하이브리드 시스템 모델링 및 시뮬레이션 - 제2부: 시뮬레이터 연동 환경*

Hybrid Systems Modeling and Simulation - Part II: Interoperable Simulation Environment

임성용**, 김탁곤***
Seong Yong Lim and Tag Gon Kim

Abstract

Hybrid simulation may employ different types of simulators based on which models in different system types are developed. The simulation requires simulation time synchronization and data exchange between such simulators, which is called simulators interoperation. This paper develops such interoperable simulation environments for modeling and simulation of hybrid systems whose components consist of continuous and discrete event systems. The environments, one for centerized and the other for distributed, support interoperation between a discrete event simulator of DEVSim++ and a continuous simulator of MATLAB. The centerized environment, HDEVSim++, is developed by extending the existing DEVSim++ environment; the distributed environment, HDEVSimHLA, is developed using the HLA/RTI library. Verification of both environments is made and performance comparison between the two using a simple example is presented.

Key Words: Hybrid Systems, Interoperable Simulation, HDEVSim++, HDEVSimHLA, HLA/RTI

* 본 논문은 2000년 삼성 휴먼테크 논문 경시대회에서 발표한(금상 수상) 내용을 보완한 것 중 후반부(2부)임.
** 한국전자통신연구원 네트워크기술연구소
*** 한국과학기술원 전자전산학과

1. 서론

하이브리드 시스템은 두 가지 이상의 다른 형태를 갖는 부 시스템으로 구성되어 있다. 본 연구에서는 연속 시스템과 이산사건 시스템이 혼재하는 시스템을 하이브리드 시스템이라고 정의하였다. 본 연구의 1 부에서는 이러한 하이브리드 시스템의 모델링 형식론이 제안되었고, 이들 모델을 시뮬레이션하는 개념에 대하여 다루바 있다. 제안된 모델링 형식론은 기존의 연속 시스템 모델링 형식론인 미분방정식과 이산사건 시스템 형식론인 DEVS를 그대로 사용 할 수 있었다. 이렇게 함으로써 기존에 개발된 연속 모델과 이산사건 모델을 재사용이 가능하였다. 본 연구의 2 부인 본 논문에서는 연속 모델을 시뮬레이션하는 시뮬레이터와 이산사건 모델을 시뮬레이션 하는 시뮬레이터를 연동 시켜서 전체 하이브리드 시스템을 시뮬레이션하는 환경 개발을 다루고 자 한다.

본 연구에서는 연속 모델 시뮬레이터로 MATLAB[2]를, 이산사건 모델의 시뮬레이터로 DEVSim++[3]를 사용하였다. MATLAB[2] 은 공학분야에서 가장 많이 쓰이는 시뮬레이터 중 하나로서 다양한 응용분야(예: 전자공학, 기계공학등)에 걸쳐서 연속모델 라이브러리를 제공 한다. DEVSim++[3]는 DEVS 모델 개발과 이의 시뮬레이션을 지원하는 추상 시뮬레이션 알고리즘을 C++ 언어로 구현한 것이다. DEVSim++는 이산사건 시스템을 객체지향적인 관점에서 계층적으로 모듈화하여 모델링 할 수 있는 환경이다.

MATLAB 과 DEVSim++를 연동하기 위해서는 두 시뮬레이터의 시뮬레이션 진행 시간을 동기화하고, 두 시뮬레이터 사이의 데이터 교환이 가능해야 한다. 본 연구의 제 1 부에서는 이러한 시간 동기화를 위하여 연속 시뮬레이터의 시뮬레이션 과정에서 pre-simulation 개념이 고안되었고, 데이터 교환을 위하여 A/E(Analog-to-Event) 및 E/A(Event-to-Analog) 변환기가 제안되었다.

시뮬레이터 연동에 관한 연구는 최근 들어 미 국방성에서 각 군에서 개발한 여러 다른 기종의 군사용 시뮬레이터들을 함께 연동시켜 운용할 필요성이 대부되면서 시작되었다. 이 경우, 지역적으로 떨어져서 운용되고 있는 이 기종의 군사용 시뮬레이터들을 네트워크 환경에 묶어서 운용함으로써 합동 군사훈련, War Game 등에 효과적으로 활용 할 수 있다. 미 국방성에서는 이러한 연동을 위한 상위레벨 개념인 HLA (High Level Architecture)를 제안하고 이를 구현한 시뮬레이터 연동용 인터페이스 라이브러리에 해당하는 RTI(Run Time Infrastructure)를 표준화 한 바 있다[7]. 그러나, 이 경우의 연동은 이 기종 이산사건 시뮬레이터 간의 연동에 중점을 두고 있으며, 이산사건 시뮬레이터와 연속 시뮬레이터 연동에 관한 연구는 보고된 바 없다.

시뮬레이터 연동은 기존의 시뮬레이터에서 사용되고 있는 모델은 수정 없이 재사용 할 수 있어야 한다. 그러나 연동을 가능하게 하기 위해서는 모델을 수행하는 시뮬레이터 알고리즘을 수정하는 경우와 전혀 수정하지 않는 경우로 나누어서 생각할 수 있다. 본 논문에서는 이들 두 가지 경우를 모두 구현하고 이들의 성능을 간단히 비교하였다. 전자의 경우, 기존의 DEVSim++를 확장하여 MATLAB과 연동이 가능한 HDEVSim++ 환경을 개발하였고, 후자의 경우는 HLA/RTI를 사용하여 DEVSim++와 MATLAB을 연동시킨 HDEVSimHLA 환경을 개발하였다.

본 논문의 구성은 아래와 같다. 2절에서는 연동 시뮬레이터와 관련된 시뮬레이션 배경지식을 설명한다. 3절에서는 이산 사건 시뮬레이터와 연속 시뮬레이터가 하나로 통합된 형태인 통합형 시뮬레이션 환경 HDEVSim++을 설명하고, 4절에서는 실시간 분산 시뮬레이션을 위한 구조인 HLA를 적용한 분산형 연동 시뮬레이션 환경 HDEVSimHLA을 제안한다. 5절에서는 이동 로봇 제어 시스템 예제를 통해서 제안하는 방법론의 적용 예를 확인하고, 마지막으로 6절에서는 결론을 맺고 추후 과제를 제시한다.

2. 시뮬레이터 구현 관련 배경 지식

2.1 하이브리드 시스템 기술 형식론: HDEVS

본 연구의 1에서 제안된 HDEVS (Hybrid Discrete Event Systems Specification) 형식론은 미분방정식으로 표현된 연속 모델들과 DEVS 로 표현된 이산사건 모델을 재 사용하여 하이브리드 시스템을 모델링 할 수 있었다. HDEVS 형식론은 Zeigler 가 제안한 DEVS을 확장한 것이다. 집합론에 근거한 DEVS 형식론은 이산 사건 시스템을 모듈별로 나누고 이를 계층적인 연결로 모델링 할 수 있는 수학적인 기반을 제공한다. HDEVS 형식론에서 연속 시스템의 입출력 특성은 다중 입출력 미분방정식으로 기술한다. 또한, 이종 시스템간의 입출력 교환을 위해서 이산 사건과 연속적인 입출력 사이의 관계를 기술하는 Conversion Functions을 적용하여 시간과 정보의 상호 전달을 가능하게 한다.

