マルチメディア通信と分散処理56－11
（１９９２．　７．　10）

OSI 7層ボードにおける通信ボード用OS

井戸上 彰　　加藤 聰彦　　鈴木 健二
国際電信電話(株) 研究所

筆者らは、パソコンやワークステーションを対象として、OSIの7層すべてのプロトコルをサポートするOSI 7層ボードの開発を行っている。OSI 7層ボードは、応用層までの様々なプロトコルを汎用的にサポートすることを目的としており、複雑で大規模なプログラムを高速で実行するために、32ビットCPUを搭載している。OSI 7層ボード上で実行される多種多様なプロトコル処理プログラムの開発/実行を効率的に行うためには、通信ボード用OSが必要となる。本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、各層のプログラムの独立開発を可能とし、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、メッセージの優先度に基づく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。本稿では、OSI 7層ボード用OSの実装とその評価について述べる。

Operating System for OSI 7 Layer Board

Akira IDOUE, Toshihiko KATO and Kenji SUZUKI
KDD R & D Laboratories
2-1-15, Ohara, Kamifukuoka-shi, Saitama, 356 Japan

We have been developing the OSI 7 layer board, which supports all seven layer protocols of OSI, for personal computers and workstations. The OSI 7 layer board is intended to support various protocols up to the application layer, and equipped with a 32-bit CPU to execute the complex and large-scale programs. In order to develop and execute the software for the board efficiently, it is needed to implement an operating system for the board. The operating system will offer the down-loading of program modules and memory protection between program modules to enable protocol program modules to be developed independently. It will also support the efficient task scheduling and inter-module communication for the purpose of the high performance execution. This paper presents the implementation and evaluation of the on-board operating system for the OSI 7 layer board.

1. はじめに

筆者らは、パソコンやワークステーションの拡張ボード上で、OSI(開放型システム間相互接続)のプロトコルをサポートするOSI通信制御ボードの開発を進めている[1]。現在、OSIの7層すべてのプロトコルをサポートする汎用OSI 7層ボード[2,3]の開発を行っている。

汎用OSI 7層ボードは、多種のネットワークに柔軟に対応し、応用層までの様々なプロトコルを、高いスループットで実行することを目的としている。このため、ボードのハードウェアを、対象とするネットワークに依存した下位モジュールと、ネットワークとは独立な上位モジュールとにモジュール化する構成を採用している。

上位モジュールでは、大規模で複雑なプロトコル処理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80386SX)を搭載し、様々なプロトコルに対応するために、プログラムをホストからダウンロードする機能を持たせている。

そこで、上位モジュール上で実行される多種多様なプロトコル処理プログラムの開発/実行を効率的に行うために、通信ボード用OSが必要となる。本OSは、独立に作成された複数のプログラム・モジュールをロードし、モジュール間のメモリ保護を実現する機能を提供する。また、OSは、高い性能を得るために、プログラム・モジュールのスケジューリングや、モジュール間のデータのやり取りを効率的にサポートする。

本稿では、このOSI 7層ボード用OSの実装とその評価について述べる。以下、2. ではOSIプロトコルのサポートの観点から、OSに対する要求条件を述べる。3. ではOSI 7層ボード用OSの持つ機能の概要を示し、4. で各機能の実現方式の詳細について述べる。5. ではOSの実装結果と評価を示し、6. でそれらに関して考察する。

2. OSI 7層ボード用OSに対する要求条件

OSI 7層ボード用OSは、OSIプロトコルを処理するプログラムの開発と実行を効率的にサポートすることを目的としている。このため、OSの開発にあたっては、OSIプロトコルの特徴を考慮し、次のような要求条件を満足する必要がある。

(1) OSのオーバヘッドの削減

伝送路の高速化に伴い、OSI通信に対して高い性能が要求されている。このため、OSI 7層ボード用OSは、OSIプログラムを高速に実行するために必要となる機能のみを実現し、OS自身のオーバヘッドを削減する必要がある。

