2005－OS －98（10）
2005／2／23

異種無線ネットワークにおける透過的な接続を実現する経路制御機構

高橋　ひとみ[†]　斉藤　匡人[†]
間　博人[†]　徳田　英幸[†,††]

近年，爆発的な無線端末の普及により無線ネットワークに多くの形態が出現している．しかし，様々な無線ネットワークは同一の無線メディアで構成され，異種の無線ネットワーク間における即興的な相互接続は考慮されていない．本論文では無線ネットワーク間を相互に接続し，終端端末間で無線メディアを仮想的に同一デバイスとして使用が可能な IWNR (Interchangeable Wireless Network Routing Protocol) を設計・実装した．IWNR は異種無線ネットワークを相互に接続する経路制御機構であり，MANET のプロトコルを利用する．宛て先は複数の無線メディアを所持する端末へ動的な経路を生成し，中間ノードが適切に無線メディアを切り替え，他の無線ネットワークへの接続を可能とする．また，無線メディアが独自に持つユーザへの API を IWNR が提供することで，特定の無線メディアに依存するアプリケーションの動作を可能とする．本機構を Linux カーネル上へ実装し，無線メディアとして 802.11b と Bluetooth を用い実機にて評価した．転送を行う中間ノードのホップ数が増加する毎に，転送性能の劣化，遅延の増加が見られた．

A Mobile Ad Hoc Routing Protocol for Heterogeneous Wireless Networks

Hitomi Takahashi [†], Masato Saito [†], Hiroto Aida [†]
and Hideyuki Tokuda [†,††]

Recently, new systems of a wireless network are appeared as a spreading wireless media. Though many kinds of wireless networks exist, there is no way to connect them to each other. We proposal Interchangeable Wireless Network Routing Protocol (IWNR) to connect different wireless networks to each other. IWNR let user access any wireless networks without considering a kind of wireless media he uses. In IWNR, wireless nodes use technique of MANET and build a route to a node with some wireless media. Additionally, IWNR offers virtual APIs that are specialized in one wireless media. An application depending on a specific API of one wireless media can run with the virtual API on a node without its wireless media actually. We design, implement and evaluate IWNR on Linux kernel for LKM. Using wireless media for this implementation are 802.11b and Bluetooth. Throughput and delay of IWNR become worse with increasing the number of hops for a route.

1. はじめに

近年，無線技術の発達に伴い，より高性能になった無線規格が多く出現している．携帯電話を用い高速なデータ通信を行う規格である IMT-2000，また無線 LAN の規格である，802.11a，802.11b，近距離無線の Bluetooth など，日常生活において目にしない日はないほど無線端末の普及が進んでいる．一方，無線端末の普及と共に機器の小型化が進み，Mote などセンサデバイスを用いたセンサネットワークの研究も進んでいる．

数多くの無線メディアが存在する無線ネットワークでは，Bluetooth，センサデバイス，802.11b の各メディア毎にネットワークが構築され，各無線ネットワークとの相互接続が考慮されていない．そのため，ユーザは宛て先の端末へ通信を試みようとした場合，宛て先が所持する無線メディアを特定し，同一の無線メディアによって通信をする必要がある．つまりユーザが通信に用いる無線メディアを意識しなければならず，数多く無線メディアが存在する今日では，ユーザへの利便性が低下する．

ユーザの利便性を考慮すると，それぞれのデバイス，ネットワークの境界をなくし，希望する端末へ透過的に接続できる新たな無線ネットワーク体系が必要である．無線メディアの異なる端末同士が集まった場合，ユーザが終端端末間において無線メディアを仮想的に同一に見せることで，透過的な無線ネットワークを構築できる．そこで，本論文では無線メディアを仮想的に同一に見せる機構の具体的な実現手法として，経路制御を使用した，IWNR(Interchangeable Wireless Network Routing Protocol) を提案する．IWNR は MANET 技術を用い，異種無線ネットワーク間を相互に接続する経路制御プロトコルである．本プロトコルを用いることで，ユーザは無線デバイスの差違を意識せずにネットワークへの接続が可能になる．

