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OSI7層ボードの実装と評価

井戸上 彰† 加藤聰彦† 鈴木健二†

OSIに基づく各種情報通信システムの構築において，パソコンやワークステーションを利用することが不可欠となってきている．このため筆者らは，これらの計算機の拡張ボードとして，7層すべてのOSIプロトコルをサポートするOSI 7層ボードの開発を行った．OSI 7層ボードは，さまざまなネットワークに柔軟に対応し，応用層までの各種プロトコルを高いスループットで実行する．ボードのハードウェアは，対象とするネットワークに依存した機能を持つ回線制御モジュールと，ネットワークとは独立に高速なプロトコル処理を実現するベースモジュールから構成される．ベースモジュールでは，OSI 7層ボード用のオペレーティング・システムを搭載し，タスクごとの仮想アドレス空間とメモリ保護機能によって各層ごとに独立したプログラム開発を可能とする．また，OSが提供する共有メモリを使用した高速なタスク間通信機能や，コピーを伴わないPDU作成/解析処理などによって効率的なプロトコル処理を実現している．本ボードを用いることにより，パソコンやワークステーションをベースとした各種OSI通信システムの構築を，簡便かつ低コストに実現することが可能となる．

Implementation and Evaluation of OSI 7 Layer Board for Personal Computers and Workstations

AKIRA IDOUE,† TOSHIHIKO KATO † and KENJI SUZUKI †

Recently, it has been required to use personal computers and workstations as components of various OSI communication systems. For this reason, we have developed the OSI 7 layer board, which supports OSI protocols from physical through application layers, for these kinds of computers. This board supports a variety of networks and executes OSI protocols with high throughput. The hardware of the board is composed of circuit-control module and base module. The circuit-control module supports the functionalities that depend on the target network, while the base module realizes high speed protocol processing with a 32 bit CPU. On the base module, an operating system for the OSI 7 layer board is installed in order to facilitate layer-by-layer development of protocol programs and to execute these programs efficiently. Protocol programs on the board achieve high performance by avoiding data copying in PDU composition/decomposition. By making use of the OSI 7 layer board, it becomes possible to construct personal computer or workstation based OSI communication systems concisely and inexpensively.

1. はじめに

近年，さまざまな情報通信システムにおいて，メーカや機種の異なる計算機間の相互接続が必須のものとなってきており，開放型システム間相互接続(OSI：Open Systems Interconnection)プロトコルの採用が進みつつある．さらに，このような情報通信システムの構築においては，求められる機能をいかに迅速・低コストで実現するかが重要な課題であり，このため高性能化・低価格化が著しいパーソナルコンピュータ(以下パソコンと略す)やワークステーションをその構成要素として利用することが不可欠となっている．

このような背景から，筆者らは，パソコンやワークステーションの計算機本体(以下ホストと呼ぶ)に挿入される拡張ボードとして，7層すべてのOSIプロトコルをサポートする「OSI 7層ボード」の開発を行った[1),2)]．OSI通信機能の実装においては，各種の計算機上に容易に移植でき，保守性にすぐれること，回線速度に見合った実用的な性能が得られること，さまざまな通信システムを構築する上で汎用的かつ簡便に利用できること，などが要求される．OSI 7層ボードでは，プロトコル処理をボードで行わせることにより，MS-DOSパソコンなどのメモリ空間に制限のある計算機においても，OSI通信機能を容易に実現できる．また，

† 国際電信電話株式会社 研究所
KDD R & D Laboratories

他機種のパソコンやワークステーションに移植する場合でも同一のプロトコル処理プログラムが使用できるため，複数の異なる機種の計算機に対するOSI通信機能の開発・保守が容易となる．開発したボードは，さまざまなネットワークに柔軟に対応可能なハードウェア構成を持ち，通信ボード用の独自のオペレーティング・システム(OS)の搭載や，データ・コピーを伴わないプロトコル処理の実現などにより，応用層までの各種プロトコルを高いスループットで実行する．OSI 7層ボードを用いることにより，各種端末，サーバ，ゲートウェイなど，業務に応じたさまざまな情報通信システムの構築を，パソコンやワークステーションをベースとして，簡便かつ低コストで実現することが可能となる．

本稿では，OSI 7層ボードの実装と性能評価について述べる．以下，2章ではOSI 7層ボードの実装における基本方針を述べる．続いて，3章および4章で，OSI 7層ボードのハードウェアおよびソフトウェアの概要を述べる．また5章では，ボードを収容するホスト上のソフトウェアについて述べる．さらに6章では，FTAM(File Transfer, Acccess and Management)プロトコルを例として，OSI 7層ボードの実装結果と性能評価を示し，7章で実装および性能評価に関して考察する．

2. 基本方針

OSI 7層ボードのハードウェア/ソフトウェアの実装においては，以下の基本方針を採用した．

① パソコンやワークステーションをベースとした各種OSI通信システムの構築を容易化すること，プロトコル処理プログラムの移植性・保守性を向上させること，ホスト計算機にプロトコル処理の負荷をかけないことなどを目的として，ボード上でFTAMやMHS(Message Handling System)などの特定応用層までを含む7層すべてのプロトコル処理をサポートする．また，要求に応じて，さまざまなプロトコルを同一のハードウェア上で実行可能とする．

② 最小限のハードウェア変更で各種のパソコンやワークステーションに対応可能とするとともに，パケット網，電話網，ISDN，LANなどのさまざまなネットワークに柔軟に対応できるハードウェア構成を採用する．

③ 伝送路の高速化を考慮して，プロトコル処理のオーバヘッドをできるだけ削減し，ホスト上のアプリケーション・プログラムまでを含めて数Mbps程度のスループットを実現する．

④ ボード上のプロトコル処理プログラムの開発・保守を容易化するため，各層および応用サービス要素(ASE：Application Service Element)ごとの独立なプログラム開発をサポートする．

⑤ ホスト上のアプリケーション・プログラムの作成を簡便にするため，ボードとホストの間で各種プロトコルに対応するサービス・プリミティブを送受信するための汎用的なアプリケーション・プログラミング・インタフェース(API)を提供する．

3. OSI 7層ボードのハードウェア

図1にOSI 7層ボードのハードウェア構成を示す．パケット網，電話網，ISDN，LANなどの各種のネットワークに対応するため，ネットワークに依存したハードウェアを実現する「回線制御モジュール」と，ネットワークとは独立に応用層までのプロトコルの高速処理を実現する「ベースモジュール」から構成することとした．回線制御モジュールは対象とするネットワークごとに，ベースモジュールは対象とするホストごとに独立に開発される．回線制御モジュールは，接続コネクタを介してベースモジュールに付加される構造となっており，両者を組み合せることにより，各種のホストやネットワークに幅広く対応できる．

