伝搬遅延時間の異なる複数の TCP コネクションを考慮したウィンドウ型フロー制御方式の安定性解析

高垣 景一　　大崎 博之　　村田 正幸

大阪大学大学院 基礎工学研究科
〒 560-8531 大阪府豊中市待兼山町 1-3

(Phone) +81-6-6850-6616
(Fax) +81-6-6850-6589
(E-mail) takagaki@ics.es.osaka-u.ac.jp

あらまし　パケット交換網において、ベストエフォート型のトラヒックを効率的に網に収容するためには、フィードバック型の輻輳制御が不可欠である。現在広く普及しているインターネットでは、ウィンドウ型のフロー制御方式が TCP (Transmission Control Protocol) で用いられている。近年、一般に広く用いられている TCP Reno よりも、高い性能を示す TCP Vegas が注目されている。本稿では、TCP Vegas に基づくウィンドウ型のフロー制御方式を対象とし、その安定性を制御理論の手法を用いて明らかにする。特に、距離の異なる複数の TCP コネクションが存在する場合を対象とし、距離の違いに応じてどのように制御パラメータを決定すればシステムを安定化できるかを明らかにする。その結果、ウィンドウサイズの変化量を決定する制御パラメータを、伝搬遅延時間の比率よりも小さな値に設定すれば、ネットワークを安定化できることを明らかにする。
和文キーワード　TCP/IP ネットワーク、ウィンドウ型フロー制御、TCP Vegas、制御理論、安定性解析

Stability Analysis of a Window-based Flow Control Mechanism for TCP Connections with Different Propagation Delays

Keiichi Takagaki　　Hiroyuki Ohsaki　　Masayuki Murata

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) takagaki@ics.es.osaka-u.ac.jp

Abstract　A feedback-based congestion control mechanism is essential to realize an efficient best-effort service in high-speed networks. A window-based flow control mechanism called TCP (Transmission Control Protocol), a sort of feedback-based congestion control mechanism, has been widely used in the current Internet. Recently, TCP Vegas has been proposed, which is another version of TCP mechanism, and achieves much better performance than currently used TCP Reno. In this paper, we focus on a window-based flow control mechanism based on the congestion avoidance mechanism of TCP Vegas, and analyze its stability using a control theoretic approach. The main objective of this paper is to analyze the dynamics of the window-based flow control mechanism when TCP connections have different propagation delays. We analytically show that the network always becomes stable if the control parameter of each connection, which determines the amount of window size increase/decrease per round-trip time, is set to be less than the ratio of its propagation delay to the shortest one among connections.
英文 key words　TCP/IP Network, Window-Based Flow Control Mechanism, TCP Vegas, Control Theory, Stability Analysis

1 研究の背景

パケット交換ネットワークにおいて、データ系のサービスを効率的に収容するためには、フィードバック型の輻輳制御が不可欠である。現在広く普及しているインターネットでは、フィードバック型の輻輳制御として、ウィンドウ型のフロー制御方式である TCP (Transmission Control Protocol) が用いられている。TCP には、ネットワーク内でパケットが失われた場合に、失われたパケットを再び受信側ホストに送出するパケット再送機能と、ネットワークの輻輳状況に応じてウィンドウサイズを変更し、送出するパケット数を調整する輻輳制御機能が備えられている。

最近、現在広く使用されている TCP(TCP Reno) よりも良い性能を示す TCP の実装例として、TCP Vegas が提案されている [1]。TCP Vegas では、TCP Reno に比べて以下の点が改良されている。すなわち、(1) 新しいタイムアウト機構、(2) ネットワーク内でのバッファの占有量を制御する輻輳回避機構、(3) スロースタート機構である。特に、(2) に挙げた輻輳回避機構では、パケットを送出してからそのパケットに対応した ACK (Acknowledgment) パケットを受信するまでの時間、すなわち RTT(ラウンドトリップ時間) を測定し、これを輻輳状態に関するネットワークからのフィードバック情報としてウィンドウサイズの変更を行う。このため、TCP Reno のように、ネットワーク内で発生する輻輳を検出するために、パケット損失を待つ必要がない。その結果、定常状態においてウィンドウサイズがある値に収束し、その結果 TCP Reno に比べてスループットが向上する。

