平成28年度　卒 業 論 文

和文題目

NTMobileを用いたアプリケーション層における移動透過性の実現

英文題目

Mobility Realization in Application Layer Using NTMobile

情報工学科　渡邊研究室

(学籍番号: 130441001)

赤堀　蒼磨

提出日: 平成29年2月10日

名城大学理工学部

概要

スマートフォンやタブレット端末の普及に伴い, モバイル端末で移動透過性と通信接続性の必要性が高まっている. 著者らはこの二つの要点を同時に実現する NTMobile(Network Traversal with Mobility) を提案してきた. さらに NTMobile をアプリケーション層上で実現し, OS に依存せず動作する NTMobile のフレームワークを提案している. このフレームワーク版 NTMobile における移動透過性に関する部分についての実装および性能評価を行った. 本研究によりアプリケーション層において移動透過性の実現ができた. 処理時間を計測した結果, アドレス変化を検出するまでの時間が平均で 5.8 秒かかることが分かった. それに対してトンネル再構築にかかった時間は平均 0.1 秒弱である. この結果を踏まえて今後はアドレス変化検出の処理にかかる時間が短くする検討を行う必要がある.

目 次

第1章　はじめに

近年のスマートフォンやタブレット端末といったモバイル端末の普及により, どこにいても手軽にインターネットを利用することが可能となった. このような端末では 3G 回線, LTE 回線, Wi-Fi など様々なネットワークインタフェースを用いてインターネットを利用する. 利用者は必要に応じてこれらのインターフェースを切り替えて通信を行う. 3G から Wi-Fi, LET から Wi-Fi, Wi-Fi から別のアクセスポイントのように接続先が変化すると IP アドレスも変化する. IP アドレスは通信識別子の役割を果たしているため, IP アドレスの変化前に行っていた通信が変化後に継続できない. 近年は無料で使える公衆 Wi-Fi の普及が進んでおり, 上記のようなモバイル端末は頻繁に IP アドレスの変化が発生することが想定される.

現在ネットワークにはこれ以外にも様々な問題がある. それが IPv4 アドレスの枯渇が問題である. この問題の解決のために NAT を用いたローカルネットワークの活用と IPv6 アドレスが登場した.　NAT を用いたローカルネットワークを構築すると NAT 配下の端末からの通信開始を行うことができるが, NAT の外側から NAT 配下の端末へ通信開始を行うことができない. また IPv4 アドレスから IPv6 アドレスへの移行が完了していないため, 両者が混在した環境となっている. IPv4 アドレスと IPv6 アドレスには互換性がないため異なるバージョンの IP アドレス間での通信ができない. これらの問題の解決のため, 接続するネットワーク環境にかかわらず通信開始ができる通信接続性と, 通信中にネットワークを切り替えることができる移動透過性の実現が必要となっている.

我々はこれらの問題を同時に実現する技術として NTMobile(Network Traversal with Mobility) を提案している. [1–4] NTMobile はインターネット上に DC(Direction Coodinator) を設置する. NTMobile 機能を搭載した端末 (NTM 端末) 同士の通信は仮想 IP アドレスパケットを実 IP アドレスでカプセル化してトンネル通信を行う. その際使用する仮想 IP アドレスは DC によって配布される. DC は端末間のトンネル構築の指示も行う. IPv4・IPv6 間通信や異なる NAT 配下同士の端末間の通信には RS(Relay Server) と呼ばれる装置がパケットを中継する. このようにして, 現在のネットワークに存在する様々な制約を除去することができるシステムである.NTMobile ログイン時には AS(Account Server) にアクセスし, 端末が正規の端末であるか判定する.

NTMobile は現在 Linux カーネルでの動作を検証している. しかし Android や iOS 等のモバイル端末向けの OS では, カーネル操作には特殊な操作が必要である.　これらの操作は OS 配布者のサポート外となり, 一般ユーザが導入するのは困難である. そこで現在は NTMobile の実用化に向けたフレームワークの実装開発が行われている. この実装では全ての処理をアプリケーション層で行うため, カーネル空間で特殊な処理を行わず OS に依存しないシステムとなっている. 本稿ではフレームワークの移動透過性を実現している部分に着目してカーネルモジュール実装型 NTMobile の仕様上の問題の解決策について検討を行った. また, アドレス変化の検出, NTM 端末間のトンネル

再構築の処理の実装をおこない,Linux 上にて評価を行った.

第2章　NTMobile

2.1　NTMobile の概要

NTMobile のネットワーク構成を図 1 に示す. DC(Direction Coodinator) は, NTM 端末すべての位置情報を管理し, NTM 端末同士の通信を行う際には UDP トンネル構築の指示を行う. この機器は複数台設置可能な仕様となっているため, トラフィックがサーバの性能を上回っても対応可能となっている. AS(Account Server) は, NTM 端末が NTMobile ネットワークにログインする際に端末認証を行う. RS(Relay Server) は, 一般端末との通信時及び IPv4/IPv6 アドレス間で通信を行うとき, 異なる NAT 配下同士のようなエンドツーエンド通信ができない場合パケットの中継を行う.

NTMobile の通信は端末が移動しても変化しない仮想 IPv4/仮想 IPv6 アドレスを用いる. この仮想 IP アドレスは DC が配布する. 仮想 IP アドレスに対応した FQDN は AS が配布する. 通信開始時には,　相手の FQDN を指定して通信を行う. NTM 端末間の通信は, 実 IP アドレスで仮想 IP アドレスを UDP カプセル化を行うことによりトンネル通信を行う. 仮想 IP アドレスは端末が移動しても変化しないため, 通信中に実 IP アドレスが変化しても通信の継続を行うことができる. カプセルの内部は暗号化を行い通信経路にて盗聴されたとしても解読することができない. NTM 端末は定期的に DC および通信中の NTM 端末と Keepalive と呼ばれるパケットを送受信している. これにより NAT 配下の端末に対しても DC 側からのアクセスが可能となる. これらの機能を実装することにより通信接続性と移動透過性の実現を可能とする.

