既存 OS の枠組みを用いたクラスタシステム向け高速通信機構の提案

住元真司[†] 堀敦史[†] 手塚宏史[†]
原田浩[†] 高橋俊行[†] 石川裕[†]

本論文では，既存 OS の枠組みを用いたクラスタシステム向け高速通信機構を提案する．本機構では，通信の信頼性を確保する軽量通信プロトコルはネットワークインタフェースハードウェアのデバイスドライバより呼び出される．また，アプリケーションによるメッセージ受信待ちのときのハードウェア割込みのオーバヘッドを削減するため，*interrupt reaping* 方式を提案する．本機構を採用した通信機構 PM/Ethernet を Linux 上に実装し，Pentium III 500 MHz PC，Packet Engines 社 G-NIC II 上にて評価した結果，バンド幅 77.5 MB/s，ラウンドトリップ遅延 37.6 μs の通信性能を実現している．TCP/IP では，バンド幅 46.7 MB/s，ラウンドトリップ遅延 89.6 μs である．また，NAS 並列ベンチマーク IS では，TCP/IP 上の MPI の結果に比べ，75%良いことを確認している．

High Performance Communication for Cluster Systems Using an Existing Operating System Framework

Shinji Sumimoto,[†] Atsushi Hori,[†] Hiroshi Tezuka,[†]
Hiroshi Harada,[†] Toshiyuki Takahashi[†] and Yutaka Ishikawa[†]

This paper proposes a scheme to eliminate network protocol processing overhead on an existing operating system for a high performance communication facility. In this scheme, a reliable light-weight network protocol is handled by a network device driver. The *interrupt reaping* technique is proposed to eliminate the hardware interrupt overhead when an application waits for a message. *PM/Ethernet*, an instance of the scheme, achieves 77.5 MB/s bandwidth and 37.6 μs round trip time latency although TCP/IP achieves 46.7 MB/s bandwidth and 89.6 μs round trip time latency using Pentium III 500 MHz PCs. The NAS parallel benchmark IS result shows that MPI on PM/Ethernet achieves 75% better performance than MPI on TCP/IP.

1. はじめに

コモディティハードウェアを用いたクラスタシステムは，コストパフォーマンスの高いクラスタとして広く使われている．このようなクラスタシステムには，100 Mbps Ethernet や Myrinet [1] などのギガビットクラスのネットワークが使われている．ホストプロセッサ性能とメモリバンド幅性能の向上のため，100 Mbps Ethernet と TCP/IP 上の通信ライブラリである MPI や PVM を用いても，物理的なネットワーク性能である 12.5 MB/s に近い性能を出すことができる．Myrinet などのギガビットクラスネットワークを用いる場合，物理的なネットワーク性能を引き出すため，PM [2]，GM [3]，AM [4,5]，FM [6] や BIP [7] など，ハードウェアに特化したネットワークプロトコルとデバイスドライバが設計され実装されている．

Gigabit Ethernet が LAN 環境上で広まるにつれ，次世代のコモディティハードウェアを用いたクラスタシステムは，100 Mbps Ethernet から Gigabit Ethernet を使うようになると予想される．しかしながら，TCP/IP プロトコルは，Gigabit Ethernet の持つ物理的なネットワーク性能を引き出せていない．たとえば，Linux を搭載した Pentium III 500 MHz PC と Packet Engines 社 Gigabit Ethernet カード G-NIC II 上での TCP/IP のバンド幅性能は 46.7 MB/s と，物理的なネットワーク性能である 125 MB/s の半分にも満たないことを測定で確認している．

Gigabit Ethernet の持つ物理性能を引き出すため，我々は GigaE PM [8] を開発した．GigaE PM では，

† 新情報処理開発機構つくば研究センタ
Tsukuba Research Center, Real World Computing Partnership

ホストとネットワークインタフェースカード（以降，NIC）間の情報交換コストについて議論した結果，PM プロトコルと呼ぶ通信の信頼性を確保する軽量プロトコルを Gigabit Ethernet NIC 上に実現している．これは，プロセッサを持つ NIC を利用した場合の高速通信機構の設計という点において有効である．

しかしながら，Gigabit Ethernet NIC のハードウェアのコストダウンにより，プロセッサを持たない NIC が実装されるようになってきた．したがって，GigaE PM の技術的な有効性はあるものの，その適応範囲には制限が出ている．

本論文では，Ethernet 向けの PM 通信機構である PM/Ethernet で採用している既存 OS の枠組みを用いたクラスタシステム向け高速通信機構について提案する．PM/Ethernet では，PM ネットワークプロトコルはホスト上に実装されている．既存のプロトコル処理の解析により，Unix 上の TCP/IP プロトコルの実装のようにプロトコル処理用スレッドでプロトコルを処理する代わりに，PM プロトコル処理は，デバイスドライバの割込みハンドラ処理から直接呼び出される．また，ハードウェア割込みのオーバヘッドを削減するため，*interrupt reaping* 方式を提案し，評価している．*interrupt reaping* 方式は，アプリケーションがメッセージ受信待ちのときにハードウェア割込みを削減する．

