OSI 7層ボードのためのオペレーティング・システム

井戸上 彰[†] 加藤聰彦[†]
鈴木健二[†] 小野欽司[††]

筆者らは，パソコンやワークステーションにOSI通信機能を実現するために，CPUやメモリを搭載する拡張ボードでOSIの7層すべてのプロトコルをサポートするOSI 7層ボードを開発した．OSI 7層ボードには複雑で大規模なプロトコル・プログラムが搭載される．このようなプログラムを容易に開発し，かつボード上で効率的に実行させるためには，OSI 7層ボード用のオペレーティング・システム（OS）が必要となる．そこで，OSIプロトコルに最適化されたOSI 7層ボード用OSを開発した．本論文では，OSI 7層ボード用OSの設計と各機能の実現方式，および性能評価について述べる．OSI 7層ボード用OSは，層ごとの独立なプログラム開発をサポートするために，各層のプロトコル・プログラムをそれぞれ独立したタスクとし，タスクごとの仮想アドレスとメモリ保護機能を提供している．また，効率的なプロトコル処理の実行を可能とするため，プロトコル処理を考慮したスケジューリングや，グローバル・バッファを用いた高速なタスク間通信機能を実現している．タスク切替えやタスク間通信に関する性能評価を行った結果，実装したOSI 7層ボード用OSは，他の汎用的なOSと比較して，高速な処理を提供できることが示された．

Operating System for Communication Board Supporting OSI 7 Layer Protocols

Akira Idoue,[†] Toshihiko Kato,[†] Kenji Suzuki[†] and Kinji Ono[††]

In order to provide OSI communications capabilities for personal computers and workstations, we have developed the OSI 7 layer board which supports OSI protocols of physical through application layers. Since large and complicated protocol programs are installed on the OSI 7 layer board, it is required to implement an on-board operating system to facilitate the development and execution of these programs. We have implemented the operating system for the OSI 7 layer board which offers the functionalities optimized for supporting OSI protocols. This paper presents the design, implementation and evaluation of the operating system for the OSI 7 layer board. In order to help layer-by-layer development of protocol programs, the operating system realizes a protocol program as an independent task, and provides virtual address space and memory protection between tasks. It also supports the efficient task scheduling and inter-task communication mechanism to achieve high throughput. Performance measurement results show that the operating system provides high speed task switch and inter-task communication compared with other general purpose operating systems.

1. はじめに

開放型システム間相互接続（OSI: Open Systems Interconnection）の標準化の進捗に伴い，OSIに準拠した通信機能をどのように実現するかが重要な課題となっている．OSIでは7つの層に対してそれぞれ通信プロトコルを規定しているため，OSI通信機能は全体で複雑なものとなる．そこで，広く普及したパソコンやワークステーションにOSI通信機能を実現するためには，CPUやメモリを搭載する拡張ボードによりOSIの通信プロトコルをサポートする方式が，利便性や移植性などの点で有効である[1)~4)]．このため筆者らは，OSIプロトコルをサポートする各種OSI通信ボード[1)~3)]の検討を進めているが，このたびOSIの7層すべてのプロトコルを拡張ボード上で実装する「OSI 7層ボード」の開発を行った[3)]．

OSI 7層ボードでは，7層全体のプロトコルを処理する大規模なプログラムが実装される．このようなプログラムを容易に開発し，かつボード上で高速に実行するためには，OSI 7層ボード用のオペレーティング・システム（OS）が必要となる．通信ボードのハードウェア的な特徴を考慮し，OSIプロトコルの開発・実行に最適な機能を提供するためには，OSI 7層ボード用OSとして，既存の機器組み込み型OSや汎用OSを

† 国際電信電話株式会社研究所
KDD R & D Laboratories
†† 学術情報センター
National Center for Science Information Systems

利用することは得策でない．機器組み込み型 OS は，比較的小規模なプログラム開発を想定しているため，タスクごとの仮想アドレスやメモリ保護機能を持たない場合が多い．これに対し，OSI 7層ボード上のプログラムは，複数のプログラマにより開発されるため，OSI 7層ボード用 OS は，プログラムやデータの保護機能を強化する必要がある．また，汎用 OS は多様な業務に対応する機能を持っている反面，OS のオーバヘッドも大きくなりがちである．OSI 7層ボード用 OS は，高速なプロトコル処理を実現するために，プロトコル処理に必要な機能のみを実装し，OS のオーバヘッドを最小化する必要がある．

一方，OSI プロトコルをサポートする通信ボードも報告されているが[4),5)]，これらのボードでは搭載するプロトコルを固定的に定めているため，層ごとの独立な開発を容易にするためのメモリ保護機能などは提供していない．これに対し，筆者らの OSI 7層ボード用 OS は，OSI の7層全体のプロトコルの実装を目的として，独立に作成された複数のプログラム間のメモリ保護を実現するとともに，高いスループットを得るために，プログラムのスケジューリングや，プログラム間のデータのやり取りを効率的にサポートしている[6),7)]．

本稿では，筆者らが開発した OSI 7層ボード用 OS の設計，実装および性能評価について述べる．2章では，OSI 7層ボード用 OS に対する要求条件を述べる．3章では，それらの要求条件に基づいて，OSI 7層ボード用 OS の設計方針を述べ，4章で，OS の提供する各機能の実現方式について述べる．5章では，筆者らが開発した OSI 7層ボード上で動作する OS の実装結果と性能評価を示し，6章で OS の機能および性能に関して考察する．

