令和元年度　修士論文

和文題目

NTMobileを利用したプライベートアドレス型WoTサーバに関する研究

英文題目

A Study on Private Address-Type WoT Server using NTMobile Technology

情報工学専攻　渡邊研究室

(学籍番号: 183426006)

黒宮　魁人

提出日: 令和2年1月28日

名城大学大学院理工学研究科

概要

モノがインターネットに繋がる IoT（Internet of Things）が普及しつつある中，様々なフレームワークが乱立しサイロ化することが懸念されている．そこで，プラットフォームに依存しない Web の技術を利用することにより，IoT やアプリケーションを連携する WoT（Web of Things）が World Wide Web Consortium によって提唱されている．しかしながら，WoT を支えるサーバはインターネットのグローバル空間に設置する必要があるため，サーバは常に DDoS 攻撃を始めとした脅威に晒される．2016 年にマルウェアである Mirai が IoT 機器に爆発的に感染したことにより，IoT 機器の脆弱性が明るみになり，今に至るまで新種のマルウェアの開発と対策が繰り返されている．そのため，本研究では NTMobile（Network Traversal with Mobility）を利用して，Web サーバをプライベート空間に設置し，異なるプライベートアドレス空間どうしで直接 WoT の通信を実現する方法を提案する．これによって，WoT を構成する全てのサーバや IoT 機器をプライベートアドレス空間に設置することができ，外部からの攻撃から保護することができる．この方式であればセキュリティが脆弱な IoT 機器であっても，マルウェアなどの脅威から保護できる．検証作業として RaspberryPi をラジコン戦車に拡張したものに制御用の Web サーバを構築し，提案手法の形式にて画像通信と制御が可能であることを確認した．

Abstract

While IoT (Internet of Things) is spreading in the world, there appears so many frameworks, and there are concerns that system is going to be silo. Therefore, WoT (Web of Things), which makes IoT and applications work together, has been proposed based on the Web technology that does not depend on specific platforms. However, However, WoT servers are always exposed to threats such as DDoS attacks, because WoT servers need to be located on the Internet of global addresses. In 2016, Mirai, a malware, exploded on IoT devices. This has revealed the vulnerability of IoT devices. Since then, new types of malware countermeasures have been studied. In this study, we propose a system to set a WoT server in a private address network and achieve the direct communication between the WoT server and clients in the different private address networks by using NTMobile. According to this method, all the WoT Server and IoT devices that make up WoT can be set in the private address areas. So, WoT Server can be protected from external attacks. As for verification, a RaspberryPi is used for the WoT server and extended to RC (Radio - Controlled) Tank (Hereafter, RasPiTank). We have confirmed that image communication and motor controls are possible by the proposed method.

目 次

第1章　序論

モノがインターネットに繋がるIoT（Internet of Things）が普及しつつある中，各プラットフォームでIoTシステムがが乱立しデータのサイロ化することが懸念されている．そこで，プラットフォームに依存しないWebの技術を利用することにより，IoTやアプリケーションを連携するWoT（Web of Things）がW3C（World Wide Web Consortium）によって提唱されている [1]．W3Cは，世界中で「すべて」を相互接続するために，標準化が重要であるとしており，One Web，Web for Allをスローガンに Web技術の標準化に取り組んできた [2]．そしてW3CはWoTを相互運用・多言語化・多様な入出力方法・アクセシビリティの特徴を持つOSやハードに依存しないプラットフォームになるよう期待している．また，電子書籍やVR（Virtual-Reality），教育などの幅広い範囲での応用を目指して，現在もWoTの標準化に取り組んでいる．

しかし，現在のWoTを構成するサーバやIoT機器はインターネットから直接アクセスできる物が多く存在し，悪意のある攻撃者から攻撃をされるリスクが存在する．実際，2016年にはIoT機器を利用したDDoS攻撃の脅威が報告されている [3] [4]．文献 [3] では，マルウェア “Mirai” に感染したIoT機器で構成されたボットネットによるDDoS攻撃や，“Mirai” の亜種による攻撃によって，DNSの提供会社が標的にされGitHubやTwitterといった大手のSNSに通信障害が発生するなどの事例から，IoT機器を利用したDDoS攻撃が現在顕在化しており，今後の社会において多大な影響を与える可能性があると指摘している．さらに，US-CERT（米国コンピュータ緊急事態対応チーム）[4] においても，“Mirai” を応用したDDoS攻撃が増加する可能性があるとして警告を行っている．そのため，Miraiや関連するマルウェアのDDoS対策が求められている．IoT機器は，IDとパスワードがハードコーディングされたまま出荷されているものが多く，パスワードが変更可能なものであってもユーザがパスワードの変更をしないケースも多い．加えて，23番ポートや2323番ポートでtelnetが動作するIoT機器が存在していたことが “Mirai” の爆発的な感染を後押しした．文献 [5] によると2016年10月の時点で51万5000台以上であると推定されている．また，“Mirai” に対抗して，善意のワームを名乗る “Hajime” や，“BrickerBot” といったマルウェアも出現した．これらのマルウェアもパスワードが初期設定の状態のIoT機器に感染し，爆発的に感染していった．しかし，これらのマルウェアは “Mirai” とは異なり，IoT機器をボットネット化してDDoS攻撃のために利用するわけではなく，“Hajime” は，“Mirai” の感染経路として利用される23, 5538, 5555, 7546番のポートをブロックし，IoT機器に作者からの警告文を出力させるというものであった．また，“BrickerBot” は，感染したIoT機器のストレージドライブにランダムビットを書き込み，TCPのタイムスタンプを無効に設定し，カーネルスレッドの最大数を1に変更する．これによって，カーネルが機能せず，インターネットを利用した通信が不可能になり，IoT機器を破壊する [6]．しかし，“BrickerBot” は “Mirai” に感染するセキュリティが脆弱なIoT機器に

対しての攻撃であり，作者である Janit0r 氏は，“Mirai” が作り出したボットネットに対する荒療治であり，ユーザとメーカに対する啓蒙であると語っている．結果として，“Mirai” や，“Hajime”, “BrickerBot” などのマルウェアが IoT 機器に感染する経路の奪い合いが始まっており，これらの事例から，IoT 機器に対するセキュリティの確保は急務であるとされている．

そこで，本研究ではセキュリティが脆弱な IoT 機器でも，悪意のある攻撃による被害を防ぐために，WoT を構成するサーバを NAT 配下に設置し，許可された端末からのみサーバにアクセスが可能なシステムの構築を目指す．そのために，NAT 越え問題を解決する必要があるが，現在 NAT 越え問題が解決できる技術の一例として，ICE (Interactive Conectivity Establishment) と OpenVPN (Open Virtual-Private-Network) がある [7] [8] [9]．ICE は，同じく NAT 越え技術として RFC にて標準化されている STUN (Session Traversal Utilities for NATs) [10] と TURN (Traversal Using Relay around NAT) [11] を組み合わせ，NAT 配下に存在する端末と NAT の情報を収集し，可能な限りサーバを中継しない最適な経路を提供する．ICE は，W3C が標準化を進めているオープンソースの WebRTC（Web Real-Time-Communication）にも採用されている．WebRTC は，ブラウザを利用して P2P（Peer to Peer）のリアルタイムコミュニケーションを実現する技術である．この時，WebRTC を実行している Peer は NAT 配下に存在する場合が大半であるので，ICE を利用することで NAT 越え問題を解決し，P2P 通信を実現する．しかし，ICE はアプリケーション作成時に ICE のライブラリを埋め込む必要があるため，既存システムへの適用が難しいという課題がある．OpenVPN は，NAT 越えを解決するために開発された技術ではないが，公衆の無線 LAN から自宅のネットワーク（自宅の NAT 配下）に安全にアクセスするという用途で利用されることが多い．OpenVPN を利用すると OpenVPN がインストールされた端末同士はトンネリング通信が行われる．これによって，第三者からはパケットの宛先は分かっても内容がわからないため，パケット盗聴の被害を防ぐことができる．また，基本的に OpenVPN を利用する端末同士は相互認証しているため，第三者によって改竄されたパケットは不正なものとして破棄される．そのため，公衆の無線 LAN からでも安全に自宅に対して通信を開始することが可能である．これを実現するために，OpenVPN はポートフォワーディングを利用して，公衆無線 LAN からのパケットを中継するサーバが必要となるが，NAT や Firewall の設定が必要になるため，設定が煩雑となりがちである．

そのため，本研究では，渡邊研究室，鈴木研究室と愛知工業大学の内藤研究室合同で開発が行われてきた NTMobile（Network Traversal with Mobility）を利用することで研究の目的である WoT を構成するサーバを NAT 配下に設置し，許可された端末からのみサーバにアクセスが可能なシステムの構築を行う [12–16]．NTMobile は，NAT や Firewall の変更を必要とせずに NAT 越え問題を解決し，IPv4/IPv6 ネットワークが混在した環境においても，端末の通信接続性を実現する通信技術である．さらに，端末のネットワークを切り替えても通信を継続することができる．DDoS 攻撃や Replay 攻撃，MitM（Man-in-the-Middle）攻撃に耐性を持ち，通信が全てカプセル化されるため，第三者からの盗聴に対しての耐性を持つ．現在まで NTMobile は，カーネルに実装するモデルや，通信ライブラリとしてアプリケーションに実装するモデル，Android 向けに VpnService を利用するモデルなど様々な種類が実装されてきたが，本研究では，TUN 型 NTMobile を利用する．これによって，TUN 型 NTMobile はカーネルの改造を必要とせずに既存アプリケーションに変更を加

えることなく，NTMobile 通信を実現できる．

研究の目的であるシステムの検証と評価を行うため，RaspberryPi に市販されているキットを装着して，RaspberryPi の GPIO ポートの出力によって自走するラジコン戦車を作成した．ラジコン戦車の中に WoT サーバを構築した．WoT サーバを構築するにあたり，RaspberryPi 向けの IoT フレームワークである WebIOPi を利用した．WebIOPi は，ブラウザから RaspberryPI の GPIO ポートの制御と監視を提供する．これによって，ブラウザからの操作でラジコン戦車の制御を可能とする．性能評価では，名城大学の多段 NAT 配下にある Wi-Fi ルータに接続したラジコン戦車に対して，UQ Mobile が提供するセルラーネットワークに接続したラップトップ PC から通信の確認を行った. RTT を TUN 型 NTMobile を利用した通信と TUN 型 NTMobile を利用しない通信で比較したところ，TUN 型 NTMobile を利用した通信は約 76%の性能になることが分かった．以降，2 章で NAT 越え手法に関する既存研究について述べ，3 章で NTMobile について説明する．4 章では提案と検証，評価について説明し，最後に 5 章でまとめる．

第2章　NAT越え手法に関する既存研究

本章では，現在インターネットに存在するNATの種類と挙動について説明した後に，安全なNAT越えを提供する既存研究について説明する．

2.1　NATの分類

以降の説明において，参考文献 [10] の形式に従い，NATの内側に存在する端末を “内部ホスト”，NATの外側にある端末を “外部ホスト” と呼称する．説明を簡略化するために，内部ホストのIPアドレスとポート番号を $(IP_{in}x : port_{in}x)$ と表記する．この時 x は内部ホストのラベルを示す．また，内部ホストが一台のときは x は省略する．外部ホストも同様に，IPアドレスとポート番号を $(IP_{ex}x : port_{ex}x)$ と表記する．この時 x は外部ホストのラベルを示す．また，外部ホストが一台のときは x は省略する．また，NATに内部ホストとマッピングされたインターネットに開放しているIPアドレスとポート番号を $(IP_{in'}y : port_{in'}y)$ と表記する．この時の y は内部ホストと対応したNAT外部のIPアドレスとポート番号のラベルとする．

2.1.1　RFC3489 における NAT の分類

RFC3489 [10] によると，NATの分類は4種類あり，それぞれが “Full Cone”, “Restricted Cone”, “Port Restricted Cone”, “Symmetric” と定義されている．

Full Cone　制約が最も弱いNATであり，$(IP_{in} : port_{in})$ が $(IP_{ex} : port_{ex})$ と1対1でマッピングされるNATであり，任意の外部ホストは $(IP_{in'} : port_{in'})$ に対してパケットを送信することで内部ホストにパケットを送信することができる．

Restricted Cone　$(IP_{in} : port_{in})$ が $(IP_{ex} : port_{ex})$ と1対1でマッピングされるNATであるが，内部ホストが以前にパケットを送信した宛先IPアドレスからのパケットのみ，内部ホストにパケットを送信する．Restricted Coneは外部ホストのIPアドレスのみ参照するため，例えば内部ホスト $(IP_{in} : port_{in})$ が外部ホストの $(IP_{ex} : port_{ex})$ に対してのエントリ $(IP_{in'} : port_{in'})$ を作成した時，$(IP_{in'} : port_{in'})$ に送信されるパケットの情報が $(IP_{ex} : port_{Any})$ となっていれば内部ホストにパケットが送信される．

Port Restricted Cone　振る舞いとしてはRestricted Coneと似ているNATであるが，Port Restricted Coneは，Restricted Coneとは異なり，受信パケットのIPアドレスだけでなくポート番号も参照するため，例えば内部ホスト $(IP_{in} : port_{in})$ が外部ホストの $(IP_{ex} : port_{ex})$ に対してのエントリ $(IP_{in'} : port_{in'})$ を作成した時，$(IP_{in'} : port_{in'})$ に送信されるパケットの情報が $(IP_{ex} : port_{ex})$ となっていれば内部ホストにパケットが送信される．

Symmetric 制約が最も強い NAT であり，同じ IP アドレスとポート番号から送信したパケットであっても，外部ホストの IP アドレスとポート番号毎に新しくマッピング情報を生成する．Symmetric は内部ホスト $(IP_{in} : port_{in})$ が外部ホストの $(IP_{ex}A : port_{ex}A)$, $(IP_{ex}B : port_{ex}B)$ に対してのエントリとして $(IP_{in'}A : port_{in'}A)$，$(IP_{in'}B : port_{in'}B)$ を生成する．Symmetric では，例えば $(IP_{in'}A : port_{in'}A)$ は，$(IP_{ex}A : port_{ex}A)$ から受信するパケット以外を全て破棄する．そのため，内部ホストにパケットを送信することができるのは，内部ホストからのパケットを実際に受信した外部ホストのみである．

しかしながら，RFC4787 によると，RFC3489 による NAT の分類では実際の NAT の挙動を説明することができず，“Full Cone” といった名称が多くの混乱を招いていると指摘している [17].

