NTMobileにおけるアドレス変化検出方法の検討と実現

140441067　佐藤 洸輔
渡邊研究室

1. はじめに

スマートフォンやタブレット端末の普及により，手軽にインターネット接続が可能となった．モバイル端末においては，電波帯域の逼迫により，通信中でも携帯網のトラフィックを任意の Wi-Fi ネットワークにオフロードしたいという要求が出てきている．IP 通信では，一般にネットワークを切り替えると IP アドレスが変わるため，通信が継続できない．そこで，このような場合でも通信を継続できる移動透過性が求められている．NTMobile（Network Traversal with Mobility)[2] は移動透過性を実現できる有用な手段である．従来の NTMobile は，カーネル空間上で実装しているためモバイル端末では NTMobile を実装することができず，NTMobile の普及が見込まれない．そこで，モバイル端末で NTMobile の機能を実装するため，アプリケーション層で処理を行うフレームワーク版 NTMobile が必要となった．本稿ではフレームワーク版 NTMobile におけるアドレス変化検出部分の実装及び評価について報告する．

2. NTMobile

NTMobile は，移動を前提とした処理を行う NTMobile を搭載した端末 (NTM 端末) と，Direction Coordinator (DC) により構成される．DC は，NTM 端末の管理，仮想 IP アドレスの管理及び通信経路の管理などを行う．NTMobile では端末の移動にともなう実 IP アドレスの変化を隠蔽するために，アプリケーションはノード識別子である仮想 IP アドレスを用いて通信を行う．このとき，使用する仮想 IP アドレスは DC より配布される．NTM 端末は仮想 IP アドレスによる IP パケットを，位置識別子である実 IP アドレスを用いてカプセル化することによりトンネル通信を行う．そのため，移動して実 IP アドレスが変化してもアプリケーションは同一仮想 IP アドレスを継続して利用可能であり，移動透過性を実現することが可能である．

3. アドレス変化検出方法

これまでの NTMobile は Windows や Linux といったすべての OS のアドレス変化検出を NTMobile 内で処理しようとしていたが，Windows，Linux など OS ごとでネットワークインタフェース情報が違い，特に Android などのモバイル端末では，LTE や Wi-Fi といった様々なネットワークインタフェース情報を単一のアプリケーションで処理しようとすると複雑になり実装できないため，アドレス変化検出部分をそれぞれの OS ごとに別アプリケーションとして実装する方式とした．

各 OS ではアドレス変化検出アプリケーションがカーネル空間からユーザ空間へネットワークインタフェース情報を通知する仕組みを用いる．アドレス変化検出アプリケーションはアドレス変化の通知を受け，NTMobile フレームワークへアドレス変化検出を通知する．IP アドレスの変化検出の通知を受けた NTMobile フレームワークは再度トンネル構築を行うことで移動透過性を実現する．

4. 実装と評価

アドレス変化検出アプリケーションを Linux で実装を行った．アプリケーションのモジュール構成を図 1 に示す．

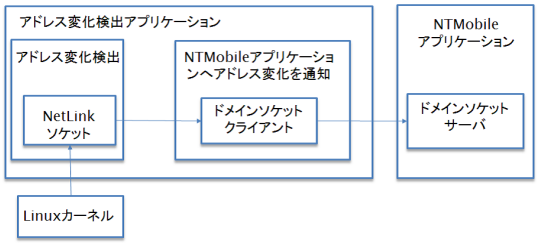


図 1: アドレス変化検出アプリのモジュール構成

Linux では，netlink と呼ばれるカーネル空間とユーザ空間の通信を行う仕組みが存在する．この netlink を用いることでカーネル空間でネットワークインターフェースを監視し，IP アドレスが変化した際にユーザ空間に存在するアプリケーションへ通知を行うことができる．この IP アドレスの変化検出を行うアプリケーションは，ドメインソケットと呼ばれるアプリケーション間のデータのやり取りを用いることで，NTMobile フレームワークへアドレス変化検出を通知する．

表 1: 処理時間

測定箇所	処理時間 $[ms]$
NAT に接続するまでの時間	3,416
NAT の認証にかかる時間	318
IP アドレス取得時間	18
IP アドレス変化検出時間	18
トンネル再構築時間	88

次に，アドレス変化アプリケーションを用いた NTMobile 通信の動作検証と処理時間の測定を行った．評価用ネットワーク構成は 2 台の NTM 端末と DC，NAT が同一ネットワーク上に存在する．通信開始後一方の端末を NAT 配下に移動する．その後 NTM 端末が NAT に接続されるまでの時間，NTM 端末と NAT が認証を行う時間，DHCP による NAT からのアドレス取得時間，アドレス変化検出を行い NTMobile へ通知されるまでの時間，トンネル再構築までの時間の測定を 10 回行った．これらを平均した結果を表 1 に示す．測定結果より，OS が移動前の接続先からネットワークを切断し，移動後のネットワークへ接続する時間が処理の大部分を占めていることが分かった．

5. まとめ

本稿では NTMobile におけるアドレス変化検出の方法を提案し，Linux においてアドレス変化検出アプリケーションを実装した．今後，Windows や Android といった別 OS におけるアドレス変化検出アプリケーションの実装を検討していく．

参考文献

[1] 上酔尾．　他：情報処理学会論文誌 Vol.54, NO.10, pp.2288-2299, 2013

NTMobileにおける
アドレス変化検出方法の検討と実現

140441067
渡邊研究室　佐藤 洸輔

Watanabe Lab.

研究背景

- スマートフォンやタブレット端末の普及
  - 携帯網が逼迫
    - Wi-Fiへのオフロード
- オフロードに係る課題
  - 接続先変化によるIPアドレスの変化
    - 通信が切断
- 通信制約の課題
  - NAT越え問題
  - IPv4 - IPv6が非互換

NTMobile（Network Traversal with Mobility）

NTMobileの概要

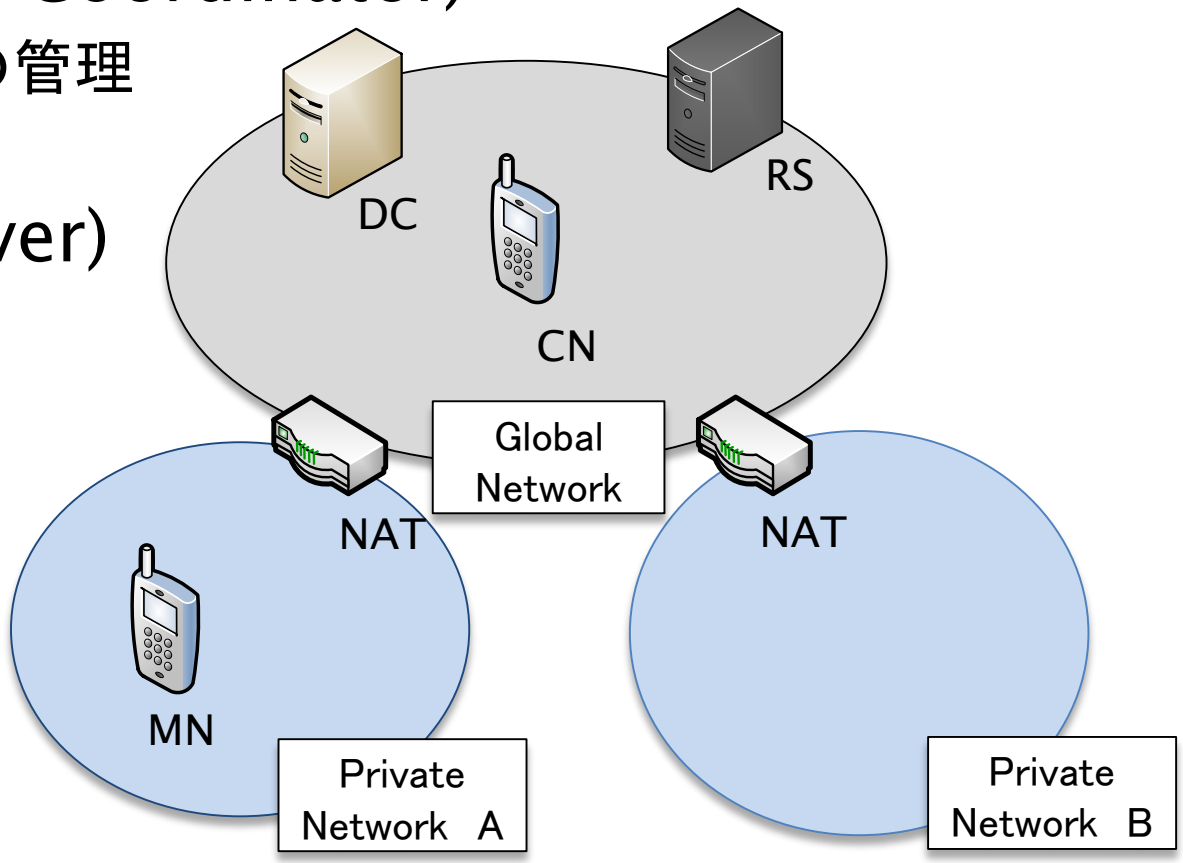
- *NTMobile（Network Traversal with Mobility）
  - 通信接続性と移動透過性を同時に実現する技術
- 通信接続性の実現
  - NAT越え問題の解決
  - IPv4 – IPv6間通信の実現
- 移動透過性の実現
  - ネットワークが切替わっても通信が継続

*鈴木 秀和.他 “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.1pp-367-379(2013).