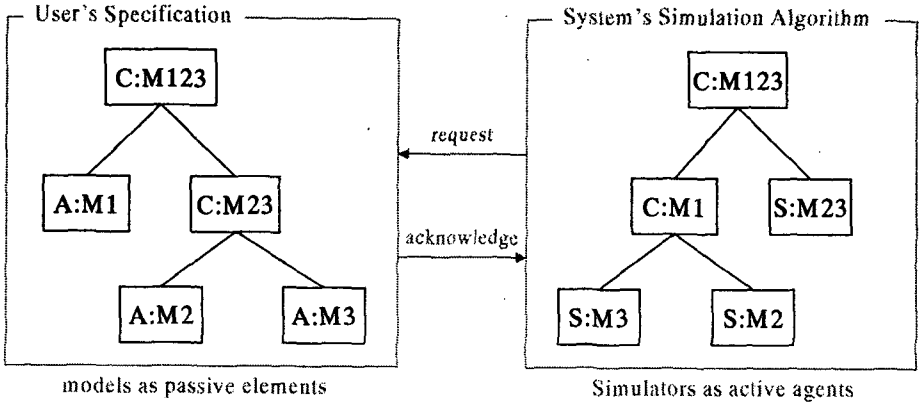
2.2 DEVS 모델 시뮬레이션: 추상 시뮬레이터

추상 시뮬레이터는[4] DEVS 형식론에 의해 명세된 모델을 해석하여 시뮬레이션 하는 시뮬레이션 알고리즘이다. 추상 시뮬레이터는 DEVS 형식론의 두 가지 형태의 모델인 Atomic 모델과 Coupled 모델 각각에 대하여 다르게 정의되었다. Atomic 모델의 추상 시뮬레이터를 Simulator, 그리고 Coupled 모델의 추상 시뮬레이터를 Coordinator 라고 정의하였다.

Simulator 는 Atomic 모델이 언제(시뮬레이션 시각), 무엇(상태천이, 출력 등)을 수행하느냐를 지시하며, Coordinator 는 Simulator들 사이의 통신과 시뮬레이션 시각 관리를 담당한다. 즉, 모델은 수동적인 명령어 집합에 해당하고 추상 시뮬레이터는 능동적인 에이전트에 해당한다고 볼 수 있다. DEVS 시뮬레이션 환경의 설계 개념은 이러한 수동적 모델과 능동적 추상 시뮬레이터를 완전히 분리하는데 있다. 이렇게 함으로써 사용자들은 DEVS 형식론에 의거하여 이산 사건 모델 개발에만 전념 할 수 있고 개발된 모델들은 추상 시뮬레이터에 의하여 시뮬레이션 될 수 있다는 장점이 있다. <그림 1>은 이러한 추상 시뮬레이터 동작 개념을 나타내고 있다.

2.3 분산 시뮬레이션과 HLA

분산 시뮬레이션은 시뮬레이션 프로그램을 여러 개의 컴퓨터에 분산하여 수행하는 기법을 총칭해서 말한다. 이러한 기법은 전자공학, 전산학, 산업공학, 경제학 등 규모가 크고 복잡한 시스템을 시뮬레이션 할 시 순차적 컴퓨터를



<그림 1> Abstractor Simulators Concept

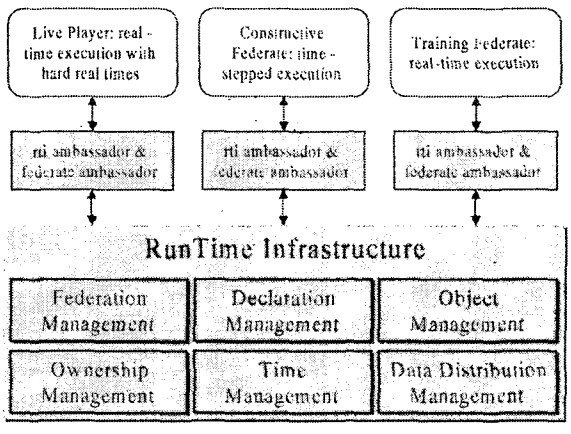
사용함으로써 발생하는 과도한 계산시간을 줄이기 위한 것이다. 그러나, 수치해석 프로그램과는 달리 이산사건 시뮬레이션은 비수치 프로그램으로 모델의 데이터 구조 등이 매우 비정형적이며, 입력 데이터에 따른 변화가 크므로 벡터프로세싱 컴퓨터에 의한 병렬 처리는 효율이 저하된다.

분산 시뮬레이션에서는 전체 시뮬레이션 프로그램이 몇 개의 프로세스(process)로 나누어진 후 이들 프로세스 그룹들이 여러 개의 프로세서(processor)에 할당되어 시뮬레이션이 진행된다[9]. 이사사건 시뮬레이션이 시간에 따라 순차적으로 스케쥴링된 사건들을 순서대로 수행되는 점을 고려할 때, 프로세서 A 내의 어떤 프로세스 a1 에서 사용되는 시뮬레이션 시각과 프로세서 B 내의 어떤 프로세스 b1 에서 사용되는 시뮬레이션 시각 사이에 동기화가 필요하게 된다.

분산 시뮬레이션에서 분산된 프로세서 내에서 실행되고 있는 다른 프로세스들 사이의 시뮬레이션 시각 동기화 기법에는 보수적 알고리즘과 낙관적 알고리즘이 있다. 보수적 알고리즘에서 각 프로세스는 모든 프로세스의 전체 시뮬레이션 시간을 확인 한 후 각자의 시뮬레이션 시간을 진행시킨다. 이렇게 함으로써 각 프로세스들의 시뮬레이션 시계(local simulation clock)가 전체 시뮬레이션 시계(global simulation clock)와 동기화된다. 반면, 낙관적 알고리즘에서는 각 프로세스들은 자기의 시뮬레이션 시계를 사용하여 다른 프로세스와 독립적으로 시뮬레이션을 진행시킨다. 그러나 이러한 독립적 시뮬레이션 결과 프로세스들의 사건 처리가 시간에 따라 순차적으로 수행되어야 하는 조건에 모순되는 경우가 발생하게 된다. 이러한 경우 프로세스는 시간의 모순이 발견되는 즉시 앞서 한 사건처리를 무효화하고 시뮬레이션 시각을 되돌려야 한다. 이러한 과정을 rollback 이라고 한다. 한 프로세스의 rollback 은 이 프로세스로부터 메시지를 받아 사건을 이미 처리한 다른 프로세스의 rollback을 야기시키고 이러한 과정이 되풀이된다. 따라서, 시뮬레이션 진행은 현재의 시각에서 과거의 시각으로 복귀되면서 그 사이에 일어난 모든 시뮬레이션은 무효화된다.

보수적 알고리즘은 구현이 간단하지만 낙관적 알고리즘에 비하여 평균적으로 시뮬레이션 시간이 더 많이 소요된다. 낙관적 알고리즘에서 시뮬레이션에 소요되는 시간은 rollback 의 횟수에 관련되며 응용 프로그램에 따라 이러한 rollback 의 횟수가 달라진다[10]. 대표적인 보수적 알고리즘으로는 Chandy-Misra 알고리즘이[5] 있고, 낙관적 시뮬레이션 알고리즘으로는 Time-warp 알고리즘[6] 이 대표적이다.

