TCP Reno と TCP Vegas の混在環境における公平性の評価

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専攻
〒 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, murata}@ics.es.osaka-u.ac.jp

‡ 大阪大学経済学部
〒 560-0043 大阪府豊中市待兼山町 1-7
Phone: 06-6850-5233
E-mail: hasegawa@econ.osaka-u.ac.jp

あらまし TCP Vegas バージョンはそれ自体は従来の TCP に比べて高スループットを得ることができるとして期待されている。しかし、将来 TCP Vegas が普及する過程で重要になると考えられる、従来バージョン (TCP Reno / Tahoe) と TCP Vegas との混在環境におけるバージョン間の公平性に関してはこれまで評価がなされていない。本稿では、ボトルネックルータにおいて TCP Reno と TCP Vegas のコネクションが共存している環境における、バージョン間の公平性についての検討を解析及びシミュレーションによって行う。評価においてはルータのバッファ処理規律として FIFO (First In First Out) 及び RED (Random Early Detection) の両方式を対象にし、バッファ処理規律がバージョン間の公平性に与える影響についても検討を行う。最後に、それらの検討結果を基に TCP Vegas がインターネットに普及するためのシナリオを提案する。

キーワード TCP (Transmission Control Protocol)、輻輳制御方式、TCP Reno、TCP Vegas、公平性、RED (Random Early Detection)

Fairness Comparisons between TCP Reno and TCP Vegas for Future Deployment of TCP 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, murata}@ics.es.osaka-u.ac.jp

‡Faculty of Economics, Osaka University
1-7, Machikaneyama, Toyonaka, Osaka 560-0043, Japan
Phone: +81-6-6850-5233
E-mail: hasegawa@econ.osaka-u.ac.jp

Abstract TCP Vegas version is expected to achieve higher throughput than TCP Tahoe and Reno versions, which are currently used in the Internet. However, we need to consider a migration path of TCP Vegas when it is deployed in the Internet. In this paper, we focus on the situation where multiple TCP Reno and Vegas connections coexist at the bottleneck router, by which the fairness property is investigated to seek the possibility of future deployment of TCP Vegas. We consider FIFO (First In First Out) and RED (Random Early Detection) as buffering discipline at the router, and evaluate the effect of RED algorithm on fairness enhancement.

Keyword TCP (Transmission Control Protocol), Congestion Control Algorithm, TCP Reno, TCP Vegas, Fairness, RED (Random Early Detection)

1 はじめに

現在のインターネットにおけるトランスポート層プロトコルのデファクトスタンダードである TCP (Transmission Control Protocol) [1] は、WWW (World Wide Web) やファイル転送に代表される多くのアプリケーションによって使われている。従って、将来ネットワークインフラが大幅に増強され、その構造が変化したとしても、TCP は引き続いて使われることが考えられる。しかし、現在使われている TCP (主に Tahoe, Reno バージョン) の輻輳制御方式は必ずしも性能のよいものではないことが過去の研究で盛んに指摘されており、TCP のスループット、公平性等を改善するための様々な方式が提案されている。

その中でも、1995 年に提案された TCP Vegas バージョン [2] が、高スループットを得ることができ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 バージョンとは違い、パケットの RTT (Round Trip Time) の変動を観測して輻輳を検出し、ウィンドウサイズを動的に増減させる。その結果従来バージョンに比べてパケットロス数が減少し、スループットが向上すると考えられる。例えば [2] では、従来バージョンの TCP に比べて 40%スループットが改善するという報告もなされている。

しかし一方で、TCP Vegas が従来バージョンの TCP と共存している環境においては TCP Vegas がスループット面で非常に不利になるという指摘もなされている [3]。しかし、[3] では TCP Vegas と TCP Reno コネクションが 1 本のみ存在する場合のについての検討がなされており、またその不公平性を改善する有効な方式も提案されていない。将来のネットワークにおける TCP Vegas のスムーズな普及を図るためには、より一般的な環境における TCP のバージョン間の公平性を評価する必要がある。

