インターネットにおけるパケット伝送遅延時間のモデルを用いたレート制御方式の設計に関する一検討

森田 光茂† 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: 06-6850-6616, Fax: 06-6850-6589
E-mail: m-morita@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
Phone: 06-6879-8793, Fax: 06-6879-8794
E-mail: {oosaki,murata}@cmc.osaka-u.ac.jp

あらまし インターネットでは、エンド–エンド間で動作する輻輳制御方式として TCP (Transmission Control Protocol) が用いられている。現在広く普及している TCP Reno では、パケット棄却の有無をもとに輻輳制御を行っている。一方、ラウンドトリップ時間の変動をもとに輻輳制御を行う、TCP Vegas と呼ばれる方式が提案されている。ラウンドトリップ時間の変動をもとに輻輳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ことが可能となり、スループットの向上が期待できる。ただし、TCP Vegas はアドホックな手法で設計された輻輳制御機構であり、理論的な裏付けが十分なされていない。そこで本稿では、インターネットにおけるパケット伝送遅延時間のモデルに対して古典制御理論を適用することにより、ラウンドトリップ時間にもとづくレート制御方式を設計する。ラウンドトリップ時間が一定となるように、送信側ホストからのパケット送出レートを動的に制御することにより、ネットワーク内でのパケット棄却を防ぐこととともに、エンド–エンド間での効率的な通信を実現する。いくつかのシミュレーション実験により、設計したレート制御方式の有効性を定量的に評価する。
和文キーワード レート制御方式、パケット伝送遅延、古典制御理論

Study on Designing a Rate-based Congestion Control Mechanism using Packet Delay Dynamics Model in TCP/IP networks

Mitsushige Morita† Hiroyuki Ohsaki‡ Masayuki Murata‡

†Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka, Japan
Phone: +81-6-6850-6616, Fax: +81-6-6850-6589
E-mail: m-morita@ics.es.osaka-u.ac.jp

‡Cybermedia Center, Osaka University
1-30 Machikaneyama, Toyonaka, Osaka, Japan
Phone: +81-6-6879-8793, Fax: +81-6-6879-8794
E-mail: {oosaki, murata}@cmc.osaka-u.ac.jp

Abstract In the Internet, TCP (Transmission Control Protocol) has been used as an end-to-end congestion control mechanism. Of all several TCP implementations, TCP Reno is the most popular implementation. TCP Reno is a loss-based approach since it estimates the severity of congestion by detecting packet losses in the network. On the contrary, another implementation called TCP Vegas is a delay-based approach. The main advantage of a delay-based approach is, if it is properly designed, packet losses can be prevented by anticipating impending congestion from increasing packet delays. However, TCP Vegas was designed using not a theoretical approach but an ad hock one. In this paper, we therefore design a delay-based adaptive congestion control mechanism by utilizing the control theory. The rate control block dynamically adjusts the packet transmission rate to stabilize the round-trip time for utilizing the network resources and also for preventing packet losses in the network. Presenting thorough simulation results in various network configurations, we quantitatively show the robustness and the effectiveness of our delay-based adaptive congestion control mechanism.
key words Delay-based Congestion Control, Packet Delay, Classical Control Theory,

1 はじめに

インターネットでは、エンド–エンド間で動作する輻輳制御方式として、TCP (Transmission Control Protocol) が用いられている。現在広く普及している TCP Reno では、パケット棄却の有無をもとに輻輳制御を行っている。一方、ラウンドトリップ時間の変動をもとに輻輳制御を行う方式として、例えば、TCP Vegas [1] が提案されている。このような遅延にもとづく輻輳制御では、ラウンドトリップ時間の変動をもとに輻輳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ことが可能となり、スループットの向上が期待できる。

これまでに、TCP のようなウィンド型輻輳制御方式ではなく、レート型輻輳制御方式が提案されている [2, 3, 4]。文献 [2] では、TCP との公平性を考慮したレート型輻輳制御方式を提案している。また、文献 [3, 4] では、バッファ内パケット数を制御するフロー制御方式を提案している。

我々はこれまで、文献 [5, 6] において、システム同定を用いて、パケット伝送遅延時間の特性をモデル化する方法を提案した。そこでは、送信側ホストから見たネットワークを 1 入力 1 出力の動的システムと定義し、パケット伝送遅延時間の特性を ARX (Auto-Regressive eXogenous) モデルによってモデル化した。例えば、文献 [6] では、ネットワークへの入力としてパケット送信レート、ネットワークからの出力としてラウンドトリップ時間を用いている。また、LAN および WAN 環境で測定したデータをもとにモデル化の有効性の検証を行った。その結果、ボトルネックリンクが小数のユーザで共有されている場合、うまくモデル化できることを明らかにした。

本稿では、システム同定を用いて得られた、ラウンドトリップ時間のモデル化手法を利用して、レート制御方式を設計する。具体的には、ラウンドトリップ時間のモデルに対して古典制御理論を適用することで、遅延にもとづくレート制御機構を設計する。本稿では、古典制御理論の中でも、状態フィードバックによる極配置および状態観測器という手法を利用する。送信側ホストからみたネットワーク全体を制御対象ととらえ、ラウンドトリップ時間を一定にすることを制御目標とする。さらに 2 種類のシミュレーション実験を行い、提案するレート制御方式が効率的に動作していることを示す。

