論文96-33A-6-1

다양한 마크로 처리를 위한 ASN.1 도구 세트의 구현에 관한 연구

(A Study on the Implementation of an ASN.1 Tool Set for various Macro Processing)

金 弘 烈 *, 林 濟 鐸 *

(Hong Ryul Kim and Chae Tak Lim)

요 약

분산 개방 응용시스템들의 프로토콜 명세및 서비스 정의들은 다양한 ASN.1 표기법으로 정의된다. 따라서 메시지 처리 시스템과 같은 개방 응용 시스템의 개발을 위해서는 ASN.1 으로 정의된 프로토콜 데이터들을 그들의 전송구문으로 변환하기위한 부호화 및 복호화 모듈들이 정확히 구성되어야하나, 이러한 작업은 많은 시간이 요구되고 수작업시 많은 오류를 발생시키므로 이를 자동으로 수행하는 소프트웨어 도구가 요구된다.

본 논문에서는 BER/DER 부호화 규칙을 지원하는 새로운 ASN.1 실행 라이브러리 HY BER/DER 과 개선된 ASN.1-to-C 컴파일러인 HYASNC+ 를 포함하는 새로운 ASN.1 도구 세트를 설계 구현하였다. HY BER/DER 실행 라이브러리는 X.209 및 X.690 에 정의된 부호화 규칙을 지원하는 각 ASN.1 타입에 대한 부호화, 복호화 및 다양한 유틸리티 함수들을 포함한다. 개선된 HYASNC+는 X.208 권고에 정의된 기본 ASN.1 타입, 서브 타입 및 X.400 MHS 시스템을 위한 모든 마크로 정의들에 대한 C 언어 인코더/디코더 스텁 파일 및 헤드 파일들을 자동적으로 생성하는 기능을 갖는다. 그리고, HYASNC+ 컴파일러 및 HY BER/DER 실행 라이브러리에 대한 성능 및 호환성 시험을 수행하였다.

Abstract

Protocol specifications and service definitions for distributed open system applications are defined using ASN.1. Therefore, to implement an open system application likes MHS, it is necessary to have well defined encoding/decoding modules which translate ASN.1 protocol specifications into their transfer syntaxes. However, that work is usually tedius, time consuming, and error prone.

In this paper, we designed and implemented a new ASN.1 tool set which includes a new ASN.1 run-time library, called HY BER/DER, and an enhanced ASN.1-to-C compiler, called HYASNC+. HYASNC+ automatically generates C language encoder/decoder stub files and header files for basic ASN.1 types and subtypes defined in X.208 recommandation, and all X400 MHS system macro definitions. And, we evaluated the performance of HYASNC+ compiler and HY BER/DER run-time library, and tested the interoperability of ASN.1 run-time library.

I. 서 론

이기종 컴퓨팅 환경에서는 컴퓨터 시스템에 따라 다양한 데이타에 대한 내부 정보 표현의 방법이 상이한 경우가 많다. 따라서 이들 시스템들의 상호 접속을 위해서는 데이터 타입이나 값 정의에 대해 표준화된 추상 표기법 및 이와 관련된 표준 부호화 규칙이 필수적이다. ITU-T 와 ISO 에 의해 표준화된 ASN.1(Abstract Syntax Notation One)[1][13] 은 분산 개방 시스템들의 상호접속을 위해 널리 사용되는 표준 외부 데이타 추상 표기법이며, BER(Basic Encoding Rules)/DER (Distinguished Encoding Rules)/CER(Ca

* 正會員, 漢陽大學校 電子工學科 竝列處理 硏究室
(Dept. of Elec. Eng., Hanyang University Parallel Processing Lab.)

接受日字: 1996年3月12日, 수정완료일:1996年5月7日

nonical Encoding Rules)[12][14][17] 은 ASN.1 타입 값들에 대한 전송 구문(transfer syntax) 부호화 규칙이다. X.400 MHS[15] 및 X.500 Directory 시스템[16] 과 같은 개방 응용 시스템의 구현을 위해서는 정의된 다양한 ASN.1 프로토콜 데이터들에 대한 부호화 및 복호화 모듈들을 정확하게 구현하여야하는데, 이러한 작업은 많은 시간이 요구되며 또한 많은 오류를 포함한다. 따라서, 이러한 작업을 자동적으로 수행하는 소프트웨어 도구에 대한 요구에 의해 ISODE [18], SNACC [19] 등과 같은 ASN.1 컴파일러들이 구현되었으나, 이들은 인터페이스가 복잡하고, 일부 구현상의 오류도 포함하고 있으며, 기능의 확장이 용의하지 않은 단점이 있다. 본 논문에서는 처리기능이 개선된 ASN.1 to C 컴파일러인 HYASNC+ 와 BER/DER 부호화 규칙을 지원하는 새로운 ASN.1 실행 라이브러리인 HY BER/DER 를 포함하는 새로운 ASN.1 도구 세트를 설계 구현하였다. 개선된 ASN.1 to C 컴파일러인 HYASNC+는 다양한 ASN.1 데이터 명세와 X.400 MHS 시스템을 위한 모든 마크로 정의(macro definitions) 입력에 대한 C 언어 헤드 파일들과 이들에 대한 부호화/복호화 모듈인 BER/DER 인코더 스텁(encoder stub)함수 및 디코더 스텁(decoder stub)함수들을 자동 생성하는 기능을 제공한다. 그리고, 새로운 ASN.1 실행 라이브러리인 HY BER/DER 라이브러리는 정의된 ASN.1 타입 값들을 그들의 전달구문으로 또는 그 역으로 변환하는 라이브러리 함수들과 다양한 유틸리티 함수들을 포함한다.

