NTMobile フレームワークの Java ラッパーの提案と実装

清水 一輝 †*　八里 栄輔 ‡　納堂 博史 †　鈴木 秀和 †　内藤 克浩 §　渡邊 晃 †
† 名城大学　‡ バレイキャンパスジャパン　§ 愛知工業大学

1　はじめに

モバイルネットワークの普及に伴って，ネットワーク環境に関わらず通信を開始することができる通信接続性とネットワークが切り替わった際に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）は両者を同時に実現する次世代の技術である [1].

NTMobile は Linux カーネルに実装を行っていたが，スマートフォン等ではルート権限が必要となるため利用できなかった．そこで NTMobile をアプリケーションに移植し通信ライブラリを提供する方法として，NTMobile フレームワークを検討している [2]．フレームワークは C 言語によって記述されているため，Java で利用するには変換が必要である．そこで本研究では，NTMobile フレームワーク用の Java ラッパーを実現する方式について検討した．Java アプリケーションでは NTMobile をほとんど意識することなく，Java 標準 API を使用すると自動的に NTMobile フレームワークを呼び出すことができる．

2　NTMobile

2.1　概要

NTMobile の基本構成は，NTMobile 機能を有する NTM 端末と NTM 端末の端末情報管理やトンネルの経路指示を行う DC（Direction Coordinator）によって構成される．

NTM 端末は，起動時に自身の実 IP アドレスを DC に対して登録し，DC から仮想 IP アドレスを取得する．アプリケーションは仮想 IP アドレスを利用して通信を確立する．仮想 IP アドレスにより生成されたパケットは全て実 IP アドレスによってカプセル化する．この方法により NTM 端末は IPv4 グローバル/IPv4 プライベート/IPv6 アドレスの違いを意識することなく相互に通信ができ，通信中にネットワークを切り替えることが可能である．

Proposal and Implementation of Java Wrapper for NTMobile Framework
†Kazuki Shimizu, ‡Eisuke Yasato, †Hiroshi Nodo, †Hidekazu Suzuki, §Katsuhiro Naito, †Akira Watanabe
†Meijo University
‡Valley Campus Japan
§Aichi Institute of Technology

2.2　フレームワークの処理

図 1 に NTMobile フレームワークで行う処理を示す．通信開始側の端末を MN（Mobile Node），通信相手側の端末を CN（Correspondent Node）と呼ぶ．フレームワークの処理は大きく以下の 3 つに分けられる．

(i) 登録処理
　MN 及び CN 起動時に自身の実 IP アドレスを DC に登録し，DC から仮想 IP アドレスの取得する．これにより，NTMobile 通信のための準備が整う．上位アプリケーションに対して `ntmfw_ntm_init` 関数を提供する．

(ii) トンネル生成
　MN は通信開始時に CN の FQDN を指定して DC に経路指示を依頼する．MN と CN は DC の指示に従ってトンネル経路を構築する．上位アプリケーションに対して `ntmfw_getaddrinfo` 関数を提供する．

(iii) トンネル通信
　パケット送受信時に，フレームワークを呼び出し，NTMobile によるトンネル通信を行う．上位アプリケーションに対して名前が異なる C ソケットと同じ関数を提供する．

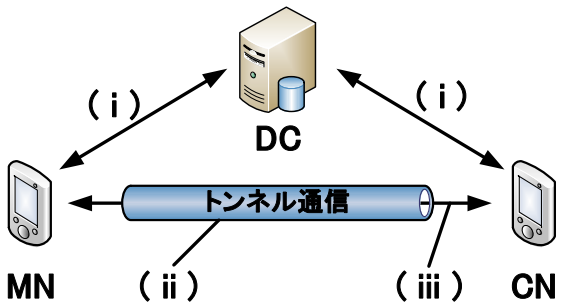


図 1: NTMobile フレームワークで行う処理

3　Java ラッパーの構成

図 2 に Java ラッパーを用いた NTM 端末のモジュール構成を示す．Java アプリケーションにて NTMobile 通信を行うには，C 言語で記述された NTMobile フレー

ムワークの関数を呼び出す必要がある．C 言語と Java では引数の型の違いがあるため，NTMobile フレームワークを呼び出す前に Java ラッパーを介して型の変換処理を行う．NTMobile の通信処理は NTMobile フレームワークが行うので，Java では関与する必要はない．

Java アプリケーションでは，Java 標準 API を使用して自動的に NTMobile 通信を実現できるように，Java 標準の通信クラスのメソッドを NTMobile 通信用のメソッドでオーバーライドする．これにより Java の通信に関するメソッドを使用した際に Java ラッパーによって NTMobile フレームワークの関数を自動的に呼び出すようになる．

図 2 の `init` と `getByName` の 2 つの処理は，NTMobile フレームワーク特有の処理を行うため，Java 標準 API をオーバーライドすることができない．そこでこの 2 つのメソッドは新たにクラスを作成し，そこに定義する．`init` は NTMobile を起動するために必要な処理であるため，アプリケーション側で使用する際に引数を予め知っておく必要がある．`getByName` は `ntmfw_getaddrinfo` を呼び出すが，Java 標準 API でも提供するメソッドであるため同じ引数で使用できるようにした．その他の Java ソケットは，型の変換後にそのまま NTMfw C ソケットを呼び出す．

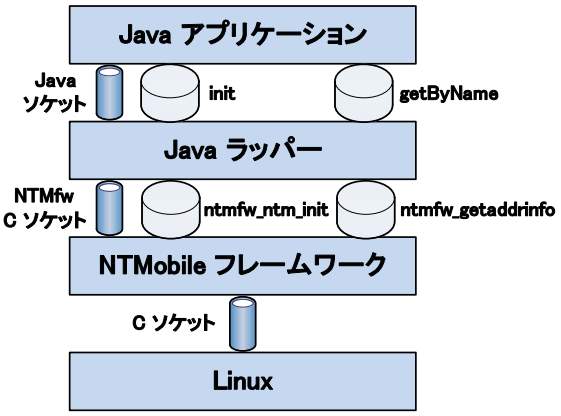


図 2: Java ラッパーを用いた NTM 端末のモジュール構成

4 評価

Java ラッパーを実装して NTMobile 通信を行い，動作検証と処理時間の測定を行った．

1 台のホストマシン上に DC，MN，CN の仮想マシンを構築し，DC，MN，CN を同一 IPv4 プライベートネットワークに接続し，動作検証を行った．処理時間の測定には，Java の System クラスにおける `nanoTime` メソッドを使用した．MN と CN 間で，Java ラッパーを用いて NTMobile で UDP でのトンネル通信を行った際に要する時間の測定を 7 回行った．Java では動的にクラスをロードするため，初回起動時等では立ち上がりが遅い．そこで 7 回測定した結果の最初の 2 回を除いた 5 回を平均した結果を表 1 に示す．表 1 の Linux とは，NTMobile を利用せず Java 標準 API にて用意されている UDP 通信用の DatagramSocket クラスの `send`/`receive` メソッドを使用し，送信/受信するのに要した時間である．この測定結果は図 2 の Java ラッパーと NTMobile フレームワークを除いた処理と同等であるため，表 1 の Linux の処理時間とみなすことができる．

表 1: 送信/受信の処理時間の平均

測定箇所	送信時 [ms]	受信時 [ms]
Java ラッパー	0.13	0.16
NTMobile	2.91	2.37
Linux	0.07	0.01
合計	3.11	2.54

表 1 より，NTMobile 通信を行うと送信時では，約 3.1 ミリ秒，受信時では約 2.5 ミリ秒の時間を要した．NTMobile 通信時にかかる時間の多くが NTMobile フレームワークであることが分かった．NTMobile フレームワークにて多くの時間を要するのは，メッセージ全体の暗号化/復号処理等が含まれるためであると考えられる．

Java ラッパーでの処理に要した時間は送信時では，約 0.13 ミリ秒，受信時では約 0.16 ミリ秒であった．

5 まとめ

本稿では，NTMobile フレームワーク用の Java ラッパーを実現する方式を提案した．UDP 通信部分の実装を行い，Java 標準 API を使用して NTMobile 通信を実現できることを確認した．今後は，TCP 通信部分の実装検討及び性能評価を行う予定である．

参考文献

[1] 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，pp2288-2299，2013.

[2] K.Naito，K.Kamienoo，H.Suzuki，A.Watanabe，K.Mori and H.Koboyashi，”End-to-end IP mobility platform in application layer for iOS and Android OS”，IEEE CCNC 2014，pp.276-281，2014.