2.1.2 RFC4787 における NAT の分類

RFC4787 では NAT の挙動を “External NAT mapping characteristics” と “External Filter characteristics” の組み合わせによって分類している．RFC4787 において “External NAT mapping characteristics（外部 NAT マッピング特性）” は，以下のように定義されている．

Endpoint-Independent Mapping $(IP_{in}A : port_{in}A)$ から $(IP_{ex}A : port_{ex}A)$ と $(IP_{ex}B : port_{ex}B)$ にマッピングされる情報は $(IP_{in'}A : port_{in'}A) = (IP_{in'}B : port_{in'}B)$ となる．

Address Dependent Mapping $(IP_{in}A : port_{in}A)$ から $(IP_{ex}A : port_{ex}A)$ と $(IP_{ex}B : port_{ex}B)$ にマッピングする際に，登録される情報は $IP_{ex}A = IP_{ex}B$ のときに $(IP_{in'}A : port_{in'}A) = (IP_{in'}B : port_{in'}B)$ となる．

Address and Port Dependent Mapping $(IP_{in}A : port_{in}A)$ から $(IP_{ex}A : port_{ex}A)$ と $(IP_{ex}B : port_{ex}B)$ にマッピングする際に，登録される情報は $(IP_{ex}A : port_{ex}A = (IP_{ex}B : port_{ex}B)$ のときに $(IP_{in'}A : port_{in'}A) = (IP_{in'}B : port_{in'}B)$ となる．

次に，“External Filter characteristics（外部フィルタ特性）” は以下のように定義される．

Endpoint-Independent Filtering 内部ホスト $(IP_{in} : port_{in})$ に送信されていないパケットを無視し，$(IP_{in} : port_{in})$ に送信されているパケットはどのようなパケットであっても転送する．

Address Dependent Filtering 内部ホスト $(IP_{in} : port_{in})$ に送信されていないパケットを無視する．さらに，$(IP_{in} : port_{in})$ が以前に $(IP_{ex} : port_{Any})$ にパケットを送信していない場合，$(IP_{ex} : port_{ex})$ が $(IP_{in} : port_{in})$ に送信したパケットは無視する．

Address and Port Dependent Filtering 外側のポートが関連していることを除いて，Address Dependent Filtering と同様である．この外部フィルタ特性を持つ NAT において，特定の外部ホスト $(IP_{ex} : port_{ex})$ からのパケットを受信するためには内部ホスト $(IP_{in} : port_{in})$ が $(IP_{ex} : port_{ex})$ にパケットを送信する必要がある．

また，RFC4787 の定義を利用して，RFC3489 の NAT の挙動を当てはめたものを表 1 に示す．

RFC3489 における NAT の種類	External NAT mapping characteristics	External Filter characteristics
Full Cone	End-point Independent Mapping	Endpoint-Independent Filtering
Restricted	End-point Independent Mapping	Address Dependent Filtering
Port Restricted	End-point Independent Mapping	Address and Port Dependent Filtering
Symmetric	Address and Port Dependent Mapping	Address and Port Dependent Filtering

表 1　RFC4787 の定義に従った RFC3489 における NAT の分類

2.2　ICE (Interactive Conectivity Establishment)

これらの NAT の特性を踏まえた上で, NAT 越えを実現する技術である ICE (Interactive Conectivity Establishment) について説明をする.

2.2.1　ICE の概要

ICE (Interactive Connectivity Establishment) は，RFC8445 にて定義されている NAT 越え技術である [7]．ICE は同じく NAT 越え技術である STUN (Session Traversal Utilities for NATs) [10] と TURN (Traversal Using Relay around NAT) [11] を組み合わせて，優先度の高い経路を選択する．このとき，NAT や TURN サーバを経由しない経路が高い優先度を持つ.

2.2.1.1　STUN (Session Traversal Utilities for NATs)

NAT 配下に存在する端末に通信開始をするために，外部ホストは内部ホストの上位に存在する NAT にマッピングされている $(IP_{in'} : port_{in'})$ を知っている必要がある．STUN では，各内部ホストの $(IP_{in'} : port_{in'})$ を把握するために，STUN サーバを構築し，利用する．STUN サーバは，主に以下の機能を有するサーバである.

- 内部ホストからの Hole Punching を受け付け, 記憶する．Hole Punching は内部ホストから NAT の外に存在する端末にパケットを送信することで, NAT に $(IP_{in'}A : port_{in'}A) \rightarrow (IP_{in}A : port_{in}A)$（$\alpha \rightarrow \beta$ は，β に α のパケットを転送することを示す）エントリを作成する.
- Hole Punching を受け付けた端末との Keep Alive を受け付ける．これによって，Hole Punching によって作成された NAT エントリが時間経過によって削除されることを防ぐ.

実際に，STUN を利用して NAT 越え通信を行う際には，アプリケーションがインターネット上に，別途 Signaling Server を構築し，これを中継して通信相手の NAT のエントリ情報を取得する．この時，内部ノードを構築する NAT の種類が RFC4787 における定義のうち，Address Dependent Filtering，もしくは Address and Port Dependent Filtering である場合は，NAT のフィルタリング機能によってパケットが内部ホストに到達せずに破棄されるため，後述する TURN の利用が必須になる.

2.2.1.2　TURN (Traversal Using Relay around NAT)

TURN は RFC5766 にて定義されている STUN の拡張機能である．TURN では，インターネット上に設置された TURN サーバがパケットの中継を行うことで，NAT 越えを実現する．通信を行う

両端末が TURN サーバと通信を行うため，RFC3489 における Symmetric NAT や，RFC4787 における Address and Port Dependent Mapping かつ Address and Port Dependent Filtering という最も制約の強い NAT でもローカルネットワークに接続している端末に通信を開始することが可能である．

2.2.2 ICE の動作

ICE では，通信経路及びに通信相手の端末とのアドレス候補（以降，Candidate）を収集して，経路の探索を行い最適な経路を決定する．ICE で利用される Candidate は以下の種類がある．

Host Candidate 各端末のプライベート IP アドレスとポート番号

Server Reflexive Candidate NAT のグローバル IP アドレスとポート番号 (STUN を利用して収集)

Relayed Candidate TURN サーバの中継用グローバル IP アドレスとポート番号

これらの Candidate を SIP(Session Initiation Protocol) [18] や WebRTC (Web Real-Time Communication) [19] におけるシグナリングサーバを利用して，通信する端末はお互いの Candidate を交換する．次に，通信相手の Candidate を受信した端末は，自身の Candidate とのペア（Candidate Pair）を作成し，優先順位をそれぞれの Candidate Pair に割り振っていく．この時の優先順位は，TURN サーバを経由しないものが高い値を持つ．最後に，Candidate Pair の優先度が高いものから接続確認を行っていき，通信を行う両端末が STUN Binding Request を送信し，応答として STUN Binding Response を受信すれば，ICE を利用したコネクションが確立できる．

ICE の動作全体を簡略化して図示したものを図 1 に表す．図 1 は，ローカルネットワークに接続している Node A と Node B と各ノードが STUN サーバに KeepAlive している事を表す緑色の破線，TURN サーバを利用しない経路（緑色の直線の矢印），TURN サーバを中継している橙色の矢印によって構成されている．ICE では，これらの経路のうち接続可能な経路の中から，より優先度の高い経路を選択してコネクションを確立する．

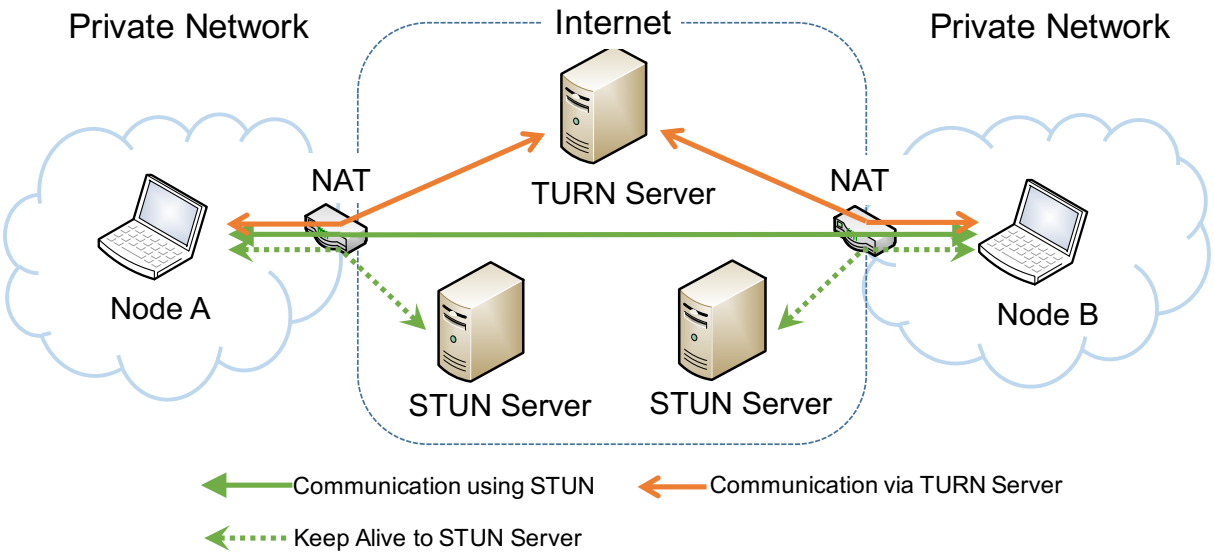

図 1　ICE の全体像

2.2.3 ICE の課題

これまで，ICE の有用性について述べてきたが，ICE にも課題は存在する．ICE は，ライブラリでの提供となるため，既存のシステムに対しての適用が難しい．また，STUN/TURN といったサーバ群，シグナリングサーバ等もアプリケーション提供者が準備をする必要がある．

2.3 OpenVPN (Open Virtual-Private-Network)

OpenVPN は，OpenVPN Technologies, Inc. を中心に開発が行われているオープンソースの VPN ソフトウェアである [8] [9]．OpenVPN は TUN/TAP と呼ばれる仮想のインタフェースを利用する事で，カーネル空間の書き換えを必要とせず，VPN 接続を実現することができる．

2.3.1 TUN/TAP について

TUN/TAP は仮想インターフェースであり，これを利用すると OS からは NIC (Network Interface Card) が 1 枚増設したと認識する．この，仮想インタフェースには IP アドレスを別途割り当てることができる．TUN と TAP の違いは，TUN が IP データグラムを扱うのに対して，TAP は Ethernet フレームを取り扱うことである．つまり，TUN デバイスを利用するとレイヤー 3（OSI 参照モデルではネットワーク層），TAP デバイスを利用するとレイヤー 2（OSI 参照モデルではデータリンク層）でトンネリングを行うことができる．

2.3.2 OpenVPN の目的と動作

OpenVPN は，公衆の無線 LAN を安全に利用したいときや，自宅のパソコンをリモートコントロールする時などに利用される．例えば前者であれば駅や飲食店に設置されるものが多いが，これらの無線 LAN は暗号化されていないものが存在し，尚且つ不特定多数の端末が接続していることが多い．このような無線 LAN で Web サイトにパスワードを送信した場合，悪意のある端末からの盗聴や改竄の標的にされる可能性がある．後者の場合，公衆の無線 LAN から自宅の PC に対してポートフォワーディングをしてリモートコントロールしていることが分かってしまうため，自宅の PC がパスワード解析攻撃の対象になる可能性がある．そのため，OpenVPN を利用すると OpenVPN がインストールされた端末同士はトンネリング通信が行われるため，第三者からはパケットの宛先は分かっても内容がわからないため，パケット盗聴の被害を防ぐことができる．また，基本的に OpenVPN を利用する端末同士は相互認証しているため，第三者によって改竄されたパケットは不正なものとして破棄される．そのため，公衆の無線 LAN からでも安全に自宅に対して通信を開始することが可能である．

