マルチパスネットワークを持つPCクラスタにおける動的経路制御システム

三浦信一[†] 岡本高幸[†]
朴泰祐[†,††] 塙敏博[††]

VFREC-Net は VLAN 技術を用いることで，安価な Layer-2 の Ethernet スイッチのみで高性能なマルチパスネットワークを構成できるクラスタ向けネットワークである．しかし既存の VFREC-Net では，それらのマルチパスを制御するルーティングテーブルが静的に決定されていたため，アプリケーションの通信パターンによっては用意した複数の経路にトラフィックが均等に分散されず偏りが生じ，結果的に通信の衝突による性能低下を引き起こす．この問題を解決するために，アプリケーションレベルでこのルーティングテーブルを動的に変更可能にするフレームワークを開発した．このフレームワークを用いることで，既存の動的通信最適化アルゴリズムが適用可能になり，結果として VFREC-Net がより理想的な形で運用可能になる．これに加えて本論文では MPI プログラム上から直接ルーティングテーブルを変更可能にするためのインタフェースを開発した．ネットワーク経路上の通信衝突が問題になる NPB Kernel-CG において本システムを適用し，最適なルーティングテーブルになるようにアプリケーション中から動的に設定することで，ルーティングテーブルが静的な場合と比較してより高い性能を示すことが確認された．

A Dynamic Route Control System for PC Clusters with Multi-path Network

Shin'ichi Miura,[†] Takayuki Okamoto,[†] Taisuke Boku[†,††] and Toshihiro Hanawa[††]

VFREC-Net is a network construction technology which allows to configure a multi-path network routing on inexpensive Layer-2 Ethernet switches based on tagged-VLAN technology. Current VFREC-Net system implies a problem on traffic balancing when the communication pattern of the application does not fit the network topology, due to its static routing scheme. We have developed a framework to solve this problem by allowing dynamic routing table rewriting from the application level. When an appropriate algorithm to optimize the VLAN-id allocation according to the application's behavior, it enables to balance the traffic on Fat-Tree topology with user-level controlling of VLAN. We also provide an API library for MPI programming to use above framework, and confirmed its effectiveness through the communication optimization on NPB Kernel-CG benchmark.

1. はじめに

一般の PC クラスタの多くは，ノード間を接続するネットワークとして Ethernet を採用している．特に Gigabit Ethernet（以後，GbE）は，そのコストパフォーマンスの高さから多くのクラスタ環境で使用されている．一般的にコストパフォーマンスの良い Layer-2 GbE スイッチは，24 ポート程度の比較的小規模なものに限られる．そのため，これを上回る規模のクラスタでは，複数台のスイッチを Tree 構造等で相互に結合する必要がある．クラスタの性能をノード数に合わせて向上させるためには，このスイッチ間のバンド幅もあわせて増強する必要があるが，Ethernet ではその性質上ネットワーク上にループ構造を作ることができず，スイッチ間はただ 1 つのパスで結ばなければならない．例外として LACP [1] 等の trunk 技術による複数パスの利用が考えられるが，トポロジの制限（2 台のスイッチ間での平行結線のみ）が存在する．そのため，Ethernet を用いて大規模な HPC クラスタを構築するには，スイッチ間のバンド幅が問題にな

† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†† 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba

る．この問題を解決する 1 つの方法として VLAN ルーティング法[2] が提案されている．これを用いることで，Ethernet を用いた場合でもクラスタネットワークに存在したトポロジの制限が緩和され，より柔軟なネットワークを構築できる．その結果，単純な Tree 構造のネットワークに比べて，高いネットワークバンド幅をユーザに提供することが可能になり，典型的な HPC ベンチマークにおいて高い実行性能を得ることができる．我々はこの VLAN ルーティング法を発展・拡張させた，VFREC-Net を開発している[3]．VFREC-Net は VLAN ルーティング法で得られる機能をよりシステムレベル側で実現することで，本機能をユーザへ容易に提供するとともに，より高いクラスタシステムの拡張性を提供する．しかし既存の VFREC-Net では VLAN ルーティング法で得られる複数の通信経路に対して，それぞれのノードが使用する経路を一意に固定しているため，一部のアプリケーションで十分な性能が得られないことが明らかになっている．そこで本論文では，問題となる通信経路の管理方法を改良し，動的に変更可能にするフレームワークを提案・実装し評価する．

2. VFREC-Net

本論文で述べる VFREC-Net の概要については，すでに文献 3) で述べているが，ここでは本論文の内容を理解するために要点のみを述べる．

2.1 VFREC-Net の概要

VLAN ルーティング法[2] は，tagged-VLAN（IEEE 802.1Q）[4] を用い，各ノードから送信するパケットの VLAN ID（以後，VID）を制御することで，Ethernet を用いた場合でもさまざまなトポロジを構築可能にする．上位スイッチへの接続にそれぞれ異なる VLAN を割り当てることで，物理的にループのあるネットワーク構成を，論理的にループのない複数のネットワークに展開する．これらのネットワークをノード側から明示的に使い分けることで，Ethernet のスイッチ間バンド幅ボトルネックを解決する．文献 2) で提案されている VLAN ルーティング法を既存の Linux 環境で実現するためには，それぞれの VLAN に対して関連付けられた仮想ネットワークデバイスを用意し，それらに異なる IP アドレスとサブネットを設定する．通信経路の決定は，使用するサブネットを変更することで実現できる．しかしながら，その方法では本来ノードを特定するための IP アドレスが，経路を決定するために使用される．そのため，個々のノードを特定するために別の何らかの識別子が必要になり，既存のソケットアプリケーションがそのままでは利用できないという問題がある．

我々の開発している VFREC-Net（VLAN-based Flexible, Reliable and Expandable Commodity Network）[3] は，この VLAN ルーティング法を改良し，システムレベル（デバイスドライバ）で実装したうえで，より PC クラスタ向けに利用しやすくしたものである．VFREC-Net は独自のドライバ（VFN ドライバ）を利用することで，ただ 1 つの仮想デバイスと，それに割り当てられた唯一の IP アドレスで複数の経路（VLAN）を利用できる．経路の選択には，IP よりも下位の通信層である MAC アドレスを用いている（2.3 節参照）．すなわち，既存の VLAN ルーティング法と異なり，個々のノードの識別には IP アドレスをそのまま利用でき，通常のソケット API を用いたすべてのプログラムがいっさいの変更なしで使用可能になる．その結果，Open MPI[5] 等の並列プログラミング環境に対してもそのまま適用可能になっている．

VFREC-Net の中核となる VFN ドライバは，デバイスドライバとして Linux カーネルモジュールの形で実装している．そのデバイスドライバは OS に対して仮想的な Ethernet デバイスとして振る舞う．デバイスドライバ中では OS から呼ばれる送信処理ハンドラを用意し，ハンドラ内部で利用できる複数の VID から適切な VID をパケットに設定した後に，実際に送信処理に用いるネットワークデバイスを管理するデバイスドライバ中の送信ハンドラを用いて送信する．受信処理では，実際に受信処理を行うネットワークデバイスのドライバに対して，関連付けられた VID の付いたパケットをあたかも VFN デバイスで受信したように設定する．これらの処理は，すべて Ethernet デバイスドライバに用意されている標準的なインタフェースを用いて行うため，VFN ドライバは，実際に用いる NIC のハードウェアやそのドライバに依存していない．

