マルチメディア通信と分散処理 78－13
（1996. 9. 27）

ギガビットを実現する通信インタフェースの実装方式に関する一検討

長谷川 亨　　加藤 聰彦

国際電信電話（株）　研究所

〒356 埼玉県上福岡市大原 2-1-15

近年、ATMやHIPPIなどの数百メガビット/秒からギガビット/秒のネットワークの出現に伴い、これらのネットワークを介したギガビットレベルのTCP/IP通信が重要となっている。従来のTCP/IP実装では大半の時間を主記憶上でのパケットのコピーならびにTCPのチェックサム計算が占めているため、ホスト計算機で実行していた機能の一部を通信ボードで実行させることにより、上記のオーバヘッドを低減する研究が行われている。しかし、市販のワークステーションのCPUや主記憶装置の性能向上は著しく、それに伴って、有効なギガビット通信方式を再考する必要も考えられる。そこで本稿では、最新のワークステーションを用いて、従来のTCP/IP実装ならびに代表的なギガビット通信の研究例を実装した場合の性能を推定し、各方式を評価するとともに、その改良方法について検討した結果を示す。

A Study on Implementation of Gigabit Communication Interface

Toru Hasegawa　　Toshihiko Kato

KDD R & D Laboratories

Ohara 2-1-15, Kamifukuoka, Saitama 356, JAPAN

As the network bandwidth becomes more than 1 Gbps, it becomes important to achieve Gigabit TCP/IP communication between high speed workstations. In the traditional TCP/IP implementations, memory copy and checksum calculation prevent high throughput due to the bottle neck of the memory bandwidth of workstations. Therefore, several methods are developed in order to reduce the above overhead by the help of communication interface board. The workstations, however, are becoming faster and faster; therefore, the reevaluation of the methods are required. In this paper, we estimate the throughput of the proposed TCP/IP implementations using high speed workstations and consider their defects and improvements.

1 はじめに

近年、ATM (Asynchronous Transfer Mode) や HIPPI (High Performance Parallel Interface) に代表されるように、数百メガビット/秒 (Mbps) からギガビット/秒 (Gbps) を越える伝送速度を提供可能なネットワークが出現しており、これらのネットワークを介したギガビットレベルの計算機通信が重要となっている [1]。このようなギガビット通信の検討は、大容量データ転送に使用されている TCP/IP を対象として広く行われている。従来 TCP/IP は、4.3BSD における TCP/IP 実装に代表されるように、ホスト計算機上のソフトウェアのみにより実現されていた。その動作を解析した結果、主記憶上でのパケットのコピーならびに TCP のチェックサム計算のオーバヘッドが大半の処理時間を占めているため [2]、通常のワークステーションを用いると約 100Mbps 以上のスループットの実現は困難であると考えられていた。このため、通信ボードを高機能化し、従来ホスト計算機で実行していた機能の一部を通信ボードで実行させ、上記のオーバヘッドを低減する研究が行われてきた [3]。これらの研究では、市販のワークステーションを用いて、約 200Mbps から 400Mbps のスループットを達成している [4-7]。しかし、ワークステーションの CPU や主記憶装置の性能向上は著しく、それに伴って、有効なギガビット通信方式を再考する必要も考えられる。そこで本稿では、最新のワークステーションを用いて、従来の TCP/IP 実装ならびに代表的なギガビット通信の研究例を実装した場合の性能を推定し、各方式を評価するとともに、その改良方法について検討した結果を示す。

2 TCP/IP の高速化技法

従来の TCP/IP 実装において問題であった、主記憶上でのメモリコピーならびにチェックサムのオーバヘッドを削減するために、以下の試みが行われてきた。

(1) ゼロコピー

4.3BSD の実装では、カーネル空間の送受信パケット用のバッファ(ソケットバッファ) とユーザ空間の送受信領域の間でのデータのコピーが最も時間のかかる処理である。そこで、主記憶上の送受信パケットに対するアクセス回数の削減が、多数検討された。これらの試みでは、通信ボード上のメモリ (ネットワークメモリ) をホスト計算機の送受信バッファとして使用するか [4,5]、ホスト計算機上で送受信バッファをカーネルとユーザプロセスで共有するかにより [6,7]、主記憶上の送受信パケットのユーザデータに対するアクセスを、通信ボードとの間の転送時のみとし、データコピーを無くしている。

