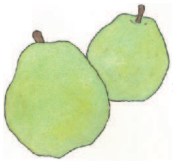
インターネットの輻輳制御技術



技術解説

長谷川　剛*

Congestion control mechanisms in the Internet

Key Words：Internet, TCP, Congestion Control

はじめに

Transmission Control Protocol（TCP）[1] は現在のインターネットにおけるトランスポート層プロトコルとして最も多くのネットワークアプリケーションが利用しており、TCP トラヒックは現在のインターネットトラヒックの大部分を占めている。また、インターネットは様々な種類のネットワークを取り込むことによって大規模化し、指数関数的な拡大を続けている [2]。その結果、TCP が誕生した 1970 年代当初には想定することができなかったネットワーク環境が発生している。TCP において最も重要な機能はネットワーク輻輳を回避・検知・解消する輻輳制御機構 [3] であり、多様化するネットワーク環境において安定的に TCP による通信を行なうことができる大きな要因である。

TCP がどのようなネットワーク環境においても安定的な通信を行なうことができる、というロバスト性は、逆に言うと、個々のネットワーク環境における性能の最適化の観点では劣る場合があるということを意味する。これは、インターネットがさまざまなネットワークを IP というルーティングプロトコルで接続しているという性質を鑑みると、止むを得ない性質であると考えられる。しかし、特に近年の光ファイバ技術や無線ネットワーク技術によるアクセスネットワーク環境の劇的な進歩にともない、そのような環境における TCP の性能が着目されることが多くなり、様々な問題が指摘されつつある。例えば、ネットワーク輻輳に加えてリンクエラーによりパケット廃棄が発生、および変動する無線ネットワークや、端末の移動によりエンド間の経路が通信中に変化し、スループット低下が発生するモバイル環境などが挙げられる。これらを始めとするさまざまな問題を解決するために、これまでに多くの TCP に対する改善が行われてきた。

また、近年のネットワークの高速化により、TCP コネクションが利用できるネットワークの帯域遅延積（リンク帯域とエンドホスト間の伝播遅延時間の積）が飛躍的に増大している。例えば、ラウンドトリップ時間（RTT）が約 130 msec となる太平洋を狭んだ 2 台のエンドホスト間の最低帯域が 100 Mbps から 1 Gbps 程度である、という環境も一般に利用可能となりつつある。このような高速・高遅延ネットワーク環境において、現在多くの OS の TCP 実装が基本としている TCP Reno を用いると、その輻輳制御方式の特徴が原因となって、リンク帯域を十分使うことができない、という問題が指摘されている。これは、TCP Reno が旧来の低速ネットワークを想定して設計されていること、またインターネットユーザがよりスループットなどの性能に敏感になっていることなどに起因していると考えられる。

本稿では、このような新たなネットワーク環境に対応するために行われてきた、TCP の輻輳制御機構に関する近年の研究動向について概説する。特に、無線ネットワーク環境、および高速・高遅延ネットワーク環境における従来 TCP の輻輳制御機構の問題点を整理し、提案されている改善手法について述べる。

*Go HASEGAWA

1974年3月生
大阪大学大学院基礎工学研究科博士前期課程修了（1997年）
現在、大阪大学　サイバーメディアセンター先端ネットワーク環境研究部門
准教授　博士（工学）　情報ネットワーク
TEL：06-6850-6864
FAX：06-6850-6868
E-mail：hasegawa@cmc.osaka-u.ac.jp

1　TCPの輻輳制御方式

TCPはウィンドウサイズと呼ばれる、一度にネットワークに送出することができるデータ量を動的制御することによって、ネットワークの輻輳制御を行っている。TCP Renoの輻輳制御方式は、スロースタートフェーズおよび輻輳回避フェーズと呼ばれる2つのフェーズから構成され、それぞれにおいて輻輳ウィンドウサイズ（cwnd）の増加速度が異なる。スロースタートフェーズにおいては、1つのACKパケットを受信するごとに輻輳ウィンドウサイズを1パケット増加させる。一方、輻輳回避フェーズにおいては、1つのACKパケットを受信するごとに輻輳ウィンドウサイズをその逆数分だけ増加させる。すなわち、TCP Renoの輻輳ウィンドウサイズをw_{reno}とすると、その更新アルゴリズムは以下のように表すことができる。

$$w_{\mathrm{reno}} \leftarrow \begin{cases} w_{\mathrm{reno}} + 1 & (w_{\mathrm{reno}} < s_{\mathrm{reno}}) \\ w_{\mathrm{reno}} + \dfrac{1}{w_{\mathrm{reno}}} & (w_{\mathrm{reno}} \geq s_{\mathrm{reno}}) \end{cases} \quad (1)$$

