フロー単位バッファ制御における高速かつ柔軟なフロー状態管理機構

篠原　悠介[†a)]　　戸出　英樹[††]　　村上　孝三[†]

Fast and Flexible Flow State Management Mechanism on a Per-Flow Buffer Control Scheme

Yusuke SHINOHARA[†a)], Hideki TODE[††], and Koso MURAKAMI[†]

あらまし　現在の IP 網では，基本的に Best Effort サービスのみを提供しているため，各フローに対応した QoS 保証が困難である．このため，今後はルータに高速動作，高スケーラビリティを提供する制御機構を具備する必要がある．筆者らはこれまでに高い性能を提供するバッファ制御方式 Dual Metrics Fair Queueing（DMFQ）を提案している．DMFQ は既に計算機シミュレーションによってその有効性を確認されている．また，DMFQ はハードウェア設計され，約 6.8 Gbit/s のスループットを達成しているが，現段階では将来の 10 Gbit/s イーサネットに対応することは困難である．また，実ネットワーク環境における動作実験を通した性能の検証が行われていない．そこで本論文では，DMFQ をベースとし，更に高速なネットワークに対応可能な方式へと拡張提案するとともに，実証実験を通してその Feasibility を検証する．

キーワード　ルータ，バッファ制御方式，動作速度，ハードウェア設計，実証実験

1. まえがき

近年，ブロードバンドネットワークの普及が進んでおり，これにより様々な種類のトラヒックが急増している．このため，各種トラヒックに対する QoS（Quality of Service）保証の必要性が高まっている．しかし，現在の IP（Internet Protocol）網は，主に Best-Effort 型サービスのみを提供しているため，QoS 保証の実現は困難である．そこで，ルータの出力バッファ部においてパケット廃棄制御やスケジューリングを行う Active Queue Management（AQM）と呼ばれる機構が盛んに研究されている．AQM の代表的な例として Random Early Detection（RED）[1] が挙げられるが，フロー間において帯域配分等に関する不公平性の生じる可能性がある [2]．そこで，RED における不公平性の問題を解決するためにフロー管理を行い，バッファ内滞留パケット数やフロー到着レートなどの情報を用いてパケット廃棄を行う Fair Random Early Drop（FRED）[2]，Core-Stateless Fair Queueing（CSFQ）[3]，Rainbow Fair Queuing（RFQ）[4] や，厳密なフロー管理を行わず，公平性向上を目指した Stochastic Fair Blue（SFB）[5] といった様々なアルゴリズムが提案されている．一方，パケットの送出スケジューリングを行うことにより，公平性を向上させる方式として，Deficit Round Robin（DRR）[6] 等が提案されている．DRR はフローごとにキューを用意し，重み付けを行った転送量をもとに，RR でサービスを行う方式であり，極めて公平性の高いアルゴリズムである．また，DRR は，実装を考慮に入れ，公平性を維持しつつ制御を簡単化する方式も提案されている [7]．他にも，Virtual Finish Time をもとに転送順を管理する Weighted Fair Queueing（WFQ）ベースの方式が多数存在する [8]〜[10]．このように，スループット，公平性の向上を目指したアルゴリズムが多数提案されている．しかし，これらのパケット送出スケジューリング方式は公平性の改善や，スループット向上には有効であるものの，個別のキューを多数必要とするなど，実装面における複雑性や，適切なパラメータ設定が困難であるといった

† 大阪大学大学院情報科学研究科情報ネットワーク学専攻，吹田市
Department of Information Networking, Graduated School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan
†† 大阪府立大学大学院工学研究科電気・情報系専攻知能情報工学分野，堺市
Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, Sakai-shi, 599–8531 Japan
a) E-mail: shinohara.yusuke@ist.osaka-u.ac.jp

問題がある．

また，複数サービスクラス化を実現する Differentiated Service（DiffServ）[11]，Propotional Differentiated Services [12], [13] が提案されている．しかし，同一クラス内における不公平性は依然残るものと考えられ，同一クラス内におけるコネクション間の公平性の実現が課題である．

筆者らはこれまでに公平性の向上，QoS 保証の実現を目指し，バッファ制御方式 DMFQ（Dual Metrics Fair Queueing）[14] を提案している．DMFQ は，フローごとの瞬時的，履歴的なネットワーク資源使用量を考慮に入れた公平性の実現を目指しており，ハードウェア実装を想定したアルゴリズムである．実際にハードウェア設計を行った結果，約 6.8 Gbit/s のスループットを達成している．しかし，現段階では将来普及が予想される 10 Gbit/s イーサネット等の高速なネットワークに対応するスループットが達成されておらず，処理能力がボトルネックになる恐れがある．また，実ネットワーク環境における実証実験が行われていないため，運用上の挙動が不明である．

そこで本論文では，バッファ制御 DMFQ を拡張し，高スループットと実ネットワークへの高い適用性を実現するフロー管理機構を提案する．具体的には，スループット向上のため，DMFQ のボトルネックであるパケット到着時の処理を簡単化する．これにより，DMFQ と同等の性能を維持しつつ，10 Gbit/s イーサネット等の高速なネットワークに対応するスループットを達成する．

2. DMFQ

2.1 概　　要

DMFQ [14] ではフローごとの公平性実現のために瞬時的なネットワーク資源使用量だけでなく，それまでに消費してきた履歴的なネットワーク資源使用量も考慮し，公平性を向上させる．具体的には，瞬時的，履歴的なネットワーク資源使用量の指標としてそれぞれ，パケット到着レート，フロー継続時間を用いる．以下にそのアルゴリズム，問題点を概説する．なお，より詳細な説明は文献 [14] を参照されたい．

2.2 フロー管理

DMFQ におけるフローは，同じヘッダ情報をもちある一定時間間隔内に到着するパケット系列として定義される．DMFQ におけるフロー管理では実際にバッファ内にパケットをもつアクティブフローだけでなく，

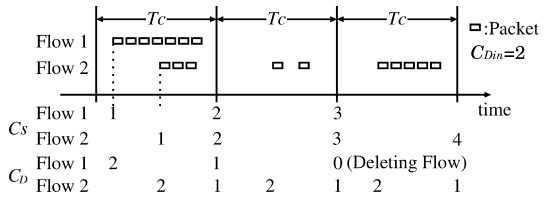

図 1　継続カウンタ C_S と削除カウンタ C_D の更新例
Fig. 1　An example of the succession and deletion counters.

