JPE : AJAX 기반의 비동기 통신을 지원하는 Java Push Engine

박종은* · 권오진* · 이홍창** · 이명준***

JPE : Java Push Engine Supporting Asynchronous Communication Based on AJAX

Jong-Eun Park* · O-Jin Kwon* · Hong-Chang Lee** · Myung-Joon Lee***

이 논문은 현대중공업 지원에 의한 울산대학교 전기공학부 일류화연구비에 의하여 연구되었음

요 약

정보를 공유하기 위하여 널리 이용되는 웹은 클라이언트-서버 모델을 사용한다. 클라이언트-서버 모델은 클라이언트의 명시적인 요청을 통하여 서버가 응답하는 방식으로서 오늘날의 급변하는 인터넷 정보를 효과적으로 제공하기에는 많은 어려움이 따른다. 서버 푸시는 클라이언트-서버 모델 기반의 웹에서 클라이언트의 요청이 없더라도 서버가 능동적으로 정보를 제공할 수 있는 통신 기술이다. 이러한 서버 푸시 기술을 구현하기 위하여 다양한 연구가 이루어지고 있지만 푸시 어플리케이션의 효과적인 개발을 지원하는 푸시 엔진의 부재로 많은 어려움이 따르고 있다. 본 논문에서는 인터넷에서 서버 푸시 서비스를 효과적으로 제공하도록 지원하는 Java Push Engine인 JPE의 개발에 대하여 기술한다. JPE는 Epoll을 지원하는 JPE 코어와 비동기 통신을 지원하는 JPE 라이브러리로 구성된다. 그리고 JPE는 다양한 푸시 기능을 정의하고 이를 지원하는 프로그래밍 인터페이스를 제공한다. JPE를 이용하여 개발된 푸시 어플리케이션은 Epoll 기법을 이용하여 클라이언트 연결을 효과적으로 처리하며, Ajax 기반의 비동기 통신을 통하여 다양한 푸시 서비스를 제공한다.

ABSTRACT

The Web is widely used to share information, utilizing the client-server model. In the client-server model, since the server only responds according to explicit requests from the client, the model seems hard to support sharing of massive information rapidly changing in today's Internet. The technology known as Server Push enables the server to actively provide information to clients without explicit requests from the clients using Web pages. Although various studies have been done to realize the Server Push technologies, there are many problems in the development of push application without push engines which support infrastructures for the effective development of push application. In this paper, we develop JPE(Java Push Engine) which presents the effective support for push services over the Internet. JPE is composed of two main components: the JPE Core supporting Epoll and the JPE Library supporting asynchronous communication. In addition, JPE defines various push functions and provides programming interfaces supporting the functions. Push applications developed using JPE effectively manages client connections with Epoll mechanism, providing push services through AJAX-based asynchronous communication.

키워드

서버 푸시, 푸시 엔진. 에이젝스, 롱 폴링, 이폴

Key word

Server Push ,Push Engine, AJAX, Long Polling, Epoll, JPE

* 준회원 : 울산대학교 전기공학부
** 정회원 : 울산대학교 전기공학부
*** 정회원 : 울산대학교 전기공학부 교수 (교신저자, mjlee@ulsan.ac.kr)

접수일자 : 2010. 12. 09
심사완료일자 : 2011. 01. 25

Ⅰ. 서 론

오늘날 인터넷은 많은 사람들이 정보를 검색하거나 공유하기 위한 가장 일반적인 환경으로 널리 사용되고 있다. 인터넷은 HTTP 프로토콜을 기반으로 정보를 제공하는 서버와 정보를 요청하거나 제공받는 클라이언트로 구성된다. 이러한 클라이언트/서버 모델은 사용자들이 요청한 정보를 서버에서 수동적으로 제공해주는 기존의 정보 제공 서비스 형태에는 적합하였으나 서버 상의 정보가 수시로 변하고 빈번하게 새로운 정보가 발생하는 상황에서는 비효율적이다. 특히, 오늘날에는 급변하는 산업, 금융, 뉴스 등의 정보들을 신속하고 정확하게 제공받기 위한 요구가 높아지면서 사용자의 요청이 없이도 능동적으로 필요한 정보를 적시에 제공되는 환경이 필요하다.

서버 푸시[1,2]는 클라이언트의 명시적인 요청이 없이도 서버가 능동적으로 정보를 제공하는 통신 기술이다. 기존의 HTTP 프로토콜은 항상 클라이언트의 요청에 따라 서버가 정보를 제공하는 방식으로 동작한다. 이러한 방식은 많은 클라이언트의 빈번한 요청이 발생하는 경우 서버에 과부하가 걸리기도 하며, 클라이언트의 요청이 없다면 중요한 정보가 변경되더라도 사용자에게 제공되지 않는 상황이 발생하기도 한다. 폴링(Polling)은 클라이언트/서버 모델에서 서버 푸시를 구현하는 대표적인 기술로서 클라이언트가 서버에게 암묵적으로 요청을 계속 전송하면서 서버 상의 정보가 수정되면 정보를 제공받는 방식이다. 폴링은 일반적인 HTTP 기반 어플리케이션에 쉽게 구현이 가능하기 때문에 널리 사용되고 있다. 그러나 폴링은 잦은 요청으로 인한 서버 자원의 낭비와 지속적인 연결에 따른 네트워크 트래픽 문제로 그 효율성이 지적되어 왔다. 최근 AJAX(Asynchronous JavaScript + XML)[3.4]와 같은 비동기 통신 객체의 활용 기법이 등장하면서 폴링의 문제점을 개선하고 효율적인 서버 푸시를 구현을 지원하는 롱 폴링(Long Polling)[5], HTTP 스트리밍(HTTP Streaming)[6,7]과 같은 푸시 기술이 사용되고 있다.

롱 폴링 혹은 HTTP 스트리밍을 기반으로 푸시 어플리케이션 구현을 지원하기 위한 다양한 프레임워크가 등장하고 있다. 대표적으로 Pushlet[8], Adobe 사의 BlazeDS[9.10] 그리고 Weelya사의 APE(AJAX Push Engine)[11]등이 있다. 이러한 프레임워크들은 다양한 기능과 빠른 성능을 표방하고 있지만, 개발자 친화적인 편리한 환경을 제공하지 않거나, 플랫폼 종속적인 실행 문제로 인하여 다양한 형태의 푸시 어플리케이션의 개발을 지원하지 못하고 있는 실정이다.