PM/Ethernet を Linux 上に実装した．Linux への実装は，既存の Ethernet デバイスドライバへの変更を行わず，かつ，TCP/IP など他の通信プロトコル処理を禁止することなく行われている．評価の結果，Packet Engines 社 G-NIC II を搭載した Pentium III 500 MHz PC 上で，バンド幅 77.5 MB/s，ラウンドトリップ遅延 37.6 μs の通信性能を実現している．

本論文の構成は，2 章では，TCP/IP プロトコル処理を取り上げ，OS 内における TCP/IP プロトコル処理のオーバヘッドを解析する．3 章では，この解析結果をふまえたうえで，既存 OS の枠組みを用いて通信プロトコル処理オーバヘッドを削減する方式の提案を行う．4 章，5 章で，この方式を採用した PM/Ethernet の実装，および，評価として基本通信性能と NAS 並列ベンチマークを TCP/IP と比較する．6 章に関連研究に関して述べる．そして，7 章でまとめる．

2. TCP/IP 処理オーバヘッドの解析

図 1 に示すのは，Network Interface Card（以降，NIC）を含む Linux，および，UNIX ベースの OS の通信プロトコル処理構成図である．一番下に NIC のハードウェアがあり，次にデバイスドライバ，次に Data Link 層（たとえば，Ethernet や FDDI）があり，その上に IP，TCP，および，ICMP が続き，Socket という構成となっている．本章では，実際の通信プロトコル処理層による通信性能の違いを測定し，通信プロトコル処理のオーバヘッドを解析する．測定環境を**表 1** に示す．

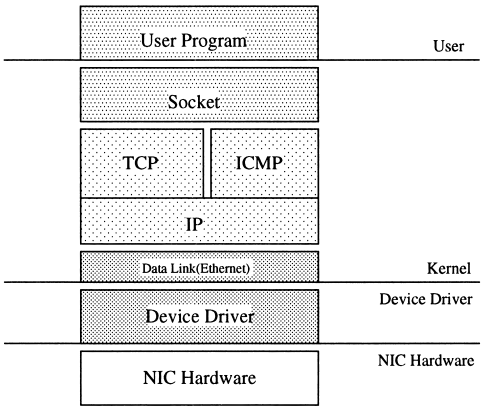

図 1 OS の通信プロトコル処理構成
Fig. 1 Protocol processing in UNIX based Operating System.

表 1 測定環境
Table 1 Evaluation environment.

ハードウェア	Pentium III 500 MHz 搭載 PC （440BX chipset，512 MB SDRAM）
NIC	Packet Engines 社 G-NIC II 33 MHz clock, 64 bit PCI 32 bit PCI スロットにて測定
ホスト OS	Redhat 6.1 Linux（2.2.12 kernel）
デバイスドライバ G-NIC II	hamachi.c:v0.11 8/21/99 Written by Donald Becker

2.1 プロトコル処理層による通信性能の違い

各プロトコル層の通信性能を調べるため，Data Link 層，IP，および，TCP/IP でのラウンドトリップ時間を測定した．加えて，Data Link 層と TCP/IP ではバンド幅性能を測定した．

TCP/IP での測定には，netperf-2.1pl3[9] を用い，IP の測定には，ICMP を用いたプログラムを作成して測定した．また，Data Link 層での測定には以下のドライバと制御のためのユーザプログラムを作成して測定した．

- ラウンドトリップ：送信側では，Ethernet フレーム作成と送信，受信側ではフレーム受信時に，送信側に送り返すドライバを作成して測定．
- バンド幅：送信側で Ethernet フレーム作成と送信，受信側で Ethernet フレームを解放するドラ

表 2 プロトコル処理層の違いによるラウンドトリップ時間（RTT）とバンド幅
Table 2 Round trip time and bandwidth in protocols.

	RTT	バンド幅
TCP/IP プロトコル	89.6 μs	46.7 MB/s
IP プロトコル	58.6 μs	-
Data Link 層	36.6 μs	90.4 MB/s

表 3 TCP/IP オーバヘッド
Table 3 TCP/IP overhead.

処理	オーバヘッド	%
システムコールと socket	1.6 μs	3.6
TCP	15.5 μs	34.6
IP	6.2 μs	13.8
プロトコル処理切替え	3.2 μs	7.1
デバイスドライバ	4.7 μs	10.5
ハードウェア割込み	5.9 μs	13.2
NIC+メディア遅延	7.7 μs	17.2
合計	44.8 μs	100

イバを作成して測定．

表 2 に，Data Link 層，IP と TCP/IP でのラウンドトリップ時間を示す．なお，Data Link 層と TCP/IP についてはバンド幅も示す．

