# OSIトランスポートおよびセション・プロトコルの実装†

鈴木健二†† 加藤聰彦†† 浦野義頼††

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に開放型システム間相互接続(OSI)の標準化を進めており，特にトランスポートレーヤとセションレーヤについてはそのプロトコル(TP, SP)標準が作成されている．筆者らは，OSIプロトコルを通信インタフェースとして持つホストシステムの構築を目指し，①TPの全クラス，SPの全仕様を実現する，②複数のコネクションを同時に確立することができ，これらは異なったクラスや機能単位を使用しても良い，の2点を満足するTP/SPプログラムをVAX上にC言語で実装した．これらの実装のうち，TPの実装では，複数のプロトコル機構が関連しあい，コネクションの管理，コネクション確立の処理，障害後の再割り付け等の処理が複雑であること，またSPの実装では，個々の機能単位の間の関係が少なく，独立に実装できる反面，状態遷移表が複雑となり実装に工夫を要すること，などの問題点がある．本論文では上記の問題点に対して筆者らが採用した解決方法を述べるとともに，実装の結果およびTPとSPの実装比較について考察する．

## 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)の標準化を進めており，特にトランスポート・レーヤ(TL)とセション・レーヤ(SL)についてはそのサービス定義とプロトコル仕様が確定し，両機関とも勧告化並びに標準化作業を完了している[1),2)]．

これらのレーヤは，業務内容に応じて必要となるデータ転送機能および対話機能を，さまざまな業務に対して共通に提供するためのものである．このため，OSIプロトコルに従った通信システムの構築においては，これらのレーヤのプロトコルをいかに実装するかが重要な問題となる．

最近TLやSLの一部の仕様を実装したプログラム製品が発表されているが，筆者等は標準化の進捗に合わせて，トランスポート・プロトコル(TP)のすべてのクラス，セション・プロトコル(SP)の全仕様をVAX上に作成している[3)-5)]．さらに，これらのプログラムおよびその一部をパーソナルコンピュータへ移植したプログラム[6)]は現在パソコン通信で実用化するに至っている．

本稿ではTPとSPの全仕様の実装概要，キーポイント，結果について述べる．

## 2. TPとSPの概要

### 2.1 TPの概要

TLはOSI参照モデルの下位から第4層に位置し，SLが要求するサービス品質を満足するデータ転送機能を提供するもので，クラス0から4までの5種類のプロトコル・クラスが規定されており，各クラスに応じて表1のようなプロトコル機構が定義されている．使用するクラスは，トランスポート・コネクション(TC)をネットワーク・コネクション(NC)に多重化するか否かと，想定するNCの品質により決定される．

クラス0と1は多重化機能を持たず，クラス2から4は多重化機能を持つ．また，NCの品質がよくTLで誤り回復を行う必要がない場合はクラス0と2が使用される．ネットワーク・レーヤ(NL)からの切断やリセットが生ずる場合は，クラス1または3が使用される．これらのクラスは障害後の再割付けや再同期のプロトコル機構を用いて，NLから通知される障害から回復する．さらにNCの品質が悪く，伝送誤りをTLで検出かつ回復する必要がある場合はクラス4が使用される．クラス4では，伝送誤りを検出するためにチェックサム，伝送誤りによるTPDU (Transport Protocol Data Unit)の消失からの回復にはタイムアウトによる再送，TPDUの重複や順序誤りからの回復にはリシーケンシングの各プロトコル機構が設けられている．またNLの障害からの回復とスループットの向上等を目的として，一本のTCを複数のNCに割付けるスプリッティング/リコンバイニングの機能も提供する．

### 2.2 SPの概要

SLはOSI参照モデルの下位から第5層に位置し，業務内容に応じて多種多様に存在するアプリケーション・プロセス間で同期をとりながら，何らかの意味を

† Implementation of OSI Transport and Session Protocols by KENJI SUZUKI, TOSHIHIKO KATO and YOSHIYORI URANO (KDD Kamifukuoka R & D Laboratories).

†† 国際電信電話(株)上福岡研究所

表 1 TP のプロトコル機構
Table 1 Protocol mechanisms of Transport protocol.

| プロトコル機構 | 種別 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| 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 | | * | * | * | * | * |
| 普通解放 | 暗示 | * | | | | |
| | 明示 | | * | * | * | * |
| エラー解放 | | * | | * | | |
| TPDU の TC への関連付け | | * | * | * | * | * |
| DT TPDU 番号付け | 普通 | | * | m | m | m |
| | 拡張 | | | o | o | o |
| 優先データ転送 | NL の普通データ | | 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 | | |
| チェックサム | 有り | | | | | * |
| | 無し | * | * | * | * | o |
| リファレンス凍結 | | | * | | * | * |
| タイムアウトによる再送 | | | | | | * |
| リシーケンシング | | | | | | * |
| インアクティビティ制御 | | | | | | * |
| プロトコル・エラーの処理 | | * | * | * | * | * |
| スプリッティング/リコンバイニング | | | | | | * |

*: 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)

もつデータ (情報) を送受する方法を管理する. これは, 一般的に対話 (ダイアログ: dialogue) の管理と総称され, 具体的には以下に示すような12種類の機能単位 (Functional Unit) として整理される. これらの機能単位の使用は, セション・コネクション (SC) の確立時にセション・サービス (SS) ユーザによりネゴシエートできる.

(1) カーネル: SC の確立/解放, データ転送の SL の基本的な機能を提供する.

(2) ネゴシエート・リリース: SC の解放指示を受信した際, これを拒否し, SC の継続的使用を許す.

(3) 半二重: データ・トークン (送信権) を用いた半二重通信モードを実現する.

(4) 全二重: 普通データの両方向同時転送を行う.

(5) 優先データ: 普通データの流れとは別に優先的にデータを転送する.

