

데이터 분배 서비스 시스템 설계 및 분석

(Design and Analysis of the Data
Distribution Service System)

박 충 범 [†] 권 기 정 [†]
(Choong-Bum Park) (Ki-Jeong Kwon)

차 다 함 [†] 최 훈 ^{**}
(Da-Ham Cha) (Hoon Choi)

김 점 수 ^{***}
(Chum-Su Kim)

요 약 통신 미들웨어는 응용프로그램에서 담당하던 데이터 교환을 대행한다. 여러 가지 통신 미들웨어 기술이 있지만, 다양한 디바이스들이 동적으로 네트워크 도메인을 형성하고 동일한 타입의 데이터를 빈번히 주고받는 통신환경에서는 데이터 중심 발간/구독 방식의 데이터 교환이 적합하며 OMG의 DDS(Data Distribution Service)에서 이러한 방식을 표준으로 채택하였다. 본 연구에서는 OMG의 DDS 표준 규격을 준수하고 시스템 관리 자동화가 가능한 데이터 분배 서비스 시스템인 ReTiCoM을 설계하고, 그 성능을 유사한 미들웨어인 JMS와 비교 분석하였다.

키워드 : 데이터 분배 서비스, 데이터 발간/구독

Abstract : the data-centric publish/subscribe middle-

- 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다. (계약번호 UD070030AD)
- 이 논문은 2007 한국컴퓨터종합학술대회에서 '데이터 분배 서비스 시스템 설계'의 제목으로 발표된 논문을 확장한 것이다

[†] 비 회 원 : 충남대학교 컴퓨터공학과
here4you@cnu.ac.kr
kjkwon@cnu.ac.kr
dhcha@cnu.ac.kr

^{**} 정 회 원 : 충남대학교 컴퓨터공학과 교수
hc@cnu.ac.kr

^{***} 비 회 원 : 국방과학연구소 해국전투체계개발팀
chskim@add.re.kr
논문접수 : 2007년 10월 1일
심사완료 : 2008년 1월 11일

Copyright©2008 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 작품의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제2호(2008.4)

ware is suitable for a communication environment in which various devices dynamically forms a network domain and same type of data are frequently exchanged. For this purpose, OMG has standardized DDS (Data Distribution Service) specification. In this study, we designed the RiTiCoM, data distribution service system that observes the OMG DDS (Data Distribution Service) standard specification and supports the automation of system management, and analyzed the performance and compared with the JMS.

Key words : DDS, Data Publish-Subscribe

1. 서 론

IT기술의 발달로 다양한 디바이스와 통신 서비스가 출현하였고 이러한 디바이스와 서비스의 출현과 함께 여러가지 통신 기법이 개발되었다. 하지만 통신 기법의 확장은 이에 종속되는 응용프로그램의 개발을 어렵게 한다.

응용프로그램간의 직접적인 데이터 교환을 구현하는데 따르는 응용프로그램 개발자의 부담을 줄이기 위해서는 표준화된 통신 기법과 데이터 교환을 전담하여 대행하는 통신 미들웨어가 필요하며 CORBA(Common Object Request Broker Architecture)나 JMS(Java Message System)와 같은 통신 미들웨어가 개발된 바 있다[1,2].

그러나 통신 장비들이 동적으로 도메인을 형성하고 동일한 타입의 데이터를 빈번하게 주고받는 환경에서는 데이터 중심의 발간/구독(publish/subscribe) 방식이 적합하다. 이러한 환경에 적합한 통신 미들웨어의 표준 규격으로서 OMG(Object Management Group)의 DDS (Data Distribution Service)가 있는데 응용프로그램 및 하부 통신 네트워킹 기능과의 인터페이스를 정의하고 있다[3,4].

