2004 年度 修士論文

輻輳制御機能を付加した DiffServ スケジューラの構成法

提出日： 2005 年 2 月 2 日

指導：後藤滋樹教授

早稲田大学 大学院理工学研究科 情報・ネットワーク専攻
学籍番号： 3603U071-6

佐々木洋美

目次

図一覧

表一覧

第 1 章

序論

1.1 研究の背景

1.1.1 インターネットの現状と QoS

インターネットの爆発的な普及により、音声や動画のような帯域や遅延の保証を必要とするリアルタイム性の IP アプリケーションが数多く使用されるようになってきた。しかし、現行のインターネットの基本特性は、指定された 2 点間で最善努力型 (best effort) の品質の伝送サービスを提供することである。したがって、リアルタイム通信において実用的な品質を得るには適切な QoS (Quality of Service) 保証が必要不可欠となっている。そのような状況の中で、インターネット上を流れる様々なデータの種類に応じたサービスを提供するための研究開発が盛んに進められている。

1.1.2 QoS 技術

インターネット上でリアルタイム性のアプリケーション (IP 電話、映像配信ソフト) を使用する場合、通信相手のデータが届くまでの遅延が大きかったり、途中で破棄されたり、届く間隔にばらつきがあったり、いろいろな品質劣化が生じる可能性がある。 QoS とは、アプリケーションが快適に通信できるように帯域を確保し、一定した伝送速度を提供すること、あるいは重要なアプリケーションを優先させるためにトラフィックに優先順位を付ける機能をネットワーク上で保証することをいう。 IP 層で QoS を実現するための技術の基本的な原理は、ネットワーク内の各ゲートウェイ上で、到着する多数のパケットの入出力順序を制御することである。中でも、IETF (Internet Engineering Task Force) において標準化されている DiffServ (Differentiated Services) は、スケーラビリティに優れており、比較的簡単に QoS を実現できる技術として注目を集めている。

1.1.3 DiffServ による QoS

DiffServ とは、同一の QoS を必要とする複数のフローのパケットに同一の DSCP (Differentiated Services Codepoint) を付加することによって 1 つの集合 (aggregate) とし、 aggregate ごとにクラスを設定して優先度の差別化を行う技術である。クラスごとの処理は PHP (Per Hop Behavior) と呼ばれ、 EF (Expedited Forwarding) クラス、 AF (Assured Forwarding) クラス、 BE (Best Effort) クラスのいずれかにフローを集約する。 PHP ごとに優先制御を提供するために、 DiffServ 網を構成する各ゲートウェイ上で動作する様々なパケットスケジューリング方式やキューバッファ管理方式が考案されている。

1.2 研究の目的

本研究では、 DiffServ 上で動作するパケットスケジューリング方式の 1 つである DRR (Deficit Round Robin) を拡張することを目的とする。 DRR では、トラフィックが量的にあるクラスに偏っている場合に、トラフィック量の多いクラスの遅延が増大してサービスの順番が逆転してしまうことがある。そこで本研究では DRR に輻輳制御機能を付加し、リアルタイム性のアプリケーションを DiffServ の EF クラスと AF クラスで使用する。さらにシミュレーションによってその有効性を評価する。

第 2 章

DiffServ

2.1 DiffServ の概要

DiffServ (Differentiated Services) [1] は、内部ゲートウェイがパケットヘッダの特定フィールドだけを見て処理内容を決定し、サービスできるようにする QoS 技術である。性能と拡張性を設計の最重点に置いており、その定義は機能ではなく実現方法を指定して行われる。ここでは、まず基本的な原理を説明する。

2.1.1 DSCP

IPv4 のヘッダの TOS (Type of Service) と呼ばれるフィールド (8 ビット) 、または IPv6 パケットの Traffic Class フィールド (8 ビット) の上位 6 ビットに DSCP (Differentiated Services codepoint) を記述しておき、ゲートウェイのトラフィック制御機構の動作をこのコードだけで決定する (図 2.1) 。通常、特定の QoS 処理が必要なフローに属するパケットには同一の DSCP を与える。逆に、同じ DSCP を持つパケットは、フローにかかわらず同一のサービスを受ける。すなわち、 DiffServ のサービス単位は必ずしもフローではなく、同一の DSCP を持つフローの

