QoS 技術〜 Intserv と diffserv

長 健二朗（ソニーコンピュータサイエンス研究所）

1998 年 12 月 16 日

Internet Week 98　国立京都国際会館

（社）日本ネットワークインフォメーションセンター編

目次

1 概要

このチュートリアルでは、以下の 3 つの内容について、それぞれの全体像を掴んでいただくことを目標として、まんべんなく解説していきます。

- トラフィック管理と QoS ～概要とバックグラウンドの要素技術
- Intserv と RSVP ～資源予約をして QoS を保証するタイプのサービスについて
- Diff-serv ～最近話題のもの

2 トラフィック管理と QoS

インターネットにおけるトラフィック管理の目的は、資源を効率的に配分して、ユーザに多様なサービスを提供することです。トラフィック管理とは、さまざまな要素技術を組み合わせて、ポリシーを実現することです。インターネットにおけるトラフィック管理では、ユーザの要求が多様で、かつ変化が激しいのが特徴です。

ISP の立場から、トラフィック管理の経済原理を考えると、次の 3 点が目標となります。

- 多様なユーザの要求を満たすために、豊富なサービス品目を用意する
- 音声とデータなどで、サービスインフラを統合する
- ニーズを集約することで、単価を下げる ～ 反面、対故障性の問題が発生

では、トラフィック管理を理解する上で重要な要素理論と技術を、まず解説していきましょう。

2.1 トラフィックモデル

トラフィックモデルの代表例である電話網では、発呼到着と通話時間の両方がポアソン分布で近似できることが分かっています。また、キューイング（待ち行列）理論もモデルにうまくマッチしていました。しかし、最近では、FAX やデータ通信の登場によって、旧来のモデルが使用できなくなりつつあります。

これに対して、インターネットのトラフィックモデルは、LAN や WAN といったネットワーク毎に特徴が異なることや、質的な変化が極めて急速であることが特徴です。したがって、様々なモデルが考えられます。たとえば、LAN と WAN とアクセス網のそれぞれにモデルが考えられますし、単一のフローと複合フローによるモデル、実測モデルと合成（数学）モデルなどです。

特に、ネットワークのトラフィックに特徴的なのは、「トラフィックの自己相似性」です。これは、トラフィックがどの時間スケールで見ても、バースト的であること（フラクタルの様であること）を意味します。すなわち、集積しても滑らかに集約しません。

また、自己相似性と同時に、「Heavy Tail」と呼ばれる特徴、すなわち分散値が無限大の値を持つことも指摘されています。ファイルサイズやWWWページのサイズのように、いくらでも大きなものがいつまでも出現しうるようなものが「Heavy Tail」です。これらの特徴を示す原因はまだ明らかになっていませんが、ネットワークのトラフィックがこのような特徴を示すことは、周知の事実として認められつつあります。

2.2 キューイング理論

キューイング理論は、ARPANET 時代からデータ通信と共に発展した理論体系で、システムのスループットを計算するためなどに利用されます。ある確率関数で表されるレート（ポアソン分布）で入力が発生し、そのそれぞれに同様に、ある確率関数で表されるレート（指数分布）のサービス時間がかかる場合のスループット計算が最も基本的なものです。

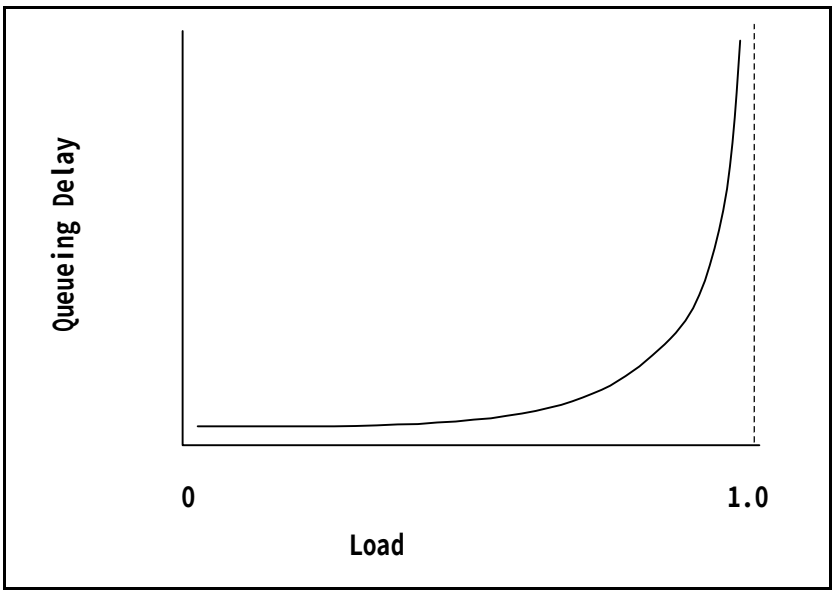
キューイング理論における重要な法則として、「リトルの法則」を挙げることができます。これは、

$$N = \lambda T$$

N：システム内の平均プロセス数
λ: 平均到着レート
T: 平均サービス時間

で表されるもので、平均プロセス数が、平均到着レートと平均サービス時間の積に等しいことを示しています。

キューイングシステムの挙動を観測すると、負荷があるポイントを超えると、待ち時間が急激に上昇することが分かります。



道路の渋滞などもこの理論で説明することができます。キューイングシステムにおいては、負荷の 6 〜 7 割程度までは、到着するリクエストのゆらぎによって負荷が吸収されますが、それを超えると、極端に悪化するのです。

このキューイング理論は、システムの性能計算や、ボトルネックの発見に使われています。ネットワークシステムのコンポーネントを、それぞれキューに置き換えてブラックボックス化して、全体の効率を探ることになります。たとえば、WWW サーバの例ですと、ディスクアクセスやリクエストを受け付ける部分などを、それぞれキューイングシステムと見立てて全体の性能を予測します。

キューイング理論は、統計的な性質を利用したものですから、マクロ的な解析にしか利用することができません。また、キャパシティを超える入力がある場合（例外状態）の解析はできませんし、入力にフィードバックや相関がある場合は解析が非常に困難です。

2.3 トラフィック管理

ネットワークのトラフィック制御は､さまざまなレベルで行うことが考えられます。時間粒度の順で、考えられることを挙げてみましょう。

- Long Term
  容量設計をどうするか
- Day
  ピーク時刻に基づく料金体系
- Session
  サービス別の料金、アドミッション制御 (コネクションの制限)、ルーティング
- Round Trip Time
  エンドーエンドのフロー制御
- Packet Time
  キューイング方式（パケットの優先制御）
- それ以下
  データリンク層に依存