本稿の構成は以下のとおりである。まず 2 章で、システム同定、状態フィードバックによる極配置、状態観測器について簡単に説明する。3 章では、本稿で設計するレート制御方式を説明する。4 章で、シミュレーション実験を行い、設計するレート制御方式の有効性を示す。最後に 5 章でまとめと今後の課題について述べる。

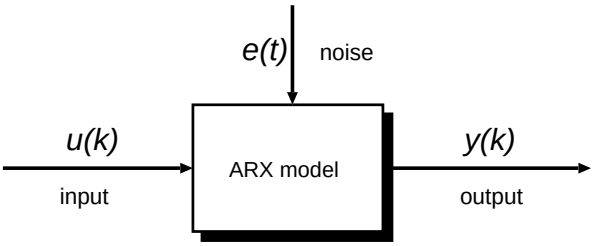


図 1: ARX モデル

2 システム同定および状態フィードバック制御

以下では、レート制御機構を設計するために必要となる、システム同定、状態フィードバックによる極配置、状態観測器について簡単に説明する。

2.1 システム同定

ARX モデルとは、システム同定で広く用いられている、線形時不変のパラメトリックモデルであり、1 入力 1 出力のシステムを表現することができる。ARX モデルのブロック図を図 1 に示す。システム同定とは、観測した入出力データから、ARX モデルのパラメータを推定する手法である。

ここで、一定のサンプリング周期ごとに測定した、k 番目の入力データおよび出力データを、$u(k)$、$y(k)$ とすると ARX モデルは以下のように定義される。

$$\begin{aligned} A(q)\,y(k) &= B(q)\,u(k-n_d) + e(k) \qquad (1)\\ A(q) &= 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \\ B(q) &= b_1 + b_2 q^{-1} + \ldots + b_{n_b} q^{-n_b+1} \end{aligned}$$

ここで、q^{-1} は $q^{-1}u(k) \equiv u(k-1)$ のように定義されるシフトオペレータである。n_a と n_b は ARX モデルの次数であり、n_d は入力と出力の間の遅れを意味する。また、$e(k)$ は白色雑音であり、出力データに含まれる測定不可能な雑音を意味する。式 (1) における、a_1 から a_n および b_1 から b_n はすべて ARX モデルのパラメータである。

システム同定は、観測した入出力データから、対象とするシステムの数学的なモデルを推定する手法である [7]。以下では、システム同定により、ARX モデルのパラメータを推定する方法を説明する。表記の簡単化のために、ARX モデルのパラメータを以下のように表記する。

$$\theta = [a_1 \ldots a_{n_a} b_1 \ldots b_{n_b}]^T \qquad (2)$$

ここで、ARX モデルのパラメータ θ を用いて、$k-1$ 番目のスロットまでに測定された入出力データから予測した出力を 1 段先予測と呼び、以下のように定義する。

$$\hat{y}(k|\theta) \equiv \psi(k)\,\theta \qquad (3)$$

ただし、

$$\psi(k) = [-y(k-1),\ldots,-y(k-n_a), u(k-n_d-1),\ldots,u(k-n_d-n_b)]^T$$

また、実際の出力 $y(k)$ と ARX モデルの 1 段先予測 $\hat{y}(k|\theta)$ の差を予測誤差 $\varepsilon(k|\theta)$ と呼び、以下のように定義する。

$$\varepsilon(k|\theta) \equiv y(k) - \hat{y}(k|\theta) \tag{4}$$

システム同定では、この予測誤差が小さくなるように ARX モデルのパラメータ θ を決定する。特に、以下の損失関数が最小となるように θ を決定する、最小 2 乗法が広く用いられている。

$$J_N(\theta) \equiv \frac{\sum_{k=1}^{N}\varepsilon(k|\theta)^2}{N} \tag{5}$$

ここで N は、システム同定に用いる入出力データの数 (サンプル数) である。

システム同定には 2 種類の同定方法 (一括同定法および逐次同定法) がある [7]。一括同定法では、あらかじめ測定された入出力データを用いて、オフラインでシステム同定を行う。一方、逐次同定法では、入出力データを測定しながら、オンラインでシステム同定を行う。最小 2 乗法の場合、一括同定法では逆行列演算が必要となるが、システム同定に必要なサンプル数が少なくてすむ。逆に、逐次同定法では逆行列演算が不要であるが、システム同定に必要なサンプル数が多くなる。

以下では、本稿で設計するレート制御機構で用いる、逐次同定法のアルゴリズムを説明する。$\hat{\theta}(k)$ をスロット k におけるパラメータ推定値とする。また、$\mathbf{P}(k)$ をスロット k における共分散行列とする。

初期値:

$$\hat{\theta}(0) = 0$$
$$\mathbf{P}(0) = \gamma\mathbf{I} \ (\gamma > 0、\mathbf{I} は単位行列)$$