過去に通過したフローを一定時間管理する（以下，このフローを管理フローと呼び，アクティブフローと使い分ける）．また，一定時間パケットが到着しないフローは管理エントリから削除される．このフロー削除，及び継続時間の抽出を簡単化するために，継続カウンタ C_S と削除カウンタ C_D を導入している．これらのカウンタは，一定間隔 T_C ごと（これを “更新サイクル” と呼ぶ）に増減される（図 1）．

継続カウンタ C_S は継続時間の目安として用いられる．新規フローとして登録されたときに，1 に初期化され，T_C ごとに 1 ずつインクリメントされる．また，削除カウンタ C_D はフロー削除のために使用される，当該フローのパケットが到着するたびに C_{Din} に初期化される．そして，T_C 間隔で 1 ずつデクリメントされる．その結果，C_D の値が 0 になると一定時間パケットの到着がないことを示すため，そのフローを管理テーブルから削除する．

2.3 アルゴリズム

DMFQ ではフローごとの到着レート（W_{rate}）とフロー継続時間（W_{con}）を用いてパケット廃棄を行う．W_{rate}，W_{con} は式 (1) によって算出される．ただし，R_c，pkt_size，now，T_f はそれぞれ Time Sliding Window（TSW）法のウィンドウ幅 [15]，到着パケットのサイズ，現在の時刻，前回当該フローのパケットが到着した時刻である．

$$W_{rate} \leftarrow \frac{W_{rate} \cdot R_c + pkt\_size}{now - T_f + R_c}, \quad W_{con} = C_S \tag{1}$$

また，これらのクラス内平均値として W'_{rate}，W'_{con} を算出する（式 (2)）．ただし，$Service\_rate$，$Fact$，F は当該クラスの利用可能な帯域，アクティブフロー数，管理フロー数を表す．

$$W'_{rate} = Service\_rate/Fact,\ W'_{con} = \textstyle\sum_F C_S/F \tag{2}$$

DMFQ では，これらの情報をまとめて扱うために，フロー状態係数 W，そのクラス内平均値 W_{avr} を算出する（式 (3)）．

$$W = W_{rate} \cdot W_{con}, \quad W_{avr} = W'_{rate} \cdot W'_{con} \quad (3)$$

最後に，W と W_{avr} の比較によりパケット廃棄確率 P を算出する．しかし，W と W_{avr} を単純に比較して算出された廃棄確率では，推定誤差による不適切な廃棄が懸念されるため，W_{rate} が W'_{rate} を上回るフローに対してのみ比較に基づいた式 (4) の廃棄確率に従うパケット廃棄を行う．

$$P = \begin{cases} 0 & (W_{rate} < W'_{rate}) \\ \max\left[0, 1 - W_{base}/W\right] & (W_{rate} \geq W'_{rate}) \end{cases} \quad (4)$$

ここで，W_{base} は比較基準値であり，式 (5) で算出される．

$$W_{base} = W_{avr} \cdot th/q_{avr} \quad (5)$$

ただし，q_{avr}，th はそれぞれ平均キュー長，バッファしきい値を表す．q_{avr} と th の比較結果を W_{base} に反映させることにより，バッファを有効に活用することが可能となる．

2.4 ハードウェア設計

DMFQ では，高速処理を実現するために，以下の方針に基づいたハードウェア設計が行われている．

2.4.1 平均値の算出

平均値である W_{base}，W_{avr}，q_{avr} の演算は，パケット到着の一連の処理とは独立に行う．パケット到着時に使用する各平均値については，その時点で既に算出されている値を参照する．これにより，到着時の演算量が削減される．

2.4.2 設定パラメータの制限

W_{rate} の算出の際，処理に時間のかかる W_{rate} と R_c の乗算処理が必要になる．これを高速化するため，R_c に制約を設け，算出に時間がかかる乗算を，その算出が簡単なシフト演算（1 clock）で行う．

2.4.3 パケット廃棄処理

パケット廃棄処理では，P を算出した後，0〜1 のランダム変数 $Rand$ と比較する．そして，その大小によって到着パケットを廃棄する．しかし，$Rand$ は 0〜1 のランダム変数であるため，$1 - Rand$ もランダム変数である．よって廃棄条件は，$W_{base}/W \leq Rand$ とすることができる．更に，W_{base} は，パケット到着時の値をそのまま使用するため，廃棄処理時には算出処理は不要である．また，$Rand$ はランダム変数であり，他の変数をもとにした算出処理が不要であるため，廃棄条件を式 (6) に示すようにすることで，W の算出と W_{base}/W の算出を同時に行うことができる．

