3F－2　コネクション間の公平なサービスに着目したルータにおけるパケットスケジューリングアルゴリズムの評価[1]

長谷川剛　松尾孝広　村田正幸
大阪大学大学院基礎工学研究科情報数理系専攻

1　はじめに

現在のインターネットは基本的にベストエフォート型サービスであるが、インターネットの商用化やマルチメディアアプリケーションの増加に伴い、ルータにおいて特定のコネクション(フロー)に固定的に帯域を保証したり、コネクション(フロー)毎に公平に帯域を分配するようなサービスも必要になっている。例えば、[1]では、ユーザとISP (Internet Service Provider)が、支払う料金に応じた契約を行い、それによって異なるサービス(帯域)を提供する方式を提案している。

本稿では、TCP (Transmission Control Protocol) [2]によるデータ転送サービスを対象にし、各コネクションが公平なスループットを得ることができるようなパケットスケジューリングアルゴリズムに関する評価を、コンピュータ上のシミュレーション及び解析によって行う。評価においては、FIFO (First In First Out)方式、RED (Random Early Detection)方式、及びDRR (Deficit Round Robin)方式を対象にし、コネクション間の公平性についての評価をコンピュータ上のシミュレーション及び解析的手法によって行う。また、DRR方式は公平なサービスを行えない場合があるので、DRRの各個別バッファでREDによる処理を行うDRR+方式を提案し、その有効性を評価する。

さらに、DRR+方式にはRED方式に起因する不公平性が依然として存在するので、以前の研究[3]で得られた結果を元に、まずRED方式の持つ不公平性を改善するために、ルータのバッファにおけるパケット廃棄率を各コネクションの帯域に応じて設定することを提案し、適切な廃棄率の導出を解析的に行う。また、その結果をDRR+方式に適用することによってDRR+方式の性能がさらに向上することを示す。

次に、Ill-behaved flowと呼ばれる、TCPの輻輳制御方式に従わずに、他のTCPコネクションに比べて高いスループットを得ようとするコネクションが存在する場合に、各パケットスケジューリングアルゴリズムの下でどのような影響がでるかについて検討し、Ill-behaved flowがどの程度のスループットを得ることができるのか、また他のコネクションがどの程度影響を受けるのかについての評価を行う。さらに、近年従来のTCP Renoバージョンよりも高いスループットを得られるとして注目されているTCP Vegasバージョン[4, 5]を適用した場合の評価も行う。

以下、2章では対象にしたパケットスケジューリングアルゴリズム及びTCPの各バージョンについての説明を簡単に行う。また3章ではネットワークモデルの説明、及び本稿で考える「公平なサービス」の定義を行う。次に4章で[3]で得られた各パケットスケジューリングアルゴリズムの公平性についての結果をまとめ、問題点を挙げる。次に5章ではRED方式を拡張することを提案し、その有効性を評価する。続いて6章においてTCP Vegasを適用した場合の評価を行い、7章でIll-behaved flowが与える影響についての考察を行う。最後に8章で本稿の結論及び今後の課題について述べる。

2　モデル

2.1　パケットスケジューリングアルゴリズム

本章では、本稿において検討の対象としたルータのアルゴリズム(FIFO、RED、DRR、DRR+)について簡単に説明を行う。RED、DRRについての詳細については、[6, 7]等を参照されたい。

2.1.1　FIFO (First In First Out)

FIFO方式は、実装が容易であるため現在最も広く用いられている。この方式では、ルータに到着したパケットはその到着順にバッファに格納されて、処理される。ルータのバッファ使用率が高くなり、新たに到着したパケットを格納する余裕が無い場合にはそのパケットを破棄する。この方式では、複数のコネクションが存在する場合に、各コネクションのバッファ占有率やパケット到着率とは無関係に、特定のコネクションにバースト的なパケットロスが生じる可能性がある。その結果、TCPのFast Retransmit [2]による再送が有効に働かなくなる[6]。また、特定のコネクションのパケットがバースト的に破棄された場合に、そのコネクションが、よりいっそうバースト的にパケットを送信してしまうという問題点も指摘されている[6]。

[1] Performance evaluation of packet scheduling algorithms for fair service among connections
Go Hasegawa, Takahiro Matsuo and Masayuki Murata
Osaka University
1-3, Machikaneyama, Toyonaka, Osaka 560-8531, Japan

