DEVS 시뮬레이션의 네트워크 기반 분산처리 방법 연구

송호섭
포스코 ICT SW 융합기술팀
e-mail : webcus@poscoict.com

A Study on Method of Destributed Processing For DEVS Simulation Based on Network

Ho Seop Song
SW Convergence Technology Team

요 약

DEVS(Discrete Event System Specification)형식론은 계층적이고 모듈화 된 형태로 이산사건 시스템을 기술하는 수학적 이론이다. DEVS 형식론에 기반한 모델링과 시뮬레이션은 여러 시뮬레이션시스템에 적용되고 있으며, 보통 단일서버에 단일 프로세스로 구현되고 있다(3). 본 연구는 DEVS 모델을 네트워크기반의 여러 분산서버에 나누어서 모델링하며, 이를 시뮬레이션 하기 위한 처리 기법에 관한 것이다.

1. 서론

Zeigler 에 의해 개발된 DEVS 형식론은 계층적인 모델을 구성 할 수 있는 특성을 가지고 있다. 기본적인 구성요소들을 더 복잡한 연결 모델로 구성 할 수 있도록 수학적틀을 제공 한다. DEVS 모델은 atomic 모델과 coupled 모델로 나뉜다. Atomic 모델은 모든 모델을 구성하는 가장 기본단계의 구성 모델이며, 이 atomic 모델들을 조합하여 Coupled 모델을 구성 할 수 있다. 이러한 Coupled 모델은 또다른 Atomic 모델 또는 Coupled 모델과 조합하여 새로운 Coupled Model 을 구성하는 계층적 관계를 지원한다.
Atomic 모델은 다음과 같이 정의된다(3).

M = < X, Y, S, δ ext, δ int, λ , ta >.
X : 외부 입력 집합
Y : 외부 출력 집합
S : 상태변수 집합
δ int : S → S : 내부사건 처리함수
δ ext : Q × X → S : 외부사건 처리함수
λ : S → Y : 출력함수
ta : S → Real : 시간진행함수
Q = {(s,e) | s∈S, 0≤ e≤ ta(s)} : 모든 상태의 집합

Coupled 모델은 하위 모델들을 포트로 연결하여 계층적인 관계를 구성하고 표현법은 다음과 같이 정의 된다.

DN = < X, Y, M, EIC, EOC, IC, SELECT >
X : 외부 입력 집합
Y : 외부 출력 집합
M : 구성요소 이름 집합
EIC : 외부 입력 연결 관계
EOC : 외부 출력 연결 관계
IC : 내부 연결 관계
SELECT : 구성요소의 우선권

대다수의 DEVS 형식론을 적용한 시뮬레이션들은 보통 단일서버의 단일프로세스에서 구성이 되어진다. 이는 시뮬레이션 수행 시 모델링 객체의 단일 시스템 점유에 따른 전체시스템의 부하를 가져올 수 있다.
본 연구에서는 네트워크 환경하에 DEVS 모델을 분산시켜 모델링 하고, 이를 처리 할 수 있는 방법을 다룬다.

2. DEVS 의 네트워크기반 분산처리 방법

본 연구에서 제시한 분산처리를 위한 시스템은 그림 1. 에서와 같이 네트워크, 모델링 객체 DB, 단위시뮬레이션 시스템과, 이벤트기반 분산처리 인터페이스로 구성되어 있다. 각각의 역할은 다음과 같다.

가) 네트워크 환경의 역할
단일 서버에서 구현되는 시뮬레이션과는 달리 네트워크환경하의 여러 서버에 시뮬레이션 모델링 객체들이 분산되어 설계 되어진다. 이에 따라서 네트워크는

전체 시스템을 이루는 구성 요소 중 하나이다.

나) 모델링 객체 DB 의 역할

네트워크 환경하에 각각 떨어져서 존재하는 모델링 객체들의 데이터 공유를 위하여 DB 에 모델링객체의 시뮬레이션 결과 정보들을 저장한다. 객체의 정보를 찾기 위하여 Data ID 를 필수 컬럼으로 둔다.