2.2 TCP/IP プロトコル処理オーバヘッドの解析

表 3 に TCP/IP の 1/2 ラウンドトリップにおける算出結果を示す．以下に，それぞれの項目についての算定方法を述べる．

- システムコールと Socket
  Pentium III プロセッサのハードウェア clock counter を用いて測定した．
- TCP
  TCP の 1/2 ラウンドトリップ時間は，IP と TCP 処理時間の和であるゆえ，TCP の処理時間は以下のようになる．
  $$89.6/2 - 58.6/2 = 15.5$$
- IP
  IP の 1/2 ラウンドトリップ時間は，システムコールと socket のオーバヘッド，プロトコル切替え時間，そして Data Link 層時間の和である．したがって，IP の処理時間は以下のようになる．
  $$58.6/2 - 1.6 - 3.2 - 36.6/2 = 6.2$$
- プロトコル処理切替え
  プロトコル処理切替え時間は，Data Link 層と上位層である IP プロトコル処理への切替え時間であり，ソフトウェア割込みにかかる時間に等しい．この処理時間は，ハードウェア clock counter を用いて測定した．
- デバイスドライバ
  このオーバヘッドは，デバイスドライバの実行時間である．この処理時間は，ハードウェア clock counter を用いて測定した．
- ハードウェア割込み
  このコストは，NIC がプロセッサへの割込みレジスタに書き込んだ後，デバイスドライバの割込み処理関数が呼び出されるまでの時間である．このコストは Essential Gigabit Ethernet NIC のファームウェアプログラムを用いて測定した．このコスト 5.9 μs の内訳は，割込みが発生した後，プロセッサが割込み応答するまでの時間が 1.6 μs，割込みコントローラの制御のための I/O レジスタへのアクセスが 4 回（1.2 μs），残りがコンテキストの save やデバイスドライバのエントリ検索などのソフトウェアのオーバヘッドである．
- NIC+メディア遅延
  Data Link 層の 1/2 ラウンドトリップ時間は，デバイスドライバ，ハードウェア割込み，そして NIC+メディア遅延の和である．したがって，NIC+メディア遅延のコストは以下のようになる．
  $$36.6/2 - 4.7 - 5.9 = 7.7$$

表 3 より，オーバヘッドのうち，TCP/IP プロトコル処理の占める割合は 48.4%（34.6 + 13.8）である．次にソフトウェアで削減可能なものとしては，ハードウェアの割込みのオーバヘッドがある．

システムコールと socket のオーバヘッドは，1.6 μs と，全コストのうち 3.6%程度である．このシステムコールと socket が占める割合は，ユーザレベル通信を導入するかどうかを判断するうえで重要である．

たとえば，片道 10 μs 以下の遅延を持つ Myrinet 上の通信機構[2)~7)]の場合は，システムコールと socket のオーバヘッドは全体の 16%を占めることになり，ユーザレベル通信を導入した場合の遅延削減の効果を期待できる．しかし，片道 48.4 μs の G-NIC II 上の TCP/IP の場合は，システムコールと socket のオーバヘッドは 3.6%にすぎず，ユーザレベル通信を導入した場合でも遅延削減の効果を期待できない．

3. クラスタシステム向け通信プロトコル処理機構の設計

3.1 クラスタシステム向け通信プロトコル処理

Linux，および，UNIX ベースのシステムにおける TCP/IP 処理の実装では，プロトコル処理は，ソフトウェア割込みを用いたカーネル内スレッドとして実行される．この方式は，ネットワークはハードディスク

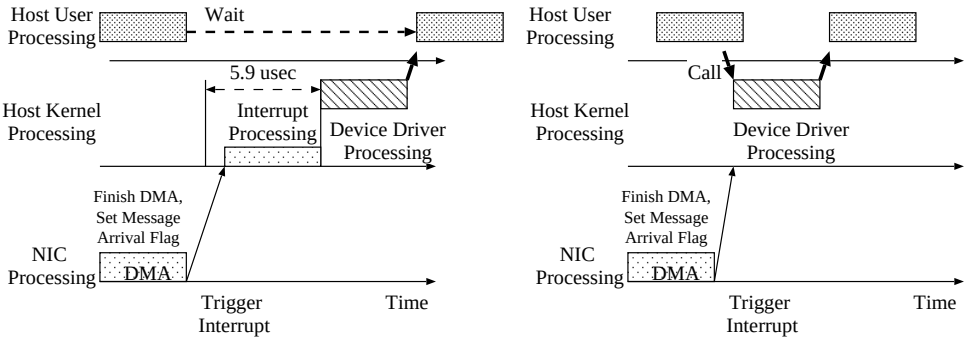

図 2 interurpt reaping 方式
Fig. 2 The interrupt reaping technique.