入出力データ測定ごと:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \mathbf{P}(k)\psi(k)[y(k) - \psi^T(k)\hat{\theta}(k-1)]$$
$$\mathbf{P}(k) = \mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\psi(k)\psi^T(k)\mathbf{P}(k-1)}{1+\psi^T(k)\mathbf{P}(k-1)\psi(k)}$$

ここで、γ は通常 10^3 から 10^4 程度に設定される。

2.2 状態フィードバックによる極配置

まず、制御対象が 1 入力 1 出力の線形システムの場合を考える。状態フィードバックによる極配置とは、制御対象の状態変数をフィードバックすることにより、システムの安定性・過渡特性を自由に決定する手法である (図 2)。この図において、制御対象の状態変数 $\mathbf{x(k)}$ をフィードバックすることで、入力 $u(k)$ を決定する。このような閉ループシステムにおいて、システムの安定性・過渡特性を決定するという問題は、システムの特性を表す極を配置するという問題に帰着できる。つまり、図 2 のフィードバックゲイン $\mathbf{F}$ を適切に決定すれば、極を任意の位置に配置することができるため、結果としてシステムの安定性・過渡特性を任意に決定することができる。

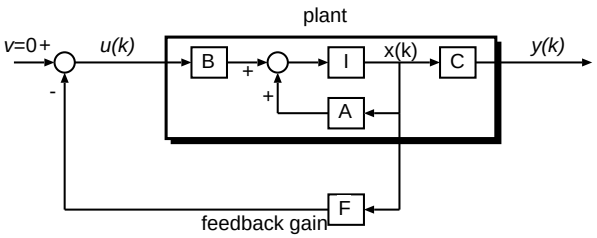


図 2: 状態フィードバック

表 1: 記号の定義

$\mathbf{x}(k)$	スロット k での状態変数 (n 次元ベクトル)
$u(k)$	スロット k での入力 (スカラー)
$y(k)$	スロット k での出力 (スカラー)
$\mathbf{A}$	定数行列 $(n \times n)$
$\mathbf{B}$	定数行列 $(n \times 1)$
$\mathbf{C}$	定数行列 $(1 \times n)$

状態空間表現を用いると、図 2 における 1 入力 1 出力の制御対象は、以下のように記述できる。

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k) \\ y(k) = \mathbf{C}\mathbf{x}(k) \end{cases} \tag{6}$$

ここで、$u(k)$ および $y(k)$ は、それぞれ k 番目のスロットにおけるシステムへの入力およびシステムからの出力である。式 (6) における記号の定義を表 1 に示す。

いま、制御対象の状態変数 $\mathbf{x}(k)$ が正確に測定できると仮定し、システムへの入力として以下のような状態フィードバックを考える。

$$u(k) = -\mathbf{F}\mathbf{x}(k) \tag{7}$$

このとき、式 (7) を 式 (6) に代入することにより、次式が得られる。

$$\begin{cases} \mathbf{x}(k+1) = (\mathbf{A} - \mathbf{B}\mathbf{F})\mathbf{x}(k) \\ y(k) = \mathbf{C}\mathbf{x}(k) \end{cases} \tag{8}$$

式 (8) で与えられるシステムにおいて、フィードバックゲイン $\mathbf{F}$ を適切に決定すれば、その極を自由に配置できる [8]。これはつまり、式 (8) のシステムの安定性・過渡特性を自由に決定できることを表わしている

ただし、状態フィードバックによる極配置を行うためには、制御対象が可制御であることが必要である [8]。あ

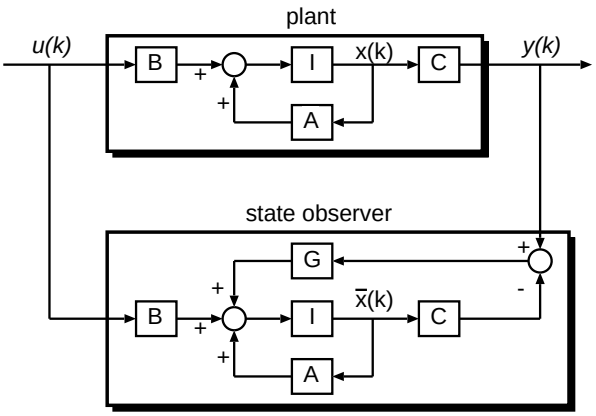


図 3: 状態観測器

るシステムが可制御であるとは、入力 $u(k)$ によって状態変数 $\mathbf{x}(k)$ が制御できるということである。具体的には、あるシステムが可制御であるためには、可制御行列

$$V \quad = \quad [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}]$$

に対して、 rank $V = n$ が成り立つ必要がある。

2.3 状態観測器

状態観測器は、観測した入出力データから、制御対象の状態変数を推測する手法である (図 3)。 前節の状態フィードバックによる極配置において、制御対象の状態変数はすべて観測できると仮定していた。しかし、一般には状態変数 $\mathbf{x}(k)$ は測定できない場合が多い。

