

NTMobile機能を持つアダプタの実現方式の検討

尾久 史弥¹ 納堂 博史¹ 鈴木 秀和¹ 内藤 克浩² 渡邊 晃¹

概要：現状のネットワークには様々な制約があるため、これらの制約を排除したエンドツーエンド通信の実現が求められている。エンドツーエンド通信を実現できる有用な技術に NTMobile がある。NTMobile framework と呼ばれる通信ライブラリを利用するとアプリケーションレベルでエンドツーエンド通信を実現できる。しかし、組み込み型家電やサーバーのように NTMobile の実装が困難な通信装置が存在する。そこで、本論文では、これらの装置に対して隣接設置することにより NTMobile の機能を付与するアダプタを提案する。本アダプタを Linux 上に実装し、一般の通信装置に手を加えることなく NTMobile の機能を付与できることを確認した。

Researches on the Realization Method of an Adapter having Functions

FUMIYA OGYU¹ HIROSHI NODO¹ HIDEKAZU SUZUKI¹ KATSUHIRO NAITO²
AKIRA WATANABE¹

1. はじめに

高速無線技術の発展やスマートフォンをはじめとする携帯端末の普及により、ユーザがインターネットを利用する機会が飛躍的に増大している。インターネットをはじめとする現在のネットワーク基盤は、ほぼ IP ネットワークで実現されていることから、今後の通信インフラは、IP ネットワークを前提に発展していくものと考えられる。

しかし、IP ネットワークには以下に示すようないくつかの課題がある。IPv4 ネットワークでは、IPv4 グローバル空間側から IPv4 プライベートアドレス空間に対して通信の開始ができない問題がある (NAT 越え問題)。この課題は、IPv4 が存在する限り続き、アプリケーションの実現方法が制約を受け続ける。また、今後 IPv6 アドレスが徐々に使われていくと考えられるが、IPv4 と互換性がなく直接通信を行うことができない。今後 IPv6 アドレスしか取得できない端末が出てくると、IPv4/IPv6 間の相互通信の実現は必須の課題となる。さらに、通信中にネットワークを切り替えると IP アドレスが変化して、通信を継続できない場合がある。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おり、これを可能とする移動透過性技術が必須になると考えられる。

これらの IP ネットワークの課題に対応するため、現在の通信システムはほとんどがクライアント/サーバモデル (以下 CS モデル) で実現されている。CS モデルはサーバをグローバルアドレス空間に置くのが一般的である。クライアントがどのようなアドレス空間に存在しても、サーバとの接続性が保証できるため、サーバにすべての処理を実行させることにより多くのユーザの要求を満たすことが可能である。ただし、通信の開始は必ずクライアント側でなければならない。CS モデルは、現在のネットワークの課題を容認しつつ、現状のアプリケーションの要求を満たすことができる最適の方法といえる。しかし、CS モデルはサーバが処理ネックになる可能性がある。また、CS モデルはサーバのセキュリティ対策や二重化対策など、管理負荷が大きいという課題がある。

エンドツーエンド通信が可能であれば、CS モデルを前提にする必要がない。ここで言うエンドツーエンド通信とは、エンド端末がどのようなアドレス空間に接続してようと、常に最適な通信経路での相互通信が可能で、かつ通信中にネットワークを切り替えても通信が継続できる通信のことである。アプリケーションから見るとネットワーク

¹ 名城大学
Meijo University

² 愛知工業大学
Aichi Institute of Technology

全体があたかも巨大な LAN のように見え、ネットワークの制約を一切意識する必要がない。

ここで、エンドツーエンド通信を実現するための技術として、DSMIPv6(Dual Stack Mobile IP version 6)[1], HIP(Host Identity Protocol)[2], NTMobile(Network Traversal with Mobility)[3], [4], [5] がある。これらの技術は、NAT 越え問題の解決、IPv4/IPv6 間相互通信、移動透過性を同時に可能とするものである。

DSMIPv6 は IPv6 対応の移動透過性技術 MobileIPv6[6] をベースに、IPv4 が混在する環境に拡張した方式である。しかし DSMIPv6 は、IPv4 ネットワークにおいて MobileIPv4[7] の課題をそのまま引き継いでいる。例えば、全ての移動端末に IPv4 グローバルアドレスが必要となり、アドレス枯渇問題に逆行するという課題がある。

HIP は IP アドレスから通信識別子としての役割を分離し、HI(Host Identity) と呼ぶ新たな通信識別子を導入することによって、通信接続性と移動透過性を確保することができる。しかし、NAT 越え技術として ICE[8] を利用しているため、NAT を跨る移動が極めて複雑で、シグナリングに要する時間が大きくなるという課題がある。また、HIP は TCP 層と IP 層の間に HIP 層を定義することにより実現されているため、カーネルの改造が避けられない。そのためスマートフォンなどへの適用が困難であり、アプリケーションによる通信ライブラリの実現には適さない。

NTMobile はシステム内で一意となる仮想アドレスを各エンド端末に割り当て、全ての通信パケットを実アドレスでカプセル化する方式である。NTMobile は、DSMIP や HIP で述べたような課題は存在しない。NTMobile の通信パケットは、仮想アドレスによる通信と、実アドレスによるカプセル化機能が明確に分離されており、アプリケーションレベルで実現できる。NTMobile をアプリケーションレベルで実現した通信ライブラリは、NTMobile framework(以下 NTMfw)と呼ばれている。

ここで、通信装置に NTMobile の機能を付与するためには、NTMfw を組み込む必要がある。NTMfw は NTMobile 通信用のソケット API を提供しており、アプリケーションは、このソケット API を利用することで NTMobile 通信を利用することができる。

新規に開発するアプリケーションは、本ライブラリを使用してエンドツーエンド通信を実現できる。しかし、組み込み型の家電や安定性を重視するサーバなどは、NTMfw を新たに組み込むことができない場合がある。組み込み型の家電は、工場出荷後にプログラムの書き換えができないことが考えられる。また、サーバは安定性を重視して稼働しているため、プログラムの書き換えが許可されないことが考えられる。仮にサーバのプログラムの書き換えができたとしてもサーバで提供している全てのアプリケーションを書き換える必要がある。

そこで本論文は、これらの通信装置に隣接設置することにより、一般通信を NTMobile 通信に変換するアダプタ(以下 NTMA)を提案する。アダプタの実装には、NTMfw の機能を流用した。一般端末と NTMA との間の通信は、RAW ソケットで実現し、NTMA と NTM 端末との間の通信は、NTMfw を改造した API を介して通信を行う。Linux 上で提案方式の実装を行い、動作確認並びに性能評価を行ったので報告する。

以降、2 章ではエンドツーエンド通信の利点を述べる。3 章で NTMobile について述べ、4 章で NTMobile framework について述べる。5 章で提案方式について述べる。6 章で提案方式の実装について述べ、7 章で評価を行う。最後に 8 章でまとめる。

2. エンドツーエンド通信の利点

現在主流となっている CS モデルの構成は図 1 のとおりである。CS モデルはインターネット上にサーバを設置することが前提となる。クライアントは携帯電話網を含め、ほとんどがプライベートアドレス空間に存在する。CS モデルはクライアント側から通信を開始することが前提であるため、クライアントとサーバは必ず通信を確立でき、NAT 越え問題は発生しない。IPv4/IPv6 間の相互通信も、サーバをデュアルスタックネットワークに設置して両者を仲介することにより、理論的に可能である。移動透過性の処理においても、サーバ側のアドレスが固定であることから対応しやすいという利点がある。このように CS モデルは、IP ネットワークの課題をある程度解決することができる。

しかし、CS モデルには以下のような課題がある。CS モデルはすべての情報をサーバに集中させるため、無駄なトラフィックが増え、サーバが処理ネックとなりやすい。アプリケーションによっては好ましくない遅延が発生する。サーバからの情報漏洩に対して万全の対策が必要である。サーバがダウンすることは許されないため、二重化対策が必須となる。物理的にはサーバ設置場所に空調設備が必須である。このように CS モデルは通信性能に係わる課題だけでなく、サーバを運用するためのユーザ側の負担が大きいという課題がある。

E2E モデルの構成を図 2 に示す。E2E モデルはネットワークに接続するすべての装置が自由に通信できることを前提にしたモデルである。ネットワークはあたかも巨大な LAN とみなすことができ、任意の装置間で最適経路のエンドツーエンド通信ができ、かつ通信中に任意の場所に移動しても通信が継続できる。インターネットは、もともとエンドツーエンド通信ができるネットワークを目指していたが、NAT の普及によりそれが実現できなくなっている。IPv6 が普及すればエンドツーエンド通信が可能になるといわれているが、IPv4 の普及状況をみると、NAT の

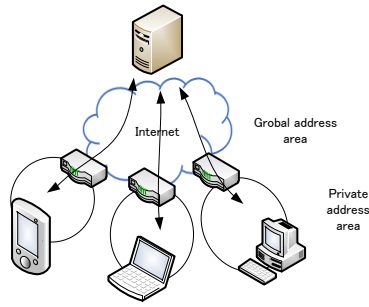


図 1 CS モデルの構成

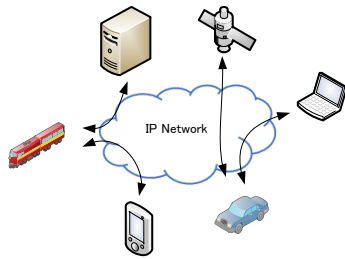


図 2 E2E モデルの構成

存在はもはや無視できない。

E2E モデルを仮にカーネルレベルで実装ができたとしても、機種が限定されることや、カーネルの頻繁なバージョンアップに追従しなければならないという課題がある。また、スマートフォンではインストールのために管理者権限が必要になるため、メーカーの保証が得られず一般ユーザが利用できなくなる。そこで、E2E モデルを普及させるためには、通信ライブラリをアプリケーションレベルで提供することが重要である。

3. NTMobile

本章では、エンドツーエンド通信を実現する技術である NTMobile について説明する。

3.1 NTMobile の構成と原理

図 3 に NTMobile の構成を示す。NTMobile は、NTMfw と呼ばれる通信ライブラリを組み込んだ NTM 端末、実 IP アドレスと仮想 IP アドレスの管理、および通信経路を指示する DC (Direction Coordinator)、エンドツーエンドの通信が行えない場合にパケットの中継を行う RS (Relay Server) によって構成される。DC 及び RS は、グローバルネットワーク上に設置する。ネットワークの規模に応じて複数台設置することが可能で、負荷分散が可能である。

NTMobile は、NTM 端末に対して、位置に依存しない仮想 IP アドレスを割り当て、アプリケーション上では仮想 IP アドレスに基づいた通信を行う。DC は DNS サーバの

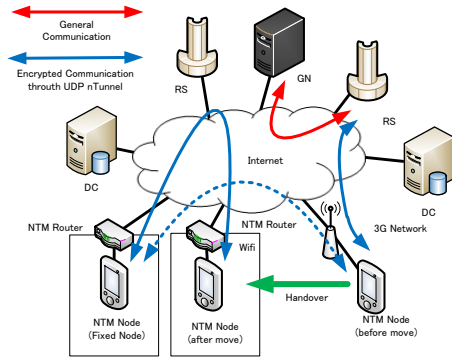


図 3 NTMobile のネットワーク構成

機能を有し、通信相手を発見するとともに NTM 端末に対して最適な通信経路の指示を行う。NTM 端末は、DC に対して定期的に Keep Alive を行っており、DC からの通信経路の指示をいつでも受信できる。DC が NTM 端末に対して適切な経路を指示することにより、NAT 越えを実現することが出来る。

アプリケーションによって生成された仮想 IP アドレスに基づくパケットは、NTMobile の機能により実 IP アドレスでカプセル化し、通信相手に送信する。通信中に端末がネットワークを切り替えると実 IP アドレスが変化するが、仮想 IP アドレスは変化しないので通信を継続できる。

端末同士が直接通信できない場合は、RS 経由の通信を行う。直接通信が行えない場合とは、NTM 端末が一般端末 GN と通信を行う場合、一方が IPv4、もう一方が IPv6 端末の場合、および両 NTM 端末がいずれも Symmetric NAT 配下に存在する場合である。RS を経由する場合であっても、複数の RS の中から 1 つを選択し、冗長の少ない経路を生成できる。

3.2 NTMobile 端末起動時の処理と通信開始時の動作

以降の説明では、通信開始側の NTM 端末を MN (Mobile Node)、通信相手側の NTM 端末を CN (Correspondent Node) として説明する。また、端末 N の FQDN を $FQDN_N$ 、実 IP アドレスを RIP_N 、仮想 IP アドレスを VIP_N と表記する。

端末起動時に MN は自身を管理する DC に対して RIP_{MN} などの端末情報の登録を行う。DC は MN の端末情報をデータベースに登録した後、MN に対して VIP_{MN} を配布する。CN 側も同様の処理を行っておく。この処理を以後、登録処理と呼ぶ。

通信開始時に MN は自身を管理する DC に対して、通信相手の $FQDN_{CN}$ で名前解決およびトンネル構築の指示を依頼する。DC は、DNS サーバの仕組みを利用し、CN の端末情報を取得する。その後、DC は MN および CN の端末情報を基にトンネル経路を判断し、MN と CN に対して

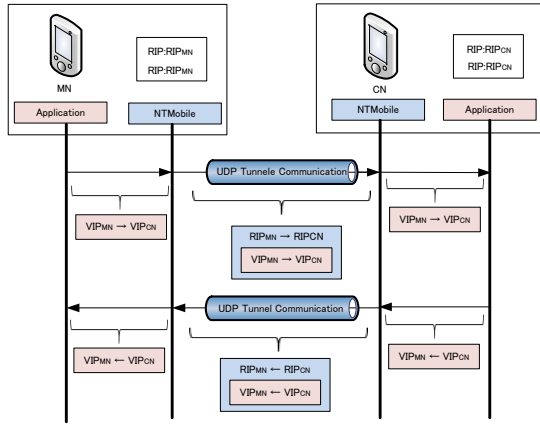


図 4 トンネル通信時の動作

トンネル構築の指示を行う。MN と CN は DC による通信経路の指示に従い、トンネルを構築する。これら一連の処理を以後、シグナリング処理と呼ぶ。

3.3 トンネル通信時の動作

図 4 にトンネル通信時の動作を示す。通信経路上に NAT が存在する場合があるが、動作は同様なのでここでは省略する。MN のアプリケーションは、仮想 IP アドレスを用いてパケット（送信元： VIP_{MN} 、宛先： VIP_{CN} ）を生成する。その後、仮想 IP アドレスに基づくパケットは NTMobile の機能により実 IP アドレス（送信元： RIP_{MN} 、宛先： RIP_{CN} ）でカプセル化して CN へ送信する。MN から送信されたパケットを受け取った CN は、NTMobile の機能によりパケットのデカプセル化を行い、仮想 IP アドレスに基づくパケットを取り出す。その後、CN のアプリケーションに仮想 IP アドレスに基づくパケットを届ける。逆方向の通信も同様の処理が行われる。

4. NTMfw

NTMobile は NTMfw と呼ぶ通信ライブラリにより実現されている。本章では、NTMobile の機能を使用する NTMfw のモジュール構成と課題について説明する。

4.1 NTMfw のモジュール構成

NTMfw は、アプリケーション層で動作する通信ライブラリであり、上位アプリケーションに対して、NTMobile 通信の API を提供する。図 5 に NTMfw のモジュール構成を示す。NTMfw は、仮想 IP スタックに lwIP (A Lightweight TCP/IP stack) を使い、NTM ソケット API は基本的に lwIP に接続される [10]。lwIP の仮想ネットワークインターフェースには仮想 IPv4/v6 アドレスを登録する。このため、アプリケーションが送信するパケットは lwIP により仮想 IP アドレスを用いて TCP/IP または UDP/IP ヘッダが付与され、パケット操作モジュール (PMM) に処理が渡

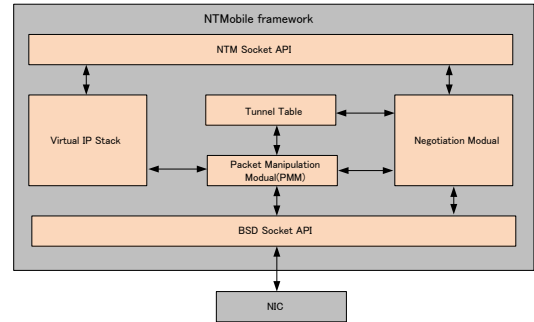


図 5 NTMfw のモジュール構成

される。PMM はこのパケットに対して NTM ヘッダを付与して、暗号化及び MAC 付与を行った後、C 言語標準ソケット API (BSD ソケット API) を用い、通信相手端末の実 IP アドレス宛に送信する。また、PMM は、カプセル化パケットを BSD ソケット API を用いて受信し、当該パケットを MAC 検証及び復号し、lwIP に処理を渡す。アプリケーションは、NTM ソケット API の `recv` 関数等によりこのパケットを受信する。以上の処理により、アプリケーションは仮想 IP アドレスを用いてパケットの送受信を行うことができる。

4.2 NTMfw の課題

NTMobile 通信は、NTMfw をエンド端末に組み込むことで実現する。しかし、NTMfw を実装できない通信装置の場合は、NTMobile の機能を利用することができない。NTMfw を実装できない通信装置の例として、組み込み型家電およびサーバが挙げられる。

組み込み型家電では、プログラムの書き込みに使用するメモリがマスク ROM であった場合、工場での製造後に新たに NTMfw を組み込むことができない。また、一般端末に NTMfw を組み込むためには、既存のアプリケーションを改造する必要があるが、サーバのように安定性を重視する通信装置ではアプリケーションの変更を一切許可されない場合が考えられる。また、サーバには複数のアプリケーションが稼働していることが考えられ、これらのアプリケーションを書き換えないと NTMobile を利用できない。更に Windows のように NTMobile をサポートしていない OS では利用できない。

5. NTMA の提案

本章では、NTMA の動作について述べる。NTMA は、イニシエータ/レスポнда側の両方で利用することが考えられる。今回は、NTMA をイニシエータ側の一般端末 GN (General Node) に設置した場合の動作について確認したので、NTMA をイニシエータ側の GN へ接続した場合の記述をする。ここでの GN は一般ユーザが使う Windows

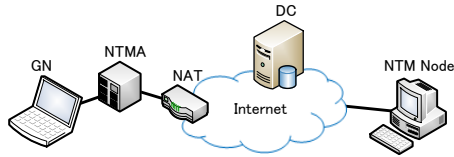


図 6 提案方式のネットワーク構成

端末が想定される。

5.1 ネットワーク構成と前提条件

図 6 に提案方式のネットワーク構成を示す。NTMA は物理 NIC を 2 枚用意して、一方は一般端末 GN とブリッジ接続する。他方はプライベートネットワークとブリッジ接続し、NAT を介してインターネットに接続する。DC と GN の通信相手の NTM 端末は、インターネット上に設置されている。

NTM 端末と NTMA は、起動時に DC に対して登録処理を行う。NTM 端末と NTMA は、DC に対して定期的に Keep Alive を行い、いつでも DC からの通信経路の指示を受けることが出来る。GN の IP アドレスは、NTMA が DC から割り当てられた仮想 IP アドレスを使用する。

5.2 NTMfw の通信シーケンス

図 7 に提案システムの通信シーケンスを示す。通信開始時に、GN は相手 NTM 端末の $FQDN_{CN}$ を指定して DNS クエリを送信する。DNS クエリを受け取った NTMA は、NTMfw の機能を利用して、シグナリング処理により、NTMA と NTM 端末との間でトンネル構築を行う。トンネル構築終了後、NTMA は、相手 NTM 端末の仮想 IP アドレスを取得するので、このアドレスを DNS レスポンスに載せて GN へ送信する。GN は通信相手を CN の仮想アドレスとして認識する。次に GN は、相手 NTM 端末の仮想 IP アドレス宛にパケットを送信する。このパケットを受信した NTMA は、NTMA と相手 NTM 端末の実 IP アドレスでカプセル化して送信する。相手 NTM 端末からパケットが返信されてきた場合は、NTMA がデカプセル化を行い、一般通信のパケットに変換してから GN へ送信する。

6. 実装

本章では、提案方式の実装とその動作検証について述べる。NTMA は、Linux 環境での実装と確認を行った。

6.1 NTMA のモジュール構成

図 8 に NTMA のモジュール構成を示す (イニシエータ側)。NTMA と相手 NTM 端末との間の通信は、R-NTMfw(Remodeled-NTMfw) が提供する NTMobile 通信

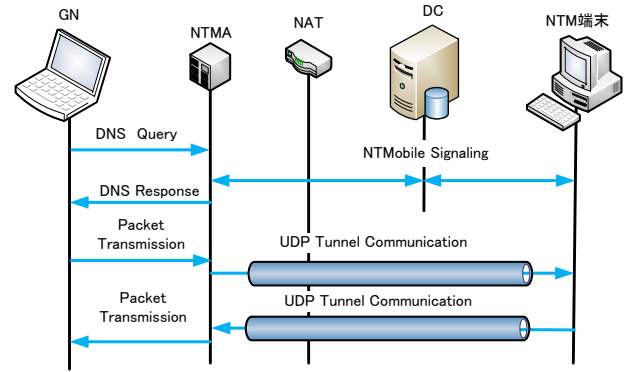


図 7 NTMA の通信シーケンス

で実現する。R-NTMfw は NTMfw 中の lwIP をスキップするよう改造したモジュールである。NTMA と GN との間の通信は、RAW ソケットで実現する。Init モジュールは、GN の起動時に DC に対して GN を NTM 端末として登録処理をする。Init モジュールは DHCP サーバの機能を包含し、GN からの IP アドレス要求に対して DC から取得した仮想 IP アドレスを割り当てる。

イニシエータモジュールは、GN から送信されたパケットを RAW ソケットで受信する。受信したパケットが DNS クエリであった場合、NTMobile シグナリングによりトンネル構築処理を行う。また、トンネル構築終了後に DNS クエリの返信を GN に送信する。GN 側から受信したパケットが UDP/TCP のパケットであった場合は、R-NTMfw の API を介してトンネル通信でパケットを送信する。NTM 端末が NTMA に対して応答パケットを送信した場合は、NTMA は R-NTMfw でパケットを受信し、その後 RAW ソケットを介して GN へ応答パケットを中継する。GN はパケット送信時に、NTM 端末の仮想 IP アドレスをターゲットとする ARP リクエストを送信する事があるので、NTMA は自らの MAC アドレスで ARP リプライを返信する必要がある。GN から受信したパケットが DHCP パケットであった場合、処理を Init モジュールに渡し、仮想 IP アドレスを GN に配布する。

6.2 R-NTMfw

NTMfw は lwIP によりユーザデータから仮想 IP アドレスによるパケットを生成する機能があるが、NTMA は GN が送信する IP パケットを RAW ソケットでそのまま受信する。そのため、NTMfw 内部で仮想 IP パケットを生成する必要がない。また、相手 NTM 端末から受信したパケットは、仮想 IP アドレスのパケットを抽出した後、IP ヘッダを除去せずに、そのまま GN へ中継すればよい。仮想 IP アドレスに基づくパケットは、NTMfw 内の lwIP で処理していたが、提案システムではこの処理を GN で実行する。そのため、R-NTMfw では lwIP の処理をスキップするよ

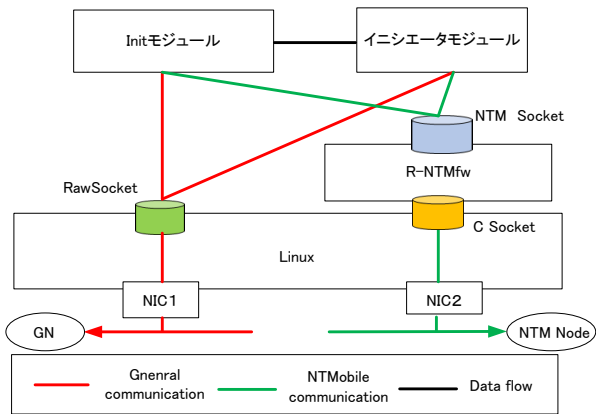


図 8 NTMA のモジュール構成

うに改造した。また、NTMfw は、カーネルが認識する全ての NIC に対して NTMobile の処理を行うように実装されているため、R-NTMfw では NTMobile 通信を行わない NIC を認識しないように変更した。

7. 評価

本章では、NTMA の動作検証と性能評価について述べる。

7.1 動作検証

提案方式の動作検証を行うにあたって、Init モジュールとイニシエータモジュールの中継処理部を C 言語で Linux に実装した。動作検証にあたり、NTMA と NTM 端末間でのトンネル構築処理はあらかじめ終了させた。また、GN の IP アドレスとして NTMA が DC から割り当てられた仮想 IP アドレスを静的に設定した。この状態で GN から NTM 端末の仮想 IP アドレス宛にパケットを送信した。NTMA は、このパケットを NTMobile 通信に変換して送信し、NTM 端末で受信することを確認した。また、NTM 端末が送信した応答パケットを NTMA が受信し、一般通信に変換して GN へ送信することを確認した。

7.2 評価構成と評価手順

図 9, 10 に評価構成のイメージと実際の評価構成を示す。表 1 に仮想マシンの仕様を示す。VMware Workstation Player を使用して、Linux による NTMA, DC, AS, NTM 端末を仮想マシンとして構築した。NTMA には、仮想 NIC を 2 枚用意してを GN にブリッジ接続し、VNIC1 はインターネット側とブリッジ接続した。NTMA 以外の仮想マシンの VNIC は、全てインターネット側にブリッジ接続した。GN は Windows による実機を使用して、ホストマシンの NIC1 にブリッジ接続した。

評価手順は、動作検証時と同様の手順で行った。通信開始時に GN から NTM 端末の仮想 IP アドレス宛に所定の文字列を送信する。この手順を 10 回繰り返して、NTMA

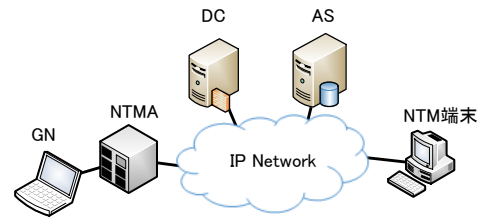


図 9 評価構成のイメージ

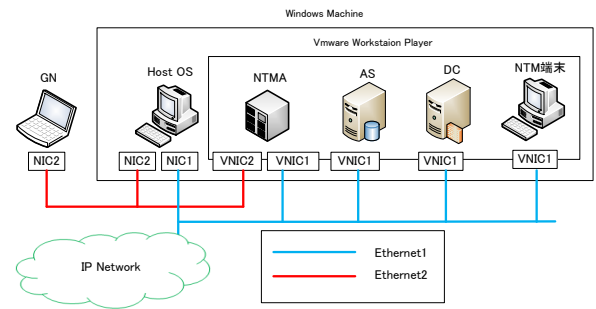


図 10 実際の評価構成

表 1 仮想マシンの仕様

	NTMA(Virtual Machine)
OS	Ubuntu 14.04
CPU	Intel Corei7-930(2.80GHz)
Memory	2GB

表 2 NTMA の測定結果

区分	時間 (us)
ARP 処理時間	246.1
パケット中継時間	12.0
R-NTMfw 処理時間	206.5
応答パケット中継時間	324.0

の処理に要した平均時間を NTMA の評価結果とした。

7.3 評価結果

図 11 に測定を行った通信シーケンス、表 2 に測定結果を示す。ARP 処理時間は、GN が送信する ARP リクエストを NTMA が受信して、そのリプライを送信するまでに要する時間である。パケット中継時間は、NTMA がパケットを受信して相手 NTM 端末へ中継する処理に要する時間である。R-NTMfw 処理時間は、R-NTMfw の API で送信するのに要する時間である。応答パケット中継処理時間は、NTM 端末が応答パケットを送信した場合に、GN へパケットを中継するのに要する時間である。

処理時間を含めても 1m 秒以下で処理を終えており、想定した時間内で処理が完了していることを確認した。

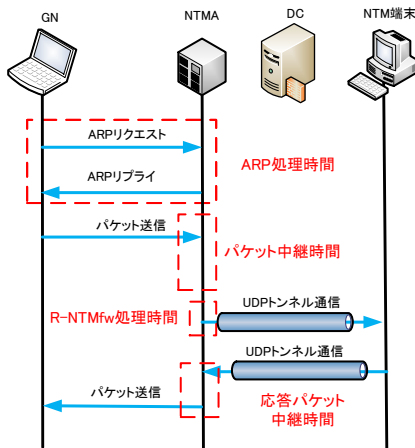


図 11 測定を行った動作シーケンス

8. まとめ

本論文では、NTMobile を実装できない一般端末のために、一般端末に隣接設置して NTMobile 通信を代行する NTMA を提案した。この手法により、一般端末に一切手を加えることがなく、NTMobile の機能を付与することができる。Linux 上での試作により NTMA が想定した動作を行うことを確認した。また、提案方式の性能評価を行い、パケット中継の時間が僅かであることを確認した。

今回の実装はレスポンド側の NTMA にも適用できるため、レスポンド側の検討を引き続き進める予定である。

参考文献

- [1] H.Soliman : Mobile IPv6 Support for Dual Stack Hosts and Routers, RFC5555, IETF(2009).
- [2] R.Moskowitz, T.Heer, P.Jokela, and T.Henderson : Host Identity Protocol version 2 (HIPv2), RFC7401, Updated by RFC8002, IETF(2015).
- [3] 鈴木秀和, 上酔尾一真, 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. 380-393 (2013).
- [4] 内藤克浩, 上酔尾一真, 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).
- [5] 上酔尾一真, 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).
- [6] 上酔尾一真, 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).
- [7] C.Perkins, D.Johnson, and J.Arkko : Mobility Support in IPv6, RFC6275, IETF(2011).
- [8] C.Perkins, Ed. : IP Mobility Support for IPv4, Revised, RFC5944, IETF(2010).
- [9] J.Rosenberg. : Interactive Connectivity Establishment(ICE) : A protocol for Network Address Translator(NAT) Traversal for Offer/Answer Protocols, IETF(2010).

- [10] lwIP - A Lightweight TCP/IP stack - Summary.
<http://savannah.nongnu.org/projects/lwip/>.

