仮想マシンの広域ライブマイグレーションを実現するゲスト透過なMobile IPv6トンネリング機構

広渕 崇宏 中田 秀基 伊藤 智 関口 智嗣
産業技術総合研究所 情報技術研究部門

我々は、データセンタの運用柔軟性を向上させるため、仮想マシン（VM）の広域ライブマイグレーションに注目している。例えば、電力需要が逼迫した際に一部の仮想マシンを一時的に遠隔拠点に待避してサービスを継続できる。このときゲストOSがIPアドレスを維持したまま透過的に通信を継続できる必要があり、我々はMobile IPv6（MIPv6）技術に着目してきた。しかし、MIPv6は強力なトンネリング機構を備えるものの、既存のMIPv6技術をそのままマイグレーションに用いることは難しい。ゲストOSを改変することなく透過的にトンネリングを可能とし、VM一つからでも柔軟にマイグレーションできる機構が必要である。そこで、我々はゲストOSにとって透過的なMIPv6トンネリング機構（Kagemusha）を提案する。提案機構はホストOS上で動作し、Client MIPv6のシグナリングやトンネリングをゲストOSに対して透過的に行う。ゲストOSにMIPv6に関するプログラムを導入する必要はない。さらに、既存のHome Agent（HA）や仮想マシンモニタを一切変更することなくそのまま利用できる。プロトタイプ実装を用いて評価実験を行った。その結果、提案機構はHAと正しく通信でき、そのトンネリングオーバーヘッドはわずかであることが確認できた。またQemu/KVMのライブマイグレーションと正しく連係動作し、移動先ネットワークにおいてもゲストOSに対して透過的なネットワーク接続を提供できた。このときマイグレーションにともなうダウンタイムの増加はわずか数百msにとどまることが確認できた。

A Guest-Transparent Mobile IPv6 Tunneling Mechanism for Wide-Area Live Migration of Virtual Machines

Takahiro Hirofuchi Hidemoto Nakada Satoshi Itoh Satoshi Sekiguchi
National Institute of Advanced Industrial Science and Technogy (AIST)

We are developing a wide-area live migration mechanism that allows dynamic load balancing of virtual machines (VMs) among datacenters. We consider that Mobile IPv6 (MIPv6) is a promising technology to support transparent network reachability when VMs migrate to foreign networks. Existing MIPv6 mechanisms, however, are not suitable for VM migrations; real-world IaaS datacenters require guest-transparent and flexible tunneling mechanisms, which are not provided by existing MIPv6 programs. In this paper, we propose a guest-transparent MIPv6 tunneling mechanism (Kagemusha), which performs Client MIPv6 signaling and tunneling on a host operating system. No MIPv6 program is required to be installed into a guest operating system. The proposed system is fully compatible with existing home agents (HAs). It basically works with any virtual machine monitors. Through experiments, we confirmed that our prototype system successfully established MIPv6 tunnels with HAs, and its performance overhead was negligible for normal use cases. We also confirmed that the prototype system successfully worked with live migrations; the downtime of migration increased only by several hundred milliseconds.

1 はじめに

我々が進めている省エネデータセンタプロジェクトでは、複数の計算機センタやデータセンタ間をまたがって仮想マシン（VM）の配置を調整することで、資源運用の効率性を高め稼働コストを削減することを目指している。仮想マシンモニタ（VMM）が備えるライブマイグレーション機能によって、VMを一切停止することなく異なる物理ノード上に再配置できる。この機能を応用すれば、例えば、ある計算機センタの電力使用量が電力使用制限値に近づいた場合に、一部のVMを一時的に電力に余裕のある遠隔の計算機センタに透過的に待避できる。サービスを継続して提供しながら、移動元計算機センタの物理サーバの一部を停止し消費電力を抑制できる。

VMのライブマイグレーション機構は本来LAN環境を想定して設計されているため、異なる拠点間でマイグレーションするためには、解決すべき課題が存在する。

我々はこれまでに、複数拠点からなる仮想化データセンタの管理システム [1] や、VM が利用する仮想ディスクの広域マイグレーションシステム [2] 等の開発を進めてきた。本稿では、VM が遠隔拠点に移動した際に、どのようにしてネットワーク到達性を維持するのか、という点に焦点を当てる。

広域ライブマイグレーションにおいて VM が異なるネットワークに移動した際にも、ゲスト OS が IP アドレスを変更することなく動作を継続するためには、何らかの手法で移動元ネットワークに対してパケット転送（トンネリング）を行う必要がある。我々は、トンネリングを行う手法として、今後普及が期待される Mobile IPv6 (MIPv6)[3] 技術に着目している。その理由として、

- 標準化されたオープンなプロトコルである。そのプロトコルは実際に動作する実装を用いてインタオペラビリティを検証しながら設計されている。
- セキュリティ機能が基本的な要件としてあらかじめ組み込まれている。
- 複数のオープンソースの実装やルータベンダによる実装が存在する。
- トンネル中継装置（Home Agent）の多重化等、大規模なネットワークでの運用を想定した機能が存在する。
- トンネリングの切り替え時間の短縮（高速ハンドオーバー）等、VM 移動にとっても有益な発展的な拡張が検討されている。
- つくばエクスプレスの無線 LAN や WIMAX、移動端末ネットワーク等で関連技術が利用されている、もしくは利用されることが期待されている。

等が挙げられる。VM の広域ライブマイグレーションに対して応用した場合にも、その強力な機能の恩恵を被ることができると考えている。

しかしながら、現状、VM マイグレーションと一緒に使うことは一般的ではない。VM がモバイルノードとして動作するためにはカーネルやユーザランドの専用プログラムが VM 内部で正しく設定されている必要がある。しかし、現状 MIPv6 を実装したオペレーティングシステムは一部のバージョンの Linux カーネルや BSD 系 OS 等、限定的である。ユーザが自由にゲスト OS を選択・カスタマイズできる IaaS クラウドサービスにおいて、すべての VM 中に MIPv6 が導入され、正しく設定されることを前提とすることが難しい。VM 内部に MIPv6 を導入することなく、 透過的に MIPv6 のトンネリングが可能になる技術が必要である。

