移動指向ネットワークアーキテクチャの設計と実装

舌 間 一 宏[†,☆] 寺 岡 文 男[††]

ノードの移動は前提であるという観点からネットワークアーキテクチャを再検討し，移動指向ネットワークアーキテクチャを提案する．新アドレス体系として，ノードが接続しているサブネットを示す“ネットワーク指示子”とノードそのものを指し示す“ノード識別子”からなる移動指向アドレスを用いる．トランスポート層より上位層では，ノード識別子のみを用いて通信相手を識別することで移動透過性を実現する．また本アーキテクチャをIPv6に実装し，無視できるほどのオーバーヘッドで移動透過性が実現できることを示す．

Design and Implementation of Mobility Oriented Network Architecture

Kazuhiro Shitama[†,☆] and Fumio Teraoka[††]

From the viewpoint that every node is mobile, this paper reviews current network architectures and proposes the “mobility oriented network architecture”. The new address format consists of the “network prefix” that specifies the subnet to which the node is connected, and the “node identifier” that identifies the node itself. Making use of the node identifier in the transport and upper layers achieves transparent mobility. The proposed architecture is implemented onto IPv6 and shows that mobility is achieved with negligible overheads.

1. はじめに

インターネット上では移動ノードに対応するための様々なプロトコルが数多く提案されてきた[1)~5)]．現在提案されているプロトコルのほとんどは，ノードの移動を前提としていない既存のインターネットアーキテクチャ上で，移動ノードを例外として対応するものである．このアプローチは既存のインターネットとの互換性を考える上では有用であるが，今後モーバイルコンピューティングの形態がさらに多様化し普及していった場合，必ずしも適当な手段とはいえない．そのため，既存との互換性を重視する既存の方式ではなく，ノードの移動は前提であるとしてネットワークアークテクチャ自体を再検討し，既存のインターネットアーキテクチャに代わる，新しいネットワークアーキテクチャの構築が必須であると考えられる．

本稿では，ノードの移動は前提であるという観点からネットワークアーキテクチャを再検討し，モーバイルコンピューティングに適した新たなネットワークアーキテクチャを提案する．特に，通信するノードの位置や移動に関係なく，そのノードと通信できる性質である移動透過性を今後のモーバイルコンピューティングにおけるキーワードとし，まず，この移動透過性を提供する上で最適なアドレス体系を提案する．さらに提案したアドレス体系を基にネットワークアーキテクチャの設計を行う．また，提案したネットワークアーキテクチャのプロトタイプを実装し，基本動作に関して評価を行い，本ネットワークアーキテクチャの有用性について議論する．

なお，本方式の提案は文献[6)]でも行っているので，本稿では実装および評価に重点を置いて記述する．

2. 移動指向ネットワークアーキテクチャ

2.1 アドレスフォーマット

ノードの移動が前提となる場合に，ノードに対して必要な情報とは以下に示す2点である．

- ノードの移動に対して不変なノード自体を識別するための“ノード識別子”
- ノードまで経路制御を行なうためのノードの“位置指示子”

そこで本稿では位置指示子とノード識別子の2つの情報を格納した新しいアドレス体系として，移動指向ア

† 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University
☆ 現在，ソニー(株)インフォメーション&ネットワーク研究所
Presently with Information & Network Technologies Laboratories, Sony Corp.
†† (株)ソニーコンピュータサイエンス研究所
Sony Computer Science Laboratories, Inc.

ドレス (Mobility-oriented Address) を提案する．移動指向アドレスはネットワーク指示子 (Network Prefix) とノード識別子 (Node Identifier) という，2 つの異なる役割をもつ部分から構成されている．

ネットワーク指示子はノードの位置指示子の役割を果たし，IP アドレスのようにノードのインターフェースの位置を示すのではなく，ノードが接続しているサブネットの位置を示す．ノード識別子は，ノードがどのサブネットに接続しているかに関係なく，ノード自体を識別するための不変的な情報である．ここで示すノード識別子は，MAC アドレスのような NIC(Network Interface Card) を示すインターフェース識別子とは異なるものである．つまり，ノードに挿している NIC を入れ替えることでインターフェース識別子が変わってもノード識別子は不変である．また，ルータのような複数のインターフェース識別子を持つノードでも，そのノードに割り当てられるノード識別子は必ず 1 つとなる．

