 3. 이벤트 채널 연결

가. 이벤트 채널은 네이밍 서비스를 사용하여 생성되고, 네이밍 서비스에 저장된다.

나. 소비자는 네이밍 서비스로부터 이벤트 채널에대한 레퍼런스를 얻고, 소비자는 이벤트 서비스에 저장된다.

다. 소비자는 이벤트 채널로부터 ConsumerAdmin 인터페이스에대한 러퍼런스를 얻는다.

라. 소비자는 ConsumerAdmin 인터페이스를 사용하여 ProxyPushSupplier 인터페이스에 대한 레퍼런스를 얻는다.

마. 마지막으로 소비자들은 ProxyPushSupplier 인터페이스를 사용하여 이벤트 채널(Event Channel)에 연결한다.

바. 공급자 등록은 소비자 등록과 동일하며, 일단 공급자가 이벤트 채널에 바인드 되면, 공급자는 이벤트 채널에 등록된 소비자에게 채널을 통하여 전송 되어진다.

소비자와 이벤트 채널과의 연결 단계를 ETD (Event Trace Diagram)으로 표현하면 그림 4와 같다.

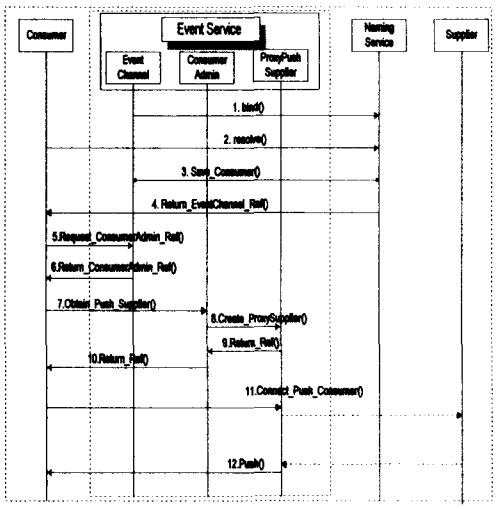

그림 4. 생산자, 소비자, 채널사이의 ETD

3.2 이벤트 분해/통합 서비스 객체 구조 및 기능

이벤트 분해/통합 서비스 객체는 다중 질의를 단일 질의로 나누어 처리하는 객체로서 그림 5와 같은 구성요소로 구성된다.

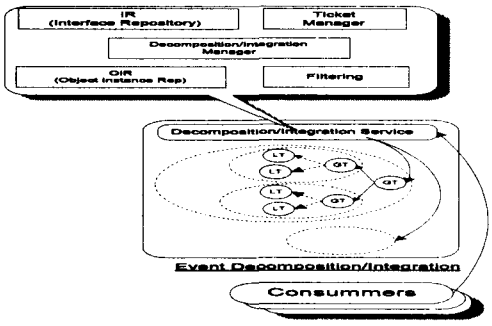

그림 5. 이벤트 분해/통합 서비스 객체

3.2.1. 이벤트 분해/통합 관리자 (Event Decomposition/Integration Manager)

이벤트 분해/통합 관리자는 소비자에의해 요구된 이벤트들을 분해하는 과정에서 객체를 생성하는 역할을 담당한다. 또한 이벤트들을 처리한 후

객체를 삭제하는 역할을 한다. 소비자로부터 이벤트가 들어오면 관리자는 IR(Interface Repository) 객체을 사용하여 이벤트 처리객체를 생성하고, 티켓관리자를 통하여 생성된 객체에 대해 전역적으로 유일한 티켓번호와 함께 이벤트 처리 객체를 생성한다. 생성된 이벤트 처리객체의 레퍼런스를 OIR(Object Instance Repository)에 저장한다. OIR에 저장된 정보는 후에 객체를 삭제할 때 이용하게 된다. 알고리즘은 그림 6과 같다.

```
MultipleEvent( τ_i^G )
begine
    Get τ_i^G's MultipleEvnent Ticket
    Decompose τ_i^G into τ_i[ ]
    SubgroupNum = Size( τ_i[ ])
    while DecomposeCount < SubgroupNum
    begine
      Localbegine( τ_i[DecomposeCount], SingleEventNum())
      DecomposeCount = DecomposeCount + 1
    end
    Compose τ_i[ ] to τ_i^G
end

SingleEvent( τ_1 , k)
begine /* τ_1 is sub Event , k is subgroup number assigned
       /* by SingleEventNum() function
  if τ_1 is single event then
  begin
   connect to Naming Service and receive EventChannel's Reference
   connect to EventChannel with EventChannel's Reference
   query Single Event to supplier throught ProxyPull supplier
  end

  else

  begine
   decompose τ_1     to Sub τ_1[ ]
   SubgroupNum = Size(Sub τ_1[ ])
   while DecomposeCount < SubgroupNum
   begine
    Localbegine(sub τ_1[DecomposeCount], SingleEventNum())
    DecomposeCount = DecomposeCount + 1
   end
  end
  compose sub τ_1[ ] to τ_1
end
```

