한국산학기술학회논문지
Vol. 4, No. 4, pp. 380-386, 2003

광역 커뮤니티서비스를 위한 메시징 시스템의 구조 및 QoS 엔지니어링 모델의 설계

궁상환*

Design of Messaging System's Architecture and QoS Engineering Model for Global Community Services

Sang Hwan Kung*

요 약 인터넷의 활용으로 사회는 점차 가까워지게 되고, 지리적, 시간적인 제약을 탈피한 대화 및 정보공유가 더욱 활발해 지고 있다. 또한 최근에는 무선기술의 발달에 힘입어 핸드폰, PDA 등 무선 단말에서의 메시지교환이 더욱 활발해 짐에 따라, 기업이나 학교 등 공식적인 조직 뿐 아니라 취미나 관심을 같이하는 사람들끼리의 비공식적인 조직이 급격히 활성화 되는 추세에 있다. 이러한 새로운 변화는 채팅, 메신저, 단문메시지, 전자게시판, 동영상 파일 공유 등 다양한 커뮤니티 도구들을 활용하여 더욱 활성화 되고 있고, 메시징 시스템은 이러한 응용들을 효과적으로 지원하는 도구로써 활발히 활용될 것으로 예상된다. 본 논문은 다양한 광역 커뮤니티 응용을 지원하는 메시징 시스템의 설계에 초점을 두고 있다. 서비스 컨텐츠의 종류나, 메시지의 길이 등 다양한 요건을 충족하는 엔진의 설계 뿐 아니라 이질적인 분산 환경에서 메시징 서비스의 품질을 최적화하기 위한 서비스 품질관리방법과 메시징 시스템에의 적용시의 효과적인 방안을 제안하고 있다.

Abstract As Internet is spreaded widely, the virtual communities among people with the same interest and hobby are dynamically created. The exchange of fruitful opinion and information makes them to be closed, not only with the help of the messaging system but also wireless terminals. The messaging system is a core technology supporting the global community services like chatting, messenger, short message services, and file sharing services. The paper focus on the design of software architecture for messaging system and the engineering model for QoS support.

Key words : messaging system, QoS, JMS, community service

1. 서 론

통신망과 인터넷의 발달은 데이터를 이용한 편리한 정보교환을 확산시키는 데 많은 기여를 하였다. 과거 데이터를 이용한 사람간의 통신수단은 전자우편(e-mail)을 중심으로 발달되어 왔으나 최근에는 단문메시지나 메신저, 채팅 등을 활용함으로써, 메시지 교환방식도 전자우편시스템과 같이 단방향으로 정보를 전달하는 방식에서, 양방향 그리고 상호작용 위주로 정보를 교환하는 방식으로 변화하고 있다. 메시징 시스템은 기본적으로 비동기식 방식에 의하여 메시지를 전달하는 방식이지만, 메시지의 송신자와 수신자가 같은 시간에 존재할 경우 동기식, 실시간 서비스를 위해 충분히 활용 가능한 서비스이다.

또한 최근에는 통신 대역폭의 제공능력이 확장됨에 따라 문자 외에도 음성이나 그래픽, 동영상을 실시간으로 교환하는 서비스가 활성화되어 다채널 멀티미디어 메시징 서비스가 더욱 요구되고 있다.

이러한 광역 커뮤니티 서비스는 서비스의 유형은 다양하지만 메시지를 전달하는 방식에서는 메시징 시스템이라는 공통된 프레임워크를 토대로 한다. 본 논문은 여러 유형의 메시지를 효과적으로 전달하는 메시징 시스템의 설계에 관한 연구로, 무엇보다 다양한 서비스 요구사항을 충족시키기 위한 기능적, 구조적 설계상 고려와 함께, 대규모의 사용자들과 분산된 사용자 집단을 고려한 서비스 품질의 최적화를 함께 달성하고자 연구되었다.

우선, 2장에서는 광역 커뮤니티 응용을 중심으로 메

*천안대학교

시징 시스템의 요구사항을 분석하고, 3장에서는 이러한 요구사항을 만족하기 위한 메시징 시스템의 전체적, 부분적인 설계를 기술한다. 4장에서는 메시징 시스템의 성능최적화를 달성하기 위한 QoS의 설계를 다루고, 5장은 연구내용의 평가와 향후연구과제들을 중심으로 결론을 맺고자 한다.

