機能拡張した通信ライブラリを呼び出すNode.jsラッパーの検討

清水　一輝 †*，鈴木　秀和 †，内藤　克浩 ‡，渡邊　晃 †
(† 名城大学，‡ 愛知工業大学)

Study of Node.js Wrapper that calls Extended Communication Library

Kazuki Shimizu†*, Hidekazu Suzuki†, Katsuhiro Naito‡, Akira Watanabe†
(†Meijo University, ‡Aichi Institute of Technology)

1　はじめに

ライブラリは処理速度や移植性，他言語との連携といった観点から，C 言語で実装されることが多い．しかし，Web やスマートフォンのアプリケーションは，C 言語よりも抽象度の高い高級言語にて開発されることが一般的である．そのため，アプリケーション開発時に C 言語ライブラリを使用したい場合は，一般的に呼び出し元の言語に応じたラッパーを作成し，ラッパーを経由して C 言語ライブラリを使用する必要がある．

既存のラッパー生成方法では，呼び出し元の言語が呼び出すライブラリの API (Application Programming Interface) を意識する必要がある．しかし，通信ライブラリのようにプログラミング言語の標準 API として備わっている機能の場合は標準 API と同じ使用方法で使用できる方が望ましい．

そこで本稿では，JavaScript の中でもネットワークプログラムを作成する際によく用いられる Node.js にて，エンドツーエンド通信を可能とするライブラリである NTMobile framework Libray (NTMfw) [1] を Node.js の標準 API と同じ使用方法で使用可能とする Node.js ラッパーを検討した．

2　既存技術：SWIG

ラッパーを生成するツールとして SWIG (Simplified Wrapper and Interface Generator) [2] が存在する．しかし，SWIG により生成されたラッパーは，C/C++ ライブラリの API を意識して使用する必要がある．このため，C/C++ ライブラリを使用したことのない開発者が C/C++ ライブラリを意識する必要があるので，開発負荷が高くなる．

通信用 API といった，呼び出し元の標準 API にも同様な機能が用意されている場合は，呼び出し元の標準 API と同じ使用方法でライブラリを使用できることが望ましい．

3　検討方式

<3・1>必要モジュール　ラッパーを作成する際に必要になるモジュールは，通信ライブラリによって提供される通信プロトコルに対応した Node.js の各種通信用モジュールと，C 言語通信ライブラリと Node.js との連携を行うための FFI (Foreign Function Interface) モジュールである．今回対象とする C 言語通信ライブラリである NTMfw によって提供される通信プロトコルは UDP と TCP である．そのため，UDP では dgram モジュール，TCP では net モジュールを継承して NTMfw 用のモジュールを作成する．また，FFI は Node.js に ffi というモジュールが存在するため，このモジュールを使用することで Node.js から NTMfw へのアクセスを実現する．

<3・2>動作フロー　Fig. 1 に Node.js ラッパーのモジュール構成を示す．アプリケーションが NTMfw によって提供されるエンドツーエンド通信を行う場合，UDP による通信であれば `ntmobile_dgram` モジュール，TCP による通信であれば `ntmobile_net` モジュールを使用して通信を行う．

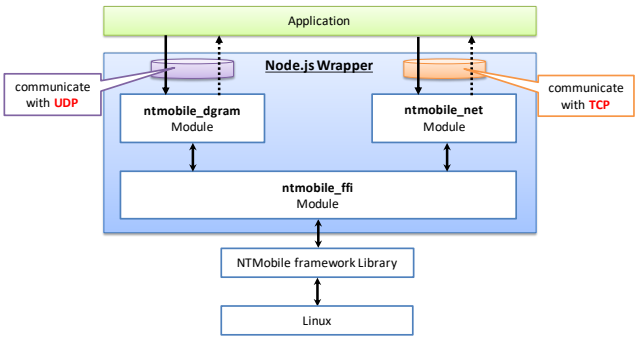


Fig. 1　Module Configuration of Node.js Wraper

`ntmobile_dgram` モジュールと `ntmobile_net` モジュールは，Node.js の標準通信用モジュールである `dgram` モジュールと `net` モジュールを継承し，NTMfw による通信を行うようにオーバーライドする．オーバーライドすることで API の名称や引数は変化しないため，アプリケーション開発者は Node.js 標準の通信用 API と同じ使用方法で NTMfw によるエンドツーエンド通信を実現できる．但し，オーバーライドするにあたって Node.js から得られる引数の値から NTMfw の API が必要とする引数を満たせない場合は，Node.js の言語仕様に応じた値を補完する．また，`ntmobile_dgram` モジュールと `ntmobile_net` モジュールは，処理過程の中で `ntmobile_ffi` モジュールを使用することで Node.js から NTMfw へアクセスする．

`ntmobile_ffi` モジュールでは，C 言語と Node.js における変数の型の違いを取り除いて定義することで，NTMfw へのアクセスを実現する．NTMfw は Linux を通じてパケットの送受信を実行し，結果をラッパーに応答することにより，アプリケーションは結果を知ることができる．

4　まとめ

本稿では，Node.js の標準 API と同じ使用方法でエンドツーエンド通信ライブラリを使用可能にする Node.js ラッパーについて検討した．今後は，HTTP 通信機能追加の検討や検討方式の実装及び性能評価を行う予定である．

文　献

[1] 納堂．他：実用化に向けた NTMobile フレームワークの実装と評価．情報処理学会研究報告，第 82 回 MBL・第 53 回 UBI 合同研究発表会，2017.
[2] SWIG developers : Simplified Wrapper and Interface Generator, http://www.swig.org/.

Watanabe Lab.