NTMobileの構成

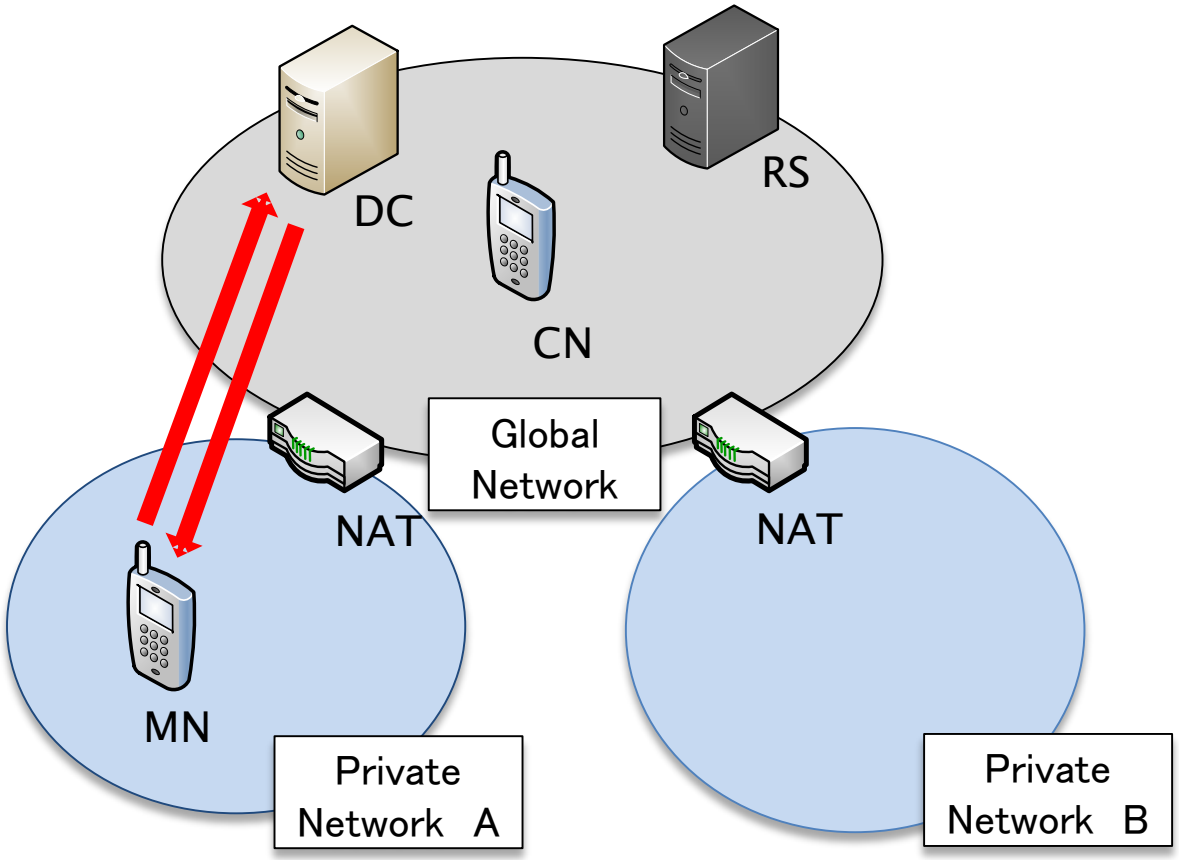
- NTM端末
  - NTMobileを実装した端末(MN CN)
- DC(Direction Coordinator)
  - 仮想IPアドレスの管理
  - 通信経路指示
- RS(Relay Server)
  - 通信の中継

MN(Mobile Node)
CN(Correspondent Node)



