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

2005－OS－100 (12)
2005／8／3

細粒度パケット間隔制御の実装と評価

菅原 豊　　稲葉 真理　　平木 敬

東京大学 大学院情報理工学系研究科 コンピュータ科学専攻
〒113-8656 東京都文京区本郷 7-3-1
Email: {sugawara, mary, hiraki}@is.s.u-tokyo.ac.jp

要旨

近年、クラスタ間で複数 TCP ストリームを用いた高速なデータ転送を行う事が求められている。高速・長距離ネットワークを用いた TCP データ転送では、パケット落ちを抑えるために送信間隔制御が必須である。しかし、既存のパケット間隔制御方式は複数ストリームに対する効果には限界がある。まず、スロースタートフェイズに同期的なパケットロスにより送信帯域が低下する。また、ストリーム間のパケット間隔が制御されないため、輻輳回避フェイズのパケット落ちが最小化されない。本研究では、FPGA を搭載した専用ゲートウェイを用いて複数ストリームのパケット送信順序・間隔を制御する方式を提案する。また、10 ギガビット・イーサネット実験装置を用いて提案システムの実装を行う。

Implementation and Evaluation of Fine-grain Packet Interval Control

Yutaka Sugawara　　Mary Inaba　　Kei Hiraki

Department of Computer Science, Graduate School of Information Science and Technology,
University of Tokyo
7-3-1 Hongo Bunkyo-ku Tokyo 113-8656, Japan
Email: {sugawara, mary, hiraki}@is.s.u-tokyo.ac.jp

Abstract

Recently, it is required to realize high-speed data transfer between cluster systems using multiple TCP streams. To reduce packet loss, transmission interval control is necessary for TCP data transfer on high-speed long distance networks. However, with existing packet interval control methods, the effect for multiple TCP streams is limited. First, the transmission bandwidth is reduced by synchronous packet loss in slow start phase. Secondly, the packet drop is not minimized in congestion avoidance phase because the interval between different streams is not controlled. In this research, we propose a method to control the transmission order and interval of multiple streams using a specialized gateway equipped with FPGAs. In addition, we implement the proposed system using a 10 Gigabit Ethernet experimental system.

1 はじめに

近年、クラスタ間でTCPを用いて高速にデータを転送する技術が求められている。例えば、Data-Reservoir [5] では長距離広帯域ネットワークで結ばれたクラスタ間でハードディスクのデータ共有を行うため、高速なデータ転送が必要である。

現在では、10Gbpsに達する長距離ネットワークがクラスタ間データ転送に使用可能である。例えば、日本ではSuperSINET、米国ではAbilene、日米間ではIEEAF、欧州および米欧間ではSURFnet等の10Gbpsネットワークが利用可能である。さらに、将来は40Gbpsまたはそれ以上の高速な広域ネットワークが利用可能になる見込みである。このような高帯域かつ遅延が大きいネットワークはLong Fat pipe Network (LFN) と呼ばれる。

このような長距離高速ネットワークを用いたデータ転送のために、協調動作する複数のTCPストリーム (並列TCPストリーム) を用いてデータを転送する技術が研究されている [6][12][14]。これは、与えられたデータを分割し、分割されたそれぞれのデータを別々のTCPストリームで送るというものである。データ転送中に分割の仕方を変更できる場合と、分割の仕方が予め決められている場合とがある。Data-Reservoirは後者である。これは、クラスタの各ノードが、自分が持つデータを自分が張ったストリームを用いて送るためである。

並列TCPストリームを用いる一つの理由は、クラスタシステムの場合、送信するデータが別々のマシンに分散している場合があるためである。例えば、Data-Reservoirでは送信すべきデータはクラスタの各ノードのディスクに分散して格納されている。そのため、クラスタ毎に別々のストリームで通信を行う。

