高効率なネットワークプロトコル処理方式の検討と開発

仲川和志† 横井伸浩† 高谷幸宏† 飯泉謙††

近年，プロセッサの進化は，1チップに搭載するプロセッサ数を増やすマルチコア化へと向かっており，今後もコア数の増加が見込まれている．このようなマルチコアを活用してサーバ等のシステム性能向上を達成するには，ボトルネックとなり得るネットワーク処理の高速化が必要である．本稿では，コア数の増加に見合ったネットワーク性能向上を目的として，高効率なネットワークプロトコル処理方式を提案する．TCP/IP のようなコネクション型のプロトコルでは，同一コネクションのパケットを単純に並列処理できないため，従来技術の多くは1パケットの処理時間を短縮する方向で高速化を実現している．提案方式では，連続受信パケット処理における並列化可能な処理と並列化できない順序処理をそれぞれ別モジュールで処理して効率的に処理する．さらに，本報告では，提案方式を FPGA (Field-Programmable Gate Array)で構成したネットワークオフロードエンジンに実装し，実機評価を行った．実機評価の結果，一般的な通信状況においても，理論性能 9.3Gbps 超に対して単一コネクションで 8.6Gbps の並列処理を実現可能なことを確認した．

Study and Development on Efficient Network Protocol Parallel Processing Method

KAZUSHI NAKAGAWA† NOBUHIRO YOKOI†
YUKIHIRO TAKATANI† KEN IIZUMI††

1. はじめに

従来，プロセッサの性能向上は，動作周波数を高めることで実現されてきた. しかし,近年では動作周波数向上は，チップの発熱問題のために限界に達しており，代わってプロセッサの進化は，1チップに搭載するプロセッサ数を増やすマルチコア化へと向かっている．例えば，Intel®社は 60 の演算コアを搭載するマルチコアプロセッサ Xeon Phi®[1)]の製品化を発表し，また Tilera®社は 72 コアを搭載するプロセッサを製品化予定 [2)]である．今後もプロセッサのコア数はますます増加することが見込まれている．

このようなマルチコアを活用してサーバ等のシステム性能向上を達成するには，ネットワーク入出力性能がキーとなる．つまり利用可能なコア数が増えて，その上で動作するアプリケーションの並列処理能力が高まったとしても，データ入出力帯域が十分ではない場合，システム性能向上を見込めなくなってしまう．

また，TCP/IP 処理に代表されるコネクション型プロトコルのネットワーク処理は，同一コネクションのパケットを単純に並列処理できないため，一般的に専用の単一コアを割り当てて実現されている．そのため，アプリケーション用のコア数が増加するのに伴って，ネットワーク処理専用コアの負荷は増加してボトルネックとなり得る．

このような背景のもとに，本稿では，コア数の増加に見合ったネットワーク性能向上を目的として，高効率なネットワークプロトコル処理方式を提案する．提案方式では，連続受信パケット処理における並列化可能な処理と並列化できない順序処理をそれぞれ別モジュールで処理して効率的に処理する．課題は，並列処理結果がパケット到着順とは異なる順序で出力される入れ違い発生であるが，提案方式ではヘッダ解析結果とペイロード処理結果を別々に扱うことでこの課題を解決する．

さらに，本報告では，提案方式を FPGA (Field-Programmable Gate Array)で構成したネットワークオフロードエンジンに実装し，実機評価を行った．本ネットワークオフロードエンジンは，サーバ通信を想定して 10G Ethernet のワイヤーレート通信に対応し，連続受信パケットを 5 並列で処理する．実機評価の結果，上記の課題が発生する一般的な通信状況においても，理論性能 9.3Gbps 超に対して，単一コネクションで 8.6Gbps の並列処理を実現可能なことを確認した．

2. ネットワークプロトコル処理の並列化検討

2.1 従来技術

10G Ethernet に代表されるようなネットワークの広帯域化・高速化に伴い，ネットワーク処理高速化に関する様々な提案や実用化がなされている．

専用ハードウェアによるプロトコル処理高速化は，古くから TOE(TCP/IP Offload Engine)として研究されている．例えば文献 3)は，データパケットに関する処理を専用ハード

