HLA/RTI 기반 분산시뮬레이션의 객체 관리 성능향상

김세환*, 채수환*, 이상민**
*한국항공대학원 컴퓨터공학과
**REALTIMEVISUAL (주)
e-mail:fnsksk@hotmail.com

Object Management Performance Enhancement of Distributed Simulation Based on HLA/RTI

Se-Hwan Kim*, Soo-Hoan Chae*, Sang-Min Lee**
*Dept of Computer Engineering, Graduate Shchool, HanKuk Aviation University
**REALTIMEVISUAL

요 약

HLA는 상호운영성과 재사용성을 목적으로 미국 국방성 산하 기관인 DMSO에서 개발되어진 분산 시스템 프레임워크이다. 이러한 HLA가 민간 부분 적용을 위해 표준이 발표되었고, HLA는 여러 분야의 범용성을 가지기 위해 복잡한 구현 시스템을 가지게 된다. 특히, federate들내에 존재하는 객체와 객체의 속성을 사용하기 위해서는 복잡한 절차를 구현해한다. 또한 객체의 속성변화를 반영하기 위해서는 많은 통신 부하가 발생한다. 이런 단점을 해결하기 위해 HLA 기반의 시뮬레이션에서 객체를 분리 관리하는 방안을 제안한다.

1. 서론

HLA(High Level Architecture)는 개발자에게 시뮬레이션 응용을 정의 및 구성 할 수 있게 해주는 범용 프레임워크이다. HLA는 다른 범주의 시뮬레이션들 간의 상호운용성(interoperability) 및 재사용성(reuse) 제공을 목표로 한다. 이 목표들을 실현하기 위해 HLA 중요한 두 가지의 구성 요소를 가진다. 첫 번째 구성요소는 OMT(Object Model Template)로써 재사용성을 제공하기위해 특정 모델에서 사용되어지는 데이터들에 대한 문서화이다. 두 번째 구성요소는 Federate Interface Specification으로써 시뮬레이션 모델들 간에 상호운용성을 제공하는 범용 통신 인터페이스이다.

즉, HLA는 시뮬레이션의 수행중에 데이터 교환 및 운용에 대한 조율, Federate Interface Specification에 의하여 정의되어지는 서비스들의 집합을 제공하기 위한 RTI(runtime infrastructure) 소프트웨어를 이용하는 아키텍처이다.[1]

초기 HLA는 미국 국방성(DOD) 산하 기관인 DMSO (Defense Modeling & Simulation Office)의 주도하에 JWAR(Joint Warfare System), JSIM(Joint Simulation System), JMASS(Joint Modeling and Simulation System)등 군 관련 시뮬레이션을 위해 개발되었다. 하지만 HLA의 민간부분 적용의 논의가 활발해지면서 최근 2000년 HLA 버전 1.3을 기반으로 한 IEEE 1516 버전이 표준으로 책정되었다. 민간부분의 적용을 염두에 둔 HLA는 여러 분야에서 응용될 수 있는 범용성을 가져야 한다. 이 결과로 HLA의 구현 절차는 복잡해진다. 특히, 객체와 속성을 관리하기 위한 초기화 및 종료 절차에 이러한 현상이 두드러진다. 또한 시뮬레이션 수행중 객체의 속성 변화 전송시 불필요한 내용을 포함하여 전달하는 경우가 발생함에 따라 통신 부하 역시 증가 한다.

본 논문에선 이러한 문제점을 해결하기위해 HLA 기반의 분산 시뮬레이션에서 객체들을 분리 관리하

여 구현 절차의 간소화와 및 통신 부하를 줄이는 방법을 제안한다.

2. 관련연구

HLA는 시뮬레이션들을 더 큰 시뮬레이션으로 통합가능하게 하는 아키텍처이다. 예를 들어 항공기 기종별 조종을 위한 시뮬레이션과 관제사들을 훈련하기 위한 항공관제 시뮬레이션이 있다고 가정하자. HLA를 이용해 각 요소들을 통합한다면, 거대한 항공 관련 시뮬레이션이 가능해진다.

