IP ルータにおけるポリシーサービス提供のための効率的なキャッシュ構成法

阿多 信吾　村田 正幸　宮原 秀夫

大阪大学 大学院基礎工学研究科 情報数理系専攻
〒 560–8531 大阪府豊中市待兼山町 1–3
Tel: 06–850–6588, Fax: 06–850–6589
E-mail: {ata, murata, miyahara}@ics.es.osaka–u.ac.jp

あらまし　さまざまなアプリケーショントラヒックが混在する現在のインターネットにおいて、トラヒックを適切に処理するためには、トラヒック特性や公平性を考慮した制御が不可欠である。そのような、ポリシーサービスを実現するためには、ルータ上でフロー識別をしながら、フロー特性に応じたスケジュールを実現する必要がある。しかしながら、フロー識別のためにはパケット内の複数のフィールドを参照する必要があり、処理の負荷が大きくなる。そこで本稿では、高速 IP ルータにおいてフロー識別を行なうために、一般的に入手可能な CPU およびメモリを用いた高速検索アルゴリズムを提案し、さらに CPU のキャッシュメモリを用いた高速化、特にキャッシュメモリにおけるパラメータが検索速度に与える影響について検証する。

キーワード　IP ルータ、キャッシュメモリ、ポリシーサービス、フロー情報、CPU サイクル

Study on Efficient Cache Structures Providing Policy–Based Services in IP Routers

Shingo ATA　Masayuki MURATA　Hideo MIYAHARA

Department of Infomatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 560–8531, Japan
Tel: +81–6–850–6588, Fax: +81–6–850–6589
E-mail: {ata, murata, miyahara}@ics.es.osaka–u.ac.jp

Abstract　In the Internet shared by the various applications, it is necessary to realize the policy–service based on the traffic characteristics. For this purpose, the router is required to forward the packets according to the requirements of the traffic flow that the packet belongs to. However, the packet (and flow) classification could easily become a bottleneck in the high–speed router because the router is required to search multiple fields within the packet. In this paper, we propose a new packet classification algorithm capable of following the packet forwarding rate of the high–speed routers with the commercially available CPU, RAM, and cache memories. Through our experiments, we show the effects of parameters of the cache memory on the processing rate of packet classification.

key words　IP router, cache memory, policy service, flow statistics, CPU cycles

1 はじめに

近年の急速なインターネットの普及と、動画像転送などのマルチメディアアプリケーションの増加によって、より高速なネットワークインフラの構築が要求されている。また一方では、混在するさまざまなアプリケーショントラヒックを適切に処理するためには、単にパケットを送出するだけでなく公平性、トラヒック特性を考慮した制御が不可欠である。さらに、あらかじめプロバイダ間で定められた SLA (Service Level Agreement) を実現するためには、トラヒック量に応じた経路選択も重要となる。このような背景から、IP ルータおいてもポリシーサービスをサポートする必要性が強まっている。

IP ルータがポリシーサービスを提供するためには、パケットに対するフロー識別が必須となるが、フロー識別のためにはパケット内の複数のフィールドを参照する必要があり、処理の負荷が大きい。特に、高速ネットワークに適応するためにはフロー識別を高速に行なう必要がある。これまでにも、ルータにおけるアドレス検索問題の高速化に関する研究が行なわれているが [1]、検索する情報量が急激に増大することから、フロー識別へ直接適用することは難しい。したがって、フロー識別に適した新たな検索アルゴリズムを考える必要がある。

文献 [2] では、高速ルータにおけるフロー識別のためのアルゴリズムが提案されているが、期待される性能を得るためには複数の検索を同時並行実行するためのメカニズムが必要となり、その結果フロー識別のためのハードウェアが必要になるため、コストが増大する。現在、パーソナルコンピュータの普及により、市場で入手可能な CPU の価格は下落傾向にあり、これらの CPU を利用することにより、低価格のフロー識別ルータを実現することが期待できる。本稿では、高速 IP ルータにおいてフロー識別を行なうために、一般的に入手可能な CPU およびメモリを用いた検索アルゴリズムの提案を行う。

