전자메일 서비스를 이용한 이벤트 알림 시스템의 설계 및 구현

이승민*, 김명일*, 정희석*, 김재성*, 이상민*
*한국과학기술정보연구원 슈퍼컴퓨팅본부
e-mail:smlee76@kisti.re.kr

Design and Implementation of Event Notification System Using E-Mail Service

Seung-Min Lee*, Myoung-Il Kim*, Hee-Seok Jeong*, Jae-Sung Kim*, Sang-Min Lee*
*Supercomputing Center, Korea Institute of Science and Technology Information

요 약

응용 프로그램을 실행할 때 프로세스간의 상호 연동이 필요한 경우 통신을 하게 된다. 프로세스간의 통신에서 서비스를 요청하는 프로세스의 상태에 따라 동기(Blocking) 모드와 비동기(Non-blocking) 모드로 나누어진다. 동기 모드는 서비스를 요청받은 프로세스의 작업이 완료할 때까지 서비스를 요청한 프로세스의 상태가 멈추는 방식이고, 비동기 모드는 다른 프로세스에 서비스를 요청한 후 자신은 다른 유용한 작업을 계속 수행할 수 있는 방식으로 통신 상대가 여럿이거나 서비스 수행시간이 긴 작업의 경우 비동기 모드를 사용한다. 프로세스 간의 비동기적인 작업 수행이 이벤트 기반으로 이루어지도록 지원하는 이벤트 알림 시스템은 전자메일(E-mail) 서비스와 유사한 특징이 있다. 전자메일 서비스는 컴퓨터 통신망을 통해 편지를 주고 받을 수 있는 서비스 및 해당 편지를 일컫는 용어로, 본 논문에서는 이전에 제시된 이벤트 알림 시스템의 구성과 잘 알려진 전자메일 서비스의 특징을 파악하여 전자메일 서비스를 이용한 이벤트 알림 시스템의 설계 및 구현 방법을 제시한다.

1. 서 론

응용 프로그램을 실행할 때 프로세스간의 상호 연동이 필요한 경우 통신을 하게 된다. 프로세스 간의 통신에서 서비스를 요청하는 프로세스의 상태에 따라 동기 모드와 비동기 모드로 나누어진다. 동기 모드는 서비스를 요청받은 프로세스의 상태가 멈추는 방식이고, 비동기 모드는 다른 프로세스에 서비스를 요청한 후 자신은 다른 유용한 작업을 계속 수행하는 방식으로 통신 상대가 여럿이거나 서비스 수행시간이 긴 작업의 경우 비동기 모드를 사용한다. 연결을 유지하는 동기 모드와는 달리 비동기 모드의 프로세스간 통신에 있어서는 두 가지 방식이 가능하다. 첫째는 폴링(polling) 방식으로 클라이언트-서버 환경에서 클라이언트가 서버에 주기적으로 상태를 확인하는 방법이다. 이 방식은 서버에 주기적으로 요청하는 횟수의 간격이 문제가 된다. 두 번째로는 이벤트를 처리할 수 있는 시스템이 존재하여 서버와 클라이언트의 비동기적인 처리를 위임하는 방식이다.

프로세스 간의 비동기적인 작업 수행이 이벤트 기반으로 이루어지도록 지원하는 이벤트 알림 시스템은 전자메일 서비스와 유사한 특징이 있다. 전자메일(E-mail) 서비스는 컴퓨터 통신망을 통해 편지를 주고 받을 수 있는 서비스 및 해당 편지를 일컫는 용어로, 본 논문에서는 이전에 제시된 이벤트 알림 시스템의 구성과 잘 알려진 전자메일 서비스의 특징을 파악하여 전자메일 서비스를 이용한 이벤트 알림 시스템의 설계 및 구현 방법을 제시한다.

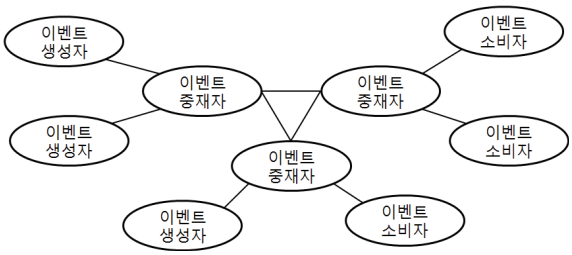
본 논문의 구성은 다음과 같다. 2절에서는 기존에 제시된 이벤트 알림 시스템과 전자메일 서비스를 소개한다. 3절에서는 전자메일 서비스의 특징과 구성 요소를 통해 이벤트 알림 시스템의 구성 요소들과의 대응 관계와 전자메일 서비스를 이용한 이벤트 알림 시스템을 제시한다. 4절에서는 제시한 이벤트 알림 시스템의 구현과 관련 이슈에 대하여 알아본다. 그리고 마지막 5절에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

본 절에서는 이벤트 알림 서비스에 대한 기존 연구들을 통해 이벤트 알림 시스템에 필요한 구성 요소를 확인하고, 전자메일 서비스의 특징을 알아본다.

