한국시뮬레이션학회 논문지
Vol. 19, No. 2, pp. 157-164 (2010. 6)

기능 검증 및 성능 평가 통합 접근 방법을 통한 통신 프로토콜 개발을 위한 SDL-OPNET 코-시뮬레이션 기법

양치평[1] · 김태형[1†]

SDL-OPNET Co-Simulation Technique for the Development of Communication Protocols with an Integrated Approach to Functional Verification and Performance Evaluation

Qi Ping Yang · Tae Hyong Kim

ABSTRACT

While both functional verification and performance evaluation of a system are necessary for the development of effective and reliable communication systems, they have been usually performed individually through functional modeling with formal language tools and performance modeling with professional network performance evaluation tools, respectively. However, separate and duplicated modeling of a system may cause increase of the cost and inconsistency between the models. In order to overcome this problem, this paper proposes an integrated design technique that estimates the performance of a communication protocol designed in SDL with SDL-OPNET co-simulation. The proposed technique presents how to design a co-simulation system with the environment functions of Tau and the external system module of OPNET. InRes protocol was used as an example to show the applicability and usefulness of the proposed technique.

Key words : Functional verification, Performance evaluation, SDL, OPNET, Co-simulation

요 약

우수하고 신뢰성 있는 통신 시스템의 개발을 위해 시스템에 대한 기능 검증과 성능 평가가 모두 필수적인데 반해 이들은 주로 형식 언어 도구를 이용한 기능 모델링과 전문 네트워크 성능 평가 도구에 의한 성능 모델링을 통해 개별적으로 수행되어 왔다. 그러나 한 시스템을 별도로 중복하여 모델링 하는 것은 비용의 증가와 모델 간 불일치를 가져오게 된다. 본 논문은 이 문제를 해결하기 위해 SDL-OPNET 코-시뮬레이션을 통해 SDL로 설계된 통신 프로토콜의 성능을 평가하는 통합 설계 기법을 제안한다. 제안 기법은 Tau의 환경함수와 OPNET의 외부시스템 모듈을 이용하는 코-시뮬레이션 시스템의 설계 방법을 제시한다. InRes 프로토콜이 예로 사용되어 제안 기법의 적용가능성과 효용성을 보여준다.

주요어 : 기능 검증, 성능 평가, SDL, OPNET, 코-시뮬레이션

1. 서 론

최근 초고속, 멀티미디어, 이동 통신 등 통신 서비스 요구사항이 다양하고 복잡해짐에 따라 끊임없이 새로운 통신 시스템이 등장하고, 이에 따른 통신 시스템 구현제품이 개발되고 있다. 일반적으로 새로운 통신 시스템을 설계할 경우 혹은 기존 시스템의 성능을 개선할 경우, 설계

*이 논문은 2006년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2006-331-D00471).
2010년 5월 17일 접수, 2010년 6월 20일 채택
[1] 금오공과대학교 컴퓨터공학부
주 저 자 : 양치평
교신저자 : 김태형
E-mail; taehyong@kumoh.ac.kr

된 시스템의 성능을 평가하기 위해 성능 분석을 수행하게 된다. 성능 분석 방법은 수학적 해석이나 구현제품의 실제 성능측정을 사용할 수도 있으나, 설계의 정확도 및 기능 동작의 복잡도가 증가함에 따라 시뮬레이션 방법이 현실적이고 신뢰성 있는 방법으로 인정받고 있다.

다른 한편으로 다양하고 복잡한 서비스 요구사항을 만족시키는 통신 시스템을 설계하기 위해서는 통신 시스템의 구조나 기능 동작, 서비스 간의 관계 등에 대한 명확한 기술이 필수적이다. 이러한 필요성에 따라 SDL[1]과 같은 형식 기술 언어(formal description languages)가 개발되어 현재 ITU-T에서는 새로운 통신 프로토콜의 표준화 시 프로토콜 동작에 대한 SDL 명세를 제공하도록 권고하고 있다. IBM 사의 Rational Tau SDL Suite[2] (이하 Tau)와 같은 SDL 설계 도구를 이용할 경우 설계된 형식 모델 동작의 손쉬운 검증 뿐 아니라 구현을 위한 코드를 자동으로 생성할 수 있다.

이렇듯 새로운 통신 시스템의 설계에 있어서 시스템의 성능 평가와 형식적 설계를 통한 기능 검증이 모두 중요한데, 실제로는 이 두 과정을 모두 충실히 수행하여 통신 시스템을 설계하는 경우가 흔하지 않다. 즉, 시스템의 기능 검증과 성능 평가 중 보다 중요하다고 판단되는 한 부분에 치중하여 설계를 수행하고 다른 한 부분은 생략되거나 추후로 미루어지게 된다. 이는 프로토콜에 대한 기능 검증과 성능 평가가 일반적으로 서로 다른 설계 환경에서 별도의 모델링을 통해 독립적으로 수행되기 때문이다. 이 경우, 중복 모델링에 따라 비용이 증가하게 되며 설계된 모델 간 불일치 문제가 발생하지 않도록 모델 동일성 보장이 필요하다.

