논문 2010-05-09

통합 모델 설계 방식 기반 통신 프로토콜 개발을 위한 SDL-OPNET 모델 변환 기법

(SDL-OPNET Model Conversion Technique for the Development of Communication Protocols with an Integrated Model Design Approach)

김 재 우, 김 태 형*
(Jae-Woo Kim, Tae-Hyong Kim)

Abstract : Although both functional verification and performance evaluation are necessary for the development of effective and reliable communication systems, they have been often performed independently; by functional modeling with formal language tools and by performance modeling with professional network performance evaluation tools, respectively. Separate and repeated modeling of one system, however, would often result in cost increase and inconsistency between the models. This paper proposes an integrated model design approach in order to overcome this problem that evaluates the performance of a communication protocol designed in SDL with SDL-OPNET model conversion. The proposed technique generates OPNET skeleton code from Tau-generated C code of the SDL model by analyzing the relations between SDL and OPNET models. IEEE 802.2 LLC protocol was used as an example of model conversion to show the applicability and effectiveness of the proposed technique.

Keywords : Functional verification, Performance evaluation, SDL, OPNET, Model conversion

Ⅰ. 서론

통신 프로토콜은 일반적으로 프로토콜 요구사항으로부터 프로토콜 명세를 작성하고 이 명세를 따르는 프로토콜 프로토타입을 설계한 뒤 최종적으로 프로토콜 제품을 완성하는 과정을 통해 개발된다. 이러한 개발 과정의 각 단계에서 구성된 프로토콜을 대상으로 이전 단계의 요구 사항에 대한 일치여부를 확인하는 시험이나 검증을 수행하고 그 결과를 개발과정에 피드백하여 프로토콜 제품의 정확성과 신뢰성을 확보하게 된다.

프로토콜 제품 개발 시 핵심적인 요구사항은 두 가지로 정리할 수 있다. 첫째는 기능 정확성으로 시스템의 구조 및 동작, 서비스 간의 관계 등 기능 부분에 대한 설계의 정확성이 만족되어야 한다. 둘째는 성능 정확성, 즉 시스템의 설계 요구사항에 부합하는 통계적 성능 요구사항이 만족되어야 한다. 프로토콜 명세는 이러한 기능 및 성능 요구 사항을 정확히 기술하며, 개발된 프로토콜이 이러한 요구사항을 만족시키는지를 검증하기 위하여 형식 언어 및 기능 검증 도구를 통한 기능 검증과 함께 수학적 분석, 실제 성능 측정, 또는 시뮬레이션을 통한 성능 평가를 수행한다. 성능 평가 측면에서는 최근 시스템의 복잡성 및 네트워크 환경의 다양성이 증가함에 따라 정확성과 시험용이성을 함께 만족시킬 수 있는 프로토콜 모델 구현을 통한 시뮬레이션 기법이 널리 사용되고 있다. 그런데 주목할 점은 기능 검증과 성능 검증에 모두 모델 설계를 통한 분석 방법이 주로 사용되고 있지만, 기능 검증 모델과 성능 검증 모델은 대개의 경우 독립적인 모델링 언어와 검증 도구를 사용하여 개별적으로 진행되고 있

* 교신저자(Corresponding Author)
논문접수 : 2010. 05. 17., 수정일 : 2010. 06 . 02.,
채택확정 : 2010. 06. 13.
김재우, 김태형 : 금오공과대학교 컴퓨터공학부
※ 이 논문은 2006년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2006-331-D00471).

다는 점이다. 즉, 기능 모델링 및 검증에는 표준 기술 언어인 SDL(Specification and Description Language)[1]을 사용하는 SDL 설계 도구가 주로 사용되고, 성능 모델링 및 검증에는 자체 모델링 기술을 사용하는 OPNET Modeler(이하 OPNET)[2]나 ns-2[3]와 같은 전문 성능 시험 도구가 주로 사용된다. 이는 지금까지 기능 검증과 성능 검증이 개별적으로 연구되어왔다는 사실을 보여주고 있으며, 실제로 프로토콜 개발 시 기능 검증과 성능 검증 모두를 충실히 수행하는 경우는 흔하지 않다. 기능 검증과 성능 검증을 모두 수행하기 위해서는 동일한 프로토콜에 대하여 중복하여 개별적인 모델링을 수행하고 검증해야 하고, 중복 설계 모델 간의 동일성을 보장해야 하기 때문이다.

프로토콜의 기능 모델과 성능 모델은 서로 다른 목적을 위해 설계되고 설계 대상도 차이가 있지만 프로토콜의 실행 흐름 정보를 공유하기 때문에 상당 부분 서로 중첩될 수밖에 없다. 최근 이러한 점을 고려하여 기능 검증과 성능 검증을 일관성 있는 모델 설계를 통해 연계시키는 통합 설계 기법이 주목받고 있다. 즉, 기능 모델에 성능 명세를 추가하고 기능 검증 도구를 이용하여 성능평가를 함께 수행하는 방법[4], 먼저 시스템의 기능 모델을 설계하고 기능 모델을 성능 모델로 변환하는 방법[5], 기능 모델을 사용할 수 있는 별도의 성능 평가 시스템을 구축하는 방법[6,7] 등이 그 예이다. 지금까지 연구된 통합 설계 기법들은 모델 설계를 위해 기존 형식언어를 그대로 사용하느냐 아니면 보다 확장된 구문을 사용하느냐에 따라, 또한 기존 성능평가 도구를 이용하느냐, 자체 성능평가 도구를 사용하느냐에 따라 분류될 수 있다. 그런데 기존 형식언어 구문을 확장하는 경우나 자체 성능평가 도구를 사용하는 경우는 범용성과 신뢰성의 저하로 사용자를 확보하는 것이 쉽지 않기 때문에 표준 형식언어와 기존의 검증된 성능평가도구를 사용하려는 시도가 늘고 있다. 기존 성능 평가 도구를 이용하는 방법은 다시 기능 검증 도구와 성능평가 도구를 연결시키는 도구연결 기법과 기능 모델을 성능 모델로 변환하는 모델매핑 기법으로 나눌 수 있는데 표준 형식언어와 도구결합 기법을 사용하는 통합설계 기법으로는 SDL 도구인 IBM Rational Tau(이하 Tau)[8]와 네트워크 성능평가 도구인 ns-2를 연계하는 기법[9]이 제안된 바 있다. 모델매핑 기법을 사용하는 통합 설계 기법으로는 SDL 구문을 성능평가 명세를 기술할 수 있도록 확장하여 이를 전문 성능평가 도구 모델로 변환하는 방법[10]이 제안된 바 있으나 구체적 변환 방법보다는 SDL 구문 확장에 주안점을 둔 방법으로, 표준 형식언어와 모델매핑 기법을 사용하는 통합설계 기법은 아직 구체적인 연구결과가 없다.

