平成 13 年度

修士学位論文

自己同期パイプラインによる クラスベース QoS 制御機構

A Self-Timed Pipeline Implementation of Class Based QoS Control Mechanisms

1045024　　細美 俊彦

指導教員　　寺田 浩詔

2001 年 12 月 28 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報通信ネットワークコース

要 旨

自己同期パイプラインによるクラスベース QoS 制御機構

細美 俊彦

光伝送技術の発展により，あらゆる形式の情報を自由に疎通できる情報環境の実現がまさに可能になろうとしている．このような情報ネットワーク上で多様な新サービスを柔軟に実現するためには，多様なパケット流を受容し，それぞれのリンクの実速度に対応した処理能力を持つ超高速プログラマブルルータチップの実現が中心的な課題となる．

本研究では各サービスや各フローに応じた柔軟なルーティングを可能とするチップの実現に必須となる，超高速 QoS 制御方式を提案している．本方式では，従来は機能的に制限が強いとされてきたハードウェア・キュー機構の実現法に，折り返し型自己同期パイプライン機構を導入することによって，各パケットが自律的に自身の優先度に応じて，待ち行列を形成できる自律型優先キュー機構を実現している．さらに，この機構を活用して，Diffserv 等のクラスベース QoS 制御をポリシーに応じてソフトウェア定義可能な QoS 制御向きデータ駆動型マルチプロセッサを構成する．提案方式をシミュレーション評価した結果，キュー機構内の折り返し点をクラス毎に調節して設定することによって，ネットワーク間の QoS ポリシー (SLA) 等に応じたクラス別 QoS 制御 (帯域制御，遅延時間/ジッタ制御，廃棄制御) が，広い範囲で実現できることを確認した．さらに，提案キュー機構を搭載したマルチプロセッサ上で典型的な Diffserv スクリプトを実行した場合の処理性能を見積もった結果，平均約 59Gbps の IPv4 パケットを処理可能なことを確認した．今後，提案方式の有効活用のために，パケットの内容に従った高速パケット分類方式の検討が残されている．

キーワード　QoS，自律型優先キュー，データ駆動，折り返し型自己同期パイプライン，SLA

Abstract

A Self-Timed Pipeline Implementation of Class Based QoS Control Mechanisms

Toshihiko Hosomi

With the advancement of the photonic transmission technology, there is about to realize true information communication that allows us to transfer all kind of information via networks freely. To provide many kind of flexible services in these information networks, the most important thing is to realize an ultra high-speed programmable router chip that can accept multiple packet-streams at speed of each optical link.

This paper proposes an ultra high-speed QoS control scheme per service or per flow which will be an essential function to implement the router chip that have flexible forwarding capability. In this scheme, there can realize an autonomous queuing mechanism implemented by a self-timed folded pipeline circuit although conventional synchronous circuit implementation of priority queuing was limited in function. Furthermore, this paper proposes a configuration scheme for a high-performance data-driven multiprocessor chip employing the proposed queuing mechanism and a novel instruction set dedicated to class based QoS control functions such as bandwidth cintrol, delay or jitter control and rejection control. Simulation results indicate that the proposed queuing mechanism can flexibly and efficiently differentiate mixed IP datagrams in accordance with their own priority of the class even if the QoS policy is decided through service level agreement (SLA) between network carriers. Furthermore, estimation results show the proposed multiprocessor can control the typical mixed IP packet streams at about 59 G b/s. Remaining future work will be to establish a high-performance packet classification scheme.

key words QoS, autonomous priority queue, data-driven, folded self-timed pipeline, SLA

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展により，ネットワークの帯域幅の拡大が著しく，あらゆる形式の情報を，必要とする時に，必要とする形で，必要とする品質で伝達しあう情報環境の実現が目前まで来ている [1]．現在のインターネット利用の主な目的は，情報検索サービス，ソフトウェアダウンロードあるいはニュースや企業・製品情報の閲覧などであるが，将来は，インターネット放送，音楽配信あるいはオンラインゲームのようなリアルタイム系のコンテンツの割合が増加してくると思われる．ネットワークでデータを転送するとき，送る側はできるだけ良い状態の情報を送りたい．ここでいう良い状態とは，たとえば画像ならばできるだけ大きい解像度であり，音声ならばできるだけ広い周波数帯域である．それに対して，データの受け手は，送る側の良い状態に加えてストレスのない情報を要求する．つまり，テレビの衛星生中継で会話をするときのように話し手と聞き手の間のタイムラグや画像・音声の切れ・飛びといった視聴していてストレスを感じるようなことがない情報を要求するのである．インターネットは不特定多数のデータの中から自分に必要な情報を検索し，選び出すことのできる学習の場でもある．そのような場でストレスを感じるほどのネットワークならばうまく情報を得ることもできないし，学習の効果もあがらない．本研究は，通信時に感じるこのようなストレスをどうすれば解決できるのか，そして多様な品質を最も効率的に実現する方法を探り，その実現方法を提案するためにおこなわれた．

インターネットをサービスの変遷で見てみると，まず，第一世代として，インターネットの黎明期，1960 年代から 1993 年のウェブブラウザの Mozaic が発表され，1994 年に Netscape 社が誕生するころまでの，主に学術研究部門でファイル転送や e-mail に使われた

時代，そして，インターネットにつながることが前提となり，Web，インターネット電話，電子商取引 (electric commerse) の行われる第二世代，そして，同じインターネットに接続していても，携帯電話のメールや i モードのように，もはやネットワークを意識せずにインターネットにつながっている第三世代では，いろいろなコンテンツ配信，インターネット越しの対戦ゲームや遠く離れた場所から家電制御が可能になった．

接続する端末も第一世代ではワークステーションやパーソナルコンピュータ，第二世代ではモバイル端末，そして第三世代では情報家電，ゲーム端末，あるいは自動車などあらゆる電化製品にひろがった．一方，インターネットに接続するアクセス網についても第一世代では電話網による接続，第二世代では新たに CATV，携帯電話，PHS が加わり，第 3 世代では IMT-2000 や光ファイバで接続するようになった．

数年前までは大部分のエンドユーザーは 28.8kbps (bit / 秒) の帯域幅のアナログモデムで接続していたにすぎなかったが，ISDN の出現で 64kbps に高速化された．2000 年代に入ると，ケーブルネットワーク，ADSL (Asymmetric Digital Subscriber Line) などの普及により，1Mbps を越える帯域幅が利用可能になった．バックボーンにおいても，10Gbps の帯域幅がすでに実現され，さらに広帯域化が進められている．LAN の環境においても．100Mbps は常識で，1Gbps も利用できるようになり，10Gigabit イーサネットの標準化もおこなわれている．このような回線の高速・広帯域化によって，従来使い物にならなかった多様なサービスが可能になろうとしている．たとえば，インターネット電話では，遅延時間をなるべく小さくし，ジッタを最小にする必要がある．また，電子商取引で多用されるトランザクション処理では廃棄を最小にする必要があり，リアルタイムストリーミングでは，常に一定量のデータを受信する必要があるため，十分な帯域幅を常に確保する必要があるほかに，遅延，ジッタが発生しないことが求められる．また，情報家電ばかりではなく，膨大な数の電化機器がインターネットに接続し，通信を行なうようになれば，通信頻度も爆発的に増加する．

100Gbps 以上のデータストリームが伝送される新世代の光通信網に完全に対応できるプログラム可能な高速パケットルータを開発する必要性が高まってきており，現にいくつかの

プログラム可能なネットワークプロセッサが開発されつつある [2]．それらのプロセッサは RISC (Reduced Instruction Set Computer) エンジンとアドレスルックアップや QoS 処理を実現するいくつかの専用ハードウェアモジュールからなるマルチプロセッサ構成になっている．しかしながら，これらのプロセッサの性能はマルチプロセッサスケジューリングのオーバーヘッドやプロセッサ間の通信遅延に制約される中央集中的な処理アーキテクチャのために低下させられている．

本研究の最終目的は，データ駆動アーキテクチャの高速ストリーム処理能力と，DDMP (Data Driven Multimedia Processor)[3] の自己同期型スーパーパイプライン機構の柔軟な省電力能力を活用した，これまでになかったネットワークプロセッサを開発することである．これまでに，複数プロトコルが混在するパケットフローの処理 [4]，さらにアドレスルックアップ高速処理の実現の研究 [5] が行われた．

インターネットが誕生してから現在まで，限られた帯域幅のもと，パケット廃棄率，転送遅延，転送時間のジッタなどの通信の品質については最大限の努力はするが保証はしないというベストエフォート型をベースにしてきた．通信の量に対して帯域幅が十分に確保されている状態ならベストエフォート型の通信でも問題はないが，現在はもちろん，近い将来，ネットワークの隅々まで帯域幅が広がれば，リアルタイムで広帯域幅の通信の要求が現在よりさらに高まるであろうし，また，すべての電化製品がネットワークにつながり，通信を行なうようになれば，ネットワーク上にありとあらゆる品質要求を持つ無数のフローがあふれるであろう．

そこで，それぞれの通信に対して要求された品質を確保するための技術が必要になる．この技術は現在，2 つに大別できる．一つはそのアプリケーションが必要とする帯域幅を前もって予約 (帯域予約) しておくギャランティー型の技術で QoS(Quality of Service:サービスの品質) とよばれている．もう一つの方法はアプリケーションごとに転送の優先度をつけておいて，その順位に従って転送するという方法で，このような技術を Cisco 社では CoS(Class of Service:サービスクラス) と呼んでいる．ここでは，一般的な名称である QoS に統一して呼ぶ．

両者とも品質の制御をネットワーク上のルータで設定・実行するものであるが，前者はIETF(Internet Engineering Task Force) によって Intserv(Integrated Services:統合サービス)[6] として標準化されている．Intserv では，送信元のノードと受信先のノードの間に存在するルータの間で通信を行い，転送するデータが必要とする帯域幅と転送経路をあらかじめ予約しておき，転送中に帯域幅が十分に取れなくなると転送経路を動的に変更しながら帯域幅を確保し通信品質を保つ方式である．Intserv は RSVP (Resource Reservation Protocol) によって動作するが，RSVP は図 1.1 のようなユーザアプリケーションが必要とする帯域幅やバッファなどのネットワークリソースを動的に確保するための制御用プロトコルである [7].

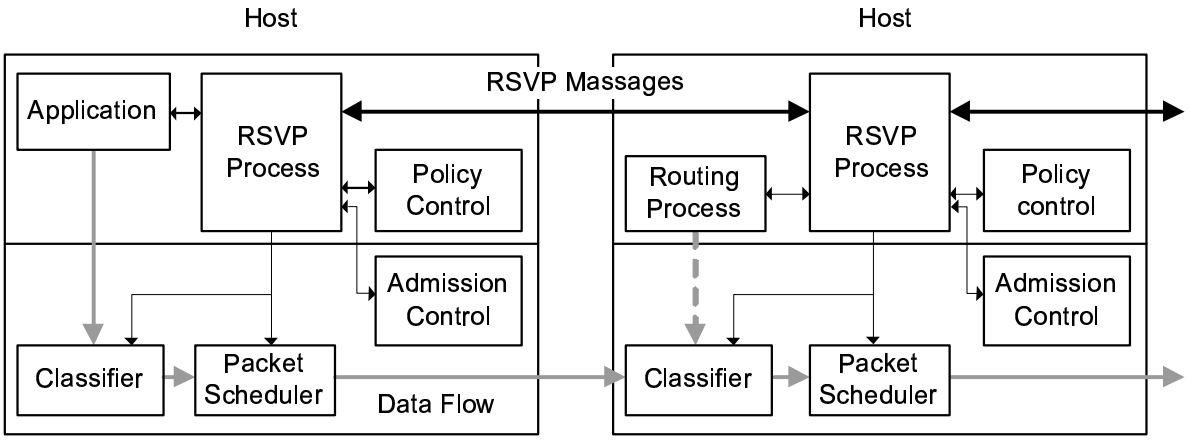

図 1.1　RSVP の構造

RSVP では，図 1.2 のように送信ホストは，まず，受信ホストがデータを転送するルート上においてリソース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ホスト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，この時，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のパスに沿って「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受信ホストは，送信ホスト方向に「RSVP 予約メッセージ」を

送信する．このとき，「RSVP 予約メッセージ」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．

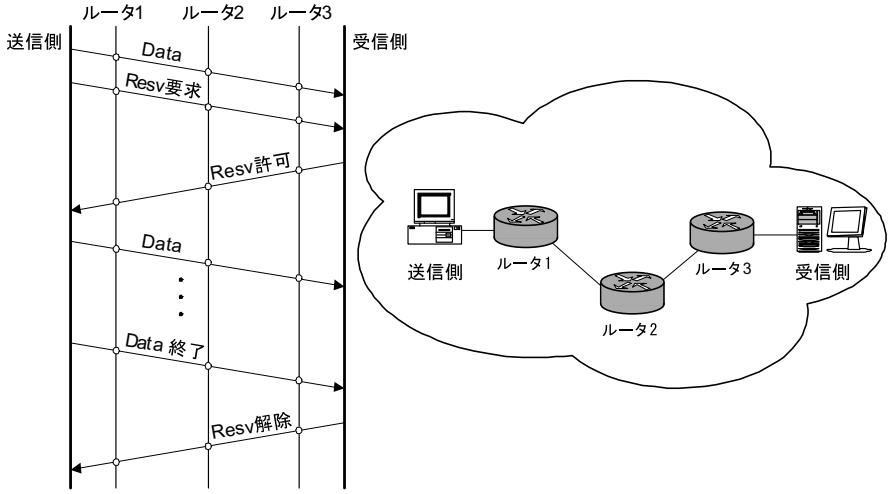

図 1.2　RSVP メッセージ

以上のように，Intserv では実際に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 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 RESV メッセージが流れる．また，予約確立後も，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の少ない，限られた範囲でのみ有効で，インターネットには不向きであることを表している．

それに対して，後者はネットワークの転送サービスに複数のクラスを設け，帯域を共有しながらデータを転送するベストエフォート (best effort) 型の QoS 制御方式である．この制御方式では，同一のクラスに属するフローは，同じネットワーク内では同一のルール (帯域幅，廃棄機構，遅延制御など) によってパケットを転送する．この方式も IETF によって標準化されており，Diffserv (Differentiated Services) とよばれる [8].

Diffserv 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，あらかじめ決められたルールでクラス分け (Classify) し，クラスに対応した DSCP 値*1[9] を図 1.4 の IP パケットヘッダにセットするエッジ (境界) ルータと，パケットヘッダの DSCP 値に基づいて，あらかじめ決められたルールで転送処理をするコアルータに分けられる．

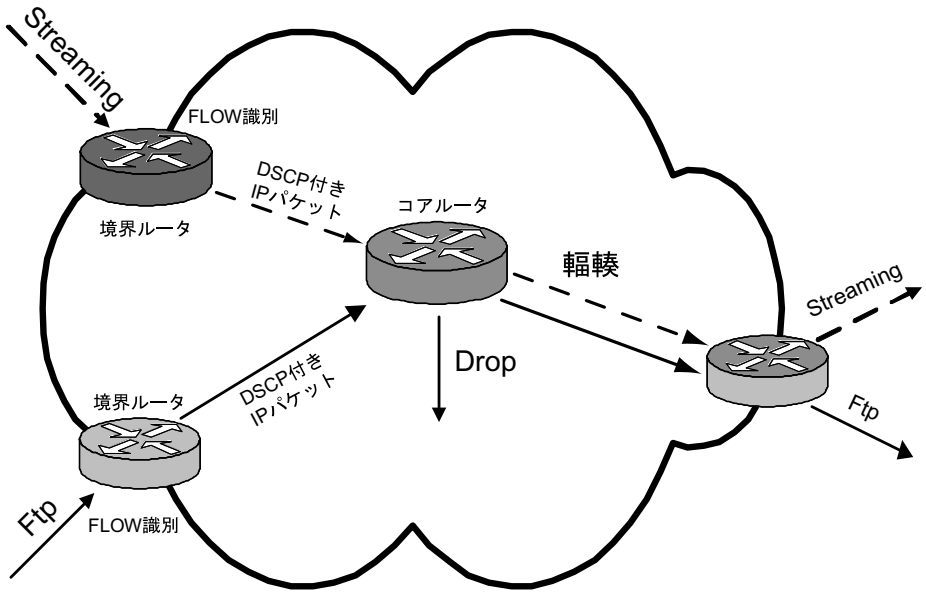

図 1.3　DS ドメイン

境界ルータでは，フローを識別し，クラス分けする作業や，DSCP 値をセットする作業が行われるが，フローの量が少ないためあまり広帯域である必要はない．一方，コアルータでは境界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，複雑な処理は必要ないので，高速化が可能である．また，ルータが行なうパケットの転送処理はすべて IP ヘッダ内の DSCP に基づいて行われるので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．そのため，Diffserv は Intserv に比べスケーラビリティがあり，インターネットで QoS を実現する方式に適している (図 1.3)．

しかしながら，Diffserv においては，コアルータに入ってきたフローをクラシファイヤ

*1 TOS(Type Of Service) フィールドを再定義

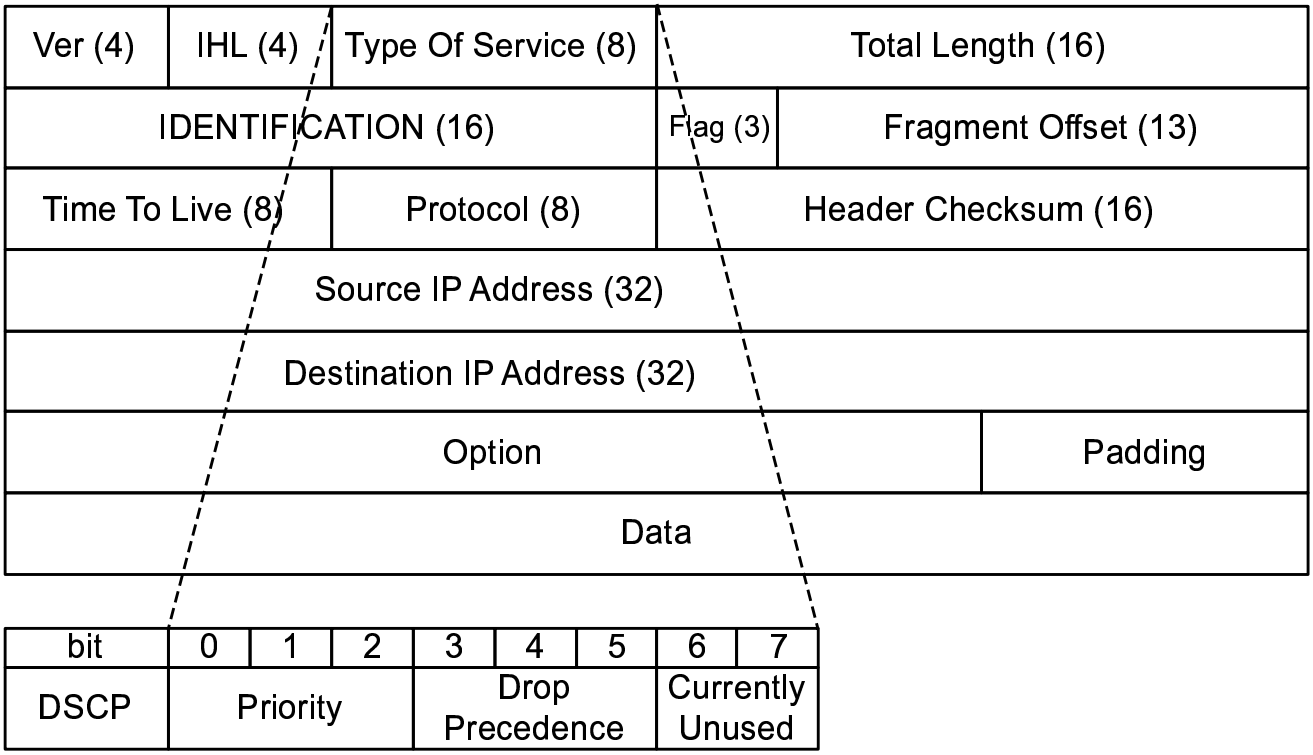

図 1.4　IP ヘッダ

(Classifier) でクラス分けし，クラスごとの優先度に応じて，出力するパケットの順序を変える必要がある．そのため，出口において出力の順番待ちを行なう必要があり，クラスの数だけのキューが必要である．また，廃棄率，遅延時間あるはジッタのような QoS 要素を的確に管理して，その上，柔軟な優先処理を行なうためにはプログラム中のスクリプトによる細かなパラメータの調整が必要であり，処理の高速化が難しい．そのため，非常に高速の回線速度のネットワークでは，優先制御およびキュー管理がボトルネックになり，Diffserv は低速の回線速度のネットワークのみで有効であるとされていた．また，処理の高速化を図るためには，優先制御やキュー管理を ASIC でハードウェア化して高速処理を行なうことが考えられるが，プロセッサの開発コストに加えて，ハードウェア処理では細かなパラメータの調節ができず，柔軟な優先処理を行なうことが困難である．

本研究では，上記のような問題点を解決するために，ルータに QoS 制御処理を実装する方式として，クラスベースの QoS 制御機構を DDMP の自己同期パイプラインによって実現

