フローを考慮した経路制御機構

小巻 賢二郎† 所 真理雄‡

†慶應義塾大学大学院 理工学研究科 計算機科学専攻
‡慶應義塾大学

〒223–8522 神奈川県横浜市港北区日吉 3–14–1
Tel: 045–560–1150 Fax: 045–560–1151
E-mail: komaki@mt.cs.keio.ac.jp

あらまし 本研究では、(終点アドレス、始点アドレス、終点ポート番号、始点ポート番号) の 4 つの値によりフローが一意に定まることに注目し、フロー毎に経路制御を行う機構として **froute** を提案する。フローを考慮した経路制御機構の実現方法は様々であるが、froute の設計は既存の経路制御表に拡張を加えるものとし、既存の経路制御表に用いられている Radix Tree 構造を利用する。その際、可変長アドレスや不連続ネットマスクへの対応といった Radix Tree 構造の特徴を利用し、既存の経路制御機構と透過的に扱える経路制御機構を実現する。さらに既存の経路制御表や関連研究との比較評価を行い、froute は十分使用に耐え得り、実用性があることを示す。

キーワード 経路制御機構、経路制御表、フロー、Radix Tree、二分探索

A Routing Mechanism based on the Flow

Kenjiro Komaki† Mario Tokoro‡

†Department of Computer Science, Faculty of Science and Technology, Keio University
‡Keio University

3–14–1 Hiyoshi, Kohoku–ku, Yokohama 223–8522, Japan
Tel: +81-45–560–1150 Fax: +81-45–560–1151
E-mail: komaki@mt.cs.keio.ac.jp

Abstract In this research, we notice that (destination address, source address, destination port number, source port number) identifies **flow** uniquely, so we suggest routing mechanism **froute**, which can route every flow individually. In various way to accomplish froute, we choose expanding routing table and utilize Radix Tree mechanism which is used in existing routing table. With the special feature of Radix Tree mechanism which is the support for variable length address and non-consecutive netmask, we accomplish the routing mechanism which can be treated transparently with existing mechanism. Futhermore, we compare froute with existing mechanism and related works, then prove froute is useful.

key words routing mechanism, routing table, flow, Radix Tree, binary search

1　はじめに

インターネット上での経路制御は、各ルータが保持する経路制御表と転送パケット中のヘッダに記載されている終点アドレスにより成り立っている。ルータはパケットを受信すると、その終点アドレスを検索キーとしてそのルータが保持している経路制御表を検索する。これにより各終点アドレスに対応した次ホップルータが決定し、受信パケットを次ホップルータへ転送できる。

そのインターネットも近年、急激な成長を遂げ、世界規模の巨大ネットワークに発展した。それに伴い、利用目的の多様化による様々なアプリケーションや、ネットワーク技術の発展による多様な特性を持つ通信経路が登場している。こういった様々なアプリケーションや多様な通信メディアの登場により、終点アドレスだけでなく始点アドレスやポート番号も考慮した経路制御が必要とされ始め、既存の経路制御機構では不十分な面が発生している。

そこで本研究ではフローの概念に着目し、フローを考慮した経路制御機構**froute**を提案する。各パケットを、そのパケットが属するフロー毎に経路制御することにより、始点アドレスやポート番号を考慮した経路制御を行うことが可能となる。froute は既存の経路制御表に採用されている Radix Tree 構造 [5] の特徴を利用して実装する。これにより既存の経路制御機構を変更することなく、終点アドレスのみのエントリとフローに対するエントリを透過的に検索できる。

本稿は本章も含めて 6 章で構成されている。2 章では既存の経路制御機構について述べ、その問題点を示す。3 章ではフローを考慮した経路制御機構 froute を提案、設計、実装する。4 章では 3 章で実装した froute の評価を、検討を行う。5 章では今後の課題について述べる。6 章で関連研究について述べ、7 章では本稿のまとめを述べる。

