高遅延広帯域ネットワークにおける並列TCPストリームの公平性解析

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，長距離広帯域ネットワーク上で複数のTCPストリームを用いたクラスタ間データ転送が行われる．広帯域で遅延が大きいネットワークでは，TCPストリーム間の速度ばらつきにより転送速度が低下することが知られている．本稿では，並列TCPストリーム間で速度不平衡が生ずるメカニズムを観察する．10ギガビット・イーサネットと遅延装置からなる実験環境を用いて並列TCPストリームを用いた遠距離通信をモデル化する．ハードウェアを用いて100nsの時間精度でトラフィックを観測する事で並列TCPストリームの微視的な振る舞いを可視化する．解析の結果より，ストリーム間のパケット送出順が保存されることにより速度不平衡が生じる事を示す．また，通信開始のタイミングを調整することにより速度を平均化できる事を示す．

Fairness Analysis of Parallel TCP Streams on Long-latency and High-bandwidth Network

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

Inter-cluster data transfer is performed using multiple TCP streams on high-speed long-distance network in recent applications. It is known that bandwidth become different among TCP streams on long-latency high-speed network. As a result, data transfer throughput is degraded. In this paper, we observe the mechanism that the throughput difference among streams is generated. Long-distance communication using parallel TCP streams is modeled using an experimental environment consists of 10 gigabit Ethernet and a delay emulator. Microscopic behavior of parallel TCP streams is visualized by observing traffic with 100ns time accuracy using hardware. From the result of the analysis, we show that the throughput become different because the packet transmission order is preserved among streams. In addition, we show that throughput difference is mitigated by adjusting the start time of communication.

1 はじめに

近年，高速な長距離ネットワークが実用化された．日本–アメリカ間ではIEEAF，アメリカ–ヨーロッパ間ではSURFnetなど，10ギガビット/秒クラスの広域ネットワークが利用可能になった．このような広域ネットワークのインフラにより，クラスタ間で高

速なデータ転送を行う応用が可能となった [4][6][2]. 例えば, Data Reservoir [4] では, ディスクデータを共有するために拠点間でクラスタ間データ転送を行う. これらの応用ではノードごとに別々に張った TCP ストリーム (並列 TCP ストリーム) を用いてデータ転送が行われる. 太平洋間, 大西洋間など遠距離のデータ転送では, ネットワークのパケット往復時間 (Round Trip Time, RTT) は数百ミリ秒に達する場合がある.

広帯域で遅延が大きいネットワーク (Long Fat pipe Network, LFN) では, TCP を用いた高効率なデータ転送が難しいことが知られている. TCP では, ネットワークの輻輳を避けるために, 送信済み受信未確認のデータ ("in-flight data") のサイズを送出側ホストで制限する. この制限値は輻輳ウィンドウサイズと呼ばれる. 輻輳ウィンドウサイズは新たな ACK パケットを受け取る毎に増やされ, パケットロス等により ACK が届かないと減らされる. TCP の流量はおおよそ以下の式で表せる.

$$\text{輻輳ウィンドウサイズ}/RTT$$

そのため, 遅延が大きいネットワークで高い帯域を得るには輻輳ウィンドウサイズが大きい必要がある. しかし, RTT が大きいと輻輳ウィンドウサイズの増え方が遅い. そのため, LFN では TCP ストリームが流量を回復するには長い時間がかかる.

並列 TCP ストリームでは, それ自身が起こす輻輳により一部のストリームが十分な流量を得ることができない場合がある [3]. その結果, ストリーム間で速度不平衡が生じる. LFN では遅いストリームの流量回復に時間がかかるため, この不平衡は短時間では縮まらない. その結果, 遅いストリームがボトルネックとなりシステム全体としてのデータ転送速度が低下する. 速いストリームに動的に多くのデータを割り当てる対処法では, ホスト間でのデータ分割を可変にする特別な機構を必要とする.

データ転送速度を改善するために, 我々はストリーム間の速度を平均化する方法を提案してきた [3][8]. DECP [3] では, Linux カーネルの TCP スタックレベルでストリーム間の速度を平均化する. Stream Harmonizer [8] は, 送信側サブネットのゲートウェイにおいてハードウェアによりパケットをスケジューリングしてストリーム間の速度を平均化する. これらの方式はストリーム間に一度生じた不平衡を元に戻すという原理での最適化であった. 不平衡が生じる事を未然に防ぐ最適化と組み合わせることにより, さらに性能が向上する可能性がある.

