マルチメディア通信と分散処理 84－9
（1997. 9. 12）

CMIPボード用XMP/XOMインタフェースの実装

杉山敬三　吉原貴仁　堀内浩規　小花貞夫

国際電信電話株式会社

既存のCMIPボードにXMP/XOMインタフェースを持たせるためのCMIPボード用XMP/XOMインタフェースソフトウェアの実装について述べる．ワークスペースの実装では，OMオブジェクトをXOM関数の処理に適した形で管理し，また既存のモジュールが出力するOMパッケージのファイルから情報を抽出してOMパッケージを管理した．XMP/XOM関数とプリミティブの対応付けでは，OMオブジェクトとCMIPボードのプリミティブのデータ構造が類似しているため，容易に対応付けられる．ASN.1自動符号化/復号機能は，既存のGDMOパーザ/インタプリタを利用することで効率的に実現した．評価の結果，実装したソフトウェアが実用的な性能を有することを確認した．

Implementation of XMP/XOM Interface Software for CMIP Board

Keizo SUGIYAMA　Kiyohito YOSHIHARA　Hiroki HORIUCHI　Sadao OBANA

KOKUSAI DENSHIN DENWA CO., LTD.

This paper describes the implementation of XMP/XOM interface software for the existing CMIP board. We discuss the realization method for XOM workspaces, mapping between XMP/XOM functions and CMIP board primitives and an automatic encoding/decoding. As to XOM workspace, OM objects are maintained so as to make the processing of OM functions easy and OM packages are managed by using OM package files derived from the existing module. We find out that the mapping is easy because of the similarity of data structure between OM objects and CMIP board primitives. The automatic encoding/decoding is efficiently realized by using the existing GDMO parser/interpreter. Through the performance evaluation, we show that the software can be practically used.

1. はじめに

TMN(電気通信管理網)の標準化の進捗に伴い，OSI(開放型システム間相互接続)管理に準拠した網管理システムの構築が進んでいる．OSI管理では，通信プロトコルとしてCMIP(共通管理情報プロトコル)[1]を用い，マネージャとエージェント間でGDMO(Guidelines for the Definition of Managed Objects)[2]により定義される管理情報を交換する．筆者等は，これまでにCMIPを含むOSI 7層のプロトコル処理を行う既存のCMIPボード[3,4]を利用し，各種網管理アプリケーションの開発を行った．

このCMIPボードでは，CMIS(共通管理情報サービス)に対応した独自のプリミティブをAPI(応用プログラムインタフェース)として提供している．一方，オープンシステムに関する標準化組織であるX/Openでは，CMIS及びインターネットの管理用プロトコルであるSNMP (Simple Network Management Protocol)の両方に適用可能なAPIとして，XMP (X/Open Management Protocols)[5]/XOM (X/Open OSI-Abstract-Data Manipulation)[6]を規定しており，XMP/XOMに対応したアプリケーションが普及しつつある．そこで筆者等は，CMIPボード上で動作するアプリケーションの拡大を図るため，CMIPボードにXMP/XOMインタフェースを持たせるためのCMIPボード用XMP/XOMインタフェースソフトウェアを実装した[7]．本稿では，その概要について述べる．

2. XMP/XOM 及び CMIP ボード API の概要

2.1. XMP/XOM の概要

(1)XMP

XMP[5]は，m-GETreq 等 CMIS の各サービスや GetNextReq 等 SNMP の各サービスの両方に対応して，C 言語の関数に基づいた API を規定している．表 1 に XMP の関数一覧を示す．Get-req 等の CMIS/SNMP と対応する関数は，応答を受信するまで処理をブロックする同期型及びブロックしない非同期型の両方のモードを持つ関数が存在する．また，アプリケーションとインタフェース間のセッションを開始する Bind 等の CMIS/SNMP と対応しない関数は，インタフェースに関連する情報の設定や取得等を行う．

表 1 XMP の関数

CMIS/SNMP と対応する関数	Abort-req, Action-req*/rsp, Assoc-req*/rsp, Cancel-Get-req*/rsp, Create-req*/rsp, Delete-req*/rsp, Event-Report-req*/rsp, Get-Next-req*, Get-req*/rsp, Release-req*/rsp, Set-req*/rsp,
CMIS/SNMP と対応しない関数	Abandon, Bind, Error-Message, Get-Assoc-Info, Get-Last-Error, Initialize, Negotiate, Receive, Shutdown, Unbind, Validate-Object, Wait

