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

2007－HPC－111 (45)
2007／8／3

ゲートウェイによる並列TCPのウィンドウサイズ平均化

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

並列TCPストリームを用いたデータ転送では，ストリーム間の速度ばらつきがあるとデータ転送時間が増大する事が知られている [6]．パケットロスにより一部のストリームで輻輳ウィンドウサイズが減少すると速度ばらつきの原因となる．本研究では，輻輳を検出した時に並列TCPストリーム全体の流量を絞ることで，輻輳ウィンドウサイズの減少を緩和する方式を提案する．広域ネットワークの入り口に配置したゲートウェイが輻輳を検出し並列TCPストリーム全体の流量を抑制する．輻輳はACK番号の重複より検出される．ACKパケットの受信側ウィンドウサイズを変更することにより流量が制御される．ns-2シミュレータを用いた評価により，提案方式により輻輳ウィンドウサイズの減少を緩和し，データ転送時間を短縮する事が可能である事を示す．

Balancing Window Size of Parallel TCP Streams using Gateway

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

In data transfer using parallel TCP streams, it is known that throughput difference among streams increases the transfer time [6]. When the congestion window size is reduced by packet loss for part of streams, it leads to a throughput difference among streams. In this paper, we propose a method to mitigate congestion window size reduction by throttling down the flow of the parallel TCP streams when congestion is detected. A gateway that is placed at the entrance of the wide-area network detects congestion and throttles down the flow of the parallel TCP streams. Congestion is detected by duplicate acknowledgement packets. The flow of the parallel TCP streams is controlled by modifying the receiver window size fields of the acknowledgement packets. Through an evaluation using ns-2 simulator, we show that the proposed mechanism can mitigate congestion window size reduction and reduce data transfer time.

1 はじめに

近年，長距離・広帯域ネットワークを用いたクラスタ間データ転送技術が求められている．例えばData-Reservoir [2] では拠点間でクラスタのディスクデータを共有するため，広域ネットワークを用いたデータ転送を行う．クラスタ間でのデータ転送でノードごとに別々のTCP/IPコネクションを張ってデータ転送を行う場合，システム全体としては複数のTCPストリームを用いて転送を行う．このように協調してデータ転送を行うTCPストリームの組を並列TCPストリームと呼ぶ．

広帯域で遅延が大きいネットワーク (Long Fat pipe Network, LFN) における並列TCPストリームを用いたデータ転送では，遅いストリームが転送時間のボトルネックとなる事が知られている [6]．TCPではACKが届く毎に流量を増やし，パケットロス等でACKが届かない場合に流量を絞る．輻輳が起きた時にストリーム毎にパケットの落ち方が異なる等の理由から，ストリーム間で同じ流量を保てない場合がある．その結果，他のストリームより遅いストリームが発生し全体のデータ転送時間のボトルネックとなる．LFNではパケット往復時間 (Round Trip Time, RTT) が長いためACK到着による流量の回復に時間を要する．そのため，遅いストリームが速度を回復するために時間を要する．ストリーム間の速度ばらつきに対応するためにアプリケーションレベルで速いストリームに多くのデータを割り当てる手法は，常に適用可能とは限らない．

TCPストリーム間の速度ばらつきを緩和する方法が提案されてきた．プロトコルレベルでは，BIC-TCP [3] 等のアルゴリズムはストリーム間の公平性を保てるよう考慮されている．しかし，本稿で示すように短期的に見た場合は公平性が必ずしも保たれるとは限らない．ストリーム毎に独立に制御を行うのではなく，ストリーム間で情報を交換し流量制御に用いる方式が提案されている．DECP [6] では，個々のストリームの状態を中央のサーバで集計し，各ストリームのCWNDを調整する．この方式により，ストリーム間の速度ばらつきを抑えられる事が示されている．しかし，エンドホストと相互に情報をやりとりする特別な機構が必要である．

本稿では，広域ネットワークの入り口に配置したゲートウェイにおいて，遅いストリームの発生を避けるために輻輳発生時に並列TCPストリーム全体の流量を抑制する方式を提案する．提案方式は，スロースタート時のパケットロスが長く続くことで一部のストリームの流量が極端に減る現象を防ぐ事を目的としている．どれか一つのストリームで輻輳を検出した場合に，全てのストリームの流量を抑制する．輻輳はACK番号の重複から検出する．ACKパケットの受信側ウィンドウの値を小さな値に書き換えることで流量を抑制する．ネットワークの中間経路で得られる情報のみを用いることで，ゲートウェイで集中的かつ独立して制御を行う事を可能とする．本稿ではns-2ネットワークシミュレータを用いた評価を行う．評価では，輻輳により一部のストリームが極端に遅くなる現象を提案方式により抑制し，データ転送時間を短縮することが可能である事を示す．