などのデバイスより遅いものであるという前提に基づいており，ネットワークが，キーボード，マウス，ハードディスクとともにネットワークが同時に使われる環境においては，良い設計であるといえる．

しかしながら，Gigabit Ethernet クラスのネットワークを用いたクラスタシステム向け通信では，ネットワークはハードディスクなどのデバイスより遅いという前提は成り立たず，TCP/IP と同じ枠組みでプロトコル処理を行うとオーバヘッドが大きくハードウェア性能を引き出せない（表 2）．

それゆえ，我々は，メッセージの遅延を最小化するために Data Link 層から直接通信プロトコル処理を呼び出す方式を提案する．

3.2 信頼性のある軽量通信プロトコル

表 3 によれば，TCP/IP のプロトコル処理オーバヘッドは全体の 48.4%を占める．このオーバヘッドは最小限にしなければならない．しかし，たとえば Ethernet のようにネットワークがハードウェアレベルでメッセージ転送を保証していない場合は，メッセージ転送とメッセージの順序を保証しなければならない．このため，メッセージ転送とメッセージの順序を確保するために，GigaE PM ネットワークプロトコル[8]を採用する．

3.3 interrupt reaping 方式

クラスタシステム上での並列計算（特にデータ並列計算）では，メッセージが到着するまで次の処理が実行できないことが多く，メッセージの到着後から次の処理を開始するまでの時間は最小限にすべきである．

しかし，通常の OS 処理では，メッセージ受信処理実行時にメッセージがない場合，メッセージ受信待ちが発生する．通常の OS 処理におけるメッセージ処理を**図 2** (a) に示す．メッセージ待ちが発生した場合は，ハードウェア割込み処理（図 2 (a) 中 Interrupt Processing）の後，受信メッセージが処理され（図 2 (a) 中 Device Driver Processing），その後，ユーザプログラムはメッセージ処理することが可能になるが，メッセージ到着後，NIC から OS にハードウェア割込みが入り，実際のデバイスドライバの処理に入るまで時間がかかる．たとえば，表 3 に示すとおり Pentium III 500 MHz のシステムでハードウェア割込みは 5.9 μs の時間を要する．

そこで，既存 OS の枠組みを利用し，ハードウェア割込みのオーバヘッドを削減できる *interrupt reaping* 方式を提案する．

interrupt reaping 方式では，メッセージ受信待ちのとき，ユーザプログラムが NIC のデバイスドライバの割込みハンドラを直接実行してメッセージ受信処理を行う（図 2 (b)）．この実行はハードウェア割込みを禁止して行うため，メッセージ受信処理中は割込みは入らず，メッセージの処理後は，デバイスドライバの割込みハンドラが割込み発生用レジスタを更新し割込みを止める．

本処理方式は，PCI バスのように複数のデバイスが割込みレベル（IRQ）が共有でき，デバイスドライバが他のデバイスと IRQ を共有可能なように書かれていれば動作する．なぜなら，このようなデバイスドライバには，まず自分宛のハードウェア割込みかをチェックするコードが書かれているからである．しかし，他のデバイスと IRQ を共有可能になっていないデバイスドライバは，ハードウェア割込みが入っていない場合にエラー，あるいは，割込みが入ったものとして処理する場合があり，動作しない可能性がある．

このように，*interrupt reaping* 方式の導入により，メッセージ受信待ち時のハードウェア割込み処理の終了を待つことなくメッセージ受信処理が実行できる．

4. PM/Ethernet の実装

3 章に述べた方式を用いて，Gigabit Ethernet 向けのクラスタ通信機構 PM/Ethernet を実装した．

4.1 PM/Ethernet のアーキテクチャ

PM/Ethernet は，PM[2]で採用されたチャネルと呼ばれる仮想ネットワークを提供しており，TCP/IP のようにコネクション型の通信ではなく，信頼性のあるデータグラム通信を実現している．並列アプリケーションの各プロセスは，チャネルを排他的に利用し，同じ番号のチャネルで通信することにより通信を行う．チャネルの数は，Myrinet，Ethernet などネットワークごとにシステム資源で制約がある．たとえば，PM/Ethernet は，16 チャネルをサポートしている．チャネル数以上の並列アプリケーションプロセスを実行できるように，SCore を開発している[10]．

PM/Ethernet のアーキテクチャを**図 3** に示す．PM/Ethernet は，PM のインタフェースを実現する PM/Ethernet ライブラリ，通信プロトコルとデバイスドライバの制御を行う PM/Ethernet 部，および，Data Link 層にフレームのタイプにより PM/Ethernet か IP その他のプロトコルのフレームかを切り分けるコードが埋め込まれている．その下には，既存の Ethernet デバイスドライバがある．

メッセージの送受信処理は `ioctl()` を用いて行われる．メッセージ送信時は，PM/Ethernet ライブラリより呼び出される PM/Ethernet 部でメッセージを Ethernet フレーム上に作成する．そして，デバイスドライバの送信機構を利用してメッセージを送信する．また，メッセージ受信時には，デバイスドライバ処理の後，Data Link 層処理でフレームを分配する．PM/Ethernet 向けのフレームであった場合は PM/Ethernet の処理を，そうでない場合は既存プロトコル処理を行う．

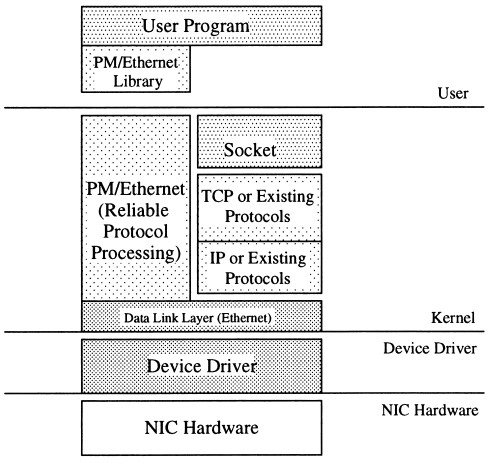

図 3 PM/Ethernet のアーキテクチャ
Fig. 3 The PM/Ethernet architecture.

