社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－179 (28)
2008／8／7

ExpEtherにおけるRDMA通信のためのソフトウェア環境の構築

今田啓介[†] 酒井洋介[†] 大塚智宏[†]
鈴木順[††] 樋口淳一[††]
飛鷹洋一[††] 天野英晴[†]

NECにより開発が進んでいるExpEtherは，PCI ExpressとEthernetを統合するネットワークインタフェースである．本稿では，ExpEtherを対象とするRDMA通信機構を実装したネットワークインタフェースコントローラを利用するためのソフトウェア環境がRDMA通信機構の性能に与える影響についての評価を述べる．実験用システムにおいて，RDMA通信に必要な通信用バッファのPhysical Buffer List (PBL)の取得に要する時間を測定した結果，1MByteの領域に対して8.35μsecで済み，同じ領域に対するピンダウン処理の時間の半分程度であり，十分実用的であることがわかった．また，PCI Express NICに対するメモリアクセスをユーザレベルで行った場合のメモリアクセスレイテンシは0.58μsecであり，カーネルを経由した場合と比べて54.3%削減できることを確認した．

Software Environment for RDMA communication over ExpEther

Keisuke Imada,[†] Yosuke Sakai,[†] Tomohiro Otsuka,[†]
Jun Suzuki,[††] Junichi Higuchi,[††] Youichi Hidaka[††]
and Hideharu Amano[†]

ExpEther by NEC is a network interface for a bridge between PCI Express and Ethernet for network connected virtual computer environment. In this paper, evaluation of the software environment which supports access to ExpEther network interface card (NIC) is described. On our experimental system, it takes 8.35μsec to get Physical Buffer List (PBL) for RDMA data transfer using 1MByte buffer. It is almost a half of time for pin-down the same memory area, and practical. The user-level memory access latency was 0.58μsec, and the overhead of using the kernel corresponding to 54.3% of execution time is removed.

1. はじめに

近年，ハードウェアプラットフォームにおいて，パーソナルコンピュータ(PC)やサーバ，ストレージ，ゲートウェイルータなどのデバイスを高速なネットワークで接続することにより，高い性能と信頼性を持ち拡張性にも富んだシステムが，安価に構築できるようになっている．

このようなシステムを構築するために用いるネットワークには，高い通信性能が要求され，様々なネットワーク規格が混在している．その結果，ハードウェアプラットフォームは高度で複雑なものとなり，その管理およびその上に構築するシステムの運用は容易ではない．

† 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†† 日本電気株式会社
NEC Corporation

このような状況の中，日本電気株式会社(NEC)[1]が研究開発を行うネットワークインターフェースであるExpEther[2]は，システムの内部バスとして広く普及しているPCI Express[3]と，Local Area Network (LAN)の標準規格であるEthernetを統合し，容易なハードウェア管理およびシステム運用を可能とする．ExpEhterを介して接続するEhternet LANは，Gigabit Ethernet[4]以降の高速なEthernet規格を用いて構築され，既存のクラスタ向けのインターコネクトと同等の高い性能が期待できる．

ExpEtherでは，システム内部におけるPCI Expressバス接続が，Ethernetによる LAN接続へと延長されるI/O仮想化がなされている(図1左)．この仮想化技術によってExpEtherを介してコンピュータに接続されたI/Oデバイスは，仮想的にシステムの内部バスを介してコンピュータに接続されたと見なすことができる．このI/O仮想化技術によって，コンピュータとExpEtherを介したI/Oデバイス群との間の接続が可能となる．