† 株式会社 日立製作所
Hitachi, Ltd.
†† 株式会社 日立ハイテクノロジーズ
Hitachi High-Technologies Corporation

ウェアにより処理し，その他の制御用パケットは汎用的なCPU で処理するというハイブリッド構成の NPengine™を提案している．NPengine™は，1 データパケットに関わる一連のプロトコル処理を極力並列化することで 1 パケットの処理時間の短縮を実現している．

汎用的な CPU によるプロトコル処理高速化は, 実用面ではより進んでいる．文献 4)は，マルチコアプロセッサの 1 コアをネットワーク処理専用とする“TCP Onloading”を提案している．ネットワーク処理専用コアを設けることにより，このようなタスクをメインのアプリケーション処理から切り離すことが可能となる．また，文献 4)では，受信パケットを NIC(Network Interface Card）からホストメモリを経由せずに直接キャッシュメモリに書き込む Direct Cash Access や非同期メモリコピー等により，通信性能が向上するとしている．

しかしながら，このような 1 パケットの処理時間を短縮する方向の高速化だけでは，動作周波数の向上が見込めない中でさらなる高速化は困難である．

2.2 並列処理化の課題

TCP/IP5)に代表されるコネクション型プロトコルは，コネクション毎に状態情報（以後コネクション情報と記す）を持ち，このコネクション情報を参照・更新しながら処理を進めていく．そのため，連続受信するパケットを単純に複数コアに振り分けて並列処理するような構成をとる場合, コネクション情報にロックをかけるなどの排他制御を実行しなければならないため，実質的に単一コアによる処理となってしまうことが多い．

そこでまず，ヘッダ解析やチェックサム計算等の並列処理可能な部分とコネクション情報の参照・更新に関わる順序処理部分にプロトコル処理を分割し，前者を複数の並列処理コアで実施し，後者を単一の順序処理コアで実施するような構成が考えられる(図 1)．このような構成をとることで, 並列性を活かした処理の高速化を図ることができる．

しかしながら，受信パケットの長短に応じて並列処理コアの処理時間は一定とならないため，並列処理結果は，パケットの到着順とは異なる順序で順序処理コアへ出力され得る(図 2)．一般的にプロトコル処理は，整列されたパケット受信に最適化されており, このような順序入れ違いは, 連続するパケットの一部が欠損するパケットロスと判定される等により，パフォーマンス低下につながる．また，並列処理結果をパケット到着順に整列（ソート）してから，順序処理コアへ出力する方法も考えられるが，整列は入れ違い要素数をＮとした場合，Ｎの２乗に比例した計算量が必要であり，非常にコストが高い．

以上の従来技術と課題を踏まえて，本報告では，同一コネクションの連続受信パケットの並列処理化を検討する．

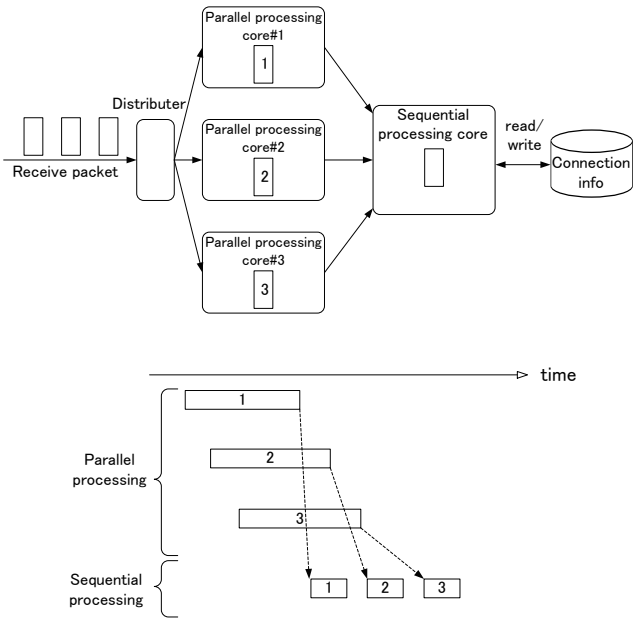

図 1　提案並列処理システム構成とパケット処理タイムチャート

Fig. 1 Structure of proposed parallel processing system and packet processing time chart