状態観測器は、観測した入力データ $u(k)$ および出力データ $y(k)$ から、制御対象の状態変数 $\mathbf{x}(k)$ を推測する。 ここで、制御対象の状態変数の推測値を $\bar{\mathbf{x}}(k)$ とすると、図 3 を参考にして、以下の式が得られる。

$$\bar{\mathbf{x}}(k+1) \quad = \quad (\mathbf{A} - \mathbf{GC})\bar{\mathbf{x}}(k) + \mathbf{G}y(k) + \mathbf{B}u(k) \quad (9)$$

このとき、$q(k) \equiv \mathbf{x}(k) - \bar{\mathbf{x}}(k)$ とすれば、式 (6) および式 (9) より、次式が得られる。

$$q(k+1) \quad = \quad (\mathbf{A} - \mathbf{GC})q(k) \qquad (10)$$

これはつまり、$\mathbf{G}$ を適切に選択すれば、式 (10) で与えられるシステムの極を自由に配置できることを意味している [8]。その結果、$k \to \infty$ の時に $q(k) \to 0$ とすることができる。

ただし、状態観測器を実現するためには、制御対象が可観測であることが必要である [8]。 あるシステムが可観測であるとは、出力 $y(k)$ を観測することによって状態変数 $\mathbf{x}(k)$ を一意に決定できるということである。具体的には、あるシステムが可観測であるために、可観測行列

$$N \quad = \quad [\mathbf{C}^T, \mathbf{A}^T\mathbf{C}^T, (\mathbf{A}^T)^2\mathbf{C}^T, \ldots, (\mathbf{A}^T)^{n-1}\mathbf{C}^T]^T$$

に対し、rank $N = n$ が成り立つ必要がある。

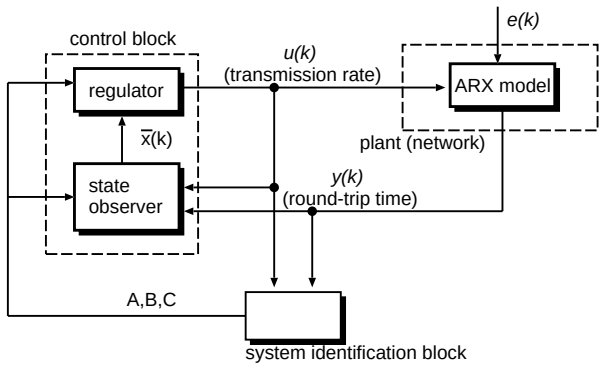


図 4: 設計するレート制御方式のブロック図

3 レート制御方式の設計

3.1 設計するレート制御方式の概要

本稿で設計するレート制御方式のブロック図を図 4 に示す。ここでは、送信側ホストからみた、ネットワーク全体を制御対象と考え、送信側ホストからの出力 (ネットワークへの入力) をパケット送信レートとしている。また、送信側ホストへの入力 (ネットワークからの出力) をラウンドトリップ時間としている。なお、送信側ホストでラウンドトリップ時間を測定するために、受信側ホストは、送信側ホストからのパケットに対して応答パケットを返送するものとする。本稿では、遅延にもとづく輻輳制御を実現するため、制御目標をラウンドトリップ時間を一定にすることと設定する。

図 4 のように、送信側ホストは、システム同定ブロックおよび制御ブロックから構成される。システム同定ブロックでは、ネットワークへの入出力データを測定することで、ラウンドトリップ時間の変動をあらわすモデルを得る。制御ブロックでは、システム同定ブロックで得られたモデルに古典制御理論を適用することで、送信側ホストからの送信レートを決定する。

レート制御機構を設計するためには、いくつかある性能指標の中から、制御目標を選択する必要がある。レート制御方式の性能指標としては、以下のようなものが挙げられる。

- エンド–エンド間でのスループット
- エンド–エンド間でのパケット棄却率
- エンド–エンド間でのパケット伝送遅延およびその分散

本稿で設計するレート制御方式は、ラウンドトリップ時間の変動にもとづき輻輳制御を行う。つまり、制御目標を、送信側ホストで観測するラウンドトリップ時間を一定にすることと設定する。制御目標となるラウンドトリップ時間の値を適切に選択すれば、エンド–エンド間でのパケット伝送遅延およびその分散を小さくすることが可能である。また、ボトルネックとなるルータのキュー長 (バッファ内パケット数) を低く抑えることが可能となり、

ボトルネックとなるルータを有効に利用できる。つまり、エンド–エンド間でのスループットを向上させ、パケット棄却率を低くおさえることが期待できる。

3.2 システム同定ブロック

システム同定ブロックでは、観測した入出力データからシステム同定を用いて、ラウンドトリップ時間の変動をあらわすモデルを作成する。本稿では、送信側ホストからみたネットワーク全体を、1 入力 1 出力の ARX モデルによってモデル化する。ネットワークへの入力 $u(k)$ はパケット送信レートであり、ネットワークからの出力 $y(k)$ はラウンドトリップ時間である。これらの変動を送信側ホストにおいて測定し、2 章で示した逐次同定法を用いて、ARX モデルのパラメータを決定する。

