국제 통신 표준 언어를 이용한 통신 프로토콜 설계 및 검증 방법론 연구

노철우†

요 약

본 논문에서는 통신 프로토콜 개발 시 사용되는 PDU, SDU, SAP, 서비스 프리미티브를 어떻게 정의하고 사용하는지에 대한 명확한 설계 개념과 검증 방법을 ITU에서 통신 규격 및 설계 언어로 권고하고 있는 SDL과 국제 통신 표준 언어인 ASN.1, MSC, TTCN을 사용하여 정립한다. 통신 프로토콜의 예로 잘 알려진 Inres 프로토콜을 확장하여 SDL로 설계하며, SDL의 설계 규격에 비트 스트링 전송을 위한 ASN.1 메시지의 삽입, 규격에 대한 설계 검증을 위한 MSC의 생성, 검증으로부터 TTCN을 이용한 시험 케이스의 생성 및 적합성 시험 등 프로토콜 개발 순기 전반에 걸친 개발 방법론을 정립한다.

키워드 : SDL, ASN.1, MSC, TTCN, 프로토콜

A Study on the Design and Validation Methodology of Communication Protocols Using International Communication Standard Languages

Cheul-Woo Ro†

ABSTRACT

In this paper, We set up the development methodology of communication protocols as well as the concrete design concept concerning for how to define and use the PDU, SDU, SAP, and service primitives using SDL, which is recommended by ITU-T, and other international standard languages such as ASN.1, MSC, and TTCN. This methodology covers the SDL design of extended Inres protocol, a well known protocol example, insertion of ASN.1 message for transportation of bit string, generation of MSC for validation of design specification, generation of test cases using TTCN from validation, and performance of conformance test.

1. 서 론

ITU 등 국제 표준화 기구는 3G (UMTS/IMT-2000), WAP, 블루투스, WLAN 등 새로운 통신표준을 공표하고 있으며, 시스템 개발 회사들은 표준화 작업이 완료하기 이전에 제품 개발을 수행하고 개발 과정시 소요되는 시간과 time-to-market을 단축하기 위하여 고심하고 있다. 규격 및 설계용 국제 표준 언어인 SDL(Specification and Description Language)[1-3]과 ASN.1(Abstract Syntax Notation)[4], MSC(Message Sequence Charts)[5], TTCN(Tree and Tabular

† 정회원 : 신라대학교 컴퓨터정보공학부 부교수
논문 접수 : 2002년 9월 18일, 심사완료 : 2002년 10월 20일

Combined Notation)[6-7] 등의 국제 통신 표준 언어와 함께 개발 및 시험환경을 구성함으로써 복잡한 대규모 통신 시스템의 개발 및 검증을 정형화된 수단을 통해 자동으로 수행함으로써 이를 해결하려 하고 있으며 설계, 구현, 검증 및 시험의 자동화 수단은 표준 규격의 잦은 갱신, 새로운 기능의 빈번한 추가 그리고 전문 인력의 이직 등에 따라 그 필요성이 날로 더해 가고 있다.

본 논문에서는 이러한 추세에 부응하여, 설계서와 구현된 프로그램 사이의 엄격한 일관성을 보장받을 수 있는 ITU-T의 SDL을 이용하여 시스템을 설계하고, 국제적으로 공인된 개발 환경을 이용하여 실행 가능한 소프트웨어를 개발하고 실제 타겟 시스템과 동일한 환경에서 개발된 기능을 검증하는 개발 순기를 모두 수용하는 검증된 개발 방법론을 정립하고자 한다.

계층 개념을 갖는 통신 프로토콜은 자신의 계층에서 전송할 패킷인 PDU(Protocol Data Unit)를 생성하고 이를 상대편의 같은 계층으로 전송 (peer-to-peer)하게 된다. 실제 전송은 생성된 PDU에 필요한 heading 정보를 추가하여 SDU(service Data Unit)를 생성하고 이 SDU를 전송 통로인 SAP(Service Access Point)를 통하여 하위계층에 전송하게 된다. 수신 측에서는 하위계층으로부터 SDU를 수신한 후 추가된 heading 정보를 제거하고 실제 전송하고자 하는 PDU 만을 동급 계층에 전송함으로서 peer-to-peer 통신이 이루어지게 된다. 또한 이러한 전송의 요청은 서비스 프리미티브(Service Primitive : SP)의 사용에 의하여 이루어진다[8-9].

본 논문에서는 통신 프로토콜 설계 시 언급된 PDU, SDU, SAP, SP를 어떻게 정의하고 사용하는지에 대한 명확한 설계 개념과 개발 방법을 국제 통신 표준 설계 언어인 SDL을 사용하여 정립한다. 통신 프로토콜의 예로 잘 알려진 Inres 프로토콜[9]을 확장하여 SDL로 설계하고 설계 규격에 비트 스트링 전송을 위한 ASN.1 메시지의 삽입, 규격에 대한 설계 검증을 위한 MSC의 생성, 검증으로부터 TTCN을 이용한 시험 케이스의 생성 및 적합성 시험 등 프로토콜 개발 순기 전반에 걸친 개발 방법론을 정립한다. Inres 프로토콜 설계로 프리미티브와 PDU 전송에 의한 계층간 peer-to-peer 통신을 수행하는 통신 프로토콜의 명확한 설계 개념을 정립하며, 통신 패킷 전송 시 실제 전송되는 비트 스트링을 구현하기 위하여 ASN.1 인코딩/디코딩을 확장하여 프로토콜을 설계한다. 시뮬레이션과 MSC 생성에 의한 프로토콜의 검증 특성을 검증할 수 있으며, 생성된 MSC로부터 규격과의 적합성을 검증할 수 있는 적합성 시험케이스와 시험 슈트를 적합성 시험 언어인 TTCN에 의하여 표현 및 생성하며 이를 토대로 적합성 시험을 수행할 수 있다.

