2U-5

OSI FTAM実装における抽象構文に関する一考察

加藤聰彦　鈴木健二
国際電信電話株式会社 研究所

1. はじめに

CCITTやISOでは異機種計算機・端末相互間の通信を実現するため、開放型システム間相互接続(OSI)の標準化を進めている。筆者等はこれまでにセション層以下のプロトコルをVAX11/780上に実装した。本稿では上位2層の実装に関して、OSI FTAM (File Transfer Access and Management)[1]を実装するための応用層(AL)とプレゼンテーション層(PL)におけるデータの表現形式についての検討結果を報告する。

2. 抽象構文と転送構文

一般に各計算機内のプロセスでは、データを表現するために独自のデータ表現形式(以下では構文と呼ぶ)を使用している。OSI環境下でプロセス間通信を行う場合には、双方でデータの意味を共通に解釈するため、転送されるデータの構文を決める必要がある。このためにアプリケーション層(AL)とプレゼンテーション層(PL)[2]は密接な関連をもつ。

ALにおいてデータの意味を表現・解釈するために用いられる構文を抽象構文と呼ぶ。この抽象構文は各プロセス独自の構文をモデル化したもので、図1に示すようにローカルな環境で表現されたデータは、OSI環境ではALの抽象構文として表現される。例えばFTAMの場合は、FTAMで使用されるプロトコルデータ単位(PDU)のフォーマットを規定した抽象構文と、転送するファイルの情報を表す抽象構文が使用される。

またPLは、ALで扱われる抽象構文に基づいて、プロセス間で実際に転送されるデータを具体的に表現する構文(転送構文)を管理・制御する。

PLはALから使用する抽象構文を通知され、それに応じて転送構文を双方の合意のもとで決定する。またALとPLの間でやりとりされるデータは抽象構文を用いて表され、PLではそのデータを転送構文に変換する。この変換規則をエンコーディング規則と呼ぶ。抽象構文を記述する方法としては、ASN.1 (Abstract Syntax Notation One)[3]、X.409がそれぞれISO、CCITTで検討されている。これらはほぼ同様な記述能力とエンコーディング規則を持つ。

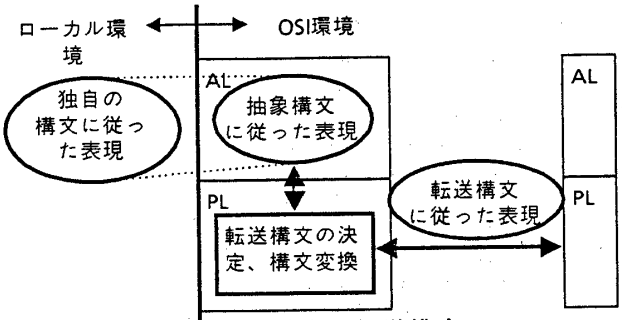

図1 抽象構文と転送構文

3. FTAMにおけるローカル構文の実現方法の検討

3.1 ローカル構文の実現方針

実際のALプログラムでは、抽象構文と等価な独自の構文(以下ローカル構文と呼ぶ)を使用しており、PLプログラムでは、PDUの送信時にはローカル構文から転送構文に従ったフォーマッティング、受信時にはPDUのフォーマットチェックとローカル構文への変換を行う。そこでOSIの上位2層のプログラムを作成するためには、どのようなローカル構文を用い、PLにおいてどのような構文変換を行うかが問題となる。

ローカル構文は、PLプログラムにおける構文変換とALプログラムの処理が容易なように定める必要がある。ここでは、次の2つの方針によりFTAMのローカル構文の実現方法を検討した。

① 抽象構文に従って構造化されたデータ構造をローカル構文として持つ。

② ALプログラムの処理を簡単化するために、可能な限り、ALプログラムが使用する計算機内部の制御用のデータ構造(OSのファイル管理機能で使用する制御ブロック等)をローカル構文に対応させる。

3.2 FTAMのローカル構文の実現法

以下でファイルの属性を調べるためのFTAMのPDUであるF-READ-ATTRIB-requestとF-READ-ATTRIB-responseを例にとり、対応するローカル構文を検討する。F-READ-ATTRIB-requestによ

A Study on the Abstract Syntax in the Implementation of OSI FTAM
Toshihiko KATO, Kenji SUZUKI
KDD R & D Laboratories