2章でストリーム間の速度ばらつきについて述べる．次に，3章でゲートウェイを用いて速度ばらつきを抑える方式について説明する．4章では，ネットワークシミュレータns-2を用いた評価を行う．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 ストリーム間の速度ばらつき

パケットロスによる流量の変化がストリーム間で異なると，速度ばらつきの原因となる．TCPでは，ACKを待たずに送出できるデータ量を輻輳ウィンドウサイズと呼ばれる値以下に制限している．そのため，データを送ってからACKが返るまでの間，すなわち1 RTTの間に送れるデータ量が輻輳ウィンドウサイズ以下に制限される．輻輳ウィンドウサイズはACKが到着する毎に増やされ，パケットロス等でACKが届かない事が検出されると減らされる．図1は，並列TCPストリームを用いた通信をネットワークシミュレータns-2上で行った時の輻輳ウィンドウサイズとACK番号の時間変化を示す．エンドホストでは全てBIC-TCP[3]を用いた．各ストリームが62500パケットずつを転送する．図から，初期の輻輳で1つのストリームの輻輳ウィンドウサイズが極端に低下し，データ転送完了に他のストリームの約3倍の時間を要していることが分かる．

TCPではスロースタートを行う上限となる輻輳ウィンドウサイズをssthreshと呼ばれるパラメータで管理している．輻輳ウィンドウサイズがssthresh

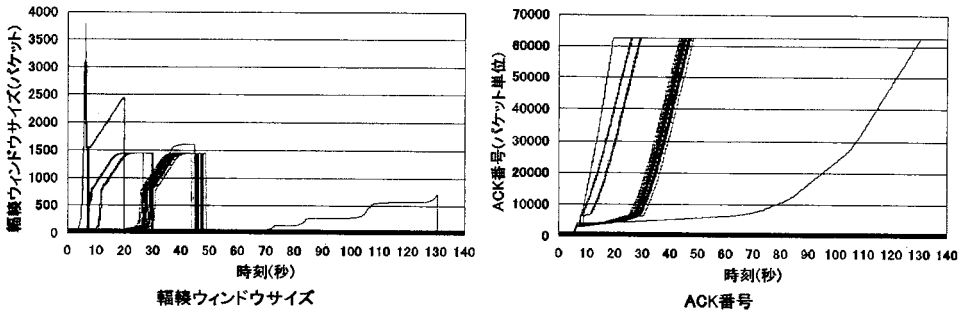

図 1: 輻輳ウィンドウサイズと ACK 番号の変化

値を越えるとスロースタートから輻輳回避フェイズに移り，以降輻輳ウィンドウサイズの増加が遅くなる．輻輳を検出した場合，回復処理において輻輳ウィンドウサイズが ssthresh 値として保存される．

スロースタート後のパケットロスで一部のストリームで ssthresh 値が低下し，ストリーム間の速度格差が生じている．図 2 は，スロースタートフェイズで輻輳が発生した時刻付近での各ストリームの ssthresh 値の変化を示す．最も遅いストリームは ssthresh の値が 2 パケット分まで低下している．その結果，以降のスロースタートは輻輳ウィンドウサイズが 2 に達した時点で終了してしまう．そのため，輻輳ウィンドウサイズの増加速度が低下し，他のストリームに比べて速度が低下している．

輻輳の検出時刻が一部のストリームで遅れている事が ssthresh 値が低下する要因になっている．図 3 にスロースタートにおける輻輳発生時刻付近での各ストリームの輻輳ウィンドウサイズ変化を示す．図から，輻輳を検出して輻輳ウィンドウサイズが減少する時刻がストリームによって異なる事が分かる．そのため，あるストリーム速い時刻に輻輳ウィンドウを減少させた後に別なストリームが流量を絞らずにパケットを送り続ける．その結果，輻輳ウィンドウを大きく減少させたストリームが再びパケットロスを起こし，低下した輻輳ウィンドウサイズが ssthresh 値として保存される．

