이벤트 알림 서비스의 구조설계와 성능분석

한영태*, 민덕기*

Architecture Modeling and Performance Analysis of An Event Notification Service

Youngtae Han, Dugki Min

Abstract

Event notification service is a event-based messaging middleware service needed for various vertical domains, such as, business applications, distributed system management, and web service integration. In this paper, we investigate the architecture of an event notification service that includes a subject-based event dissemination service and a flexible message communication service. The event dissemination service is in charge of transferring events asynchronously but speedy according to the subjects of events and their environmental knowledge. It also includes content-based message filtering. The message communication service provides a common communication infrastructure supporting variety types of messages and variety of protocols. Depending on application domains and situation, we can re-configurate the communication infrastructure in order to optimize the efficiency and usability. This paper shows the performance analysis of our event notification service with various types of message formats and protocols.

Key Words: Event Notification Service, Message Communication Infrastructure, Event Dissemination Service, Performance Analysis

* 건국대학교 컴퓨터·정보통신공학과

1. 서 론

이벤트 기반 시스템에 대한 연구는 분산 어플리케이션 미들웨어, 이벤트 기반 메시징 모델, 이벤트 구동 시스템, 그리고 메시지 기반 미들웨어(MOM: Message Oriented Middleware) 등의 영역에서 많은 연구들이 이루어지고 있다[1]. 특히 인터넷 기반에서 확장성 있는 분산 시스템 구축을 위해 좀더 유연성 있는 통신 구조와 시스템 구성을 위한 연구가 이루어졌다.

일반적으로, 응용 프로그램에서 사용하는 이벤트 기반 서비스를 다음의 두 가지로 나누어 볼 수 있다. 첫 번째는 이벤트를 발생시키는 생산자(Supplier)와 이벤트를 통지받는 소비자(Consumer)가 채널과 같은 중간의 중재자를 통해서 통신하는 간접적인 통신(Indirect Communication) 방식이다. 두 번째는 생산자와 소비자 빠른 실시간 알림을 위한 직접적인 통신(Direct Communication) 방식이다.

본 논문에서 제시하는 이벤트 알림 서비스는 생성자와 소비자 간에 직접적인 통신을 하기 위하여 개발되었다. 제시된 이벤트 알림 서비스는 이벤트 분배 프로세서와 이벤트 통신 인프라스트럭처로 구성되어 있다. 이벤트 분배 프로세서는 한 호스트에서 여러 응용 프로세서들이 생성하는 이벤트를 다른 이벤트 분배 프로세서로 전송하기 위한 공유 프로세서이다. 이벤트 분배 프로세서의 경우 다음 세 가지 특징을 가지고 있다. 1) 비동기적 이벤트 분배, 2) 주제 기반 메시지 분배, 그리고 3) 내용 기반 메시지 필터링이다. 이벤트 통신 인프라스트럭처는 다수개의 기계간 통신을 위해 개발되었다. 다양한 메시지 형식 전송과 다양한 통신 프로토콜을 지원하기 위하여 Configurable 컴포넌트로 구현되어 적응성(Flexibility)이 좋다. 또한 빠른 전송을 위해 최적화되어 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존에 개발되어진 이벤트 알림 서비스들을 소개한다. 3 장에서는 이벤트 알림 서비스의 구조를 제시하고, 설계와 구현에 대하여 알아보겠다. 그리고 응용 프로그램 적용 이슈에 대해 살펴보고자 한다. 4 장에서는 구현된 서비스에 대한 성능 측정 결과를 살펴본다. 그리고 마지막 5 장에서 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

이벤트 기반 서비스에 대한 연구는 다음 세 가지 관점에서 이루어져 왔다. 이벤트 모델의 구성 형태에 따른 분류, 아키텍쳐 구성에 따른 분류, 그리고 Subscription 방식에 따른 분류 관점이다.