2.2 VFREC-Net で想定するネットワーク

大規模なクラスタを構築する場合，スイッチどうしを Tree 構造等で相互に結合しなくてはならない．しかし，スイッチ間を流れるデータトラフィックが性能ボトルネックを引き起こすため，クラスタの性能をノード数に合わせて向上させるためには，その部分のバンド幅をできる限り増強する必要がある．そこで Myrinet[6] や InfiniBand[7] 等の SAN（System Area Network）で Tree 構造のネットワークを構築する場合，**図 1** (a) に示す，Fat-Tree と呼ばれるトポロジ構成を用いてきた．この構造のネットワークは，上段へ

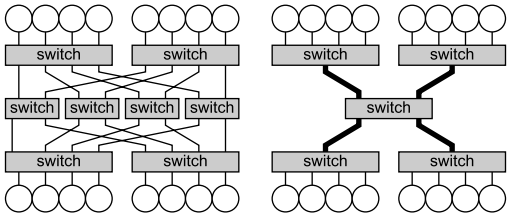


図 1 Fat-Tree トポロジの構成例
Fig. 1 Example of Fat-Tree topology.

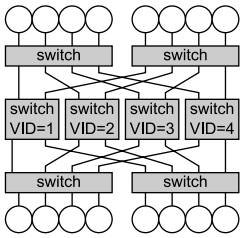


図 2 VLAN-based Fat-Tree トポロジの構成例
Fig. 2 Example of VLAN-based Fat-Tree topology.

の接続を複数のスイッチに分散させることで，下段のスイッチからのトラフィックを上段のスイッチへ均等に分散させることができる．このようにして構築する Fat-Tree は，多くの HPC アプリケーションおよびランダムトラフィックに有効である．

一方，Ethernet では複数経路を持つネットワークを構成できない．そのため，Fat-Tree 構成をとる場合，上段のスイッチへの接続を下段への接続に対して高速な接続とする図 1 (b) に示すような構成をとる．この構成は，図 1 (a) に示すネットワークと同等になるが，このような接続はクラスタの規模にともなって Tree の構造が大きくなることで，上段への接続により高速な接続が必要になる．そのため，このようなスイッチ間の接続は拡張性が低くなり，Ethernet を用いたシステムでは，クラスタの大規模化という点で大きな問題があった．

しかし，VLAN ルーティング法[2] を用いることで，Ethernet を用いた場合でも複数の経路を持つネットワークを構成可能になる．そこで，VLAN ルーティング法では，Fat-Tree 形のトポロジである VLAN-based Fat-Tree（以後，VBFT）を提案している．VBFT 構成のネットワークの例を**図 2** に示す．VBFT は，Tree 構造の上位階層のバンド幅を物理的に増強するとともに，それらの複数経路を VLAN を用いることで論理的に単一の経路に限定する．そのうえで，これらの経路を各ノードが均等に使い分けることで，上段スイッチへのトラフィックを分散させ，全体のスループットを向上させる．使用する VID の数は VBFT の最上層のスイッチ数となり，たとえば図 2 で示したネットワークの場合は 4 個となる．各ノードごとに VID を割り当てるわけではないので，管理する VID 数が多くなることはない．それぞれの VID ではそれらの最上層のスイッチを頂点とする Tree ネットワークとなる．各ノードはそれぞれどの Tree ネットワークを使用するのかを，送信時の VID を変更することで選択する．各ノードが送信に用いる VID を分散して使用することで，ネットワークのトラフィックを個々の Tree ネットワークに分散させ，高いバイセクションバンド幅を得ることができる．

VFREC-Net ではこの VBFT を基本的なネットワークトポロジとして実装および評価している．

2.3 VFREC-Net でのルーティング方法

VLAN ルーティング法では，送出するパケットに対する VID を決定することで，ネットワーク経路が決定する．用意した複数のスイッチ間の複数経路を偏りなく利用するためには，使用する VID がなるべく均等に利用される必要がある．VFREC-Net では，使用する経路を決定する際の指標としてパケットに含まれる送信先 MAC アドレスを使用する．この送信先 MAC アドレスとそのとき用いる VID の組合せのテーブルが，その送受信ノード間での通信経路を決定する．このため，このテーブルを以後ルーティングテーブルと呼ぶ．ルーティングテーブルはドライバの初期化時にドライバ内に設定され，その後の通信ではこの固定されたルーティングテーブルを使用する．

このルーティングテーブルは Ethernet スイッチの MAC アドレス学習アルゴリズムを考慮した設定が必要となる．一般的な IEEE 802.1Q 対応の Layer-2 スイッチの MAC アドレス学習は，VID ごとに独立して行われ，異なる VID 間ではこの情報は共有されない．そのため一対のノード間の通信において，双方のパケットが異なる VID を設定した場合，中継するスイッチは相手ノードの MAC アドレスとそこに到達するための経路（ポート）の関係をいつまでも学習できない（**図 3**）．その結果，スイッチはいつまでもパケットをフラッディング（ブロードキャスト）送信する．フラッディング送信はパケットを間違いなく送信先ノードに配送することを目的とするが，これが長期に繰り返された場合，不要なパケットがネットワークにブロードキャストされクラスタシステム全体の性能低下につながる．したがって，あるノード間の通信では，どちらが送信/受信側であるかにかかわらず，同じ VID を使用するように努めなくてはならない．すな

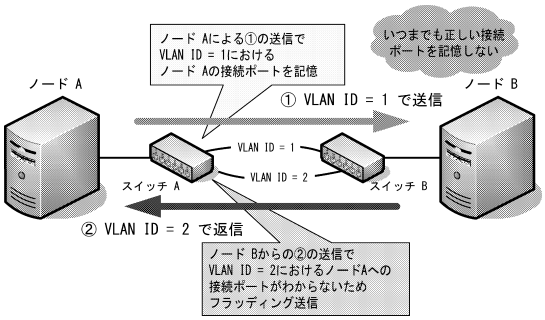


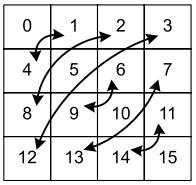
図 3 VLAN ルーティング法を用いた場合の Layer-2 スイッチの MAC アドレス学習アルゴリズムの問題点
Fig. 3 Problem on MAC address learning mechanism on Layer-2 switches for VLAN routing method.

