Vol. 39 No. 10 情報処理学会論文誌 Oct. 1998

制御理論を用いた TCP Vegas にもとづくウィンドウ型フロー制御方式の特性解析

大崎博之† 村田正幸†
潮俊光† 宮原秀夫†

パケット交換ネットワークにおいて、データ転送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。フィードバック型の輻輳制御では、ネットワークからのフィードバック情報に応じて、送信側ホストからのトラヒック流入量を動的に制御する。これにより、ネットワーク内部でのパケット棄却を防ぐとともに、網資源の有効利用が可能となる。現在、広く普及している TCP/IP ネットワークでは、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) が用いられているが、その改良に関する研究も盛んに行われている。その中でも、高い性能を示すものとして TCP Vegas が最近注目されている。そこで本稿では、TCP Vegas をもとにしたウィンドウ型のフロー制御方式を対象とし、その安定性と過渡特性を制御理論を用いて明かにする。さらに、解析結果に基いて、制御パラメータの最適化制御を行うことにより、システムの性能が大幅に改善されることを示す。

A Control Theoretical Analysis of Window-Based Flow Control Mechanism based on TCP Vegas

Hiroyuki Ohsaki ,† Masayuki Murata ,† Toshimitsu Ushio † and Hideo Miyahara†

A feedback-based congestion control mechanism is essential to provide efficient data transfer services in a packet-switched network. The feedback-based congestion control mechanism regulates traffic flows from source terminals. With an appropriate congestion control mechanism, it is possible to prevent packet losses in the network, and to utilize network resources effectively. A window-based flow control mechanism called TCP (Transmission Control Protocol), a sort of feedback-based congestion control mechanisms, has been widely used in current TCP/IP networks. Recently, TCP Vegas has been proposed, which is another version of the TCP mechanism and has potential to achieve much better performance than current TCP Tahoe and Reno. In this paper, we focus on a window-based flow control mechanism based on the congestion avoidance mechanism of the TCP Vegas, and analyze its stability and transient behavior based on a control theoretic approach. We further demonstrate that its performance can be dramatically improved by dynamically adjusting its control parameters based on our analytic results.

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。フィードバック型の輻輳制御では、ネットワークからのフィードバック情報に応じて、送信側ホストからのトラヒック流入量を動的に制御することにより、ネットワーク内部でのパケット棄却を防ぐとともに、ネットワーク資源の有効利用を可能とする。現在広く普及している TCP/IP ネットワークでは、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棄却された場合に、棄却さ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、ウィンドウサイズの調整によって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。

BSD UNIX に実装されており、現在広く使用されている TCP (TCP Reno) では、ネットワーク内でのパケット棄却の発生をフィードバック情報として、

† 大阪大学 大学院基礎工学研究科
Graduate School of Engineering Science, Osaka University

送信側ホストがウィンドウサイズの変更を行う[6),9)]。TCP Reno では、基本的にパケット棄却が発生しない限り、送信側ホストは連続的にウィンドウサイズを増加させる。やがてウィンドウサイズが利用可能な帯域を超えると、余分なパケットはネットワーク内のルータ中のバッファに蓄えられる。さらにウィンドウサイズが増加すると、ルータにおいてバッファあふれが発生し、その結果パケットが廃棄される。送信側ホストは、タイムアウトなどによりパケット棄却を検出し、ウィンドウサイズを減少する。ウィンドウサイズを減少したことにより、パケット棄却が発生しなくなれば、再びウィンドウサイズを連続的に増加させるといった制御を行う。このように、TCP Reno では、少しづつウィンドウサイズを増加させてゆき、パケット棄却が発生した時点でウィンドウサイズを減少させるという制御を行うことにより、ネットワークの輻輳制御を行う。

最近、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),2)]。TCP Vegas では、TCP Reno に比べて、以下の点が改良されている。(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。これにより、TCP Reno と比較して、スループットが 37–71 % 改善され、再送されるパケット数が 1/5 から 1/2 程度に減少することが報告されている[2)]。特に、(2) の輻輳回避機構では、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、すなわちラウンドトリップ時間 (Round-Trip Time) を測定し、これを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。つまり、TCP Vegas では、パケットを転送するために要したラウンドトリップ時間を測定することによって、ルータのバッファ内に蓄えられているパケット数を推測し、この値が一定となるようにウィンドウサイズの調整を行う。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棄却を待つ必要がないため、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。

従来の研究では、ウィンドウサイズの増加量や減少量などの制御パラメータをアドホックに定めた上で性能評価を行い、良い結果が得られることを示しているものがほとんどである。しかし、ネットワークの高効率化のためには、ネットワークの過渡特性や安定性などの観点から、制御パラメータを決定するための理論的な裏付けが不可欠である。データ系のネットワークは基本的にフィードバック系であるため、制御理論の適用によってデータ系ネットワークの理論的な枠組を構築できると考えられる。

そこで本稿では、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動作を、制御理論を用いて解析する。ただし、TCP Vegas の輻輳回避機構では、ラウンドトリップ時間ごとに 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、本稿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与えるようなモデルを扱う。また、文献[5)] では、TCP Vegas の公平性と安定性を数学的解析手法を用いて明らかにし、TCP Vegas の性能を改善する方式を提案している。これは、TCP Vegas がウィンドウサイズを変化させるアルゴリズムを改良し、ウィンドウサイズが一定となる領域をなくすというものである。本稿で評価の対象とするウィンドウフロー型輻輳制御方式は、文献[5)] で提案されている方式と同じように、ウィンドウサイズが一定となる領域をなくしたものを用いる。

このような、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、その特性を制御理論を用いて明かにする。まず、定常状態においてウィンドウサイズが収束するための条件 (システムの安定条件) を導出する。いくつかの数値例を用いて、コネクション数や、伝搬遅延時間といったネットワークのパラメータと、システムの安定性との関連について検討を行う。さらに、システムの過渡特性に着目し、ウィンドウサイズの変化量を決定するパラメータをどのように選択すれば、安定性を保ちながら最適な過渡特性が得られるのかを明らかにする。