図 2 に，OpenVPN の動作の全体像を示す．Home PC は，自宅のネットワークに接続している PC であり，Web Server は，Internet に設置されている一般的な Web サーバを想定している．VPN Client は，出先の Wi-Fi に接続した OpenVPN がインストールされたクライアント端末である．また，VPN Server は OpenVPN がインストールされた端末であり自宅のネットワークに接続されている．VPN Client は，Web Server や Home PC に接続する際に，出先のネットワークからではパケッ

トが盗聴や改竄される可能性があるため，OpenVPN のトンネリング通信を利用して VPN Server を中継して通信を開始する．そして，VPN Server は，VPN Client からパケットを受信時，デカプセル処理を行った後，VPN Client の代わりに Web Server や Home PC との通信を行う．これらの内容は，再度 VPN Server でカプセル処理を行ってから VPN Client に送信される．

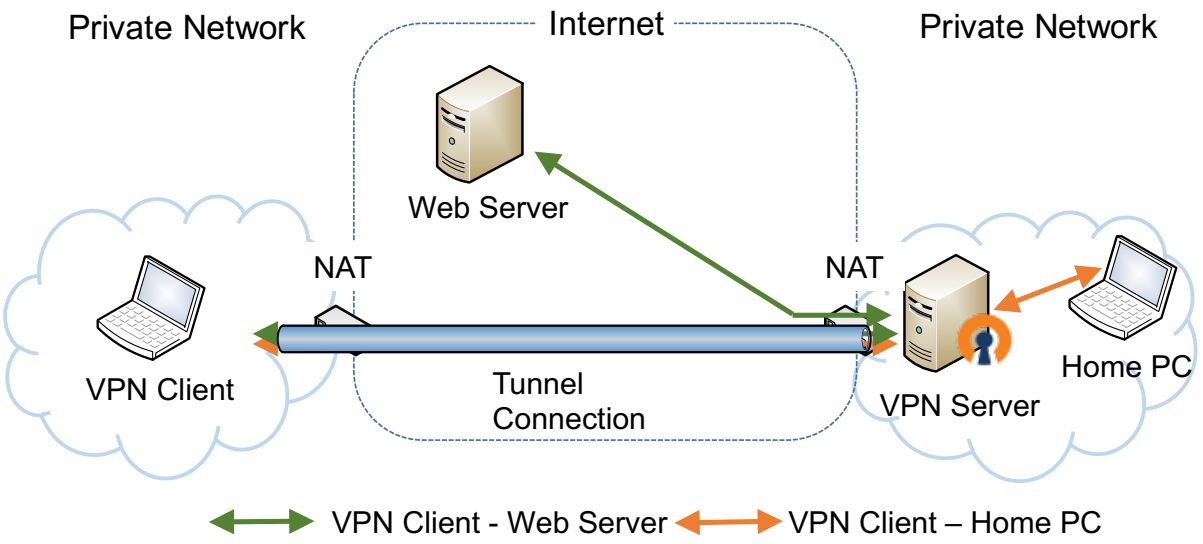

図 2　OpenVPN の全体像

2.3.3　OpenVPN の課題

これまで，OpenVPN の有用性について述べてきたが，OpenVPN にも課題は存在する．まず，OpenVPN における VPN Server は，自宅の端末にアクセスするために，自宅のネットワークに接続している必要がある．そのため基本的に NAT の配下に設置される．これによって，OpenVPN のために NAT のポートを開放する必要がある（ポートフォワーディング）．OpenVPN では，デフォルトで 1194 番ポートが用意されているため，OpenVPN を運用する事前準備として NAT やファイアウォールの設定で 1194 番ポートを通過させるように設定する必要がある．

第3章　NTMobile (Network Traversal with Mobility)

本章では，NAT 越え問題を解決した上で，IPv4/IPv6 の相互通信と移動透過性が実現できる NTMobile について説明する．また，文献 [20] における，EID（Endpoint Identifier）を通信識別子，RLOC（Routing Locator）を位置識別子と呼称する．

3.1　NTMobile 概要

NTMobile は，NAT の変更を必要とせずに NAT 越え問題を解決し，IPv4/IPv6 ネットワークが混在した環境においても，端末の通信接続性を実現する通信技術である．また移動透過性も有する．これによって NTMobile がインストールされた端末は IPv4/IPv6 ネットワークが混在した環境で自由な双方向通信が可能であり，通信中にネットワークを切り替えても通信を継続することができる．図 3 に NTMobile の構成を示す．NTMobile による通信を利用するためには，NTMobile がインストールされた端末（以降，NTM 端末）の他に，端末情報の管理や通信経路の指示，仮想 IP アドレスの割り当てを行う DC（Direction Coordinator），NTM 端末が直接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，RS といった NTMobile の装置群はデュアルスタックネットワークに設置されていることが前提である．また，DC と RS は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ある．これによって，各装置の負荷が分散できる．NTMobile では，DC が NTM 端末に対して，NTM 端末を識別する FQDN (Fully Qualified Domain Name) を割り当てる．また，この FQDN に紐づいた仮想 IP アドレスを割り当てる．この仮想 IP アドレスは IANA にて規定されている “198.18.0.0/15” の帯域を使用している [21]．Web などの一般アプリケーションはこの仮想 IP アドレスを通信識別子として通信を行い，ルータ等から割り当てられる実 IP アドレスを位置識別子としたカプセル化通信を行なうことで，ネットワークが切り替わっても通信識別子が変化しないため，そのまま通信を継続することができる．NAT 越え問題の解決は，DC から NTM 端末の通信経路の指示を行うことで実現している．これは，DC に対して NTM 端末が端末の登録処理を行っており，NAT の配下から KeepAlive を送信していることが前提である．インターネットに設置されている DC は，KeepAlive で開放されているポート番号を利用して通信経路の指示を送信する．これにより，NTM 端末は，通信相手とのトンネリング通信を構築する．通信が IPv4/IPv6 の相互通信の場合，もしくは NTM 端末が異なる NAT 配下に存在する場合は RS が通信の中継を行う．ただし，後者の場合，NAT の種類によっては通信を RS で中継せずに直接通信に切り替えることができる [22].

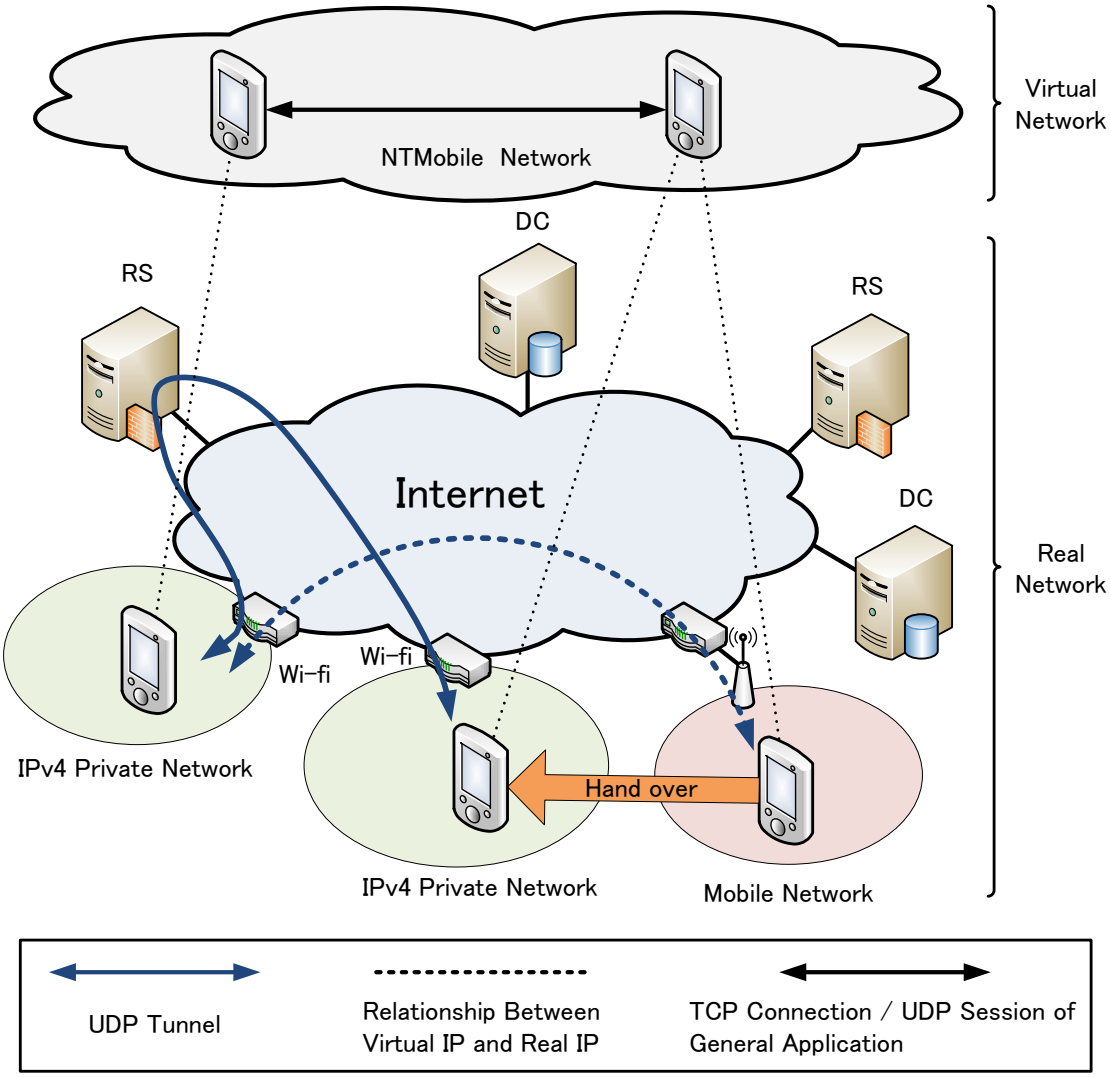

図 3　NTMobile の構成

3.2　NTMobile の動作

3.3　NTMobile Framework (NTMfw)

NTMfw は，BSD ソケット API と互換性のある NTM ソケットの API を提供する NTMobile の通信ライブラリである．図 4 に NTMfw のモジュール構成を示す．また，各モジュールの機能は以下の通りになっている．

NTM Socket API　NTMfw がパケットを送受信する際に用いる標準ソケット API であり，BSD ソケット API の代わりとしてアプリケーションに提供される．NTM ソケットの関数は，BSD ソケット API の関数に “ntmfw_” のプレフィックスを付与することで呼び出すことが可能である．NTMfw の初期化を行う処理と名前解決を行う処理を除き，Virtual IP Stack に処理を渡す．

Virtual IP Stack General Application が送受信する TCP/IP の処理を行う．NTMfw では lwIP (A lightweight TCP/IP stack) を Virtual IP Stack として利用している．

Tunnel Table 通信相手毎に，実 IPv4/IPv6 アドレス，仮想 IPv4/IPv6 アドレス，FQDN，共通鍵や RS の実 IPv4/IPv6 アドレス等をメンバとするエントリを持つ．

Packet Manipulation Module Genetal Application の通信パケットやシグナリングパケットの生成と解析を行う．General Application が通信に利用するパケットに対しては，NTMobile のヘッダ（以降，NTM ヘッダ）の操作，暗号化/復号化処理，MAC (Message Authentication Code) 認証処理を行い BSD ソケット API を通じて実インターフェースから送信を行う．

Negotiation Module NTMobile の初期化処理やシグナリング処理を行う．NTM ソケット API の名前解決を行う関数や，他の端末からの通信要求があった場合に，このモジュールでトンネル構築処理を行う．トンネル構築後は Tunnel Table を書き換える．