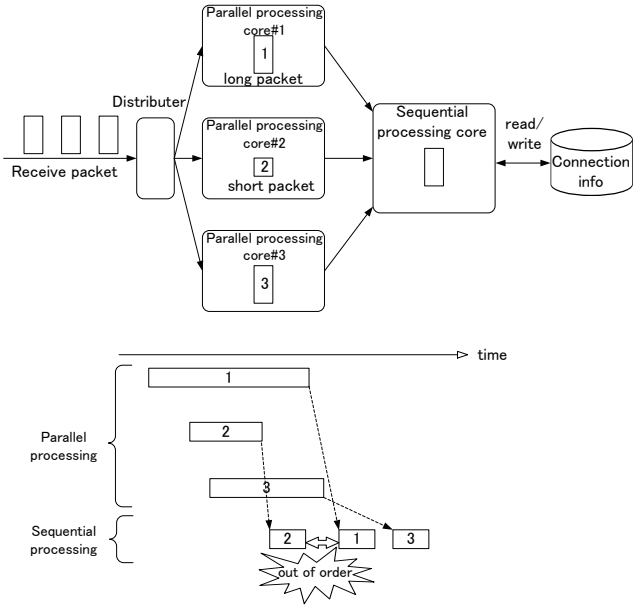

図 2　パケット順序入れ違い発生の例

Fig. 2 The situation of out-of-order packet

2.3 ネットワークプロトコル並列処理方式の提案

プロトコル処理における並列化可能な処理には，ヘッダ解析処理とペイロード処理がある．ヘッダ解析とは，当該パケットのプロトコル種別やペイロード長，シーケンス番号等の情報を後段の順序処理で利用しやすい形式に変換する処理である．また，ペイロード処理とは，代表的にはチェックサム計算であり，主にペイロード部分に対するデータ処理である．ペイロード処理の他の例としては，IPsec（Security Architecture for Internet Protocol）による暗復号処理やアプリケーション層での圧縮・伸長等の応用が考えられる．ヘッダ解析は，処理時間を一定とすることができる処理であり，一方でペイロード処理は，ペイロード長に応じて処理時間が変動する処理である．

提案手法による並列処理タイムチャートを図 3 に示す．並列処理コア＃１，＃２，＃３の順にそれぞれ長パケット，短パケット，中パケットが入力されたものとする．本提案手法では，並列処理コアはヘッダ解析結果とペイロード処理結果を別々に出力する．ヘッダ解析結果は一定時間後に出力されるため，パケット到着順に出力を確定させることができ，順序入れ違いは発生しない．順序処理コアは，ヘッダ解析結果をもとにしてコネクション情報の一時更新を実施し，またその際にヘッダ処理回数カウンタをインクリメントする．また，続くペイロード処理結果は，正常かエラーかを示す情報であり，正常を示していれば，上記ヘッダ処理回数カウンタをデクリメントする．仮にエラーの場合は上記一時コネクション情報を破棄すればよい．すべてのヘッダ解析結果とペイロード処理結果を順序処理コアが処理した場合に，ヘッダ処理回数カウンタはゼロとなり，コネクション情報を確定させる．

本提案手法では，あるパケットに関するヘッダ解析結果とペイロード処理結果を紐づけて管理せずに，数が一致することをもって，コネクション情報を確定させる．そのため，整列のようなコストの高い処理は不要となり，順序入れ違いの課題を解決できる．

ただし，ヘッダ解析結果のみで更新するコネクション情報は，一時的なものであり，パケットの連続受信によっていつまでも確定しない場合がある．この対策としては，ある一定時間確定しない場合や一定数のパケット処理後に確定しない場合には，一旦本方式を停止させて，ヘッダ解析とペイロード処理の完了まで待ち合わせた後にパケット到着順に順序処理を実施すればよい．