그간 이러한 문제를 해결하기 위해 기능 검증과 성능 평가를 통합하고자 하는 다양한 연구가 수행되어 왔다[3-5]. 통합 설계 기법을 이용하여 신뢰성 있고 우수한 성능의 프로토콜을 개발하는 과정은 대략 다음과 같다. 먼저 형식 모델링 언어를 지원하는 설계도구로 프로토콜 모델을 설계하고 기능상의 정확성을 검증한 다음, 검증된 모델에 대해 성능 평가 도구를 이용하여 시뮬레이션 환경을 구성하고 성능평가를 통해 최종 모델을 얻게 된다. 이러한 통합 설계 기법이 실제 통신 프로토콜 설계와 개발에 적극적으로 활용되기 위해서는 통합 설계 기법의 신뢰성과 효용성이 입증되어야 한다. 그런데, 통합 설계 기법이 표준 형식언어가 아닌 비표준 확장언어를 사용한다거나[6] 기존 성능평가 도구가 아닌 자제 개발 성능평가도구를 사용하는 경우[3], 제안 기법의 확장성이나 신뢰성을 검증받기 어렵다. 따라서 통합 설계 기법의 효용성과 신뢰성 확보를 위해서는 표준 형식기술 언어와 검증된 설계 및 평가 도구를 이용하는 것이 필요하다.

기존의 검증된 평가 도구를 사용하는 통합 설계 기법은 기능 모델을 성능 모델로 변환하는 모델 매핑 방식과 기능 검증 도구와 성능 평가 도구를 연계하는 도구 연결 방식으로 분류될 수 있다[4,5]. 최근 도구 연결 방식으로 SDL 설계 도구 Tau와 공개 성능 평가 도구 ns-2를 연계하는 통합 설계 기법[7]이 제안되었지만, 성능 평가 도구로 가장 높은 신뢰성을 인정받고 있는 OPNET Modeler(이하 OPNET)[8]는 상용 도구라는 문제로 인해 아직 통합 설계 기법에 거의 사용되지 못하고 있다.

본 논문은 전술한 통합 설계 기법들의 문제점들을 극복하기 위하여 표준 형식 언어인 SDL을 사용하는 설계 및 검증 도구로 가장 널리 사용되고 있는 Tau와 가장 우수한 네트워크 성능 평가 도구로 알려진 OPNET 간의 도구 연결을 이용하는 통합 설계 기법을 제안한다. 두 도구 모두 상용도구인 관계로 도구 연결 방식이 제한적인데, 본 논문에서는 Tau를 이용하여 기능 검증이 완료된 설계 모델에 대해 OPNET에서 제공하는 외부시스템 관련 모듈과 Tau에서 제공하는 환경함수를 이용하는 코-시뮬레이션(co-simulation)을 수행하여 성능을 평가한다.

본 논문의 구성은 다음과 같다. 2절에서는 OPNET과 Tau에서 제공하는 외부시스템 인터페이스에 대해 설명하고, 3절에서 SDL 설계 모델의 성능 시험을 위한 SDL-OPNET 코-시뮬레이션 구조를 제안한다. 4절에서 샘플 프로토콜을 대상으로 코-시뮬레이션 수행 과정 및 성능 평가 결과를 보인 후 5절에서 제안 기법에 대한 토의와 함께 결론을 맺는다.

2. 도구 연결을 위한 외부 인터페이스 환경

2.1 Tau의 외부 환경 인터페이스

Tau는 다른 시뮬레이터와의 코-시뮬레이션을 위한 특별한 인터페이스를 제공하지 않으나 타겟 코드 생성을 위한 외부 환경 인터페이스를 이용하여 외부 시뮬레이터와 메시지 교환을 통한 연동을 수행할 수 있다. Tau의 외부 환경 인터페이스에서 제공하는 환경 함수는 다음과 같다. xInitEnv()는 SDL 시스템이 초기화될 때, xCloseEnv()는 SDL_Halt() 함수에 의해 각각 호출되는 환경 함수로 SDL 시스템이 시작될 때와 종료할 때 수행해야 할 코드를 포함시킨다. xOutEnv()는 신호를 외부로 보내는 것을 제어하는 환경 함수로, 먼저 출력 신호를 확인한 다음, 외부로 나가는 출력이 있을 때 이에 필요한 동작을 수행한

다. xInEnv()는 외부로부터 오는 신호를 SDL 시스템으로 보내는 역할을 하는 환경 함수로, 외부로부터 오는 신호의 유무 확인을 위해 시스템은 논리 상태에 머무는 동안 계속해서 이 함수를 호출하게 된다. 만약 SDL 시스템으로 보내는 신호가 있는 것을 확인하면 xInEnv()는 SDL_Output() 함수를 통해 신호를 SDL 시스템으로 보낸다. 그림 1은 Tau의 환경 함수를 통한 외부 영역 접속 방법을 요약해 보여준다.

xInEnv() 함수의 경우 폴링을 이용하여 외부 환경으로부터 오는 신호의 유무를 확인하기 때문에 시스템이 매우 복잡할 경우 신뢰성이 떨어지는 문제가 생긴다. 따라서 신뢰성 향상을 위해 외부 환경으로부터 오는 신호의 유무를 모니터링하는 쓰레드를 설계하여 사용하는 방법을 사용할 수 있다. Tau는 SDL 모델 컴파일 시 위 외부 함수에 대한 골격 코드(skeleton code)만 생성하므로 외부와의 완전한 연결을 위해서는 위 외부함수들을 완성시켜 주어야 한다.