そこで本稿では、ボトルネックルータにおいて TCP Reno と TCP Vegas のコネクションが複数本共存している環境における、TCP のバージョン間の公平性についての検討を数学的解析手法を用いて行い、TCP Vegas が不利になる原因、及びその不公平性の改善方法を検討する。さらにルータのバッファのパケット処理規律として FIFO (First In First Out) 方式及び RED (Random Early Detection) 方式 [4] を対象に挙げ、バッファ処理規律がバージョン間の公平性に与える影響についても検討する。

以下、2 章では TCP Reno 及び TCP Vegas の輻輳制御方式を簡単に説明し、3 章では解析及びシミュレーションで用いるネットワークモデルを説明する。次に 4 章においてボトルネックルータのバッファ処理規律として FIFO 方式及び RED 方式を用いた場合のスループット解析を行う。5 章では解析結果を用いて数値例を挙げ、TCP のバージョン間の公平性についての評価を行い、TCP Vegas が普及するための条件を考察する。最後にまとめと今後の課題を 6 章で述べる。

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 [5] (TCP Reno)、[2] (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。

2.1 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 [5]。

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

ここで、$ssth$ [packets] は TCP が Slow Start Phase から Congestion Avoidance Phase に移行する際の閾値で、セグメントロス発生時に、以下のように更新される [5]。

$$ssth = cwnd(t)/2 \quad (2)$$

式 (1) より、ウィンドウサイズはセグメントロスが発生するまで増加し続け、周期的な動作をすることがわかる。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 (タイムアウト) 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) [5] される。タイムアウトによって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。また、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 TCP Vegas

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。

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

$$\text{だたし、} diff = \frac{cwnd}{base\_rtt} - \frac{cwnd}{rtt} \quad (3)$$

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スピードが TCP Reno の 1/2 になっており、以下のよ

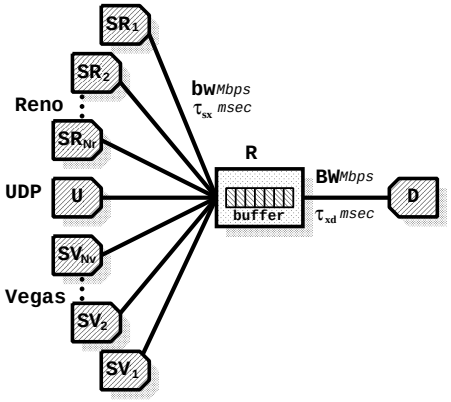


図 1: ネットワークモデル

うに $cwnd$ が更新される。

$$cwnd(t + t_A) = cwnd(t) + 1/2$$

3 ネットワークモデル

本稿で用いるネットワークモデルを図 1 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 (D)、ボトルネックルータ (R)、及びそれらを接続するリンクからなる。TCP Reno 及び TCP Vegas を用いる送信側端末数をそれぞれ Nr、Nv、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 [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 [Mbps] = μ [packets/sec]、ボトルネックルータ (R) 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 τ_{sx} [sec]、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 τ_{xd} [sec] 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 FIFO (First In First Out) 方式あるいは RED (Random Early Detection) [6] を用いる。

次章以降の解析及びシミュレーションにおいては、送信側端末が TCP Reno/Vegas を用いてデータを送信した場合の平均スループットの比較を行い、TCP のバージョン間の公平性に関する評価を行う。

4 スループット解析

本章では、3 章で説明したネットワークモデルを用いて、複数本の TCP Reno と TCP Vegas のコネクションがボトルネックルータを共有した時の、各バージョンの TCP の平均スループットを数学的解析手法を用いて導出する。

次節の解析においては、同一バージョンの TCP コネクションはすべて公平なスループットを得ているとする。また、各コネクションのスループットは、ボトルネックルータにおけるバッファ占有率に比例すると仮定する。

