マルチメディア通信と分散処理　62－13
（1993. 9. 21）

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

加藤 聰彦　　井戸上 彰　　鈴木 健二

国際電信電話(株)研究所

〒356　埼玉県上福岡市大原2-1-15

あらまし　OSI標準に準拠したコンピュータ通信システムの普及に伴い、その高性能化が重要な課題となっている。階層構造を採用しているOSIプロトコルにおいては、各層のPDU(Protocol Data Unit)が、制御情報(PCI : Protocol Control Information)と、ユーザデータとして扱われる上位層のPDUから構成される。高速なOSIシステムを実装するためには、各層のプロトコル処理においてユーザデータのコピー避ける必要がある。本稿では、PCIとユーザデータを別個に扱い、ユーザデータのコピーを伴わずにPDU作成解析処理を実現するバッファ制御方式を提案する。さらに、本方式の性能を、各層でユーザデータをコピーする方式と比較した結果について述べる。

和文キーワード　OSI, 高速通信, PDU, バッファ制御

A Buffer Control Method Avoiding User Data Copying
for OSI Protocol Implementation

Toshihiko KATO　　Akira IDOUE　　Kenji SUZUKI

KDD

2-1-15, Ohara, Kamifukuoka-shi,

Saitama 356

Abstract　The high performance is one of the most important issues in the OSI protocol implementation. OSI has the layered architecture and each layer introduces its own PDU (Protocol Data Unit) which comprises a PCI (Protocol Control Information) of the layer and user data containing a PDU of the adjacent higher layer. In order to achieve the high performance, it is required to avoid user data copying in the protocol processing. This paper proposes a buffer control method for PDUs which realizes the PDU encoding and decoding without copying user data. This paper also describes the results of the performance measurement experiment which compares the proposed method and the conventional method where user data are copied in each layer.

英文 key words　OSI, High Speed Communication, Protocol Data Unit, Buffer Control

1. はじめに

近年、異なるシステムとの相互接続を目的として、OSI通信プロトコルに準拠したコンピュータ通信システムの構築が行われている[1-3]。一方、光LANなどの高速伝送路の導入に従い、プロトコル処理のオーバヘッドを削減し、高性能なデータ転送を行なうコンピュータ通信システムの実現が重要となっている。従って、今後OSI準拠のコンピュータ通信システムを広く普及させるためには、その高性能化がキーとなると考えられる。

OSIは階層構造を採用しており、各層または応用層の応用サービス要素(ASE)において、独立にプロトコルが規定されている(以下明示的に必要となる場合以外では、"層またはASE"を"層"で代表する)。第(N)層のプロトコルは、独自の制御情報(N)-PCI(Protocol Control Information)を定め、第(N+1)層で扱われる(N+1)-PDU(Protocol Data Unit)をユーザデータとして、それに(N)-PCIを付加することにより、第(N)層のPDUを作成する。

このような構造のPDUを各層のプロトコルで作成解析するために、従来のOSIシステムでは以下のような方法を用いていた[1-3]。

- ある層のPDUの作成において、上位からのユーザデータにPCIを付加するために、PDU全体を格納可能なバッファを確保し、PCIとユーザデータをコピーする。
- ある層のPDUの解析において、そのPDUに含まれるユーザデータの長さ分のバッファを確保し、上位層に渡すユーザデータをコピーする。

この実装方法では、ユーザデータのコピーが層の数だけ行なわれる。これまでのOSIプロトコルの実装は、主にパケット網やISDNの基本インタフェースを対象としており、ユーザデータのコピーの負荷が問題ではなかったと考えられる[1-3]。しかし、高速伝送路に対応したOSI通信を実現するためには、層ごとに行なわれるコピーを避ける必要がある。

このためには、各層においてPCIとユーザデータを別個に扱い、PCIの付加/除去やPDUのセグメンティング/リアセンブリングなどのPDU作成解析処理を、ユーザデータのコピーを伴わずに実現できるバッファ制御方式を導入することが必要となる。さらにその制御方式は、

- セグメンティングやリアセンブリングが、複数の層で行なわれることや、
- 第6層と第7層のPDUの構造と符号化規則は、ASN.1 (Abstract Syntax Notation One)により形式的に定義されること