並列TCPストリームを使用するもう一つの理由は、ストリーム1本の場合に比べて高い通信帯域を実現しやすいためである。TCPの通信帯域を上げるためには、輻輳ウィンドウ (Congestion Window, CWND) 値を上げる事が必要である。輻輳ウィンドウ値とは、TCPにおいて受信側からのACKを待たずに送る事ができるデータのサイズである。一定のTCP通信帯域を実現するために必要な輻輳ウィンドウ値はネットワークの往復遅延時間 (Round Trip Time, RTT) に比例する。また、広く使われているTCPの実装、TCP-RenoとTCP-NewRenoでは、輻輳ウィンドウ値が増加する速度はRTTに反比例する。これらの理由により、LFNではリンク帯域を使い切る値まで輻輳ウィンドウ値を増加させる事が難しい。そのため、単一ストリームではリンク帯域に近い通信速度を出すことが難しい。しかし、N本のストリームを用いる事により、同じ輻輳ウィンドウ値に対して単一ストリームの最大N倍の通信帯域を実現可能である。そのため、並列TCPストリーム全体としての通信帯域をリンク帯域に近づける事が容易である。

TCP通信で通信帯域を上げるためには、パケット落ちを抑える事が必要である。これは、TCPではパケットが落ちるたびに輻輳ウィンドウ値が減らされるためである。

LFNにおいてパケット落ちを抑えるには、パケット送出間隔の制御が不可欠である。現在広く使われているTCP-Reno/TCP-NewRenoは、そのまま使用した場合にパケットを間欠的・バースト的に送る性質がある。図1は、バースト的なパケット送信を示すデータ例である。この性質のため、長い時間での平均送信帯域が利用可能帯域を下回っている場合でも、一時的に送信帯域が利用可能帯域を上回る瞬間が生じる。そのため、ボトルネックリンクの直前のルータにおいてパケットのバッファリングが行われる。バースト的に転送されるデータの量が多い場合、バッファがあふれてパケットロスが生じる。このパケットロスを抑えるには、できるだけ一定のレートでパケットを送る必要がある。すなわち、連続してパケットを送る際に適切な間隔をあける事が必要である。我々は、パケット間隔を制御するためのClustered spacing [8] を提案し、また、これを用いたTRC-TCP[3] を実装した。

しかし、並列TCPストリームを用いる場合、既存の間隔制御方式には限界がある。既存方式では、並列TCPストリーム全体のパケット列の送信間隔・順序を制御することができない。そのため、以下の要因によりTCP送信帯域が低下する。

1. スロースタートフェイズにおいて同期的なパケットロスが生じる
2. 輻輳回避フェイズにおいてパケットロスの最小化ができない

1.は、送信間隔を制御された複数のストリームがある場合、ネットワークの利用可能帯域を超えた後、全てのストリームが一斉にパケットロスを起こす [1]

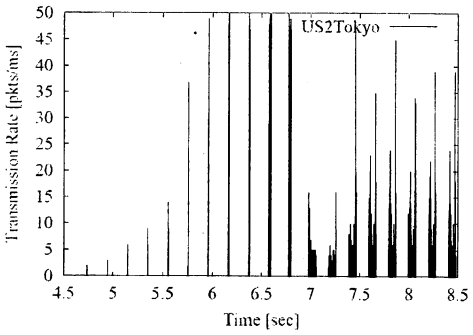

図 1: TCP-Reno のバースト的なパケット送信

というものである。この結果、全てのストリームの輻輳ウィンドウ値が同時に減らされるため、送信帯域が大きく減少する。

2. は、異なるストリーム同士のパケット送出間隔が制御されないため、間隔が不ぞろいになる事に起因する。輻輳回避フェイズでは、あるパケットが落ちた場合、次のパケットまでの間隔が短い場合は次のパケットが落とされる確率が高い [11]。その結果、パケットロス率が増加する。このパケットロスを抑えるには、パケット間隔をできるだけ等しくする事が必要である。