(1) ベースモジュール

ベースモジュールは，応用層までの大規模なプロトコル処理プログラムを高速に処理するために，CPUとしてインテル80386 SX(32ビット，クロック周波数16 MHz)を搭載し，8 Mバイトの RAM と 512 K バイトの ROM を実装する．ボード上のプログラムは，ホストからのダウンロードまたは ROM からの読み出しにより，RAM 上に展開されて実行される．

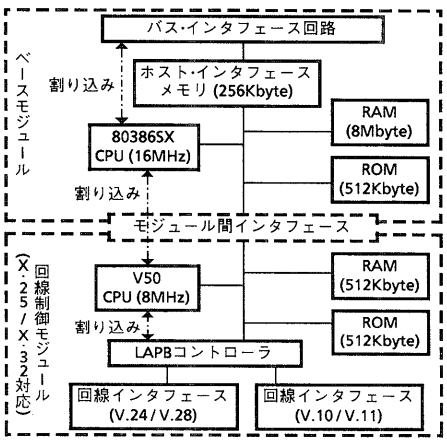


図1 OSI 7層ボードのハードウェア構成
Fig. 1 Hardware configuration of OSI 7 layer board.

ベースモジュールは，ボードとホストの間のデータ転送のために，256 Kバイトのホスト・インタフェース・メモリ(デュアルポート・メモリ)と，対象となるホストに対応したバス・インタフェース回路を持つ．ボードおよびホストのCPUは，転送するデータをホスト・インタフェース・メモリに設定し，割り込みによって互いにデータ転送要求を通知する．ボード内部のローカルなRAMとは独立に，インタフェース専用のメモリを設けることにより，ホストCPUとボードCPUのアクセス競合を軽減している．

デュアルポート・メモリを使用したホスト・ボード間のインタフェース方式はすべての機種に対して共通であり，バス・インタフェース回路の変更のみでさまざまなホストに対応可能である．

(2) 回線制御モジュール

図1に，パケット網と電話網(X.25/X.32)に対応する回線制御モジュールの構成を示す．本モジュールは，NEC V50 CPU(16ビット，クロック周波数8 MHz)およびLAPBコントローラを搭載し，それぞれ512 Kバイトの RAMとROMを実装する．また，V.24/V.28(RS-232C)とV.10/V.11の回線インタフェースをサポートする．搭載するLAPBコントローラの性能上，最大2 Mbpsの回線速度まで対応可能である．なお，本モジュールは，先に開発済のOSI 5層ボードのハードウェアとほぼ同等のものである[3]．

(3) モジュール間のインタフェース

ベースモジュールと回線制御モジュールの間のデータのやり取りは，ベースモジュールのCPUが回線制御モジュールのRAM領域に直接アクセスすることにより実現している．データ転送要求は，ホストとのインタフェースの場合と同様，互いのCPUへの割り込みによって通知する．

4. OSI 7層ボードのソフトウェア

OSI 7層ボードのソフトウェア開発においては，プロトコル処理の大部分を32ビットCPUを搭載したベースモジュール上に実現することとした．このため本章では，ベースモジュール上のソフトウェアの概要について示す．

4.1 実装方針

プログラム開発を容易化し，高速なプロトコル処理を実現するために，ベースモジュール上のソフトウェアに対して以下の実装方針を採用した．

① 各層/ASEのプログラムの独立開発と効率的な実行をサポートするため，OSI 7層ボード用のOSを導入し[4]，OSの機能を用いて，プロトコル処理やホスト等との入出力処理を実現するプログラムを実装する．OSは，仮想アドレス空間とメモリ保護を提供するタスク管理，共有メモリ空間を用いる高速なタスク間通信，オーバヘッドの少ないスケジューリングなどの機能を実現する．

② プロトコル処理を行うプログラムの開発・保守を容易にするため，各層およびASEを独立したプログラム・モジュールとして作成し，それぞれOSI 7層ボード用OSのもとで独自のアドレス空間を持つタスクとして動作させる．

③ 各層/ASEの間でやり取りされるサービス・プリミティブは，多数のパラメータを持ち，その多くが可変長やオプションとなっている．このようなデータをタスク間で効率的に転送するために，プリミティブとして，構造体，共用体，ポインタなどにより構造化されたデータ型を採用する[5]．

④ 高いスループットを実現するため，各層/ASE内部におけるプロトコル・データ単位(PDU)の作成/解析処理において，ユーザデータのコピーを伴わないバッファ制御方式を採用する[6]．

⑤ ホストおよび回線制御モジュールとの入出力処理に関しては，対象とするホストまたは回線制御モジュールに依存する機能や，特定のメモリ空間やI/Oポートへのアクセスなどが要求されるため，プロトコル処理を行うタスクとは独立に，入出力専用のタスクを設ける．

以上のような方針に基づき，パケット網/電話網対応の回線制御モジュールを持つOSI 7層ボードにおいて，FTAMを含むOSIプロトコルを実装した場合のベースモジュールのソフトウェア構成を**図2**に示す．この例では，トランスポート層以上をベースモジュール上に搭載している．

4.2 OSI 7層ボード用OS

OSI 7層ボード用OSは，プロトコル処理プログラムの開発と実行のために最適化された機能のみをサポートし，OS自身のオーバヘッドを最小限としている．本OSの代表的機能を以下に示す[4]．

(1) タスク管理とスケジューリング

● それぞれ独自の仮想アドレス空間を持つ複数のタスクの実行・管理機能を提供する．

● 要求に応じて各種のプロトコルを実行可能とするため，初期化時にプロトコル処理プログラムをホストからダウンロードする機能を提供する．

● タスク切替えのオーバヘッドを最小限とするために，スケジューリングはキューを介したプリミティブの送受信時を契機とする．通常は，各タスクがプリミ

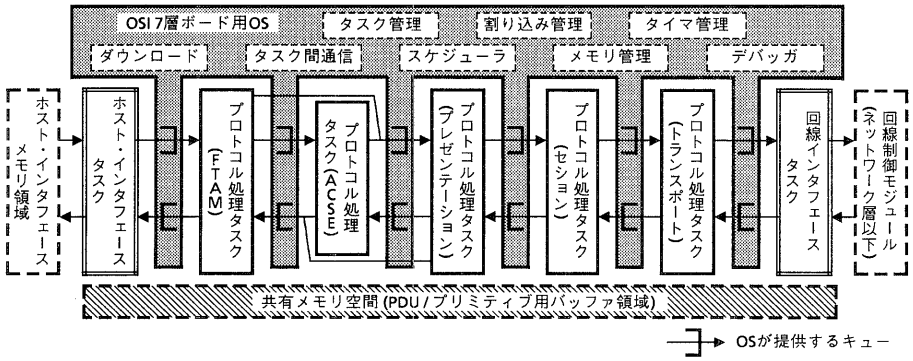


図2 OSI 7層ボード(ベースモジュール)のソフトウェア構成
Fig. 2 Software configuration of OSI 7 layer board (base module).