NTMobileアダプタの実現方式の検討

尾久 史弥† 納堂 博史† 鈴木 秀和† 内藤 克浩† 渡邊 晃†

†名城大学 理工学研究科

‡愛知工業大学 情報科学部



研究背景

◆ 現在の主流はIPv4ネットワーク

➤ IPv4グローバルアドレスの枯渇問題

- NATを利用したプライベートネットワークの構築が一般的

◆ NAT越え問題

➤ インターネット側端末から通信を開始できない

インターネット側端末

プライベート側端末A

プライベート側端末B

Global Network

Private Network

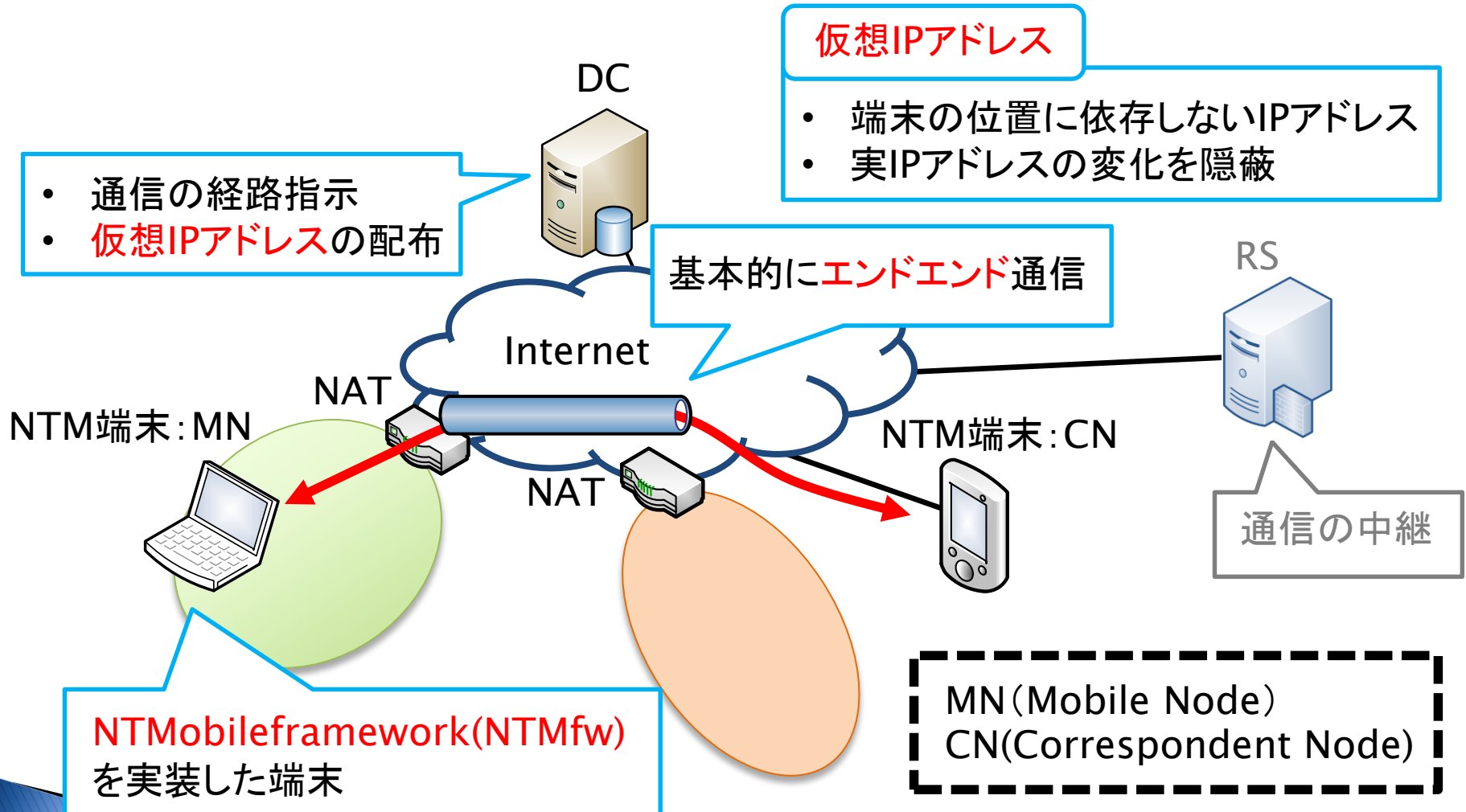
NAT越え問題

NAT越え問題を解決する技術

*NTMobile(Network Traversal with Mobility)が提案されている

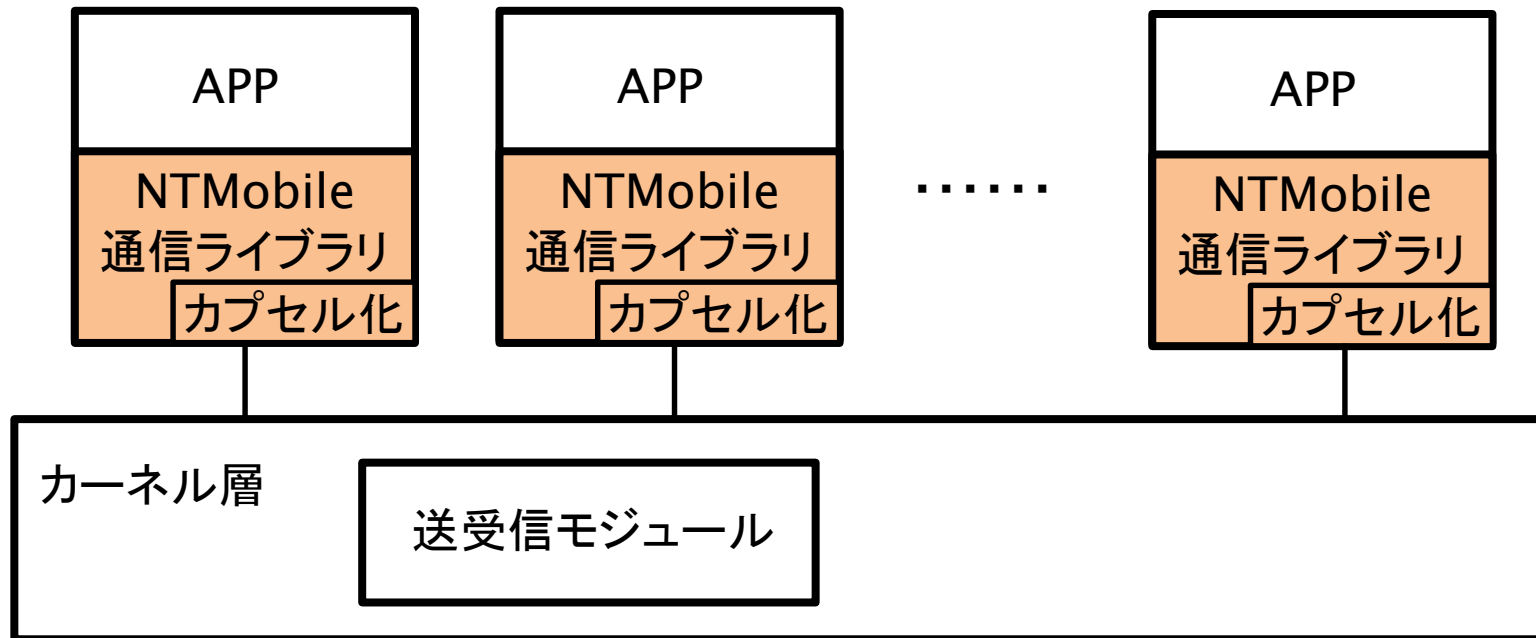
NTMobile

- 移動透過性の実現と**NAT越え問題の解決**を同時に実現する技術
- NTM端末, DC(Direction Coordinator), RS(Relay Server)で構成



NTMfw (1)

- 上位アプリケーションに対してNTMobile機能を提供
 - NTMobile通信用のAPIを提供



	通信ライブラリ	アップデート	ソフトウェア	対応可能OS
NTMfw	アプリケーション層	必要なし	変更必要あり	Linux, Android, iOS

NTMfw (2)

一般通信アプリケーション例

```
#include <stdio.h>
:
int main()
{
    getaddrinfo(...);

    sock = socket(...);

    sendto(...);

    recvfrom(...);

    return 0;
}
```

NTMobile通信アプリケーション例

```
#include <stdio.h>
#include <ntmfw_api.h>
:
int main()
{
    ntmfw_getaddrinfo(...);

    sock = ntmfw_socket(...);

    ntmfw_sendto(...);

    ntmfw_recvfrom(...);

    return 0;
}
```

NTMobileの課題

- NTMobile非対応端末では**利用できない**

✖ WindowsなどのOSで利用不可

	対応可能OS
NTMfw	Linux, Android, iOS

- NTMfw(NTMobile framework)を組み込めない

✖ 組み込み型の家電

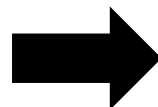
- 工場出荷後はプログラムを**書き換えられない**

✖ アプリケーションサーバー

- 安定性を重視することから、新しい機能の追加が**許可されない**

既存アプリケーション

```
getaddrinfo(...);  
socket(...);  
sendto(...);
```



NTMobileアプリケーション

```
ntmfw_getaddrinfo(...);  
ntmfw_socket(...);  
ntmfw_sendto(...);
```


NTMobileの課題

- NTMobile非対応端末では**利用できない**

✖ WindowsなどのOSで利用不可

	対応可能OS
NTMfw	Linux, Android, iOS

- NTMfw(NTMobile framework)を組み込めない

✖ 組み込み型の家電

- 工場出荷後はプログラムを**書き換えられない**

✖ アプリケーションサーバー

- 安定性を重視することから、新しい機能の追加が**許可されない**

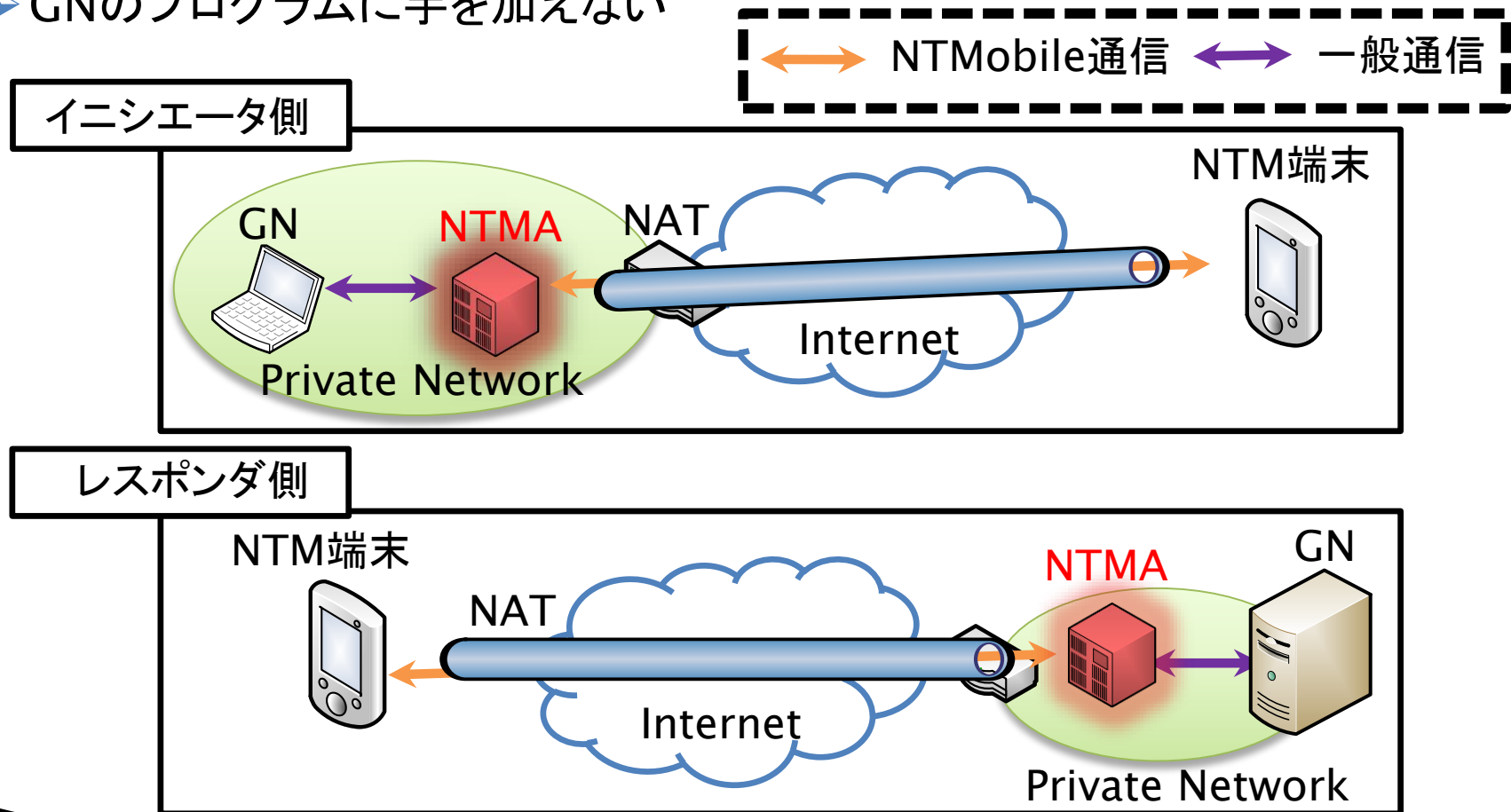
既存アプリケーション

NTMobileアプリケーション

NTMobileアダプタ(NTMA)の提案

研究目的

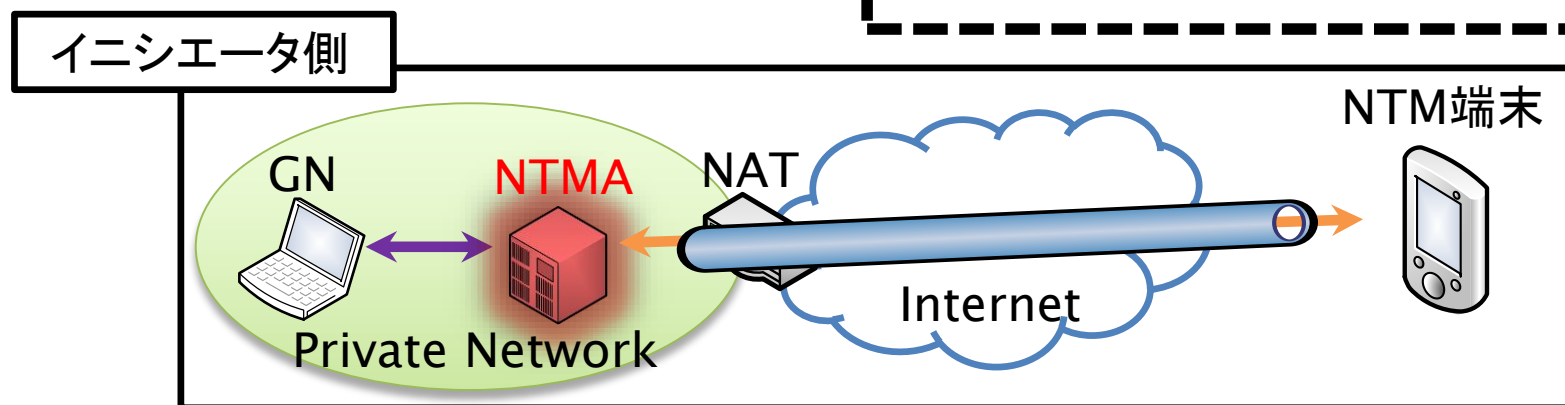
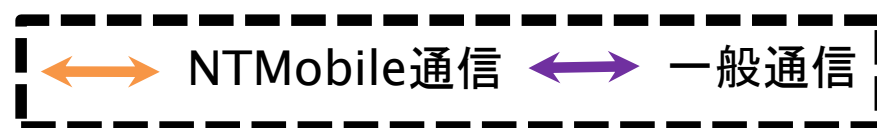
- 一般端末(GN:General Node)に**NTMA**を隣接設置
 - GNの通信をNTMobile通信に変換
 - GNのプログラムに手を加えない



研究目的 (イニシエータNTMA)

■ イニシエータNTMA

- 通信開始側(イニシエータ)のGNに設置
- GNの通信をNTMobile通信に変換



GNの想定環境

- ✓ Windows PC
- ✓ NTMfw未実装端末

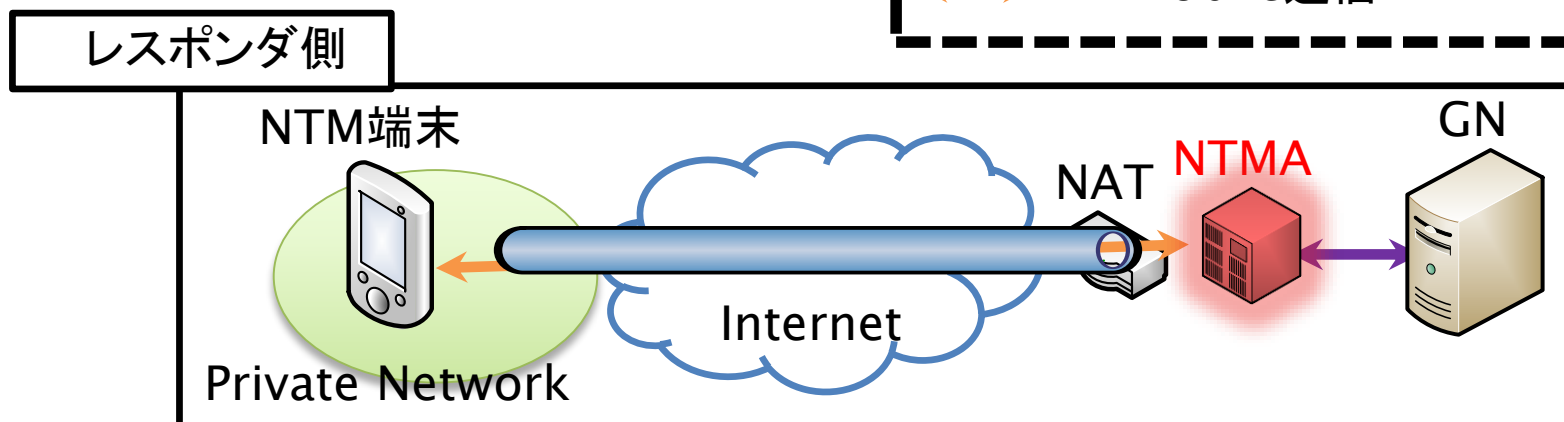
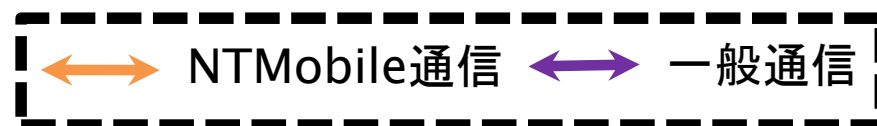
NTMobileの課題

- ✓ NTMobile未対応OS
- ✗ Windowsは未対応
- ✗ NTMfwが未実装の端末

研究目的 (レスポндаNTMA)

■ レスポндаNTMA

- 通信開始される側(レスポнда)のGNに設置
- NTMobile通信を一般通信に変換



GNの想定環境

- ✓ サーバー
- ✓ 組み込み型家電

NTMobileの課題

- ✓ NTMfwを組み込めない
- ✗ 新しい機能の追加が許可されない
- ✗ 工場の出荷後はプログラムを書き換えられない

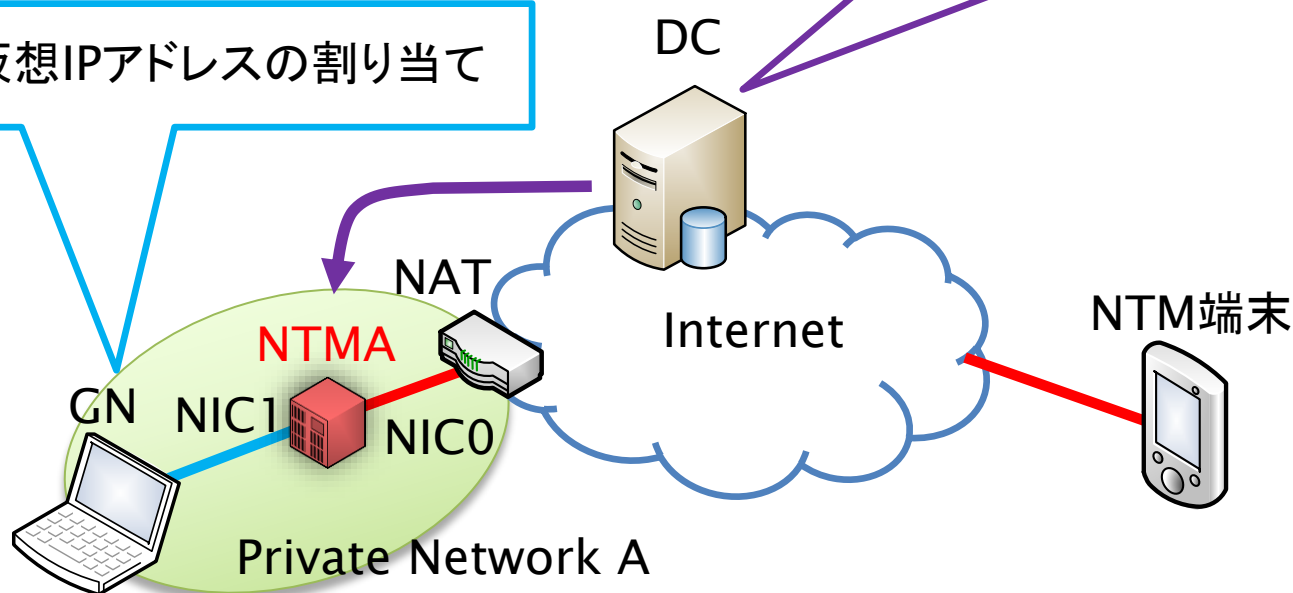
提案方式の構成 (イニシエータ)

- ◆ NICを2枚用意
 - NIC0をインターネット側にブリッジ接続
 - NIC1をGNにブリッジ接続

NIC: Network Interface Card
✓ LANに接続するための装置

仮想IPアドレスの割り当て

仮想IPアドレスの割り当て

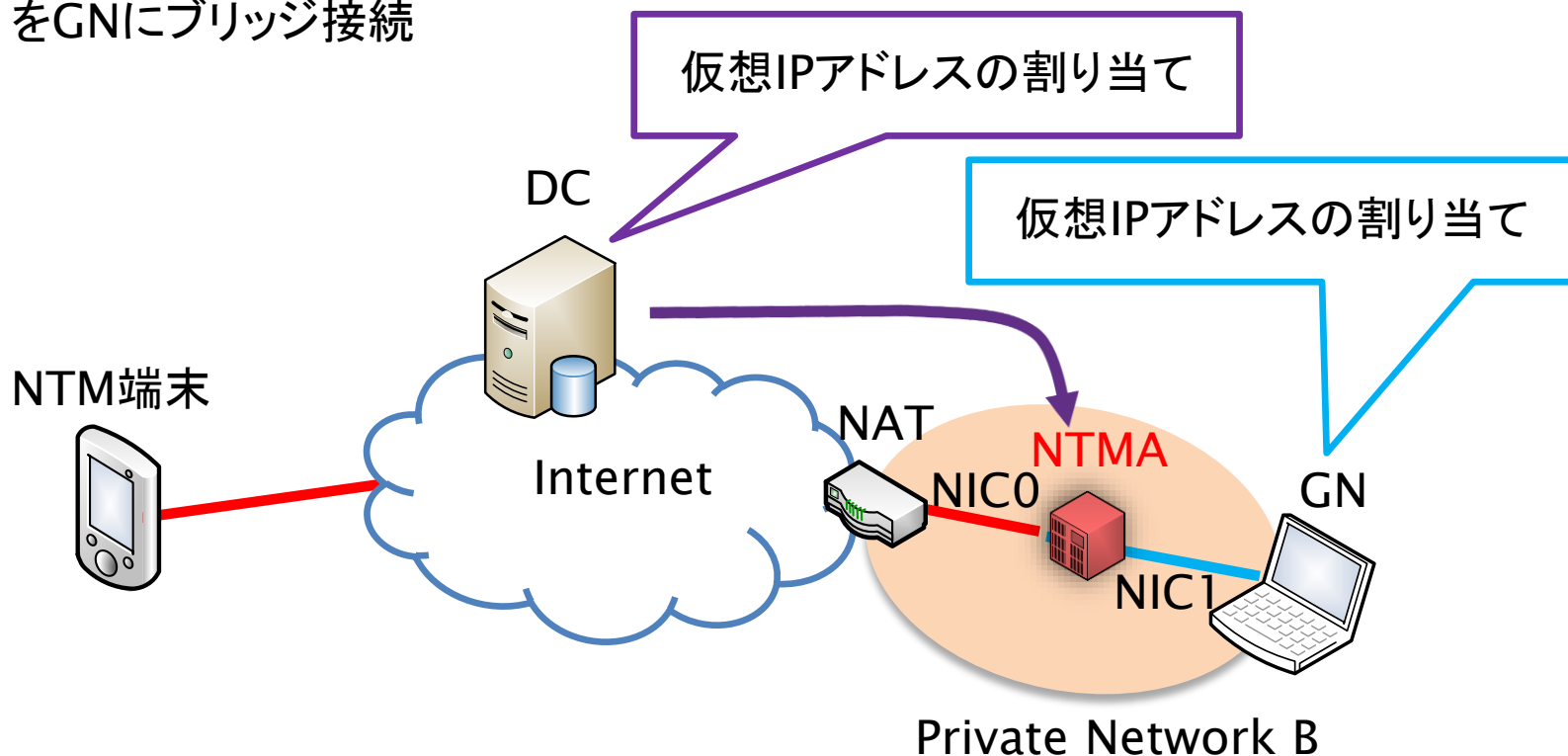


イーサネット0 イーサネット1

提案方式の構成 (レスポнда)

- ◆ NICを2枚用意
 - NIC0をインターネット側にブリッジ接続
 - NIC1をGNにブリッジ接続

NIC: Network Interface Card
✓ LANに接続するための装置

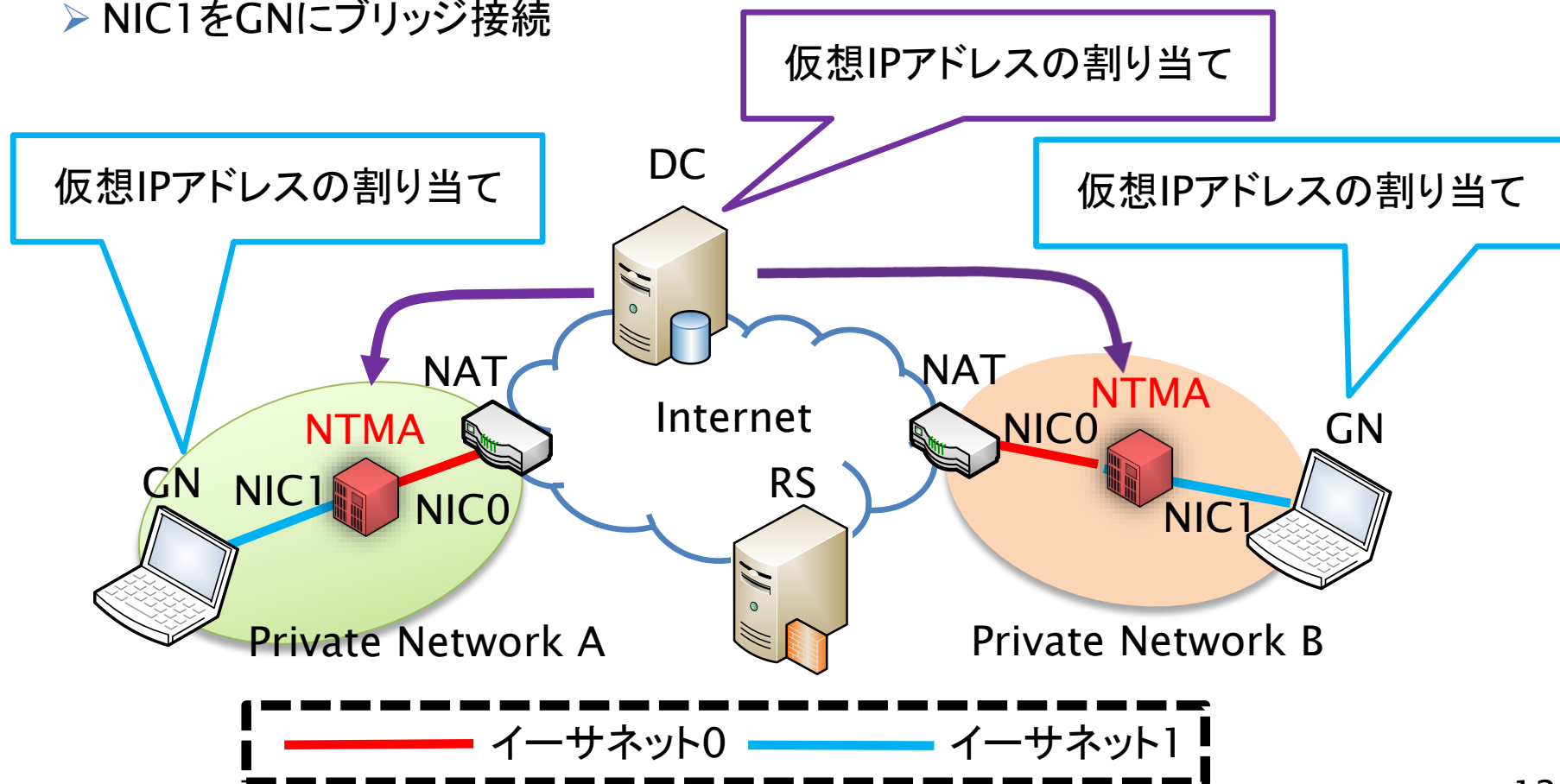


— イーサネット0 — イーサネット1

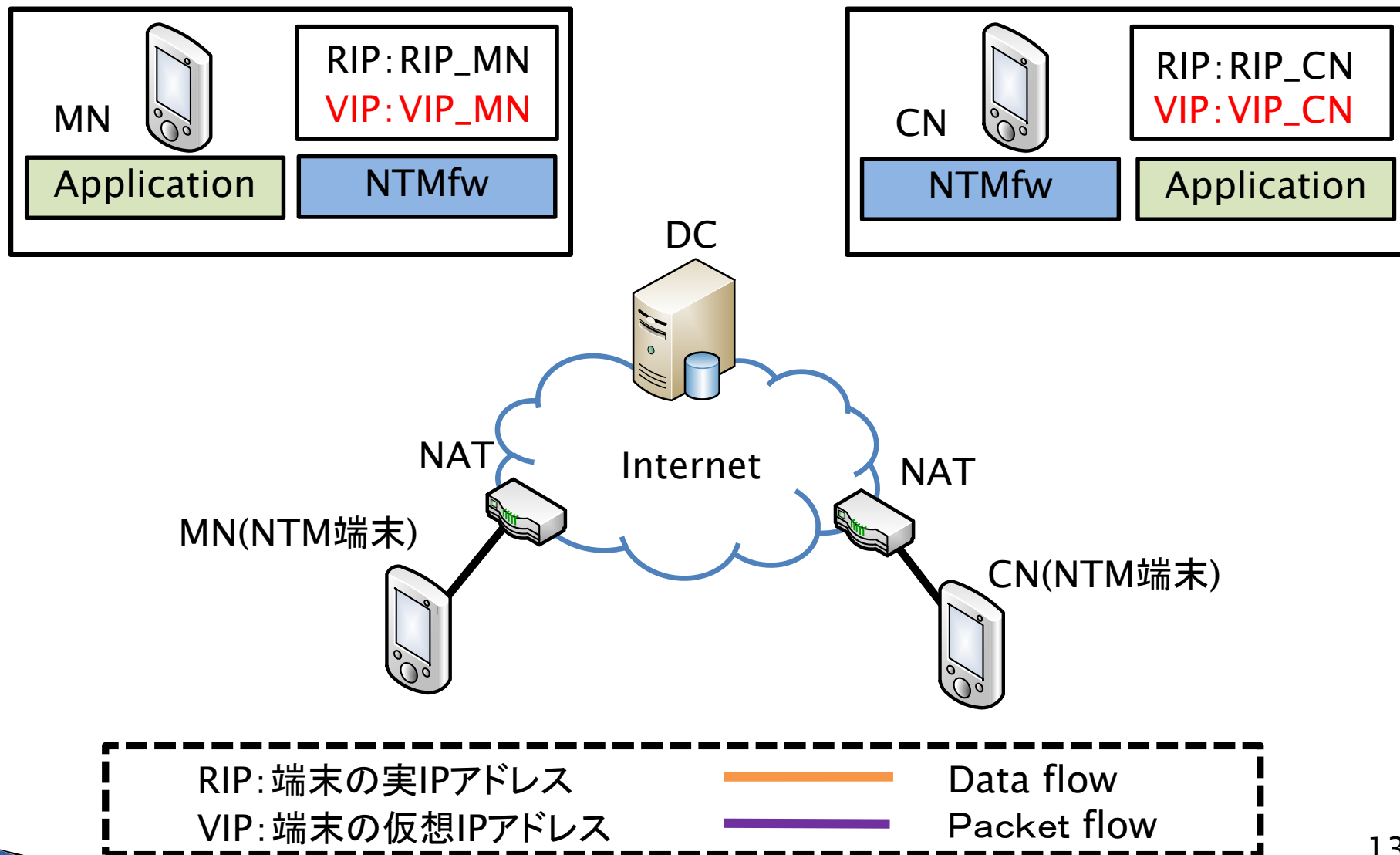
提案方式の構成 (イニシエータ/レスポнда)

- ◆ NICを2枚用意
 - NIC0をインターネット側にブリッジ接続
 - NIC1をGNにブリッジ接続

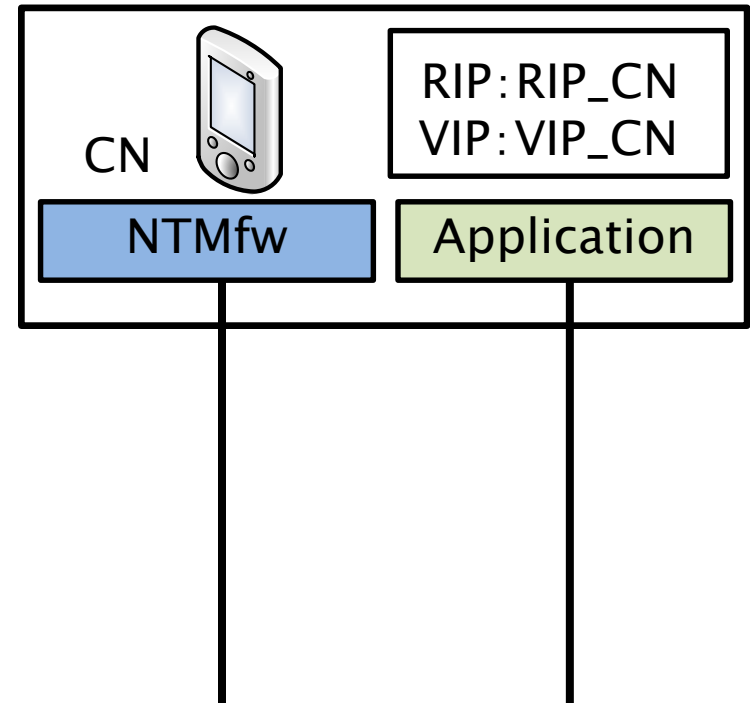
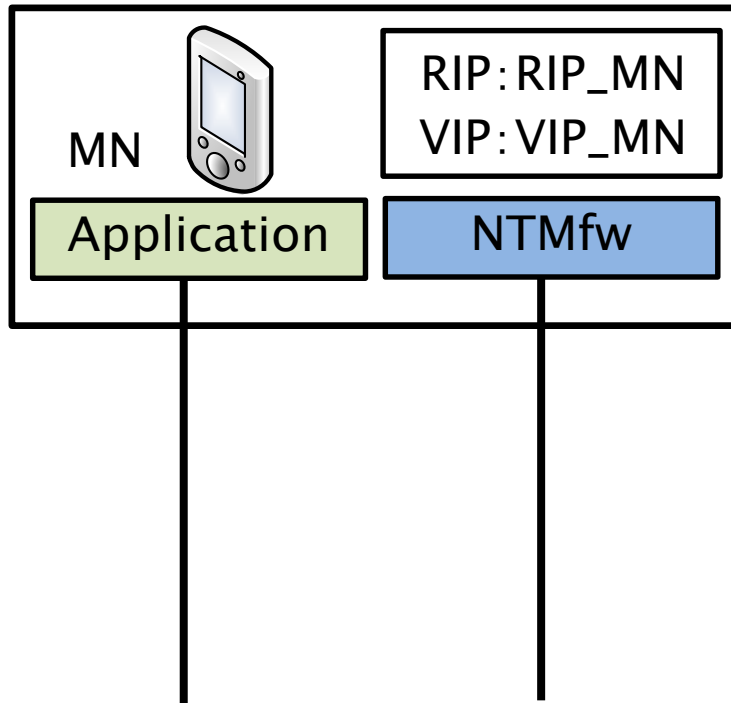
NIC: Network Interface Card
✓ LANに接続するための装置



