韓國컴퓨터情報學會 論文誌
第15卷 第6號, 2010. 6.

2010-15-6-3-1

BlazeDS에서의 효과적인 Push 서비스를 위한 메시지 필터링

이홍창*, 김보현**, 오훈***, 이명준****

Message Filtering for Effective Push Service in BlazeDS

Hong-Chang Lee*, Bo-Hyeon Kim**, Hoon Oh***, Myung-Joon Lee****

요 약

일반적으로 HTTP 프로토콜은 클라이언트의 요청에 따라 서버가 응답하는 형태로 정보를 제공한다. 이러한 방식은 서버가 능동적으로 정보를 제공할 수 없기 때문에 급변하는 정보를 효과적으로 제공하기 어렵다. HTTP 프로토콜의 이러한 단점을 극복하기 위하여 등장한 Server Push 기술은 클라이언트의 명시적인 요청이 없이도 서버가 클라이언트에게 변경된 정보를 능동적으로 제공할 수 있도록 한다. Adobe BlazeDS는 Push 서비스를 지원하는 웹 서버로서 이를 확장하여 웹 기반의 Push 어플리케이션을 개발할 수 있다. 하지만, BlazeDS는 여러 유형의 사용자들에 따라 푸시될 정보를 수정하는 필터링 기능의 부재로 다양한 상황을 효과적으로 처리할 수 있는 Push 서비스를 개발하는데 많은 어려움이 따른다. 본 논문에서는 BlazeDS 기반의 Push 서비스를 효과적으로 개발할 수 있도록 BlazeDS에 메시지 필터링 기능을 추가하고 이를 활용하는 방법을 제시한다. BlazeDS에 추가되는 메시지 필터링 기능은 사용자의 요청에 따라 푸시될 정보를 수정하고 각 정보들을 요청한 사용자들에게 전송한다. 메시지 필터링 기능이 지원되는 BlazeDS는 다양한 사용자들의 개별적인 상황에 따라 맞춤형 정보를 지원하는 Push 서비스를 손쉽게 개발할 수 있는 환경을 제공한다.

Abstract

In general, an HTTP server sends information in response to requests from clients. Since it does not support active information delivery to clients, it can not efficiently provide the rapidly changing information to clients. Overcoming this shortcoming of the HTTP protocol, the technology known as server push enables the HTTP server to actively provide information to clients without explicit requests from clients. Adobe BlazeDS is a web server supporting the server push technology, helping users to develop web-based push applications. Unfortunately, since the BlaseDS server have no functions of filtering the information to be pushed according to various types of users, there are difficulties in developing push applications handling various situations in a efficient way. In this paper, to support effective development of push applications on BlazeDS,

• 제1저자 : 이홍창 교신저자 : 이명준
• 투고일 : 2010. 01. 07, 심사일 : 2010. 02. 04, 게재확정일 : 2010. 02. 25.
* 울산대학교 컴퓨터정보통신공학부 박사 수료 ** 울산대학교 컴퓨터정보통신공학부 석사과정
*** 울산대학교 컴퓨터정보통신공학부 부교수 **** 울산대학교 컴퓨터정보통신공학부 교수

we present the methods of adding a message filtering facility to BlazeDS and utilizing it. According to the filtering request of clients, the message filtering facility modifies information to be pushed, sending the modified information to the clients. The extended BlazeDS server with the message filtering facility provides environments to easily develop push services customized for various clients with their own situations.

▸ Keyword : 서버 푸시(Server Push), 블레이즈DS(BlazeDS), 메시지 필터링(Message Filtering)

Ⅰ. 서 론

인터넷을 통하여 정보를 검색하고 공유하는 사용자들이 늘어남에 따라 효과적으로 정보를 제공하는 방법에 대하여 다양한 연구들이 시도되고 있다. 인터넷은 HTTP 프로토콜을 기반으로 통신을 수행하는 클라이언트와 서버로 구성된다. 인터넷에서 사용되는 HTTP 프로토콜은 클라이언트의 요청에 따라 서버가 응답하여 정보를 제공하는 형태로 동작한다. 이러한 방식은 클라이언트가 필요로 하는 정보를 서버가 수동적으로 제공하는 어플리케이션에는 적합하나 서버 상에서 수시로 변경되는 정보를 서버가 능동적으로 클라이언트에게 제공하는 형태의 어플리케이션에는 쉽게 적용하기 힘들다. 특히, 오늘날에는 주식 및 환율정보, 국제유가정보, 부동산정보와 같은 중요한 산업, 금융 정보가 수시로 급변함으로 기존의 HTTP 프로토콜을 활용한 인터넷 기술만으로는 정보를 효과적으로 제공하는 어플리케이션을 개발하기에 많은 어려움이 따른다.

서버 푸시(Server Push)[1,2] 혹은 푸시(Push) 기술은 기존의 HTTP 프로토콜을 이용한 요청/응답 방식의 수동적인 정보 제공 형태가 아닌, 클라이언트의 명시적인 요청 없이도 서버 상의 변경된 정보를 서버가 능동적으로 클라이언트에게 제공하는 인터넷 기술이다. 기존의 HTTP에서는 주식정보 제공 어플리케이션처럼 클라이언트의 주기적인 요청에 의하여 서버에 부하가 걸려 서버자원이 낭비되기도 하며, 시스템 모니터링 어플리케이션처럼 때로는 서버 상에 중요한 정보의 변경되었음에도 클라이언트가 요청하지 않아 정보가 적시에 제공되지 않는 상황이 발생하기도 하였다. 서버 푸시는 롱 폴링(Long Polling), HTTP 스트리밍[3,4] 등과 같은 기법을 이용하여 서버가 클라이언트에게 능동적으로 정보를 전송할 수 있는 기술이다. 서버 푸시 기술을 지원하는 서버는 클라이언트와의 연결을 유지하고 있으면서 서버 상의 정보가 변경되면 이벤트가 발생되어 연결된 클라이언트에게 바로 변경된 정보를 전송한다.

Adobe[5] 사의 BlazeDS[6]는 웹 서버에서 동작하는 오픈 소스 푸시 서비스 모듈로서 원격개체 호출, 메시지 푸시 서비스 등의 다양한 기능을 제공한다. BlazeDS는 서버 푸시 기술을 이용하여 다양한 어플리케이션을 개발할 수 있는 프레임워크 및 API를 제공하며, 이를 이용하여 채팅, 메시징 서비스와 같은 실시간 어플리케이션을 손쉽게 개발할 수 있다. 하지만, 여러 유형의 사용자들에 따라 푸시될 정보를 수정하는 필터링 기능의 부재로 다양한 상황을 효과적으로 처리할 수 있는 푸시 서비스를 개발하는데 많은 어려움이 따른다.