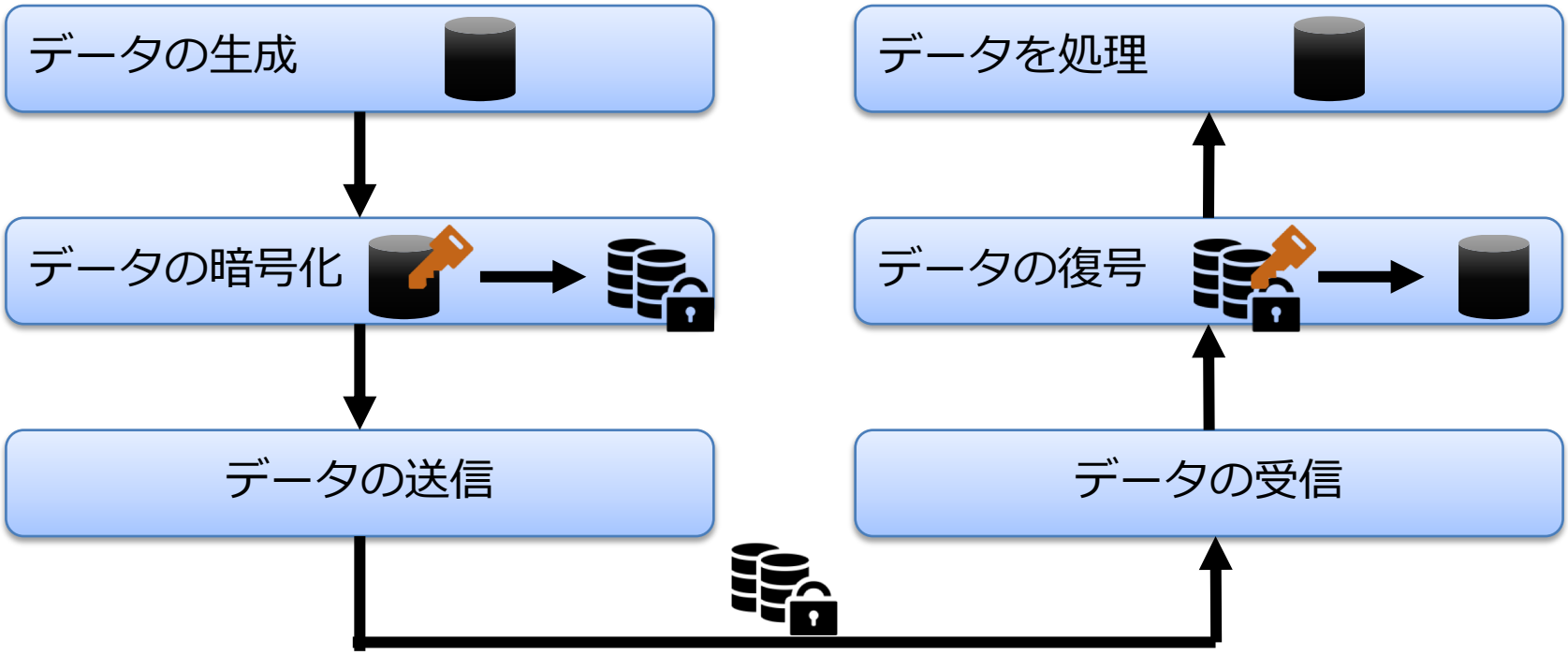
機能拡張した通信ライブラリを呼び出すNode.jsラッパーの検討

清水 一輝†
鈴木 秀和†　内藤 克浩‡　渡邊 晃†

†名城大学
‡愛知工業大学

研究背景

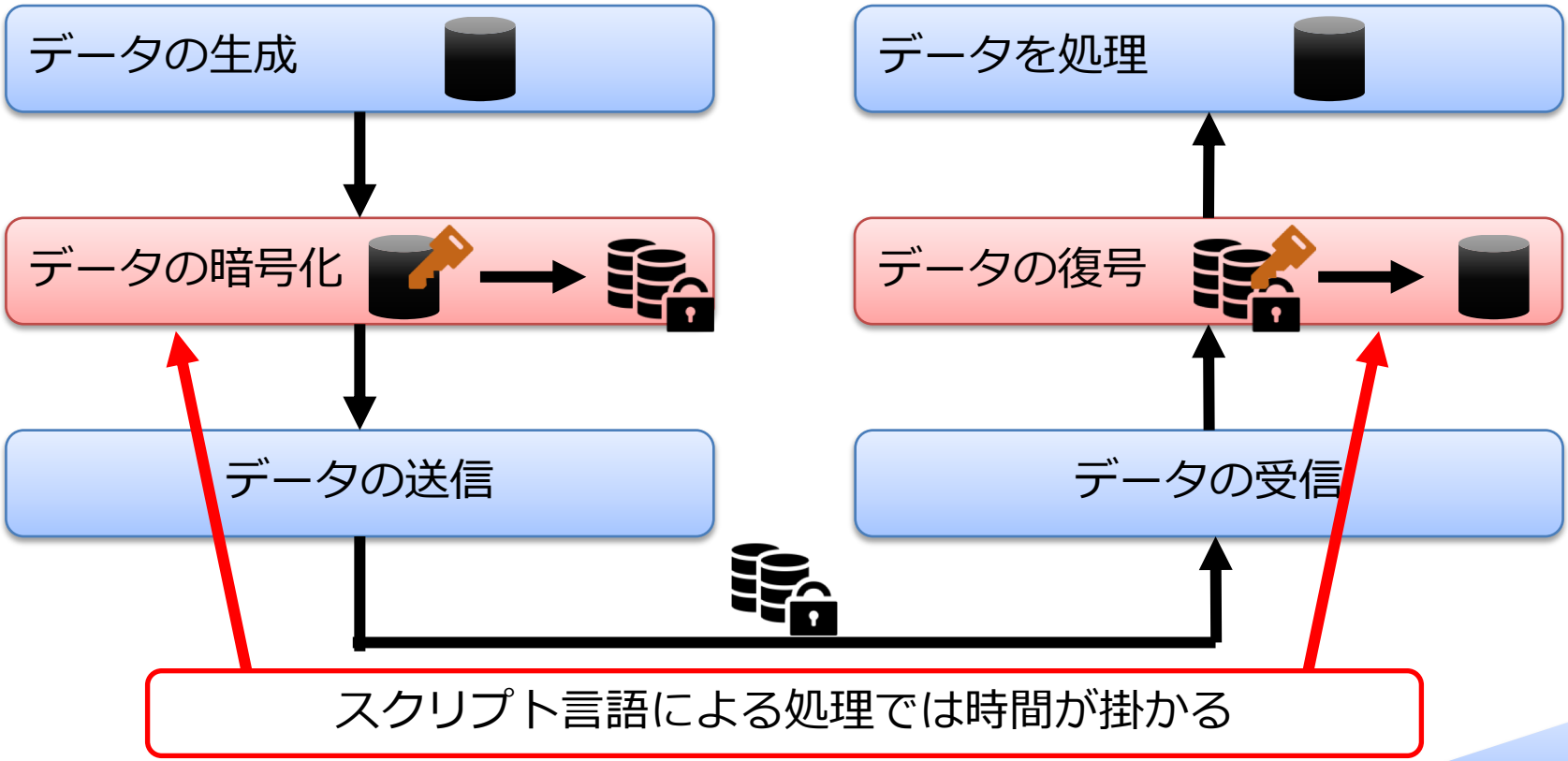
- c言語によるライブラリの実装
  - 実行時における**処理速度**の速さ
  - プラットフォーム依存の言語仕様の少なさによる**移植性**の高さ
  - 他のプログラミング言語との**連携**



：スクリプト言語による実装

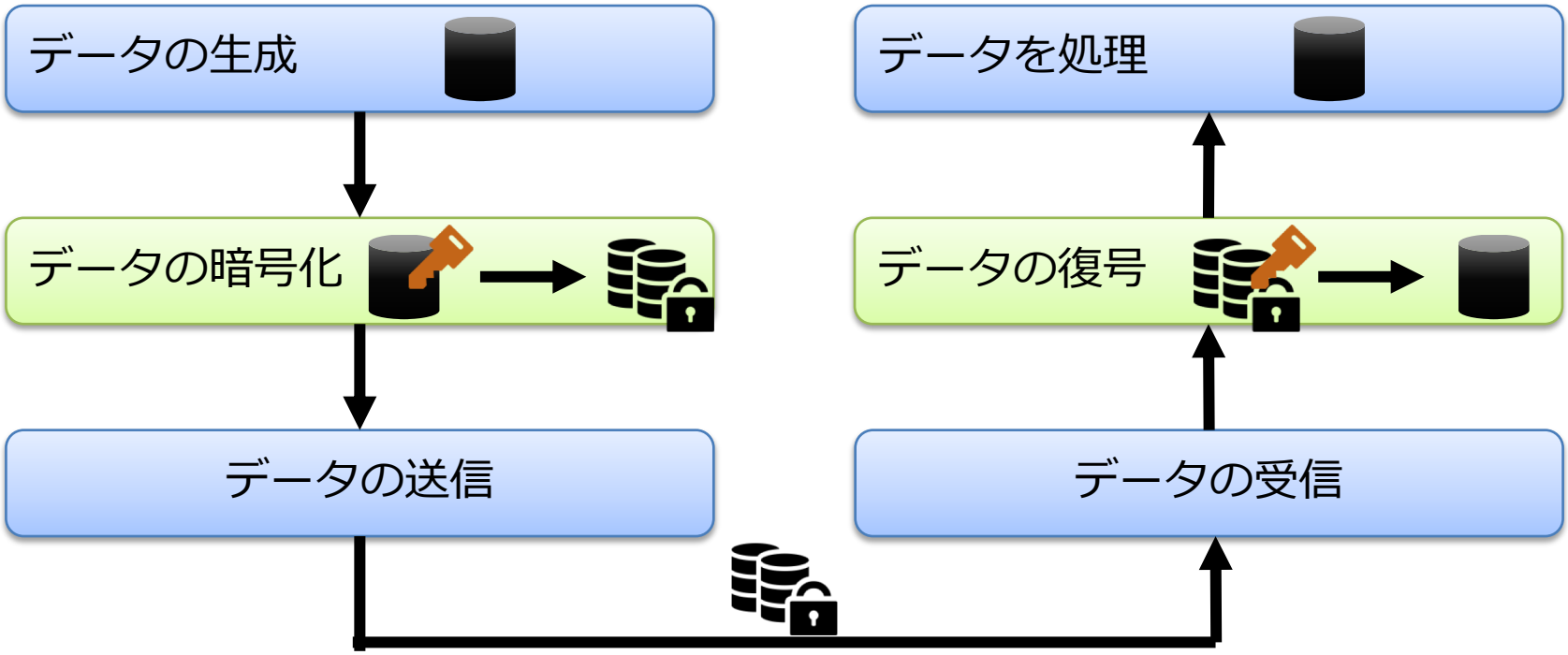
研究背景

- c言語によるライブラリの実装
  - 実行時における**処理速度**の速さ
  - プラットフォーム依存の言語仕様の少なさによる**移植性**の高さ
  - 他のプログラミング言語との**連携**



研究背景

- c言語によるライブラリの実装
  - 実行時における**処理速度**の速さ
  - プラットフォーム依存の言語仕様の少なさによる**移植性**の高さ
  - 他のプログラミング言語との**連携**