ティブ受信待ちとなった時点でタスクを切り替え，優先データ転送要求などの優先的に処理する必要のあるプリミティブについては，その送信時点で受信タスクに制御を移すことを可能とする．

（2） メモリ管理

● 1つのタスクのアドレス空間におけるコード，データ，スタックの各領域を別個のセグメントで実現することにより，セグメント単位の仮想アドレス空間とメモリ保護を提供する．

● ポインタなどを含む複雑な構造のデータを，すべてのタスクから同一の仮想アドレスでアクセス可能な共有メモリ空間を提供する．

● バッファ領域の輻輳を避けるため，バッファの使用率が一定以上になると，ホスト・インタフェース・タスクなどに対して，ボードへのデータ入力を制限させる機能を提供する．

（3） タスク間通信

● 高速なタスク間通信を実現するために，コピーを伴わずに共有メモリ空間上のデータへのポインタのみを転送するキューを提供する．

（4） 割り込み/タイマ管理

● ホストや回線制御モジュールからの割り込みによってそれぞれの入出力インタフェース用のタスクを起動する機能や，ハードウェア・タイマからの割り込みによってカウントされるタイムアウト通知機能を実現する．

（5） デバック機能

● 各層/ASEのタスクからのデバッグメッセージの出力機能や，メモリダンプ，レジスタダンプなどの機能を持つデバッガを実装する．

4.3 プロトコル処理タスク

以下に，各層/ASEごとに独立に開発されるプロトコル処理タスクの基本的な処理の流れと，効率的なプロトコル処理を実現するために導入したプリミティブのデータ構造およびPDUの作成/解析処理の概要について述べる．

（1） 基本的な処理の流れ

各プロトコル処理タスクは，コネクションまたはアソシエーションの管理，上位層/下位層からのプリミティブやPDUの解析処理，状態遷移処理，PDUやプリミティブの作成処理などから構成され，次のような処理手順に従う．

各タスクは，最初に起動された際に，隣接する層/ASEとの間のキューの生成などの初期化処理を行った後，キューに対するプリミティブの到着を待つ．キューにプリミティブが到着し，タスクの実行が再開されると，そのタスクは，上位層および下位層からのキューからプリミティブを受け取り，必要な状態遷移処理やプリミティブ送信処理を行う．タスクはキューに到着しているすべてのプリミティブを処理した後，新たなプリミティブの到着を待つ．

OSI 7層ボード用OSは，プリミティブの送信先のタスクを実行可能状態とし，実行中のタスクがプリミティブ受信待ちとなった時点で他の実行可能状態のタスクに制御を移す．このため，各タスクが上記のような処理手順に従うことによって，上位層から下位層へ，あるいは下位層から上位層へのプリミティブの流れにしたがって，実行が必要なタスクが順次起動され，OSIプロトコル処理の流れを自然に実現している．

（2） プリミティブのデータ構造

構造化されたプリミティブのデータ型の例として，FTAMが提供するF-INITIALIZE要求およびF-DATA要求のデータ型の一部を図3に示す．

図に示すように，コンテント・タイプ・リストは，

```
typedef struct {  /*** F-INITIALIZE req ***/
   prim_head_t       PrimHead;
   psap_t            CallingPSAP, CalledPSAP;
   short             ServiceClass;
   short             FuncUnits;
   cont_type_list_t  ContList; ...
} f_initialize_req_t;

typedef struct {
   long                    Num;
   pointer_t(cont_type_t)  ContTypes;
} cont_type_list_t;

typedef struct {
   obj_id_t          DocType;
   obj_id_t          AbsSyntx;
} cont_type_t

typedef struct {
   long Num;  long ObjIdComp[OBJ_ID_MAX];
} obj_id_t;

typedef struct {  /*** F-DATA req ***/
   prim_head_t       PrimHead;
   user_data_list_t  DataVal;
} f_data_req_t;

typedef struct {
   long Num; pointer_t(user_data_t) UDList;
} user_data_list_t;

typedef struct {
   long  Len;
   pointer_t(PDU_t) Data;/* データへのポインタ */
} user_data_t;

#define pointer_t(TYPE) struct { \
   TYPE *Ptr; char Pad[2]; }
```

図3 F-INITIALIZE要求およびF-DATA要求プリミティブのデータ型(一部)

Fig. 3 Data structure of F-INITIALIZE request and F-DATA request primitives.

ドキュメント・タイプ名および抽象構文名を示すオブジェクト識別子から成る構造体(cont_type_t)の配列へのポインタと，配列の有効個数を示す整数で表される．また，オブジェクト識別子は，要素となる整数の配列とその有効数からなる構造体(obj_id_t)で表される．コンテント・タイプ・リスト，あるいはF-DATA要求のユーザデータ・リストやユーザデータの型のように，可変長の要素を持つ場合は，ポインタを示すpointer_tマクロにより要素の型の配列を指示する変数と，その個数を示す変数とを使用する．pointer_tマクロは，OSI 7層ボード上で6バイト長のポインタを用いるため，パディングを含めたポインタ・パラメータ領域に8バイトを割り当て，機種によるポインタ変数サイズの違いを吸収するために設けた．

ボード上の各タスクは，OSから割り当てた共有メモリ空間上のバッファにポインタを含む複雑な構造のプリミティブを作成し，キューを介してコピーを伴わずに転送する．

(3) PDUの作成/解析処理

各層/ASEのプロトコル処理プログラムでは，PDUの作成/解析において，ユーザデータのコピーを避けるために，以下のようなバッファ制御方式を採用した[6]．

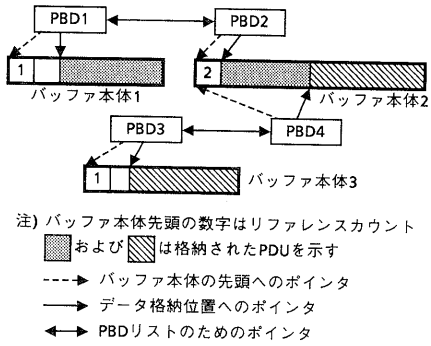


図4 PDUバッファの構成

Fig. 4 Configuration of PDU buffer.