본 논문에서는 BlazeDS의 메시징 서비스를 바탕으로 효과적인 푸시 서비스를 개발할 수 있도록 BlazeDS에 메시지 필터링 기능을 추가하고 활용하는 방법을 제시한다. 기존의 BlazeDS의 메시징 서비스는 사용자들이 등록된 채널만을 구별하여 일괄적으로 정보를 전송하기 때문에 각 사용자들의 요구에 부합하는 정보를 가공하여 제공할 수 없다. BlazeDS에 추가되는 메시지 필터링 기능은 사용자의 요청에 따라 푸시될 정보를 수정하고 각 정보들을 요청한 사용자들에게 전송한다. 메시지 필터링 기능이 지원되는 BlazeDS는 다양한 사용자들의 개별적인 상황에 따라 맞춤형 정보를 지원하는 푸시 서비스를 손쉽게 개발할 수 있는 환경을 제공한다.

본 논문의 구성은 다음과 같다. 서론에 이어 2장에서는 푸시 서비스와 관련 기술에 대하여 다루며, 3장에서는 BlazeDS의 메시징 서비스에 대한 심층적인 분석과 이를 바탕으로 메시지 필터링 기능의 설계를 다룬다. 그리고 설계를 바탕으로 메시지 필터링 기능을 구현하고 여러 푸시 시스템들과 기능을 비교해보도록 한다. 마지막으로 4장에서 결론을 다룬다.

Ⅱ. 관련 연구

1. Server Push

서버 푸시는 서버와 연결되어 있는 여러 클라이언트들 사이에서 클라이언트의 명시적인 요청이 없어도 서버 혹은 다른 클라이언트가 능동적으로 정보를 전송할 수 있는 인터넷 기반의 통신 스타일을 가리킨다. 서버 푸시 기술의 일반적인 모델로는 Publish/Subscribe 모델이 사용된다. 이 모델에서 클

라이언트는 서버의 특정 주제(topic)에 구독(subscribe)을 하게 되고 다른 클라이언트 혹은 서버에 의하여 이 주제로 정보가 배포(publish)되면 그 정보가 구독한 클라이언트들에게 자동으로 전달된다.

HTTP 프로토콜은 항상 클라이언트의 요청에 따라 서버가 응답하는 방식이기 때문에 서버 푸시 개념이 적용되기 어려운 점이 많았다. 이 때문에 주기적으로 서버에게 요청을 하여 서버 푸시처럼 보이는 서비스를 구현하는 폴링 기법이 주로 사용되었다. 폴링 기법은 클라이언트가 주기적으로 서버에게 요청을 보내는 방식으로 프로그래밍이 쉽고 서버 구현도 간단하다는 장점이 있다. 하지만 클라이언트는 서버의 변경된 자원을 확인하지 않고 요청을 주기적으로 보내기 때문에 많은 부하가 걸리며, 서버 또한 주기적으로 응답을 하게 되어 많은 부하가 걸린다. 그에 따라 의미 없는 정보의 송수신이 많아져 네트워크 역시 많은 부하가 걸리게 된다. 특히, 서버 자원의 중요한 변경에도 불구하고 클라이언트의 요청이 없어서 적시에 제공되지 않는 경우가 빈번하여 실시간 서비스를 제공함에 있어서 크나큰 약점을 지니고 있다.

최근 Ajax[7,8]와 같은 비동기 통신 기술이 사용되고, Flash와 같은 비동기 객체의 활용 기술이 보급됨에 따라 이를 활용한 여러 서버 푸시 기법들이 널리 사용되고 있다. 표 1은 인터넷에서 사용되는 대표적인 서버 푸시 기법들을 보여준다.

표 1. HTTP에서 서버 푸시를 구현하기 위한 기법들
Table 1. Server Push Technologies in HTTP

기법	특징
롱 폴링	· AJAX 등 비동기 프레임워크에서 사용 · 서버가 클라이언트의 요청에 바로 응답하는 형태가 아니라 요청을 받은 후 서버의 정보가 변경되는 이벤트가 발생되면 응답
HTTP 스트리밍	· 실시간 스트리밍처럼 서버와 클라이언트 간의 연결을 유지하며 이벤트가 발생하면 HTTP Chunked 방식으로 정보를 전송 · 구현과 예외처리가 어렵고 호환성이 낮음 · HTTP의 연결 timeout에 관련하여 예기치 못한 시기에 연결이 끊어질 수 있음

2. Push 시스템

Ajax와 같은 비동기 통신 기술들이 등장하면서 인터넷 기반의 푸시 서비스를 제공하는 여러 푸시 엔진들이 등장하였다.

APE(Ajax Push Engine)[9]는 Ajax를 기반으로 실시간 푸시 서비스를 제공하는 오픈소스 푸시 엔진으로서 APE 자바스크립트 프레임워크와 APE 서버로 구성된다. APE 자바스크립트 프레임워크는 MooTools[10] 라이브러리를 기반으로 APE 서버와 효과적인 비동기 통신을 수행하는 클라이언트를 개발할 수 있는 환경을 제공한다. 이러한 비동기 통신을 수행하기 위하여 Ajax 개체를 활용하며, APE 서버와는 APE 프로토콜을 주고받는다. APE 서버는 Epoll-driven[11] 방식의 Comet[12,13] 서버로서 APE 자바스크립트 프레임워크로 구현된 클라이언트와 비동기 통신으로 메시지를 푸시할 수 있다. 특히, Epoll 방식으로 클라이언트와의 연결을 처리하여 빠른 속도로 많은 클라이언트를 관리할 수 있다.

Pushlets[14]은 자바애플릿이나 별도의 플러그인 없이 서버단의 자바객체에서 클라이언트의 DHTML(Dynamic HTML)로 정보를 푸시할 수 있는 서블릿 기반의 서버 푸시 기술이다. Pushlets은 HTTP 스트리밍을 기반으로 서버 측에서 동작하는 서블릿이 클라이언트와 연결을 지속하여 수정된 정보를 적시에 제공한다. 클라이언트는 DHTML을 바탕으로 서버로부터 받은 정보를 브라우저 화면에 바로 보여줄 수 있다.

Adobe BlazeDS는 원격 객체 호출과 웹 메시징을 지원하는 서버 모듈로서 Flex나 AIR와 같은 클라이언트와 통신하며 분산된 서비스를 제공하거나 정보를 푸시하는 기능을 제공한다. Adobe는 RIA(Rich Internet Application)[15,16]를 위한 클라이언트 개발 플랫폼으로 Flex[17]와 AIR[18]를 개발하였고 이를 기반으로 한 클라이언트와 통신하여 다양한 기능을 제공할 수 있는 서버로 LCDS(LifeCycleDS)[19]와 BlazeDS를 개발하였다. BlazeDS의 주요 특징은 그림 1과 같다.