させることを目指している．その際，QoS 制御のボトルネックとなる優先処理およびキュー管理を高速，高性能化するために，自己同期パイプラインの一つの適用形態である折り返し型パイプラインを適用した新しい方式である“自律型優先キュー”を提案し，その有効性をシミュレーション実験で実証する．同時に ISP (Internet Service Provider) 間の QoS ポリシーの調停を行なう SLA (Service Level Agreement) に基づく QoS 制御が自己同期パイプラインのパラメータの調整によって柔軟に実現できることを示す．

この折り返し型自己同期パイプラインは，クロック信号を必要としないために待ち合わせ時に電力消費量が極小になり，省電力である．さらに，クロック同期回路では実現できないパイプライン中でのパケットデータの追い越しをパケットデータ自身が持つ優先度をもとに行なうことができ，優先処理に伴う遅延時間を極小化できる．

また，既存の命令を QoS 処理用に複合命令化し，ルータの各機能要素を DDMP のプログラム可能なナノプロセッサに割り当てることで，ナノプロセッサの構成を変えるだけでルータの用途に対応できるようになり，処理に伴う遅延を局所化しスループットに影響を与えないとともに，構成の変更に柔軟性が高くなる．本研究では複合命令，ハードウェアサポートおよび新命令を提案する．

本論文の第 2 章において，クラスベースの QoS 制御の概要およびその問題点について述べる．第 3 章では，クラスベースの優先制御への DDMP の自己同期パイプライン方式の適用方法について述べる．第 4 章では DDMP マルチプロセッサをクラスベースの優先制御に適用するための構成方法ならびにナノプロセッサ上にプログラムするデータ駆動型ソフトウェア (複合命令) の提案を行なう．第 5 章では自律型優先キュー方式の従来方式との性能の比較，ISP 間の SLA を確立するためのパラメータの設定に関する評価，DDMP マルチプロセッサ複合命令に関する評価および総合性能の評価を行なう．そして第 6 章では本研究の成果，今後の課題を考察する．

第 2 章

クラスベース QoS 制御の課題

2.1 緒言

ネットワークの帯域幅の拡大が著しい現在，大きな帯域幅，小さな遅延時間そして安定したジッタを要求するリアルタイムアプリケーションがそれまでのデータ転送に代わってネットワーク上の主流になりつつある．表 2.1 は通信主体とそれに対応するネットワークアプリケーションの種類を表したものである．この表を見てわかることは，まずリアルタイム型ア

表 2.1　通信主体とアプリケーションの種類

通信主体	人	機械
リアルタイム型	インターネット電話 ストリーミング デスクトップ会議	自動監視システム ロボット制御
非リアルタイム型	電子メール Web ページ閲覧	ファイルのダウンロード 自動巡回ソフト 蓄積型コンテンツ

プリケーションは，できるだけ遅延およびジッタの少ない，安定した通信が常に要求されるが，非リアルタイム型アプリケーションはデータの正確さが問題であって，少々の遅延やジッタは問題にならないということである．通信主体で見ると，人が主体の場合は，リアルタイム系アプリケーションはサイズの大きいデータを扱うので，広帯域幅が要求されるが人間の知覚と思考のタイムラグの存在から，遅延はまだ許される．一方，機械が主体の場合は，一度に送信するデータサイズは大きくないので，帯域幅はそれほど必要ないが，例えば

オートメーション工程を制御する場合に，わずかの遅延も事故につながることから遅延およびジッタに対する要求は厳しい．非リアルタイム系では機械が主体の場合は，サイズの大きなデータを定期的にゆっくり収集するのに適しており，人が主体の場合は小さいサイズのデータをランダムに送信する．本章では，このようにバラエティーに富んだユーザ (人および機械) の QoS 要求を実現するために，ルータに必要な構成要素を QoS の要素毎に述べ，さらに高速ネットワークでクラスベース QoS をルータに実装するための課題を述べる．

2.2　クラスベース QoS 制御の概要

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，それまで問題にならなかったネットワーク内のあらゆる遅延が記憶機能として無視できない存在となる．したがって，特に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にの分散的処理システムの実現が重要な課題になる．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．

クラスベース QoS とは，ネットワークにおける転送サービスに複数のクラスを設け，ネットワークに入ってくるフローをネットワークの入り口でクラスに分類し，クラスごとにあらかじめ決められたバッファや帯域幅を割り当ててパケットを転送する方式である．その際，個々のアプリケーションのフローに帯域幅等を割り当てるのではなく，個々のアプリケーションのフローを集約したクラスに対して QoS を制御する方式である．

また，ルータがクラスベースの QoS に対応するためには，それぞれのクラスに対してユーザが要求する帯域幅，許容する遅延時間およびジッタを設定する必要がある．しかし，ユーザによってクラスに対して要求する品質はまちまちであり，同じクラスでも，例えばあるドメインの中で取り決められた遅延時間の値と別のドメインの中で取り決められた遅延時間の値には違いがある．そのため，インターネットでの QoS の確保には，ドメイン間でポリシー情報を含む SLA(Service Level Agreement) を交換する必要がある．以上のような機能

および動作をまとめたものが図 2.1 である [10].

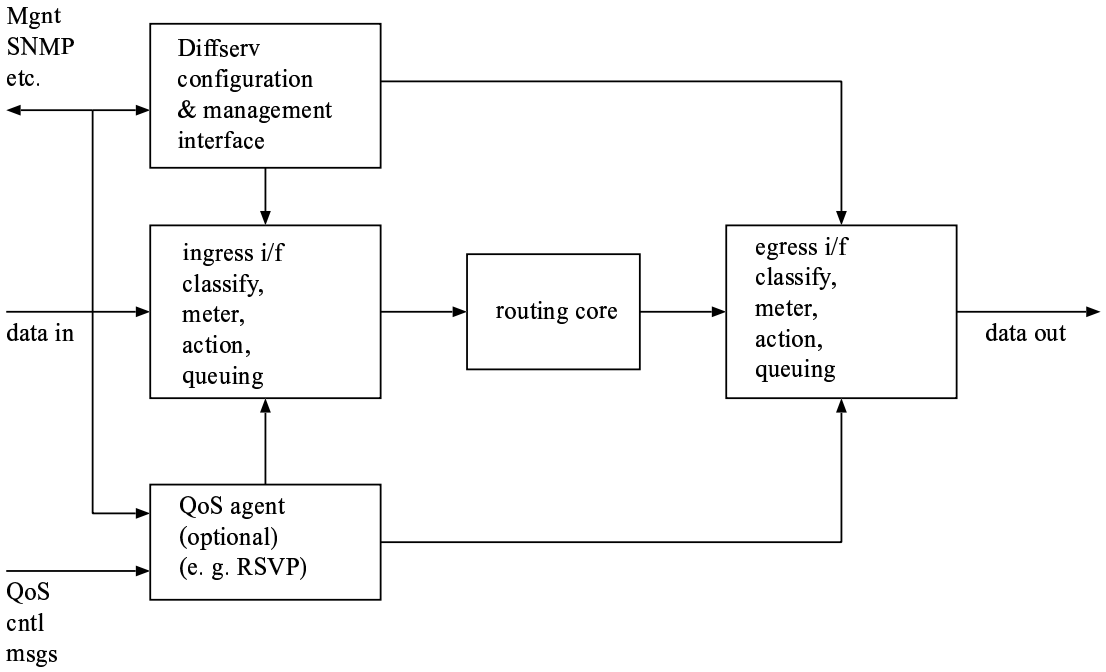

図 2.1　Diffserv ルータの構成

したがって，クラスベースの 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次にあげるような機能が必要である．

1. 帯域保障制御機能
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なく転送するためには，十分な帯域幅が必要である．そのため，ルータにも大量のパケットを高速で処理することが求められる．しかしながら，同じ転送路を共有して多くの種類のフローが転送されるインターネットでは基本的にベストエフォート型の転送方式がとられているので，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，帯域幅を占有してしまい，他のフローを転送できない場合が考えられる．そのため，優先度の高いフローの帯域幅を常に監視する必要がある．

2.3.1　メータ (Meter)

メータは転送路上を単位時間当たりに流れるデータの量を計算する構成要素である [8]．クラスに分けられたパケットはクラス毎にメータに入り，計算した結果と設計値を比較する．設計に違反していれば優先度を下げ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過しなければ設計を満たしているので，パケットは次の段階に進める．しかし，設定された量を超過すると，そのフローは設計を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．なお，廃棄するとき，課金その他の取り決めがある場合は報告する必要があるので，カウンタで廃棄量をカウントする必要がある [26]．メータに必要なパラメータは，次の 2 つである．

   - AverageRate(設定平均レート　例:120kbps)
   - Delta(単位時間　例:100msec)

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を決定するメータである．

   そのパラメータは，

   (a) AverageRate　例:25kbps

   (b) Delta　例:10μsec

   (c) Gain(直前の平均転送速度に乗ずる重み　例:$\frac{1}{16}$)

で，式は 2.1 になる．

$$\begin{aligned} \text{avg\_rate}(t) &= (1 - \text{Gain}) \times \text{avg\_rate}(t') + \text{Gain} \times \text{rate}(t) \\ t &= t' + \text{Delta} \end{aligned} \tag{2.1}$$

3. TB(Token Bucket) メータ

   上に述べたメータより，さら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中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 Backet を説明する [11].

   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っている．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すことができる．これを厳格な適合 (strict conformance) という．一方，トークンの残りが L 未満のときには，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 (loose 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．

2.3.2 マーカ (Marker)

クラス分けされたパケットの DSCP に，提供するサービスに 1 対 1 に対応するクラスの DSCP の値 (2.6 節参照) を書き込む構成要素．この値は入力側の境界ルータで書き込まれ，ネットワーク内でのみ通用する．この構成要素は DSCP マーカともいわれる．

2.3.3 マルチプレクサ (Multiplexor)

クラス分けされ，その後メータ，マーカなどで必要な処理が行なわれたフローを多重化してまとめる構成要素.

2.3.4 カウンタ (Counter)

パケットを処理したとき，そのバイト数をカウントする構成要素．この統計値は顧客への課金，サービスの検証あるいはネットワークエンジニアリングのために使われる．具体的には，アブソリュートドロッパで廃棄するときや，ある機能要素から別の機能要素にパケットが移るときにカウントされる．

2.3.5 SLA(Service Level Agreement)

ある単一のドメイン内での QoS は，データがネットワーク内に入ってきたときに境界ルータでマークされた DSCP に基づいてコアルータで転送処理を行なうことで実現できるが，複数のドメインをまたいで転送されるフローの QoS をドメイン間で共有しようとすると，特定の DSCP に対応する QoS を ISP あるいはドメイン間で取り決める必要がある．この取り決めを SLA という．たとえば，DSCP が 001010(AF11) なら輻輳時のパケット通過率を何%確保するか，遅延時間は何 μs までなら許せるか，あるいはジッタはどれくらいまでなら許せるかなどについて細かく取り決める．通常 SLA はネットワークの品質，サービスの品質あるいは顧客ケア品質などがあるが，QoS に関することは，例えば，UUNET[12] では，WorldCom のネットワーク内での往復遅延時間が北米内では 65ms 以内，ニューヨーク，ロンドン間で 120ms 以内，また平均パケット転送成功率が 99%以上，そして可用性が 100%となっている．

Intserv のような帯域予約型の QoS 制御方式では予約をするたびにフローごとに転送路上のすべてのルータがポリシールールをポリシーデータベースに問い合わせる必要があり，管理が非常に難しく，SLA を取り決めるのは困難である．一方，Diffserv では，どのクラスの場合はどのようなポリシーで転送するかを事前にルータに設定しておくので，ドメイン間でもルータの設定を共通にしておけば SLA の実現が可能である．

2.4 遅延時間制御

ネットワークの遅延は，次の 4 種類に大別される．

1. 伝送路を構成する物質の物理的な限界により生じる遅延
2. 伝送路での輻輳による遅延
3. ネットワークノード内のキューイング遅延
4. ネットワークノード内のソフトウェア処理によって生じる遅延

物理的な遅延は伝送路の長さと媒体の物理的な性質 (光ファイバや銅線中の光や電子の伝播速度) がわ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．それに対し，輻輳による回線上の遅延時間は予測不可能であり，その上，伝送路の帯域幅が非常に大きくなって来たため，伝送路は単なる線ではなく，メモリとしての記憶効果をもつようになり，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だろう．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，順番がくれば出力される．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なうバッファ管理であるが，本研究

では優先制御にキューエレメントを積極的に活用する．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，キューの入り口までやってきたパケットを順番に記憶し，出力時には，入ってきた順番で出力する．そのため，単独では優先制御を行なうことができない [10].

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なう方式，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なう必要があり，従来方式のキューでは大きな遅延を生じる [13].

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エレメント．出力するデータの量に重み付けを行なう方式と，行なわない方式がある．

1. Round Robin(ラウンドロビン) 方式

   優先度によらず，同じパケット数づつ出力する方式，例えば，3 クラスの優先制御を行なう場合，優先度の高いクラスを 5 パケット出力し，次に 2 番目の優先度のクラスのパケットを 5 個出力し，最後に一番優先度の低いクラスのパケットを 5 個出力し，これを繰り返す．重み付けは行なわない．

2. Fair Queue(フェアキュー) 方式

   クラスごとに順番に出力する方式，ということでは上述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，同じバイト数づつ出力する．例えば，最も優先度の高いパケットを 100 バイト出力したあとに 2 番目の優先度のパケットを 100 バイト出力，そして最後に最も優先度の低いクラスのパケットを 100 バイト出力し，これを繰り返す方式．重み付けは行なわない [14].

3. WRR(Weighted Round Robin) 方式

   ラウンドロビン方式に重み付けを行なう方式．例えば，最も優先度の高いクラスのパケットを 10 個出力したあと，2 番目の優先度のパケットを 5 個出力し，最後に最も優先度の低いクラスのパケットを 2 個出力し，これを繰り返す方式．この場合の重みは 10:5:2 になる [15].

4. WFQ(Weighted Fair Queue) 方式

   フェアキュー方式に重み付けを行なう方式．例えば優先度の高いほうから順番に 100 バイト，50 バイト，20 バイト出力し，これを繰り返す場合，重みは 10:5:2 になる [16].

2.4.3 遅延時間制御のまとめ

従来のキューイングハードウェアは，集中制御で逐次処理を行うため生じる処理の柔軟性の欠如など，同期回路に生来的に存在する機能的制約により，キュー内での柔軟なパケットの入れ替えが不可能である．そのため，処理速度を重視した場合，ハードウェアキューでパケットの待ち合わせ，スケジューラでパケットの出力される順番を変える優先処理を行なう必要がある．しかし，キューが単なる記憶装置として使われることにより，クラスの数にあわせてハードウェアキューを用意する必要がある．そのため，

1. ハードウェアの肥大化が避けられず，
2. 同時に，スケジューリング遅延が増大する

という問題が生じる．また，ハードウェアを肥大化させないために，単一のキューからパケットを取り出し，スケジューリングを行なおうとすると，キューの先頭から出力するという FIFO の原則が崩れ，キューの途中から出力することになるため，

1. キュー管理が複雑化し，
2. 処理遅延が増大する

という問題が生じる．

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の能力を超えたとき，処理しきれないデータはキューにたまっていく．たとえば，伝送容量が 1Mbps で，パケットの到着レートが 2Mbps のとき，キュー内のデータの量は毎秒 1M ビットの割合で増加する．もし，キュー長が L バイト，伝送容量が T bps，パケット到着レートが P bps ならば，$\frac{8L}{P-T}$ 秒後にキューのデータがあふれ廃棄が起こる計算になる．このときキューに入力されたパケットは $\frac{8L}{T}$ 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 が悪化してくる．

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始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．

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たまってくるとキューの入り口で廃棄を行なうことになるが，そこで廃棄を行なうか行なわないかを決める要素は，

- 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たまってくる．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)

   2.6 節で説明する EF(Expedited Forwarding) クラスのフローは，最大の帯域幅が与えられ，廃棄率，遅延とも最小のクラスであるが，転送路の帯域幅を独占してしまって他のクラスのフロー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，超過した分は無条件で廃棄される．このとき廃棄を行な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)

   パケットはルータに入力されて出力されるまでの間はキューにとどまるが，出力側の転送速度が小さいと，キューにたまるデータが増えてくる．そのままにしておくとキューをすべて使い果たし，入力されたパケットはキューの入り口で優先度に関係なくすべて廃棄される．そのため，すべての TCP フローはウインドウサイズを小さくして再送する．そのため，転送速度が落ちる．この状態をグローバル同期といい，ネットワークのレスポンスが急激に悪化する．そのような状態を引き起こさないように，キューがパケットであふれる前に廃棄を始め，輻輳を起こさないようにする工夫が必要になる．アブソリュートドロッパは，キューにたまったデータ量にある閾値を決め，その量を超えるとキューがあふれる前に廃棄を始める方式である (図 2.2)．しかしながら，一旦廃棄を始めると，キューに余裕ができない限りこのままではすべてのクラスのパケットが廃棄されてしまうので，実際にはクラスごとにキューを用意し，廃棄開始の閾値を変え，通過率に差をつける必要がある [10].

3. RED(Random Early Detection)

   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の量が閾値を超えるとそれ以後やってくるパケットは 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インドウサイズを小さくして再送し [17]，それによってネットワークのレスポン

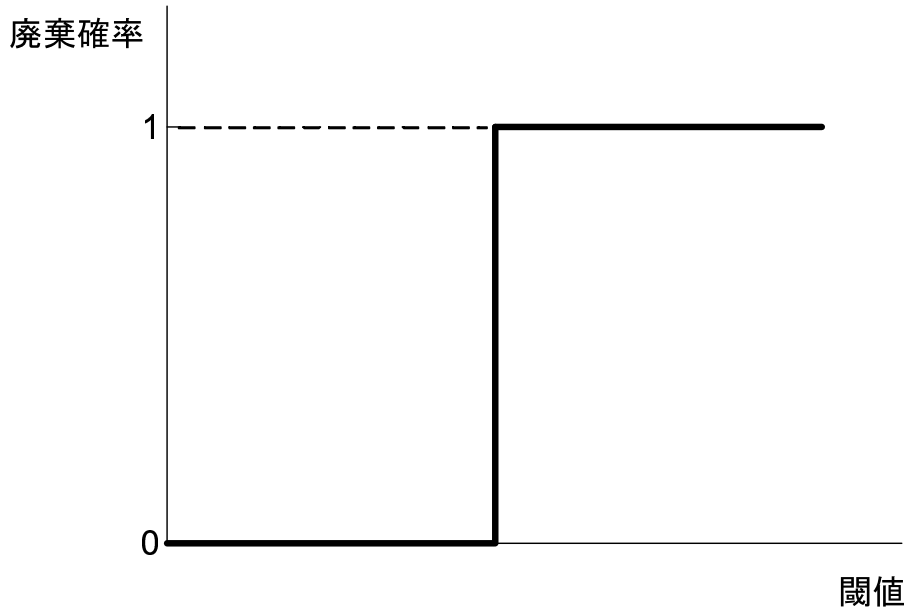

図 2.2 Algorithmic Dropper

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確率的に廃棄させていくと，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，輻輳が起こらないようにできる．この廃棄アルゴリズムが RED(Random Early Detection) である (図 2.3)[18].

ここでは，大小 2 つの閾値 Max および Min を設定しておいて，キューにたまったデータの量 Depth が Min より少ないとき (Depth < Min) にはすべてキューに入力させるが，キューにたまったデータの量が Max より大きくなると (Depth > Max) すべてキューの入り口で廃棄する．そしてその中間のとき (Min < Depth < Max) にパケットを確率的に廃棄させ，グローバル同期がおこらないようにする方法である．つまり，乱数を rand，最大廃棄確率を MaxProb とすると，

if (Depth < Min)

　　キューへ

else if (Depth > Max)

　　廃棄

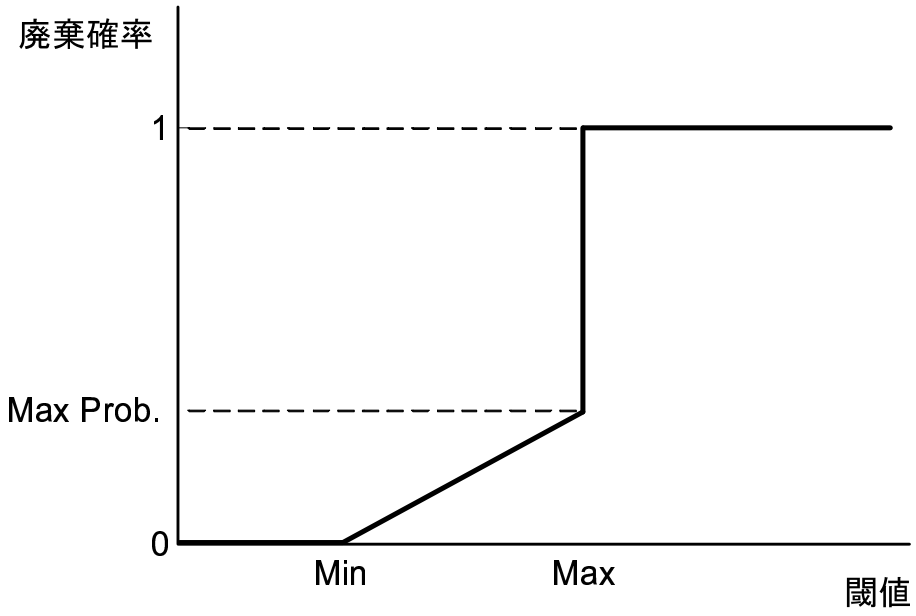

図 2.3　Random Early Detection

else

if ($\frac{\text{MaxProb}}{\text{Max}-\text{Min}} \times (\text{Depth} - \text{Min}) \times \text{rand} < \frac{1}{2}$)

キューへ

else

廃棄

となる.

