광대역통합망에서의 비동기 구조를 이용한 신뢰성 있는 웹 서비스 제공방안

김 형 민[†] · 정 의 헌[††] · 김 화 성[†††]

요 약

최근 정보통신환경은 통신, 방송, 인터넷이 통합되는 디지털 융합 서비스 제공 형태로 급속히 변화하고 있다. 이에 따라 통신환경이 개방형 네트워크 형태의 광대역통합망으로 변화하였다. 이런 광대역통합망에서는 Open API(Application Program Interface)에 의하여 third-party 응용의 제공을 가능하게 한다.

Open API에는 대표적으로 Parlay API가 있으며 Parlay API와 웹 서비스를 접목시키고 한 단계 더 추상화한 Parlay X API가 있다. Parlay X API는 웹 서비스를 이용한다. 따라서 Parlay X API는 웹 서비스의 장점을 가지고 있지만 웹 서비스의 단점도 가지고 있다. 웹 서비스의 가장 큰 단점으로는 웹 서비스에 QoS(Quality of Service)제공을 위한 방법이 제시되어 있지 않다는 것이다.

본 논문에서는 웹 서비스를 광대역통합망에서 효율적으로 동작시키기 위해서 QoS의 여러 문제 중 신뢰성 지원방안에 대하여 제시하고자 한다. 제안된 방안은 웹 서비스의 신뢰성 지원을 위하여 비동기 구조를 적용하고, 예상응답시간을 이용하여 클라이언트와 서버간의 재전송 효율을 증가시키고자 한다. 시뮬레이션을 통하여 성능평가를 실행하였으며 기존의 방법과 비교하여 보았다. 그 결과 제안된 방법이 기존의 방법보다 좋은 성능을 보인 것을 확인할 수 있었다.

키워드 : 광대역통합망, 웹 서비스, 개방형 API, 개방형 서비스, 신뢰성, 비동기

A Reliable Web Service Support Mechanism based on Asynchronous Architecture in BcN

Kim Hyoung-min[†] · Jung Yeu-hun[††] · Kim Hwa-sung[†††]

ABSTRACT

Recently, the telecommunication network is in a transition toward the BcN (Broadband convergence Networks) that integrates the Internet, the telecommunication and the broadcasting. The BcN makes it possible to provide of the third-party application by using the Open API

The Parlay API is an example of the Open API and the Parlay X API is grafted in the Web Service and is abstracted from the Parlay API. So, the Parlay X API does not only have advantages of a Web Service, but also disadvantages of the Web Service. The main disadvantage of the Web Service is that it does not support the any QoS mechanism.

In this paper, we propose the architecture that solves the reliability among the QoS issues for the Web Service. The proposed mechanism adopts the Asynchronous architecture for the reliable Web Service and improves the performance of the retransmission between the client and the server using the expected response time. We perform the performance evaluation through the simulation in order to compare with the existing mechanism. Consequently, we can confirm that the performance of the proposed mechanism performs better than the existing mechanism.

Key Words : BcN, Web Service, Open API, Open Service, Reliable, Asynchronous

1. 서 론

최근 정보통신환경은 통신, 방송, 인터넷이 통합되는 디지털 융합 서비스 제공 형태로 급속히 진행되고 있다. 그리고 네트워크 기능과 성능의 획기적 발전으로 네트워크 적용범위가 가전, 자동차, 영상, 콘텐츠 등의 거의 모든 영역으로 확대되고 있다. 이에 따라 컴퓨터, 통신, 방송 등 모든 정보통신 기기가 하나의 네트워크에 접속되는 개방형 광대역통합망(BcN) 기반의 네트워크로 진화할 전망이다.

BcN에서는 개방형 인터페이스를 통하여 third-party 업체

※ 본 연구는 대학 IT 연구센터 육성지원사업의 연구결과로 수행되었음.
† 준 회 원 : 광운대학교 전자통신학과 박사과정
†† 준 회 원 : ㈜헤리트 부설연구소 연구원
††† 정 회 원 : 광운대학교 전자통신공학과 교수
논문접수 : 2005년 9월 23일, 심사완료 : 2005년 10월 5일

들이 서비스를 제공할 수 있다. 이러한 서비스를 제공하기 위한 개방형 인터페이스의 대표적인 예로서 Parlay API가 있다. Parlay API는 Parlay 그룹에서 제안한 새로운 방식의 open programmable network API규격이고 Parlay 그룹은 이 규격이 상용제품의 구현에 채택될 수 있도록 촉진하는 것을 목표로 하고 있다. 현재 Parlay API는 버전 5까지 발표 되었다. 최근 Parlay 그룹은 최근 가장 각광받는 기술인 웹 서비스를 이용하여 서비스 제공이 가능한 Parlay X API규격을 발표하였다. Parlay X API는 CORBA 기반의 IDL(Interface Definition Language)로 정의된 Parlay API들을 더욱 추상화하고 단순화하여 XML 기반의 WSDL(Web Service Description Language)로 표준화가 된 개방형 서비스 인터페이스이다. 현재 13가지 종류의 통신망 기능들이 정의된 Parlay X Web service API 2.0까지 재정되었다[1], [2], [3].

