OSI 7層ボード用プロトコル･プログラムのライブラリ化

1T－3

井戸上 彰　　加藤 聰彦　　鈴木 健二
KDD研究所

1. はじめに

筆者らは先に、パソコン/ワークステーション用のOSI 7層ボード[1]を開発しているが、Unixワークステーションや Windowsパソコンなどの高性能化･低廉化に伴い、これらの計算機本体上で効率的に動作するOSIソフトウェアへの要求も高まっている。OSI 7層ボード用のプロトコル･プログラムは、独自のOSI 7層ボードOS[2]のもとで開発･実行されるが、ボード上以外にも移植可能なように設計されている[3]。そこで本稿では、OSI 7層ボード用のプロトコル･プログラムを用いて、UnixやWindowsなどのホストOS上でOSIライブラリ･パッケージを実現するための実装方式について述べる。

2. 基本方針

OSI 7層ボード用のプロトコル･プログラムを用いて、移植性の高いOSIライブラリ･パッケージを実現するため、以下の方針を採用した。

- ホストOS上で動作するユーザ･プロセスに含まれるライブラリとして実現する。
- OSI 7層ボード上に加えて、UnixやWindowsなどの様々なホストOS上において、同一のプロトコル･プログラムを使用可能とする。
- OSI 7層ボードOSが提供する実行環境と同等の機能を擬似OSモジュール(以下擬似OSと呼ぶ)として実装する。
- 各層/ASEのプロトコル･モジュールや回線インタフェースとの入出力モジュールは、擬似OSから呼ばれるサブルーチンとして実行する。
- メモリ領域の割り当て、タイマ、回線などのデバイスに対する入出力処理などの方法は、ホストOSに応じて異なる。移植性を高めるため、このようなホストOSに依存する部分はできるだけ局所化する。
- 様々な処理を行うユーザ･プログラムの開発をサポートするため、プロセスのメイン関数はユーザ･プログラムが作成し、ライブラリが提供する関数を呼び出すことによって簡便にOSI通信機能を利用可能とする。

3. ライブラリの実現方式

OSI 7層ボード用プロトコル･プログラムをベースとしたOSIライブラリの構成を図1に示す。

3.1 擬似OS

擬似OSは、以下に示す機能を提供しており、UnixやWindowsなどに応じて開発されるホストOS依存部と、ホストOS独立部から構成される(表1参照)。

① スケジューリング

各層/ASEのプロトコル･モジュールのスケジューリングは、キューを介したプリミティブ送受信を契機として実現する。さらに、回線入出力モジュールに関しては、キューを介したプリミティブ受信に加え、デバイスからの受信イベント発生によっても起動する。

回線入出力モジュールのスケジューリング方法は、ホストOSに依存しない方式が望ましい。そこで擬似OSは、回線インタフェースなどの入出力デバイスを識別するデバイス･ディスクリプタと、対応する入出力モジュールを記述したデバイス･イベント管理テーブルを設け、入出力モジュールの状態管理を行っている。ホストOS依存部では、個々のホストOSに応じた方法でデバ

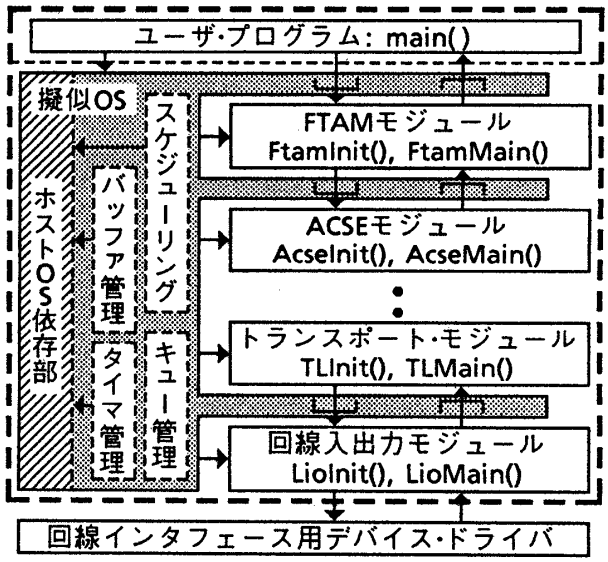

図1 OSIライブラリの構成

Implementation Method of OSI Library using Protocol Programs for OSI 7 Layer Board
Akira IDOUE, Toshihiko KATO and Kenji SUZUKI
KDD R & D Laboratories

