ASN. 1支持工具与应用层网络协议开发环境

戴珂

(通信测控技术研究所,石家庄 050081)

ASN. 1 SUPPORT TOOLS AND NETWORK APPLICATION PROTOCOL DEVELOPMENT ENVIRONMENT

Dai Ke

(*The Communication, Telemetry and Telecontrol Research Institute, Shijiazhuang* 050081)

Abstract Abstract Syntax Notation One (ASN. 1) plays an important role in description and development of OSI upper network protocol. The structure of ASN. 1 support system and OSI application protocol development environment (DE) based on ASN. 1 are presented in this paper, finally, some protocols developed in the DE are given.

摘要 抽象语法表示ASN. 1在OSI网络高层协议的描述与开发方面起着重要的作用,本文介绍了ASN. 1支持工具的构成及基于ASN. 1的OSI网络应用层协议的开发环境,并给出了开发实例.

§1. 引 言

近年来,我们相继承担了CCITT X. 400 MHS(文电作业系统)和CCITT X. 500 DS(目录服务)等课题的研究,MHS主要是在开放环境下实现的电子邮件服务,为了为MHS提供寻址功能[7],又开发了X. 500服务,根据OSI标准层次结构的划分,DS和MHS都属于应用层服务功能.

OSI应用层协议用以控制通信应用进程间信息的交互过程,构成所谓的开放系统. 同一个协议可能是由不同的部门和不同的开发者并行开发,所开发的协议与标准的一致性程度首先要取决于对协议规范的共同理解,这就要求对协议的描述必须精确而无二义性. 理论和实践已经证明:采用形式化描述技术(FDT)是一种最佳的选择. 通过抽象的描述,使协议的开发者和维护者能清楚地理解实体的行为,减少设计失误和便于维护. 作为一种形式化描述语言,ASN. 1特别适用于应用层协议的描述. 与网络低层的情况不同,应用层信息是直接面向用户的,因此所要传送的信息都采用可读的形式,如报文或其它文件等. 大多数情况下所传送信息

本文1990年7月23日收到,1991年1月7日定稿. 作者戴珂,工程师,1989年硕士毕业于西安电子科技大学,主要研究领域为OSI高层网络协议、软件工具.

的数据结构也相当复杂，如报文的的信头还应包括收端和发端的地址、通知收端应采取的应答方式、设置报文的有效期、路径信息等，而每种信息本身还可能含有更为复杂的数据结构，如何能唯一性地表示这些复杂的数据结构成为应用层协议开发的关键问题.

综上所述，OSI 高层协议的开发主要是解决两方面的问题：首先是应用层协议数据单元 PDU(Protocol Data Unit)的唯一性定义问题，其次是每次通信过程中所使用的 PDU 的一个例样(instance)如何按照国际标准编码以便在通信线路上传送，针对这两方面的要求，ISO 制定了两个有关的国际标准：ISO 8824 和 ISO 8825[4][5]. 前者规定了 ASN.1 的语法规则，后者给出 ASN.1 的基本编码规则. 目前 OSI 应用层协议全都是用 ASN.1 描述的，即在协议规范中给出各种 PDU 的 ASN.1 定义，由此可见，建立面向 ASN.1 的翻译机制是应用层协议实现的关键，这种翻译机制除了能够将 PDU 的 ASN.1 描述形式的例样直接转换为符合 ISO 8825 规定的编码形式外，还能够对 ASN.1 描述的合法性进行检查.

基于上述思想，我们开发了支持 ASN.1 的编码器、解码器、语法检查器、制导编辑器、代码调试器和协议测试器等，这些工具组合组成了一个网络应用层协议的开发环境.

目前，国内外的一些大学和研究机构已开发出各种 ASN.1 的支持工具，如美国的 NBS、加拿大的 MONTREAL 大学等，国内一些单位在这方面也做了大量的工作，有些已经形成综合的网络协议开发环境，其核心工具是 ASN.1 的编/解码器.

考虑到这类软件的价格、对硬件和操作系统的要求以及可移植性等因素，根据实际工作的要求，我们开发了一套 ASN.1 完整的支持系统，可运行于 SUN 工作站、VAX 小型机和 PC 286、PC 386 等机型，可运行的操作系统包括 VMS、UNIX、DOS 等多种环境，其工具的组合方便，可支持应用层网络协议的设计、实现和测试等过程.