본 논문은 통합설계 방식을 통한 프로토콜 개발 방법으로 통신 프로토콜의 표준 기술 언어인 SDL과 가장 신뢰성 있는 네트워크 성능평가도구로 알려진 OPNET 간의 프로토콜 모델 변환을 수행하는 SDL-OPNET 모델 변환 기법을 제안한다. OPNET 모델은 성능평가를 위한 모델이지만 계층적 설계 구조와 EFSM(Extended Finite State Machine) 형식의 기능 설계 방식을 가진다는 점에서 SDL 모델과 많은 유사점을 갖는다. 본 논문은 SDL-OPNET 모델 간 비교 분석을 통해 모델 변환 접근 방법을 결정하고 자동화와 효율성을 고려한 SDL-OPNET 모델 변환 기법을 제시한다.

본 논문의 구성은 다음과 같다. 2절에서 SDL과 OPNET 모델 간의 비교분석을 통해 SDL-OPNET 모델 변환 접근 방법을 도출하고, 3절에서 제안하는 OPNET-SDL 모델 변환 기법을 자세히 설명한다. 4절에서는 제안하는 변환기법을 이용한 프로토콜 통합 설계 예를 보이고 5절에서 결론을 맺는다.

II. SDL 및 OPNET 설계 모델 분석

1. SDL과 OPNET 설계 모델 비교

통신 프로토콜과 같은 분산 시스템 설계를 위한 SDL 모델과 네트워크 시스템 성능시험을 위한 OPNET 모델은 설계 시 상호 유사점이 많이 있지만 세부적인 부분에서는 적잖은 차이가 있다. 따라서 SDL과 OPNET 모델 및 설계 방법의 차이를 정확히 파악하는 것은 SDL 모델로부터 OPNET 모델을 얻는 데 매우 중요한 과정이다.

먼저, SDL 모델과 OPNET 모델은 계층구조로 설계된다는 공통점을 갖지만 일반적인 블록구조의 계층을 갖는 SDL 모델과는 달리 OPNET 모델은 지리적 네트워크를 설계하는 네트워크 모델, 프로토콜 모듈 스택으로 구성된 노드 모델 등 네트워크에 특화된 계층 구조를 갖는다. 또한 두 모델은 그래픽 기반 모델을 지원하여 모델 설계를 용이하게 한다는 장점을 공유하지만, SDL 모델은 그래픽 모델과 텍스트 모델이 완벽히 동일하게 대응되는 구조로 두 모델 간 상호변환이 용이하지만, OPNET 모델은 그래픽 모델을 컴파일하여 C 언어로 된 텍스트 코

드를 구성하는 방식으로 텍스트 코드로부터 그래픽 모델을 구성하기 어렵다는 점에서 차이가 있다.

두 모델은 구체적인 모델 설계를 위한 설계 도구 측면에서도 세부적인 차이를 보인다. 두 모델 모두 동작 기술을 위해 EFSM 기반의 프로세스 모델을 사용하지만, 입력 큐 측면에서는 SDL 프로세스 모델이 내재된(implicit) 무한 큐를 사용하는데 반해 OPNET의 경우는 입력 큐를 사용하는 큐 모델과 사용하지 않는 기본 프로세스 모델로 나뉜다. 또한 OPNET 모델은 일반적인 상태머신의 상태와는 달리 녹색상태와 적색상태라는 이종의 상태를 가지고 있으며 상태천이도 입력기반이 아닌 조건기반으로 되어있는 등 일반적인 EFSM 구문을 확장시킨 형식으로 프로세스 모델을 설계한다. 링크의 설계 측면에서는, SDL 모델에서 모델 간 모든 채널이 단순한 메시지 전송 역할만을 하는 데 반해 OPNET 모델에서는 노드를 연결하는 물리 링크에 실제 물리적 특성을 부여할 수 있도록 설계된다.

특별히, SDL 모델이 오직 타이머를 이용해 시스템의 시간을 진행시키지만 OPNET 모델은 시스템 클록을 갱신하는 사건기반 시뮬레이션 기술에 기반하고 있어 보다 다양하고 세부적인 시뮬레이션 환경을 지원한다. 즉, 검증을 목적으로 하는 SDL과는 달리 OPNET은 성능 시뮬레이션을 위해 다양한 커널함수를 직접 모델에 사용하는 방식으로 시뮬레이션 기반의 모델을 구성하고 그 통계적 결과를 보여주는데 특화되어있다. 표 1은 SDL과 OPNET의 설계 모델 간의 주요한 차이점을 정리한 것이다.

표 1. SDL과 OPNET 설계 모델 비교
Table 1. Comparison between SDL and OPNET design models

구분		SDL	OPNET
설계방식	설계대상	분산시스템	통신네트워크
	계층설계	시스템-블록*-프로세스	프로젝트-노드-프로세스
	그래픽-텍스트 변환	쌍방향 (상호변환)	단방향 (컴파일방식)
설계도구	프로세스 입력	무한 큐	일반/큐 모델
	상태 설계	단일 상태	적/녹 상태
	상태 천이	입력 기반	조건 기반
	동작기술위치	천이	상태, 천이
	메시지 전송	시그널	패킷, ICI
	링크 설계	가상접속	물리매체
	시간 관리	타이머	클록 갱신
시뮬레이션	인터럽트	타이머,시그널	스트림,셀프 등
	시뮬레이션설정	모델 외부	모델내부,외부
	통계정보수집	미지원	지원

*블록은 다중 계층으로 설계될 수 있음

2. SDL-OPNET 모델변환 접근방법

SDL과 OPNET 모델간 모델 매핑은 본 논문에서 고려하고 있는 기능 및 성능 검증을 위한 통합 설계의 핵심적인 부분이다. 그러나 SDL 모델로부터 OPNET 모델로의 모델 변환을 위해서는 앞에서 살펴본 설계 모델 간의 차이점이 극복되어야 한다. 특별히 네트워크 시스템 성능평가에 특화되어 있는 OPNET은 다양한 기능의 모델 설계 도구를 지원하여 훨씬 넓은 설계 자유도를 갖는다. 따라서 SDL 모델로부터 OPNET 모델로의 매핑은 다양한 방식으로 이루어질 수 있으며 최적의 상응 모델을 결정하는 것은 설계의 목적과 환경에 따라 달라질 수 있다. 한편 효율적 기능 및 성능 검증을 위한 통합 설계는 일반적으로 네트워크 시스템 전체가 아닌 하나의 프로토콜을 대상으로 한다는 점도 모델 매핑 시 고려되어야 한다. 본 논문에서는 단일 프로토콜을 대상으로 한 SDL-OPNET 모델 매핑 접근방법으로 변환우선방법, 이해우선방법, 및 성능우선방법의 세 방법을 고려한다.