注：*は同期型でも非同期型でも呼出し可能な関数であり，それ以外は同期型のみで呼出しを行う．

(2)XOM

XOM[6]は XMP や XDS(ディレクトリ)等種々の応用固有な API と組み合わせて使用され，それら API の関数のパラメタを操作する汎用的な API である．XOM で扱うデータの単位は OM オブジェクトと呼ばれ，表 2 に示すようにタイプ・シンタックス・値からなる OM 属性のリストから構成される．表 2 では CMIS-Event-Report-Argument という OM オブジェクトを示しており，表の各行が OM 属性に対応する．XMP の Event-Report-Req 関数(以下文中の XMP/XOM 関数には各々 mp_/om_という接頭語を付与する)発行時には，argument パラメタにこの OM オブジェクトのインスタンスが設定される．

OM オブジェクトには，そのデータ構造が XOM の仕様で規定されるパブリック OM オブジェクト(PUB)と，XOM の仕様で規定されず OM 属性に直接アクセスできないプライベート OM オブジェクト(PRI)が存在する．PUB のデータ構造は，タイプ・シンタックス・値をメンバとする構造体の配列である．PUB はさらに，インスタンスの管理主体がアプリケーションと XOM 内部のどちらかで各々 CPUB と SPUB に分けられる．XOM では，PRI や SPUB を操作するため表 3 に示すように om_create 等の関数を規定している．

表 2 CMIS-Event-Report-Argument OM オブジェクトの構造

タイプ	シンタックス	値
managed-Object-Class	Object(Object-Class)	ポインタ
managed-Object-Instance	Object(Object-Instance)	ポインタ
event-Time	String(Generalized-Time)	文字列
event-Type	Object(Event-type-Id)	ポインタ
event-Info	any	不定

注：シンタックスの Object(x)という記法は，別の OM オブジェクト x(サブオブジェクトと呼ぶ)を指すことを示す

表 3 XOM の関数

関数	機能
copy	PRI のコピー
copy_value	PRI 間での文字列のコピー
create	PRI の生成
decode	符号化された PRI の復号
delete	PRI 又は SPUB の削除
encode	PRI の符号化
get	PRI の一部/全部から SPUB を生成
instance	SPUB/PRI のクラスの確認
put	SPUB/PRI から PRI へ属性のコピー・挿入
read	PRI 中の文字列のセグメントを読み出し
remove	PRI の属性の削除
write	PRI へ文字列のセグメントを書き込み

例えばある管理操作を発行する場合，その関数のパラメタをアプリケーションで CPUB に設定して関数を起動し，関数の出力パラメタに PRI として格納された管理操作の応答を，アプリケーションが om_get 関数により SPUB に設定して値の表示等を行う．om_get 関数を除く XMP/XOM の全関数で，出力パラメタの OM オブジェクトの種類は PRI である．PRI や SPUB のインスタンスの生成・削除や扱う OM オブジェクトのクラス(以下 OM クラスと呼ぶ)を管理する記憶領域は，ワークスペースと呼ばれる．ワークスペースは mp_initialize で生成され，mp_shutdown で削除される．また異なるベンダが実装した XOM 上で動作するアプリケーションの相互運用性を保証するため，ワークスペースで管理する一部のデータ構造を XOM で規定している．

(3)XMP/XOM における ASN.1 符号化/復号

XMP/XOM では，GDMO で定義される管理情報の符号化/復号方法に関して，①XMP/XOM のアプリケーションが符号化/復号を行う，②XMP/XOM の内部で符号化/復号機能を提供する，の 2 種類の選択肢が存在する．これは，オプション機能を選択する mp_negotiate 関数を用いてアプリケーションが指定する．①の場合，アプリケーションと XMP/XOM は，ASN.1 基本符号化規則(BER)で符号化された管理情報を OM オブジェクトのシンタックス String(Encoding)を用いて授受する．②の場合，アプリケーションと XMP/XOM は管理情報の GDMO 定義に対応する OM オブジェクトを授受し，XMP/XOM の内部で OM オブジェクトから ASN.1 BER への符号化及びその逆の復号を行う．これは，om_encode と om_decode という関数により行われる．