2　既存の経路制御機構

本章では、既存の経路制御表のエントリの種類、その構造について説明した後、既存の経路制御表の問題点を挙げる。

2.1　経路制御表のエントリ

既存の経路制御機構ではパケットの終点アドレスを検索キーとして経路制御表を検索するが、検索対象である経路制御表のエントリには、次の 3 種類がある。

- ホスト経路

  ある一つのホストへの経路。

- ネットワーク経路

  あるネットワークに属するホストへの経路。

- デフォルト経路

  他の全てのエントリにマッチしなかった場合の経路。

表 1に経路制御表の例を示す。

表 1: 経路制御表の例

終点アドレス	次ホップルータ	経路の種類
default	131.113.82.1	デフォルト経路
131.113.32/24	131.113.82.33	ネットワーク経路
131.113.32.240	131.113.82.2	ホスト経路

2.2　経路制御表の構造

既存の経路制御表は Radix Tree 構造 [5] を利用して構成されている。Radix Tree は図 1のような二分木構造となっている。経路制御表の各エントリの IP アドレスを単なるビット列と考え、その各ビットが 0 か 1 かによって各分岐では枝が分かれていき、最終的にそれぞれの葉にエントリ一つ一つが対応する。図 1の Radix Tree 構造に対応する経路制御表を表 2に示す。ここでは経路制御表の各エントリの終点アドレスが Radix Tree 構造で互いにどのような形をとるかを問題にしているため、表 2では次ホップルータのエントリは重要ではなく、代わりに Radix Tree 構造の理解を助けるため、各エントリの終点アドレスの 16 進数表現を示す。

表 2: 図 1 に対応する経路制御表

Destination Address	16 進数表現
default	0x00000000
128.32.33.5	0x80202105
140.252.13.32	0x8cfc0d20
140.252.13.33	0x8cfc0d21
140.252.13.34	0x8cfc0d22
140.252.13.35	0x8cfc0d23
140.252.13.65	0x8cfc0d41

Radix Tree による経路制御表の検索には次の 3 つの種類がある。

- 完全一致 (Full Match)
- ネットマスクを考慮した検索 (Network Match)

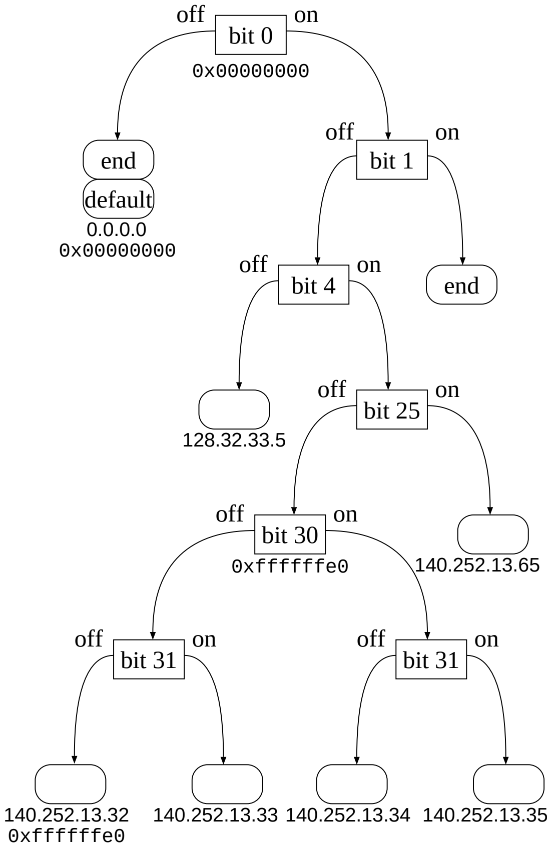

図 1: Radix Tree 構造の例

- バックトラッキングの発生する検索 (Matching with Back Tracking)

ここで図 1、表 2に示す経路制御表を保持するルータにおいて、Radix Tree 構造を利用した経路制御表の検索を行う際の手順を示す。

- 完全一致 (**Full Match**)

  終点アドレスが 140.252.13.65 のパケットを受信したとする。この終点アドレスが検索キーとなる。木の根から検索を開始し、140.252.13.65 は 0 ビット目は 1 であるため右の枝をたどる。そうして順に、1 ビット目は左、4 ビット目は右、25 ビット目は右の枝へと進み、140.252.13.65 の枝に到達する。最終的に葉に到達した場合、葉の終点アドレスと検索キーのアドレスとを比較する。この場合、到達した葉の保持しているアドレスと完全一致 (**Full Match**) するため、次ホップルータのアドレスが定まりこの経路を採用する。

- ネットマスクを考慮した検索 (**Network Match**)