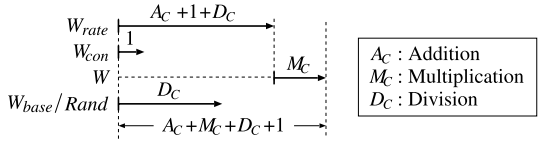

図 2 DMFQ のタイミングチャート
Fig. 2 Timing chart for the arrival phase of DMFQ.

$$\begin{cases} W < \dfrac{W_{base}}{Rand} & \rightarrow \text{バッファへ格納} \\ W \geq \dfrac{W_{base}}{Rand} & \rightarrow \text{廃棄} \end{cases} \quad (6)$$

以上のようなハードウェア設計により，加算，乗算，除算処理にそれぞれ，A_C (=1 clock)，M_C (=2 clock)，D_C (=4 clock) かかるものとすると，$(A_C + M_C + D_C + 1)$ clock(=8 clock) で入力処理を完了することができる（図 2）．

更に，入力処理で最も処理量が大きいのは W_{rate} の算出処理（$(A_C + D_C + 1)$clock(=6 clock)）である．そこで，パイプライン処理を導入することにより，6 clock ごとに入力を受付けることが可能になる．

以上のような指針のもと，実際にハードウェア設計を行い，Synopsys Design Compiler [16] を用いて論理合成を行った結果，64 byte パケットが連続到着する環境下で，約 6.8 Gbit/s（0.18 μm CMOS テクノロジー）のスループットを達成することが明らかになっている．

3. 拡張方式とその設計法

3.1 概　　要

DMFQ は公平性を向上させ約 6.8 Gbit/s のスループットを達成する効果的なバッファ制御方式である．しかし，10 Gbit/s イーサネットのような高速なネットワークにおいては，その処理がボトルネックになる．DMFQ の処理において，ボトルネックになるのはパケット到着時の処理であり，$(A_C + M_C + D_C + 1)$ clock の処理量が必要となる．この処理において，特に W_{rate} は TSW 法を用いて式 (1) のように算出するが，この算出処理がボトルネックとなり，$(A_C + D_C + 1)$ clock

の処理量が必要となる．そこで本論文では，DMFQ を拡張し，DMFQ と同等の公平性を維持しつつ，W_{rate} の算出処理を簡単化することにより，より高速に動作する方式を提案する．なお，本方式は基本的には DMFQ と同様のアーキテクチャである．

3.2 到着レート算出方式

提案方式では，処理量の大きい除算処理を削除するために到着レート更新間隔 T_R（固定）を導入し，T_R ごとに全フローの W_{rate} を更新する．これにより，パケット到着時には，既に算出されている当該フローの W_{rate} を読み出すのみでこれを取得することが可能となる．また，W_{rate} 更新時においても，式 (1) における演算 $now - T_f$ を固定値 T_R とすることができ，T_R の設定値に制約条件を加えることにより，複雑な除算処理を簡単なシフト演算とすることができる．

具体的には，全フローのフロー到着レートを T_R ごとに式 (7) のように算出する．

$$W_{rate} \leftarrow \frac{W_{rate} \times R_c + total\_pkt\_size}{T_R + R_c} \qquad (7)$$

ただし，$total\_pkt\_size$ は，当該フローの前回の到着レート更新以降に到着したパケットサイズの合計値を表す．

更に，$service\_rate \times (T_R + R_c)$ は全フロー共通で用いるので，W'_{rate} の算出に使用することにより，更に算出処理を削減することができる（式 (8)，(9)）．

$$W_{rate} \leftarrow W_{rate} \times R_c + total\_pkt\_size \qquad (8)$$

$$W'_{rate} = service\_rate \times (R_c + T_R)/Fact \qquad (9)$$

以上のような算出法により，DMFQ と同様の W_{rate} を算出しつつ，到着処理を高速化することができる．

3.3 ラウンドロビン，更新の並列化

本提案方式におけるハードウェア設計の方針を以下に示す．本設計では，到着レート更新点に更新処理の負荷が集中するのを避けるため，W_{rate} をフロー ID が小さいフローから順に，ラウンドロビンで更新する．T_R を “スロット” に分割し，1 スロット中に 1 フローの W_{rate} を更新する．このとき，フローごとに到着レート更新点からのタイムラグが発生するが，それぞれのフローのみに注目すると，T_R ごとに更新しているので，W_{rate} は正しく更新される．更に，W_{rate} の算出精度を向上させるためには，T_R を小さく設定する必要がある．しかし，過度に小さく設定すると W_{rate} を更新できるフロー数が制限されてしまう．そこで，

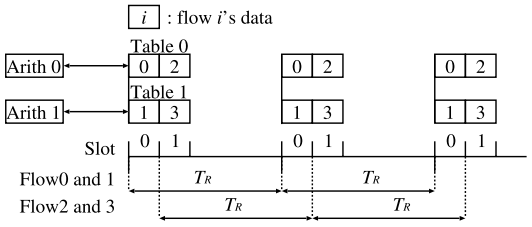

図 3 到着レートの更新
Fig. 3 Parallel calculation of packet arrival rate with round robin.

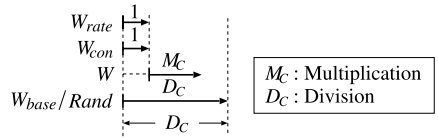

図 4 提案方式のタイミングチャート
Fig. 4 Timing chart for the packet arrival process in the extended DMFQ.