4.1 FIFO 方式

図 2 は FIFO 方式におけるボトルネックルータのバッファ内パケット数の変化を示したグラフである。TCP Reno のコネクションはパケットロスが発生するまでウィンドウサイズを増加するので、混在環境においてもパケットロスをきっかけに周期的な動きをする。バッファが一杯になった後、パケットロスは 1 コネクションで 1 [packet] ずつ発生し、各コネクションはパケットロスを全て Fast Retransmit によって検出し再送を行うことができると仮定すると、バッファが一杯になった後に送信側端末がパケットロスを検知し、ウィンドウサイズを小さくするまでには 1 RTT(ラウンドトリップ時間) がかかる (図 2)。

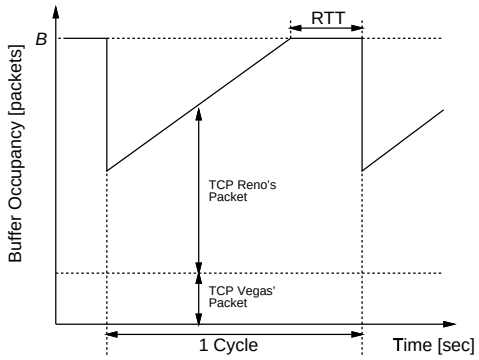


図 2: バッファ内パケット数の変化

TCP Vegas のコネクションは送信パケットのラウンドトリップ時間を観測し、送受信端末間のコネクション上に蓄積されるバッファ内パケット数を $\alpha \sim \beta$ の間に保とうとする [7]。従って、ラウンドトリップ時間が大きくなるとウィンドウサイズを小さくする。一方 TCP Reno のコネクションはラウンドトリップ時間の増減に関係なくセグメントロスが発生するまでウィンドウサイズを増加するので、TCP Vegas のコネクションのラウンドトリップ時間も大きくなる。従って TCP Vegas のコネクションはウィンドウサイズを小さくし続け、常に最小値である 1 [packets] になる。よって、TCP Vegas のコネクションのウィンドウサイズの合計 W_v [packets] は常に、

$$W_v = N_v \tag{4}$$

となる。したがって、ルータのバッファが一杯になりパケットロスが発生する時の、TCP Reno のコネクションのウィンドウサイズの合計 W_r [packets] は、

$$W_r = 2\tau\mu + B - W_v \tag{5}$$

となる。バッファが一杯になった後、バッファ溢れによって L 個のパケットロスが発生するとすると、式 (1) より、TCP Reno のコネクションは 1 RTT で 1 [packet] ずつウィンドウサイズを増加させるので、

$$L = 1 \times N_r = N_r \tag{6}$$

となる。各 TCP コネクションでパケットロスが発生する確率は各コネクションのウィンドウサイズの比に比例すると仮定すると、1 回のバッファ溢れによって TCP Reno 及び TCP Vegas のコネクションで発生するパケットロス数 L_r [packets]、L_v [packets] は、

$$L_r = L \cdot \frac{W_r}{W_r + W_v} \tag{7}$$

$$L_v = L \cdot \frac{W_v}{W_r + W_v} \tag{8}$$

となる。Fast Retransmit によってパケットロスを検出した TCP Reno のコネクションはウィンドウサイズを 1/2 にするので (2.1 節参照)、パケットロス直後の TCP Reno のコネクションのウィンドウサイズの合計 W_r' [packets] は、式 (7) より、

$$\begin{aligned} W_r' &= \frac{1}{2} \cdot \frac{W_r}{N_r} L_r + \frac{W_r}{N_r}(N_r - L_r) \\ &= \frac{W_r + 2W_v}{2(W_r + W_v)} \cdot W_r \end{aligned} \tag{9}$$

となる。従って、図 2 より、TCP Reno のコネクション

のウィンドウサイズの合計の平均値 $\overline{W_r}$ [packets] は、以下のように求めることができる。

$$\overline{W_r} = \frac{\frac{1}{2}(W_r + W_r')\frac{W_r - W_r'}{N_r} + W_r}{\frac{W_r - W_r'}{N_r} + 1} \quad (10)$$