(2) チェックサムのハードウェア化/オンザフライチェックサム

チェックサムの計算を通信ボード上で実行するか、あるいは通信ボードとホスト計算機の転送時に同時にチェックサムを計算することにより、チェックサム計算のオーバヘッドを削減する。

この 2 つの手法を用いた代表的な研究例として、以下の 3 つがある。

Afterburner は HP (ヒューレットパッカード) ブリストル研究所で開発された、HP 社 PA-RISC ワークステーションのグラフィックバスに装着する通信ボードである。Afterburner によるゼロコピー方式 [4] では、ボード上のネットワークメモリを送受信バッファとして使用し、プログラム I/O を用いて主記憶上にデータ転送し、その際にオンザフライチェックサムを行う。

GigabitNectar [5] は、CMU (カーネギメロン大学) で開発されたネットワークメモリを用いる方式である。Afterburner によるゼロコピー方式と異なり、ネットワークメモリとの通信に DMA を使用し、ハードウェアチェックサム機能を有する。

アリゾナ大学で提案された fbuf [6] は、アリゾナ大学で Mach オペレーティングシステム上に実装され、Sun-Microsystems 社で Solaris 上に実装されている [7]。fbuf では、ホスト計算機上で送受信バッファを用意し、それをカーネルとユーザプロセスで共有する方式を採用している。

そこで 3 章では、従来の 4.3BSD の TCP/IP 実装、4.3BSD の実装にチェックサムのハードウェア化を行った場合、上記の 3 つの研究例について、各方式を最近のワークステーション上で動作させた場合の性能を推定する。

3 従来方式の性能の推定

3.1 推定の条件

以下では、各方式の性能評価を行うため、64K バイトならびに 8K バイトのパケットを受信する処理の処理時間を推定する。SGI 社のマルチプロセサワークステーション Challenge (CPU：MIPS R4400、クロック：150MHz、OS：IRIX5.2) 程度の計算機を使用することを想定し、性能に関して以下の仮定を用いる。

- TCP/IP のプロトコル処理：70 マイクロ秒
  (文献 [8] に報告されている TCP/IP のプロトコル処理時間に基づいて算出)
- 割り込みハンドラの起動：50 マイクロ秒
  (SGI 社マニュアル [9] に基づく)
- コンテキストスイッチ:70 マイクロ秒
  (SGI 社マニュアル [9] に基づく)
- CPU からの主記憶アクセス速度：150 メガバイト/秒 (MB/s)
  (Challenge 上で 64K バイトの bcopy スループットの実測値の 72MB/s より算出)

- 主記憶のバンド幅： 1.2GB/s
  (SGI 社技術レポート [10] に基づく。Challenge では複数の CPU からのアクセス要求をインタリーブし、合計で 1.2GB/s のバンド幅を有する)
- 通信ボートとの DMA 転送速度:200MB/s
  (SGI 社 HIO バスは、64 ビット幅で 47.6MHz で動作する外部バスである。ホスト計算機のデバイスドライバのオーバヘッドを考慮して、この約 6 割とした)
- 通信ボートとのプログラム I/O 速度:40MB/s
  (HIO バスでは、アービトレーションなどに 4 クロック、アドレスの送出/読み出しで 2 クロック必要であり、さらに主記憶への転送では、47.6MHz のシステムバスで 5 クロックは必要である。そこで、5MHz 程度で 64 ビットの読み書きを行うと仮定する)

3.2 4.3BSD の実装

4.3BSD の実装では、図 1に示すように、以下の手順で受信処理を行う。

1. 通信ボードは受信パケットを、ホスト計算機のカーネルバッファに DMA 転送し (64K バイト：320 マイクロ秒、8K バイト：40 マイクロ秒)、ホスト計算機に割り込みをかける。
2. ホスト計算機の割り込み処理ハンドラが起動され (50 マイクロ秒)、受信パケットを受信バッファを管理するための構造 (mbuf 構造体と呼ばれる) に設定してから、IP コードに渡す。
3. カーネルレベルの TCP/IP コードにコンテキストスイッチし (70 マイクロ秒)、プロトコル処理 (70 マイクロ秒)、ならびにチェックサム計算 (主記憶のパケットへアクセスするため、64K バイトの場合は 430 マイクロ秒、8K バイトの場合は 50 マイクロ秒かかる) が行われてから、受信パケットは受信ソケットバッファに繋がれる。
4. ユーザプロセスにコンテキストスイッチし (70 マイクロ秒)、受信パケットの中身をユーザ空間の受信領域にコピーする (64K バイト：860 マイクロ秒、8K バイト：100 マイクロ秒)。

