2次記憶装置を利用可能としたプロトコル用バッファ制御方式

5O－3

井戸上　彰　　大岸　智彦　　加藤　聰彦

国際電信電話株式会社　研究所

1. はじめに

筆者らは先に、高速なOSIプロトコル処理を実現するために、PDU (Protocol Data Unit)の作成／解析時に、すべての層にわたってユーザデータのコピーを行わないバッファ制御方式を開発した[1]。本方式は、複数のメモリバッファに分散して格納されたデータを、1つのPDUとして管理／操作することを可能としている。一方、MHS (Message Handling System)では、画像情報などの大規模なデータを含むメッセージを1つのPDUとして扱う必要がある。このような場合、メッセージ送信時はファイルに格納された画像データなどを使用し、メッセージ受信時はRT (Reliable Transfer)においてファイルに格納された受信データをそのまま使用して、ヘッダ部分やASN.1の制御情報のみをメモリ上で作成／解析する機能が要求される。そこで本稿では、データがディスクファイルなどの2次記憶装置に格納された状態においても、ファイル全体の読み出しやコピーを伴わずにPDU作成／解析処理を実現するPDUバッファ制御方式の概要について述べる。

2. 設計方針

以下に、2次記憶装置を利用可能なPDUバッファ制御方式の設計方針を示す。

(1) ディスクファイルなどの2次記憶装置上では、対象とするファイルシステムに応じた物理的／論理的なファイルブロックをPDUの格納単位として管理し、複数のメモリ上のバッファやファイルブロックに分散して格納されたデータを1つのPDUとして管理／操作可能とする。

(2) 先に開発した、メモリ上でのPDUバッファ制御方式と互換性を持つPDUバッファ制御ライブラリを提供し、メモリバッファ上およびファイルブロック上に存在するPDUを同一のプログラミング・インタフェースで操作可能とする。

(3) 1つのPDUが複数のファイルにまたがって納される場合や、1つのファイルに複数のPDUが格納される場合などにも対応可能とする。

A Protocol Buffer Control Method without Copying User Data Stored in Secondary Storage Devices

Akira IDOUE, Tomohiko OGISHI and Toshihiko KATO

KDD R&D Laboratories

3. 2次記憶装置を利用可能なPDUバッファ制御方式

3.1 データ構造

設計方針で述べた機能を実現するために、以下のようなデータ構造を導入する(図1参照)。

① PDUエレメント・ディスクリプタ(PED)

PDUエレメント・ディスクリプタ(PED)は、PDUを構成する個々のデータ要素を管理するもので、後述のメモリバッファ・ディスクリプタやファイルブロック・ディスクリプタへのポインタ(MemBufPtr/FBlockPtr)を保持する。データがメモリバッファ上に存在する場合は、バッファ本体のヘッダ用空き領域(WorkHead)やPDU格納領域(PduHead)へのポインタを持ち、データがファイルブロックに格納されている場合は、ブロック内のPDU開始オフセット(PduOffset)を保持する。各PEDが保持するPDUの長さはPduLenで示される。PDUを複数のPEDのリストで構成するために、前後のPEDへのポインタ(PrevPtr / NextPtr)が使用され、リストの先頭PEDのTotalLenがPDU全体の長さを表す。

② メモリバッファ・ディスクリプタ(MBD)

