TCP バージョン間の不公平性を解消するパケット廃棄方式

倉田 謙二† 長谷川 剛‡ 村田 正幸‡

† 大阪大学大学院基礎工学研究科情報数理系
〒 560-8531 大阪府豊中市待兼山 1-3
Phone: 06-6850-6616, Fax: 06-6850-6589
E-mail: k-kurata@ics.es.osaka-u.ac.jp

‡ 大阪大学サイバーメディアセンター
〒 560-0043 大阪府豊中市待兼山町 1-30
E-mail: {hasegawa, murata}@cmc.osaka-u.ac.jp

あらまし 本稿では、ルータにおいてフロー間の公平なサービスを実現するための方式として、ZL-RED (Zombie Listed RED) を提案する。ZL-RED は、SRED において提案されている Zombie List を用いて、他のフローに比べてパケット到着率の高いフロー (mis-behaving フロー) を検出し、そのフローのパケット廃棄率を高く設定することで、フロー間の公平性を向上させる。さらに、シミュレーションによる ZL-RED の性能評価を行い、TCP のバージョン間の不公平性を大きく改善できることを示す。また、ルータに収容するコネクションが多い場合でも、SRED よりも高い公平性を維持できることを明らかにする。

キーワード TCP (Transmission Control Protocol)、TCP Reno、TCP Vegas、公平性、RED (Random Early Detection)、SRED (Stabilized RED)

More Results on ZL-RED for fairness enhancement between TCP Reno and Vegas

Kenji Kurata† Go Hasegawa ‡ Masayuki Murata‡

†Department of Infomatics and Mathematical Science
Graduate School of Engineering Science, Osaka University
1-3, Machikaneyama, Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-6850-6616, Fax: +81-6-6850-6589
E-mail: k-kurata@ics.es.osaka-u.ac.jp

‡Cybermedia Center, Osaka University
Toyonaka, Osaka 560-0043, Japan
E-mail: {hasegawa, murata}@cmc.osaka-u.ac.jp

Abstract In this paper, we propose ZL-RED (Zombie Listed RED) algorithm, which ehhances SRED algorithm to provide better fairness among many flows at the router buffer. ZL-RED uses the *Zombie List*, which is also used by SRED, to detect mis-behaving flows which send packets to the router at higher rate than the other flows, and sets larger packet discarding probability to those flows. We evaluate the effectiveness of ZL-RED by simulation experiments, and show that ZL-RED can improve fairness among TCP Reno connections and TCP Vegas connections, and that it can provide higher performance than SRED, even when the number of accommodated connections is large.

Keyword TCP (Transmission Control Protocol), TCP Reno, TCP Vegas, Fairness, RED (Random Early Detection), SRED (Stabilized RED)

1 はじめに

近年、インターネットユーザーは爆発的に増加しており、それらのユーザーが利用するアプリケーションも、電子メールや WWW アクセス、ファイル転送といったデータ系のアプリケーションだけでなく、音声や映像のストリーミング、オンラインゲームといったリアルタイムアプリケーションというように多様化している。

しかし、これまでのインターネットはデータ系アプリケーションのみを対象としているため、様々な問題が発生しつつある。例えば、リアルタイムアプリケーションが主に用いるトランスポート層プロトコルである UDP (User Datagram Protocol) のコネクションと、データ系アプリケーションが用いる TCP (Transmission Control Protocol) のコネクションが同一パス上に存在すると、TCP コネクションのスループットが低下し、UDP コネクションとの間に不公平が生じることがこれまでにわかっている [1]。

このような不公平性は、TCP のコネクション間にも存在する。現在のインターネットにおいては、TCP Reno と呼ばれるバージョンが最も一般的に用いられている。TCP Reno は、Fast Retransmit や Fast Recovery といったさまざまな機能 [2] を持つが、さらに高い性能を得るための高機能化を図った新しいバージョンの TCP に関する研究も行われている。その中で、1995 年に提案された TCP Vegas バージョン [3] が、高スループットを得られるとして注目されている。TCP Vegas は、パケットロスによってのみネットワークの輻輳を検出する従来の TCP Tahoe/Reno バージョンとは異なり、パケットの RTT (Round Trip Time) を観測してネットワークの輻輳を検出し、ウィンドウサイズを増減させる。

