社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＱＡＩ－7　(7)
2003／5／21

ロバストなクラス別優先制御を実現するマルチパスルーティングアルゴリズム

大倉 昭人　五十嵐 健　川上 博　平田 昇一

NTT DoCoMo ネットワーク研究所 〒239-8536 神奈川県横須賀市光の丘 3-5

E-mail: {okura igarashi, kawakami, hirata}@netlab.nttdocomo.co.jp

あらまし　インターネット上で幅広く用いられている OSPF 等の既存のルーティングプロトコルにおいては最短経路のみを利用するため，クラス毎に優先制御を実現する DiffServ と共に用いた場合，トラヒックが集中したルータでは優先制御が有効に機能しない。また TOS ルーティング等の複数経路を利用するルーティングプロトコルでは DiffServ クラスと対応する経路の関係は静的に定まり、トラヒックの変動に合わせて経路を切り替える場合にはクラス毎に独立した計算が必要になる。本研究はトラヒックの状況に応じて必要な数だけ経路を作成し、DiffServ クラスと経路との対応を動的に変化させることでロバストなクラス別優先制御を実現する。

キーワード　QoS，DiffServ，マルチパスルーティング，線形計画法，シンプレックス法

Multipath Routing Algorithm for Robust Class-based Priority Queuing

Akihito OKURA　Ken IGARASHI　Hiroshi KAWAKAMI　and　Shoichi HIRATA

Network Laboratories, NTT DoCoMo　3-5 Hikarinooka, Yokohama, Kanagawa, 239-8536 Japan

E-mail: {okura igarashi, kawakami, hirata}@netlab.nttdocomo.co.jp

Abstract　An existing routing protocol such as OSPF widely used on the Internet forwards the packet by using only the shortest path. Therefore, DiffServ, class-based priority queuing, does not function effectively in the router on which traffic concentrates when OSPF is used. Moreover, if multipath routing protocol, such as TOS routing, are used, the correspondence of a DiffServ class to a routing path is decided statically. Therefore, when the paths are switched according to the change of traffic, an independent path calculation of each DiffServ class is not avoidable. In order to realize the robust Class-based priority queuing, this paper proposes two algorithms, one for optimal paths addition and another for the optimal correspondence of a DiffServ class to a routing path.

Keyword　QoS，DiffServ，Multi Path Routing，Liner Programming，Simplex Method

1. まえがき

次世代ネットワークのバックボーンとして ALL-IP ネットワークを構築する動きが広がっている。この次世代 IP ネットワークでは、帯域条件やリアルタイム/非リアルタイムといった様々な品質要求を持つトラヒックが１つのバックボーンを共有するため、トラヒックの品質要求に合ったサービス（QoS：Quality of Service）を提供する仕組みが必要である。

現在のインターネットはベストエフォート通信が主流であり、トラヒック毎の QoS 制御は行われていない。DiffServ[1]はこうした QoS ニーズの高まりを受け、クラス単位で優先制御を行う方式として誕生した。

パケットが DiffServ ネットワークに入ってくると、パケットの IP ヘッダには境界ルータでトラヒックに応じた DSCP（DiffServ Code Point）が書き込まれる。ネットワーク内部のルータには DSCP 毎にキューの割り当てやスケジューリングに関する PHB（Per Hop Behavior）が規定されている。

DiffServ は前述のようにクラスという粗い粒度で優先制御を行うためスケーラビリティに富む方式であるが、厳密な品質保証ではないため、優先度の高いトラヒックが集中した場合などトラヒック品質が劣化することが知られている。特に DiffServ は経路制御を既存のルーティングプロトコルに依存しているため、この問題が生じやすい。

OSPF[2]、RIP[3]といった既存のルーティングプロトコルは経路候補の中からコストやメトリックといった基準で最短になる経路情報のみを経路制御情報として利用する。そのため特定の経路に負荷が集中する一方、帯域に余裕のある経路も存在するなどネットワークリソースを有効に活用できる仕組みではない。またクラス毎に経路を切り替えるという概念はなく、全てのク