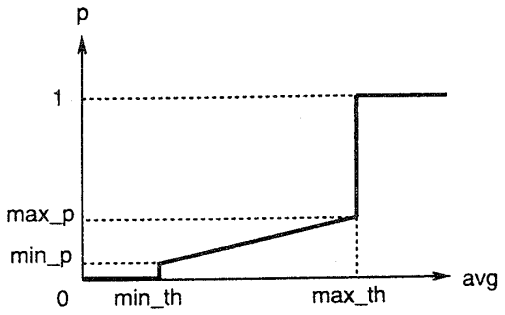


図 1: RED 方式におけるパケット破棄率 $p(x)$

2.1.2 RED (Random Early Detection) [6]

RED 方式においては、バッファでパケットがバースト的に破棄されることを防止するために、バッファに余裕がなくなる前に確率的にパケットを破棄する。具体的には、ルータにおける平均バッファ内待ちパケット数を計算し、その値によってルータに到着するパケットの処理方法を以下のように変更する。平均バッファ内待ちパケット数を avg、2 つのしきい値を min_{th}、max_{th} とすると、

- $avg < min_{th}$ の時: 到着したパケットはすべてバッファに格納され、処理される。
- $min_{th} < avg < max_{th}$ の時: 到着したパケットをある確率 $p(x)$ (x はバッファ内パケット数) で強制的に破棄する。
- $max_{th} < avg$ の時: 到着したパケットはすべて破棄する。

$p(x)$ は図 1 に示すような、バッファ内パケット数 x に対する関数である。パケットを確率的に破棄することによって、バースト的なパケットの到着に対しても、特定のコネクションのパケットがバースト的に破棄されることがなくなり、TCP がパケットをより効率的に再送することができる。また、コネクションに関係なく一定の確率でパケットを破棄するので、各コネクションの到着レートに応じてパケットが破棄される [6]。

2.1.3 DRR (Deficit Round Robin) [7]

DRR 方式は従来のラウンドロビンアルゴリズムの拡張である。ラウンドロビンアルゴリズムでは、ルータのバッファを複数に分割し (以下、個別バッファと呼ぶ)、それを各コネクションごとに別に割り当て、与えられた割合 (Quantum Size) で各個別バッファ内にあるパケットを順に処理していく (図 2)。しかし、ラウンドロビンアルゴリズムにおいては、特にパケットサイズに大きなばらつきがある場合に、各個別バッファに与えられた帯域を使い切れず、不公平が生じることがある [7]。そこで DRR 方式においては、パケット長を考慮し、使い切れなかった帯域を保存し、後で使えるようにすることによって不公平性を解消している。また、各コネクションをどの個別バッファに割り当てるかはハッシュ関数を用いて決定される。

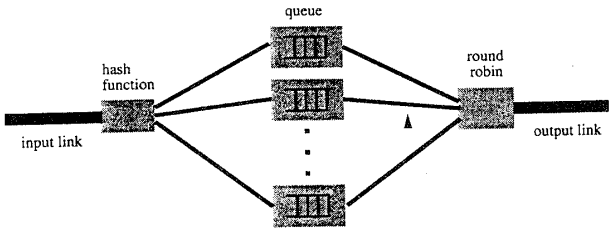


図 2: DRR 方式

2.1.4 DRR+ [3]

DRR 方式では、個別バッファがコネクション数だけ用意できずに、複数のコネクションが同一の個別バッファに収容される場合には公平性が著しく劣化することがあきらかになっている [3]。そこで [3] において提案した DRR+方式においては、DRR 方式の各個別バッファにおいて RED アルゴリズムによるパケット廃棄を行う。こうすることによって DRR 方式の持つ不公平性をある程度解消することができる。

2.2 TCP の輻輳制御方式

本章では、本稿において対象にした TCP の各バージョンの輻輳制御方式について簡単に説明する。輻輳制御方式の詳細については [2] (TCP Reno)、[4, 5] (TCP Vegas) 等を参照されたい。

TCP はウィンドウフロー制御を行っており、ウィンドウサイズ ($cwnd(t)$) をネットワークの輻輳状況に応じて動的に変更することによりネットワークへ送出するデータ量を調節している。以下では、TCP の各バージョンのウィンドウサイズ制御方式についてまとめる。

2.2.1 TCP Reno

TCP Reno は、時刻 t にセグメントを送出した後、送信側端末が時刻 $t+t_A$ に ACK セグメントを受け取ると、ウィンドウサイズを以下のように変更する [2]。

$$cwnd(t+t_A) = \begin{cases} \text{(Slow Start Phase :)} \\ cwnd(t)+1, & \text{if } cwnd(t) < ssth; \\ \text{(Congestion Avoidance Phase :)} \\ cwnd(t)+\dfrac{1}{cwnd(t)}, & \text{if } cwnd(t) \geq ssth; \end{cases} \quad (1)$$

ここで、$ssth$ [packets] は TCP が Slow Start Phase から Congestion Avoidance Phase に移行する際のしきい値で、セグメントロスが発生した時に、以下のように更新さ

