JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 5 : 1144~1150 May. 2015

다중 서버 구조에 의한 새로운 XMPP/SIP 프레즌스 서비스 시스템

이기수 · 장춘서*

A New XMPP/SIP Presence Service System by Multiple Servers Architecture

Ky-Soo Lee · Choonseo Jang*

Department of Computer Engineering, Kumoh Institute of Technology, Gumi 730-701, Korea

요 약

프레즌스(Presence) 정보는 사용자의 온라인 상태, 현재 위치, 네트워크 접속 방식 및 접속 주소 등 사용자에 관한 다양한 내용을 제공하며 XMPP(Extensible Massaging and Presence Protocol) 기반 프레즌스 정보와 SIP(Session Initiation Protocol) 기반 프레즌스 정보의 두 종류가 사용된다, 본 논문에서는 이 두 종류의 프레즌스 정보를 모두 처리 할 수 있는 XMPP/SIP 프레즌스 서비스 시스템에서 서버 부하 증가에 따른 확장성의 제한을 해결하기 위하여 사용자 수의 증가에 따라 동적으로 서버가 추가되고 각 서버의 부하를 효율적으로 제어 할 수 있는 새로운 다중 서버 구조 방식을 제안하였다. 이 방식에서는 부하 제어를 위한 새로운 XMPP 스탠자 구조와 SIP 프레즌스 정보 데이터 구조가 설계되었고, 동적 서버 제어를 위한 각 서버와 사용자 사이의 메시지 교환 절차가 제시되었다. 제안된 시스템의 성능은 시뮬레이션 실험을 통하여 분석하였다.

ABSTRACT

Presence information provides various informations about users such as on-line status, current location, network connection method and connection address, and there are two kinds of presence information, SIP(Session Initiation Protocol) based presence information and XMPP(Extensible Massaging and Presence Protocol) based presence information. In this paper, a multiple server architecture that can handle these two kinds of presence information has been proposed. In this architecture, severs are added dynamically according to number of users to provide system scalability, and load of each server can be effectively controlled. In this system, a new XMPP stanza architecture and presence information data format are designed for load control. Furthermore message exchanging procedures between servers and users for dynamic server control has been also suggested. The performance of the proposed system has been analysed by simulation.

키워드 : 프레즌스 서비스, XMPP, SIP, 다중 서버

Key word : Presence Service, XMPP, SIP, Multiple Servers

Received 18 March 2015, **Revised** 06 April 2015, **Accepted** 21 April 2015

*** Corresponding Author** Choonseo Jang(E-mail:csjang@kumoh.ac.kr, Tel:+82-54-478-7521)
Department of Computer Engineering, Kumoh Inst. of Tech., Gumi 730-701, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2015.19.5.1144 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

프레즌스 서비스 시스템에서 서버는 사용자들이 프레즌스 자원들에 대한 등록을 위해 보내온 등록 요청 메시지를 처리해야 하고, 등록된 프레즌스 자원들에서 프레즌스 정보의 상태 변화가 있을 때 발생하는 프레즌스 이벤트 통지도 처리해야 한다. 현재 XML 요소를 스트림화 하기위한 프로토콜인 XMPP 기반의 프레즌스 서비스[1,2]와 기존 SIP 기반의 프레즌스 서비스[3,4]가 모두 사용되고 있고 모바일 환경에서 각 사용자들이 요구하는 프레즌스 자원의 수는 크게 증가하고 있다. 따라서 사용자 수가 증가 할수록 프레즌스 서버의 부하는 급격히 커지게 되어 시스템 확장성을 제한하는 요소로 작용하며 서버의 부하를 효율적으로 분산 처리 할 수 있는 구조가 필요하다.

이를 위하여 본 논문에서는 XMPP/SIP 프레즌스 정보 처리에 따른 서버의 부하 레벨이 증가하면 동적으로 서버가 추가되고 각 서버의 부하를 효율적으로 제어할 수 있는 다중 서버 구조에 의한 프레즌스 서비스 시스템을 새롭게 제시하였다. 이 구조에서 각 서버들은 현재 부하 레벨을 실시간으로 교환하며 특정 서버의 부하 레벨이 지정된 값을 초과하게 되면 주 서버는 동작 중인 서버 중 가장 부하가 적은 서버를 선택하여 이 서버가 XMPP/SIP 프레즌스 서비스를 분산 처리하도록 한다.