본 논문에서는 웹에서 서버 푸시 서비스의 손쉬운 개발을 지원하는 JPE(Java Push Engine)에 대하여 기술한다. JPE는 Epoll[12]을 지원하는 푸시 서버의 개발을 위한 JPE 코어와 Ajax 기반의 비동기 통신을 수행하는 푸시 클라이언트를 위한 JPE 라이브러리로 구성된다. JPE 코어는 자바 NIO(Non-blocking I/O)를 이용하여 Epoll을 지원하면서 동시에 플랫폼 독립적으로 동작한다. 그리고 정보 푸시를 위한 채널이나 사용자 정보 등을 편리하게 관리하기 위한 인프라를 지원한다. JPE 라이브러리는 AJAX를 기반으로 푸시 서버와 연결되며 서버의 푸시 서비스를 활용하기 위한 다양한 프로그래밍 인터페이스를 제공하여 효과적인 푸시 클라이언트 개발을 지원한다. JPE 기반의 푸시 어플리케이션은 Epoll을 기반으로 많은 클라이언트 연결을 효율적으로 관리할 수 있으며, Ajax 기반의 비동기 통신을 지원하여 푸시 서비스를 제공할 수 있다. 또한, 플랫폼 독립적으로 동작하기 때문에 다양한 분야에 쉽게 응용될 수 있다.

본 논문의 구성은 다음과 같다. 서론에 이어 2장에서는 서버 푸시를 구현하는 기술과 그 기술을 이용하여 기존에 출시되어 있는 푸시 엔진에 대해서 다룬다. 3장에서는 JPE의 필요성과 JPE 코어와 JPE 라이브러리의 설계와 개발에 대해 기술하며, 효과적인 통신을 하기 위하여 정의한 프로토콜에 대해 다룬다. 또한 완성된 JPE를 이용하여 푸시 웹 어플리케이션을 구축할 수 있는 사용 방법과 이를 활용한 프로토타입을 제시하고 다른 시스템과의 기능을 비교한다. 마지막으로 4장에서는 결론을 다룬다.

Ⅱ. 관련 연구

1. 서버 푸시

서버 푸시는 클라이언트의 명시적인 요청이 없어도 서버가 정보를 능동적으로 제공하는 통신 기술이다. 표 1은 인터넷에서 서버 푸시 기술을 구현하기 위한 대표적

인 기법들을 보여준다.

표 1. 주요 서버 푸시 기법
Table 1. Major Server Push Techniques

기법	특징
롱 폴링	· AJAX 등 비동기 프레임워크에서 사용 · 서버는 클라이언트의 요청을 받으면 연결을 유지하고 정보가 변경되는 이벤트가 발생하면 응답 후 HTTP 트랜잭션 종료 · 클라이언트는 응답을 받으면 재요청
HTTP 스트리밍	· 실시간 스트리밍처럼 서버와 클라이언트 간의 연결을 유지하며 정보가 변경되는 이벤트가 발생하면 HTTP Chunked[13] 방식으로 데이터를 전송 · 구현과 예외처리가 어려우며 호환성이 떨어짐

2. 서버 푸시를 위한 프레임워크

서버 푸시의 필요성이 부각되면서 서버 푸시 기반의 어플리케이션을 효과적으로 개발하기 위한 프레임워크가 등장하였다.

Pushlets은 자바 애플릿과 같은 별도의 플러그인 없이 서버의 자바 객체에서 클라이언트의 DHTML(Dynamic HTML)으로 정보를 제공할 수 있는 서블릿 기반의 푸시 프레임워크이다. Adobe BlazeDS는 Flex[14]나 AIR[15]와 같은 프레임워크 기반의 클라이언트와 통신하며 정보를 제공하는 기능을 제공하는 플랫폼이다. Pushlets과 BlazeDS는 HTTP 스트리밍 기반으로 서버와 클라이언트의 비동기 통신을 지원한다. APE는 AJAX를 기반으로 실시간 푸시 서비스를 제공하는 오픈 소스 푸시 엔진으로서 APE 서버와 APE 클라이언트로 구성된다. APE 서버는 Epoll 방식의 Comet[16,17] 서버로서 동작하며 APE 클라이언트와 비동기 통신을 수행하며 효과적인 정보 푸시 기능을 제공한다. Epoll은 이벤트 통지기반의 입출력 처리 메카니즘으로 엣지-트리거[18]와 레벨-트리거[19] 인터페이스를 지원하여 빠르게 다수의 클라이언트를 처리한다. 표 2는 APE의 주요 특징을 보여준다.

표 2. APE의 주요 특징
Table 2. Major Features of APE

APE 서버	APE 클라이언트
· C 언어로 작성 · 배포자/구독자 모델 기반 · 리눅스, BSD & 맥OS 지원 · E-poll Driven 이용 · 완전 비동기 · 서버-사이드 자바스크립트 작성 기법 지원 · 벌트-인 메시지 큐잉 시스템 · 확장성 있는 서버 구현 지원	· Mootools[20] 기반 · 클라이언트로부터 Socket API 사용 가능 · 다양한 브라우저 지원 · 플러그인을 위한 프레임워크 지원 · 세션 관리& 멀티 채널 지원 · 전송 메소드 지원

Ⅲ. 본 문

1. 자바 기반의 푸시 프레임워크 필요성

오픈소스로 개발된 APE는 효율적인 클라이언트 연결을 처리하기 위하여 Epoll 기반으로 동작한다. Epoll은 리눅스 등의 특정 플랫폼에서만 동작하기 때문에 APE는 윈도우즈를 비롯한 다양한 플랫폼 기반의 푸시 어플리케이션 개발을 지원하지 못하고 있다. 따라서 Epoll의 효율적인 클라이언트 연결 처리 기법과 다양한 플랫폼을 지원할 수 있는 푸시 프레임워크가 필요하다.

자바 NIO의 Selector 클래스는 JVM(Java Virtual Machine) 버전 6.0과 리눅스 커널 버전 2.6 이상에서 Epoll 방식을 지원하며, 또한 Reactor 패턴[21]이나 스레드 풀링 기법을 지원하기 때문에 리눅스 이외의 플랫폼에서도 비동기 통신을 수행할 수 있는 장점을 가지고 있다.

본 연구에서는 자바 NIO 기반의 푸시 프레임워크를 개발하여 Epoll 방식의 효율적인 클라이언트 연결 관리와 함께 다양한 플랫폼의 푸시 어플리케이션 개발을 지원하는 환경을 제공하여 기존 푸시 프레임워크의 문제점을 개선하고자 한다.

2. JPE 개발

2.1 JPE 시스템

JPE는 롱 폴링 기법을 이용하여 푸시 서비스를 제공하는 프레임워크로서 푸시 서비스를 제공하는 서버를 위한 JPE 코어와 푸시 서비스를 요청하고 정보를 활용

하는 클라이언트의 개발을 위한 JPE 라이브러리로 구성된다. 개발자는 JPE 코어와 JPE 라이브러리를 이용하여 각각 푸시 어플리케이션의 서버와 클라이언트를 구현할 수 있으며, 사용자는 이 클라이언트를 통하여 서버의 푸시 서비스를 손쉽게 이용할 수 있다. JPE 기반의 푸시 어플리케이션에서의 일반적인 동작과정은 다음과 같다.