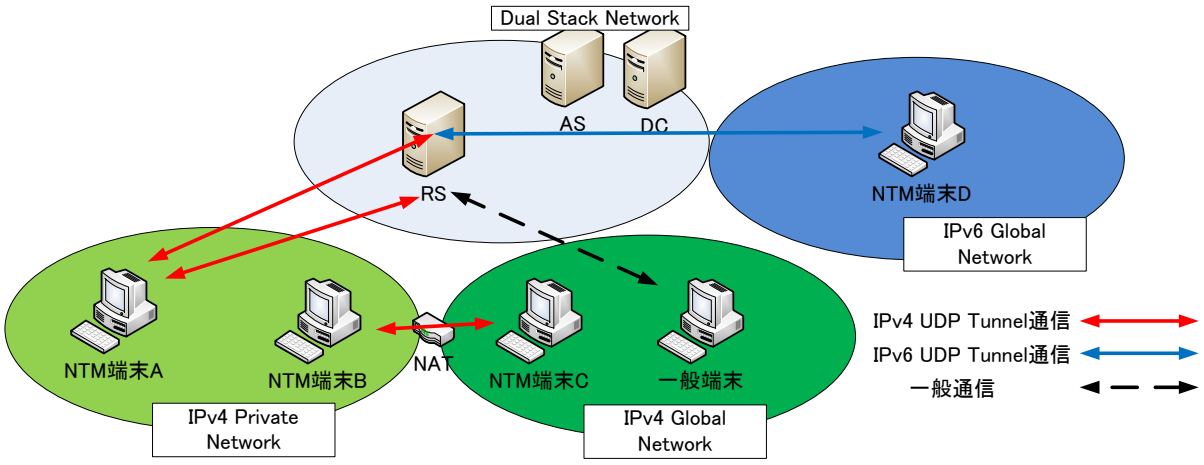

図 1　NTMobile のネットワーク構成

2.2 ログインシーケンス

図 2 にログインシーケンスを示す. このフェーズで NTMobile を利用するユーザが正規のユーザであるかどうかをチェックする. これにより NTMobile 通信の安全性を向上させる. 各パケットについて説明する. まず NTM 端末は, あらかじめ AS に登録しているメールアドレスとパスワードを AS に送信する. AS で Login Request パケットを受信したらその二つのペアが正しいか認証を行う. AS は MN と DC で利用される共通鍵情報が記載された Key Distribution パケットを DC へ送信する. AS は Key Distribution の応答パケットを受信したら NTM 端末へ Login Response パケットを送信する. 以上により NTMobile のログインが完了する. Key Distribution パケットは AS が生成した MN と DC との共通鍵を DC へ送信するパケットである. Login Response パケットは DC との共通鍵と MN の FQDN が格納されたパケットである. DC との通信の際はこの共通鍵を使用し, NTM 端末どうし通信にはこの FQDN を指定する.

2.3 通信介し時のシーケンス

このフェーズでは DC は今後トンネル構築の要求があった場合 DC 内にある各端末の情報に基づいてトンネル構築の指示を各端末へ行う. トンネル構築時には, エンドツーエンド通信が可能な場合とそうでない場合によって異なるシーケンスでトンネルを構築するためそれぞれ説明する. どちらの場合も最初に Direction Request パケットを DC へ送信する. DC 側でそのパケットを受信した際に仮想 IP アドレスと対になる Path ID を生成している. その後 Route Direction パケットにて Path ID を各端末へ配布する. NTM 端末間の通信は, この Path ID を用いて通信を確立する. DC は仮想 IP アドレスや実 IP アドレス, Path ID の関係を記述したトンネルテーブルで各端末の位置情報を管理する.

2.3.1 エンドツーエンド通信が可能な場合

図 3 に NTM 端末間のトンネル構築時のシーケンスを示す. Direction Request パケットには通信相手の FQDN を指定し DC へ送信する. DC は Direction Request 内に記載されている FQDN より

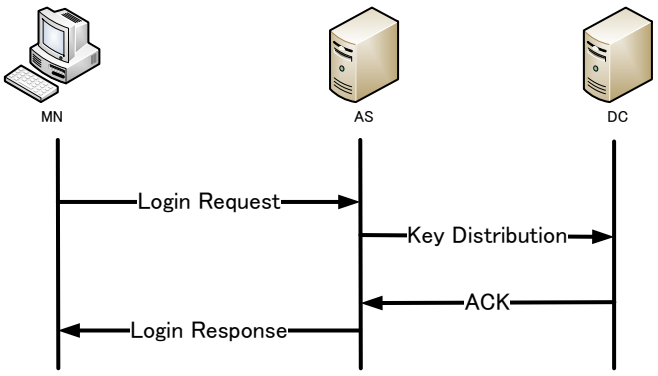

図 2 ログインシーケンス

自身の持つハッシュテーブルから CN の情報を参照する. その情報をもとに CN に対して MN の情報を付加した Route Direction を送信する. DC が CN から応答パケットを受信すると MN へ対して CN の情報を付加した Route Direction を送信する. MN は Route Direction より CN の情報を取得し CN へ対して Tunnel Request を送信する.