2.4 QoS

インターネットは、Best Effort サービスを基本としています。ネットワークは、全てのパケットを公平に扱い、それぞれについて最善を尽くしますが、何の保証もしない、簡潔で自由な精神に基づくものです。しかし、リアルタイムなどのユーザの要求には、Best Effort サービスでは応えられなくなりつつあるのが現状です。

そこで、Best Effort と完全保証の間にあるさまざまなサービス品質を提供する「QoS（Quarity of Services）」が登場します。QoS の目的は、次のようにまとめることができるでしょう。

- 期待されるアプリケーション性能を満足する
  リアルタイム性を要求するアプリケーション（音声や画像）が登場してきています。
- ネットワーク資源の分配を制御する
  多様なアプリケーションと多様なサービスの出現によって、トラフィック変動の幅が広く、速度が速くなってきています。また、ネットワークの巨大化によって、従来のエンドーエンドによるフィードバック制御が機能しなくなりつつあります。
- ISP による幅広いサービスを実現する
  品質重視のプレミアムサービスと、コスト重視のディスカウントサービスの両方が、ユーザ、ならびにアプリケーションから求められています。

ここで、QoS を定義しておきましょう。IETF の Int-serv ワーキングループによれば、狭義の QoS（Quarity of Service）とは、「定量的に表すことができる絶対的性能」を指します。たとえば、帯域が○ Mbps、遅延が○ ms といったものです。QoS に対して、「Class of Service（CoS）」という言葉も使われます。これは、クラス間の相対的な性能差を表すもので、Diff-serv などで使われます。

QoS を具体的に述べると、アプリケーションから見えるネットワーク性能であると言えるでしょう。たとえば、帯域や遅延、ジッタ（パケットのゆらぎ）、損失率などです。QoS を制御するためには、パケットスケジューリングや、バッファ、CPU パワーなど、ネットワーク上の資源配分をコントロールします。

2.5 QoS の実現

QoS を実現するための要素がいくつかありますが、中でもキューイングによる遅延が最も大きな影響を及ぼします。ネットワークインタフェースの出口、特に LAN から WAN への接続点や、複数の流入口から特定の出口に向かう点では、キューの中でパケットが長時間を過ごすことになります。パケットがキューの中で過ごす時間をコントロールすることができれば、有効な QoS 制御ができることになります。

「Over-provisioning（過剰投資）を行って回線速度を十分に持てば、遅延を最小にすることができるだろう」という、QoS とは対極的な考え方もあります。小規模なネットワークでは、この方法もある程度有効なのですが、トラフィックの集中は予測できませんし、トラフィックは自己相似的ですから、広域なネットワークではコスト的に割があいません。

QoS と Over-provisioning は、二者択一の方法ではなく、コストと運用を考慮したバランスを取ることが重要でしょう。効果的な QoS を実現するには、ある程度の余裕、すなわち Over-provisioning が必要です。

2.6 QoS の技術要素

ここから、QoS の要素技術について紹介していきましょう。

- アドミション制御（admintion contorol）
  RSVP のように、セッション毎に資源の予約をコントロールする方法です。セッションの開始時に、セットアッププロトコルによってパス上の資源を確保し（signaling)、確保に失敗するとセッションが開始されません。資源を解放する方法（特に故障時）が重要です。優先的に資源を割り当てるポリシー、他のパスを上手に使うルーティング、資源に対する課金などを考慮する必要があります。

- クラシファイア（classifier）
  ルータの中にあって、到着したパケットをグループに分類する方法です。IP では、ソースアドレス、ディスティネーションアドレス、ソースポート、ディスティネーションポート、プロトコルを使ってパケットをフィルタリングして、特定ホスト向けの特定サービスを抜き出します。また、TOS（Type of Service）を使用する方法もあります。実装上は、フィルタにおけるワイルドカードの扱いが問題となります。

- パケット・スケジューラ (packet scheduler)
  グループに分類されたパケットを送出する、スケジューリングを調整する方法です。キューイング方法とバッファ管理方法によって、さまざまな方法があります。

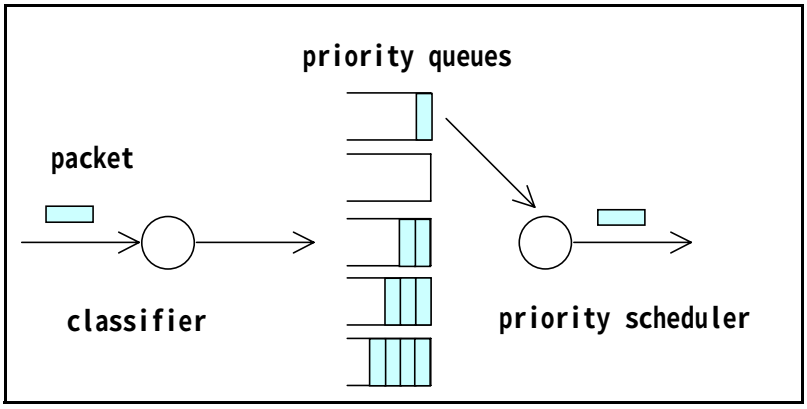
- シェーパ (shaper)
  バーストを一定レートにならす技術です。

2.7 キューイング方法

代表的なキューイング方法をいくつか紹介しましょう。

2.7.1 Priority Queueing

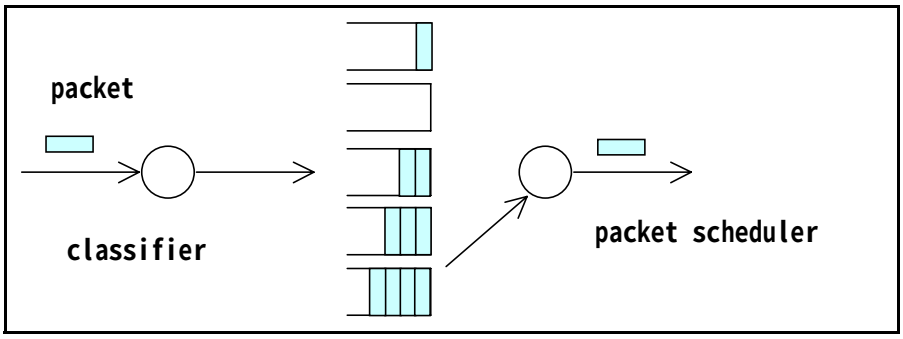
優先度（priority）毎にキューを作り、優先度が高いものから処理していくスケジューリング方法です。