なお，振り分け部における振り分け方法としては，アイドル状態の並列処理コアにパケットを振り分けるアウトオブオーダー法を採用する．これにより，パケット到着順に並列処理を開始することが可能となり，並列処理コアで待ち合わせが生じることによる順序入れ違いを防ぐ．

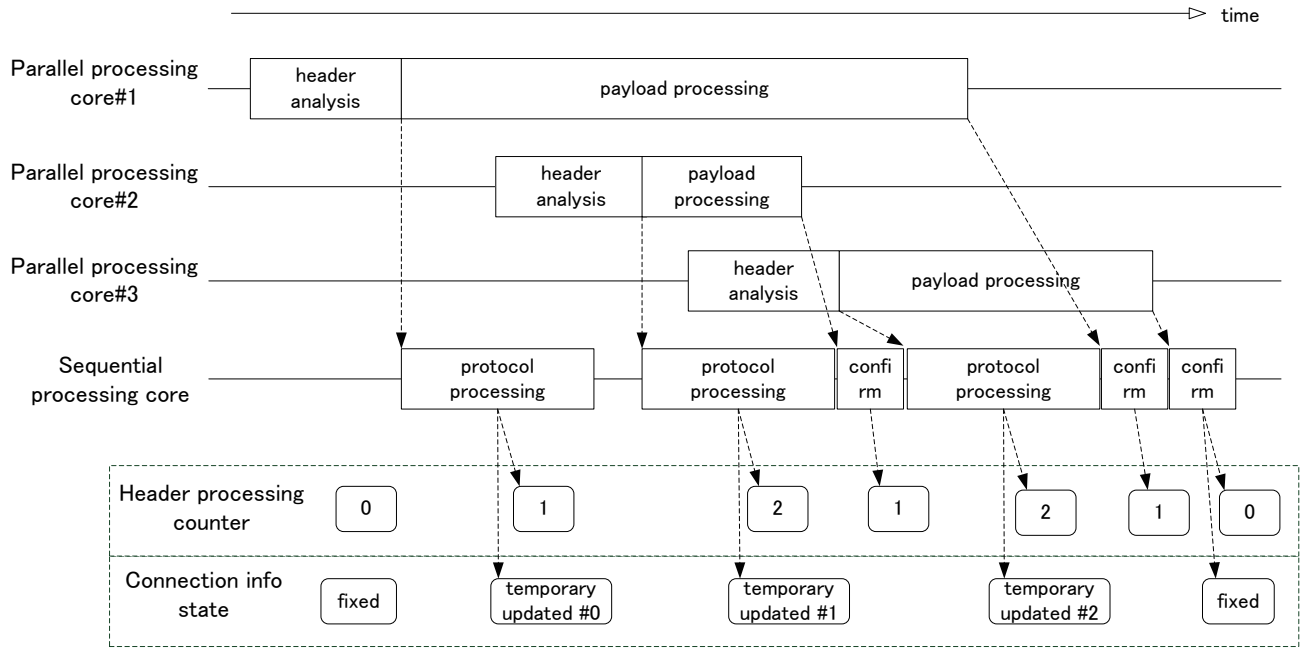

図 3　提案並列処理方式のタイムチャート

Fig. 3 A time chart of proposed parallel processing method

3. 実装及び結果

3.1 概要

提案方式の評価を目的として，FPGA で構成された 1G Ethernet 対応ネットワークオフロードエンジン 6)をベースにして, 10G Ethernet 対応オフロードエンジンを開発した．目標性能は，1 秒当たりの処理パケット数 0.81M[pps (packet per second)]である. これは Ethernet の一般的な MTU (Maximum Transmission Unit)値 1.5k バイトを前提として 10Gbps ワイヤーレート通信を実現可能な性能であり（10G[bps]/((1.5k+38)[byte]*8) ≈ 0.81M[pps], ただしイーサヘッダ，CRC，プリアンブル，フレーム間ギャップで合計 38 バイト分とした), 1 パケット当たりの処理時間に換算すると 1/0.81Mpps ≈ 1.2us となる．また，1G Ethernet 対応ネットワークオフロードエンジン 6)における 1 パケット(1.5k バイト)分の順序処理時間と並列処理時間を算出すると, それぞれ 3us と 12us であった. そのため本目標性能を達成するために，まずクロック周波数の 2 倍化とプロトコル処理チューニング(順序処理サイクル数 20%削減)を実施し，順序処理時間を 3us/2*0.8 = 1.2us まで削減した．同様に，1 パケット当たりの並列処理時間を，クロック周波数の 2 倍化と 5 並列実装とすることにより，12us/(2*5) = 1.2us に削減し，目標性能達成の見込みを得た.

