SNMP/OSI管理ゲートウェイにおけるGDMO定義の変更に対する汎用性の実現と評価

1０－5

黒木哲也　堀内浩規　杉山敬三　小花貞夫

国際電信電話株式会社　研究所

1．はじめに

筆者等は、既存の TMN(電気通信管理網)装置を、何ら変更を加えずに、SNMP(Simple Network Management Protocol)マネージャから監視／制御を可能とする SNMP／OSI 管理ゲートウェイ(以下、単にゲートウェイと呼ぶ)を実装した[1, 2]。ここでは種々の TMN 装置に対応可能とするため、TMN 装置毎に異なる管理情報定義(GDMO 定義)に対して汎用性をもたせる方式を採用した[3]。本稿では、この方式の実現と評価結果を述べる。

2．SNMP／OSI管理ゲートウェイの概要

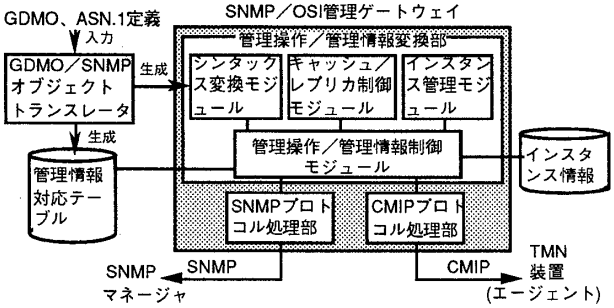

図１ SNMP／OSI管理ゲートウェイのシステム構成

ゲートウェイ(図１)は、① CMIP(共通管理情報プロトコル)及び SNMP プロトコル処理部、② SNMP マネージャと TMN のベースとなる OSI 管理の間で管理操作と管理情報を対応づけるための情報を格納した管理情報対応テーブル、③ TMN 装置の管理オブジェクト(MO)インスタンスの情報を保持するインスタンス情報、④NMF (Network Management Forum) の IIMC (ISO/CCITT and Internet Management Coexistence)[4]における管理情報定義の対応づけ規則に基づき、SNMP と OSI 管理の間で管理操作と管理情報の変換を行う管理操作／管理情報変換部、⑤ GDMO／SNMP オブジェクトトランスレータから構成される。このうち④は、さらに、⑥ MO の属性の値と SNMP オブジェクトの値との変換を行うシンタックス変換モジュール、⑦シンタックス変換モジュールと管理情報対応テーブル、インスタンス情報を利用して管理操作／管理情報変換を行う管理操作／管理情報制御モジュール、⑧管理操作／管理情報制御モジュールから呼出され、管理操作の発行回数を削減するためのキャッシュ、レプリカ等の処理を行うキャッシュ／レプリカ制御モジュール⑨ TMN 装置からの MO 生成／削除通知等を用いてインスタンス情報を維持・管理するインスタンス管理モジュールから構成される。

3．GDMO 定義の変更に対する汎用性の実現

GDMO 定義に依存する、管理情報対応テーブルと、管理操作／管理情報変換部のシンタックス変換モジュールは、GDMO 定義の異なる様々な TMN 装置を、ゲートウェイに容易に収容するため、GDMO／SNMP オブジェクトトランスレータ(図２)により GDMO 定義とそこで参照する ASN.1 定義から自動的に生成するようにした。ITU-T 勧告 X.721 の MO クラス alarmRecord への適用例を以下に述べる。