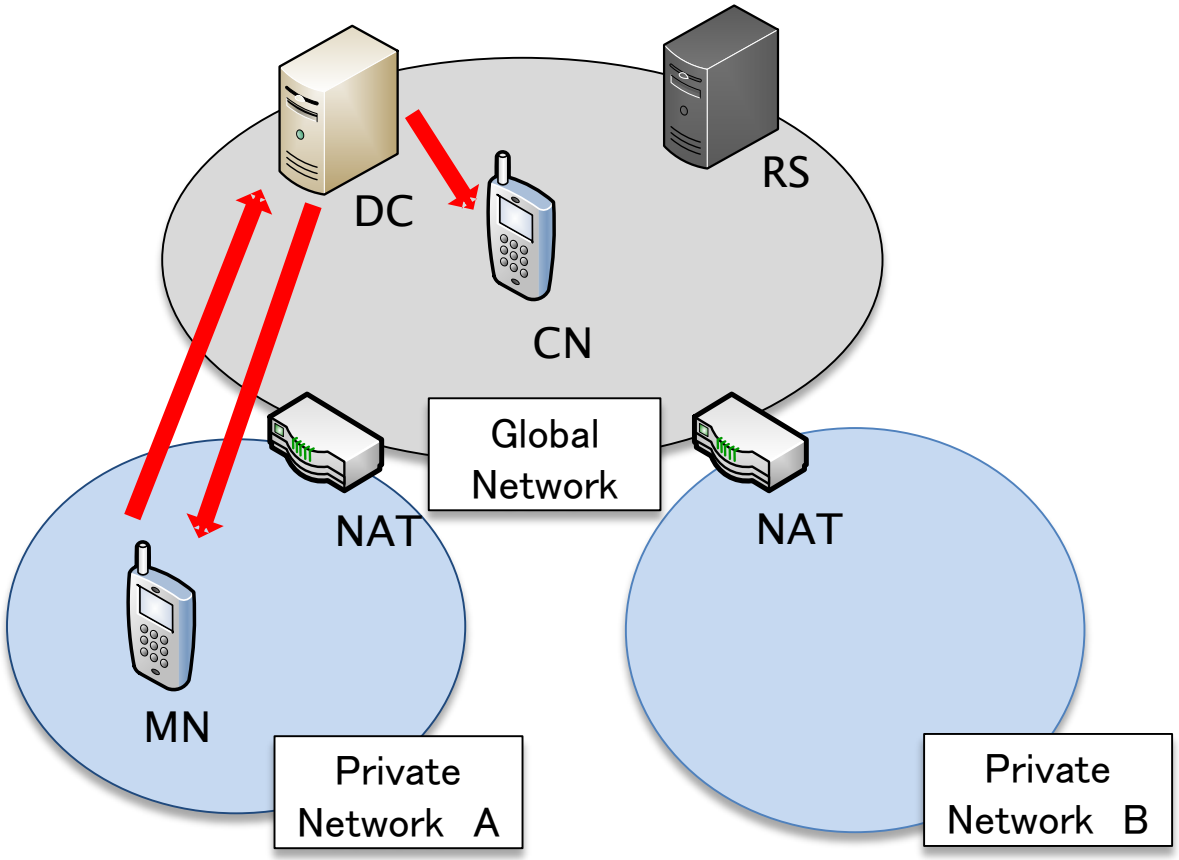
NTMobileの動作

- 端末起動時
  - DCに実IPアドレスを登録
  - DCから仮想IPアドレスを配布



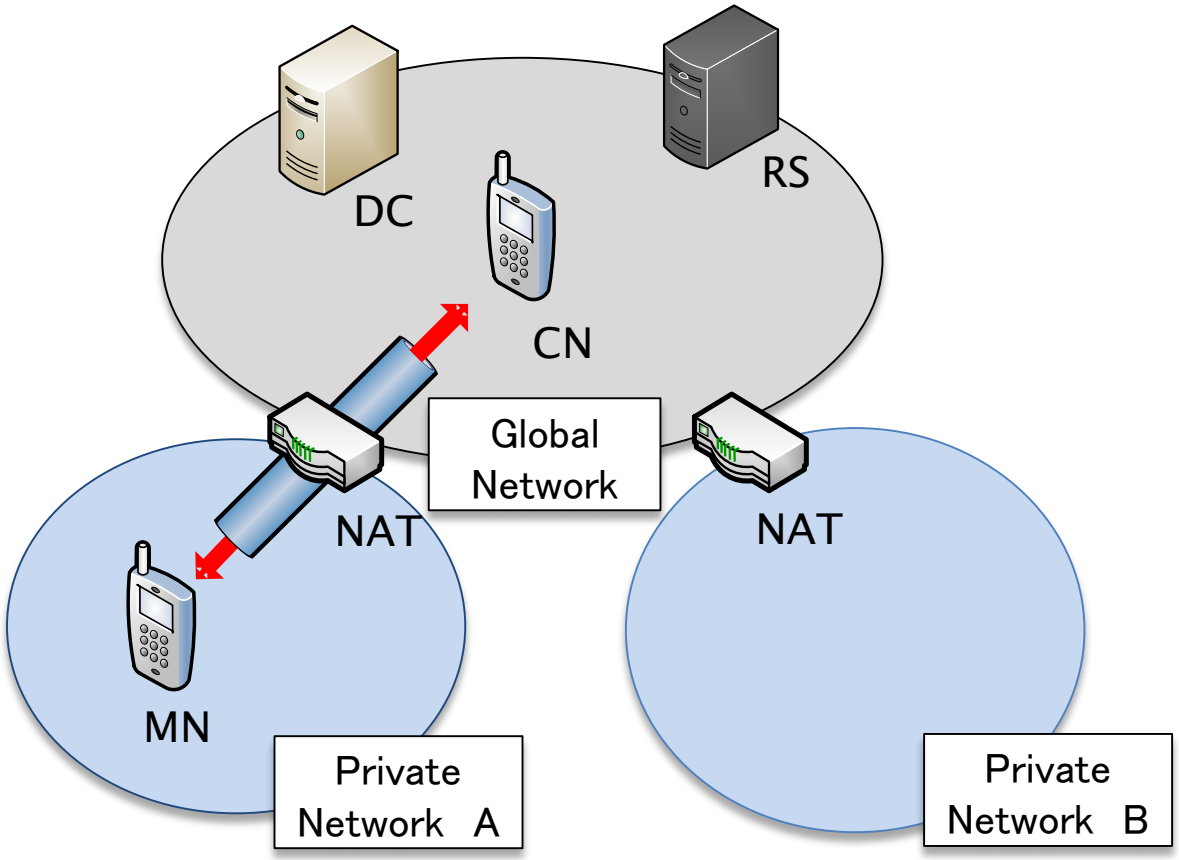
NTMobileの動作

- 通信開始時
  - MNがDCに対して経路指示要求を出す
  - DCがMN・CNに対して経路指示を行う



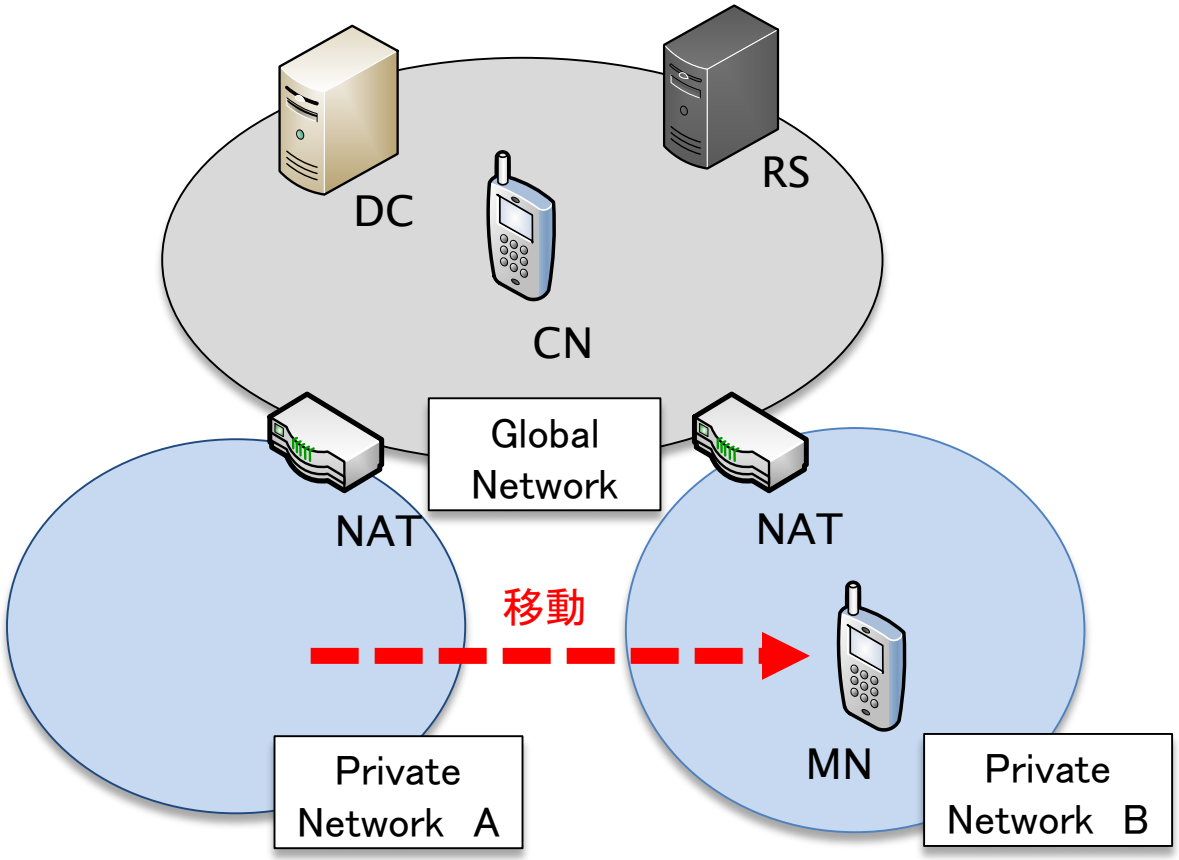
NTMobileの動作

- 通信開始時
  - 指示に従いMN・CN間でトンネル経路を構築する



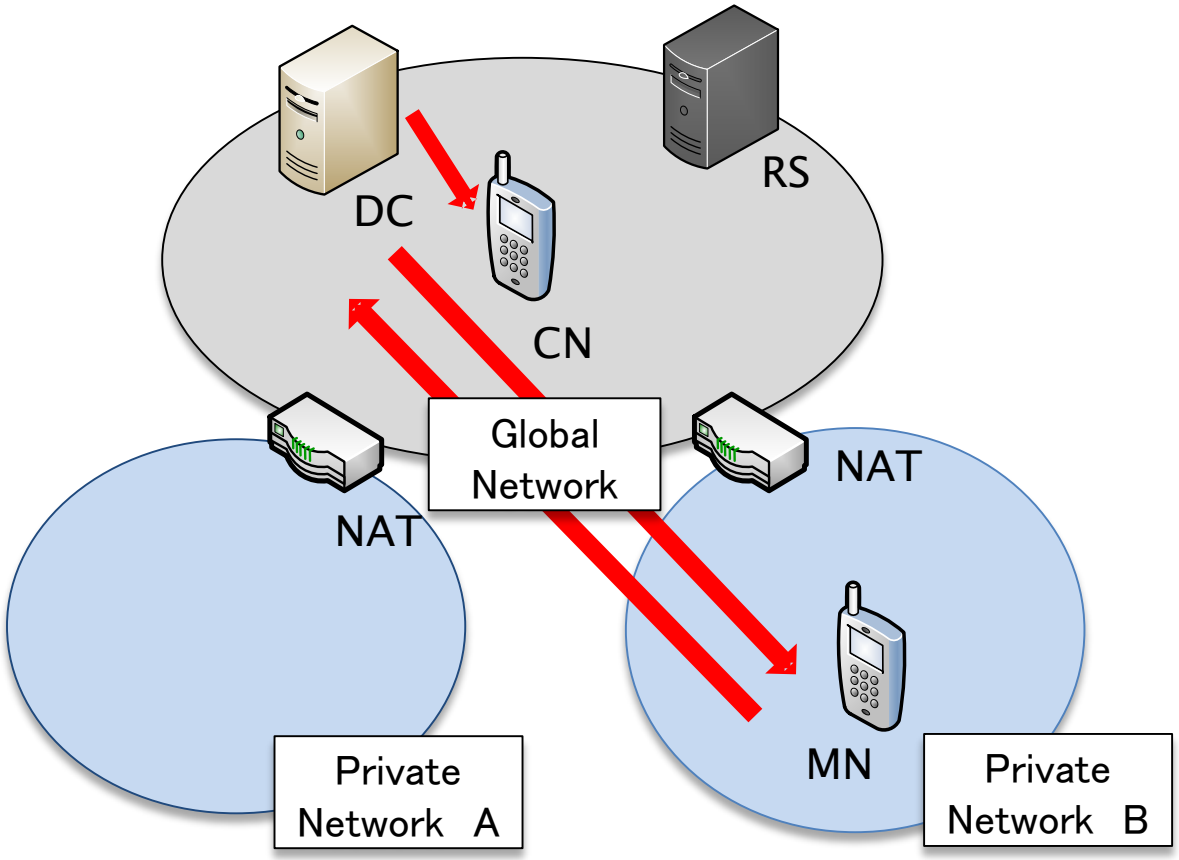