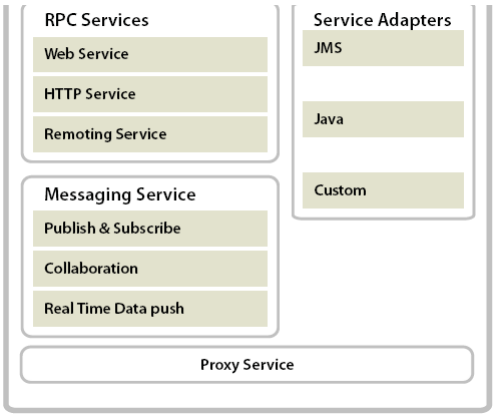

그림 1. Adobe BlazeDS의 주요 특징
Fig. 1. Main Features of Adobe BlazeDS

BlazeDS는 원격 객체 호출(RPC)과 메시징(Messaging), 프록시(Proxy) 기능 등을 오픈 소스로 제공하는 무료 플랫폼이다. BlazeDS는 HTTP나 웹 서비스를 통하여 원격 객체 호

출 서비스를 제공하며, 실시간 정보 푸시이나 배포/구독 모델을 바탕으로 한 메시징 기능을 제공한다.

III. 본 론

1. BlazeDS의 구조와 메시지 필터링 설계

1.1 서버 푸시 기술과 메시지 필터링의 필요성

사용자들은 인터넷을 통하여 다양한 정보를 접할 수 있으며, 필요한 정보를 습득하고 활용할 수 있다. 인터넷 뉴스는 인터넷을 기반으로 사용자들이 정보를 습득할 수 있는 가장 대표적인 서비스이다. 인터넷 뉴스는 다양한 범주의 정보들을 가지고 있으며, 날짜와 시간대별로 급변하는 정보를 제공할 수 있다. 인터넷 뉴스의 정보 제공 과정은 다음과 같다.

(1) 뉴스거리/사건의 발생
(2) 뉴스 기자의 정보 수집 및 디지털 문서로 저장
(3) 정보의 범주 및 메타 정보 입력 후 뉴스 제공 사이트에 전송
(4) 뉴스 제공 사이트는 입력된 뉴스 정보를 메타 정보에 기인하여 분류하고 제공
(5) 사용자들은 뉴스 제공 사이트에 접속하여 필요한 정보를 발췌하여 확인

위 과정을 바탕으로 인터넷 뉴스를 통하여 정보를 확인하는 세 명의 사용자가 있다고 가정하자. 사용자 A는 뉴스 제공 사이트의 모든 뉴스들을 제공받으려고 한다. 사용자 B는 경제 범주의 전체 부동산 뉴스를 제공받으려 하며, 사용자 C는 경제 범주의 울산 지역 부동산 뉴스를 제공받으려고 한다. 각 사용자들은 뉴스 제공 사이트에 자주 접속하여 범주를 확인해 가며 뉴스를 발췌하여 확인하면서 정보를 구할 수 있다. 이러한 형태로 인터넷 뉴스가 제공되는 과정에서 두 개의 큰 문제점이 발생할 수 있다. 우선 수집되는 뉴스 정보가 적시에 제공될 수 없다는 것이다. 뉴스 기자가 빠른 시간에 정보를 뉴스 제공 사이트에 입력했다고 하더라도 사용자가 바로 접속하여 정보를 찾지 않는다면 이 정보는 적시에 제공된다고 할 수 없다. 또 다른 하나는 사용자가 필요로 하는 정보를 찾기가 어렵다는 것이다. 사용자는 필요로 하는 정보를 얻기 위하여 여러 범주를 탐색하며 불필요한 목록과 뉴스들을 확인해야함으로 오랜 시간에 걸려 뉴스를 습득하거나 때로는 부정확한 정보를 습득할 수도 있다. 이러한 문제점을 개선하고 보다 효과적으로 정보를 제공하기 위해서는 사용자의 특별한 요청이 없이도 정보가 서비스 될 때, 즉시 사용자에게 전송이 되어야 하며, 사용자가 어렵지 않게 정보를 습득할 수 있도록 사용자가 필요로 하는 정보만 필터링하여 제공해야 한다.

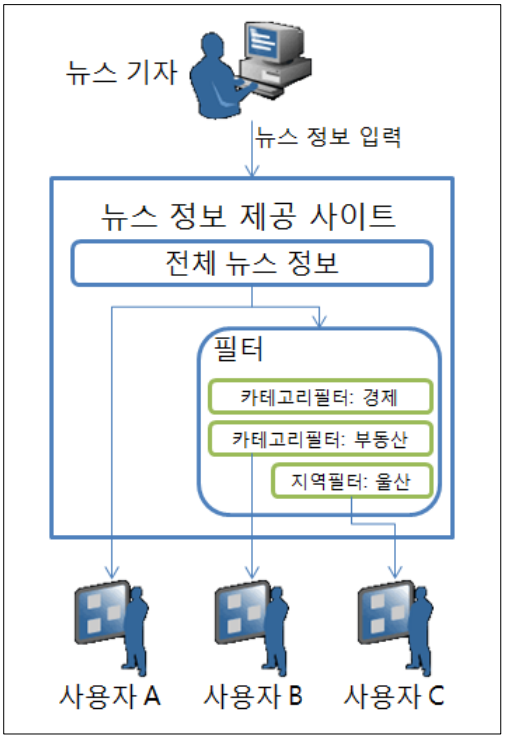

그림 2. 메시지가 필터링되는 예
Fig. 2. An Example of Message Filtering

그림 2는 메시지 필터링 기능을 이용하여 여러 사용자들의 요구에 부합하도록 정보를 분류하며, 서버 푸시 기술을 이용하여 각 사용자에게 정보를 제공하는 과정을 보여준다.

1.2 BlazeDS의 구조와 정보 전송 기법

BlazeDS는 J2EE 기반의 웹 서버의 모듈로서 동작하여 Flex 혹은 AIR 기반 클라이언트와 통신을 수행하며 다양한 서비스를 제공한다. BlazeDS는 클라이언트와 통신하기 위하여 연결을 정의하는 다양한 Endpoint를 가지며 내부 프로세싱을 거친 후 클라이언트와 연결된 Endpoint를 통하여 정보를 전송한다. 메시지가 전송되는 과정을 보여주는 BlazeDS의 내부 구조는 그림 3과 같으며, 다음과 같은 과정을 통하여 메시지가 전달된다.

