平成 14 年度

学士学位論文

クラスベース QoS 制御モジュールの並列実現法

The Parallel Implementation of Class Based QoS

1030243　　岩井　秀樹

指導教員　　岩田　誠

2003 年 2 月 24 日

高知工科大学 情報システム工学科

要 旨

クラスベース QoS 制御モジュールの並列実現法

岩井　秀樹

ネットワークの接続形態の高度化および多様化、また通信技術の飛躍的進展を受けて、要求に見合う適切な通信品質や先進的機能を提供できる通信サービスが求められている。この要求を満たすべく、QoS(Quarity of Service) 制御に関する研究が進められているが、QoS 制御は複雑かつ柔軟な処理を必要とするため、単純に ASIC 化等によって処理の高速化を計ることが困難である。このため、近年、固定的なネットワーク処理のみ専用ハードウェアモジュール化した、プログラマブルなネットワークプロセッサの研究開発が盛んに行われているが、集中制御方式に起因するマルチスケジューリングオーバーヘッドやプロセッサ間通信遅延のため、性能を十分に活かすことが難しい。

本研究では、クラスベース QoS 制御モジュールを並列実現するために、スケジューリング不要、かつ負荷に対する緩衝特性を活かした DDMP を用い高速かつ柔軟なクラスベース QoS 制御の実現を目指した。

機能モジュールを構成するにあたり、非同期であるデータ駆動方式のフロー処理過程では、メモリ処理においての reader/writer 問題や、除算に対する問題、小数点以下の計算に対する問題が挙げられたが、それぞれ、メモリ管理の一元化、シフトによる演算の実装・近似などの手法を用いて対処している。

性能評価の結果から、スループット最大 12.1Mpps のパケット処理が可能であることを確認している。また、Algorithmic Dropper と RED がボトルネックとなっており、この処理の高速化によって、クラスベース QoS 制御機構の更なる高速化が期待できる。

キーワード　QoS、クラスベース QoS、DDMP

Abstract

The Parallel Implementation of Class Based QoS

The communication services are demanded, high-development and diversify of network connection form network, rapidly progress communication technology, correspond to require appropriate communication quarity and advanced function. Researches about QoS(Quarity of service) control are proceeded for this require,but QoS control needs complicate and flexible process. Therefore, it is diffcult that is processed by ASIC. Thus researches are proceeded heatly about programable network processor. However cause of difficult to harnesse performance is multiprocessor scheduling overhead and communication delay between processors.

In this research, for parallel implementation of class based QoS control, aim implementation of high-speed and flexible class based QoS control on DDMP(Data Driven Multimedia Processor).

For composing function modules, the problems are reader/writer error in memory process and dividing, calculation below a decimal point. As opposed to these problem, unification of memory management and shift processing, are solutions.

As a result of performance evaluation, max throughput is 12.1Mpps. Also Algorithmic Dropper and RED are bottleneck. Therefore, by improvement in processing speed, it is able to hope for processing performance of class based QoS control.

key words QoS, class based QoS, DDMP

目次

図目次

表目次

第 1 章

はじめに

近年の光増幅技術や光波長多重技術などの光伝送技術の進歩によりネットワークの帯域幅の拡大が著しい。更に、ネットワークへの接続のための情報端末も高度化・多様化が進んでいる。これに伴い、サービスの要求に見合う適切な通信品質や OLTP などの先進的機能を提供できる QoS（Quarit of Service）制御に関する研究が進められているが、QoS 制御は複雑かつ柔軟な処理を必要とする [1]。

QoS とは、一般的には、パケットのフローのレート、遅延時間、ジッタ、パケットロスなどを保証した上で、様々なサービスを提供する考え方で、例えば

- レート

  最低レート保証

  平均レート保証

  保証なし

- 遅延時間

  最大遅延時間保証

  保証なし

- ジッタ

  最大ゆらぎ幅の保証

  平均ゆらぎ幅保証

  保証なし

- パケットロス

  最大ロスレートの保証

  保証なし

などがあげられる。

このような QoS 制御のうち、パケットのフロー単位で行うものを Intserv、パケットのクラス単位で行うものを Diffserv という。

Intserv は送信側と受信側の間で通信を行い、転送に必要な帯域幅と転送路をあらかじめ確保しておき、転送中に帯域幅が十分に取れなくなると転送路を動的に変更しながら帯域幅を確保して通信を行う方式である。Intserv は、ネットワークのリソースを動的に確保するプロトコルである RSVP（Resource Reservation Protocol）によって動作している。

RSVP では、図 1.1 のように送信側は、まず、受信側がデータを転送するルート上においてリソースの予約を行えるように、ルートに沿って RSVP パスメッセージを受信側に向かって送信する。ルート上の各ルータは、次々にこのパスメッセージを受信し、転送するが、このときに、各ルータは RSVP プロセスで利用するためのパス情報テーブルをルータ内部に作成する。パス情報テーブルには、送信ホスト方向への逆方向パスに沿って「RSVP リソース予約メッセージ」を送信するため、受信したパスメッセージが最後に通過したルータの IP アドレスを含む。パスメッセージを受信したホストは、送信ホストに「RSVP 予約メッセージ」を送信する。このとき、メッセージを受信したホストは、内部にリソース予約の情報テーブルを作る。転送するルートが帯域幅などの確保の関係で、変更になることがあるので、送信ホストと受信ホストは予約したリソースの維持をするために、定期的に予約メッセージを交換しなければならない。

このように、Intserv では、ネットワーク上をパケットが転送される前にアプリケーションからクライアントに向かって接続要求メッセージである PATH メッセージが、クライアントからアプリケーションに向かって許可メッセージである RSVP メッセージが流れる。また、予約の確立後も定期的に RSVP メッセージを流す必要があり、ネットワークリソー

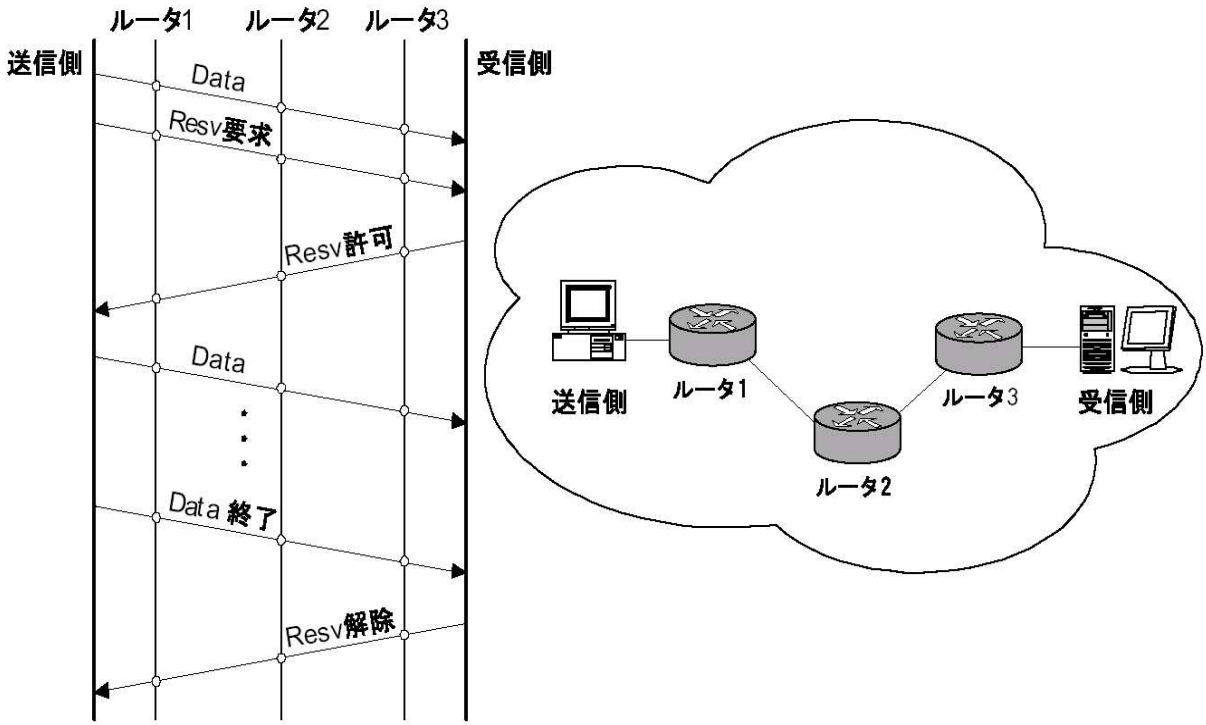

図 1.1　Intserv 処理

スの消費が大きい。

このことは、Intserv が LAN（Local Area Network）や WAN（Wide Area Network）のようなルータの数が少ない限られたネットワークに適している事を示している。