既存の MIPv6 関連技術においては、モバイルノードに MIPv6 関連プログラムを導入することなく透過的にトンネリングを行う技術が存在する。サブネットワークの透過的な移動を提供する NEMO[4] においては、Mobile Router（MR）とよばれる特殊なルータがサブネットワーク配下のノードと同時に移動し、MIPv6 のシグナリングやトンネリングを担う。サブネットワーク中に存在するノードは、サブネットワークが異なるネットワーク中に移動したことを意識することはない。しかし、VM の広域移動に用いる場合、同一サブネットワークに存在する複数の VM を同時に移動する際には適しているものの、そのうち一つの VM のみを遠隔拠点に移動する場合には適さない。移動対象の VM に加えて、MR の役割を担うノード（VM）も移動する必要があり、マイグレーションのコストが大きい。

また、携帯電話網を念頭に設計された Proxy Mobile IPv6（PMIPv6）[5] では、モバイルノードの代わりに、ネットワーク機器が MIPv6 のシグナリングやトンネリングを担う。ゲスト OS に MIPv6 のプログラムを導入することなく、VM を透過的に広域移動できる点で有用であると考えられるものの、我々の知る限り一般に利用できるオープンソースの実装は存在しない。また、モバイルノードと外部との通信は常に HA を経由して行われる設計であり、そのネットワーク到達性は常に HA に依存してしまう。HA の冗長化技術等は提案されているものの、安定して長期的に運用するノウハウは一般的に十分蓄積されているとは言い難い。我々は、基本的には VM を起動元拠点で運用しつつ、需要が逼迫した場合には、一時的に余裕がある外部の拠点へ待避させることを想定している。平時の運用においては、HA を経由せずに直接 VM が外部と通信できるようにして十分な通信性能を維持し、また HA を運用するコストを省略したいと考えている。

そこで、我々は、VM の広域ライブマイグレーションを実現すべく、ゲスト OS にとって透過的な MIPv6 トンネリング機構（Kagemusha）を新たに提案する。Kagemusha はハイパーバイザーと連動して動作し、MIPv6 のシグナリングやトンネリングをゲスト OS の代わりに行う。ゲスト OS に MIPv6 に関するプログラムを導入する必要はなく、ゲスト OS にとって透過的に動作する。VM が移動元拠点に存在する場合には、HA を経由することなく直接通信できる。既存の HA は一切変更することなくそのまま利用できる。ホスト OS (Xen[6] においては Domain 0）レベルで動作する軽量な実装であり、Qemu/KVM[7] や Xen 等においてハイパーバイザーに特殊な変更を行わずにそのまま用いることができる。MIPv6 において必須とされるシグナリングメッセージの暗号化に対応する。

以降、2 節で、提案機構について説明し、3 節で評価

について述べる。4 節で今後の課題を述べ、5 節で関連研究にふれる。6 節で本稿をまとめる。

2 提案機構

本節では MIPv6 の動作概要について説明し、次に我々の提案機構について述べる。

2.1 Mobile IPv6

本稿では特に断りがない場合、Mobile IPv6 (MIPv6) という用語で、その拡張的な仕様である NEMO や PMIPv6 を指すのではなく、最も基礎的かつ一般的な仕様である Client Mobile IPv6（CMIPv6）を指す。図 1 にその概要を示し、その動作を（VM の移動に特化しない、一般的な視点から）説明する。

移動ノード（Mobile Node）が移動元拠点（ホームネットワーク、Home Network）に存在する場合には、MIPv6 特有の振る舞いはなく、通常の IPv6 プロトコルに沿ったパケット送受信を行う。移動ノードがホームネットワークで使用しているアドレスをホームアドレス（Home Address）と呼ぶ。移動ノードが遠隔拠点（訪問先ネットワーク、Foregin Network）に移動した場合には、MIPv6 の仕組みにより移動ノードは引き続きホームアドレスを使用して通信を行うことができる。アプリケーションは引き続きホームアドレス宛のパケットを受信し、またホームアドレスを送信元アドレスとしてパケットを送信できる。このとき実際には、ホームネットワークに存在するホームエージェント（Home Agent, HA）を経由してパケットの送受信が行われている。移動ノードは訪問先ネットワークで気付けアドレス（Care of Address, CoA）を取得しており、このアドレスを用いて HA との間でパケットのトンネリングを行っている。

訪問先ネットワークにおいて、MIPv6 では大きく分けて 2 つの処理を行う。第一はシグナリングであり、トンネルを確立するのに必要な情報を移動ノードと HA 間で伝達する処理である。特に重要な処理として、Binding Update（BU）（移動ノードが CoA を HA に通知し、トンネルの確立を依頼するメッセージ）と Binding Acknowledgement（BA）（Binding Update の成否を HA から移動ノードに通知するメッセージ）が存在する。

第二はトンネリングであり、移動ノードと HA 間でパケットを転送する処理である。移動ノード上のアプリケーションが（ホームネットワークに存在していたときと同様に）ホームアドレスを送信元としてパケットを送出すると、移動ノードの MIPv6 対応カーネルはそのパケットを IPv6-over-IPv6（IP6IP6）トンネリング・パケットにカプセル化した上で HA に転送する。HA はそのカプセル化を解き、移動ノードの代わりに、ホームアドレスを送信元とするパケットを通信相手に対して送出する。一方、通信相手からホームアドレス宛のパケットがホームネットワークに届くと、HA が移動ノードの代わりに受信し、そのパケットを移動ノードに対して同様にトンネリングして転送する。移動ノードは IP6IP6 トンネリングのカプセル化を解き、ホームアドレスを送信先とするパケットをアプリケーションに届ける。