なお、制御理論を用いてフィードバック型の輻輳制御方式を解析したものとしては、文献[4),7),8),10)] などが存在する。しかし、これらはすべてレート型の輻輳制御方式を対象としており、TCP Vegas のようなウィンドウ型の輻輳制御方式には適用できない。また、ほとんどの場合、使用されている輻輳制御方式 (レート決定アルゴリズム) は線形の単純なものである。本稿では、パケットのラウンドトリップ時間を測定し、ネットワーク内のバッファに蓄えられているパケット数を推測するという、TCP Vegas の輻輳回避機構に基づいたウィンドウフロー型の輻輳制御方式に対して解析を行う。

以下、2 章において、本稿で用いる解析モデルおよび対象とするウィンドウフロー型の輻輳制御方式を説明する。また、3 章において、対象とするウィンドウ

フロー型輻輳制御方式の安定性解析を行い、さらに制御パラメータと安定性との関連について検討を行う。さらに 4 章において、システムの過渡特性を最適化する制御パラメータを導出する。5 章では、本稿で得られた解析結果に基いて、制御パラメータの最適化制御を行うことによって、システムの性能が大幅に改善されることを示す。最後に、6 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

まず、TCP Vegas の輻輳回避メカニズムについて簡単に説明する。詳細なアルゴリズムについては、文献 [2] を参照されたい。まず、各送信側ホストは、ネットワークが輻輳していない時のラウンドトリップ時間 τ を保持している。バッファにおける待ち時間がないため、これは往復伝搬遅延時間に相当する。この値には、通常測定したラウンドトリップ時間のうち最小のものを用いる。送信側ホストは、1 ラウンドトリップ時間中に、ウィンドウサイズ w 分のデータを送出することができるため、ネットワークが輻輳していなければ、スループットは w/τ となるはずである。実際にパケットを送出し、そのパケットの ACK パケットを受信するまでの時間を測定することによって、実際のラウンドトリップ時間 r を得る。このラウンドトリップ時間中に送出したデータ量が $\overline{w}$ であれば、実効スループットは $\overline{w}/r$ となる。そこで、予想されるスループットの差と、実行スループットの差 d を以下のように計算する。

$$d = \frac{w}{\tau} - \frac{\overline{w}}{r}$$

そして、d の値を 2 つの制御パラメータ α および β と比較し、その大小関係によってウィンドウサイズ w を増減する。もし $d < \alpha$ ならば、次のラウンドトリップ時間でウィンドウサイズを 1 パケット分だけ線形的に増加させる。一方、$\beta < d$ ならば、次のラウンドトリップ時間でウィンドウサイズを 1 パケット分だけ線形的に減少させる。

図 1 に、本稿で用いる解析モデルを示す。N 個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。TCP Vegas では、ラウンドトリップ時間を単位としてウィンドウサイズを変化する。そこで本解析では、ラウンドトリップ時間を単位として時間をスロットに分割し、各スロットにおけるシステムの状態を離散時間モデルで考える。

スロット k における、送信側ホスト $n\,(1 \le n \le N)$ のウィンドウサイズを $w_n(k)$ とする。これは、送信側ホスト n は、スロット k において、ネットワーク内に $w_n(k)$ 個のパケットを送出することが可能であることを意味する。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、各スロットにおいて必ず $w_n(k)$ 個のパケットを送出すると仮定する。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、送信側ホストのウィンドウサイズ $w_n(k)$、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 として、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。

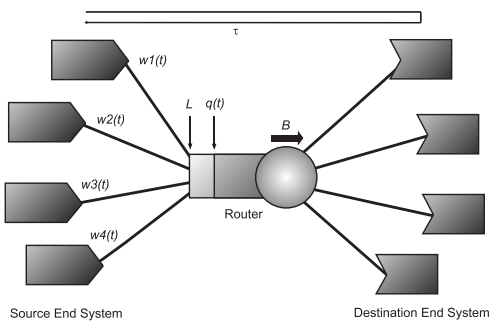

図 1 解析モデル
Fig. 1 The analytic model.

ウィンドウ型のフロー制御では、ラウンドトリップ時間中に、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、全てのコネクションのラウンドトリップ時間が等しいと仮定すれば、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。

$$q(k+1) = \min(\max(\sum_{n=1}^{N} w_n(k) - B\,r(k), 0), L)$$

ここで $r(k)$ はスロット k におけるラウンドトリップ時間である。

TCP Vegas では、ラウンドトリップ時間を測定し、この値に応じてウィンドウサイズを変更する。送信側ホスト n では、スロット k において測定されたパケッ

トのラウンドトリップ時間 $r(k)$ に基いて、以下の値 $d(k)$ を計算する。

$$d(k) = \frac{w_i(k)}{\tau} - \frac{w_i(k)}{r(k)} \quad (1)$$

ここで、τ はバッファにおける待ち時間を除いたラウンドトリップ時間 (往復伝搬遅延時間) である。実際にはラウンドトリップ時間 $r(k)$ は、ルータのバッファ内に存在するパケットの個数によって変化し、以下の式で与えられる。

$$r(k) = \tau + \frac{q(k)}{B}$$

TCP Vegas では、$d(k)$ の値に基いて、ラウンドトリップ時間ごとにウィンドウサイズを変更する。具体的には、送信側ホスト n のウィンドウサイズ $w_n(k+1)$ は以下のように変更される。

$$w_n(k+1) = \begin{cases} w_n(k)+1 & \text{if } d(k) < \alpha \\ w_n(k)-1 & \text{if } d(k) > \beta \\ w_n(k) & \text{otherwise} \end{cases} \quad (2)$$

ここで α および β は、ラウンドトリップ時間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) において、$w_n(k+1)$ の計算方法を以下のように変更したモデルを扱う。

$$w_i(k+1) = \max(w_n(k) + \delta(\gamma - d(k)), 0) \quad (3)$$

ここで δ は、ラウンドトリップ時間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。

TCP Vegas では、$d(k)$ の値が $[\alpha, \beta]$ の時はウィンドウサイズを変更しないが、このためにコネクション間の公平性が低下してしまうことが知られている [5]。そこで式 (2) において、ウィンドウサイズが固定されてしまう場合をなくすために、α および β のかわりに単一の制御パラメータ γ を導入している。これにより、コネクション間の公平性の向上が可能となる。また、ウィンドウサイズの変化量を決める制御パラメータ δ を導入し、制御理論の適用を可能とするとともに、過渡特性の改善を図っている。