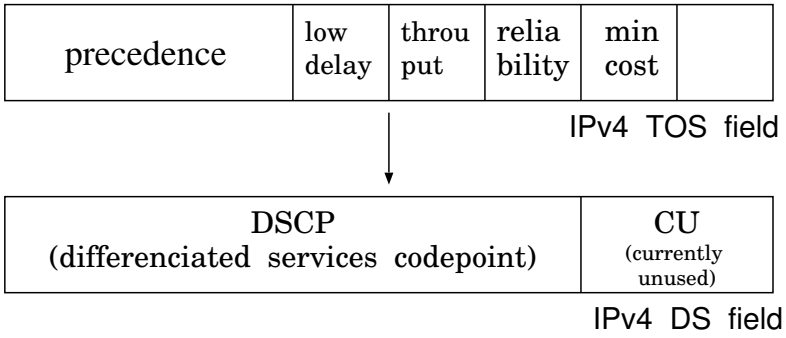

図 2.1: IPv4 の DSCP

集合 (aggregate) である。

2.1.2 DS ドメイン

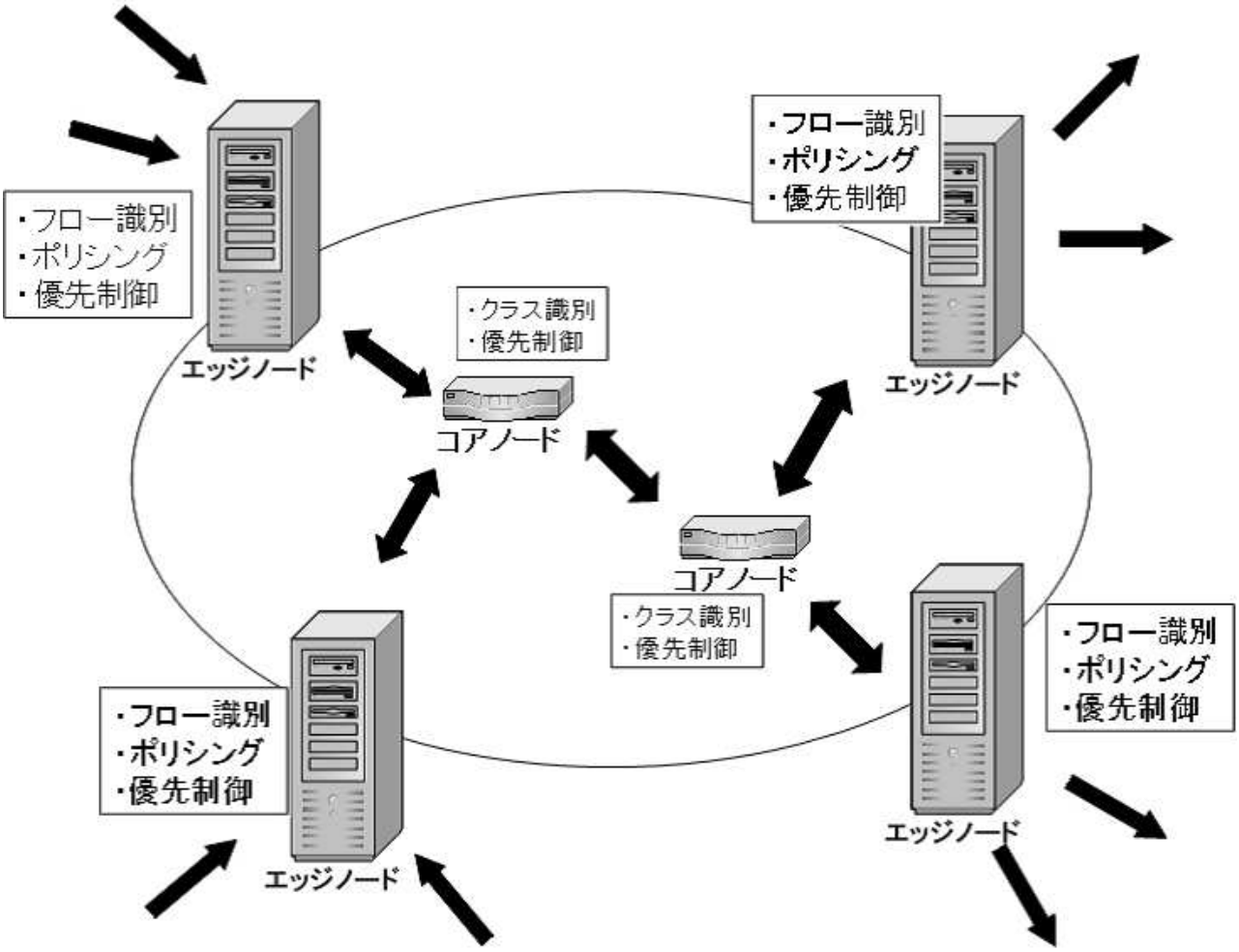

図 2.2: DS ドメイン

同一のサービス方針で定義された DiffServ のノード (以下、 DS ノード) からなるネットワーク領域のことを DS ドメインという。 DiffServ ではエンドツーエンドのサービスを規定することはせずに、 QoS を実現するフレームワークとその構成部分の一部を規定する。これは性能向上と拡張性を考慮に入れているからである。原則としてトラフィック調整は DS ドメインの入り口と出口だけで行う。同一 DS ドメイン内に存在するノード同士が通信する場合には、一貫した QoS が提供できる。異なる DS ドメインに存在するノード同士が通信する場合には、各々のドメインが持つポリシーをドメイン間の契約によって連携させなければならない。その契約を一般に SLA (Service Level Agreement) という。 DS ドメインは、エッジノード、コアノードと呼ばれる DS ノードから構成される。 (図 2.2)

- エッジノード
  DS ドメインの境界にある DS ノードである。主に、送信元アドレスやポート番号からサービスの種別を識別し、顧客と ISP の間の SLA に基づいて DSCP のマーキング、トラフィックのポリシング、優先制御を行う。

- コアノード
  DS ドメイン内に存在する DS ノードである。 DSCP に従って優先制御を行う。

2.1.3 PHB

PHB (Per Hop Behavior) とは、 DS ドメインに到着したパケットの振る舞いのことである。DiffServ では、 DS ドメインに流入してきたパケットを、 DSCP の値に従ってクラス分けすることによってトラフィックの優先制御を行う。 DS ノードは、 DSCP と PHB を対応付けるテーブルを保持しており、入力パケットの DSCP とテーブルを照らし合わせて対応する PHB を選択する。これは 32 ビットの符号なし整数で表され、その数値に対してバッファ容量や帯域幅、優先順位、さらに遅延やパケット破棄の許容量に関する規則が記述されている。それぞれの DS ドメインが DSCP を定義し、それに応じた動作をルータで設定・実現させる。これは DS ドメインごとに独立して決めることができるので、独自のサービスを提供することができる。以下に、 DiffServ で規定されている主な PHB を示す。

Default PHB

Default PHB とは、 DS ノードが受信したパケットが持つ DSCP に対して、適切な PHB を割り当てることができないときに選択される PHB である。従って最も優先度が低い Best Effort として扱われる。

AF PHB

AF PHB (Assured Forwarding PHB) [3] とは、特性の異なるトラフィックに対してそれぞれの特性に合わせたクラスを提供し、それらのパケットの転送を保証する。 AF PHB には、パケットバッファや帯域幅などのリソースを提供する 4 つのクラスと、パケット破棄優先度を示す 3 つのレベルがある。 DS ノードにおいて輻輳などが発生した場合は、破棄優先度が高いパケットから優先的に破棄される。標準では 4 つのクラスと 3 つのレベルを組み合わせることにより、それぞれのトラフィックに合わせた 12 種類のサービスが提供可能である。

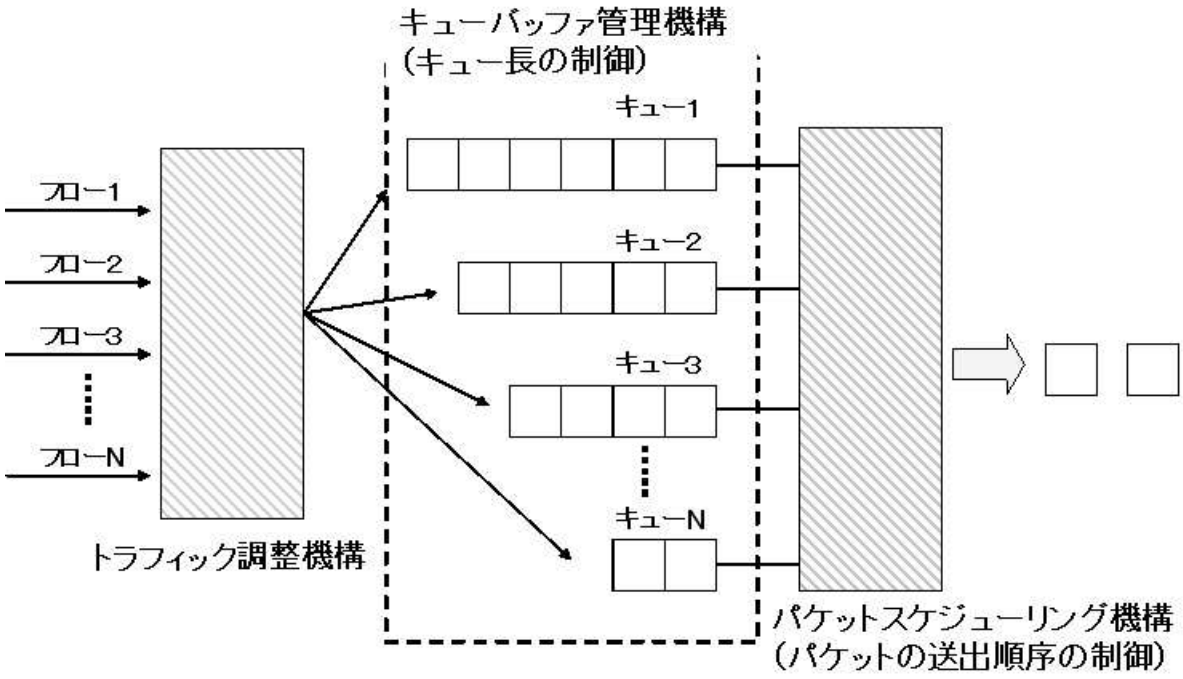

図 2.3: DiffServ における優先制御機構

EF PHB

EF PHB (Expedited Forwarding PHB) [2] は、他のどのトラフィックよりも優先的に処理することを指定する PHB である。 EF PHB では、すべてのトラフィックを設定された最低送出レート以下のレートにする厳格なポリシングが必要がある。これは、インターネット上に仮想専用線が構築されることを意味する。また、 EF PHB はプレミアサービスとも言われる。

