ギャップパケットを用いたソフトウェアによる精密ペーシング方式

高野了成[†,††] 工藤知宏[†] 児玉祐悦[†]
松田元彦[†] 石川裕[†,†††] 岡崎史裕[†]

本論文では，専用ハードウェアを用いず，ソフトウェアによって精密なペーシングを実現する方式を提案する．提案方式は，実パケット間にギャップパケットを送信することで，精密な帯域制御とバースト性トラフィックの平滑化を実現する．ギャップパケットとして IEEE 802.3x で規定される PAUSE パケットを利用することで，ギガビットイーサネットを持つ一般的な PC を用いて，8 Kbps から 930 Mbps の範囲で，100 個の IP 通信のパケットフローごとに精密にペーシングできることを示す．さらに，往復遅延が 200 ms の高遅延環境において，TCP/IP 通信性能を，ほぼ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ように改善できることを示す．

Precise Software Pacing Method Using Gap Packets

Ryousei Takano,[†,††] Tomohiro Kudoh,[†] Yuetsu Kodama,[†]
Motohiko Matsuda,[†] Yutaka Ishikawa[†,†††]
and Fumihiro Okazaki[†]

In this paper, we propose a precise software pacing method, which achieves accurate network bandwidth control and smoothing bursty traffic without requiring special purpose hardware. The proposed method controls an inter-packet gap through the transmission of additional packet (gap packet) between adjacent packets. In order to realize a gap packet, the IEEE 802.3x PAUSE packet is employed. With the method, it is possible to provide bandwidth control and smoothing for each of 100 flows by use of a commodity PC. In the case of gigabit Ethernet, the transmission bandwidth can be set for a range from 8 Kbps to 930 Mbps for each of IP flows. Furthermore, it is shown that TCP/IP communication performance over high bandwidth-delay product networks is almost fully utilized by the method.

1. はじめに

近年，グリッド技術やストリーミング配信など，インターネット上に分散した計算機資源を利用した大規模計算，大規模データ通信のニーズが高まっている．このような環境でネットワークの利用効率を上げ，かつ安定した通信性能を得るためには，通信が集約されるボトルネックリンクにおいて，各通信の合計帯域がそのリンクの利用可能帯域を超えないようにデータ送信量を制御する必要がある．しかし，一般的に利用されるトークンバケット方式の帯域制御では，平均帯域は保証されるが，短時間にはバースト的にパケットが送信される．そして，このようなバースト性トラフィックが重なると，ピーク帯域が利用可能なリンク帯域を超えてしまう．超過分のパケットは，ルータやスイッチのバッファに蓄えられるが，そのバッファに蓄えきれないパケットは破棄され，パケットロスが発生する．グリッド環境のように，帯域遅延積が大きなネットワークで TCP/IP 通信を行う場合，パケットロスの影響を大きく受け，通信性能が著しく低下する可能性がある[1)]．また，ストリーミング配信において，パケットロスは受信映像品質の劣化を意味する．

バースト性トラフィックを抑制し，バッファあふれを防ぐには，目標帯域に従ってパケットの送信間隔が均等になるように，送信タイミングを正確に制御する必要がある．このような制御をペーシングと呼び，多くの研究[10)~12)]が行われている．我々は，ペーシングの効果に着目し，SC2003 Bandwidth Challenge において，日米間で大容量ファイル転送実験を行った際，ボ

† 産業技術総合研究所グリッド研究センター
Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology (AIST)
†† 株式会社アックス
AXE, Inc.
††† 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

トルネックリンクの手前で，ハードウェアネットワークテストベッド GtrcNET-1[3] を用いてペーシングした．この結果，各通信の帯域変動を最小限に抑えることができ，パケットロスのない，非常に安定した通信を達成することができた[4]．しかし，このような専用ハードウェアを利用できない一般の環境でペーシングを用いることは困難である．

本論文では，特別なハードウェアを用いることなく，ソフトウェアによる精密なペーシングを実現する方式について提案する．本方式では，本来送信すべきパケットの間に，任意長の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に調整する．そして，イーサネット上でギャップパケットを実現するために，IEEE802.3x で規定される PAUSE パケットを利用する．また，提案手法を用いたペーシングの効果を，往復遅延が 200 ms の高遅延環境における TCP/IP 通信性能によって評価する．

以下，2 章で既存の帯域制御機構の問題と，ソフトウェアによるペーシングを実現するうえでの問題点について述べる．続いて，3 章で提案方式の設計と，Linux オペレーティングシステム上への実装方法について述べる．4 章で提案方式が精密なペーシングの定義を満たすことを示し，さらに 5 章で高遅延環境における TCP/IP 通信性能への効果を評価する．6 章で得られた結果に対する議論を行う．7 章で関連研究について述べる．最後に 8 章でまとめを行う．

2. 背　　景

ペーシングは，目標帯域に基づきパケット送信間隔を均一に平滑化する手法である．たとえば，ギガビットイーサネット（GbE）では，1,500 バイトのパケットを送信するのに要する時間は 12 μs である．ここで帯域を 500 Mbps にペーシングするには，**図 1** (a) に示すように，パケット送信間隔を 24 μs に制御する必要がある．

これに対し，従来から帯域制御を実現するために，一般的に用いられているトークンバケット方式では，目標帯域を基に一定タイマ間隔でトークンをバケットに追加し，バケットにたまったトークン分のパケットを送信する．この方式では，平均帯域の正確さは保証するが，バケットサイズ分のバースト送信を許容するので，バースト送信する ON 期間と，無通信の OFF 期間を繰り返す ON-OFF トラフィックが発生する．バケットサイズを小さく保ったまま，精密に帯域制御するには，トークンの追加タイミングを制御するタイマ間隔を十分短くする必要がある．

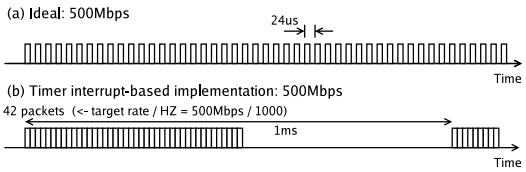


図 1 ペーシング
Fig. 1 Pacing.

Linux などのオペレーティングシステムでは，タイマ源として 1〜10 ms 程度の間隔のタイマ割込みが用いられるが，1 パケットの送信に要する時間はタイマ割込み間隔と比べて短く，ソフトウェアで送信のタイミングを正確に制御することは困難である．パケット送信間隔をタイマ割込みによって制御する場合，目標帯域/HZ のバースト送信は避けられない．なお，HZ はタイマ割込み間隔の逆数である．つまり，目標帯域が 500 Mbps で，タイマ割込み間隔が 1 ms の場合，図 1 (b) で示すように，42 パケット（62.5 KB）分のバースト送信が発生する．より精密なパケット間隔制御を行うために，タイマ割込み間隔を短くすると，割込み処理の増加により CPU 負荷が増加するので，計算性能や通信性能に悪影響を与える[13]．また，割込みレイテンシによる処理のばらつきも問題となる．したがって，ソフトウェアによるペーシングを実現することは困難である．

3. ソフトウェアによる精密ペーシング方式

3.1 目　　的

提案方式の目的は，次の 2 つである．

(1) 専用ハードウェアを用いず，一般的な PC 上で，ソフトウェアによるペーシングを実現する．
(2) パケットフローを IP アドレスやポート番号の組合せに基づいてクラス分けし，クラスごとに正確な帯域制御を実現する．