ラスのトラヒックが同じ経路に集中するため優先クラスの集中による DiffServ の問題が顕在化する。

このように DiffServ は既存の IP ルーティングとの組み合わせではその効果を十分発揮することができず、クラス毎の経路制御が求められる。

2. 従来研究

DiffServ と経路制御を組み合わせた手法として MPLS/DiffServ[4]が提案されている。MPLS ネットワークではパケットにラベルがつけられ、MPLS ルータはラベルに応じた経路にパケットを転送する。ラベルは Hop-by-Hop に更新され、このラベルによる経路を LSP（Label Switching Path）とよび、エンドーエンドの経路制御が可能である。MPLS/DiffServ は QoS に有力な仕組みとされているが、3 つの課題を抱えている。

課題 1 は IP ルーティングから MPLS へ経路制御を変更する際に生じる問題である。MPLS ルータで DiffServ を行うためには MPLS ヘッダフィールドで DSCP を識別しなくてはならない。このための手法として E-LSP と L-LSP という 2 方式がある。E-LSP は Shim ヘッダの Exp フィールドに DSCP のマッピングを行う。しかし DSCP が 6 ビットであるのに対し、Exp フィールドは 3 ビットであるため完全な対応はできない。また ECN（Explicit Congestion Notification）[5]も考慮すると利用可能フィールドはさらに減少する。L-LSP はラベルをそのまま PHB と対応付けするため DSCP の数に関する問題は生じない。しかし LSP のセットアップ時にラベルと PHB のマッピング情報など特別なシグナリングが必要になる。

課題 2 は、MPLS の LSP 構築は IP ルーティングの経路表作成に比べ処理が複雑という点である。一般に LSP はルーティングテーブルに従い LDP（Label Distribution Protocol）を用いて構築される。帯域や遅延に関する品質要求があるトラヒックに関しては、送信ノードが OSPF などリンクステートプロトコルを利用してネットワーク全体の情報を集め、CSPF（Constraint Shortest Path First）といった制約ベースルーティングで経路を求める。そしてその経路に沿って RSVP(Resource ReSerVation Protocol)[6] や CR-LDP（Constraint-based Routing LDP）[7]の明示的ルート指定オプションを利用して LSP を確立する。この処理は経路の計算と LSP 確立のためのシグナリングが独立している。ルーティングプロトコルは経路の計算と経路表作成が連動していることに比べると処理が冗長である。

課題 3 は、クラス別の経路制御手法に対し、必要な経路の本数及びどのクラスをどの経路に流すかという対応付けである。MPLS/DiffServ ではこの課題は扱われていない。課題 2 で述べたように MPLS では QoS 要求トラヒックに対し送信ノードが制約ベースルーティングにより経路を決定する。しかしこの計算は送信ノード毎に独立しているため経路の最適性が保証されない。例えば優先度の低いトラヒックが最短パスに集中している状態で、優先度の高いトラヒックを流すことを考える。その状態のネットワークに対して CSPF を実行すると、優先度の高いトラヒックに対して迂回経路が割り当てられてしまう。また予め LSP を複数設定しておき、優先度の高いトラヒックは最短経路、優先度の低いトラヒックは迂回経路という設定を行っていれば上記問題は防げる。しかし帯域に余裕がある場合でも迂回経路を通るトラヒックが出るなど通信品質の劣化を招くと共に、必要以上にネットワークリソースを消費する。

3. 研究目的

本研究では上記課題を解決する QoS システムの構築を目的とする。まず課題 1・課題 2 から IP レイヤでパケットを転送することとし、新たな IP マルチパスルーティングプロトコルを提案する。また課題 3 から DiffServ クラスとマルチパスの最適対応アルゴリズムを考案する。

本章以降の構成を以下に示す。第 4 章ではマルチパスルーティングプロトコルの概要と既存の OSPF に対する拡張について述べる。第 5 章ではマルチパスルーティングと DiffServ を組み合わせるための IP ヘッダ定義や、システムアーキテクチャについて説明する。第 6 章ではネットワーク問題に対応可能な最適化手法として線形計画法について説明し、第 7 章で線形計画法の解法であるシンプレックス法の特性を利用した最適な経路群生成及び経路とトラヒックの最適対応アルゴリズムについて説明する。第 8 章ではシミュレーション結果を示し、第 9 章で結論を述べる。