Parlay X API는 서비스의 배포와 위치의 변경이 용이하며, 웹의 어디서나 실행이 가능하다. 또한 웹 컴포넌트의 개발에 의한 재사용을 통하여 비용감소 및 개발기간의 단축 등의 장점을 가진다. 이에 따라 통신망에서 새로 제공되는 서비스 응용들이나 기존의 응용들이 개발이 용이하고 사용하기 편리한 웹 서비스로 개발되거나 변화할 것이라고 기대된다. 하지만 웹 서비스는 QoS에 관한 어떤 것도 지원하고 있지 않기 때문에 웹 서비스를 BcN에서 그대로 적용하기는 어렵다. 따라서 웹 서비스를 BcN에서 적절하게 동작시키기 위해서는 QoS 지원 및 활용화 방안이 제시되어야 할 것이다.

본 논문은 앞서 설명한 Parlay X API를 이용한 통신망 서비스 제공 시 여러 QoS문제 중 신뢰성 제공방법에 대하여 제시하고자 하며, 본 논문의 구성은 다음과 같다. 제 2장에서는 일반적인 웹 서비스와 BcN에 웹 서비스를 적용하기 위한 방법 등에 대하여 소개하고, 3장에서는 BcN에서 신뢰성 제공을 위한 웹 서비스 구조를 제안하며, 4장에서는 제안하는 신뢰성 있는 웹 서비스의 구조에 관한 모의실험을 통하여 제안된 방안을 검증한다. 마지막으로 5장에서 본 논문의 결론을 맺는다.

2. 웹 서비스와 광대역통합망

2.1 웹 서비스

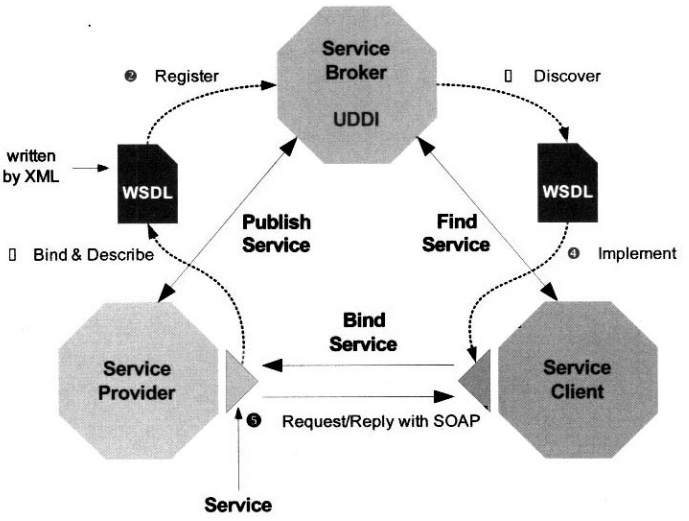
웹 서비스는 공동사용이 가능한 기계에서 기계로 상호작용을 지원하기 위하여 디자인된 소프트웨어 시스템으로 기계간의 처리가 가능한 인터페이스를 정의하였다(WSDL). 다른 시스템과 통신을 위해서 일반적으로 HTTP에 SOAP 메시지를 실어 웹 서비스를 사용한다[4]. 웹 서비스에서는 WSDL을 사용한 통신 구조를 사용하고 있다. WSDL은 웹 서비스 클라이언트가 웹 서비스를 이용하는데 필요한 정보를 담고 있으며 WSDL을 사용하여 응용과 데이터베이스에 접근 및 호출을 할 수 있다[5].

웹 서비스에는 크게 서비스 제공자, 서비스 요청자, 서비스 중개자의 세가지 역할이 존재한다.

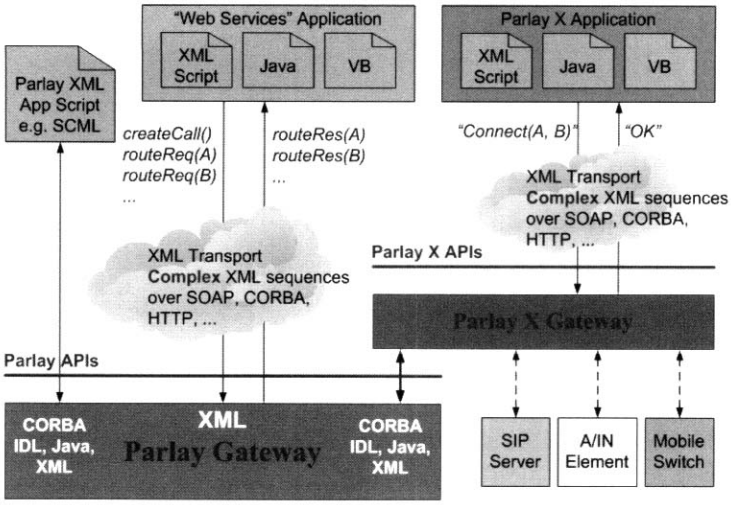
2.2 광대역통합망에서의 웹 서비스

Parlay X API는 텔레포니 응용의 개발을 가능하게 하기 위하여 설계되었다. Parlay X API는 80%의 응용이 망 자원의 20%만 이용한다는 80/20 원칙과, 복잡한 Parlay API를 보다 단순하게 정의하자는 KISS(Keep it Simple, Stupid) 원칙에 의해 정의되었다. Parlay X는 현재의 Parlay API의 제공 능력을 블록화 하여 최상위로 추상화한 API형태를 제공하며, 추상화된 메소드들을 통하여 응용을 보다 손쉽게 개발할 수 있도록 해준다.