2. 광역 커뮤니티 서비스를 위한 메시징 시스템의 요구사항

최근 인터넷에서 많이 활용되는 커뮤니티 서비스로는 채팅이나 게시판, 자료공유 등을 들 수 있다. 이러한 서비스들의 특징은 임의의 그룹에 속한 사용자들 간에 메시지를 전달하거나 교환한다는 점이다. 채팅 서비스의 특징이라면 각각의 메시지 길이는 짧은 반면, 전달되는 메시지는 다대다의 연결자간에 상호 교환된다는 점을 들 수 있다. 이에 비해 게시판의 경우는 임의의 사용자가 지정하는 하나의 게시판에 정보를 게재 즉, 전달한다는 점을 들 수 있다. 자료공유의 경우는 채팅이나 게시판이 짧은 메시지를 다룬다는 점에 비해 정보가 파일과 같이 긴 메시지를 대상으로 한다는 점이 다르다고 하겠다.

메시징 엔진은 이러한 요구사항을 공통적으로 지원하는 기반시스템이 된다. 선 마이크로시스템사를 중심으로 규격화된 JMS(Java Message Service)는 이러한 메시징 시스템이 갖춰야 할 API 차원의 표준규격을 제공한다. 따라서 JMS에서 제공하는 서비스 모델과 기본적인 API는 본 연구를 위한 중요한 요구사항이 된다.

JMS에서 소개하는 메시징 시스템의 서비스 모델은 Figure 1에 예시된 바와 같이 크게 두 가지로 구분된다. 하나는 point-to -point 모델이며, 다른 하나는 publish-and- subscriber 모델이다. Point-to-point 모델은 메시징 클라이언트가 queue라고 불리는 가상 채널을 이용하여

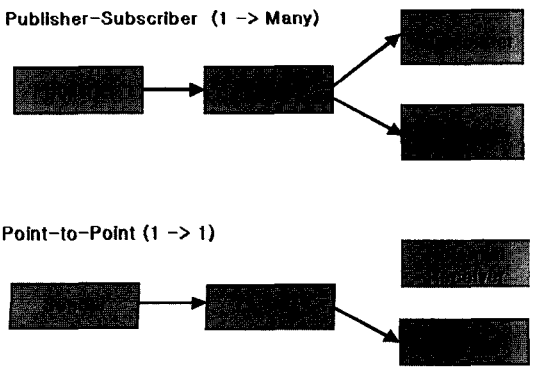

Fig. 1. 메시징 서비스 모델

송신자와 수신자가 일대일의 관계로, 동기식 또는 비동기식으로 메시지를 송수신하는 방식이다. 이에 비해, publish-and-subscribe 모델은 메시지 출판자(publisher)와 구독자(subscriber) 사이의 관계가 일대다의 연결형태를 구성하며, 이들간 topic이라는 가상채널을 통해 메시지를 전송하는 통신방식이다.

또 다른 메시징 엔진의 요구사항은 W3C의 규격인 SMIL(Synshronous Multimedia Integration Language)의 멀티미디어 연출규격을 고려할 수 있다. 멀티미디어 연출의 경우 하나의 세션에서 음성이나 텍스트, 동영상이 번갈아 연출되어야 한다. 따라서 이를 위해서는 각각의 미디어 채널의 서비스 정보와 함께 데이터 정보를 통제하기 위한 제어정보가 하나의 세션 내에서 동시에 전달 될 수 있는 메카니즘이 제공되어야 한다.

이와 같이 본 연구의 요구사항은 대규모의 분산 클라이언트들의 즉, 채팅이나 메신저, 화상회의, 공동저작 등의 그룹웨어 응용 뿐 아니라, 멀티미디어 연출과 같이 시간 축 상에서 다양한 미디어의 연출 시나리오를 원격에서 play 시킬 수 있도록 멀티미디어 정보를 제어하면서 전달시킬 수 있는 데이터 전달 및 교환기능, 그리고 SUN사를 중심으로 표준화된 JMS(Java Messageing Service)의 메시징 서비스 모델을 충족하는 것이라고 하겠다. 이러한 요구사항을 중심으로 본 연구에서는 네트워크 분산환경에서 대규모 집단 사용자의 메시징 요청을 효율적으로 처리하도록 하기 위한 분산 메시징 시스템의 구조와 연결방법을 설계하고, 메시징 시스템이 갖는 성능영향 요소를 분석하여 메시징 엔진을 설계 및 구현 하였다.