表1 擬似OSの機能分担

	ホストOS依存部	ホストOS独立部
スケジューリング	デバイスからの受信イベント待ちとチェック	受信キュー/イベントに基づく各モジュールの状態管理と実行制御
バッファ管理	メモリ領域の確保/解放	キューイング用のヘッダ領域の管理
キュー管理	－	キューの生成と送受信処理
タイマ管理	一定間隔の経過時間計測	タイマ用データの管理とタイムアウト通知

イスからの受信イベント待ちを行い、その発生結果をデバイス･イベント管理テーブルに設定する。この結果に基づき、ホストOS独立部が回線入出力モジュールの実行制御を行う。

② バッファ管理

プリミティブやPDUを作成するためのバッファの確保/解放は、malloc()/ free()などのホストOSによって提供される機能を用いて実現する。ホストOS独立部では、プリミティブのキューイングなどのために、バッファ確保時にヘッダ領域を設け、その管理を行う。

③ キュー管理

プリミティブ転送用のキューの処理に関しては、ホストOSに依存せず、OSI 7層ボードOSと同一の機能を実現している。

④ タイマ管理

プロトコル･モジュールが使用するタイマは、ホストOS独立部において、タイムアウトまでの残り時間などを記述したタイマ用データ･バッファのリストとして管理されており、タイムアウトはキューを介して特殊なプリミティブとして通知される。さらにホストOS独立部は、ホストOS依存部が用意する一定間隔の時間計測機能を使用してタイムアウト･チェックを実行する。

3.2 プロトコル･モジュール

各層/ASEのプロトコル･モジュールは、それぞれの初期化関数(TLInit()等)と、プロトコル処理のメイン関数(TLMain()等)の2つのエントリを持つ。初期化関数では、管理テーブルやパラメータの初期化、プリミティブ転送用のキューの生成などを行う。一方、メイン関数では、キューを介して上下の層/ASEからのプリミティブを受信し、必要な処理を行ってプリミティブをキューに送信する処理を繰り返す。すべての受信プリティブの処理を終了した後、擬似OSにリターンして他のモジュールに制御を移す。

3.3 回線入出力モジュール

回線入出力モジュールは、擬似OSによるスケジューリングのために、デバイス･イベント管理テーブルに対するデバイス･ディスクリプタの登録/削除と、受信イベントのチェックを行う擬似OSの関数を使用する。デバイス･ディスクリプタを登録することにより、該当デバイスからの受信イベントの発生に応じて回線入出力モジュールのメイン関数が呼ばれる。メイン関数では、受信イベント･チェック関数の結果に基づき、該当デバイスからのデータ受信処理などを行う。

3.4 ユーザ･プログラムにおける処理

OSIライブラリを利用するユーザ･プログラムは、以下の処理手順に従う(図2参照)。

- 擬似OSやプロトコル･モジュールの初期化処理を行う関数を呼ぶ(OsiSysInit())。
- 必要に応じて、デバイスからの受信やユーザからのキー入力などの受信イベントを待つ処理を呼ぶ(OsiSysSelect())。
- 各層/ASEのスケジューリングを行うために、擬似OSが提供するスケジューラ関数を呼び出す(OsiSysScheduler())。

```
main() {
  ... /* ユーザ固有の初期化処理 */
  OsiSysInit(); /* 擬似OSと各層の初期化 */
  for(;;) {
    OsiSysSelect(); /* 受信イベント待ち */
    ... /* ユーザ固有の処理*/
    OsiSysScheduler(); /* 各層の実行 */ }
}
```

図2 ユーザ･プログラムの処理例

4. 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やWindowsなどのOSのもとで、ユーザ･プロセスに組み込まれるライブラリとして移植するための実現方式を述べた。ライブラリの核となる擬似OSは、ホストOSに依存する部分と独立な部分を分離し、移植性を高めている。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, 小野, "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, May 1994.

[3]: 井戸上, 藤長, 加藤, 鈴木, "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," 第48回情処全大, 1D-3, Mar. 1994.