(1) 클라이언트의 요청에 따라 HTTPEndpoint 객체 생성

(2) 클라이언트는 연결된 HTTPEndpoint 객체에게 메시지를 전송
(3) HTTPEndpoint 객체는 MessageBroker 객체에게 메시지를 전달
(4) MessageBroker 객체는 Service 영역에서 메시징 서비스를 제공하는 MessageService 객체에게 메시지를 전달
(5) MessageService 객체는 연결된 클라이언트들에게 메시지를 전송하는 MessageDestination 객체에게
(6) Destination 객체는 Endpoint를 통하여 연결된 클라이언트들의 목록을 저장하고 있으며, 메시지를 전달받으면 연결된 클라이언트들에게 메시지를 전송

BlazeDS는 HTTPEndpoint나 AMF(Action Message Format)[20]을 이용하는 AMFEndpoint를 통하여 클라이언트와 통신하게 된다. 클라이언트는 서버와의 연결을 정의하는 Channel 객체를 통하여 정보를 요청하거나 메시지를 전달한다. 클라이언트와 연결된 BlazeDS의 Endpoint는 내부의 MessageBroker 객체에게 메시지를 전달하거나 요청한다. MessageBroker는 BlazeDS의 객체로서 전달받은 메시지를 Service 영역으로 전달하는 역할을 한다. MessageBroker는 정보를 분석하여 인증 과정을 거친 뒤, 메시지의 목적지를 파악하여 적합한 Service 객체로 전달한다.

Service 영역은 클라이언트로부터 전달받은 메시지를 처리하고 그 결과를 Destination 영역으로 전달하는 역할을 한다. 메시지의 목적에 따라 RemotingService, HTTPProxyService, MessageService 객체가 각각 메시지를 처리하게 되는데 메시징 서비스의 경우는 MessageService 객체가 그 메시지를 처리한 후 Destination 영역으로 전달한다.

Destination 영역은 일반적인 푸시 시스템에서의 채널과 비슷한 용도로 쓰이는데, 메시지가 전송이 될 목적지 혹은 사용자들을 정의한다. 클라이언트가 접속할 때 자신이 속할 Destination을 지정할 수 있으며 연결이 완료되면 BlazeDS 서버의 해당 Destination에 등록이 된다. BlazeDS의 메시지는 헤더에 Destination 정보를 포함하고 있는데, BlazeDS 서버는 이 정보를 바탕으로 해당 메시지를 전송한다. 메시지를 전달받은 Destination은 Endpoint에 연결된 사용자들 중에 자신에게 등록된 사용자들에게만 메시지를 전달하여 메시지 전송을 완료한다.

1.3 메시지 필터링 설계

BlazeDS는 메시징 서비스에 필요한 다양한 기능을 지원하지만 복합적인 상황에 따라 메시지를 제어하는 필터링 기능을 지원하지 못하고 있다. BlazeDS를 이용하여 푸시 서비스를 제공하는 경우에 모든 클라이언트에게 전체 정보를 전송하고 각 클라이언트가 필요한 정보를 스스로 필터링해서 활용하였다.

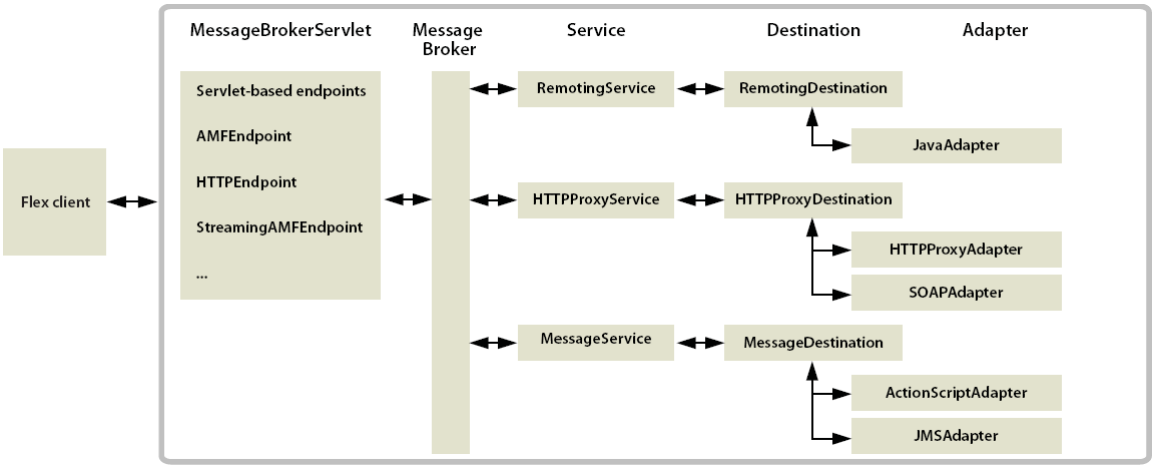

그림 3. Adobe BlazeDS 서버의 구조
Fig. 3. Architecture of Adobe BlazeDS Server

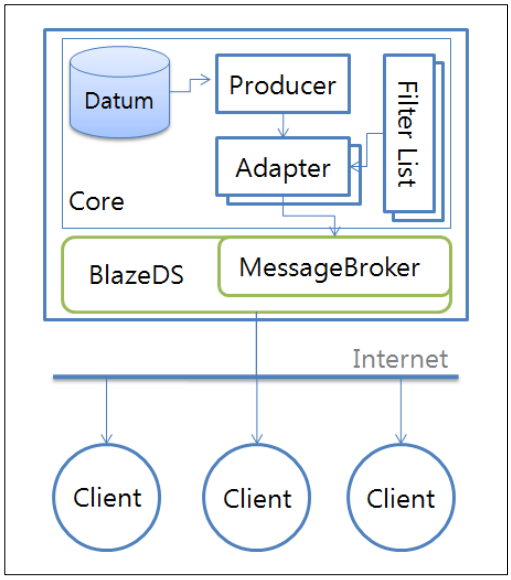

그림 4. 메시지 필터링을 위한 확장된 BlazeDS의 구조
Fig. 4. Structure of the Extended BlazeDS for Message Filtering

그림 4는 메시지 필터링 기능을 지원하도록 확장된 BlazeDS에서 메시지를 전송하는 과정을 보여준다. BlazeDS에 추가된 메시지 필터링 기능은 메시지를 클라이언트에게 바로 전송해주는 것이 아니라 메시지에서 클라이언트가 필요로 하는 부분만 필터링하여 그 결과만을 전송한다. 이러한 기능을 수행하기 위해서 우선적으로 필터링 기능을 실행할 핵심 로직으로서 코어 객체가 필요하다. 코어 객체는 메시지생성 객체(producer) 객체, 필터 객체들, 어댑터 객체들을 생성하고 관리한다. 메시지생성 객체는 서버 상에서 생성되거나 클라이언트로 전달받은 메시지를 다른 클라이언트들에게 전송을 요청하는 역할을 한다.