2.3.2 エンドツーエンド通信ができない場合

IPv4, IPv6 間で通信を行う場合, 両 NTM 端末が異なる NAT 配下に存在している場合エンドツーエンド通信ができないため RS 経由で通信を行う. その場合のシーケンスを図 4 に示す. DC が MN を受信し RS 経由で通信を行わないといけないと判定したら, RS に対して Relay Direction を送信する. DC が応答パケットを受信したら CN へ Route Direction を送信する. その後 MN へ Route Direction を送信する. CN は Route Direction を受信したら Hole Punching パケットを RS へ送信する. これにより NTM 端末 2 は RS との通信経路を確保する. MN は Route Direction を受信したら RS 経由で CN へ Tunnel Request を送信する. MN 側で Tunnel Response を受信したら一連のシーケンスは完了となる.

2.4 NTM 端末移動時のシーケンス

端末が移動した後通信中の NTM 端末とのトンネルを再構築する必要がある. エンドツーエンド通信が可能な場合と RS が必要な場合のシーケンスをそれぞれ図 5, 図 6 に示す. 初回通信時との違いは, DC へ対して Registration Request を送信している点である. このパケットには自身の端末情報を記載している. DC 側でこのパケットを受信したらパケット内に記載されている NTM 端末の移動後の情報をもとにデータベースの更新を行う. その後は通信開始時と同様, Route Direction にて DC が端末間のトンネル構築の指示を行う.

