

OSIアプリケーションプロトコルのための 2E-4 スタブジェネレータへの抽象サービス定義の適用

長谷川 亨 西山 智 野村 真吾

国際電信電話株式会社 上福岡研究所

1. はじめに

計算機の分散環境の進展にともない、遠隔地の計算機のプログラムをクライアント/サーバモデルに基づいて通常の手続き呼び出しの形式で実現する遠隔手続き呼び出し(RPC)が採用されている^[1]。クライアント/サーバはRPCの呼び出し/実行を実現する手続きを用いて作成されるが、RPCの引数、戻り値のインタフェースの規定からスタブジェネレータによりこの手続きを自動的に生成する試みが一般的に行われている。OSIのアプリケーション層においてもRPCを実現するROSEプロトコル^[2]上にMHS等のプロトコルが標準化されており、同様なジェネレータが必要である。そこで、筆者らはそのインタフェースの規定に抽象サービス定義^[3]を用いることを提案し、本稿ではその方法および問題点について考察する。

2. 抽象サービス定義からのスタブ生成

2.1 スタブジェネレータの必要性

RPCはトランスペアレントな転送プロトコル上に定義され、①クライアントとサーバを結合(バインド)、②遠隔手続きを引数とともにサーバに対して転送しサーバが実行した結果を受信して、遠隔手続きの戻り値として返す(オペレーション)、③クライアントとサーバの切り放し(アンバインド)の操作手順で遠隔手続きを実行する。

MHS等のROSE上に実現される通信システムは、オペレーションを呼び出し/実行を行うクライアント/サーバのプログラムとして実現される。各操作を規定するインタフェースからスタブ(下位のプロトコルを用いて操作の転送、結果の受信を行い、戻り値として受信した結果を返す手続き)を自動生成することにより、RPCの手順を実現するプログラムを作成する手間が省け、開発効率が向上する。

2.2 抽象サービス定義の適用

抽象サービス定義は①クライアント/サーバ、②各操作、③各操作の下位プロトコルでの転送のしかたの規定から構成される。これらはASN.1^[4]のマクロ機能を用いて、それぞれ①オブジェクト、②オペレーション(バインド、アンバインドを含む)、③アプリケーションコンテキストマクロとして定義される。また、オペレーションの引数、戻り値のデータ型はASN.1を用いて定義される。

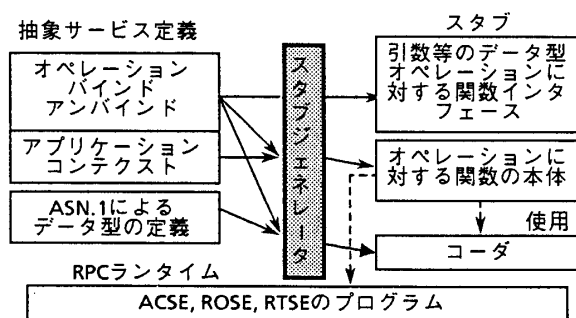


図1 スタブジェネレータの機能

そこで、オブジェクト毎にスタブを生成することを考えた場合、以下のように抽象サービス定義を用いてスタブを生成できる(図1)。

(1) 関数呼び出しインタフェース

各操作を実現する関数をユーザが呼び出せるように、バインド、オペレーション等の規定から各関数の呼び出し形式をヘッダファイルとして生成する。また、ASN.1により定義された引数等のデータ型をプログラミング言語のデータ型に変換し、転送されるバイト列との間の変換をASN.1の符号化規則に従って符号/復号化するコードを生成する。

(2) 操作を実現する関数

各操作を実現する関数を、アプリケーションコンテキストに従って、オペレーションを転送、受信するプログラムを生成する。この関数は、下位プロトコルであるACSE, ROSE, RTSEを1つのプログラムとして準備したプログラム(RPCランタイム)を用いて、転送を行う。

以下ではオペレーションの一例を用いて抽象サービス定義からのプログラム生成について述べる。

3. オペレーションからのスタブ生成

3.1 関数呼び出しインタフェース

図2の記述ではクライアント用のオペレーションoperation-1とバインドAuthenticateUserが定義されており、この記述より図3の関数operation-1-clientのインタフェースを生成する。

オペレーションはこの関数呼び出しに対応し、図2のARGUMENTに対応する構造体を引数としてRESULTの整数を返す関数になる。また、ERRORで指定される呼び出し失敗時の情報を引数として返す。バインドも同様な関数を生成するが、バイン

```

/*オペレーション*/
operation-1 ::= ABSTRACT-OPERATION
  ARGUMENT type-a /*引数*/
  RESULT type-b /*結果*/
  ERRORS { error-1 } /*エラー情報*/
  ::= 1 /*識別子*/
type-a ::= SEQUENCE { age INTEGER, name IA5STRING }
type-b ::= INTEGER
error-1 ABSTRACT-ERROR PARAMETER ::= 1
/*バインドオペレーション*/
AuthenticateUser ::= ABSTRACT-BIND
  ARGUMENT credential Credential
  BIN-ERROR ENUMERATED { password-invalid(0) }
  ::= 1
Credential ::= IA5STRING /*文字列のデータ型*/