主記憶のバンド幅が 1GB/s であるため、DMA 転送と CPU におけるプロトコル処理がオーバラップ可能と仮定する。従って、64K バイトおよび 8K バイトの 1 パケット処理時間は 1550 マイクロ秒および 410 マイクロ秒であり、受信処理の推定スループットは、それぞれ 338Mbps および 160Mbps である。

3.3 チェックサムのハードウェア化

4.3BSD の実装に対して、チェックサムをハードウェア化すると、1 回主記憶アクセスが削減され、64K バイトおよび 8K バイトのパケット処理時間は 1120 マイクロ秒、および 360 マイクロ秒である。この時の、受信処理の推定スループットは、それぞれ 468Mbps および 182Mbps である。

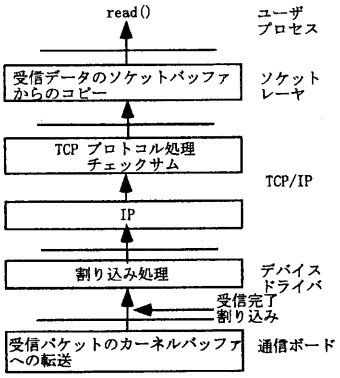

図 1: 4.3BSD 実装での受信処理

3.4 Afterburner によるゼロコピー方式

Afterburner [4] は図 2に示す構成の通信ボードであり、以下の機能を有する。

- ネットワークメモリとして 3 ポートの VRAM を搭載しており、グラフィックバスを介して、プログラム I/O によりホスト計算機から読み書き可能である。
- 送信要求、受信通知などのための FIFO を有する。

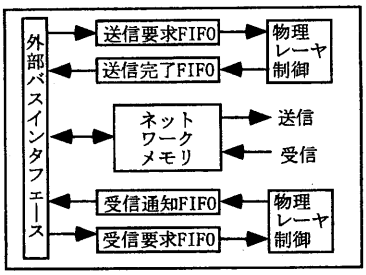

図 2: Afterburner の構成

ネットワークメモリは、ホスト計算機のカーネルのアドレス空間にマップされており、送信/受信パケットのバッファとして使用する。ユーザプロセスが read システムコールを呼び出した時点で、直接通信ボードのネットワークメモリから、ユーザ空間の受信領域に直接転送することにより無駄なコピーを省く。具体的には、以下の手順で受信処理を実行する。

1. 通信ボード (Afterburner) はパケットを受信すると、受信通知を FIFO に書き込み、割り込みをかける。
2. 割り込みハンドラが起動され (50 マイクロ秒)、mbuf 構造体を作成する。mbuf 構造体のデータ本体には、ネットワークメモリ上のパケットが指されている。
3. カーネルにコンテキストスイッチし (70 マイクロ秒)、ネットワークメモリ上のパケットのヘッダを読み出しながらプロトコル処理を実行する (70 マイクロ秒)。
4. read システムコールを呼び出したユーザプロセスにコンテキストスイッチし (70 マイクロ秒)、ネットワークメモリ上のデータをユーザ空間の受信領域にコピーする (64K バイト：1600 マイクロ秒、8K バイト：200 マイクロ秒)。この処理はアセンブリ言語で記述されており、1 ワード単位にネットワークメモリからレジスタに転送し、このレジスタの内容をチェックサム計算用のレジスタに足し合わせてから、ユーザ領域に書き込むため、チェックサム計算のオーバヘッドはほとんど存在しない。

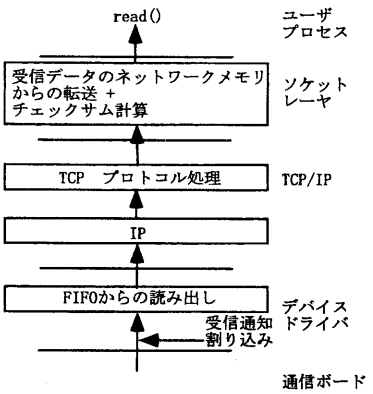

図 3: Afterburner での受信処理

従って、本方式では、64K バイトおよび 8K バイトの 1 パケット処理時間は 1860 マイクロ秒、および 460 マイクロ秒であり、受信処理の推定スループットは、それぞれ 282Mbps および 142Mbps である。