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，サービスの種類によって要求される QoS が異なる．そのため，転送されるときに帯域幅，廃棄率，遅延時間およびジッタの絶対的な値を制御することが必要である．クラスベースの 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毎に QoS 制御ルールを適用して転送する．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.

2.6.1 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よって分類する．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．境界ルータではフローの種類をいろいろな情報をもとに識別しクラス分けする．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 ヘッダのTOS(TYPE OF SERVICE) フィールド8 ビットのうち，上位6 ビットを再定義) の情報のみをもとにフローをクラス分けする方法．同一のDSCP をも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レート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障され，ジッタが少ない専用線的な (Vertual Leased Line) サービスが受けられる．ただし，流入するフローの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，超過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，下に述べるBE(ベストエフォート) クラスと共存するためには，トラフィック調整を行い，帯域を占有しないようにする必要がある [19]．DSCP=101110 である．

   (b) AF(Assured Forwarding) クラス

   パケット転送保障型，最低帯域幅が保障された転送クラス．転送レベルの高いほうから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 つのレベルに分けられ，高廃棄レベルから順に廃棄がおこなわれる．各ク

ラス・レベルと，それに対応する DSCP は表 2.2 のとおりである [20].

表 2.2 Assured Forwarding の各クラスおよびレベルと DSCP の対応

	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(Best Effort) クラス

ベストエフォート型，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されない [9]．DSCP=000000 である．

(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とき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，他の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 [9]．DSCP=XXX000(X は 0 または 1) である．

2. MF(Multi Field) クラシファイヤ

コアルータでは DSCP のみをもとにパケットのクラスを分類したのに対して，境界ルータでは IP ヘッダの他のフィールドや TCP ヘッダ (図 2.4) あるいは UDP ヘッダ (図 2.5) の内容に基づいてフローを識別し，クラス分けを行なう．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 [8].

3. その他のクラシファイヤ

上述のクラシファイヤは，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なったが，下位のレイヤに基づく分類も可能である．

(a) データリンクレイヤの活用

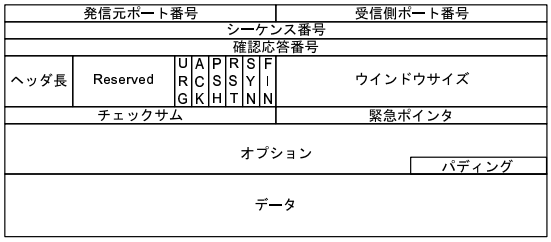

図 2.4 TCP ヘッダ

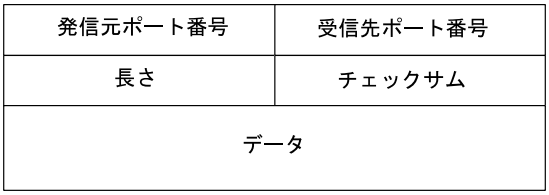

図 2.5 UDP ヘッダ

IEEE[*1] 802.1p によって設定された優先度[*2]タグを読み取り，分類する優先化 LAN．この方法は，データ・フレームごとに個別に優先処理を行なうための順序付けサービスである [21]．また，パケット中に VLAN 用 ID 情報を追加する IEEE 802.1q VLAN 優先度によるクラス分けもその一例である [22] .

(b) IP または論理あるいは物理インタフェース識別子によるクラス分け

例えば，複数の入力チャネルがあるときに，その入力チャネル番号によるクラス分け．

2.7 結言

ネットワークの伝送路の帯域幅が拡大したため，あらゆる種類のサービスの多様な QoS を満足するためには，広帯域，低遅延，低ジッタのネットワークの構築が急務である．そのため，ルータでの処理がネットワーク全体のボトルネックにならないように，処理遅延を最小に抑え，スループットを最大にあげる必要がある．しかしながら，従来のハードウェアキューで柔軟な処理を実現するためには，回路を追加することが必要である．また，従来のソフトウェア処理ではノイマン型処理を基礎にしているため，メモリアクセス遅延やプロセッサ間通信遅延などの種々の遅延が積み重なり，全体のスループット性能に悪影響を与える．そのため，従来のネットワークプロセッサでは，帯域幅，キューイング遅延時間，パケット廃棄率等の QoS 項目を同時に制御することができない．

[*1] Institute of Electrical and Electronic Engineers:米国電気電子技術者協会

[*2] マルチメディアアプリケーションの優先順位制御

本章では，ルータの構成要素の特徴を述べたが，高速ネットワークでその機能を実現する上での問題点および対処方法は，処理機構をハードウェアで構成した場合とソフトウェアで構成した場合の利点および問題点を列記することで浮き彫りにされてくる．

1. 遅延時間制御

   従来のハードウェア処理では，低遅延の処理が可能だが，柔軟性がない．一方，ソフトウェア処理では，柔軟な処理が可能だが，管理が複雑で，しかも遅延が大きくなる．

2. スループット

   ハードウェアで構成した場合は性能が高いが，ソフトウェアで構成した場合は性能が低い．

3. プログラマブルな構成

   ハードウェアでは用途毎に異なる ASIC [*3]を用意する必要があり，構成の柔軟性に欠けるが，ソフトウェアではルータの用途に応じたソフトウェアをインストールすることで柔軟な構成が実現できる．

以上の 3 点からわかるように，処理の高速化と柔軟な処理という相反する目標を同時に満たすためには，どの構成要素にハードウェアを適用するか，また，どの構成要素にソフトウェアを適用するか，適切な選択をする必要がある．

本研究では，キュー機構には遅延時間を最小にするため，ハードウェアを適用し，それ以外の構成要素は用途によって構成を変更するためプログラマブルな構成が必要なため，ソフトウェアで実現する方式を採用した．キューについての適用方法は第 3 章で，その他の構成要素の適用方法については第 4 章で提案する．

[*3] Application Specific Integrated Circuit 特定用途向き集積回路

第 3 章

クラスベースの優先制御への自己同期パイプライン方式の適用

3.1 緒言

優先制御は，転送優先度をもとに，入力されたパケットのスケジューリングを行なうことである．将来の情報通信ネットワークで優先制御を効果的に実現するための要件は次の 2 つである．

1. 光ネットワークに対応した高速処理

   転送路はバックボーンでは光ファイバー化が終わり，収容線でも ADSL，ケーブルネットワーク，FTTH などで高速化がなされてきた．そのため，ルータにおける処理がボトルネックになる恐れが非常に大きい．今後，ルータの処理は高速化が必須であり，QoS 処理についても同様のことがいえる．

2. スクリプトなどによる柔軟な優先・廃棄制御

   高度な QoS 処理を行なうためには転送路の輻輳状況に対応したパラメータの細かなチューニングが不可欠である．そのため，スクリプトなどによってプログラマブルで，輻輳状況の変化に柔軟に対応できることが求められる．

本章では，クラスベースの優先制御を行なう方法として，従来方式のパイプライン処理 (廃棄制御として RED，スケジューリングとして WRR を使用) の問題点について上の 1. および 2. の観点から述べ，その問題点を解決するために DDMP の自己同期型パイプライ

ンをキューに適用した新方式の動作原理とその回路構成法を述べる．

3.2　従来方式の問題点

高速処理と柔軟な優先処理という相反する要件を満たすために，これまでもいろいろな取り組みが行なわれてきたが，その方法は主に，

1. ハードウェア化による高速化
2. ソフトウェアによるキュー管理

である．しかし，これらの方法はそれぞれに以下のような問題点を含んでいる．

3.2.1　ハードウェア・キューの並列化

一般に，クラスベースの優先制御をハードウェアによって実現するためには，キューに入力してから出力するまでにパケットの順序の入れ替えが必要である．そのため，図 3.1 のように，優先度の異なるフローごとにキューを用意して廃棄制御をおこなった後データをキューに入力し，キューからの出力時に何らかのスケジューリングを行なって，優先度の高いパケットを優先的に出力する必要がある．

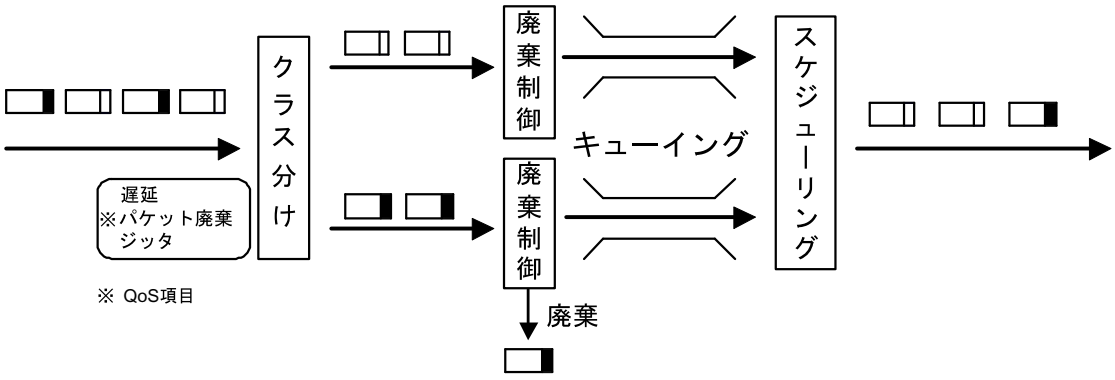

図 3.1　従来方式のクラスベース優先制御

これらの処理ルーチンをハードウェアに組み込むと，確かに高速化は実現できるが，優先制御のパラメータは，SLA の交換などのように，ISP 間の取り決めで変化するので，ひとつ

ひとつの取り決めに対して処理を最適化するためのチューニングが必要になり，ハードウェアで実現するのは非常に困難である．また，クラス構成によっては，メモリの利用効率が悪い．

3.2.2 ソフトウェア制御

ソフトウェア制御では，ハードウェアによる優先処理と違い，大きなサイズのキューをひとつ用意しておいて，その中に論理的なパイプラインを複数組むことができ，キューの使用効率は高くなる．しかし，その一方，パケットの順序変えやキュー管理がハードウェアで実現したときに比べ格段に複雑になり，高速化という点で実現困難である．

3.2.3 解決法

クラスベースの優先制御を光ネットワークに対応した高速で処理し，さらにスクリプトなどによって柔軟に行なうためには次のことが同時に解決されることが必要である．

1. 単一のキューでハードウェアの肥大化を防ぎ，同時に使用効率を高める．
2. 自律的にパケットの順序が入れ替わることでスケジューリングの遅延を低く抑える．

この相反する 2 つの要素を同時に解決する方法として，本研究では，折り返し型自己同期パイプライン方式をキューに適用した，“自律型優先キュー” を提案する [23].

3.3 自律型優先キュー

ここでは，パイプライン内部でのデータの移動および廃棄原理の説明を行なう [23].

3.3.1 動作原理

自律型優先キューは図 3.6 のようにはしご型になっている．以下に詳しく説明する．

この図で丸付き数字はデータの優先度を表し，キューに入力されたデータは，

1. パイプの前の階層にデータがなければ前進し，データがあればそのままの位置にとどまる (図 3.2).

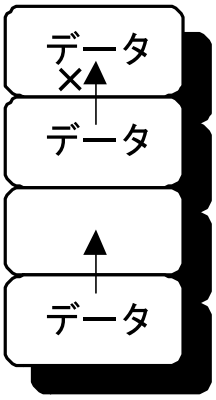

図 3.2 前進・ストップ

2. パイプの同じ階層の隣にデータがなければ横に移動する (図 3.3. 遅延の減少).

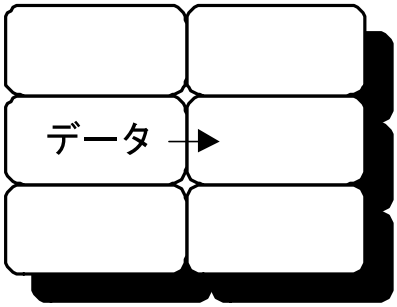

図 3.3 横へ移動

3. パイプ中のデータがない階層に前述の 1. および 2. の移動が可能なデータがある場合 (データの衝突)，優先度の高いデータが移動してくる (図 3.4. 追い越し).
4. 3. の状態で，同じ優先度のデーが衝突した場合は，前進してきたデータが優先する (図 3.5 同一フローのデータの順番の確保).

という規則に従って移動し，最終的にデータはキューの出口から出力される.

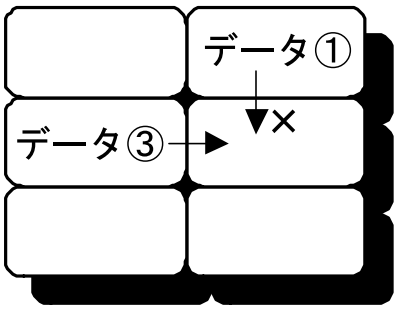

図 3.4 追い越し

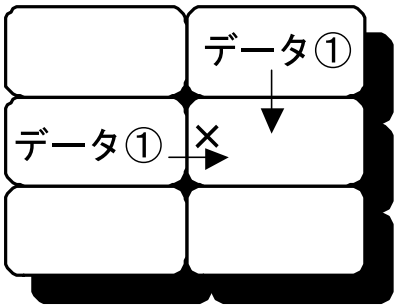

図 3.5 順番の確保

3.3.2 廃棄条件

キューに入力される際のパケットの廃棄原理は，すでに 2.5 節で説明したように，自律型優先キューではキューの内部でも廃棄処理が行なえる．その原理は，キューのはしごを上っていくデータは廃棄優先度 (AF クラスの説明参照) にしたがって到達可能な高さが異なり，その高さに達したデータは廃棄される (図 3.6)，というものであり，これによって優先度の高いパケットの遅延時間およびジッタが減少する．

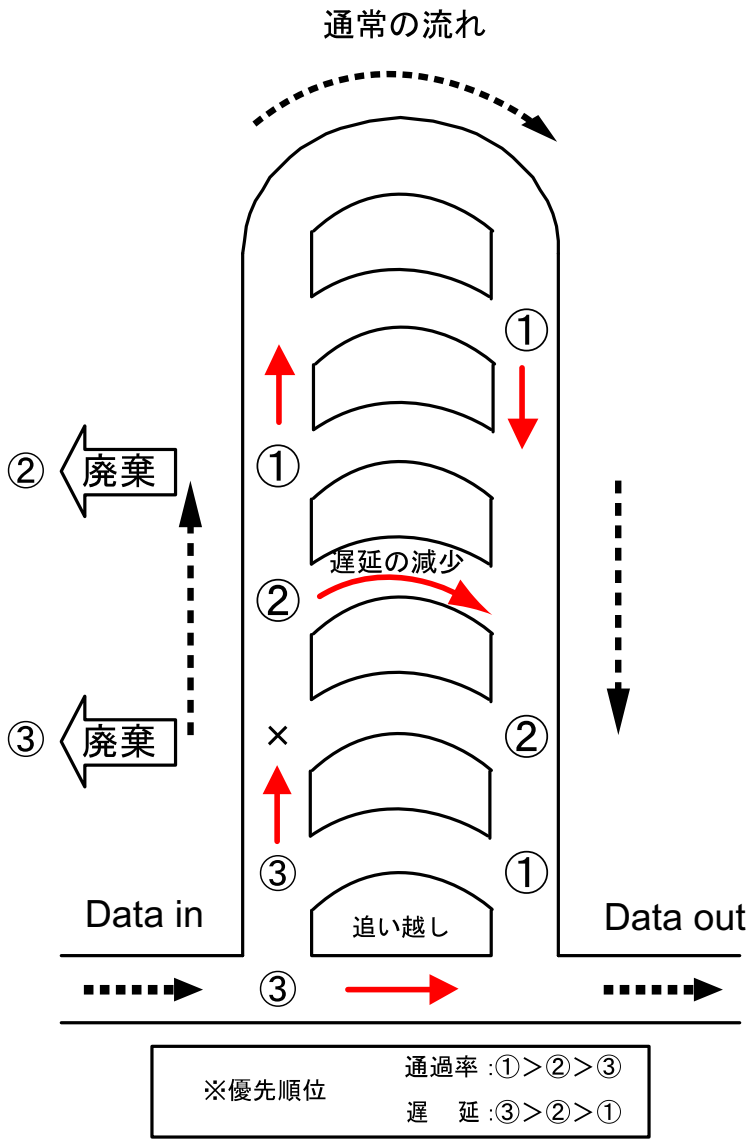

図 3.6　自律型優先キューの動作原理

3.4　折り返し型自己同期パイプライン

3.4.1　自己同期型パイプライン機構

クロック同期型のパイプライン機構では，図 3.7 のように全パイプラインに常にクロックが供給されていて，送信側と受信側がクロック信号を共有していて，このクロック信号でデータを送るタイミングを取る．そのため，データが流れていなくても動作する．また，送信側と受信側でクロック信号を共有するために余分な電線と回路が必要となり，よりコスト

高になる．

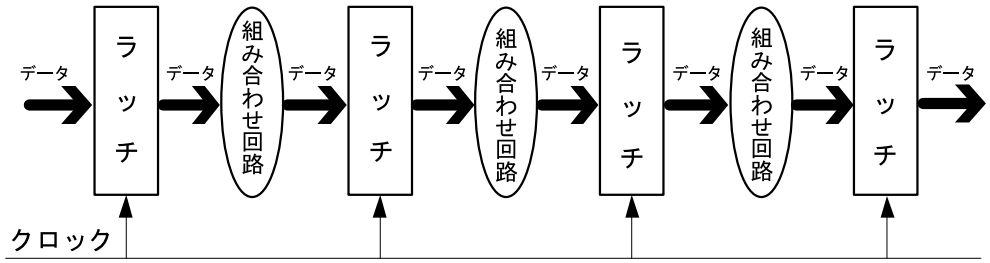

図 3.7　同期型パイプライン機構

それに対して，自己同期型パイプライン機構では，PLL(Phase Locked Loop) などで位相を合わせる必要はなく，パイプラインの送信側と受信側の間に C 素子[*1]という一種のハンドシェイク回路を設け，擬似的なクロック信号を自己発生させている (図 3.8)．

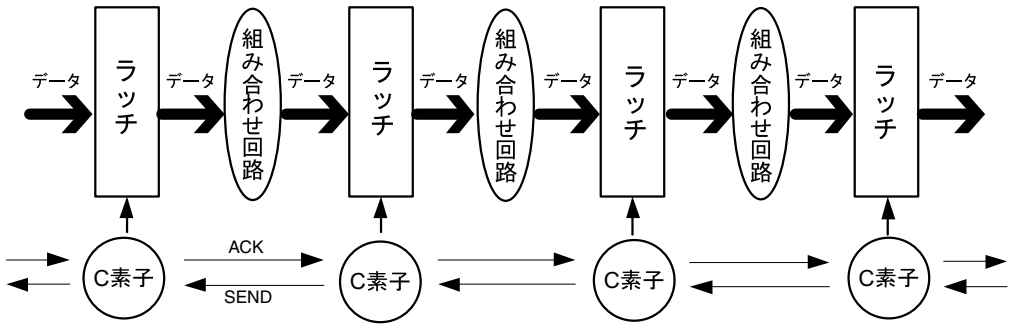

図 3.8　自己同期型パイプライン機構

この信号はデータがパイプラインに入ったときのみ有効な信号で，データが来ないときには動作せず，自然に省電力化できる回路となっている．ラッチにデータが入ると，C 素子はパイプラインの次の段の C 素子に転送要求 (ACK) を出す．もし次の段のラッチにデータがなく，転送可能ならその段の C 素子は前の段の C 素子に転送可能 (SEND) を返す．そしてデータはロジック回路を通り，次の段のラッチに移動する．もしデータが次の段にあって，転送不可能なら C 素子は禁止信号を返す [24]．

[*1] Localized handshake data transfer control circuit

3.4.2　折り返し型自己同期パイプライン機構

折り返し型自己同期パイプライン機構は，図 3.9 のように，パイプラインの途中でバイパスするように C 素子間で ACK および SEND 信号をやり取りできるようにしたものである．