本論文の構成はとして，まず第 1 節で研究背景である無線端末の普及について述べ，第 2 節で無線ネットワークに対する問題を指摘し，指摘した問題を解決する方法

† 慶應義塾大学大学院 政策・メディア研究科
Graduate School of Media and Governance, Keio University
†† 慶應義塾大学 環境情報学部
Faculty of Environmental Information, Keio University
本研究は文部科学省科学技術振興調整費「人間支援のための分散リアルタイムネットワーク基盤技術の研究」，総務省「ユビキタスネットワーク制御・管理技術の研究開発 (ubila プロジェクト)」の下に行われています

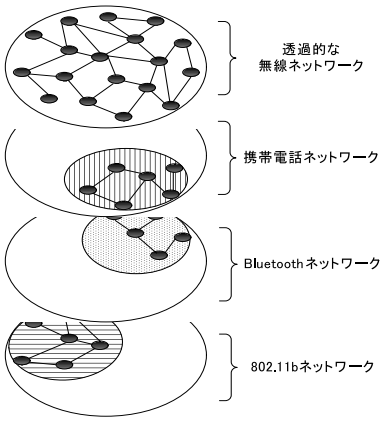

図 1 新たな無線ネットワーク

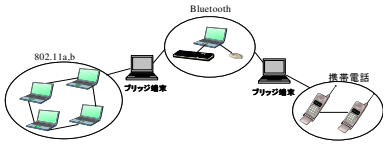

図 2 ブリッジ端末

を提案する．次に，第 3 節で提案した手法を用いる関連研究を述べ，第 4 節で終端端末間の無線メディアを仮想的に同一にみせる IWNR の設計，第 5 節で IWNR の実装について説明する．最後に，第 6 節で実機にて IWNR を動作させ評価の結果を報告し，第 7 節で今後の課題について述べる．

2. 無線ネットワークの問題

本節では，無線ネットワークに対する問題を指摘し，その問題を解決する手法を述べる．

既存の無線ネットワークが持つ問題の大きな焦点は，多種多様な無線メディアが混在する既存の無線ネットワークにおいて，個々の無線ネットワーク同士の相互接続が不可能な点である．そのため既存の無線ネットワークでは，Bluetooth，802.11b の無線ネットワークなど，無線メディアによってネットワーク毎に壁が存在することで，ユーザは端末が所持する無線メディアを意識しなければならない．ユーザが端末の無線メディアを意識せず，大きな枠組の無線ネットワークとして，ネットワークの使用が可能となることはユーザにとって利便性が高くなり，透過的に各無線ネットワークが相互接続されることで，個々の無線ネットワークが持つ問題も解決される．既存の無線ネットワークが持つ問題を解決するためには，個々の無線ネットワークを抽象化し，無線ネットワークという大きな枠組みでの新しいネットワークを構築する必要がある．

無線ネットワークを抽象化した透過的なネットワークを構築する手法として，送信元と宛て先において仮想的に無線メディアを同一デバイスとして使用する方法が挙げられる．この手法を用い構築された無線ネットワークは，図 1 に示すように Bluetooth，802.11b，携帯電話のネットワークなど，個々の無線ネットワークを包括する．この透過的な無線ネットワーク上では，あるユーザからは Bluetooth のネットワーク，他のユーザからは携帯電話のネットワークとしてネットワークが使用可能となる．

仮想的に無線メディアを同一デバイスとして使用可能とする，具体的なモデルはいくつか挙げられる．例えば，無線によるオーバーレイネットワークを構築し，そのネットワーク上でアプリケーションを動作させるモデルや，全無線メディアに共通する普遍的な API を新しく提案し，その API を使用しアプリケーションを動作させる手法などが考えられる．しかし，これらのモデルは既存アプリケーションの変更が必要となってしまう．ユーザが自分の所持している無線メディアや宛て先の無線メディアを意識させないためには，既存のアプリケーションが変更なしで，全ての端末においても動作可能でなければならない．そこで，本論文は仮想的に無線メディアを同一デバイスとして使用可能とする実現モデルとして，経路制御 (IWNR:Interchangeable Wireless Network Routing Protocol) による手法を提案する．

