부분 Publication 및 Notification에 기반한 SIP 프리젠스 서비스 구현

김원식* 조현규* 장춘서*
*금오공과대학교 컴퓨터공학과
e-mail:{plusw, blackjo, csjang}@kumoh.ac.kr

Implementation of SIP Presence Service based on Partial Publication and Notification

Won Sik Kim* Hyun Gyu Jo* Choon Seo Jang*
*Dept. of Computer Engineering, Kumoh National Institute of Technology

요 약

프리젠스(Presence) 서비스는 SIP(Session Initiation Protocol) 확장 서비스의 하나로서 사용자들 사이에 현재 위치의 정보나 온라인 정보 등과 같은 통신 상태의 변화를 등록(Subscription)과 Notification을 통해서 제공한다. 이때 프리젠스 정보가 변화될 때마다 전체 프리젠스 문서를 이용하여 업데이트하는 기존의 Publication 및 Notification 방식을 사용할 경우 프리젠스 정보 전송에 상당한 오버헤드가 발생한다. 이를 해결하기 위해 본 논문에서는 부분 Publication 및 Notification 방식을 사용하여 변화되는 최신 프리젠스 정보만으로 구성된 부분 프리젠스 문서를 메시지 바디에 포함시켜 전달하도록 함으로써 기존의 방식에 비해 효율적인 프리젠스 관련 정보 전송이 가능한 새로운 프리젠스 서비스를 구현하였다.

1. 서론

프리젠스 서비스는 SIP를 이용한 확장 서비스의 하나로 네트워크상에서 변화되는 통신 상태나 위치 변화 등의 최신 프리젠스 정보를 사용자들에게 제공하는 서비스로서 VoIP, 인스턴트 메시징 서비스 등 사용자의 상태정보를 필요로 하는 여러 응용 서비스와 연계하여 폭 넓게 활용할 수 있는 유용한 기능이다[1].

프리젠스 서비스에서의 Publication은 사용자가 PUBLISH 메시지를 사용하여 자신의 현재 상태나 정보들을 서버에 알리는 것이고 Notification은 와처(Watcher)들에게 NOTIFY 메시지를 통하여 프리젠스 정보를 전달하는 것을 의미한다[2][3].

이때 기존의 프리젠스 서비스에서는 프리젠스 정보가 이전에 업데이트된 이후 얼마나 변화되었는지 상관없이 전체 프리젠스 정보를 전달하는 방식을 사용한다. 그에 따라 프리젠스 정보 전송 시 네트워크상에 불필요한 오버헤드를 감수해야 했다.

따라서 본 논문에서는 이러한 문제점을 해결하기 위해 부분 Publication 방식과 부분 Notification[4],[5] 방식을 사용하여 프리젠스 정보에 변화가 있을 경우 변화되는 프리젠스 정보만으로 구성된 부분 프리젠스 문서를 메시지 바디에 포함시켜 전달하는 방법을 이용함으로써 기존의 방식에 비해 효율적인 프리젠스 관련 정보 전송이 가능한 새로운 프리젠스 서비스를 구현하였다.

본 논문의 구성은 서론에 이어 2장에서 부분 Publication과 부분 Notification에 관하여 설명하고 3장에서는 구현된 시스템의 전반적인 내용을 기술하며 4장에서 결론을 맺는다.

2. 관련 연구

2.1 프리젠스 서비스의 구성 요소와 동작

프리젠스 서비스를 제공하기 위한 기본 구성 요소는 PUBLISH 메시지를 생성하고 이를 전송하는 PUA(Presence User Agents)와 받은 요청을 처리하는 PA(Presence Agent), 그리고 처리된 내용의 알림 대상인 와처로 구성된다.

기본적인 동작은 PUA가 사용자의 프리젠스 정보가 담긴 프리젠스 문서를 PUBLISH 메시지의 바디에 포함시켜 PA에 보내면, PA는 요청받은 PUBLISH 메시지의 헤더정보, 바디유무, 사용자의 유효 시간을 비교하여 현재의 이벤트 상태를 판별한 이후 등록된 와처들에게 처리된 데이터가 포함된 NOTIFY 메시지를 전달한다.

프리젠스 문서를 위한 기본 포맷은 PIDF(Presence Information Data Format)[6]를 사용하고, PIDF는 XML(Extensible Markup Language) 문서를 사용한다.