GDMO 定義から対応する OM オブジェクトへの変換(以下 GDMO/XOM 変換と呼ぶ)規則は，X/Open で規定されている[8]．この規則では，GDMO 定義を入力として，表 4 に示す一連の情報(OM パッケージと呼ぶ)を出力する．アプリケーションが使用する OM パッケージは mp_negotiate 関数を用いて指定する．

表 4 OM パッケージ

種類	詳細
OM ﾊﾟｯｹｰｼﾞ定義	OM ﾊﾟｯｹｰｼﾞ名とｵﾌﾞｼﾞｪｸﾄ識別子(OID)，GDMO の OID，OM ｸﾗｽのﾘｽﾄ，OM 属性のﾘｽﾄ，OM 属性値のﾘｽﾄ
符号化定義	OM ｸﾗｽ/OM 属性に対応する ASN.1 型及びﾀｸﾞ
C 言語ﾍｯﾀﾞ	OM ﾊﾟｯｹｰｼﾞの OID，GDMO 中の OID，OM ｸﾗｽ定数

2.2. CMIP ボード API の概要

CMIP ボード[3]は，CMIP の ASN.1 定義に対応して C 言語の構造体をネストさせ構造化したプリミティブを API として提供する．M-EVENT-REPORTreq プリミティブのデータ型の一部を図 1 に示す．CMIP ボードはプロトコル制御情報の処理のみを行い，属性値や通知情報(EventInfo)等の GDMO で定義される管理情報(図 1 の any_t に対応)は，アプリケーションが符号化・復号を行う．

```
typedef struct { prim_header_t          PrimHead;
        u_long  InvokeId;
        u_long  Mode;   /* Confirmed or not*/
        obj_class_t     MOClass;
        obj_instance_t  MOInstance;
        cm_time_t       EventTime;
        cmip_ident_t    EventType;
        u_long          EventInfoLen;
        pointer_t(PDU_t) EventInfo;}  m_event_report_req_t;
typedef struct { u_long     FormFlag;
        union    { object_id_t          GlobalForm;
                   u_long  LocalForm;} Id;} cmip_ident_t;
typedef struct { u_long     FormFlag;
        union {dist_name_t              DN;
               char_str_t               NonSpecific;
               dist_name_t              LocalDN;}                Id;}
obj_instance_t;
typedef struct { u_long  Num;  pointer_t(rdn_t)  Rdn;} dist_name_t;
typedef struct { u_long  Num;  pointer_t(ava_t)  Ava;} rdn_t;
typedef struct {object_id_t  AttribType;  any_t  AttribVal;} ava_t;
typedef struct { u_short    ValFlag ;
        union {  char_32_t              Comp ;
                 cm_char_str_t          Buff ;
                } Val ;    } any_t;
```

図 1 M-EVENT-REPORT 要求プリミティブ

3. CMIP ボード用 XMP/XOM インタフェースソフトウェアの実装

3.1. 基本方針

①CMIP ボードのプリミティブ自身は変更せず，XMP/XOM と CMIP ボードのプリミティブとの対応付けを行う．
②表 1 及び表 3 に示した XMP/XOM の全関数を同期/非同期モードとも実現する．また，マネージャ/エージェントともに動作可能とし，SNMP(バージョン 1)もサポートする．
③XMP/XOM のオプション機能である自動符号化/復号機能と，アソシエーションの確立・解放を XMP/XOM の内部で行う ACM(Automatic Connection Management)機能をサポートする．
④GDMO/XOM 変換や ASN.1 の符号化/復号には既存のモジュールを利用する．
⑤実装環境は UNIX(SunOS 4.1.3) とし，CMIP ボードには既存の OSI 7 層ボード[4]を使用する．SNMP の PDU(プロトコルデータ単位)の符号化/復号を行う SNMP モジュールは独自のものを使用する．

3.2. ソフトウェア構成

CMIP ボード用 XMP/XOM インタフェースソフトウェアは，図 2 に示すように XMP/XOM アプリケーションと CMIP ボードのデバイスドライバあるいは SNMP モジュールの間に位置する．図 2 の網掛けの部分が今回実装した部分であり，CMIP ボード及び SNMP モジュールとは，各々 UNIX の file descriptor(fd)及びソケットを介して通信を行う．また，GDMO/XOM 変換モジュール[9]が出力した OM パッケージのファイルや GDMO 定義の解析及び対応する ASN.1 の符号化/復号を行う GDMO パーザ/インタプリタ[10]は，3.5 で述べる自動符号化/復号機能で使用する．