IWNR において，各無線ネットワークとの相互接続を可能にするためには図 2 に示す，各無線ネットワークとのブリッジの役割を果たす端末 (ブリッジ端末) が必要不可欠である．このブリッジ端末はネットワークの境界に接し，それぞれの無線メディアを複数所持することで，各無線ネットワークへデータの配送を行う．しかし，各無線ネットワークへデータを転送するだけでは，ユーザへ透過的な接続は提供できない．例えば，小規模なミーティングがあった場合，無線デバイス A を所持する端末のデータを，無線デバイス B のみを搭載している無線端末を持つユーザへ転送する場合，現状では，無線デバイス A，B の両方を搭載する端末を所持するユーザが，無線デバイス A を用い配布データを取得し，デバイス B を用いユーザへ配布する．この一連の作業では，ユーザがデバイスを意識しなければならず，煩雑な手続きが必要となってしまう．各ユーザが無線デバイスを意識せず，各無線ネットワークへ透過的に接続を行い，下位のレイヤではブリッジ端末への経路を動的に構築するフレームワークが必要である．

そこで IWNR は，MANET の技術を用いブリッジ端末まで動的に経路を構築し，適切なインタフェースに切り替え通信を行うことで，透過的な接続を可能にする．

3. 関 連 研 究

本節は IWNR のように異種ネットワークを相互接続する機構の関連研究について述べる．また，本論文は IWNR を実機にて実装し経路制御の実装に関して重視している．そこで，MANET の経路制御プロトコルの実装に関する既存研究も述べる．

異なるネットワークを透過的に接続する関連研究として，H.Lou らが提案する UCAN(A Unified Cellular and Ad-hoc Network Architecture)[1] が挙げられる．UCAN は，802.11b の MANET と携帯電話のセルラーネットワークとの相互接続を行い，直接基地局まで携帯電話の電波が届かない場合，MANET によって構築したネットワークを利用し携帯電話と基地局までの通信を可能とする．ただし，802.11b と携帯電話の混在した環境ではなく，宛て先から携帯電話の基地局までを 802.11b の MANET で構築し，基地局までデータ転送を行うものである．この経路制御では終端端末のみが携帯電話となり，本論文で提案する異種無線ネットワーク間の透過的な接続は実現できない．

MANET における実装のアプローチに焦点を当てた関連研究として以下の 3 つが挙げられる．まず，V.Kawadia ら [2] はカーネルへ独自に実装してある経路情報やパケットの転送を制御する関数の API として ASL (Ad-hoc Support Library) を提供し，その API を用い，ユーザ空間

にデーモンとして MANET のプロトコルを実装できるフレームワークを述べている．実際の実装では，ASL を用いて AODV[3] をユーザアプリケーションのデーモンとして実装している．この論文では多種多様な OS への依存を最小にした MANET プロトコルの実装が目的である．しかし IP を用いた MANET の構築を想定しているため，IP を用いない無線メディアは想定していない．次に I.Chakeres ら[4] は，AODV をユーザアプリケーションのデーモンとして実装している．この実装では Linux カーネルの機構である Netfilter を用いパケットを取得しているため，カーネルの変更は必要ない．ただし，Netfilter は第 3 層の IP プロトコルのみで使用されており，IP を使用しないデバイスは考慮されていない．D.Maltz[5] は Monarch で提案された DSR[6] を実機にて実装を初めて行った論文であり，DSR をカーネル内の第 3 層に追加し実装を行っている．この実装方法は IPv4 に特化した実装となっており，上位層および下位層の汎用性が考慮されていない．

4. IWNR の設計

IWNR (Interchangeable Wireless Network Routing Protocol) は MANET の技術を応用し，ユーザが無線メディアを意識せずに無線ネットワークを透過的に接続するための機構である．この機構はユーザに様々な無線メディアで構築されるネットワークを，透過的なネットワークとして提供できる．そのため，ユーザは宛て先が実際に搭載している無線メディアを考慮することなく，ユーザの希望する無線メディアの端末として宛て先と通信が可能である．ただし，送信元のユーザは，宛て先のアドレスを予め知っている必要がある．