첫째로, 이벤트 모델의 구성 형태에 따라 다음의 3 가지 모델로 나누어 연구되어 왔다[2]. 튜플 기반, 레코드 기반, 그리고 객체 기반 모델로 나눌 수 있다. 튜플 기반 모델에서는 이벤트 정보를 문자열의 셋으로 정의하고 있다. 예를 들어 (EventDomain, 5, datainfo, ...) 과 같이 표현되는 것을 말하는 것이다. 레코드 기반 모델에서는 이벤트 정보를 이름과 값을 가지는 정형화된 필드의 집합으로 정의하고 있다. 대부분의 이벤트 서비스가 이 모델을 사용하고 있다. 마지막으로 객체 기반 모델은 레코드 필드와 더불어 메소드의 집합으로 이벤트를 정의하고 있다.

둘째로, 기본 소프트웨어 구성 요소의 아키텍쳐 관점에서 클라이언트-서버 모델과 Peer-to-peer 분산 모델로 나누어 연구되었다. 클라이언트-서버 모델에서 이벤트 서버는 이벤트를 수신하고 저장하고 분배하는 역할을 수행하며 확장성을 보장하기 위하여 분산 서버 환경으로 구현되기도 한다. 이벤트 클라이언트는 이벤트를 생성하여 이벤트 서버에 전

송하거나 이벤트를 소비거나, 또는 두 가지 역할을 다하기도 한다[2,3,4]. Peer-to-peer 분산 모델에서는 모든 노드가 이벤트 생성, 이벤트 분배, 그리고 이벤트 소비하는 모든 역할을 다하는 방식이다. 이벤트를 분배하기 위한 Root 노드를 중심으로 응용 프로그램 레벨의 멀티캐스트 라우팅으로 이벤트를 전송하는 방식이다[5,6,7].

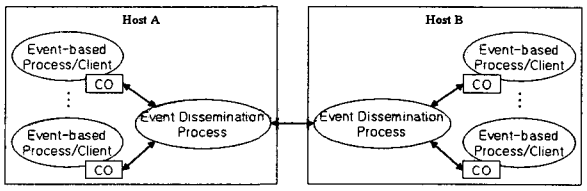
마지막으로, 이벤트 소비자의 Subscription 방식에 따라 내용 자유(Content-free), 주제 기반(Subject-based), 내용 기반(Content-based), 그리고 이벤트 조합(Event Combination) 방식으로 나누어 연구되었다. 내용 자유 방식은 Subscription을 정해진 채널과 하고 있으면서, 채널에 등록된 모든 이벤트를 모두 수신하는 방식을 말하는 것이다. 대표적인 구현으로 CORBA Notification Service[8]를 들 수 있다. 다음으로 주제 기반 방식은 이벤트 소비자들이 관련된 주제를 Subscription 하여 선택적인 데이터 전송을 받는 방식을 말한다[5,6]. 이를 지원하기 위해서는 모든 이벤트에 주제를 포함시켜야 한다. 다음으로 내용 기반 방식은 Subscription 할 때 관심 포함된 내용 중에서 특정 조건을 만족하는 이벤트를 표현하기 위한 식을 기술하는 방식이다[7,9,10]. 이때 기술되는 식은 Disjoint elementary expressions, Compound expressions, Regular expressions 등을 사용하고 있다. 이벤트 조합 방식은 전송된 한 개 이상의 이벤트들에 대한 조합을 Subscription에 기술하는 방식이다. 이 방식은 이벤트에 대하여 캐싱이 필수적이다. 대표적인 구현으로는 Yeast[11] 가 있다.

3. 이벤트 알림 서비스

본 절에서는 이벤트 알림 서비스를 제시하고 구현에 대하여 살펴보도록 한다. 이 서비스는 이벤트 분배 프로세서를 이용하여 생성자와 소비자 간의 전송 효율을 증대시켰다. 그리고 이벤트 통신 인프라스트럭쳐를 정의하여 다양한 통신 프로토콜과 데이터 형식을 제공하고 있다.

3.1 이벤트 알림 서비스 구조

본 논문에서 제시하는 이벤트 알림 서비스는 이벤트 생성자가 생성한 이벤트를 이벤트 소비자에게 직접 전송하는 것을 지원하기 위해 개발된 서비스이다. 다수의 이벤트 생성자가 생성한 이벤트는 알려진 이벤트 소비자에게 전송이 되기도 하지만 불특정 이벤트 소비자들에게 전송되어 질 수 있다. 이런 다수의 이벤트 생성자와 소비자 간의 알림 서비스를 제공하는 구조로 설계하였다. 그림 1 은 이벤트 알림 서비스의 기본적인 구조를 표현한 것이다.