서론에 이어 Ⅱ 장에서는 ASN.1 과 BER/DER 부호화 규칙에 대해 설명하고, Ⅲ 장에서는 HYASNC+ 컴파일러와 HY BER/DER 실행 라이브러리에서 적용되는 ASN.1 타입과 C 자료구조 간의 변환규칙을 제시하고, Ⅳ 장에서는 새로운 기능을 갖는 HYASNC+ 컴파일러와 HY BER/DER 라이브러리의 설계 및 구현 원리에 대해 기술하고, Ⅴ 장에서는 이 들에 대한 성능 분석 결과를, Ⅵ 장에서는 본 논문에 대한 결론을 정리한다.

Ⅱ. ASN.1 표기법

1. ASN.1 타입 및 값

ASN.1 은 추상 데이터 타입(types) 및 값(values) 들을 명세하기위한 추상구문 표기법 이다. ASN.1 에서는 타입과 값이 상호 연관을 갖는데 하나의 타입은 전송가능한 정보들에 대한 무한 크기를 갖는 값들의 집합이며, 값은 타입의 값 집합내의 하나의 요소이다. ASN.1 타입에는 INTEGER, BOOLEAN, REAL 등과 같은 단순타입(simple types), 구성요소들을 포함하는 SEQUENCE, SET, SET OF, SEQUENCE OF 등과 같은 구조적 타입(structured types), 태깅(tagging)에 의한 태그 타입(tagged types), parent types 에 제약 사항을 가한 서브 타입(subtypes)과 CHOICE, ANY, EXTERNAL 과 같은 타입들이 있다.[11][13] 타입과 값들은 참조를 위한 이름들을 갖으며, 구조적 타입의 구성요소들은 식별자(identifier)을 갖는다. ASN.1 타입 및 값들은 각각의 할당 표기법(assignment notation)에 따라 정의된다. 타입 할당은 타입 참조자, 할당연산자(::=) 및 적절한 타입표기를 통해 이루어지고, 값 할당은 값 참조자, 타입 참조자, 할당연산자에 이어 참조 타입 값 정의를 포함한다. ASN.1 타입 및 값 정의의 예는 <그림 1>과 같다.

```
WeatherReport ::= SEQUENCE {
        stationNumber           INTEGER  (1..9999),
        timeOfReport            UTCTime,
        pressure                INTEGER  (850..1100),
        temperature             INTEGER  ( -100..60),
        windVelocity            INTEGER  (0..500),
        windDirection   INTEGER  (0..48)}

sampleReport   WeatherReport  ::= {
                                   73290,
                                   "9601011200000Z",
                                   1056,
                                    3,
                                   15,
                                   0}
```

그림 1. ASN.1 타입 및 값 정의 예
Fig. 1. Example of an ASN.1 type and value definition.

2. ASN.1 모듈

ASN.1 에서는 정의된 타입 및 값에 대한 참조 범위는 제한되며, 동일한 모듈내에서만 같은 의미를 갖는다. 모듈은 타입과 값들에 대한 관련된 정의들에 대한 집합이다. 따라서 다른 모듈의 타입 및 값에 대한 참조는 import/export 문 정의를 통해서만 이루어질 수 있다. import 문은 키워드 IMPORTS 와 참조할 다른 모듈내의 타입/값 참조자 및 모듈 정보를 포함하며, export 문은 키워드 EXPORTS 와 현재 모듈에서 다른 모듈

에서도 동일한 의미로 참조될 참조자들을 포함한다.

```
WeatherReporting {2 6 6 247 1} DEFINITION ::=
  BEGIN
        EXPORTS WeatherReport, sampleReport;
        IMPORTS Temperature, rainbow
                FROM Weather {2 6 6 247 0};

        WeatherReport ::= SEQUENCE { ..... }
        sampleReport WeatherReport ::= { ..... }
  END
```

그림 2. ASN.1 모듈 정의 예
Fig. 2. Example of an ASN.1 module definition.

동일 모듈 내에 정의된 모든 참조자들은 서로 구분되어야하며, 모듈은 모듈식별자, 키워드 DEFINITIONS, 태그 기본형, 할당연산자 및 모듈 몸체(module body)로 구성되며, 모듈 몸체는 BEGIN 과 END 키워드 사이에 export 문, import 문, 다양한 타입 및 값 할당 정의들을 포함한다. 타입 및 값 정의를 포함하는 ASN.1 모듈정의의 예는 <그림 2>와 같다.

3. ASN.1 태그

이기종 컴퓨터시스템에서 어떤 값에 대한 표현을 정확히 해석하기 위해서는 그 값에 대한 타입을 알아야 한다. CHOICE/ANY 타입 이외의 모든 ASN.1 타입들은 양의 정수 값과 태그 클래스로 구성되는 태그(tag)를 갖는다. 태그 번호가 같은 타입들은 추상적으로 동일하다.[6] X.208에서는 UNIVERSAL, APPLICATION, CONTEXT-SPECIFIC 및 PRIVATE 태그 클래스와 UNIVERSAL 타입들에 대한 태그 값들을 정의하고 있다. UNIVERSAL 태그 이외의 태그 값들은 여러 개방시스템 응용의 프로토콜 명세에서 정의되거나, explicit tagging 또는 implicit tagging 에 의해 얻어진다.[2]

4. 마크로 표기법

ASN.1 에서는 사용자들이 자신 또는 다른 사용자의 사용 목적을 위해 확장 가능한 마크로 표기법 (macro notation)을 제공한다. 사용자의 표기법 확장은 마크로 정의 표기법으로 이루어지며 각 마크로 정의는 마크로 참조자와 타입 및 값 정의에 대한 grammar 들을 포함하고 있다. 마크로 정의들은 타입 및 값 정의 처럼 import 되거나 export 될 수 있다. 마크로는 사용자 자신의 표기법을 갖는 하나의 새로운 ASN.1 타입으로 볼 수 있으며 ASN.1 타입이나 값이 정의될 수 있는 모든 곳에서 정의 가능하다.[1][7]

```
EXTENSION MACRO ::=
 BEGIN
  TYPE NOTATION ::= DataType Critical | empty
  VALUE NOTATION ::= value (VALUE ExtensionType)

  DataType ::= Type Default | empty
  Default ::= "DEFAULT" value | empty
  Critical ::= "CRITICAL FOR" CriticalityList | empty
  CriticalityList ::= Criticality | CriticalityList "," Criticality
  Criticality ::= "SUBMISSION" | "TRANSFER" |
                  "DELIVERY"
 END -- of EXTENSION
```

그림 3. 마크로 정의 예
Fig. 3. Example of a macro definition.