本研究では、上記の点を改善するため、並列 TCP ストリーム全体のパケット送信タイミングを集中的に制御する方式を提案する。パケット送信のタイミング制御は、LAN と WAN の境目に配置した FPGA 化ゲートウェイ [16] を用いて行う。このゲートウェイが個々のストリームを認識し、パケットのバッファリング、送出順序制御、間隔制御を行う。このゲートウェイは一種の Performance Enhancing Proxy (PEP) [4] である。1. については、送信間隔を保ったままで、各ストリームのパケットを可能な限り連続して送信する事で解決する。2. については、並列ストリーム全体のパケット送信間隔をスケジューラで制御する事により解決する。本研究では、上記のシステムの実装を行う。実装には、2 つのポートと FPGA を持つネットワーク実験装置を用いる。ハードウェアを制御に使用する理由は、高速なパケット処理と高精度な送信タイミング制御をソフトウェア方式よりも容易に実現できるためである。

本稿の構成を以下に述べる。まず 2 章で、並列 TCP ストリームを用いた場合に通信帯域が低下する要因を議論する。次に 3 章で、ゲートウェイを用いたパケットの間隔・順序制御方式を提案する。また、4 章で、提案方式の実装を行う。5 章では関連研究の紹介を行う。最後に 6 章で本稿のまとめを行う。

2 並列 TCP ストリーム使用時の性能低下要因

2.1 スロースタートフェイズの同期パケットロス

送信間隔が制御されていない TCP ストリームが複数する場合は、個々のストリームがバースト的にパケット送信を行う。図 2 にこの送信パターンを模式的に示した。始めは全ストリームがスロースタートフェイズである。上段が個々のストリームがパケットを送信するタイミング、下段が、ボトルネックリンク直前のルータのバッファ使用率である。この通信パターンでは、最初のパケットロスが起きるタイミングがストリームによって異なる。これは、各ストリームのパケットがまとめて送られているためである。パケットロスが起きている時に偶然パケットを送信していなかったストリームはパケットロスが起きない。このストリームが後にパケットを送った際に始めてパケットロスが起きる。例えば、図 2 では、パケットロスが生じ始めた瞬間にパケットを送っていたストリーム 3 は、すぐにパケットロスが起きている。しかし、ストリーム 1 はこの時点ではパケットを送っていなかったため、パケットロスが生じていない。もう少し後にパケットを送信し始めた時にはじめてパケットロスが起きている。また、ストリーム 2 にパケットロスが生じるのはストリーム 1 よりさらに後である。

結果として、各ストリームがスロースタートフェイズから輻輳回避フェイズへ移行するタイミングが異なる。したがって、他のストリームが輻輳回避フェイズに移行して一時的に送信帯域が減った時に、スロースタートフェイズにあるストリームは送信帯域を増やし続ける。結果として、リンク帯域の使用率が高く保たれる。

一方、送信間隔が制御されたストリームが複数存在する場合、スロースタートフェイズでパケットロスが同期して起きる。個々のストリームの送信間隔が制御されている場合、全てのストリームがどの瞬間にもパケット送信を行っている。そのため、ルー

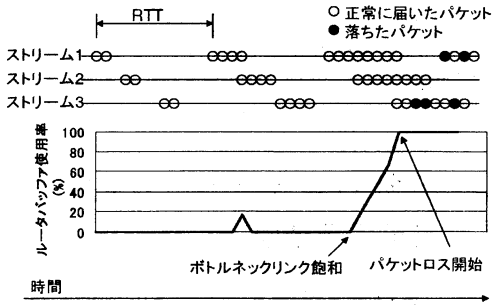

図 2: 間隔制御されない並列 TCP ストリームのパケットロス

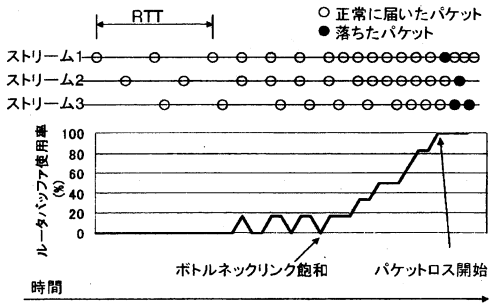

図 3: 間隔制御された並列 TCP ストリームのパケットロス

タのバッファが溢れると、多数のストリームのパケットが一斉に落ちる。図 3 は、この様子を模式的に示したものである。結果として、多数のストリームが一斉に輻輳回避フェイズに移行する。そのため、送信帯域が一時的に大きく低下する。

