한국군사과학기술학회지 제14권 제6호, pp. 1081~1090, 2011년 12월

학술논문 정보기술 부문

MOM 서비스에 의한 HLA 페더레이션 연동 성능 분석

Performance Analysis of Interoperated HLA Federations for MOM Service

유 민 욱* 김 탁 곤*
Min Wook Yoo Tag Gon Kim

Abstract

High Level Architecture(HLA) is a specification for interoperation among heterogeneous simulators which are executed in a distributed environment. HLA originally allows a number of federates to join in a federation using a single RTI(Run-Time Infrastructure). To interoperate federations without modifying RTI, agent federate, which represents behavior of a federation, can be used. Agent federate can use MOM(Management Object Model) service or agent-user protocol to acquire information of a federation. This paper performs various experiments to measure performance of two architectures. MOM service shows a loss of performance but can be applied without modifying user federates. Experiment results can be used to determine appropriate architecture for interoperation of federations.

Keywords : HLA, RTI, Interoperation of Federations(페더레이션 연동), MOM Service(MOM 서비스), Performance Anaylsis(성능 분석)

1. 서 론

HLA(High Level Architecture)는 다양한 환경에서 개발되는 시뮬레이터들의 연동을 위해 제안된 표준으로 2000년에 IEEE1516 표준으로 선정되었다[1]. RTI(Run-Time Infrastructure)는 HLA 표준 서비스를 제공하는 소프트웨어이다. HLA는 분산된 환경에서 수행되는 개별 시뮬레이션을 페더레이트로 정의하고 페더레이트들의 집합을 페더레이션으로 정의한다. HLA/RTI 환경에서 연동 시뮬레이션의 수행을 위해서 페더레이트들로 페더레이션을 구성하고 FOM(Federation Object Model)에 페더레이트들 간에 주고받을 객체 정보를 기술한다. 페더레이션 내에서 페더레이트들은 RTI가 제공하는 HLA 서비스를 이용하여 연동 시뮬레이션을 수행한다.

HLA는 하나의 페더레이션에서의 시뮬레이션만을 정의하기 때문에 페더레이션 연동을 지원하지 않는다. 하지만 두 페더레이션의 통합 시뮬레이션이 필요한 경우가 있다. 예를 들어 서로 다른 국가에서 개발된 워게임 시뮬레이션을 통합해야 수행하는 경우를 생각해 볼 수 있다. 정보 보안으로 인해 서로 다른 페더레이션의 페더레이트들간에 주고받을 수 있는 정보는 제한된다. 내부 결정 과정에서 주고받는 정보는 다른 페더레이션의 페더레이트가 접근할 수 없어야 하고 위치 정보와 같이 공통으로 필요한 정보는 다른 페더레이션의 페더레이트로 전달해야 한다. 하지만 한 페더레이션으로 통합하여 시뮬레이션을 하게 되면 각 페더레이

† 2011년 9월 15일 접수~2011년 11월 25일 게재승인
* 한국과학기술원(KAIST)
책임저자 : 유민욱(mwyu@smslab.kaist.ac.kr)

트들은 FOM에 기술되어 있는 모든 정보에 접근할 수 있기 때문에 정보 제한을 할 수 없어 보안 문제가 발생한다.

HLA는 페더레이션 연동을 정의하지 않기 때문에 현재 개발되어 있는 RTI는 이와 관련된 서비스를 지원하지 않는다. RTI의 수정 없이 페더레이션 연동을 지원하도록 하는 방법으로는 페더레이션의 대리자 페더레이트를 사용하는 방법이 있다[2]. 각 페더레이션에 다른 페더레이션의 동작을 대행해주는 대리자 페더레이트를 참여시켜 연동을 가능케 한다. 페더레이션의 동작을 대행하기 위해서는 대리자 페더레이트는 참여한 페더레이션의 정보를 얻어야 한다. RTI는 대리자 페더레이트도 일반 페더레이트로 똑같이 취급하기 때문에 페더레이션의 정보를 얻기 위하여 대리자 페더레이트는 RTI가 제공하는 조회 서비스와 콜백 서비스만을 이용할 수 있다. 하지만 몇몇 서비스에 대해서는 필요한 모든 정보를 얻을 수가 없기 때문에 연동에 문제가 발생하게 된다[3]. 이를 해결하기 위해서 크게 두 가지 방법을 사용할 수 있다. 먼저, 사용자 페더레이트와 대리자 사이에 정보 교환 프로토콜을 정의하여 직접 정보를 주고 받는 방법이 있다[4]. 또한, MOM (Management Object Model) 서비스를 이용하여 대리자 페더레이트가 RTI로부터 필요 정보를 받는 방법이 있다[3].

HLA의 MOM 서비스는 페더레이트가 페더레이션과 다른 페더레이트의 정보를 제공한다. MOM 서비스를 이용하게 되면 대리자 페더레이트는 RTI로부터 다른 페더레이트의 정보를 확인할 수 있기 때문에 사용자 페더레이트는 연동을 위한 추가적인 기능을 구현할 필요가 없다. 하지만 RTI는 MOM 서비스를 사용하게 되면 전체 서비스의 진행이 느려지기 때문에 연동 시뮬레이션 성능에 영향을 준다. 사용자 페더레이트와 대리자의 정보 교환 프로토콜을 정의하는 경우에는 MOM 서비스를 사용하지 않기 때문에 그에 따른 성능 저하는 피할 수 있다. 하지만 사용자 페더레이트들은 일부 서비스를 사용할 때, 필요 정보를 대리자 페더레이트로 보내주는 작업이 필요하다[5].