4. マルチパスルーティングプロトコル

IP ルーティングは大きく分けてディスタンスベクタ型とリンクステート型の 2 つがあるが、効果的なマルチパスを作成するため各ルータがネットワークの状態を保持するリンクステート型を採用する。リンクステート型の代表的ルーティングプロトコルは OSPF が知られている。OSPF は各ルータがリンクステートデータベースを構築し、このデータベースから経路コストが最小になる経路をダイクストラのアルゴリズムにより求める。

OSPF でのリンクコストは全てのトラヒックに対して均一であるが、マルチパスルーティングを行う場合、最短経路、2 番目の経路といったルーティングクラス

毎にリンクのコストが異なる方が扱いやすい。そこでOSPF に対し、ルーティングクラス毎に複数コストの設定を可能にする機能拡張を加える。

OSPF は RFC1583[8]において TOS ルーティングをサポートしていた。TOS ルーティングではリンクに IPv4 ヘッダの TOS フィールド毎にコストを設定し、送信先アドレスと TOS の 4 ビットに応じた経路制御を行う。TOS ルーティングは一般に普及せず、現行の OSPF で削除されてしまったが、OSPF メッセージヘッダには互換性のため TOS ルーティング用に 8 ビットのフィールドが残されている。そのためこのフィールドを再利用して複数コストに対応したメッセージを作成する。TOS に変わる識別子として 8 ビットの RCP（Routing Code Point）を定義する。そして RCP 毎にデータベースを作成し、それぞれダイクストラのアルゴリズムを適用することで目的地と RCP をキーにしたマルチパスルーティングテーブル（図 1）を作成する。

Destination	NextHop
R10	R2
R11	R3
R12	R4

⇨

Destination	RCP	NextHop
R10	Default	R2
R10	00000001	R3
R11	Default	R3
R11	00000010	R2
R12	Default	R4

図 1　マルチパスルーティングテーブル

5. システムアーキテクチャ

5.1. IP ヘッダ

前章で定義した RCP と、DiffServ における DSCP を IP ヘッダにマッピングする。IP は IPv6 を利用する。まず DSCP は RFC3168[5]に従い Traffic Class フィールドの先頭 6 ビットに設定する。そして RCP は Flow Label フィールドの先頭 8 ビットに設定する。図 2に構成を示す。このように定義すると DSCP と RCP がお互いに干渉しないため、DSCP と RCP の対応を自由に変更することができる。

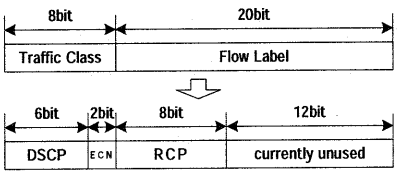

図 2　IP ヘッダフィールド定義

5.2. QoS Manager

DSCP と RCP を動的に組み合わせて最適な QoS を実現するためにはネットワークを集中管理的にコントロールするノードが必要になる。このノードを QoS Manager とする。QoS Manager は他のルータと同様に OSPF のメッセージからリンクステートデータベースを作成し全ての送受信ノード間の経路情報を持つ。またネットワーク内の各ルータからどの種類のトラヒックがどのノードにどれだけ流れるのかといったトラヒックの監視を行う。

QoS Manager はこの経路とトラヒックの情報から、6 章・7 章で説明する線形計画法に基づきマルチパスと DiffServ クラスの最適な対応を求める。この対応はデータベース化され、ネットワーク内のルータに通知される。

ネットワーク内のルータはこのデータベースに基づき QoS 制御を行う。この QoS 制御は DiffServ と同様にエッジルータ、コアルータで仕組みが異なる。エッジルータは外部からの流入トラヒックに対し、データベースに応じて IP ヘッダの DSCP、RCP フィールドの操作を行う。コアルータはこの DSCP、RCP フィールドに応じた優先制御やルーティングを行う。図 3にシステムアーキテクチャ概略を示す。