CPU を用いたフロー識別では、フロー情報検索のためのメモリアクセスがボトルネックとなる。CPU の高速化が著しい近年では、CPU の動作速度に対して記憶領域へのアクセス速度は非常に遅く、CPU 資源の利用率低下の原因となりうる。このため、一般的な CPU ではより高速にアクセス可能なキャッシュメモリを用いることで、メモリのアクセス遅延を小さくしている。いいかえれば、キャッシュメモリを効率的に利用することで、メモリアクセス回数の低減を実現できることになる。したがって本稿では、キャッシュメモリを効率よく利用するためのフロー検索テーブルのデータ構造についてまず検討する。さらに、CPU のキャッシュメモリを用いた高速化、特にキャッシュメモリにおけるパラメータが検索速度に与える影響について検証する。

本稿では、まず 2 章でフロー識別のために必要な情報とフロー情報保持のための必要領域について明らかにする。また、3 章ではキャッシュメモリの構造とパラメータによる影響について説明する。次に 4 章で提案するフロー情報検索アルゴリズムを示し、5 章で数値結果によるキャッシュメモリの効果について示す。最後に 6 章でまとめと今後の課題について述べる。

2 フローの識別と保持する情報

本節では、フロー識別に必要な情報と、フロー情報の保持に必要なメモリ領域について検討する。

2.1 フロー識別子

フローを識別するには、同一コネクション上を流れるパケットの集合を 1 つのフローと考えるのが適当である。したがって、本稿ではフローを以下のパケット内情報によって選別する。

{*Src IP, Dest IP, Src Port, Dest Port, Protocol*}

しかし、この場合にはフロー識別に要するフィールドは全部で 104 ビット長になり、検索のための情報量が膨大なものとなる。また HTTP などでは、1 つの Web ページを表示させるためのテキストや画像をそれぞれ別の TCP コネクションによって送信する。しかし、1 つの TCP コネクションで通信する他のアプリケーションとの公平性を考えた場合、これらの複数コネクションを集約し、1 つのフローと考えた方が適当である。さらに、*Protocol* は TCP/UDP/IP など、IP パケットの種類を示すフィールドであるが、複数の異なる *Protocol* を持つアプリケーションはほとんどなく、フロー識別を行うための情報としては無視できる。したがって、本稿ではフロー検索を

{*Src IP, Dest IP, Port Number* }

で示されるフロー識別子を用いて行なうものとする。ここで *Port Number* は、*Src Port* あるいは *Dest Port* のうち、値の小さいものとする。これは、代表的なアプリケーションの大半が一般ユーザが利用できない 1024 以下のポート番号を使用しているためである。

2.2 保持するフロー情報

フロー情報は、適用するポリシーサービスの種類によりさまざまなパラメータを取る。ここでは、いくつかの例を示し必要パラメータの種類とサイズについて検討する。以下の例より、本稿ではフロー情報として 2 バイトの領域を使用する。

フロー分類に基づいた RED

ルータにおいて、コネクションの送出レートに応じて公平にパケット棄却を行うためには、RED (Random Early Detection) 方式 [3] が有効であるとされている。FIFO (First In First Out) 方式では、一度パケット棄却が発生すると続いて到着するパケットが連続して棄却されるため、特定のコネクションだけにパケット棄却が集中するという不公

平が生じる。RED 方式では、あらかじめキュー長に閾値 (Threshold) を設け、閾値を越えた場合に確率 α でパケットを棄却することで、特定のコネクションにパケット棄却が偏らなくなる。しかし単純な RED 方式では、アプリケーション間の公平性や送信端末によるパケット棄却率の重み付けなど、柔軟な設定ができない。そこで、フロー分類に基づいた RED 方式を考える必要がある。

フロー分類に基づいた RED では、到着したパケットをフロー情報に基づいて分類し、フローごとに閾値を定義する。パケットが到着したとき、キュー長がフロー情報に書かれている閾値を越えている場合、確率 α でパケットを棄却する。この方式を実現するためには、フロー情報にキュー長の閾値を記入する必要がある。バッファサイズが 128KB ～256KB であれば、情報量として 2 バイトあれば良いことになる。2 バイトの領域で保持できるキュー長の範囲は 64KB までであるが、2 ビットシフトさせる (1 ビット = 4 バイト) ことで、256KB まで表現することは可能である。4 バイトはパケットサイズに対して十分小さい値であり、4 バイト単位で閾値を設定することによる影響はほとんどないものと考えられる。