などのOSIプロトコルの特徴を考慮する必要がある。

本稿では、OSIプロトコルを実装するための、ユーザデータのコピーをまったく行なわないバッファ制御方式を提案する。以下、2.において、提案するPDUバッファ制御方式の概要を示し、3.および4.で、それぞれ、本方式をOSIの第5層以下と、第6層/第7層のプロトコルに適用する方法を示す。5.において、性能測定実験を通して本方式を評価し、6.で本方式に対して考察を加える。

2. PDUバッファ制御方式の概要

2.1 PDUバッファ制御方式の目的

PDUバッファ制御方式は、各層のプログラム・モジュールに対して、次のようなPDU作成解析処理機能を提供することを目的としている。

- PDUの作成解析のためのPDUバッファは、すべてのプログラム・モジュールからアクセス可能な共有メモリ領域上に確保される(図1①)。
- PDUの作成においては、(N+1)層プログラム・モジュールが、PDUバッファを確保し(N+1)-PDUを作成する(図1②)。次に、そのPDUバッファのポインタを通知された(N)層プログラム・モジュールが、そのPDUバッファ上に(N)-PCIのための領域を確保し値を設定する(図1③)。この場合、(N+1)-PDUのコピーは行なわれない。
- PDUの解析においては、逆に、(N)層プログラム・モジュールが(N)-PDUを保持するPDUバッファから、(N)-PCIを除去することにより(N+1)-PDUを取り出し(図1②)、PDUバッファのポインタを(N+1)層プログラム・モジュールに通知する。この場合も(N+1)-PDUはコピーされない。

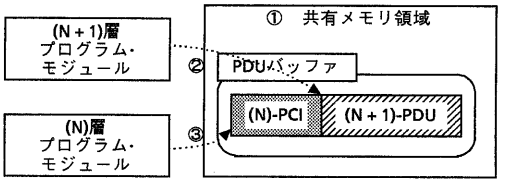


図1 PDUバッファによるPDU作成解析処理

このようなバッファ制御により、各層のプログラム・モジュールは、他の層とは独立にPDUの作成解析を行ないつつ、ユーザデータのコピーを避けることができる。

2.2 PDUバッファの構成

PDUバッファは、図2に示すように、PDU自身を蓄えるデータ・バッファと、データ・バッファを管理するPDUバッファディスクリプタ(以下では略してPBDと呼ぶ)を構成要素として持つ。

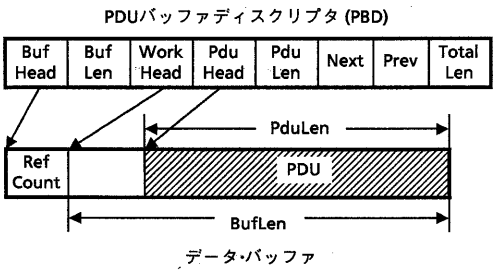


図2　PDUバッファの構成要素

PBDは次のようなフィールドを有する。

- **BufHead**: このPBDが管理するデータ・バッファの先頭へのポインタ。
- **BufLen**: このPBDが管理するデータ・バッファの長さ。
- **WorkHead**: データ・バッファ上でPDUの作成に用いることのできる作業領域の先頭へのポインタ。
- **PduHead**: データ・バッファ上で実際にPDUが格納されている領域の先頭へのポインタ。
- **PduLen**: データ・バッファ上で実際にPDUが格納されている領域の長さ。
- **Next/Prev**: 1つのPDUが複数のデータ・バッファにより保持されている場合には、各データ・バッファを管理するPBDをリストで連結する。Next/Prevは、リストにおいてこのPBDの次または前のPBDへのポインタを示す。
- **TotalLen**: PDUの全体長。1つのPDUがPBDのリストにより保持される場合には、各PBDのPduLenの総和となり、リストの先頭のPBDにおいてのみ意味を持つ。

また、データ・バッファは複数のPBDから参照される場合があるため、そのデータ・バッファを参照しているPBDの数を示すRefCountフィールドを持つ。

PDUバッファは、1つまたはリストされた複数のPBDから構成され、各PBDは各々1つのデータ・バッファを管理する。PDUバッファの利用者は、リストの先頭のPBDへのポインタを、PDUバッファへのポインタとして扱う。このようなPDUバッファにより

