DEVSim-HLA: DEVS 형식론과 High Level Architecture에 기반을 둔 이 기종 시뮬레이션 환경

김 용 재(), 김 탁 곤
한국과학기술원 전기 및 전자공학과

A Heterogeneous Simulation Environment Based on DEVS Formalism and High Level Architecture

Yong Jae Kim and Tag Gon Kim
Dept. of EE., KAIST

본 논문에서는 DEVS 형식론과 High Level Architecture에 기반을 둔 이 기종 시뮬레이션 환경의 구축에 대해 기술한다. DEVS 형식론은 여러 가지 방법으로 기술된 모델들을 동일한 형식론으로 간주하기 위해 사용되었다. 즉, 기존의 이산사건 모델링을 위한 세 가지 세계관(world view)으로 기술된 시뮬레이션 모델들을 DEVS 형식론으로의 변환을 통해 전체적으로는 DEVS 형식론만을 사용한 것과 동일한 형태로 표현되도록 하였다. High Level Architecture는 시뮬레이션 수행시의 상호 연동성을 보장하기 위해 사용되었다. 이때, DEVS 형식론과 High Level Architecture에서의 시뮬레이션 시간 진행 방법이 다르기 때문에 이의 해결을 위해 Synchronizer, EOS 방법을 제안하였다.

1. 서론

병렬/분산 시뮬레이션은 주어진 시뮬레이션 모델을 여러 개의 노드에 분산 실행시킴으로써 전체적인 성능향상을 얻기 위한 방법이다. 이때, 각 노드는 동일한 시뮬레이션 환경을 사용한다. 그러나, 시스템이 복잡해질수록 단일 모델 기술 방법으로는 시스템의 여러 가지 특성을 효율적으로 나타내기 어려워지므로, 여러 가지 방법으로 기술된 모델들이 필요하다. 이종 모델들의 시뮬레이션 시에는 다양한 형태의 이 기종 시뮬레이션 환경들이 포함되므로, 전체 시뮬레이션 환경을 구축할 때 성능 향상보다는 상호 연동성에 더 중점을 두게 된다.

이 기종 시뮬레이션의 필요성은 하드웨어와 소프트웨어를 동시에 개발하고자 하는 co-design에서 잘 찾아볼 수 있다[1]. [2]는 CORBA에 기반을 둔 이 기종 시뮬레이션 환경인 pLUG&SIMTM을 제안하였다. 여기서는 이종 모델들이 계층적으로 연결되었고, 시뮬레이터들간의 관계를 client-server 형태로 간주하였다.

본 논문에서는 DEVS(Discrete Event Systems Specification) 형식론과 HLA(High Level Architecture)에 기반을 둔 이 기종 시뮬레이션 환경의 구축에 대해 기술한다. DEVS 형식론은 여러 가지 방법으로 기술된 모델들을 통합된 형식론으로 간주하기 위해 사용되었다. 즉, 기존의 이산사건 모델링을 위한 세 가지 세계관(world view)으로 기술된 시뮬레이션 모델들을 DEVS 형식론으로

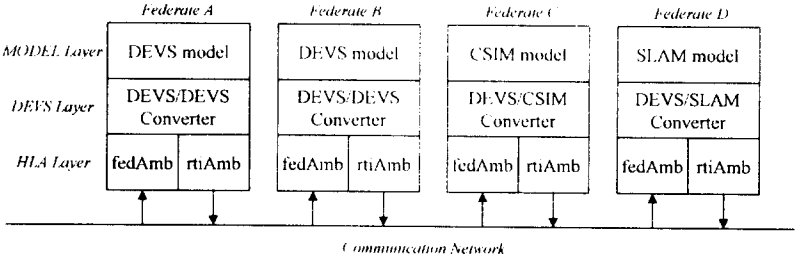

그림 1 이 기종 시뮬레이션 환경

기술된 모델과 같은 의미(semantics)로 변환함으로써, 전체적으로는 DEVS 형식론만을 사용한 것과 동일한 형태로 표현되도록 하였다. High Level Architecture는 시뮬레이션 수행시의 상호 연동성을 보장하기 위해 사용되었다.