ここで、s_{reno}は、TCP Renoがスロースタートフェーズから輻輳回避フェーズに移行する時のしきい値ssthreshである。送信側端末においてACKパケットを受信する度に上式が用いられることによって、$w_{\mathrm{reno}} < s_{\mathrm{reno}}$の場合にはRTTごとにcwndが2倍になり、$w_{\mathrm{reno}} \geq s_{\mathrm{reno}}$の場合にはRTTごとにcwndが1だけ増加する。

一方、パケット廃棄を検出した場合には、次式のように輻輳ウィンドウサイズを減少させる。

$$w_{\mathrm{reno}} \leftarrow \begin{cases} w_{\mathrm{reno}}/2 & （重複ACK） \\ 1 & （タイムアウト） \end{cases} \quad (2)$$

すなわち、TCP Renoはパケット廃棄を検出するまで輻輳ウィンドウサイズを増加させ続け、パケット廃棄をきっかけに減少させる。これは、TCP Renoがパケット廃棄の発生をネットワーク輻輳の指標と見なしていることに起因する。本稿ではこのようにパケット廃棄をネットワーク輻輳の指標として用いる手法をloss-based手法と呼ぶ。

2　無線環境におけるTCP輻輳制御機構

2.1　問題点

無線ネットワーク環境におけるTCPの性能評価に関しては、数多くの研究がこれまでに行われており、それらを通じていくつかの問題が明らかになっている。本節では、それらのうち、輻輳に無関係なパケット廃棄、およびコネクション間の不公平性に関して説明する。

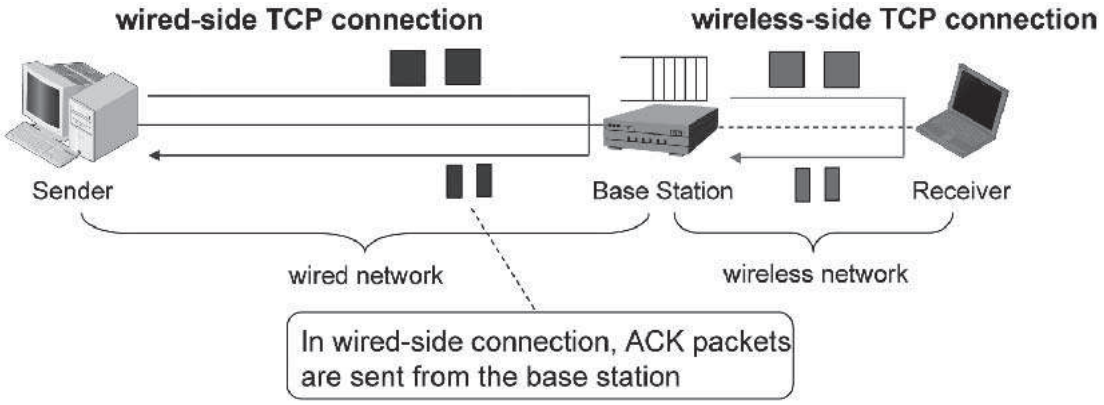
無線ネットワークは有線ネットワークに比べて通信路のビットエラー率が非常に高く、無線リンクロス、すなわち、ビットエラーが原因となるパケット廃棄が頻繁に発生する。一方、TCPはパケット廃棄を検出すると、それはネットワーク輻輳の徴候であると判断し、ウィンドウサイズを減少させることによって、自身のデータ転送速度を低下させる[3]。したがって、無線ネットワーク環境におけるTCPデータ転送において、無線リンクロスによってパケット廃棄が発生すると、不必要なウィンドウサイズの減少が発生し、スループットが低下する。