4.2 信頼性を確保した軽量通信プロトコル

Ethernet は，ハードウェアレベルでメッセージ転送を保証していないので，メッセージの到着と順序性を保証する軽量通信プロトコルが必須である．PM/Ethernet は，GigaE PM[8]と同様にメッセージの到着と順序性を保証する軽量通信プロトコルとして，処理が簡単で性能が出せるため **GO back N** プロトコルを採用している[11]．**GO back N** プロトコルでは，送信側は i 番目から $(i+N)$ 番目までのデータを受信側からの ACK メッセージを待たずに送信できる．受信側は i 番目のメッセージの受信時に送信側に ACK メッセージを送信する．送信側は i 番目の ACK メッセージを受信後に $(i+1)$ 番目から $(i+1+N)$ 番目のデータを受信側に送信できる．

もし，送信側が既定時間内に i 番目の ACK メッセージを受信しない場合は，送信側は i 番目とそれに続くメッセージを再度送信する．これは，i 番目のデータが失われたことを意味する．もし，受信側がシーケンス番号 i より大きな番号のメッセージを受信した場合，受信データは廃棄され，送信側に **LOSE** メッセージを送信する．

なお，受信側のメッセージバッファがフルになった場合の対策として，**STOP and GO** フロー制御を採用している．

4.3 Linux への実装

我々は，Linux 上に PM/Ethernet を実装した．PM/Ethernet のコードのほとんどの部分は，デバイスドライバとして実現されているが，以下に述べる点については Linux 2.2.12 のカーネルコードを変更している．

- デバイスドライバの割込みハンドラをカーネル内のプリミティブで呼び出すため，デバイスドライバの割込みハンドラの関数へのポインタと引数をネットワークデバイスの構造体に格納する．
- Data Link 層で PM/Ethernet 用の Ethernet フレームを振り分けるためにディスパッチャを挿入する．
- PM/Ethernet の通信プロトコルとユーザインタフェースを実現する PM/Ethernet デバイスドラ

イバを追加している．このドライバはダイナミックロード可能なドライバとなっている．

4.4 interrupt reaping 方式の実装

interrupt reaping の処理はデバイスドライバで実現している．カーネル内の実行手順を以下に示す．

```
if(ret = epm_get_message(channel)
   == NO_MESSAGE)) {
  disable_interrupt();    // disable interrupt
  do_interrupt_handler(); // execute handler
  enable_interrupt();     // enable interrupt
  ret = epm_get_message(channel);
}
```

do_interrupt_handler() は，NIC のデバイスドライバの割込み処理関数を呼び出すコードである．PM プロトコル処理は，NIC のデバイスドライバの割込み処理関数より直接呼び出される PM/Ethernet 用の Ethernet フレームを振り分けるためのディスパッチャを介して実行される．最後に epm_get_message() を再度呼び出しているのは，システムコールのオーバヘッドを抑えるためである．

デバイスドライバ内ではメッセージ受信待ちは行わず，メッセージ受信待ちを行う場合には，ユーザのプログラムでメッセージ受信処理を繰り返し実行することにより実現する．

5. 評　　価

5.1 通 信 性 能

基本通信性能としてアプリケーションレベルのバンド幅とラウンドトリップ時間を TCP/IP と比較する．測定に用いた環境は表 1 と同じである．なお，通信性能測定時 Switch は利用していない．TCP/IP の通信性能は netperf-2.1pl3 [9]，PM/Ethernet は pmtest [12] コマンドを用いて測定を行った．

5.1.1 アプリケーションレベルバンド幅性能

図 4 に，PM/Ethernet，TCP/IP，そして *interrupt reaping* 機能を無効にした場合の PM/Ethernet のアプリケーションレベルのバンド幅性能を示す．図中，IR は *interrupt reaping* を意味する．

図 4 の結果より，TCP/IP では，1,280 バイトメッセージの場合に，46.7 MB/s のバンド幅性能であるが，PM/Ethernet は 1,468 バイトメッセージの場合に 77.5 MB/s のバンド幅性能を実現している．また，*interrupt reaping* 機構を無効にした場合のバンド幅は 1,440 バイトメッセージの場合に 64.6 MB/s である．

5.1.2 アプリケーションレベルラウンドトリップ時間

図 5 にアプリケーションレベルのラウンドトリップ時間の測定結果を示す．図では，PM/Ethernet，TCP/IP，そして *interrupt reaping* 機能を無効にした場合の PM/Ethernet のラウンドトリップ時間を示す．