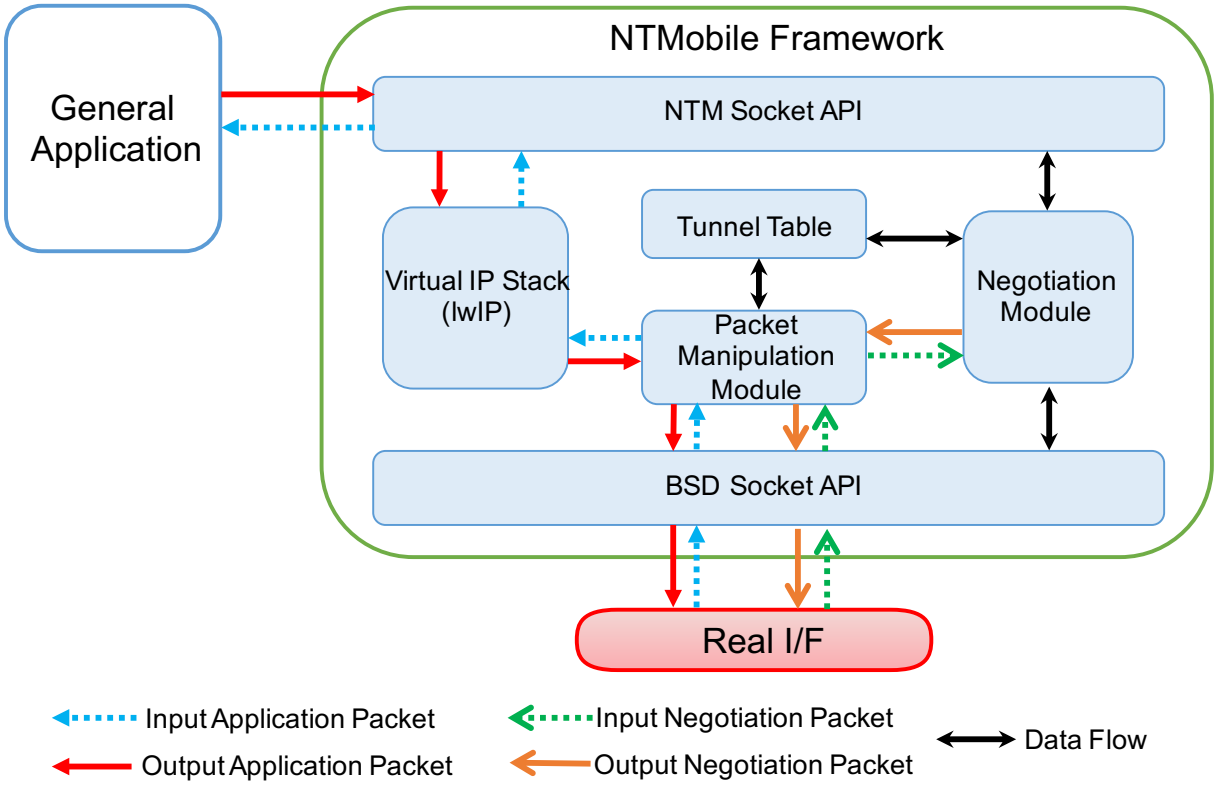

図 4　NTMfw のモジュール構成

3.4　TUN 利用型 NTMobile

TUN 型 NTMobile は，Linux の TUN サービスと，NTMfw を利用して作成されたアプリケーションである．TUN 型 NTMobile は TUN インターフェースに仮想 IP アドレスを割り当てて利用する．よって，NTMobile 通信を利用するアプリケーションが送信する仮想 IP アドレス宛の IP パケット

は，全て TUN インターフェースにルーティングされる．TUN インターフェースに流れるパケットは全てユーザ空間の NTMobile の通信ライブラリがフックして実インタフェースに書き込むため，既存のプログラムの書き換えやカーネルの改造は一切行う必要がない．これにより，端末の全ての通信は TUN 型 NTMobile のアプリケーションを起動するだけで NTMobile 通信ができる．

第4章　NTMobileを利用したプライベートアドレス型WoT Serverに関する研究

本章では，Web of Things にて利用されるサーバを保護する手法について説明する．

4.1　提案

図 5 に，に提案システムの構成を示す．Web of Things を構成する Web サーバ（以降，WoT サーバ）を NAT 配下に設置し，NTMobile を利用して通信する．これによって，NAT やファイアウォールの変更せずにどのような環境からでも WoT サーバに通信が開始できる．このとき，NTMobile 通信でない端末は NAT によって WoT サーバのアドレスが隠蔽されるため，通信を開始することができない．そのため，大規模な DDoS 攻撃から WoT サーバを守ることができる．NTMobile では，装置間では TLS 双方向による認証，端末は全て共通鍵で認証した通信を行っているため，MitM（Man-in-the-Middle）攻撃に耐性を持ち，リプレイ防御ウィンドウによるリプレイ攻撃の対策もされている．また，DDoS 攻撃対策として，MAC (Message Authentication Code) 認証が実装されている．さらに，シーケンス番号と共通鍵を利用した簡易認証をパケット受信時に実行することで不正なパケットをより高速に破棄できる．現在，DC に NTMobile の通信グループの設定を行う検討がされており，これが実装されれば，NTMobile 通信であっても，許可されていない端末には DC が通信経路の指示をしない．これによって，悪意のある攻撃の被害が小さくなると考えられる．

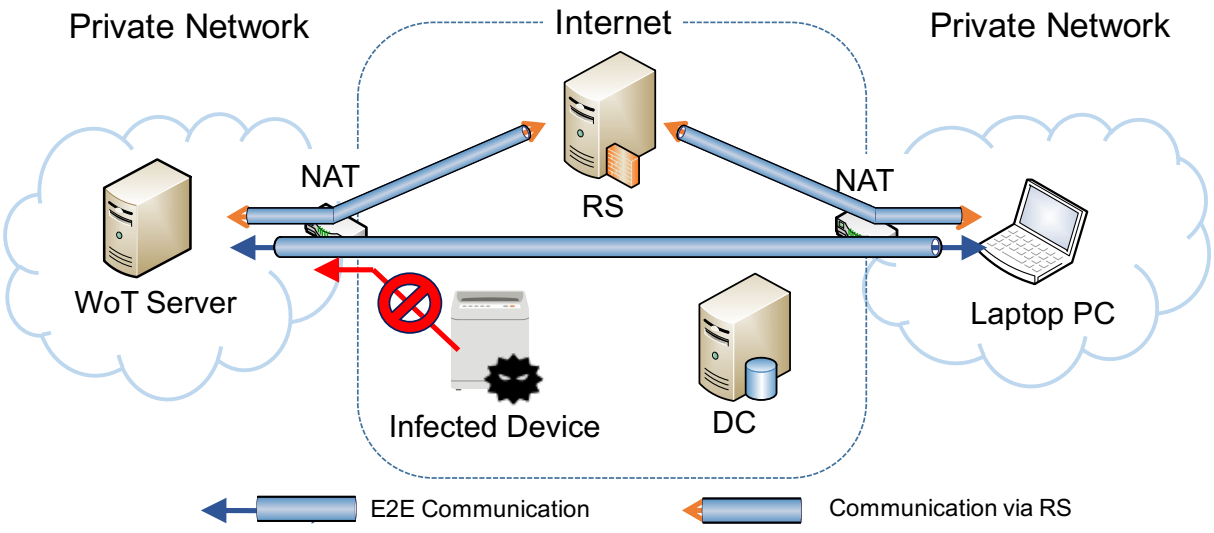

図 5　提案システムの全体像

4.2　検証

検証作業では，RaspberryPi の中に WoT サーバを構築し，異なるローカルネットワークからの制御を確認した．

4.2.1　検証の概要

RaspberryPi に市販されているキットを装着して，RaspberryPi の GPIO ポートの出力によって自走するラジコン戦車を作成した（以降，RasPiTank）．検証では，この RasPiTank の中に WebIOPi を利用して WoT サーバを構築し，異なるローカルネットワークに接続されたラップトップ PC のブラウザから WoT サーバにアクセスし，RasPiTank を制御した．

4.2.2　検証に利用した機材

検証を行うにあたり，以下の機材を準備した．

4.2.2.1　TAMIYA の楽しい工作シリーズ　タンク工作基本セット

「TAMIYA の楽しい工作シリーズ　タンク工作基本セット」(以降, RC Tank セット) は, StreamTechnology 社から発売されているキットである [23]．図 6 に，RC Tank セットの外観を示す．RC Tank セットは，モータ，単三電池ボックス，ギヤボックス，ホイール，キャタピラによって構成されている．

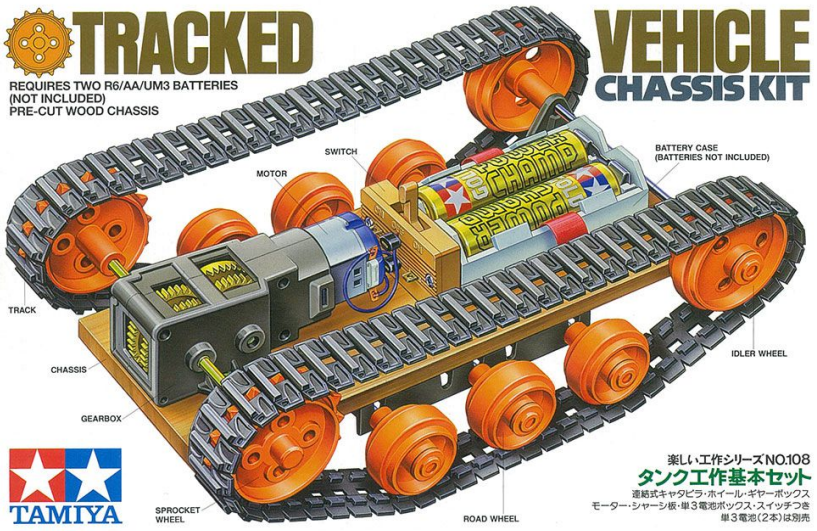

図 6　RC Tank セットの外観

4.2.2.2　デュアルモータドライバ（型番：DRV8835）

Pololu Corporation から発売されているデュアルモータドライバである [24]．図 6 に，RC Tank セットの外観を示す．

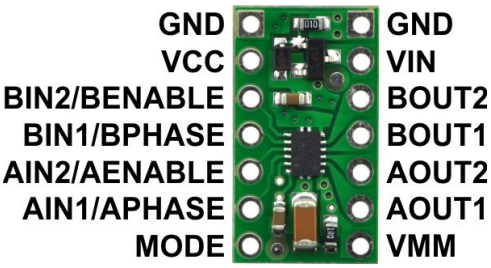

図 7　DRV8835 の外観

表 2　DVR8835 のピンの名称と機能

ピン	名称	機能	ピン	名称	機能
1	VMM	モータ電源	12	VCC	ロジック電源
2	AOUT1	A 出力 1	11	MODE	モード設定
3	AOUT2	A 出力 2	10	AIN1	A 入力 1
4	BOUT1	B 出力 1	9	AIN2	A 入力 2
5	BOUT2	B 出力 2	8	BIN1	B 入力 1
6	GND	グラウンド	7	BIN2	B 入力 2

ここで，DRV8835 のピンの名称と機能を表 2 に示す．

DRV8835 は，IN/IN モードと PHASE/ENABLE モードの切り替えをすることができる．IN/IN モードは，細かい出力が調整できるモードであり，PHASE/ENABLE モードは 1 つのピンでモータの方向を決定し，もう一つのピンでモータの出力が調整できるモードである．DRV8835 の各モードの入力と出力の対応は表 3, 4 に示す通りである．

表 3　IN/IN モード（Mode = 0）における入力と出力の対応

Mode	xIN1	xIN2	xOUT1	xOUT2	FUNCTION
0	0	0	Z	Z	Coast（空転）
0	0	1	L	H	Reverse（反転）
0	1	0	H	L	Forward（正転）
0	1	1	L	L	Brake（ブレーキ）

表 4　PHASE/ENABLE モード（Mode = 1）における入力と出力の対応

Mode	xENABLE	xPHASE	xOUT1	xOUT2	FUNCTION
1	0	x	L	L	Brake（ブレーキ）
1	1	1	L	H	Reverse（反転）
1	1	0	H	L	Forward（正転）

本検証において，DRV8835 は IN/IN モードを使用し，RC Tank セットの単三電池による出力をモータ電源に接続し RaspberryPi の VCC 出力をロジック電源に接続した．

4.2.3 WebIOPi

WebIOPi は，RaspberryPi を対象に WoT の仕組みを提供するフレームワークである [25]．WebIOPi を利用することで，Web ブラウザから RaspberryPi の GPIO ポートの制御や監視，デバッグを行うことができる．図 8 に，文献 [25] のチュートリアルとして，WebIOPi の動作シーケンスが紹介されていたため，引用する．