  終点アドレスが 140.252.13.60 のパケットを受信したとする。Full Match の時と同様に、0 ビット目から検査ビットを順に調べていく。すると、最終的に 140.252.13.32 に到達するが、Full Match には失敗する。今度はネットマスクを考慮した検索 (**Network Match**) を行う。140.252.13.32 のネットマスクは `0xffffffe0` であり、検索キーである 140.252.13.60 とマスクした結果 140.252.13.32 となる。これは葉の保持しているアドレスと一致するのでこの経路を採用する。

- バックトラッキングの発生する検索 (**Matching with Back Tracking**)

  終点アドレスが 140.252.13.62 のパケットを受信したとする。Full Match、Network Match と同様に 0 ビット目から検索していくと、最終的に 140.252.13.34 に到達するが、Full Match、Network Match ともに失敗する。この場合、Radix Tree を根の方まで戻る。そしてネットマスクを保持している分岐に到達した所で止まる。ここでは bit 30 の分岐である。この処理をバックトラッキングという。bit 30 の保持しているネットマスク `0xffffffe0` で検索キー 140.252.13.62 をマスクすると、140.252.13.32 となる。このアドレスを新たな検索キーとして、bit 30 を頂点としたサブツリーで再び Full Match から検索を開始する。すると最終的に 140.252.13.32 の葉に到達し、これは検索キーと完全一致するため、この経路を採用する。

Radix Tree の根、すなわち bit 0 の分岐では `0x00000000` のネットマスクを保持しているため、全てのマッチングに失敗して根に戻ってきた場合は、全て左の枝をたどることになる。そして、デフォルト経路が設定されていれば必ずそれにマッチし、最終的にデフォルト経路が採用されることがわかる。

2.3 既存の経路制御表の問題点

2.1節、2.2節で述べたように、Radix Tree 構造を利用した既存の経路制御表ではパケットの終点アドレスのみでしか検索することができない。しかし近年多様な目的のアプリケーションや多様な特性を持つ通信経路が登場しており、終点アドレスだけでなく始点アドレスやポート番号を考慮した経路制御機構が要求されている。次にその例を示す。

1. ポート番号による経路制御

   終点アドレスのみによる経路制御では、同一の終点アドレスを持つパケットは同一の経路で配送される。しかし、WWW や FTP のようなバースト性を持つアプリケーションと telnet のような遅延に敏感なアプリケーションのパケットが同一経路で配送されると、telnet のパケットは WWW や FTP のパ

ケットに帯域を奪われ、十分な遅延応答を得られない。ポート番号を考慮した経路制御が可能であれば、WWW や FTP のパケットと telnet のパケットを異なる経路に配送でき、互いに干渉することはない。

さらにポート番号によって異なる経路を選択できれば、経路の特性によって最適なアプリケーションを選択することが可能となる。例えば、広帯域だが遅延が大きい衛星回線は、WWW や FTP のようなバースト性のあるアプリケーションに適している。それに対し帯域は狭いが遅延は非常に小さい専用線は、telnet のような帯域を多く必要としないが遅延に敏感であるアプリケーションに適している。

2. 始点アドレスによる経路制御

   課金しているユーザが送信したパケットのみをある経路に経路制御したい場合を考える。従来の終点アドレスのみによる経路制御機構では、終点アドレスが同じパケットは始点アドレスに関わらず同じ経路に配送されるため、送信者による区別はできない。終点アドレスに加え始点アドレスも考慮した経路制御が可能であれば、送信ユーザ毎に異なる経路を選択することができる。特に近年、資源予約通信が注目されており、資源予約通信を行う場合にはこの始点アドレスによる経路制御は重要であると考えられる。

上記の問題を解決するために、3章では始点アドレスやポート番号を考慮できる新たな経路制御機構の提案を行う。

3 フローを考慮した経路制御機構 froute の設計と実装

2.3節で示したように、パケットの終点アドレスに加え、始点アドレス、ポート番号も考慮した経路制御機構の需要は高くなっている。そこで本研究ではフロー (**flow**) という概念に着目し、フローを考慮した経路制御機構 **froute** を提案する。フローとは、「インターネット上のある始点から終点への一連のパケットの流れ」である。パケットの属するフローはフロー識別子 (終点アドレス、始点アドレス、終点ポート番号、始点ポート番号) で一意に識別できる。フロー識別子を考慮することにより、始点アドレスやポート番号を考慮した経路制御が可能となる。

froute の実装方法は様々考えられるが、本研究では既存の経路制御表のエントリに拡張を加える。既存の経路制御表に採用されている Radix Tree 構造の特徴である、