본 논문에서는 실험을 통해 MOM 서비스 사용에 의한 페더레이션 연동 성능을 측정해본다. 페더레이션 연동에 MOM 서비스를 이용하는 구조와 대리자-사용자 프로토콜을 이용하는 구조의 시뮬레이션 성능을 비교하여 MOM 서비스를 이용하였을 때 성능에 어느 정도 영향을 미치는지 확인해본다. 데이터 송/수신 페더레이트들을 두 페더레이션에 참여시키고 연동하여 정보 교환 성능을 측정하였다. 실험 결과를 이용하여 원하는 성능에 따라 페더레이션 연동 구조를 결정하는데 도움을 줄 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 페더레이션 연동 구조에 대해서 소개하고 MOM 서비스에 대해서 알아본다. 3장에서는 MOM 서비스를 이용하는 구조와 대리자-사용자 프로토콜을 이용 구조에서 페더레이션을 연동하여 실험을 한다. 성능 측정 페더레이트들 두 페더레이션에 분포시켜 정보 교환 실험을 하고 결과를 분석한다. 마지막으로 4장에서는 결론을 맺는다.

2. Background

가. 페더레이션 연동

Fig. 1은 대리자 페더레이트를 이용하는 페더레이션 연동 구조를 나타낸다. 각 페더레이션에 대리자 페더레이트가 하나씩 참여하고 두 대리자는 서로 정보를 주고받을 수 있도록 연결되어 있다.

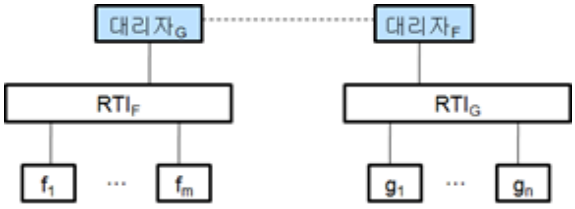

Fig. 1. 대리자를 이용한 페더레이션 연동 구조

대리자 페더레이트는 참여한 페더레이션에서 반대편 페더레이션의 동작을 대행한다. Fig. 1에서 대리자 G는 페더레이션 G의 동작을 대행하고 대리자 F는 페더레이션 F의 동작을 대행하여 참여 페더레이션에서 RTI와 소통한다.

Fig. 2는 페더레이션 연동 구조에서의 서비스 수행 과정을 나타낸다. 먼저, 대리자 페더레이트는 조회 서비스와 콜백 서비스를 이용하여 페더레이션의 정보를 얻는다. 시간 정보와 같이 RTI가 조회 서비스를 제공하는 경우에는 주기적으로 해당 정보를 조회하고 변화된 정보를 확인하게 된다. 객체 갱신과 같은 서비스는 다른 페더레이트의 동작에 따라서 RTI가 대리자 페더레이트로 콜백 서비스를 호출해준다. 대리자 페더

레이트는 획득한 정보와 내부에 저장하고 있던 정보를 이용하여 서비스 요청이 필요한 경우 반대편으로 필요 정보와 함께 보낸다. 정보를 받은 대리자 페더레이트는 참여한 페더레이션에 서비스를 요청하여 반대편 페더레이션의 동작을 대행한다.

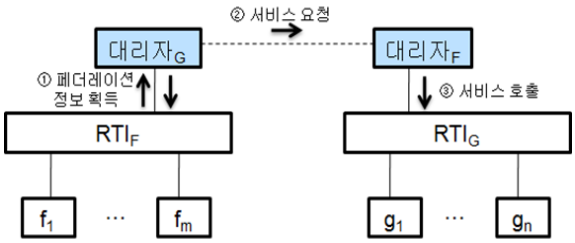

Fig. 2. 페더레이션 연동 과정

대리자 페더레이트가 RTI 서비스를 통해 충분한 정보를 얻을 수 있는 서비스의 경우에는 위 과정을 통해 정상적으로 진행할 수 있다. 하지만 동기화 서비스와 같이 모든 페더레이트가 서비스 종료를 보고해야만 진행되는 서비스에서 대리자로 인해 서비스를 진행할 수 없는 상황이 생긴다. 대리자는 다른 페더레이트들의 서비스 호출 상태를 확인해야만 해당 동작을 대행해줄 수 있는데, 해당 정보를 확인할 수 있는 서비스를 RTI는 제공하지 않는다. 이로 인해서 대리자 페더레이트는 서비스 종료를 보고할 수 없고 전체 서비스가 진행되지 못하는 상황이 발생하게 된다[3].

이러한 문제를 해결하기 위해서 대리자 페더레이트가 다른 페더레이트의 서비스 호출 정보와 같이 일반 RTI 서비스로 얻을 수 없는 정보를 얻을 수 있도록 해야 한다. RTI를 수정하지 않고 이를 해결할 수 있는 방법은 크게 두 가지가 있다. 먼저, RTI의 MOM 서비스를 사용하는 방법이 있다. MOM 서비스를 이용하게 되면 설정에 따라 RTI는 페더레이트의 상태 정보 및 서비스 호출 정보를 객체 관리 서비스를 통해 다른 페더레이트로 보내주게 된다. MOM 서비스는 RTI 설정 파일을 통해 사용하도록 할 수 있기 때문에 페더레이션 연동을 고려하지 않고 개발된 페더레이트도 수정 없이 사용할 수 있다. 하지만 MOM 서비스를 사용하게 되면 RTI가 내부에서 처리해야 되는 일이 증가하기 때문에 성능이 저하된다. 두 번째로 Fig. 3과 같이 사용자 페더레이트와 대리자의 정보 교환 프로토콜을 정의하고 사용자 페더레이트가 필요한 정보를 보내도록 하는 방법이 있다. 이 경우에는 MOM 서비스를 이용하지 않기 때문에 RTI가 더 좋은 성능을 보일 수 있다. 하지만 몇몇 서비스에 대해서 사용자 페더레이트가 정보를 보내줘야 하기 때문에 기존 페더레이트를 사용하고자 할 경우, 수정이 필요하게 된다.