MIPv6 の経路最適化機能は本稿で取り扱わないため説明しない。訪問先ネットワークにおいても、HA を経由しないで直接通信相手と通信する機構である。しかし、その機能を利用するためには通信相手側も MIPv6 に対応する必要がある。不特定多数の端末を通信相手とするウェブサーバが大半の利用を占める一般的な IaaS クラウドサービスでは利用が難しい。

2.2 提案機構に対する要件

ここで、1 節で述べた議論を踏まえ、提案機構に対する要件をまとめる。以上に述べた MIPv6 の動作においては、移動ノード上の MIPv6 対応カーネルおよびデーモンプログラムがシグナリング処理とトンネリング処理を行っている。しかし、IaaS クラウドサービスにおいて VM 内部でそれらの処理を担うのは難しく、ゲスト OS にとって透過的な MIPv6 機構が必要である。さらに、VM 一つからの広域マイグレーションに柔軟に対応し、移動元拠点では HA を経由しない通信を可能にするためには、CMIPv6 に対応した仕組みである必要がある。

以上の議論を踏まえて、提案機構に対する要件は、

- CMIPv6 のシグナリング処理やトンネリング処理をゲスト OS が意識することなく VM の外部で実行する機構であること。

とまとめることができる。

2.3 提案機構（Kagemusha）の概要

そこで我々は、ゲスト透過な MIPv6 トンネリング機構である Kagemusha を提案する。ハイパーバイザーと連動して動作し、VM 外部のホスト OS 上で CMIPv6 のシグナリングおよびトンネリングを行う。ホスト OS（Xen においては Domain 0）レベルで動作する軽量な実装であり、Qemu/KVM や Xen 等においてハイパーバイザーに特殊な変更を行わずにそのまま用いることができる。広域ライブマイグレーションにおいて提案機構を用いると、ゲスト OS が MIPv6 に対応することなしに、 MIPv6 の移動支援技術を享受できる。VM が遠隔拠点に移動後も同じ IPv6 グローバルアドレスを使用し続けられる。MIPv6 の普及を妨げる要因として対応 OS の少なさが挙げられるが、提案機構はこの点を解消し MIPv6 の普及を促進できる可能性があると考えている。

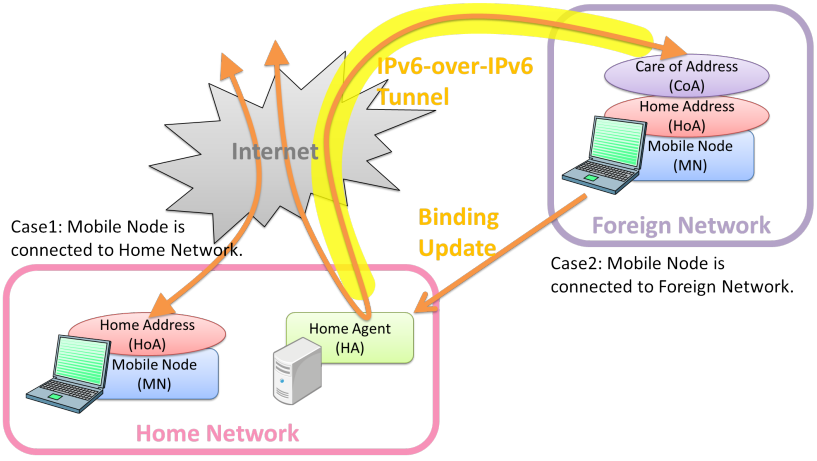

図 1　Mobile IPv6 の概要

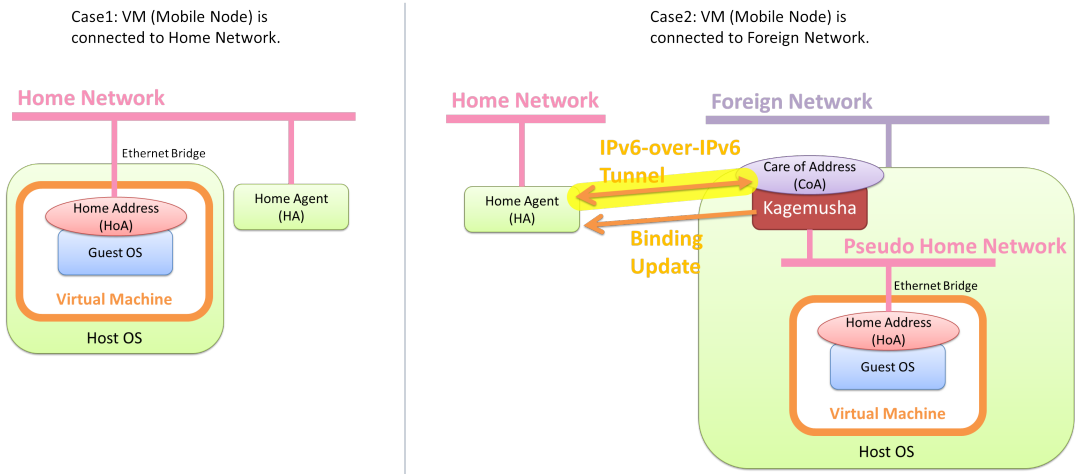

図 2　提案機構（Kagemusha）の概要

その概要を図 2 に示す。VM が移動元拠点に存在する場合には、VM をホームネットワークに対してブリッジ接続し、ホームアドレスを静的にゲスト OS に割り当てる。この時点では MIPv6 に特化した処理は何ら行われず、ゲスト OS はホームネットワーク上の IPv6 ノードとして通常のネットワーク通信処理を行う。

一方、VM が広域マイグレーションによって遠隔拠点に移動した際には、提案機構が動作してゲスト OS に対して透過的な MIPv6 トンネリングを可能とする。ホスト OS 上で動作するプログラムである Kagemusha は、以下の 3 つの機能を担う。