2.2 輻輳回避フェイズの同期パケットロス

送信間隔が制御された TCP ストリームが一つだけ存在する場合、2 つのパケットの間隔はほぼ一定に保たれている。そのため、パケットが落ちる確率が低く保たれている。

一方、送信間隔が制御されたストリームが複数存在する場合は、連続してパケットが落ちる可能性が高くなる。これは、ストリーム間のパケット送信間隔が制御されていないためである。パケットロスがあるストリームで生じた場合に、直後のパケットとの間隔が狭かった場合は、そのパケットが同時に落

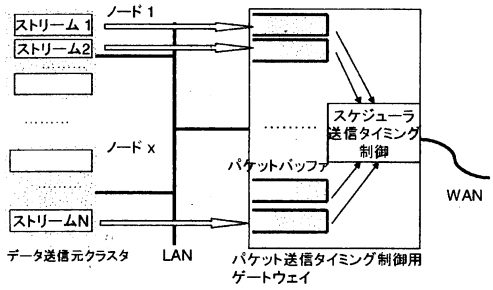

図 4: ゲートウェイを用いたパケット送信順序・間隔制御

ちる確率が高い。複数のストリームがパケットロスにより同時に輻輳ウィンドウ値を減らすと、一時的に送信帯域が大きく減少する。

3 ゲートウェイを用いたパケット送信順序・間隔制御

本研究では、特定の並列 TCP ストリームを識別可能なゲートウェイを用いる。図 4 に、クラスタからの送信パケットをゲートウェイで制御する様子を示す。このゲートウェイは認識した並列 TCP ストリームに対して、パケットをバッファリングし、送信タイミングを制御する事ができる。制御すべき並列 TCP ストリームの送信元 IP アドレスとポート番号は、あらかじめゲートウェイに入力しておく必要がある。また、ゲートウェイは並列 TCP ストリームの送信元ノードから TCP スタックの情報を受け取る事ができる。本研究の方式では、各ストリームの輻輳ウィンドウの値と、スロースタートフェイズか否かの情報を用いる。以上の機能の具体的な実装法については 4 章で述べる。

この章の以下の説明では、並列ストリームは N 本のストリームから成るものとする。また、i 番目のストリームの輻輳ウィンドウ値を $CWND_i$ と表記する。

3.1 送信順序制御

送信順序制御は、スロースタートフェイズの同期パケットロスを減らすために行う。スロースタートフェイズにおいて同期パケットロスが生じる原因は、

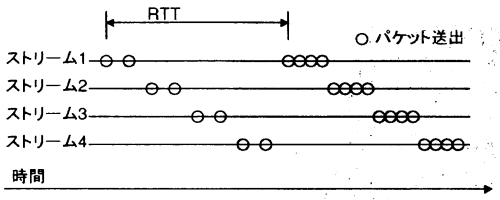

図 5: 同一ストリームパケットの連続送信

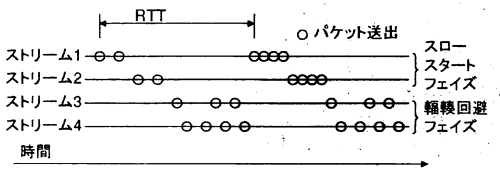

図 6: ストリームのモードが異なる場合の送信パターン例

複数の TCP ストリームのパケットが混ざって送信される事である。したがって、図 5 のように各ストリームのパケットができるだけ連続して送られるようにすれば改善可能である。

ただし、この方式を適用すると、ストリーム毎に送信帯域に格差が生じる。与えられたデータのストリームへの割り当てが予め決まっている場合、この格差により転送所要時間が長くなる。これは、転送所要時間は最も転送に時間がかかったストリームにより決まるためである。この場合、ストリーム間で輻輳ウィンドウの値を調停する [17][14] 等の対策が必要である。一方、データをストリームに動的に割り当てる事が可能な場合は、ストリーム間の速度格差は問題にならない。