れる [2]。

$$ssth = \frac{cwnd(t)}{2} \quad (2)$$

式 (1) より、ウィンドウサイズはいずれの Phase においても増加し続けることがわかる。従って最終的にはセグメントロスが必ず発生する。セグメントロスは各セグメントに設定されたタイマによって検出されるか、あるいは Duplicate ACK (同じセグメント番号が書かれた ACK セグメント) を受信することによって検出 (Fast Retransmission)[2] される。セグメントロスが検出されるとそのセグメントを再送し、ウィンドウサイズを 1 に戻して Slow Start Phase を始める。従って、ウィンドウサイズはセグメントロスをきっかけにして周期的に変動する。

さらに、TCP Reno においては、Fast Retransmission によってセグメントロスを検出した場合に、ウィンドウサイズを 1 にせずに直前のウィンドウサイズの 1/2 にする。またその後、Fast Recovery Phase と呼ばれる Phase に入り、再送セグメントに対応した ACK セグメントを受信するまでは、さらに Duplicate ACK を受信した場合に一時的にウィンドウサイズを増加させる。

2.2.2 TCP Vegas

TCP Tahoe、TCP Reno は、セグメントロスを利用して大きくなりすぎたウィンドウサイズの調整を行う。従ってセグメントロスの発生直後にウィンドウサイズが必要以上に小さくなるため、スループットが低下する。

一方 TCP Vegas は送信したセグメントの RTT (Round Trip Time) を観測し、その変動をウィンドウサイズの調整に利用する。つまり RTT が大きくなればネットワークが輻輳していると判断してウィンドウサイズを小さくし、RTT が小さくなれば逆にウィンドウサイズを増加させる。すなわち、Congestion Avoidance Phase においては、ウィンドウサイズは以下のように更新される。

$$cwnd(t+t_A) = \begin{cases} cwnd(t)+1, & \text{if } diff < \frac{\alpha}{base\_rtt} \\ cwnd(t), & \text{if } \frac{\alpha}{base\_rtt} < diff < \frac{\beta}{base\_rtt} \\ cwnd(t)-1, & \text{if } \frac{\beta}{base\_rtt} < diff \end{cases}$$

だたし、$diff = \frac{cwnd}{base\_rtt} - \frac{cwnd}{rtt}$

ここで rtt はラウンドトリップ時間、$base\_rtt$ は最小のラウンドトリップ時間、α, β は定数である。このウィンドウサイズ制御が理想的に動作すると、ウィンドウサイズは一定値に固定され、セグメントロスは発生せず、安定したスループットを得ることができる。

また、Slow Start Phase におけるウィンドウサイズの増加のスピードが TCP Tahoe、TCP Reno の 1/2 になっている。つまり、Slow Start Phase においては、以下のように $cwnd$ が更新される。

$$cwnd(t+t_A) = cwnd(t) + \frac{1}{2}$$

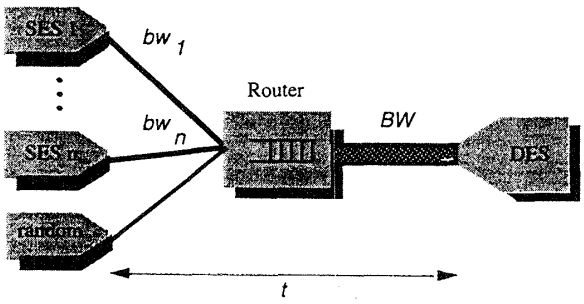


図 3: ネットワークモデル

2.3 ネットワークモデル

本論文で用いるネットワークモデルを図 3 に示す。

ネットワークモデルは n 個の送信ホスト (SES1, SES2, ..., SESn)、ランダムトラヒックを発生する送信ホスト (random)、および受信ホスト (DES)、ルータとそれらを繋ぐリンクから構成される。各送信ホストとルータとの間のリンク (入力リンク) の帯域は $bw_1, bw_2, \ldots, bw_n$ Kbps とし、ルータと受信側端末との間のリンク (出力リンク) の帯域は BW Kbps とする。また送受信ホスト間の伝搬遅延時間は対象とするルータ以外のルータでの遅延時間も含めて $t = 100$ msec とし、ルータのバッファサイズは 60 Kyte としている。

以降のシミュレーションでは、全てのコネクションが同時に TCP による通信を行う。TCP のバージョンは TCP Reno [2] を用い、送信するデータパケットのサイズは 2 Kbyte に固定している。各送信ホストは送信するデータを無限に持っていると仮定し、TCP のウィンドウサイズが許す限りデータパケットの送信を続けるものとする。また、n 本の TCP のコネクションとは別に、全体のトラヒックに対して 5% のランダムトラヒックがルータに到着する。ランダムトラヒックは TCP コネクションとは別のトラヒックとして扱う [8]。

また、以下で述べるのシミュレーション結果は、コネクション数が 4 本で、各コネクションの入力リンクの帯域がそれぞれ 64 Kbps、128 Kbps、256 Kbps、512 Kbps である。

2.4 「公平性」の定義

本稿においては、出力リンクの帯域が入力リンクの帯域の総和よりも小さい場合には、各コネクションが、自身の入力リンクの帯域比に応じて出力リンクの帯域を利用している状態を、公平なサービスが行われている状態であると定義する。すなわち、各コネクション i の