기존에 일괄적으로 정보를 전체에게 전송하던 BlazeDS와는 달리 이 메시지생성 객체는 사용자의 요청에 따라 메시지를 필터링 할 수 있는 어댑터 객체를 설정할 수 있다. 어댑터 객체는 메시지생성 객체로부터 생성된 메시지를 실제로 BlazeDS의 메시징 서비스에 전달하는 역할을 한다. 이 과정에서 어댑터 객체는 메시지를 단순히 전달만 하는 것이 아니라 클라이언트의 요구에 맞춰 메시지에 여러 필터들을 적용하여 메시지를 필터링한 후 변경된 정보를 전송한다. 어댑터는 필터링이 끝난 메시지를 BlazeDS의 MessageBroker 객체에게 바로 전송하여 메시징 서비스로 전달되도록 한다. 어댑터 객체에서 사용되는 필터 객체는 메시지를 사용자가 지정한 속성 값에 따라 필터링하는 역할을 한다. 동일한 정보를 다양한 사용자의 요구에 맞춰 수정 후 제공할 수 있도록 메시지 생성 클래스는 다양한 어댑터를 설정할 수 있으며, 각 어댑터는 메시지에 대하여 필터를 중복 적용할 수 있도록 다양한 필터를 설정할 수 있다.

앞 절에서 정의한 메시지 필터링 기능을 바탕으로 표 2에서와 같이 기본 인터페이스를 설계하였다.

표 2. 메시지 필터링 기능을 정의하는 인터페이스
Table 2. Interfaces for Message Filtering Functions

인터페이스	목적	주요 메소드
Core	메시지 필터링 기능 실행	· start()
Producer	메시지 생성	· setAdapter() · createData()
Adapter	사용자의 요구별로 생성되어 메시지에 필터 적용시키고 전송	· setFilter() · pushData()
Filter	사용자가 지정한 속성 값으로 메시지 수정	· setItem() · setAttribute() · setData() · filtering()

2. 메시지 필터링의 구현과 적용

2.1 메시지 필터링의 구현

앞 장에서 메시지 필터링 기능을 정의한 인터페이스는 표 3에 기술된 내용의 의미를 가진다.

표 3. 메시지 필터링 기능의 구현 내용
Table 3. Implementation of Message Filtering

구현 클래스	메소드 구현내용
Core	· void start() – 어플리케이션 실행
Producer	· void setAdapter(Adapter adapter) – Adapter 객체 설정 · void createData(Adapter[] adapterArray) – 메시지 생성 후 어댑터 객체에 전달
Adapter	· void setFilter(Filter filter) – 정보를 필터링할 Filter 객체 설정 · void pushData(Object message) – Filter 객체의 filtering() 메소드 호출 – 필터링된 메시지를 BlazeDS 서버로 전송
Filter	· void setItem(String name) – 필터링할 속성 설정 · void setAttribute(String value) – 필터링 속성의 속성값 설정 · void setData(Object message) – 필터링할 메시지 설정 · Object filtering() – 필터링 기능을 실행 후 처리된 결과 반환

필터링된 메시지 전송을 지원하는 푸시 어플리케이션은 Core 인터페이스를 구현한 객체로 정의된다. 이 객체는 Producer, Adapter, Filter 인터페이스를 구현한 클래스의 객체를 생성하고 관리한다. Producer 인터페이스 구현 객체는 실제로 메시지를 생성하는 역할을 한다. Producer 구현 객체는 생성된 정보를 필터링하고 BlazeDS의 messageBroker 객체에게 전달하기 위하여 setAdapter() 메소드의 매개변수를 통하여 Adapter 인터페이스 구현 객체를 설정하고 createData() 메소드를 실행한다. createData() 메소드는 메시지를 생성한 다음에 설정되어 있는 여러 Adapter 구현 객체의 pushData() 메소드를 호출한다. 각 Adpater 구현 객체는 사용자들의 요구별로 생성되는데 요구에 부합하도록 메시지를 필터링하기 위하여 여러 Filter 인터페이스 구현 객체를 설정한다.

Filter 구현 객체는 setAttribute() 메소드를 이용하여 필터링할 속성값을 지정하며, 메시지에 중복하여 적용할 수 있도록 Adapter 구현 객체에 다수의 Filter 구현 객체가 등록될 수 있다. 본 연구에서 구현된 Filter 클래스는 태그와 속성으로 데이터가 분류되는 XML 형태의 메시지를 처리할 수 있도록 정의되며, 체계적인 필터링을 위하여 XML의 태그와 속성의 값을 추출, 비교하여 처리한다. Adapter 구현 객체는 Filter 구현 객체의 setData() 메소드를 통하여 메시지를 전달하며, Filter 구현 객체는 전달받은 메시지를 filtering() 메소드로 처리하여 필터링된 결과 메시지를 Adapter 구현 객체에게 반환한다. 필터링 작업이 모두 끝난 메시지는 Adapter 구현 객체의 pushData() 메소드에서 MessageBroker 객체로 전달되어 각 목적지별로 전송된다. 그림 5는 Producer 구현 객체의 부분 의사코드를 보여준다.

```
class NewsProducer implements Producer{
    …
  void createData(Adapter[ ] adapterArray){
      //Produces and sends messages
    //adapterArray: an array of Adapters
      //message: an original message
    for i←1 to adapterArray.length-1 do
        v←adapterArray[i]
      v.pushData(message)
        }
    …
  }
```

그림 5. Producer 구현 객체의 부분 의사코드
Fig. 5. A Part Pseudocode of a Producer Instance

그림 5에서 NewsProducer 클래스는 Producer 인터페이스를 구현한다. NewsProducer의 createData() 메소드는 메시지를 생성한 후에 필터링할 각 Adapter 구현 객체들에게 전달한다. 그림 6은 Adapter 구현 객체의 부분 의사코드를 보여준다.

```
class LocationAdapter implements Adapter{
    …
  public void pushData(Object message){
   //Filters and sends messages to BlazeDS
    //message: an original message
    //filterArray: a array of Filters
   //mBroker: a MessageBroker instance
    for i←1 to filterArray.length-1 do
        v←filterArray[i]
      v.setData(message)
        w←v.filtering()
      mBroker.sendMessage(w)
        }
    …
  }
```