HLA[7] 는 미 국방성에서 제안한 시뮬레이터 연동을 위한 상위레벨 구조로서 다른 기종의 시뮬레이터들이 네트워크 상에서 상호 운용할 수 있는 기반을 제공한다(<그림 2> 참조). HLA는 HLA Rules, Interface Specification, 그리고 Object Model Template (OMT)의 세 가지로 정의된다. 먼저, HLA Rules는 federation에 포함되는 구성 요소들의 역할과 상호관계에 관한 10개의 전반적이고, 기본적 규칙들이다. Interface Specification은 각 federate와 RTI (Runtime Infrastructure)간의 기능적 인터페이스에 관한 규약으로 여섯 가지의 관리영역으로 나누어 기술하고 있다. RTI는 Interface Specification을 시스템 기종 및 프로그램 언어별로 Library 형태로 구현한 것이다. DEVS 방



<그림 2> Functional View of HLA

법론과 HLA/RTI 라이브러리를 이용하여 이 기종 시뮬레이터들을 연동한 분산시뮬레이션 기법[11] 및 이를 구현한 환경[12]이 보고된 바 있다.

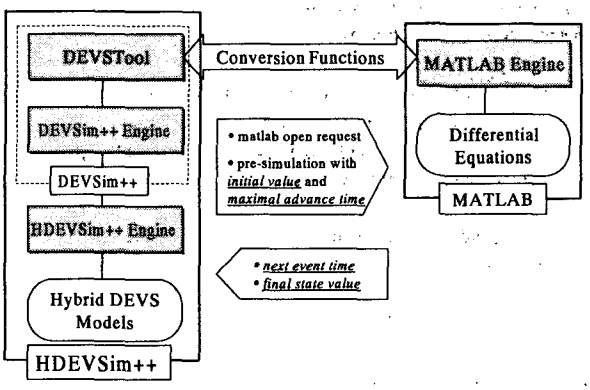
OMT는 federation을 구성하는 federate 들 간에 이루어지는 공통 데이터 영역을 구조적, 기능적으로 기술하는데 사용되어진다. OMT는 federation를 구성하는 federate 들 사이의 공유 데이터 교환 구조를 서술하는 FOM(Federation Object Model)과 시뮬레이션 시스템이 federation에 제공하는 기능을 표현하는 SOM(Simulation Object Model)로 구성된다.

3. 통합형 연동 시뮬레이션 환경: HDEVSim++

3.1 HDEVSim++의 구조

본 논문에서는 하이브리드 시스템 모델은 연속 모델과 이산사건 모델로 구성된다고 정의하였다. 이러한 하이브리드 모델의 시뮬레이션을 위해서 연속 모델의 시뮬레이터는 MATLAB, 그리고 이산사건 모델의 시뮬레이터는 DEVSim++를 사용하였다.

MATLAB 은 연속 모델 분야의 다양한 라이브러리를 제공하며 C 언어 인터페이스를 통해 외부 프로그램에서 MATLAB 시뮬레이션을 요청 할 수 있다. 반면 DEVSim++ 는 이산사건 모델 개발을 위해 C++ 클래스를 제공하며 시뮬레이션 환경이 C++ 라이브러리 형태로 제공되기 때문에C/C++ 언어를 통하여 외부 프로그램에서 DEVSim++ 시뮬레이션을 요청할 수 있다.

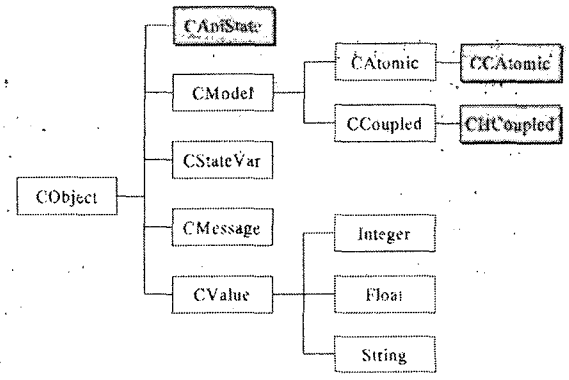


<그림 3> HDEVSim++ for Simulators Interoperation

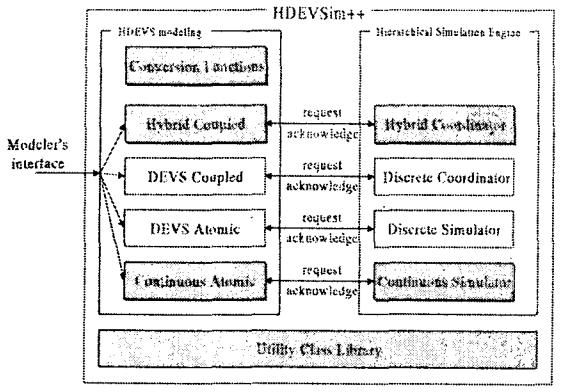
개발된 하이브리드 시뮬레이션 환경인 HDEVSim++ 는 DEVSim++ 의 추상 시뮬레이터를 확장하여 DEVSim++ 와 MATLAB 의 연동을 가능 하도록 한 것이다. 구체적으로 말하면, HDEVSim++은 MATLAB 에 열기요청과 동시에 pre-simulation 메시지인 (pre,tN) 를 보냄으로서 MATLAB 이 일정 시간동안 연속 모델을 시뮬레이션 하게 한다. MATLAB은 pre-simulation 결과로 (e, tL) 메시지를 HDEVSim++ 에 돌려주고 이 메시지를 입력으로 하여 DEVSim++ 시뮬레이션이 진행된다.

3.2 HDEVSim++ 의 구현

<그림 4>와 <그림 5>에서와 같이 HDEVSim++은 기존의 DEVSim++을 확장한 형태로 모델러는 DEVS Coupled (DEVS-CM) 모델로부터 상속받는 Hybrid Coupled Model (HCM)과 DEVS Atomic(DEVS-AM) 모델로부터 상속받는 Continuous Atomic 모델(CAM) 그리고 A/E 및 E/A 변환기는 제안된 HDEVS 형식론에 의거하여 작성한다. 동시에 CAM의 세부 명세는 MATLAB의 Simulink를 통하여 미분방정식 모델로 작성한다. 2.2 절에서 언급한 추상



<그림 4> Class Hierarchy of HDEVSim++



<그림 5> Architecture of HDEVSim++

시뮬레이터 개념에 의거하여 HCM의 추상 시뮬레이터 알고리즘은 Hybrid Coordinator(HC)로 그리고 CAM의 추상 시뮬레이터 알고리즘은 Continuous Simulator(CS: 연속시뮬레이터)로 구현된다.

3.3 연속 시뮬레이터 와 MATLAB

HDEVSim++에서 연속시뮬레이터는 상위 HC로부터 받은 pre-simulation 메시지인 (pre,tN)를 바탕으로 CAM의 시간을 진행시키는 역할을 수행한다. (pre,tN) 메시지를 받은 연속시뮬레이터는 MATLAB에서 제공하는 C 인터페이스를 사용하여 Simulink 모델의 시간을 진행시킨다. HDEVSim++에서 사용하는 인터페이스 함수들은 다음과 같다.