2. 제안된 이 기종 시뮬레이션 환경

2.1 시스템 구성

제안된 이 기종 시뮬레이션 환경은 크게 모델 계층, DEVS 계층, 그리고 HLA 계층으로 이루어진다 (그림 1). 모델 계층에서는 복잡한 대형 시스템을 여러 가지 형태의 모델 기술 방법들로 나타낸다. 예를 들면, DEVS, CSIM, SLAM등의 모델 기술 방법들을 사용할 수 있다.

DEVS 계층에서는 모델 계층에서 기술된 여러 가지 모델들을 DEVS 형식론의 공통의 형태로 표현한다. 이때, DEVS 모델을 제외한 다른 형태의 모델들의 경우에는 모델 해석 방법, 즉, 시뮬레이션 프로토콜이 다르기 때문에 프로토콜 변환기가 필요하다. DEVS/DEVS 변환기는 DEVS 모델사이의 변환을 의미하고, DEVS/CSIM 변환기는 DEVS 모델과 CSIM 모델사이의 변환을 의미한다.

마지막으로, HLA 계층에서는 분산된 여러 시뮬레이션 노드들을 동일한 시간 관리 및 데이터 전달 방법으로 통합한다. 이때, DEVS 계층과의 연결 시에 시간 전진 방법이 다르기 때문에 Synchronizer, EOS 방법을 사용한다.

2.2 DEVS 형식론

이산사건 시스템의 기술 방법인 DEVS 형식론은 제조 시스템, 통신 시스템, 컴퓨터 소프트웨어 및 하드웨어 등의 여러 가지 시스템들을 효과적으로 나타낼 수 있다.

DEVS 형식론은 크게 atomic 모델과 coupled 모델의 두 가지로 모델들을 분류한다. Atomic 모델은 시스템의 동적 특성을 나타내기 위해 사용되며, 시스템의 입력 사건 집합, 출력 사건 집합, 상태 변수 집합, 상태 천이 함수, 출력 함수, 그리고 시간 전진 함수로 구성된다. Coupled 모델은 시스템의 계층적 구성을 나타내기 위해 사용되며, 구성 요소 집합, 연결 함수 등으로 구성된다.

DEVS 형식론은 입력 사건과 출력 사건이 구분이 되고, 상태 천이 함수가 외부 사건에 의한 것과 내부 사건에 의한 것의 두 가지 형태가 있으며, 시간 전진 함수로 다양한 형태의 시간을 포함하는 이산 사건 시스템을 기술할 수 있으므로, FSM이나 Petri Net등의 모델링 능력보다 우수하다. 이에, 본 논문에서는 DEVS 형식론을 사용하여 여러 가지 모델들을 공통으로 표현하는데 사용하고자 한다.

2.3 High Level Architecture

HLA는 미 국방성에서 제안한 시뮬레이션 구조로써, 여러 가지 형태의 시뮬레이터들이 상호 운용할 수 있는 기반을 제공한다. HLA는 크게 Federation Rules, Interface Specification, 그리고 Object Model Template(OMT)의 세 가지로 정의된다[3][4][5].

Federation Rules는 federation과 각 federate들이 만족해야 할 규칙을 정의하고, Interface Specification은 federation과 federate가 어떻게 상호 작용을 하는지에 대해 기술한다. OMT는 시뮬레이션 객체 정보를 표현하는 공통의 방법을 제공한다.

본 논문에서 HLA를 하부구조로 사용하는 이유는, 기존의 PVM, MPI, CORBA등의 분산 통신 환경에 비해서 HLA가 이 기종 분산 시뮬레이션을 위해 여러 가지 기능을 제공해 주기 때문이다. 예를 들면, Interface Specification의 시간 관리는 보수적인(conservative) 시간 동기 방법을 기본적으로 제공한다. 또한, federate들이 interaction을 상호 전달할 때 time stamp ordering 방법 등을 제공한다.

3. 시뮬레이션 프로토콜 변환

3.1 이산사건 모델링의 세 가지 세계관

일반적으로 이산사건 시스템을 기술할 때 크게 event scheduling, activity scanning, 그리고 process interaction의 세 가지 방법을 사용한다. Event scheduling은 각 event가 발생했을 때 시스템이 어떻게 변하는지를 기술하는 방법이고, activity scanning은 event가 발생했을 때 어떤 activity가 시작되거나 끝나는지를 기술하는 방법이다. 마지막으로, process interaction은 도착 event부터 출발 event까지 activity의 진행에 대해 기술한다[6].