これまでの研究で、TCP Vegas は、ネットワーク内に TCP Vegas コネクションのみが存在する環境においては非常に高い性能が得られることがわかっている [4, 5, 6]。しかし、TCP Vegas コネクションと従来の TCP Tahoe/Reno コネクションがネットワーク内に混在する環境においては、TCP Vegas コネクションのスループットが低下することが指摘されている [7]。そこで我々のこれまでの研究 [8, 9] において、TCP Reno コネクションと TCP Vegas コネクションの間の公平性を解析的手法、及びシミュレーションによって評価し、TCP のバージョン間の不公平性は、双方の輻輳制御方式が持つ、ネットワーク内のルータのバッファの利用方針の本質的な違いが原因であることを指摘した。また、ルータにおいて Drop-Tail バッファ用いた場合には、不公平性が非常に大きいことを示した。

これらの不公平性を解決する方法の一つとして、ルータのバッファにおいて RED (Random Early Detection) [10] を用いることが挙げられる。RED は、輻輳の初期段階から積極的にパケットを確率的に廃棄することにより、バッファ溢れを抑制する。[11] において、RED を用いることによって TCP Reno コネクションのウィンドウサイズの不当な増大を抑え、TCP Vegas コネクションとの間の不公平性をある程度解消できることを示した。しかし RED には、コネクション数の増加に応じてキュー長が増大するという問題や、上述の UDP コネクションと TCP コネクションの間の不公平性を解決することができない問題があることが指摘されている [12]。

そこで、[12] では RED による確率的なパケット処理を利用し、コネクション数を見積り、そのコネクション数に応じてパケット廃棄率を変化させることでキュー長を安定化させる SRED (Stablized RED) が提案されている。また、SRED が持つ機能を利用することにより、コネクション間の不公平性を解消できることも指摘されているが、具体的な方法は提案されていない。

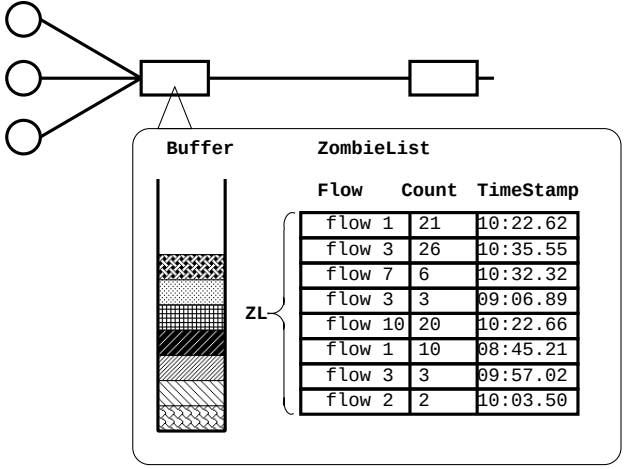


図 1: Zombie List

そこで本稿では、SRED を利用し、コネクション間の不公平性を改善する方式として ZL-RED (Zombie Listed RED) を提案する。ZL-RED では、[12] で提案された Zombie List のメカニズムを利用することにより misbehaving フロー、すなわち他のフローより多くのパケットを送信しているフローをルータで検出し、そのフローのパケットをより高い確率で廃棄する。また ZL-RED によって、TCP のバージョン間の公平性を向上できることを、シミュレーションによって示す。

以下、2 章、3 章ではそれぞれ SRED、ZL-RED について述べる。次に、シミュレーションによって SRED と ZL-RED の性能評価を 4 章で行い、ZL-RED の有効性を検証する。最後に 5 章でまとめと今後の課題を述べる。

2 SRED: Stabilized RED

本章では、まず SRED のアルゴリズムについて簡単に説明し、次にその特徴を挙げ、改善すべき点が存在することを指摘する。SRED の詳しいアルゴリズムについては [12] を参照されたい。