변환우선방법은 모델 변환의 용이성을 최우선으로 한 모델 매핑 방법이다. 매핑 대상인 프로토콜의 기술 시 사용되는 EFSM 모델을 예로 들면 SDL의 상태는 적색상태로, SDL의 분기점은 녹색상태로 매핑하는 방법으로 SDL의 천이 함수들을 가장 손쉽게 변환할 수 있다는 장점을 갖는다. 반면 SDL 모델에서 EFSM의 천이가 복잡하게 설계되었을 경우 다수의 녹색상태로 인해 OPNET의 EFSM이 복잡하고 거대해지는 문제점이 발생한다. 이해우선방법은 모델 매핑 시 SDL의 설계 구조를 최대한 보존함으로써 모델의 외형과 설계자의 이해도를 높이고자 하는 방법이다. 프로토콜의 EFSM 매핑 시 OPNET의 녹색상태를 사용하지 않고 SDL의 상태와 일대일 대응된 적색상태로만 OPNET의 EFSM을 설계하는 경우가 이 방법에 해당된다. 그러나 설계 일관성과 거시적 이해도가 높아지는 반면 천이 함수들의 매핑이 복잡해지는 등 변환규칙의 복잡도가 높아지는 문제가 발생할 가능성이 높다.

위의 두 방법의 경우, 변환 또는 이해의 용이성을 우선하다보니 모델 매핑을 통해 OPNET 모델 설계의 특성을 살려 OPNET 도구에 특화시킨 모델을 얻기는 어렵다. 따라서 위 두 방법으로 설계된 모델은 OPNET이 제공하는 설계 도구를 충분히 활용하여 설계된 모델에 비해 설계 효율성이 떨어질 수 있다. 성능우선방법은 시뮬레이션의 효율성을 향

상시킬 수 있도록 모델 매핑 시 OPNET의 설계 특성을 최대로 고려하는 접근방법이다. 하지만 이 방법은 다양한 OPNET의 특성을 고려하여 변환규칙을 생성해야 하므로 변환과 변환과정의 자동화에 비용이 많이 들 수 있다는 단점이 있다. 본 논문에서는 모델 변환이 가장 당면한 목적이므로 위 세 가지 접근 방법 중 변환 용이성을 위한 변환우선방법을 기본으로 하되, 녹색 상태가 과다하게 생성되는 것을 막기 위해 녹색 상태를 생성하지 않을 경우 변환 과정에 많은 비용이 드는 경우로만 녹색상태 생성을 제한하는 방법을 사용한다.

한편, 본 논문에서는 SDL 설계 도구 중에서 가장 널리 사용되는 Tau를 SDL-OPNET 모델 변환에 사용한다. OPNET 모델의 기본 설계 언어는 C인데 Tau의 SDL 컴파일러가 SDL 모델로부터 C 코드를 생성하므로 Tau가 모델 변환에 적합한 SDL 도구라 할 수 있다.

III. OPNET-SDL 모델 변환 기법

1. SDL-OPNET 모델변환 과정

본 논문에서 제안하는, SDL 모델로부터 OPNET 모델로의 모델변환 과정의 흐름도는 그림 1과 같다.

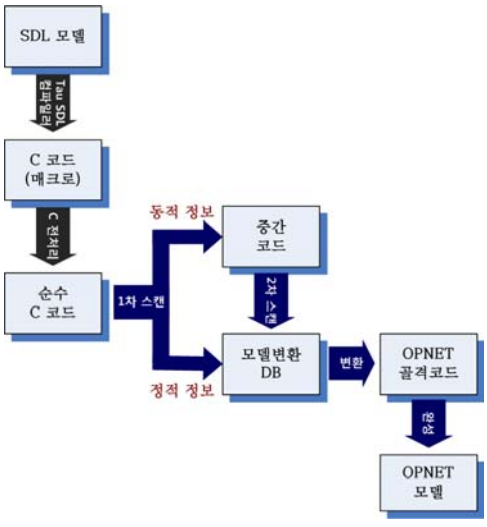

그림 1. SDL-OPNET 모델변환 과정
Fig. 1. SDL-OPNET model conversion process

SDL 모델로 설계된 시스템으로부터 Tau의 SDL 컴파일러에 의해 C 코드가 생성된다. 이 컴파일된 C 코드는 매크로로 기술된 부분이 전체 코드 중 상당 부분을 차지하여 해석과 변환에 어려움이 있으므로 C 전처리기를 통해 매크로가 제거된 순수 C코드로 변환한다. 이 코드는 OPNET 모델 구성에 필요한 정보의 추출을 위해 자체 설계한 구문해석기(lexical analyzer)를 통해 정적정보와 동적정보로 재단된 후, 정적정보는 모델변환 DB 형태로, 동적정보는 중간코드로 저장된다. 중간코드 형태의 동적정보는 2차 구문해석 처리를 통해 정리되어 모델변환 DB에 저장된다. 이 모델변환 DB는 OPNET 프로세스 모델의 EFSM 구조 및 천이 함수 변환코드를 담고 있으며, OPNET 도구로의 매핑을 통해 OPNET 모델의 골격(skeleton)을 생성한다. 여기에 성능 시뮬레이션 및 통계자료 수집 등에 필요한 OPNET 시뮬레이션 설정 과정을 거쳐 최종 OPNET 모델을 얻게 된다.