3.2 DEVS 시뮬레이션 프로토콜

DEVS 형식론은 위의 세 가지 세계관을 통합적으로 표현할 수 있는 시스템 이론적 모델링 기법이다[7].

DEVS 모델의 시뮬레이션을 위해 STAR, X, Y, DONE의

네 가지 메시지가 사용된다. 그림 2는 coupled 모델 PEL이 두 개의 atomic 모델인 BUFF와 PROC로 구성되었을 때, 각각에 해당하는 시뮬레이터인 C:PEL, S:BUFF, 그리고 S:PROC가 계층적으로 연결된 모습이다.

네 가지 DEVS 메시지의 의미는 다음과 같다. STAR 메시지는 예정된 시간이 되었음을 나타낸다. 이때, 출력이 발생하면 Y 메시지가 생성되어 상위 시뮬레이터를 통해 해당 시뮬레이터에 X 메시지로 전달, 처리된다. STAR 메시지나 X 메시지의 수행을 마친 후에는 다음의 예정 시간을 구하여 DONE 메시지로 상위 시뮬레이터에게 전달한다.

3.3 시뮬레이션 프로토콜 변환

Event scheduling, activity scanning, 그리고 process interaction 방식의 시뮬레이션 수행은 각자의 방법(프로토콜)을 사용한다. 이러한 시뮬레이션 방법들은 DEVS 시뮬레이션 프로토콜과는 다르므로 시뮬레이션 프로토콜 변환을 해주어야 한다.

본 논문에서는 client/server 형태의 모델에 대한 프로토콜 변환 방법을 제안한다. 여기서, server 모델은 외부 client 모델의 서비스 요청을 수행하여 결과를 다시 client 모델에게로 전달해주는 형태를 의미한다.

그림 3에 제안된 프로토콜 변환방법을 나타내었다. 여기서, SIM SERVER MODEL은 DEVS 시뮬레이션 모델이 아닌 다른 임의의 server 모델을 의미한다. SIM SERVER MODEL이 프로토콜 변환기를 통해서 DEVS server 모델

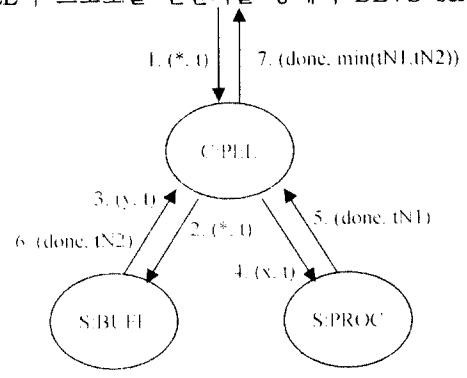

그림 2 DEVS 시뮬레이션 프로토콜

과 동일한 형태를 갖게 된다.

프로토콜 변환 방법은 다음과 같다. 먼저, 정상적인 DEVS 모델의 경우, 외부에서 X 메시지를 받으면 다음 사건 수행 예정 시간을 갖는 DONE 메시지를 보내주어야 한다. 그리고, 예정된 시간이 되면 STAR 메시지를 받고 Y 메시지를 생성한다. 그러나, 프로토콜 변환기에서는 X 메시지를 받으면 이 메시지의 실제 수행은 SIM SERVER MODEL에서 이루어지므로 예정 시간을 미리 알 수 없기 때문에, DONE 메시지를 바로 보내지 않는다. 이때, 외부에서 받은 X 메시지는 프로토콜 변환기를 통해 SIM SERVER MODEL에게 전달되고, SIM SERVER MODEL은 수행 결과를 수행 시간과 함께 프로토콜 변환기에 전달한다. 프로토콜 변환기는 이제 수행 시간을 알았으므로 DONE 메시지를 보내고, 수행 결과는 상태 변수로 저장한다. 예정된 시간이 되면, 프로토콜 변환기는 STAR 메시지를 받게 되고 저장해 놓은 수행 결과를 Y 메시지로 만들어 보낸다.