이벤트 시스템은 아키텍쳐 관점에서 클라이언트-서버 모델과 P2P(Peer-to-Peer) 모델로 나누어 구성된다. 클라이언트-서버 모델에서는 이벤트 서버가 이벤트를 수신하고 저장하고 분배하는 역할을 수행하며 이벤트 클라이언트는 이벤트를 생성하여 이벤트 서버에 전송하거나 이벤트를 소비하는 역할을 수행한다. P2P 모델에서는 모든 노드가 동일한

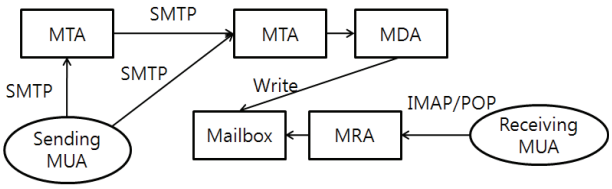
기능을 하게 구현되어 있으나, 두 경우 모두 이벤트 생성자와 이벤트 소비자의 효율적인 연결을 위하여 이벤트 중재자를 두고 있다.



(그림 1) 이벤트 알림 서비스 구조도

(그림 1)은 앞서 설명한 이벤트 알림 서비스의 개념도를 나타내고 있다. 이벤트 시스템의 구성 요소 외에도 이벤트 시스템은 알림을 위한 이벤트 정보의 정의와 표현에 따라 튜플 기반, 레코드 기반, 객체 기반 모델로 나눈다. 튜플 기반 모델은 이벤트 정보를 문자열의 셋으로 정의하며, 레코드 기반 모델은 이벤트 정보를 이름과 값을 가지는 정형화된 필드의 집합으로 정의하고 있으며, 객체 기반 모델은 레코드 필드와 더불어 메소드의 집합으로 이벤트를 정의하고 있다.[2]

전자메일 서비스는 컴퓨터 통신망을 통해 편지를 주고받을 수 있는 시스템과 해당 편지를 일컫는 용어로 전자메일 서비스를 이용하여 메일을 전송하고, 전송한 메일을 사용자가 확인하는 방식이다.



(그림 2) 전자메일 전송 메커니즘

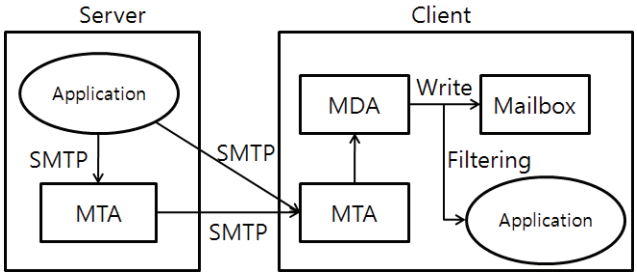
(그림 2)는 전자메일 서비스를 이용하여 메일을 전송하고 사용자가 확인하는 메커니즘을 나타내고 있다. (그림 2)에서 볼 수 있듯이 전자메일 서비스는 MTA, MDA, MRA, MUA로 구성되어 있다.

- MUA(Mail User Agent)
  이메일 클라이언트로 사용자가 메일을 읽고 쓰는 기본적인 기능과 메일을 관리하는 기능을 포함하는 프로그램을 지칭한다.
- MTA(Mail Transfer Agent)
  MUA와 다른 MTA로부터 메일을 받거나 다른 MTA로 메일을 전달하는 기능을 수행한다. MTA가 메일을 주고받는 통신은 SMTP(Simple Mail Transfer Protocol)을 기반으로 한다.
- MDA(Mail Delivery Agent)
  MTA가 받아들인 메일을 사용자 환경에 맞도록 저장하는 역할을 수행한다.
- MRA(Mail Retrival Agent)
  사용자가 요청하는 메일을 전달하는 기능을 수행한다. MRA와 MUA의 통신은 IMAP(Internet Message Access Protocol)/POP(Post Office Protocol)을 기반으로 한다.