3.5 GigabitNectar

GigabitNectar [5] は図 4に示す構成により、以下の機能を実現する通信ボードを使用する。

- ホスト計算機においてユーザ空間の送受信領域は必ずしも連続的な物理ページを割り当てられないため、通信ボードは scatter-gather の DMA 転送を行う。
- 受信パケットのヘッダを切り出す機能を有する。まず、パケットヘッダに関しては、通信ボードがパケットを受信した時点で、あらかじめ通知されているカーネルバッファに DMA 転送する。一方、ユーザデータに関しては、ユーザプロセスが read システムコールを発行した時に、ユーザ空間の受信領域に直接 DMA 転送する。

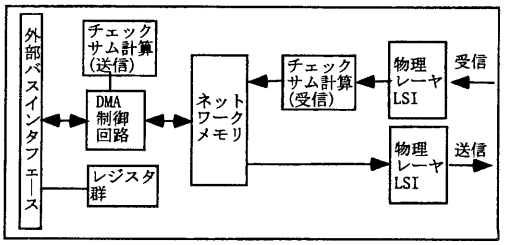

図 4: GigabitNectar の通信ボードの構成

本方式では、通信ボードからの受信終了処理などは、割り込みハンドラなどのカーネルコンテキストで実行されるため、ユーザ空間の送受信領域を送受信の度に、カーネル空間にマッピングする。さらに、DMA 転送するデータは主記憶上に存在する必要があるため、送受信領域を主記憶に固定 (pinn) する。以下では、64K バイトならびに 8K バイトのデータのカーネル空間へのマッピング、メモリへの固定に約 200 マイクロ秒ならびに 70 マイクロ秒 (Challenge での mpin システムコールの測定結果に基づき算出) かかることとする。

本方式の受信処理は以下の通りである。

1. 通信ボードはパケットを受信すると、受信パケットのヘッダのみをメモリに DMA 転送し、割り込みをかける。
2. 割り込みハンドラで受信ヘッダ用の mbuf 構造体を作成する (50 マイクロ秒) 。
3. カーネルの TCP/IP ソフトウェアへコンテキストスイッチし (70 マイクロ秒)、ヘッダだけを参照して、プロトコル処理を行う (70 マイクロ秒)。
4. ユーザプロセスにコンテキストスイッチし (70 マイクロ秒)、read システムコールにおいてユーザ空間の受信領域をカーネル空間にマッピングし、主記憶に固定する (64K バイト：200 マイクロ秒、8K バイト：70 マイクロ秒)。次に、通信ボードに対して、対応するパケットのユーザデータの転送を依頼し、このプロセスはブロックする。
5. 通信ボードは、受信パケットのユーザデータを主記憶へ DMA 転送する (64K バイト：320 マイクロ秒、8K バイト：40 マイクロ秒)。
6. DMA が終了すると、通信ボードは割り込みで終了を通知し、当プロセスにコンテキストスイッチする (約

70マイクロ秒)。最後に、受信領域とカーネル空間をアンマップし終了する。

本方式では、ユーザデータのDMA転送中は、readあるいはwriteシステムコールを発行したユーザプロセスがスリープする必要がある。また、ユーザデータのDMA転送中は外部バスが占有され、通信ボードに新たなパケットを受信してもホスト計算機に通知できない。このため、上記の処理はほぼオーバラップすることなく実行される。従って、64Kバイトおよび8Kバイトのパケットの受信処理時間は、それぞれ約850マイクロ秒ならびに440マイクロ秒であり、推定スループットは、それぞれ617Mbpsおよび149Mbpsである。

3.6 fbuf

fbufでは、以下の機能を有する通信ボードを用いて[11, 12]、ゼロコピーを実現する。

- ユーザ空間に直接DMA転送するため、scatter-gatherのDMA転送を行う。
- ネットワークメモリが無いため、パケット受信時に、そのパケットを受信するユーザプロセスの受信バッファを特定する。
- 送信要求、受信通知を送受するためのFIFOを有する。