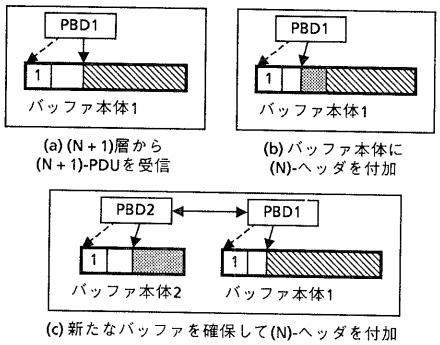


図5 PDUヘッダの付加

Fig. 5 Addition of PDU header.

① PDUバッファの構造

本方式は，共有メモリ空間上において，PDUのヘッダやユーザデータを保持するバッファ本体と，バッファ本体を管理するためのPDUバッファ・ディスクリプタ(PBD)を用いる．PBDは，バッファ本体の先頭アドレスとバッファ本体の長さ，バッファ本体内のデータ格納位置へのポインタとデータの長さ，複数のPBDをリストとして連結するための前後のPBDへのポインタなどから構成される．また，複数のPBDが同一のバッファ本体を参照する場合があるため，バッファ本体上に，対応するPBDの数(リファレンスカウント)を示すフィールドを設ける．本方式に基づくPDUバッファの構成例を図4に示す．図4において，PBD1とPBD2，およびPBD3とPBD4のリストがそれぞれ1つのPDUを示している．

② ヘッダの付加/削除

新たなPDUを作成する場合は，確保したバッファ本体に，下位層のヘッダ用に一定サイズ(現在は64バイトとしている)の空き領域を設けておく(図5(a)参照)．

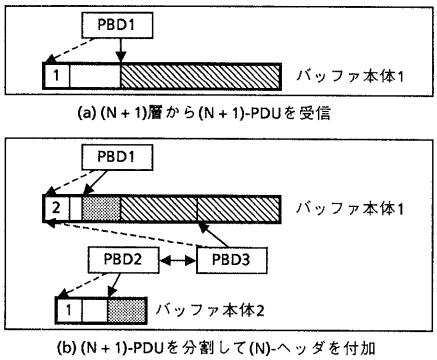


図6 PDUのセグメンティング
Fig. 6 Segmenting of PDU.

(N+1)層から送信を要求された(N+1)-PDUに(N)層のヘッダを付加する場合，(N+1)-PDUを格納するバッファ本体に(N)-ヘッダ長以上の空き領域が存在すれば，その空き領域にヘッダを付加し，PBDのデータ格納位置を調整する(図5(b)参照)．(N+1)-PDUのバッファ本体に十分な空き領域がない場合は，新たに別のPBDとバッファ本体を確保して(N)-ヘッダを作成し，ヘッダ用に確保したPBDと，(N+1)-PDUを表す元のPBDを連結して(N)-PDUを構成する(図5(c)参照)．

受信した(N)-PDUから(N)-ヘッダを削除する場合，(N)-PDUが複数のPBDのリストで構成されている場合は，先頭から順に(N)-ヘッダのみを含むバッファ本体とPBDの組を解放する．(N+1)-PDUを含むバッファ本体に到達すると，そのバッファ本体を参照するPBDのデータ格納位置を(N+1)-PDUの先頭に設定する．

③ セグメンティング/リアセンブリング

(N+1)層から送信を要求されたPDUを(N)層においてセグメンティングする必要がある場合，最初の(N)-PDUに関しては，②で述べたように，バッファ本体に十分な空き領域が存在すれば，元の(N+1)-PDUのバッファ本体上に(N)-ヘッダを付加する．2番目以降の(N)-PDUに関しては，新たにPBDとバッファ本体の組を確保して(N)-ヘッダを作成し，さらに元のバッファ本体上のユーザデータ部分を指し示すPBDを作成して(N)-ヘッダ用のPBDと連結する(**図6**参照)．

受信した(N)-PDUをリアセンブリングする場合は，各(N)-PDUのヘッダを削除した後，すべてのPBDを順に連結して(N+1)-PDUを完成させる．

④ バッファ解放のタイミングと再送制御

送信PDUに関しては，それを作成する層がバッファを確保し，最下位の層で解放処理を行う．受信PDUに関しては，最下位の層がバッファを確保し，該当PDUを処理し終えた最上位の層で解放する．再送などのために送信PDUを保持しておく必要がある場合は，同一のバッファ本体を参照するPBDのリストを新たに作成し，連結された各バッファ本体のリファレンスカウントをインクリメントする(**図7**参照)．PDUバッファを解放する際は，対応するPBDのリストを解放するとともに，各バッファ本体のリファレンスカウントをデクリメントし，リファレンスカウントが0となったバッファ本体のみを実際に解放する．

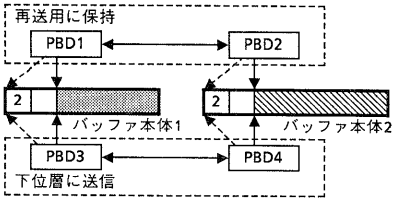


図7 再送のためのPDUの2重化
Fig. 7 Duplication of PDU for retransmission.

以上の方法により，応用層とプレゼンテーション層におけるASN.1の符号化/復号時を含め[6]，すべての層/ASEにわたってユーザデータのコピーを避けている．また，複数の層で再送制御が必要な場合においても，必要に応じてPBDリストを複製し，バッファ本体のリファレンスカウントを調節することによって，データ・コピーを全く行わずに実現可能である．

4.4 ホスト・インタフェース・タスク

ホスト・インタフェース(HIF)タスクは，ホスト・インタフェース・メモリ領域にアクセスし，ホストとボードの間でプリミティブの転送を行う．プリミティブは，OSI 7層ボード内の共有メモリ空間においては，関連するパラメータやPDUが複数のバッファに分散された複雑な構成をとる．このようなプリミティブをホストとの間で転送するためには，複数の小さなサイズの関連データを1つにまとめたり，ホスト・インタフェース・メモリ領域上のバッファサイズよりも大きなサイズのデータを複数に分割して転送する必要がある．このため，HIFタスクとホストとの間で交換されるデータ形式は，個々のプリミティブのデータ型とは独立に，ホスト・ボード間の転送に最適化される必要がある．さらに，ホストCPUとボードCPUの間で整数のバイト順序などのデータ表現方法が異なる場合は，ホスト・ボード間のデータ転送において，その変換を行う必要がある．そこでHIFタスクは，以下の方法によりホスト・ボード間のプリミティブの転送を行

う(図8(a)参照).

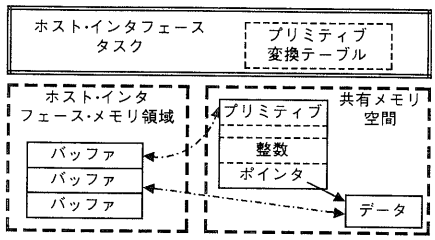
● バイト順序などのデータ表現変換は，ボード上のHIFタスクが行う.