NTMobileフレームワーク用 Javaラッパーの提案と実装

清水 一輝† 八里 栄輔‡ 納堂 博史†
鈴木 秀和† 内藤 克浩§ 渡邊 晃†

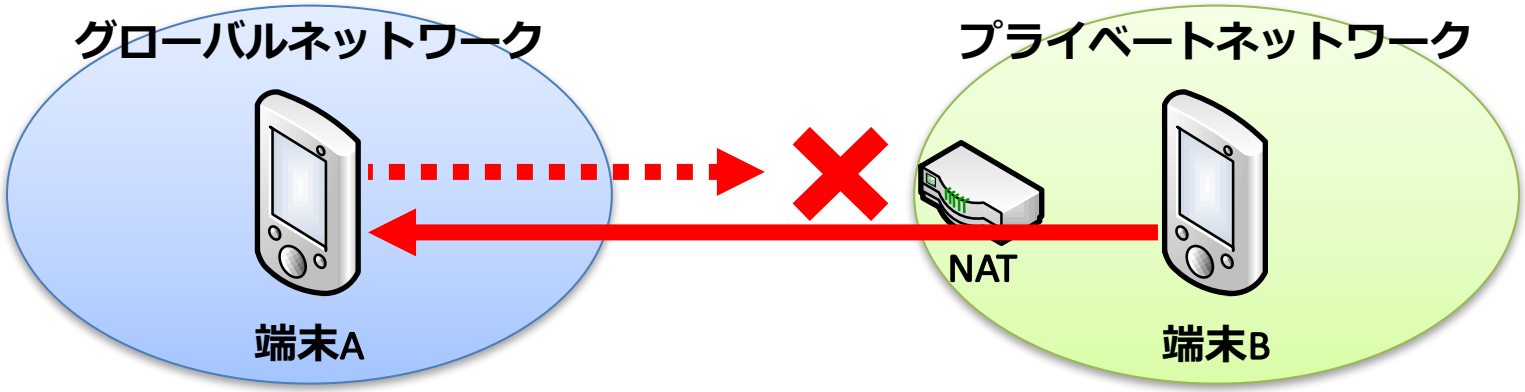
†名城大学

‡バレイキャンパスジャパン

§愛知工業大学

研究背景

- インターネット通信の需要増加
  - スマートフォンなどの移動通信端末の普及
- 通信接続性の課題
  - IPv4アドレスの枯渇に伴い，NATによるプライベートネットワークを構築することが一般的
  - NATの外側にあるネットワークから，NATの内側にあるネットワークにアクセスできない（NAT越え問題）



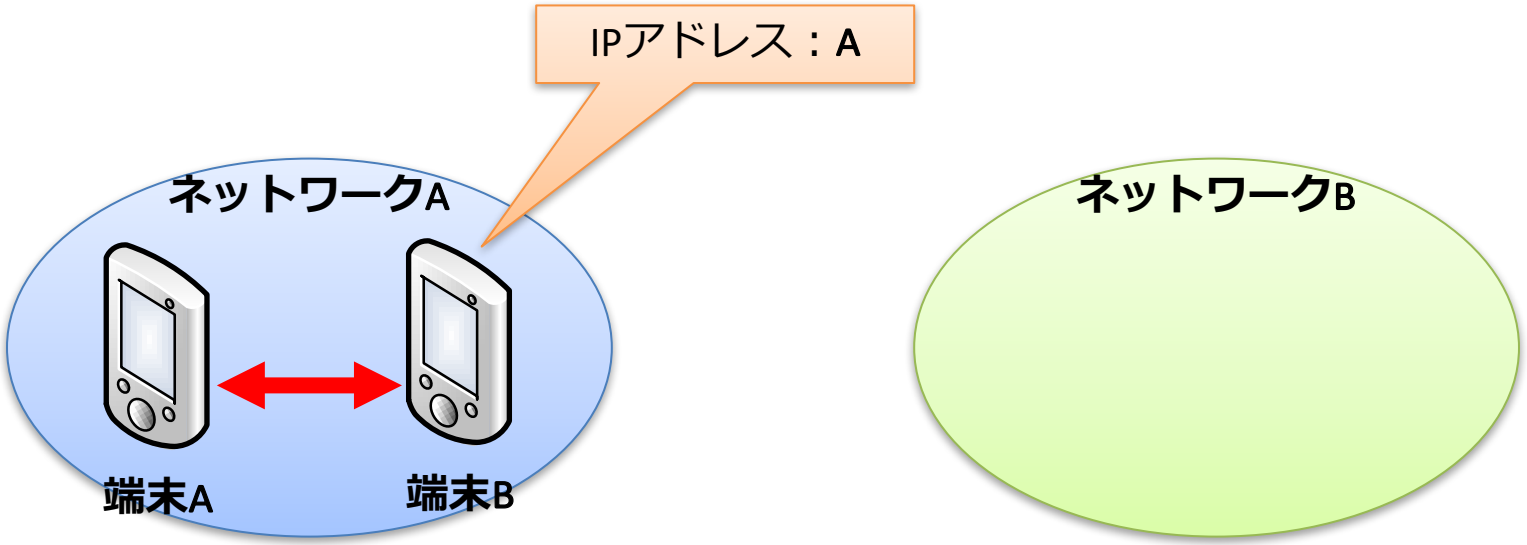
端末間で自由に通信できない

NAT：Network Address Translation

研究背景

移動透過性の課題

- 現在のネットワークでは，IPアドレスを通信識別子としている
- 端末移動時などにネットワークが切り替わると，
  端末のIPアドレスが変化し，通信を継続できない

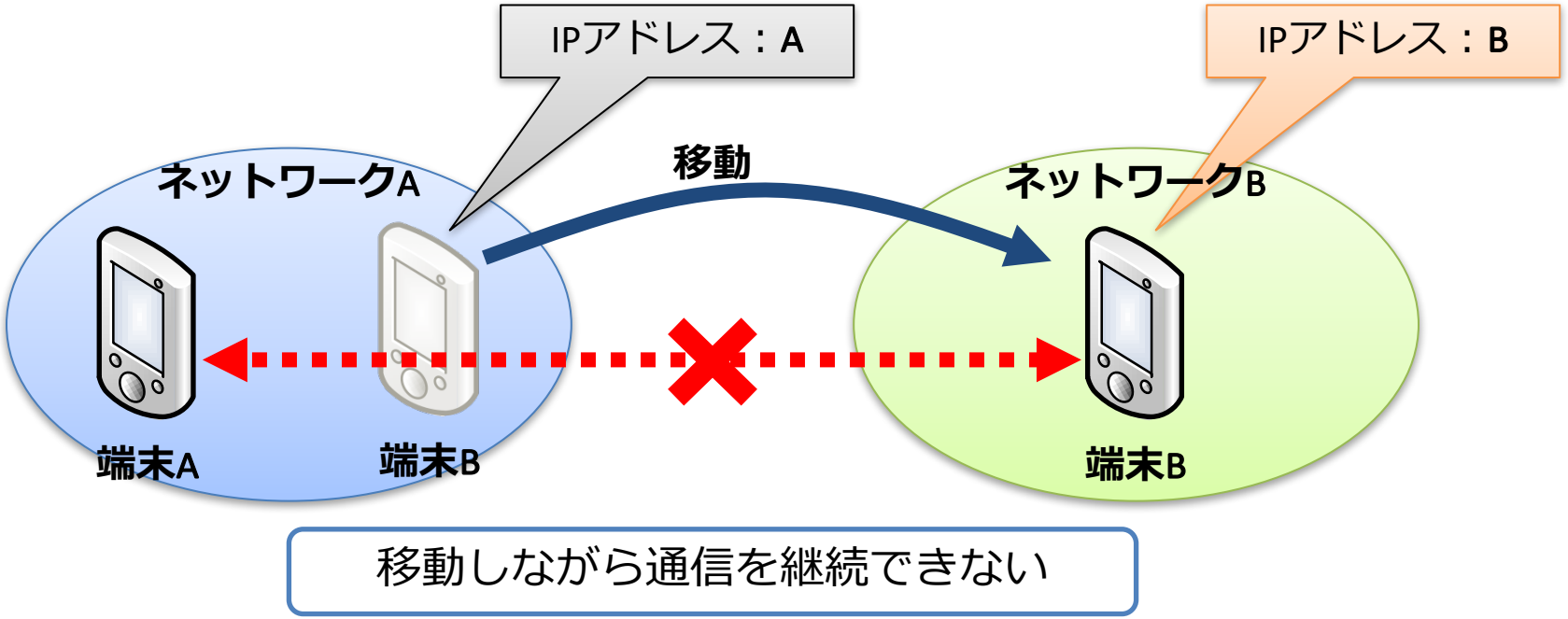


移動しながら通信を継続できない

研究背景

移動透過性の課題

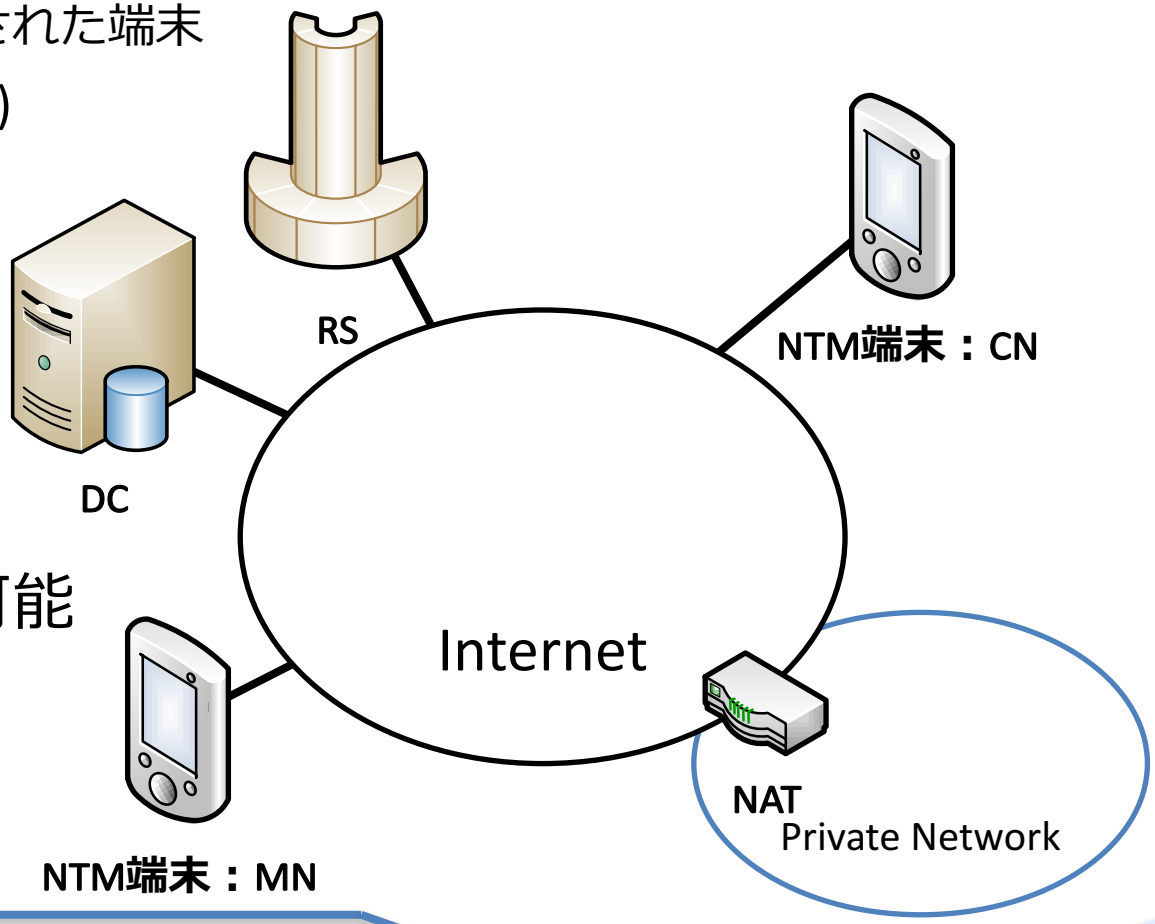
- 現在のネットワークでは，IPアドレスを通信識別子としている
- 端末移動時などにネットワークが切り替わると，
  端末のIPアドレスが変化し，通信を継続できない