(1) 사용자가 원하는 정보를 요청
(2) JPE 라이브러리가 동작하여 푸시 서버에 요청
(3) 푸시 서버의 요청 확인 및 클라이언트 연결 정보 저장
(4) 서버 상의 정보 변경 이벤트 발생
(5) 연결된 클라이언트에게 정보 푸시
(6) 클라이언트에서 정보를 수신 후 JPE 라이브러리가 처리
(7) 사용자가 요청을 취소하기 전까지 2~7의 과정 반복

사용자가 원하는 정보를 요청하면 JPE 라이브러리는 AJAX의 XHR(XMLHttpRequest)[22] 객체를 이용하여 JPE 코어 기반 푸시 서버와 연결 및 정보 요청을 처리한다. XHR 객체는 클라이언트 내부에서 사용자 인터페이스와 독립적으로 동작하여 클라이언트가 서버와 비동기적으로 통신을 수행할 수 있는 롱 폴링 기법을 구현한다. XHR 객체는 정보의 효과적인 표현을 위하여 JSON(java Script Object Notation)[23]객체를 이용하며 서버와 정보를 주고받는다. JPE 라이브러리는 이 XHR 객체를 활용하여 푸시 서비스를 정의하고 이를 구현하는 다양한 메소드를 지원하여 푸시 어플리케이션의 효율적인 클라이언트 개발 환경을 제공한다.

JPE 코어는 자바 NIO의 Selector 클래스를 활용하여 클라이언트의 연결 및 요청을 처리한다. 연결된 클라이언트를 분류하고 저장하는 수단으로 채널이라는 통신 통로를 이용한다. 이 채널은 서버 상의 XML 메타 파일을 통하여 정의되며 사용자에 의해 동적으로 생성, 수정될 수 있다. 연결된 클라이언트는 객체로 표현되어 서버 상의 지정된 채널에 저장된다. 해당 채널에 제공할 정보가 발생하면 채널에 저장된 연결 객체를 통하여 클라이언트에게 정보를 제공하게 된다. JPE 코어는 이러한 채널을 정의하고 정보를 효과적으로 제공하기 위한 메소드를 제공하여 푸시 서버 개발을 지원한다. 그림 1은 JPE 기반의 푸시 어플리케이션의 전체 동작 구조를 보여준다.

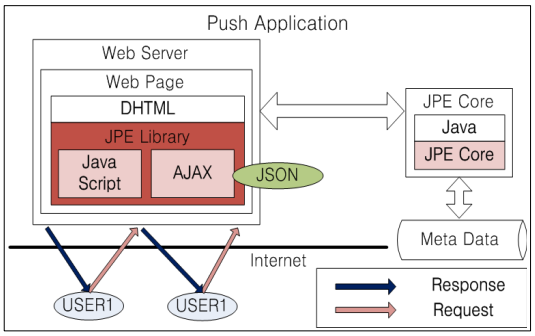


그림 1. JPE 기반 푸시 어플리케이션의 전체 구조도
Fig. 1. Overall Structure of JPE-based Push Application

2.2 JPE 코어

JPE 코어는 푸시 어플리케이션의 서버 개발을 위한 프레임워크로서 서버 상에서 푸시 서비스 구현을 위한 핵심 기능을 정의한다. 그리고 이를 추상화된 메소드로 제공하여 간단한 메소드의 활용을 통하여 푸시 서버를 손쉽게 만들 수 있는 환경을 지원한다. 그림2는 JPE 코어 내부 클래스들간의 전체 구조를 보여준다.

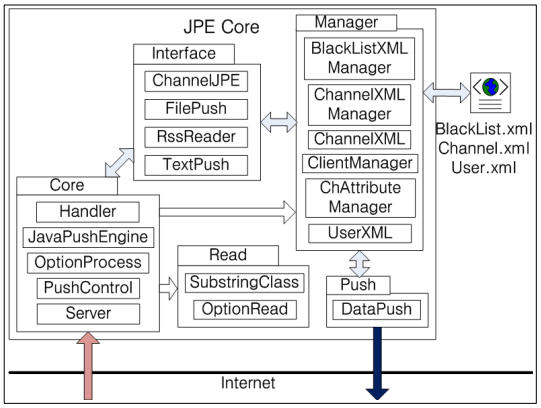


그림 2. JPE 코어의 구조
Fig. 2. Architecture of JPE Core

JPE 코어의 Core 패키지는 푸시 서비스를 실행하기 위한 초기화와 클라이언트의 연결 등 기본적인 실행 환경을 설정하는 역할을 수행한다. 푸시 서버가 실행되고 클라이언트에게 정보를 제공하는 과정은 다음과

같다.

(1) Core 패키지의 Server 클래스 객체를 생성하면서 푸시 서비스를 위한 초기화 작업을 수행
(2) 클라이언트가 접속하여 메시지를 전달하면 Handler 클래스 객체는 이 메시지를 PushControl 클래스 객체로 전달
(3) PushControl 객체는 ChAttbuteManager 객체를 통하여 채널을 검색하고 정보 제공 목적 채널을 가리키는 ChannelManager 객체를 호출
(4) ChannelManager 객체를 통하여 연결된 클라이언트 목록을 ClientManager 객체에서 불러오며, Push 패키지의 DataPush 객체를 이용하여 클라이언트에게 정보를 제공

JPE 코어는 푸시 서비스의 손쉬운 활용을 위하여 JavaPushEngine 클래스를 제공한다. JavaPushEngine 클래스는 내부적으로 FilePush, RssReader, TextPush 객체들을 포함하여 다양한 형태의 데이터를 푸시할 수 있는 기능을 지원하며, 이를 추상화하여 메소드 형태로 제공함으로써 푸시 서버 개발자가 손쉽게 푸시 서비스를 개발, 활용할 수 있도록 한다.

JPE 코어에서는 채널이나 사용자 간련 정보들을 XML 형태의 설정파일을 통하여 구조적으로 관리한다. ChannelList와 User 파일은 각각 채널과 사용자의 상세 정보를 기록하며, JPE 코어는 ChannelJPE 클래스를 통하여 이 파일에 필요한 정보를 불러오거나 수정한다. JPE 코어에서 효과적인 정보 분류 및 클라이언트 목록을 표현하기 위하여 채널을 사용한다. ChannelManager 클래스를 통하여 저장된 채널 목록 및 목록 내부의 클라이언트 목록에 계층적으로 접근하며, 특정 채널을 지정하여 정보를 제공할 수 있다. 그림 3은 JPE의 클라이언트가 속한 채널의 관리를 위한 계층적인 자료구조를 보여준다.