- 実際にPDUを格納した領域の前に、下位層のPCIを作成するための作業領域を用意すること(図3のデータ・バッファ1)、
- 複数のデータ・バッファにまたがったデータを、1つのPDUとすること(図3のPBD1とPBD2)、
- 1つのデータ・バッファ内のデータを分割して、複数のPDUとすること(図3のPBD2とPBD3)、
- データ・バッファに格納された1つのデータを、異なるPDUとして扱うこと(図3のPBD3とPBD4)

などが可能となる。なお以下では、図中でのPBDからデータ・バッファに対する矢印は、PduHeadを示すものとする。

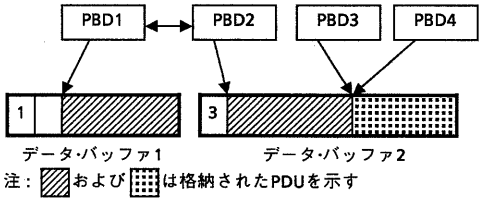


図3　PDUバッファの構成例

2.3 PDUバッファによるPDU作成解析処理の実現法

次に、PDUバッファにより、ユーザデータをコピーせずにPDU作成解析処理を実現する方法を示す。

(1) PCIの付加と除去

(N)層プログラム・モジュールは、(N+1)層より受信した(N+1)-PDUに、以下のように(N)-PCIを付加する(図4参照)。

- (N+1)-PDUを格納するデータ・バッファにおいて、(N)-PCIを作成する領域が確保できる場合は、同一のデータ・バッファを用いて(N)-PCIを付加する(図4(b))。
- そうでない場合は、新たに(N)-PCI用のデータ・バッファを確保し、そこに(N)-PCIを作成し、(N+1)-PDUを格納するデータ・バッファとPBDで連結する(図4(c))。この場合、これまでのPBDを、新たに確保したデータ・バッファに対応させる必要がある。

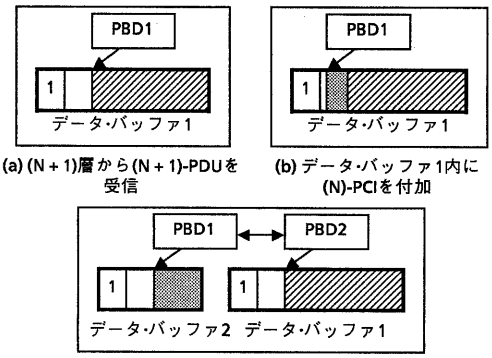
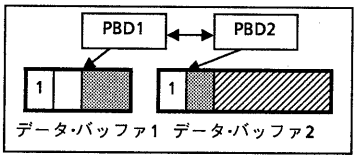


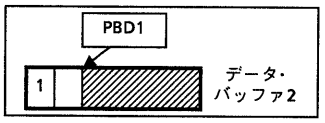
図4　PCIの付加方法

(N)層プログラム・モジュールは、(N−1)層から受信した(N)-PDUに対して、(N)-PCIを解析しPBDのPduHeadをずらすことにより(N)-PCIを除去する。その際、図5に示すように、データ・バッファが(N)-PCIのみを含む場合は、そのデータ・バッファを解放し、

リストの先頭のPBDを、ユーザデータの先頭を含むデータ・バッファに対応させる。



(a) (N－1)層から(N)-PDU受信



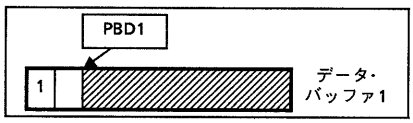
(b) (N)-PCIを除去

注：(N)-PCI (N＋1)-PDU

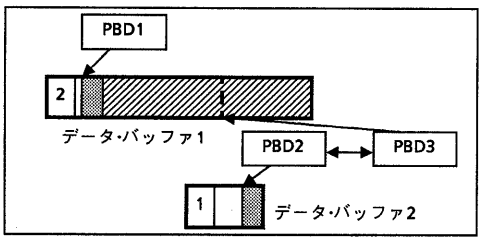
図5　PCIの除去方法

(2) セグメンティングとリアセンブリング

(N)層プログラム・モジュールは、(N＋1)層より受信した(N＋1)-PDUの長さが、(N)-PDUの最大長よりも大きい場合は、図6のようにセグメンティングを行なう。分割される最初の(N)-PDUに対しては、(N＋1)-PDUを格納するデータ・バッファを用いて(N)-PCIを付加し、その(N)-PDUにより運ばれるユーザデータの範囲を指定するようにPBDのPduLenを変更する。次の(N)-PDUに対しては、(N)-PCIを格納するデータ・バッファを新たに確保し、PBDにより、残りのユーザデータと連結する。