Diffser では、同一のルールによって運用されるネットワークを Diffserv ドメイン（DS ドメイン）という。DS ドメインは、ドメイン内と外部の境界にあり、ドメイン内に入ってくるデータフローを決められているルールに基づいてクラス分け（Classifier）し、クラスに対応した DSCP 値を IP パケットヘッダにセットするエッジ（境界）ルータと、IP パケットヘッダの DSCP 値に基づいて、転送処理をするコア（中心）ルータに分けられる（図 1.2)。

エッジルータで行われる処理としては、フローを識別し、クラス分けする作業や、DSCP 値をセットする処理が行われる。一方コアルータではエッジルータからのフローが集約され DSCP に基づいて、転送されるだけで、複雑な処理の必要がないので、高速化が可能である。

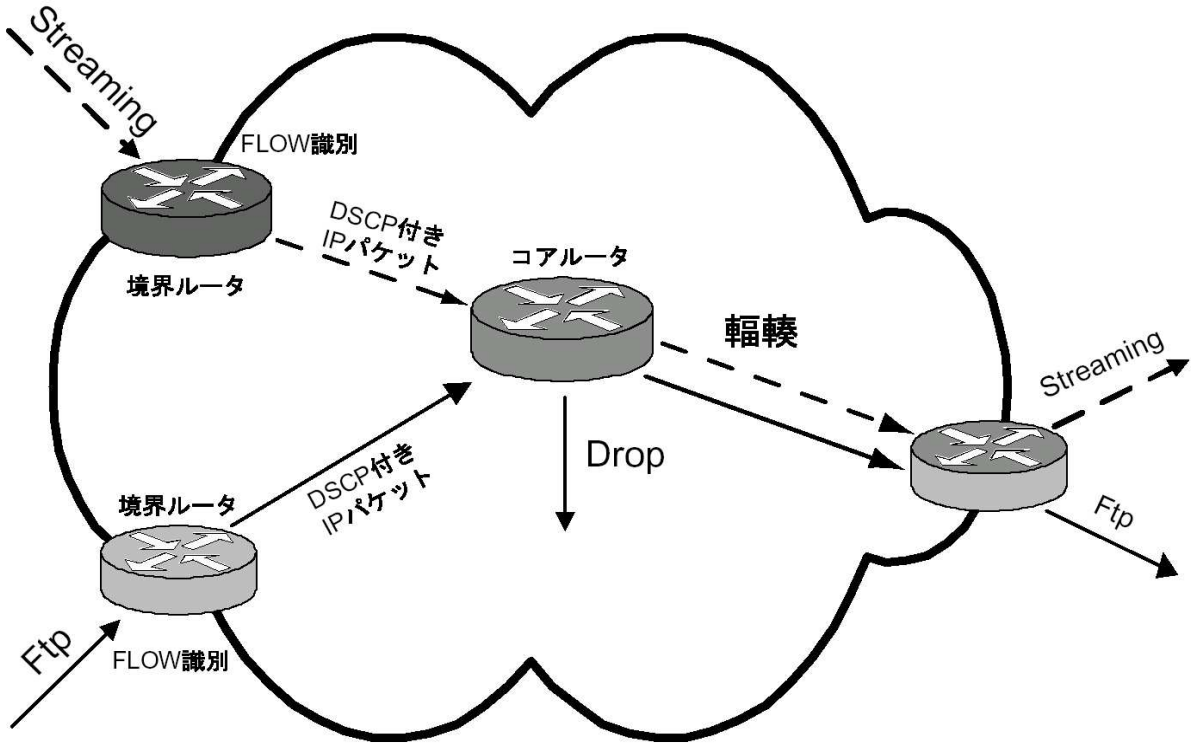

図 1.2　Diffserv 処理

また、ルータが行うパケットの転送処理は、IP ヘッダ内の DSCP 値に基づいて行われるので（図 1.3)、ネットワーク上に余分なメッセージが流れることがなく、ネットワークリソースの消費が少ない。

Intserv を公衆のネットワークで使おうとすると、Intserv そのもののシステムが巨大化し、ネットワークリソースの消費も大きくなる。それに対して、Diffserv は DS ドメインを連結するだけで、容易に拡張ができる。このため、Diffserv はインターネットでの QoS の実現に向いている。

しかしながら、単純に ASIC 化等により、QoS の高速化を図ることは柔軟性を欠くため困難である。このため、近年、固定的なネットワーク処理のみ専用ハードウェアモジュール化した、プログラマブルなネットワークプロセッサの研究が盛んに行われている。だが、集中制御方式に起因するマルチプロセッサスケジューリングオーバーヘッドやプロセッサ間の通信遅延のため、性能を十分に活かすことは難しい。

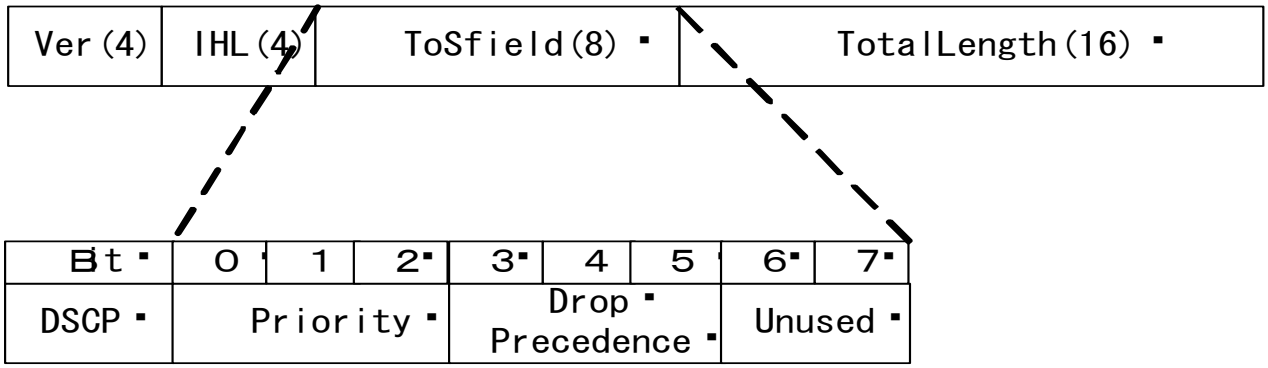

図 1.3　IPheader 構成

そこで、本研究では、データ駆動方式の高速フロー処理特性と、自己同期パイプラインの負荷に対する緩衝能力を活かした DDMP(データ駆動型マルチメディアプロセッサ)[2] を用いて、クラスベース QoS 制御モジュールを並列構成し柔軟かつ高速な QoS 制御の実現を目指す。

2 章では、クラスベース QoS 制御のために必要なそれぞれの機能要素についての説明を行う。3 章では、非同期である、データ駆動方式のアーキテクチャを考慮に入れて構成した各機能モジュールの構成について述べる。4 章では、構成した機能モジュール、および優先制御機構の性能評価を行う。

第 2 章

クラスベース QoS 制御機構

2.1 緒言

本章では、クラスベース QoS 制御に必要な機能要素である、帯域保証制御機構、遅延時間制御機構、パケット廃棄制御機構、パケット分類機能 [3] のそれぞれの処理が担う役割と代表的な構成要素について述べる。

2.2 クラスベース QoS 制御概要

超高速ネットワークでは、ビットレートが飛躍的に向上し、それまで問題とならなかったネットワーク内のあらゆる遅延が問題として浮き彫りとなる。したがって、ネットワークノードにおいて、伝送および処理遅延の影響を局所化し、スループットの低下を極小化するための分散的処理システムの実現が課題となる。このため、パケットに含まれる情報のみで、サービスクラスを決定し、局所的に制御可能なクラスベース QoS を実現する必要がある。

そして、クラスベース QoS を実現するためには、ルータにはルーティング機能のほかに以下のような機能が必要となる。

1. 帯域保証制御機能
2. 遅延時間制御機能
3. パケット廃棄制御機能
4. 優先度に応じたパケット分類機能

2.3　帯域保証制御

大量のデータを遅延無く転送するためには、十分な帯域が必要である。そのため、ルータにも大量のデータを高速で処理することが求められる。現在のネットワークではベストエフォート型の転送方式の割合が高く、優先度の高いフローが契約よりも大きな帯域幅で転送されると帯域幅を占有されてしまい、他のフローが転送ができないといった悪影響をあたえる。そこで、優先度の高いフローを常に監視する必要がある。この制御をしているのが帯域保証制御である。

