프로토콜 평가를 위한 DEVSim++ 와 NS2 의 연동 환경

김회준* · 김탁곤**

DEVSim++ - NS2 Interoperating Environment for Protocol Evaluation

Hoijun Kim · Kim Tag Gon

Abstract

This paper proposes a methodology for development of protocol models. The methodology attempts to employ two modeling environments in models development, NS2 and DEVSim++, which will interoperate during simulation. NS2 is a widely used network simulator in protocol research, which employs an informal modeling approach. Within the approach time and state information of protocol models are not explicitly described, thus being hard to validate model. On the other hand, the DEVS formalism is a mathematical framework for modeling a discrete event system in a hierarchical, modular manner. In DEVS, model's time and state information is described explicitly. By using DEVS formalism, models can easily be validated and errors in the modeling stage can be reduced. However, the DEVS simulator, DEVSim++, supports a small amount of models library which are required to build simulation models of general communication network. Although NS2 employs an informal modeling approach and models validation is difficult, it supports abundant models library validated by experimental users. Thus, combination of DEVS models and NS2 models may be an effective solution for network modeling. Such combination requires interoperation between DEVSim++ simulator and NS2 simulator. This paper develops an environment for such interoperation. Correctness and effectiveness of the implemented interoperation environment have been validated by simulation of UDP and TCP models.

Key Words : DEVS, DEVSim++, NS2, 프로토콜, 연동 시뮬레이션

1. 서론

프로토콜을 DEVS로 모델링하면 동일 모델로 정확성 검증과 성능 평가를 할 수 있어 프로토콜의 개발주기를 단축시킨다. 기존 네트워크 시뮬레이터 NS2는 프로토콜의 성능 평가에 널리 쓰여왔고 사용자들에 의해 경험적으로 검증된 모델 라이브러리를 갖추었으나 모델검증에 어려움이 있다.

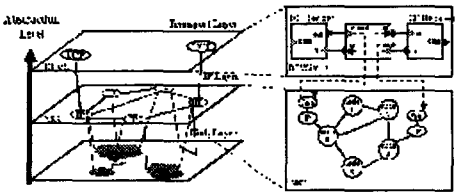

그림 1. 연동 환경에서의 시뮬레이션 모델 예

본 논문에서는 DEVS와 NS2의 장점을 조합하여 프로토콜의 DEVS 모델과 기존 NS2의 모델 라이브러리로 그림1과 같은 시뮬레이션 모델을 구성하여 프로토콜을 성능평가 할 수 있는

* 한국과학기술원 전자전산학과
** 한국과학기술원 전자전산학과

연동환경을 제안하고 구체적인 스펙을 정한 후, 구현결과의 정확성검증과 성능평가 결과를 보인다.

2. DEVSim++와 NS2의 개요

2.1 DEVSim++ 개요

DEVSim++는 시스템의 수학적 표현인 DEVS 모델을 실행가능한 프로그램으로 구현하는 환경이다. DEVS 모델을 DEVSim++ 로 구현하는 과정은 기계적이고 현재 자동화 추진중이다.

프로토콜의 개발은 그림 2와 같이 자연어 또는 가상코드로 주어진 불명확한 스펙에서 시작하여 유한상태기계 또는 오토마타 기반의 논리모델을 생성 논리검증을 마친 후 성능모델로 변환하여 시뮬레이션으로 성능을 평가한다.

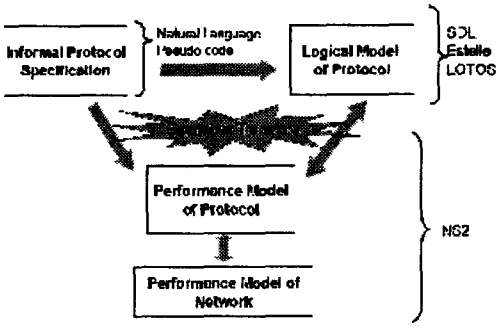

그림 2. 프로토콜 개발 주기

프로토콜의 DEVS 모델은 상태와 시간정보를 모두 가지고 있으므로 논리검증과 성능평가를 단일모델로 해결할 수 있다. 이런 점은 논리검증을 마친 모델을 성능모델로 변환하는 과정에서 개입하는 오류를 없애므로 개발 주기를 단축시킨다. 또한 DEVS 모델은 포트를 이용하여 다른 모델과 통신하고, 모델은 시뮬레이터로부터 명확히 구분된다. 그러므로, 모델러는 프로토콜에만 집중할 수 있으므로 오류가 적은 모델링이 가능하다.