(6) タイプト・データ: データ・トークンの有無にかかわらず, データを送信できる機能を提供する.

(7) ケーパビリティ・データ交換: 受信確認を必要とするデータ転送を実現する機能単位で, 例えば相手の受信端末能力の折衝に用いられる.

(8) 小同期: 一連の普通データ転送の途中で小同期点を挿入する機能を提供する.

(9) 大同期: 一連の普通データ転送の途中で, 大同期点を挿入する. 大同期点は小同期点と異なり必ず

送達確認を必要とし，送達確認が得られるまで新たな普通データ，優先データの送出ができない．

(10) 再同期：同期点の再設定およびトークンの再配置を行う機能を提供する．

(11) 例外報告：SSプロバイダないしはSSユーザが検出したエラーを，各々SSユーザおよび相手SSユーザに通知する機能を提供する．

(12) アクティビティ管理：プロセスが行うひとまとまりのデータ転送をアクティビティとして管理し，その開始，終了，再開，中断，廃棄を可能とする．

## 3. 実装の方針

### 3.1 方　　針

TPとSPを実装するに当たり，次のような基本設計方針をたてた．

① 各レーヤのプログラムは独立に実行されるプロセスとして実現する．これらのプログラムは上位プログラムとも独立に実行され，またSPプログラムは複数の上位プログラムと同時にインタフェースできるようにする．

② 複数のTCまたはSCを同時に確立し，システム内のリソースを管理することができる機能を実装する．これらのコネクションは，TCの場合には異なったクラスでもよく，SCの場合には異なる機能単位を使用できるようにする．

③ プレゼンテーションレーヤ(PL)とSL間，SLとTL間およびTLとNL間のインタフェースは，各々のサービス定義の標準に定められたプリミティブのみを使用し，そのフォーマットとしてはPDU (Protocol Data Unit) に近い形式を用いる（図1参照）．

④ SSAP (Session Service Access Point) とTSAP Transport SAP) が複数存在することを許す．またNSAP (Network SAP) は，DTEアドレスを固定的に割り当てる．ここでSSAPは上位プログラムを識別するために使用する．またSSAP，TSAP，NSAPごとに複数のSCEP (Session Connection Endpoint)，TCEP (Transport CEP)，NCEP (Network CEP) が存在することが可能な構成とする．

⑤ 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に，SS CID (Session Service Connection Identifier)，TS CID (Transport Service CID)，NS CID (Network Service CID) と呼ぶ識別子を使用する．これらの付与の仕方は，イニシエータ側では，上位層が割り当てて確立要求 (CONreq) のプリミティブで下位層に通知し，レスポンダ側では，下位層が割り当てて確立指示 (CONind) のプリミティブで上位層に通知する．CIDの重複を避けるために，本システムでは上位層が割り当てる場合は奇数，下位層が割り当てる場合は偶数を使用する．

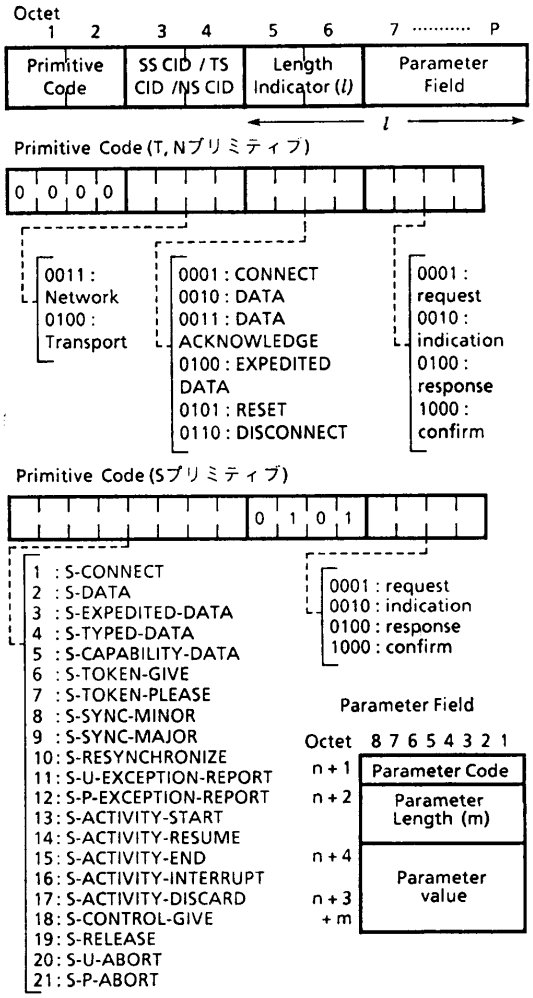


図1 プリミティブのフォーマット
Fig. 1 Format of primitives.

⑥ 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，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実現し，フロー制御のための特別なプリミティブは用意しない．

⑦ TPの実装では，TPの標準に従い，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．また

コンフォーマンス規則に従ったクラス・ネゴシエーション機能を実現する．さらに運用に応じて，任意のクラスを選択的に実行できるようにする．

⑧ SPの実装では，同一のプログラムで複数の上位プログラムが異なる機能単位を使用できるようにし，またプログラムを移植する際に，容易に機能単位をくくりだすことができるソフトウェア構成とする．

以上の方針のもとでTP/SPをVAX 11/780のOSであるVMS下で実装する．プロセス間の通信は，VAX/VMSが提供するメールボックスにより行う．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の送達確認および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ほかに，VAX/VMSが提供するメールボックスの生成や入出力およびタイマに関するシステム・サービスを用いる．

### 3.2 ソフトウェア構成

次に作成したTPおよびSPプログラムのソフトウェア構成について述べる．

TPの実装では，各クラスのプロトコル手順を実行するプログラムをできるだけクラスごとに閉じた形で実現することと，運用に応じて必要なクラスのみを実行させることを目的として，各クラスのプログラムを，独立したプロセスとして実行させる．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または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．そこで，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．