```

図2 インタフェースの規定

ドしたアソシエーション(下位コネクション)の識別子を関数の戻り値として返し、以後のオペレーションではこの値を用いて相手サーバを指定する。

```

(1) type-a,bのデータ型
struct type-a { int age; /*ageのデータ型*/
  char *name; }; /*nameのデータ型*/
typedef type-b int;
(2) オペレーション失敗時の情報
struct result-1 { int result; /*成功:0, 失敗:1*/
  int error; /*エラー情報*/ };
(3) クライアント用関数インタフェース
type-b operation-1-client (id, arg, res, association)
  int id; type-a *arg; result-1 *res, int association;
  /*association: アソシエーションの識別子*/

```

図3 関数インタフェース

3.2 操作を実現する関数

図3の関数はRPCランタイムのプログラムを使用して、手続きおよび戻り値を送受するが、このRPCランタイムの使用法はアプリケーションコンテキストにより規定できる(図4)。

```

context-1 APPLICATION-CONTEXT
  APPLICATION SERVICE ELEMENT { acse }
  /*バインドが使用する下位プロトコル: ACSE*/
  BIND AuthenticateUser
  REMOTE OPERATIONS { rose }
  /*オペレーションが使用する下位プロトコル: ROSE*/
  ...

```

図4 アプリケーションコンテキスト

図4の記述より、バインド、アンバインドはACSEに、オペレーションはROSEにマッピングされることがわかり、図3の関数は図5のように実現される。この関数は引数をROSEプロトコルのプリミティブRO-VOKE-Requestにより転送し、戻り値をRO-RESULT-Indicationにより受信し、その結果を戻り値として返す。またエラー情報は、RO-ERROR-Indicationにより受信する。引数、戻り値等は下位のプロトコルのプロトコル要素(PDU)のユーザデータとして転送されるため、エンコーダ/デコーダによりバイト列に変換して下位プロトコルに渡す。ただし、アプリケーションコンテキストではオペレーションをマッピングするプリミティブ、およびRO-VOKEプリミティブのオペレーションクラス等のパラメータの値は規定されないため抽象サービス定義に加えてこれらの選択

を規定する方法が必要である。

```

type-b operation-1-client (id, arg, res, association)
  type-a *arg; result-1 *res, int association;
  {
    OBJECTIDENTIFIER op-value = {整数の配列} /*オペレーションの識別子*/
    OCTETSTRING btr; /*バイト列のデータ型*/
    int id; /*RO-VOKEの呼び出し番号*/
    btr = encode(arg); /*引数をバイト列に変換*/
    RO-VOKEの呼び出し番号idの選択;
    RO-VOKEの送信ランタイム関数の呼び出し;
    /*引数: btr, id, association, オペレーションクラス, オペレーションの識別子*/
    RO-VOKEの呼び出し番号idに対応するプリミティブの読み出しランタイム関数の呼び出し;
    if (受信プリミティブ == RO-RESULTIndication)
    {
      res->result = 0; /*成功*/
      ユーザデータのデコード結果をリターン
    }
    if (受信プリミティブ == RO-ERRORIndication)
    {
      res->result = 1; /*失敗*/
      res->error = ユーザデータのデコード結果;
      リターン
    }
  }

```

図5 関数operation-1-client

4. 考察

抽象サービス定義からスタブを生成するためには、さらに以下の点を検討する必要がある。

(1) オペレーションの非同期な呼び出し

オペレーションの呼び出しにはRO-VOKEプリミティブの非同期クラスに対応して、オペレーションを呼び出した時に応答を待たず後で結果を受信するものがある。この実現には、関数の呼び出し後、結果を受信するための関数を呼び出す方法が考えられる。この受信関数を呼び出す場合オペレーションを識別する必要がある、オペレーションに対応するROSEプロトコルのRO-VOKEプリミティブの呼び出し番号を呼び出し関数の戻り値にする方法が考えられる。

(2) 下位プロトコルの異常時の扱い

下位プロトコルとしてACSE, ROSEが使用される場合、例えば、オペレーションの途中でアソシエーションが切断した時の異常動作は抽象サービス定義により規定されない。ACSE, ROSE間での情報のやりとりも含めて、どのようにユーザにこれらの情報を知らせるかを検討する必要がある。

4. おわりに

本稿では抽象サービス定義からスタブを生成するジェネレータの処理方式について検討した。今後、抽象サービス定義では不十分な点も含めて、オペレーションから手続きを生成する方法について検討を進める予定である。最後に日頃御指導頂くKDD上福岡研究所小野所長、湯口次長、小西通信ソフトウェア研究室長、柳平コンピュータ通信研究室室長、若原通信ソフトウェア研究室主任研究員、鈴木コンピュータ通信研究室主任研究員に感謝する。

参考文献

- [1]: A.D.Birrel, B.J.Nelson, "Implementing Remote Procedure Calls", ACM Trans. Computer Systems, Vol. 2 No. 1, Feb. 1984.
- [2]: CCITT, Draft Rec. X.219/229, Dec. 1987.
- [3]: CCITT, Draft Rec. X.407, June 1987.
- [4]: ISO, "ASN.1", ISO / IS 8824 / 8825