：スクリプト言語による実装　　：c言語による実装

研究背景

アプリケーションの開発

- Android OS → Java
- iOS → Swift
- Webアプリ → JavaScript

ラッパーによるコードの統一化

- Android OSとiOSの両方で同じアプリケーションを作成したい場合,
  コアな処理部分をc言語で実装し統一できる

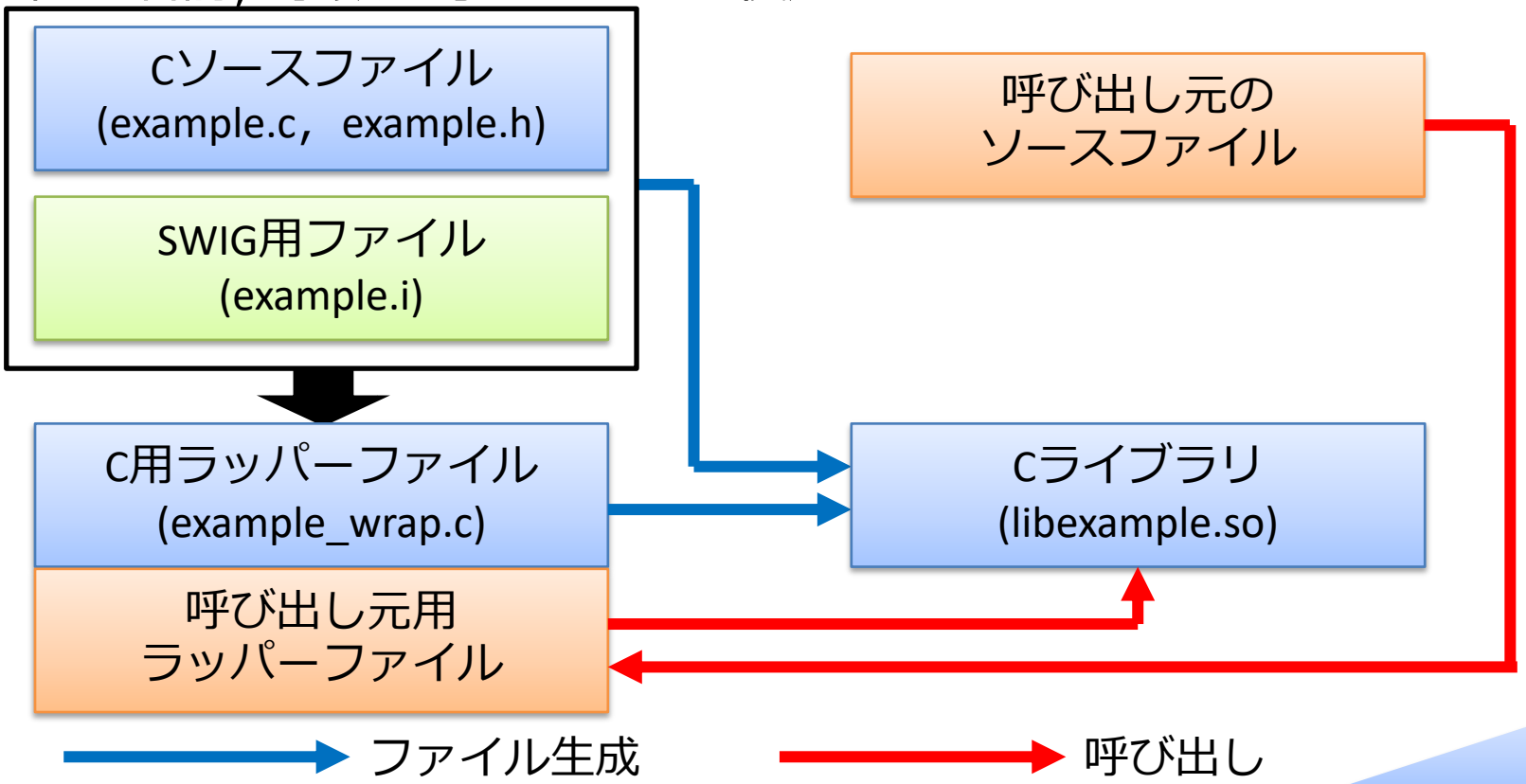
呼び出し元の言語がc言語以外の場合では**ラッパー**が必要

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
  他の言語から利用できるようにするもの

既存技術

- SWIG (Simplified Wrapper and Interface Generator)
  - C/C++ライブラリ用のラッパーを生成するためのツール
  - 生成にはSWIG用の独自記述ファイル (i ファイル)が必要
  - 生成されたラッパーはC/C++ライブラリに実装済みの関数と同じ名前，引数を与えることで使用できる

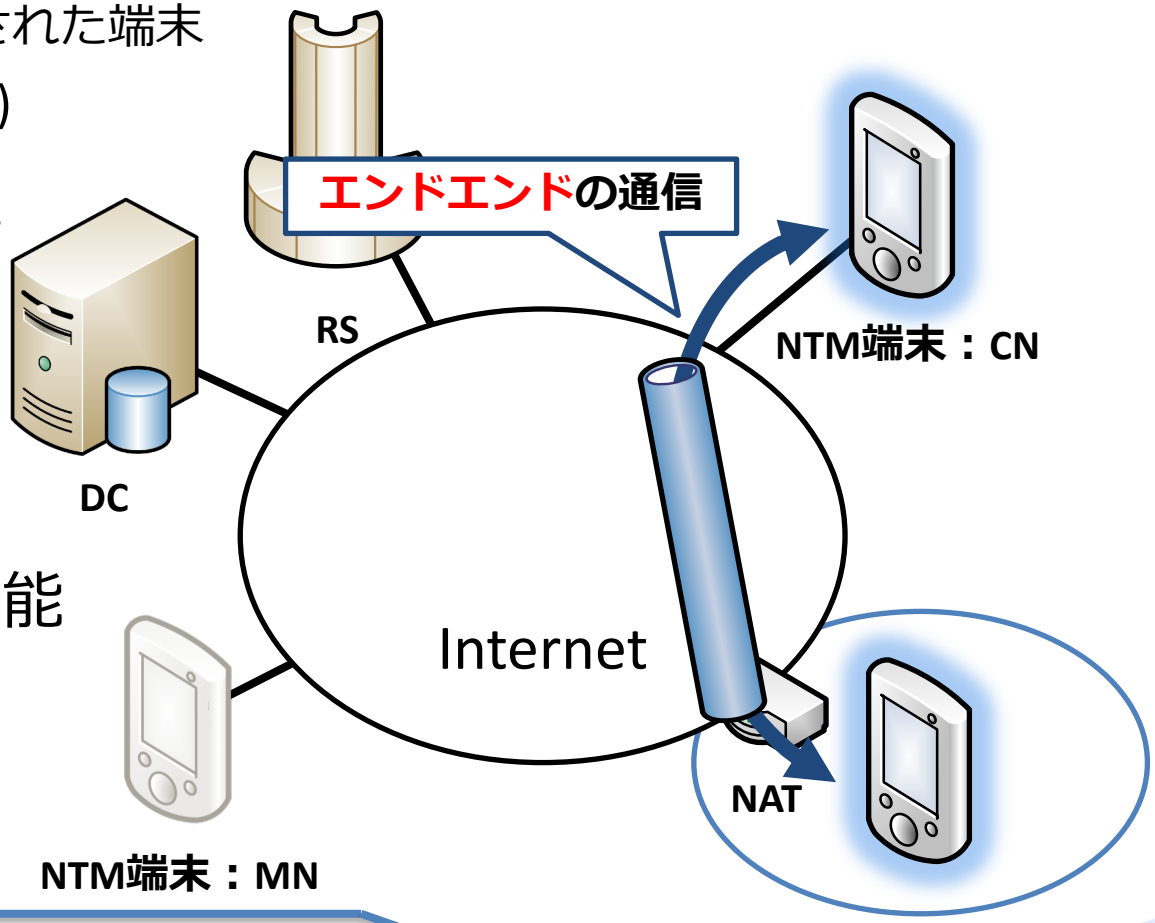


既存技術

- SWIGの課題
  - ライブラリの使用方法が呼び出すC/C++ライブラリに**依存**する
- 呼び出し元の標準ライブラリに**存在しない新機能**の場合
  - Ex.) OpenCVによる**画像の特徴点抽出**
  - 使用方法が元のC/C++ライブラリに依存しても問題ない
- 呼び出し元の標準ライブラリに**存在する機能**の場合
  - Ex.) NTMobileによる移動透過性を実現する**通信**
  - C/C++ライブラリで提供される使用方法ではなく，
    呼び出し元の標準ライブラリと同じ使用方法であるほうが望ましい

