논문 98-23-12-27 한국통신학회논문지 '98-12 Vol. 23 No.12

적합성 시험을 위한 시험 스위트의 오류 발견 능력 분석 방법

정회원 김 광 현,* 허 기 택**

The Fault Coverage Analysis Method of Test Suite for Conformance Testing

GwangHyun Kim*, GiTaek Hur** *Regular Members*

요 약

통신 프로토콜의 적합성 시험은 상호 운용성과 효율적인 컴퓨터 통신을 위해서 매우 중요하다. 지금까지 시험 스위트의 생성에 관한 연구는 주로 시험항목의 크기를 최소화하는 방법에 집중되어 왔다. 그러나 최근에는 최대의 오류발견 능력을 갖는 시험항목 생성에 관한 연구가 진행되고 있다.

본 논문에서는 최대의 오류 발견 능력을 갖는 시험항목 생성을 위해 프로토콜 구현에서 발생할 수 있는 오류를 네 가지 형태로 분류하였고, 네 가지 오류 모델에 따른 오류 발견 방법을 제안한다. 시험 스위트의 오류 발견 능력 분석을 위해 기존 평가 방법의 문제점을 지적하였고, 오류 발견 능력 분석 시뮬레이터를 구현하였다. 이를 이용하여 Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜에 적용한 결과를 보였다. Inres와 B-ISDN UNI SSCS 프로토콜에 적용 결과, 오류 모델 1과 3은 100%의 높은 오류 발견율을 보였다. 오류 발견 능력 분석 시뮬레이터는 다양한 프로토콜에 적용함으로써 새로운 오류발견 능력 분석 도구로 사용될 수 있다.

ABSTRACT

Protocol conformance testing is crucial to interoperability and effective computer communication. so far, research about the test case generation has focused on minimizing the test case length. Recently, the research has advanced the test case generation with high fault coverage capability. In this paper, I will classify four types of fault models that compromise protocol implementation and present a fault detection method in accordance with the fault model. I pointed out the weaknesses of the existing method for fault coverage analysis and implemented a simulator for fault coverage analysis. In addition, I showed an application example for Inres protocol and B-ISDN UNI SSCS protocol. Analysis results for Inres protocol and B-ISDN UNI SSCS protocol, fault model 1 and 3, have a high fault coverage of 100%. This simulator is widely used with new fault coverage analysis tools by applying it to various protocols.

I. 서 론

정보통신 제품들 간의 원활한 서비스 제공을 위해서는 정보통신 제품의 표준화, 적합성 시험 및 상호 운용성 보장이 필수적이다. 프로토콜을 구현함에 있어 이미 표준안이 제시된 프로토콜에 대해서도 서로 다른 형태의 독립된 구현 제품들이 가능하다. 따라서 서로 다른 제품들 간의 상호 운용성

*광주대학교 컴퓨터학과(E-mail:ghkim@hosim.kwangju.ac.kr), **동신대학교 컴퓨터학과(E-mail:gthur@dongshinu.ac.kr)
논문번호 : 98335-0803, 접수일자 : 1998년 8월 3일
※본 논문은 1997년도 정보통신부 대학기초연구지원사업의 연구 결과입니다.

을 보장하기 위한 사전 절차로써 적합성 시험이 필요하게 된다[1,2]. 프로토콜의 효율적인 개발을 위한 과정은 다음과 같이 세 단계로 나눌 수 있다.

첫 번째 검증(Verification)단계에서는 프로토콜의 명세(Specification) 자체에 특정한 오류를 갖고 있는지를 검사하는 단계이다. 두 번째로 적합성 시험(Conformance testing)단계는 주어진 프로토콜의 명세대로 구현되었는지를 검사하여 프로토콜 명세로부터 시험항목을 생성하여 구현된 프로토콜에 적용하여 오류 여부를 검사하는 과정이다. 세 번째의 상호운용성 시험(Interoperability testing)단계는 적합성 시험을 통과한 두 구현 프로토콜이 통신상의 문제가 없는가를 시험한다. 적합성 시험을 통과한 경우라도 실제로 구현 상태가 다를 수 있으므로 상호운용성 시험이 필요하다. 프로토콜 개발 과정은 그림 1과 같은 여러 단계를 거쳐서 개발되고 있다.

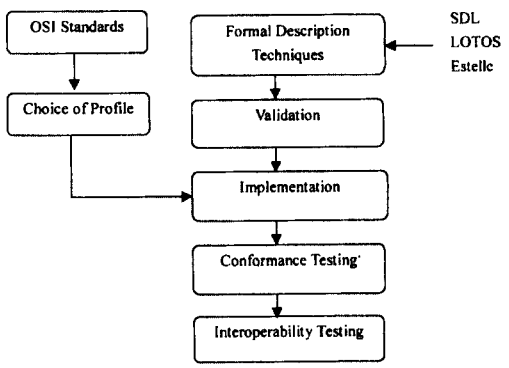

그림 1. 일반적인 프로토콜 개발 과정

새로운 프로토콜이 개발되어 실제로 사용 가능하기 위해서는 앞의 세 단계를 거쳐야 한다. 검증 단계는 프로토콜을 설계하고 개발하는 과정이고, 상호운용성 시험은 아직까지 연구가 미진한 분야이다. 그러나 적합성 시험의 경우는 어느 정도 관련 분야 연구가 활성화되어 있고 계속적인 연구가 필요한 분야이다[2,4]. 먼저 OSI(Open System Interconnection) 표준 프로토콜은 다양한 형태의 프로토콜 구현이 가능하다. 즉 다양한 프로파일을 선택하여 프로토콜을 구현하게 된다. 이렇게 여러 형태로 구현된 프로토콜이 표준에 맞게 구현되었는지를 검사하는 적합성 시험 과정이 필요하게 된다.

본 논문에서는 시험 스위트의 오류 발견 능력 분석을 위한 이론적 분석 방안을 제시하도록 한다. 시험 스위트 생성과 함께 생성된 시험항목을 구현 프로토콜에 적용하여 얼마나 많은 오류를 찾아낼 수 있는가를 평가하는 오류 발견 능력 분석이 필요하다. 시험 스위트에 대한 오류 발견 능력 평가 방법은 주로 수학적 평가 방법과 시뮬레이션 방법을 이용한 연구가 이루어져 왔다[9,11]. 본 논문에서는 기존 평가 방법의 문제점을 지적하고, 오류 모델을 사용하여 생성된 시험항목이 어느 정도 오류 발견 능력을 갖는지를 평가하는 구조적 평가 방법을 제시한다. 그리고 제안된 방법이 실제 프로토콜에 적용 가능하도록 오류 발견 능력 분석 시뮬레이터를 구현한다. 또한 구현된 시뮬레이터가 다양한 프로토콜에 적용 가능성을 살펴보기 위해 Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜에 적용한 결과를 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 적합성 시험과 시험 스위트를 생성하는 일반적인 방법을 기술하고, 3장에서는 시험 스위트의 오류 발견 능력 분석에 대한 여러 가지 방법을 서술하고자 한다. 4장에서는 본 논문에서 제안한 시험 스위트의 구조적 분석 방법을 제시하고, 구현한 오류 발견 능력 시뮬레이터에 적용한 프로토콜의 결과를 분석한다. 마지막으로 앞으로의 연구 방향과 결론을 맺도록 한다.

II. 시험 스위트 생성과 오류 발견 방법

2.1 시험 스위트 생성

적합성 시험 과정에서 가장 중요한 단계는 프로토콜 규격으로부터 시험 스위트를 도출해 내는 과정이다. 지금까지는 시험 스위트 생성은 수작업을 통해 자연어로 작성된 규격에서 시험 목적을 반영하여 각각의 시험항목을 도출하였다. 이러한 수작업으로 시험항목을 생성하게 되면 시험 수행자의

능력에 따라 시험 품질이 좌우되는 문제점이 발생한다. 따라서 시험 전 과정에서 수작업 이 최소화되는 자동화가 필요하게 된다. 표준화된 형식 기술언어(FDT:Formal Description Technique)의 사용을 기본으로 FDT가 프로토콜 공학에 핵심적인 방법론을 제공함에 따라 FDT를 기본으로 한 자동화에 대한 연구 활동이 활발하게 진행되고 있다[10,12].

