DDS 미들웨어의 디스커버리 동작 시험을 위한 응용의 설계 및 구현

안성우

경남정보대학교

Design and Implementation of the Application for Testing Discovery Operation of the DDS Middleware

Sungwoo Ahn

Kyungnam College of Information & Technology

E-mail : ahnsw@kit.ac.kr

요 약

OMG 그룹에서 제안하고 있는 DDS(Data Distribution Service) 미들웨어에서 디스커버리 프로토콜은 미들웨어 간의 상호 운용성 지원을 위한 기본 요소이다. DDS 미들웨어가 적용되는 대부분의 환경에서는 다수의 노드에서 생성된 참여자를 찾기 위해 한꺼번에 많은 수의 디스커버리 정보가 교환되기 때문에 노드 및 네트워크의 성능에 많은 영향을 미친다. 본 논문에서는 DDS 미들웨어 개발 후에 시험이 되어야 하는 디스커버리 기능의 검증과 성능 측정을 위한 응용을 설계하고 이를 구현하였다. 디스커버리 시험 응용은 디스커버리 시험 전체를 관리하는 제어 응용, DDS 미들웨어를 통하여 DDS 개체를 생성하고 이들간의 정보교환을 담당하는 DDS 응용, 그리고 DDS 응용을 통제하고 실행 결과를 수집하여 제어 응용으로 전송하는 데몬 서버로 구성된다. 구현된 디스커버리 시험 응용은 디스커버리 기능 검증 및 성능 정보의 수집을 프로세서 간의 통신을 통하여 효과적으로 수행한다. 또한, DDS 응용의 동작 권한을 제어 응용이 위치한 노드에 집중시킴으로써 많은 수의 노드에서 시험이 필요한 환경에 적합하도록 설계되었다.

키워드

DDS, Discovery, Test Program, Real-time Middleware

Ⅰ. 서 론

DDS 미들웨어에서는 시스템 간 상호 운용성을 제공하기 위해 표준 프로토콜을 제시하고 있다. DDS 표준은 크게 DCPS (Data-Centric Publish Subscribe) [1]와 RTPS (Real-Time Publish-Subscribe) [2]의 두 개의 계층 구조로 정의되어 있다. DCPS는 데이터 모델 정의를 통해 DDS 응용으로 표준 인터페이스와 데이터 교환 최적화를 위한 QoS 설정 기능을 제공한다. RTPS는 네트워크를 통한 DDS 데이터 패킷 전송 포맷에 대한 규격을 정의하고 있다.

DDS 미들웨어를 사용해서 통신을 하는 시스템에서 가장 중요한 요소는 통신에 참여하는 개체(Entity)들의 정보를 자동으로 교환하여 서로의 존재를 인식하는 디스커버리 과정이다. 상호 운용성을 위해 RTPS 표준명세에서는 2단계로 이루어진 단순 디스커버리(Simple Discovery) 프로토콜을 정의하고 있다. 이 과정에서 서로 관심있는 정보의 종류와 QoS를 확인하고 통신 관계를 맺을지 판단을 하게 된다. 디스커버리 과정의 특징으로 인하여 DDS 통신 네트워크에 참여하는 개체가 증가할수록 개체 간에 교환되는 정보 역시 기하급수적으로 증가하게 된다. 따라서 고성능의 DDS 미들웨어를 개발하기 위해서는 DDS 통신 참여 개체의 변화에 따라 디스커버리 과정에서 발생하는 리소스 부하를 측정하여 분석하는 것이 필요하다.

본 논문에서는 DDS 미들웨어의 디스커버리 기능 및 성능을 시험할 수 있는 디스커버리 시험 응용을 설계하고 이를 구현하였다. 구현된 디스커

버리 시험 응용은 중앙 서버에서 디스커버리 전 과정을 제어하고 정보를 수집하는 방식을 사용함으로써 많은 수의 노드가 DDS 미들웨어 통신에 참여하는 환경에서 손쉽게 정보 수집이 가능하다. 측정된 정보는 DDS 미들웨어의 기능 검증과 동시에 성능저하 요소 파악을 통하여 개발 중인 DDS 미들웨어의 튜닝에 활용될 수 있다.