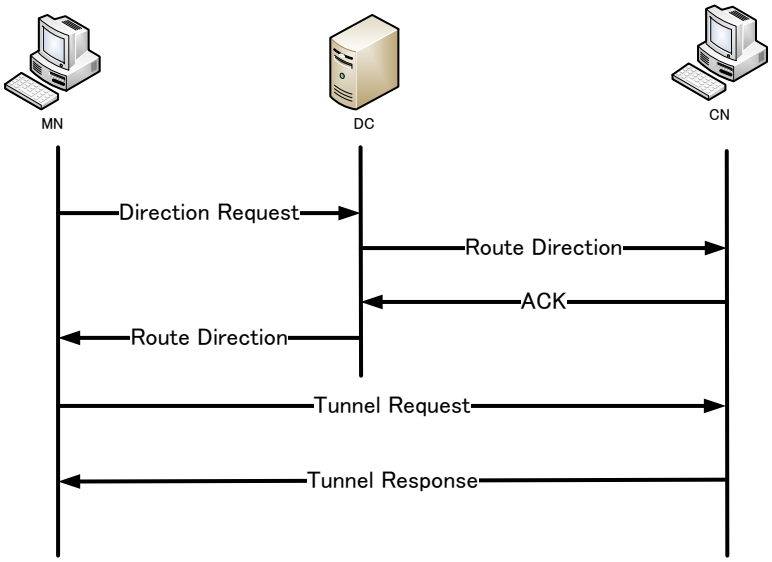

図 3 トンネル構築のシーケンス

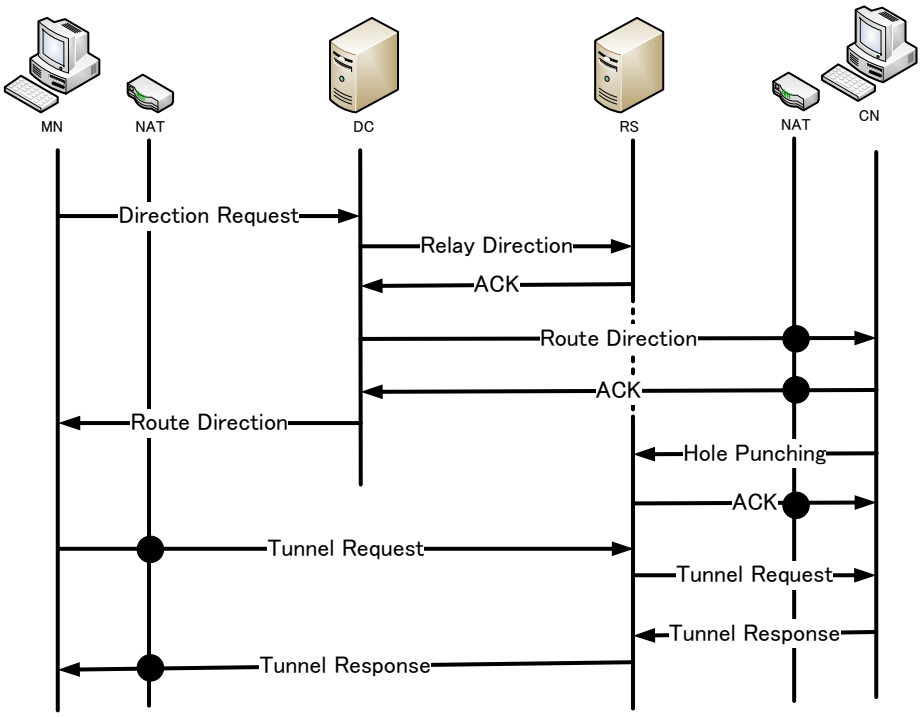

図 **4**　**RS** を用いたトンネル構築のシーケンス

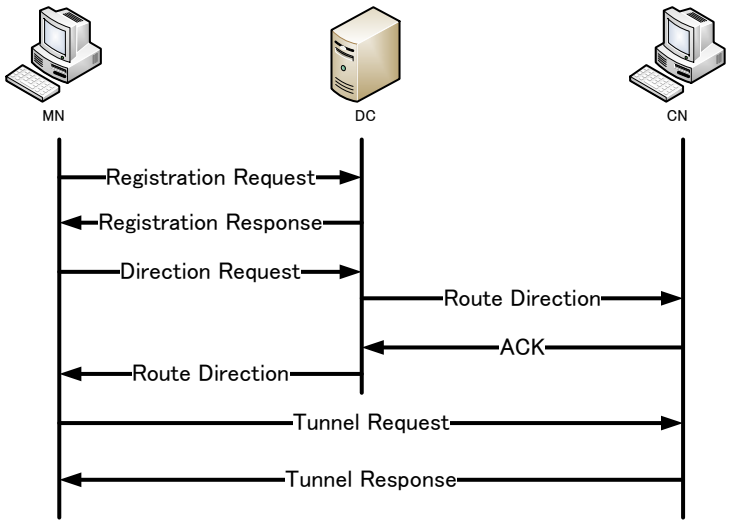

図 **5**　トンネル再構築のシーケンス

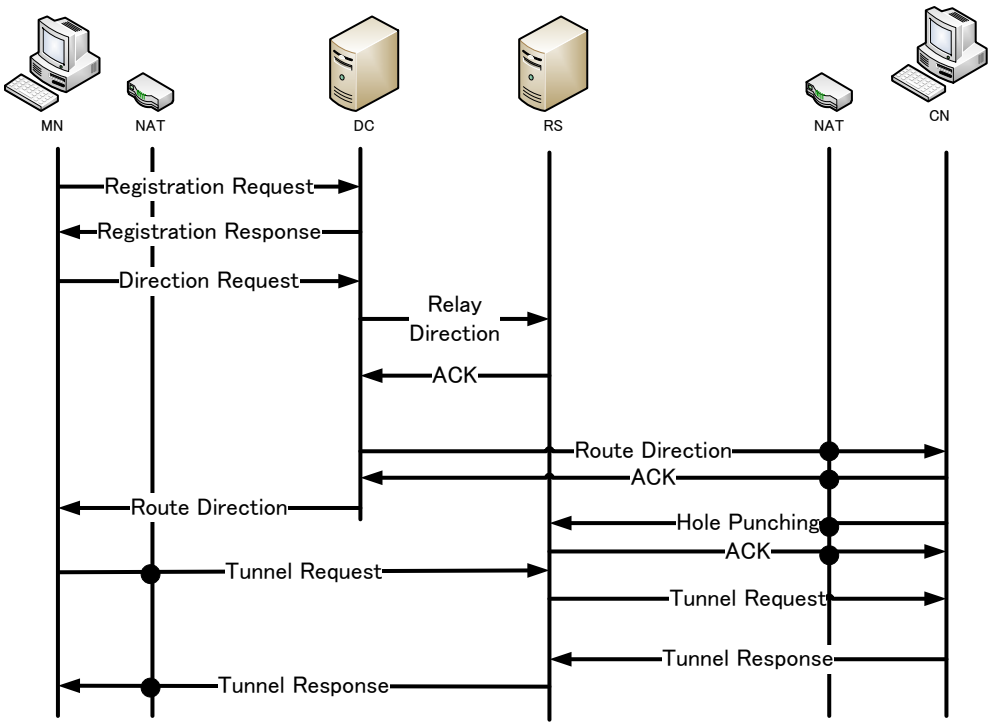

図 **6**　**RS** を利用する場合のトンネル再構築のシーケンス

第3章　NTMobileの実装方式と仕様に見直し

NTMobile にはいくつかの実装方式があるが, 検証済のカーネル実装型 NTMobile とフレームワーク組込型 NTMobile について解説する.

3.1　カーネルモジュール実装型 NTMobile

本実装方式はパケットの暗号化とカプセル化処理, IP アドレスの変化検出処理をカーネル空間で実装している.　このシステムの構成を図 7 に示す. アプリケーションがパケットを送信するときそのパケットを Netfilter でフックする. そのパケットを NTM Kernel Module にて暗号化及びカプセル化を行う.　もし, フックしたパケットが DNS 問い合わせのパケットであればアプリケーション層の NTM Deamon へ渡す. NTM Deamon では名前解決やトンネル構築指示などの処理を行い, これをアプリケーション空間に実装する. カーネルモジュール型におけるメリットは, カーネル空間で全パケットのフックを行うため既存のアプリケーションに手を加えることがなく NTMobile を利用できる点である. しかしカーネル空間の改造が可能な OS は限られている. 例えば Android OS の場合 root 権限が必要で一般ユーザが導入するのは困難であり, root 権限座取得できたとしてもセキュリティ面でも危険が伴う. iOS ではカーネルが暗号化されておりブラックボックスとなっている. さらにこの実装モデルで使用している Netfilter は Linux　カーネル特有の機能であるため Windows 等の OS では実装ができない.　Linux カーネルのバージョンがアップデートされるとそれに対応したシステムの開発が必要になる. これらの理由により現状本システムがサポートしているのは Linux OS の一部のカーネルバージョンのみとなっている. なお, 本実装方式ではアドレス変化時の処理については未実装であり, それに伴う排他制御についても実装していない.