理想的なスループット ${}_{opt}\rho_i$ は、

$$ {}_{opt}\rho_i = BW \cdot \frac{bw_i}{\sum_i bw_i} \quad (3) $$

となる。例えば、図3のネットワークモデルにおいて3本のコネクションが存在し (n=3)、bw_1 = 64 Kbps, bw_2 = 128 Kbps, bw_3 = 256 Kbps、BW = 336 Kbps の場合を考える。このとき各コネクションの入力リンクの帯域比は 1 : 2 : 4 であるから、その比に応じて、各コネクションが出力リンクの帯域 (BW) を 48 Kbps、96 Kbps、192 Kbps と利用すれば公平であると考える。

また、次章以降では、以下に定義するコネクション i のスループット比 r_i を性能指標に用いて性能評価を行う。

$$ r_i = \frac{\rho_i}{bw_i} \quad (4) $$

従って、以下の式が成立する時に完全に公平なサービスが行われていると言うことができる。

$$ r_1 = r_2 = r_3 = \cdots = r_n $$

3 公平性の評価

本章では、[3] で得られた各パケットスケジューリングアルゴリズムの公平性についての結果をまとめ、その問題点をまとめる。詳細については [3] を参照されたい。図4は、FIFO、RED 及び DRR 方式におけるスループット比 (式 4) を表わしたグラフである。

3.1 FIFO 方式

FIFO 方式は実装が簡単である反面、バッファが一杯になった時にはそれ以降に到着するパケットをそのまま廃棄するため、パケット廃棄がコネクションの入力リンクの帯域に関係なく発生し、コネクション間の公平性が維持できない (図 4(a) 参照)。Identical Case においては平均スループットは公平になるものの、短時間でのスループットに注目すると不公平が発生する。また、Different Case の場合は全く公平性が維持できず、特に入力リンクの帯域の大きなコネクションはパケット廃棄がバースト的に発生するため、スループット損失が大きい。

3.2 RED 方式

RED 方式においてはルータのバッファにおけるパケット廃棄がコネクションの入力リンクの帯域にほぼ比例して発生するため、Different Case における公平性は FIFO 方式よりも優れている。しかし、スループット比が完全に等しくはならず、特に出力リンクの帯域が小さい時には各コネクションのスループットが等しい値に近づき、公平性が劣化する (図 4(b) 参照)。これは、RED による確率的なパケット廃棄の下では TCP のウィンドウ制御がスループットが公平になるようには働かないためである。このことは [3] において解析的手法によって明らかにした。

3.3 DRR 方式

DRR 方式においては、コネクション毎に個別バッファを用意して、入力リンクの帯域に比例して処理が行われるため、Difference Case においても公平性は非常に高い (図 4(c) 参照)。しかし、バッファサイズの制限や、コネクション数の増大により個別バッファをコネクション毎に用意できない場合には、公平性は大きく劣化する。これは、DRR 方式における各個別バッファ内では到着パケットが FIFO によって処理されるためである。従って、個別バッファ単位での公平性は維持できるが、一つの個別バッファに割り合てられた複数のコネクション間の公平性は全く維持できない。

3.4 DRR+方式

DRR+方式では各個別バッファにおいて RED によるパケット廃棄を行うので、個別バッファ内の公平性も DRR+方式に比べて若干向上する。しかし、3.2 章で述べたように RED 方式では完全に公平なサービスはできないので、DRR+方式においても完全に公平にはならない。したがって、さらに公平性を向上させる方式が必要である。

4 RED 方式の拡張

4.1 提案方式

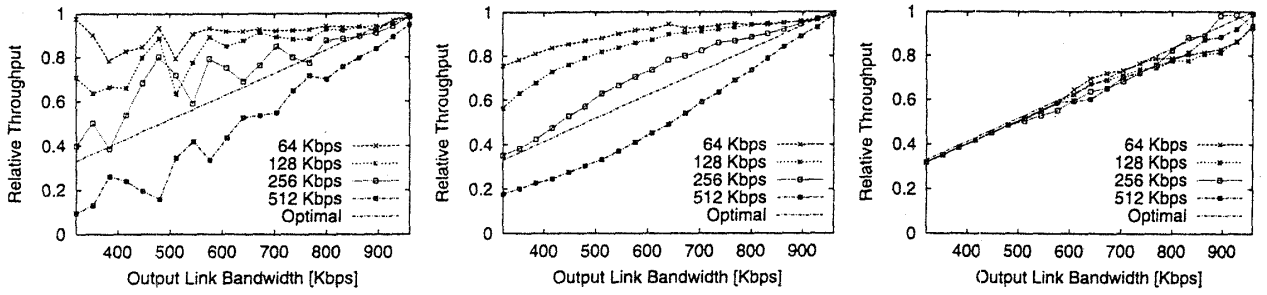
[3] 及び 3 章において指摘したように、RED によるパケット棄却方式の下では、TCP によるウィンドウ制御ではスループットが公平にならないことがわかった。これは、各コネクションは異なる入力リンクの帯域を持つため、それぞれのコネクションの帯域遅延積 (bandwidth-delay product) が異なるにもかかわらず、すべてのコネクションのパケットを同じ廃棄率 p で廃棄するため、平均ウィンドウサイズがコネクションの帯域遅延積にかかわらず同じ値になるためである。