- 可変長アドレスへの対応
- 不連続ネットマスクへの対応

という二点を利用することにより、経路制御機構に変更を加えずにフロー識別子を考慮した経路制御機構が可能となる。また既存の経路制御表を利用しているため、終点アドレスのみのエントリとフローのエントリを透過的に扱うことができる。

froute の実現のために Radix Tree の検索キーである `sockaddr_in` 構造体に注目する。従来の終点アドレスのみのエントリでは検索キーとして `sockaddr_in` 構造体を用いているが、froute ではフロー毎のエントリの検索キーとして `sockaddr_fl` を用いる。図 2に `sockaddr_in` 構造体を、図 3に `sockadddr_fl` 構造体を示す。

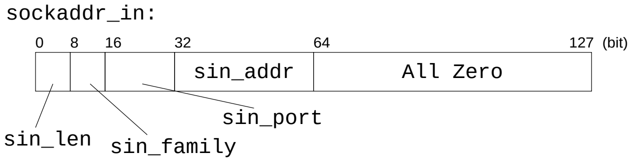

図 2: `sockaddr_in` 構造体

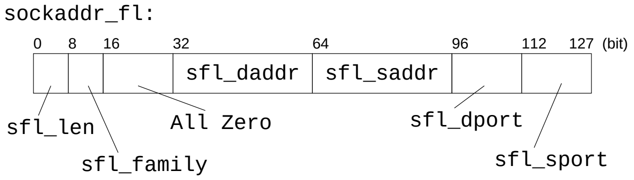

図 3: `sockaddr_fl` 構造体

ここで `sockaddr_in` 構造体の下位 64 ビットが空き領域であることに注目する。`sockaddr_fl` では終点アドレスに続くこの下位 64 ビットに、始点アドレス、終点ポート番号、始点ポート番号を格納する。これにより `sockaddr_fl` は従来の `sockaddr_in` 構造体と同じ大きさで実現でき、froute では既存の経路制御機構のエントリとフローのエントリを透過的に扱える。

図 2において、`sin_len` はアドレス長、`sin_family` はアドレスファミリ、`sin_port` はポート番号、`sin_addr` は IP アドレスを示す。アドレス構造体にはポート番号も含まれているが、実際の経路制御の際には 32 ビット目からが有効であり、`sin_port` メンバは参照されない。図 3では、`sfl_len` はアドレス長、`sfl_family` はアドレスファミリ、`sfl_daddr` は終点アドレス、`sfl_saddr` は始点アドレス、`sfl_dport` は終点ポート番号、`sfl_sport` は始点ポート番号を示す。

さらに froute では、Radix Tree 構造を利用した経路制御表の持つ可変長アドレスへの対応や不連続ネットマ

スクへの対応という特徴を利用する。`sockaddr_in` 構造体では 32 ビットの終点アドレスを示す `sin_addr` メンバのみを検索キーとして使用していた。`sockaddr_fl` では可変長アドレスへの対応という点を利用し、`sfl_daddr`、`sfl_saddr`、`sfl_dport`、`sfl_sport` の計 96 ビットを一つのアドレスのとして考え、検索キーに使用する。同様にネットマスクも、終点アドレスのネットマスク、始点アドレスのネットマスク、終点ポート番号のネットマスク、始点ポート番号のネットマスクの計 96 ビットの一つのネットマスクとする。ここで、ポート番号は指定されていればネットマスクは全て 1、指定されていなければ全て 0 となる。96 ビットのネットマスクは不連続になるが、Radix Tree 構造は不連続ネットマスクに対応しているため、問題はない。この際、Radix Tree における検索機構は最長ネットマスク一致を行うため、経路制御表のエントリ中で最適な経路が選択される。