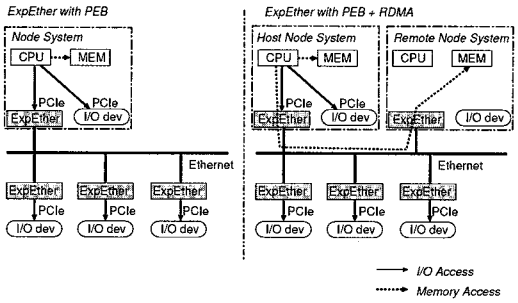

図 1 ExpEther システム構成

しかし，これまでの枠組みでは，コンピュータ-コンピュータ間の接続は，実現されていなかった．そこで我々は，ExpEther を対象としたネットワークインタフェースコントローラに Remote Direct Memory Access (RDMA) 通信機能を実装することで，ハードウェアレイヤでのメモリ仮想化を行い，ExpEther によるコンピュータ-コンピュータ間接続を実現する (図 1 右)．RDMA 通信機構の設計には，RDMA 通信プロトコルの標準化に向けてまとめられた iWARP[5] の仕様を採り入れ，上位互換性の獲得を狙う．

ExpEther を対象とする RDMA 通信機構を目的とした本研究においては，これまでに，ハードウェアに関する評価として，RDMA 機構を実装したネットワークインタフェースコントローラ (NIC) 上のコアモジュールにおける通信性能の評価[6] がなされいる．本稿では，同 NIC を用いた RDMA 通信機構において，アプリケーションが利用するソフトウェア環境がネットワークシステムの通信性能に与える影響についての評価を示す．

2. ExpEther

ExpEther の研究開発は，PCI Express と Ethernet を統合するハードウェアレイヤ仮想化技術を目的としている．将来は，ソフトウェアレイヤで実装されるミドルウェアや Virtual Machine (VM) などと協調動作させることで，システムの構築と運営を容易にすることが期待されているが，現在はハードウェアの一部のみが実装されている．

以下に，ExpEther において実装済みである 2 つの要素技術である Ether Forwarding Engine (EFE)[7] および PCI Express-to-Ethernet Bridge (PEB) の概略を述べる．

2.1 Ether Forwarding Engine

EFE は Ethernet における MAC レイヤで end-to-end の輻輳制御および再送制御を行い，パケットロスのないネットワーク構築をサポートする．EFE を組み込んだ通信プロトコルスタックを実装することで，ExpEther を介して Ethernet LAN につながるノード間で通信を行う場合の TCP プロトコル処理による CPU 負荷を取り除くことができる．EFE における処理は TCP プロトコル処理より単純であるため，NIC 上に TCP OffloadEngine (TOE) を実装するよりも高い通信性能が得られる．

2.2 PCI Express-to-Ethernet Bridge

PEB は，PCI Express のトランザクションレイヤパケットを Ethernet フレームでカプセル化し，PCI Express におけるトランザクションレイヤプロトコルと Ethernet の Media Access Control (MAC) レイヤのブリッジを実現する．PEB によって，通常はシステムの内部に限定されていた PCI Express バス接続が Ethernet による LAN へと仮想的に延長され，高い拡張性と柔軟性をもったシステムの構築が可能となる (図 1 左)．

3. Remote Direct Memory Access

3.1 RDMA 通信

PEB による PCI Express のトランザクションレイヤと Ethernet の MAC レイヤのブリッジ機能により，コンピュータは，ExpEther ネットワークを介して PCI Express 接続の I/O デバイスにアクセスすることが可能である．しかし，システム構成はコンピュータ-I/O デバイス間接続に限定されており，コンピュータ-コンピュータ間接続はサポートされない．

そこで，Remote Direct Memory Access (RDMA) 技術を ExpEther に適用することで，ExpEther を介したコンピュータ-コンピュータ間接続を実現する．RDMA により，図 1 右のように，ExpEther を介して接続されたリモートノード上のメモリへのアクセスが可能となる．

RDMA 通信を用いない通信プロトコルでは，通信オペレーションにおけるユーザプロセスとネットワークインタフェース間のデータコピーは，カーネル空間でバッファリングされる (図 2，破線)．この際のデータコピー処理は，Programmed I/O (PIO) で行われるため，CPU への負荷が大きい．