我々はこれまで、TCP Vegas の最大の特徴である、(2) の輻輳回避機構を用いたウィンドウフロー型の輻輳制御方式の特性を制御理論を用いて解析した [2]。ただし、TCP Vegas の輻輳回避機構では、RTT ごとに 1 パケットぶんだけウィンドウサイズを線形的に増減するが、ここではウィンドウサイズの変化量を制御パラメータとして扱える方式を対象とした。このようなシステムに対して制御理論を適用することにより、その安定性と過渡特性を定量的に明らかにした。しかし、文献 [2] で扱った解析モデルは、すべてのコネクションの伝搬遅延時間が等しいことを仮定していた。実際のネットワークにおいては、各コネクションの伝搬遅延時間はそれぞれ異なるため、その解析結果をそのまま現実のネットワークに適用することは困難であった。

TCP Vegas の特性を数学的な手法を用いて解析したものとしては、文献 [3, 4, 5] などが存在する。これらの研究では、送信側ホストのウィンドウサイズの変化を流体モデルなどで近似し、TCP Vegas のスループットを導出している。しかし、これらの論文で用いられている解析モデルでは、単一のコネクション [3, 4]、もしくは 2 本のコネクション [5] のみを対象としている。このため、伝搬遅延時間の異なるコネクションが複数存在するような、より現実的なネットワークには適用できない。また、TCP Vegas の安定性についてはまったく明らかにされていない。

そこで本稿では、[2] で用いた解析手法を拡張することにより、各コネクションの伝搬遅延時間が異なる、より一般的なネットワークモデルを対象とした解析を行う。まず、伝搬遅延時間の異なる複数のコネクションが存在する場合に、定常状態におけるウィンドウサイズやルータのバッファ内パケット数を導出する。これにより、各コネクションの伝搬遅延時間が異なる場合でも、各コネクションのバッファ占有量を制御するパラメータを等しく設定すれば、各コネクションのスループットが公平になることを示す。さらに、制御理論を適用することにより、ウィンドウ型フロー制御方式の安定性解析を行なう。これにより、ルータの処理能力や伝搬遅延時間などのシステムパラメータが変化した場合に、システムを安定化できる制御パラメータの領域がどのように変化するかを定量的に明らかにする。

距離の異なる複数のコネクションを考慮して、フィードバック型の輻輳制御方式に制御理論を適用した研究としては [6, 7, 8] などが存在する。これらの論文では、ATM ネットワークにおけるレート型の輻輳制御方式を対象とし、制御理論を適用することによってシステムのコントローラを設計している。しかし、これらの結果を、本稿で扱うようなウィンドウ型のフロー制御方式には適用できない。

以下、2 章において、本稿で対象とするウィンドウ型フロー制御方式およびその解析モデルを説明する。また、3 章において、制御理論を適用することにより対象とするウィンドウ型フロー制御方式の安定性解析を行う。4 章では、制御パラメータと安定性との関連について検討を行う。最後に、5 章において本稿のまとめと今後の課題について述べる。

2 解析モデル

本稿で用いる解析モデルを図 1 に示す。複数の送信側ホストが単一のルータを経由して、それぞれ対応する受信側ホストに接続されている様子を示している。伝搬遅延時間が異なる M 種類のコネクションが存在しており、往復伝搬遅延時間が τ_m $(1 \leq m \leq M)$ であるコネクションの本数を N_m (≥ 1) とする。一般性を失なうことなく、$\tau_1 < \tau_2 < \cdots < \tau_{M-1} < \tau_M$ と仮定する。また、往復伝搬遅延時間 τ_m の比をあらわす既約な正整数 Δ_m $(1 \leq m \leq M)$ が存在すると仮定する。すなわち、

$$\frac{\tau_1}{\Delta_1} = \frac{\tau_2}{\Delta_2} = \cdots \frac{\tau_{M-1}}{\Delta_{M-1}} = \frac{\tau_M}{\Delta_M}$$

とする。ここでバッファにおけるパケットの待ち時間が無視できるとすれば、RTT(ラウンドトリップ時間) の比もまた Δ_m で与えられる。TCP Vegas では、RTT を単位としてウィンドウサイズを変化させる [1]。このため、τ_m/Δ_m を単位として時間をスロットに分割すれば、システムの状態を離散時間モデルで考えることができる。つまり、往復伝搬遅延時間が τ_m であるコネクションは、Δ_m スロットごとにウィンドウサイズを変更することになる。