- **MIPv6 シグナリング処理:** 遠隔ネットワークにおいて CoA を取得し、ゲスト OS の代わりに MIPv6 シグナリングおよびトンネリング処理を行う。HA に対して BU リクエストを送信し、IP6IP6 トンネルの作成を要請する。
- **MIPv6 トンネリング処理:** ゲスト OS が送出したパケットを受け取り、IP6IP6 トンネルパケットにカプセル化して HA に対して送信する。また HA から IP6IP6 トンネルパケットを受信し、カプセル化を解いてゲスト OS に対して受け渡す。
- **疑似ホームネットワーク処理:** Kagemusha とゲスト OS と間でのパケットの受け渡しにはホスト OS 上に作成した擬似的なホームネットワークを介して行う。ゲスト OS がホームアドレスを保持したまま移動元拠点と同様にパケットを送受信することを可能にする。

我々は、ゲスト OS が有するホームアドレスおよび

VM の MAC アドレスを Kagemusha が把握しているという前提を置いている。Kagemusha は、シグナリング処理においてホームアドレスの情報を BU および BA メッセージに含める必要がある。また Kagemusha が MAC アドレスをあらかじめ把握できていれば、疑似ホームネットワークにおける近隣探索処理を軽減できる。一般的に IaaS クラウドサービスにおいて、ゲスト OS に割り当てるグローバルアドレスや VM の MAC アドレスはサービス運営者側が管理しており、以上は十分妥当な前提であると考えている。

2.4 設計

提案機構の基本的な有効性を迅速に検証することを念頭にプロトタイプを設計した。コードの移植性やメンテナンスのしやすさを考えて、ユーザランドにおいて極力標準的な socket API を用いて開発した。

Kagemusha は遠隔拠点のホスト OS 上で動作するデーモンプロセスであり、VM 一つに対して一つ起動する。Kagemusha は以上に述べた機能をそれぞれ実装するコンポーネントからなる。また、プロトタイプ実装の簡略化のため、ゲスト OS はデフォルトルータのアドレスを知っており、あらかじめ静的に設定されているという前提を置く。ホームネットワークが複数のルータを持つ場合も現状では考えない。

2.4.1 MIPv6 シグナリング処理

BU メッセージの送信および BA メッセージの受信のため、`IPPROTO_MH` タイプの RAW ソケットを作成する。HA に対して BU を送信して、移動先を通知し IP6IP6 トンネルの作成を促す。また BA を受信して、BU の成功を確認する。さらに稼働している間は、定期的に BU の送信を繰り返し、トンネリングを維持する。あらかじめホスト OS に複数の IPv6 グローバルアドレスを割り当てておき、そのうちのひとつを CoA として用いる。BU メッセージ中の終点オプションヘッダおよび BA メッセージ中の経路制御ヘッダは、sendmsg() および recvmsg() の補助データにより設定・取得する。モビリティヘッダ中のチェックサムは、チェックサム計算に用いる疑似 IPv6 ヘッダにおいて CoA ではなく HoA を用いる必要があるため、通常のカーネル内のルーチンに任せるのではなく、ユーザランドで独自に計算する。

2.4.2 MIPv6 トンネリング処理

IP6IP6 トンネル・パケットの送受信のため、`IPPROTO_IPv6` タイプの RAW ソケットも作成する。疑似ホームネットワークを介して VM が送出したイーサーネットフレームを受け取り、宛先が IPv6 グローバルアドレスであるパケットについてトンネリングを行う。IPv6 ヘッダ以降をペイロードとして上記ソケットから送信する。一方、上記ソケットからパケットを受信すると、適切なイーサーネットヘッダを付加して、疑似ホームネットワークへ送出し、VM に対してパケットを受け渡す。

2.4.3 疑似ホームネットワーク処理

疑似ホームネットワークは、ホスト OS のイーサーネットブリッジ機能を用いる。Kagemusha は tap デバイスを作成し、そのデバイスファイルを読み書きすることで、疑似ホームネットワークに対するイーサネットフレームの送受信を行う。VM の仮想 NIC として機能する tap デバイスと、Kagemsusha が作成した tap デバイスをイーサーネットブリッジで接続する。

Kagemusha は、あらかじめゲスト OS が有する IPv6 グローバルアドレス（ホームアドレス）および仮想 NIC の MAC アドレスを知っているので、近隣探索をすることなく VM に対してイーサーネットフレームを送信できる。一方、ゲスト OS は移動元拠点の場合と同様に近隣探索を行う可能性があり、Kagemusha は適切に対応する必要がある。現状では近隣要請メッセージへの対応のみ行っている。Kagemusha は近隣要請メッセージを受信すると、常に自らが管理する MAC アドレスを応答する。ゲスト OS に対して、ホームネットワーク上のノード[*1]宛のパケットはすべて Kagemusha 宛に送信させる。Kagemusha はそれらのパケットをトンネルを介して宛先ノードに転送する。

ライブマイグレーション直後のゲスト OS の近隣キャッシュの状態によっては、ゲスト OS が新たに近隣探索をすることなく疑似ホームネットワーク上にパケットを送信する可能性がある。この場合、Kagemusha は非要請近接応答メッセージを返して、リンク層アドレスの変更をゲスト OS に通知する。

このリンク層アドレスの変更は、ライブマイグレーション直後にゲスト OS の通信においてパケットロスが発生する要因となる。そこで、ライブマイグレーション後にデフォルトルータのリンク層アドレスが変わってしまうことを防ぐために、あらかじめホームネットワーク上のデフォルトルータの MAC アドレスがわかっているときには、Kagemusha の MAC アドレスとしてデフォルトルータの MAC アドレスを指定できる。

この疑似ホームネットワーク処理において、本来のホームネットワークの挙動をどの程度エミュレーションすべきか、という点については議論の余地がある。後述する今後の課題にて補う。

2.5 実装

Debian GNU/Linux（Squeeze）上にプロトタイプを実装した。コード量は C 言語で 1000 行程度である。

[*1] デフォルトルータも含む.