2.2 부분 Publication 및 Notification

기존의 Publication과 Notification 방식에서는 application/pidf+xml 포맷의 프리젠스 문서를 사용하여 일관된 상태로 유지되었다. 따라서 프리젠스 정보가 이전에 업데이트된 이후 약간의 변화가 생기더라도 PUBLISH와 NOTIFY 메시지의 바디에는 전체 프리젠스 문서를 포함시켜 보냈다. 그에 따른 문제점으로 통신 환경에 따른 낮은 대역폭과 많은 사용자 수에 따른 빈번한 링크들로 인해 상당한 오버헤드가 발생될 수 있었다.

이러한 문제점을 해결하는 방법은 기존에 프리젠스 정보의 변경 시에도 일관된 포맷으로 사용되었던 프리젠스 문서를 application/pidf-diff+xml 포맷[7]에 따라 프리젠스 정보의 변화되는 데이터만으로 이루어진 부분 프리젠스 문서로 바꾸는 것이다. 이 방법을 이용한 부분 Publication 방식과 부분 Notification 방식을 사용하여 부분 프리젠스 문서를 전송함으로써 기존의 방식에 비해 효율적인 프리젠스 정보 전송이 가능하게 하였다.

표1. 이벤트 상태 판별 Operations

Operation	Content-Length	SIP-If-Match	Expires Value	Presence Document
Initial	>0	no	>0	yes (전체)
Refresh	0	yes	>0	no
Modify	>0	yes	>0	yes (부분)
Remove	0	yes	0	no

표1은 SIP 메시지 헤더의 Content-Length와 SIP-If-Match 그리고 Expires 값을 이용해서 각각의 Operation 상태를 판별하여 메시지 바디의 포함 여부와 메시지 바디를 전체 프리젠스 문서를 사용할지 아니면, 부분 프리젠스 문서를 사용할지 여부를 알려주게 된다.

또한 전체 프리젠스 문서와 부분 프리젠스 문서의 각각에는 전송되는 메시지가 손실 없이 프리젠스 정보를 제대로 전달했는지 확인하기 위해 프리젠스 문서 내에 버전(version) 카운트를 두어 관리한다.

3. 시스템 구현

3.1 시스템의 구성

프리젠스 서버는 프록시(Proxy) 서버와 로케이션 서버(Location Server), PUBLISH 메시지를 처리하여 NOTIFY 메시지를 생성하기 위한 PA로 구성하였으며, 모든 사용자의 정보 및 상태 관리를 용이하게 하기 위해서 테이블 PST(Presence Subscription Table)와 테이블 PUST(Presence User State Table)를 내부적으로 두었다. PST 테이블은 사용자 등록을 위해, PUST 테이블은 현재 로그인 되어 있는 사용자들의 현재상태, 연결주소 및 서비스 유효 시간 등을 저장 또는 갱신하기 위해 사용하였다.

PUA는 PUBLISH 메시지를 생성하고 NOTIFY 메시지를 처리하여 적용하는 기능을 수행하도록 구현하였고, 시스템 내에 필요한 보안은 HTTP 다이제스트 인증을 사용하였다.

3.2 프리젠스 이벤트 상태와 문서의 판별

프리젠스 서버는 일반적인 SIP 요청 메시지를 받으면 프록시 서버로 동작하며 SUBSCRIBE 요청 메시지를 받으면 사용자 인증을 거쳐 등록 작업을 수행하고, PUBLISH 메시지를 수신하면 그림1과 같은 메시지 처리 흐름으로 동작한다.

먼저 PUBLISH 메시지를 받으면 PA는 SIP-If-Match 헤더의 포함 여부를 확인하여 이 헤더를 포함하지 않으면, 임의의 값으로 SIP-ETag값을 생성하고, 테이블 PUST에 값을 저장한다. 또한 Initial 상태이므로 전체 프리젠스 문서를 분석하여 NOTIFY 메시지의 바디에 실어 와처들에게 알린다.

만약 SIP-If-Match 헤더가 포함되어 있으면 테이블에 저장되어 있던 SIP-ETag값과 SIP-If-Match 헤더 값을 비교하여, 두 값들이 일치하지 않으면 412(Precondition Failed) 응답코드를 보내고 세션을 종료한다. 그러나 헤더 값이 동일하면 응답 메시지를 위한 SIP-ETag값을 생성하며 테이블 PUST에서 해당 사용자의 SIP-ETag값을 갱신한다.

