4P－2

適応的QoS制御のためのQoS指定モデル

松井 康範†、木原 誠司†、盛合 敏†、徳田 英幸‡

日本電信電話株式会社†、慶應義塾大学‡

1　はじめに

インターネットにおいて、大容量連続メディアデータ伝送の要求が年々高まっている。ところで、バックボーンの伝送速度は急速に向上し、ビットあたりの伝送単価は急速に低下しているものの、各家庭を接続する加入者線の回線速度はなかなか増強されない。そのため、加入者線のボトルネックになるという、いわゆる Last-one-Mile 問題が今後顕著になると予想される。時間的制約を持つ連続メディアは再送によるエラー訂正を行ないにくいため、輻輳時のパケット廃棄による品質低下の影響を受けやすい。そのため、輻輳が発生しやすいネットワークの出入口での限定された帯域下における QoS 制御が重要になってくる (図 1)。

インターネットで連続メディアを伝送するために、IETF diffserv WG などでパケットの優先度を利用したトラフィック差別化の手法が議論されている。diffserv が実現されると、課金などによる制約条件の下、パケットに優先度ビットを付与することになる。ここで、複数のアプリケーション／ユーザ間で QoS 要求が衝突する場合は、それらを調停して送出制御または優先度付与を行なうことになるため、先に述べたボトルネックにおける QoS 制御と本質的に同じ制御が必要になる。

我々は、Last-one-Mile ボトルネックにおけるトラフィックの QoS 制御および diffserv 環境における優先度ビット付与を行なう、エッジトラフィックコントローラの開発を行なっている [1]。本論文では、エッジトラフィックコントローラに対してアプリケーションから QoS 指定を行なうためのモデル QoSPath について述べる。QoSPath によってアプリケーションは QoS の嗜好を反映させた QoS 効用関数を指定することができる。QoSPath モデルの特徴として、多様なアプリケーション QoS、ネットワーク QoS フォーマットに対応できること、ネットワーク制御のために必要な、アプリケーション QoS とシステム QoS の変換／逆変換が行ないやすいこと、モデルを利用したプログラミングの簡潔さと記述力を兼ね備えていることが挙げられる。現在、提案モデルを実装し、効用関数を利用した QoS 調停アルゴリズムの評価を行なっている。

QoS Specification Model for Adaptive QoS Control
Yasunori Matsui, Seiji Kihara, Satoshi Moriai, Hideyuki Tokuda
NTT, Keio University

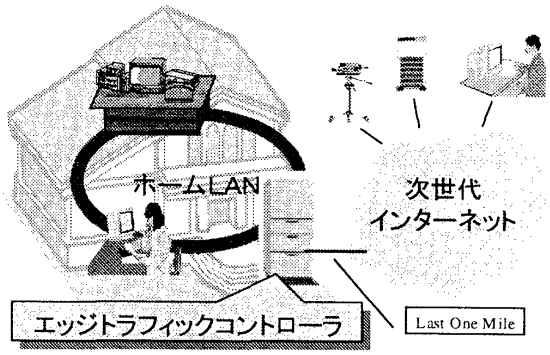

図 1: Last-one-Mile ボトルネックとエッジトラフィックコントローラ

2　QoS 指定に関する考察

動画など、QoS 制御が必要であるアプリケーションは、{ フレームサイズ、フレームレート、色深度、最大遅延 } といったアプリケーション依存の QoS 表現を持つ。このようなアプリケーション QoS はメディアや圧縮フォーマットによって異なってくる。これに対して、制御対象であるネットワークは { 最大ビットレート、キューの優先度、重み } といったネットワークとしての QoS 表現を持つ。ネットワーク QoS は利用するネットワークメディアやスケジューリングアルゴリズムによって異なる。

QoS 指定を行なうためには、アプリケーションプログラマは QoS をアプリケーション QoS によって指定するようにし、システムが提供するライブラリによってネットワーク QoS に変換できることが望ましい。これによって、アプリケーションにおいては自然な表現での QoS 指定が可能になり、さらに利用するシステムに独立に記述できるためアプリケーションのプラットフォーム依存性が減少する利点が生じる。

ところで、適応的な QoS 制御システムを考えた場合、アプリケーションは実現を希望する単一の QoS 値（例えば {640x480 pixels, 30fps}）をネットワークに対して指定するだけでは十分ではない。ネットワークの状況によっては指定 QoS を完全には満たせないかも知れないが、若干落ちる程度の品質が（例えば先の指定に対して {320x240, 10fps}）実現できたとき、その品質をユーザーが利用することを望む可能性があるためである。適応型アプリケーションにとってはこのような QoS 指定が可能にならなければならない。

