OSIトランスポート･プロトコルのインプリメントと製品検証

鈴木健二　　加藤聰彦
国際電信電話株式会社研究所

1. はじめに

ISO、CCITTでは、異機種計算機間通信を実現するために、開放型システム間相互接続(OSI：Open Systems Interconnection)の標準化を進めており、トランスポート･レーヤについても、そのサービス定義とプロトコル仕様が確定している[1]。また個々のOSI製品が標準プロトコルを正しく実装しているかを検査する製品検証も重要な課題となっている。

筆者等は、公衆パケット網においてMHS(Message Handling System)等の高度通信処理サービスを提供する網内ホストの構築過程で、OSIトランスポート･プロトコル(TP)のクラス0から3をインプリメントし、またオートマトン･モデルに基づく製品検証方式[2]をTPに適用した。そこで本稿では、TPのインプリメント上の問題点及びTPの製品検証方式について報告する。

2. TPのインプリメント

TPはOSI参照モデルのトランスポート･レーヤ(TL)のプロトコルであり、セッション･レーヤ(SL)の要求するサービス品質にみあうデータ転送機能を提供するものである。

TPでは、想定するネットワーク･コネクション(NC)の品質とトランスポート･コネクション(TC)をNCに多重化する機能に着目して0から4までの5つのクラスが決められている。0から3までのクラスが提供するプロトコル機構を表1に示す。

表1 TPのプロトコル機構

プロトコル機構	種別	0	1	2	3
NCへの割付け		*	*	*	*
TPDU転送		*	*	*	*
セグメンティング/アセンブリング		*	*	*	*
コンカティネーション/セパレーション			*	*	*
コネクション確立		*	*	*	*
コネクション拒否		*	*	*	*
通常解放	暗示	*			
	明示		*	*	*
エラー解放		*		*	
TPDUのTCへの関連付け		*	*	*	*
DT TPDU番号付け	通常		*	m	m
	拡張			o	o
優先データ転送	NLの通常データ		m	*	*
	NLの優先データ		ao		
障害後の再割付け			*		*
TPDUの保留	NLの送達確認		ao		
	確認応答TPDU		m		*
再同期			*		*
マルチプレクシング/デマルチプレクシング				*	*
フロー制御	有り			m	*
	無し	*	*	o	
リファレンス凍結			*		*
プロトコル･エラーの処理		*	*	*	*

*:その手順を含む、 m:ネゴシエート可(実装は必須)、 o:ネゴシエート可(実装は任意)、 ao:ネゴシエート可(実装は任意、NLに依存)

2.1 インプリメントの方針

TPのクラス0から3をインプリメントするに当たり、次の様な基本的設計方針をたてる。

① 同一ないしは異なったクラスの複数のTCを同時に張ることができ、システム内のリソースを管理する機能を実装する。

② TPの勧告に従い、プロトコル機構を明確に意識したプログラム構成とする。

③ コンフォーマンスに従ったクラス･ネゴシエーション機能を実現する。

④ SLとTL間およびTLとネットワーク･レーヤ(NL)間のインターフェスは、各々サービス定義勧告に定められたプリミティブのみをPDU(Protocol Data Unit)に近い形式でフォーマット化して用いた。(図1参照)

本システムでは、TLとSL間のTSAP(Transport Service Access Point)が複数存在することを許し、またNLとTL間のNSAP(Network Service Access Point)は、OSI勧告においてNSAPアドレスとDTEアドレスの関係が未だ明確になっていないために、DTEアドレスに固定的に割当てた。またTSAP、NSAPごとに複数のTCEP(Transport Connection Endpoint)、NCEP(Network Connection Endpoint)が存在することが可能な構成とした。

CEPのIdentifier(ID)はその一部にSAPアドレスを含んでいるが[3]、システム内の隣接レーヤのコネクションを対応付けるためにはより少ない情報で充分である。そこで、本システムではシステム内の隣接レーヤ間で一意にコネクションを識別するためTransport Service Connection Identifier (TS CID)、Network Service Connection Identifier (NS CID)という識別子を用いる。TS CID(NS CID)はシステムがそのTC(NC)のイニシエータである場合は、SL(TL)が割当て、T(N)CONreqによりTL(NL)に通知する。またレスポンダである場合は、TL(NL)が割当て、T(N)CONindによりSL(TL)に通知する。TS CID(NS CID)の重複を避けるために、本システムではSL(TL)が割当てる場合は奇数、TL(NL)が割当てる場合は偶数を使用する。

またTP勧告ではクラスの異なるTCを同一のNCに多重化できるが、本システムでは現実性をふまえ、同一クラスのTCのみを一本のNCに多重化する。

さらに隣接レーヤ間のインタフェースは、プリミティブ用処理待ちキューを用いて実現しており、キューへのアクセスは、各レーヤの処理が独立に行えるよう、プリミティブを連結した後すぐに制御を戻す非同期形のアクセス法を用いている。またレーヤ間のフロー制御は、処理待ちキューを用いて行われ、フロー制御のための特別なプリミティブは用意しない。