(2) 層ごとに独立なプログラム開発

OSIでは、各層および各応用サービス要素(ASE)に対して通信プロトコルが定められており、通常各プロトコルを個別のプログラマに独立に開発させ、それらを組み合せてOSIボードに搭載することが要求される。このような層ごとのプログラム開発/保守を容易にするためには、OSは、各層に対応するプログラム・モジュールを独自のアドレス空間を持つタスクとして実現し、アドレス空間に対するメモリ保護機能を持つことが不可欠である。さらに、各層のプログラムがOSの内部情報を誤って破壊しないよう、OSに対する保護を実現する必要がある。

(3) 複雑なデータ構造をサポートする高速な層間の通信

OSIの各層の間でやり取りされる情報は、サービス・プリミティブとして定義されている。これらのサービス・プリミティブは、可変長やオプションのパラメータを多く含んだ複雑な構造を有する。このためOSは、ボード上で実行される各層のプロトコル処理プログラムの間で、構造体、共用体、ポインタなどを含む複雑なデータ構造の情報を、高速に転送する必要がある。

(4) プロトコル処理を考慮したスケジューリング

OSIの1つの層の処理の流れは、上位層または下位層からのプリミティブを待ち、受信したプリミティブと現在の状態に応じた処理を行い、上位層または下位層へプリミティブを出力して再び入力を待つという定形的な手順の繰り返しである。一方、優先データを含むプリミティブなどについては、その処理のプライオリティを高めることができる必要がある。そこで、コンテキスト・スイッチのオーバヘッドを避けるため、通常は1つの層のプログラムが一連の処理を終了するまで、他の層に制御を渡さないが、優先的な処理が必要な特定の層間の通信に対しては、それを契機として即時

に他の層に制御を渡すといったプロトコル処理の内容を考慮したスケジューリングが必要である。

(5) 通信プログラムのダウンロード機能

さまざまな業務に対応するOSIプログラムをボード上で実行させるために、複数のプログラムモジュールを、自由に組み合せて、ホスト計算機からダウンロードする機能を持つ必要がある。

(6) ボード上のプログラムに対するデバッグ

各層のプロトコル処理プログラムは、開発途上において、ボード上で実行/試験する必要がある。そこで、OSには、プログラムをボード上で動作させてデバッグを行う機能が要求される。

3. 機能概要

2.で述べた要求条件に基づき、OSI 7層ボード用OSの機能、およびその上でのOSIプログラムの実装方法を定めた。その概要を以下に示す。

① 本OSはオーバヘッドを削減するために、タスク管理やメモリ管理などについて、使用するCPUであるインテル80386に最適化された実現方法を採用する。OSは80386の持つプロテクト仮想アドレス・モードの機能を前提とし、タスクごとのコンテキスト情報が格納されるタスク管理テーブルや、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セグメント管理テーブル等のCPU固有の機能を用いる。

② 各層および各ASEのプログラム・モジュールを、独立した仮想アドレス空間を持つタスクとして実現する。さらに、80386の持つセグメンテーション機能を用いて、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、メモリ保護を行う。また、80386がサポートする特権レベルを使用し、OSには最高特権レベルを、通常のユーザ・タスクには最低特権レベルを割り当て、カーネルの保護を行う。

③ 各層に対応するタスクの間でやり取りされる、複雑な構造を持つサービス・プリミティブを、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・セグメント上に実現する。グローバル・バッファ上のプリミティブへのポインタを含んだメッセージを、メッセージ・キューにより受け渡すことにより、高速なタスク間通信を提供する。

④ プロトコル処理の内容を考慮し、メッセージの送受信を要求した時点を、タスク・スイッチを行う契機とする。また、プロトコル処理に対応する優先度に基づくスケジューリングを可能とするために、タスク間通信のメッセージに優先度を設け、優先メッセージを送信した時点で受信タスクに制御を移すことを可能とする。