DEVS 시뮬레이터 DEVSim++는 DEVS 모델로부터 자동화된 코드의 생성을 지원하고 시뮬레이션에 필요한 수학 라이브러리와, I/O trace analyzer, GUI기반의 시뮬레이터를 제공한다.

2.2 NS2 개요

UC Berkeley, LBL,USC/ISI, Xerox PARC 가 공동 수행한 Virtual INternet Testbed (VINT) 프로젝트의 결과 Network Simulator 2가 개발되었다. NS2는 다양한 프로토콜, 라우터, 링크의 모델 라이브러리와, 스크립트를 이용하여 시뮬레이션 모델을 빠르게 구성할 수 있는 환경을 제공한다.

NS2 에서 모델은 OTcl 또는 C++ 의 객체에 기반한다. 모델의 상태변수는 객체의 멤버변수로, 모델의 상태변환은 객체의 메쏘드로, 모델간의 통신은 암묵적으로 지정된 send(), receive() 함수로 수행한다. 통신하는 두 모델은 상대 모델을 서로 알아야 하고, 상대 모델의 receive() 메쏘드를 호출함으로써 통신한다.

모델의 시간정보는 명시적으로 기술되지 않고 모델이 스케쥴하는 이벤트의 처리시각에 내포된다. 스케쥴한 이벤트가 처리될 때 모델이 지정한 처리함수가 호출되고 이 함수가 모델이 수행할 상태전이함수, 출력함수를 호출함으로써 모델은 시간진행에 따른 동작을 한다.

3. DEVSim++와 NS2의 모델, 시뮬레이터 비교

이 장에서는 연동 설명에 앞서 DEVSim++와 NS2의 모델, 시뮬레이터를 비교 설명한다.

3.1 모델 비교

DEVSim++에서 모델은 명시된 입출력 포트를 이용하여 통신한다. Simulator 는 atomic model의 출력에 현재 시각의 정보를 덧붙여 메세지로 만든 후 coupled model의 연결정보를 이

용하여 라우팅 되도록 한다. Atomic model은 자신의 포트가 연결되어 있는 수신모델을 전혀 알지 못 하고 coupled model의 연결정보가 수신모델과 수신 포트를 결정한다.

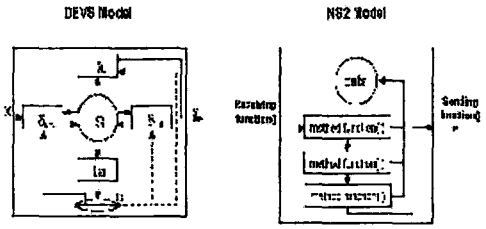

그림 3. DEVSim++와 NS2의 모델 비교

NS2에서 모델은 입출력 함수를 이용하여 통신한다. 각각의 NS2 모델은 통신하는 수신 모델을 알고, 수신 모델의 입력함수를 호출하고 보낼 데이터를 인자로 전달함으로써 통신한다. 수신 모델은 입출력이 발생할 때의 시간을 global simulation clock을 읽어서 알아낸다.

3.2 시뮬레이터 비교

DEVSim++ 시뮬레이터는 계층적인 구조를 가진다. Simulator는 atomic model에 명시된 동작을 수행하고, coordinator는 coupled model에 명시된 모델 포트간의 연결 정보를 이용하여 메세지를 전달한다. Simulator와 coordinator들은 X, Y, *, DONE의 4가지 메세지를 주고받으며 시뮬레이션을 진행한다. 시뮬레이션의 진행은 최상위 coordinator가 simulator 들이 DONE-메세지로 보고하는 next event time 들 중 최소를 현재 시각으로 업데이트하고 최소의 next event time 을 보고한 simulator 에게 *-메세지를 보내어 atomic model을 구동하고 Y-메세지가 발생하면 이를 처리하는 것을 하나의 transaction 으로 한다. 모든 simulator 가 무한대의 next event time 을 보고할 때까지 이는 반복된다. DEVSim++ 모델러는 이러한 시뮬레이션 구조에 대해서 전혀 알 필요가 없고, 기술할 필요도 없다.