만일 동작 중인 모든 서버들의 부하 레벨이 지정된 값에 이르러 추가로 부하를 처리할 상황이 아닌 경우에는 동적으로 새로운 서버를 생성하여 부하를 분산 처리한다. 이와 같은 동작을 위하여 본 연구에서는 시스템 부하의 분산 처리에 필요한 요소들이 추가된 새로운 프레즌스 정보 데이터 구조와 XMPP IQ 스탠자 구조를 설계하였고 서버 사이의 메시지 교환 절차가 제시되었다. 이와 같이 제안된 방식에 의하여 전체 시스템의 부하를 적절히 제어 할 수 있으며 시스템의 확장성도 높일 수 있게 된다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련 연구로서 SIP 프레즌스 정보 데이터 구조 및 XMPP와 관련된 기존 연구에 대해 설명하고 Ⅲ장에서는 본 논문에서 새롭게 제안하는 다중 서버 구조의 XMPP/SIP 프레즌스 서비스 시스템의 설계 및 동작을 설명하고 이어서 시스템 부하의 분산 처리에 필요한 요소들이 추가된 새로운 프레즌스 정보 데이터 구조와 서버 사이의 메시지 교환 절차에 대해 설명한다. 다음 Ⅳ장에서는 구현된 시스템의 성능 분석을 한 후 Ⅴ장에서 결론을 맺는다.

Ⅱ. 관련연구

프레즌스 정보 데이터의 기본 구조는 PIDF(Presence Information Data Format)이고 MIME 형태는 'application/pidf+xml'이며 XML 기반의 문서이다[5]. 프레즌스 정보 데이터의 최상위 요소는 <presence>이고 각 프레즌스 정보를 구분하기 위한 하위 요소인 복수개의 <tuple>이 있다. 각 <tuple>은 하위 요소로 각 사용자의 현재 프레즌스 정보를 나타내는 <status>를 가진다. 프레즌스 정보 데이터는 이외에도 여러 개의 추가적인 정보를 나타내는 하위 요소 및 속성을 가질 수 있다. 이와 같은 프레즌스 정보 데이터의 예를 그림 1에 보였다.