2.1 アルゴリズム

SRED は、ルータで Zombie List と呼ばれる固定サイズのテーブルを保持する (図 1 参照)。このテーブルは、ルータにパケットが到着する毎に、到着パケットに関するエントリを書き込むことによって順次更新される。

Zombie List の各エントリ (Zombie と呼ぶ) は、(フロー ID、カウンタ、タイムスタンプ) の組で構成される。パケットがルータに到着したときに、Zombie List のエントリに空きがあれば、そこへ到着パケットの情報から新たな Zombie を構成し Zombie List のエントリとして追加する。Zombie List に空きがなければ、Zombie List からランダムに Zombie を 1 つ選び、その Zombie のフロー ID と到着パケットのフロー ID とを比較する。もしフロー ID が一致すれば (これを「ヒット」と呼ぶ)、その Zombie のカウンタを増加させる。ヒットしなかった場合は、確率 p_{swap} (定数) でその Zombie を到着パケットのフロー ID で上書きし、カウンタを 0 に初期化する。

次に、以下の式を用いることにより、キュー長に応じてパケット廃棄率を決定する。

$$p_1 = \begin{cases} 0 & if \quad 0 \le q_{len} < \frac{1}{6}B \\ \frac{1}{4} \times p_{max} & if \quad \frac{1}{6}B \le q_{len} < \frac{1}{3}B \\ p_{max} & if \quad \frac{1}{3}B \le q_{len} \end{cases} \tag{1}$$

ここで、B、q_{len} はそれぞれルータのバッファサイズ、パケット到着時のルータのバッファ内のパケット数を表しており、p_{max}、th_{min} [packets] は RED と同様の定数である。また、$P(t)$ を到着パケットがヒットする確率の平均値 (平均ヒット率) とする。このとき、$P(t)$ の逆数 $1/P(t)$ は、各コネクションのパケット到着レートが等しければ、ルータ内に存在するアクティブフロー数にほぼ等しくなる [12]。

次に、フロー数の推測値 $1/P(t)$ を用い、フロー数が少ない時はパケット廃棄率を小さくし、フロー数が多い時はパケット廃棄率を大きくする。

$$p_2 = p_1 \times min\left(1, \frac{1}{(256 \times P(t))^2}\right) \tag{2}$$

最後に、以下の式を用いて、到着パケットがヒットしたか否かにより、パケット廃棄率を変化させる。

$$p_3 = p_2 \times \left(1 + \frac{Hit(t)}{P(t)}\right) \tag{3}$$

ここで、$Hit(t)$ は

$$Hit(t) = \begin{cases} 1 & if \quad \text{ヒットした場合} \\ 0 & if \quad \text{ヒットしなかった場合} \end{cases}$$

である。

2.2 特徴

SRED は、式 (2) を用いることによって、フロー数が少ない時にはパケット廃棄率を小さくし、フロー数が多い時にはパケット廃棄率を大きくしている。これにより、フロー数に関係なくバッファ内パケット数を安定させることができる [12]。

また、平均ヒット率 $P(t)$ の逆数 $1/P(t)$ によって、アクティブフロー数を推測することができる。式 (2) ではこの性質を利用し、フロー数に応じてパケット廃棄率を変化させている。しかし、特にフロー数が多くなると、推測が不正確になる。図 2 は、図 3 のネットワークモデルを用いて、10〜1280 本の TCP Reno コネクションが同時にデータ転送を行ったときの実コネクション数と推測コネクション数 ($1/P(t)$) の関係を表わしている。図中の ZL は Zombie List のサイズ (エントリ数) を示す。この図から、フロー数が大きくなると、推測フロー数が実フロー数よりも小さくなることがわかる。これは、フロー数が大きくなることにより、各フローのパケット到着レートがばらつき、フロー数の推測に大きな影響を与えるためである [12]。

さらに SRED は、式 (3) を用いて、ヒットしたパケットの廃棄率を大きくする。このことにより、フロー間の公平性が向上することが考えられる。これは、パケット到着レートの高いフローほど、Zombie List 内に多くのエントリを含むため、そのフローの到着パケットがヒットする確率が高くなるためである。しかし、[12] で示されているように、SRED では公平性を大幅に向上することはできない。そこで、次章で説明する提案方式 (ZL-RED) では、Zombie List のエントリ内に含まれる情報をより有効に利用することにより、mis-behaving フローを検出し、パケット廃棄率を大きく設定することにより、フロー間の公平性を向上させる。