以下にSunMicrosystemsによる実現方式[7]を示す(図5参照)。

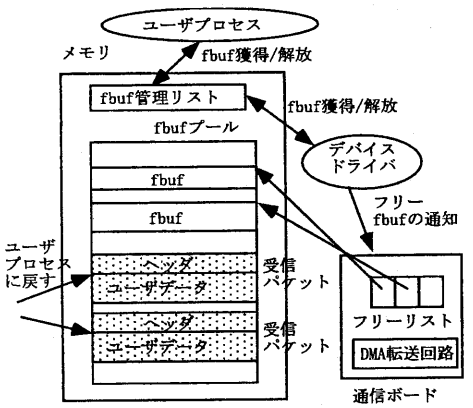

図5: fbufの実現方式

- 通信開始時にメモリ上に送受信パケットを保存するバッファ(fbufと呼ばれる)用の領域ならびにフリーのfbufの管理リストを確保する。これらは、カーネルとユーザ空間にマッピングされ、共有される。また、fbuf用の領域は通信ボードとの間でDMA転送が行われるため、常にメモリに固定しておく。
- ユーザプロセス、あるいはカーネル内のデバイスドライバからのfbufの獲得/解放は、fbuf管理リストに対してスピンロックでアクセス権を得てから行う。
- 通信ボードから直接、受信パケットをfbufにDMA転送できるように、通信ボード上のFIFOに受信パケット用のfbufの先頭アドレスを通知しておく。

本手法では、以下の手順で受信処理が実行される。

1. 通信ボードはパケットを受信すると、受信パケットを指定されたfbufにDMA転送し、割り込みをかける。(64Kバイト:320マイクロ秒、8Kバイト:40マイクロ秒)
2. 割り込みハンドラ内の処理を行う(50マイクロ秒)。
3. カーネルのTCP/IPソフトウェアへコンテキストスイッチし(70マイクロ秒)、プロトコル処理を行う(70マイクロ秒)。
4. ユーザプロセスへコンテキストスイッチし(70マイクロ秒)、readシステムコールでfbufのユーザデータの先頭アドレスを結果として戻す。

64Kバイトの場合、DMA転送とその他の処理が完全にオーバラップしてパイプライン的に処理されれば、DMA転送がボトルネックとなり、1パケットの受信時間は約320マイクロ秒である。一方、8Kバイトの場合は、その他の処理がボトルネックとなり、約260マイクロ秒である。従って、64Kバイトおよび8Kバイトの場合の、推定スループットは、それぞれ1.6Gbpsおよび252Mbpsである。

4 考察

4.1 従来方式の性能推定結果に関する考察

(1) ゼロコピーおよびチェックサム処理の削減の有効性

ゼロコピーならびにチェックサムのハードウェア化は、最新の計算機でも有効であり、TCP/IPのスループットを向上させることが期待される。しかし、Afterburnerによるゼロコピーでは、プログラムI/Oの速度がボトルネックとなり、4.3BSDの実装と同程度のスループットしか実現できない可能性が高い。従って、通信ボードとの通信にDMAを使用することは、ギガビットを実現する通信インタフェースでは必須と考えれる。

(2) fbuf方式の問題点

比較した方式では、fbuf方式が最高のスループットを提供するが、以下の問題点がある。

- fbuf方式では、ユーザプロセスが指定した仮想アドレスではなく、パケットを受信した仮想アドレスがユーザプロセスに読み出しの結果として戻る。さらに、図5に示すように、連続した領域はパケット単位であり、さらにパケットヘッダもfbuf領域に取られるため、読み出した結果が複数のパケットに対応する場合、非連続な領域が戻る。

- カーネル空間とユーザ空間でパケットの保存領域を共有するため、送受信パケットのセキュリティが著しく低下する。例えば、通信ボードがパケットをメモリに転送した後で、ユーザプロセスがカーネルからパケットを読み出す前に、誤ってユーザプロセスがパケットの中身を書き換える可能性がある。

(3) 割り込み/コンテキストスイッチの最小化

いずれの方式を用いても、パケットサイズが8Kバイトの場合は、割り込みならびにコンテキストスイッチ時間が支配的になり、250Mbps 程度以上のスループットの達成は困難である。従来にも、ホスト計算機上の受信キューが0個から1個になった時点で割り込みをかけることにより、パケット受信毎の割り込みを避ける方法 [12] や、一定周期のソフトウェアタイマ発火の度に通信ボードからまとめて受信パケットを転送する方法が提案されている [13] が、さらに検討が必要である。

4.2 従来方式の改良方法

fbuf の高速性をそのまま活かして、ユーザプロセスが受信領域を指定できない問題を解決するには、ユーザプロセスが前もって受信領域のアドレスをデバイスドライバに通知する非同期の通信インタフェースを採用する方法が考えられる。例えば、図6に示す構成で、以下の方法を用いる。