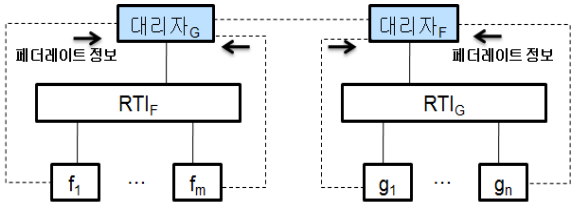

Fig. 3. 대리자-사용자 프로토콜

페더레이션 연동 구조로 MOM 서비스의 이용과 대리자-사용자 간 프로토콜의 선택에서 중요한 것은 MOM 서비스에 사용에 의한 RTI 성능 저하이다. 성능 저하가 크지 않다면 새로운 프로토콜을 정의할 필요가 없다. 반면, 성능 저하가 크다면 추가 구현을 하더라도 더 빠른 시뮬레이션을 위해 새로운 프로토콜을 사용할 수 있게 된다. 따라서 페더레이션 연동 구조를 선택하기 위해서는 MOM 서비스 이용에 따른 페더레이션 연동 성능의 측정이 필요하다.

나. MOM 서비스

HLA는 페더레이션의 내부 정보 확인과 RTI의 기능을 통제할 수 있도록 RTI가 MOM 서비스를 제공하도록 명시하고 있다[6]. MOM은 각 페더레이트 및 페더레이션의 정보를 저장하는 객체 모델로 사용자 객체 모델과 같은 형식으로 정의되어 있어 RTI는 해당 모델을 이용하여 페더레이트로 정보를 제공한다. 일반적으로 MOM 서비스를 사용하게 되면 RTI의 성능이 저하되기 때문에 설정파일에서 MOM 서비스 사용여부를 설정할 수 있도록 되어 있다.

Fig. 4는 MOM 서비스는 두 가지 기능을 보여준다. 각 페더레이트는 설정에 따라 다른 페더레이트들을 관리 하는 페더레이트와 관리를 받는 페더레이트로 나뉘어진다. 관리자 페더레이트는 MOM 서비스를 통해 참여한 페더레이션과 다른 페더레이트들의 정보를 확인할 수 있다. 각 페더레이트 이름, 핸들, 객체 관리 선언 정보 등 RTI가 내부에서 관리하는 여러 정보들을 관리자 페더레이트는 받게 된다. 이를 위해 RTI는 주기적으로 정보를 갱신하거나 페더레이트가 서비스를 호출할 때마다 해당 정보를 보내주는 작업을 하게

된다. 많은 정보를 이용하게 될수록 성능의 저하가 클 수 있기 때문에 페더레이션 연동에서는 필요한 최소한의 정보를 사용해야만 한다. 따라서 대리자 페더레이트는 조회/콜백 서비스를 통해 얻을 수 없는 다른 페더레이트의 서비스 호출 정보만을 MOM 서비스를 통해 확인한다. 두 번째 기능으로 관리자 페더레이트가 다른 페더레이트가 특정 서비스를 호출하도록 할 수 있다. 다른 페더레이트들의 동작을 관리하도록 하기 위한 기능으로 페더레이션 연동에서는 대리자 페더레이트가 다른 시뮬레이션에 영향을 주어서는 안 되기 때문에 사용하지 않는다.

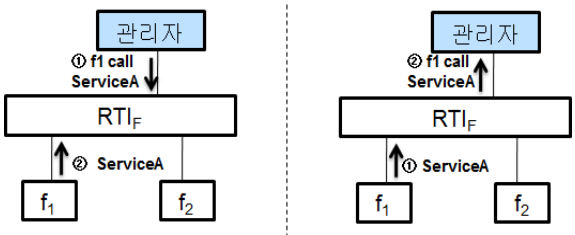

Fig. 4. MOM 서비스

MOM 서비스를 이용하게 되면 RTI는 페더레이트로 정보를 제공하거나 페더레이트로부터 정보를 받아 처리해야하는 동작을 하게 된다. 페더레이션 연동 시에 사용되는 서비스 호출 상태 보고를 사용하게 되면 RTI는 참여한 페더레이트들이 서비스를 호출할 때마다 호출 페더레이트, 해당 서비스 이름, 매개 변수 값 등 모든 정보를 대리자 페더레이트로 제공한다. HLA 정의에서 특정 서비스만 보고하도록 할 수는 없기 때문에 RTI가 처리해야 되는 작업량이 매우 많아지게 되고 이로 인한 성능 저하가 발생하게 된다.

3. 정보 교환 실험

MOM 서비스 사용에 의한 페더레이션 연동 성능을 분석하기 위해서 성능측정 페더레이트들를 두 페더레이션으로 나누어 구성하고 연동하였다. 연동된 구조에서 전체 시뮬레이션을 수행하고 그 성능을 측정하였다. 실험에는 대리자-사용자 페더레이트 프로토콜 사용한 구조와 MOM 서비스를 이용하는 구조가 사용되었다. 두 구조는 동기화 서비스 등 문제가 되는 서비스에 대해 정보를 얻는 부분을 제외하고 동일하게 구현되어 MOM 서비스 사용에 의한 성능 차이만을 측정할 수 있도록 하였다. 또한 일반적인 HLA/RTI 환경에서의 시뮬레이션 성능과 비교하기 위하여 한 페더레이션에서 모든 페더레이트들을 참여시켜 실험을 수행하여 두 구조의 성능을 비교하였다. 실험은 Intel Core (TM) i5 2.67Ghz, 3GB RAM 컴퓨터들과 1Gbps LAN 환경에서 수행되었고 RTI는 DMSO RTI NG 1.3v6가 사용되었다.

페더레이션의 성능에 영향을 주는 요소로는 페더레이트 수, 등록된 객체 수 등 여러 가지 요소가 있다[7]. MOM 서비스 사용 여부도 여기에 포함되기 때문에 MOM 서비스를 사용하는 구조는 한 페더레이션에서의 성능에 영향을 주게 된다. 페더레이션 연동 구조는 이에 더해서 연동 기능으로 인한 추가 요소가 있다. 대리자 페더레이트의 내부 변환 작업, 대리자 간의 정보 교환 등 1개의 페더레이션을 운용하는 시뮬레이션에서는 시행되지 않는 작업들을 필요하게 되어 성능에 영향을 줄 수 있다. 본 논문에서는 실험을 통해서 각 요소들에 의한 페더레이션 연동 성능을 분석해보았다.