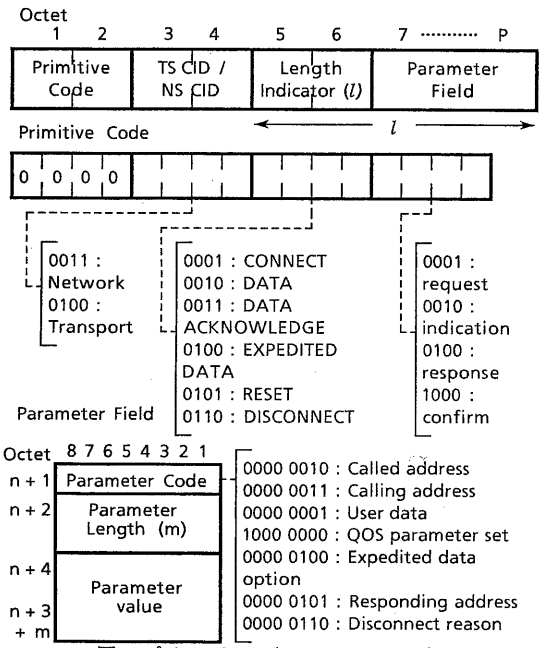


図1 プリミティブのフォーマット

以上の方針のもとでTPをVAX11/780のOSであるVMS下でインプリメントする。VAX/VMSでは、プロセス間の通信はメールボックスにより行われる。またNLのプロトコルは、VAX/VMS下で動作するPSI(Packetnet System Interface)のX.25プロトコルを用いる。このPSIの機能の制約により、NLの送達確認及びNLの優先データは使用しない。

また、使用する言語はC言語とし、C言語の標準的な関数の他に、VAX/VMSが提供する、メールボックスの生成･入出力及びタイマに関するシステム･サービスを用いる。

2.2 ソフトウェア構成

2.2.1 モジュール分割の方法

同一ないしは異なったクラスの複数のTCが同時に存在するシステムでは、処理の効率やインプリメントの容易さ、拡張性を考慮したモジュール構成が必要となる。

まず、各クラスのプロトコル手順を実行するプログラムは、できるだけクラスごとに閉じる必要があり、この理由から、独立したプロセスとして実行する方がよい。一方、複数のTCが存在することにより、各TCと上位および下位のコネクション(SC又はNC)との対応付けや各TCの状態を一元的に管理する機能も必要となる。本システムでは、前者をクラス･プロセス、後者をTマネジャというプロセスで実現している。

また、この他にPSIとインタフェースをとり、NSプリミティブとPSIコマンドとの変換を行うT/Nインタフェース、SLの代わりにユーザとインタフェースをとるS/Tインタフェースというプロセスを用いる。(図2参照)

以下にTLを構成するTマネジャとクラス･プロセスについて概説する。

2.2.2 Tマネジャ

Tマネジャでは、複数のTCに関係し一元的に行われるべき処理と、処理すべきクラス･プロセスが決定されていない時点での処理を実行する。その代表的機能を以下に示す。

(i) システムが提供する複数のTCを管理するために、各TCごとに図2.3に示すTCMT (Transport Connection Management Table)というテーブルを用意し、その情報を保持する。

(ii) アドレス対応テーブルを用いて、発着のTSAPアドレスから発着NSAPアドレス(本システムではDTEアドレス)を定める。

(iii) クラスを決定する。即ち、イニシエータではTCONreqの指定内容、NLのサービス品質(QOS)、自システム内に実装されているクラスを考慮し、希望クラス、代替クラスを決定する。しかし勧告上QOSの取り扱いが明確でないため、本システムでは、システムに実装されたクラスの内で3から0の順に選択する方法を用いた。レスポンダでは、受信したCR TPDUにより要求されたクラスと自システムの実装を比較し、そのTCで使用すべきクラスを決定する。

(iv) 自局のリファレンスの割当て、イニシエータでは、割付けるNCとそのNS CIDの決定、レスポンダでは、TS CIDの決定等、システムが提供するTCに関する一元的な情報を用いる処理を実行する。