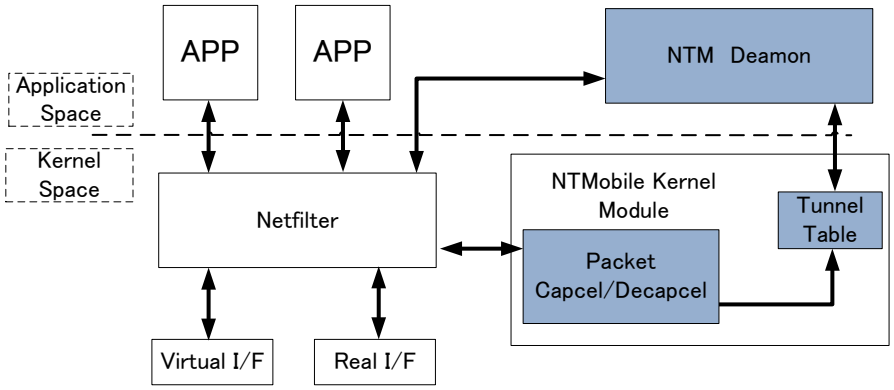

図 7　カーネルモジュール実装型 NTMobile の構成

3.2 フレームワーク組込型 NTMobile

Android や iOS 等のモバイル端末向けの OS において, NTMobile を一般ユーザが導入するには困難であるため, NTMobile の普及にはカーネルモジュール実装型 NTMobile はふさわしくない. そこでフレームワーク組込型 NTMobile を開発している. 本システムのモジュールの構成を図 8 に示す. NTMobile の機能を全てアプリケーション空間に移植する. カーネルモジュール実装型 NTMobile ではパケットの暗号化やカプセル化, 移動検出といった機能をアプリケーション空間に移植し, フレームワークとして提供する. そのため Android OS や iOS 等カーネルモジュール実装型では導入が困難であった OS においても NTMobile システムを利用することができる. パケットの暗号化とカプセル化は lwip(lightweight IP) [5] を利用することでアプリケーション空間での実装を実現した. 移動ネットワーク切り替え時の処理を実装するため実 IP アドレスと仮想 IP アドレス、Path ID の関係を記述したトンネルテーブルをダイナミックに書き換える必要がある. カーネルモジュール実装型では未実装であった, トンネルテーブル書き換え時にデータの整合性が取れなくなる状況を避けるための排他処理を実装する. トンネルテーブルの Key は仮想 IP アドレス, node ID, Path ID のどの値からも探索できるハッシュテーブルを利用している. このトンネルテーブルに通信相手の端末情報を格納し NTMobile 通信を行う際に参照する. この Key のどれか一つを指定すれば中にある通信相手の情報を取り出すことができる.

本システムのインタフェースは C 言語に準拠しており, 開発者が開発をしやすい環境を整えている. 例えば C 言語ではパケットを送信するとき send 関数を利用するが, NTMobile では引数は同じ ntm_send 関数というものを準備している. NTMobile のログイン処理や終了処理に関しては開発者に組み込んでもらう必要がある. さらに java や swift 等のプログラミング言語で NTMobile のラッパー関数を作成すれば Android や iOS のようなモバイル端末での動作が可能となる.

3.3 排他制御に係る仕様の見直し

これまでの NTMobile の仕様では Registration Request 受信時に毎回乱数を用いて DC で Path ID を生成していた. しかしこの生成方法ではハンドオーバ時に毎回 Path ID が変わってしまう. Path ID は仮想 IP アドレスと結びつけられたデータで通信識別子として利用されるため, 排他制御を行うには同一の値を使い続けないといけない. そこでこれまで DC で生成していた Path ID を初回トンネル構築時に NTM 端末側で生成し, Direction Request を送信するときに新たに Path ID を付加して送信する. アドレス変化時の Direction Request は初回に生成した Path ID を付加することによって同一の Path ID を使い続けることができる. この処理の変化前, 変化後のシーケンスについては図 9, 図 10 に示す. 以上により移動後も同一の通信として認識できるため, 排他制御を確実にが実現できる.