2. Tau 생성 코드 구문해석기 설계

모델변환 과정에서 핵심적인 부분으로 C 코드로부터 두 단계에 걸쳐 모델변환 DB를 생성하는 구문해석기의 설계는 Tau의 SDL 컴파일러가 생성하는 C 코드의 구조분석에 기초한다. SDL 모델로부터 Tau가 생성하는 C 코드는 시스템, 블록, 프로세스의 계층 구조, 데이터 구조, 채널 및 시그널, 프로세스 내부 상태 및 변수 선언 정보들을 계층적으로 담고 있는 선언부와 선언된 시스템, 블록, 프로세스, 채널 등의 계층 구조 설정 값, 상태 배열, 입력 배열, 천이 동작 함수 등 세부 동작 정보를 담고 있는 처리부로 구성된다.

그림 1에서 1차 스캔을 통한 구문 해석은 Tau 생성 C 코드에서 주로 선언부에서 담고 있는 정보가 기록되어 있는 위치를 탐색하여 해당 정보를 모델 변환 DB에 저장하고 처리부에 존재하는 천이 동작 함수들의 정보를 수집하여 중간코드를 생성한다. 2차 스캔을 통한 구문 해석은 대상 OPNET 프로세스 모델의 EFSM 구조의 생성과 직접 관련된 부분으로 2.2절에서 다룬 모델변환 접근 방법에 따라 구문해석기의 설계가 달라진다.

제안하는 모델 변환 기법은 기본적으로 변환우선방법을 사용하지만 OPNET 프로세스 모델의 EFSM 구조가 지나치게 복잡해지는 것을 막기 위하여 녹색상태의 생성 조건을 SDL 프로세스의 천이에서 조건 분기 시 분기된 천이들의 다음 상태가 동일하지 않은 조건 분기의 경우로 한정한다. 그림 2는 SDL 프로세스 모델 P에 대해 Tau가 생성한 C 코드 $Tau(P)$로부터 OPNET 프로세스 모델의 상태

천이 머신 (S_R, S_G, T)를 생성하는 알고리즘을 보여준다. 여기서 S_R은 적색상태 집합, S_G은 녹색상태 집합, $T=S_R \cup S_G \times S_R \cup S_G \times P \times A$는 천이 집합을 의미한다(단, P는 조건 집합, A는 동작함수 집합). 알고리즘은 깊이우선탐색(depth-first search) 방법으로 천이 내의 조건문을 찾고 분기 후 다음 상태의 동일 여부를 확인하여 녹색 상태를 생성한다. 알고리즘에서 *visited*(v), *next_state*(v,p), *generate_action_function*(s,s',p)은 각각 노드 v의 방문여부, 노드 v로부터 조건 p 이후의 다음 상태, 상태 s로부터 조건 p 이후의 다음 상태 s'까지 천이의 동작 함수를 결정하는 함수이다.

```
Algorithm Generate_OPNET_STD (Tau(P))
  S_R ← ∅, S_G ← ∅, T ← ∅; /* initialize OPNET STD */
  p ← NULL;  /* initialize condition */
  for each state s ∈ S  /* S = the state set of P */
    S_R ← S_R ∪ s, s_p ← S_R;  /* initialization */
    for each input i ∈ I  /* I = the input set of P */
      v ← s, p ← i;  /* initialization */
      repeat
        v ← DFS_for_next_branch(v, p);
        if (visited(v)) then  /* when popped */
          if ((C(v)=∅)∧(s_n∉S_G)) then /* first revisit */
            C(v) ← C(v) ∪ s_n;
            P(v) ← P(v) ∪ p;
          else if ((s_n∉C(v)∨(s_n∈S_G)) then
            S_G ← S_G ∪ v;
            for each element p in P(v)
              s_n ← next_state(v, p);
              f ← generate_action_function(v, s_n, p);
              T ← T ∪ (v, s_n, p, f);
            endfor
            s_n ← v;  /* green state saved */
          endif
        else if (v ∈ S) then /* when next state visited */
          s_n ← v;
          if (s_p ∈ (S_R∪S_G)) then
            f ← generate_action_function(s_p, v, p);
            T ← T ∪ (s_p, v, p, f);
          endif
        else  /* when new branch visited */
          C(v) ← ∅, P(v) ← ∅;  /* initialization */
        endif
      until (all branch from (s, i) is visited);
    endfor
  endfor
```

그림 2. OPNET 상태 천이 머신 생성 알고리즘
Fig. 2. State transition machine generation algorithm for an OPNET model

한편, 중간 코드의 구문을 분석하는 2차 스캔에서는 그림 2의 알고리즘에 따라 OPNET의 상태 천이 머신을 생성하고, 결정된 상태 정보와 천이의 동작 함수 정보를 모델변환 DB에 저장한다.

3. OPNET 골격코드의 생성 방법

SDL-OPNET 모델변환 과정에서 모델변환 DB로부터 OPNET 골격코드로의 변환은 모델 간 분석된 매핑 관계에 따라 수행된다. 그림 3은 모델변환 DB로부터 OPNET 골격코드의 매핑관계를 보여준다.

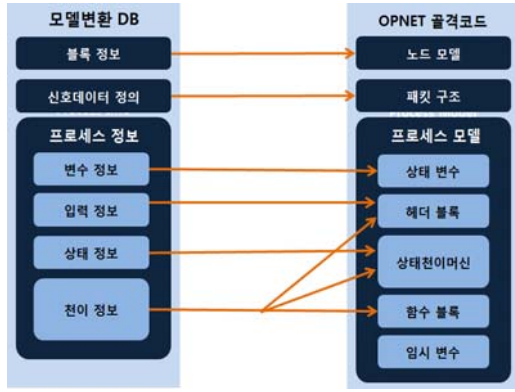

그림 3. 모델변환 DB로부터 OPNET 골격코드로의 매핑 관계
Fig. 3. The mapping relation from model conversion DB to OPNET skeleton code

모델변환 DB의 정적정보인 SDL 모델의 블록 정보 및 신호의 매개변수 데이터 정의는 각각 OPNET의 네트워크/노드모델 및 패킷 모델과 같은 정적 모델을 구성하는데 사용된다. 일반적으로 통합 설계의 대상은 개별 프로토콜이므로, 모델 변환의 주 대상은 모델변환 DB의 동적정보인 프로토콜의 프로세스 모델이다. SDL 프로세스 모델은 내재된 무한 입력 큐를 갖기 때문에 모델 변환 시 역시 무한 입력 큐를 갖는 큐 모델을 OPNET 프로세스 모델로 선택해야 하고, 패킷 스트림 인터럽트 발생 시 항상 패킷을 큐에 저장하고 입력 패킷은 항상 큐로부터 추출하는 방식으로 모델 변환이 이루어진다.