2.2　改善手法

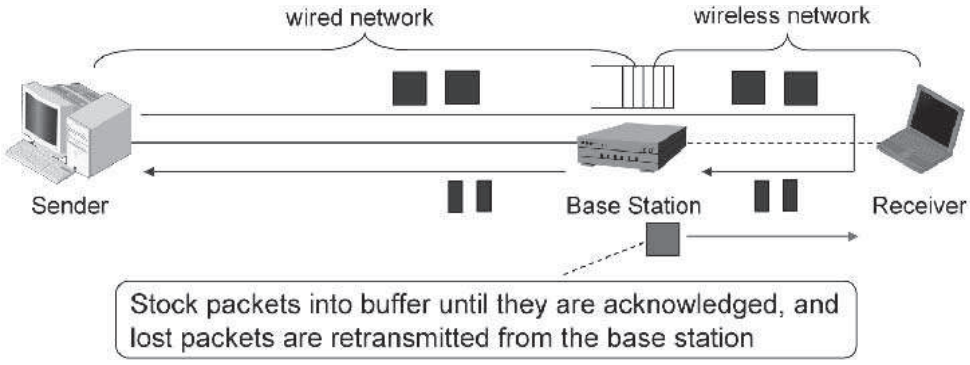
無線ネットワークにおけるTCP性能の改善に関する研究はこれまで多数行われており、無線基地局との連携手法、送信側TCPの改変手法、およびMAC層とTCP/IP層の連携に基づくクロスレイヤ制御などが挙げられる。本章では、それらの中でも特に、無線基地局の改変を必要とするものに関して、利点と欠点を述べる。なお、送信側TCPの改変手法は、無線基地局の改変が必要な方式に比べて、送信側TCPの変更のみで実現することができるため、より導入が容易であると考えられる。が、得られる性能改善効果は、一般的に無線基地局の改変が必要な方式に比べて低くなる。

2.2.1　無線基地局の改変をともなう手法

無線ネットワークにおけるTCP性能向上手法の1つとして、有線ネットワークと無線ネットワークの境界に存在する無線基地局を改変するものが挙げられる。これは、有線ネットワークと無線ネットワークの特性の違いが与える影響をそれぞれ局所化することによってTCP性能を向上させるものである。これらの手法は主に、基地局においてコネクション



（a）TCPコネクションの分割



（b）TCPコネクションのスヌーピング

図1：無線基地局の改変をともなう手法

分割を行なうもの、およびコネクション監視（スヌーピング）を行なうものに分類される。図1に、無線ネットワークに存在する端末が受信側TCPとなる場合の、両方式の挙動を示す。

コネクション分割を行なう手法においては、通常エンド端末間に1本設定されるTCPコネクションを、無線基地局において分割する（図1(a)）。有線ネットワーク側においては、有線側のエンド端末とのデータパケットおよびACKパケットのやり取りを無線側端末に代わって行なう。こうすることによって、有線ネットワークにおけるデータ転送速度が、無線ネットワーク環境に影響を受けないようにすることができる。また、無線ネットワーク側においては、発生するパケット廃棄を全て無線リンクロスに基づくものと見なすことができるため、無線リンクロスによるパケット廃棄に対してはウィンドウサイズを減少させない、といった制御により、スループット低下を回避することができる。また、無線ネットワークで発生したパケット廃棄に対するパケット再送を、基地局から行なうことができるため、再送効率が向上する。

一方、文献[4]などにおいて提案されているコネクション監視（スヌーピング）手法は、廃棄されたパケットの基地局からの再送を、コネクション分割を行なわずに実現する手法である（図1(b)）。具体的には、基地局に監視のためのエージェントを導入し、通過するTCPコネクションのデータパケットを、対応するACKパケットが逆向きに通過するまでキャッシュとして保存し、ACKパケットのシーケンス番号を監視することによって、再送すべきデータパケットを決定する。また、重複ACKパケットが通過する際には、それを適宜削除することによって、エンド端末からのパケット再送を抑制する。