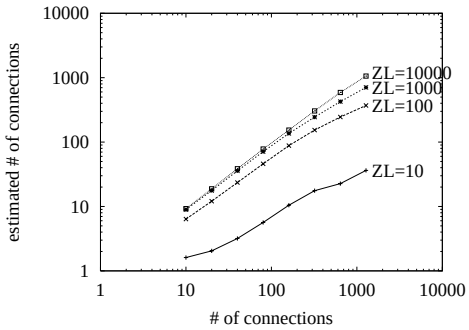


図 2: 実コネクション数と推定コネクション数の関係

フロー間の公平性を向上させるための方式としては、DRR (Deficit Round Robin) [13] のように、フロー毎にキューを用意して到着パケットを処理する方式や、FRED (Flow Random Early Drop) [14] のように、各フローの情報をルータで保持しておき、その情報を用いてパケットの処理を行う方式などがある。これらの方式を用いると非常に高い公平性を得ることができるが、フロー数に比例した処理を必要とするため、フロー数が多くなると、十分な実行速度が得られない可能性がある。

それに対して SRED では、フローごとの情報を持たず、固定サイズの Zombie List を用いるため、フロー数の増加に対してその処理が大きくなることはない。

3 ZL-RED: Zombie Listed RED

本章では、SRED を拡張し、2 章で述べた SRED の持つ特長を維持しながら、SRED では改善することのできないフロー間の公平性を向上させる方式として、ZL-RED (Zombie Listed RED) を提案する。ZL-RED は、他のフローよりも高いレートでパケットを送出している mis-behaving フローを検出し、そのフローのパケットの廃棄率をより高く設定することより、公平性を向上させる。

ZL-RED では、SRED が持つ Zombie List はそのまま利用し、まず基本となるパケット廃棄率を式 (1)-(3) を用いて求める。次に、Zombie List 内の全ての Zombie のカウント値の和 T と、到着パケットのフローに相当する Zombie のカウント値の和 F を求める。そして、以下の式が成立する場合に、到着パケットのフローは mis-behaving フローであると判断する。

$$F > T \times P(t) \tag{4}$$

式 (4) の右辺 $T \times P(t)$ は、Zombie List 内の各フローのカウント値の平均値を表しているため、式 (4) が成立することは、そのフローのパケット到着レートが、平均のパケット到着レートより高いことを表す。従って、式 (4) を用いることによって、mis-behaving フローを検出できると考えられる。mis-behaving フローだと判断された場合には、以下の式を用いてパケット廃棄率を増加させる。

$$p_4 = p_3 \times \frac{F}{T \times P(t)} \tag{5}$$

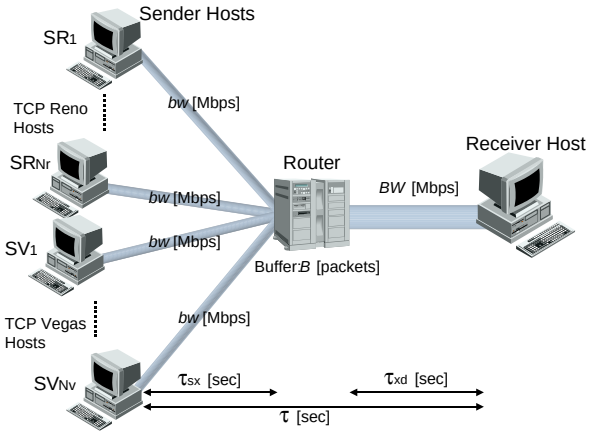


図 3: ネットワークモデル

最後に、全体のパケット廃棄率が、SRED のそれ (式 (3)) に等しくなるように、パケット廃棄率を以下のように調整する。これは、式 (5) によって mis-behaving フローのパケット廃棄率が変化すると、全体のパケット廃棄率も変動してしまい、SRED の特長の一つであるバッファ内パケット数の安定性が失われるためである。

$$p_5 = p_4 \times \frac{p_3}{P_{avg}(t)} \tag{6}$$

ここで、$P_{avg}(t)$ は、これまでの平均パケット廃棄率である。以上のアルゴリズムにより、平均パケット廃棄率を SRED と同等に維持しながら、mis-behaving フローのパケットをより高い廃棄率で廃棄することができるため、フロー間の公平性が向上すると考えられる。

