L-018

帯域公平性をもつ輻輳制御

A Congestion Control Scheme with Fair Bandwidth Sharing

井上 貴照 † 木村 昌弘 †† 中村 奉夫 ††
Yoshiaki Inoue Masahiro Kimura Tomoo Nakamura

1 はじめに

現在、インターネットは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用され、トラフィックがますます増大している。インターネットの標準的信頼性プロトコル TCP(Transmission Control Protocol) では、輻輳によるパケットロストに対する制御機能がある。TCP では、輻輳制御として、ロスベースに代表される Reno と遅延ベースに代表される Vegas が提案されている。ただ、Vegas のスループットが、Reno と Vegas が混在する環境において大幅に低下する問題点がある [3]。Reno との公平性を考慮した方式 [1][2] も提案されているが Vegas との不公平性がある。

本研究では、Reno、Vegas 両方の帯域公平性の向上を目的とし、Vegas の輻輳ウィンドウサイズ制御アルゴリズムに、RTT(Round Trip Time) の履歴を用いて改善を行い、シミュレーターによりその評価を行った。

2 TCP による輻輳制御

TCP は、スロースタート段階と輻輳回避段階が存在し、ACK を受信するごとにデータ転送量を調整する輻輳ウィンドウ ($Cwnd$) を更新する。ここでは、Reno、Vegas の動作を説明する。

2.1 Reno

Reno のスロースタート段階は、指数的にウィンドウサイズを増加させる。また輻輳回避段階は、線形的にウィンドウサイズを増加させることによって $Cwnd$ を調整している。
パケットロスが発生した場合は、直前のウィンドウサイズの 1/2 にする

2.2 Vegas

Vegas は、計測した最小 RTT($baseRTT$) と現在の RTT からスループットの差を用いることによって、制御を行っている。また、$\alpha, \beta(\alpha < \beta)$ は定数であり、中間ノードにキューイングされるパケット数を決定する。

$Cwnd =$

$$\begin{cases} [\text{スロースタート}] \\ Cwnd + 1/2 \\ [\text{輻輳回避段階}] \\ Cwnd + 1 & (RTT < \frac{Cwnd}{Cwnd-\alpha} baseRTT) \\ Cwnd & (\frac{Cwnd}{Cwnd-\alpha} baseRTT < RTT < \frac{Cwnd}{Cwnd-\alpha} baseRTT) \\ Cwnd - 1 & (RTT < \frac{Cwnd}{Cwnd-\beta} baseRTT) \end{cases}$$

3 提案手法

Vegas が、計測スループット値を期待 (理論) スループット値に収束させることで制御しているのに対し、提案手法では、一定時間間隔で計測した、時刻 t における RTT[t] と 1 間隔前の RTT[t-1] との差 ($DiffRTT$) および、$DiffRTT$ の平均値 ($aveDiffRTT$) を従来の Vegas に加えて制御に用いる。

$$DiffRTT = RTT[t] - RTT[t-1]$$

$$aveDiffRTT = | (1-w) * aveDiffRTT + w * DiffRTT |$$

$aveDiffRTT$ の w は、重み係数である。Vegas の輻輳回避段階を次のような改善を行う。RTT 履歴の差を用いる事より、RTT 変動の激しい場合に追加でウィンドウ制御を行う考えである。以下に制御を示す。

$Cwnd =$

$$\begin{cases} [RTT < \frac{Cwnd}{Cwnd-\alpha} baseRTT] \\ Cwnd + 1 \\ [\frac{Cwnd}{Cwnd-\alpha} baseRTT < RTT < \frac{Cwnd}{Cwnd-\beta} baseRTT] \\ Cwnd + 1 \quad (if \quad DiffRTT < \frac{Cwnd}{Cwnd-\alpha} aveDiffRTT) \\ Cwnd \quad (else) \\ [RTT < \frac{Cwnd}{Cwnd-\beta} baseRTT] \\ Cwnd \quad (if \quad DiffRTT < \frac{Cwnd}{Cwnd-\beta} aveDiffRTT) \\ Cwnd - 1 \quad (else) \end{cases}$$

4 評価実験

4.1 実験環境

ネットワークシミュレータ NS-2[4] より、図 1 のようなネットワークトポロジに対して実験を行う。Node1、Node2 は各輻輳制御である。Node5、Node6 は Node1、Node2 の受信側である。また、Node3、Node4 は中間 Node である。中間 Node におけるパケット廃棄制御は、Drop-Tail とする。フローに対する受信 Node は、TCP Sink とする。

4.2 実験条件

ここでは、RTT の平均値を用いて Vegas を改善した輻輳制御 [1][2] を Y.KVegas とする。また、Y.KVegas では、平均値 RTT の重み係数 w の値が記載されていない [1][2] ため再送タイムアウトに用いられる $w = 1/8$ とした。さらに、我々の提案手法を Y.IVegas とし、$w = 1/4$ とする。定数 α, β は、$\alpha = 1, \beta = 3$ と設定する。Node1、Node2 に Reno、Vegas、