⑤ ボード上で動作する複数のプログラム・モジュールを、初期化時にホスト計算機からダウンロードする機能を提供する。

⑥ デバッグ・メッセージのホスト計算機への出力などのデバッグ機能をサポートする。

以下では、これらの機能の実現方式の詳細について述べる。

4. 機能の詳細

OSI 7層ボード用OSの構成を図1に示す。OSは、タスク管理、スケジューリング、メモリ管理、タスク間通信、タイマ、タスクのダウンロード等の機能を持ち、ユーザ・レベルの各タスクは、システム・コールを通してタスク間通信やタイマなどのOSの提供する機能を利用する。

各機能の特徴を以下に示す。

4.1 タスク管理とスケジューリング

(1) タスク管理のためのデータ構造

7層ボードOSは、タスク・コントロール・ブロック(TCB)を用いて、ボード上で動作するタスクを管理する。TCBは、タスクの論理名、タスクID、タスク状態、メモリの割り当て状況など、OSによるタスク管理に必要となる情報と、80386が使用するタスク管理テーブルへのポインタを保持する(図2参照)。

タスク管理テーブルには、汎用レジスタなどのコンテキスト情報や、タスクごとのセグメント管理テーブルへのポインタが格納される。このように、OSが80386の提供するタスク管理テーブルを直接利用することにより、コンテキスト・スイッチや内部処理のオーバヘッドを削減している。

(2)タスクの状態管理とスケジューリング

各タスクは、現在CPUが割り当てられている「実行中」、実行可能でCPUの割り当てを待って

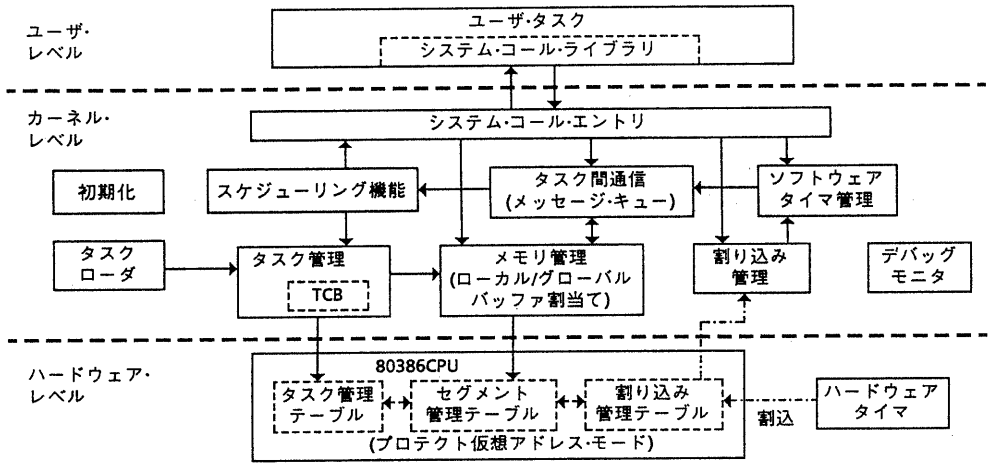

図1　OSI 7層ボード用OSの構成

```
● Next/Prev-TCBポインタ      /* タスク状態ごとのTCBリスト用*/
● タスク名                   /* タスクの論理名 */
● タスクID                   /* タスクのID */
● 特権レベル                 /* CPUの特権レベル(0/3) */
● タスク状態                 /* 実行中、実行待ち、etc. */
● メモリ割当てリスト         /* バッファ管理用 */
● キュー割当てリスト         /* 受信メッセージ・キュー */
● タスク管理テーブルへのポインタ    /* 80386固有 */
```

図2　TCB内のデータ

いる「実行待ち」、タスク間通信用のキューへのメッセージ到着を待っている「待機中」の3つの基本状態をとる。さらに、実行中のものに関しては、ユーザ・タスク自身のコード領域を実行中であることを示すユーザ・モードと、システム・コールを通してOSのカーネル内のコードを実行中であることを示すカーネル・モードに分けられる。また、プロトコル処理を考慮したスケジューリングをサポートするために、タスク間でやりとりされるメッセージに2レベルの優先度を設けており、実行待ち状態のタスクは、受信したメッセージの優先度に応じて通常優先度と高優先度の2つの状態に分けられる。