표준 프로토콜로부터 추상 시험 스위트를 생성하는 과정은 적합성 시험에서 매우 중요하다. 시험 스위트 생성을 위해 형식 사양으로부터 시험 목적에 부응하는 시험 스위트가 추출되며, 시험 목적은 적합성 시험 요구 사항의 일부라고 규정하고 있다. 프로토콜의 시험항목 생성시 구현에서 잘 발생할 수 있는 오류를 고려하여 시험항목을 생성하면 오류 검출 능력을 높일 수 있는 방법이 된다. 적합성 시험을 수행하기 위해서는 표준 프로토콜 규격으로부터 시험 목적을 반영한 시험항목이 생성되어야 한다. 따라서 프로토콜의 효율적인 적합성 시험을 위해 자동적인 시험항목 생성이 필요하다.

적합성 시험을 한다는 것은 제어 관찰점을 통하여 IUT에 이벤트를 인가하고 IUT로부터 발생된 이벤트를 관찰하여 IUT의 상태를 진단하는 과정이다. IUT를 FSM이라고 해석할 때 FSM의 완벽한 테스트는 초기 상태로부터 도달 가능한 모든 상태 공간을 확인하는 것이다. 그러나 실제 프로토콜은 너무 방대하고 엔티티 사이에 다양한 상호 작용을 포함하고 있기 때문에 모든 상태를 시험한다는 것은 불가능하다. 따라서 프로토콜의 적합성 시험을 위한 시험항목 생성에 많은 시간이 필요하고 시험 목적에 부응하는 시험항목을 생성하여야 한다. 지금까지 프로토콜 적합성 시험을 위한 시험 스위트를 생성하는 여러 방법들이 제안되었다. FSM에 기초한 시험항목을 생성하는 여러 방법 중에서 UIO(Unique Input/Output) 방법이 현재 가장 많이 사용되고 있다[4,5]. UIO 방법 중에서 SUIO와 MUIO를 사용하여 시험항목의 크기를 줄인 연구결과가 나와있다. 그리고 시험 스위트의 생성과 함께 시험 스위트의 공유 및 개발에 소요되는 시간을 줄이는 점에도 관심을 갖게 되었다[6,8]. 이러한 요구를 충족시키기 위하여 ISO에서는 시험 스위트 표기와 모호성을 배제한 형식 기술 언어인 TTCN(Tree and Tabular Combined Notation)을 개발하였다[1,13].

2.2 시험 스위트의 오류 발견 방법

생성된 시험 스위트를 구현된 프로토콜에 적용하여 보다 많은 오류를 발견하는 것이 중요하다. 기존의 시험항목을 IUT에 적용했을 때 오류를 발견하지 못하는 경우가 발생할 수 있다. 이러한 미발견 오류는 시험항목의 길이를 증가시킴으로써 발견 할 수 있다. 적합성 시험의 여러 단계 중에서 현재까지 가장 많은 연구가 이루어진 부분은 시험 스위트를 생성하는 부분이다[13,16]. 그 결과 많은 부분이 적합성 시험에 이용할 수 있게 되었다. 그러나 비용과 자동화 측면을 고려할 때 아직 미진한 상태이다. 구현된 프로토콜의 오류 발견 방법은 표준 사양으로부터 생성된 시험항목을 구현된 프로토콜에 적용하고 출력을 관찰하여 표준 사양으로부터 생성된 시험항목의 출력과 비교함으로써 오류를 찾아내게 된다. 그림 2은 생성된 시험 스위트를 구현된 프로토콜에 적용하여 IUT의 오류 발견 과정을 보여주고 있다.

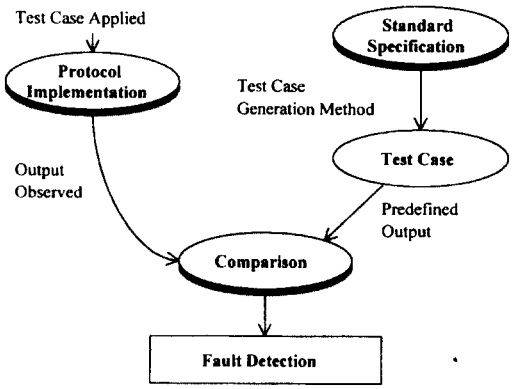

그림 2. IUT의 오류 발견 과정

지금까지 시험항목 생성에 관한 연구는 주로 시험항목의 크기를 최소화하는 방법에 집중되어 왔

으며, 최근에는 최대의 오류 발견 능력을 갖는 시험 항목을 생성하는 방법에 관한 연구가 진행되고 있다[8,10]. 현재까지는 주로 효율적인 시험 항목을 생성하는 방법에 관한 연구가 많았지만, 이를 정량화하여 수식으로 표현하고 분석하는 연구는 상대적으로 부족하였다.

실제 구현 프로토콜에서 UIO의 중복으로 인한 미발견 오류 조건을 정의하고 이를 발견할 수 있는 연구 결과도 나와 있다[17,18]. 오류를 미발견하는 조건을 제시하여 이 조건을 만족하는 시험항목을 생성하도록 하였는데, 이러한 방법을 사용하면 오류 발견 능력이 향상된 시험항목을 생성할 수 있다.

III. 시험 스위트의 오류 발견 능력 분석 방법

3.1 오류 모델

본 논문에서는 다양하게 발생하는 오류의 형태를 적절하게 제안하여 네 가지로 분류하고, 오류 형태에 따라 오류 발견 능력이 향상된 새로운 시험항목 생성 방법을 제안한다. 이러한 네가지 오류는 구현과정에서 발생할 수 있다는 것으로 가정한다. 따라서 다양하게 발생할 수 있는 오류를 적절한 형태로 제한함으로써 효율적인 시험 스위트 생성과 오류 발견 능력 분석에 이용될 수 있도록 한다[5,17]. 오류 모델의 종류는 다음과 같다.

가. 출력 오류

출력 오류(output faults)는 구현 프로토콜의 한 상태에 입력을 적용하여 발생되는 출력이 표준 프로토콜과 다른 출력을 발생시키는 오류이다. 구현 프로토콜(M_I)의 천이에서 발생하는 출력과 사양 머신(M_S)에서 정의하고 있는 출력이 서로 다른 경우를 말한다. 즉 $M_S\{\lambda(S_i, \alpha)\} \neq M_I\{\lambda(S_i, \alpha)\}$의 성질을 갖는 오류이며 이를 오류 모델 1로 정의한다.

나. 천이 오류

구현 프로토콜의 어떤 상태의 입력에 대한 천이가 표준에서 정의하고 있는 것과 다른 상태로 천이가 발생하는 오류이다. 구현 프로토콜(M_I)에서 발생하는 천이가 사양 머신(M_S)에서 정의하고 있는 상태로 천이가 발생하지 않고 상태함수에 의하여 다른 상태로 천이가 발생하는 오류이다. 상태 천이(transfer faults) 오류가 발생했다고 하며 이를 오류 모델 2로 정의한다.

다. 상태 병합 · 분리 오류

어떤 특정 상태와 입력에 대하여 하나 이상의 상태가 추가되거나 제거되는 경우가 발생하는 오류를 말한다. 구현 프로토콜(M_I)에서 발생하는 천이가 사양 머신(M_S)에서 정의하고 있는 상태에 대하여 하나 이상의 상태가 추가되거나 제거되는 경우이다. 한 상태에서 두 상태로 나누어지는 상태 분리 오류(state split fault)와 두 상태에서 하나의 상태로 합쳐지는 상태 병합 오류(state merge fault)가 있다. 이러한 경우의 오류 형태를 오류 모델 3으로 정의한다.