NTMobile (Network Traversal with Mobility)

Watanabe Lab.

- 通信接続性と移動透過性を**同時**に実現する技術
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

仮想IPアドレス

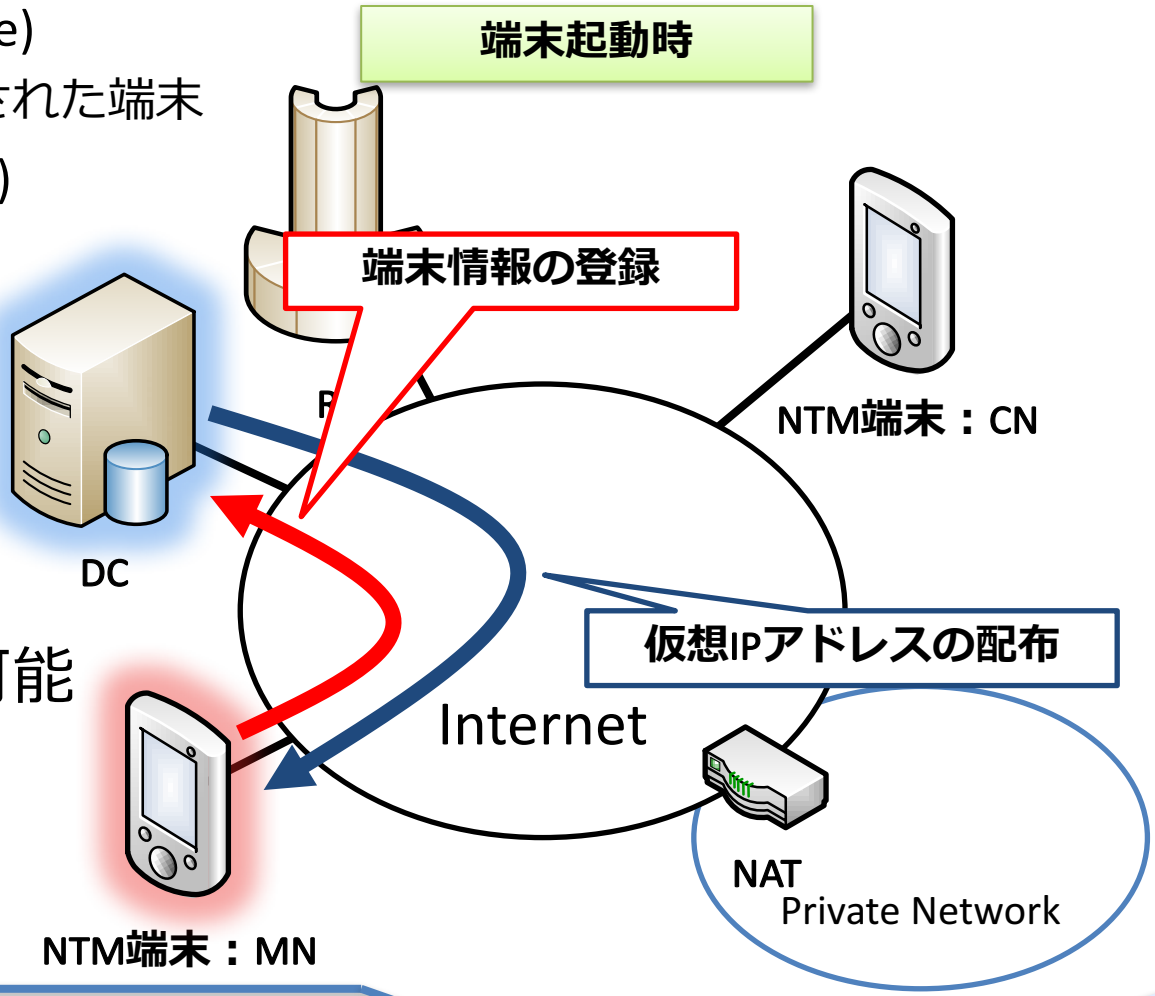
・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

NTMobile (Network Traversal with Mobility)

Watanabe Lab.

- 通信接続性と移動透過性を**同時**に実現する技術
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

仮想IPアドレス

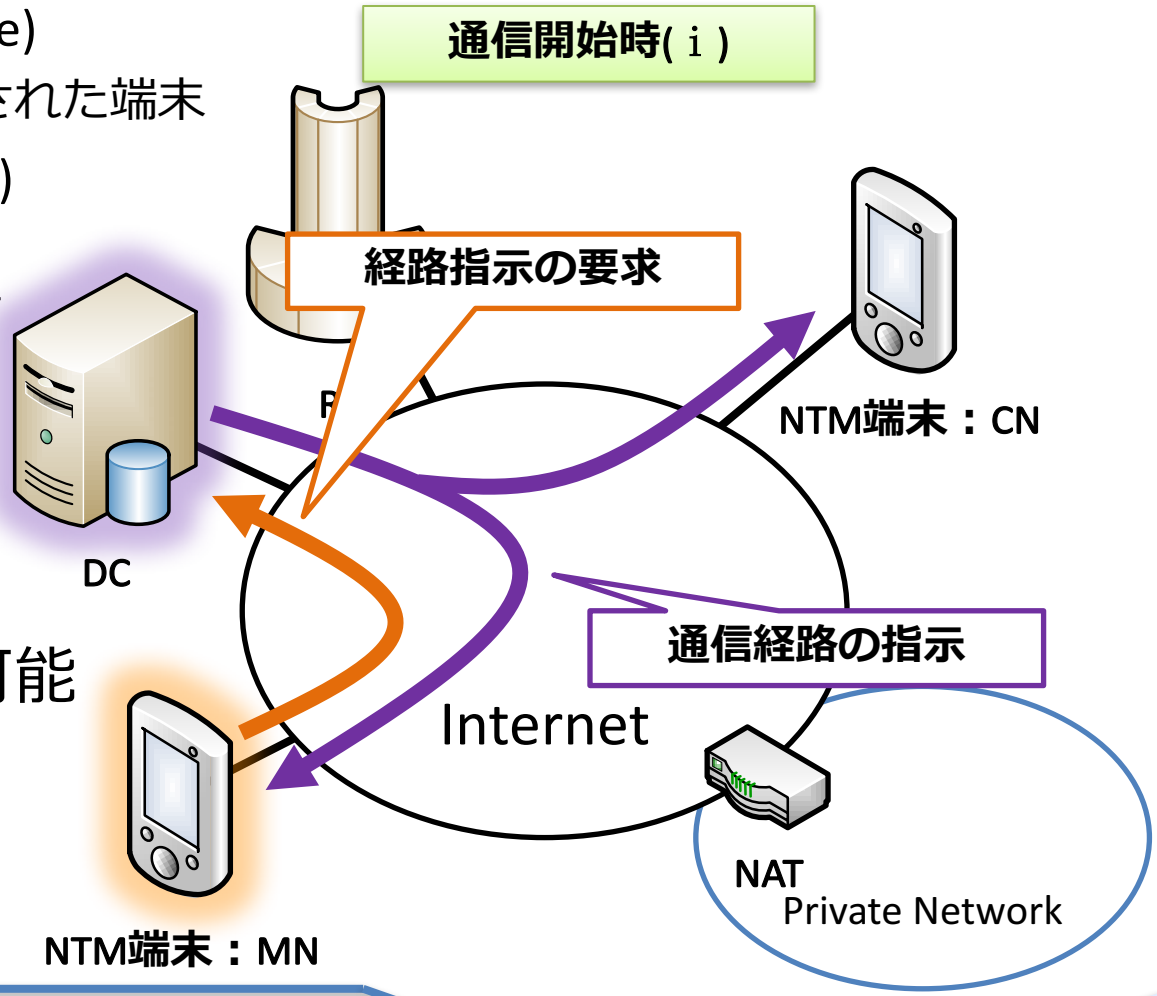
・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

NTMobile (Network Traversal with Mobility)

Watanabe Lab.