논문의 구성은 다음과 같다. 2장에서는 DDS 미들웨어의 구성요소와 디스커버리 과정에 대해서 살펴본다. 3장에서는 디스커버리 시험 응용의 설계 및 구현에 대해서 설명한다. 마지막으로 4장에서 결론을 기술한다.

II. DDS 미들웨어

최근 급속히 발전하는 네트워크 기술과 함께 네트워크 참여자 간에 교환되는 정보의 양은 엄청나게 증가하고 있다. 이에 따라 네트워크에 연결된 시스템이 입출력하는 정보를 원활하게 처리해야 할 뿐만 아니라 교환된 정보를 실시간으로 사용하고자 하는 사용자의 요구가 증가하고 있다. OMG(Object Management Group)에서는 분산 응용 간의 데이터 중심 통신을 지원하기 위해 DDS(Data Distribution Service) 미들웨어에 대한 표준을 제공하고 있다. DDS 미들웨어는 중앙 서버없이 단순한 발간/구독 방식의 통신을 사용함으로써 분산 환경에서 실시간 데이터 배포를 효과적으로 할 수

있는 특징을 가지고 있다 [3,4].

DDS 네트워크를 구성하는 요소는 그림 1과 같다. DDS 네트워크는 데이터의 공유 영역인 도메인(Domain)으로 구분되며 도메인 내에는 하나 이상의 참여자(Participant)를 포함하고 있다. 각 참여자는 하나 이상의 발간자(Publisher) 또는 구독자(Subscriber)를 가질 수 있으며 발간자는 자신의 정보를 전달하기 위해서 하나 이상의 발간개체(Writer)를, 구독자는 상대편 발간자의 정보를 수신하기 위해 하나 이상의 구독개체(Reader)를 포함할 수 있다 [1].

DDS 응용에서는 특정 데이터의 주제를 나타내는 토픽(Topic)을 생성하고 이를 발간개체와 구독개체에 관심 토픽으로 등록함으로써 이들 간에 정보를 교환하게 된다. 관심 토픽이 다른 발간자와 구독자는 토픽 정보를 교환하지 않기 때문에 이들 간에는 간섭이 발생하지 않는다. 관심 토픽에 대응되는 정보는 발간개체를 통하여 데이터 분배 서비스 네트워크를 통해서 "DDS 데이터" 패킷으로 전송되며 구독개체는 수신된 "DDS 데이터" 패킷에 포함된 토픽 정보를 추출하여 응용에게 전달한다.

발간개체와 구독개체를 통한 DDS 미들웨어 간의 데이터 교환은 두 단계로 나누어진다 [2,5].

첫 번째 단계에서는 디스커버리 과정을 통하여 단말개체 정보 교환 및 단말개체 간 연결 설정을 한다. DDS 미들웨어의 단말개체가 정보를 교환하기 위해서는 서로 관심을 가지는 토픽과 토픽 전달방법을 알고 있어야 한다. 이를 위해서 표준 명세에서는 DDS 미들웨어에서 제공해야 할 공통 디스커버리 프로토콜을 정의하고 있다. DDS 미들웨어 디스커버리는 먼저 SPDP (Simple Participant Discovery Protocol)를 통하여 네트워크에 존재하는 참여자의 정보를 교환한다. 그 후 SEDP (Simple Endpoint Discovery Protocol) 과정을 통하여 참여자에 속한 단말개체의 정보를 교환한다. SPDP와 SEDP 이후 발간개체와 구독개체는 관심 토픽이 동일한 단말개체의 연결정보를 저장한다.

