社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

ソフトウェアによる精密ペーシング方式を用いたTCP通信性能の改善

高野　了成[†,††]　工藤　知宏[†]　児玉　祐悦[†]　松田　元彦[†]　岡崎　史裕[†]
石川　裕[†††,†]

† 産業技術総合研究所, グリッド研究センター　〒101–0021 東京都千代田区外神田 1–18–13 秋葉原ダイビル 11F
†† 株式会社アックス　〒101–0021 東京都千代田区外神田 3–14–3 福栄秋葉原ビル 7F
††† 東京大学理学部情報科学科　〒113–0033 東京都文京区本郷 7–3–1
E-mail: †takano-ryousei@aist.go.jp

あらまし　本来送信すべきパケットの間に，ギャップパケットを挿入することによって，パケット送信間隔を，精密かつ任意の大きさにスケジューリングする方式を提案する．提案方式を用いてペーシングを実現するソフトウェアPSPacer を実装し，高帯域遅延積ネットワークにおける TCP/IP 通信性能への効果を評価する．その結果，スロースタート時のパケットロスを削減し，通信性能を向上できることを示す．さらに，ボトルネック帯域が既知の場合は，ボトルネック帯域に近い，高いネットワーク利用効率が得られることを示す．
キーワード　パケットスケジューリング，ペーシング，ギャップパケット，BIC TCP

Improving TCP Performance by Using Precise Software Pacing Method

Ryousei TAKANO[†,††], Tomohiro KUDOH[†], Yuetsu KODAMA[†], Motohiko MATSUDA[†], Fumihiro OKAZAKI[†], and Yutaka ISHIKAWA[†††,†]

† Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Japan　1–18–13, Sotokanda, Chiyoda–ku, Tokyo, 101–0021, Japan
†† AXE, Inc.　3–14–3, Sotokanda, Chiyoda–ku, Tokyo, 101–0021, Japan
††† University of Tokyo, Dept. of Information Science　7–3–1, Hongo, Bunkyo–ku, Tokyo, 113–0033, Japan
E-mail: †takano-ryousei@aist.go.jp

Abstract　We propose a packet scheduling method, which precisely controls an inter-packet gap through the transmission of gap packet between adjacent packets. We have developed a software called PSPacer which realizes traffic pacing by using the proposed method. We evaluate PSPacer on TCP/IP communication over high bandwidth-delay product networks. The result shows that PSPacer reduces the number of packet losses, and improves the communication performance. In addition, when the bottleneck bandwidth is given, PSPacer achieves almost fully utilization of the bandwidth.
Key words　Packet Scheduling, Pacing, Gap Packet, BIC TCP

1. はじめに

帯域遅延積の大きなネットワークでは，TCP/IP 通信の実効帯域が低下することはよく知られている．この問題の原因はTCP のウィンドウ制御とバースト性制御のそれぞれにある．前者に対しては BIC TCP [7] などの改良が提案されており，本稿では後者に着目する．TCP はラウンドトリップ時間 (RTT) あたりのパケット送信量を輻輳ウィンドウサイズ (cwnd) によって制御し，RTT 期間中の送信タイミングを ACK クロッキングによって決める．しかし，ACK クロッキングが働かないスロースタート時などには，パケットが偏って送信されるバースト送信が発生する．帯域遅延積が大きいほどバースト性は大きくなり，ボトルネックルータにおいてパケットロスが発生する可能性が高くなる．そこで，パケット送信間隔が $RTT/cwnd$ と均一になるように，バースト送信を平滑化するためにペーシング [3] することが有効である．一般的に用いられるタイマ割込みベースのペーシングでは，その精密さはタイマ割込みの精度に依存する．1Gbps で 1 パケットの送信に要する時間は 12μs と通常のオペレーティングシステムで用いられているタイマ割込み間隔と比べて短い．また，タイマ割込み間隔を短くすると，

割込み処理の増加による負荷が増大するので，ソフトウェアによる実現は困難だった [4].