(a) (N＋1)層から(N＋1)-PDUを受信

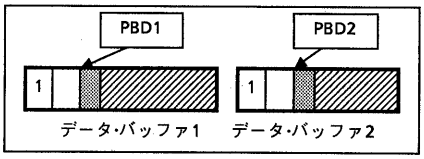


(b) (N＋1)-PDUをセグメンティング

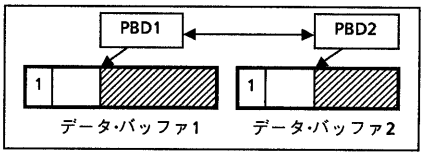
注：(N)-PCI (N＋1)-PDU

図6　セグメンティングの方法

また、(N)層プログラム・モジュールは、(N－1)層より受信した(N)-PDUが、(N＋1)-PDUをセグメンティングしている場合は、関連するすべてのデータ・バッファをPBDで連結し、完全な(N＋1)-PDUを作成する(図7参照)。



(a) (N－1)層から(N)-PDU受信



(b) ユーザデータを連結

注：(N)-PCI (N＋1)-PDU

図7　リアセンブリングの方法

(3) コンカティネーションとセパレーション

ある層のPDUを複数連結するコンカティネーションと、それを分離するセパレーションも、それぞれリアセンブリングとセグメンティングの処理と同様にして実現することができる。

3. 第5層以下のOSIプロトコルへの適用

3.1 第5層以下におけるPDU作成解析処理の特徴

OSIの第5層のセション層以下におけるPDUの作成や解析の処理は、次のような特徴を持つ。

- PCIは必ずユーザデータの先頭に付加される。
- データ転送用のPDUにおいては、PCI自身の長さは、固定的に定められているか、PCIの先頭に明示的に長さが示されるかであるため、PCIとユーザデータを容易に分離することができる。
- 実装においては、各層で定義されたPDUの構造に従って、プログラマが個別にPDUの作成解析の処理を実現するのが通常である。
- セグメンティング/リアセンブリング、コンカティネーション/セパレーションが各層で独立に行なわれる。

3.2 PDU作成解析処理の実現方法

上述の特徴を考慮して、PDUバッファを用いて第5層以下のPDUの作成解析処理を実現するために、以下のような方法を用いる。

① 2.で述べたPDUバッファを制御するために、表1に示すような関数を準備する。これらのPDUバッファ制御用関数をプログラマに提供し、PDUの作成解析処理を実装させる。

② プログラマは表1の関数を用いて、次のように、PDUの作成処理を実装する。

- PCI付加の処理においては、PCIの長さを求め、`PDUHeaderAlloc()`を用いてその領域を確保す

表1　プログラマに提供されるPDUバッファ制御用関数

関数名	引数	関数の処理概要
PDUAlloc	size_t size	PBDとsize分のデータを格納できるデータ･バッファを自動的に割り当て、PBDへのポインタを返す。下位層のPCIのための作業領域を確保するために、PduHeadはWorkHeadから64バイトの位置を指す。
PDUFree	PBD_t *pbdp (PBD_tはPBDのデータ型を示す)	pbdpの指すPDUバッファを解放する。pbdpを先頭とするリスト中のPBDを解放し、対応するデータ･バッファのRefCountを1減らし、RefCountが0となったデータ･バッファを解放する。
PDUTotalLen	PBD_t *pbdp	pbdpの指すPDUバッファに対応するPDUの全体長を返す。
PDUGetPtr	PBD_t *pbdp, size_t offset, *restLen	pbdpの指すPDUバッファにおいて、先頭からoffsetの位置のバイトへのポインタと、そのバイトから連続的にアクセスできる(すなわち同じデータ･バッファに格納された)PDUのデータの長さrestLenを返す。
PDUHeaderAlloc	PBD_t *pbdp, size_t len	pbdpの指すPDUバッファにおいて、2.で述べた方法により、len分のPCI用の領域を、PDUの前に確保する。
PDUStripHeader	PBD_t *pbdp, size_t len	pbdpの指すPDUバッファの先頭から、len分のPCIの領域を削除する。
PDUAppend	PBD_t *pbdp1, *pbdp2	pbdp1の指すPDUバッファに対して、pbdp2の指すPDUバッファのデータをつないで、1つのPDUとする。
PDUBreak	PBD_t *pbdp1, *pbdp2, size_t len,	pbdp1の指すPDUバッファが保持するPDUのデータを、先頭からlen分だけのデータを持つPDUとし、残りのデータを保持するPBDバッファを割り当てて、pbdp2に代入する。
PDUGetData	PBD_t *pbdp, size_t offset, len, char *dp	pbdpの指すPDUバッファが保持するPDUのデータを、先頭からoffsetの位置から、lenバイト分を連続領域dpにコピーする。