OPNET 프로세스 모델 내의 상태변수, 헤더 블록, 함수 블록 및 상태 천이 다이어그램 내 상태 블록에 입력되는 코드는 상태 천이 다이어그램 구성 정보와 함께 표 2의 OPNET 골격코드 매핑규칙에 의해 자동으로 생성된다. 표 2의 매핑 규칙에서 동

표 2. 모델변환 DB로부터 OPNET 골격코드로의 주요 매핑 규칙

Table 2. The Main mapping rules from model conversion DB to OPNET skeleton code

DB정보{변수명}		매핑규칙
프로세스	input {$1,$2…}	·헤더 블록에 다음 코드 생성 enum {$1, $2, …}

천이[index]	start state{$1}
	input{$2}
		.헤더 블록에 다음 코드 생성
		#define $1_$3 ₩
		((op_intrpt_type() == ₩
		OPC_INTRPT_STRM) && ₩
		(sig_value == $3))
		//input이 타이머일 경우
		#define $3_EXPIRED 상수
		#define $1_TIMER ₩
		(op_intrpt_type() == ₩
		OPC_INTRPT_SELF) && ₩
		(op_intrpt_code()==
		$3_EXPIRED)
		.적색 상태(EXIT)에 다음 코드 입력
		if (op_intrpt_type() == ₩
		OPC_INTRPT_STRM){ pkptr =₩
		op_pk_get(op_intrpt_strm());
		op_pk_nfd_get_int32(pkptr,"id",₩
		(int *)&(sig_value);
		}

	predicate {$1}	·녹색 상태에 다음 코드 생성
		if ($1)
		condition == TRUE;
		else
		condition == FALSE;

	action : /* $1 */ {$2}	·함수 블록에 다음 코드 생성
		static void action_index_common() {$1*}
		static void action_index()
		{
		Packet *pkptr;
		FIN(action_index);
		pkptr = op_pk_create_fmt ₩
		("packet_format");
		$2*
		op_pk_send(pkptr, ₩
		STRM_MACRO);
		FOUT;
		}

*일부 동작코드 변환은 세부적인 변환 규칙에 따름

표 3. OPNET 골격코드의 구조

Table 3. Structure of OPNET skeleton code

구분 (항목)		내용	형식
헤더파일		데이터, 시그널 구조	C
패킷구조		패킷 필드 정보	텍스트
프로세스 정보	EFSM	OPNET 상태천이머신	텍스트*
	헤더블록	인덱스, 천이조건 등	C
	상태변수	지역 변수 등	C
	함수블록	각 천이의 동작 함수	C
시뮬레이션 정보		시뮬레이션설정 주석	텍스트

*OPNET 상태 천이 다이어그램으로 변환 가능

작코드의 세부 변환 규칙에 포함되는 것은 패킷 필드의 조작 및 타이머 관련 스케줄링 등에 관한 OPNET 커널 프로시저 입력으로 자세한 내용은 지면 제약으로 생략한다. 표 3은 모델변환 DB로부터 생성된 OPNET 골격코드의 구조를 보여준다. 표 2의 OPNET 골격코드 매핑규칙에 따라 자동 생성된 코드는 C 언어 형식이며, 그림 2의 OPNET 상태천이 머신 생성 알고리즘에 따라 생성된 EFSM과 패킷구조는 텍스트 형식이다. OPNET 골격코드의 시뮬레이션 정보는 SDL 모델 내 주석문의 형태로 기술되는 시뮬레이션 환경설정 정보를 의미한다.

마지막으로 OPNET 골격코드에 시뮬레이션 모델 및 환경 설정 정보 즉, 성능지표 통계자료 수집을 위한 커널 함수 추가, 시뮬레이션 시나리오 설정, Tau 생성 함수 및 헤더파일의 연결 과정을 거쳐 성능 시뮬레이션을 실행할 수 있는 최종 시스템이 완성된다. OPNET은 상업용 도구로 모델의 데이터 파일의 구조를 공개하고 있지 않기 때문에, 자체 편집기를 이용해 설계해야 하는 노드 모델, 패킷 구조, 상태 천이 다이어그램의 생성 등은 모델변환 DB의 정보로부터 사용자가 직접 완성하여야 한다. 표 4는 제안하는 변환 기법의 OPNET 모델 생성 자동화 수준을 정리한 것이다. OPNET 패킷모델과 상태천이 다이어그램은 골격코드의 해당정보를 그대로 모사하여 완성하고, 그 외 코드 수준의 OPNET 프로세스 모델에는 골격코드를 그대로 사용할 수 있다. 시뮬레이션 모델은 SDL 기능 모델의 영역이 아니므로 사용자가 성능 평가 목적에 맞게 추가로 설계하여 완성한다.

표 4. OPNET 모델 생성의 자동화 수준

Table 4. Automation level of OPNET model generation

구분 (항목)		수준	변환 정보
패킷모델		모사	골격코드의 패킷구조
프로세스 모델	EFSM	모사	골격코드의 EFSM
	상태함수	복사	골격코드의 매핑규칙
	헤더블록	복사	골격코드의 헤더블록
	상태변수	복사	골격코드의 상태변수
	함수블록	복사	골격코드의 함수블록
시뮬레이션모델	커널함수	설계	요구된 수집통계정보
	모델설정	참조	골격코드의 시뮬레이션 정보

4. 변환 모델 간 동일성 문제

제안된 SDL-OPNET 모델 변환 기법은 SDL 모

델로부터 변환 과정을 통해 직접 OPNET 모델을 생성하기 때문에 SDL 모델과는 별도로 OPNET 모델을 생성하는 경우와 달리 두 모델 간의 동일성을 유지시킬 수 있다는 장점을 갖는다. 특별히 신뢰성 있는 SDL-to-C 컴파일러인 Tau를 사용하고, 생성된 C 코드를 직접 OPNET 모델에 적용함으로써 생성된 OPNET 모델의 신뢰성을 높이고 있다. 따라서 여기서는 제안 기법에서 사용하는 OPNET 상태천이 머신 생성 알고리즘, 프로세스 모듈 유형 선택 방법 및 OPNET 골격코드 변환 규칙에 의한 모델 불일치 가능성에 대해서만 간략히 검토한다.