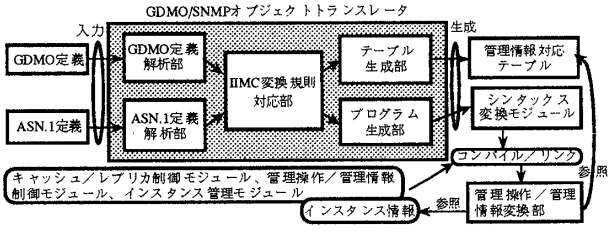

図２ GDMO/SNMP オブジェクトトランスレータ

3．1 生成した管理情報対応テーブル

MO クラス alarmRecord について生成した管理情報対応テーブルを図３に示す。このテーブルには、SNMP マネージャの操作中の SNMP オブジェクトのオブジェクト識別子(OID)から、対応する TMN 装置の管理操作と MO クラスを識別するための、①SNMP オブジェクトの OID(図中央：smi2AlaRecCorNotSourceObjectInst(4)等)と②その SNMP オブジェクトの種別(図中央：属性型 correlatedNotifications 等)、および③ MO クラス(図上：CLASS(alarmRecord))の情報が含まれる。さらに、②の種別が属性型(correlated Notifications)の場合には、その SNMP オブジェクトと対応する TMN 装置の MO の属性値を取得/設定するために必要な情報、④属性の識別子(図下：correlatedNotifications, 2.9.3.2.7.12)とそのシンタックス(図下：ICorrelatedNotifications)等の情報が含まれる。

```
+-smi2alarmRecord(1)                                          : CLASS(alarmRecord)
  +-smi2AlarmRecordTable(1)                                   : TABLE
    +-smi2AlarmRecordEntryNumber(1)                           : NUMBER
    +-smi2AlarmRecordEntry(2)                                 : ENTRY
      :
    +-smi2AlaRecCorrelatedNotifications(10)                   : SIDE-TABLE(3)
      :
  +-smi2AlaRecCorrelatedNotificationsTable(3)                 : TABLE
    +-smi2AlaRecCorrelatedNotificationsNumber(1)              : NUMBER
    +-smi2AlaRecCorrelatedNotificationsEntry(2)               : ENTRY
      +-smi2AlaRecCorNotTableIndex(1)                         : INDEX
      +-smi2AlaRecCorNotFlag(2)                               : FLAG
      +-smi2AlaRecCorNotCorrelatedNotifications(3)            : SIDE-TABLE(4)
      +-smi2AlaRecCorNotSourceObjectInst(4)                   : correlatedNotifications
      :
  +-smi2AlaRecCorNotCorrelatedNotificationsTable(4)           : TABLE
      :
    +-smi2AlaRecCorNotCorNotNotificationIdentifier(3)         : correlatedNotifications

loggingTime              2.9.3.2.7.59  iLoggingTime              N  OOOOOOOOOO
managedObjectClass       2.9.3.2.7.60  IObjectClass             N  OOOOOOOOOO
managedObjectInstance    2.9.3.2.7.61  IObjectInstance          N  OOOOOOOOOO
notificationIdentifier   2.9.3.2.7.16  INotificationIdentifier  N  OOOOOOOOOO
correlatedNotifications  2.9.3.2.7.12  ICorrelatedNotifications Y  XOOXOO-OOO
```

図３ 管理情報対応テーブルの例

3．2 生成したシンタックス変換モジュール

alarmRecord の属性 correlatedNotifications のシンタックスについて生成したシンタックス変換モジュール(SNMP オブジェクトの値の取得)は、図４に示すように、①インターフェース部、②シンタックス変換部、③ライブラリ部のプログラムから成る。

インターフェース部は、入力される情報(SNMP オ

Realization and Evaluation of Flexibility in GDMO Definitions in SNMP/OSI Management Gateway
Tetsuya KUROKI, Hiroki HORIUCHI, Keizo SUGIYAMA and Sadao OBANA KDD R&D Labs.