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
      entity="sip:customer3215@pres.svr.com">
  <tuple id="b76gjstrj">
    <status>
      <basic>open</basic>
    </status>
    <contact>sip:david23@example.com</contact>
  <tuple id="r8tujjs">
    <status>
      <basic>closed</basic>
    </status>
    <contact>sip:coam451@transatl.com</contact>
</presence>
```

그림 1. 프레즌스 정보 데이터 예
Fig. 1 An Example of Presence Information Data

프레즌스 서버는 이와 같은 프레즌스 정보 데이터를 사용자에게 제공하며 사용자는 프레즌스 서버에게 등록할 프레즌스 자원의 SIP 주소를 담은 SUBSCRIBE 메시지를 사용하여 필요한 프레즌스 서비스 등록을 한다. 프레즌스 정보의 변화가 발생하면 SIP PUBLISH 메시지를 사용해 프레즌스 서버에게 알리고 프레즌스 서버는 SIP NOTIFY 메시지를 사용해 해당 사용자에게 이를 통지한다.

이때 프레즌스 서비스 사용자 수의 증가에 따라 프레

즌스 리소스에 대한 처리와 프레즌스 정보 데이터의 통지 및 리프레시 양도 크게 증가하여 프레즌스 서버의 부하를 높이게 된다. XMPP는 XML 요소를 스트림화 하기위한 프로토콜이며 메시지나 프레즌스 정보를 실시간에 가까운 속도로 교환하기 위한 목적으로 사용된다[6,7]. XML 스트림 내에서 전송되는 XML 요소들인 XMPP 스탠자는 세 가지 종류가 있으며 이들은 각각 메시지 스탠자(<message>), 프레즌스 스탠자(<presence>) 및 IQ 스탠자(<iq>)이다. 이들 XML 스탠자를 담고 있는 컨테이너인 XML 스트림은 <stream> 태그로 시작하며 그림 2에 XML 스탠자들이 XML 스트림 내에 위치하는 형태를 보였다.

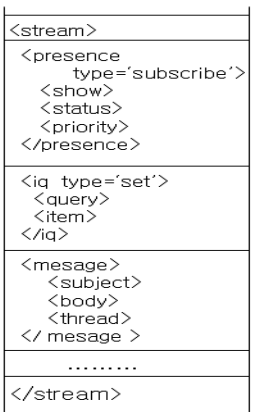

그림 2. XML 스트림 예
Fig. 2 An Example of XML Stream

현재까지 프레즌스 서버의 부하를 줄이기 위한 연구로는 프레즌스 정보 문서의 크기를 줄이기 위한 부분 통지(partial notification) 방식이나 프레즌스 리소스를 등록하는데 걸리는 시간을 줄이는 방식이 제안되고 있다[8-10]. 또는 자주 사용되는 문서 태그 부분을 압축하는 방식과 XML 스트림 내에서 교환되는 XMPP 메시지 양을 줄여 서버의 부하와 네트워크 트래픽을 감소시키는 방안도 제시되고 있다[11]. 그러나 이와 같은 프레즌스 서비스 시스템은 XMPP를 지원하지 않거나 XMPP를 지원하는 시스템의 경우 단일 프레즌스 서버를 사용한다는 제약이 있으며 따라서 XMPP와 SIP 프레즌스 서비스를 모두 지원하는 대용량 프레즌스 서비스 시스템으로의 확장이 어렵다는 문제점이 있다. 본 연구에서는 사용자 수의 증가에 따라 동적으로 XMPP/SIP 프레즌스 서버가 추가되고 각 프레즌스 서버의 부하를 효율적으로 제어 할 수 있는 다중 XMPP/SIP 프레즌스 서버 구조를 제안하여 대용량 XMPP/SIP 프레즌스 서비스 시스템의 구성이 가능하도록 하였다.

Ⅲ. 시스템 설계 및 구현

3.1 다중 서버 구조 설계

