モバイルアプリケーション向け 移動透過通信フレームワークの実装と評価

上醉尾 一真[1]　鈴木秀和[1]　内藤克浩[2]　渡邊晃[1]

概要：モバイルアプリケーションの発展に伴い，移動しながら通信を行うことやモバイル端末同士で直接通信を行いたいという要求が増加している．一方で現在のインターネットは，NAT 越え問題や互換性のない IPv4 ネットワークと IPv6 ネットワークの混在により，エンド端末への接続性を確保することが困難である．著者らは，IPv4/IPv6 混在環境において確実な接続性の確保と，通信中のネットワーク切り替えを可能とする移動透過性を同時に実現する NTMobile（Network Traversal with Mobility）を提案してきた．従来の NTMobile の実装方式ではインストールのために端末の root 権限が必要となるため，一般ユーザが所有するモバイル端末において気軽に利用することが難しく，一般ユーザへの普及が課題であった．本稿では，アプリケーションレベルで NTMobile の移動透過通信を実現することにより，IPv4/IPv6 混在環境に適したモバイルアプリケーションの開発基盤となる移動透過通信フレームワークを提案する．提案手法のプロトタイプを用いたモバイルアプリケーションを市販の Android スマートフォンおよび iPhone へ実装し，IPv4 ネットワークと IPv6 ネットワーク間の通信および，移動が可能なことを確認した．

Implementation and Evaluation of Mobile Communication Framework for Mobile Applications

Kamienoo Kazuma[1]　Suzuki Hidekazu[1]　Naito Katsuhiro[2]　Watanabe Akira[1]

1. はじめに

スマートフォンやタブレットなどの高性能なモバイル端末の普及や無線技術の発展により，モバイル端末向けのインターネットサービスやモバイルアプリケーションの需要が増加している．一方で，モバイルアプリケーションの多くは従来の固定通信で用いられているクライアント・サーバ型の通信システムを適用しており，モバイル端末同士で直接通信を行うようなアプリケーションが発展していない．これは，現在のインターネットにおいて，エンド端末への接続性を確保することが難しいためである．現在インターネットで主に使用されている IPv4 では，グローバル IP アドレスの枯渇が深刻な問題となっており，すべてのモバイル端末に対してグローバル IP アドレスを割り当てることができない．そのため，IPv4 ネットワークでは NAT（Network Address Translation）を導入してプライベートネットワークを構築することが一般的であり，CGN（Carrier Grade NAT）のようにキャリアレベルでも NAT が導入され始めている [1]．NAT が導入された環境においては，グローバルネットワーク側の端末からプライベートネットワーク側の端末に対する接続性を確保できない，NAT 越え問題と呼ぶ課題があり，エンドツーエンドの接続性というインターネット本来の理念を損なう要因となっている．また，現在のインターネットは IPv4 から IPv6 への過渡期にあり，互換性のない IPv4 と IPv6 が混在した環境が広まりつつある．これらのことから，モバイル端末同士で直接通信を行うためには NAT 環境や IPv4 と IPv6 が混在した環境において，モバイル端末への接続性を確保することが必要である．

[1] 名城大学大学院理工学研究科
Graduate School of Science and Technology, Meijo University, Aichi, 462–8502, Japan
[2] 三重大学大学院工学研究科
Graduate School of Engineering, Mie University, Mie, 514–8507, Japan

これまでに，NAT 越え問題を解決するための技術や IPv4 ネットワークと IPv6 ネットワーク間の接続性を確保する技術が提案されてきた [2–6]．しかし，既存技術の多くは，NAT やプライマリ DNS（Domain Name System）サーバ等の特定の装置に改造が必要になることや，トンネリングを行うためにエンド端末への改造が必要になること，NAT 越え問題の解決と IPv4 ネットワークと IPv6 ネットワーク間の通信を同時に実現することを想定していないなどの課題がある．

また，スマートフォンには 3G や Wi-Fi，WiMAX など複数の無線インタフェースが搭載されており，必要に応じてインタフェースを切り替えて通信を行うことができる．しかし，IP ネットワークでは通信端末のインタフェースに割り当てられた IP アドレスを用いて通信を管理しているため，インタフェースやネットワークの切り替えに伴い IP アドレスが変化すると通信を継続することができない．このような問題を解決する技術を移動透過性技術と呼び，現在までに様々な移動透過性技術が提案されている [7–12]．一方で，既存の移動透過性技術の多くは IPv6 ネットワークを想定しており，IPv4 ネットワークへの適用が検討されている技術であっても，NAT が導入された環境においては移動や通信が制限されることや，経路が冗長になるという課題がある．IETF（Internet Engineering Task Force）では IPv4/IPv6 混在環境において移動透過性を実現する DSMIPv6（Dual Stack Mobile IPv6）が標準化されているが，IPv4 ネットワークにおける冗長経路の問題や HA（Home Agent）を分散設置できないことなどが課題となっている．また，これらの移動透過性技術は Android OS を搭載したスマートフォン（以後 Android スマートフォン）や iPhone などの現在普及しているモバイル端末へ実装されておらず，実装するためには OS やカーネルの改造が必要となるため，一般ユーザが気軽に利用することが困難である．

著者らは，IPv4 ネットワークと IPv6 ネットワークが混在した環境において確実な接続性を確保し，移動透過性を実現する NTMobile（Network Traversal with Mobility）を提案してきた [13–15]．NTMobile は NAT 越えの技術を兼ね備えており，IPv4/IPv6 混在環境において，NAT に改造を加えることなく NAT 配下の移動端末（以後 NTM 端末）に対する接続性を確保することができる．NTMobile では NTM 端末に仮想 IP アドレスを割り当て，アプリケーションが仮想 IP アドレスに基づいた通信を行うことにより，ネットワークの切り替えに伴う実 IP アドレスの変化を隠蔽し，アプリケーション間の通信を継続する．また，NAT の有無や IPv4 ネットワークと IPv6 ネットワーク間の通信など，通信端末が接続しているネットワークに応じて最適な経路でトンネルを構築し，アプリケーションが生成したパケットを転送する．NTM 端末間に構築されるトンネルは，特定の状況を除きエンドツーエンドで構築されるため，経路が冗長になりにくいという特徴を持つ．NTMobile はすでに Android スマートフォンへ実装しており，IPv4/IPv6 混在環境において有効性を確認済みである．しかし，従来の NTMobile の実装手法では，スマートフォンへカーネルモジュールおよびデーモンプログラムを実装するため，一般ユーザが気軽に利用することが難しい．