마크로 정의는 <그림 3>과 같이 마크로 참조자, 키워드 MACRO, 할당연산자(::=) 및 키워드 BEGIN, END 사이에 타입과 값에 대한 production set 을 포함하는 마크로 몸체로 구성된다. 마크로 정의 구문들은 동적 확장(dynamic extension)이 가능한데, 컴파일러는 일반적으로 role base 로 구현되므로 동적 확장 지원이 용의하지 않다. 또한, 새로운 마크로 정의들의 syntax reduction 이 무한하게 될 수도 있으며, 일반적으로 마크로 정의는 단순히 확장 구문만 제공하므로, 컴파일러 구현시 이들에 대한 구문(syntax) 및 의미(semantics)를 정확히 파악하여 설계 구현하여야 한다.[7][11]

5. BER 부호화 규칙

BER 부호화 규칙은 이기종 시스템들간의 데이터 전송을 위해 각 ASN.1 타입 값들을 어떻게 8 비트의 옥테트 스트림(octet stream)으로 부호화하는가를 규정한다. BER 부호화에서는 값의 타입과 값의 길이를 아는지 여부에 따라 프리미티브 한정길이 부호화, 구조적 한정길이 부호화 및 구조적 무한길이 부호화 방법과 같은 3 가지 방법을 규정한다. 이들 부호화 방법중 구조적 무한 길이 부호화 방법은 ASN.1 타입 값의 길이를 알수 없는 경우에만 적용되고 나머지 두방법은 값의 길이를 아는 경우의 부호화에만 적용된다.[11][7]

Identifier octets(T)	Length octets(L)	Contents octets(V)	End of contents octets(EOC)

그림 4. BER 부호화의 구조
Fig. 4. Structure of an encoding in BER.

각 부호화 방법에서는 <그림 4> 와 같은 구성요소들을 포함한다. 앞의 3 가지 요소는 항상 포함되며 마지막의 EOC 는 선택적이다. "식별자 옥테트"는 부호화된 ASN.1 타입 식별에 사용되며, "Class P/C Tag" 의 3 가지의 필드로 구성된다. "Class" 는 태그클래스를 나타내며 2 비트로 부호화되고, "P/C" 는 부호화되는 값이 단순타입인지 또는 구조적타입인지를 나타내며 1 비트로 부호화되고, "Tag" 는 태그번호의 크기가 30 보다 작은 경우 5 비트로 부호화되나, 31 이상인 경우 2 개 이상의 옥테트로 구성된다. "길이 옥테트"는 한정길이부호화에서는 내용 옥테트의 길이를 나타내며, 하나 또는 그 이상의 옥테트로 구성된다. 무한길이부호화에서는 길이가 무한함을 나타내며 "0x80" 값을 갖는 하나의 옥테트로 구성된다. "내용 옥테트"는 프리미티브 한정길이부호화에서는 오직 값들의 옥테트로 구성되며, 구조적 타입인 경우는 값 구성요소들의 BER 부호화의 연결을 나타내며, 명시적 태깅에의해 유도되는 타입인 경우는 기반 값의 BER 부호화를 나타낸다. "내용끝 옥테트"는 구조적 무한길이 부호화에서만 사용되며 내용 옥테트의 끝을 나타내며 그 값은 "0x80 0x80" 이다.

6. DER 부호화 규칙

X.690[8] 에서는 새로운 ASN.1 부호화 규칙인 DER 부호화 규칙을 정의한다. DER 은 BER 부호화 규칙의 서브 세트로 디지탈 서명이 ASN.1 값으로 연산되는 경우와 같이 유일한 부호화가 요구되는 응용을 위해 새롭게 표준화되었다. DER 부호화 규칙들은 BER 과 유사하나 다음과 같은 제약이 적용된다.

(1) 길이 옥테트

최소의 옥테트 수로 부호화 하기 위해 한정길이 부호화 방법으로만 부호화 한다. 즉, 무한길이 부호화는 사용되지 않는다.

(2) 스트링 부호화 형태

BIT STRING, OCTET STRING 및 제한된 문자 스트링 값들의 부호화에는 구조적 부호화를 적용하지 않는다.

(3) SET 구성요소

SET 타입의 구성요소들의 값들의 부호화는 다음과 같은 태그 순서에 따라 부호화한다. UNIVERSAL, APPLICATION, CONTEXT- SPECIFIC 및 PRIVATE 태그 타입의 구성요소 순으로 부호화 하며 각 태그 클래스 타입에서는 태그 번호의 오름차순으로 부호화 한다.

(4) 기타 제약 사항들

1) 부울 타입의 값이 "TRUE" 경우는 내용 옥테트는 "0xFF" 으로 부호화 한다.

2) 비트 스트링의 부호화에서는 마지막 옥테트의 unused bit 들은 반드시 "0" 으로 세트 한다.

3) SET 또는 SEQUENCE 구성요소 값이 기본값(default value)인 경우는 부호화 하지 않는다.

4) 기타 제약 사항들은 X.690[8] 을 참조한다.