본 논문에서 제안하는 데이터 분배 서비스 시스템 (ReTiCoM: Real Time Communication Middleware)은 DDS 인터페이스 규격에 준수하여 응용프로그램들을 하나의 네트워크 도메인으로 묶고 그간의 효율적인 데이터 교환을 제공하고, 나아가 시스템 관리자의 개입 없이 응용프로그램의 배치나 고장 복구 등의 시스템 관리의 자동화(automation)가 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 주요 통신 미들웨어의 연구 동향에 대해 소개하고 3장에서는 데이터 중심의 발간/구독 방식의 특징에 대해 기술한다. 4장에서는 본 연구에서 제안하는 데이터 분배 서비스 시스템의 내부 구조와 QoS 제어를 통한 효율성 향상 기법에 대해 설명하고 끝으로 5장에서 성능 비교 분석 결과를 소개한 후 결론 및 향후 과제를 기술한다.

2. 통신 미들웨어

일반적으로 네트워크와 서비스 환경에 적합한 통신 프로토콜에 응용프로그램이 종속되게 된다. 응용프로그램이 통신 프로토콜에 종속되는 문제는 응용 계층과 운영체제 계층의 사이에서 통신 기능을 대행하는 미들웨어를 통해 해결할 수 있다.

2.1 CORBA(Common Object Request Broker Architecture)

CORBA는 분산 컴퓨팅환경에서 서로 다른 원격 컴퓨터의 오브젝트 정보의 교환을 목적으로 하며 OMG에서 정의되었다. 원격지에 위치한 오브젝트의 액세스(access)에는 유용하지만 비교적 짧은 길이의 데이터를 신속히 주고받아야 하는 환경, 즉 데이터 중심의 데이터 분배를 목적으로 하는 시스템에는 적합하지 않다[1].

2.2 JMS(Java Message System)

JMS는 SUN사의 Java 기반 메시지 전송 방식으로 유선 네트워크로 연결된 기업 업무 시스템에 대규모 데이터 전송에 적합하다[2]. 하지만 QoS 지원 기능이 부족하고, 저장 후 전달(store-and-forward) 전송방식으로 인한 오버헤드(time/space)가 커서 실시간 시스템에는 적합하지 않다.

2.3 DDS(Data Distribution Service)

DDS는 데이터 중심의 실시간 데이터 분배를 목적으로 OMG에서 정의되었다[3]. 도메인에 속한 불특정 다수에게 데이터를 발간 및 구독이 가능하다.

3. 데이터 중심의 발간/구독

도메인(domain)내의 노드들이 동일한 타입의 데이터를 빈번하게 교환할 경우에는 데이터 중심 통신(data-centric communication) 기법이 적합하다. 또한 도메인에 참여하는 노드들의 수가 많고, 동적으로 등록과 등록해제가 이루어지는 경우에는 응용프로그램에서 데이터를 교환할 상대를 찾고 연결을 맺는 서버/클라이언트 방식의 데이터 교환보다는 상대에 대한 인지 없이 자신이 원하는 데이터를 발간(publish)/구독(subscribe)하는 방식의 데이터 교환이 더 효과적이다. 데이터를 발간하는 Publisher들과 데이터를 구독하는 Subscriber들이 모여서 하나의 도메인을 설정하고 서로 약속한 타입의 데이터 샘플을 멀티캐스트방식으로 분배한다[6].

DDS의 데이터 발간/구독 세부 과정은 그림 1과 같다. 하나의 도메인에는 다수의 Publisher와 Subscriber가 존재하며 미리 약속된 데이터 타입인 Topic의 샘플을 교환하게 된다.

DDS를 구성하는 주요 모듈은 다음과 같다.

- Domain: Publisher와 Subscriber의 모임

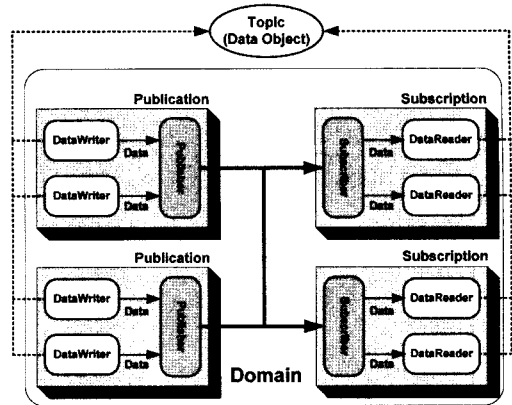


그림 1 DDS의 데이터 발간/구독 과정

- Topic: 도메인 내에서 교환하는 데이터 타입
- DataWriter: 응용으로부터 송신할 데이터를 받아서 Topic 샘플을 생성하고 도메인으로 데이터를 전송
- DataReader: 도메인으로부터 수신된 Topic 샘플을 응용프로그램이 사용 가능한 메시지로 변환하여 응용에게 전달