● プリミティブと関連するポインタ・データを再構築したり，整数のバイト順序変換を実行するために，HIFタスクは，プリミティブ中のポインタの位置やデータ表現変換を行う必要があるパラメータの種類と位置を示す変換テーブルを，必要なすべてのプリミティブに対して用意する.

● HIFタスクは，ホスト・インタフェース・メモリ領域を介して受信したデータをもとに，変換テーブルを参照してポインタの設定やデータ表現変換を行い，ボードの共有メモリ空間上にプリミティブを再構築する.

● ボードからホストへの転送においては，HIFタスクが変換テーブルを参照しつつ，共有メモリ空間上に展開された関連データをたどり，ホスト・インタフェース・メモリ領域に書き込む.

図3に示したF-INITIALIZE要求プリミティブに対応する変換テーブルの例を図8(b)に示す．変換テーブルは各構造体ごとに存在し，構造体パラメータが存在すると，対応する構造体の変換テーブルを次々に参照することによってすべてのパラメータの変換を行う.



(a) ホスト・インタフェース・タスクの処理

f_initialize_req_t 構造体変換テーブル		
種別	構造体変換テーブル	変数位置
・・・	・・・	・・・
short	-	FuncUnitsのoffset
struct	cont_type_list_t	ContListのoffset
・・・	・・・	・・・

cont_type_list_t 構造体変換テーブル		
long	-	Numのoffset
pointer	cont_type_t	ContTypesのoffset

cont_type_t 構造体変換テーブル		
struct	obj_id_t	DocTypeのoffset
struct	obj_id_t	AbsSyntxのoffset

obj_id_t 構造体変換テーブル		
long	-	Numのoffset
long[]	-	ObjIdCompのoffset

(b) F-INITIALIZE要求変換テーブル(一部)

図8 ホスト・インタフェース・タスクの処理とプリミティブ変換テーブルの例

Fig. 8 Processing of host interface task and an example of primitive conversion table.

5. ホスト上のソフトウェア

OSI 7層ボードを搭載するホスト上のソフトウェア構成を図9に示す.

5.1 デバイス・ドライバ

デバイス・ドライバは，対象とするホストのOSごとに個々に開発する必要がある．そこで，デバイス・ドライバとしては，OSI 7層ボードに対するプログラムのダウンロード機能や，ボードのホスト・インタフェース・メモリを介してホスト・ボード間で最適化されたデータ転送を行う基本的な入出力機能(write/readなど)のみをサポートし，アプリケーション・プログラムとの間の実際のプリミティブの送受信は，APIライブラリによって実現することとした.

5.2 APIライブラリ

APIライブラリは，業務に応じたアプリケーション・プログラムの開発をサポートするため，OSI 7層ボードで使用される構造化されたデータ型を持つプリミティブをボードとの間で送受信するための関数(osi_snd/osi_rcv)を提供する[7].

これらの関数は，デバイス記述子(ファイル・ディスクリプタ)とプリミティブ・バッファへのポインタを引数に持ち，次のように使用される.

● osi_snd/osi_rcvの戻り値として，成功/失敗のほかに，不完全入出力が設けられている.

● アプリケーション・プログラムからのosi_snd/osi rcvの発行に対して，ボードとの間で複数に分割したデータ転送が必要な場合も，APIライブラリは1回のwrite/readのみを発行する．その結果，プリミティブ全体を読み書きできなかった場合は，不完全入出力の戻り値を返す.

● アプリケーション・プログラムは，戻り値が不完全入出力である場合は，そのプリミティブに対して，成

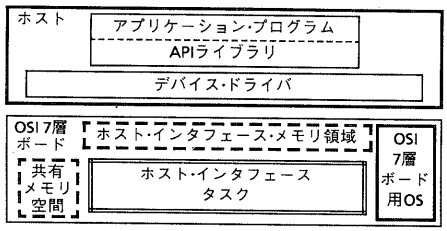


図9 ホスト上のソフトウェア構成

Fig. 9 Host software configuration.

功または失敗が返されるまで，osi_snd/osi_rcv の発行を繰り返す．

この方法により，ホスト・インタフェース・メモリ領域のバッファよりも大きなユーザデータの送受信においても，アプリケーション・プログラムからの osi snd/osi_rcv の発行回数と，OSI 7層ボードに対する write/read の発行回数を同一にすることができる．これにより，API ライブラリがデバイス・ドライバの write/read 機能のみを使用し，ホストの OS に依存しないようにすることができる．また，osi_snd を発行するごとにボードに対する書き込みが可能かどうかのチェックを行うことにより，ボード上のバッファ領域がビジーで送信できない場合にアプリケーション・プログラムのデッドロックを防ぐことができる．

6. 実装結果と性能評価

6.1 実装結果

以上のような実装方式に基づいて，OSI 7層ボードのハードウェアとして，これまでに，PC-9800 シリーズ用と VME バス用のベースモジュールと，パケット網/電話網と ISDN の基本速度インタフェースに対応する回線制御モジュール[8] を開発した．また，OSI 7層ボード上で動作する FTAM，MHS P2/P7[9]，MHS P1/P2/RTS[10] などを含む OSI プロトコル用のソフトウェアを開発した．本章では，このうち，パケット網/電話網に対応する回線制御モジュールを持ち，FTAM を搭載するパソコン用の OSI 7層ボードの実装結果と性能評価について報告する[11]．

表1 に，ベースモジュール上に実装した FTAM までのプロトコル処理プログラムの仕様と実行モジュール・サイズを示す．ACSE 以下のプログラム・モジュールは，MHS などの他の応用層プロトコルを搭載する場合も共通である．

FTAM プログラム・モジュールは，カーネル，読み出し，書き込み，限定ファイル管理，グルーピングの各機能単位をサポートし，転送(T)，管理(M)，および転送と管理(TM)の3つのサービス・クラスに対応している．また，ドキュメント・タイプとして，FTAM-1(無構造テキスト)および FTAM-3(無構造バイナリ)をサポートする．

本実装がサポートするサービス・クラスにおいては，F-SELECT，F-DESELECT，F-CREATE，F-DELETE，F-OPEN，F-CLOSE，F-READ-ATTRIB の各プリミティブは，F-BIGIN-GROUP と F-END-GROUP の間で，定められた順序にしたがってグルーピングされる．そこで，**表2** に示すように，これらのプリミティブの取りえる組み合せを1つにまとめたプリミティブを API として提供することにより，ホスト・ボード間の転送の効率化やアプリケーション・プログラム作成の容易化を図っている．

6.2 性能評価

性能評価実験では，2 Mbps の疑似回線により OSI 7層ボードを搭載した2台のパソコンを直結した形態と，1台のパソコンを用いて OSI 7層ボードのトラン

表 1 プロトコル処理プログラムの仕様
Table 1 Specification of protocol programs.