また、TCP Vegas のコネクションのウィンドウサイズの合計の平均値 $\overline{W_v}$ [packets] は、$\overline{W_v} = W_v$ となる。従って、各バージョンのコネクション 1 本の平均バッファ内パケット数 $\overline{B_r}$ [packets]、$\overline{B_v}$ [packets] は、

$$\overline{B_r} = \overline{W_r} \cdot \frac{B}{2\tau\mu + B},\ \overline{B_v} = \overline{W_v} \cdot \frac{B}{2\tau\mu + B} \quad (11)$$

となる。各コネクションのスループットは、ボトルネックルータにおけるバッファ占有率に比例すると仮定しているので、各コネクションの平均スループット ρ_r、ρ_v [packets/sec] は以下のようにして得ることができる。

$$\rho_r = \mu \cdot \frac{B_r}{B_r + B_v},\ \rho_v = \mu \cdot \frac{B_v}{B_r + B_v} \quad (12)$$

4.2 RED 方式

RED 方式においては、バッファ溢れによってパケットロスが発生する前に、バッファ内パケット数がある閾値を越えると確率的にパケットを廃棄する [4]。本節においては、パケットロスは全て RED 方式による確率的廃棄によってのみ発生するとし、パケット廃棄率はバッファ内パケット数に関係なく一定値 p であるとする。

RED 方式を用いた場合にも、TCP Reno のコネクションはパケットロスが発生するまでウィンドウサイズを増加し続ける。従って、FIFO 方式の場合と同様に、TCP Vegas のコネクションはウィンドウサイズを増加させることはできない。よって、TCP Vegas のコネクションのウィンドウサイズの合計 W_v 及びその平均 $\overline{W_v}$ について以下の式が成立する。

$$\overline{W_v} = W_v = N_v \quad (13)$$

一方、TCP Reno のコネクションのウィンドウサイズは、FIFO 方式の場合と同様に、確率的なパケット廃棄をきっかけとする周期的な変化をする。これを 1 サイクルと定義すると、1 サイクルで送信できる平均パケット数は $1/p$ である。また、TCP Reno においては、ウィンドウ内で 2 個以上のパケットロスが発生すると、全ての廃棄パケットを Fast Retransmit で検出、再送することができずに、タイムアウトを待ち、再送を行う [8]。従って、TCP Reno の 1 本のコネクションにおいてパケットロスが発生する時の平均ウィンドウサイズを $\overline{w_r}$ [packets] とすると、RED 方式による確率的な廃棄の下でタイムアウトが発生する確率 p_{to} は以下のように導出できる。

$$p_{to} = \sum_{i=2}^{\overline{w_r}} p^i (1-p)^{\overline{w_r} - i} \quad (14)$$

以下、パケットロスをタイムアウトによって検出するか、Fast Retransmit によって検出するかによって場合分けを行って解析を行う。

タイムアウトが発生する場合、ウィンドウサイズは 1 [packet] にリセットされ、$\overline{w_r}/2$ になるまで Slow Start Phase によるウィンドウサイズの増加を行う。式 (1) から、Slow Start Phase の長さ $T_{to,1}$ [sec]、及び Slow Start Phase における送信パケット数 $A_{to,1}$ [packets] は以下のようになる。

$$T_{to,1} = rtt \cdot \log_2(\overline{w_r}/2) \quad (15)$$

$$A_{to,1} = (\overline{w_r}2) - 1 \quad (16)$$

ここで rtt [sec] はパケットの平均ラウンドトリップ時間である。さらに、その後の Congestion Avoidance Phase の長さ $T_{to,2}$ [sec] 及び Congestion Avoidance Phase における送信パケット数 $A_{to,2}$ は、式 (1) より、

$$T_{to,2} = rtt(\overline{w_r} - (\overline{w_r}/2)) \quad (17)$$

$$A_{to,2} = \frac{1}{2}(\overline{w_r} + (\overline{w_r}/2))(\overline{w_r} - (\overline{w_r}/2)) \quad (18)$$