また、1 つのストリームのパケットを連続して送ると、受信マシンにおいて NIC 受信バッファが溢れる可能性がある点に注意が必要である。バッファ溢れが問題になる場合は、パケット送信順序を変更して同一ストリームのパケットを一定個数以上連続して送らないようにする必要がある。

送信順序制御は、スロースタートフェイズのストリームに対してのみ適用する。輻輳回避フェイズのストリームに対するパケット順序制御は行わない。図 6 に、スロースタートフェイズと輻輳回避フェイズのストリームが混ざっている場合のパケット送信パターン例を示す。

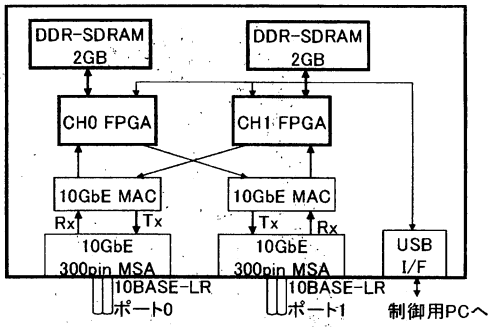

図 8: 10 ギガビット・イーサネット実験装置ブロック図

3.2 送信間隔制御

連続した複数のパケットが同期して落とされる事を防ぐため、パケット送信間隔の制御を行う。並列 TCP ストリーム全体として RTT 毎に送られるデータの量は $\sum_i CWND_i$ である。したがって、単位時間当たりに送るべきデータ容量は $\sum_i CWND_i/RTT$ である。よって、パケットのペイロード長を l として、時間 $l \cdot RTT/\sum_i CWND_i$ 毎に 1 パケットを送出すれば良い。この制御は Deficit Round Robin [10] 等のアルゴリズムを用いる事により実現可能である。この制御は、スロースタートフェイズ、輻輳回避フェイズ両方の TCP ストリームに対して行う。

4 実装

10 ギガビット・イーサネット実験装置を用いて、提案システムの実装を行った。図 7 に装置基板写真を示す。この装置は、10 ギガビット・イーサネットポート、MAC、FPGA、DDR-SDRAM 等をそれぞれ 2 つずつ搭載している。搭載している FPGA は Xilinx XC2VP50-5 である。

図 8 に装置のブロック図を示す。片方のポートから受信したデータは FPGA に入力され処理された後、もう片方のポートから送信される仕組みである。データが流れる方向は 2 方向あり、各方向毎に別々の FPGA が処理を行う。