2. OSI 7層ボード用 OS に対する要求条件

OSI の7層すべてのプロトコルを実現するボードのための OS は，OSI プロトコルの特徴や，ボードのハードウェアの制限から，次のような要求条件を持つ．

①OSI 7層ボードでは，1つの層または応用サービス要素（ASE: Application Service Element）のプロトコルごとに，別々のプログラマがプログラム開発を行うのが通常である．このため，あるプログラムの誤りが，他のプログラムの内部情報を破壊するといった，複数の層/ASE にまたがった障害が発生しないように，ボード用 OS は各プログラムに対する保護機能を提供する必要がある．

②OSI では，各層/ASE の間でやり取りされるデータはサービス・プリミティブとして規定される．サービス・プリミティブは，多数のパラメータを持ち，またそれらの多くが可変長やオプションとなる複雑なデータ構造をとる．したがってボード用 OS は，ボード上で実行される各層/ASE のプログラムの間で，複雑な構造を持つサービス・プリミティブを効率的に転送できる必要がある．

③伝送路の高速化に伴い，OSI 通信に対して高いスループットが要求されている．このため，ボード用 OS は自分自身のオーバヘッドをできるだけ削減し，OSI プロトコルを高速に処理する必要がある．

④ボードのハードウェア制限により，搭載したメモリのサイズによって使用可能なバッファが制限される．このため，ボード用 OS は，すべてのバッファを消費しつくして，プロトコル処理の継続が不可能となる事態を避ける必要がある．

⑤ボードは，外部とインタフェースする装置として，搭載される本体計算機（以下，ホストと呼ぶ）および回線制御用のハードウェアを有するのみである．したがって，ボード用 OS は，限定された外部装置との入出力を効率的に行う機能を持つ必要がある．

⑥各層/ASE のプログラムは，開発途上において，ボード上で実行/試験される．そこで，ボード用 OS には，プログラムをボード上で動作させ，効率的にデバッグする機能が要求される．

3. OSI 7層ボード用 OS の設計方針

2章で述べた要求条件に基づき，OSI 7層ボード用 OS の設計を行った．OS 自身のオーバヘッドを削減するため，OSI プロトコルの処理を高速に実行するために必要な最低限の機能のみを実現することとし，OS の各機能に対して以下のような設計方針をたてた．

（1） タスク管理とスケジューリング

①タスクによる各層/ASE の実現

各層/ASE のプログラムに対する保護機能を提供し，各プログラムの独立開発をサポートするために，各層/ASE をそれぞれ独自の仮想アドレス空間を持つタスクとして実現する．

②高速なタスク切替え

各層/ASE をタスクとして実現するため，プロトコル処理におけるタスク切替えがスループットに重要な

影響を与える．このため，タスクの切替えごとに設定が必要なタスク管理情報を最小化し，高速なタスク切替えを実現する．

③プロトコル処理を考慮したタスク切替えとスケジューリング

OSI の1つの層/ASE の処理は，その上位や下位の層/ASE からのサービス・プリミティブを待ち，受信プリミティブに応じた処理を行い，上位または下位へプリミティブを出力するという手順の繰り返しである．一方，優先データなどを含むプリミティブについては，その処理を優先的に行う必要がある．このようなプロトコル処理の特徴を考慮し，タスク間通信によるプリミティブの送受信時を，タスク切替えの契機とする．また，プリミティブに優先度を設け，優先プリミティブを送信した時点で受信タスクに制御を移すスケジューリングを行う．さらに，ホストや回線制御ハードウェアからのデータ転送要求に迅速に対応するため，これらの外部装置からの割り込みがあった場合は，後述のホスト・インタフェース・タスクや回線インタフェース・タスクを優先的に実行する．

④ホストからのダウンロードによる静的なタスク生成

さまざまな業務に対応する OSI プログラムを実行可能とするため，各層/ASE のタスクに対応するプログラムを，初期化時にホストからダウンロードする機能を提供する．ボードの動作中におけるタスクの動的な生成や消滅はサポートしない．

（2） メモリ管理

①セグメント単位のアドレス空間とメモリ保護

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により実現し，セグメント単位の仮想アドレス空間とメモリ保護を実現する．

②フロー制御機能を持つバッファ管理

OS は，タスク間通信やタスク内部の情報などのために動的なバッファ割当て機能を提供する．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，ボードに対するデータの入力を制限させる機能を提供する．

（3） タスク間通信

①グローバル・バッファとキューを利用した高速なタスク間通信

OS は，すべてのタスクから同一の仮想アドレスによってアクセス可能なグローバル・バッファの割当て機能をサポートし，各層に対応するタスクの間でやり取りされるサービス・プリミティブを，グローバル・バッファ上に実現する．これにより，構造体，共用体，ポインタなどを含む複雑なデータ構造のプリミティブを直接転送可能とする．キューを介して，グローバル・バッファ上のプリミティブへのポインタのみを受け渡すことにより，高速なタスク間通信を提供する．