2.2 OPNET의 외부시스템 인터페이스

OPNET은 다른 외부 소프트웨어와 OPNET 간의 코-시뮬레이션을 지원하기 위해 OPNET 코-시뮬레이션 패키지를 제공한다. OPNET 코-시뮬레이션 패키지는 Esys 모듈, 외부시스템정의(External System Definition; ESD) 모델, Esys 관련 커널프로시저(Kernel Procedure; KP) 및 외부시스템접속(External System Aceess; ESA) API로 구성되어 있다.

OPNET의 Esys 모듈은 OPNET이 외부 시스템과 데이터를 교환하기 위해 사용하는 특수한 게이트웨이로서, 패킷 스트림과 통계정보선 연결을 사용하는 프로세스 모듈과 유사한 형식으로 노드 모델 내에서 정의된다. Esys 모듈은 코-시뮬레이션 수행을 위해 외부 코-시뮬레이션 코드와 OPNET 사이의 인터페이스 즉, 외부시스템인터페이스(External System Interface; ESI)를 정의하는 ESD 모델을 갖는다. OPNET은 ESD에서 정의된 ESI를 통해 코-시뮬레이션 메시지를 주고받게 되는데, 이를 위해 Esys 모듈의 프로세스 모델 내에는 Esys 인터페이스를 통해서 언제 어떻게 데이터를 읽고 쓰는지를 제어하기 위한 op_esys 커널프로시저가 사용된다.

외부 시스템 인터페이스를 위한 코드는 OPNET 메인 함수를 OPNET 내부에 두느냐 외부에 두느냐에 따라 설계방법에 차이가 있는데, 외부 소프트웨어와의 연결을 사용하는 코-시뮬레이션의 경우 일반적으로 OPNET 외부 코드에 메인함수를 두어 전체 코-시뮬레이션 실행을 제어하게 된다. ESA API는 OPNET 외부 코드에서 호출되는 C API로, OPNET 모델과 코-시뮬레이션 코드를 연결하여 시뮬레이션 초기화 및 로딩, 흐름제어 관리, Esys 인터페이스 처리, Esys 인터페이스를 통한 데이터 교환 등을 수행하는데 사용될 수 있다. 그림 2는 코-시뮬레이션을 위한 OPNET 외부시스템 인터페이스의 구조를 보여준다.

3. 제안하는 코-시뮬레이션 구조

3.1 코-시뮬레이션 기반 통합 설계 기법

제안하는 통신 프로토콜의 단일 모델 기반 통합 설계 기법은 설계된 SDL 모델을 기능 검증과 성능평가 모두에 그대로 사용한다. 그림 3은 제안하는 통합 설계 과정을 도시한 것이다. 통신 프로토콜 개발을 위해 프로토콜 요구사항으로부터 표준 형식언어 SDL을 이용하여 프로토콜 모델을 설계한다. 프로토콜 설계 과정은 Tau에서 제공하는 통합개발환경(IDE)에서 수행된다. 설계된 SDL 모델에 대해 Tau에서 제공하는 Simulator와 Validator를 이용해 기능검증을 수행하고 그 결과에 따라 모델 수정 등을 거쳐 요구사항에 부합되는 검증된 SDL 설계모델을 얻

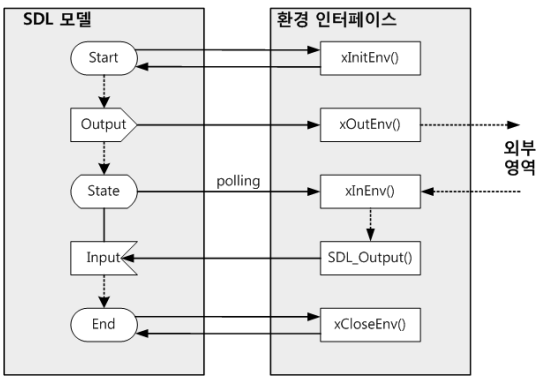

그림 1. Tau 환경 함수의 역할
Fig. 1. Roles of Tau's environment functions

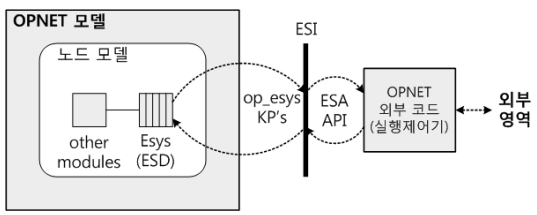

그림 2. OPNET 외부시스템 인터페이스 구조
Fig. 2. Stucture of OPNET's external system interfaces