層/ASE	仕様	サイズ (K バイト)
FTAM	ISO 8571 カーネル，読み出し，書き込み，限定ファイル管理，グルーピングの各機能単位 ドキュメント・タイプ： FTAM-1, FTAM-3	300
ACSE	X. 217/X. 227	162
プレゼンテーション	X. 216/X. 226 カーネル機能単位	213
セション	X. 215/X. 225 バージョン2，全機能単位	297
トランスポート	X. 214/X. 224 クラス0およびクラス2	95

表 2 グルーピングされた FTAM プリミティブ
Table 2 Grouping of FTAM primitives.

ボードが提供するプリミティブ	FTAM プリミティブの組み合せ (F-BEGIN/END-GROUP を含む)
F-Sel-RdAttr-Open	F-SELECT, READ-ATTRIB, OPEN
F-Cre-RdAttr-Open	F-CREATE, READ-ATTRIB, OPEN
F-Cls-RdAttr-Desel	F-CLOSE, READ-ATTRIB, DESELECT
F-Cls-RdAttr-Del	F-CLOSE, READ-ATTRIB, DELETE
F-Sel-RdAttr-Desel	F-SELECT, READ-ATTRIB, DESELECT
F-Sel-RdAttr-Del	F-SELECT, READ-ATTRIB, DELETE
F-Cre-RdAttr-Desel	F-CREATE, READ-ATTRIB, DESELECT
F-Cre-RdAttr-Del	F-CREATE, READ-ATTRIB, DELETE

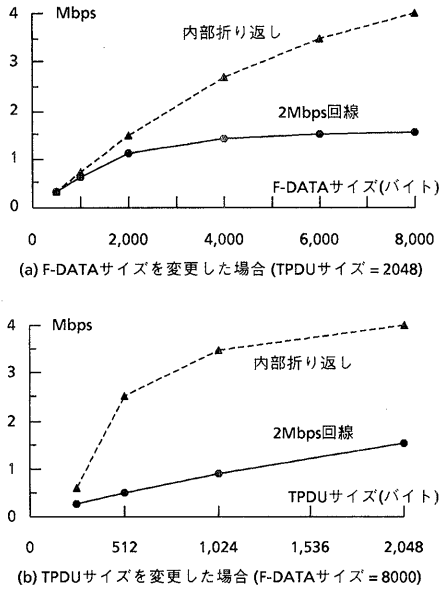


図10 OSI 7層ボードのスループット特性(FTAM)
Fig. 10 Throughput performance of OSI 7 layer board (FTAM).

スポート層の下位で内部折り返しを行う形態を採用した．パソコン本体上には，FTAMユーザ・プログラムを動作させる．実験では，FTAMユーザが1回あたりのF-DATA要求としてボードに転送するデータ・サイズ(F-DATAサイズ)，およびトランスポート層のTPDUサイズを変更して測定を行った．スループットはRAMディスク上の2Mバイトのファイル転送に要した時間から求めたファイル転送速度とし，ボード内部で折り返す場合は，得られた転送速度の2倍の値を近似的にトランスポート層以上の片方向のスループットとした．FTAMのドキュメント・タイプとしては無構造バイナリ(FTAM-3)を使用し，ネットワークのパケット・サイズは2048バイト，ウインドウ・サイズおよびデータリンクのアウトスタンディング・フレーム数はともに7とした．

TPDUサイズを2048バイトとし，1回当たりのF-DATAサイズを変更した場合のスループットを**図10**(a)に示す．また，F-DATAサイズを8000バイトとしてTPDUサイズを変更した場合のスループットを図10(b)に示す．さらに，2 Mbpsの疑似回線を用いて直結し，TPDUサイズを2048バイト，F-DATAサイズを8000バイトおよび2000バイトとした場合におけるベースモジュール上の各タスクの相対的な処理時間を**図11**に示す．

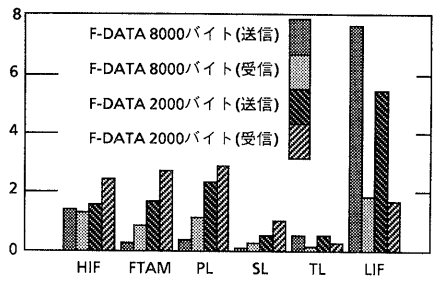


図11 タスクごとの相対処理時間
Fig. 11 Relative processing time of each task.

7. 考　　察

7.1 実装に関する考察

(1) FTAMやMHSなど応用層までのすべてのOSI通信機能を提供するOSI 7層ボードにより，OSIに従ったさまざまな情報通信システムの構築を，市販のパソコンやワークステーションを用いて容易に実現可能となる．特に，本ボードを複数枚搭載することにより，端末機能のみならず，メッセージ中継機能を持つMHSシステムや各種ゲートウェイなどの構築を，パソコンやワークステーションをベースとして低コストで簡便に実現することも可能である[12),13)]．

(2) 既存のOSI用通信ボードとしては，筆者らが開発したOSI 5層ボード[3)]を含め，ACSEまたはセション層までのOSIプロトコルをサポートするものが報告されている[14),15)]．これらのボードでは，対象とするホストとネットワークの組み合せごとに個別のハードウェアを開発している．また，搭載するソフトウェアに関しては，各層をメインプログラムから呼ばれるサブルーチンとして実現する単純な方式を採用している．

これに対し，開発したOSI 7層ボードでは，回線制御モジュールの変更によって各種ネットワークに柔軟に対応できる．また，OSI 7層ボードでは，要求に応じて，応用層などのプログラムを新規に開発することを考慮し，タスク管理機能やタスク間のメモリ保護機能をサポートするOSI 7層ボード用OSを搭載している．

(3) OSI 7層ボード用OSでは，OS自身のオーバヘッドを小さくするため，ボード上のプロトコル処理プログラムの開発・実行を行うために必須の機能のみを実現した．ある層/ASEのプログラム障害が他の層/ASEの動作に影響を与えないことを保証するタスク毎のメモリ保護機能は，大規模なOSIプログラムの開発には不可欠である．また，複雑な構造のプリミティ

ブを層/ASE間で効率的に転送するためには，ポインタでつながれたデータを，複数のタスクからアクセス可能な共有メモリ空間のサポートが必須である．さらに，隣接する層/ASEからプリミティブを受信し，内部処理を行ってプリミティブを送信するというプロトコル処理の特徴を考慮し，プリミティブの送受信時を契機としたスケジューリングを行うことにより，不要なタスク切替えを避けることができる．

