OSレベルでのTCPのフロー制御によるQoSの実現

塩川 泰広　　本多 弘樹　　弓場 敏嗣
電気通信大学大学院情報システム学研究科

概要

単一ネットワーク上に複数のTCPコネクションが確立され、ネットワークの帯域が使い果たされた状況を考える。ミッションクリティカルなアプリケーションがあったとしても、各コネクションは平等に扱われる。その結果、優先したいコネクションが存在しても、その得られるスループットが大幅に減少してしまう場合が生じる。そこで、OSレベルで各コネクションに優先順位をつけ、優先順位が低いコネクションに対して流量を制限することで、優先度の高いコネクションのスループットを増加させる方式を提案する。本方式では、流量制限の方法としてTCPフロー制御であるウィンドウサイズ制御を利用する。これにより、一般ユーザが手軽にQoSを実現することが可能となる。

A realization of QoS by TCP flow-control at OS level

Yasuhiro SHIOKAWA, Hiroki HONDA, Toshitsugu YUBA
Graduate School of Information Systems, University of Electro-Communications

abstract

When all TCP connections in a network are fully being used at once, each TCP connection is treated fairly although in one connection there could be an important application. We propose a method that sets priority to TCP connections at OS level. The method is to reduce low priority TCP connections such that high priority TCP connections can have high throughput. This method uses the window size control mechanism which is one of the known TCP flow-control. Our proposed method is to ensure that a normal user can easily realize a QoS.

1 はじめに

種々のネットワークアプリケーションを使用していて、複数のTCPコネクションがはられていても、通常時は特に問題ない。ところが、使用している通信路の帯域幅がフルになってしまうと、お互いのコネクションが圧迫しあい、それぞれのコネクションが得られるスループットは落ちる。全てのコネクションはみな平等に扱われるため、ミッションクリティカルなアプリケーションが存在しても、その優先させたいコネクションは他のコネクション同様にスループットは減少してしまう。

この問題を解決するにはQoS(Quality of Service)を実現すればよいが、一括りにQoSと言っても、多種多様な種類が存在する。各通信路やルータ上で帯域予約を行うRSVP(Resource reSerVation Protocol)[1]、

インターネットやWANへのゲートウェイ上に専用の機器を設置し流量制御を行うPacket Shaper [2]、パケットにビットをたてることで他のパケットと差別化を行うDiffserv(Differenciated Services) [3][4]、データの入出力キューを制御するCBQ(Class-Based Queueing) [5]などが挙げられる。

これらの技術は、個人レベルで手軽に実現するのには困難がある。その理由は、ネットワーク全体でプロトコルの実装しなければならなかったり、ネットワーク内のポリシを管理できる権限や専用のハードウェアが必要であるからである。またルータキューイング方式では、基本的にアウトバンドトラフィックに対してしか機能しない。各々のアプリケーションごとにアプリケーションレベルでQoSを実現するという案もあるが、それぞれ別個に優先順位をつける必要があり簡便に運用するのは困難である。さらにQoSに対応していないアプリケーションではどうしようもない。

そこで、ユーザがOS上で各コネクションに優先順位をつけ、QoSを実現する方法を提案する。本研究の対象とする環境は、ISP(Internet Service Provider)にダイヤルアップIP接続しているとき、ローカルホストのOS上のみで各TCPコネクションの優先度制御を行いたい状況を主に対象としている(図1)。ISPにダイヤルアップIP接続していなくとも、以下の2つの要素が成立していれば、対象となりうる。まず1つめは、ローカルホストに対してボトルネックとなる通信路が直結している場合である。ISPにダイヤルアップIP接続している場合を例にとると、ローカルホストとISP側のダイヤルアップサーバとを結ぶ通信路は電話回線となるが、その電話回線の回線速度がボトルネックとなってしまう場合である。2つめは、アウトバンドトラフィックに比較してインバンドトラフィックの方が多い場合である。これは、FTPサーバやWebサーバといった情報を提供するサーバでない限り、通常の使用ではインバンドトラフィックの方が多いので、ほとんどの場合に当てはまる。