가. Object Throughput

Throughput은 단위 시간당 얼마나 많은 메시지를 주고 받을 수 있는 지를 측정한 값이다[8,9]. 연동 시뮬레이션은 페더레이트들이 메시지를 주고받으며 진행되기 때문에 빠른 시뮬레이션을 위해서는 페더레이트는 짧은 시간 안에 많은 메시지를 교환할 수 있어야만 한다. HLA/RTI 환경에서 페더레이트가 메시지를 보낼 때, 대상의 수신 확인과 관계없이 보내기 때문에 Sending Throughput과 Receiving Throughput이 다른 값을 갖게 된다.

Sending Throughput을 측정하기 위해서는 다음과 같은 동작을 하는 페더레이트들을 이용한다. 먼저 송신 페더레이트는 최대한 빠르게 정해진 횟수만큼 객체정보를 RTI로 갱신하고 걸리는 시간을 측정한다. 수신 페더레이트는 RTI로부터 객체 갱신 정보를 받게 되고, 모든 정보를 받는데 걸리는 시간을 측정한다. Throughput은 (메시지 수)/(갱신 시간)으로 측정 할 수 있다. 본 실험에서는 10바이트 크기의 10만개의 메시지를 보내도록 하여 Throughput을 측정하였다.

1) 페더레이션 간 데이터 교환

페더레이션 간의 데이터 교환 성능을 측정하기 위해서 두 페더레이션에 송/수신 페더레이트들을 분산시

키고 시뮬레이션을 수행해본다. 먼저, 데이터가 한 방향으로만 전송되는 경우를 테스트해보았다. Fig. 5는 실험 페더레이션 구성을 보여준다. 한 페더레이션은 송신 페더레이트들로 구성하고, 다른 페더레이션은 수신 페더레이트로만 구성한다.

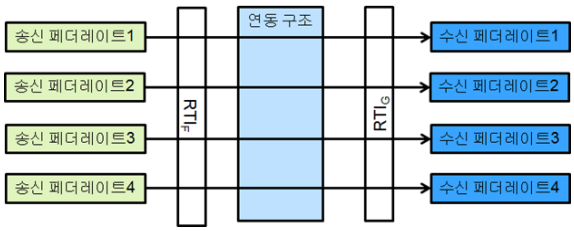

Fig. 5. 한 방향 데이터 교환 실험

Fig. 6은 페더레이트 수에 따른 Object Sending Throughput을 측정한 결과이다. MOM 서비스를 이용할 경우, 하나의 페더레이션을 구성할 때에 비해 1/16 정도로 감소하는 것을 확인할 수 있다. RTI는 송신 페더레이트가 메시지를 보낼 때마다 갱신 서비스 호출 정보를 MOM 서비스를 통해 보내게 된다. 매 호출 시 마다 새로운 정보를 구성해서 보내야 하기 때문에 메시지를 처리량이 크게 감소하는 것으로 보인다. 반면, 대리자-사용자 프로토콜을 이용할 경우에는 한 페더레이션을 구성할 때와 거의 같은 값을 보인다. 송신 페더레이트는 수신 페더레이트의 동작과 상관없이 진행할 수 있기 때문에 MOM 서비스를 이용하지 않을 경우에는 대리자의 유무와 상관없는 성능을 보이게 된다. 각 구조에서 페더레이트 수가 증가함에 따라 Sending Throughput은 조금 감소하기는 하지만 크게 영향을 받지는 않았다.

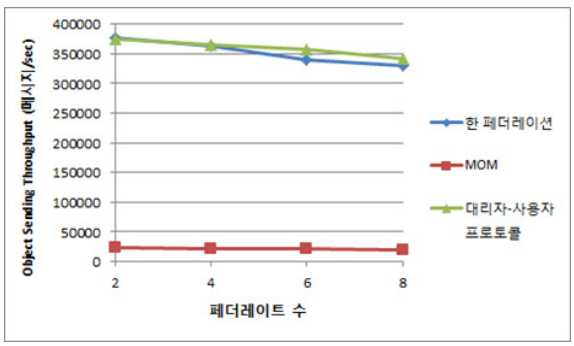

Fig. 6. Object Sending Throughput
(페더레이트 수에 따른 변화)

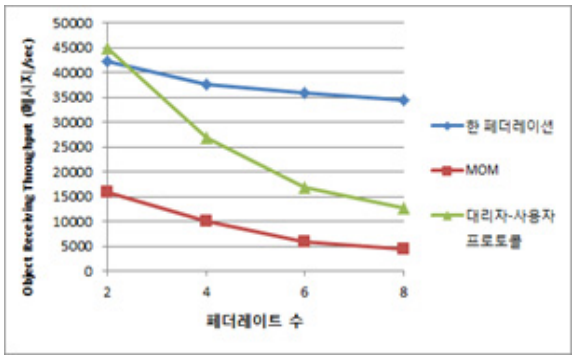

Fig. 7. Object Receiving Throughput
(페더레이트 수에 따른 변화)

Fig. 7은 페더레이트 수에 따른 Object Receiving Throughput을 보여준다. MOM 서비스를 이용할 경우, 하나의 페더레이션을 구성할 때에 비해 작은 값을 보이고 페더레이트 수가 증가함에 따라 감소하는 것을 확인할 수 있다. 대리자-사용자 프로토콜을 사용하는 경우에는 페더레이트가 2개일 때는 한 페더레이션에서와 차이가 없었지만 페더레이트 수가 증가함에 따라 차이가 벌어지는 것을 확인할 수 있다. 두 구조 모두 대리자가 한 페더레이션의 수신 정보를 모두 받아서 다른 쪽으로 넘겨주기 때문에 대리자의 정보 교환 성능 한계로 인해 페더레이트 수가 증가할수록 Receiving Throughput이 감소하는 것으로 보인다. MOM 서비스를 이용하는 경우에는 대리자간 정보교환 성능 이외에 페더레이트가 정보를 수신할 때마다 수신 서비스 정보를 MOM 서비스를 통해 대리자로 보내기 때문에 적은 페더레이트에서도 더 작은 값을 갖게 되는 것으로 보인다.

