1D－3　OSI 7層ボード用プロトコル・プログラムのUnixへの移植に関する一検討

井戸上 彰　藤長 昌彦　加藤 聰彦　鈴木 健二
KDD研究所

1. はじめに

筆者らはこれまでに、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ている[1]。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本体にプロトコル処理の負荷をかけずに、各種OSI通信システムを簡便に実現することができる。一方、近年ワークステーションのCPUが高速化し、搭載メモリ容量も増大していることから、ワークステーション本体上のソフトウェアとしてOSIプロトコルを実現する要求も高まっている。そこで本稿では、独自の7層ボードOS[2]のもとで実行されるOSI 7層ボード用のプロトコル・プログラムを、Unixに移植する方法について検討した結果を述べる。

2. 7層ボード用プロトコル・プログラムの移植性

OSI 7層ボード用のプロトコル・プログラムは、7層ボードOS以外に移植することを考慮し、次のような特徴を有している。

- 各層/ASE(Application Service Element)ごとに独立したタスクとして開発・実行されるが、同一アドレス空間上でも動作可能なように、関数名などの重複を避けている。
- 整数のバイト順序等、CPUによって異なるデータ表現に依存しない。

このため、移植先において以下の機能を実現することにより、プロトコル・プログラム自身を変更することなく移植することが可能である。

- プリミティブやPDUは、すべての層/ASEのプログラムからアクセス可能とする。
- 7層ボードOSが提供するタスクのスケジューリング機能と、バッファの割当て/解放、タスク間通信用のキューの生成とキューへの送受信、タイマの開始/停止とタイムアウト通知、現在時刻の取得などのシステム・サービスを、プロトコル・プログラムが利用可能とする。
- 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を実現するPDUバッファ制御ライブラリ[3]を必要に応じて最適化する。

プロトコル・プログラムのUnix上への移植にあたっては、共有バッファを実現するため、ユーザ・プロセスに含まれるライブラリとして実現する形態と、カーネル内に組み込む形態を検討することとした。以下では、それぞれの形態について、7層ボードOSの機能やバッファ管理などの実現方式に関する検討を述べる。

3. ライブラリとして実現する形態

本形態は、以下のように実現される(図1)。

- 7層ボードOSが提供する実行環境と同等の機能を7層ボードOS模擬機能として実装する。スケジューリングやキュー操作に関するシステム・サービスは、OS模擬機能が7層ボードOSと同一の機能を実現する。
- タイマ管理や時刻取得などに関するシステム・サービスは、OS模擬機能がUnixのシステム・サービスを利用して実現する。
- プリミティブやPDU用の共有バッファは通常のmalloc()を用いて容易に実現可能である。このため、7層ボード用のPDUバッファ制御ライブラリをそのまま利用できる。
- 各層/ASEのスケジューリングを行うために、アプリケーション・プログラムも7層ボードOS模擬機能を用いて作成し、その制御のもとで動作させる必要がある。
- デバイス・ドライバとの間で転送されるデータ形式は、プリミティブのデータ型とは独立に規

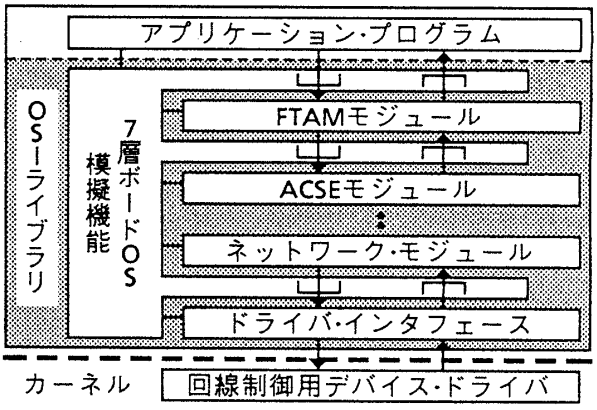

図1　ライブラリによる実現形態

A Study on Porting Protocol Programs for OSI 7 Layer Board to Unix Operating System
Akira IDOUE, Masahiko FUJINAGA, Toshihiko KATO and Kenji SUZUKI
KDD R & D Laboratories

定される。このため、デバイス·ドライバに対するデータとプリミティブとの変換を行なうドライバ·インタフェースをライブラリ内に設ける。

4. カーネル内に組み込む形態

Unixでは、カーネル内のドライバの実装において、複数の処理モジュールを階層的に構築できるSTREAMSと呼ばれる機能をサポートしている。各層/ASEをSTREAMSモジュールとして移植する形態を以下に示す(図2)。