本設計では演算器，及び到着レート管理テーブルを複数実装し，この処理を同時並行的に行うことを考える．提案方式では，各スロットにおいて，各テーブルに管理されているフローの W_{rate} を，対応する演算器によって更新する．このように演算を並列化し，異なるフローの W_{rate} を同時に更新することにより，W_{rate} の算出精度を高めつつ，DMFQ と同等のフロー数を管理することができる．図 3 は二つの演算器（Arith0, 1）で更新処理の並列化を図っている例である．

以上のように到着レートを更新することにより，パケット到着時の処理量は D_C(clock)(=4 clock) に軽減することができる（図 4）．更に，T_R をより小さく設定することが可能となり，その結果，精度の高い W_{rate} を得つつ，DMFQ とほぼ同数のフロー数を扱うことが可能になる．

3.4 更新間隔の動的制御

以上のように，到着レート更新間隔 T_R を短く設定することにより，精度の高い到着レートを得つつ，高速な動作を実現することが可能である．しかし，T_R を固定値とすることにより，トラヒックの到着が少ないようなネットワークにおいても更新を頻繁に行うため，無駄な処理が増えてしまうという問題がある．そこで，ネットワーク状況に応じて T_R を動的に制御することにより，精度の高い到着レートを算出しつつ，不必要な処理を削除する．

提案方式では，以下のように T_R の制御を行う．

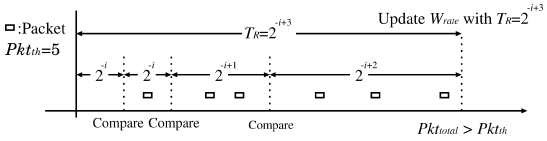

図 5　到着レート更新間隔の動的制御
Fig. 5　The dynamic control mechanism for T_R.

（1）　しきい値の導入

まず，T_R を制御する目安として，区間中に到着するパケット数のしきい値 pkt_{th} を導入する．提案方式では，1 更新区間中の到着パケット数 pkt_{total} が pkt_{th} を上回れば，十分にパケットが到着したことを示すため，到着レートの更新を行う．

（2）　更新間隔値の限定

提案方式では，T_R の設定値が常に 2^{-x} になるように制御するものとする（x の設定については，次項で詳述）．これにより，W'_{rate} の算出（式 (9)）において，$service\_rate \times (R_c + T_R)$ の加算，乗算処理を，算出が簡単なシフト演算と加算処理で行うことができる．

（3）　比較間隔の導入

最後に，比較間隔を導入する．本提案方式では，pkt_{total} と pkt_{th} の比較を比較間隔ごとに行い，$pkt_{total} > pkt_{th}$ であれば到着レートの更新を行う．更新間隔の制限を考慮に入れて，最短比較間隔 $T_{Rmin} = 2^{-i}$ を導入し，到着レートが更新されるまで比較間隔を $2^{-i}, 2^{-i}, 2^{-i+1}, 2^{-i+2}, \cdots$ と増加させる（図 5）．この比較間隔ごとに比較，更新を行うことにより，提案方式は T_R を $2^{-i}, 2^{-i+1}, 2^{-i+2}, 2^{-i+3}, \cdots$ とすることができる．一方，カウンタ値に制限を設けるため，最長比較間隔 T_{Rmax} を導入する．比較間隔が十分長い間隔 T_{Rmax} となれば，強制的に更新を行う．これにより，常に到着レートの更新を 2^{-x} ごとに行うことができる．

以上のような設計で Synopsys Design Compiler [16] を用いて論理合成を行った結果，本提案方式は表 1 のような諸元となり，16 K フローの識別が可能であり，64 byte パケットが連続到着する環境下で，約 10.2 Gbit/s（0.18 μm CMOS テクノロジー）のスループットを達成することが明らかになった (注1)．

4.　計算機シミュレーションによる性能評価

本章では，OPNET Modeler 10.5 [17] を用いた計算機シミュレーションにより，提案方式の性能を評価する．

表 1　提案方式の諸元
Table 1　The main characteristics of DMFQ implementation.

テクノロジー	0.18 μm CMOS
ゲート数	約 80 K
クロック周波数	160 MHz

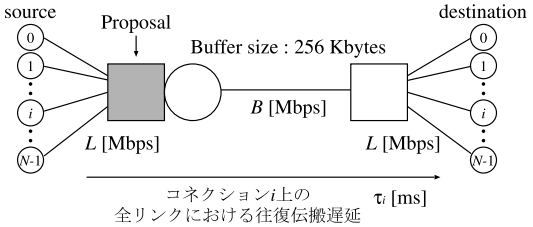

図 6　シミュレーションモデル
Fig. 6　Simulation model.

4.1　シミュレーションモデル

本評価では図 6 に示すシミュレーションモデルを用いる．送受信端数を $N = 32$ とし，送信端 i $(i = 0, 1, \cdots, 31)$ からルータ，及び次段のルータから受信端 i $(i = 0, 1, \cdots, 31)$ への回線速度を $L_i = L = 100$ [Mbit/s] とする．また，ボトルネックリンクの回線速度を $B = 100$ [Mbit/s]，送信端 i から受信端 i $(i = 0, \cdots, 31)$ までのコネクション i 上の全リンクにおける往復伝搬遅延を τ_i [ms] とする．TCP は，NewReno 実装 [18] に準拠したものを用い，比較対象方式として DRR [6]，RED [1]，FRED [2]，SFB [5]，CSFQ [3]，RFQ [4]，DMFQ [14] を用いた．なお，本シミュレーション評価で用いる提案方式のモデルは，フロー情報を格納しているメモリに対して複数のアドレスに同時並行的にアクセスすることができると仮定している．このため，同時に全フローの到着レートを更新することが可能である．一方，ハードウェアで実装した提案方式は，メモリに対して複数のアドレスに同時並行的にアクセスすることができないため，到着レートの算出は逐次的に実行され，フローによっては更新されるタイミングが異なる．しかし，**3. 3** の提案により，各フローの到着レートの更新時点は異なるものの，更新間隔は常に T_R であり，到着レートはシミュ

(注1)：提案方式において 16 K フロー以上を管理するためには，容量の大きい外部メモリにフロー情報を格納する必要があるが，外部メモリに対するアクセスは数クロックサイクルを要するため，10.2 Gbit/s のスループットを達成することができなくなる．また，0.18 μm CMOS テクノロジーにおけるメモリのアクセスタイムは 1〜3 ns 程度であり，バッファとして用いるメモリのビット幅を 64 bit 以上にすることにより，1 bit につき 100 ps 以内に処理を完了することができるため，10 Gbit/s の動作を実現することができると考えられる．

表 2　W_{rate} 算出誤差，更新回数
Table 2　Error and update times for W_{rate}.