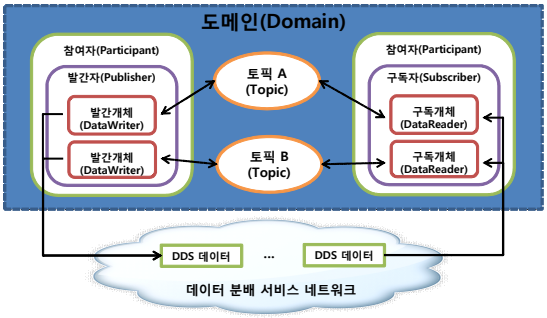


그림 1. DDS 네트워크 구성 요소

DDS 표준에서는 SPDP, SEDP를 기반으로 하는 단순 디스커버리의 구현을 필수요소로 정의하고 있으며 성능향상을 위해 추가적인 디스커버리 프로토콜의 제공을 허용하고 있다. 어떤 방식의 프로토콜을 사용하던 간에 디스커버리 단계에서는 참여자와 단말개체의 변경정보를 실시간으로 탐지하기 위해 많은 "DDS 데이터" 패킷이 교환된다. 이로 인해 네트워크와 노드의 리소스에 많은 부하가 발생하게 된다. 고성능의 DDS 미들웨어를 개발하여 적용하기 위해서는 다양한 시험 방법 및 과정이 요구되지만 디스커버리 단계에서 노드와 DDS 개체의 변경에 따른 시스템 리소스의 부하를 탐지하고 분석하는 것이 우선적으로 필요하다.

III. 디스커버리 시험 응용의 설계 및 구현

본 논문에서는 실제 환경에 적용 전 또는 개발을 진행 중인 DDS 미들웨어의 디스커버리 정확성을 검증하고 성능을 확인하기 위한 디스커버리 시험 응용을 설계하고 구현하였다. 그림 2는 디스커버리 시험 응용의 구성 요소 및 동작 방식을 보여주고 있다.

DDS 미들웨어를 사용하는 응용은 하나의 컴퓨터 노드에 다수가 실행될 수 있으며 서로 다른 n

개의 컴퓨터 노드에 떨어져서 실행될 수도 있다. 따라서 동일 노드가 아닌 다수의 노드에서 동작하는 DDS 미들웨어를 편리하게 제어하기 위해서는 DDS 응용과 직접 통신이 가능한 서버 형태의 응용이 적합하다. 그림 2에서 DDSApp는 DDS 미들웨어를 사용하는 메인 프로그램으로 미들웨어 내에 참여자, 토픽, 발간개체, 구독개체 등을 생성하고 디스커버리 프로토콜을 실행시킨다. Daemon은 각 노드에 하나씩 생성되는 TCP 데몬 서버로 중앙 제어 응용인 Command로부터 메시지를 받아서 DDSApp을 실행시키고 DDSApp으로부터 디스커버리 결과 및 성능정보를 받아서 Command로 전달하는 역할을 한다. Command는 제어 응용으로 오직 한 노드에서만 실행되며 Daemon으로 메시지를 보내서 DDSApp의 동작과 이에 대한 정보 수집을 담당하는 TCP 클라이언트이다.

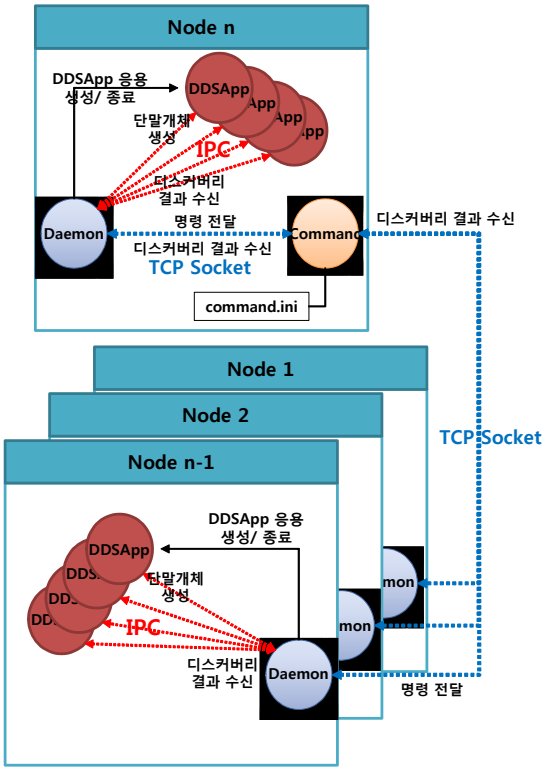


그림 2. 디스커버리 시험 응용의 구성

각 응용 간의 통신은 프로세스 간의 통신 방식을 사용한다. Daemon은 동일 노드에 존재하는 DDSApp과 통신을 하면 되기 때문에 IPC Pipe를 이용하여 정보를 교환한다. Command와 Daemon은 물리적으로 떨어진 노드간의 통신을 포함하기 때문에 TCP 소켓을 사용하여 정보를 교환한다.