Core-Stateless ルータへの適用

フローに対して公平なレート制御を行う Core-Stateless ルータ [4] では、エッジルータにおいて、フローのレートを計算し、パケットにラベルとして記入する。コアルータでは、到着したパケットに書かれたレートに基づきレートを制御することで、コアルータのオーバーヘッドを大きくすること無くフロー間の公平性を実現している。エッジルータでは、到着したパケットをフローに分類し、レートを計算、ラベルに記入する作業が行なわれるため、パケット到着レートをフロー情報として記録する必要がある。コアルータ内部では、フロー間で公平なレートを提供 (fair share rate) している。このため、レートは (帯域 / フロー数) で表される。ルータに接続されているリンク容量が分かっていることから、フロー数で分割した単位の整数値を保持できればよいことになる。情報量を 2 バイトとすれば、同時に 65,000 フロー到着した場合のレートを表現できる。

DRR 利用した fairness の実現

DRR (Deficit Round Robin) [5] 処理規律を使うことで、フローごとに重みを付けたレート制御を行なうことができる。DRR で必要な情報は、Round Robin でそのキューにパケット送出権がある場合のパケット送出量である。1 度に最低 1 個のパケットを送り出すことができるようにするためには、送出量は最大パケットサイズ (64KB) を指定できるようにすればよい。この情報を保持するためには、2 バイトの領域が必要である。

3 キャッシュメモリの特性

本節では、CPU のもつキャッシュメモリの特性について簡単に説明し、キャッシュメモリの持つパラメータの影響について検討する。

3.1 キャッシュメモリの構造

現在広く用いられている 2 段構成キャッシュメモリでは、CPU に最も近い 1 次キャッシュにおいてキャッシュミスが発生すると、CPU は 2 次キャッシュを検索し、データが存在する場合、2 次キャッシュの内容を 1 次キャッシュに読み込む。2 次キャッシュにおいてデータが存在しない場合は、メインメモリの内容を読み込む。1 次キャッシュは、通常 CPU のチップ内に存在するため、CPU 速度と同じスピードで読み書きできるが、CPU チップの面積の制約上実装できる容量は比較的小さく、4 ～ 数十 KB 程度である。

キャッシュメモリの持つパラメータとしては、次の 3 つが考えられる。以下では、各パラメータの変動による影響について考察する。その定量的な評価については後に示す。

ブロックサイズによる影響

キャッシュミスが発生した場合、メモリからキャッシュメモリに読み込むデータの単位をブロックサイズと呼ぶ。ブロックサイズが大きければ、連続するアドレスへの読み書きがすべてキャッシュメモリで行われるため、キャッシュヒット率が上昇するが、キャッシュメモリへの読み込みにかかる遅延 (= ブロックサイズ / バス幅 × メモリアクセス遅延) が増大する。通常、16、32、64 バイトがよく用いられている。

セット連想マッピング方式

キャッシュメモリは、検索を高速にするためにメモリアドレス領域によって使用できるキャッシュブロック番号をダイレクトに指定している。このため、同一ブロック番号に属するメモリ領域は同時に 1 箇所しかキャッシングを行えず、キャッシュブロックによる利用率の偏りが発生する。そこで、複数のブロックを 1 つのセットとし、同一セットに属するメモリ領域を複数キャッシングできるセット連想マッピング方式が広く用いられている。n ウェイセット連想マッピング方式では、同一セットに属する n 箇所のメモリ領域を同時にキャッシングできる。n を増加させることで、ブロック間の利用率の偏りが減少し、キャッシュの利用率が向上するが、セット内でブロックを識別するため連想記憶が必要となる。高速動作を行う連想記憶はコストが非常に高いため、n を変化させた場合の効果を調べる必要がある。