NS2 시뮬레이터는 단일 이벤트 스케쥴링 큐에 이벤트를 스케쥴하고 시간순에 따라 이벤트를 처리함으로써 시뮬레이션을 진행한다. 이벤트는 발생할 시각과 핸들러로 이루어지고, 핸들러의 이벤트 처리 함수를 변경함으로써 핸들러를 소유한 모델이 다양한 동작을 하도록 한다. 시뮬레이션의 진행은 이벤트 스케쥴링 큐의 선두 이벤트를 꺼내서 이 이벤트의 스케쥴된 시각으로 시뮬레이션 시간을 업데이트 하고 이벤트의 핸들러를 호출하고, 그 결과 새로운 이벤트가 이벤트 스케쥴링 큐에 정렬삽입되는 것을 하나의 transaction 으로 한다. 이벤트 스케쥴링 큐가 비워질 때까지 이는 반복된다. NS2 모델러는 반드시 이벤트 스케쥴링 큐와 이벤트 클래스, 핸들러 클래스에 대해 알아야하고, 데이터를 주고받을 모델과 입출력 함수에 대해 알아야 한다.

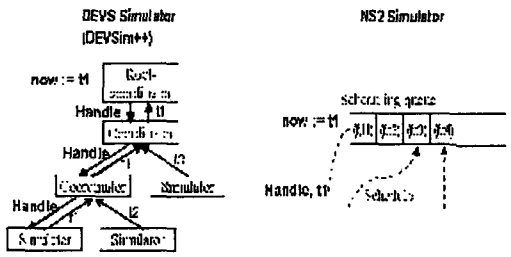

그림 4. DEVSim++와 NS2의 시뮬레이터 비교

4. DEVSim++와 NS2의 연동환경

4.1 연동 요건

3장에서 보였듯 DEVSim++와 NS2는 모델 구조와 시뮬레이션 진행방법이 서로 다르다. 이러한 두 시뮬레이터를 연동하기 위한 요건은 아래와 같다.

- 데이터 전달
  - ▸두 시뮬레이터에게 지정된 통신 방법이 있고, 이를 이용하여 통신한다.

- ▸두 시뮬레이터에게 알려진 전달할 이벤트와 데이터의 타입이 있고, 이를 전달한다.
- ▸두 시뮬레이터는 타 시뮬레이터의 모델과 coupling 할 수 있는 인터페이스를 사용자에게 제공해야 한다.

● 시뮬레이터 간 시간 동기화

- ▸두 시뮬레이터는 시간 동기를 맞추어 수행해야 한다.
- ▸두 시뮬레이터는 타 시뮬레이터의 모델에게 전달해야 할 이벤트가 있을 때, 이벤트가 발생한 시각을 함께 전달해야 한다.

본 구현에서 기본적으로 데이터 전달은 공유메모리를 이용하고, 시간 동기화는 전역가상시간(global virtual clock)을 이용하였다.

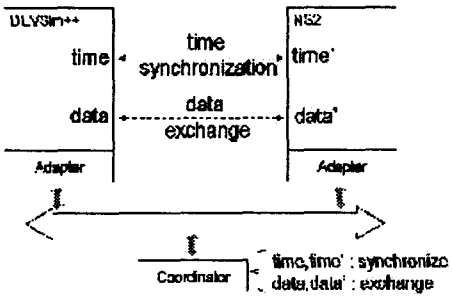

그림 5. 시뮬레이터 연동의 필요요건

4.2 연동 환경 구현

시뮬레이터를 제어할 수 있는 모듈이 필요하다. 이 모듈은 기본적으로 시뮬레이터를 초기화, 일시중지, 재개, 종료할 수 있어야 한다. 또한, 시뮬레이터의 시각을 가 점프할 다음 시각을 상대 시뮬레이터를 제어하는 모듈과 상의한 후, 시뮬레이터를 일시중지 시키고 상대 시뮬레이터가 먼저 진행하기를 기다리던지, 시뮬레이터를 진행할 것인지를 결정해야 한다. 이러한 시뮬레이터를 제어하는 모듈을 연동적응자(adapter)라 정하기로 한다. 그림 6은 연동적응자의 동작을 보인다. 연동에 참여하는 시뮬레이터들은 각각 1개의 연동적응자에 의해 제어된다.