Ⅲ. ASN.1 과 C 타입 변환규칙

모든 ASN.1 데이타 타입과 C 타입간의 변환은 단순한 직접 매핑으로 변환 가능하나, 몇몇 ASN.1 타입들에 대해서는 변환을 위한 사용자 정의 C 자료 구조형이 요구된다. <표 1> 은 본 논문에서 제안한 ASN.1 과 C 와의 변환 규칙을 보여준다. 이들 변환 규칙들은 임의의 ASN.1 명세에 대한 C 언어 헤드 파일 및 인코더/디코더 스텁 파일 생성시 적용된다. INTEGER, REAL 및 OCTET STRING 과 같은 단순 타입들은 <표 1> 에 나타난 long int, double 및 char * 와 같은 C 타입들로 직접 변환이 가능하다.

표 1. ASN.1 과 C 와의 타입 변환
Table 1. Type conversion between ASN.1 and C.

ASN.1 types	C data types
BOOLEAN	char
OCTET STRING	char *
OBJECT IDENTIFIER	struct HYOID { int oidlen; long *oidval; };
INTEGER	long int
REAL	double
IA5String	unsigned char *
BIT STRING	struct BITS { int bitlen; char *bitval; };
SEQUENCE, SET	struct { ..... };
SEQUENCE OF, SET OF	struct xxx { struct YYY *parm; struct xxx *next; };
CHOICE	struct { int offset; union {.....} un; };

OBJECT IDENTIFIER 는 C 타입 HYOID * 으로 변환되며 oidval 는 오브젝트 식별자 구성 요소를 저장하고, oidlen 는 오브젝트 식별자의 길이에 대한 정보를 저장한다. BIT STRING 은 C 타입 BITS * 로 변환되며 bitval 는 비트 스트링 데이터 값들을 저장하고, bitlen 는 비트 스트링의 길이에 대한 정보를 저장한다. 단순 타입과 달리 구조적 타입들은 사용자 정의 C 자료 구조로 변환이 가능하다. SET 및 SEQUENCE 타입은 단순 타입 및 구조적 타입의 구성 요소들을 포함하는데, 단순 타입 구성 요소들은 <표 1> 에 정의된 C 타입들로 직접 변환되고, 구조적 타입 구성 요소들은 사용자 정의 자료구조 struct { ..... } 의 pointer 변수로 변환된다. SEQUENCE 타입에서의 "COMPONENTS OF Type" 정의는 임의의 Type 을 포함(inclusion)시키는 경우에 사용된다. 여기서 "Type" 은 반드시 SEQUENCE 타입이어야 한다. 따라서 COMPONENTS OF 에 대한 C 자료구조 생성 시에는 포함될 타입의 자료구조를 참조하여야만 하는데 이에 대한 변환 규칙에 대한 예는 <그림 5> 와 같다.

ASN.1 타입	변환된 C 자료구조

모듈명 : Sample
Bee ::= SEQUENCE {
ttt INTEGER,
COMPONENTS OF Flower
}
 	
Flower ::= SEQUENCE {
aaa INTEGER,
bbb BOOLEAN
}
	struct Sample_Bee {
	int ttt;
	int aaa;
	char bbb;
	};
	struct Sample_Flower {
	int aaa;
	char bbb;
	};

그림 5. COMPONENTS OF 타입 변환의 예
Fig. 5. Example of COMPONENTS OF type translation.

SET OF/SEQUENCE OF 타입은 하나의 타입을 반복적으로 정의할 때 사용된다. 이들의 구성 요소는 단순 타입 또는 구조적 타입이 될 수 있으며 이들 타입은 단순 타입이나 구조적 타입의 링크드-리스트 C 데이터 타입의 pointer 로 변환된다. CHOICE 타입은 구성 요소중 오직 하나의 타입 만 선택하는 경우에 사용되며, 이들 구성 요소들은 단순 타입이나 구조적 타입을 포함할 수 있다. 이들은 구성 요소 타입의 상대적 위치에 대한 offset 을 갖는 사용자 정의 union 타입으로 변환된다. 또한, 구조적 타입의 모든 구성 요소들은 식별자를 가져야만 타입들간의 모호함을 완전히 배제할 수 있다. 그러나, 식별자가 생략된 경우에도 타입 참조명만으로도 타입 상호간의 식별이 가능하다. 하지만, C 언어에서는 식별자가 존재하지 않는 ASN.1 타입에 대해, 이에 대응하는 C 자료형은 반드시 식별자를 가져야 하므로 자동적으로 식별자를 생성해 주는 메카니즘을 제공한다.[15]

Ⅳ. ASN.1 도구 세트의 설계 및 구현

ASN.1 도구 세트의 HYASNC+ 컴파일러는 모든 ASN.1 타입 정의에 대한 BER/DER 부호화 및 복호화를 위한 C 언어 스텁 파일 및 헤드 파일들을 자동적으로 생성하며, 또한 MHS 시스템에서 요구되는 다양한 마크로 타입 및 값 정의에 대한 확장 처리가 가능하며, HY BER/DER 실행 라이브러리는 X.209[12] 및 X.690[14] 에 따른 부호화, 복호화 및 다양한 유틸리티 함수들을 포함하도록 약 35,000 라인의 C 언어로 구현하였다. HYASNC+ 컴파일러 및 HY BER/DER 실행 라이브러리는 UNIX(System V 및 BSD) 와 Windows 95 환경에서 운용 가능하다.

1. HYASNC+ 컴파일러

HYASNC+ 컴파일러는 <그림 6>과 같이 7 개의 기능 모듈로 구성되며, 기존의 ASN.1 컴파일러[8][9] 와 달리 중간 코드 생성 없이, 헤드화일 및 인코더/디코더 스텁 파일들을 직접 생성하도록 처리과정을 단순화 하였다. "사용자 인터페이스 모듈"은 ASN.1 입력 및 옵션 선택을 위한 인터페이스를 제공하며, UNIX 용은 명령 라인(command line) 인터페이스를 제공하며, PC 용은 Windows 기반의 대화형 인터페이스를 제공한다.