3. 메시징 시스템의 설계

3.1 메시징 시스템의 구조

메시징 시스템의 구조는 Figure 2에서 보는 바와 같이 크게 메시지 송신의 역할을 담당하는 메시지 출판부분과 메시지 수신의 역할을 담당하는 구독부분, 그리고 메시지 출판자와 수신자를 관리하는 부분과 메시지 자체의 저장검색을 지원하는 부분으로 구분된다.

Figure 3의 우측 그림은 메시지 통신을 위한 계층구조를 보여 준다. 제일 하위계층인 채널계층은 소켓을 이용하여 종단간 양방향으로 정보를 전달하는 계층이다. 그 위의 계층인 세션계층은 응용 프로그램의 한 세션을 위한 계층이며, 한 세션은 하나 또는 여러 개의 서비스를 수용할 수 있다. 예를 들어, 멀티미디어 전송에서 음성데이터와 영상데이터를 별도의 서비스로 보낼 수가 있다. 최종계층인 응용계층은 point-to-point 또는 publisher-subscriber 서비스를 형편에 따라 이용할 수 있다. 응용계층은 이러한 서비스를 위해 하나의 채널을 활용할 수

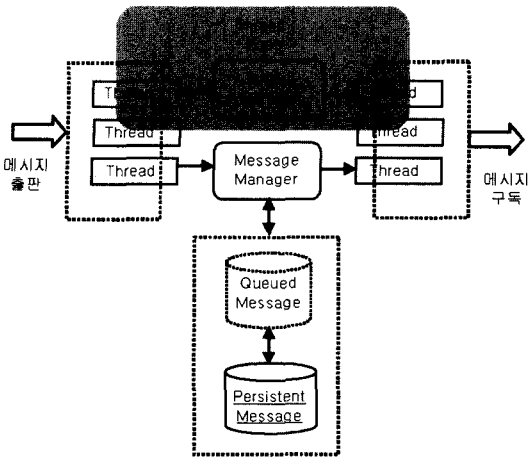

Fig. 2. 메시징 시스템의 기본구조

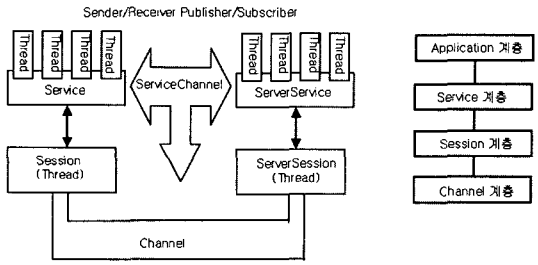

Fig. 3. 메시지 통신을 위한 계층구조

도 있으며, 경우에 따라서는 복수의 채널을 각각 복자적인 서비스를 이용하여 활용할 수 있도록 설계되었다.

한편 Figure 3의 좌측 그림은 클라이언트와 서버간의 통신을 하는 경우를 모델로 메시징 시스템의 통신계층을 설명하고 있다. 특히 이 그림에서는 각 세션과 서비스들은 독자적인 스레드로 동작시켜, 최적의 성능을 유지할 수 있도록 고려하고 있음을 알 수 있다.