スロット k における、往復伝搬遅延時間が τ_m である n $(1 \leq n \leq N_m)$ 番目の送信側ホストのウィンドウサイズを $w_{m,n}(k)$ とする。これは、n 番目の送信側ホストは、スロット k において、RTT 中に $w_{m,n}(k)$ 個のパケットを送出できることを意味する。本解析では、送信側ホストは常に転送するデータを持っており、1RTT あたり必ず $w_{m,n}(k)$ 個のパケットを送出すると仮定する。

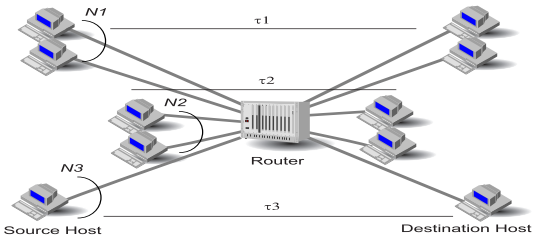

図 1: 解析モデル ($M = 3$ の場合)

さらに、スロット k においてルータのバッファ内に存在するパケット数を $q(k)$、ルータのバッファ容量を L とする。なお、各送信側ホストからルータに到着するパケットは、到着順に FIFO (First-In First-Out) でバッファに蓄えられ、ルータによって順番に処理されるものとする。また、ルータの処理能力を B とする。なお、以下では送信側ホストのウィンドウサイズ $w_{m,n}(k)$、ルータのバッファ内パケット数 $q(k)$、ルータのバッファ容量 L、ルータの処理能力 B は、パケット長で正規化した値 (単位として “パケット” もしくは “パケット/ ms”) を用いるものとする。

ウィンドウ型フロー制御では、RTT 中にウィンドウサイズ分のパケットをネットワーク内に送出することが可能である。このため、往復伝搬遅延時間が τ_m であるコネクションは、1 スロットあたり平均して $w_{m,n}/\Delta_m$ 個のパケットを送出すると考えることができる。従って、スロット $k+1$ においてルータのバッファ内に存在するパケット数 $q(k+1)$ は以下の式で与えられる。

$$q(k+1) = \min\left[\max\left(\sum_{m=1}^{M}\sum_{n=1}^{N_m}\frac{w_{m,n}(k)}{\Delta_m} - B\,\Delta, 0\right), L\right]$$

ここで Δ は 1 スロットの長さ ($\equiv \tau_m/\Delta_m$) である。

TCP Vegas では、RTT を測定し、この値に応じてウィンドウサイズを変更する [1]。往復伝搬遅延時間が τ_m の n 番目の送信側ホストでは、スロット k において測定したパケットの RTT$r_m(k)$ に基づいて、以下のように $d_{m,n}(k)$ を計算する。

$$d_{m,n}(k) = \left(\frac{w_{m,n}(k)}{\tau_m} - \frac{w_{m,n}(k)}{r_m(k)}\right) \times \tau_m \qquad (1)$$

RTT$r_m(k)$ は、ルータのバッファ内に存在するパケット数によって変化し、以下の式で与えられる。

$$r_m(k) = \tau_m + \frac{q(k)}{B}$$

TCP Vegas では、$d_{m,n}(k)$ の値に基いて、RTT ごとにウィンドウサイズを変更する [1]。具体的には、送信側ホストのウィンドウサイズ $w_{m,n}(k+\Delta_m)$ は以下のように変更される。

$$w_{m,n}(k+\Delta_m) = \begin{cases} w_{m,n}(k)+1, & \text{if } d_{m,n}(k) < \alpha_{m,n} \\ w_{m,n}(k)-1, & \text{if } d_{m,n}(k) > \beta_{m,n} \\ w_{m,n}(k), & \text{otherwise} \end{cases} \qquad (2)$$

ここで $\alpha_{m,n}$ および $\beta_{m,n}$ は、RTT ごとにどれだけネットワーク内に余分なパケットを送出してもよいかを決定する制御パラメータである。本解析では、式 (2) における $w_{m,n}(k+\Delta_m)$ の計算方法を、以下のように変更したモデルを扱う [2]。

$$w_{m,n}(k+\Delta_m) = \max(w_{m,n}(k) + \delta_{m,n}(\gamma_{m,n} - d_{m,n}(k)), 0) \qquad (3)$$