本稿では，アプリケーションレベルで NTMobile による移動透過通信を実現することにより，一般ユーザが所有するモバイル端末において，IPv4 ネットワークと IPv6 ネットワーク間の通信および移動を実現する移動透過通信フレームワークを提案する．モバイルアプリケーション開発者は移動透過通信フレームワークを用いることにより，モバイル端末間において直接通信を行うようなアプリケーションを容易に実現することができる．また，NTMobile にて用いる DC（Direction Coordinator）や RS（Relay Server）と呼ぶ装置は，DNS サーバのようにインターネット上に分散設置し，様々なアプリケーションが共有することができる．そのため，大規模なサーバ群を用意することなく，ビデオ会議やネットワーク型のゲームなど，リアルタイム性が要求されるアプリケーションを低コストで開発することができる．

また，移動透過通信フレームワークのプロトタイプを利用したモバイルアプリケーションを開発し，市販の Android スマートフォンおよび iPhone においてハンドオーバ処理の動作検証および性能評価を行った．

以下，2 章で関連技術の概要と課題を述べ，3 章で NTMobile について概説する．4 章にて移動透過通信フレームワークの概要および実装，5 章にて性能評価および課題について述べ，6 章でまとめる．

2. 関連技術

本章ではアプリケーションレベルの実装で移動透過性を実現する技術の概要と課題について述べる．文献 [10] では，SIP（Session Initiation Protocol）を用いることにより移動透過性を実現する，SIP モビリティが提案されている．SIP はリアルタイムアプリケーションの制御プロトコルであり，IP アドレスの変化を通信相手に通知することによりセッションを継続する機能を搭載している．SIP モビリティではアプリケーションが SIP による通信を行い，ネットワークが切り替わった際にはセッション継続機能により通信が継続される．SIP モビリティでは OS やカーネルに対して特別なモジュールを実装する必要が無いため，市販のモバイル端末においても容易に実現することができると考えられる．一方で，SIP モビリティでは UDP による通信のみを想定しており，TCP セッションを継続できないことが課題となっている．

SIP モビリティの課題を解決した技術として，All-SIP モ

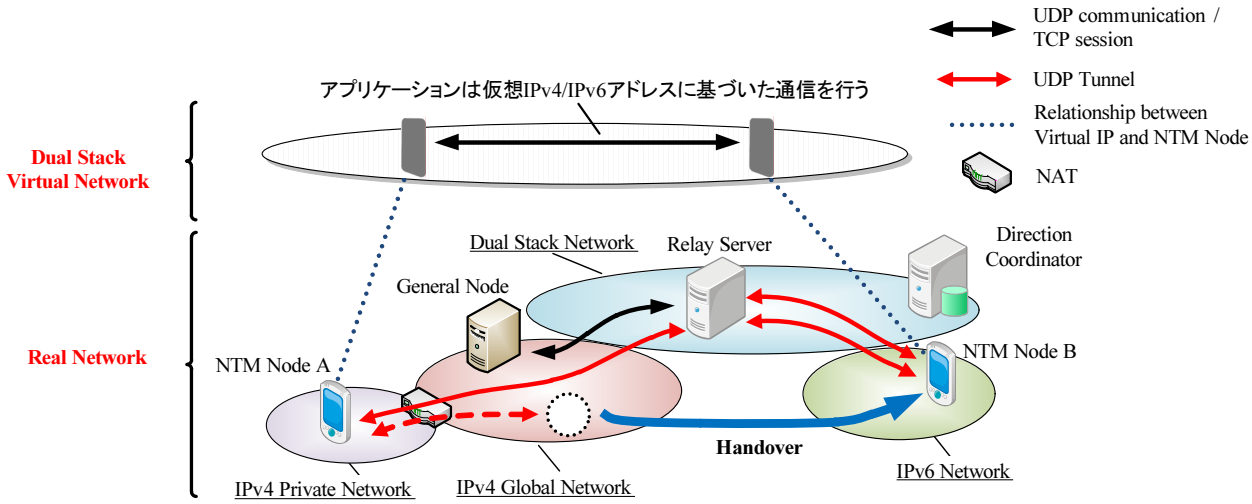


図 1　NTMobile の概要
Fig. 1　Overview of NTMobile.

ビリティが提案されている [11]．All-SIP モビリティでは，アプリケーションは仮想インタフェースに割り当てられた仮想 IP アドレスに基づいた通信を行うことにより，通信中にネットワークを切り替えても TCP セッションを継続することができる．一方で，仮想インタフェースより送信されたパケットはユーザ空間へ実装された SMC（Session and Mobility Contoroller）モジュールにおいてカプセル化されるため，カーネル空間にてカプセル化する技術と比較してスループットが低下しやすいという課題がある．また，仮想インタフェースおよび SMC の実装が必要となるため，一般ユーザが市販のスマートフォンにおいて気軽に利用することが難しい．

文献 [12] では，動的に通信ソケットを切り替えることにより移動透過性を実現する，MobileSocket が提案されている．MobileSocket では，アプリケーションは OS が提供している通信 API（Application Programming Interface）の代わりに，MobileSocket が提供するソケット通信 API を用いて通信を行う．通信中にネットワークが切り替わった際には，MobileSocket により通信ソケットが再生成され，自動的にコネクションが再構築される．これにより，アプリケーションはネットワークが切り替わっても通信を継続することができる．MobileSocket はモバイル端末へ実装されていないものの，アプリケーション実装の変更のみで移動透過性を実現することができるため，一般ユーザはアプリケーションをインストールするだけで気軽に移動透過通信を利用することができる．一方で，NAT 越え問題の解決や IPv4 ネットワークと IPv6 ネットワーク間における接続性に関する検討が行われておらず，実際のネットワーク環境においては接続性の確保に課題がある．

3. NTMobile

3.1 概要

本章では NTMobile の概要について説明する．NTMobile では NTM 端末に実際のネットワークに依存しない仮想 IP アドレスを割り当て，アプリケーションは仮想 IPv4 アドレスまたは仮想 IPv6 アドレスのどちらかに基づいた通信を行う．これにより，アプリケーションはネットワークの切り替えや通信経路上の NAT，実 IPv4 ネットワークおよび実 IPv6 ネットワークの違いに影響されることなく，自由に通信を行うことができる．なお，仮想 IP アドレスに基づくパケットは，NTM 端末間に構築される UDP トンネルによって送信される．この UDP トンネルは特定の状況を除きエンドツーエンドで構築されるため，通信端末は常に最適な経路でトンネル通信を行うことができる．