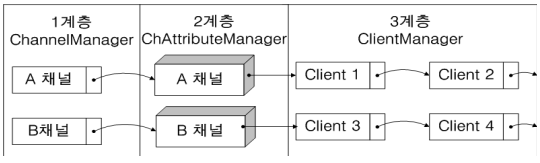


그림 3. 클라이언트 관리를 위한 채널의 계층 구조
Fig. 3. Multi-Layer Structure of Channel for Management of Clients

2계층의 ChAttributeManager 클래스는 채널의 다양한 기능을 효율적으로 활용할 수 있는 속성을 저장한다. 표 3은 채널을 구성하는 속성과 기능을 보여준다.

표 3. 채널의 속성과 기능
Table 3. Attributes and Functions of Channel

속성	기능
title	채널 이름
max	해당 채널의 최대 접속 클라이언트 수
isopen	채널의 열림/닫힘 여부 설정
isgroup	그룹 사용자 이용 여부 설정
type	채널의 공개 여부 설정
password	비공개 채널의 접속 비밀번호 설정
userXML	개인사용자 인증 서비스 제공시 참조하는 XML 파일

채널의 구현은 JPE 코어와 독립적으로 존재하는 XML 파일로 작성하여 서버가 동작하고 있어도 ChannelXML 파일에 현재 채널의 속성을 저장할 수 있도록 하였다. 그림4는 채널을 구성하는 속성의 구조를 정의한 DTD(Document Type Definition)[24] 파일을 보여준다.

```
<!ELEMENT Channel (title, max, isOpen, isGroup, type,
password, usersXML) >
    <!ELEMENT title (#PCDATA) >
    <!ELEMENT max (#PCDATA) >
    <!ELEMENT isOpen (#PCDATA) >
    <!ELEMENT isGroup (#PCDATA) >
    <!ELEMENT type (#PCDATA) >
    <!ELEMENT password (#PCDATA) >
    <!ELEMENT usersXML (#PCDATA) >
```

그림 4. 채널의 속성 구조 정의
Fig. 4. Definition of Attribute Structure of Channel

푸시 기능의 구현을 위해 클라이언트 소켓 통신을 지원하는 자바 NIO의 SocketChannel 클래스의 메소드를 이용하였다. 그림5는 채널에 있는 클라이언트를 불러와서 정보를 제공하는 소스 코드를 보여준다.

```
public void Push(ClientManager queue, ByteBuffer buf){
  SocketChannel sc = null;
  for(int s = queue.queueManager.size(); s>0; ss--){
      sc = queue.queueManager.pop();
    //클라이언트가 서버와 연결된 클라이언트라면 푸시
    if(sc.isConnected()){
      sc.write(buf);
      sc.close();
```

그림 5. 정보를 푸시하는 소스코드
Fig. 5. Sample Code for Information Push

ClientManager 객체에 포함되는 SocketChannel 객체를 불러와서 isConnected() 메소드를 통해 푸시 서버와 연결이 된 클라이언트인지 확인 하고, 연결이 된 클라이언트라면 write() 메소드를 통해 메시지를 전송한다.

2.3 JPE 라이브러리

JPE 라이브러리는 푸시 서버와 통신을 수행하면서 푸시 서비스의 요청 및 푸시 정보를 수신하는 클라이언트의 개발을 위한 프레임워크다. JPE 라이브러리는 푸시 서버와 관련된 설정 정보들을 관리하는 Config 파일과 푸시 서버와 통신을 수행하며 요청 및 정보 수신 등의 기능을 제공하는 JPEclient 라이브러리로 구성된다. 특히, JPEclient는 푸시 서버 접근, 푸시 서비스의 요청, 채널 설정 및 정보 수신 등의 핵심 기능을 추상화하여 제공함으로써 푸시 클라이언트의 손쉬운 개발 환경을 제공할 수 있다. 그림 6은 JPE 라이브러리의 내부 구조와 동작 과정을 보여준다.

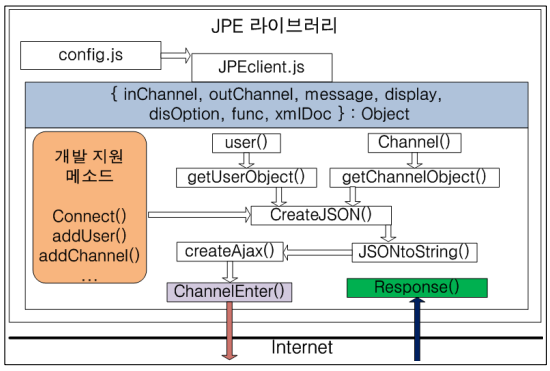


그림6. JPE 라이브러리의 동작 과정
Fig. 6. Work Process of JPE Library

JPEclient 라이브러리는 내부적으로 푸시 서버와 통신, 푸시 서비스 요청 및 정보 수신 등의 기능을 구현하고 있으며 이 기능들을 자바 스크립트 함수로 추상화하여 제공한다. 클라이언트는 이 함수를 바탕으로 푸시 서비스를 구현, 제공할 수 있다.

JPEclient에서 푸시 서버로 전송할 메시지는 inChannel, outChannel, message 속성 등으로 구성되며 각각 접속 채널, 목적 채널, 메시지 내용을 정의한다. 이러한 메시지는 사용자의 요청에 따라 CreateJSON() 메소드에서 JSON 형태로 생성되며, 문자열 형태로 변환된다. 그리고 변환된 메시지는 ChannelEnter() 메소드에서 XHR 객체를 통하여 서버로 전송된다. JPEclient의 Response() 메소드는 서버로부터 푸시되는 메시지를 처리한다. Response() 메소드는 JPE 라이브러리에서 제공하는 출력 방식을 지원하는 display, disOption과 같은 속성을 지원하며, 사용자가 별도로 지정한 방식을 사용하여 출력하기 위한 func 속성을 지원한다. 또한, xmlDOC 속성을 통하여 XML 형태의 메시지를 처리할 수 있다.