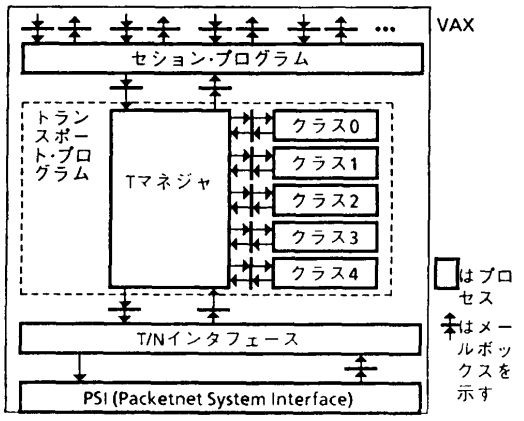


図2 ソフトウェア構成
Fig. 2 Software configuration.

SPの実装では，1つのプログラムで，異なる上位プログラムが必要とする機能単位を提供するために，TPと対照的にSPの機能をすべて1つのプロセスとして実現する必要がある．またこのプロセスは，複数の上位プロセスとインタフェースするためのメールボックスを持つ．

また，このほかに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インタフェースというプロセスを用いる．

このソフトウェア構成を図2に示す．

## 4. TPの実装

### 4.1 概　　要

#### 4.1.1 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．

① システムが提供する複数のTCおよびNCとTCの対応関係を管理するために，後述のように，TCごとにTCMT (Transport Connection Management Table) というテーブルを，NS CIDごとにNS CID MT (NSCID Management Table) というテーブルを用意し，必要な情報を保持する．

② 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．

③ クラスを決定する．すなわち，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．しかし現在の標準化においては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クラス4からクラス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．

④ 自局のリファレンスの割当て，割付けるNCとそのNS CIDの決定（イニシエータ側），TS CIDの決定（レスポンダ側）等，TCに関する一元的な情報を用いる処理を実行する．

⑤ 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実行

する．

#### 4.1.2 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 TC の処理を行う．各クラス・プロセスではそのクラスに属する 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，ウインドウ・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，プリミティブの送出等を含む，Tマネジャで行う以外のプロトコル機構を実行する．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となっている．

#### 4.1.3 TP プログラム内の情報の流れ

図2に示すように，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．この場合のセション・プログラムまたは 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 マネジャとクラス・プロセスとの間の情報の流れは次のとおりである．

① セション・プログラムからのT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，TCMT 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 N 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，TCMT と N プリミティブをクラス・プロセスに伝達する．

③ 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う．その後変更を加えた TCMT とTまたは 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，セション・プログラムないしは 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．

このよう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．


**図 3** TCMT と NSCID MT の構成
Fig. 3 Structure of TCMT and NSCID MT.

### 4.2 TP 実装のキーポイント

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．そこでプロトコル機構の処理の関連性などの TP の実装のキーポイントについて述べる．

#### 4.2.1 T マネジャにおける TC と NC の管理

Tマネジャでは，同時に接続される TC と NC の間の，マルチプレクシング，スプリッティング等を管理するために，4.1 で述べたように，次のようなテーブルを用意している．

① TCMT：TC を管理するために使用するテーブル（**図3**参照）．TPDU の TC への関連付けや障害後の再割付の処理を効率的に行うために自局のリファレンス順に並べられている．

② NSCID MT：TC と NC の割当て状況を管理するためのテーブル（図3参照）．受信した N プリミティブと NSCID MT を効率的に関係付けるために NS CID 順に並べられている．

さらに，TCMT が自局のリファレンス順に並べられているため，セション・プログラムから受信したTプリミティブの TS CID から対応する TCMT を効率的に選択するために，TS CID と自局のリファレンスの対を保持する TSCID MT (TS CID Management Table) という補助的なテーブルを使用している．

これらの3種類のテーブルは自局のリファレンスと NS CID によりリンクされ効率的に選択される．マルチプレクシング，スプリッティングの場合を含むテー

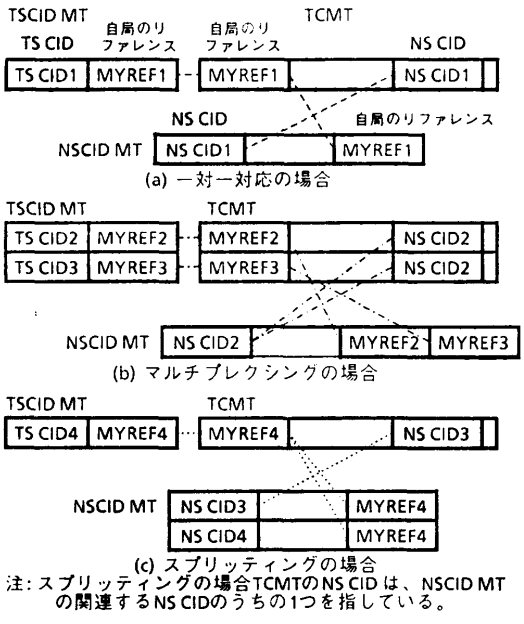


**図 4** TSCID MT, TCMT, NSCID MT のリンクの状態
Fig. 4 Relationship of TSCID MT, TCMT and NSCID MT.

ブルのリンクの状態を図 4 に示す．また，クラス・プロセスとの情報のやりとりは，TCMT を用いて行う．そのため TCMT と NSCID MT の間に，自局の NSAP アドレスと NC の状態の情報が重複している．

これらのテーブルは，次のように確保され使用される．イニシエータ側では SL からのコネクション確立要求 (TCONreq) に対して，空きの TCMT を確保し，自局のリファレンス，クラス，使用する NC を決定する．NC を新たに確立する場合は空きの NS CID を選択し NSCID MT を確保する．このあとテーブルに各情報をセットする．