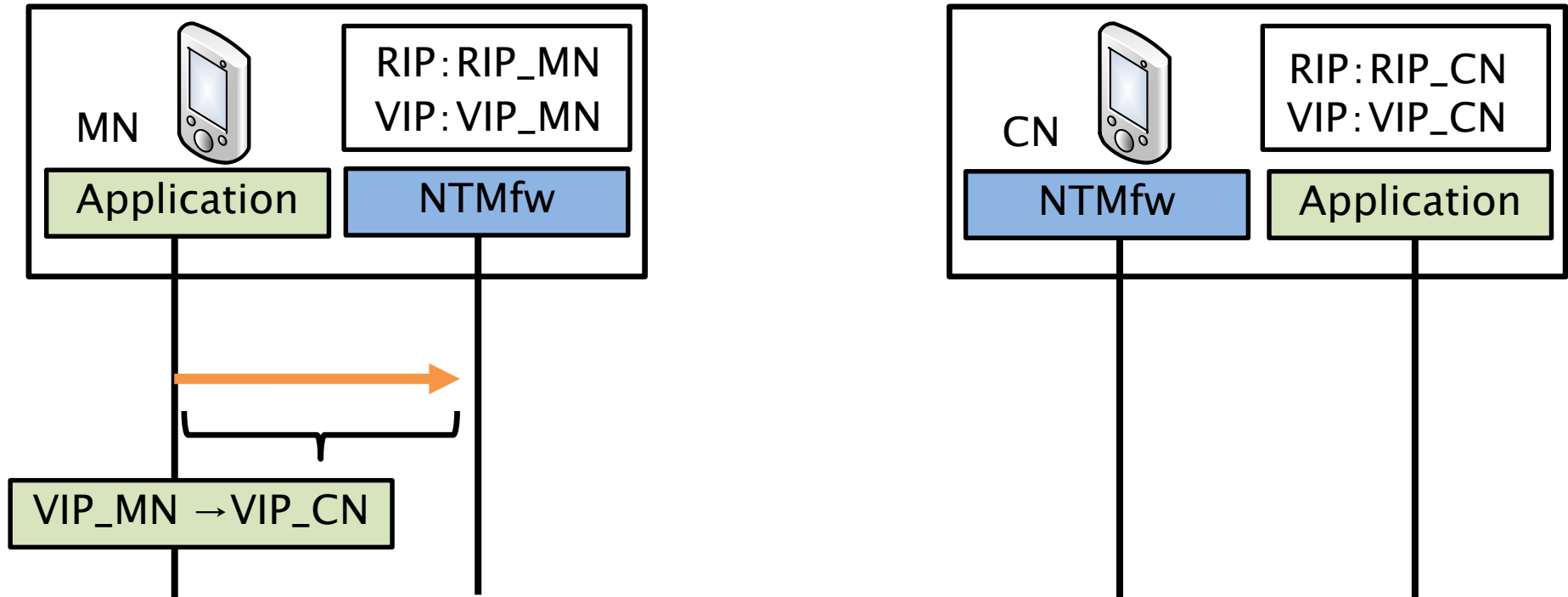
NTMobile (通信時)



NTMobile (通信時)



NTMobile (通信時)



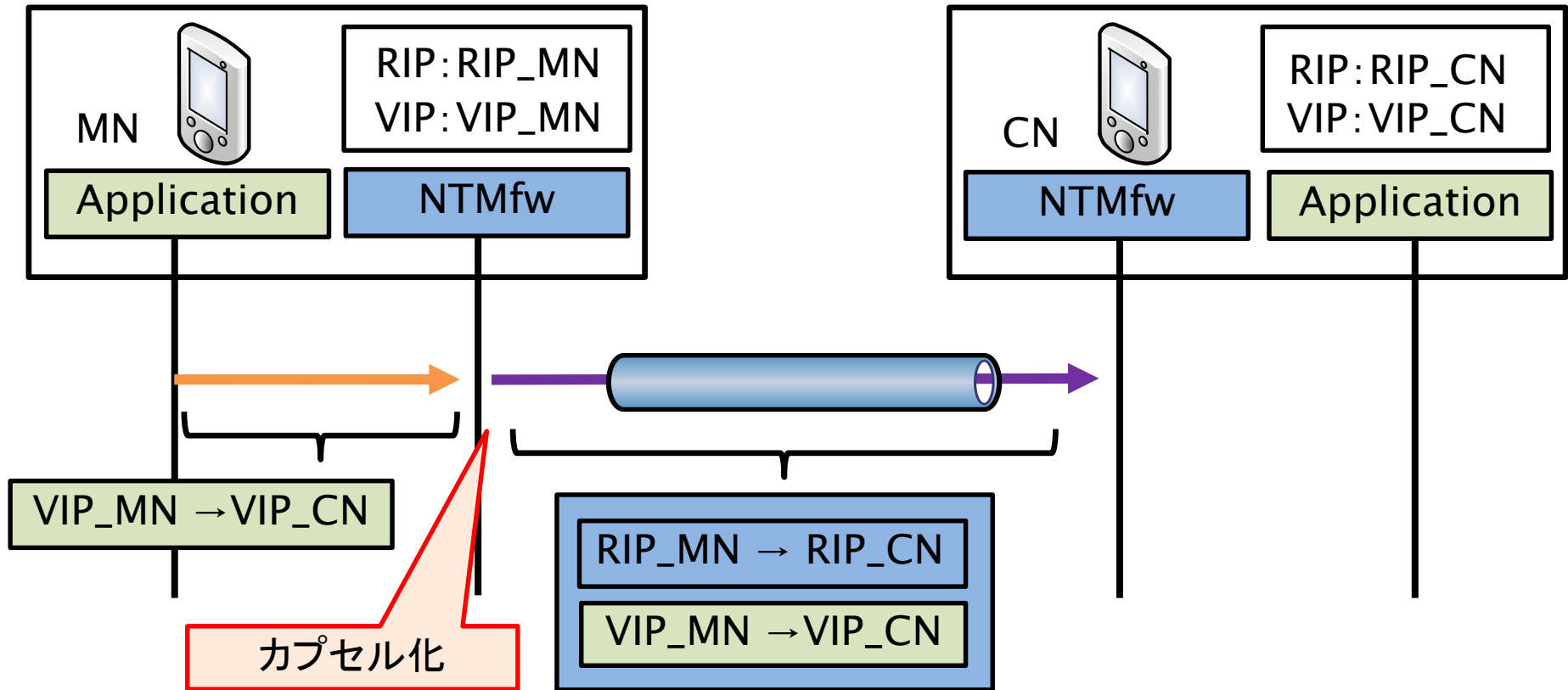
RIP: 端末の実IPアドレス

VIP: 端末の仮想IPアドレス

Data flow

Packet flow

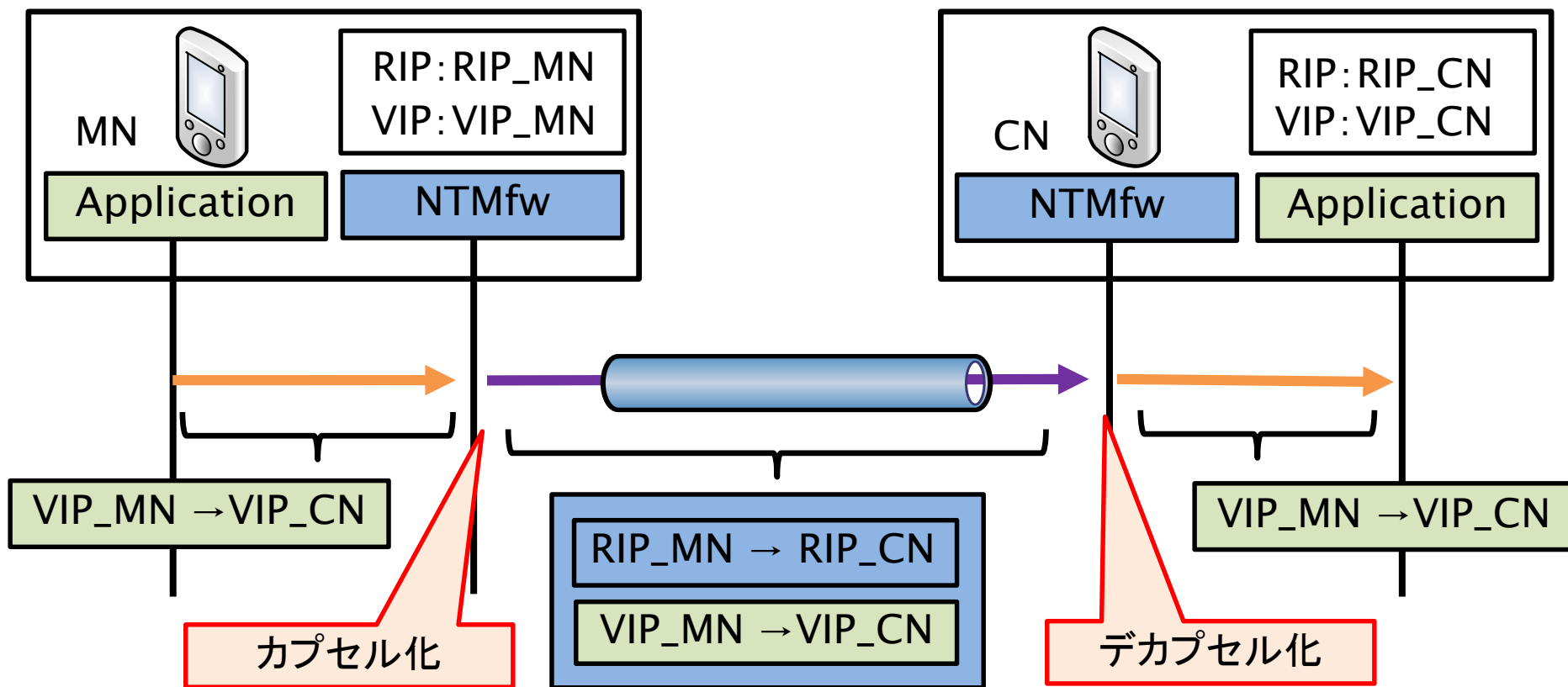
NTMobile (通信時)



RIP: 端末の実IPアドレス
 VIP: 端末の仮想IPアドレス

— Data flow
 — Packet flow

NTMobile (通信時)



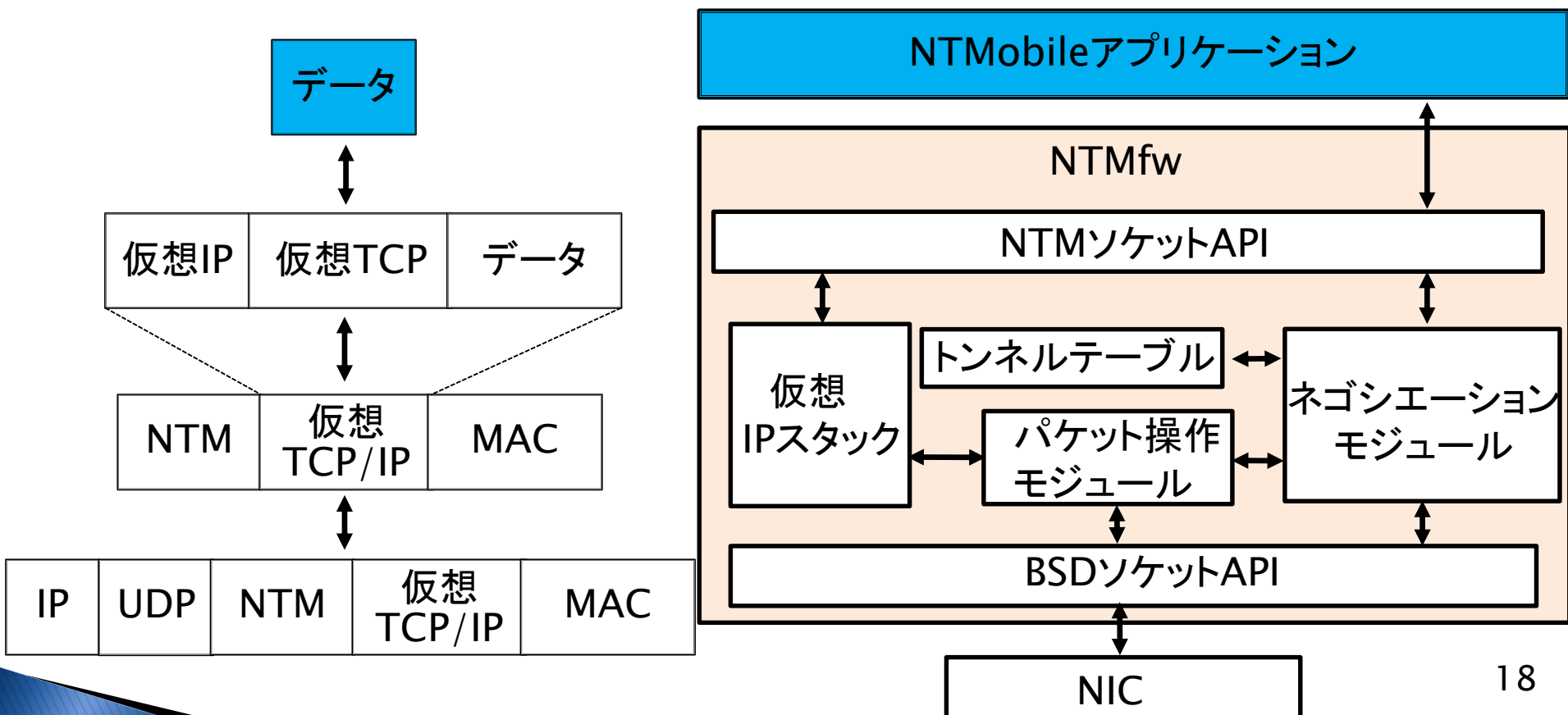
RIP: 端末の実IPアドレス
 VIP: 端末の仮想IPアドレス

Data flow
 Packet flow

NTMfwのパケット遷移

NTMobileアプリケーション

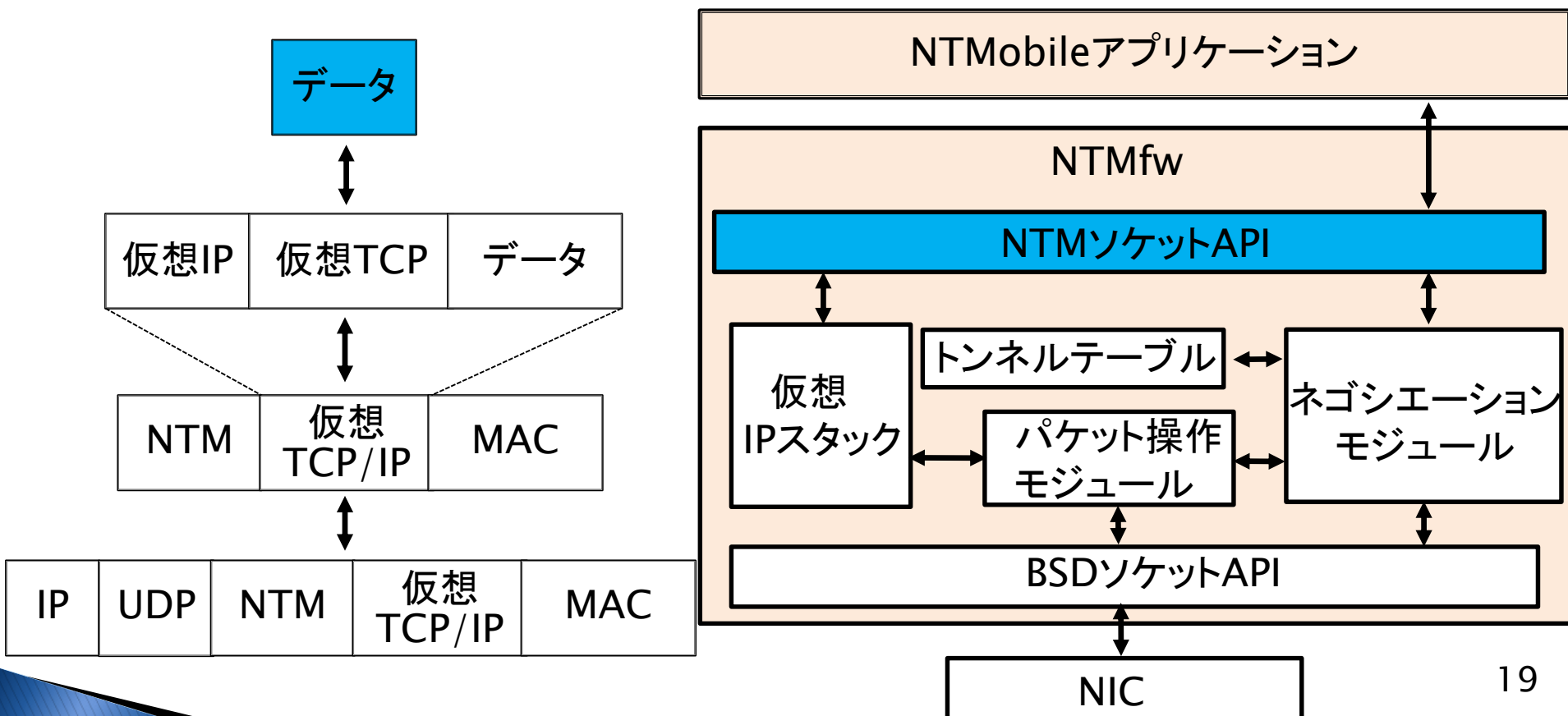
- NTMソケットAPIで記述されたアプリケーション



NTMfwのパケット遷移

NTMソケットAPI

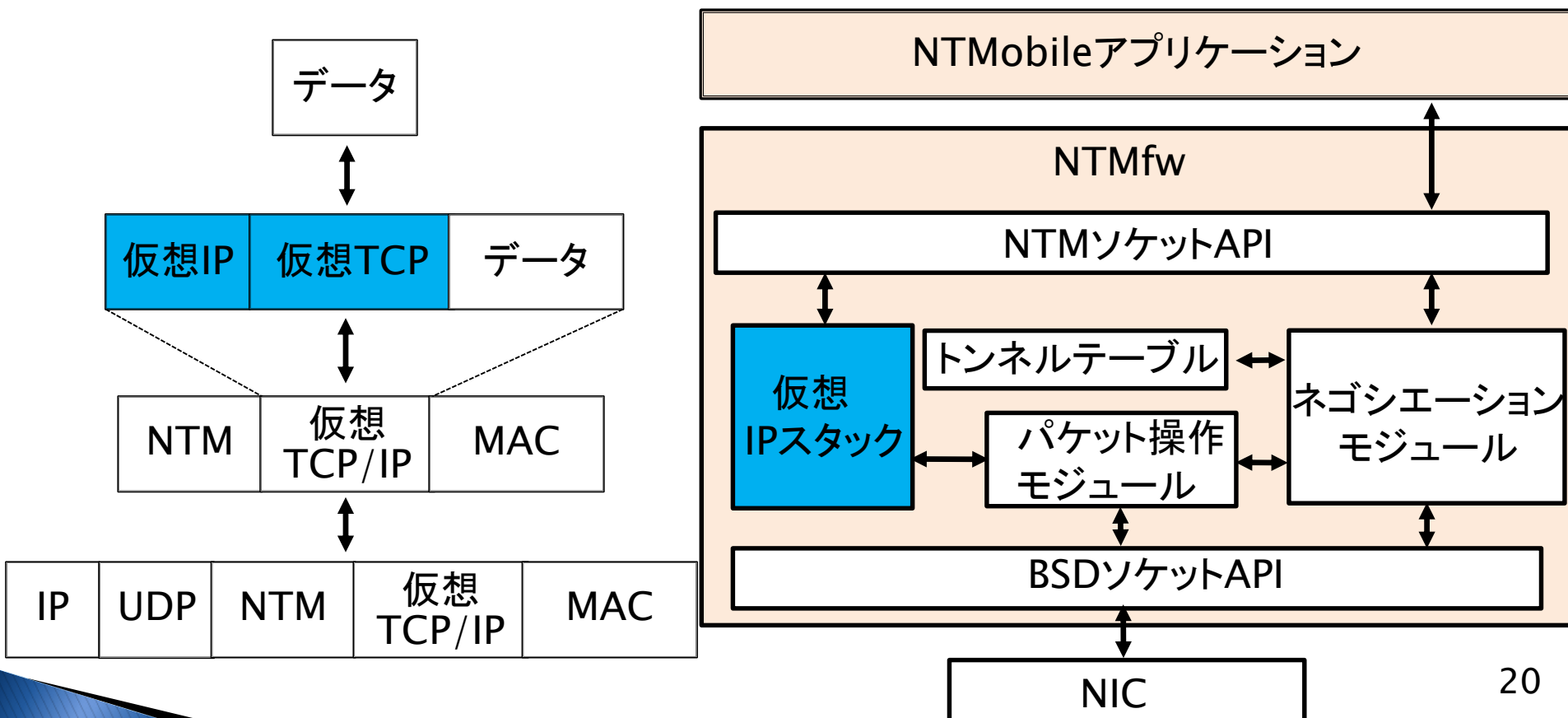
- BSDソケットAPI互換のソケットAPI
- アプリケーションデータを仮想IPスタックへ渡す



NTMfwのパケット遷移

仮想IPスタック

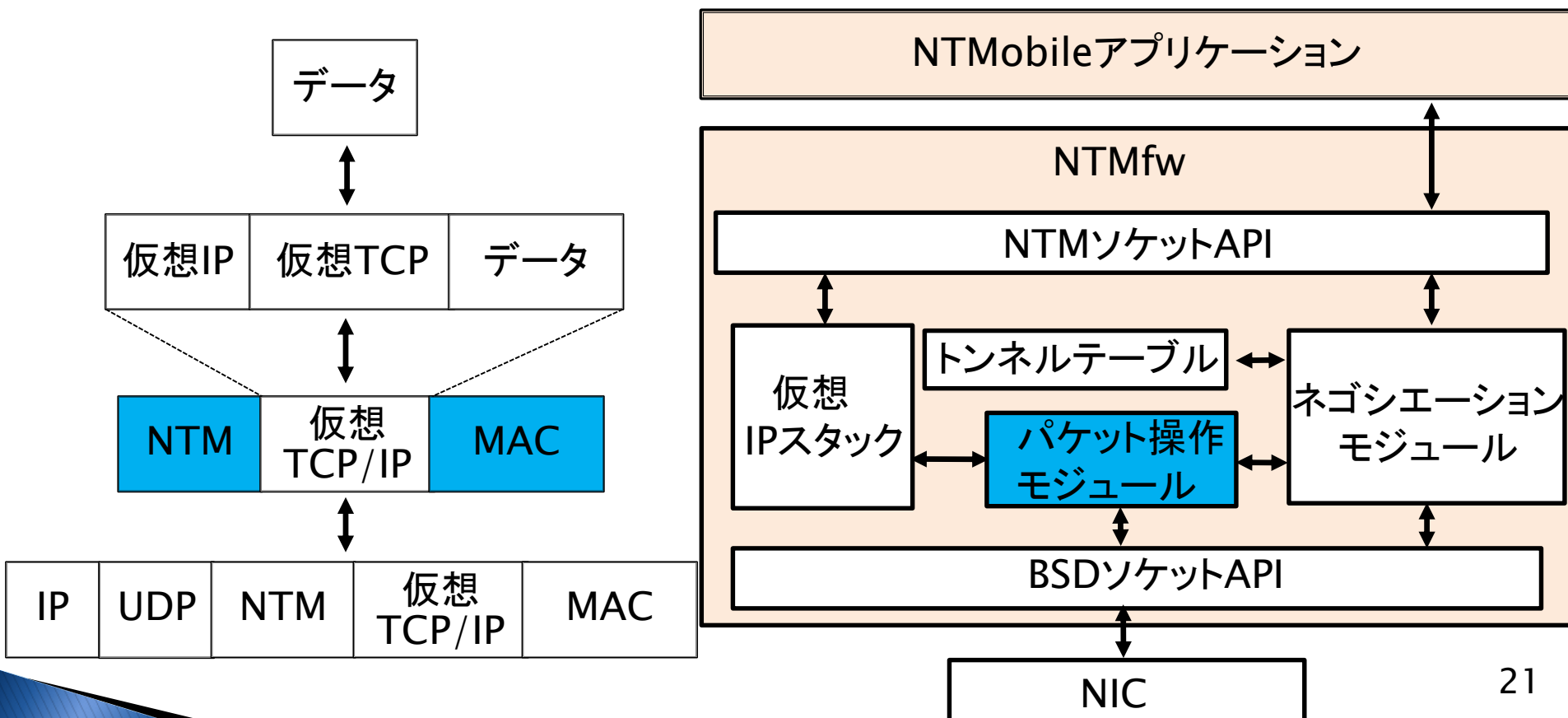
- アプリケーションデータから仮想TCP/IPパケットを生成



NTMfwのパケット遷移

パケット操作モジュール

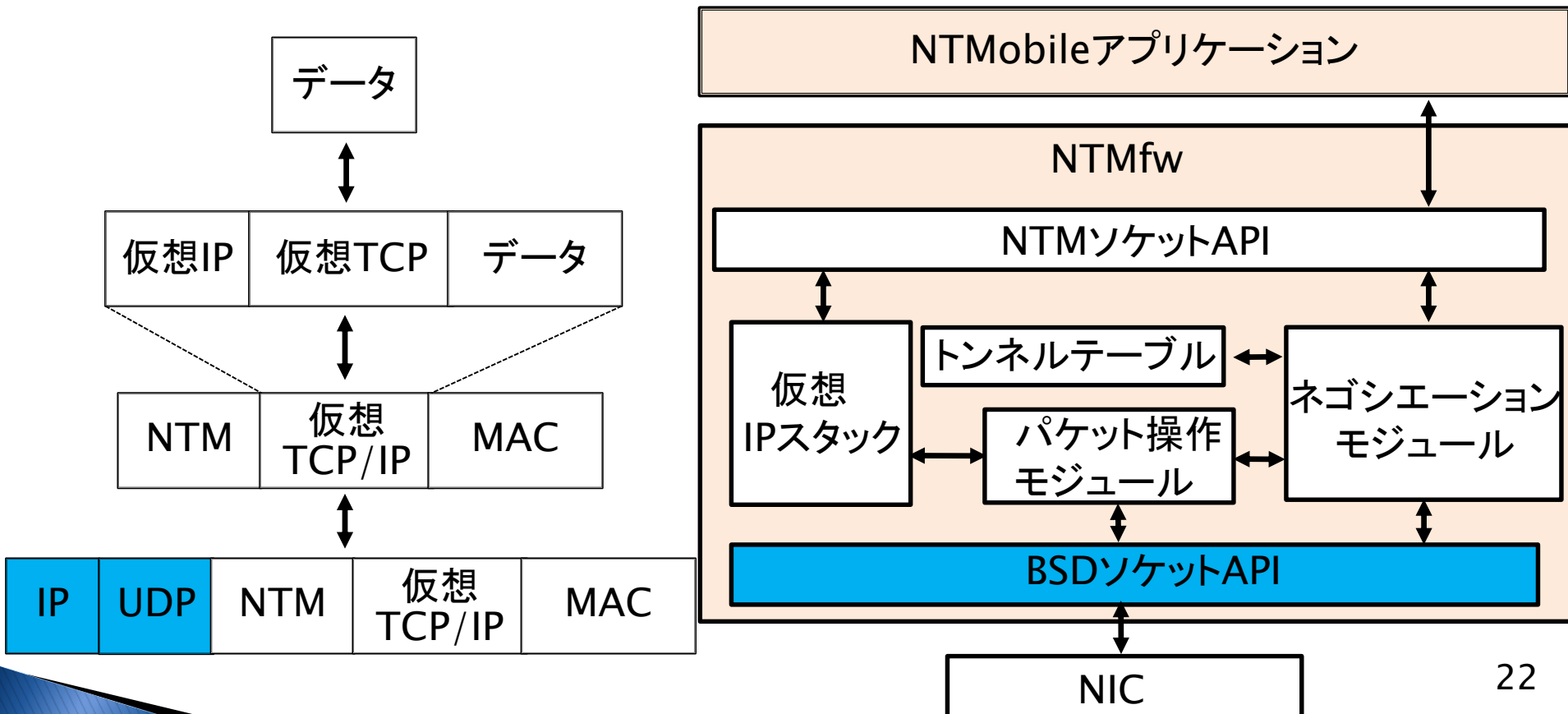
- NTMヘッダの付与
- MAC付与/検証等



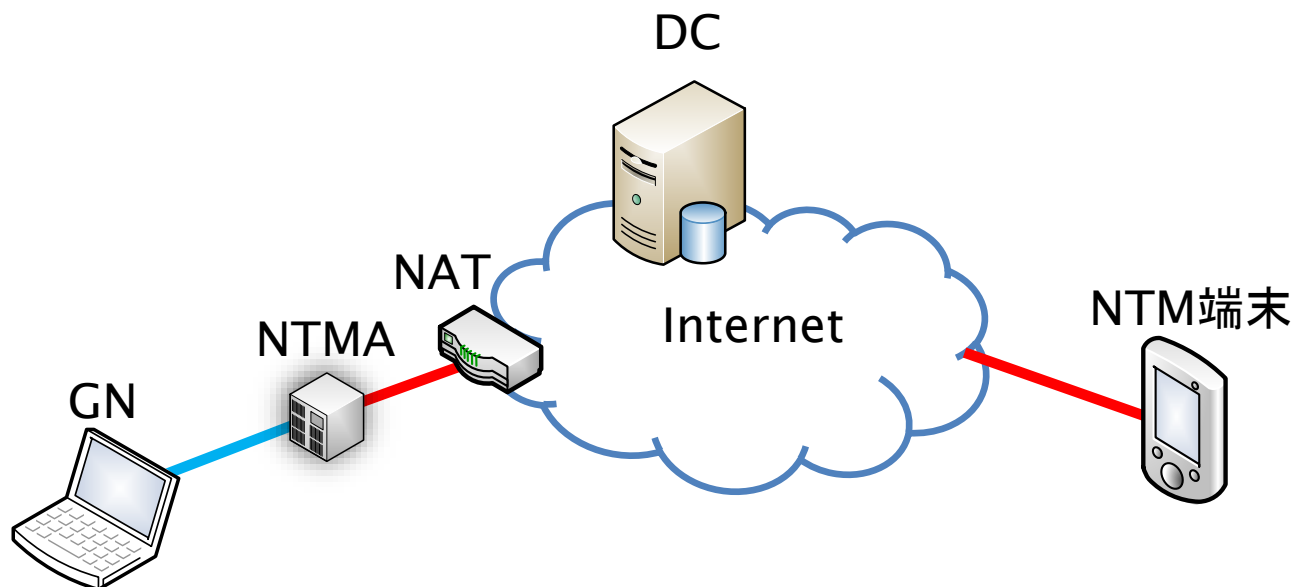
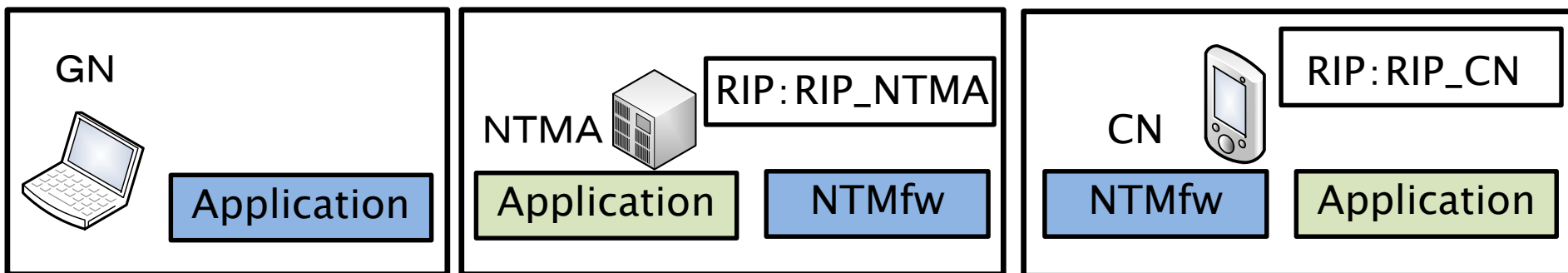
NTMfwのパケット遷移

BSDソケットAPI

- C言語のソケットAPI
- データパケットの送受信(カプセル化/デカプセル化)



提案方式 (通信時)