보다 일반적인 경우에서도 확인하기 위해 Fig. 8과 같이 페더레이션을 구성하고 메시지가 양방향으로 흐르는 경우를 측정하였다. 페더레이트 4, 8개 인 경우에는 양방향으로 각각 2종류의 데이터가 전송되고 6개인 경우에는 한 방향은 한 종류, 다른 방향은 두 종류의 데이터가 전송되도록 하고 측정하였다. 한 페더레이션의 실험은 데이터가 다른 페더레이션으로 전송되지 않으므로 이전 실험과 동일하다.

Fig. 9는 메시지가 양방향으로 전송될 때의 Sending Throughput을 보여준다. RTI에서의 데이터 전송은 받는 페더레이트의 동작과 상관없이 진행할 수 있기 때문에 모든 구조에서 한 방향으로 데이터가 전송될 때와 별다른 차이를 보이지 않는다.

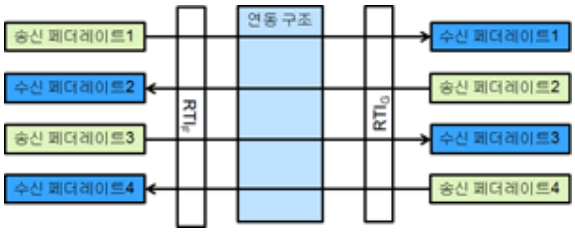

Fig. 8. 양방향 데이터 교환 실험 구조

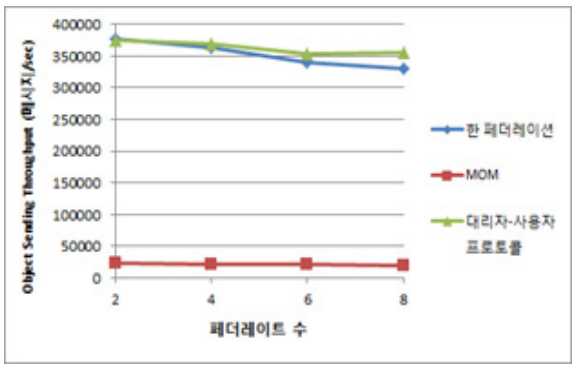

Fig. 9. Object Sending Throughput
(양방향 데이터 교환)

Fig. 10은 메시지가 양방향으로 전송될 때의 Receiving Throughput을 측정한 결과이다. 한 방향으로 데이터가 전송될 때와 별다른 차이를 보이지 않는다. 한 방향으로 보낼 때와 비교하면 대리자의 송신량 또는 수신량이 1/2 송신량 + 1/2 수신량으로 바뀐 것이 되는데, 현재 구현된 대리자는 두 경우가 같은 성능을 보이는 것으로 확인되었다. 이 실험 결과는 대리자의 데이터 교환 방법에 따라 다른 결과를 보일 수도 있다.

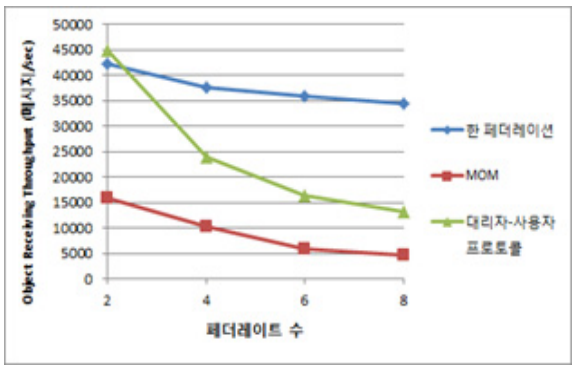

Fig. 10. Object Receiving Throughput
(양방향 데이터 교환)

2) 페더레이션 내의 데이터 교환

이번 실험에서는 페더레이션 연동 환경에서의 시뮬레이션을 수행할 때, 페더레이션 연동 환경을 구비한 페더레이션 내에서의 데이터 교환에 대한 영향을 측정한다. Fig. 11에서와 같이 두 페더레이션에 각각 송/수신 페더레이트 짝을 배치하고 실험을 수행한다. 이 실험 시뮬레이션에서는 다른 페더레이션으로 데이터가 전송되는 일이 없기 때문에, 독립적인 두 개의 시뮬레이션을 수행하는 것과 같다. 한 페더레이션을 수행할 때에 비해서 한 페더레이션 당 처리해야 되는 데이터량이 적어지게 되어 그 영향을 확인할 수 있다.

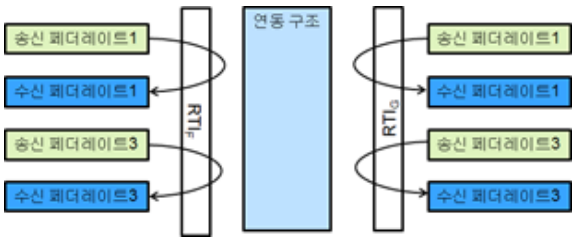

Fig. 11. 페더레이션 내의 데이터 교환 실험 구조

Fig. 12은 페더레이션 내에서 데이터가 전송될 때의 Sending Throughput 측정 결과이다. 이전 실험에서 측정한 결과를 봤을 때, 한 페더레이션에서 페더레이트가 늘어나도 성능이 크게 감소하지 않았기 때문에 페더레이션 간 데이터가 교환될 때와 별다른 차이를 보이지 않는다. MOM 서비스를 이용한 경우, 페더레이션 간의 정보교환에 상관없이 서비스 호출 시 마다 해당 정보를 보내줘야 하는 작업을 해야 하기 때문에 낮은 성능을 보이게 된다.