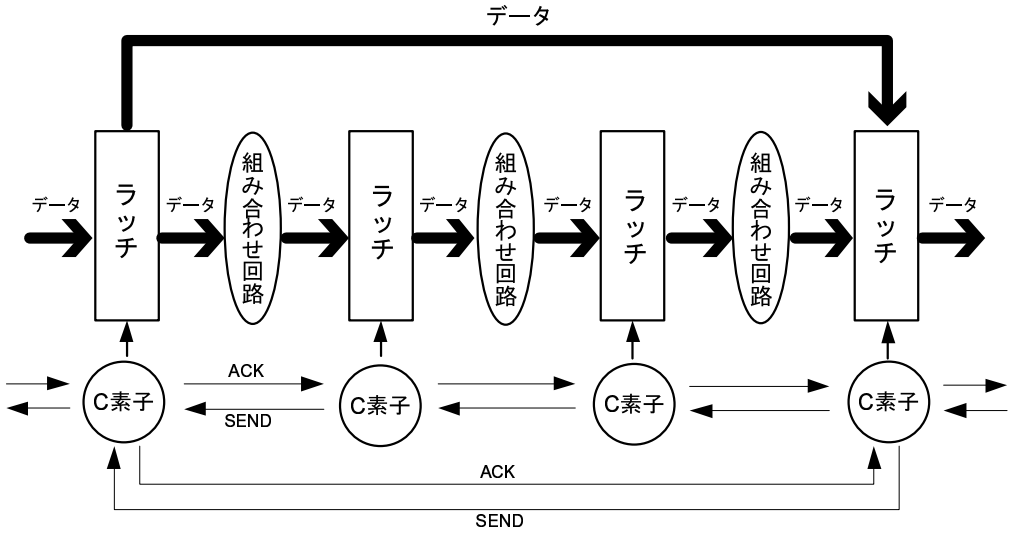

図 3.9　折り返し型自己同期パイプライン機構 1

これをパイプの中央で折り返すと図 3.10 のように，隣接する同じ階層の C 素子どおしをハンドシェイクで結んだものになる．このパイプライン構成で，優先制御と廃棄制御を自律的に行なうことができる．

図 3.10 では，ラッチ A にデータがあるとすると，まず，パイプラインの中でラッチ A の C 素子は，1 段上の階層および隣の同じ階層の C 素子に ACK 信号を出す．

もし，

1. 隣 B にデータがないと，
   (a) 空いている段のひとつ前のラッチ C から同じく ACK 信号が来たとき，
      i. 優先度が A のデータ $>$ C のデータ の場合，

         B の C 素子は A の C 素子に SEND を返し，ラッチ A のデータがラッチ B に移動する (優先度の違いによるデータの追い越し)
      ii. 優先度が A のデータ $\leq$ C のデータ の場合，

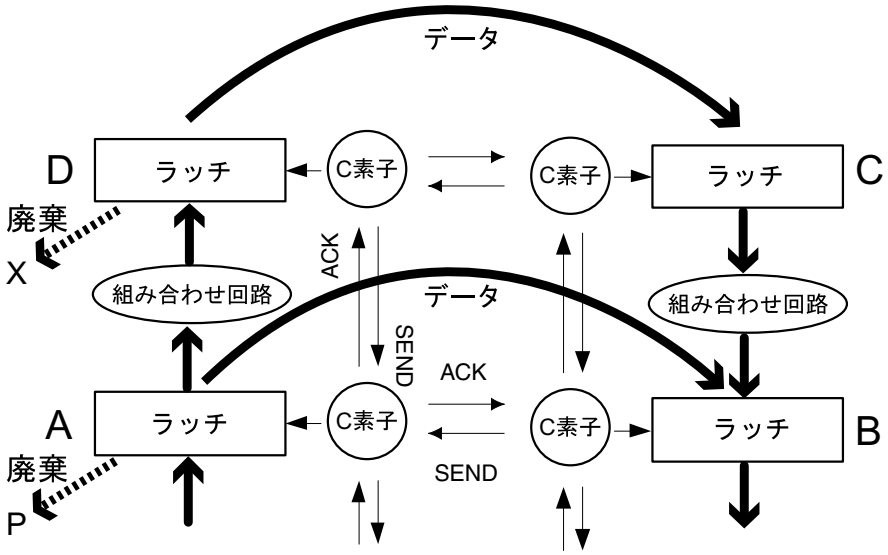

図 3.10　折り返し型自己同期パイプライン機構 2

B の C 素子は C の C 素子に SEND を返し，ラッチ C のデータがラッチ B に移動する (通常の移動)

(b) C のラッチが空いているとき，

B の C 素子は A の C 素子に SEND を返し，ラッチ A のデータはラッチ B に移動する

2. 隣のラッチが空いていないとき，

(a) A の上の階層 D のラッチが空いているとき，

D の C 素子は A の C 素子に SEND を返し，ラッチ A のデータはラッチ D に移動する (通常の移動)

(b) 上の階層が空いていないとき

ラッチ A のデータはそのまま

次に，図 3.10 でパイプラインをデータが前進していき，

1. 廃棄優先度が高いデータ P は階層 A まで上ったとき消去され，
2. 廃棄優先度が低いデータ X は階層 D まで上ったときに消去される

ようにすれば，廃棄優先度の違いによるレベル分けが実現できる．この廃棄優先度の違いによるデータの消去される高さの変更は，スクリプトによって可能である．

3.5　結言

ここでは，DDMP の折り返し型自己同期パイプライン機構をキューに適用した，自律型優先キューの動作原理とその構成法を提案した．この方式では，

1. スケジューラなしに優先度を基にしたパケットの追い越しが可能
2. キューの入り口だけでなく，キューの中でもパケット廃棄が可能

という点から従来型の優先制御に比べて高速で柔軟な優先制御および廃棄制御が期待できる．定量的な評価については第 5 章に述べる．

第 4 章

クラスベースの優先制御が可能な DDMP マルチプロセッサ

4.1 緒言

第 2 章ではクラスベースの優先制御を行なうために必要な構成要素を列記し，ルータの構成を図示した．また，第 3 章では折り返し型自己同期パイプラインによる自律型優先キューの実現方法について述べた．

本章では，これまで述べてきたルータの構成要素を DDMP ナノプロセッサにプログラムする上で必要なハードウェア，そのハードウェアを制御するための新命令およびナノプロセッサ上のプログラムについて述べる．

4.2 DDMP マルチプロセッサのアーキテクチャ

4.2.1 DDMP-AUDIO の内部データパケット構造と IP パケットの構造

IP パケットは 7 ページ図 1.4 のように 32 ビット単位で表示される．また，パケット長も 32 ビットの整数倍である．つまり，IP パケットの処理には 32 ビット単位のデータを一度に扱えるプロセッサが適している．

DDMP-AUDIO(以下，AUDIO と記述する) は音声信号処理用に開発されたデータ駆動型プロセッサで，内部データは図 4.1 のように入力データを 32 ビット単位でパケットに分割し，一連の世代番号 (同じ LN# で PX# が 0,1,2,3... と増えていく) を割り当てプロセッサ

内を転送する．

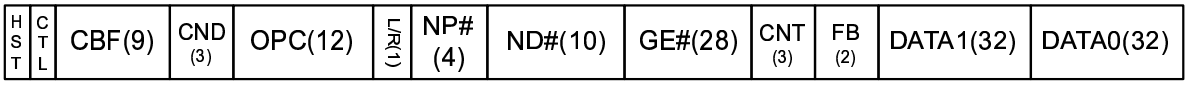

図 4.1 DDMP-AUDIO 内部データパケット構造

なお，表 4.1 はデータの各領域の役割を表す．

表 4.1 AUDIO 実行パケットの各領域説明

領域名	役割
HST	ホスト転送フラグ，T のときホスト転送パケットを示す
CTL	制御フラグ，T のとき制御フラグを示す
CBF	OPC が示す命令の動作に対し補足をする
CND	FB 領域と組み合わせて演算の実行/不実行を制御する
OPC	命令コード
L/R	データの左右属性の識別
NP#	命令を実行する nPE の識別
ND#	ノード番号
GE#	世代番号 FD#，LN#，PX#からなる
CNT	
FB	演算の実行/不実行を制御する
DATA1	左データ
DATA0	右データ

これは，AUDIO と IP パケットの構造との相性が非常に良いことがわかる．つまり，1 個の IP パケットを AUDIO のデータパケットに分割すると，32 ビット単位に分割されたデータとなる．そして，クラス識別を行なう DSCP はその第 1 番目のデータパケットの 8〜13 ビットにセットされる．また，Average Rate Meter，アルゴリズミックドロッパ，RED な

どで参照される IP パケットの Total Length も第 1 番目のデータパケットの 16〜31 ビットにセットされる (図 4.2).

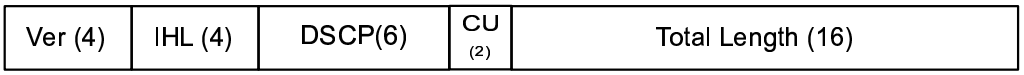

図 4.2 IP ヘッダ第 1 列の構成

つまり，処理に AUDIO を使うと，IP パケット中の QoS 情報は一連の AUDIO 内部パケットの第 1 番目のパケット (PX#=0) のみを参照すればよいことになる．したがって，今回のプログラムは AUDIO の命令セットを使用する [25].

4.2.2 優先制御機構の構成

クラスベースの優先制御機構の例として，図 4.3 を説明する [10]．まず，入力された IP パケットは入ってきた順にメモリに記憶される．そして，ヘッダの最初の 32 ビットだけ取り出され (世代番号 PX#=0)，クラシファイヤに入力される．以下，この 32 ビットをヘッダと記述する.

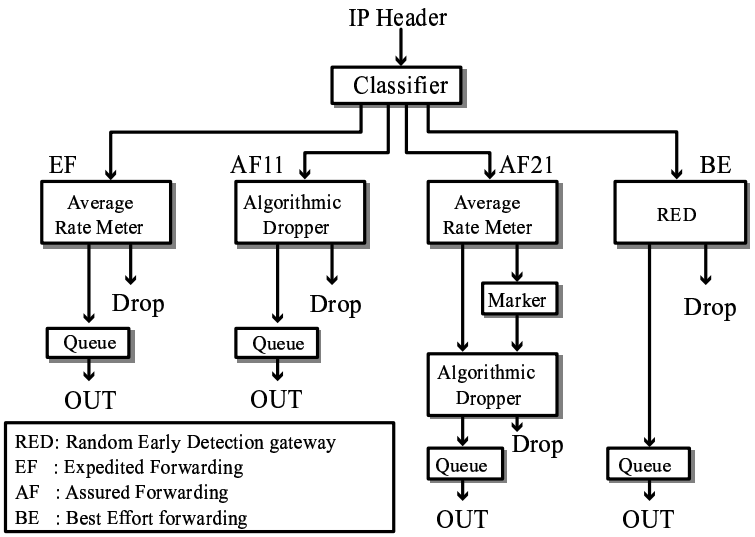

図 4.3 構成例

クラシファイヤでは，入力されたヘッダ中の DSCP の上位 3 ビットを読み取り，その値

によってクラス分けを行なう (表 4.2).

表 4.2 クラスと DSCP 上位 3 ビットの対応

クラス	ビット構成
EF クラス	101
AF クラス 1	001
AF クラス 2	010
AF クラス 3	011
AF クラス 4	100
BE クラス	000

EF クラスの場合は，Average Rate Meter に入る．Average Rate Meter では，タイマによる時間計測とパケット中の TL(Total Length) 値を読み取り，転送速度を計算する．もし，契約より転送速度が大きい場合は廃棄 (Drop) されるが，契約より転送速度が小さい場合はキューに入る．

また，AF クラスのとき，優先度の高い場合 (この例では AF21) には，EF クラスと同じように Average Rate Meter に入り，転送速度を計算し，転送速度が契約より小さい場合はアルゴリズミックドロッパに入るが，大きい場合はマーカに入り，DSCP の値を書き換えて優先度を低くした後，アルゴリズミックドロッパに入る．

アルゴリズミックドロッパでは，廃棄優先度ごとに異なる廃棄閾値をパラメータとして持ち，キューにたまったパケットの TL の合計が閾値より小さい間はキューに入り，閾値より大きくなるとキューの入り口で廃棄される．また，キューから出るときには，TL の大きさだけ queue depth から引かれる．

最後に，BE クラスの場合は RED に入る．RED では queue depth の閾値が二つあり，queue depth が最小閾値より小さければ無条件でキューに入り，最大閾値より大きければ無条件で廃棄される．最小閾値と最大閾値の間のときには，確率的に廃棄される．

4.2.3 複合命令

以上，AUDIO でのクラスベースの優先制御の例を説明したが，AUDIO はネットワーク処理用に開発されたプロセッサではないので，ネットワーク処理に特有の命令は存在しない．そのため，実行しようとすると，他のネットワーク用のプロセッサに比べ，不利になる部分が多くなる．その対応としてはハードウェアで対応するか，ソフトウェアで対応するかになる．

もし，ハードウェアで作りこむことにすれば，確かに高速で実行できるが，本研究の目的の一つである多様なサービスに対応する柔軟な優先制御が達成できない．したがって，ここではできるだけ既存の命令の組み合わせを複合命令化することで高速化を図るが，既存の命令で解決できないときには新ハードウェアおよび新命令を提案する．

4.2.4 QoS 制御用ナノプロセッサ

図 4.4 の QSCnPE(QoS controller nanoprocessor) は，QoS 制御用ナノプロセッサで，内部にハードウェアタイマモジュール (TM)，テーブルメモリモジュール (Int.TBL) を持ち，Average Rate Meter に必要なタイマ機能を例えばマッチングメモリ (MM)，数値論理演算装置 (ALU)，プログラム記憶 (PS) のような普通のデータ駆動制御機能に追加している [26]．ハードウェアタイマモジュールは循環するクロック信号を発生し，テーブルメモリ

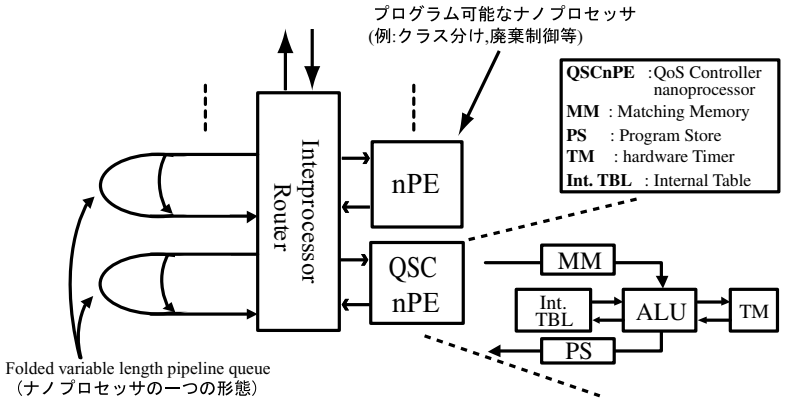

図 4.4 自己同期パイプライン優先制御の構成

モジュールはセットした時間のリード/ライト機能の他にインターロック機能をサポートする．ALU はこれら 2 つのハードウェアモジュールと他の QoS 命令のインタフェースを供給する．

その他のナノプロセッサには，クラシファイヤやアルゴリズミックドロッパなどの構成要素をプログラムする．

4.2.5 マルチプロセッサの構成

データ駆動型ネットワークプロセッサでは，自己同期型パイプラインキューモジュール (Folded variable length pipeline queue)，QoS ナノプロセッサ，そして通常のナノプロセッサはそれらの間でデータ駆動型パケットを交換する単純なインタープロセッサルータを介して結合している．それぞれの種類のナノプロセッサとキューイングモジュールの数は，コアルータ，境界ルータ，ホームゲートウエイなどのような目的とするアプリケーションドメインによって柔軟に決定される．そのため，計画されたネットワークプロセッサは，処理能力を向上させるために相互接続された 1 つあるいは複数の自己同期キューモジュールやいくつかの QoS ナノプロセッサを簡単に設計できる．

例えば，38 ページ図 4.3 のクラスベースの優先制御機構は図 4.5 のように表される．こ

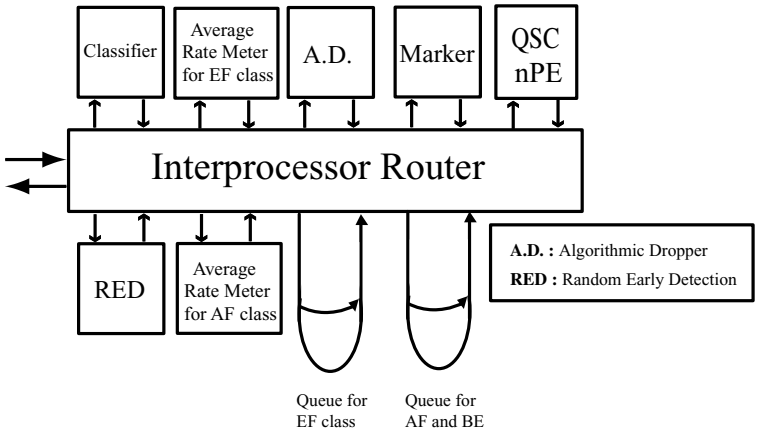

図 4.5 優先制御機構の構成例

の場合，4 つのクラスの廃棄制御機構は別々のナノプロセッサ上のデータ駆動型ソフトウェ

アによって実現され，キュー機能は本研究で提案された折り返し型自己同期パイプラインモジュールによって実現されるが，キューの数はナノプロセッサにどのような機能を適用するかで変えることができる [27].

4.3 提案する複合命令および新命令

クラスベースの QoS 制御でボトルネックとなる部分を以下に述べ，ナノプロセッサ上にモジュール化するフローグラフを示す．その際今回開発した複合命令および新命令を提案する．

1. Average Rate Meter

   メータはビットレートを計算する部分である．そのため，経過時間を測る機能と，入力されたデータのビット数を記憶する部分が必要である．DDMP はデータ駆動型のプロセッサなので，各命令はデータがそろった部分から発火する．したがって命令の実行のため同期を取る必要がなく，クロックを持たないので，ソフトウェアでは実現できない．今回はタイマ部分をハードウェアで対応する．

2. クラシファイヤ

   クラシファイヤは，DSCP のビット列から転送クラスを読み取り，フローを分類する部分であり，クラスベースの QoS 制御を行なう上での前提条件である．この部分の遅延の原因はビット列の比較の繰り返しである．

3. マーカ

   マーカは Average Rate Meter で契約以上の転送レートのフローの DSCP を再マーキングして，優先度を落とす機能である．

4. アルゴリズミックドロッパ

   アルゴリズミックドロッパはパケットの廃棄制御を行なう部分である．キュー中に存在するデータのビット数 (Total Length) の合計 (queue depth) が閾値より小さい場合はキューの入り口にやってきたデータはキューに入力されるが，閾値より大きくなると

キューの入り口で廃棄される．この部分の遅延の原因は大小関係の比較である．

5. RED

   RED はパケット廃棄制御を確率的に行なう部分である．ここではこの処理部分に入ってきたパケットの TL を抽出して，queue depth に加える．そしてその値と大小 2 つの閾値との大小比較を行い，queue depth が大きいほうの閾値より大きい場合は廃棄，小さいほうの閾値より小さい場合は次の段階に進める．そして queue depth が大小の閾値の間の場合は乱数をかけて，廃棄するかキューに通過するか決定する．この部分の遅延の原因は大小関係の比較および乱数の発生である．

6. キューの出口での処理

   38 ページ 4.2.2 で記述したように，キューに入力されるのは IP パケットの最初の 32 ビットである．それ以降の IP ヘッダを含むデータは全てメモリに記憶されていて，キューの出口でキューから出力される DDMP パケットのもつ世代番号中の LN#と等しいものが PX#=0 から順番に Total Length 分だけメモリから読み出される．しかし，AUDIO はデータ駆動型プロセッサなので，メモリからデータを 1 個 (32 ビット) 読み出すごとにトリガを発生させて発火させる必要がある．この部分の遅延の原因はトリガの発生である．

4.3.1 Average Rate Meter

Average Rate Meter に必要なデータは経過時間と流入してくるデータのビット数である．まず，経過時間はタイマで計るが，これは一定の時間間隔でタイムスタンプを出力するもので，タイムスタンプはループして元に戻る．またデータのビット数は，パケットの Total Length:L から読み取る [26].

処理の流れは，あらかじめ決められた計測インターバル I と，パケットがやってきたときの時刻 c と直前のパケットがやってきた時刻 a の差，$c - a$ を比較し，

1. $c - a > I$ ならば a を c で書き換えたのちに average rate: $\frac{L}{I}$ の値でメモリ m の内容を

更新する.

2. $c - a \leq I$ ならばメモリ m の内容を参照し，average rate: $\frac{L}{I}$ に加え，その値で m を更新する.

そして，m と average rate の制限値 r の比較をおこない,

1. $r \geq m$ ならば
   (a) EF クラスならキューに進み,
   (b) AF2 クラスならアルゴリズミックドロッパに進む.
2. $r < m$ ならば
   (a) EF クラスなら廃棄され,
   (b) AF2 クラスなら Marker に進み，DSCP を書き換えられた後にアルゴリズミックドロッパに進む

ここでタイマ参照命令 TM，テーブルメモリ参照命令 VTSELG およびテーブルメモリ更新複合命令 VTREPG を追加する.

図 4.6 のように，TM は 1 入力 1 出力命令で, 入力はトリガである. 入力があった時点のハードウェアタイマの値を参照して出力する.

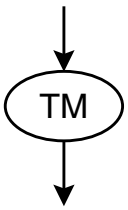

図 4.6　新命令 TM

図 4.7 のように，VTSELG は 1 入力 1 出力命令で, 入力はトリガである. 内蔵テーブルメモリを参照して出力する.

図 4.7　新命令 VTSELG

図 4.8 のように，VTREPG は 1 入力 1 出力命令で, 内蔵テーブルメモリを入力データで参照および更新して出力する.