JPEclient에서 푸시 서버의 주요 메타 정보의 설정을 요청하기 위해서 user()와 Channel() 메소드를 이용한다. 클라이언트에서 user()와 Channel() 메소드를 통하여 각각 사용자와 채널에 관련된 정보를 전송하고 푸시 서버는 이 정보의 유효성을 검증한 뒤 XML 형태의 메타 파일로 관련 정보를 저장하며 시스템에 반영한다.

```
//푸시 서버에 요청을 위한 메소드
function ChannelEnter(msgJSON, func, xmlFunc,
      display, disOption, inChannel){
  myAjax = createAjax();
  myAjax.open("post","http://"+JPEIP,true);
  myAjax.onreadystatechange = function call(){
    //응답 받으면 실행될 함수
    response(msgJSON, func, xmlFunc, display,
          disOption, inChannel);
    };
  myAjax.send(msgJSON);
```

그림 7. XHR을 이용한 비동기 통신의 구현
Fig. 7. Implementation of Asynchronous Communication using XHR

JPE 라이브러리의 구현은 푸시 서버와의 비동기 통신을 수행하기 위한 XHR 객체의 생성과 이 객체를 다

루는 메소드의 구현으로 이루어진다. 그림 7은 비동기 통신을 위하여 XHR 객체를 생성하고 XHR 객체를 통하여 서버와 비동기 통신을 수행하는 소스 코드의 일부를 보여준다.

createAJAX() 메소드는 통하여 비동기 통신을 수행하기 위한 XHR 객체를 생성한다. 이 후 open() 메소드를 통하여 지정된 푸시 서버에 비동기 통신 연결을 요청하며, 서버로부터 정보를 제공받으면 onreadystatechange 속성에 지정된 call() 메소드를 통하여 그 정보를 처리하게 된다. call() 함수는 response() 메소드를 호출하여 전달 받은 정보를 분석, 처리한다. 그림 8은 call() 함수에서 호출된 response() 메소드에서 제공 받은 정보를 처리하는 과정을 보여준다.

```
//응답을 받으면 실행
function response(msgJSON,func,xmlFunc,
      display,disOption,inChannel){
  var able = 1;
  if(myAjax.readyState == 4){
      if(myAjax.status == 200 || myAjax.statusText ==
"OK"){
                        ...
  //disOption에 따라 출력양식 지정
  if(disOption=="value"){
      display.value = myAjax.responseText;
  }else if(disOption=="innerHTML"){
      display.innerHTML = myAjax.responseText;
  }else if(disOption=="alert"){
                        ...
   //푸시 서버에 다시 연결
  ChannelEnter(msgJSON,func,xmlFunc,display,
      disOption, inChannel);
```

그림 8. response() 메소드의 정보 푸시 과정
Fig. 8. Process of Information Push in "response()"

XHR 객체는 readyState 속성을 통하여 푸시 정보의 유효성을 확인하고 나서 disOption 속성을 통하여 푸시 정보의 형태를 분석한다. 그리고 분석된 형태에 따라 출력 형태를 분류한다. 화면에 출력이 완료되고 나면 ChannelEnter() 메소드를 통하여 푸시 서버와의 비동기 통신 연결을 다시 시도하여 다음 푸시 정보를 전달받기 위한 환경을 설정한다.

2.4 JPE 통신 프로토콜

JPE 코어 기반의 푸시 서버와 JPE 라이브러리 기반의 클라이언트는 JPE 통신 프로토콜을 바탕으로 요청과 응답 정보를 교환한다. JPE 통신 프로토콜은 효과적으로 요청과 응답의 정보를 표현하기 위하여 여러 속성을 정의하고 있으며 이는 JSON 객체를 기반으로 구현된다. 표4는 JPE에서 정의한 JSON 객체의 주요 속성과 기능을 보여준다.

표 4 JPE 통신 프로토콜의 기본 속성
Table 4. Standard Attributes of JPE Communication Protocol

속성	기능
inChannel	클라이언트가 접속하려는 채널
outChannel	클라이언트가 정보를 제공할 채널명
message	푸시하려는 메시지
option	푸시 서버에 정의된 요청을 하는 속성
optionMessage	option속성을 지원하는 JSON 객체

option 속성은 JPE 코어에 정의되어 있는 요청을 하기 위한 용도로 XML 파일의 변경을 요청하거나 비밀번호가 있는 채널에 접속하기 위한 기반이 된다. 표 5는 JPE 코어에 정의된 option 속성의 기능을 보여준다.

표 5 option 속성의 속성값과 기능
Table 5. Values and Functions of "option" Attribute

속성값	기능
addChannel	채널 추가
delChannel	채널 삭제
modChannel	채널 수정
closeEnter	비공개 채널에 접속
userLogin	개인 사용자 접속
addUser	사용자 등록
delUser	사용자 삭제

JPE 클라이언트는 서로 다른 option 기능을 요청하기 위하여 option 속성에 따라서 optionMessage를 동적으로 생성한다. optionMessage 는 JSON 객체를 기반으로 푸시 서버로 요청하는 메시지를 가지며, 표 6에서 option Message 속성의 기능을 보여준다.

표 6 option 속성에 따른 optionMessage 속성
Table 6. "optionMessage" Attribute along with "option"

option 종류	속성	설명
addChannel delChannel modChannel	chName	채널 이름
	maxUser	클라이언트의 최대 접속 허용 수
	open	공개 설정 여부
	close	비공개 설정 여부
	chPw	비밀번호 설정 여부
	pw	채널 비밀번호
closeEnter userLogin addUser delUser	channel	채널 이름
	id	사용자 아이디
	password	사용자 비밀번호

3. JPE 기반 푸시 어플리케이션 개발

3.1 JPE 적용 방법

JPE 코어를 이용한 푸시 서버의 구현 과정은 다음 순서와 같다.

(1) 구현하는 푸시 서버에 JPE 코어를 포함
(2) JavaPushEngine 클래스의 객체 생성 후 start(int port) 메소드를 이용하여 서버 실행
(3) JavaPushEngine에 포함되어 있는 Interface 패키지의 클래스를 이용하여 푸시 서버의 기능을 정의

JPE 코어는 푸시 어플리케이션의 내부에 포함되어 푸시 서비스를 제공하는 핵심 프로세스가 된다. 푸시 서버는 JPE 코어의 JavaPushEngine 클래스의 객체를 생성하고 초기화함으로써 푸시 서비스를 시작할 수 있다. 그림 9는 JPE 코어를 이용하여 푸시 서비스를 제공하는 푸시 서버 구현 코드의 일부를 보여준다.

```
function class Test{
    //JavaPushEngine 객체 생성
    private JavaPushEngine jpe = new JavaPushEngine();
    Test(){
        int port = 7268;
        jpe.JavaPushEngineStart(port);   //서버 동작
    }
...
public void run{
    Date date = null;
    while(true){
        //현재 시간을 지정된 채널로 푸시
        date = new Date();
        jpe2.textPush.push(date.toString(), "ChannelA");
        jpe2.textPush.push(date.toString(), "ChannelH");
        try {
        Thread.sleep(1000);
```

그림 9. JPE 코어를 이용한 푸시 서버 예제
Fig. 9. Sample Code of Push Server based on JPE Core

서버 상의 현재 시간을 문자열 형태로 변환하여 지정된 채널에 연결된 각 클라이언트들에게 푸시하기 위하여 JavaPushEngine 객체의 textPush 내부 객체를 이용한다. textPush 객체는 문자열 형태의 데이터를 전송하기 위한 하부구조를 지원하며 이를 위하여 push() 메소드를 구현하였다. push() 메소드는 자바의 다형성을 기반으로 구현되어 있어 다양한 형태의 데이터를 다루는 푸시 서비스들을 하나의 인터페이스로 활용할 수 있게 한다. 표 7은 푸시 서비스를 개발하기 위하여 JPE 코어에서 제공하는 핵심 메소드를 보여준다.