る。確保された領域はデータ･バッファ上の連続領域であるため、PDUGetPtr()により先頭のバイトのポインタを得て、その位置から連続的にPCIの値を設定する。

- セグメンティングを行なう必要がある場合は、そのPDUに含まれるべきユーザデータをPDUBreak()により切り出し、それにPCIを付加するという操作を繰り返す。

③ プログラマは次のように、PDUの解析処理を実装する。

- PDUの受信時にPCIを除去する処理においては、まずPDUGetPtr()により先頭のバイトのポインタを得て、受信したPDUの種別およびPCI自身の長さを求める。次に必要に応じてPCIの情報を、PDUGetData()により連続領域にコピーして解析する。最後にもとのPDUからPCI分の長さをPDUStripHeader()により除去する。
- そのPDUが上位層PDUの一部のみを含む場合は、ユーザデータを保持しておき、その上位層PDUの最後の部分を運ぶPDUを受信した時点でPDUAppend()により上位層のPDUを作成する。

③ 各層のセグメンティングなどの処理は、その層のPDU作成解析処理の中で独立に行なわせる。

4. 第6層/第7層のOSIプロトコルへの適用

4.1 第6層/第7層におけるPDU作成解析処理の特徴

OSIの第6層のプレゼンテーション層(PL)および第7層の応用層のASEにおいては、PDUがASN.1により形式的に定義される。このため、第6層/第7層のPDU作成解析処理は次のような特徴を持つ。

- ASN.1の定義するPDUでは、PCIとユーザデータとが混在するのが一般的である。例えばPLでは、応用層PDUを運ぶために、応用層PDUの構造を示すプレゼンテーション･コンテクスト識別子とユーザデータ自身を含むPDV-listを、SEQUENCE OF型により連結したパラメータを用いる。このため、PDU作成解析処理においては、PCIがユーザデータの先頭に付加されるという構造を前提とすることができない。
- 転送されるPDUのオクテット形式は、ASN.1の基本符号化規則に従う。ASN.1で定義されたデータ構造では、パラメータにタグが割り当てられ、符号化される場合には、タグ、値の長さ、値の3つ組として符号化される。1つのデータ構造のあるパラメータが、別のデータ構造により定義されている場合は、そのパラメータの値の中に、別のデータ構造を符号化したものが入る。このため、ユーザデータを運ぶパラメータの位置は、PDUを先頭から順に解析していかなければ特定できない。
- このようなASN.1に従ったPDUの作成解析を実現するために、通常の実装では、ASN.1に従った符号化/復号を行なうルーチンを生成するASN.1コンパイラ[4]を使用するのが一般的である。

4.2 PDU作成解析処理の実現方法

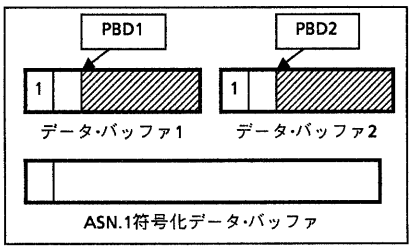
第6層/第7層のPDUの作成解析処理をPDUバッファを用いて実現するために、以下の方法を用いる。

① PDUの作成処理においては、以下の手順で、新たにデータ・バッファを確保し、そのPDUの複数箇所に存在するPCIのみをそのデータ・バッファ上に作成する(図8参照)。

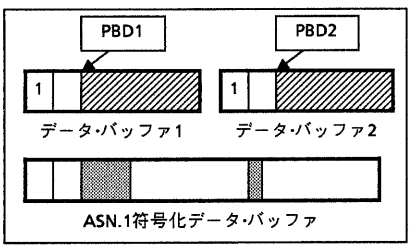
(a) 上位のASEのPDUを受信すると、その層/ASEで作成すべきPDU全体長を計算し、その長さ分のデータ・バッファを確保する。以下このデータ・バッファをASN.1符号化データ・バッファと呼ぶ。