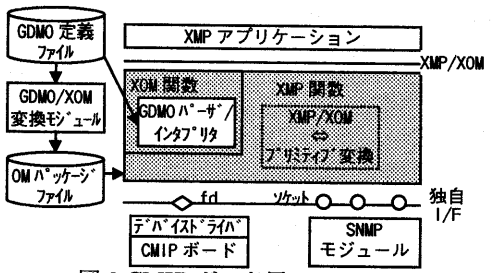

図 2 CMIP ボード用 XMP/XOM インタフェースソフトウェア構成

3.3. ワークスペースの実現

PRI の記憶領域やデータ構造等の OM オブジェクトの管理や OM パッケージの管理は，以下の 2 点を考慮して実現する(図 3)．
①PRI はワークスペースが削除されると共に削除されるが，SPUB はそれを生成したワークスペースの削除には影響を受けない．
②om_delete 関数を SPUB に適用しても含まれる PRI サブオブジェクトはアクセス不可能にならず，PRI へのハンドルは有効なままである．

3.3.1. OM オブジェクトの管理

(1)PRI/SPUB の記憶領域及びデータ構造

PRI の記憶領域は，処理の高速化を図るため SPUB と同様主記憶上に生成する．データ構造については，SPUB は XOM の規定に従い OM 属性の配列とし，一方，PRI は om_put や om_remove

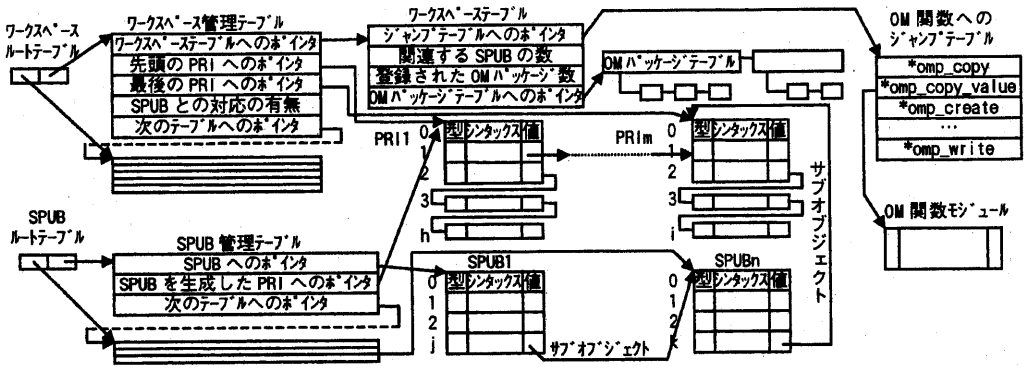

図 3 ワークスペースの実現方法

関数による OM 属性の挿入･削除を容易とするため，一つの PRI が保持する個々の OM 属性をポインタでリンクする．ただし，SPUB や PRI といった OM オブジェクトの種類や OM クラスの情報等が格納される 0 番目及び 1 番目の OM 属性は，2.1 で述べた相互運用性のために XOM が規定する OM 属性の配列とする．

(2)PRI/SPUB の管理

上記①及び②の処理を容易とするため，PRI と SPUB は独立に管理する．具体的には，PRI は個々のワークスペースを管理するテーブル(ワークスペース管理テーブル)から直接ポインタをリンクするが，SPUB はワークスペースとは独立なテーブル(SPUB ルートテーブル)に個々の SPUB を管理するテーブル(SPUB 管理テーブル)のポインタをリンクし，そこから SPUB のインスタンスを指す．ワークスペースを削除する mp_shutdown 起動時には，対象となるワークスペースにリンクされた PRI に対してのみ om_delete 関数を発行して削除する．また，SPUB に対する om_delete 時には，SPUB のサブオブジェクトのポインタのみをたどって削除し，PRI サブオブジェクトは削除しない．