CO: Communication Object

그림 1 이벤트 서비스 기본 구조

이벤트 기반 프로세서/클라이언트(Event-based Process/Client) 는 이벤트를 생성하고 소비하는 프로그램들이며 이들은 CO (Communication Object)를 사용하여 이벤트 통신을 사용한다. CO 에 대한 내용은 3.3 절을 참고하기 바란다.

이벤트 분배 프로세서(Event Dissemination Process) 는 한 호스트 내에 존재하는 다수개의 이벤트 기반 프로세서/클라이언트가 이벤트 분배 과정을 공유할 수 있게 제공되는 서비스이다. 이는 모든 프로세서/클라이언트가 분배 로직을 수행하는 부담을 줄여 전체적인 계산 성능을 높이는 효과가 있다. 또한 동일

호스트 내의 다수 이벤트 기반 프로세서/클라이언트가 이벤트를 수신하는 환경의 경우 네트웍적인 자원 공유가 가능하다는 장점이 있다. 이벤트 분배 프로세서에 대한 자세한 내용은 3.2 절을 참고하기 바란다.

본 논문에서 제시하는 이벤트 기반 시스템에서 이벤트 전송 과정은 다음과 같다.

1. 이벤트 기반 프로세서/클라이언트가 이벤트 생성
2. 이벤트 기반 프로세서/클라이언트가 CO를 통하여 이벤트 분배 프로세서로 전송
3. 이벤트 분배 프로세서는 CO 나 다른 이벤트 분배 프로세서로 이벤트 분배
4. 이벤트 기반 프로세서/클라이언트가 이벤트 수신이벤트 기반

그림 1에서 호스트 A 내에 존재하는 이벤트 분배 프로세서와 다수의 이벤트 기반 프로세서/클라이언트는 내부 네트웍 통신을 하고 있다. 이는 특정 호스트에 이벤트 분배 프로세서가 존재하지 않을 경우 다른 호스트에 존재하는 이벤트 기반 프로세서를 통하여 이벤트 알림 서비스를 이용할 수 있게 하기 위해서이다.

3.2 이벤트 분배 프로세서

이벤트 분배 프로세서는 한 호스트에서 여러 응용 프로세서들이 생성하는 이벤트를 다수의 이벤트 소비자들에게 전송하기 위한 공유 프로세서로 비동기적 이벤트 분배, 주제 기반 이벤트 분배, 그리고 내용 기반 메시지 필터링을 제공하고 있다. 이벤트 분배 프로세서는 그림 2 와 같다.

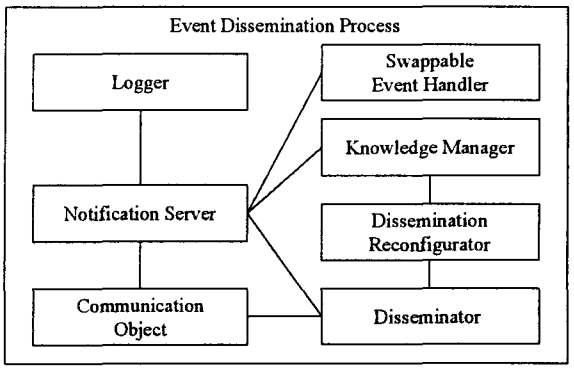


그림 2 이벤트 분배 프로세서 개념도

이벤트 분배 프로세서에서 비동기적 이벤트 분배란 이벤트 생성자와 이벤트 소비자 간의 프로그램 상태가 상호 영향을 주지 않음을 의미한다. 즉 생성자 측에서 소비자에게 이벤트을 전송한 후 소비자 측에서 수신 여부와 관계없이 다른 이벤트를 전송할 수 있다. 이는 본 논문에서 제시하는 이벤트 분배 프로세서가 이벤트 생성자가 생성한 이벤트 전송을 대행해 주기 때문에 비동기적 이벤트 분배가 가능하다.