となる。

一方、タイムアウトが発生せず、Fast Retransmit でパケットを再送する場合は、パケット再送後のウィンドウサイズはパケットロスの直前の 1/2 になり、Slow Start Phase は行わずに Congestion Avoidance Phase に移り、1 RTT で 1 [packet] 分ずつウィンドウサイズを増加させる (2.1 節参照)。従って、Slow Start Phase、Congestion Avoidance Phase の長さと送信パケット数は以下のようになる。

$$T_{fr,1} = 0,\ A_{fr,1} = 0 \quad (19)$$

$$T_{fr,2} = rtt(\overline{w_r} - (\overline{w_r}/2)) \quad (20)$$

$$A_{fr,2} = \frac{1}{2}(\overline{w_r} + (\overline{w_r}/2))(\overline{w_r} - (\overline{w_r}/2)) \quad (21)$$

従って、タイムアウト時間を rto [sec] とすると、式 (15)–(21) より、1 サイクルで送信できるパケット数及び $\overline{w_r}$ について以下の式が成立する。

$$\frac{1}{p} = p_{to}(A_{to,1} + A_{to,2}) + (1 - p_{to})(A_{fr,1} + A_{fr,2}) \quad (22)$$

$$\overline{w_r} = p_{to}\left(\frac{A_{to,1} + A_{to,2}}{T_{to,1} + T_{to,2} + rto}\right) + (1 - p_{to})\left(\frac{A_{fr,1} + A_{fr,2}}{T_{fr,1} + T_{fr,2}}\right) \quad (23)$$

式 (22)、(23) より p_{to} 及び $\overline{w_r}$ が求まるので、TCP Reno のコネクションのウィンドウサイズの合計の平均値 $\overline{W_r}$ は以下のように求めることができる。

$$\overline{W_r} = N_r \cdot \overline{w_r} \quad (24)$$

従って、各バージョンのコネクションの平均スループット ρ_r、ρ_v は、式 (11)– (12)、(13)、(24) から求めることができる。

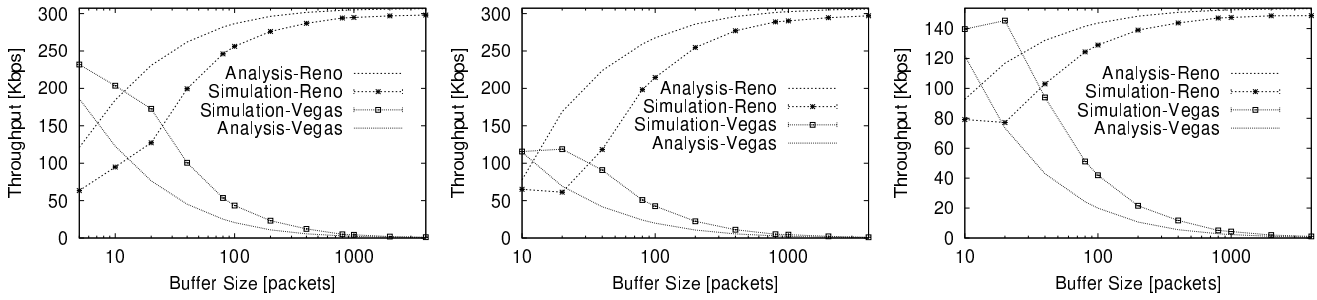
5 数値例及び考察

本章においては、4 章におけるスループット解析、及びコンピュータ上のシミュレーションによる数値例を挙げ、解析結果の妥当性を評価する。また、TCP Reno と TCP Vegas のバージョン間の公平性についての考察を行い、TCP Vegas が普及するためのシナリオを提案する。