移動指向アドレスは，ノードの位置指示子とノード識別子をアドレス内の 2 つの部分に分離して格納しているため，モーバイルコンピューティングでノードを非常に取り扱いやすいアドレス体系である．特に IP アドレスと異なり，ノードの移動に対して不変なノード識別子を格納しているため，移動指向アドレスのノード識別子部分を用いてトランスポート層でコネクションを確立することにより，移動透過性を実現することが可能となる．

2.2 プロトコル階層

移動指向ネットワークアーキテクチャは，アプリケーション層，移動指向トランスポート層，移動指向ネットワーク層，データリンク層の 4 つの階層から構成されるネットワークアーキテクチャである (図 1参照)．各層に関して，既存のインターネットアーキテクチャとの相違点を示しながら説明する．

アプリケーション層では，ユーザは通信を行なう相手ノードのノード識別子を指定し，各アプリケーションは要求されたサービスをユーザに提供する．ユーザはノード識別子を指定することにより，相手ノードの位置を考慮することなく，通信を行なうことができる．

移動指向トランスポート層はアプリケーション層に対し，指定されたノード識別子で示される相手ノード上のアプリケーションと自ノード上のアプリケーションにおける End-to-End の通信を提供する．移動に対して不変なノード識別子を用いてコネクションを確立するため，相手ノードや自ノードが移動した場合でも，コネクションは継続される．

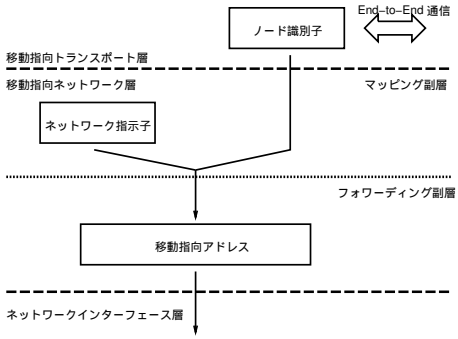

図 1　移動指向ネットワーク層の内部構造

移動指向ネットワーク層は上位層で指定されたノード識別子と，そのノード識別子に対応するノードの現在のネットワーク指示子を結合し，移動指向アドレスを生成する．さらに，移動指向アドレスを基に異なるサブネット間の経路制御を行なう．データリンク層は既存のインターネットアーキテクチャにおけるデータリンク層と同じである．

移動指向ネットワークアーキテクチャにおいて移動透過性を実現する上で，核となる層はノード識別子とネットワーク指示子の対応づけを行なう移動指向ネットワーク層である．移動指向ネットワーク層は，マッピング副層とフォワーディング副層という 2 つの副層から構成されている．

マッピング副層では，上位層で指定されたノード識別子に対応するネットワーク指示子を結合し，移動指向アドレスを生成する．指定されたノードに関するノード識別子とネットワーク指示子の対応づけは，**ACT** (Address Combination Table) で行ない，ノードごとの対応づけを **ACT** エントリと呼ぶ．

なお，マッピング副層で相手ノードに関する ACT エントリがない，つまり相手ノードの現在のネットワーク指示子がわからない場合は，相手ノードのホームネットワークを示すネットワーク指示子を相手ノードのノード識別子と結合し移動指向アドレスを生成する．ノードのホームネットワークとは，そのノードが普段接続されているサブネットを示す．なお，ホームネットワークのネットワーク指示子のことをホームネットワーク指示子と呼び，ホームネットワークに接続しているルータをホームルータと呼ぶ．ホームルータは，そのホームルータが管理している移動ノードに関する最新の ACT エントリを保持する．

フォワーディング副層では，マッピング副層で生成された移動指向アドレスを基に経路制御を行ない，パケットを送受信および中継を行なう．このように移動

指向ネットワークアーキテクチャのフォワーディング副層は，既存のインターネットアーキテクチャのネットワーク層と同じ機能を持つ．

3. 移動透過性の実現

3.1 ノード間の通信プロトコル

ノード間の通信は以下のように行なわれる．ここでは，あるノード CN(Correspondent Node) が，HR(Home Router) をホームルータとする移動ノード MN(Mobile Node) にデータパケットを送信する場合を考える (図 2参照)．