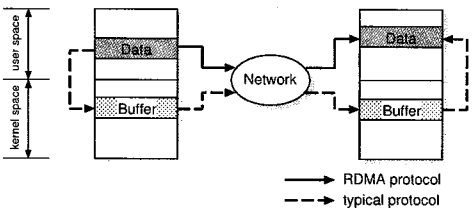

図 2 RDMA によるゼロコピーデータ通信

一方，RDMA 通信においては，通信データのバッファリング処理は生じず，ホストメモリとネットワークインタフェース間のデータコピー処理は DMA 転送

で行われる (図 2，実線).

また，今回実装した RDMA 通信機構では，ユーザアプリケーションが直接，RDMA 通信オペレーションを起動できる．したがって，通信オペレーション時のコンテキストスイッチが生じないため，通信レイテンシを短く抑えられる．

3.2 iWARP

RDMA 通信機構の設計には，iWARP の仕様を取り入れた．RDMA 通信機構の標準化を目的として定められた iWARP による RDMA 通信プロトコルスタックを図 3 左に示す．

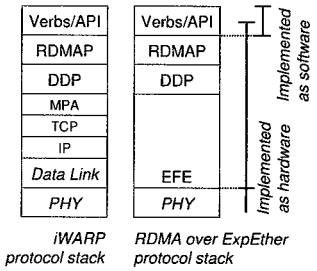

図 3 RDMA プロトコルスタックと実装レイヤ

iWARP RDMA 通信プロトコルスタックは，TCP/IP レイヤの上に定義される．我々が実装する RDMA 通信機構は ExpEther を対象とするため，iWARP が定義するプロトコルスタックのうち，Direct Data-placement Protocol (DDP) レイヤ，RDMA Protocol (RDMAP) レイヤ，Verbs レイヤを EFE の上に実装する (図 3 右).

iWARP のプロトコルスタック最上位の Verbs レイヤは，アプリケーションに対して抽象的なネットワークインタフェースへのアクセスを提供するレイヤであり，NIC 上に実装するハードウェアモジュールとファームウェア，ホストノード上で動作するデバイスドライバやライブラリなどから構成される．

4. ハードウェア構成

ハードウェアの実装は，ExpEther 開発ボードを用いて行った．開発ボードは，FPGA として Altera[8] 社の Stratix II GX EP2SGX 130G を搭載している．ホストインタフェースには PCI Express を，ネットワークインタフェースには Gigabit Ethernet を備える．ボード上には，512MB の Flash メモリが搭載されているが，今回の実装における I/O メモリには，主に FPGA 内部の Block RAM を利用している．

FPGA 上に実装した NIC の設計は，図 4 の様になる．PCI Express I/F モジュールを介してホストインタフェースに接続し，EFE がネットワークインタフェースに接続する．RDMA 通信プロトコルスタックの DDP レイヤ，RDMAP レイヤおよび Verbs レイヤの構成要素は，RDMA enabled NIC Core (RNIC Core) モジュール内に実装される．

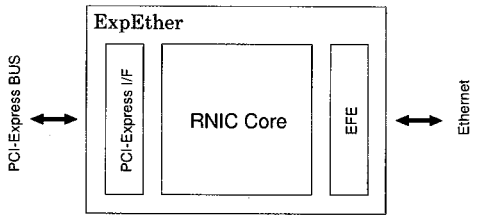

図 4 RDMA enabled ExpEther NIC のブロック図

図 5 に示す RNIC Core 上の Host Interface 中に配置された，RNIC REGs，QP REGs などのレジスタ群および SMT, CMT，PBLMT などの RAM は，ホストノード上から Verbs API を介してアクセスする．これらは，Verbs レイヤの構成要素であり，通信に利用する情報が格納される．以下に，各モジュールの役割を述べる．

