한국산학기술학회논문지
Vol. 9, No. 4, pp. 929-935, 2008

SIP 프록시 서버의 부하 최소화를 위한 분산 처리

이영민[1], 노영섭[1*], 조용갑[1], 오삼권[2], 황희융[1]

Distributed processing for the Load Minimization of an SIP Proxy Server

Young-Min Lee[1], Young-Sup Roh[1*], Yong-Karp Cho[1], Sam-Kweon Oh[2] and Hee-Yeung Hwang[1]

요 약 인터넷 전화 서비스가 시장성 있는 기술로 각광 받게 되면서 SIP를 기반으로 많은 제품들이 개발되고 있으며, 개발된 인터넷 전화 시스템은 홈 네트워크 전화서비스 및 일반 사무실 전화서비스로 이용되고 있다. 인터넷 전화에서 호 연결 방법에는 단일전화 호 연결과 그룹 호 연결로 구분할 수 있는데, 그룹 호 연결은 하나의 요청 메시지를 그룹 안에 포함된 모든 전화기에게 병렬적으로 전달하는 메시지의 포킹 기능으로 인하여 호 연결을 위한 메시지 처리에서 부하가 많이 발생하여 네트워크에 많은 트래픽을 유발시킨다. 이러한 인터넷 전화의 시스템 모델은 호 연결 기능에 대한 메시지가 프록시 서버에게 집중되게 되어 있다. 따라서 짧은 시간 동안에 다량의 호 연결 메시지가 프록시 서버에 입력되는 경우 호 연결을 위한 메시지 처리가 지연되기 때문에 정상적인 전화 서비스가 이루어질 수 없게 된다. 본 논문은 기존 시스템 모델의 메시지 처리 지연 문제를 해결하기 위해 프록시 서버의 호 연결 요청 메시지를 단순화하고, 특정그룹 호 연결 요청 메시지의 포킹 기능을 분산 처리하여, 프록시 서버의 부하를 최소화하는 시스템 모델을 제안하고, 이의 구현을 통하여 성능개선을 확인하였다.

Abstract As internet telephony services based on Session Initiation Protocol (SIP) enter the spotlight as marketable technology, many products based on SIPs have been developed and utilized for home and office telephony services. The call connection of an internet phone is classified into specific call connections and group call connections. Group call connections have a forking function which delivers the message to all of the group members. This function requires excessive message control for a call connection and creates heavy traffic in the network. In the internet call system model, most of the call-setup messages are directed to the proxy server during a short time period. This heavy message load brings an unwanted delay in message processing and, as a result, call setup can not be made. To solve the delay problem, we simplified the analysis of the call-setup message in the proxy server, and processed the forking function distributed for the group call-setup message. In this thesis, a new system model to minimize the load is proposed and the subsequent implementation of this model demonstrates the performance improvement.

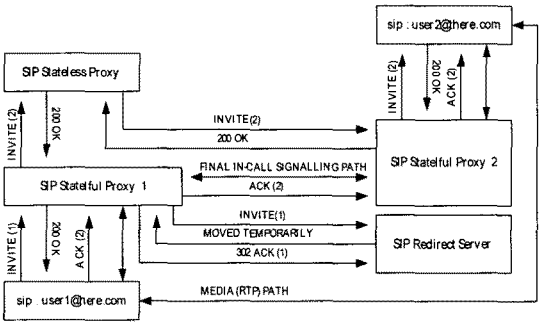
Key Words : SIP, Proxy server, VoIP, Internet telephony, Embedded system

1. 서론

세션 초기화 프로토콜(SIP, Session Initiation Protocol)은 응용 프로그램 영역에서 신호법(signaling)을 정의한 프로토콜로 인터넷 기반의 컨퍼런스, 전화, 음성메일, 이벤트 알림, 그리고 인스턴트 메시지 등과 같은 Transmission Control Protocol (TCP), User Datagram Protocol (UDP), 또는 Stream Control Transmission Protocol (SCTP) 기반의 멀티미디어 서비스의 응용에 사용되고 있는 데, 이는 SIP가 문자 기반의 프로토콜로서

[1]서울벤처정보대학원대학교 임베디드시스템학과
[2]호서대학교 컴퓨터공학부
*교신저자: 노영섭(ysroh@suv.ac.kr)
접수일 08년 6월 2일 수정일 08년 8월 10일 재재확정일 08년 8월 11일

[그림 1]과 같이 사용자를 구분하기 위하여 IP(Internet Protocol) 주소 대신 SIP URL(Uniform Resource Locator)을 전자우편 주소처럼 사용할 수 있어 제품에 적용하기가 쉽기 때문이다[1][2].