IWNR では図 3 に示すように，無線メディアとして IF1 のみを所持する端末である送信元が，無線メディアとして IF2 を所持する宛て先までデータ通信を行なうとする．送信元は IF1，IF2 の無線メディアを両方所持するブリッジ端末まで動的に経路を構築し，データの宛て先により IF1，IF2 を適切に切り替えることで，異種無線ネットワークへ透過的な接続を提供できる．

IWNR は MANET の技術を用いて，経路の発見およびデータの転送を行い，端末の移動も考慮するため，リンクが途切れた場合における経路の再構築を行う．また，適切なインタフェースの切り替えを行う機構も有する．

ただし IWNR は既存の MANET と異なり，多くの無線ネットワークを透過的に接続するため，既存の MANET では考慮されていない問題も発生してくる．そこで，それらの問題を解決する新たな機構を設けないとならない．以下に具体的なシナリオと共に，必要となる新たな機能に関して述べる．

4.1 IWNR のシナリオモデル

スタック改変が不可能な端末

既存の MANET は参加端末として高性能な無線端末であるラップトップや PDA などを想定し，端末に搭載されるカーネルの改変が可能である，もしくは作成したアプリケーションの動作が可能であることが前提であった．しかし Bluetooth などの小型な近距離無線メディアは，デジタルカメラや携帯電話など組み込み機器に搭載されている場合が多く，カーネルの改変や作成したアプリケーションを自由に実行できない．

作成したプログラムが端末上で動作不可能であれば，IWNR が構築するネットワーク (IWNR ネットワーク) に参加できず，ユーザの利便性が低下すると考えられる．

図 3 ブリッジ端末によるインタフェース切り替え

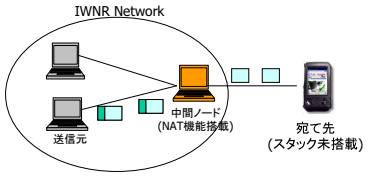

図 4 スタック未搭載端末による IWNR ネットワークへの接続

そこでソフトウェアおよびカーネルの改変なしで，端末が IWNR ネットワークに参加できるよう，IWNR では図 4 に示す NAT 機能を有する．図 4 では IWNR ネットワークと IWNR が未搭載の端末である宛て先を接続する，中間ノードが存在する．中間ノードは転送先が IWNR スタックが未搭載であり，かつ転送元が IWNR スタックを搭載した端末である場合，IWNR のヘッダをカプセリング化，および，アドレス変換を行う．この中間ノードの NAT 機能により，ソフトウェアが改変できない機器端末でも IWNR ネットワークへ参加できる．

デバイス依存なアプリケーション

いくつかのネットワークデバイスは独自の API を提供しており，独自の API を利用し通信を行うアプリケーションはその API が利用できない場合，正常に動作しない．

例えば，Bluetooth のスタックである Bluez の API を考えた場合，IPv4 を使用するアプリケーションは，socket システムコールのドメインとして PF_INET を呼び出すが，Bluetooth 依存のアプリケーションはドメインとして，PF_BLUETOOTH を呼び出す．socket のドメインを PF_BLUETOOTH で呼び出す Bluetooth 依存のアプリケーションは，Bluetooth が搭載されてない端末上では動作しない．そのような状況下で IWNR を利用し Bluetooth メディアのみを搭載した送信元と，802.11b のみを搭載した宛て先への接続を行なうとする．送信元のアプリケーションが Bluetooth 依存であり PF_BLUETOOTH を呼び出すプログラムであった場合，送信元で動作しているアプリケーションは Bluetooth が未搭載な宛て先では動作できない．そこで IWNR は無線デバイス独自が提供している API を IWNR が仮想的に提供することで，ユーザは無線デバイスを意識せずにアプリケーションの使用が可能となる．

4.2 IWNR の必要機能

前節で述べたシナリオモデルを考慮した際，IWNR に必要な機能は，スタック改変が不可能な端末を接続するための NAT 機能，無線デバイスが提供する独自の API を提供する機能となる．もちろんアドホックネットワークを構築する機能も，IWNR では必要となる．以上を踏まえた上で IWNR の機能要件は以下の項目となる．

(1) MANET ルーティング機能 (経路発見，データ転送，経路維持)
(2) 無線デバイス切り替え機能
(3) NAT 機能
(4) 無線メディア独自 API の提供