（4） 割り込み管理と入出力制御

①割り込み管理

OS は，ホスト，回線制御ハードウェア，ハードウェア・タイマなどからの割り込みを処理する．オーバヘッドを避けるため，割り込み時の処理はできるだけ簡略化する．

②ホスト/回線制御インタフェース

ホストおよび回線制御ハードウェアとの間の通常のデータ入出力に関しては，OS 自身に外部ハードウェアに依存する複雑な機能を持たせることを避けるため，それぞれをタスクとして実装する．

（5） デバッグ機能

①デバッグ用入出力とデバッガ

OS はホストとの間で，通常の入出力機能とは別に，デバッグ・メッセージの出力やデバッグ用コマンドなどの入力をサポートする．また，ホスト以外のコンソールとの間で文字入出力を可能とするため，非同期ポートによる入出力機能も提供する．これらのデバッグ用入出力機能を用いて，ボードのメモリ・ダンプや各種状態表示などが可能なデバッガを実装する．

4. OS の機能の実現方式

OSI 7層ボード用 OS の構成を**図1**に示す．図1で示されるカーネル・レベルが，OS 自身のコードやデータなどのプログラム領域に対応する．本 OS は，3章で述べた設計方針に従って，タスク管理，スケジューリング，メモリ管理，タスク間通信などの機能を実現している．OS 自身にはプロトコル処理の機能は含まれず，OSI の個々の層/ASE に対応するプロトコル・プログラムが，本 OS のもとで，それぞれ独自の仮想アドレス空間を持つユーザ・レベルのタスクとして実行される．これらのタスクは，システム・コール・エントリを通して本 OS の機能を利用する．

4.1 タスク管理とスケジューリング

（1） タスク管理情報

OSI 7層ボード用 OS は，タスク管理やタスク切替

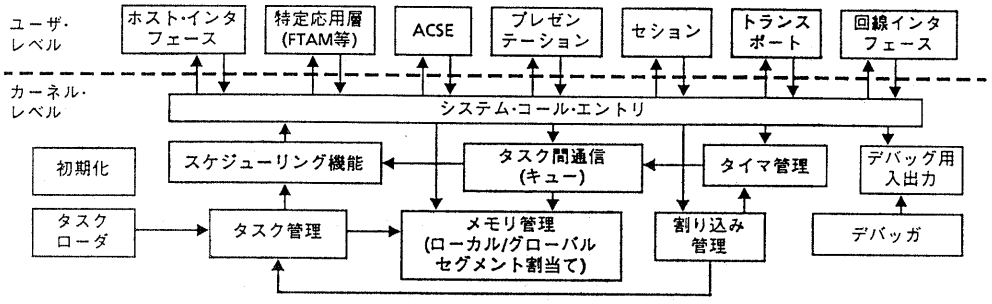


図 1 OSI 7層ボード用 OS の構成
Fig. 1 Configuration of operating system for OSI 7 layer board.

- タスク名： プログラマが指定したタスクの論理名
- タスクID： タスク起動時にOSが割り当てるタスクのID
- 特権レベル： タスクの特権レベル
- タスク状態： 「実行中」、「実行待ち」などのタスクの状態
- CPU時間： タスクのCPU使用時間
- 空きバッファ・リストへのポインタ：
  そのタスクにローカルな空きバッファ・リストへのポインタ
- 割り当てた受信キュー・リストへのポインタ：
  そのタスクが割り当てた受信キューのリストへのポインタ
- コンテクスト情報：
  タスク切替えの際にセーブされる汎用レジスタなどの情報
- ローカル・セグメント管理テーブル：
  タスクごとのコード領域やデータ領域などをセグメントとして仮想アドレス空間を実現するためのテーブル(4.2参照)

図 2 タスク管理情報
Fig. 2 Task control information.

えのオーバヘッドを削減するために，タスクごとの管理情報を必要最小限とし，**図2**に示す情報のみを使用する．

（2） タスクの状態管理とスケジューリング

OSI 7層ボード用 OS は，各タスクに対して，以下の状態を割り当てる（**図3**参照）．

- 現在 CPU が割り当てられている状態として，「ユーザ・モード実行中」と「カーネル・モード実行中」
- タスク間通信用のキューへのプリミティブ到着を待っている「待機中」
- プリミティブを受信し，実行可能となっている状態として，優先プリミティブが到着した「高優先度実行待ち」と，通常プリミティブが到着した「通常優先度実行待ち」

OS は以下の方法により，タスクのスケジューリングを行う．

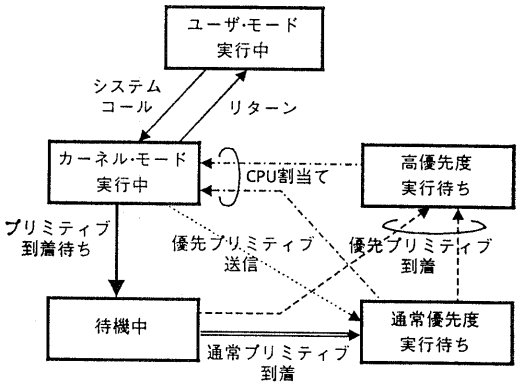


図 3 タスクの状態遷移
Fig. 3 Task state transition diagram.