(그림 2)는 Parlay 웹 서비스와 Parlay X API, Parlay API와의 관계를 보여 주고 있다. Parlay X 응용은 Parlay X API 인터페이스를 통해서 Parlay X 게이트웨이를 이용할 수 있다. Parlay X 게이트웨이는 기존의 Parlay API 인터페이스를 이용하여 Parlay 게이트웨이를 거쳐 네트워크에 접속하는 것도 가능하도록 되어 있고, 아니면 직접 네트워크에 접속할 수 있게 되어 있다. 또한 (그림 2)에는 Parlay X API가 Parlay API보다 더 간결화 되어 있고 어떤 방법으로 네트워크와 연



(그림 1) 웹 서비스 모델



(그림 2) Parlay API와 Parlay X API

결되는지 알 수 있다. 일반적인 웹 서비스 응용과 Parlay X 응용이 각각의 인터페이스를 통하여 연결을 맺고 call을 전송하는 부분이다. Parlay X 응용에서는 Parlay X API 인터페이스를 이용하여 간단하게 Connect(A, B)만을 호출하여 call설정을 간단하게 한다. 하지만 그에 비해서 Parlay API 인터페이스를 사용하는 응용은 call을 설정하기 위해서 Parlay X API보다 많은 메소드들을 사용해야만 한다. 따라서 더 많은 메소드들을 구현하고 호출하기 위해서 더 많은 다른 동작들이 필요하게 된다.

3. 광대역통합망에서 신뢰성 제공을 위한 웹 서비스 구조

3.1 웹 서비스 QoS

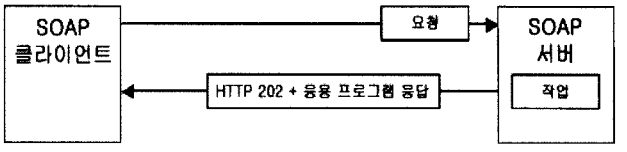
3.1.1 웹 서비스 QoS 요소

서비스 제공자들은 서비스 요청자들에게 서비스 요청의 캐싱과 부하 균등화 같은 다양한 접근방식을 통해 높은 QoS를 제공할 수 있다. 캐싱과 부하 균등화는 웹 서버 레벨과 웹 응용서버 레벨에서 수행될 수 있다. 부하 균등화는 다양한 유형의 트래픽들의 우선순위를 정하고 각각의 요청이 적절한 가치로서 취급될 수 있도록 해야 한다. 서비스 제공자는 다양한 고객과 서비스 유형에 필요한 용량을 결정하는데 용량 모델을 사용하고 다양한 응용과 고객들의 적절한 QoS 레벨을 확인함으로써 차별화된 서비스를 제공할 수 있다. 예를 들어, 멀티미디어 웹 서비스는 좋은 처리율을 필요로 하지만 은행 웹 서비스는 보안과 트랜잭션 QoS(transactional QoS)가 필요하다. 웹 서비스는 다음과 같은 요구조건을 가진다[6], [7].

- 가용성(Availability)
- 접근가능성(Accessibility)
- 무결성(Integrity)
- 퍼포먼스(Performance)
- 신뢰성(Reliability)
- 표준(Regulatory)
- 보안(Security)

3.1.2. 비동기 구조를 이용한 웹 서비스 제공 방법

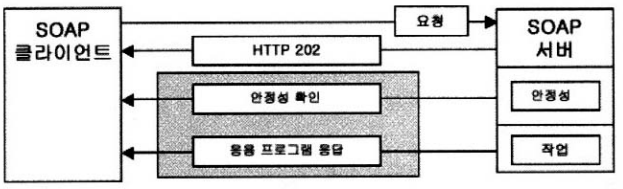
웹 서비스는 기본적으로 HTTP를 전송 프로토콜로 사용하고 있다. HTTP는 구현이나 관리가 쉽지만 근본적으로 메시지에 대한 책임을 지지 않기 때문에 신뢰성이 결여되어 있다. HTTP는 메시지의 수신 종점에서 요청을 수신하였다는 것을 보장하는 서비스가 존재하지 않는 사실로 증명할 수 있다. HTTP는 네트워크의 자원 부족이나 치명적인 일반 오류에 대한 대응 네트워크 계층의 기능은 있지만, 클라이언트의 응답의 수신이나 요청에 대한 확인 메커니즘이 없다. 일반적인 경우 HTTP 오류는 단순한 재전송 작업으로 처리되지만 이런 재전송 작업은 불필요한 트래픽을 만들기 때문에 비효율적이다[8]. 따라서 이런 웹 서비스의 문제점을 해결하기 위해서는 대칭적이며 비동기적 구조가 필요하다. 대칭적 구조는 각 메시지는 한번만 처리되고 하나의 승인만을 생성해야 하는 것을 의미한다. 일반적인 웹 서비스의 동기 구조에 대하여 알아보면 다음 (그림 3)과 같다.