2.2 DiffServ における優先制御

DiffServ における優先制御機構は、トラフィック調整機構、キューバッファ管理機構、パケットスケジューリング機構から成る。 (図 2.3)

2.2.1 トラフィック調整機構

トラフィック調整機構はエッジノードに備わっており、トラフィック分類機構が処理した結果を元に SLS (SLA の一部) の方針に適合するようにトラフィックを調節する。ポリシングともいう。トラフィック調整機構は Meter、 Marker、 Shaper、 Dropper の組み合わせから構成される。

Meter

Meter は、 TCA (Traffic Conditioning Agreement) (SLA の一部) に規定されたトラフィックの時間的特性を計測し、 TCA に適合するかどうかを検査する。代表的な Meter に、 TSW3CM (Time Sliding Window Three Color) [5] がある。パケットは TSW3CM によって Green、 Yellow、 Red に相当する 3 種類の DSCP にマークされるようになり、 AF PHB の破棄制御に使用される。

Maker

Meter から渡された結果を元に、適合パケットに所定の DSCP を記入する。

Shaper

TCA 非適合パケットに、所定のトラフィック特性に適合するまで遅延を与える。

Dropper

TCA の記述に基づきパケットを破棄する。非適合パケットを破棄することによって TCA に強制的に適合させる。

2.2.2 キューバッファ管理機構

到着パケットをキューに収容する際に動作する機構である。パケットを破棄することでキューの長さを制御して QoS 制御、輻輳防止をはかるとともに、送信元に輻輳を通知する役割を果たす。到着パケットをキューに収容するか破棄するかの判断、既にキューにあるパケットを破棄するかどうかの判断を行う。代表的なキューバッファ管理方式に、 RED (Random Early Detection) [6] 、 RIO (RED IN/OUT) [7] がある。

RED

RED とは、平均のキューサイズに応じて、確率的にパケットを破棄する方式である。確率的にパケットを破棄するので、バッファを多く占有しているものほど、破棄される確率が高くなる。閾値として平均キュー長を使用することにより、混雑を早期に発見することができる。図 2.4 において、 min thresh は平均キュー長の下限値、 max thresh は平均キュー長の上限値である。markdrop は破棄確率である。次に、平均キュー長、パケット破棄確率の計算法を示す。

1. 第 n パケットが到着したときの平均キュー長 avg_n

- キューが空でないとき

$$avg_n = (1 - w_q)avg_{n-1} + w_q \times qlength \tag{2.1}$$

(w_q は avg 推定用ローパスフィルタの重み係数)

- キューが空のとき

$$avg_n = (1 - w_q)^m avg_{n-1} \tag{2.2}$$

(m はキューが最後に空になってから現在までの経過時間)

2. 破棄確率 p_a

$$p_b = max_p(avg - min_{th})/(max_{th} - min_{th}) \qquad (max_p \text{ は } p_b \text{ の最大値}) \tag{2.3}$$

$$p_a = p_b/(1 - count \times p_b) \tag{2.4}$$

($count$ は最後に破棄を行ってからキューに収容されたパケット数)

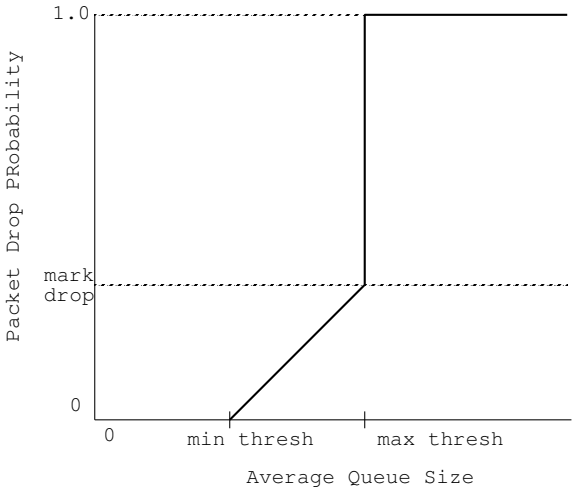

図 2.4: RED

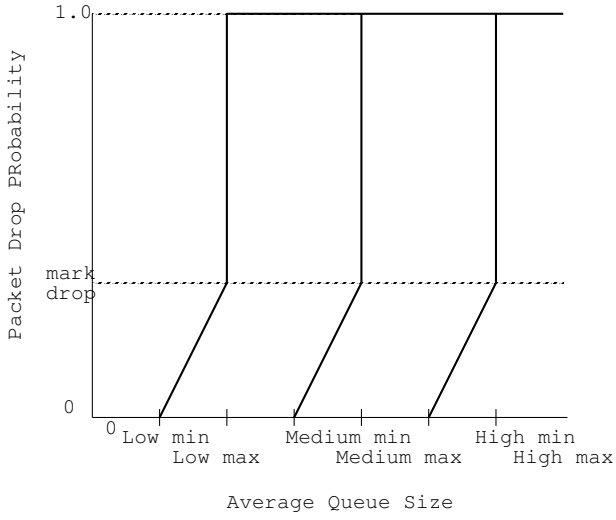

図 2.5: RIO

RIO

RIO とは、 RED (Random Early Detection) を拡張したキューバッファ管理方式である。 RIO は、さらにパケットをマークで分類し、パケットを破棄する確率をマークごとに変える。主に DiffServ の AF PHB で使用される。 AF PHB の破棄優先度ごとに平均キュー長の上限値、下限値、パケット破棄確率を設定することができる。キュー長が長くなると、パケット破棄優先度の高いものからパケットが破棄される。

2.2.3 パケットスケジューリング機構

ゲートウェイが現在のサービス (出リンクへの出力) を終了した際に、次にサービスするパケットの選択を行う機構である。パケット送出順序を制御することによって各フローに約束した QoS を提供する。例えば、輻輳時には低優先度のフローを無視して高優先度のフローを優先的にフォワードすることで高優先度のフローの最低帯域を確保し、遅延を抑える。代表的なスケジューリングアルゴリズムとしては、 WFQ、 WRR、 DRR がある。本研究ではこの内の DRR の拡張を行う。第 3.2 節では詳細に DRR の原理を説明する。

第 3 章

提案方式

3.1 既存方式の問題点

DiffServ においてクラス間に優先度の差別化を行う機構として、パケットスケジューリング機構 (第 2.2 節) がある。そのための既存のスケジューリングアルゴリズムとして、 WFQ、 WRR、 DRR がある。しかし、これらの手法はクラスごとの重み付けによって固定的にリソースの割り当てを行うため、トラフィックが量的にあるクラスに偏っている場合にトラフィック量の多いクラスの遅延が増大し、サービスの順番が逆転してしまうことがある。

3.2 DRR (Deficit Round Robin)

DRR [4] は可変長パケットのフローに対応した、公平性の高いスケジューリングアルゴリズムである。以下に DRR の基本的な動作原理を示す。フローの総数を n、ラウンドロビンの 1 巡で送出する総データ量の平均値を F とする。

1. 各フロー i に出リンクレートの ϕ_i, $(\sum_{j=1}^{n} \phi_j = 1)$ の割合を割り当てる。
2. 各フロー i に不足カウンタ $\phi_i F$ を設置する。
3. フロー i に順番が回ってきたら、合計データサイズ $\phi_i F$ の範囲内でフロー i からパケットをできるだけ送出する。
4. 実際に送出されたデータ長が $\underline{\phi}_i F$, $(\underline{\phi}_i \leq \phi_i)$ とすると、次回の不足カウンタは $(\phi_i + \phi_i - \underline{\phi}_i)F$ にセットされる。
5. 以下、ラウンドロビンで (3)　(4) を繰り返す。