以上の機能はTCを確立するための最初の入力のTCONreq又はCR TPDUが加えられた時点においてそれぞれの処理ルーチンの中で実行される。

(v) TPDUを処理するクラス･プロセスを決定するために、セパレーション、TPDUのTCへの関連付け、デマルチプレクシングのプロトコル機構を

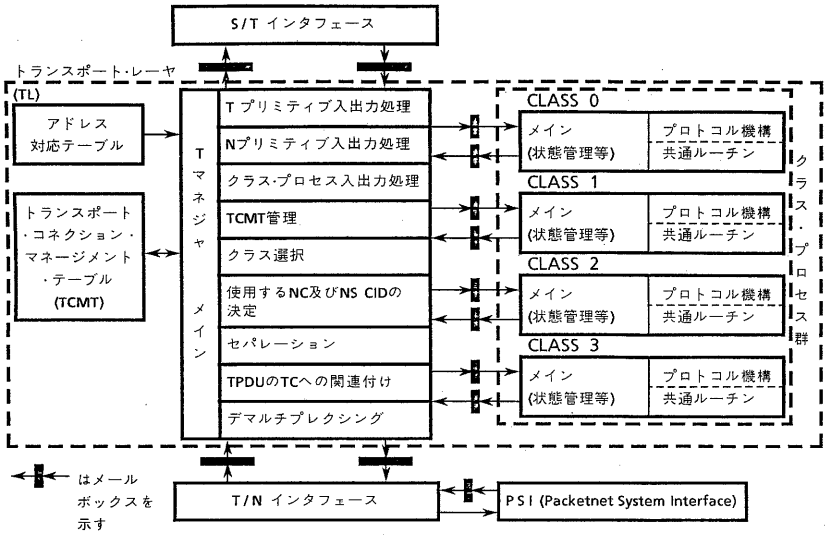


図2TPのソフトウェア構成図

自局のTSAPアドレス	(10)
相手局のTSAPアドレス	(10)
自局のリファレンス	(2)
相手局のリファレンス	(2)
自局のNSAPアドレス	(14)
相手局のNSAPアドレス	(14)
TS CID	(2)
NS CID	(2)
TCの状態	(2)
NCの状態	(2)
クラス 代替クラス・フラグ (class0) 代替クラス・フラグ (class1) 代替クラス・フラグ (class2) 代替クラス・フラグ (class3)	[4] [1] [1] [1] [1]

( )及び[ ]内の数字は各々バイト数、ビット数を示す。

図3TCMTの構成

実行する。

2.2.3 クラス・プロセス

これらのプロセスは、TマネジャでTCに対するTCMTが確保された後、クラスのプロトコル手順に従った各TCの処理を行う。それぞれのクラスのTCを管理し、各TCに対して、TPDUサイズ、拡張フォーマットの使用、ウィンドウ・サイズ等クラスごとのパラメータのネゴシエーション、状態管理、具体的なプリミティブの送出を含む、Tマネジャで行う以外のプロトコル機構等を実行する。各TCの管理には、TCMTに、クラス固有のパラメータや変数を加えたテーブルを用いる。各クラスのプログラムは、対応する状態遷移表を有し、各時点の状態と入力により、遷移表の示す処理を行うという形のプログラム構成にすることができる。

2.2.4システム内の情報の流れ

図2に示す様に、システム内の各プロセス間のインタフェースは、メールボックスを介して行われる。この場合S/TインタフェースまたはT/Nインタフェースから入力されるプリミティブに応じたTマネジャからクラス・プロセスへの情報の流れは以下の通りである。

(1) S/Tインタフェースからのプリミティブ受信時には、Tプリミティブに含まれるTS CIDより対応するTCMTを見出し、これとプリミティブをクラス・プロセスに伝達する。

(2) T/Nインタフェースからのプリミティブ受信時には、それがNDTindの場合にはセパレーションを行いNSDUをTPDUに変換し、自局/相手局のリファレンス等から関連するTCMTを捜し、これとTPDUをクラス・プロセスに伝達する。その他のプリミティブの場合にはNS CIDにより対応するTCMTを捜し、それとNプリミティブをクラス・プロセスに伝達する。

一方、クラス・プロセスではTマネジャから受信したTCMTと入力によりTC、NCの状態の遷移や各種プロトコル機構の処理を行い、変更を加えたTCMTとT/Nプリミティブを Tマネジャに伝達する。これらの情報によりTマネジャは最終的にTCMTを変更しS/TないしはT/Nインタフェースにプリミティブを出力し、一連の処理を終了する。

この様に本システムでは 隣接レーヤとの間で授受する情報はすべてTマネジャを通過する。

2.3TPのプロトコル機構の実現方法

TPのプロトコル機構には密接な関係を持つものがあり、プログラム設計においてこれらの関係を有効に用いる必要がある。そこで2.2に示したプロセス区分に従ったTPの実現において、プロトコル機構の処理の関連性、Tマネジャとクラス・プロセスへの処理の分担について述べる。

2.3.1 コネクション確立

本プロトコル機構は、新たにTCを生成するもので、TCONreq、CR受信、TCONresp、CC受信処理という4つの異なる機構に分割でき、各々が異なる処理形態をとる。

2.3.1.1 TCONreq処理

この機構はTマネジャとクラス・プロセスで分担して行われる。Tマネジャでは、この機構のうち2.2.2で述べた様に、提供される全てのTCに関連するTCMTやアドレス対応テーブルを用いる必要のある

機能を実行する。この機構はNCへの割付け、再割付け、マルチプレクシングの機構と関係が深く、これらの機構と共に、TマネジャのTCONreq処理ルーチンの中で実行される。

クラス･プロセスでは、Tマネジャの処理に引き続いて、新たにNCを張る場合はクラスにおけるNCへの割付けの処理を実行した後で、クラス固有のパラメータをセットしたCR TPDUを送信し、TS1タイマを起動し、TCの状態を遷移する。