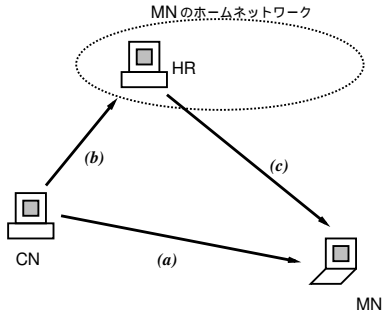

図 2　通信プロトコル

CN が MN に関する ACT エントリを保持している，すなわち，MN が現在どのサブネットに接続しているかを CN が知っている場合は，MN が現在接続しているサブネットのネットワーク指示子と MN のノード識別子を移動指向ネットワーク層のマッピング副層で結合して移動指向アドレスを生成する．フォワーディング副層では，その移動指向アドレスにしたがって経路制御するため，データパケットは，直接 MN に向けて転送される (図 2(a))．

一方，CN が MN に関する ACT エントリを保持していない，すなわち，MN が現在どのサブネットに接続しているか知らない場合は，MN のホームネットワーク指示子と MN のノード識別子をマッピング層で結合して移動指向アドレスを生成する．フォワーディング副層では，その移動指向アドレスにしたがって経路制御するため，データパケットは，MN のホームネットワークに向けて転送される (図 2(b))．そのデータパケットを MN のホームルータである HR が受信すると，HR は MN に関する ACT エントリを必ず保持しているため，受信したデータパケットに記載されている宛先ノードに関するネットワーク指示子部分をホームネットワーク指示子から現在 MN が接続しているサブネットを示すネットワーク指示子に書き換え，MN へ転送する (図 2(c))．

3.2 ノードの移動プロトコル

ノードの移動プロトコルを以下に示す．ここでは，ノード CN と通信中である移動ノード MN が別のサブネットに移動した場合を考える (図 3参照)．なお，MN のホームルータを HR とし，MN が新しく接続したサブネット上のルータを FR(Foreign Router) とする．

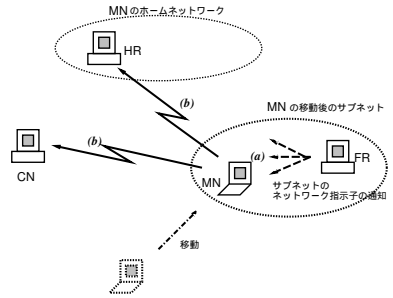

図 3　移動プロトコル

MN は新たなサブネットに移動すると，接続先のサブネット上の FR から定期的にそのサブネットに対してブロードキャストされているそのサブネットを示すネットワーク指示子を受信する (図 3(a))．受信したネットワーク指示子と移動ノードのノード識別子を結合し，移動先での移動指向アドレスを生成する．このように，移動指向アドレスを使用すると，既存のインターネットにおける DHCP のようなステートフルアドレス自動設定プロトコル (Statefull Address Autoconfiguration Protocol) は特に必要なく，移動先サブネット上で流れているネットワーク指示子に関する情報を受信するだけで移動先でのアドレスを獲得することが可能となる．

移動先のサブネットで新たに移動指向アドレスを獲得した MN は，HR および CN へ移動通知メッセージを送信し，別のサブネットへ移動したことを報告する (図 3(b))．この際に用いられるメッセージを **ACT** コントロールメッセージと呼ぶ．

なお，ACT コントロールメッセージのやりとりには，ホームルータや通信中の相手ノードに不正な ACT エントリが作成されることを回避するために，メッセージが改竄されていないことと送信ノードの認証を必ず行なわなければならない．

ACT コントロールメッセージを受信した CN や HR は, メッセージの送信ノードを認証した後，メッセージに格納されている最新の移動指向アドレスを抽出し，そのノードが保持する ACT エントリに登録する．

これにより，MN 宛に送信されたパケットを受信した HR は最新の ACT エントリに従い，MN へその受信パケットを転送することが可能となる．また CN は登録された MN に関する ACT エントリに従って，MN 宛のパケットを HR を経由することなく，直接的に MN へ送信することが可能となる．

なお，自ノードの移動時にのみ ACT コントロールメッセージを送信するだけでなく，そのノードのホームルータ経由でパケットを受信した場合も，そのパケットの送信ノードが自ノードに関する ACT エントリを保持していないと判断し，その送信ノードへ向けて ACT コントロールメッセージを送信する．

3.3 名前サーバとの連携

移動指向ネットワークアーキテクチャにより，移動指向ネットワーク層より上位層のプロトコルやアプリケーションはノード識別子を用いることで移動透過性を実現することが可能となる．しかし，ノード識別子はあくまでもビット列であるため，非常に覚えにくく扱いにくい．さらに，2.2節で述べたように，移動指向ネットワーク層のマッピング副層で通信相手ノードに対する ACT エントリがない場合のために，通信相手ノードのホームネットワーク指示子を獲得するための機構が必要となる．そこで，この 2 点を解決するために名前サーバを利用する．つまり，ノードのホスト名に対してそのノードのノード識別子とホームネットワーク指示子を名前サーバの資源レコードとして格納しておく．