図 3 で，ワークスペースルートテーブル･SPUB ルートテーブル及び 2.1 で述べた相互運用性のために XOM が規定する OM 関数へのジャンプテーブルは静的なテーブルであり，その他のテーブルはワークスペースや OM オブジェクトの生成・削除に伴い動的に生成・削除する．OM パッケージテーブルの管理は次節で述べる．

3.3.2. OM パッケージの管理

OM オブジェクトのインスタンスの生成や自動符号化/復号を行うためには，ワークスペースにおいて関連する OM パッケージを管理しなければならない．ここでは，OM パッケージの追加や変更に柔軟に対応するため，GDMO 定義ファイルから GDMO/XOM 変換規則に従って変換された OM パッケージのファイルを解析し，対応関係を抽出するようにした．以下では，OM パッケージファイルの解析で重要となる OM クラス定義の作成処理を示す．括弧内は値の例を示す．

①OM クラス情報の抽出

C 言語ヘッダファイルから，該当する OM クラス名(ISO_AE_TITLE_LIST)に対応する ASN.1 型(AE-titleList)を抽出する．

②OM 属性情報の抽出

OM パッケージ定義ファイルから，上記(1)の OM クラスにおける OM 属性名(ISO_AE_TITLE)と OM 属性のシンタックス(Integer)を抽出する．シンタックスが他の OM オブジェクトを含む(Object (ISO_AE_TITLE))，すなわち OM サブオブジェクトの場合，その ASN.1 型名(AE-title)も抽出する．

③ワークスペースへの設定

図 3 に示したワークスペーステーブルに，mp_negotiate 関数により登録された OM パッケージの情報を OM パッケージテーブルのリストとしてリンクする．各 OM パッケージテーブルには，上記①②で抽出した OM オブジェクトの情報を，OM クラスの個数分リンクして保持させる．その情報には，OM クラス名，OM 属性名，OM クラスに対応する ASN.1 型名，OM サブオブジェクトが存在する場合にはその名前と対応する ASN.1 型名が含まれる．

3.4. XMP/XOM とプリミティブの対応付け

XMP/XOM と CMIP ボード及び SNMP のプリミティブは，以下のように対応付ける．

(1)管理操作/通知の対応付け

CMIP/SNMP の管理操作/通知の要求(req)や応答(rsp)は，表 1 の CMIS/SNMP に対応する関数で示したように，XMP の関数と一対一の対応付けが可能である．管理操作/通知の指示(ind)は mp_receive に対応付ける．また，管理操作/通知の確認(cnf)は，同期/非同期モードにより以下のように対応付けが異なる．

●同期モードの場合

要求(req)の関数が呼ばれたら，インボーク ID を生成して CMIP ボードのプリミティブを発行し，確認(cnf)プリミティブが到着するまで fd を監視する．ここで，受信するプリミティブは連結応答であったり他のプリミティブの可能性があるため，プリミティブを一旦本ソフトウェア内のキューに格納することとした．要求と同一のインボーク ID を持つプリミティブを受信したら，要求を発行した関数の出力パラメタに全ての連結応答を OM オブジェクトとして含め，関数からリターンする．

●非同期モードの場合

要求(req)の関数が呼ばれたら，プリミティブを発行し即座にリターンする．この時，本モジュールで生成したインボーク ID をアプリケーションに通知する．その後 mp_receive 関数が呼ばれた時点で，キューまたは fd にプリミティブが到着しているかを調べる．連結応答や確認(cnf)プリミティブが到着していたら，その種類に対応する OM オブジェクトを生成し，個々のプリミティブ毎に mp_receive 関数からリターンする．データが到着していない場合，その旨をフラグ(Completion-Flag)にセットしてリターンする．

(2)アソシエーションの対応付け

ACM 機能が disable，すなわちアソシエーションの確立・解放をアプリケーションが行う場合は，mp_assoc_req / rsp，mp_release_req / rsp，mp_abort_req，mp_receive 関数と ACSE(アソシエーション制御サービス要素)のプリミティブを対応付ける．ACM 機能が enable の場合，アソシエーションの確立・解放のタイミングは XMP では規定されない．そこでセッションを開始/終了する mp_bind/mp_unbind 関数の起動時に各々アソシエーションの確立/解放を行うようにした．