본 논문에서 설계한 다중 서버 구조의 XMPP/SIP 프레즌스 서비스 시스템을 그림 3에 보였다. 여기서 각각의 프레즌스 서버는 부하제어 모듈, XMPP/SIP 변환모듈, XMPP/SIP 프레즌스 정보 데이터베이스, XMPP 프레즌스 처리모듈, XMPP 스트림 처리 모듈, SIP 프레즌스 처리모듈 및 SIP 메시지 처리모듈로 구성된다.

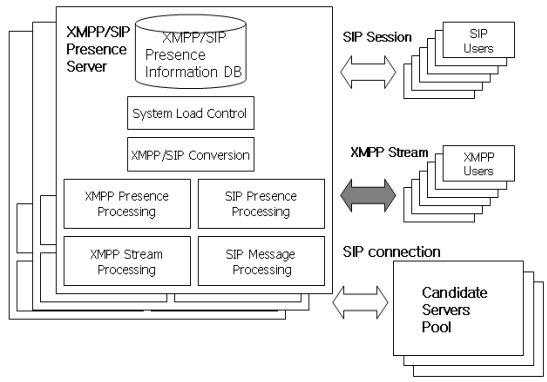

그림 3. 다중 서버 구조의 XMPP/SIP 프렌즌스 서비스 시스템
Fig. 3 XMPP/SIP Presence Service System with Multiple Servers Architecture

부하제어 모듈은 각 서버의 부하 레벨을 측정하고 동작 중인 서버들과 부하 상태에 관한 정보를 실시간으로 교환하며 사용자의 할당 등 전체 시스템의 부하를 분산 처리하는 가능을 제공한다. 여기서 XMPP/SIP 프레즌스 정보 데이터베이스는 본 논문에서 설계한 부하 제어 기능이 추가된 새로운 프레즌스 정보 데이터 구조를 사용하여 설계되었다. 새로운 사용자가 주 프레즌스 서버에게 프레즌스 정보 등록 요청을 하면 3.2절에 나오는 현재 동작 중인 각 서버들의 부하 레벨을 나타내는 프레즌스 정보 요소인 <svr-loadlevel-list>의 값을 조사하여 가장 부하 레벨이 낮은 서버를 선택한다.

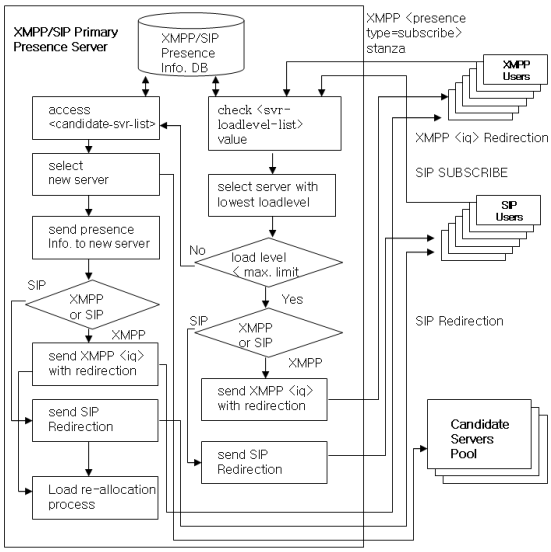

그림 4. 다중 서버 구조 프레즌스 시스템 동작과정 흐름
Fig. 4 Operation Procedure Flow of Presence System with Multiple Server Architecture

새로운 사용자가 XMPP 프레즌스 정보를 사용하는 경우 XMPP IQ 스탠자 <iq>의 하위요소 <redirection>에 이 선택된 서버의 URI를 넣어 보내고, SIP 프레즌스 정보를 사용하는 경우 상태 코드 번호가 302인 SIP 메시지에 선택된 서버의 URI를 넣어 보내어 이후 사용자가 이 선택된 서버와 프레즌스 메시지를 교환하도록 한다.

모든 서버의 부하 레벨이 정해진 기준치에 이르러 새로운 프레즌스 서버를 추가해야 할 경우 주 프레즌스 서버는 3.2절에 나오는 선택 가능한 새로운 프레즌스 서버들의 목록을 가지고 있는 프레즌스 정보 요소인 <candidate-svr-list>에서 새로운 프레즌스 서버를 선택한다. 다음 사용자의 프레즌스 정보 형태에 따라 이 서버의 URI를 담은 XMPP IQ 스탠자나 SIP 302 메시지를 해당 사용자에 보내어 이후 사용자가 이 선택된 서버와 프레즌스 메시지를 교환하도록 한다. 다음 기존 서버들의 부하를 이 새로운 프레즌스 서버에게 분산시키기 위하여 부하 레벨이 가장 높은 서버 순으로 해당 서버가 처리하는 사용자를 새로운 프레즌스 서버에게 할당한다. 그림 4에 설계된 다중 서버 구조의 프레즌스 시스템의 동작 과정 흐름을 보였다. 이 그림에서는 사용자가 주 프레즌스 서버에게 SIP SUBSCRIBE 메시지나 XMPP <presence> 스탠자를 보내어 프레즌스 정보 등록 요청을 한 경우 전체 시스템 부하 상태에 따라 가장 부하 레벨이 낮은 서버를 선택하거나 새로운 서버를 추가하는 과정이 제시되어 있다.