単純な機構でリアルタイム性を保証することができますが、優先度が低いキューは、いつまでも処理されない恐れがあります。

2.7.2 Weighted Fair Queueing（WFQ）

フロー毎に独立したキューを作り、各キューから順にパケットを処理していくスケジューリング方法です。



WFQ では、他のフローからの影響を一定以下に抑えることができます。ただし、フローの数だけのキューが必要になり、現実的ではないので、ハッシュによってキューを選択するなどの近似法が使われます。

トークンバケットと WFQ を組み合わせて使用することで、最大遅延時間を計算できることを、Parekh が証明しました。トークンバケットについては後述しますが、シェーピングに類似した方法によって、発生するトラフィックのバースト遅延を一定値以内に納める方法です。WFQ では他のフローの影響による遅延を一定値以内に納めることが可能ですから、その総和を求めることができます。

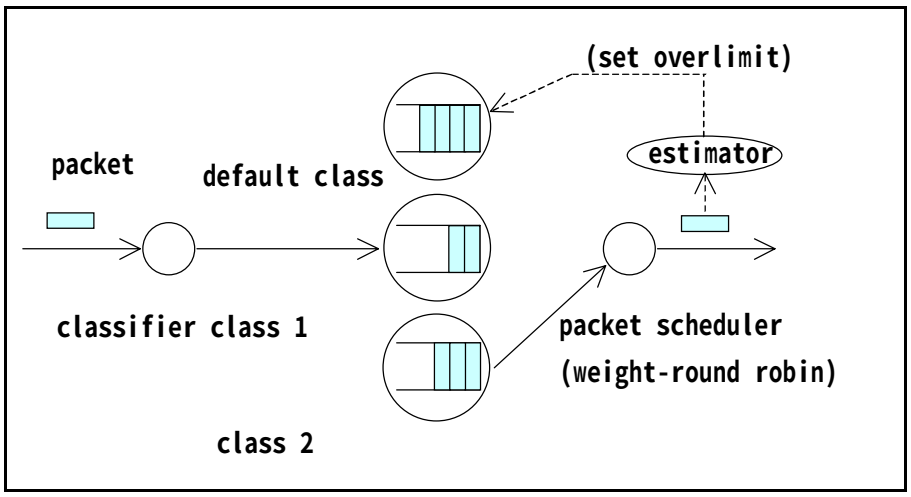
すなわち、

最大遅延時間＝バースト遅延＋自フローのゆらぎによる遅延＋
他のフローによる遅延の総和

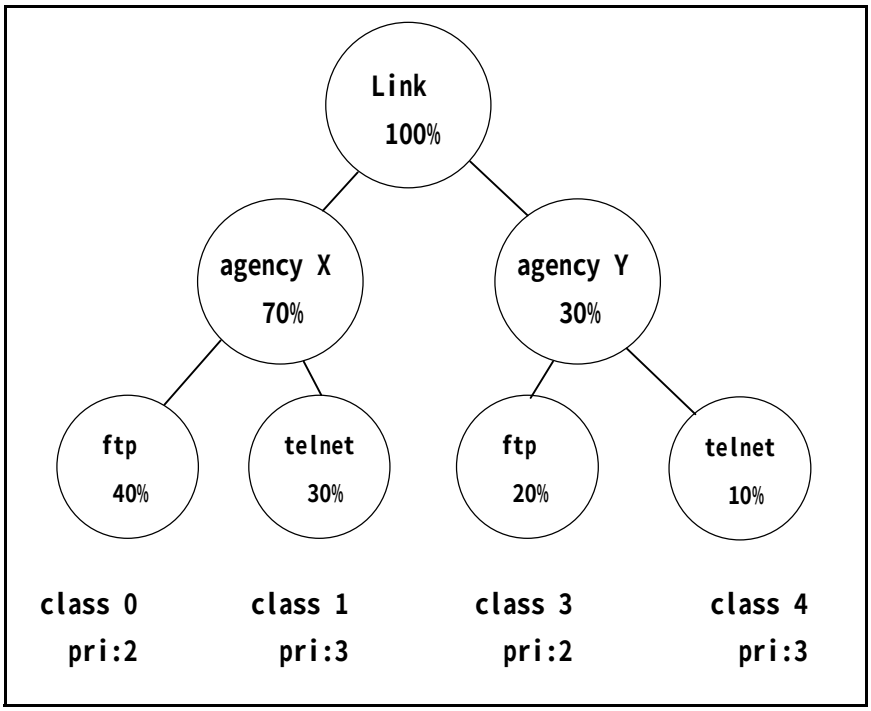
によって、最大遅延が求められます。この Parekh モデルから、トークンバケットと WFQ を組み合わせると、最大遅延を一定値内に納める QoS が実現できることから、さまざまな製品にこのモデルが応用されています。

2.7.3 Class-Based Queueing（CBQ）

クラス毎に独立したキューを作り、それぞれのキューからラウンドロビンでパケットを送出していくスケジューリング方法です。パケットスケジューラには estimator（評価者）が組み込まれており、それぞれのクラスが使用している帯域幅を監視しています。クラスのトラフィックが、定められた値を超えた場合には、スケジューラがそのクラスのキューからの送出をスキップして、クラスのトラフィックを抑えます。一種のシェーピングが行われると考えればよいでしょう。