そこで本稿では、スループットが公平になるように RED のパケット廃棄率 p をコネクションの入力リンクの帯域に応じて変化させることを考える。そのために、[3] において示した RED のスループット解析の結果を利用する。

解析においては、各コネクションの入力リンクの帯域 $bw_1, \ldots, bw_N$、出力リンクの帯域 BW、RED のパケット廃棄率 p が与えられた時に各コネクション i のスループット ρ_i を導出した。ここではその解析結果を利用して、ρ_i が bw_i に比例し、スループット比が等しくなるように各コネクション i のパケット廃棄率 p_i を設定する。

4.2 パケット廃棄率の導出方法

各コネクション i の適切な導出方法を以下に示す。



(a) FIFO Case.　(b) RED Case.　(c) DRR Case.

図 4: Relative Throughput with TCP Reno

1. すべてのコネクション i のパケット廃棄率 p_i を初期化する。
2. 現在のパケット廃棄率 p_i を利用した場合の各コネクション i のスループット ρ_i を解析結果を用いて導出する。
3. スループット比が等しくなれば現在のパケット廃棄率 p_i を最終的な値に設定し終了する。
4. そうでない場合は各コネクション i の理想的なスループット (3 式) と比較し、差が最も大きなコネクションのパケット廃棄率を以下のように変更し、2. へ戻る。
   - スループットが理想値よりも大きい場合にはパケット廃棄率を 1/10 だけ大きくする。
   - スループットが理想値よりも小さい場合にはパケット廃棄率を 1/10 だけ小さくする。

4.3 性能評価

本節では、4.2 節で提案した方法の効果を、シミュレーションによって評価する。以下のシミュレーションにおいてはシミュレーション時間を 2000 秒とする。

図 5(b) は Different Case (4 本のコネクションの入力リンクの帯域がそれぞれ 64 Kbps、128 Kbps、256 Kbps、512 Kbps) において、各コネクションのパケット廃棄率を解析結果に基いて設定した時 (Enhanced RED と呼ぶ) の、出力リンクの帯域を各コネクションのスループット比の関係を示している。図から、全てのコネクションで同じパケット廃棄率を使用した場合 (Original RED と呼ぶ。図 5(a)) に比べて公平性が大きく向上していることがわかる。このことから、提案したパケット廃棄率の導出方法が有効であることがわかる。

しかし、RED 方式において収容するコネクション数が大きくなると、その効果が小さくなってくる。図 5(b) は、コネクションが 8 本 (入力リンクの帯域が 64 Kbps、128 Kbps、256 Kbps、512 Kbps のコネクションがそれぞれ 2 本ずつ) の場合の結果を示している。図から、コネクション数が大きくなると公平性が劣化していることがわかる。これは、コネクション数が大きくなったために、解析では想定していないラウンドトリップタイムの変動が大きくなるためであると考えられる。

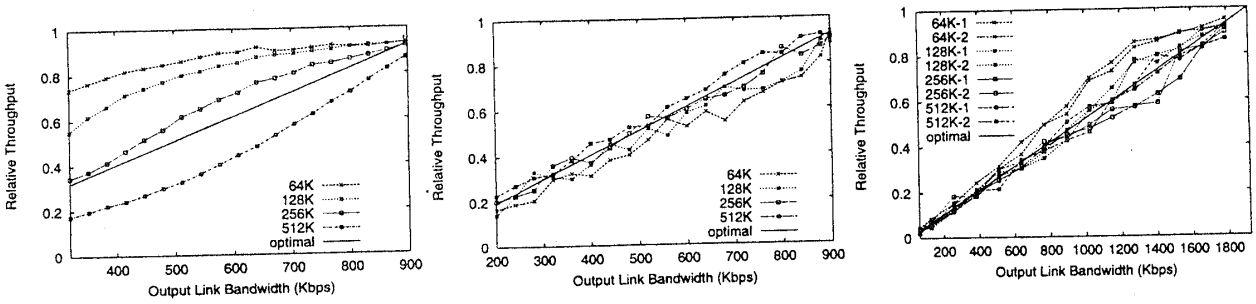
従って、提案したパケット廃棄率の設定方法を効果的に適用するためには、適用するコネクション数を小さくする必要がある。そのための一提案として、次節では、DRR+方式に Enhanced RED を適用することを考える。