このようなタスクの状態に基づき、7層ボード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3参照)。

① 実行中のタスクが、受信キューへのメッセージ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、通常メッセージの送信システムコールを発行した場合は、そのメッセージの受信タスクが待機中であれば、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、優先メッセージの送信システム・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、優先メッセージの受信タスクが、待機中または通常優先度の実行待ちであれば、それ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。

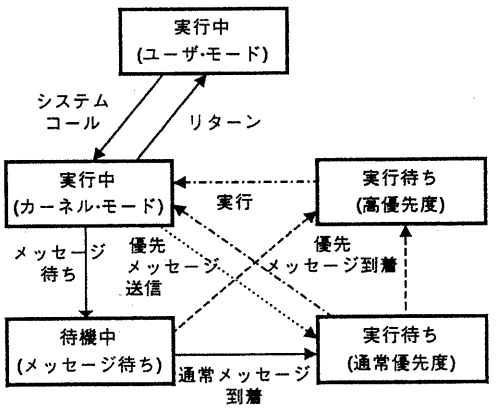

図3　タスクの状態遷移

4.2 メモリ管理

(1) セグメントの割り当て

前述のように、OSI 7層ボード用OSは80386のセグメンテーション機能を用いたメモリ管理を行う。80386は、各セグメントを、そのベース・アドレスとサイズ、読み出し/書き込み/実行などのアクセス属性、および特権レベルなどを記述したセグメント管理テーブルによって割り当てる。セグメント管理テーブルには、システム全体で共有されるグローバル・セグメント管理テーブルと、タスクごとに定義可能なローカル・セグメント管理テーブルが存在する。

OSは、それ自身のコードとデータ領域、および共有バッファ等をグローバル・セグメント管理テーブルを用いて管理し、すべてのタスクからアクセス可能としている。

一方、各タスクのコード、静的データ、ユーザ・スタックおよびカーネル・スタックの各領域に対しては、ローカル・セグメント管理テーブル上で独立したセグメントを割り当て、タスクごとの仮想アドレスとメモリ保護を実現している(図4参照)。OSは、タスクのロード時に、そのタスクに対応するローカル・セグメント管理テーブルを動的に生成し、テーブルの各エントリに記述されるセグメントのベース・アドレスを定める。

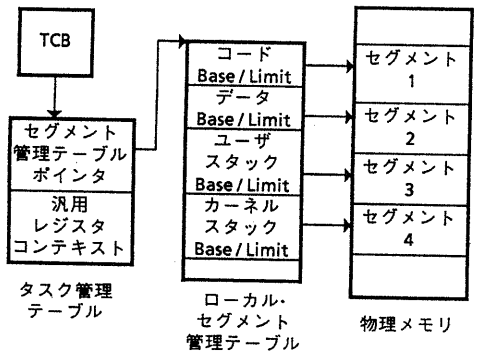

図4 タスクごとのメモリ・セグメント

(2) フロー制御機能を持つヒープ領域の管理

OSI 7層ボード用OSは、バッファ用のセグメントのためのヒープ領域を管理し、タスクごとに保護されるローカル・バッファ・セグメントと、すべてのタスクで共有されるグローバル・バッファ・セグメントを提供している。ローカル・バッファは、それを割り当てたタスクのみにアクセスが許されており、互いに保護されている。一方、グローバル・バッファは、すべてのタスクからアクセス可能であり、後述のタスク間通信のために使用することができる。

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メッセージを通知する。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、下位インタフェース・タスクが下位モジュールのネットワーク層にウインドウを閉じさせ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てヒープ領域のサイズの回復を待つことができる。ヒープ領域の残りのサイズが一定以上に回復すると、警告解除メッセージを通知して先のフロー制御を解除させる。これらのメッセージは、タスク間通信用のキューを介して、OSからユーザ・タスクに通知される。