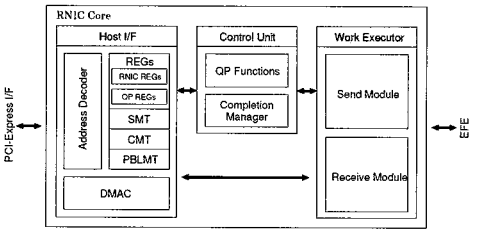

図 5 RNIC Core のブロック図

- **RNIC Registers (RNIC REGs)**
  RNIC Core の起動や動作設定に用いられるレジスタ群であり，主にデバイスドライバ (カーネル) を介してアクセスされる．
- **PBL Management Table (PBLMT)**
  ユーザ空間で確保された RDMA 通信用バッファに対応する Physical Buffer List (PBL) が格納される．
- **STag Management Table (SMT)**
  通信データは，ヘッダに含まれる Steering Tag (STag) によって，メモリ領域が識別される．SMT では，PBL の識別情報など，STag に関連付けられた情報を格納する．
- **Queue Pair Registers (QP REGs)**
  RDMA 通信オペレーションを含む Work Request (WR) の発行先となる Send Queue (SQ) と Receive Queue (RQ) の設定レジスタおよび，各キューへの窓口となる要求発行レジスタで構成

される．設定レジスタへのアクセスはカーネルを介す必要があるが，要求発行レジスタは QP が関連付けられたプロセスのメモリ空間にマッピングされるため，ユーザプロセスでも直接アクセスできる．

- **Completion Management Table (CMT)** Completion Queue (CQ) に関連する情報を格納する．CQ は，我々の実装ではホストメモリ上に生成するため，その領域の物理アドレスも格納される．ユーザプロセスは CMT には直接アクセスできないが，CQ に格納される完了情報のポーリングはカーネルを経由することなく行える．

5. ソフトウェア環境

ユーザアプリケーションが NIC を利用して RDMA 通信を行うためには，Verbs API (図 3) を介して，QP などのリソースを割り当てられる必要がある．Host Interface (図 5) 上のリソース管理は，主にデバイスドライバが行う．デバイスドライバは，RDMA 通信に必要な情報を Host Interface 上に登録した上で，プロセスにリソースを割り当てる．

Verbs API 関数の一部を表 1 に示す．アプリケーションは，Verbs API 関数を利用することで，プロセスやメモリを保護した通信を行うことができる．

表 1 主な Verbs API 関数

`open_rnic`	NIC のオープン
`allocate_pd`	Protection Domain の割り当て
`create_cq`	Completion Queue の生成
`create_qp`	Queue Pair の生成
`modify_qp`	Queue Pair の状態遷移など
`allocate_mr`	Memory Region の割り当て
`get_pbl`	PBL の取得
`post_sq`	Send Queue への命令の発行
`post_rq`	Receive Queue への命令の発行
`poll_cq`	Completion Queue のポーリング

RDMA 命令は `post_sq`, `post_rq` 関数によって，QP の SQ，RQ へ Work Request (WR) として発行する．これらの Verb の処理にはカーネルは関与せず，通信オペレーションの発行に際してコンテキストスイッチは発生しない．

5.1 アドレス変換

RDMA 通信機構では，RNIC Core 内の DMA Controller (DMAC) がマスターとなるホストメモリとの DMA 転送を行う．したがって，ホスト上のソフトウェアが，ターゲットとなるメモリ領域がスワップアウトされないようにピンダウン処理を行い，さらに NIC 上にメモリ領域の Physical Buffer List (PBL) と呼ばれる物理アドレスのリストを登録する必要がある (図 6)．これらの処理は一般的な通信機構では不要なため，通信プロトコルにおけるオーバーヘッドと見なされる．