ここで $\delta_{m,n}$ は、RTT ごとにどれだけウィンドウサイズを変化させるかを決定する制御パラメータである。

3 安定性解析

以下では、伝搬遅延時間が等しい送信側ホストの、ウィンドウサイズの初期値や制御パラメータはすべて等しいと仮定する。その場合、送信側ホストの制御パラメータを $\delta_m \equiv \delta_{m,n}$ ($1 \le m \le M$、$1 \le n \le N_m$) などと表記する。また、全ての送信側ホストは式 (3) に従って、RTT ごとにウィンドウサイズを変更するとする。この時、スロット $k+1$ におけるバッファ内パケット数 $q(k+1)$ は以下の式で与えられる。

$$q(k+1) = \min\left(\max\left(\sum_{m=1}^{M}\frac{N_m w_m(k)}{\Delta_m} - B\,\Delta, 0\right), L\right) \qquad (4)$$

ただし、$w_m(k) \equiv w_{m,n}(k)$ ($1 \le n \le N_m$) とする。

$w_m(k)$、$q(k)$、$d_m(k)$ の平衡点をそれぞれ w_m^*、q^*、d_m^* とする。式 (1)、(3)、(4) において、$w_m(k+\Delta_m) = w_m(k)$ などとおくことにより、w_m^*、q^*、d_m^* は容易に求めることができる。ここで、スロット k におけるシステムの状態と平衡点の差を

$$\mathbf{x}(k) \equiv \begin{bmatrix} w_1(k) - w_1^* \\ \vdots \\ w_M(k) - w_M^* \\ q(k) - q^* \end{bmatrix}$$

と定義する。

$w_m(k)$ が非線形であるため、これを平衡点の近傍において線形化 [9] し、さらに Δ_L を $\Delta_1 \cdots \Delta_M$ の最小公倍数とすれば、$\mathbf{x}(k+\Delta_L)$ は以下のように書くことができる。

$$\mathbf{x}(k+\Delta_L) = \mathbf{A}\,\mathbf{x}(k) \qquad (5)$$

ここで $\mathbf{A}$ はシステムの安定化行列であり、この行列の固有値によってシステムの安定性や過渡特性が決定される。つまり、式 (5) で与えられる離散時間システムにおいて、平衡点が局所漸近安定となるためには、$D(s) \equiv |s\mathbf{I} - \mathbf{A}| = 0$ によって得られる、特性方程式のすべての解 s_i ($1 \le i \le M+1$) に関して、$|s_i| < 1$ であればよい [10]。なお、ある安定化行列 $\mathbf{A}$ がこの条件をみたすかどうかは、Jury の判別法などを用いて簡単に調べることが可能である [10]。

以下では、例として $M = 2$ (伝搬遅延時間の異なる 2 種類のコネクションが存在)、$\Delta_1 = 1$、$\Delta_2 = 2$ (伝搬遅延時間の比が 1:2) の場合について議論する。ただし、$M > 3$ や伝搬遅延時間の比が異なる場合でも、同様な方法で議論することが可能である。

$M=2$、$\Delta_1=1$、$\Delta_2=2$ の場合、システムの平衡点は以下のようになる。

$$w_1^* = \frac{\gamma_1(v_1+v_2+v_3)}{2v_1} \quad (6)$$

$$w_2^* = \frac{\gamma_2(v_2+v_3)}{v_1} \quad (7)$$

$$q^* = \frac{v_1-v_2+v_3}{4} \quad (8)$$

ただし、

$$\begin{aligned} v_1 &= 2(N_1\gamma_1+N_2\gamma_2) \\ v_2 &= 2B\tau+N_2\gamma_2 \\ v_3 &= \sqrt{(v_1-v_2)^2+8B\tau_1 v_1} \end{aligned}$$

である。ここで、ルータのバッファでのパケットの待ち時間が、伝搬遅延時間に比べて十分に小さい ($B\,\tau_m \to \infty$) とすると、w_m^*、q^* は式 (6)–(8) より以下のように近似できる。

$$\begin{aligned} w_m^* &\simeq \frac{\gamma_m B\,\tau_m}{N_1\gamma_1+N_2\gamma_2} \\ q^* &\simeq N_1\gamma_1+N_2\gamma_2 \end{aligned}$$