NTMobileの動作

- 端末移動時
  - MNが通信中に移動



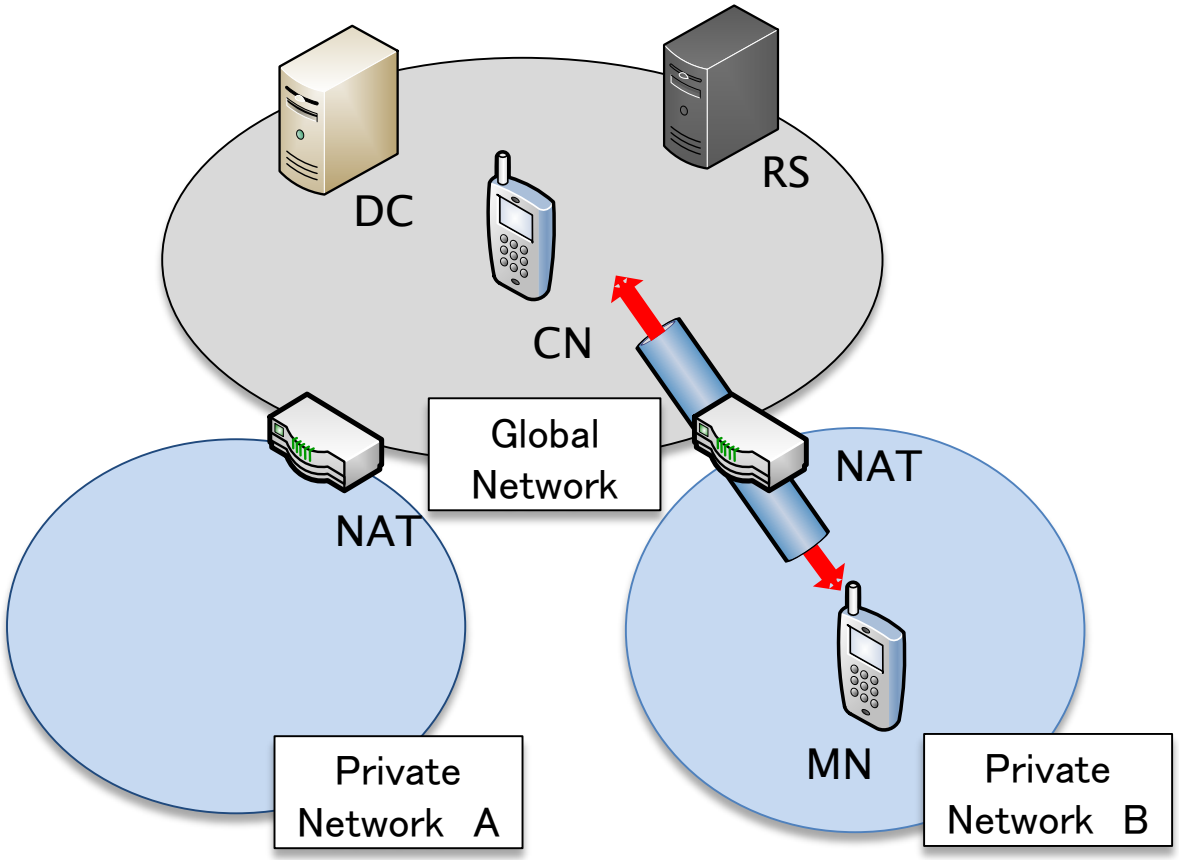
NTMobileの動作

- 端末移動時
  - MNがDCに移動を通知
  - DCがMN・CNに対して経路指示を行う



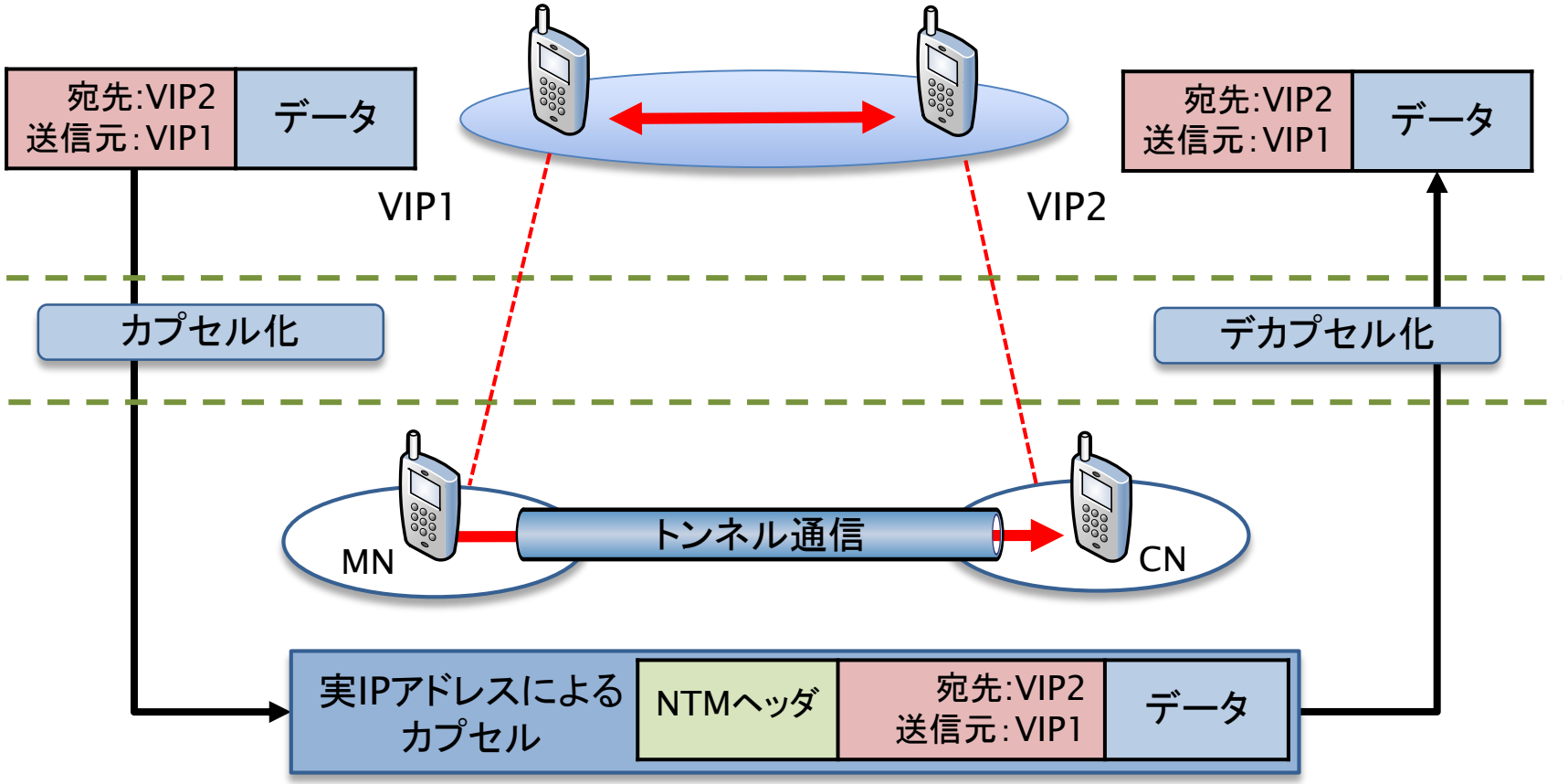
NTMobileの動作

- 端末移動時
  - 新しいトンネル経路で通信継続

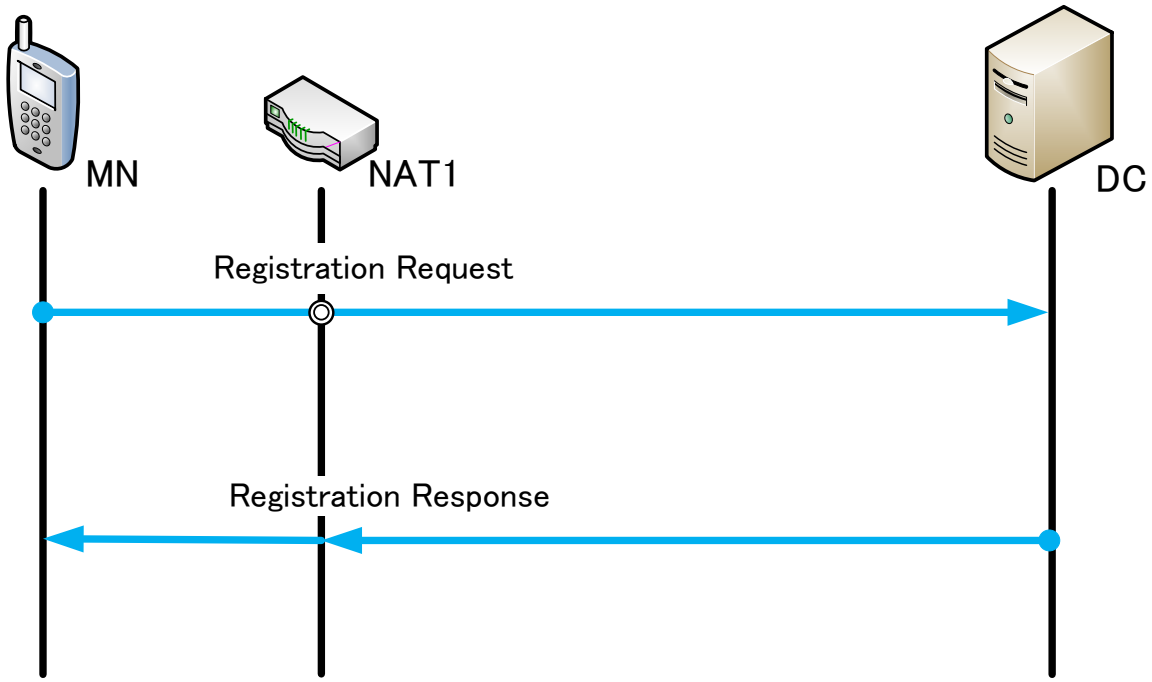


NTMobileの通信

- アプリケーションは仮想IPアドレスを認識