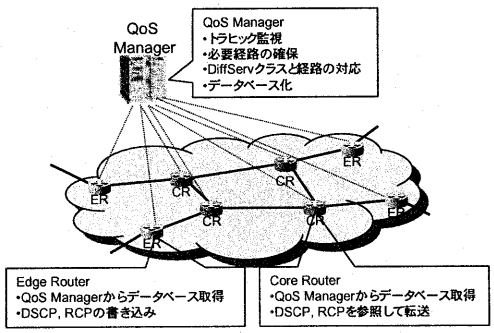

図 3　アーキテクチャ図

6. 線形計画法とネットワーク

6.1. 線形計画法

QoS Manager が最適な対応を決めるアルゴリズムには線形計画法を改良したアルゴリズムを利用する。まず線形計画法の概要を述べる。

最小化または最大化すべき目的関数が線形式で、しかも複数の等号式または等号・不等号式であらわされる制約条件式の全てが線形式からなる場合を線形計画問題という。具体的には N 個の独立変数についての目的関数

$$f = a_{01}x_1 + a_{02}x_2 + \cdots + a_{0N}x_N \qquad (1)$$

を最大化（最小化）する。ただし各変数は主条件

$$x_1 \geq 0, x_2 \geq 0, \cdots, x_N \geq 0 \qquad (2)$$

を満たし、さらに 3 つの形式で表される制約条件を満たす。

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{iN}x_N \leq b_i (b_i \geq 0) \qquad (3)$$

$$a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jN}x_N \geq b_j \geq 0 \qquad (4)$$

$$a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kN}x_N = b_k \geq 0 \qquad (5)$$

これらの a_{ij} は正でも負でも 0 でも良い。制約条件を満たす値の集合を可能ベクトル、目的関数を最大化(最小化)する可能ベクトルを最適可能ベクトルという。

6.2. ネットワーク流への適用

ネットワークを実際に線形計画問題にする際の変数、制約条件、目的関数の設定法について述べる。

まず変数には各経路のトラヒック量を採用する。つまり初期の変数は OSPF で求まる最短経路のトラヒック量である。トラヒックの状況によって新規経路が追加されるとその経路の分だけ変数も増加する。制約条件には(3)(4)(5)の 3 つの種類が存在するが、(3)はリンクの帯域に関する制約を表す。つまりあるリンクを通る変数(経路のトラヒック量)を合計すると、リンクの帯域以下という制約である。(5)は経路に流れるトラヒック量に関する制約を表す。この制約は送受信ノード間に複数の経路が存在する場合、その経路の変数を合計すれば送受信ノード間のトラヒック量に等しくなることを意味する。この 2 つは必須条件である。最後に(4)は明示的な品質保証に用いる。ある特定経路の流量に下限を設けることで、優先度の高いトラヒック分流量を確保することができる。

目的関数はネットワークリソース消費を最小にするように設定する。リンク l が単位トラヒックを流す際に消費するネットワークリソースを c_l とすると、経路 p_i の消費するネットワークリソースは(6)となる。

$$C_{P_i} = \sum_{l \in P_i} C_l \tag{6}$$

経路 p_i を流れるトラヒックを x_i とすると、目的関数は(7)と設定できる。

$$f = \sum_{i \le n}^{i=1} C_{P_i} x_i \tag{7}$$

この目的関数を最小化することで制約条件を満たす中でネットワークリソース消費を最小化する最適な経路流量を求めることができる。

7. 最適対応アルゴリズム

前章に従い目的関数、制約条件を設定して線形計画法を解けば各経路の最適トラヒック量を求めることができる。しかしこれは経路が十分に用意されている場合であり、経路が不足する場合は新規の経路を追加しなければならない。従来の線形計画法は予め与えられている目的関数・制約条件に対し最適化を行うのみであるため、経路が不足することがわかってもそこから経路の追加を行うアルゴリズムは存在しない。予め経路の候補を複数用意して新規経路の追加を押さえることも考えられるが、変数の増加につながるので計算負荷が増大する。またトラヒックの変動によりボトルネックも変化するため、予め効率的な経路を用意することは困難である。

そこで本章では、線形計画法の解法であるシンプレックス法の特性を利用して、線形計画法と連動した経路追加アルゴリズムを提案する。さらに各経路の流量と DiffServ クラスの優先度を基にした DiffServ クラスと経路の対応アルゴリズムも説明する。