3.2 스레드풀의 설계

대규모 접속을 가정하는 메시징 시스템에서 메시징 서버의 성능에 영향을 주는 상황은 동시에 많은 메시지 사용자가 서버에 접속할 때 발생한다. 특히 메시징 서버에 메시지를 출판하거나 구독할 때 서버는 각각의 클라이언트별로 메시지의 송수신을 처리할 서버측의 스레드 할당이 필요하다. 여기서 고려해야 하는 문제는 우선 스레드가 무한정 생김으로써 발생할 수 있는 장애이며, 또 한가지는 스레드를 생성하는 데 소요되는 시간의 문제이다. 여기서 전자는 일정한 크기의 스레드만을 유지하도록 하여 스레드가 무한정 증가하는 것을 막을 수 있고, 후자에 대한 처리방법은 미리 스레드를 생성해 둔 후 필요시 처리기능을 이 스레드에 연결시켜 매번 스레드를 구동하는 시간을 절약하는 방법이 있다. 실험을 위해 먼저 스레드를 생성시키고 start()를 호출하여 실제 스레드가 동작하게 되기까지의 시간소요를 예제 프로그램의 작성을 통해 살펴보았다. 즉, 1000개의 문자열을 만드는 데 소요되는 시간(1)과 1000개의 스레드를 만드는 데 소요되는 시간(2), 그리고 1000개의 스레드를 만듬과 동시에 이를 실행시키는 시간(3)을 비교, 실험한 결과를 요약하면 다음과 같다.

(1)의 경우 0.006 milisecond
(2)의 경우 0.066 milisecond
(3)의 경우 2.433 milisecond

즉, 이 결과는 메시징 서버에서 스레드를 실행시키는 시간이 성능 저하에 중요한 요인이 되고 있음을 보여주고 있다.

스레드의 생성 및 기동시간은 다른 처리에 비해 많은 시간이 소요되므로, 대규모의 클라이언트 연결을 대비하기 위하여 일정 수의 스레드를 스레드풀 안에 먼저 생성하는 것이 필요하게 된다. 이후 메시지를 생산하거나 또는 구독하기를 희망하는 클라이언트가 접속을 요청하면 클라이언트의 요청을 처리할 객체를 스레드에 연결시키도록 하여 클라이언트의 요청을 받아들일 때마다 스레드를 생성하는 오버헤드를 없앨 수가 있다.

Figure 4는 사전에 스레드풀을 형성하고 생성된 스레드에 클라이언트의 요청을 담당할 객체를 연결시키는 과정을 설명하고 있다. 스레드풀에 생성된 각각의 스레드들은 클라이언트의 연결 요청을 처리할 객체들을 담고 있는 일종의 큐 역할을 수행하는 벡터를 지속적으로 감시한다. 클라이언트의 연결요청을 처리하는 객체는 현재 처리할 객체가 할당되지 않았거나 할당된 객체가 메시지 처리를 완료하여 가용상태에 있는 경우, 클라이언트로부터 연결요청을 받아들이게 되며, 메시지 출판

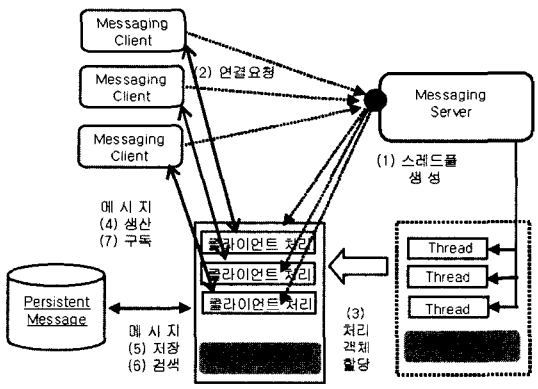

Fig. 4. 스레드풀 관리구조

의 경우는 클라이언트의 메시지를 수신하여 저장하고, 메시지 구독의 경우는 저장된 메시지를 읽어서 클라이언트에 송신을 해준다.

3.3 메시지 관리 구조

메시징 시스템은 기본적으로 한 사용자에 의해 저장된 메시지를 다른 사용자들에게 전달해 주는 것이다. 메시징 시스템은 전반적인 메시지 관리를 위해 별도의 메시지 관리자를 운영한다. 입력되는 모든 메시지는 이 관리자에 의해 등록되어 이 메시지를 기다리는 사용자들에게 분배되며, 아울러 유효기간이 지난 메시지들의 폐기처리 역할도 담당한다. 메시징 시스템의 클라이언트-서버간 통신프로토콜에서 사용하는 메시지의 유형은 다음과 같다. 세션을 처리하는 메시지, 서비스를 처리하는 메시지, 그리고 데이터를 전송하는 메시지들로 구성된다.