이 방법을 통해 SIM SERVER MODEL이 DEVS SERVER MODEL과 동일한 형태로 표현되었다. 즉, 서로 다른 방법으로 기술된 시뮬레이션 모델들을 한 시뮬레이션에서 동시에 실행시킬 수 있게 되었다.

4. 시뮬레이션 환경 구현

본 논문에서는 DEVS 모델의 분산 시뮬레이션 환경인 D-DEVSim++[8]와 process interaction 방식의 시뮬레이션 환경인 CSIM을 이용하였다. HLA 계층으로는 DMSO에서 구현한 RTI version 1.0.3을 사용하였다.

4.1 DEVS 시뮬레이션 프로토콜의 구현

DEVS 시뮬레이션 프로토콜을 구현하기 위해 먼저, 네 가지 메시지를 정의해야 한다. 이때, 메시지는 RTI의 Interaction으로 다음과 같이 정의한다.

(1) STAR : time, src, dest
(2) X : time, src, dest, port, value
(3) Y : time, src, dest, port, value
(4) DONE : time, src, dest, tN

DEVS/DEVS converter는 받은 메시지를 변환 없이 곧바

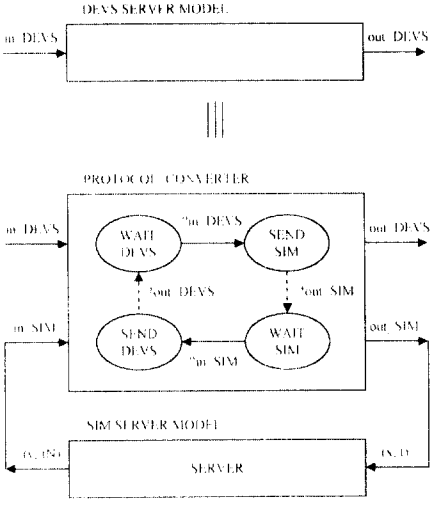

그림 3 프로토콜 변환기

로 전달해주면 되므로 간단하게 구현된다.

DEVS/CSIM converter의 경우 외부에서 X 메시지를 받으면, 이 메시지를 처리할 CSIM 프로세스를 새로 생성하고 이 프로세스가 메시지를 처리하는데 소요되는 시간과 처리 결과를 구하면 된다. 실제로는, 성능 개선을 위해

STAR와 DONE 메시지는 각 노드에서 처리하게 되므로 외부로는 Y 메시지만 전달해주면 된다.

4.2 시간 진행 방법

본 논문에서 DEVS 모델, CSIM 모델 등은 각각 독립적인 federate에서 수행된다. 이때, federate들이 time stamp order로 메시지를 주고받기 위해서 각 federate의 시간 진행 방법으로 logically time synchronized 방법을 사용한다. DEVS/DEVS converter의 시뮬레이션 main routine을 그림 4에 나타내었다.

4.3 시뮬레이션의 시작 및 종료

PVM/MPI등을 이용한 기존의 병렬/분산 시뮬레이션 시스템들에서는 fork/join 형태의 프로세스 생성방법을 사용하였다. 본 논문에서는 이 기종 시뮬레이션 환경이므로 그러한 fork/join 형태의 프로세스 생성은 어렵고, 각 federate가 개별적으로 federation에 join하여야 한다. 이때, 뒤늦게 참여하는 federate의 federate time은 이미 참여했던 federate들에 의해 진행된 시간이 된다. 예를 들어, 현재 federation time, tF = 10 이라 할 때, 뒤늦게 참여한 federate는 tF = 10부터 시작하게 된다.

이것은 federation에 참여한 각 federate의 시뮬레이션 시작 시점을 서로 다르게 설정하는 것이 되므로, war game과 같은 실시간 시뮬레이션의 경우에는 별 문제가 없겠지만, 본 논문에서 수행하고자 하는 이산사건 시스템 시뮬레이션에는 중대한 오류가 된다.

이 문제는 synchronizer federate로 해결된다. synchronizer federate는 다른 모든 federate가 join 할 때까지 time advance request를 하지 않고 기다리는 federate이다.