パケット送出タイミング制御機構は、装置の FPGA と DDR-SDRAM を用いて実現する。図 9 に、FPGA 上に実装した回路のブロック図を示す。

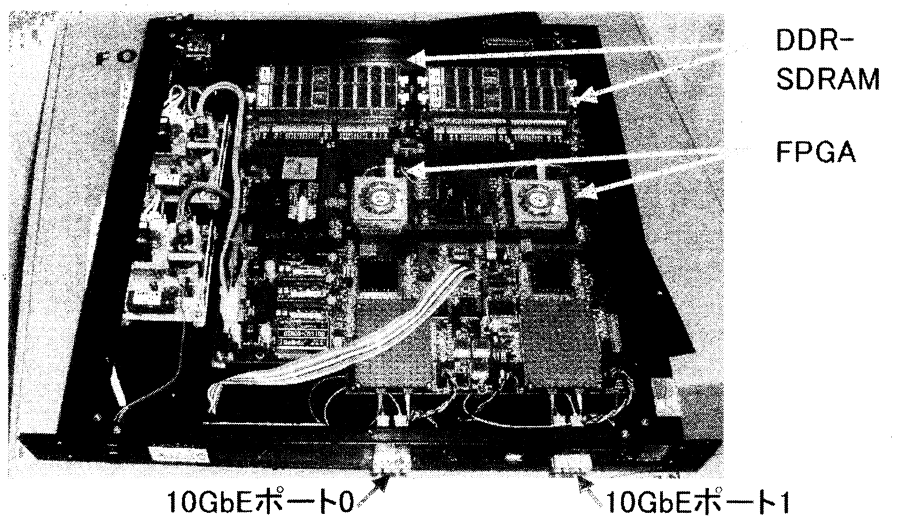

図 7: 10 ギガビット・イーサネット実験装置外観

実験装置の片方のポートを WAN 側、もう片方を LAN 側に接続する。LAN 側に並列 TCP ストリームを用いてデータを送信するクラスタを接続する。LAN 側から送られたデータは FPGA により受信され、TCP ストリームの特定が行われる。ストリームが予め指定された並列 TCP ストリームに属している場合は、送信順序、間隔の制御が行われる。まず、受信したパケットは DDR-SDRAM を用いたバッファに格納される。格納されたパケットは、スケジューラにより決められたタイミングで送信される。スケジューラが送信タイミングを決めるためは各ストリームの輻輳ウィンドウ値等の情報が必要である。これらの情報は TCP スタック状態テーブルに保持されている。クラスタの各ノードから TCP スタックの新しい状態が通知された場合は、このテーブルに新しい状態が記録される。

表 1 に FPGA の資源使用量を示す。合成には Xilinx ISE6.3i を用いた。

4.1 ストリーム認識

提案方式の実現のためには、制御対象となる TCP ストリームのパケットを認識する必要がある。本研究では、送信側の IP アドレスとポート番号で並列 TCP ストリームを判別する仕様とした。

実装を簡単にするため、並列 TCP ストリームの

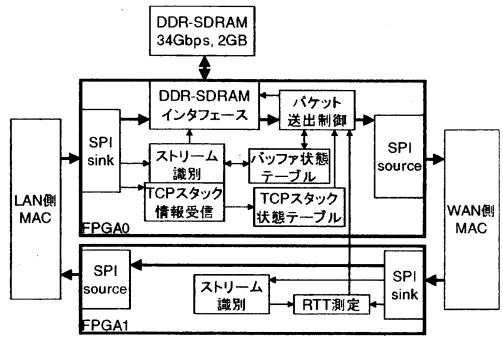

図 9: パケット間隔制御機構ブロック図

送信アドレスは一つの IP アドレスとマスクの組で指定する仕様とした。例えば、送信 IP アドレスが 192.168.1.32 でマスクが 255.255.255.240(0xfffffff0) の場合、192.168.1.32–192.168.0.47 の IP アドレスから張られたストリームが並列 TCP ストリームのメンバとして扱われる。

ポート番号は、開始ポート番号と 1 ホストあたりのストリーム本数の組で指定する。例えば、1 ホストあたりのストリーム本数が 10 で開始ポート番号が 10000 の場合、送信ポート 10000—10009 のストリームが制御対象となる。本研究の実装では、簡単のため開始ポート番号とホストあたりのストリーム本数は全てのノードで共通とした。現在の実装で

		使用数	使用率
FPGA0	スライス	12769/23616	54%
	ブロック RAM	38/232	16%
FPGA1	スライス	9908/23616	41%
	ブロック RAM	30/232	12%

表 1: FPGA 資源使用率

は、制御できるストリームは最大 512 本である。

以上の情報は、USB インタフェースを介して制御 PC から入力する。

4.2 パケットのバッファリング

パケットのバッファリングは、実験装置の DDR-SDRAM を用いて行う。DDR-SDRAM は容量が 2GB あるため、遅延帯域積が最大 2GB のネットワークまで対応可能である。DDR-SDRAM は PC2100 の 2 チャンネル構成で、アクセスバンド幅は 34.1Gbps である。したがって、10Gbps での読み出しと書き込みを並行して行う事が可能である。

4.3 送信ホストからの情報取得

輻輳ウィンドウの値と、スロースタートフェイズか否かのフラグを送信ホストからゲートウェイに通知する必要がある。本研究では、独自フォーマットの UDP パケットを定期的に送ることでゲートウェイに通知を行うものとした。ゲートウェイの FPGA には、UDP パケットを解釈して各ストリームの状態を記録する回路が実装されている。