レスポンダ側ではネットワーク・レーヤからのコネクション確立指示 (NCONind) により，NSCID MT を確保する．次にその NC から受信される最初の CR TPDU により TCMT を確保し自局のリファレンス，クラスを決定し，テーブルに各情報をセットする．

テーブルが確保された後は，Tプリミティブ受信時はTプリミティブの TS CID から TSCID MT により，Nプリミティブ受信時はその NS CID または TPDU に含まれる自局のリファレンスから NSCID MT により，TCMT を選択する．

### 4.2.2 コネクション確立

コネクションを確立する時点では，コネクション確立，NC への割付け，障害後の再割付け，マルチプレクシング等のプロトコル機構が複数に関係している．

コネクション確立時には，TCMT および NSCID MT の更新，TS CID/NS CID の決定等はTマネジャで行われ，各クラス固有のパラメータ値の決定等はクラス・プロセスで行われる．

以下に特に処理が複雑な CR 受信時と CC 受信時の処理を示す．

(1) CR 受信処理

CR 受信時には，Tマネジャにおいて，それが最初の CR であれば，TCMT を確保する必要があるが，再送された CR の場合はすでに確保された TCMT を用いる．受信した CR TPDU の DST-REF が NSCID MT の自局のリファレンスに設定されていれば，それに対応する TCMT を選ぶ．しかし，NSCID MT の自局のリファレンスに設定されていない場合も，障害後の再割付け等で，再送された CR TPDU である可能性がある．

そこで，Tマネジャでは 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次の手順が実行される．

① 受信した CR TPDU の DST-REF が NSCID MT に登録されている場合は，それから参照された TCMT が対応する TC に関するものであるとして，その TCMT と受信した TPDU をクラス・プロセスに送る．

② 受信した CR TPDU の DST-REF が NSCID MT に登録されていない場合は，TCMT 全体を DST-REF の値でサーチし，該当する TCMT を探す．

- 該当するものがなければ，新たに受信した CR TPDU として，空の TCMT を確保して，クラスを決定し，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 および NSAP アドレスを TCMT に設定し，クラス・プロセスに送る．この時，クラス 2，3 および 4 の場合はマルチプレクシングが行われる場合がある．
- 該当するものがあれば，クラスが 1 か 3 で再割付け中の場合，またクラス 4 の場合は，その TC に関連付けられる CR TPDU として，TCMT と受信した TPDU をクラス・プロセスに送る．

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） CC 受信処理

CC 受信時にはクラス・ネゴシエーション・ダウンが行われる場合がある．この場合はCC TPDUを受信した時にTCMTとNSCID MTのクラスの情報を変更し，新たなクラス・プロセスに処理をまかせる必要がある．しかし，CC TPDU受信後の処理は，クラス1または3で再同期が行われている等，もとのクラスに依存することがあるために，受信したCC TPDUはいったんもとのクラス・プロセスに送られる．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．

#### 4.2.3 障害後の再割付け

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．

本システムでは処理を簡単にするため，イニシエータ側は再割付けにおいて同じNCを再度割り付けるようにしている．そのため，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，再割付けはTマネジャにおいて以下のように，NCONindの処理とTPDUのTCへの関連付けの処理の中で行われる．

① NCONindを受信すると空きのNSCID MT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，そのTCに関してTCMTとNSCID MT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かつ再割付け中であるTCMTを捜す．そのようなTCMTが存在すれば再割付けの処理として，TCMTとNSCID MTのリンクをとり，TCMTのNS CIDおよびNCの状態を更新し，TCMTとTPDUをクラス・プロセスへ伝達する．

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，クラス1および3のプロセスでは，状態管理，TWRタイマの起動，そのタイムアウトの処理等を行う．すなわち，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．

#### 4.2.4 スプリッティング

スプリッティングを行うためにNCを複数確立する契機は，最初のNC上でTCが確立し必要なQOSが設定された直後の他に，NCの切断が通知された場合，再送のためのタイムアウトが生じた場合（T1タイムアウト）などである．このようにNCを確立する契機はクラス4の状態管理と関係が深いために，クラス4プロセスで管理する必要がある．一方，NCはTマネジャが一元的に管理しているため，NC確立に必要なNS CIDはTマネジャが割当てクラス4プロセスに通知する必要がある．通知されたNS CIDはクラス4プロセスで保持される．

そこでTマネジャ/クラス4プロセス間にNS CID要求，NS CID割当ての2つの制御情報を新たに用意した．クラス4プロセスはNCを確立する必要が生ずるとTマネジャにNS CID要求を送る．Tマネジャはそれに対し空きNS CIDを選択しNS CID割当てによりクラス4プロセスに伝える．クラス4プロセスはこのNS CIDを用いてコネクション確立要求(NCONreq)を発行する．

本TPプログラムにおいては，割当てられたNCを順に用いてTPDUを転送し，またT1タイムアウトが生じた場合は，1本NCを切断し新たに確立しなおすようにしている．

#### 4.2.5 チェックサム

送信するTPDUへのチェックサムの設定はTPDUを作成する時点に行うため，クラス4プロセスで実行するのが適当である．しかし受信したTPDUのチェックサムの検査は，そのTPDUをTCに関連付けるまえに行う必要があり，Tマネジャで実行される．特に本システムのように，クラス4を含む複数の異なったクラスのTCが存在するシステムでは，本来クラス4のTCに関連付けるべきTPDUが，伝送誤りにより偶然自局のリファレンスの部分に誤りが生じ，これがもとで他のクラスのTCに関連付けられないよう

処理する必要がある．このため，NC が確立れさた直後に受信した TPDU に対しては，クラスに係わりなく，チェックサムパラメータの有無を調べ，有った場合はその検査を行う必要がある．

