高速回線のための TCP 輻輳制御方式(TCP-Fusion)の実装評価

Experimental Study of TCP-Fusion for High-speed Networks

兼子 和巳　　甲藤 二郎
Kazumi KANEKO　　Jiro KATTO
早稲田大学大学院理工学研究科
Graduate School of Science and Engineering, Waseda University

１．はじめに

ネットワークの高速化により、現在標準的に用いられている TCP-Reno では利用可能帯域を十分に使い切ることが困難であるという問題が顕在化してきた。その 1 つの要因として、TCP-Reno の輻輳制御方式自体が考えられており、High-speed TCP[1](以下、HSTCP)など、数多くの提案がなされている。しかしながら、それらの方式では、TCP-Reno と競合した場合にどちらか一方のスループットが押し下げられる問題があり、普及するには至っていない。上記の問題に対して、近年、高速性だけでなく、TCP-Reno との親和性を両立することの出来る TCP-Adaptive Reno[2]、Compound TCP[3]などの方式が提案されており、我々も同様に、TCP-Fusion[6]を提案している。本稿では、TCP-Fusion 方式を Linux 上に実装して、その評価を行った結果を報告する。

2．TCP-Fusion 方式

(1) 輻輳ウインドウ減少方法

TCP-Fusion では、TCP-Westwood[4]に基づく減少方法を採用している。パケット廃棄が検出された場合には、以下のように輻輳ウインドウを減少させる。

$$cwnd_{new} = \max(\frac{RTT_{\min}}{RTT} cwnd_{last}, \frac{cwnd_{last}}{2})$$

ここで、$cwnd_{new}$、$cwnd_{last}$ は、それぞれ更新後及び更新前の輻輳ウインドウを示す。RTT_{min} は RTT の最小値である。輻輳ウインドウを半減させた閾値を設けているのは、TCP-Reno によって押し下げられるのを防ぐためである。

(2) 輻輳ウインドウ増加方法

TCP-Fusion では、TCP-Vegas[5]と同様に 3 つのフェーズを持つ。それらは、ボトルネックとなるルータのキュー内パケット数の推定値(*diff*)で制御されており、以下のように計算される。

$$diff = cwnd \frac{(RTT - RTT_{\min})}{RTT}$$

この *diff* の値によって、現在のネットワークの状況を推定し、3 つのフェーズを適応的に切り替え、輻輳ウインドウを調節する。

$$cwnd_{new} = \begin{cases} cwnd_{last} + W_{inc} / cwnd_{last}, & \text{if } diff < \alpha \\ cwnd_{last} + (-diff + \alpha) / cwnd_{last}, & \text{if } diff > 3*\alpha \\ cwnd_{last}, & \text{otherwise} \end{cases}$$

$$\text{cwnd}_{new} = reno\_cwnd, \quad \text{if } cwnd_{new} < reno\_cwnd$$

ここで、*reno_cwnd* は TCP-Reno と同様の制御を行う輻輳ウインドウである。α、W_{inc} はそれぞれフェーズを制御する閾値及び高速に増加させるパラメータであり、以下のように設定する。

$$\alpha = \frac{RE * tcp\_tick}{packet\_size * 8}, \quad W_{inc} = \max(1, BE / 12Mbps)$$

ここで、*BE*(Bandwidth Estimation)、*RE*(Rate Estimation)は、それぞれ TCP-Westwood と同様に求めた回線帯域及び現在の使用帯域の推定である。*tcp_tick* は TCP のタイマー粒度であり、[6]では 1ms と設定しているが、本実験では 4ms と設定した。それは、エミュレータと送受信端末の内部遅延が 3ms 程度あることが確認されたためである。