これより、ルータの処理能力 B や伝搬遅延時間 τ_m が大きくなれば、平衡点におけるウィンドウサイズ w_m^* は、それに比例して大きくなることが分かる。さらに、コネクション数 N_1 や N_2 が増えれば、平衡点におけるウィンドウサイズ w_m^* は、それに反比例して小さくなることが分かる。また、平衡点におけるバッファ内パケット数 q^* より、各送信側ホストはそれぞれ γ_m 個のバッファを常に占有することがわかる。このため、各送信側ホストの制御パラメータ γ_m の値を変更することにより、平衡点でのバッファ内パケット数を自由に調整することが可能である。

以下、各コネクションのスループットに着目する。平衡点における、伝搬遅延時間が τ_m であるコネクションのスループット ρ_m は以下の式で与えられる。

$$\rho_m \equiv \frac{w_m^*}{r_m} = \frac{w_m^*}{\tau_m+\frac{q^*}{B}}$$

式 (1) および (3) より、平衡点において $d_m^* = \gamma_m^*$ となることが分かる。そこで、式 (3) において $d_m^* = \gamma_m^*$ とおくことにより、

$$w_m^* = \frac{\gamma_m(\tau_m+\frac{q^*}{B})}{\frac{q^*}{B}}$$

が得られる。これより、スループット ρ_m は以下の式で与えられる。

$$\rho_m = \frac{\gamma_m B}{q^*} \quad (9)$$

従って、伝搬遅延時間が異なるコネクションのスループットの比 ρ_1/ρ_2 は γ_1/γ_2 で与えられる。これは、伝搬遅延時間の異なるコネクションのスループットの比は、伝搬遅延時間やコネクション数などには依存せず、単一の制御パラメータ γ_m だけで決まることを意味している。

さらに、平衡点におけるバッファ内パケット数 $q^* > 1$ である状況を考えると、この時ルータの利用率は 1 となることから、ρ_m は以下のように求めることができる。

$$\rho_m = \frac{\gamma_m B}{N_1\gamma_1+N_2\gamma_2}$$

以上の考察から、すべての送信側ホストで γ_m を同じ値に設定すれば、平衡点において各コネクションのスループットを公平とすることが可能であることがわかる。

ただし、このような関係は、本解析のようにコネクションが単一の (ボトルネックとなる) ルータを経由する場合にしか成り立たない。我々のこれまでの研究から、TCP Vegas のスループットは制御パラメータだけでなく、コネクションが経由するルータの数 (ホップ数) にも依存することが分かっている [11]。従って、実際に各コネクションのスループットを公平とするためには、コネクションが経由するルータの段数も考慮して γ_m を決定する必要がある。

$M=2$、$\Delta_1=1$、$\Delta_2=2$ の場合、システムの安定化行列 **A** は以下のようになる。

$$\mathbf{A} = \begin{bmatrix} a_2^2+N_1b_1 & N_2b_1/2 & a_1b_1 \\ 0 & a_2 & b_2 \\ N_1a_1 & N_2/2 & N_1b_1 \end{bmatrix} \quad (10)$$

ただし、a_m、b_m は以下の式で与えられる。

$$a_m = 1-\delta_m+\frac{B\delta_m\tau_m}{B\tau_m+q^*}, \qquad b_m = -\frac{B\delta_m\tau_m w_m^*}{(B\tau_m+q^*)^2}$$

4 数値例

図 2 に、システムの安定条件をみたす δ_1–δ_2 平面の範囲を示す。ここでは、コネクション数 $N_1=N_2=10$、復伝搬遅延時間 $\tau_1=1$ [ms]、$\tau_2=2\tau_1=2$ [ms]、さらに $\gamma_1=\gamma_2=3$ パケットと固定し、ルータの処理能力 B を 2–2,000 と変化させた時の安定領域をプロットしている。なお、$B=20$ は、パケット長を 1 K バイトとすると 163.8 Mbit/s に相当する。この図において、(δ_1, δ_2) が境界線の内側 (原点を含む領域) であればシステムは安定となる。

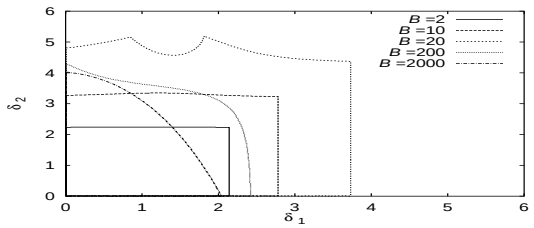

図 2: ルータの処理能力 B を変化させた場合の安定領域 ($N_1=N_2=10$、$\tau_1=1$ ms、$\gamma_1=\gamma_2=3$ パケット)