NTMobile の概要を図 **1** に示す．NTMobile は NTM 端末，DC および RS によって構成される．DC や RS はデュアルスタックネットワークに設置し，ネットワークの規模に応じて複数台設置することができる．

- NTM 端末
  NTM 端末は移動先のネットワークから割り当てられる実 IP アドレスと，DC から割り当てられる仮想 IP アドレスの 2 種類のアドレスを保持している．仮想 IP アドレスはネットワークに依存しないアドレスであり，NTM 端末が接続先のネットワークを切り替えても変化しない．NTM 端末には常に IPv4 と IPv6 の仮想 IP アドレスが割り当てられており，アプリケーションは仮想 IPv4 アドレスまたは仮想 IPv6 アドレスのどちらかに基づいた通信を行う．これにより，NTM 端末が通信中にネットワークを切り替えても，アプリ

ケーションは通信を継続することができる．

- DC（Direction Coordinator）
  DC は仮想 IP アドレスの割り当て管理や，NTM 端末に対してトンネル構築などの指示を出す装置である．NTM 端末に割り当てられる仮想 IP アドレスは一意なアドレスであり，各 DC は自身に割り当てられたアドレス空間から重複が起きないよう割り当てを行う．また，DC はデータベースを用いて NTM 端末のアドレス情報を管理している [16]．DC のデータベースには NTM 端末の FQDN（Fully Qualified Domain Name），実 IPv4 アドレス，実 IPv6 アドレス，仮想 IPv4 アドレス，仮想 IP6 アドレス，NAT の外側の実 IPv4 アドレスなどが登録されている．
- RS（Relay Server）
  RS は，異なる NAT 配下に位置する NTM 端末同士が通信を行う場合や，NTMobile 非対応端末と通信を行う場合など，特定の状況において通信を中継する装置である．また，図 1 における NTM 端末 A と移動後の NTM 端末 B のように，通信端末が異なるアドレスファミリのネットワークに接続している場合には，プロトコルの違いから直接通信を行う事ができない．そのため，NTM 端末 A と NTM 端末 B は，デュアルスタックネットワークに設置した RS との間にトンネルを構築し，RS を経由した通信を行う．RS はインターネット上に分散設置することが可能であり，中継負荷や経路の冗長性を考慮してトンネル構築時に最適な RS を選択することができる．なお，異なる NAT 配下に位置する NTM 端末同士が通信する場合であっても，経路最適化を行うことにより，RS 経由のトンネル通信経路をエンドツーエンドのトンネル通信経路へ切り替えることができる [17]．

DC と各端末は信頼関係があることを前提としており，NTMobile で使用されるメッセージは各端末間で共有している暗号鍵を用いて暗号化および MAC（Message Authentication Code）が付加される．また，NTM 端末間や NTM 端末と RS の間で行われるトンネル通信は，トンネル構築時に DC より配布される共通鍵を用いて暗号化および MAC が付加される．

3.2 NTM 端末の実装と課題

NTMobile はすでに Linux PC および Android スマートフォンへ実装しており，IPv4/IPv6 混在環境において有効性を確認済みである [15]．図 **2** に NTM 端末の実装構成を示す．NTM 端末はトンネル構築処理を行うデーモンプログラム（以後 NTM デーモン），仮想 IP アドレスに基づくアプリケーションパケットのカプセル化や暗号化などの処理を行うカーネルモジュール，および仮想インタフェースを実装することにより動作する．カーネルモジュールでは，Linux カーネルの機能である Netfilter を用いることにより DNS の名前解決処理や仮想 IP アドレス宛のパケットをフックする．NTM 端末は名前解決要求パケットをフックすると，NTM デーモンによるトンネル構築処理を実行する．トンネル構築完了後，NTM デーモンは通信相手の仮想 IP アドレスを記載した名前解決応答パケットを生成し，DNS リゾルバを経由してアプリケーションへ渡す．これにより，アプリケーションは仮想 IP アドレスを通信相手の IP アドレスとして認識し，仮想 IP アドレスに基づいた通信を開始する．仮想 IP アドレスに基づく通信パケットはカーネルモジュールにてカプセル化および暗号化され，NTM 端末間に構築された UDP トンネルによって通信相手に送信される．NTM 端末はカプセル化されたパケットを受信すると，カーネルモジュールにて復号およびデカプセル化し，抽出した仮想 IP パケットを仮想インタフェースより受信する．以上により，NTM 端末のアプリケーションは仮想 IP アドレスに基づいた通信を行う．

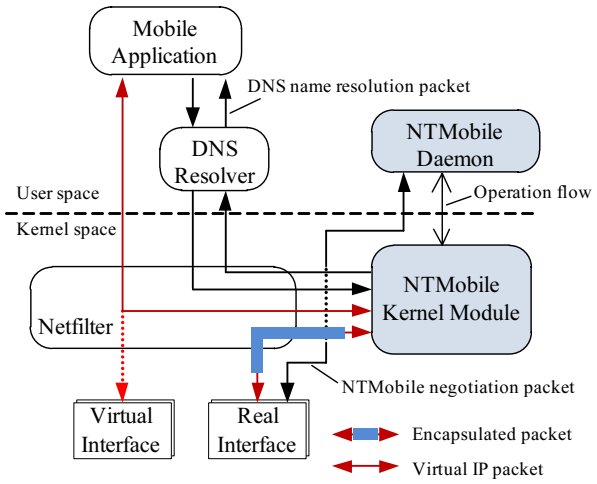


図 **2** カーネルモジュールを用いた NTM 端末の実装構成
Fig. 2 Module configuration of NTM node using kernel module.