(그림 3) 웹 서비스 동기 구조

동기 구조의 경우 클라이언트에서 서버로 요청을 전송하면, 서버에서는 그 요청에 의한 작업이 완료되면 응답메시지로 HTTP 202와 응용 프로그램 응답을 돌려주게 된다. HTTP 202는 어떤 처리의 완료를 나타내는 HTTP 상태 코드가 아니라 단순하게 메시지를 잘 받았다는 것을 의미하는 코드이다. 이런 동기 구조에서는 외부 소스에 의해 연결이 끊어져서

요청이나 응답이 누락될 수 있고, 서버의 오프라인이나 과부하로 인하여 시간 초과가 발생할 수 있다. 따라서 이런 문제를 해결하기 위해서 비동기 구조를 적용할 수 있다.

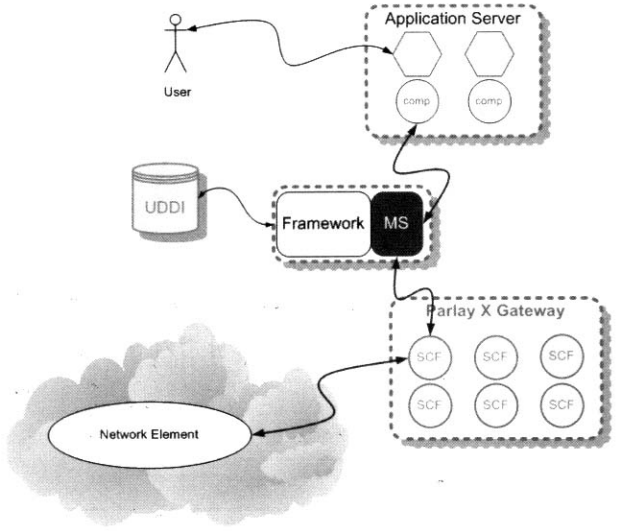


(그림 4) 웹 서비스 비동기 구조

비동기 모델에서는 어떤 요청을 수신하게 되면 서버가 네트워크 계층에서 메시지 스트림을 읽고 바로 클라이언트에게 HTTP 202응답을 돌려 보내게 된다. 이 프로세스는 서버에 메시지를 전송하고 HTTP 202를 수신하는 동안에만 동기적이므로 연결에 의해 발생 가능한 문제를 해결할 수 있다. 이후 메시지를 응용 프로그램에 전달 한다. 이 방법은 메시지가 서버의 큐에서 시간초과로 누락되는 것을 방지할 수 있다.

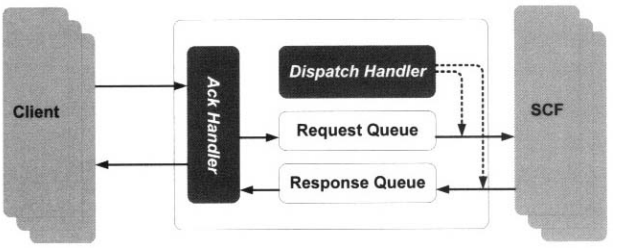
3.1.3. 광대역통합망에서 신뢰성 있는 웹 서비스 제공 방안

광대역통합망에서 신뢰성 있는 웹 서비스를 제공하기 위하여 앞 절에서 설명한 비동기 구조를 적용하여 MS(Message Server)라 칭하고, MS를 적용한 웹 서비스 구조는 (그림 5)와 같다.



(그림 5) BcN에서 웹 서비스 구조

MS는 응용서버와 Parlay X 게이트웨이의 SCF사이를 비동기 방식으로 연결하고 응용서버에게 재전송 시점을 알려준다. 현재 일반적인 웹 서비스구조에서는 타임아웃이 발생하면 서비스 사용자가 그 서비스에 대하여 다시 재전송을 요구할 수도 있고 하지 않을 수도 있다. 그러나 광대역통합망에 웹 서비스 적용 시 재전송의 주체는 응용서버가 될 것이며 SCF에게서 타임아웃이 발생하면 정해진 횟수 등에 의해서 재전송할 것이라고 예상한다. MS는 다수의 응용서버와 다수의 SCF에 연결된다. MS의 구성은 (그림 6)과 같다.



(그림 6) MS의 구성

MS의 기능은 비동기 구조의 적용, AS로부터 Request수신, SCF에게 Request 전송, SCF의 응답속도 측정, Queuing Delay 계산 등의 기능을 가진다. MS는 AS로부터의 메시지를 저장하여 SCF에게 전달 시에 SCF에서 어떤 오류나 네트워크 과부하로 인하여 메시지 누락이 발생하면 AS에서 Request를 재전송하는 것이 아니라 MS에서 재전송하여 메시지를 빠른 시간 안에 복구할 수 있고, SCF로 메시지 전송 시에 SCF 응답속도 등을 측정하여 하나의 SCF에 부하가 집중되지 않도록 부하 균등화한다. 그 각각의 기능을 Dispatch Handler와 Ack Handler에서 담당한다.

-Dispatch Handler

Dispatch Handler는 SCF의 응답시간을 측정하여 (그림 7)과 같은 응답시간 테이블은 유지한다. R은 순간순간의 SCF의 응답을 측정한 것이며 RTS(s)는 R을 이용하여 산술 평균을 취한 값이다. MS는 RTS(s)의 값이 가장 작은 SCF를 선택하여 Request를 전송하게 된다.