라. 혼합 오류

혼합 오류는 오류가 발생할 때 상태나 에지의 한 부분에서만 발생하지 않고 이 두 가지 형태의 오류가 동시에 발생하는 오류이다. 구현 프로토콜(M_I)에서 발생하는 오류가 사양 머신(M_S)에서 정의하고 있는 상태 천이와 출력의 한 곳에서만 발생하지 않고 이 두 가지 오류가 동시에 발생하는 경우를 말한다. 혼합 오류(hybrid faults)는 앞에서 정의한 오류 중에서 하나 이상이 복합적으로 발생하는 오류이다. 즉 출력 오류와 상태 천이 오류가 동시에 발생하는 오류이며 이를 오류 모델 4로 정의한다.

3.2 오류 발견 능력 분석 방법

표준으로 정의한 프로토콜 사양의 구현 오류에 대한 오류 발견 능력의 정확한 측정은 불가능하지만 오류 발견 능력을 평가하기 위한 다양한 방법들이 사용되고 있다. 시험항목의 오류 발견 능력 평가는 오류를 갖고 구현된 프로토콜이 생성된 시험항목으로 어느 정도까지 오류를 발견해 낼 수 있는지를 평가함으로써 적합성 시험의 효율성을 평가하는 하나의 기준으로 사용될 수 있다. 시험항목의 오

류 발견 능력의 평가 방법은 주로 수학적 분석 방법과 시뮬레이션을 이용하고 있다. 첫 번째 방법은 시험항목 생성과 관련한 여러 조건들을 고려하여 수학적인 분석을 통해 평가하는 방법이다[14,16]. 두 번째 방법은 시뮬레이션 기법을 이용하여 오류를 갖는 FSM을 생성하고 시험항목을 오류 FSM에 적용하여 통과한 FSM 수와 통과하지 못한 FSM을 계산하여 오류발견 능력을 평가하는 방법이다[14,19].

수학적인 평가 분석을 위해 통신 프로토콜의 시험되어야 할 FSM의 수가 매우 많기 때문에 모든 상태를 시험한다는 것은 거의 불가능하다. 실제로 FSM이 n개의 상태, m개의 입력, p개의 출력 상태를 갖는 프로토콜이 있으면 $(np)^m$개의 시험 상태가 존재하게 된다. 현실적으로 이렇게 많은 FSM의 상태 모두를 시험한다는 것은 불가능하다. 따라서 시험항목의 오류 발견 능력을 평가하기 위한 방법으로써 여러 조건들을 고려하여 수학적으로 평가하는 방법을 사용하고 있다. Ms를 사양 기계라 하고, TS는 테스트 스위트, 그리고 Impl(m, M_s)는 사양을 구현한 것이라고 정의하면, 이것을 기본으로 하여 오류 발견 능력을 평가하는 식은 다음과 같다.

$$FC(m,M_s,TS)=\frac{N_t(m,M_s)-N_p(m,M_s,TS)}{N_t(m,M_s)-N_c(m,M_s)}$$

- $N_t(m, M_s)$: $Impl(m, M_s)$에 있는 기계의 총 수;
- $N_c(m, M_s)$: M_s와 일치하는 (conforming) $Impl(m, M_s)$에 있는 기계의 수;
- $N_p(m, M_s, TS)$: 주어진 테스트 스위트를 통과한 $Impl(m, M_s)$에 있는 기계의 수;
- $N_t(m, M_s) - N_c(m, M_s)$: M_s와 일치하지 않는 (not-conforming) $Impl(m, M_s)$에 있는 기계의 수;
- $N_t(m, M_s) - N_p(m, M_s, TS)$: 주어진 테스트 스위트를 통과하지 못하는 $Impl(m, M_s)$ 에 있는 기계의 수;

시험 스위트의 오류 발견능력 분석을 위해 수학적인 평가 방법과 함께 Monte Carlo 시뮬레이션 기법을 사용하고 있다. 시뮬레이션 기법은 오류를 갖는 FSM을 생성하고 시험항목을 오류 FSM에 적용하여 통과한 FSM 수와 통과하지 못한 FSM을 계산하여 오류 발견 능력을 평가하는 방법이다. 이 방법은 발생 가능한 오류 형태를 10개의 클래스로 한정하여, 임의의 오류를 갖는 FSM을 생성하여 시험항목이 오류를 발견해 내는 능력을 평가하는 방법이다[11,15].

3.3 구조적 평가 방법

시험항목의 오류 발견 능력 평가 방법의 목적은 시험항목 생성시 최대의 오류 발견 능력을 갖도록 하는 생성 방법을 도입하는 것이다. 현재까지 최적의 시험항목 생성 방법은 알려져 있지 않다. 따라서 각 프로토콜의 시험 환경을 고려한 방법을 선택하여야 한다. 지금까지는 주로 시험항목의 길이를 최소화하여 적합성 시험의 효율성만을 고려하였으나, 이제는 적합성 시험의 효율성과 함께 오류 발견 능력이 최대가 되는 시험항목 생성 방법이 요구되고 있다[17,18].

본 논문에서 제안하는 구조적 평가 방법은 다양하게 발생할 수 있는 오류를 발견하기 위해서 상태 확인이 필요하다는 사실에서 출발한다. 예를 들어 그림 3에서 상태 A에서 C로 천이되는 과정에서 입력과 출력값만을 확인해서는 IUT가 올바르게 구현되었다고 말할 수 없다. 왜냐하면 잘못된 구현 상태 A에서 D를 거쳐 상태 C로 천이되는 과정에서도 입출력(p_m/i_m-p_n/n_n) 값이 올바르게 구현된 IUT와 동일한 입출력 값이 발생하기 때문이다. 따라서 이러

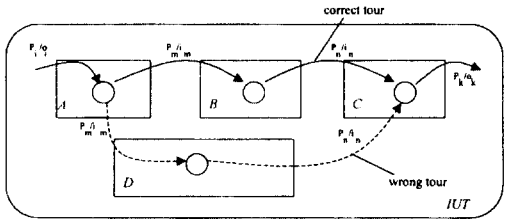

그림 3. 복합오류(coupling error) 형태

한 복합적인 오류를 발견할 수 있는 시험 스위트 생성이 요구된다. 따라서 이러한 오류를 발견할 수 있는 방법으로 시험이 진행될 때 올바른 상태에 도착했는지를 확인하는 과정이 필요하다.

여기서는 수학적 평가 방법과 시뮬레이션 평가 방법을 서로 보완한 구조적 오류 발견 평가 방법을 제시한다. 먼저 구조적 오류 평가식을 도입하기 위해 다음과 같은 정의를 이용한다.

[정의 3.1]

$\delta(S_1,\alpha)=S_i$ 이고 $\delta(S_1,\alpha x) = S_j$를 만족하면 M_s에서 천이 <S_i; i/o; S_j>는 TS에 의해 포함된다고 한다. 여기서 α, x는 시험항목의 일부분이다.

[Proof]

상태 S_1 에 입력 순서열 α를 적용했을 때 S_i 상태로 천이가 발생했고 또한 상태 S_1에서 S_j로의 천이는 상태 S_i에서 입력 순서 x를 추가하여 발생했다. 여기서 x란 입력 순서열을 i/o로 표현한 것이다. 즉 상태 S_i에서 S_j 로 천이는 초기 상태 S_1 에 입력 순서열 α를 적용한 후 다시 x(i/o)를 적용하면 된다.