(1) simget(MODEL) : CAM의 실제 모델인 Simulink 모델을 시뮬레이션 하는 환경 변수를 읽는 함수. 환경 변수가 포함하는 내용은 미분방정식을 풀어나가는 step 변수, 출력 변수, 초기값, 미방해법도구. 결과는 환경 변수 'OPTIONS'.

(2) simset(OPIONS, 'NAME1', VALUE1, 'NAME2', VALUE2, ...) : 기존의 환경 변수 'OPTIONS'에 포함된 내용 중에 수정되는 값을 적용키는 함수. 역시 결과는 환경 변수 'OPTIONS'

(3) sim('model', TIMESPAN, OPTIONS, UT) : Simulink 모델인 'model'을 시간 간격인 'TIMESPAN' 동안 환경 변수 'OPTIONS'를 가지고 외부 입력 'UT'를 바탕으로 진행하는 함수. 결과는 모델 내부의 상태 변수를 가지는 상태변수 집합과, 사용자가 정의한 출력 변수 집합.

위의 기본적인 세 함수를 바탕으로 연속시뮬레이터는 CAM의 시간 수행을 <그림 6>에서와 같은 순서로 요청한다. 먼저 시간 간격 T1을 이전 시뮬레이션 종료 시간 tL과 다음 진행 시간 tN 사이로 정한다. simulink 모델이 가지는 상태 변수의 초기값을 적용하기 위해서 simset 명령을 통하여 이전 시뮬레이션 종료시 결정되어진 종료값을 새로운 초기값으로 확정한다. 다음으로 시간 간격 T1, 환경 변수 option, 그리고 외부 입력 함수인 input_func을 기반으로 sim명령을 통해 시뮬레이션을 수행한다. Pre-simulation 기간동안 연속 시뮬레이션을 진행시키면서 시뮬레이션 결과에 따른 A/E 변환을 하게 된다. A/E 변환의 한 예로서는 연속 변수의 값이 영점을 통과하는 경우 이벤트

```
Continuous Simulator
: t_L -> last simulation end time
: zc_points -> zero cross points variables from Conversion Functions

'option = simget('InitialState', [is1, is2, ... ]);'
When receive (pre,t_N) from parent
    'T1 = [t_L ,t_N];'
    'option = simset(option, 'InitialState', [s1, s2, ... ]);'
    '[T X Y1 Y2 ... ] = sim('model_name', T1, option,
                        [T1, input_func1, input_func2, ... ]);'
    if [zc_points] == 1 at time t_L'
        t_L = t_L'
        send (e, t_L) to parent
        return
    endif
    t_L = t_N
    send (NULL, t_L) to parent
End Continuous Simulator
```

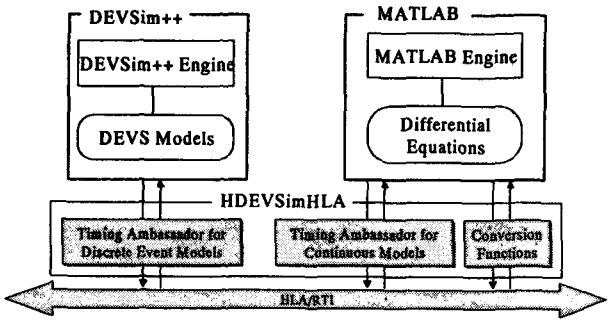
<그림 6> Interface Between CS and MATLAB

가 발생한 것으로 하는 것이다. 이 경우, 연속 시뮬레이터는 상위 HC 에 A/E 변환된 이벤트 메시지를 반환하고, 그렇지 않은 경우에는 연속 시뮬레이터는 pre-simulation에서 지정한 시뮬레이션 종료 시간까지 시뮬레이션을 진행한 후 NULL 출력을 HC 에 돌려준다.

4. 분산형 연동 시뮬레이션 구조: HDEVSimHLA

4.1 HDEVSimHLA의 구조

<그림 7>에는 <그림 3>에 나타낸 통합형 연동 시뮬레이션 기법에 비해 좀 더 진보한 연동 시뮬레이션 환경을 제안하고 있다. 진보된 환경세서는 기존에 정의한 A/E 및 E/A 변환을 정의한 Conversion Functions을 모델과 시뮬레이션 내부에서 분리하여 독립적인 함수로 구축하였다. 또한 시뮬레이터 간의 시간 동기화는 각 시뮬레이터의 내부 시간 진행 알고리즘을 그되로 유지한 채 외부의 스케쥴 대리자(Ambassador)를 통해서 시간 진행을 허락 받는 방법을 사용하였다. 이러한 개념은 기존의 도구가 가지고 있는 스케쥴 알고리즘에 독립적인 스케쥴 대리자의 적용으로 포괄적이고 자연스러운 연동 시뮬레이션 환경을 구현할 수 있게 한다.



<그림 7> HDEVSimHLA for Simulators Interoperation

4.2 HLA와 RTI에서 시뮬레이션 시간 관리

RTI(Run-Time InfraStructure)는 HLA를 구현한 라이브러리로서 6가지의 함수 집합으로 구성되며, 이 중 시간에 관련된 Time Manage -ment 함수들의 개략적인 설명은 아래와 같다.

- Time Management -

enableTimeRegulating : 각 모델들이 이 함수를 수행하게 되면 모델에서 출력되는 모든 사건들은 시간 순서에 정렬되어 발생됨을 보장하게 된다.

enableTimeConstrained : 어떤 모델이 이 함수를 수행하게 되면 다른 모델에서 자기 모델에게 메시지가 전달되는 경우 항상 시간 순서에 정렬하여 받기를 원한다는 정보를 RTI에게 알려준다.

timeAdvanceRequest : 어떤 모델이 시간을 진행하고자 하는 경우에 자기 모델에게 전달되는 예약된 사건 중에서 주어진 시간까지의 모든 사건을 전달하고 시간 진행을 요청하는 방법으로 시간 간격 시뮬레이션 (Time Stepped Simulation)에 적용되며, 전달되는 시간까지 출력되는 사건은 존재할 수 없다.

nextEventRequest : 어떤 모델이 시간을 진행하고자 하는 경우에 자기 모델에게 전달되는 예약된 사건 중에 하나의 사건만을 전달하여 사건에 의한 시뮬레이션 (Event Driven Simulation)에 적용된다.

timeAdvanceGrant : timeAdvanceRequest와 nextEventRequest의 결과로 RTI가 각 모델에게 전달하는 함수로 전달된 시간까지는 다른 사건이 전달되지 않으므로 안심하고 시간 진행을 해도 시간 순서에 어긋나지 않음을 보장해 준다.

queryMinNextEventTime : 현재 자신 모델에게 전달되는 예약된 사건들 중에서 최소 시간을 얻고자 하는데 수행된다.

4.3 DES Ambassador, CS Ambassador 와 RTI

본 절에서는 4.2 절에서 서술된 RTI의 Time Management 함수들을 사용하여 이산 사건 모델과 연속 모델의 시간 진행의 연동을 담당하는 대리자의 알고리즘을 제안한다. 먼저 모든 모델들은 초기화 과정에서 enableTime Regulating과, enableTimeConstrained 함수를 실행시켜 시간 순서에 정렬된 사건들만 주고 받을 수 있다는 사실을 RTI에 알려준다. 이 방법은 분산 시뮬레이션 분야에서 사용하는 보수적 (conservative) 시간 동기 방법과 동일한 경우이다. <그림 8>의 좌측에 나타낸 이산 사건 모델의 스케줄 대리자인 DES Ambassador의 수행 순서는 다음과 같다.