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

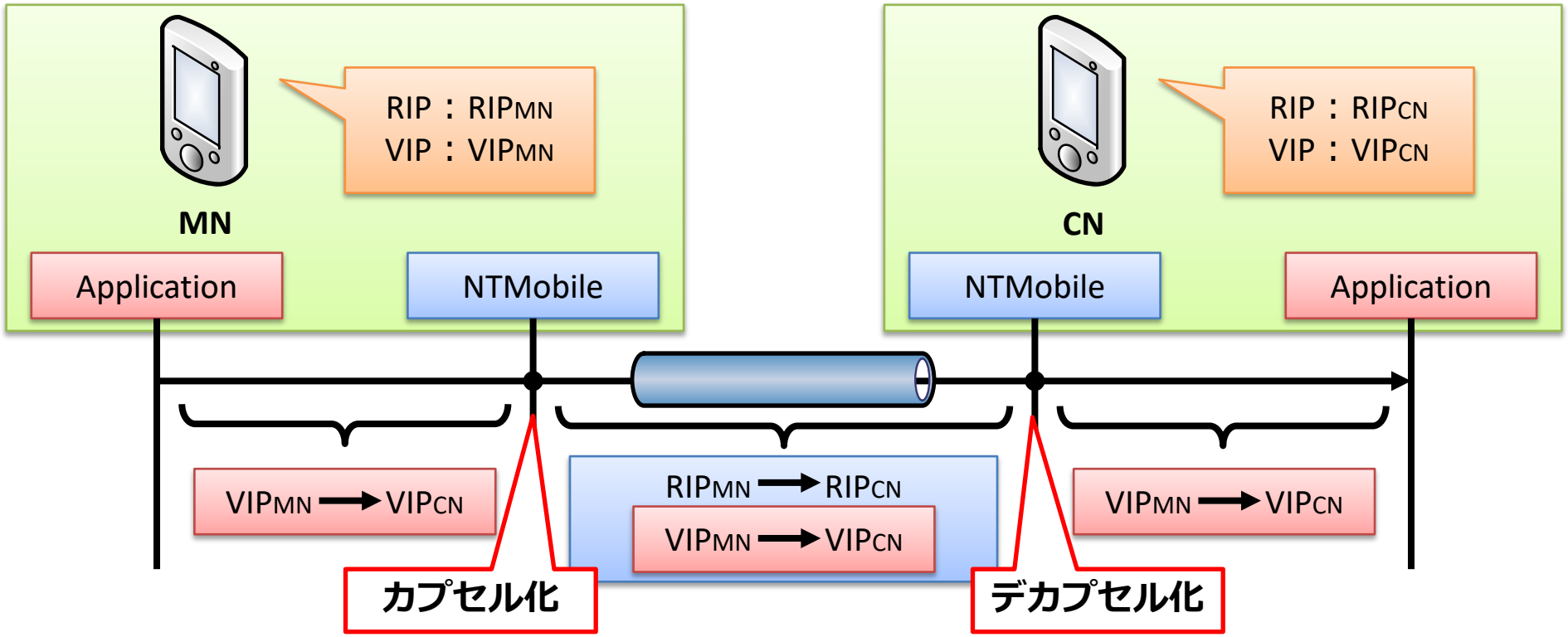
仮想IPアドレス

・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の通信

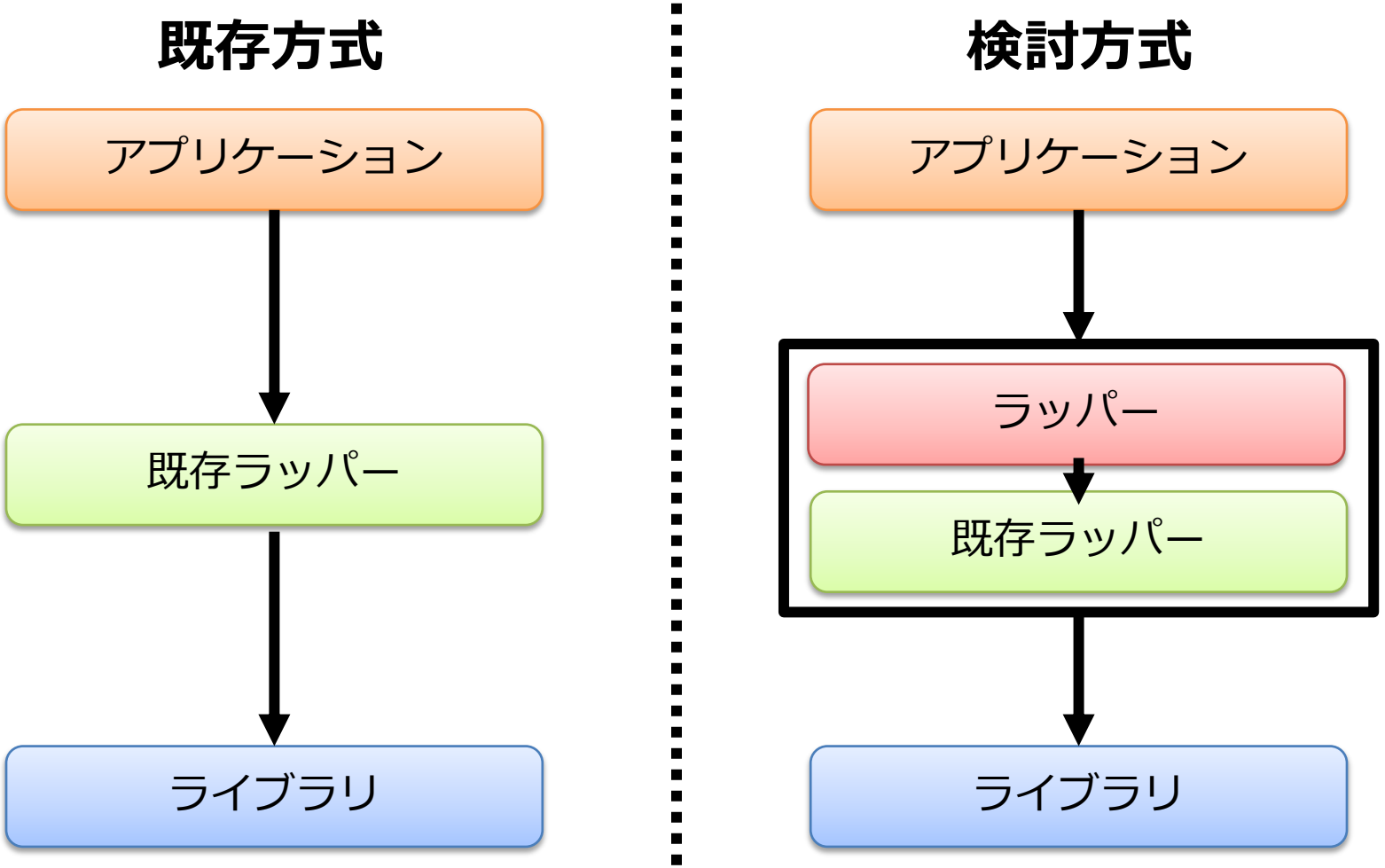
- パケットを実IPアドレスでカプセル化した通信
  - **仮想IPアドレス**
    - 通信端末の位置に依存しない
    - 実IPアドレスの変化を隠蔽する



RIP：実IPアドレス　VIP：仮想IPアドレス

検討方式

- 2段階でラップ

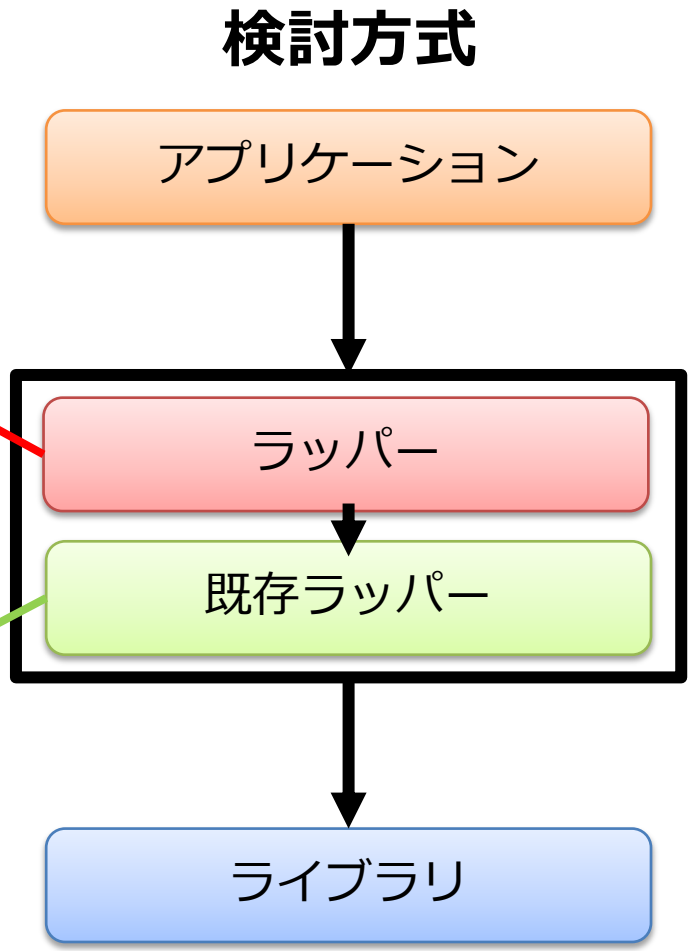


検討方式

2段階でラップ

- アプリケーションの開発言語と同じ使用方法のAPIを実装
- 関数の内部でライブラリの関数を呼び出す

- アプリケーションがライブラリを使用できるようにする

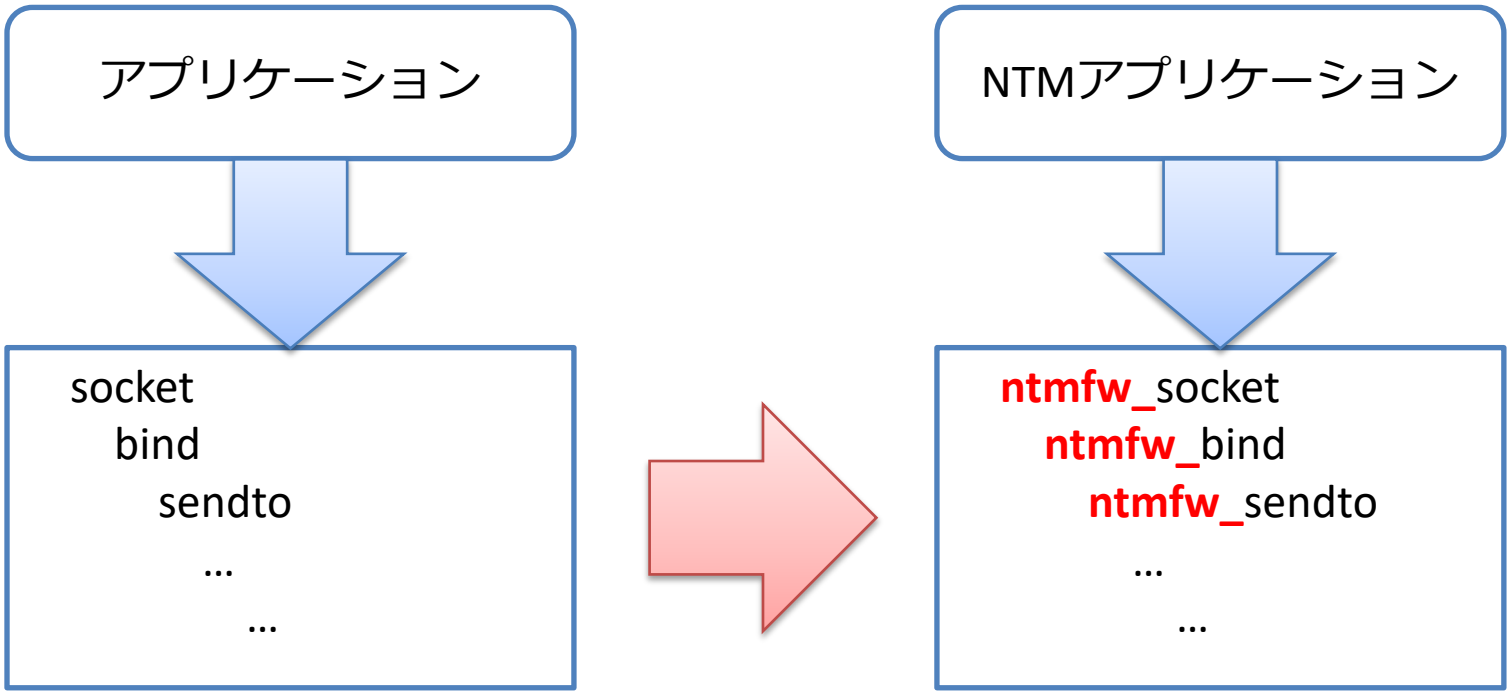


実装

- Node.jsにて仮実装
- 対象はNTMobile framework library
- NTMobile framework library
  - NTMobileの機能をアプリケーション用にライブラリ化したもの
  - ネットワーク上の制約を受けずに通信を可能とする通信ライブラリ
  - C言語によって実装

NTMobileフレームワーク

- NTMobileの処理を全てアプリケーションで実現
- アプリケーションが利用するソケットAPIを置換
  - BSDソケットAPIの代替ソケットAPI(**NTMソケットAPI**)を提供
  - アプリケーション開発者はNTMソケットAPIを利用する