- 通信接続性と移動透過性を**同時**に実現する技術
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

仮想IPアドレス

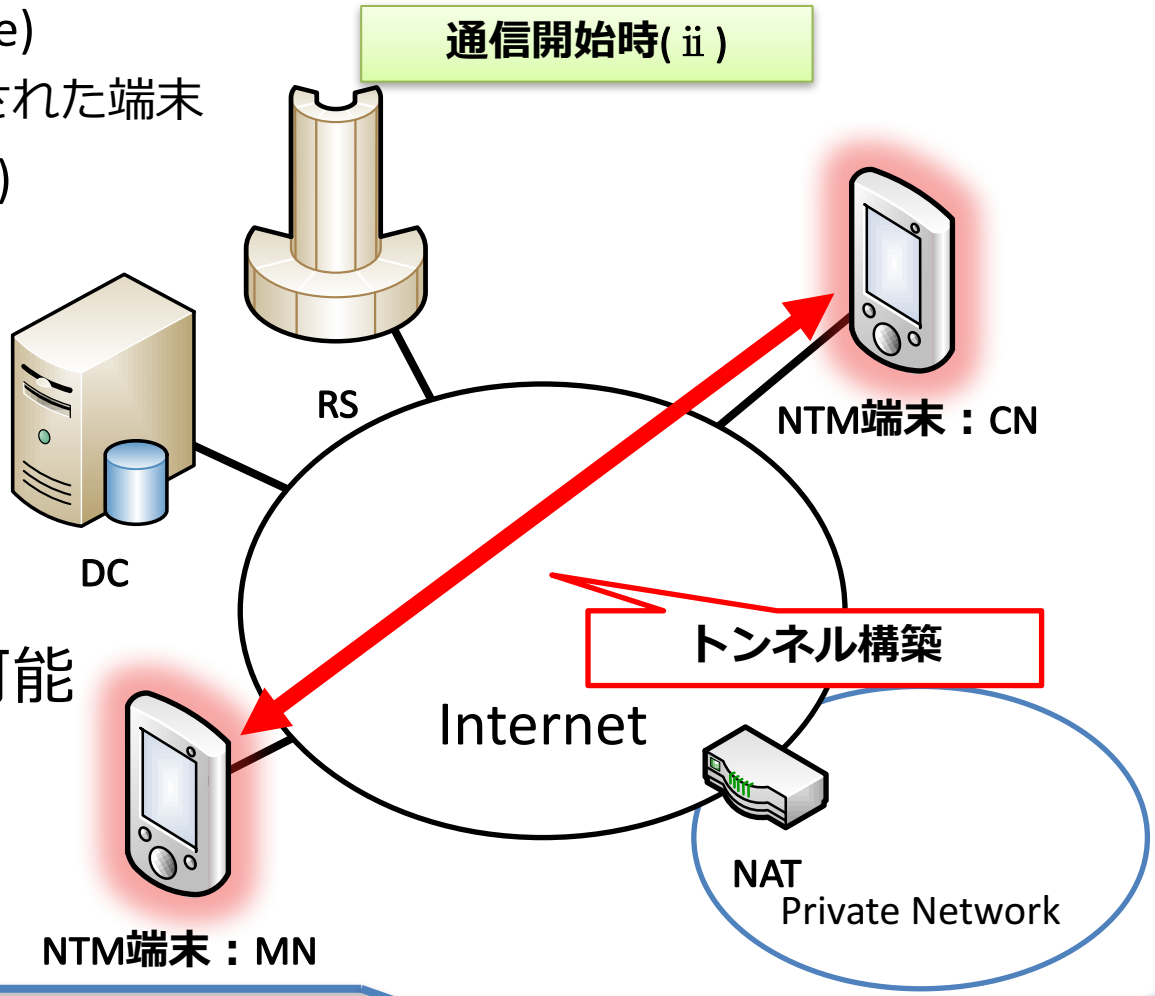
・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

NTMobile (Network Traversal with Mobility)

Watanabe Lab.

- 通信接続性と移動透過性を**同時**に実現する技術
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

仮想IPアドレス

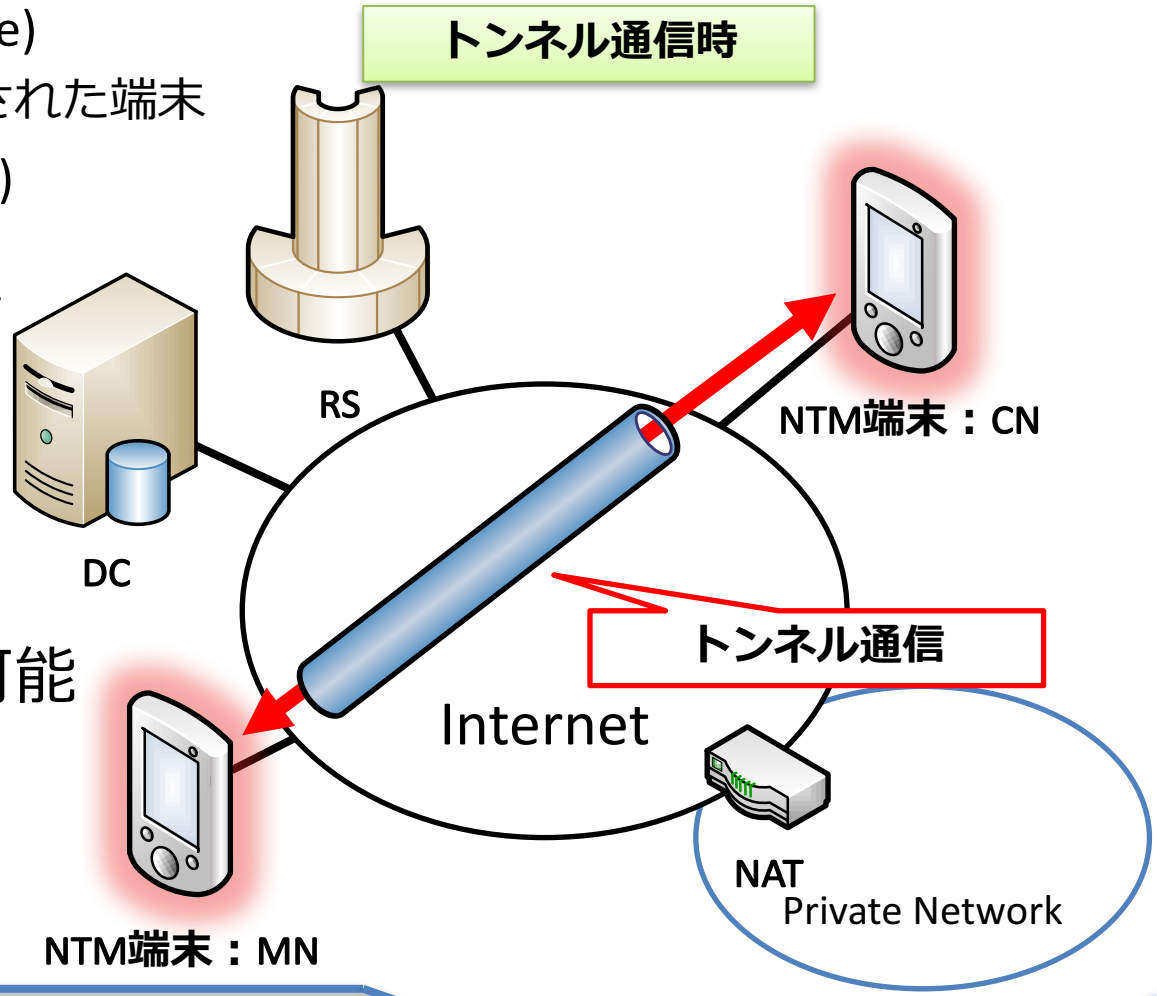
・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

NTMobile (Network Traversal with Mobility)

Watanabe Lab.

- 通信接続性と移動透過性を**同時**に実現する技術
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