2.3.1.2 CR受信処理

この機構も、Tマネジャとクラス･プロセスで分担される。Tマネジャでは、この機構のうち、CR TPDUを処理するクラス･プロセスの決定と、リファレンスとTS CIDの決定が行われる。この機構はNCへの割付け、再割付け、マルチプレクシングの機構と関係が深く、TPDUのTCへの関連付けの処理の中で呼ばれるCR処理ルーチンにおいて以下の手順で実行される。

(1)希望クラスに指定されたクラスが実装されている場合は希望クラスを選択し、実装されていない場合はCR TPDUの代替クラスと自システムに実装されているクラスからネゴシエーション規則に基づきクラスを選択する。

(2)まず、選択されたクラスがクラス1、3で、そのTCが再割付け中の場合は、再割付けの処理を行い、クラス･プロセスに通知する。

(3)それ以外の場合は、CR TPDUを運んだNDTindのNS CIDによりTCMTを全てサーチし、

(3.1)クラス0、1の場合は、対応するTCが解放されている場合に限り、初めてのCR TPDUとして、自局のリファレンスとTS CIDを決定し、クラス、自局相手局のリファレンス、TS CID及びNSAPアドレスをTCMTに設定する。

(3.2)クラス2、3の場合は、

①対応するTCが解放されていれば、NCONind後初めてのCR TPDUとして(3.1)の処理を行い、

②そうでなければ、CR TPDUの指定する相手局のリファレンスを共有するTCMTがあれば、そのTCに対するCR TPDUであると解釈し、

③それ以外は、新たにTCMTを確保し、新たなCR TPDUとして処理する(マルチプレクシングの処理)。

(4)その後TCMTとCR TPDUをクラス･プロセスに伝達する。

クラス･プロセスでは、CRの持つパラメータと自システムの機能とを比較し、そのTCで用いるパラメータを決定し、TCONindを送出し、TCの状態を遷移する。

2.3.1.3 TCONresp処理

CR TPDUを受信した後の処理がクラスにより異なるため、この機構は主にクラス･プロセスで実行される。クラス･プロセスはTマネジャからTCONrespを受信するとクラス固有のパラメータをセットしたCC TPDUを送出しTCの状態を遷移する。

2.3.1.4 CC受信処理

この機構では、クラス･ネゴシエーションをTマネジャが、それ以外をクラス･プロセスが実行する。

Tマネジャは、CC TPDUを受信すると、その希望クラスのプロセスに通知する。クラス･プロセスはCC TPDUの選択クラスが自分と同じ場合は、パラメータのセットを行い、TS1タイマを停止し、TCONconfを送出しTCの状態を遷移する。

CC TPDUの選択クラスが自分と異なる場合はTS1タイマを停止し、そのCC TPDUとTCMTをTマネジャに送る。TマネジャはTCMTのクラスを選択クラスに設定し、それらを新たなクラス･プロセスに伝達し、そのプロセスでCC TPDU受信の処理が行われる。

2.3.2 NCへの割付け、障害後の再割付け

NCへの割付けの機構はTCをNCに割付けるものであり、障害後の再割付けの機構はクラス1、3においてNLから通知されるNCの切断より回復するためのものである。これらの機構は、イニシエータとレスポンダで処理内容が異なる。

イニシエータでは、NCへの割付けの処理の内、使用するNCとそのNS CIDの決定はTマネジャのTCONreq処理ルーチンで行われ、NCONreqの送出とNCONconf受信によるTCとNCの状態の更新はクラス･プロセスにより行われる。またイニシエータにおける再割付けは同じNCを再度割付ける処理として、クラス･プロセスで行われる。即ち、クラス1、3のプロセスはNDISindを受信すると、TTRタイマが動いていなければそれを起動し、それまで割り当てられていたNCのNS CIDを用いて、NCONreqを送出する。

レスポンダでは、NCへの割付け及び再割付けの内TCMTの設定は関係が深く、これらの機構はTマネジャにおいて以下の様に、NCONindの処理とTPDUのTCへの関連付けの処理の中で行われる。

①NCONindを受信すると空きのTCMTを確保してそのNCのNS CID、NCの状態、NSAPアドレスを設定しNCONrespを送出する。再割付け中の場合は、この時点においてはそのTCに関して、NCの切断が通知されたTCMTと新たに張られたNCに関連する

TCMTの二つが存在する。
②次にそのNCでのNDTindに対するTPDUのTCへの関連付けの処理において、NSAPアドレス対とリファレンスを共有し且つ再割付け中であるTCMTを捜す。その様なTCMTが存在すれば再割付けの処理として、再割付け中のTCMTのNS CID及びNCの状態を更新し、TCMTとTPDUをクラス･プロセスへ伝達し、新たなNCに関連するTCMTを解放する。
再割付け中のTCMTが存在しなければNCONind後初めて受信されたTPDUとして処理し、それがCR TPDUの場合はCR受信処理の処理を行う。