- DES Ambassador -

(1) 자신의 다음 사건 시간 (tN)을 계산하고 nextEventRequest(tN) 을 수행하여 RTI로부터 시간 진행을 허락 받을 때까지 기다리게 된다.

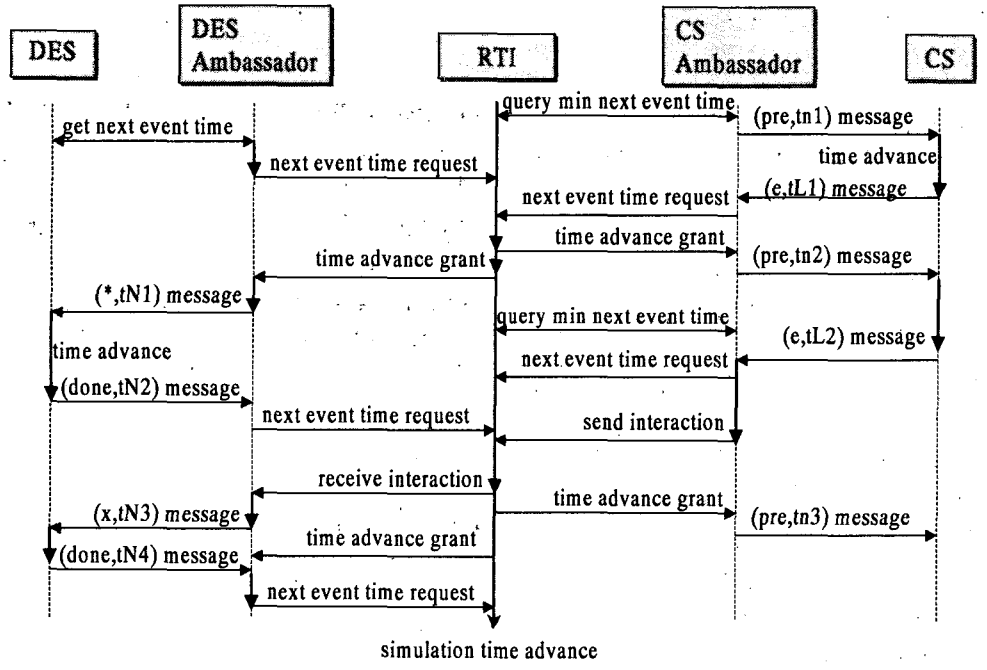
(2) RTI에 예약되어 있는 사건 중에서 tN1 이전에 사건이 존재하지 않는 다고 보증할 수 있는 경우에는 timeAdvanceGrant(tN1)이 실행되고 모델은 이산 사건 모델의 시간 진행을 위해서 (*,tN1) 메시지를 전달하게 된다.

(3) RTI로부터 tN 이전에 발생한 사건이 존재할 경우에는 사건을 받게 되고, 이 사건은 이산 사건 모델에게는 외부 상태 전이함수를 수행하기 위해서 (x,tN3) 메시지를 전달하게 되고, 시간 진행은 사건의 시간 tN3 까지 진행할 수 있다.

(4) **(2)**의 내부 상태 전이 함수가 수행되거나, 혹은 **(3)**의 외부 상태 전이 함수가 수행되든, 시간 진행의 결과와 다음 사건 시간을 계산하기 위해서 (done,tN2) 메시지를 DES Ambassador는 받게 된다.

(5) **(1)** **(4)** **(2)** or **(3)** 을 반복하여 수행한다.

<그림 8>의 우측에 나타낸 연속 모델의 스케줄 대리자인 CS Ambassador의 수행순서는 다음과 같다.



<그림 8> Sequence Diagram in Simulators Interoperation

- CS Ambassador -

(1) queryMinNextEventTime을 수행하여 자신에게 들어오기로 예약된 최소 시간 tn1을 얻을 수 있다.

(2) tn1 까지는 사건이 들어오지 않음을 보증할 수 있지만 연속 모델의 출력이 언제 발생할지는 알 수 없다. 따라서 연속 모델의 시간 진행을 명령하는 (pre,tn1) 메시지를 전달하게 된다.

(3) 연속 모델의 결과에 따라 다음 사건 시간을 의미하는 (e,tL1)을 받게 되고 연속 모델의 사건은 tL1 이전에는 발생하지 않음을 의미하는 nextEventRequest(tL1)을 수행하게 된다

(4) RTI로부터 시간 진행의 허락을 받는 timeAdvanceGrant(tn2) 을 받게 되면 tn2이후의 시간에 대한 시뮬레이션 시간 진행을 위해 (1) (2) (3) 을 반복하게 된다.

<그림 9>는 위에서 설명한 DES Ambassador 및 CS Ambassador의 수행 순서를 알고리즘으로 표현한 것이다. 이들 알고리즘을 구현한 연동 시뮬레이션 환경의 검증은 여러 가지 하이브리드 시스템의 모델링 및 시뮬레이션 결과를 통하여 이루어졌다.

5. 예제 및 비교 분석

5.1 이동 로봇 제어 시스템

5.1.1 하이브리드 시스템 모델링

<그림 10>은 장애인을 위해 휠체어를 자동으로 운전하는 이동 로봇 제어 시스템을[8]과 이것의 하이브리드 모델을 나타내고 있다. 본 절에서는 <그림 10>의 하이브리드 시스템을 HDEVS 형식론을 바탕으로 모델링하고 개발된 연동 시뮬레이션 환경을 통해서 시뮬레이션을 수행하고자 한다. <그림 10>의 하이브리드 모델에서 이산 사건 부 시스템 모델은 휠체어를 조절하는 사람 모델과 휠체어가 움직이는 위치 정보를 표현하기 위한 환경 모델로 이루어진다. 또한 연속 모델은 사람 모델의 출력을 이동 로봇으로 전달하기 위한 조이스틱 모델과 이동 로봇의 운동학 모델이 포함되는 이동 로봇 모델로 구성되어진다. 이들 모델의 간단한 설명은 아래와 같다.