CBQ においては、階層的に集約されたフローのそれぞれがクラスであり、全体が混み合ってきたときにのみ、定められた割合で帯域が分割されます。



2.8 バッファ管理

バッファが満杯になり、いずれかのパケットを捨てる必要がある場合に、どのパケットを捨てるかを決定する代表的なアルゴリズムをいくつか紹介します。

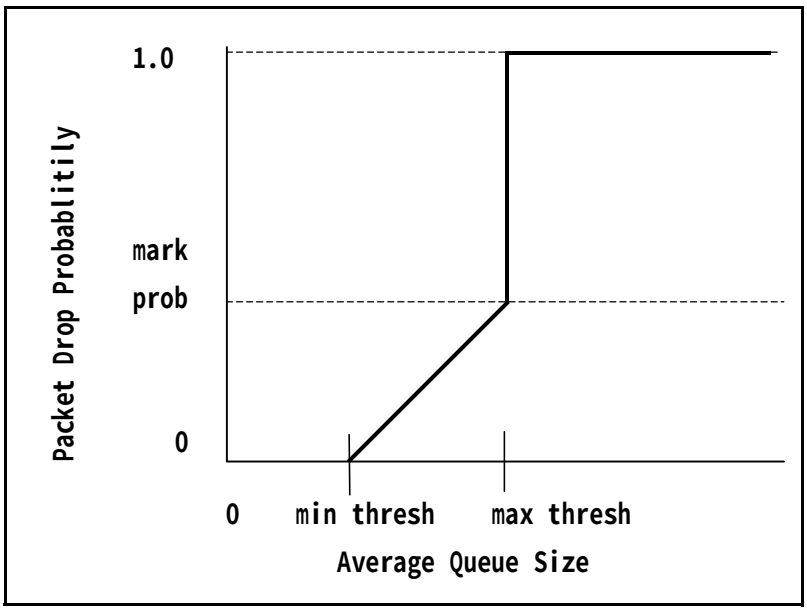
- Drop-Tail
  バッファの末尾を捨てるアルゴリズムです。

- Drop-Head
  バッファの先頭を捨てるアルゴリズムです。

- Drop-Random
  ランダムに捨てるアルゴリズムです。

- Random Early Detection（RED）
  平均のキューサイズに応じて、確率的にパケットを破棄するアルゴリズムです。RED では平均のキューサイズを使ってパケットの破棄を決定することで、短時間のバーストを許容します。確率的にパケットを破棄しますので、バッファを多く占有しているものほど、破棄される確率が高くなります。平均のキュー長の使用によって混雑を早期に検出できるため、TCP のキューが溢れるをことを防ぎ、良好な反応を得ることができます。ただし、パケットが破棄された場合に流量をコントロールしないトランスポート層に対しては、役に立ちません。

RED によってパケットが破棄される確率がどのように変化するかを示します。キューの最大長が 50 ほどの場合には、Min Threshold に 5、Max Threshold に 15、mark probablility に 10% 程度を使用するのが一般的になっています。



3 Intserv と RSVP

インターネットにおける QoS の取り組みは、ストリーム指向のアプリケーションの登場を受けて、89 年頃から研究が開始されていました。94 年からは、IETF の 3 つのワーキンググループで標準化が進められています。

- Int-serv
  サービスの分類や、QoS パラメータを定義しています。
- RSVP
  セットアッププロトコルのリファレンスモデルを定義しています。実装は、南カリフォルニア大学によって行われています。
- Integrated Services over Specific Link Layers（ISSLL）
  特定のデータリンク層に依存する仕様を定義しています。

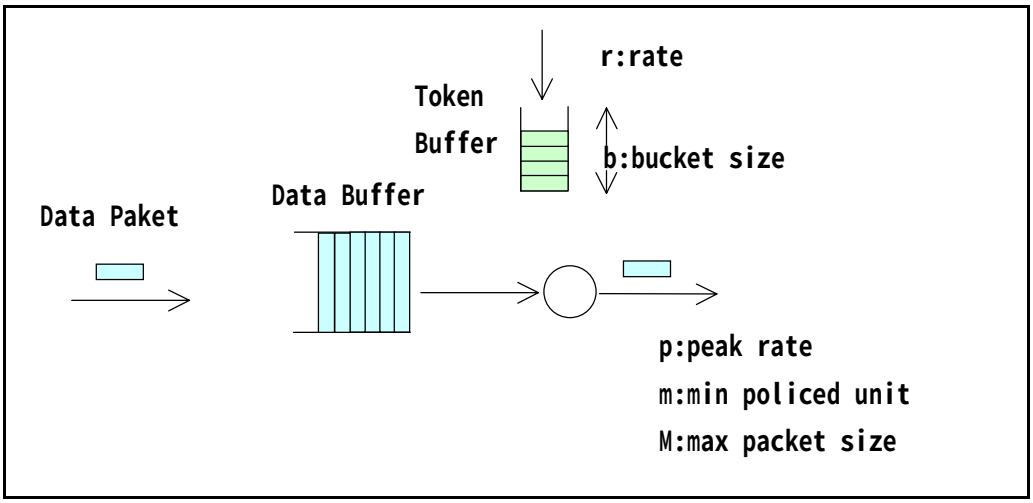
実装に依存するトラフィック制御機構や、ポリシ制御機構については、標準化が進められていません。

3.1 Int-serv

Int-serv では、帯域と最大遅延を保証する「Guaranted Service」と、遅延の目標値を持ち、流量を自ら制御する適応型アプリケーションを想定した「Controlled Load Services」の 2 つのモデルを想定しています。後者では、ネットワークが混雑している場合であっても、利用率が低い Best Effort ネットワークと同等程度に動作することを目指しています。実際の実装は、Controlled Load Services を実現しようとするものです。

リアルタイムアプリケーションにおいて、再生を始める前にバッファに溜め込む流量を決定するためには、最大遅延を保証することが必要です。そのためには、他のトラフィックからの影響を受けずに、自分自身のトラフィックを一定化するためのフロー制御とシェーピングを行わなければなりません。

シェーピングを実現するために、Token Backet TSpec という方法がとられます。これは、一定レートで入ってくるトークンがトークンバケットにある間は、キューにあるパケットの送信を許すものです。