ブジェクトの OID、属性のシンタックス、属性の値等)のうち属性のシンタックスが ICorrelatedNotifications の場合には、対応するシンタックス変換部のプログラム stxCvt_DecCorrelatedNotifications の呼出しを行う。シンタックス変換部には、属性 correlatedNotifications のシンタックスが図5に示す ASN.1 の構造形で定義されることから、(a)stxCvt_DecCorrelatedNotifications と(b)stxCvt_DecCorNotCorNotNotificationIdentifier の2つのプログラムが生成される。このうち(a)は、インターフェース部からの呼出しにより、SNMP オブジェクトの OID からメンバを識別してその値を取出す。さらに、そのメンバが correlatedNotifications の場合には(b)を、また sourceObjectInst の場合にはライブラリ部を呼び出して、取出した値を SNMP オブジェクトのシンタックスと値へ変換する。また、(b)は(a)から呼出され、OID によりメンバ correlatedNotifications が保持する繰り返し型構造の値から1つの NotificationIdentifier 形(INTEGER 形)の値を識別し、ライブラリ部を呼出してこの値を変換する。ライブラリ部のプログラム(c)stxCvt_DecObjectInstance と(d)EncINTEGER は、シンタックス変換部から渡された値について、(c)は ObjectInstance 型から Object Identifier 型のシンタックスと値へ、(d) は INTEGER 型相互間の変換を行う。

```
/********* インターフェース部プログラム(SNMP オブジェクトの値の取得時の変換) ********/
int  stxCvt_DecMain( stxId,oid,cmipVal,snmpVal,mode )
int    stxId; /* 属性のシンタックス情報 */   OBJECT_IDENTIFIER   *oid ; /* OID */
STRING  *cmipVal ; /* 属性値 */   STRING  *snmpVal ;/* SNMP オブジェクトの値 */
int    mode ; /* 1:NUMBER オブジェクト, 2:FLAG オブジェクト, ・・・ */
{  ERROR  error ;   void  *val ;   int  r ;   BUFFER  asnBuff ;
  decode(&val,cmipVal,&asnBuff,stxId,&error) ;
  /* 属性のシンタックスに対応するシンタックス変換部のプログラムを識別し起動する*/
  switch(stxId) {  case  INotificationIdentifier : /* NotificationIdentifier 型 */
          r = stxCvt_DecNotificationIdentifier(0,oid,val,snmpVal,&error,mode) ;   break ;
   case  ICorrelatedNotifications : /* CorrelatedNotifications型 */
          r = stxCvt_DecCorrelatedNotifications(0,oid,val,snmpVal,&error,mode) ;break ;
    /* 省略 */
  default :  return(FALSE) ;  }   if( r == FALSE )return(FALSE) ;   return(TRUE) ;  }
```

```
/**** シンタックス変換部プログラム(ASN.1型 : CorrelatedNotifications 型の変換行う) ****/
int  stxCvt_DecCorrelatedNotifications(idx,oid,val,snmpVal,error,mode)
int    idx ;   OBJECT_IDENTIFIER  *oid ;   CorrelatedNotifications *val ;
STRING   *snmpVal ;   ERROR  *error ;   int  mode ;
{  int  i ;   void  *valx ;   int  size ;   size = val->size ;
  if( (mode == 1) && (oid->data[idx] == 3) ) {  /* NUMBER オブジェクトの値の取得 */
      if(EncINTEGER(&size,snmpVal)==FALSE){ error->errlevel = 2 ; return(FALSE) ; }
      return(TRUE) ;  }   for(i=0;i<=oid->data[idx+4];i++){ reml_buf(val,&valx) ; }
  /* SNMP オブジェクトの OID と対応するメンバを識別する */
  switch(oid->data[idx]){ /* テーブルレベルの OID 値で識別する */
   case  3 : /* メンバ : sourceObjectInst */
      if(stxCvt_DecObjectInstance(val->sourceObjectInst,snmpVal
       ,error)==FALSE){ error->errlevel = 2 ;  return(FALSE) ; }  break ;
   case  4 : /* メンバ : correlatedNotifications */
     if(stxCvt_DecCorNotCorNotNotificationIdentifier(idx,oid,val->correlatedNotifications
       ,snmpVal,error,mode)==FALSE){error->errlevel = 2 ;  return(FALSE) ; }  break;
   default :   error->errlevel = 2 ;  return(FALSE) ;   }   return(TRUE) ;  }
/******** CorrelatedNotifications 型のメンバ : correlatedNotifications の変換を行う ********/
int  stxCvt_DecCorNotCorNotNotificationIdentifier(idx,oid,val,snmpVal,error,mode)
int    idx ;  OBJECT_IDENTIFIER  *oid ;  SetOfNotificationIdentifier  *val;
STRING   *snmpVal ;   ERROR  *error ;  int  mode ;  { int  i; void  *valx;
  for(i=0;i<=oid->data[idx+5];i++){ reml_buf(val,&valx) ;}
  if(EncINTEGER(valx,snmpVal)==FALSE){ error->errlevel = 2 ;  return(FALSE) ; }
 return(TRUE) ;  }
```

```
/*ライブラリ部プログラム*/ /*ObjectInstance 型から OID(ObjectIDentifier)型へ変換を行う*/
int  stxCvt_DecObjectInstance(val,snmpVal,error)
ObjectInstance  *val ;   STRING  *snmpVal ;   ERROR  *error ;
{  static OBJECT_IDENTIFIER oidNo={(1,2,124,360501,15,0,0,3,1,0,1,0},14,0) ;
   static OBJECT_IDENTIFIER objId ;
  memcpy( &objId,&oidNo,sizeof(OBJECT_IDENTIFIER));
  objId.data[6] = GWGetRecmId(val) ;  objId.data[8] = GWGetRegId(val) ;
  if( encode(&objId,snmpVal,IObjectIdentifier)==FALSE ){
     error->errlevel = 2 ;  return(FALSE) ;  }   return(TRUE) ;  }
/*************************** INTEGER 型の値の符号化を行う ***************************/
int  EncINTEGER(val,synVal)  INTEGER  *val ;   STRING  *synVal;
{  if( encode(val,synVal,IInteger)==FALSE){ return(FALSE) ; }  return(TRUE) ;  }
```