FPGA 実装時の 10G Ethernet 対応ネットワークオフロードエンジンの全体構成を図 4 に示し, 実装仕様を表 1 に示す. 10G Ethernet 対応ネットワークオフロードエンジンは, パケットの解析と生成を専用に行うパケット解析部と生成部, コネクション情報の参照･更新を実施する通信制御部, 受信パケットを振り分けるネットワーク I/F バッファ，パケットデータの一時保存を行う内部 I/F バッファから構成される．通信制御部はコネクション情報を格納する RAM や CAM (Content Addressable Memory)，制御 PC からのレジスタ読み書きを受け付けるシステムコンソール等にも接続されている．パケット解析部とパケット生成部を 5 並列実装することで並列処理を実現している．また，前節で述べた提案方式によって，パケット解析部は，ヘッダ解析結果とペイロード処理結果を別々に出力し，その結果を受け取る通信制御部は，一時コネクション情報の更新とヘッダ処理回数カウンタの管理により順序入れ違いに対応する.

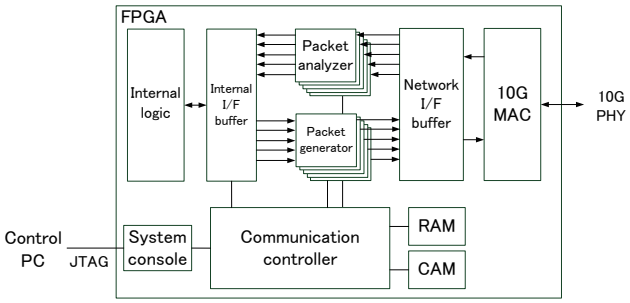

図 4　ネットワークオフロードエンジンの全体構成

Fig. 4 Block diagram of network offload engine

表 1　ネットワークオフロードエンジン仕様

Table 1 Network offload engine specification

#	Item	Value
1	Bandwidth	10Gbps
2	Core clock frequency	62.5MHz
3	Maximum number of connections	16 connections
4	Supported protocol	ARP, TCP/IP
5	Internal bus width	32bit
6	Parallel number	5
7	MTU	16k byte
8	Packet per second	0.81M pps

3.2 開発結果

市販の FPGA 評価ボード(Altera®社製 Stratix®IV GX FPGA 開発キット 7))上の FPGA(Altera®社 Stratix®IV GX 530)をターゲットに論理合成(Altera®社 Quartus®II 使用)を行った．結果を表 2 に示す.

表 2　論理合成結果

Table 2 Result of logic synthesis

#		Number of combinational ALUTs	Number of logic registers	Block memory bits
1	Communication controller	21.3k	16.5k	1352.7k
2	Packet analyzer (one module)	6.4k	0.4k	0
3	Packet generator (one module)	1.7k	0.2k	0
4	Network I/F buffer	1.1k	1.3k	1597.4k
5	Internal I/F buffer	29.3k	3.7k	2097.1k
6	FPGA System Total (Resource usage percentage)	101.6k (24%)	36.0k (8%)	8843.8k (42%)