NTMobile ではカーネルモジュールを用いることにより，DNS による名前解決処理を行うアプリケーションであれば，どのようなアプリケーションでも移動透過通信を実現することができる．一方で，市販のスマートフォンにカーネルモジュールを実装するためには，root 権限の取得やカーネルの再構築などが必要であり，一般ユーザが NTMobile を気軽に利用することは難しい．そのため，現在市販されているスマートフォンに NTMobile を適用することが難しく，一般ユーザへの普及が課題となっている．また，iPhone のように Linux 以外のカーネルを搭載しているスマートフォンへ実装することができないという課題がある．

4. 移動透過通信フレームワーク

4.1 概要

本稿では，NTM 端末の機能をアプリケーションレベルのフレームワークとして実装することにより，市販の Android スマートフォンや iPhone において，IPv4 ネットワークと IPv6 ネットワーク間の通信や移動透過性を実現する移動透過通信フレームワークを提案する．

図 **3** に移動透過通信フレームワークの概要を示す．提案手法では，IPv4/IPv6 混在環境における相互接続性の確保や NAT 越え問題の解決，移動透過性を実現するために NTMobile を利用する．そのため，3 章にて説明した NTMobile のネットワーク構成と同様に，実ネットワークへ DC および RS を分散設置する．NTM 端末は，モバイルアプリケーションをインストールした Android スマートフォンや iPhone などの市販のモバイル端末である．提案手法では，従来の NTM 端末とは異なり，各モバイルアプリケーションに NTM 端末の機能を持った移動透過通信フレームワークを実装する．これにより，ユーザは自身のモバイル端末へモバイルアプリケーションをインストールするだけで，NTMobile による移動透過通信を利用することができる．

図 **4** に移動透過通信フレームワークのモジュール構成を示す．移動透過通信フレームワークは，トンネル構築処理を行うネゴシエーションモジュール，カプセル化や暗号化処理を行うパケット操作モジュールなどによって構成される．また，移動透過通信フレームワークはアプリケーションに対して，仮想 IP アドレスによる通信を行うための仮想ソケット通信 API を提供する．アプリケーションは OS が提供している通信 API の代わりに，移動透過通信フレームワークが提供する API を用いて通信を行う．これにより，従来の NTM 端末においてカーネルモジュールが行っていた名前解決処理の検出や，仮想 IP アドレスに基づくパケットのカプセル化，暗号化処理をアプリケーションレベルで実現する．なお，移動透過通信フレームワークが提供する仮想ソケット通信 API は，BSD（Berkeley Software Distribution）ソケットのような一般的なソケット通信 API と互換性を持っているため，アプリケーションは通常のソケット通信 API と同様の手順で利用することができる．

- ネゴシエーションモジュール（Negotiation module）
  ネゴシエーションモジュールはアプリケーションにより登録 API が呼び出された際に，NTM 端末がネットワークから取得した実 IP アドレスを DC へ登録する．また，仮想ソケット通信 API により名前解決処理が呼び出された場合やネットワークが切り替わった際に，NTMobile によるトンネル構築処理を行う．

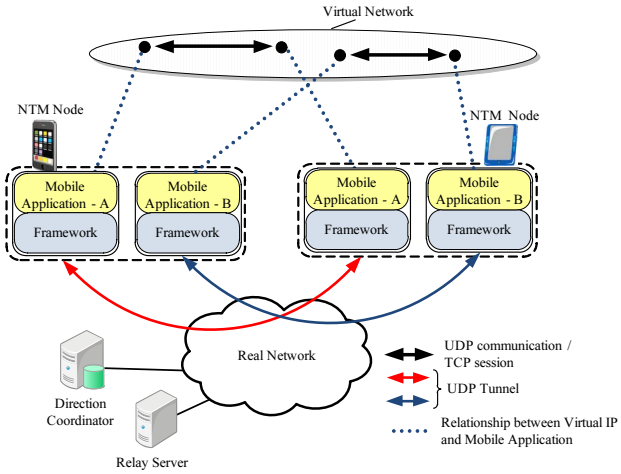


図 **3** 移動透過通信フレームワークの概要
Fig. 3 Overview of Mobile Communication Framework.

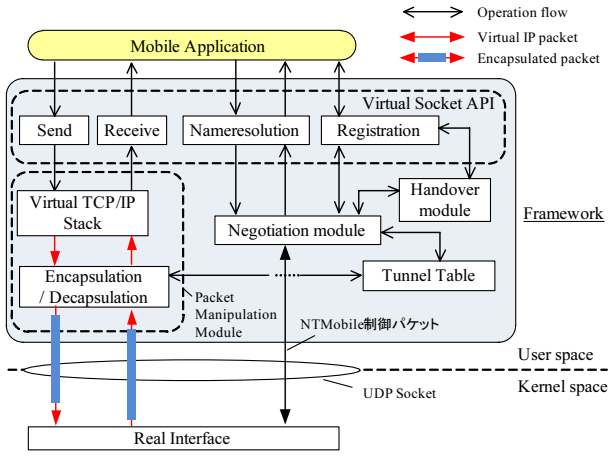


図 **4** 移動透過通信フレームワークのモジュール構成
Fig. 4 Module configuration of Mobile Communication Framework.

- パケット操作モジュール
  （Packet manipulation module）
  パケット操作モジュールは仮想 IP パケットを処理する仮想 TCP/IP スタックを保持しており，仮想 IP パケットの送受信処理を行う．アプリケーションは移動透過通信フレームワークが提供する仮想ソケットおよび仮想ソケット通信 API を用いることにより，仮想 IP アドレスに基づいたソケット通信を行う．また，パケット操作モジュールは仮想 IP アドレスに基づくパケットのカプセル化・デカプセル化処理，および暗号化・復号処理を行う．
- トンネルテーブル（Tunnel Table）
  トンネル構築処理によって構築されたトンネル情報を管理するテーブルである．通信相手の仮想 IP アドレス，トンネルの構築先の実 IP アドレスやポート番号，暗号化に用いる共通鍵などが登録されている．