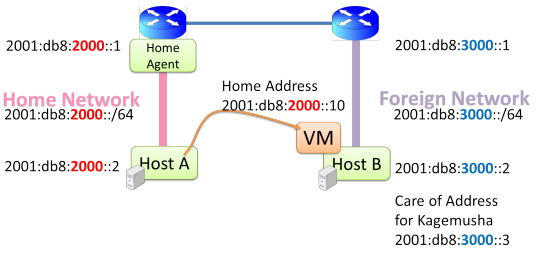

図 3　実験環境

Linux Kernel 2.6.32（および実装時点で最新のカーネルであった 2.6.39）においては、Kagemusha における BA の受信処理において、経路制御オプションヘッダを含むパケットがカーネル内で破棄されてしまうという問題があった。経路制御オプションヘッダの受信処理（`net/ipv6/exthdr.c` 内の `ipv6_rthdr_rcv()` 関数）において、経路制御オプションヘッダに指定されたホームアドレスがいずれかのネットワークインタフェースに設定されたアドレスであることを確認し、もし失敗すればパケットを捨てるという記述がある。Kagemusha の場合、ホスト OS のネットワークインタフェースはいずれもホームアドレスを持たない（ゲスト OS のネットワークインタフェースのみが持つ）ため、誤ってパケットが捨てられてしまっていた。`ipv6_rthdr_rcv()` 関数の該当部分を修正して、プロトタイプを動作させることができた。[*2]

3　評価

プロトタイプ実装を用いて評価実験を行った。実験環境を図 3 に記す。ルータ間のネットワークは 100Mbps でリンクアップされている。

HA には Linux における MIPv6 の実装である UMIP を使用した。本稿執筆時点の最新版（git レポジトリのコミット 74528e1）を用いた。

はじめにプロトタイプの基本的な動作を確認し、次に NEMO との比較実験を行った。さらにライブマイグレーションとの連係動作を確認した。VMM として qemu-kvm-0.12.5 および（ライブマイグレーションとの連係動作においては）qemu-kvm-0.14.1 を用いた。

3.1　基本動作の確認

はじめに Kagemusha により移動先ネットワークから透過的にトンネリングできることを確認した。Host B 上で Kagemusha を設定した上で、同ホスト上で VM を起動した。ゲスト OS からホームネットワーク上の Host A に対して `ping6` を実行し、MIPv6 シグナリングおよびトンネリングの動作確認を行った。Host B の物理ネットワークインタフェースのパケットキャプチャ（`tcpdump` の出力）の一部を Listing 1 に示す。基本動作の確認のため IPSec による保護は無効にしている。最初のパケットは、Kagemusha が CoA（`2001:db8:3000::3`）から HA（`2001:db8:2000::1`）に対して送信した BU メッセージである。2 番目のパケットは、この BU の成功を通知する BA メッセージである。lifetime が 60 秒に設定されていることから、トンネリングを維持するために 60 秒ごとに BU メッセージが送信されることがわかる。3 番目のパケットは、CoA から HA に対して送信された IP6IP6 トンネリングパケットである。ペイロードには、ゲスト OS が Host A に送信した ICMP6 Echo Request メッセージが IP ヘッダごと格納されている。4 番目のパケットは、この Echo Request に対する Echo Reply メッセージである。

次に、MIPv6 シグナリングメッセージを IPSec ESP で保護して同様の実験を行った。MIPv6 の規格では BU および BA メッセージを IPSec により保護することを必須としている。Kagemusha においては、ホスト OS 上でセキュリティ・ポリシーおよびセキュリティ・アソシエーションを設定する際に CoA を使用する。UMIP におけるモバイルノードの設定において HoA を用いる点と対照的である。Listing 2 における第 1 番目のパケットが BU メッセージであり、mobility header が IPSec ESP により暗号化されていることがわかる。第 2 番目のパケットである BA メッセージについても同様に正しく暗号化されている。MIPv6 トンネリングを確立できたため、ホームネットワーク上の Host A に対する Echo Request が成功している。

以上により、Kagemusha はゲスト OS に対して透過的に MIPv6 トンネリングが行えることが確認できた。既存の HA を改変することなくそのまま利用できることも確認できた。

3.2　NEMO との比較

ゲスト OS 透過な MIPv6 トンネリングを可能とする機構である NEMO と比較実験を行った。Kagemusha は、CMIPv6 のトンネリングに対応しており、VM 単体からの遠隔マイグレーションに対しても適用しやすい点で NEMO とは異なる。しかし、ゲスト OS 透過なトンネリングを実現する点で両者は類似しており、それらの性能オーバーヘッドを比較するため実験を行った。

NEMO の実験においては、MR として機能する VM（MR VM）をホスト B 上に新たに追加した。MR VM

[*2] その他にも XFRM 関連の処理（おそらく MIPv6 の実装である UMIP を前提としたコード）をコメントアウトするなど若干の改変が必要であった。

```
1 16:37:13.262580 IP6 2001:db8:3000::3 > 2001:db8:2000::1: DSTOPT mobility: BU seq#=13 AH
    lifetime=60
2 16:37:13.263017 IP6 2001:db8:2000::1 > 2001:db8:3000::3: srcrt (len=2, type=2, segleft=1,
    [0]2001:db8:2000::10) mobility: BA status=0 seq#=13 lifetime=60
3 16:37:13.956676 IP6 2001:db8:3000::3 > 2001:db8:2000::1: IP6 2001:db8:2000::10 > 2001:db8
    :2000::2: ICMP6, echo request, seq 78, length 64
4 16:37:13.956858 IP6 2001:db8:2000::1 > 2001:db8:3000::3: IP6 2001:db8:2000::2 > 2001:db8
    :2000::10: ICMP6, echo reply, seq 78, length 64
```