わちこれは Source-Destination ルーティングとなる．

すべてのノードが用意された経路を分散して使うために，クラスタ全体では全ノード対に関する VID をルーティングテーブルとして持つ必要がある．しかしノード数を n とした場合，この組合せは $O(n^2)$ となってしまうため，クラスタの規模が大きくなるに従って，このルーティングテーブルの設定ファイルは大きく，かつ複雑となる．この問題を解決するために，VFREC-Net のルーティングテーブルを決定する指標として，各ノードにデフォルト VID と優先度を与える．実際に経路を決定する場合は，通信するノード対の優先度を比較し，優先度が高い方のノードに設定されているデフォルト VID を使用して通信を行う．ネットワークの接続性を確保するため，記述がないノードと通信する場合はデフォルト VID を用いて通信を行う．具体的な設定ファイルには，各ノードの MAC アドレスとその場合に用いる VID，そして優先度を記述する．このような仕組みを利用することで設定ファイルサイズは $O(n)$ となり，設定が容易になる．初期設定に用いる優先度は『ノード番号（MPI ランク番号）0〜$(n-1)$ の低い方を優先する』という単純なものである．これにより，n^2 通りの送受信ノードの組合せにおいて，n の記述で全通信で使用する VID を設定できる．これに加えて，各ノードのデフォルト VID を使用する VID の範囲でサイクリックに設定することで，結果的に使用する VID を全ノードへ均等に設定できる．この方法は多くの場合において有効に機能し，少ない設定ファイルの記述で大きな効果を得られた[3]．

2.4 既存の VFREC-Net における問題

一部のベンチマークにおいて，VFREC-Net が標準で与えるルーティングテーブルでは通信パケットが特定の経路に偏るという問題があった．以下では実



組合せ	最適化前	最適化後
1-4	2	2
2-8	3	3
3-12	4	4
6-9	3	4
7-13	4	3
11-14	4	2

図 4 NPB Kernel-CG における問題となる通信の VID 割当て
Fig. 4 VID mapping on NPB Kernel-CG.

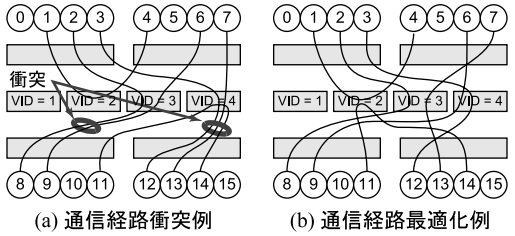


図 5 NPB Kernel-CG における通信経路での通信衝突例と最適化例
Fig. 5 Example of communication routes congestions and optimizations on NPB Kernel-CG.

例として NAS Parallel Benchmarks（以後，NPB）Kernel-CG を取り上げ，VFREC-Net の開発中に解析した VLAN によるルーティングとトラフィックの偏り問題を述べる[3]．

図 2 に示すようなノード数 16，上位層のスイッチを 4 台持つ 2 段の VBFT 構成で Kernel-CG 中で発生する通信を考える．図中の数字は各ノードのランク番号を示している．Fat-Tree を構成する最上段のスイッチ数は 4 としたため，経路を制御する VID の数は 4 となる．各ノードに 1〜4 までの VID をノード番号の若い順にサイクリックに割り当て，ノード番号が若いノードほど優先度が高くなるように設定する．

Kernel-CG における行列のランク番号へのマッピングと問題となる通信のパターンを**図 4** に示す．それらの通信がルーティングテーブル中で，どのような VID に設定されているかを同時に図 4 の表中の『最適化前』に示している．ここで示した組合せの通信がベンチマーク中の通信で大部分を占めている．これらが実際にどのような通信経路になるのかを**図 5** (a) に示す．図 5 (a) 中の丸で囲んだ部分が示すように，ノード 2-8 間と 6-9 間の通信が，またノード 3-12 間，7-13 間および 11-14 間の通信がスイッチ間経路で衝突している．これらは，VBFT によって用意された複数の経路が有効に活用されていないことを示し，本来得られるべきネットワークバンド幅増大の効果が性能向上に結び付かない．実際のベンチマーク結果においても，Kernel-CG では VBFT における速度向上が不十

分であった．これをふまえたうえで，図 4 の『最適化後』に示すように，使用する経路で衝突しないよう，Kernel-CG の通信パターンに最適化したルーティングテーブルを設定したうえで評価したところ，性能が大きく改善できることを確認している．この最適化した通信経路を図 5 (b) に示す．この場合ではスイッチ間経路で衝突する通信がない．

ここで重要なのは，ルーティングテーブル（VID）の決定方法である．最適化前後で 4 つの VID のうち 3 つしか利用していない点に変わりはなく，VID の均等利用という意味ではどちらも等価であり負荷の分散がうまくいっているように見える．しかし実際の経路利用状況を見てみると，最適化後では通信経路の衝突が回避されている．VBFT の構成で得られる大きなバイセクションバンド幅を効率良く使うためには，使用する VID を均等に割り振るだけでなく，通信パターンを注意深く観察しスイッチ間の経路で通信が衝突しないように VID を設定する必要がある．

3. 通信最適化手法の検討

3.1 通信最適化のアプローチ

これまでの VFREC-Net で評価してきた VBFT は，アプリケーション上の通信トラフィックを Fat-Tree 網の上位階層にフラットに分散させることを目的に実装されてきた．しかし前章で示したように，ある種のアプリケーションにおいては，その通信パターンが VBFT の構成と適合しないことにより，通信トラフィックが衝突し，本来 VBFT が持つバイセクションバンド幅増強の効果が十分発揮されない場合がある．しかし，既存のルーティングテーブル決定方法では，複数の通信パターンに対応することが難しい．ここで我々がとるべき方法がいくつかあげられる．

(1) 通信の局所性を高め，なるべく通信が同一のスイッチ内部で収まるようにアプリケーションをプログラムする．
(2) 初期設定のルーティングテーブルを最適になるように工夫する．
(3) 初期設定のルーティングテーブルを用いつつ状況に応じて適宜ルーティングテーブルを更新し，通信量が多いノード間の通信がなるべく同一のパスを通らないようにする．

我々のターゲットとする HPC アプリケーションでは，(1) のように通信パターンを考えたプログラミングとそれらのプロセスの割当てが最適になることを前提としている．しかし，計算のフェーズごとに大きく通信パターンが変わるようなアプリケーションでは，それらを同時に実現することが難しい．一方 (2) の手法では，アプリケーションごとに最適なルーティングテーブルが異なる場合，すべての通信パターンに対して最適なルーティングテーブルは存在しない可能性がある．よって我々は (3) の手法を実現し，アプリケーションごとに，あるいはアプリケーション内での処理フェーズごとに，ノード対が使用する VID の割当てを動的に変更することでこの問題に対応する．この場合，VBFT の構造やネットワーク中のスイッチの設定をいっさい変更せず，各ノードで送信に使用する VID を変更することで物理ネットワーク上での通信衝突を回避するというものであり，その可能性は前章で述べた解析により実証された．そこで本論文ではアプリケーション実行中に動的に VID の割当てを変更し，ネットワーク上の経路制御を積極的に行う．それによって通信パターンに応じた最良のルーティングテーブルを設定可能にする．

これに代わる手段として，Layer-3 スイッチを利用し VBFT のようなネットワーク構造を作り，それらのスイッチのルーティングテーブルを変更することで，ネットワーク内の経路制御を行うこともできる．しかし，本システムと比較して Layer-3 スイッチのためコストが高く，遅延時間オーバヘッドも大きくなる．また，Layer-3 スイッチに対してルーティングテーブルを変更するための処理が必要になる．VFREC-Net では，スイッチ自体の設定変更は必要なく，送受信ノード内での VID の変更のみで事実上のルーティング変更が可能である．これは，ハードウェアとソフトウェアの両者にまたがったルーティング方法という，VLAN ルーティングならではの特徴を利用するもので，小さなオーバヘッドで大きな効果を生むことが期待できる．