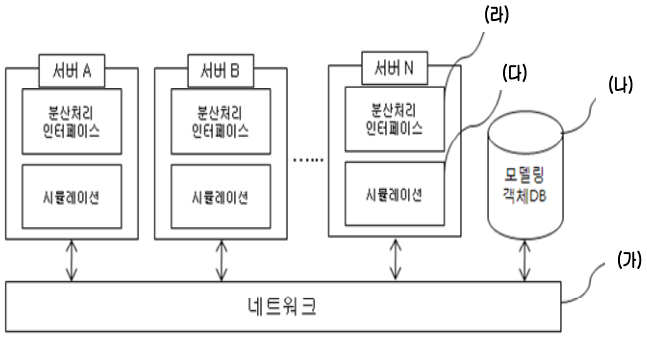
다) 단위 시뮬레이션 시스템의 역할

이벤트기반 시뮬레이션 시스템으로써 DEVS 기반 시뮬레이션 알고리즘을 인터페이스와 동작하도록 수정하여 사용 할 수 있다. 모델링 객체간에 데이터 전달처리가 이벤트가 발생할 때마다 이루어 진다.
분산 처리 시뮬레이션을 위하여 객체모델링시에 기능으로 나누어 설계 하며, 나누어진 모델링 객체는 네트워크환경하의 또 다른 시뮬레이션시스템에 존재 하게 된다.
이는 병렬로 분산수행 하게 된다.
DEVS 시스템에 아래 두가지의 기능이 추가 된다.

1. 분산된 네트워크에 존재하는 모델링 객체와의 이벤트 상관관계를 분산처리 인터페이스에 등록한다.
2. 분산된 네트워크에 존재하는 특정 모델링 객체로의 Data 전송을 위하여 분산처리 인터페이스로의 전송정보를 등록한다.

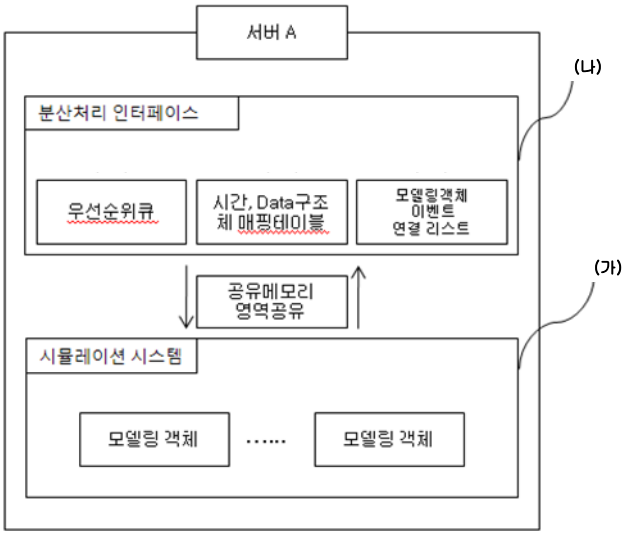
라) 분산처리 인터페이스의 역할

분산 네트워크 상에서 기능별로 모델링 되어진 객체들의 상호 연결 관계를 관리 하며, 그 객체간에 데이터를 전달하는 역할을 수행한다.



(그림1.) DEVS 의 네트워크기반 분산처리 방법

하나의 시뮬레이션 시스템에서 모델링 되어진 객체들은 시뮬레이션 수행을 위하여 지정된 시간에 수행을 하게 된다.
만약 한 개의 객체가 부하가 많은 알고리즘을 수행하여 CPU 를 점유하게 된다면 다른 객체들은 수행하고 있는 객체가 처리를 완료할 때까지 기다렸다 뒤늦게 수행을 하게 되며, 결국은 자신들이 수행되어져야 할 시간에 수행을 못 하게 된다.
이런 점을 해결 하기 위하여 객체 모델링을 기능상으로 여러 시뮬레이션시스템으로 분산하여 설계를 하고 이를 인터페이스를 통하여 연결 시켜 준다. 시뮬레이션간의 인터페이스는 네트워크 환경하에 원격으로 존재하며, 마치 한 개의 단일 시스템에서 시뮬레이션 하는 것처럼 이벤트 처리를 수행 하게 된다.
여러 시뮬레이션 시스템에 나누어 존재하는 여러 모델링 된 객체는 서로의 데이터를 공유하기 위하여 DB 에 시뮬레이션 결과 및 객체 정보를 저장하게 된다.



(그림 2.) 분산처리 인터페이스와 시뮬레이션 시스템과의 상관 관계

그림 2. 의 시뮬레이션 시스템의(가) 동작은 다음과 같다. 이벤트 기반의 시뮬레이션 시스템으로써 각 모델링 된 객체들은 알고리즘 처리를 위하여 수행되어져야 할 시간을 스케쥴링 한다.
모델링 객체들은 상태정의에 따른 이벤트 발생 목록을 가지고 시뮬레이션을 수행하며, 객체끼리의 연결관계 목록을 가지고 있다.
다른 분산 시스템에 존재하는 모델링 객체와의 연결관계는 분산처리 인터페이스에 등록 하며,
이 연결관계는 아래와 같이 구성되어진다.