"어휘분석기 모듈"은 ASN.1 입력 모듈 내에 있는 ASN.1 키워드 및 식별자들을 구분하여 구문 분석을 위한 토큰(token)들을 발생하는 기능을 수행한다. 현재 어휘분석기는 X.208, X.219, X.407, X.411, X.413, X.420 및 X.509 에 정의된 모든 ASN.1 키워드 및 식별자들에 대한 어휘를 분석하여 해당 토큰을 생성하도록 GNU flex 를 사용하여 설계 및 구현하였다. "구문분석기 모듈" 은 어휘분석기에서 입력되는 토큰 스트링들이 X.208, X.219, X.407, X.411, X.413, X.420 및 X.509 에 정의된 ASN.1 구문과 일치 하는지를 체크

하고, 구문에 일치하면 헤드 및 스텁 파일 생성을 위한 파싱 트리를 생성하도록, GNU bison 을 사용하여 설계 및 구현하였다. "헤드 및 인코더 스텁생성기 모듈"은 ASN.1 타입 및 값 정의에 대한 파싱정보를 이용하여 C 데이터 자료구조를 생성하고, 이들에 대한 인코더 스텁 함수를 생성한다. "디코더 스텁생성기 모듈"은 ASN.1 타입에 대한 디코더 스텁 함수를 생성한다.

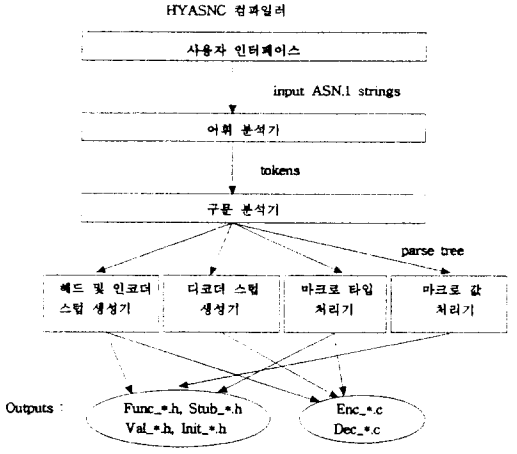

그림 6. HYASNC+ 컴파일러 구조
Fig. 6. Structure of HYASNC+ compiler.

표 2. HYASNC+ 에의해 처리 가능한 마크로
Table 2. Processable macros by HYASNC+.

권고	마크로 정의
X.219	OPERATION, ERROR, APPLICATION CONTEXT, APPLICATION SERVICE ELEMENT
X.407	OBJECT, PORT, ABSTRACT-BIND, ABSTRACT UNBIND, ABSTRACT OPERATION, ABSTRACT-ERROR
X.411	EXTENSIONS, EXTENSION, EXTENSION ATTRIBUTE, TOKEN, TOKEN DATA, SECURITY-CATEGORY
X.413	ATTRIBUTE, AUTO-ACTION
X.420	HEADING-EXTENSION, EXTENDED BODY PART-TYPE
X.509	ALGORITHM, SIGNED, SIGNATURE, ENCRYPTED

"마크로 타입처리기 모듈"은 마크로 타입 정의에 대한 파싱정보를 이용하여 헤드 파일, 인코더 및 디코더 스텁 함수들을 생성한다. 마크로 타입 정의는 동적으로 확장되기 때문에 모든 ASN.1 마크로 타입 정의를 범용적으로 처리할 수 있는 컴파일러의 구현은 불가능하다.[7][11] 따라서, 본 연구에서는 <표 2> 에 정의된 24 개의 마크로 타입 정의에 대한 동적 확장 구문을 구문분석기에 내장하였고, 이들에 대한 헤드, 인코더 스텁, 디코더 스텁 파일들을 생성하도록 구현 하였다. 향후 이외의 마크로 정의에 대해서도 확장이 용의 하도록 설계 구현하였다. "마크로 값 처리기 모듈"은 마크로 값 정의에 대한 파싱정보를 이용하여 헤드 파일, 인코더 및 디코더 스텁 함수들을 생성한다. 임의의 ASN.1 입력에 대해 HYASNC+ 컴파일러는 다음과 같은 파일들을 생성한다. <그림 7>은 HYASNC+ 컴파일러 출력화일들이 어떻게 사용자 프로그램 에서 사용되는지를 보여준다.

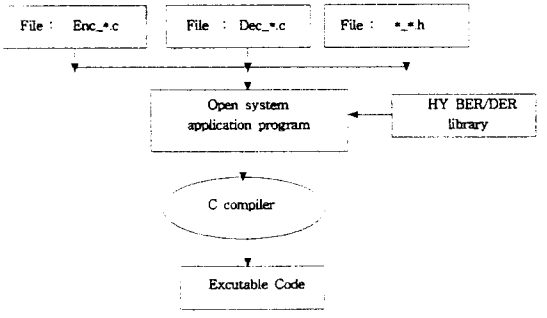

그림 7. HYASNC+ 출력 파일의 사용
Fig. 7. Using HYASNC+ outputs.

"인코더 스텁 파일(Enc_*.c)"은 입력 ASN.1 데이터 타입들을 BER/DER 옥테트 스트링으로 변환하기 위한 인코딩 라이브러리 함수들의 스텁들을 포함한다. (여기서, "*" 는 ASN.1 입력 파일명을 나타낸다.) "디코더 스텁 파일(Dec_*.c)"은 BER/DER 옥테트 스트링으로 부터 ASN.1 타입 데이터들을 추출하기 위한 디코딩 라이브러리 함수들의 스텁들을 포함한다. "헤드 화일(Func_*.h, Stub_*.h, Val_*.h, Init_*.h)"들은 인코더/디코더 스텁화일 및 사용자 프로그램에서 참조되는 관련 자료구조 및 초기 값들에 대한 정보를 포함하고 있다. Func_*.h 는 인코딩/디코딩 스텁 함수에 대한 함수 원형을, Stub_*.h 는 ASN.1 타입에 대응되는 C 자료형을, Val_*.h 는 ASN.1 값 정의에 대한 C 자료형을, 그리고 Init_*.h 는 초기화 데이터에 대한 C 자료형을 포함하는 헤드 파일이다.