XMPP 프레즌스 처리모듈은 각 사용자들의 프레즌스 리소스 목록을 XMPP/SIP 프레즌스 정보 데이터베이스에 생성, 수정 및 관리하고 XMPP 스트림 내의 프레즌스 및 IQ 스탠자를 처리하여 프레즌스 정보의 등록과 통지기능을 제어한다. XMPP 스트림 처리 모듈은 스트림 헤더를 처리하고 스트림 인증 및 리소스 바인딩을 하며 XMPP 사용자와 프레즌스 서버사이의 스트림 생성 및 유지를 담당한다. SIP 프레즌스 처리모듈은 SIP 프레즌스 정보의 등록과 등록된 SIP 프레즌스 정보의 변화가 발생했을 경우 SIP 통지기능을 담당한다. SIP 메시지 처리모듈은 SIP 프로토콜 스택을 가지며 SIP 기반의 프레즌스 서비스 사용자와 프레즌스 서버 사이의 메시지 교환 및 연결 제어를 담당한다.

3.2. 프레즌스 정보 데이터 구조 설계

본 논문에서는 다중 서버 구조의 프레즌스 시스템을 구현하기 위하여 다음과 같은 필요 요소들이 추가된 프레즌스 정보 데이터 구조가 설계되었다. 먼저 서버의 부하 제어를 위하여 최상위 요소 <presence>의 하위 요소로 <svr-loadlevel-list>가 새롭게 추가되었다. <svr-loadlevel-list>의 하위 요소로는 각 서버의 부하 레벨과 최대 부하 허용값을 가지고 있는 <svr-load>를 두었다. 부하 레벨은 각 서버에서 프레즌스 서비스 처리를 위한 부하 지수를 나타내며 각 서버가 담당하는 현재 사용자 수와 등록된 프레즌스 리소스의 합을 기준으로 산출한다. <svr-load> 요소는 속성으로 각 서버를 구분하기위한 'svr-id'를 가지며 하위 요소로 현재 부하 레벨인 <current-loadlevel>과 최대 부하 허용값을 나타내는 <max-loadlevel>이 설계되었다.

다음으로 새롭게 프레즌스 시스템에 추가 될 수 있는 서버들을 나타내기위한 요소인 <candidate-svr-list>가 설계되었다. 이 요소는 하위 요소로 <svr-uri>를 가지며 실제 서버들의 URI를 요소 컨텐츠로 가진다. 현재 동작중인 전체 프레즌스 서버의 개수를 나타내기 위한 요소인 <active-svr-number>도 추가되었다.

다음 각 서버가 현재 담당하고 있는 사용자 리스트인 <current-users-list> 요소와 이의 속성으로 각 프레즌스 서버를 구분하기위한 'server-id'를 설계하였다. 이 요소

는 하위요소로 해당 프레즌스 서버가 현재 처리하고 있는 사용자수를 표시하는 <current-users-number>와 각 사용자에 대한 정보를 나타내는 <info-users>를 가진다. <info-users> 요소는 속성으로 각 사용자를 구분하기 위한 'user-id'를 가지며 하위요소로 XMPP와 SIP를 구분하기위한 <pres-type>과 등록된 프레즌스 리소스 수를 나타내는 <resource-number> 및 각 사용자에 대한 프레즌스 정보를 가지고 있는 <pres-info>를 가지도록 설계되었다. 또 전체 프레즌스 서비스 사용자 수를 나타내는 요소인 <total-users-number>도 추가되었다. 그림 5에 이와 같이 설계된 프레즌스 정보 데이터 구조를 사용한 XMPP/SIP 프레즌스 시스템에서의 메시지 교환 절차를 보였다.

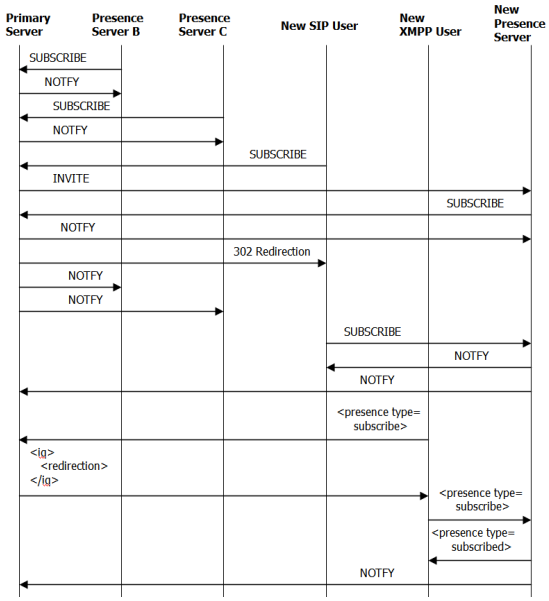

그림 5. XMPP/SIP 프레즌스 시스템에서의 메시지 교환 절차
Fig. 5 Message Exchange Procedure of XMPP/SIP Presence System