[그림 1] SIP 신호법(signaling)[1][2]

이러한 상황에서 최근 인터넷 전화 서비스가 시장성 있는 기술로서 각광을 받게 되면서, 각 회사들은 대부분 SIP를 기반으로 제품을 개발하고 있으며, 주로 아파트단지의 홈 네트워크 전화서비스 및 일반 사무실 전화서비스로 이용되고 있다.

SIP를 이용하여 인터넷 전화 서비스를 제공하는 시스템은 일반적으로 특정전화기의 호 연결 기능과 특정그룹 호 연결 기능을 지원한다. 특정전화기의 호 연결이란 하나의 전화기에 유일한 전화번호를 할당하여 그 전화번호에 해당하는 전화기와 호 연결을 하는 것이고, 특정그룹 호 연결이란 여러 개의 전화기에 공통된 전화번호를 할당하고 그 전화번호를 사용하여 호 연결을 요청하면 공통된 전화번호를 할당 받은 모든 전화기에게 호 연결 요청을 보내고 그 중 응답 가능한 전화기와 호 연결을 하는 것이다. 특정그룹 호 연결의 예로 홈 네트워크 전화서비스에서 세대 단위 호 연결과 일반 사무실에서 부서 단위 호 연결이 이에 해당한다.

SIP의 구성요소인 사용자 에이전트(UA, User Agent)는 전화기에 해당하며, 상황에 따라 호출자로 사용하면 사용자 에이전트 클라이언트(UAC, User Agent Client)로 동작하거나, 피호출자로 사용하면 사용자 에이전트 서버(UAS, User Agent Server)로 동작한다. 전화 연결을 위해 UA(C)와 UA(S) 사이에 프록시 서버(proxy server)가 존재하는데, 프록시 서버는 UA(C)와 UA(S)의 전화 연결을 위해 사용되는 메시지들을 수신하여 그 메시지가 송신되어야 할 UA의 위치를 분석한 후 해당 UA에게 그 메시지를 전달한다. 프록시 서버는 메시지 송수신 후, 이전 상태를 기억하지 않는 스테이트리스 프록시 서버(stateless proxy server)와 이전 상태를 기억하는 스테이트풀 프록시 서버(stateful proxy server)로 구분된다. 스테이트리스 프록시 서버는 메시지의 수신과 송신이 일대일로 처리하는 전달(forwarding) 기능을 지원하고, 스테이트풀 프록시 서버는 수신 메시지가 송신되어야 할 UA의 위치가 한 곳일 경우는 스테이트리스 프록시 서버와 같이 메시지 전달 처리를 하며, 송신 위치가 여러 곳일 경우는 여러 위치로 메시지를 병렬적으로 전달하는 메시지 포킹(message forking) 기능을 지원한다.

프록시 서버의 전달 기능은 특정전화기 호 연결 기능을 수행하고, 메시지 포킹 기능은 특정그룹 호 연결 기능을 수행한다. 따라서 기존 시스템 모델의 프록시 서버는 두 기능을 모두 수행 할 수 있는 스테이트풀 프록시 서버가 이용된다.

기존 시스템 모델은 호 연결 기능에 대한 처리를 위해 호 연결 메시지 처리 방법(전달 또는 메시지 포킹)에 대한 분석 처리와 메시지 포킹 처리가 프록시 서버에게 집중되어 있다. 따라서 이 모든 처리는 프록시 서버의 부하로 작용하여 짧은 시간 동안에 다량의 호 연결 메시지가 프록시 서버에게 입력될 경우 호 연결을 위한 메시지 처리가 지연되게 된다. 이 문제는 시스템에 중요한 장애로 인식되어 앞서 연구 되었다[3][4][5].

본 논문에서는 이 문제를 해결하기 위해 프록시 서버의 호 연결 요청 메시지 분석 기능을 단순화하고, 대부분 통화대기 상태로 있는 UA의 잉여자원을 활용하여 호 연결 요청 메시지의 포킹 기능을 분산 처리하여 프록시 서버의 부하를 최소화하는 시스템 모델을 제안하고 구현을 통하여 성능개선을 확인한다.