7.1. 経路追加アルゴリズム

現在の経路がトラヒックに対して十分な帯域を提供していない場合、かならずボトルネックリンクが存在する。このボトルネックリンクを迂回するように経路を追加すれば必要箇所に必要な数だけ経路を追加しているとみなせる。そのためまずシンプレックス法の解導出の概念を説明し、その特性からボトルネックリンクを発見する仕組みを説明する。

7.1.1. シンプレックス法における解導出の概念

シンプレックス法では、まず N 次元空間全体を始点とし、制約条件に合わない空間を削っていく。全ての制約条件について空間処理を行った後に残る領域を可能領域と呼ぶ。この可能領域が存在しない場合、制約条件を全て満たす可能ベクトルは存在しない。また可能領域が存在する場合、最適可能ベクトルは可能領域の境界面に存在する。なぜなら目的関数が線形であるため勾配は 0 ではなく、この勾配に沿って可能領域を進むといつか境界の壁に当たるためである。シンプレックス法はこの特性を利用して可能領域の境界に沿って最適可能ベクトルを探していく。

7.1.2. シンプレックス法における計算

シンプレックス法の実際の計算は、可能ベクトルが存在するかどうかの判定を行う step1 と最適可能ベクトルを求める step2 の 2 つに別れる。まず線形計画問題を標準系に変換するが、詳しい説明は文献[9]に譲りここでは概略を述べる。

線形計画問題の標準系とは(3)(4)の不等号の制約条件を持たず、(5)のような等式条件と(2)の非負条件だけからなる問題を指す語である。(3)(4)にスラック変数と呼ばれる非負変数 x_{N+i} を加え等式に変形し、(3)(4)(5)を移項および人工変数 z_i を用いて書き直すと(8)のようになる。なお c_{iN+i} は ±1,0 のいずれかを表す。

$$z_i = b_i - a_{i1}x_1 - \cdots - a_{iN}x_N - c_{iN+i}x_{N+i} \tag{8}$$

この(8)と(3)(4)(5)は同じ問題ではない。これらが同じ問題になるのは全ての z_i が 0 のときだけである。ここで制約条件の数を M として補助目的関数を式(9)のように設定する。

$$f' = -\sum_{i=1}^{M} Z_i \tag{9}$$

step1 ではまずこの補助目的関数と制約条件にシンプレックス法を適用する。人口変数 z_i は非負であるから、補助目的関数が最大になるのは全ての z_i が 0 のときである。この step1 は可能ベクトルの初期値を作り出す処理に相当する。ここで 0 にならない z_i が存在すると、そのとき可能基底ベクトルは存在しない（可能領域が存在しない）ことがわかる。z_i が 0 になれば step2 へ、0 にならなければ解なしで終了する。

step2 では step1 で得られた解を初期の可能ベクトルとし、目的関数についてシンプレックス法を解くことで最適可能ベクトルを得ることができる。

7.1.3. ボトルネックリンク発見原理

前項で述べたようにシンプレックス法では step1 で補助目的関数を利用して可能領域の存在を確かめる。補助目的関数についてシンプレックス法を解くとは、7.1.1で述べたように補助目的関数がある制約条件による超平面にぶつかった後、補助目的関数をその超平面上に射影した勾配に沿って進み、新しい制約条件による超平面にぶつかると、再度射影を行い勾配に沿って進むことを繰り返す。そしてある頂点にたどり着きその先に領域が存在しないにもかかわらず補助目的関数が 0 にならない場合、可能領域が存在しないと判定される。

この時の頂点座標を仮の可能ベクトルとすると、仮の可能ベクトルはその頂点に至るまでに移動した超平面、つまり制約条件を満たす連立方程式の解である。ネットワークの制約条件は6.2に示すようにリンク帯域とトラヒック量に関する不等号式であり、リンク帯域の条件により変数（流量）の上限が定められる。つまりこの仮の可能ベクトルとは、リンク帯域条件の超平面で補助目的関数の増加が停止した状態を表す。その先にトラヒック量に関する制約の超平面があるにもかかわらず、リンクの帯域に対する制約条件とお互いに疎であるため可能領域が存在しないのである。

このことから補助目的関数が 0 にならず停止する状態はトラヒックを現在の経路の限界まで流しているがまだ流すべきトラヒックが存在することを表すことがわかり、仮の可能ベクトルは現在の経路に対して限界までトラヒックを流した場合の各経路の流量を表すことがわかる。この各経路の流量と各リンクの帯域を参照することでボトルネックとなるリンクが発見される。