1.1 통신 프로토콜 표준 언어

1.1.1 SDL

ITU에서 시스템 명세 및 기술용 표준 언어로 권고하고 있는 SDL은 통신시스템의 동작을 계층적 구조로 나타내고 시스템을 명확하게 기술하기 위해 통신 시스템의 특징인 실시간 처리 프로세스들 간의 상호통신을 확장된 유한상태 기계의 개념을 기초로 정의한 언어이다. 블랙박스의 개념을 도입하고 계층 구조로 설계되며 시스템과 외부 환경, 블록들 간, 그리고 프로세스들 간 통신은 신호(signal)와 신호의 매개변수를 사용하여 이루어진다. SDL과 일반적인 프로그래밍 언어의 차이점은 SDL은 규격 및 설계 전용 언어로 규격에 대한 설계와의 일치성을 분석, 시뮬레이션 및 검증에 의하여 보장할 수 있는 반면 프로그래밍 언어는 수행의 추론에 기초를 하고 표면적인 규격 명세로 인하여 설계에 대한 정확성을 보장할 수 없으며 결국 코딩이 아닌 시험 단계에서 내재되어 있는 설계에러 때문에 많은 시간과 비용을 소비하게 된다. SDL 시스템은 객체지향 성격을 갖고 있으며 다음 요소들로 구성되어 있다[2].

- 구조(structure) : 시스템, 블록, 프로세스의 계층 구조를 갖는다.
- 통신(communication) : 채널과 신호경로(signal route)를 통한 신호에 의한 비동

기적 통신과 원격 프로시듀어 호출에 의한 동기적 통신을 제공한다.

- 행위(behavior) : 프로세스를 통해 기술한다
- 데이터(data) : 상속화, 일반화 그리고 특수화될 수 있는 추상 데이터 타입 지원, 권고안 Z.105를 따르는 ASN.1 데이터 타입을 지원한다.
- 타입 개념(type concept) : 상속성, 일반화 그리고 특수화를 이용하여 타입 계층 기술을 지원한다.

1.1.2 ASN.1

ASN.1은 서로 다른 인터페이스를 가진 원격 통신 시스템에서 그들 간의 공유되는 데이터를 표현하기 위해 ISO와 ITU에 의해 표준화된 표기법으로 통신 프로토콜 명세에 사용되는 언어이다. ASN.1은 서로 다른 응용 계층을 통해 적용되는 복잡한 메시지 자료구조를 비트패턴으로 변환하는 인코딩 규칙을 제공하고 각각의 최종 시스템에서 이를 디코딩 함으로써 동일한 자료구조를 공유할 수 있게 한다. 규격에 정의되어 있는 ASN.1 메시지는 인코딩 규칙과 ASN.1컴파일러에 의하여 직접 사용하는 방법과 ASN.1 메시지 타입에 대응되는 SDL 자료구조를 정의하여 사용하는 방법으로 나눌 수 있다. ASN.1은 TTCN과 SDL에서 사용될 수 있는 모듈들 내에 정의될 수 있다. 따라서 SDL 명세와 TTCN 시험 계열(test suite) 어플리케이션의 데이터 타입으로 사용될 수 있다. 이로 인하여 시스템 명세와 시험 명세에서 전달되는 정보간의 일관성을 보장한다.

1.1.3 MSC

ITU의 Z.120 권고안인 MSC는 SDL 시스템의 동적 행위를 기술하는데 적합한 표준 언어이다. MSC는 SDL 명세로부터 생성된 추상 통신 트리에서 하나의 노드에서부터 다른 노드까지 하나 이상의 신호 흐름을 기술하며 SDL 프로세스 실행에 대한 추적을 보여준다.

1.1.4 TTCN

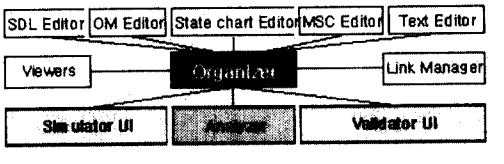
TTCN은 상위수준의 시험 스크립트 언어로 ISO/IEC 9646-3에서 표준화하여 탄생 하였으며 적합성 시험을 위하여 설계되었다. 시험 스크립트는 구현과는 독립적이며 white box 시험이 아닌 black box 시험 방식으로 작성된다. 시험 동작은 명령을 보내고 응답을 관찰하며 주어진 시간 내에 원하는 결과를 응답으로 주는지에 따라 시험 통과인 합격과 불합격을 판정한다.

2. 프로토콜 개발 방법

2.1 개발환경 SDT(SDL Development Tool)

CASE 도구 개발 업체의 하나인 Telelogic사에서 개발하여 상용화한 프로토콜 개발 및 검증 도구인 SDT는 SDL에 관한 국제 표준 권고안 Z.100 ~ Z.105까지를 완벽하게 지원하며 여러 가지 도구들을 포함하고 있는 통합 개발 환경으로 TTCN 환경인 ITEX 와 함께 Tau suite[10]로 서비스되고 있다.

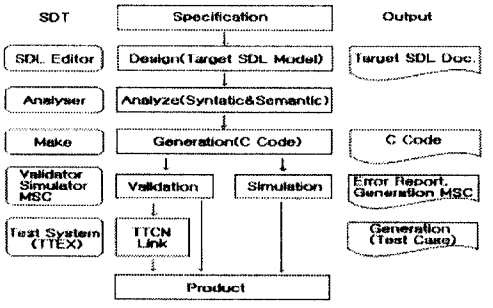
여러 도구들 중 Organizer는 모든 다이어그램과 SDL 계층구조, MSC, 객체 모델 다이그램, TTCN 문서와 텍스트 문서들을 포함하고 있으며 다른 도구들을 관리한다. 그밖에 주요 도구로써 SDL 편집기, 텍스트 편집기, MSC 편집기, 그리고 SDL 설계에 대한 구문 및 의미 분석에 대한 오류 및 경고 메시지를 생성시킬 수 있는 Organizer log가 있다. 또한 SDL 시스템에서 정의와 상호 참조에 관한 정보를 생성하는 기능을 가지고 있는 SDT 분석기(Analyzer)가 있고, 추적 목적으로 MSC를 이용할 수 있는 검증 도구인 Simulator UI와 상태 폭발 및 교착 상태를 검증을 할 수 있는 Validator UI가 있다. (그림 1)은 SDT (Tau)의 구성을 보여준다.[11]



(그림 1) SDT 구성

2.2 통신 프로토콜 개발 방법

(그림 2)는 SDL 환경에서의 통신 프로토콜 개발과정을 보여준다.