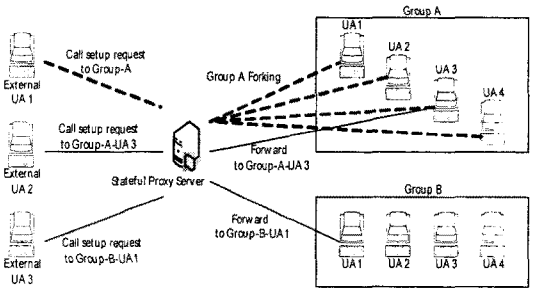
개선된 시스템 모델은 다음과 같이 구현하였다. 첫째 프록시 서버를 스테이트리스 서버로 대체하여 모든 입력된 호 연결 메시지를 전달한다. 둘째 스테이트리스 프록시 서버가 특정그룹 호 연결을 전달하기 위해 그룹을 대표하는 UA를 만든다. 셋째 대표 UA는 프록시 서버로부터 수신한 그룹 호 연결 메시지를 그룹 안의 모든 UA들에게 메시지의 포킹 처리를 한다. 따라서 개선된 시스템 모델에서 프록시 서버는 입력된 호 연결 메시지를 오직 전달 처리만 하므로 메시지 분석 기능이 단순화 되었고, 메시지 포킹 기능을 각 그룹에 잉여자원으로 분산 처리하여 호 연결 요청 메시지 처리에 의한 부하를 최소화하였다.

2. 기존 시스템 모델

2.1 기존 시스템 모델 개요

기존 시스템 모델은 프록시 서버를 스테이트풀 프록시

서버를 사용하여 특정전화 호 연결과 특정그룹 호 연결에 대한 처리를 프록시 서버에 구현된다.



[그림 2] 기존 시스템 모델의 호 연결 기능 예

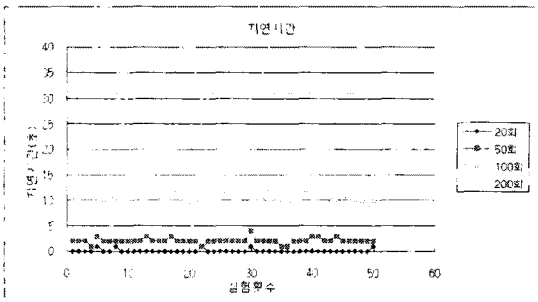
[그림 2]에서는 프록시 서버에서 특정전화 호 연결을 전달 처리와 특정그룹 호 연결에 대한 메시지를 포킹 처리하는 예를 나타내었다. 프록시 서버는 외부 UA1의 A 그룹 호 연결 요청을 A그룹 안에 존재하는 UA1부터 UA4에게 호 연결 메시지를 포킹 처리하고, 외부UA2와 그룹 A의 UA3의 특정전화 호 연결 요청은 해당 UA에게 직접 호 연결 메시지를 전달한다.

2.2 기존 시스템 모델의 문제점

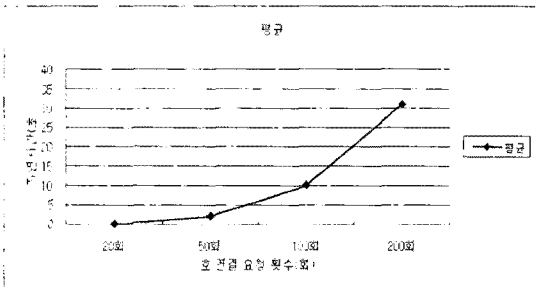
기존 시스템 모델의 프록시 서버에게 연속적으로 다량의 특정그룹 호 연결 메시지의 처리를 요구하였을 때 지연시간을 측정하기 위해 프록시 서버와 UAC는 각각 독립된 PC에 설치, UAS들은 한 개의 PC에 설치 후 다음과 같이 실험하였다. 프록시 서버가 한 개의 특정그룹 호 연결 메시지를 처리하는 과정을 포킹된 메시지를 수신한 모든UAS 로 부터 180 메시지(ringing message)를 받는 것까지로 제한한다. 이 내용은 [그림 6]에서 F1부터 F10까지에 과정이다. 이 과정을 20회, 50회, 100회, 200회 단위로 다량의 호 연결 요청을 하였을 때 지연시간을 측정하였다. 또한 실험치의 정확성을 위해 각 50회씩 실험하였다. 그 결과는 [그림 3]과 같다.