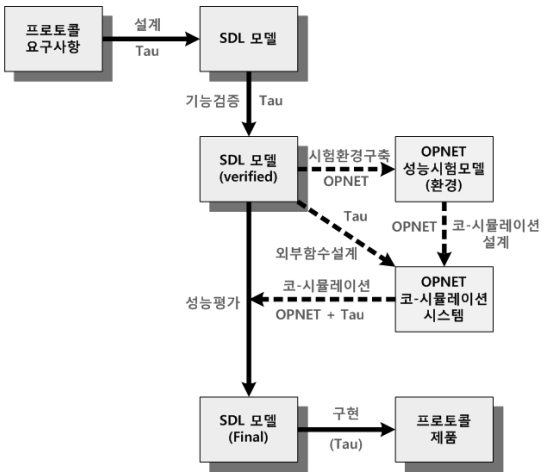

그림 3. 제안하는 단일 모델 기반 통합 설계 과정
Fig. 3. The proposed single model-based integrated design process

게 된다. 이때 Simulator를 이용하여 검증을 수행할 경우 시뮬레이션을 위한 별도의 테스터 모델을 구축할 수 있다. 다음으로 설계된 프로토콜의 성능평가를 위해 OPNET을 이용하여 네트워크 구성과 같은 성능평가 환경모델을 구축한다. SDL-OPNET 코-시뮬레이션 시스템은 OPNET 성능평가 모델과 SDL 기능 모델간의 시뮬레이션을 제어하여 성능평가 결과를 얻게 된다. 코-시뮬레이션 환경 구축을 위해 Tau 환경 함수와 OPNET 외부 시스템 인터페이스 코드가 설계된다. SDL(Tau)과 OPNET 간의 코-시뮬레이션 설정에 대한 자세한 내용은 3.2절에서 다룬다. SDL-OPNET 코-시뮬레이션을 통한 성능평가가 수행되면 그 결과에 따라 SDL 모델이 다시 수정될 수 있다. 이러한 피드백 과정을 거쳐 최종 SDL 모델이 완성되면 Tau에서 제공하는 타겟 포팅 등의 구현 과정을 통해 프로토콜 제품을 얻을 수 있다.

3.2 코-시뮬레이션 시스템의 구조

SDL로 설계된 프로토콜 모델의 성능평가를 위해 SDL 시뮬레이터 Tau와 네트워크 시뮬레이터 OPNET을 연결시켜 코-시뮬레이션을 수행하기 위해서는 두 도구의 연결 방법과 코-시뮬레이션의 제어 방법을 결정해야 한다. 두 도구는 각자 외부 시스템 인터페이스를 가지고 있으므로 기본적인 도구 연결 방식은 도구 간 메시지 교환이 될 것이다. 여기서 대상 프로토콜 모델은 Tau에서 동작하고 그 외 성능평가를 위한 네트워크 구성은 OPNET에서 동작해야 하므로 OPNET 시뮬레이션 시스템을 이러한 구조

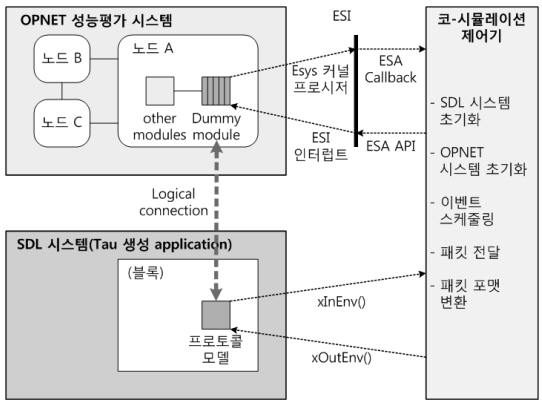

그림 4. 제안하는 SDL-OPNET 코-시뮬레이션 시스템 구조
Fig. 4. The proposed SDL-OPNET co-simulation system structure

에 맞도록 적절히 설계해야 한다. 본 논문은 OPNET에서 성능 평가 환경 모델 설계 시 대상프로토콜을 빈 모듈(dummy module)로 설계하고 외부 코-시뮬레이션 제어 코드를 통한 데이터 교환 방식으로 SDL 모델과 연결하는 방법을 사용한다. 그림 4는 본 논문에서 제안하는 SDL-OPNET 코-시뮬레이션 시스템의 구조를 도시한 것이다.

OPNET 네트워크 모델에서 동작 기술이 빠진 Esys 모듈로 설계되는 대상 프로토콜은 OPNET 내부의 다른 프로토콜 모델로부터 들어오는 패킷에 대해 Esys 커널 프로시저를 통해 ESI에 설정한다. 이 때 ESA callback 함수가 자동으로 호출될 수 있도록 외부 코-시뮬레이션 제어기에 ESA callback 함수를 등록한다. 자동으로 호출되는 callback 함수는 외부 코-시뮬레이션 제어기 내에 정의하며 OPNET에서 보낸 패킷을 ESA API를 통해 가져와 SDL 메시지 포맷에 맞게 포맷을 변환한 다음 SDL 환경함수 xOutEnv() 설계를 통해 SDL 프로토콜 모델로 전송한다. SDL 프로토콜 모델이 메시지 수신에 따른 동작을 수행하고 메시지를 출력하면 코-시뮬레이션 제어기에 설계된 환경함수 xInEnv()는 이 메시지를 OPNET 패킷 포맷에 맞게 변환한 다음 OPNET ESA API를 이용하여 ESI에 설정한다. 이 때 OPNET 시스템에서 ESI 인터럽트가 발생하고 OPNET Esys 모듈은 Esys 커널 프로시저를 이용해 이 패킷을 가져오게 된다.