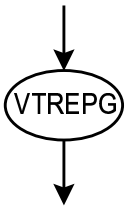

図 4.8　新命令 VTREPG

図 4.9 は EF クラス用，図 4.10 は AF2 クラス用の Average rate Meter 用のフローグラフである．

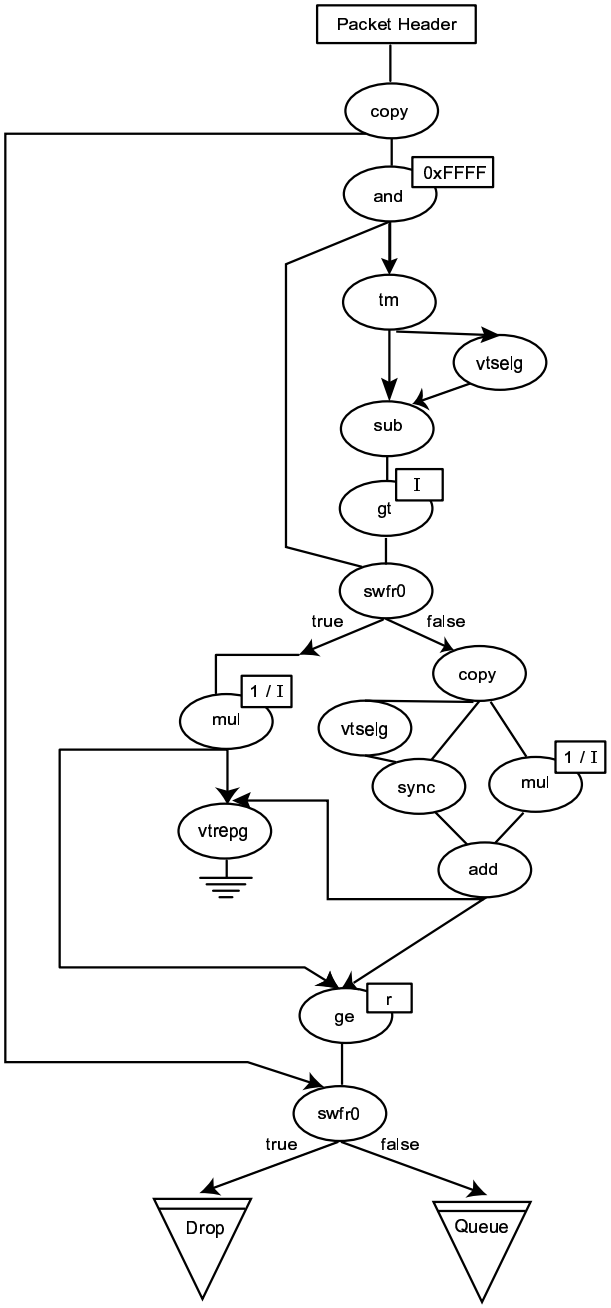

図 4.9　EF クラスの Average Rate Meter

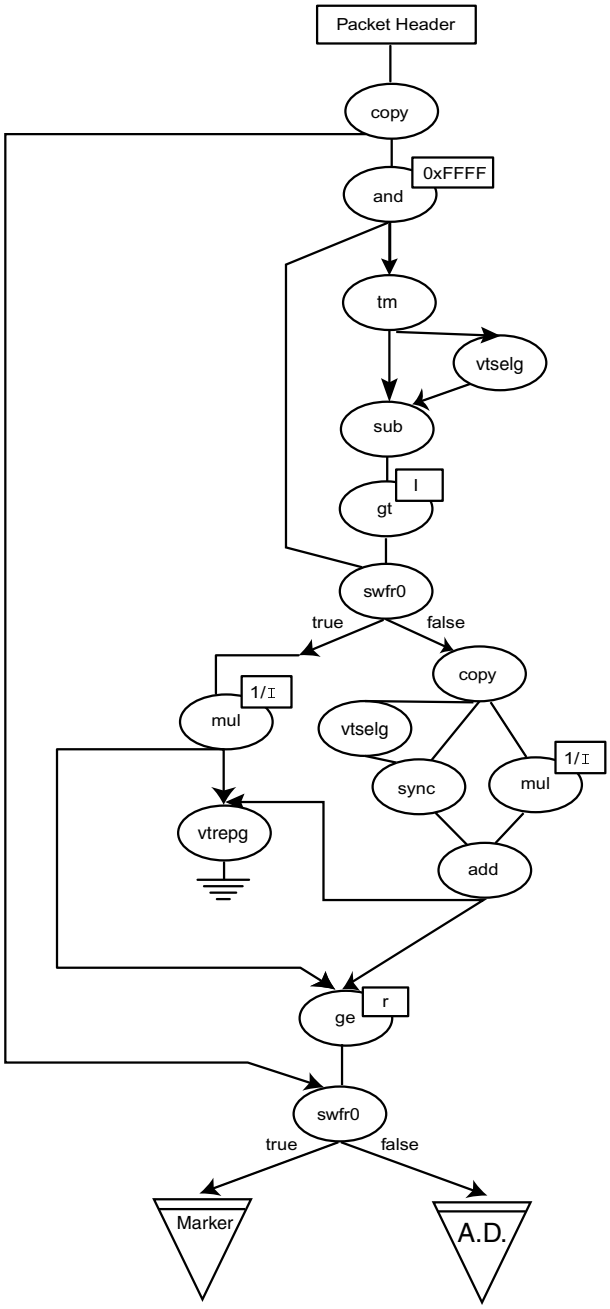

図 4.10　AF2 クラスの Average Rate Meter

4.3.2　クラシファイヤ

クラシファイヤでは，ヘッダの DSCP の上位 3 ビット (第 8〜10 ビット) の値 (表 4.2 参照) によってクラス分けを行なう．ここで入力されたデータをその値をもとに 8 分岐する間接ジャンプ命令 BRD8 を追加する．図 4.11 のように，BRD8 は 2 入力 1 選択出力で,R-DL 下位 3 ビットを ND#に加算して出力する．

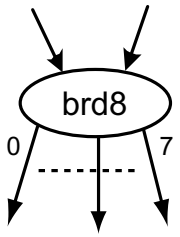

図 4.11　新命令 BRD8

クラシファイヤのフローグラフを図 4.12 に示す．

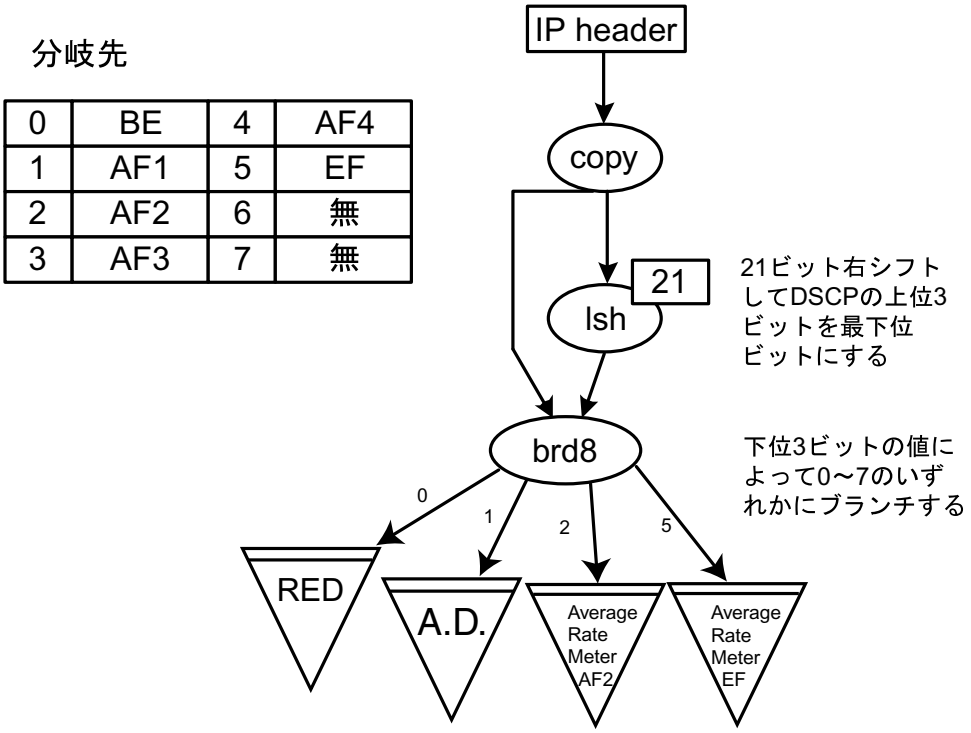

図 4.12　クラシファイヤ

4.3.3　マーカ

マーカでは，入力されたヘッダの DSCP を 0 にセットしてから各クラスに対応する数値をマーキングする．図中の命令 MRKDP は複合命令で，マークする DHCP の値は表 4.2 のいずれかの値を 21 ビット左シフトした値である．フローグラフを図 4.13 に示す．

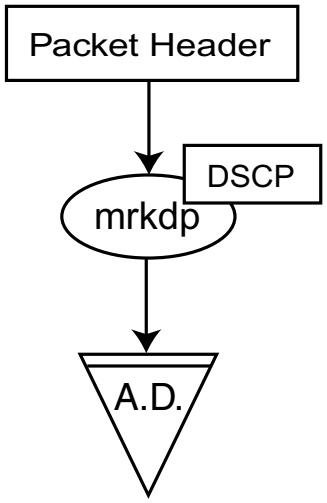

図 4.13　マーカ

4.3.4　アルゴリズミックドロッパ

アルゴリズミックドロッパでは，キューに存在するデータのビット数 (queue depth) にヘッダの Total Length(第 16〜31 ビット) を加えた値:Q と廃棄閾値:T を比較し，

1. Q < T ならばパケットはキューに進み，
2. Q ≥ T ならばパケットは廃棄される．

フローグラフを図 4.14 に示す．

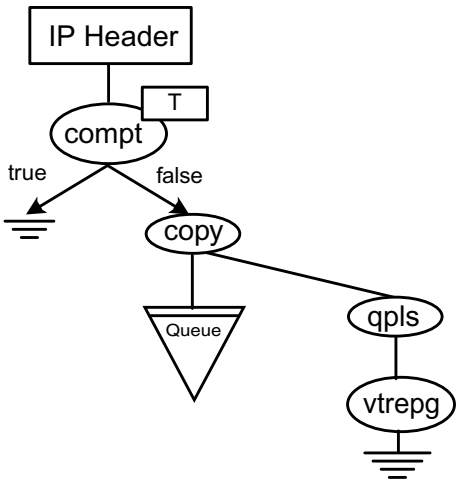

図 4.14　アルゴリズミックドロッパ

ここで，複合命令 QPLS は，IP Header の Total Length を 3 ビット左シフトしてビット数に変化し，テーブル Q を参照して加算する命令である (図 4.15).

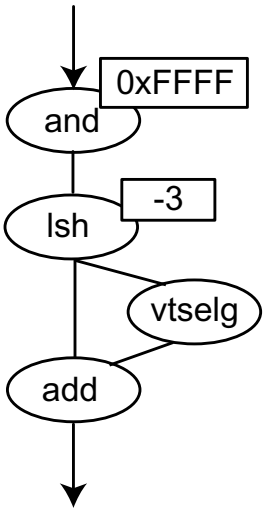

図 4.15　複合命令 QPLS

また，複合命令 COMPT は入力した IP ヘッダの Total Length と現在のキューに存在するデータのビット数の和を閾値 T と比較，選択出力する命令である (図 4.16).

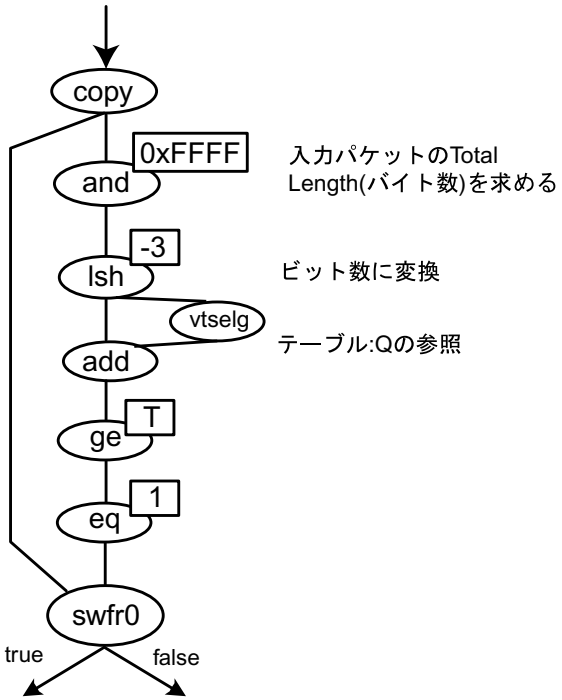

図 4.16　複合命令 COMPT

4.3.5　RED

RED では，キューに存在するデータのビット数 (queue depth) にヘッダの Total Length(第 16〜31 ビット) を加えた値 Q と最小閾値:Min および最大閾値:Max を比較し，

1. Q < Min ならば，パケットはキューに進み，
2. Q ≥ Max ならばパケットは廃棄され，
3. Min ≤ Q < Max のときは，確率的に廃棄される．

3. で確率的に廃棄するには，閾値 Max に対するキューに存在するデータの割合に乱数をかけ，その結果から廃棄するかキューに入力するか決定するので，乱数を生成する命令が必要である．ここで乱数発生命令 RND を追加する．

図 4.17 のように RND は 1 入力 1 出力命令で入力はトリガである．

新命令 RND をサポートした RED のフローグラフを図 4.18 に示す．

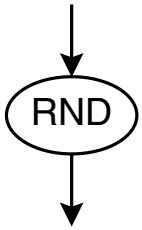

図 4.17　新命令 RND

このフローグラフでキューの閾値 Min および Max との比較の部分に複合命令 COMPT を使うとフローグラフは次の図 4.19 になる.

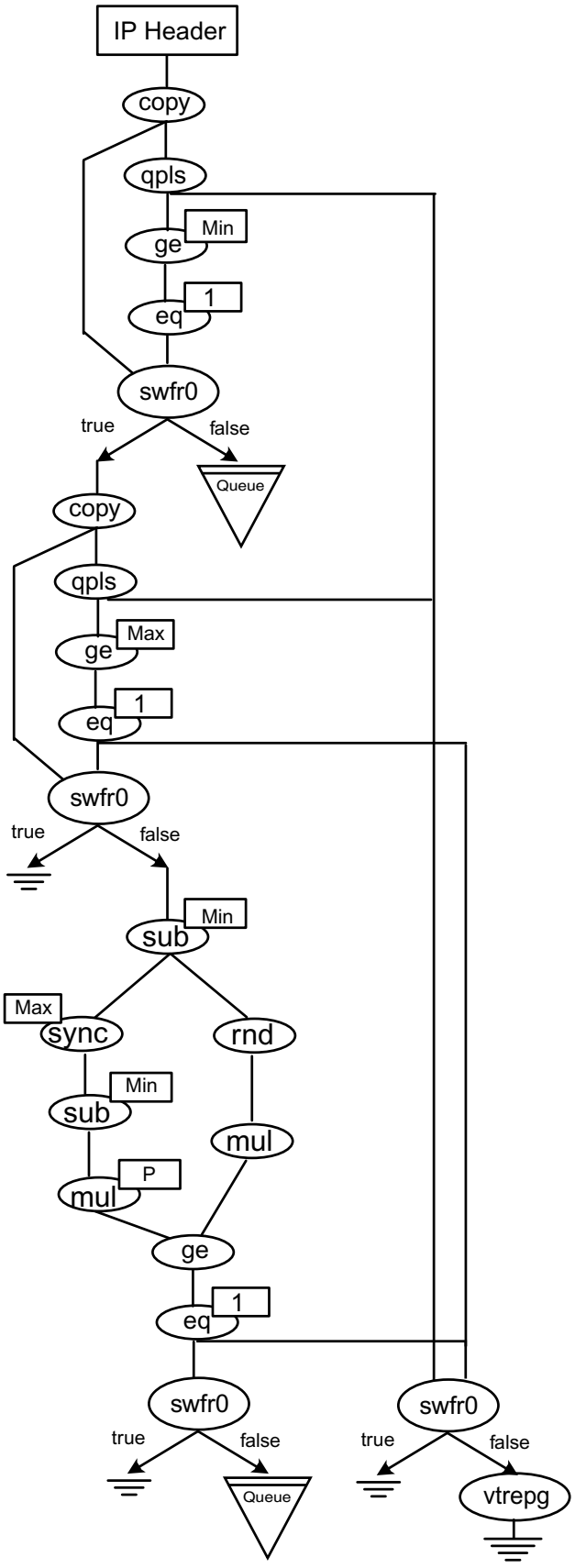
IP Header
copy
qpls
ge
Min
eq
1
swfr0
true
false
Queue
copy
qpls
ge
Max
eq
1
swfr0
true
false
sub
Min
Max
sync
rnd
sub
Min
mul
mul
P
ge
eq
1
swfr0
swfr0
true
false
true
false
Queue
vtrepg

図 4.18 RED

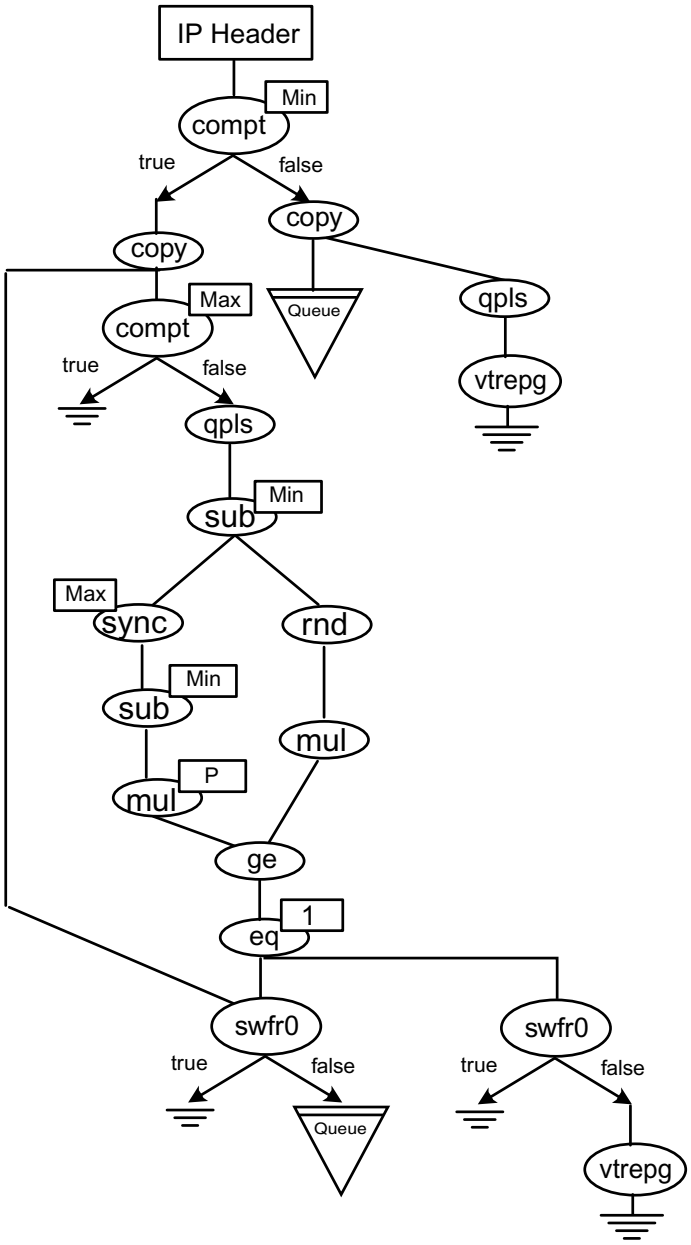

図 4.19　複合命令化 RED

4.3.6　キューの出口での処理

メモリ中で IP パケットは図 4.20 のように分割して記憶されており，同一の IP パケットから生じた DDMP パケットの LN 番号はすべて同じである．

キューの出口に PX#=0 のデータが出力されると，IP パケットを再構築するために，ヘッダの Total Length から IP パケットのバイト数を読み取り，IP パケット 1 個を構成す

GE		DATA
LN	PX	
0	0	Ver,IHL,TOS,Total Length
0	1	▪
0	▪	▪
0	▪	▪
0	n-1	▪
1	0	Ver,IHL,TOS,Total Length
▪	1	▪
▪	▪	▪

※ n=Total Length / 4

図 4.20　メモリ構造

る DDMP パケットの個数を計算する．そして，メモリから PX 番号が 0～$\frac{\mathrm{TotalLength}}{4} - 1$ まで連続して DDMP パケットを参照する必要がある．

本稿では，内蔵メモリの内容を連続して参照する新命令 TCSELG を追加する．図 4.21 で，左入力は PX#=0 の DDMP パケット，右入力は連続して参照される DDMP パケット数である．この命令により，左入力された DDMP パケットの PX 番号から，同じ LN 番号のパケットが右入力数値と同じ個数だけ PX 番号順に出力される．

この新命令を使って，IP パケットを再構築するためのフローグラフが図 4.22 である．そしてこのプログラム全体を複合命令 PKOUT とする．

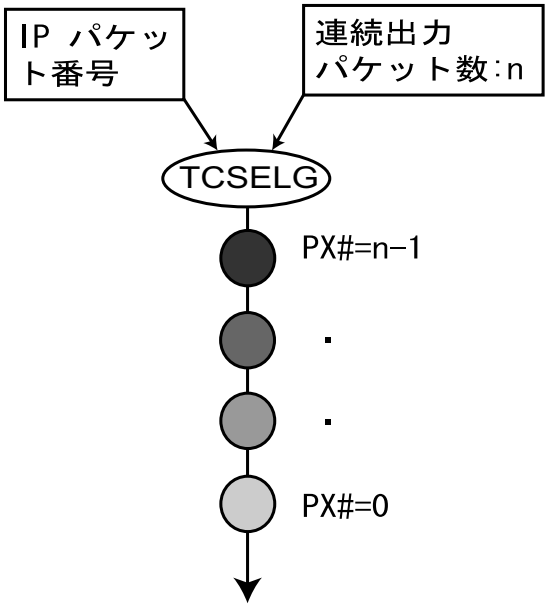

図 4.21　新命令:TCSELG

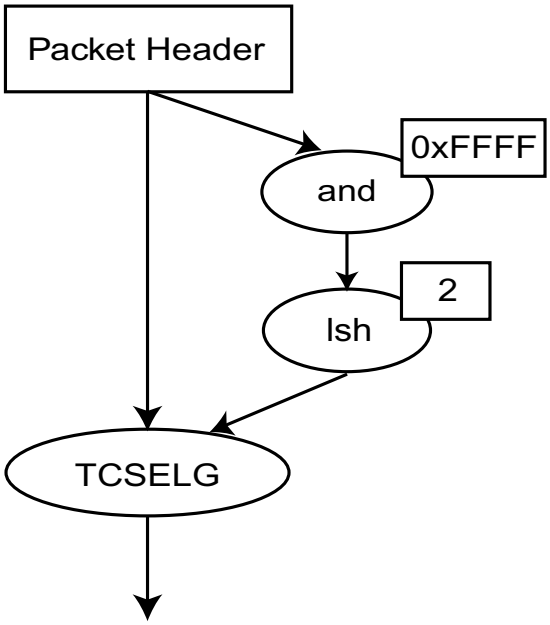

図 4.22　複合命令 PKOUT

4.4　結言

本章では，クラスベースの優先制御に DDMP-AUDIO ナノプロセッサを適用するために新命令，

- タイマ参照命令 TM
- テーブルメモリ参照命令 VTSELG
- ブランチ命令 BRD8
- 乱数発生命令 RND
- パケット連続出力命令 TCSELG

複合命令，

- マーカ MRKDP
- テーブルメモリ加算命令 QPLS
- 閾値選択出力命令 COPMT
- IP パケット再構成命令 PKOUT

および DDMP-AUDIO ナノプロセッサモジュール

- Average Rate Meter
- クラシファイヤ
- アルゴリズミックドロッパ
- RED
- IP パケット再構築

を追加，および定義した．

モジュールの定義より，QoS 制御の各要素が DDMP ナノプロセッサに適用でき，ルータの用途によって自由なモジュールの追加，機能の調整ができるようになる．複合命令を採用することによって実行命令数の削減が実現し，パケット処理能力向上が期待できる．また，新命令により，新ハードウェアの制御が可能になる．

第 5 章

性能評価

5.1 緒言

本章では,

1. 自律型優先キューと従来方式のパイプライン処理で行なうキューおよびスケジューリングのシミュレーション実験による性能評価
2. シミュレーション実験による SLA の QoS 項目の設定に関する評価
3. DDMP マルチプロセッサ複合命令に関する評価

を行なう.