キャッシュサイズ

キャッシュサイズを大きくすることでキャッシュヒット率が向上するが、大容量キャッシュメモリはコストが急激に上

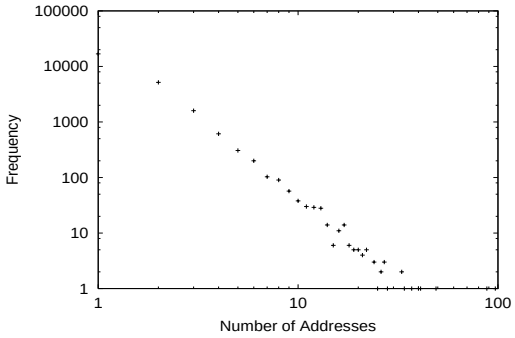

図 1: 上位 16 ビットによる相手先アドレスの分類

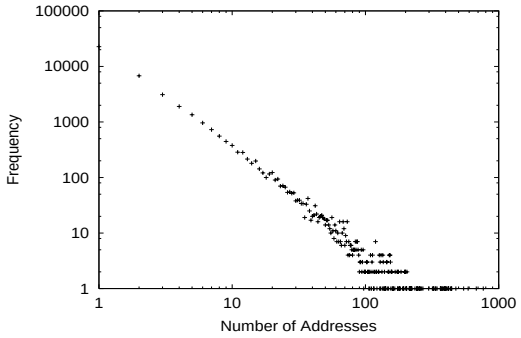

図 2: 受信アドレスごとの送信元アドレスの種類

昇する。しかし一方で、すでに十分なキャッシュ容量があればキャッシュヒット率はほとんど向上しない。したがって、フロー識別を行うのに十分なキャッシュ容量を明らかにする必要がある。

4 フロー情報の検索

4.1 フロー識別子の分割

フロー識別子を元にフロー情報を検索する場合、メモリアクセス時間がもっとも大きいオーバーヘッドとなる。いいかえれば、メモリ参照回数を少なくすることで、フロー情報を高速に検索することが可能となる。もちろん、最も高速に検索するためにはメモリ参照回数を 1 回にすればよいが、そのためには 2×2^{80} バイトのメモリ領域が必要となる。このように、速度とメモリ領域はトレードオフの関係にあり、データ構造の決定には実現可能なメモリ量と要求される速度の両面から考慮する必要がある。例として、大阪大学の外部ルータへのアクセス回線に接続されたルータにおけるトレースデータ 2,700 万パケットについて上位 16 ビットごとに相手先 IP を分類したときの、下位 16 ビットのアドレスの種類の関係を図 1 に示す。相手先 IP の種類は、$\alpha = 1$ の Zipf 法則 [6] に従うことが分かる。また、過去に参照されたアドレスが一つも存在しないネットワークは 40,000 以上にのぼり、全体 (65,536) の 60% 程度のメモリ空間が利用されていないことになる。さらに、過去に参照されたアドレスの種類が最も多いネットワークについても、その種類の数は 97 と、下位 16 ビットが取り得る値の 1% 程度であることが分かる。このことからも、参照回数を 1 回にするために 2^{32} 個 (IP アドレスの個数) の配列を用意するよりも、複数のテーブルを持たせる方がメモリ利用効率の上で効率的であることが分かる。そこで、本稿ではフロー識別子を 1. 受信先 IP (1〜16 ビット)、2. 受信先 IP (17〜32 ビット)、3. 送信元 IP およびポート番号、の 3 つの部分 (キー) に分割し、検索を行なうこととする。