この図より、システムを安定化できる δ_m の値は、ルータの処理能力によって大きく変化することがわかる。また、システムを安定化できる δ_m の最大値は、伝搬遅延時間の大きいコネクションのほうが大きいことがわかる。この傾向は、ルータの処理能力 B が大きくなるにつれてより顕著になる。例えば、$B=2,000$ [パケット/ms] の場

合には、システムを安定化できる δ_m の最大値は、それぞれおよそ $\delta_1 = 2$、$\delta_2 = 4$ であり、往復伝搬遅延時間の比に等しくなっている。これは、TCP Vegas では RTT ごとにウィンドウサイズを変化させるからである。つまり、伝搬遅延時間の大きいコネクションは、伝搬遅延時間の小さなコネクションに比べてウィンドウサイズを変更する頻度が少ないため、伝搬遅延時間の小さなコネクションよりも大きくウィンドウサイズを変化させても、システムが不安定になりにくいためである。言い換えれば、伝搬遅延時間にかかわらず、すべてのコネクションの δ_m が同じ値であれば、システムの安定性は伝搬遅延時間の小さいコネクションによって決定されることになる。

ルータの処理能力 B が小さくなると、システムを安定化できる δ_m の最大値は、伝搬遅延時間にかかわらずほぼ同じ値となっている。例えば、$B = 2$ の場合、δ_1、δ_2 ともに 2 以下であればシステムは安定となる。これは、ルータの処理能力が非常に低い場合は、伝搬遅延時間に比べて、パケットがルータのバッファで待たされる時間が大きくなるためと考えられる。つまり、ルータのバッファでの待ち時間が大きい場合は、RTT は伝搬遅延時間の差にかかわらずほぼ等しくなる。その結果、ウィンドウサイズを変化させる頻度がほぼ等しくなるため、システムを安定化できる δ_m の最大値もほぼ同じ値になると考えられる。

図 3 に示すように、コネクションの伝搬遅延時間が変化した場合は、ルータの処理能力 B が変化した場合 (図 2) と同じように安定領域が変化する。ここでは、$B = 20$ とし、伝搬遅延時間の比率を 1:2 と固定したまま、往復伝搬遅延時間 τ_1 を 0.1–100 [ms] と変化させている。その他のパラメータについては図 2 と同じ値を用いている。

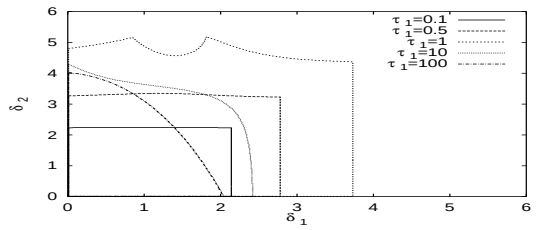

図 3: 伝搬遅延時間 τ_1 を変化させた場合の安定領域 ($B = 20$ パケット/ms、$N_1 = N_2 = 10$、$\gamma_1 = \gamma_2 = 3$ パケット)

図 2 と図 3 を比較すると、伝搬遅延時間が大きくなると、ルータの処理能力が大きくなる場合とまったく同じように安定領域が変化していることがわかる。これは、式 (6)– (8)、(10) において、B および τ_m がすべて $B \times \tau_m$ の積として含まれていることからも分かる。つまり、今対象としているウィンドウ型フロー制御方式にとって、ルータの処理能力 B を大きくすることと、伝搬遅延時間 τ_1 を大きくすることは、まったく同じ意味を持つことになる。

次に、コネクション数 N_1 および N_2 を、それぞれ変化させた時の安定領域を図 4 および図 5 に示す。ここでは、ルータの処理能力 $B = 20$[パケット/ms]、往復伝搬遅延時間 $\tau_1 = 1$ [ms]、$\gamma_1 = \gamma_2 = 3$[パケット] と固定し、N_1 および N_2 をそれぞれ 0–1,000 と変化させている。

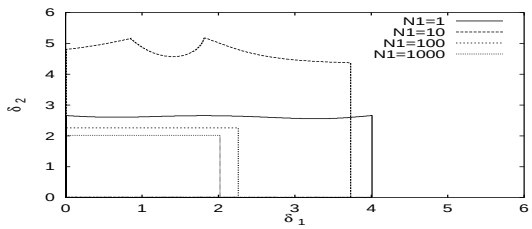

図 4: コネクション数 N_1 を変化させた場合の安定領域 ($B = 20$ パケット/ms、$N_1 = 10$、$\tau_1 = \tau_2 = 1$ ms、$\gamma_1 = \gamma_2 = 3$ パケット)