ここでは、ネットワークへの入力および出力を具体的に定義する。システム同定では、入力および出力の平均は 0 であることが望ましい。そのため本稿では、入力をパケット送信レートと平均送信レート $\overline{u}$ との差とする。また、出力をラウンドトリップ時間と制御目標 y^* との差とする。サンプリング周期を T とすれば、$u(k)$ および $y(k)$ は以下の式で定義される。

$$\begin{aligned} u(k) &= \frac{\sum_{i\in\phi_s(k)} l(i)}{T} - \overline{u} \\ y(k) &= \frac{\sum_{i\in\phi_r(k)} (t_r(i) - t_s(i))}{|\phi_r(k)|} - y^* \end{aligned}$$

$t_s(i)$ は、送信側ホストで測定した i 番目のパケットの送信時刻であり、$t_r(i)$ は、送信側ホストで受信した i 番目のパケットに対応する 応答 パケットの受信時刻である。また $l(i)$ は、i 番目のパケットのパケット長である。$\phi_s(k)$ および $\phi_r(k)$ は、k 番目のスロットで、送信もしくは受信したパケットの集合であり、以下のように定義される。

$$\begin{aligned} \phi_s(k) &\equiv \{n : kT \le t_s(n) < (k+1)T\} \\ \phi_r(k) &\equiv \{n : kT \le t_r(n) < (k+1)T\} \end{aligned}$$

$\overline{u}$ は、送信レートの平均であり、次式のようなローパスフィルタによって推定する。

$$\overline{u} \leftarrow \alpha\,\overline{u} + (1-\alpha)\,u(k) \tag{11}$$

ここで α $(0 < \alpha < 1)$ はローパスフィルタの重みである。y^* は、ラウンドトリップ時間の制御目標であり、次節で説明する。

本稿で設計するレート制御方式は、送信レートを制御するため、あるサンプリング期間中にパケットが存在しない場合がある。そのような場合は、それまでの入力/出力データの最小値を補完する。

3.3 制御ブロック

制御ブロックでは、状態観測器を用いて、送信側ホストにおいて観測した入出力データから制御対象の状態変数を推測する。次に推測した状態変数をもとに、状態フィードバックによる極配置を行い、送信レートを制御することで、ラウンドトリップ時間を制御目標に近づける。

状態観測器は、システム同定ブロックで得られた、ARX モデルの状態変数を推測するシステムである。まず、システム同定により推測した ARX モデルを、状態空間表現に変換することにより、$\mathbf{A}$、$\mathbf{B}$、$\mathbf{C}$ を求める。次に、式 (9) を用いて、制御対象の状態変数 $\mathbf{x}(k)$ を推定する。そして、式 (10) で与えられるシステムの極が λ_i $(1 \le i \le max(n_a, n_b) + n_d - 1)$ となるように、$\mathbf{G}$ を決定する。ただし、システム同定ブロックでは逐次同定を行っているため、ARX モデルのパラメータ、つまり $\mathbf{A}$、$\mathbf{B}$、$\mathbf{C}$ は、時間とともに変動する。そのため、N_o サンプルごとに $\mathbf{G}$ を再計算する。

状態観測器によって推測した、ARX モデルの状態変数 $\bar{x}(k)$ を用いて、状態フィードバックによる極配置を行う。極配置により、ラウンドトリップ時間が制御目標に収束するような、レート制御方式を実現する。具体的には、式 (8) を用いて状態フィードバックによる極配置を行う。つまり、式 (8) で与えられるシステムの極が μ_i $(1 \le i \le max(n_a, n_b) + n_d - 1)$ となるように、$\mathbf{F}$ を決定する。ただし、システム同定ブロックでは逐次同定を行っているため、ARX モデルのパラメータは変動する。そのため、 N_r サンプルごとに $\mathbf{F}$ を再計算することとする。

送信側ホストでは、上記の状態フィードバックを用いたフィードバック制御により、送信レートを決定する。ここで、送信レートの変更は、サンプリング周期ごとに行う。また、送信側ホストは、UDP を用いてパケットを送出する。なお、簡単のため、パケット棄却には上位層のプロトコルが対処するとし、設計するレート制御方式では再送制御は行わない。

ラウンドトリップ時間の制御目標 y^* を適切に決定する必要があるが、本稿では、TCP Vegas [1] で用いられている手法を利用し、以下のように y^* を決定する。

$$y^* = \tau + \frac{\nu}{\overline{u}} \tag{12}$$

ここで、τ は送信側ホストで観測した往復伝搬遅延時間、ν は制御パラメータである。

4 シミュレーション

4.1 ネットワークモデル

本章では、設計したレート制御方式の有効性を評価する。評価方法として、実際のラウンドトリップ時間が、設定したラウンドトリップ時間の制御目標にどのように収束するかを調べる。また、設計したレート制御方式により、どの程度のスループットが得られるかについても調べる。本稿では、2 種類のネットワークモデルを対象とし、ns2 [9] を用いたシミュレーションによりレート

表 2: シミュレーションで使用したパラメータ

T	サンプリング周期	30 [ms]
n_a, n_b, n_d	ARX モデルの次数、遅れ	3, 3, 3
N_r	$\mathbf{F}$ の更新間隔	10 [スロット]
N_o	$\mathbf{G}$ の更新間隔	10 [スロット]
N_c	モデル作成期間	500 [スロット]
λ_i	状態フィードバックの極	0.5
μ_i	状態観測器の極	0.5
α	ローパスフィルタの重み	0.95
ν	制御パラメータ	100 [パケット]
γ	逐次同定法に用いる正定数	10000