3. 시스템 설계

본 절에서는 전자메일 서비스의 특징을 통하여 이벤트 알림 시스템을 구성하기 위한 모듈과 이를 활용한 이벤트 알림 시스템의 설계를 제시한다.

이벤트 알림 시스템의 기본 구성 요소 중 이벤트 소비자는 관심있는 이벤트의 주제나 이벤트의 패턴을 등록하고 발생된 이벤트에서 관심 있는 내용을 비동기적으로 통지받는 역할을 한다. 이벤트 생성자는 이벤트를 이벤트 중재자에게 통지하는 역할을 수행한다. 그리고, 이벤트 중재자는 상호 연결하여 이벤트 생성자와 이벤트 소비자를 연결하는 기능을 수행한다. 상호 통신을 원하는 프로세스는 이벤트 생성자와 이벤트 소비자의 역할에 해당하며, 전자메일 서비스의 구성 요소 중 MTA, MDA는 이벤트 중재자의 역할을 담당하게 된다.

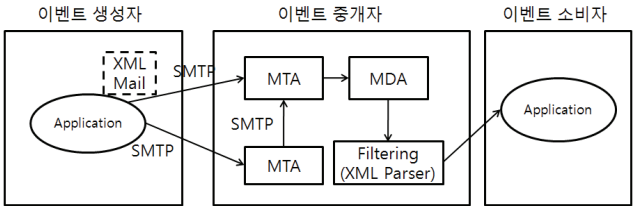


(그림 3) 메일 시스템을 통한 이벤트 송신 및 수신

(그림 3)은 서버 측의 응용 프로그램에서 이벤트를 전자메일 서비스에 통지하고 전자메일 서비스가 클라이언트의 응용 프로그램에 이벤트를 전달하는 것으로 전자메일 서비스를 이용한 이벤트 알림 시스템의 흐름도를 나타낸 것이다. 클라이언트에서 서버측으로 이벤트 정보를 통하여 작업을 요청하는 방식도 (그림 3)과 동일하게 이루어진다. 즉, 클라이언트의 응용 프로그램에서 서버측으로 전자메일을 통해 원하는 작업을 요청하면 서버측의 MTA가 전자메일을 확인하고 MDA가 전자메일을 메일함(Mailbox)에 저장을 하기 위해 관여하게 된다. 이 때, 전자메일의 필터링 기능을 이용하여 이벤트 정보를 구분하고, 이벤트를 수행할 응용 프로그램에 해당 내용을 전달하게 된다. 이 때 클라이언트의 응용 프로그램에서 서버측으로 이벤트 정보를 요청할 때 클라이언트의 MTA를 이용하거나 또는 서버측의 MTA에 직접적으로 연결이 가능하다. 서버측의 프로세스에서 클라이언트의 프로세스에 결과를 전송하는 방식도 MTA를 통해 통신이 이루어지고 모니터링 및 필터링은 MDA에서 이루어지게 된다. MDA는 이벤트 소비자에 해당하는 응용 프로그램과 같은 시스템에 위치할 경우

시스템 호출 등을 통하여 실행할 수 있으며 원격의 경우 원격 절차 호출(RPC: Remote Procedure Call)과 같은 통신이 가능한 경우에 한해 구성이 가능하다.

응용 프로그램 간의 통신을 위한 이벤트 정보는 전자메일 서비스를 통하여 이루어지므로 이벤트 정보의 표현은 MIME(Multipurpose Internet Mail Extensions)을 이용하게 된다. MIME은 전자우편을 위한 인터넷 표준 포맷으로 문자열 뿐 아니라 바이너리 파일들도 전송이 가능하다. MIME 형식으로 전송되는 이벤트 정보 중 필터링에 필요한 패턴과 요청 정보는 전자메일 본문 또는 헤더를 이용하여 MDA가 처리할 수 있도록 한다.



(그림 4) 전자메일 서비스를 이용한 이벤트 알림 시스템

(그림 4)는 전자메일 서비스를 이용한 이벤트 알림 시스템의 설계로 구성 요소의 시스템적인 위치가 아닌 이벤트 알림 시스템의 기본 구성 요소에 대응하는 전자메일 서비스의 모듈을 배치한 것이다.

4. 시스템 구현

본 절에서는 클라이언트-서버 환경에서 제시한 이벤트 알림 서비스를 이용하여 응용 프로그램이 통신하는 방법의 구현을 살펴본다.