5.2 自律型優先キューと従来方式の性能評価

5.2.1 実験項目

実験項目は次の 3 項目である．ただし，データは優先度を 1，2，3 の数字で表し，1000 個のデータを連続して投入し，1 回分のシミュレーションを行なう．違ったデータを 5 系列用意し，5 回の平均を結果とした．データの並び方は乱数で決定し，1，2，3 のデータの割合はほぼ等分である．パケット Total Length はすべて同じでシミュレートした．なお，自律型優先キューのパイプライン効率は 50%でシミュレートした.

1. データ通過率

   この項目は，キューの入り口まで来たパケットがキューの出口に出力される割合であ

る．クラス分けは廃棄優先度の検証を意識して，AF1，AF2，AF3 クラスでおこなった．廃棄優先度が AF3 > AF2 > AF1 となるように廃棄閾値を AF1=150，AF2=75，AF3=50 とした．

2. 遅延

   キューの入り口にパケットが入ってから出口から出てくるまでの時間の平均とした．DDMP の自己同期パイプラインの動作速度を 240M データ/秒として，データを $\frac{1}{2.4\times10^8}$ 秒のクロック周期で投入することとし，パイプライン内の各ステップはこの時間間隔で実行される．それぞれのパケットはキューに投入されたときに time=0 とされ，クロックごとに 1 プラスされる．

3. ジッタ

   遅延時間のゆれである．time の標準偏差で表される．

5.2.2 シミュレータの正当性の検証について

今回のシミュレータは Java 言語でプログラムしたものである．結果の出力形式はテキストファイルで，順に通過率，遅延時間，ジッタの順である．これを表計算ソフトで読み込み，シミュレーション結果を検討した．シミュレータの正当性を検証するために，[26] で用いられたシミュレータと同じデータでシミュレーションを行ったところ，同様の結果が得られた．これにより今回使用したシミュレータの正当性は実証されたものとする．

5.2.3 従来方式

従来方式では，廃棄制御を RED，スケジューリングを WRR でシミュレートした．キューは 3 クラスにそれぞれ長さ 200 を割り当て (IP パケットの平均長は約 420 バイト [28] なので，それぞれほぼ 84k バイトに相当する)，スケジューリングでは，出力の優先度を AF3 > AF2 > AF1 とするため，連続出力数を AF1=2，AF2=5，AF3=10 とし，最初に AF3 を 10 個出力し，次に AF2 を 5 個出力，最後に AF1 を 2 個出力するようにした．ま

た，あるクラスの出力数が連続出力数まで達していなくてもキューが空になれば次のクラスが出力するようにして，遅延時間を小さくした．通過率，遅延時間そしてジッタそれぞれのグラフを図 5.2(通過率)，図 5.4(遅延時間)，図 5.6(ジッタ) に示す．

グラフの横軸は，すべてキューへの入力レートと出力レートの比 output rate/input rate，縦軸は，図 5.2 はキューの入り口にやってきたパケット数とキューの出口に到着したパケット数の比 (単位%)，図 5.4 は遅延時間 (単位 μs)，そして図 5.6 はジッタ (単位秒) である．

5.2.4　自律型優先キュー方式に関する評価

図 5.1 は今回のシミュレーションした自律型優先キューの構造である．従来方式と違い，全クラスがひとつのキューにはいるので，パイプラインは 1 本だけである．パイプラインは 100 で折り返し，往復 200 のパイプライン長とした．キューに入ったパケットは，パイプラインを進んでいくが，入り口での廃棄閾値はパイプライン中の各クラスのパケット数が AF1=150，AF2=75，AF3=50 で従来方式と同じ閾値でシミュレーションした．そして，パ

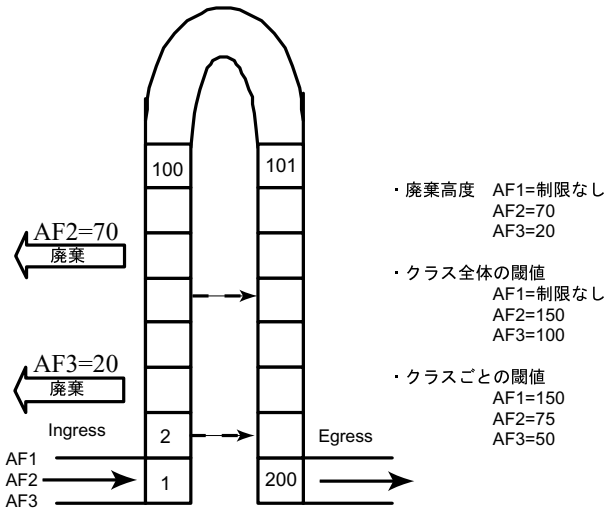

図 5.1　シミュレーション

イプラインが 1 本なので，その中の全体のパケット数も考慮することにして，閾値を AF1=制限なし，AF2=150，AF3=100 としてシミュレーションを行なった．その結果を図 5.3(通過率)，図 5.5(遅延時間)，図 5.7(ジッタ) に示す．

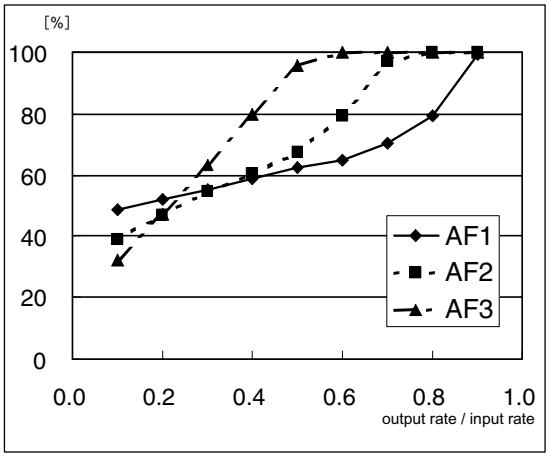

図 5.2　RED 付き WRR(通過率)

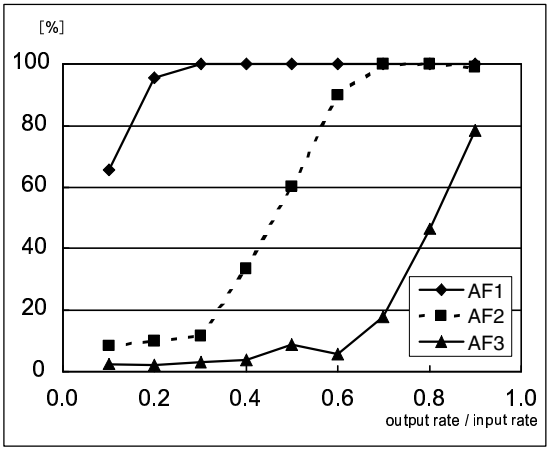

図 5.3　自律型優先キュー (通過率)

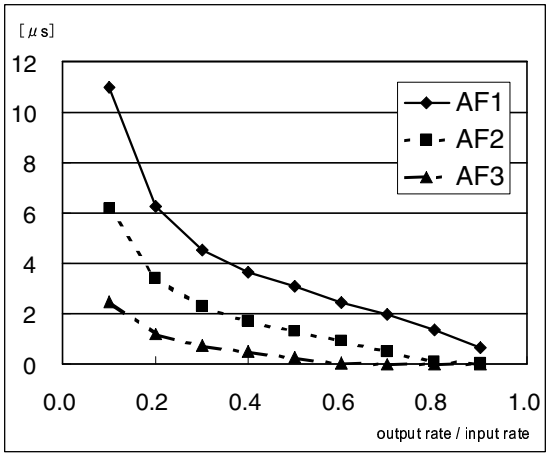

図 5.4　RED 付き WRR(遅延時間)

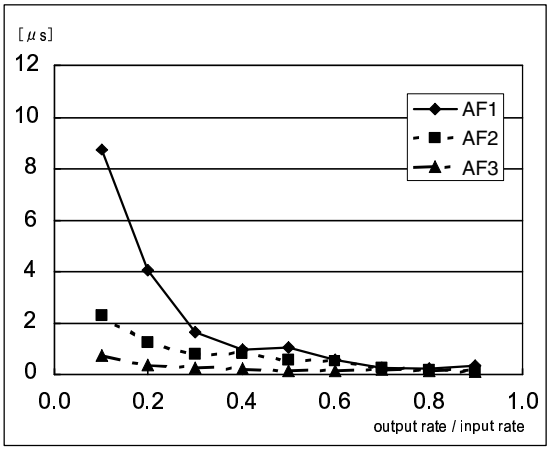

図 5.5　自律型優先キュー (遅延時間)

グラフの横軸，縦軸の単位は従来方式の評価グラフと同じである．また，従来方式との比較が容易にできるよう，同じ項目を横に並べた．

このグラフからわかることは，

1. 通過率では，従来方式は差別化がうまくできていない，特に出力レートが割合大きいとき，つまりリンクの輻輳の度合いが比較的小さいときの通過率の順番が廃棄優先度の逆になっていてうまく優先制御ができていないことがわかる．それに比べ，自律型優先キューは入力レートと出力レートの比がどのようになっても，廃棄優先度が高いほど (AF3 > AF2 > AF1) 通過率が小さくなっており，クラスによる廃棄率の差別化ができ

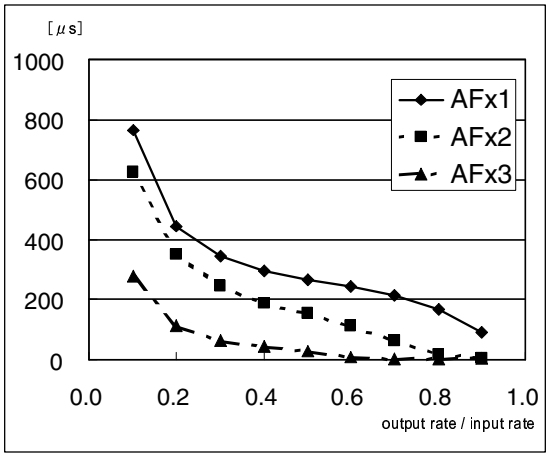

図 5.6　RED 付き WRR(ジッタ)

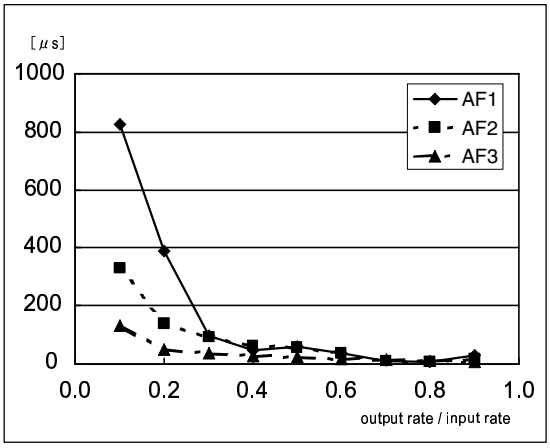

図 5.7　自律型優先キュー (ジッタ)

ていることがわかる．

2. 遅延時間では，従来方式，自律型優先キューともに優先度の順番 (AF3 > AF2 > AF1) の逆になっており (AF1 > AF2 > AF3)，どちらもクラスによる差別化ができている．しかし，その絶対量は自律型優先キューのほうが小さくなっており，従来方式に比べ，約 1/3 になっている．特に AF2，AF3 で性能が良いことがわかるが AF1 はそれほど性能の向上が見られない．これは，優先度の低い AF1 は追い越しができなくてパイプラインの高いところまで上ってしまい，なかなか降りてこられないことことが理由であろう．
3. ジッタでも遅延時間と同じような傾向が出ているが，ここでは出力側の輻輳の度合いが比較的小さいときの改善率が大きい．AF1，AF2 では約 1/10 に改善されている．しかしながら，AF1 は出力側の輻輳の度合いが大きいときは従来方式に比べて，かえって大きくなっている．これは遅延時間の場合と同じ理由であろう．

5.2.5　偏りのあるフローの場合の性能

上の結果は，入力されたパケット中のクラスの数比がほぼ AF1:AF2:AF3=1:1:1 のものである．しかし，ネットワークを流れるデータはこのようにクラス比が均等になっているとは考えにくい．したがって，各クラスの帯域保証制御も自律的におこなえるのではないかと

いう期待と，クラス比の偏りが優先制御や廃棄制御に与える影響を確認するために，次のように各クラスに偏ったデータを与えてシミュレーションを行なった (パラメータは 5.2.4 と同じにした).

1. AF1:AF2:AF3=8:1:1

優先度の低いクラスのデータが多く流れているときのシミュレーションである．図

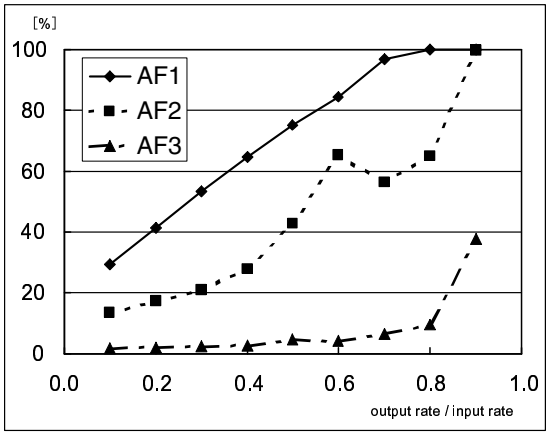

図 5.8　閾値 AF1:AF2:AF3=200:150:100(通過率)

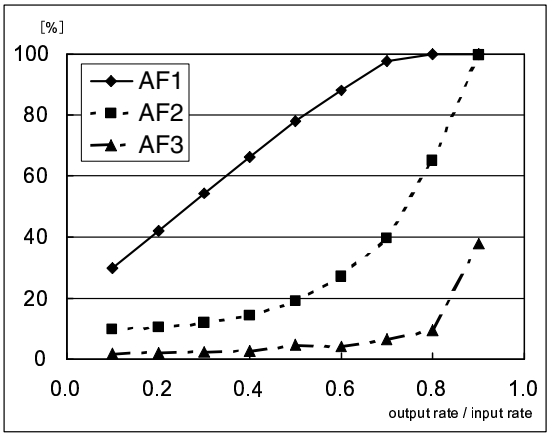

図 5.9　閾値 AF1:AF2:AF3=200:125:100(通過率)

5.8(通過率) で出力側のレートが低いときの通過率が均等のデータのとき (図 5.3) に比べて悪くなっているのがわかるが，これは AF1 の閾値が高いといっても限りがあるので，仕方のないことである．また，AF2 の通過率に非連続な部分がある (レート比 0.6 ～0.8 の間) が，これは AF2 の閾値を 150 → 125 に下げることで緩やかなカーブに変わる (図 5.9)．また，遅延時間やジッタに影響はない (図 5.10，図 5.11).

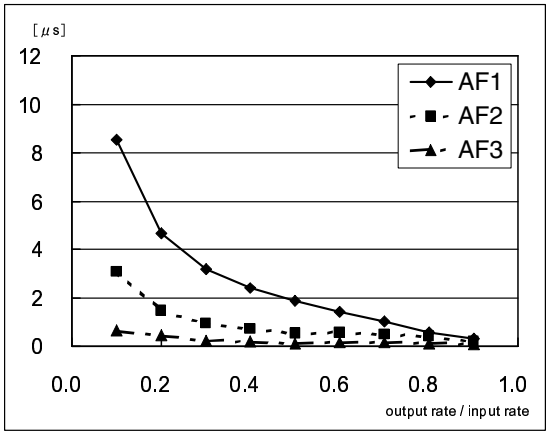

図 5.10　遅延時間

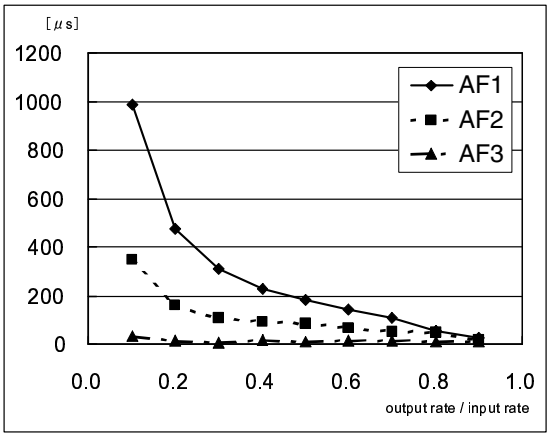

図 5.11　ジッタ

2. AF1:AF2:AF3=1:8:1

AF2 に偏ったデータの場合，通過率 (図 5.12) は均等なデータのグラフに似ているが，AF1 はデータ数が少ないので，100%通過していて，帯域が確保されている．遅延 (図 5.13) およびジッタ (図 5.14) は全体に小さくなってきているが，ジッタの AF1 と AF2 の大小が逆転している．これは，AF2 のデータが入力されてすぐに出力されるものと，パイプラインの上の方まで上るものとに分かれているためである．

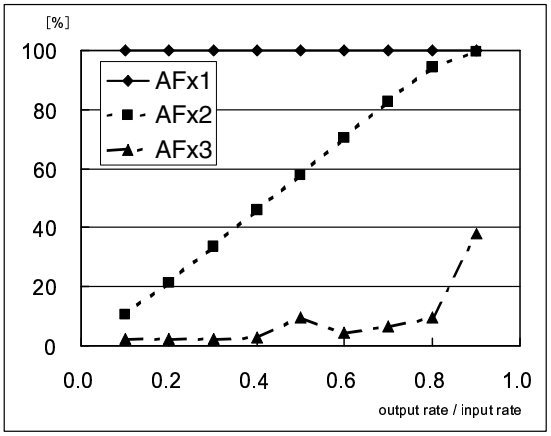

図 5.12　AF1:AF2:AF3=1:8:1(通過率)

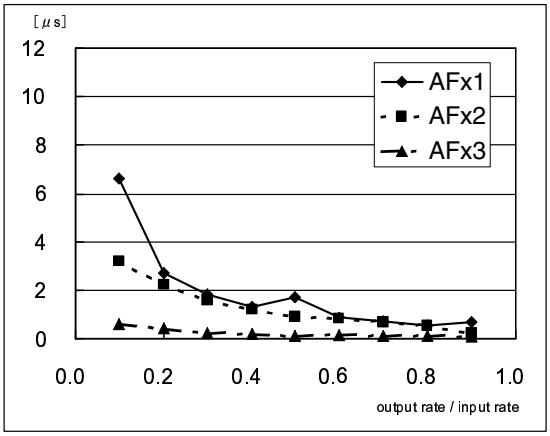

図 5.13　AF1:AF2:AF3=1:8:1(遅延)