표 7 JPE 코어에서 제공하는 기본 메소드
Table. 7. Standard Method of JPE Core

객체	함수	기능
channelJPE	createCH()	채널 생성
	deleteCH()	채널 삭제
	modifiCh()	채널 수정
testPush	push()	텍스트 기반 문자열 푸시
	getClient()	해당채널의 모든 클라이언트 정보 확인
filePush	imagePush()	이미지 파일 푸시
	fileInfoPush()	xml과 text 기반의 파일 푸시
RessPush	rsstoXML()	RSS 를 다운받고 XML로 저장

3.3.2 JPE 라이브러리 적용 방법

JPE 코어 기반의 푸시 서버와 비동기 통신을 수행하며 푸시 서비스를 활용하는 JPE 라이브러리 기반의 푸시 클라이언트를 개발하는 과장은 다음 순서와 같다.

(1) config.js 와 JPEclient 파일을 웹 페이지에 포함
(2) config.js 파일을 열어 푸시 서버의 IP 주소와 포트 번호를 작성

(3) JPE 라이브러리의 객체를 생성
(4) 생성된 객체를 이용하여 요청과 응답 방식 정의

그림 10은 JPE 라이브러리를 기반으로 푸시 서비스를 활용하는 푸시 클라이언트의 구현 코드 일부를 보여준다.

```
<script type="text/javaScript" src="config.js"></script>
<script type="text/javaScript"
                src="client/JPEclient.js"></script>
...
<script type="text/javaScript">
    var jpe = new JavaPush();//JPE 라이브러리 객체 생성
    jpe.setServer(JavaPushEngineIP); //서버 설정
    jpe.setDisplay(document.getElementById("time"),
        "value");
    jpe.connect("ChannelA"); //ChannelA 채널에 접속
</script>
```

그림 10. JPE 라이브러리를 이용한 클라이언트 예제
Fig. 10. Sample Code of Push Client based on JPE Library

비동기 통신을 위하여 JPEclient 라이브러리를 포함하며 이 라이브러리를 기반으로 자바스크립트 코드에서 JavaPush 객체를 생성한다. JavaPush 객체의 setServer() 메소드를 통하여 연결할 푸시 서버를 설정하고 connect() 메소드를 통하여 서버의 특정 채널로 접속한다. setDisplay() 메소드를 이용하여 푸시 서버로부터 제공 받은 정보의 출력 형태를 손쉽게 지정할 수 있다. 표 8은 JPE 라이브러리에서 지원하는 출력 형태의 속성을 보여준다.

표 8. JPE 라이브러리의 출력 형태 속성
Table 8. Attributes of Output Type in JPE Library

속성	설명
value	Document 객체의 하위 객체에 값을 출력
inner HTML	Document 객체 자체의 푸시 내용 출력
alert	경고창에 푸시 내용 출력
image	이미지 파일 출력
reload	새로고침
value+	푸시받은 내용을 기존내용에 추가

JPE 라이브러리는 개발자가 직접 출력 형태를 정의한 메소드를 작성하여 클라이언트가 응답을 받으면 그 메소드가 실행되도록 하는 메소드를 지원한다. 그리고 JPE 프로토콜의 option 속성을 이용하는 요청 메소드와 XML 파일 추출 기법을 지원하는 메소드를 제공한다. 표 9는 푸시 서비스의 적용을 지원하는 JPE 클라이언트의 핵심 메소드를 보여준다.

표 9 ChannelJPE의 메소드
Table 9. Methods on "ChannelJPE"

함수	설명
setDisplay()	출력 양식 지정
setFunction()	사용자 정의 함수로 응답 방식 지정
setMessage()	목적지 채널에 전송할 메시지 설정
addChannel()	채널 생성 요청
setUserLogin()	개인 로그인 요청
connect()	해당 채널에 연결

3.3 어플리케이션 프로토타입 개발

본 연구에서는 JPE를 이용하여 실시간으로 웹 정보를 푸시할 수 있는 푸시 어플리케이션의 프로토타입을 개발하였다. 본 어플리케이션은 JPE 코어 기반의 경매정보 푸시 서버와 JPE 라이브러리 기반의 푸시 클라이언트로 구성된다.

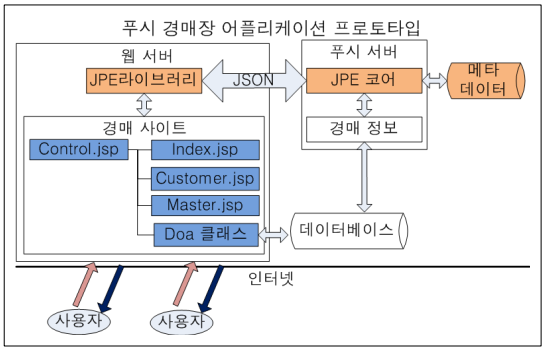


그림 11. JPE 기반 푸시 어플리케이션의 구조
Fig. 11. Structure of JPE-based Push Application

그림 11은 경매 정보 푸시 어플리케이션의 전체 구조도를 보여준다. 본 어플리케이션을 실행하면 JPE 라이브러리에서 푸시 서버에 접속하고, 푸시 서버는 경매정

보가 변경되거나, 다른 클라이언트에서 경매 정보를 변경하면 클라이언트에게 정보를 제공한다. 본 어플리케이션은 경매 상황을 한눈에 볼 수 있는 Index 페이지와, 경매 품목에 입찰 할 수 있는 Customer 페이지로 구성된다. 그림 12는 경매 정보를 실시간으로 제공하는 푸시 어플리케이션의 일부를 보여준다.