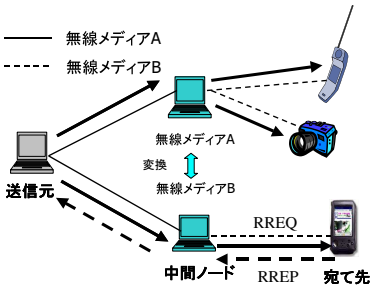

図 5　IWNR の経路発見

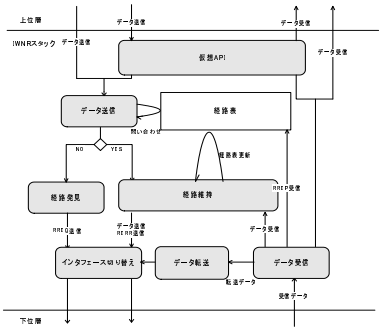

図 6　IWNR のモジュール構成図

4.3　動 作 手 順

本小節では IWNR の動作手順について述べる．IWNR の動作手順は，経路制御のアルゴリズムとして何を採用するかにより変化する．本論文では MANET のルーティングプロトコルとして，DSR をもとにした簡単なルーティングアルゴリズムを用いる．

経 路 発 見

図 5 に IWNR の経路発見手順を示す．まず送信元は宛て先までの経路が判明しない場合，経路要求パケット (RREQ) をブロードキャストで送信し，ネットワーク全ての端末に対し送信する．RREQ を受信した端末は RREQ の経路情報，インタフェース情報を経路表に格納し，RREQ をブロードキャストにて他の端末へ転送する．もし RREQ を受信し，かつ無線メディアを複数所持する端末は全ての無線メディアに対して，RREQ を転送する．各端末は RREQ を転送時に自分のアドレス，転送に使用する無線インタフェースの情報を RREQ に格納する．

宛て先に指定された端末が RREQ を受信した場合，送信元へ経路応答パケット (RREP) を送信する．宛て先は RREQ より送信元への転送先の端末，インタフェースを判別できるため，ユニキャストにて RREP を送信元へ送信する．複数無線メディアを所持する中間ノードは，RREP の情報を端末が保持する経路表に登録する．その後，中間ノードは適切な無線メディアを使用し，RREP を送信元へ転送する．RREP を受信した送信元は，RREP に格納された情報をもとに宛て先へのデータ通信を開始する．

経 路 維 持

IWNR が提供するネットワークは無線端末の移動を前提としており，経路発見後端末の移動により通信に使用していた経路の崩壊が予想される．その際，経路上の転送端末は経路の途切れを検知し，送信元へ再び宛て先への経路発見を行なうよう，通知しなくてはならない．端末が転送データを受信した場合，転送元の端末に対し IWNR 独自の ACK を送信する．もし，転送元が転送先からの ACK を受信できない場合，転送先にデータの再送を行ない，ある一定の回数の再送が失敗した場合そのデータを破棄し，使用経路が途切れた事を検知する．その際，転送元は途切れた経路の転送先とは逆方向へ，途切れた経路の情報を格納した経路エラーパケット (RERR) を送信する．RERR を受信した全端末はその経路を含む経路表に格納された経路を全て削除する．

アドレス変換

IWNR ではスタックの改変ができない端末もネットワークへ接続できるよう，端末はアドレスの変換，パケットのカプセル化を行なう．IWNR が動作する端末 (IWNR 端末) が RREQ を転送後，IWNR 端末は各メディア毎に IWNR を使用せず，もとの無線メディアのスタックへ RREQ の宛て先アドレスを宛て先として，パケットの送信を試みる．もし IWNR スタック内で近隣端末と認識していない転送端末から返答があった場合，IWNR スタックを所持していない端末として扱われる．IWNR ネットワークと，IWNR を搭載していない端末との中間に存在する IWNR 端末は，アドレスの変換やパケットのカプセル化を行ない，IWNR 未搭載端末との接続を行なう．