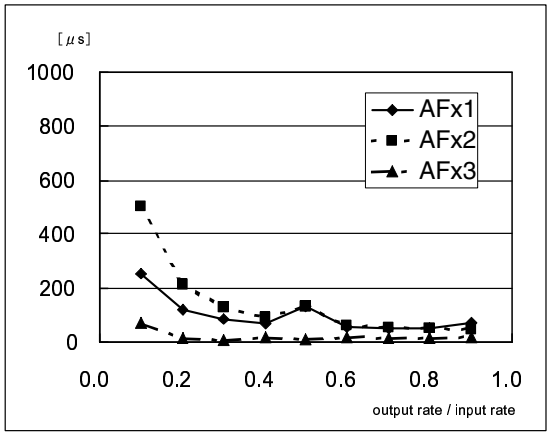

図 5.14　AF1:AF2:AF3=1:8:1(ジッタ)

3. AF1:AF2:AF3=1:1:8

AF3 に偏ったデータの場合は，AF1 の通過率 (図 5.15) は 2. の場合と同じく 100%通過している．今回はそれに加えて，AF2 もほぼ 100%通過している．これは，データ自体の数が少なく，閾値まで達しないからであろう．

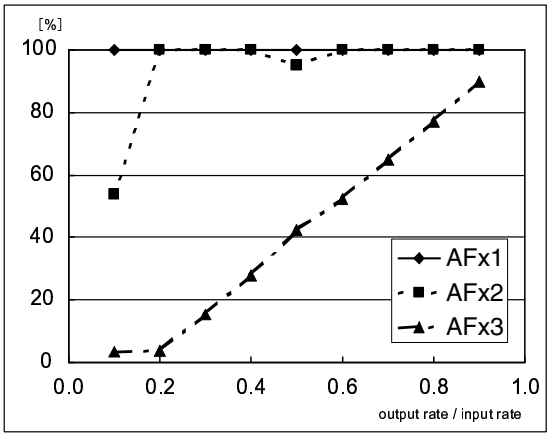

図 5.15　AF1:AF2:AF3=1:1:8(通過率)

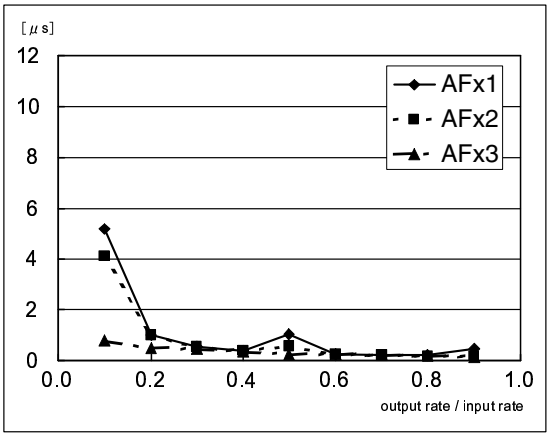

図 5.16　AF1:AF2:AF3=1:1:8(遅延)

AF3 の通過率は，レート比が 0.2 から増えるにつれてほぼ直線的に上昇している．これは他のクラスのデータはレート比が小さいときからほぼ 100%通過しているので，レート比の上昇が AF3 の通過率の上昇につながったからであろう．AF1，AF2 ともに優先度の高いデータのバーストに対してほとんど影響を受けていないので，帯域

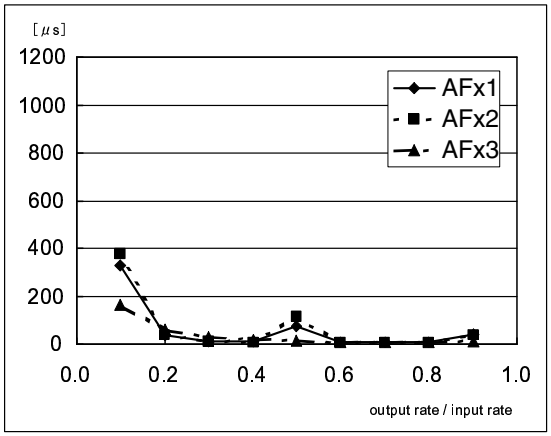

図 5.17　AF1:AF2:AF3=1:1:8(ジッタ)

保証制御が容易だと思われる．すなわち，優先度の高いフローの監視の役目もこの自律型優先キューが持っていることになる．遅延 (図 5.16)，ジッタ (図 5.17) ともに AF1:AF2:AF3=1:8:1 のときの遅延およびジッタに比べ減少している．

5.2.6　パイプライン効率に関する評価

今回のシミュレーションは，パイプライン効率 50%で行ったが，現行の DDMP のパイプライン効率は 80%である．今回の結果からパイプライン効率 80%の場合の評価を行う．

パイプライン効率 50%に比べ 80%ではパイプライン中のデータ数が 1.6 倍になるので，出力側で輻輳しなければ，最大 1.6 倍のスループット向上が得られる．また，データの追い越しも頻繁になるので，優先制御も効率よく行われる．一方，出力側が輻輳した場合でも，自己同期パイプライン中のデータは前が空いていれば前進できるので，最終的にはどちらも同じパイプラインの状態になり，最悪でも同等のスループットである．また，出力側が輻輳状態から回復したとき，処理遅延より転送遅延が小さいので，パイプライン効率 80%は 50%に比べ，パイプラインの込み具合が早く回復する．そのため，パイプラインの入り口での廃棄が早く終り，通過率は向上する．同時に遅延時間，ジッタも小さくなると予想される．

5.3 SLA 用のパラメータの設定に関する評価

5.3.1 実験方法

折り返し型パイプライン機構の“ある高さまで上ったパケットを選択的に消去できる”という特徴を活用して，クラスによって到達できる高さを変えて，パイプラインの中での優先度の差別化をさらに細かく調整できるようになる (図 5.18)．このパラメータを AF1 では高さによる廃棄をさせずに無制限にし，AF2，AF3 をともに高さ 2〜100 の間で到達可能高度を変化させると同時にパイプライン中の全パケット数の廃棄閾値を次のように細かく設定してシミュレーションをおこなった．SLA(14 ページ 2.3.5 参照) に基づく QoS 制御が自己同期パイプラインのパラメータ調整によって可能であることを示す．

$$(\mathrm{AF1}, \mathrm{AF2}, \mathrm{AF3}) = \begin{matrix} (200,100,100) & (200,100,125) & \cdots & (200,100,200) \\ (200,125,100) & (200,125,125) & \cdots & (200,125,200) \\ \vdots & \vdots & \vdots & \vdots \\ (200,200,100) & (200,200,125) & \cdots & (200,200,200) \end{matrix}$$

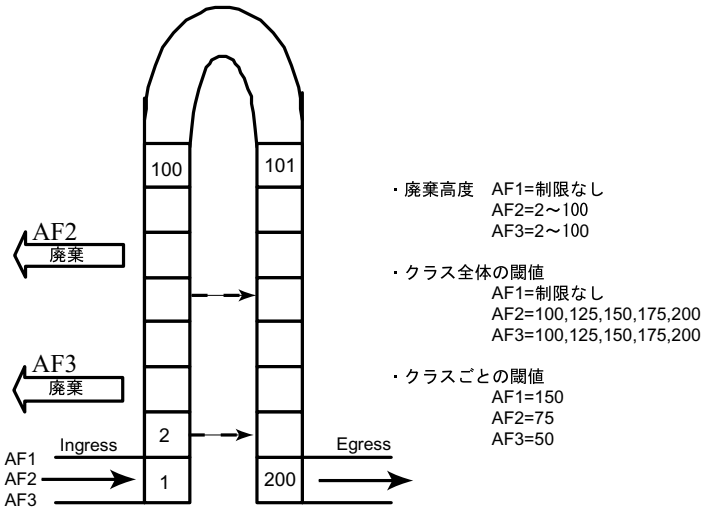

図 5.18 SLA シミュレーション

今回のシミュレーションは，入力レートと出力レートの比:output rate / input rate=0.30 とし，出力リンクが輻輳しているときの状況をシミュレートした．キュー中のクラス全体の

閾値を変えていき，それぞれの閾値において廃棄高度をクラスごとに連続的に変化させてシミュレーションをおこない，通過率，遅延時間，ジッタについて評価する．

5.3.2　シミュレーション結果

図 5.19 および図 5.20 は通過率のグラフである．図中の廃棄高度は折り返しパイプライン中でクラス別に到達可能な高度を表す．図 5.19 は AF2 クラスは 90，AF3 クラスは 40 まで上ると廃棄され，図 5.20 では AF2 クラス，AF3 クラスとも 100 まで上ると廃棄される．また，AF2 および AF3 閾値はキューの入り口にそれぞれのクラスのパケットが到着したときにキューに入力されるか廃棄されるかの閾値で，全クラスのパケットの総数がこの値以上になるとそのパケットはキューの入り口で廃棄される．

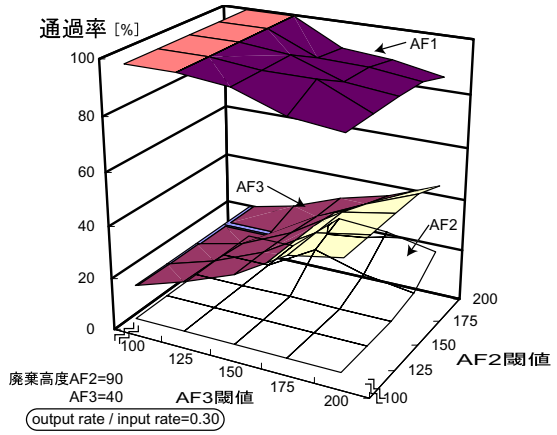

図 5.19　通過率の変化 1

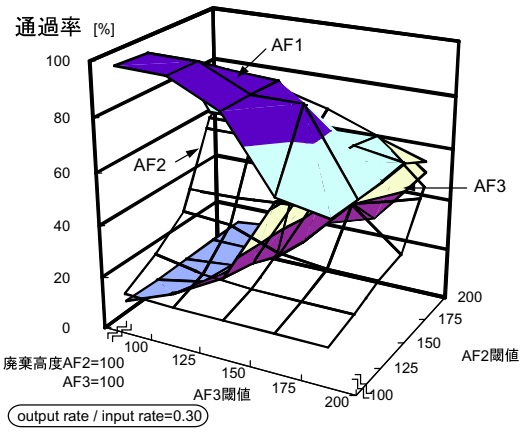

図 5.20　通過率の変化 2

図 5.21 および図 5.22 は遅延時間，図 5.23 および図 5.24 はジッタの変化を同じパラメータでシミュレーションしたものである．

5.3.3　評価

図 5.19 では，各クラスの閾値の大小に関わらず，通過率がほぼクラスの通過優先順序どおりになっている．これは，輻輳している状態ではキューがほぼつまった状態が続いているので，通過率はクラスの到達できる高度が高いほど大きく，低いほど小さいということがわ

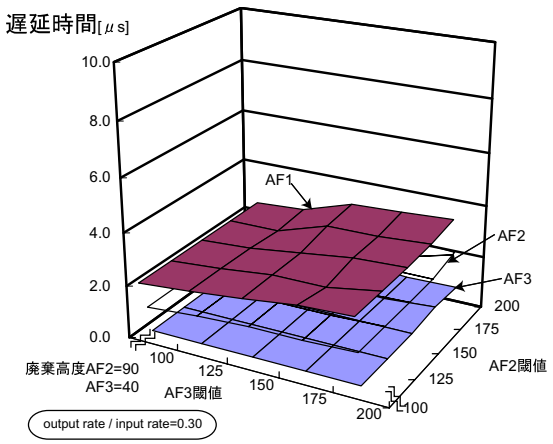

図 5.21　遅延時間の変化 1

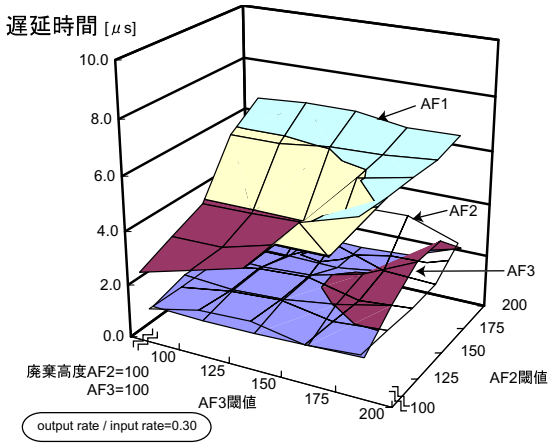

図 5.22　遅延時間の変化 2

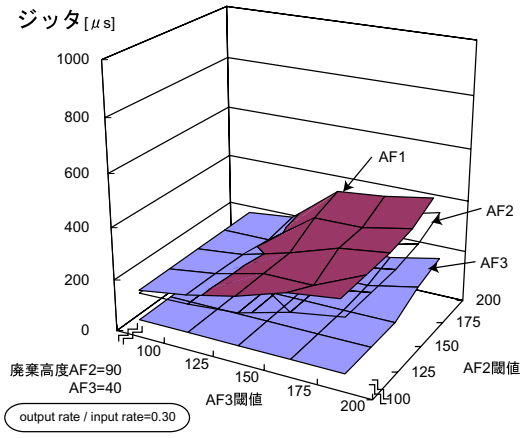

図 5.23　ジッタの変化 1

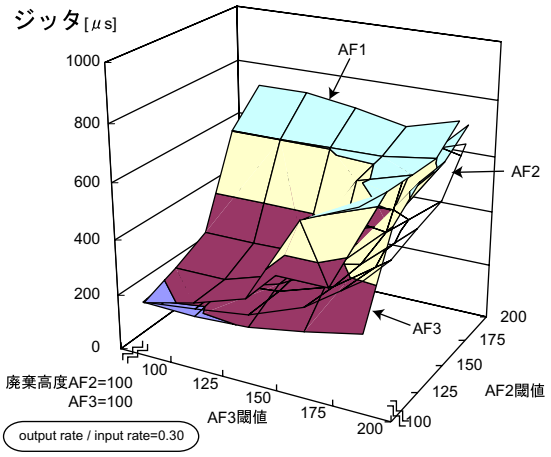

図 5.24　ジッタの変化 2

かる．これは，図 5.21 の遅延時間，図 5.23 のジッタのグラフでも優先度どおりになっている傾向が出ていることから確かめられる．

一方，廃棄高度が制限いっぱいになっているときには通過率，遅延時間，ジッタに直接影響をおよぼすパラメータはキュー中の各クラスの閾値になる．各 QoS 要素はこの閾値によって大きく変化する．例えば，図 5.20 では，各クラスの閾値が大きくなると各クラスの通過率が優先度どおりでなくなり，逆転する現象が起こる．これは一面から言えば非常に都合の悪い現象になるが，パラメータの変化に敏感に反応していると考えれば，これらのパラメータをリンクの輻輳状態に応じて設定すれば，QoS 項目の細かい調整が可能であるという

ことになる．SLA においては，リンクの状態に応じて厳密な運営が必要であり， パイプラインの廃棄高度と，クラス毎の閾値による調整ができれば実用化もできると思われる．

5.4 DDMP マルチプロセッサ複合命令に関する評価

5.4.1 DDMP シミュレータの原理

本研究のシミュレーションは，DDMP-AUDIO のフローグラフシミュレータで行なう．このシミュレータは，DDMP チップの構成を基にした，グローグラフシミュレーションツールである．フローグラフプログラムを作成し，シミュレータがこれを解析して演算する．入力データは世代番号とエントリデータを持つパケットデータであり，本研究では 1 パケットのデータを IP パケットデータ 32 ビット分として，IP パケットを $\frac{\mathrm{TotalLength}}{4}$ 個の DDMP パケットに分割している．

5.4.2 シミュレーション手法

1. 従来方式での性能

   従来の命令で構成したプログラムの性能をシミュレータで評価する．

2. 複合命令での性能

   第 4 章で提案した複合命令で構成したプログラムの性能を見積もる．

3. 比較

   従来方式での性能と複合命令での性能を比較する．

5.4.3 測定方法

測定式は以下のものとする．

1. 命令数

   プログラム中で，データが通過する命令数．

2. スループット

スループットは，1 秒あたりの実行命令数 t を処理を始めてからパケット出力までに必要な命令数 C で割ったものになる (式 5.1)．単位は pps $(packets/sec)$ である．

$$\text{Throughput} = \frac{\text{t}}{\text{C}}\ (pps) \tag{5.1}$$

3. レスポンスタイム

1 データが入力されてから出力されるまでに要した時間で，単位は ns (ナノ秒) である．

1. は各命令に実行時間が割り振られており，命令時間の長い命令を多用したり同じ処理を繰り返し実行したりするとデータフローレートが低下する．このため，命令数がプログラムの実行速度を左右する．

2. はコンピュータシステムの処理能力を測る評価指標であり，本研究では，一定時間に何パケットのデータの処理ができるかを測定する．

3. も性能評価指標の一つで，データが入力されてから出力されるまでの時間のことで，本研究では遅延時間の測定をこの指標で行なう．

5.4.4 クラシファイヤ

従来の命令セットでクラシファイヤを構成すると，あるクラスを分類するためだけに図 5.25 のように 4 命令必要である (この図では EF クラスのヘッダが true ノードに出力される)．

表 5.1 命令数の比較

	最大命令数	平均命令数
従来方式	20	12
複合命令	4	4

また，別のクラスを分類するためには，eq 命令の定数を変えたものを別に用意して，図

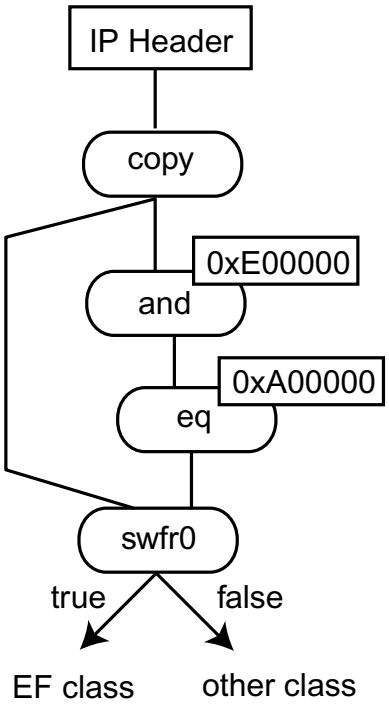

図 5.25 従来方式のクラシファイヤ

5.25 の false ノードにつなぐ必要がある．クラスの数が増えれば増えるほどクラシファイヤの段数が増えて，遅延が大きくなる．また，クラスの構成を変えたりする場合，あるいは将来，クラス分けの方法が変わった場合には，プログラムしなおすことが必要である．図 5.26 に全クラスの分類を行なうフローグラフを示す．

このフローグラフの処理を複合命令では，48 ページの図 4.12 の 3 命令で行なうのである．表 5.1 は従来方式と複合命令の性能を最大命令数 (BE クラス) と平均命令数で比較したものである．結果は，最大命令数のときで約 $\frac{1}{7}$，平均命令数のときで $\frac{1}{4}$ の命令数になる．

また，表 5.2 はスループットの見積もりである．

表 5.2 スループットの見積もり

	最小スループット	平均スループット
従来方式	1.20×10^7 *pps*	2.00×10^7 *pps*
複合命令	6.00×10^7 *pps*	6.00×10^7 *pps*

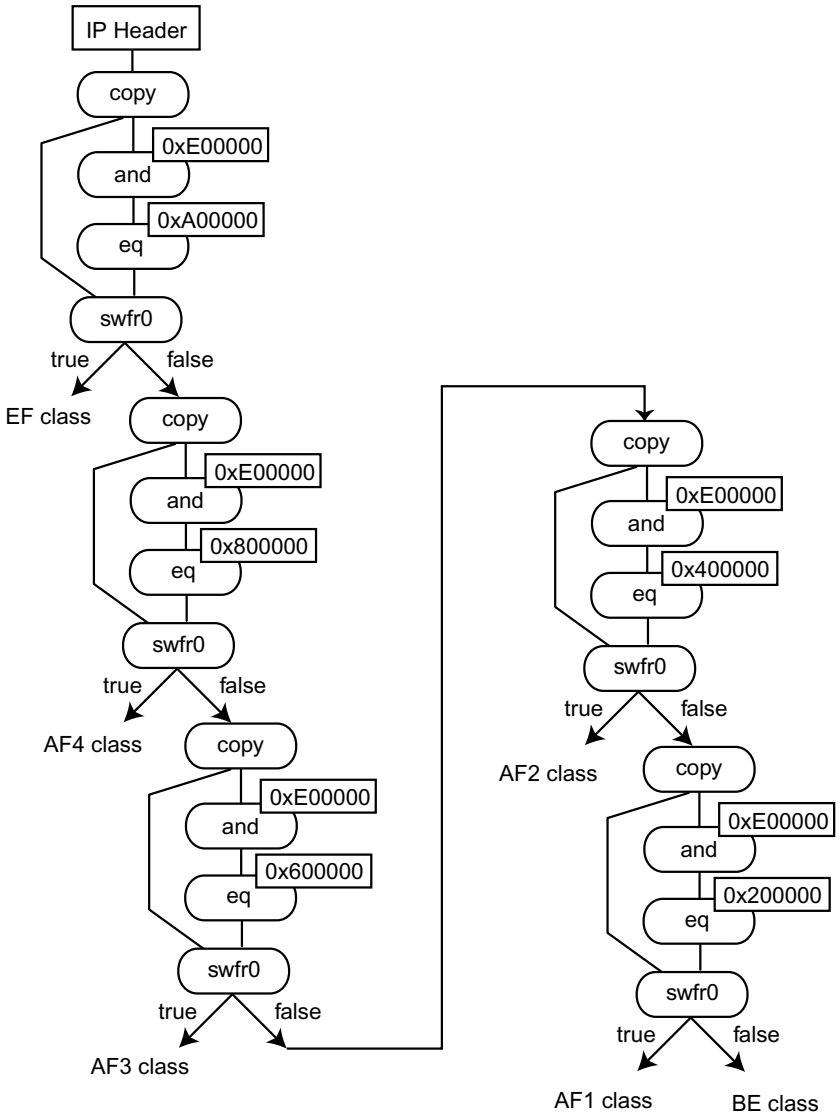

図 5.26　全クラス用のクラシファイヤ

5.4.5　Average Rate Meter

Average Rate Meter にはハードウェアタイマのサポートが必要であり，従来の方法では実現不可能であった．したがってここでは，性能比較はおこなわず，複合命令での命令数およびスループットの見積もりを行なう．