一方レスポンダにおける再割付けの内、状態管理、TWRタイマの起動、そのタイムアウトの処理等はクラス･プロセスで行われる。即ち、クラス1、3のプロセスはNDISindを受信すると、TWRタイマが動いていなければそれを起動し次に受信されるTPDUを待つ。

2.3.3 マルチプレクシング/デマルチプレクシング

本機構は複数本のTCが同時に一本のNCを共有するものであり、コネクション確立、TPDUのTCへの関連付けと関係が深く、次の様に実行される。

･コネクション確立時のマルチプレクシングの処理はTマネジャのTCONreq処理ルーチン又はCR処理ルーチンの中で行われる。

･TPDU受信時のデマルチプレクシングは、TPDUのTCへの関連付けの処理の中で行われる。

またこの機構に関して、本システムではマルチプレクシングを行っている場合のNRSTind又はNDISindの処理を以下の様に行っている。Tマネジャがそのに割り付けられているTCの内一本のTCに関連するTCMTを選択しそのTCMTとプリミティブをクラス･プロセスに伝達する。クラス･プロセスでは送られたTCMTの示すTCと同一のNCに多重化された全てのTCに対してNRSTind又はNDISindの処理が行われる。対応するNRSTresp又はNCONreqはNRSTind又はNDISindが通知されたTCにより出力される。この様な方法を用いたのは同一クラスのTCのみに多重を行うという本システムの前提条件下で、Tマネジャとクラス･プロセス間の情報の流れを少なくするためである。

2.3.4 コネクション拒否

本プロトコル機構は、CR TPDUを受信したTLがTCの確立を拒否するというものであり、CR受信処理の機構と関係が深く、Tマネジャとクラス･プロセスにおいてで行われる。Tマネジャでは、空きTCMTが確保できなかった場合かクラス･ネゴシエーションができなかった場合に、又クラス･プロセスではパラメータのネゴシエーションができなかった場合にDR TPDUを送信する。

2.3.5 コンカティネーション/セパレーション

本機構は複数のTPDUを一つのNSDUで伝送する際に必要で、セパレーションはTマネジャにおいて、NDTindを受信した場合の最初の処理として行われる。

一方、コンカティネーションでは、一般的には、異なる入力に対する処理の結果出力されるTPDUを1つのNSDUに連結する必要がある。そのためには一つの入力の一連の処理に、他の入力の処理を割り込ませて実行する必要がある。本システムでは、各入力の処理は一連の処理として実行するプログラム構造であるため、1つの入力に対して複数のコンカティネーション可能なTPDUが出力される場合にのみこの処理を行った。

2.3.6 フロー制御、DT TPDU番号付け

フロー制御は、他のレーヤとは独立にDT TPDUの流れを制御するもので、DT TPDU番号付けは再同期、フロー制御等に使用される。これらの機構は共にクラス･プロセスにおいて、DT TPDUを送受信するときに用いられ、DT TPDU送信時は、DT TPDU番号付けはフロー制御のプロトコル機構から呼ばれる。

2.3.7 TPDUの保留

TPDUの保留はTPDUの損失から回復するために使用する機構で、送信したTPDUを保留する機能、保留したTPDUを参照する機能、保留したTPDUを消去する機能が用意され、クラス･プロセスにおけるコネクション確立、フロー制御、優先データ転送、通常解放、再同期から呼ばれる。

2.3.8 再同期

本機構は、クラス1、3でNRSTindを通知された後、又は再割付けに引き続いて、TPDUの流れを正常にするためのもので、クラス1、3のプロセスで実行される。その場合、NRSTindを受信した後は引き続いて送信されたTPDUを受信する可能性があるため、再同期の処理がNRSTindの後のものか再割付けの後のものかを知っている必要がある。この機構は、TCの状態により異なる処理を必要とし、また保留、フロー制御(データ再同期において)のプロトコル機構を使用する。

TPプログラム実現の上ではTマネジャの構成が重要であり、上述のプロトコル機構の相互関係をより明確にするため、Tマネジャの処理の流れと実行されるプロトコル機構を図4に示す。