그림 6. 이벤트 분해/통합 서비스 객체 알고리즘

3.2.2 인터페이스 저장소(Interface Repository)

인터페이스 저장소는 객체를 생성하기 위해 필요한 정보를 보관하며, 이벤트 분해/통합기 관리자의 요구에의해 템플릿을 반환한다.

3.2.3 객체 인스턴스 저장소 (Object Instance Repository)

객체 인스턴스 저장소는 이벤트 분해/통합기 관리자에의해 생성된 객체들을 저장하는 저장소이다. 관리자에의해 생성된 모든 객체들은 티켓관리자로부터 할당 받은 식별자와 객체에대한 레퍼런스를 객체 인스턴스 저장소에 저장한다. 인스턴스저장소에 저장된 내용은 객체를 삭제할 경우 인스턴스 저장소에 저장된 레퍼런스를 이용하여 객체를 삭제하게 된다. 이벤트를 분해하는 도중 에러가 발생할 경우 기존에 생성된 모든 객체들을 삭제하고, 재구성하게 되는데 이때 객체를 삭제할때도 사용하게 된다.

3.2.4 티켓 관리자(Ticket Manager)

소비자에의해 요청된 다중 이벤트는 검색할 수 있는 작은 단위의 단일 이벤트로 분해하게 되는데 이때 각 이벤트 처리객체들은 티켓 관리자로부터 그룹내에 유일한 식별자를 부여받게된다. 이벤트 분해/통합기 객체의 요구에의해 티켓을 부여하게되며 하나의 이벤트 처리객체에 한 개의 식별자를 부여하게된다. 이 식별자는 이벤트를 통합할 때 사용하게된다. 티켓 관리자 객체에서는 식별자를 할당할 때 NTNT(Nested Tickets Method for Nested Transaction)알고리즘을 사용한다[7].

NTNT의 주요한 개념은 모든 단계에서 전역 트랜잭션에 유일한 티켓을 부여하는 것이다. 즉, 부모 트랜잭션과 자식 트랜잭션들 모두 티켓을 얻게된다. 티켓을 부여받을 때 자식 트랜잭션은 부모 트랜잭션보다 큰 티켓번호를 부여받게 되고, 할당받은 태켓번호가 이전에 할당받은 티켓 번호보다 작은 경우에는 트랜잭션을 다시 구성하게된다. 이때 객체 인스턴스 저장소에 저장된 객체에 대한 레퍼런스를 사용하여 객체를 삭제하게된다.

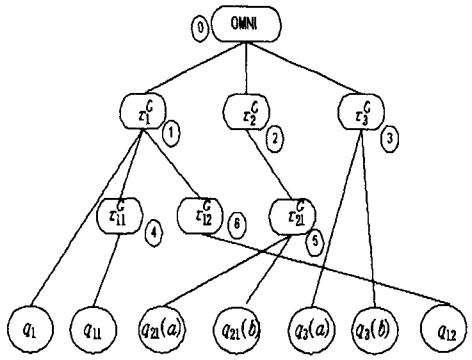

그림 7. 다중 이벤트 분해 구조

분해/통합 관리자에 요청된 다중 이벤트는 티켓관리자에 의해 티켓을 할당받게 되는데 구조가 그림 7과 같은 형태의 단일 이벤트가 분해된다고 가정하면 TN(OMNI) = 0, TN(τ_1^G) = ①, TN (τ_2^G) = ②, TN(τ_3^G) = ③, TN(τ_{11}^G) = ④, TN

(τ_{12}^{C}) = ⑥, TN(τ_{21}^{C}) = ⑤을 가진다(OMNI : 가상 트랜잭션, ① ~ ⑥ : 티켓번호(식별자)).