(b) ASN.1符号化データ・バッファを用いて、その層/ASEのPDUの符号化を行なう。あるパラメータの値が上位のASEのPDUである場合は、それをASN.1符号化データ・バッファにはコピーせず、バッファ内のポインタのみを進める。

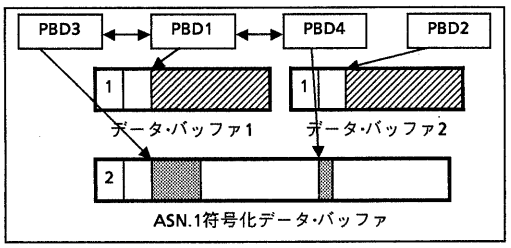
(c) ASN.1符号化データ・バッファ内で値を設定した部分と、上位のASEのPDUを含むデータ・バッファとをPBDで連結する。



(a) ASN.1符号化データ・バッファを確保



(b) その層/ASEのPCIをASN.1符号化データ・バッファ上に作成



(c) PBDで連結

注：その層/ASEのPCI　上位のASEのPDU

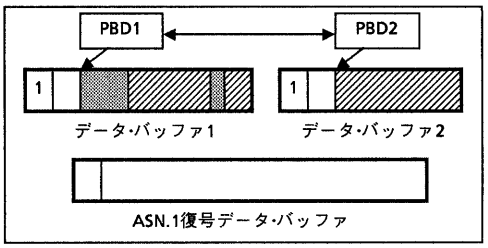
図8　第6層/第7層のPDU作成処理

② ASN.1で定義されたPDUの復号では、PDUを連続した領域に格納し先頭から順に処理する方法が効率的である。そこで、第6層/第7層のPDUの解析処理は、そのPDUが、セグメンティングにより複数のデータ・バッファに分割されたか否かにより別の方法をとることとした。PDUが1つのデータ・バッファに格納されている場合は、そのデータ・バッファを用いて復号しPDUの解析を行なう。PDUが複数のデータ・バッファに格納されている場合は、次のような手順で、その層/ASEのPCIを含むデータ・バッファを復号用のバッファにコピーして処理する(図9参照)。

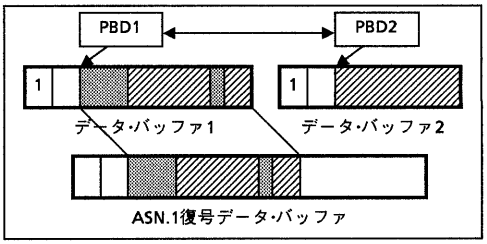
(a) 受信したPDUの長さ分のデータ・バッファ(ASN.1復号データ・バッファと呼ぶ)を確保する。

(b) PDUを構成する先頭のデータ・バッファのデータをASN.1復号データ・バッファにコピーして復号を行なう。復号を進める際に、その層/ASEのPCIが、別のデータ・バッファに格納されている場合は、そのデータ・バッファのデータを復号データ・バッファにコピーする。

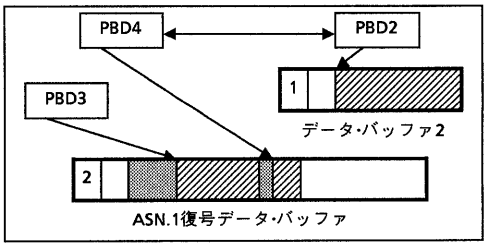
(c) 復号結果を解析し、ユーザデータに対して新たなPBDを割り当てる。



(a) ASN.1復号用データ・バッファを確保



(b) 先頭のデータ・バッファを、ASN.1復号データ・バッファにコピーし復号



(c) 復号結果にPBDを付加

注：その層/ASEのPCI　上位のASEのPDU

図9　下位層で分割された第6層/第7層のPDU解析処理

③ このような手順を、ASN.1コンパイラの生成する符号化/復号ルーチンに実装する。今回は、別途作成したASN.1コンパイラ[4]のライブラリを変更することにより上記手順を実現した。

5. 評価

PDUバッファ制御方式の評価を目的として、FTAM (File Transfer, Access and Management) 、PL、セション層(SL)、トランスポート層(TL)、ネットワーク層(NL)のデータ転送動作を対象として、PDUバッファを用いてPDU作成解析処理を行なうプログラムを作成し、計算機内部における折り返しデータ転送のスループットを測定した。さらに比較のために、各層でのPDU作成解析処理においてユーザデータのコピーを行なった場合のスループットも測定した。