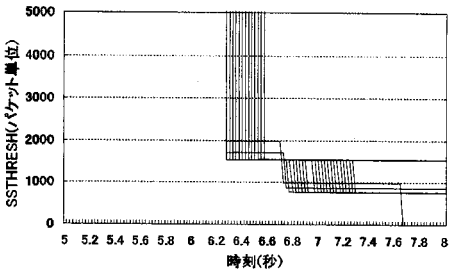

図 2: 輻輳直後の ssthresh 値の変化

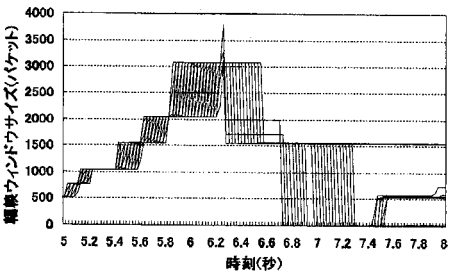

図 3: 輻輳直後の輻輳ウィンドウサイズの変化

3 ばらつきの抑制

提案方式では ssthresh 値の低下を防ぐことにより TCP ストリーム間の速度ばらつきを緩和する事を目的とする．図 1 の例では，輻輳が起きて輻輳ウィンドウサイズが縮小した後に再びパケットロスが起き ssthresh の値が低下していた．提案方式では輻輳が起き次第，速やかに流量を抑えることにより，続けてパケットロス起きる事を防ぐ事を目的とする．

図 4 のように送信側の広域ネットワークの入り口に配置した専用のゲートウェイにおいて輻輳の検出と流量制御を行う．どれか一つのストリームで輻輳が検出され次第，全てのストリームの流量を抑制する．一部のストリームでエンドホストでの輻輳検出が遅れ，流量を絞る制御が遅れる現象を防ぐ事を目

的とする．制御対象となるストリームはあらかじめIP アドレス等により指定されるものと想定する．

ACK 番号の重複から輻輳の開始を検出する．一般的な TCP の実装では輻輳によりパケットロスが起きた場合，その直前のシーケンス番号に対するACK パケットが複数連続して送出される．ゲートウェイにおいて ACK パケットを監視することで，ACK 番号の重複を検出する．ACK 番号が重複した場合，ACK 番号が新しい値に変化するまで流量を抑制する．何回 ACK 番号が重複した場合に流量の抑制を行うか選択の余地がある．提案方式は LFN での TCP スロースタート時に起きる大規模な輻輳を対象とするため，評価では 100 回の重複で抑制を行う事とした．

ACK パケットの受信側ウィンドウサイズを書き換える [4] ことで送信側の流量を抑制する．ネットワーク中間経路で送信側に流量を抑制させる方式としては，(1) パケットをバッファリングして RTT を伸ばす，(2) 受信側ウィンドウサイズを書き換える，(3) パケットを落として輻輳ウィンドウサイズを減少させる，(4) ECN[5] を用いる，などの方法がある．(1) は RTT が伸びる事から TCP の Retransmission Timeout (RTO) を引き起こさないためのメカニズムを必要とする．(3) と (4) では抑制の操作自体が ssthresh 値を減少させてしまう可能性がある．そのため，提案方式では (2) の方法を用いる．受信側ウィンドウサイズは，流量を抑えるため直前に観測されたパケット流量よりわずかに小さな値に設定する．評価では観測された流量の 0.8 倍とした．

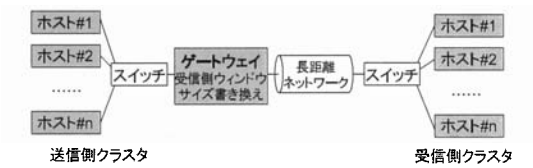

図 4: ゲートウェイにおけるウィンドウサイズの制御

4 評価

ネットワークシミュレータ ns-2 を用いた評価を行う．ssthresh 値の低下が性能のボトルネックになっている場合は提案方式により ssthresh 値を増加させデータ転送時間を短縮できる事を示す．

4.1 評価方法

ネットワークシミュレータ ns-2 を用いて提案方式によるストリーム間速度ばらつき抑制の効果を評価する．エンドホストの TCP 輻輳制御アルゴリズムには Linux 等に標準実装されている BIC[3] を用いる．TCP パケットのデータサイズは 100Kbit とする．

評価に用いるネットワークではクラスタ間での並列 TCP 通信をモデル化する．図 5 に評価で用いるネットワークトポロジを示す．TCP 送信ホストと受信ホストを n 台ずつ用意する．クラスタ全体では n 本の並列 TCP ストリームを用いてデータを転送する．TCP 送信側と受信側ホスト 1 台ずつからなるペアを n 組作る．各ペアは，送信ホストから受信ホストへ 1 本の TCP コネクションを貼ってデータ転送を行う．$n = 32$ の場合を基本とするが $n = 8$ の場合を別途評価する．