これらの動的ルーティング変更手法には，『そもそもどのように最適な経路分布を発見するのか』という，アルゴリズム的な問題が存在する．最適通信経路分布問題は，VLAN ルーティング法に限らずネットワークでは一般的な問題であり，本論文はこれに対する回答を与えることは目的としない．その代わり，最適な VID の分布が見つかった場合に，これを簡単にアプリケーションにおいて適用可能とするシステムの枠組みを提供する．具体的には，これまで VFREC-Net システムにおいて初期化ファイルによって固定テーブルとして提供されていたルーティング情報を，アプリケーションレベルから動的に変更可能にするシステムを構築する．経路最適化問題は一般的な問題であると考え，どのような最適化アルゴリズムが想定されてもこれを適用可能なフレームワークを提供することが本論文の

目的である．

3.2 ルーティングテーブル変更手法

通信パターンに応じてルーティングテーブルを変更する際，VFREC-Net システムがどのようにこれを決定するかについて，以下の 2 通りの方法がありうる．

- ユーザからの指示によるコントロール
  ユーザプログラム自身からの明示的な指示によってルーティングテーブルの変更を行う．多くの場合で，ユーザは自身のプログラムの通信パターンを把握可能であり，それをネットワークトポロジと比較することで最良なルーティングテーブルを設定できる．またプログラム中で通信パターンが大きく変わる場合には，その時点で最適なルーティングテーブルに変更することもできる．一方で，ユーザはネットワークトポロジと VID の関係を把握する必要があり，またプログラムの可搬性は小さくなる．
- 自律的なシステムによるコントロール
  ユーザの特別な指示なしに，システムが自律的に状況に適したルーティングテーブルを導きだし，その結果を用いてルーティングテーブルを設定する．ネットワークの全トラフィックを監視できる管理サーバを用意し，その管理サーバが特定の評価関数に基づいてアダプティブなルーティングテーブルを求め，すべてのノードがその指示に従ってルーティングテーブルを変更する．この管理サーバがすべてのノードから通信量を取得しそれらを合成することでどの経路が最も混雑しているかを求める．これにより，どのようなアプリケーションを動作させた場合においても，それに応じたアダプティブなルーティングが設定され，ユーザはそれに対して，特別な意識をする必要がなくなる．一方で，ルーティングテーブルの変更タイミング等によって，最適化されるまでにある程度の時間が必要であり，また適切な評価関数が設定されていなければ，さらに良いルーティングテーブルがあっても最適解に近づかない場合も考えられる．

最終的にこれら 2 つの欠点を補うために，2 つの手法を適宜選択して使用することを目標とする．本論文ではこれらを実現するためのフレームワークを開発し，そのうえでユーザ指示によるコントロール手法を実装・評価する．

4. 実　　装

フレームワークのイメージを**図 6** に示す．今回開発したものは，ルーティングテーブルを管理するシステム **VFN Network Manager**，およびユーザ指示によって動的にルーティングテーブルを変更する **MPI 用制御ライブラリ**である．次に，これら 2 つの実装と，ルーティングテーブルの動的変更による送受信処理への影響について述べる．

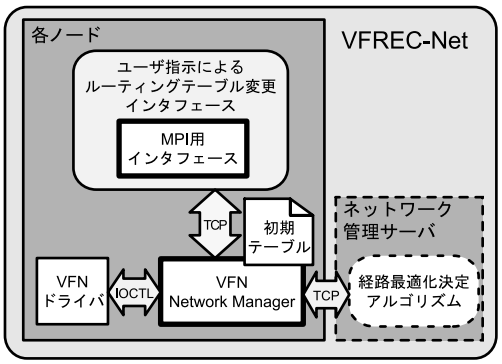


図 6 動的ルーティングテーブル変更システムを含む VFREC-Net システムのフレームワーク
Fig. 6 Framework of enhanced VFREC-Net system with dynamic routing table modification.

4.1 VFN Network Manager

ルーティングテーブルを動的に変更する機能を実現するために，各ノードで新たに VFN Network Manager（以後，VFN-NM）を動作させる．VFN-NM の最も大きな役割は各ノードで動作しているデバイスドライバ（VFN ドライバ）を管理することである．VFN ドライバは VFREC-Net の機能を提供するための核となるデバイスドライバであり，OS やユーザからは，通常の Ethernet デバイスドライバとして見える．そのドライバ内部のルーティングテーブルの変更には *ioctl( )* を通じた特別な操作が必要になるため，VFN-NM がこれらの処理を代表して行う．VFN-NM は，以下に示す機能を提供する．

(1) 初期ルーティングテーブルの設定
(2) 動的なルーティングテーブルの変更
(3) 各ノードへの送信量の収集
(4) Network Interface の各種情報収集

今回はこれらの VFN-NM の機能を実装するために，既存の VFN ドライバに各 VID でどの程度の送信があったのかを確認するためのログの収集機能，ドライバ内部のルーティングテーブルのセット/リセット機能を追加した．

VFN-NM は TCP を用いて各種ライブラリや外部のシステム管理サーバ等からの接続を受け付ける．そして，それらのシステムが要求する処理（たとえばルーティングテーブルの変更等）を実行する．これらの機能を利用することで，ユーザアプリケーションレ

ベルで動的なルーティングテーブルの変更を可能とする．また今後この機能を用いたうえで，新たなシステムを構築することでアダプティブなルーティングテーブルを設定することができる．

4.2 MPI 環境における動的ルーティングテーブル制御ライブラリ

本論文ではプログラミング環境として MPI を用いることを前提とし，MPI アプリケーション上からルーティングテーブルを変更するインタフェースを提供する．ユーザが通信パターンを把握していることを前提とし，与えられたネットワークトポロジから特定の経路で通信が集中しないように，各ランク番号の対ごとに使用する通信経路すなわち VID を決定する．この環境をユーザに提供する以下の 3 つの関数を用意した．

- *VFN_InitMPI( )*
  ユーザはランク番号の対から使用する通信経路（VID）を決定する．この関数は必要なルーティングテーブルの初期化処理を行う．MPI が動作していることを前提にしているため，ライブラリ内部の通信は MPI を介して行う．なおこの段階のドライバ内のルーティングテーブルには，あらかじめ用意された標準のルーティングが設定される．
- *VFN_FinalizeMPI( )*
  この関数が呼び出されると，所持しているランク番号と MAC アドレスのテーブルが即座に開放され，また，プログラム中で変更されたルーティングテーブルを初期状態にもどす．
- *VFN_SetRouteMPI(rank1, rank2, vid)*
  ユーザの指示によりローカルのルーティングテーブルに設定されている VID を *vid* に変更する．*rank1* および *rank2* は通信を行う 2 ノードのランク番号であり，これらの 2 ノード間では必ず同じ VID を用いるため，source，destination の区別はされない．初期化関数によって作成されたテーブルを基に，MPI のランク番号を実際の MAC アドレスに変換し，ローカルホストで動作している VFN-NM にドライバ内のルーティングテーブル変更要求を行う．ルーティングテーブルの設定は，この関数が呼ばれたのちに即座に行われる．この関数は *VFN_InitMPI( )* と *VFN_FinalizeMPI( )* の間でのみ動作する．各々のノードで動作する VFN-NM では，そのノードが通信に関わるルーティング情報のみをテーブル上に設定する．