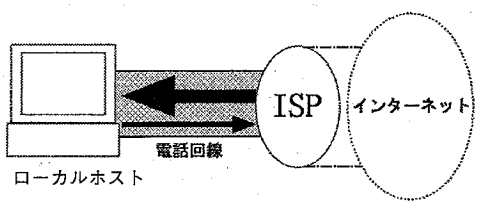

図1 研究対象環境

2 優先度制御のしくみ

優先度制御をどのように実現し、どういった効果が得られるかについて述べる。まずTCPには、ウィンドウサイズ制御というフロー制御の機構が用いられている。これはTCPコネクションを利用してデータをやりとりするさいに、受信側が送信側にあとどれだけデータを受信できるかという許容範囲を通知する。ウィンドウサイズ通知を受け取った送信側は、その分の量だけデータを一度に送信するというものである。

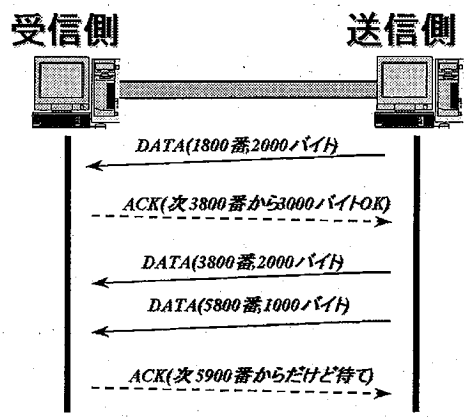

図2 ウィンドウサイズ制御

通信量が増していくと、ローカルホストに直結している通信路はすぐに帯域を使い果たしてしまう。そして輻輳状態になると、通常はどのコネクションでもパケットロスが生じる。パケットロスを検知した送信側は、自らウィンドウサイズを縮小し、一度に送るデータを減少させる。その結果、スループットが減少し、輻輳は回避される。本来、どのコネクションに対しても、ウィンドウサイズは縮小していくものである。しかし優先度の低いコネクションのみウィンドウサイズを意図的に縮小させることにより、優先度の高いコネクションのウィンドウサイズは大きいままに維持され、高いスループットを確保できる。そうすることによって優先度制御が可能となる。

図3 優先度制御のしくみ

ここで、優先度の低いコネクションの現在のウィンドウサイズを無理矢理縮小させるのではなく、そのコネクションの最大ウィンドウサイズの値を縮小させる。現在のウィンドウサイズを変更してしまうと、その後さらに激しい輻輳が発生したときに、対応しきれなくなる。そのため、最大ウィンドウサイズのみを定義し、本来のTCP フロー制御の機能を有効に働かせる。そして輻輳に対しても柔軟に対応できるようにする。

3 実装方法

対象OSはLinux(カーネル2.0.38)とする。まず、ローカルホストに直結しているボトルネックとなる通信路が、帯域を使い果たしてしまった状態を検知するため、その通信路の最大帯域幅を OS に告知する。これは bottlenet というコマンドによって[Kb/s]単位で指定する。実際のスループットを監視してボトルネックに達したかを検知するので、理論値ではなく実測値を用いる。

Linux(カーネル 2.0.38)の標準の実装では、全TCP コネクションの最大ウィンドウサイズは 32767(=MAX_WINDOW)と固定に定義されている。これを自作のシステムコールを通して、各コネクションごとに動的に定義できるようTCP モジュールを変更した。先程述べたように、常時最大ウィンドウサイズが変更されるのではない。総スループットがbottlenetコマンドで設定した値に達したときに、TCP モジュール内にフラグをたてる。そのフラグが立っているときに限り、優先度の低いコネクションの最大ウィンドウサイズが変更されるようにした。そして、ユーザが最大ウィンドウサイズを定義するための、netpriority というシス

テムコールを作成した。

上記のシステムにおいて、一般ユーザが優先度を自由に定義できるように、nine コマンドというものを作成した。

nine コマンドは、基本的に UNIX の nice コマンドの使用方法に似ている。まず nice コマンドに関して説明する。nice コマンドとは、ある実行したいコマンドに対して、1～20 までの数値を指定する。すると、その値の分だけ CPU に割り当てられる優先度が低くなるコマンドである。

(nice コマンドの使用例)

$ nice -4 gcc test.c

"gcc test.c"のコマンドが、4/20 段階 CPU の割り当て時間が少なくなり、処理する優先度が低くなる。

これに対して、nice コマンドはネットワークの優先度を低くしたいアプリケーションに対して、1 から 32767 までの値を設定する。すると、その値が最大ウィンドウサイズを設定する。そして、輻輳が発生したときに限り、設定された最大ウィンドウサイズとなり、流量が抑制される。