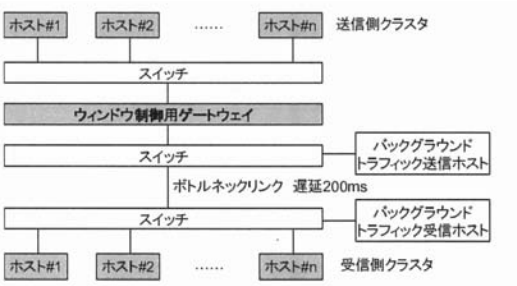

図 5: 評価に用いるネットワークトポロジ

評価では途中経路に遅延が大きいボトルネックリンクを入れ，LFN を想定したパラメータを設定する．図 5 で，200ms の遅延を持つリンクがボトルネックリンクという位置づけである．このリンクにバックグランドトラフィックとして UDP パケットを流す．図中のリンク帯域は全て同一であるため，このリンクにより TCP データ転送レートの上限が決まる．リンク帯域が 10 Gbps の場合と 20Gbps の場合について評価を行う．ボトルネックリンクに接続するスイッチのキューサイズは，ボトルネックリンクの遅延帯域積に等しい．それ以外の部分はパケットロスが起きないよう十分に大きなキューサイズとした．キューは全て drop-tail アルゴリズムで管理される．バックグラウンドトラフィックの流量はリンク帯域の 25%の場合と 75%の場合を評価する．図中，特に表記がないリンクの遅延は 1μs であ

表 1: 評価パラメータ一覧

リンク帯域	バックグラウンド	ストリーム本数	パケット送出個数
10Gbps	2.5Gbps	32	62500
20Gbps	5Gbps	32	125000
10Gbps	7.5Gbps	32	20833
10Gbps	2.5Gbps	8	25000

る．したがって，TCP 送信ホスト-受信ホスト間の RTT はおよそ 400ms である．なお，実際の LFN では RTT 約 500ms，リンク帯域が約 10Gbps という例がある [1]．ボトルネックリンクから見て送信ホスト側にウィンドウ制御用のゲートウェイが配置されている．このゲートウェイで提案方式の制御を行った場合とそうでない場合とを比較する．

各送信ホストが同時に転送を開始し，一定個数のパケットを送り終えたホストは転送を停止する．各ホストが送出するパケット数は，ボトルネックリンクの利用可能帯域を全ストリームで公平に残さず使用した場合に全てのパラメータで同じ時間 (26.7 秒) で転送が終わるよう調整した．データ転送はシミュレーション上の時刻で 1 秒の時点から開始する．表 1 に評価パラメータ一覧を示す．パケット送出個数は 1 ストリームあたりの個数である．

4.2 ばらつき抑制の効果

図 6 に，提案方式を適用した場合の輻輳ウィンドウサイズと ACK 番号の変化を示す．ストリーム 32 本，バックグランドトラフィック 2.5Gbps，リンク帯域 10Gbps である．図 1 は，同じ条件下で提案方式を適用しなかった場合の輻輳ウィンドウサイズと ACK 番号である．図から，提案方式を適用した場合はデータ転送のボトルネックとなる最も遅いストリームの速度が改善されている事が分かる．その結果，提案方式を適用しない場合のデータ転送時間 129.0 秒が適用時は 46.1 秒と約 30%程度に改善されている．

表 2 に，リンク帯域，バックグラウンド帯域，ストリーム本数を変化させた場合の ssthresh 最小値と転送所要時間を示す．通信を開始してから，全てのストリームが所定のパケットを送り終えて ACK を受信するまでの時間を転送の所要時間とする．ssthresh 値については，転送開始直後のスロースタート段階で輻輳が起きた後の値で，全てのストリームの中で

表 2: ssthresh 最小値と転送所要時間

リンク帯域	バックグラウンド	stream 本数	提案方式	ssthresh 最小値	転送所要時間 (秒)
10Gbps	2.5Gbps	32	なし	2	129.0
10Gbps	2.5Gbps	32	あり	768	46.1
20Gbps	5Gbps	32	なし	2	174.7
20Gbps	5Gbps	32	あり	1536	56.55
10Gbps	7.5Gbps	32	なし	82	54.35
10Gbps	7.5Gbps	32	あり	78	57.1
10Gbps	2.5Gbps	8	なし	3014	65.0
10Gbps	2.5Gbps	8	あり	3014	65.0