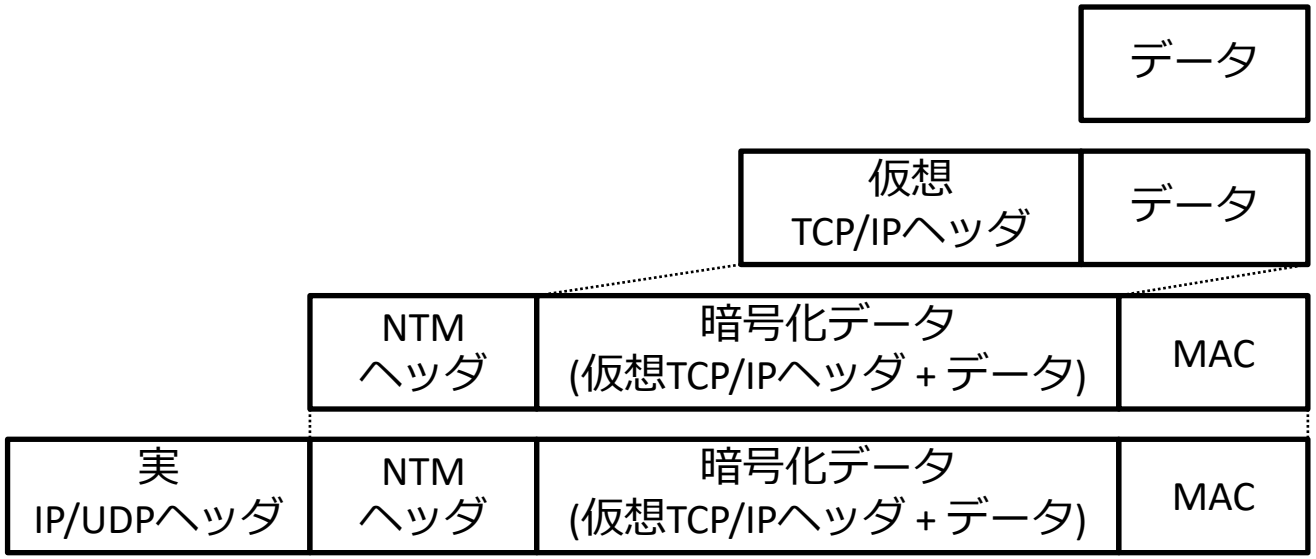


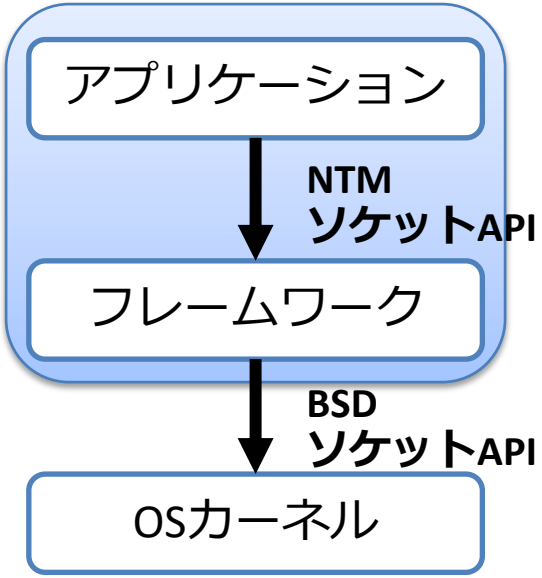
NTMobileフレームワークの動作

Watanabe Lab.

アプリケーション → NTMobile フレームワーク → OSカーネル

- NTMソケットAPIでデータを送信
- 仮想IPアドレスを用いてパケットを生成
- パケットの暗号化やMACを付与
- BSDソケットAPIで送信
- 実IPアドレスを用いてパケットを生成し送信





MAC：Message Authentication Code

NTMobileフレームワークの動作

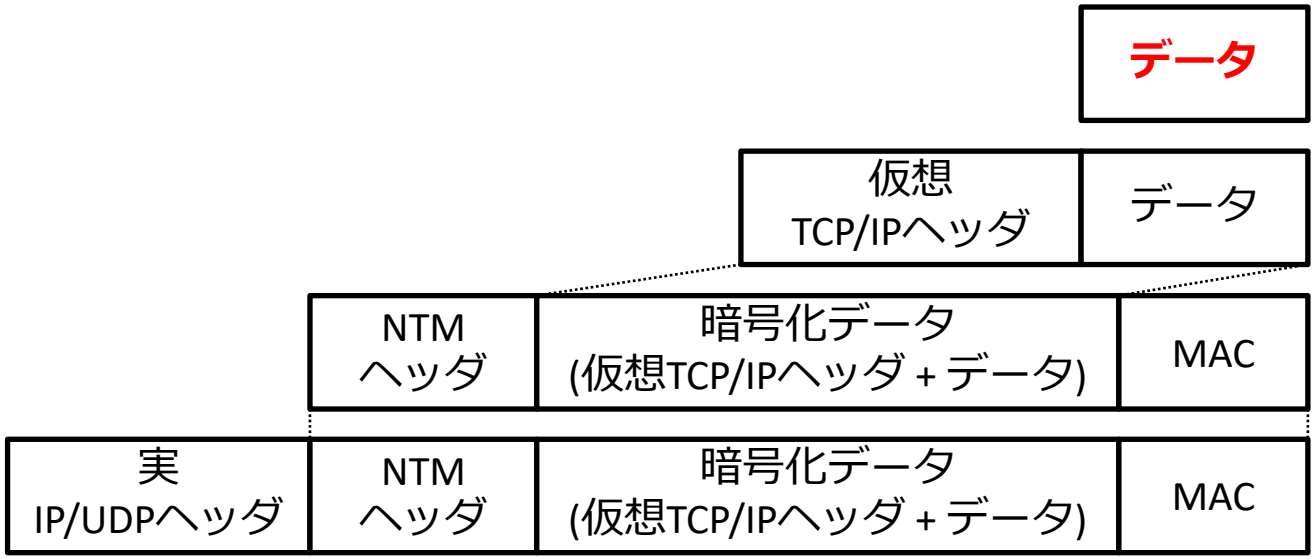
Watanabe Lab.

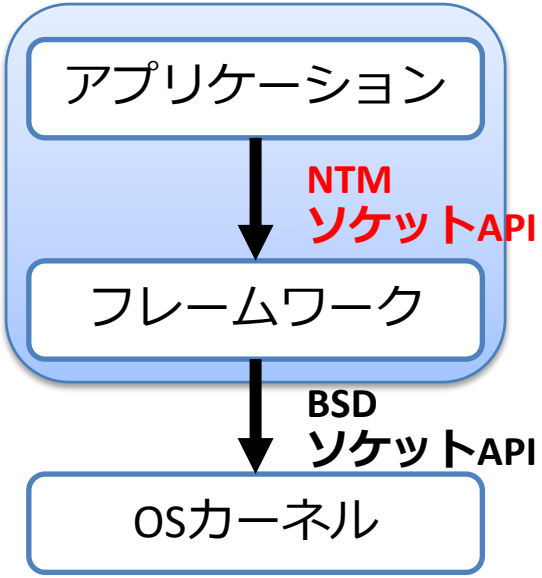
アプリケーション → NTMobile フレームワーク → OSカーネル

- **NTMソケットAPIでデータを送信**

- 仮想IPアドレスを用いてパケットを生成
- パケットの暗号化やMACを付与
- BSDソケットAPIで送信

- 実IPアドレスを用いてパケットを生成し送信





MAC：Message Authentication Code

NTMobileフレームワークの動作

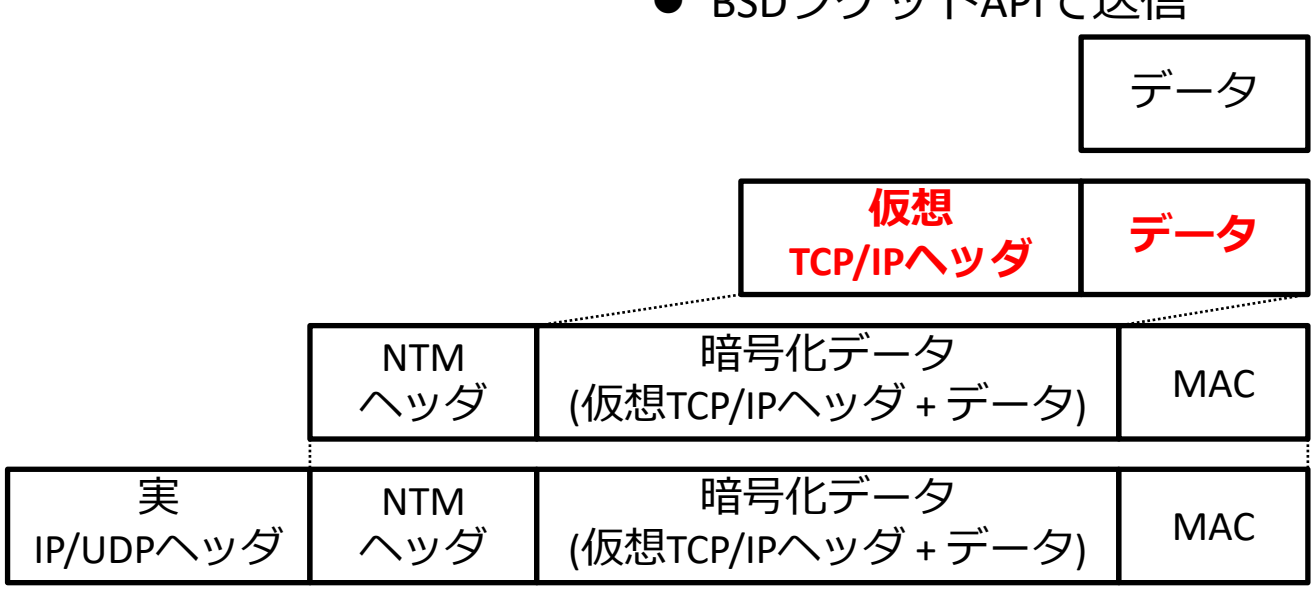
Watanabe Lab.