4. 移動指向インターネット MOInet

本稿では，2節で述べた移動指向ネットワークアーキテクチャのプロトタイプを，次世代インターネットプロトコルとして注目されている IPv6(Internet Protocol version 6)[7] をベースに実装を行なった．このプロトタイプを移動指向インターネット **MOInet** (Mobility-Oriented Internet) と呼ぶ．

4.1 移動指向 IPv6 アドレス

2.1節で述べた移動指向アドレスの概念を適応した IPv6 アドレスを移動指向 **IPv6** アドレスと呼び，そのフォーマットを図 4に示す．移動指向 IPv6 アドレス 128 ビットの上位 64 ビットには，ネットワーク指示子を格納し，下位 64 ビットにノード識別子を格納する．このように 64 ビット + 64 ビットという分割方式を採用した理由は，上位 64 ビットの部分が，IETF で提案されている現在の IPv6 アドレスの仕様[8] での上位 64 ビットと結果として同じ役割となり，既存の経路制御機構を変更しなくてすむからである．

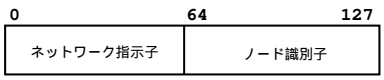

図 4　移動指向 IPv6 アドレスフォーマット

また，このフォーマットにより，MOInet でサポートできるノード数は $2^{64}(\cong 10^{19})$ となる．この値は，IETF の IPng(Intenet Protocol next generation) Working Group で議論された次世代インターネットとして最低限サポートすべきノード数 10^{12} を十分に満足している[9]．

4.2 MOInet の実装方針

MOInet の実装は，FreeBSD 2.2.7-Release 上でノート PC のための PAO パッケージおよび KAME プロジェクトから配布されている IPv6 パッケージをインストールした PC 上で行なった．

2.2節で述べた移動指向ネットワーク層は，IPv6 層を改良することにより，**moIP** 層として実装を行なった．また，トランスポート層では，通信ノードの位置や移動に依存しない End-to-End 通信を提供するために，パケットのデマルチプレックス時は，自ノードおよび相手ノードの移動指向 IPv6 アドレス下位 64 ビットのノード識別子のみに注目し，上位 64 ビットのネットワーク指示子は無視するように改良を行なった．

また，3.1節で述べたノード間の通信プロトコルやノードの移動プロトコルにおいて必要となる ACT コントロールメッセージの送受信や ACT エントリの追加削除の機能は，ユーザ空間において **actd** デーモンとして実装した．

本実装では，アプリケーションが通信相手ノードのホスト名からノード識別子に変換するときに名前サーバに問い合わせる際に，同時にノードのホームネットワーク指示子を取得するようにした．また，名前サーバにノードのホームネットワーク指示子とノード識別子を格納するための資源レコードを新たに追加するのではなく，既に用意されている 128 ビットの AAAA レコードの上位 64 ビットをノードのホームネットワーク指示子，下位 64 ビットをノード識別子としたため，名前サーバ自体に特別な改良は行なっていない．

4.3 ACT の実装

ノードのネットワーク指示子とノード識別子の対応づけを管理する ACT はカーネル内の構造体として実装している．ACT の各エントリのフォーマットを図 5に示す．ACT エントリは 4 つのフィールドから構成されており，1 エントリのサイズは 24 バイトである．

ACT エントリはハッシュテーブルにより管理されている．ハッシュ値は，エントリに格納されているノー

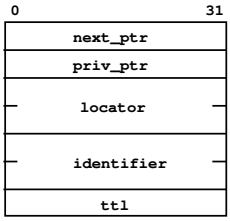

図 5　ACT エントリ

ド識別子の上位 32 ビット値と下位 32 ビット値の排他的論理和をエントリ数ACT_HASH_SLOTSIZE(デフォルト値は 1023) で割った剰余である．エントリの追加による衝突が発生するとハッシュスロットにリストとして追加される．カーネル内に定義されている ACT の操作関数を以下に示す．

- moinet_init() :ACT の初期化
- moinet_slowtimo() :エントリのエージング
- moinet_hashact() :ハッシュ値の計算
- moinet_searchact() :エントリの検索
- moinet_addact() :エントリの追加
- moinet_deleteact() :エントリの削除
- moinet_initact() :全エントリの削除