이벤트 데이터에 대한 기본적인 분배 규칙은 프로그램에서 정의한 주제를 기반으로 하고 있다. 따라서 이벤트를 사용하는 프로그램에서는 주제를 이벤트의 사용 목적에 따라 정의하면 된다. 주체 기반 이벤트 분배 방식은 이벤트를 사용하는 프로세서/클라이언트의 변경에 유연하게 대처할 수 있을 뿐만 아니라 이벤트 생성자가 이벤트 소비자들을 알 필요가 없다는 장점이 있다.

주제 기반 이벤트 분배를 위한 컴포넌트는 Disseminator, Dissemination Reconfigurator, 그리고 Knowledge Manager 가 있다. 이벤트 데이터에 대한 분배는 Disseminator 에 의해 수행된다. 즉 Disseminator 는 이벤트 데이터에 정의된 주제를 보고 전송 대상을 결정하며 결정된 대상으로 이벤트 데이터를 분배한다. 이때 사용되는 주제에 따른 분배 규칙 정보는

Knowledge Manager에 의하여 관리된다. Knowledge Manager의 정보는 관리자 UI를 통해서나 실행 시간에 분석된 정보를 바탕으로 변경된다. 분배 규칙 정보의 변경은 Dissemination Reconfigurator 에 의해 수행된다.

내용 기반 메시지 필터링은 수없이 많이 전송되는 이벤트 중에서 유효한 데이터 구분해 내기 위해 정의되고 사용된다. 이벤트 핸들러는 수신된 이벤트 데이터에서 의미 있는 메시지인지를 판단하기 위한 컴포넌트이다. 현재 구현된 이벤트 핸들러는 특정 키워드에 대한 매칭을 검사하는 방식을 사용하고 있다. 그리고 이벤트 핸들러는 더욱 복잡한 필터링 로직으로 확장 가능하게 swappable 컴포넌트로 구현되었다.

3.3 이벤트 통신 기반 구조

이벤트 통신 기반 구조는 이벤트 데이터를 네트워크를 통하여 전송하기 위하여 개발된 기본 API(Application Programming Interface)이다. 이벤트 기반 프로세서/클라이언트와 이벤트 분배 프로세서는 이 API를 사용하여 상호 통신을 한다. 그림 3 은 이벤트 통신 기반 구조에 대해 보여주고 있으며 다음과 같은 객체로 구성되어 있다.

Communication Process: 이벤트 통신을 위한 프로그램으로 이벤트 기반 프로세서/클라이언트나 이벤트 분배 프로세서를 의미하는 것이다.

Communication Manager: 이벤트 통신을 위한 Communication Object 의 생성, 소멸, 그리고 관리를 위한 객체이다.

Configurator: XML 과 같이 미리 정의한 설정 정보를 이용하여 Communication Object와 DocFormat Object를 생성할 수 있게 하기 위한 객체이다.

Communication Object: 통일된 네트워크 데이터 전송 구현을 환경을 제공하기 위하여 정의된 객체이다.

DocFormat Object: 다양하게 정의되는 이벤트 표현 형식을 변환하기 위하여 사용하는 객체이다. 예를 들어, 데이터를 송신하는 측은 XML 형식을 사용하는 반면 수신하는 측에서는 Payload를 사용하는 경우 표현 형식 변환이 필요하다.

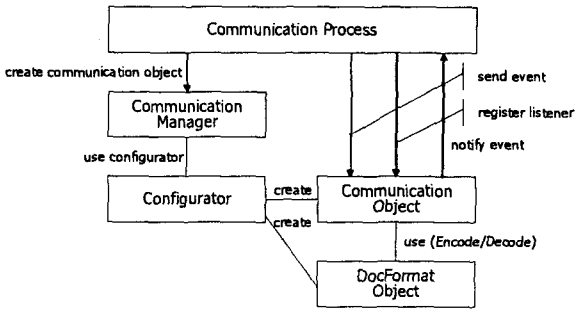


그림 3 이벤트 통신 기반 구조 개념도

본 논문에서 제시한 이벤트 통신 기반 구조에서 Communication Object는 TCP 방식과 UDP 방식을 지원하고 있다. 한편 DocFormat Object 는 XML 표현 방식과 이벤트 데이터를 Payload로 구분한 문자열 표현 방식, 그리고 객체 직렬화 기법을 이용한 표현 방식을 지원하고 있다.