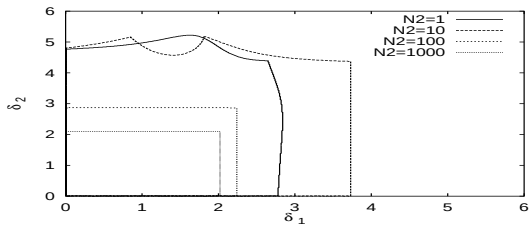

図 5: コネクション数 N_2 を変化させた場合の安定領域 ($B = 20$ パケット/ms、$N_2 = 10$、$\tau_1 = \tau_2 = 1$ ms、$\gamma_1 = \gamma_2 = 3$ パケット)

これらの図から、コネクション数が増えるにつれて、システムを安定化できる δ_m の最大値が小さくなっていることがわかる。例えば、図 4 では、N_1 が大きくなるにつれて δ_1 の最大値が小さくなっている。この現象は以下のようにして説明できる。伝搬遅延時間が同じコネクションは RTT も同じであるため、同期してウィンドウサイズを変化させる。このため、伝搬遅延時間が同じコネクションの数が増えると、システム全体としてウィンドウサイズの変化量が増えることになる。従って、システムを安定化させるためには、コネクション数が増えるにつれて、ウィンドウサイズの変化量を決定する δ_m を小さくする必要があるためである。

しかし、図からわかるように、コネクション数にかかわらず、δ_1、δ_2 が共に 2 以下ならば常にシステムは安定する。現実のネットワークでは、アクティブなコネクション数は時間に応じて変動するが、このような場合でも δ_1 および δ_2 を 2 以下に設定しておけば、システムを常に安定させることが可能である。さらに、図 2–図 5 より、システムパラメータや制御パラメータによらず、常にシステムを安定化できる (δ_1, δ_2) の領域が存在すると予想できる。実際、式 (10) において、ルータの処理能力 B、コネクション数 N_m、伝搬遅延時間 τ_m、制御パラメータ γ_m などを、それぞれ ∞ とした時の安定領域を求めたところ、$\delta_1 < 1$ かつ $\delta_2 < 2$ ならば、システムは常に安定することがわかった。なお、$\delta_1 < 1$ かつ $\delta_2 < 2$ の領域から (δ_1, δ_2) の値を選択するためには、システムの過渡特性をも考慮する必要がある。これについては [2] と同じ手法により、過渡特性が最適となる δ_m の値を決定することが可能であるが、その定量的評価は今後の課題である。

最後に、ウィンドウサイズの変化量を決定する制御パラメータ δ_m が、安定条件を満たす場合と、満たさない場合で、システムのふるまいがどのように変化するかを調

べる。ここではルータの処理能力 $B = 20$[パケット/ms] とし、(δ_1, δ_2) の値として (3.0, 4.0) (図 6)、及び (4.0, 5.0) (図 7) を使用した。その他のパラメータについては図 2 と同じ値を用いている。これらの図では、式 (5) を用いてシステムの状態を数値的に計算した時の、ウィンドウサイズとバッファ内パケット数の変動を示している。なお、ウィンドウサイズ $w_m(k)$ およびバッファ内パケット数 $q(k)$ の初期値として、平衡点の値から 20%離れた値を使用している。

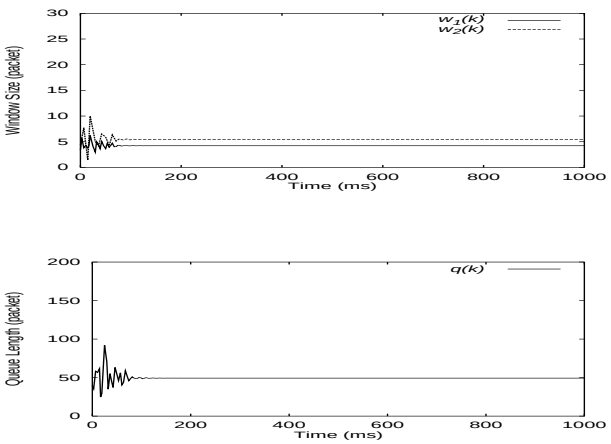

図 6: 安定条件をみたす場合 ($B = 20$ パケット/ms、$N_1 = N_2 = 10$、$\tau_1 = 1$ ms、$\gamma_1 = \gamma_2 = 3$ パケット、$\delta_1 = 3.0$、$\delta_2 = 4.0$)