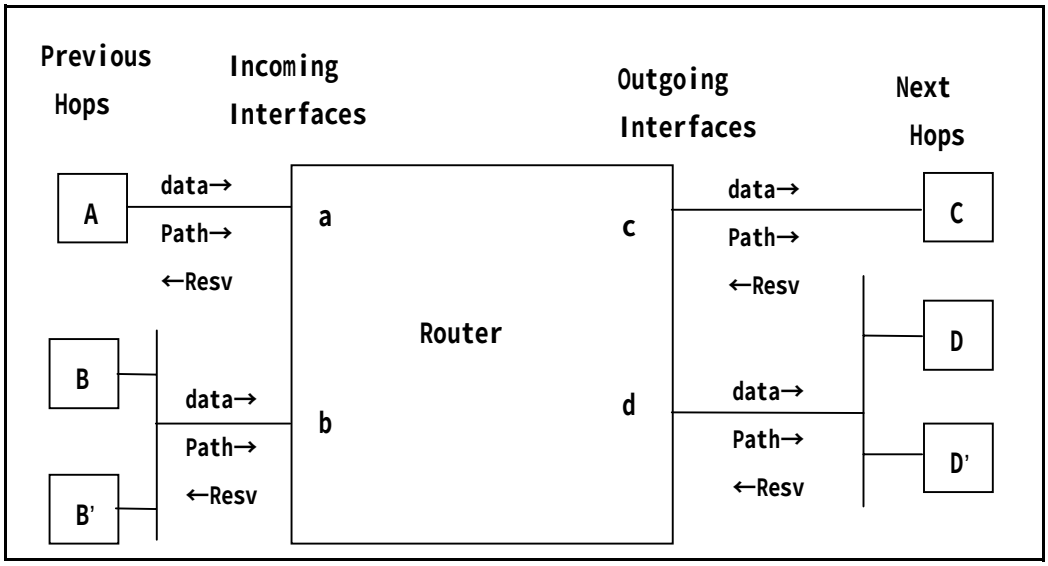
これによって、小さいバーストを許しながら、流量を一定化することができます。

3.2 RSVP

まず、RSVP の特徴を列挙します。

- マルチキャストとユニキャストの両方で使える資源予約プロトコルである。
- IPv4 と IPv6 の両方で使える。
- 一方向のデータフローに対する資源予約を行う。
- レシーバが予約を起動する。
- 経路やマルチキャストのメンバ変更に、動的に適応する（ソフトステート）
- ルーティングプロトコルから独立している。
- 予約要求は、トラフィック制御モジュールやポリシ制御モジュールで判断される。
- ソースとディスティネーションが一定（Fixed Filter）、複数のペアのリスト（Shared-Explicit）、受信者が一定（Wildcard-Filter）という、3 種の予約スタイルが定義される。
- RSVP をサポートしないネットワークを透過して動作する。

RSVP のモデルを示します。上流である A/B/B’ から、下流である C/D/D’ にデータが流れる場合に送られる、メッセージの種別を示しています。



次に、RSVP で使われている主要なメッセージタイプを説明します。

- Path
  送り手がフロー情報をアナウンスするメッセージです。送り手は定期的にこのメッセージを送り、フローを指定するための情報と、オリジナルのフロー情報を通知します。中間のルータは、フロー情報を書き換えながら、このメッセージを中継します。

- Resv
  Path に応答して、受信者が予約を確立するためのメッセージです。受信者は、到着した Path メッセージに応答して、定期的にこのメッセージを送り、予約したい QoS 指定情報と、フローの指定を通知します。中間のルータは、このメッセージに従って予約を確保して、上流に中継します。この時に、複数の受信者からの予約をマージすることがあります。

- PathTear、ResvTear
  予約の終了を行うためのメッセージです。

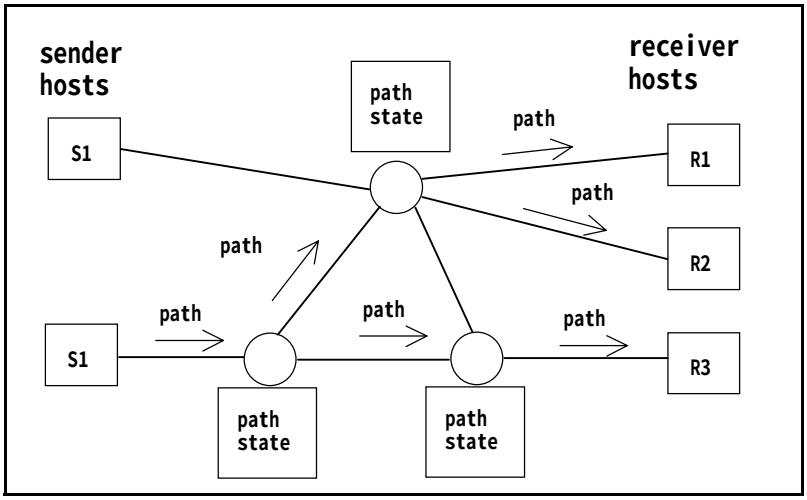
- PathErr、ResvErr
  予約の失敗を通知するためのメッセージです。

- Confirm
  予約を確認するためのメッセージです。

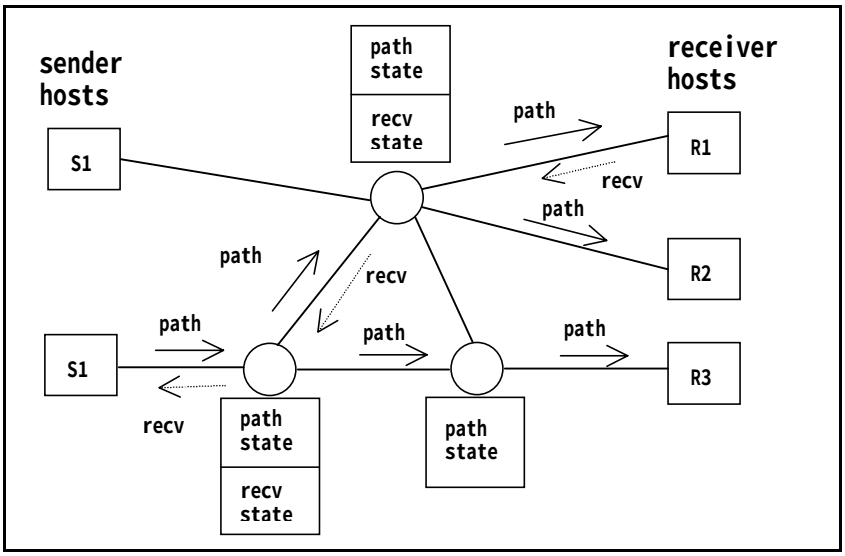
- DiagReq
  診断を要求するメッセージです。

- DiagRep
  診断に応答するメッセージです。

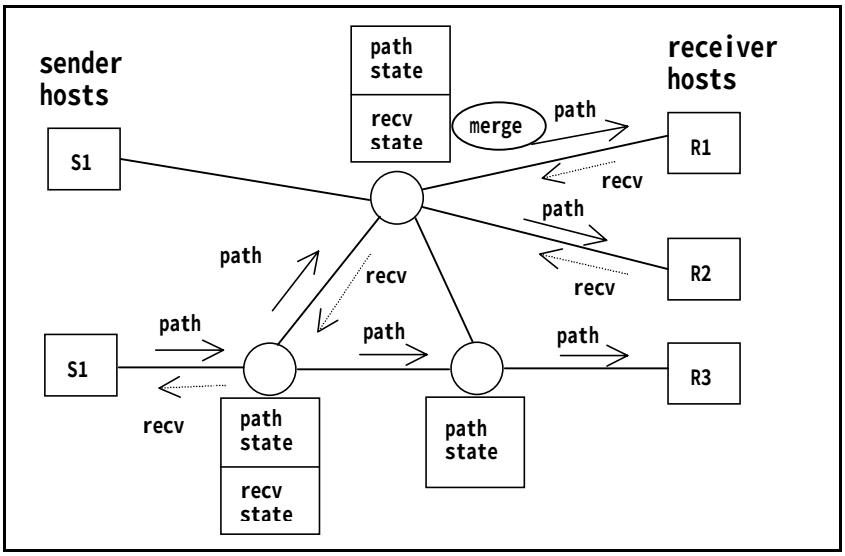
Path メッセージと Resv メッセージの例を示します。S2 からマルチキャストによって Path メッセージが送られ、途中のルータには Path State が生成されます。