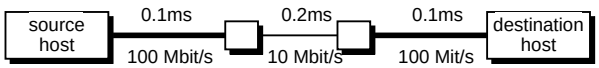


図 5: ネットワークモデル **N1**

制御方式の性能を評価を行う。シミュレーション実験におけるパラメータを表 2 に示す。本稿で設計するレート制御方式では、転送開始直後 N_c スロット間は、制御ブロックによるレート制御は行わず、同定ブロックによるモデル作成のみを行う。この間、送信側ホストからのパケット送信間隔が、平均 10 [Mbit/s] の指数分布に従うようにした。

4.2 ネットワークモデル N1

ネットワークモデル **N1** は、他のトラヒックが存在しない単純なモデルである (図 5)。ネットワーク **N1** では、送信側ホストと受信側ホストの間には、2 個のルータが存在し、送信側ホスト、受信側ホストからルータへは 100 [Mbit/s] のリンクで接続されている。ルータ間は 10 [Mbit/s] のリンクで接続されており、ここがボトルネックとなる。ボトルネックリンクの伝搬遅延時間は 0.4 [ms]、ルータのバッファサイズは 200 [パケット] とした。

ネットワーク **N1** におけるシミュレーション結果を、図 6 に示す。図 6(a) は、各スロットにおいてシステム同定ブロックが計算した、ARX モデルのパラメータである。この図から、約 10 スロット程度で、ARX モデルのパラメータがほぼ収束していることがわかる。図 6 (b) は、送信側ホストのパケット送信レートの変動を示している。レート制御が開始した後 (500 スロット以降) は、レート制御ブロックがパケット送信レートを決定している。この図より、レート制御が開始した直後に、パケット送信レートが 10 [Mbit/s] で安定していることがわかる。ボトルネックリンクの帯域は 10 [Mbit/s] であるため、ネットワーク資源を有効に利用できていることがわかる。図 6(c) は、ボトルネックルータにおけるキュー長の変動 (各スロットごとの平均値) である。この図から、500 スロットにおいてレート制御が開始すると、約 50 スロット程度 (約 l.5 sec) でキュー長が安定していることがわかる。これは、レート制御方式の制御目標 $\nu = 100$ [パケット] に一致している。このことから、提案するレート制御方式が適切に動作していることがわかる。図 6 (d) は、送信側ホストで測定したラウンドトリップ時間 (各スロットごとの平均値) である。この図から、提案するレート制御方式は、意図した通りにラウンドトリップ時間を一定に制御できていることがわかる。以上の結果から、他のトラヒックが存在しない単純なネットワークにおいて、提案するレート制御方式が非常に効率的に動作することが分かる。

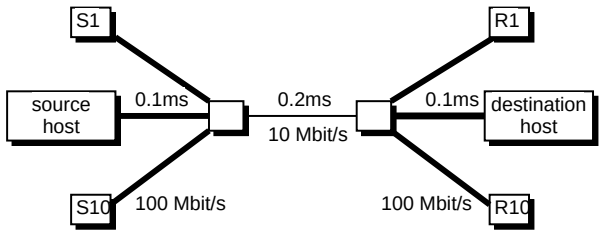
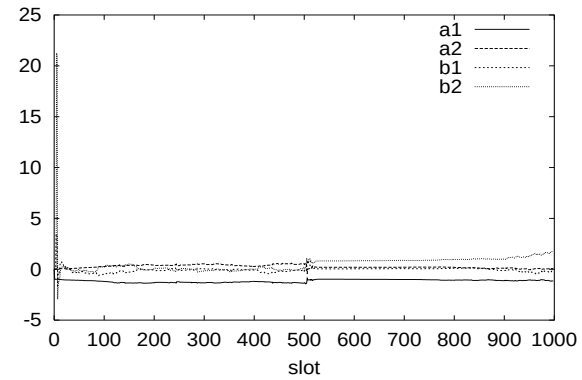


図 7: ネットワークモデル **N2**

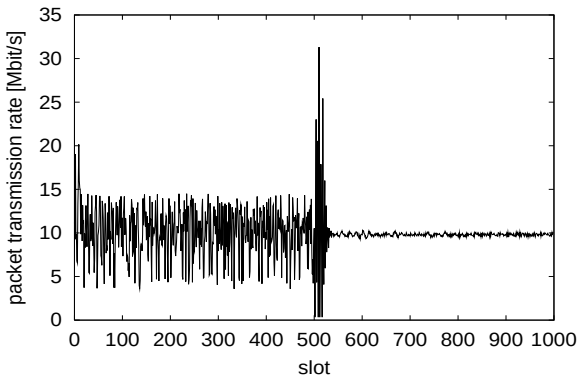
4.3 ネットワークモデル N2

ネットワークモデル **N2** は、ボトルネックリンクを他のトラヒックと共用しているネットワークモデルである (図 7)。送信側ホストと受信側ホストとの間のルータの数、リンク容量、伝搬遅延時間およびルータのバッファサイズは、ネットワークモデル **N1** と同じである。ネットワークモデル **N1** に、$S1$ から $R1$ および $S2$ から $R2$ へのトラヒックを追加したものが、ネットワークモデル **N2** となる。$S1$ および $S2$ は、それぞれ UDP パケットを平均 3 [Mbit/s] で送信する。$S1$ から $R1$ へのパケット送信は、シミュレーション開始後 1000 スロット目に開始し、シミュレーションが終了するまでパケットを送信する。また、$S2$ から $R2$ へのパケット送信は、シミュレーション開始後 1500 スロット目に開始し、シミュレーションが終了するまでパケットを送信する。