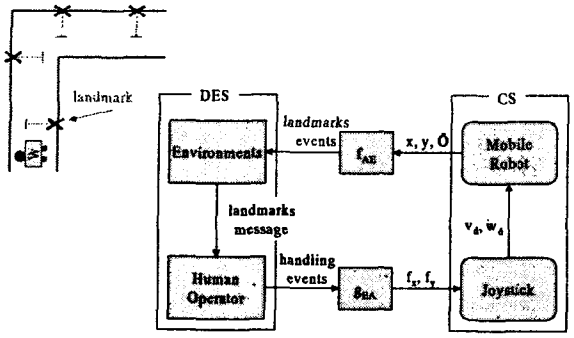
DES Ambassador:

```
main loop
    request grant for next event time from RTI
    if grant time = next event time
        send grant for next event to DES model
until end of simulation

when receive a grant message from RTI
    grant time = time of the message

when receive a next event message
                                from DES model
    next event time = time of the message
```

CS Ambassador:

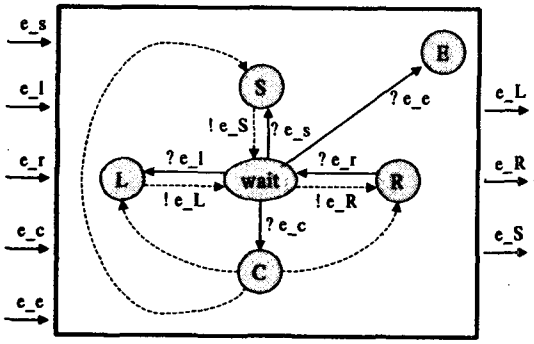
```
main loop
    request pre-simulation time from RTI
    request grant for time advance from RTI
    if grant time = next event time
        sent grant for time advance to CS model
until end of simulation

when receive a pre-simulation time from RTI
    request CS model to simulate within the time

when receive a grant message from RTI
    grant time = time of the message

when receive an event from CS model
    next event time = time of the event
```

<그림 9> Algorithms of DES and CS Ambassadors



<그림 10> Mobile Robot Control Hybrid System

사람: 입력으로 표지판의 종류를 받아서 표지판의 내용에 따라 휠체어의 움직임을 결정하는 출력을 내게 된다(<그림 11>).

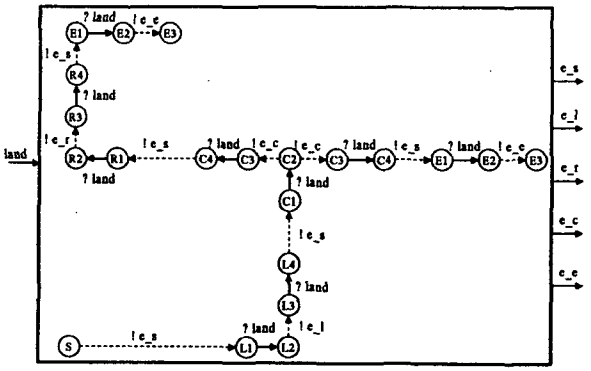


<그림 11> Human Operator Model

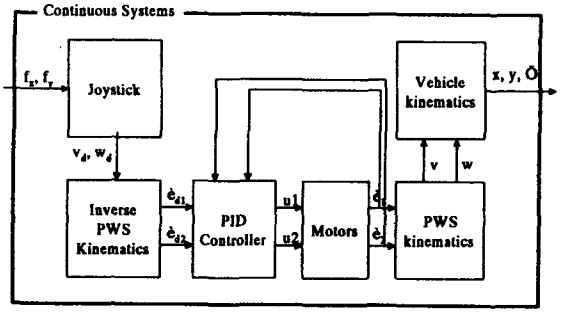
환경: 휠체어의 위치에 의해 발생되는 표지판 사건을 입력으로 받아서 지도 정보와 대조하여 방향 표지판을 출력으로 한다(<그림 12>).

조이스틱: 사람의 출력인 조이스틱을 움직이는 좌우 힘을 목표로 하는 속도와 각속도의 형태로 출력하게 된다(<그림 13> 좌위, <그림 16>).

이동로봇: 원하는 속도와 각 속도를 입력받은 PID 제어기를 이용하여 두 모터를 제어함으로서 이동 로봇의 움직임이 결정된다(<그림 13>, <그림 16>).

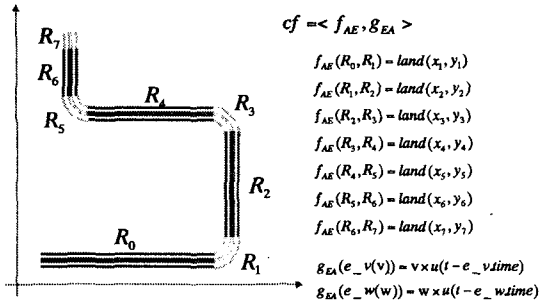


<그림 12> Environment Model

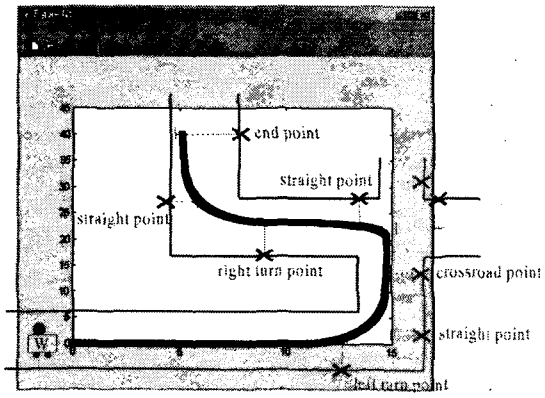


<그림 13> Block Diagram of Continuous Subsystem

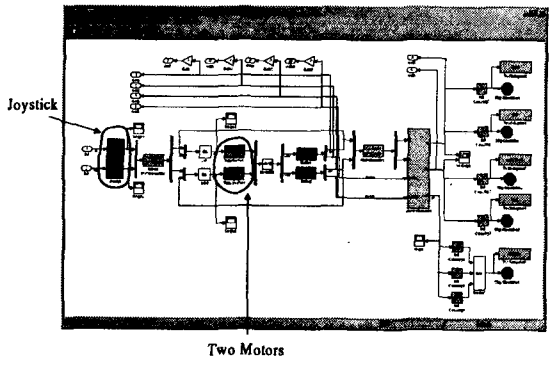
Conversion Functions: 먼저 휠체어의 위치가 변하여 표지판이 보이는 위치를 지나게 되면 A/E 변환기에는 사건이 발생하게 되고, 이것이 환경 모델에 전달된다. 사람 모델은 조이스틱을 조절하는 힘을 사건의 형태로 E/A 변환기에 전달하면E/A 변환기는 이 사건을 조이스틱의 움직임 정도, 즉 원하는 선속도와 각속도의 형태로 전환한다(<그림 14>).



<그림 14> A/E and E/A Conversion



<그림 15> Simulation Result of MRCHS



<그림 16> Continuous Subsystem Modeling in MATLAB SIMULINK

5.1.2 시뮬레이션 결과

앞 절에서 설명한 하이브리드 시스템인 이동 로봇 제어 시스템은 본 논문에서 개발한 연동 시뮬레이션 환경인 HDEVSim++와 HDEV SimHLA을 사용하여 하이브리드 모델로 구현하였다. HDEVSim++ 와 HDEVSimHLA 환경은 기능적으로 동일하며 개발된 모델과 시뮬레이션 결과도 동일하다. 다만, HDEVSim++ 에서는 이산 사건 모델과 연속 모델이 같은 프로세스에 할당되어 시뮬레이션이 수행되지만, HDEV SimHLA 에서는 이들 두 모델들이 서로 다른 프로세스에 할당되어 연동된다는 점이 다른 점이다. <그림 15>는 MATLAB에서 지원하는 FIGURE 창을 사용하여 X축과 Y축은 휠체어의 위치를 의미하고 휠체어가 지나가는 궤적에 환경모델이 가지고 있는 표지판의 위치를 겹쳐서 표현하였다. 각 모델은 아래의 순서대로 시뮬레이션을 수행한다.

(1) 사람 모델은 초기화되면서 초기방향은 우향, 행동은 직진을 휠체어에게 지령하고 휠체어는 우측으로 직진운동을 시작한다.