4.2 ハッシュテーブルを用いたテーブルサイズの圧縮

各キーのフィールド長よりエントリ数が非常に小さいテーブルについては、ハッシュテーブルを用いることとする。

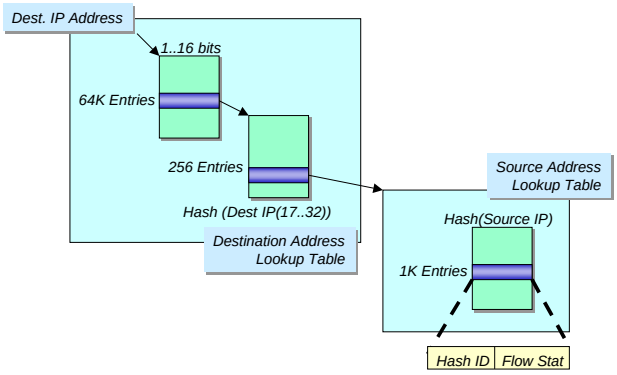

図 3: テーブル構造

ハッシュテーブルを用いることで、配列による検索に対してほとんどオーバーヘッドがかかることなくテーブルサイズを圧縮させることが可能である。

受信先 IP の下位 16 ビットをテーブルに格納する場合、図 1 より最大 97 のエントリが同時に格納できればよいので、256 程度のハッシュテーブルがあれば十分である。ハッシュテーブルの重複があった場合、空きのエントリが存在するまで次のテーブルエントリを検索する。メモリのキャッシングはブロック単位で行われていることから、この検索は 1 次キャッシュ上で行われるため、ハッシュ値が重複した場合でもメモリアクセス回数を軽減することができる。

次に、図 2 に受信先 IP ごとに分類した、(送信元 IP、ポート番号) の種類の数を示す。図より、受信先 IP ごとに最大 3000 程度の送信元 IP が存在することから、5000 程度のエントリを持つハッシュテーブルを作成すればよい。また、ハッシュ値を 16 ビットで保持することで、ほぼ重複することなく送信アドレスを特定することができるようになる。

4.3 検索アルゴリズム

以上のキーを元に、フロー情報を検索するためのテーブルのデータ構造を図 3 に示す。検索テーブルは大きく 4 つの部分に分かれる。

1. 受信先 IP 上位 16 ビットを検索するテーブル。2. へのポインタが格納されている。

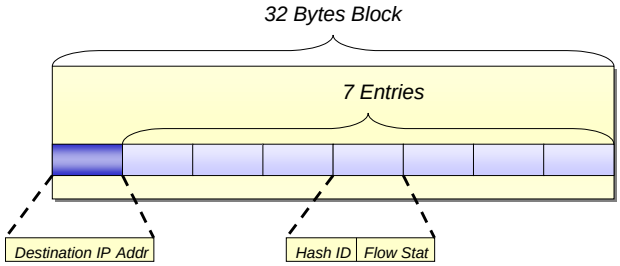

図 4: キャッシュメモリ 1 ブロックあたりのデータ構造

2. 受信先 IP 下位 16 ビットに対するハッシュテーブル。
   3. へのポインタが格納されている。
3. 送信元 IP、ポート番号を格納するハッシュテーブル。
   ハッシュエントリは、以下の構成になっている。
   - 受信先アドレス。検索結果が正しいかをチェック
   - 16 ビット長の送信元に関するハッシュ値
   - フロー情報 (2 バイト)

キャッシュメモリのブロックサイズを 32 バイトとした場合の、1 ブロックあたりのデータ構造を図 4 に示す。検索結果が正しいかをチェックする受信先アドレスが各ブロックの先頭に記録されている。これにより、受信先アドレスのチェックによってブロックがキャッシングされ、フロー情報へのアクセス時にメモリから読み書きする必要がなくなる。この時、1 ブロックあたり保持することのできるフロー情報は、$(32-4)/4=7$ となり、受信先 IP ごとに送信元 IP とポート番号を格納するためのハッシュテーブルは、128 ブロック程度で実現することができる。最後に、検索に必要なアドレスの計算アルゴリズムを以下に示す。

- `Ptr1 = HeadTable[Dest Addr(1..16)]`
- `Ptr2 = Ptr1[Hash(dest)(Dest Addr (17..32))]`
- `statePtr = Ptr2 × 32 × 128 + 32 × (Hash(src) / 7) + (Hash(src) mod 7) + 4`

5 評価結果

本節では、フロー識別の処理時間とキャッシュメモリの効果についての評価結果を示す。評価には、以下に示す PC 上で提案アルゴリズムを実装し、パケットトレースデータを用いて行った。

- Intel Pentium II 450 MHz
  - 16 KB 4 way associative L1 cache
  - 512 KB 4 way associative L2 cache
- 256 MB memory
- Linux 2.0.36
- GNU gcc version 2.95.2 19991024 (release)