本稿では, LFN をモデル化した実験用ネットワークを用いて, 並列 TCP ストリーム間で速度不平衡が生じるメカニズムを観察する. 解析に 10 ギガビット・イーサネット (10GbE) と遅延装置からなる実験装置を用いる. この実験装置により, LFN の帯域とレイテンシをモデル化する. ジッタや競合などの影響を排除した実験環境を用いることで, ストリーム間の速度不平衡がこれらの要因に依存せずに生じることを示す. ハードウェアを用いて 100ns の時間精度でパケットヘッダを記録する. これにより並列 TCP ストリームの微視的なトラフィックの振る舞いを可視化する. ストリーム間のパケット送出順が保存されること, その結果送出順が遅くなったストリームほど速度が低下するという観測結果を示す. また, 通信開始のタイミングを制御することによりストリーム間の速度不平衡を解消することが可能である事を示す.

本稿では, 2 章で実験で使用する BIC TCP の概要を述べる. 次に 3 章で解析に用いる環境について説明する. 4 章で並列 TCP ストリームの振る舞いを解析する. 5 章で関連研究について述べる. 最後に 6 章でまとめを行う.

2 BIC TCP

評価で用いる BIC (Binary Increase Congestion Control) TCP [5] の概要を述べる. BIC TCP は加算的増加 (Additive Increase) と二分探索増加 (Binary Search Increase) を組み合わせた輻輳ウィンドウサイズ制御により, LFN でのリンク利用率向上と競合するストリームとの公平性を両立する事を目的とした TCP である. 現在広く使われている TCP Reno のトラフィックと親和性を持つよう設計されている.

BIC TCP では輻輳が起きると, その時点での輻輳ウィンドウサイズに近い値をターゲットウィンドウサイズとして記録する. 現在の輻輳ウィンドウサイズとターゲットウィンドウサイズの関係により, 輻輳ウィンドウサイズの増加が速い加算的増加と増加が遅い二分探索増加のいずれかが選ばれる.

現在の輻輳ウィンドウサイズとターゲットウィンドウサイズの差が一定以上である場合は 輻輳ウィンドウが加算的に増やされる. 輻輳ウィンドウサイズが 1 RTT ごとにパケット S_{max} 個分増やされる.

一方, 輻輳ウィンドウサイズがターゲットウィンドウサイズに近い場合は二分探索増加に切り替えられる. このフェイズでは輻輳ウィンドウサイズが RTT の対数オーダーでゆっくり増やされる. ウィンドウサイズの増加を遅くする事には, (1) 前回輻輳を起こした流量に近いのでウィンドウサイズの急激な増加を抑える, (2) 前回輻輳を起こした直前の流量で長時間安定することでリンク帯域を有効活用する, (3) 競合する TCP Reno のトラフィックが流量を伸ばす時間を与え公平性を向上させる, という目的がある.

パケットロスが起きた場合は輻輳ウィンドウサイ

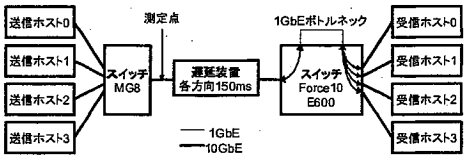

図 1: 実験用ネットワークの構成

ズが $1-\beta$ 倍 $(0<\beta<1)$ に減らされる (Multiplicative Decrase).

BIC TCP では Additive Increase の係数 S_{max} を TCP Reno の 1 パケットより大きくし，Multiplicative Decrease のパラメータ β を TCP Reno の 0.5 より小さく設定することで，LFN で輻輳ウィンドウサイズを大きな値に保つことを可能にする．