R= SCF 응답시간
RTS: RTS(s)=(1-a)R+aRTS(s-1)

	SCF1	SCF2	SCF3
R	a`	b`	c`
RTS(s)	a	b	c

	SCF1	SCF2	SCF3
R	1	2	3
RTS(s)	1.5	2.3	3.4

(그림 7) MS의 SCF연결설정을 위한 응답시간 테이블

-Ack Handler

Ack Handler는 Request를 응용서버로부터 수신하였을 때 큐에 저장될 위치를 판단하여, 예상응답을 예측하여 응용서버에게 돌려주게 된다. 큐는 Weighted Round Robin을 적용한다. 큐에 저장되었을 경우 현재 전송중인 어떤 클래스의 큐에서 전송 중인지 판별하여 같은 큐인 경우와 다른 큐인 경우 계산을 달리하게 된다. 그 계산 방법은 다음 <표 1>과 같다.

〈표 1〉 예상응답시간 계산 방법

같은 클래스의 큐인 경우	[1/qn*(rn+mn)*(q(n+1)+q(n+2))+rn+1]*RTS(s)
다른 클래스의 큐인 경우	[(rn/qn+1)*(q(n+1)+q(n+2))+rn-m(n+1)-m(n+2)+1] *RTS(s)

qn : n 클래스 큐에서 패킷을 dispatch하는 비율
mn : n 클래스 큐에서 주기 동안 빠져나간 패킷
rn : n 클래스 큐에 남아있는 패킷

Ack Handler의 동작은 Request를 수신하였을 경우 우선 그 클래스에 해당하는 큐가 가득 차 있는지를 판별하게 된다. Max라면 낮은 등급의 다른 클래스의 큐에 저장하였을 경우와 비교하여 더 빨리 처리가 가능하다면 그 클래스의 큐에 저장한다. 아니라면 drop이 발생하고, drop 카운터를 하나 증가시켜 예상응답시간을 생성한다. 큐에 공간이 있는 상황이라면 큐에 저장하고 앞서 설명한 방식으로 예상응답시간을 생성하여 알려주게 된다.

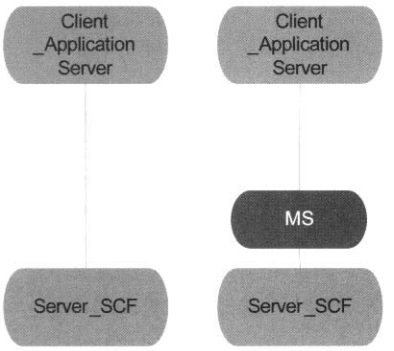
4. 모의실험 및 결과 분석

본 장에서는 앞서 제안한 광대역통합망에서 신뢰성 있는 웹 서비스를 제공하기 위하여 MS를 적용한 경우와 적용하지 않은 경우에 대하여 비교하였다. 모의실험에서 응용서버와 SCF는 AXIS기반의 자바언어로 개발한 응용을 사용하였으며 MS 또한 자바언어로 개발하였다. 네트워크는 다른 트래픽에 영향을 받으면 안되므로 로컬네트워크를 구성하여 모의실험을 진행하였다.

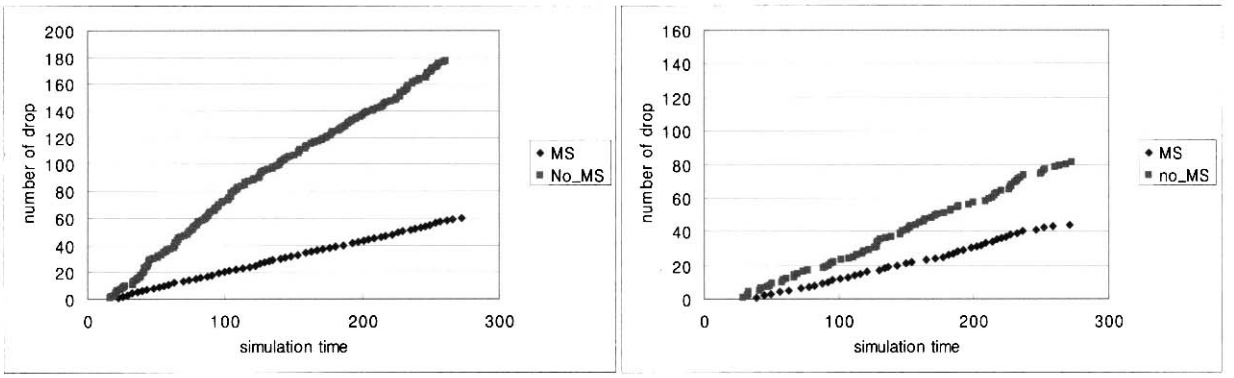
4.1 단일서버 환경에서의 모의실험

4.1.1 모의실험 환경

(그림 8)은 모의실험을 위한 환경을 보여준다. 단일서버 환경에서의 모의실험은 하나의 응용서버에 하나의 SCF가 존재하는 경우이다. 이때 MS의 유무와 큐의 크기차이에 따른 변화에 대하여 비교 하여 보았다.