(그림 1)에 보인 바와 같이 통합되어진 거대한 항공 관련 시뮬레이션을 federation이라한다. 또한 federation에 참가하는 각 요소(예에선 항공기 기종별 시뮬레이션, 항공관제 시뮬레이션)들을 federate라 부른다.[2]

2.1. HLA의 구성요소

HLA를 이용한 시뮬레이션을 구현하기 위해서 필요한 요소로 HLA Rules, Interface Specification, OMT의 3가지가 있다[3][4][5].

(1) HLA Rules : 시뮬레이션 수행중에 federate와 federation이 지켜야하는 규칙들의 모임.

(2) Interface Specification : federate들 사이에서 정의되어진 기능적 인터페이스의 설계, 즉, RTI.

(3) OMT : SOM(Simulation Object Model)과 FOM(Federation Object Model)을 기술하기 위한 표준 문서화 방법.

2.2. RTI 서비스

(그림 1)에서와 같이 federation의 제어 관리, 각 federate들의 데이터 교환등을 위하여 아래의 6가지 서비스를 지원한다[4].

(1) Federation Management : federation 수행의 생성, 제어 관리, 수정, 삭제를 관리하는 서비스 집합

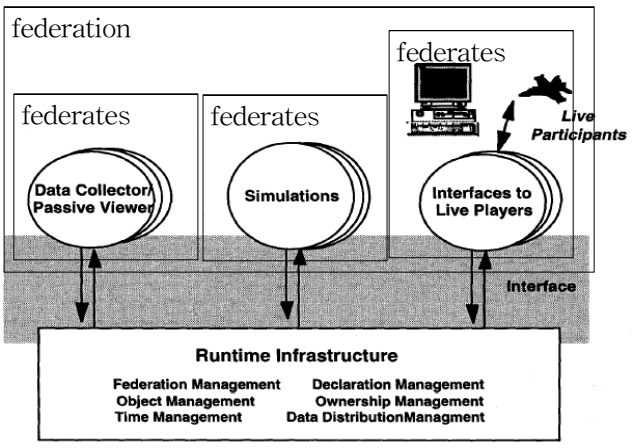
(2) Declaration Management : 참가한 federate들이 사용할 정보의 생성 및 전달을 관리하는 서비스 집합

(3) Object Management : 참가한 federate들이 사용한 객체의 등록, 수정, 삭제 및 interaction의 교환을 관리하는 서비스 집합