個々のメモリ上のバッファを管理するために、メモリバッファ・ディスクリプタ(MBD)を使用する。MBDは、バッファ本体へのポインタ

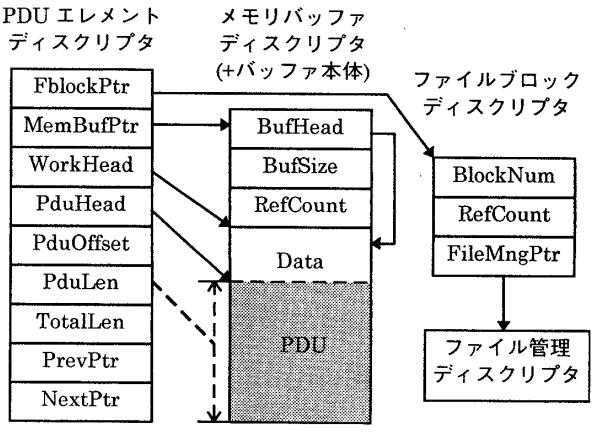

図1　PDUバッファのデータ構造

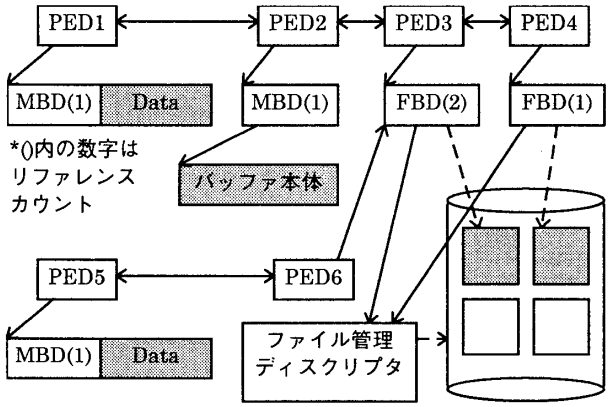

図2 PDUバッファの構成例

(BufHead)や、本バッファを参照するPEDの数に対応するリファレンスカウント(RefCount)などを持つ。データが格納される領域は、通常は本MBDに続く連続領域(Data)として確保されるが、ユーザが用意する別のバッファ本体を参照することも可能である。

③ ファイルブロック・ディスクリプタ(FBD)

個々のファイルブロックを管理するために、ファイルブロック・ディスクリプタ(FBD)を用いる。FBDは、ブロック番号(BlockNum)、本ブロックを参照しているPEDのリファレンスカウント(RefCount)、および各ファイルのファイル管理ディスクリプタへのポインタ(FileMngPtr)を持つ。

④ ファイル管理ディスクリプタ

ファイル管理ディスクリプタは、ファイル名や、ファイルブロックのサイズ、オープン時のファイルディスクリプタ(fd)の値、装置番号などのファイルシステム固有の情報など、個々のファイルに関する管理情報を保持する。

以上のようなデータ構造によるPDUバッファの構成例を図2に示す。図2において、PED 1〜4のリストと、PED5, 6のリストが、それぞれ1つのPDUを表している。

3.2 PDU作成／解析処理手順

ファイルに格納されたデータを用いてPDUを作成／解析する処理は以下のように実現される。

PDU作成時は、最初にユーザデータを含むファイルをオープンしてファイル管理ディスクリプタを作成し、PEDとFBDの組を割り当ててPDUバッファを構成する。ヘッダやASN.1の制御情報を付加する場合は、ヘッダ用の新たなPEDとMBD(+バッファ本体)の組を割り当て、ファイルブロックを保持するPEDに連結することによって実現できる。セグメンティングが必要な場合は、元のPEDが保持するメモリバッファやファイルブロックの一部を参照する新たなPEDリストを作成することで実現可能である。

ファイルブロックに格納されたPDUの解析が必要な場合は、MBD(+バッファ本体)を割り当てて該当ファイルブロックをバッファ上に読み出し、バッファ上の指定データ位置へのポインタを返す処理を行う。PDUのリアセンブリングが必要な場合は、1つのPDUを構成するPEDリストの後に、別のPDUに対応するPEDリストを連結することによって実現できる。

3.3 ファイル入出力処理

Unixなどでは、一般にファイル入出力(read/write)が完了するまでプロセスの実行がブロックされる。すべての層のプロトコルが1プロセスで実装されるような場合[2]において、ファイル入出力中に他の層や別のコネクションの処理を実行可能とするためには、ファイル入出力処理をスレッドで分割するなどの対処が必要となる。一方、ファイル入出力要求とその完了通知が非同期で行われるシステムも存在する。

いずれの場合においても、PDUバッファ制御ライブラリにおいて、データの書き込み／読み出しを要求する関数と、入出力の完了が通知された後に処理を再開する関数を設けるなどの工夫が必要となる。

4. おわりに

本稿では、OSIなどの階層構造を持つプロトコルの実装において、データがファイルに格納された状態で、PDU作成／解析時に、ファイルからのデータ読み出しや、メモリ上でのコピーを伴わないバッファ制御方式について述べた。今後、本方式に基づいて、大規模データを転送可能なプロトコル・プログラムを実装し、その評価を行う予定である。最後に、日頃御指導頂くKDD研究所村上所長に感謝する。

参考文献

[1] 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,” マルチメディア通信と分散処理研究会, 93-DPS-61-28, Sept. 1993.

[2] 井戸上, 加藤, 鈴木, “OSI 7層ボード用プロトコル・プログラムのライブラリ化,” 第50回情処全大, 1T-3, Mar. 1995.