BIC TCP は最近の Linux に標準実装されている．本稿では，Linux の標準実装をそのまま用いる．

3 解析方法

10GbE と遅延装置からなる実験環境を用いて並列 TCP ストリームの振る舞いを解析する．評価に用いるネットワークを 図 1 に示す．送信ホスト 4 台と受信ホスト 4 台の間で並列 TCP での通信を行う．1 ホストごとに 1 ストリームを張り，合計で 4 ストリームで通信を行う．これらのストリームは 1GbE のボトルネックリンクを通過する．このリンク以外は全て 10GbE である．ネットワークの途中経路に遅延装置 (ANUE GE10) を挿入する．この装置は，バッファを用いて仮想的に遅延を発生させる．実験では遅延を片方向 150ms，往復で 300ms に設定する．この機能により長距離ネットワークの振る舞いをモデル化する．遅延はハードウェアにより正確に挿入されるため，ジッタなどの外乱を排除した評価が可能である．スイッチ Foundry MG8 を用いて遅延装置と 4 台の送信ホストを接続する．スイッチ Force10 E600 を用いて遅延装置と 4 台の受信ホストを接続する．遅延装置と受信ホストの間を行き来するトラフィックは VLAN 設定により 1GbE のボトルネックリンクを通過する．スイッチ MG8 と遅延装置を結ぶ 10GbE リンクにおいて両方向のトラフィックを解析する．

3.1 トラフィック解析装置 TAPEE

トラフィックの解析には TAPEE [7] を用いる．この装置により 10GbE のリンクを流れるパケットのヘッダをワイヤレートで記録することが可能である．各ヘッダには 100ns 精度のタイムスタンプが付加される．これらの機能により，10GbE を流れるトラフィックの微視的な振る舞いを可視化することが可能である．図 2 に装置外観を示す．この装置は 10GbE のポートを 2 つ持つ．2 方向のトラフィックを同時に測定可能である．TAPEE は，受信したパケットのヘッダ部分を抽出する．同時に，タイムスタンプを付加する．抽出したヘッダとタイムスタンプを複数パケット分まとめる．このデータを 1 個の UDP パケットのペイロードに格納し外部の記録用ホストに送出する．ヘッダだけを抽出する事でデータ量を減らす．この処理により記録用ホストで取りこぼし無く受信する事を可能にする．

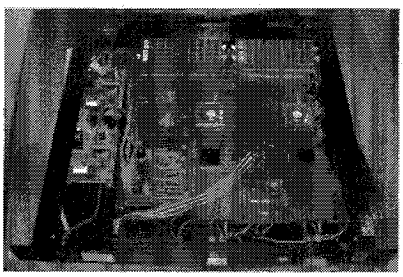

図 2: トラフィック解析装置 TAPEE の外観

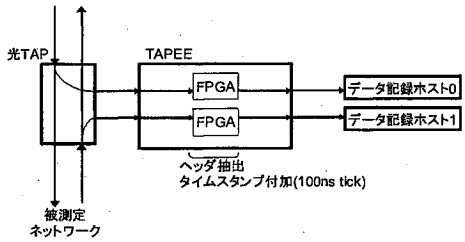

図 3: TAPEE を用いたヘッダのキャプチャ

図 3 に TAPEE の接続図を示す．被測定ネットワークに光 TAP を入れて，2 方向の光信号を TAPEE に入力する．TAPEE により抽出されたヘッダとタイムスタンプのデータは各方向毎に別々のデータ記録ホストに送出される．データ記録用ホストでは受信したデータをディスクに記録する．

3.2 エンドホストの設定

8 台のエンドホストは全て同じ仕様である．表 1 にエンドホストの仕様を示す．輻輳制御には Linux で標準的に用いられる BIC TCP を用いる．BIC TCP の Additive Increase 係数 (`max_increment`) は 16，Multiplicative Decrease 係数は $1-\beta = 819/1024(\approx 0.8)$ とした．これらはいずれもデフォルトの値である．ロスが発生する事が想定されるため selective ACK (SACK) を用いる．OS は Linux 2.6.16.28 を用いる．以前 Linux に存在した BIC TCP の実

表 1: エンドホストの仕様

CPU	Xeon 2.4GHz × 2
メモリ	1GB
OS	Linux 2.6.16.28 (i386)
輻輳制御	BIC TCP, SACK あり
NIC	Chelsio T110 (TOE なし)
NIC ドライバ	cxgb2toe-2.2.0

表 2: iperf のコマンドライン引数

送信側	-c サーバ名 -i 1 -w 256M -M 9100 -l 128k -t 100 -F (mmap ファイル)
受信側	-s -i 1 -w 256M -M 9100 -l 64k