以下の数値例においては、図 1 のネットワークモデルを用い、各パラメータは τ_{sx} = 0.0015 [sec]、τ_{xd} = 0.005 [sec]、bw = 10 [Mbps]、BW = 1.5 [Mbps] とする。また、RED 方式の閾値は th_{min} = 5 [packets]、th_{max} = $0.6B$ [packets] とする。

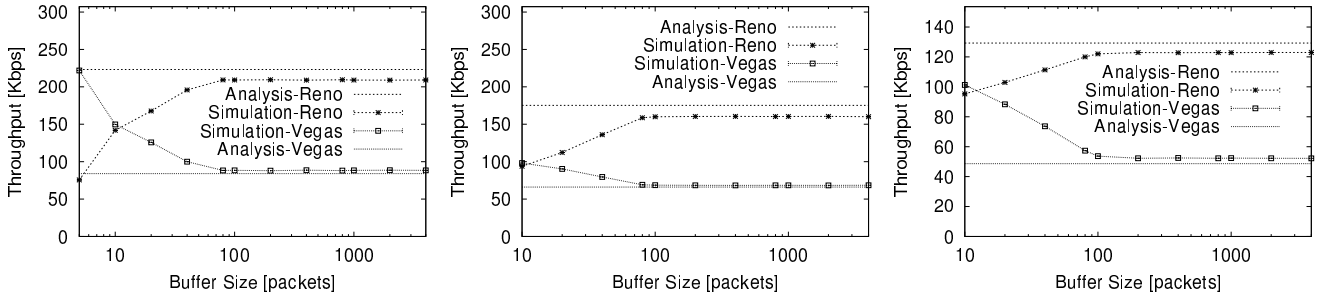
5.1 FIFO 方式

図 3 は、FIFO 方式を用い、TCP Reno 及び TCP Vegas のコネクション数 (N_r、N_v) を、それぞれ N_r = 5、N_v = 5 (図 3(a))、N_r = 5、N_v = 10 (図 3(b))、N_r = 10、N_v = 5 (図 3(c)) にしたときの、ルータのバッファサイズと各バージョンの TCP の平均スループットの関係を示したグラフである。図中には、解析結果に加えてシミュレーションによる結果をあわせて示している。これらの図から、4 章において行った解析の結果が、コネクション数



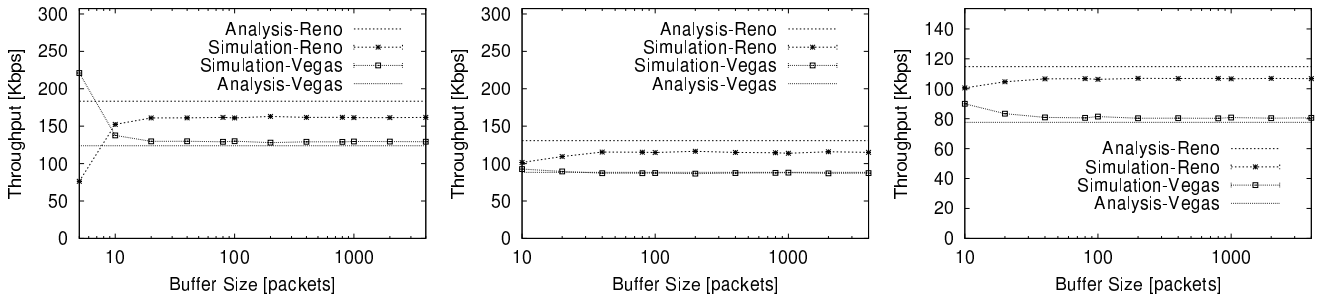
(a) N_r = 5, N_v = 5. (b) N_r = 5, N_v = 10. (c) N_r = 10, N_v = 5.

図 3: FIFO 方式



(a) N_r = 5,N_v = 5. (b) N_r = 5, N_v = 10. (c) N_r = 10, N_v = 5.