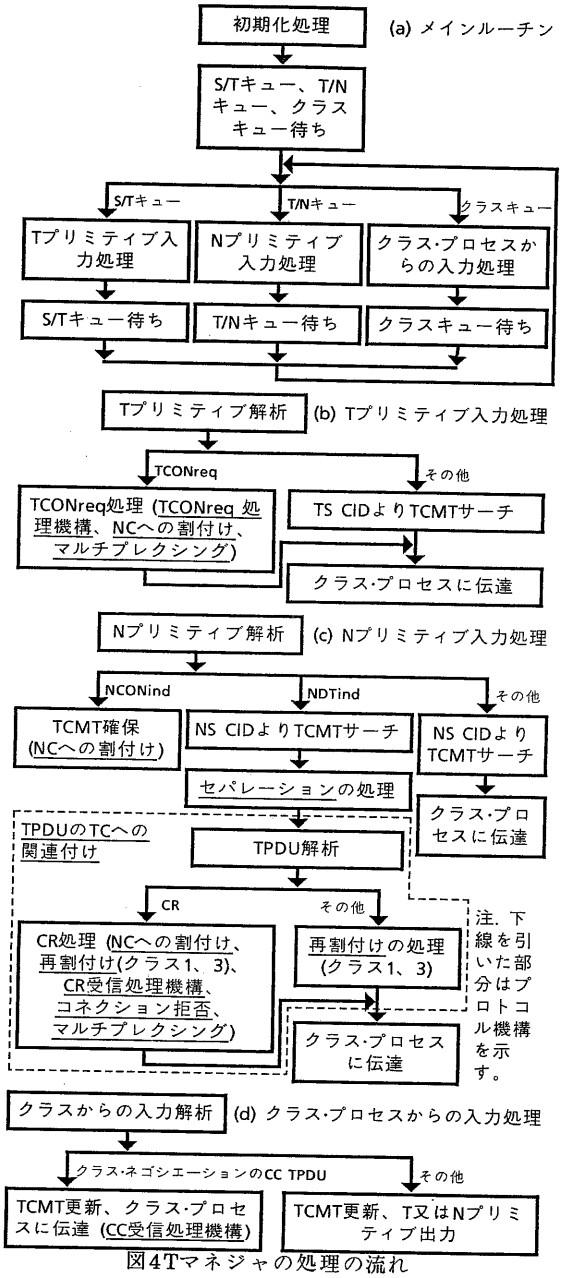


図4Tマネジャの処理の流れ

2.4結果と考察

本TPプログラムは以上の様に、TP自身の構造に基づいて作成されており、一般性の高い実現形態であるといえよう。

表2及び3に、プログラムの規模を推定する意味で、それぞれTマネジャ、クラス・プログラムのソース・プログラムの実効行数を、プロトコル機構又は機能別に示す。これは、C言語のソース・プログラムから、コメント行と空白行を除いた行数である。ここでクラス・プログラムの共通モジュールというのは、複数のプロトコル機構により共通に使用されているルーチンである。

表2Tマネジャの実効行数

モジュール名	ステップ数
メイン(宣言文等も含む)	318
Tプリミティブ入出力処理	227
Nプリミティブ入出力処理	206
クラス・プロセス入出力処理	131
TCMT管理	189
クラス選択	215
使用するNC及びNS CIDの決定	29
セパレーション	48
TPDUのTCへの関連付け/デマルチプレクシング	429
合　計	1792

表3クラス・プログラムの実効行数

モジュール名	class0	class1	class2	class3
メイン(状態管理等)	363	1178	544	1264
共通モジュール	389	955	497	1134
NCへの割付け	46	68	46	68
TPDU転送	31	24	31	24
セグメンティング/アセンブリング	148	185	226	296
コンカティネーション				54
コネクション確立	254	60	247	60
コネクション拒否	40	42	43	42
通常解放　暗示	23			
明示		353	59	352
エラー解放	56		86	
DT TPDU番号付け		21	19	57
優先データ転送		226	206	341
障害後の再割付け		162		186
TPDUの保留		322		550
再同期		551		545
マルチプレクシング			19	10
フロー制御			255	385
リファレンス凍結		13		13
プロトコル・エラーの処理	47	24	41	24
合計	1397	4184	2319	5405

これらの数値はプログラムの記述方法により変動するが、OSIのTPプログラムの規模の概数が把握できる。これらの表において

① クラス1、3における、コネクション確立の値が小さいのは、これらのクラスでは再同期でもCR及びCC TPDUを作成する場合があり、これらの機能を共通ルーチンとして用意したためである。これらのルーチンの実効行数を加算すると、クラス1、3のコネクション確立の値はそれぞれ、292、309行となる。

② 通常解放の処理は、クラス0では暗示の解放であり、またクラス1、3はクラス2に比して状態遷移の種類が多く、状態による処理が異なること等により差が生じる。

である。

3.オートマトン・モデルによるTPの製品検証方式

3.1製品検証の実行方式

多くの異なった実現形態の製品を対象とするプロトコルの製品検証では、TPプログラムの詳細を調べて試験を行うことは困難であるために、TPプログラムの入出力応答を観測するという形で行われる。ここでは製品検証の方式として、図5の様な、検証シス

テム(プロトコル·テスタ)と被検証システムを接続しプロトコル·テスタがテスト系列を発生することにより検証を実行する方式を用いる。

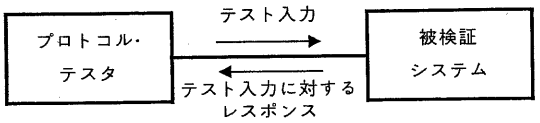


図5プロトコル製品検証の形態

OSI製品ではOSI参照モデルのレーヤ構造に従ったプログラム構造を実現しており、TPを実現する機能モジュールは、T又はNプリミティブ入力、相手機能モジュールからのTPDU入力並びにシステム内部のタイマ入力の三種類の入力を有する。その場合被検証TPプログラムに対して、外部のテスタからのTPDU、Nプリミティブは容易に与えることができるが、Tプリミティブは外部のテスタからテスト入力として与えるには工夫を要する。