また、SRED では、パケットの確率的廃棄を開始するバッファ内パケット数の閾値は、バッファサイズに比例した値となっている (式 (1))。バッファ内パケット数がこの閾値に達するまでは到着パケットは完全な FIFO 規律によって処理されるため、バッファサイズ B が大きく、閾値が大きい場合には、FIFO 規律が原因で公平性が著しく低下する [9]。従って、ZL-RED は、式 (7) に示すようにパケットの確率的廃棄を始める閾値をバッファサイズに依存しない定数 th_{min} とする。

$$p'_1 = \begin{cases} 0 & if \quad 0 \le q_{len} < th_{min} \\ \frac{1}{4} \times p_{max} & if \quad th_{min} \le q_{len} < \frac{1}{3}B \\ p_{max} & if \quad \frac{1}{3}B \le q_{len} \end{cases} \tag{7}$$

4 数値例

本章では、3 章で提案した ZL-RED を用いたシミュレーション結果を示し、ZL-RED の有効性の評価を行う。

4.1 ネットワークモデル

本章で用いるネットワークモデルを図 3 に示す。モデルは TCP Reno を用いる送信側端末 (SR_1, ... SR_{Nr})、TCP Vegas を用いる送信側端末 (SV_1, ... SV_{Nv})、受信側端末、ボトルネックルータ、及びそれらを接続するリンクからなる。送信側端末とルータとの間のリンクの帯域はすべて等しく $bw = 45$ [Mbps]、ルータと受信側端末とのボトルネックリンクの帯域を $BW = 45$ [Mbps]、ボトルネックルータのバッファサイズを B [packets]、送信側端末とボトルネックルータとの間の伝搬遅延時間を $\tau_{sx} = 20$ [msec]、ボトルネックルータと受信側端末との間の伝搬遅延時間を $\tau_{xd} = 1$ [msec] とし、$\tau = \tau_{sx} + \tau_{xd}$ とする。また、ルータのバッファにおけるパケット処理規律として、2 章で説明した SRED と、本稿で提案する ZL-RED を用いる。また、シミュレーション時間を 100 [sec] とし、SRED、ZL-RED 共通のパラメータとして $p_{max} = 0.15$、ZL-RED のパラメータとして $th_{min} = 5$ [packets]、$p_{swap} = 0.25$ とする。

シミュレーションにおいては、送信側端末がデータを送信したときの平均スループット、及びパケットロス率の比較を行い、フロー間の公平性に関する評価を行う。

4.2 TCP のバージョン間の公平性

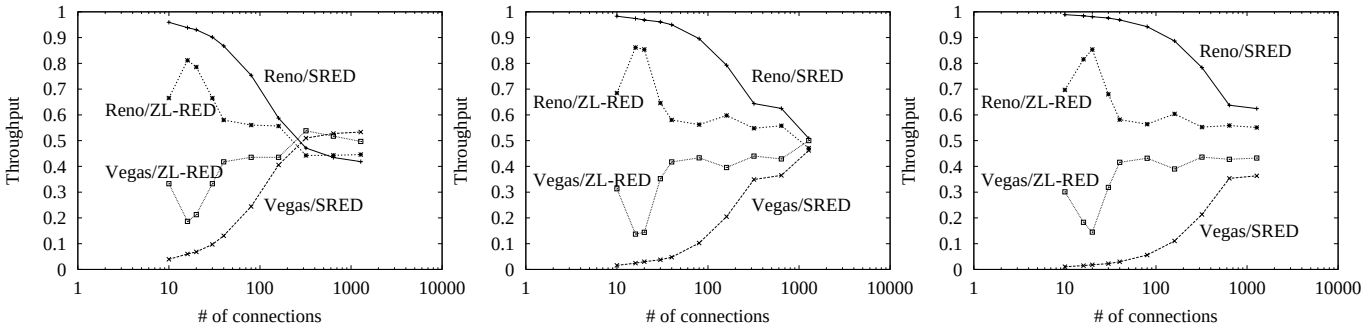
本節では、送信側端末 SR_1, ... SR_{Nr} は TCP Reno を用い、送信側端末 SV_1, ... SV_{Nv} は TCP Vegas を用いた場合の評価結果を示す。