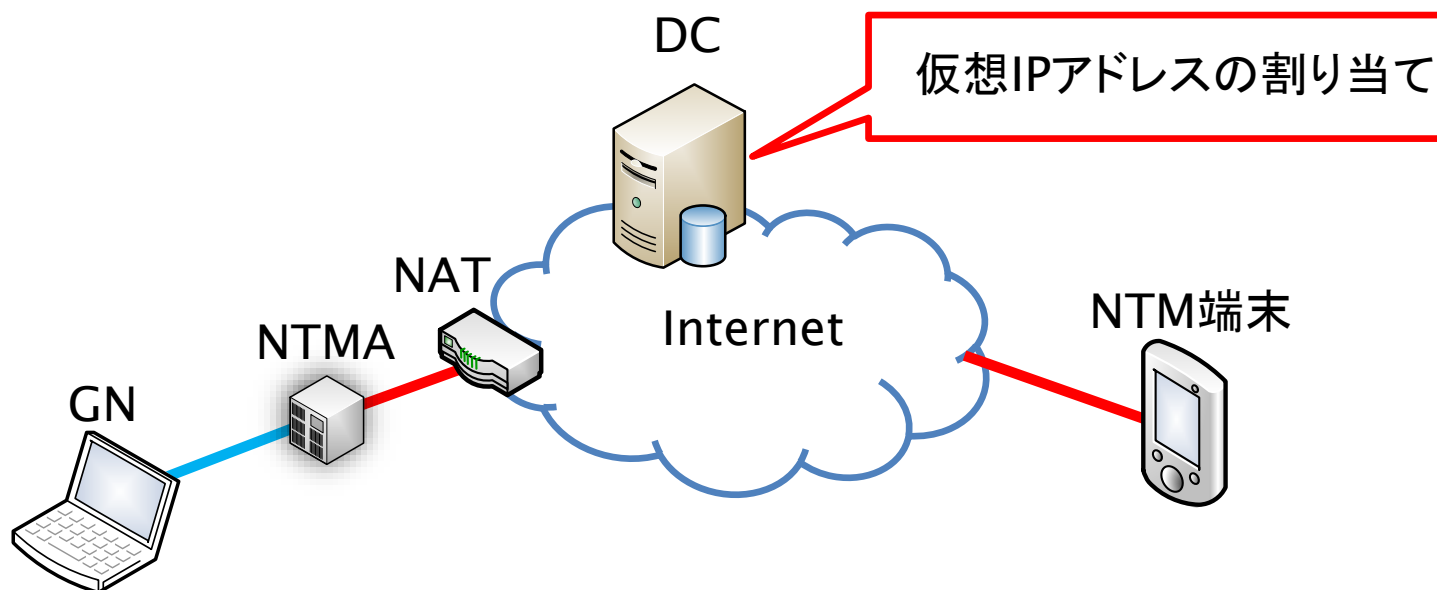
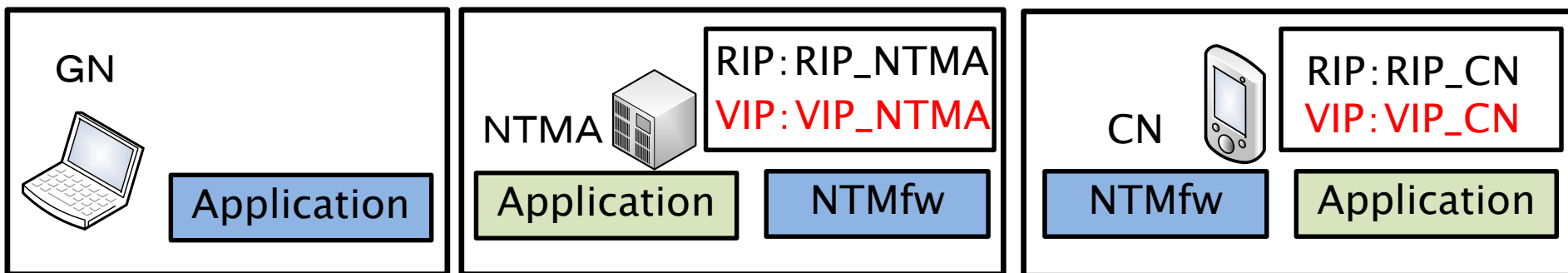


RIP: 端末の実IPアドレス
VIP: 端末の仮想IPアドレス



Data flow
Packet flow

提案方式 (通信時)

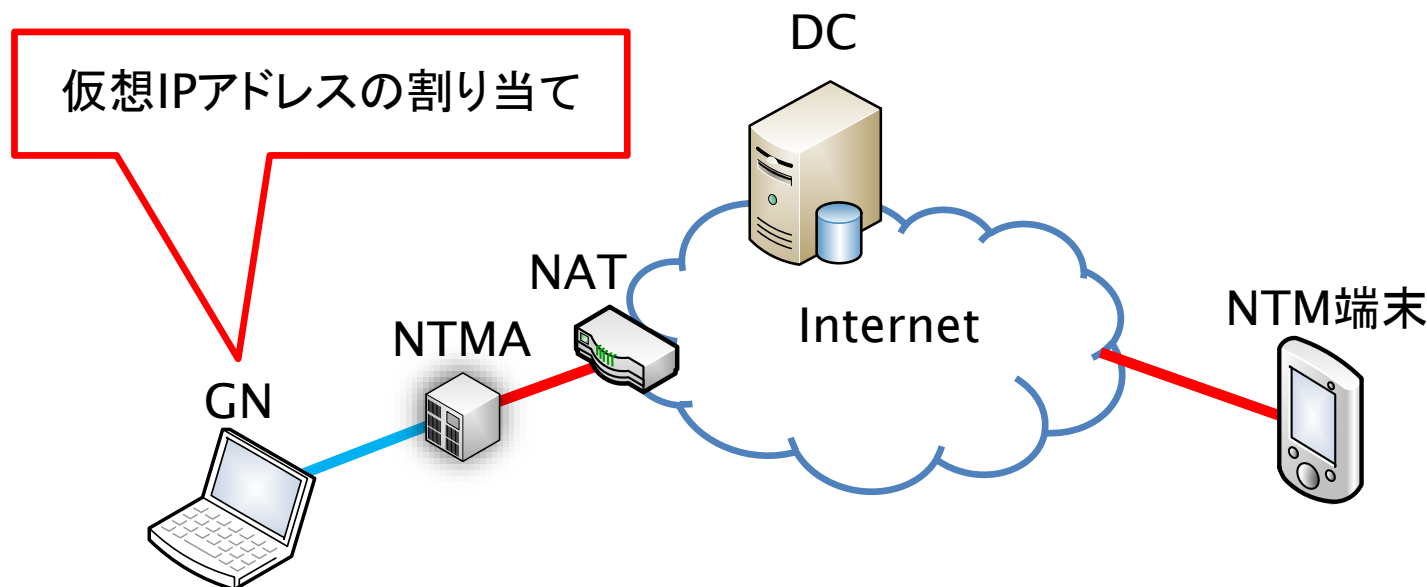
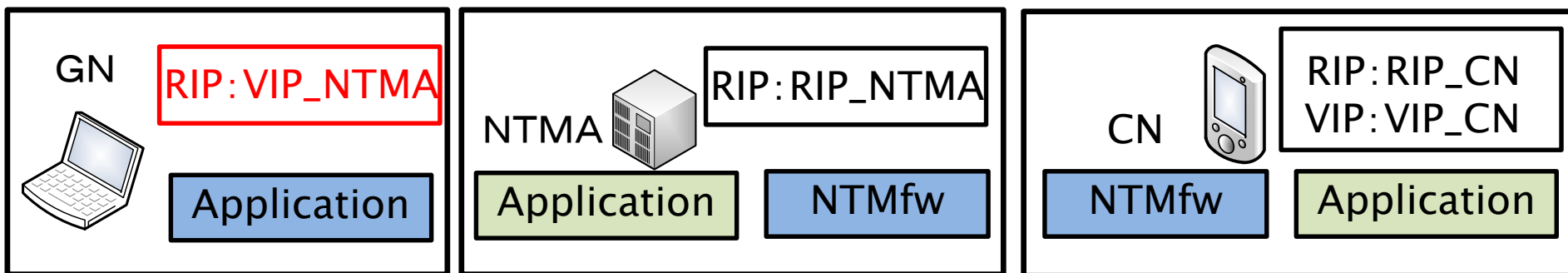


RIP: 端末の実IPアドレス
VIP: 端末の仮想IPアドレス

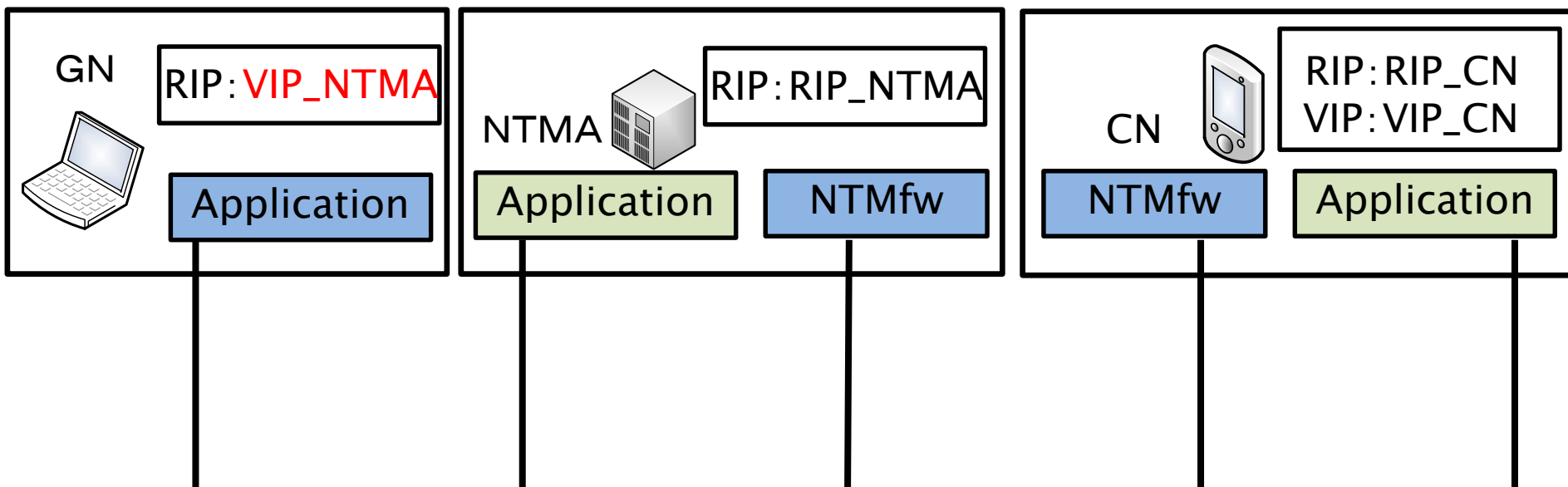


Data flow
Packet flow

提案方式 (通信時)



提案方式 (通信時)

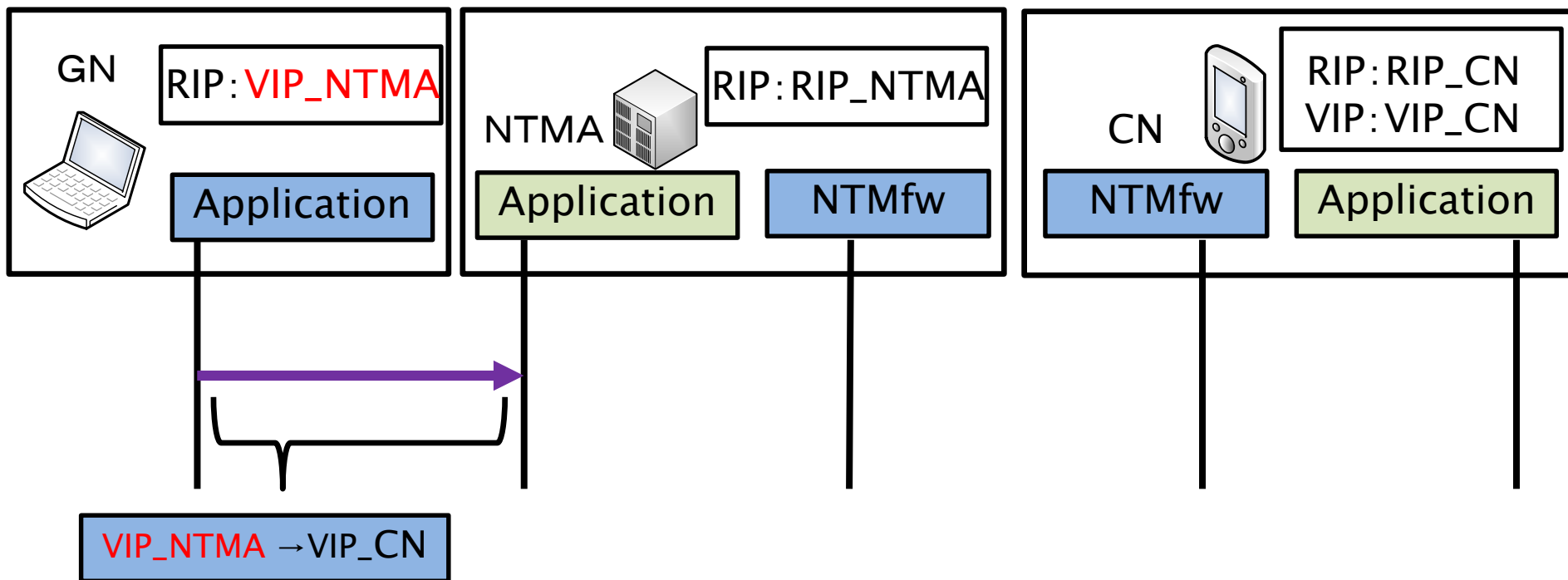


RIP: 端末の実IPアドレス
VIP: 端末の仮想IPアドレス



Data flow
Packet flow

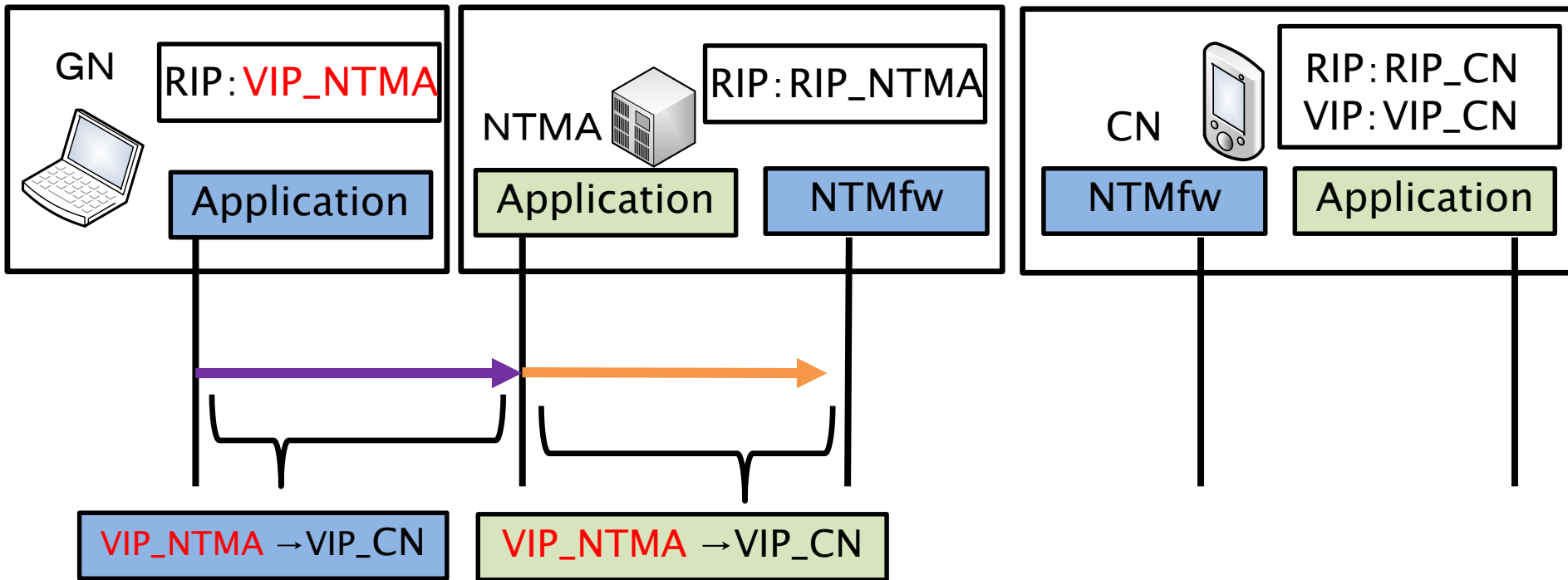
提案方式 (通信時)



RIP: 端末の実IPアドレス
VIP: 端末の仮想IPアドレス

Data flow
Packet flow

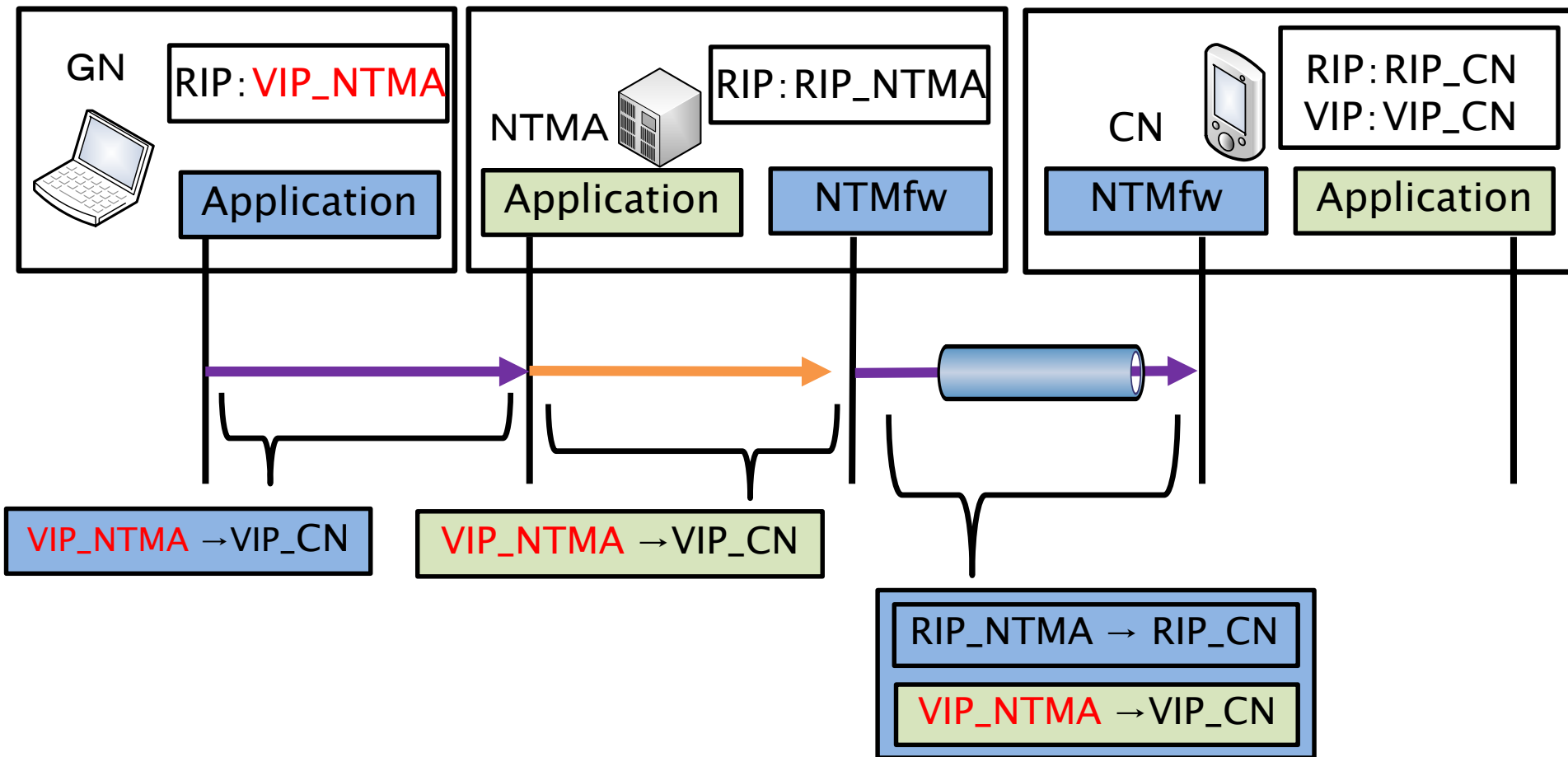
提案方式 (通信時)



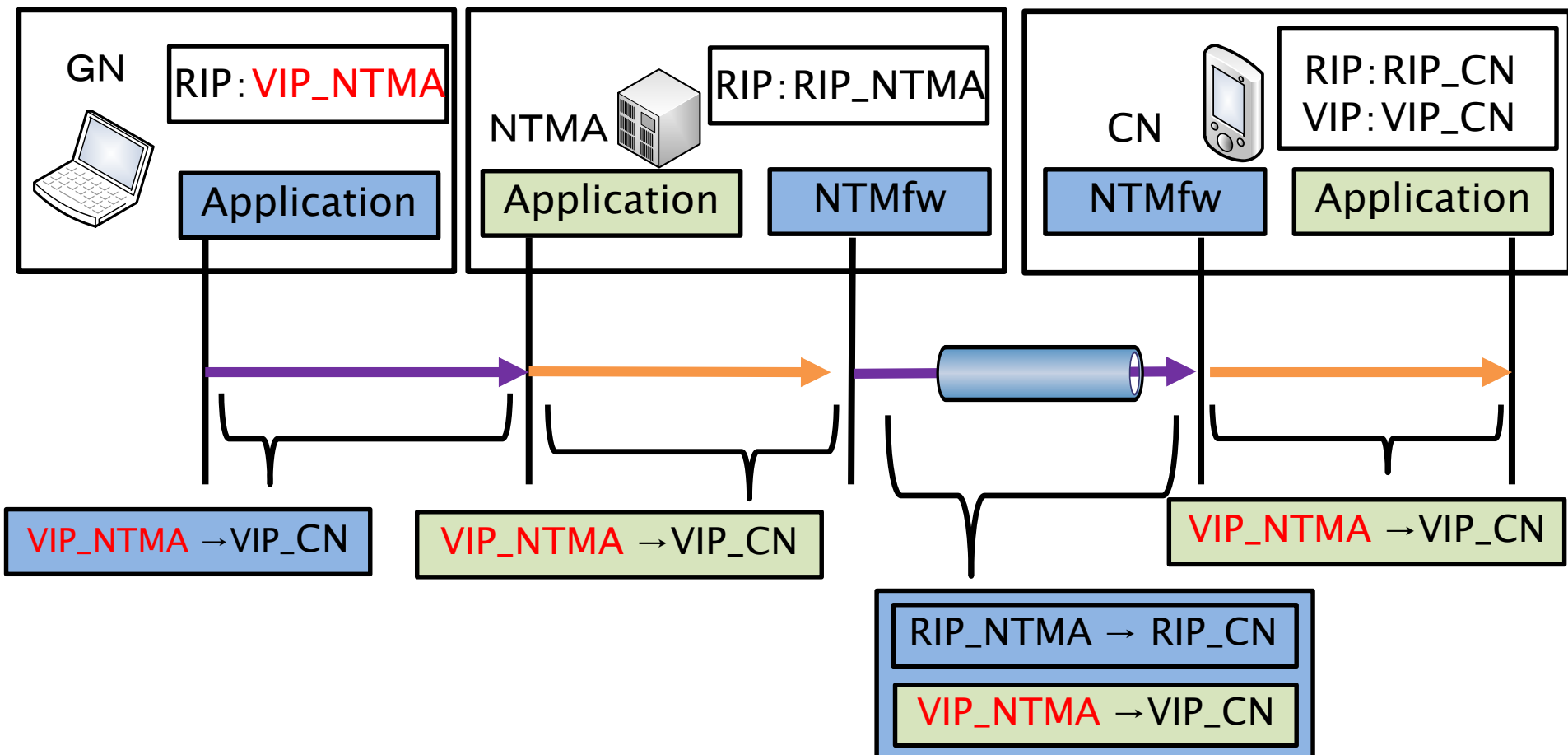
RIP: 端末の実IPアドレス
VIP: 端末の仮想IPアドレス

Data flow
Packet flow

提案方式 (通信時)



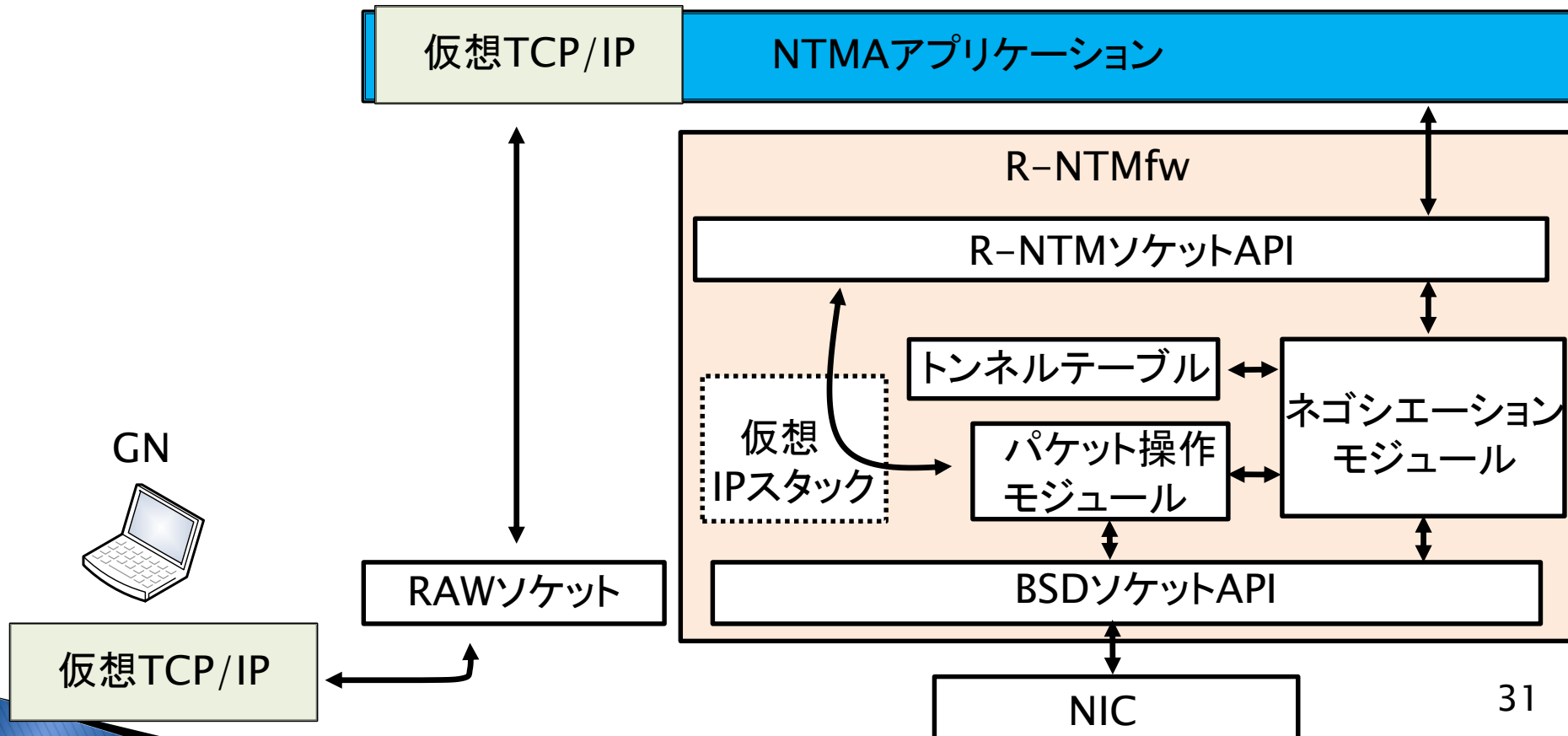
提案方式 (通信時)



NTMAのモジュール構成

NTMAアプリケーション

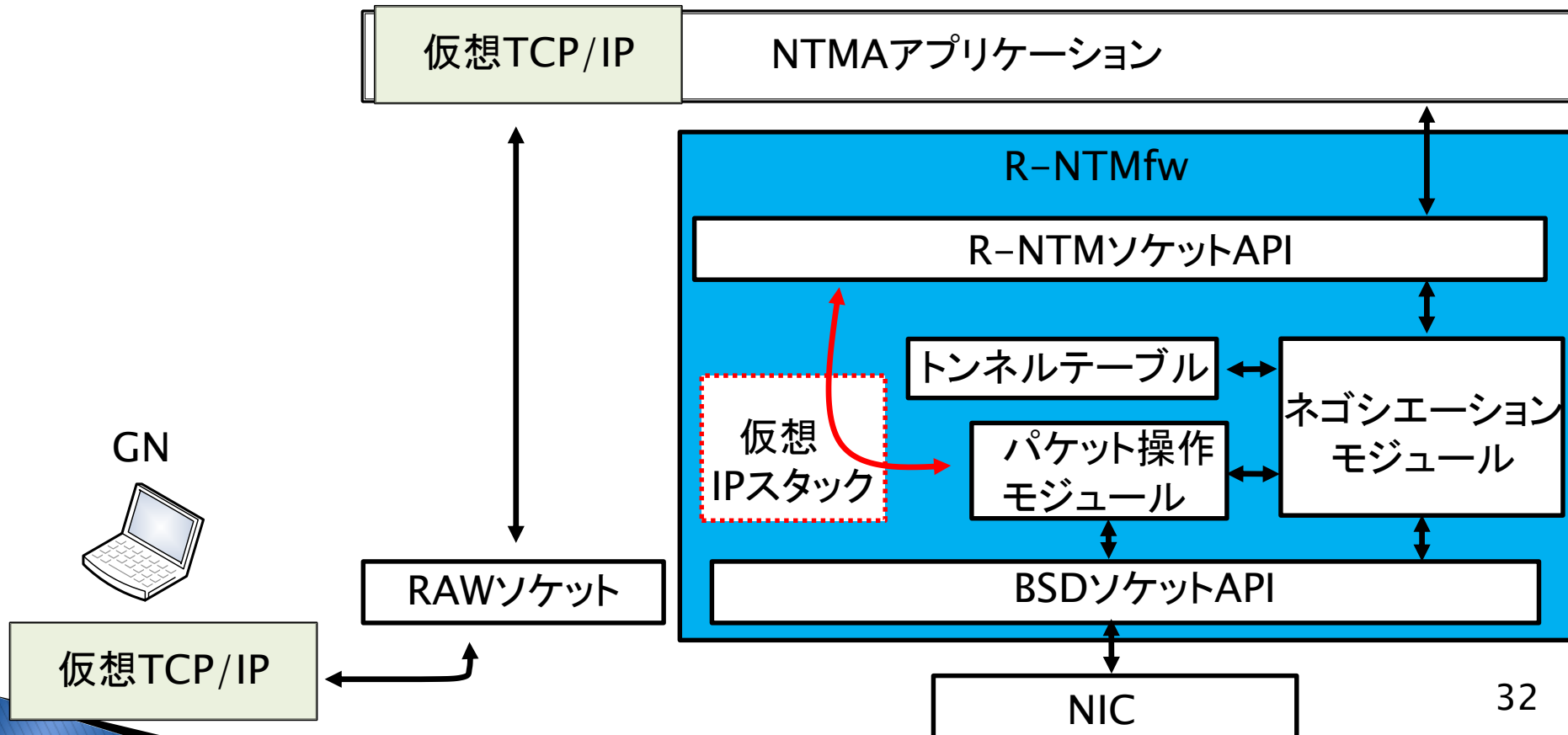
- NTM端末との通信は, R-NTMfw(Remodeled NTMfw)のAPIで送受信
- GNとの通信は, RAWソケットで送受信