(1) 세션 처리 메시지
BEGIN_OF_SESSION
BEGIN_CONFIRM
SESSION_END
END_CONFIRM

(2) 서비스 처리 메시지
SERVICE_OPEN
OPEN_CONFIRM
SERVICE_DATA
DATA_ACK
SERVICE_CLOSE
CLOSE_CONFIRM

(3) 데이터 메시지
TEXT_MESSAGE
BYTE_MESSAGE
OBJECT_MESSAGE

메시지의 저장은 크게 두 가지 방식에 의한다. 한 가지는 실시간 서비스를 위한 성능요건을 만족하기 위해 메모리 버퍼를 활용하는 방식이며, 다른 하나는 신뢰성을 보장하기 위해 파일버퍼를 이용하는 방식이다. 또한, 저장과 동시에 하나의 메시지는 단 하나의 구독자에 의해 활용될 수도 있지만 복수의 여러 구독자들에 의해서도 활용될 수 있다는 점을 고려해야 한다. Figure 5에서 MessageEntry 클래스는 서버에 저장되는 특정한 엔트리를 관리하는 모듈이며, 신뢰성 보장의 필요성에 따라 메모리 버퍼 또는 파일 버퍼에 구분되어 저장된다. 저장버퍼의 유형에 따라 메시지 전달 성능에도 많은 차이

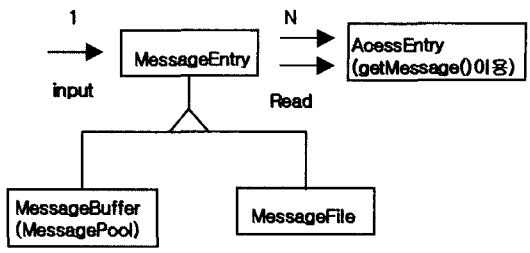

Fig. 5. 메시지 저장 및 접근구조

가 있게 된다. 이 그림에서 AccessEntry 클래스는 복수의 메시지 활용 클라이언트를 고려해서 별도로 설계된 클래스이며, 클라이언트별로 객체가 생성된다.

3.4 오류 및 흐름제어

메시지 서비스 사용자가 사용하는 데이터 채널은 ServiceChannel이라는 객체를 활용한다. 이 채널은 궁극적으로 소켓통신을 이용하는 Channel 객체를 통하여 정보를 교환하지만, Channel 객체에서는 고려하지 않는 오류 및 흐름제어를 담당한다. 즉, 각각의 미디어 정보가 전송되는 서비스 채널 단위로 오류 및 흐름제어가 적용되고 있다. 이 과정은 기본적으로 파라미터로 지정된 윈도우 크기를 토대로 오류 및 흐름제어가 수행되며, 세부적인 수신측과 송신측의 수행 알고리즘은 다음과 같다.

(1) 수신확인처리 알고리즘 // acknowledge()

```
if(수신 메시지의 번호 < 수신 예정 메시지의 번호)
오류값 return; 기 수신 메시지의 중복 수신
if(수신 메시지의 번호 > 이전 수신메시지의 번호)
{
NAK 메시지를 생성함;
NAK 메시지 번호 <- 이전 수신 메시지의 번호;
// 정상적으로 받은 최종 메시지 번호 통보
NAK 메시지를 전송함;
ACK 카운터를 clear함;
return;
}
if(수신 메시지의 번호 == 수신 예정 메시지의 번호)
{
ACK 카운터 증가; // 한 윈도우 내에서 몇 개의 데이터를 받았는가 계산
if(ACK 카운터 > 윈도우 크기)
{
ACK 메시지 생성;
ACK 메시지 번호 <- 최종 수신 메시지 번호;
ACK 메시지 송신;
```

```
}
return;
}
```