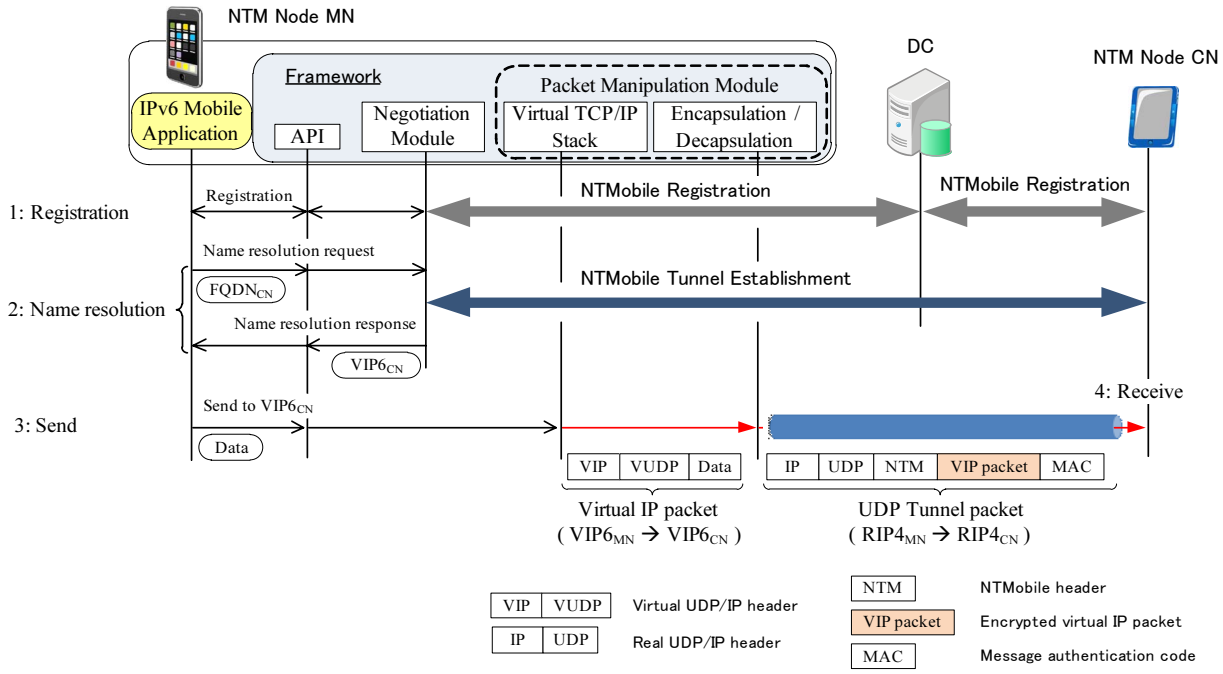


図 5　移動透過通信フレームワークの通信手順

Fig. 5　Communication procedure of Mobile Communication Framework.

- ハンドオーバモジュール（Handover module）
  iOS における SystemConfiguration framework や Android における ConnectivityManager など，OS が提供する通信管理機能によりネットワークの接続状況の変化を検出する．ネットワークの接続状況が変化した際には，ネゴシエーションモジュールによるトンネルの再構築処理，および DC への実 IP アドレス登録処理を実行する．

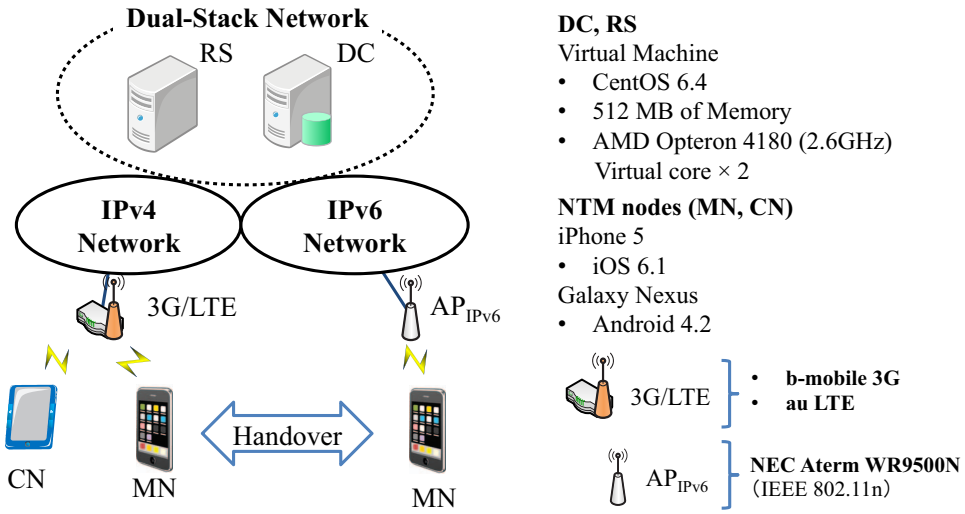
4.2　通信処理手順

図 5 に IPv4 グローバルネットワークへ接続した NTM 端 MN（Mobile Node）が CN（Correspondent Node）と，移動透過通信フレームワークを利用したモバイルアプリケーションによる仮想 IPv6 の UDP 通信を行う様子を示す．以後，端末 X の FQDN を FQDN$_X$，仮想 IPv6 アドレスを VIP6$_X$，実 IPv4 アドレスを RIP4$_X$，実 IPv6 アドレスを RIP6$_X$ とする．また，MN と CN の間に構築されたトンネルを識別するための Path ID を PID$_{MN-CN}$ とする．

(1) 登録処理
MN と CN のアプリケーションは起動時に登録 API を呼び出すことにより NTMobile の登録処理を行う．これにより，MN と CN がネットワークより取得した実 IP アドレスが DC へ登録される．なお，移動透過通信フレームワークを用いることによりアプリケーションが行う特別な処理は登録処理のみであり，その後は通常のソケット通信 API を用いる場合と同様の手順で処理する．今回の例では，MN と CN のアプリケーションは仮想ソケット通信 API により仮想ソケットを生成し，他の端末からの通信を待機する．

(2) 名前解決処理
MN のアプリケーションは CN へ通信を開始する前に，仮想ソケット通信 API により FQDN$_{CN}$ の名前解決処理を実行する．このとき，ネゴシエーションモジュールによるトンネル構築処理が実行され，文献 [16] の手順により，MN と CN 間にトンネルが構築される．トンネル構築完了後，MN のアプリケーションは名前解決処理の結果として，VIP6$_{CN}$ を取得する．

(3) 送信処理
名前解決処理が完了した後，MN のアプリケーションは仮想ソケット通信 API により仮想ソケットを生成し，VIP6$_{CN}$ 宛にデータを送信する．パケット操作モジュールは，仮想ソケットに割り当てられたポート番号と自身の仮想 IP アドレス VIP6$_{MN}$ を元に，VIP6$_{CN}$ 宛の仮想 IP パケットを生成する．その後，パケット操作モジュールは宛先の仮想 IP アドレス VIP6$_{CN}$ を検索キーにトンネルテーブルを検索し，該当エントリに従い仮想 IP パケットをカプセル化する．カプセル化の際には，PID$_{MN-CN}$ を記載した NTM ヘッダを仮想 IP パケットに付加し，暗号化および MAC の付加を行う．その後，カプセル化パケットをトンネルの構築先である RIP4$_{CN}$ へ送信する．送信時には OS が提供するソケット通信 API により生成した UDP ソケットから送信されるため，アプリケーションが VIP6$_{CN}$ へ送信した仮想 IP パケットは実 IP アドレスに基づい



	DC-RS	Galaxy Nexus				iPhone 5			
		b-mobile 3G		Wi-Fi		au LTE		Wi-Fi	
		DC	RS	DC	RS	DC	RS	DC	RS
RTT [ms]	0.3	446.0	508.3	10.3	10.4	279.3	172.9	31.9	33.4

図 6　測定環境
Fig. 6　Evaluation environment.