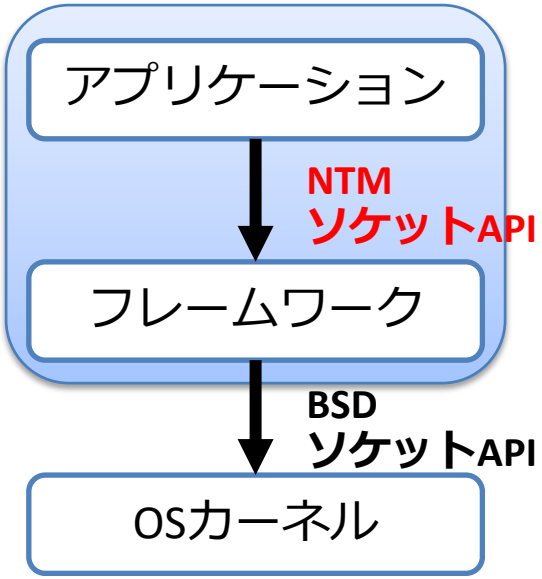
アプリケーション → NTMobile フレームワーク → OSカーネル

- NTMソケットAPIでデータを送信

- **仮想IPアドレスを用いてパケットを生成**
- パケットの暗号化やMACを付与
- BSDソケットAPIで送信

- 実IPアドレスを用いてパケットを生成し送信



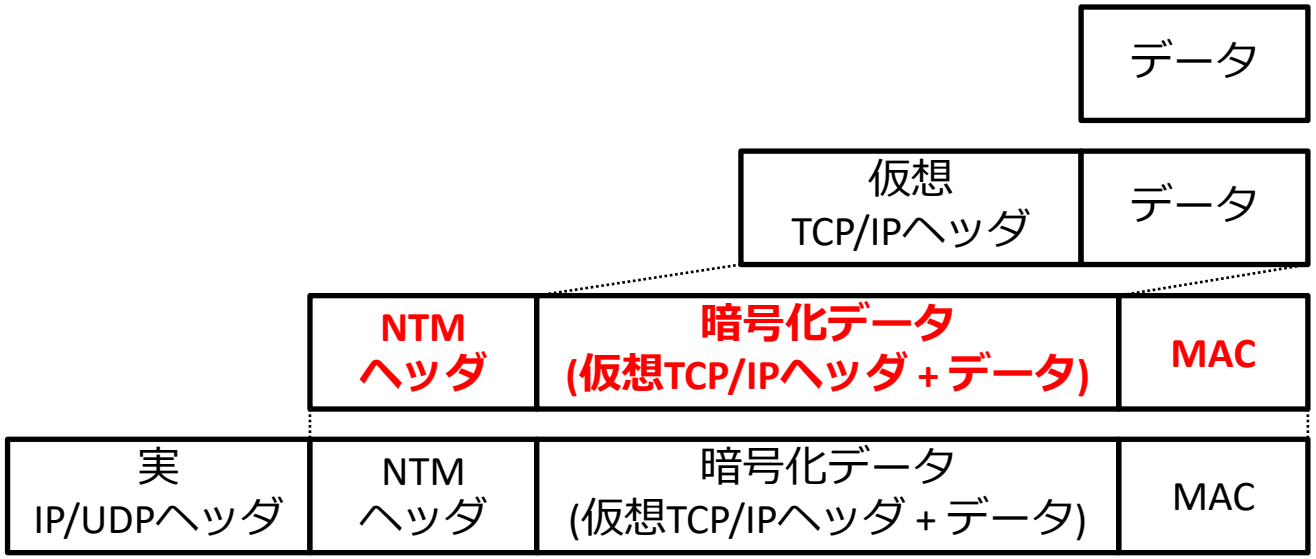


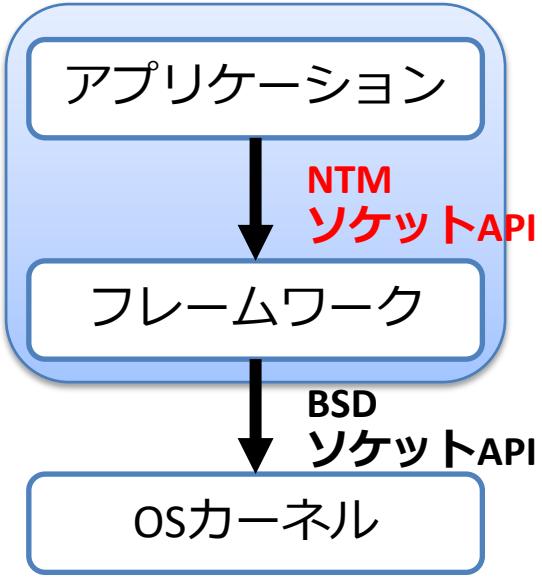
MAC：Message Authentication Code

NTMobileフレームワークの動作

アプリケーション → NTMobile フレームワーク → OSカーネル

- NTMソケットAPIでデータを送信
- 仮想IPアドレスを用いてパケットを生成
- **パケットの暗号化やMACを付与**
- **BSDソケットAPIで送信**
- 実IPアドレスを用いてパケットを生成し送信



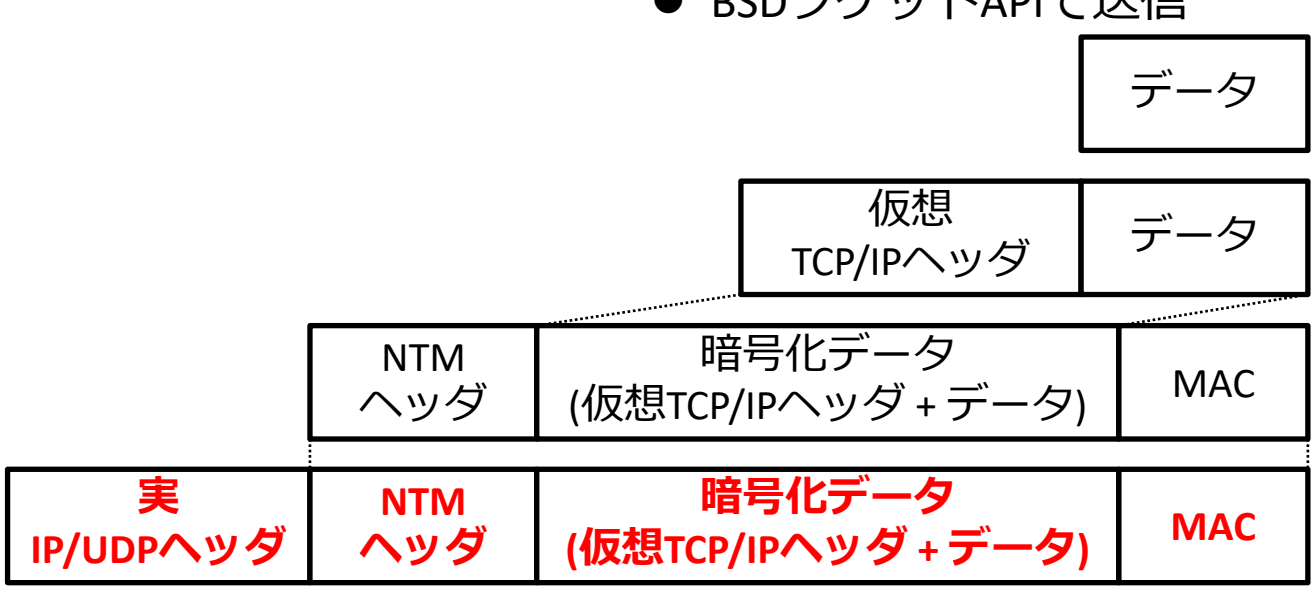


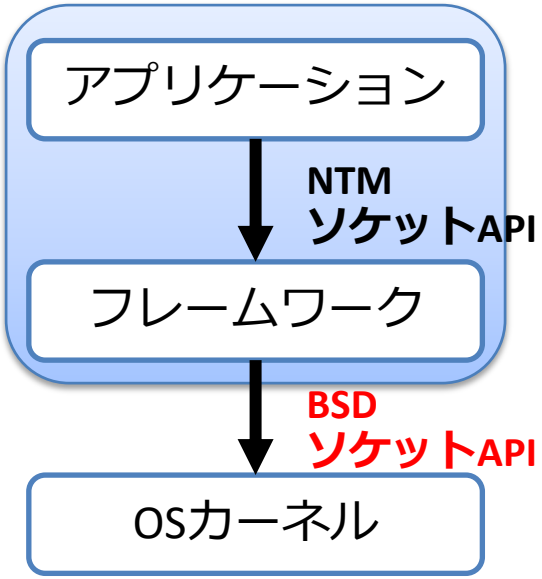
MAC：Message Authentication Code

NTMobileフレームワークの動作

アプリケーション → NTMobile フレームワーク → OSカーネル

- NTMソケットAPIでデータを送信
- 仮想IPアドレスを用いてパケットを生成
- パケットの暗号化やMACを付与
- BSDソケットAPIで送信
- **実IPアドレスを用いてパケットを生成し送信**