以降，これらの目的を実現するために提案するギャップパケットと，イーサネット上での実現方法について述べる．

3.2 ギャップパケット

ペーシングの実現には，送信側の計算機がパケット送信タイミングを正確に制御できることが必要である．そこで，タイマ割込みの代わりに，ペーシング対象の実パケット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ことによりパケット送信タイミングを制御する．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．

図 2 に，ギャップパケットを用いたパケット間ギャップ制御を示す．ペーシングしない場合（図 2 上）は，バースト送信が発生しており，パケット間ギャップが偏っている．提案方式では，実パケット間にギャップパケットを挿入することで，目標帯域に従ってパケット間ギャップを平滑化する（図 2 下）．さらに，正確な帯域制御を実現するために，実パケットサイズに比例して，パケット間ギャップを調整する．

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超えて，ネットワークをずっと伝搬してはいけない．また，ギャップパケットとして，存在しない宛先へのパケットを送信した場合，フラッディングによるストームが発生する問題がある．

そこで，我々は，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 パケットを用いることを提案する☆．PAUSE パケットは，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 パケットは，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 パケットは，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．ただし，提案方式とフロー制御を同時に利用することはできないので，PC から対向機器方向のフロー制御を無効にする必要がある．通常，PC の受信処理は十分高速で PAUSE パケットを用いてフロー制御することは稀である．また，IEEE802.3x に対応した一般的なスイッチでは，フロー制御無効が出荷時設定であり，本方式を用いるために特別な設定を行う必要はない．

3.3 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．まず，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．

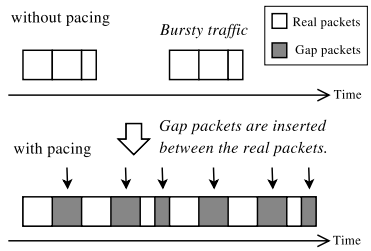


図 2 ギャップパケットを用いたパケット間ギャップ制御
Fig. 2 Inter packet gap control using gap packets.

$$\frac{pkt\_size}{target\_rate} = \frac{pkt\_size + ipg}{max\_rate} \quad (1)$$

パケット間ギャップは，式 (1) から導出できる．

$$ipg = \left(\frac{max\_rate}{target\_rate} - 1\right) \times pkt\_size \quad (2)$$

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）を減算した値になる．イーサネットの場合，ハードウェアギャップは IFG（Inter Frame Gap），プリアンブル，フレームチェックサムの合計である．$\#pkts$ は実パケットとギャップパケットを合わせたパケット数である．

$$gappkt\_size = ipg - (hw\_gap \times \#pkts) \quad (3)$$

ギャップパケットの最小サイズは，イーサネットの制限より 64 バイトである．GbE を用いた場合，このときの帯域は，935.2 Mbps となる．一方，パケット間ギャップが MTU サイズを超える場合は，実パケット間に複数のギャップパケットを送信する．現在は，目標帯域の最小値を 8 Kbps に設定している．このときのパケット間ギャップは，MTU が 1,500 バイトで約 190 MB になる．

3.4 スケジューリングアルゴリズム

パケットフローごとに帯域制御する必要もあると考えられる．そこで，パケットフローを IP アドレスやポート番号の組合せに基づきクラス分けし，クラス単位にペーシングを行う．クラス単体でのパケット間ギャップは，式 (2) より計算できるが，複数のクラスが存在する場合，パケット間ギャップを調整する必要がある．そこで，他クラスの実パケットもギャップパケットと見なして，パケット間ギャップを調整する．

クラスには，規定の目標帯域に従ってペーシングするペーシングクラスと，ペーシング非対象のノーマルクラスの 2 種類が存在する．各クラスは，ペーシングクラス，ノーマルクラスごとにリストで管理され，それぞれ送信キューを持つ．パケット送信の優先度は，

☆ データリンク層ではフレームと呼ぶ方が正確であるが，本論文ではパケットに記述を統一する．
☆☆ 一般には XON/XOFF の二値による制御として実装されることが多く，停止時間 0 に設定されたパケットを XON と扱う．

ノーマルクラスよりもペーシングクラスが高く，さらに目標帯域が高いペーシングクラスほど優先度が高い．したがって，ペーシングクラス，ノーマルクラスの順で実パケットをスケジューリングし，さらに無通信期間が存在する場合には，ギャップパケットをスケジューリングする．また，適切なパケット間ギャップと目標帯域を達成するためには，ペーシングクラスの目標帯域の合計が，NIC の最大転送帯域以下である必要がある．

ギャップパケットは，パケット間ギャップを 1 バイトの送信に要する時間単位で精密に制御できるので，実時刻ではなく，送信バイト数を仮想時刻として用いる．まず，NIC から送信される総バイト数をグローバルクロックとして保持する．グローバルクロックは現在時刻を意味する．各ペーシングクラスは，そのクラスの目標帯域と，前回の送信時刻とパケットサイズから求めた，次パケットの送信予定時刻を保持する．これをクラスクロックと呼ぶ．ノーマルクラスはパケット送信タイミングの制御が不要なので，クラスクロックを持たない．

パケットスケジューリングの流れを次に示す．

(1) 送信対象パケットの検索，送信

ペーシングクラス (1a)，ノーマルクラス (1b)，ギャップパケット (1c) の順で送信可能なパケットを検索し，いずれかのパケットを送信した後，(2) へ進む．

(1a) ペーシングクラスを優先度順に検索し，クラスクロックがグローバルクロックより小さい場合，そのクラスの送信キューの先頭パケットを送信する．ただし，送信キューにパケットがキューイングされていない場合は，通信が停止していると判断し，通信停止フラグをセットする．

(1b) (1a) において，ペーシングクラスに送信可能なパケットが存在しない場合は，ノーマルクラスをラウンドロビンに従って検索する．

(1c) すべてのノーマルクラスの送信キューが空の場合は，無通信期間なので，ギャップパケットを送信する．この際，パケット間ギャップは，グローバルクロックと各クラスクロックの差分の最小値に設定する．このパケット間ギャップを基に，式 (3) よりギャップパケットサイズを求める．

(2) グローバルクロックの更新

(2a) グローバルクロックに (1) で送信したパケットサイズを加算する．

(3) クラスクロックの更新

(3a) (1a) において，ペーシングクラスのパケットを送信した場合，クラスクロックに「パケットサイズ + パケット間ギャップ」を加算する．パケット間ギャップは，式 (2) より求める．

(3b) 送信停止フラグがセットされている場合は，送信再開時におけるバースト送信の発生を防ぐためクラスクロックをグローバルクロックに合わせる．

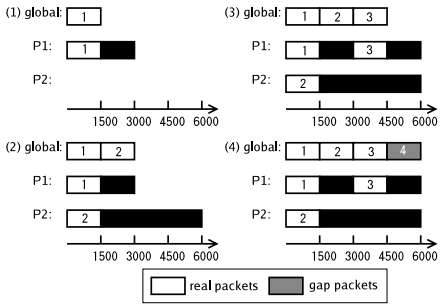


図 3 ギャップパケットを用いたパケットスケジューリング（global はグローバルクロック，P1，P2 はクラスクロックを示す．P1，P2 の目標帯域は，それぞれ 500 Mbps，250 Mbps である）

Fig. 3 Packet scheduling using gap packets: 'global' shows a global clock. 'P1' and 'P2' show class clocks. Target bandwidths of 'P1' and 'P2' are 500 Mbps and 250 Mbps, respectively.