つまり、 DRR の特徴はある順序で使い残した不足カウンタの余剰を次回に繰り越すことである。続いて、 F とフロー i のサービスレート r_i の関係を示す。簡単のために、すべてのフローのキューがアクティブであると仮定する。

出リンクのデータレートを R、フロー i の第 m ラウンド終了時刻を t^m、 m 回のラウンドロビンで実際にサービスされたデータ量を M_i^m、フロー i に属する最大パケット長を $L_{i,max}$ とする。M_i^m は $mF\phi_i$ と等しいか、 $mF\phi_i$ からその時点での不足カウンタの余剰を引いた値になる。余剰は $L_{i,max}$ を超えることはないので

$$(m\phi_i F - L_{i,max}) \leq M_i^m \leq m\phi_i F \tag{3.1}$$

が成り立つ。フロー i の第 m ラウンドの平均出力サービスレートを r_i^m とすると、

$$R = \sum_{i=1}^{n} r_i^m = (\sum_{i=1}^{n} M_i^m)/t^m$$

したがって、 (3.1) より

$$(mF - \sum_{i=1}^{n} L_{i,max}) \leq Rt^m \leq mF$$

$$\iff \frac{1}{mF} \leq \frac{1}{Rt^m} \leq \frac{1}{mF - \sum_{i=1}^{n} L_{i,max}} \tag{3.2}$$

$$(m\phi_i F - L_{i,max}) \leq r_i^m t^m \leq m\phi_i F \tag{3.3}$$

(3.2)、 (3.3) より

$$\frac{(\phi_i F - L_{i,max}/m)R}{F} \leq r_i^m \leq \frac{R\phi_i F}{F - (\sum_{i=1}^{n} L_{i,max})/m}$$

$$\iff \phi_i R(1 - L_{i,max}/m\phi_i F) \leq r_i^m \leq \frac{\phi_i R}{1 - (\sum_{i=1}^{n} L_{i,max})/mF}$$

$$\therefore r_i = \phi_i R \qquad (\because r_i = \lim_{m\to\infty} r_i^m) \tag{3.4}$$

したがって、フロー i のサービスレート r_i は F の値に関係なく $\phi_i R$ となる。

3.3 提案方式の概要

本研究では、第 3.1 節の問題を解決するために DRR (第 3.2 節) を拡張する。拡張機能は単純である。クラスごとに輻輳の度合いを測定して割り当てられた重みを動的に変更する。その際の閾値として、そのクラスのキュー長 (以下 $qlength$) かあるいはパケットのキュー内平均滞在時間 (以下 avg_time) [10] を用いる。以下にその具体的なステップを示す。

1. 各クラス i $(1 \leq i \leq n)$ に出力レートの割合 ϕ_i (以下レート) の初期値を割り当てる。
2. 各クラス i のキューからパケットが出力されるたびに、その瞬間のキュー長 $qlength_i$ もしくはクラス i のパケットのキュー内平均滞在時間 avg_time_i を算出する。
3. $qlength_i$、もしくは avg_time_i がある閾値 $maxth_i$ を超えた場合には輻輳とみなして、輻輳してないクラス j $(j \neq i)$ の ϕ_j の 10% を借り受ける。このとき、$\phi_i = \phi_i + \phi_j * 0.1$, $\phi_j = \phi_j - \phi_j * 0.1$. クラス i のこの状態を $state(1)_i$ とする。
4. $qlength_i$ もしくは ave_time_i がある閾値 $minth_i$ を下回った場合、輻輳解除とみなしてレートを借り受けたクラス j にレートを返す。このとき、$\phi_i = \phi_i - \phi_j * 0.1$, $\phi_j = \phi_j + \phi_j * 0.1$.
この状態を $state(0)_i$ とする。開始直後は $state(0)_i$ である。

通信中は、この手続きを継続する。レートを貸し借りする際の諸条件は以下の通りである。

- レートを他クラスから借りる、もしくは貸すことができるのは $state(0)$ のときだけである。
- レートを借りるクラスの優先順位は、 $n, n-1, \cdots, 1$ とする。これによって $1, 2, \cdots, n$ の優先度順にクラスの差別化を行う。
- クラスごとに貸すことのできるレートに限界値を定め、最低帯域を確保する。
- レートを借りる適合クラスがない場合は、強制的にクラス n から借りる。

ここで重要な点は、 $maxth$ と $minth$ を同一の値にはしない (図 3.1) ことである。スケジューリングアルゴリズムに輻輳制御機能を付加する場合の、重みが頻繁に切り替わる現象をフラップ現象 [11] というが、これを防ぐためである。

キュー内平均滞在時間の算出法

RED における平均キュー長の算出法 (式 2.1) にマッピングされた avg_time の算出法が知られている。 [10]

avg_time の算出は、パケットがキューの出口にたどり着いたときに行われる。クラス i の第 m

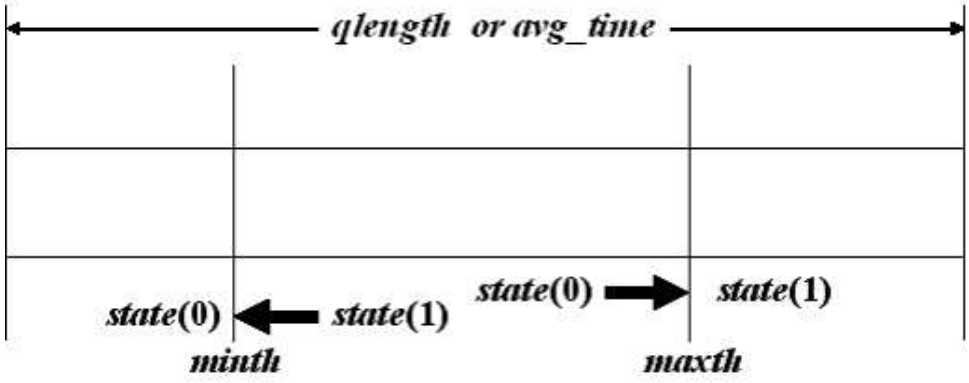

図 3.1: 輻輳制御の閾値

番目のパケットがキューの出口にたどり着いたときの $avg\_time_{i,m}$ は

$$avg\_time_{i,m} = (1 - w)avg\_time_{i,m-1} + wT_{i,m} \tag{3.5}$$

(w はローパスフィルタの重み係数、 $T_{i,m}$ は第 m パケットのキュー滞在時間である。)

第 4 章

シミュレーションによる評価

4.1 概要

提案した DRR を DiffServ 上で使用したときの性能を、 ns-2 (Network Simulator ver.2) [8] によるシミュレーションによって検証した。

ネットワーク構成

シミュレーションで用いるネットワークのトポロジを図 4.1 に示す。ネットワークを構成するノードは以下の通りである。

- トラフィックを発生させるノード
  VoIP クライアント 1　 n 台、ビデオストリーミングサーバ 6 台、 FTP 及び
  Telnet のサーバ 3 台
- DS ノード
  エッジノード 2 台、コアノード 1 台
- トラフィックを受信するノード
  VoIPSink (VoIP トラフィックを受信するノード) 、ストリーミングクライアント 2 台、
  FTP 及び Telnet のクライアントが 3 台