装の不具合はこのバージョンでは修正されている．Chelsio T110 10GbE ネットワークカード (NIC) を通信に使用する．T110 の TCP offload engine (TOE) は使用しない．ドライバには cxgb2toe-2.2.0 を用いた．

iperf を用いて 100 秒間の通信を行う．送信側の iperf は mmap によりユーザスペースからカーネルスペースへのコピーを省略したバージョン [1] を用いた．受信側では iperf 1.7.0 を改造無しで使用した．iperf のコマンドライン引数を送信側，受信側それぞれ表 2 に示す．

4 評価

3 章で示した実験環境を用いて並列 TCP ストリームの振る舞いを解析する．まず，全てのストリームが同時に通信を開始した場合について解析を行う．この場合，パケット送出のわずかな時間の差が拡大されてストリーム間に不公平性が生じる事を示す．次に，パケット送出のタイミングを制御することでストリーム間の不公平性を抑えられる事を示す．

4.1 パケット送信順によるばらつきの発生

4 つのストリーム全てを同時に開始した場合の各ストリームの流量変化を図 4 に示す．横軸が通信開始からの経過時間，縦軸が流量である．全時間を 1 RTT (300ms) の区間ごとに分け，各区間ごとに流量の平均をとりプロットした．5 秒付近までは全てのストリームがほぼ同じ流量である．その後，stream 3, 1, 2, 0 の順に流量低下がおきている．その結果，ストリーム間で速度格差が生じている．その後は，どのストリームもほぼ一定の値を保っている．序盤でついた速度格差が保存されている．速度格差は少

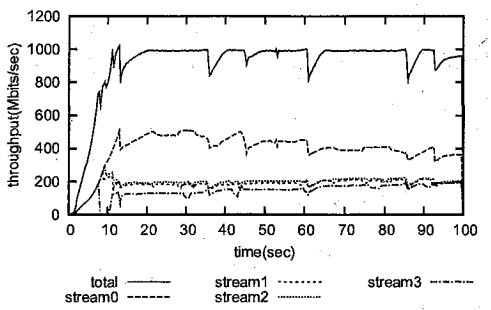

図 4: 各ストリームの流量，300ms 区間ごとの平均

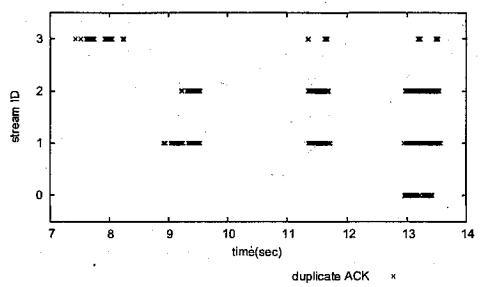

図 5: 各ストリームの重複 ACK パケット観測時刻

しずつ減少しているが，少なくとも 100 秒の範囲内では解消されていない．

各ストリームでパケットロスがおきた時刻を調べるため，番号が重複した ACK パケットが検出された時刻をプロットした．結果を図 5 に示す．重複 ACK パケットの発生時刻より，stream3, 1, 2, 0 の順にロスが発生している事が示されている．

ストリーム毎に別のタイミングでロスが生じる理由を調べるため，1 RTT 内でのパケット送出のタイミングを調べた．図 6 に SYN パケットの次の RTT における各ストリームのパケット送出タイミングを示す．ストリーム毎に送出のタイミングがわずかに異なる．最も早い stream 0 と最も遅い stream 3 では約 2ms の差が生じている．

次に，図 7 に stream 3 の流量が落ちた時刻付近における微視的な流量の変化を示す．10ms の区間ごとに流量の平均を取った．4 つのストリームが 1RTT 内のある区間でまとめてパケットを送っている．また，ストリーム毎にパケット送出の区間が分かれている．stream 0, 2, 1, 3 の順にパケットを送出している．図 6 のパケット送出順がそのまま保存されている．順番が後のストリームは直前に他のストリームがパケットを送っているためスイッチのバッファにパケットがたまっている可能性が高い．そのため，ロスを起こしやすい．実際，図 5 は stream 3, 1, 2, 0 の順にロスが発生している事を示している．