表 3に、本研究で提案するフローを考慮した経路制御表の例を示す。表 3に示す経路制御表を持つルータが、フロー識別子 (131.113.82.93, 131.113.82.89, 23, 1023) のパケットを受信したとすると、マッチするエントリは最長ネットマスク一致により 3 のエントリとなる。

froute の実装を FreeBSD 2.2.5-RELEASE 上で行った。既存の経路制御表はカーネル内に実装されており、froute は既存の機構を拡張するため、カーネルに変更を加えている。

4 評価および検討

本章では、froute の実用性を説明するために処理時間の評価を行う。その際、比較対象として既存の経路制御機構の他に、froute と同等の機能を提供する SuMiRe[7] を加えた。SuMiRe については 6.3節で説明する。

4.1 評価

本研究で新たに提案する経路制御機構 froute の実用性を示すために、既存の経路制御機構、froute、SuMiRe の 3 つに関する性能比較評価を行う。froute、SuMiRe ともに FreeBSD 2.2.5-RELEASE の実装を利用した。評価は図 4に示す環境で行う。

4.1.1 評価方法

Router B において froute および SuMiRe を実装し、froute、SuMiRe、通常のカーネルの三種類において、次の処理にかかる時間を測定した。

評価 1 経路制御表の操作

評価 2 パケットの転送

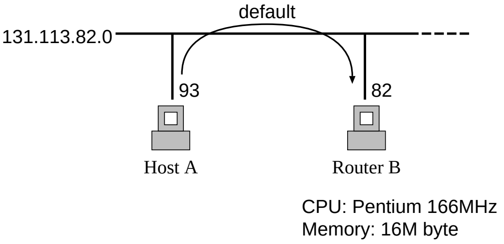

図 4: 評価環境

経路制御表の操作には追加、削除などが考えられるが、一番オーバーヘッドが大きいと思われるエントリの “追加” について測定を行う。このエントリ追加時間の測定は Router B 単独で行うが、パケットの転送時間の測定は Host A を使用する。Host A のデフォルト経路を Router B に設定しておき、経路制御でデフォルト経路がマッチするパケットを送信する。このパケットを Router B が受信した際、転送処理を行う時間を測定した。さらに経路エントリの増加に伴う処理時間の変化を測定するため、評価 1、評価 2 ともに経路エントリ数を 10 から 10000 まで変化させ、それに対する処理時間を測定した。

エントリ追加時間の測定、および経路エントリ数に対する転送時間の測定時には、追加するエントリは通常のカーネルでは終点アドレスのみのエントリ、froute、SuMiRe の場合はフローに対するエントリとする。両者とも、追加するエントリはランダムに選ばれる。

4.1.2 評価結果

評価 1 の結果を図 5に、評価 2 の結果を図 6にそれぞれ示す。ともに、x 軸は対数軸である。ここで x 軸は経路エントリ数であり、y軸は評価 1 では経路の追加に要する時間、評価 2 では転送処理に要する時間 (ともに単位は μsec) を示す。評価 1、評価 2 ともに各値は 100 回測定の平均値である。

4.2 検討

4.2.1 既存の経路制御機構との比較

先に述べたように froute は既存の機構と同じく Radix Tree 構造を利用しており、両者の異なる点はエントリ内のアドレス長のみである。既存の機構では終点アドレス 32 ビット、froute ではフロー識別子のアドレス長 96 ビットを検索するため、両者の処理時間には差が生じると考えられる。また Radix Tree は木構造であり、二分木の探索には一般に $O(\log n)$ かかるため、処理時間は対数軸に対して直線となると考えられる。

図 5、図 6より、既存の経路制御機構と froute の処理時間に差は少しあるものの、既存の機構と froute の処理

表 3: フローを考慮した経路制御表の例

	終点アドレス	始点アドレス	終点ポート番号	始点ポート番号	エントリの種類
1	*	*	*	*	デフォルト経路
2	131.113.82.93	131.113.82.89	22	1023	ある特定のフロー
3	131.113.82.93	131.113.82.89	*	*	ホストからホストへの任意のフロー
4	131.113.82/24	131.113.82.89	*	*	ホストからネットワークへの任意のフロー
5	131.113.82.93	*	*	*	既存の経路制御表と同じエントリ
6	131.113.82.93	*	23	*	あるホストへの telnet のフロー