Scheme	算出誤差	更新回数
提案方式（T_R 固定）	0.00863	61440
提案方式（T_R 動的制御）	0.00874	55260

レーション評価で用いたモデルで算出した値とほぼ同じ値を算出することができるため，この差は性能評価上無視できるものである．

パラメータに関して，各方式の推奨パラメータを基本的に採用し，EWMA の重み定数 $w = 0.002$, Quantum Size = 1.5（DRR），$(min_{th}, max_{th}) = (5, 192)$ [kByte]（RED），$(64, 192)$ [kByte]（FRED），$max_p = 0.10$（RED, FRED），$min_q = 2$（FRED），Bin Size = 13，$\delta = 0.01$（SFB），$th = 128$ [kByte], $K = K_\alpha = K_C = 400$ [ms]（CSFQ），$P = 100$ [Mbit/s]，$q\_threshold = 153.6$ [kByte]（RFQ），$R_c = 100$ [ms]，$C_{Din} = 2$，$T_C = 5.0$ [s]，$th = 64$ [kByte], (DMFQ，提案方式)，$(T_{Rmin}, T_{Rmax}) = (2^{-10}, 2^{-1})$ [s]，$pkt_{th} = 5$（提案方式）とした．

なお，全送信端はシミュレーション開始後 0〜3 s の間に一様分布に従いトラヒックの転送を開始させる．シミュレーション時間を 60 s（Sec. 4. 2，4. 3，4. 4），600 s（Sec. 4. 5）とし，それぞれ 10 回の平均を結果とした．

4. 2　到着レート W_{rate} にかかわる処理量

本節では，T_R を固定とした提案方式，T_R を動的に制御した提案方式の，到着レート W_{rate} の算出精度と，これにかかわる処理量を評価する．

送信端 i $(i = 0, \cdots, 31)$ は TCP トラヒックとして，3.0 MByte のファイルを発生させ，TCP セグメントに分割化し，IP パケットにカプセル化して転送する．1 ファイル転送が終了すると，平均 1.0 s の指数分布に従う時間間隔の後，ファイル転送を再開する．往復伝搬遅延は $\tau_i = 40$ [ms] する．また，本節では，同環境における W_{rate} の算出精度を比較するため，全方式におけるパケット廃棄規律として，DMFQ で算出した W_{rate} を用いた．更に，T_R を固定とした提案方式の設定パラメータとして，$R_c = 100$ [ms]，$C_{Din} = 2$，$T_C = 5.0$ [s]，$th = 64$ [kByte]，$T_R = 2^{-10}$ とした．

表 2 に各方式における算出誤差，W_{rate} の更新回数を示す．ただし，算出誤差は DMFQ で算出した W_{rate}/W'_{rate} と各方式で算出した W_{rate}/W'_{rate} の，全パケットにおける残差平方和の平均を用いた．

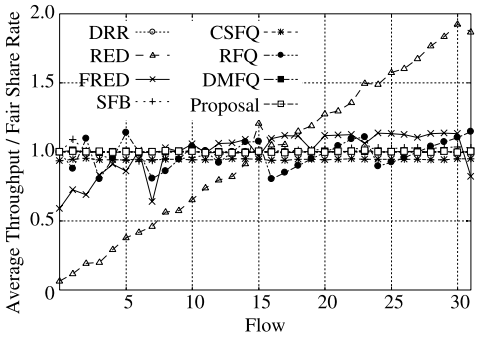

図 7　正規化スループット（Sec. 4. 3）
Fig. 7　Normalized throughput.
(Sec. 4. 3)

表 3　Fairness Index，リンク利用率
Table 3　Fairness index, link utilization.

	Sec. 4. 3		Sec. 4. 4		Sec. 4. 5		
Scheme	Fair	Link	Fair	Link	Fair	Link	Files
DRR	0.999	1.00	0.999	1.00	0.986	0.996	2507.2
RED	0.766	0.900	0.0352	0.999	0.722	0.850	1927.2
FRED	0.961	1.00	0.895	1.00	0.800	0.907	2281.9
SFB	0.998	1.00	0.955	1.00	0.888	0.734	1855.0
CSFQ	0.999	0.942	0.990	0.969	0.904	0.938	2352.9
RFQ	0.991	0.971	0.904	0.933	0.983	0.907	2262.3
DMFQ	0.999	1.00	0.998	1.00	0.975	0.957	2402.1
提案方式	0.999	1.00	0.997	1.00	0.975	0.957	2403.9

表 2 より，両方式とも，DMFQ と同等の高い算出精度を示している．また，パケット到着の少ない時間においても更新を行う “T_R を固定値とする提案方式” より，“動的に制御する提案方式” の方が更新回数を 10%削減できている．