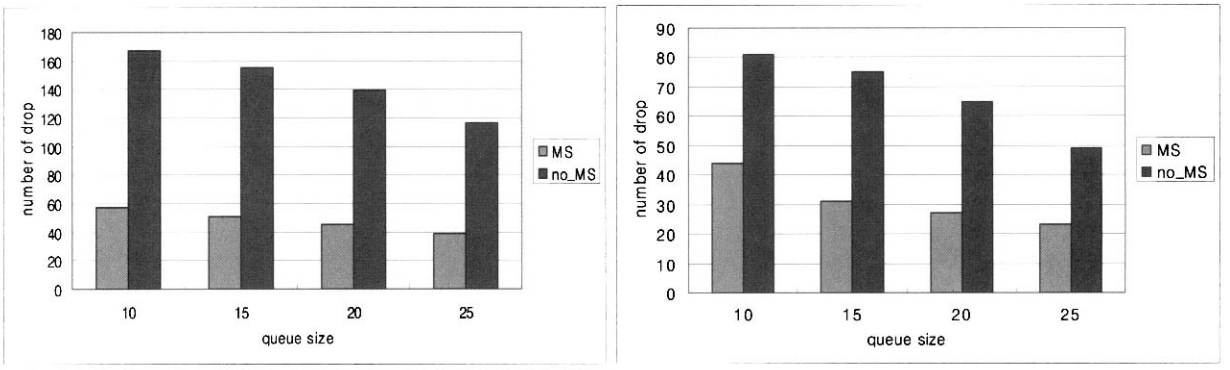


(그림 8) 모의실험 환경



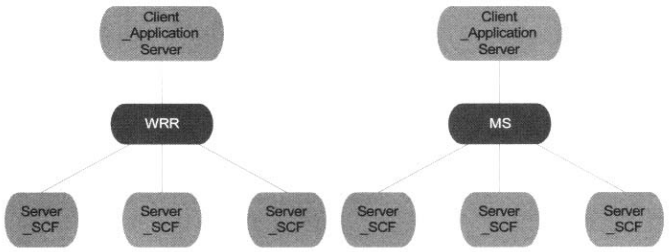
A. SCF 응답 시간 3~6　　B. SCF 응답 시간 2~4

(그림 9) SCF응답시간 차이에 따른 Request의 Drop수 비교

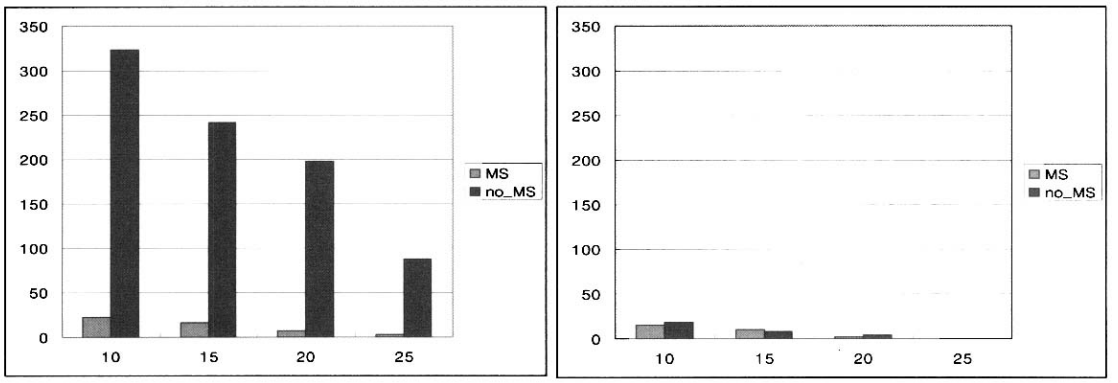


A. SCF 응답 시간 3~6　　B. SCF 응답 시간 2~4

(그림 10) 큐의 크기에 따른 Request의 Drop수



(그림 11) 모의실험 환경



A .과부하 상황에서 큐의 크기에 따른 Drop수

B. 일반적인 상황에서 큐의 크기에 따른 Drop수

(그림 12) 큐의 크기에 따른 Drop수

4.1.2. 모의실험 시나리오

응용서버는 0~3초 사이에서 Request를 생성하게 된다. SCF는 3~6초 사이의 응답시간을 가지는 경우와 2~4초 사이의 응답시간을 가지는 경우로 구분하였다. 또한 각각의 경우에 대하여 큐의 크기를 10, 15, 20, 25로 달리하면서 실험하였다. 실험 시간 동안 응용서버는 100개의 Request를 생성한다.

4.1.3 실험 결과

위의 (그림 8)의 모의실험 환경을 통하여 실험을 수행하였을 경우 결과는 (그림 9)에서 확인할 수 있다. (그림 9)의 X축은 실험 시간이 되겠고 Y축은 Request의 Drop수를 의미한다. 서버에 용량이 부족하여 연결할 수 없는 상황인데 클라이언트의 연결요청이 발생할 때를 Drop으로 간주한다. MS의 적용 유무에 따라 Drop수의 차이가 약 3배정도 나타난다.

(그림 10)은 큐의 크기를 달리하여 실험하여 Drop수를 비교한 것이다. X축은 큐의 크기를 의미하고, Y축은 Drop수를 의미한다. MS의 적용하지 않은 경우 큐의 크기가 커질수록 Drop수가 적어지지만 MS를 적용한 경우 큐의 크기에 따라 큰 차이를 보이지는 않는다. 같은 경우에 Drop수가 적어진다는 것은 예상응답시간 전송을 통하여 서버의 상태를 반영하기 때문에, 서버의 상태를 모르고 클라이언트의 판단만으로 재전송을 하는 방식보다 쓸모 없는 재전송이 줄었다는 것을 의미한다.