4. ReTiCoM(Real-Time Communication Middleware)

항공 관제 시스템이나 함정 전투체계(combat system)과 같이 다양한 장비로 구성되는 분산시스템에서는 여러 제작사에서 제공되는 각종 센서와 무장들을 다수의 다기능 콘솔들에 연동하여 운영해야 하며 통신 네트워크에 대한 의존도가 높기 때문에 통신 노드들의 환경 차이에 따른 상호 운용성 문제, 신뢰성 문제 등을 해결하기 위한 표준 통신 미들웨어를 사용할 필요가 있다.

예를 들어, 함정 전투체계에서의 정보처리하는 센서로부터 수집한 데이터에 의하거나 또는 운용자의 명령에 의해 촉발되기 때문에 데이터 중심의 통신 특성이 강하고, 여러 노드들에게 동시에 전파되는 경우가 많다. 이러한 함정 전투체계에 적합한 미들웨어 규격으로는 OMG의 DDS가 있는데 응용프로그램/하부 통신 네트워킹 기능과의 인터페이스 및 미들웨어 기능 요구사항을 정의하고 있다. 현재 DDS를 구현한 상용 제품으로는 미국 RTI(Real-Time Innovations)사의 NDDS(Network Data Delivery Service)와 네덜란드 Thales사의 Splice-DDS 등 소수에 그치고 있다. 이 논문에서는 DDS 표준 규격을 준수하도록 본 연구팀이 설계한 ReTiCoM 시스템의 내부 기능 구조에 대해 소개 한다(그림 2).

4.1 ReTiCoM 구조

ReTiCoM의 주요 모듈은 다음과 같다.

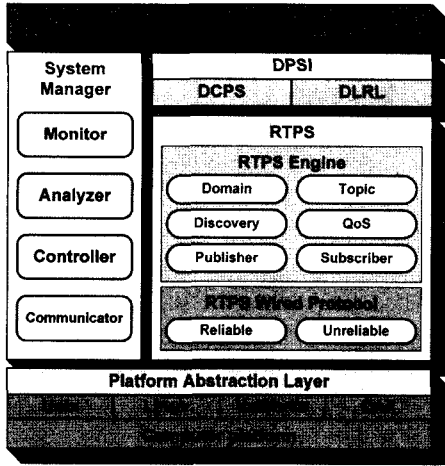


그림 2 ReTiCoM 구조

① Platform Abstraction Layer

시스템 하부의 운영체제를 추상화하는 계층으로서 ReTiCoM이 탑재되는 플랫폼의 운영체제가 바뀌더라도 ReTiCoM의 수정 없이 운영이 가능하게 한다[5].

② RTPS(Real-Time Publish-Subscribe)

실제 데이터가 발간/구독되는 계층으로서 RTPS Engine과 RTPS Wired Protocol로 구분된다. RTPS Engine은 도메인 형성과 사용할 데이터의 타입을 결정하고 도메인에 참여하는 원격의 ReTiCoM들을 동적으로 발견한다. 또한 Publisher와 Subscriber에 의해 데이터가 발간/구독된다. RTPS Wired Protocol은 실제 데이터를 통신하는 프로토콜로서 RTPS Engine이 설정한 QoS에 따라 신뢰성(reliable) 혹은 비신뢰성(unreliable)을 갖는 데이터 통신을 수행한다.