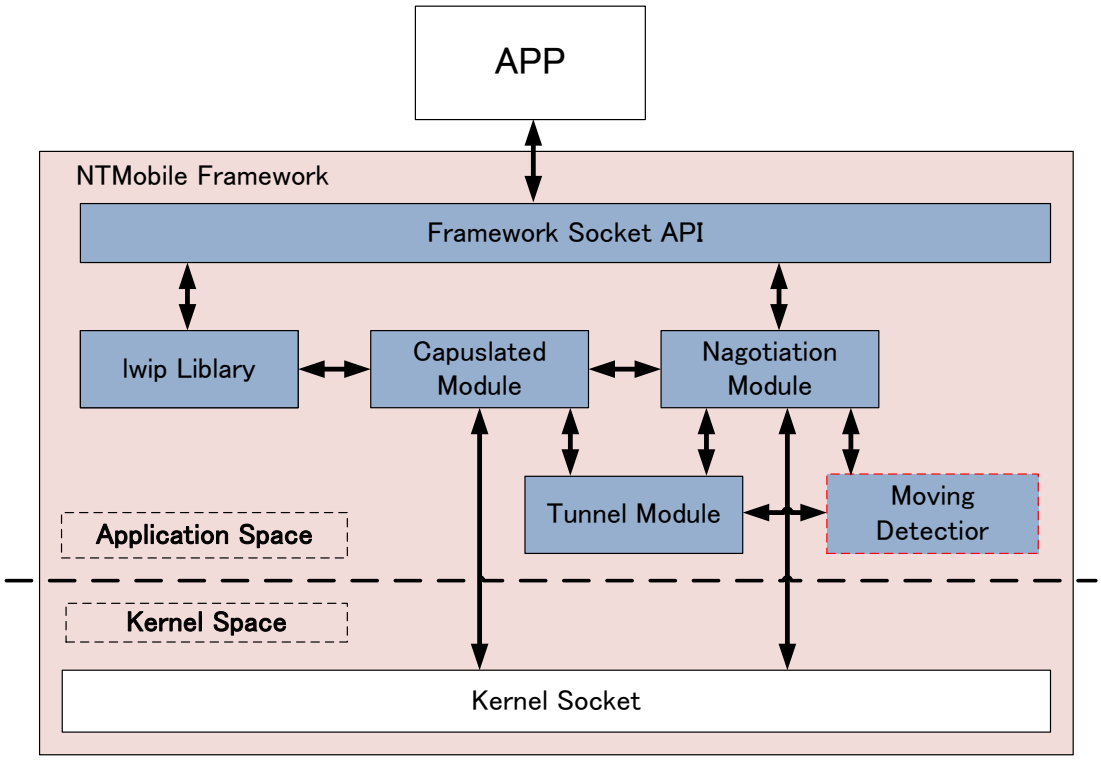

図 8　フレームワーク組込型 **NTMobile** の構成

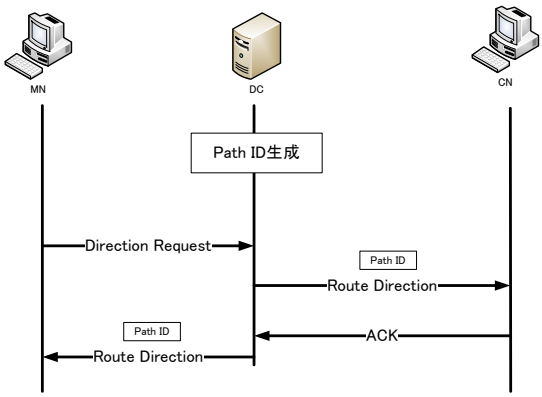

図 9　仕様変更前の **Path ID** 生成方法

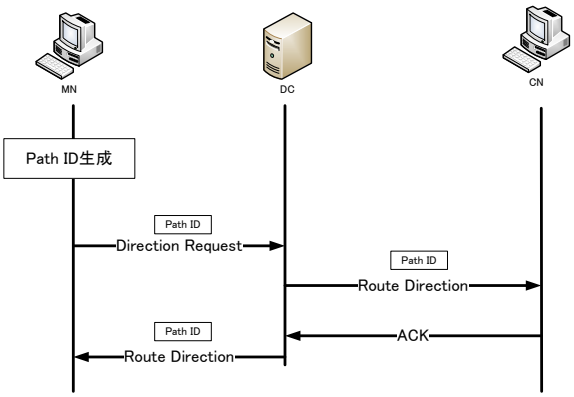

図 10　仕様変更後の **Path ID** 生成方法

第4章　実装と評価

フレームワーク組込型 NTMobile の移動に関する処理についての実装及び性能評価を実施した.

4.1　実装

移動検出および移動時の処理については図 8 の Moving Detector により実現する. タイマーにより IP アドレスを監視するスレッドを作成した. このスレッドを 1 秒毎に呼び出す. スレッド内のフローチャートを図 11 に示す. スレッドを生成したらローカル変数として定義されている Mutex をロックする. 次に現在の IP アドレスを取得する. NTMobile ログイン時のアドレス情報はグローバル変数に格納されており, その情報と現在の情報との比較を行う. そこでアドレスの変化が確認されらトンネル再構築の処理を行う. DC や NTM 端末と行っている Keepalive を停止し, アドレス変化前に作成された通信ソケットの初期化を行う. その後図 5, 図 6 のシーケンスに従いトンネルの再構築を行う. その際トンネルテーブルの各データに定義されている Mutex をロックする. 相手端末側ではトンネル再構築処理が完了したらトンネルテーブルの更新を行う.　トンネルテーブル更新の際に排他制御を実装し, トンネルテーブル内のデータの整合性を確保する. 以上の処理が終了し, Keepalive を再開した後 Mutex の解除を行う.

4.2　動作検証

図 12 に動作検証した環境を示す. また MN および CN のマシン構成については表 1 のとおりである. DC については VMware Player を用いてホスト PC 上に構築した. この仮想マシンの構成は表 2 に示す. MN を DC と同じネットワーク上から NAT 配下へ移動し, その後トンネル再構築処理が正常に完了し, 通信が継続していることを確認した.　これによりアプリケーション層において移動透過性の実現ができた.

表 1　MN, CN の構成

	MN	CN
CPU	intel Core i7-2660CPU 3.40GHz	intel Core i7-860CPU 2.80GHz
memory	7.9GB	7.9GB
OS	ubuntu 14.04LTS 32bit	ubuntu 14.04LTS 32bit