(3)パラメタの対応付け

CMIP の ASN.1 定義が構造形の場合には CMIP ボードのプリミティブを表わす C 言語の構造体と OM オブジェクトを対応付け，単純形の場合には構造体のメンバと OM 属性値を対応付ける．例えば，CMIP の managed-Object-Class パラメタの ASN.1 定義は ObjectClass という構造形であり，表 2 と図 1 に示したように，OM オブジェクト Object-Class と CMIP ボードのプリミティブの構造体 cmip_ident_t を対応付ける．さらに，これら OM オブジェクトと構造体を単純形になるまで展開し，String(ObjectIdentifier) というシンタックスを持つ OM 属性値と，オブジェクト識別子の型を持つ GlobalForm という構造体のメンバを対応付ける．

CMIP の ASN.1 定義が ANY の場合，3.5 で述べる自動符号化/復号機能を使用しない時には，アプリケーションはその OM 属性のシンタックスを String (Encoding)とし，属性値に直接符号化列を設定するため，any_t 構造体のメンバ(Comp/Buff)とその OM 属性値を対応付ける．自動符号化/復号機能を使用する時は，om_encode/decode 関数のパラメタである Encoding という PRI の OM 属性 ObjectEncoding に符号化列を格納するため，それを any_t 構造体のメンバと対応付ける．

SNMP の PDU の各パラメタは ASN.1 の単純形で定義されるため，上記の CMIP と同様 OM 属性と対応付ける．

3.5. ASN.1 自動符号化/復号機能の実現

自動符号化/復号機能は，図 4 に示すように，CMIP ボードからのプリミティブの受信時や Set_req 等の XMP 関数の起動時に om_encode/decode 関数を XMP/XOM の内部で起動することで実現し，OM オブジェクトと ASN.1 BER 符号化列間で管理情報の符号化/復号を行う．プログラムを変更せずに任意の GDMO 定義に対応可能とするため，om_encode/ decode 関数の処理には，3.4.2 で述べた OM パッケージ管理のモジュール及び GDMO インタプリタを用いる．

(1)事前設定

om_encode/om_decode 関数は，図 4 に示したように，既存の GDMO パーザが符号化/復号の対象となる GDMO 定義ファイルを解析して生成した C 言語の構造体(ASN.1/C 構造体と呼ぶ)を保持する．これは，後述する GDMO インタプリタが使用する．また，OM パッケージ管理モジュールは，符号化/復号対象の GDMO 定義に対応する OM クラスの情報をワークスペースに設定する．

(2)関数起動時の処理

●om_encode 関数起動時

①OM パッケージ管理モジュールが保持する OM クラスと ASN.1 型名の対応表に基づいて，アプリケーションから渡された OM オブジェクトの

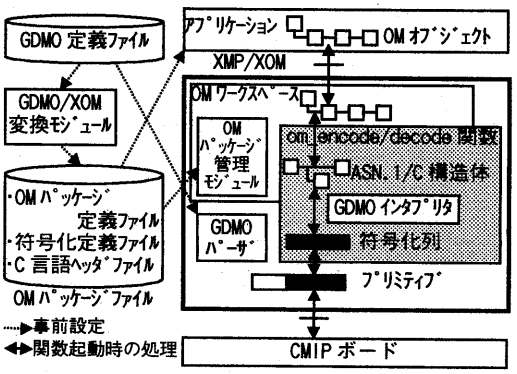

図 4 自動符号化/復号処理の流れ

属性値を対応する ASN.1/C 構造体のメンバに設定する.
②属性値が設定された ASN.1/C 構造体を入力として GDMO インタプリタを呼び出し, ASN.1 BER の符号化列を得る.
③Encoding という PRI を生成し, String (Encoding) シンタックスを持つ OM 属性 ObjectEncoding の値に符号化列を格納する.
●om_decode 関数起動時
om_encode 関数とは逆の流れになり, GDMO インタプリタにより復号された ASN.1/C 構造体から対応する OM オブジェクトを生成する.

4. 評価

実装した CMIP ボード用 XMP/XOM インタフェースソフトウェアを 2 台のワークステーション間で LAN を介しマネージャとエージェントとして対向させ, 各関数を起動する試験用のツールを用いて CMIP 及び SNMP により通信できることを確認した. また, 公開されたサンプルプログラム[11]が本ソフトウェア上で正常に動作することを確認した. このプログラムは, XMP/XOM 上で動作して, m-SET や m-EVENT-REPORT の授受を同期/非同期モードで行う.

