コネクションごとのTCP最適化機構の設計と実装

2D−3

田原 裕市郎† 小川 晃通† 杉浦 一徳†

慶應義塾大学院 政策メディア研究科†

1 はじめに

現在, インターネットに接続されているホストコンピュータ, またはそれに類する情報端末の種類は多岐に渡っている. 利用可能なネットワーク帯域に限ってみても, 広く一般的に利用されている電話回線を利用した10Kbps程度のものからイーサネットを利用した100Mbpsまで存在し, 最近実用化が進んでいるWDM伝送装置を利用した場合は10Gbpsという巨大な通信帯域を提供する.

インターネットは, 通信に伴うハードウェアの特性を柔軟に吸収することによって広く普及することができたネットワークシステムである. しかし, 今日のこれほどのハードウェア格差は当初の想定を遥かに超えている. そのような格差があるにも関わらず単一の仕組みを適用しようとしているため通信効率の理論値と実測値が大きく開いてしまうという問題が発生している.

そのため近年では, 特に通信効率が悪化する場合に対応するため, インターネットの通信方式であるTCPに改良を施す研究が数多くなされている. しかし, それらの研究成果は優れた結果が得られても, あまり普及していない. その原因として, 元々のTCPが高い汎用性を持つのに対して, 特定の環境下での効率化を前提になされたTCP改良は別の環境下で性能がでないという問題を抱えているためである.

そこで, 本研究では通信環境に応じてTCPの挙動を変更させることで, 効率的に転送を行う機構を提案し, その有効性を示す.

2 環境設定

本稿では, 特に以下のような環境下における通信効率化を目指す.

イントラネットのように, 同一の組織や建物内部における通信でも, TCP/IPを用いて通信を行うケースは多い. これはユーザから見て, インターネットと内部への通信それぞれに対して同一のアプリケーションを使用できる利便性があるためである.

しかし, インターネットを利用した通信では, 輻輳などによるパケット損失や, 十分な通信帯域が確保できないなど, エンドノードの性能限界まで転送効率を高めることができるとは限らない. TCPには, そのような場合に対応するためフロー制御機構や輻輳制御機構を備えている.

そのようなTCPの機構は, インターネットを介する通信では効率的に作用するが, イントラネットなどの内部間通信では, その限りではない. 同一セグメントにおける通信ではルータによるパケット損失の可能性は無い. また、同一組織ならば, 通信インフラの問題は改善することが容易である. しかしながら, 通信インフラとエンドノードの能力に余力があっても, 通信性能がTCPによって抑えられていることは多い.

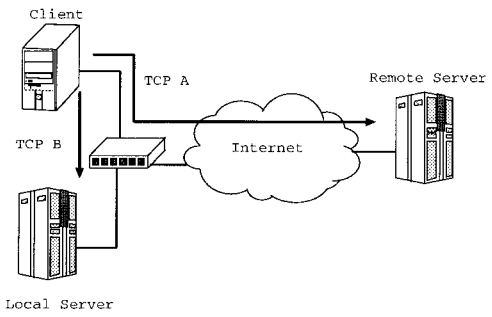

図 1: 概念図

そのため, 図1のように通信先によって挙動を変化させるTCPが存在すれば内部間での通信効率を高めることができる本実装は, 外部との通信では通常のTCP機構を, 内部との通信では高速に通信を行うTCP機構を, 動的に切り替えながら利用できる機構を実現する.

3 設計

3.1 方針

TCP通信を高速に行う手段として次のようなものが考えられる.

1. コネクションの確立処理を省略する.
2. 初期ウィンドウサイズを大きくする [1].

"The Design and Implementation of Optimezed TCP Mechanism per Connections"
Yuichiro Tahara† Akimichi Ogawa† Kazunori Sugiura†
Graduate School of Media and Governance, Keio University†
Faculty of Environmental Information, Keio University‡
Keio University Shonan Fujisawa Campus
5322, Endo, Fujisawa, Kanagawa 252, Japan
E-Mail: ash@sfc.wide.ad.jp

Members of Murai LAB, especially those in STREAM

3. 送受信バッファサイズを大きくする.

本設計では、比較的容易に実現できる2番目の方式を採用する。

一般的にTCPの実装ではフロー制御機構としてスロースタート機構を有している. これはネットワークに突発的なパケット流量を起こさないための機構であるが, 同一セグメントにおける通信では, ウィンドウサイズが早期に適正サイズになることを阻害している. 初期ウィンドウサイズを大きくすることで通信開始時から速い転送速度で通信できる.

3.2 実装環境

実装を行った環境は表1の通り.

表 1: 実装環境

Kernel	Linux 2.4.7
libc	glibc 2.2.2
Cコンパイラ	egcs-1.1.2

本実装は,Linuxカーネルの TCPプロトコルスタックを改変することで実現した.

3.3 手法

本実装はウィンドウサイズの初期部に対する改変なので, 改変を施すのはクライアント側となる. 図2は TCP のスリーウェイハンドシェイク時にカーネル内部で呼ばれる関数を表したものである。まず,TCP 層では tcp_v4_connect() 関数が呼ばれ,SYN パケットを送出する. その後,TCP の状態を SYN_SENT に移行して,tcp_rcv_ stats_process() 関数を呼び出し,SYN:ACK を待つ.SYN:ACK を受け取ると tcp_init_ metrics() を呼び出し, 通信設定を行う.

tcp_init_metrics() は初期ウィンドウを設定するために,tcp_init_cwd() を呼び出す.

本実装は, 送信先 IP アドレスがローカルエリアの場合に tcp_init_cwd() 関数ではなく, 初期ウィンドウサイズを通信先によって広告された最大ウィンドウサイズに設定する.

4 まとめ

本稿では, 送信先 IP アドレスをもとに TCP の初期ウィンドウサイズを切り替える実装を行った. これは、すなわちコネクションごとに TCP の挙動を変化させたことを意味する. 最近の UNIX の実装でも,TCP のパラメータは動的に変化させることができる. 例えば, FreeBSD や Linux は sysctl によって,Sun Microsystems の Solaris では /dev/tcp によって行える. しかし, それらはカーネル全体に対する変更であって, コネクションごとに変更を行うことはできない. 送信先アドレスによって区別することでコネクションごとの細かな TCP の挙動変化を実現した.

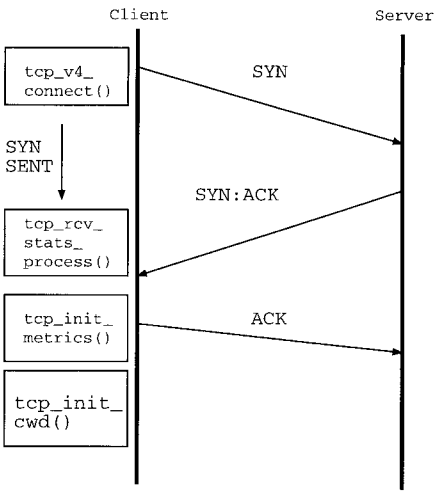

図 2: コネクション確立時の関数

4.1 今後の課題

今回は IP アドレスによる区分を行ったが, アプリケーションごとに TCP の挙動を変化させるにはポート番号などで区別する手法が考えられる.

また,TCP の挙動を変化させるには送信・受信バッファや拡張アルゴリズムなど様々な要素が存在する [2]. 今後, それらを切り替えられるようにすることを目指す.

参考文献

[1] M. Allman,S. Floyd,C. Partridge, ' Increasing TCP's Initial Window', RFC2414, September 1998

[2] M. Mathis, J. Mahdavi, 'TCP Rate-Halving with Bounding Parameters', [MM97], December 1997