4.4 DRR+への適用

DRR+方式は、RED 方式と比べて各個別バッファに収容されるコネクション数が小さいため、前節までに提案した Enhanced RED が効果的に適用できると考えられる。本節では、DRR+の各個別バッファに Enhanced RED を適用した場合の評価を行う。ここでは、2 つの個別バッファを用意し、入力リンクの帯域が 64 Kbps、128 Kbps のコネクションを 1 つの個別バッファに、256 Kbps、512 Kbps のコネクションをもう 1 つの個別バッファに収容した場合を考える。

図 6(a) 及び 6(b) は、Original RED を適用した DRR+方式及び Enhanced RED を適用した DRR+方式の、出力リンクの帯域と各コネクションのスループット比の関係を示している。これらの図から、提案した方式によって DRR+方式の公平性が大幅に向上していることがわかる。

図 5(c) はコネクション数を 8 (入力リンクの帯域が 64 Kbps、128 Kbps、256 Kbps、512 Kbps のコネクションがそれぞれ 2 本ずつ) にし、2 つの個別バッファに 4 本ずつ (64 Kbps、128 Kbps、256 Kbps、512 Kbps のコネクションをそれぞれ 1 本ずつ) 収容した場合の結果を示している。この図と、コネクション数が大きくなり、RED 方式で公平性が劣化する場合 (4.3 節、図 5(c)) とを比較することにより、DRR を適用し、1 つの個別バッファあたりのコネクション数を減少させて Enhanced RED を適用することによって、公平性が向上することがわかる。

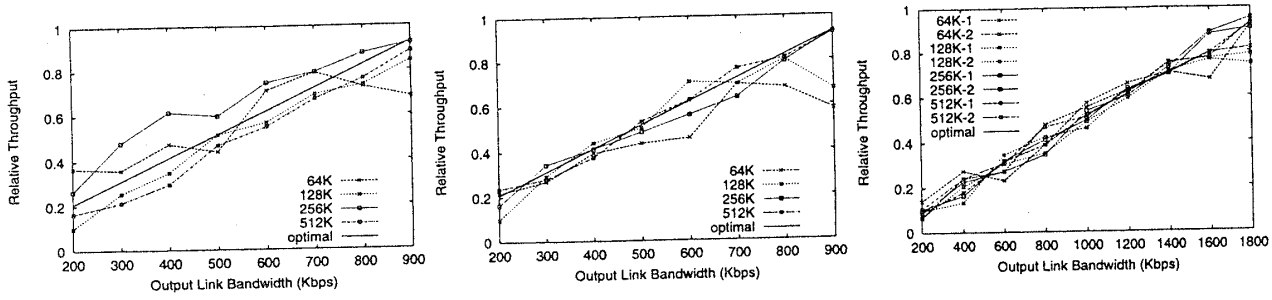


(a) Original RED.

(b) Enhanced RED (4 connections)

(c) Enhanced RED (8 connections)

図 5: RED の拡張方式の効果



(a) Original DRR+.

(b) DRR+に Enhanced RED を適用した場合 (4 connections)

(c) DRR+に Enhanced RED を適用した場合 (8 connections)

図 6: DRR+方式への Enhanced RED の適用

5 TCP Vegas での評価

本章では、TCP Vegas バージョンを使ってデータ転送を行った場合の評価を行い、TCP Vegas の有効性を検証する。

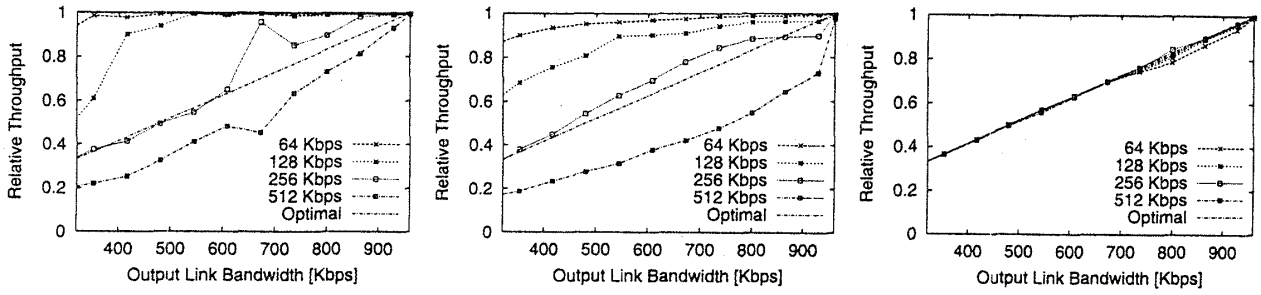
図 7(a) は、FIFO 方式の場合のシミュレーション結果である。TCP Reno の場合 (図 4(a)) と比べて不公平性が増大している。特に、入力リンクの帯域の小さなコネクションが非常に有利になっている。これは、TCP Vegas においては安定時にはウィンドウサイズが固定されるが、TCP Vegas がウィンドウサイズの制御 (式 (3)) に使う *base_rtt* (最小のラウンドトリップ時間) が全てのコネクションで等しい値になり、ウィンドウサイズが各コネクションの入力リンクの帯域に比例した値にならないためである。