り、ファイルを管理する側にその作成日時や作成者等の調べたいファイル属性を伝え、その属性の値がF-READ-ATTRIB-responseで返される。

(1)抽象構文

上記のPDUの抽象構文は次のようである。

```
F-READ-ATTRIB-request ::= SEQUENCE {
  attribute-name [0] IMPLICIT BITSTRING
    { ··· read-date-and-time-of-creation (3), ···
      read-identity-of-creator (7), ··· } }
F-READ-ATTRIB-response ::= SEQUENCE { ···
  attributes Attributes OPTIONAL, ··· }
Attributes ::= [APPLICATIN 11] IMPLICIT
      SEQUENCE OF CHOICE { ···
  date-and-time-of-creation [3] Date-and-Time-Attribute, ···
  identity-of-creator [7] User-Identity-Attribute, ··· }
Date-and-Time-Attribute ::= CHOICE {
  no-value-available [0] IMPLICIT NULL,
  actual-values [1] IMPLICIT GeneralisedTime }
User-Identity-Attribute ::= CHOICE {
  no-value-available [0] IMPLICIT NULL,
  actual-values User-Identity }
User-Identity ::= [APPLICATIN 4] IMPLICIT
  GraphicString
```

(2)方針①に基づくローカル構文の例

(1)のPDUに対するローカル構文のデータ構造の例を以下に示す。この例ではASN.1のSEQUENCEに対してレコード型を対応させ、OPTIONALやCHOICEに対してはその有無を示すフラグを用いている。またSEQUENCE OFは、ポインタによるリスト構造で表現している。ここではAdaによる実現例を示している。

```
type FReadAttirbReqType is record
  AttribName : BitArrayAttribNameType;  end record;
type BitArrayAttribNameType is
  array ( AttribNameType ) of Boolean;
type AttribNameType is ( ··· TimeOfCreation, ···
                  IdOfCreator, ··· );
type FReadAttirbRespType is record ···
  AttribFlag : Boolean;
  Attrib : AttribType; ···  end record;
type AttribType is record
  AttribId : AttribNameType; ···
  TimeOfCreation : TimeAttribType; ···
  IdOfCreator : UserIdType; ···
  NextAttrib : AccessAttribType  end record;
type AccessAttribType is access AttribType;
type TimeAttribType is record
  Flag : range 1 .. 2;
  ActualValue : Time;  end record;
type UserIdType (length : integer) is record
  Flag : range 1 .. 2;
  ActualValue : String(1 .. length);  end record;
```

このようなデータ構造を使用することにより、ALプログラムでのPDUの処理が容易になる。

(3)方針②を考慮したローカル構文の例

次にFTAMをVAX/VMS上に実装する場合に、VMSの制御ブロックを考慮に入れたローカル構文の検討結果について示す。VMSのファイル管理機能は、RMS (Record Management Service)としてまとめられている。ファイルの生成、レコードの読み書き、ファイルの属性の参照/設定等はRMSのシステムサービスを発行することにより実現される。そこでFTAMのPDUに対応するローカル構文として、RMSのシステムサービスが使用する制御ブロックを用いることができれば、データ構造の変更が少なくなると考えられる。

RMSではまたファイルの属性の参照に対してはDISPLAYというシステムサービスが用意されておりこれはFAB (File access block) とXAB (Extended attribute block)という制御ブロックを使用する。

```
DisplayFab : FAB_TYPE;
XabDat : XAB_TYPE ( XAB_C_DAT );
DisplayFab.XAB := XabDat'ADDRESS;
DISPLAY ( FAB => DisplayFab );
```

とすると、XabDat.CDTというフィールドにバイナリで表現されたファイルの作成日時が代入され、

```
DisplayFab : FAB_TYPE;
XabPro : XAB_TYPE ( XAB_C_PRO );
DisplayFab.XAB := XabDat'ADDRESS;
DISPLAY ( FAB => DisplayFab );
```

とすると、XabPro.UICにファイルを作成したユーザのUICコード(VMSでユーザを識別するID)が代入される。そこでF-READ-ATTRIB-responseに対するローカル構文として、次のようにこの制御ブロックを使用することができると考えられる。

```
type AttribType is record
  AttribId : AttribNameType; ···
  TimeOfCreation : XAB_TYPE ( XAB_C_DAT ); ···
  IdOfCreator : XAB_TYPE ( XAB_C_PRO ); ···
  NextAttrib : AccessAttribType  end record;
```

data-and-time-of-creationに対応する情報はバイナリ時間として保持される。そこでPLプログラムではこれをASCII時間に変換して、F-READ-ATTRIB-responseをフォーマッティングすればよい。一方identity-of-creatorについてはUICコードとして表現されておりこれをPLプログラムでユーザ名に変換する必要がある。

4. おわりに

本稿ではFTAMの実装におけるローカル構文の実現方法に関する検討結果を述べた。本検討はFTAMを強く意識したものであり、個々の実装におけるローカル構文は、PLプログラムの汎用性、プログラム構成、プログラム間でのデータの受け渡し方法等に依存して定められると考えられる。最後に日頃御指導頂くKDD研究所野坂所長、小野次長、浦野情報処理研究室長に感謝します。

参考文献 [1]: ISO, DIS 8571/1-4, Feb. 1986.
[2]: ISO, DIS 8822, DIS 8823, May 1985.
[3]: ISO, DIS 8824, DIS 8825, Mar. 1985.