図 4: RED 方式: $p = \frac{1}{30}$



(a) N_r = 5, N_v = 5. (b) N_r = 5, N_v = 10. (c) N_r = 10, N_v = 5.

図 5: RED 方式: $p = \frac{1}{10}$

に関係なく概ね妥当であることがわかる。しかし、特にバッファサイズが小さい場合に、解析結果が TCP Reno のコネクションのスループットを高く、TCP Vegas のコネクションのスループットを低く見積っている。これは、解析においては TCP Vegas のウィンドウサイズが常に 1 [packet] となると仮定しているが、シミュレーションでは不規則に TCP Vegas のコネクションのウィンドウサイズが 2 [packets] 以上になることもあるため、特にバッファサイズが小さい場合に TCP Vegas のコネクションのスループットが高くなるためである。

また、バッファサイズが大きくなると TCP Reno のコネクションが非常に有利になり、TCP Vegas のコネクションはほとんど転送を行えていないことがわかる。これは、TCP Reno のコネクションは、パケット廃棄が発生するまでウィンドウサイズを増加し続けることによって、バッファを完全に使い切ることができるのに対し、TCP Vegas のコネクションはウィンドウサイズをほとんど 1 [packet] から増加することができないためである。従って、TCP Reno と TCP Vegas のコネクションが混在している環境では、FIFO 方式の下では公平性を維持できないことがわかる。

5.2 RED 方式

図 4 は、RED 方式を用い、パケット廃棄率 p を 1/30 にした場合の結果を示している。図から、特にバッファサイズが小さい場合にシミュレーション結果と解析結果が一致していないことがわかる。これは、バッファサイズが小さい場合には、RED 方式による確率的なパケット廃棄に加えて、バッファ溢れによるパケット廃棄が発生しているが、4 章における解析においては、バッファ溢れによるパケットロスを考慮していないためである。しかし、バッファサイズが十分大きい場合には、解析結果は高い精度の近似解を示しているといえる。

また、FIFO 方式 (図 3) と比較すると TCP Reno/Vegas 間の公平性が向上していることがわかる。これは、RED 方式によってバッファ溢れが発生する前に積極的にパケット廃棄を行うことによって、TCP Reno のコネクションの平均ウィンドウサイズが小さくなり、TCP Vegas のコネクションのウィンドウサイズに接近するためである。この傾向は、RED 方式のパケット廃棄率を高くした場合

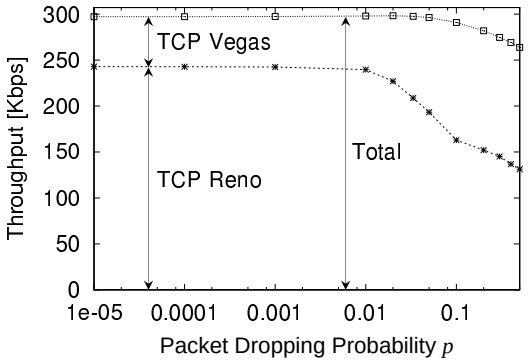


図 6: パケット廃棄率 p とスループットの関係 (B = 100 [packets])

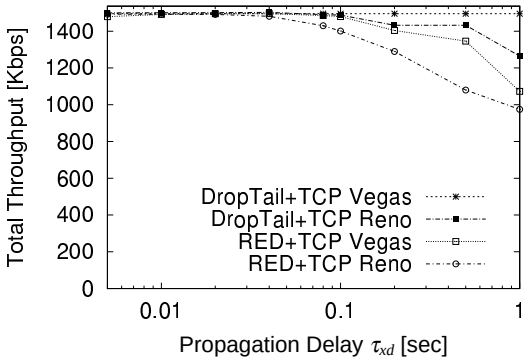


図 7: ボトルネックリンクの伝搬遅延時間 τ_{xd} とスループットの関係 (B = 100 [packets])