3.2.5 필터링(Filtering)

CORBA 이벤트 서비스는 공급자로부터 모든 이벤트들이 소비자에게 전달되는데, 만약 공급자로부터 전달되는 이벤트에 대해서 어떤 부분집합에 대해서만 관심을 두고 있는 소비자라면 필요치 않은 이벤트들을 취소 시킬 수 있는 이벤트 필터링이 필요하다. 본 논문에서 적용할 독립형 이벤트 필터링의 구조는 그림 8과 같다.

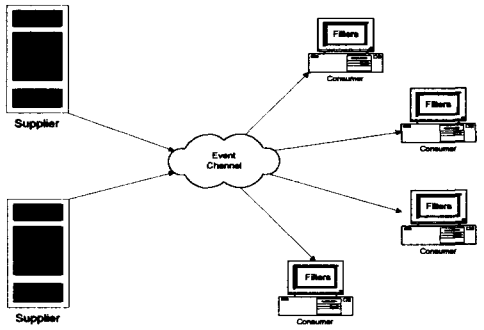

그림 8. 독립형 이벤트 필터링

다중 공급자로부터 반환된 결과들은 이벤트 분해/통합기 서비스 객체에서 이벤트들을 통합할 때 필터링 객체를 사용하여 정보을 추출하게된다. 만약 필터링을 사용하지 않는 경우에는 처리과정에서 비용(cost)의 낭비를 가져오기 때문이다.

IV. 결 론

사용자들의 요구의 구조가 복잡해 짐에 따라 사용자들은 다양한 검색조건을 요구한다. 이러한 요구을 만족시키기 위해서는 여러조건의 다중 이벤트들을 동시에 검색할 수 있는 이벤트의 분해/통합기법이 필요하다.

본 논문에서는 다중 구조의 이벤트처리를 위해 기존의 이벤트 서비스에 다중 이벤트를 단일 이벤트로 나누어 수행하고, 이를 통합하여 결과를 제공하는 분해/통합객체를 추가하여 기존의 이벤트서비스를 확장하였다. 다중 이벤트들을 단일 이벤트들로 분해하여 처리하는 과정에서, 각각의 처리객체에 유일한 식별자를 제공하였다. 분해/통합 관리자가 단일 이벤트들을 통합하는 과정에서 사용할 수 있게하였다.

기존의 CORBA가 지원하는 생산자와 소비자간의 다중 응용 프로그램간에 비동기적 통신을 지원할 수 있도록 확장하였다. 본 논문에서 설계한 확장한 이벤트 서비스는 데이터베이스와 다중 질의 검색을 요구하는 서비스에 사용 될 수 있다.

향후 연구과제로는 비동기적 수행 방법의 문제점인 신뢰성 및 안전성 확보를 위한 계속적인 연구가 필요하다.

참고문헌

[1] OMG, "CORBA 2.0 Specification" http://www.omg.org/corba/iiop.htm, 1995

[2] J. Siegel, CORBA Fundamentals and Programming, John Wiley & Sons Inc., pp. 196-204, 1966

[3] OMG, "Event Management Service," ftp://ftp.omg.org/pub/docs/formal/97-07-13.ps, 1997

[4] P. Felber, R. Guerraoui, and A. Schiper. Replicating Objects using the CORBA Event Service. In Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems, Tunis, October 1997.

[5] D.C. Schmidt, S. Vinoski, Object Interconnections: The OMG Events Service(Columm 9), SISG C++ report Magazine, February 1997.

[6] 정혜영, 김현규, 박양수, 구자록, 이명준, "자바를 이용한 CORBA 이벤트 서비스의 개발" 한국정보과학회, 가을학술발표논문집, Vol. 24, No. 2, 10, 1997.

[7] Dogac.,A., Dengi, C., Kilic, E., Ozhan, G., Ozcan, F., Nural, S., Evrendilek, C., Halici, U., Arpinar, B., Koksal, P., Kesim, N., Mancuhan, S., "A Multidatabase System Implementation on CORBA", 6th Intl. Workshop on Research Issues in Data Engineering (RIDE-NDS '96), New Orleans, Febru

[8] D.C. Schmidt, S. Vinoski, Object Interconnections: The OMG Events Service(Columm 10), SISG C++ report Magazine, February 1997.