3. 安定性解析

以下では、全ての送信側ホストのウィンドウサイズの初期値は等しく、また全ての送信側ホストは式 (3) に従ってウィンドウサイズを変更すると仮定する。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。

$$q(k+1) = \min(\max(N\,w(k) - B\,r(k), 0), \quad (4)$$

ただし、

$$w(k) \equiv w_n(k), \qquad 1 \le n \le N$$

とする。

まず、$w(k)$、$q(k)$、$d(k)$ の平衡点をそれぞれ w^*、q^*、d^* とする。式 (1)、(3)、(4) において、$w(k+1) = w(k)$ 等とおき、式 (3) において

$$B\,r(k) \simeq B\tau$$

と仮定することにより、w^*、q^*、d^* は以下のように求められる。

$$w^* = \tau\left(\frac{B+\gamma N}{N}\right) \quad (5)$$
$$q^* = \gamma N \tau$$
$$d^* = \gamma$$

次に、$w(k)$ 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する [11]。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、

$$\mathbf{x}(k) = \begin{bmatrix} w(k) - w^* \\ q(k) - q^* \end{bmatrix}$$

とすれば、$\mathbf{x}(k+1)$ は以下の式で与えられる。

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) \quad (6)$$

ただし、

$$\mathbf{A} = \begin{bmatrix} 1 - \frac{\delta}{\tau} + \frac{B\delta}{(B+\gamma N)\tau} & -\frac{B\delta}{N(B+\gamma N)\tau} \\ N & 0 \end{bmatrix}$$

式 (1)、(3)、(4) で与えられる離散時間システムにおいて、平衡点 (w^*, q^*) が局所漸近安定となるためには、

$$D(s) = |s\mathbf{I} - \mathbf{A}| = 0 \quad (7)$$

によって得られる特性方程式の解 $s_i (i = 1, 2)$ に関して、$|s_i| < 1$ であればよい。ここで、特性方程式 $D(s)$ は 2 次式であるため、$|s_i| < 1$ は以下の条件式と同値である。

$$D(1) > 0$$
$$D(-1) > 0$$
$$|D(0)| < 1$$

これより、システムの平衡点が局所漸近安定となるための条件は以下の式で与えられる。

$$\delta > 0 \quad (8)$$
$$\frac{\delta(B-\gamma N)}{(B+\gamma N)\tau} + 2 > 0 \quad (9)$$
$$\frac{B\delta}{(B+\gamma N)\tau} < 1 \quad (10)$$

図 2 に、システムの安定条件をみたす δ の範囲を示す。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 $\tau = 1$ ms、さらに $\gamma = 3$ パ

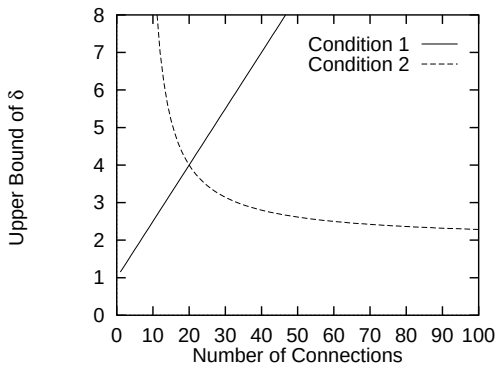

図 2 安定条件をみたす δ の範囲 ($B = 20$ パケット/ms, $\tau = 1$ ms, $\gamma = 3$ パケット)
Fig. 2 Stability condition for δ ($B = 20$ packet/ms, $\tau = 1$ ms, $\gamma = 3$ packet)

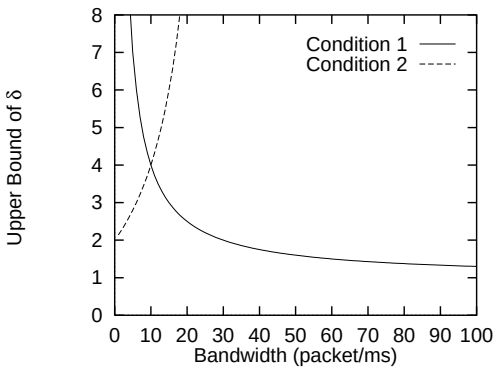

図 3 ルータの処理能力を変化させた時の δ の範囲 ($N = 10$, $\tau = 1$ ms, $\gamma = 3$ パケット)
Fig. 3 Stability condition for δ ($N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet)

ケットと固定し、コネクション数 N を 1–100 と変化させた時の、式 (9) および式 (10) の値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。図中の「Condition 1」および「Condition 2」は、それぞれ式 (10) および式 (9) に対応する。この図において、δ の値が「Condition 1」および「Condition 2」の両方よりも小さい場合に、平衡点においてシステムが安定する。

これより、コネクション数が少ない (約 20 以下) 時には、コネクション数が増えるにつれて、システムが安定化する δ の範囲が広くなっていることがわかる。しかし、コネクション数がさらに増えると、δ の取りうる範囲が逆に小さくなっている。このため、コネクション数が変化するようなネットワーク環境下では、δ の値を注意深く決定する必要がある。例えば、ここで用いたパラメータの場合には、$\delta = 1$ と設定しておけば、たとえコネクション数が変動したとしても常にシステムの安定性を保つことが可能である。しかし、$\delta = 3$ のように設定してしまうと、コネクション数の変動によってシステムが不安定となる状況が発生してしまう。

図 3 に、コネクション数 $N = 10$ と固定し、ルータの処理能力 B を 1–100 パケット/ms と変化させた時の、システムが安定となる δ の範囲を示す。ここでは、図 2 と同じく、$B = 20$ パケット/ms、$\tau = 1$ ms、$\gamma = 3$ パケットと設定している。この図より、適切な δ の値は、送信側ホストの数だけでなく、ルータの処理能力にも依存していることがわかる。従って、パケットが処理能力の異なる複数のルータを経由するような場合には、最も処理能力の高い、もしくは最も処理能力の低いルータにあわせて δ の値を設定する必要がある。