#### 4.2.6 クラス4で用いられるタイマ

本プログラムでは，TP クラス4で定義しているタイマのうち，以下のものを実装している．

- 再送用タイマ T1
  タイマは個々の TPDU ごとには用意せず，TC に1つ用意する．ただしデータ転送フェーズにおいては DT, ED, AK 用の3つが用意される．
- AK の再送用タイマ W
- インアクティビティタイマ I
- リファレンスの凍結用タイマ L

プログラムは，これらのタイマ値を自由に設定できるように作成されているが，現在 T1, W, I の3つのタイマには T1<W<I の関係を持たせ，それぞれ60秒，180秒，1080秒の値を採用している．また L の値は120秒を用いている．

#### 4.2.7 状態遷移表

TP の標準ではクラス0と2，クラス1と3で状態遷移表を共有しており，データ転送状態の動作等が明確に書かれていない．

本実装では，TP の標準との対応付けを考慮し，状態遷移表の状態としては基本的には TP の標準の定めるもののみを定義した．さらに，データ転送中の個々の状態を識別するために，以下のようなフラグを用意している．

- ED TPDU を送信したか否か：クラス1から4
- タイマが動作中か否か
- 再同期の契機がリセットか切断か：クラス1と3
- ウィンドウ・エッジ：クラス1から4

状態遷移表は，タイムアウト以外については，状態変数と入力イベントを用いた二重の switch 文で実装し，タイムアウト入力についてはタイマ割り込みルーチン（VAX/VMS による AST ルーチンを使用）により実装した．

## 5. SP の実装

### 5.1 概　　要

SP の実装では，3章で述べたように，すべての SP の機能を1つのプロセスで実現し，複数の上位プログラムとインタフェースできるような構成になっている．現在は最大8個の上位プログラムとインタフェースするためのメールボックスを有している．

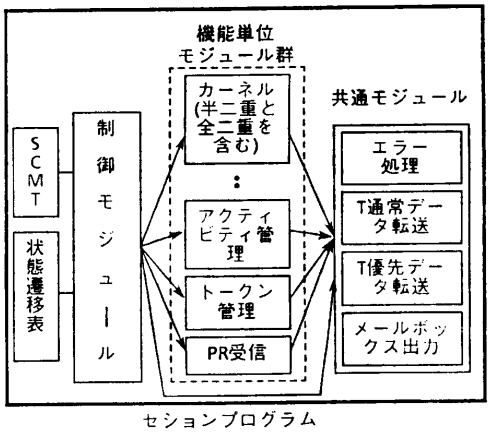


図5 セションプログラムのモジュール構成
Fig. 5 Software configuration of Session program.

また，プログラムの移植において必要な機能単位をくくりだすという方針に基づいて，機能単位に準拠したモジュール化を行っている．作成したセションプログラムでは，図5に示すように，制御モジュール，機能単位モジュール群，共通モジュールという3階層のモジュール構造をとっている．各モジュールの機能概要は以下のとおりである．

① 制御モジュールは，メインプログラムを含むモジュールであり，

(i) 上位またはTマネジャからのプリミティブの読みこみ，解析，
(ii) 状態遷移表の管理，
(iii) 機能単位モジュールまたは共通モジュールのコール，
(iv) 各 SC への SCMT (Session Connection Management Table) の割り当てと，SCMT を用いた複数 SC の管理，

を行う．

② 機能単位モジュールは，各機能単位に関連するサービスプリミティブと SPDU に対する処理を実行するもので，原則的に機能単位ごとに作られている．これらのうち，カーネルに半二重と全二重の機能を含ませている．また，トークンは複数の機能単位で共通に使用されるため，新たにトークン管理モジュールを設け，S-TOKEN-GIVE, S-TOKEN-PLEASE プリミティブと GT, PT-SPDU の処理を行う．PR-SPDU はトランスポート層の優先データ転送サービスが使用可能な場合に，小同期，大同期，再同期，アクティビティ管理で共通に用いられ，PR-SPDU を受信した時

点ではそれがどの機能単位で用いられたものかが分からない．したがって PR-SPDU の受信処理を一元的に行うための機能単位モジュールを設けた．

③ 共通モジュールは，制御モジュール，各機能単位モジュールが共通に使用する処理の集まりで，プロトコルエラーの処理，上位からのSプリミティブのエラー（ユーザエラー）の処理，SPDU の転送を行うために必要なトランスポートのサービスプリミティブを作成しメールボックスに出力する処理等を行う．

### 5.2 SP 実装のキーポイント

#### 5.2.1 SC と TC の管理

SL では TL と異なり，SC と TC は1対1に対応するため，セションプロトコルでは SCMT という1つの表を用いて，SC とそれに対応する TC に関する情報を一元的に管理する．SCMT には以下のような各 SC に関連するすべての情報が蓄えられる．

- SC 識別子，サービス品質 (QOS) のうちの Extended Control と Optimized Dialogue Transfer パラメータ，その SC で使用する機能単位，トークンの配置，発着の SSAP アドレス等，SC に関して上位レーヤから指示される情報
- SC の状態と，設定した同期点の順序番号に関する変数 V(M)/V(A)/V(R)，アクティビティの中か否かを示す変数 Vact 等，SC の状態を特徴付けるために，SP の標準で定義されている変数
- SC に対応する TC の発着の TSAP アドレスと，TP の優先データが使用可能かを示す変数 TEXP，TC レスポンダ側か否かを示す変数 Vtca 等の TC を特徴付けるために標準で定義されている変数
- SC の SS CID と使用するメールボックスのチャネル番号，対応する TC の TS CID 等の上位および下位レーヤとの対応をとるための情報
- タイマ TIM が動作中か否かのフラグ，入力されたプリミティブを保留するためのバッファエリアへのポインタ，SCONreq のユーザデータを保留するエリアへのポインタ，DT/TD SPDU のリアセンブリング用のバッファエリアへのポインタ等のためにセションプログラムが使用する作業領域