3.3 코-시뮬레이션의 제어

코-시뮬레이션 제어 코드에서 OPNET-SDL 코-시뮬레이션 수행을 위해서 단순히 메시지 정합만을 수행할 경우 OPNET과 Tau는 각각 개별적으로 시뮬레이션 시간 흐름

을 관리하게 된다. OPNET과 Tau는 모두 불연속 이벤트 기반(discrete event-driven) 시뮬레이션과 실시간(real-time) 시뮬레이션을 함께 지원하지만 두 도구의 시뮬레이션 진행 방법과 시간 관리 방법은 상당한 차이가 있다. SDL은 기본적으로 타이머를 통한 시간 진행 외에는 별도의 시간 진행 방법이 없기 때문에, SDL 모델에 타이머가 존재하지 않는다면 SDL 모델에서의 동작은 시간 진행을 발생시키지 않으므로 OPNET 모델의 시뮬레이션 시간 흐름 제어만으로도 코-시뮬레이션 관리가 가능하다. 하지만, SDL 모델에 타이머를 통한 시간 진행이 존재한다면 SDL-OPNET 간 정확한 코-시뮬레이션을 위해서는 두 시뮬레이션 시스템의 시간 동기를 위해 별도의 제어방법이 필요하다. 제안하는 SDL-OPNET 코-시뮬레이션 제어 기법은 코-시뮬레이션 제어 코드에서 OPNET 시스템과 SDL 시스템의 이벤트를 통합하여 관리함으로써 SDL 모델 및 OPNET 모델 시뮬레이션의 실행을 제어한다. 즉, 코-시뮬레이션 제어 코드에 하나의 메인 함수가 존재하고 Esa_Execute_Until() 함수와 SDL_Execute() 함수를 호출하여 각각 OPNET과 Tau의 시뮬레이션을 제어한다.

코-시뮬레이션 제어 코드의 역할은 OPNET 시스템과 SDL 시스템의 초기화 및 종료, OPNET 이벤트들과 SDL/Tau 이벤트들의 스케줄링, OPNET 및 SDL 시뮬레이션 간 시간 동기화, OPNET과 SDL 간의 패킷 전달 및 패킷 포맷 변환 등이다. 그림 5는 SDL-OPNET 코-시뮬레이션의 스케줄링 알고리즘을 나타낸 것이다. SDL_RQ, SDL_TQ 및 OPNET_Q는 각각 SDL 대기 큐(ready queue), SDL 타이머 큐 및 OPNET 이벤트 큐를 의미하며, n(Q)는 큐 Q의 크기, ev(Q)와 evtime(Q)는 각각 큐 Q에서 다음에 실행될 이벤트와 그 이벤트의 실행시간을 얻는 함수이다.

```
Algorithm SDL-OPNET Co-simulation Scheduling
  Initialize OPNET system;
  Initialize SDL system;
  while (n(OPNET_Q)+n(SDL_RQ)+n(SDL_TQ)>0)
    if (evtime(OPNET_Q)>evtime(SDL_RQ, SDL_TQ))
      Run SDL Unit Simulation with
          ev(SDL_RQ, SDL_TQ);
    else
      Run OPNET Unit Simulation with ev(OPNET_Q);
    endif
  endwhile;
  Terminate Co-simulation;
```

그림 5. SDL-OPNET 코-시뮬레이션 스케줄링 알고리즘
Fig. 5. SDL-OPNET co-simulation scheduling algorithm

그림 5의 스케줄링 알고리즘에서 OPNET 시스템과 SDL 시스템의 초기화를 통해 초기 이벤트들이 생성되며 모든 이벤트 큐가 비워질 때까지 시뮬레이션이 수행된다. 스케줄링 알고리즘에 따라 이벤트 큐에서 가장 빠른 수행시간을 갖는 이벤트가 결정되면 해당 모델에서 이 이벤트에 의한 단위 시뮬레이션이 수행된다. 시뮬레이션 수행으로 새로운 이벤트가 생성되면 각 이벤트 큐는 이를 자동으로 반영하고 갱신되어, 스케줄링 알고리즘의 다음 이벤트 결정에 사용된다.