図 4(a)、4(b) 及び 4(c) は、TCP Reno コネクション、TCP Vegas コネクションのそれぞれのコネクション数と、TCP Reno/Vegas コネクションの平均スループットの関係を示している。また、図 4(d)、4(e) 及び 4(f) はパケットロス率を示している。ここで、SRED 及び ZL-RED の Zombie List のサイズ (エントリ数) は 1000 エントリとし、ルータのバッファサイズをそれぞれ 1000 [packets] (図 4(a)、4(d))、5000 [packets] (図 4(b)、4(e))、10000 [packets] (図 4(c)、4(f)) とした場合の結果を示している。

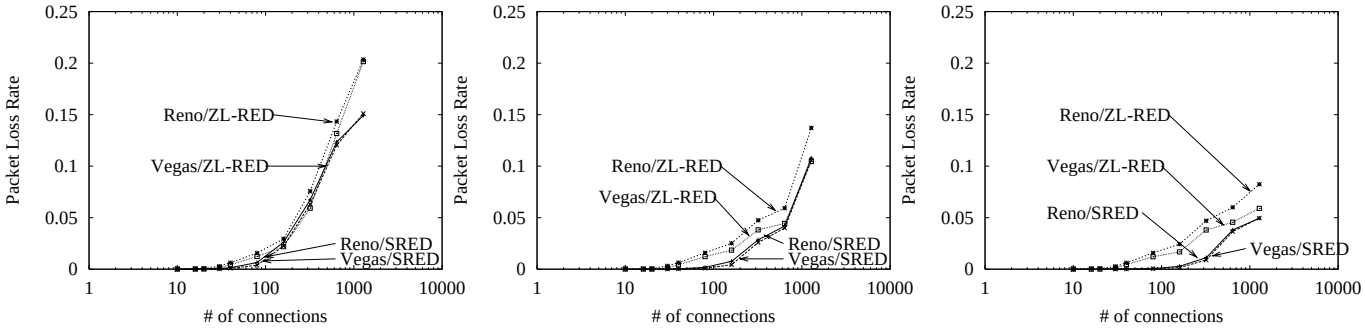
これらの図から、SRED 方式ではルータのバッファサイズに関係なく TCP バージョン間の公平性を維持することができず、特にバッファサイズが大きくなると、公平性が悪化することがわかる。これは、3 章で述べたように、SRED においては確率的なパケット廃棄を始めるバッファ内パケット数の閾値 th_{min} が、バッファサイズに比例しているため、バッファサイズが大きくなると、同時に閾値も大きくなり、th_{min} までのバッファ領域は完全な FIFO 規律で処理されるためである。FIFO 規律における TCP のバージョン間の不公平性は、TCP Reno と TCP Vegas の輻輳制御方式の違いが原因で発生することが [9] で示されている。

また、ZL-RED の場合は、バッファサイズに関係なく、公平性が改善されており、特にコネクション数が多いときに効果が大きいことが分かる。これは、式 (4)、(5) によって、送信パケット数が平均より多いフロー、つまりここでは TCP Reno コネクションのパケットがより高い確率で廃棄されるため、TCP Reno コネクションのウィンドウサイズが小さくなり、TCP Vegas との差が小さくなるためである。