† 龍谷大学大学院 理工学研究科 電子情報学専攻
†† 龍谷大学 理工学部 電子情報学科

Y.KVegas、Y.IVegas から選択することで比較実験を行う。

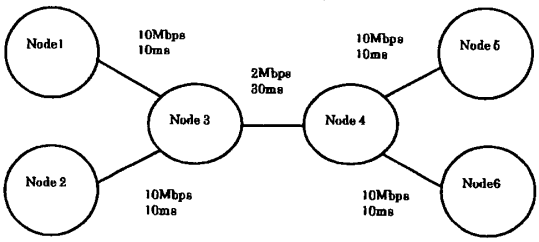

図 1　ネットワークトポロジ

5　実験結果

5.1　実験結果 1

Reno のコネクションが開始して 20 秒後に Vegas または、Y.IVegas のコネクションが開始した時の輻輳ウィンドウサイズの変化を図 2、図 3 に示す。

図 2、図 3 より、Y.IVegas が Vegas に比べ Reno に対して積極的にウィンドウサイズの増加、維持を行っているのがわかる。これは、相手が RTT 変動の激しい Reno であるため、$aveDiffRTT$ の値が大きくなることからウィンドウサイズの増加・維持を行っていると考えられる。

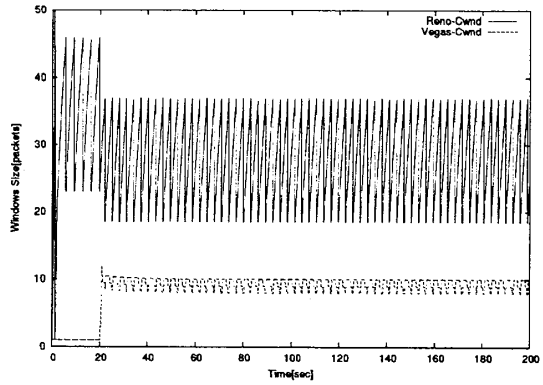

図 2　Reno、Vegas による輻輳ウィンドウサイズの変化

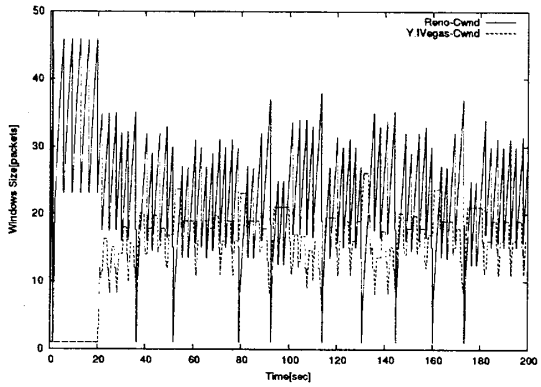

図 3　Reno、Y.IVegas による輻輳ウィンドウサイズの変化

5.2　実験結果 2

Node1 を Reno または Vegas による転送とし、Node2 に 4 個の輻輳制御による転送とする。同時に、2 個のコネクションを開始させ、一定時間での転送量を求めた。小さい方の転送量を図 4 に示す。

図 4 より、今回提案した Y.IVegas は、Reno に対して若干不公平性が残っているが、Vegas に対する公平性を考慮した結果となっていることがわかる。また、Y.KVegas は Reno との不公平性を解決しているが、Vegas に対して、従来通りの不公平性を残している。ただ、重み係数 w によって結果は変動する可能性もある。

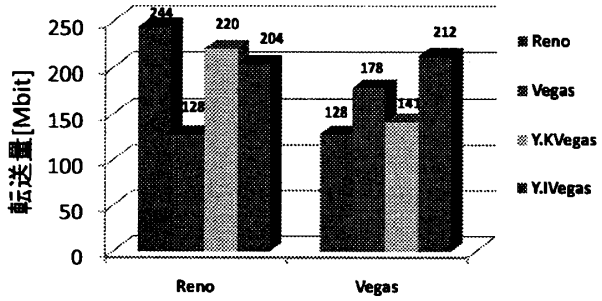

図 4　転送量の最小値

6　おわりに

Vegas を改善した Y.IVegas を提案し、ネットワークシミュレータを用いて評価実験を行った。提案方式は、未だに Reno に対して若干不公平性を残しているが、Vegas との公平性を性能が向上していることがわかった。今後の課題としては、Vegas との公平性を維持しつつ、Reno との公平性を向上させる点にある。

謝辞

本研究は、科学研究費補助金基盤研究 (C)(No.20500147) の補助を受けた。

参考文献

[1] 山口 一郎, 甲藤 二郎,"Reno との親和性を考慮した TCP Vegas の改善とその応用", 電子情報通信学会 情報ネットワーク研究会,IN2002-274,Mar.2003.

[2] 山口 一郎, 甲藤 二郎,"TCP Reno との公平性を考慮した TCP Vegas の改善", 第 13 回 ITRC 研究会,Mar.2003.

[3] J. Mo, R. J. La V.Anantharam and J. Walrand,"Analysis and Comparison of TCP Reno and Vegas,"Proc IEEE INFOCOM'99,MArch 1999.

[4] The Network Simulator - ns-2 http://www.isi.edu/nsnam/ns/