③ DPSI(Data Publish-Subscribe Interface)

응용프로그램에 제공되는 데이터 발간/구독 기능 인터페이스로서 DCPS(Data-Centric Publish-Subscribe)와 DLRL(Data Local Reconstruction Layer)로 구분된다. 이 계층을 통해 응용프로그램은 데이터를 교환할 상태에 대한 인지 없이 원하는 데이터의 발간/구독만을 수행하게 된다. 기본 인터페이스인 DCPS는 read()/write() 방식의 API를 제공하여 읽기/쓰기 방식으로 응용프로그램간의 데이터 교환 기능을 제공한다. DLRL은 선택사항으로 주고받는 데이터를 로컬 오브젝트로 변환하여 객체 참조 방식으로 데이터의 발간/구독이 이루어지게 한다.

④ System Manager

앞서 설명한 모듈들은 DDS 규격을 준수하며 이 모듈들만으로도 데이터 분배 서비스가 가능하다. 하지만 응용프로그램의 배치나 장애가 발생한 노드의 복구 등의

관리는 시스템 관리자에 의해 직접 수행되어야 한다. 따라서 사람의 개입 없이도 원활한 서비스 수행을 지원하기는 System Manager를 설계하였다[6].

Monitor는 시스템의 상태정보를 수집한다. 수집된 정보는 Analyzer에서 분석되며 시스템이 취해야 할 행동이 결정된다. Controller는 Analyzer의 판단에 따라 응용프로그램의 설치/실행/종료/삭제/갱신 등을 수행한다. Communicator는 자신이 속한 도메인내의 다른 Communicator들과 별도의 통신 채널을 설정하여 연동된다. 이는 도메인내의 한 DomainParticipant가 오작동할 때 해당 Communicator를 통해 직접 알리기 위함이다. 또한 응용프로그램 서버로부터 응용프로그램을 다운로드 하는 기능도 Communicator가 수행하다.

4.2 ReTiCoM 내부 설계

ReTiCoM의 내부 설계는 그림 3과 같다.

DomainParticipant는 도메인에 참여하는 ReTiCoM을 의미한다. Publisher와 Subscriber는 자신이 사용할 Topic 타입의 개수만큼의 DataWriter와 DataReader를 생성하여 각 DataWriter와 DataReader가 한 종류의 Topic만을 담당하도록 한다. 이는 Publisher와 Subscriber가 여러 타입의 Topic 데이터를 처리할 때 발생하는 오버헤드를 줄이기 위함이다. 각 Publisher와 Subscriber는 전달받은 Topic 데이터의 담당 DataWriter와 DataReader를 확인하고 담당자에게 전달하는 역할을 한다.

Listener와 WaitSet, Condition은 응용프로그램의 데이터 수신을 위해 사용된다. 응용프로그램은 ReTiCoM이 제공하는 각종 Listener 인터페이스를 구현하여 Subscriber에게 등록하고 데이터가 구독되는 순간 등록된 Listener가 실행되어 응용프로그램에게 즉시 데이터 전달이 가능하다. WaitSet과 Condition은 응용프로그램이 자신의 작업을 수행하다가 필요에 의해 응용프로그램

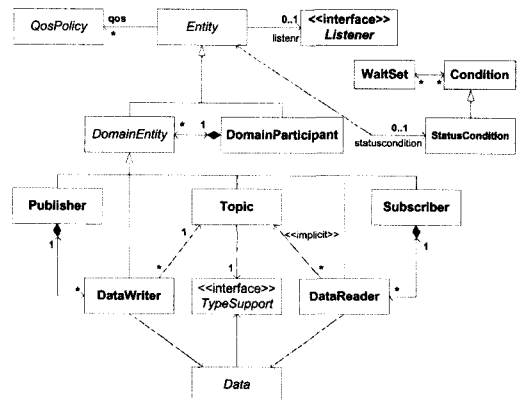


그림 3 ReTiCoM 데이터 전달 기능 내부 설계

래이 직접 Subscriber로부터 데이터를 읽어오는 능동적인 방식이다.

4.3 ReTiCoM QoS

DDS에서는 표 1과 같은 QoS정책을 지원하고 있다. ReTiCoM에서도 DDS의 QoS를 채택하여 효율적인 데이터 분배 서비스 제공한다[7].