그림 12. 경매 정보를 제공하는 푸시 어플리케이션
Fig. 12. Push Application providing Auction Information

Index 페이지는 남은 시간과 최저가를 확인할 수 있다. 남은 시간은 푸시 서버가 실시간으로 제공하며, 최저가는 다른 클라이언트에서 입찰하는 가격을 푸시 서버를 통해 제공받는다. 푸시 서비스의 적용을 위해 JPE 라이브러리를 이용하여 경매 사이트에서 푸시 서비스를 구현하였다. 그림 13은 경매 정보 푸시 어플리케이션의 클라이언트 소스 코드의 일부를 보여준다.

```
//Index 페이지 소스코드
var channeltime = new JavaPush();
channeltime.setFunction("doTime"); //응답 시 처리 메소드
channeltime.connect("Channeltime"); //해당 채널에 접속
var channelCoast = new JavaPush();
channelCoast.setFunction("doCoast");//응답 시 처리  설정
channelCoast.connect("Channelcoast");//해당 채널에 접속
function doTime(){  //시간을 제공 받으면 실행 될 메소드
document.getElementById("channel"+recieve).innerHTML=time;
}
function doCoast(){//최저가를 제공 받으면 실행 될 메소드
document.getElementById("coast"+recieve).innerHTML=coast;
}
...
//Customer 페이지 소스코드
function sendCoast(){
    var channelDSend = new JavaPush();
    //Channelcoast 채널에, 입찰할 가격을 메시지로 설정
    channelDSend.setMessage("Channelcoast",
        document.getElementById("charge").value);
    channelDSend.connect("");  //푸시 서버와 연결
```

그림 13. JPE 라이브러리를 이용한 클라이언트 소스 코드
Fig. 13 Sample Code of JPE Library-based Client

Customer 페이지 소스코드에서 setMessage() 메소드를 이용하여 입찰자 채널에 새로운 입찰 가격 정보를 푸시 서버에게 전송한다. 푸시 서버는 새로운 입찰 가격 정보를 확인하고, Index 페이지에 있는 입찰 가격 채널에 새로운 입찰 가격을 제공한다. 그림 14는 경매 정보 푸시 어플리케이션에서 경매 정보가 실시간으로 사용자의 클라이언트 정보로 제공되는 경과를 보여준다.

그림 14. 경매 정보가 실시간으로 제공되는 경과
Fig. 14. Realtime Progress of Providing Auction Information

4. 시스템 평가

4.1 시스템 특징

JPE 푸시 엔진은 다양한 사용자들의 요구를 충족시키는 푸시 어플리케이션을 개발하기 위한 효과적인 플랫폼으로 개발되었으며, 기존의 여러 푸시 엔진들과의 특징에 대한 비교를 표10에 기술하였다.

표 10 여러 푸시 시스템의 특징
Table 10. Features of Various Push Systems

	APE	Pushlets	BlazeDS	JPE
프로토콜	APE Protocol	HTTP	HTTP	HTTP
서버 구현기술	C	Servlet	Java	Java
클라이언트 구현기술	Ajax, DHTML	Ajax, DHTML	Flex	Ajax, DHTML
푸시 기술	롱 폴링	HTTP 스트리밍	HTTP 스트리밍	롱 폴링
지원 O/S	리눅스	윈도우즈, 리눅스	윈도우즈, 리눅스	윈도우즈, 리눅스
멀티 채널 지원	○	×	○	○

Pushlets은 서블릿 컨테이너를 지원하는 웹 서버와 연동되어 푸시 서비스 개발이 용이하지만, 정보를 분류하여 제공하기 위한 채널을 지원하지 않는다. BlazeDS와 APE는 멀티 채널을 지원하여 클라이언트에게 채널별로 정보를 제공할 수 있다. 그러나 BlazeDS는 Flex 기반으로 동작하며, APE는 리눅스 기반의 어플리케이션만을 지원하기 때문에 플랫폼에 종속적인 문제가 있다.

본 연구에서 개발된 JPE는 기존의 푸시 엔진의 단점을 보완하여 멀티 채널을 활용하고, 자바 NIO 기반의 플랫폼 독립적인 푸시 어플리케이션 개발 환경을 제공한다.

4.2 시스템 성능 비교

본 연구에서는 개발된 JPE와 함께 롱 폴링 기법의 푸시 서비스를 제공하는 APE의 성능을 다음과 같은 방법으로 비교, 분석하였다.

(1) 클라이언트가 푸시 서버에 접속하면 일정한 크기의 데이터를 제공하는 기능의 푸시 서버 구현
(2) 클라이언트가 최초 요청시 현재 시간을 측정
(3) 클라이언트가 1ms 간격으로 1000번의 메시지를 제공 받으면 현재 시작을 측정 후 (2)의 시간의 차이를 기록
(4) (2)~(3)의 과정을 10회 반복후 시간 기록
(5) 데이터 크기를 변경하여 (2)~(4)과정을 반복

그림 15는 두 시스템간의 성능을 비교한 결과를 보여준다.

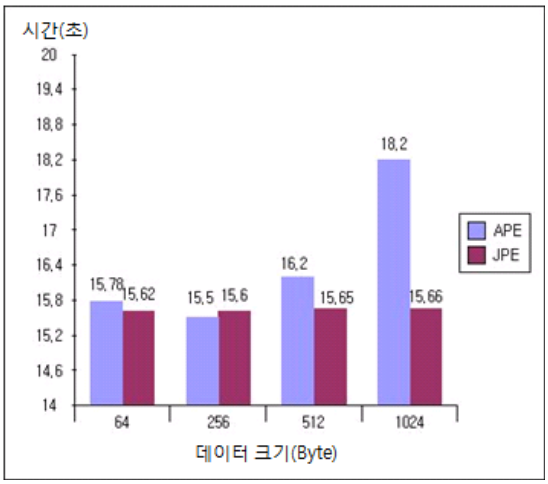


그림 15. APE와 JPE의 성능 비교
Fig. 15. Performance Comparison Between APE and JPE

각각 JPE와 APE 기반의 푸시 어플리케이션은 전송할 데이터의 크기가 작은 환경에서는 거의 유사한 성능을 보여주었다. 하지만 전송 데이터의 크기가 512 바이트를 넘어서면서부터 APE는 전송 시간이 점점 늘어나는데 반해 JPE는 전송 시간의 변화가 크게 두드러지지 않았다. 특히 1024 바이트의 데이터 전송 시에 APE는 급격한 전송 시간 지연 현상이 나타났지만 JPE는 꾸준히 성능이 유지되는 모습을 보여주었다.

JPE는 손쉬운 푸시 서비스를 개발하는 환경을 제공하는 것은 물론, 다양한 데이터를 전송하면서도 꾸준한 전송 성능을 유지하기 때문에 다양한 푸시 어플리케이션을 개발하기 위한 효과적인 플랫폼으로 사용될 수 있다.

Ⅳ. 결 론

본 논문에서는 푸시 어플리케이션의 효과적인 구현을 지원하는 JPE의 개발에 대하여 기술하였다. JPE는 푸시 서버의 손쉬운 구현을 지원하는 JPE 코어와 이와 연

동하는 푸시 클라이언트를 위한 JPE 라이브러리로 구성된다.