[정의 3.2]

$\delta(S_1, \alpha)=S_i$ $\delta(S_1, \alpha x) = S_j$, $\delta(S_1, \beta) = S_k$ 그리고 $\lambda(\delta(S_1,\alpha x), \gamma) \neq \lambda(\delta(S_1, \beta),\gamma)$를 만족하면 Ms에서 천이 <$S_i$; i/o; S_j>의 tail 상태 S_j는 TS(Test Suite)에 의해 다른 상태 S_k와 식별된다고 한다. 이것을 Tail_Dis라고 한다. 여기서 α, β, γ는 테스트 스위트(TS)에 속하는 시험항목의 일부분이다.

[Proof]

상태 천이 함수 δ는 상태 S_i에 입력 순서 α를 적용하여 상태 S_i으로 천이가 이루어지는 것은 정의 3.1과 동일하다. $\lambda(\delta(S_1,\alpha x), \gamma) \neq \lambda(\delta(S_1, \beta), \gamma)$에서 $\delta(S_1,\alpha x)$는 S_j가 되고 여기에 입력 순서열 γ를 적용했을 때 출력 값과 $\delta(S_1, \beta)$를 수행했을 때 천이는 Sk가 되고 여기에 입력 순서열 γ를 적용했을 때의 출력 값이 서로 다르게 되면 S_j는 S_k와 서로 다른 상태에 있게 된다.

[정의 3.3]

tail 상태 S_i와 S_k를 상태 천이 <S_i; i/o; S_j>에 식별될 수 없을 때 Tail_NDis라 하고, Tail_NDis(<S_i; i/o; S_j>, TS)는 {S_1,S_2, ...,S_n} - Tail_Dis(<S_i; i/o; S_j>, TS)와 같다.

[Proof]

tail 상태 S_i와 S_k를 입력 순서열 α, β, γ에 의해 두 상태를 구별하지 못하면 동일한 상태로 인식하게 된다. 즉 $\lambda(\delta(S_1, \alpha x), \gamma) = \lambda(\delta(S_1, \beta), \gamma)$와 같은 관계를 갖게 된다. 그러므로 Tail_NDis(<S_i; i/o; S_j>, TS)는 {S_1, S_2, ..., S_n} - Tail_Dis(<S_i; i/o; S_j> ,TS)와 같게 된다.

[정리 3.3]으로 부터 손쉽게 Tail_Dis와 Tail_NDis 수를 계산할 수 있다.

- |Tail_NDis(<S_i; i/o S_j>, TS) |O| = n|O|
- $M_t\ (M_s) = |\ Impl(M_s)\ | = (n\ |O|)^{n|T|}$: 구현 기계의 전체 수.
- $N_c\ (M_s) = (n\text{-}1)!\ (n\ |O|)^{n|T|-|D_s|}$: Ms 와 적합한 (conforming) 구현 기계의 수.

수학적 평가 방법에서 FC의 값을 구하기 위해서 $N_p(M_s, TS)$의 정확한 값을 필요로 한다. 이 값을 계산하기 위해서는 TS가 사양 기계에 대해 통과하는지를 시험하여 하여야 하는데 이는 현실적으로 너무 많은 비용이 들어 불가능하다. 그러므로 N_p를 $\overline{N_p}$로 대치하여 [식3-1]과 같은 새로운 평가식을 생성한다.

$$FC_e(M_s,TS) = \frac{N_t(M_s) - \overline{N}_p(M_s,TS)}{N_t(M_s) - N_c(M_s)} \qquad (3\text{-}1)$$

[식 3-1]에서 M_s에 적합한 구현 머신($N_c\ (M_s) = (n\text{-}1)!\ (n\ |O|)^{n|T|-|D_s|}$)과 테스트 스위트를 통과한 구현 기계 중에서 큰값을 선택하면 [식 3-2]가 된다.

$$FC_e(M_s,TS) = \frac{N_t(M_s) - \max(N_c(M_s), \overline{N}_p(M_s,TS))}{N_t(M_s) - N_c(M_s)} \qquad (3\text{-}2)$$

[식 3-3]은 앞의 수학적 평가식에서 상용 대수를 취한 식으로써 적절한 오류 발견 능력의 평가식으로 사용된다.

$$FCo(m,M_s,TS)=\frac{\log(N_t(m,M_s)-N_p(m,M_s,TS))}{\log(N_t(m,M_s)-N_c(m,M_s))} \quad (3\text{-}3)$$

IV. 구조적 오류 발견 능력 분석 방법 및 적용 사례

프로토콜 시험항목에 대한 성능 평가의 두 가지 측면은 시험항목이 시험 시스템에서 얼마나 빨리 수행될 수 있는가(efficiency)와 생성된 시험항목에 의해 얼마나 많을 오류를 찾아낼 수 있는냐의 문제(effectiveness)를 다루는 것으로 구분할 수 있다. 현재까지 시험항목 생성에 관한 모든 연구는 최소의 길이를 갖는 시험항목의 생성에 집중되어 왔다. 최소 길이의 시험항목 생성은 시험 시스템에서 수행 속도(efficiency) 측면을 고려한 성능 평가라고 할 수 있다[14,18]. 그러나 이제는 수행 속도뿐만 아니라 시험항목이 구현 프로토콜에서 얼마나 많은 오류를 발견할 수 있는가(effectiveness)를 고려한 성능 평가가 요구된다. 따라서 시험항목의 오류발견 능력 분석을 통해 정량적으로 평가할 수 있는 방법이 필요하다. 본 연구에서는 최대의 오류 검출 능력을 갖는 시험항목의 생성에 관한 새로운 기준을 제공하기 위해 구조적 평가 방법을 위한 시뮬레이터 설계하고 구현하도록 한다[20,21]. 시뮬레이터의 적용 가능성을 보이기 위해 Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜에 대한 오류 발견 능력을 분석한 결과를 기술한다.

4.1 오류 발견 능력 분석 시뮬레이터의 설계 및 구현

본 논문에서는 구조적 방법을 이용한 오류 발견 능력을 분석하기 위해 3장에서 정의한 오류 모델에 적용할 수 있는 오류 발견 능력 분석 시뮬레이터를 설계, 구현하였다. 본 연구에서 구현한 시뮬레이터는 시험항목의 오류 발견 능력 분석 뿐만 아니라 FSM으로 표현된 프로토콜에 대해 효율적인 시험항목도 가능하도록 하였다. 시뮬레이터는 기본적으로 FSM으로 표현된 프로토콜을 이용하여 시험항목을 생성하는 부분과 이를 오류FSM에 적용하여 오류 발견 능력을 분석하는 부분으로 구성하였다. 이 절에서는 시뮬레이터의 구현 환경과 구현 모듈을 살펴보고, 구현한 실행 화면을 보여주고 실제 프로토콜에 적용 사례를 통해 시뮬레이터의 유용성을 보인다.

4.1.1 구현 환경과 구현 모듈

지금까지 구현된 도구들은 주로 적합성 시험의 전 과정에 이용될 수 있는 도구의 개발에 치중하였다. 따라서 시험항목의 오류 발견 능력을 평가하는 측면에서는 상당히 부족하였다. 본 시뮬레이터는 이러한 점을 고려하여 시험항목의 생성과 함께 시험항목의 오류 발견 능력을 평가하는 도구로 사용될 수 있도록 하였다. 본 연구에서 구현한 오류 발견 시뮬레이터는 SunOS 5.5에서 C언어를 사용하여 구현하였고 인터페이스는 Tcl/Tk를 이용하여 구현하였다. 시뮬레이터의 기본 구성은 FDT를 사용한 프로토콜 사양을 FSM 모델로 표현하도록 한다. 그림 4는 본 연구에서 구현한 시뮬레이터의 개괄적인 구조를 보여주고 있다.