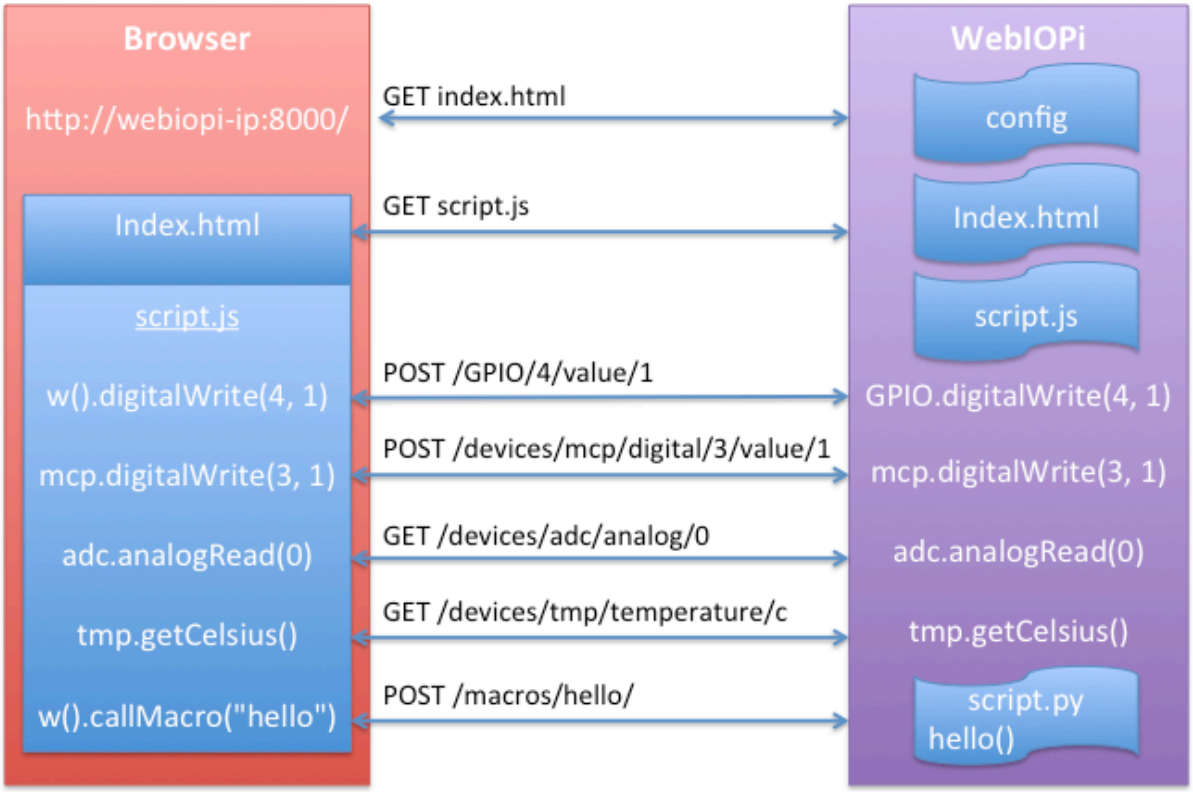

図 8　WebIOPi の動作概要

図 8 の動作の流れについて軽く説明する．クライアントはブラウザから WebIOPi によって構築されたサーバにアクセスをすると html ファイルと WebIOPi のスクリプトを取得することができる．このスクリプトは，サンプルとして digitalWrite() や，getCelsius() といった処理が用意されているため，html ファイルからイベントとして設定すれば，クライアント端末のブラウザから RaspberryPi の GPIO ポートを制御することができる．

4.2.4 プロトタイプ実装

4.2.2 節で説明した機材を用いて RasPiTank を構築し，RasPiTank を自走させるための Web サーバと Web ページのプロトタイプ実装を行ったため記述する．

4.2.4.1　RasPiTank の構成

まず，4.2.2 にて紹介した機材をもとに組み立てた RasPiTank の外観を，図 9 に提示する．使用した RaspberryPi は RaspberryPi3 ModeB を利用した．OS は Raspbian 9.11 がインストールされている．また，RaspberryPi には，RaspberryPi Camera Module V1（Rev1.3）を接続し，主電力として BUFFALO 製の 5200 mAh のモバイルバッテリー（型番：BSMPB5201P2WH）を利用している．次に，図 10 の左半分が RasPiTank の GPIO ポートであり，デュアルモータドライバとの配線を提示している．ここで，Motor Power は，RC Tank セットに同梱されている単三電池ボックスに単三電池を装着したものである．また，DRV8835 は IN/IN モードを使用するため，MODE のポートは GROUND に繋いでいる．

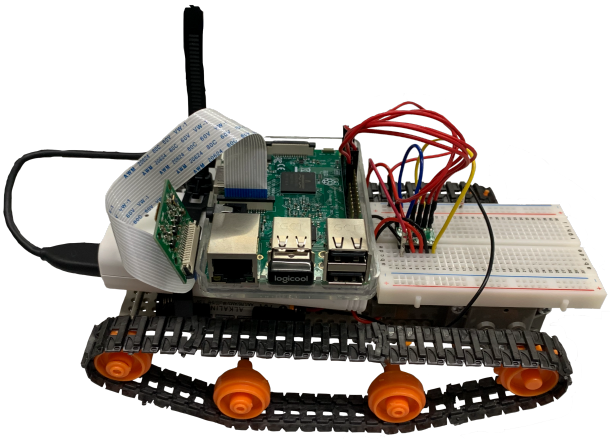

図 9　RasPiTank の外観

4.2.4.2　作成した Web サーバと Web ページ

図 11 に作成した Web ページを示す．図 11 における画像部分は，mjpg-streamer を利用して，Web ページを構成する html ファイルに埋め込んだ [26]．mjpg-streamer は，Web ベースのネットワークを介して JPEG ファイルを Web カメラからブラウザにストリーミングできる．図 11 における矢印は，RasPiTank を矢印の方向に走行させるボタンである．これは WebIOPi をインポートした python ファイルに GPIO ポートを操作する処理を記述して，ボタンに対応した移動方向にモータが回転するように設定した．また，ボタンのアイコンは Font awesome を html ファイルにインポートして利用した．

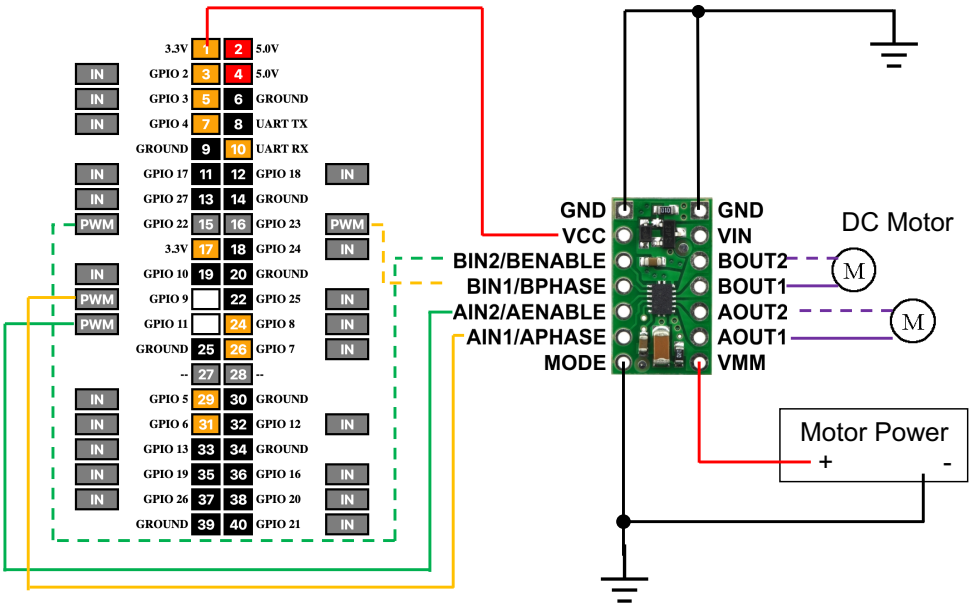

図 10　RasPiTank の配線

4.3　評価

表 5 に，既存研究として挙げた ICE，OpenVPN，TUN 型 NTMobile の定性的な評価を行った結果を示す．既存システムに適用の項目では，ICE はアプリケーション作成の際に ICE のライブラリを組み込む必要があるため×に，OpenVPN と TUN 型 NTMobile はそれぞれアプリケーションを起動すれば，ローカルネットワークの WoT サーバに対して通信が可能であるため○とした．次に，NAT とファイアウォールの設定であるが，OpenVPN は提案方式の構成においては，VPN Server をローカルネットワークに設置する必要があるが，この時，NAT とファイアウォールの設定を変更する必要がある場合があるため△としている．最後に OS の自由度であるが，ICE や OpenVPN は Windows や iOS，Linux と Android と幅広い OS に対応しているため○にしている．TUN 型 NTMobile は基本的に検証は Linux で行われており，現段階では Windows や iOS にて動作しないが，原理的には動作可能であるため，△の評価を与えている．DDoS 対策への耐性の項目は，ICE は組み込まれたアプリケーションに依存するため評価を行っていない．また，DDoS 攻撃による不正なパケットは NTMobile の簡易認証によって，既存研究より高速に処理ができるため◎の評価を与えている．

表 5　既存研究との比較

	ICE	OpenVPN	NTMobile
既存システムへの適用	×	○	○
NAT とファイアウォールの設定	○	△	○
対応する OS の種類	○	○	△
DDoS 攻撃への耐性	-	○	◎

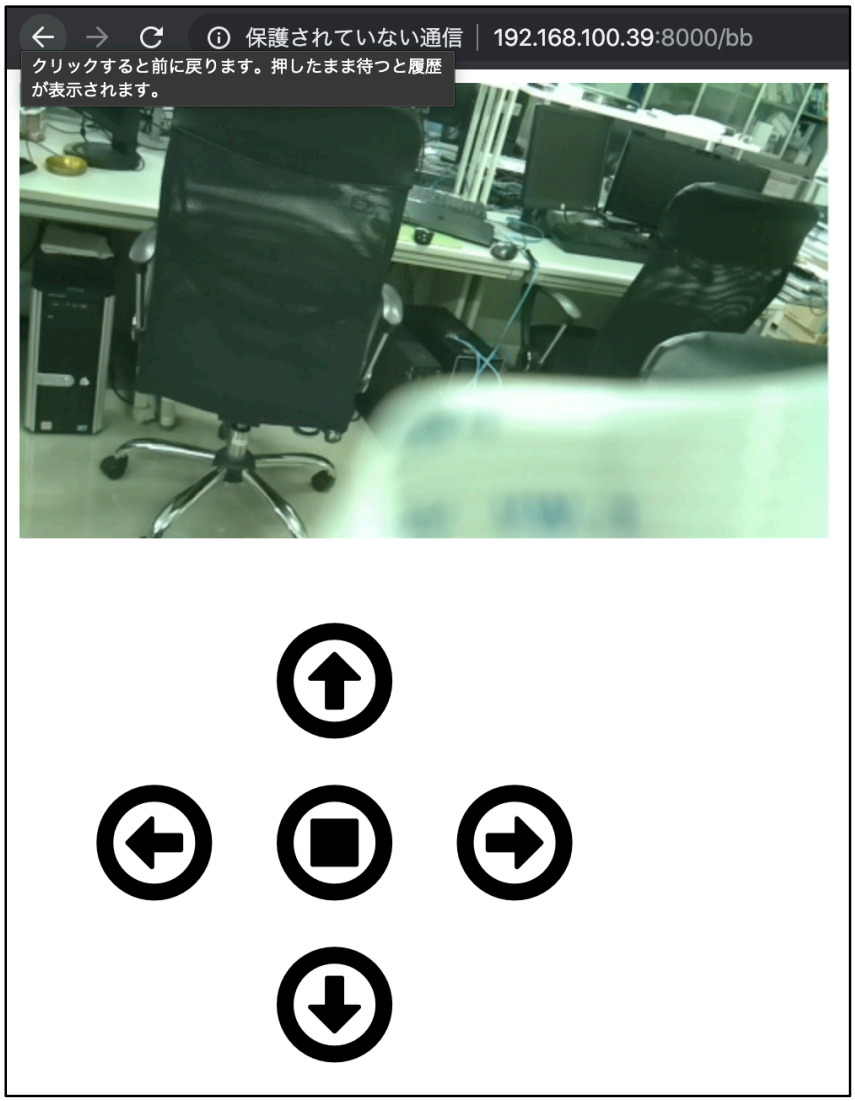

図 11　作成した Web ページのキャプチャ

次に，TUN 型 NTMobile の性能を評価するため，Mobile Network から RS を経由し，RasPiTank にメッセージが到達するまでの時間を測定した．測定に使用した諸元を表 7 に示す．DC と RS は ablenet が提供している VPS (Virtual Private Server) を使用している．

表 6　Specification of each equipment.