2.3.1　メータ（Meter）

メータは転送路上を単位時間あたりに流れるデータの bit 量を計算する構成要素である。クラスに分けられたパケットは、クラスごとにメータに入り、計算した結果と設定した値を比較する。帯域の占有率に違反がなければキューに送られ、違反があれば優先度を下げられてキューに送られるか廃棄される。以下に、その種類と特徴を述べる。

1. Average Rate Meter

   設定された単位時間以内に到着したパケットのビット数あるいは累積ビット数を表示する。もし単位時間内に到着したパケットのビット数が、設定された量を超えていると、そのフローは設計に違反していることになり、そのパケットはドロッパに送られ廃棄される。Average Rate Meter に必要なパラメータは以下の 2 つである。

   （a）Average Rate（設定平均レート）

   （b）Delta(単位時間)

2. EWMA(Exponential Weighted Moving Average) メータ

   EWMA メータは、直前の時刻 t' の平均転送速度 avg.rate(t') と現在の時刻 t の転送速度 rate(t) から、パケットを廃棄させるか通過させるか決定するためのメータである。そのパラメータは、

   （a）Average Rate

（b）Delta

（c）Gain(直前の平均転送速度に乗じる重み) で、式は以下のとおりである。

$$avg.rate(t) = (1 - Gain) \times avg.rate(t') + Gain \times rate(t)$$

$$t = t' + Delta$$

3. TB(Token Bucket) メータ

１，２で述べたメータより更に高度なメータである。TB メータは、到着したパケットの速度を論理的にバケツのなかにあるトークンと比較し、その値を閾値としてバースト性のフローの通過を許す。一例として、二つのトークンを閾値とするトークンバケット Two-ParaMeter Token Bucket を説明する。このメータは、R(average token rate) と B(burst size) の 2 つのパラメータを持つ。長さ L のパケットが到着したときに、バケツの中のトークンが L 以上残っているときには、バースト性のトラフィックでも通過可能である。これを厳格な適合 (strict conformance) という。一方、トークンの残りが L 未満の場合には、将来割り当てられるトークンを先取りして通すことができる。これをゆるい適合 (lose conforming) という。また、複数のトークンバケットを設定すれば、クラスごとに違ったパラメータを設定でき、複数のクラスに適合できる。

2.4　遅延時間制御

ネットワークの遅延は、大きく次の 4 種類に分けられる。

- 伝送路を構成する物質の物理的限界による遅延
- 伝送路での輻輳による遅延
- ネットワークノード内のキューイング遅延
- ネットワークノード内のソフトウェア処理の処理遅延

物理的な遅延は伝送路の長さと触媒の物理的な性質 (光ファイバや、銅線中の光や電子の伝達速度) が分かれば予測可能であるが、この遅延をゼロにすることは不可能である。これに

対し、輻輳による回線上の遅延時間は予測不可能であり、さらに、伝送路の帯域幅の広帯域化により、伝送路は単なる線ではなく、メモリとしての、記憶効果を持つようになり、輻輳および輻輳の結果生じるデータの廃棄によって、大きな遅延を生じるようになる。また、キューイング遅延は出力側リンクの輻輳状況に大きく影響され、予測不可能である。最後に、ソフトウェア処理の遅延は計算可能であり、プログラム構成の分散化によって小さく抑えることが可能である。

インターネット電話のようなリアルタイム通信では、遅延時間ももちろん小さく抑えなければならないが、遅延時間のゆれであるジッタを最小にしなければ通信している当人には不快なものになる。そのため、遅延時間とともにその標準偏差である、ジッタの制御を行い、最良の状態を提供する必要がある。この項では、ネットワークデバイス内の遅延時間を制御する機能要素である、キューエレメントとスケジューラについて述べる。

2.4.1　キュー (Queue) エレメント

キューは文字通り待ち行列である。ルータの入り口に流れてきたパケットは、出力されるまでの間バッファに記憶され、順番が来れば出力される。従来のキューの役目は、パケットの記憶とキューがパケットであふれた場合に廃棄を行うバッファ管理である。以下にキューエレメントの種類を述べる。

1. FIFO(First-In First-Out) キュー

   最も簡単なキューで入ってきた順番で出力する。

2. CBQ(Class Based Queue)

   キューをクラスごとに用意して、出力時に優先制御を行う方式である。優先度の高いパケットを優先して出力するために、順番の入れ替えを行う必要があり、従来方式のキューでは大きな遅延を生じる。

2.4.2　スケジューラ (Scheduler)

キューの先頭に位置するパケットを優先度に応じて出力する部分であり、出力するデータの量に重み付けを行う方式と行わない方式がある。以下に代表的なスケジューラの方式について述べる。

1. Round Robin（ラウンドロビン）方式

   優先度によらず、同じパケット数を出力する方式。たとえば、3 クラスの優先制御を行う場合、優先度の高いクラスを 5 パケット出力し、次に 2 番目に優先度の高いクラスを 5 パケット出力し、最後に一番優先度の低いクラスのパケットを 5 個出力する。これを繰り返し、重みつけは行わない。

2. Fair Queue（フェアキュー）方式

   クラスごとの順番に出力する点は、上記のラウンドロビン方式と同じだが、同じパケット数出力するのではなく、同じバイト数出力する。たとえば、最も優先度の高いパケットを 100 バイト出力した後に、次に優先度の高いクラスのパケットを 100 バイト、次に、最も優先度の低いクラスのパケットを 100 バイト出力する。これを繰り返し行う。また、重み付けは行わない。

3. WRR（Weighted Round Robin）方式

   重み付けを行うラウンドロビン方式で、たとえば、最もう優先度の高いクラスのパケットを 10 パケット出力した後に 2 番目に優先度の高いクラスのパケットを 5 パケット出力、最後に、最も優先度の低いクラスのパケットを 2 個出力する。これを繰り返し行う。この場合の重みは 10:5:2 となる。

4. WFQ（Weighted Fair Queue）方式

   フェアキュー方式に重みをつけたもの。たとえば、優先度の高い方から順に、100 バイト、50 バイト、20 バイト出力しこれを繰り返す。この場合の重みは 10:5:2 となる。

2.5　パケット廃棄制御

キューに入力されるパケットの到着レートが転送処理能力を超えたとき、処理しきれないデータがキューにたまる。たとえば、伝送容量が 1Mbps で、パケット到着レートが 2 Mbps のとき、キュー内のデータの量は毎秒 1 M ビットの割合で増加する。もしキュー長が L バイト、伝送両々が Tbps、パケット到着レートが Pbps ならば、8 L/P-T[秒] の遅延を起こす。また、廃棄の起こる時間を遅らせようとして、キューの量を多くとると、遅延時間も増大し QoS が悪化する。

このような事態にならないように、パケットがキューにたまりはじめたときに、キューの量に応じて積極的にパケットを廃棄する機構が必要である。以下に代表的な廃棄制御について示す。

2.5.1　ドロッパ (Dropper)

ドロッパはパケットを廃棄するエレメントである。パケットがキューに溜まってくると、キューの入り口で廃棄を行うことになるが、そこで廃棄を行うか行わないかを決定する要素は、

- パケットの転送速度

  優先度の高いパケットに適用する。優先度が高いと、そのパケットのみ転送され、優先度の低いパケットはキューにたまり、廃棄されてしまう。そのため、優先度の高いパケットに転送速度の制限をつけ、それに違反するパケットを廃棄する。
- キューの深さ

  出力側で転送速度が小さくなり、輻輳が起こるとキューにパケットが溜まってくる物理的にキューにたまるデータの量は決まっているので、キューをすべて使い果たすと入力されたパケットは、キューの入り口で廃棄される。

である。ドロッパをクラスベースの優先制御に適用する上での問題は、廃棄アルゴリズムの複雑さによる遅延である。以下に代表的なドロッパを述べる。

1. アブソリュートドロッパ（Absolute Dropper）

   EF(Expedited Forwarding) クラスのフローは、最大帯域幅が与えられ、廃棄率、遅延とも最小音クラスであるが、帯域幅を占有して、他のクラスのフローにの転送が不可能にならないように常に転送速度が監視されている。もし、このフローが契約した転送帯域幅を超えると、超えた分は無条件で廃棄される。このとき廃棄を行う要素をアブソリュートドロッパという。