NTMAのモジュール構成

R-NTMfw

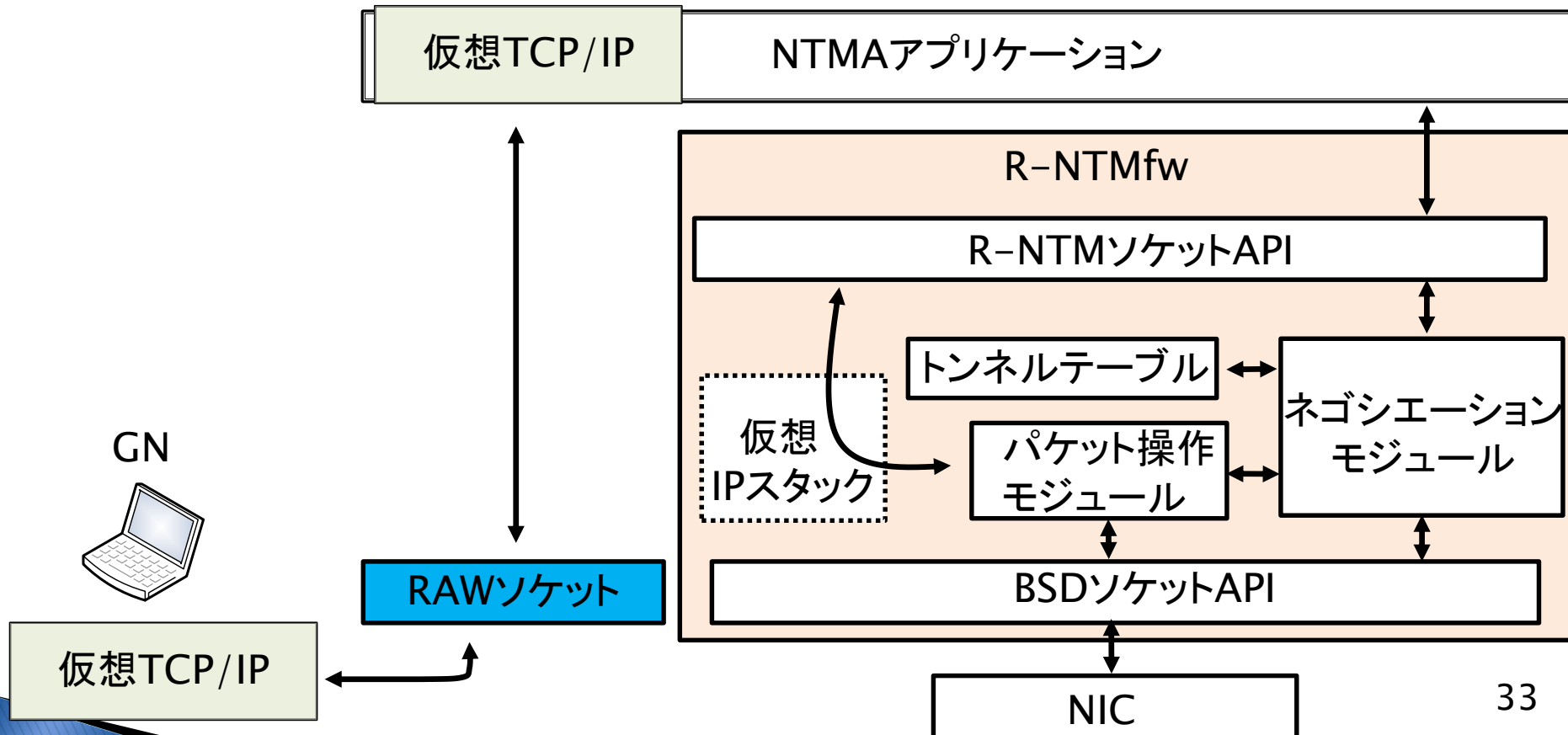
- NTMfwから仮想IPスタックの処理をスキップ



NTMAのモジュール構成

RAWソケット

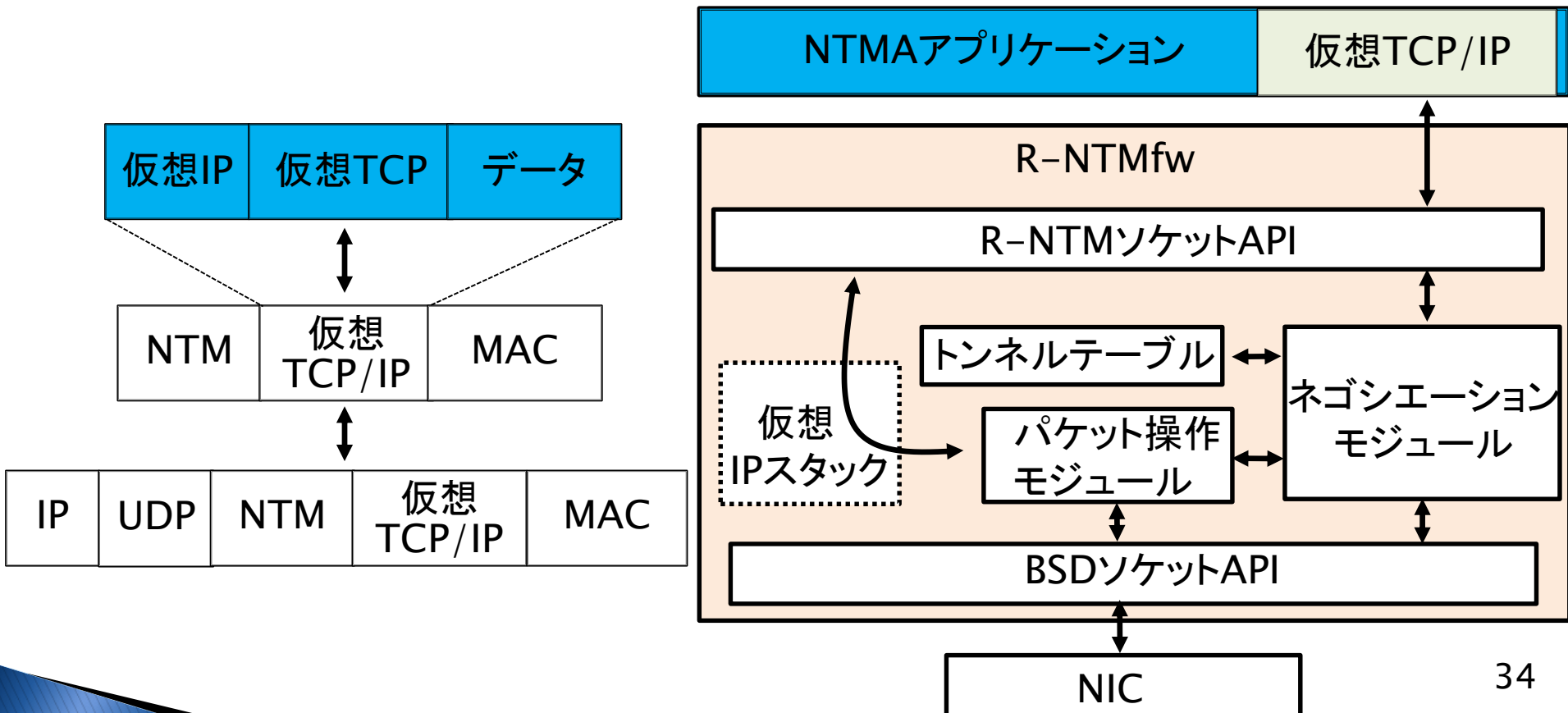
- 生のパケットをダイレクトに送受信
- 自インターフェース宛/以外のパケットも送受信できるように設定



NTMAのパケット遷移

NTMAアプリケーション

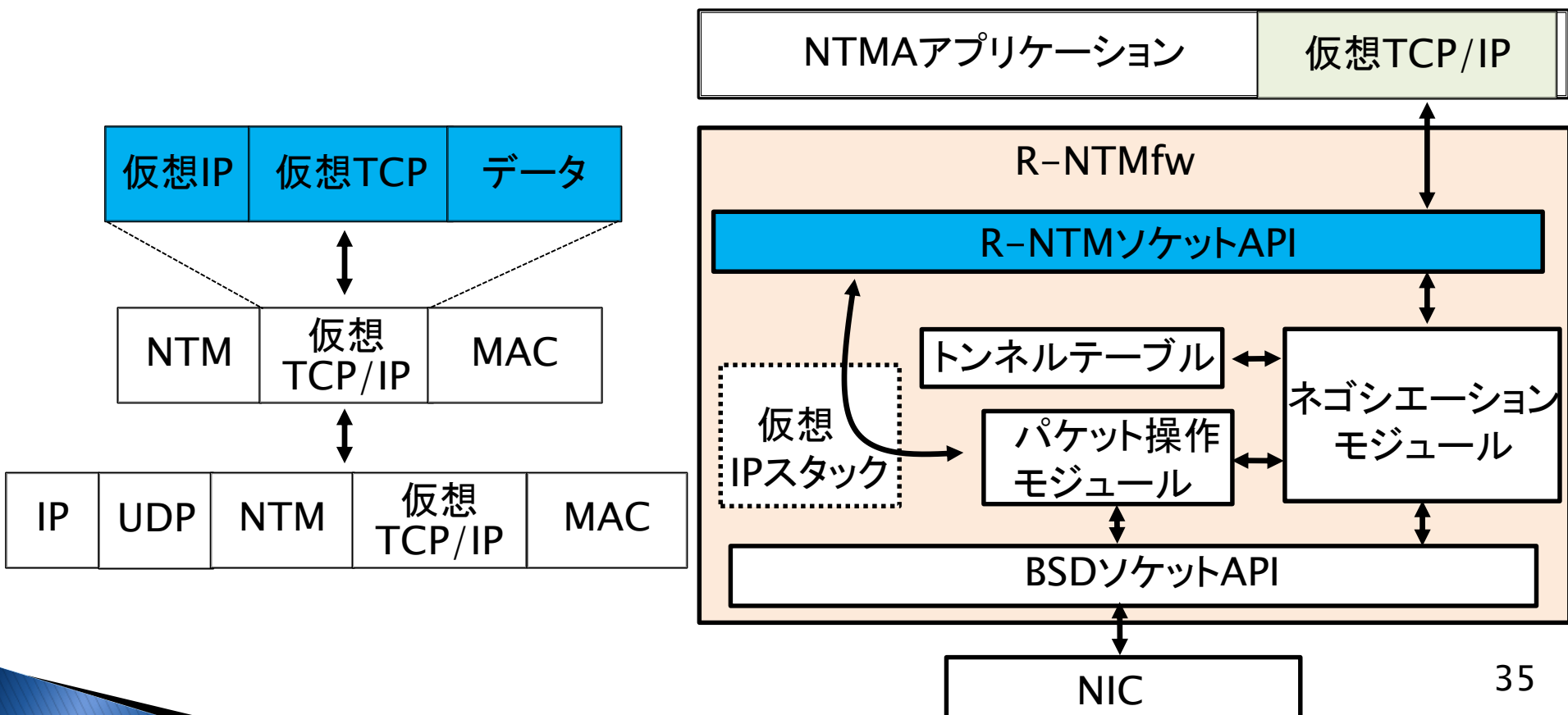
- GNとの通信は, RAWソケットで送受信
- NTM端末との通信は, R-NTMfwのAPIで送受信



NTMAのパケット遷移

R-NTMfwソケットAPI

- 仮想IPスタックの処理をスキップするNTMfwのAPI
- パケット操作モジュールへ処理を渡す



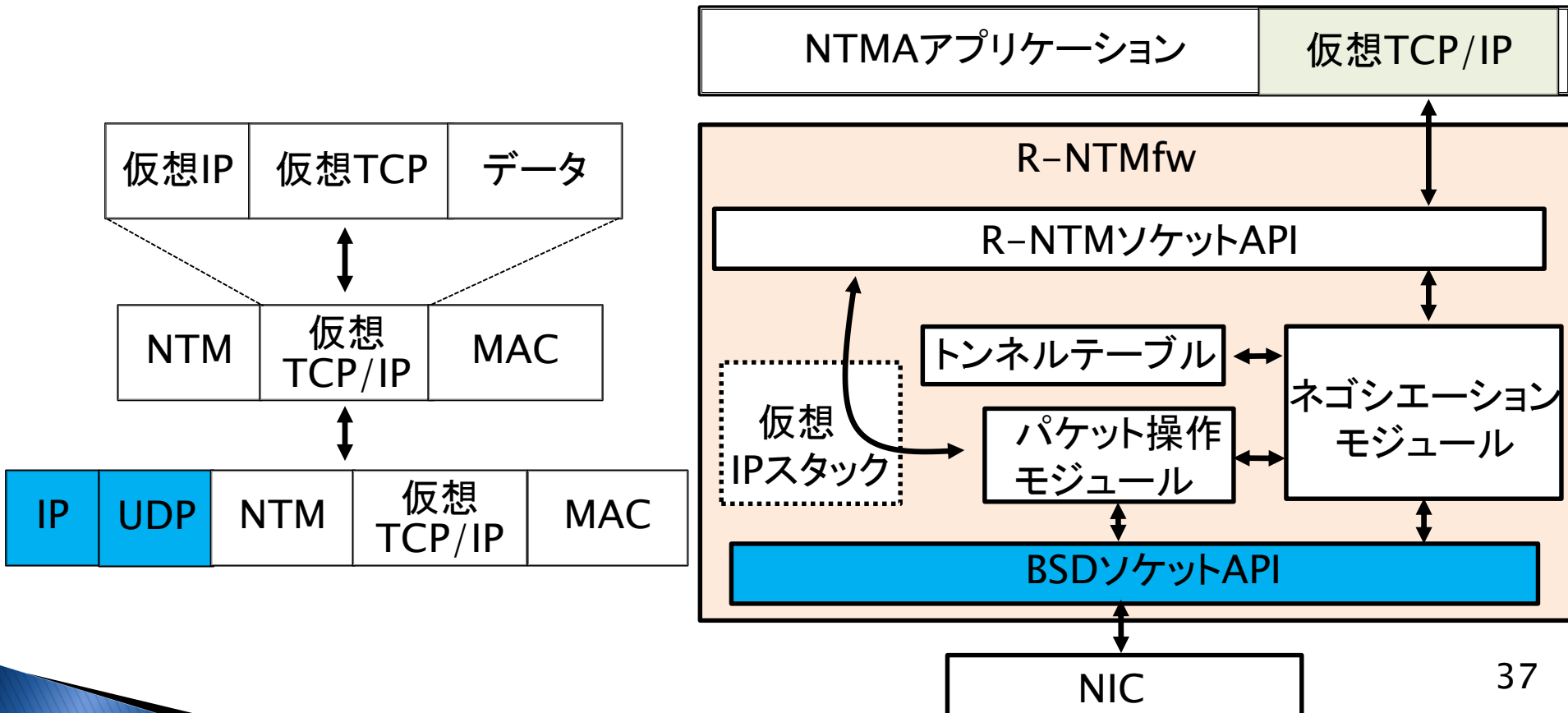
- NTMfwと同様の処理



NTMAのパケット遷移

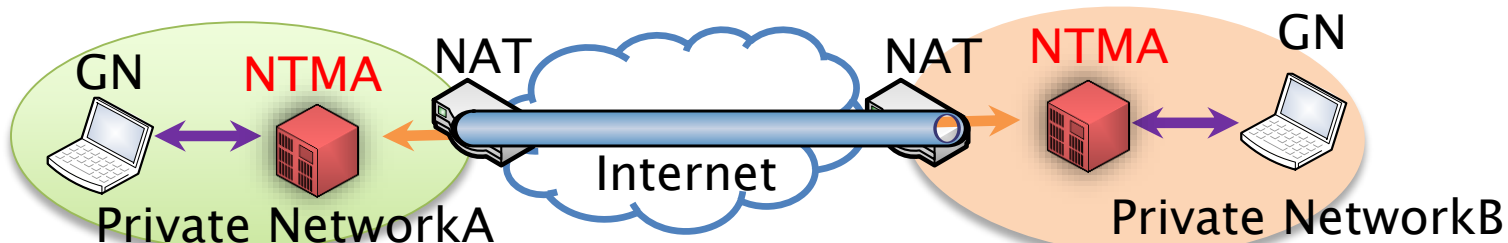
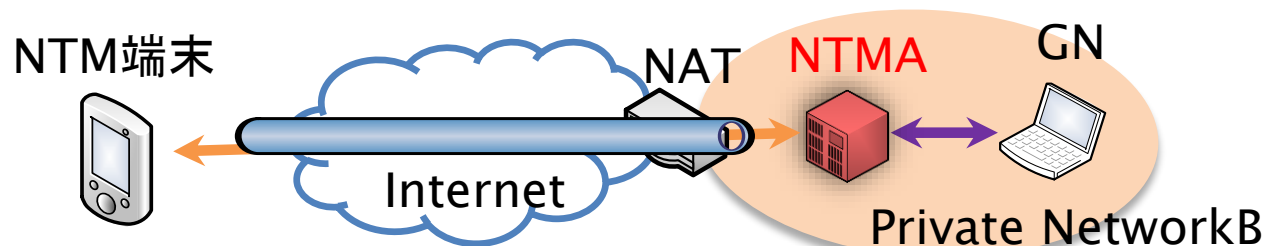
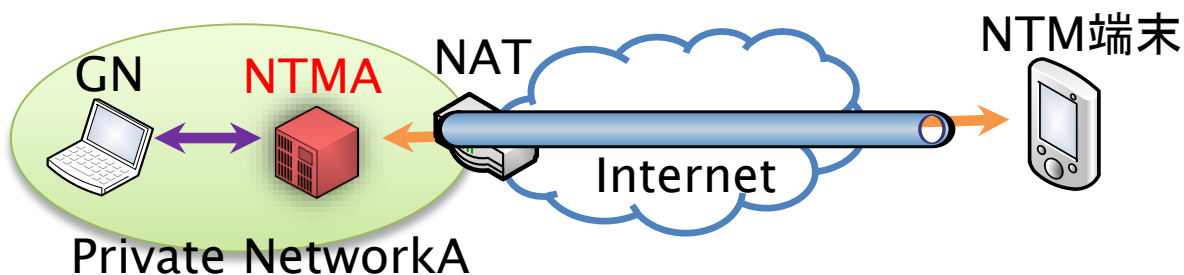
BSDソケットAPI

- NTMfwと同様の処理



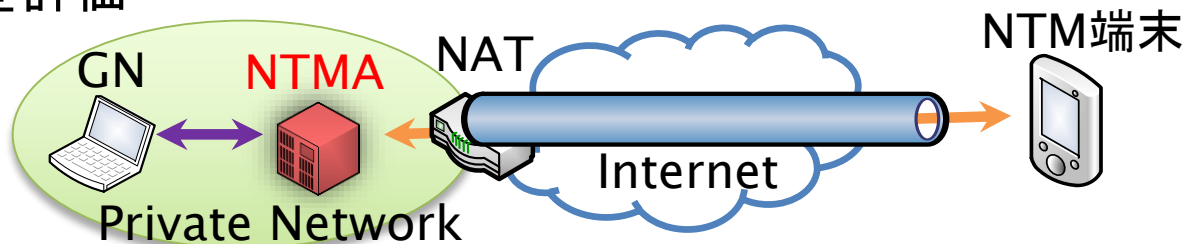
動作検証

- ◆ 提案方式をLinux上に実装
 - 一般通信をNTMobile通信に変換する動作を確認



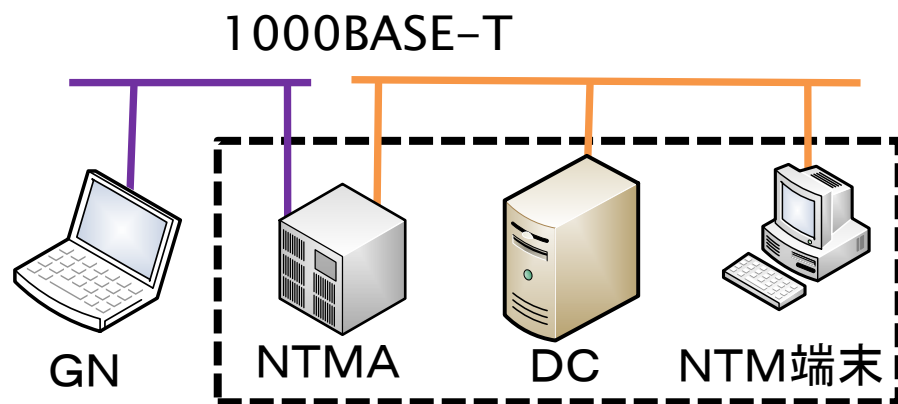
評価構成

◆ NTMAを評価



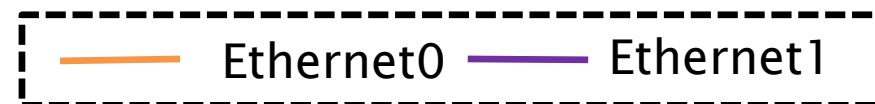
◆ 測定環境

- 1000BASE-Tの有線環境
- 仮想マシン上にNTMAを実装
- 仮想マシン上にNTM端末,DCを構築
- Windows実機のGN



Virtual Machine(Vmware Workstation 12 Player)

NTMA(Virtual Machine)	
OS	Ubuntu 14.04
CPU	Intel Corei7-930(2.80GHz)
Core	1コア
Memory	2GB



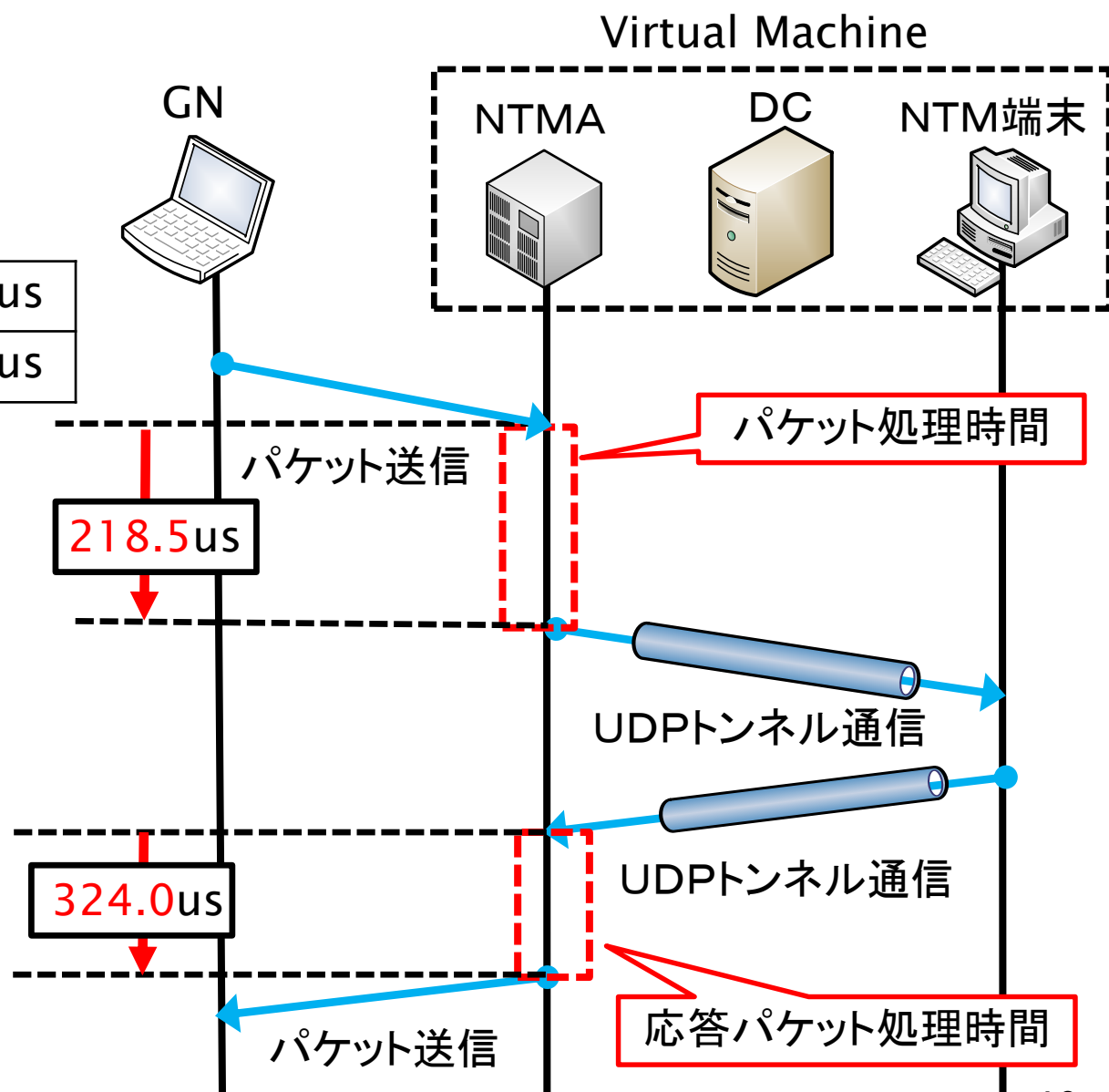
性能評価

10回の平均値

パケット処理時間	218.5us
応答パケット処理時間	324.0us

NTMAがパケット処理に
要する時間は**僅か**

※モジュール内部の処理時間



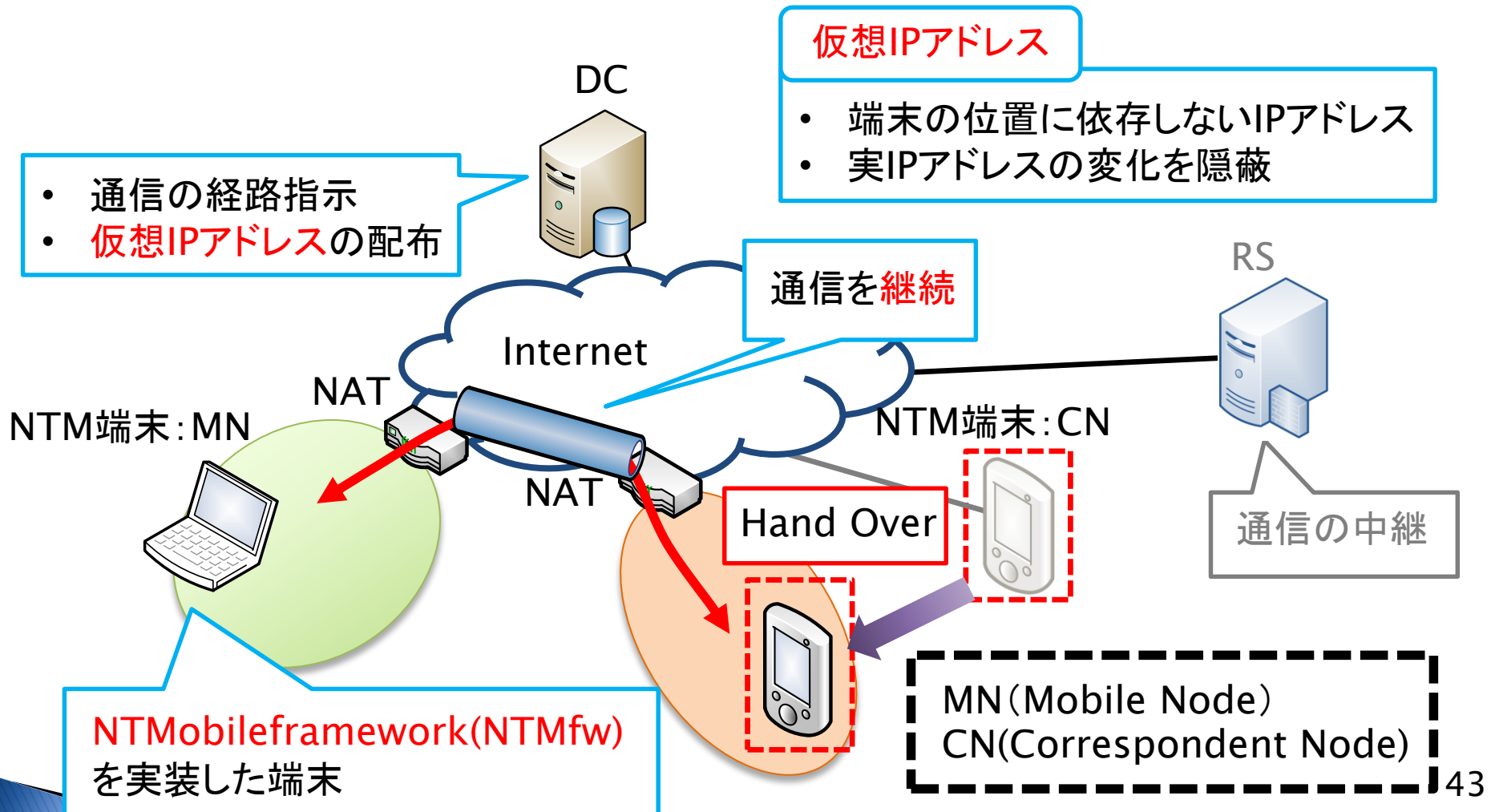
まとめ

- ◆ NTMobileアダプタの実現方式の検討
 - 一般端末にNTMAを隣接設置
 - 一般通信をNTMobile通信に変換
 - 一般端末のプログラムに手を加えない
- ◆ 今後の方針
 - NTMAの実装の完了
 - GNに割り当てるIPアドレスの検討

付録

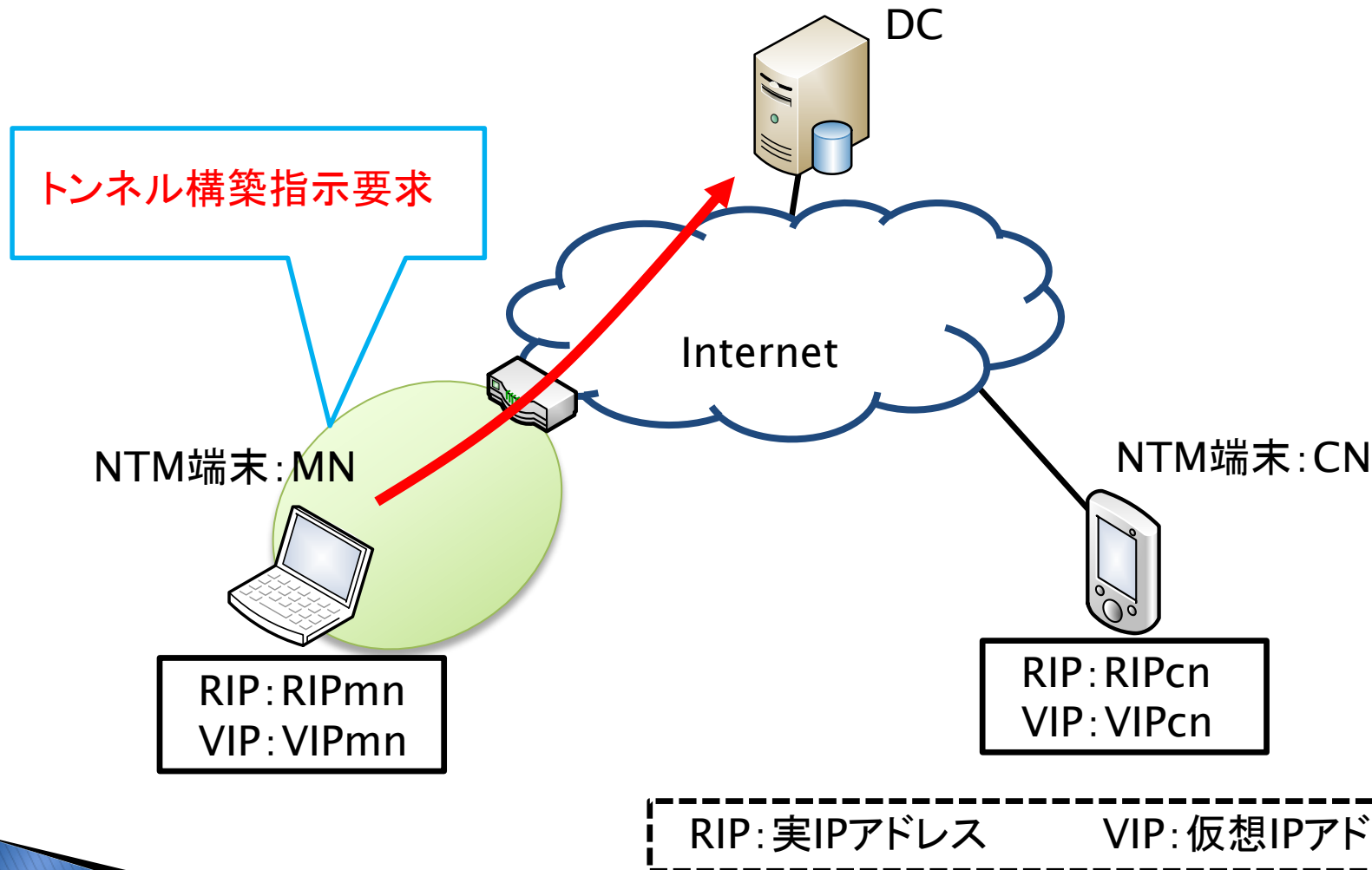
NTMobile

- 移動透過性の実現とNAT越え問題の解決を同時に実現する技術
- NTM端末, DC(Direction Coordinator), RS(Relay Server)で構成