#### 5.2.2 状態遷移表の実現方法

SP では TP と異なり，状態遷移表の規模が大きく（状態数 29 入力数 77），また同一の状態と入力の組に対しても条件により異なる動作が定義されており，その構造も複雑になっている．このため，遷移表の保守や変更，必要な機能単位のくくりだしが容易なように，状態遷移表をいかに実現するかが問題となる．作成したセションプログラムでは，機能単位に従ったプログラムのモジュール化に応じるよう，次のような構成にした．

① 遷移表は制御モジュールが一元的に保持し，遷移表の参照も制御モジュールが行う．

② 遷移表の保守や変更を容易にするためには，遷移表に記述される内容を少なくする必要がある．そこで，遷移表の各エントリには，受信したプリミティブや SPDU を処理すべき機能単位または共通モジュール名と，各モジュールに処理の内容を指示する処理コードだけが記入されている．

制御モジュールは，受信したセションまたはトランスポートのサービスプリミティブを解析し，状態遷移表を参照する．その後，遷移表の該当するエントリに示されたモジュールに受信したプリミティブ/SPDU と処理コードを渡す．一方，機能単位または共通モジュールは受けとった処理コードが示す条件判断，状態の遷移，プリミティブや SPDU の出力，順序番号等に対応する変数の更新を行う．大同期を挿入するための MAP-SPDU を受信した場合を例にとり，その処理の流れを図 6 に示す．

このような構成の遷移表を，C言語によるコーディングにおいて，状態変数と入力イベントを用いた二重の switch 文で実装した．

#### 5.2.3 必要な機能単位のくくり出し法

作成した SP プログラムではすべての機能単位を実装しているが，上位プログラムの必要とする機能単位のみを実装するような SP プログラムを実現するためには，以下のようにして必要な機能単位のみをくくり出すことができる．

① 状態遷移表は一切変更を加えない．

② 使用しない機能単位に対応するモジュールを，以下のように変更する．すなわち，セションプリミティブの受信に対応する制御コードに対しては，一律にユーザエラーを処理する関数を呼び出し，また SPDU の受信に対応する制御コードに対しては，一律にプロトコルエラーを処理する関数 P-ERROR を呼び出すようにする．

この方法では，状態遷移表についてはさらにくくり出しを進める余地が残されているが，本 SP プログラムでは 5.2.2 で述べたように，状態遷移表は制御コードを用いてコンパクトに実現しているため，状態遷移表のくくり出しを行ってもプログラム規模がそれほど

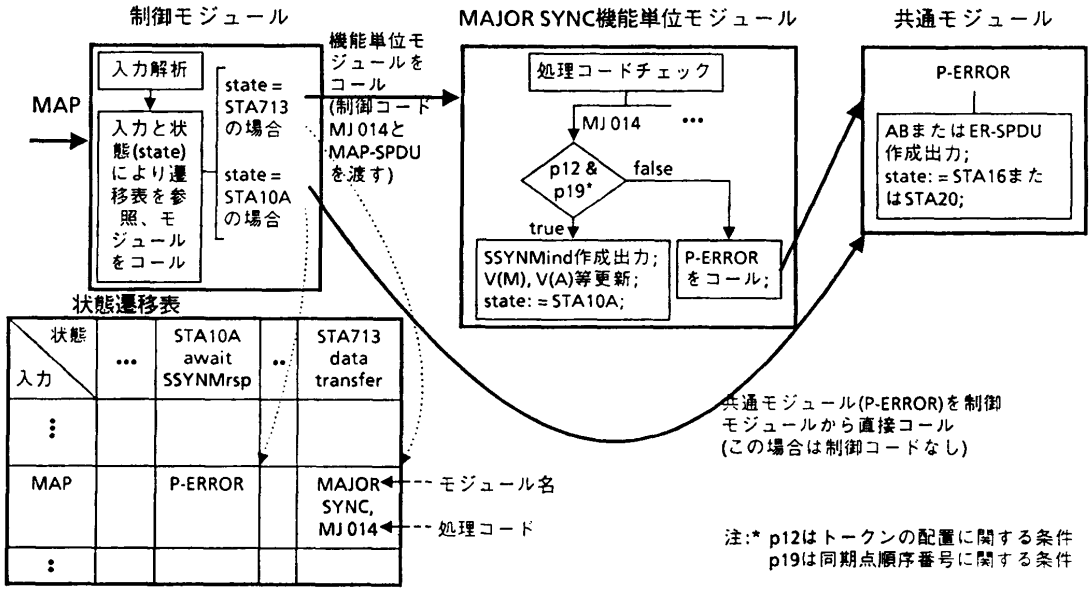


**図 6**　MAP-SPDU を受信した場合の処理の流れ
Fig. 6　Processing flow at the reception of MAP-SPDU.

小さくならないと考えられる.

### 5.2.4　T. 62 のサポート方法

テレテックス用の勧告 T. 62 に従った通信を実現するためには，T. 62 固有のパラメータの取り扱いを可能とする必要がある．本 SP プログラムでは，T. 62 固有のパラメータの処理を行うプログラムは SP プログラムの上位プログラムとして実現するという方針により，以下のような内部処理の変更で，容易に T. 62 をサポートすることができる.

① 上位レーヤが T. 62 特有のパラメータを送信する場合はサービスプリミティブのユーザデータとしてこれらのパラメータをセションプログラムに渡す．セションプログラムではこれらのパラメータを，T. 62 の定めるフォーマットに従って SPDU に展開する.

② 逆に，T. 62 特有のパラメータを持つ SPDU を受信した場合は，対応するサービスプリミティブのユーザデータに入れる.

