HIPPI上で高速通信を実現する軽量プロトコルに関する一考察

5O－8

長谷川 亨　長谷川 輝之　加藤 聰彦
国際電信電話株式会社 研究所

1. はじめに

近年、大型計算機間における高速ファイル転送などに、800Mbpsまたは1,600Mbpsの伝送速度を提供するHIPPI (High Performance Parallel Interface)[1]が使用されている。しかし、HIPPIを用いて接続した計算機間で伝送速度に対応するスループットを実現することは容易でない。この実現には、プロトコル処理、コンテクスト切替えなどのオペレーティングシステム処理、ユーザ空間とカーネル空間でのデータの移動、キャッシュミスなどのオーバヘッドを総合的に削減する必要がある。そこで、筆者らは、この内、まずプロトコル処理を削減した軽量プロトコルの有効性を評価することを目的として、HIPPI用の軽量プロトコルを設計・実装した。本稿では、その設計・実装の概要ならびに通信実験結果について述べる。

2. HIPPI用の軽量プロトコル

2.1 HIPPIの概要

HIPPIの物理層HIPPI-PH[1]はコネクション型の手順であり、パケットはコネクション上で256ワード長(1ワードは4または8バイト)のバーストに分割されて転送される。HIPPI-PH上のHIPPI-FP[2]は、複数の上位プロトコルを識別して転送可能である。HIPPI-FPは、1016バイトまでの制御情報を転送するD1エリアと、上位プロトコルのプロトコルデータ単位(PDU)を転送するD2エリアから構成されるフレームを規定する。

2.2 設計方針

以下の方針に従って、信頼性のある双方向通信を提供する、HIPPI用の軽量プロトコルを設計した。

(1) HIPPIを使用するアプリケーションプログラムでは、数百キロバイトから数メガバイトのデータをメッセージ単位で交換するため、TCPのようなバイトストリームではなく、サービスデータ単位(SDU)毎の転送機能を提供する。

(2) TCP/IPなどの他のプロトコルと同時に動作可能なように、HIPPI-FPの上位プロトコルとして実現する。

(3) HIPPIを用いて接続した計算機どうしの通信を対象とし、HIPPIの有する機能を利用してプロトコル処理を簡素化する。

(4) 受信処理は送信処理に比較して重いため、可能な処理は送信側で実行し、受信側の負担を軽くする[3]。

(5) HIPPIの伝送品質は高く、パケット紛失の確率は低いため、誤り再送は簡便な方法を用いる。

(6) 具体的なプロトコルとしては、ATM上の確認型データ転送プロトコルのSSCOP(Service Specific Connection Oriented Protocol)[4]をベースに、HIPPI向きに簡素化する手法を用いる。

2.3 機能

以下に軽量プロトコルの主な機能を示す。

(1) PDUの最大長以上のSDUを転送できるように、SDUの分割・組み立てを実現する。

(2) 受信側でSDUに相当するユーザデータだけを簡単に切り出せるように、PDUの制御情報をD1エリアで転送し、PDUのユーザデータをD2エリアで転送する。

(3) PDU処理が簡潔になるように、D1エリアで転送するPDUの制御情報は、48バイト長固定で、パラメタは4バイト境界にアラインしている。また、オプショナルなパラメタは持たず、フォーマットも固定である。

(4) データ転送フェーズにおいて、プロトコルプログラムがコネクションを高速に識別できるように、コネクション確立時に識別子を割り当てる。

(5) 送達確認(ACK)については、送信側がデータ(DT)PDUのACK要求ビットあるいはPOLL PDUを用いて、受信側に明示的に要求する。ACK要求にも順序番号を付与して、ACK要求とACKの対応づけを可能とする。図1の正常時のシーケンスに示すように、受信側はACKを要求するDTを受信する度にACKを返送する。ACKでは、ACK要求の順序番号と送達確認するDTの順序番号を転送している。一方、受信側がDTの抜けを検出した場合は、NACKにより再送を要求する。また、ACK受信時にもDTの抜けを検出すると再送を行う。ただし、無駄なDTの再送が無いように、NACKを受信した送信側は、これ以前に送信したACK要求に対応するACKが再送を要求しても、DTを再送しない。

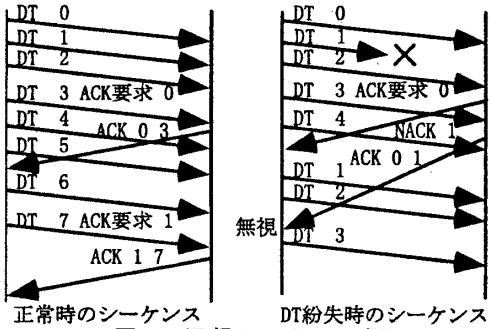

図1: 通信シーケンス例

(6) 再送に関しては、Go-back-Nによる再送を採用し、

"A Study on Light Weight Protocol over HIPPI"
Toru HASEGAWA, Teruyuki HASEGAWA and Toshihiko KATO
KDD R & D Laboratories

一方、フロー制御に関しては、受信バッファの残量をクレジットとして返送する方法を採用する。
(7) HIPPI-PH ではパリティを用いた誤り検査を行うため、CRC などの誤り検査を行わない。