これらの手法は、インターネットにおけるプロトコル設計の際の指針として従来考えられてきたエンドツーエンド原理[5]に反するものであるが、近年はオーバレイネットワークやファイアウォールシステムなど、ネットワーク内においてセッションの切

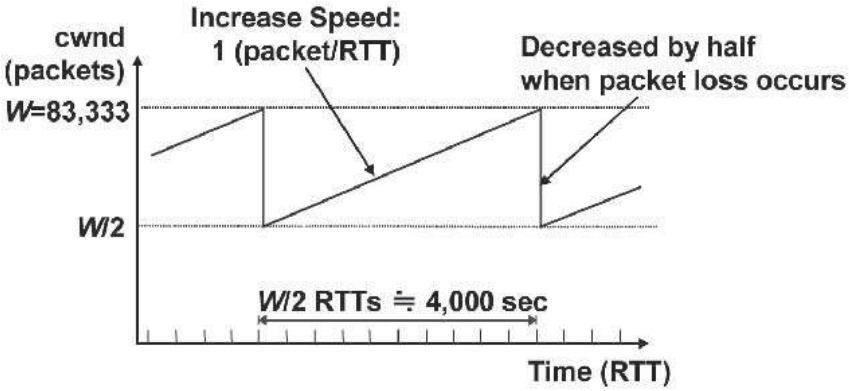


図2：TCP Reno の高速・高遅延環境における問題点

断・中継を前提とするシステムが普及しており、導入障壁は以前ほど高くはないと考えられる。

3　高速・高遅延環境における TCP 輻輳制御機構

3.1　問題点

TCP Reno を高速・高遅延ネットワーク環境において用いると、大きなリンク帯域を十分使う程度のスループットを得ることができないという問題が指摘されている [6]。図2に、送受信端末間のラウンドトリップ時間（RTT）が 100 msec、リンク帯域が 10 Gbps であるような大きな帯域のリンク上で、パケット長が 1500 Byte である1本の TCP Reno コネクションを用いてデータ転送を行ったときの、輻輳ウィンドウサイズの変化の様子を示す。この環境においてはリンク帯域を十分使う程度のスループットを得るためには、パケット廃棄率が 2×10^{-10} 以下である必要がある [6]。これは現在の光ファイバ技術では実現困難な性能である。また図2から、一度パケット廃棄が発生すると、輻輳ウィンドウサイズが回復するまでに、40000 RTT（約 4000 秒）以上の時間を要することがわかる。これは、TCP Reno において 1 Gbps を超えるスループットを継続的に得ることは現実的には不可能であることを表している。この問題は、式 (1) からわかるように、輻輳ウィンドウサイズの増加の幅がラウンドトリップ時間（RTT）ごとに1パケットと非常に小さいにもかかわらず、式 (2) に示すように、パケット廃棄を検出した際にウィンドウサイズを 1/2 以下へと大きく減少させるために、輻輳ウィンドウサイズがなかなか大きくならないことに起因している。

このような高速・高遅延ネットワークにおける TCP Reno の問題点に対する数多くの改善手法が近年提案されている。本章では、それらのうち主要なものを紹介する。それらの改善手法の多くは、TCP Reno がパケット廃棄をネットワーク輻輳の指標として判断しているのに対して（あるいはそれに加えて）、別の指標を導入している。本章では新たに導入している指標によって分類を行っている。また本章では、特に指定しない限りは、輻輳回避フェーズにおける輻輳ウィンドウサイズの増減アルゴリズムについて説明する。なお、スロースタートフェーズに関しては、多くの改善手法が TCP Reno と同じアルゴリズム（式 (1) の1行目）を採用している。また、輻輳回避フェーズにおいては、輻輳ウィンドウサイズがある値以下である場合には、TCP Reno と同じアルゴリズム（1節）を用いることで、低速ネットワーク環境における TCP Reno との親和性を確保している手法も存在する。