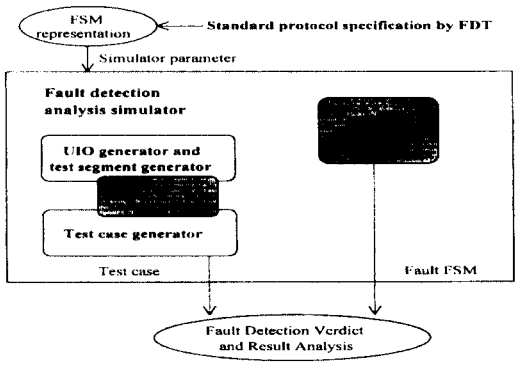

그림 4. 오류 발견 능력 분석 시뮬레이터의 구조

4.1.2 시뮬레이터의 구조 및 실행 환경

시뮬레이터는 기본적으로 FSM으로 표현된 프로토콜을 이용하여 시험항목을 생성하는 부분과 이를 오류 FSM에 적용하여 오류 발견 능력을 분석하는 부분으로 구성하였다. 본 시뮬레이터의 수행 과정은 형식 기술 기법을 사용하여 기술된 프로토콜을 FSM 형태로 변환하고 UIO 기법을 사용하여 시험항목을 생성한다. 오류 모델에 따라 생성된 오류 FSM에 적용하여 어느 정도의 오류를 발견하는지를 평가한다. 구현된 시뮬레이터에서는 시험항목이 오류 FSM 에 모델별로 적용 가능하도록 하였다. 시뮬레이터에서 구현된 프로그램의 주요 기능을 자료 구조와 함께 살펴봄으로써 어떻게 시험항목을 생성하고 오류를 발견하는지를 알 수있다.

```
/* Q에 삽입되는 노드의 자료구조 ****************/
/* gd : v-> g에서 입출력 i/o 에 의하여 천이되는 상태 */
/* Pnew : v->pnew 배열내의 모든 상태에 대하여,
   입출력 i/o 에 의해                          */
/*천이되는 상태들의 배열                       */
/* Tnew : v->tnew 배열내의 초기 상태 g에서 gd까
   지 천이되는동안의 모든 입출력 저장          */
/*Pnew가 NULL이 되면 g의 uio로써 Tnew를 결정
   할 수 있다.                                 */
/ ************************************** /

  void uio( Graph G,
        상태 g,
        상태개수 n,
        상태 g의 uio 들을 저장하기 위한 구조 N[ ] )
  {
  Q <- 초기 상태 node 구성, 삽입;
  for(cnt=0; Q가 empty 상태가 아니고 cnt < n*n*2
이면 계속; cnt++)
  {
      노드 v <- Q의 첫 노드;
for(v->g 상태에서 가능한 모든 입출력 i/o에 대하여)
      {
       입력 i/o에 대하여 v->g 상태의 천이 상태를
조사하고;
       v->g 상태에서 가능한 입출력 i/o에 대한 v-
>pnew 배열내의 모든 상태들의 천이 조사;
       v->tnew에 i/o에 추가;
     만약 pnew 가 NULL이면 tnew를 g의 uio로 결정;
     그렇지 않으면 새로운 node v를 구성;
      Q <- v;
     }
   }
   Q를 clear 한다.
  }
```

시험항목 생성을 위한 과정은 생성된 UIO를 사용하여 각 천이에 대한 테스트 세그먼트를 구하고 천이의 시험을 위한 test subsequence를 구한다. 다음은 시험항목 생성을 위해 최단거리 알고리즘을 적용하여 subsequence를 구하는 프로그램 구조이다. 시험항목의 길이를 최소화하는 방법은 overlapping을 사용하였다.

```
  int makepath(source s,
        destination d,
        s에서 d로의 shortest path 저장, 배열 path,
        path tree 내에서의 각 상태들의 level)
  {
     if (source와 destination이 같으면) return 1;
  {
  /*root를 s로 하여 구축된 shortest path tree를 이용
한다. */
     for(shortest path tree에서 s로부터 나갈 수 있는
     모든 노드를 조사)
     {
      나갈 수 있는 길이 있으면 level이 증가한 값
      으로 makepath를 recursive 호출;
      호출된 makepath에서 d를 찾았으면 현재 노
      드의 level에 해당하는 path 위치에 s를 저장;
      상위 프로그램에 path를 찾았음을 알린다
      (return 1);
     }
     source와 destination이 같으면 d의 level에 해
```

```
당하는 path내 위치에 d를 삽입;
      return 1;
    }
  char *spath(source s, destination d)
  {
      tree 구조를 clear 한다;
        root를 s로 하는 shortest path tree를 구성한다.
      /*모든
   edge는 동등 */
        s에서 d로의 path를 저장할 path 배열을 할당;
        /* 노드의 level은 shortest path tree에서의 level이다.*/
      makepath(s, d, path, 에 node의 level이 저장된 배열);
  }
```

그림 5는 UIO를 사용하여 test segment를 생성하고 이를 이용하여 시험항목을 생성하는 과정을 보여 준다. 오류 FSM의 생성은 오류 모델에 다라 다르게 생성된다. 각 오류 모델별로 100,000개의 오류 FSM을 생성하였다.

그림 6는 시험항목의 오류 평가를 위해 오류 모델에 따라 오류 FSM을 자동적으로 생성하는 부분을 보여 준다. 다음 프로그램 구조는 생성된 시험항목을 오류 FSM에 적용하여 오류를 발견하는지 못하는지를 판정하는 부분이다.

```
float TestFaultModel(조사할 시험항목 FaultModel)
{
  for(test 횟수)
  {
    시험항목에 적합한 오류 FSM 구성;
    오류 FSM에 대한 uio 생성;
   for(오류 FSM 내의 모든 천이(s->d)에 대하여)
     {
       초기 상태에서 s까지의 shortest path 조사;
s ->d의 입출력 조사;
       d의 UIO 검사;
     }
  }
}
```

그림 7은 시뮬레이터 실행의 최종단계로 오류 모델에 따라 오류를 발견하는 화면을 보여주고 있다.