図 3 に，目標帯域がそれぞれ 500 Mbps と 250 Mbps のペーシングクラス P1，P2 が存在する場合のスケジューリングを示す．説明を簡単にするため，パケットサイズは 1,500 バイト均一とした．式 (2) より，P1，P2 のパケット間ギャップはそれぞれ 1,500 バイト，4,500 バイトになる．最初に目標帯域の高い P1 クラスからパケットを送信すると，(1) グローバルクロックは 1,500 バイト，P1 のクラスクロックは 3,000 バイトになる．続いて，P1 のクラスクロックはグローバルクロックより進んでおり，送信可能時刻に達していないので，P2 からパケットを送信し，(2) グローバルクロックは 3,000 バイト，P2 のクラスクロックは 6,000 バイトになる．次は，P1 が送信可能になるので，(3) グローバルクロックは 4,500 バイト，P1 のクラスクロックは 6,000 バイトになる．この時点で，P1，P2 のクラスクロックはともに，グローバルクロックより進んでおり，パケットは送信できない．そこで，送信可能時刻までの 1,500 バイト分のギャップパケットを送信し，(4) グローバルクロックは 6,000 バイトになる．上記のようにスケジューリングした結果，P1，P2，ギャップパケットが帯域に占める割合は，2：1：1 になる．

3.5 実　　装

提案方式を，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，実装し

た．これを PSPacer[1),2)] 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（Quality of Service）ポリシを実装する仕組みとして，Qdisc（Queuing Discipline）を提供する．Qdisc は，プロトコルスタックとデバイスドライバに対して，インタフェースキューの内部実装を隠蔽し，キューイング操作関数（enqueue, dequeue 関数など）を提供する．さらに，Qdisc は，階層的な帯域制御を実現するために，インタフェースごとにツリー構造の親子関係を持つことができる．

PSPacer は，**図 4** に示すように Qdisc モジュールとして実装されており，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．また，ローダブルカーネルモジュールであり，導入のためにカーネルを再構築する必要はない．PSPacer は，プロトコルスタックから enqueue 関数が呼ばれると，パケットをフィルタの設定に従ってクラス分けし，クラスごとに対応付けられた送信キュー（サブ Qdisc）にキューイングする．そして，デバイスドライバから dequeue 関数が呼ばれると，3.4 節に示したアルゴリズムに従ってパケットを選択し，送信する．送信キューが空の場合，dequeue 関数は呼ばれないので，通信停止時に無駄なギャップパケットが送信されることはない．

ギャップパケットの前半部は PAUSE パケットフォーマットであり，後半部はパケットサイズ調整用のパディング領域である．PSPacer では，メモリ使用量を抑えるために，初期化時に MTU サイズのギャップパケットを生成し，dequeue 時には，そのパケットを参照するソケットバッファ構造体（sk_buff）だけを生成，操作することで，異なるサイズのパケットを送信する☆．

3.6 使 用 例

iproute2 を利用することで，他の Qdisc モジュールと同様に tc コマンドからクラスの追加，削除，目標帯域の設定や，統計情報の取得が可能である．また，既存クラス分けフィルタの利用，PSPacer と他の Qdisc モジュールの連携も可能である．

目標帯域が 500 Mbps，250 Mbps のペーシングクラスと，ノーマルクラスを追加する例を，次に示す．なお，Qdisc はハンドル番号，クラスはクラス ID によって識別できる．まず，ルート Qdisc を作成する．

```
# tc qdisc add dev eth0 root handle 1: \
  psp default 3
```

☆ skb_clone 関数で sk_buff のクローンを生成し，skb_trim 関数でパケットサイズを変更する．

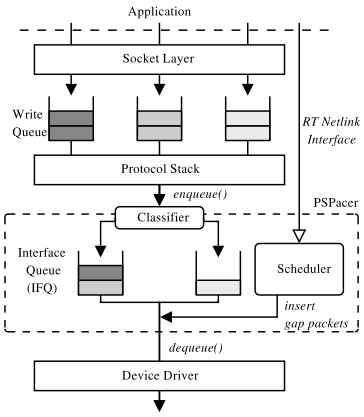


図 4 PSPacer の実装
Fig. 4 Implementation of PSPacer.

次に，ルート Qdisc に対して，クラスを追加する．ペーシングクラスの場合は，目標帯域を指定する．

```
# tc class add dev eth0 parent 1: \
  classid 1:1 psp rate 500mbit
# tc class add dev eth0 parent 1: \
  classid 1:2 psp rate 250mbit
# tc class add dev eth0 parent 1: \
  classid 1:3 psp mode normal
```

続いて，各クラスに対してサブ Qdisc を割り当てる．ここでは単純な FIFO を提供する PFIFO を使用した．

```
# tc qdisc add dev eth0 parent 1:1 \
  handle 10: pfifo
# tc qdisc add dev eth0 parent 1:2 \
  handle 20: pfifo
# tc qdisc add dev eth0 parent 1:3 \
  handle 30: pfifo
```

最後に，パケットフローを各クラスに振り分けるためのフィルタを追加する．ここでは宛先 IP アドレスをフィルタに用いており，これらの条件に合わないパケットはデフォルトクラスに振り分けられる．

```
# tc filter add dev eth0 parent 1: \
  protocol ip pref 1 u32 match ip \
  dst 192.168.2.0/24 classid 1:1
# tc filter add dev eth0 parent 1: \
  protocol ip pref 1 u32 match ip \
  dst 192.168.3.0/24 classid 1:2
```

4. PSPacer の評価

4.1 バースト性の定義

ATM の時代から，定量的なバースト性の定義について提案されているが[9)]，多くは統計的手法を用いており，実際の通信性能に関係するパケットロスやキューイング遅延を直接反映していない．ここでは，1 パケットフローのバースト性を定量的に定義するために，そのパケットフローの平均帯域（BW_{avg}）を持つ仮想的

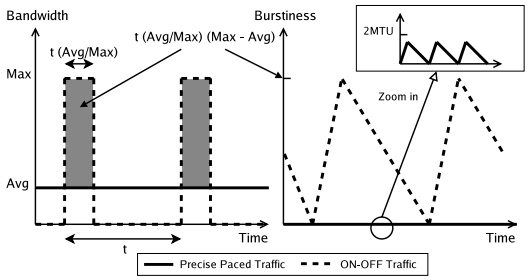


図 5 バースト性
Fig. 5 Burstiness.