QoS 정책을 통한 효율성 향상 기법의 예는 그림 4와 같다.

특정 Publisher가 유사한 값을 너무 자주 발간하는 경우 불필요한 데이터의 분배로 인한 대역폭 손실이 발생하게 된다(그림 4-A). 이러한 경우 TIME_BASED_FILTER QoS의 minimum separation을 설정하면 데이터 발간시점에서 설정된 minimum separation 기간 동안 응용이 write하는 데이터가 드롭(drop)되게 된다(그림 4-B). 반대로 Publisher가 너무 오래 데이터 발간을 하지 않는 경우 해당 Publisher의 고장여부를 확인해야 한다. 이러한 경우 DEADLINE QoS를 이용하면 Publisher는 deadline안에 발간할 데이터가 없을 경우에도 deadline 만료 전에 heart bit을 Subscriber에게 전달하여 신뢰성을 높일 수 있다(그림 4-C).

표 1 ReTiCoM QoS Policy

| QoS Policy | QoS Policy |
|--------------------|-----------------------|
| USER_DATA | TIME_BASED_FILTER |
| TOPIC_DATA | PARTITION |
| GROUP_DATA | RELIABILITY |
| DURABILITY | TRANSPORT_PRIORITY |
| DURABILITY_SERVICE | LIFESPAN |
| PRESENTATION | DESTINATION_ORDER |
| DEADLINE | HISTORY |
| LATENCY_BUDGET | RESOURCE_LIMITS |
| OWNERSHIP | ENTITY_FACTORY |
| OWNERSHIP_STRENGTH | WRITER_DATA_LIFECYCLE |
| LIVELINESS | READER_DATA_LIFECYCLE |

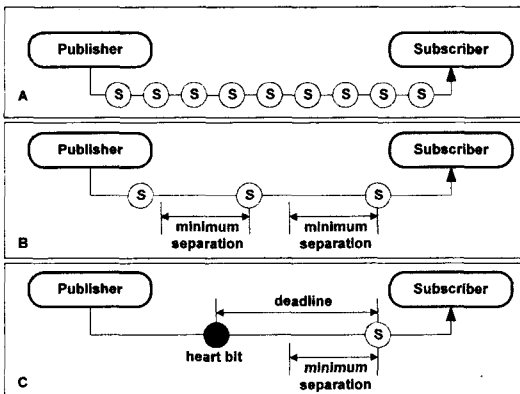


그림 4 QoS를 이용한 효율성 향상

5. 성능 평가

동적으로 네트워크 도메인이 형성되고 도메인에 참여하는 노드들의 빈번한 등록 및 해제가 발생하며 데이터 중심의 발간/구독이 일어나는 환경에서 DDS와 기존의 서버/클라이언트 기반 미들웨어와의 성능차이를 평가하였다. 현재 ReTiCoM은 개발단계인 관계로 OMG의 DDS 규격을 만족하는 상용 제품인 NDDS 4.0을 데이터 중심 발간/구독 통신 기법의 성능 평가 대상으로 사용하였다[9]. 비교 대상으로는 SUN의 JMS 규격을 만족하는 OpenJMS 0.7.7을 사용하였다. JMS는 응용단에서 Topic을 발간/구독한다는 점은 DDS의 발간/구독 방식과 같다. 그러나 디스커버리 기능이 부족하기 때문에 도메인에 참여하는 모든 Publisher와 Subscriber가 메시지 서버에 등록하는 서버/클라이언트 방식이다. JMS Publisher들로부터 발간된 Topic들이 메시지 서버의 큐(queue)에 수집되고 메시지 서버가 수집된 Topic을 등록된 Subscriber들에게 전달한다.

5.1 스트레스 실험

1:1 통신 환경에서 데이터를 지속적으로 발간할 때 데이터 구독에 소요되는 시간을 측정하였다. 극한 상황을 연출하기 위해서 128 바이트 크기의 캐릭터 배열 메시지를 100개부터 50,000개까지 증가시키면서 반복문을 이용하여 연이어 발간하였을 때 Subscriber가 그 메시지를 모두 구독하는데 까지 소요되는 시간을 ms단위로 측정하였다(표 2).