図4 シンタックス変換モジュールの例

```
CorrelatedNotifications  ::= SET OF SEQUENCE {
      correlatedNotifications  SET OF NotificationIdentifier,
      sourceObjectInst         ObjectInstance  OPTIONAL  }
NotificationIdentifier  ::=  INTEGER
```

図5 属性 correlatedNotifications のシンタックス

4. 評価と考察

4.1 管理操作／管理情報変換の処理時間

生成した管理情報対応テーブルとシンタックス変換モジュールを含む、管理操作／管理情報変換部の処理時間を測定した(表1)。処理時間は、同変換部が、SNMP プロトコル処理部から管理操作を受信して CMIP プロトコル処理部へ変換した管理操作を発行するまでの時間と、CMIP プロトコル処理部から管理操作に対する応答を受信して SNMP プロトコル処理部へ変換した応答を発行するまでの時間の合計である。全測定項目とも、プログラムを自動生成した場合と手作りした場合(括弧内の数値)で、処理時間に差がないことから、効率的な処理を行うプログラムを自動生成できたと言える。

表1 管理操作／管理情報変換の処理時間

測定項目	操作	処理時間
①属性のシンタックスの ASN.1型が単純形の場合。 単純形(AdministrativeState)	値の取得	35.2 (34.8) ms
	値の設定	25.4 (25.1) ms
②属性のシンタックスのASN.1型が構造形の場合。 構造形(CorrelatedNotifications)のメンバ(sourceObjectInst)	値の取得	39.2 (38.7) ms
	値の設定	47.5 (47.1) ms
③ASN.1型が構造形でさらにそのメンバが構造形の場合。 構造形(CorrelatedNotifications)のメンバ(correlatedNotifications)	値の取得	44.5 (43.1) ms
	値の設定	49.8 (49.1) ms

注1) MO クラスは ITU-T 勧告 X.721 の"alarmRecord"と"log"を使用した。

注2) 処理時間の括弧内の数値は、プログラムを手作りした場合の処理時間を示す。

注3) 計算機には SUN670MP(CPU は 1 個)を使用。

4.2 プログラム規模

一般的に使用される MO クラス、属性等を含む ITU-T 勧告 X.721, M3100, Q821, Q822 の属性のシンタックスから47種類の ASN.1 型を抽出して、この変換を行うシンタックス変換モジュールのプログラム規模を測定した。自動生成したシンタックス変換モジュールの規模は約4.5Kステップであり、比較的コンパクトなプログラムを自動生成できた言える。

また、GDMO／SNMP オブジェクトトランスレータのプログラム規模は約9.2Kステップであった。この内訳は、GDMO 定義と ASN.1 定義の解析部がそれぞれ約2.4Kと5Kステップ、IIMC 変換規則対応部が約0.3Kステップ、テーブル生成部とプログラム生成部は、それぞれ約0.5Kと1Kステップであった。

5. おわりに

本稿では、SNMP/OSI 管理ゲートウェイにおける GDMO 定義の変更に対する汎用性の実現について述べた。また、自動生成したシンタックス変換モジュール等を含む管理操作／管理情報変換部の処理時間、およびプログラム規模の観点から評価し、その有効性を示した。最後に、日頃ご指導頂く KDD 研究所村上所長に感謝します。

参考文献

[1]:堀内, 黒木, 杉山, 小花, 鈴木"SNMPによるTMN装置の監視／制御のためのSNMP/OSI管理ゲートウェイの実装", 情処研究会資料、DPS 72-9, Sept. 1995.

[2]:堀内, 黒木, 杉山, 小花 "SNMP/OSI管理ゲートウェイの実装と評価", 情処第53回全大、1Aa-03, Mar. 1996.

[3]:黒木, 堀内, 杉山, 小花 "SNMP/OSI管理ゲートウェイにおけるGDMO定義に対する汎用性の実現方式", 情処第53回全大、1Aa-04, Mar. 1996.

[4]:NM Forum "Forum 030 : Translation of ISO/CCITT GDMO MIBs to Internet MIBs", Oct. 1993.