次に、SRED 及び ZL-RED の Zombie List のサイズを 10、100、1000、10000 エントリとしたときの結果を図 5 に示す。図 5(a) 及び 5(b) は、TCP Reno コネクション及び TCP Vegas コネクションのコネクション数と、各バージョンの平均スループット及びパケットロス率との関係をそれぞれ示している。これらの図から、Zombie List サイズが 100 エントリ以上であれば、公平性には大きな影響を与えないことがわかる。これは、図 2 に示したように、ある程度 Zombie List のサイズが大きければ、推定フロー数は Zombie List のサイズに依存しないためである。しかし、フロー数に対して Zombie List のサイズが極端に小さい場合は、推定フロー数が小さくなるため (図 2) 式 (3) によってパケット廃棄率が小さくな

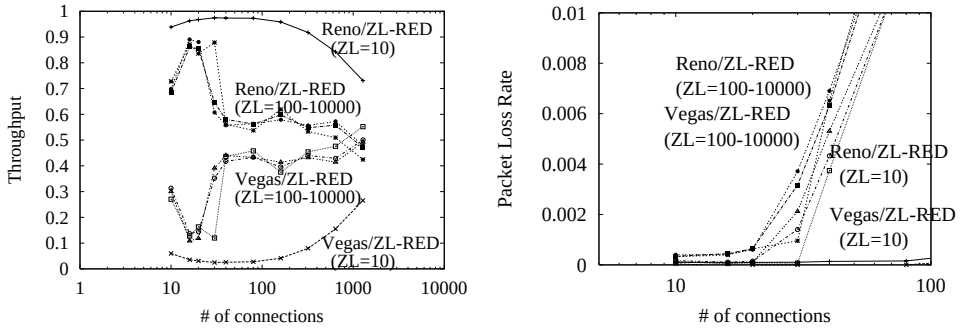


(a) スループット：B=1000 [packets] , ZL=1000 エントリ (b) スループット:B=5000 [packets],ZL=1000 エントリ (c) スループット:B=10000[packets],ZL=1000 エントリ



(d) パケット廃棄率：B=1000 [packets],ZL=1000 エントリ (e) パケット廃棄率：B=5000 [packets],ZL=1000 エントリ (f) パケット廃棄率：B=10000 [packets],ZL=1000 エントリ

図 4: SRED と ZL-RED を用いたときの TCP の各バージョンのスループット及びパケット廃棄率



(a) スループット：Buffer size=5000 (b) パケット廃棄率：Buffer size=5000

図 5: 異なる Zombie List サイズを用いたときの TCP の各バージョンのスループット及びパケット廃棄率

りバージョン間の不公平性が大きくなる (図 5)。

しかし、ZL-RED 方式を用いても、特にフロー数が少ない領域では、公平性の改善の度合いは小さい。これは、SRED はフロー数が少ない場合は、バッファ内パケット数を安定させるためにパケット廃棄率を小さく設定するため、全体のパケット廃棄率を SRED と等しくしながら mis-behaving フローのパケット廃棄率を大きくする ZL-RED では、TCP Reno コネクションのパケット廃棄率を大きく設定することができないためである (図 4(d)、4(e)、4(f))。

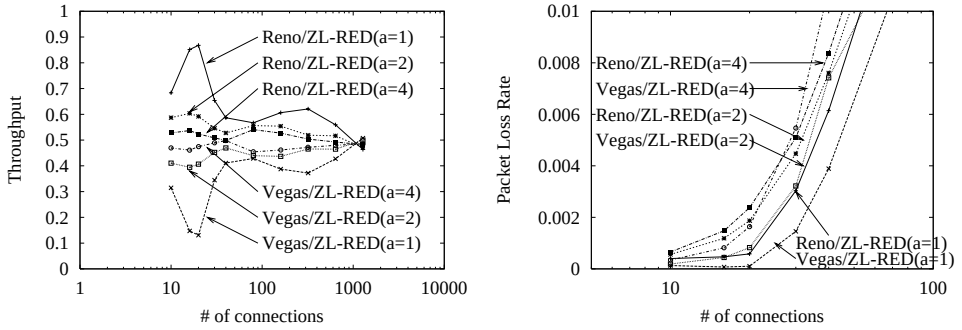
従って、コネクション数が少ない場合に公平性をさらに向上させるには、全体のパケット廃棄率を向上させて、mis-behaving フローの廃棄率を大きくする必要がある。そこで、ZL-RED のパケット廃棄率の計算式 (4) を次の式 (8) のように変更する。