次に、ウィンドウサイズの変化量を決定するパラメータである δ が、安定条件をみたすパラメータを取る場合、または安定条件をみたさないパラメータを取る場合に、システムがどのように動作するかについて検討を行う。$B = 20$ パケット/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ パケットといったパラメータの場合には、式 (8)–(10) の安定条件より、$0 < \delta < 2.5$ である必要がある。そこで、図 4 および図 5 に、それぞれ $\delta = 2.4$ および $\delta = 2.6$ の時の、ウィンドウサイズ $w(k)$ およびルータのバッファ内パケット数 $q(k)$ の時間的変動を示す。なお、ウィンドウサイズおよびバッファ内パケット数の初期値として、平衡点の値 w^* および q^* から 20 % 離れた値を用いている。

δ が安定条件をみたす場合 (図 4) には、700 ms 程度でウィンドウサイズ、バッファ内パケット数ともに一定値に収束し、システムの動作が安定していることがわかる。しかし、δ の値がわずかに大きくなり、$\delta = 2.6$ となって安定条件をみたさなくなると、図 5 に示すようにシステムの動作が不安定となる。また、最大バッファ内パケット数に着目すると、システムが安定となる場合 (図 4) は 50 パケット程度であるのに対して、システムが不安定となる場合 (図 5) は 130 パケット程度と増加している。このため、バッファ内での待ち時間により、パケットの転送遅延時間が大幅に (この例では 4 ms) 増加してしまう。さらに、図 5 では、システムが不安定となり、ウィンドウサイズが大きく振動しているため、ルータのバッファが空になる時間が存在している。このことから、システムが不安定になるとネットワークのスループットが低下してしまうことがわかる。

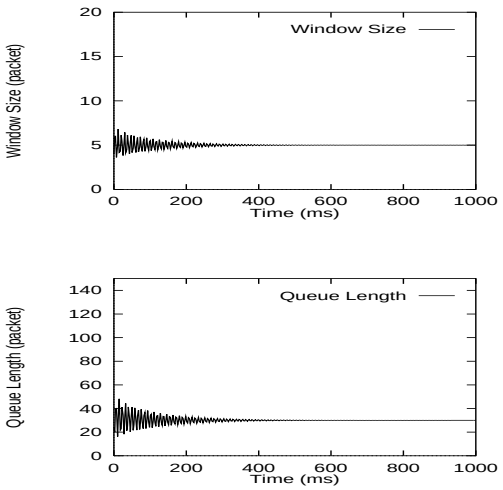

図 4 δ が安定条件をみたす場合 ($\delta = 2.4$, $B = 20$ パケット/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ パケット)
Fig. 4 Case of appropriate δ ($\delta = 2.4$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet)

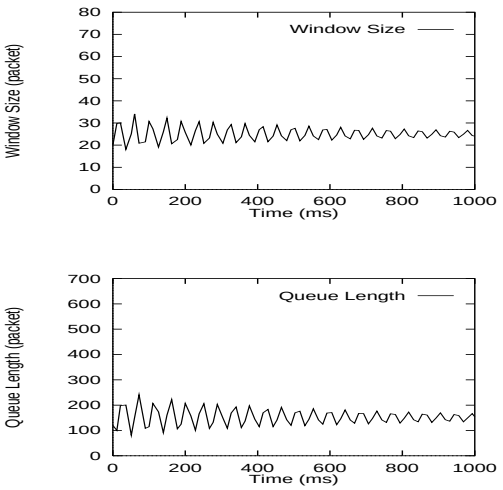

図 6 安定条件をみたす δ の場合 ($\delta = 12$, $B = 20$ packet/ms, $N = 10$, $\tau = 5$ ms, $\gamma = 3$ パケット)
Fig. 6 Case of appropriate δ ($\delta = 12$, $B = 20$ packet/ms, $N = 10$, $\tau = 5$ ms, $\gamma = 3$ packet)

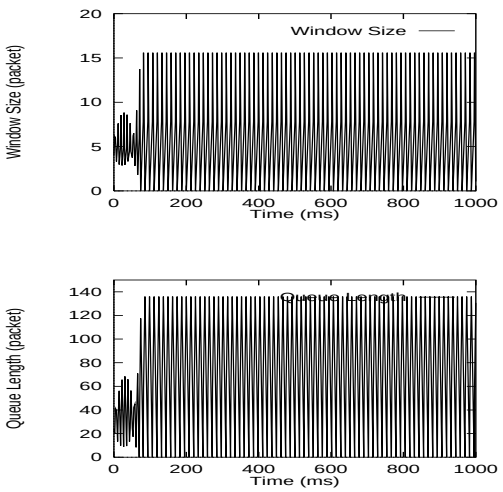

図 5 δ が安定条件をみたさない場合 ($\delta = 2.6$, $B = 20$ パケット/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ パケット)
Fig. 5 Case of inappropriate δ ($\delta = 2.6$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet)

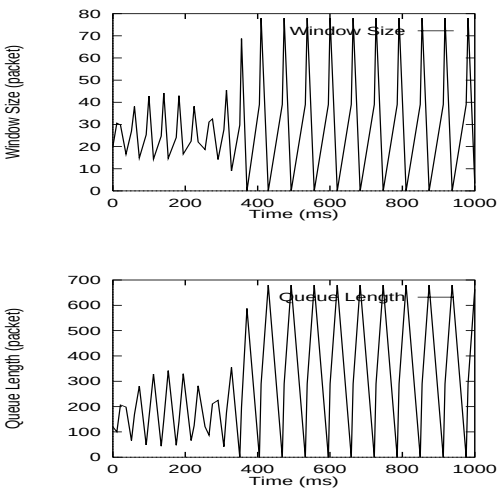

図 7 安定条件をみたさない δ の場合 ($\delta = 13$, $B = 20$ パケット/ms, $N = 10$, $\tau = 5$ ms, $\gamma = 3$ パケット)
Fig. 7 Case of inappropriate δ ($\delta = 13$, $B = 20$ packet/ms, $N = 10$, $\tau = 5$ ms, $\gamma = 3$ packet)

このような現象は、伝搬遅延時間が増加するとさらに顕著となる。図 6 および図 7 に、往復伝搬遅延時間 $\tau = 5$ ms とした時のウィンドウサイズおよびバッファ内パケット数の変動を示す。式 (8)–(10) の安定条件より、$0 < \delta < 12.5$ の時システムがが安定化する。そこで、図 6 および図 7 では、それぞれ $\delta = 12$ (安定条件をみたす場合) および $\delta = 13$ (安定条件をみたさない場合) を用いている。