이 그림에서는 각 프레즌스 서버들이 SIP SUBSCRIBE 메시지와 NOTIFY 메시지를 사용하여 본 논문에서 설계한 구조를 갖는 프레즌스 정보 데이터를 교환한다. 새로운 SIP 프레즌스 서비스 사용자가 프레즌스 서비스의 대표 주소를 가지고 있는 주 프레즌스 서버에 SUBSCRIBE 메시지를 사용하여 프레즌스 리소스 등록을 시도하면 주 프레즌스 서버는 프레즌스 정보 데이터베이스를 통하여 현재 동작 중인 각 서버의 부하 레벨을 조사한다. 새로운 프레즌스 서버를 추가해야 할 경우 <candidate-svr-list> 요소에서 서버를 선택 후 이 서버에게 INVITE 메시지를 보내어 프레즌스 시스템에 참가 시킨다. 이 새로운 서버는 SUBSCRIBE와 NOTIFY 메시지를 사용해 주 프레즌스 서버로부터 현재의 프레즌스 정보 데이터를 얻는다. 다음 프레즌스 리소스 등록을 요청했던 사용자에게 Redirection SIP 메시지를 보내어 새로운 프레즌스 서버와 서비스 연결을 맺도록 한다. 새로운 XMPP 프레즌스 서비스 사용자가 <presence type=subscribe> 스탠자를 사용해 프레즌스 리소스 등록을 요청하는 경우 주 프렌즌스 서버는 <redirect> 요소를 가진 <iq> 스탠자를 사용해 새로운 프레즌스 서버와 연결을 맺도록 하며 이후 이 서버는 NOTIFY 메시지를 사용하여 프레즌스 정보 데이터의 변화를 통지한다.

Ⅳ. 성능 분석

본 논문에서 제안한 다중 서버 구조의 XMPP/SIP 프레즌스 서비스 시스템의 성능을 시뮬레이션으로 분석하였다. 이를 위하여 사용자 수를 증가시켜가며 본 연구에서 제안한 다중 서버 구조의 경우와 기존 단일 서버 구조의 경우에서의 프레즌스 서버에서의 프레즌스 등록 처리 시간을 비교 측정하였다. 이때 다중 서버 구조의 경우 프레즌스 시스템 정책으로 최대 부하 허용값을 나타내는 프레즌스 정보 데이터 요소인 <max-loadlevel> 값을 100으로 하여 처음에 하나의 프레즌스 서버로 동작하다가 사용자가 증가하여 동작 중 최대 부하 허용값이 이를 초과하면 새로운 프레즌스 서버가 추가되도록 하였다. 전체 사용자 중 XMPP와 SIP 비율은 각각 50% 씩으로 하였고 각 사용자는 20개의 프레즌스 리소스를 가지며 1초에 1회의 프레즌스 등록 요청을 하는 것으로 설정하였다. 그림 6에 이의 결과를 보였다.

여기서 제안된 다중 서버 방식을 사용하는 경우 사용자 수가 100명에서는 하나의 서버만을 사용하므로 기존 방식과 동일한 성능을 보인다. 100명을 초과하는 경우 서버가 추가되어 다중 서버 방식의 경우 처리 시간의 증가가 거의 없음을 보여준다. 기존 방식과 비교해 200명인 경우 평균 처리 시간이 45.6% 감소하고, 300명 일 때는 63.3% , 500명 일 때는 72.9%가 감소하

여 사용자가 수가 증가할수록 본 논문에서 제안한 방식의 개선 효과가 커짐을 알 수 있다.

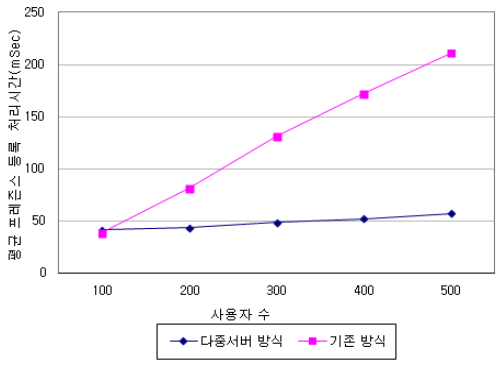

그림 6. 평균 프레즌스 등록 처리시간
Fig. 6 Average Presence Subscribe Processing Time

다음 최대 부하 허용값 <max-loadlevel> 값을 200으로 하여 사용자 수를 증가시켜가며 프레즌스 서버에서의 평균 등록 처리 시간을 비교 측정하였다. 다른 조건은 그림 6의 경우와 동일하다. 그림 7에 이의 결과를 보였다.

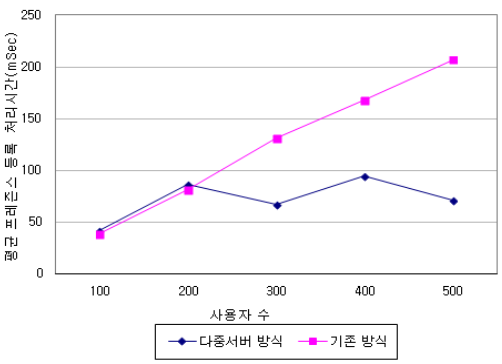

그림 7. 평균 프레즌스 등록 처리 시간(최대 부하 허용값=200)
Fig. 7 Average Presence Subscribe Processing Time(Max. Load Level=200)