그림 6. Adapter 구현 객체의 부분 의사코드
Fig. 6. A Part Pseudocode of a Adapter Instance

그림 6에서 LocationAdapter 클래스는 Adapter 인터페이스를 구현한다. LocationAdapter의 pushData() 메소드는 전달받은 메시지를 Filter 구현 객체들에게 전달하여 필터링을 요청하며, 필터링되어 반환된 메시지들을 BlazeDS의 MessageBroker 객체에게 전달한다. 그림 7은 Filter 구현 객체의 부분 의사코드를 보여준다.

```
class XMLFilter implements Filter{
    …
  public Object filtering(){
   //Filters and returns messages
    //message: an original message
   //resultMessage: a result message
    //aName: an attribute name
    //aValue: an attribute value
  Document doc = (Document)message
    for i←1 to doc.length-1 do
        v←doc.elementsAt(i)
      if v.attribute(aName) is aValue
          resultMessage.add(v)
      return resultMessage
        }
    …
  }
```

그림 7. Filter 구현 객체의 부분 의사코드
Fig. 7. A Part Pseudocode of a Filter Instance

그림 7에서 XMLFilter 클래스는 Filter 인터페이스를 구현한다. XMLFilter의 filtering() 메소드는 Adapter 구현 객체로부터 전달받은 메시지를 속성이름과 속성값을 바탕으로 필터링하고 처리된 결과 메시지를 반환한다. 그림 8은 XMLFilter의 실제 구현 코드의 일부를 보여준다.

```
public class XMLFilter implements Filter{
  public void setItem(String name){ ... }
 public void setAttribute(String aValue){ ... }
 public void setData(Object message){ ... }
  public Object filtering() {
  Document doc = (Document)message;
    NodeList nodeList =
   doc.getElementsByTagName("News");
      ...
    Node node =
     nodeMap.getNamedItem(name);
 if((node.getNodeValue()).equals(aValue)){
     //내용 추출 로직
  //resultMessage: 필터링 결과 메시지
      }
      ....
  return resultMessage;
    }
  }
```

그림 8. 날씨 정보를 필터링하는 XMLFilter 클래스
Fig. 8. XMLFilter Class for Filtering Weather Information

그림 8에서 XMLFilter 클래스는 Adapter 구현 객체로부터 전달받은 뉴스 정보를 필터링한다. Adapter 구현 객체는 XMLFilter 객체의 setData() 메소드를 통하여 "doc" 필드에 원본 메시지를 설정하며, setItem() 메소드를 통하여 필터링할 속성으로 "name" 필드에 "location" 속성을 지정하고 setAttribute() 메소드를 통하여 필터링할 값으로서 "aValue" 필드에 "Ulsan"이라는 값을 설정한다. 그 후 호출되는 filtering() 메소드는 "doc"에서 정보를 추출하여 "name"으로 지정된 속성에 값과 "aValue"에 지정된 값을 비교하여 일치하면 해당 정보만 "resultDoc" 필드에 저장한다. "doc" 필드의 모든 정보를 다 비교한 다음 필터링된 정보만 "resultDoc"에 저장하고 이를 반환한다. 그림 9는 원본 메시지와 XMLFilter를 통하여 필터링된 메시지를 각각 보여준다.

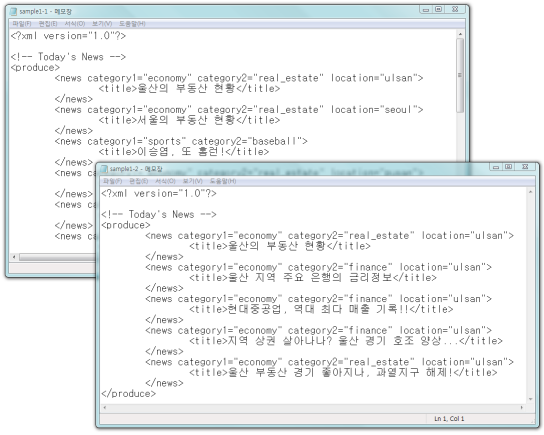

그림 9 원본 메시지와 필터링된 메시지
Fig. 9. Original Message and Filtered Message

2.2 메시지 필터링의 적용

앞 장에서 메시지 필터링 기능의 구현 방법과 이를 토대로 구현한 프로토타입을 살펴보았다. 사용자는 XML 형태의 데이터를 이용한 메시지 푸시 어플리케이션을 사용할 때 이 프로토타입 메시지 필터링 기능을 바로 적용할 수 있다. 앞 장에서 구현된 XMLFilter 클래스에 필요한 속성과 속성값을 부여함으로써 사용자가 사용하는 XML 기반 메시지를 필터링할 수 있다. 또한, 사용자가 직접 정의한 메시지 정의의 데이터를 이용한 어플리케이션을 개발하기 위해서 사용자 정의의 새로운 Filter 구현 클래스를 생성 및 적용할 수 있다. 이러한 Filter 구현 클래스는 Filter 인터페이스를 상속받아 사용자가 직접 구현할 수 있으며, 앞 절에서 구현된 프로토타입 필터인 XMLFilter를 확장하여 구현할 수도 있다. XMLFilter는 메시지를 설정하고 반환하는 형태로 자바의 최상위 클래스 형태인 "Object"를 사용하기 때문에 사용자는 자신이 사용하는 메시지를 설정하고 filtering() 메소드를 구현함으로써 필요한 필터링 기능을 개발할 수 있다. 그림 10은 XMLFilter 클래스를 상속받아 사용자 정의의 메시지를 필터링하는 새로운 필터 클래스를 보여준다.

```
public class TextFilter implements XMLFilter{
    ...
    public Object filtering() {
        File file = (File)message;
        FileReader fr = new FileReader(file);
        BufferedReader br = new BufferedReader(rf);
        Scanner s = new Scanner(br);
        s.useDelimiter("/");
        while(s.hasNext()){
            //필터링 로직
            //resultFile: 필터링 결과 파일
        }
        ...
        return (Object)resultFile;
    }
}
```

그림 10. 사용자가 재정의한 TextFilter 클래스
Fig. 10. TextFilter Class Redefined by User

3. Push 어플리케이션 프로토타입 개발

3.1 뉴스 정보 푸시 어플리케이션 개발

본 연구에서는 BlazeDS에 추가된 메시지 필터링 기능을 이용하여 뉴스 정보를 사용자별로 편집하여 제공하는 뉴스 정보 푸시 어플리케이션의 프로토타입을 개발하였다. 본 어플리케이션은 메시지 필터링 기능을 지원하도록 확장된 BlazeDS 서버와 Flex로 구현된 클라이언트로 구성되며, HTTP 스트리밍을 이용하여 비동기 통신으로 연결된다.