図 6 では、δ の値が安定化条件をみたすため、収束の速度は遅いけれども徐々に安定する状態に近付いていることがわかる。さきほどの $\tau = 1$ ms の場合と比べて収束が遅くなっているのは、TCP Vegas ではラウンドトリップ時間ごとにウィンドウサイズを変化させるため、伝搬遅延時間が大きくなるとそれだけウィンドウサイズの制御頻度が少なくなるためである。

δ の値が安定条件をみたさない、図 7 の場合には、

ウィンドウサイズ、バッファ内パケット数ともに大きく振動していることがわかる。システムが不安定なために、ウィンドウサイズが 0 になるような時間も存在している。このような場合、送信側ホストからのパケット転送が、送信側ホストのバッファで待たされてしまうことになる。また、最大バッファ内パケット数が 670 程度と、非常に大きくなっている。このため、ルータにおけるパケット棄却を防ぐためには、さきほどの例と比べてルータに大量のバッファが必要となる。

このように、ウィンドウサイズの変化量を決定する δ の選択によって、システムの安定性が大きく変化することがわかる。特に、伝搬遅延時間が大きいような状況では、δ を適切に選ばなければネットワークが不安定となり、その性能が大きく劣化してしまう。また、伝搬遅延時間が大きくなると、それにともなって収束の速度が遅くなってしまう。δ の値が、式 (8)–(10) の安定条件をみたせば平衡点は必ず漸近安定となるが、より高効率のネットワークを考えると、安定条件だけでなく、過渡状態における収束の速度にも注意して δ の値を決定する必要がある。そこで、4 章では、システムの過渡特性を改善するために、どのような制御パラメータを選ぶべきかについて検討を行う。

4. 過渡特性解析

式 (1)、(3)、(4) で与えられる離散時間システムの過渡特性は、その特性方程式である $D(s)$ (式 (7)) の解によって決定される。まず、特性方程式 $D(s)$ の解 s_i は、以下の式で与えられる。

$$s_i = \frac{1}{2(B+\gamma N)\tau}\left(-\delta\gamma N + B\tau + \gamma N\tau \pm \sqrt{-4B\delta\tau(B+\gamma N) + (\delta\gamma N - B\tau - \gamma N\tau)^2}\right)$$

システムの収束の速度は $|s_i|$ (s_i は複素数であるので、ユークリッド空間におけるベクトル長) によって決まる。つまり、$|s_i|$ が 0 に近いほど過渡状態における収束の速度が速くなる。このため、ウィンドウサイズの変化量を決定するパラメータである δ には、過渡状態における収束の速度を決定する

$$|s| \equiv max(|s_1|, |s_2|) \tag{11}$$

を最小化し、なおかつ式 (8)–(10) の安定条件をみたすような値を選べばよい。

図 8 に、δ の値を変化させた時に、その値に対応する $|s|$ の値がどのように変化するかを示す。なお、ここではバッファ内パケット数 $B = 20$ パケット/ms および $\gamma = 3$ パケットと固定し、伝搬遅延時間 τ を 1、3、5、10 ms と変化させている。これより、ある

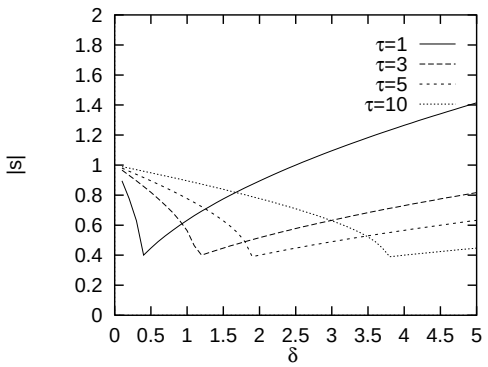

図 8 δ と $|s|$ の関係 ($B = 20$ パケット/ms, $N = 10$、$\gamma = 3$ パケット)
Fig. 8 Relation between δ and $|s|$ ($B = 20$ packet/ms, $N = 10$、$\gamma = 3$ packet)

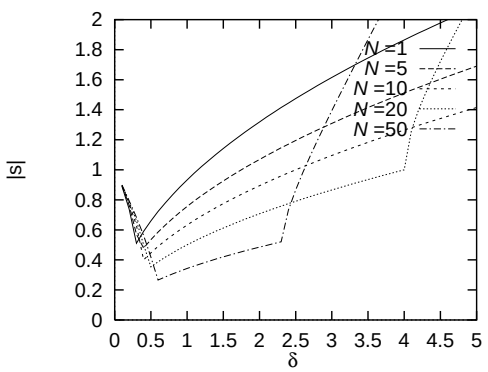

図 9 δ と $|s|$ の関係 ($B = 20$ パケット/ms, $\tau = 1$ ms、$\gamma = 3$ パケット)
Fig. 9 Relation between δ and $|s|$ ($B = 20$ packet/ms, $\tau = 1$ ms, $\gamma = 3$ packet)

τ の値に対して、$|s|$ を最小化する δ の値が存在することがわかる。さらに、$|s|$ を最小化する δ の値は、τ によって変化することもわかる。数値的に計算することによって、例えば、$\tau = 1$ ms の時は $\delta = 0.397$、$\tau = 5$ ms の時は $\delta = 1.877$ において $|s|$ が最小値をとることがわかる。また、図 9 に、$\tau = 1$ ms と固定し、コネクション数を 1、5、10、20、50 と変化させた時の、δ と $|s|$ との関係を示す。この図より、$|s|$ を最小化する δ の値は、コネクション数が変わってもそれほど変化していないことがわかる。

次に、過渡特性を考慮して δ を適切に選ぶことによって、システムの性能がどのように改善されるかを図 10 および図 11 に示す。図 10 では、図 4 における δ を 2.4 から 0.4 に変更したものである。また、図 11 では、図 6 における δ を 12 から 1.9 へと変化させている。なお、どちらの場合でも、δ はシステムの安