(4) Ownership Management : 참가한 federate들 사이의 속성에 대한 소유권 이동을 관리하는 서비스 집합

(5) Time Management : federation 수행중의 시간진행을 관리하는 서비스의 집합

(6) Data Distribution Management : 참가한 federate들 통신구역을 정의하는 서비스의 집합



(그림 1) HLA federation의 구조

3. HLA의 객체관리 문제점

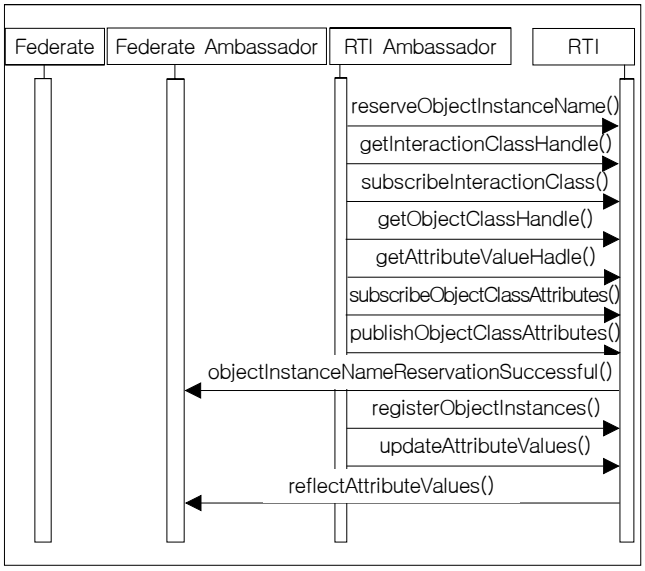
HLA를 이용한 시뮬레이션에서 객체를 사용하기 위해서는 모든 federate에 사용되어질 객체를 (그림 2)의 FDD(FOM Document Data)에서처럼 미리 정의해야한다. FDD에 정의되어진 객체는 Declaration Management 서비스를 이용하여 object class를 다루기 위한 핸들을 정의해야하며, 객체에 소속되어진 각 속성 역시 핸들을 정의해야한다. 이렇게 정의되어진 객체는 PublishObjectClassAttributes()를 통하여 공표되어지고 SubscribeObjectClassAttributes()로 federation에 참가한 federate들에게 객체에 대하여 알린다. 이렇게 정의되어진 객체를 실제로 사용하기 위해서는 Object Management 서비스에서 제공하는 RegisterObjectInstance()를 통하여 object instance를 등록하고, UpdateAttributeValues()를 이용하여 객체의 속성을 실제로 수정하여 참가한 federate들에게 전달하고, ReflectAttributeValues()를 통하여 반영되어진다. 이 과정은 (그림 3)과 같다.

이러한 일련의 과정들은 새로운 객체, 속성의 추가 등의 동적인 사용을 저해하는 요소가 된다. 또한 UpdateAttributeValues()를 통한 federate들 간의 속성의 값 전달시 불필요한 데이터가 발생하게 된다. (그림 4)와 같은 해당 객체의 속성들을 모두 묶은 구조의 AttributeHandleValueMap의 데이터 형을 사용하기 때문이다. 만약 해당 객체의 속성 값으로 x, y, z의 3차원 위치 좌표를 가진다면, 해당 객체가 x 좌표만 이동하여도 좌표를 모두 전송해야한다. 이

경우 참여한 federate의 수가 많을 경우 발생하는 통신 부하는 큰 폭으로 증가한다. HLA에서 사용되어지는 통신 방법은 peertopeer 방식이 아닌 broadcasting 방식을 사용하기 때문이다.

```
<objectClass name="Restaurant" sharing="Neither">
        <attribute name="position" ownership="NoTransfer"
        sharing="Publish" dimensions="DimX"
        transportation="HLAreliable" order="TimeStamp"/>
<objectClass name="Serving" sharing="Neither">
        <attribute name="type" dimensions="NA"
        transportation="HLAreliable" order="TimeStamp"/>
</objectClass>
<objectClass name="Boat" sharing="Neither">
        <attribute name="spaceAvailable" dimensions="NA"
        transportation="HLAreliable" order="TimeStamp"/>
        <attribute name="cargo" dimensions="NA"
        transportation="HLAreliable" order="TimeStamp"/>
</objectClass>
<objectClass name="Actor" sharing="Neither">
        <attribute name="servingName" dimensions="NA"
        transportation="HLAreliable" order="TimeStamp"/>
<objectClass name="Chef" sharing="Neither">
        <attribute name="chefState" dimensions="NA"
        transportation="HLAreliable" order="TimeStamp"/>
</objectClass>
<objectClass name="Diner" sharing="Neither">
        <attribute name="dinerState" dimensions="NA"
        transportation="HLAreliable" order="Receive"/>
</objectClass>
```

(그림 2) FDD내의 객체 정의



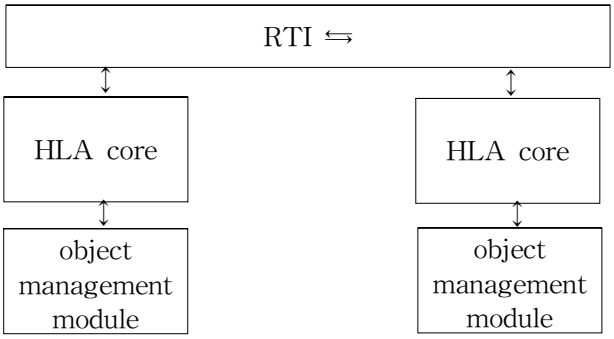
(그림 3) HLA 객체의 정의 및 통신 순서

handle	-> add	(x_point	y_point	z_point)
		.....	.....	.....