- 各STREAMSモジュールに対しては、カーネルによってメッセージ送受信用のキューが提供されている。7層ボードOS模擬機能は、7層ボードOSが提供するキューの処理をSTREAMSのキュー操作にマッピングする処理を行なう。
- STREAMSモジュールのスケジューリングはカーネルによってサポートされているため、スケジューリング機能を設ける必要はない。
- 7層ボードOS模擬機能は、タイマなどのシステム·サービスをカーネル·サポートルーチンを利用して実現する。
- STREAMSモジュール間で共有され、キューを介して転送されるメッセージのバッファ構造はカーネルによって規定されている。回線制御用ドライバが扱うデータはこのバッファ構造に従うため、ドライバとの送受信におけるコピーを避けるために、OSI 7層ボードで実現したPDUバッファと同様の機能を、カーネルが定めるバッファ構造に従って実現する。この場合の実現方式を図3に示す。PDUバッファは、複数のメッセージ·ブロック(MB)、データ·ブロック(DB)、およびバッファ本体から構成される。データ·ブロックはバッファ本体のアドレスを保持し、DB2のように複数のメッセージ·ブロックが同じバッファを参照することを可能とする。

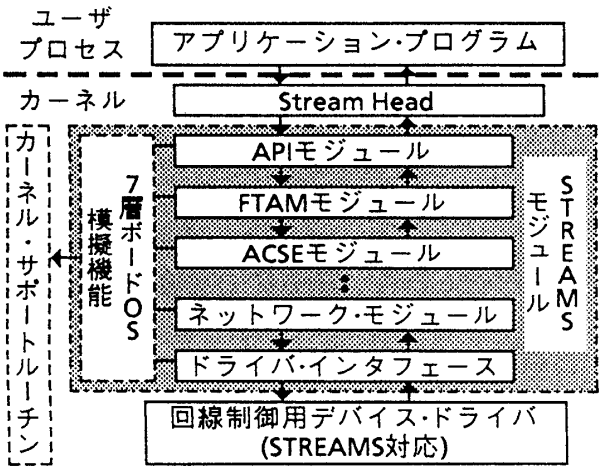

図2 カーネル内での実現形態

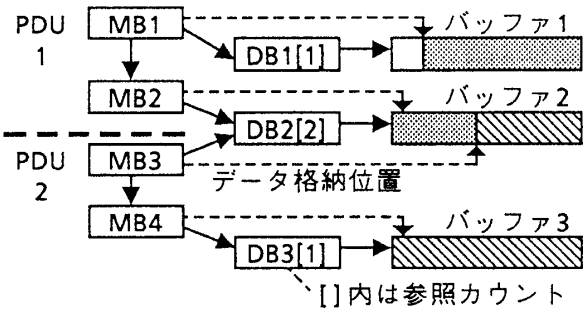

図3 STREAMSにおけるバッファの構成

メッセージ·ブロックにはバッファ内のデータ格納位置が保持されており、MB1とMB2、MB3とMB4のように、複数のメッセージ·ブロックを連結することによって1つのPDUを構成できる。

- プリミティブ転送用の共有バッファは、図3に示したバッファ本体へのポインタを返す関数を用意することにより実現可能である。この場合、対応するメッセージ·ブロックとデータ·ブロックはOS模擬機能が管理する。
- ユーザ·プロセスとカーネルとの間では、Stream Headを介して、ある一定サイズの連続領域のデータ転送が行なわれる。このため、プリミティブとユーザ/カーネル間の転送データとの変換を行なうAPIモジュールを設ける。

5. おわりに

本稿では、OSI 7層ボード用のプロトコル·プログラムをUnixワークステーション上のソフトウェアとして実行するために、ユーザ·プロセスに組み込まれるライブラリとして実現する形態と、STREAMS機能を用いてカーネルに組み込む形態に関する検討結果を述べた。7層ボードOSの機能を模擬する処理を実現することにより、両者の形態とも、プロトコル·プログラムは変更を伴わずに移植可能である。いずれの形態を採用するかは、アプリケーション·プログラム作成の利便性や要求される性能などを考慮して選択する必要がある。最後に、日頃御指導いただくKDD研究所浦野所長、眞家次長に感謝する。

参考文献

[1]: 井戸上, 加藤, 鈴木, "OSI 7層ボードの実装と評価," 情報処理学会マルチメディア通信と分散処理研究会, 93-DPS-61-28, July 1993.

[2]: 井戸上, 加藤, 鈴木, "OSI 7層ボードにおける通信ボード用OS," 情報処理学会マルチメディア通信と分散処理研究会, 92-DPS-56-11, July 1993.

[3]: 加藤, 井戸上, 鈴木, "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," 情報処理学会マルチメディア通信と分散処理研究会, 93-DPS-62-13, Sept. 1993.