(2) 송신측 재전송 알고리즘 // retransmit()

```
/*
 * 별도의 스레드가 메시지 송신결과에 대한 수신측 확인정보를 읽어 버퍼에 저장
 */
수신확인 메시지 버퍼 read;
if(메시지== null)
return; // 아직까지 수신된 확인 메시지 없음
if(수신확인 메시지의 번호 >= 최종 메시지 번호)
return; // 최종 메시지 : 메시지 송신자가 메시지 채널을 close 한 경우
do {
if(재전송 메시지 버퍼 크기 <= 0)
break; // 버퍼 처리 종료
버퍼의 메시지 read;
if(메세지 == ACK 메시지)
break; // 최종 확인 메시지에서 버퍼 clear 동작을 중지
else // NAK 메시지 또는 ACK 메시지이나 아직 clear할 버퍼의 메시지가 남아 있는 경우 버퍼에 있는 메시지 재전송;
} while(true);
```

4. 메시징 시스템의 QoS 모델

4.1 메시징 시스템의 QoS 요구사항

Table 1은 메시징 시스템의 QoS 관련 주요자원과 이러한 자원의 관리요건에 대해 소개하고 있다. 각각의 자원은 여러 가지 제약조건하에서 운영되기 때문에 서비스 품질을 저해하는 요소들을 분석하여 최적의 성능을 보장할 수 있도록 자원의 규모를 설정하고 통제하는 일이 필요하다.

4.2 메시징 시스템의 QoS 개발절차

메시징 시스템의 성능평가를 포함하는 전반적인 QoS 개발절차는 비단 메시징 시스템이 개발이 완료된 이후에 수행해야 할 업무는 아니며, 개발 초기의 요구사항 정립단계부터 운영단계에 이르기까지 단계적으로 고려되어야 한다.

Figure 6은 메시징 시스템의 소프트웨어 개발단계에 따른 QoS 개발절차를 명시하고 있다. 이절차를 정의하는 중요한 이유는 메시징 시스템과 같이 성능 등 QoS 요건이 중요하게 다루어 져야 하는 시스템은 명확한 QoS 요구사항의 정의와 함께, QoS 요건을 만족하기 위한 구조설계, 구현결과에 대한 QoS 시험, 최종 튜닝과 아울러 시스템 활용환경에 맞는 QoS 파라미터 및 적용함수의 구성이 체계적으로 수행되어야 하기 때문이다.

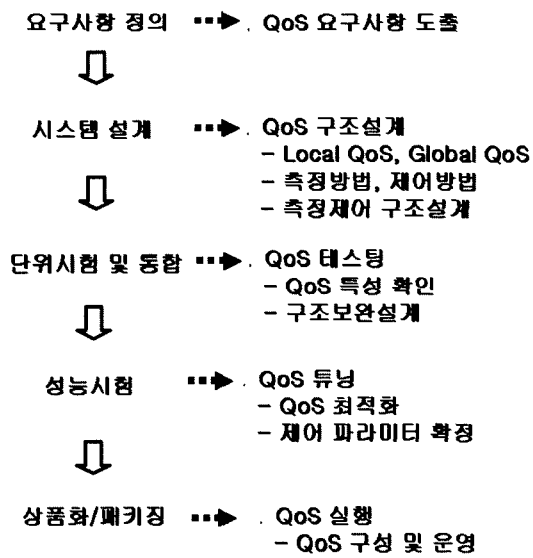

Fig. 6. 메시징 시스템의 QoS 개발절차

Table 1. 메시징 시스템의 QoS 요구사항

QoS 관련 자원	발생 현상	발생 사유	제어 방법
o CPU o 메모리, 버퍼 o 타이머 o 네트워크 o 디스크, DB	o 일부 스레드 동작 지연 o 스레드 처리상 O/H o 버퍼에 넣지 못해 정지 o 너무 크면 처리 지연 o 너무 작으면 다른 task에 지장 o 데이터 손실, 재전송 O/H o 동시 사용자로 인한 지연 o Lock으로 인한 사용 지연	o Thread 초과 할당 o 크기, 동시 사용 o 시간간격 o 오류, 폭주, 동시사용자 o 동시용자, 사용량, 횟수	o 동시 사용자 수 제한 o Thread Pool 활용 o 버퍼크기의 동적 조정 o 증가, 감소 o QoS감시, 윈도우 조정 o 동시 사용자 제한

4.3 메시징 시스템의 QoS 테스팅

메시징 시스템의 성능 측정은 크게 두 가지 관점에서 이루어진다. 하나는 개발된 메시징 엔진이 어느 정도의 효율을 보일 것인가 하는 관점에서 수행하는 블랙박스 시험이다. 블랙박스 시험은 메시징 시스템에 복수의 출판자와 구독자 또는 송신자와 수신자 각각의 개별적인 성능 특성 보다는 집합적으로 집단 전체의 특성에 관심이 있다.