実験は、SPARCServer 670MPを用いて行ない、SUN OSのもとで動作する1つのプロセスとして、ユーザ/FTAM/PL/SL/TL/NL/ループの各プログラム・モジュールを実現した。これらは、プロセス内のヒープ領域を共有メモリ領域として使用し、以下のような動作を行なう。

- ユーザ・モジュールは、送信するファイル・ブロックの用意と受信したファイル・ブロックの解放を行なう。PDU作成解析処理におけるコピーのオーバヘッドに着目するために、本モジュールではコピーを行なわないこととした。
- FTAMモジュールでは、ユーザ・ルーチンとの間でやり取りされるファイル・ブロックを

```
FADU ::= OCTET STRING
```

というPDUを用いて転送する。

- PLモジュールでは、FADUを

```
PUserData ::=
   [APPLICATION 1] IMPLICIT FulEncData
FulEncData ::= SEQUENCE OF PDVList
PDVList ::= SEQUENCE {
   pcId    INTEGER,
   pdv    CHOICE {
      singleASN1 [0] ANY,
      octetalign [1] IMPLICIT OCTETSTRING,
      arbitrary  [2] IMPLICIT BITSTRING}
```

というPDUを用いて転送する。

- SLモジュールでは、ユーザデータの前に、Give Tokens SPDUとDATA SPDUのPCIの計7バイトを付加する。
- TLモジュールでは、フロー制御を用いないクラス2の動作を想定し、DATA TPDUのPCI(5バイト)の付加/除去と、指定されたTPDUサイズに従ったセグメンティング/リアセンブリングを行なう。
- NLモジュールでは、X.25レベル3を想定し、DTパケットの3バイトのPCIを用い、指定されたパケット・サイズに従ったセグメンティング/リアセンブリングを行なう。フロー制御のデータ転送性能への影響を排除するために、RRパケットを用いたフロー制御は実装していない。
- ループ・モジュールは、NLモジュールの送信したDTパケットをプロセス内で折り返す。ネットワークの処理を模擬するために、PDUバッファ方式/コピー方式とも、一旦パケットを連続領域にコピーすることとした。

16Mバイトのデータを、ファイル・ブロック・サイズ/TPDUサイズ/パケット・サイズを変化して、スループットを測定した結果を図10と図11に示す。

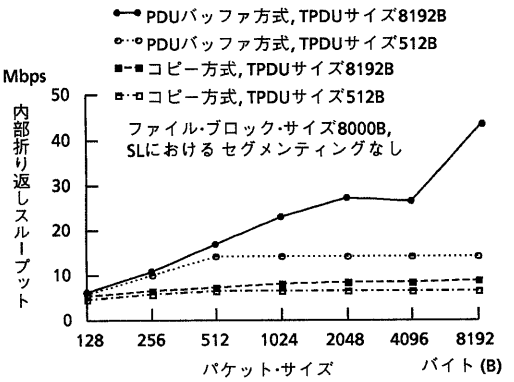


図10 TPDUサイズとパケット・サイズを変更した場合のスループット

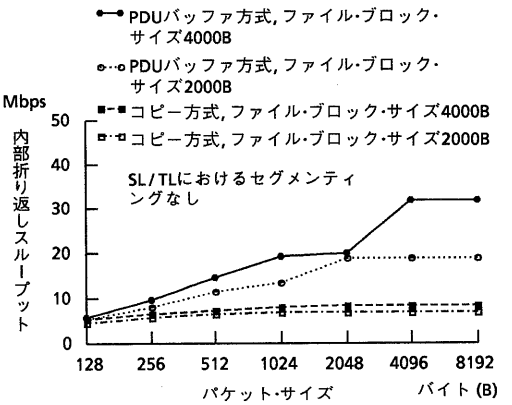


図11 ファイル・ブロック・サイズとパケット・サイズを変更した場合のスループット

6. 考察

(1) スループットの測定結果より、提案するPDUバッファ制御方式を用いたPDU作成解析処理が、各層でコピーを行なう場合に比べて、高い性能を達成することが明らかとなった。特に、PDUのサイズが大きい場合は、コピー方式に比べて4倍から5倍の性能を実現している。この結果から、ユーザデータをコピーするために必要となる、バッファの割り当て/データのコピー/バッファの解放のオーバヘッドが、プロトコル処理のボトルネックの1つであると判断できる。