デバイスドライバにおけるアドレス変換機構では，

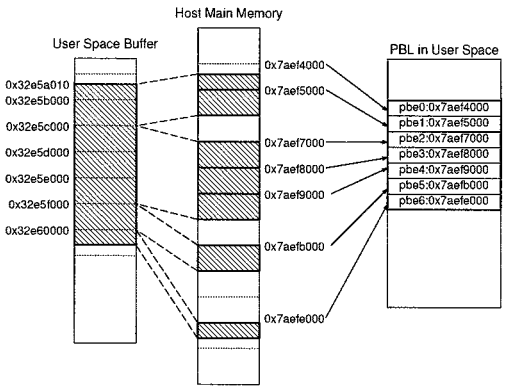

図 6 Phyisical Buffer List の構造

図 7 に示すように仮想アドレスと cr3 レジスタをキーとして変換テーブルを辿り，ページ番号を取得する．

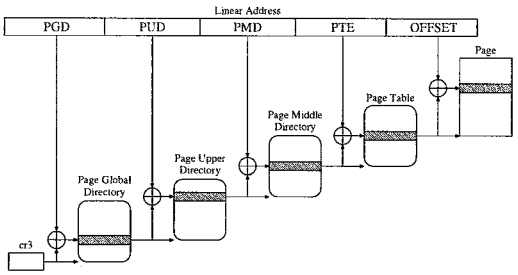

図 7 アドレス変換

5.2 I/O メモリアクセス

ユーザアプリケーションが直接 RDMA オペレーションを起動するためには，カーネルを介さない NIC へのアクセスが必要となる．アプリケーションは，Verbs API 関数を利用し，Host Interface (図 5) 上にある QP に，直接命令を書き込むことで RDMA オペレーションを起動できる．ユーザレベルでの NIC へのアクセスは，QP モジュールへのインタフェースとなる NIC 上のメモリ領域をアプリケーションのメモリ空間へマッピングすることで可能となる (図 8)．

RDMA オペレーションに関わらず，Verbs を利用した NIC へのアクセスの多くは，QP への命令発行と同様に，NIC 上のメモリリソースのアプリケーションのメモリ空間へのマッピングによって実現される．

6. 評　　価

ExpEther を介した RDMA 通信において，ソフトウェア環境が通信性能に与える影響を評価した．まず，RDMA 通信機構における通信オーバーヘッドの一つである，ホストメモリの Physical Buffer List (PBL)

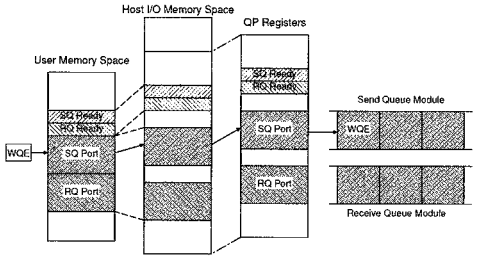

図 8 Queue Pair I/F のマッピング

の取得に要する時間を測定した．次に，ユーザアプリケーションによる I/O メモリへのアクセスレイテンシを測定した．

6.1 評価環境

評価に用いたシステムの主な仕様を表 2 に示す．

表 2 評価環境

CPU	Intel Core 2 Duo E6600 2.40GHz
Chipset	Intel G965 Express Chipset
Memory	2048MByte
OS	CentOS 5 .1 (kernel 2.6.18-53.1.4.el5)
NIC Controller	Intel 82566DC GbE Controller

ExpEther NIC がまだ開発途中であるため，I/O メモリアクセスレイテンシの測定には，ExpEther と同じ PCI Express 接続の NIC を使用した．

6.2 PBL の取得時間

RDMA 通信機構では，NIC がマスターとなってホストメモリに対する DMA 転送を行うことで，通信オペレーション単位のレイテンシを削減できる．しかし，そのためには前もって，ホスト上の通信バッファ領域をピンダウンした上で，PBL を NIC に登録する必要がある．これらの処理に要する時間が大きいと，アプリケーション開発者にとって RDMA 通信を選択しづらくなる．