4.2 멀티서버 환경에서의 모의실험

4.2.1 멀티서버 환경에서의 실험 환경

(그림 11)은 모의실험을 위한 환경을 보여준다. 멀티서버환경에서의 모의실험은 하나의 응용서버에 3개의 SCF가 존재하는 경우이다. 이때 MS를 적용하였을 경우와 단순히 큐 알고리즘만을 적용하였을 경우를 비교한다.

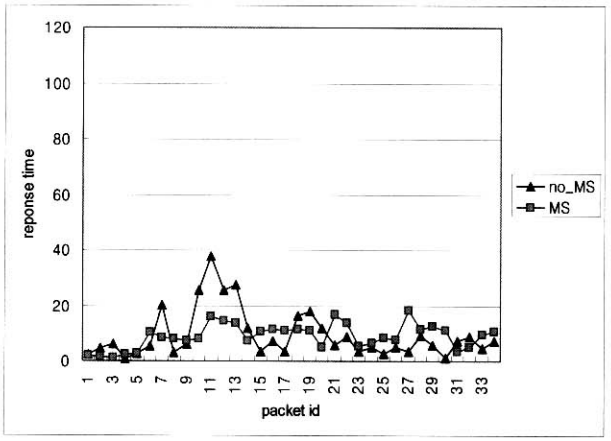
4.2.2 모의실험 시나리오

응용서버는 각 클래스 별로 총 100개의 Request를 생성한다. 응용서버의 요청 생성속도를 조절하여 과부하 상황과 일반적인 상황을 설정하였다. SCF는 2~4초, 2~6초, 3~9초 사이의 응답시간을 가지는 경우로 3가지가 연결된다. 또한 각각의 경우에 대하여 큐의 크기를 10, 15, 20, 25로 달리하면서 실험하였다.

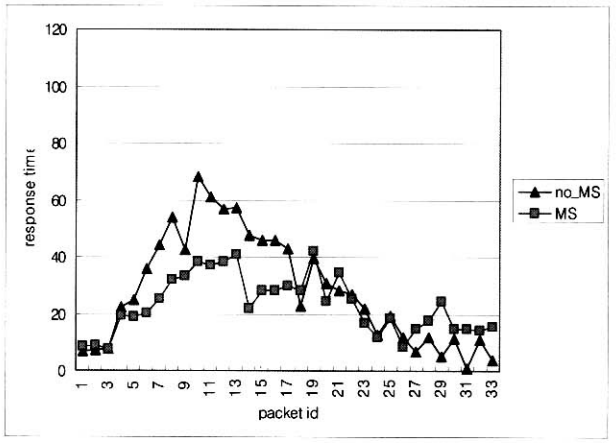
4.2.3 실험 결과

위의 (그림 11)의 모의실험 환경을 통하여 과부화 상황과 일반적인 상황을 가정하여 실험을 수행하였을 경우, 결과는 (그림 12)에서 확인할 수 있다. (그림 12)의 X축은 큐의 크기를 의미하고 Y축은 Request의 Drop수를 의미한다. Drop수가 큐의 크기에 따라 각각 10배정도의 차이를 보인다. 멀티서버 환경에서의 실험이 단일서버 환경에서의 실험보다 Drop수가 많이 증가하는 이유는 단일서버 환경에서의 실험에서는 클래스를 적용하지 않았지만 멀티서버 환경에서의 실험에서는 클래스를 적용하였기 때문이다. drop수는 대부분 클래스3에 몰려있는데 클래스3에서 drop이 발생하면 클래스3의 큐에 자리가 생길 때까지 기다리는 시간이 단일서버 환경에서의 실험에서 큐에 자리가 생길 때까지 기다리는 시간보다 많이 걸리기 때문이다. (그림 12)에서 일반적인 상황과 과부하의 상황을 비교하면 일반적인 상황에서 MS의 역할을 크게 나타나지

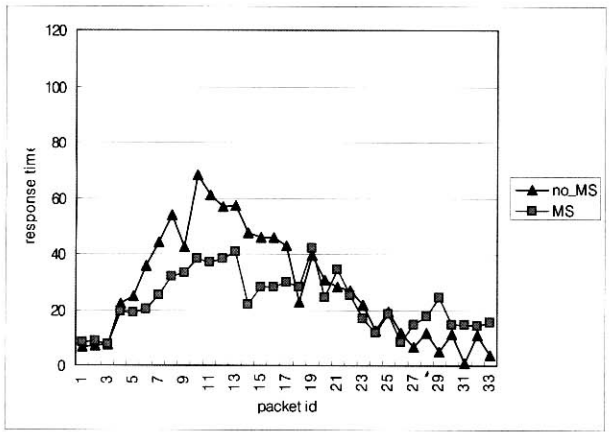
않는다. 하지만 과부하의 상황에서는 응용에게 대기하도록 하기 때문에 무의미한 트래픽을 줄일 수 있다.



(그림 13) MS의 유무에 따른 클래스1의 응답시간



(그림 14) MS의 유무에 따른 클래스2의 응답시간



(그림 15) MS의 유무에 따른 클래스3의 응답시간

(그림 13)에서 (그림 15)는 각 클래스에 따라 처리되는데 걸린 응답시간이다. MS를 적용한 경우에 클래스1은 0-20 사이의 응답시간을 가지고 클래스2의 경우 0~40 사이의 응답시간을 가진다. MS를 적용하지 않은 경우에는 클래스1은 0-40사이의 응답시간을 가지고 클래스2의 경우 0~75사이의 응답시간을 가진다. 이런 결과는 MS에서 SCF의 응답시간을 측정하여 가장 빨리 응답이 가능한 SCF에게 Request를 전송하기 때문에 최적화된 응답시간을 가지기 때문이다. 그러나 클래스3의 경우에는 큐의 대기 시간이 너무 길어져서 응답시간의 차이를 확인하기 어렵다.