3. 実装

SGI ONYX ワークステーションならびに富士通 VPX210 スーパコンピュータ上に、D1 エリアおよび D2 エリアの読みだし (read)・書き込み (write)、ならびに poll システムコールを用いた入出力の監視を提供する HIPPI-FP のアプリケーションプログラムインタフェース (API) を用いて、軽量プロトコルを実装した。実装したプログラムはユーザレベルで実行されるライブラリであり、図2に示すように、送信・受信ならびにタイマを実現するルーチン群と、コネクション管理テーブルなどから構成される。テーブルは、ウィンドウの上限、下限などのプロトコルの状態を保持する。

送信・受信処理はこのテーブルと SDU を管理するキューを用いて行う。例えば、受信処理は以下の通りである。まず、poll システムコールにより PDU の受信を待つ。PDU を受信すると、D1 エリアを HIPPI ボードから読み出し、読み出した制御情報に応じたプロトコル処理を行う。この時、PDU の制御情報に記された SDU の大きさ分の領域を、プログラムライブラリとユーザプログラムの共有バッファ内に確保する。最後に、D2 エリアのユーザデータを HIPPI ボードからこの領域に読み出す。SDU が複数の PDU からなる場合は、最初に必要な領域が全て確保されているため、以降の PDU の D2 エリアをこの領域に書き加えるだけである。

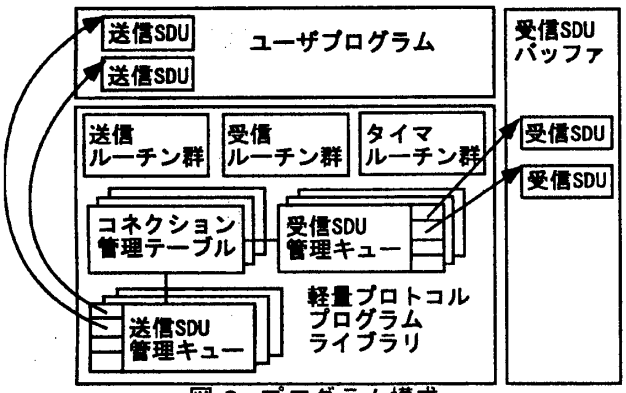

図 2: プログラム構成

4. 評価実験

HIPPI スイッチを介して接続した ONYX(CPU は R4400 150MHz) と VPX210 のメモリ- メモリ間で SDU を転送し、スループットを測定した。図3に、VPX210 から ONYX へ転送した場合のスループットを示す。この時、SDU の分割・組み立ては行っておらず、送信側は 20 個の DT を送信する度に ACK を要求した。さらに、プロトコル処理オーバヘッドを示すために、図には HIPPI-FP のスループットも併せて示している。

図3に示すように、SDU 長に関係なく、プロトコル処理のオーバヘッドは低い。例えば、SDU 長が 61440 バイトの場合、HIPPI-FP の 294.3Mbps のスループットに対して、軽量プロトコルは 272.1Mbps を達成している。

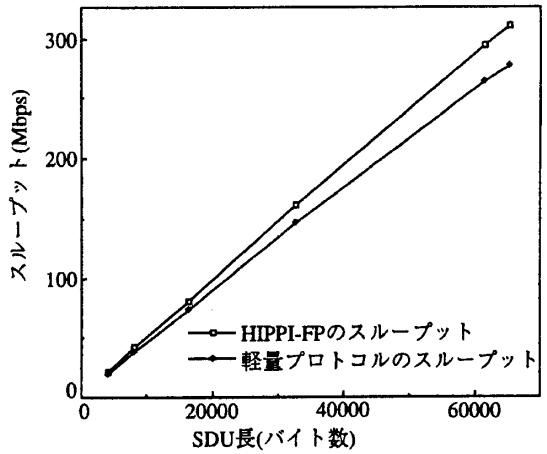

図 3: スループット

次に、プロトコル処理時間を詳しく評価するために、ONYX の提供する 21 ナノ秒の精度のハードウェアクロックを用いて、3 章で示した PDU 受信時の各処理時間を測定した。D2 エリア受信以外の処理時間は、SDU 長にかかわらずほぼ一定であった。そこで、表 1 には、SDU 長が 61440 バイトの場合の測定結果を示している。なお、この PDU 受信では、ACK を返送していない。

表 1: 処理時間の内訳

処理の内容	処理時間 (マイクロ秒)
PDU 受信の監視	39.1
D1 エリアの受信	106.6
プロトコル処理	74.2
D2 エリアの受信	1510.0

表1に示すように、プロトコル処理時間はメモリ管理などを含めても約 74 マイクロ秒にすぎず、プロトコルの軽量化は通信の高速化に有効であったと考えられる。ただし、通信処理時間の多くは、HIPPI ボードとホスト計算機 (ONYX) 間のデータ転送が占めており、この転送を高速化することが重要である。このためには、オペレーティングシステムや HIPPI ボードの処理の軽量化を検討する必要がある。

5. まとめ

本稿では、プロトコル処理を削減した軽量プロトコルの有効性を評価することを目的として、HIPPI 用の軽量プロトコルを設計・実装し、その性能評価を行った。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] ANSI, "HIPPI-PH," X3T9.3/92-REV8.2, Mar. 1993
[2] ANSI, "HIPPI-FP," X3T9.3/Project 702/92-REV4.4, Mar. 1993
[3] W. Doeringer, et.al., "A Survey of Light-Weight Transport Protocols for High-Speed Networks," IEEE Trans. on COMMUNICATIONS, Vol.38, No.11, Nov. 1990
[4] ITU-T, Rec.Q.2110, Nov. 1994