(그림 4) AttributeHandleValueMap의 구조

4. 문제 해결 방안

위에서 언급한 바와 같이 객체의 사용을 위한 HLA의 복잡한 선언 과정, 불필요한 데이터 전송이 발생하는 통신 방법을 해결하기 위해서 새로운 객체 관리 모듈을 설계하였다. 이 모듈은 (그림 5)과 같이 구성되어져 있다. HLA core와 object management module 사이에 공통적으로 속성의 식별자 (id)만을 유지하고 HLA core는 객체에 대한 정보를 포함하지 않는다. 즉, (그림 3)의 object class, object instance의 선언 부분은 HLA core에서 삭제되어진다. 이 모듈은 HLA의 객체 선언 방식을 따르는 것이 아니라 보통의 객체 class를 정의하고 사용하는 방법으로 관리한다.

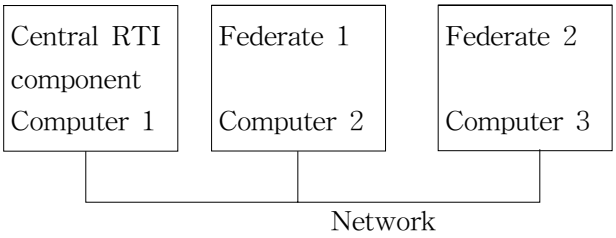


(그림 5) object management module의 구성

기존 HLA의 구현에서 AttributeHandleValueMap을 사용하여 해당 객체의 속성을 일괄적으로 전송하던 방식의 통신 부하를 줄이기 위해서 위에서 설명한 속성의 식별자(id)를 이용한다. (id, value) 형태의 interaction만을 사용하게 함으로써 실제 수정이 이루어지는 속성만을 전달하게 된다.

5. 시뮬레이션

본 연구에서는 기존의 Object Management 서비스에서 제공하는 객체 관리방법을 이용한 시뮬레이션(A)과 (그림 5)의 object management module을 추가한 시뮬레이션(B) 2개를 구현하였다. 각 시뮬레이션은 (그림 6)과 같이 총 3개의 노드로 구성되어 있다.



(그림 6) 시뮬레이션의 구성

Central RTI component는 전체 federation의 모니터링을 수행한다. federate 1, 2는 시뮬레이션 (A, B)의 방법으로 2가지 형태로 구현되어지며, 실제적인 시뮬레이션의 수행을 담당한다.

시뮬레이션(A, B)은 모두 (그림 7)의 4개의 속성을

가지는 MyObjec 객체를 사용한다. 단, 시뮬레이션(A)은 기존의 객체 정의 절차를 따라 객체를 생성하고 UpdateAttributeValues를 이용해서 4개의 속성을 상대 federate들에 전달한다.

```
class MyObject {
int       texture; //사용 texture 결정
int       x_point; //객체의 x 좌표
int       y_point; //객체의 y 좌표
int       live;    //객체의 상태 결정
...
}
```

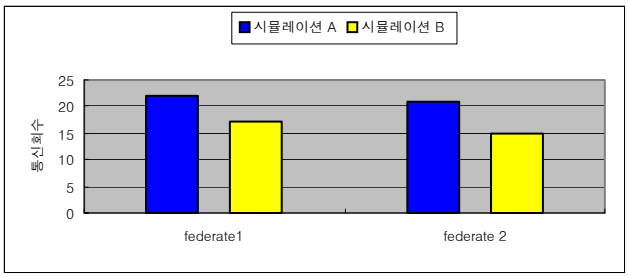
(그림 7) 사용 객체

시뮬레이션(B)은 object management module 방식을 사용하고, (id, value) 형태의 interaction을 통하여 속성을 전달한다.