なボトルネックを仮定する．このボトルネックを通過するパケットフローの帯域が BW_{avg} を超えた場合には，超過分のパケットは入力バッファに保持される．この入力バッファにキューイングされたデータ量，すなわちキューサイズ（Q）をバースト性と定義する．また，パケットフローにおける Q の最大値を最大バースト性と呼ぶ．バースト性が小さいほど，ルータに対する負荷は小さく，複数のパケットフローを規定の帯域を超えないように集約する場合も，個々のバースト性を小さくした方が，パケットロスの可能性が低いと考える．

パケットが完全に均一な間隔で送信される場合，すべてのパケットは，後続のパケットが到着すると同時に，送信が完了する．したがって，バースト性はそのパケットサイズ以下，つまり 1 MTU 以下になる．ある計算機から送信されるパケットフローが 1 つの場合は，バースト性が 1 MTU 以下になるように，パケットをスケジューリングできる．しかし，あるパケットの送信中に，これを中断して別のパケットを送信することはできないので，複数のパケットフローをスケジューリングする場合，送信時間が競合し，1 パケット分スケジューリングがずれる可能性がある．したがって，最大バースト性が，たかだか 2 MTU になることを精密なペーシングと定義する．

図 5 は，平均帯域は同一だが，バースト性が異なる 2 種類の通信における通信帯域（図 5 左）とバースト性（図 5 右）である．精密なペーシングの場合，つねに通信帯域は平均帯域と一致している．一方，ON-OFF トラフィックの場合，バースト性は ON 期間で上昇し，OFF 期間で下降する．1 組の ON-OFF 期間の時間を t とすると，ON 期間の時間は $t\times(Avg/Max)$ となるので，最大バースト性は，$t\times(Avg/Max)\times(Max-Avg)$ になる．一方，精密なペーシングの場合，バースト性は図右上の拡大図に示すように，2 MTU 以下になる．

表 1 実験環境の諸元
Table 1 Host PC specifications.

	送信側クラスタ（16 台）	受信側クラスタ（16 台）
CPU	Intel Xeon 2.8 GHz dual	Intel Xeon 2.8 GHz dual
Chipset	Intel E7501	ServerWorks GC-LE
Memory	1 GB	1 GB
NIC	Intel 82546EB	Intel 82546EB
I/O Bus	PCI-X 133 MHz/64 bit	PCI-X 133 MHz/64 bit
OS	FedoraCore 3（kernel 2.6.11.12）	
NIC Driver	e1000 5.6.10.1-k2-NAPI	
TCP	BIC TCP	

4.2 実験環境

前節で定義したバースト性を用いて PSPacer を評価するために，2 つの 16 台構成の PC クラスタを 1 本のリンクで接続した実験環境を用意した．クラスタ内はそれぞれ単一のノンブロッキングスイッチ（Catalyst 2970）で接続されており，クラスタ間はレイヤ 3 スイッチ（Catalyst 6506）で接続されている．C2970 と C6506 間のリンクには，ハードウェアネットワークテストベッド GtrcNET-1[3] を配置した．GtrcNET-1 はネットワーク環境を模擬する他に，通信挙動に影響をあたえることなく，帯域測定やパケットキャプチャが可能である．

評価に用いた計算機の諸元を**表 1** に示す．Linux カーネルは，2.6.11.12 であり，TCP 輻輳制御アルゴリズムとして BIC TCP[16] を用いた．BIC TCP は Linux ではデフォルトで選択されるアルゴリズムであり，帯域遅延積が大きなネットワークにおいて Reno よりも性能が改善されている．なお，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[17] 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見なす[5]．この影響を避けるために，インタフェースキュー長を 10,000 パケットに設定した．これは，今回の実験に対して十分な長さである．PSPacer との比較として，Linux カーネルに含まれるトークンバケット方式の実装である TBF（Token Bucket Filter）を使用した．ベンチマークソフトウェアとして，UDP および TCP のバルク通信性能を測定する Iperf を用いた．

4.3 精密な帯域制御

ギャップパケットを目標帯域に従って挿入することで，期待どおりの通信性能が得られるかを評価した．実験には，TCP 輻輳制御による影響を避けるため，UDP を用いた．帯域測定には GtrcNET-1 を用いて

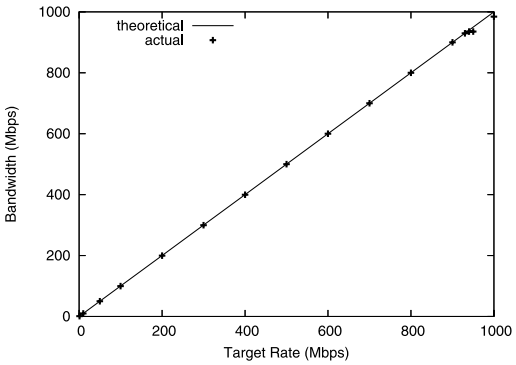


図 6 目標帯域と実効帯域の関係
Fig. 6 Effective bandwidth while varying target bandwidth.

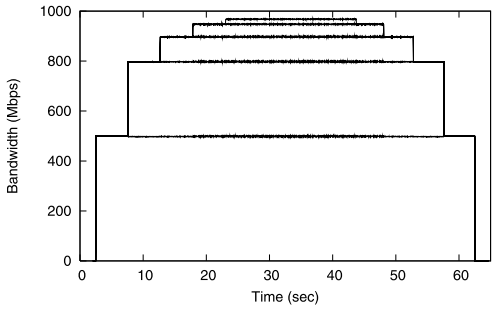


図 7 複数目標帯域の場合の帯域制御（目標帯域：500，300，100，50，20 Mbps）
Fig. 7 Bandwidth control in the case of multiple target rates (Target Bandwidth: 500, 300, 100, 50, 20 Mbps).

おり，以降，断りのない限り，データリンク層の帯域を示す．また，目標帯域が 1 種類のペーシングクラスを用いる場合を単一目標帯域，複数の種類を用いる場合を複数目標帯域と呼ぶ．各通信は，パケットフローごとにクラス分けする．

図 6 は，目標帯域を変えながら，Iperf を実行したときの実効帯域である．PSPacer が動作を保証する 8 Kbps から 930 Mbps の範囲において，目標帯域と実効帯域は一致する．なお，目標帯域が 1 Gbps，つまりギャップパケットを挿入しない場合の送信帯域は 984 Mbps である．この結果より，ギャップパケットによって目標どおりの通信性能が得られることが分かる．

図 7 は，5 つのペーシングクラスの目標帯域を 500，300，100，50，20 Mbps に設定し，Iperf を実行したときの結果である．測定間隔は 10 ms である．パケットフロー数が増減したときでも，各クラスの帯域に乱れは観測されていない．これより，複数目標帯域の場合でも，期待どおりの通信性能が得られることが分かる．

表 2 単一目標帯域，1 対 1 通信：1 パケットあたりの帯域と最大バースト性（帯域の単位は bps）
Table 2 Single target rate, 1-to-1 communication: Bandwidth per 1 packet and max burstiness (The unit of bandwidth is bps).