ここでは，ピンダウン処理は mlock システムコールによって可能であるため，仮想アドレスと物理アドレスの変換による PBL の取得に要する時間を測定し，mlcok システムコールに要する時間と比較する．

6.2.1 測定方法

PBL の取得時間の測定は，以下の手順で行った．

(1) ユーザ空間でバッファ領域を確保し (malloc)，スワップアウトされないようにピンダウン (mlock) する．
(2) 時刻取得 (1 回目) を行う (gettimeofday)．
(3) バッファ領域の先頭アドレスおよび長さを，同じくユーザ空間で確保 (malloc) した PBL を保持するためのバッファの先頭アドレスと共にデバイスドライバに入力する (ioctl)．
(4) デバイスドライバは，バッファに対してページサイズ毎にアドレス変換処理 (図 7) を行い，取得した物理アドレスのリストをユーザ空間に確保された PBL を保持するためのバッファにコピーする．
(5) デバイスドライバの呼び出し (ioctl) からユーザプロセスに処理が戻る．
(6) 時刻取得 (2 回目) を行い (gettimeofday)，1 回目の時刻との差を PBL 取得に要する時間とする．

PBL を取得するターゲットバッファ領域のサイズは，4096Byte (1 ページ) から 1GByte まで変化させた．なお，今回の実装では，PBL 取得命令の発行時 (ioctl の呼び出し時) および取得完了時 (ioctl からのリターン時) の 2 回のみコンテキストスイッチが発生する．

6.2.2 測定結果

PBL の取得時間の測定結果を図 9 に示す．グラフは，横軸にバッファサイズ (Byte) を，縦軸に PBL 取得時間 (msec) をとったものである．グラフから，PBL の取得に要する時間は，バッファサイズに比例して大きくなっていることがわかる．正確には，バッファ内のページ数に比例して大きくなる．

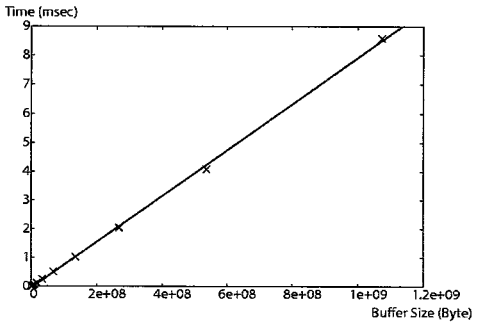

図 9 PBL の取得時間

1MByte のバッファ領域に対する PBL の取得時間は平均 8.35μsec，1GByte のバッファ領域に対しては平均 8.58msec であった．同じサイズのバッファ領域を mlock システムコールによってピンダウンするのに要する時間がそれぞれ約 15.1μsec および 624msec であることから，PBL の取得によるオーバヘッドは大きくなく，十分実用的な値であることがわかった．

6.3 NIC メモリへのアクセスレイテンシ

6.3.1 測定方法

NIC 上のメモリへのユーザレベルのアクセスレイテンシの測定を，以下の手順で行った．

(1) mmap システムコールによってデバイスドライバを呼び出し，I/O メモリ空間をユーザメモリ空間にマッピングする．

(2) 時刻取得 (1 回目) を行う (gettimeofday).
(3) ユーザプロセス内で，マッピングした領域から 1Byte(8bit) 単位の読み出しを行う．これを 1,000,000 回繰り返す．
(4) 時刻取得 (2 回目) を行い (gettimeofday)，1 回目の時刻との差を 1,000,000 で割った値をユーザレベルでのメモリアクセスレイテンシとする．