4.4　IWNR モジュール

本小節では IWNR の機能モジュールを詳細に説明する．IWNR モジュール全体の構成図を図 6 に示す．

上位層から送信要求があったデータは，IWNR が提供するデータ送信モジュールへ送られる．またデータ送信を行うアプリケーションによっては，仮想 API 通じてデータ送信モジュールへデータが送られる．データ送信モジュールは，各端末が保持する IWNR スタック内の経路表に宛て先までの経路が存在するか問い合わせる．もし宛て先までの経路がなければ経路発見モジュールを呼び出し，経路要求 (RREQ) を送信する．宛て先までの経路が経路表に存在すれば，送信データを経路維持モジュールへ登録し IWNR ヘッダを付随させデータ送信を行う．IWNR スタックから送信されるデータは必ず，インタフェース切り替えモジュールを通過する．このモジュールは適切なインタフェースへデータを送信する機能を持つ．

IWNR スタックが下位層からデータを受信した場合，データ受信モジュールにデータが渡される．このモジュールは RREQ に対する経路応答 (RREP) や経路エラー (RERR) を受信した場合，経路表へ RREP，RERR の経路情報を更新する．またデータの転送が必要な場合は，データ転送モジュールへ転送データを渡し，経路維持を行なうため転送データ，受信データの情報を経路維持モジュールへ渡す．経路維持モジュールは，渡された情報をもとに明示的にデータを転送してきた端末へ ACK を返す，またデータの転送が失敗した場合，転送データの再送を行う．もし使用していた経路が途切れた場合は，経路維持モジュールより RERR の送信を行ない，経路情報を更新する．受信したデータが自分宛てのデータパケットである場合，IWNR の独自ヘッダを削除し適切なプロトコルスタックもしくは，仮想 API モジュールへデータを渡す．また，IWNR スタックは複数の無線メディアを想定したプロトコルスタックとなる．そこで無線メディアが使用できるアドレスとの依存を避けるため，IWNR スタック内では独自のアドレスを用い，端末の認識を行なう．

5.　IWNR の実装

本節では IWNR の実装環境および実装について述べ

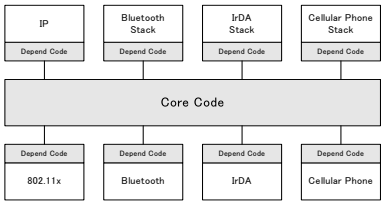

図 7 IWNR における実装のポリシ

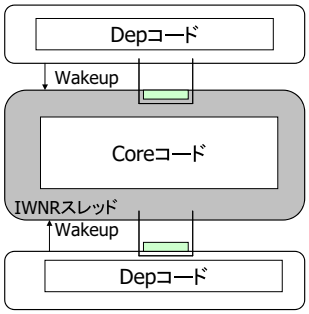

図 8 IWNR のカーネルスレッド

る．IWNR は多くの無線メディアに対応させるため，無線メディアやスタックに依存する部分がなるべく少なくなるよう実装されている．

5.1 実装環境

IWNR の実装では OS として，Linux-2.4.20 を使用する．今回用いる Linux カーネルは，動的にカーネル機能を削除，追加できる機構である LKM (Loadable Kernel Module) をサポートしてる．本機構ではユーザが簡単に本機構の使用を可能にするため，LKM を用い，実装を行う．

使用する無線メディアとして，802.11b，Bluetooth を使用する．Bluetooth は通常の TCP/IP プロトコルを用いず，独自のスタックを使用し動作している．本実装では Bluetooth プロトコルスタックとして，Linux で標準搭載されている Bluez-2.3 を用いる．Bluez は LKM として実装されているため，Bluez 自体に大幅な改良が可能である．将来的には今回実装に用いた 802.11b, Bluetooth のみだけではなく，多くの無線メディアも IWNR へ対応する予定である．

また本実装では前節で示した IWNR モジュールのうち，経路維持モジュール以外のモジュールを実装した．そのため，経路維持モジュールの実装説明は省略する．

5.2 実装方針

IWNR は将来的に多くの無線メディアの対応を予定している．新たな無線メディアをサポートする場合に，新たな無線メディアのために，IWNR の大幅なコードの追加や変更は余り望ましくない．