3.4 응용 프로그램 적용 이슈

본 절에서는 현재 구현된 이벤트 알림 서비스를 이용하여 응용 프로그램을 작성할 때의 이슈에 대하여 살펴본다. 이때 고려 사항으로는 성능, 확장성, 그리고 구현 시스템 특성 등을 들 수 있다.

대부분의 응용 프로그램 개발자들은 성능 측면을 매우 중요하게 생각한다. 특히 실시간 시스템(Real-time system) 같이 시간 제약을 받는 시스템의 경우 신속한 이벤트 처리에 대한 요구는 당연하다. 이런 실시간 시스템에 적용하기 위한 이벤트 분배 서비스는 XML 파싱과 같은 많은 계산 시간을 요하는 구현은

피해야 한다. 또한 데이터 손실을 허용하는 경우에는 UDP 네트워크 프로토콜을 사용하는 것이 좋다. TCP 네트워크 프로토콜은 최대 처리량 이상 데이터에 대한 전송 지연이 커질 수 있다.

기존 응용 프로그램에서 사용되는 방식이 존재하거나 새로운 방식의 추가될 가능성이 큰 경우에는 확장성을 측면이 중요하다. 시스템/네트워크 관리 시스템과 같은 솔루션은 그 기종과 구현이 다양할 뿐만 아니라 기존 구현과 다른 새로운 시스템의 추가 가능성이 크다. 확장성 좋은 구현을 위해서는 표준화된 데이터 형식의 정의와 네트워크 프로토콜의 사용이 필수적이다.

앞에서 언급한 두 가지 경우와는 달리 구현 시스템 특성에 따라 구현 방법이 결정될 수 있다. 예를 들어, 구현된 시스템이 Web Service 에 적용될 이벤트 알림 서비스는 XML 기반의 데이터 형식을 다룰 수 있어야 한다.

4. 성능 측정 및 분석

4.1 성능 측정 환경

본 논문에서 제시한 이벤트 알림 서비스에 대한 테스트 환경은 Intel PIII 800 MHz CPU, 256 MB 메모리를 사용하는 시스템이며 운영체제는 RedHat Linux 9.0 이다. 이벤트 분배 프로세서와 이벤트 생성 프로그램과 이벤트 수신 프로그램은 JDK 1.3 기반에서 개발하였다. 이벤트 생성 프로그램에서는 초당 20 개의 이벤트를 발생하는 Sender 수를 증가하는 방식을 사용하였으며 각 Sender 는 1000 개의 이벤트 발생시켰다.

4.2 성능 척도

이벤트 알림 서비스의 처리 성능을 다음 두 가지 측면에서 분석하고자 한다.

초당 전송량(Throughput): 이벤트 발생 클라이언트 수에 따라 초당 처리된 이벤트의 수를 계산한 것이다. 전송 시작 시간을 TS_0 라 하고 n 개의 이벤트가 전송된 시간을 TE_n 이라 하면 초당 전송량은 $\frac{TE_n - TS_0}{n} * 1000$ 이다.

이벤트당 평균 전송 시간: 이벤트 데이터가 이벤트 생성자에서 소비자에게 전송되는 시간의 평균값이다. i 번째의 이벤트 전송 시작 시간을 TS_i 라 하고 이벤트 전송 완료 시간을 TE_i 라 하면 n 개의 이벤트 전송 시간은 $\frac{\sum_{i=1}^{n}(TEi - TSi)}{n}$ 이다.

4.3 측정 결과

성능 측정은 이벤트 데이터를 XML 방식을 사용하는 경우, 객체 직렬화를 사용하는 경우, 그리고 Payload를 사용하는 경우로 구분하여 테스트 하였다. 그리고 각 이벤트 데이터 표현에 대하여 TCP 프로토콜을 사용하는 경우와 UDP 프로토콜을 사용하는 경우를 나누어 테스트하였다.