①実行中のタスクが，受信キューへのプリミティブ到着待ちのシステム・コールを発行した時点で，そのタスクを待機中とする．

②実行中のタスクが，通常プリミティブの送信システム・コールを発行した場合は，そのプリミティブの受信タスクを通常優先度の実行待ちに遷移させた後，送信タスクの実行を継続する．

③実行中のタスクが，優先プリミティブの送信システム・コールを発行すると，そのタスクを通常優先度の実行待ちとし，優先プリミティブの受信タスクを高優先度の実行待ちとする．

④実行中のタスクが待機中または実行待ちとなった場合，高優先度の実行待ちタスクを優先的に選んで実行する．

以上のような通常のスケジューリングに加えて，ホストや回線制御ハードウェアからの割り込みに対しては，ホスト・インタフェース・タスクや回線インタフェース・タスクを高優先度の実行待ちに遷移させる．

（3） タスクのダウンロード

各層/ASE のプログラムに対応するタスクは，OSI

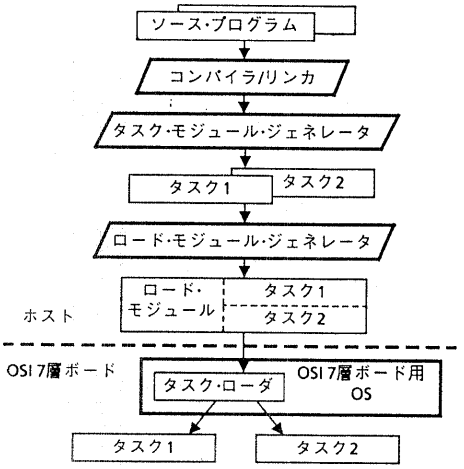


図 4 タスクのダウンロード
Fig. 4 Download of task modules.

7層ボード用 OS のタスク・ローダにより，ホストからダウンロードされる（図4参照）．

ダウンロードされるロード・モジュールは，使用するコンパイラ/リンカに依存せず，一定のフォーマットに従う必要がある．このため，以下の方法でロード・モジュールを生成することとした．

①既存のコンパイラ/リンカによって生成されるオブジェクト・ファイルから，OSI 7層ボード用タスク・モジュール・ジェネレータを用いて個々のタスクに対応するモジュールを作成する．

②ボード上で実行されるタスクの種類は，静的に定まっているため，ダウンロードの手順を簡易化するために，ロード・モジュール・ジェネレータにより，複数のタスク・モジュールを組み合せて1つのロード・モジュールを作成する．

OS 内のタスク・ローダは，ホストから転送されたロード・モジュールを一旦ボード上のバッファに蓄える．その後，個々のタスク・モジュールの制御情報を参照して，コード領域やデータ領域用のメモリ・セグメントを割り当て，指定の物理アドレス上に展開し，タスク管理情報を初期化してタスクとして登録する．

4.2 メモリ管理

（1） セグメントの割り当て

OSI 7層ボード用 OS は，ボード上のメモリを，自分自身のコードやデータを格納するカーネル領域，ロードされたタスクのコード等を格納するタスク領域，動的に確保されるバッファのためのヒープ領域に分けて管理する（図5参照）．OS のコードとデータ，

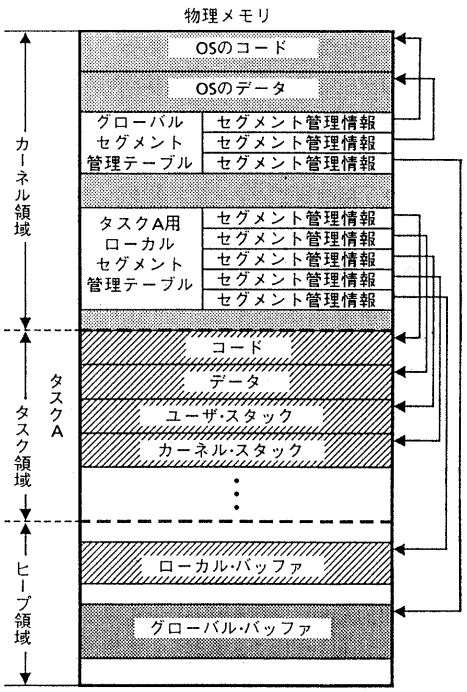


図 5 メモリの構成
Fig. 5 Memory configuration.

各タスクのコード，データ，ユーザ・スタックおよびカーネル・スタックに対しては，別個のセグメントを割り当てることによりメモリ保護を実現する．ヒープ領域に関しては，動的に確保されるバッファに対して，一定のサイズ（現在は4Kバイトとしている）を単位としてセグメントを割り当てる．

OS のコード/データとグローバル・バッファは，システム全体で共有されるグローバル・セグメントとして実現される．また，各タスクのコードなどの領域と，タスク内でローカルに割り当てられたバッファは，ローカル・セグメントとして実現される．OS のコードおよびデータはグローバル・セグメントであるため，OS に対しては，プロトコル処理を行う通常のタスクよりも高い特権レベルを割り当て，タスクが誤ってカーネル領域を破壊しないよう保護している．

OS は，各セグメントのベース・アドレスとサイズ，読出し/書込み/実行などのアクセス属性，および特権レベルなどのセグメント管理情報を保持する．グローバル・セグメントのセグメント管理情報は，OS のデータ内に保持されるグローバル・セグメント管理テーブルにより，また，ローカル・セグメントのセグメント管理情報は，タスクごとに存在するローカル・セグメ