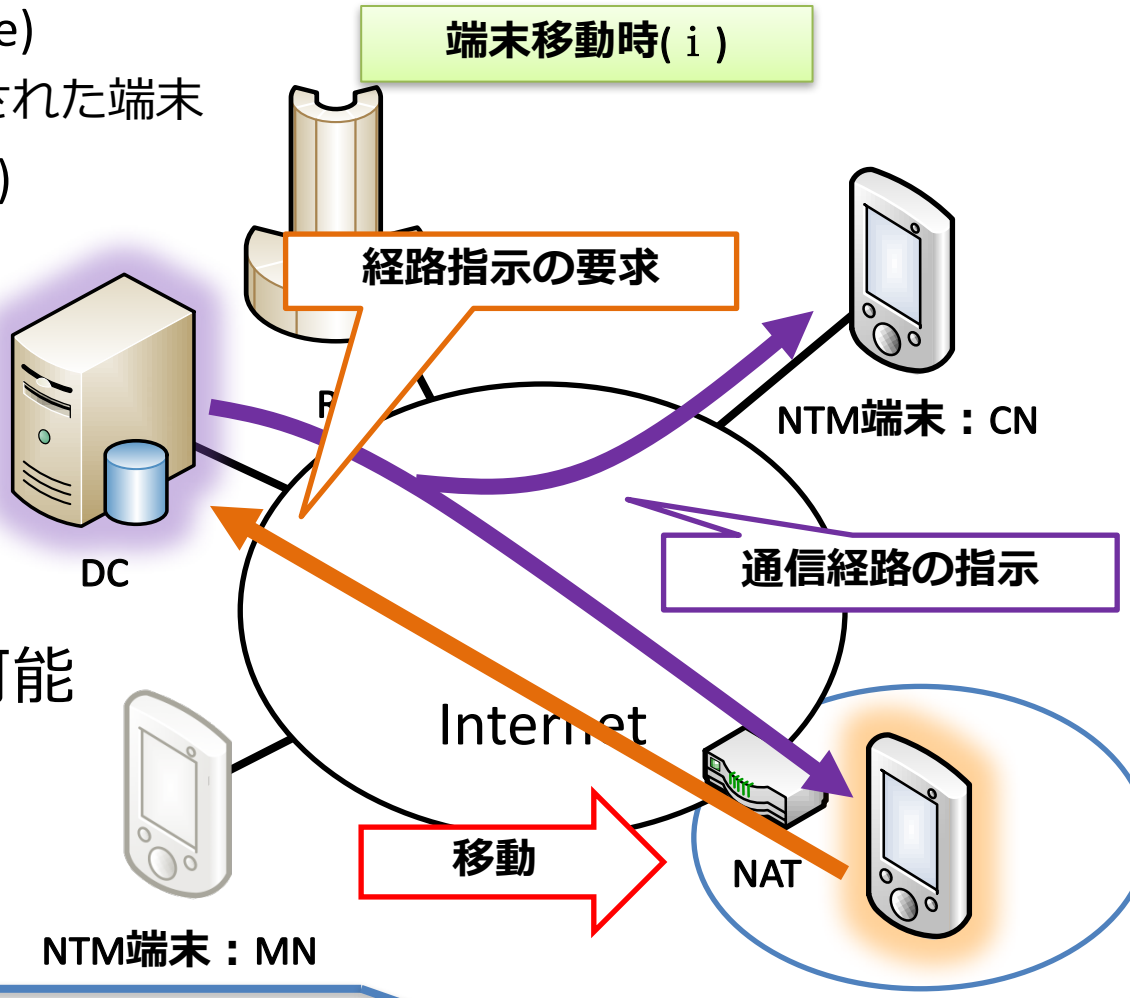
仮想IPアドレス

・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

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
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

仮想IPアドレス

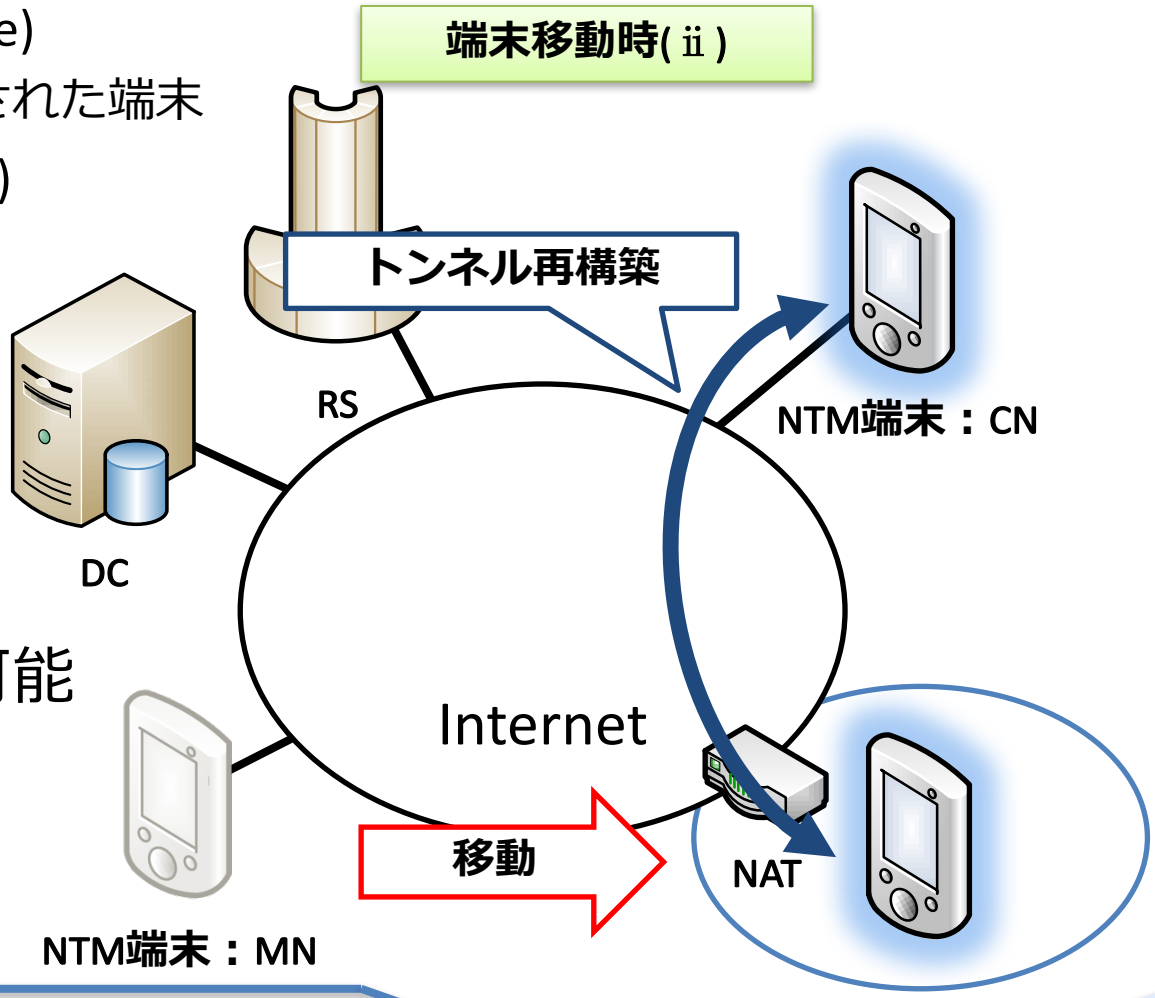
・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

NTMobile (Network Traversal with Mobility)

Watanabe Lab.

- 通信接続性と移動透過性を**同時**に実現する技術
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

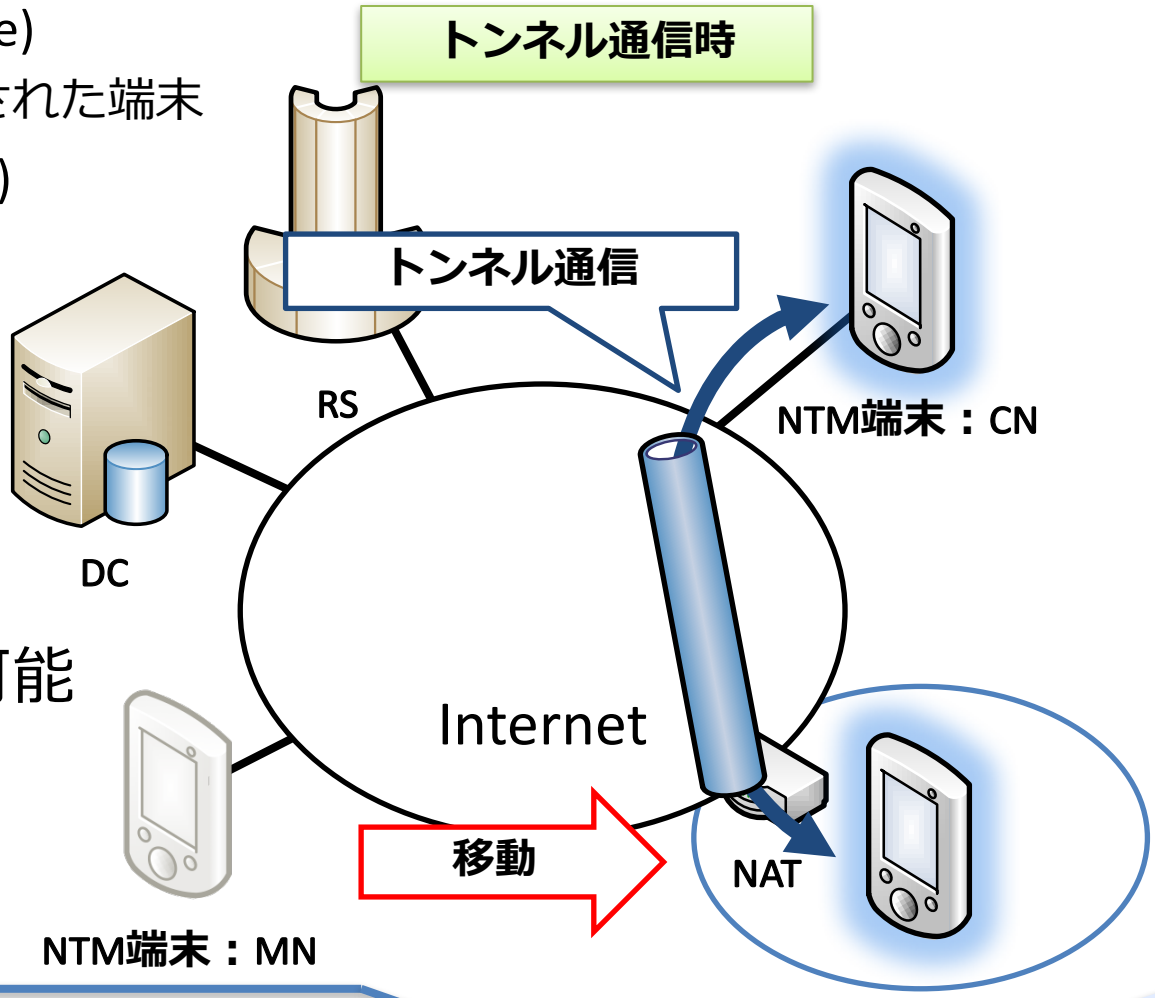
仮想IPアドレス

・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

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
  - NTM端末(NTMobile Node)
    - NTMobile機能が実装された端末
  - DC(Direction Coordinator)
    - 通信経路の指示
    - **仮想IPアドレス**の配布
  - RS(Relay Server)
    - 直接通信不可の際，通信を中継