§ 2. 网络应用层 PDU 的 ASN.1 描述与编码方法

OSI 应用层协议主要表现为一组 PDU 的交互过程，与网络低层不同，应用层 PDU 不是二进制的码流，而是表现为可读形式的各种字符和符号构成的数据结构. 由应用层的特点决定，这种数据结构通常是比较复杂的，包括集合、序列、记录等，通常还要用到由嵌套和递归形式定义的更为复杂的结构，为了能够描述任意复杂结构的 PDU，OSI 在应用层采用了 ASN.1 描述.

ASN.1 形式上类似 Backus-Naur 文法，主要提供各种数据类型(type)的定义. 除简单类型(simple type)外，数据结构通常被定义为各种结构类型(structure type)，根据各种数据结构组成的特点(由简单类型构成的复杂类型)，在 ASN.1 中采用这样一种结构化的定义技术：即首先定义一些简单的类型，如整型(INTEGER)、布尔型(BOOLEAN)、比特型(BIT STRING)和八位位串(OCTET STRING)等. 有了这些定义类型，就可以由生成式(production)进一步定义任意复杂的数据结构.

如下面给出一个关于文件逻辑描述(logical description)PDU 的 ASN.1 定义：

```
LogicalDescriptor DEFINITIONS::=
  BEGIN
    LogicalDescriptor::=SEQUENCE {
      LogicalObjectType,
```

```
LogicalDescriptorBody }
LogicalObjectType::=INTEGER {document (0), paragraph (1) }
LogicalDescriptorBody::=SET {
pageHeading [3] IMPLICIT T61String OPTIONAL,
presentationDirectives [5] IMPLICIT PresentationDirectives OPTIONAL }
  PresentationDirectives::=SET {
    alignment [0] IMPLICIT Alignment OPTIONAL,
    graphicRendition [1] IMPLICIT GraphicRendition OPTIONAL }
    Alignment::=INTEGER { leftAlignment(0), centred(2), justified(3)}
      GraphicRendition::=SEQUENCE OF GraphicRenditionAspect
      GraphicRenditionAspect::=INTEGER
END
```

2.2 ASN.1的编码的方式

使用ASN.1可以定义一个PDU的逻辑结构,对于所定义PDU的一个例样,称为该PDU的一个值(value),ASN.1要求类型定义必须与给定的值相一致,下面给出一个PDU的完整定义:

```
Connect_PDU::=SEQUENCE {
 myAddress          NetworkAddress,
 yourAddress        NetworkAddress,
 reverseCharging    BOOLEAN,
 userData [UNIVERSAL 6] IMPLICIT OCTET STRING }
 NetworkAddress::=OCTET STRING
```

这一定义描述了Connect_PDU的结构,即说明该PDU中含有以八位位串表示的网络地址,一个布尔类型用以说明付费情况是否被传送回来,最后给出一个用户提供的数据,每个类型都给予一个名字(如myAddress、reverseCharging、userData等).

该PDU的一个例样(值)可以表示为:

```
overture Connect_PDU::= {
 myAddress          "The Communication Research Institute",
 yourAddress        "China Computer Software Company",
 reverseCharging    TRUE,
 userData           "Let's talk" }
```

每个值由值定义(value definition)指定一个名字,类型定义和值定义结合起来构成一个模块(module),通常一个协议中的类型定义和值定义放在一个模块中.

由此可见,以ASN.1描述的PDU结构清晰,可读性强,但要对这种以可读形式表示的PDU在通信线路上传送,需将其译成二进制的编码形式.这就要求有一个编码方式,例样的每个类型都对应着一个称为数据元素(data element)的八位位串的编码.在编码规则中,每个数据元素必须说明三个要素:类型标识符(identifier)、被编码值的长度(length)和值的内容(content),如overtrueConnect_PDU的编码结果如下:

编 码		说 明	
30	56	SEQUENCE	结构类型,总长86个八位位串
04	24	"The Communication Research Institute"	八位位串类型,长度36个八位位串
04	1F	"China Computer Software Company"	八位位串类型,长度31个八位位串
01	01	FF	布尔类型 TRUE
06	0A	"Let's talk"	八位位串类型,长度10个八位位串

值得说明的是,目前国际标准只给出 ASN.1的基本编码方式,将来很可能还会有其它编码方式,如压缩和加密等.

§3. ASN.1工具系统的构成与编/解码器的生成过程

3.1 词法分析器和语法分析器的生成

在 ASN.1支持工具中包括两种编译器:ASN.1到某种可执行语言的编译器和 ASN.1的编/解码器.编译程序一般包括词法分析、语法分析、代码生成和优化等过程,词法分析器的设计和生成利用了 UNIX 环境下的 LEX 工具,LEX 接受词法的正则式描述,产生一个 C 语言的词法分析程序,这一过程使得词法分析器的设计和实现大大简化,其修改也十分方便.