프록시 서버는 호 연결 메시지를 수신 할 때마다 처리 방법에 대한 분석 처리와 메시지 포킹 처리를 수행한다. 이런 수행은 프록시 서버의 부하로 작용하여 수신된 호 연결 메시지 처리의 지연이 발생한다. 실험 결과 프록시 서버에게 요청 메시지 부하를 걸었을 때 요청 메시지 중 마지막으로 수신한 요청 메시지를 포킹 처리하는데 걸리는 평균지연시간은 [그림 4] 과 같이 호 연결 요청이 20회 일 경우 0.1초, 50회 일 경우 2.08초, 100회 일 경우 10.08초, 200회 일 경우 31.16 초로 측정되었다.

결과적으로 짧은 시간 동안에 다량의 특정 그룹 호 연결로 인하여 프록시 서버의 메시지 처리가 급격히 느려지는 현상을 보이고 호 연결의 지연은 사용자의 불편을 야기하며 실용 가능한 시스템으로서의 가치를 상실하게 만든다.



[그림 3] 기존 시스템 모델의 프록시 서버 메시지 처리 부하 실험



[그림 4] 기존 시스템 모델의 지연시간 평균 값

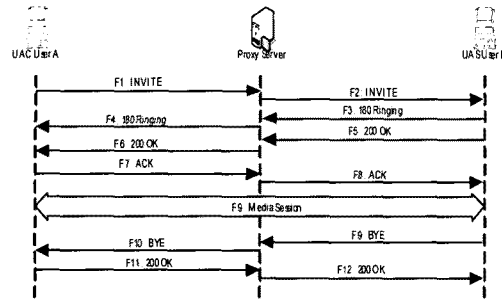
3. 호 연결 메시지 전달과 메시지 포킹의 메시지 처리량

본 절에서는 기존 시스템 모델에서 문제점으로 지적된 프록시 서버에서의 메시지 처리량을 파악하기 위하여 호 연결의 메시지 처리 방법에 따라 처리해야 하는 메시지 량을 비교하고 메시지 처리 방법에 따른 프록시 서버의 부하를 측정하였다.

3.1 메시지 전달

[그림 5]는 사용자 A(UAC)가 사용자 B(UAS)에게 전화를 걸어 호 연결을 요청하였을 때 프록시 서버가 호 연결을 위해 메시지 전달 처리를 수행하는 흐름을 나타내었다. [그림 5]에서 보면 UAC의 호 연결 요청에 의해 세션 연결 시작부터 세션 연결 종료까지 프록시 서버에서

입출력 처리하는 메시지 개수는 12개이다.



[그림 5] 호 연결 요청에 대한 메시지 전달 처리 흐름[1]

따라서 프록시 서버에서 입출력 처리 메시지 개수는 [수식 1]과 같다.

$$F1(x) = 12x$$

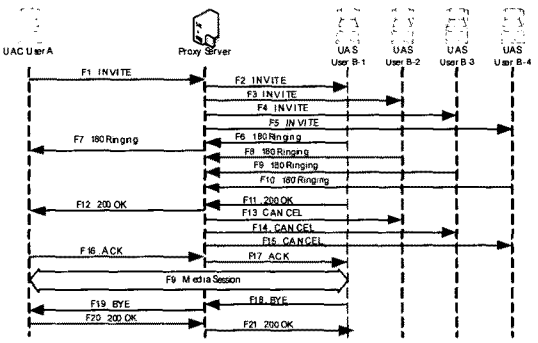
[수식 1] 프록시 서버에서의 메시지 처리 개수

x : 호 연결 요청 수

2.2 메시지 포킹

[그림 6]은 사용자 A(UAC)가 사용자 그룹 B에게 그룹 호 연결 요청을 하였을 때 사용자 B에 해당하는 UAS Group(사용자B-1부터 사용자B-4)에게 프록시 서버가 호 연결을 위해 메시지 포킹 처리를 수행하는 흐름을 나타내고 있다. [그림 6]에서 UAC의 호 연결 요청에 의해 세션 연결 시작부터 세션 연결 종료까지 프록시 서버에서 입출력 처리하는 메시지 개수는 21개이다.

따라서 [그림 6]의 경우 프록시 서버에서 입출력 처리되는 메시지의 개수는 [수식 2]와 같다.