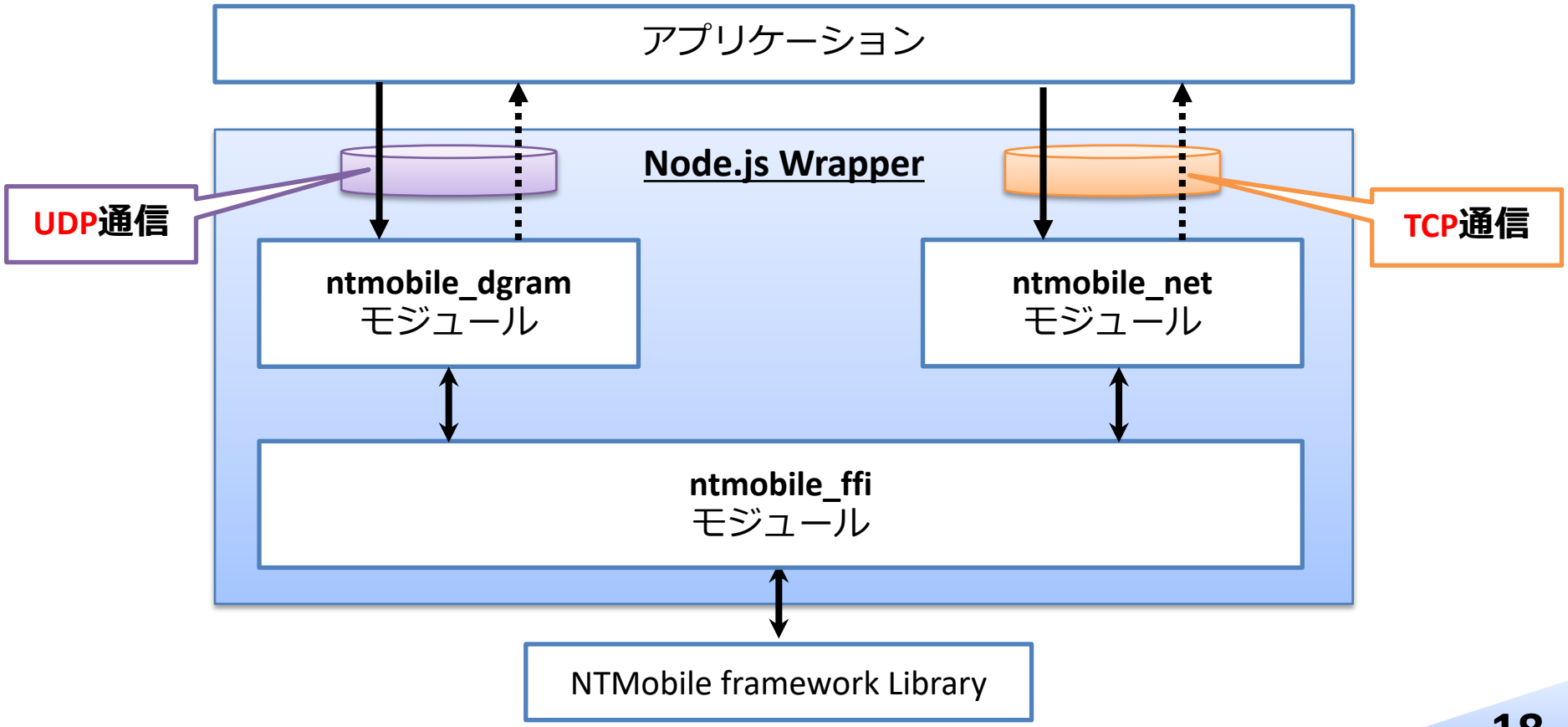




MAC：Message Authentication Code

Node.jsラッパーの構成

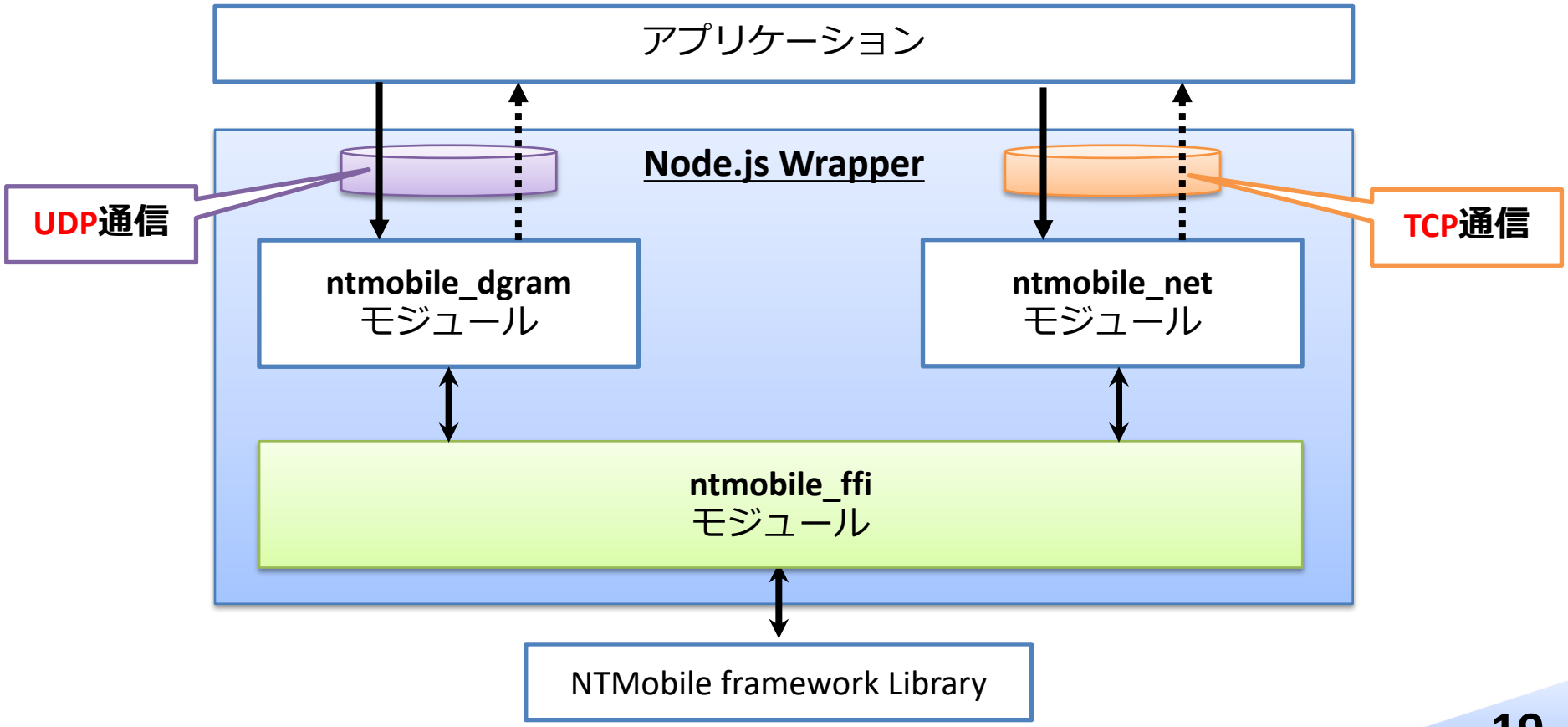
- フレームワークのNTMソケットAPIでパケットを送受信
  - ラッパーは型変換等の結果をNTMソケットAPIに渡す



Node.jsラッパーの構成

ntmobile_ffi モジュール

- C言語とNode.js間での変数の型の違いを除去
- NTMobile framework Libraryが提供するAPIをNode.jsで使用できるように定義

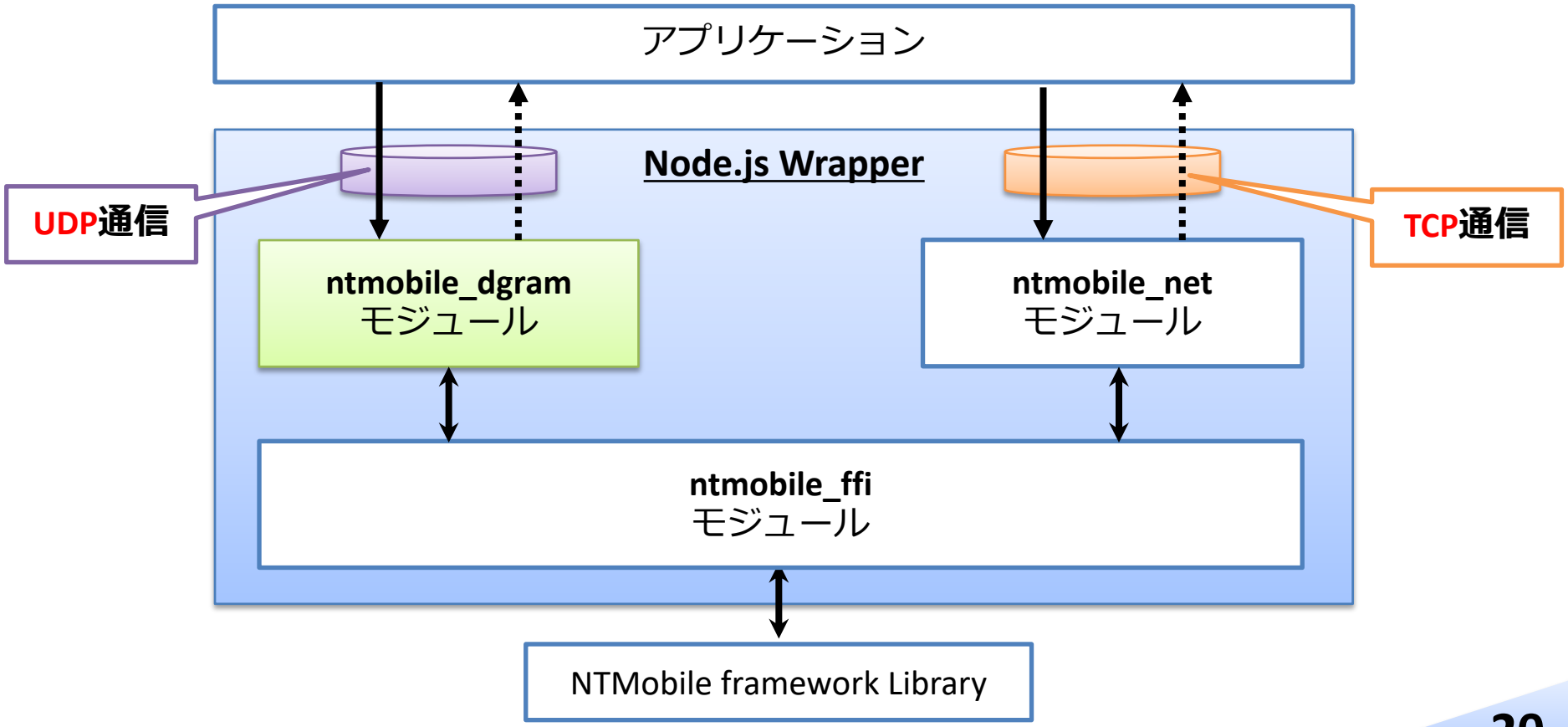


Node.jsラッパーの構成

Watanabe Lab.

ntmobile_dgram モジュール

- Node.jsの標準UDP通信モジュールを継承したNTMobile用のUDP通信モジュール
- 標準UDP通信モジュールと同じ使用方法にてNTMobileによるUDP通信を実現