4.3 タスク間通信

タスク間通信機能としてキューを介したメッセージの転送を提供している(図5参照)。

(1) メッセージ・キューの機能

各キューは、送信側と受信側のタスクが同じ論理名を指定することによって動的に割り当てられる。1つのキューによるメッセージの転送は片方向のみで、受信側タスクは1つに制限されるが、送信側としては複数のタスクが許される。また、ある層に対応するタスクが、その上位層および下位層のタスクとの間に設けた複数のキューからの受信を同時に処理することを可能とするために、キューの送信/受信システム・コールはノンブロック(非同期)型とし、割り当てたすべての受信キューへのメッセージ到着を待つためのシステム・コールを用意している。

(2) メッセージ用バッファ

キューを介してやりとりされるメッセージ用のバッファとしては、前述のグローバル・バッファを使用する。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じアドレスを持つものとして割り当てられるため、通常のOSの共有メモリ[7]と異なり、メッセージ用の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

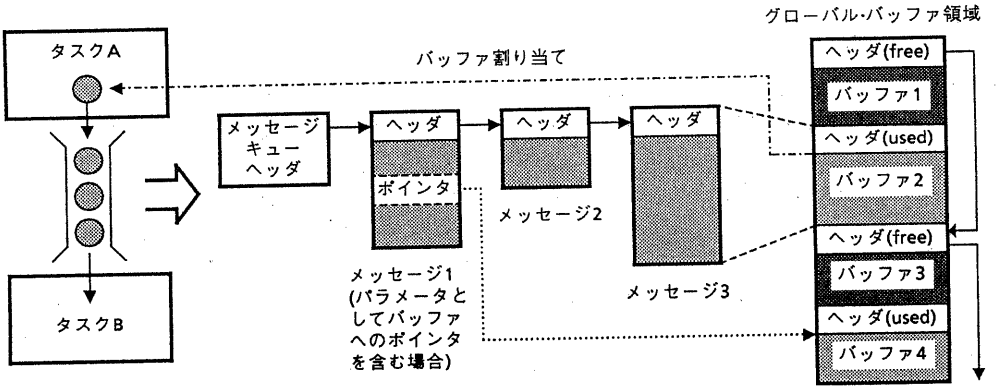

図5　メッセージ・キューとメッセージ・バッファ

理を必要としない。従って、割り当てたバッファを直接任意のタスクに送信することができ、また受信したバッファをそのまま処理することが可能である。また、OSI 7層ボード用OSは、転送されるグローバル・バッファの大きさや、その上に作成されるメッセージのフォーマットにも特に制約を設けていないため、メッセージとして、さらに別のグローバル・バッファへのポインタを含む複雑な構造体を直接やりとりすることができる[5]。

カーネル内でバッファをキューイングするために必要なヘッダ領域は、各バッファの割当て/解放のための管理用のヘッダ部分をそのまま利用しており、キューイングのために特別なヘッダ領域を確保することはない。

4.4 タイマ

OSは一定時間ごとのハードウェア・タイマからの割り込みによってカウントされるソフトウェア・タイマを提供している。ソフトウェア・タイマは、各タイマのタイマID、要求元のタスクID、およびタイムアウトまでの残り時間などを記述したデータ・バッファのリストとして実現している。OSはタイムアウト(残り時間が0)となったバッファを、そのまま各タスクの受信キューに接続し、タイムアウトをメッセージとして通知する(図6参照)。

4.5 タスクのダウンロードと起動

7層ボードOSは、各層のプロトコル処理プログラムに対応するユーザ・タスクを複数組み合せてホスト計算機からダウンロード可能としている。このためのプログラムの開発とダウンロードは次のように実現される(図7参照)。

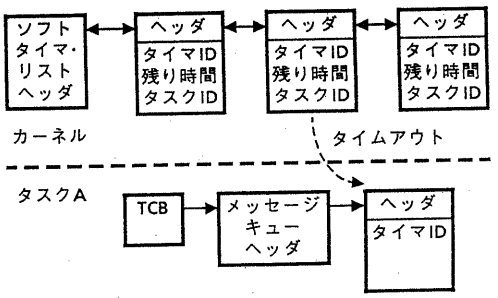

図6　ソフトウェア・タイマ

① 既存の80386用のコンパイラ/リンカによって生成される実行ファイルから、7層ボード用プログラム・モジュール・ジェネレータを用いて個々のプログラム・モジュールを作成する。