$$F > T \times P(t) \times a \tag{8}$$

ここで a は定数である。図 6 は、a を 1 (ZL-RED に相当)、2、及び 4 にした時のシミュレーション結果を示している。この図から、a を大きくし、mis-behaving フローのパケット廃棄率を大きくすることにより、TCP のバージョン間の公平性が向上することがわかる (図 5(a))。しかし、a を大きくすることによって、全体のパケットロス率が大きくなり (図 5(b))、パケット廃棄率を向上させると、全体のスループットが低下する場合があると考えられる [9] ので、パラメータを適切に設定することが重要であると考えられる。

5 まとめと今後の課題

本稿では、ルータにおいてフロー間の公平なサービスを実現するための方式として、SRED で用いられている Zombie List を使い、mis-behaving フローを検出してそのパケット廃棄率を高く設定することで、フロー間の公平性を向上させる ZL-RED (Zombie Listed RED) を提案した。さらに、シミュレーションによって ZL-RED の性能評価を行い、TCP のバージョン間の公平性が大き



(a) スループット:B=5000 [packets], ZL=1000 エントリ

(b) パケット廃棄率：B=5000 [packets], ZL=1000 エントリ

図 6: 式 (8) を用いた場合の TCP の各バージョンのスループット及びパケット廃棄率

く向上することを示した。しかし、特にコネクション数が少ない場合には、公平性の改善の程度は小さく、さらに公平性を向上させるためには、全体のパケット廃棄率を SRED のそれよりも高く設定する必要があることが明らかになった。

今後は、全体のパケット廃棄率を極力高くすることなくコネクション間の公平性をさらに向上させる方式についての検討を行いたい。また、ZL-RED の UDP コネクションと TCP コネクション間の不公平性に対する効果を検証することも今後の課題としたい。

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」、科学技術庁の平成 10 年度科学技術振興調整費による「高度医療ネットワークに関する研究調査」、通信・放送機構「次世代広帯域ネットワーク利用技術の研究開発プロジェクト」、及び (財) 電気通信普及財団の研究助成「超高速ネットワークのためのトランスポート層プロトコルに関する研究」によっている。ここに記して謝意を表す。

参考文献

[1] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE/ACM Transactions on Networking*, vol. 6, August 1999.

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

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

[4] 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.

[5] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, “Evaluation with TCP Vegas: Emulation and experiment,” *ACM SIGCOMM Computer Communications Review*, vol. 25, pp. 185–195, August 1995.

[6] U. Hengartner, J. Bolliger, and T. Gross, “TCP Vegas revisited,” in *Proceedings of IEEE INFOCOM 2000*, March 2000.

[7] J. Mo, R. J. La, V. Anantharam, and J. Walrand, “Analysis and comparison of TCP reno and vegas,” in *Proceedings of IEEE INFOCOM’99*, March 1999.

[8] 倉田 謙二, 長谷川 剛, 村田 正幸, “TCP Reno と TCP Vegas の混在環境における公平性の評価,” **電子情報通信学会技術研究報告** (SSE99-98), pp. 67–72, November 1999.

[9] Kenji Kurata, Go Hasegawa, Masayuki Murata, “Fairness comparisons between TCP Reno and TCP Vegas for future deployment of TCP Vegas,” to be presented at *INET2000*, January 2000.

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

[11] 倉田 謙二, 長谷川 剛, 村田 正幸, “TCP バージョン間の公平性向上のための RED の改善方式に関する検討,” **電子情報通信学会 技術研究報告** (SSE2000-52), pp. 13–18, June 2000.

[12] T. J. Ott,T. V. Lakshman,and L. Wong, “SRED : Stabilized RED,” in *Proceedings of IEEE INFOCOM’99*, March 1999.

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

[14] D. Lin and R. Morris, “Dynamics of random early detection,” in *Proceedings of ACM SIGCOMM’97*, pp. 127–137, October 1997.