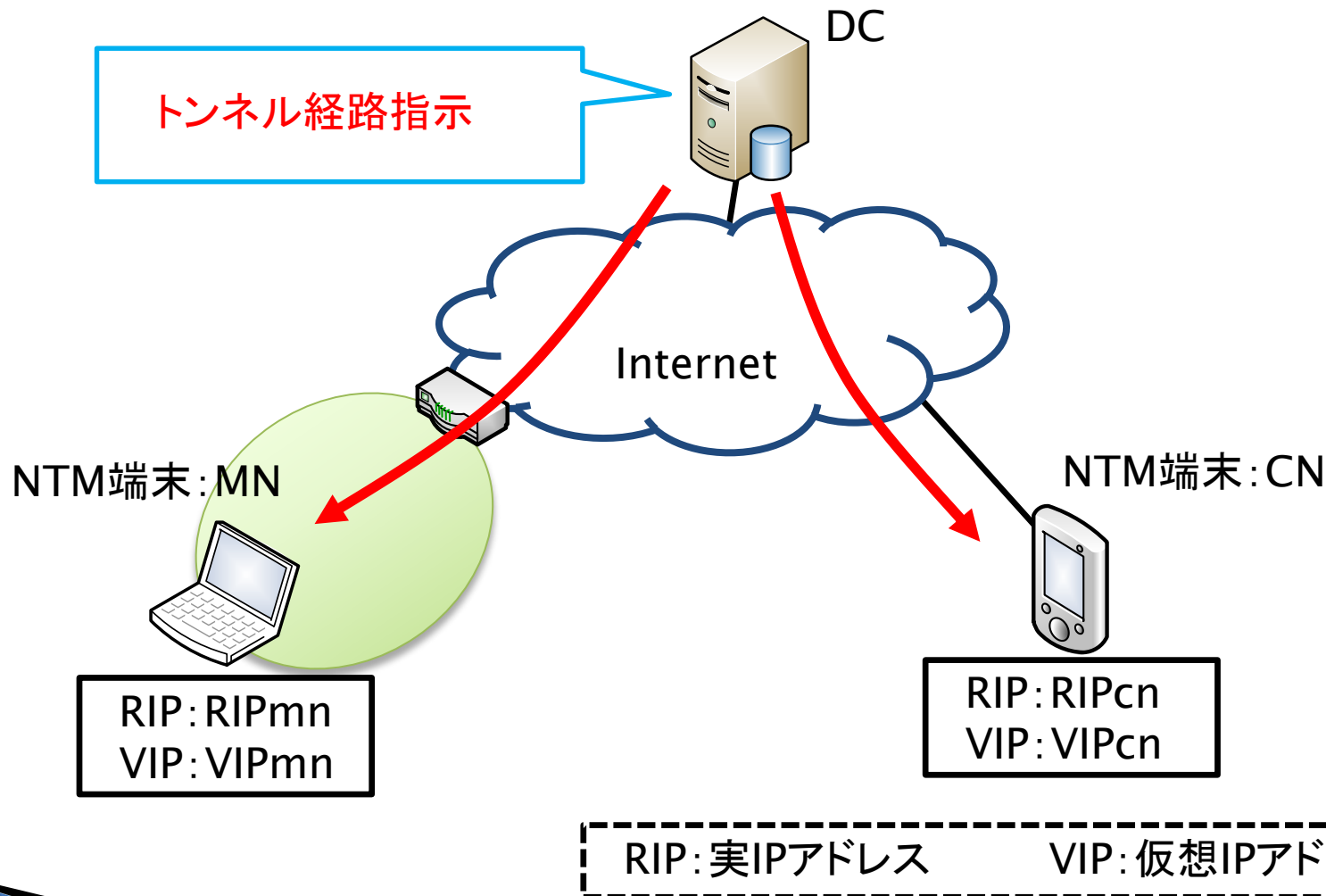
NTMobileシグナリング

- NTM端末が通信開始時に行う処理
 - 両NTM端末間でトンネル構築処理を行う



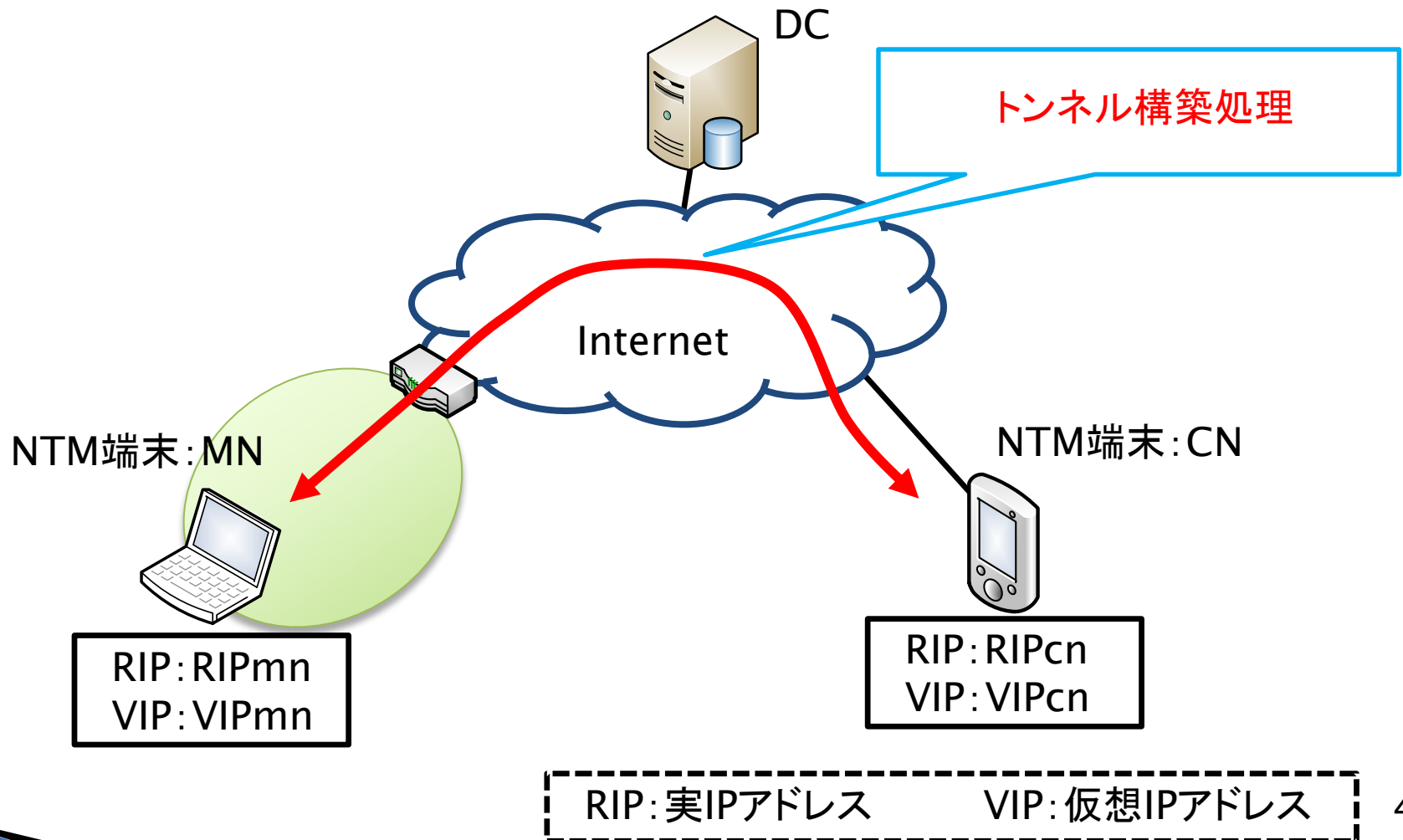
NTMobileシグナリング

- NTM端末が通信開始時に行う処理
 - 両NTM端末間でトンネル構築処理を行う



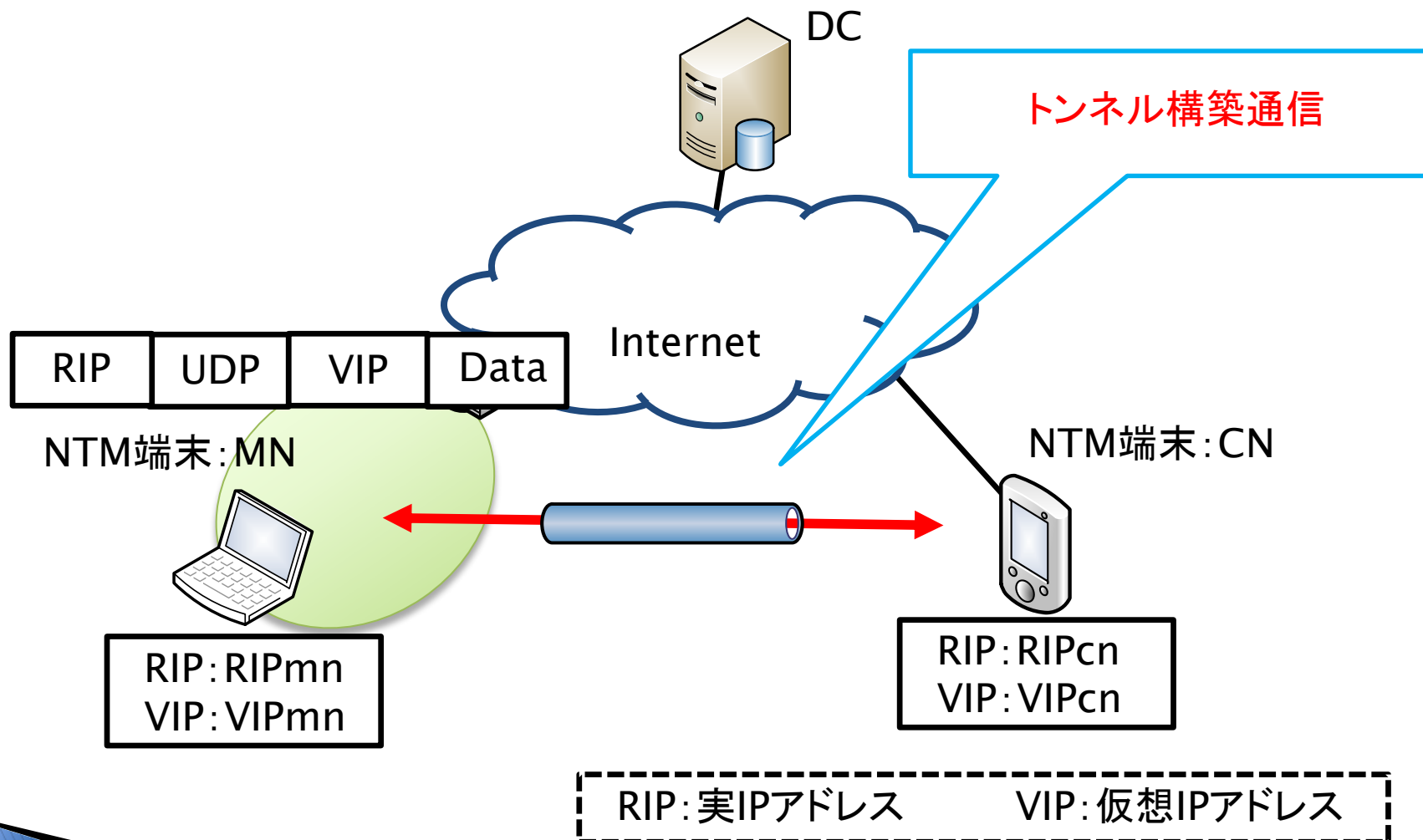
NTMobileシグナリング

- NTM端末が通信開始時に行う処理
 - 両NTM端末間でトンネル構築処理を行う

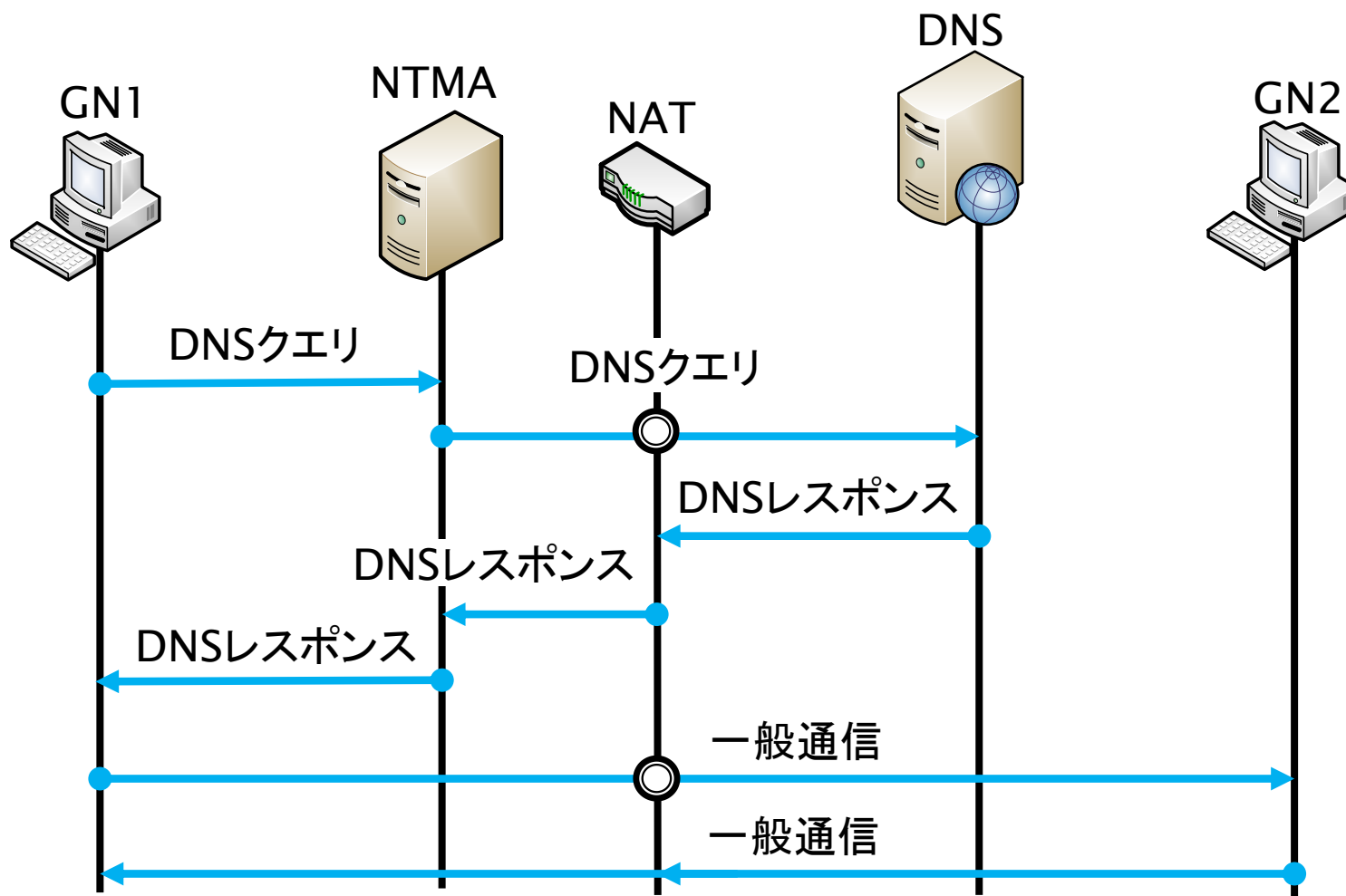


NTMobileの実際の通信

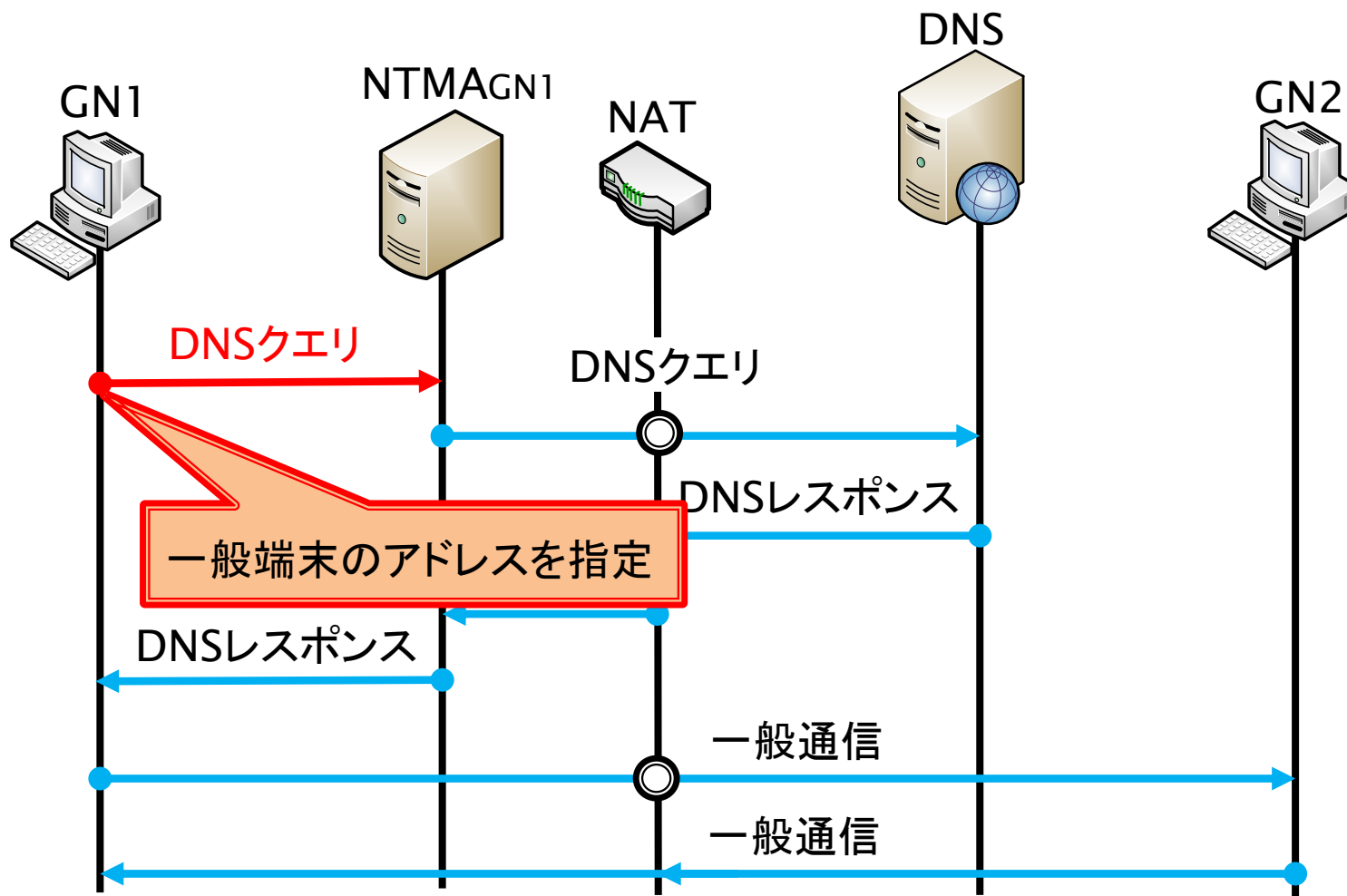
- UDPTunnel通信
 - 実IPパケットで仮想IPパケットをカプセル化



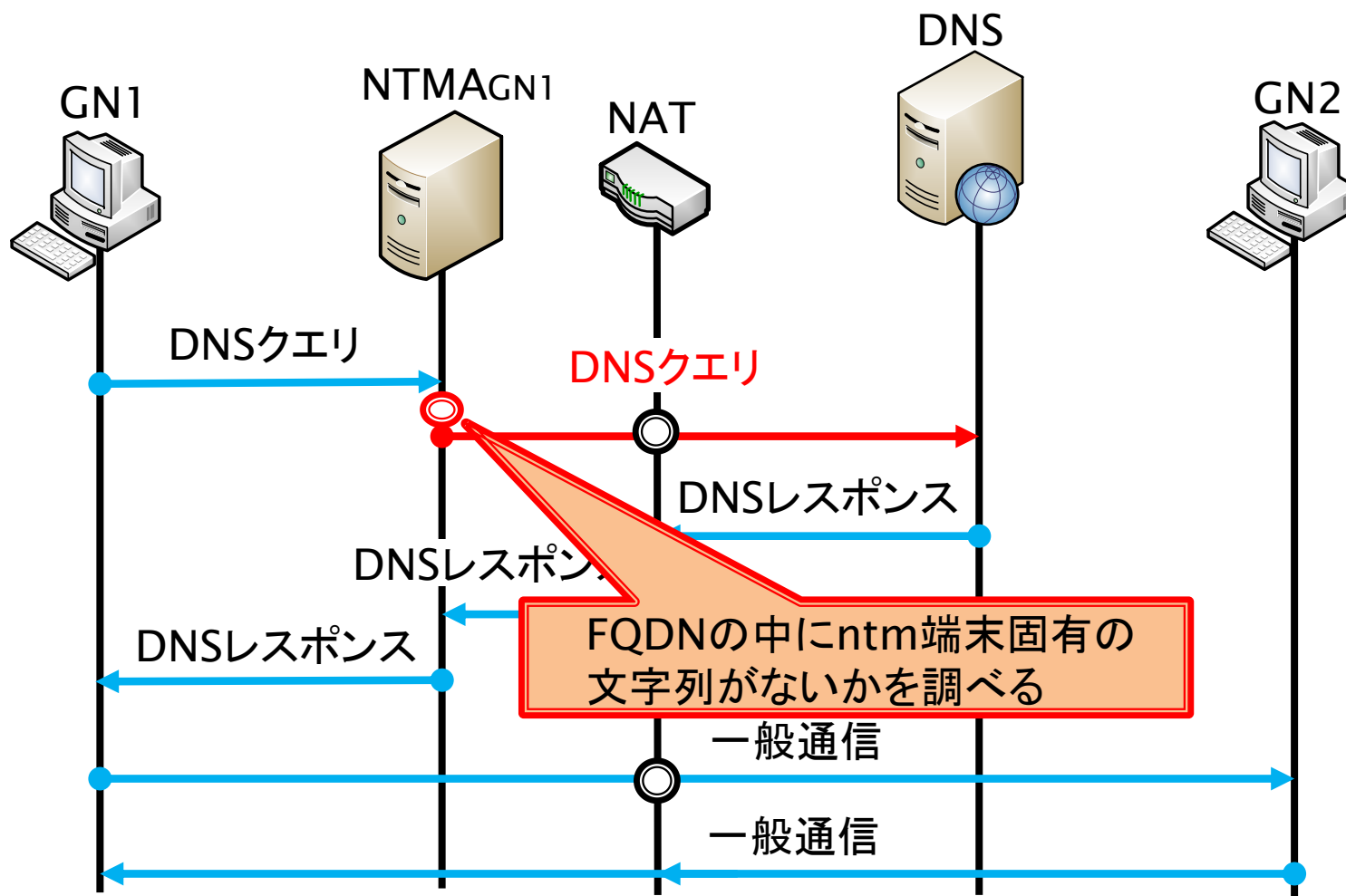
提案方式 -動作シーケンス-



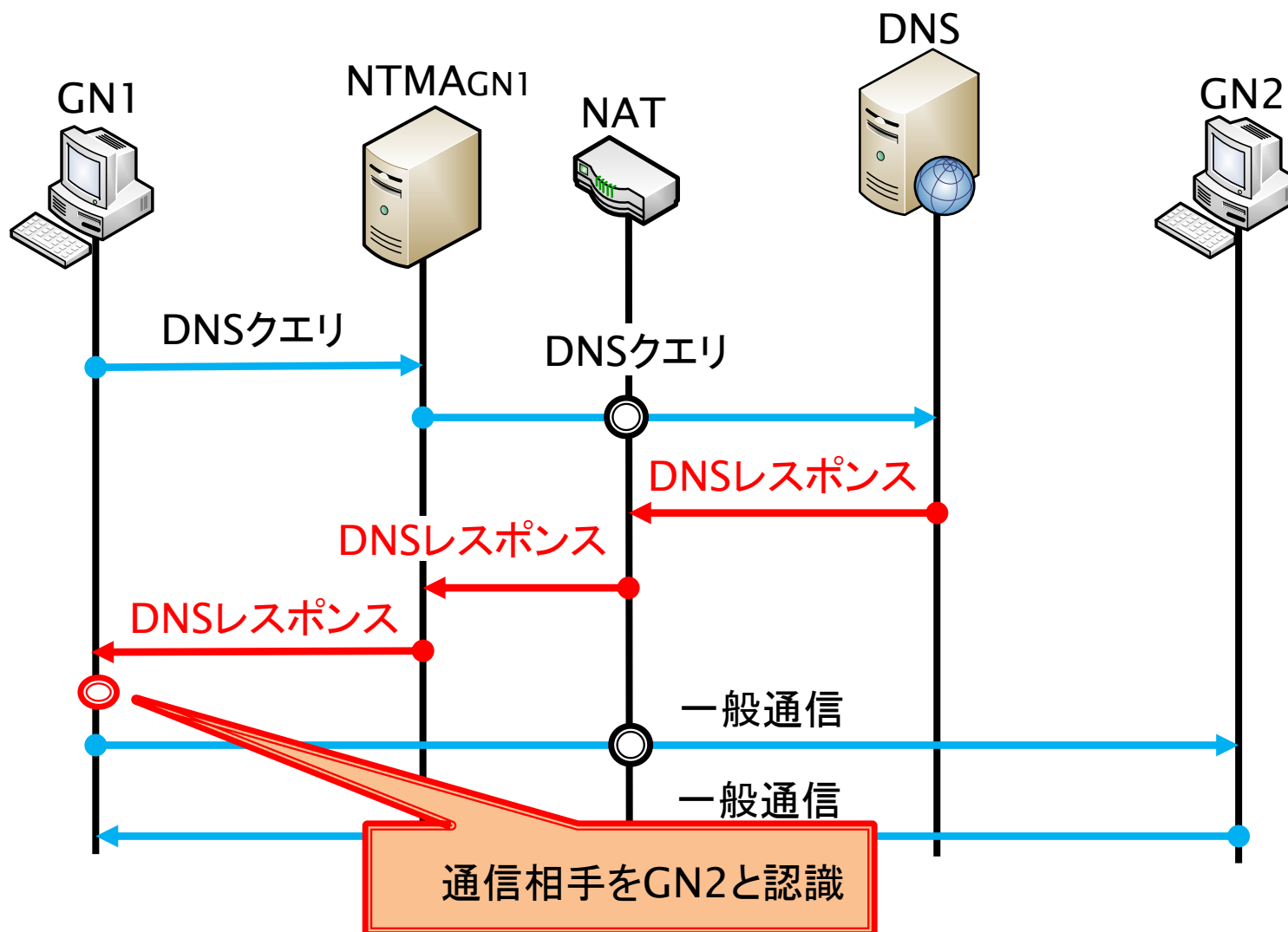
提案方式 -動作シーケンス-



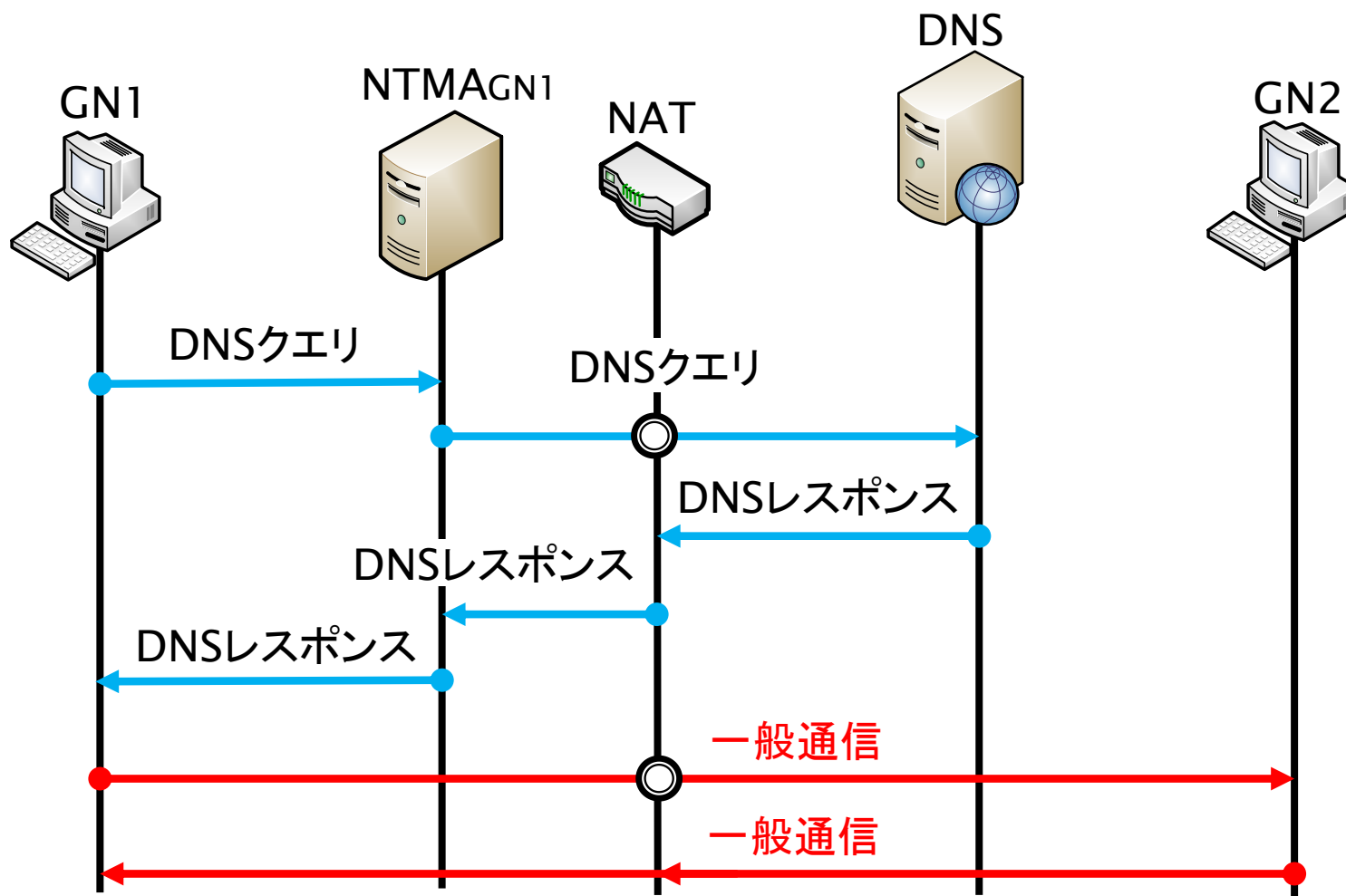
提案方式 -動作シーケンス-



提案方式 -動作シーケンス-

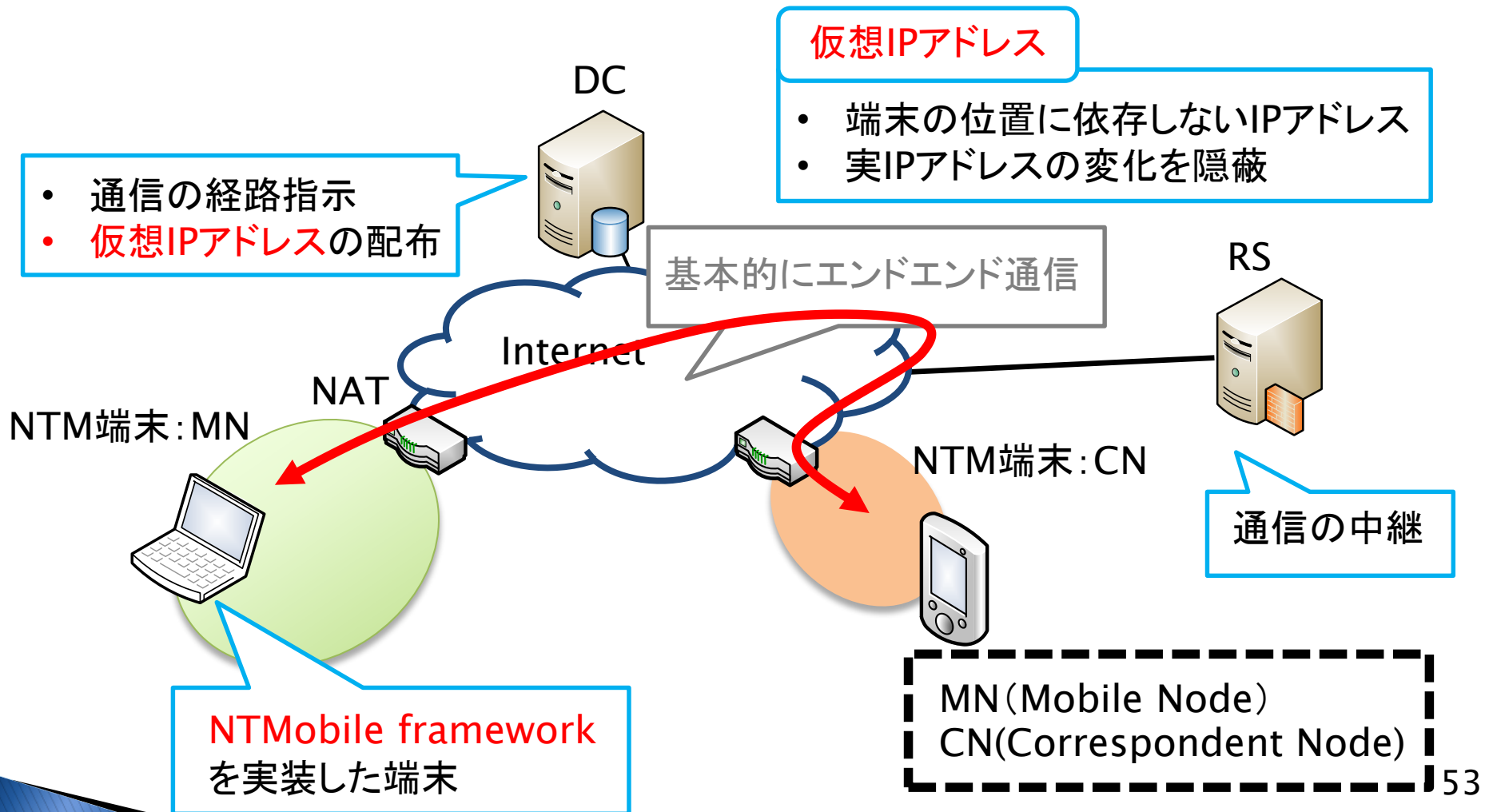


提案方式 -動作シーケンス-

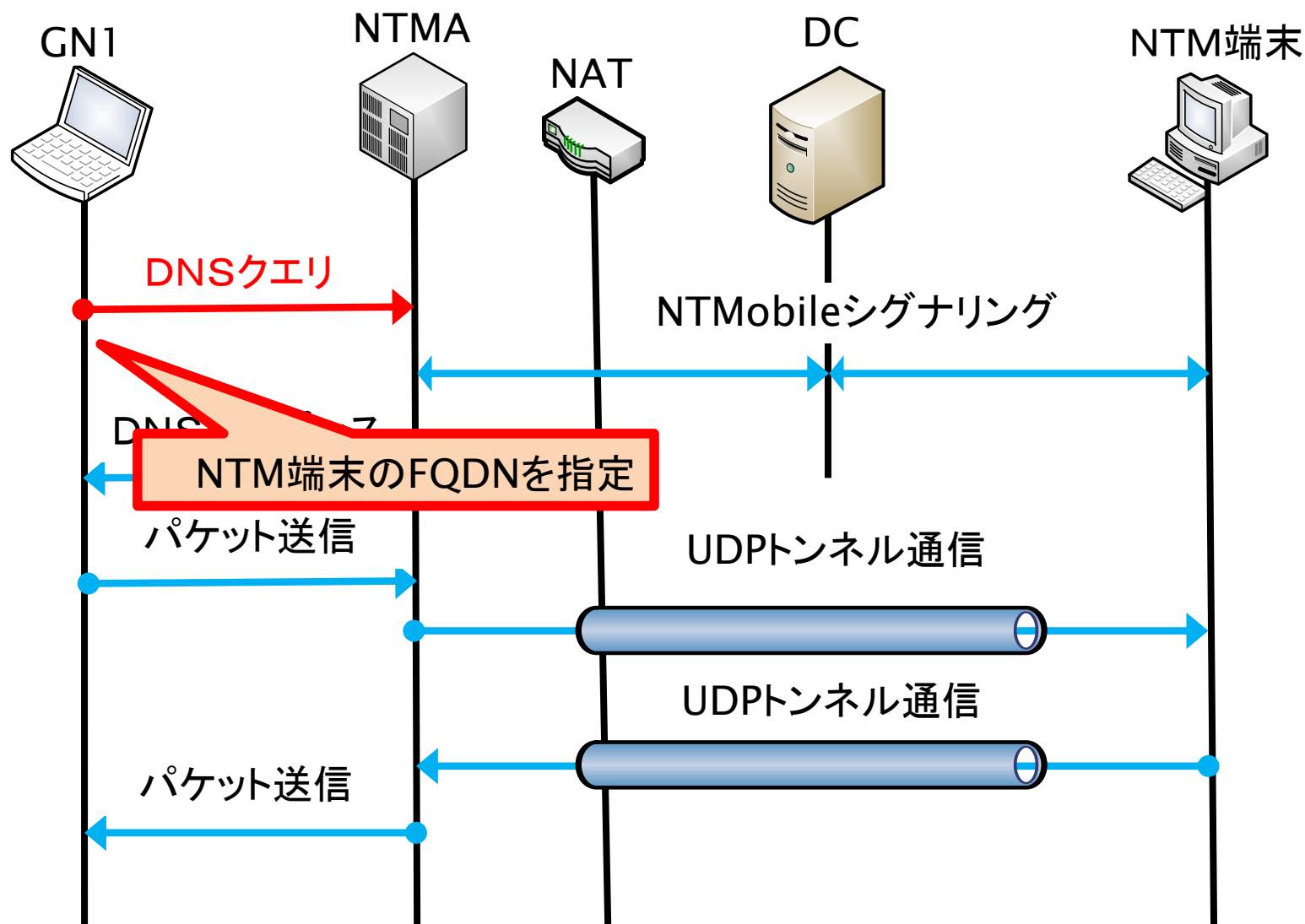


NTMobile(Network Traversal with Mobility)