（4） OSI 7層ボードでは，プログラムを独立に作成するために，各プログラム・モジュールをタスクとして実現する方式を採用している．この方式は，次のような利点を有する．

- 個々のプログラム・モジュールは，関数やグローバル変数などの名前が互いに独立であり，ソース・プログラムを維持・管理しやすい．
- OSI 7層ボード用OSが，ボード上のタスクに互いに独立したアドレス空間を与え，タスク間のメモリ保護機能を提供するため，プログラム誤りを局所化しやすくデバッグを容易化できる．

OSI 7層ボード用のプロトコル処理プログラムの開発においては，各層/ASEの仕様変更や機能追加などに対応すること，通常は各層/ASEごとに別々のプログラマが開発・デバッグを行うこと，プログラム誤りの検出および修正を層/ASE単位に局所化することなどが要求されるため，基本的に各層/ASEをそれぞれ独立したプログラム・モジュールとし，かつそれぞれをタスクに対応させて開発・実行している．

一方，プログラム・モジュールをタスクとして実行させる方法には，一般的に，性能上のオーバヘッドが増大する，ポインタなどを多用した複雑なデータ構造を共有できないなどの問題点が存在する．OSI 7層ボード用OSは，タスク切替えのオーバヘッドを最小限とし，かつ不要なタスク切替えを避けること，すべてのタスクから同一の仮想アドレスでアクセス可能な共有メモリ空間をサポートすることなどにより，これらの問題点を解決している．

さらに，OSI 7層ボードでは，層/ASEを必ず別タスクで実現する必要はなく，場合に応じて柔軟なタスク構成を採用可能である．例えば，OSI 7層ボード用のMHS P2/P7プログラムにおいては，実行効率の向上を目的として，ROSE(Remote Operations Service Element)を独立したタスクとはせず，P2/P7モジュールに含まれるライブラリとして開発している(**図12**参照)．

（5） OSI 7層ボードのベースモジュールでは，コピーを伴わないPDU作成/解析処理により，すべての層にわたってユーザデータを一度もコピーしていない．これにより，バッファの割当て・解放やデータコピーの処理を省くことができ，高いスループットを得ることができたと考えられる．

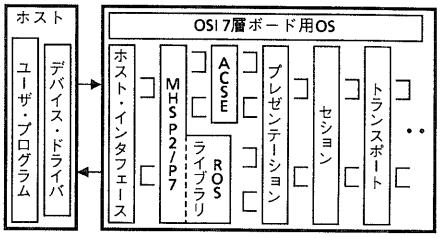


図12 MHS P2/P7のタスク構成
Fig. 12 Task configuration of MHS P2/P7.

（6） 構造化されたプリミティブのデータ型を採用したことにより，プロトコル処理プログラムの開発を容易にし，かつ高いスループットを実現することができた．しかし，このようなデータをボードとホストの間でやり取りするためには，データの分解/再構築やデータ表現変換などを伴う複雑な処理が必要となる．そこで，ホスト・ボード間でプリミティブの変換と転送を行うホスト・インタフェース・タスクを設け，プリミティブのデータ構造に対応するプリミティブ変換テーブルに基づいて変換を行う汎用性の高い方式を採用した．

（7） パソコンやワークステーションにOSI通信機能を実装する場合，これらの計算機本体上で動作するソフトウェアとして実現する方式も存在する．近年，パソコン/ワークステーションの処理能力が向上しており，単に性能を追求した場合には，プロトコル処理を拡張ボードで実行するよりも有利な場合もあると考えられる．しかし，OSI 7層ボードでは，7層すべてをボードでサポートすることにより，OSI準拠の各種通信システムの開発の簡便化・迅速化を図ることができること，メモリ容量に制限のあるパソコンも利用可能であること，他機種への移植性や，プロトコル・プログラムの保守性が向上すること，ホストのCPUやメモリにプロトコル処理の負荷がかからないため，アプリケーション・プログラムの設計における自由度が向上することなどの利点が存在すると考えられる．

（8） 開発したOSI 7層ボードは，ネットワーク層以下に関しては公衆パケット網や公衆ISDNとの接続性を確認済であるとともに，応用層までを含めた7層すべてに関しては，FTAM, MHS P1/P2, MHS P2/P7などを用いて，これまでに国内や国外で開発された他の製品との相互接続実験を行い，その動作確認を行

っている．

（9） OSI勧告/標準では，その実装や使用がオプションとなっている機能やパラメータなどが数多く存在する．OSI 7層ボードでは，ACSEおよびプレゼンテーション層以下のプロトコルに関しては，さまざまな応用層プロトコルに対応可能とするため，勧告で規定されている機能を基本的にすべて実装している．一方，FTAMやMHSなどの個別の応用層プロトコルに関しては，既存の実装規約や機能標準(Functional Standard)，国際標準化プロファイル(ISP：International Standardized Profile)などを考慮し，これらに対応する必須機能やオプション機能を実装している．PDUサイズやオプション機能の使用/不使用などのパラメータ指定に関しては，初期化時にこれらを設定する初期化データをホストからボードに通知することによって実現している．この初期化データは，物理層におけるV.24/V.28またはV.10/V.11の回線インタフェースの選択，データリンク層におけるタイマ値やアウトスタンディング・フレーム数，ネットワーク層におけるパケット・サイズ，ウインドウ・サイズ，各種ファシリティの指定，トランスポート層のクラス指定やTPDUサイズ，セション層のバージョン指定など，サービス・プリミティブでは提供されない各種オプションの選択を実現しており，ホスト上の初期化データ設定ツールを用いて容易に変更可能となっている．

7.2 性能評価に関する考察

（1） 実験の結果，2 Mbpsの回線を用いた測定では，TPDUサイズが2048バイトでF-DATAサイズが4000バイト以上の場合に約1.5 Mbpsのスループットが得られた．また，内部折り返しによるトランスポート層以上の測定では，F-DATAサイズを8000バイトとすることによって約4 Mbpsの高いスループットが得られた．これは，PDUやプリミティブの作成/解析時にコピーを避ける方式を採用したことなどの効果が現れたものと考えられる．

（2） 図10(b)より，2 Mbpsの回線上でTPDUを小さくした場合，ほぼ直線的にスループットが低下していることがわかる．これは，ベースモジュールの処理能力がX.25/X.32対応の回線制御モジュールに対して高く，ネットワーク層上の転送データ・サイズが小さくなると，回線制御モジュールの処理オーバヘッドがネックとなるためと考えられる．