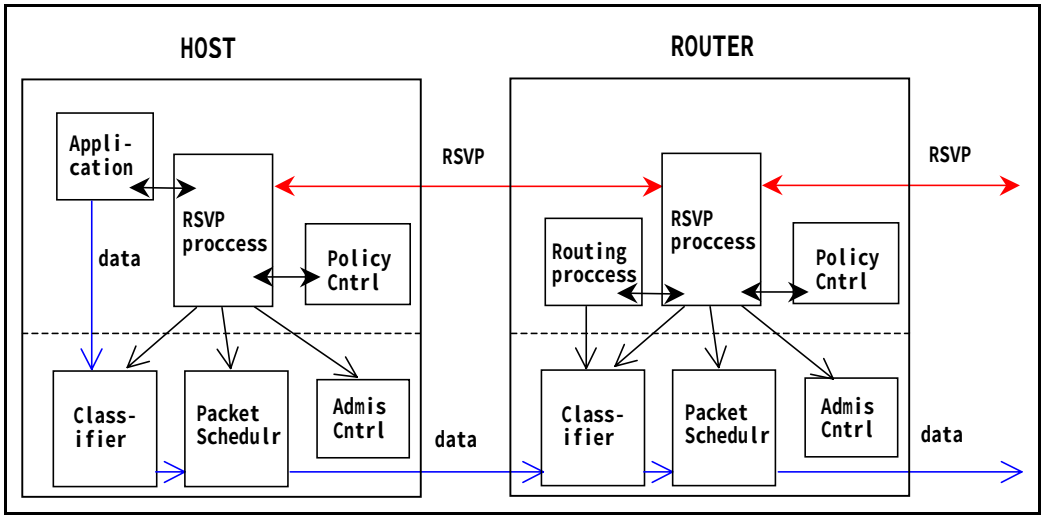
R1 から Resv メッセージが返され、途中のルータには Resv State が生成されて、予約が完了します。



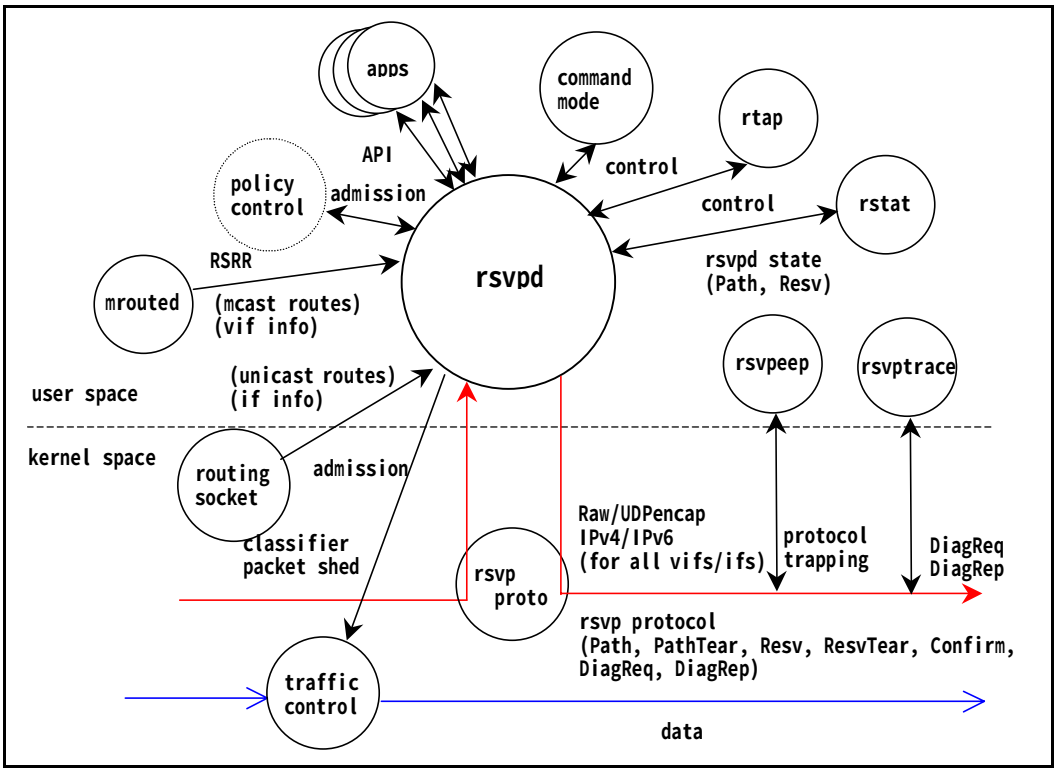
さらに、R2 から Resv メッセージが返されると、予約がマージされて、Resv State が更新されます。必要がなければ、Resv State は上流に伝達されません。



RSVP の実装モデルを示します。ホストとルータの 2 種類の実装モデルにて、どちらも RSVP プロセスが中心的な働きをすることと、やりとりを行うモジュールが示されています。予約のしくみと、データ自体が別の系統で伝達されることに注意してください。



ISI RSVPD の実装をモデル化したものを示します。カーネル内で RSVP プロトコルを検出すると、それを横取りして rsvpd に渡すしくみとなっています。rsvpd はさまざまなデーモンやカーネルモジュールとやりとりを行い、カーネル内に実装されたトラフィックコントロールモジュールを制御します。svpeep と rsvptrace は、デバッグ用のモジュールです。



3.3 Integrrated Services over Specific Link（ISSLL）

Int-serv を対象にした、リンク層のしくみを規定しています。次の仕様があります。

- IS802
  Ethernet 系を対象に、サブネットにおけるバンド幅をコントロールするしくみを規定しています。
- ISATM
  Int-serv パラメータと ATM パラメータのマップを規定しています。
- ISSLOW
  低速回線（ダイヤルアップから T1 程度）を対象に、ヘッダ圧縮やフレーミングを工夫して、パケット遅延が大きな場合のしくみを規定しています。