これらのノード間で、 DS ドメインをわたるトラフィックを送受信させる。それぞれのノードの具体的な PHB の割り当てを表 4.1 に示す。

本研究においては、 QoS の効果をわかり易くするためにクライアントノード側のエッジノードとコアノードをボトルネックにして意図的に輻輳状態にしている。ボトルネックの出リンクレートは 5Mbps である。優先度は、 EF クラス (Premium) 、 AF1 クラス (Gold) 、 AF2 クラス (Silver) 、 AF3 クラス (Bronze) 、 BE クラス (Best Effort) の順に設定してある。

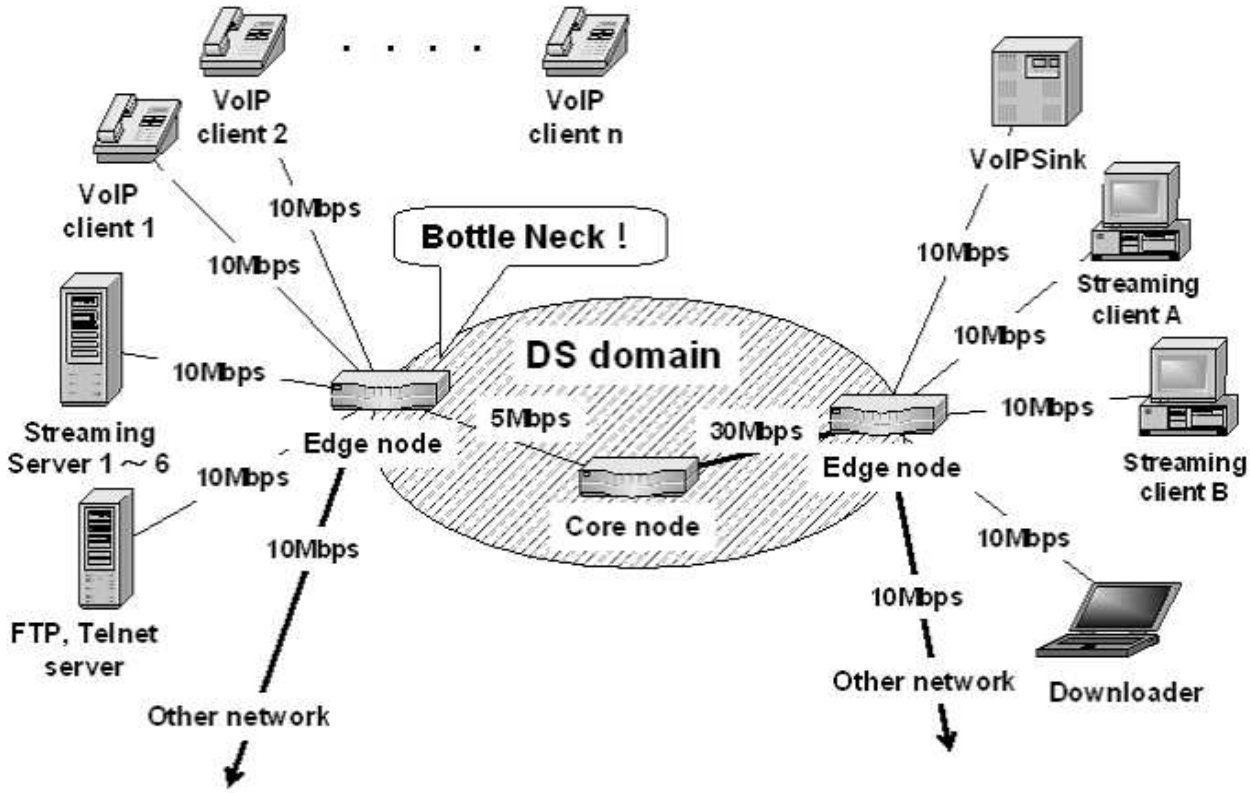

図 4.1: シミュレーション上ネットワークのトポロジ

4.2 EF クラスの保証

EF クラスを使用する場合には、一般的に、帯域、遅延などを保証するために Meter での厳格なポリシングが推奨されている。本来ならば、パケットがキューに入力される前に Token Bucket などを用いてトラフィックを規定の条件に合うように整形しなければならないが、本研究ではある程度輻輳した状況での振る舞いを検証するため Token Bucket を使用していない。

4.2.1 実験の目的

EF クラスに、様々なアプリケーションの中でも厳格な QoS 保証が必要な VoIP トラフィックを流してその品質を測定する。 EF クラスの保証帯域内でも、他のクラスがある程度輻輳してくるとその影響を受け、品質が十分に保証されないことが既に実験でわかっている [12]。この実験では、そのような状況でもパケットロスを防ぎ、より低ジッタ、低遅延を目指し、通常方式と提案方式の性能差を比較する。

表 4.1: シミュレーションにおけるトラフィックコンディショニング

Traffic type (class)	PHP	DSCP	DRR rate	Meter	Queue manager
VoIP (G.711) (Premium)	EF	10	0.1	–	Droptail
Audio and video streaming (Gold)	AF11 AF12 AF13	15 16 17	0.15	TSW3CM CIR 250Kbps PIR 500Kbps	RIO-C Green 80, 160, 0.02 Yellow 40, 80, 0.10 Red 20, 40, 0.20
Audio and video streaming (Silver)	AF21 AF22 AF23	20 21 22	0.15	TSW3CM CIR 250Kbps PIR 500Kbps	RIO-C Green 80, 160, 0.02 Yellow 40, 80, 0.10 Red 20, 40, 0.20
Telnet FTP (Bronze)	AF31 AF32	30 31	0.15	Token Bucket CIR 250Kbps CBS 20Kbyte	WRED 60, 120 0.2, 0.4
Other (Best Effort)	Default	40	0.5	–	Droptail

4.2.2 測定の方法

実験では VoIP の規格に G.711 を採用した。 G.711 はパケットサイズが 200byte、転送レートが 64kbps である。実験ではノードごとにバースト時間が 1 秒、アイドル時間が 1.6 秒と交互にトラフィックを発生させているので実質的な転送レートは約 25kbps である。 EF クラスにはレートとして 0.1 割り当てられている (表 4.1) から、 500kbps が保証帯域内であり、理論的には約 20 セッションまでの通信は品質が保証されるはずである。そこで、セッション数ごとに 100 秒間 VoIP トラフィックを流し、通常方式と提案方式の性能を比較する。提案方式については、閾値 (図 3.1) としてキュー長 *qlength* とを採用した場合と、パケットのキュー内平均滞在時間 *avg_time* を採用した場合の 2 つのパターンを検証する。