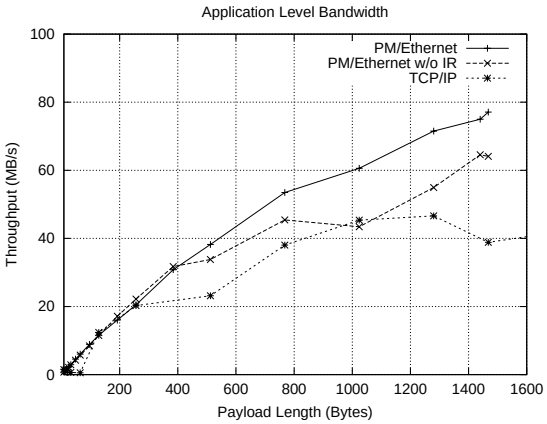

図 4 アプリケーションレベルバンド幅性能
Fig. 4 Application level bandwidth.

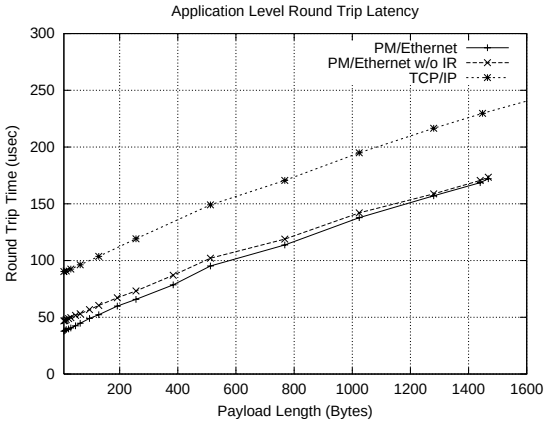

図 5 アプリケーションレベル遅延（RTT）時間
Fig. 5 Application level round trip time.

図 5 より，4 バイトメッセージ時において，TCP/IP は 89.6 μs，PM/Ethernet は 37.6 μs のラウンドトリップ時間を実現している．また，*interrupt reaping* 機能を無効にした場合の PM/Ethernet は 4 バイトメッセージ時に 47.6 μs のラウンドトリップ時間である．

5.2 NAS 並列ベンチマーク：IS

実際のアプリケーション性能を見るために，PM/Ethernet 上の MPICH で NAS 並列ベンチマーク ver 2.3 [13] IS Class A を測定し，その差を TCP/IP と比べる．TCP/IP の MPI としては，LAM [14] を用いた．なお，測定に用いた環境は表 1 と同じであり，Switch として 3Com 社の SuperStack II Switch 9300 を用いている．

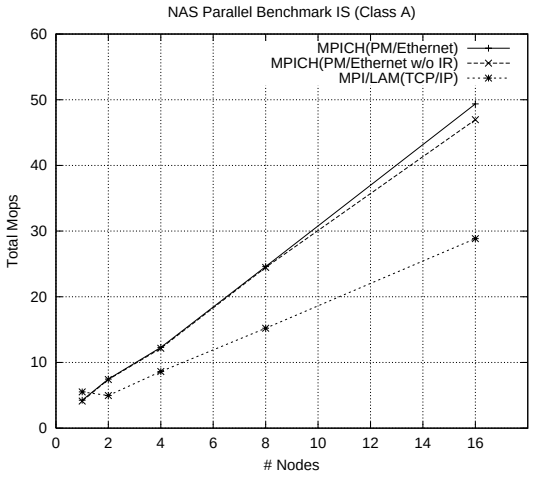

図 6 NAS 並列ベンチマーク IS（クラス A）
Fig. 6 NAS parallel benchmarks IS (Class A).

図 6 に結果を示す．PM/Ethernet 上の MPI は 16 ノードにおいて 11.8 倍の性能向上，*interrupt reaping* 機構を無効にした場合は，16 ノードで 11.2 倍の性能向上を示している．一方，TCP/IP 上での MPI では，16 ノードで 5.2 倍の性能向上を示している．PM/Ethernet 上の MPI の IS の結果は，TCP/IP 上の MPI の IS の結果より，1.75 倍の性能向上を示している．

5.3 interrupt reaping 方式の効果

本節では，*interrupt reaping* 方式の効果を検証する．3.3 節で述べたように，*interrupt reaping* 方式を導入することにより，ハードウェア割込みのオーバヘッドの削減を期待できる．表 3 より，割込みのオーバヘッドは 1/2 ラウンドトリップ時間で 5.9 μs なので，ラウンドトリップ時間では，*interrupt reaping* 方式により 11.8 μs の時間の削減が見込めることになる．

これを 5.1.2 項の結果と比較する．5.1.2 項より，*interrupt reaping* 機構を有効にした場合のラウンドトリップ時間が 37.6 μs に対して，*interrupt reaping* 機構を無効にした場合の PM/Ethernet は 47.6 μs のラウンドトリップ時間であった．

結果，ハードウェア割込みのオーバヘッドである 11.8 μs のうち，*interrupt reaping* 機構によりラウンドトリップ時間は 10.0 μs 削減されている．残り 1.8 μs の差が出るが，この時間は *interrupt reaping* 機構のために呼び出したデバイスの割込みハンドラ処理時間である．

また，NAS 並列ベンチマークの ver 2.3 IS Class A において，*interrupt reaping* 機構を入れることにより，16 ノードで 5%の性能向上が得られている．

表 4 1/2 ラウンドトリップ時間におけるプロトコル処理プロトコル処理オーバヘッド比較
Table 4 Comparison of protocol processing cost analysis on 1/2 of the round trip time.