여기서는 최대 부하 허용값의 증가로 그림 6 경우와는 달리 사용자 수가 200명 일 때까지는 1개의 서버만을 사용하다가 이를 초과하면 프레즌스 서버가 새롭게 추가되고 400명을 초과하면 다시 서버가 추가되는 식으로 동작한다. 측정 결과 300명에서 48.8%, 400명에서 44.0%, 500명에서 65.7% 만큼 감소하여 역시 처리 시간이 기존 방식과 비교해 크게 감소함을 보여준다. 이와 같이 본 연구에서는 프레즌스 정책으로 최대 부하 허용값을 상황에 따라 다르게 설정하여 각 서버가 담당하는 사용자 수를 제어 할 수 있다.

다음 그림 8은 프레즌스 서버에서 통지 메시지를 처리하는데 걸리는 시간을 기존 방식과 서로 비교한 결과이다. 이때 프레즌스 통지 메시지의 크기는 50Kbyte로 하였고 각 사용자 당 20개씩의 프레즌스 리소스에서 초당 1회의 프레즌스 정보가 생성되도록 하였다. 최대 부하 허용값 <max-loadlevel>는 100으로 하였다. 여기서는 다중 서버 방식의 경우 사용자 수가 100명을 초과하면 프레즌스 서버가 추가되어 처리 시간의 증가가 거의 없는 반면 기존 방식은 크게 증가하는 것을 보여준다. 기존 방식과 비교하여 제안된 방식에서는 사용자 수 200명에서 43.1%, 사용자 수 300명에서 58.6%, 사용자 수 500명에서 69.4% 까지 단축시킬 수 있음을 보여준다.

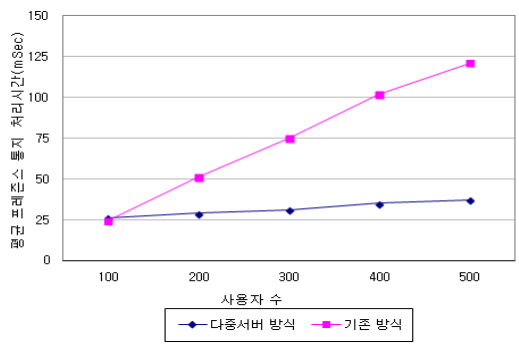

그림 8. 평균 프레즌스 통지 처리 시간
Fig. 8 Average Presence Notification Processing Time

Ⅴ. 결 론

본 논문에서는 XMPP/SIP 프레즌스 서비스 시스템에서 서버의 부하 레벨이 증가하면 동적으로 서버가 추가되고 각 서버의 부하를 효율적으로 제어할 수 있는 다중 서버 구조를 새롭게 제시하였다. 이 구조에서 특정 서버의 부하 레벨이 지정된 값을 초과하면 주 서버는 동작 중인 서버 중 가장 부하가 적은 서버를 선택하여 이 서버가 XMPP/SIP 프레즌스 서비스를 분산 처리하도록 하고 만일 동작 중인 모든 서버들의 부하 레벨이 지정된 값까지 상승하면 동적으로 새로운 서버를 생성하여 부하를 분

산 처리한다. 이를 위하여 본 연구에서는 시스템 부하의 분산 처리에 필요한 요소들이 추가된 새로운 프레즌스 정보 데이터 구조와 XMPP IQ 스탠자 구조를 설계하였고 서버 사이의 메시지 교환 절차가 제시되었다. 제안된 시스템의 성능 측정 결과 본 연구에서 제안한 다중 서버 구조를 적용한 경우 서버에서의 평균 프레즌스 등록 처리 시간이 44.0%에서 72.9%까지 감소하였고 평균 프레즌스 통지 처리 시간은 43.1%에서 69.4%까지 감소하여 사용자 수가 증가할수록 제안한 방식의 개선 효과가 커짐을 보여 주었다. 향후 과제로는 모바일 환경에서 사용 대역폭을 최소화 할 수 있는 XMPP/SIP 프레즌스 서비스 시스템을 연구 할 필요가 있다.

감사의 글

본 연구는 금오공과대학교 학술연구비에 의하여 연구된 논문입니다.

REFERENCES

[1] Ronny Klauck and Michael Kirsche, "Combining Mobile XMPP Entities and Cloud Services for Collaborative Post-Disaster Management in Hybrid Network Environments," *Mobile Networks and Applications,* Volume 18 Issue 2, pp. 253-270, April 2013.