Command는 Daemon에게 전달할 명령의 설정정보를 파일로 저장하고 읽어 들이도록 하여 실험 환경변화에 따라 쉽게 정보를 편집할 수 있도록 하였다. 그림 3은 설정정보 파일에 저장되는 정보의 예를 보여주고 있다. 설정 정보는 노드의 개수, 도메인의 개수, 디스커버리 후 토픽 교환 여부, 노드별 실행 DDSApp의 개수, 노드의 IP, DDSApp별 생성할 발간개체, 구독개체의 개수 등을 지정한다.

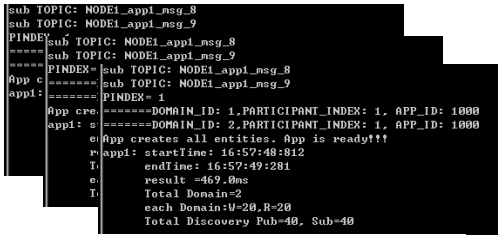
```
// *** GLOBAL configuration ***
// 1. Total number of nodes
// 2. Total number of domain
// 3. Flag for exchanging topic btw. endpoints
// *** NODE# configuration ***
// 1. Number of DDSApps
// 2. IP Address of Node
// 3. Number of Publisher(Writer)
// 4. Number of Subscriber(Reader)

////////////// GLOBAL ////////////////
2
1
1

////////////// NODE1 ////////////////
6
192.168.0.2
10
5

////////////// NODE2 ////////////////
10
192.168.0.3
20
30

// ...
```

그림 3. Command 설정파일 포함 정보

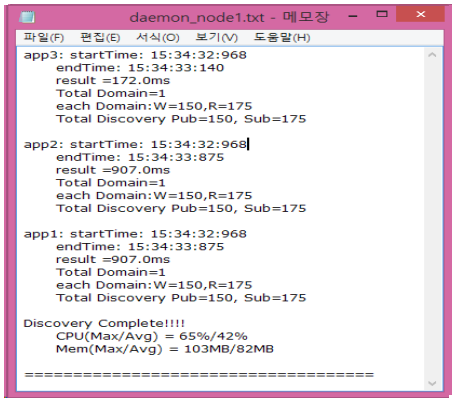
그림 4는 Command로부터 명령을 받아 디스커버리를 실행시킨 결과화면을 보여주고 있다. Command로부터 그림 3에서 설정된 개체 생성 명령을 받으면 Daemon은 대상 노드에서 설정정보에 따라 DDSApp을 생성하고 각 DDSApp에 발간개체와 구독개체를 만든다. 모든 노드에서 DDSApp 생성이 완료된 후 Command로부터 디스커버리 명령이 전달되면 Daemon을 통하여 모든 노드에 있는 DDSApp들 간에 디스커버리 프로토콜이 동작하여 단말개체 간의 정보를 교환한다.

디스커버리 실행이 완료된 후 각 DDSApp의 화면에는 그림 4와 같이 디스커버리 완료여부와 디스커버리 수행시간 등의 정보를 표시함과 동시에 Daemon에 자신의 정보를 보낸다. Daemon은 자신의 노드에서 실행중인 모든 DDSApp의 결과를 수신하면 daemon_node#.txt를 생성하여 Command에게 전달한다. Command가 모든 노드에 있는 Daemon으로부터 정보 수신을 완료하면 디스커버리 과정이 종료된다. Daemon으로부터 전달되는 daemon_node#.txt에는 각 DDSApp별 디스커버리 시간, 디스커버리 성공여부, CPU 사

용률, 메모리 사용률 정보가 포함되어 있다.