4.1. プログラムサイズについて

開発したソースプログラムは, XMP 関数が約 1250K バイト, XOM 関数が約 500K バイト, また流用した GDMO パーザ及びインタプリタ部分が約 760K バイトであった. om_encode/om_decode 関数の実現に既存の GDMO パーザ/インタプリタを流用したことにより, XOM 関数を効率的に開発できた.

4.2. 性能について

本ソフトウェアの性能を評価するため, XMP/XOM と CMIP ボードのプリミティブ間での対応付けに要する時間を測定した. 測定は SPARC 670MP(単一 CPU)で行った. 結果を表 5 に示す. 表 5 で, 要求(req)の値は mp_get_req や mp_set_req 関数が起動されてから CMIP ボードにプリミティブを発行するまでの時間であり, 指示(ind)の値は CMIP ボードよりプリミティブを受信してから mp_receive 関数がリターンするまでの時間である. ①は属性 ID を指定しない m-GETreq/ind であり, ②と③は 2 属性 (administrativeState 属性と systemTitle 属性) を含む m-SETreq/ind であるが, ②は自動符号化/復号機能を使用し③は使用していない. また, ①②③ともに CMIS の基底オブジェクトのパラメタは system(相対識別名 1 個から識別名を構成)とし, その他のパラメタ値はデフォルトとした.

表 5 ソフトウェアの処理時間(単位:ms)

操作	要求(req)	指示(ind)
①m-GET(属性 ID 指定無)	6.5	2.3
②m-SET(2 属性, 自動符号化 On)	11.2	12.2
③m-SET(2 属性, 自動符号化 Off)	9.1	6.1

表 5 から, 管理情報を含まない場合(①)や自動符号化/復号を行わない場合(③)には 10ms 以下で対応付け処理が行われ, 要求より指示に要する時間の方が短い. また, 自動符号化/復号を行う場合(②)には符号化より復号に時間を要している.

以上の結果より, 本ソフトウェアは実用的な処理時間を達成できたと考えられる.

5. おわりに

本稿では, 既存の CMIP ボードに XMP/XOM インタフェースを持たせるための CMIP ボード用 XMP/XOM インタフェースソフトウェアの実装を述べた. ワークスペースの実装では, OM オブジェクトを XOM 関数の処理に適した形で管理し, また既存のモジュールが出力する OM パッケージのファイルから情報を抽出して OM パッケージを管理した. XMP/XOM 関数とプリミティブの対応付けでは, OM オブジェクトと CMIP ボードのプリミティブのデータ構造が類似しているため, 容易に対応付けられた. ASN.1 自動符号化/復号機能は, 既存の GDMO パーザ/インタプリタを利用することで効率的に実現した. 評価の結果実装したソフトウェアが, 実用的な性能を有することを確認した.

最後に日頃ご指導頂く KDD 研究所村上仁己所長, 鈴木健二副所長に感謝します.

参考文献

[1]:ITU-T Rec.X.711,"Common Management Information Protocol Specification for CCITT Applications",1991
[2]:ITU-T Rec. X.722, "Guidelines for the Definition of Managed Objects", 1992
[3]:加藤他「パソコン用CMIPボードの開発」,1992信学秋大
[4]:井戸上他「パーソナルコンピュータおよびワークステーションのためのOSI 7 層ボードの実装と評価」情処論文誌 Vol.36 No.3
[5]:X/Open CAE Spec. C306,"Systems Management: Management Protocols API(XMP)",Mar.1994
[6]:X/Open CAE Spec. C315,"OSI-Abstract-Data Manipulation API(XOM), Issue 2",Feb.1994
[7]:杉山他「CMIPボード用XMP/XOMインタフェースの設計」,第51回情処全大, 1994
[8]:X/Open CAE Spec. C502,"Systems Management: GDMO to XOM Translation Algorithm",Oct.1995
[9]:HP OpenView Distributed Management Developer's Guide, "ovpackgen", 1994
[10]:K.Sugiyama et.al. "Implementation and Evaluation of MIB Tester for OSI Management", IM'97 1997
[11]:OSF DME NMO1.0 "Application Development Guide and Reference", 1994