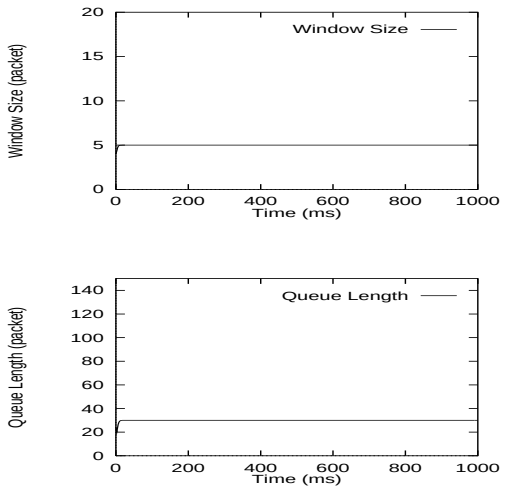

図 10 δ を適切に選んだ場合 ($B = 20$ パケット/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ パケット, $\delta = 0.4$)
Fig. 10 Case of appropriate δ ($B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet, $\delta = 0.4$)

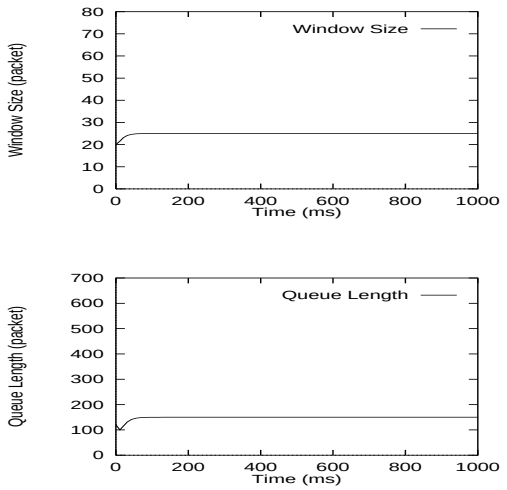

図 11 δ を適切に選んだ場合 ($B = 20$ パケット/ms, $N = 10$, $\tau = 5$ ms, $\gamma = 3$ パケット, $\delta = 1.9$)
Fig. 11 Case of appropriate δ ($B = 20$ packet/ms, $N = 10$, $\tau = 5$ ms, $\gamma = 3$ packet, $\delta = 1.9$)

定条件をみたしている。

これらの図より、過渡特性をも考慮して δ の値を決定することにより、安定性をみたしながら、システムの過渡特性を大幅に改善できることがわかる。例えば、図 6 と図 11 を比較すると、最大バッファ内パケット数が 250 パケットから 150 パケットへと減少している。また、図 6 では 1000 ms 内ではシステムが安定していないが、図 11 の場合には 50 ms 程度で完全に安定している。

ただし、実際のネットワークにおいては、アクティブなコネクション数や伝搬遅延時間は時間に応じて変化する。このため、あるパラメータ条件下で最適となるように δ を設定したとしても、ネットワークの状態が変化することによって δ の値が不適切なとなる可能性がある。図 8 からわかるように、最適な δ の値は伝搬遅延時間に大きく依存している。そこで、例えば最も伝搬遅延時間が小さい状況にあわせて δ の値を設定すれば、伝搬遅延時間が大きい状況でも性能劣化をある程度抑えることが可能である。しかし、伝搬遅延時間が大きい状況にあわせて δ の値を設定すれば、伝搬遅延時間が小さい状況でシステムの動作が不安定になると考えられる。

そこで 5 章では、ネットワークの状況が変化した場合にも、δ の値が常に最適となるような制御を行うことで、システムの性能が大きく向上できることを示す。

5. 制御パラメータの最適化制御

3 章の安定性解析および 4 章の過渡特性解析により、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どのように設定するかによって、システムの性能が大きく変化することがわかった。システムが効率的に動作するためには、制御パラメータの値は式 (8)–(10)、(11) に基づいて決定されなければならない。ところが、実際のネットワークにおいては、ほとんどのシステムパラメータは固定されていない。例えば、アクティブなコネクション数は時間に応じて変化するし、ネットワーク内でのルーティング処理などによって、ボトルネックとなるルータや、伝搬遅延時間 (ラウンドトリップ時間からルータでの待ち時間を除いたもの) も変化してしまう。このため、実際のネットワークへの適用を考えると、ネットワークの状況に応じて制御パラメータを適切に変化させる必要がある。そこで本章では、これまでの解析結果を利用して、送信側ホストにおける制御パラメータを最適化する制御を提案し、その性能評価を行う。

ここでは、送信側ホストにおいてウィンドウサイズの変化量を決定する制御パラメータである δ を、ネットワークの状況に応じて適切に変化させることを考える。安定条件 (式 (8)–(10)) および過渡特性の結果 (式 (11)) より、最適な δ の値はルータの処理能力 B、コネクション数 N、伝搬遅延時間 τ、および制御パラメータ γ に依存することが分かる。なお、ここで γ は送信側ホストにおいて設定される制御パラメータであり、また、対象とするシステムでは送信側ホストは

常に τ の値を保持している。このため、送信側ホストにおいて、最適な δ の値を計算するためには、ルータの処理能力 B とコネクション数 N を何らかの方法で推測すればよい。

送信側ホストにおいて、ボトルネックとなっているルータの処理能力 B を推測する方法として、パケット・ペアと呼ばれる方式が知られている [6),7)]。この方式では、送信側ホストが 2 つのパケットを連続して送出し、それらのパケットに対する ACK パケットの到着間隔を計測する。そして、連続するパケットの間隔の最小値はボトルネックとなるルータの処理能力によって決まる、という性質を利用してルータの処理能力を推測する。

一方、コネクション数 N については、システムが定常状態となり安定していれば、ウィンドウサイズの値 $w_n(k)$ から推測することが可能である。つまり、定常状態におけるウィンドウサイズとコネクション数の関係は、式 (5) で与えられるため、コネクション数 N を以下のように計算することができる。

$$N = \max(\frac{B\,\tau}{w^* - \gamma\tau}, 1) \qquad (12)$$

このようにして推測したバッファサイズ B およびコネクション数 N を用いて、送信側ホストにおいて最適な制御パラメータ δ を計算する。具体的には、測定したルータの処理能力 B、コネクション数 N、伝搬遅延時間 τ のいずれか変化するたびに、安定条件 (式 (8)–(10)) を満たし、なおかつ過渡特性を最適化する (式 (11) を最小化する) δ の値を計算する。なお最適な δ の値を決定するためには、式 (11) を最小化する δ の値を求める必要があるが、これには例えば Brent の手法を用いて数値的に計算することができる [3)]。