2. アルゴリズミックドロッパ（Algorithmic Dropper）

   キューにデータがたまりすぎて輻輳が起きないようにするためのもので、キューにたまったデータ量に閾値をもうけ、その量を超えるとキューがあふれる前に廃棄を行う機構である（図 2.1）。

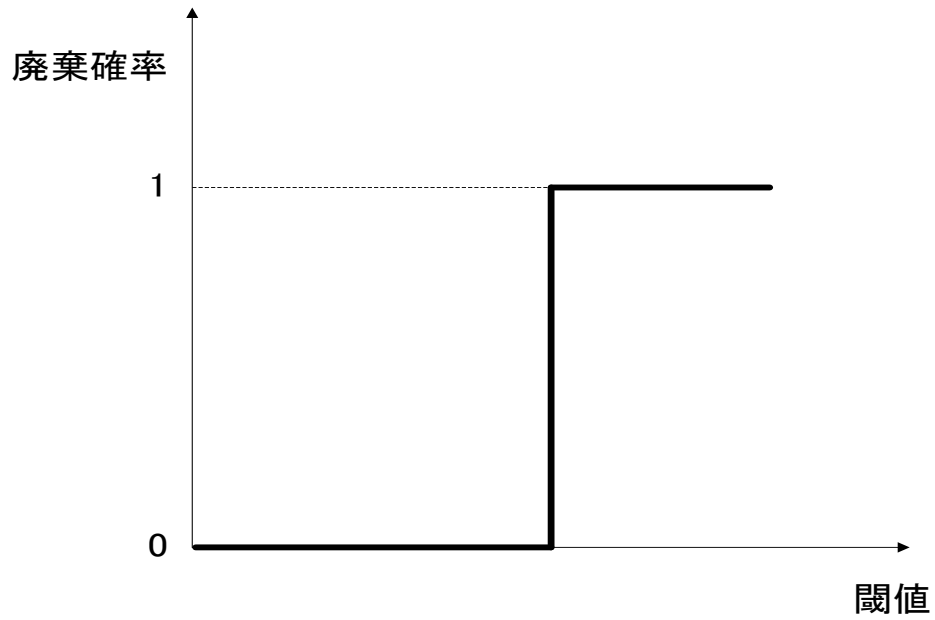

図 2.1　Algorithmic Dropper

3　RED（Random Early Detection）[4]

アルゴリズミックドロッパではキューにたまったデータ量が閾値を超えるとそれ以降入力されるパケットは 100 %廃棄されるので、廃棄された全ての TCP アプリケーションはウィンドウサイズを小さくして再送し、それによってネットワークのレスポンスが急激に悪化する。キューにデータがたまってきたときにその時のキューサイズに応じてリニアな確率で廃棄させていくと TCP 送信者には間接的に通信レートを下げるように通知することになり、輻輳が防ぐことが可能である。この廃棄アルゴリズムが RED である。

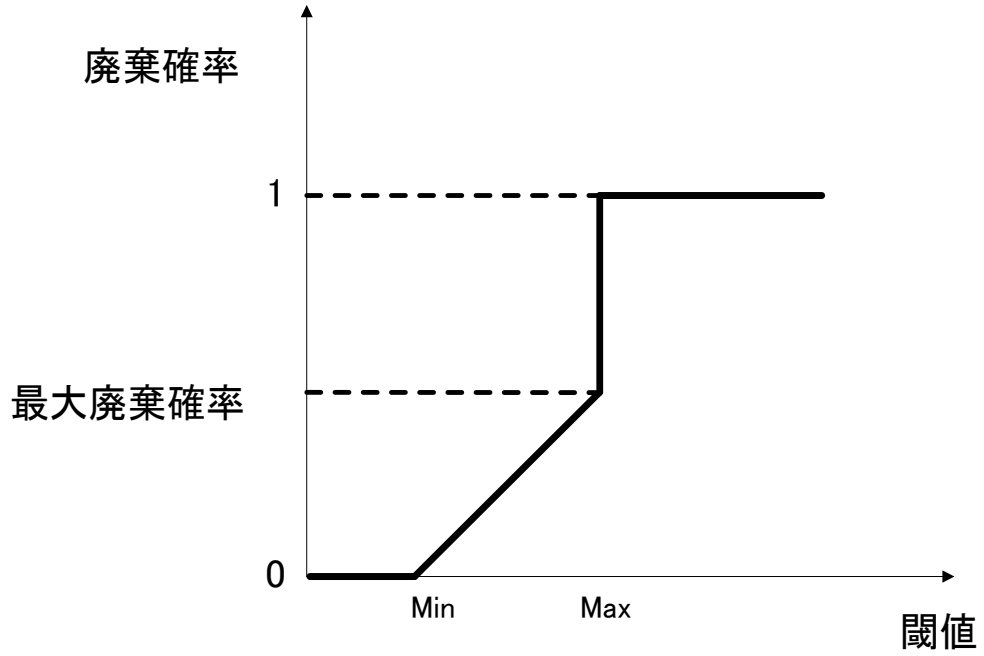

図 2.2　Random Early Detection

RED の処理について以下に述べる。

初期値として

$avg = 0$

$count = -1$

を与える。

キューにパケットが存在している間は:

avg として、平均キューサイズを計算する。

キューが空でなければ

$avg = (1 - W_q) \times avg + W_q \times Q$

キューが空ならば

$m = f(time - qtime)$

$avg = (1 - W_q)^m \times avg$

$(1 - W_q)^m \times avg = (1 - m \times W_q)avg$（$W_q$ が十分小さい場合）

もし、$Min \leq avg < Max$ なら

Count を増加

確率 P_a を計算:

$P_b = Max_p(avg - Min)/(Max - Min)$

$P_a > P_b/(1 - count \times P_b)$

ならば、到着パケットを廃棄し、このとき Count を 0 に設定

もし、$avg > Max$ ならば

到着パケットを廃棄し Count を 0 に設定

$Min > avg$ の場合は、Count を-1 に設定

キューが空になったら

$qtime = time$

このような流れで処理を行う。

各パラメータの説明

- avg:平均キューサイズ
- qtime:キューが空の状態にある時間
- count:最後にパケットが廃棄されてからのパケット数
- Wq:重み付け
- Min:最小値

- Max:最大値
- Maxp:最大廃棄確率
- Pa:パケット廃棄確率
- q:現在のキューサイズ
- time:現在の時間

2.6　優先度に応じたパケット分類

入力されたパケットは、サービスの種類ごとに求められる QoS が異なる。クラスベース QoS 制御では、転送サービスに複数のクラスを設け、ネットワーク内では、クラスごとに QoS 制御ルールに応じて転送する。そのためパケットを優先度によって分類する機構が必要である。以下にその代表的なものを示す。

- クラシファイヤ（Classifier）

  クラシファイヤは入力されたパケットを一定のルールに従って分類する。コアルータでは IP パケットヘッダ内の情報のみによってクラス分けする。クラシファイヤはルールによっていくつかの種類に分かれる。

1. BA（Behavior Aggregate）クラシファイヤ

   コアルータで、サービスクラスを IP パケットヘッダの DSCP フィールド（IPv4 ヘッダでは TOS フィールドの 8 ビットのうち上位 6 ビットを再定義）の情報のみを基にフローをクラス分けする方法。同一の DSCP を持つパケットはすべて同一の扱いを受ける。これを PHB(Pre Hop Behavior) とよび、クラスは現在、以下のように、EF、AF、CS、BE の 4 種類がある。

   (a) EF(Expedited Forwarding) クラス

   優先的なパケット処理型、送信キューに入力されるパケットの例とより出力されるパケットレートが大きい転送クラス。遅延が発生しないので、帯域幅が保証され、

ジッタが少ない専用的な（Vertual Leased Line）サービスが受けられる。ただし、フローレートが出力側の帯域幅を超えるとキューでパケットの滞留が起こるので、超えた分のパケットを廃棄する必要がある。そのため、常にメータで監視する必要がある。また、下で述べる BE(ベストエフォート) クラスと共存するためにはトラフィックの調整を行い、帯域を占有しないようにする必要がある。DSCP=101110である。

（b）AF（Assured Forwarding）クラス

パケット転送保障型、最低帯域幅が保証された転送クラス。転送レベルの高い方から、AF1、AF2、AF3、AF4 の 4 クラスに分かれる。また、ネットワークの輻輳時には廃棄が行われるが、廃棄優先度で高（レベル 3）中（レベル 2）低（レベル 1）の 3 つに分けられ、高廃棄レベルから順に廃棄が行われる。各クラスレベルと、それに対する DSCP は次の表に示す。