제안된 OPNET 상태 천이 머신 알고리즘은 단일 유형의 상태를 사용하는 SDL 프로세스 상태천이 모델로부터 적색 상태와 녹색 상태를 사용하는 OPNET 프로세스 상태천이 모델을 얻는 알고리즘이다. OPNET의 녹색 상태는 인터럽트를 기다리지 않고 천이 가능한 OPNET 프로세스 모델 고유의 구문으로 SDL 프로세스 모델의 천이 내 조건문에 의한 분기점에 대응될 수 있다. 이때 II-2절의 SDL-OPNET 모델변환 접근방법에서 설명한 바와 같이 다양한 변환 접근방법을 생각할 수 있으나 제안 알고리즘은 모델변환 용이성을 고려하면서도 모델의 이해 용이성을 위해 녹색 상태를 필요한 수만으로 제한하여 생성한다. 즉, 제안 알고리즘은 SDL 프로세스 모델의 천이 내 분기점 중 녹색 상태로 변환할 곳을 선택하는 알고리즘으로 상태 천이 모델의 기능에는 영향을 주지 않는다.

SDL 프로세스 구조로부터 OPNET 큐 프로세스 모듈 구조로의 매핑은 III-3절에서 설명한 바와 같이 무한 입력 큐를 가지며 자동으로 큐 입출력을 처리하는 SDL 프로세스의 특성에 따른 것이다. 즉, OPNET 프로세스 모델과 SDL 프로세스 모델의 동일성을 위해서 OPNET 프로세스 모델로 큐 프로세스 모델을 선택하고 패킷스트림 인터럽트가 발생할 때 패킷을 큐에 저장하며 입력 패킷은 큐로부터 추출하는 변환과정을 수행한다. OPNET 골격코드의 매핑 규칙도 천이 실행을 위해 시그널 및 타이머 인터럽트만을 갖는 SDL 프로세스의 특성을 고려하여 이에 상응하는 스트림 인터럽트와 셀프 인터럽트만을 사용하도록 하고 있다. 이외의 변환규칙 역시 SDL 구문과 일대일로 대응 가능한 코드로의 매핑을 규정하고 있기 때문에 변환 모델의 기능에는 변화가 없다. 따라서 제안 기법에 의해 변환된 모델이 원 모델과 기능적으로 일치하지 않는 문제는 수동 입력 시의 오류를 제외하고는 발생하지 않는다고 할 수 있다.

V. SDL-OPNET 모델변환을 통한 통합설계 예

1. LLC 프로토콜의 모델변환을 통한 성능평가

LLC(Logical Link Control) 프로토콜은 IEEE 802.2 표준 랜 프로토콜로 데이터링크 계층의 상위 부계층(sublayer)에 위치한다[11]. LLC 프로토콜 표준은 무응답형 비연결성 서비스, 접속 기반 서비스, 응답형 비연결성 서비스 등의 동작 모드를 정의하고 있는데, 본 논문에서는 흐름 제어 기법을 제공하는 LLC 접속 기반 서비스를 대상으로 제안하는 통합 설계 기법을 적용한다. LLC 접속 기반 서비스는 데이터 전송 시 흐름제어를 위해 슬라이딩 윈도우를 사용하는 Go-Back-N ARQ(Automatic Repeat Request) 또는 Selective Reject ARQ 기법을 사용할 수 있다[12]. 본 통합 설계 예에서는 LLC 접속 기반 서비스 프로토콜에 대해 송신부와 수신부를 담당하는 두 개의 블록 LLC_Sender와 LLC_Responder를 SDL 모델로 설계하였다. 그림 4는 LLC 프로토콜 블록 외에 LLC 프로토콜의 기능 검증을 위한 테스트 블록과 간소화된 MAC(Medium Access Control) 블록을 포함하는 SDL 시스템 다이어그램을 보여준다.

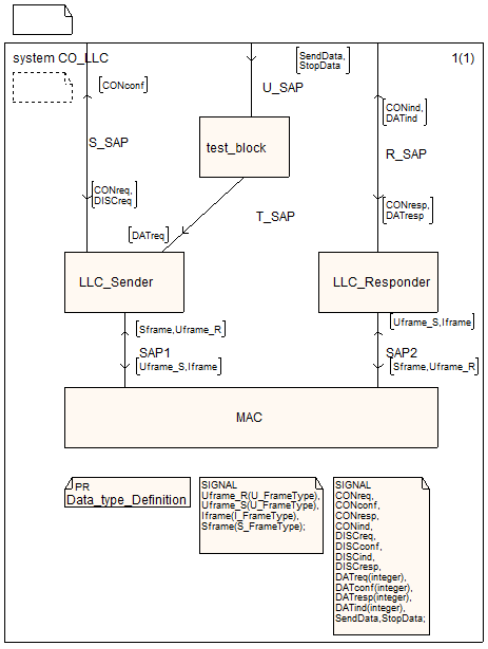

그림 4. LLC 프로토콜의 SDL 시스템 다이어그램
Fig. 4. SDL system diagram of LLC protocol

LLC 프로토콜 서비스 명세는 LLC_Sender 블록 내 LLC_S 프로세스로 설계된다. LLC_S 프로세스의 상태 천이 다이어그램은 그림 5와 같이 접속 설정 단계와 데이터 전송을 처리 4개의 상태를 갖는다. LLC_Responder 블록은 3개의 상태를 갖는 LLC_R 프로세스를 갖는다.

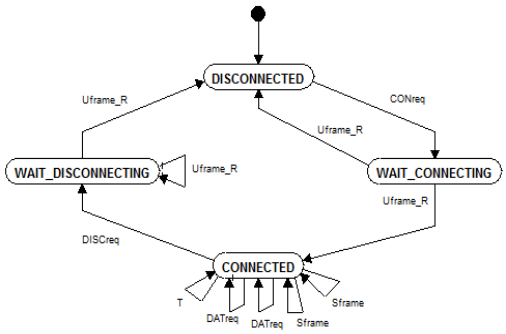

그림 5. LLC_S 프로세스 모델의 상태 천이 다이어그램

Fig. 5. State transition diagram of LLC_S process model

본 통합 설계 예에서는 LLC 프로토콜의 SDL 모델을 제안하는 모델 변환 기법을 이용하여 OPNET 프로세스 모델로 변환하고 성능 평가 시뮬레이션 환경을 구성한 다음 LLC 프로토콜이 지원하는 Go-Back-N ARQ 흐름제어 기법의 성능을 평가한다. 그림 6은 LLC 프로토콜을 사용하는 OPNET 송신 노드 모델로 SDL LLC_S 프로세스 모델로부터 변환되는 llc_sender 프로세스 모델과 접속 관리 및 패킷 생성 제어를 위한 user1 프로세스 모델, 성능평가를 위해 간단하게 설계된 mac 프로세스 모델로 구성된다.