  - 実IPアドレスの変化をアプリケーションから隠蔽



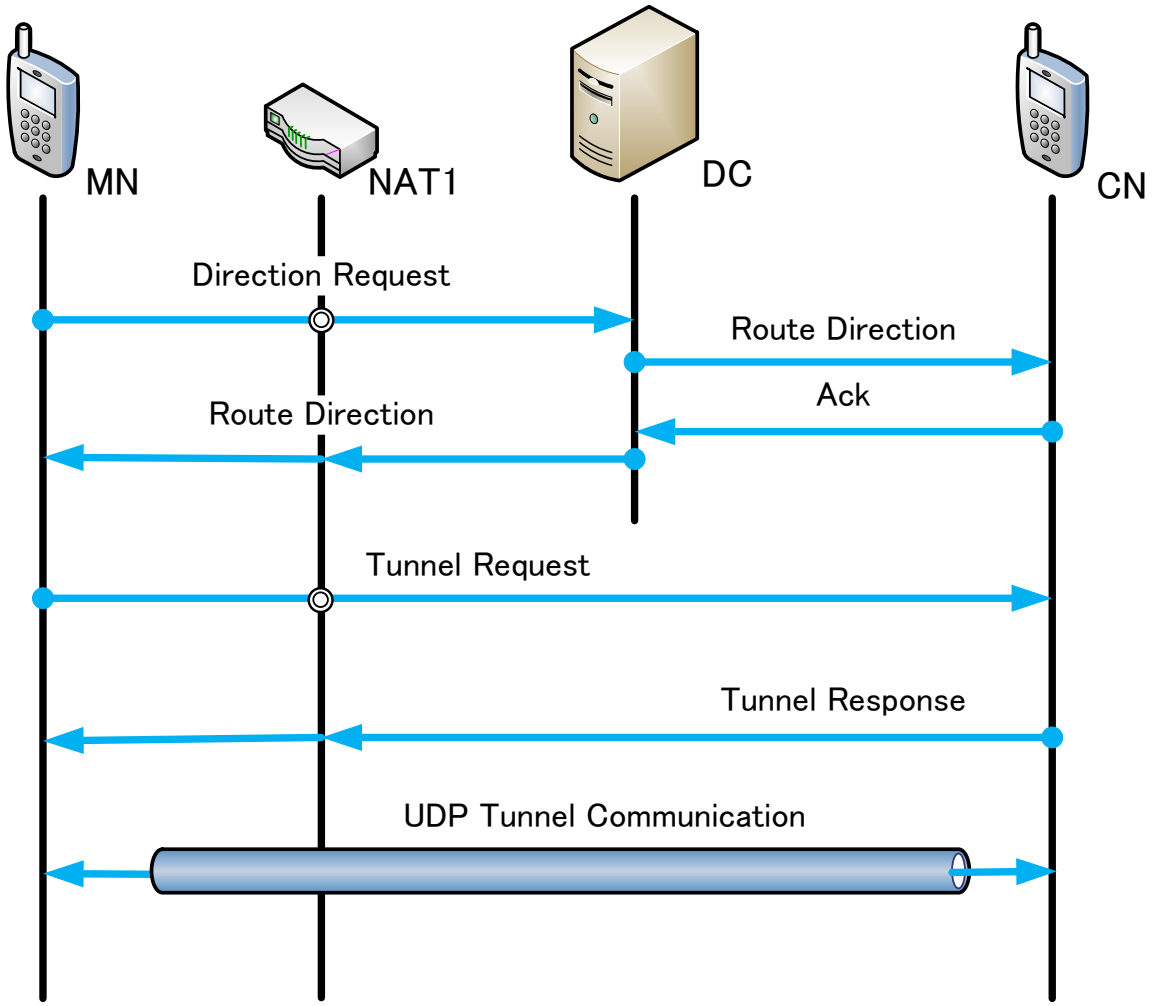
シーケンス（起動時）



登録処理

端末情報を登録

シーケンス（通信開始時）



DCへ経路指示要求

CNへ経路指示

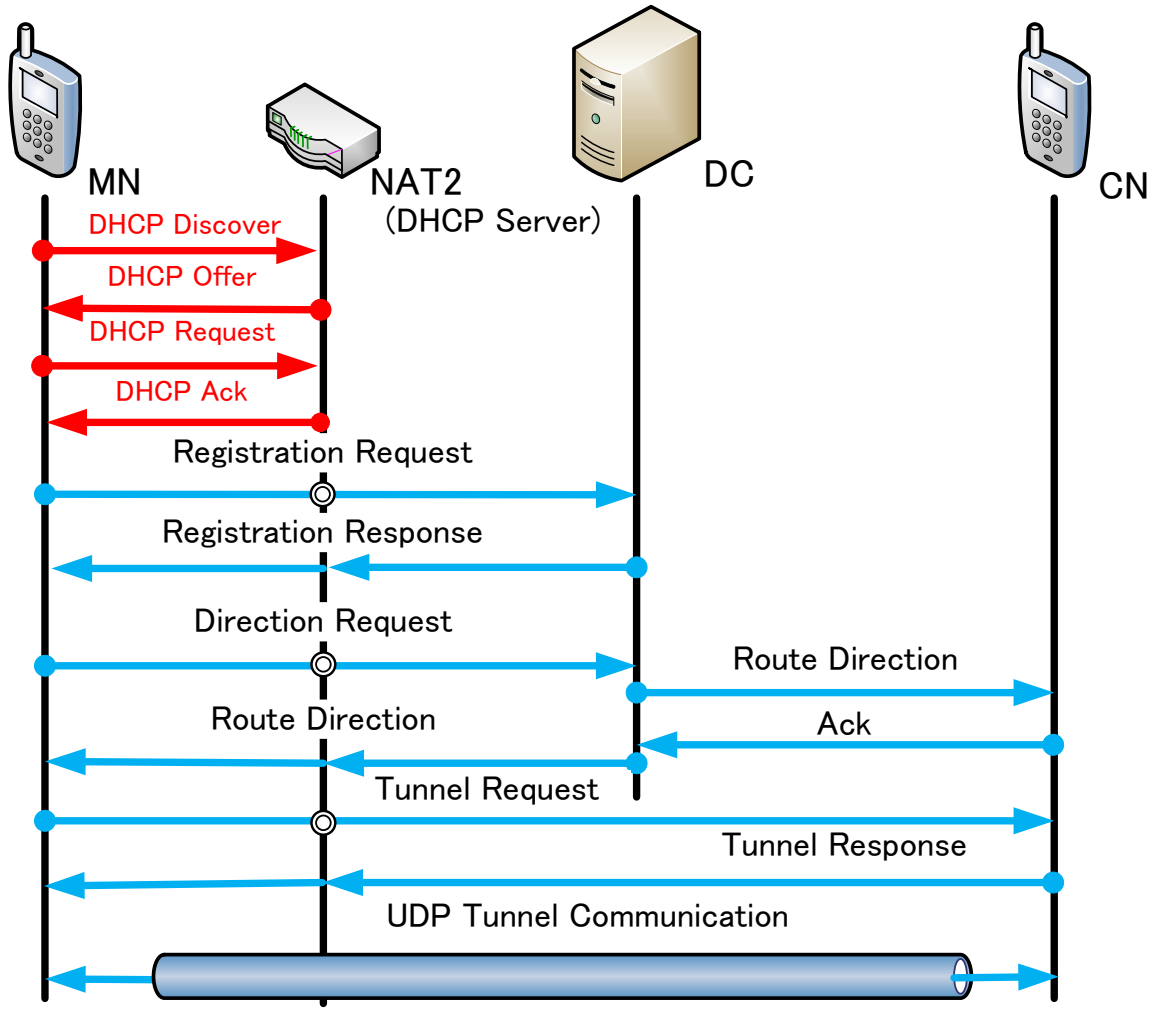
MNへ経路指示

トンネル要求

トンネル応答

トンネル通信

シーケンス（移動時）



アドレス取得

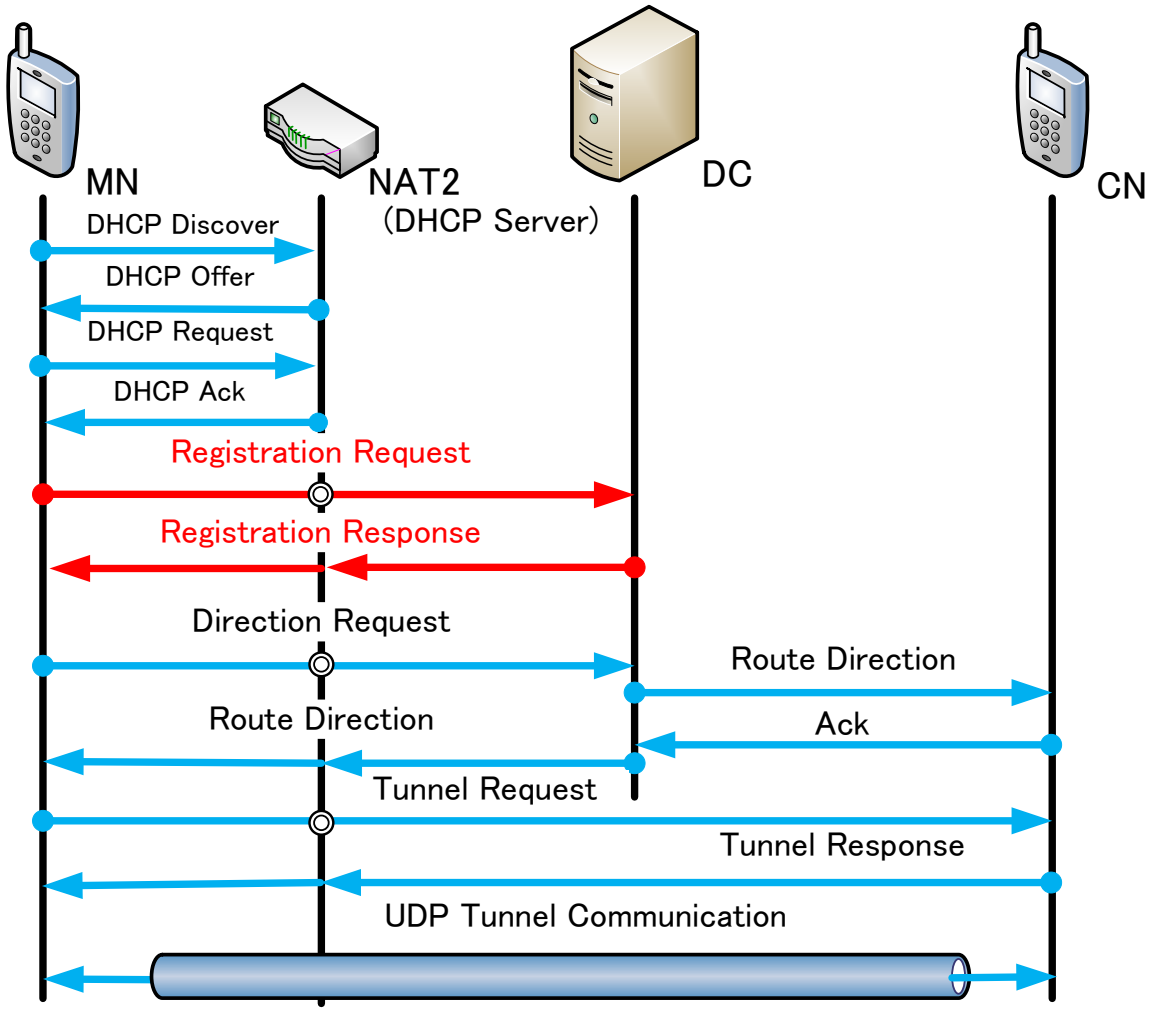
登録処理

DCへ変化後
IPアドレスを登録
起動時と同じ処理

トンネル再構築

通信開始時と同じ処理

シーケンス（移動時）