4. 코-시뮬레이션을 통한 성능시험 예

4.1 InRes 프로토콜 코-시뮬레이션 시스템 구조

InRes 프로토콜은 Initiator 및 Responder 프로토콜로 구성된 비대칭 접속기반(connection-oriented) 데이터 전송 프로토콜로 간단한 데이터 송수신 프로토콜이지만 접속제어 및 오류제어 기능을 충실히 갖추고 있어 통신프로토콜공학 연구에 샘플 프로토콜로 자주 사용되는 시험용 프로토콜이다[9]. 그림 6은 Initiator 노드와 Responder 노드로 이루어진 InRes 프로토콜의 구성을 보여준다. Initiator 노드는 Initiator 프로토콜과 이를 이용하여 통신을 수행하는 Initiator User로 구성되며, Responder 노드 역시 Responder 프로토콜과 Responder User로 구성된다. 본 논문에서는 Initiator 프로토콜을 대상으로 SDL 모델을 설계 및 검증하고 제안된 SDL-OPNET 코-시뮬레이션 기법을 통해 성능평가를 수행한다.

InRes 프로토콜 코-시뮬레이션 시스템의 구조는 그림 7과 같다. 그림 7에서 OPNET의 Initiator 모듈은 외부 접속을 위한 Esys 모듈로 작성되며, Initiator 모듈의 ESD 명세는 상위 Initiator User와 하위 link 연결에 대해 코-

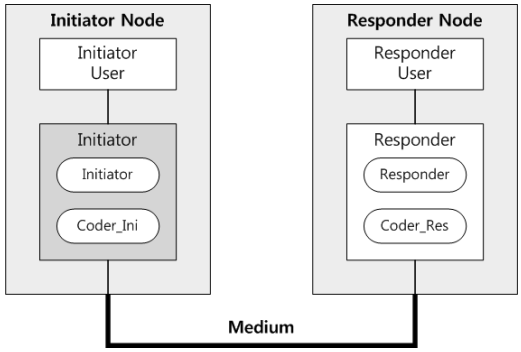

그림 6. InRes 프로토콜 시스템 구성
Fig. 6. InRes protocol system construction

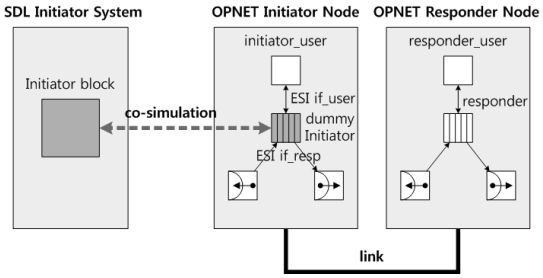

그림 7. InRes 프로토콜 코-시뮬레이션 시스템 구조
Fig. 7. InRes protocol co-simulation system structure

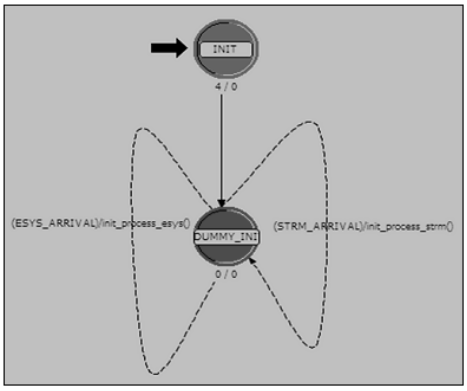

그림 8. Initiator Esys 모듈의 상태 천이 다이어그램
Fig. 8. State transition diagram of Initiator Esys module

시뮬레이션을 위한 두 개의 ESI, if_user와 if_resp을 가진다. if_user 인터페이스는 Initiator User와 SDL Initiator 간 패킷 전송을 위해 사용되고 if_resp 인터페이스는 SDL Initiator와 Responder 간 패킷 전송에 사용된다. Initiator 모듈을 제외한 Initiator 노드와 Responder 노드 및 노드 간 링크에 대한 OPNET 모델은 InRes 프로토콜 명세에 따라 설계되며, Initiator 프로토콜의 실제 모델은 SDL 모델로 SDL Initiator 시스템 내에 설계된다.

4.2 InRes 프로토콜 코-시뮬레이션 시스템 설계

InRes 프로토콜에 대한 SDL-OPNET 코-시뮬레이션 시스템 설계의 주요 항목은 다음과 같다. 첫째, OPNET 시스템에서 다른 프로토콜로부터 Initiator Esys 모듈로 들어오는 패킷을 ESI에 설정하고 반대로 ESI로부터 패킷을 가져와 다른 프로토콜로 보내도록 Initiator Esys 모듈을 설계하는 것, 둘째, SDL 모델과 OPNET과의 메시지 교환을 위해 Tau 환경 함수와 패킷 변환 함수를 설계하는 것, 마지막으로 이벤트 스케줄링을 포함한 코-시뮬레이션

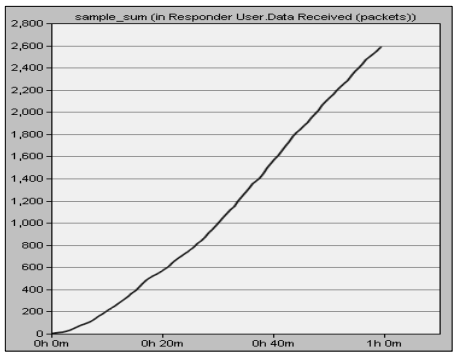

그림 9. 코-시뮬레이션 결과(누적 수신 패킷 수)
Fig. 9. A co-simulation result(the accumulated number of received packets)

의 제어를 수행하는 메인 함수를 설계하는 것이다.