新たな無線メディアを IWNR でサポートする場合，無線メディアに依存する部分であるごく一部のコードを追加すれば，IWNR へ対応できるよう IWNR を実装すべきである．そこで IWNR の実装方針として，図 7 に示すように無線メディアに依存する部分 (Depend Code) のコード (Dep コード) と，無線メディアに依存せずに使用でき，IWNR 機能において中核となる部分 (Core Code) のコード (Core コード) を明確に分け，なるべく，Dep コードのコード量が少なくなるよう実装を行う．この実装方針により，新たな無線メディアのサポートをする際に無線メディアに特化した Dep コードを IWNR へ新たに追加することで簡単にサポートできる．

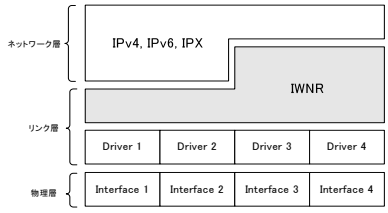

図 9 無線 LAN における IWNR の実装

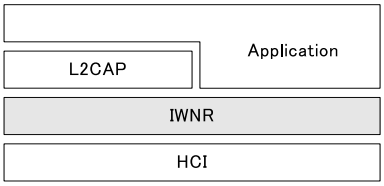

図 10 Bluetooth における IWNR の実装

また，図 8 で示すように Core コードが動作するプロセスは，IWNR 独自のカーネルスレッドで動作している．上位，下位層から渡されるデータは全てキューを使用し Core コードのプロセスに渡される．

5.3 802.11b における実装

802.11b の実装では，IWNR スタックが図 9 で示すようにリンク層とネットワーク層に位置する．送信時は IWNR スタックは仮想デバイスドライバとして存在するためリンク層となる．リンク層に実装することで，IWNR スタックの上位層であるネットワーク層で使用されるプロトコルによらず IWNR が実装できるため，リンク層に実装を行った．また受信時ではイーサヘッダにあるプロトコル番号により，IWNR スタックにデマルチプレクスされるため，ネットワーク層に存在することになる．

802.11b における IWNR は仮想デバイスドライバとして認識されているため，通常の Unix コマンドである ifconfig によるイーサデバイス一覧で表示される．今回の実装では上位層に IPv4 のみを用いることを前提としているため，各端末が所持する IWNR アドレスは，通常の IP アドレスとして用いられる．最終的には，IP アドレスではなく MAC アドレスより 32bit の IWNR アドレスを自動生成することを想定している．

5.4 Bluetooth における実装

Bluetooth の実装では，IWNR スタックが図 9 で示す様に Host Control Interface (HCI) と L2CAP の中間に存在する．L2CAP から送信されるデータは HCI 層に送信する以前に IWNR スタックを通過し，HCI 層で受信した ACL データは全て IWNR スタックに受信される．また IWNR は Bluez 上に存在する IWNR スタックとは別に，Bluetooth のデバイスを所持していない場合でも Bluetooth の API に依存するアプリケーションが使用できる．

6. IWNR の評価

本節では IWNR の評価について述べる．まず，実機にてマルチホップの環境を構築し，転送性能と遅延を測定した．次に経路発見時間，IWNR スタックの処理時間を測定し，IWNR のオーバヘッドについて考察した．

6.1 評価環境

本小節では IWNR の評価環境について述べる．IWNR を RedHat Linux-2.4.20 に LKM として実装し，Blue-

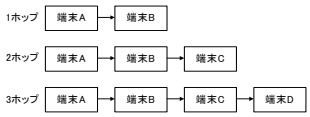

図 11 実験環境

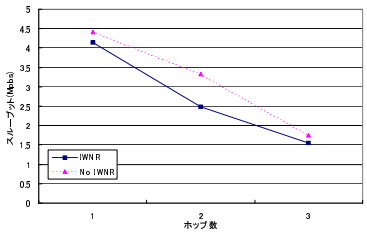

図 12 802.11b におけるスループット

tooth および 802.11b を無線メディアとして用い評価を行った．ただし，経路維持モジュールは実装していないため，IWNR 層によるデータ転送毎の ACK は返していない．