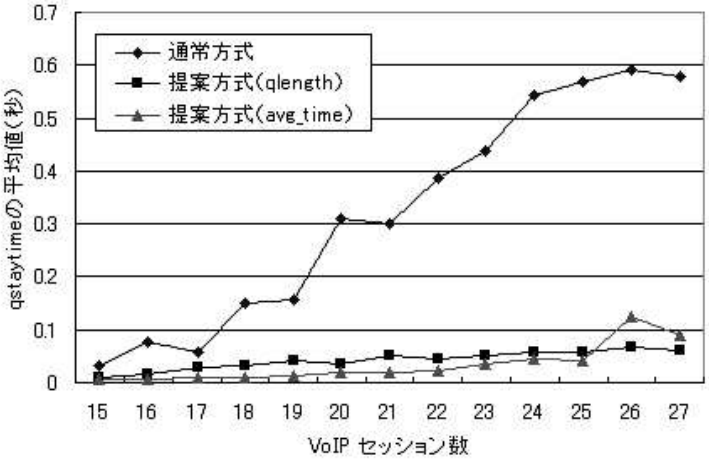

図 4.2: VoIP セッション数と *qstaytime* の平均値

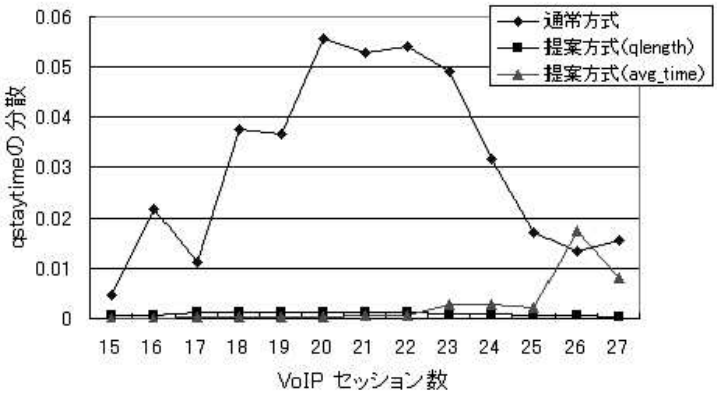

図 4.3: VoIP セッション数と *qstaytime* の分散

4.2.3 測定結果と考察

図 4.2 は、 VoIP セッション数と、キューを通過した VoIP パケットのキュー内滞在時間 (以下 *qstaytime*) の平均値である。図 4.3 はその分散である。図 4.4 はパケットロス率の推移を示している。

図 4.2 より、通常方式だとセッション数が 20 に満たない場合でも *qstaytime* が急激に増え、その結果遅延も増大してしまっている。一方提案方式では、閾値が *qlength*、 *avg_time* いずれの場合も遅延を抑えるのに成功している。また、図 4.3 から、 *qstaytime* の分散も抑えられていることもわかる。これは、 *qstaytime* のばらつきが少ないということであり、低ジッタを実現していることがわかる。さらに図 4.4 より、パケットロス率をほぼ 0% に保っていることがわかる。

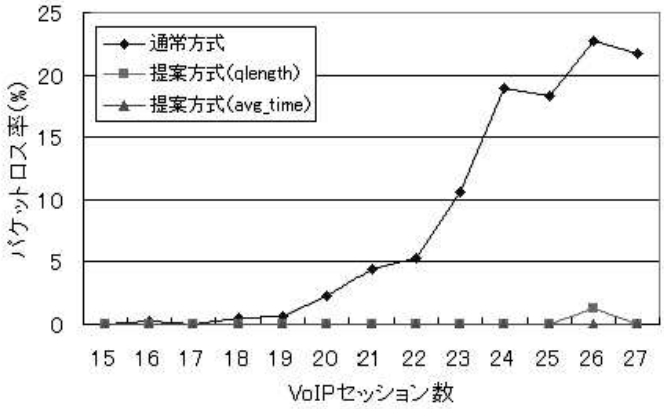

図 4.4: VoIP セッション数とパケットロス率

続いて、セッション数が 20 のときの *qlength* と *qstaytime* の時間分布の、各方式ごとの違いを図 4.5 から図 4.10 に示す。これらを見ると、閾値が *qlength*、 *avg_time* いずれの場合も *qlength* の増大を防ぎ、遅延が抑えられている結果が得られた。また若干ではあるが、閾値が *avg_time* である場合の方が性能がよいことがわかった。

VoIP の遅延については、 ITU-T から G.114 として勧告されている (表 4.2) 。一般的にネットワーク上の遅延で大きな影響を及ぼすのは、ゲートウェイやルータなどでパケットが送信待ちキューに入力されてから送出されるまでの伝送待ち時間、すなわち *qstaytime* である。ある程度の輻輳したゲートウェイ上でも *qstaytime* を 0.1 秒以下に抑えることができたので、このようなゲートウェイを多段ホップしなければ規定通りの通信品質を得られるはずである。

表 4.2: ITU-T G.114 の音声遅延のガイドライン

片道方向遅延 (ms)	説明
0　150	多くのアプリケーションで利用可
150　400	送信時間がユーザー・アプリケーションの送信品質に 影響することを承知していれば利用可 (ユーザーが承知していることが前提であり推奨はできない)
400 以上	一般的に利用不可

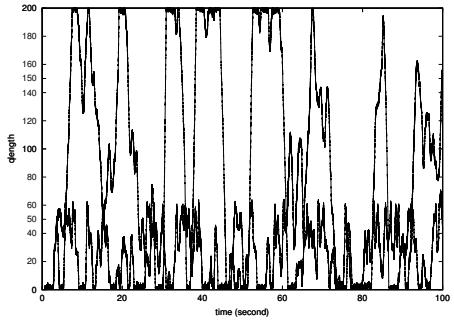

図 4.5: 通常方式の *qlength*

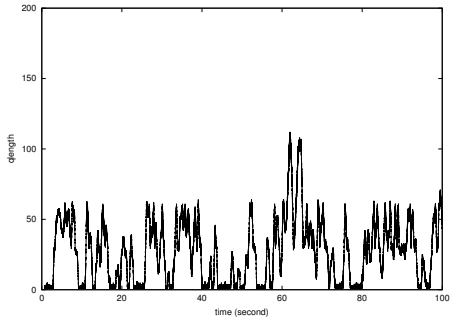

図 4.6: 提案方式 (*qlength*) の *qlength*

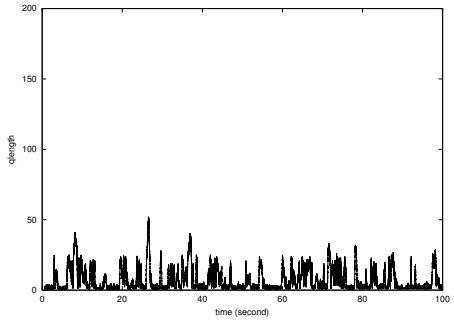

図 4.7: 提案方式 (*avg_time*) の *qlength*

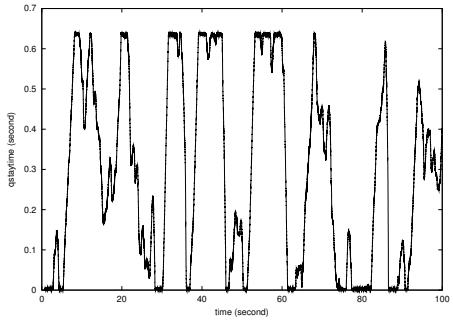

図 4.8: 通常方式の *qstaytime*

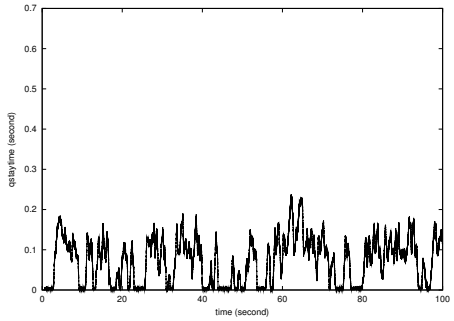

図 4.9: 提案方式 (*qlength*) の *qstaytime*

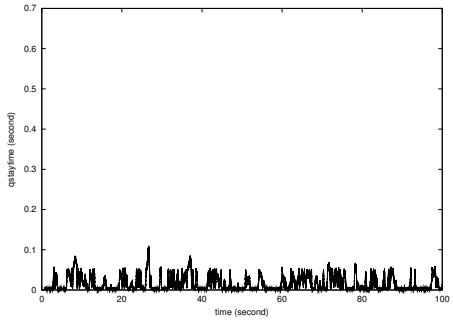

図 4.10: 提案方式 (*avg_time*) の *qstaytime*

4.3 AF クラスの保証

AF クラスは、 4 つの独立したクラス (本研究で用いるのは 3 つ) において最低帯域が保証され、さらに各々のクラスごとに 3 つの破棄優先度設定される。破棄優先度は、優先度順に Green、Yellow、 Red の色別で定義される。本研究では、 DSCP マーキングに用いる Meter として、TSW3CM を用いる。 TSW3CM は、 CIR (平均データレート)、 PIR (ピークレート) を設定し、以下の法則にしたがってマーキングを行う。