(그림 2) 통신 프로토콜 개발 과정

2.2.1 설계

1) 시스템 설계

계층적 구조로 표현되는 SDL 설계에서 가장 상위 설계인 시스템 설계는 개발 대상의 명세와 각 기능들이 수행될 수 있도록 상호 작용들을 고려하여 설계된다. 계층 설계를 위해서 프로토콜 계층을 하나의 블록으로 설계하고 기능적인 분류에 따른 부 계층을 프로세스로 설계하였다. 시스템은 블록으로 구성되며 블록간의 통신 경로인 채널과 채널 상에서 메시지 전달을 위한 시그널이 정의되어 있다.

2) 블록 설계

블록 설계는 시스템이 행하는 구체적인 동작이 기술되는 프로세스를 설계하는 단계이다. 프로세스간 통신 수단인 SDL의 신호 경로를 정의하고, 프로세스간 전역변수를 선언한다.

3) 프로세스 설계

SDL 심벌을 이용하여 실제 시스템이 수행하는 보다 구체적인 동작을 기술하는 단계로서 내부변수의 정의와 프로시듀어의 설계를 수행한다.

2.2.2 분석 및 코드생성

SDL 개발환경인 Tau에서는 설계된 SDL인 SDL GR(그래픽 형태)을 SDL PR(텍스트 형태)[10]로 변환하는 Analyze 과정과 코드 생성기에 의하여 실행코드를 생성하는 Make 과정을 수행함으로써 시뮬레이션을 위한 실행코드를 생성한다. 이 후 Tau의 시뮬레이션 도구인 Simulator UI를 수행함으로써 자동으로 MSC를 생성한다.

2.2.3 검증

1) 시뮬레이션

요구 명세에 따른 SDL 설계를 마치고 구문 및 의미 분석 시 발생된 설계상의 오류를 수정한 후 시뮬레이션을 수행하여 산출된 MSC와 규격으로부터 설계 시 작성되는 동작 시나리오와의 일치성으로 검증을 수행한다. SDL 시뮬레이터는 개발환경과의 상호작용, 프로세스의 동적 생성 및 종료, 시그널 송수신과 타이머 처리 및 프로세스의 상태와 동작 표현을 갖는 SDL 모델을 시뮬레이션 한다. MSC는 시뮬레이션 실행결과의 광범위한 검토를 제공해 주며 SDL 다이아그램 소스에 대한 자동 추적을 가능하게 해준다.

2) 검증(Validation)

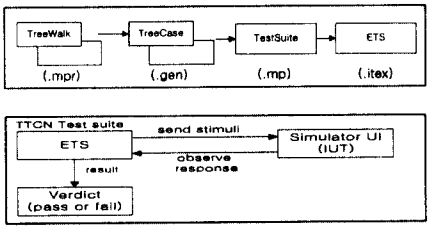
검증은 시뮬레이션과 디버깅을 통해 찾기 어려운 run-time 에러를 찾기 위한 수단으로 SDL 시스템의 상태공간을 탐색하여 설계 시 내재되어 있는 논리에러를 찾아내는 과정을 수행한다. SDL 개발환경에서는 Tau의 Make 과정에서 validator를 위한 실행코드를 생성한 후 Validator UI(검증기)를 통해 설계에서 가능한 모든 경로의 탐색을 수행하여 데드록, 규정되지 않은 시그널 수신 등 프로토콜 검증 특성을 검사한다.

SDL 설계에 대한 검증은 SDL 시스템을 도달성 그래프로 보고 그래프의 노드인 시스템 상태에 대한 탐구를 수행하며 탐색을 하게 된다. FSM에 기반을 둔 SDL 시스템으로부터 생성되는 도달성 그래프는 항상 상태 폭발의 문제를 갖게 되기 때문에 다양한 탐색 및 탐구 알고리즘을 가져야 하며 그래프나 트리의 최대 depth를 지정함으로써 상태폭발의 문제를 해결

한다. SDL 개발환경인 Tau의 validator도 상태공간에서의 탐구 알고리즘인 비트 상태 탐구, random walk, exhaustive 탐구, MSC verification, tree walk를 갖고 있으며 이들 방식을 적절히 사용하여 검증을 수행 한다. 이중 tree walk는 여러 개의 tree walk 보고서(TreeWalk로 명명)를 생성하며 이들로부터 MSC에 대응하는 시험 케이스를 생성하여 적합성 시험에 이용한다[12].

2.2.4 시험

검증 실행결과로 생성되는 TreeWalk 각각은 TTCN에서 활용할 수 있는 시험 케이스로 변환할 수 있다. 원하는 시험 케이스를 선택하여 mpr 파일로 저장하고 이를 gen파일로 변환한 후 이들 gen 파일들을 하나의 시험 슈트(test suite) 파일로 만든다. 생성된 시험 슈트인 mp 파일은 파일변환에 의하여 Tau의 시험환경용 TTCN 파일인 itex 파일로 변환한다. TTCN의 컴파일 옵션을 지정하고 컴파일하면 Simulator UI와 Co-Simulation을 수행할 수 있는 실행파일인 ETS(Executable Test Suite)이 생성된다. Tau-TTCN Suite에서는 호스트상 에서의 적합성 시험을 co-simulation에 의하여 수행하게 되는데[10], 시뮬레이션 시 생성되는 실행코드는 Simulator UI로 읽혀지고 IUT(Implementation under Test)[6]의 역할을 수행하며 TTCN 시험 스크립트에 대한 시험 슈트 실행파일인 ETS는 IUT로 해당 시그널을 전송하고 IUT로부터의 응답에 의하여 pass/fail 여부를 판단하게 된다[12]. (그림 3)은 TreeWalk로부터 실행파일 인 ETS 까지의 파일 변환 단계와 co-simulation에 의한 적합성 시험 구조를 보여준다.