Listing 1 Kagemusha による MIPv6 シグナリングおよびトンネリングの様子（Kagemusha 起動ホストにおけるパケットキャプチャ）

```
1 19:41:06.123736 IP6 2001:db8:3000::3 > 2001:db8:2000::1: DSTOPT ESP(spi=0x000003e8,seq=0xd),
    length 68
2 19:41:06.124199 IP6 2001:db8:2000::1 > 2001:db8:3000::3: srcrt (len=2, type=2, segleft=1,
    [0]2001:db8:2000::10) ESP(spi=0x000003e9,seq=0x1b), length 52
3 19:41:06.557939 IP6 2001:db8:3000::3 > 2001:db8:2000::1: IP6 2001:db8:2000::10 > 2001:db8
    :2000::2: ICMP6, echo request, seq 394, length 64
4 19:41:06.558117 IP6 2001:db8:2000::1 > 2001:db8:3000::3: IP6 2001:db8:2000::2 > 2001:db8
    :2000::10: ICMP6, echo reply, seq 394, length 64
```

Listing 2 Kagemusha による MIPv6 シグナリングおよびトンネリングの様子（Kagemusha 起動ホストにおけるパケットキャプチャ、MIPv6 シグナリングを IPSec ESP で保護した場合）

は二つの仮想 NIC を有しており、一方を物理ネットワークにブリッジ接続し、もう一方を Mobile Network Node（MNN）として振る舞う VM の仮想 NIC とブリッジ接続している。HA については、図 3 の設定を若干変更し、MR に対してサブネットの管理を委譲するように設定し直している。

`iperf` によりトンネリングを経由するネットワークスループットを計測した。Host B 上のゲスト OS で動作する `iperf` クライアントから、Kagemusha あるいは MR VM を経由して、Host A 上の `iperf` サーバまでのスループットを計測している。計測結果を表 1 に示す。どちらの場合もスループットは 85Mbps 前後であり大きな差はなかった。これは、MIPv6 トンネリングを行わずにゲスト OS が移動先ネットワークのアドレスを取得して、`iperf` を計測した場合（87.9Mbps）と同等の値である。Kagemusha と NEMO を比較した場合、若干 NEMO の場合の方が高いスループットを示している。現状の Kagemusha の実装では、2 つのプロセスがそれぞれ、tap デバイス（あるいは IP6IP トンネル）から到着したパケットを逐次的に IP6IP6 トンネル（あるいは tap デバイス）へ転送している。実験に用いたノードの CPU は 2 コアでありながら、ホスト OS 上ではアクティブなプロセスが 1 つの VM プロセスを含め全部で 3 つあるため、若干スケジューリングの効率が悪いからではないかと推測している。

表 1 中の CPU 使用率とは、ホスト OS 上で計測した Kagemsusha および MR VM プロセスの CPU 使用率である。Kagemusha は 14% である一方、MR VM は 92% と非常に大きい。`iperf` クライアントを動かす VM は 70% 程度である。VM において NIC を仮想化するオーバーヘッドは比較的大きく、仮想 NIC を 2 つ有する MR VM の CPU 使用率が大きくなってしまった。また、表 2 に示すように、ゲスト OS から `iperf` サーバノードに対する RTT も、NEMO の場合大きくなっている。仮に、オーバーヘッド低減する準仮想化ドライバを用いたとしても、MR VM のゲスト OS カーネルの IP スタック処理等は回避できないため、同等の傾向になると推測される。なお、MIPv6 トンネリングを行わずにゲスト OS が移動先ネットワークのアドレスを取得した場合、RTT は 0.37ms 程度である。Kagemusha による MIPv6 トンネリングのオーバーヘッドは、RTT に関して 0.07ms 程度の増加と非常に小さいことが確認できた。

Kagemusha のメモリ消費量（RSS 値）は 1.5MB 程度であり、MR として VM を起動する場合よりも遙かに少量で済んだ。

3.3 ライブマイグレーションとの連係動作

次に、提案機構をライブマイグレーションと連係させて評価実験を行った。ホームネットワーク上の Host A から移動先ネットワーク上の Host B へライブマイグ

表 1　Kagemusha と NEMO の性能比較（iperf）

	スループット	CPU 使用率
Kagemusha	84.4Mbps	14%
NEMO	87.8Mbps	92%

表 2　Kagemusha と NEMO の性能比較（ping6）

	RTT	RTT ジッタ
Kagemusha	0.44ms	0.028ms
NEMO	0.72ms	0.046ms

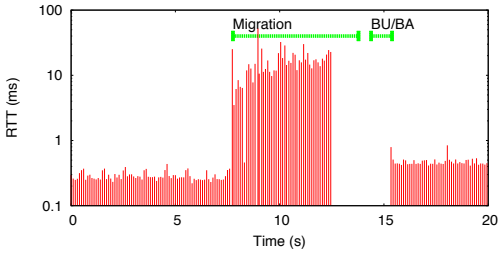

図 4　ライブマイグレーション前後の ping6 応答時間の推移（縦軸はログスケール）

レーションを行う。Host A 上に仮想ディスクイメージを配置し、ネットワークを介したストレージアクセスプロトコルの一つである NBD プロトコルを用いて Host A および Host B 上から接続した。仮想ディスクの接続には、我々が開発したストレージマイグレーション機能を備える xnbd-server[8] を用いたが、今回の実験では MIPv6 トンネリングに焦点を当てるためストレージマイグレーション機能は利用していない。VM のメモリサイズを 128MB に設定した。マイグレーション完了直後に Host B から HA に対して BU を発行するように、Kagemusha および Qemu に対して簡易なラッパースクリプトを作成した。qemu-kvm に備わっている一般的なプレコピー型のライブマイグレーションを行った。