の最小値を示す．表では，リンク帯域を 20Gbps に増やした場合は提案方式による改善がみられる．提案方式による改善がみられた場合は，元々の ssthresh 値が小さく，提案方式を用いた場合に ssthresh の値が増加している事が分かる．一方，ストリーム本数を 8 本にした場合，およびバックグラウンドトラフィック帯域を 7.5Gbps に増やした場合は提案方式による改善がみられない．これらの例では元々 ssthresh の値がそれほど低下しておらず，提案方式による ssthresh 値の改善がみられない．

評価の結果，ssthresh 値の減少によりストリーム間に大きなばらつきが生じている場合，提案方式を適用すると ssthresh 値の減少が緩和されデータ転送時間が短縮されることが分かった．元々 ssthresh 値がそれほど減少していない場合は，提案方式による変化はみられない事が分かった．

5 関連研究

TCP の輻輳制御アルゴリズムレベルでストリーム間の速度公平性を改善する試みがなされている．BIC-TCP [3] 等の方式では，ストリーム間で公平性が保たれるよう考慮されている．しかし，本稿で示したように LFN では一度ストリーム間に生まれた速度格差が短時間では縮まらない場合がある．Explicit Congestion Notification(ECN) [5] はパケットロスに依らず中間ノードのスイッチから明示的に輻輳情報をエンドホストに通知する方式である．この方式によりストリーム間の公平性が向上する事が示されている．ECN を適用するためには中間経路のスイッチが ECN に対応している必要がある．

公平性向上のためにストリーム間で情報を交換するアプローチとして，DECP [6] が提案されている．

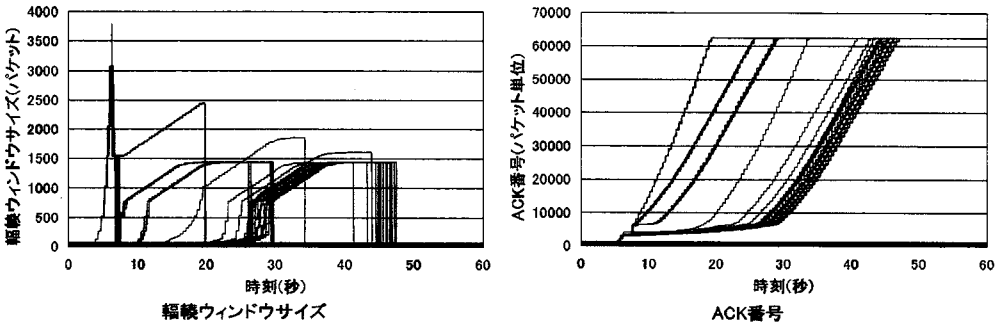

図 6: 輻輳ウィンドウサイズと ACK 番号の変化, ウィンドウ制御有り

DECP では専用の情報交換用のサーバを用いて個々のストリームの情報を交換し流量を調整する. この方式により, ストリーム間の速度格差を緩和できる事が示されている. この方式ではエンドホストに専用のソフトウェアが必要である.

6 おわりに

本稿では, 輻輳発生時に並列 TCP ストリーム全体のトラフィックを抑えることで, 一部のストリームが遅くなる現象を緩和する方式を提案した. 提案方式では, 広域ネットワークの入り口に配置したゲートウェイにおいて輻輳の検出とトラフィックの抑制を行う. ACK 番号の重複から輻輳の開始を検出する. ACK パケットに含まれる受信側ウィンドウサイズを書き換えることによりトラフィックを抑制する. 本稿ではネットワークシミュレータ ns-2 を用いて提案方式の評価を行った. 評価の結果, ssthresh 値の減少によりストリーム間に大きなばらつきが生じている場合, 提案方式を適用すると ssthresh 値の減少が緩和される事が分かった. その結果, 評価の条件下では転送所要時間が最善の場合で約 30%程度に改善された. 元々 ssthresh の値に落ち込みが見られない場合は提案方式による改善はみられない事が分かった. 今後は, より一般的な条件下での評価, アルゴリズムの改良, ハードウェアを用いた実装・評価を行う予定である.

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/lsr-200612-02/.

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

[3] 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).

[4] Narvaez, P. and Siu, K.: An Acknowledgment Bucket Scheme for Regulating TCP Flow over ATM, in *Computer Networks and ISDN Systems Special issue on ATM Traffic Management* (1998).

[5] S. Floyd, : TCP and Explicit Congestion Notification, in *ACM Computer Communications Review*, Vol. 24, pp. 8–23 (1994).

[6] 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).