処理	TCP/IP	PM/Ethernet
システムコールと socket	1.6 μs	1.6 μs
TCP	15.5 μs	-
IP	6.2 μs	-
PM	-	4.8 μs
プロトコル処理切替え	3.2 μs	-
デバイスドライバ	4.7 μs	4.7 μs
ハードウェア割込み	5.9 μs	-
NIC+メディア遅延	7.7 μs	7.7 μs
Total	44.8 μs	18.8 μs

5.4 PM/Ethernet のプロトコル処理オーバヘッドの解析

表 4 に，TCP/IP と PM/Ethernet のプロトコル処理オーバヘッドの比較を示す．表 4 より，PM/Ethernet は，3 章で提案した方式を採用したことにより，プロトコル処理切替えとハードウェア割込みのオーバヘッドを削減し，プロトコル処理オーバヘッドも 77.9%削減していることがわかる．

6. 関連研究

クラスタシステム向けに，数多くの高性能通信機構が開発されている．

AM [4),5)]，AM-II [15)]，FM [6)]，GM [3)]，BIP [7)]，VMMC-2 [16)] と PM [2)] は Myrinet [1)] をベースとしており，Myrinet 上のファームウェアとユーザレベル通信により実装されている．Myrinet はギガビットクラスのネットワークでハードウェアレベルでメッセージ転送を保証している．それゆえにネットワーク上でメッセージが失われることは考える必要はなく，受信バッファのフロー制御を行えばよい．

Ethernet NIC を用い TCP/IP を利用しないで高い通信性能を実現しているものに U-Net [17)] がある．U-Net は，システムコールと割込みのオーバヘッドを減らすためにユーザレベル通信を採用している．PM/Ethernet は，カーネル内で既存の Ethernet デバイスドライバを用いながら，割込みのオーバヘッドを回避している点が異なる．

VIA [18)]（Virtual Interface Architecture）はマイクロソフトの Windows 上のギガビットクラスのネットワークに広く実装されている．VIA はその上に socket や MPI のような通信ライブラリが実装できるように設計されている．VIA はコネクションベースの通信をサポートしており，信頼性のある通信は VIA specification Version 1.0 ではオプションである．

以上述べた，どのクラスタシステム向けの通信機構も，特定のネットワークデバイスに依存したコードを持つ通信機構であるが，PM/Ethernet は，既存の OS の枠組みを用い，かつ，既存のネットワークデバイスドライバを変更することなく，高い通信性能を実現している点が異なる．

7. ま と め

本論文では，既存 OS の枠組みを用いたクラスタシステム向けの通信機構における OS の通信プロトコル処理オーバヘッドの削減方式を提案した．本方式では，信頼性のある軽量通信プロトコルは，TCP/IP のソフトウェア割込みを用いて実行されるのではなく，ネットワークデバイスドライバから直接呼び出される．また，低遅延通信に有効な手段として，*interrupt reaping* 方式を提案した．

本論文で述べた OS の通信プロトコル処理オーバヘッドの削減方式と *interrupt reaping* 方式を採用した PM/Ethernet は，Linux 上に既存の Ethernet デバイスドライバを変更することなく実現されている．それゆえ，PM/Ethernet は，他の多くの Ethernet デバイスドライバ上で利用可能である．

PM/Ethernet を Pentium III 500 MHz PC, Packet Engines 社 G-NIC II 上にて評価した結果，TCP/IP では，バンド幅 46.7 MB/s，ラウンドトリップ遅延 89.6 μs に比べ，既存 OS の枠組みを使いながら，バンド幅 77.5 MB/s，ラウンドトリップ遅延 37.6 μs の通信性能を実現している．

NAS 並列ベンチマーク IS Class A を用いた評価でも，PM/Ethernet を用いることにより，TCP/IP を用いる場合に比べ，75%性能が向上している．*interrupt reaping* 方式を用いた場合の結果は，*interrupt reaping* 方式を用いない場合の結果より 5%の性能向上である．