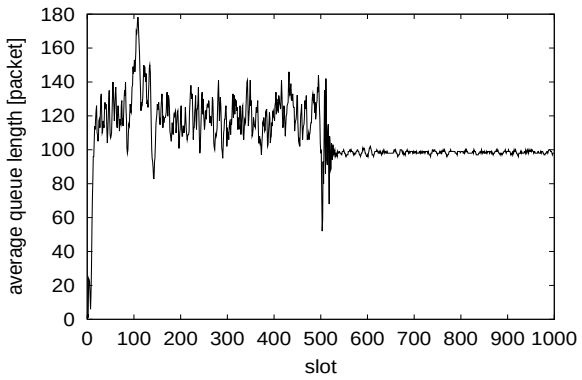
ネットワーク **N2** におけるシミュレーション結果を、図 8 に示す。図 8(a) から、バックグラウンドトラヒックの量が変化した直後 (1000 スロットおよび 1500 スロット) に、パラメータがゆっくりと収束していることがわかる。また、図 8(b) から、バックグラウンドトラヒックの量が変化した後、約 100 スロット程度 (約 3 sec) でパケット送信レートは安定していることがわかる。$S1$ および $S2$ の平均パケット送信レートは 3 [Mbit/s] であるため、ネットワーク資源を有効に利用できていることがわかる。図 8(c) から、バックグラウンドトラヒックの量が変化した直後では、ルータにおいてバッファ溢れ



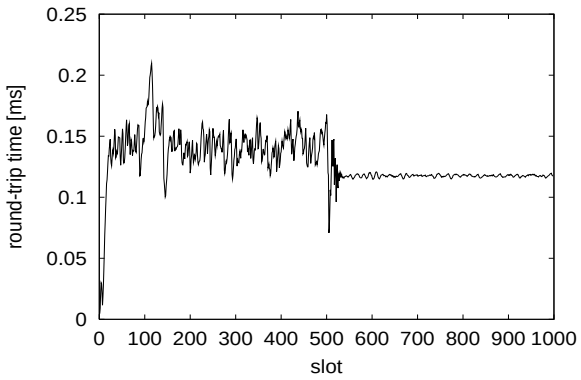
(a) ARX モデルのパラメータ



(b) 送信側ホストのパケット送信レート



(c) ボトルネックルータでのキュー長



(d) 送信側ホストで観測したラウンドトリップ時間

図 6: シミュレーション結果 (ネットワークモデル **N1**)

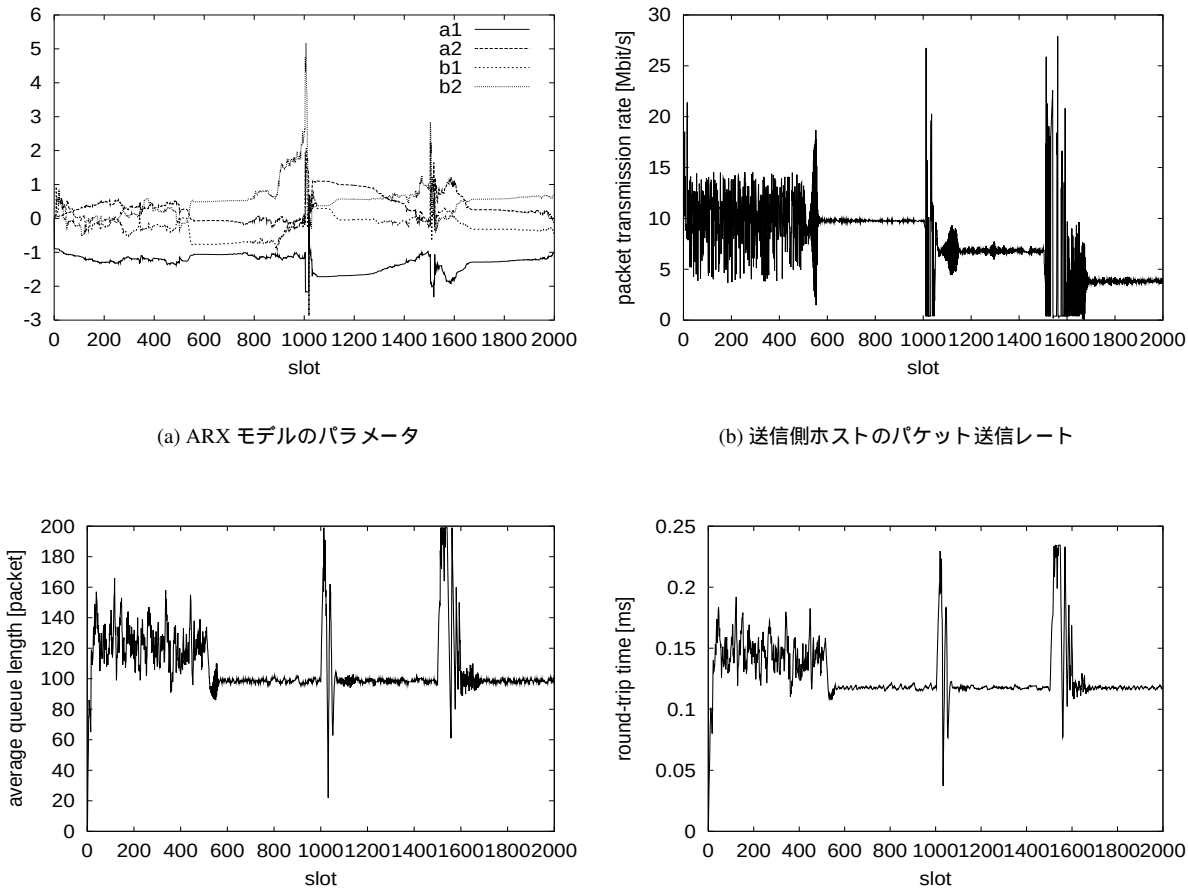
が発生しているが、約 100 スロット程度 (約 3.0 sec) でキュー長が 100 [パケット] で安定していることがわかる。このことから、提案するレート制御方式が適切に動作していることがわかる。また、図 8 (d) から、ラウンドトリップ時間を一定に制御できていることがわかる。以上の結果から、バックグラウンドトラヒックが存在する場合でも、提案するレート制御方式が効率的に動作していることが分かる。