7.1.4. 経路の追加

ボトルネックリンクが発見されると、そのリンクコストを上昇させる。ダイクストラのアルゴリズムは最小コスト経路を求めるため、必然的にボトルネックリンクを迂回する経路が生成される。この新たに求まった経路を経路群変数に加え、再度シンプレックス法による最適化を行う。このシンプレックス法による最適化と経路の追加処理を再帰的に行うことでネットワークの帯域が存在する限り経路が加わるため、ネットワークの限界までトラヒックを流すことができる。ボトルネックリンクを迂回するように経路を計算しても新規経路がもう存在しない場合、ネットワークにはもうトラヒックを流せる経路が存在しないことを表す。

7.2. DiffServ クラスと経路の対応アルゴリズム

シンプレックス法の結果得られた経路流量に対して、優先度の高い DiffServ クラスから順に良い経路に割り当てる。なお(4)式で記述したように、あるクラスを特定経路で流すため制約がある場合はそれを優先する。

8. シミュレーション

NS2(2.1b9)[10]に改良を加え、2 つのシミュレーションを行った。まず経路制御に既存の OSPF を利用した OSPF/DiffServ と提案手法とのスループットの比較を行う。

次に経路制御に MPLS を利用した MPLS/DiffServ と提案手法の遅延時間の比較を行う。MPLS/DiffServ としては CSPF により動的に LSP を設定する MD-CSPF と、予めクラス毎に LSP を設定しておく MD-Static の 2 方式を扱う。

8.1. シミュレーション詳細

シミュレーションに利用したトポロジーを図 4に示す。数字は各リンクの帯域を表し、各リンクの遅延は 5ms とする。

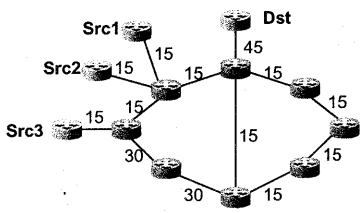

図 4　シミュレーショントポロジー

トラヒックには ClassA、ClassB、ClassC の 3 つのクラスを設け、表 1のように品質要求を設定した。

表 1　トラヒッククラス

	ClassA	ClassB	ClassC
Throughput	5Mbps	5Mbps	Best Effort
Delay	50ms	-	
Priority	High	Middle	Low

キューイングには RED（Random Early Detection）を組み込んだ WFQ（Weighted Fair Queuing）を利用した。3 つのキュー（Queue1、Queue2、Queue3）を設定し、PHB の設定により ClassA は Queue1、ClassB は Queue2、ClassC は Queue3 にそれぞれマッピングされ、

Queue1:Queue2:Queue3 は 10:5:1 でスケジューリングされる。各キューのバッファ量は 10 パケットである。

Src1、Src2、Src3 とも ClassA、ClassB、ClassC それぞれの 5MbpsUDP トラヒックを Dst に向けて送信する。シミュレーション開始時にトラヒックは無く、1 秒後に Src1 が送信開始、2 秒後に Src3 が送信開始、3 秒後に Src2 が送信開始、4 秒後に Src2 が送信終了、5 秒後に Src3 が送信終了、6 秒後に Src1 が送信終了とした。

ネットワーク内のルータはトラヒック変動があると QoS Manager に内容を通知し、そのタイミングで経路の追加および DiffServ クラスと経路の対応アルゴリズムが実行される。

8.2. OSPF/DiffServ との比較

まず OSPF/DiffServ と提案手法のスループットの比較を行った。OSPF/DiffServ の結果を図 5、提案手法の結果を図 6に示す。

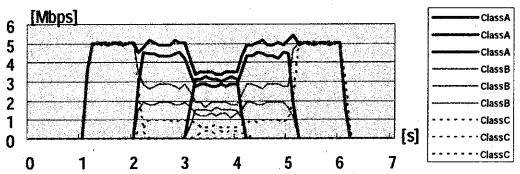

図 5 OSPF/DiffServ のスループット

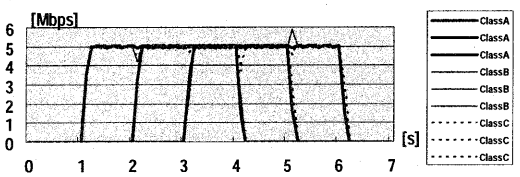

図 6 提案手法のスループット

OSPF/DiffServ ではトラヒックが最短経路に集中するためパケットロスが生じ、スループットが低下している。また DiffServ を利用しても優先トラヒックの QoS は実現されていない。提案手法はトラヒックの状況によりクラスと経路の対応が切り替わるためパケットロスは生じずスループットは維持される。