DDSApp들의 디스커버리 실행결과



Daemon에 의해 수집된 DDSApp의 결과

그림 4. 디스커버리 수행 후 Damon이 전달한 로그 정보

그림 5는 각 DDSApp에서 생성한 상세 디스커버리 로그를 보여주고 있다. 로그에는 각 시간대별로 단말개체의 생성 정보, 다른 DDSApp에서 보내온 디스커버리 정보, 그리고 디스커버리 정보 수신 후 관심 토픽이 같은 단말개체의 연결 정보 등을 포함하고 있다. 디스커버리가 완료된 후 이 정보는 모두 Command가 있는 노드로 모여 적용된 DDS 미들웨어의 검증 및 성능 분석 자료로 활용된다. 그림 4의 Daemon이 전달한 정보는 DDS 미들웨어의 성능 측정 및 분석에 활용되고 그림 5의 DDSApp이 전달한 정보는 디스커버리 프로토콜의 검증에 활용이 가능하다.

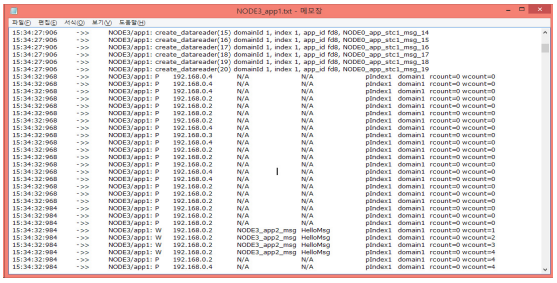


그림 5. DDSApp에서 저장한 상세 디스커버리 로그 정보

Ⅳ. 결론 및 향후연구

본 논문에서는 DDS 미들웨어의 주요 성능 측정 요소 중 하나인 디스커버리 기능의 검증과 동시에 성능을 측정하기 위한 디스커버리 시험 응용을 설계하고 구현하였다. DDS 디스커버리 프로토콜은 DDS 응용 간의 통신을 위한 가장 기본적인 요소이지만 생성된 단말개체 수가 증가하면 단말개체의 연결정보를 교환하기 위해 한꺼번에 많은 수의 디스커버리 정보가 교환되어 노드 및 네트워크의 성능에 많은 영향을 미친다. 따라서 고성능의 DDS 미들웨어를 개발하기 위해서는 다양한 환경에서 디스커버리 시험을 거쳐 성능저하 요소를 줄여야 한다. 본 논문에서 구현된 디스커버리 시험 응용은 프로세서 간의 통신을 통하여 물리적으로 분리된 노드에 존재하는 DDS 응용의 디스커버리를 효과적으로 제어하고 결과의 수집이 가능하도록 하였다. 또한, 디스커버리 전 과정을 하나의 노드에서 제어할 수 있도록 함으로써 많은 수의 노드에서 시험이 필요한 환경에 적합하도록 설계되었다.

향후 연구로는 DDS 표준에서 제공하는 다양한 QoS를 디스커버리 시험 응용의 입력 요소로 사용하여 디스커버리 단계뿐만 아니라 QoS의 변화에 따라 단말개체 간 연결이 이루어진 후의 성능 변화를 측정할 수 있도록 하는 것이 필요하다.

참고문헌

[1] OMG, "Data Distribution Service for Real-time Systems," Version 1.2, OMG (2007).

[2] OMG, "The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification," Version 2.1, OMG (2008).

[3] 전형국, 이수형, 김원태, 김경태, 박승민, “DDS 미들웨어 표준 기술 동향,” 정보통신산업진흥원 주간기술동향 1456호, pp.1-13, 2010.

[4] D.C. Schmidt, A. Corsaro, and H. Hag, “Addressing the Changenges of Tactical Information Management in Net-Centric Systems with DDS,” The Journal of Defense Software Engineering, pp.24-29, 2008.

[5] 안성우, 최종우, 최윤석, “DDS 미들웨어의 상호운용성 제공을 위한 표준 디스커버리 프로토콜,” 한국해양정보통신학회 춘계종합학술대회, pp.205-208, 2011.