3.4 まとめ

RSVP による標準化はほとんど完了して一段落している状態です。しかし、RSVP の課題として、次の点が挙げられます。

- ポリシー制御～ほとんど未着手
- トンネリング～ほとんど未着手
- ステートの集約～ State が大きくなってスケールしない
- 2 階層資源管理～広域の資源管理では階層化したプロトコルが必要だろう
- RSVP over MPLS ～ネットワークレイヤとその下のレイヤの統合
- RSVP over diff-serv ～スケールを目指す

また、研究主導で製品開発がついてきておらず、十分に普及していないことや、メディアの過剰な期待への反動などから、RSVP は既に不要であるとの意見が聞かれることもあります。しかし、広域ネットワークではスケーラビリティが問題になるものの、イントラネット規模ではまずまず実用になるものと考えられます。

4 Diff-serv

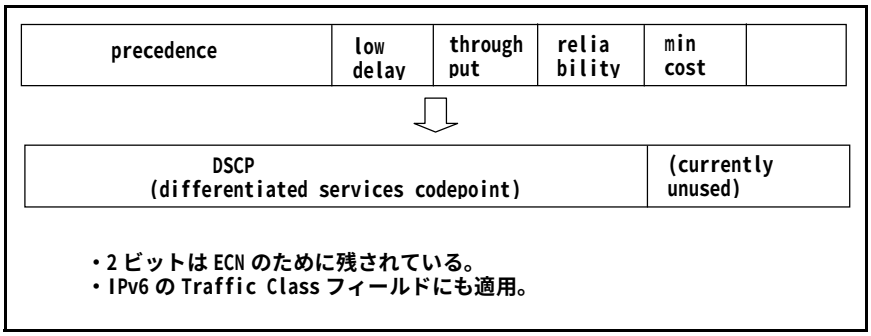
Diff-serv の仕様は、ここ 2 年間くらいに急速に定められてきており、現在も作業が進められています。Int-serv/RSVP の複雑さや、スケーラビリティに対する疑問、ISP からのプレミアサービスを行いたいという要求、ルータベンダーの思惑などが絡まりあって、次のような特徴を持った Diff-serv が登場してきたのです。

- スケールする～フロー毎の State を持たない
- シンプルであること～ TOS フィールドを再定義して Classifier を簡単に行える
- イントラネット／インターネットドメインのサポート～ ISP のビジネスモデルにマッチする

Diff-serv は 97 年 8 月にミュンヘンで開催された IETF にて、Int-serv ワーキンググループが、Diff-serv BOF を開催したことから始まります。そこでの話題は、Premium Service Model（V. Jacobson@LBL）、Drop preference Model（D. Clark@MIT）、Cisco’s CoS（F. Baker@Cisco）の 3 つでした。98 年 3 月には IETF に Diff-serv ワーキンググループが設立され、大学や政府、ベンダ、ISP の大物達が協力しあって、急速に標準化が進んでいます。

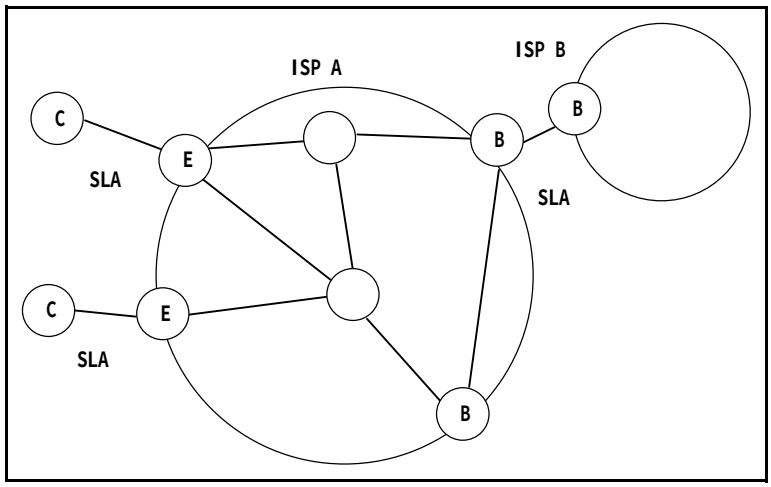
4.1 TOS フィールド

IP ヘッダには Type of service（TOS）と呼ばれる 8 ビットのフィールドがあります。このフィールドの定義を示しますが、それぞれの意味があいまいであることから、これを 6 ビットの Diffeerentiated services codepoint と、2 ビットの ECN のために予約されたビットに定義し直します。この TOS フィールドを使って、Diff-serv による QoS が実現されます。IPv6 における Traffic Class フィールドも同様のしくみです。このため、IPv4 と v6 の両方で、Diff-serv を実現することができます。

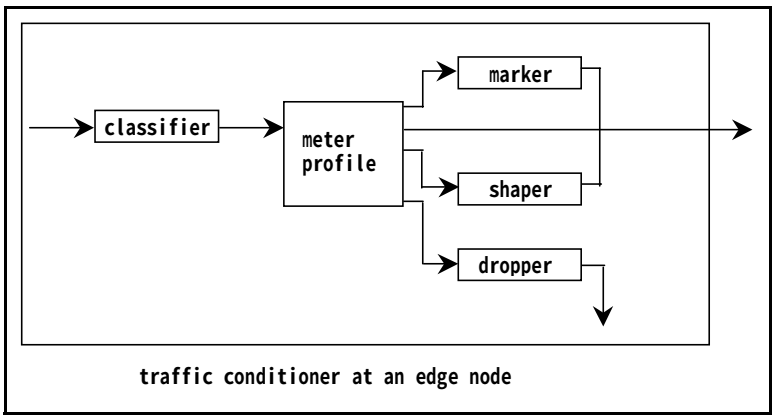


4.2 Diff-serv のモデル

Diff-serv は、ネットワークの入り口でトラフィックをコントロールするしくみを実現します。ネットワークの境界にあるエッジノードで、顧客と ISP の間の Service Level Agreement に基づいて、TOS フィールドのコードポイントを設定します。中間ノードでは、コードポイントに応じてパケットスケジューリングを行います。ネットワークをまたがるバウンダリノードでは、取り決めに従ってコードポイントを書き換える作業を行います。