アドレス取得

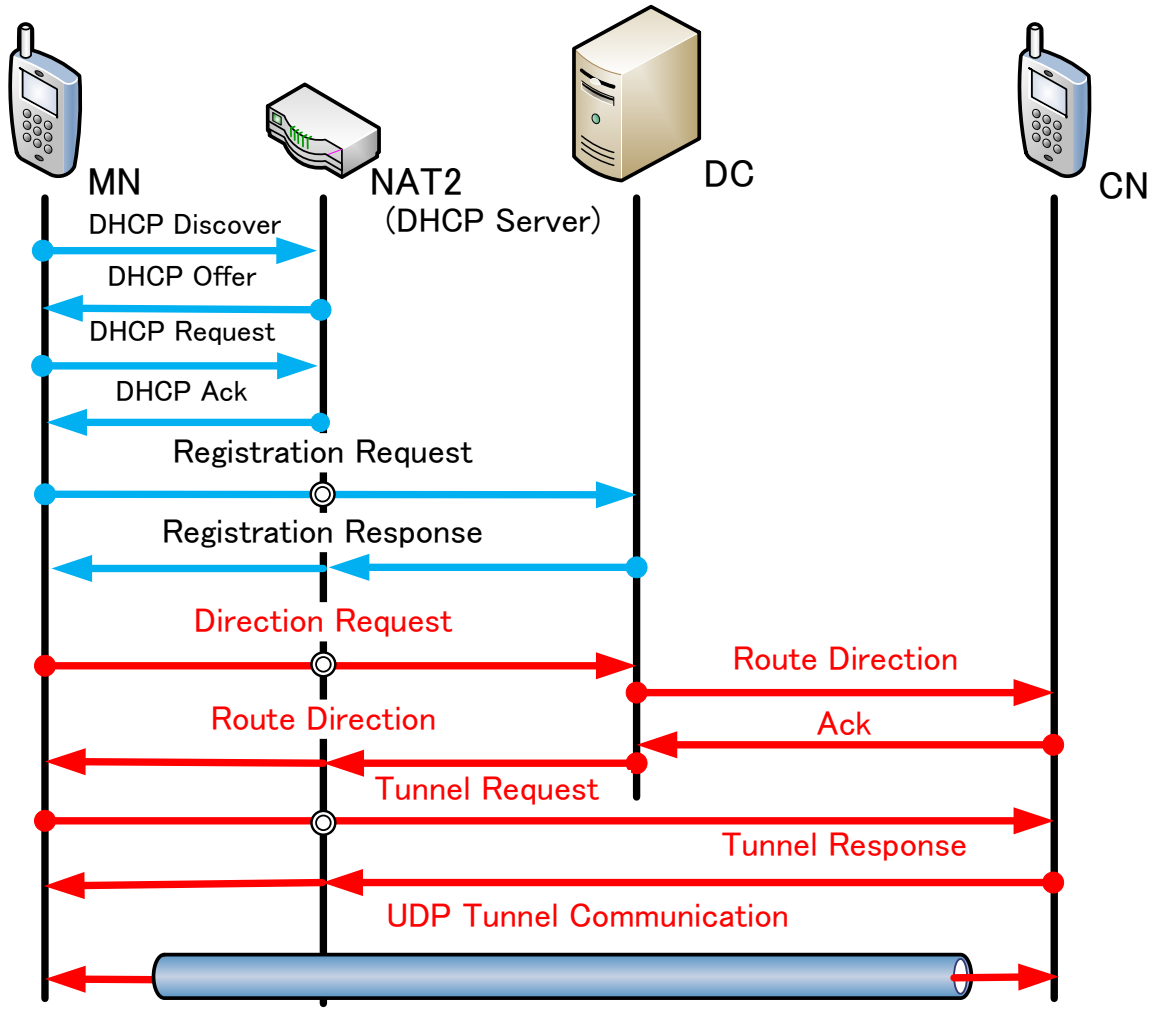
登録処理

DCへ変化後
IPアドレスを登録
起動時と同じ処理

トンネル再構築

通信開始時と同じ処理

シーケンス（移動時）



アドレス取得

RIP3取得

登録処理

DCへ変化後
IPアドレスを登録
起動時と同じ処理

トンネル再構築

通信開始時と同じ処理

NTMobile

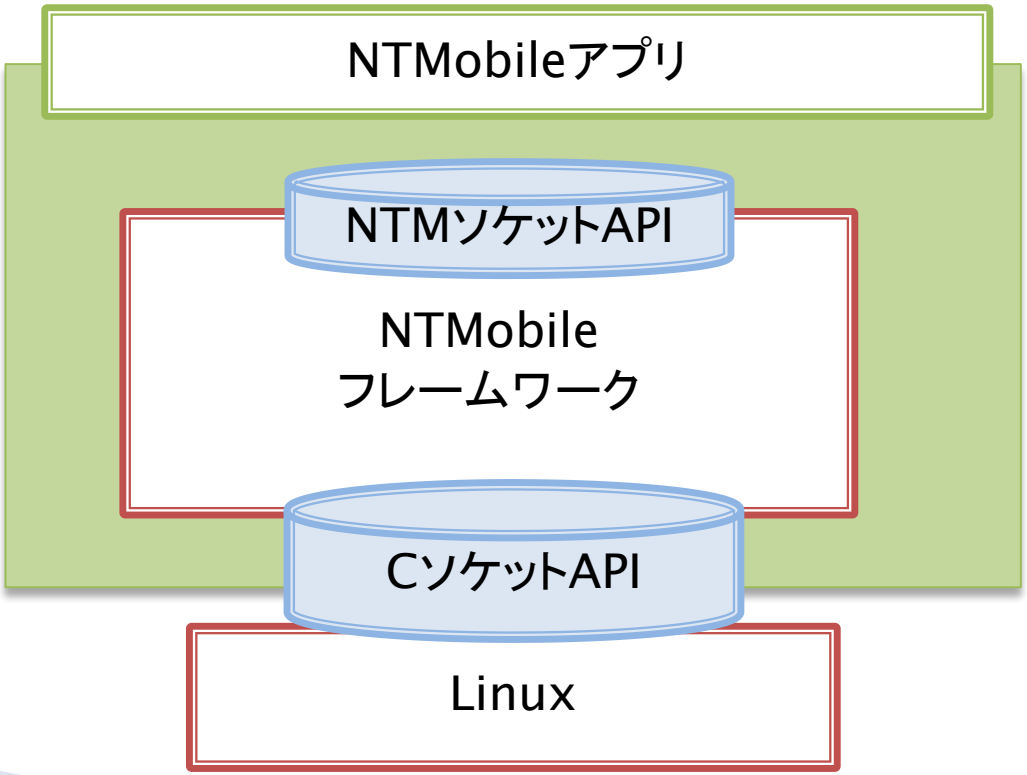
カーネル実装にはルート権限が必要

スマートフォンではルート権限を取得できない

NTMobile機能をユーザ空間で実装で実装する必要がある

NTMobileフレームワーク(NTMfw)