3.2　Loss-based 手法

3.2.1　HighSpeed TCP(HSTCP) [6]

HSTCP は、高速・高遅延環境向けの改善手法として比較的初期に提案された手法である。HSTCP は TCP Reno と同様、パケット廃棄のみをネットワーク輻輳の指標として利用するが、現在の輻輳ウィンドウサイズの大きさに合わせて、輻輳ウィンドウサイズの増加速度およびパケット廃棄検出時の減少幅を調整する。すなわち、HSTCP は現在の輻輳ウィンドウサイズが大きいほど、その増加速度を大きくし、減少幅を小さくすることを意味している。これにより、HSTCP はネットワークの帯域遅延積の

大きさに応じたウィンドウサイズの増加・減少速度を用いることができる。

しかし、このようなTCPの改良手法は、TCP Renoに比べて積極的に輻輳ウィンドウサイズを増加させるため、そのようなプロトコルがTCP Renoコネクションと共存した場合、それらのプロトコルは高いスループットを獲得する反面、共存するTCP Renoコネクションのスループットを低下させるという問題点がある。この問題に対する改善手法として我々の研究グループではgentle HighSpeed TCP手法を提案している[7]。

3.3 Delay-based手法

Loss-based手法はネットワーク内でパケット廃棄が発生するまで輻輳ウィンドウサイズの増加を停止しないため、理想的に動作した場合においても、周期的なパケット廃棄の発生を避けることができない。一方、ルータの出力リンクにおいて高負荷時にパケットが蓄積されるバッファがFIFO規律に従っている場合、バッファが一杯になりパケット廃棄が発生する前に、そのリンクにおいてバッファリング遅延が増大することが期待される。そこで、各パケットのRTTを監視し、その増大をネットワーク輻輳の初期段階の指標として利用する手法（本稿ではdelay-based手法と呼ぶ）が提案されている。理想的に動作すると、loss-based手法では避けることのできないパケット廃棄を完全に回避することが可能となる。

3.3.1 FAST TCP[8]

FAST TCPはTCP Vegasと同様に、観測された最小のRTTである*base RTT*と現在のRTTである*RTT*を利用し、以下の式にしたがって輻輳ウィンドウサイズを増減させる。

$$w_{\text{fast}} \leftarrow \min\left\{2w_{\text{fast}}, (1-\gamma)w_{\text{fast}} + \left(\frac{baceRTT}{RTT}w_{\text{fast}} + \alpha\right)\right\}$$

αはパラメータであり、ネットワーク内滞留パケット数の目標値に相当する。TCP Vegasと異なり、目標となる輻輳ウィンドウサイズが現在値に比べて大きい場合には輻輳ウィンドウサイズを指数的に増加させるため、ネットワークの帯域遅延積が大きい場合にもすばやくネットワーク利用率を向上させることができる点が特長である。その反面、パラメータαの適切な設定が難しいという欠点を持つ。

3.4 Hybrid手法

FAST TCPに代表されるdelay-based手法は、それが単独で用いられる場合にはスループット、公平性、収束速度などの面で優れていることが明らかになっている。しかし、TCP RenoやHSTCPのようなloss-based手法と混在した環境においては、delay-based手法を用いるコネクションのスループットが低下するという問題が文献[9]などにおいて指摘されている。これは以下の理由による。混在環境においてネットワーク帯域が使い切られ、ルータバッファにパケットが蓄積し始めると、RTTが増加する。その際、delay-based手法はRTTの増加にともない輻輳ウィンドウサイズを小さくするが、loss-based手法はパケット廃棄が発生するまで輻輳ウィンドウサイズを大きくし続ける。したがって、ボトルネックリンクを両手法のコネクションが共有した場合、delay-based手法のコネクションは、loss-based手法のコネクションに比べてスループットが低下する。