(그림 3) ETS 생성을 위한 파일 변환과 Co-simulation

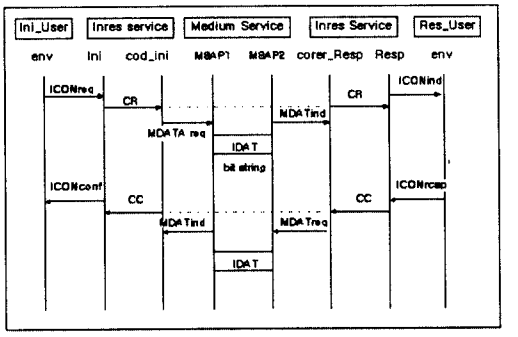
3. 확장된 Inres 통신 프로토콜 개발

3.1 프로토콜 규격

Inres 프로토콜은 실제 운영되고 있는 통신 시스템의 프로토콜은 아니나 OSI의 계층개념을 갖고 있으며 사용자에게 연결지향 서비스를 제공해 주는 프로토콜로 통신 프로토콜 설계 개념을 이해하기에 적합한 프로토콜이다. 본 논문에서는 이 프로토콜을 확장하여 통신 프로토콜의 설계 방법을 정립한다.

3.1.1 Inres 서비스

SDU는 서비스 프리미티브의 매개변수로 전송된다. 서비스는 두 SAP에 의하여 제공될 수 있다. SAP(ISAP.ini)에서 사용자인 initiator는 연결을 초기화하여 설정을 완료하고 그 후 데이터를 전송할 수 있다. 상대측 SAP인 ISAP.res에서 다른 사용자인 responder는 연결요청을 수용하거나 거절할 수 있다. Inres는 연결지향 프로토콜로 (그림 4)는 연결설정단계에 대한 신호 순서도인 MSC를 보여준다. ICONreq, ICONind 등은 사용자와 시스템사이에서 전송을 위해 사용되는 서비스 프리미티브이다.



(그림 4) 프로토콜 규격-MSC

3.1.2 Medium 서비스

Symmetrical하며 비연결성 모드로 동작하는 medium 서비스는 MSDU 타입의 매개변수를 갖는 서비스 프리미티브인 MDATreq,

MDATind를 사용하며 두 SAP인 MSAP1, MSAP2에서 제공된다.

3.1.3 Inres 프로토콜

Inres 프로토콜은 두 프로토콜 엔티티인 initiator와 responder 사이에서 동작하는 연결 지향 프로토콜이다. 프로토콜 엔티티들은 peer-to-peer 통신을 위하여 <표 1>의 PDU를 상호 교환한다. 연결 지향 프로토콜인 Inres 프로토콜은 아래 세 단계를 가지며 각 단계에서 사용되는 서비스 프리미티브와 PDU를 지정한다.

<표 1>PDU

PDU	용도	매개변수	서비스 프리미티브
CR	연결설정	없음	ICONreq ICONind
CC	없음	없음	ICONresp ICONconf
DT	데이터 전송	sequence number ISDU	IDATreq IDATind
AK	응답	sequence number	-
DR	연결해제	없음	IDISreq IDISind

1) 연결설정단계

Initiator 엔티티는 ICONreq 서비스 프리미티브에 의하여 연결설정을 초기화시키며 CR PDU를 responder 엔티티에게 전송하게 한다. CR을 수신한 responder 엔티티는 responder 사용자에게 ICONind를 전송함으로써 연결요청이 있음을 알려주며 사용자는 연결수락을 하고 싶으면 ICONresp를, 거절하고 싶으면 IDISreq를 엔티티에게 응답한다. 엔티티는 ICONresp를 받으면 CC를, IDISreq인 경우는 DR로 initiator 엔티티에게 응답을 한다. CC를 medium 서비스를 통하여 응답받은 initiator 엔티티는 사용자에게 ICONconf를 보냄으로써 연결설정을 알려주고 자신의 상태천이에 의하여 데이터 전송단계로 들어간다. (그림 4)에서 CR이 하위계층의 전송을 거쳐 상대편으로 peer-to-peer 전송이 됨을 보여준다.

2) 데이터 전송단계

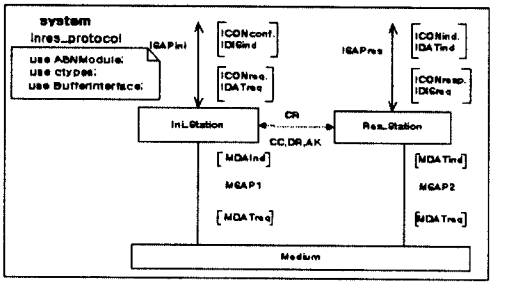
ICONconf를 수신한 initiator 사용자는 상대편인 responder 사용자에게 전송하고자 하는 데이터인 ISDU를 매개변수로 갖는 IDATreq를 엔티티에게 전송함으로써 데이터 전송을 시작하며 엔티티는 이를 위한 PDU인 DT를 작성하고 이를 peer-to-peer로 전송한다.

3) 연결해제 단계

IDISreq에 의한 연결해제는 initiator, responder 양측에서 전부 발생할 수 있다. DR PDU 전송 후 IDISind 프리미티브 수신에 의해 해제 단계를 완료한다.

3.2 시스템 설계

(그림 5)는 SDT상에서의 Inres 시스템 설계도를 보여준다. Inres 시스템은 두 프로토콜 엔티티에 대응되는 Ini_Station, Res_Station과 medium 서비스를 제공하는 서비스 제공자인 Medium의 세 블록으로 구성된다. 각 블록은 시그널 라우터에 의해 연결되는 프로세스를 갖으며 각 프로세스는 SAP의 행동에 대한 모델을 포함한다. (그림 5)에서 두 블록간의 PDU 전송을 점선으로 표시하였으나 실제 전송은 SAP를 경유하여 시그널로 표현되는 프리미티브에 의하여 이루어진다.