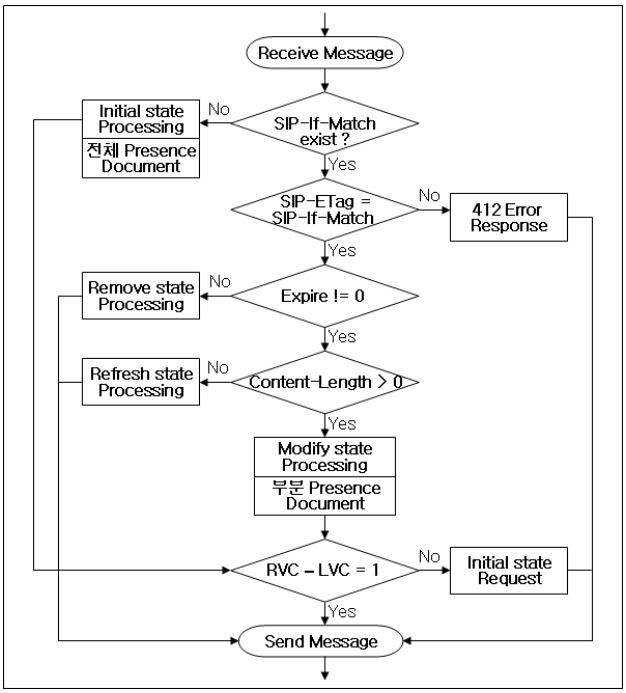


그림 1. 프리젠스 상태 판별의 처리 흐름도

다음으로 Expire 헤더 값을 비교하여 그 값이 0이면 Remove 상태로 사용자가 세션을 종료하기 위한 메시지를 생성한다. 또한 테이블 PUST에서 해당 사용자의 튜플을 삭제 후 그 사용자의 세션이 종료되었음을 와처들에게 알린다.

다음 단계로 Content-Length가 0인 경우이면 Refresh 상태로 사용자의 유효시간을 갱신할 필요가 있을 때 사용한다. Refresh 상태이면 와처들에게 NOTIFY 메시지를 보내지 않고, 테이블 PUST에서 해당 사용자의 Expire값을 3600으로 갱신한다.

만약 Content-Length가 0보다 크다면 포함된 부분 프리젠스 문서로 현재 사용자의 상태를 판별하는 Modify 상태이다. Modify 상태가 되면 PUBLISH 메시지에 포함된 사용자의 현재 상태를 테이블 PUST의 상태정보에 갱신한 후 NOTIFY 메시지의 바디에 부분 프리젠스 문서를 포함시켜 와처들에게 알린다.

마지막 단계로 프리젠스 문서의 Received Version Count (RVC)값이 테이블 PUST의 Local Version Count (LVC)보다 단지 1이 더 크다면 패킷의 손실 없이 사용자들의 상태정보를 제대로 전달한 상태가 되고, 그렇지 않다면 패킷 전달에 문제가 있었다고 보고 지금까지 적용되지 않은 프리젠스 정보를 위해 Initial 상태의 전체 프리젠스 정보를 요청하게 된다.

다음 그림 2는 부분 Publication 및 Notification을 사용하여 세션을 설정한 예이다.

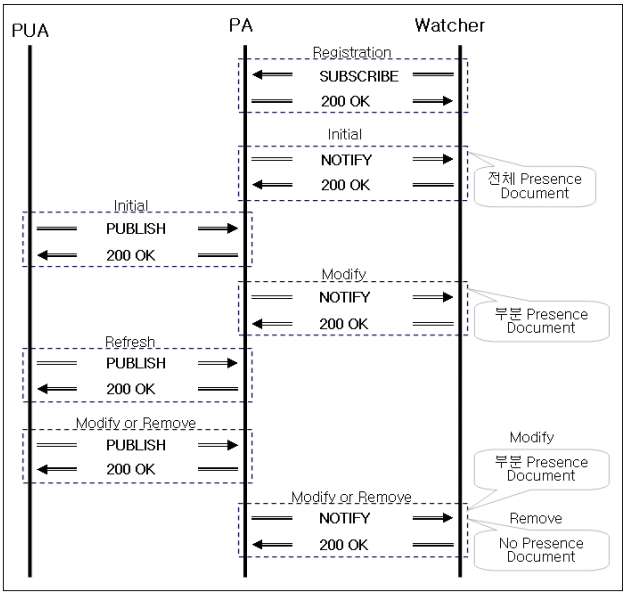
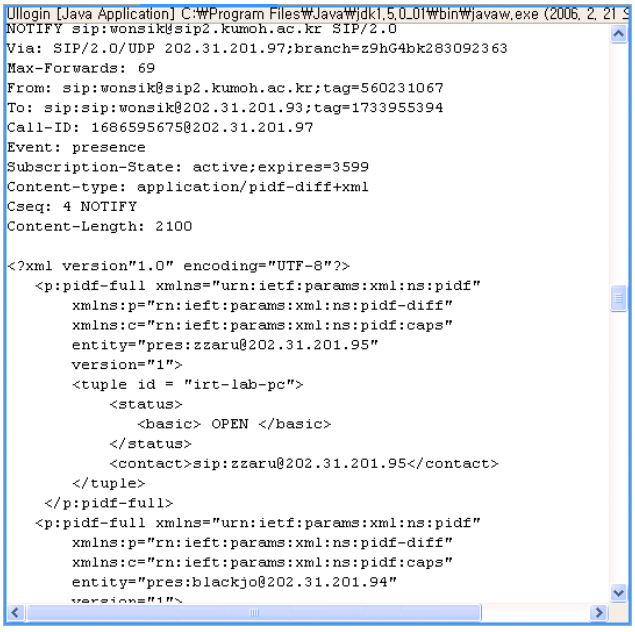