[그림 6] 그룹 호 연결 요청에 대한 메시지 포킹 처리 흐름[6]

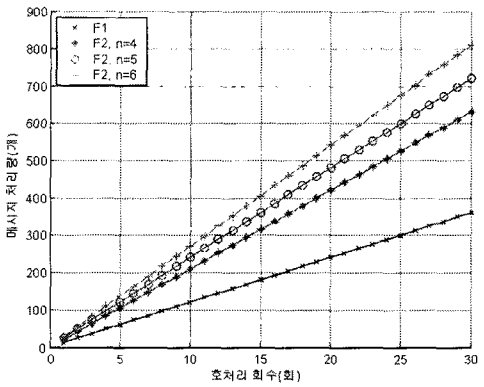
$$F2 = (9 + 3n)x$$

[수식 2] 프록시 서버에서의 메시지 처리 개수

x : 호 연결 요청 수,

n : 그룹내 UAS의 개수

2.3 메시지 처리량 비교 및 결과



[그림 7] 호 연결 요청 당 프록시 서버의 메시지 처리량

[그림 7]은 [수식 1]과 [수식 2]를 이용하여 프록시 서버가 처리하는 메시지량을 나타낸 그래프이다. F1은 [수식 1]을 사용하였고, F2는 [수식 2]에서 포킹해야 할 UAS들의 개수를 4, 5, 6으로 설정하였다.

그 결과 메시지 전달 처리량 보다 메시지 포킹 처리량이 UAS들의 개수가 4인 경우 1.75배, 5인 경우 2배 그리고 6개인 경우 2.25배 많았다. 따라서 메시지의 포킹처리로 인한 프록시 서버에서의 메시지량이 UAS의 개수에 따라 증가한다는 것을 알수 있고 이로 인하여 프록시 서버에서 메시지 처리에 지연이 발생하여 원활한 호처리를 할 수 없게 된다.

4. 개선된 시스템 모델

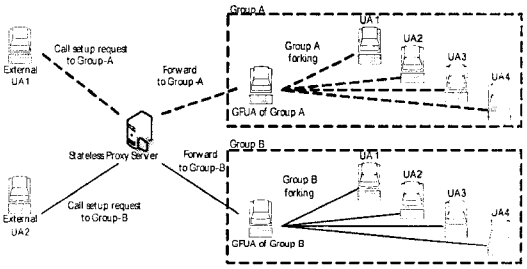
4.1 개선된 시스템 모델 개요

개선된 시스템 모델은 기존 시스템 모델에 프록시 서버의 메시지 처리 지연 문제를 해결하기 위해 프록시 서버의 기능을 최소화하여 구현하였다. 따라서 프록시 서버는 호 연결 요청 메시지를 전달 처리만 수행하고, 메시지 포킹 기능 및 입력된 요청 메시지 처리 방법에 대한 분석 기능은 제거되었다.

이렇게 제거된 기능은 특정그룹 호 연결을 위해 필요한 기능이며, 특정그룹 호 연결의 특징은 그룹단위로 호 처리가 이루어진다. 이 점을 이용하여 프록시 서버에서

제거된 기능을 그룹단위로 분산 처리하여 구현하였다. 즉 그룹 호 연결 기능을 수행하는 그룹 포킹 사용자 에이전트(GFUA, Group Forking User Agent)를 구현하고, GFUA를 그룹 단위로 잉여자원에 설치하여 기존 시스템 모델의 기능을 모두 제공한다. 이러한 GFUA는 통상 하나의 그룹 네트워크에 연결하는 대표 시스템 내에 구성되게 된다.

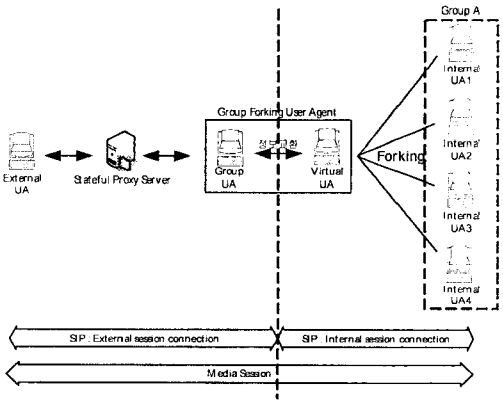
GFUA의 기능은 다음과 같다. 첫째 프록시 서버가 전달하는 그룹 호 연결 메시지를 받아 기존의 프록시 서버가 하던 역할을 대신한다. 둘째 그룹 내에 존재하는 UA들에게 프록시 서버로부터 입력 받은 호 연결 메시지를 포킹한다. 이러한 기능에 대해 [그림 8]에 나타내었다.