ント管理テーブルにより，それぞれ管理される．

（2） ヒープ領域の管理

OSI 7層ボード用OSは，割り当てたタスクのみにアクセスが許されるローカル・バッファ用のセグメントと，すべてのタスクからアクセス可能なグローバル・バッファ用のセグメントを，各タスクからの要求に応じて割り当てる．グローバル・バッファは，すべてのタスクの仮想アドレス空間上で同一のアドレスを持つように割り当てられる．このため，Unixなどの共有メモリ[8]と異なり，別のタスクが同じグローバル・バッファにアクセスする場合に，ポインタ変換などの特別な処理を必要としない．ユーザ・レベルにおける実際のバッファ割当て/解放に関しては，バッファ用セグメントをさらに分割して管理するライブラリを提供し，セグメント生成のオーバヘッドを削減している．またOSは，各バッファを管理するための情報を利用して，バッファの異常解放を検知して警告メッセージを出力したり，空きバッファのみを含むバッファ用セグメントを解放する処理なども実現している．

OSは，ヒープ領域の残りのサイズが一定以下になると，ホスト・インタフェースなどの特定のタスクに警告を通知する．これにより，ホスト・インタフェース・タスクがホストからのデータ送信要求を待たせたり，トランスポート層やネットワーク層がウィンドウを閉じたりすることによってフロー制御を行い，ボードへのデータの入力を制限し，ヒープ領域のサイズの回復を図ることができる．

4.3 タスク間通信

OSは，タスク間通信機能として，プリミティブの転送を行うためのキューを提供している．1つのキューによるプリミティブの転送は片方向で，キューを生成したタスクのみが，そのキューからプリミティブを受信できる．キューへの送信は複数のタスクに許可されており，送信側タスクはキューの論理名を指定することによって，そのキューへのアクセスが可能となる．

キューを介してやり取りされるプリミティブは，**図6**に示すように，グローバル・バッファにより実現される．OSは，タスクが割り当てたグローバル・バッファをコピーせずに転送するために，キューイング用の特別なヘッダ領域の確保などを行わず，グローバル・バッファの管理領域をタスク間通信用の管理領域として使用している．また，OSは，転送されるバッファの大きさやフォーマットに制約を設けていない．

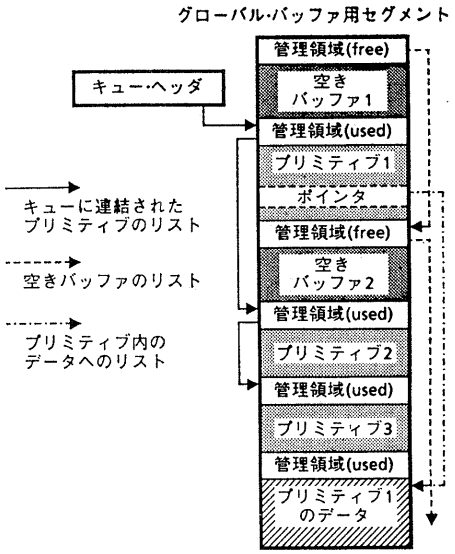


図6 プリミティブ転送のためのバッファとキューの実現方法

Fig. 6 Global buffer and queue for primitives.

このため，プリミティブとして，別のグローバル・バッファへのポインタやリストなどを含む複雑な構造体を直接やり取りすることができる[9]．

4.4 割り込み管理と入出力制御

（1） 割り込み管理

OSI 7層ボード用OSは，各割り込みのエントリ・アドレスを保持する割り込み管理テーブルを使用して，ホスト，回線制御用ハードウェア，ハードウェア・タイマ，およびデバッグ用非同期ポートからの割り込みを処理する．ホストおよび回線制御ハードウェアからの割り込みに対しては，前述のようにホスト・インタフェース・タスクまたは回線インタフェース・タスクが優先的に実行されるようにする．またOSは，一定間隔のハードウェア・タイマ割り込みによってカウントされるソフトウェア・タイマの機能を提供する．

（2） ホスト・インタフェース/回線インタフェース

OSI 7層ボード用OSでは，ホストや回線制御用ハードウェアとの入出力機能をタスクとして実現している．これにより，OSはホストや回線制御用ハードウェアとの入出力手順を知る必要がなく，これらの外部インタフェース・タスクに対して，特定のハードウェア・レジスタなどの物理アドレスに対応する仮想アドレスを通知する機能や，割り込みがあった場合に優先的にスケジューリングする機能などを提供するのみでよい．

4.5 デバッグ機能

OS は，ホストまたはデバッグ用非同期ポートとの間で，キーボード入力やメッセージ出力を行うデバッグ用入出力機能を提供しており，ボード上の各タスクはシステム・コールによってこれらの入出力機能を使用できる．これらの入出力機能を用いて，ボード上に試験用プログラムを実装することも可能であり，ホスト上のユーザ・プログラムが特に存在しない段階においても，ボード上のプログラムを実行・試験することができる．

また OS は，特殊なコマンドによって起動されるデバッガも実装しており，デバッグ用入出力を介して，OS やタスクの各種状態表示，メモリ・ダンプ，レジスタ・ダンプ，ブレーク・ポイントの設定などを行うことができる．