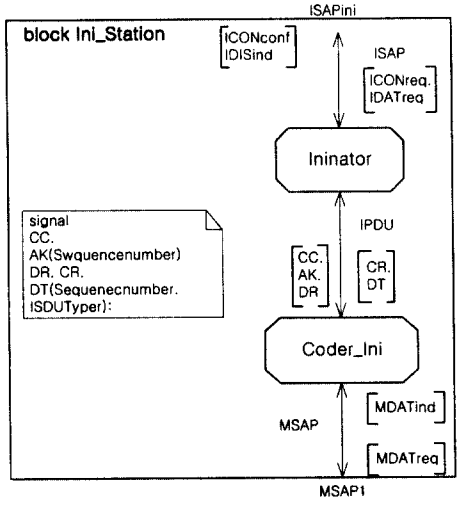


(그림 5) Inres 프로토콜 시스템 다이아 그램

3.3 블록 설계

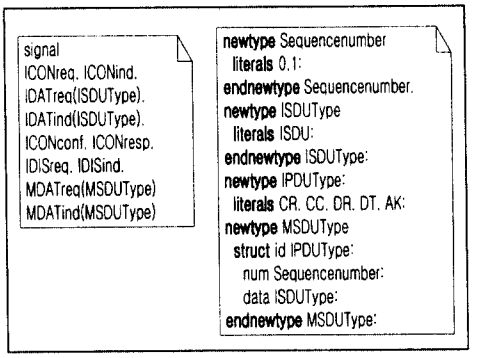
프로토콜 엔티티를 표시하는 블록은 프리미티브에 대응하는 해당 PDU를 생성하는 프로세스 타입(Initiator, Responder 프로세스)과 생성

된 PDU를 하위계층의 SDU로 전송하는 프로세스 타입(Coder_Ini, Coder_Res)으로 구성된다. 결국 프리미티브의 매개변수는 PDU에 포함되고 이는 하위계층에 SDU로 전달된다.
채널로 표현되는 SAP인 ISAPini는 블록에서 프로세스간 신호경로인 ISAP, IPDU, MSAP를 경유하여 하위계층 SAP인 MSAP1으로 연결됨을 (그림 6)에서 보여준다.



(그림 6) Inres 프로토콜 블록 다이아 그램

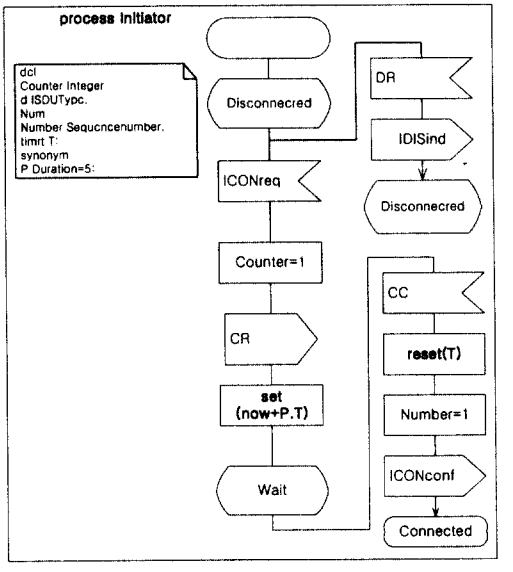
SAP(ISAPini, ISAPres, MSAP1, MSAP2)는 SDL에서 채널로 표현되고 프리미티브(ICONreq, IDATreq)는 시그널로, PDU(IPDU-Type)와 SDU(MSDUType)는 시그널의 매개변수로 표현된다. (그림 7)에서 이들에 대한 자료구조를 보여준다. SDL에서 PDU, SDU의 자료구조로 사용되는 Newtype ~ Endnewtype은 C 언어의 STRUCT 문에 해당된다.



(그림 7) 시그널과 PDU, SDU 자료구조

3.4 프로세스 설계

3.4.1 initiator와 coder_ini 프로세스



(그림 8) Inres 프로토콜 프로세스 다이아 그램(initiator)

사용자로부터 ICONreq 프리미티브를 수신한 initiator 프로세스는 상대편 responder 프로세스에게 peer-to-peer 전송으로 PDU인 CR을 전송함으로써 ICONind 프리미티브를 상대편 사용자에게 알려준다. 이를 위하여 자신의 하위 프로세스인 Coder_ini 프로세스에게 CR을 송신하며 상대측으로 부터의 응답을 기다린다.

이때 타이머 T를 set하여 P 시간동안에 응답이 없으며 타임아웃에 의하여 재전송 등 필요한 행동을 취하며 타임아웃 이전에 원하는 응답 PDU인 CC를 수신하면 자신의 사용자에게 ICONconf 프리미티브를 전송함으로써 연결설정단계를 완료하며 자신은 데이터 전송을 위한 connected 상태로 상태천이를 하고 데이터 전송 프리미티브에 의한 데이터를 기다리게 된다.

responder 프로세스는 CR을 받으면 자신의 사용자 또는 상위 계층 프로세스에게 ICONind를 전송함으로써 송신 측에서부터 연결요청이 왔음을 알리고 ICONresp를 기다린다. 그리고 CR에 대한 응답으로 CC를 전송하여 연결설정단계를 수행한다. 이 들 연결설정에 대한 MSC는(그림 4)의 규격대로 수행될 것이며 이는 시뮬레이션 수행결과인 (그림 12)에서 검증된다.

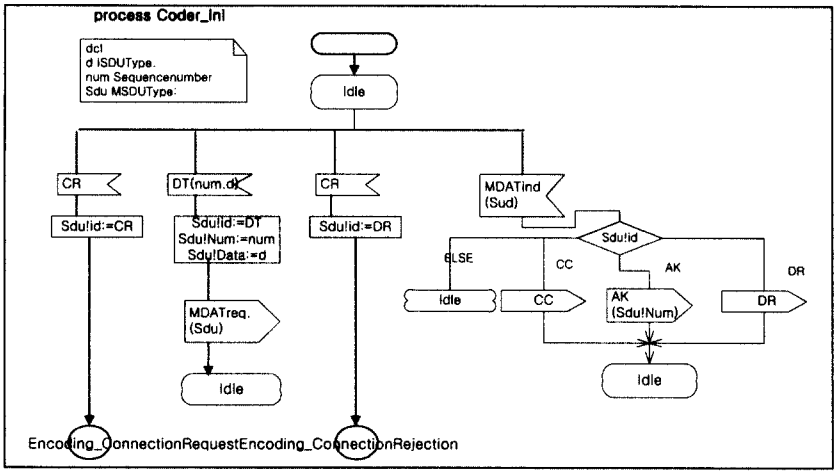
프로세스 initiator로부터 CR PDU를 받은 Coder_Ini 프로세스는 Sdu!id:=CR 타스크문에 의하여 CR에 대한 SDU를 작성한다(그림 9). 작성된 SDU는 MDATreq 프리미티브에 의하여 전송매체를 통하여 수신 측으로 전송된다. 수신 측 프로세스인 Coder_Res는 MDATind로 전송된 SDU를 수신하며 이후 heading 작업에 의하여 수신된 SDU가 CR임을 알고 CR PDU를 상위 프로세스인 Responder로 알려준다. 즉 PDU CR이 송신 측 프로세스인 initiator에서 수신 측 프로세스인 Responder로 peer-to-peer 전송됨을 보여준다.