4. 3　UDP トラヒックのみが存在する環境

送信端 i $(i = 0, 1, \cdots, 31)$ は，公平共有帯域である 3.125 Mbit/s（100 Mbit/s/32 = 3.125 Mbit/s）の i+1 倍の Constant Bit Rate（CBR）で送信する UDP トラヒック（Greedy Model）を発生させる．往復伝搬遅延 $\tau_i = 20$ [ms] $(i = 0, \cdots, 31)$ とする．

図 7 に各コネクションの正規化スループット，表 3 に Fairness Index（Fair），ボトルネックリンクの利用率（Link）を示す．正規化スループットは当該フローの単位時間当りのスループットを公平共有帯域で正規化した値とした．なお，**4. 3**，**4. 4**，**4. 5**，**5. 2**，**5. 3** で用いる Fairness Index は公平性を示す指標であり，サンプル値を x_i $(1 \le i \le n)$ とすると，以下の式で求められ，この値が 1 に近いほど公平性が高いことを示す [20].

$$\text{Fairness Index} = \left(\sum_{i=1}^{n} x_i\right)^2 / n \sum_{i=1}^{n} x_i^2 \qquad (10)$$

本評価では，各コネクションの平均スループットをサンプル値とした．一方，リンク利用率は，平均スループットをボトルネックリンクの回線速度で正規化した値として定義する．

図 7，表 3 より，RED，FRED，RFQ を用いた場合，公平に帯域を割り当てることができていない．一方，DRR，SFB，CSFQ，DMFQ，提案方式を用いた場合，ほぼ公平に帯域を割り当てることができている．

4.4 UDP，TCP トラヒックが混在する環境

送信端 0 は UDP トラヒック（100 Mbit/s，CBR）を，送信端 i $(i = 1, \cdots, 31)$ は TCP トラヒック（Greedy Model）を発生させる．往復伝搬遅延 $\tau_i = 20$ [ms] $(i = 0, \cdots, 31)$ とする．

図 8 に各コネクションの正規化スループット，表 3 に Fairness Index（Fair），ボトルネックリンクの利用率（Link）を示す．

図 8，表 1 より，RED を用いた場合，フローごとの情報を用いないため公平性が非常に低い．また，FRED，SFB，RFQ を用いた場合，RED よりは公平性は向上するものの，公平共有帯域を各フローに配分することができていない．一方，DRR，CSFQ，DMFQ，提案方式を用いた場合，ほぼ公平な帯域配分が可能となっていることが分かる．

4.5 TCP によるファイル転送

送信端 i $(i = 0, \cdots, 31)$ は TCP トラヒックとして，3.0 MByte のファイルを発生させ，TCP セグメントに分割化し，IP パケットにカプセル化して転送する．1 ファイル転送が終了すると，平均 1.0 s の指数分布に従う時間間隔の後，ファイル転送を再開する．往復伝搬遅延は 4 送受信端対ごとに $\tau_i = 10, 20, 30, 40, 50, 60, 70, 80$ [ms] とし，各送信端は往復伝搬遅延が同じ四つの受信端からランダムにあて先を選択する．

図 9 に送信端ごとの転送完了ファイル数，表 3 に Fairness Index（Fair），ボトルネックリンクの利用率（Link），転送完了ファイル数を示す．

図 9，表 1 より，RED，FRED ではコネクションの伝搬遅延の違いにより，転送ファイル数に大きな違いが生じている．また，SFB，CSFQ では，公平性は向上するものの，伝搬遅延の大きなコネクションにおける転送ファイル数が非常に少ない．RFQ は公平な帯域配分ができているものの，転送完了ファイル数が依然少ない．一方，DMFQ，提案方式では，伝搬遅延の大きなコネクションにおける転送ファイル数，及び公平性，スループットにおいても DRR と同等の優れた性能を示している．

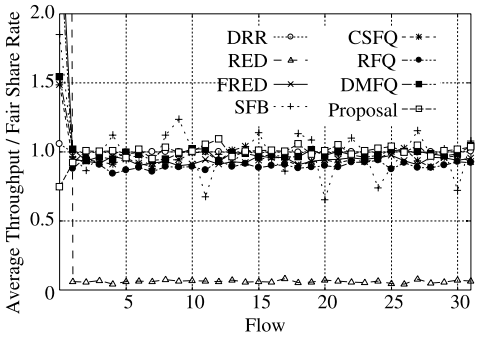

図 8 正規化スループット（Sec. 4. 4）
Fig. 8 Normalized throughput. (Sec. 4. 4)

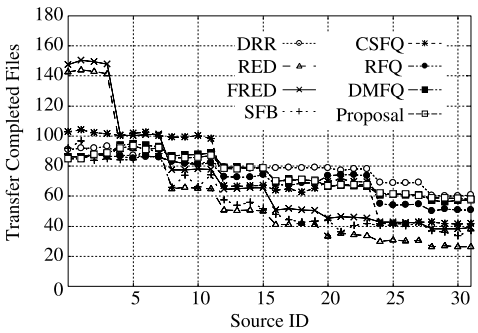

図 9 転送完了ファイル数（Sec. 4. 5）
Fig. 9 Transfer completed files. (Sec. 4. 5)

以上の結果から，本方式は，高速に動作しつつ，高い公平性を実現することができる方式であるといえる．

なお，本提案方式の性能は，その目安として用いている DRR に代表される，厳密なスケジューリングを行う方式に対し，公平性の面で劣っている．これは，提案方式はパケット到着時の廃棄制御に限定されていることがボトルネックになっている．しかし，高スケーラビリティ，高速な動作を実現するために，管理フロー数だけキューが必要なパケットスケジューリング方式ではなく，その実装がシンプルなパケット廃棄制御方式による公平性の実現を目指している．