3.3 実機評価結果

10G Ethernet 対応ネットワークオフロードエンジンを FPGA 評価ボードに実装し，汎用サーバとの通信評価実験を実施した．評価システム構成を図 5 に示し，外観写真を図 6 と図 7 に示す．対向する汎用サーバには，Linux® OS (kernel 2.6.18-274)を採用し，ネットワークカードに市販の Intel®社製 10G Ethernet 対応 NIC (Network Interface Card), CPU に Intel® Core®i7-3930K，メモリ 64GB を備えた高性能サーバを用意した．制御 PC は，FPGA 評価ボードと USB/JTAG で接続されており，ネットワークオフロードエンジンのレジスタ初期設定等の制御を行う．汎用サーバと FPGA 評価ボード間は，10G スイッチ(アラクサラネットワークス社製 AX3630S-24S2XW)を介した接続，あるいは，スイッチを介さない直接接続とし，両者で結果に違いはないことを確認している．汎用サーバは，その上で動作する通信テスト用の socket プログラムにより，コネクション確立要求を送信し，コネクション確立後は，1000 万パケット連続送信し，その際のアプリケーション層での送信データスループットを計測する．なお，本評価は，提案方式によるプロトコル処理の実機検証を目的としているため，FPGA 評価ボードで受信したパケットデータは，プロトコル処理後にホスト PC 等の外部へ受け渡されることなく，FPGA 内部論理にてデータがそろっていることのみチェックして破棄する仕様とした．

また，評価では，（1）ペイロード長を均一にした場合と，（2）一般的な通信環境を想定したペイロード長が均一にならない場合の 2 通りで評価を実施した．後者（2）の場合は，NIC の MTU 値によりペイロード長を最大 8192 バイトに制限し，socket プログラム上の送信データサイズを 1024 バイトとして再現させた．これは，一般的な OS のプロトコルスタックには，アプリケーション層から渡された送信データを極力まとめて送信する機能(nagle アルゴリズム)8)があるため，上限 8192 バイトまでの長短パケットが汎用サーバ側から送信されるためである．

以上の条件のもとに，まず提案方式未実装のオフロードエンジンの基本性能を（1）ペイロード長を均一にした場合(1460 バイト，2048 バイト，4096 バイト，8192 バイトの 4 通り)でスループットを計測した．その結果を図 8 に示す．すべてのペイロード長において，スループットは，ほぼ理論性能を達成している．これにより汎用サーバとの通信互換性に問題がないこと，および，10Gbps ワイヤーレート通信の目標性能を達成したことを確認できた．

次に，（2）ペイロード長が均一にならない場合で計測した．その結果を表 3 に示す．提案方式未実装のオフロードエンジンでは，順序入れ違い発生によるパケットロスによって極端にスループットが落ちてしまい，スループットは約 175Mbps となった．一方で提案方式実装済みオフロードエンジンではパケットロスなく，約 8.6Gbps となった．

また，本評価における最短パケット長 1024 バイトの場合で送信データスループットの理論性能は 9.3Gbps であるため，8.6Gbps では理論性能には到達していない．この主な原因は，オフロードエンジン側がパケット連続受信によってコネクション情報を確定できずに ACK 応答を長時間送出できなくなり，汎用サーバ側が送信レートを落としているためであることがわかった．2.3 節で述べたコネクション情報確定のための対策は，全体の性能に与えるオーバーヘッドも含めて，今後の検討課題である．