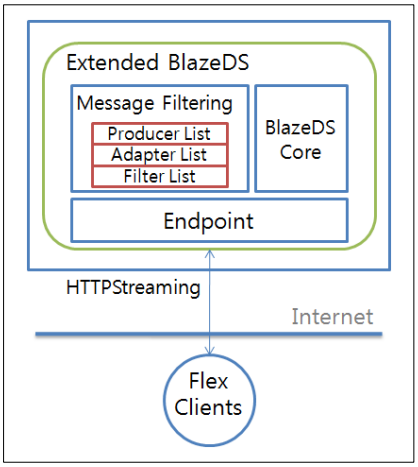

그림 11. 뉴스 정보 푸시 어플리케이션의 전체 구조도
Fig. 11. Structure of News Information Push App.

그림 11은 뉴스 정보 푸시 어플리케이션의 전체 구조도를 보여준다. 사용자는 브라우저에서 클라이언트를 실행하며, 실행 후 서버로부터 뉴스 범주 목록과 해당 범주에 적용 가능한 필터 목록을 받게 된다. 자신이 원하는 범주를 선택하고 해당 범주에 적용할 필터를 선택하면 서버는 이후 해당 범주의 뉴스 정보가 업데이트 될 때마다 사용자가 선택한 필터를 적용하여 필터링된 정보를 전송하게 된다.

뉴스 정보 푸시 어플리케이션의 서버는 자동적으로 뉴스 정보를 업데이트하는 NewsProducer 클래스를 가지고 있으며, 사용자의 요구별로 XMLFilter 클래스의 객체를 설정한 여러 NewsAdapter 클래스의 객체를 가지고 있다. 그림 12는 Flex로 구현된 클라이언트 인터페이스의 일부를 보여준다.

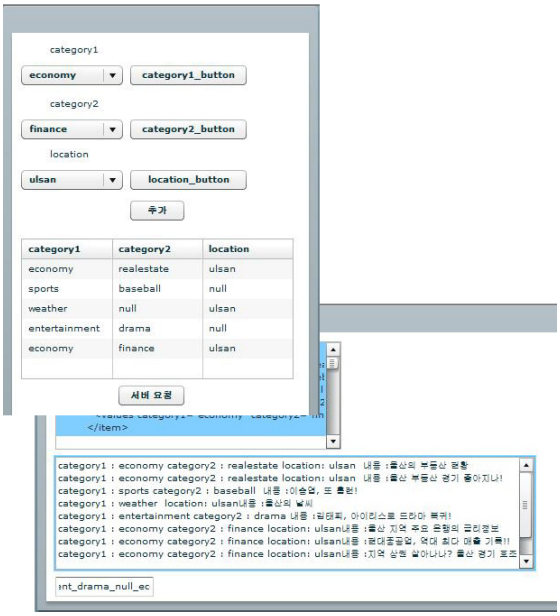

그림 12. 클라이언트의 인터페이스 일부
Fig. 12. A Part of the Client Interface

사용자는 그림 12에서의 왼쪽 프레임을 이용하여 뉴스 범주를 선택하고 해당 범주에서 필요한 부분만큼만 이용하기 위하여 적용 가능한 필터 목록에서 필터들을 선택한다. 선택된 정보와 필터 목록은 서버로 전달되며, 서버는 뉴스 정보가 업데이트 될 때마다 메시지 필터링 기능을 통하여 정보를 필터링한 후 각 사용자가 필요로 하는 정보를 전송한다. 사용자는 서버로부터 받은 정보를 클라이언트의 오른쪽 프레임을 통하여 확인할 수 있다.

3.2 필터 관리도구

뉴스 정보 푸시 어플리케이션과 같이 메시지 필터링 기능이 지원되는 BlazeDS를 이용한 어플리케이션은 제공되는 정보와 적용 가능한 필터들을 사용자들이 손쉽게 확인하고 선택할 수 있어야 한다. 이를 위하여 뉴스 정보 푸시 어플리케이션에 필터 관리 도구가 추가되었다. 필터 관리도구는 생성되는 정보의 형태에 부합하도록 필터링 가능한 속성의 목록과 속성값을 기반으로 필터 객체들을 생성, 관리할 수 있으며, 사용자들이 선택한 필터들을 조합하여 적용하는 Adapter 구현 객체를 생성하여 확장된 BlazeDS에 추가한다. 그림 13은 필터 관리도구 인터페이스의 일부를 보여준다.

그림 13. 필터 관리도구 인터페이스의 일부
Fig. 13. A Part of Filter Management Tool Interface

관리자는 그림 13의 상부 프레임을 통하여 뉴스 정보에 필터링 가능한 속성을 추가 및 관리할 수 있다. 예를 들어, 뉴스 정보가 범주별로 정의되어 있다면 이러한 속성별로 사용자가 선택할 수 있도록 필터 객체들을 만들어준다. 또한, 관리자는 중앙 프레임을 통하여 사용자의 요구를 반영할 수 있는 필터별 속성값을 지정할 수 있다. 그리고 하부 프레임을 통하여 현재 필터에 등록된 속성과 속성값을 확인하고 관리할 수 있다. 이와 같은 기능들을 바탕으로 필터 관리도구를 이용하면 다양한 사용자의 요구에 맞춰 어플리케이션에서 Adapter와 Filter 구현 객체를 미리 만들어둘 필요가 없어 어플리케이션 관리에 효과적이며, 새로운 형태의 요구에도 즉각적으로 반응할 수 있어 푸시 어플리케이션을 실시간적으로 관리할 수 있다.

4. 평가

메시지 필터링 기능을 지원하도록 확장된 BlazeDS는 다양한 사용자의 요구를 충족시키는 푸시 어플리케이션을 개발하기 위한 효과적인 플랫폼이 될 수 있다. 표 4는 기존의 여러 푸시 엔진 혹은 시스템들과 함께 메시지 필터링 기능을 지원하는 BlazeDS의 비교를 보여준다.