5. 実証実験による性能評価

本章では，本提案方式をハードウェア実装した Field Programable Gate Array（FPGA）搭載ボードでの，実ネットワーク環境における実証実験により，本方式の性能を評価する．

5.1 検証条件

本評価では，図 10 に示すネットワーク環境を用いる．なお，本評価ではボトルネックリンクに 100BASE-TX を用いる．UDP トラヒックの発生には，トラヒック発生装置である SmartBits [21] を用い，TCP トラヒックの発生には，Iperf [22] を用いた．なお，各送信端では 0 s で転送が開始されるものとし，パケットサイズを 1.5 kByte とする．なお，本評価において，比較対象方式として FIFO，RED，DMFQ を FPGA 搭載ボードに実装し，使用した．

実証実験に際し，パラメータに関して，各方式の推奨パラメータを基本的に採用し，$(min_{th}, max_{th}) = (3072, 64)$ [kByte]，$max_p = 0.1$（RED），$R_c = 125$ [ms]，$T_C = 5.0$ [s]，$C_{Din} = 2$（DMFQ，提案方式），$th = 256$ [kByte]（DMFQ），$(T_{Rmin}, T_{Rmax}) = (2^{-10}, 2^{-1})$ [s]，$pkt_{th} = 5$（提案方式）とした．

5.2 UDP トラヒックのみが存在する環境

転送レートが異なるトラヒック入力における帯域配分の公平性を評価するため，送信端 i（$i = 0, \cdots, 2$）にそれぞれ 40，70，100 Mbit/s（CBR）で送信する UDP トラヒック（Greedy Model）を発生させる．

図 11 に，各コネクションの正規化スループットを示す．また，表 4 に各コネクションの平均スループットに対する Fairness Index（Fair）[20]，ボトルネックリンクの利用率（Link）を示す．

図 11 と表 4 より，FIFO，RED を用いた場合，各フローの入力レートに応じた帯域を与えており，各フローに公平に帯域を与えることができていない．一方，DMFQ，提案方式を用いた場合，各フローに公平共有帯域を与えることができている．

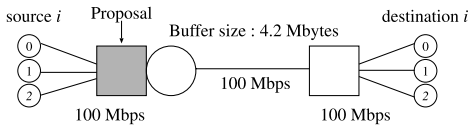

図 10 実験環境
Fig. 10 Experiment environment.

5.3 UDP，TCP トラヒックが混在する環境

送信端 0 に UDP トラヒック（100 Mbit/s，CBR，Greedy Model）を発生させ，送信端 1，2 に TCP トラヒック（Greedy Model）を発生させる．

図 12 に，各コネクションの正規化スループットを示す．また，表 4 に各コネクションの平均スループットに対する Fairness Index（Fair）[20]，ボトルネックリンクの利用率（Link）を示す．

図 12 と表 4 より，FIFO，RED を用いた場合，Flow 0 として過負荷な UDP トラヒックを発生させると，大量のパケット廃棄により，TCP フローである Flow 1，2 はトラヒックの発生を中止している．そ

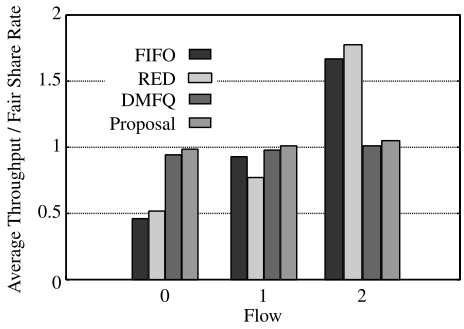

図 11 正規化スループット（Sec. 5. 2）
Fig. 11 Normalized throughput. (Sec. 5. 2)

表 4 Fairness Index，リンク利用率
Table 4 Fairness index, link utilization.

Scheme	Sec. 5. 2		Sec. 5. 3	
	Fair	Link	Fair	Link
Drop-Tail	0.808	1.00	0.343	1.00
RED	0.780	1.00	0.350	1.00
DMFQ	0.999	0.967	0.866	0.981
提案方式	0.999	1.00	0.857	0.985

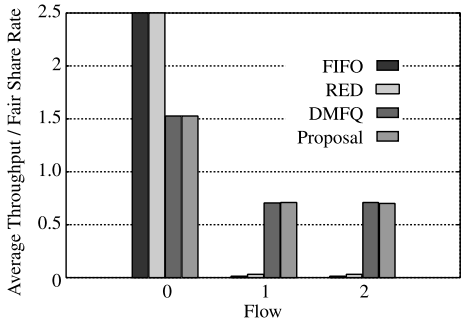

図 12 正規化スループット（Sec. 5. 3）
Fig. 12 Normalized throughput. (Sec. 5. 3)

の結果，Flow 0 がすべての帯域を独占してしまっている．一方，DMFQ，提案方式を用いた場合，Flow 0 に割り当てる帯域を下げ，Flow 1，2 にも帯域を与えることができている．TCP のレート増減のメカニズムにより，その実効スループットは UDP フローの約半分になっている．

以上の結果から，本提案方式は実ネットワーク環境においても主要なネットワークトラヒックである TCP トラヒックに対して公平な転送を行うことができることを確認した．このことから，本方式は 10 Gbit/s イーサネットに対応できるスループットを実現しつつ，公平性の高いフロー管理機構であるといえる．

6. むすび

本論文では，バッファ制御方式 DMFQ について述べ，拡張方式として，パケット到着時処理のボトルネックである，到着レート算出処理を高速に行う方式を提案した．そして，本提案方式のハードウェア設計を行い，実装を行った結果，10.2 Gbit/s のスループットを達成することを明らかにした．また，計算機シミュレーションに加え，FPGA 搭載ボードを用いて実ネットワーク環境における実証実験を行い，提案方式の有効性を確認した．

今後の課題として単一ノードの公平性に加え，End-to-End 間で有効な研究が挙げられる．

文　献

[1] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” IEEE/ACM Trans. Netw., vol.1, no.4, pp.397–413, Aug. 1993.