	OS	CPU	Memory
WebServer	Raspbian 9.11	Cortex-A53 4 Core 1.2GHz	1GB
Client	Ubuntu16.04LTS	Intel Corei5-2520M 2.5GHz	2GB
DC	CentOS 6.9	Virtual CPU 1 Core 3.3GHz	512 MB
RS	CentOS 6.9	Virtual CPU 2 Core 3.1GHz	1536 MB

図 12 に，性能評価を行った環境を示す．

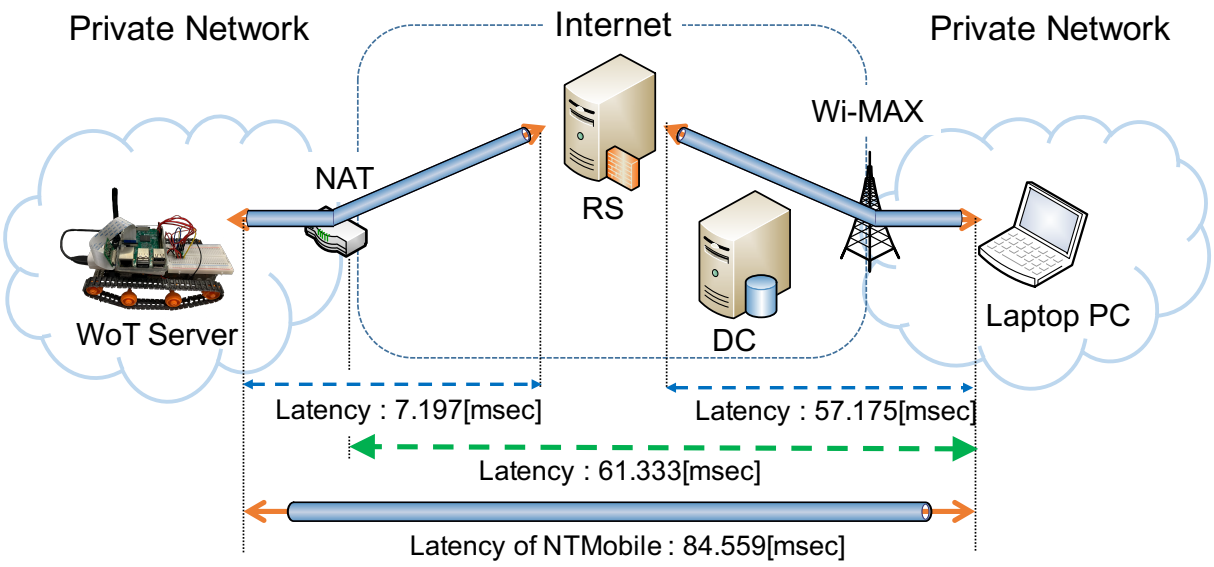

図 12　性能測定を行なった環境

WoT サーバを実装した RasPiTank を名城大学の多段 NAT 配下にある Baffalo 社の WAPM-1266R に接続し，UQ Wi-MAX2+のセルラーネットワークに接続したラップトップ PC から RS 経由の経路で通信の確認を行った．RasPiTank を制御するパケットは，NTMobile を利用しない状態では，パケットが到達しないため，性能評価行うための代替の手段として，Ping による RTT を利用する．図 12 における，水色の破線矢印は，RS から各端末までの RTT である．次に，緑色破線の矢印は，Laptop PC から WoT サーバの名城大学の上位 NAT までの RTT である．最後に，橙色の矢印であるが，これは，NTMobile を利用して RS を中継した Laptop PC から WoT サーバまでの経路の RTT である．このパケットは 67.929 msec にて Laptop PC から RasPiTank に到達した．また，この値は 100 回送信した際の平均値である．この性能は NTMobile を利用しない通信の約 92%であった．図 13 は，NTMobile を利用した通信と NTMobile を利用しない通信の平均値と標準偏差を示している．外れ値 $z_n = \frac{x_n - \mu}{s}$ が 5 より大きいものは測定結果から除外している．NTMobile を利用する通信は RTT が，$\mu = 67.929$，$s = 12.667$ という結果が，NTMobile なしの通信では，$\mu = 62.646$，$s = 9.327$ という結果が得られた．また，データのヒストグラムを図 14 に示す．次に，NTMobile を利用した通信と NTMobile なしの通信で通信の性能が等価であるかを検定するために図 14 に対して t 検定を行った．この時，等分散性を確認するために “2 群間の分散が等しい” という帰無仮説のもと有意水準 0.05 の F 検定を行った．この時，$P(F \leq f) = 0.00128660015481121$ となり等分散であるという帰無仮説は棄却された．そのため，t 検定はウェルチの t 検定を採用した．ウェルチの t 検定では，“2 群間の平均値は等しい” という帰無仮説のもと，有意水準 0.05 にて検定を行った．結果，$P(T \leq t) = 00.000489127$ となり，帰無仮説は棄却された．

測定結果に関する考察であるが，NTMoible を利用した通信と NTMobile なしの通信の性能が等価であるということが示せない上に，実環境の通信では，RS の設置場所による遅延や RS が大多数の通信を中継することによる遅延などが含まれると考えられる．これに関しては，NTMobile の経路最適化を利用することで，ある程度改善すると考えられる．また，図 13 において，NTMobile

を利用した通信の方が，標準偏差が大きくなっているが，これは，NTMobile がパケットを処理する時間のばらつきや RS がパケットを中継する時間のばらつきによって標準偏差が大きくなったものであると考えられる．これらのことから，リアルタイム性が必須とされるシステムでなければ，安全に WoT システムの構築が可能であると考えられる．

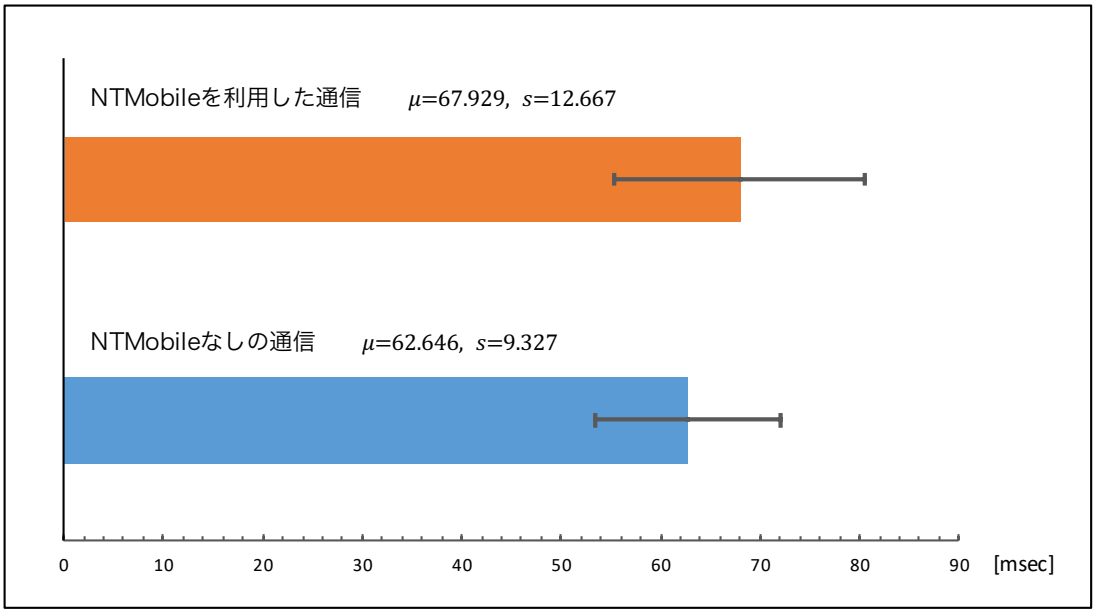

図 13　性能測定の結果

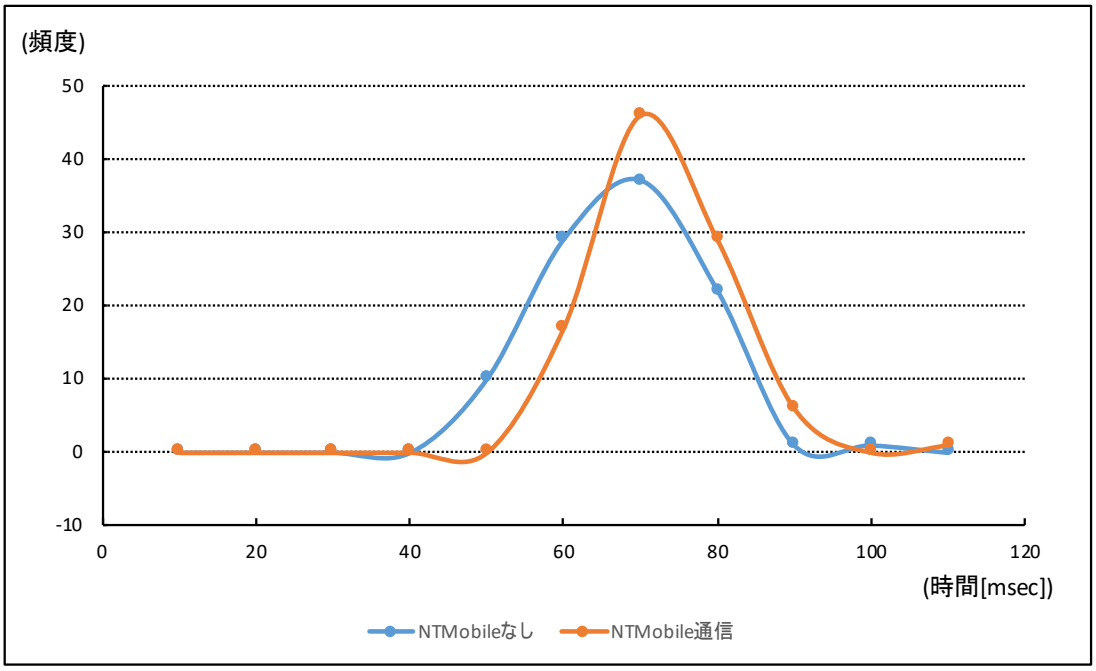

図 14　測定結果のヒストグラム

第5章　結論

本研究ではNTMobileを利用して，セキュリティが脆弱なIoT機器でも，悪意のある攻撃による被害を防ぐために，WoTを構成するサーバをNAT配下に設置し，許可された端末からのみサーバにアクセスが可能なシステムの提案を行った．提案システムを検証するため，RaspberryPiのGPIOポートの出力によって自走するラジコン戦車を作成し，WoTサーバをWebIOPiを利用して構築した．このWoTサーバはブラウザからの操作でラジコン戦車の制御するためのプログラムが記述してある．性能評価では，名城大学の多段NAT配下にあるWi-Fiルータに接続したラジコン戦車に対して，UQ Mobileが提供するセルラーネットワークに接続したラップトップPCから通信の確認を行った. NTMobileを利用する通信はRTTが，$\mu = 67.929$，$s = 12.667$という結果が，NTMobileなしの通信では，$\mu = 62.646$，$s = 9.327$という結果が得られた．これらの結果に対して，通信が等価であるかを確かめるために“2群間の平均値は等しい”という帰無仮説のもとウェルチのt検定を行ったところ，$P(T \leq t) = 00.000489127$となり，帰無仮説は棄却された．そのため，本研究にて提案したシステムは，リアルタイム性が必須とされるシステムでなければ，安全にWoTシステムの構築が可能であると考えられる．

謝辞

本研究を進めるにあたり，多大なるご指導とご教授を賜りました，名城大学理工学部情報工学科の渡邊晃教授に心から感謝いたします.

本研究を進めるにあたり，様々なご指導とご意見を賜りました，名城大学理工学部情報工学科の鈴木秀和准教授に深く感謝いたします.

本研究を進めるにあたり，ご意見とご助言を賜りました，愛知工業大学情報科学部の内藤克浩准教授に深く感謝いたします.

本論文を作成するにあたり，副査を快諾して頂きました，名城大学理工学研究科情報工学専攻の柳田康幸教授に心より感謝いたします.

本研究は，公益財団法人中部電気利用基礎研究振興財団の平成 30 年度国際交流援助海外渡航費助成を受けて行われたものであり，ここに記し，感謝いたします.

最後に，本研究を進めるにあたり，親身かつ丁寧なご指導を賜りました，渡邊研究室及び鈴木研究室の先輩方と，数々の有益なご助言を賜りました渡邊研究室及び鈴木研究室の諸氏に感謝いたします.

参考文献

[1] : World Wide Web Consortium. https://www.w3.org/.

[2] 芦村和幸:WoT と W3C の Web 技術標準化 (2019). https://www.nri.com/-/media/Corporate/jp/Files/PDF/news/event/lst/2019/mcs/wot_ideathon/20190912_1.pdf?la=ja-JP&hash=45A9ACE728F771BD9452B31408302512DA730E3C.

[3] 辻　宏郷：顕在化した IoT のセキュリティ脅威とその対策 (2017). https://www.ipa.go.jp/files/000062277.pdf.

[4] : Heightened DDoS Threat Posed by Mirai and Other Botnets. https://www.us-cert.gov/ncas/alerts/TA16-288A.

[5] : Shodan. https://www.shodan.io/report/aE9jvAXo.

[6] Cimpanu, C.: New Malware Intentionally Bricks IoT Devices. https://www.bleepingcomputer.com/news/security/new-malware-intentionally-bricks-iot-devices/.

[7] : Interactive Connectivity Establishment (ICE):
A Protocol for Network Address Translator (NAT) Traversal. https://tools.ietf.org/html/rfc8445.

[8] : openvpn: VPN Software Solutions & Services For Business. https://openvpn.net/.