トレースデータは、大阪大学のルータ上に設置した OC3MON [7, 8] で収集した、約 2,700 万パケットのデータを使用した。また、キャッシュメモリのパラメータに対する影響を調べるため、キャッシュメモリシミュレータ Dinero IV version 7 [9] を使用した。さらに、実際の検索にかかる

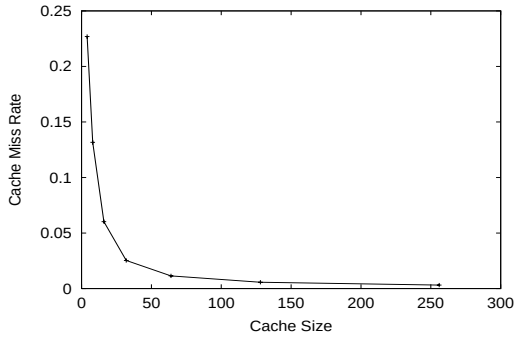

図 5: キャッシュサイズによる効果 (L1)

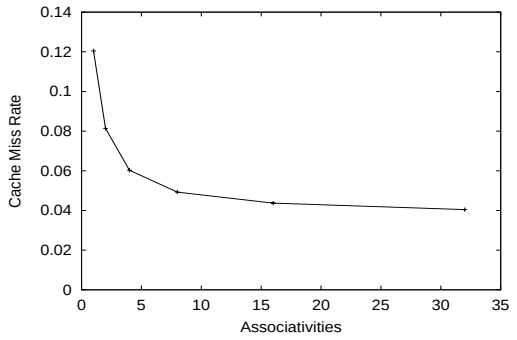

図 6: セット連想マッピングの効果 (L1)

平均クロック数を求めるため、Pentium II の RDTSC 命令を用いた CPU サイクルの計測 [10] を行った。

Dinero IV では、全メモリアクセス回数 N 及び L1、L2 キャッシュごとのキャッシュミス率 M_{L1}、M_{L2} を出力する。この結果と、L1、L2、メインメモリに対するアクセス遅延 N_{L1}、N_{L2}、N_{Mem} を用いることで、以下の式により 1 パケットあたりの検索遅延を導出することができる。

$$\begin{aligned} meanaccessdelay &= N/P \times (1-M_{L1})N_{L1} \\ &\quad \times M_{L1}(1-M_{L2})N_{L2} \\ &\quad \times M_{L1}M_{L2}N_{Mem} \end{aligned} \quad (1)$$

5.1 1 次キャッシュによる影響

はじめに、1 次キャッシュのパラメータによる影響についての結果を示す。図 5 は 1 次キャッシュにおける、キャッシュサイズとキャッシュミス率の関係を表している。キャッシュサイズの増加によって、キャッシュミス率が急激に減少していることが分かるが、64 KB 以上のキャッシュサイズでは、ほとんど効果が見られないことが分かる。

また図 6 で、1 次キャッシュのセット連想マッピングの効果を示す。この場合も、セット数を増加させることでキャッシュミス率が減少しているが、8 以上では目立った効果が見られないことが分かる。

5.2 2 次キャッシュによる影響

次に 2 次キャッシュのパラメータによる影響について、キャッシュサイズ、セット数の効果による効果をそれぞれ図 7、8 に示す。1 次キャッシュの場合と同様、過剰なキャッシュサイ

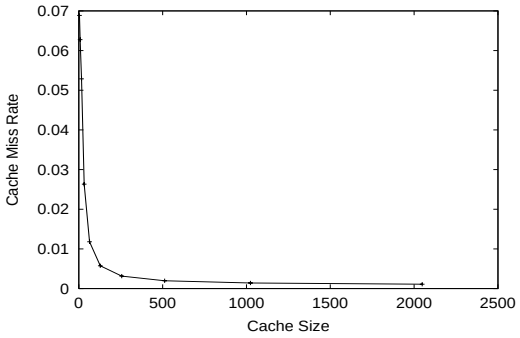

図 7: キャッシュサイズによる効果 (L2)