4.4　moIP 層の実装

moIP 層のブロックダイヤグラムを図 6に示す．移動指向ネットワーク層のモデル的には，マッピング副層とフォワーディング層を分離して実装するのが理想的であるが，本実装では実装の容易さを考慮して，2 つの副層の明確な分離をしていない．なお，フォワーディング副層の機能は，4.1節で述べたように既存の経路制御機構を利用できるように移動指向 IPv6 アドレスを定義したため，既存の IPv6 層におけるip6_input() 関数，ip6_forward() 関数，ip6_output() 関数の利用が可能である．そこで，マッピング副層の機能を実現するための関数としてmoinet_input() 関数，moinet_output() 関数を新しく作成した．

- moinet_output() 関数
  宛先アドレスの下位 64 ビットが示す宛先ノード識別子に対応する ACT エントリを検索する．エントリが存在する場合はエントリが示すネットワーク指示子を宛先アドレスの上位 64 ビットとし，ない場合は宛先アドレスの上位 64 ビットをそのまま用いる．
- moinet_input() 関数
  自ノード宛以外のパケットを受信した場合，宛先アドレス下位 64 ビットのノード識別子に対応する ACT エントリを検索する．エントリが存在する場合は，宛先アドレス上位 64 ビットでエントリを更新する．この動作は，ホームルータのみの機能である．

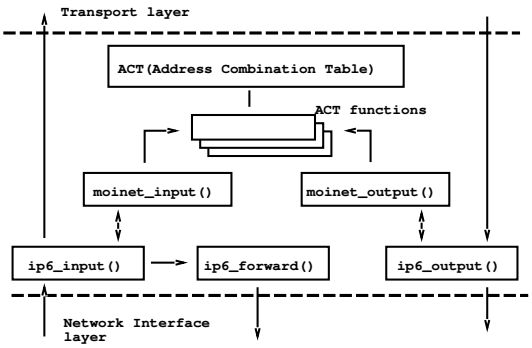

図 6　moIP 層のブロックダイヤグラム

4.5　actd の実装

actd は，メッセージ送受信モジュールと ACT 管理モジュールから構成されている．メッセージ送受信モジュールは移動発生シグナルを受信すると，ホームルータおよび通信中の相手ノードへ最新のネットワーク指示子を格納した ACT コントロールメッセージを送信する．また，ACT コントロールメッセージの受信を行ない，ACT 管理モジュールに対してエントリ操作要求を行なう．ACT 管理モジュールはメッセージ送受信モジュールからのエントリ更新要求に伴い，カーネル内の ACT に新規エントリの登録，および既に存在するエントリの更新を行なう．

メッセージ送受信モジュールにおける ACT コントロールメッセージの送受信は UDP を用いて実装している．ACT コントロールメッセージには次に示す 2 つの種類がある．移動通知メッセージは，ノードの移動に伴いホームルータおよび通信中の相手ノードに自ノードの最新のネットワーク指示子を通知する場合に用いる．応答メッセージは，受信した移動通知メッセージに対する確認応答として用いられる．なお，3.2節で述べたように，悪意のあるユーザからの ACT コントロールメッセージによって，ホームルータや通信中の相手ノードに不正な ACT エントリが追加されることを回避するための，移動通知メッセージの正当性と送信ノードの認証には，AH (Authentication Header)[10] を用いている．メッセージフォーマットを図 7に示す．ACT コントロールメッセージは 6 つのフィールド (合計 32 バイト) から構成されている．

5.　MOInet の評価

5.1　評価項目および環境

図 8に示す実験ネットワークにおいて，以下の 3 点

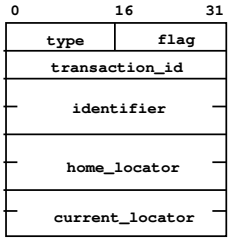

図 7　AMT コントロールメッセージ (UDP)

について評価を行った．

- 通信処理時間 :送受信ノードにおける moIP 層での処理時間およびホームルータにおけるパケット転送に要する moIP 層での処理時間．
- 登録処理時間 :ノード移動時におけるコントロールメッセージの送受信および ACT エントリの更新に要する時間
- ACT 操作時間 :ACT のエントリ数の増加に伴う，エントリの操作にかかる処理時間．