그림 2. 부분 Publication 및 Notification을 사용한 세션 설정 예

세션 설정은 SUBSCRIBE 메시지로 호를 설립하면서 시작된다. 이때 와처는 등록된 모든 사용자들의 정보로 이루어진 전체 프리젠스 문서가 포함된 NOTIFY 메시지를 받게 된다. Modify 이벤트 상태를 받으면 변경된 사용자의 상태정보를 부분 프리젠스 문서를 포함한 NOTIFY 메시지를 통해서 와처들에게 알린다. Remove 이벤트 상태를 받으면 사용자가 종료되었음을 프리젠스 문서를 포함하지 않은 NOTIFY 메시지를 통해서 와처들에게 알리고 세션을 종료한다.



```
Ulogin [Java Application] C:₩Program Files₩Java₩jdk1.5.0_01₩bin₩javaw.exe (2006. 2. 21 오
NOTIFY sip:wonsik@sip2.kumoh.ac.kr SIP/2.0
Via: SIP/2.0/UDP 202.31.201.97;branch=z9hG4bk283092363
Max-Forwards: 69
From: sip:wonsik@sip2.kumoh.ac.kr;tag=560231067
To: sip:sip:wonsik@202.31.201.93;tag=1733955394
Call-ID: 1686595675@202.31.201.97
Event: presence
Subscription-State: active;expires=3599
Content-type: application/pidf-diff+xml
Cseq: 4 NOTIFY
Content-Length: 2100

<?xml version"1.0" encoding="UTF-8"?>
   <p:pidf-full xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:p="rn:ieft:params:xml:ns:pidf-diff"
       xmlns:c="rn:ieft:params:xml:ns:pidf:caps"
       entity="pres:zzaru@202.31.201.95"
       version="1">
       <tuple id = "irt-lab-pc">
           <status>
              <basic> OPEN </basic>
           </status>
           <contact>sip:zzaru@202.31.201.95</contact>
       </tuple>
    </p:pidf-full>
   <p:pidf-full xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:p="rn:ieft:params:xml:ns:pidf-diff"
       xmlns:c="rn:ieft:params:xml:ns:pidf:caps"
       entity="pres:blackjo@202.31.201.94"
       version="1">
```

그림 3. 전체 프리젠스 NOTIFY 메시지의 예

```
Ulogin [Java Application] C:₩Program Files₩Java₩jdk1.5.0_01₩bin₩javaw.exe (2006. 2. 21 오후 5:3
NOTIFY sip:webtain1@sip2.kumoh.ac.kr SIP/2.0
Via: SIP/2.0/UDP 202.31.201.97;branch=z9hG4bk840409629
Max-Forwards: 69
From: sip:webtain1@sip2.kumoh.ac.kr;tag=1243036142
To: sip:wonsik@202.31.201.93;tag=700329515
Call-ID: 1288022535@202.31.201.97
Event: presence
Subscription-State: active;expires=3600
Content-type: application/pidf-diff+xml
Cseq: 6 NOTIFY
Content-Length: 461

<?xml version"1.0" encoding="UTF-8"?>
   <p:pidf-diff xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:p="rn:ieft:params:xml:ns:pidf-diff"
       entity="pres:webtain1@202.31.201.96"
       version="2">
       <p:replace sel="* / tuple [@id='irt-lab-pc'] / status / basic">
          NoSeat </p:replace>
       <contact>sip:webtain1@202.31.201.96</contact>
    </p:pidf-diff>
```