表 2.1　各 AF クラスの DSCP 対応表

	低廃棄レベル	中廃棄レベル	高廃棄レベル
クラス 1	001010(AF11)	001100(AF12)	001110(AF13)
クラス 2	010010(AF21)	010100(AF22)	010110(AF23)
クラス 3	011010(AF31)	011100(AF32)	011110(AF33)
クラス 4	100010(AF41)	100100(AF42)	011110(AF43)

（c）BE(ベストエフォート) クラスベストエフォート方、デフォルトのインターネット転送クラスで、パケットの転送に最善の努力はするが品質の保証はされないクラス。DSCP=000000

（d）CS(Class Selector) クラス

クラス選択型、DSCP を従来の ToS フィールドとして使う時のパケット転送クラス。ToS の優先度は上位 3 ビットで表されることから、下位 3 ビットは 0 でうめられ、他クラスと識別される。Cisco が実装している IP Precedence を使う方法と互換性がある。DSCP = XXX000 である。

2. MF（Multi Field）クラシファイア

   コアルータでは DSCP のみを基にパケットのクラスを分類するのに対して、エッジルータでは IP ヘッダの他のフィールドや TCP ヘッダあるいは UDP ヘッダの内容に基づいてフローを識別し、クラス分けを行う。IP ヘッダでは、送信元および受信側 IP アドレスおよびプロトコル番号、TCP ヘッダおよび UDP ヘッダでは、ポート番号によって識別する。

3. その他のクラシファイア

   上記のクラシファイヤでは、IP、TCP および UDP パケットヘッダに含まれる情報に基づいてパケットの分類を行ったが、下位レイヤに基づく分類も可能である。

2.7 結言

本章では、各機能モジュールについて説明した。実際に構成した機能モジュールとそれぞれのモジュールの並列性を考慮した構成や、DDMP のアーキテクチャに依存する問題への対処については 3 章で述べる。

第 3 章

クラスベース QoS 機能モジュールの構成

3.1 緒言

今回構成した、clasifier、Average Rate Meter、Algorithmic Dropper、RED(Random Early Detection) の構成および非同期型のシステムの特徴を考慮した処理内容について述べる。

3.2 Classifier

Classifier はパケットを IP ヘッダの先頭 32 ビット（表 3.1）に示される DSCP6 ビットの内上位 3 ビットに記述されている内容に基づいて、クラス分けするためのものである。

表 3.1 IP ヘッダ先頭 32 ビット

4 ビット	4 ビット	DSCP6 ビット	空きビット 2 ビット	トータルレングス 16 ビット

また、クラス分けに使ったクラスは、EF クラス、AF 1 から AF 4 クラスと BE クラスの全 6 クラスを用いた。

処理としては、DSCP の上位 3 ビットにだけマスクをかけ、その値と各クラスの DSCP 値との、比較を行い結果ごとに選択出力を行っている。

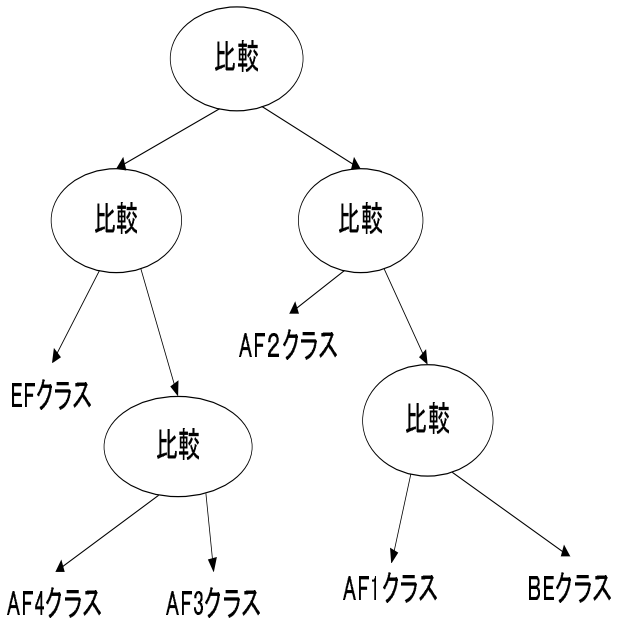

図 3.1 Classifier の構成 1

図 3.1 の構成をとることにより、全てのクラスのパケットが集中するクラシファイアの並列性を高めた。クラシファイアで、パケットの差別化を行うならば図 3.2 の構成のクラシファイアを用いるべきなのだが、今回は、処理負荷と処理速度を考慮した結果、図 3.1 の構成を用いている。

3.3 Average Rate Meter

Average Rate Meter は単位時間当たりのビットレート br を計算し、その値に基づいて、パケットをキューに入れるか廃棄するかの制御を行っている。Average Rate Meter の処理に必要な情報は、計測された単位時間 t_i と、流入してくるパケットのビット数である。単位時間は本来ならハードウェアで計測するのだが、今回はプログラムをループさせ 1 度ループするのにかかる時間をもとに経過時間を計測した。

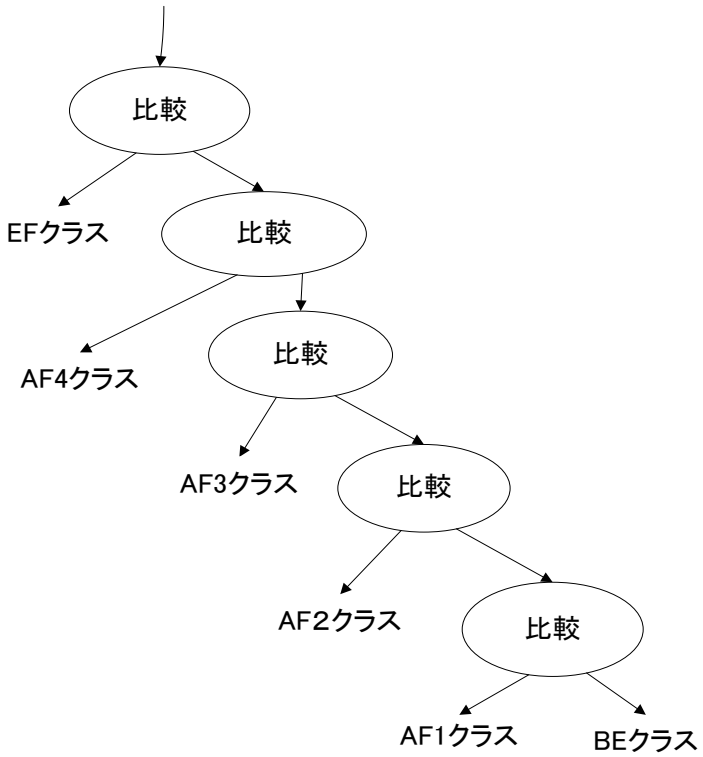

図 3.2　Classifier の構成 2

また、Average Rate 計算のために 2 つの閾値を設定する。単位時間 t と単位時間内の流入した、パケットのビット数 b を閾値とする。

1. $t_i < t$ かつ $br < b$ ならばパケットをキューに通し、単位時間とビットレートの計算を続ける。
2. $t_i < t$ かつ $br \geq b$ ならば $t_i = t$ となるまで到着するパケットをすべて廃棄する。そして、$t_i = t$ となった場合に、t_i と br の初期化を行う。
3. $t_i \geq t$ かつ $br < b$ ならば t_i および br の初期化を行う。またこの単位時間内にパケットの廃棄が起こらなかったことになる。