これらの関数は各ノードのローカルで動作している VFN ドライバのルーティングテーブルを変更するだけなので，本来は VFN-NM を介さなくても直接ルーティングテーブルの変更ができる．しかし今回の実装では，これらはローカルで動作する VFN-NM に対して TCP/IP を用いて通信することで目的の処理を行う．テーブルの変更要求等には VFN-NM へ通信というオーバヘッドが発生するが，本フレームワークはアプリケーションの実行環境が MPI だけでなく，より広範囲な実行環境を想定している．特に自律的制御を考えた場合では，外部からのリクエストの処理も想定しなくてはならない．それらに矛盾ない対応をすることが必要になるため，我々はこのような VFN-NM を介した制御を行うことにする．この実装を拡張すれば将来的に VFN-NM どうしの通信や，それによる自立制御も理論的に可能である．

ルーティングテーブルの変更頻度は，1 度の変更に要する時間，また 4.3 節で述べるフラッディング問題を考慮し，プログラム単位，もしくはループ構造を持つアプリケーション内でのイタレーション単位とする．パケット単位でルーティングテーブルを変更することは，変更で発生するオーバヘッドを考慮し想定しない．ある程度大きな粒度でルーティングテーブルを変更することで，テーブル変更のコストと，フラッディングによるネットワーク全体への負荷を軽減させる．このコストについては，5.1 節において評価する．

4.3 ルーティングテーブル変更による送受信処理への影響

VFREC-Net ではあらかじめ通信で使用されるすべての VID でパケットを受信可能にしている．すなわち送信側がどの VID を用いて送信してきた場合でも，受信側では内部のルーティングテーブルがどのような設定になっているかにかかわらずパケットを受信する．これは，送受信の双方のルーティングテーブルが異なる過渡状態においても正常に通信を成立させるためである．送信側では受信先のルーティングテーブルが変更されたかどうかを意識することなく送信処理を行うことができる．受信先がこれらのパケットをすべて受信することで，無駄な再送を抑えることができる．

テーブルの変更タイミングとネットワークの負荷バランスによってはパケットの追い越しが発生する可能性が考えられる．ただし，本システムに限らず Ethernet ではパケットの到着順序は保証されていない．本来，これらの問題はデータリンク層（Ethernet）よりも上位の層で保証されるべきものである．そのため一般のシステムの多くでは，到着順序の保証に上位層のプロトコルである TCP/IP を用いる．複数の経路を用いる他のシステム（たとえば Ethernet の Link Aggregation 技術である IEEE 802.3ad）でも同様で

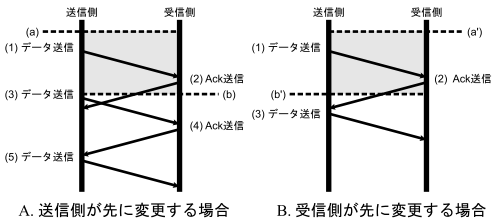


図 7 MAC アドレス学習までの流れ
Fig. 7 A flow of the MAC address learning.

ある．VFREC-Net でも，この問題を上位のプロトコル（TCP/IP 等）の順序制御によって解決する．

これらをふまえて**図 7** に基づき，送信側および受信側のルーティングテーブルが変更されるタイミングによりどのような動作になるのかを考察する．上位の通信プロトコルとして TCP/IP を利用することを想定して説明するが，TCP の細かい通信制御については説明の簡略のため割愛する．

A. 送信側が先の場合 送信側が (a) のタイミングでルーティングテーブルを書き換えると仮定する．灰色に示す部分が 2 つのノード間でルーティングテーブルが異なっている状態を示している．まず送信側が（1）において新しい VID を用いてパケットの送信を開始する．この VID に対しては受信側の経路をどのスイッチも学習していないため，パケットはフラッディングされる．このとき，この新しい VID での送信側への経路が各スイッチで学習される．このパケットはフラッディングされるため，他のノード間通信を阻害する恐れがある．前述のように受信側では，送信側がどの VID を用いて送信してきたかにかかわらずフレームを受信する．受信側ではルーティングテーブルは書き換わっていないため過去の VID で Ack を送信する（2）．Ack パケットに設定されている VID は，過去の通信で学習済みのため，正しい経路で送信側に到達する．送信側も受信側と同様にすべての VID のパケットを受信できるため，この Ack パケットを受信することができる．送信側は受信側のルーティングテーブルの状態を気にせずにパケットを送信し続ける（3）．タイミング (b) で受信側のルーティングテーブルが書き換わると，以後の Ack パケットは送信側の VID と同じ VID で送信されるようになる（4）．この Ack パケットによって新しい VID での受信側の経路をスイッチが学習する．すでに送信側の経路は（1）の通信によって学習されているので，以後このノード間の通信ではフラッディングは発生せず，正しい経路のみを用いて通信が行われる．

B. 受信側が先の場合 受信側が (a') のタイミングでテーブルを書き換えると仮定する．送信側は受信側がテーブルを書き換えたことに気づかず，古い VID のままデータ送信を開始する（1）．この送信に用いた VID では，すでに過去の通信によって送受信のノードの MAC アドレスがネットワーク中のスイッチに学習されており，正しい経路を通って受信側に到達する．受信側ではこのパケットに対する Ack パケットを新しい VID を用いて送信する（2）．この VID では，スイッチは送信側への経路を学習していないため，パケットはフラッディングされる．このとき同時に新しい VID における受信側への経路情報が各スイッチに学習される．送信側では (b') のタイミングでテーブルが書き換わり，以後の通信では新しい VID を用いて通信を始める（3）．受信側への新しい VID における経路はすでにスイッチが学習済みなので，正しい経路で受信側に到達する．またこの送信で送信側の正しい経路がスイッチに学習され，以後のこのノード間の送受信ではフラッディングは起こらず，正しい経路で送受信が行われる．

以上 2 つの場合において，2 ノード間でテーブルの同期がとれていない場合でも通信が成立することを述べた．2 ノード間でテーブルの同期がとれていない (a)–(b) もしくは (a')–(b') の間はフラッディング通信が発生することになる．フラッディングが継続して発生すると，ネットワーク全体のスループットを低下させる原因になるが，この状況は一時的なもので，通信相手の VID 変更が終了するとともに解消されることが分かる．そこで変更要求前後に MPI 同期関数（*MPI_Barrier( )* 等）で同期をとることで，このフラッディングによって生じるネットワークのスループットの低下を最小に抑えることもできる．もちろん，この同期そのものが通信をともなうためにフラッディングの原因になりうるが，同期をとらずにバースト転送を始めるよりも影響は小さくなる．この問題については 5.2 節で詳しく考察する．また，クラスタの規模が大きくなった場合には，各々のノードがさまざまな VID を用いて通信を行い，それによってネットワーク中のスイッチでそれらのノードへの経路がすでに学習されているケースも十分考えられる．その結果として，それらのノードがルーティングテーブルを変更した場合でも，その初めの送信でフラッディングが起る確率が低くなる．

5. 評　　価

実装したシステムの動作を確認し，それらのシステムでルーティングテーブルを変更した場合のコストを評価する．加えて前述のフラッディングがどのようにネットワークに影響を及ぼすのかを示す．最後に，最適なルーティングテーブルが判明しているベンチマークである NPB Kernel-CG に対して本システムを適用し，最適にルーティングテーブルを固定した場合と比較して，動的にルーティングテーブルを書き換える場合が同等の性能が得られることを確認する．

評価環境として**表 1** に示すノード構成で 16 ノードを用意した．これらのノード環境を基に，**図 8** に示すネットワークからなるクラスタを構築した．それらは，(a) すべてのノードを 1 台のスイッチに収容した Flat 構成，(b) 4 台のスイッチに 4 ノードずつ接続しスイッチ間を Tree 接続した Tree 構成，(c) 4 台のスイッチにそれぞれ 4 ノードずつ接続しスイッチ間に 4 つの経路を用意した VBFT 構成の計 3 種類のクラスタである．

表 1 評価環境
Table 1 Evaluation environment.