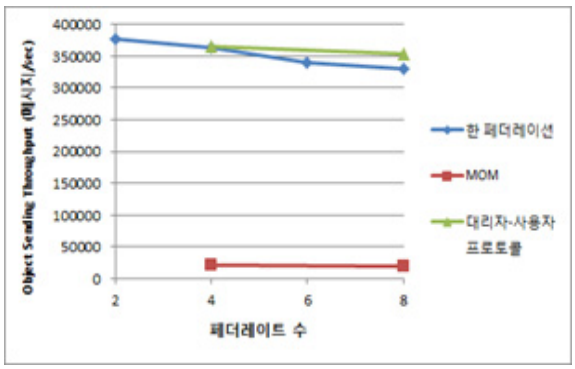

Fig. 12. Object Sending Throughput
(페더레이션 내 교환)

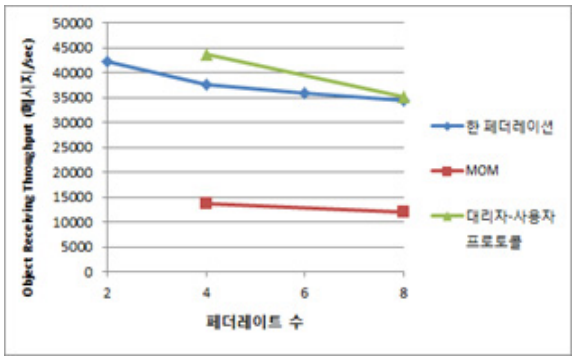

Fig. 13. Object Receiving Throughput
(페더레이션 내 교환)

Fig. 13은 데이터가 내에서 데이터가 교환할 때의 Receiving Throughput을 보여준다. 대리자-사용자 프로토콜을 사용하는 경우에는 한 페더레이션을 구성했을 때와 비슷하거나 조금 나은 성능을 보여준다. 한 RTI에서 처리하는 데이터양이 감소한 영향으로 확인된다. 반면 MOM 서비스를 이용하는 경우는 MOM 서비스 사용에 의한 성능 감소로 인해 낮은 성능을 보인다. 페더레이션 간에 데이터가 교환될 때와 비교하면 초기에는 값이 비슷하지만 대리자를 통해서 데이터가 전송되지 않아도 되기 때문인 페더레이트 수가 늘어도 감소폭이 적다.

3) 객체 수 증가에 따른 Throughput

HLA/RTI 환경에서는 공유 객체를 식별하기 위해서 등록 시 할당되는 핸들 값을 사용한다. 페더레이션 사이에서 공유되는 객체의 경우 대리자 페더레이트가 대표 페더레이션에 등록된 객체와 동일한 객체를 참여페더레이션에 등록하여 사용한다. 이 때, 객체의 핸들 값은 페더레이션마다 다르기 때문에 대리자는 변환 정보를 저장해두고 갱신 시 사용하게 된다. 객체 수가 증가하게 되면 이를 변환하는 작업이 성능에 영향을 줄 수 가 있다. 본 실험에서는 그를 측정하기 위해 Fig. 4와 같이 페더레이트들을 배치하고 여러 객체를 등록한 이후, 정보 교환 실험을 수행한다.

Fig. 14는 객체 수 변화에 따른 Sending Throughput을 보여준다. 이전 실험에서 확인했던 바와 같이 MOM 서비스를 사용하는 경우가 한 페더레이션 나 대리자-사용자 프로토콜을 사용한 구조에 비해서 더 나쁜 성능을 보이는 것으로 확인할 수 있다. 하지만 모든 구조에서 등록 객체가 늘어나더라도 성능에 별다른 영향이 없었다. 객체를 갱신할 때, 영향을 주는 요소는 등록 객체를 찾는 과정과 정보를 보내는 과정이다. 한 페더레이션에서의 실험 결과를 통해 RTI에서는 객체가 많이 등록되더라도 갱신하는 속도에 별다른 영향을 주지 않는 것을 확인할 수 있다. 실험 결과를 보면 두 연동 구조에서 모두 등록 객체가 늘어나더라도 값이 큰 변화가 없는 것을 확인할 수 있다. 객체 변환 과정이 RTI와 마찬가지로 객체를 찾는 데 걸리는 시간이 얼마나 걸리지 않아 별다른 영향을 주지 않는 것으로 확인되었다.

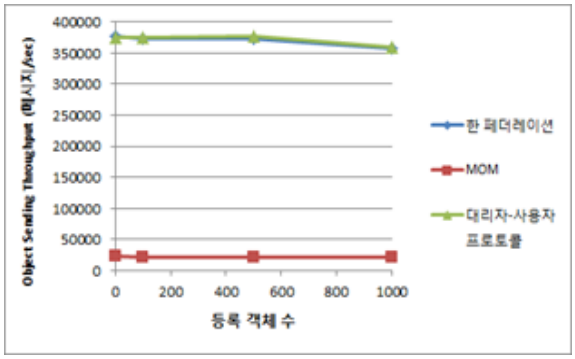

Fig. 14. Object Sending Throughput
(객체 수에 따른 변화)

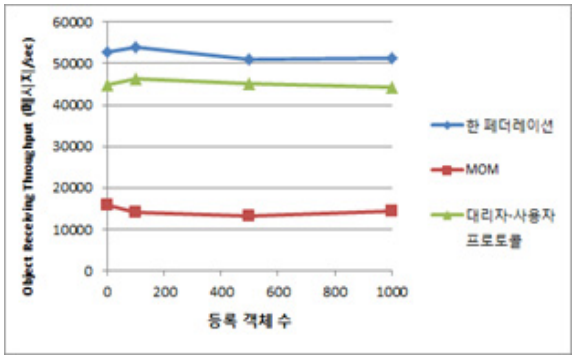

Fig. 15. Object Receiving Throughput
(객체 수에 따른 변화)

Fig. 15는 객체 수 변화에 따른 Receiving Throughput을 보여준다. Sending Throughput과 마찬가지로 Receiving Throughput도 등록 객체 수가 늘어나도 별다른 영향이 없는 것을 확인할 수 있다. Receiving Throughput의 경우 Sending Throughput보다 작기 때문에 보내는

속도에 영향이 없다면 받는 속도에도 영향이 없게 된다.