표 4. 여러 푸시 시스템의 특징
Table 4. Features of Various Push Systems

	APE	Pushlet	BlazeDS	Extended BlazeDS
통신 프로토콜	APE Protocol	HTTP	HTTP	HTTP
서버 구현기술	C	Servlet	Java	Java
클라이언트 구현기술	AJAX + DHTML	AJAX + DHTML	FLEX	FLEX
푸시 기술	Long 폴링	HTTP 스트리밍	HTTP 스트리밍	HTTP 스트리밍
플러그인 기능	○	○	○	○
메시지 채널 지원	○	×	○	○
메시지 필터링	×	×	×	○
필터 관리기능	×	×	×	○

APE는 Ajax 기술과 롱 폴링 기술을 이용하여 실시간 푸시 서비스 개발을 지원하는 대표적인 푸시 엔진이다. APE는 플러그인 기능을 통하여 서비스를 확장할 수 있으며 채널을 통하여 메시지를 분류하여 제공할 수 있다. 하지만 채널은 단순히 메시지를 분류하는 역할만 하기 때문에 채널별로 메시지를 다양한 형태로 변경해서 제공할 수 없다. Pushlet은 서블릿 기반으로 푸시 서비스를 제공하는 기술로서 서블릿 컨테이너를 지원하는 웹 서버와 쉽게 연동되며 푸시 서비스 개발이 용이하다. 하지만 메시지를 분류하여 제공하기 위한 채널과 같은 기능을 지원하지 않으며 필터링 기능 역시 없기 때문에 다양한 목적에 활용되기 위하여 손쉽게 활용되기에 많은 어려움이 따른다.

본 연구에서 확장, 개발된 BlazeDS는 기존 BlazeDS의 장점을 그대로 지원하면서 사용자의 다양한 요구를 충족시킬 수 있도록 메시지를 수정, 제공해줄 수 있는 메시지 필터링 기능을 지원한다. 메시지 필터링 기능은 기존의 단방향 정보 제공이 아닌 사용자의 요구와 선택에 따라 메시지를 변형하여 제공해줌으로써 효과적인 푸시 서비스 개발 환경을 제공한다. 또한, 필터 관리기능이 포함되어 실용성이 한층 증대되었다.

IV. 결 론

본 논문에서는 BlazeDS의 메시징 서비스를 바탕으로 효과적인 푸시 서비스를 개발할 수 있도록 BlazeDS에 메시지 필터링 기능을 추가하고 활용하는 방법에 대하여 기술하였다. 기존의 BlazeDS의 메시징 서비스는 사용자들이 등록된 채널만을 구별하여 일괄적으로 정보를 전송하기 때문에 각 사용자들의 요구에 부합하는 정보를 가공하여 제공할 수 없다. BlazeDS에 추가된 메시지 필터링 기능은 사용자가 선택한 필터에 따라 정보를 가공하며, 가공된 정보를 사용자별로 자동으로 전송할 수 있다. 이러한 기능을 추가하기 위하여 BlazeDS의 메시징 서비스 구조를 분석하여 이에 적합한 필터링 구조를 설계하였으며, 그에 따라 Core, Producer, Adapter, Filter 인터페이스들을 정의하였고 이를 구현하는 클래스들을 추가하였다. 그리고 BlazeDS에 추가된 메시지 필터링 기능을 활용하는 프로토타입으로서 뉴스 정보 푸시 어플리케이션을 개발하였다. 이 어플리케이션에서 사용자는 자신이 원하는 형태로 뉴스 정보를 제공받기 위하여 여러 필터를 선택할 수 있으며, 서버는 선택된 필터를 조합하여 뉴스 정보를 편집하여 사용자가 필요한 뉴스만을 제공한다. 또한, 필터를 손쉽게 추가, 삭제하고 관리하기 위하여 필터 관리도구가 추가되었으며, 이를 통하여 보다 능동적으로 다양한 상황에 메시지 필터링 기능을 활용할 수 있었다.

향후 연구로는 메시지 필터링 기능의 정보 표현 방법을 더욱 추상화하여 다양한 형태의 정보를 지원할 수 있도록 확장하는 것이다. 또한, 실시간 정보를 제공하는 스트리밍 서비스에 메시지 필터링 기능을 적용하는 방법에 대해서도 연구하도록 한다.

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

〔4〕*"http://ajaxpatterns.org/HTTP_Streaming,"* HTTP Streaming.

〔5〕 *"www.adobe.com,"* Adobe Systems Incorporated.

〔6〕 "*http://opensource.adobe.com/wiki/display/blazeds/BlazeDS,"* Adobe BlazeDS.

〔7〕 어세룡, "웹2.0을 위한 Ajax 플랫폼," 정보과학회지, 한국정보과학회, 제 26권 제 9호, 47-57쪽, 2008년.

〔8〕 *"http://en.wikipedia.org/wiki/Ajax,"* Ajax

〔9〕 *"http://www.ape-project.org/,"* APE

〔10〕 Aaron Newton, "MooTools Essentials: The Official MooTools Reference for JavaScript and Ajax Development," 1st Edition, Apress, 2008.

〔11〕 Cui Bin, "Realization of EPOLL-based Linux Online Games Server," CONTROL AND AUTOMATION, Weijisuanji Xinxi Zazhishe, Vol.172, pp. 64-66, 2006

〔12〕 Dave Crane, Phil McCarthy, *"Comet and Reverse Ajax: The Next-Generation Ajax 2.0,"* 1st Edition, Apress, 2008

〔13〕 *"http://en.wikipedia.org/wiki/Comet_(programming),"* Comet Programming

〔14〕 *"http://www.pushlets.com/,"* Pushlet

〔15〕 박찬, 유관희, "차세대 웹을 위한 RIA(Rich Internet Application) 기술의 현황 및 전망," 한국콘첸츠학회지, 한국콘텐츠학회, 제 6권, 제 4호, 10-14쪽, 2008년.

〔16〕 *"http://en.wikipedia.org/wiki/Rich_Internet_application,"* Rich Internet Application

〔17〕 *"http://www.adobeflex.co.kr/aboutflex/down/2.pdf,"* Adobe Flex 2

〔18〕 *"http://www.adobe.com/products/air/,"* Adobe AIR

〔19〕 *"http://www.adobe.com/products/livecycle/,"* Adobe Lifecyle Data Service

〔20〕 "Action Message Format — AMF3," Adobe Systems Inc.

저 자 소 개

이 홍 창

2006년 : 울산대학교 컴퓨터정보통신공학부 졸업(공학사)
2008년 : 울산대학교 컴퓨터정보통신공학부 졸업(석사)
2010년 : 울산대학교 컴퓨터정보통신공학부 박사 수료

김 보 현

2009년 : 울산대학교 컴퓨터정보통신공학부 졸업(공학사)
2009년~현재 : 울산대학교 컴퓨터정보통신공학부 석사과정

오 훈

1995년 : 텍사스A&M대학교 전산학 졸업(박사)
1996년 : 삼성전자 중앙연구소 수석연구원
2005년~현재 : 울산대학교 컴퓨터정보통신공학부 부교수

이 명 준

1991년 : 한국과학기술원 전산학과 졸업(박사)
1982년~현재 : 울산대학교 컴퓨터정보통신공학부 교수
1993년~1994년 : 미국 버지니아대학 교환 교수
2005년~2006년 : 미국 캘리포니아 주립대학 교환 교수