100개부터 1,000개의 메시지를 연이어 전송하는 실험까지는 DDS와 JMS 모두 유사한 성능을 보이지만 5,000개의 메시지를 전송하는 실험에서부터 JMS의 성능저하가 발생하였다(그림 5).

표 2 메시지 수 증가에 따른 전송 소요시간 측정 (ms)

| | 100 | 500 | 1,000 | 5,000 | 10,000 | 50,000 |
|-----|-------|-------|-------|---------|---------|----------|
| DDS | 119.5 | 202.6 | 375.5 | 1912.4 | 4000.8 | 19864.7 |
| JMS | 692 | 2837 | 5403 | 25668.8 | 52109.6 | 259608.4 |

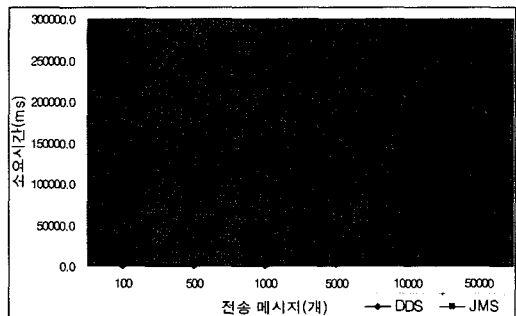


그림 5 메시지 수 증가에 따른 전송 소요 시간

5.2 확장성 실험

도메인에 참가하는 Publisher와 Subscriber의 수가 증가함에 따라 발생하는 전송 소요 시간의 변화를 측정하였다. n 개의 Publisher가 있을 때 각각이 1,000ms마다 128 바이트 크기의 캐릭터 배열 메시지를 100번 n 개의 Subscriber들에게 발간할 때(n : n 통신), 한 Subscriber가 이 100n 개 메시지를 모두 구독하는데 소요되는 시간을 측정하였다(표 3).

실험 결과 Publisher와 Subscriber가 7:7 통신을 할 경우까지는 DDS와 JMS가 유사한 성능을 보였으나, 8:8 통신부터는 DDS의 성능이 월등하게 우수함을 알 수 있었다(그림 6).

더욱이 JMS의 경우 20:20 통신부터는 기능이 중지되어 결과값을 측정할 수 없었다. 이에 반해 DDS의 경우 20:20 통신까지 평균 111,964ms의 소유시간 보였으며, 이는 Publisher가 1,000ms마다 메시지를 100번 발간하는데 소요되는 시간 100,000ms과 비교하면 평균 10% 안팎의 고른 지연시간임을 알 수 있다. DDS가 예측 가능한 지연시간을 갖는다는 특징은 실시간성 보장에 핵심적인 요소이다. 25:25 통신은 NDDS에서 제공하는 도메인당 최대 DomainParticipant 개수(50개)에 해당하며 지연시간은 증가되었지만 메시지 손실 없이 정상 동작하였으며 이러한 특징은 신뢰성 보장에 유리하게 작용한다.

앞서 기술한 두 실험에서 미들웨어가 처리해야 하는 메시지 통신량이 적을 때 JMS 성능이 다소 좋은 이유는, JMS 메시지 서버에 모든 클라이언트의 위치정보가 사전에 등록되기 때문이다. DDS는 별도의 중앙 서버가 없으므로 도메인 내의 모든 Publisher들은 각자 Sub-

scriber들을 디스커버리(discovery)해야 하는 초반 부담이 존재한다. 하지만 통신량이 늘어나면서 메시지 서버의 병목현상으로 급격한 성능저하를 보이는 JMS에 반해 DDS는 신뢰성과 실시간성을 보장할 수 있다.