図 3.3 のように Average Rate Meter はビットレートを計算し、その結果に応じて比較選

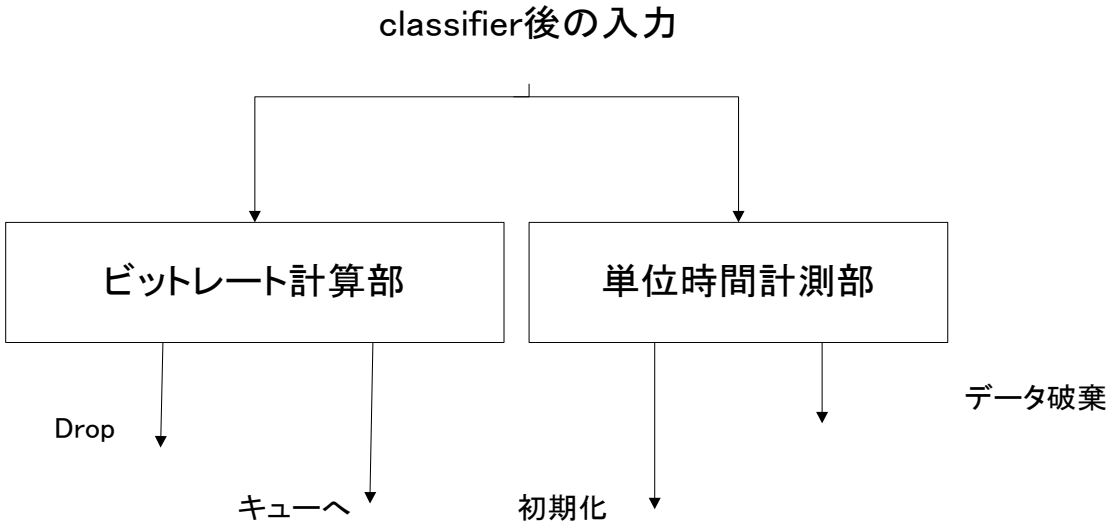

図 3.3　Average Rate Meter

択出力する処理部と、単位時間を計測し、その結果次第で、t_i および br の初期化を行うか行わないかを処理している部分である。この 2 つの処理はお互いに独立であるため、並列に処理が可能である。

また、単位時間計測部では、タイマの参照、比較、初期化の処理の流れの中で、参照と初期化のために、メモリを使用している事により、reader/writer 問題が発生した。メモリを参照し、初期化が行われるのだが、初期化が行われる前に次々とパケットは参照比較され続ける。そして、誤った値を参照したパケットは初期化の処理へ流れ続け、最初に初期化が完了する前に参照され続けたパケットは、誤った結果を持ったまま初期化し続ける。この初期化のタイムラグによる誤差は、計測を繰り返すにつれ累積する。この問題の解決のために、初期化の情報を持つフローに最初の 1 つだけを通すような構成を組み込むことによって対応している（図 3.4)。

最初の比較の部分で、単位時間と計測時間の比較が行われ、計測時間が単位時間未満ならば 0（false）が出力される。加算処理には 0 が送られ、1 を出力する部分からは 1 が出力される。単位時間を満たしていない場合は、加算からは 0 が出力され続け、1 との比較が行わ

れるので初期化が行われない。

計測時間が単位時間以上ならば、1（true）が出力される。最初に計測時間が単位時間以上となった場合は加算からは 1 が出力され、1 を出力する命令からも 1 が出力される。左右の比較の結果、等しいので初期化処理が行われるフローへとデータが流れる。

この時点ではまだ初期化は完了しておらず、この処理部の最初の比較からは 1 が出力され続ける。しかし、2 つ目以降のパケットでは、加算命令から、2 以上が出力され続け、もう一方からは 1 が出力され続け比較の結果、初期化のフローにはデータが流れることがなくなり、初期化誤差のタイムラグの解消となっている。

また、初期化処理の部分では、単位時間計測部分の初期化、ビットレート計算部分の初期化、そして、この誤差解消のために使われている、加算カウンタの初期化も行っている。

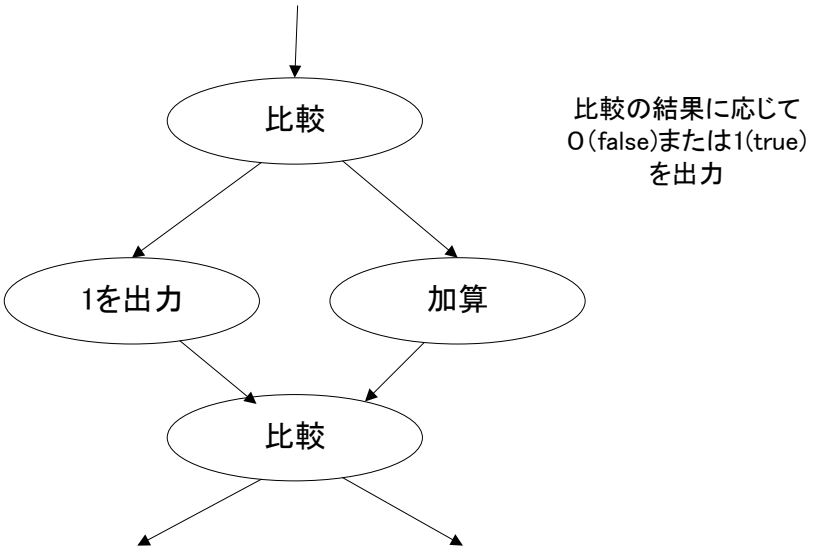

図 3.4　誤差解消構成

また、ビットレートの計算のために、除算を用いるところは、シフト命令による処理で実現した。これは、除算を用いると処理が複雑になること、また、除算をハードウェアで実現した場合に必要とされる資源などを考慮し、計測単位時間を 2^n とし n ビット右シフトを行い、処理の簡略化および Average Rate Meter の処理速度の向上を行っている。

3.4 Algorithmic Dropper

アルゴリズミックドロッパは、キュー内のトータルパケット長をもとに廃棄もしくはキューに通すかを識別しており、キューサイズを参照する必要がある。今回は、ソフトウェア処理で擬似的にキューを用意し、処理を行った。

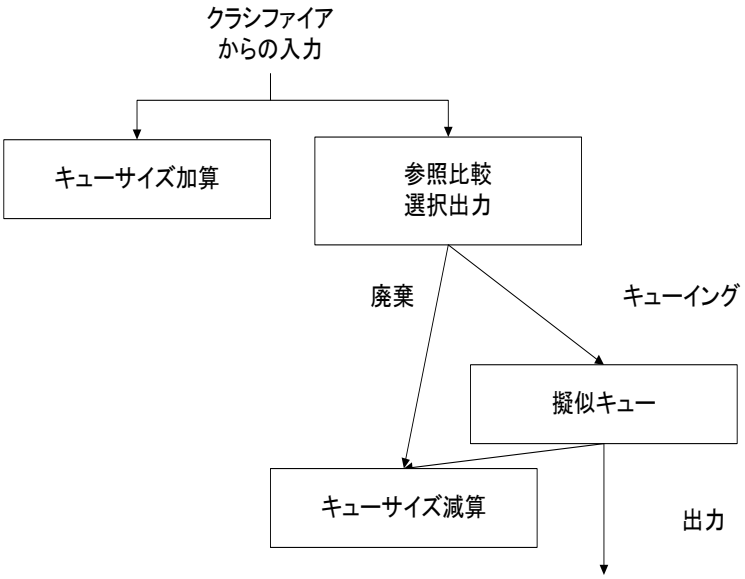

図 3.5 アルゴリズミックドロッパの構成

処理は大きく分けて、キューサイズを常に更新している部分と、キューサイズを参照し、廃棄制御をしている部分がある（図 3.3)。キューサイズ加算部では入力されたパケットの下位 16 ビットを参照し、加算更新している。また、もう一方のフローの参照比較選択出力部でキューサイズを参照し閾値との比較を行い、廃棄もしくはキューイングの制御を行っている。キューイングされるパケットは、擬似的に用意されたキューに入り出力に意図的に遅延をもうけている。プログラムから出力される場合には、キュー減算の処理を行っている。ただし、キューの管理には、メモリを使用しており、reader/writer 問題がある。起きた問題は、キューサイズの更新を複数箇所で行っていたために、参照と更新のタイムラグから正しい値に更新されずにキューサイズに誤りが生じる。この繰り返しにより、正しくキューサイズが管理されていなかった。