5. 実装結果と性能評価

5.1 OSI 7層ボードの概要

開発した OSI 7層ボードは，大規模で複雑なプロトコル・プログラムを高速で実行するために，32ビット CPU（インテル 80386 SX：クロック周波数 16 MHz）と 8 M バイトの RAM を搭載している．また本ボードは，多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機能を CPU を搭載した回線制御ハードウェアとしてモジュール化している．これまでに，ネットワーク層以下として X.25 および X.32 をサポートする回線制御ハードウェアを持つボードを開発しており，本ボードでは，OSI 7層ボード用 OS の制御のもとで，トランスポート層から応用層までのプロトコル・プログラムを実行する．

5.2 OSI 7層ボード用 OS の実装結果と性能

実装した OSI 7層ボード用 OS は，アセンブラと C言語で作成し，その実行モジュール・サイズは約30 Kバイトである．このうち，初期化とタスクのダウンロードを含むタスク管理とスケジューリングに関する部分が約 25%，メモリ管理に関する部分が約 20%，タスク間通信が約 8%，割り込み管理とタイマが約8%，デバッグ機能に関する部分が約 20% である．

開発した OSI 7層ボード上で動作させた場合の OS の基本機能の処理時間を図7に示す．システム・コールに関しては，OS 内部で特に処理を行わないものを用いて測定し，タスク切替えに関しては，強制的にタスク切替えを行うシステム・コールにより，2つのタスクを交互に切り替えて測定した．バッファ割当てに関しては，割当てと解放を交互に繰り返して測定した．また，タスク間通信に関しては，プリミティブ受信待ち，受信，および受信プリミティブをそのまま通常プリミティブとして送信する3つのシステム・コールを繰り返すタスクを2つ用意し，交互にプリミティブを転送して測定した．それぞれの処理時間は，上述の処理を 10 万回繰り返し，ボードのハードウェア・タイマのカウント数によって1回当たりの処理時間を求めたものである．また，参考のために同一の CPU で 1 K バイトのメモリ・コピーに要する時間も示している．

● システム・コール：	約30μ秒
● タスク切替え：	約80μ秒
● バッファ割当て/解放：	約100μ秒
(割当てと解放の組み合わせ、ローカルとグローバルのバッファで同じ処理時間)	
● タスク間通信：	約190μ秒
(通常プリミティブ、タスク切替えの時間を含む)	
● メモリ・コピー(1Kバイト)：	約140μ秒

図 7 OSI 7層ボード用 OS の性能
Fig. 7 Performance of operating system for OSI 7 layer board.

6. 考　　察

6.1 機能に関する考察

OSI 7層ボード用 OS は，他の汎用的な OS と比較して，制限された機能のみをサポートしている．一方，プロトコル・プログラムの開発・実行を効率的に行うために必須の機能に関しては，それぞれ OS のオーバヘッドが最小限となるように実現方式の工夫を行った．以下では，OSI 7層ボード用 OS と通常の汎用 OS との機能の比較や，OSI 7層ボード用 OS がサポートする機能に関する考察を述べる．

（1） タスク管理とスケジューリングについて

① OSI 7層ボード用 OS は，層/ASE ごとに独立したタスクを対応させ，タスク間通信を契機としたスケジューリングを行う．これにより，各タスクがキューへのプリミティブ受信と送信を含む一連の処理を繰り返すことにより，次の層/ASE に順次，制御を移すというプロトコル処理の流れを自然に実現することができた．

②本 OS では，一定時間ごとのタイマ割り込みによるタスク切替えを行っていない．①で述べたように次に実行が必要なタスクが自然に起動されることや，タスクの数も限られていることなどから，実行中のタスクの処理を特に中断して他のタスクに制御を移す必要性

はないと考えられる．加えて，タスクの動的な生成・消滅を行わない，タスクごとのユーザ管理が必要ない，ファイルなどのタスクに割り当てられた外部の計算機資源が存在しないなどの理由により，一般の汎用OSに比べてタスク管理を単純化でき，OSのオーバヘッドを小さくすることができた．

③層/ASEごとに固定的にタスクを設ける方式では，複数のコネクションを同一のタスクで処理するため，コネクション数の増加によってOS自身のオーバヘッドが増加することはない．また，プリミティブはその到着順に処理されるため，各コネクションに対応するプリミティブを負荷に応じて均等に処理することができると考えられる．

④優先プリミティブの送信時にタスクを切り替える処理は，トランスポート層のAK TPDUなど，ウィンドウを開くために通信相手に短い遅延時間で到着する必要があるデータを転送する場合などに有効であると考えられる．

（2） メモリ管理方式について

①OSI 7層ボード用OSは，セグメントを用いることによって，タスクごとの仮想アドレス空間のサポート，ローカル/グローバル・バッファの割当て機能の提供，メモリ保護などの要求される機能をすべて実現しており，タスク領域のメモリ構成が静的に定まっていることや，ページングを行わないことなどにより，汎用OSよりも簡易化されたメモリ管理機能を持っている．