CPU	Intel Xeon 3.0 GHz EM64T 1-way
Memory	DDR2/400 1.0 Gigabytes
NIC	Intel PRO/1000 MT Dual Port Server Adapter （PCI-X 64 bit/133 MHz）1 ポートのみ使用
OS	Linux Kernel 2.6.20
MPI	Open MPI Ver. 1.2.3
Compiler	GCC Ver. 3.4.6
スイッチ	DELL PowerConnect 5224 Gigabit Ethernet 24 Port

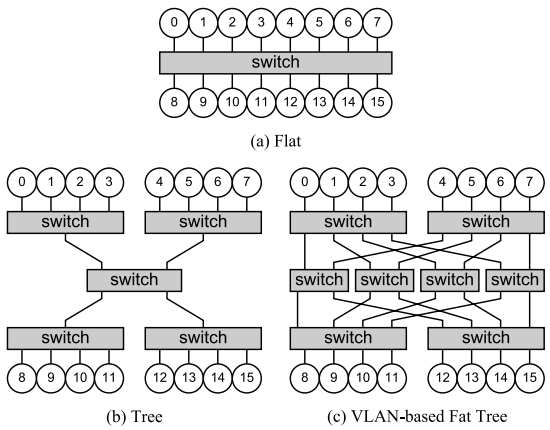


図 8 実験ネットワーク
Fig. 8 Experimental networks.

5.1 ルーティングテーブル変更時間

ルーティングテーブルの変更時間について評価する．実装した MPI のインタフェースを用いて，ローカルで動作している VFN-NM に対してルーティングテーブルの変更を要求し変更が有効になるまでの時間を計測する．次に，VFN-NM 内でルーティングテーブルの変更に要する時間を計測する．それぞれの計測では，1,000 回変更した場合の平均値を算出する．

その結果，MPI のインタフェースを用いて，ルーティングテーブルの変更が終了するまでに必要な時間は約 39.5 μsec，またそのうち VFN-NM 内部でルーティングテーブル変更に要する時間は約 8.3 μsec となった．

これらの結果より MPI のインタフェースを用いたルーティングテーブル変更で必要とする時間の多くは VFN-NM 内でルーティングテーブルの変更に要する時間ではなく，MPI インタフェースから VFN-NM への通信時間であることが分かる．しかしながら，先に述べたように VFN-NM の役割は，ノード内部からのリクエストだけでなく外部からのリクエストの処理も念頭に入れて設計している．また，我々の想定するルーティングテーブル変更の粒度はプログラム単位もしくはイタレーション単位である．一般的に通常の Ethernet と TCP/IP を用いた場合の片道通信遅延時間はおよそ 30 μsec であるので，この評価結果の示す変更コストは，想定するルーティングテーブル変更粒度で十分実用になると考えられる．

5.2 テーブル変更にともなうフラッディングの影響

次にルーティングテーブルを変更したときに発生するフラッディングがネットワークのパフォーマンスに与える影響を評価する．図 8 (c) に示すネットワーク構成の計 16 台のノード構成において，ノード n からノード $n+8$ へパケットをいっせいに送信する（$n = 0, 1, 2 \cdots, 7$）．初めのルーティングテーブルではすべてのノードが VID = 1 のみ用いて通信を行う．この通信の最中にノード 0-8 間の通信の VID を $1 \rightarrow 2$ に変更する．フラッディングが及ぼすネットワークへの影響を観察するために，まず初めにノード 0 のルーティングテーブルを変更の後，間隔をあけてノード 8 のルーティングテーブルを変更する．今回は実験開始から 20 秒でノード 0 のルーティングテーブルを変更し，40 秒でノード 8 のルーティングテーブルを変更した．この 20 秒から 40 秒の間の 20 秒間でノード 0 から 8 への通信はすべてフラッディングが発生する．

結果を**図 9** に示す．縦軸に受信側ノード 8-15 で測

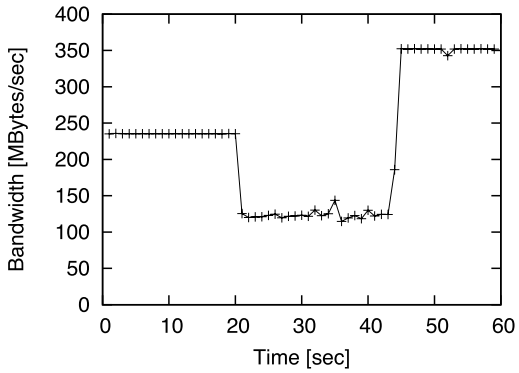


図 9 ルーティングテーブル変更によるネットワークへの影響
Fig. 9 Evaluation results of effect of network by a routing table change.

定されたバンド幅の合計値，横軸に経過時間を示している．ルーティングテーブルを変更する 20 秒までは，図 8 (b) で示す基本となる Tree と同じとなる．そこで得られるバンド幅は，ノード 0-3 とノード 8-11 間，ノード 4-7 とノード 12-15 間でそれぞれ 1 経路ずつとなり，理論ピーク値で合計 2 経路 250 MBytes/sec となる．評価の結果でも，この部分で得られた値は約 235 MBytes/sec となり，ほぼ予想された結果となる．その後の 20 秒からは，ルーティングテーブルを変更している過程でノード 0 がフラッディング送信を行うため，その影響からバンド幅が低下している．結果が示すように，一部のノードでルーティングテーブルを変更することでフラッディング送信が発生し，それがネットワーク全体のスループットを低下させている．その後 40 秒でノード 8 でのルーティングテーブルが書き換わることで，フラッディングによる送信はなくなり，またノード 0-8 間の通信経路が切り替わったことで，結果として通信経路が増えることになり，その結果としてバンド幅は向上する．