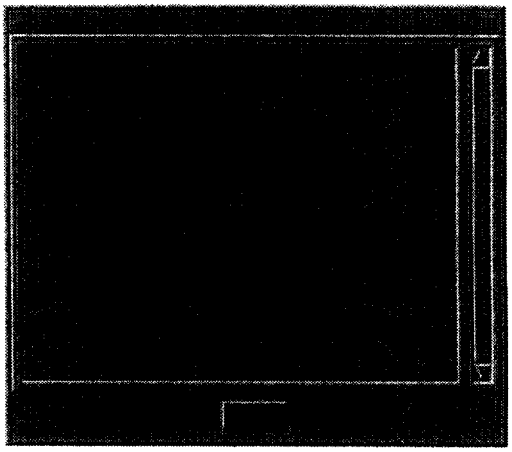

그림 5. 시뮬레이터의 시험항목 생성 화면

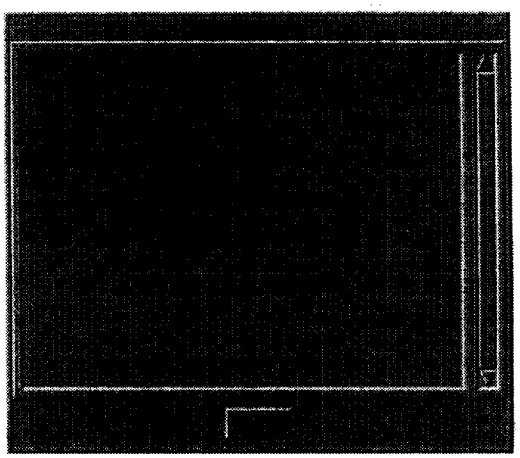

그림 6. 시뮬레이터의 오류 FSM 생성 화면

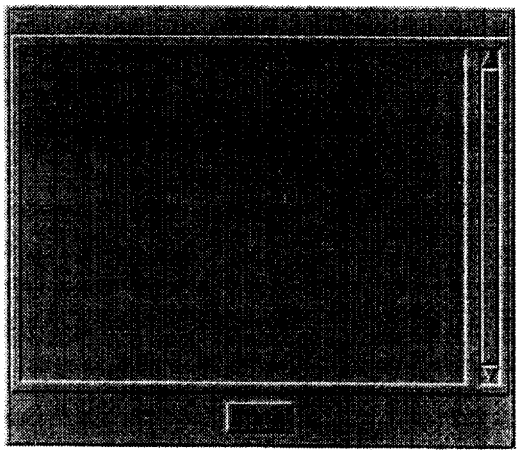

그림 7. 오류 FSM에 대한 오류 판정 화면

4.2 적용사례

시뮬레이터를 이용하여 시험항목이 어느 정도 오류 발견 능력을 갖는지를 실질적으로 평가하기 위해 실제 프로토콜에 적용하도록 한다. 기본적으로 적용될 프로토콜은 FSM으로 표현 가능한 형태로 변환한다. 원래의 프로토콜 규격을 완전하게 FSM으로 변환하는 것은 불가능하지만 현실적으로 시뮬레이터에 적용 가능성을 고려하면 FSM으로 표현하는 것이 하나의 방법이 된다. 본 연구에서도 프로토콜을 FSM으로 표현하여 시뮬레이터에 대한 입력으로 사용한다. 이 절에서는 오류 발견 능력 분석 시뮬레이터를 이용하여 3장에서 제안한 오류 모델 4가지에 대한 오류 발견 능력을 분석한다. 분석대상 프로토콜은 OSI 프로토콜과 비슷한 Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜에 대한 적용예를 보인다.

4.2.1 Inres 프로토콜

Inres 프로토콜은 OSI의 기본 개념을 다수 포함하고 있는 가상의 프로토콜로 이해하기 쉽고 크기가 크지 않으므로 여러 연구에서 참조용으로 널리 쓰이고 있는 프로토콜이다. Inres 프로토콜은 OSI 참조 모델의 어떤 특정 계층에도 해당되지 않으며 일반적 프로토콜의 한 계층을 모델링한 것이다. Inres 시스템은 두 개의 서비스와 하나의 프로토콜로 구성된다.

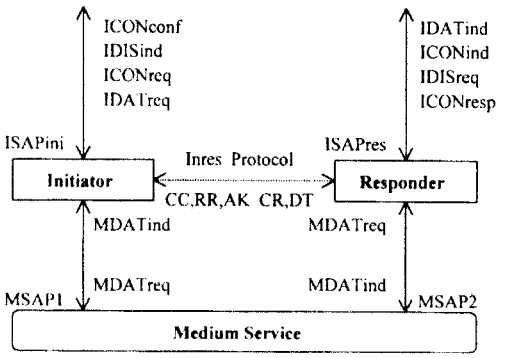

그림 8. Inres 프로토콜 구조

Inres 프로토콜은 Medium 서비스를 이용하여 상위 계층 가상의 사용자에게 Inres 서비스를 제공한다. Inres 프로토콜 구조는 그림 8과 같고, Initiator와 Responder, 두 개체간에 동작되는 연결 위주의 프로토콜이다. 이 두 개체는 CR, CC, DT, AK, DR의 5가지 PDU를 교환하면서 통신한다. 오류 모델 4가지에 대한 오류 발견 능력을 평가하기 위해 시뮬레이션에 의해 생성된 시험항목을 각각의 오류 모델에 적용하였다.

시험항목을 오류 FSM에 끝까지 적용할 때까지 출력 값이 동일하면 미발견 오류에 포함시킨다. 오류 FSM 중간에 오류가 발생되면 시험을 중단하여 시험 수행 시간을 절약하는 효과를 갖도록 하였다. 각 오류 모델별로 오류 FSM은 여러 가지 가능성을 고려하여 100,000개를 임의적으로 생성하였다. 이 정도의 오류 FSM은 발생 가능한 오류를 충분하게 포함한다. [표 1]의 시뮬레이션 결과를 보면 오류 모델 1과 3은 100%의 오류 발견율을 보이고 있다. 그러나 오류 모델 4는 오류모델 2에 비해 오류 발견

표 1. Inres 프로토콜의 오류 발견 능력 분석 결과

Fault model	Fault FSM	Detected FSM	Fault coverage(%)
1	100,000	100,000	100.0
2	100,000	99,960	99.96
3	100,000	100,000	100.00
4	100,000	98,661	98.66

율이 다소 떨어지고 있음을 알 수 있다. 이는 출력 오류와 천이 오류가 상호 결합되어 미발견 오류가 증가되기 때문이다.

4.2.2 B-ISDN UNI SSCS 프로토콜

ITU-T에서는 OSI 참조 모델을 참조하여 B-ISDN에서 사용되는 프로토콜을 권고하고 있다. B-ISDN UNI에서 기본적으로 사용되는 프로토콜은 물리 계층, ATM 계층, AAL 및 네트워크 계층으로 구성된다. B-ISDN 프로토콜 중 ATM 적응 계층(AAL: ATM Adaptation Layer)은 ATM 계층과 상위 사용자 서비스 계층의 중간으로서 주된 기능은 ATM 계층이 제공하는 서비스와 사용자가 요구하는 서비스의 차이를 해소하는 것이다. 이를 위해서 AAL은 사용자 서비스 정보와 ATM 셀 포맷으로 결합시키고, 전송 오류의 처리, 손실 또는 삽입된 셀의 처리, 오류 셀의 처리 등의 기능을 수행한다. 또한 사용자 요구 서비스 품질을 만족시켜 주기 위한 사용자, 제어, 관리 평면에서 요구되는 기능들을 수행한다. 또한 AAL에서 수행되는 기능은 상위 계층에서 요구되는 것에 따라 달라진다. SSCS(Service Specific Convergence Sublayer) 계층은 SAAL을 구성하는 한 부분으로써 SSCOP(Service Specific Connection-Oriented Protocol) 부계층과 SSCF(Service Specific Coordination Function) 부계층으로 나누어진다.

그림 9는 B-ISDN UNI 프로토콜 구조를 보여 준다. 그림 9에서 보는 바와 같이 ATM 적응 계층은 다음 상위 계층에 의해 요구되는 기능들을 지원하

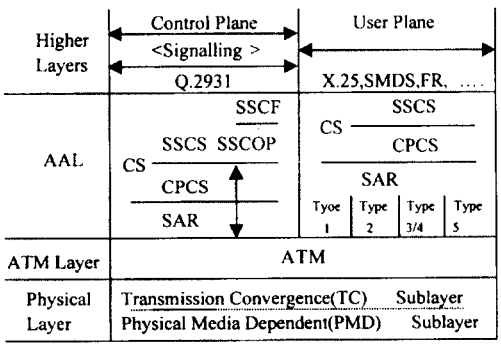

그림 9. B-ISDN UNI 프로토콜 구조

기 위하여 ATM 계층에 의해 지원되는 서비스들을 향상시키는 역할을 한다. SSCOP 프로토콜은 다양한 기능을 구현하기 위해 16개의 다른 PDU를 사용하고 있다. 기본적으로 SSCOP 프로토콜의 PDU는 사용자 정보 전송을 위한 SD, 수신된 SD에 오류가 있다고 판단하였을 때 발생하는 USTAT, 송신측에서 주기적으로 발생하여 수신측의 정보를 요구하는 POLL, 수신된 SD에 대한 ACK 및 새로운 수신측 정보를 전송하는 STAT, 그리고 POLL의 기능을 갖는 SDP PDU 등으로 메시지를 분류할 수 있다.