6. 결론

다수의 디바이스들이 동적으로 도메인을 형성하고 동일한 타입의 데이터를 빈번하게 교환할 경우에는 데이터 중심 발간/구독 방식의 통신 기법이 유리하며 이러한 환경에 적합한 미들웨어 규격으로는 OMG의 DDS가 있다. 본 논문에서 제안하는 데이터 분배 서비스 시스템(ReTiCoM: Real-Time Communication Middleware)은 데이터 중심 발간/구독 방식의 통신 미들웨어로서 데이터 교환을 위한 응용프로그램의 부담을 줄일 수 있으며 기본적으로 DDS 표준 규격을 준수하여 이미 개발된 DDS 응용프로그램을 재사용할 수 있는 장점이 있다. 또 System Manager를 통하여 DDS만으로는 어려운 시스템 관리 자동화가 가능하다.

또 NDDS와 OpenJMS를 이용하여 다수의 디바이스들이 동적으로 도메인을 형성하고 빈번하게 데이터를 교환하는 실제 환경에서 데이터 중심 발간/구독 방식이 기존의 서버/클라이언트 방식에 비해 효율적임을 검증하였다.

참고 문헌

- [1] OMG, "The Common Object Request Broker : Architecture and Specification, Revision 2.0," OMG TC Document, 1994.
- [2] Kupsys A, Ekwall R, "Architectural Issues of JMS Compliant Group Communication," Network Computing and Applications, Fourth IEEE International Symposium, pp. 139-148, July. 2005.
- [3] Pardo-Castellote, "OMG Data-Distribution Service: Architectural Overview," ICDCSW'03, 2005.
- [4] Gerardo Pardo-Castellote, Bert Farabaugh, Rick Warren, "An Introduction to DDS and Data-Centric Communications," 2005 Real-Time Innovations, August. 2005.
- [5] 김명선, 유인선, 최 훈, "실시간을 지원하는 리눅스 인터페이스용 미들웨어 설계 및 구현", 한국정보과학회 KCC2006 추계학술발표대회, 논문집 제33권 제2호(A), pp. 313-317, 2006. 10.
- [6] 박충범, 유용덕, 최 훈, "개인 자율 컴퓨팅 기법", 한국차세대PC 논문지 제2권 제3호, pp. 50-57, 2006.9.
- [7] OMG, "Data Distribution Service Specification version 1.2," OMG, OMG Technical Document, Jan. 2007.
- [8] 원영암, 김명선, 최 훈, "DDS 기반 WTIS 설계 및 구현", 한국통신학회 2006 추계학술대회, 논문 초록집 제 32호, pp.206, 2006. 11.
- [9] Network Data Distribution Service(NDDS). [Online]. Available: <http://www.rti.com/docs/NDDS.pdf>

표 3 Publisher와 Subscriber 증가에 따른 성능 측정(ms)

| | 1:1 | 2:2 | 3:3 | 4:4 | 5:5 | 6:6 | 7:7 |
|-----|---------|---------|---------|---------|---------|---------|---------|
| DDS | 110,002 | 110,016 | 109,933 | 109,858 | 110,158 | 109,963 | 110,816 |
| JMS | 108,573 | 109,149 | 110,143 | 109,845 | 110,591 | 111,074 | 113,268 |
| | 8:8 | 9:9 | 10:10 | 15:15 | 20:20 | 25:25 | |
| DDS | 110,820 | 111,076 | 111,809 | 113,960 | 125,141 | 212,523 | |
| JMS | 152,698 | 193,770 | 236,357 | 430,127 | - | - | |

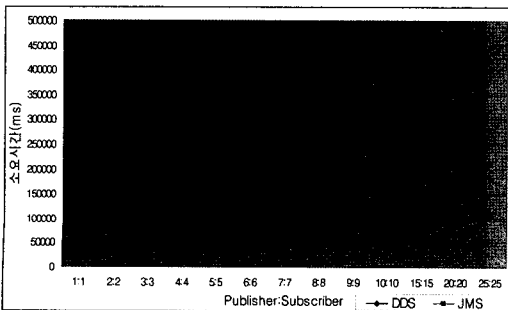


그림 6 Publisher와 Subscriber 증가에 따른 성능 비교