8.3. MPLS/DiffServ との比較

次に MPLS/DiffServ と提案手法の平均トラヒック遅延の比較を行った。なお MPLS/DiffServ ではクラス毎に経路を切り替えるためパケットのロスは生じず、スループットは図 6に近い結果となる。MD-CSPF は通信開始時にネットワーク状況を収集し帯域要求についてトポロジフィルタリングを行った後に最短経路を求め、経路上に LSP を設定してパケット転送を行う。MD-Static は各 Src-Dst 間で 3 本の LSP を持っておき、ClassA は最短経路、ClassB は 2 番目、ClassC は 3 番目の経路のラベルが付加される。

提案手法、MD-CSPF、MD-Static 各手法のクラス別平均トラヒック遅延を表 2に示す。

表 2 平均トラヒック遅延 [ms]

	Proposal			MPLS/DiffServ-CSPF			MPLS/DiffServ-Static		
	ClassA	ClassB	ClassC	ClassA	ClassB	ClassC	ClassA	ClassB	ClassC
Src1	30.4	40.9	53.0	30.1	30.9	31.7	30.3	47.1	71.5
Src2	31.4	47.5	72.3	69.5	70.3	71.1	31.4	47.4	72.1
Src3	36.8	42.4	54.8	41.3	42.2	42.9	36.8	42.0	66.4
Ave.	32.7	42.1	55.7	38.2	39.0	39.8	32.6	45.4	69.9

MD-CSPF は帯域が空いている経路から順にトラヒックが流れる。そのため最後に通信を開始する Src2 から流れるトラヒックは優先度に関係なく迂回経路を流れるため遅延時間が増大している。ClassA の遅延要求も満たしていない。MD-Static はクラス毎に最短パス・迂回経路と対応が決まっているため優先度に応じた遅延になっている。しかし提案手法と比較すると、ClassB・ClassC トラヒックの遅延が多い。これは MD-Static は最短経路に余裕があるときでも ClassB・ClassC トラヒックは迂回経路を通るのに対し、提案手法では最短経路に余裕があるときは ClassB・ClassC トラヒックも最短経路に対応付けされるためである。提案手法は優先制御を実現しつつ、MPLS/DiffServ に比べネットワーク全体の遅延時間を縮小している。

9. 結論

本論文では DiffServ の優先制御と IP マルチパスルーティングを組み合わせ、線形計画法に基づく最適 QoS システムを提案し、シミュレーションにより有効性を示した。今後はより実用的なトラヒックモデルを用いたシミュレーションから適用領域を明確化すると共に、実装に向け QoS Manager とルータ間通信プロトコルなど詳細を検討して行く必要がある。

文 献

[1] S. Blake, et al "An Architecture for Differentiated Services," RFC2475, December, 1998.

[2] J. Moy, "OSPF Version 2," RFC2328, April, 1998.

[3] G. Malkin, "RIP Version 2," RFC2453, November, 1998

[4] F. Le Faucheur, et al "MPLS Support of Differentiated Services," RFC3270, May, 2002.

[5] K. Ramakrishnan, et al "The Addition of Explicit Congestion Notification (ECN) to IP," RFC3168, September, 2001.

[6] R. Braden, et al, "Resource ReSerVation Protocol," RFC2205, September, 1997.

[7] B. Jamoussi, et al, "Constraint-Based LSP Setup using LDP," RFC3212, January, 2002.

[8] J. Moy, "OSPF Version 2," RFC1583, March, 1994.

[9] W. H. Press, et al "Numerical Recipes in C++," Cambridge univ. press, 2002.

[10] The Network Simulator NS-2, http://www.isi.edu/nsnam/ns/