- NTMobile機能をユーザ空間で実現する実装方式
- アプリケーションはC言語の標準ソケットAPIに代わり，NTMソケットAPIを使用する



C標準ソケットAPI

- bind
- sendto
- recvfrom

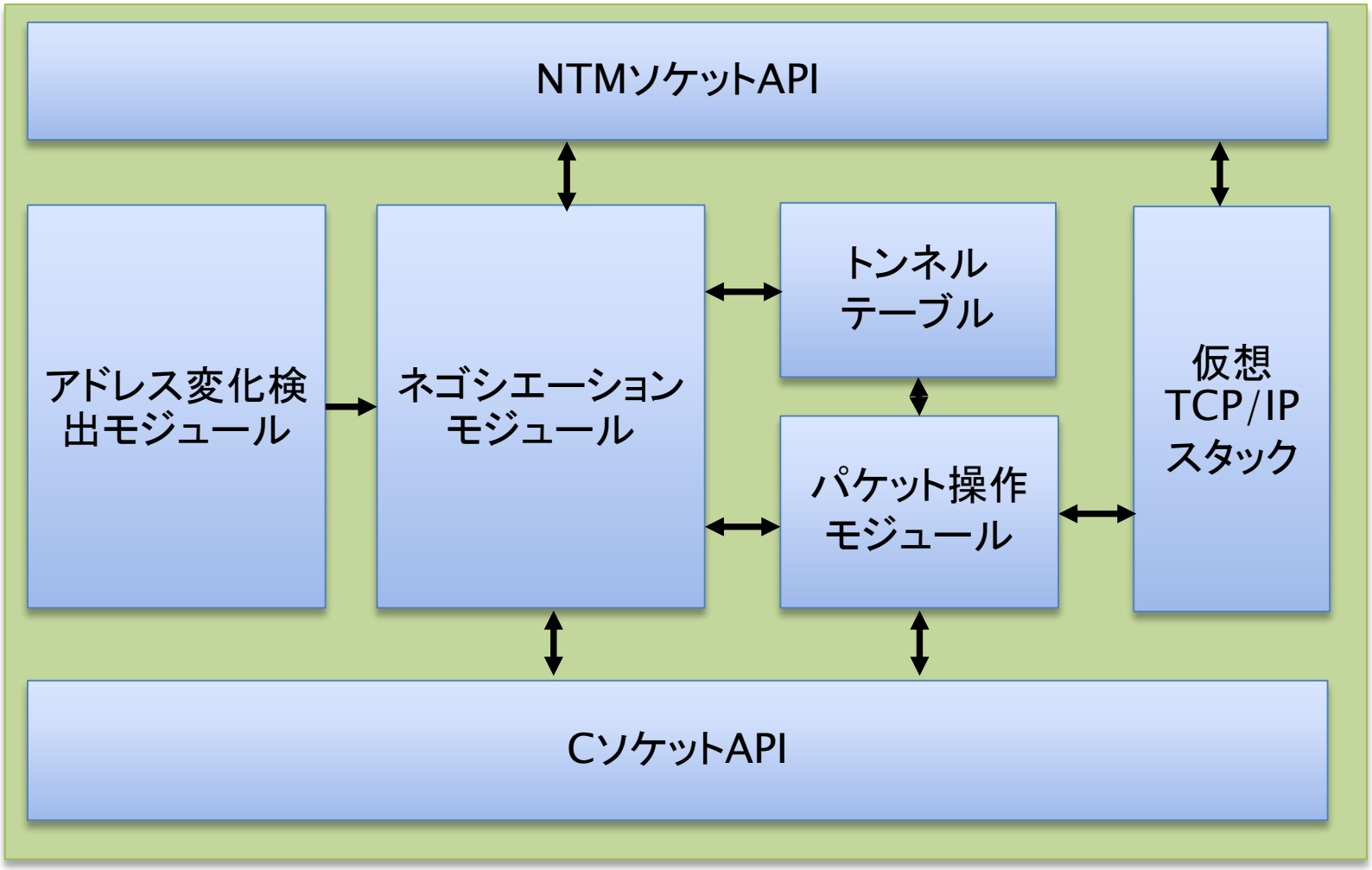
⋮

NTMソケットAPI

- Ntmfw_bind
- Ntmfw_sendto
- Ntmfw_recvfro m

⋮

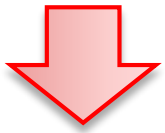
NTMobileフレームワークの構成



課題と目的

- OSごとにネットワークインタフェース情報が違うため処理が複雑になる

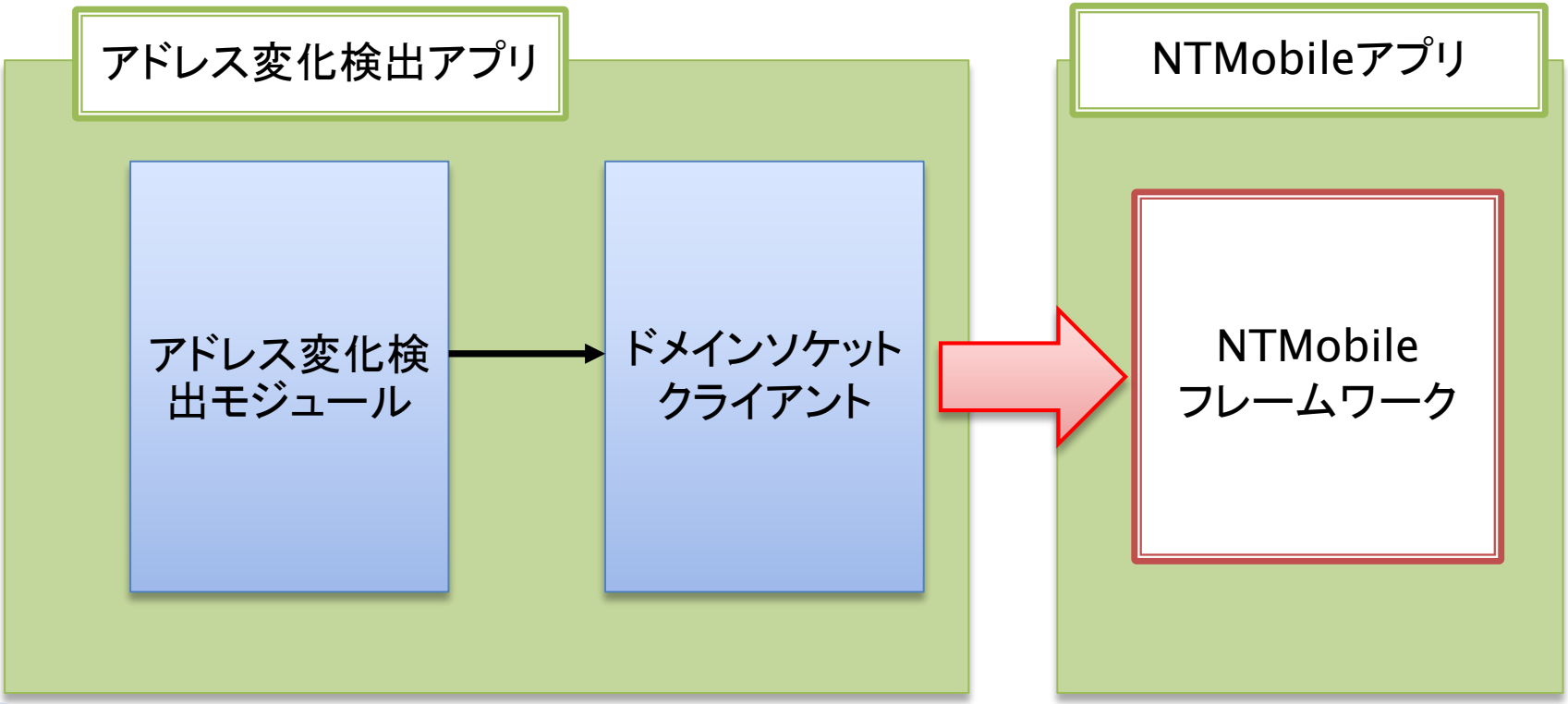
アドレス変化検出部分を
別アプリケーションとして切り離す



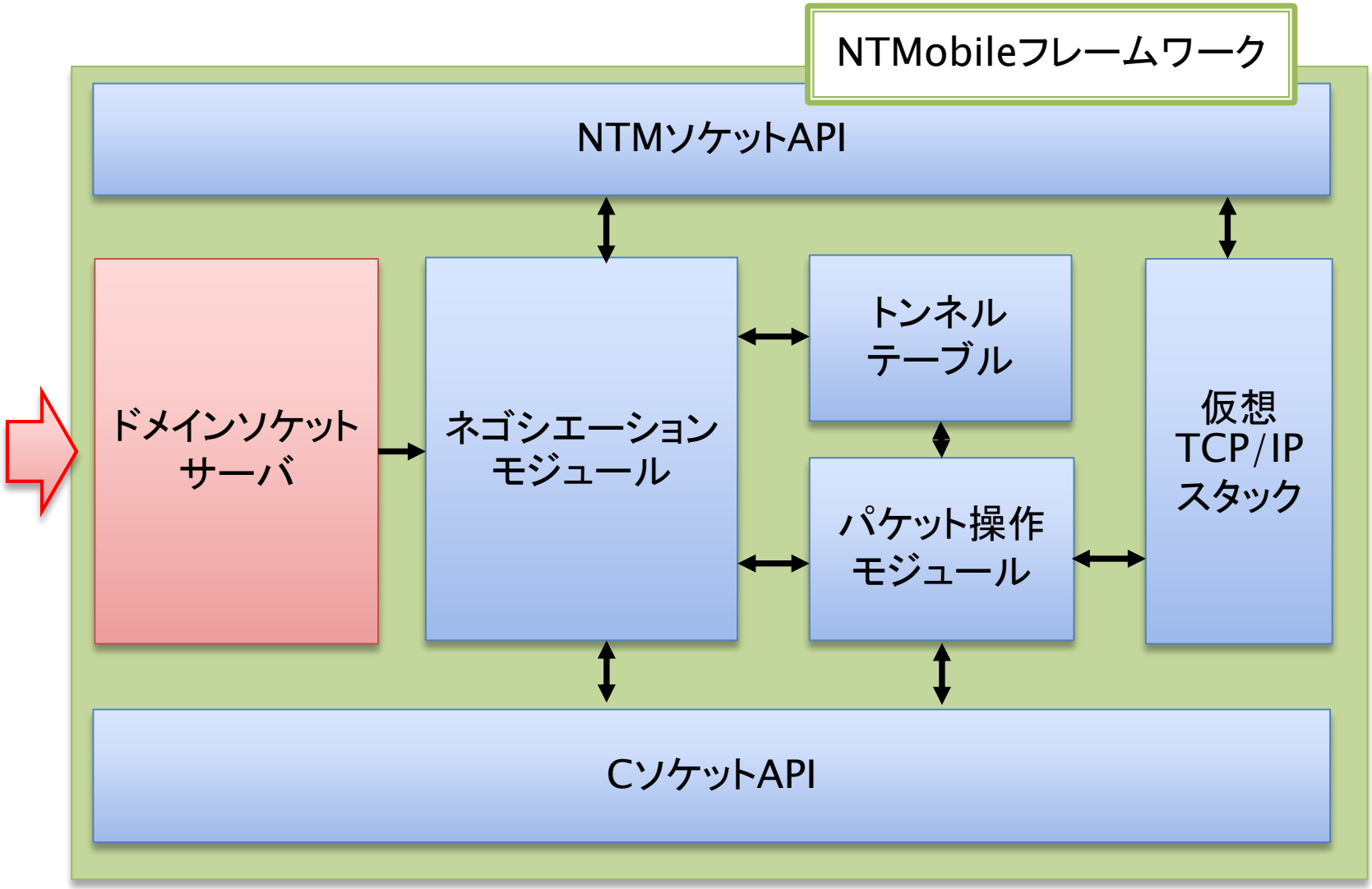
NTMobileフレームワークを全OSで共通化する

提案方式

- アドレス変化検出アプリでIPアドレスの変化を検出
- ドメインソケット経由でNTMobileフレームワークへアドレス変化を通知する

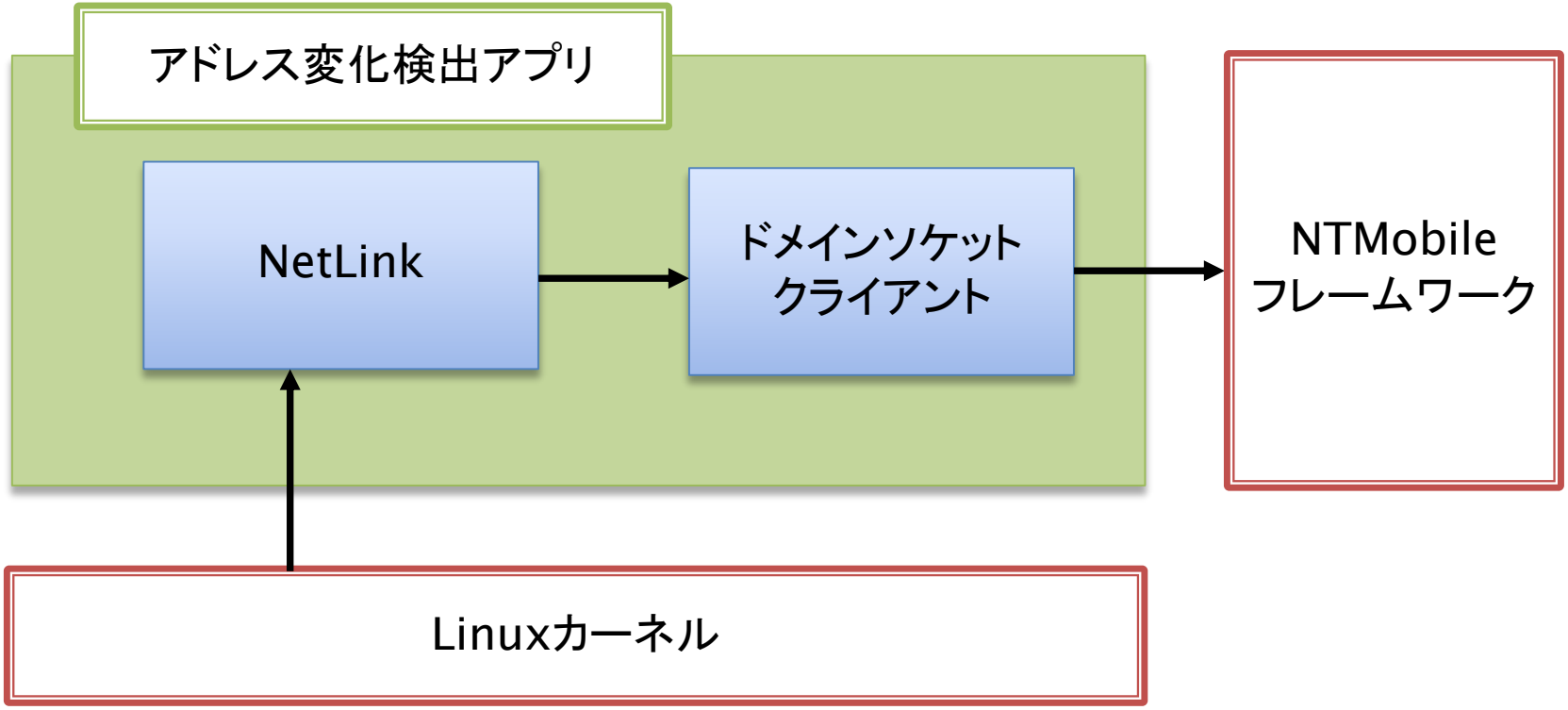


シグナリングの起動


NTMobileフレームワーク
NTMソケットAPI
ドメインソケット
サーバ
ネゴシエーション
モジュール
トンネル
テーブル
パケット操作
モジュール
仮想
TCP/IP
スタック
CソケットAPI