目標帯域	最低帯域	最大帯域	平均帯域	最大バースト性
8 K	7.96 K	7.96 K	7.96 K	1 MTU
10 M	9.95 M	9.95 M	9.95 M	1 MTU
500 M	495 M	500 M	498 M	2 MTU
930 M	918 M	931 M	926 M	2 MTU

4.4 バースト性

各パケットの送信時刻を t_i，パケットサイズを pkt_size_i とすると，n パケットあたりの帯域は，$(\sum_{k=i}^{i+n} pkt\_size_i)/(t_{i+n} - t_i)$ となる．n パケットあたりの最大帯域が NIC の物理帯域と等しい場合は，ある連続した $n+1$ パケットのバースト送信が発生したことを意味する．パケット間ギャップが平滑化され，バースト性が最小限に抑えられていることを確認するために，n パケットあたりの送信帯域，および 4.1 節で定義した最大バースト性を求めた．

評価のため，通信開始から約 25 万パケットを GtrcNET-1 を用いてキャプチャした．取得したパケットには，$2^{24}Hz = 59.6$ ns の時間分解能を持つタイムスタンプが付加されている．このタイムスタンプとパケットサイズを基に n パケットあたりの帯域を求めるが，その誤差は平均帯域の増加に従い顕著になる．たとえば，平均帯域が 930 Mbps のとき，1 パケットあたりの帯域の誤差は約 2 Mbps になる．同じパケットを用いて，パケットフローごとの最大バースト性を求めた．なお，評価プログラムの制限のため，バースト性はバイト単位ではなく，パケット単位で切り上げている．

4.4.1 単一目標帯域，1 対 1 通信

目標帯域を変えながら，1 対 1 通信を行ったときの結果を，**表 2** 示す．平均帯域は目標帯域のほぼ 99.5%を示している．目標帯域が 930 Mbps のときに，最大帯域が 931 Mbps となっているが，これはタイムスタンプの精度による誤差範囲内である．したがって，最大帯域も目標帯域を超えていない．また，すべての場合で，バースト性がたかだか 2 MTU 以下であり，精密なペーシングの条件を満たしている．目標帯域が 8 Kbps，10 Mbps のときの最大バースト性は 1 MTU であるのに対して，500 Mbps，930 Mbps のときは 2 MTU である．これもタイムスタンプの精度が原因だと考えられる．

表 3 単一目標帯域，1 対多通信：1 パケットあたりの帯域と最大バースト性（帯域の単位は bps）
Table 3 Single target rate, 1-to-many communication: Bandwidth per 1 packet and max burstiness (The unit of bandwidth is bps).

目標帯域	最低帯域	最大帯域	平均帯域	最大バースト性
10 M	9.89 M	9.92 M	9.91 M	1 MTU

表 4 複数目標帯域：n パケットあたりの帯域と最大バースト性（帯域の単位は bps）
Table 4 Multiple target rate: Bandwidth per n packets and burstiness (The unit of bandwidth is bps).

n	目標帯域	最低帯域	最大帯域	平均帯域	最大バースト性
1	20 M	14.4 M	30.3 M	20.5 M	2 MTU
	50 M	30.0 M	98.4 M	33.1 M	2 MTU
	100 M	61.9 M	246 M	103 M	2 MTU
	300 M	168 M	494 M	332 M	2 MTU
	500 M	345 M	990 M	506 M	2 MTU
2	20 M	16.6 M	24.1 M	19.9 M	2 MTU
	50 M	39.8 M	74.2 M	49.9 M	2 MTU
	100 M	68.0 M	143 M	100 M	2 MTU
	300 M	205 M	493 M	304 M	2 MTU
	500 M	405 M	658 M	500 M	2 MTU

4.4.2 単一目標帯域，1 対多通信

1 台の送信ホストから，16 台の受信ホストに対して，それぞれ 6 パケットフロー，合計 96 パケットフローを送信した．この際，パケットフローを宛先 IP アドレスとポート番号を基にクラス分けし，それぞれ 10 Mbps に帯域制御した．結果を **表 3** に示す．平均帯域は 1 対 1 通信よりも 40 Kbps 低いが，約 100 にクラス分けした通信時でも，精密なペーシングが実現できていることが分かる．

4.4.3 複数目標帯域

図 7 の 25 から 45 秒区間は，5 種類のペーシングクラスが同時に通信している．この場合の n パケットあたりの帯域と最大バースト性を **表 4** に示す．複数クラスが競合し，均等なスケジューリングはできないので，n パケットあたりの帯域は，単一クラスの場合ほど正確ではない．特に，目標帯域が 500 Mbps の最大帯域が，$n = 1$ のときに 990 Mbps であることから分かるように，2 パケットがバースト送信されている．しかし，最大バースト性はすべての場合において 2 MTU であり，精密なペーシングの条件を満たしている．

5. ペーシングによる TCP/IP 通信性能の改善

前章で示したように，PSPacer はソフトウェアにより精密なペーシングを実現する．本章では，ペーシングによる TCP/IP 性能通信の改善について述べる．

表 5 ルータにおける FIFO サイズと Iperf スループット，パケットロス数の関係（帯域の単位は Mbps）
Table 5 Iperf throughput and the number of packet losses on a router (The unit of bandwidth is Mbps).

	制限なし		TBF		PSPacer	
FIFO	帯域	ロス数	帯域	ロス数	帯域	ロス数
16 KB	29.4	219	26.9	131	474	0
64 KB	210	257	191	402	474	0
256 KB	223	394	379	261	473	0
1024 KB	256	1196	419	12	474	0
4096 KB	459	1283	471	0	474	0

5.1 高遅延環境における TCP/IP 通信性能

TCP/IP 通信では，パケットロスを検出すると，送信側が輻輳ウィンドウを半減させることで，送信量を減らすが，通信遅延が大きくなると，輻輳ウィンドウの回復に時間がかかるため，ネットワーク利用効率が低下することが知られている[18]．

このような高遅延環境におけるペーシングの効果を評価するため，GtrcNET-1 を用いて高遅延環境を模擬し，1 対 1 通信および 2 対 2 通信の 2 種類の実験を行った．ボトルネックリンクの帯域は 500 Mbps，RTT（Round Trip Time）は 200 ms とした．ルータのバッファリング方式として，Drop Tail 方式を模擬したので，通信のバースト性がルータの FIFO サイズを超えると，パケットロスが発生する．ソケットバッファサイズは実験環境において十分な大きさである 25 MB とした．

5.2 1 対 1 通信

帯域制限なし，TBF（Token Bucket Filtering），PSPacer ごとに，ルータの FIFO サイズを 16 KB から 4 MB に変えながら，5 分間の 1 対 1 通信を行った結果を **表 5** に示す．TBF と PSPacer の目標帯域は，ボトルネックリンク帯域である 500 Mbps に設定した．表中の帯域は Iperf が出力したスループットであり，TCP/IP ヘッダのオーバヘッドを含むので，GtrcNET-1 で測定される帯域よりも低い．パケットロス数はルータにおける Drop Tail 数である．

PSPacer の場合，FIFO サイズにかかわらず，パケットロスが発生しない．一方，制限なし，および TBF の場合は FIFO サイズが小さいほど，帯域も低い．制限なしの場合，FIFO サイズが小さいほど，輻輳ウィンドウが小さいときにパケットロスが発生し，輻輳ウィンドウの回復に時間がかかるので，帯域利用率が低い．TBF の場合，FIFO サイズが 4 MB あれば，スロースタート時のバースト性をバッファリングできるので，

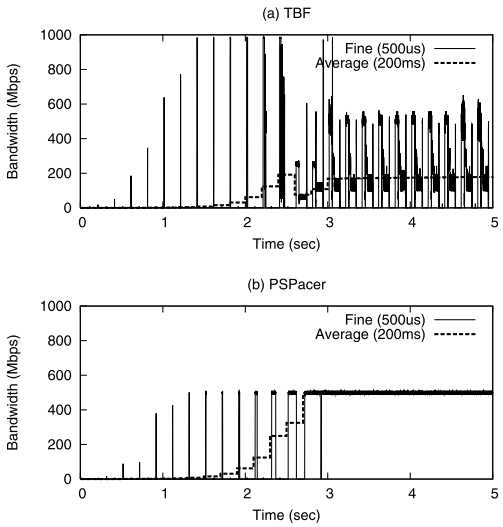


図 8 1 対 1 通信時におけるスロースタートの挙動（ボトルネック帯域 500 Mbps, RTT 200 ms, FIFO サイズ 1 MB）
Fig. 8 Behavior of slow start phase on 1-to-1 communication (Bottleneck bandwidth 500 Mbps, RTT 200 ms, FIFO size 1 MB).