[그림 8] 프록시 서버의 전달 처리와 GFUA를 이용한 특정그룹 호 연결

4.2 GFUA(Group Forking User Agent)

GFUA는 그룹 사용자 에이전트(GUA, Group User Agent)와 가상 사용자 에이전트(VUA, Virtual User Agent)로 구성되다. [그림 9]에 A그룹 호 연결을 요청하는 외부 UA와 그 요청을 GUA와 VUA를 이용하여 그룹 안의 UA들에게 메시지가 포킹 처리되는 호 연결을 나타내었다.



[그림 9] GUA와 VUA를 이용한 그룹 호 연결

4.2.1 GUA(Group User Agent)

GUA는 특정그룹의 내부 UA들을 대표하는 UA로써, 그룹 호 연결을 요청하는 외부 UA와 세션을 연결한다. 이렇게 연결되는 세션을 [그림 9]에서 외부 세션 연결(external session connection)이라고 하였다.

4.2.2 VUA(Virtual User Agent)

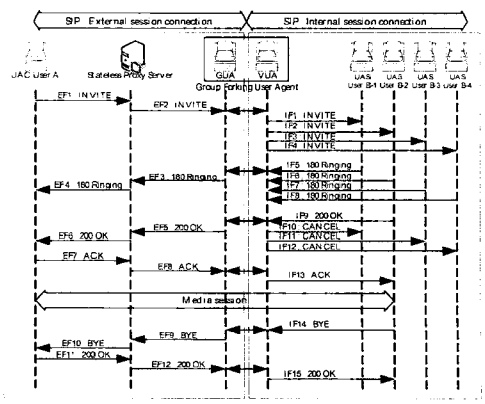
VUA는 GUA와 세션을 연결한 외부 UA를 대신하는 가상의 UA로써 내부 UA들이 외부 UA와 직접 세션을 연결하는 것과 같은 동작을 할 수 있도록 해준다. 이렇게 연결된 세션을 [그림 9]에서 내부 세션 연결(internal session connection)이라고 하였다. VUA는 그룹 내의 모든 UA들에게 외부 UA의 호 연결 메시지를 포킹 처리한다. 그리고 내부 UA 중에서 실제 연결된 내부 UA와 내부 세션을 연결한다.

4.2.3 GUA와 VUA 세션 정보 교환

GUA는 외부 UA와의 세션 정보를 VUA에게 제공하고, VUA는 내부 UA와의 세션 정보를 GUA에게 제공하여 외부 세션과 내부 세션은 동일한 상태가 유지될 수 있도록 한다.

4.3 GFUA를 이용한 그룹의 메시지 처리의 흐름

[그림 10]은 외부 UA가 특정그룹으로 호 연결을요청하였을 때 외부 UA와 GUA 사이의 세션 연결 흐름과 VUA와 내부 UA들 사이의 세션 연결 흐름을 나타낸 것이다. 외부흐름(EF, External Flow) EF1 부터 EF12는 [그림 5]에서 나타낸 전달 처리 흐름과 동일하다. 내부 흐름(IF, Internal Flow) IF1 부터 IF15는 [그림 6]에서 나타낸 메시지 포킹 처리 흐름과 동일하다. 그리고 GUA와 VUA는 서로 세션정보를 내부적으로 교환한다.

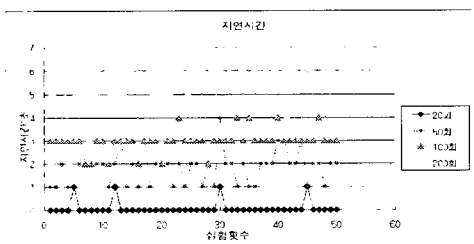


[그림 10] GFUA를 이용한 그룹 호 연결 흐름

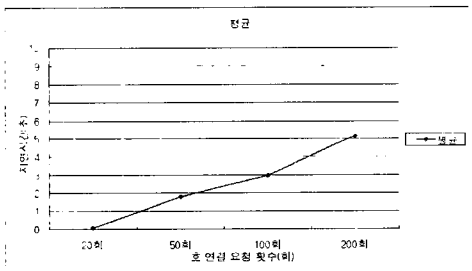
4.4 결과

개선된 시스템 모델의 프록시 서버에게 연속적으로 다량의 특정그룹 호 연결 메시지의 처리를 요구하였을 때 지연시간을 측정하기 위해 프록시 서버와 UAC는 각각 독립된 PC에 설치, UAS들과 GFUA는 한 개의 PC에 설치 후 다음과 같이 실험하였다. 프록시 서버가 한 개의 특정그룹 호 연결 메시지를 처리하는 시간은 GFUA로부터 180메시지를 받는 시점까지로 정의하였다. 이 내용은 [그림 10]에서 EF1, EF2, EF3의 과정이다. 이 과정을 20회, 50회, 100회, 200회 단위로 다량의 호 연결 요청을 하였을 때 처리지연을 측정하였다. 또한 실험의 정확성을 위하여 각 50회씩 실험하였다. 그 결과는 [그림 11]과 같다.