本稿執筆時点で最新の安定版である qemu-kvm-0.14.1 においては、組み込みの NBD クライアント機能およびライブマイグレーション機能がいずれも IPv6 に対応していなかった。NBD 接続に関しては、ホスト OS の NBD ドライバおよび IPv6 に対応した xnbd-client を用いた。ライブマイグレーションに関しては、Qemu が備える、マイグレーションにともなうデータ転送を外部コマンドに委託する機能を用いた。Qemu のモニタインタフェースから、例えば `migrate EXEC: socat TCP6-CONNECT:[2001:db8:3000::2]:4444` などと命令し、IPv6 対応のプロキシコマンドである socat を起動している。

図 3 のルータ間のネットワークにノードを 1 台追加し、そのノードからゲスト OS に対する ping6 応答時間を計測した。0.1 秒間隔で Echo Request を発行しながら、ライブマイグレーションを行う。図 4 に計測結果を示す。パケットキャプチャの結果から、マイグレーションに使用した TCP コネクションの存在期間（図中 Migration 線）および BU および BA の送信時刻（図中 BU/BA 線の左端および右端）も図示した。

ネットワーク到達性が失われた期間（ダウンタイム）が約 3 秒弱存在するものの、その後到達性は回復し Echo Request が成功している。Kagemusha 経由時には、トンネリングの影響で 0.2ms 程度 RTT が増加している。マイグレーション中は 20ms 程度まで RTT が増加しているが、100Mbps のネットワーク帯域がほぼ飽和した影響や dirty page tracking 処理の負荷によるものであると推測する。

ダウンタイムについて詳しく見ると、マイグレーションが完了する以前から 1 秒程度発生している。プレコピー型ライブマイグレーションの最終段階においては、移動元の VM を停止し、VM の状態を移動先にコピーし、移動先で VM を再開する、という処理が発生する。この実験に用いた VM の場合、少なくともデバイスの状態など数十 MB のデータを最終段階でコピーする必要があるため、ルータ間のネットワークが 100Mbps でリンクアップされていた本実験では、1 秒程度の停止期間が存在してしまう。

次に、実際に BU が送信されるまでに、数百 ms 経過している。現状ラッパスクリプトがマイグレーションを開始する際には、移動元ホスト上の Qemu モニタインタフェースから migrate コマンドを発行している。そして、このコマンドの完了によってマイグレーションが完了したことを把握し、移動先ホストで起動している Kagemusha に対して BU の送信を依頼している。この処理は複数のスクリプト群からなるため、その処理に数百 ms 要していると推測する。

その後、HA が BU を受信してからトンネルを設定し BA を返信するまでに 1 秒程度要している。本実験では BU および BA メッセージは IPSec ESP で保護している。しかし、IPSec で保護しない場合も同等の時間を要した。この HA の処理時間は提案機構に依存したものではなく、提案機構で対応できる余地はない。しかし、MIPv6 をライブマイグレーションにおいて使用することを考えれば、HA においてトンネリングを開始するまでの時間はより短い方が望ましく、今後 HA の実装の改良方法を検討していきたい。

以上より、提案機構はライブマイグレーションと正しく連係動作することが確認できた。提案機構の導入にともなうダウンタイムの増加は数百 ms という十分小さい値であることがわかった。今後ラッパスクリプト等を改良すればさらに短縮できる可能性がある。

4 今後の課題

プロトタイプ実装は初期的な段階であり今後も開発を続けていく。現状、疑似ホームネットワークの実装において PATH MTU 処理を実装していない。例えば、VM が移動先ネットワークにおいて、ホームネットワーク上と同様に 1500B のパケットを送信してしまうと、IP6IP6 トンネル通過時にフラグメント化されてしまう。動作上問題がないことを確認しているものの、トンネリングの負荷を低減するためには、Kagemusha が PATH MTU に対応することが望ましい。提案手法では疑似ホームネットワークが物理ネットワークの状態を隠蔽してしまう。しかし、提案手法においても、既存の準仮想化ドライバを拡張するなどすれば、ネットワークの状態をある程度ゲストに対して通知できると考えている。

疑似ホームネットワークにおいて、どの程度本来のホームネットワークの挙動をエミュレーションすべきなのか、という点については議論の余地があり、今後実際的な運用を通して検証していく。例えば、リンクローカルアドレスを用いたパケットは、CMIPv6 の仕様上トンネリングを行えないため、Kagemusha によって破棄される。疑似ホームネットワーク上では、ホームネットワークに存在するノードとリンクローカルアドレスを用いた通信はできない。しかしグローバルアドレスを用いた通信は依然として可能であり、運用上問題が生じることは希であると現状考えている。

現状、IPv6 ネイティブのネットワークは広く一般に普及しているとは言い難く、場合によっては、そのことが原因で提案機構を実環境で用いることが難しい可能性がある。そこで我々は Dual Stack MIPv6[9] に着目している。例えば、移動先ネットワークが IPv4 ネットワークのみしか提供しないしない場合でも、ホームネットワークに対するトンネリングが可能になる。今後提案機構を拡張し Dual Stack MIPv6 に対応させることを検討している。

また、提案機構を拡張し NEMO に対応させることも実装上は可能である。既存の実装と比べ大半をユーザランドで実装できるため、性能的には劣る反面、可搬性に優れたものになる可能性がある。

我々は先行研究においてストレージマイグレーション機構 [2] を開発した。VM の仮想ディスクを遠隔拠点に透過的に移動できる。この機構と本稿の提案機構を連携させて、より実際的な環境で広域 VM マイグレーションを評価する予定である。