図 8において，ノード 3 台は全て同じ性能をもつ PC (CPU Pentium 90MHz, RAM 32M) であり，使用するネットワークは 10Base T である．ノード M はサブネット N1 をホームネットワークとする移動ノード，ノード H はノード M のホームルータ，ノード C はノード M の通信相手ノードである．各ノードの ACT のハッシュスロットサイズ`ACT_HASH_SLOTSIZE` はデフォルト値の 1023 を用いた．

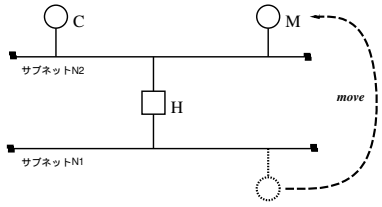

図 8　実験トポロジ

5.2　通信処理時間

ノード C からノード M に向けて ICMPv6 Echo パケットを`ping6` コマンドを用いて 100 回送信し，その際の各ノードの moIP 層での処理時間を測定し，その平均値を求めた．なお，ノード C および M には互いのノードに関する ACT エントリではないランダムな 10 個の ACT エントリを，ノード H にはノード M に関する ACT エントリとランダムな ACT エントリ 9 個を，それぞれ追加した状態で測定を行なった．ACT のエントリ数が ACT エントリの検索時間にほとんど影響を与えないことは 5.4節で示す．

表 1　ネットワーク層における基本動作の比較

測定項目	moIP 層	IPv6 層	オーバーヘッド
送信 (ノード C)	19.9 μsec	11.2 μsec	77.7 %
転送 (ノード H)	32.1	23.2	38.4
受信 (ノード M)	12.0	11.0	9.1

測定結果を表 1に示す．表 1で示されるように，送信時および転送時におけるオーバーヘッドは受信時のそれと比較して非常に大きい．これは，送信時では上位層から指定された宛先ノードに関する ACT エントリの検索，またホームルータでの転送時では受信したパケットの宛先ノードに関する ACT エントリの検索が，それぞれ必要なためである．5.4節で詳しく述べるように，ACT エントリの検索時間はエントリ数に関わらず約 7 ~ 8μ秒となる．IPv6 層に比べ moIP 層は，送信時で 8.7μ秒，転送時で 8.9μ秒の処理時間が増加していることから，このオーバーヘッドの主要因が，ノード識別子に対応する ACT エントリの検索時間であることがわかる．それに対して受信時では，moIP 層は IPv6 層と比較して特に特別な処理を行なっていないため，処理時間のオーバーヘッドは小さい．

送信および転送時における，ACT エントリの検索による処理時間の増加を大きいと見積もるかどうかであるが，この約 10μ秒足らずの処理時間の増加で，既存のインターネットアーキテクチャで不可能であったノードの移動透過性を実現することができるのであれば，十分考慮できるといえる．また，この ACT エントリの検索時間は，通信する 2 ノード間の RTT(Round Trip Time) と比較すれば十分小さく，ほぼ無視できる値といえる．

5.3　登録処理時間

図 8において，ノード C と通信中のノード M が，サブネット N1 に接続したとする．それに伴う, ノード M，ノード C，ノード H 上で動作しているノード M から ACT コントロールメッセージ (通知メッセージ) を 100 回送信した際の，各ノードでの各処理時間を測定し，その平均値を求めた．ノード C およびノード M の ACT の初期設定は 5.2節の測定と同様である．

測定結果を図 9に示す．なお，ノード M での応答メッセージの受信に関する処理時間は，図の都合上ノード C が送信した応答メッセージに関する処理時間のみを示す．

図 9において，移動を通知する側のノードで必要な処理時間は約 280μ秒，また通知を受信する側のノードで必要な処理時間は約 130μ秒であり，ノードの移動に伴う処理において各ノードでかかる処理時間の合計