た UDP パケットによりカプセル化され，$\mathrm{RIP4_{CN}}$ へ送信される．

(4) 受信処理
CN のアプリケーションのパケット操作モジュールはカプセル化パケットを受信すると，カプセル化パケットの NTM ヘッダに記載された $\mathrm{PID_{MN-CN}}$ を検索キーにトンネルテーブルを検索し，該当エントリに従って復号およびデカプセル化を行う．パケット操作モジュールは，抽出した仮想 IP パケットの宛先ポート番号が割り当てられた仮想ソケットへ仮想 IP パケットのペイロード部分を渡す．これにより，CN のアプリケーションは仮想ソケット通信 API により生成した仮想ソケットより MN のアプリケーションが送信したデータを受信する．

以上により，MN と CN のアプリケーションは仮想 IP アドレスに基づいた通信を行うことができる．

4.3　ハンドオーバ処理

ハンドオーバモジュールは，モバイル端末の移動や Wi-Fi の On/Off などによりネットワークが切り替えられた際に，ハンドオーバ処理を実行する．ハンドオーバモジュールは，ネゴシエーションモジュールを呼び出し，アプリケーション起動時と同様の手順により，ネットワークから取得した実 IP アドレスを DC へ登録する．これにより，NTM 端末に対する到達性を確保する．

また，ネゴシエーションモジュールは通信開始時の同様の手順により CN との間にトンネルを再構築する．トンネル構築時には，MN と CN が接続しているネットワークに応じて，DC が最適なトンネル通信経路を指示し，MN と DC 間にトンネルが再構築される．MN と CN のアプリケーションは仮想 IP アドレスに基づいた通信を行っているため，実 IP アドレスが変化してもその影響を受けることなく通信を継続することができる．

5.　評価と課題

5.1　性能評価

移動透過通信フレームワークのプロトタイプを用いたモバイルアプリケーションを iPhone および Android スマートフォンへインストールし，ハンドオーバ処理の動作検証および性能評価を行った．なお，プロトタイプの仮想 TCP/IP スタックには UDP による通信機能のみを実装した．

図 6 に測定環境および各装置間の平均 RTT（Round-Trip Time）を示す．なお，RTT は ping を 100 回実行することにより測定した．DC および RS は仮想マシンにより構築し，MN と CN には Apple iPhone 5 および Samsung Galaxy Nexus を用いた．$\mathrm{AP_{IPv6}}$ はルータ機能を無効にして，一般的なアクセスポイントとして動作させた．なお，今回使用した Galaxy Nexus では，DHCP（Dynamic Host Configuration Protocol）による IPv4 アドレスの取得処理に失敗した際に AP から強制的に切断されてしまい，IPv6 ネットワークへ接続することができなかった．そのため，

Galaxy Nexus が AP_{IPv6} へ接続する際には,Wi-Fi インタフェースに IPv4 アドレスとして 0.0.0.0 を静的に設定した．iPhone 5 および Galaxy Nexus が AP_{IPv6} へ接続する際には，IEEE 802.11n にて接続し，暗号化および認証機能には WPA/WPA2-PSK（AES）を使用した．また，Wi-Fi 無効時には，iPhone 5 は au の LTE，Galaxy Nexus は b-mobile の 3G による IPv4 モバイルネットワークへ接続する．

iPhone 5 と Galaxy Nexus を MN および CN とし，MN が CN との通信中に接続先のネットワークを切り替えることにより発生する通信断絶時間を測定した．CN は常に IPv4 モバイルネットワークへ接続しておき，MN の接続先ネットワークを次のように手動で切り替えた．

Case 1 IPv4 モバイルネットワーク経由での通信中に Wi-Fi を有効にし，接続先を AP_{IPv6} へ切り替える．

Case 2 AP_{IPv6} 経由での通信中に Wi-Fi を無効にし，接続先を IPv4 モバイルネットワークへ切り替える．

このとき，MN を iPhone 5，CN を Galaxy Nexus とした場合を Case 1-1 および Case 2-1，MN を Galaxy Nexus，CN を iPhone 5 とした場合を Case 1-2 および Case 2-2 とする．

測定用に UDP パケットを 50 ms 間隔で送信するモバイルアプリケーションを作成し，MN から CN へ通信を行った．NTMobile のハンドオーバ処理に要した時間を測定するために，処理ごとのタイムスタンプを出力するようプログラムを追加し，その差分からハンドオーバ処理に要した時間を算出した．また，通信状況を観測するために，MN および CN へ LAN アナライザツールである tcpdump をインストールした．

図 **7** にハンドオーバに伴う通信断絶時間を 10 回測定した平均値を示す．Case 1-1 では 2476.2 ms，Case 1-2 では 2178.5 ms 間通信が断絶された．iPhone 5 および Galaxy Nexsu では，AP_{IPv6} への認証・接続が完了した後に 3G ネットワークから切断され，通信断絶時間が発生した．AP_{IPv6} への接続完了後に IPv6 Stateless Address Autoconfiguration [18] による IPv6 アドレスの取得処理が行われ，Case 1-1 では 355.0 ms，Case 1-2 では 1860.9 ms を要した．この処理時間には，移動透過通信フレームワークがハンドオーバを検出するまでの処理遅延が含まれる．IPv6 アドレス取得後，NTMobile によるハンドオーバ処理が実行され，Case 1-1 では 2064 ms，Case 1-2 では 284 ms を要した．その後，Case 1-1 では 62.3 ms，Case 1-2 では 32.9 ms 後にアプリケーションによる通信が再開された．

Case 1-2 では，IPv6 アドレス取得処理に Case 1-1 の約 5 倍の時間を要した．IPv6 アドレス取得時には DAD（Duplicate Address Detection）による重複確認が行われ，生成した IPv6 アドレスが有効になるまでの間に待機時間が発生する．Case 1-2 では，Galaxy Nexus から DAD による NS が送信されてから，NTMobile によるハンドオーバ

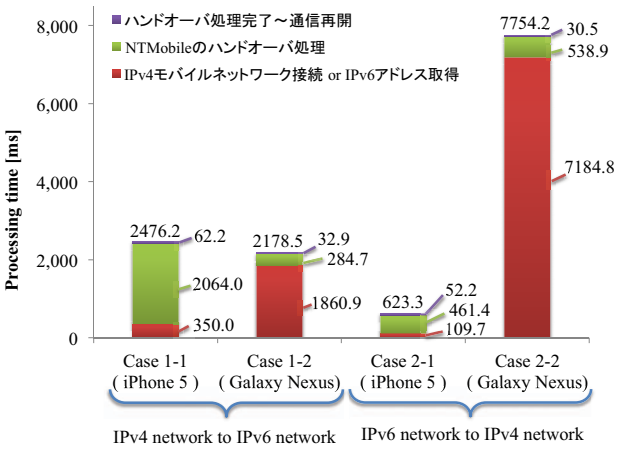


図 **7** ハンドオーバに伴う通信断絶時間の測定結果

Fig. 7 Results of suspended time by handover.