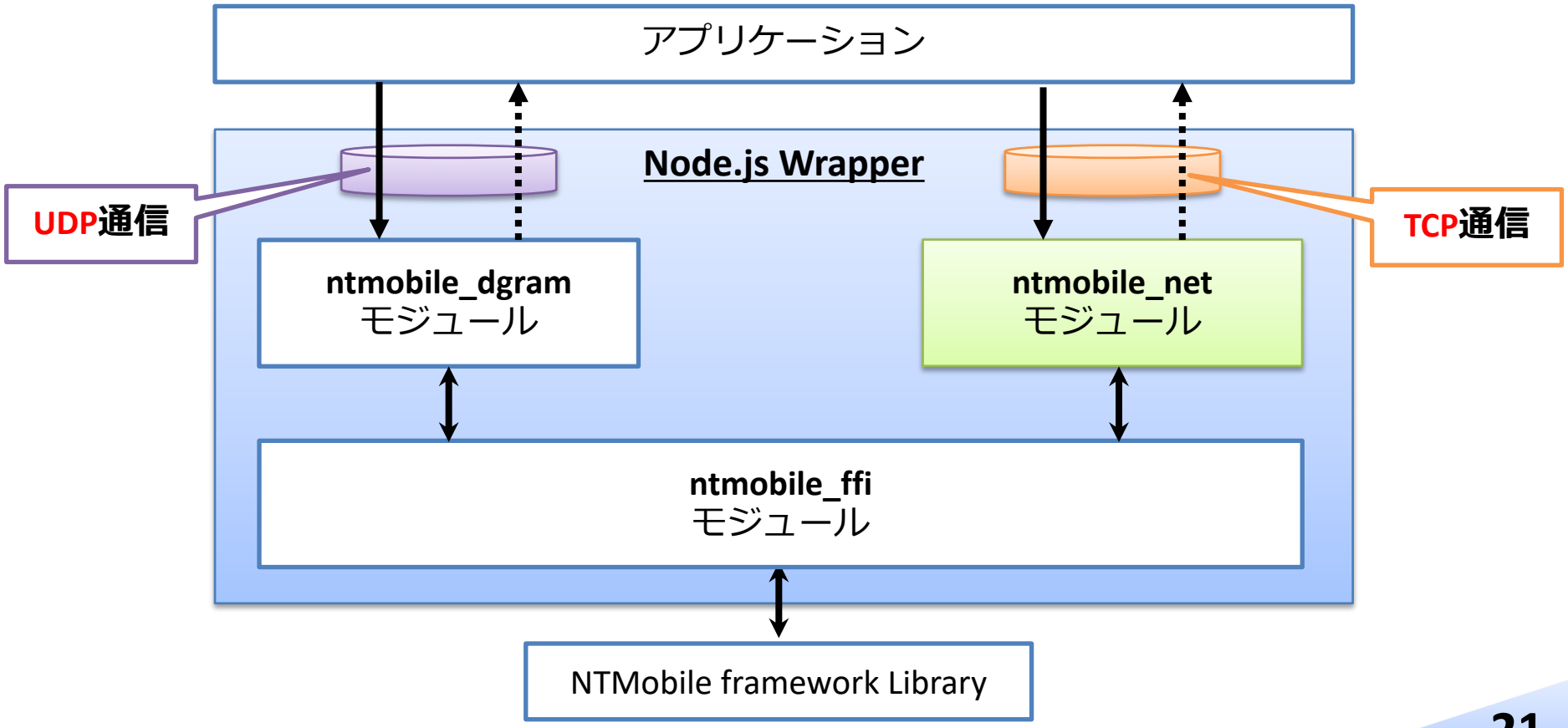


Node.jsラッパーの構成

Watanabe Lab.

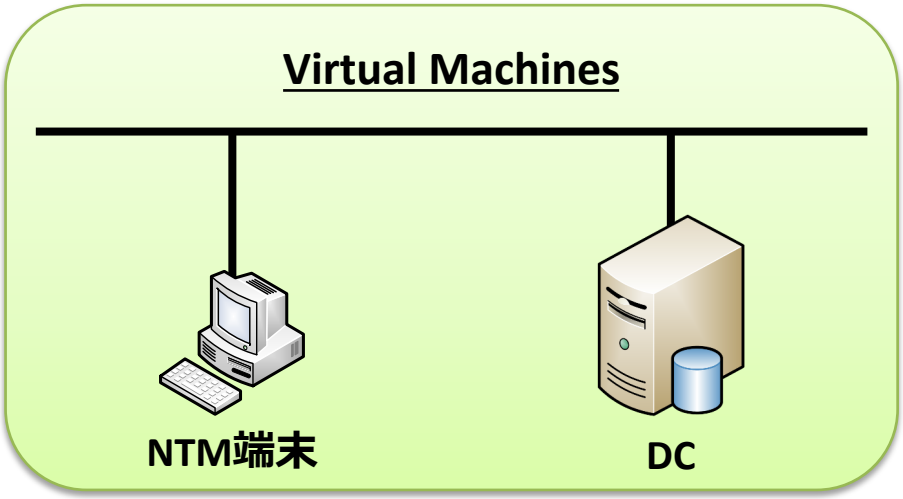
ntmobile_net モジュール

- Node.jsの標準TCP通信モジュールを継承したNTMobile用のTCP通信モジュール
- 標準TCP通信モジュールと同じ使用方法にてNTMobileによるTCP通信を実現



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	NTM端末	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	1Core	1Core
Memory割り当て	2.00GB	1.00GB

動作検証

- Node.jsからNTMobile framework Libraryの初期化を実行

- 初期化実行時に行われる内容
  - NTM端末からDCへ端末情報の登録
  - NTM端末がDCから仮想IPアドレスを取得

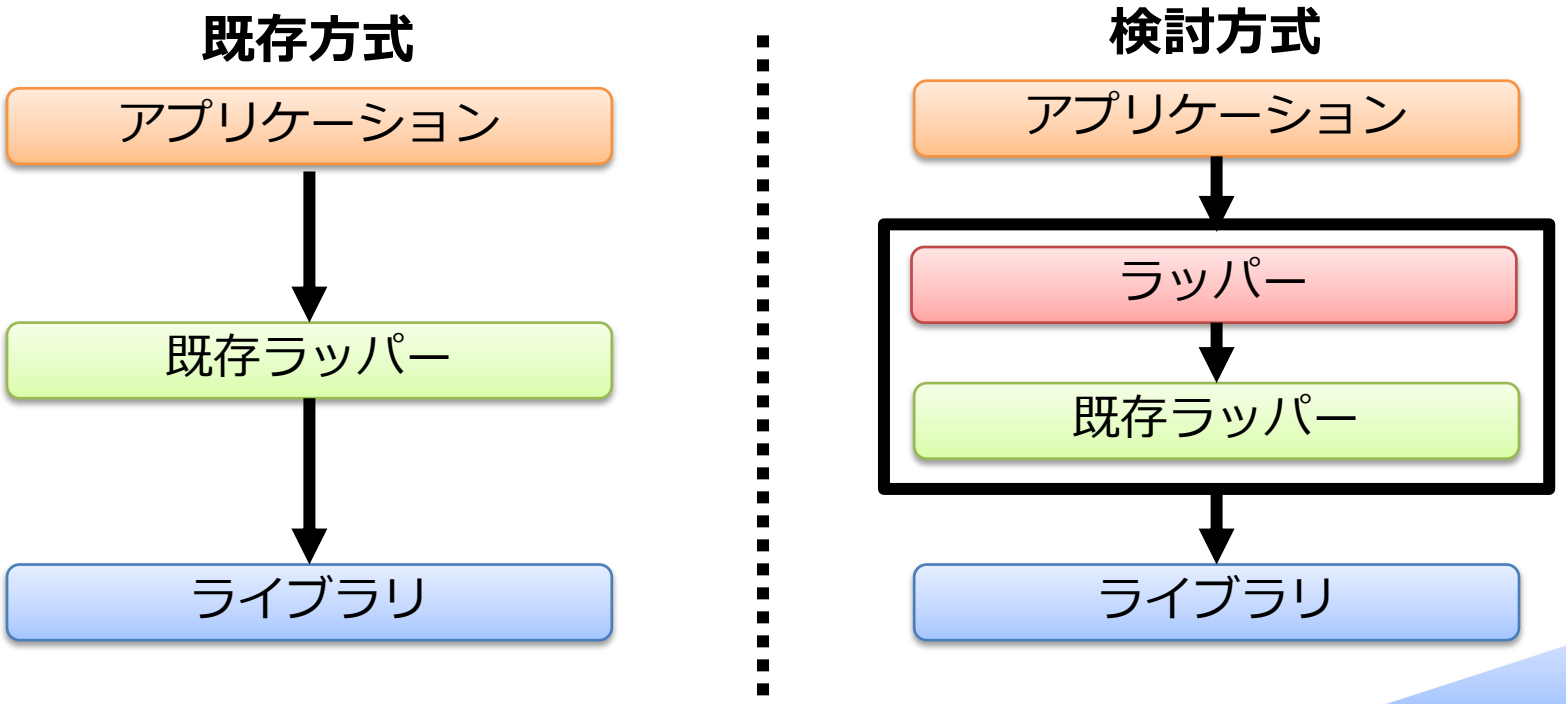
初期化処理

動作確認済

参考

- Javaラッパーを用いたUDP通信時の処理時間
  - 100回の平均

使用した方式	送信API[ms]	受信API[ms]
既存方式	1.222	1.330
検討方式	1.277	1.363
差 (既存ー連携方式)	**0.055**	**0.033**



まとめ

- 機能拡張した通信ライブラリ用のラッパー
  - ライブラリが提供する通信APIの使用方法ではなく，
    呼び出し元の標準通信APIと同じ使用方法であるほうが望ましい
- Node.jsで仮実装
  - 動作確認
- 今後
  - UDP通信モジュールの作成
  - TCP通信モジュールの作成