そこで外部テスタからのテスト入力の与え方による2種類の検証方式が考えられる。

方式1: 検証のためのソフトウェアを被検証システム内に用意し、TPDUとNプリミティブに加えてTプリミティブも用いて検証する方式。

方式2: プロトコル·テスタから与えることのできるTPDUとNプリミティブのみを用いて検証する方式。

方式1で用いる検証のためのソフトウェア(検証用プログラムと呼ぶ)は、プロトコル·テスタからの指示によりTプリミティブを他の入力と同期的に発生する機能を有する。

方式1は検証能力は高いが、追加のプログラミングの必要がある。一方、方式2は実現性は高いが、被検証プログラムの全ての動作を検査できないという弱点がある。

3.2 オートマトン·モデルによるテスト系列の構成

製品検証のために、対象となるTPの動作を厳密に表現するオートマトン·モデルを作成する。

次に検証用のテスト系列を、次の様な有限オートマトンの同定の方法に基づいて構成する[2]。

(1).検証対象となるオートマトンの状態の数が正しいオートマトンの状態数と等しいという前提条件のもとで、判定系列を用いて各状態を判定する。判定系列とは各状態固有の出力列を送出させる入力列のことである。

(2).テスト系列では、まず各状態に判定系列を加えて、全ての状態が存在することを確認する。つぎに各状態にそれぞれの入力を加えた場合の遷移と出力を確認する。これはオートマトンの状態を各状態に設定し、入力を加え出力を観測し、つぎに判定系列を加えて遷移先の状態を調べるという手順で行われる。

方式1ではTPプログラムの全ての入力を検証に用いることができるために、TPプログラムを通常の有限オートマトンとみなし、この方法で検証用テスト系列を構成できる。

一方、方式2では

·判定系列は原則的にプロトコル·テスタからのTPDU、Nプリミティブによって構成する。

·有効に検証できる範囲は、テスタからの入力によって互いに遷移できる状態の集合に限られる。

という条件の下で、検証用テスト系列を構成する必要がある。

3.3製品検証実験

オートマトン·モデルによる製品検証の実験を、VAX11/780上に作成したTPプログラムに対して行った。実験ではプロトコル·テスタもVAX11/780上に実現し、折り返し試験の形態をとった。方式1の検証実験形態を図6に示す。TPのプロトコル·テスタ

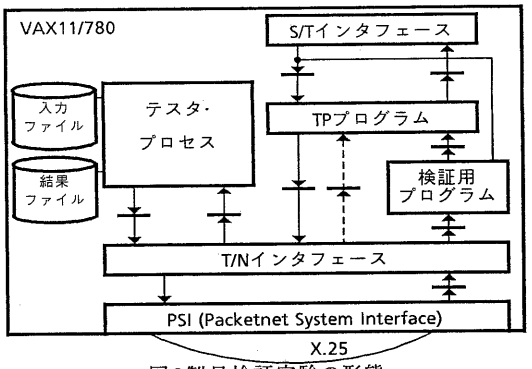


図6製品検証実験の形態

の処理を実行するテスタ·プロセスは、T/Nインタフェース上に実現され、入力ファイルからテスト入力と期待する応答を読み、入力がTプリミティブである場合はDT TPDUのユーザ·データにその発生指示をセットして送信し、それ以外の場合はNCを通じてその入力を与える。その後、一定時間応答を待ち、応答を解析し、結果ファイルに出力する。また検証用プログラムは、被検証プログラムの変更が少なくなるようにT/NインタフェースとTPプログラムの間にプロセスの形で実現した(変更はTPプログラムのメールボックス名を変えるだけとなる)。検証用プログラムは受信されるTPDUを監視し、Tプリミティブの発生を指示したDT TPDUならば指定の上位プリミティブを指定のタイミングでS/Tから

TPへのメールボックスにいれ、DT TPDUを破棄する。それ以外はそのままTPへ渡す。

方式2では、検証用プログラムが不要の外は方式1と同様な形態で製品検証実験が行われる。検証においてTプリミティブが必要な場合は、テスタ・プロセスはその発生をコンソールにより指示し、そのTプリミティブは他の入力とは非同期にS/Tインタフェースから加えられる。

3.4クラス0の製品検証

TPのクラス0・イニシエータの製品検証を例に示す。クラス0・イニシエータのオートマトン・モデルは勧告の示す状態遷移表にプロトコル・エラーの処理中の状態を加えて表4の様に規定される。

表4クラス0(イニシエータ)の状態遷移表(一部)