この結果の示すところは，たとえ 1 つのノード間のルーティングテーブルが異なるだけでも，それがフラッディング送信を引き起こし，ネットワーク全体のバンド幅に大きな影響を及ぼすことである．このような状況がクラスタ中の多くのノード間通信で起きた場合では，より大きな影響を引き起こす．この問題を解決する 1 つの方法は 4.3 節で述べたようにルーティングテーブルの変更の際に何らかの同期処理を入れることである．同期処理で行われる通信は最小限のパケット数で行われるため，ここでの評価のように同期をとらずにバースト転送を行う場合と比較して，送信によるフラッディングの発生数を少なくする．同期処理を入れるタイミングはルーティングテーブルを変更する前後に入れることが望ましいが，ルーティングテーブルの変更する前後の処理特性によって，前後どちらか一方での同期で済ませることも可能である．一般的には，ルーティングテーブルを変更した後すぐにバースト通信を始めるのならばルーティングテーブルの変更後であり，バースト転送直後であるならばルーティングテーブルの変更前になる．これらは，前後の特性を考慮に入れてユーザが個別に制御することとする．

5.3 NPB Kernel-CG

最後に本システムが正しく動作していることを確認するために，2.4 節で解析を行った NPB Kernel-CG（ver.3.2 CLASS=B NPROCS=16）を用いて実証する．ここでは次の 5 つの場合を比較する．

(1) 図 8 (a) で示した Flat 環境
(2) 図 8 (b) で示した Tree 環境
(3) 図 8 (c) で示した VBFT で既存の固定ルーティングテーブルを用いた場合
(4) 図 8 (c) で示した VBFT であらかじめ最適化を行った固定ルーティングテーブルを用いた場合
(5) 図 8 (c) で示した VBFT で最適化ルーティングテーブルを本システムを用いて与えた場合

VBFT 構成におけるルーティングテーブルの最適化では，すでに 2.4 節で示した最適化例を適用する．(4) では既存の固定ルーティング方法を使うが，VID の割当てテーブルを図 4 に示す「最適化例」の表を考慮した設定とする．この場合，ルーティングテーブルでは，特定の VID が多く利用される偏った設定となっているが，Kernel-CG の通信パターンでのみ最適になるようになっている．(5) の場合では，アプリケーション実行中に *VFN_SetRouteMPI* を用いてルーティングテーブルを設定した．

(4)，(5) の 2 つの方法は最終的に設定されているルーティングテーブルがほぼ等価になる．両者の違いは (4) が Kernel-CG のカーネルループに入る前から偏った VID テーブルによる通信を行っていることに対して，(5) はカーネルループ中にこれを適用している点である．この 2 つの評価によって新しく実装したシステムが正しく動作していることを確認する．

ルーティングテーブルを動的に変更するために，プログラムソース中で変更用関数を呼び出す．**図 10** に実際に Kernel-CG のソースファイル（cg.f）中に加えた変更を示す．Kernel-CG の，MPI の初期化・終了処理の前後に *initialize_mpi* と *VFN_FinalizeMPI* で VFREC-Net での動的ルーティングテーブル変更のための初期化・終了処理を行う．そして，CG のイタレーションごとにルーティングテーブルの変更を行

```
call initialize_mpi    (cg.f 中の mpi 初期化関数)
call VFN_InitMPI    (VFREC-Net の初期化処理)
  ⋮
do it=1, niter
call conj_grad(···)    (CG のイタレーションのメイン部分)
  ⋮
enddo
  ⋮
call VFN_FinalizeMPI    (VFREC-Net 終了処理)
call mpi_finalizer    (cg.f 中の mpi 終了関数)
- - - - - - - - - - - - - - - - - - - - - - - - -
subroutine conj_grad(···)
  ⋮
call VFN_SetRouteMPI(1, 4, 2)
  ⋮   (テーブルの書換え)
call VFN_SetRouteMPI(11, 14, 2)
  ⋮
```

図 10 Kernel-CG に追加した変更（cg.f）
Fig. 10 An additional example code of NPB Kernel-CG.

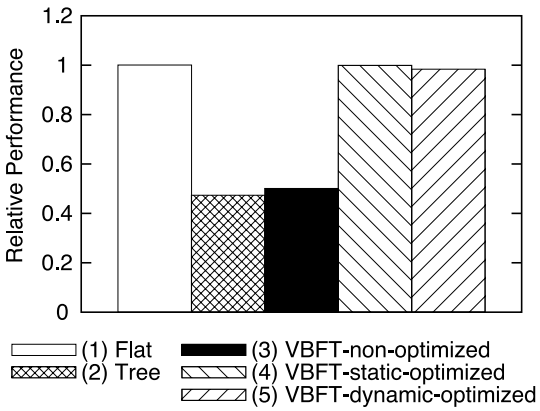


図 11 NPB Kernel-CG における評価結果
Fig. 11 Result of performance evaluation on NPB Kernel-CG.

う．CG での各イタレーションでメインとなる部分は，*conj_grad(···)* であり，この内部で，図 4 に示している，最適化後のルーティングテーブルを設定する．

すべての通信を同一スイッチ内で閉じることが可能な Flat 構成のネットワークは，他の構成のネットワークと比較して，最も良い結果を得ることができると考えられる．そこで，本評価結果は (1) で得られた結果を基にした相対性能で評価する．評価結果を**図 11** に示す．予想どおり，すべての通信が同一スイッチ内で閉じる (1) の環境が，最も性能が良い．一方で多くの通信で衝突が発生する (2) は最も性能が悪い．VBFT 構成を用いた (3) の場合，(2) よりも若干性能が良い．固定されたルーティングテーブルで，かつルーティングテーブルを最適化していない場合でも，複数ある通信経路に分散して送受信が行われるため，わずかであるが (2) と比較して結果は良くなる．しかし，2.4 節に述べた問題により VBFT 構成のメリットがほとんど得られていない．さらに，固定ルーティングテーブルの VBFT 構成の場合でも，ルーティングテーブルを最適化した (4) で性能は大幅に（約 2 倍）改善され，その性能は (1) とほぼ等しい．このことから VBFT で得られる高いバイセクションバンド幅を，より効率的に使うためには，通信パターンを考慮に入れたルーティングテーブルの設定が重要であることが分かる．

最後に VBFT 構成を用いて初期化時に最適化を施した (4) とプログラム実行時にルーティングテーブルを設定した (5) を考察する．前述のように，この 2 つのルーティングテーブルはほぼ等価となり，実行前に最適化されたルーティングテーブルが設定されているか，実行時に必要な部分のみ最適なルーティングテーブルを設定するかの違いとなる．ルーティングテーブルの変更はイタレーション数分だけ呼び出され，そのたびに 5.1 節で評価したテーブル変更のコストが必要になる．Kernel-CG class B におけるイタレーション数は 75 であるため，この合計コストは約 2,965 μsec となる．動的にルーティングテーブルを設定した場合，ベンチマークの計算時間は約 26 sec であった．そのためこのルーティングテーブル変更で必要になった時間は，全体の時間に比べて十分に小さく，これがベンチマークに及ぼす影響は小さい．評価結果では，それ以上の差として見えているが，これは誤差であると考えられる．結果として，プログラム開始前に最適なルーティングテーブルを固定した場合でも，動的に変更した場合でもベンチマークで示された結果に大きな差が現れず，動的にテーブルを変更することによる通信最適化機能が正しく動作していることが確認された．