2. HY BER/DER 실행 라이브러리

HY BER/DER 실행 라이브러리는 HYASNC+ 컴

파일러가 생성하는 인코더 스텁 및 디코더 스텁 함수에서 요구되는 ASN.1 단순 타입에 대한 공통 라이브러리 함수 루틴 및 다양한 유틸리티 함수들을 포함한다. 설계 구현된 HY BER/DER 라이브러리 루틴들은 <표 3>과 같다. 구조적 타입들은 이들의 조합으로 구성된다.

(1) BER/DER 인코딩 라이브러리 루틴

BER/DER 인코딩 라이브러리 루틴들은 옥테트 복잡도가 최소가 되도록 한정길이 부호화 방법만 지원하도록 동일한 함수들로 설계 구현하였다.

(2) BER 디코딩 라이브러리 루틴

BER 디코딩 라이브러리 루틴들은 한정길이 및 무한길이 부호화 방법으로 인코딩된 옥테트 스트링을 모두 처리하도록 설계 구현하였다.

(3) DER 디코딩 라이브러리 루틴

DER 디코딩 라이브러리 루틴들은 BER 디코딩 라이브러리와 달리 오직 한정길이 부호화 규칙에 따라 인코딩된 옥테트 스트링만을 처리 하도록 설계 구현하였다. 따라서, BER 복호화시 부호화 방법에 대한 검색에 소요되는 시간 및 무한 길이 옥테트에 대한 옥테트 길이 파악에 많은 시간이 소요되는 단점을 피할 수 있어 성능 개선이 기대된다.

표 3. HY BER/DER 라이브러리 루틴들
Table 3. HY BER/DER library routines.

BER/DER Encoding routines	BER Decoding routines	DER Decoding routines
EncBool()	DecBERBool()	DecDERBool()
EncInt()	DecBERInt()	DecDERInt()
EncBitStr()	DecBERBitStr()	DecDERBitStr()
EncOctStr()	DecBEROctStr()	DecDEROctStr()
EncNull()	DecBERNull()	DecDERNull()
EncReal()	DecBERReal()	DecDBERReal()
EncNumStr()	DecBERNumStr()	DecDERNumStr()
EncPrintStr()	DecBERPrintStr()	DecDERPrintStr()
EncTeletexStr()	DecBERTeletexStr()	DecDERTeletexStr()
EncVideotexStr()	DecBERVideotexStr()	DecDERVideotexStr()
EncIA5Str()	DecBERIA5Str()	DecDERIA5Str()
EncGraphicStr()	DecBERGraphicStr()	DecDERGraphicStr()
EncVisibleStr()	DecBERVisibleStr()	DecDERVisibleStr()
EncGeneralStr()	DecBERGeneralStr()	DecDERGenStr()
EncUTC()	DecBERUTC()	DecDERUTC()
EncGenTime()	DecBERGenTime()	DecDERGenTime()
EncOID()	DecBEROID()	DecDEROID()
EncODE()	DecBERODE()	DecDERODE()
EncAny()	DecBERAny()	DecDERAny()
EncETR()	DecBERETR()	DecDERETR()

(4) 유틸리티 함수

HY BER/DER 라이브러리에서 공통적으로 사용되는 유틸리티 함수 루틴들은 <표 4>와 같으며, 이들은 응용 프로그램 작성에 자주 요구되는 함수들이다.

표 4. 유틸리티 함수들
Table 4. Utility functions.

유틸리티 함수	기 능
HYalloc ()	HY 동적 메모리 할당
HYfree ()	HY 동적 메모리 해제
SeqAdd ()	SEQNENCE 타입의 구성요소 연결
SetAdd ()	SET 타입의 구성요소 연결
SeqofAdd ()	SEQNENCE OF 타입의 구성요소 연결
SetFind ()	HY 리스트에서 SET 타입 확인
datechk ()	윤년 및 날짜 관련 처리 함수
HYError ()	에러 메시지 출력 함수
HYOidCmp ()	Object Identifier 비교
HYdebugTest ()	Encoded Octet string 에 대한 디버깅
PrintInputArg ()	입력 데이터의 화면 출력
HY2Stream ()	HY 리스트를 옥테트 스트림으로 변환
Octet2HY ()	옥테트 스트림을 HY 리스트로 변환

V. 성능 평가

1. HYASNC+ 성능 평가

HYASNC+ 컴파일러는 X.208 에 정의된 모든 기본 ASN.1 타입/값 정의, 서브타입 및 MHS 시스템의 프로토콜에서 요구되는 다수의 마크로 타입 및 값 정의에 대한 처리 기능을 제공한다. HYASNC+ 와 기존의 ASN.1 컴파일러인 ISODE 8.0[8], SNACC[9] 과의 처리 기능 비교표는 <표 5>와 같다.

표 5. ASN.1 컴파일러의 처리 기능 비교
Table 5. Comparison of ASN.1 compiler's processing capbilities.