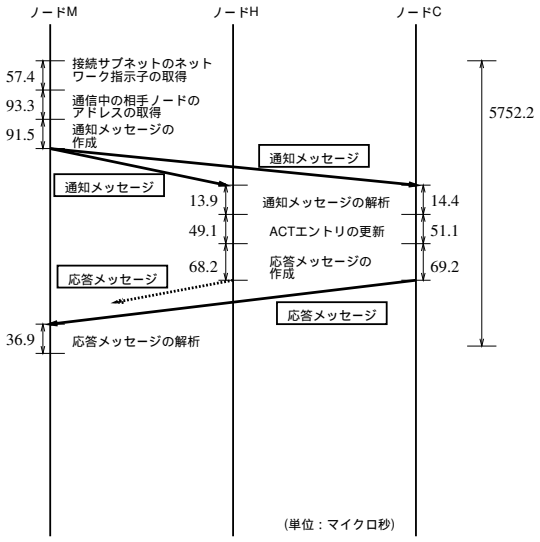

図 9　actd の処理時間の測定結果

は約 410μ秒となる．それに対し，移動を通知するノードにおいて，移動を検知してから応答メッセージの受信処理が完了するまでにかかる合計時間は約 $5.8m$ 秒である．このことから各ノードでかかる処理時間は，ノード間の RTT と比較すると非常に小さいことがわかる．また今回の測定は同じサブネットに接続しているノード間で測定を行なっており，移動通知に関するメッセージのやりとりを行なうノード間が数ホップ離れている場合を考えると，この各ノードでの処理時間はほとんど無視できるといえる．

5.4　ACT 操作時間

ACT のエントリ数の増加に伴う，検索，追加，削除の処理時間を測定した．図 8のノード M において，4.3節で述べた`moinet_searchact()` 関数，`moinet_addact()` 関数，`moinet_deleteact()` 関数を呼び出す`moinet_evalact()` 関数を，測定のためにカーネル内に便宜上作成した．エントリ数が 1，10，50，100，500，1000 の場合に，`moinet_evalact()` 関数が，各関数を呼び出して，その関数から戻ってくるまでの時間を 100 回測定し，その平均値を求めた．ACT 内の各エントリはランダムに作成し，追加するエントリもランダムに作成したものを用いた．検索および削除するエントリは ACT エントリからランダムに抽出したものを用いた．

追加，削除，検索の処理時間の測定結果を図 10に示す．図 10で示されているように，検索，追加，削除とも ACT のエントリ数の増加に関わらず，多少振動があるがほぼ一定の処理時間となっている．これは，4.3節で述べたように ACT を線形リストではなく，ハッシュで管理しており，理論上は $O(1)$ のオーダーとなるためである．しかし，エントリ数の増加に伴い，ハッシュ値の衝突が発生する確率が高くなるため，どの処理時間も若干ではあるが徐々に線形増加している傾向が読み取れる．

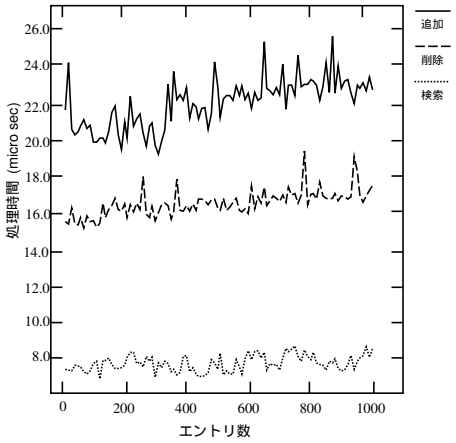

図 10　追加，削除，検索に関する処理時間の測定結果

追加および削除の処理時間が，検索の処理時間よりも大きくなっているのは，検索がただ単にハッシュ値の計算および衝突が生じた際のリストの走査だけでよいのに対し，その作業に加えて，追加の場合は新しいエントリのための`MALLOC` のコスト，削除の場合はエントリを削除するための`FREE` のコストがそれぞれ必要となるからである．なお，追加と削除の処理時間の差は，`MALLOC` と`FREE` の処理時間の差と考えられる．

この`MALLOC` と`FREE` によって追加および削除の処理時間が検索の処理時間よりも大きくなってしまうことを回避するための方法としては，ノードの起動時に予め決められた数の ACT エントリを`MALLOC` しておき，それらを割り当てスプールとして管理する方法が上げられる．つまり，追加要求に応じてスプールからエントリを取り出し，また削除要求に応じてエントリをスプールに戻し，そしてスプールが空になった場合に限り，`MALLOC` を実行するすといった作業を行なうことにより，追加/削除ごとに毎回`MALLOC`/`FREE` を実行する必要がなくなるため，処理時間が大幅に小さくなると考えられる．

6. 考　　察

6.1　Mobile IPv6 との比較

本稿で提案した MOInet は，Mobile IPv6[5]などの既存の方式と比較し，以下のような利点を有する．

- IPsec を利用して移動ノードが他のノードと通信

している場合，MOInet ではノードが移動してもの SA (Security Association) を維持することができる．Mobile IPv6 ではノードが移動するごとに SA の確立をやり直さなくてはならない．