JPE 코어는 자바 NIO의 Selector 클래스를 기반으로 클라이언트와의 비동기 통신을 지원하며, 리눅스의 Epoll을 지원함으로써 많은 클라이언트를 동시에 처리한다. 또한, 자바 기반으로 동작하여 다양한 플랫폼에서의 동작을 지원하며 멀티 채널을 통하여 사용자별로 정보를 분류하여 제공하기 때문에 응용 범위가 다양하다. 푸시 서버 개발자는 JPE의 JavaPushEngine 클래스에서 제공하는 메소드들을 통하여 손쉽게 푸시 기능을 활용하고 이를 이용한 다양한 푸시 서비스를 구현할 수 있다.

JPE 라이브러리는 JPE 코어 기반의 푸시 서버와의 비동기 통신을 지원하는 자바스크립트 클라이언트 프레임워크로서 AJAX를 통하여 통신을 수행한다. AJAX의 XHR 객체는 푸시 서버와의 지속적으로 연결을 유지하여 클라이언트와 서버와의 롱 폴링 푸시 기법의 구현 기반이 된다.

또한, XHR은 JSON 객체를 이용하여 클라이언트의 정보를 효과적으로 표현하여 전송한다. 자바스크립트 형태의 JPE 라이브러리는 다양한 브라우저에서 동작하며, 푸시 서비스의 추상화를 통하여 유연한 푸시 클라이언트 개발 환경을 지원한다. 특히, JPE는 채널과 사용자 정보 등을 메타 파일로 관리하며, 이를 쉽게 접근하고 설정할 수 있는 다양한 프로그래밍 인터페이스를 제공함으로써 푸시 어플리케이션의 관리 기능을 쉽게 구현할 수 있게 한다. 본 연구에서는 개발된 JPE를 통하여 경매 정보를 실시간으로 제공하는 푸시 어플리케이션의 프로토타입을 구현하였으며, APE 기반 어플리케이션과 비교를 통하여 JPE의 효율성을 확인하였다.

개발된 JPE는 푸시 기술 구현을 위한 프로그래밍 인터페이스를 제공하여 웹 기반 푸시 어플리케이션의 효과적인 개발을 지원한다. JPE는 방대하고 급변하는 정보를 효율적으로 제공하는 푸시 어플리케이션을 위한 프레임워크로서 널리 활용될 것으로 기대된다.

참고문헌

[1] Shishir Gundavaram, "CGI Programming on the World Wide Web," 1st Edition, O'Reilly Media, 1996.

[2] "http://en.wikipedia.org/wiki/Push_technology," Push Technology.

[3] 어세룡, "웹2.0을 위한 Ajax 플랫폼," 정보과학회지, 한국정보과학회, v.26 no.9, pp.47-57, 2008.

[4] "http://en.wikipedia.org/wiki/Ajax," Ajax

[5] "http://en.wikipedia.org/wiki/Long_polling." Long Polling

[6] Sachin Deshpande, Wenjun Zeng, "HTTP streaming of JPEG2000 images," in the Proceeding of ITCC'01, IEEE Comput. Soc, pp.15-19, 2001.

[7] "http://ajaxpatterns.org/HTTP_Streaming," HTTP Streaming.

[8] "http://www.pushlets.com/," Pushlet

[9] "http://opensource.adobe.com/wiki/display/blazeds/BlazeDS," Adobe BlazeDS.

[10] 이홍창,김보현,오훈,이명준 "BlazeDS에서의 효과적인 Push 서비스를 위한 메시지 필터링," 한국 컴퓨터정보학회논문지, 제15권, 제 6호, 37-44쪽, 2010년 6월.

[11] "http://www.ape-project.org/," APE

[12] Cui Bin, "Realization of EPOLL-based Linux Online Games Server," CONTROL AND AUTOMATION, Weijisuanji Xinxi Zazhishe, v.172, pp.64-66, 2006

[13] "http://en.wikipedia.org/wiki/Chunked_transfer_encoding." Chunked transfer encoding

[14] "http://www.adobeflex.co.kr/aboutflex/down/2.pdf," Adobe Flex 2

[15] "http://www.adobe.com/products/air/," Adobe AIR

[16] Dave Crane, Phil McCarthy, "Comet and Reverse Ajax: The Next-Generation Ajax 2.0," 1st Edition, Apress, 2008

[17] "http://en.wikipedia.org/wiki/Comet_(programming)," Comet Programming

[18] "http://en.wikipedia.org/wiki/Edge_triggered_interrupt," Edge-triggered

[19] "http://en.wikipedia.org/wiki/Edge_triggered_interrupt," Level-triggered

[20] Aaron Newton, "MooTools Essentials: The Official MooTools Reference for JavaScript and Ajax Development," 1st Edition, Apress, 2008.
[21] "http://en.wikepedia.org/wiki/Reactor_pattern." Reactor Pattern
[22] "http://en.wikepedia.org/wiki/XHR." XML HttpRequest
[23] "http://en.wikepedia.org/wiki/JSON." JSON
[24] 김영란 "XML DTD의 효율적인 검색을 위한 구조 정보 및 인덱스 메카니즘," 한국컴퓨터정보학회논문지, 제 8권, 제 3호, 80-86쪽, 2003년 9월.

박종은(Jong-Eun Park)

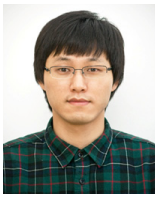
2011년 울산대학교 컴퓨터정보 통신공학부 졸업(학사)
현재 울산대학교 정보통신공학 석사과정

※관심분야 : 협업 시스템, 모바일 정보시스템, 분산 프로그래밍, 클라우드 컴퓨터

권오진(O-Jin Kwon)

2011년 울산대학교 컴퓨터정보 통신공학부 졸업(학사)
현재 울산대학교 정보통신공학 석사과정

※관심분야 : 협업 시스템, 모바일 정보시스템, 분산 프로그래밍, 클라우드 컴퓨터



이홍창(Hong-Chang Lee)

2006년 울산대학교 컴퓨터정보 통신공학부 졸업(학사)
2008년 울산대학교 컴퓨터정보 통신공학부 졸업(석사)
2010년 울산대학교 컴퓨터정보통신공학부 박사 수료
※관심분야 : 웹기반 정보시스템, 클라우드 컴퓨터, 모바일 프로그래밍

이명준(Myung-joon Lee)

1980년 서울대학교 수학과 졸업 (학사)
1982년 한국과학기술원 전산학과 졸업(석사)
1991년 한국과학기술원 전산학과 졸업(박사)
1982년 ~ 현재 울산대학 컴퓨터정보통신공학부/전기공학부 교수
1993년 ~ 1994년 미국 버지니아대학 전산학과 교환교수
2005년 미국 캘리포니아 주립대학 교환교수
※관심분야 : 웹기반 정보시스템, 프로그래밍언어, 생물 정보학, 센서네트워크 프로그래밍환경