[9] 塩見豊久，紫藤政義：OpenVPN で構築する超簡単 VPN 入門，「OpenVPN」出版プロジェクト (2006).

[10] : Session Traversal Utilities for NAT (STUN). https://tools.ietf.org/html/rfc5389.

[11] : Traversal Using Relays around NAT (TURN):
Relay Extensions to Session Traversal Utilities for NAT (STUN). https://tools.ietf.org/html/rfc5766.

[12] 鈴木秀和，水谷智大，西尾拓也，内藤克浩，渡邊　晃：NTMobile における相互接続性の確立手法と実装，マルチメディア，分散，協調とモバイル（DICOMO2011）シンポジウム論文集，Vol. 2011, No. 1, pp. 1339–1348 (2011).

[13] 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. 52, No. 9, pp. 2549–2561 (2011).

[14] 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. 367–379 (2013).

[15] 内藤克浩，上酔尾一真，水谷智大，西尾拓也，鈴木秀和，渡邊　晃：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).

[16] 山田貴之，鈴木秀和，内藤克浩，渡邊　晃：IPv4/IPv6 混在環境に対応した VPNService 型 NTMobile の性能評価，マルチメディア，分散，協調とモバイル（DICOMO2015）シンポジウム論文集，Vol. 2015, pp. 1792–1799 (2015).

[17] : Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. https://tools.ietf.org/html/rfc4787.

[18] : RFC 3261 - SIP: Session Initiation Protocol. https://tools.ietf.org/html/rfc3261.

[19] : WebRTC Home — WebRTC. https://webrtc.org/.

[20] : RFC 6830 - The Locator/ID Separation Protocol (LISP). https://tools.ietf.org/html/rfc6830.

[21] : RFC 2544 - Benchmarking Methodology for Network Interconnect Devices. https://tools.ietf.org/html/rfc2544.

[22] 納堂博史，鈴木秀和，内藤 克浩晃：NTMobile における自律的経路最適化の提案，情報処理学会論文誌，Vol. 54, No. 1, pp. 394–403 (2013).

[23] : RasPiTank 工作キット ST-TANK-KIT - Stream Technology. https://www.streamtechnology.co.jp/RasPiTank/.

[24] : DRV8835 Dual Motor Driver Carrier. https://www.pololu.com/product/2135.

[25] : The Raspberry Pi Internet of Things Toolkit. https://webiopi.trouch.com/.

[26] : mjpg-streamer. https://github.com/jacksonliam/mjpg-streamer.

[27] : ConnectivityManager — Android Developers. https://developer.android.com/reference/android/net/ConnectivityManager.html.

研究業績

研究会・大会等（国際会議　査読あり）

（1）黒宮魁人，田中久順，鈴木秀和，内藤克浩，渡邊 晃：Implementation and Evaluation of IP Mobility Functions in NTMobile for Android，Proceedings of The 11th International Conference on Mobile Computing and Ubiquitous Network (ICMU 2018), Oct. 2018.

（2）黒宮魁人, 田中久順, 鈴木秀和, 内藤克浩, 渡邊 晃：Proposal of Private Address-Type Web Server using NTMobile Technology,　IEEE International Conference on Consumer Electronics(ICCE 2020), Jan. 2020.

研究会・大会等（国内会議　査読あり）

（1）黒宮魁人，田中久順，鈴木秀和，内藤克浩，渡邊 晃：Android 向け NTMobile の移動機能の実装と評価，マルチメディア，分散，協調とモバイル (DICOMO2018) シンポジウム，Vol.2018, No.7C-2，pp.1390 － 1396，2018 年 7 月 6 日.

研究会・大会等（国内会議　査読なし）

（1）黒宮魁人，清水一輝，鈴木秀和，内藤克浩，渡邊 晃：スマートデバイスにおける NTMobile の経路生成方式の提案，平成 29 年度電気・電子・情報関係学会東海支部連合大会論文集, Vol.2017, No.C3-4, Sep. 2017.

（2）黒宮魁人，清水一輝，鈴木秀和，内藤克浩，渡邊 晃：スマートデバイスにおける NTMobile の経路生成方式の提案，第 15 回情報学ワークショップ（WiNF2017）論文集，Vol.2017, No.D-4, Nov. 2017.

（3）黒宮魁人，清水一輝，鈴木秀和，内藤克浩，渡邊 晃：VPNService を利用した移動透過性実現方式の提案，第 80 回情報処理学会全国大会講演論文集，Vol.2017, No.1，6T-07, Mar. 2018.

（4）黒宮魁人，田中久順，鈴木秀和，内藤克浩，渡邊 晃：Android 向け NTMobile の移動機能の実現，平成 30 年度電気・電子・情報関係学会東海支部連合大会論文集，Sep. 2018.

（5）黒宮魁人，田中久順，鈴木秀和，内藤克浩，渡邊 晃：実環境を用いた Android 向け NTMobile の移動機能の確認，第 16 回情報学ワークショップ（WiNF2018）論文集，Vol.2018, Nov. 2018.

展示会

（1）Interop Tokyo 2019

（2）2019 年度中部地区医療・バイオ系シーズ発表会

付 録A　VpnService型NTMobileの移動機能に関する研究

本章では，Android 向けに動作する VpnService 型 NTMobile に移動透過性を実現するための研究を行ったため，説明する．同時に，本文では説明しなかった，NTMobile のシーケンスについても説明する．

A.1　研究の概要

Android にて動作する NTMobile は，VpnService を利用して実現されている．VpnService 型 NTMobile は一般アプリケーションを使用して NAT 越えや異なるバージョンの IP アドレスによる相互通信などの一部の機能を確認済みである．しかし，NTMobile 通信を実行するライブラリが，アドレス変化検出を異なる OS で共通で実現できないため，移動透過性が実現できていない．本研究では，VpnService 型 NTMobile に移動検出機能を実装することで，ネットワークが切り替わった時に NTMfw の移動処理を実行させることで，VpnService 型 NTMobile に移動透過性を実現した．

A.2　NTMobile の補足

A.2.1　VPNService 型 NTMobile

VPNService 型 NTMobile は，NTMfw をスマートデバイスアプリケーションとして利用するために実装されたモデルである [16]．この実装方式は，VPN 通信を行うための API（以降，VPN API）である VPNService と NTMfw を利用している．VPNService 型 NTMobile は VPN API によるカプセル化 / デカプセル化機能を利用することで，既存のアプリケーションに対して NTMobile 上での通信を実現することが可能である．Java アプリケーションとして実装されており，C で記述された NTMfw を呼び出すために JNA（Java Native Access）を使用する．VPNService 型 NTMobile では NTMobile の NAT 越えやエンドツーエンドの直接通信を確認済みである．しかし，NTMfw が OS によって異なるインタフェースの名前を共通で検出することが出来ないため，端末の IP アドレスが変化しても移動処理を実行することが出来ないという課題が残されている．

A.2.2　NTMobile の動作シーケンス

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）とする．図 15 に NTMobile の通信開始時のシーケンスを示す．図 15 では，MN，CN は

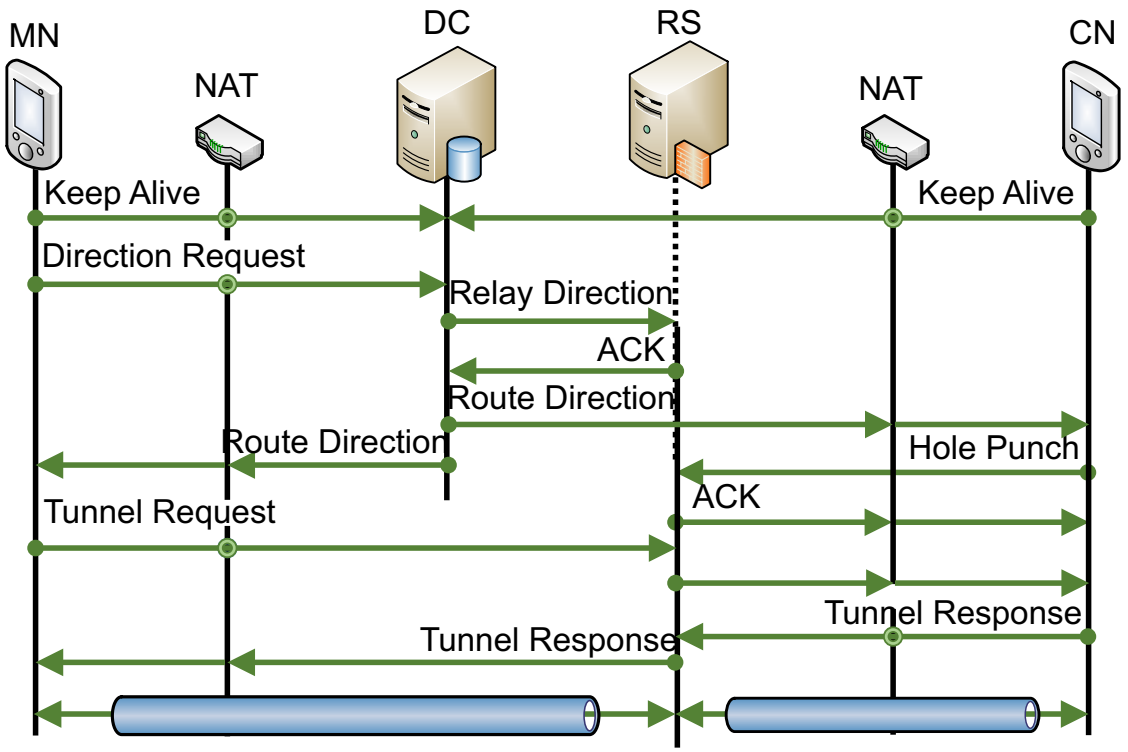

図 15　通信開始時の NTMobile のシーケンス

異なる NAT 配下に存在している．また，簡略化のため DC，RS は 1 台としている．前提として，DC には MN と CN の端末情報が既に登録されており，定期的な Keep Alive が行われている．

通信開始時，MN は DC に経路指示要求として CN の FQDN を含む Direction Request を送信する．DC は受信した Direction Request と，登録済みの CN の情報を確認する．図 15 のネットワーク構成では，MN と CN が共に NAT 配下であることから RS を経由した通信経路を構築する必要があると判断する．DC は RS に対して通信の中継指示である Relay Direction を送信し，RS は ACK を返信する．次に DC は MN と CN に経路指示として Route Direction を送信する．Route Direction を受信した CN は RS からのパケットを受信可能とするため RS に対して Hole Punch を送信する．MN は UDP によるトンネルを構築するために Tunnel Request を RS を経由し CN に送信する．Tunnel Request を受信した CN は Tunnel Response を RS を経由して MN に返信する．これにより MN と RS 間の UDP トンネルと RS と CN 間の UDP トンネルが構築される．

A.2.3　NTMobile の移動通信シーケンス

図 16 に移動に係る NTMobile の通信シーケンスを示す．図 16 では，カプセル通信を行っている途中で CN がアドレス変化した場合を想定したシーケンスである．NTMobile では，アドレスの変化を検出した場合，DC に実 IP アドレスの登録作業（Registration）を実行した後に再度シグナリングを行うことで，移動先のアドレスでカプセル化された新たなトンネル経路を生成する．