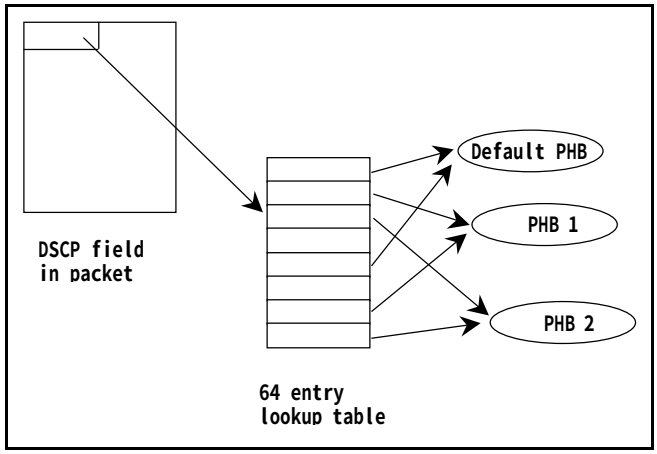
エッジノードでは、アドレスやポート番号から、classifier によってサービスの種別を識別します。meter profile が、それとユーザ毎のプロファイル（契約内容）を比較し、その結果に応じてコードポイントの設定やシェーピングの処理が行われます。



4.3 Forwarding Behavior

中間ノードでは、パケットに含まれるビット列、すなわちコードポイントを参照して、パケット中継のふるまい（Forwarding Behavior）を変更します。それぞれの ISP がコードポイントを定義して、それぞれのコードポイントに応じた動作をルータで設定・実現します。ルータがコードポイント毎にどのような動作をするかを、ネットワーク毎に独立して決めることができるので、ISP 独自のサービスを提供することができるわけです。ただし、グローバルに使える標準的な方法がいくつか定義される予定となっています。

Forwarding Behavoir の基本として、Per Hop Behavior（PHB）が挙げられます。すなわち、それぞれのルータがパケットを中継するしくみのことです。ルータにおけるパケットフォワードの動作は、ルーティングテーブルをどのように作るかというポリシーと、そのルーティングを使ってどのようにパケットを送り出すかというメカニズムに分類することができます。Diff-serv でも同様に、PHB は、何かのテーブルを作り出すポリシーと、テーブルを参照して動作するメカニズムに分類できます。具体的には、コードポイントを参照して、利用する PHB を決定するメカニズムが使われます。



コードポイントの 6 ビット空間は、次のように分割定義されています。

- xxxxx0：Standard PHBs
  000000：デフォルト PHB
  xxx000：Class Selector PHBs （7 個：IP Precedence 互換）
  +++++1：Assured Forwarding PHBs（12 個：内 4 つは Class Selector と重複）
  101110：Expedited Forwarding PHB
- xxxx11：Experimental/Local Use
- xxxx01: Experimental/Local Use （予約）

標準的な PHB として、次の 3 つが挙げられています。

- Default PHB
  コードポイント 0 のものは、Best Effort として扱われます。
- Class Selecor PHBw
  Cisco が実装している IP precedence を使う方法と互換性のある方法です。
- Assured Forwarding PHBs
  Clark の Drop Preference モデルに基づくもので、適応型アプリケーションに適します。相対的な QoS を持つ PHB グループを定義して、最低帯域保証のある Best Effort を実現します。エッジにて契約分を超えたトラフィックにマークを付け、中間ノードが混雑した場合にはマークがついたパケットから破棄します。

  4 つのクラスと 3 つのドロップレベルを定義したものが、IETF の RFC となっています。

- Expandited Forwarding PHBs
  Jacobson の Premium サービスを基にした仮想専用線モデルで、エンドーエンドの契約帯域を保証します。エッジにて契約分のみのトラフィックを通し、中間部では契約分の総和を上回る帯域を確保します。中間ノードでは、Priority Queue を用いてトラフィックをコントロールします。ATM の CBR サービスと同じです。

  このモデルを一般化して、エンドーエンド以外のスコープにも適用できるようにし、中間部において統計多重を使った容量計算を行うものが、IETF の RFC となっています。

4.4 まとめ

Diff-serv はシンプルで柔軟な枠組みを定義するもので、コードポイントが PHB を選択するという、最低限の標準化は完了しています。具体的なサービスは実装・運用依存であり、いくつかの標準モデルが推奨されていますが、ISP がどのようなサービスを作るかが問われると言えるでしょう。

Diff-serv の今後の課題を示します。メカニズムは標準化が進んでいますが、ポリシーの決定や交換方法の標準化がまだまだの状態と言えます。

- 受け手が優先度をコントロールするしくみが必要
  現在は、送り手がコードポイントを決定しています。エッジ間で優先度をやりとりするしくみを作ろうとする動きがありますが、まだまとまっていません。
- RSVP over Diff-serv
  Diff-serv と組み合わせることによって、RSVP のスケール問題を解消する試みが行われています。
- Diff-serv over MPLS
  Diff-serv は ATM などで使われているラベル付けと相性が良いことから、標準化が期待されています。
- 2 階層資源管理
  ISP 境界でサービスを交換するしくみはほとんど定義されていません。

5 終わりに

ユーザの要求が多様化して、多様なサービスを提供するために QoS の技術が登場してきました。

Int-serv/RSVP の登場によって、QoS 技術への理解と発展には役立ちましたが、現状はイントラネット向きのサービスであり、期待したほどの普及は進んでいません。この反省から、より現実的な Diff-serv が登場し、主要なメカニズムの標準化がまとまりつつありますが、運用面での課題が多く残されています。現在は、QoS 技術の大きな転換期であると言えるでしょう。

6 参考文献

- IETF
  http://www.ietf.org/
- IETF Diff-serv ワーキンググループ
  http://www.ietf.org/html.charters/diffserv-charter.html
- IETF Int-Serv ワーキンググループ
  http://www.ietf.org/html.charters/intserv-charter.html
- IETF ISSLL ワーキンググループ
  http://www.ietf.org/html.charters/issll-charter.html
- IETF RSVP ワーキンググループ
  http://www.ietf.org/html.charters/rsvp-charter.html
- RSVP
  http://www.isi.edu/rsvp/
- Diff-serv at MIT
  http://diffserv.lcs.mit.edu/
- CBQ
  http://www-nrg.ee.lbl.gov/floyd/cbq.html
- ALTQ
  http://www.csl.sony.co.jp/person/kjc/software.html
- 「Quality of Service」
  P. Ferguson and G. Huston. Wiley.
  ISBN 0-471-2358-2
- 「An Engineering Approach to COmputer Networking」
  S. Keshav. Addison-Wesley.
  ISBN 0-201-63442-2

- 「High-speed Networks: TCP/IP and ATM Design Principles」
  William Stallings. Prentice Hall.
  ISBN 0-13-904954-1
- 「Gigabit Networking」
  Craig Partridge. Addison-Wesley.
  ISBN 0-201-56333-9 （邦訳あり）
- 「Queueing Systems vol. 1 and vol. 2」
  LeonardKleinrock. Wiley-Interscience.
  ISBN 0-471-49110-1、0-471-49111-X