그림 4. 부분 프리젠스 NOTIFY 메시지의 예

그림 3과 그림 4는 각각 전체 프리젠스 문서를 실은 NOTIFY 메시지와 부분 프리젠스 문서를 실은 NOTIFY 메시지의 예이다. 그림 3에서 Content-Length 값은 2100이며 이는 프리젠스 문서의 크기를 나타낸다. 그에 반해 그림 4의 Content-Length 값은 461이며 문서의 크기가 감소함을 알 수 있다. 현재 로그인 되어 있는 사용자 수가 증가할수록 전체 프리젠스 문서의 크기는 커진다. 즉 프리젠스 정보의 적은 변화에 부분 프리젠스 문서를 사용하는 것이 전체 프리젠스 문서를 사용하는 것보다 메시지 전송에 더 효율적이라는 것을 알 수 있다.

본 논문에서는 PUBLISH 메시지에서도 전체 프리젠스 문서와 부분 프리젠스 문서를 구분하여 구현하였다. 이 경우에 프리젠스 문서의 크기가 NOTIFY 메시지만큼 큰 차이는 나지 않지만, 빈번한 프리젠스 정보의 변화에 따라 변화되는 프리젠스 문서의 크기를 줄일 수 있다.

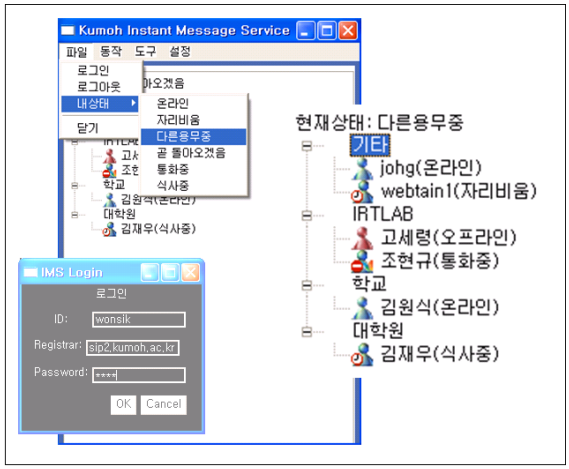


그림 5. 사용자 인터페이스

사용자 인터페이스는 그림 5와 같으며 메인 화면과 사용자 인증을 위한 화면을 캡쳐한 화면으로서 윈도우즈 환경과 비슷한 자바 SWT(Standard Widget Toolkit)를 사용하여 구현하였다.

4. 결론

기존의 SIP 프리젠스 시스템에서는 변화되는 프리젠스 정보를 전체 프리젠스 문서를 사용하여 업데이트함으로 인해 프리젠스 정보 전송 시 상당한 오버헤드가 발생할 수 있었다. 따라서 본 논문에서는 프리젠스 정보의 크기를 줄이기 위해 부분 Publication 및 Notification 방식을 사용하여 변화되는 프리젠스 정보만으로 구성된 부분 프리젠스 문서를 메시지 바디에 포함시켜 전달하는 프리젠스 서비스를 구현하였다. 이 방식의 사용으로 프리젠스 정보 전송 시 불필요한 트래픽 발생을 최소화 시켜 프리젠스 시스템의 반응 속도를 높일 수 있었다.

앞으로 구현된 프리젠스 서비스를 기반으로 VoIP, 인스턴트 메시지 등에 활용할 예정이다.

참고문헌

[1] J. Rosenberg, “A Presence Event Package for the Session Initiation Protocol (SIP),” RFC3856, August 2004.

[2] A. Niemi, Ed., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC3903, October 2004.

[3] A. B. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification,” RFC 3265, June 2002.

[4] A. Niemi, M. Lonnfors, E. Leppanen, "Publication of Partial Presence Information", draft-ietf-simple-partial-publish-03, July 18, 2005.

[5] M. Lonnfors, J. Costa-Requena, E. Leppanen, H. Khartabil, "Session Initiation Protocol (SIP) extension for Partial Notification of Presence Information", draft-ietf-simple-partial-notify-06, October 21, 2005.

[6] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, J. Peterson, "Presence Information Data Format (PIDF)", RFC3863, August 2004.

[7] M. Lonnfors, E. Leppanen, H. Khartabil, J. Urpalainen, "Presence Information Data format (PIDF) Extension for Partial Presence", draft-ietf-simple-partial-pidf-format-05, October 21, 2005.