我々は，本来送信すべきパケットの間に，ギャップパケットを挿入することによって，パケット送信間隔を，精密かつ任意の大きさにスケジューリングできる方式を提案し，これを用いて精密なペーシングを実現する PSPacer を開発している．PSPacer は，規定の目標帯域を基にペーシングする静的ペーシングと，TCP プロトコルスタック内部に保持している cwnd と RTT から見積もった目標帯域を基にペーシングする動的ペーシングの 2 モードを提供し，ネットワークの性質によって使い分けることができる．PSPacer によるペーシングの効果を評価するために，高遅延環境を模擬したネットワークにおける TCP/IP 通信実験を行う．

以下，**2.** 節で提案方式の設計と，PSPacer の実装方法について述べる．**3.** 節で高遅延環境における TCP/IP 通信性能によって PSPacer を評価する．**4.** 節で関連研究について述べる．最後に **5.** 節でまとめを行う．

2. ソフトウェアによる精密ペーシング方式

2.1 ギャップパケット

パケット送信タイミングを正確に制御するために，タイマ割込みの代わりに，本来送信すべき実パケットの間にギャップパケットと呼ぶダミーのパケットを送信する．パケット送信に要する時間は，パケットサイズによって正確に決まる．したがって，必要なパケット送信間隔の大きさのギャップパケットを実パケット間に送信すれば，実パケットの送信タイミングを 1 バイトの送信に要する時間単位で精密に制御することができる．以下，パケット送信間隔をパケット間ギャップと呼ぶ．

ギャップパケットを用いてパケット間ギャップを制御することで，ペーシングを実現する例を図 1 に示す．ペーシングしない場合 (図 1 左) は，バースト送信が起きており，パケット間ギャップが偏っている．一方，実パケット間にギャップパケットを挿入することで，目標帯域にしたがってパケット間ギャップを平滑化できる (図 1 右)．このとき，パケット間ギャップは，目標帯域だけではなく，実パケットサイズにもあわせて，調整する必要がある．

ギャップパケットは，実際に PC の NIC(Network Interface Card) から送信されるパケットでなくてはならない．一方，ギャップパケットは，PC が接続されているスイッチやルータを越えて，ネットワークをずっと伝搬してはいけない．そこで，ギャップパケットとして，イーサネットのフロー制御規格である IEEE 802.3x で規定される PAUSE フレームを用いる(注1)．PAUSE フレームは，本来は対向する装置 (PC の NIC やスイッチ，ルータ) に送信を一定時間停止することを求めるために用いられる．PAUSE フレームは，対向する装置に PAUSE 要求が伝われば用済みなので，その装置の入力ポートで破棄され，それ以降に伝搬されない．PAUSE フレームは，通信の停止時間を指定するフィールドを持つが，ギャップパケットでは，停止時間を 0 に設定する．

(注1)：ギャップパケットはデータリンク層で実現しているので，ギャップフレームと呼ぶ方が正確であるが，本稿ではギャップパケットと呼ぶ．

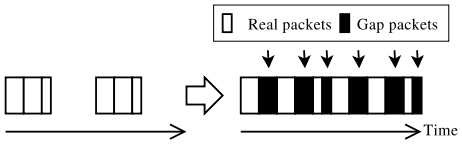

図 1　ギャップパケットを用いたパケット間ギャップ制御
Fig. 1　Inter packet gap control using gap packets.

2.2 目標帯域の見積り

トラフィックが固定的な帯域を持つネットワークパスに対して，ペーシングを適用する場合には，目標帯域を静的に設定することが有効である．これを静的ペーシングと呼ぶ．一方，通常の広域網では，トラフィックの変動により，利用可能帯域が変化するため，ネットワークの利用効率を高めるには，目標帯域に対してフィードバックを与える必要がある．TCP 通信の場合は，コネクションごとに $cwnd$ と RTT を保持しているので，これらを利用し，次式のように目標帯域を見積ることができる．

$$target\_rate = \frac{cwnd \times pkt\_size}{RTT} \tag{1}$$

これを動的ペーシングと呼ぶ．この場合，パケット間ギャップは $(cwnd \times pkt\_size)/RTT$ になるよう均一に平滑化される．

2.3 帯域制御

目標帯域から挿入すべきギャップパケットサイズを計算する方法を次に示す．1 パケットを目標帯域で送信する時間は，そのパケットに加えてパケット間ギャップ (ipg) 分のギャップパケットを，NIC の物理帯域 (max_rate) で送信する時間に等しい．

$$\frac{pkt\_size}{target\_rate} = \frac{pkt\_size + ipg}{max\_rate} \tag{2}$$

パケット間ギャップは，式 2 から導出できる．

$$ipg = \left(\frac{max\_rate}{target\_rate} - 1 \right) \times pkt\_size \tag{3}$$