Initiator Esys 모듈과 ESI 간의 패킷교환은 상태 천이 다이어그램을 포함한 Esys 모듈의 설계를 통해 구현된다. Initiator Esys 모듈의 상태 천이 다이어그램은 그림 8과 같이 단일 적색 상태와, Esys 모듈과 ESI 간의 각 방향 패킷 전송을 처리하기 위한 함수 init_process_esys()와 init_process_strm()를 각각 호출하는 두 개의 천이로 구성된다. 두 패킷 전송 함수는 op_esys 커널 프로시저를 이용하여 구현된다.

코-시뮬레이션 제어 코드는 그림 5의 스케줄링 알고리즘에 따라 SDL 및 OPNET 시뮬레이션을 수행하는 코-시뮬레이션 메인 루프 함수와 OPNET 패킷을 받아 SDL 메시지로 변환하여 보내는 ESA Callback 함수들 및 SDL 출력 메시지를 OPNET 패킷으로 변환하여 OPNET ESI에 보내는 SDL 환경함수 등으로 구성된다.

4.3 InRes 프로토콜 코-시뮬레이션 수행 및 결과

InRes 프로토콜에 대한 SDL-OPNET 코-시뮬레이션 시나리오는 다음과 같다. Initiator User로부터의 Initiator와 Responder 노드 간 연결 요청 및 연결 해제 요청과 Initiator User로부터 Initiator에게 주어지는 데이터 전송 요청의 발생 간격(inter-arrival time)은 각각 λ 값이 20, 10, 30인 exponential 분포에 따라 랜덤하게 결정되며 OPNET 시뮬레이션 시간은 1시간이다.

코-시뮬레이션을 통해 SDL Initiator 모델와 OPNET Responder 모델 간 연결 설정과 데이터 전송은 원활하게 이루어졌으며, 그림 9의 시뮬레이션 결과와 같이 Responder User는 1시간 동안 Initiator User가 보낸 약 2,600개의 패킷을 성공적으로 받았음을 확인하였다.

4.4 코-시뮬레이션 시스템 설계 시 고려사항

본 논문에서 제안하는 통합 설계 기법에 따라 검증된 SDL 모델의 성능 평가를 수행하기 위해 SDL-OPNET 코-시뮬레이션 시스템을 설계하고자 할 때 염두에 두어야 할 것은 SDL-OPNET 코-시뮬레이션 시스템 설계의 난이도가 OPNET 모델의 설계 방법과 연관된다는 점이다. SDL 모델의 경우 프로세스 모델 간 정보 교환 수단으로 프로세스를 연결하는 채널을 통한 시그널 전송을 사용한다. 즉, SDL 시스템은 모델 간의 메시지 포맷만 일치시키면 상호 정보 교환에 무리가 없는 분산시스템 환경을 충실히 따르고 있다. 하지만, OPNET 모델의 경우에는 모델 설계 시 모델 간 정보를 전달하기 위해 패킷 스트림 인터럽트 외에도 다양한 종류의 인터럽트를 필요에 따라 사용할 수 있다. 또한, OPNET에서 제공하는 프로토콜 모델은 표준 프로토콜 구조를 충실히 따르기 보다는 성능 평가 수행을 목적으로 특화시켜 설계된 모델이기 때문에 표준 패킷 포맷을 따르지 않거나 표준 프로토콜 스택을 단순화시켜 구성하기도 한다.

따라서 SDL 프로토콜 모델을 성능 평가하기 위한 OPNET 환경 모델로 기존 OPNET 모델을 사용할 경우, OPNET 모델의 모델 간 패킷 전달 방법이 복잡하지 않은지, 패킷 포맷이 SDL 표준 모델과 얼마나 다른 지 등을 확인해야 한다. OPNET 프로토콜 모델 설계 방식이 매우 복잡하고 표준 모델과 차이가 많이 날 경우, 제안된 통합 설계 기법을 적용하기 위해서는 코-시뮬레이션 시스템 설계 비용이 많이 들 수 있다. 실제로 OPNET UMTS 모델의 경우[10], UMTS 표준 프로토콜 스택과 패킷 포맷을 충실히 따르지 않고, 교차 계층 간 메시지 교환의 구현을 위해 원격 프로세스 인터럽트를 사용하는 등의 문제로 인해 특정 프로토콜이나 노드를 대상으로 SDL- OPNET 코-시뮬레이션을 수행하는 데 어려움이 있다.

5. 결론 및 토의

본 논문은 단일 모델을 기반으로 기능 검증 및 성능 평가를 수행하는 통합 설계 방식의 프로토콜 개발을 위해 SDL로 설계된 기능 모델의 성능평가 환경을 OPNET 모델로 구성하여 SDL-OPNET 코-시뮬레이션을 통해 설계된 SDL 모델의 성능 평가를 수행하는 방법을 제시하였다. 제안된 방법은 Tau와 OPNET 도구를 함께 사용하고 OPNET의 성능 평가 기능을 사용할 수 있는 장점이 있으므로 SDL과 OPNET에 모두 능숙한 설계자가 Tau의 기능 검증 기능과 OPNET의 성능 평가 기능을 직접 연계하고자 할 경우에 적극적으로 사용할 수 있는 방법이다. 이 때 기능 검증된 SDL 모델에 대해 별도로 OPNET 모델을 구성하지 않고 성능평가를 수행하므로 성능 평가 결과의 신뢰성 향상과 함께 프로토콜 설계 및 개발 기간의 단축을 기대할 수 있다.