参 考 文 献

1) Boden, N.J., Cohen, D., Felderman, R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N. and Su, W.-K.: Myrinet – A Gigabit-per-Second Local-Area Network, *IEEE MICRO*, Vol.15, No.1, pp.29–36 (1995).
2) Tezuka, H., Hori, A., Ishikawa, Y. and Sato, M.: PM: An Operating System Coordinated High Performance Communication Library, *High-Performance Computing and Networking*, Hertzberger, P.S.B. (Ed.), Vol.1225 of Lecture Notes in Computer Science, pp.708–717. Springer-Verlag (1997).
3) http://www.myri.com/GM/doc/gm_toc.html.
4) von Eicken, T., Culler, D.E., Goldstein, S.C. and Schauser, K.E.: Active Messages: A Mechanism for Integrated Communication and Computation, *Proc. 19th ISCA*, pp.256–266 (1992).
5) von Eicken, T., Avula, V., Basu, A. and Buch, V.: Low-Latency Communication over ATM Networks Using Active Messages, *Proc. Hot Interconnects II*, Palo Alto (1994).
6) Pakin, S., Lauria, M. and Chein, A.: High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet, *Proc. Supercomputing '95*, San Diego, California (1995).
7) Prylli, L. and Tourancheau, B.: BIP: A new protocol designed for high performance, *PC-NOW Workshop, Help in Parallel with IPPS/SPDP98*, Orlando, USA (1998).
8) Sumimoto, S., Tezuka, H., Hori, A., Harada, H., Takahashi, T. and Ishikawa, Y.: The Design and Evaluation of High Performance Communication Using a Gigabit Ethernet, *International Conference on Supercomputing '99*, pp.243–250. ACM SIGARCH (1999).
9) http://www.netperf.org/.
10) 堀 敦史，手塚宏史，高橋俊行，住元真司，曽田哲之，原田 浩，石川 裕：クラスタ上のプログラミング開発環境—SCore クラスタシステムソフトウェア，情報処理学会研究報告 99-HPC-77（SWoPP '99），pp.83–88 (1999).
11) Bertsekas, D. and Gallager, R.G.: *Data Networks*, Perntice-Hall (1987).
12) http://pdswww.rwcp.or.jp/dist/score/reference/man/man8/pmtest.html.
13) http://www.nas.nasa.gov/Software/NPB/.
14) http://www.mpi.nd.edu/lam/.
15) Mainwaring, A.M., Chun, B.N. and Culler, D.E.: Virtual Network Transport Protocols for Myrinet, *Hot Interconnect '97* (1997).
16) Chen, Y., Damianakis, S., Dubnicki, C., Bilas, A. and Li, K.: VMMC-2: Efficient Support for Reliable, Connection-Oriented Communication, *Hot Interconnect '97* (1997).
17) Basu, A., Buch, V., Vogels, W. and von Eicken, T.: U-Net: A User-Level Network Interface for Parallel and Distributed Computing, *Proc. 3rd International Symposium on High Performance Computer Architecture* (*HPCA*) (1997).
18) http://www.viarch.org/.

(平成 11 年 12 月 10 日受付)
(平成 12 年 4 月 6 日採録)

住元　真司（正会員）

1986年同志社大学工学部電子工学科卒業．同年（株）富士通入社.（株）富士通研究所にて並列オペレーティングシステム，並列分散システムソフトウェアの研究開発に従事．1997年より新情報処理開発機構に出向．コモディティネットワークを用いた高速通信機構の研究開発に従事．並列分散システムのアーキテクチャ，システムソフトウェア等に興味を持つ．

堀　　敦史（正会員）

1979年早稲田大学理工学部電気工学科卒業．1981年同大学院理工学研究科計測制御工学専攻修士課程修了．同年（株）三菱総合研究所入社．1992年より技術研究組合新情報処理開発機構に出向．JSPP '98最優秀論文賞受賞．並列オペレーティングシステムの研究に従事．並列プログラミング言語，並列アーキテクチャ等に興味を持つ．工学博士（東京大学工学部）．

手塚　宏史（正会員）

1980年東京大学教養課程中退．1981年（株）生活構造研究所入社．1985年ソニー（株）入社．1988年（株）ソニーコンピュータサイエンス研究所入社．1990年ソニー（株）入社．1993年北陸先端科学技術大学院大学研究生．1995年より技術研究組合新情報処理開発機構研究員．現在に至る．オペレーティングシステム，リアルタイム処理，マルチメディア処理等に興味を持つ．日本ソフトウェア科学会会員．

原田　　浩（正会員）

1988年東京理科大学理学部物理学科卒業．同年（株）ソフトウェア・リサーチ・アソシエイツ入社．1997年より技術研究組合新情報処理開発機構研究員．現在に至る．オペレーティングシステム，並列・分散システム等に興味を持つ．ACM会員．

高橋　俊行（正会員）

1993年東京理科大学理工学部情報科学科卒業．1995年同大学院修士課程修了．1995〜1998年東京大学理学系研究科情報科学科博士課程．1998年より新情報処理開発機構研究員．現在に至る．プログラミング言語におけるメタレベルアーキテクチャと並列計算ソフトウェア技術に興味を持つ．理学修士．

石川　　裕（正会員）

1987年慶応義塾大学大学院理工学研究科電気工学専攻博士課程修了．同年電子技術総合研究所入所．1988〜1989年カーネギー・メロン大学客員研究員．1990年日本ソフトウェア科学会高橋奨励賞を受賞．1993年から新情報処理開発機構に出向．並列・分散システム，適応可能並列プログラミング言語/環境/処理系，リアルタイム処理等に興味を持つ．日本ソフトウェア科学会，ACM，IEEE各会員．工学博士．