그림 4는 XML 데이터 형식을 사용하는 TCP 통신 프로토콜과 UDP 통신 프로토콜의 전송량과 평균 전송 시간을 나타낸 것이다. 실험 결과 초당 최대 전송량은 TCP 통신 프로토콜의 경우 197 Event/sec이며 UDP 통신 프로토콜은 187 Event/sec이다. 그러나 초당 최대 전송량을 보이는 전송자 수 10 인 부분에서 평균 전송 시간이 급격하게 증가하고 있다. 이 경우 오랜 시간 전송을 고려하면 전송자 수 9 인 180 Event/sec를 초당 최대 전송량으로 볼 수 있다. 이 결과는 객체 직렬화 방식이나 Payload를 사용하는 방식과 비해 가장 낮게 나타났다. 이는 XML 방식이 엔코딩/데코딩을 하는데 많은 시간을 사용하기 때문이다. 한편 평균 전송 시간은 TCP 통신 프로

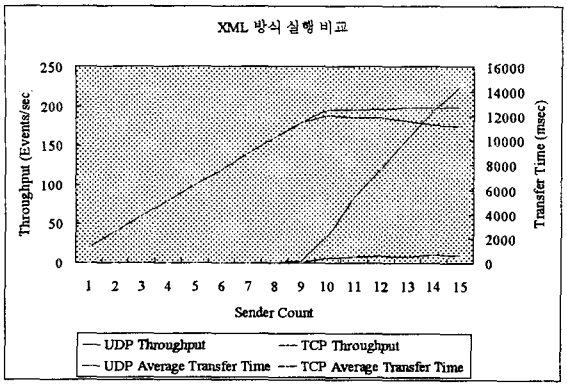


그림 4 XML 방식의 성능 측정 결과

토콜이 UDP 통신 프로토콜보다 크게 나왔으며, 특히 최대 전송량 이후에는 TCP 통신 프로토콜이 급격히 커지는 것을 볼 수 있다. UDP 통신 프로토콜의 경우 데이터 손실이 발생되었다.

그림 5 의 경우 객체 직렬화 기법을 사용할 때의 성능을 측정한 것이다. 이 테스트의 경우 TCP 프로토콜은 자바 언어에서 제공하는 API를 사용하였다. UDP 프로토콜의 경우 객체 직렬화를 제공하기 않기 때문에 테스트에서 제외하였다. 실험 결과 평균 전송 시간을 고려했을 때의 초당 최대 전송량은 전송자 수 35 때의 670 Event/sec 로 볼 수 있다.

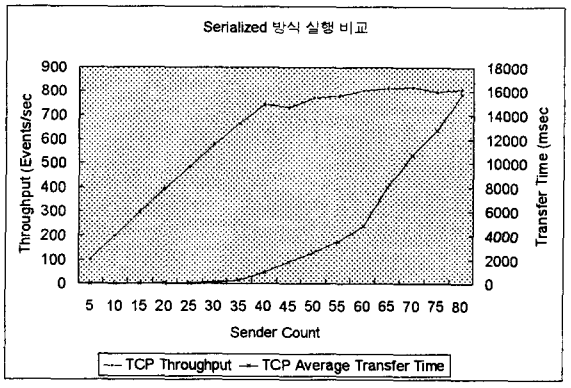


그림 5 객체 직렬화 방식의 성능 측정 결과

그림 6 의 경우 이벤트 데이터를 Payload를 사용하여 표현한 경우 TCP 통신 프로토콜과 UDP 통신 프로토콜의 전송량과 평균 전송 시간을 나타낸 것이다. 실험 결과 평균 전송 시간을 고려했을 때의 초당 최대 전송량은 전송자 수 70 때의 1383 Event/sec 로 볼 수 있다. 이 실험의 결과도 XML 데이터 형식을 사용할 때와 같이 최대 전송량 이후에 TCP 통신 프로토콜의 평균 전송 시간이 증가하는 것을 확인할 수 있으며, UDP 통신 프로토콜에서는 데이터 손실을 확인할 수 있다.