5 関連研究

LFN において、単一 TCP ストリームの通信帯域を増やす方式としては、High-Speed TCP [9]、Scalable TCP [13]、FAST TCP [2]、TCP Vegas [7] 等がある。High-Speed TCP と Scalable TCP では、輻輳ウィンドウが TCP-Reno/NewReno より増えやすいようパラメータを調節している。すなわち、ACK 到着時の輻輳ウィンドウの増加量を多く、ACK 不到着時の輻輳ウィンドウ減少量を少なくしている。そのため、LFN のリンク帯域に近い帯域を出すために必要な高い輻輳ウィンドウ値を得やすい。また、FAST TCP と TCP Vegas では、パケット往復時間から途中経路での輻輳によるバッファリングを検出する。この情報に基づき、輻輳ウィンドウ値を増減させる。結果として、パケットロスを抑えつつ安定して高い輻輳ウィンドウ値を保つ事が可能である。だが、TCP Vegas は、TCP-Reno/NewReno 等のパケットロスに基づく方式のコネクションが混在した場合に帯域を奪われる問題がある。逆に、FAST TCP は非輻輳状態で流量を積極的に増やすため、他の実装の TCP を圧迫する。そのため、占有的ネットワーク以外での使用に適さない。以上の単一ストリームを高速化するアプローチは、クラスタ間通信のように送信データが複数マシンに分散している場合への適用は困難である。

並列 TCP ストリームを用いて帯域増大を図る方式としては、まず、単一ホストによるアプローチ [6][12][17] がある。[12] では輻輳ウィンドウ値の増加速度を調節し、他の TCP コネクションとの公平性を保つよう工夫されている。また、[6] では、ストリーム群全体が 1 本の TCP-Reno/NewReno のストリームと似た振る舞いをするような方式を提案している。[17] では FPGA を搭載した NIC を用いてパケット送出間隔を制御する。そのため、バースト送信によるパケットロスを抑える事が可能である。同時に、ストリーム間での速度格差を抑えるため、送信帯域の公平分配を行う。これらのアプローチは単一ホストでのアプローチであり、送信データが複数ホストに分散している場合にそのまま適用する事はできない。

複数ホストにデータが分散した場合を対象とした並列 TCP ストリーム方式には [14] や [15] がある。しかし、[14] ではパケット間隔制御が行われていない。[15] は FPGA を用いたフィルターによりパケット間隔を調節している。しかし、利用可能なネットワーク帯域に応じて動的に送信速度を調整する機能が無い。また、スロースタートフェイズの同期パケットロスを防ぐための対策はなされていない。

6 おわりに

本稿では、FPGA 搭載ゲートウェイを用いて並列 TCP ストリームのパケット送出の順序と間隔を制御する機構の提案・実装を行った。パケット送出順序の制御は、スロースタートフェイズの同期パケット

ロスを防ぐために行う。この制御では、複数のTCPストリームのパケットが同時に落ちる事を防ぐため、同一ストリームのパケットを連続して送信する。また、パケット間隔の制御は、輻輳回避フェイズにおいて連続的にパケットが落ちることを防ぐために行う。この制御では、全てのストリームのパケットを一旦バッファリングし、均等な間隔で送信を行う。実装は、FPGAと2つの10ギガビット・イーサネットポートを搭載するネットワーク実験装置を用いて行った。

今後は、実装したシステムを長距離高速回線を用いて実験・評価する予定である。また、個々のストリームのTCPスタック状態をゲートウェイ内でエミュレートする事により、送信ホストからのTCPスタック状態通知を不要にする予定である。最終的には、個々のストリームへの帯域割り当てを統一的に行うシステムを構成・評価する予定である。

謝辞

本研究は、文部科学省科学技術振興調整費「重要課題解決型研究等の推進-分散共有型研究データ利用基盤の整備」および科学技術振興事業団CRESTによる研究領域「情報社会を支える新しい高性能情報処理技術」研究課題「ディペンダブル情報処理基盤」および21世紀COEにより実施された。

参考文献

[1] Aggarwal, A., Savage, S. and Anderson, T.: Understanding the Performance of TCP Pacing, in *Proc. of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2000)*, Vol. 3, pp. 1157–1165 (2000).