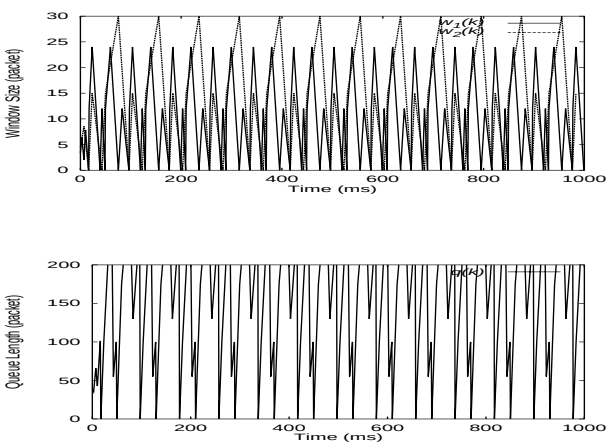

図 7: 安定条件をみたさない場合 ($B = 20$ パケット/ms、$N_1 = N_2 = 10$、$\tau_1 = 1$ ms、$\gamma_1 = \gamma_2 = 3$ パケット、$\delta_1 = 4.0$、$\delta_2 = 5.0$)

δ_m が安定条件をみたす場合 (図 6) では、150 [ms] 程度でシステムは完全に定常状態となり、安定していることがわかる。しかし、δ_m が安定条件をみたさない場合 (図 7) は、ウィンドウサイズ、バッファ内パケット数ともに激しく振動しており、不安定な動作となっている。

なお、本解析の妥当性を検証するために、シミュレーションとの比較によって、解析で得られた安定領域がどの程度一致しているかを調べた。紙面の都合上、実際のシミュレーション結果についてはここでは省略するが、その結果、δ_m に関して ± 1 程度の誤差で一致していた。

5 まとめと今後の課題

本稿では、TCP Vegas の輻輳回避機構を用いたウィンドウ型フロー制御方式を対象として、その安定性を制御理論を用いて明らかにした。特に、各コネクションの伝搬遅延時間が異なる、ネットワークモデルを対象とした解析を行った。

今後の課題としては、安定性だけでなく過渡特性をも考慮して制御パラメータを決定する必要がある。特に、各送信側ホストにおいてウィンドウサイズの初期値が異なる場合に、良好な過渡特性を得るための制御パラメータ決定方法に関しての検討が必要である。

参考文献

[1] 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.

[2] 大崎 博之, 村田 正幸, 潮 俊光, 宮原 秀夫, "制御理論を用いた TCP/IP ネットワークにおけるウィンドウ型フロー制御方式の特性解析," 電子情報通信学会技術研究報告 (IN98-101), pp. 91–98, October 1998.

[3] O. A. Hellal and E. Altman, "Analysis of TCP Vegas and TCP Reno," in *Proceedings of IEEE ICC '97*, pp. 495–499, June 1997.

[4] A. Kumar, "Comperative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 485–498, August 1998.

[5] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and TCP Vegas," in *Proceedings of IEEE INFOCOM '99*, March 1999.

[6] B.-K. Kim and C. Thompson, "Optimal feedback control of ABR traffic in ATM networks," in *Proceedings of IEEE GLOBECOM '98*, pp. 844–848, 1998.

[7] H. Zhang and O. W. Yang, "Design of robust congestion controllers for ATM networks," in *Proceedings of IEEE INFOCOM '97*, pp. 302–309, April 1997.

[8] A. Kolarov and G. Ramamurthy, "A control theoretic approach to the design of closed-loop rate based flow control for high speed ATM networks," in *Proceedings of IEEE INFOCOM '97*, pp. 293–301, April 1997.

[9] 平井 一正, 池田 雅夫, 非線形制御システムの解析. オーム社, 1986.

[10] R. Isermann, *Digital control systems, Volume 1: fundamentails, deterministic control*. Springer-Verlag Berlin Heidelberg, 1989.

[11] H. Ohsaki, M. Murata, and H. Miyahara, "Analysis of window-based flow control mechanism in heterogeneous network environment," in preparation, 1999.