(2) 実験結果より、提案する方式を使用した場合のスループットは、FTAMからネットワーク層までを用いて、送受信同時のデータ転送において、最大約43Mbpsであった。使用したプログラム・モジュールは、フロー制御を省いたデータ転送の機能のみを実装する実験用のものであるが、この結果は、本方式を用いたOSIプログラムが、高速通信にも充分対応できるという可能性を示している。

(3) これまでにも制御情報とユーザデータを別個に扱うバッファを導入し、コピーを行なわないプロトコル処理を実現する方式は提案されている[5,6]。しかし、これらはTCP/IPプロトコルを対象とするものに限られている。本方式はOSIプロトコルを対象とし、

- ASN.1により定義された第6層と第7層のPDUの処理においても、ユーザデータのコピーを避ける機能を導入していること、
- セグメンティング/リアセンブリングおよびコンカティネーション/セパレーションが複数の層で行なわれた場合にも対処可能なこと、
- PDUバッファを割り当てた時点で、下位層のPCIのための作業領域を確保し、PBDならびにデータ・バッファの割り当て/解放のオーバヘッドを減らしていること

などの特徴を持つ。

(4) 本方式では、セグメンティングがまったく行なわれない場合は、最初に割り当てたデータ・バッファをのみを用いてPDUの作成解析処理が行なわれる。このため、図10においてTPDUサイズとパケット・サイズがともに8192バイトの場合や、図11においてファイル・ブロック・サイズが4000バイトでパケット・サイズが4096バイトの場合などでは、他に比べて高いスループットを実現している。

(5) パケット・サイズが小さくなるにつれて本方式のスループットが低下しており、128バイトの場合はコピーを行なう方式と同程度となっている。PDUのサイズが小さく、セグメンティングされるPDU数が増加する場合については、さらに最適化を行なう必要がある。

7. むすび

本稿では、OSI通信プロトコルの実装において、各層の制御情報とユーザデータを別個に扱うことにより、ユーザデータのコピーを行なわずに、PDUの作成や解析処理を実現するためのバッファ制御方式を提案した。本方式を用いることにより、ASN.1でPDUが定義される第6層と第7層を含めて、OSIのすべてのプロトコル処理において、一度もユーザデータをコピーしない実装が可能となる。また、ネットワーク層からFTAMまでを用いたデータ転送を対象として、各層でユーザデータをコピーする方式と本方式とのスループットを測定したところ、本方式はコピー方式に比べ4倍から5倍の性能を実現し、その有効性が明らかとなった。本方式は、筆者らが別途開発したOSI 7層ボード[7]において用いられているが、今後はさらに最適化を行なうとともに、本方式を用いて高速な通信システムを構築していく予定である。最後に、日頃御指導頂くKDD研究所浦野所長、眞家次長に感謝する。

参考文献

[1] 鈴木, 加藤, 浦野: OSIトランスポートおよびセションプロトコルの実装と評価, 情処論, Vol. 29, No. 12, pp. 1180~1192 (1988)

[2] A. Idoue, T. Kato, K. Suzuki and Y. Urano: Design and Implementation of OSI Communication Board for Personal Computers and Workstations, Proc. of ICCC '90, pp. 585~592 (1990)

[3] 中川路, 勝山, 茶川, 水野: 国際標準仕様に準拠したファイル転送プロトコルの実現と評価, 情処論, Vol. 29, No. 11, pp. 1071~1078(1988)

[4] T. Hasegawa, S. Nomura and T. Kato: Implementation and Evaluation of ASN.1 Compiler, J. Inf. Process., Vol. 15, No. 2 (1992)

[5] X. Zhang and P. Seneviratne: An Efficient Implementation of a High-Speed Protocol Without Data Copying, Proc. of , pp. 443~450 (1990)

[6] N. Hutchinson: The x-kernel: An Architecture for Implementing Network Protocols, IEEE Trans. Software Eng., Vol. 17, No. 1, pp. 64~76 (1991)

[7] 井戸上, 加藤, 鈴木: OSI 7層ボードの実装と評価, 情処, マルチメディア通信と分散処理研究会, Vol. 93, No. 58, pp. 211~218 (1993)