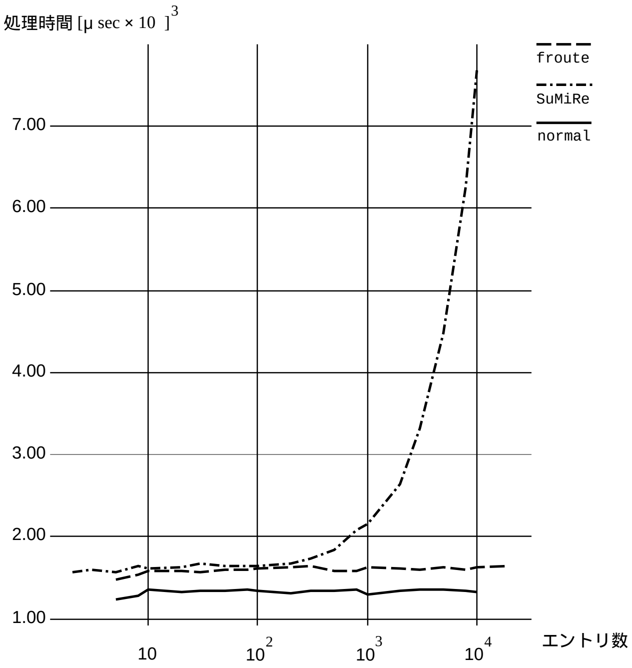

図 5: 経路の増加に対するエントリ追加処理時間

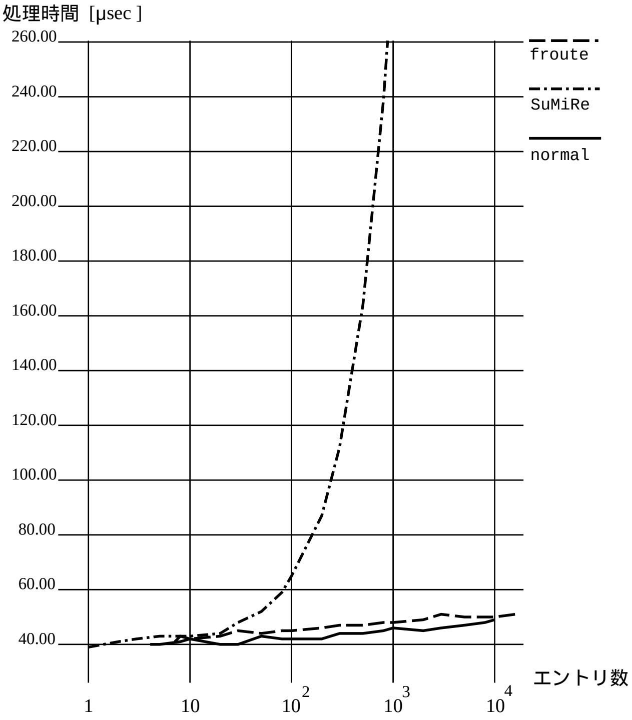

図 6: 経路の増加に対する転送処理時間

時間はエントリ数 (対数軸) に対してほぼ直線であり、その差にはほとんど変化は見られない。これより、froute は既存の機構よりも処理時間はある程度増加するが、エントリ数の増加に対して処理時間はほぼ $O(\log n)$ で変化するため、エントリ数が莫大な数になる場合でも既存の機構とほぼ同等な性能を得られる。

4.2.2 SuMiRe との比較

froute は木構造を利用しているため、エントリ数に対し処理時間は $O(\log n)$ で増加していく。それに対し SuMiRe はフローのエントリを独自の経路リストで管理しており、経路の検索はリストの先頭から順に行うため、エントリ数に対する処理時間は線形増加 $O(n)$ になると考えられる。

実際に図 5、図 6より SuMiRe の処理時間はエントリ数に対しほぼ線形に増加している。この場合エントリ数が増加すると莫大な処理時間がかかってしまい、フローのエントリ数が多くなりすぎると性能が顕著に悪化してしまう。しかし、これは評価の際に追加するエントリをランダムに選択していたためであり、使い方次第では SuMiRe のリスト構造は大きな利点を持つ。重要なエントリ、頻繁に参照するエントリを明示的にリストの先頭近くに配置すればその検索時間は微小で済むため、処理時間は短縮されると考えられる。

また SuMiRe ではフローの経路を追加する際に、その次ホップルータへの経路を経路制御表に作成する。これは、フローの経路を追加した後で経路制御表のエントリが変化した場合に動的に対応できないという欠点もある。