さらに，ギャップパケットサイズ ($gappkt\_size$) は，パケット間ギャップから，ハードウェアギャップ (hw_gap) を減算した大きさになる．イーサネットの場合，ハードウェアギャップは IPG(Inter Packet Gap)，プリアンブル，フレームチェックサムの合計である．また，パケット間ギャップが MTU サイズより大きい場合は，複数のギャップパケットを送信する必要がある．

$$gappkt\_size = ipg - (hw\_gap \times \#pkts) \tag{4}$$

$\#pkts$ は実パケットとギャップパケットを合わせた数である．

$$\#pkts = ceiling\left(\frac{gappkt\_size}{MTU} \right) + 1 \tag{5}$$

2.4 PSPacer の実装

Linux オペレーティングシステムのトラフィック制御機構である iproute2 を利用して，ギャップパケットを用いたペーシング機構を実装した．これを PSPacer [1] と呼ぶ．iproute2 は，ネットワークトラフィックに対するクラス分け，優先度付け，帯域制御などの機能を提供するためのフレームワークであり，さ

まざまな QoS ポリシを実装する仕組みとして，Qdisc(Queuing Discipline) を提供する．PSPacer は，Qdisc モジュールとして実装したので，デバイスドライバや通信プロトコルに依存せず，アプリケーションの変更も不要である．

PSPacer はプロトコルスタックからの enqueue 要求を受けて，パケットをインタフェースキューにキューイングし，デバイスドライバからの dequeue 要求を受けて，目標帯域にしたがって実パケットもしくはギャップパケットを返す．静的ペーシングの場合，目標帯域とパケットサイズから，ギャップパケットをスケジューリングできる．動的ペーシングの場合，目標帯域を見積もるために，コネクションごとの cwnd と RTT を知る必要がある．これらは sk_buff 構造体から，プロトコルスタック内部のデータ構造をたどることで取得できる．

3. 評　　価

3.1 実験環境

PSPacer の評価を行うために，2 台の PC を用意し，ギガビットイーサネットで接続した．各 PC の諸元は，CPU が Intel Xeon/2.4GHz dual，メモリ 2GB (DDR266)，PCI-X 133MHz/64bit，そして NIC が Intel 82545EM である．PC 間のリンクには，高遅延環境の模擬と帯域測定のためにハードウェアネットワークテストベッド GtrcNET-1 [2] を配置した．GtrcNET-1 では，リンクの帯域を 500Mbps，RTT を 100ms に設定し，1MB のバッファを持つ Drop Tail ルータの挙動を模擬した．また，GtrcNET-1 は，通信挙動に影響をあたえることなく，高い精度で帯域を測定することができる．

使用した Linux カーネルは，2.6.14.2 であり，TCP 関連の統計情報を取得するために Web100 2.5.6 [9] を使用した．TCP 輻輳制御方式として，Linux 標準の BIC TCP [7] を用いた．BIC TCP は現在の利用可能帯域，つまり前回パケットロスしたときの $cwnd$ を W_{max} とすると，二分探索手法を用いて $cwnd$ を W_{max} に収束させる binary search increase と，線形に $cnwd$ を拡大する additive increase と呼ばれる 2 つのモードを使い分け，$cwnd$ を素早く回復させる．さらに，$cwnd$ が W_{max} に到達すると利用可能帯域を更新するため，スロースタートする．なお，初期スロースタート閾値 (initial_ssthresh) は標準で 100 パケットとなっており，実験環境のように帯域遅延積が大きな場合，通信開始直後の $cwnd$ の立ち上がりに時間がかかる．

使用した Linux カーネルには，大量の SACK (Selective ACK) パケットを受信した場合の再送キュー処理に問題があり，一時的に通信が中断する．そこで，Scalable TCP 実装に含まれている SACK-tag パッチ [8] を適用した．また，Linux を含め，一般的な TCP 実装では，インタフェースキューあふれを，パケットロスと同様に輻輳検出とみなす．この影響を避けるために，インタフェースキュー長を 10000 パケットに設定した．ソケットバッファサイズは 12.5MB とした．

3.2 高遅延環境における TCP/IP 通信性能

PSPacer を用いない noPSP，PSPacer の動的ペーシングを用いた PSPD，および帯域の上限を 500Mbps に設定し，見積り帯域がこれを超えないように静的ペーシングを用いた PSPS の 3 種類に対して Iperf を 1 分間実行したときの結果を比較した．スループットは noPSP が 200Mbps，PSPD が 315Mbps，PSPS が 433Mbps となり，ルータにおけるパケットロス数は noPSP が 1455，PSPD が 285，PSPS が 0 となった．この結果より，動的ペーシングを用いることで，パケットロスを削減し，通信性能を向上できることがわかる．さらに，ボトルネック帯域が既知の場合は，静的ペーシングを用いることで，ボトルネック帯域に近い，高いネットワーク利用効率が得られる．