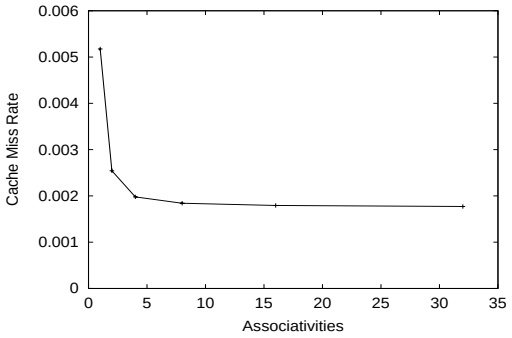

図 8: セット連想マッピングの効果 (L2)

ズの増大などはあまり効果がないことが分かる。また、図より 2 次キャッシュの場合の最適なパラメータは、256 KB、セット数 4 であることが分かる。

5.3 CPU 処理サイクル数と必要メモリ容量

前述の Linux PC 上で実装しアルゴリズムにトレースデータを適用した場合の 1 パケットあたりのフロー識別に要する CPU 処理サイクル数及び遅延時間を表 1 に示す。実装を行った PC 上では、1 秒間に約 200 万パケット処理できることから、Gbps クラスの回線容量にも適用できることが分かる。また、必要なメモリサイズは、49.1 MB となった。これは、全てのフローに対して同じ容量のテーブル領域を確保したためで、実際にはフローごとに異なるテーブルサイズを用いることで、必要となるメモリ容量を軽減することが可能である。これについては今後の課題である。

6 まとめと今後の課題

本稿では、ポリシーサービスを提供する IP ルータにおける、フロー情報検索のためのデータ構造を提案し、アルゴリズムを実現するサイクル数、メモリアクセス時間等を考慮したパケット処理能力の導出、メモリ利用効率の改善について検討を行なった。また、シミュレーションによってキャッシュメモリのパラメータによる効果を明らかにした。今後の課題としては、大阪大学のトレースデータ以外にパブリックドメインのトレースアーカイブの使用し、提案アルゴリズムの一般性を示す必要がある。またアルゴリズムをルータ実装し、稼働させた場合の評価を行っていく予定である。

表 1: CPU 処理サイクル数

CPU サイクル数	処理時間	パケット処理能力
228.772335	508.383 nsec	1.967 MPPS

参考文献

[1] Tzi-cker Chiueh and P. Pradhan, “High–performance IP routing table lookup using CPU caching,” in *Proceedings of IEEE INFOCOM ’99*, April 1999.

[2] T. V. Lakshman and D. Stiliadis, “High–speed policy–based packet forwarding using efficient multi–dimensional range matching,” in *Proceedings of ACM SIGCOMM ’98*, pp. 203–214, September 1998.

[3] S. Floyd and V. Jacobson, “Random early detection for congestion avoidance,” *IEEE/ACM Transactions on Networking*, pp. 397–413, July 1993.

[4] I. Stoica, S. Schenker, and H. Zhang, “Core–stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks,” in *Proceedings of ACM SIGCOMM ’98*, pp. 118–130, September 1998.

[5] M. Shreedhar and G. Varghese, “Efficient fair queueing using deficit round robin,” *IEEE/ACM Transactions on Networking*, vol. 4, pp. 375–385, June 1995.

[6] M. Aida, N. Takahashi, and T. Abe, “A proposal of dual Zipfian model for describing HTTP access trends and its application to address cache design,” *Transactions of IEICE*, vol. E81–B, pp. 1486–1496, July 1998.

[7] K. Thompson, G. J. Miller, and R. Wilder, “Wide–area Internet traffic patterns and characteristics,” *IEEE Network*, pp. 10–23, November 1997.

[8] S. Ata, M. Murata, and H. Miyahara, “Analysis of network traffic and its application to design of high–speed routers,” to appear in *IEICE Transactions on Information and Systems (D-I)*, May 2000.

[9] J. Edler and M. D. Hill, “Dinero IV : Trace-driven uniprocessor cache simulator,” available at `http://www.cs.wisc.edu/~markhill/DineroIV`, 1994.

[10] Intel Corporation, “Using the RDTSC instruction for performance monitoring,” Pentium II Application Note, available at `http://developer.intel.com/drg/pentiumII/appnotes/RDTSCPM1.HTM`, 1998.