3.4.2 ASN.1 인코딩

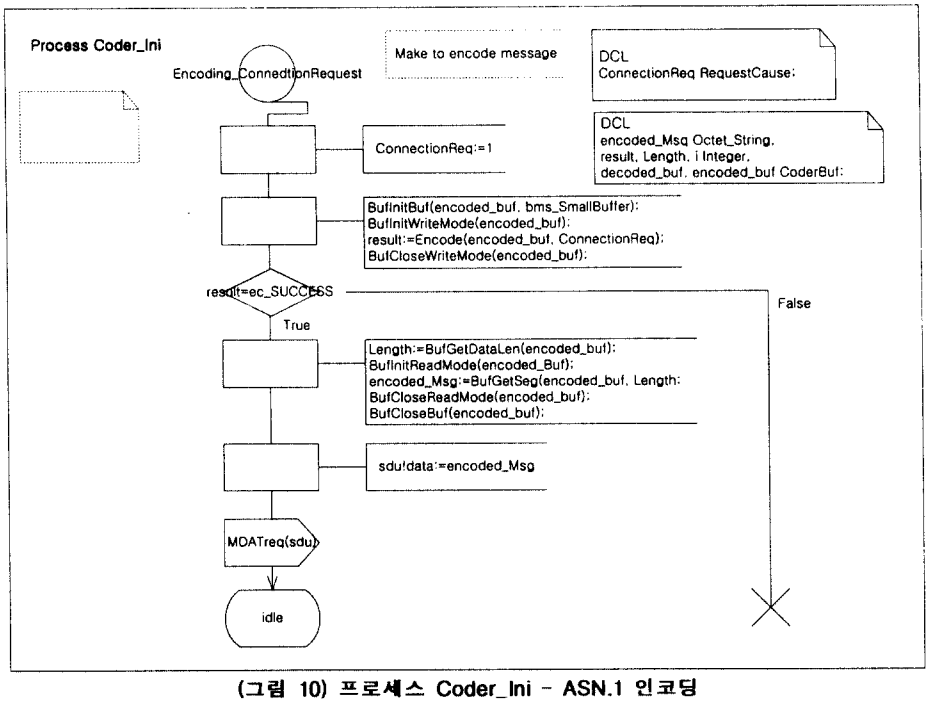
통신 매체를 통한 데이타 통신인 medium 서비스는 PDU 형식의 데이타를 비트 스트링으로 변환하여 전송하게 된다. 이를 표시하기 위하여 MDATreq 프리미티브에 의한 PDU는 ASN.1 인코딩 룰을 적용하여 ASN.1 메시지가 비트 스트링 값으로 변환되어 전송될 수 있도록 설계하였다.

연산자 Encode를 이용하여 PER 변환[4]을 수행하고 옥텟 스트링을 생성한다. (그림 10)은 인코딩에 의하여 생성된 옥텟 스트링인 encoded_Msg가 MDATreq 프리미티브의 매개변수로 하위 계층인 Medium 블록으로 전송되는 과정을 보여준다. 이 옥텟 스트링은 MSAP_manager 프로세스에서 비트 스트링으로 변환된 후 전송된다.

SDL 설계에 ASN.1 메시지 타입을 사용하기 위해서는 'Reference' 심볼 내에 'use'명령어를 이용하여 Tau에서 인코딩과 디코딩을 위한 codeBuf.sun과 ctypes.sun 패키지를 함께 정의하고 인/디코딩 변환 시 오류를 정의해 놓은 SDL 커널 파일인 'codererror.sdl'를 설계에 추가로 포함시켜야 한다. CodeBuf.sun 패키지를 로드하면 자동으로 BufferInterface로 재명명되어 정의된다.



(그림 9) Inres 프로토콜 프로세스 다이아 그램(coder_ini)

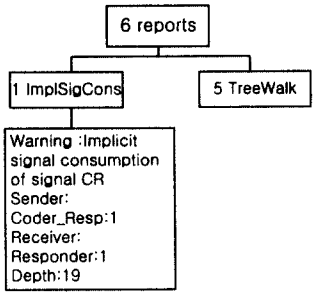


(그림 10) 프로세스 Coder_Ini - ASN.1 인코딩

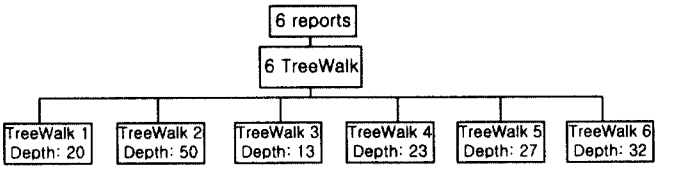
ASN.1 메세지는 다음과 같이 각각의 타입명과 타입, 그리고 범위를 정의하여 작성한다.

```
ASNModule DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
RequestCause ::= INTEGER(1..4)
ResponseCause ::= INTEGER(5..8)
RejectionCause ::= INTEGER(0..10)
TransMessae-Info-Type ::= SEQUENCE {
numberOfTransBlocks INTEGER(1..10),
asizeTypeNum INTEGER(1..5) }
END
```