STATE / EVENT	CLOSED	WFNC*	WFCC*	OPEN	ERROR
	N	A1	A2	B	C
TCON req	A1 NCONreq	無視	無視	無視	無視
NCON conf	有り得ない	A2 CR TS1 S**	有り得ない	有り得ない	有り得ない
CC	有り得ない	有り得ない	B TCONconf TS1 P**	C ER TS2 S	C

*WFNC:Wait for Network Connection, WFCC:Wait for CC
**Sはタイマ・スタート、Pはタイマ・ストップを示す。

クラス0・イニシエータの方式1による検証は

N:	TCONreq, NCONconf	(出力: NCONreq, CR)
A1:	NCONconf	(出力: CR)
A2:	TDTreq, CC, TDTreq	(出力: 無し, 無し, DT)
B:	TDTreq, CC, TDTreq	(出力: DT, ER, 無し)
C:	TDTreq, CC, TDTreq	(出力: 無し, 無し, 無し)

という判定系列を用いて図7(a)の様に実行できる。

また方式2の製品検証は、状態CLOSEDにおけるTCONreqを非同期に加えることにより(これを(tconreq)で示す)、判定系列

N:	(tconreq)	(出力: NCONreq)
A1:	NCONconf	(出力: CR)
A2:	CC,DT,CC	(出力: 無し,無し,ER)
B:	CC,DT,CC	(出力: ER,無し,無し)
C:	CC,DT,CC	(出力: 無し,無し,無し)

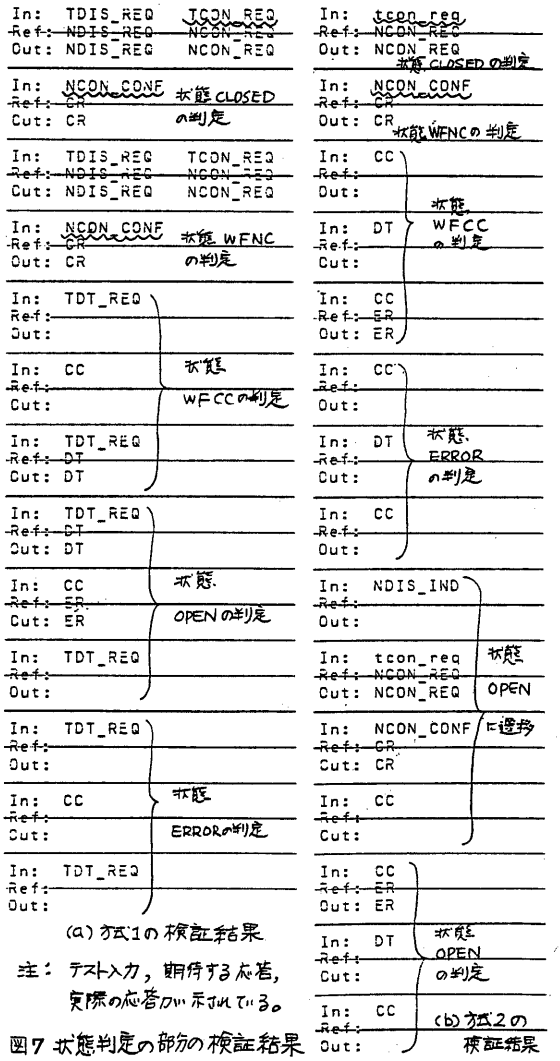
を用いて図7(b)の様に実行できる。

方式1ではテスト系列長は521入力であり、遷移表のうち検査対象となる58遷移の全てを検証できる。検証用プログラムはC言語で271行であり、規模は小さい。また方式2ではテスト系列長は401入力(うち非同期に加えるTCONreqが59)で、遷移表のうち39遷移を検証でき、かなりの範囲が検証可能である。検証実験の結果、状態WFNC及びWFCCでのTDISreqの処理、状態ERRORでのTDISreq、CC、ER、NRSTindの処理に誤りが発見された。勧告の遷移表ではERRORの状態が明確でないために状態ERRORでの処理に誤りが多く発見された。これにより検証で用いたオートマトン・モデルの有効性が実証された。

4. おわりに

今後ともOSI各種プロトコルのインプリメントと製品検証について検討していきたい。最後に、日頃御指導頂くKDD研究所鍛冶所長、野坂副所長、深田次長、小野情報処理研究室長に感謝します。

参考文献 [1]: CCITT, Rec. X.214, X.224, March, 1984
[2]: 斉藤,加藤,猪瀬,"オートマトン・モデルによるHDLCプロトコル製品検証の一方式",信学論(D),J63-D,8, Aug., 1980
[3]: CCITT, Rec. X.200, March, 1984
[4]: 鈴木,加藤,"OSIトランスポート・プロトコルのインプリメントについて", 第28回情処全大, 2D-3, March, 1984
[5]: 加藤,鈴木,"OSIトランスポート・プロトコルの製品検証", 第28回情処全大, 2D-4, March, 1984



(a) 方式1の検証結果

注: テスト入力, 期待する応答, 実際の応答が示されている。

(b) 方式2の検証結果

図7 状態判定の部分の検証結果