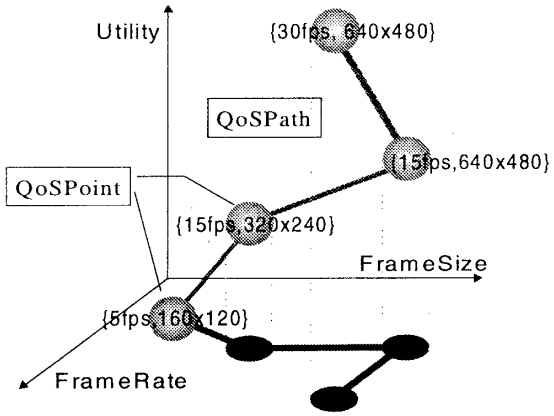

図 2: QoSPath の概念図

このような中間品質に対してはアプリケーションの利用形態などによってどの QoS 値を優先するかの嗜好が発生する。すなわち、フレームレートを優先したり、フレームサイズを優先したり、といった指定ができることが望ましい。このような嗜好の指定は、効用関数によって行なうことができる [4][2]。しかし、効用関数は一般に多次元空間の QoS 等高面として定義されるため、そのままではユーザーが指定するのは難しく、また効用関数を利用した調停を記述し、実行することは困難である。

3 QoSPath モデル

以上で述べた問題を考慮しながら QoS 指定を行なうために QoSPath モデルを提案する（図 2）。概念的には、QoSPath は一般に多次元等高面で表される QoS の効用関数の稜線を線形近似したものである。QoSPath はいくつかの代表的な QoS 値である QoSPoint と、その間を線形近似する QoSEdge によって構成される。例えば図 2では、最高品質の QoS を {30fps, 640x480}、最低 QoS を {5fps, 160x120} として、さらに 2 つの中間品質を指定している。QoSPath によって資源が不足し要求が完全には満足できない場合の品質低下の方法が表現されている。

アプリケーションやシステムが利用する QoS フォーマットの多様性は QoSPoint の型を変更することで対応する。すなわち、`MPEGQoSPoint` や `CBQQoSPoint` といったものを用意することで、さまざまなアプリケーションから統一的なモデルで QoS を指定することを可能とし、またアプリケーション QoS からシステム QoS への変換および逆変換を実現しやすくしている。

QoSPath モデルの特徴として、単純な QoS 指定は少ない QoSPoint で（例 非適応型なら 1 個の QoSPoint、適応型なら最小 2 個から）表現でき、きめの細かい QoS 指定にも QoSPoint を増やすことで対応できることがある。これによって、プログラミングの簡潔性と記述力の両立をはかっている。

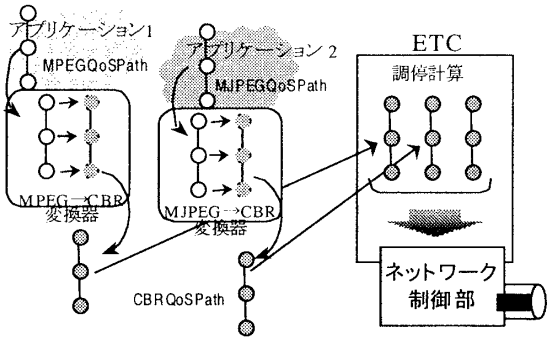

図 3: QoSPath を利用したエッジトラフィックコントローラ

4 エッジトラフィックコントローラへの実装

現在、提案した QoSPath モデルを用い、アプリケーションからの QoS 指定を変換／調停してネットワーク制御を行なうエッジトラフィックコントローラを実装中である (図 3)。各アプリケーションはそれぞれの利用データフォーマットに従った QoS （アプリケーション QoS） によって QoSPath を指定する。指定された QoSPath は、QoS 変換ライブラリによってネットワーク QoS の QoSPath に変換され、エッジトラフィックコントローラに送られる。コントローラでは、QoSPath を利用した QoS 調停計算を行ないネットワークを制御する。調停計算部はプラグイン可能とすることで、制御ポリシを変更することを可能としている。

実装は FreeBSD の上で行ない、QoS 変換／調停部分を Java 言語で記述し、ALT-Q [3] によって実現されている CBQ パケットスケジューラを制御するようにしている。調停計算として Rajikumar ら [4] が提案した効用関数の和を最大化するアルゴリズムや、Bianchi ら [2] による効用の公平化をはかるアルゴリズムなどを実装している。その上で適応型連続メディアアプリケーションを作成し、評価を行なっている。

参考文献

[1] 松井他 ： 「緩い QoS 保証を目的とした適応型トラフィックコントローラ」、日本ソフトウェア科学会第 15 回大会、1998.

[2] Bianchi *et al.* : "Supporting Utility-Fair Adaptive Services in Wireless Networks", In *Proceedings of 6th IFIP IWQoS*, 1998.

[3] Cho, K.: "A Framework for Alternate Queueing: Towards Traffic Management by PC-Unix Based Routers", In *Proceedings of 1998 USENIX Annual Technical Conference*, 1998.

[4] Rajkumar, R.*et al.* : "A Resource Allocation Model for QoS Management", In *Proceedings of IEEE Real Time System Symposium 1997*, 1997.