一方，カーネルを経由した場合のアクセスレイテンシの測定は以下のように行った．

(1) 時刻取得 (1 回目) を行う (gettimeofday).
(2) ioctl システムコールによってデバイスドライバを呼び出し，デバイスドライバに対して 1Byte(8bit) のデータ読み出し要求を送る．
(3) デバイスドライバ内では，カーネル空間にマッピングした I/O メモリ領域 (ユーザレベルでの測定においてアクセスしたアドレスと同じ) から読み出し，取得したデータをユーザプロセスに返す．
(4) 2~3 を 1,000,000 回繰り返す．
(5) 時刻取得 (2 回目) を行い (gettimeofday)，1 回目の時刻との差を 1,000,000 で割った値をカーネルを経由した場合のメモリアクセスレイテンシとする．

6.3.2 測定結果

NIC 上のメモリへのアクセスレイテンシの測定結果を表 3 に示す．結果より，ユーザレベルのメモリアクセスレイテンシはカーネルを経由した場合より 1.27μsec - 0.58μsec = 0.69μsec 小さいことがわかる．

表 3 NIC メモリへのアクセスレイテンシ

ユーザレベル	カーネル経由
0.58μsec	1.27μsec

カーネルを経由した場合のオーバヘッドの主な原因は，カーネル空間でのバッファリング処理およびコンテキストスイッチである．よって，ユーザレベルメモリアクセスによる通信レイテンシの低減は，通信命令を多く発行する状況，すなわち I/O メモリアクセスが多く生じる状況において有効であると考えられる．

7. おわりに

PCI Express と Ethernet を統合するネットワークインタフェース，ExpEther を対象とした RDMA 通信機構について述べ，実装する NIC の上に構築するソフトウェア環境が通信システムの性能に与える影響についての評価を示した．

まず，RDMA 通信で利用するバッファの PBL を取得するのに要する時間および同じバッファに対するピンダウン処理に要する時間を測定し比較した．次に，NIC へメモリアクセスレイテンシをユーザレベルで行った場合のレイテンシ削減率を測定した．

評価の結果，Physical Buffer List (PBL) の取得に要する時間は，1MByte の領域に対して 8.35μsec，1GByte の領域に対して 8.58msec であり，同じ領域に対するピンダウン処理に対する比率はそれぞれ 55.3%，1.4%であることから，実用的な値であることが確認された．また，PCI Express NIC に対するメモリアクセスをユーザレベルで行った場合のメモリアクセスレイテンシは 0.58μsec であり，カーネルを経由した場合と比べて 54.3%削減できることを確認した．

今後は，ExpEther 開発ボードを用いた RDMA 通信機構の実装を進め，通信ノード上のソフトウェア環境を用いた動作検証および通信性能の評価を行う予定である．

8. 謝　辞

本研究の一部は，総務省の委託研究「次世代バックボーンに関する開発研究」プロジェクトの成果である．

参考文献

1) 日本電気株式会社. http://www.nec.co.jp.
2) . J Suzuki, Y Hidaka, J Higuchi, T Yoshikawa, and A Iwata. ExpressEther - Ethernet-Based Virtualization Technology for Reconfigurable Hardware Platform. *High Performance Interconnects*, pp. 45–51, Aug. 2006.
3) :. Creating a Third Generation I/O Interconnect. Intel Developer Network for PCI Express Architecture.
4) :. IEEE P802.3an (10GBASE-T) Task Force. http://www.ieee802.org/3/an/.
5) RDMA Consortium:. http://www.rdmaconsortium.org.
6) 内山幸憲, 今田啓介, 辻聡, 大塚智宏, 鈴木順, 樋口淳一, 飛鷹洋一, 天野英晴. ExpEther における RDMA 通信機構の実装. 「ハイパフォーマンスコンピューティングとアーキテクチャの評価」に関する北海道ワークショップ (HOKKE-2008), pp. 175–180, Mar. 2008.
7) Hideyuki Shimonishi, Junichi Higuchi, Takashi Yoshikawa, and Atsushi Iwata. A Congestion Control Algorithm for Very Short Distance Communications. *HOTI'07 to be appeared*, p. 8, 2007.
8) Altera Corporation: . http://www.altera.com.