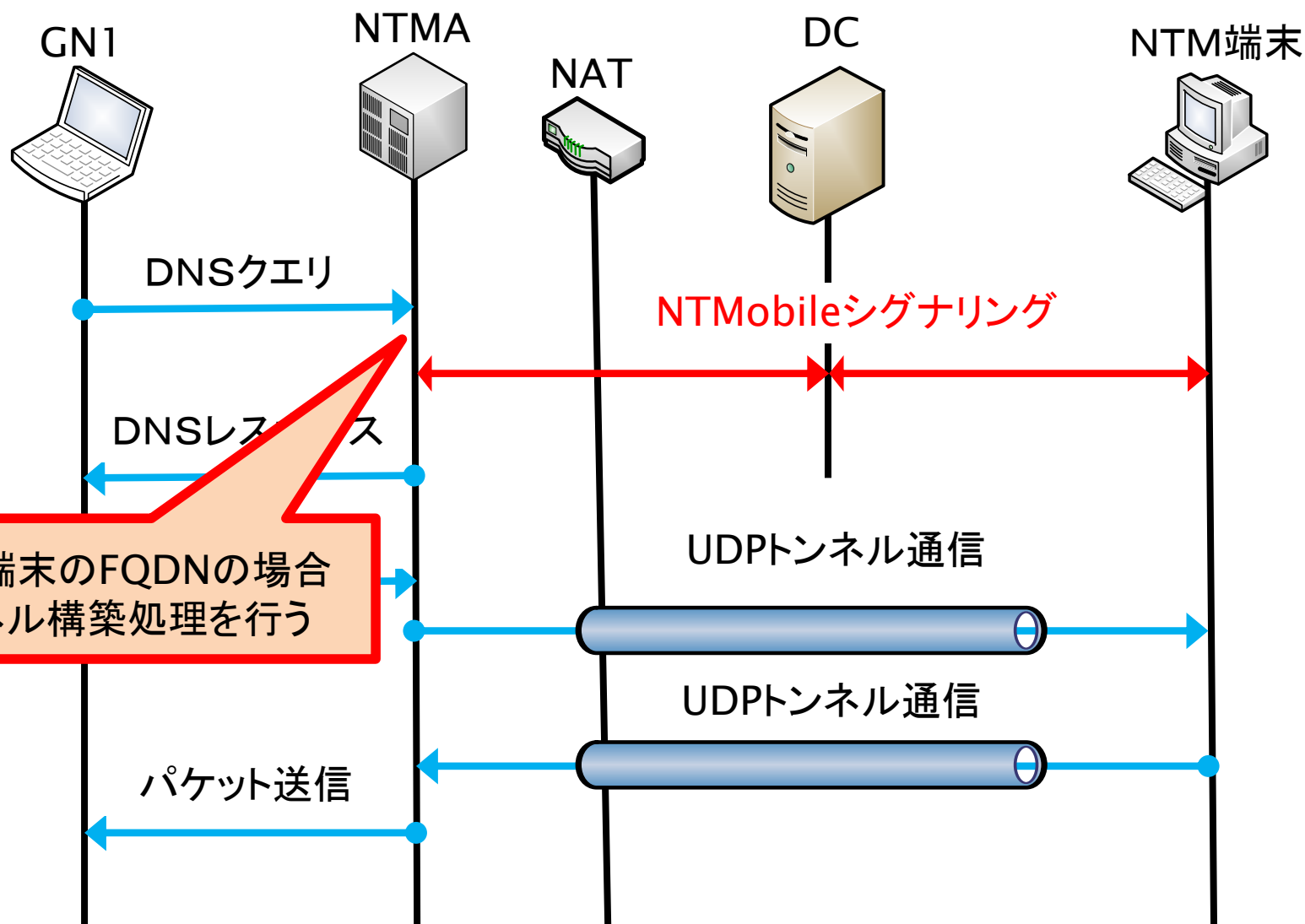
- NAT越え問題を解決する技術
- NTM端末, DC(Direction Coordinator), RS(Relay Server)で構成



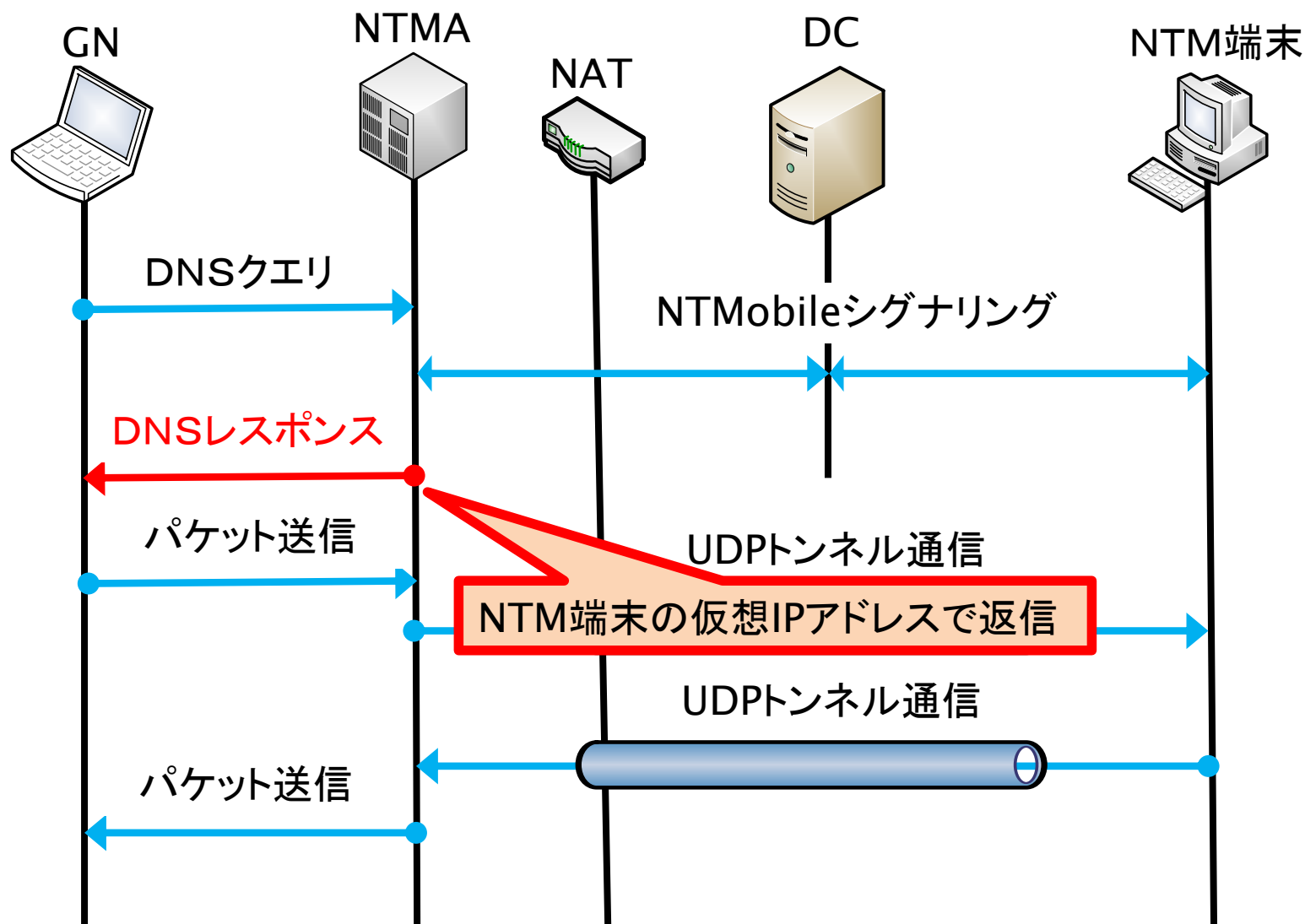
提案方式 -動作シーケンス-



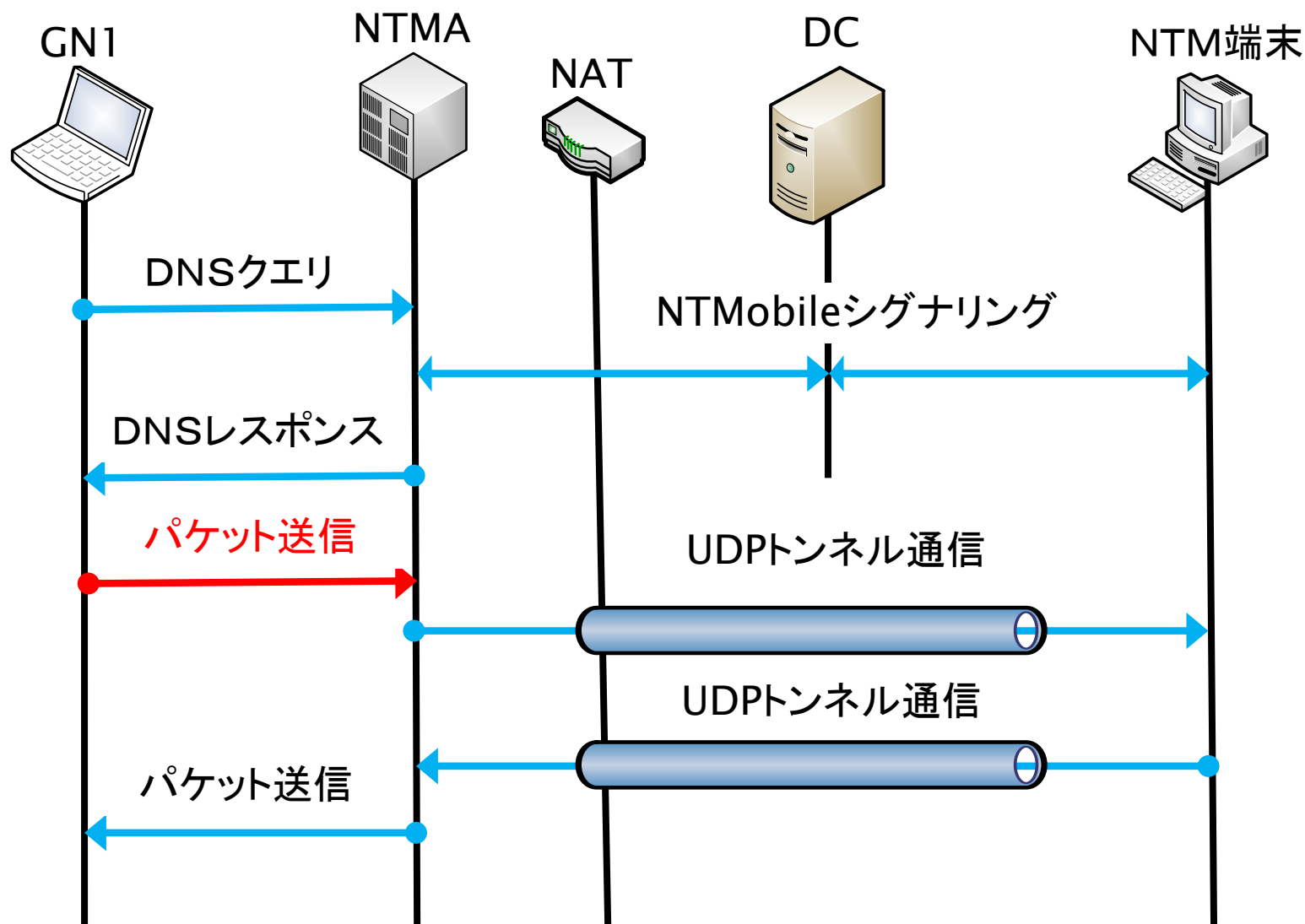
提案方式 -動作シーケンス-



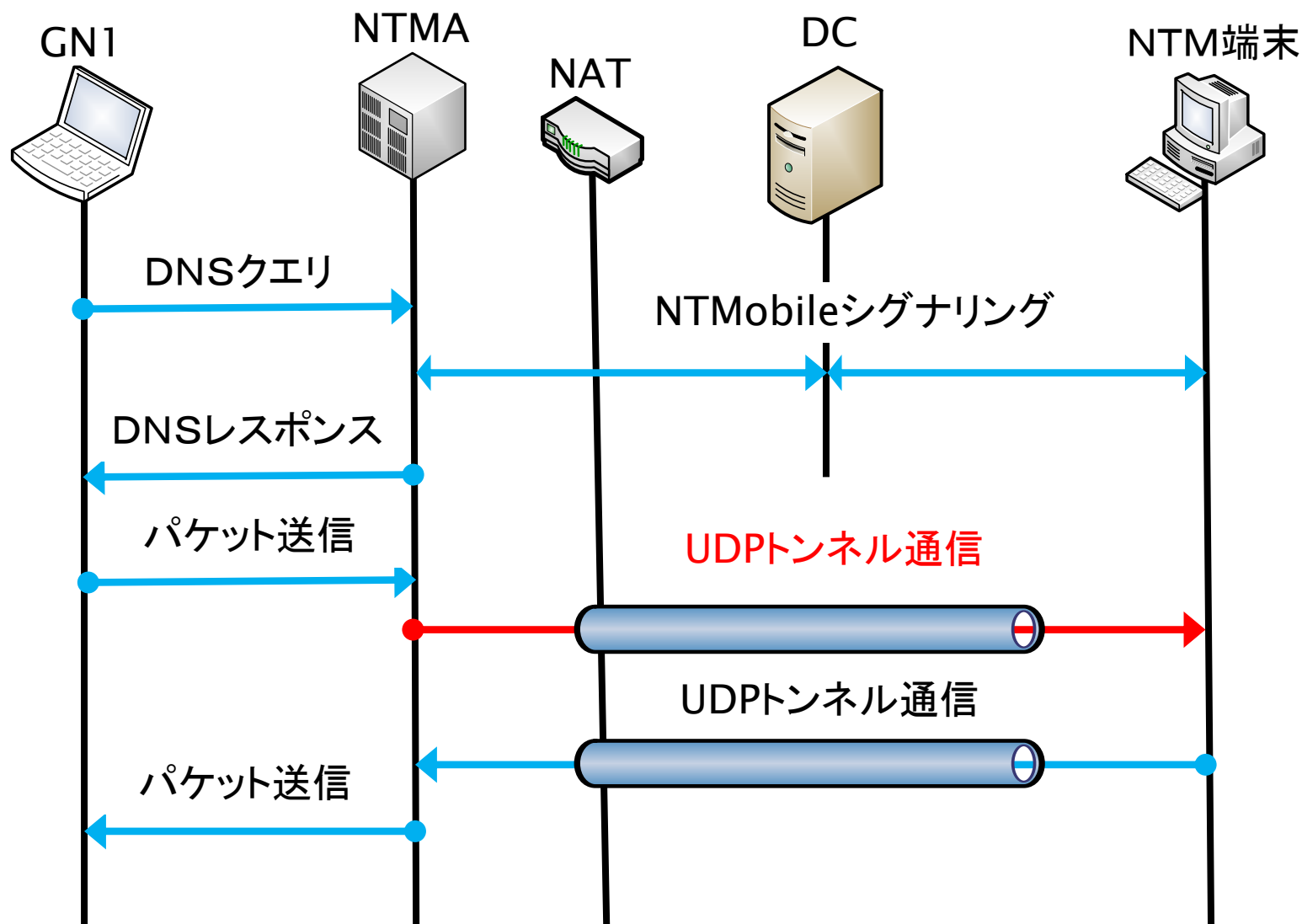
提案方式 -動作シーケンス-



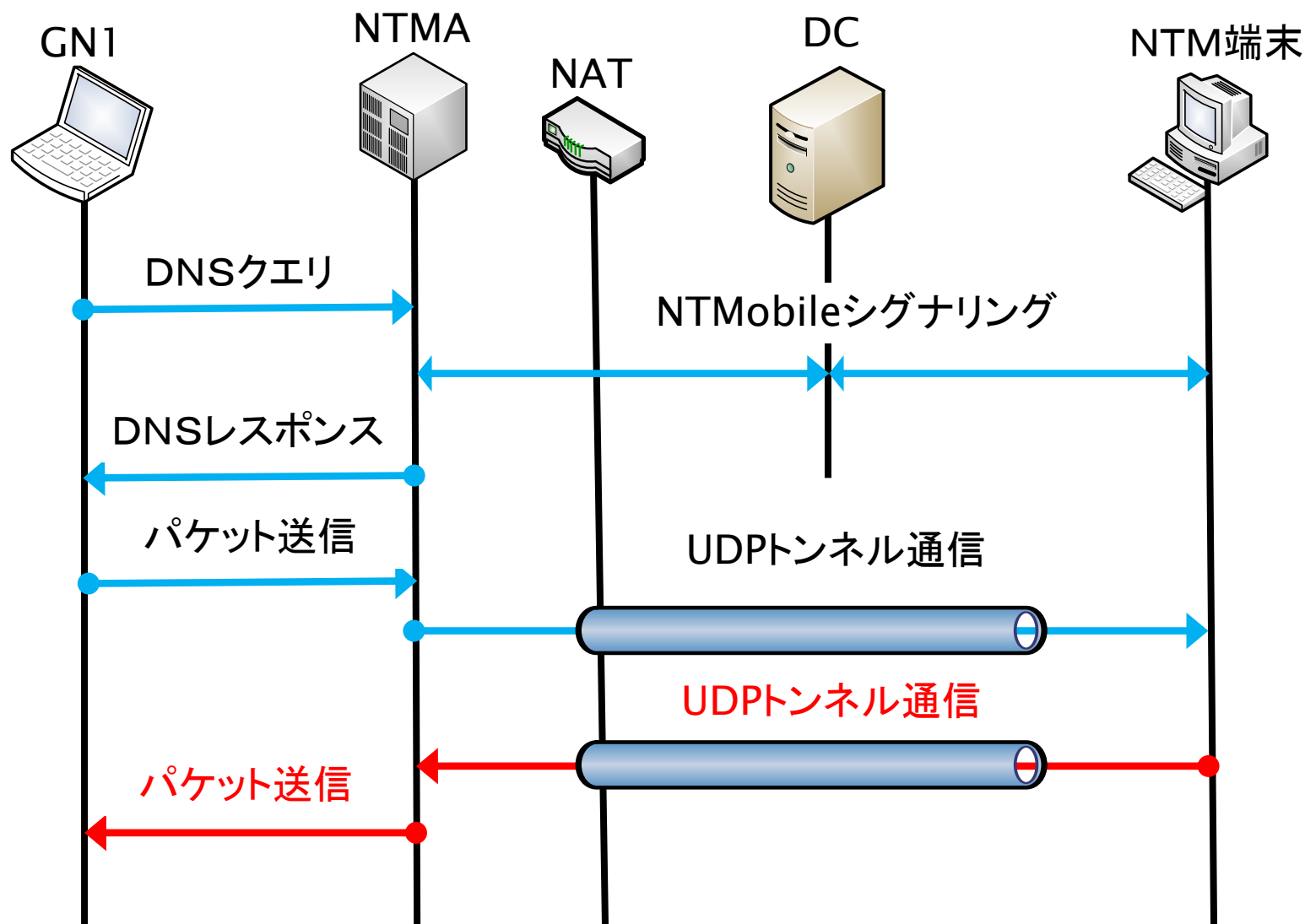
提案方式 -動作シーケンス-



提案方式 -動作シーケンス-



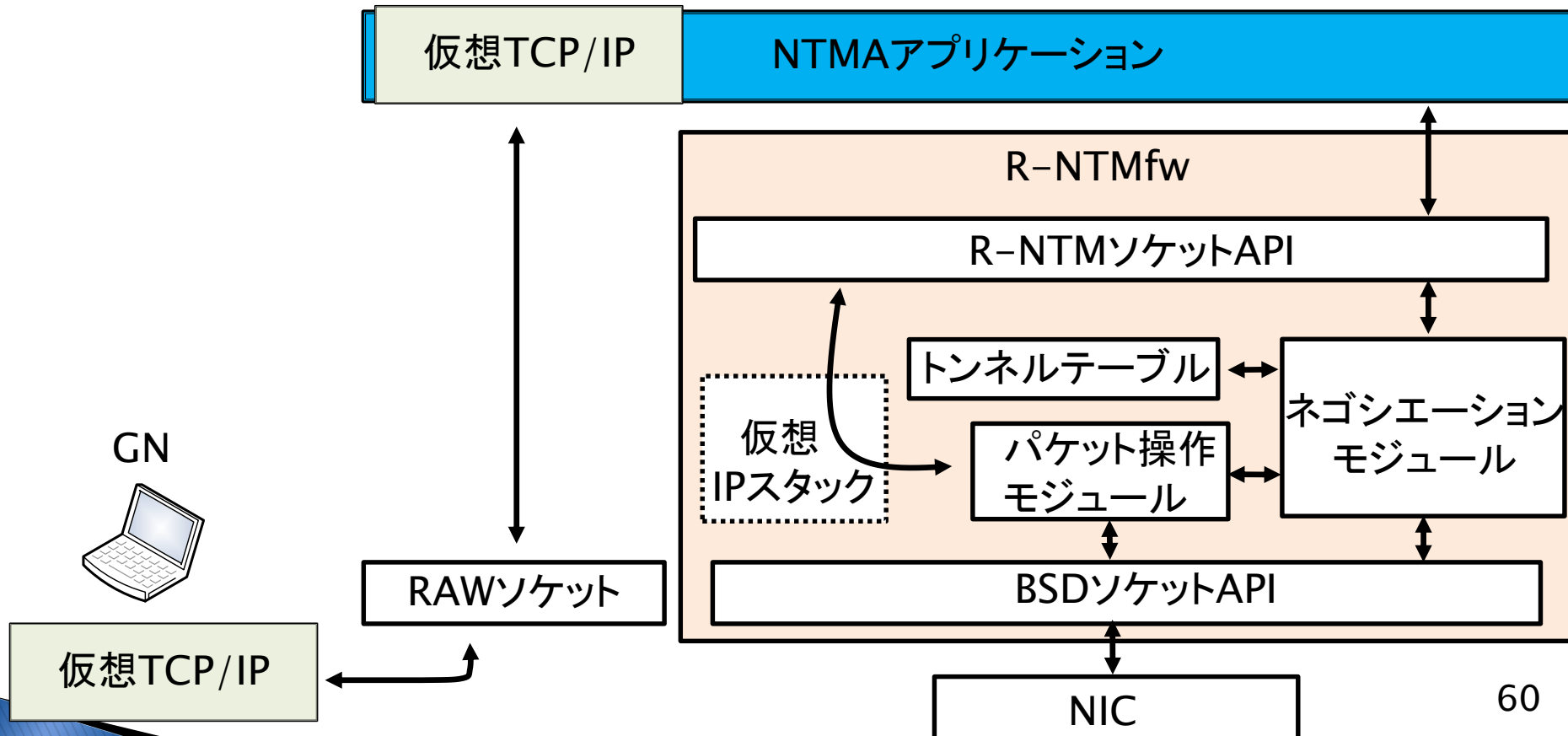
提案方式 -動作シーケンス-



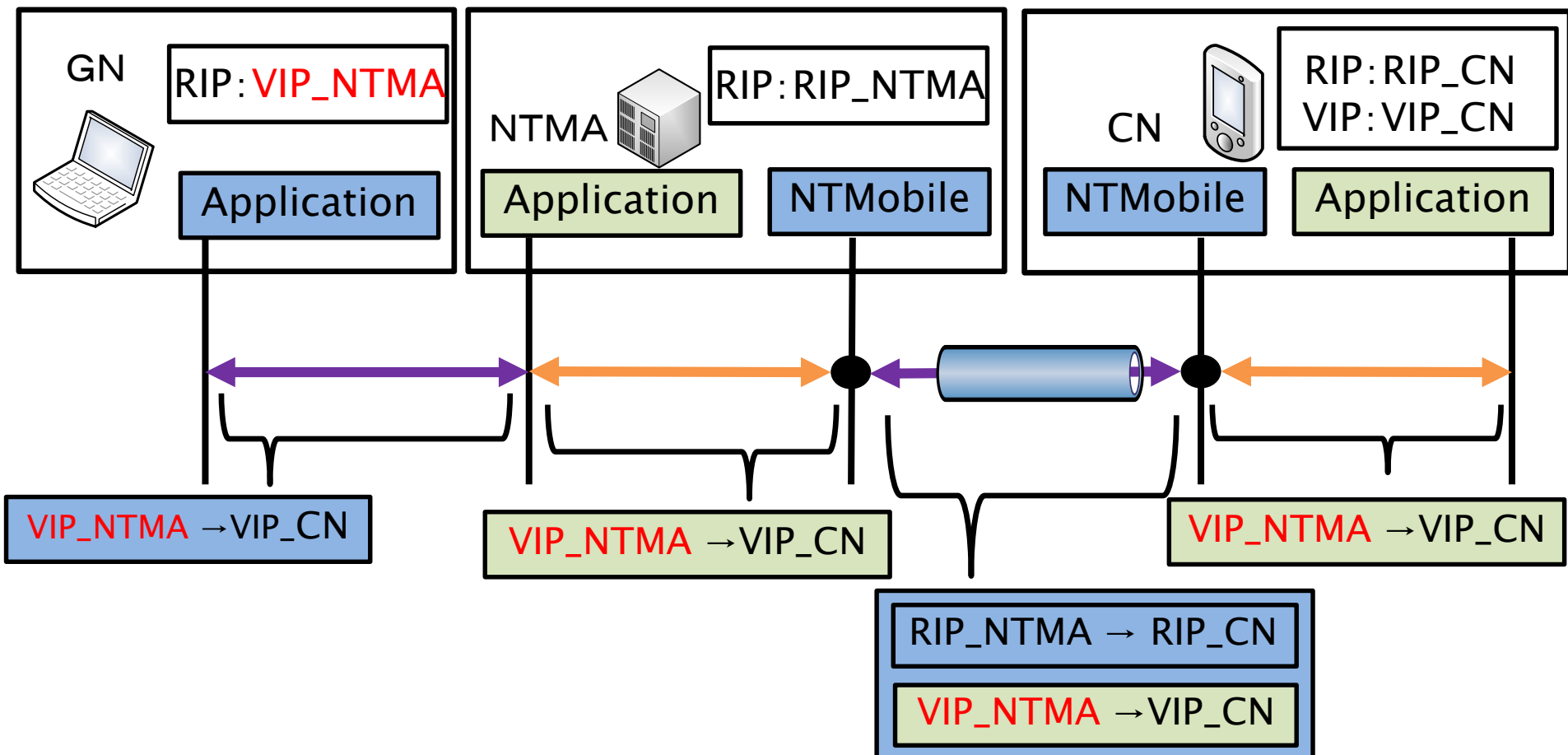
NTMAのモジュール構成

NTMAアプリケーション

- GNとの通信は, RAWソケットで送受信
- NTM端末との通信は, R-NTMfw(Remodeled NTMfw)のAPIで送受信

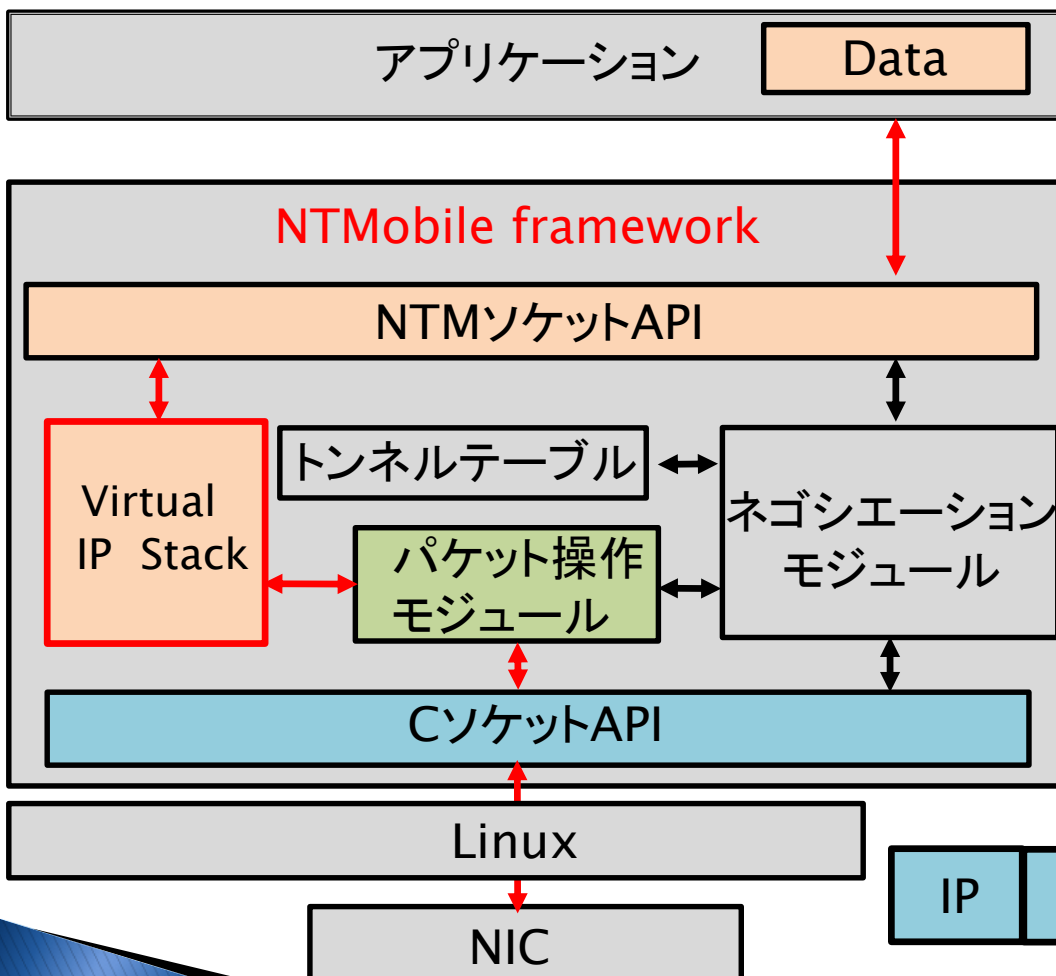


提案方式 (通信時)



NTMfw (NTMobile framework)