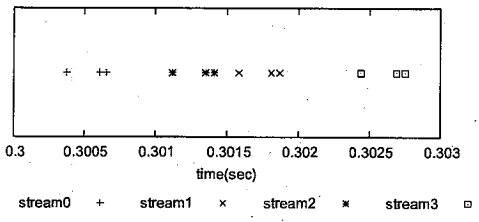

図 6: 通信開始時のパケット送出時刻

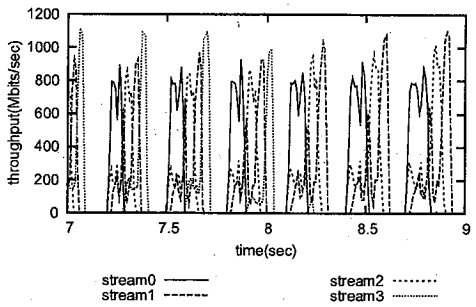

図 7: 微視的な流量変化, 10ms 区間ごとの平均

通信開始時のパケット送出順が保存されるメカニズムを調べるため，各 RTT 毎に stream 0 と stream 3 の送出時間差を調べる．RTT 内でこれら 2 つのストリームで最初に送られたパケット同士の時刻を比較する．図 8 に結果を示す．図では，ある RTT に送られたデータパケットに対して ACK パケットの間隔が開いている．送出レートが短期的に 1Gbps を越えると 1GbE のボトルネックリンクを通過する際にパケット間隔が広がる．その結果，ACK パケットの間隔は同じ RTT に送出したデータパケットの間隔より広くなる．TCP では ACK を受信してから次のデータパケットを送る．そのため，ACK パケットの間隔が次の RTT でのデータパケット送出間隔として保存される．そのためストリーム間の送出時間差が拡大されながら保存される．

以上より，通信開始時のわずかな送出タイミングの違いが拡大されながら保存されること，その結果送出順が後ろのストリームほど早期に流量が低下することが分かった．

4.2 送出タイミング調整による性能改善

送出タイミングの調整によりストリーム間の不公平性を解消可能である事を示す．前節では送出順が遅いストリームほど流量が低下する事を示した．ストリーム間でパケットロスの起きやすさが異なっていた事が不公平性の原因であった．パケットロスの起きやすさをストリーム間でそろえる一つの手法として，通信の開始時間をずらす方法を示す．スト

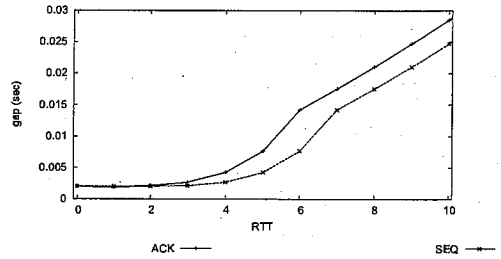

図 8: stream 0 と stream 3 の送出時間差，RTT 内で最初に送られたパケット同士で比較

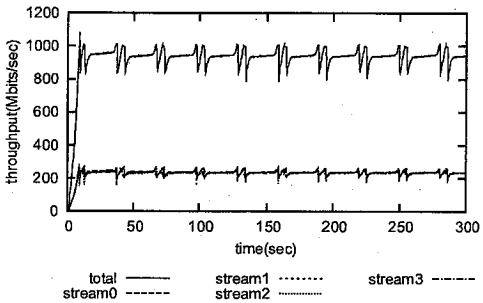

図 9: 送出タイミングを 1/4 RTT ずつずらした場合の流量変化, 300ms 区間ごとの平均

リームが n 本ある場合に，これらのストリームを RTT/n ずつ間隔をあけながら順に開始する．本稿の例では $n = 4$ なので，1/4RTT = 75ms ずつ間隔をあけながら順に 4 つのストリームを開始する．こうすることで 4 つのストリームが 1 RTT の間に等間隔で通信を行う．

実際に 1/4 RTT ずつ開始時間をずらして 4 つのストリームを開始させた場合の流量変化を図 9 に示す．4 つのストリームがほぼ 250Mbps で安定している．1Gbps のボトルネックリンク帯域を公平かつ 100%に近い効率で使用している．通信開始後に全てのストリームで流量がほぼ同じタイミングで落ちている．その結果，全てのストリームがほぼ同じ流量で安定した．何回か実験を繰り返したが，全て同様の結果になった．