46 ページ図 4.9 および 47 ページ図 4.10 より，最大命令数は 11 命令，平均では 9.5 命令である．表 5.3 に命令数とスループットの見積もりを示す．

表 5.3 Average Rate Meter の性能見積もり

	最小	平均
スループット	2.18×10^7 *pps*	2.53×10^7 *pps*
命令数	11	9.5

5.4.6 アルゴリズミックドロッパ

アルゴリズミックドロッパでは，従来方式 9 命令，複合命令は 4 命令である．表 5.4 に命令数およびスループットの見積もりの比較を示す．

表 5.4 命令数およびスループットの比較

	命令数	スループット
従来方式	9	2.67×10^7 *pps*
複合命令	4	6.00×10^7 *pps*

5.4.7 RED

RED では，従来方式では最大 21 命令，平均 20 命令である．一方，複合命令では，最大 14 命令，平均 9 命令である．表 5.5 に命令数，表 5.6 にスループットの見積もりの比較を示す．

表 5.5 命令数の比較

	最大命令数	平均命令数
従来方式	21	20
複合命令	14	9

表 5.6 スループットの見積もり

	最小スループット	平均スループット
従来方式	1.14×10^7 pps	1.20×10^7 pps
複合命令	1.71×10^7 pps	2.67×10^7 pps

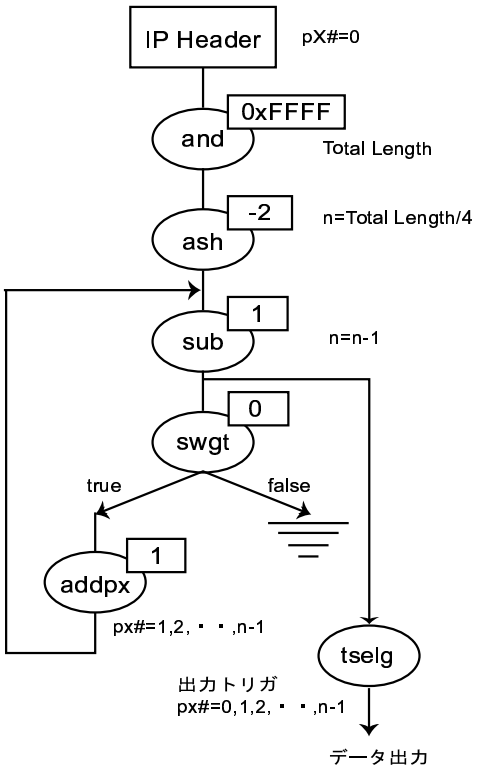

図 5.27 IP パケット出力

5.4.8 IP パケット再構築および出力

新命令 TCSELG(56 ページ図 4.21 参照) では左入力に IP パケット番号，右入力に連続出力する DDMP パケット数を入力すれば，メモリから IP パケット 1 個分のデータが連続して出力されるが，従来の方法では，図 5.27 のように，IP ヘッダ中の Total Length を 4 で割り，px 番号を 1 ずつ加えながらループさせてトリガを出力し，それによってメモリから連続してデータを読み出す．このとき問題になるのは，メモリからデータを読み出すのに，トリガ 1 個の出力に 3 命令，さらにメモリ参照命令 tselg も加えると 4 命令必要なことであ

る．例えば，平均パケット長である 420 バイトの IP パケット [28] を 1 個出力するためには $\frac{420}{4} \times 4 + 2 = 422$ 命令必要である．これを新命令では 56 ページ図 4.22 からわかるように，3 命令で実行できる．さらに複合命令化すると 1 命令になる．つまり，IP パケットのサイズが大きいほど効率が良くなるのである．表 5.7 に命令数およびスループットの見積もりの比較を示す．

表 5.7　命令数およびスループットの比較

	命令数	スループット
従来方式	422	5.69×10^5 *pps*
複合命令	1	2.40×10^8 *pps*

5.4.9　総合性能

DDMP ナノプロセッサ 1 個に QoS 制御用機能をプログラムすることによって表 5.8 の性能を達成できる見通しが得られた．

表 5.8　性能の見積もり

	最小	平均
スループット	1.12×10^7 *pps* 37.6 *Gbps*	1.78×10^7 *pps* 59.8 *Gbps*
命令数	21.5	13.5

表より，平均で 59 *Gbps* 以上の性能が期待でき，ネットワーク伝送路の広帯域化に十分対応できることが実証された．

今回の性能見積もりは，DDMP ナノプロセッサ 1 個にプログラムした場合の見積もりだが，プログラムするナノプロセッサの数を増やし，それぞれのナノプロセッサの処理負荷を適切に分散すれば性能はさらに向上する．そのためには，キューの深さや入力ストリーム中の IP パケットのクラス構成比の動的な変化にリアルタイムに対応して QoS 制御のパ

ラメータを更新するすることが必要である．入力ストリームのクラス毎のビットレートは，Average Rate Meter によって計測可能であり，その値の変化はリアルタイムで内部テーブルメモリに書き込まれる．また，キューの深さも Average Rate Meter と同じく内部テーブルメモリにリアルタイムに書き込まれるので，その値を各機能要素が参照することで，最適なパラメータに更新できる．したがって，ハードウェアを追加することなく各ナノプロセッサの処理コストを平均化して，動的に処理遅延を最小化することは可能であり，複数のナノプロセッサに個々の機能要素をプログラムし，処理コストを最小にすることで，最大のスループットを得ることは可能である．

5.5 結言

以上，本章では次の 3 項目，

1. 自律型優先キューと従来方式のパイプライン処理で行うキューおよびスケジューリングのシミュレーション実験
2. 1．のシミュレーション実験による SLA の QoS 項目の設定に関する評価
3. DDMP マルチプロセッサ複合命令に関する評価

の性能評価を行った．

その結果，1．では，

- 廃棄優先度による通過率の差別化が複雑なスケジューリングなしで実現する．
- キュー内の遅延時間については，クラスごとの違いを保ちながら全体に小さくなり，処理能力が向上している．
- ジッタについても遅延時間と同じ傾向である．
- パイプライン効率 80%の状況下ではシミュレーションを行ったパイプライン効率 50%に比べ，スループットでは最大で 1.6 倍，最悪で同等の性能が得られ，また，デー

タ通過率，遅延時間およびジッタについても同等以上の性能が得られると思われる．

同じく 2．では，

- クラスによる差別化をはっきりと実現したいときにはキュー中の各クラスの到達可能高度の差を大きくつける．
- QoS 項目を細かく調整したい場合には，到達可能高度を各クラスとも大きくとり，キュー内の閾値を調整すればよい．

最後に 3．では，

- DDMP ナノプロセッサにルータの構成要素を割り当てることによって，柔軟な優先制御がパラメータの調整によって高速に処理できる．
- ハードウェアサポートおよび新命令によってさらに高速処理が可能である．
- 入力ストリームのクラス比の動的な変化にリアルタイムに対応して，処理の負荷を各ナノプロセッサに均等に分散することで，処理遅延を最小にすることが可能である．

という結果が得られた．

第 6 章

おわりに

多様な情報通信サービスの要求に対して，光伝送技術の進展によってネットワークの帯域幅の拡大が著しく，その結果，ネットワークトラフィックが飛躍的に増加し，同時に，多様な QoS の提供が可能になりつつある．本研究では，近い将来ルータでの処理がネットワークのボトルネックになるという見込みのもと，QoS 制御を広帯域伝送路に応じた速度で高速かつ柔軟に実現するため，ルータにクラスベース QoS 制御機構を適用する方法について検討をおこなった．

クラスベースの優先処理を高速かつ柔軟に実現するため，まず新しいキューの実現法を提案した．従来から行われてきた方法は，処理に柔軟性がなく，また全てのクラスに独立したキューが必要でメモリの利用効率も劣るとされてきたハードウェア実現法や，プログラムによって柔軟な処理はできるが，優先制御，スケジューリングおよびキュー管理が複雑になり，高速化が見込めないソフトウェア実現法であった．それに代わり，本研究で提案した実現法は，従来の同期回路ではなく，折り返し型自己同期パイプライン方式をキューに適用したものである．

この方法では，キュー内の各パケットが自身の内部に持つ DSCP の優先度に応じて自律的に待ち行列を形成できるのである．本研究ではこのキューを自律型優先キューとよぶ．このキューの実現方式により，従来型のパイプライン機構では，キューの出口で WRR や WFQ などのスケジューラで実現されるが，管理が複雑で，遅延の大きい，優先度に基づくパケットの追い越しを，キューの内部で高速かつ柔軟に行えることを示した．またパケットの廃棄処理も，従来のキューの入り口における RED 等の処理に加え，キューの内部に折り返し点を設け，そこまで上ったパケットを廃棄させることにし，これにより遅延時間および

ジッタを小さく抑えることができることを示した．さらにこの折り返し点をクラス毎に調節することで，パケットの廃棄優先度に基づく廃棄率の差別化を実現した．同時に遅延時間およびジッタの減少にも適していることを示した．また，この機構を活用して，Diffserv 等のクラスベース QoS 制御を実現するため，データ駆動型マルチプロセッサの QoS 制御用命令セットを追加した．一方，ソフトウェアでは実現できない機能要素については必要なハードウェアおよび命令を追加し，さらに複数の命令をまとめて複合命令化することで命令数の削減を行った．そして QoS 構成要素を DDMP ナノプロセッサ上に実現するモジュールを提案した．これによって QoS ポリシーに応じて DDMP ナノプロセッサにソフトウェアで機能を追加することが可能になり，また分散的な処理を行うことで遅延を局所化し，さらなる高速化が可能になる．

自律型優先キューがクラス毎に QoS を差別化できることは，従来方式による QoS 処理とパケット通過率，遅延時間およびジッタを比較するシミュレーション実験で実証された．まず通過率においてはキュー内のデータの閾値を，廃棄優先度の高いクラスほど小さく設定することでクラス毎に差別化できることがわかった．また，遅延時間およびジッタについても従来方式に比べ一部の例外を除き，約 1/3 から 1/10 に改善されることがわかった．特定のクラスにパケットが偏った場合も割合が均等な場合と同様，クラスによる通過率の差別化や遅延およびジッタの減少が確認できた．キュー内での折り返し点の調節をクラス毎に細かく行なうことで，QoS をさらに細かく調整でき，そのことから，ネットワーク間の SLA に応じたクラス別 QoS 制御 (帯域制御，遅延時間/ジッタ制御，廃棄制御) が広い範囲で実現できることを確認した．

現行の DDMP のパイプライン効率である 80%において，今回シミュレーションしたパイプライン効率 50%に比べ，パイプライン中のデータ数が 1.6 倍になるので，出力側で輻輳しなければ，最大 1.6 倍のスループット向上が得られる．また，データの追い越しも頻繁になるので，優先制御も効率よく行われる．一方，出力側が輻輳した場合でも，自己同期パイプライン中のデータは前が空いていれば前進できるので，最終的にはどちらも同じパイプラインの状態になり，最悪でも同等のスループットである．また，出力側が輻輳状態から回復し

たとき，処理遅延より転送遅延が小さいので，パイプライン効率 80%は 50%に比べ，パイプラインの込み具合の回復がはやい．そのため，パイプラインの入り口での廃棄が早く終るので，通過率は向上する．同時に遅延時間，ジッタも小さくなる．

DDMP のプログラム可能なナノプロセッサに QoS 制御に必要な機能を割り当てることで転送レートの制限やキューの閾値等のパラメータによる柔軟な処理がおこなえる．そのため，ハードウェアタイマやテーブルメモリおよび新しいハードウェアを制御する新命令の提案を行った．これにより，従来できなかった転送レートを計測するメータが実現でき，さらに柔軟な優先制御をおこなうことができる．また，新命令および複合命令を実装することよってクラシファイヤ，Average Rate Meter，アルゴリズミックドロッパ，RED 等 QoS 処理のボトルネックになる恐れのある部分の高速化が可能になった．これらの機能要素を，1 個のナノプロセッサにプログラムした場合，総合性能が平均約 59 $Gbps$ であることを命令数の見積もりから示した．

さらに性能を向上させるためには，ルータの各機能要素の処理負荷を平均化して，時間コストを下げる必要がある．そのためには，キューの深さや入力ストリーム中の IP パケットのクラス構成比の動的な変化にリアルタイムに対応して QoS 制御のパラメータを更新するすることが必要である．入力ストリームのクラス毎のビットレートは，Average Rate Meter によって計測可能であり，その値の変化はリアルタイムで内部テーブルメモリに書き込まれる．また，キューの深さも同じく内部テーブルメモリにリアルタイムに書き込まれるので，その値を各機能要素が参照することで，最適なパラメータに更新できる．したがって，ハードウェアを追加することなく各ナノプロセッサの処理コストを平均化して，動的に処理遅延を最小化することは可能であり，複数のナノプロセッサに個々の機能要素をプログラムし，処理コストを最小にすることで，最大のスループットを得ることは可能である．

最後に今後の課題であるが，今回検討したルータの構成要素はクラスベースの QoS 制御全体を網羅しているわけではなく，境界ルータにおける高速パケット分類の方法 [29] の検討が残されている．これは，ネットワークに入ってくるフローをどのような基準をもとに分類するのか，そのルールについての検討が必要である．クラス分類のために参照するヘッダ

フィールドを IP ヘッダではどれにするか，TCP ヘッダではどのフィールドにするか，また UDP ヘッダではどうするか，また，例えばアプリケーション層のような，さらに上位のネットワーク階層の情報も使うのか，といった問題とともに，パケット分類のアルゴリズムの検討も必要である．これは処理時間のコストとともにハードウェア資源コストとのバランスを視野に入れて決定する必要があるからである．

本研究の提案に加えて，上の課題が解決することによって，ノード単位はもちろん，ネットワーク全体，さらには全地球規模のネットワークにおいて，本研究の最終目標である，将来の帯域幅の拡大に十分対応できるように高速で，さらに，今後ますます多様化するであろうサービスが要求する品質に対応できる柔軟にクラスベースの QoS 制御の行えるルータを実現するネットワークプロセッサが開発できると思われる．

謝辞

指導教員の寺田 浩詔高知工科大学副学長には大変お忙しいにもかかわらず折りをみては暖かい声をかけていただき，また的確なご助言もいただき大変感謝いたします．

情報システム工学科長 島村 和典教授には副査として有益なご助言をしていただき深く感謝します．

岩田 誠教授には大学院に入学してから一貫して懇切丁寧に，また時には厳しくご指導いただき本論文をまとめることができました．心より感謝いたします．

情報システム工学科 大森 洋一助手には DDMP プログラムについての重要な示唆をいただきました．ここに感謝の意を表します．

情報システム工学科の諸先生方には，授業および研究で熱心なご指導，ご助言をいただきました，ここに感謝の意を表します．

大学院博士課程 林 秀樹氏にはディスカッションなどを通じて最新の価値ある情報を提供していただくとともに研究についてご助言頂きました．ご協力に感謝いたします．大学院修士課程岩田研究室の橋本 正和氏，別役 宣奉氏および森川 大智氏には DDMP および Java プログラミングについて貴重なご意見を頂きました．感謝いたします．大学院同期の清水研究室 田鍋 潤一郎氏および岡田 実氏とは，うれしいときには共に喜び，苦しいときには共に支えあい，この 2 年間楽しく過ごすことができました．本当にありがとうございました．

岩田研究室ネットワークチームの志摩 浩氏には Java 言語で標準偏差メソッドを提供していただき，たいへん助かりました．感謝いたします．最後になりましたが，岩田研究室学部生の皆さん，関係者ご一同に心からお礼を申し上げます．

参考文献

[1] H.Terada, "Impact of Photonic Technology on the Future Communication," *IEICE Trans. Comm.,* vol. E77-B, No. 2, pp. 96-99, February 1994.

[2] L. Gwannap and B. Wheeler, "A Guide to Network Processors," *MicroDesign Resources,* 2000.

[3] H. Terada, S. Miyata and M.Iwata, "DDMP's: Self-Timed Super-Pipelined Data-Driven Multimedia Processors," *Proc. of the IEEE,* Vol. 87, No. 2, pp. 282-296, February 1999.

[4] 林 秀樹, 岩田 誠, 孔 令杰, 米田 進, 寺田 浩詔, "超広帯域ネットワークにおける分散型マルチプロトコルルータ" 情報研報 *2001-DSM-22,* pp. 1-6, July 2001.

[5] D. Morikawa, H. Hayashi, M. Iwata and H. Terada, "Superpipelined IP-Address Lookups on a Data-Driven Network Processor," *in Proc. of Int. Conf. on Parallel and Distributed Computers and Systems,* pp.431-436, August 2001.

[6] S. Shenker and J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements," *RFC 2215,* September 1997.

[7] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP)," *RFC 2205,* September 1997.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services," *RFC 2475,* December 1998.

[9] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," *RFC 2474,* December 1998.

[10] Y. Bernet, S. Blake, D. Grossman and A. Smith, "An Informal Management Model for Diffserv Routers," *http://www.ietf.org/internet-drafts/draft-ietf-diffserv-model-*

06.txt, February 2001.

[11] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker," *RFC 2697,* September 1999.

[12] http://www.uu.net/us/support/sla/

[13] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks" *IEEE/ACM Trans. on Netw.,* Vol. 3 No. 4, pp. 365-386, August 1995.

[14] J. Nagle, "On Packet Switches with Infinite Storage," *RFC 970,* December 1985.

[15] A. K. Parekh and R. G. Gallagar, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Network: The Single Node Case," *IEEE/ACM Trans. Netw.,* vol.1, no.3, pp.344-357, June 1993.

[16] A. Demers, S. Keshav and S.Shenker, "Analysis and Simulations of a Fair Queuing Algorithm," *In Proc. of ACM SIGCOMM 1989 Symposium,* pp. 1-12, September 1989.

[17] J. Postel, "Transmission Control Protocol, DARPA Internet Program Protocol specification," *RFC 793,* September 1981.

[18] 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.

[19] V. Jacobson, K. Nichols and K. Poduri, "An Expedited Forwarding PHB," *RFC 2598,* June 1999.

[20] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski, "Assured Forwarding PHB Group," *RFC 2597,* June 1999.

[21] IEEE Project 802. IEEE P802.1p: "Supplement to MAC Bridges: Trac Class Expediting and Dynamic Multicast Filtering," *IEEE, Incorp. in IEEE Standard 802.1D, Part 3: Media Access Control (MAC) Bridges: Revision,* 1998.

[22] IEEE Standard for Local and Metropolitan Ares Networks: "Virtual Bridged Local

Area Networks," 1998.

[23] 細美 俊彦, 岩田 誠, 林 秀樹, 寺田 浩詔 "自己同期パイプラインによる Diffserv 制御機構", 電気関係学会四国支部連合大会, pp. 199, September 2001.

[24] T.Yamasaki, K. Shima, S. Komori, H. Takata, T. Tamura, F. Asai, T. Ohno, O. Tomisawa and H. Terada, "VLSI Implementation of a Variable-Length Pipeline Scheme for Data-Driven Processors," *IEEE Journal of Solid-State Circuits,* vol. 24, no. 4, pp. 933-937, August 1987.

[25] Sharp 株式会社, "DDMP-AUDIO DADT Library 仕様書第 1.00 版", May 2001（非公開）.

[26] H. Hayashi, M. Iwata, T. Hosomi and H. Terada, "A Self-Timed Pipeline Implementation of Class-Based QoS Control," *8th International Conference on High Performance Computing (HiPC2001) Workshop on Embedded Systems,* pp.6-10, December 2001.

[27] 細美 俊彦, 岩田 誠, 林 秀樹, 寺田 浩詔, "自己同期パイプラインによる SLA 制御の実現法", 電子情報通信学会全国大会, March 2002（発表予定）.

[28] http://www.caida.org/analysis/AIX/plen_hist/

[29] P. Gupta and N. McKeown, "Algorithms for Packet Classification," *IEEE Network Special Issue,* March/April 2001, vol. 15, no. 2, pp. 24-32.