- ◆ アプリケーション層で動作する通信ライブラリ
 - 上位アプリケーションにNTMobile機能を提供



● NTMソケットAPI

- 上位アプリケーションにNTMソケットAPIを提供

ntmfw_sedto()..etc

● Virtual IP Stack

- 仮想IPパケットの生成



● パケット操作モジュール

- 暗号化・MAC付与



● CソケットAPI

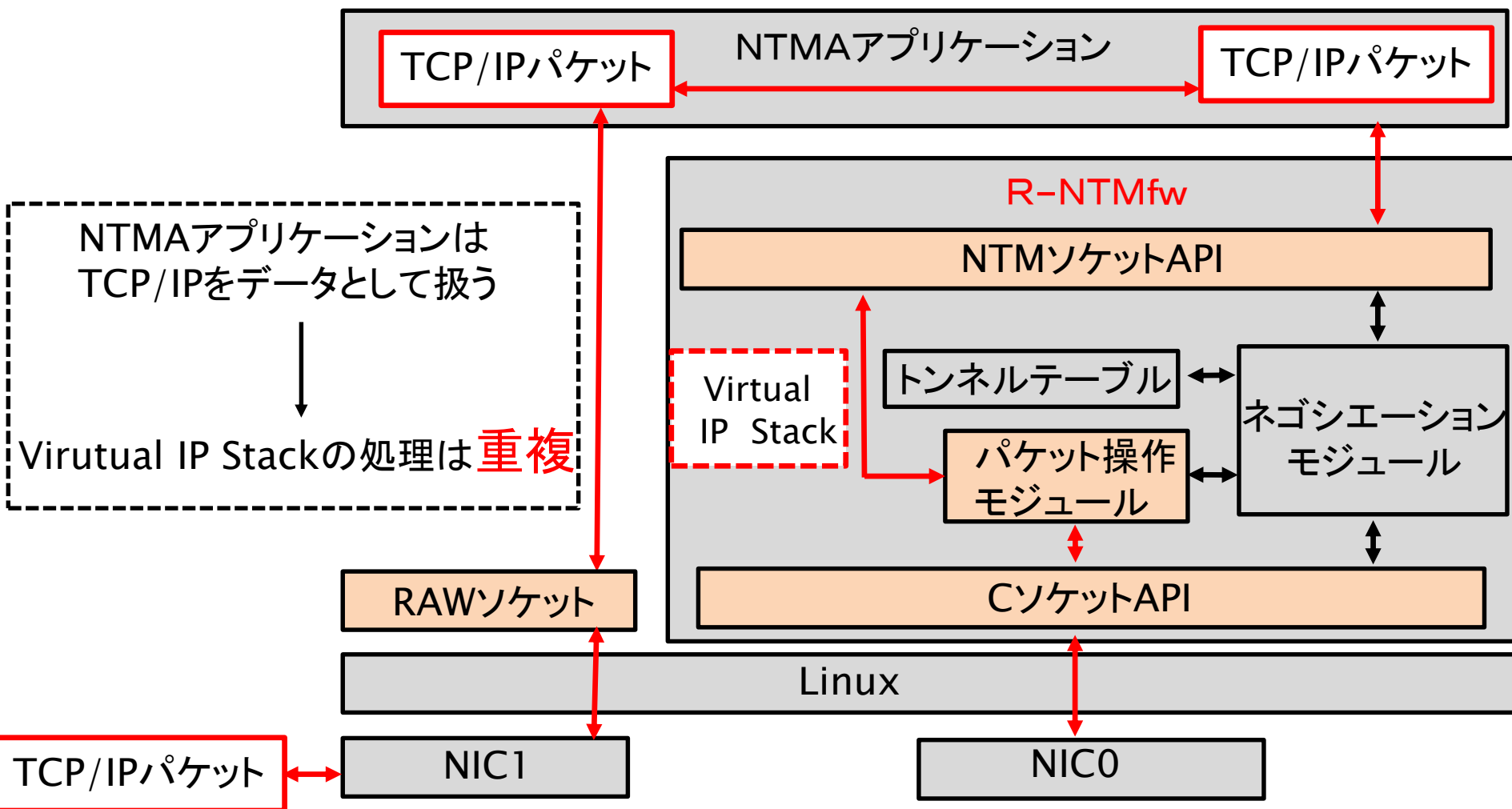
- Linuxが提供するCソケット



R-NTMfw(Remodeled NTMfw)

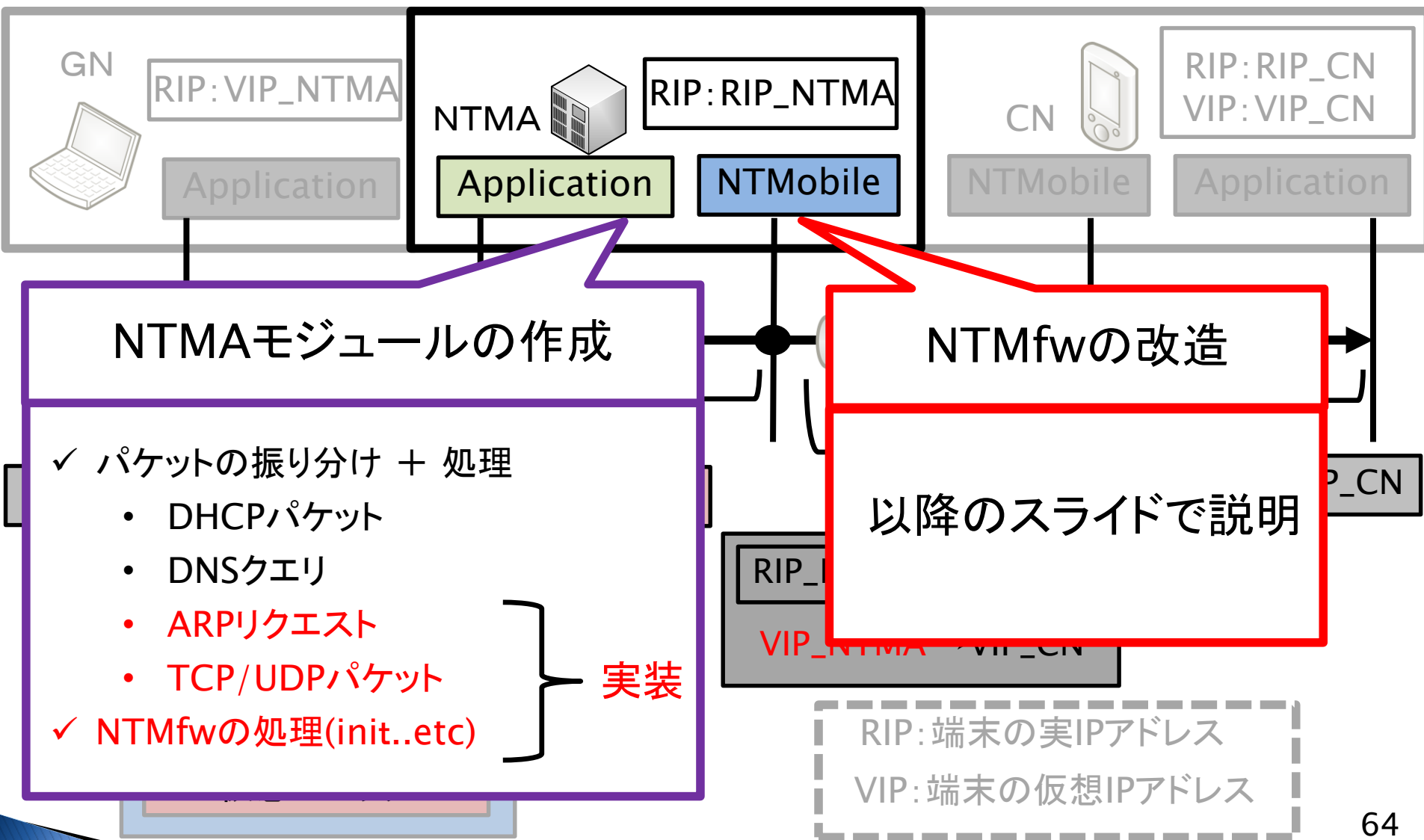
◆ R-NTMfw

➤ NTMfwを改造したアプリケーション



※RAWソケット 生のパケットをダイレクトに送受信するソケット

提案方式 (通信時)



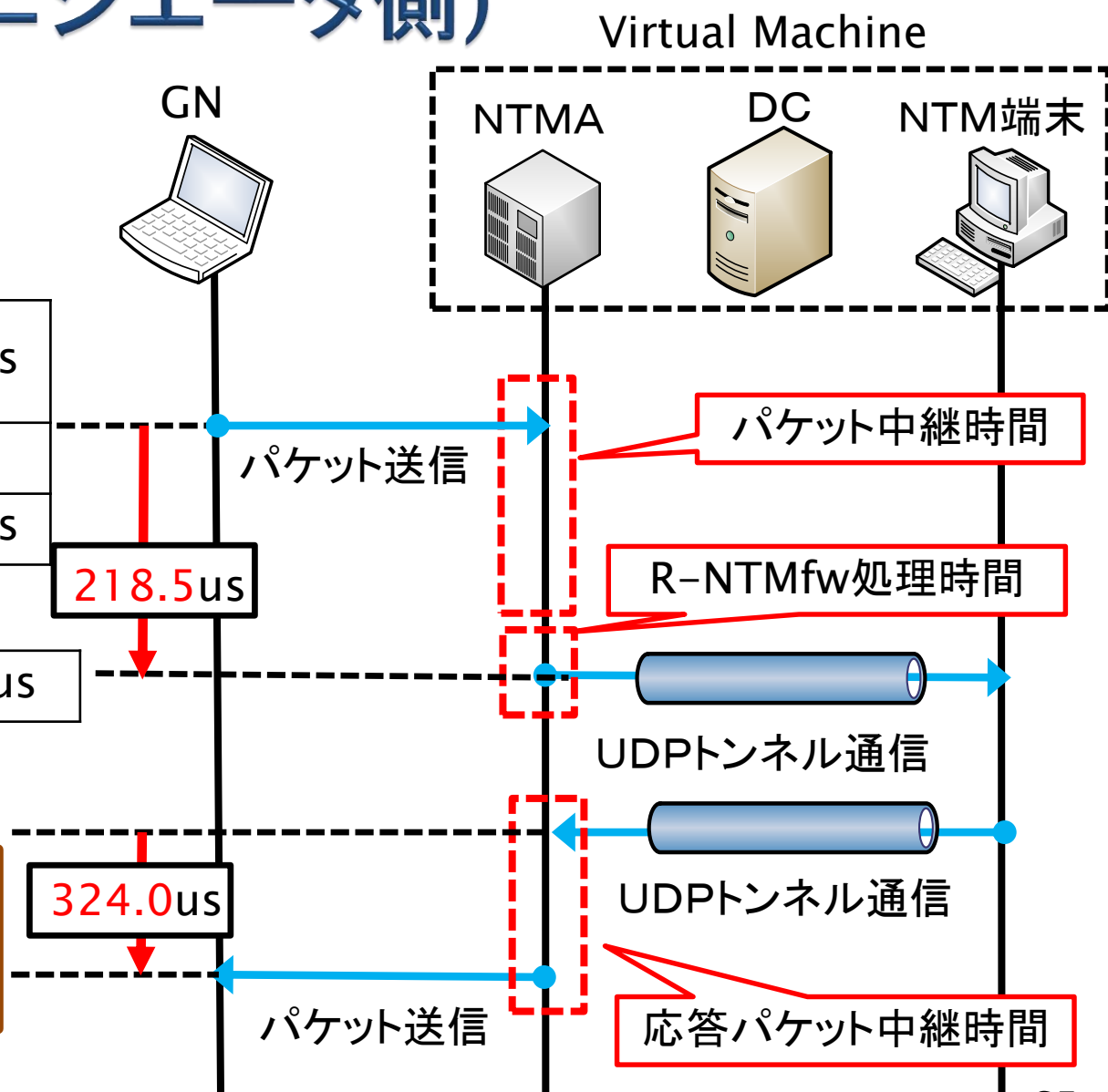
性能評価（イニシエータ側）

10回の平均値

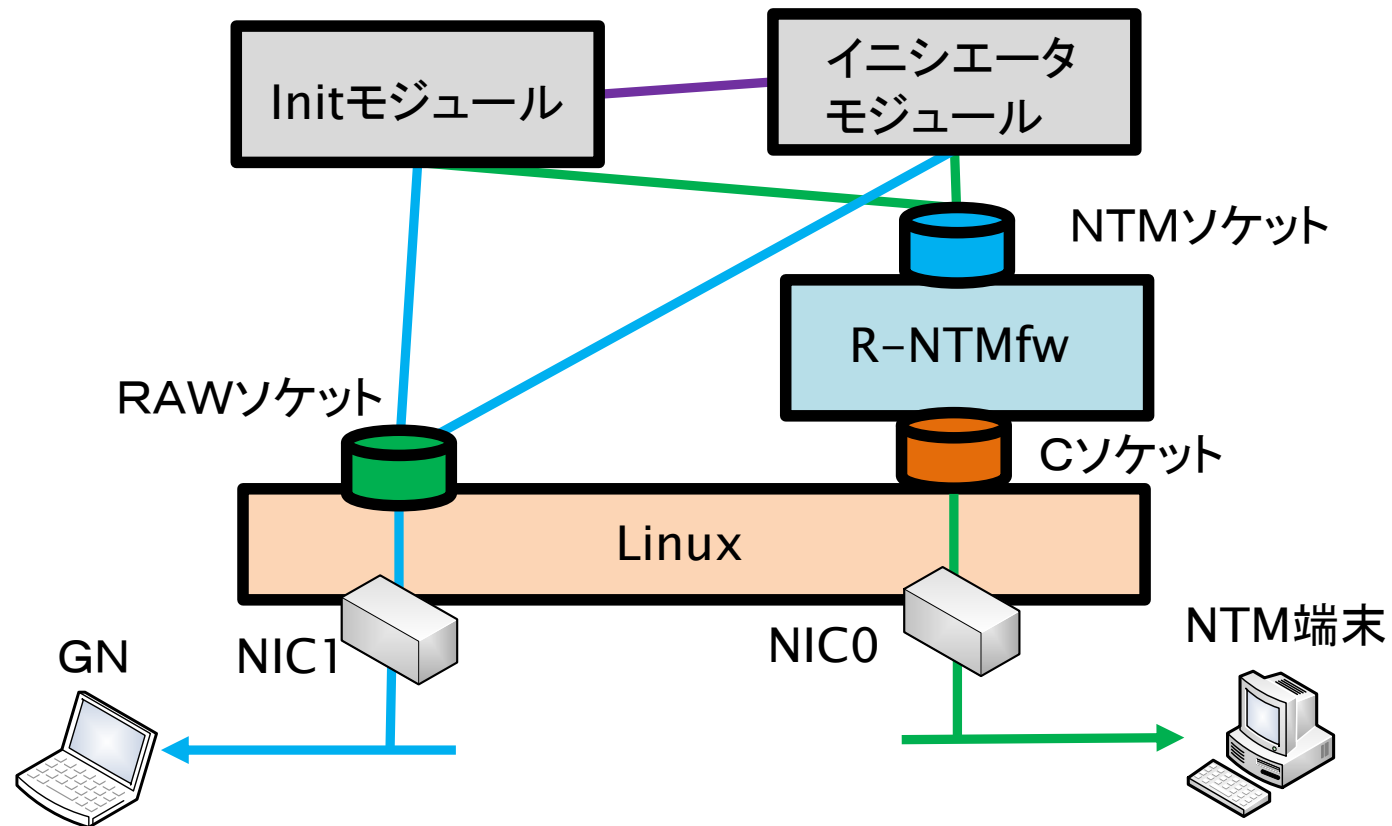
パケット中継に 要する時間	218.5us
パケット中継時間	12.0us
R-NTMfw処理時間	206.5us

応答パケット中継時間	324.0us
------------	---------

NTMAがパケット中継に
要する時間は**僅か**

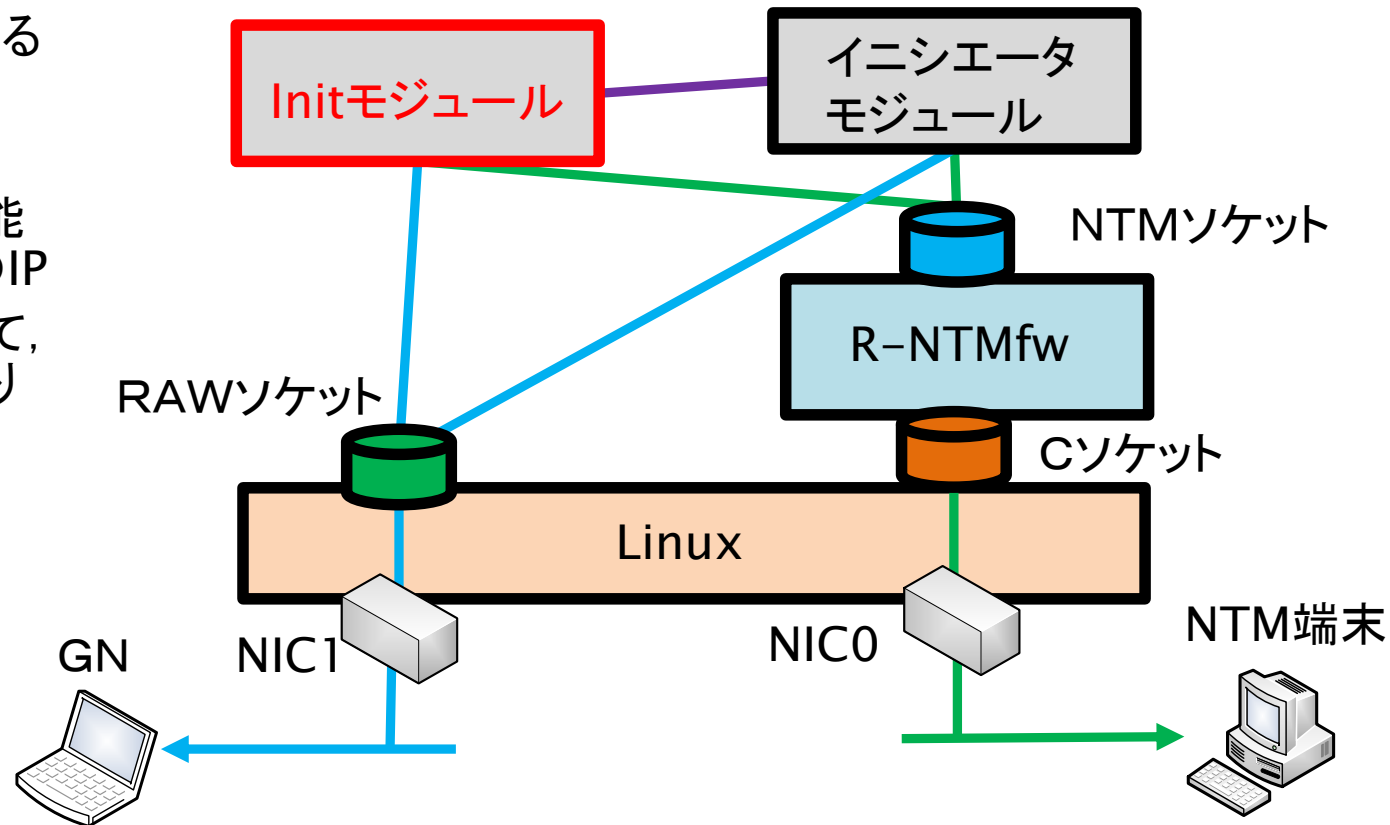


NTMAのモジュール構成



NTMAのモジュール構成

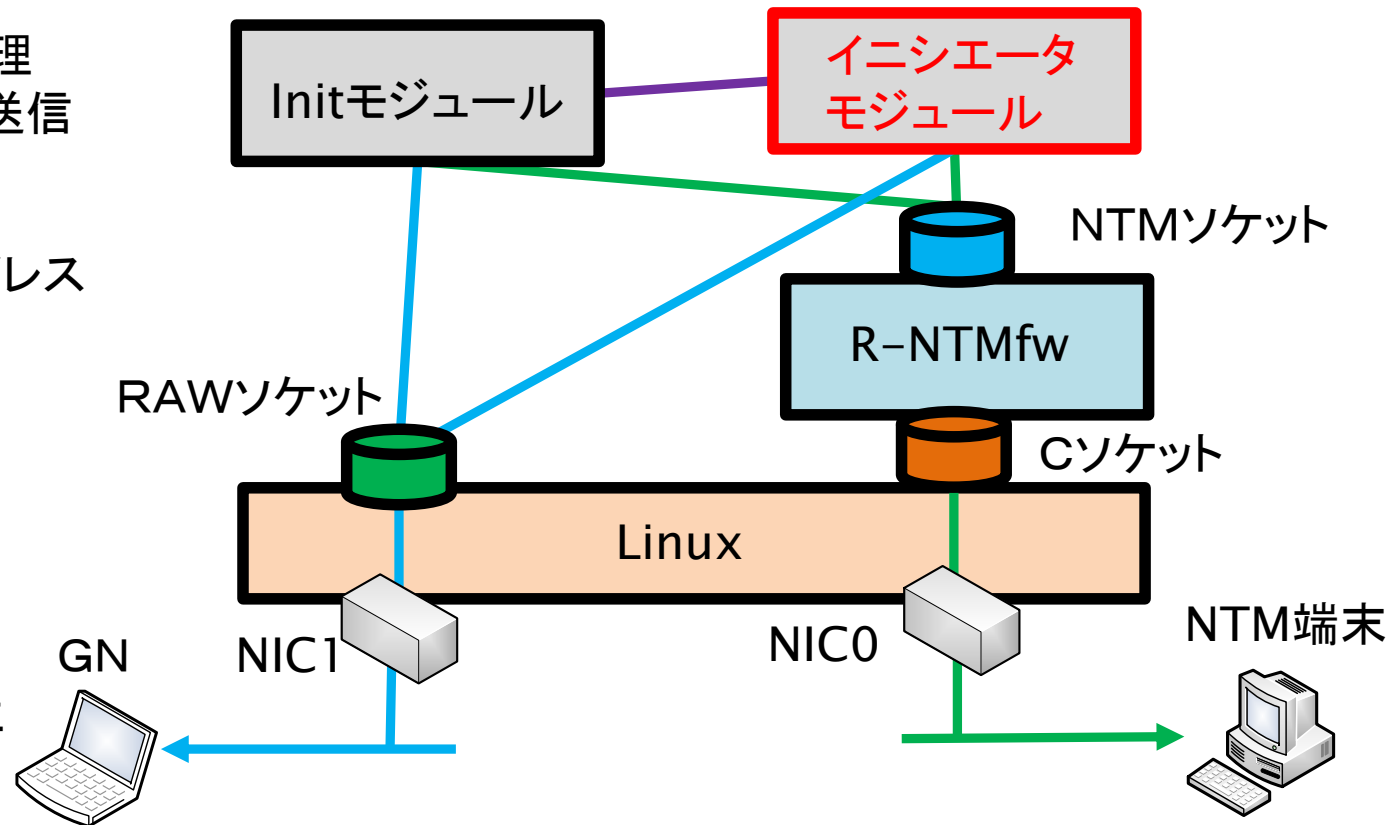
- DCから仮想IPアドレスの割り当てを受ける
- DHCPサーバの機能を包含し、GNからのIPアドレス要求に対して、仮想IPアドレスを割り当てる.



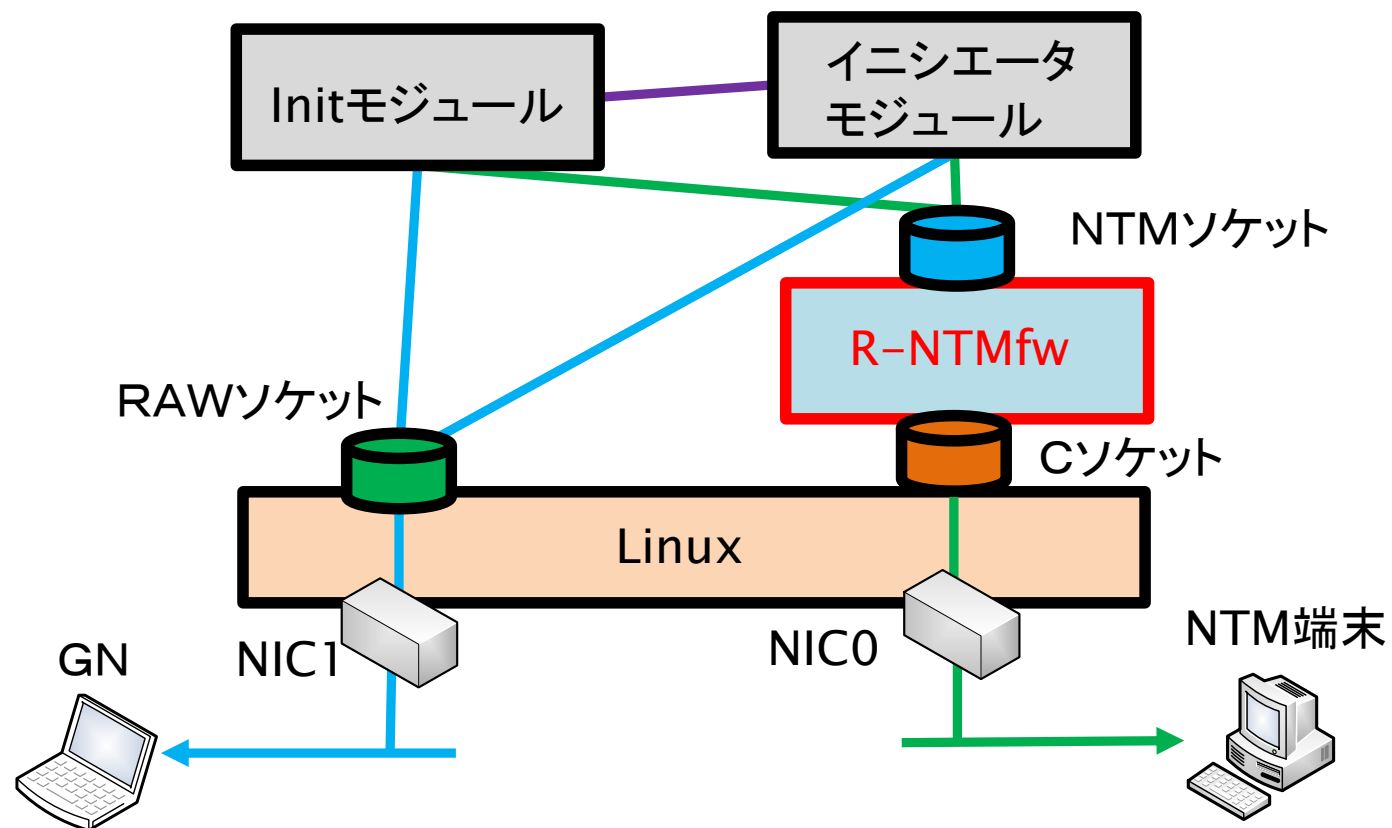
NTMAのモジュール構成

GNから送信されるパケットを
RAWソケットで受信

- DNSクエリ
 - トンネル構築処理
 - DNSリプライの送信
- APR
 - 自身のMACアドレス
でARPリプライ
- UDP/TCPパケット
 - トンネル通信で
パケットを送信
- DHCP
 - Initモジュールに
処理を渡す



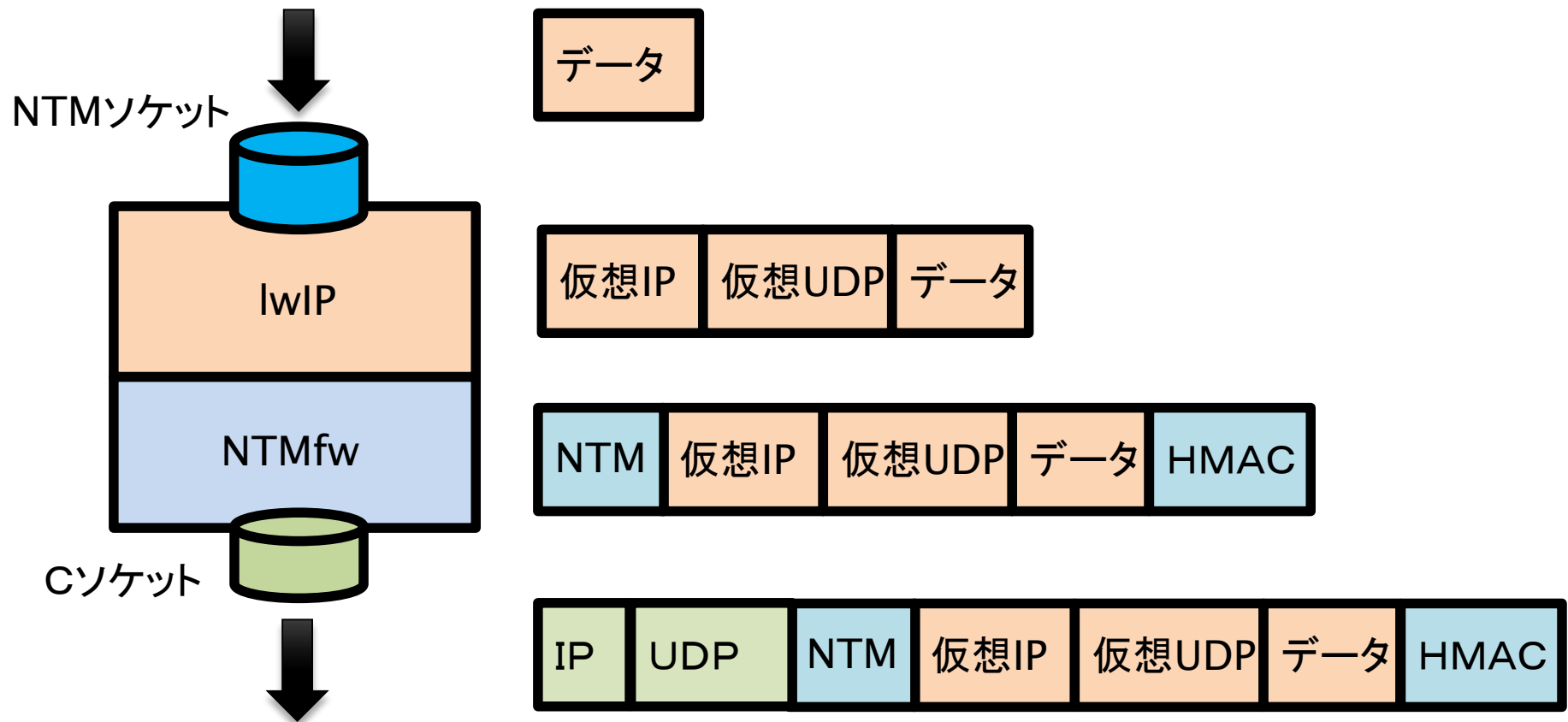
NTMAのモジュール構成



R-NTMfw(Remodeled NTMfw)

■ NTMfwを改造したアプリケーション

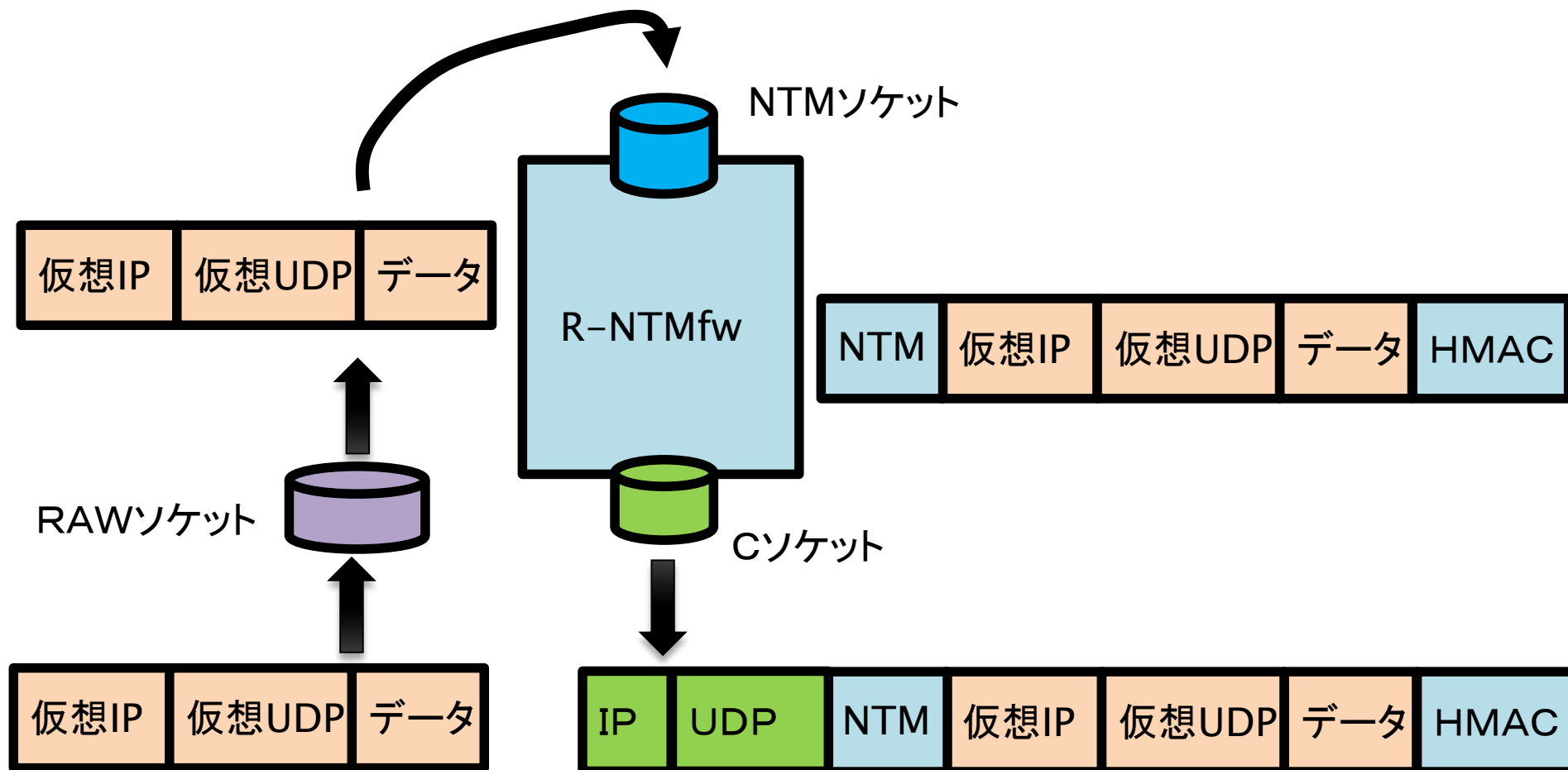
➤ lwIPの処理をスキップ



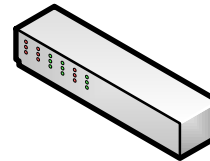
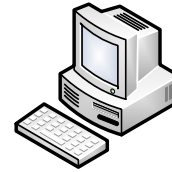
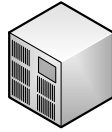
R-NTMfw(Remodeled NTMfw)

■ NTMfwを改造したアプリケーション

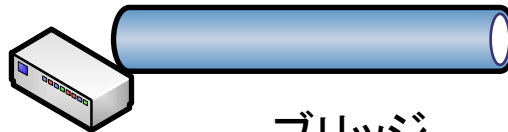
➤ lwIPの処理をスキップ



肥溜め



スイッチ



ブリッジ