- DC，RSは複数台設置可能

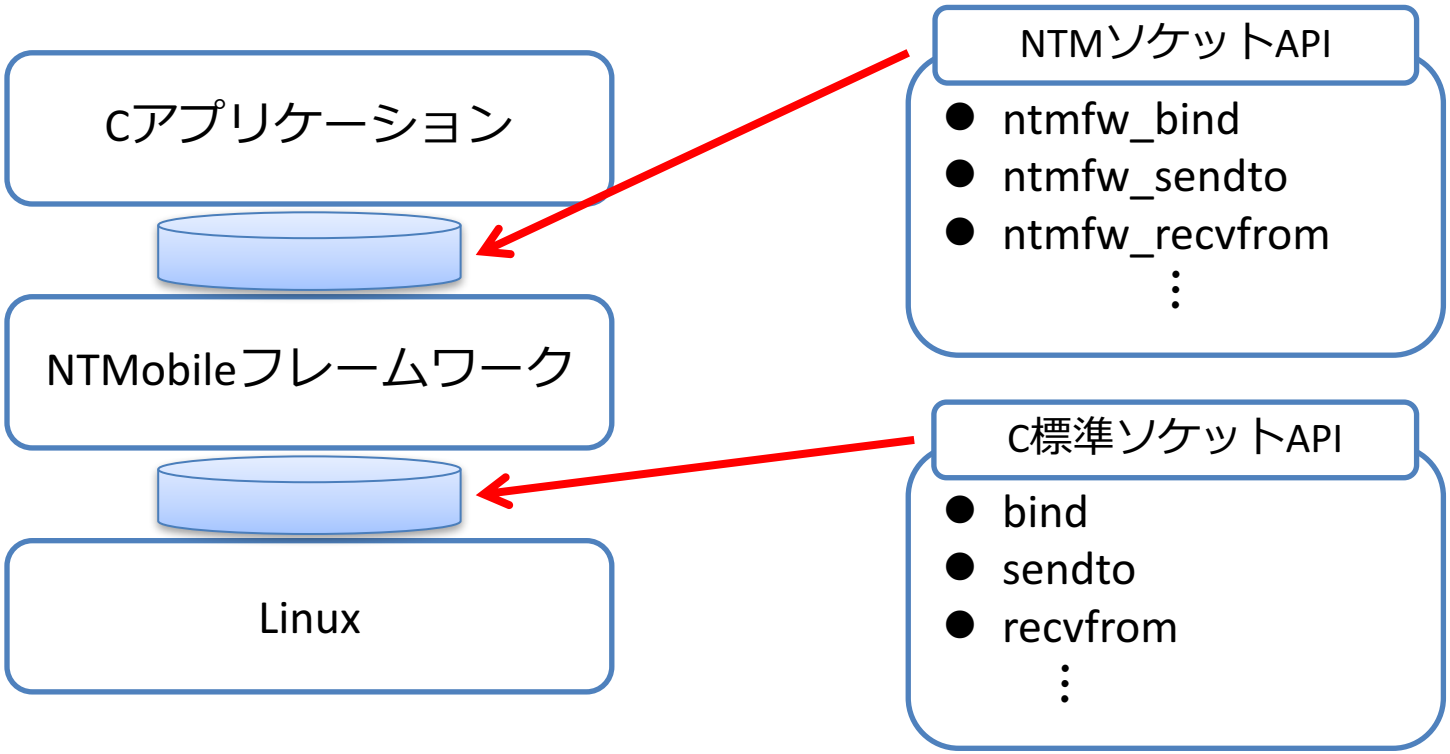
仮想IPアドレス

・端末の位置に依存しない
・実IPアドレスの変化を隠蔽

MN：Mobile Node　CN：Correspondent Node

NTMobileフレームワーク

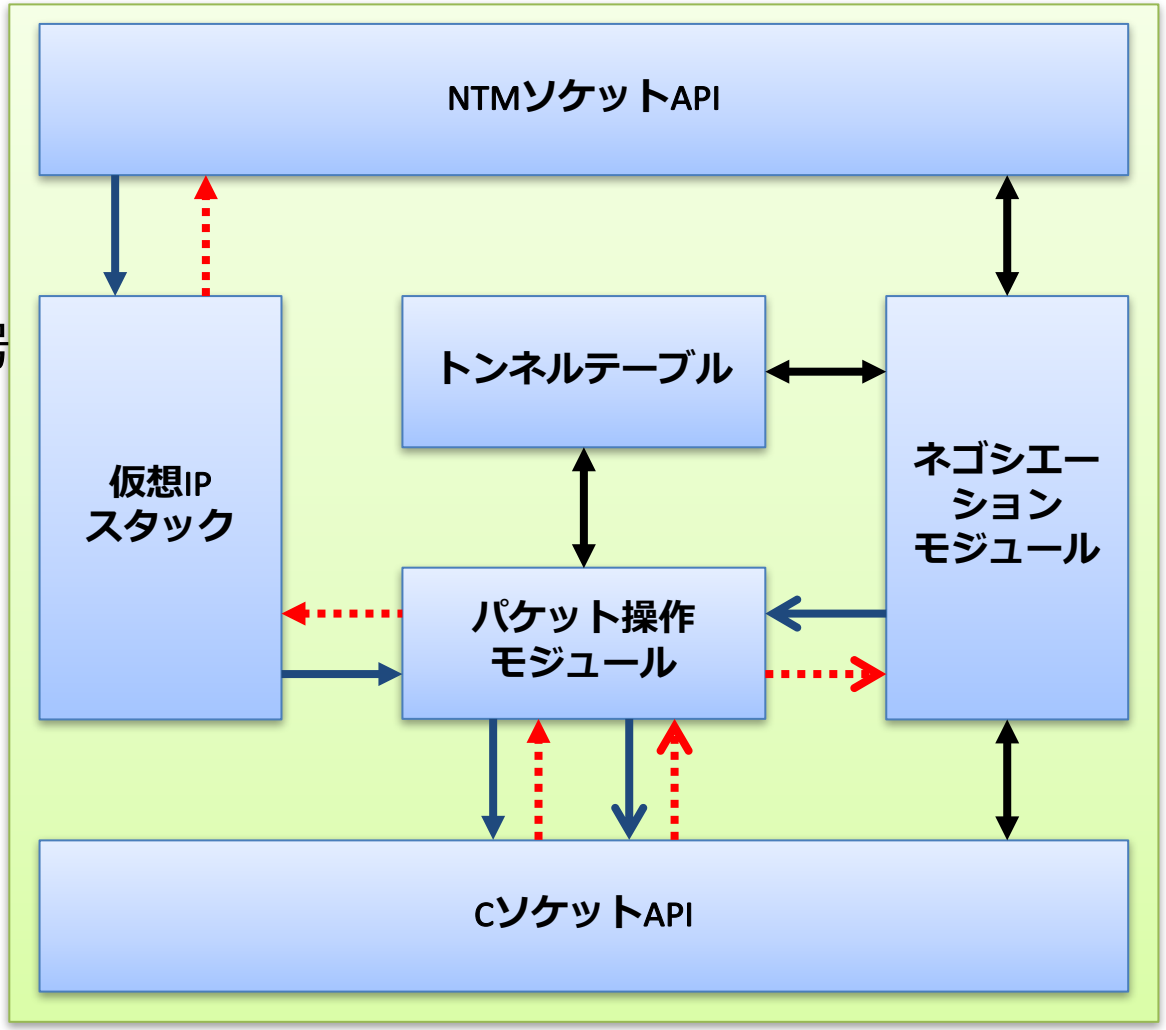
- NTMobile機能をユーザ空間にて実現する実装方式
- アプリケーションはC言語の標準ソケットAPIに代わり，NTMソケットAPIを使用する



API：Application Programming Interface

NTMobileフレームワークの構成

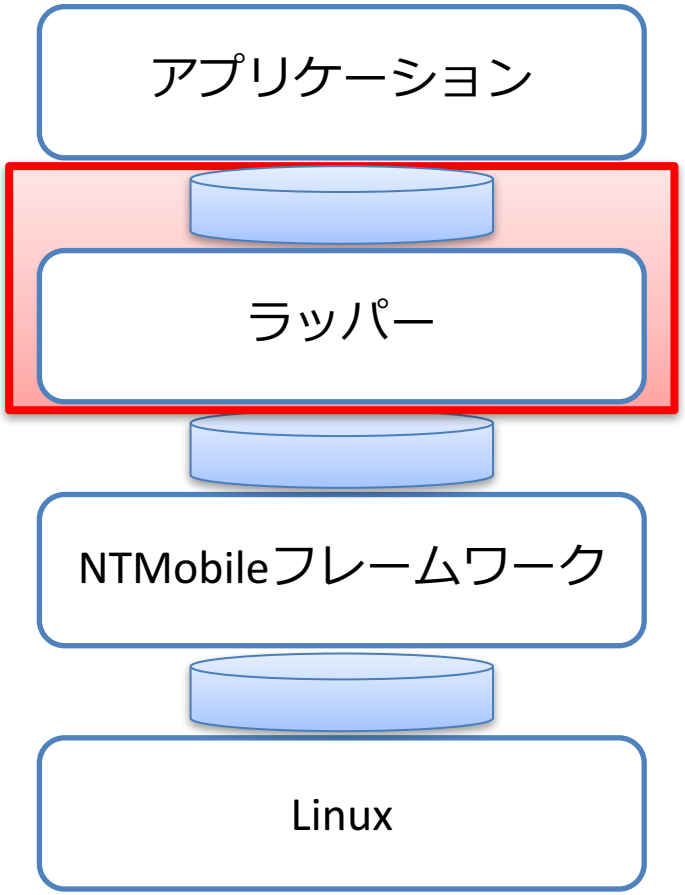
- 仮想IPスタック
  - 仮想IPアドレスの提供
- パケット操作モジュール
  - パケットの暗号化/復号
  - 改ざん検知のためのMAC付与/検証
- トンネルテーブル
  - 実/仮想IPアドレス等の関係を所持
- ネゴシエーションモジュール
  - 通信経路のやり取り