パケットロスは発生しないが，1 MB よりも小さくなると，パケットロスが起きる．1 MB，256 KB の場合は，スロースタート以降の輻輳回避フェーズにおいては，パケットロスが発生せず，帯域は安定する．64 KB 以下の場合は，輻輳回避フェーズにおいてもパケットロスが発生し，輻輳ウィンドウが十分大きくならないので，帯域利用率が低いままである．

これらのパケットロスの原因として，スロースタート時のバースト性と 2 章で述べたタイマ割込み駆動が原因のバースト性（この場合，62.5 KB）の 2 つが考えられる．スロースタート時は，ACK クロッキングが働かず，バースト送信が発生するので，パケットロスが発生する可能性が高い[1),2)]．スロースタート時の挙動を詳細に比較するために，500 μs 間隔で帯域を測定した．FIFO が 1 MB，TBF と PSPacer の場合の結果を **図 8** に示す．実線が 500 μs，破線が 200 ms 平均の帯域である．TBF の場合（図 8 (a)），RTT である 200 ms 周期の ON-OFF トラフィックになっている．そして，2.5 秒近辺でバースト性がルータの FIFO サイズである 1 MB を超え，パケットロスが発生している．このように，TBF は，スロースタート時にはバケットサイズ分のバースト性トラフィックが発生する．一方，PSPacer ありの場合，ボトルネック帯域を超えないようにトラフィックは平滑化されるので，パケットロスは発生せず，安定した通信が継続する．FIFO サイズが 16 KB と小さい場合でも，同様な結果になる．この結果より，実際の広域ネットワークにおいて，PSPacer を使えば，FIFO サイズが小さなルータを経由する場合でも，安定した通信が期待できる．

5.3 2 対 2 通信

規定のボトルネック帯域に対して，複数のパケットフローをそれぞれ送信側でペーシングした場合の効果を評価するため，2 つのパケットフローが 1 つのボトルネックリンクを通る場合の挙動について実験した．ルータの FIFO サイズは 32 KB とした．パケットフロー（フロー A）の開始 5 秒後に，もう一方のパケットフロー（フロー B）を開始し，それぞれ 120 秒，110 秒通信する．フロー A，B は PSPacer と TBF の組合せ（PSPacer+PSPacer，TBF+TBF，PSPacer+TBF，TBF + PSPacer）で，それぞれ 200 Mbps に帯域制限する．フローの目標帯域の合計はボトルネック帯域以下なので，ACK クロッキングの効果が小さく，ON-OFF トラフィックになる傾向がある．それぞれの結果を，**図 9** に示す．破線が各パケットフローの帯域であり，実線がその合計である．測定は 1 秒間隔で行った．

PSPacer + PSPacer の場合（図 9 (a)），それぞれの帯域は正確に 200 Mbps に制御されている．合計帯域も 400 Mbps で安定しており，ネットワークを効率良く利用できている．一方，TBF では 5.2 節の実験で示したように，スロースタート時のバースト送信を抑えることができないため，500 Mbps のボトルネック帯域に対して，十分小さな 200 Mbps に帯域制御しているにもかかわらず，パケットロスが発生する．したがって，TBF + TBF の場合（図 9 (b)），立ち上がりは鈍く，輻輳回避フェーズでもパケットロスが起きるため，合計帯域が 200 Mbps に達しない．PSPacer + TBF（図 9 (c)），TBF + PSPacer（図 9 (d)）の場合，2 パケットフロー合計のバースト性は TBF + TBF より小さいので，合計帯域は大きい．また，TBF のバースト性に巻き込まれ，PSPacer のフローでもパケットロスを起こしているが，いずれもほぼ 200 Mbps を保持できている．

6. 議　　論

6.1 バスボトルネックの問題

PSPacer は，システムの最大送信帯域に占める目標帯域の割合に基づいて，帯域制御とバースト性トラフィックの平滑化を実現する．したがって，精密なペーシングを実現するためには，NIC の最大転送帯域でパケットを送信できる必要がある．しかし，GbE NIC を 33 MHz/32 bit PCI バスに接続している場合，ボトルネックは PCI バスになり，システムは 1 Gbps の

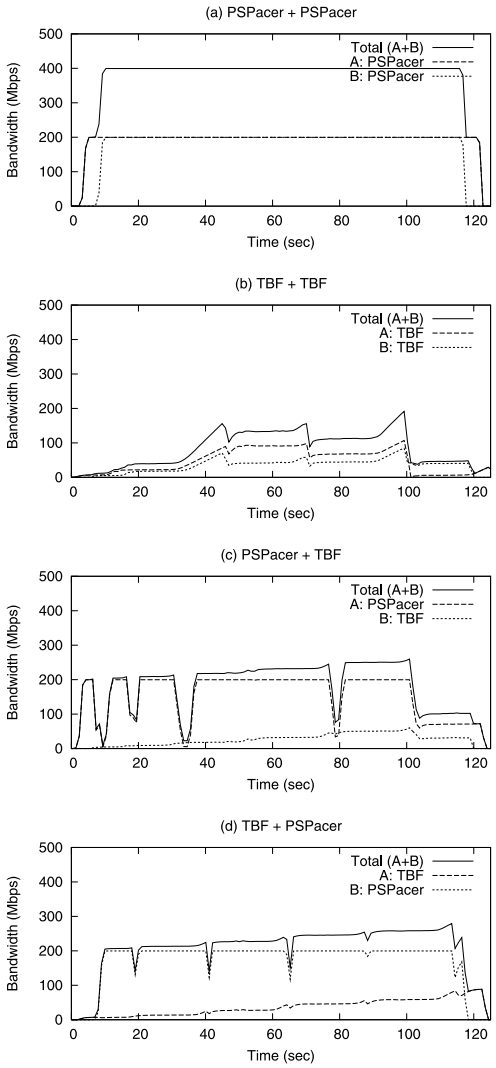


図 9 2 対 2 通信時の帯域（ボトルネック帯域 500 Mbps，RTT 200 ms，FIFO サイズ 32 KB）
Fig. 9 Bandwidth of 2-to-2 communication (Bottleneck bandwidth 500 Mbps, RTT 200 ms, FIFO size 32 KB).