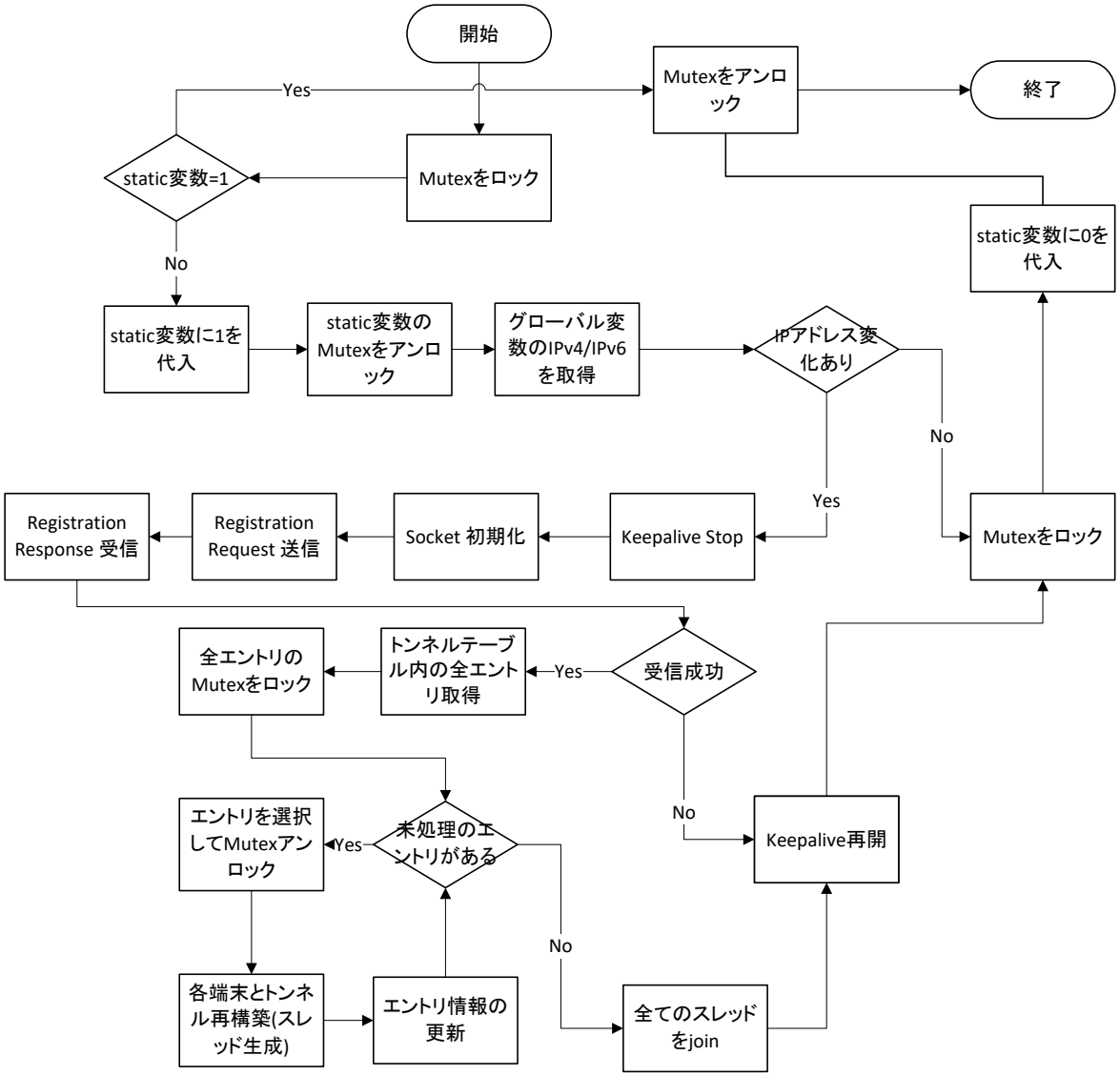

図 **11**　トンネル再構築時の処理手順

表 **2**　仮想マシンとそのホストマシンの構成

	DC(仮想 OS)	ホストマシン
CPU	intel Core i7-6700CPU 3.40GHz	intel Core i7-6700 3.4GHz
memory	2.0GB	8.0GB
OS	ubuntu 12.04LTS 32bit	Windows7 64bit

4.3　評価

フレームワーク組込型 NTMobile の移動に関する性能評価にを実施した.

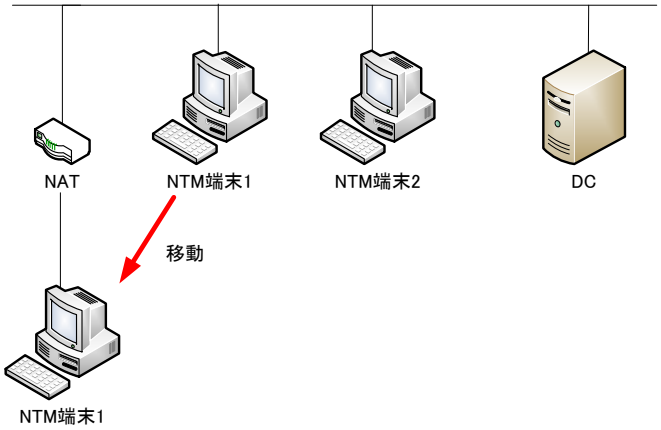

図 12　フレームワーク組込型 NTMobile の構成

4.3.1　評価方法

実験環境は図 12 において示したものと同じ環境とする. NTM 端末 1 が NAT 配下へ移動後 MN より DHCP Discover パケットを送信する時刻を計測のスタートとして NTM 端末 2 から Tunnel Response パケットを受信する時間を計測の終了時間として 10 回計測を行い平均値を算出した. パケットの観測には wireshark を利用する. また全体の処理時間をアドレス取得時間, アドレス変化検出時間, ハンドオーバ処理時間の 3 つに分割し各フェーズの平均値も算出した. それぞれの時間の定義は以下のとおりである.　また図 13 にシーケンスとともに示す.

- アドレス取得時間
  NTM 端末 1 が NAT へ DHCP Discover を送信してから DHCP ACK が返ってくるまでの時間
- アドレス変化検出時間
  ACK 受信時から DC へ Registration Request を送信するまでの時間
- ハンドオーバ処理時間
  DC へ Registration Request を送信してから NTM 端末 2 から Tunnel Response を受信するまでの時間

4.3.2　結果

表 3 に実験結果を示す. この表を見ると全体処理時間の平均が 7.002 秒に対してアドレス変化取得時間の割合が 6.911 秒であり全体処理時間の約 98.7 %がアドレス変化検出時間であることが分かる. アドレス取得時間やハンドオーバ処理時間が非常に短いことが分かったので, 今後アドレス変化検出ついて問題点を調査し時間を短くしていくことが必要である.