6. 関 連 研 究

本論文で提案している VFREC-Net を開発するきっかけとなった工藤らの提案した VLAN ルーティング法[2] は，Linux の標準の VLAN 実装を用いて，複数の仮想的な Ethernet デバイスを作り，それぞれに独立した IP アドレス空間を与えることで，送信に用いる VID を制御している．しかし，実際にはそれらの仮想デバイスに独自の IP アドレスを割り当てなくても，同じ IP アドレスを割り当てたうえで，iproute 等

の Linux のルーティング機能を用いることで使用する VID を制御することが可能である．これを行うことで，Linux 標準の VLAN 実装と iproute 機能だけで VLAN ルーティング法を実現することができるとともに，本論文で示したようなルーティング変更が行えることが期待できる．一方で，システムの維持には多くの仮想 Ethernet デバイスと複数のシステムにまたがった統合的な管理が必要になり，メンテナンス性が低い．また，本論文の評価で示したような短時間でのルーティングテーブルの切替えも期待できない．一方の VFREC-Net ではモジュールによるデバイスドライバが必要であり，これに加えてルーティングは MAC アドレスにより行われるため，初期の導入コストが高くなる．しかし，1 度システムを導入さえすれば，以後の管理では VFN デバイスドライバとそれを管理する VFN-NM のみでシステムが成り立ちメンテナンス性は高くなる．本システムは初期の VLAN ルーティング法と比べて，VID の制御を 1 つの閉じたシステムで管理できることが利点となる．また，本論文の評価で示したルーティングテーブルの変更時間により，ある程度細かい粒度でのルーティングテーブルの変更にも対応できる．また，本システムが最終的な目標とする自律システムを構築する場合，既存の VLAN ルーティング法では何らかの他システムを用いてノード間の通信量等を取得する必要があるのに対して，我々の提供するシステムでは VFN ドライバ中で細かいレベルのログを他のシステムの介在なしに取得でき，これは今後の拡張性にも役立つ．

大塚らの研究[8] ではルーティングを行うための VID のタグ付けをノードではなくスイッチで行う．そのため送受信のノードはスイッチ間のルーティングのために特別なドライバや設定は必要とせず，スイッチの機能のみを用いて VLAN ルーティング法を実現する．この実装においても本論文で述べているようなトラフィックの偏りに対して，経路変更を行うシステムが必要になる．経路変更処理はスイッチの設定変更で行うことができるが，そのためには，スイッチに対する何らかのインタフェースが必要になる．我々の実装では，ルーティングの管理は送信ノードにおけるパケットへの VID 割当てのみで行われ，これはソフトウェア制御で実現されている．

最後に InfiniBand での類似システムとして，OpenFabrics Alliance[9] が提供する OpenSM があげられる．InfiniBand では，Fat-Tree のような複数経路を持つネットワークトポロジを設計段階から許容している．これらのトポロジを許容するために，InfiniBand ではネットワークを管理するシステムとして Subnet Manager と呼ばれるサービスを必要とする．OpenSM は，この Subnet Manager の実装の 1 つであり，ネットワークトポロジとして，複数経路を持つ Fat-Tree をサポートし，パケットのルーティングテーブル等の管理を行っている．本論文のターゲットとしているネットワークは Ethernet であるが，VFN-NM が提供する機能はこの OpenSM と近い部分がある．ただし，本技術は InfiniBand でのみ提供される機能であり，本論文で提案・実装したシステムは，より広く使われている Ethernet の標準技術を用いて実現できる．

7. おわりに

本論文では VFREC-Net における動的ルーティングテーブルの設定手法を提案し，それを実現するためのフレームワークを開発した．これを用いることで VID の割当ての最適化を行い，ネットワーク上の経路制御を積極的に行うことが可能になる．このようなことから，各種通信衝突制御アルゴリズムを VFREC-Net に適用可能になっている．このフレームワークは各ノードで動作する新規に開発した管理プログラムである VFN Network Manager によって提供される．今回は，このフレームワークを用いて MPI 上から制御するライブラリもあわせて開発した．このようなシステムを用い，ユーザが適切にルーティングテーブルを設定することで，より複雑な通信パターンを持つアプリケーションにおいても，ネットワークの持つ性能を最大限に利用できると期待できる．

参考文献

1) IEEE: 802.3ad — Link Aggregation. http://standards.ieee.org/getieee802/802.3.html
2) 工藤知宏，松田元彦，手塚宏史，児玉祐悦，建部修見，関口智嗣：VLAN を用いた複数パスを持つクラスタ向き L2 Ethernet ネットワーク，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG06(ACS6), pp.35–44 (2004).
3) 三浦信一，岡本高幸，朴 泰祐，佐藤三久，高橋大介：VFREC-Net：ドライバ制御による VLAN を用いたマルチパスネットワーク，情報処理学会論文誌：コンピューティングシステム，Vol.47, No.SIG12(ACS15), pp.35–45 (2006).
4) IEEE: 802.1Q — Virtual LANs. http://www.ieee802.org/1/pages/802.1Q.html
5) Graham, R.L., Woodall, T.S. and Squyres, J.M.: Open MPI: A Flexible High Performance MPI, *Proc. 6th Annual International Conference on Parallel Processing and Applied Math-*

ematics, Poznan, Poland (2005).
6) Myricom, inc.: Myrinet. http://www.myri.com/
7) InfiniBand Trade Association: InfiniBand. http://www.infinibandta.org/
8) 大塚智宏，鯉渕道紘，手塚宏史，工藤知宏，天野英晴：スイッチでタグ付けを行う VLAN ルーティング法，情報処理学会論文誌：コンピューティングシステム，Vol.47, No.SIG12(ACS15), pp.46–58 (2006).
9) The OpenFabrics Alliance. http://www.openfabrics.org/

(平成 19 年 5 月 7 日受付)
(平成 19 年 9 月 20 日採録)

三浦 信一（学生会員）
昭和 54 年生．平成 14 年千歳科学技術大学光科学部光応用システム学科卒業．平成 16 年筑波大学大学院理工学研究科修士課程修了．現在，同大学院システム情報工学研究科在学中．クラスタコンピューティングに関する研究に従事．

岡本 高幸（学生会員）
昭和 58 年生．平成 18 年筑波大学第三学群情報学類卒業．現在，同大学大学院システム情報工学研究科在学中．クラスタ用ネットワークに関する研究に従事．

朴 泰祐（正会員）
昭和 36 年生．昭和 59 年慶應義塾大学工学部電気工学科卒業．平成 2 年同大学大学院理工学研究科電気工学専攻後期博士課程修了．工学博士．昭和 63 年慶應義塾大学理工学部物理学科助手．平成 4 年筑波大学電子・情報工学系講師，平成 7 年同助教授，平成 16 年同大学大学院システム情報工学系助教授，平成 17 年同教授，現在に至る．超並列計算機アーキテクチャ，ハイパフォーマンスコンピューティング，クラスタコンピューティング，グリッドに関する研究に従事．平成 14 年度および平成 15 年度情報処理学会論文賞受賞．日本応用数理学会，IEEE CS 各会員．

塙 敏博（正会員）
平成 10 年慶應義塾大学大学院理工学研究科計算機科学専攻博士課程修了．博士（工学）．東京工科大学コンピュータサイエンス学部講師を経て，現在，筑波大学計算科学研究センター研究員．計算機アーキテクチャ，並列処理に関する研究に従事．