실험 결과 프록시 서버에게 요청 메시지 부하를 걸었을 때 요청 메시지 중 마지막으로 수신한 요청 메시지를 포킹 처리하는데 걸리는 평균지연시간은 [그림 12]와 같이 20회 일 경우 약 0.8초, 50회 일 경우 약 1.8초, 100회 일 경우 약 2.98초, 200회 일 경우 약 5.16초로 측정되었다.

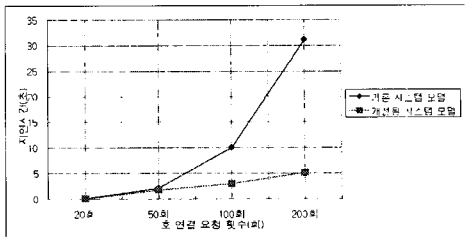


[그림 11] 개선된 시스템 모델의 프록시 서버 메시지 처리 부하 실험



[그림 12] 개선된 시스템 모델의 지연시간 평균 값

[그림 13]은 [그림 4]과 [그림 12]의 실험 결과를 이용하여 호 연결 요청 단위에 따라 발생하는 지연시간을 나타내었다. 그 결과 개선된 시스템 모델의 프록시 서버의 성능이 기존 시스템 모델을 사용하였을 경우에 비하여 개선되었음을 확인할 수 있다.



[그림 13] 호 연결 요청 메시지의 지연시간에 대한 비교

5. 결 론

현재까지 개발된 인터넷 전화 서비스 시스템은 특정그룹 호 연결을 위한 기능이 프록시 서버에서 모두 처리를 하는 구조로 되어 있다. 이런 시스템 모델의 프록시 서버는 호 연결을 해야 하는 그룹의 수가 늘어나거나 그룹 내의 UA가 늘어날수록 그 기능이 복잡해지며, 결과적으로 프록시 서버에서 호 연결을 위한 메시지 처리에 지연이 발생한다.

본 논문에서 이러한 문제를 해결하기 위해 프록시 서버의 기능을 단순화 및 분산 처리하여 프록시 서버의 부하를 줄이는 시스템 모델을 제안하고 구현 및 실험을 통하여 성능 개선을 확인하였다.

개선된 시스템 모델의 프록시 서버는 특정전화 호 연결 기능만 제공한다. 즉, 기존 시스템 모델의 프록시 서버에 부하로 작용하는 입력된 호 연결 메시지 분석 및 특정그룹 호 연결을 위한 메시지 포킹 처리 기능을 제거하였다.

제거된 특정그룹 호 연결 기능은 GFUA에서 구현하며, GFUA는 그룹단위로 배치되어 특정그룹 호 연결 기능을 분산 처리하여 프록시 서버에서 특정그룹 호 연결에 대한 부하를 줄여준다.

GFUA는 프록시 서버가 특정전화 호 연결 방법으로 접근하기 위해 그룹을 대표하는 대표전화기(GUA) 기능과 걸려온 호 연결을 그룹 내의 UAS들에게 호 연결 메시지를 포킹 처리(FUA)하는 기능을 수행한다.

개선된 시스템 모델은 낮은 사양의 프록시 서버로 더 많은 호 연결을 지원하는 시스템을 구현할 수 있다.

향후, 연구과제는 프록시 서버의 위치정보 처리에 대한 지연문제를 개선하는 것이다. 프록시 서버는 입력된 메시지에서 목적지 정보를 추출, 로케이션(location) 서버에게 목적지 정보 질의, 로케이션 서버에 응답 그리고 응답된 목적지 정보를 이용하여 입력된 메시지를 목적지에 전달 처리를 한다. 이 작업은 로케이션 서버와 통신이 필요하며 이 통신은 호 연결 처리 지연시간이 된다. 따라서 호 연결 처리 지연문제가 되는 위치정보 처리에 대한 연구가 필요하다.