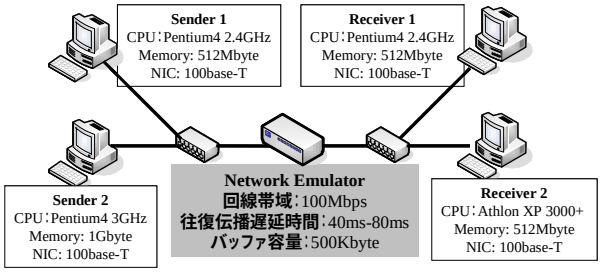

図 1.　評価環境

3．実装評価

TCP-Fusion を Linux 上(Kernel 2.6.15)に実装し、ネットワークエミュレータを用いた実験ネットワークにおいて評価を行った。評価環境と、送受信端末のスペックを図 1 に示す。全ての端末において、TCP 送受信バッファサイズは 10Mbyte に設定している。エミュレータのバッファ容量は 500Kbyte とし、これは往復伝播遅延時間を 40ms と設定した場合の帯域遅延積に相当する。TCP トラヒックは Iperf で 10 分間発生させ、スループットの計測は受信側に表示されるものを用いている。

(1) TCP-Fusion の輻輳ウインドウの動き

TCP-Fusion の輻輳ウインドウの振る舞いを確認するために、TCP-Reno と競合させて実験を行った。エミュレータの設定では、往復伝播遅延時間を 80ms と設定している。

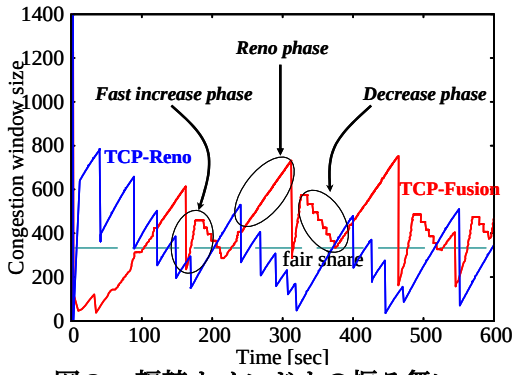

図 2.　輻輳ウインドウの振る舞い

この場合、バッファ容量の 500Kbyte は帯域遅延積の半分に相当する。図 2 に、TCP-Fusion と TCP-Reno の輻輳ウインドウを示す。パケット廃棄が同期していないために、結果的には見づらいものとなっているが、回線帯域を利用できていない時には、輻輳ウインドウを高速に増加させ、逆に利用できている時には、競合する TCP-Reno の増加に合わせて輻輳ウインドウを減少させていることを確認することが出来る。スループット計測においては、TCP-Fusion が 46.0Mbps、TCP-Reno は 39.2Mbps であった。2 本の TCP-Reno フローを流して実験した結果、平均して 38.5Mbps であったので、TCP-Fusion は TCP-Reno との親和性を実現できていることがわかる。

(2) パケット廃棄率に対する耐性評価

TCP-Fusion のパケット廃棄率に対するスループット評価を行った。エミュレータの設定は往復伝播遅延時間を 40ms とし、パケット廃棄率を 10^{-6} から 10^{-1} まで変化させた。バッファ容量は帯域遅延積である 500Kbyte に設定している。図 3 に、TCP-Fusion、HSTCP、TCP-Reno の結果を示す。TCP-Fusion はパケット廃棄率が 10^{-4} 以上の場合においても高いスループットを得ることがわかり、TCP-Reno の 7 倍程度スループット向上が見られる。

次に、図 4 には図 3 と同様の設定で、TCP-Reno と TCP-Fusion もしくは HSTCP が競合した場合のスループット評価を示す。この場合、図 3 の結果から、TCP-Reno ではパケット廃棄率が 10^{-4} 以下であるなら、公平な帯域である 50Mbps の帯域を得ることが可能なので、理想的にはパケット廃棄率が 10^{-4} 以下までは TCP-Reno と公平に帯域を分け合い、それ以降は TCP-Reno が利用することが出来ない余剰帯域を利用することが望ましいことになる。TCP-Fusion は、ほぼ理想的な動きをしており、パケット廃棄率が 10^{-5} までは TCP-Reno と公平に帯域を分け合い、それ以降は本来の高速性により、余剰帯域を利用することが出来ている。一方で、HSTCP では、TCP-Reno のスループットを押し下げてしまっている。