GtrcNET-1 で測定した入力帯域と Web100 で測定した cwnd の時間的変化を図 2 に示す．帯域の測定間隔は 500μs であり，図 2 には 100ms 単位の平均帯域を示した．cwnd は Web100 を利用して，100ms 間隔で取得した．noPSP(図 2(a))，PSPD(図 2(b)) ともに最初のスロースタート時にパケットロスを起こし，binary search increase，additive increase を経て，$cwnd$ を回復させているが，それ以降の挙動が大きく異なる．PSPD は W_{max} が十分大きく，ほぼ 500Mbps の帯域を示しているが，noPSP は W_{max} が小さい状態で binary search increase モードに遷移しているので，500Mbps に達するのに 10 秒近くかかっている．また，noPSP，PSPD ともに，帯域が 500Mbps に達した後，帯域と $cwnd$ の増加が連動していないが，このときインタフェースキューのキューサイズが増加し，RTT が増加している．一方，PSPS(図 2(c)) の場合，帯域が 500Mbps を上回らないよう正確に制限されているので，パケットロスが発生せず，安定した通信が継続している．

noPSP と PSPD の違いを調べるため，パケットロス直後の挙動を比較した．測定結果を図 3 に示す．noPSP の場合，スロースタート時のパケット送信間隔には偏りが見られ，30.5 秒付近でパケットロスが発生する．その結果，W_{max} が縮小されるので，ネットワーク利用効率が低くなる．一方，PSPD の場合，多少の偏りはあるが，バースト性が抑えられるので，スロースタート中にパケットロスは発生しない．

また，図 2 から，ペーシングした場合，通信開始から帯域が 500Mbps に達するまでの時間が 1 秒ほど遅れていることがわかる．これは，パケットをバースト送信した方が ACK が返る時間が早くなるので，輻輳ウィンドウの拡大が早くなるからであり，輻輳ウィンドウが小さい場合にその差が顕著に表れる．

4. 関 連 研 究

高速ネットワークに対して，タイマ割込みを用いてペーシングを実現することは負荷が大きく困難であるが [4]，論文 [5] では，高精度タイマを使用し，スロースタート時のバーストを均一に分割するペーシング方式 Clustered Packet Spacing を提案し，日米間の高速ネットワークを使用した実験結果について述べている．一方，提案方式は，タイマ割込みの代わりにギャップパケットを用いることで精密なペーシングを実現している．さらに，タイマ割込み処理のオーバヘッドがないので，コネクションの全期間に対して適用できる．

論文 [3] は，ペーシングされた通信は，同期ロスの影響により，通常の通信よりも性能が劣化する問題をシミュレーション結果から示している．論文 [6] は，同期ロスの問題を回避するた

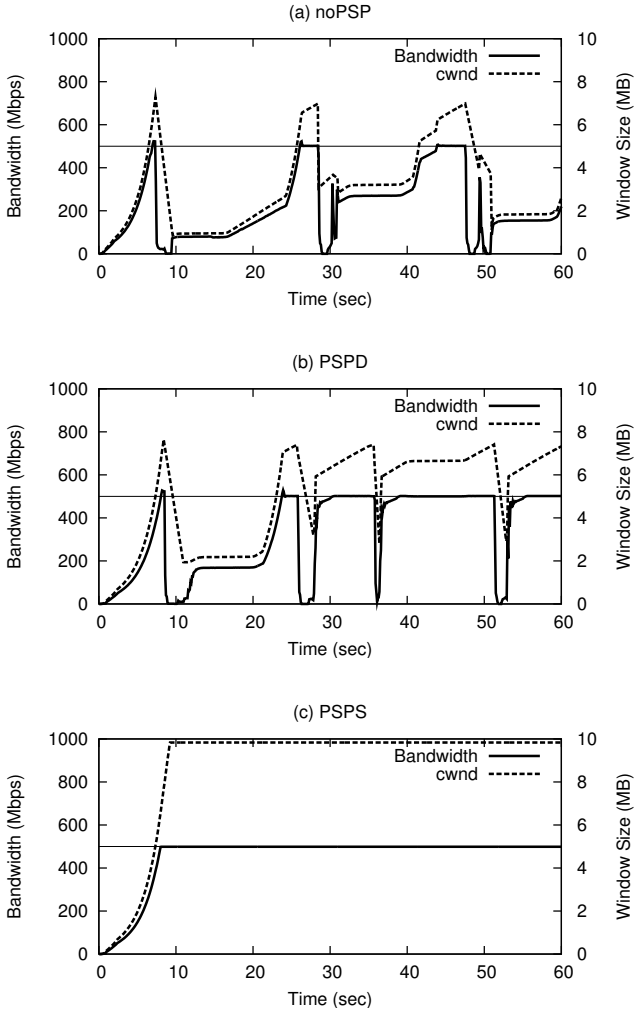

図 2　平均帯域と輻輳ウィンドウ (100ms)

Fig. 2　Average bandwidth and congestion window (100ms).