- ユーザプロセスは、受信要求に先立って、fbuf プールから、アドレスを指定して fbuf を確保する。一方、デバイスドライバは、この fbuf を通信ボード上のプログラム I/O 用のメモリ (PIO 用メモリ) 上のフリーリストに書き込む。
- 通信ボードの CPU はパケットを受信すると、GigabitNectar と同様に、ヘッダを切り出してから、ヘッダを受信パケットヘッダ用のカーネル空間に DMA 転送し、ユーザデータを fbuf に DMA 転送する。
- 受信ユーザプロセスからのフリー fbuf の通知が遅れた場合に対処するため、通信ボード上にはネットワークメモリを実装する。
- 通信ボードはパケット紛失が発生しても、通知された fbuf 領域に連続して受信パケットのユーザデータを DMA 転送するため、ホスト計算機はパケット紛失を検出すると、受信用のフリー fbuf を再度通知する。

5 おわりに

本稿では、ギガビットレベルの通信インタフェースの検討を目的として、4.3BSD の実装ならびに代表的な TCP/IP の高速化技法に関して、最新のワークステーションに実装した場合の性能を推定した。この結果、提案されている高速化の方式の内、パケットの送受信バッファをカーネルとユーザプロセスで共有するゼロコピー方式の fbuf が最も高い性能を実現する可能性があることを明らかにし、さらに、fbuf の問題点ならびにその改良方法について検討した。

最後に、日頃御指導頂く KDD 研究所 村上所長に感謝します。

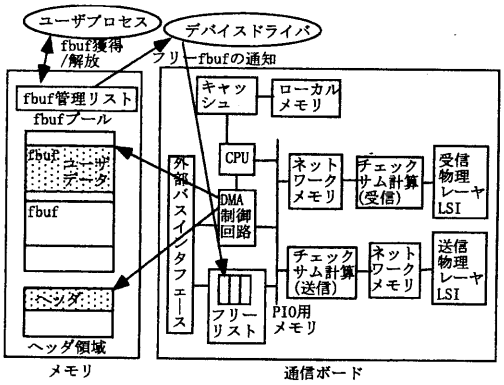

図 6: 実現例

参考文献

[1] C. Partridge, "Gigabit Networking," Addison-Wesley Publishing Company, 1994.

[2] D. Clark, et. al, "An Analysis of TCP Processing Overhead," IEEE Communications, pp. 23-29, June 1989.

[3] P. Steenkiste, "A Systematic Approach to Host Interface Design for High-Speed Networks," IEEE Computer, Vol. 26, No. 3, pp. 47-57, March 1994.

[4] D. Banks, et. al, "A High Performance Network Architecture for a PA-RISC Workstation," IEEE JSAC, Vol. 10, No.1, pp. 191-202, February 1993.

[5] K. Klinpaste, et. al, "Software Support for Outboard Buffering and Checksumming," SIGCOMM 95, August 1995.

[6] P. Druschel, et. al, "Fbufs: A High-Bandwidth Cross-Domain Transfer Facility," Proc. of the 14th SOSP, pp. 189-202, December 1993.

[7] M. Thadani, et. al, "An Efficient Zero-Copy I/O Framework for UNIX," SunMicrosystems Technical Report, SMLI TR-95-39, May 1995.

[8] T. Eicken, et. al, "U-Net: A User-Level Network Interface for Prallel and Distributed Computing," Proc. of the 15th SOSP, December 1995.

[9] Silicon Graphics Computer Systems, "IRIX System Programming Guide" 1992.

[10] M. Galles, et. al , "Peformance optimizations, implementations, and verification of the SGI Challenge multi processor," Silicon Graphics Computer Systems Technical Report

[11] B. Davie, et. al, "The Architecture and Implementation of High-speed Host Interface," IEEE JSAC, Vol. 11, No.2, pp. 228-239, February 1993.

[12] P. Druschel, et. al, "Experiences with a High-Speed Network Adpter," SIGCOMM 94, pp. 2-13, August 1994. on Cache Performance," 4 th ASPLOS, ACM, pp. 8-11, April 1991.

[13] J. Smith, et. al, "Giving Applications Access to Gb/s Networking," IEEE Network, Vol. 7, No. 4, pp. 44-52, July 1993.