ASN.1 컴파일러 / ASN.1 타입	HYASNC+	ISODE 8.0	SNACC
Simple types	○	○	○
Structured types	○	○	○
Subtypes	○	×	×
ASN.1 values	○	×	×
Macro types and Macro values	X.400 관련 20 macros 및 X.509 관련 4 macros	7 macros	5 macros

HYASNC+ 는 1) single value, cont'd subtype, value range, size range, alphabet limitation 및 inner subtyping 과 같은 서브타입 정의에 대한 처리 기능이 지원되며, 2) 모든 ASN.1 값 정의에 대한 헤드 파일 생성기능과, 3) 단순 ASN.1 타입의 기본 값 정의 처리기능 및, 4) X.400 시리즈 및 X.509 관련 마크로 타입 및 마크로 값 정의에 대한 헤드 파일, 인코더 스텁 파일 및 디코더 스텁 파일의 생성 기능이 추

가로 지원된다. <표 5>와 같이 HYASNC+ 의 처리 기능이 다른 컴파일러에 비해 개선되었음을 알 수 있다.

2. HY BER/DER 실행 라이브러리 성능 평가

ASN.1 실행 라이브러리의 성능은 임의의 ASN.1 입력에 대한 부호화된 옥테트 복잡도(octet complexity), 인코딩 및 디코딩 소요시간, 인코더/디코더 모듈의 정확성으로 평가 가능하다. HY BER/ DER 실행 라이브러리의 성능 평가를 위해 다음과 같은 ASN.1 타입을 선택한다.[11][15]

1) X.208 에 정의된 PERSONNEL RECORD 타입
2) X.411 에 정의된 MessageTransfer ABSTRACT OPERATION 마크로 정의

성능 평가는 Solaris 1.0/SUN Sparc-20 호환 기종 환경에서 HY BER/DER 라이브러리와 ISODE 8.0 라이브러리를 상호 비교 평가하였고, 또한 설계 구현된 컴파일러와 라이브러리의 기능성 및 무결성을 검증하기 위해 NCC(National Computing Center)사의 MHS Conformance Test Tool[14] 을 사용하여 호환성 시험(interoperability test)를 수행하였다.

표 6. 옥테트 복잡도 비교
Table 6. Comparison of octet complexity.

Number of PERSONLEL RECORDs	Encoded octets number using HY BER Encoder	Encoded octets number using ISODE BER Encoder
1	139	141
10	1,364	1,374
50	6,804	6,854
100	13,604	13,704
250	34,004	34,254
300	40,804	41,104
500	68,005	68,504
1,000	136,005	137,004

옥테트 복잡도에 대한 성능 평가 결과 <표 6>과 같이 한성길이 부호화 방법으로 구현된 HY BER/DER 인코딩 라이브러리가 무한 길이 부호화 방법을 사용하는 ISODE 8.0 실행 라이브러리 보다 우수함(즉, 옥테트 복잡도가 낮음)을 확인하였다.

PERSONNEL RECORD 타입에 대한 BER 인코딩/디코딩 소요시간 성능 평가 결과는 <그림 8>과 같다. ISODE 8.0 과의 인코딩/디코딩 소요시간에 대한 성능 평가 결과 인코딩 성능은 유사하나, 디코딩 성능은 무한길이 부호법을 적용하는 ISODE 에 비해 성능이 우수하였다.

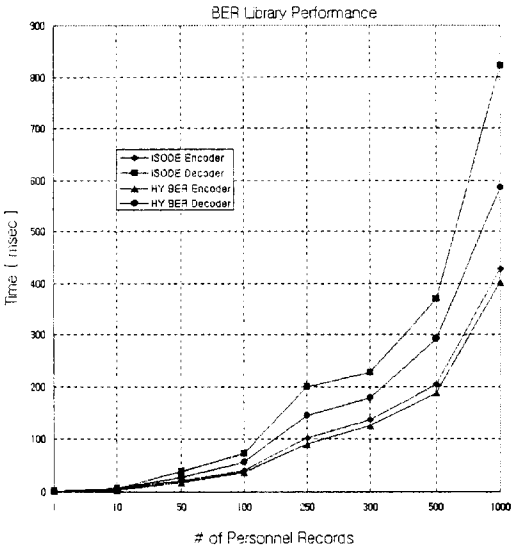

그림 8 BER 라이브러리 성능 평가
Fig. 8. Performance evaluation for BER libraries.

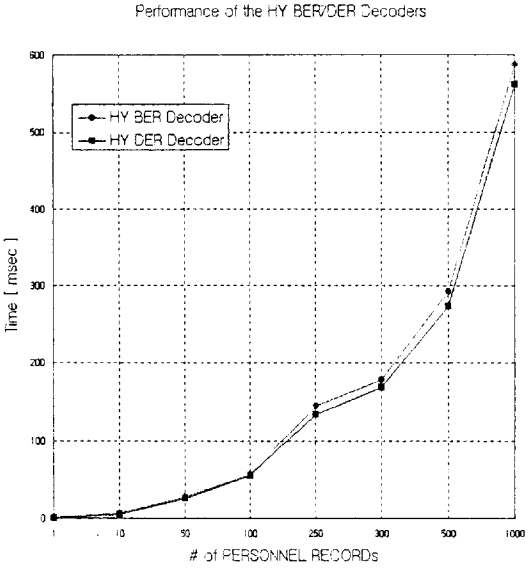

그림 9. BER-to-DER 라이브러리 성능 평가
Fig. 9. Performance evaluation for BER-to-DER libraries.

PERSONNEL RECORD 타입에 대한 DER 인코딩/디코딩 소요시간에 대한 성능 평가는 현재 구현된 비교 대상 DER 라이브러리가 없어 HY BER 라이브러리와 상대적 평가를 수행하였고, HY BER/DER 인

코딩 라이브러리는 동일하므로 복호화 라이브러리에 대해서만 평가 하였으며, 그 결과는 <그림 9>와 같다.

DER 과 BER 디코딩 라이브러리의 성능 평가 결과 DER 복호화 라이브러리의 성능이 개선 되었음을 확인 하였다.