送信レートでパケットを送信できない．この場合でも，PSPacer は，ユーザがシステムの最大送信帯域を明示的に指定することで，帯域制御できるが，PCI バスの振舞いが十分安定しないので，出力されるトラフィックは精密には平滑化できない．

6.2 CPU 負荷およびメモリ帯域への影響

PSPacer では，無通信期間にギャップパケットを送信するため，PSPacer を使わずに，同じ実効帯域で通信する場合よりも，CPU やメモリ帯域に対する負荷の増加が考えられる．ギャップパケットは PSPacer 内部で生成するので，通常のパケットとは異なり，ユーザ空間からカーネル空間へのデータコピーや，プロトコルスタックによる処理が不要であり，CPU 負荷は比較的小さい．また，送信完了割込み数の増加による CPU 負荷の増加が考えられるが，通常の GbE NIC では割込み発生を抑制する機能を持つため，性能劣化は NIC 依存である．表 1 で示した環境であれば，1 Gbps での UDP 通信時の CPU 使用率が 40%であるのに対して，TBF および PSPacer で 500 Mbps に帯域制御した場合の CPU 使用率は，それぞれ 10%，15%である．メインメモリから NIC への DMA 転送量は，最大送信帯域での通信時と変わらないが，ギャップパケットの生成にはソフトウェアによるユーザ空間からカーネル空間へのコピーを要しないので，メモリ帯域への負荷はやや小さい[1)]．今後，実アプリケーションを用いて，これらの負荷の影響を評価する必要がある．また，パケット間ギャップが大きな場合，プロトコルスタックからデバイスドライバに対して，複数パケットをまとめて授受する TCP セグメンテーションオフロード機能を利用して，ギャップパケットをまとめて授受することで負荷がさらに軽減できると考える．

6.3 さまざまな性質のネットワークへの適用

5.3 節の実験で示したように，一定の利用可能帯域が保証されたネットワークでは，すべてのパケットフローをペーシングし，それらの合計帯域がボトルネック帯域を上回らないように制御することによって，複数のパケットフローが合流しても，バースト性が最小限に抑えられ，安定して高い帯域利用率が得られる．この実験では，4 台の PC を使用したが，論文 7) では，16 台構成の 2 クラスタ間における MPI 通信に対して提案方式を適用し，NAS パラレルベンチマーク（IS）の性能が 1.6 倍から 2 倍に向上することを示している．このように規定の帯域に対して，できるだけ多くのパケットフローが安定して通信するために，提案手法は有効である．

一方，利用可能帯域が保証されないネットワークでは，(1) クロストラフィックの発生によりボトルネック帯域が変動する，(2) 送信元で精密にペーシングしても，中間ノードを経由するに従ってパケット間ギャップが乱れ，バースト性が加わってしまうという問題が考えられる．(1) に対しては，ネットワーク状況を監視し，目標帯域にフィードバックする機構が必要だと考える．1 つのアプローチとして，論文 1) では，TCP の輻輳ウィンドウサイズと RTT から目標帯域を見積もる方法を示した．この方法では，プロトコルスタックが保持している情報をそのまま利用するので，PSPacer の変更だけで実現できるという利点

がある．また，QoS 制御など，ネットワーク全体の帯域保証機構と組み合わせることも有効だと考える．(2) に対して，商用 FTTH ネットワーク（平均利用可能帯域 40 Mbps，RTT 6 ms）を経由し，PSPacer と TBF を用いて UDP フローを帯域制限した場合の挙動を調査した．その結果，平均帯域は約 40 Mbps とほぼ同じであるのに対して，パケットロス率は，PSPacer が 0.35%，TBF が 1.6%となり，受信時のパケット間ギャップは PSPacer を利用した方が平滑化されたという結果が得られた．論文 8) では，さらなる考察を行っているが，さまざまな性質のネットワークにおいて，バースト性がどの程度増加するのか，さらに評価が必要だと考える．

7. 関 連 研 究

TCP/IP のスロースタート時に対するペーシングの効果は，WEB トラフィックへの対応として，論文 10)〜12) などで提案されている．また，論文 6) では，MPI 通信において，通信再開時に，スロースタートの代わりに高精度タイマにより生成したクロックによるペーシングを用いる効果について述べている．

精密なペーシングの実現は，いかにパケット送信間隔を精密に制御できるかに依存する．多くの研究では，ペーシング用のタイマとして，1 ms や 10 ms のタイマを使用している．数 Mbps 程度までのトラフィックであれば，10 ms のタイマでもペーシングを実現できるが，1 Gbps 程度以上では難しい．一方，IA32 システムでは，高精度タイマとして APIC タイマ，HPET（High Precision Event Timer）が備わっている．しかし，論文 13) では，ソフトウェアによってペーシングを実現するには，パケットフローごとに μs の高精度タイマを維持する必要があり，オーバヘッドが大きいので，実装は困難であると報告している．論文 14) 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が提案され，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．しかし，この方式は，一定のバーストを許容するので，精密なペーシングの定義を満たすことはできない．

GtrcNET-1 [3] は，MAC 層において IPG を調整することで精密なペーシングを実現している．また，いくつかの NIC は，IPG を設定するためのレジスタを持っているが，ペーシングの実装に用いるには，制御可能な範囲が十分ではなく，実パケットに応じて，その大きさを動的に計算し，制御することもできない．提案方式では，IPG を制御するのではなく，実パケット間にギャップパケットを挿入することで同様の効果を実現した．論文 15) では，Chelsio T110 の TOE（TCP Offloading Engine）に実装されたペーシングを用いた，大陸間高速ネットワーク通信の実験結果について述べている．高速インタフェースに対して，TOE 機能は有効であるが，このようなハードウェア固有の機構を使うことで，NIC やデバイスドライバに依存した実装となる．今後，ペーシング機能も含め，オペレーティングシステムに対するインタフェースの標準化が必要であると考える．

フローごとの重み付けに応じて帯域を共有できるパケットスケジューリングは，タイムスタンプベースとラウンドロビンベースの 2 方式に分類できる．WFQ（Weighted Fair Queuing）[19] は前者の 1 つであり，フローごとにバッファを持ち，パケットごとに送信完了時刻を計算する．そして，各バッファの先頭のパケットでもっと送信完了時刻が早いものから順に送信する．提案方式のスケジューリングは，タイムスタンプベースに分類でき，無通信時間を制御するギャップパケットのために仮想的なバッファを持つと考えることができる．WFQ はアクティブなフロー間で，帯域を重み付けの比で分配するが，提案方式は各フローのパケット間ギャップを保証するために帯域を相対値ではなく絶対値で指定する点が異なる．

8. ま と め

本論文では，実パケット間にギャップパケットを送信することで，専用ハードウェアを必要としない，ソフトウェアによる精密ペーシング方式の提案と評価について述べた．ギャップパケットを用いることで，従来のタイマ割込みベースの方式では実現が困難であった，パケット送信の精密なスケジューリングが可能になった．提案方式をイーサネット上で実現するために，ギャップパケットとして，IEEE 802.3x で規定される PAUSE パケットを利用した．そして，提案手法を実現した PSPacer を実装し，ギガビットイーサネットを持つ一般的な PC を用いて，8 Kbps から 930 Mbps の範囲で，100 個の IP 通信のパケットフローごとに精密にペーシングできることを実証した．イーサネットでは，PAUSE パケットを利用できたが，他のネットワークでも，何らかの方法でギャップパケットを生成することができれば，同様の効果が期待できる．

さらに，PSPacer を用いたペーシングの効果を，往復遅延が 200 ms の高遅延環境における TCP/IP 通信性能によって評価した．その結果，トークンバケット方式と比較して，ほぼ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ように改善できることを示した．

現在の実装では，目標帯域を静的に指定するので，利用可能な帯域の変動が大きなネットワークでは適用が困難な場合がある．今後は，提案方式の適用範囲を広げるために，TCP が持つ情報を利用するなど，ネットワーク状況を目標帯域にフィードバックする手法を開発する予定である．また，利用可能帯域が保証されないネットワークにおける，提案方式の効果を評価する予定である．