잘못된 순번의 SD를 받은 수신측의 SSCOP는 송신측으로 USTAR를 전송하여 재전송을 요구하고 이 시간 이후에 수신한 메시지들은 버퍼에 저장된다. USTAT나 한전 재전송된 SD가 분실될 때는 STAT에 의해서 재전송이 요구되고, 이 때 수신측은 버퍼에서 에러가 발생한 모든 SD들의 재전송이 요구된다. POLL에 의해서 수신 버퍼가 주기적으로 조사되어 재전송을 요구하므로 하나의 메시지에 연속적으로 에러가 발생할지라도 결국 수신측에 전송되는 것이 보장된다. 버퍼 관리를 위한 정보 교환 역시 USTAT나 STAT를 통하여 이루어져 송수신측 사이의 흐름 제어 기능이 수행된다. 이전 메시지에 발생한 에러가 복구되지 않아서 순서를 맞추기 위한 SD가 저장되어 있다. SSCS 계층의 적합성 시험을 하기 위해서는 프로토콜을 FSM으로 표현하여야 한다. FSM으로 SSCS 계층을 완전하게 표현할 수는 없지만 오류 발견 능력 분석을 위해 제한적으로 FSM으로 SSCS 계층을 모델링한다. SSCS 계층의 상태 천이표는 IUT-T의 Q.2100, Q.2110와 Q2130으로부터 유도된다. 유도된 상태 천이표를 이용하여 SSCS 계층의 부계층인 SSCF와 SSCOP 사이의 상태를 FSM으로 표현한다. 상태 천이표에서 SSCS 계층 프로토콜은 상태(AAL Connection Established/Data Transfer Ready)에서 입력으로 SD, POLL, STAT 또는 USTAT를 받은 경우, 내부 변수의 값에 따라 출력이 다른 EFSM으로 다루어야 한다. 그러나 본 연구에서는 변수를 고려하지 않고 여러 가능한 출력 중 하나를 받아도 규격을 만족하는 것으로 가정하여, SSCS 계층 프로토콜을

FSM으로 처리하도록 한다. [그림 4.10]은 SSCS 계층에 대한 프로토콜을 FSM으로 표현하고 있다. 즉, SSCF와 SSCOP사이의 관계를 FSM으로 표현한 것이다.

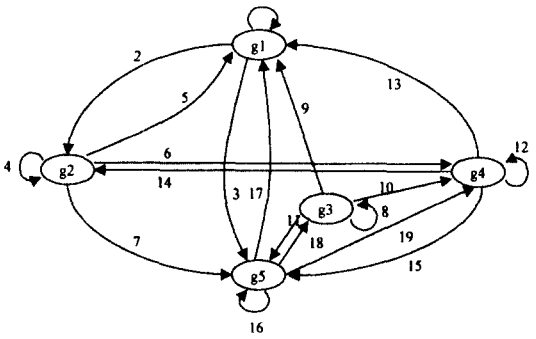

그림 10. B-ISDN UNI SSCS의 FSM

예를 들면 재전송되는 BGN 또는 RS에 대한 처리는 시험기가 BGN 또는 RS를 처음으로 송신한 후 그 상태에서 적절한 출력을 받아도 이를 받지 않은 것으로 가정하여 다시 동일한 BGN 또는 RS를 보내어 처리한다. 그림 11에서 각 상태에 대한 입출

number	Input/Output
1	BGNo/BGREJ, END/ENDAK, UD.req/UD, I11/END, I12/-
2	EST.req/BGN
3	BGN/o(BGAK,RS), RS/RSAK, UD.req/UD, I31/-
4	END/ENDAK, I32/-
5	BGREJ/-
6	REL.req/END
7	BGAK/-, BGNx/BGAK
8	BGNx/BGAK
9	EST.req/BGN
10	BGNx/BGAK
11	UD.req/UD, I2/-
12	BGN o/(BGAK,END), UD.req/UD, I4/
13	ENDAK/-, BGREJ/-, END/ENDAK
14	REL.req/END
15	BGNx/BGAK
16	BGNo/BGAK, BGNx/BGAK, DATA.req/SD, RSo/RSAK, RSx/RSAK, ERo/ERAK, UD.req/UD, SD/(USTAT,ER or-),POLL/(STAT,ER or SD), STAT/(ER,SD or-), USTAT/(SD or ER),I51/-
17	END/ENDAK, I52/-
18	EST.req/RS
19	REL.req/END

I11:BGAK,RS,RSAK,ER,ERAK,SD,POLL,STAT,USTAT I12:BGREJ,ENDAK,UD
I2:END,ENDAK,RS,RSAK,ER,ERAK,SD,POLL,STAT,USTAT,BGNo,UD
I31:BGAK,RSo,ER,ERAK,SD,POLL,RSAK,STAT,USTAT,UD
I32:BGREJ,ENDAK, I4:BGAK,RS,RSAK,ER,ERAK,SD,POLL,STAT,USTAT,UD
I51:BGAK,RSAK,ERAK,UD

그림 11. 그림 10에 대한 입출력 상태

력 FSM 상태를 살펴보면 [-]는 어떤 입력에 대해 출력이 없는 것(NULL)을 나타내고, [o]는 재전송된 PDU를 그리고 [x]는 처음 전송된 PDU를 나타낸다. SSCS 프로토콜의 오류 발견 능력 분석을 위해 UIO 순서를 이용하여 FSM으로부터 시험항목을 생성한다.

SSCS 프로토콜의 오류 발견 과정은 Inres 프로토콜에 적용한 순서와 동일하다. 표 2은 B-ISDN UNI SSCS 계층 프로토콜에 대한 오류 발견 능력 분석 결과를 보여 주고 있다. 오류 모델별로 100,000개의 오류 FSM을 생성하여 오류 발견율을 분석하였다. 오류 모델1과 3은 Inres 프로토콜과 동일하게 100%의 오류 발견율을 보였다. 그러나 오류 모델 2와 4는 Inres 프로토콜에 비해 다소 떨어진 99.13%와 96.89%의 오류 발견율을 보였다. 이는 SSCS 프로토콜을 FSM으로 변환할 때 많은 제약을 가했기 때문이다.

표 2. B-ISDN UNI SSCS의 오류 발견 능력 분석 결과

Fault model	Fault FSM	Detected FSM	Fault coverage(%)
1	100,000	100,000	100.00
2	100,000	99,135	99.13
3	100,000	100,000	100.00
4	100,000	96,892	96.89

4.3 결과 분석

시험항목에 대한 오류 발견 능력은 적합성 시험의 효율성을 평가하는 하나의 기준이 된다. 오류 발견 능력 분석이 정확하게 되면 시험항목의 생성 비용을 줄이고 적합성이 증가되어 프로토콜의 상호 연동 가능성을 높여주게 된다. 본 연구에서 적용사례로 Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜에 대한 시뮬레이션을 수행하였다. 시뮬레이터에 의한 오류 발견 능력 분석 결과는 Inres와 B-ISDN UNI SSCS 프로토콜의 오류 모델 1,3에 대해서 오류를 100% 발견하는 결과를 보였다. 따라서 시뮬레이터에 의해 분석된 오류 발견 능력은 상당히 높음을 알 수 있었다.

오류 모델 정의에서 알 수 있는 바와 같이 출력 오류는 발생 가능한 모든 오류를 발견하였다. 오류 모델3은 실제 프로토콜 구현시 오류 발생 가능성이 거의 없는 경우이다. 따라서 한 상태를 추가하거나 삭제하는 경우에 각 상태에 UIO를 적용하게 될 때 오류 모델3에서는 UIO가 존재하지 않기 때문에 오류를 찾아내게 된다. 그리고 시뮬레이터에서 오류를 판정할 때 시험항목 적용이 불가능할 때는 오류라고 판정하도록 하였다. 오류 모델 2는 다른 오류 모델에 비해 오류 발견율이 다소 떨어짐을 알 수 있다. 왜냐하면 UIO가 오류 모델에서 중복되는 경우가 존재하여 오류를 발견하지 못하게 되기 때문이다. 마지막으로 오류 모델 4는 다른 오류 모델에 비해 오류 발견율이 가장 떨어지고 있다. 이는 출력 오류와 천이 오류가 복합되어 발생되는 경우 오류 FSM에서 동일한 UIO를 생성하는 경우가 더욱 증가되기 때문이다. 그림 12는 Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜에 대한 비교 결과를 보여 준다. Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜의 오류 발견 능력을 비교했을 때 B-ISDN UNI SSCS 계층 프로토콜이 상대적으로 떨어짐을 알 수 있다. 이는 프로토콜을 FSM 으로 표현할 때 여러 가지 조건을 제한했기 때문이라고 생각된다. 그러므로 FSM을 확장한 연구가 필요함을 알 수 있다.