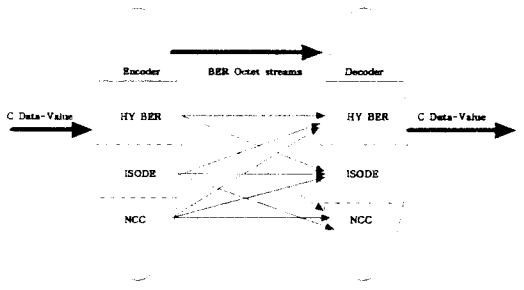

그림 10. 호환성 시험
Fig. 10. Interoperability test.

HY BER/DER 라이브러리에 대한 기능적 무결성을 검증하기 위한 호환성 시험은 HYASNC+, ISODE 8.0 및 NCC conformance test tool[11]을 사용하여 테스트 데이터에 대한 인코더/디코더 모듈을 생성한 후, <그림 10>과 같이 각각의 인코딩 옥테트 스트링들을 상호 복호화하였다.

표 7. BER 라이브러리에 대한 호환성시험 결과
Table 7. Interoperability test results for BER libraries.

Encoder \ Decoder	HYASNC	ISODE	NCC
HYASNC	○	△	○
ISODE	△	○	△
NCC	○	△	○

호환성 시험 결과는 <표 7>과 같으며, HYASNC+ 컴파일러가 자동 생성한 인코더/디코더 스텁화일들과 설계 구현된 HY BER/DER 실행 라이브러리의 무결성을 확인하였다. 호환성 시험중 ISODE 는 ASN.1 데이터의 TagDefault 가 "IMPLICIT TAGS" 인 경우에 대한 처리, CHOICE 타입의 Tag 처리, OCTET STRING 타입 처리에 있어서 다수의 설계 구현상 오류를 발견하였다.

VI. 결 론

본 논문에서는 MHS 시스템 및 개방 시스템의 표현 계층 상위 프로토콜 구현에 있어 필수적인 도구인 개선된 ASN.1 도구 세트를 설계 구현하였다. HYASNC+ 컴파일러는 다양한 ASN.1 타입 및 값 정의에 대한 헤드, 인코더 스텁 및 디코더 스텁 파일을 자동 생성하며, 기존의 ASN.1 컴파일러에서 지원하지 못했던 ASN.1 서브 타입, 마크로 타입 및 값 처리 기능을 설계 구현하였다. 또한, 보다 개선된 성능을 갖는 BER 실행 라이브러리와 디코딩 처리 시간을 보다 더 단축할 수 있는 DER 실행 라이브러리를 포함하는 HY BER/DER 실행 라이브러리를 설계 구현하였으며, 이들의 성능 개선 및 기능성을 성능 평가를 통해 검증하였다. 또한 현재까지 개방 시스템의 응용 시스템 및 표현 계층 상위 프로토콜 구현에 있어 표준 도구로 사용하던 ISODE 컴파일러 및 라이브러리의 설계상 오류를 발견하였다. 향후 연구과제로는 보다 많은 마크로 정의 처리 기능의 추가 설계 구현 및 새롭게 표준화된 부호화 규칙인 CER 및 PER 실행 라이브러리의 설계 구현 및 성능 개선이 요구된다.

감사의 글

※ 본 연구를 지원해 주신 한국통신 S/W 연구소 서원균 실장님, 강정주씨와 현대전자 임홍순 상무님, 김기창 선임께 감사드리며, 본 연구는 한국통신 및 현대전자 지원과제로 수행되었음을 알려드립니다.

참 고 문 헌

[1] ITU-T, Recommendation X.208 : Specification of Abstract Syntax Notation One, 1993.

[2] ITU-T, Recommendation X.209 : Specification of Basic Encoding Rules for Abstract Syntax Notation One(ASN.1), 1993.

[3] ITU-T, Draft new Recommendation X.680 : Information technology - Abstract Syntax Notation One(ASN.1): Specification of Basic Notation, 1994.

[4] ITU-T, Draft new Recommendation X.690 : Information technology - ASN.1 Encoding Rules: Specification of Basic Encoding Rules(BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules

(DER), 1994.

[5] ITU-T, Data Communication Networks Message Handling Systems : Rec. X.400 - X.420, 1989.

[6] ITU-T, Recommendation X.509 : The directory System - Authentification Frame work, 1993.

[7] D. Steedman, ASN.1 : The Tutorial and Reference, West Linton, 1990.

[8] M. T. Rose, The ISO Development Environment : User's Manual Vol 1. - 5, Performance Systems International, Inc., July 18, 1991.

[9] M. Sample and G. Neufeld, Snacc 1.0 : A High Performance ASN.1 to C/C++ Compiler, Feb. 1993.

[10] B. S. Kaliski, "A Layman's Guide to a Subset of ASN.1, BER, and DER", RSA Data Security, Inc., June 3, 1991.

[11] G.W. Neufeld and Y. Yang, "The Design and Implementation of an ASN.1-C Compiler", IEEE Trans. on Software Engineering, Vol. 16, No. 10, Oct. 1990.

[12] A.T. Schreiner and H.G. Friedman, "Introduction to Compiler Construction with UNIX", Prentice-Hall, Inc., 1985.

[13] A. I. Holub, "Compiler Design in C", Prentice-Hall, Inc., 1990.

[14] NCC, 88mhs Test System Release No. 2.0 Tests and Testing Guide Issue 4 draft A, The National Computing Center limited, 1995.

[15] 김 홍 렬, 임 제 탁, "개방 시스템 응용을 위한 개선된 ASN.1 컴파일러 설계 및 구현", 대한전자공학회 논문지, 제 33 권 A 편 제 3 호, pp.398-407, 1996

저 자 소 개

金 弘 烈(正會員) 第 33卷 A編 第 3號 參照
현재 한양대학교 전자공학과 박사 과정

林 濟 鐸(正會員) 第 33卷 A編 第 3號 參照
현재 한양대학교 전자공학과 교수