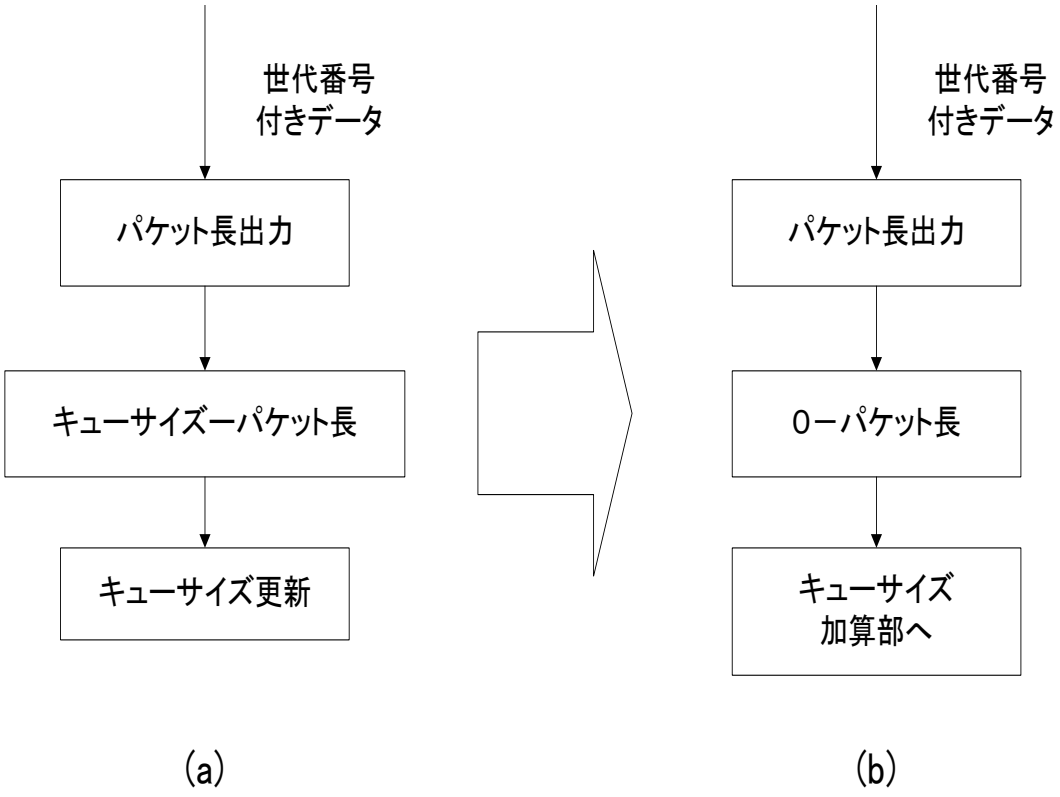

図 3.6　キューサイズ管理誤差解消構成

図 3.4 の (a) は誤差がキューサイズの管理に誤差が発生していた場合の構成である。図 3.4 の (b) との違いは、この部分でメモリの更新を行わず、減算の値を出力し、キューサイズ加算部でマイナスの値を加算していると言う点である。これにより複数箇所での更新から起こる reader/writer 問題は解消され、的確なキューの管理を用意にしている。

3.4.1　RED(Random Early Detection)

RED では、第 2 章の処理内容が示すように、avg というキューサイズと重み付けによって出された値をもとにして廃棄制御を行っている。

RED の処理では、キューが空の場合にも avg の計算をする必要がある。これは、RED に入力が開始された場合に、メインのフローとは別にもう一つ計算のためのフローを用意し、そのフローに無限ループを使い、キューサイズを参照し、キューが空の場合の計算を行うようにしている。

$P_a > P_b/(1 - count \times P_b)$ は $P_a \times (1 - count) \times P_b > P_b$ とし P_a は乱数を用いて代用している。

$avg = (1 - W_q)^m \times avg$ は W_q が 1 よりも十分小さい場合に $avg = (1 - m \times W_q) \times avg$ とすることができるため、$avg = (1 - m \times W_q) \times avg$ を計算に用いている。

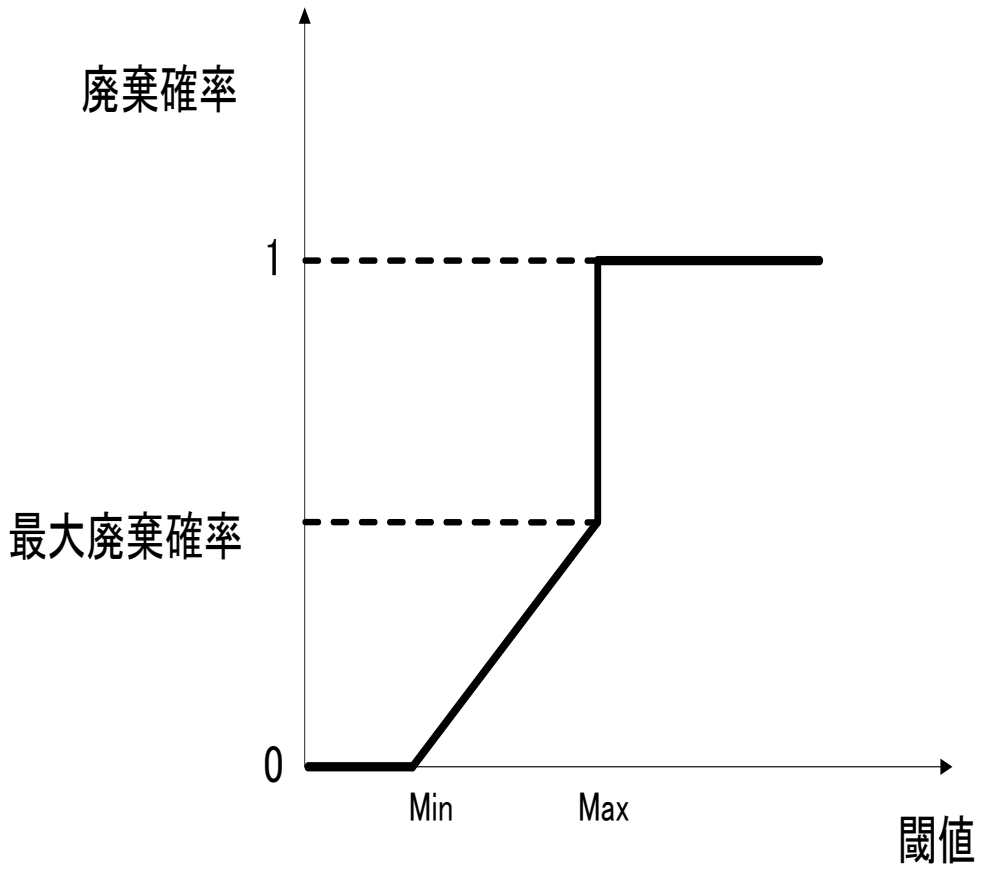

図 3.7　RED の構成

W_q は 0.01 より大きいことが望ましいとされている。今回のプログラムでは、小数点以下の数値を使用できないため、この W_q を計算に用いる処理は、一旦全ての値を 1000 倍し、計算終了後に、10 ビット右シフトすることでこの計算に対応している。そして、avg の計算

をメインのフローから切り離すことにより、RED の処理の並列性を高め、処理の高速化を図った。尚、RED のキューの管理は、先に述べたアルゴリズミックドロッパと同様の処理を行っている。

3.5　結言

本章では、それぞれの機能モジュールについて、非同期であるデータ駆動方式の処理を考慮した構成を行い、また、それに対する問題についても対処した。

次章では、並列実現をした各機能モジュールについての性能評価を行う。

第 4 章

性能評価

4.1 緒言

本章では、構成した機能モジュールの性能評価を行う。また、それぞれの評価結果に対しての考察を記述する。

4.2 評価内容

評価内容としては、Average Rate Meter、アルゴリズミックドロッパ、RED にキューを擬似的に組み込んだ上で、パケットを流し、定常状態でのスループットを計算している。評価に用いたシミュレータは、DDMP10A 用のシミュレータを用いた。

スループットは、各モジュールの定常状態においての、スループットを計測している。定常状態とは、十分にパイプライン内にパケットが充足し、パイプラインの使用率に大きな変動の無い状態を指す。なお、定常状態においても負荷が過渡にある場合は、定常状態とみなさず、ある程度揺らぎが減少してきた時点を定常状態とし、ここからスループットを計測している。

スループットの算出方法は、ある特定のパケット数 n を処理するために必要な時間を割り出している。最初のパケットから n 番目のパケットを処理するのに必要だった時間 2 番目のパケットから $n+1$ 番目のパケットを処理するために必要だった時間を順々に求め、これを定常状態が終了するまで繰り返し、その平均をスループットとしている。