② ロード・モジュール・ジェネレータにより、複数のプログラム・モジュールを組み合せて1つのロード・モジュール・ファイルを作成する。各プログラム・モジュールはリロケータブルであるため、自由に組み合わせることができる。

③ ボードのROM上には簡易なブート・ローダのみが格納されており、ボードの初期化時はブート・ローダが最初にタスク・ローダを含む7層ボードOSモジュールをダウンロードする。

④ OSに制御が渡り、タスク・ローダがロード・モジュールをダウンロードする。タスク・ローダは、これに含まれる個々のプログラム・モジュールをボードのRAM上に展開し、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、タスクとして登録する。

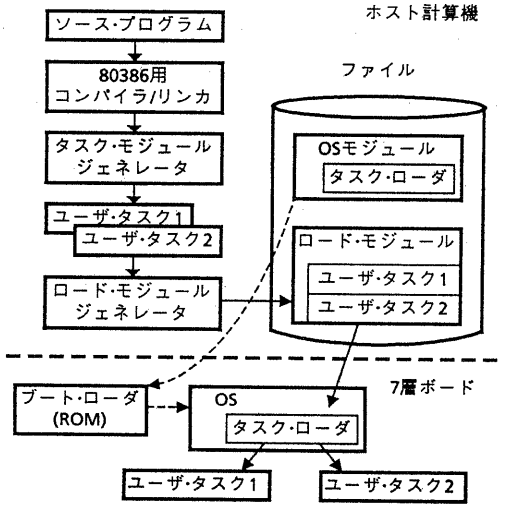

図7　プログラム・モジュールのダウンロード

⑤ すべてのプログラム・モジュール(タスク)の登録後、OSはホスト計算機からの起動コマンドを待つ。起動コマンドにより、スケジューラが動作を開始して各タスクが順次実行される。なお、ボード上で起動後に動的にタスクを生成することはサポートしていない。

4.6 デバッグ機能

各層のプロトコル処理プログラムの開発途上において、それらをボード上で実行しながらデバッグすることを可能とするため、OSはデバッグ機能をサポートしている。

各プログラムの単体デバッグなどのために、ホスト計算機からの直接キーボード入力や、ホスト計算機へのデバッグ・メッセージの表示をシステム・コールとして提供している。また、ボード上のタスクの動作状況やメモリの使用状況等を調べるために、簡易なデバッグ・モニタを備えている。デバッグ・モニタはホスト計算機からの特殊なコマンドによって起動され、OSの各種状態表示、メモリ・ダンプ、レジスタ・ダンプ等のコマンドを提供している。

5. 実装結果と評価

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25Kバイトである。このうち、OSの初期化に関する部分の割合が約10%、メモリ管理に関する部分が約20%、タスクのロードと実行管理が約15%、デバッグ・モニタに関する部分が約20%、タスク間通信が約8%、割り込み管理とタイマが約8%である。

クロック周波数が16MHzの80386SXを使用した場合のOSの基本機能の処理時間を図8に示す。システム・コールに関しては、OS内部で処理を行わないものを用いて測定し、タスク・スイッチに関しては、強制的にタスク・スイッチを行うシステム・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。また、タスク間通信に関しては、メッセージ受信待ち、受信、および受信したメッセージをそのまま通常メッセージとして送信する3つのシステム・コールを繰り返すタスクを2つ用意し、交互にメッセージ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1K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・スイッチ：	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)	
● タスク間通信：	約190μ秒
(通常メッセージ、タスク・スイッチの時間を含む)	
● メモリ・コピー(1Kバイト)：	約140μ秒