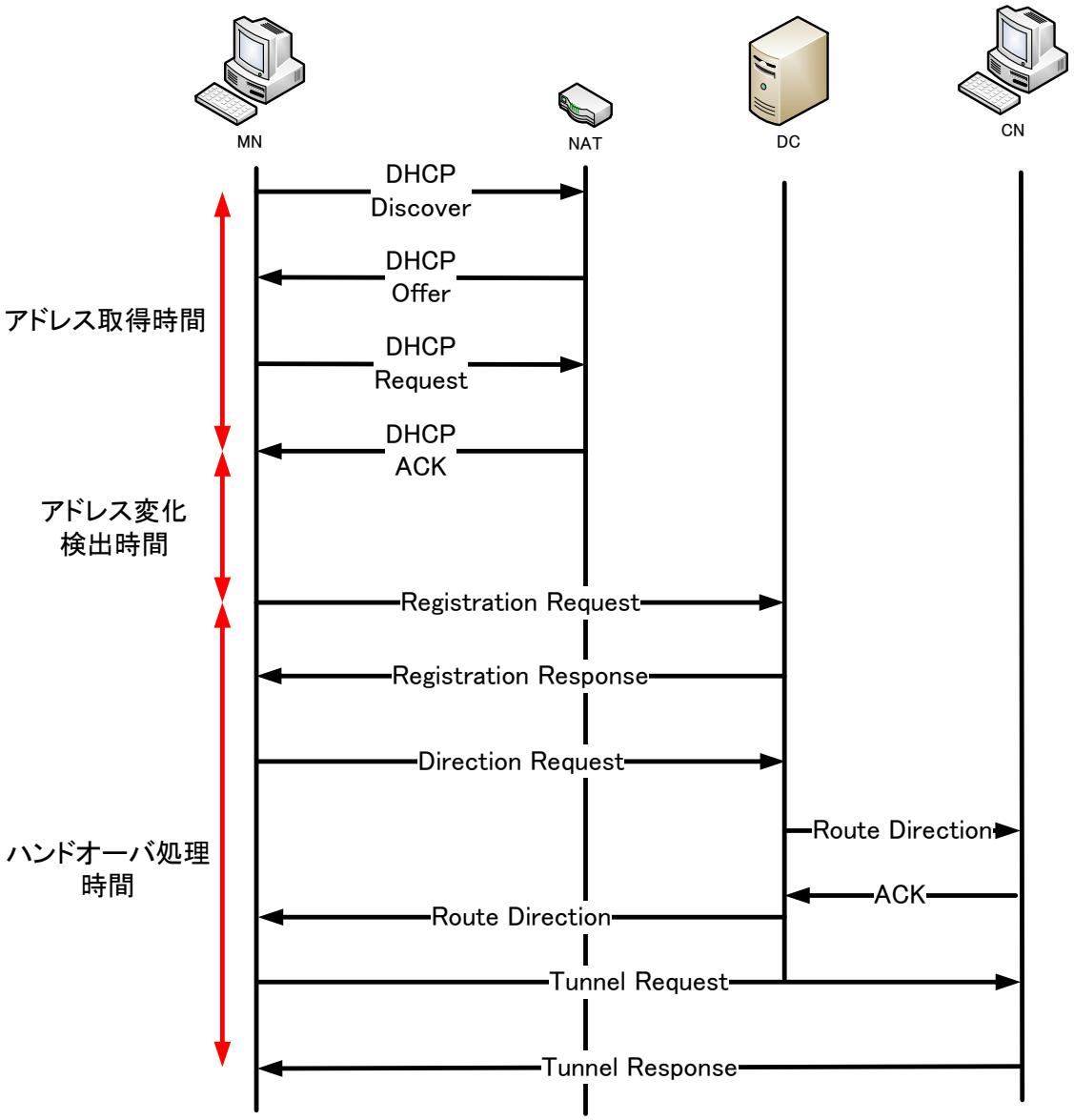

図 13　フレームワーク組込型 NTMobile の構成

表 3　移動処理に要した時間

	平均	最大	最小
全体処理時間 [sec]	7.002	9.274	1.510
アドレス取得時間 [sec]	0.007	0.008	0.006
アドレス変化検出時間 [sec]	6.911	9.201	1.477
ハンドオーバ処理時間 [sec]	0.084	0.130	0.026

第5章　まとめ

本論文ではフレームワーク組込型 NTMobile における移動部分の実装を行った. カーネルモジュール実装型 NTMobile では未実装であったため, 今回の実装により NTMobile における移動透過性が実現できた. フレームワーク組込型 NTMobile は iOS や Android といったカーネル空間がブラックボックスとなっている OS でも利用できる. NTMobile の API を各 OS に対応した言語でラッパー関数を作成することで C 言語以外でも利用可能になる. 今後はこれらの端末において実装および動作検証を行っていき, 実用化を目指す予定である.

謝辞

本研究を進めるに当たり, 終始丁寧かつ細かなご指導を承りました, 指導教官である名城大学理工学部情報工学科　渡邊晃教授に心から感謝いたします.

本研究を進めるに当たり, さまざまな助言を賜わりました, 名城大学理工学部情報工学科　鈴木秀和准教授に深く感謝いたします.

本研究を進めるに当たり, 有益なご意見を賜りました, 愛知工業大学情報科学部情報科学科　内藤克浩准教授に深謝いたします.

本研究を進めるに当たり, 常に迅速かつ適切なご意見並びに助言を賜りました, 納堂氏に心から感謝いたします.

最後に本研究を進めるに当たり, 日頃から多くの有益なご意見を賜りました, 渡邊研究室鈴木研究室及び NTMobile に携わるすべての皆様に感謝いたします.

参考文献

[1] 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 における通信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).

[2] 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. 367–379 (2013).

[3] 上醉尾一真，鈴木秀和，内藤克浩，　渡邊晃：IPv4/IPv6 混在環境で移動透過性を実現する NTMobile の実装と評価，情報処理学会論文誌，Vol. 54, No. 10, pp. 2288–2299 (2013).

[4] 土井敏樹，鈴木秀和，内藤克浩，　渡邊晃：NTMobile における RS の検討，*DICOMO202*, pp. 1162–1168 (2012).

[5] lwip: *lwIP - A Lightweight TCP/IP stack - Summary*. `http://savannah.nongnu.org/projects/lwip/` (2017/2/8 閲覧).