め，コネクションごとにバースト送信を残しつつ，ペーシングを行うゲートウェイを提案している．提案方式では，ギャップパケットの挿入位置をスケジューリングすることで，バースト性の大きさを制御することが可能である．今後，バースト送信できるパケット数を指定できるように機能拡張し，精密なペーシングと同期ロスのトレードオフを評価する予定である．

5. ま と め

本稿では，本来送信すべきパケットの間に，ギャップパケットを挿入することによって，パケット送信間隔を，精密かつ任意の大きさにスケジューリングできる方式を提案した．そして，提案方式を用いてペーシングを実現する PSPacer について述べ，高遅延環境を模擬したネットワークにおける TCP/IP 通信性能の改善について評価した．その結果，動的ペーシングを用いることで，スロースタート時のパケットロスを削減し，通信性能を向上できることを示した．さらに，ボトルネック帯域が既知の場合は，静的ペーシングを用いることで，ボトルネック帯域に近い，高いネットワーク利用効率が得られることを示した．本稿では，1 対 1 通信における実験を行ったが，今後，さ

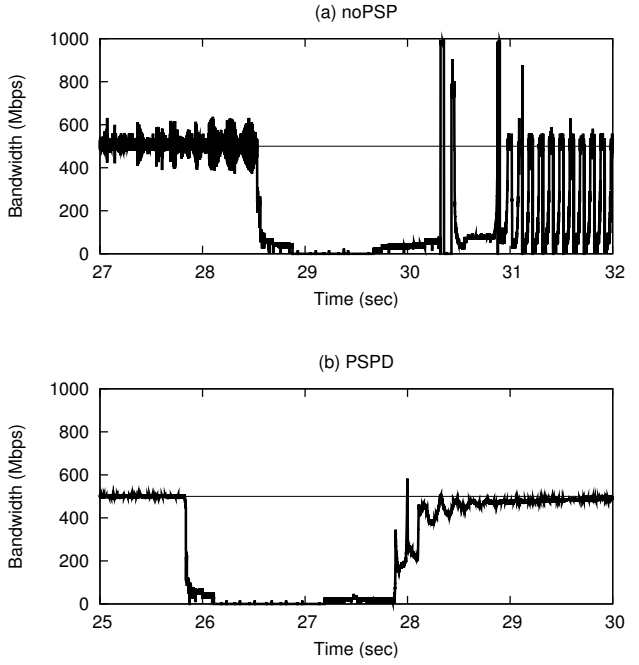

図 3　パケットロス直後後の挙動 (500μs)

Fig. 3　Behavior after packet losses (500μs).

らに実験規模を拡大し，同期ロスの影響や，パケットフロー間の公平性について評価を行う予定である．

PSPacer は GNU GPL ライセンスによるオープンソースソフトウェアとして公開しており，http://www.gridmpi.org/からダウンロード可能である．

謝　　辞

なお，本研究の一部は文部科学省「経済活性化のための重点技術開発プロジェクト」の一環として実施している超高速コンピュータ網形成プロジェクト（NAREGI：National Research Grid Initiative）による．

文　　献

[1] R. Takano et al., “Design and Evaluation of Precise Software Pacing Mechanisms for Fast Long-Distance Networks,” PFLDnet 2005, February 2005.
[2] Y. Kodama et al., “GNET-1: Gigabit Ethernet Network Testbed,” IEEE Cluster 2004, September 2004.
[3] A. Aggarwal et al., “Understanding the performance of TCP pacing,” IEEE INFOCOM, pp.1157-1165, March 2000.
[4] A. Antony et al., “Microscopic Examination of TCP flows over transatlantic Links,” iGrid2002 special issue, Future Generation Computer Systems, vol.19 issue 6, 2003.
[5] H. Kamezawa et al., “Inter-layer coordination for parallel TCP streams on Long Fat pipe Networks,” SC2004, November 2004.
[6] 菅原豊他，“細粒度パケット間隔制御の実装と評価,” 情報処理学会, SWoPP 2005, Auguest 2005.
[7] L. Xu et al., “Binary Increase Congestion Control for Fast Long-Distance Networks,” IEEE INFOCOM 2004, March 2004.
[8] T. Kelly’s SACK-tag patch, http://www-lce.eng.cam.ac.uk/ ctk21/code/.
[9] The Web100 Project, http://www.web100.org/.