4) 메시지 크기에 따른 Throughput

이번 실험에서는 페더레이션 Fig. 4와 같이 송수신 페더레이트들을 양쪽에 분산 시키고 송신 페더레이트와 수신 페더레이트들을 각각 다른 페더레이션에 분산 시키고 메시지 크기 변화에 따른 Throughput을 측정하였다.

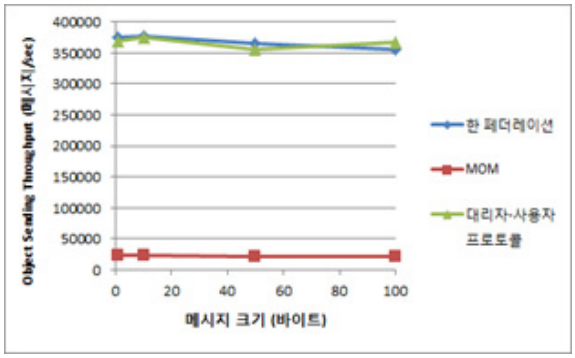

Fig. 16. Object Sending Throughput (메시지 크기)

Fig. 16은 메시지 크기 변화에 따른 Sending Throughput을 보여준다. 모든 구조에서 메시지의 크기가 증가하더라도 시간당 보낼 수 있는 메시지의 수는 큰 변화가 없는 것으로 나타났다. 따라서 메시지의 크기가 너무 커서 네트워크 성능의 한계치가 되지 않는 이상 Sending Throughput은 그 영향을 받지 않는 것으로 확인된다.

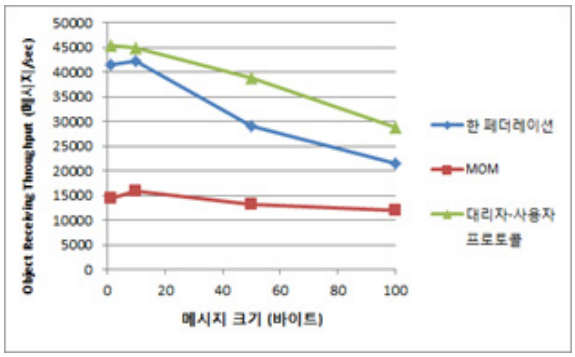

Fig. 17. Object Receiving Throughput (메시지 크기애 따른 변화)

Fig. 17은 메시지 크기 변화에 따른 Receiving Throughput을 보여준다. 한 페더레이션과 대리자-사용자 프로토콜의 경우에는 메시지의 크기가 증가함에 따라 성능이 감소하는 것을 확인할 수 있다. 메시지를 보낼 때의 네트워크성능의 한계로 감소가 생기는 것으로 분석된다. 본 실험에서는 대리자-사용자 프로토콜을 한 경우가 한 페더레이션에서보다 성능이 더 좋게 나왔는데, 큰 데이터를 보낼 때, 대리자의 네트워크 라이브러리가 RTI보다 더 좋은 성능을 보여주기 때문으로 보인다. 이 경우는 RTI나 대리자의 구현에 따라 다른 결과를 보일 수도 있다. MOM 서비스를 사용하는 경우에는 약간 감소하기는 했지만 큰 변화는 없었다. MOM 서비스 사용에 의한 성능저하로 인해 메시지를 받을 수 있는 양이 감소하는 영향이 네트워크 성능에 의한 영향보다 크기 때문으로 보인다. 따라서 메시지 크기가 더 증가하게 되면 모든 구조가 비슷한 값을 보일 것으로 예상된다.

나. Object Latency

Latency는 메시지가 송신 페더레이트에서 수신 페더레이트로 전달되는데 걸리는 지연 시간을 의미한다[8~10]. 페더레이트가 특정 정보를 받을 때까지 진행을 못하게 되는 동작을 보일 경우, 아무리 많은 데이터를 처리할 수 있어도 진행을 할 수 없기 때문에 지연 시간이 중요하게 된다.

Latency를 측정하기 위해서는 다음과 같은 동작을 하는 페더레이트들을 이용한다. 먼저 송신 페더레이트는 객체정보를 RTI로 갱신하고 수신 페더레이트의 응답을 기다린다. 수신 페더레이트는 갱신된 객체 정보를 받게 되면 객체를 갱신하는 것으로 응답하고 다시 객체가 갱신되는 것을 기다린다. 이러한 과정을 정해진 횟수만큼 반복하여 전체 실험에 걸리는 시간을 측정하면 Latency는 실험 수행 시간/(왕복횟수×2)가 된다. 본 실험에서는 10바이트 크기의 메시지가 100번 왕복하도록 하여 Latency를 측정하였다.

Fig. 18은 페더레이트 수에 따른 Object Latency를 보여준다. 두 페더레이션 연동 구조에서는 한 페더레이션보다 성능이 나빠지는 것을 확인할 수 있었다. 그 이유는 다음과 같다. 페더레이션 연동을 통해 메시지가 전달되기 위해서는 먼저 송신 페더레이트에서 RTI를 통해 대리자로 메시지가 전달 되고, 다른 대리자로 해당 정보가 전달 된다. 그리고 대리자가 RTI를 통해 수신 페더레이트를 통해 정보를 전달하게 된다. 이와

같이 RTI를 통해 메시지가 전달되는 과정을 두 번 거치게 되고 대리자간의 전달과정 과 객체와 관련된 정보 획득 및 변환 과정도 거치게 된다. 두 페더레이션 연동 구조의 성능차이는 없었다. MOM 서비스를 사용하더라도 메시지가 지연되는 시간에는 별다른 영향을 주지 않는 것으로 확인되었다.

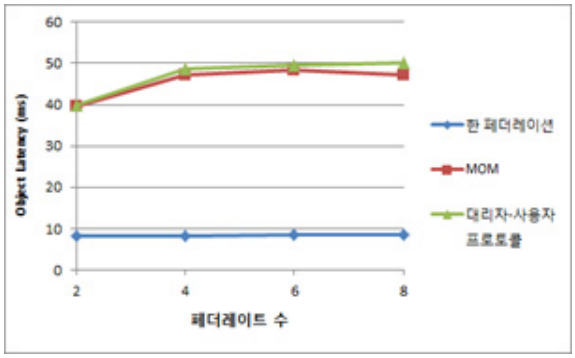

Fig. 18. Object Latency

다. Interaction Throughput

본 실험에서는 메시지로 Interaction을 사용하여 Throughput을 측정하였다. Interaction의 경우 Obejct와 달리 객체 인스턴스를 등록하지 않고 사용되기 때문에 그에 대한 영향이 있는지 확인해본다.