TS3WCM で計測されたデータレートを $Rate$ とする。

- $Rate < CIR$ のとき
  Green にマーキングされる。

- $CIR \leq Rate < PIR$ のとき
  Green にマーキングされる確率を P_0、 Yellow にマーキングされる確率を P_1 とすると、

$$P_1 = (Rate - CIR)/Rate$$
$$P_0 = 1 - P_1$$

- $PIR \leq Rate$ のとき
  Green にマーキングされる確率を P_0、 Yellow にマーキングされる確率を P_1、 Red にマーキングされる確率を P_2 とすると、

$$P_2 = (Rate - PIR)/Rate$$
$$P_1 = (PIR - CIR)/Rate$$
$$P_0 = 1 - P_1 - P_2$$

4.3.1 実験の目的

AF1 クラス、 AF2 クラスは、通常は EF クラスの次に優先するアプリケーションに適用する。本研究では、ビデオストリーミングのトラフィックを流した。この実験では、低ジッタ、低遅延を目指しつつ AF クラスの破棄制御が提案方式にどのような影響を及ぼすのかを検証する。

4.3.2 測定の方法

AF1 クラスには、 3 台のビデオストリーミングサーバからそれぞれ 250kbps、 300kbps、 350kbps のトラフィックを流した。パケットサイズは 400byte である。 AF2 クラスには AF1 クラスと同じ条件で、パケットサイズのみ 800byte に変えてトラフィックを流した。さらに、 FTP サーバと Telnet サーバを計 3 台用意し 1 台のクライアントノードとの通信を AF3 クラスに適用した。

それぞれのクラスには、レートとして 0.15、 0.15、 0.1 と割り当てている (表 4.1) 。すなわち、保証帯域は 750kbps、 750kbsp、 500kbs である。 EF クラスのときと同様に、通常方式と提案方式 (*qlength*、 *avg_time*) の性能差を比較する。実験時間は 100 秒である。

4.3.3 測定結果と考察

AF1 クラスの時間と *qlength* の関係を図 4.11 に、時間と *qstaytime* の関係を図 4.12 示す。AF2 クラスの時間と *qlength* の関係を図 4.13 に、時間と *qstaytime* の関係を図 4.14 示す。続いて、 AF1 クラス、 AF2 クラスの各方式ごとの入力パケット数、出力パケット数、破棄パケット数を表 4.3 と表 4.4 示す。

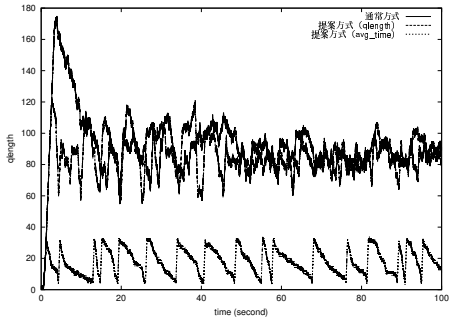

図 4.11: AF1 クラスの時間と *qlength* の関係

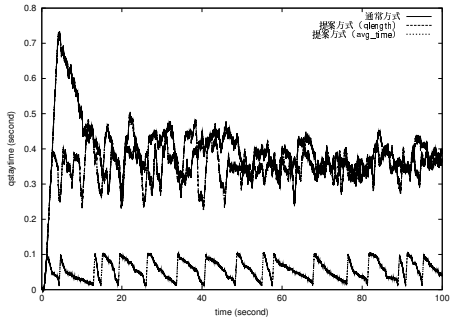

図 4.12: AF1 クラスの時間と *qstaytime* の関係

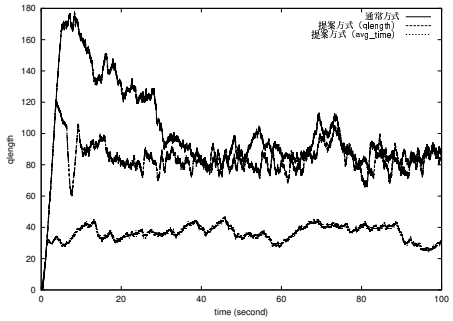

図 4.13: AF2 クラスの時間と *qlength* の関係

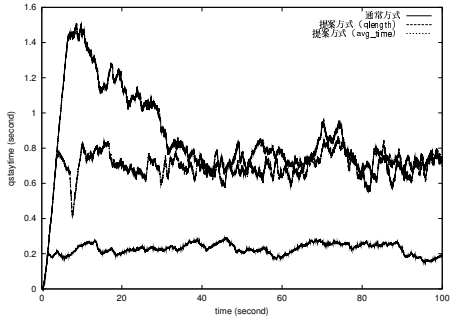

図 4.14: AF2 クラスの時間と *qstaytime* の関係

EF クラスと同様に、 AF クラスでも提案方式 (*avg_time*) は通常方式と比較して遅延、及びパケットロスを抑えることに成功している。提案方式 (*qlength*) では、提案方式 (*avg_time*) ほどの性能はみられなかった。しかし、 AF クラスのキューが短いときのバースト的なトラフィックによるキューの急激な伸長を防止している。キューの急激な伸長があると、 RED のパケット破棄制御に用いる平均キュー長の計算 (式 2.1) が遅れて、破棄制御にうまく反映されない。防止の効果は若干ではあるが、パケット破棄数の低下に表れている。提案方式 (*qlength*) は、終始 *qlength* と *qstaytime* を安定させており、低ジッタを実現していることがわかる。

表 4.3: AF1 クラスの入力パケット数、出力パケット数、破棄パケット数

方式	優先度	入力パケット数	出力パケット数	破棄パケット数	RED による破棄パケット数
通常方式	Green	23241	23143	0	98
	Yellow	7129	498	6541	90
	Red	591	0	591	0
提案方式	Green	23152	23100	0	52
($qlength$)	Yellow	7202	785	6271	146
	Red	613	0	613	0
提案方式	Green	23276	23276	0	0
(avg_time)	Yellow	7087	7087	0	0
	Red	600	594	0	6

表 4.4: AF2 クラスの入力パケット数、出力パケット数、破棄パケット数

方式	優先度	入力パケット数	出力パケット数	破棄パケット数	RED による破棄パケット数
通常方式	Green	11728	11637	0	91
	Yellow	3672	231	3414	27
	Red	522	3	519	0
提案方式	Green	11608	11598	0	10
($qlength$)	Yellow	3804	460	3267	77
	Red	504	3	501	0
提案方式	Green	11692	11692	0	0
(avg_time)	Yellow	3712	3712	0	0
	Red	518	338	31	149