ASN.1的词法分析器主要完成两方面的任务:(1)对 ASN.1描述语法进行静态检查.(2)生成类型语法树(下简称类型树).类型树以代码的生成为目的,系统中其它工具都要用到所建立的类型树,对某一 PDU 所建立的类型树是该 PDU 的一种内部表示形式,因此,系统中其它工具对类型树应具有可操作性,语法分析器是利用 UNIX 环境下的 YACC 工具自动生成的.作为一个例子,对 Connect_PDU 生成的类型树如图1所示.

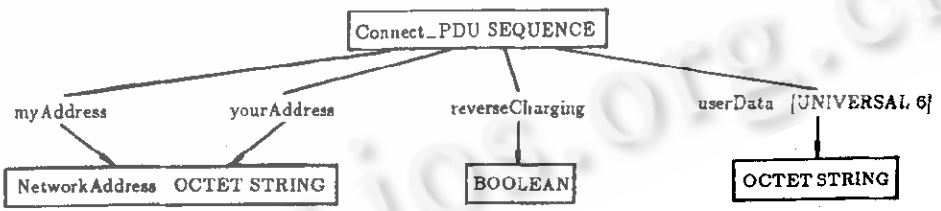

图1

3.2 ASN.1到 C 语言的编译器(ASNTOC)

由于利用了 LEX 和 YACC,考虑到其它 ASN.1工具(均用 C 语言编写)对类型树的可操作性,在对 ASN.1描述的 PDU 编码之前,应首先将其变成 C 语言的结构形式,通过 ASN.1到 C 的转换过程,使得所描述的 PDU 能够嵌入到 ASN.1工具中,为编码器的生成建立基础,这种转换可用下例说明.

```
ASN.1定义:
PDU::=
        SEQUENCE {…
            error-status
```

```
        INTEGER {
                noError(0),
                tooBig(1),
                noSuchName(2),
                badValue(3),
                readOnly(4),
                    },
    ...
        }
```

所生成的C结构经优化后具有下列形式:

```
struct type_SNMP_PDU {
#define int_SNMP_error_status_noError          0
#define int_SNMP_error_status_tooBig           1
#define int_SNMP_error_status_noSuchName       2
#define int_SNMP_error_status_badValue         3
#define int_SNMP_error_status_readOnly         4
};
```

3.3 ASN.1编码器/解码器和工具库

经过ASN.1到C的转换,将用C描述的PDU结构与ASN.1工具库(ASNT.LIB)相链接,生成ASN.1编码器和解码器.工具库中主要包括一些固定功能的模块,如静态接口的定义、类型与值的合法性检查、值树结点的分配与对应、标准文本的打印、错误处理及代码转换等,为实现快速生成,将这些模块放在一个库中.

ASN.1编码器根据生成的类型树,产生相应的值树,这样在树的每个结点上,都有类型与值的对应关系,如在3.1中给出的Connect_PDU的类型树,其值树如图2所示,经ASN.1编码器编码后,就得到PDU的内部编码,解码器则执行相反的过程.

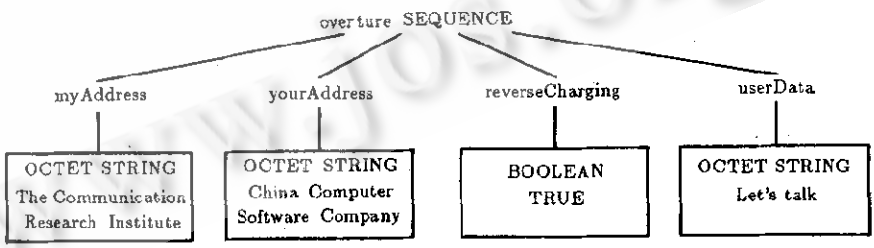

图2

§4. 基于ASN.1的应用层协议开发环境

4.1 开发环境

综上所述,ASN.1编/解码器支持这样一个过程:给出一个PDU的ASN.1描述,将ASNTOC编译器作用于该PDU,产生其C描述,具此系统将产生该PDU的编码器(ENCODER)

和解码器(DECODER),ENCODER 可将该 PDU 的一个例样转换成二进制码流,DECODER 将其还原成 ASN.1的描述形式.

由此可见,以 ASN.1编码器和解码器为基础可以建立一个网络应用层协议的开发环境,当然还要构造表示层乃至会话层的原语仿真接口.

基于上述思想,我们建立了一个 OSI 应用层协议的开发环境,在该环境下,除提供原语仿真外,还根据实际需要开发了 ASN.1的编辑器(ASNEDITOR)、代码调试器及 ASN.1测试器(ASNTESTER),这些工具的组合构成一个比较完善的应用层协议开发环境,如图3所示.