まず、コネクション数 N が変化した時に、δ の最適化を行わなければシステムの性能がどのように変化するかを示す。図 12 および図 13 に、伝搬遅延時間 $\tau = 5$ ms の場合に、δ の最適化制御を行う場合、および行わない場合に、コネクション数が変動した時のシステムの動作を示す。これらの図では、図 11 と同じシステムパラメータを用いているが、$t = 300$ ms から $t = 700$ ms の間はコネクション数を $N = 10$ から $N = 1$ へと変化させている。δ の最適化制御を行わない場合 (図 12) には、コネクション数 $N = 10$ の時の最適値である $\delta = 1.9$ を用いている。また、δ の最適化制御を行う場合 (図 13) では、10 ms ごとに式 (12) を用いてアクティブなコネクション数を予測し、この値に基いて式 (11) を最小化する δ の値を数値的に計算している。

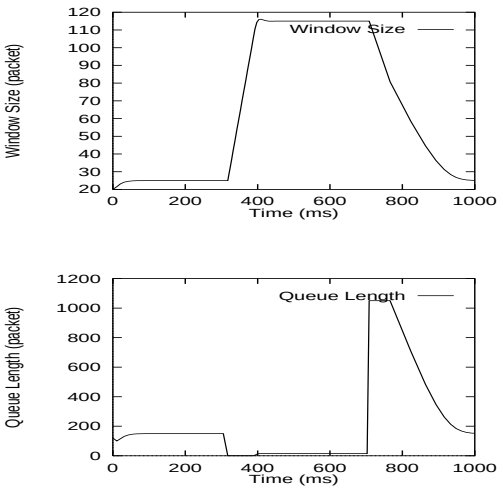

図 12 コネクション数が変化した場合 ($B = 20$ パケット/ms, $\tau = 5$ ms, $\gamma = 3$ パケット, $\delta = 1.9$ 固定)
Fig. 12 Effect of Number of Connections Variation ($B = 20$ packet/ms, $\tau = 5$ ms, $\gamma = 3$ packet, $\delta = 1.9$)

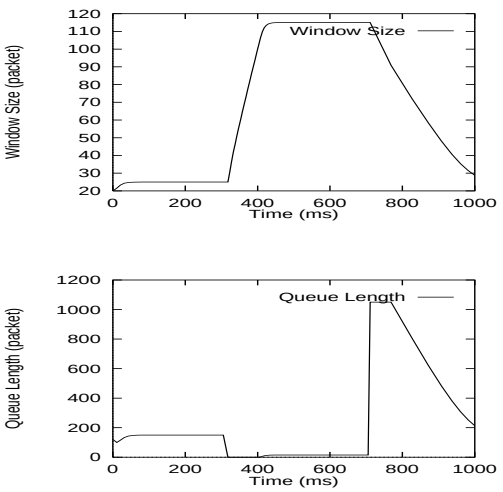

図 13 コネクション数が変化した場合 ($B = 20$ パケット/ms, $\tau = 5$ ms, $\gamma = 3$ パケット)
Fig. 13 Effect of Number of Connections Variation ($B = 20$ packet/ms, $\tau = 5$ ms, $\gamma = 3$ packet)

これらの結果を比較すると、アクティブなコネクション数を予測し δ の最適化制御を行った場合、わずかに過渡特性が改善されるが、δ の最適化制御を行わない場合と比較してもそれほど大きな違いがないことがわかる。これは、図 9 からわかるように、コネクション数 N が変化しても、最適な δ の値はそれほど変化しないためである。

しかし、伝搬遅延時間 τ が途中で変化した場合に

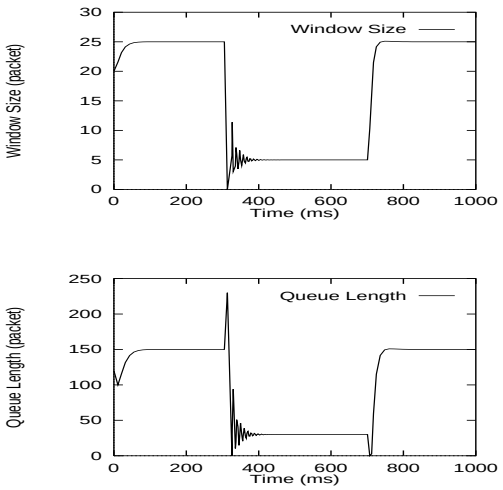

図 14 伝搬遅延時間が変化した場合 ($B = 20$ パケット/ms, $N = 10$, $\gamma = 3$ パケット, $\delta = 1.9$)
Fig. 14 Effect of Propagation Delay Variation ($B = 20$ packet/ms, $N = 10$, $\gamma = 3$ packet, $\delta = 1.9$)

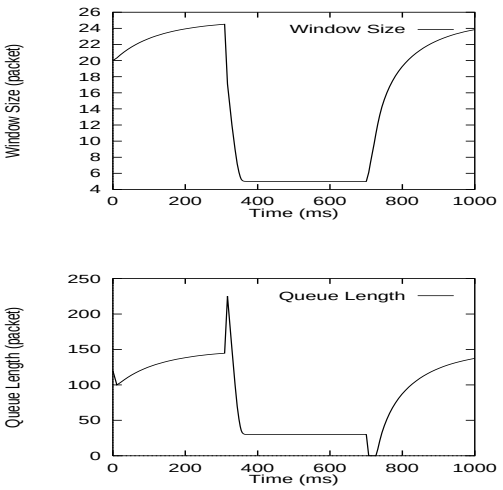

図 15 伝搬遅延時間が変化した場合 ($B = 20$ パケット/ms, $N = 10$, $\gamma = 3$ パケット, $\delta = 0.4$)
Fig. 15 Effect of Propagation Delay Variation ($B = 20$ packet/ms, $N = 10$, $\gamma = 3$ packet, $\delta = 0.4$)