참고문헌

[1] RFC 3665, Session Initiation Protocol (SIP) Basic Call Flow Examples

[2] RFC 3261, SIP : Session Initiation Protocol

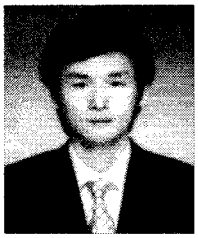
[3] 유승선, 유기형, 임평종, 현철주, 곽훈성, "SIP프로토콜 스택을 기반으로 하는 분산형 IP PBX 단말기 설계", 한국통신학회논문지, 제31권, 제44A호(2006.04)

[4] 현욱, 강신각, 김대영, "SIP Proxy에서의 트랜잭션 검출 성능 향상을 위한 Ihash기법 적용", 한국인터넷정보학회, 2006 정기총회 및 추계학술발표대회, 제7권, 제2호

[5] 현욱, 강신각, "SIP Proxy 서버 안정성 향상을 위한 Thread Pool 기법 적용", 한국인터넷정보학회, 2006 정기총회 및 추계학술발표대회, 제7권, 제2호

[6] draft-ietf-sipping-service-examples-10, Session Initiation Protocol Service Examples

이 영 민(Young-Min Lee) [정회원]



- 2005년 8월 : 호서대학교 정보재어학과 (공학사)
- 2007년 8월 : 서울벤처정보대학원대학교 임베디드시스템학과 (공학석사)
- 2007년 10 ~ 현재 : (주)하이트론시스템즈 Display팀

<관심분야>
임베디드시스템, 컴퓨터구조, 정보통신

노 영 섭(Young-Sup Roh) [종신회원]

- 1988년 2월 : 인하대학교 전자공학과(공학사)
- 1996년 8월 : 한국과학기술원 정보및통신공학과(공학석사)
- 2005년 2월 : 고려대학교 전기, 전자,전파공학과(공학박사)
- 1987년 11월 ~ 1998년 2월 : LG전자 미디어통신연구소 선임연구원
- 1998년 3월 ~ 2001년 2월 : 청강문화산업대학교 이동통신과 교수
- 2001년 3월 ~ 2005년 2월 : 주식회사 싸이버뱅크 연구개발부문 상무이사
- 2005년 3월 ~ 현 재 : 서울벤처정보대학원대학교 임베디드시스템학과 교수

<관심분야>
임베디드시스템, 모바일 컴퓨팅, 이동통신, 유비쿼터스 네트워크

조 용 갑(Yong-Karp Cho) [정회원]

- 2001년 8월 : 연세대학교 산업대학원 전자계산학과 (공학석사)
- 2006년 9월 ~ 현재 : 서울벤처정보대학원대학교 임베디드시스템학과 박사과정

<관심분야>
컴퓨터임베디드시스템, 실시간운영체제(RTOS), 컴퓨터통신

오 삼 권(Sam-Kweon Oh) [종신회원]

- 1980년 2월 : 한국항공대학교 항공전자공학과(공학사)
- 1986년 12월 :남플로리다대학교 컴퓨터과학및공학과 (컴퓨터과학석사)
- 1994년 5월 : 퀸즈대학교 컴퓨터및정보과학과(컴퓨터과학박사)
- 1995년 3월~현재 : 호서대학교 컴퓨터공학부 교수
- 1980년 2월~1984년 8월 : 삼성전자통신연구소 연구원
- 1994년 9월~1995년 1월 : 한국전자통신연구원 선임연구원

<관심분야>
임베디드시스템, 실시간및분산시스템, 컴퓨터 게임

황 희 융(Hee-Yeung Hwang) [종신회원]

- 1960년 2월 : 서울대학교 공과대학 전기공학과 (공학사)
- 1964년 2월 : 서울대학교 대학원 전기공학과 (공학석사)
- 1974년 4월 : 서울대학교 대학원 전기공학과 (공학박사)
- 1968년 2월 ~ 1993년 2월 : 서울대학교 공과대학 교수
- 1983년 12월 ~ 1984년 12월 : 미국 Florida F.I.T 객원교수
- 1989년 11월 ~ 1992년 3월 : 서울대 컴퓨터 신기술공동연구소장
- 1993년 3월 ~ 2004년 2월 : 호서대학교 공과대학 교수
- 2004년 3월 ~ 현 재 : 서울벤처정보대학원대학교 임베디드시스템학과 교수
- 2007년 4월 ~ 현 재 : 서울벤처정보대학원대학교 총장

<관심분야>
임베디드시스템, 마이크로프로세서, RTOS