연동적응자 사이의 통신은 서버-클라이언트 구조로 할 수 있으나, 확장성이 떨어져 또 다른 시뮬레이터를 연동에 참여시키기가 힘들다. 그러므로 별도의 모듈을 두어 그 모듈이 연동적응자의 통신과 실행순서를 제어하도록 한다. 이 모듈을 연동조정자(interoperation coordinator)라 하기로한다. 그림 7은 DEVSim++ 와 NS2, 연동적응자, 연동제어자 로 이루어진 연동 환경의 동작 예를 보인다.

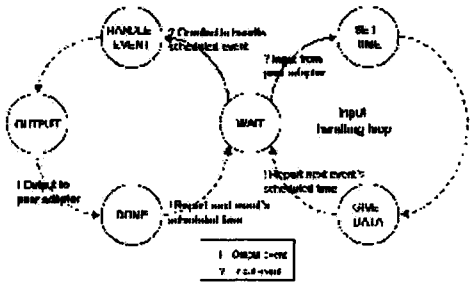

그림 6. 연동적응자의 상태전이 다이어그램

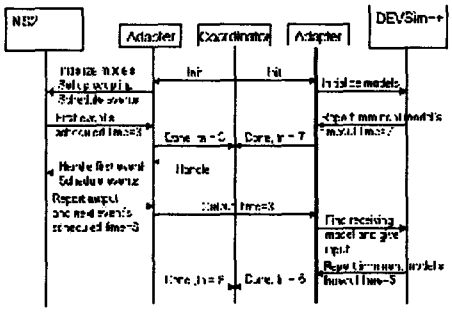

그림 7. 연동 시뮬레이션 환경의 동작 예

5. 연동 환경의 정확성 검증과 성능 평가

5.1 정확성 검증

연동 환경의 정확성을 검증하기 위해 기존 NS2 모델들을 이용하여 시뮬레이션 모델을 구

성하여 시뮬레이션하고, 이 중 일부를 동일한 DEVSim++ 모델로 교체하여 연동 시뮬레이션 모델을 구성하여 이 둘의 시뮬레이션 결과 교체된 프로토콜 모델의 I/O 트레이스가 같음을 보인다.

그림8 은 NS2의 UDP 모델을 이용한 정확성 검증용 시뮬레이션 모델을 보이고, 그림 9는 그 결과를 보인다. 그림 9의 그래프는 두 시뮬레이션 모델의 UDP 모델이 겪는 I/O 트레이스를 겹쳐서 도시한 것으로 I/O 트레이스가 일치함을 확인할 수 있다. 그림10 은 TCP 모델을 이용한 정확성 검증용 시뮬레이션 모델을 보이고, 그림 11은 결과를 보인다.