は、δ の最適化制御によってシステムの性能が大きく改善される。図 14 に、伝搬遅延時間 τ が変化した場合のシステムの動作を示す。ここでは、図 11 と同じパラメータを用いているが、$t = 300$ ms から $t = 700$ ms の間の伝搬遅延時間を $\tau = 5$ ms から $\tau = 1$ ms と変化させている。これより、$t = 300$ ms において伝搬遅延時間が 1 ms へと変化した直後に、システムが再び安定するまでに時間がかかっている (およそ 100 ms 程度) ことがわかる。これは、$\delta = 1.9$ という値は伝搬遅延時間 $\tau = 5$ ms の時の最適値であり、伝搬遅延時間 $\tau = 1$ ms の場合には不適切な値となり、その結果過渡特性が劣化したためである。一方、$t = 700$ ms において伝搬遅延時間が 1 ms から 5 ms に変化した場合には、$\delta = 1.9$ という値は $\tau = 5$ ms の時の最適値であるため、システムの動作がすぐに安定している。

次に、伝搬遅延時間 $\tau = 1$ ms にあわせて δ を設定した場合の結果を示す。図 15 では、図 14 における δ の値を、$\tau = 1$ ms の場合の最適値となるように $\delta = 0.4$ と設定している。この図より、伝搬遅延時間 $\tau = 5$ ms にあわせて δ の値を設定した結果、伝搬遅延時間 $\tau = 1$ ms の場合の過渡特性がひどく悪化していることがわかる。これは、図 14 で $\tau = 1$ ms となっている状態 ($300 \leq t \leq 700$) と比較してもその影響が明らかである。これは、対象としているウィンドウ型のフロー制御方式では、ラウンドトリップ時間を単位としてウィンドウサイズの変更を行っているためである。つまり、伝搬遅延時間が大きい場合には、それだけウィンドウサイズの変更頻度が少なくなるため、δ の値が不適切であることの影響が大きくなるからである。

そこで、伝搬遅延時間の変化にあわせて δ の値を最適となるような制御を行った場合の結果を、図 16 に示す。これより、伝搬遅延時間 τ が変化した場合でも、δ の最適化制御を行うことによって、伝搬遅延時間の値にかかわらずほぼ理想的な過渡特性が得られていることがわかる。つまり、$t = 300$ ms において伝搬遅延時間が $\tau = 1$ ms と変化した場合でも、およそ 50 ms でシステムの動作が安定している。また、$t = 700$ ms において伝搬遅延時間が $\tau = 5$ ms へと変化した場合でも、すぐにシステムの動作が安定している。

6. 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、その安定性および過渡特性を制御理論を用いて明かにした。まず、定常状態においてウィンドウサイズおよびルータのバッファ内パケット数が収束するための条件 (システムの安定条件) を導出した。その結果、システムが安定となる、ウィンドウサイズの変化量を決定する制御パラメータのパラメータ領域は、コネクション数や伝搬遅延時間によって変化することを示した。また、システムの過渡特性に着目し、過渡特性が最適となるためにウィンドウサイズの変化量を決定する制御パラ

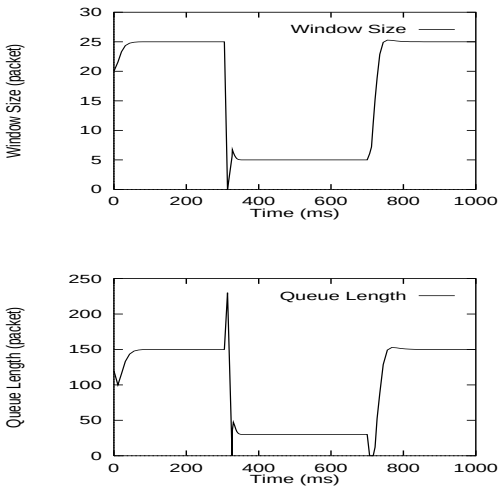

図 16 伝搬遅延時間が変化した場合 ($B = 20$ パケット/ms, $N = 10$, $\gamma = 3$ パケット)
Fig. 16 Effect of Propagation Delay Variation ($B = 20$ packet/ms, $N = 10$, $\gamma = 3$ packet)

メータがみたすべき条件を導出した。これらの結果、最適な制御パラメータの値は、さまざまなシステムパラメータに依存して決定する必要があることを定量的に示した。

現実の TCP/IP ネットワークでは TCP のトラヒックだけでなく、UDP のようにフィードバック系に分類されないトラヒックも混在する。そこで今後は、これらのトラヒックの影響をも考慮した評価も行ってゆく必要があると考えられる。

参 考 文 献

1) Brakmo, L. S., O'Malley, S. W. and Peterson, L. L.: TCP Vegas: New techniques for congestion detection and avoidance, *Proceedings of ACM SIGCOMM '94*, pp. 34–35 (1994).
2) Brakmo, L.S. and Peterson, L.L.: TCP Vegas: End to end congestion avoidance on a global Internet, *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 8, pp. 1465–1480 (1995).
3) Brent, R. P.: *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, N.J. (1973).
4) Fuhrmann, S., Kogan, Y. and Milito, R.A.: An adaptive autonomous network congestion controller, *Proceedings of IEEE CDC '96*, p.WA12 11:40 (1996).
5) Hasegawa, G., Murata, M. and Miyahara, H.: Fairness and stability of congestion control mechanism of TCP, to be presented at *ITC '98* (1998).
6) Jacobson, V.: Congestion avdoidance and control, *Proceedings of SIGCOMM '88*, pp. 314–329 (1988).
7) Keshav, S.: A control-theoretic approach to flow control, *Proceedings of ACM SIGCOMM '91*, pp. 3–15 (1991).
8) Rohrs, C. E., Berry, R. A. and O'Halek, S. J.: A control engineer's look at ATM congestion avoidance, *Proceedings of IEEE GLOBECOM '95*, pp. 1089–1094 (1995).
9) Stevens, W. R.: *TCP/IP Illustrated, Volume 1: The Protocols*, New York: Addison-Wesley (1994).
10) Zhang, H. and Yang, O. W.: Design of robust congestion controllers for ATM networks, *Proceedings of IEEE INFOCOM '97* (1997).
11) 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).