4.3　評価項目

評価した項目について以下に示す。

- 各機能モジュールの単体でのスループット
- 全ての機能モジュールを使用し、複数クラスを同時に処理した場合のスループット
- 優先パケット処理

上記の項目を評価した。

4.4　各機能モジュールの単体でのスループット

単体での計測は、一定のパケット投入レートで評価している。

各機能モジュールのスループットの計測のために、パケットの入力レートは一定にして評価を行った。

バーストの条件として、一定の時間限界の入力レートよりも高いレートで入力し、入力レートに多少余裕を持たせる。これを繰り返すこととした。

4.4.1　Classifier

Classifier のスループットを表 4.1 に示す。

表 4.1　Classifier のスループット

	スループット
Classifier	143Mpps

クラシファイアは全てのクラスが通過する機能モジュールであり、高速な処理が望まれる処理である。クラシファイアへの高速化への要求は満たしたと考えている。

今後は、パイプライン間のスピード、また、プロセッサ間の通信コストの低下を考慮することで更に高速化が見込める。

4.4.2　Avarage Rate Meter

Avarage Rate Meter のスループットを表 4.2 に示す。

表 4.2　Avarage Rate Meter のスループット

	スループット
Avarage Rate Meter	11.4Mpps

Avarage Rate Meter には単位時間計測のために、ソフトウェア処理でのタイマを用意した。また、擬似的にキューを組み込み、出力に遅延をもうけた。タイマの処理とキュー処理のために Avarage Rate Meter 本来の処理のためのリソースを削減し処理を行った。

タイマ処理は本来ハードウェア処理で実行するもので、キュー処理も管理のために多少の命令を必要とすることを考えても将来的には更なる高速化が可能であると考えている。

4.4.3　Algorithmic Dropper

Algorithmic Dropper のスループットを表 4.3 に示す。

表 4.3　Algorithmic Dropper のスループット

	スループット
Algorithmic Dropper	7.3Mpps

Algorithmic Dropper の処理にはキュー内のパケットのトータルサイズが必要であり、Algorithmic Dropper 内でキュー管理をするためのリソースが必要となった。キュー管理そのものは容易となっているが、通信コストに起因する過負荷への対処が課題となっている。ただし、Algorithmic Dropper の最適化により 1.5 倍、ハードウェアモジュール化により 1.5 倍から 2 倍、また正確な動作のために必要としたリソースは今後 Algorithmic Dropper の処理に使用可能である点を含め今後性能向上する余地があると考えている。

4.4.4　RED

RED のスループットを表 4.4 に示す。

表 4.4　RED のスループット

	スループット
RED	6.4Mpps

RED は今回構成した機能モジュールで最も複雑な構成をとなっており、本来の処理以外のためのリソースも多く使用した。

今後、タイマ処理やキュー処理の部分の高速化、また、複合命令を用いた処理、最適化を行うことによる処理性能の向上が可能である。

4.5　クラスベース QoS 制御モジュールのスループット

本セクションにおいて、以下の点を評価している。

1. 周期的に各クラスにパケット投入
2. ランダムに各クラスにパケット投入
3. BE クラスに偏りのあるパケット投入
4. EF クラスに偏りのあるパケット投入

これは、1 と 2 は各クラスの占める帯域の割合が均等で、クラスごとに適用されている各機能モジュールがこの状態にある場合の処理性能を検証するためである。

3 は、現在 BE クラスのパケットが最も多いので、その状態のスループットを計測するため、また、4 は、将来 EF クラスのパケットが増加すると考えられるので、EF クラスが増加した場合のスループットを計測するためのパターンとして考えた。

この 4 パターンについてのそれぞれのスループットを表 4.5 に示す。BE クラス、EF クラスに偏りを持たせた評価は入力周期はランダムとしている。パターン 1、パターン 2 へ入

力されるパケットフローの比率はともに EF:AF1:AF2:BE=1：1：1：1 となっている。

パターン 3 については EF:AF1:AF2:BE=1：1：1：2

パターン 4 については EF:AF1:AF2:BE=2：1：1：1

パターン 3 および 4 と全体の 4 割の入力パケットが 1 クラスに入力される。

表 4.5　複数クラス同時処理の各パターンのスループット 1

	スループット
パターン 1	11.4Mpps
パターン 2	9.6Mpps
パターン 3	8.9Mpps
パターン 4	12.1Mpps

パターン 1 は負荷が均等に分散されるため、比較的高いスループットを示した。また、パターン 4 についても、ARM のスループットと他の機能モジュールの低負荷から高速処理が可能であることがわかる。パターン 2 に関しては、入力がランダムであることから、スループットの低い機能モジュールへの負荷が、パターン 4 についても、RED へのパケットの集中による高負荷のため、十分なスループットが出せない。

ただし、先に述べたように、今後それぞれの機能モジュールの高速化が期待できる。特に、Algorithmic Dropper および RED の高速化は全体の処理の向上につながり、スループットの均一化により、負荷が均等に分散され更なる高速化が可能となる。

トラフィックの平均パケット長が 2000 ビットであることから、ビットレートを算出すると。以下のようになる（表 4.6)。

表 4.6　複数クラス同時処理の各パターンのスループット 2

	スループット
パターン 1	22.8Gbps
パターン 2	19.2Gbps
パターン 3	17.8Gbps
パターン 4	24.2Gbps

4.6　結言

本章では各機能モジュールのスループット及び複数のクラスを並列に処理する際のスループットについて述べた。廃棄制御の Algorithmic Dropper と RED のボトルネックの解消が課題となる。

第 5 章

結論

本研究では、近年のネットワークの急速な進歩に伴い高まっている、高品質なネットワークの要求へのアプローチとしてクラスベース QoS 制御に着目した。クラスベース QoS には複雑かつ柔軟な処理が要求され、盛んに研究が行われているが、マルチプロセッサスケジューリングオーバーヘッドによる問題がある。そこで、スケジューリング不要でプログラマぶるな DDMP を用いて、高速かつ柔軟なクラスベース QoS の並列実現を目的とした。

クラスベース QoS の機能モジュールとして、パケット分類の Classifier、帯域保証制御の Average Rate Meter、廃棄保証制御の Algorithmic Dropper と RED をそれぞれ構成した。

各データフロー共通のメモリ処理においての reader/writer 問題の解消や、世代番号を用いたデータの管理、除算処理の簡略化により資源及び処理時間の低減等の処理を行うことで、並列処理に適したクラスベース QoS 制御モジュールを並列実現している。

処理性能としては、Algorithmic Dropper はその構造と通信コストの問題から、また RED は、処理の複雑さからボトルネックとなっている。複数クラスの同時処理の場合も、ボトルネックとなっている RED が過負荷になる場合には十分なスループットを発揮できていないが、負荷が Average Rate Meter に集中する場合はスループットを発揮できる。

今回クラスベース QoS の並列実現を行い、使用者のポリシーに応じてパラメータ変更が可能な構成をとり、高速かつ柔軟なクラスベース QoS 制御が実現可能であることについて示した。

今後の課題としては、ボトルネックとなっている、Algorithmic Dropper と RED の最適化、複合命令の使用等により、高速化を図ることが挙げられる。

謝辞

本研究において、大変お忙しい中、懇切丁寧にまた、時には厳しくご指導いただきました岩田 誠教授に深く感謝いたします。

情報システム工学科長 島村 和典教授には副査を務めていただき、また日頃からお気遣い、心より感謝しております。

情報システム工学科 菊池豊 助教授 副査を務めていただき、また貴重な御助言をいただき、大変感謝いたします。

大学院博士課程の林 秀樹氏にはご多忙の中、貴重な時間を割いていただき、価値ある情報を提供していただくとともに、研究にも多大な影響を与えていただき、感謝いたします。修士課年の森川 大智氏には、自身の研究でご多忙の中、御助言いただきまして感謝しております。修士課程の橋本　正和氏、別役　宣奉氏におかれましては、日頃より暖かく見守っていただきまして、感謝したします。

修士課程志摩　浩氏には日頃よりの御助言感謝いたします。大学院修士課程小倉　通寛氏、三宮　秀次氏、中村　勲二氏には様々な面でお世話になり、お礼を申し上げます。

最後になりましたが学部同期の岩田研究室の皆様のおかげで、大変よい大学生活を送ることができました。ここでお礼を申し上げます。

参考文献

[1] H.Hayashi, M.Iwata, H.Terada,”An Autonomous Class-Based QoS Control Utilizing a Self-Timed Folded Pipeline,”in Proc. 4th ICACT, 469-475, 2002.

[2] H.Terada, et al., ”DDMP’s: self-timed super-pilelined data-driven multimedia processors,” Proc. of IEEE, 87(2), 282-296（1999).

[3] 細美 俊彦,”自己同期パイプラインによるクラスベース QoS 制御機構,”平成 13 年度修士論文, 2001 年 12 月

[4] S. Floyd and V. Jacobson, . “ Random Early Detection Gateways for Congestion Avoidance, ” IEEE/ACM Trans. Netw., vol.1 no.4, pp.397-413, August 1993.