図8　OSI 7層ボード用OSの性能

6. 考察

(1) 性能について

IBM PC RT(メモリ・コピーの時間：約180μ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：149μ秒、タスク・スイッチ：137μ秒、タスク間通信：1.5m秒と報告されている[8]。従って実装したOSは、既存オペレーティングシステムと比較して、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。

(2) スケジューリング方式について

OSは一定時間ごとのタイマ割り込みによるタスク・スイッチは行わず、2つの優先度を持つメッセージのやりとりを契機としたスケジューリング方式を採用している。このような比較的単純なスケジューリング方式で充分であった理由は、以下の通りである。

- OSI 7層ボードでは、各層およびASEごとに独立したタスクを対応させており、各タスクは、プリミティブ受信のためシステム・コールを発行し、受信したプリミティブに応じた処理を行うという手順を繰り返す。このような手順により、処理が必要なタスクが自動的に実行待ち状態となって順次起動される。
- OSIプロトコルにおいては、ある特定の層に対してよりも、優先データなどのPDUや、1つのコネクションなどに対して処理のプライオリティをあげることが要求される。タスク間通信のメッセージに優先度を設け、高優先度メッセージの送信時にタスクを切り替えることにより、このような複数の層にまたがってやりとりされるデータに対する優先処理が可能となる。

(3) タスク間通信方式について

OSはタスク間通信としてグローバル・バッファを用いた非同期のメッセージ・キューのみをサポートしている。本方式は、バッファ・アドレスの変換なども必要ないため、処理のオーバヘッドも小さく、バッファを複数のタスクにまたがって持ち回ることも可能である。この利点を生かして、筆者らは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ている[6]。

各層のプロトコルごとにタスクを設ける方法では、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。

(4) ホスト/下位インタフェースの分離について

OSI 7層ボードでは、ホストおよび下位モジュールとインタフェースする部分を別タスクとして実現している。割り込みのハンドリングが必要な点や、特権レベルによる保護の観点からは、これらをOSのカーネル内部のI/O機能とする方法も考えられる。しかし、ホスト・インタフェースの処理は、対象とするホスト計算機とそのインタフェース方式、およびボード上の最上位のプロトコルの種類などによって変化する。また、下位インタフェースの処理は、接続する下位モジュールの種類によって異なる可能性がある。このため、これらの機能をカーネルとは分離させ、必要に応じて独立に開発可能としている。

7. おわりに

本稿では、OSI 7層ボード用OSの機能とその実装結果について報告した。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、通信ボード用に最適化したものである。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、各層のプログラムの独立開発を可能とし、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、メッセージの優先度に基づく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、パケット網と電話網に応じた物理回線インタフェースとネットワーク層以下のプロトコル(X.25/X.32)をサポートする下位モジュールを開発しており、上位モジュール上に本OSのもとで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、2Mbpsの回線上で約1.5Mbpsのスループットを得ている。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。最後に日頃御指導頂くKDD研究所小野所長、浦野次長に感謝します。

参考文献

[1]: A. Idoue, et. al., "Design and Implementation of OSI Communication Board for Personal Computers and Workstations," Proc. of ICCC '90, Nov. 1990.

[2]: 井戸上, 加藤, 鈴木, "OSI 7層ボードの設計," 情報処理学会, マルチメディア通信と分散処理研究会, 46-11, July 1990.

[3]: 井戸上, 加藤, 鈴木, "汎用OSI 7層ボードの構成," 第42回情処全大, 7T-1, Mar. 1991.

[4]: 井戸上, 加藤, 鈴木, "汎用OSI 7層ボード用カーネルの実装," 第43回情処全大, 2T-12, Oct. 1991.

[5]: 加藤, 井戸上, 鈴木, "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回情処全大, 4L-11, Mar. 1992.

[6]: 井戸上, 加藤, 鈴木, "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回情処全大, 4L-12, Mar. 1992.

[7]: M. J. Bach, "The Design of the UNIX Operating System," Prentice-Hall, inc., 1986.

[8]: D. Duchamp, "Analysis of Transaction Management Performance," Proc. of 12th ACM Symposium on Operating Systems Principles, Dec. 1989.