[2] D. lin and R. Morris, “Dynamics of random early detection,” Proc. ACM SIGCOMM’97, vol.27, no.4, pp.127–137, Sept. 1997.

[3] I. Stoica, S. Shenker, and H. Zhang, “Core-stateless fair queueing: Achieving approximately fair bandwidth allocation in high speed networks,” Proc. ACM SIGCOMM’98, vol.28, no.4, pp.118–130, Sept. 1998.

[4] Z. Cao, Z. Wang, and E. Zegura, “Rainbow fair queueing: Fair bandwidth sharing without per-flow state,” Proc. IEEE INFOCOM 2000, vol.2, pp.922–931, March 2000.

[5] W. Fang, D. Kandlur, and D. Saha, “Stochastic fair blue: A queue management algorithm for enforcing fairness,” Proc. IEEE INFOCOM 2001, vol.3, pp.1520–1529, April 2001.

[6] M. Shreedhar and G. Varghese, “Efficient fair queueing using deficit round robin,” IEEE/ACM Trans. Netw., vol.4, pp.375–385, June 1996.

[7] L. Lenzini, E. Mingozzi, and G. Stea, “Tradeoffs between low complexity, low latency, and farness with deficit round-robin schedulers,” IEEE/ACM Trans. Netw., vol.12, no.4, pp.681–693, Aug. 2004.

[8] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm,” Proc. ACM SIGCOMM, Austin, TX, pp.1–12, Sept. 1989.

[9] J. Bennett and H. Zhang, “WF2Q : Worst-case fair weighted fair queueing,” Proc. IEEE INFOCOM’96, vol.1, pp.120–128, 1996.

[10] J. Bennett and H. Zhang, “Hierarchical packet fair queueing algorithms,” Proc.ACM SIGCOMM, pp.143–156, Aug. 1996.

[11] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated services,” RFC2475, Dec. 1998.

[12] C. Dovrolis and P. Ramanathan, “Proportional differentiated services, part II: Loss rate differentiation and packet dropping,” Proc. IWQoS 2000, pp.52–61, Pittsburgh, PA, June 2000.

[13] C. Dovrolis, D. Stiliadis, and P. Ramanathan, “Proportional differentiation service: Delay differentiation and packet scheduling,” IEEE/ACM Trans. Netw., vol.1, no.1, pp.12–26, Feb. 2002.

[14] N. Yamagaki, H. Tode, and K. Murakami, “DMFQ: Hardware design of Flow-based queue management scheme for improving the fairness,” IEICE Trans. Commun., vol.E88-B, no.4, pp.1413–1423, April 2005.

[15] D.D. Clark and W. Fang, “Explicit allocation of best-effort packet delivery service,” IEEE/ACM Trans. Netw., vol.6, no.4, pp.362–373, Aug. 1998.

[16] Design Compiler, Synopsys Inc., http://www.synopsys.com/

[17] OPNET Modeler, http://www.opnet.com/, OPNET Technologies, Inc.

[18] S. Floyd and T. Henderson, “The NewReno modification to TCP’s fast recovery algorithm,” RFC2582, Oct. 1996.

[19] W.R. Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Addison Weslay, 1994.

[20] R. Jain, A. Durresi, and G. Babic, “Throughput fairness index: An expalanation,” ATM Forum/99-0045. Feb. 1999.

[21] Smartbits, http://www.spirentcom.com/, Spirent Communications, plc.

[22] Iperf, http://dast.nlanr.net/, National Laboratory for Applied Network Research (NLANR).

（平成 20 年 4 月 25 日受付，8 月 20 日再受付）

篠原　悠介　（正員）

平 16 阪大・工・情報システム卒．平 18 同大大学院・情報科学・情報ネットワーク・博士前期課程了．同年同大学院博士後期課程入学．現在，同課程在学中．パケット廃棄制御方式，インターネットルータの構成に関する研究に従事．

戸出　英樹　（正員）

昭 63 阪大・工・通信卒．平 2 同大大学院修士課程了．平 3 年 11 月同大学院博士課程退学後，12 月より阪大・工・通信・助手．平 10 年 6 月阪大・工・情報システム・講師．平 11 年 6 月同助教授．平 14 年 4 月阪大・情報科学・情報ネットワーク・助教授．平成 19 年 4 月同准教授．平 20 年 4 月阪府大・工・電気・情報系専攻教授，現在に至る．次世代の超高速インターネットや将来の全光ネットワークを対象としたトラヒック制御，高速交換，経路選択技術，並びに，コンテンツ配信関連技術に関する研究に従事．IEEE 会員，工博．

村上　孝三　（正員）

昭 46 阪大・工・電子卒．昭 48 同大大学院修士課程了．同年富士通（株）入社．富士通研究所通信研究部門，同所マルチメディアシステム研究所を経て，平 7 阪大・大型計算機センター・教授，平 10 同大大学院・工・情報システム・教授，平 13 より同大先端科学技術共同研究センター長併任．マルチメディア情報通信システム，ネットワークコンピューティング，ホロニック光統合ネットワークの研究に従事．工博．IEEE フェロー，情報処理学会会員．