MAC：Message Authentication Code

NTMobileフレームワーク

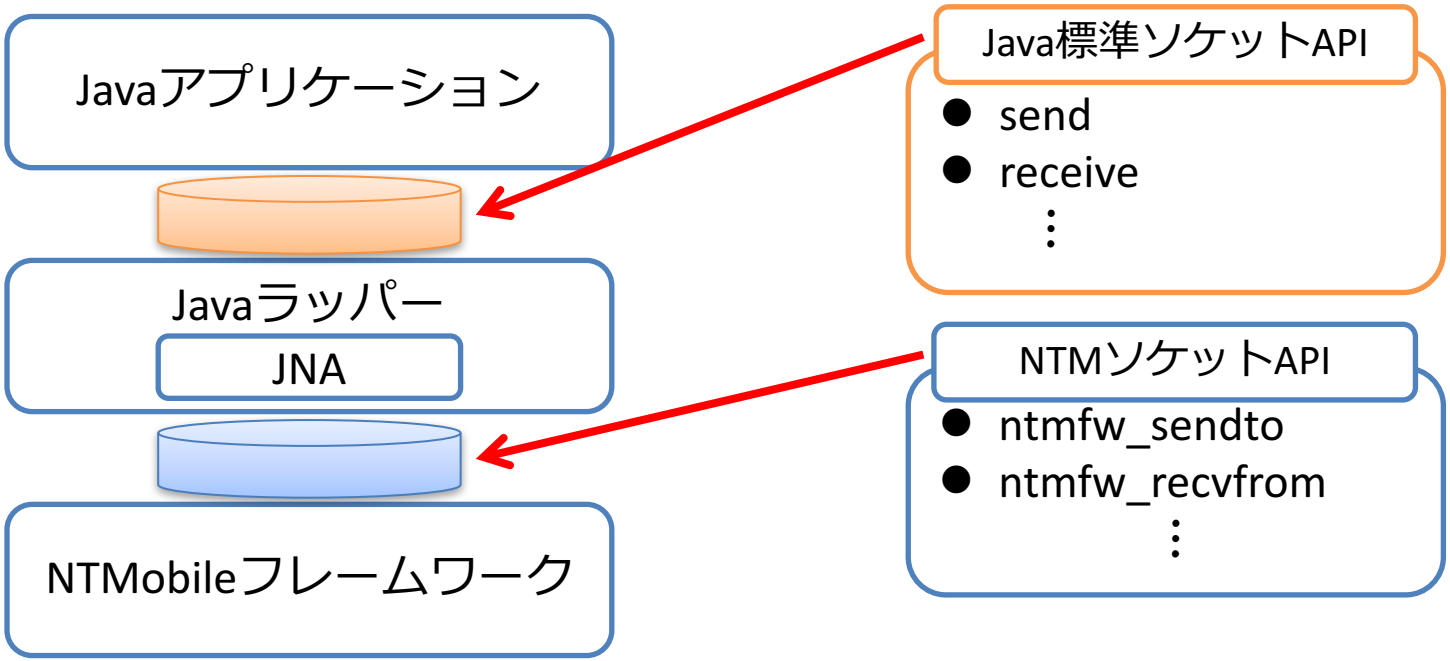
- フレームワークはc言語によって実装
  - フレームワークを利用できるアプリケーションはc言語に限定
  - 実用的な利用のためには他の
    プログラミング言語からフレームワークを
    利用可能にするための**ラッパー**が必要



- **ラッパー**とは
  - 他のプログラミング言語にて
    実装された機能などを
    利用できるようにするもの

NTMobile用Javaラッパー

- Javaラッパー内での処理内容
  - ラッパーで使用するソケットAPIのマッピング
  - C言語とJavaでの違いの除去
  - Java標準ソケットAPIをNTMobile通信を行うように再定義



NTMobile用Javaラッパー

Watanabe Lab.

- ラッパーで使用するソケットAPIのマッピング
  - JNAを使用する

JNA(Java Native Access)

・オープンソースのライブラリ
・C言語のライブラリへアクセスする手法を提供
・C言語と同じ名前でAPIを定義することでJavaで使用可能になる

- マッピング例
  - UDP送信用のNTMソケットAPI
    - C言語
      - ntmfw_sendto(NTM_SOCK fd, const void *buff, NTM_DATALEN buff_len,
        int32_t flags, struct sockaddr *addr, NTM_SOCKLEN addr_len)
    - Java
      - ntmfw_sendto(int fd, Pointer buff, NativeSize buff_len,
        int flags, sockaddr.ByReference addr, int addr_len)

NTMobile用Javaラッパー

- UDP送信用のJava標準ソケットAPIとの比較
  - マッピング後のNTMソケットAPI
    - ntmfw_sendto(int fd, Pointer buff, NativeSize buff_len,
      int flags, sockaddr.ByReference addr, int addr_len)
  - Javaの標準ソケットAPI
    - send(DatagramPacket p)

- 異なる点

	マッピング後の NTMソケットAPI	Javaの 標準ソケットAPI
名前	ntmfw_sendto	send
引数の数	6個	1個

NTMソケットAPIとJavaの標準ソケットAPIは大きく異なる

NTMobile用Javaラッパー

Watanabe Lab.

- C言語とJavaでの違いの除去
  - Javaの標準ソケットAPIでNTMソケットAPIを使用可能にする
    - send(DatagramPacket p)で
      ntmfw_sendto(int fd, … , int addr_len)を使用可能にする

1. send(DatagramPacket p)を呼び出す
2. sendの引数で得た“DatagramPacket p”から
   メッセージやIPアドレス等の情報を取り出す
3. 2.で取り出した情報をntmfw_sendtoの引数に渡す
4. ntmfw_sendto(int fd, … , int addr_len)が実行される
5. send(DatagramPacket p)が終了する

Javaの標準ソケットAPIと同じ使い方で
マッピングしたNTMソケットAPIが使用可能になる

NTMobile用Javaラッパー

Watanabe Lab.

- C言語とJavaでの違いの除去
  - Javaの標準ソケットAPIでNTMソケットAPIを使用可能にする
    - send(DatagramPacket p)で
      ntmfw_sendto(int fd, … , int addr_len)を使用可能にする

1. **send(DatagramPacket p)を呼び出す**
2. sendの引数で得た“DatagramPacket p”から
   メッセージやIPアドレス等の情報を取り出す
3. 2.で取り出した情報をntmfw_sendtoの引数に渡す
4. ntmfw_sendto(int fd, … , int addr_len)が実行される
5. send(DatagramPacket p)が終了する

Javaの標準ソケットAPIと同じ使い方で
マッピングしたNTMソケットAPIが使用可能になる

NTMobile用Javaラッパー

Watanabe Lab.

- C言語とJavaでの違いの除去
  - Javaの標準ソケットAPIでNTMソケットAPIを使用可能にする
    - send(DatagramPacket p)で
      ntmfw_sendto(int fd, … , int addr_len)を使用可能にする

1. send(DatagramPacket p)を呼び出す
2. **sendの引数で得た“DatagramPacket p”から**
   メッセージやIPアドレス等の情報を取り出す
3. 2.で取り出した情報をntmfw_sendtoの引数に渡す
4. ntmfw_sendto(int fd, … , int addr_len)が実行される
5. send(DatagramPacket p)が終了する

Javaの標準ソケットAPIと同じ使い方で
マッピングしたNTMソケットAPIが使用可能になる

NTMobile用Javaラッパー

Watanabe Lab.

- C言語とJavaでの違いの除去
  - Javaの標準ソケットAPIでNTMソケットAPIを使用可能にする
    - send(DatagramPacket p)で
      ntmfw_sendto(int fd, … , int addr_len)を使用可能にする

1. send(DatagramPacket p)を呼び出す
2. sendの引数で得た“DatagramPacket p”から
   メッセージやIPアドレス等の情報を取り出す
3. **2.で取り出した情報をntmfw_sendtoの引数に渡す**
4. ntmfw_sendto(int fd, … , int addr_len)が実行される
5. send(DatagramPacket p)が終了する

Javaの標準ソケットAPIと同じ使い方で
マッピングしたNTMソケットAPIが使用可能になる

NTMobile用Javaラッパー

Watanabe Lab.

- C言語とJavaでの違いの除去
  - Javaの標準ソケットAPIでNTMソケットAPIを使用可能にする
    - send(DatagramPacket p)で
      ntmfw_sendto(int fd, … , int addr_len)を使用可能にする

1. send(DatagramPacket p)を呼び出す
2. sendの引数で得た“DatagramPacket p”から
   メッセージやIPアドレス等の情報を取り出す
3. 2.で取り出した情報をntmfw_sendtoの引数に渡す
4. **ntmfw_sendto(int fd, … , int addr_len)が実行される**
5. send(DatagramPacket p)が終了する

Javaの標準ソケットAPIと同じ使い方で
マッピングしたNTMソケットAPIが使用可能になる

NTMobile用Javaラッパー

Watanabe Lab.