서버 시스템은 리눅스 운영체제로 CentOS 5.4 버전을 이용하였고, 클라이언트 시스템은 윈도우즈 운영체제에 Cygwin[4]을 설치하였다. 전자메일 서비스를 위한 프로그램은 서버와 클라이언트에 모두 Exim[5]을 설치하였다. 예제로는 클라이언트에서 전송한 문자열을 되돌리는 간단한 echo 프로그램을 사용하였다.

```
#/bin/sh
echo $1 > test.log
./run_post $1
```

(그림 5) 서버 측 실행 프로그램 (Echo)

(그림 5)는 서버측에서 실행하는 간단한 쉘 스크립트 예제이다. 서버에서 클라이언트의 문자열을 잘 받았는지 확인하기 위해 test.log 파일을 생성하고 해당 문자열을 저장한다. 그 이후에 run_post 프로그램을 실행하여 클라이언트로부터 받은 문자열을 다시 되돌려주는 메일을 발송하는 역할을 수행한다.

```
$ cat .forward
# Exim filter
if $h_subject: contains "event_notification" then
  pipe "/usr/local/bin/save_echo.sh $message_body &"
  seen finish
endif
```

(그림 6) Exim 필터 예제

(그림 6)은 exim 프로그램에서 필터링 기능을 사용하는 예제를 나타낸 것이다. 즉, exim에서 스팸 메일을 처리하는 모듈을 이용하여 메일 헤더에 “event_notification”이라는 문자열을 체크한 후 패턴이 존재할 경우 메일 본문에 포함되어 있는 XML 형식의 데이터를 확인하여 문자열을 추출하게 된다. 메일을 받을 때 자동으로 실행되어 필터링 기능을 수행할 수 설정하기 위해서는 홈디렉토리의 .forward 파일에 저장하고, '# Exim filter' 문구로 시작해야 한다.

```
<?xml version="1.0" encoding="euc-kr" ?>
<request_form>
    <userinfo>
        <id>${USER_ID}</id>
        <passwd>${PASSWD}</passwd>
    </userinfo>
    <data>
        <job_id>${JOB_INDEX}</job_id>
        <value>${VALUE}</value>
    </data>
</request_form>
```

(그림 7) 데이터 전송을 위한 XML 템플릿 예제

(그림 7)은 메일 본문에 포함될 XML 형식의 예제이다. 클라이언트에서 서버에 작업을 요청할 경우 전자메일 서비스에서는 권한에 대한 인증을 사용자의 아이디나 특정 키워드의 유무로 확인해야 한다. 본 예제에서는 인증을 위해 서버에 계정 정보(아이디, 비밀번호)를 메일 본문에 전달할 수 있도록 템플릿에 포함하였다. 서버에 전송하는 계정 정보는 서버가 작업을 종료한 후에 클라이언트에 보내는 메일 본문에 다시금 포함되어 클라이언트의 응용 프로그램도 해당 내용을 통하여 서버 응용 프로그램에서 생성한 이벤트에 대한 인증을 수행한다. 마지막으로 데이터 영역에서는 여러 작업을 동시에 요청할 경우, 작업의 구분을 위한 필드로 job_id 정보를 추가하였고, 예제 프로그램은 간단한 문자열만 전송하고 받기 때문에 데이터의 value에 문자열을 반영하도록 작성하였다.

5. 결론

응용 프로그램간의 상호 연결을 제공하는 이벤트 알림 시스템을 전자메일 서비스의 기능으로 구현하였다. 이를 통

하여 클라이언트가 서버로부터 작업이 종료될 때까지 연결을 유지하거나 폴링 방식으로 수행 여부를 확인하지 않고 이벤트 알림을 통해 결과 여부를 확인하는 시스템을 구현하였다. 향후에는 작업 시간이 오래 소요되는 경우 다른 통신 방법과 비교하였을 때 제시한 시스템의 효율성을 검토하고 오류 시의 복구 방안과 보안 관련한 이슈를 연구할 계획이다.

참고문헌

[1] Antonio Carzaniga, David S. Rosenblum, Alexander L. Wolf “Design and Evaluation of a Wide-Area Event Notification Service″ Volumn 19, Issue 3, ACM Transactions on Computer Systems(TOCS), 2001

[2] 한영태, 민덕기 “이벤트 알림 서비스의 구조와 성능분석” V.10 No. 3, 한국컴퓨터정보학회논문지, 2005

[3] 인터넷 메일 컨소시엄, http://www.imc.org

[4] Cygwin, http://www.cygwin.org

[5] Exim 홈페이지, http://www.exim.org