[2] Cheng Jin, David X. Wei, Steven H. Low, : FAST TCP: motivation, architecture, algorithms, performance, in *Proceedings of IEEE Infocom 2004* (2004).

[3] H. Kamezawa, M. Nakamura, J. Tamatsukuri, N. Aoshima, M. Inaba, K. Hiraki, J. Shitami, A. Jinzaki, R. Kurusu, M. Sakamoto, and Y. Ikuta, : Inter-layer coordination for parallel TCP streams on Long Fat pipe Networks, in *SC2004* (2004).

[4] J. Border et al., : RFC3135: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations (2001).

[5] Key Hiraki, Mary Inaba, Junji Tamatsukuri, Ryutarou Kurusu, Yuukichi Ikuta, Koga Hisashi, Akira Jinzaki, : Data Reservoir: Utilization of Multi-Gigabit Backbone Network for Data Intensive Research, in *Proceedings of the IEEE/ACM Supercomputing(SC2002)* (2002).

[6] Lar Eggert, John Heidemann, Joe Touch, : Effects of Ensemble-TCP, *ACM Computer Communications Review*, Vol. 30, No. 1, pp. 15–29 (2000).

[7] Lawrence S. Brakmo, Sean W. O'Malley, Larry L. Peterson, : TCP Vegas: New techniques for congestion detection and avoidance, in *Proceedings of the SIGCOMM '94 Symposium*, pp. 24–35 (1994).

[8] Makoto Nakamura, Junsuke Senbon, Yutaka Sugawara, Tsuyoshi Itoh, Mary Inaba, Kei Hiraki, : End-node transmission rate control kind to intermediate routers–towards 10Gbps era, in *PFLDnet 2004:Second International Workshop on Protocols for Fast Long-Distance Networks* (2004).

[9] Sally Floyd, : HighSpeed TCP for Large Congestion Windows (2003), RFC3649.

[10] Shreedhar, M. and Varghese, G.: Efficient Fair Queueing Using Deficit Round Robin, in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication (ACM SIGCOMM '95)*, pp. 231–242 (1995).

[11] T. Miyata, S. O., H. Fukuda: Impact of PacketSpacing Time on Packet Loss under LW for QoS of FEC-based Applications, in *IPSJ Sig Notes, Mobile Computing*, Vol. 6, pp. 7–13 (1998).

[12] Thomas J. Hacker, Brian D. Noble, Brian D. Athey, : Improving Throughput and Maintaining Fairness using Parallel TCP, in *Proceedings of IEEE Infocom 2004* (2004).

[13] Tom Kelly, : Scalable TCP: Improving Performance in Highspeed Wide Area Networks, in *First International Workshop on Protocols for Fast Long-Distance Networks* (2003).

[14] 亀澤 寛之, 中村 誠, 稲葉 真理, 平木 敬, 陣崎 明, 下見 淳一郎, 来栖 竜太郎, 中野 理, 鳥居 健一, 柳沢 敏孝, 生田祐吉：長距離・高バンド幅通信における並列TCPストリーム間の調停の実現, 先進的計算基盤システムシンポジウム (SACSIS2004), pp. 425–432 (2004年).

[15] 建部 修見, 小川 宏高, 児玉 祐悦, 工藤 知宏, 高野 了成, 関口智嗣：グリッドデータファームとGNET-1による日米間高速ファイル複製, 情報処理学会研究報告 2004-HPC-97 HOKKE2004, pp. 31–36 (2004年).

[16] 菅原 豊, 稲葉 真理, 平木敬：インテリジェントNICを用いた高帯域ネットワーク向けTCP通信方式, 情報処理学会研究報告 OS-97, No. 82 in 2004, pp. 57–64 (2004).

[17] 千本 潤介, 中村 誠, 稲葉 真理, 平木敬：FPGAを用いたNICレベルでのストリームマネージメントアーキテクチャ, 第1回リコンフィギャラブルシステム研究会論文集, pp. 263–270 (2003年).