②本OSは，高速なタスク間通信を実現するために，すべてのタスクからアクセス可能なグローバル・バッファの割当て機能をサポートする．しかし，一般に，このような共有メモリを用いたタスク間通信においては，他のタスクに送信済のバッファを誤って解放するといった，複数のタスクにまたがるプログラム誤りが混入しやすく，プログラムのデバッグが困難となる問題点が存在する．これに対して，本OSでは，送信済のグローバル・バッファの誤った解放などを検出できるように，以下の機能を設けた．

- バッファの二重解放や不当なバッファ・アドレスの解放などを検出する機能
- 各タスクがバッファを割当て/解放した時点で，そのアドレスとサイズを記録する機能
- すべてのバッファの管理領域をトレースし，管理領域の破壊や空きバッファ・リストの異常などを検出する機能

これらの機能により，実際にプロトコル・プログラムの実行時にグローバル・バッファの二重解放や解放もれなどを検出しており，デバッグの助けとなった．

③グローバル・バッファがどのタスクからも同一の仮想アドレスでアクセス可能であるため，ポインタを多用した複雑なデータを，タスク間で共有することができる．この利点を生かして，筆者らは，各層/ASEにおけるPDUの作成/解析時にユーザ・データのコピーを伴わないバッファ制御方式を実現し，ユーザ・レベルのライブラリとして提供している[10]．

（3） タスク間通信方式について

①OSI 7層ボード用OSは，タスク間通信として非同期のキューのみをサポートし，セマフォなどの同期機構は持っていない．各層/ASEごとにタスクを設ける方式では，一般に各タスクは送信済のデータに再びアクセスする必要性がないため，本OSが提供するタスク間通信だけでも十分であると考えられる．

②グローバル・バッファの割当て/解放のための管理情報とキュー用の管理情報を一体として設計することにより，効率的なプリミティブの転送を実現できた．

（4） 入出力制御について

一般の汎用OSでは，割り込み処理が必要な点や，プログラム保護の観点から，入出力制御機能をカーネル内に組み込む場合が多い．一方，OSI 7層ボード用OSでは，ホストおよび回線制御ハードウェアとインタフェースする部分をタスクとして実現している．ホスト・インタフェースの処理は，対象とするホストやボード上のプロトコルの種類によって変化し，また，回線インタフェースの処理は，接続するネットワークの種類によって異なる．このため，これらの機能をカーネルとは分離し，開発を容易にした．

6.2 関連する研究との比較

（1） 性能に関する比較

①実装したOSの性能を，汎用的なOSであるMachとAmoebaの性能と比較した．IBM PC RT（メモリ・コピーの時間：180 μ 秒/Kバイト）上のMachオペレーティング・システムの性能は，システム・コール：149 μ 秒，タスク切替え：137 μ 秒，タスク間通信：1.5 m 秒と報告されている[11]．また，Sun-3/60上のAmoebaオペレーティング・システムのプロセス間通信は，約0.5 m 秒と報告されている[12]．これらの点から，実装したOSI 7層ボード用OSは，既存の汎用OSと比較して，特にタスク間通信において十分高速な性能を実現しているといえる．また，システ

ム・コールやタスク切替えのオーバヘッドも十分小さいといえる.

②プロトコル処理の効率的なサポートを目的としたOS の例として, x-Kernel が存在する[13]. Sun-3/75 (メモリ・コピーの時間: 250 μ秒/K バイト) 上の x-Kernel の性能は, コンテクスト切替え: 38 μ秒, プロセス生成: 135 μ秒, ユーザ・モードからカーネル・モードへの遷移/復帰 (システム・コールに相当): 20 μ秒, カーネルからのユーザ・プロセス・ルーチンの直接呼び出し/復帰 (upcall と呼ばれる): 254 μ秒と報告されている[13]. したがって, 2つのユーザ・プロセス間でメッセージを転送する場合のプロセス間通信に相当する処理時間は, カーネル・モードへの遷移と upcall を合計した 274 μ秒となる. 以上の点から, OSI 7層ボード用 OS は, x-Kernel と比較しても同程度の性能が得られていると考えられる.

(2) 機能に関する比較

(i) x-Kernel との比較

①x-Kernel では, プロトコル処理を行うプログラムをカーネル内に組み込み, 個々の PDU に対応するメッセージごとにスレッドを割り当て, 1つのメッセージを層間でやり取りする際のプロセス切替えを避ける方式を採用している. しかし, このような方式ではメッセージ単位ですべての層のデータにアクセス可能であるため, ある層の誤りが他の層に波及しやすいと考えられる. OSI プロトコルは機能が豊富で複雑なため, そのプログラム規模が大きくなりがちである. このようなプログラムをボード上で効率的に開発・デバッグするために, OSI 7層ボード用 OS においては, 各層/ASE のプロトコルをそれぞれ独立したタスクとして互いに保護を行う方式を採用した.

②x-Kernel は, メッセージの分配など, 複数の上位層プロトコルをサポートするための機能を提供しているが, OSI 7層ボード用 OS ではこのような点は特に考慮していない. OSI では, 上位層の選択方法が, 各層のアドレスや応用タイトルなどによるさまざまな方式が考えられる. そこで, このような振り分け機能は, 必要に応じてプロトコル処理を行うタスクに実現させることとした.

③x-Kernel では, メッセージごとにスレッドを割り当てることにより, 層間のやり取りにおいてデータ・コピーを避けている. OSI 7層ボードでは, 前述のようにグローバル・バッファを利用した PDU 作成/解析用のバッファ制御ライブラリによって, コピーを伴わない転送を実現している[10].