処理が実行されるまでに約 1.2 秒を要していた．文献 [19] によると，Linux では DAD が実行された際に 1 秒程度待機するよう実装されており，タイムクリティカルなアプリケーションにおいては致命的な遅延になりかねないとされている．また，ハンドオーバ検出処理は iOS と Android が提供する通信管理機能を用いているため，iPhone 5 と Galaxy Nexus では異なる実装を行っている．これらの実装の違いにより，処理時間に差が生まれたと考えられる．

Case 1-1 では NTMobile によるハンドオーバ処理に Case 1-2 の約 7 倍の時間を要した．NTMobile によるハンドオーバ処理では，MN と DC および RS 間にてパケット交換処理が行われる．MN が NTMobile によるパケットを RS および DC へ送信する際には，MN から RS および DC をターゲットとした NS（Neighbor Solicitation）が 1 回ずつ送信されていた．このとき，Case 1-1 では iPhone 5 が NS を送信する前に 900ms 程度の待機時間が発生しており，NTMobile による処理は 230 ms 程度で完了していた．Case 1-2 において Galaxy Nexus が NS を送信する前に待機時間が発生していないことから，iOS と Android OS における IPv6 通信処理の実装の違いにより発生した遅延であると考えられる．

Case 2-1 では 623.3 ms，Case 2-2 では 7754.2 ms 間通信が断絶された．Case 2-1 では，Wi-Fi を無効にしてからモバイルネットワークへ接続されるまでに 109.7ms を要していた．一方，Case 2-2 では 7184.8 ms を要し，通信断絶時間の 92%を占めていた．この処理時間にはハンドオーバモジュールがハンドオーバを検出するまでの処理遅延が含まれる．モバイルネットワークへの接続完了後，NTMobile によるハンドオーバ処理が行われ，Case 2-1 では 461.4 ms，Case 2-2 では 538.9 ms で完了した．その後，Case 2-1 では 52.2 ms，Case 2-2 では 30.5 ms 後にアプリケーションによる通信が再開された．

測定結果より，ハンドオーバ時に行う NTMobile による

処理は 538.9ms 以下で完了するものの，スマートフォンにおける IPv6 通信処理の実装の違いにより，1 秒以上の遅延が発生することがわかった．また，Galaxy Nexus では Wi-Fi を無効にしてから IPv4 モバイルネットワークへ接続されるまでに 7 秒以上を要し，通信断絶時間の 92%を占めていることがわかった．リアルタイム性を要する音声通信や動画通信の利用を想定すると，通信断絶時間は短いことが望ましく，IPv6 通信処理およびモバイルネットワークへの接続処理における遅延の削減が必要である．

5.2 従来の実装手法との比較

カーネルモジュールを用いた従来の NTMobile の実装手法と提案手法の比較を表 **1** に示す．従来手法では，市販の Android スマートフォンにおいて動作検証済みであるが，デーモンプログラムおよびカーネルモジュールを実装するために root 権限の取得およびカーネルの再構築を行っている．そのため，一般ユーザが気軽に利用することが難しく，一般ユーザへ普及させるためにはモバイル端末の開発時に NTMobile を実装する必要がある．それに対して提案手法では，ユーザは移動透過通信フレームワークにより開発されたモバイルアプリケーションを所有するスマートフォンへインストールするだけで利用することができる．

従来手法では，Netfilter や Netlink などの Linux カーネル特有の機能を利用しているため，Linux カーネル以外を適用したプラットフォームへ実装することができない．それに対して提案手法では，ほとんどの機能を OS に依存しない形で実装しているため，クロスプラットフォームへの対応を容易に行うことができる．

一方，提案手法ではアプリケーションが移動透過通信フレームワークの提供する API を用いて通信を行う必要がある．従来手法では，アプリケーションに対して変更を加える必要が無いため，既存のアプリケーションにおいて移動透過通信を行うことができる．また，どちらの実装手法においてもハンドオーバ時に発生する通信断絶時間が課題となっており，従来手法においては複数のインタフェースを用いたシームレスハンドオーバ実現手法を提案している [20]．文献 [20] による手法では，Android OS が提供するフレームワークの修正が必要となるため，提案手法においては適用することができず，ハンドオーバ時に発生する通信断絶時間の削減が課題となる．

5.3 今後の検討課題

提案手法による移動透過通信フレームワークを用いることにより現在市販されているモバイル端末へ，IPv4/IPv6 ネットワーク間の通信や移動など，NTMobile による移動透過通信を提供することができる．一方で，アプリケーションが移動透過通信を行うためには仮想ソケット通信 API を用いて通信を行う必要があるため，アプリケーションが利用可能な通信プロトコルは仮想 TCP/IP スタックが提供している通信プロトコルに限定される．そのため，5.1 節におけるプロトタイプ実装では，アプリケーションは UDP 以外の通信を行うことができない．lwIP（Lightweight TCP/IP stack）[*1] および uIP（micro IP）[*2] では，ユーザランドにおける簡易的な TCP/IP の実装が公開されており，UDP だけでなく TCP や ICMP の処理を利用することができる．今後はこれらの実装を用いることにより，仮想 TCP/IP スタックにおいて TCP や ICMP などの通信プロトコルを提供することができると考えられる．

表 **1** 従来の NTMobile の実装手法との比較
Table 1 Comparison with conventional methods.