(그림 10)의 선언문에서 'ConnectionReq' 변수의 타입은 ASN.1의 'RequestCause' 타입으로 선언함으로써 SDL 타입 선언이 아닌 ASN.1의 추상화 데이터 타입으로 선언할 수 있고 타스크 심볼내에 인코딩 함수인 'Encode()'를 이용하여 추상화 데이터 값을 하위 계층에서 송수신이 가능한 옥텟 단위로 인코딩 변환이 가능하도록 설계된다. 정수형 변수 'result'는 인코딩/디코딩 에러를 정의해 놓은 SDL 커널 파일 'codererror.sdl'을 참조하여 인코딩 변환 후 성공 또는 각종 에러의 값을 할당한다. 성공할 경우 시그널 'MDATreq'의

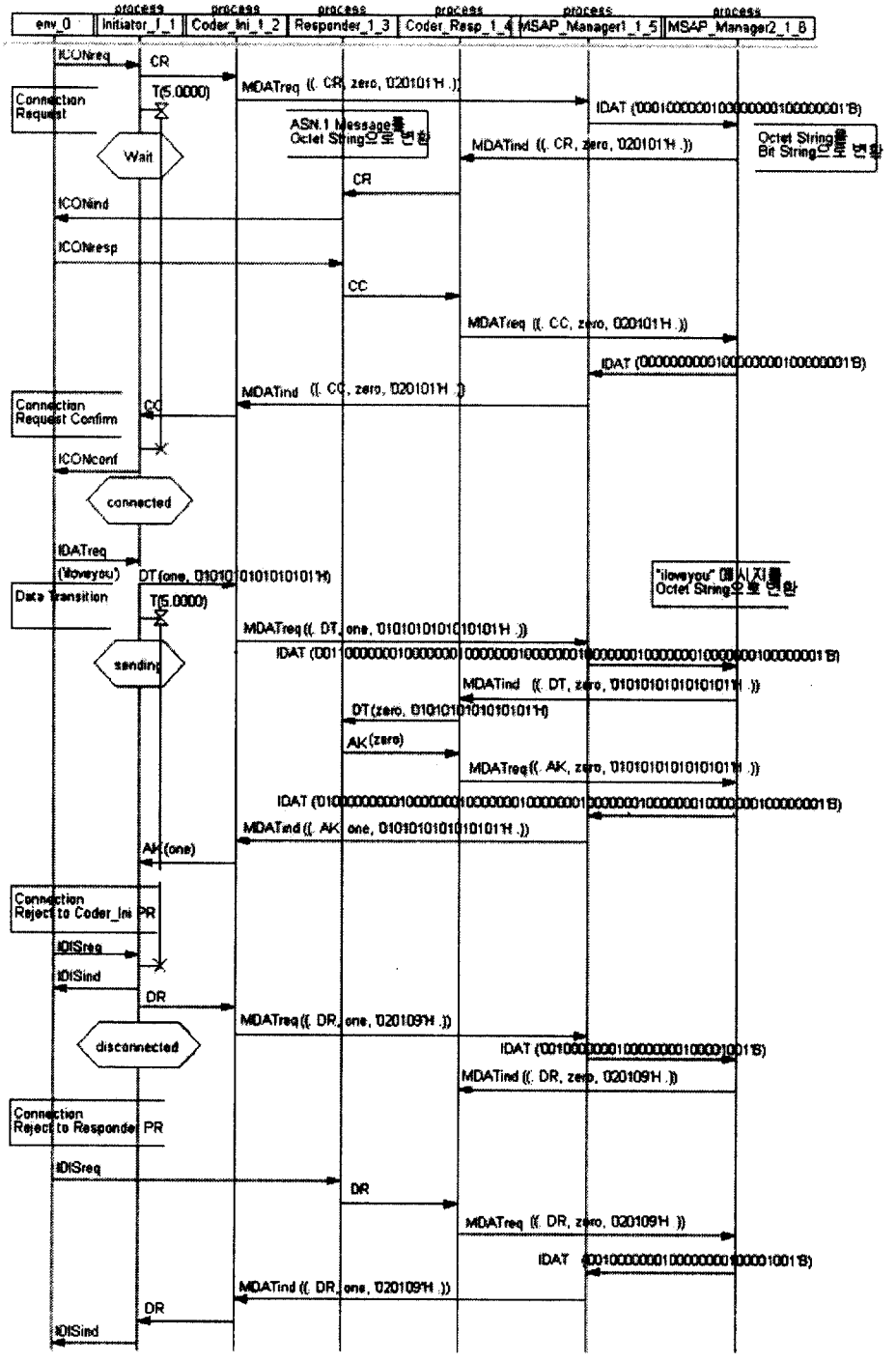


a) 에러 있는 경우



b) 에러 없는 경우

(그림 11) 검증 수행결과



(그림 12) Inres 프로토콜 수행 결과 MSC

매개변수 'sdu'에 인코딩 변환 값을 할당한 후 하위 계층(medium 블록)으로 출력하고 에러 발생으로 실패할 경우 프로세스는 종료된다.

3.5 검증 및 시험

3.5.1 시뮬레이션

SDL 편집기에 의한 SDL 설계 후 analyzer에 의한 SDL 구문 및 의미 분석을 수행하였으며 이 과정에서 잘못 정의 및 선언된 프리미티브, PDU 등에 대한 설계에러를 수정할 수 있다. 또한 make에 의한 코드생성과 시뮬레이션을

수행하여 내재되어 있는 논리에러의 검출과 규격대로 프로토콜이 수행하는 지를 확인 할 수 있다.

3.5.2 검증

시뮬레이션은 사용자가 동작 시나리오 중 시뮬레이터 외부에서의 입력((그림 12)에서 env로 표시)을 제공해 주어야 만 하고 그 경우에 대한 실행결과만을 보여주지만 검증(validation)은 발생할 수 있는 모든 경우의 실행단계를 사용자의 간섭 없이도 전부 추적할 수 있다. TreeWalk 방식에 의한 검증을 Validatior UI를 이용하여 수행한 결과 (그림 11-a)의 report viewer처럼 하나의 'ImplSigCons' 에러와 5개의 TreeWalk가 자동으로 만들어졌다. 'ImplSigCons' (Implicit Signal Consumption) 에러는 현 수신 상태에서 수신된 시그널인 CR을 처리하는 부분이 설계되어 있지 않아 발생하는 검증 특성상의 unspecified reception 경우를 보여준다. 설계 에러를 보완한 후 다시 검증을 수행한 결과 다음 (그림 11-b)의 설계 에러가 없는 TreeWalk를 얻는다.