この不公平性は RED 方式 (図 7(b)) においても見られ、TCP Reno の場合 (図 4(b)) と比べて不公平性が大きくなっている。そこで、TCP Vegas を用いた場合の各コネクションのスループットを解析的に導出することにより、RED 方式の不公平性を検証した [9]。その結果、RED 方式による確率的なパケット廃棄は、4 章で述べた RED 方式の拡張を行っても、本質的に TCP Vegas の輻輳制御方式では公平性を維持できないことがわかった。これは、TCP Vegas は理想的にはパケットロスが発生しないように動作するので、パケットを早期の輻輳の段階から強制的に廃棄する RED 方式の下では効率的に動作しないためである。

一方、DRR 方式を適用した場合は (図 7(b))、TCP Reno の場合よりも高い公平性を維持することができる。DRR 方式においては帯域をコネクション毎に分配して割り当てているものの、パケットロスに伴うスループット低下がコネクションによって異なるため、本質的にパケットロスを回避できない TCP Reno においては若干の公平性の劣化がある (図 4(b) 参照)。一方 TCP Vegas を用いると、理想的に動作した場合にはパケットロスが発生しないため、DRR 方式によって分配された帯域を完全に利用できる。従って、ルータにおいて DRR 方式を用いる場合には TCP Vegas は非常に有効であるといえる。

6 Ill-behaved flow の影響

本章では、Ill-behaved flow と呼ばれる、通常のコネクションよりも高いスループットを得るために、輻輳制御方式を改変してデータ転送を行うようなコネクションが存在する場合の影響についての考察を行う。本稿では、TCP の輻輳制御方式 (ウィンドウ制御) を変更することによって Ill-behaved flow を作りだす場合を考える。



(a) FIFO Case.　(b) RED Case.　(c) DRR Case.

図 7: Relative Throughput with TCP Vegas

6.1 Ill-behaved flow の種類

TCP のウィンドウ制御方式 [2] を変更することによって高いスループットを得ようとする場合には、以下のような改変方法が考えられる。

1. パケットロスが発生してもウィンドウサイズを小さくせずに、最大値 (通常 64 Kbyte) に固定する。
2. タイムアウトが発生してもスロースタート (Slow Start) を行わない。
3. 輻輳回避フェーズ (Congestion Avoidance Phase) におけるウィンドウサイズの増加率を上げる。
4. スロースタートフェーズから輻輳回避フェーズへ移行する時のしきい値 (ssthreth) を大きくする。

次節では、これらのうち最も影響の大きかった 1. の場合についての評価結果を示す。

6.2 Ill-behaved flow の影響

本節における評価においては、Identical Case (4 本のコネクションの入力リンクの帯域が全て 64 Kbps) で、4 本中 1 本が Ill-behaved flow である場合を考える。

図 8 は、FIFO、RED、DRR、DRR+方式における、出力リンクの帯域と各コネクションのスループット比の関係を示したグラフである。FIFO 方式 (図 8(a)) 及び RED 方式 (図 8(b)) においては、Ill-behaved flow が他のコネクションに比べて高いスループットを得ている。これは、FIFO 方式及び RED 方式においては全てのコネクションのパケットが一つのバッファに収容されるため、他に比べて多くのパケットを送信しようとする Ill-behaved flow が高いスループットを得ており、その反面他のコネクションのスループットが低下する。

一方、DRR 方式 (図 8(c)) においては、逆に Ill-behaved flow のスループットが他に比べて低下しており、他のコネクションのスループットは低下していない。これは、DRR 方式においては各コネクション毎に個別バッファが用意されるために、Ill-behaved flow のパケット廃棄数が非常に増加し、スループットが下がるためである。他のコネクションのパケットも個別に処理されるために、Ill-behaved flow の影響を受けて他のコネクションのスループットが低下することはない。従って、TCP の輻輳制御方式を変更することによって高いスループットを得ようとする Ill-behaved flow に対して DRR 方式は非常に有効であるといえる。

また、DRR+方式 (図 8(d)) においては、DRR 方式に比べて Ill-behaved flow のスループット低下が若干抑えられてしまっている。これは、DRR 方式においては各バッファは FIFO によるパケット廃棄が行われるために、バースト的なパケットロスが発生してスループットが大きく低下するのに対して、DRR+方式においてはパケットを確率的に廃棄するので、バースト的なパケット廃棄が発生せず、TCP のスループットが大きくは低下しないためであると考えられる。ただし、Ill-behaved flow 以外のコネクションに対して悪影響を及ぼすまでには至っていない。