このようにして、優先度制御を実現する。なお、TCP の仕様の一つであるウィンドウサイズ制御を利用しているため、UDP の優先度制御をすることはできない。

・bottlenet コマンド

ローカルホストに直結している通信路の帯域幅を OS に告知する。

(使用例)

bottlenet 120

総スループットが 120[Kb/s]に達したときに、優先度制御を行う。

・nine コマンド

ネットワークの優先度を低くしたいアプリケーションに、最大ウィンドウサイズ(1 から 32767)を設定。

(使用例)

$ nine -500 ftp remotehost

この ftp セッションは、総スループットが bottlenet コマンドで設定した値に達すると、最大ウィンドウサイズが 500 に設定される。その結果、ある程度までしかスループットが出ないようになり、他の優先度の高いコネクションがその分優先される。

4 実験結果

まず、TCP ウィンドウサイズ制御を用いることにより、コネクションごとに差別化したスループットが得られるかどうかを考察する。2 つのホスト間を 10M-Ethernet で結び、2 つの FTP コネクションで、同時にそれぞれ 30[MB]のデータを転送した。そのとき、コネクション 1 は通常通りの最大ウィンドウサイズ(32767)、もうコネクション 2 の最大ウィンドウサイズをそれぞれの値に変化させたときに得られるスループットを測定した。その結果を図 4 に示す。

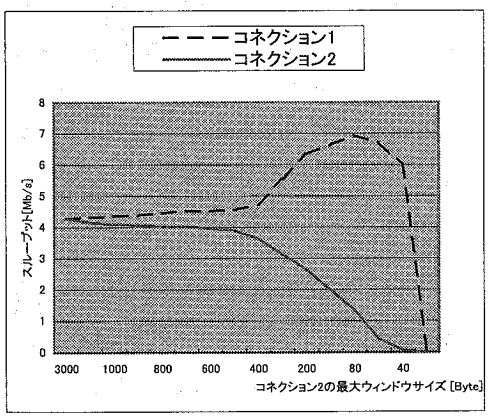

図 4 ウィンドウサイズに対するスループット

コネクション 2 の最大ウィンドウサイズ

が400を下回るあたりから、コネクション1のスループットが上昇している。このことから、ウィンドウサイズ制御を用いることで、コネクションごとに差別化したスループットが得られることが証明できる。なお、最大ウィンドウサイズが80を下回るあたりかわは、コネクション1のスループットも減少している。これは、コネクション2の最大ウィンドウサイズが小さくなりすぎて、無駄なオーバヘッドが発生してしまうためである。

同様の条件下で、30[MB]のデータを転送したときのスループットを測定した。2つのホストを直結していることから、10M-Ethernetがボトルネックとなる。よって、bottletnetコマンドの値を8000と定義した。その結果、ウィンドウサイズが未調整のままでは両コネクションとも4.30[Mb/s]だったのに対し、ウィンドウサイズを調整した場合は6.03[Mb/s], 2.13[Mb/s]という結果が得られ、優先度制御が実現できた。

5 今後の課題

本論文で述べた、TCPの最大ウィンドウサイズ調整のみでは、極度に優先度に差をつけたとき過剰なオーバヘッドが発生し、総スループットが落ちてしまう。そこで受信側OSで、最大ウィンドウサイズを一定のタイミングで通常のサイズと0を交互に切り替える方式を考える。一時的に完全に送信側のデータを停止させることにより、従来の方式で発生していた無駄なオーバヘッドを削減する。その結果、総スループットを落とさずに、優先度制御が実現できると思われる。この方式を現在実装中である。

参考文献

1) Braden, R., Zhang, L., Berson, S., Herzog, Jamin, S.: Resource ReserVation Protocol (RSVP), RFC2205
2) : Packet Shaper Web Page, http://www.packeteer.co.jp/
3) Nichols, K., Blake, S., Baker, F., Black, D.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC2474
4) Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services, RFC2475
5) Floyd, S.: References on CBQ (Class-Based Queueing) http://www.aciri.org/floyd/cbq.html
6) : The Linux Kernel Hackers' Guide, http://khg.redhat.com/HyperNews/get/khg.html
7) 上野, 木村, 海老原: 明示的輻輳通知を用いたTCPの優先輻輳方式, 情報処理学会論文誌, 40巻, 1号, 57頁-65頁 (1999).