(2) 좌측 표지판을 지나면서 발생한 사건을 받은 환경모델은 사람 모델에게 좌측으로 방향변경 명령을 지령하고, 사람 모델은 다시 휠체어를 좌측 각 속도를 가지는 운동을 지령한다.

(3) 충분한 방향 전환이 이루어진 후 표지판은 다시 직진 운동을 가리키고, 사람 모델은 각 속도 성분을 0으로 전환하는 지령을 휠체어에 전달한다.

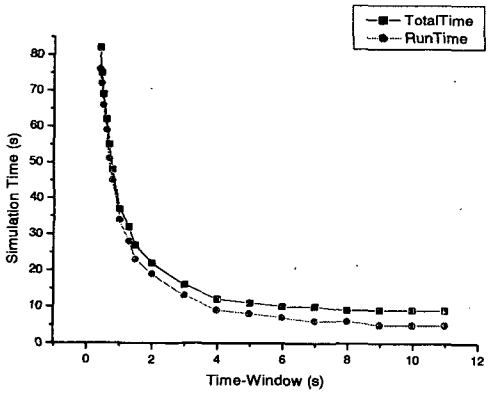
(4) 교차로 표지판을 인식한 환경 모델은 좌측 운동을 내부적으로 선택하고 사람 모델을 통해서 (2) (3)과 같은 좌측 운동을 지령한다.

(5) (2)(3)(4) 와 같은 직진, 좌측, 우측 방향 운동을 반복하다가, 휠체어가 종료 표지판을 지나가는 순간 모든 시뮬레이션을 종료한다.

5.2 각 시뮬레이터의 성능 분석

5.2.1 Pres-simulation 시간창에 따른 HDEVSim++ 성능

앞 절에서 설명한 바와 같이 하이브리드 시뮬레이션 시 연속 모델은 일정한 시간 동안 시뮬레이션을 진행하는 pre-simulation을 수행하게 된다. 이를 위해 HC(Hybrid coordinator) 는 연속 모델에게 pre-simulation 명령과 이를 수행할 최대 시간을 정해 준다. 이러한 최대 시간을 시간창(time window)이라고 부르며 시간창의 크기는 연속 시뮬레이터와 HC 의 통신횟수를 결정하며 따라서 하이브리드 시뮬레이션 수행 시간은 시간창의 크기에 따라 달라질 수 있다. 본 절에서는 이동 로봇 예제를 통해서 HDEVSim++ 시뮬레이터의 Pre-simulation 시간창 변화에 따른 시뮬레이션 수행 시간을 측정한다.

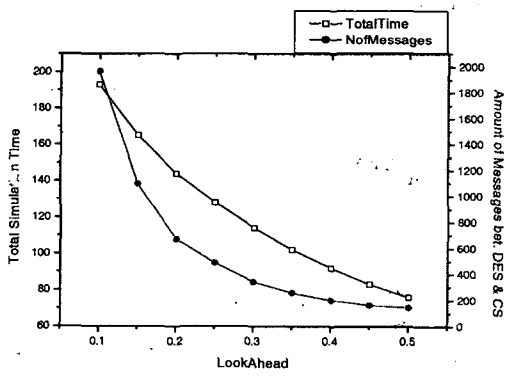


<그림 17> Simulation Time in HDEVSim++

<그림 17>에서와 같이 성능 측정은 같은 모델에 대해서 시간창을 변화시키면서 전체시뮬레이션 수행시간(<그림 17>에서 TotalTime)과, MATLAB의 로드시간을 제외한 순수 시뮬레이션 수행시간(<그림 17>에서 RunTime)을 측정하였다. 측정 결과 시간창의 값이 증가할수록 시뮬레이션 수행시간은 짧아지는 반비례 그래프를 확인할 수 있다. <그림 17>에서 알 수 있듯이 시간창이 일정한 값 이상으로 커지면 시뮬레이션 시간이 더 이상 감소하지 않는 것을 볼 수 있다. 이것은 시간창 값이 다 경과하기 전에 연속 모델의 시뮬레이션 결과 사건을 발생시키기 때문이다. 따라서 최적의 시뮬레이션 수행 속도를 위해서 pre-simulation 시간창을 연속 모델의 사건 발생 시간 간격의 최대치로 정하면 된다

5.2.2 Lookahead에 따른 HDEVSimHLA의 성능

Lookahead는 분산시뮬레이션에서 한 시뮬레이션 시각에서 진행할 수 있는 다음 시뮬레이션 시각의 최대값을 의미한다. 따라서, lookahead 값은 전체 시뮬레이션 시간에 영향을 미치게 된다. 본 절에서는 이러한 lookahead 값의 변화에 따른 시뮬레이션 수행 시간을 측정하였다.



<그림 18> Simulation Time in HDEVSimHLA

<그림 18>에서 보듯이 lookahead 의 영향은 시간창의 경우와 마찬가지로 lookahead 값과 시뮬레이션 시간은 반비례 관계에 있다. 이러한 시뮬레이션 시간 변화의 요인 중 가장 중요한 프로세스간 교환되는 메시지 수를 <그림 18>에 나타내었다. 메시지 수가 급격하게 감소하는 구간인 [0.1-0.3]에서 전체 시뮬레이션 시간은 완

만하게 감소하고 있는 것은 전체 시뮬레이션 수행시간에 비해서 네트워크 지연시간이 큰 영향을 미치지 않는 것을 알 수 있다. 이는 연속모델과 이산사건 모델을 담당하는 두 프로세스가 같은 하드웨어를 공유하고 있기 때문에 공간적 분산에 의한 시간 지연이 없는 것에 기인한다.

5.3 HDEVSim++ 와 HDEVSimHLA의 비교

시간창과 lookahead 는 시뮬레이션 진행 간격이라는 점에서 유사한 의미를 갖지만 실제로는 큰 차이가 있다. 즉, 시간창의 값은 임의의 큰 값을 가질 수 있지만 lookahead 값은 사건발생 시간 간격의 최소치 보다 작은 것이 바람직하다. 이와 같이 시간창과 lookahead 값은 똑같이 시뮬레이션 수행 시간에 반비례 하지만, 서로 상이한 개념으로 인하여 직접적인 비교는 될 수 없다. 다만, <그림 17>과 <그림 18>의 실험 결과에서 시간창과 lookahead 값이 같은 값인 경우에 대한 HDEVSim++ 와 HDEVSim HLA를 사용한 경우 시뮬레이션 시간 비교를 <표 1>에 요약하였다.

분산 환경인 HDEVSimHLA는 네트워크 통신과 시간 스케쥴링을 위해서 HLA/RTI 의 사용하는 관계로 통합된 환경인 HDEVSim++ 에 비해 시뮬레이션 시간이 길어진다. 그러나, 이러한 성능 저하에도 불구하고 HDEVSimHLA 환경은 하이브리드 모델링과 시뮬레이션 시에 기능적, 공간적인 유연성을 제공하는 장점을 가지고 있다.