- C言語とJavaでの違いの除去
  - Javaの標準ソケットAPIでNTMソケットAPIを使用可能にする
    - send(DatagramPacket p)で
      ntmfw_sendto(int fd, … , int addr_len)を使用可能にする

1. send(DatagramPacket p)を呼び出す
2. sendの引数で得た“DatagramPacket p”から
   メッセージやIPアドレス等の情報を取り出す
3. 2.で取り出した情報をntmfw_sendtoの引数に渡す
4. ntmfw_sendto(int fd, … , int addr_len)が実行される
5. **send(DatagramPacket p)が終了する**

Javaの標準ソケットAPIと同じ使い方で
マッピングしたNTMソケットAPIが使用可能になる

NTMobile用Javaラッパー

- Java標準ソケットAPIをNTMobile通信を行うように再定義
  - Javaのソケット実装ファクトリへの対応

Java標準ソケットAPI

- bind
- send
- receive
- ⋮

再定義

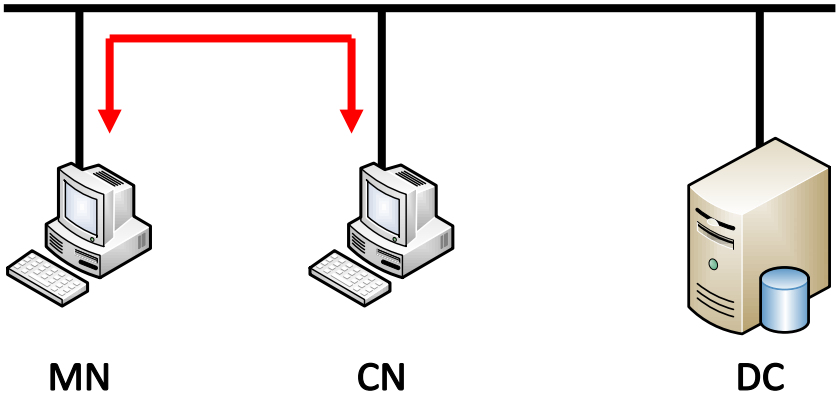
再定義後のソケットAPI

1. 引数で得た情報を
   NTMソケットAPIで
   使えるように処理する
2. 処理後の値を
   NTMソケットAPIの引数で渡す
3. Java標準ソケットAPIに代わり,
   NTMソケットAPIが処理を行う

一括で再定義することが可能

動作検証

- 装置の仕様
  - 全ての装置を1台のホストマシン上に仮想マシンで構築

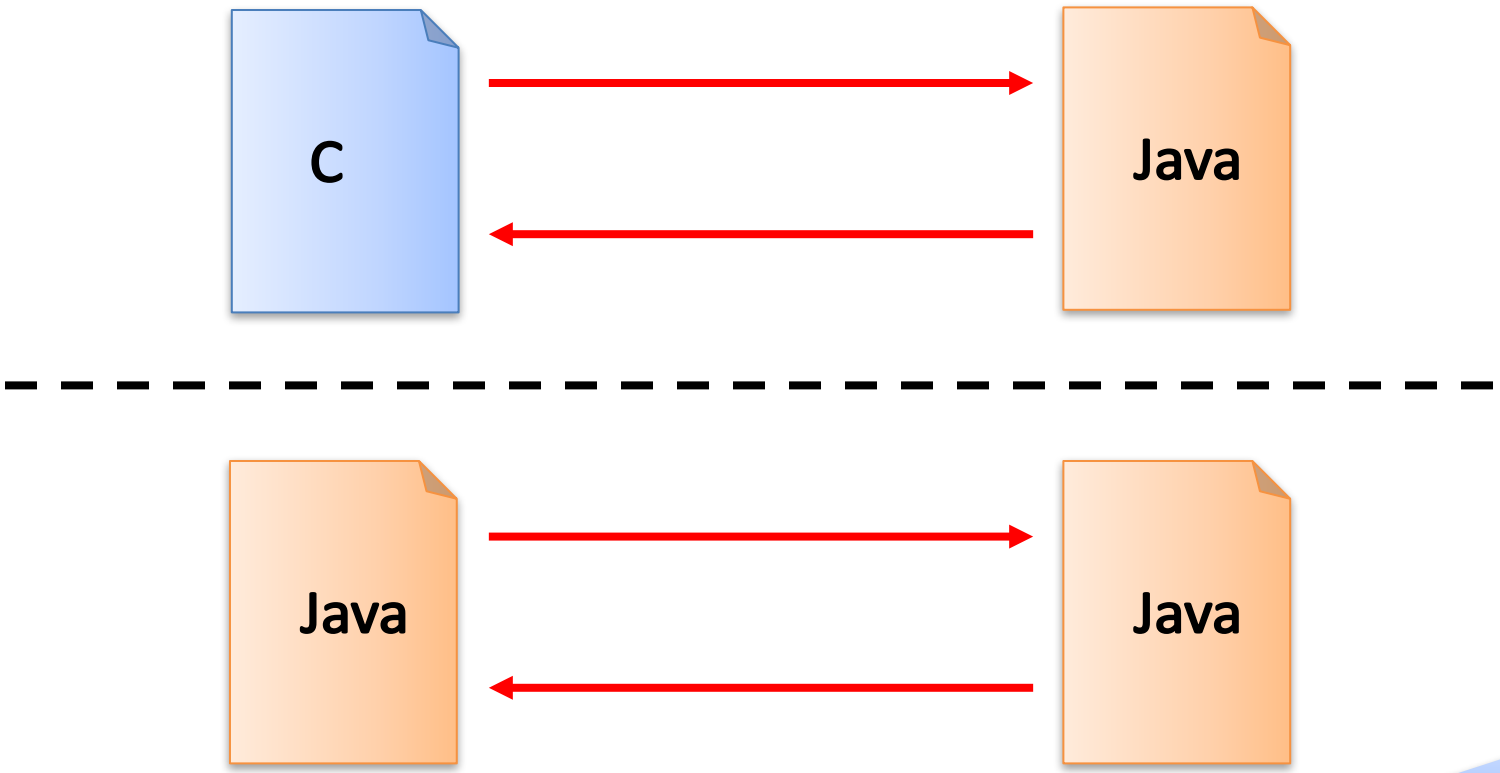


ホストマシン	
OS	Windows 10 64bit
CPU	Intel Core i7-4770 3.40GHz
Memory	8.00GB

仮想マシン	MN，CN	DC
OS	Ubuntu 14.04 32bit	Ubuntu 12.04 32bit
Kernel Version	3.13.0-24-generic	3.2.0-101-generic-pae
CPU割り当て	各1Core	1Core
Memory割り当て	各2.00GB	1.00GB

動作検証

- NTMソケットAPIを使用し，UDPによる任意のメッセージを送受信するアプリケーションをC言語とJavaで作成
- C言語とJava，Java同士の2通りで動作を検証
  - 2通りでUDP送受信の成功を確認

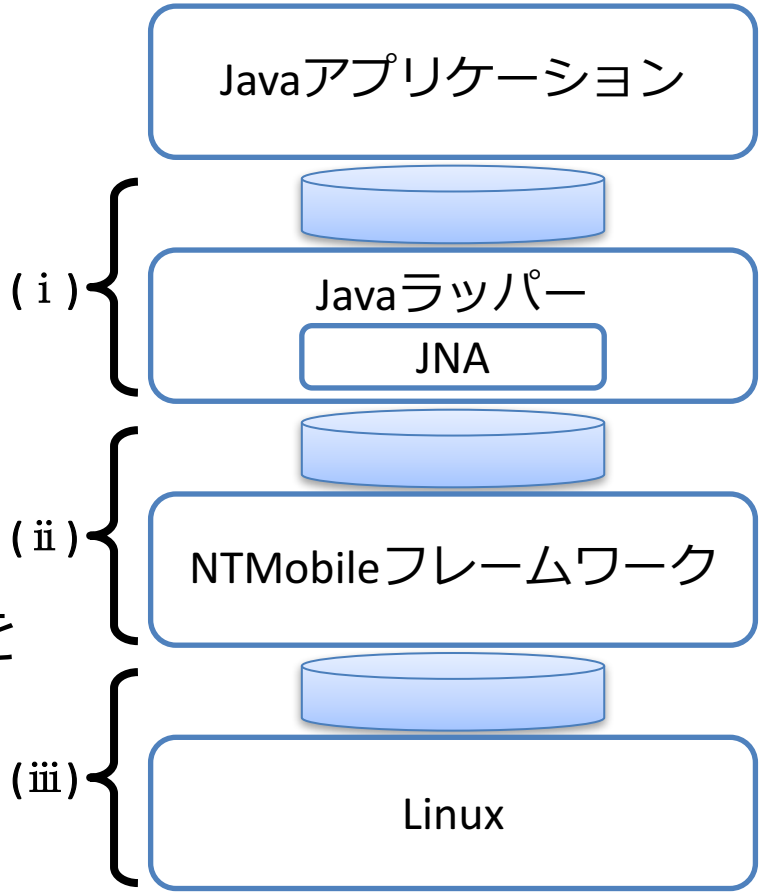


性能評価

- UDP通信時の処理時間を計測
  - 5回の平均を計算

計測箇所	送信時[ms]	受信時[ms]
(ⅰ) Javaラッパー	0.13	0.16
(ⅱ) NTMobile	2.91	2.37
(ⅲ) Linux	0.07	0.01
合計	3.11	2.54

- NTMobileではパケット暗号化/復号，改ざん検知のためにMAC付与/検証を行うため時間を要する



MAC：Message Authentication Code

まとめ

- NTMobile用Javaラッパー
  - NTMobileフレームワークをJavaにて使用可能にする
- 実装と評価
  - 仮想環境にて正常に動作することを確認
    - JavaでNTMobileを使用可能
  - C言語とJavaでの異なるプログラミング言語間による通信も確認
- 今後の予定
  - 既存アプリケーションへの実装