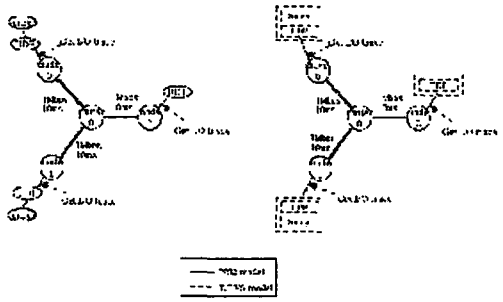

그림 8. UDP 모델을 이용한 연동 환경의 정확성 검증 예

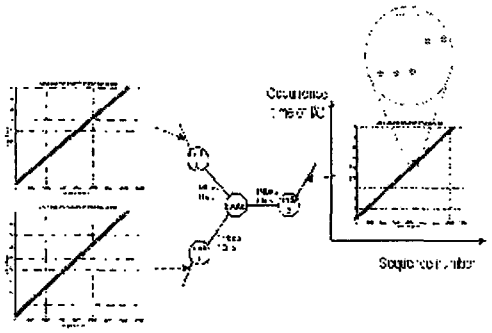

그림 9. UDP 모델을 이용한 연동환경의 정확성 검증 결과

5.2 성능 평가

I/O 레벨에서 동일한 모델을 이용하여 NS2 만으로 시뮬레이션 할 때 소요되는 시간과 DEVSim++ - NS2 연동을 이용하여 시뮬레이션 할 때 소요되는 Adapter와 adapter시간을 비교함으로써 연동 환경의 성능을 평가한다. 이용된 시뮬레이션 모델은 그림8 과 그림 10 의 모델들이다. 평균적인 시뮬레이션 소요시간은 표1 와 같다. 연동 시뮬레이션의 경우 시간이 더 소요되는 것을 확인할 수 있다.

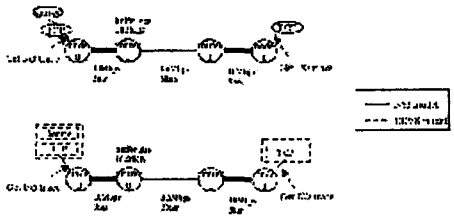

그림 10. TCP 모델을 이용한 연동 환경의 정확성 검증 예

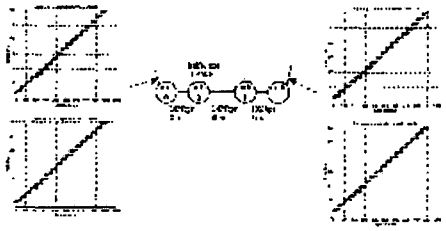

그림 11. TCP 모델을 이용한 연동 환경의 정확성 검증 결과

표 1 시뮬레이션 소요시간 비교

	소스 트래픽 타입	NS2 평균 시뮬레이션 소요시간 (초)	연동 시뮬레이션 소요시간 (초)	시간 증가분 (%)
UDP	지수분포	73.5(3.3)	114.5(5.2)	55.9
	파레토분포	52.2(2.2)	78.9(2.6)	51.4
TCP	지수분포	206.0(19.8)	276.1(1.76)	35.1
	파레토분포	235.4(12.6)	278.4(6.5)	18.4

5. 결론

본 논문에서는 프로토콜의 평가를 위한 DEVSim++ 와 NS2의 연동 시뮬레이션 환경을

제안하였다. 연동 환경은 DEVSim++와 NS2의 장점을 활용하여, DEVS 형식론에 기반한 프로토콜 모델링과 NS2의 모델 라이브러리를 이용한 빠른 성능평가 환경을 제공한다.

논문에서는 DEVSim++와 NS2의 모델과 시뮬 레이터를 비교하고 이들을 연동하기 위한 시간 동기화와 데이터 전달의 스펙을 정의하였다. 시간동기화는 global virtual time 을 이용하였고, 데이터 전달은 공유메모리를 이용한 프로세스간 통신을 이용하였다. 연동 스펙을 구현하기 위해 각각의 시뮬레이터를 제어할 연동적응자와 이들을 시간순대로 실행시킬 연동조정자를 구현하였다. UDP 모델과 TCP 모델을 이용하여 구현된 연동환경의 정확성을 검증하고 성능을 평가하였다.

참고문헌

1) Tag G. Kim and B.P. Zeigler, The DEVS Formalism: Hierarchical, Modular System Specification in an Object Oriented Framework, Proc in 1987 Winter Computer Simulation Conference, p.559-566, Dec., 1987, U.S.A
2) Tag Gon Kim, DEVSim++ User's Manual, 1994, ftp://sim.kaist.ac.kr/pub
3) Kevin Fall, Kannan Varadhan, The ns Manual, 2001, http://www.isi.edu/nsnam/ns/doc/
4) B.P.Zeigler H. Praehofer T. Kim, Theory of Modeling and Simulatio, 2nd edition, San Diego: Academic Press, 2000
5) John Heidemann, Kevin Mills, Sri Kumar, Expanding Confidence in Network Simulation, DARPA/NIST Network Simulation Validation Workshop May 1999