## 6. 結果と考察

作成した TP ならびに SP プログラムは，簡易的なファイル転送が可能な上位プログラム，MHS や FTAM 等の OSI プログラム[7),8)] と組み合わせて，X. 25 パケット網を介して正常に動作している．以下に本実装の結果と考察を述べる.

（1） 実装した TP プログラムは，TP の全仕様のうち，TPDU の送信時のコンカティネーション，クラス 4 においてスプリッティングとマルチプレクシングを同時に実行すること，クラス 3 と 4 における転送ウィンドウのリダクション，クラス 4 のウィンドウリダクションに伴う AK TPDU の再送手順以外の機能をすべて実現している．また SP プログラムは，拡張コンカティネーション以外の SP の機能を実現している．未実装の機能のうち，コンカティネーションとウィンドウのリダクションについては，実行する契機等について検討を進める必要がある.

（2） 表 2 に T マネジャ，各クラス，セションのソース・プログラムの実効行数を示す.

TP プログラムの実装は，クラス 0 から 3 までの作成とクラス 4 の追加という 2 つのフェーズに分けて行

**表 2**　ソースプログラムの実効行数
Table 2　Effective size of source program.

| プログラム | | 実効行数（K行） |
|---|---|---|
| トランスポート | Tマネジャ | 3.1 |
| | クラス 0 | 4.1 (2.3) |
| | クラス 1 | 9.3 (4.7) |
| | クラス 2 | 6.7 (3.4) |
| | クラス 3 | 10.1 (6.8) |
| | クラス 4 | 9.4 |
| セション・プログラム | | 19.8 |

注：クラス 0 から 3 については，他のクラスのプログラムを再利用した部分を除いた結果を括弧の中に示す.

ったが，クラス4の追加に伴いクラス0から3のプログラムの実装方法を変更した．このため先に報告したクラス0から3までの実効行数[3]とは異なった結果となっている．

新しい実装方法では，プロトコル機構のうちで異なるクラス間でも機能が類似しているものについては，番号の大きいクラスで実装したプログラムを番号の小さいクラスで再利用させた．表2では再利用したプログラムを各クラスで重複して計算した実効行数と，重複させない場合の実効行数を両方示している．

表2に示した数値は実装の仕方で変動するが，TPとSPのプログラム規模を把握でき，またTPとSPおよびTPの各クラス間のプログラム規模の傾向を知ることができる．クラス1および3の実効行数がクラス4より大きいのは，クラス1および3の方が状態数が多いこと，クラス4ではウィンドウのリダクションに伴うAK TPDUの再送手順を実装していないこと等のためであると考えられるが，クラス4は他クラスに比べてそれほど複雑なクラスではないといえよう．

（3） TPおよびSPプログラムをVAXが標準的に提供するX.25プログラムPSI上で動作させ，X.25網（9.6 Kbpsの回線速度で128オクテットの最大パケット長）を介した通信によりTP/SPプログラムの実用性を確かめた．

●SPプログラム上に簡便なファイル転送プログラムを作成し，100 Kバイトのデータ・ファイルを転送した場合に，VAXの機種により若干異なるものの，9.6 Kbpsの回線速度で5.2 Kから5.5 Kbpsの回線効率を得た．ここでいう回線効率は，簡便なファイル転送プログラムのユーザ・データのみを対象にしたもので，ファイル転送やセション層以下のプロトコル・ヘッダのビット数を計算に加えていない．回線効率の測定は，各層ともデータ転送フェーズにあり，転送エラーのない状況で測定しているためTPクラス0，1，2，3では差が生じないが，データ転送の機構が異なるクラス4では，他のクラスに比して若干低下している．プログラム構成上では，図2のTマネジャとクラス・プロセスとのインタフェースに負荷がかかっており，さらに高速化が必要なユーザに対しては，この部分の改良が必要となる．

（4） TP全クラスの実装とSP全仕様の実装を比較すると，以下に示すような理由でTPの方が困難であった．

●TP標準ではTPの機能をプロトコル機構ごとにまとめて記述しているため，プログラム設計時にプロトコル機構の記述から各クラスのプログラムの機能を決定する作業が必要である．

●SP標準は各入力に対する処理という記述方法を用いているため，SP標準がプログラム設計に向いていると考えられる．

●TP標準の状態遷移表が不完全であるのに対し，SP標準では状態遷移表が完備されており，さらに遷移表にはシリアル番号の管理等に関連する条件やアクションも明確に規定されている．したがってSPのプログラムの状態遷移表は標準の状態遷移表を忠実に実現すればよいが，TPでは独自に状態遷移表を設計する必要がある．

●TPではクラスのネゴシエーションにおいて，クラス・プロセス間のパラメータの受渡しを実現する必要がある，クラス4から他のクラスへのネゴシエーション・ダウンでは他のクラスでCR TPDUの再送を考慮しなければならない，クラス1や3からのネゴシエーション・ダウンでは再同期や障害後の再割付けを考慮する必要がある等，その処理が複雑である．

●SPではSCとTCの対応付けが一対一であり，TCが切断された場合はSCも解放されるのに対し，TPでは，マルチプレクシング，スプリッティング，障害後の再割付け等の機能によりTCとNCの対応付けが一対一ではない．そのためTPではコネクションの管理に工夫を要する．

●SPは機能が豊富であるが，機能単位として整理されており，異なった機能単位の間の関連は少ない．それに対しTPでは4章に述べたようにプロトコル機構の間の関連が深く，実装が困難である．

（5） 今回のTP/SPの実装はVAX/VMS下で行っているが，VAX/VMS以外の実装環境への適用に関して，OS機能への依存性，コンピュータによるバイト順序の違い，Cコンパイラの違いの3点で以下の結果を得ている[9]．