なお，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．

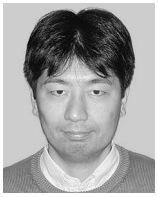
参 考 文 献

1) Takano, R., Kudoh, T., Kodama, Y., Matsuda, M., Tezuka, H. and Ishikawa, Y.: Design and Evaluation of Precise Software Pacing Mechanisms for Fast Long-Distance Networks, *PFLDnet2005* (Feb. 2005).
2) 高野了成，工藤知宏，児玉祐悦，松田元彦，手塚宏史，石川 裕：ソフトウェアによる精密ペーシング機構の提案と評価，インターネットコンファレンス 2004 (Oct. 2004).
3) Kodama, Y., Kudoh, T., Takano, R., Sato, H., Tatebe, O. and Sekiguchi, S.: GNET-1: Gigabit Ethernet Network Testbed, *IEEE Cluster 2004* (Sep. 2004).
4) Tatebe, O., Ogawa, H., Kodama, Y., Kudoh, T., Sekiguchi, S., Matsuoka, S., Aida, K., Boku, T., Sato, M., Morita, Y., Kitatsuji, Y., Williams, J. and Hicks, J.: *The 2nd Trans-Pacific Grid Datafarm Testbed and Experiments for SC2003, IEEE/IPSJ SAINT 2004 Workshops*, pp.26–30 (Jan. 2004).
5) 高野了成，石川 裕，工藤知宏，松田元彦，児玉祐悦，手塚宏史：並列アプリケーション実行における TCP/IP 通信挙動の評価，インターネットコンファレンス 2003 (Oct. 2003).
6) 松田元彦，高野了成，石川 裕，工藤知宏，児玉祐悦，岡崎史裕，手塚宏史：MPI ライブラリと協調する TCP 通信の実現，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG12 (ACS11) (2005).
7) 松田元彦，石川 裕，鐘尾宜隆，枝元真彦，岡崎史裕，鯉江英隆，高野了成，工藤知宏，児玉祐悦：GridMPI Version 1.0 の概要，*SWoPP 2005*, 情報処理学会 (Aug. 2005).
8) Takano, R., Kodama, Y., Kudoh, T., Matsuda, M., Okazaki, F. and Ishikawa, Y.: Realtime Burstiness Measurement, *PFLDnet2006* (Feb. 2006).
9) Michiel, H. and Leavens, K.: Teletraffic engineering in a broad-band era, *Proc. IEEE*, Vol.85, No.12, pp.2007–2033 (Dec. 1997).
10) Visweswaraiah, V. and Heidemann, J.: Improving Restart of Idle TCP Connections, *USC TR 97-661* (Nov. 1997).
11) Aggarwal, A., Savage, S. and Anderson, T.: Understanding the performance of TCP pacing, *IEEE INFOCOM*, pp.1157–1165 (Mar. 2000).
12) Aron, M. and Durschel, P.: TCP: Improving Startup Dynamics by Adaptive Timers and Congestion Control, Technical Report TR98-318, Rice Univ. (1998).
13) Antony, A., Blom, J., de Laat, C., Lee, J. and Sjouw, W.: Microscopic Examination of TCP flows over transatlantic Links, *iGrid2002 special issue, Future Generation Computer Systems*, Vol.19, Issue 6 (2003).
14) Kamezawa, H., Nakamura, M., Tamatsukuri, J., Aoshima, N., Inaba, M., Hiraki, K., Shitami, J., Jinzaki, A., Kurusu, R., Sakamoto, M. and Ikuta, Y.: Inter-layer coordination for parallel TCP streams on Long Fat pipe Networks, *SC2004* (Nov. 2004).
15) Nakamura, M., Kurusu, R., Marti, F., Sakamoto, M., Ikuta, Y., Tamatsukuri, J., Sugawara, Y., Aoshima, N., Inaba, M. and Hiraki, K.: Experimental Results of inter-layer cooperative hardware for FRC-TCP on 10 Gbps Ethernet WANPHY 18,500 km Network, *PFLDnet2005* (Feb. 2005).
16) Xu, L., Harfoush, K. and Rhee, I.: Binary Increase Congestion Control for Fast Long-Distance Networks, IEEE *INFOCOM 2004* (Mar. 2004).
17) T. Kelly's SACK-tag patch. http://www-lce.eng.cam.ac.uk/~ctk21/code/
18) Floyd, S.: HighSpeed TCP for Large Congestion Windows, RFC 3649 (Dec. 2003).
19) Parekh, A.K. and Gallager, R.G.: A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case, *IEEE/ACM Trans.Networking*, Vol.1, No.3, pp.344–357 (June 1993).

(平成 17 年 10 月 4 日受付)
(平成 18 年 1 月 20 日採録)

高野　了成（正会員）

1997 年東京農工大学工学部電子情報工学科卒業．1999 年同大学大学院工学研究科電子情報工学専攻博士前期課程修了．2005 年同大学院電子情報工学専攻博士後期課程単位取得満期退学．2003 年 4 月（株）アックス入社．GridMPI 開発に従事．オペレーティングシステムに興味を持つ．



工藤　知宏（正会員）

1991 年慶應義塾大学大学院理工学研究科博士課程単位取得退学．東京工科大学助手，講師，助教授を経て，1997 年より新情報処理開発機構並列分散システムアーキテクチャつくば研究室長，2002 年より産業技術総合研究所グリッド研究センタークラスタ技術チーム長．博士（工学）．並列処理，通信アーキテクチャに関する研究に従事．電子情報通信学会，IEEE CS 各会員．

児玉　祐悦（正会員）

1962 年生．1986 年東京大学工学部計数工学科卒業．1988 年同大学大学院情報工学専門課程修士課程修了．同年通産省電子技術総合研究所入所．2001 年独立行政法人産業技術総合研究所に改組．現在，同研究所グリッド研究センター主任研究員．データ駆動やマルチスレッド等の並列計算機システムの研究に従事．特にプロセッサアーキテクチャ，FPGA，ネットワークエミュレータ等に興味あり．博士（工学）．情報処理学会奨励賞，情報処理学会論文賞（1990 年度），市村学術賞（1995 年）等受賞．電子情報通信学会，IEEE CS 各会員．

松田　元彦（正会員）

1988 年京都大学理学部卒業．同年住友金属工業（株）入社．1995 年から 1999 年まで技術研究組合新情報処理開発機構に出向．2003 年より独立行政法人産業技術総合研究所．現在同研究所グリッド研究センター主任研究員．工学博士．並列計算システム，クラスタシステムおよびグリッド環境での高性能計算に関する研究に従事．

石川　　裕（正会員）

1987 年慶應義塾大学大学院理工学研究科電気工学専攻博士課程修了．工学博士．同年電子技術総合研究所入所．1993 年技術研究組合新情報処理開発機構出向．2002 年より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助教授．クラスタ・グリッドシステムソフトウェア，高信頼システムソフトウェア開発技術，実時間分散システム，次世代高性能コンピュータシステム等に興味を持つ．

岡崎　史裕（正会員）

1987 年京都大学大学院工学研究科電気 2 専攻修士課程修了．同年住友金属工業（株）入社．1998 年より 4 年間技術研究組合新情報処理開発機構の並列処理の研究に従事．2003 年より独立行政法人産業技術総合研究所グリッド研究センターに派遣．ネットワーク運用と GridMPI 開発に従事．