[2] Huber Flores and Satish Srirama, "Mobile cloud messaging supported by XMPP primitives," *MCS '13: Proceeding of the fourth ACM workshop on Mobile cloud computing and services*, June 2013.

[3] Arup Acharya et al, "Pressentials: a flexible middleware for presence-enabled applications," *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications*, August 2011.

[4] Robin Acker et a1, "Ubiquitous home control based on SIP and presence service," *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, November 2010.

[5] H. Sugano and G. Klyne, “Presence Information Data Format (PIDF),” RFC 3863, August 2004.

[6] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP):Instant Messaging and Presence," RFC 6121, March 2011.

[7] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core,"RFC 6120, March 2011.

[8] M. Lonnfors, E. Leppanen, H. Khartabil and J. Urpalainen, “SIP Extension for Partial Notification of Presence Information", RFC 6261, September 2011.

[9] A. Niemi, M. Lonnfors and E. Leppanen, "Publication of Partial Presence Information", RFC 5264, September 2008.

[10] G. Camarillo, A.B. Roach and O. Levin, "Subscriptions to Request-Contained Resource Lists in the Session Initiation Protocol (SIP)," RFC 5367, October 2009.

[11] C. Jang, "XMPP/SIP Presence Service System using Efficient Message Control Method," *Journal of Korea Inst. of. Information and Communication Eng*., vol. 13, no. 11, pp. 2678-2683, November 2014.

이기수(Ky-Soo Lee)

1982년 2월 : 서울대학교 공학석사
현재 : 금오공과대학교 컴퓨터공학과 교수
※관심분야 : SIP, 데이터베이스

장춘서(Choonseo Jang)

1993년 8월 : 한국과학기술원 공학박사
현재 : 금오공과대학교 컴퓨터공학과 교수
※관심분야 : SIP, 임베디드 시스템, 인터넷텔레포니