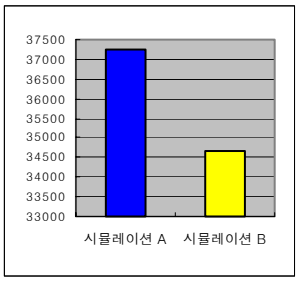
6. 시뮬레이션 결과 및 분석

초기화 비용을 산출하기 위해 Central RTI component를 실행시키고 federate 1을 federation에 참가시킨다. 이후 federate 2를 실행시켜 federation에 참가한다. 이때까지 발생한 총 데이터양과 통신 회수를 시뮬레이션(A, B)에서 각각 구한다.

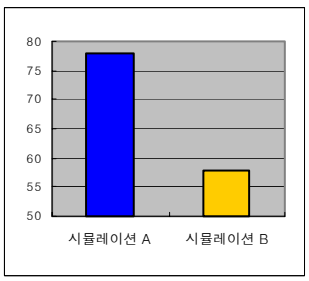
이후 시뮬레이션(A, B)에 동일한 수의 이벤트를 발생시키고 이때 발생되어지는 federate 1, 2의 각각의 데이터양의 평균을 구하여 1회 통신 데이터양을 구한다. 그 결과는 아래와 같다.



(그림 9) 초기화 통신 회수



(그림 10) 초기화 데이터 발생 양



(그림 11) 1회 통신 데이터 발생 양

object management module을 사용한 시뮬레이션 B에서 초기화 데이터양, 통신회수, 1회 통신 데이터양 모든 면에서 성능이 향상됨을 확인 할 수 있다.

초기화 통신 회수와 데이터양의 감소는 객체 정의 부분의 축소에 따른 결과이고, 1회 통신 데이터양이 78byte에서 58byte로 감소한 점은 불필요한 속성을 제외하고 전송하였기 때문이다.

7. 결론 및 향후 연구

본 논문에서는 HLA의 기반의 시뮬레이션에서 객체 관리의 성능 향상에 대하여 분석하였다. 새롭게 제안한 객체 관리 방법은 객체에 대한 사전 정의 과정을 생략할 수 있으며, 해당 객체의 속성을 불필요한 속성을 제외하고 전송하기 때문에 통신 부하를 낮출 수 있었다. 이러한 결과는 HLA를 기반으로 하는 규모가 큰 시뮬레이션의 통신 부하를 더욱 효과적으로 감소시키는 것이 가능함을 보여준다.

앞으로 이 문제를 Data Distribute Management등의 서비스와 연계하여 HLA 기반의 분산 시뮬레이션의 성능 향상을 연구 할 계획이다.

<<Acknowledgement>>

본 논문은 산업자원부 한국산업기술평가원 지정 한국항공대학교 부설 인터넷정보검색 연구센터의 지원에 의함.

참고문헌

[1] Dr. Judith S.Dahmann "High Level Architecture for Simulation" Distributed Interactive Simulation and Real Time Applications, 1997., First International Workshop on , 9-10 Jan. 1997

[2] Dr. Frederick Kuhl, Dr. Richard Weatherly, Dr. Judith Dahmann "Creating Computer Simulation Systems" Prentice Hall 1999

[3] IEEE 1516 "Standard for Modeling and Simulation High Level Architecture - Framework and Rules" IEEE 2000

[4] IEEE 1516.2 "Standard for Modeling and Simulation High Level Architecture - Federate Interface Specification" IEEE 2000

[5] IEEE 1516.1 "Standard for Modeling and Simulation High Level Architecture - Object Model Template Specification" IEEE 2000