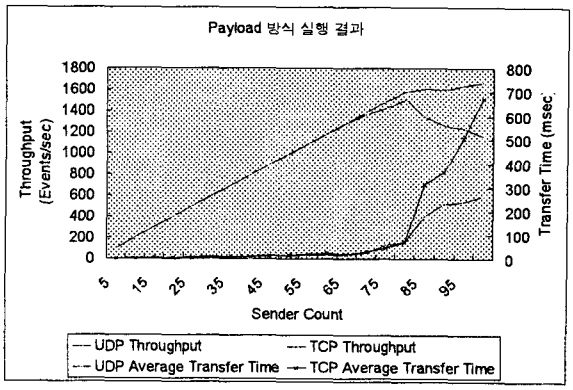


그림 6 Payload 방식의 측정 결과

5. 결론

본 논문에서는 다양한 응용 프로그램의 요구를 만족시킬 수 있는 이벤트 알림 서비스를 제안하고 있다. 이 서비스는 이벤트 분배 프로세서와 이벤트 통신 인프라스트럭처로 구성되어 있다. 이벤트 분배 프로세서는 한 호스트에서 여러 응용 프로세서들이 생성하는 이벤트를 분배하기 위한 공유 프로세서로 비동기적 이벤트 분배, 주제 기반 이벤트 분배, 그리고 내용 기반 메시지 필터링을 제공하고 있다. 이벤트 통신 인프라스트럭처는 다양한 메시지 형식 전송과 다양한 통신 프로토콜을 지원하기 위하여 Configurable 컴포넌트로 구현되어 적응성(Flexibility)이 좋다.

본 논문에서 제시된 이벤트 서비스에 대한 성능 측정 결과 Payload를 사용한 데이터 표현 방식이 XML 데이터 표현 방식보다 8 배

빠른 것으로 나타났다. 네트워크 전송 프로토콜에 대한 성능 측정 결과 TCP 방식이 UDP 방식보다는 더 많은 처리량을 보인다. 그러나 TCP 방식의 전송 시간이 UDP 방식보다 급격히 증가하는 단점이 있다.

참고문헌

[1] M. D. Spiteri, "An Architecture for the Notification, Storage and Retrieval of Events", PhD Thesis, University of Cambridge, January 2000

[2] Cugola Gianpaolo, Di Nitto Elisabetta and Fuggetta Alfonso, "The JEDI Event-based Infrastructure and its Application to the Development of the OPSS WFMS", IEEE Transactions of Software Engineering, 2001

[3] Dong Zhou, Karsten Schwan, Greg Eisenhauer and Yuan Chen, "JECho:Interactive High Performance Computing with Java Event Channels", 3rd International Workshop on Java for Parallel and Distribute Computing, 2001.

[4] A. Carzaniga, "Architectures for an Event Notification Service Scalable to Wide-area Networks", PhD Thesis, Politecnico di Milano, December 1998

[5] A. Rowstron, A-M. Kermarrec, M. Castro and P. Druschel, "SCRIBE: The design of a large-scale event notification infrastructure", NGC2001, UCL, London, November, 2001

[6] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz and J. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination", NOSSDAV, 2001

[7] R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward, "Gryphon: An Information Flow Based Approach to Message Brokering", International Symposium on Software Reliability Engineering, 1998

[8] Object Management Group, "Notification Service", August 2002,

[9] Luis Felipe Cabrera, Michael B. Jones, and Marvin Theimer, "Herald: Achieving a Global Event Notification Service.", Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII), May 2001

[10] Bill Segall and David Arnold, "Elvin has left the building: A publish/subscribe notification service with quenching", Proceedings AUUG97, September 1997

[11] B. Krishnamurthy and D. S. Rosenblum, "Yeast: A General Purpose Event-Action System", IEEE Transactions on Software Engineering, October 1995

● 저자소개 ●

한영태

1998 건국대학교 컴퓨터.정보통신공학과 석사

현재 건국대학교 컴퓨터.정보통신공학과 박사과정

관심분야: 분산 시스템, 멀티미디어 시스템, 분산 시스템 관리, 이벤트 기반 시스템, 시스템 성능 분석, 웹 솔루션

E-Mail : headline@konkuk.ac.kr

민덕기

1986 고려대학교 공과대학 산업공학 학사

1991 미시건 주립 대학교 컴퓨터공학 석사

1995 미시건 주립 대학교 컴퓨터공학 박사

1995~현재 건국대학교 정보통신대학 컴퓨터공학부 교수

2000~현재 소프트웨어 컴포넌트 포럼 기술분과 위원장

관심분야: 분산 및 병렬 시스템, 분산 객체 및 컴포넌트 기술, 미들웨어, 소프트웨어 시스템 아키텍쳐, 시스템 성능 분석, 웹 기반 분산 컴퓨팅, 웹 서비스

E-Mail : dkmin@konkuk.ac.kr