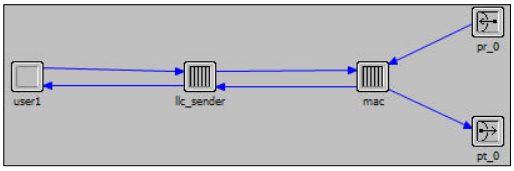

그림 6. LLC 프로토콜 송신 노드 모델

Fig. 6. LLC protocol sender node model

그림 7은 SDL LLC_S 프로세스 모델의 Tau 생성 C 코드에 그림 2의 OPNET 상태 천이 다이어그램 생성 알고리즘을 적용하여 생성한 llc_sender 프로세스의 상태 천이 다이어그램으로 LLC_S 프로세스 모델의 상태와 일대일 매칭 되는 4개의 적색상태와 새로 생성된 2개의 녹색 상태로 구성된다. SDL LLC_Responser 블록의 LLC_S 프로세스 모델은 3개의 적색 상태와 2개의 녹색 상태를 갖는 llc_responder로 변환되어 OPNET 수신 노드 모델을 구성한다.

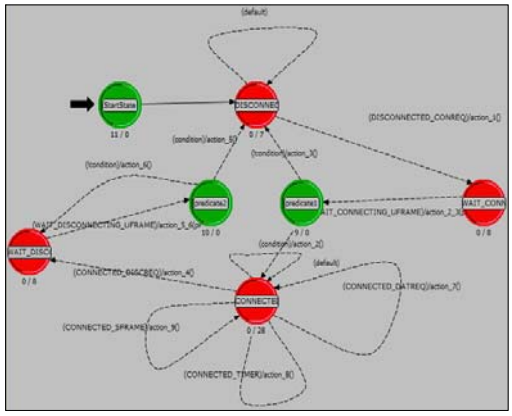

그림 7. OPNET llc_sender 프로세스 모델의 상태 천이 다이어그램

Fig. 7. State transition diagram of OPNET llc_sender process model

SDL LLC 프로토콜 모델로부터 Tau를 통해 생성된 코드는 1, 2차 스캔 과정을 거쳐 모델 변환 DB를 구성하고 표 2의 매핑 규칙을 통해 OPNET 골격 코드 즉, llc_sender 및 llc_responder 프로세스 모델의 헤더 블록과 상태 코드, 함수 블록 등으로 매핑 된다. LLC 프로토콜에서 지원하는 Go-Back-N ARQ 흐름 제어 기법의 성능 평가를 위해 LLC 프로토콜 송신 노드와 수신 노드를 연결하는 네트워크 모델을 구성하고, 성능 평가 지표인 패킷 오류율 및 링크 이용률(utilization) 수집을 위한 커널 함수를 추가하고 시뮬레이션 시나리오에 따라 네트워크 환경 값을 설정한다. 마지막으로 천이 동작 함수로 변환된 Tau 생성 코드의 실행을 위해 Tau 라이브러리 헤더 파일을 연결하여 시뮬레이션 시스템을 완성한다.

시뮬레이션 환경 설정 값은 실제 랜 환경에 맞추어 다음과 같이 설정하였다. 데이터 패킷(I-프레임)의 크기는 8000 비트, 제어 패킷(S-프레임)의 크기는 36 비트이고 링크의 데이터 전송 속도는 100Mbps, 슬라이딩 윈도우 크기는 7, 두 노드 간

전파 시간(propagation time)은 4×10^{-6}초, 비트 오류율은 2.5×10^{-7}이다. 따라서 상수 a=전파시간/패킷전송시간=0.05이고, 패킷 오류율의 이론값은 비트 오류가 독립동등분포일 때 대략 0.002가 된다. 시뮬레이션 시간을 10분으로 하여 시뮬레이션을 수행한 결과 링크 이용률과 패킷 오류율의 시간 평균값은 그림 8과 같이 각각 0.9788 및 0.00108 정도로 수렴한다. Go-Back-N ARQ의 전송성공 패킷을 기준으로 한 링크 이용률은 이론 상 $W\geq 2a+1$일 때 $(1-p)/(1+2ap)$이므로(단, W=윈도우 크기, p=패킷오류율)[12], 시뮬레이션 결과인 0.9788은 패킷 오류율 0.001에 대한 이론적 수치인 0.9989에 근접한다고 할 수 있다.

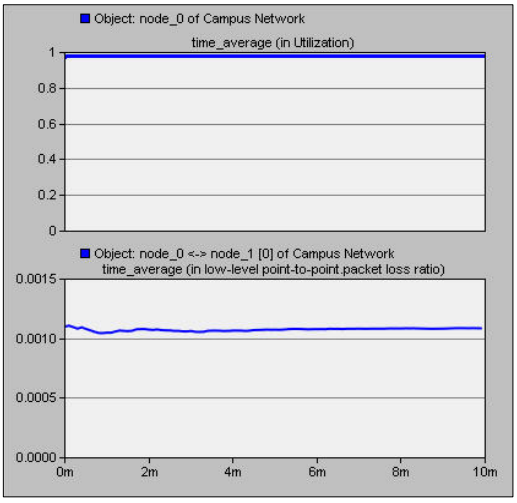

그림 8. 시뮬레이션 결과: 링크 이용률 및 패킷 오류율

Fig. 8. Simulation results: link utilization and packet error rate

2. 통합 설계 시 고려사항

본 논문에서 제안하는 SDL-OPNET 모델 변환 기법을 통한 통신 프로토콜 통합 설계 방식을 사용하고자 할 때 제안 기법의 효용성을 높이기 위해서 다음 사항들을 참고하는 것이 좋다. 먼저 SDL-OPNET 변환 용이성 및 효율성을 위해 SDL 모델 설계 시 천이 분리를 위한 커넥터(connector) 사용을 최소화하고 루프나 분기의 재 연결점 결정에 주의를 기울이는 게 좋다. 위의 구문은 Tau 생성 C 코드에서 라벨과 goto 문으로 변환되는 데 연결 지점과 모델 변환 시 생성되는 녹색 상태 위치에 따라 자동 생성되는 천이의 동작 함수의 안정성과 효율성을 떨어뜨릴 수 있다.