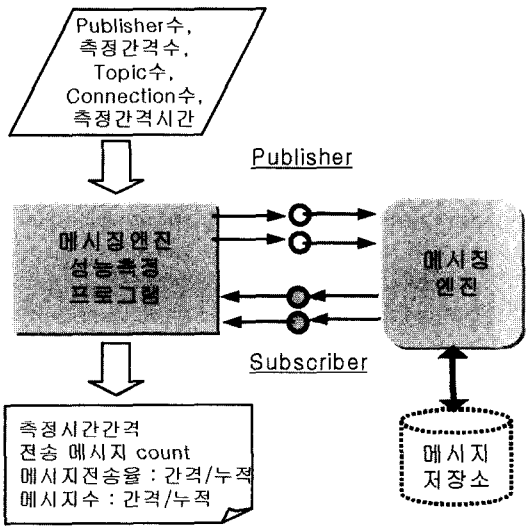

Fig. 7. 블랙박스 시험환경

다른 하나는 메시징 엔진 내부의 구조나 사용자원의 크기 활용방법에 따른 활용을 성능을 측정하기 위한 화이트박스 시험이다. 즉, 개별적인 메시징 연결이나 세션의 성능 특성을 관찰하게 되는 데, 예를 들어 많은 메시징 사용자가 동시에 서버에 연결하여 메시징 서비스를 받고자 할 때 서버의 한계로 일정 수의 스레드는 메시지의 전달에 바로 참여하지 못하고 스레드가 스케쥴링 될 때 까지 기다려야 하는 상황이 발생한다. 이러한 상황을 지원하기 위해서는 시간 간격별이 아니라, 각각의 메시징 세션에 대한 측정이 수행되어야 한다. 특별히 이 경우는 단순한 시험을 위한 것이 아니라 측정에서 발견된 문제를 대처하기 위해 파라미터를 변경하는 등의 제어 행위로 연결됨으로써 수행환경에 동적으로 적응하는 최적의 시스템이 가능하게 된다.

4.4 성능측정 라이브러리

블랙박스 시험이나 화이트박스 시험을 수행하기 위해서는 측정 및 시험을 위한 도구가 요구된다. 본 연구에서는 실시간으로 QoS를 측정하기 위해 메시징 엔진

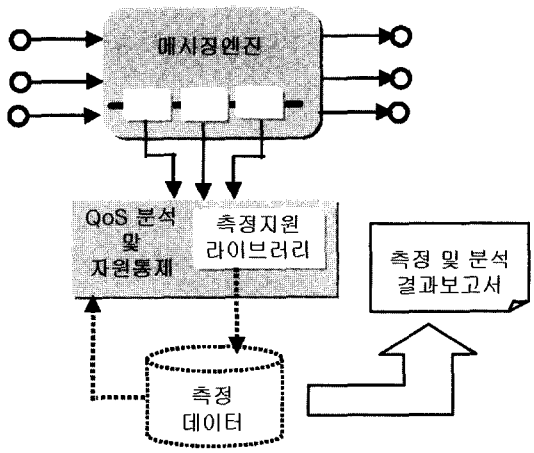

Fig. 8. 화이트박스 시험환경

의 설계개념과 일치하는 방식으로 QoS 측정모델을 Figure 9와 같이 설계하였다. 이 모델에서 전체적인 성능측정과정은 하나의 측정서비스(MeasurementService)로 표현된다. 각각의 서비스나 세션에 대응하는 측정세션(MeasurementSession)을 두고 각각의 측정세션은 여러 개의 시간간격으로 구성되도록 하였다. 이 간격을 측정트랜잭션(MessionTransaction)이라고 지정하였다. 또한 각 측정세션은 소요시간이나 전송데이터수와 같은 다양한 측정치를 포함한다. 측정 라이브러리에 포함된 주요 클래스는 다음과 같다.