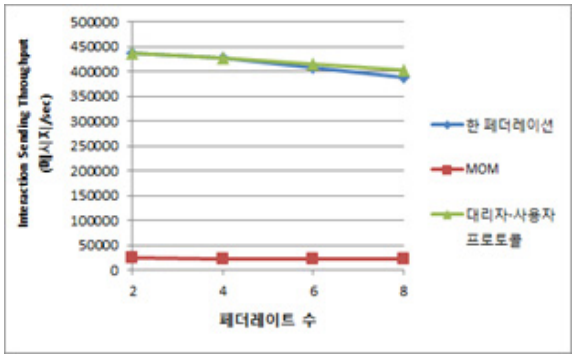

Fig. 19. Interaction Sending Throughput

Fig. 19는 페더레이트 수에 따른 Interaction Sending Throughput을 보여주고 Fig. 20은 Interaction Receiving Throughput을 보여준다. 두 실험 결과 모두 객체를 하나 등록하여 실험한 Object Throughput과 비슷한 결과를 보여준다. 즉, Object와 Interaction의 데이터 교환 성능은 별다른 차이가 없는 것으로 확인되었다.

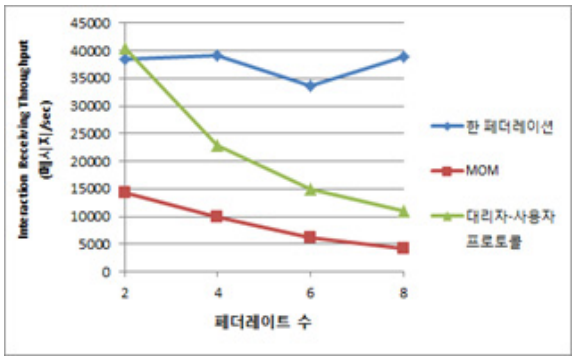

Fig. 20. Interaction Receiving Throughput

4. 결 론

본 논문에서는 페더레이션 연동을 할 때, MOM 서비스가 미치는 영향을 확인하기 위한 실험을 통해 확인하였다. RTI를 수정하지 않고 페더레이션 연동을 할 수 있도록 MOM 서비스를 사용하는 구조와 대리자-사용자 프로토콜을 사용하는 구조에서의 연동 성능을 측정하였다. 실험을 통해 MOM 서비스를 이용하게 될 경우 Sending Throughput은 크게 감소하고 Receiving Throughput도 절반 이하로 감소하는 것을 확인할 수 있었다. 따라서 다른 시뮬레이터의 응답여부에 관계없이 진행할 수 있는 시뮬레이션의 비중이 큰 경우 데이터 처리량이 중요하므로 MOM 서비스에 의한 성능 감소가 크게 된다는 것을 확인 할 수 있다. 반면 Latency는 별다른 차이를 보이지 않았다. 다른 시뮬레이터의 응답이 올 대까지 진행할 수 없는 시뮬레이션의 비중이 큰 경우에는 MOM 서비스 유무에 상관없이 비슷한 성능을 보이게 될 것이다.

본 연구를 통해 측정된 실험 결과를 이용하여 시뮬레이션의 성능의 요구 사항에 따라서 사용자는 페더레이션 연동 구조를 선택할 수 있을 것이다. 추후에는 페더레이션 연동시의 시간 진행 속도에 관련된 실험과 시간 진행 방법의 최적화 방안에 대해서 연구가 필요할 것으로 판단된다.

후 기

본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다(UD110006AD).

References

[1] IEEE Std. 1516-2000, “IEEE Standard for Modeling and Simulation(M&S) High Level Architecture(HLA) - Framework and Rules”, IEEE Computer Society, 2000.

[2] Michael D. Myjak and Sean T. Sharp, “Implementations of Hierarchical Federations”, Simulation Interoperability Workshop, 99F-SIW-180, 1999.

[3] J. Dingel, D. Garlan, and C. Damon, “A Feasibility Study of the HLA Bridge”, Technical Report, Department of Computer Science, Carnegie Mellon University, CMU-CS-01-103, 2001.

[4] M. W. Yoo, T. G. Kim, “Design and Implementation of Surrogates for Interoperation of HLA Federations”, 2009 Joint SISO/SCS European Multi-Conference, 09E-SIW-017, 2009.

[5] 유민욱, 김탁곤, “에이전트 기반 HLA 페더레이션 연동을 위한 만능 어댑터 확장”, 한국군사과학기술학회 ’09종합학술대회, pp. 485～488, 2009년 8월.

[6] IEEE Std. 1516-2000, “IEEE Standard for Modeling and Simulation(M&S) High Level Architecture(HLA) - Federate Interface Specification”, IEEE Computer Society, 2001.

[7] B. Watrous, L. Granowetter and D. Wood, “HLA Federation Performance : What Really Matters?”, Proceedings of the 2006 Fall Simulation Interoperability Workshop, Stockholm, 2006.

[8] P. Knight, A. Corder, R. Liedel, J. Giddens, R. Drake, C. Jenkins and P. Agarwal, “Evaluation of Run Time Infrastructure(RTI) Implementations”, Huntsville Simulation Conference, 2002.

[9] R. M. Fujimoto and P. Hoare, “HLA RTI Performance in High Speed LAN Environments”, Proceedings of the Fall Simulation Interoperability Workshop. 1998.

[10] T. Burks, T. Alexander, K. Lessmann and K. G. LeSueur, “Latency Performance of Various HLA RTI Implementations”, Simulation Interoperability Workshop, paper 015, 2001.