（3） 内部折り返しによるトランスポート層以上の測定では，F-DATAサイズが8000バイトでTPDUサイズが512バイト以上であれば，2.5～4 Mbpsのスループットが得られている．このことから，より高速なネットワークに対応する回線制御モジュールを開発することにより，7層全体で，より高いスループットが得られると期待される．

（4） F-DATAサイズを小さくすると，内部折り返しの場合においてもスループットが低下している．図11を見ると，F-DATAサイズが2000バイトの場合では，8000バイトの場合と比較して，セション層(SL)やトランスポート層(TL)の処理時間はそれほど増加していないが，FTAMとプレゼンテーション層(PL)における処理時間の増加の割合が大きいことがわかる．スループットの低下は，上位層におけるASN.1の符号化/復号処理の回数が増加し，そのオーバヘッドが大きくなったためであると考えられる．

8. おわりに

本稿では，応用層までのすべてのOSI通信機能を実現する，パソコン/ワークステーション用のOSI 7層ボードの実装と性能評価について報告した．本ボードは，ベースモジュールと回線制御モジュールにモジュール化されたハードウェア構成を持ち，さまざまなネットワークや応用層プロトコルに汎用的に対応できる．また，OSI 7層ボード用OSのサポートにより，各層/ASEごとの独立なプログラム開発と，効率的なプロトコル処理を実現している．FTAMを用いた性能評価の結果，2 Mbpsの回線上で約1.5 Mbps，内部折り返しによるトランスポート層以上の測定では，約4 Mbpsの高いスループットが得られることが確認された．

現在，CSMA/CDのLANを対象とした回線制御モジュールや，共通管理プロトコル(CMIP：Common Management Information Protocol)などの各種応用層ソフトウェアの開発を進めるとともに，MHSプロトコルを搭載するOSI 7層ボードを利用したパソコンベースの小型メッセージ通信処理システムなどの応用システムの検討[13]も行っており，本ボードがOSIに基づく通信システム構築のために広く使用されることが期待できる．

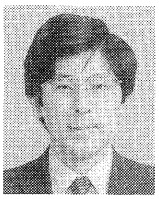
謝辞 本論文の作成にあたり，ご指導いただいたKDD研究所浦野所長，眞家次長に感謝いたします．

参考文献

1) 加藤聰彦，井戸上彰，鈴木健二：パソコン用OSI 7層ボード，電子情報通信学会1992年秋季大会，B-412, p. 3-78 (1992).
2) 井戸上彰，加藤聰彦，鈴木健二：OSI 7層ボードの実装と評価，情報処理学会研究報告，Vol. 93,

No. 58, pp. 211-218 (1993).
3) Idoue, A., Kato, T., Suzuki, K. and Urano, Y.: Design and Implementation of OSI Communication Board for Personal Computers and Workstations, *Proc. of ICCC '90*, pp. 585-592 (1990).
4) 井戸上彰，加藤聰彦，鈴木健二，小野欽司：OSI 7層ボードのためのオペレーティング・システム，情報処理学会論文誌，Vol. 35, No. 5, pp. 865-874 (1994).
5) 加藤聰彦，井戸上彰，鈴木健二：汎用 OSI 7層ボードにおけるプロトコル・プログラムの構成法，第44回情報処理学会全国大会論文集，4 L-11, pp. 1-167-168 (1992).
6) 加藤聰彦，井戸上彰，鈴木健二：OSI プロトコル実装のためのユーザデータをコピーしないバッファ制御方式，情報処理学会研究報告，Vol. 93, No. 83, pp. 95-102 (1993).
7) 加藤聰彦，井戸上彰，鈴木健二：OSI 7層ボードのアプリケーション・プログラミング・インタフェース，第45回情報処理学会全国大会論文集，6V-1, pp. 1-195-196 (1992).
8) 石倉雅巳，井戸上彰，加藤聰彦，鈴木健二：ISDN 対応 OSI 7層ボード，電子情報通信学会 1993 年春季大会，B-658, p. 3-209 (1993).
9) 井戸上彰，加藤聰彦，鈴木健二，飯作俊一：パソコン用 MHS P2/P7 ボードの開発，電子情報通信学会 1992 年秋季大会，B-413, p. 3-79 (1992).
10) 加藤聰彦，井戸上彰，鈴木健二：MHS P1/P2/RT ボードの開発，電子情報通信学会 1993 年春季大会，B-655, p. 3-206 (1993).
11) 井戸上彰，加藤聰彦，鈴木健二：汎用 OSI 7層ボードの性能評価，第45回情報処理学会全国大会論文集，6V-2, pp. 1-197-198 (1992).
12) 鈴木健二，加藤聰彦：OSI 7層ボードを用いた新しい通信システムの構想，第46回情報処理学会全国大会論文集，3P-7, pp. 1-227-228 (1993).
13) 井戸上彰，加藤聰彦，鈴木健二：OSI 7層ボードを用いた小型メッセージ通信処理システム，情報処理学会マルチメディア通信と分散処理ワークショップ論文集，pp. 121-130 (1993).
14) Johnson, W. R.: An Overview of the HP OSI Express Card, *HP Journal*, Vol. 41, No. 1, pp. 6-8 (1990).
15) 瀬戸康一郎，鈴木靖雄，浅野光春：MAP トランスポート層プロトコルの実装，情報処理学会研究報告，Vol. 91, No. 83, pp. 117-122 (1991).

(平成6年7月19日受付)
(平成7年1月12日採録)



井戸上　彰（正会員）

昭和36年生．昭和59年神戸大学工学部電子工学科卒業．昭和61年同大学院修士課程修了．同年国際電信電話(株)入社．現在，同社研究所高速通信グループ主査．この間，OSI 5層ボードやOSI 7層ボードを中心とした，OSI 通信プロトコルの実装に関連するハードウェア・ソフトウェア，オペレーティングシステムなどの研究に従事．平成4年情報処理学会全国大会奨励賞受賞．電子情報通信学会会員．

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．同年国際電信電話(株)入社．現在，同社研究所高速通信グループ主任研究員．工学博士．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学者．この間，OSI 通信プロトコルの実装，通信システムの試験技法，プロトコルの形式記述，高速・分散処理などの研究に従事．昭和60年情報処理学会学術奨励賞，平成元年度元岡賞受賞．平成5年度より電気通信大学大学院情報システム学研究科客員助教授．電子情報通信学会会員．

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．同年国際電信電話(株)入社．現在，同社研究所高速通信グループリーダ．工学博士．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，高速・分散処理の研究に従事．昭和62年度前島賞，平成4年度電子情報通信学会業績賞を各受賞．平成5年度より電気通信大学大学院情報システム学研究科客員教授．電子情報通信学会，IEEE 各会員．