また、提案方式 (avg_time) が AF クラスにおいて特に高い効果を示すのは、 RED による破棄制御が開始される前に提案方式による輻輳制御が開始されて、 $qlength$ が一定以上に伸びないためである。

4.4 シミュレーションの総括

4.4.1 転送データ・破棄データによる評価

シミュレーションによる実験のすべてのトラフィックの転送データ量と破棄データ量を図 4.15 に示す。

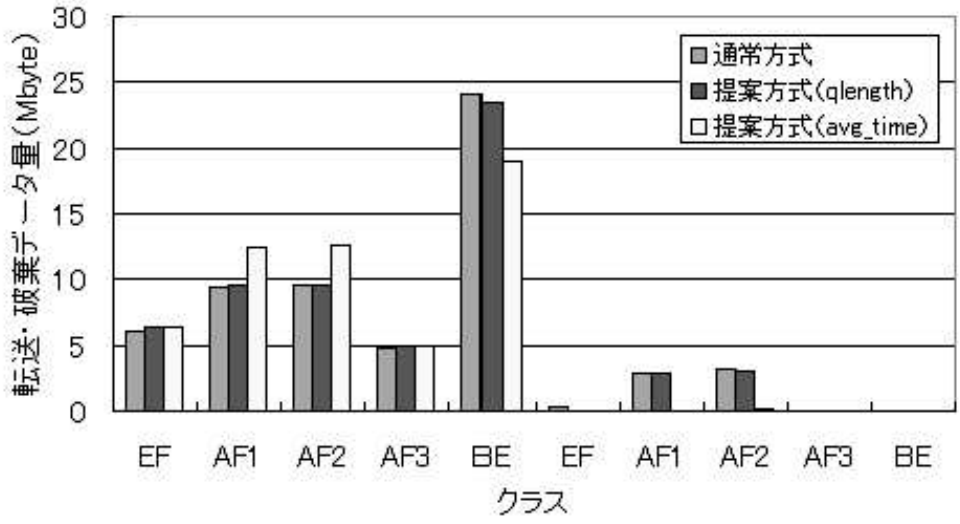

図 4.15: 転送データ (左) : 破棄データ (右)

第 4.2.3 節、第 4.3.3 節において高い効果が得られた提案方式 (*avg_time*) だが、代わりに BE クラスの全体的なスループットが約 20% ほど低下している。シミュレーションで BE クラスに流したトラフィックは主に HTTP と FTP であるため若干スループットが低下しても問題はなかった。しかしこの結果から、この方式を使用する場合にはリアルタイム通信は極力優先クラスで行うなど、各クラスで使用するトラフィックの性質をよく吟味する必要がある。

一方提案方式 (*qlength*) は、提案方式 (*avg_time*) ほどの性能を得られない場合もあるが、全体のスループットを落とすことなく純粋に品質のみを向上させることに成功している。特に EF クラスにおいては、提案方式 (*avg_time*) との差がほとんど見られなかった。

4.4.2 RED の閾値

RED (図 2.4) の閾値に、 *qlength* ではなく *avg_time* を使用して第 4.2 節、第 4.3 節と同様の実験を行った。この結果として、提案方式と組み合わせた場合、 AF クラスの *qstaytime* をさらに抑えることには成功したが、代わりにパケットの破棄率が上昇した。これは、 *qlength* を閾値とした場合よりも早い段階で破棄制御が行われるためである。なお、 EF クラスの品質にはほ

とんど影響がなかった。

4.4.3 DRR の影響

第 3.2 節において、理論的には DRR のラウンドロビンの 1 巡で送出する総データ量の平均値 F と、実際のフローのレートは関係がないことを説明した。しかし、F の値を変えながら実験をしてみると若干ではあるが F が小さいときのほうが品質が向上することがわかった。これは、提案方式ではラウンドロビンでキューに順番が回ってきたときにそのキューの輻輳度合いを測定しているので、F を小さくして巡回回数を多くしたほうがキューに対して輻輳制御がより敏感に働くようになるためである。

第 5 章

まとめ

5.1 結論

本研究では、既存のスケジューリングアルゴリズム DRR に輻輳制御機能を付加した。輻輳制御の閾値として、 $qlength$ と avg_time の 2 つのパターンを採用し、シミュレーションによりその評価を行った。 DiffServ 上で提案方式を活用した場合に、以下のことがわかった。

- 提案方式を EF クラスで使用する場合、制御の閾値として $qlength$ と avg_time のどちらを採用しても効果がある。
- 提案方式を AF クラスで使用する場合、閾値として avg_time を採用したときのほうが効果は上だが、 BE クラスのスループットの低下に注意する必要がある。 $qlength$ を採用した場合は、通信初期のキューの急激な伸長を防ぐことができ、低ジッタを実現できる。

提案方式を DiffServ のパケットスケジューリングアルゴリズムとして活用することは有効である。特に EF クラスでは、ある程度輻輳した状態でも契約通りの品質のサービスを提供できる。

5.2 今後の課題

提案方式では、閾値によってその性能が大きく左右される。閾値を決める際には、使用するアプリケーションの性質をよく見極め、慎重を要する。今後は、制御全体のパフォーマンスがより向上するための最適な閾値の組み合わせを模索していく必要がある。また、今回のシミュレーションではキューの数は高々 5 つであったが、キューの数の増加が提案方式の輻輳制御機能に及ぼす影響を調査する必要がある。

謝辞

本修士論文の作成にあたり日頃より御指導を頂いた早稲田大学理工学部の後藤滋樹教授に深く感謝致します。また、多大なる御協力を頂きました後藤研究室の諸氏に感謝致します。

参考文献

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC2475, IETF, December 1998.

[2] V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB", RFC2598, IETF, June 1999.

[3] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group", RFC2597, IETF, June 1999.

[4] M. Shreedhar, George Varghese, "Efficient Fair Queuing using Deficit Round Robin", Proc. of ACM SIGCOMM'95, September 1995.

[5] W. Fang, N. Seddigh, B. Nandy, "A Time Sliding Win-dow Three Colour Marker", RFC2859, June 2000.

[6] S. Floyd, V. Jacobson, "Random early detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking, Vol.1, No.4, pp.397–413, August 1993.

[7] D. Clark, W. Fang, "Explicit allocation of best effort packet delivery service", IEEE/ACM Transactions on Networking, Vol.6, No.4, pp.362–373, August 1998.

[8] VINT group, UCB/LBNL/VINT network simulator (version 2), 1997.
`http://www.isi.edu/nsnam/ns/`

[9] S. Andreozzi, "DiffServ simulations using the Network Simulator: requirements, issues and solutions", 2001
`http://www.cnaf.infn.it/\%7Esergio/diffserv/ns/`

[10] 後藤渉, 保岡裕, "遅延を考慮した適応型 DiffServ スケジューラとバッファ管理アルゴリズムの提案", 電子情報通信学会 コミュニケーションクオリティ研究会, CQ2002-14, November 2002.

[11] 小野寺悟, ”Diffserv 型パケットスケジューリング方式 DEWRR における性能評価と検証”, 早稲田大学大学院国際情報通信研究科 2001 年度修士論文.

[12] 佐々木洋美, ”DiffServ の環境における VoIP の品質の評価”, 早稲田大学理工学部情報学科 2002 年度卒業論文.

[13] W.Richard Stevens, 井上尚司監訳, 橘康雄訳『詳解 TCP/IP プロトコル』ソフトバンク, 1997.