o MeasurementService 클래스 : 하나의 측정 시나리오를 위한 총괄적인 관리를 담당하는 클래스이다. 여러 개의 측정 세션을 포함하며 각 세션에 해당되는 처리결과를 저장하기 위한 세션 테이블을 관리한다.

o MeasurementSession 클래스 : 복수 개 측정간격의 측정정보를 저장 하고 관리한다. 각 단위 시간의 측정정보를 저장하기 위해 측정 트랜잭션 테이블을 관리한다.

o MeasurementTransaction 클래스 : 하나의 세션 내

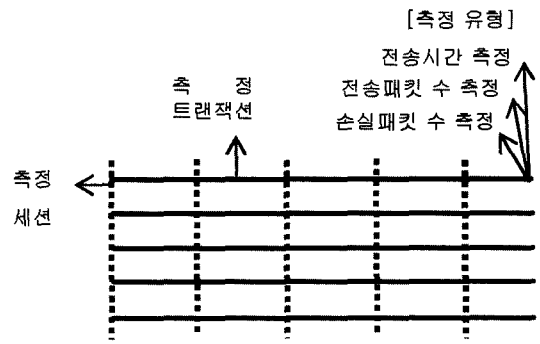

Fig. 9. 성능 측정 윈도우 개념

의 단위시간 동안 측정된 정보를 저장 및 관리한다.

o TimeMeasurement 클래스 : 하나의 간격동안의 소요시간을 측정하는 경우에 사용되는 클래스이다.

o CountMeasurement 클래스 : 하나의 간격동안의 발생횟수를 측정하는 경우에 사용되는 클래스이다.

5. 결론 및 향후 연구과제

본 논문에서는 커뮤니티 서비스의 공통요구사항을 지원하는 메시징 시스템의 구조의 설계에 관하여 기술하였다. 또한 메시징 시스템이 분산 광역 환경에서 대규모의 사용자 집단을 고려하여, 최적의 서비스 품질을 실현하기 위한 QoS 엔지니어링 모델을 함께 제시하였다. 현재 본 시스템은 자바로 프로그래밍하여 기능시험이 완료된 상태에 있고, 부분적인 성능시험이 진행 중에 있다. 이 시험이 완료되는 대로 논문에서 언급한 응용들과 통합되어 활용될 예정에 있다.

논문에서 언급되지 않은 또 다른 설계특징이라면, 본 시스템이 객체지향방식으로 개발됨에 따라 메시지 버퍼라든가 멀티 스레드 구현에서 다양한 디자인패턴이 활용되었다는 점을 들 수 있다. 따라서 결과소스를 통해 디자인패턴을 쉽게 학습할 수 있다는 점도 본 연구를 통한 부대적인 이득이 되었다고 하겠다.

향후 연구과제로는 개발된 엔진을 이용하여 복수의 메시징 엔진 사이의 클러스터링을 통한 확장과 복수 엔진 사이에서의 QoS 분배 및 최적화 등의 분야에서 지속적인 연구를 수행할 예정이다.

참고문헌

[1] Richard Monson-Hoefel & David A. Chappel, Java Message Service, O'Reilly, 2001.

[2] P.Giotta, S.Grant, M.Kovacs, S.Maffeis, K.Morrison, G. Raj, M.Kunnumpurath, Professional JMS Programming, 2000.

[3] Gopalan Suresh Rah, Java Message Service, "http://www.execpc.com"

[4] A Fiorano White Paper - A Guide to Understanding the Pluggable, Scalable connection Management(SCM) Architecture in FioranoMQ5, Feb. 2001.

[5] A Fiorano White Paper - A Benchmark Comparison Between FioranoMQ and IBM MQSeries, June 2001.

[6] Benchmarking JMS Based E-Business Messaging Providers, Java Developer's Journal, Mar. 2001.

[7] 천영환 역, 자바스레드, 인포북, 2001.

[8] 진광일, 박성철 역, 자바스레드, 한빛미디어, 2000.

[9] 홍상욱 역, 자바 퍼포먼스 튜닝, 한빛미디어, 2001.