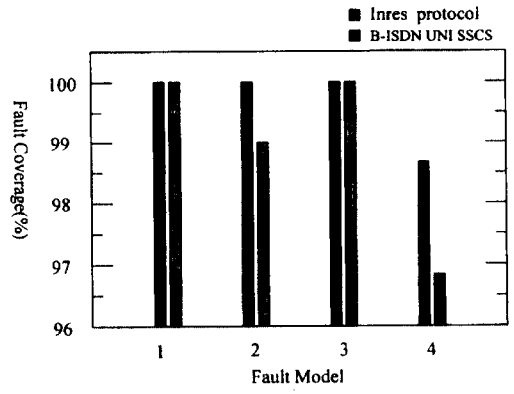

그림 12. Inres와 B-ISDN UNI SSCS 프로토콜의 오류 발견 능력 비교

실제 프로토콜 시험에서 시험항목의 크기와 오류 발견 능력은 비례하는 것으로 알려져 있다. 따라서 시험항목의 크기가 커지면 오류 발견 능력은 증대되지만 시험 시간이 길어지는 단점이 발생한다.

V. 결 론

시험스위트의 오류 발견능력 분석을 본 연구에서는 발생 가능한 오류를 네 가지 형태의 오류 모델을 정의하였다. 그리고 생성된 시험항목이 어느 정도 오류 발견 능력을 갖는지를 평가하기 위해 시뮬레이터를 설계, 구현하였다. 이를 이용하여 Inres 프로토콜과 B-ISDN UNI SSCS 프로토콜에 적용한 결과를 보였다. Inres와 B-ISDN UNI SSCS 프로토콜에 적용 결과, 오류 모델 1과 3은 100%의 높은 오류 발견율을 보였다.

본 시뮬레이터를 통합 환경에 적용할 수 있도록 확장하고, 실제 응용 프로토콜의 적합성 시험에 대한 효율성을 평가하는 새로운 기준을 제시하는 연구가 필요하다. 실제 프로토콜 시험에서 시험항목의 크기와 오류 발견 능력은 비례하는 것으로 알려져 있다. 따라서 시험항목의 크기가 커지면 오류 발견 능력은 증대되지만 시험 시간이 길어지는 단점이 발생한다. 시험 비용과 오류 발견 능력과의 상관관계에 대한 정확한 연구 결과는 아직 나와 있지 않으며 앞으로 연구되어야 할 분야이다.

참 고 문 헌

1. Paul W. King, "Formalization of Protocol Engineering Concepts," IEEE Transactions on Computers, vol. 40, no. 4, April, 1991.
2. D. Lee, K. Sabnani, D. M. Kristol, S. Paul, "Conformance testing of protocols specified as communicating FSMs", IEEE INFOCOM' 93, pp. 115-127, 1993.
3. W. J. Chun, "Test case generation for protocols specified in estelle", Ph. D Thesis, Computer and Information Science, University of Delaware,

1992.

4. F. Lombardi and Y.U. Shen, "Evaluation and Improvement of Fault Coverage of Conformance Testing by UIO Sequences," IEEE Trans. Communications, Vol. 40, No. 8, pp. 1288-1293, August 1992.
5. Pim Kar, "Test Coverage Estimation by Explict Generation of Faulty FSMs", IWPTS VII, pp.323-330, 1994.
6. M. Yao, A. Petrenko and G. v. Bochmann, "A structural analysis approach to the evaluation of fault coverage for protocol conformance testing", FORTE' 94, pp. 389-404, 1994.
7. H. Mottler, A. Chung and D. Sidhu, "Fault coverage of UIO-based methods for protocol testing", Protocol Test System, IFIP, 1994.
8. C. J. WANG and M. T. LIU, "Generating Test Cases for EFSM with Given Fault Models", IEEE INFOCOM' 93, pp. 774-781, 1993.
9. K. Naik and B. Sarikaya, " Testing Communication Protocols", IEEE Software pp. 27-37, January 1992.
10. A. R. Cavalli, J. P. Favreau and M. Phalippou, " Standardization of formal methods in conformance testing of communication protocols", Computer Networks and ISDN Systems, vol. 29, no. 1, pp. 3-14, 1996.
11. D. P. Sidhu, H. Motteler and R. Vallurupalli, "On testing hierarchies for protocols", IEEE/ACM Trans. Networking, vol. 1, no. 5, pp. October 1993.
12. Dieter Hogrefe, "Status Report on the FMCT Project", IWPTS VII, pp. 165-176, 1994.
13. A. Petrenko, N. Yevtushenko, R. Dssouli, "Testing Strategies for Communicating FSMs", IWPTS VII, pp. 181-196, 1994.
14. R. E. Miller and S. Paul, "Structural analysis of protocol specifications and generation of maximal fault coverage conformance test sequences", IEEE/ACM Trans. Networking, vol. 2, no. 5, pp. 457-470, 1994.
15. Kshirasagar Naik, " Fault-tolerant UIO Sequences in Finite State Machines", IWPTS VIII, pp. 207-220, 1995.
16. H. Ural and B. Yang, "A Test Sequence Selection Method for Protocol Testing", IEEE Trans. on Communications, vol. 39, no. 4, pp. 514-523, April 1991.
17. T. Ramalingam, A. Das and K. Thulasiraman, "Fault detection and diagnosis capabilities of test sequence selection methods based on the FSM model", Computer Communications, vol. 18, no. 2, pp. 113-122, February 1995.
18. T. Ramalingam, A. Das and K. Thulasiraman, "On testing and diagnosis of communication protocols based on the FSM model", Computer Communications, vol. 18, no. 5, pp. 329-337, May 1995.
19. A. Petrenko, G. v. Bochmann and M. Yao, "On fault coverage of tests for finite state specifications", Computer Networks and ISDN Systems, vol. 29, no. 1, pp. 81-106, 1996.
20. 김재철, 최양희, "프로토콜 시험을 위한 통합 환경의 설계와 구현", 한국정보과학회 논문지, 22권, 12호, 1695-1704(쪽), 1995.
21. 김광현, 허기택, 이동호, "통신 프로토콜 시험 항목의 오류 발견 능력 평가 방법", 한국통신학회 논문지, 21권, 8호, 1948-1957(쪽), 1996.

김 광 현(GwangHyun Kim) 종신회원
1989년 2월 : 광운대학교 전자계산학과 졸업(이학사)
1991년 2월 : 광운대학교 대학원 전자계산학과 졸업 (이학석사)
1997년 2월 : 광운대학교 대학원 전자계산학과 졸업(이학박사)

1997년 3월~ 현재: 광주대학교 컴퓨터학과 전임강사
〈관심분야〉 프로토콜공학, 차세대인터넷, 분산시스템.

허 기 택(Gitaek Hur) 정회원
1984년 2월 : 전남대학교 전산통계학과 졸업(이학사)
1986년 2월 : 전남대하교 대학원 전산통계학과 졸업(이학석사)
1994년 2월 : 광운대학교 대학원 전자계산학과 졸업(이학박사)
1989년 3월~현재 : 동신대학교 컴퓨터학과 부교수 및 정보전산센터소장
〈관심분야〉 프로토콜공학, 컴퓨터네트워크, 멀티미디어통신.