- MOInet では IPv6 ヘッダの長さが送信中に変化しない．すなわち IP 層で拡張ヘッダは基本的に付加されない．Mobile IPv6 では，送信ノードのホームアドレスを示す拡張ヘッダが IP 層で付加されるため，アプリケーションや TCP は拡張ヘッダ分をあらかじめ見込んで送信データ長を決めないと，送信する際にフラグメンテーションが発生し，通信効率が低下する．
- MOInet ではノードのネットワークインターフェイスが交換されてもノードのアドレスは変化しない．ネットワークインターフェイスの交換は，たとえば携帯コンピュータを机上で使用する際には 100BASE-TX の PC カードを使用し，会議室に移動する際には無線 LAN の PC カードを使用する，などのときに発生する．Mobile IPv6 では，ネットワークインターフェイスが交換されると IPv6 アドレスも変化してしまうため，ホームエージェントへの登録や通信相手との IPsec の SA の再確立などが必要となる．

6.2 GSE 方式との比較

IETF における IPv6 の標準化過程において，GSE 方式[11]と呼ばれるアドレス付け方式が提案された．GSE 方式も 128 ビットの IPv6 アドレスを上位 64 ビットの経路情報と下位 64 ビット位のインターフェイス識別子に分離している．本稿で提案した移動指向アドレスと GSE アドレスでは以下の 2 点が大きく異なる．

- 移動指向アドレスはノードに割り当てられるのに対し，GSE アドレスはネットワークインターフェイスに割り当てられる．したがって，上述したようにネットワークインターフェイスの交換があった場合，GSE 方式では基本的にアドレスが変化してしまう．
- GSE 方式では送信ノードは上位 64 ビットを空白にして送信し，組織の外に出るときにルータがこの部分に経路情報を代入するが，移動指向アドレスではこのような操作はしない．

また，文献[12]は，GSE 方式では TCP の接続拒否攻撃が容易に実現できるため，セキュリティに問題があると指摘している．これは，悪意のあるノードが別のノードに成りすましてサーバなどに TCP で接続すると，本物のノードから同じサーバへの TCP 接続が拒否されてしまうという問題である．同様の指摘は移動指向アドレス方式にも当てはまる．紙面の都合上詳細は割愛するが，この問題はネームサーバの逆引き (IP アドレスからホスト名を検索) とホームエージェントによる転送，および TCP のシーケンス番号の一貫性をチェックすることにより，容易に回避できる．

7. おわりに

本研究では，ノードの移動は前提であるという観点から移動指向ネットワークアーキテクチャを提案し，これを IPv6 に適用した MOInet を実装・評価した．パケットの送信，受信，および転送という基本動作において，MOInet は通常の IPv6 と比較して無視できる程度のオーバーヘッドで実現できることを示した．

参考文献

1) Teraoka, F., Uehara, K., H., S. and Murai, J.: VIP: A Protocol Providing Host Mobility, *Communications of the ACM*, Vol. 37, No. 8, pp. 67–75 (1994).
2) Teraoka, F. and Uehara, K.: Mobility Support in IPv6 based on the VIP mechanism, *Proceedings of INET'95* (1995).
3) Perkins, C.: IP Mobility Support, RFC2002, *IETF* (1996).
4) Perkins, C. and David, J.: Mobility Support in IPv6, *Proceedings of MOBICOM '96* (1996).
5) David, J. and Perkins, C.: Mobility Support in IPv6, Internet draft, work in progress, *IETF* (1998).
6) 舌間一宏, 寺岡文男: v6VIP の設計と実装, 情報処理学会マルチメディア，分散，協調とモーバイル (DICOMO'98) シンポジウム 論文集 (1998).
7) Deering, S. and Hinden, R.: Internet protocol, version 6 (IPv6) specification, RFC2460, *IETF* (1998).
8) Hinden, R., O'Dell, M. and Deering, S.: An IPv6 Aggregatable Glocal Unicast Address Format, RFC2374, *IETF* (1998).
9) Kastenholz, F. and Partridge, C.: Technical Criteria for Choosing IP: The Next Generation (IPng), RFC1726, *IETF* (1994).
10) Kent, S. and Atkinson, R.: IP Authentication Header, RFC2402, *IETF* (1998).
11) O'Dell, M.: GSE - An Altenate Addressing Architecture for IPv6, Internet draft, work in progress, *IETF* (1997).
12) Crawford, M., Mankin, A., Narten, T., Stewart III, J. W. and Zhang, L.: Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6, Internet draft, work in progress, *IETF* (1998).