하이브리드 시뮬레이션에서 시뮬레이터는 상

<표 1> Simulation Times of Two Simulators

Time window = lookahead	0.40	0.45	0.50
HDEVSim++	82sec	75sec	69sec
HDEVSimHLA	92sec	83sec	76sec

이한 시간과 정보를 가지는 시스템 사이의 연동을 지원할 수 있어야 한다. 분산형 하이브리드 시뮬레이션 환경인 HDEVSimHLA는 시간 운용의 측면에서 모듈화한 스케쥴링 기능을 제공한다. 사용자는 이산 모델과 연속 모델 사이의 정보 교환을 중심으로 설계하고, 스케쥴링 측면에서는 HLA/RTI가 내재하고 있는 스케쥴링 방법을 그대로 수용함으로써 각 시스템 내부의 시간만을 고려할 수 있으므로 시간적인 모듈화가 가능하다. 또한 공간적으로 떨어져 있는 장소에서 실제 시스템이 동작하거나, 혹은 가상의 시뮬레이션 모델을 연동 시뮬레이션 하는 경우에는 불필요한 통합 작업 없이 HLA/RTI을 사용하여 분산 시뮬레이션이 가능하기 때문에 공간적인 유연성을 제공한다고 할 수 있다.

6. 결론

본 논문에서는 HDEVS 형식론으로 기술된 하이브리드 모델에서 연속 모델과 이산사건 모델이 각각의 시뮬레이터에서 동작할 경우, 이들을 연동하여 전체 시스템을 시뮬레이션 하는 환경을 개발하였다. 연동 환경은 연속 모델을 위한 MATLAB 시뮬레이터와 이산사건 모델을 위한DEVSim++ 시뮬레이터를 이용하여 두 가지의 연동환경 - 집중형(HDEVSim++) 및 분산형(HDEVSimHLA) -을 개발하였다.

집중형 연동 환경은 검증된 DEVS 시뮬레이션 환경인 DEVSim++를 확장하여 구현하였고, 분산형 연동 환경은 DoD 표준 인터페이스인 HLA/RTI를 사용하여 구현하였다. 두 연동 시뮬레이션 환경의 동작을 검증하기 위하여 간단한 하이브리드 시스템인 이동 로봇 시스템의 하이브리드 모델링 및 시뮬레이션을 수행하였다. 아울러, 두 연동 환경의 시뮬레이션 시간을 비교하였다.

본 논문에서 개발된 시뮬레이터 연동에 의한 연속/이산사건 하이브리드 시뮬레이션은 시스템을 전체적으로 시뮬레이션 하는데 가장 효

과적이고 실용적이며 유일한 방법으로 많은 응용 분야에 이용될 것으로 사료된다.

추후 과제로는 시뮬레이션 시간 단축을 위하여 낙관적 동기화 알고리즘을 이용한 연동 시뮬레이션 기법에 대한 연구 및 환경 개발이 필요하다.

참고문헌

[1] Bernard P. Zeigler, Herbert Praehofer and Tag Gon Kim, *Theory of Modeling and Simulation 2nd edition*, Academic Press, 2000.

[2] MathWorks, *Using MATLAB Manual*, 1998, ftp:://ftp.mathworks.com.

[3] Sung-Bong Park, "DEVSim++: A Semantics Based Envirionment for Object-Oriented Modeling of Discrete Event Systems", *MS Thesis*, KAIST, 1996.

[4] Bernard P. Zeigler, *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, 1984.

[5] K. M. Chandy, and J. Misra, "Asynchronous Distributed Simulation via a Sequence of Parallel Computations", *Communication of ACM*, April, 1981.

[6] D. Jefferson and H. Sowizral, "Fast concurrent simulations using the Time Warp mechanism", *Distributed Simulation*, 15(2) : 63-39, 1985

[7] DMSO, *High Level Architecture Framework and Rules*. 2000.

[8] Chong-Hui Kim, "Implementation of Distributed Mobile Control Robot System using COTS Systems", *MS Thesis*, KAIST, 2001.Yeong R. Seong, Sung H. Jung, Tag G. Kim, and Kyu H. Park, "Parallel Simulation of Hierarchical Modular DEVS Models: A Modified Time Warp Approach," International Journal in Computer Simulation, vol. 5, no. 3, pp. 263-285, 1995.

[9] Yeong R. Seong, Tag G. Kim, and Kyu H. Park, "Mapping Modular, Hierarchical Discrete Event Models in a Hypercube Multicomputer," Simulation Theory and Practice, vol.2, no.6, pp.257-275, 1995.

[10] Ki Hyung Kim, Yeong Rak Seong, Tag G. Kim, and Kyu Ho Park, "Distributed Simulation of Hierarchical DEVS Models: Hierarchical Scheduling Locally and Time Warp Globally," Transactions of the SCS, vol. 13, no. 3, pp. 135-154, Sep. 1996.

[11] Young Jae Kim and Tag Gon Kim, "A Hetrogeneous Simulation Framework Based on The DEVS Bus and The High Level Architecture," WSC-98, pp. 421-428, Dec., 1998, Washington DC, U.S.A..

[12] Young Jae Kim, Jung H. Cho and Tag Gon Kim, "DEVS-HLA: Heterogeneous Simulation Framework Using DEVS BUS Implemented on RTI," SCSC-99, July 1999, Chicago, U.S.A.

저자소개

임성용
한국과학기술원 전기/전자공학과 학사 (1999)
한국과학기술원 전기/전자공학과 석사 (2001)
졸업 후 현재 전자통신연구원, 네트워크기술연구소에 재직하고 있으며 주 연구분야는 하이브리드 시스템, 분산시뮬레이션, 시뮬레이터 연동, 네트워크 시뮬레이션 등임.
2000년 삼성 휴먼텍 논문 경시대회에서 본 논문으로 금상 수상.

김탁곤
1988: 아리조나대학교 전자/컴퓨터공학과 공학박사
1980 - 1983: 국립수산대학(현: 부경대학교) 통신공학과 전임강사
1987 - 1989: 아리조나 환경연구소 연구 엔지니어
1989 - 1991: 캔사스 대학교, 전자전산학과, 조교수
1991 - 1998: 한국과학기술원, 전기전자공학과 조/부교수
1998 - 현재: 한국과학기술원, 전자전산학과 교수
모델링/시뮬레이션 방법론 및 환경 개발, 시뮬레이션에 기반한 시스템 분석등에 관하여 국제적으로 활발한 연구 활동을 하고 있으며 국제학술지/국제학술발표대회 논문지에 100 편 이상의 논문을 게재하였음. 시뮬레이션 모델링에 관련된 다수의 국제학술지 편집을 담당하고 있으며, 현재 국제시뮬레이션학회(SCS)에서 발간하는 학술지인 *Transactions of SCS* 의 Editior-in-Chief 을 맡고 있으며, 그외 다수의 국제 학술지 (*SIMULATION; Int. Journal of Intelligent Systems and Control; IEEE Simulation Digest)* 의 Associate Editor를 맡고 있음. 저서(공저자: B.P. Zeigler, H. Praehofer)로는 2000년 미국 Academic Press에서 발간한 모델링 시뮬레이션 전문 교재인 *Theory of Modeling and Simulation(2nd edition)* 이 있음. 한국시뮬레이션 학회 초대 편집위원장을 역임하였고, 현재 학술부회장임. 국제 학술 단체인 IEEE(국제 전기전자 학회) 및 SCS(국제 시뮬레이션 학회)의 Senior Member이며, ACM(미국전산학회) 및 Eta Kappa Nu 의 Regular Member 임.