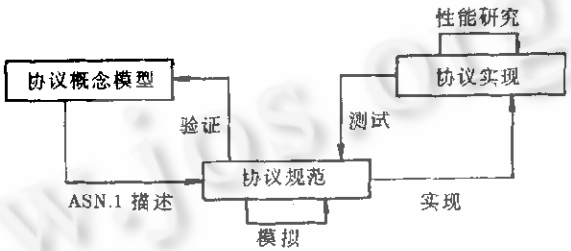

图3

4.2 ASNEDITOR

由于 ASN.1的语法规则比较复杂,协议的开发者必须按 ASN.1规定的格式定义 PDU 和输入 PDU 的例样,为克服这一不足,我们设计了一种交互式的编辑器——ASNEDITOR,在定义一个 PDU 的例样时,根据菜单的提示首先输入 PDU 的名字、各域(field)和子域的类型及标识符,编辑器根据输入的信息自动构成 PDU 的 ASN.1描述,如编辑器发现不是 ASN.1的基本类型而且也没有定义过,则将其视为结构类型,提示用户进一步定义.此外编辑器还可以对已定义的 PDU 的某些域进行修改、插入和删除,PDU 的例样输入也是根据定义给出相应的提示.

4.3 代码调试器和测试器

代码调试器用于对 OSI 应用层协议的 IUT(Implementation Under Test)测试,根据具体协议生成测试例样库,交互式测试器可对 IUT 进行基本功能测试、编/解码测试及服务元素的支持性测试.图4给出 MHS 中端系统协议(P1、P2)测试示意图.将一 PDU 例样编码后送给 IUT,并从 IUT 接收返回信息,将该信息送到分析器,分析结果显示出来,测试器还可以连续自动完成测试和分析.

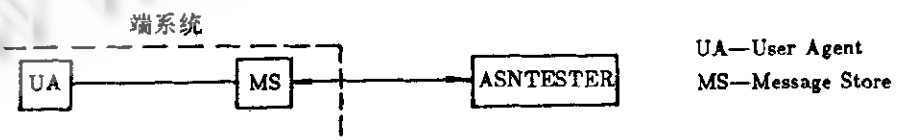

图4

4.4 开发实践

我们利用这一开发环境,在 PC-XT/AT、VAXII、SUN 工作站构成的环境下开发了 X.500协议,其层次结构如图5所示[7].其中的编码器和解码器就是由 ASN.1支持工具自动生

成的,在通信期间将其维护在表示层中.此外,我们还在这一环境开发了MHS的部分协议.

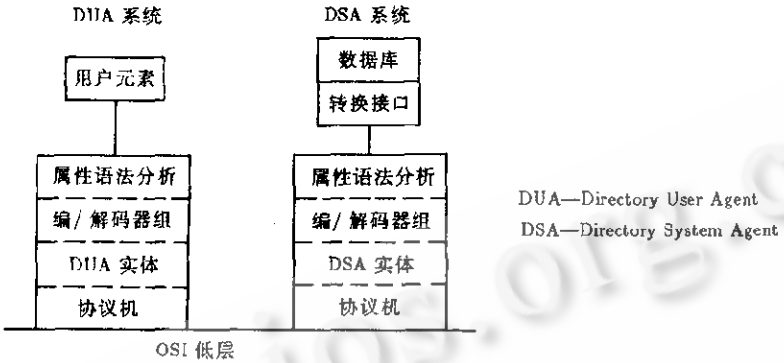

图5

结束语:综上所述,ASN.1支持系统所提供的各种工具,可以支持OSI应用层协议的设计、实现和测试过程,并以此构成强有力的OSI应用层协议开发环境,在该环境下可以高效、准确地开发出高质量的协议,比较圆满地解决了工程中存在的一些实际问题.

参考文献

[1]戴珂,X.500目录服务的研究,数据通信,1990.3.

[2]戴珂,MHS与X.500目录的互联实现,数据通信,1990.3.

[3]ISO 7498 Information Processing System——Open System Interconnection——Basic Reference Model.

[4]ISO 8824 Information Processing System——Open System Interconnection——Specification of ASN.1 (1988).

[5]ISO 8824 Information Processing System——Open System Interconnection——Specification of Basic Encoding Rules for ASN. 1 (1988).

[6]戴珂,基于ASN.1的应用层协议开发环境,通信技术,1990.3.

[7]戴珂,网络名字服务器的实现与应用,数据通信,1990.1.