7 まとめと今後の課題

本稿では、[3] において得られた各パケットスケジューリングアルゴリズムの公平性についての結果を基に、公平性を向上させるために RED 方式におけるパケット廃棄率をコネクション毎に設定することを提案し、その有効性をシミュレーションによって評価した。その結果、提案方式によって RED 方式のコネクション間の公平性が大幅に向上することが明らかになった。さらに、DRR+方式に提案方式を適用することによって、DRR+方式の公平性が改善されることもわかった。

また、TCP Vegas バージョンを適用した場合についての評価も行い、FIFO 及び RED 方式の下では TCP Reno バージョンよりも公平性が劣化すること、及び DRR 方式の下では非常に高い公平性を示すことが明らかになった。

さらに、他のコネクションに比べて不当に高いスループットを得ようとする Ill-behaved flow が存在する場合の影響についての考察を行った。その結果、FIFO 方式及び RED 方式においては Ill-behaved flow が高いスルー

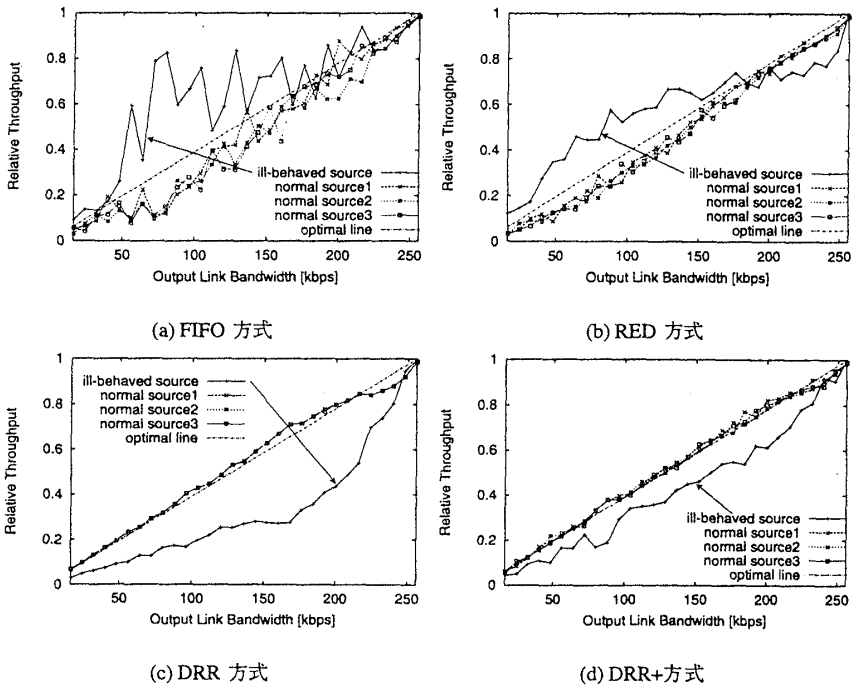


図 8: Ill-behaved flow の影響

プットを得ることができ、他のコネクションがその影響を受けてスループットが低下するのに対し、DRR 方式では逆に Ill-behaved flow のスループットが低下してしまう場合があることが明らかになった。また、[3] で提案した DRR+方式は DRR 方式が Ill-behaved source に対して持つ性質を損なうことがないことを示した。

本稿で提案した Enhanced RED 方式は、収容するコネクション数が大きくなった時に公平性が劣化するので、それを防止する方式を今後提案していく予定である。また、本稿で対象にした種類以外の Ill-behaved flow に関する評価や、Ill-behaved flow の本数が変化した場合の評価、及び Different Case における Ill-behaved flow の影響についての評価を行う予定である。

謝辞

本研究の一部は，日本学術振興会未来開拓学術研究推進事業における研究プロジェクト「高度マルチメディア応用システム構築のための先進的ネットワークアーキテクチャの研究」によっている．ここに記して謝意を表す．

参考文献

[1] Z. Wang, "Toward scalable bandwidth allocation on the internet," *On The Internet*, pp. 24–32, May/June 1998.

[2] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

[3] T. Matsuo, G. Hasegawa, M. Murata, and H. Miyahara, "Comparisons of packet scheduling algorithms for fair service among connections," *Proceedings of IWS'99*, pp. 193–200, February 1999.

[4] L. S. Brakmo, S. W.O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," *Proceedings of ACM SIGCOMM'94*, pp. 24–35, October 1994.

[5] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE Jounal on Selected Areas in Communications*, vol. 13, pp. 1465–1480, October 1995.

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, August 1993.

[7] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 375–385, June 1996.

[8] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience*, vol. 3, pp. 397–413, August 1992.

[9] G. Hasegawa, T. Matsuo, M. Murata, and H. Miyahara, "Comparisons of packet scheduling algorithms for fair service among connections on the internet," Submitting to *IEEE INFOCOM'00*, June 1999.