(3) 複数のフローが競合する場合

図 5 に、TCP-Reno と TCP-Fusion もしくは HSTCP のフローが合計 5 本競合する場合の各プロトコルのフロー1 本あたりのスループットを示す。横軸に TCP-Reno のフロー数をとり、残りが TCP-Fusion もしくは HSTCP のフロー数となる。

図 5 を見ると、HSTCP は TCP-Reno のスループットを大幅に押し下げてしまっていることが確認でき、特に、TCP-Reno のフロー数が 4 本の場合においては、4 倍以上のスループットを得ている。一方で、TCP-Fusion では、TCP-Reno のフロー数に関係なく、公平に帯域を分け合うことが出来ている。

3. まとめと今後の展開

本稿では、我々が提案している TCP-Fusion 方式を Linux 上に実装し評価を行った。評価の結果、TCP-Fusion は、パケット廃棄率が高い環境下においても高いスループットを得ることがわかった。また、TCP-Reno と競合する場合においては、ほぼ公平に帯域を分け合えることも確認することが出来た。

今後の展開として、より特性が顕著に現れる広帯域ネットワーク状況下での実験を考えている。

4. 参考文献

[1] S. Floyd, “Highspeed TCP for Large Congestion Window”, IETF RFC3649, December 2003.
[2] H. Shimonishi, and T. Murase, “Improving Efficiency-endliness Tradeoffs of TCP Congestion Control Algorithm”, in proc. of Globecom, December 2005.
[3] K. Tan J. Song, Q. Zhang, and M. Sridharan, “Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks”, in proc of PFLDnet, February 2006.
[4] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", in proc. of Mobicom, July 2001.
[5] L.S. Brakmo and L.L. Perterson: “TCP Vegas: End-to-End Congestion Avoidance on a Global Internet,” IEEE Journal on Selected Areas in Communication, Vol. 13, November 1995.
[6] 兼子，甲藤，“ 高速回線のための TCP 輻輳制御方式(TCP-Fusion)の提案” , in proc. of IEICE Tech. Report, March 2006.

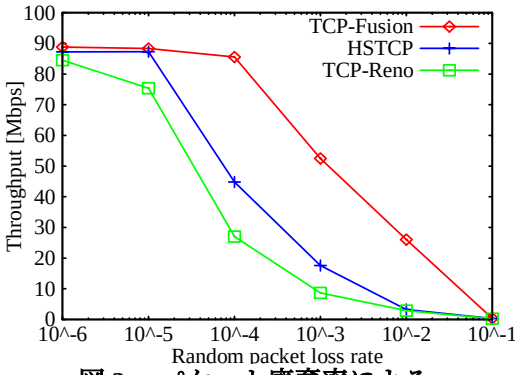

図 3. パケット廃棄率による スループット評価

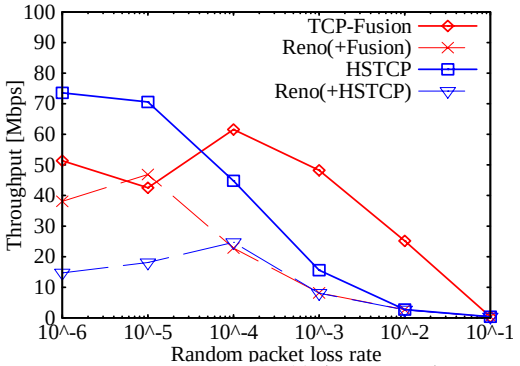

図 4. TCP-Reno と競合する場合の スループット評価

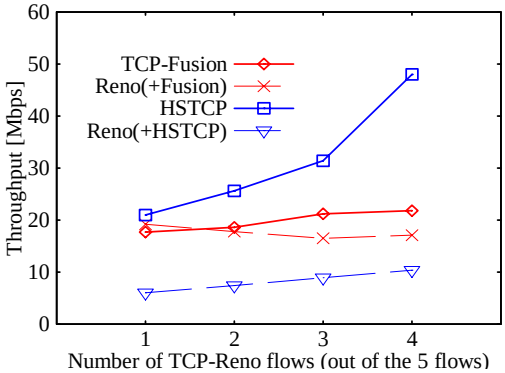

図 5. 複数フローの場合の スループット評価