●本プログラムはVMS固有の機能は使用しておらず，プロセス間通信を実現するためのメールボックスの生成/入出力と，タイムアウトの通知に関するシステムサービスのみを用いている．通常のマルチプロセスOSには，これらのシステムサービスに対応する機能があり，移植しやすい．また，本プログラムをUNIXライクのOSに移植した結果では，メールボックスの代わりにパイプを用いたプロセス間通信を実現させ，容易に移植できた．

●コンピュータによるバイト順序の違いというのは，例えば，4バイトの整数のように，複数バイトをまとめて1つの情報単位とする場合に，どのバイトを最上位とするかがコンピュータにより異なるということである．これは通信プログラムのPDUの作成や解析において常に問題となる点であるが，本プログラムにおいても，バイト順序の異なるコンピュータに移植するためにはPDUを扱う部分の変更が必要である．

●Cコンパイラの違いに関して，特に簡易なコンパイラでは，変数名の最大長が8バイトに制限されたり，構造体の要素名と変数名が重複してはならない，さらには，構造体の要素の境界が2バイトごとになるなどの制限がつく場合がある．このようなコンパイラを使用した場合の移植には，工数的な負荷がかかる．

上記3点の考察結果から，本プログラムは基本的にはVAX/VMSという開発環境に依存する部分は少なく，他へ移植する場合も少量のOS依存の部分とバイト順序の違いの部分について変更するのみで良いことがわかる．ただし，移植先のCコンパイラの性質によっては，移植工数が極端に増加する場合がある．

（6） 最近，国内外のコンピュータ・メーカ等がOSI製品の開発，販売計画を発表し，いくつかの製品が登場してきている[10]．本プログラムもその一翼を担うものであるが，他と比較して，トランスポート・レーヤではすべてのプロトコル・クラスを，セション・レーヤではすべての機能単位を実装していることに特徴がある．また最近ISOでは，各種の業界向けの通信システムがOSIプロトコルを採用する際に，相互接続性を保証し，かつまた実装プロトコルの機能を最小限にするため，機能標準の作成を進めている．今回作成したTP/SPプログラムは，OSI標準の全仕様を実装している上，トランスポート・レーヤではクラスを，また，セション・レーヤでは機能単位を選択して使用することにより，どのような機能標準をもつ通信システムにも対応できる特徴がある．

## 7. おわりに

本稿では筆者らが作成したOSIトランスポートならびにセション・プロトコル・プログラムの概要について述べた．現在これらのプログラムにインタフェースするOSI上位プログラムを開発しており，またこれらのプログラムのパーソナル・コンピュータ上への移植も行っている．今後ともOSIプロトコルの実装，処理効率の向上を行う予定である．

**謝辞** 日頃御指導いただくKDD上福岡研究所小野所長に感謝します．

## 参考文献

1) CCITT, Rec. X. 214, X. 224, Oct. 1984, ISO, IS 8072, IS 8073.
2) CCITT, Rec. X. 215, X. 225, Oct. 1984, ISO, IS 8326, IS 8327.
3) 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と製品検証，情報処理学会分散処理システム研究会，22-9 (May 1984).
4) 鈴木，加藤：OSIセッションレーヤ標準のインプリメント，情報処理学会分散処理システム研究会，24-4 (Nov. 1984).
5) 加藤，小花，鈴木：OSIトランスポート・プロトコルにおけるクラス4とNCMSの実装について，情報処理学会マルチメディアと分散処理研究会，26-1 (July 1985).
6) 鈴木，加藤：OSI準拠高度パソコン通信システム，第33回情報処理学会全国大会論文集，2U-6 (Oct. 1986).
7) 加藤，鈴木：センタ型メイルシステムELMSとMHSとの相互接続，情報処理学会マルチメディアと分散処理研究会，33-7 (May 1987).
8) 小花，加藤，鈴木：OSI FTAM, ACSE, プレゼンテーション・プロトコルの実装，情報処理学会マルチメディアと分散処理研究会，33-6 (May 1987).
9) 加藤，鈴木：パソコンへのOSIトランスポート及びセションプロトコルの移植，第32回情報処理学会全国大会論文集，6D-7 (Mar. 1986).
10) 星野：火が入るOSI，日経コミュニケーション，1988年4月18日号.

（昭和63年3月31日受付）
（昭和63年10月7日採録）

**鈴木 健二**（正会員）

昭和20年生．昭和44年早稲田大学理工学部電気通信学科卒業．昭和44年から45年までオランダのフィリップス国際工科大学に招待留学，昭和51年早稲田大学大学院博士課程修了．工学博士．同年国際電信電話(株)入社．現在，同社上福岡研究所コンピュータ通信研究室長．この間，磁気記録，パケット交換方式，ネットワーク・アーキテクチャ，通信処理などの研究に従事．昭和62年度，前島賞受賞．電子情報通信学会，IEEE各会員．

**加藤　聰彦**（正会員）

昭和31年生．昭和53年東京大学工学部電気工学科卒業．昭和58年同大学院博士課程修了．工学博士．同年国際電信電話(株)入社．現在，同社上福岡研究所コンピュータ通信研究室主査．昭和62年から63年まで米国カーネギーメロン大学計算機科学科客員研究科者者．この間，OSIプロトコルの実装やコンフォーマンス試験，通信プロトコルの形式記述技法，分散オペレーティングシステムなどの研究に従事．昭和60年情報処理学会学術奨励賞受賞．電子情報通信学会会員．

**浦野　義頼**（正会員）

昭和17年生．昭和40年早稲田大学理工学部電気通信科卒業．昭和45年同大学院博士課程修了．工学博士．同年国際電信電話(株)入社．現在同社上福岡研究所次長．この間フェイル・セイフ論理システム，パケット交換方式，ネットワーク・アーキテクチャ，ビデオテックス，ソフトウェア信頼性などの研究・開発に従事．電子情報通信学会会員．