시뮬레이션의 종료는 모든 federate가 자신의 메시지를 다 처리했고, 다른 federate들로부터 받을 메시지가 없으면 이루어진다. 실제로는, 모든 federate가 infinity로의 time advance request를 하게 된다. 이때, federation time은 미리 정해진 lookahead 값 만큼씩 증가하고, federation time이 infinity가 되면 시뮬레이션이 종료된다. 그러나, infinity는 2000000000, lookahead는 1e-09의 값으로 정해지므로, 실제 시뮬레이션이 종료되기까지는 많은 시간이 소요된다. 시뮬레이션의 종료를 분명히 하기 위해, EOS 방법을 사용한다. 각 federate는 infinity의 time advance request를 할 때마다 다른 federate들에게 EOS (End-Of-Simulation) 메시지를 보낸다. 모든 federate로부터 EOS 메시지를 받으면, 시뮬레이션은 종료된다.

5. 실험 결과

제안된 시뮬레이션 환경을 검증하기 위해 그림 5의 BUF-PROC 모델에 대한 실험을 하였다. 여기서, GENR, TRANSD, BUFF는 DEVS 모델이고, PROC은 CSIM

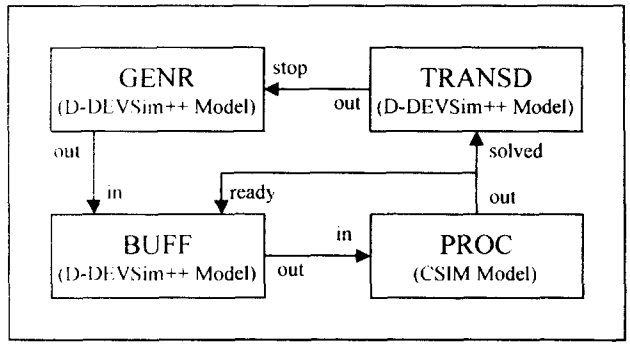

그림 5 BUF-PROC Model

server 모델이다. 시뮬레이션은 2개의 federate에서 수행되었고, BUFF에서 보낸 X 메시지를 DEVS/CSIM converter를 통해 CSIM의 PROC 프로세스가 처리하고, 계산된 결과를 예정된 시간에 Y 메시지로 전달한다.

```
while(1) {
  set reqTime minimum of minimumtN and minimum time of inputmsgQ;
  if (reqTime is less than current federate time) {
    check if there is a message to be received;
    process the first message of inputmsgQ;
  }
  else {
    federate time advance toreqTime;
    receive all TSO messages withts is less than and equals toreqTime;
    if (current federate time is infinity)
      simulation terminated;
    if (current federate time isreqTime)
      process the first message of inputmsgQ;
    else
      just skip;
  }
}
```

그림 4 Event-driven Simulation Main Routine

6. 결론

본 논문에서는 복잡한 대형 시스템을 효과적으로 모델링/시뮬레이션 할 수 있는 이 기종 시뮬레이션 환경을 제안하였다. 이를 위해, 프로토콜 변환 방법을 제안하여 세 가지 세계관으로 기술된 모델들을 DEVS 모델과 동일한 방법으로 시뮬레이션 할 수 있었다.

제안된 프로토콜 변환 방법은 client/server 모델로 제한된 경우에만 적용되었다. 앞으로, 일반적인 모델에 대한 변환 방법에 대해 연구할 계획이다.

참고 문헌

[1] "Special Issue on Hardware/Software Co-Design", Proceedings of the IEEE, vol. 85, no. 3, 1997

[2] Ernst, J., Prasad, C., and Thurston, G., "Cosimulation With Heterogeneous Simulation Algorithms Using Distributed Objects", SCSC'97, 1997, pp. 579-584

[3] DMSO 1997, "HLA Rules Version 1.2, August 13, 1997."

[4] DMSO 1997, "HLA Interface Specification Version 1.2, August 13, 1997"

[5] DMSO 1997, "HLA Object Model Template Version 1.2, August 13, 1997"

[6] Fishman, Concepts and Methods in Discrete Event Digital Simulation, John Wiley & Sons, 1973

[7] Zeigler, B., Theory of Modelling and Simulation, Robert E. Krieger Publishing Co., Florida, 1976

[8] Kim., K. H., Seong, Y. R., Kim, T. G., and Park, K. H, "Distributed Simulation of Hierarchical DEVS Models: Hierarchical Scheduling Locally and Time Warp Globally", Transactions of the SCS, vol. 13, no. 3, 1996, pp. 135-154