5 まとめと今後の課題

本稿では、古典制御理論を用いたレート制御方式を提案した。ここでは、送信側ホストからみたネットワーク全体を制御対象ととらえ、ラウンドトリップ時間を一定とすることを制御目標とした。パケット送信側ホストは、システム同定ブロック、制御ブロックからなり、システム同定ブロックにおいて、測定した入出力データからラウンドトリップ時間のモデル化を行なった。また、制御ブロックでは、状態観測器を用いて ARX モデルの状態変数を推測し、状態フィードバックによる極配置を利用して、送信レートを制御するよう設計した。本稿では、設計したレート制御方式の性能を評価するため 2 種類のシミュレーション実験を行い、その結果、レート制御方式が効率的に動作していることを示した。

今後の課題として、より複雑で現実のネットワークに近い環境において、提案したレート制御機構の性能評価を行いたいと考えている。本稿では、バックグラウンドトラヒックとして UDP パケットを送信したが、実際のネットワークでは TCP パケットがトラヒックの大部分を占めている。TCP は、ベストエフォート型の輻輳制御方式であり、パケットロスおよびタイムアウトにより輻輳を検知する。そのため、TCP トラヒックが存在するネットワーク環境では、UDP トラヒックのみの環境に比べてキュー長は不安定になることが予想される。そのため、TCP トラヒックが存在する環境において、どの程度 ラウンドトリップ時間を一定にするレート制御が可能であるかを評価する必要がある。

また、他の輻輳制御機構と性能比較を行う必要がある。本稿のように、遅延にもとづく輻輳制御方式には、例えば TCP Vegas がある。同じネットワークモデルにおいて、スループット、ラウンドトリップ時間の変動を比較



(a) ARX モデルのパラメータ

(b) 送信側ホストのパケット送信レート

(c) ボトルネックルータでのキュー長

(d) 送信側ホストで観測したラウンドトリップ時間

図 8: シミュレーション結果 (**N2**)

することで、どの程度、本稿で提案したレート制御方式が有効であるかを示したいと考えている。

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」(JSPS-RFTF97R16301) によっている。ここに記して謝意を表す。

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, Oct. 1994.

[2] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *INFOCOM (3)*, pp. 1337–1345, 1999.

[3] S. Keshav, "Packet-pair flow control," *N/A*, 1994.

[4] S. P. Morgan and S. Keshav, "Packet-pair rate control — buffer requirements and overload performance," *N/A*, Oct. 1994.

[5] 森田 光茂, 大崎 博之, 村田 正幸, "インターネットにおけるパケット伝送遅延時間の測定およびシステム同定によるモデル化に関する検討," 電子情報通信学会技術研究報告 *(SSE2001-243)*, pp. 55–62, Mar. 2001.

[6] 森田 光茂, 大崎 博之, 村田 正幸, "システム同定を用いたインターネットのパケット伝送遅延時間のモデル化に関する検討," 電子情報通信学会技術研究報告 *(NS2001-160)*, pp. 41–46, Nov. 2001.

[7] L. Ljung, *System identification — theory for the user*. Englewood Cliffs, N.J.: Prentice Hall, 1987.

[8] 伊藤正美, 自動制御概論 *(*下*)*. 昭晃堂, 1985.

[9] "The network simulator – ns2." available at `http://www.isi.edu/nsnam/ns/`.