により顕著になる。図 5 は、パケット廃棄率 p を 1/10 にした場合の結果を示している。p = 1/30 の場合 (図 4) と比較すると公平性がさらに向上していることがわかる。これは、パケット廃棄率が大きくなることにより、TCP Reno のコネクションの平均ウィンドウサイズがさらに小さくなるためである。

しかし、パケット廃棄率を大きくすると、全体のスループットが低下する場合がある。図 6 は、RED 方式のパケット廃棄率 p とスループットの関係を示したグラフである。ここでバッファサイズ B は 100 [packets] である。図から、パケット廃棄率が大きくなると、TCP Reno/Vegas 間の公平性は大きく改善するものの、全体のスループットが低下していることがわかる。従って、全体のスループットを維持し、かつ TCP のバージョン間の公平性を向上させるには、適切なパケット廃棄率 p を用いる必要がある。

5.3 TCP Vegas の普及の条件

図 7 は、TCP Reno のコネクションのみ、あるいは TCP Vegas のコネクションのみが 5 本存在する場合の、ボトルネックリンクの伝搬遅延時間 τ_{xd} とスループットの関係を示したものである。バッファサイズ B は 100 [packets] としている。この図から、1 バージョンの TCP のみが存在する場合には、TCP Vegas を FIFO 方式で用いた場合が最もスループットが高いことがわかる。これは、TCP Vegas はネットワークの輻輳を初期の段階で回避し、かつパケットロスを防止することを目的としているので、TCP Reno を用いた場合よりもスループットが高くなるためである。また、RED 方式によるパケット廃棄は、パケットロスを回避するという TCP Vegas の理想的な動作を妨げる原因になるため、RED 方式の下で TCP Vegas を用いた場合はスループットが低下する。

以上の結果から、TCP Vegas が普及するためのシナリオの一つとして、以下のような方法が考えられる。まず、TCP Vegas のコネクション数が TCP Reno に比べて少ない、あるいはほぼ等しいような初期段階においては、FIFO 方式の下では TCP Vegas のスループットが不当に低下するので、ルータ (スイッチ) のバッファにおけるパケット処理規律として RED 方式を用いる。その後、TCP Vegas が十分普及すれば、パケット処理規律を FIFO 方式に戻し、高スループットを達成できるようにすれば、TCP Vegas のスムーズな普及が可能であると考えられる。

6 まとめと今後の課題

本稿では、ボトルネックルータにおいて TCP Reno と TCP Vegas のコネクションが複数本共存している環境における、TCP のバージョン間の公平性に関する検討を、解析及びシミュレーションによって行った。その結果、ルータのバッファにおけるパケット処理規律が FIFO 方式の場合には、TCP Vegas がスループット面で非常に不利になることがわかった。また、RED 方式を用いることによって、公平性を改善することができるが、全体のスループットと公平性の間にトレードオフが存在し、適切なパケット廃棄率を用いる必要があることがわかった。

しかし、本稿における解析手法は、TCP Vegas のコネクションが一時的にウィンドウサイズを 2 [packets] 以上に大きくする現象や、RED 方式におけるバッファ溢れによるパケットロス等、無視している部分があるため、これらの点を考慮することにより解析の正確性をさらに高めたい。また、RED 方式における適切なパケット廃棄率 p の導出方式も検討していく予定である。

参考文献

[1] G. R. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Massachusetts: Addison-Wesley, 1995.

[2] 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.

[3] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP reno and vegas," *Proceedings of INFOCOM'99*, March 1999.

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

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

[6] T. V. Lakshman and U. Madhow, "Performance analysis of window–based flow control using TCP/IP: Effect of high bandwidth–delay product and random loss," *Proceedings of HPN'94 (High Performance Networking*, pp. 135–149, June 1994.

[7] G. Hasegawa, M. Murata, and H. Miyahara, "Fairness and stability of the congestion control mechanism of TCP," *Proceedings of IEEE INFOCOM'99*, March 1999.

[8] K. Fall and S. Floyd, "Simulation–based comparisons of Tahoe, Reno, and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 5–21, July 1996.