5 関連研究

Mobile IP 関連技術を VM のライブマイグレーションに用いた研究について述べる。文献 [10] では、Xen のライブマイグレーションに対して NEMO が適用できることを確認した。VM が移動するたびに移動先ホスト OS の MR 機能を設定している。ホスト OS の MR 機能を利用しているため、通常の NEMO の実装ではトンネリング先は一つに限定される。あるホスト OS 上の各 VM がそれぞれ別々のホームネットワークへトンネリングすることは難しい。

文献 [11] では、MR の役割を果たす VM を用意して、MR 配下の VM に対してネットワーク透過性が提供できることを確認している。MR 配下の VM に加え、MR の役割を果たす VM もマイグレーションする必要がある。一方、我々の提案機構では、VM 一つからの柔軟に拠点間で再配置できることを目指し、サブネットを移動する NEMO ではなく CMIPv6 のトンネリングを、ユーザランドの軽量なプログラムのみでゲスト OS に対して透過的に提供している。

文献 [12] では、VM のゲスト OS に MIPv6 を導入して、ホームネットワークとは異なる移動先ネットワークへライブマイグレーションを行っている。ホスト OS 側で VPN 等を用意する必要がない。しかし、移動先での CoA アドレス取得および BU の送信および受信によって、少なくとも 2.5 秒程度のダウンタイムが生じたことを報告している。

文献 [13] では、Proxy Mobile IPv4 と同様の機構を Xen のハイパーバイザー向けに実装している。VM が他のノードに対して送信するパケットは、すべて HA に対してトンネリングされて、そこから宛先に対して送出される。ホスト OS の proxy arp 機能やパケットトンネリング機能を利用して実装されている。一方、我々は、普段は HA を経由せずに通信でき、一時的に遠隔拠点に移動したときのみ HA を経由して通信できる機構を求めている。我々の提案機構は CMIPv6 に基づく点で異なっている。

6 まとめ

本稿では、VM の広域ライブマイグレーションを実現すべく、ゲスト OS にとって透過的な MIPv6 トンネリング機構（Kagemusha）を提案した。提案機構は、ハイパーバイザーと連動して動作し、CMIPv6 のシグナリングやトンネリングをゲスト OS の代わりに行う。ゲスト OS に MIPv6 に関するプログラムを導入する必要は

なく、ゲスト OS にとって透過的に動作する。VM が移動元拠点に存在する場合には、HA を経由することなく直接通信できる。既存の HA や VMM を一切変更することなくそのまま利用できる。またユーザランドにおいて標準的な socket API のみ用いて簡潔に実装されている。プロトタイプ実装を用いて評価実験を行った。その結果、提案機構は HA と正しく通信でき、そのトンネリングオーバーヘッドはわずかであることが確認できた。また Qemu/KVM のライブマイグレーションと正しく連係して動作して、移動先ネットワークであってもゲスト OS に対して透過的にネットワーク到達性を提供できた。このときマイグレーションにともなうダウンタイムの増加はわずか数百 ms にとどまることが確認できた。

本研究において貴重なご意見を寄せて頂いた IIJ イノベーションインスティテュートの島慶一氏に深く感謝する。本研究は科研費（23700048）および CREST（情報システムの超低消費電力化を目指した技術革新と統合化技術）の助成を受けたものである.

参考文献

[1] 広渕崇宏, 横井威, 江原忠士, 谷村勇輔, 小川宏高, 中田秀基, 工藤知宏, 田中良夫, 伊藤智, 関口智嗣. 複数拠点にまたがる e-science アプリケーション環境構築を目的としたソフトウェア導入・管理機構. 電子情報通信学会和文論文誌, Vol. J93-D, No. 10, pp. 2197–2208, 2010.

[2] 広渕崇宏, 小川宏高, 中田秀基, 伊藤智, 関口智嗣. 仮想計算機遠隔ライブマイグレーションのための透過的なストレージ再配置機構. 情報処理学会論文誌:コンピューティングシステム, Vol. ACS26, pp. 152–165, Jul 2009.

[3] David B. Johnson, Charles E. Perkins, and Jari Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[4] Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu, and Pascal Thubert. Network Mobility (NEMO) Basic Support Protocol. RFC 3963 (Proposed Standard), January 2005.

[5] Sri Gundavelli, Kent Leung, Vijay Devarapalli, Kuntal Chowdhury, and Basavaraj Patil. Proxy Mobile IPv6. RFC 5213 (Proposed Standard), August 2008.

[6] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 164–177. ACM Press, 2003.

[7] Avi Kivity, Yaniv Kamay, Dor Laor, and Anthony Liguori. kvm: the Linux virtual machine monitor. In *Proceedings of the Linux Symposium*, pp. 225–230. The Linux Symposium, 2007.

[8] Takahiro Hirofuchi. xNBD. `http://bitbucket.org/hirofuchi/xnbd/`.

[9] Hesham Soliman. Mobile IPv6 Support for Dual Stack Hosts and Routers. RFC 5555 (Proposed Standard), June 2009.

[10] 島慶一. 移動通信機能 nemo bs を用いたゲスト計算機のライブマイグレーション. *Internet Infrastructure Review*, Vol. 005, , Nov 2009.

[11] 石橋尚武, 岡本慶大, 島慶一, 関谷勇司. Wide クラウド wg 2010 年度活動報告. Technical Report wide-tr-cloud-report-2010-00.pdf, WIDE, Jan 2011.

[12] Eric Harney, Sebastien Goasguen, Jim Martin, Mike Murphy, and Mike Westall. The efficacy of live virtual machine migrations over the internet. In *Proceedings of the 2nd international workshop on Virtualization technology in distributed computing*, pp. 1–7. ACM Press, 2007.

[13] Qin Li, Jinpeng Huai, Jianxin Li, Tianyu Wo, and Minxiong Wen. HyperMIP: Hypervisor controlled mobile ip for virtual machine live migration across networks. In *Proceedings of the 11th IEEE High Assurance Systems Engineering Symposium*, pp. 80–88. IEEE Computer Society, 2008.