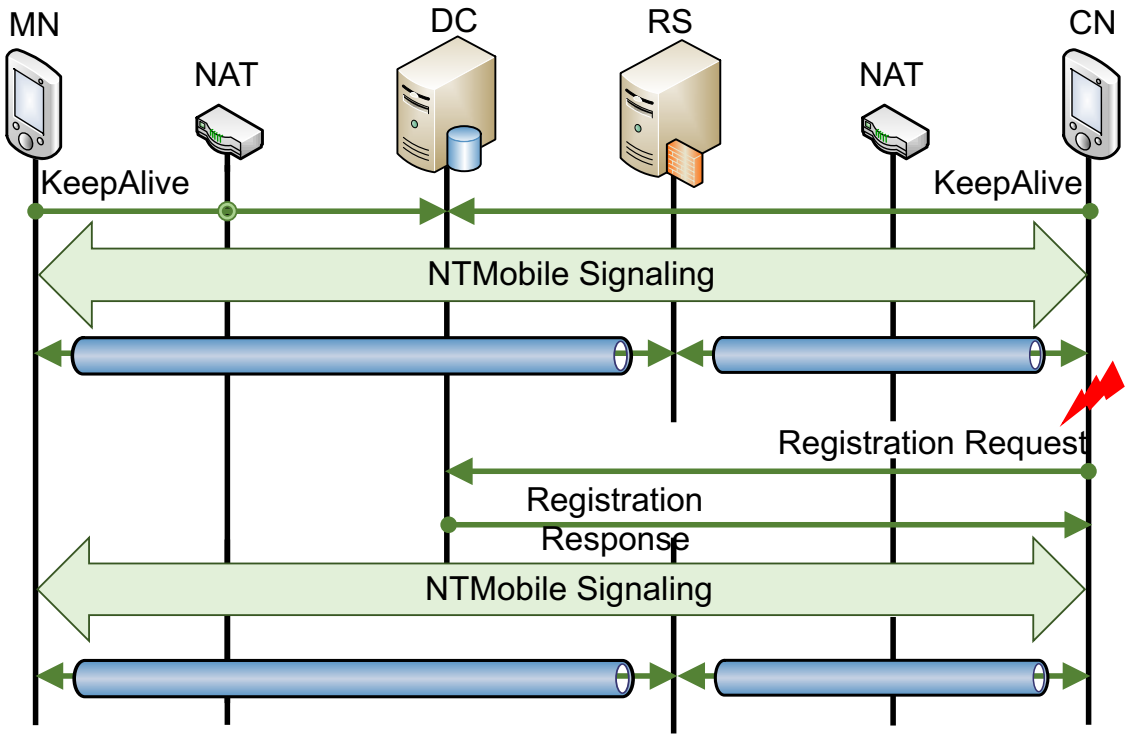

図 16　移動時の NTMobile の通信シーケンス

A.3　提案

本章では，VPNService 型 NTMobile にアドレス変化検出を実現する方法について説明する．

A.3.1　モジュール構成

図 17 にアドレス検出処理を独立させた VPNService 型 NTMobile のモジュール図を示す．

General Application は Android のスマートフォンにインストールされている一般的なアプリケーションである．Tunnel Service は，VPN API である VpnService と NTMobile Framework を一つのアプリケーションとしてプロトタイプ実装である．Tunnel Service では，まず起動時に VpnService が仮想インタフェースの TUN の作成を行う．その後，General Application は TUN インタフェースを経由してパケットの送受信を行う．NTMobile Framework の中に含まれている NTMobile Signaling Module は，Tunnel Service 起動時のアドレス登録処理や NTMobile によるシグナリングを行う．また，Packet Manipulation Module では，NTMobile によるカプセル化パケットの送受信処理を行う．提案方式では，Tunnel Service の中に NTMfw とは独立してアドレス変化検出モジュールを追加した．アドレス変化モジュールは Android のネットワークが切り替わったときに NTMobile Signaling Module にアドレス変化を通知する．その後，通知をトリガとして再度 DC に実 IP アドレスの登録作業（Registration）を実行することで，NTMobile の移動透過性を実現する．

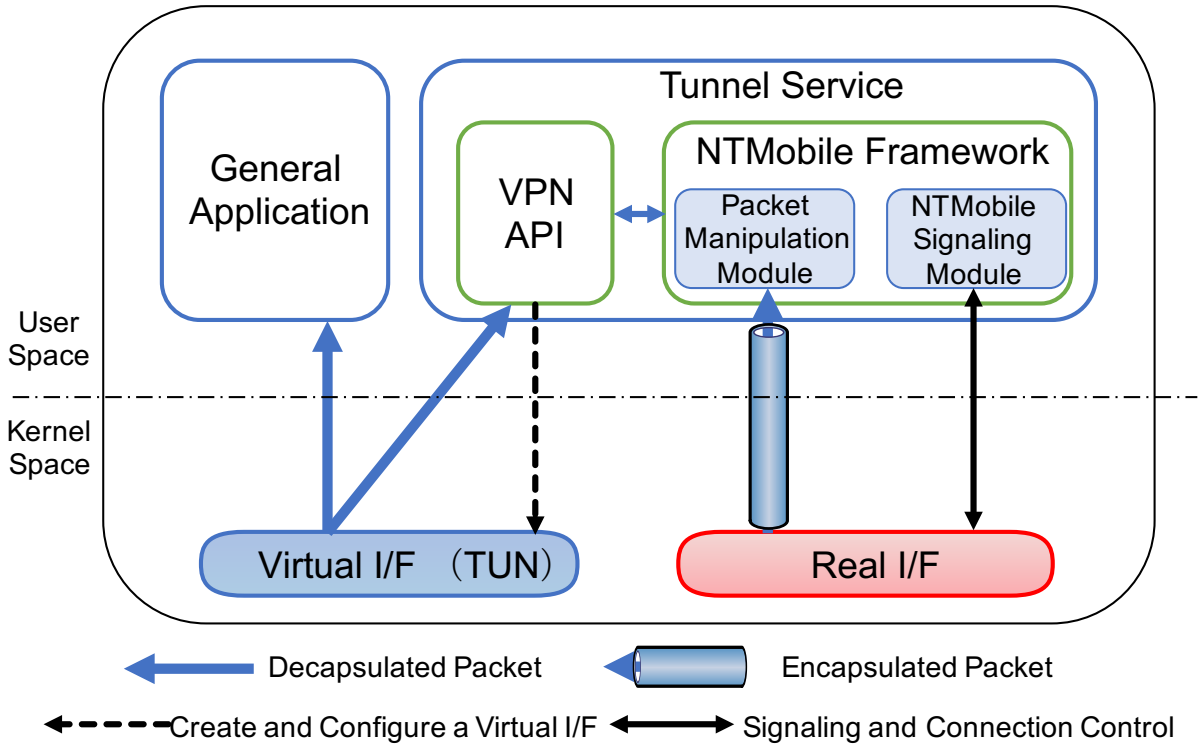

図 17　VPNService 型 NTMobile にアドレス変化検出を実装した際のモジュール図

A.4　実装

A.4.1　Tunnel Service に行った実装

アドレス変化検出は Tunnel Service に Android が提供する Connectivity Manager を使用した [27]. Connectivity Manager は，Android 端末の接続状況が変化すると，CONNECTIVITY_ACTION（“android.net.conn.CONNECTIVITY_CHANGE”）をブロードキャストする．そのため，Tunnel Service に CONNECTIVITY_ACTION を受信するレシーバとして Broadcast Receiver を継承した Connection Receiver を実装した．Tunnel Service は, Connection Receiver が受信した CONNECTIVITY_ACTION をトリガとして NTMfw の Signaling Module ネットワークの変化を通知する．これらの処理は Java によって記述されているが，NTMfw は C で記述されているため JNA（Java Native Access）を経由する必要がある．そのため，NTMfw に Android 用の端末移動時の処理を呼び出す関数を追加し，共通ライブラリの作成を行った．また，CONNECTIVITY_ACTION を受信時に，Java 側から loadLibrary メソッドを利用し NTMfw のハンドオーバー処理を呼び出した．

A.4.2　Connection Receiver の動作

Connection Receiver は Broadcast Receiver を継承させた，CONNECTIVITY_ACTION を受信するレシーバである．Connection Receiver は Wi-Fi に接続を切り替えた時，3G / 4G 通信（セルラー通信）に接続を切り替えた時，端末がオフラインとなった場合にブロードキャストを監視するリスナーとしての役割を持つ．

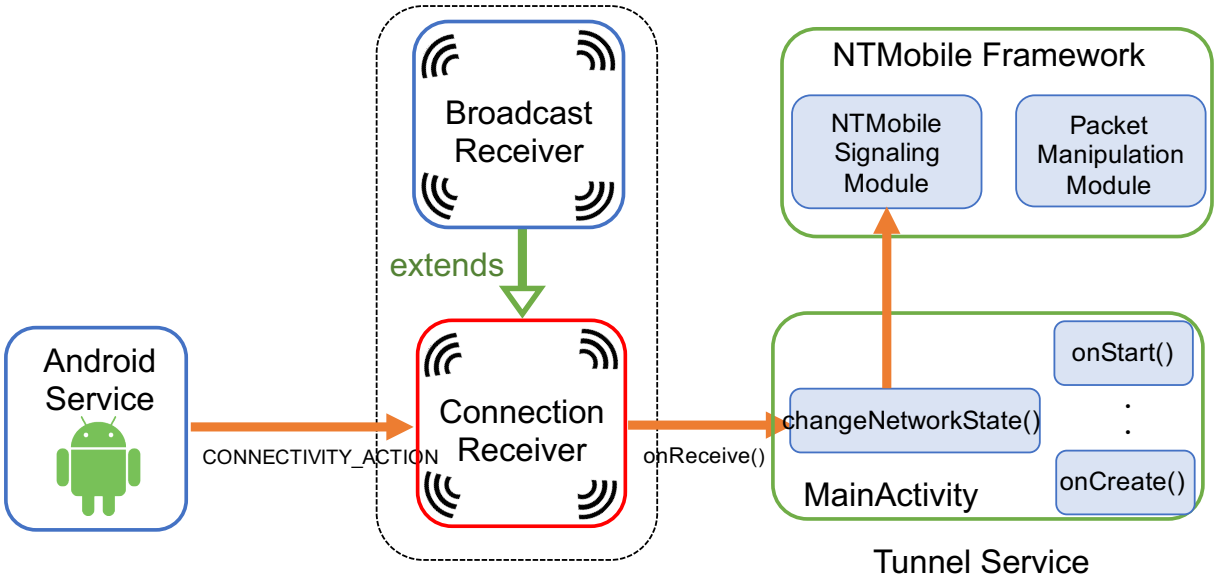

図 18　実装を行ったシステムのモジュール図

A.5　評価

本章では，アドレス変化の検出についての動作検証と性能測定について述べる．

A.5.1　動作検証

Tunnel Service が端末移動時に正しく移動処理を実行できるかを検証するため，Android に Ping を送信させたまま接続先を切り替えた．Ping の送信には，Google Play ストアから Android 向けアプリケーションとして提供されている Network Analyzer を使用した．動作検証の結果，Android 端末が移動した際に通信が途切れず VpnService 型 NTMobile に移動透過性が実現できていることを確認できた．

A.5.2　性能測定

提案方式にて移動処理にかかる時間を測定した．測定で使用した機材の仕様を表 7 に示す．合わせて，表 8 に．使用した Wifi の性能を示す．Wi-Fi1 は NEC Aterm WG2600HG のルータを，Wi-Fi2 は Buffalo WXR-1750DHP のルータを使用している．

図 19 に測定結果を示す．その結果 Wi-Fi から Mobile に通信を切り替える時間は，525[msc] であり，Wi-Fi から Wi-Fi2 に通信切り替える時間は 889[msec] であった．NTMobile のシグナリングに要する時間は 122.85[msec] であった．

表 7　各機材の性能の仕様

	MN, CN	DC, RS
OS	Android 7.1.1	Ubuntu 14.04LTS
CPU	NVIDIA Tegra K1@2.50GHz	Intel(R) Xeon CPU E3-1240 v5@3.50GHz
Memory	2GB	1GB

表 8　各 Wi-FI ルータの性能の仕様

	Wi-Fi(2.4GHz)	Wi-Fi2(5GHz)
暗号化規格	WPA2	WPA2
通信規格	802.11ac	802.11ac
周波数	2.4GHz	5GHz

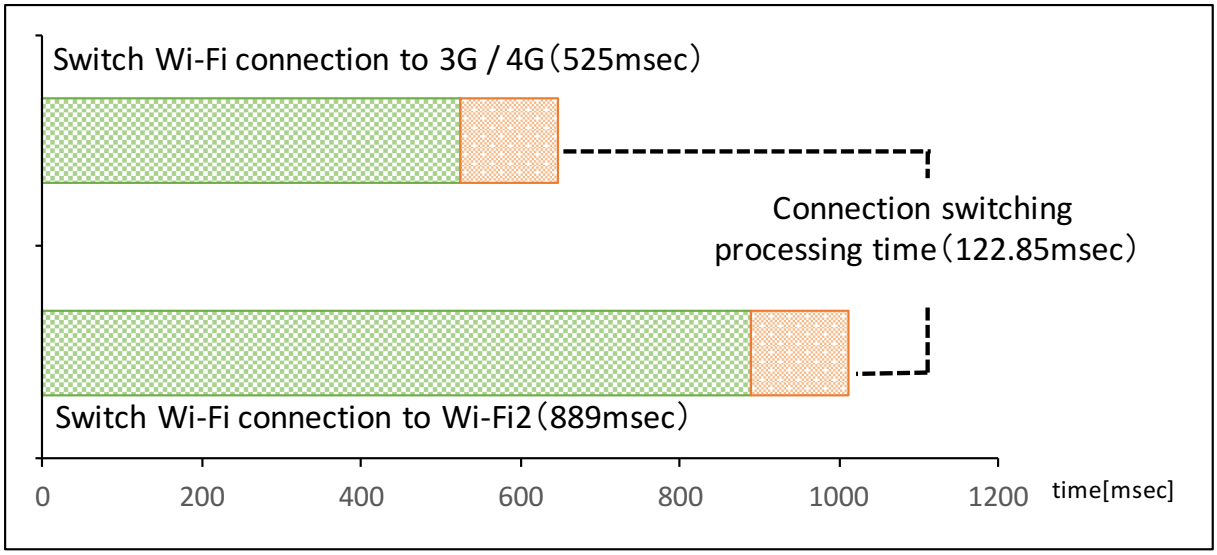

図 19　ネットワーク切り替え処理にかかった時間