5　今後の課題

本章では、本研究の今後の課題について述べる。

5.1　フロー識別子の各値の順序について

froute で使用しているアドレス構造体 `sockaddr_fl` では、フロー識別子を構成する値の順序は (終点アドレス、始点アドレス、終点ポート番号、始点ポート番号) としている。Radix Tree を利用した構造では、アドレス構造体の先頭に近い値ほど優先されるため、この順序は重要となる。例えば、表 3のような経路制御表を保持する froute ルータが、(終点アドレス、始点アドレス、終点ポート番号、始点ポート番号) = (131.113.82.93, 131.113.82.89, 23, 1023) であるパケットを受信したとすると、マッチするエントリは 6 のエントリではなく、3 のエントリとなる。

ここで、この順序の妥当性について考えてみる。経路制御には終点アドレスは必須であるため、終点アドレスが最優先であることは明らかである。また、始点ポート番号には終点ポート番号のようにウェルノウンポートが存在せず、始点ホスト上のプロセスの識別子としての役割しか持たないため、フロー識別子の中では最も重要度は低いと考えられる。問題は始点アドレスと終点ポート番号のどちらがより重要かという議論であり、これを解決するには実際にバックボーンルータが処理するパケットを調べる必要がある。

5.2　IPv6 への対応について

今回の froute の実装は IPv4 を前提としていたが、現在次世代インターネットプロトコルとして IPv6[2][3] が注目されており、IPv6 に関して様々な研究が行われている。froute も今後、IPv6 に対応していくことが必要であると思われる。IPv6 の特徴として、アドレス長が 32 ビットから 128 ビットに変更されたことが挙げられるが、Radix Tree 構造は可変長アドレスに対応しているため、このアドレス長の変化には容易に対応が可能である。また、フロー識別子として IPv4 では (終点アドレス、始点アドレス、終点ポート番号、始点ポート番号) の 4 つの値を用いたが、IPv6 の場合は IP ヘッダにフローラベルという 24 ビットの識別子を保持するため、フロー識別子は (終点アドレス、始点アドレス、フローラベル) の 3 つの値となる。

さらに、IPv4 の実装では TCP/UDP のポート番号をインターネット層から参照しプロトコル階層を侵すことをしているが、IPv6 でフローラベルを利用すれば実装はインターネット層内だけで良く、プロトコル階層を侵すこともなくなる。

6　関連研究

本章では、froute の関連研究として次の 3 つについて説明する。

- Flow Labelled IP: A Connectionless Approach to ATM[4]
- ソース IP アドレスを考慮した経路制御システムの提案 [8]
- SuMiRe[7]

6.1　Flow Labelled IP: A Connectionless Approach to ATM

[4] は、IP over ATM の研究の一つである。従来の IP over ATM では、IP の特徴であるステートレスや柔軟性、頑強性と ATM の特徴であるコネクション指向、スイッチングが噛み合わず、IP の特性が失われていた。[4] では ATM のコネクション指向の特性を捨て、ATM スイッチ上に IP ルータを実装し、コネクションレスのネットワークにおいて高速な回線として ATM を利用する方法を提言している。それに加えて、[4] では ATM のハードウェアスイッチングを利用する。パケット一つ一つを IP 層まで処理しソフトウェアで経路制御するのではなく、ハードウェアでテーブルを保持しスイッチングを行うことにより高い速度性能を得る。この方法を実装した製品は IP Switch と呼ばれている。

IP Switch ではハードウェアスイッチングの高速性を利用する際、フローの概念を利用している。通信時間が長くなると予想されるフローの場合は、スイッチのテーブルにエントリを保持しハードウェアスイッチングを行う。通信がごく僅かなフローはスイッチに状態を保持せず、プロセッサに処理が渡される。各セルの属するフローを識別するために、IP Switch では TCP/UDP のポート番号を利用している。

IP Switch は froute と同様にフローの概念を利用しているが、その目的は全く異なる。froute ではフロー毎に経路制御を行うためにフローを利用しているが、IP Switch ではハードウェアスイッチングかソフトウェアスイッチングかを選択する基準としてフローを用いている。さらにフローを考慮する層は、froute では IP 層、IP Switch ではネットワークインタフェース層である。IP Switch は IP 層では通常の経路制御を行っており、フローを考慮した経路制御は不可能である。