(ii) 他の OSI ボード用の OS との比較

OSI プロトコルをサポートする他の通信ボードとして, ACSE までをサポートする HP OSI Express Card[4]や, セション層までをサポートする OSI 通信ボード[2),5)]が報告されている. HP OSI Express Card は, LAN を対象としてプロトコルを固定的に定めており, これに最適化された実行環境を持っている[14]. 各層のプロトコル処理は, プリミティブごとに手続き呼出しを行うディスパッチャによって実行され, 層ごとの仮想アドレスやメモリ保護などはサポートしていない. また, セション層までをサポートする OSI 通信ボードでは, 各層をモニタから呼ばれるサブルーチンとして実現する単純な方法を採用している.

これらのボードでは, 搭載プロトコルが固定されているため, プロトコル・プログラムの開発を容易化するための機能は特にサポートしていない. これに対して, OSI 7層ボード用 OS においては, 複数のネットワークや応用層プロトコルに対応することを目的として, OS の独立性や層ごとの独立な開発を実現するために, 各層/ASE をタスクとして実現し, そのメモリ保護を行っている.

7. おわりに

本稿では, OSI の7層すべてのプロトコルをサポートする通信ボードのためのオペレーティング・システムの実装と評価について報告した. 本 OS は, OSI プロトコルの効率的なサポートを目的としており, OSI の通信処理に最適化されたものである. OS は, セグメント単位の仮想アドレス空間とメモリ保護機能, およびダウンロード機能のサポートによって, 各層のプログラムを独立に開発することを可能とし, また, グローバル・バッファを用いた高速なタスク間通信と, プロトコル処理を考慮したタスクの実行制御を行うことによって, プロトコル処理の効率的な実行を実現している.

これまでに, 本 OS のもとで動作するトランスポート層から FTAM (ファイル転送, アクセスおよび管理) を実装し, 2 Mbps の回線上で約 1.5 Mbps のスループットを得ている[15]. 今後さらに, OSI 7層ボード用 OS のもとで各種のプロトコル処理プログラムを実装し, その評価等を行っていく予定である.

謝辞 本論文の作成にあたりご検討いただいた KDD 研究所浦野所長に感謝します.

参考文献

1) 鈴木健二，加藤聰彦，小花貞夫：パケット通信プロトコルX.25とX.32のパーソナルコンピュータへの実装と評価，信学論，Vol. J 73 B-I, No. 6, pp. 579-587 (1990).
2) 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).
3) 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，p. 3-78 (1992).
4) Johnson, W. R.: An Overview of the HP OSI Express Card, *HP Journal*, Vol. 41, No. 1, pp. 6-8 (1990).
5) 瀬戸康一朗，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，情報処理学会研究会報告，Vol. 91, No. 83, pp. 117-122 (1991).
6) 井戸上彰，加藤聰彦，鈴木健二：汎用OSI 7層ボードの構成，第42回情報処理学会全国大会論文集，7T-1, pp. 1-251-252 (1991).
7) 井戸上彰，加藤聰彦，鈴木健二：OSI 7層ボードにおける通信ボード用OS，情報処理学会研究会報告，Vol. 92, No. 52, pp. 79-86 (1992).
8) Bach, M. J.: *The Design of the UNIX Operating System*, Prentice-Hall, Inc. (1986).
9) 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，4L-11, pp. 1-167-168 (1992).
10) 井戸上彰，加藤聰彦，鈴木健二：汎用OSI 7層ボードにおけるプロトコル・データ単位用バッファの制御方式，第44回情報処理学会全国大会論文集，4L-12, pp. 1-169-170 (1992).
11) Duchamp, D.: Analysis of Transaction Management Performance, *Proc. of 12th ACM Symposium on Operating Systems Principles*, pp. 177-190 (1989).
12) Tanenbaum, A. S. et al.: Experiences with The Amoeba Distributed Operating System, *Comm. ACM*, Vol. 33, No. 12, pp. 46-63 (1990).
13) Hutchinson, N. C.: The x-Kernel: An Architecture for Implementing Network Protocols, *IEEE Trans. Softw. Eng.*, Vol. 17, No. 1, pp. 64-76 (1991).
14) Dean, S. M., Kumpf, D. A. and Wenzel H. M.: CONE: A Software Environment for Network Protocols, *HP Journal*, Vol. 41, No. 1, pp. 18-28 (1990).
15) 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).

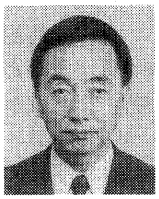
(平成5年2月18日受付)
(平成6年1月13日採録)

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．

小野 欽司（正会員）

昭和14年生．昭和37年東京大学理学部物理卒業．同年国際電信電話(株)入社，研究所にて衛星通信の研究に従事．昭和45年スタンフォード大学留学，電気工学修士修了，コンピュータ通信の研究に転じ，研究所長を経て平成5年より文部省学術情報センター教授就任現在に至る．ITU-TSAG副議長，TTC標準化会議前議長．工学博士．昭和59年度科学技術庁長官研究功績者表彰，平成4年度電子情報通信学会業績賞受賞．IEEE SM.