SDL-OPNET 코-시뮬레이션을 위해서는 Tau를 통한 SDL 모델 시뮬레이션과 OPNET 모델의 시뮬레이션을 통합하여 제어하는 코-시뮬레이션 시스템의 설계를 필요로 하기 때문에 이 시스템의 설계 비용이 제안 기법의 효용성과 밀접한 관련이 있다. OPNET은 SDL과 비슷한 계층 구조 및 EFSM 방식의 설계를 지원하지만 프로토콜 간 정보 전송을 위해 보다 다양한 인터럽트를 사용할 수 있기 때문에 SDL 모델과 연결되는 기존 모델의 설계 복잡도에 따라 코-시뮬레이션 시스템의 설계 비용이 달라진다. 따라서 패킷 스트림 인터럽트 외 다른 정보 전송 방법을 사용하는 OPNET 모델과의 용이한 연결 방법에 대해 추후 연구가 필요하다.

보다 많은 프로토콜 설계 및 개발자들이 SDL-OPNET 코-시뮬레이션을 이용한 통합 설계 기법을 이용할 수 있도록 실제적인 도움을 줄 수 있기 위해서는 제안 기법을 다양한 통신 프로토콜에 적용하여 제안 기법의 신뢰성과 안정성을 확보하는 것이 매우 중요하다. 이를 위해 현재 ATM 시그널링 프로토콜인 SSCOP의 SDL 프로토콜 모델[11]과 3GPP LTE 시스템[12]을 대상으로 SDL-OPNET 코-시뮬레이션 시스템 설계를 통한 추가 연구를 수행하고 있다.

참 고 문 헌

1. ITU, Specification and Description Language, ITU-T Recommendation Z.100, 2000.
2. IBM Co. Ltd., Rational TAU SDL Suite, Ver. 6.3, 2009. See http://www-01.ibm.com/software/awdtools/tau/.
3. M. Butow, M. Mestern, C. Schapiro, and P.S. Kritzinger. “Performance modelling with the formal specification language sdl”. In IFIP TC6/6.1 International Conference on Formal Description Techniques IX / Protocol Specification, Testing and Verification XVI, volume 69, pages 213--228. Kluwer, 1996.
4. Jörg Hintelmann, Richard Hofmann, Frank Lemmen, Andreas Mitschele-Thiel, Bruno Müller-Clostermann, “Applying techniques and tools for the performance engineering of SDL systems”, Computer Networks, Elsevier, 2001.
5. Wei Monin, Fabrice Dubois, Daniel Vincent, and Pierre

Combes, “Looking for Better Integration of Design and Performance Engineering”, SDL 2003, Lecture Notes in Computer Science 2708, pp. 1-17, Springer-Verlag Berlin, 2003.
6. J. Martins, et al., “Integrating Performance Evaluation and Formal Specification”, In Proceedings of IEEE ICC ’96, pp. 1803-1807, 1996.
7. T. Kuhn, A. Geraldy, R. Gotzhein, F. Rothländer, “ns+ SDL - The Network Simulator for SDL Systems”, SDL 2005 - Model Driven, Lecture Notes in Computer Science 3530, pp. 103-116, Springer, 2005.
8. OPNET Technology Inc., OPNET Modeler. See http://www.opnet.com.
9. Hogrefe, “OSI formal specification case study: the InRes protocol and service”, revised, report No. IAM-91-012, update May 1992, University of Berne, May 1992.
10. 3GPP, Universal Mobile Telecommunications System (UMTS). See http://www.3gpp.org/article/umts.
11. ITU, B-ISDN ATM Adaptation Layer - Service Specific Connection Oriented Protocol (SSCOP), ITU-T Recommendation Q.2110, 1994.
12. 3GPP, UTRA-UTRAN Long Term Evolution (LTE) and 3GPP System Architecture Evolution (SAE). See http://www.3gpp.org/Highlights/LTE/LTE.htm.

양 치 평 (saintwind@kumoh.ac.kr)

2001 B.S. Dept. of Telecommunication Eng., Jilin Univresity, China
2006 금오공과대학교 컴퓨터공학과 석사
2006～현재 금오공과대학교 컴퓨터공학과 박사과정

관심분야 : 프로토콜 공학, SDL, 시뮬레이션, 모바일 네트워크

김 태 형 (simul@software.korea.ac.kr)

1992 연세대학교 전자공학과 학사
1995 연세대학교 전기전자공학과 석사
2001 연세대학교 전기전자공학과 공학박사
2002～현재 금오공과대학교 부교수

관심분야 : 프로토콜 공학, 모바일 네트워크, 형식기법, CASE