또한, SDL 모델의 성능 평가는 전적으로 변환된 OPNET 모델의 성능 평가 환경 설정에 따라 이루어지나, SDL 모델 설계 시 어느 부분에 어떤 성능 지표를 평가해야 한다든지 하는 성능 평가를 위한 정보를 OPNET 변환 모델에 넘겨주어 성능 평가 환경 설정에 도움을 줄 수 있다. 즉, Tau에서 제공하는 인라인 C 코드 기능을 이용하여 '//'로 시작하는 한줄 주석으로 성능 평가 정보를 기록하면 Tau의 생성 코드에 반영되어 천이의 동작 함수에 입력되므로 성능 시뮬레이션 환경 설정 시 참고할 수 있다.

한편, SDL 모델의 경우 프로세스 모델 간 정보 교환 수단으로 프로세스를 연결하는 채널을 통한 시그널 전송을 사용하지만 OPNET 모델의 경우에는 모델 설계 시 모델 간 정보를 전달하기 위해 패킷 스트림 인터럽트 외에도 다양한 종류의 인터럽트를 필요에 따라 사용할 수 있다. 따라서 SDL 프로토콜 모델로부터 변환된 OPNET 모델을 기존 OPNET 모듈과 연결하여 시뮬레이션 모델을 구성할 경우, 연결되는 OPNET 모듈의 정보전달 방식에 맞추어 SDL 모델로부터 변환된 OPNET 모델에 추가로 정보 전달 인터페이스 구성이 필요 할 수 있다.

VI. 결론

본 논문은 단일 모델을 기반으로 기능 검증 및 성능 평가를 수행하는 통합 설계 방식의 프로토콜 개발을 위해 SDL로 설계된 기능 모델을 OPNET 성능 평가 모델로 변환하는 방법을 제시하였다. 제안된 방법은 SDL 모델 설계자가 성능평가를 위한 OPNET 모델을 쉽게 구성할 수 있도록 도움을 주어 프로토콜 설계 및 개발 기간을 단축시킬 수 있으리라 기대된다. 또한 SDL 모델을 사용해야 하는 OPNET 설계자에게도 SDL 모델로부터 직접 변환된 OPNET 골격코드는 OPNET 모델링을 신속하게 하는데 상당한 도움이 될 것이다.

본 제안 기법에 사용된 Tau와 OPNET은 모두 상용 도구로 기능 검증과 성능 평가 도구로서의 성능은 매우 우수하나, 프로그램 소스나 데이터 파일의 구조에 접근할 수 없고, 특히 OPNET의 경우 텍스트 코드로부터 비주얼 모델 변환을 지원하지 않아 반드시 자체 에디터를 사용해야 하기 때문에 변환과정의 자동화에 한계가 있다. 또한, OPNET은

SDL과 비슷한 계층 구조 및 EFSM 방식의 설계를 지원하지만 보다 다양하고 자유로운 설계 방식을 지원하기 때문에 변환 모델의 최적화, 변환 모델과 기존 모델과의 연결 등에 한계가 있는 것도 사실이다.

향후 제안 기법의 신뢰성과 안정성을 확보할 수 있도록 다양한 실제 통신 프로토콜에 적용하여 그 결과를 토대로 제안 기법의 부족한 부분을 보완하는 것이 가장 중요한 과제이다. 특별히 보다 많은 프로토콜 설계 및 개발자들이 통합 설계 기법을 이용할 수 있도록 실제적인 도움을 줄 수 있기 위해서는 실제 프로토콜 설계에의 성공적인 적용 결과와 함께 기존 방법과의 정확하고 다양한 비교로 제안 기법의 우수성을 보여주는 것이 매우 중요하다. 따라서 이 분야에 대해 앞으로도 지속적인 연구와 보완이 필요하다.

참고문헌

[1] ITU, Specification and Description Language, ITU-T Recommendation Z.100, 2000.

[2] OPNET Technology Inc., OPNET Modeler. See http://www.opnet.com.

[3] The Network Simulator ns-2, Information Sciences Institute, University of Southern California, See http://www.isi.edu/nsnam/ns/.

[4] Tae-Hyong Kim, et al., "SDL design and performance evaluation of a Mobility management technique for 3GPP LTE systems", LNCS, Vol. 4745, 2007.

[5] Wei Monin et al., "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] Jörg Hintelmann, et al., "Applying techniques and tools for the performance engineering of SDL systems", Computer Networks, Vol. 35, No.6, pp. 647-665, Elsevier, 2001.

[7] Nico de Wet, et al., "Using UML models for the performance analysis of network systems", Computer Networks, Vol. 49, No.5, pp. 627-642, Elsevier, 2005.

[8] IBM Co. Ltd., Rational TAU SDL Suite, Ver.6.3, 2009. See http://www-01.ibm.com/software/awdtools/tau/.

[9] T. Kuhn, et al., "ns+SDL - The Network Simulator for SDL Systems", SDL 2005 - Model Driven, Lecture Notes in Computer Science 3530, pp. 103-116, Springer, 2005.

[10] J. Martins, et al., "Integrating Performance Evaluation and Formal Specification", In Proceedings of IEEE ICC '96, pp.1803-1807, 1996.

[11] IEEE 802.2-1998 (ISO/IEC 8802-2:1998), IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 2: Logical Link Control, 1998.

[12] William Stallings, Data and Computer Communications, Eighth Edition, Pearson Eduation, 2007.

저 자 소 개

김 재 우

2004년 : 금오공과대학교 컴퓨터공학부 학사.
2006년 : 금오공과대학교 컴퓨터공학과 석사.
관심분야 : 프로토콜공학, 모바일네트워크.
Email : eva0191@kumoh.ac.kr

김 태 형

1992년 : 연세대 전자공학과 학사.
1995년 : 연세대 전기전자공학과 석사.
2001년 : 연세대 전기전자공학과 박사.
현재, 금오공과대학교 부교수.
관심분야 : 프로토콜 공학, 모바일네트워크, 형식기법, CASE.
Email : taehyong@kumoh.ac.kr