送出タイミングを 1/4 RTT ずらした場合の微視的な流量変化を 図 10 に示す．1 RTT 中で 4 つのストリームがほぼ等間隔にまとめてパケットを送出している．その結果，時間軸方向で見た場合に 4 つのストリームで条件がほぼ対称になっている．

以上から，送出タイミングを制御する事により並列 TCP ストリームの速度格差が発生しなくなる事が示された．

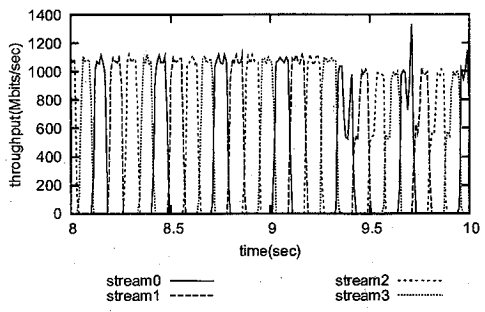

図 10: 送出タイミングを 1/4 RTT ずらした場合の微視的な流量変化, 10ms 区間ごとの平均

5 関連研究

DECP [3] は，Linux の TCP スタックレベルで並列 TCP ストリーム間での流量平均化を行う手法である．エンドホスト間で輻輳ウィンドウサイズの情報を交換する．その情報を元に速すぎるストリームはウィンドウサイズを下方修正する．Stream Harmonizer [8] は，送信側サブネットのゲートウェイにおいて並列 TCP ストリームの速度平均化を行う．速いストリームの RTT を増やすことにより速度を抑制する．Stream Harmonizer はエンドホストから透過的に速度平均化を行えるという特徴がある．これらの手法は一度生じた速度格差を元に戻すという方法で最適化であった．本研究では速度格差そのものが発生するメカニズムを議論した．

6 おわりに

本稿では，LFN 上で並列 TCP ストリームを用いた場合にストリーム間に速度格差が発生するメカニズムを解析した．評価には 10GbE と遅延装置からなる実験環境を使用した．ハードウェアを用いて 100ns の時間精度でトラフィックを解析した．解析の結果，1 RTT 内でのストリーム間の送信順序が保存される事が分かった．送信順序が遅いストリームほど早く流量が低下することが分かった．また，ストリームの開始時間を調整することで全てのストリームで流量をほぼ揃えられる事が分かった．

本稿での評価では遅延装置を使用しているためジッタなどの影響が排除されている．今後は，実際の長距離広帯域ネットワーク上を用いて並列 TCP ストリームの振る舞いを解析する予定である．また，並列 TCP ストリームで速度格差が起きることを未然に防ぐ方式の検討を行う予定である．

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] http://data-reservoir.adm.s.u-tokyo.ac.jp/press-20070508/.

[2] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, : Data Management and Transfer in High-Performance Computational Grid Environments, *Parallel Computing*, Vol. 28, pp. 749–771 (2002).

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

[4] K. Hiraki, M. Inaba, J. Tamatsukuri, R. Kurusu, Y. Ikuta, K. Hisashi, A. Jinzaki, : Data Reservoir: Utilization of Multi-Gigabit Backbone Network for Data Intensive Research, in *Proceedings of the IEEE/ACM Supercomputing(SC2002)* (2002).

[5] L. Xu , K. Harfoush and Injong Rhee, : Binary Increase Congestion Control for Fast Long-Distance Networks, in *Proceedings of Twenty-thrid Annaul Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, Vol. 4, pp. 2514–2524 (2004).

[6] N. Ali, M. Lauria, : SEMPLAR: high-performance remote parallel I/O over SRB, in *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, pp. 366–373 (2005).

[7] T. Yoshino, J. Tamatsukuri, K. Inagami, Y. Sugawara, M. Inaba, K. Hiraki, : Analysis of 10 Gigabit Ethernet using Hardware Engine for Performance Tuning on Long Fat-pipe Network, in *PFLDnet 2007 (Fifth International Workshop on Protocols for FAST Long-Distance Networks)* (2007).

[8] Y. Sugawara, M. Inaba, K. Hiraki, : Flow Balancing Hardware for Parallel TCP Streams on Long Fat pipe Network, in *International Conference on Future Generation Communication and Networking (FGCN 2007), to appear* (2007).