6.2 ソース IP アドレスを考慮した経路制御システムの提案

地方の ISP(Internet Service Provider) が相互接続するポイント、地域 IX(Internet eXchange) を構築しようとする動きが近年高まっている。地域 IX は大手 ISP と地域 ISP に接続しており、地域 ISP は地域 IX を通して大手 ISP と接続している。ここで、大手 ISP は各自が契約している地域 ISP のパケットのみを転送したい。しかし従来の終点アドレスのみによる経路制御では、パケットを送信した地域 ISP を識別することはできず、各地域 ISP から送信されたパケットを同等に扱ってしまう。

これを解決するために [8] では、終点アドレスのみでなく始点アドレスも考慮した経路制御機構 STAR を提案している。STAR では各地域 ISP 毎の経路制御表を保持しており、パケットの始点アドレスによって対応する経路制御表を選択する。

STAR は終点アドレスの他に、始点アドレスも考慮した経路制御機構である。STAR では始点アドレス毎に経路制御表を保持している。

STAR はポート番号を考慮することはできないが、ポート番号は考えず終点アドレスと始点アドレスのみを考慮した経路制御として考えると、froute では一つの Radix Tree を用い、常に 96 ビットのアドレスを検索するが、STAR のように先に始点アドレスのみで複数の Radix Tree から一つを選択し、それから通常の 32 ビットの検索を行う方が処理が高速な場合もあり、froute より STAR の方が有効になり得る。

6.3 SuMiRe

[7] は NAT、packet filtering の機能を持ち、froute と同様、(終点アドレス、始点アドレス、終点ポート番号、始点ポート番号) の値の組で経路制御を行う機能を提供している。

SuMiRe は経路制御表には変更を加えず、フロー識別子の経路リストを別に保持している。フローのエントリを追加する際にはその次ホップの経路を通常の経路制御表内に作成し、それをフローの経路リストから参照する。パケットの転送時には経路リストを先頭から線形探索し、マッチするエントリがあればその経路にパケットを転送する。マッチするエントリが無ければ、従来の経路制御表にて検索を行う。

7 まとめ

本研究ではフローを考慮した経路制御機構として、フロー識別子 (終点アドレス、始点アドレス、終点ポート番号、始点ポート番号) を基に経路制御を行う機構 froute を提案した。

froute の設計は既存の経路制御表に拡張を加えるものとし、既存の経路制御表に用いられている Radix Tree 構造を利用した。その際 Radix Tree 構造の特徴を利用し、既存の経路制御表と透過的に扱える経路制御表を実現した。

froute の関連研究として IP Switch、STAR、SuMiRe が挙げられる。それぞれ独自の機能、設計をしているが、検討の結果、フローを考慮した経路制御として考えると froute は十分利用価値があると考えられる。

今後の課題として、フロー識別子の各値の優先順位を実際のルータで処理されるパケットを調べることにより定量的に妥当なものとすること、さらには IPv6 への対応などが挙げられる。

参考文献

[1] D. Comer. *Internetworking with TCP/IP vol.I: Principles, Protocols, and Architecture 3rd Ed.* Printice Hall, 1995.

[2] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. Internet RFC1883, *IETF*, Dec. 1995.

[3] C. Huitema. *IPv6: The New Internet Protocol.* Prentice Hall, 1996.

[4] P. Newman, T. Lyon, and G. Minshall. Flow Labelled IP: A Connectionless Approach to ATM. In *Proceedings of IEEE Infocom '96*, Mar. 1996.

[5] K. Sklower. A Tree-based Packet Routing Table for Berkeley UNIX. In *Proceedings of USENIX 1991 January*, 1991.

[6] G. Wright and W. Stevens. *TCP/IP Illustrated, Volume 2 The Implementation.* Addison-Wesley, 1995.

[7] 藤澤 慎一. SuMiRe-0.3, 1998. ftp://garuda.yokogawa.co.jp/.

[8] 鍛冶武志, 小林和真, 山口英, 尾家祐二. ソース IP アドレスを考慮した経路制御システムの提案 地域 IX における経路制御問題の解決. 情報処理学会分散システム運用技術研究会, 1997.