	従来手法	提案手法
市販のスマートフォンへの適用	△	○
一般ユーザへの普及	△	○
クロスプラットフォームへの対応	×	○
既存アプリケーションの対応	○	×
シームレスハンドオーバの実現	○	×

提案手法ではアプリケーションに移動透過通信フレームワークを実装する必要があるため，既存のアプリケーションへ適用することが難しい．また，OS が提供している通信 API を利用した場合は，移動透過通信フレームワークによる機能を適用することができないことが課題である．Android OS のバージョン 4.0 以降では VpnService[*3] と呼ぶ VPN（Virtual Private Network）通信用の機能が提供されており，VpnService を利用したアプリケーションは他のアプリケーションが送信したパケットをフックすることができる．これにより，従来の NTM 端末においてカーネルモジュールが行っていたパケットのフック処理をアプリケーションのみで実現することができ，モバイル端末の改造やアプリケーションへ移動透過通信フレームワークを実装せずに，NTMobile による移動透過通信を実現することができると考えられる．一方で，仮想インタフェースより送信されたパケットをユーザ空間へ戻し，カプセル化処理を行うため，提案手法と比較してスループットの低下が大きくなると考えられる．また，Android OS 特有の機能を用いるため，iPhone や Windows Phone などモバイル端末では利用することができず，クロスプラットフォーム対応が課題となる．

6. おわりに

本稿では，IPv4/IPv6 混在環境において相互接続と移動透過性を実現する NTMobile を用いた移動透過通信フレームワークを提案した．移動透過通信フレームワークを用い

[*1] http://savannah.nongnu.org/projects/lwip/
[*2] http://dunkels.com/adam/
[*3] http://developer.android.com/reference/android/net/VpnService.html

ることにより，アプリケーション開発者は大規模なサーバ群を用意することなく，リアルタイム性の高いアプリケーションやモバイル端末間で直接通信を行うアプリケーションを容易に実現することができる．また，移動透過通信フレームワークのプロトタイプを用いたモバイルアプリケーションを市販の Android スマートフォンおよび iPhone へインストールし，動作検証および性能評価を行った．その結果，NAT や IPv4/IPv6 ネットワークの違いに影響されることなく，接続性の確保と移動透過性を実現可能なことを確認した．ハンドオーバ時にはネットワーク接続処理に起因する通信切断時間が発生するため，ネットワーク接続処理の高速化が必要である．

謝辞　本研究の一部は，総務省戦略的情報通信研究開発推進制度（SCOPE）の支援を受けて実施された．

参考文献

[1] Jiang, S., Guo, D. and Carpenter, B.: An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition, RFC 6264, IETF (2011).
[2] Rosenberg, J.: Interactive Connectivity Establishment (ICE): A Protocol for Net- work Address Translator (NAT) Traversal for Offer/Answer Protocols, *RFC 5245,IETF* (2010).
[3] 鈴木秀和，宇佐見庄五，渡邊　晃：外部動的マッピングにより NAT 越え通信を実現する NAT-f の提案と実装，情報処理学会論文誌，Vol. 48, No. 12, pp. 3949–3961 (2007).
[4] 宮崎　悠，鈴木秀和，渡邊　晃：端末の改造が不要な NAT 越え通信システム NTSS の提案と評価，情報処理学会論文誌，Vol. 51, No. 9, pp. 1234–1241 (2010).
[5] Huitema, C.: Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs), *RFC 4380,IETF* (2006).
[6] Tsirtsis, G. and Srisuresh, P.: Network Address Translation - Protocol Translation (NAT-PT), *RFC 2766,IETF* (2000).
[7] Le, D., Fu, X. and Hogrefe, D.: A Review of Mobility Support Paradigms for the Internet, *IEEE Communications Surveys*, Vol. 8, No. 1, pp. 38–51 (2006).
[8] Soliman, H.: Mobile IPv6 Support for Dual Stack Hosts and Routers, *RFC 5555,IETF* (2009).
[9] 相原玲二，藤田貴大，前田香織，野村嘉大：アドレス変換方式による移動透過インターネットアーキテクチャ，情報処理学会論文誌，Vol. 43, No. 12, pp. 3899–3897 (2002).
[10] Schulzrinne, H. and Wedlund, E.: Application-layer mobility using SIP, *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 4, No. 3, pp. 47–57.
[11] 康雄向井，春弥宮島，亮　張，秀樹　林，輝也藤井：アクセス網非依存 All-SIP モビリティ技術の実装と評価，電子情報通信学会技術研究報告. IN, 情報ネットワーク，Vol. 107, No. 222, pp. 25–30 (2007).
[12] Okoshi, T., Mochizuki, M., Tobe, Y. and Tokuda, H.: MobileSocket:Session Layer Continuous Operation Support for Java Applications, 情報処理学会論文誌，Vol. 41, No. 2, pp. 222–234 (2000).
[13] 鈴木秀和，上醉尾一真，水谷智大，西尾拓也，内藤克浩，渡邊　晃：NTMobile における相互接続性の確立手法と実装，情報処理学会論文誌，Vol. 54, No. 1, pp. 367–379 (2013).
[14] 内藤克浩，上醉尾一真，西尾拓也，水谷智大，鈴木秀和，渡邊　晃，森香津夫，小林英雄：NTMobile における移動透過性の実現と実装，情報処理学会論文誌，Vol. 54, No. 1, pp. 380–393 (2013).
[15] 上醉尾一真，鈴木秀和，内藤克浩，渡邊　晃：IPv4/IPv6 混在環境における NTMobile の実装と評価，マルチメディア，分散，協調とモバイル（DICOMO2012）シンポジウム論文集，Vol. 2012, No. 1 (2012).
[16] 細尾幸宏，鈴木秀和，内藤克浩，渡邊　晃：NTMobile における DNS 実装の変更が不要なデータベース型端末情報管理手法の検討，情報処理学会研究報告，Vol. 2012-MBL-64, No. 6, pp. 1–8 (2012).
[17] 納堂博史，鈴木秀和，内藤克浩，渡邊　晃：NTMobile における自律的経路最適化の提案，情報処理学会論文誌，Vol. 54, No. 1, pp. 394–403 (2013).
[18] Thomson, S. and Narten, T.: IPv6 Stateless Address Autoconfiguration, *RFC4862,IETF* (2007).
[19] Pongpaibool, P., Sotthivirat, P., Kitisin, S. and Srisathapornphat, C.: Fast Duplicate Address Detection for Mobile IPv6, *Proceedings of the 15th IEEE International Conference on Networks, ICON 2007*, pp. 224–229 (2007).
[20] 福山陽祐，上醉尾一真，旭　健作，鈴木秀和，内藤克浩，渡邊　晃：Android 端末における Wi-Fi/3G 間のシームレスハンドオーバの提案と実装，情報処理学会研究報告，Vol. 2012-MBL-65, No. 27, pp. 1–8 (2013).