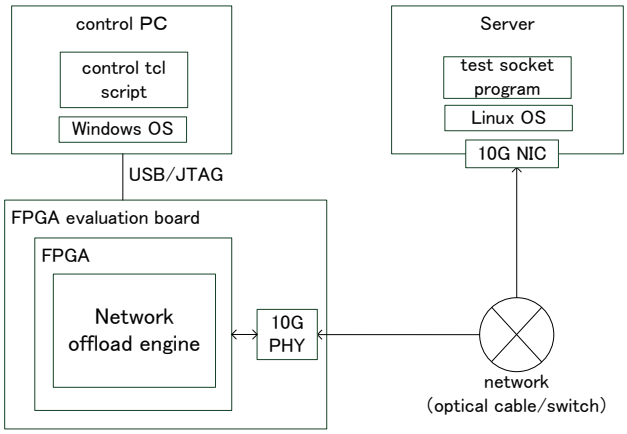

図 5　実機評価環境のシステム構成

Fig. 5 Block diagram of evaluation system

図 6 FPGA 評価ボード

Fig. 6 FPGA evaluation board

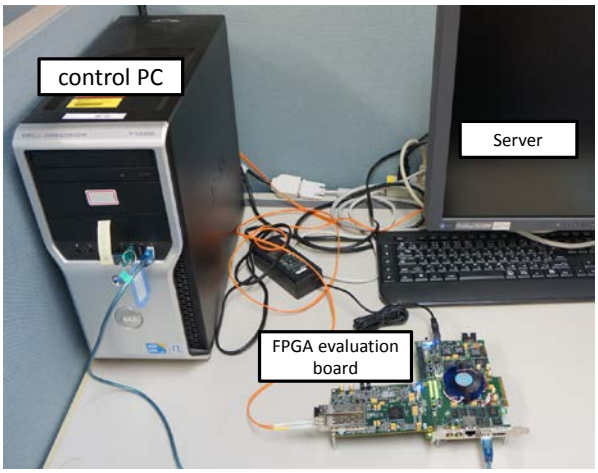

図 7　評価環境の外観

Fig. 7 A view of evaluation environment

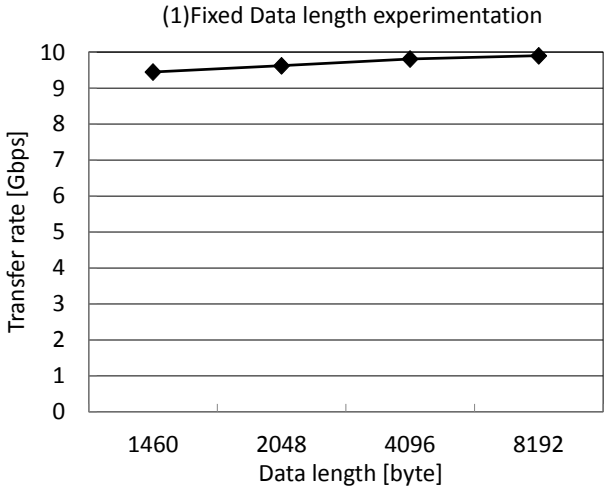

図 8　ペイロード固定長の場合のスループット性能

Fig. 8 Payload data length vs. throughput

表 3　ペイロード非固定長(1024～8192 バイト)の場合のスループット性能

Table 3 Throughput ((2) Variable data length experimentation)

Method	Throughput
Normal parallel method	175Mbps
Proposed method	**8.6Gbps**

4. おわりに

本報告は，連続受信パケットのプロトコル処理の並列性に着目し，高効率なネットワークプロトコル処理方式の提案を行った．単純な並列処理システム構成では，ペイロード長に応じて並列処理時間が変動し，パケット到着順とは異なる順序に入れ違ってしまう課題が発生する．提案方式では，ヘッダ解析結果とペイロード処理結果を紐づけて管理しないことにより，整列等のコストの高い処理を不要としてパケット並列処理を実現している．さらに，実機評価にて，FPGA で構成したネットワークオフロードエンジンにより，順序入れ違いによるパケットロスなく，理論性能 9.3Gbps 超に対して単一コネクションで 8.6Gbps を達成可能なことを確認した．

参考文献

1) Intel®社 Xeon Phi®
<http://www.intel.co.jp/content/www/jp/ja/processors/xeon/xeon-phi-detail.html>
(2013.04.26).
2) TILERA®社 TILE-Gx® processor
<http://www.tilera.com/products/processors>
(2013.04.26).
3) 田中信吾　他，“Gigabit/10Gigabit Ethernet に対応した高効率 TCP/IP オフロードエンジン，”情報処理学会論文誌 vol.52 No.12 3715-3728 (Dec. 2011).
4) G. Regnier et.al.，“TCP onloading for data center servers,” IEEE Computer, November 2004.
5) Transmission control protocol, Internet Engineering Task Force RFC793 (Sep. 1981).
6) 小日向宣昭　他，“組込み機器向けネットワークオフロードエンジンの検討と開発，”電子情報通信学会論文誌 vol.J94-B No.12 1547-1555 (2011).
7) Altera®社 Stratix®IV GX FPGA 開発キット
<http://www.altera.co.jp/products/devkits/altera/kit-siv-gx.html>
(2013.04.26).
8) アンドリュー・S・タネンバウム，“コンピュータネットワーク第4版”，日経BP社.