SDL 설계 후 시뮬레이션과 검증을 수행하여 생성된 MSC가 (그림 12)에 나와 있으며 이 MSC로부터 프로토콜 규격의 MSC인 (그림 4) 동작 시나리오와의 일치성을 검증할 수 있다.

3.5.3 시험

(그림 11)의 TreeWalk 각각은 TTCN에서 활용할 수 있는 시험 케이스로 변환할 수 있다. <표 2>는 (그림 11) TreeWalk에 의하여 생성된 시험 케이스 중 하나인 TC1의 실행결과에 대한 내용으로 연결설정 단계 중 실패에 대한 절차로서 ICONreq 수신시 IDISind 시그널의 발생 여부에 대한 판정결과로써 PASS로 적합성 판정을 내리는 경우를 보여준다.

4. 결 론

국제 표준 언어인 ASN.1, MSC, TTCN 그리고 규격 및 설계 표준 언어인 SDL과 SDL 개발환경인 Tau를 이용하여 통신 프로토콜의 예로 잘 알려진 Inres 프로토콜을 확장 설계하고 시뮬레이션과 MSC 생성에 의한 검증 및 검증 실행결과에 의한 시험 케이스의 생성과 적합성 시험 등 프로토콜 개발 전 단계에 대한 개발방법론을 정립하였다. SDL 환경하에서 프리미티브와 PDU 전송에 의한 계층간 peer-to-peer 통신을 수행하는 통신 프로토콜의 명확한 설계 개념을 정립하였으며, 통신 패킷 전송 시 실제 전송되는 비트 스트링을 구현하기 위한 ASN.1 인코딩/디코딩 모듈을 설계, 프로세스에 추가함으로써 확장된 Inres 프로토콜을 설계하였다. 시뮬레이션과 MSC 생성에 의한 프로토콜의 검증 특성을 검증할 수 있으며, 생성된 MSC로부터 규격과의 적합성을 검증할 수 있는 적합성 시험케이스를 검증 단계시 수행하는 TreeWalk로부터 적합성 시험 언어인 TTCN으로 표현 및 생성하며 이를 토대로 적합성 시험을 수행할 수 있는 방법론을 정립하였다.

본 논문의 기여도는 국제 통신 표준 언어 들을 이용한 통신 프로토콜의 개발 방법 정립에 있으며 이를 위해 설계 및 검증된 확장된 Inres 프로토콜의 SDL 설계문서를 제시하였다. 이를 토대로 3GPP 등 대규모의 복잡한 통신 프로토콜의 안정적이고 효율적인 개발이 가능 하다.[12]

호스트 상에서의 개발단계 후에는 응용시스템별로 light integration 또는 target OS와의 통합인 tight integration에 의하여 실제 환경에서 운영되는 통신 시스템을 개발하게 된다[13].

<표 2>TC1에 대한 Tau-TTCN suite의 실행결과

TTCN Suite					
TS	TC1 in TS [TS.itex]				
TS	TCI Case name		TCI		
-Test Overview	Nr	Label	Behavior Description	Constraints Ref.	Verdict
-Declaration Part	1		ISAP1! ICONreq	cTC1_004	
-Constraints Part	2		ISAPres? ICONind	cTC1_003	
-Dynamic Part	3		ISAPres! IDISreq	cTC1_002	
--Test Case ---TC1 ---TC2	4		ISAP1? IDISind	cTC1_001	PASS

참 고 문 헌

[1] A. Olsen, O. færgemend, B. Møller Pedersen, R. Reed, and J. Smith,"Systems Engineering Using SDL-92," ELSEVIER SCIENCE B. V., 1995.

[2] Jan Ellsbetger, Dieter Hogrefe and Amardeo Sarma, 'SDL Formal Object-oriented Language for Communicating Systems,' Prentice Hall, 1997.

[3] Edel Sherratt, Chris Loftus, "Designing Distributed Services with SDL," IEEE, 2000.

[4] ITU-T, ITU-T Recommendation X.680 | ISO/IEC 8824 ASN.1

[5] ITU-T, ITU-T Recommendation Z.120 MSC

[6] International Standard, ISO/IEC 9646-3, Information Technology - Open Systems Interconnection- Conformance testing methodology and framework - Part3: The Tree and Tabular Combined Notation(TTCN), 1998.11

[7] "TTCN-3: The Ultimate Test, The evolution of a multipurpose standardized test language", white paper, Telelogic, 2002.5

[8] Holzmann, G.J.,"Design and Validation of Computer Protocols," Prentice-Hall, 1991.

[9] Belina, F., Hogrefe, D. and Sarma, A. "SDT with Applications from Protocol Specification," Prentice Hall, 1991.

[10] Telelogic Tau 4.3 Tau-SDL/TTCN Suite user manual, 2001

[11] The SDT Simulator, 'Telelogic Tau 3.4 ORCA and SDT Started,' 1998.

[12] 노철우 "셀방송 서비스 프로토콜의 설계 검증 및 시험방법 연구", 전자통신연구원 최종 연구 보고서, 2001.11

[13] 송평중,노철우,노문환, "SDL 개발환경을 이용한 IMT-2000 무선 프로토콜의 호스트 시뮬레이터 설계 및 구현", 정보처리학회 논문지, 제8-C권 제 2호, 2001.4

노 철 우

1980 서강대학교 물리학과 (학사)

1982 동국대학교 전자계산학과 (석사)

1995 서강대학교 전자계산학과 (박사)

1982-1991 한국전자통신연구소 선임연구원

1991-현재 신라대학교 컴퓨터정보공학부 부교수

관심분야 : 통신 프로토콜 검증 및 성능, 이동통신, 차세대 네트워크

E-Mail : cwro@silla.ac.kr