한편, 큐에 WRR 방식을 이용하여 각 클래스에 따라 처리순서를 결정하기 때문에 클래스1는 클래스2나 클래스 3에 비하여 상대적으로 빨리 서비스를 받게 된다. 각 클래스에 따른 평균 응답시간은 <표 2>와 같다.

〈표 2〉 클래스별 평균 응답시간

	No_MS	MS
클래스1	10.3497	8.9437
클래스2	28.3282	23.1312
클래스3	63.0861	59.4981

5. 결 론

본 논문에서는 웹 서비스를 광대역통합망에서 효율적으로 동작시키기 위해서 비동기 구조를 적용하여 http202전송을 통한 ack개념을 도입하고 예상응답시간을 전송하여 클라이언트에서 서버의 상황을 적응적으로 대처할 수 있도록 하였다. 또한 SCF의 평균 응답시간을 이용하여 가장 빠른 응답을 받을 수 있는 SCF를 선택하여 Request처리 효율을 증가시켰다.

단일 서버환경의 모의실험 결과에서 확인하였듯이 제안한 MS를 적용하였을 경우와 적용하지 않았을 경우에 drop수가 약 3배정도 차이를 보였다. 또한 멀티서버환경에서의 모의실험에서는 일반적인 상황과 특별하게 과부하의 상황에 대하여 비교하였다. 그 결과 일반적인 상황에서는 MS의 유무에 크게 상관없이 서비스를 받지만 과부하 상황에서는 MS를 적용하여 응용의 요청발생속도를 제어하는 것이 효율적인 것을 보였다. 또한 멀티서버상황에서 WRR과 간단한 부하 균등화를 통하여 클래스 별로 응답속도를 보장할 수 있다.

그 결과를 통하여 SCF의 응답속도가 응용의 서비스 요구속도를 따를 수 없고, 큐의 크기가 작아질수록 즉, 서버의 동접자 수가 작아질수록 MS적용시의 효율을 증가할 것으로 예상된다. 하지만 MS의 적용은 응용의 서비스 요청에 대한처리의 예상하여 알려주어 효율을 증가시켜주지만 응답속도를 빠르게 하는 등의 성능향상에 대해서는 큰 변화를 주지 못한다. 또한 과부하라는 특수한 상황에서 좋은 성능을 보여주지만 일반적인 상황에서는 크게 변화된 모습을 보여주지 못하였다. 따라서 특별한 상황에서 잘 동작할 뿐만이 아니라 어떤 상황에 관계없이 서비스 효율을 증가시켜주고 성능을 개선할 수 있도록 하는 것이 앞으로 해결해야 할 문제라고 생각된다.

참 고 문 헌

[1] Parlay Group, Parlay API Business Benefits White Paper, June, 1999.

[2] Parlay Group, Parlay API Spec. 4.1, December, 2003.

[3] Parlay X Web Service Specification, Version 1.0, June, 2004.

[4] W3C, Web Service Architecture, February, 2004, http://www.w3.org/TR/ws-arch/

[5] W3C, Web Services Description Language(WSDL) Version 2.0 Part 1 : Core Language, March, 2004, http://www.w3.org/TR/2004/WD-wsdl20-20040326/

[6] Marco Conti, Mohan Kumar, Sajal K. Das, and Behrooz A. Shirazi, "Quality of Service issues in Internet Web Service", IEEE, June, 2002.

[7] Asit Dan, Heiko Ludwig and Giovanni Pacifici, "Web service differentiation with service level agreements", IBM, May, 2003.

[8] Eric Schmidt, "Reliable XML Web Service", Microsoft Corporation, December, 2001, http://msdn.microsoft.com/library/enus/dnexxml/html/xml11192001.asp

김 형 민

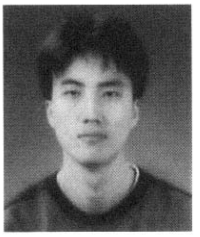
e-mail : meruru98@kw.ac.kr

2002년 광운대학교 전자공학부(학사)

2004년 광운대학교 전자통신공학과(공학석사)

2004년~현재 광운대학교 전자통신공학과 박사과정

관심분야 : BcN, 웹 서비스, 무선 이동통신 등



정 의 헌

e-mail : bbohal@lycos.co.kr

2003년 광운대학교 전자공학부(학사)

2005년 광운대학교 전자통신공학과(공학석사)

2005년~현재 ㈜헤리트 부설연구소 연구원

관심분야 : Open API, 웹 서비스, Parlay X 등

김 화 성

e-mail : hwkim@daisy.kw.ac.kr

1981년 고려대학교 전자공학과(학사)

1983년 고려대학교 전자공학과(공학석사)

1996년 Lehigh Univ. 전산학 박사

1984년~2000년 ETRI 책임연구원

2000년~현재 광운대학교 전자통신공학과 교수

관심분야 : BcN 미들웨어 환경, QoS-aware 미들웨어, 그리드컴퓨팅 등