1. Source Event : 현 객체의 이벤트
2. Target IP : 상대 객체가 존재하는 서버의 IP
3. Target Object ID : 상대 객체의 ID
4. Target Event : 상대 객체의 이벤트

분산환경하에 존재하는 상대 객체로 data 를 전송할 때는, 분산처리 인터페이스에 스케쥴링을 하며, 이때 데이터 구조는 아래와 같다.

1. TargetIP : 상대 객체자 존재하는 서버의 IP
2. Time : 데이터가 전송되어져야 할 시간
3. Target Object ID : 상대 객체의 ID
4. Target Event : 상대 객체의 Event
5. Data ID : 전송되어질 Data 의 ID

객체 모델링 단계에서 알고리즘 수행 부하가 큰 객체는 분산환경하에서 분리하여 설계가 가능하다.

여러 서버에 각각 존재하는 시뮬레이션 시스템은 개별적으로 동작을 하며, 분산처리 인터페이스에 의한 상호간 데이터 전송을 통하여 단일서버에서 동작하는 것처럼 시뮬레이션 수행이 가능하다.

그림 2. 의 분산처리 인터페이스(나)의 동작은 다음과 같다.
분산처리 인터페이스는 단위 시뮬레이션 시스템과 동일한 하드웨어에 존재하며 공유메모리를 사용하여 시뮬레이션시스템과 데이터를 공유한다.
이는 인터페이스와 시뮬레이션 간의 분산처리기술, 인터페이스와 인터페이스간의 분산처리기술을 구현한 소프트웨어시스템이다. 동작 알고리즘은 다음과 같다.

1. 분산환경하에 존재하는 모델링객체간의 연결관계를 관리 한다.
   공유메모리를 통하여 인터페이스가 연결관계구조체를 가져와서 리스트를 관리 한다.
2. 타 모델링객체로부터 전송된 데이타를 인터페이스가 받아서 리스트를 관리 하며 우선순위큐를 계속적으로 점검하여 전송되어질 시간에 도달하면 시간과 구조체 맵핑테이블를 비교하여 구조체 정보를 검색한다.
   시뮬레이션시스템의 Target Object ID 를 가진 모델링객체가 받을 수 있는 TargetEvent 로 Data 를 전송한다. 이때 Data 는 DB 에 존재하는 Data ID 를 검색하여 전송하게 된다.
3. 시뮬레이션시스템으로부터 분산환경하에 존재하는 모델링객체로 정해진 시간에 수행되어질 데이터를 전송하고자 하면 시뮬레이션으로부터 공유메모리를 통하여 구조체를 전송 받는다.
   포함된 Target IP 에 존재하는 인터페이스로 전달받은 구조체를 전송 한다.
4. 시뮬레이션시스템으로부터 인터페이스의 한 이벤트가 발생 했을 시는 Source Event 를 검색하고, Target IP 에 존재하는 인터페이스로 데이타를 전송 한다.

네트워크 환경하에 분산되어 존재하는 모델링 객체들은 서로 다른 하드웨어 시스템을 사용하므로 데이터 공유를 위하여 DB 를 사용한다. DB 는 각각의 모델링객체들의 정보를 담고 있으며, 분산 환경에서 존재하는 객체데이터를 찾기위해 객체 ID, 객체 ID, 객체이름, 인터페이스 ID 의 정보가 필수 이다.

3. 결론

본 연구에서 제시된 시스템은 모델링 되어진 객체들이 시뮬레이션 수행시에 지정된 시간에 원할하게 수행이 될 수 있도록, 네트워크 환경하에 분산처리하는 방법에 관한 것이다. 특정 객체로부터의 시스템 점유현상을 해결 할 수 있으며, 이로 인하여 각각의 객체들은 원하는 수행시간에 동작되도록 설계를 할 수 있다.

참 고 문 헌

(1) Zeigler, B.P., " Object-Oriented Simulation with Hierarchical, Modular Models, Academic Press, 1990.
(2) Zeigler, B.P., " Hierarchical Modular Discrete-Event modeling in an Object-Orient environment", Simulation, 1987, pp.219-230.
(3) Tag Gon Kim, DEVSim++ User's Manual: C++ Based Simulation with Hierarchical Modular DEVS Models, Computer Engineering Lab, KAIST, 1994.