Linux上での実装

- アドレス変化検出にNetLinkを用いる



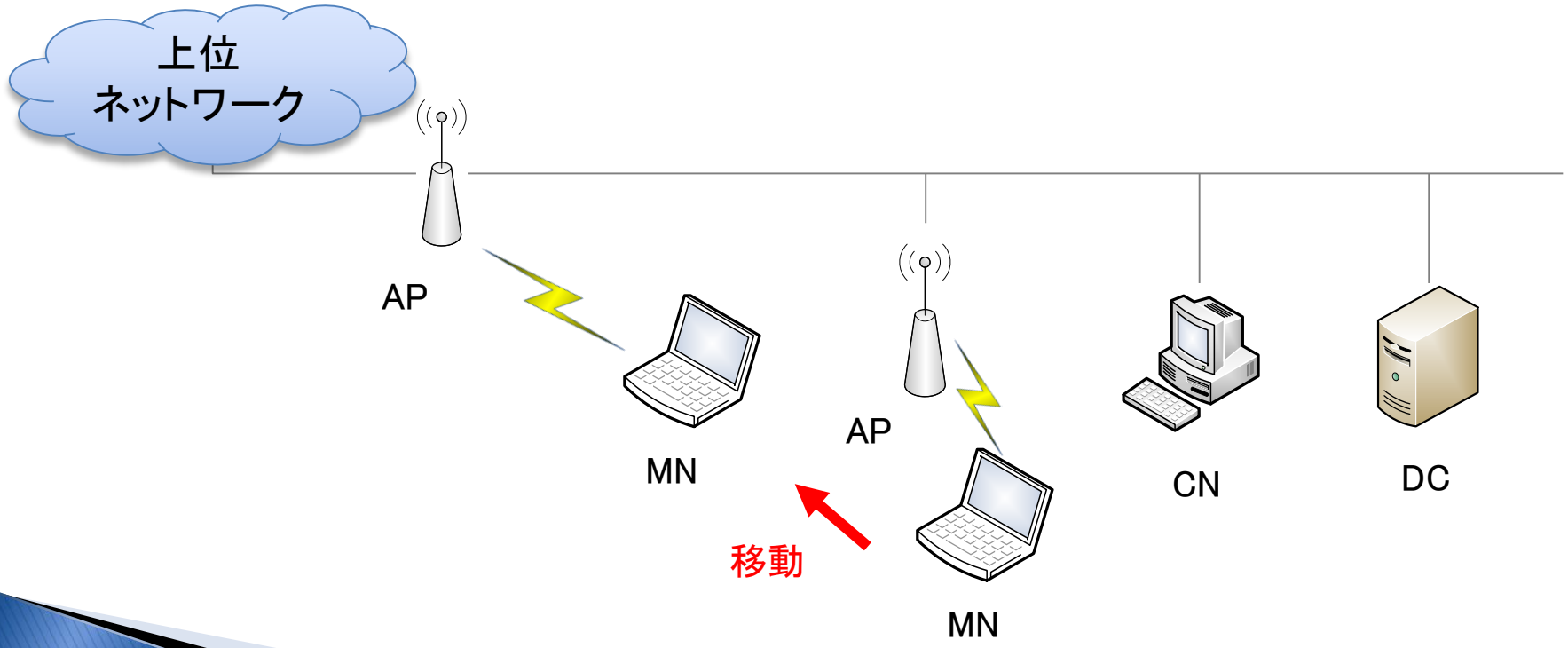
動作検証

- 装置の仕様

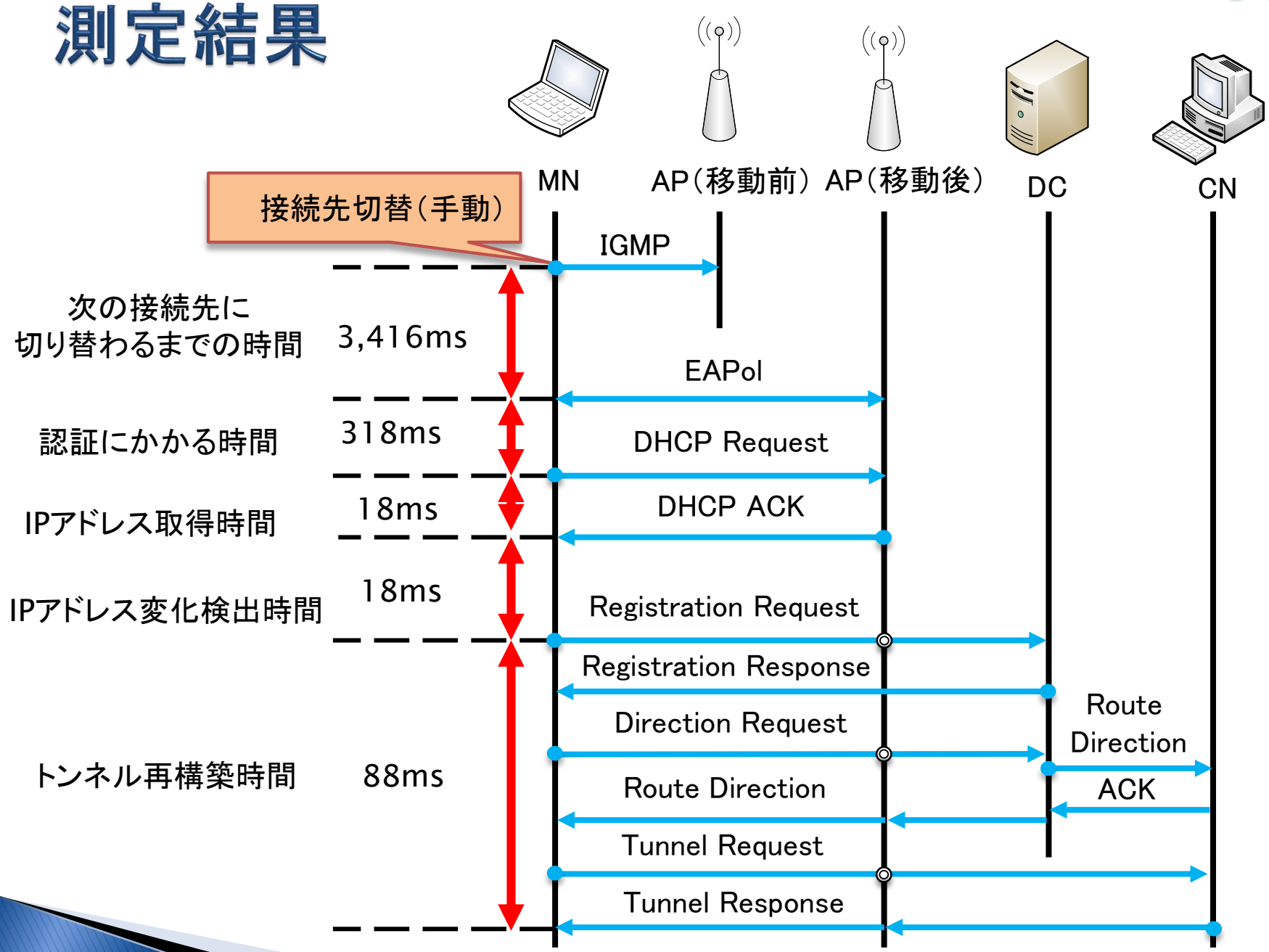
Device		OS	CPU	Memory
MN		Ubuntu 14.04	Intel Core i5-3320M 2.60GHz	4GB
CN		Ubuntu14.04	Intel Core i7-2600 3.40GHz	8GB
DC	ホストマシン	Windows 10 64bit	Intel Core i7-870 2.93GHz	20GB
	仮想マシン	Ubuntu14.04	1Core	1GB

測定結果

- APの切断から新たなAPに接続しトンネル再構築するまでの時間を計測
- 全体の処理時間を5つのフェーズに分割



測定結果


MN
AP（移動前）
AP（移動後）
DC
CN
接続先切替（手動）
IGMP
次の接続先に
切り替わるまでの時間
3,416ms
EAPol
認証にかかる時間
318ms
DHCP Request
IPアドレス取得時間
18ms
DHCP ACK
IPアドレス変化検出時間
18ms
Registration Request
Registration Response
Route
Direction
Direction Request
トンネル再構築時間
88ms
Route Direction
ACK
Tunnel Request
Tunnel Response

まとめ

- アドレス変化検出方法
  - アドレス変化検出部分をNTMfwから分離
- 実装と評価
  - Linux上で実装
    - 安定して動作することを確認
- 今後の予定
  - Windows・Androidへの移植