この問題に対し、delay-based手法にloss-based手法を組み合わせるHybrid手法が提案されている（例えばCompound TCP[10]など）。これらの手法は、通常のdelay-based手法と同様に、RTTが増加しておらずネットワーク帯域が使い切られていないと判断された場合には輻輳ウィンドウサイズをTCP Renoよりも大きく増加させる。さらに、輻輳ウィンドウサイズの増加幅をTCP Reno相当の部分と、そうでない追加部分に分けて管理する。その後、RTTが増加し始めると、輻輳ウィンドウサイズの（大幅な）増加を停止する。その後、TCP Renoと同じ増加幅で輻輳ウィンドウサイズを増加させ、loss-based手法を用いる（パケット廃棄の発生まで輻輳ウィンドウサイズを大きくし続ける）。すなわち、ネットワークの未使用帯域がある場合には、delay-based手法によってそれを高速に使い切るように動作し、輻輳時にはloss-based手法で動作することによって、共存するTCP Renoとの公平性を維持している。

3.5　その他の手法

その他、パケット廃棄が発生した時の輻輳ウィンドウサイズを記憶し、その値を基にその後の輻輳ウィンドウサイズの制御を行うCUBIC-TCP [11]（Linuxの標準プロトコル）、またACKパケットの到着間隔から現在のスループットを推測し、その値を輻輳ウィンドウサイズの制御に用いるTCP Westwood [12] などが提案されている。また、Explicit Congestion Notification（ECN）[13] などを使って、ネットワーク側から明示的に利用可能性帯域や輻輳の有無などの情報を明示的に取得することで、エンド端末における輻輳制御の効率を高める検討も行われている。

4　おわりに

本稿では、TCPの輻輳制御機構に関する近年の研究動向を概説した。主に無線ネットワーク環境および高速高遅延ネットワーク環境における問題点を整理し、近年提案されている様々な改善手法を紹介した。無線ネットワーク環境に関しては、今後もWiMAXやLTEなど、従来とは異なる特性を持つネットワークが登場するため、これらのネットワークにおいてTCPを用いた場合の性能評価や、特性を考慮した輻輳制御機構の改善が求められると考えられる。また、高速・高遅延ネットワーク環境向けの改善手法に関しては、有効な改善手法が出揃った印象があり、今後は、どの改善手法が最も有効なのか、あるいは、条件に応じた輻輳制御機構の切り替え手法などに関する研究が進むものと考えられる。

参考文献

[1] J. B. Postel, “Transmission control protocol,” *Request for Comments 793*, Sept. 1981.

[2] Hobbes’ Internet timeline 10. available at http://www.zakon.org/robert/internet/timeline/.

[3] V. Jacobson, “Congestion avoidance and control,” in *Proceedings of ACM SIGCOMM ’88*, pp. 314 － 329, Aug. 1988.

[4] H. Balakrishnan, S. Seshan, and R. H. Katz, “Improving reliable transport and handoff performance in cellular wireless networks,” *ACM/Baltzer Wireless Networks*, vol. 1, pp. 469 － 481, Dec. 1995.

[5] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM Transactions on Computer Systems*, vol. 2, pp. 277 － 288, Nov. 1984.

[6] S. Floyd, “HighSpeed TCP for large congestion windows,” *Request for Comments 3649 (Experimental)*, Dec. 2003.

[7] Z. Zhang, G. Hasegawa, and M. Murata, “Performance analysis and improvement of HighSpeed TCP with TailDrop/RED routers,” *IEICE Transactions on Communications*, vol. E88-B, pp. 2495 － 2507, June 2005.

[8] C. Jin, D. X. Wei, and S. H. Low, “FAST TCP: motivation, architecture, algorithms, performance,” *in Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

[9] G. Hasegawa, K. Kurata, and M. Murata, “Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet,” *in Proceedings of IEEE ICNP 2000*, Nov. 2000.

[10] K. T. J. Song, Q. Zhang, and M. Sridharan, “Compound TCP: A scalable and TCP friendly congestion control for high-speed networks,” *in Proceedings of PFLDnet 2006*, Feb. 2006.

[11] I. Rhee and L. Xu, “CUBIC: A new TCP friendly high-speed TCP variant,” *in Proceedings of PFLDnet 2005*, Feb. 2005.

[12] TCP WESTWOOD Home Page. available at http://www.cs.ucla.edu/NRL/hpi/tcpw/.

[13] K. Ramakrishnan, S. Floyd, and D. Black, “The addition of explicit congestion notification (ECN) to IP,” *RFC 3135*, Sept. 2001.