評価環境として実機へ IWNR をインストールし，アドホックネットワークを構築し測定を行った．測定を行う実験環境は図 11 に示すように，端末 A を送信元として固定し，1 ホップでは端末 A，B，2 ホップでは端末 A，B，C，3 ホップでは端末 A，B，C，D の経路を構築した．また，Bluetooth のリンクを含める評価の場合，端末 A，B 間のリンクのみを Bluetooth とした．

マルチホップ経路の構築では，送信元と宛て先がお互いの電波を送受信できない距離に端末を設置したのではなく，指定した MAC アドレスのパケットを破棄するコードをカーネル内に追加し，故意的にマルチホップの経路を構築している．そのため，端末 A，B，C，D は，実際ならば全端末のパケットを受信できてしまう距離で密集しており，評価へ無線電波の干渉による影響が現れると考えられる．

6.2 スループット

本小節では IWNR を用いて転送性能の評価を行った．まず，802.11b のみの無線メディアを用い TCP スループットの測定を行った．次に 802.11b と Bluetooth を用い，Bluez のツールとして公開されている l2test プログラム [7] を用い，L2CAP 上で転送性能の評価を行った．

802.11b におけるスループット

802.11b のみを無線メディアとして使用し，端末 A を送信元とし，経路が決定した後，1 から 3 ホップまでの TCP スループットを測定した．NetPerf[8] を使用し，TCP データの送受信を 10 秒間，10 回繰り返したスループット平均値を図 12 に示す．

図 12 の横軸はホップ数を，縦軸は TCP スループット (Mbps) を示している．IWNR は IWNR スタックを用い，マルチホップの経路を構築し測定を行った値であり，No IWNR は経路表を手動で書き換え，静的にマルチホップの経路を構築したネットワークで測定した値である．

No IWNR，IWNR ともにホップ数が増加する毎にスループットが減少している．2 ホップの状況で No IWNR と IWNR におけるスループットの差が一番大きく，IWNR は 0.8Mbps 劣化している．しかし，No IWNR とのスループットを比べ IWNR のスループットは平均 0.86 倍ほどしか劣化していないため，それほど IWNR の負荷は高くないと考えられる．

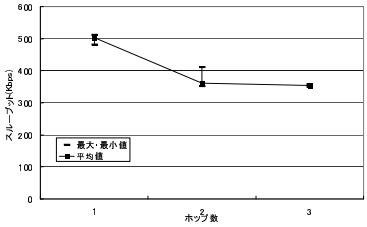

図 13 Bluetooth，802.11b におけるスループット

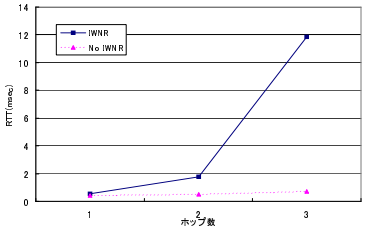

図 14 802.11b における RTT

802.11b と Bluetooth におけるスループット

次に 802.11b の端末で Bluetooth のデバイスに依存するアプリケーションである l2test を用い，Bluetooth と 802.11b 上でスループットを測定した．802.11b の端末では IWNR が提供する Bluetooth の仮想 API を用いることで，実際に Bluetooth を搭載していない端末上でも，Bluetooth に依存するアプリケーションを使用できる．使用無線メディアは，端末 A，B 間のリンクのみ Bluetooth となり，端末 A，B 間以外のリンクは 802.11b となる．

図 13 は l2test を用い，10 秒間の送受信を 10 回繰り返した最大値，最小値，平均値を示している．縦軸はスループット (kbps) を，横軸はホップ数を示している．1 ホップである場合は Bluetooth のみの経路となり，2 ホップ以上の経路は初めの 1 ホップ以外は全て 802.11b で構築される経路となる．1 ホップから 2 ホップにかけて，143kbps と大きく減少している．2 ホップの測定では，中間ノードが Bluetooth から 802.11b への無線メディアの切り替えや転送処理の負荷が原因でスループットが低下していると考えられる．また 2 ホップから 3 ホップへのスループット減少が 802.11b のみの測定と比べ低いことが分かる．原因として，Bluetooth の帯域は最大 1Mbps であり，802.11b は 11Mbps であるため，Bluetooth のリンクがボトルネックとなり，802.11b のみの経路を増加させた場合でも，スループットに対する影響が少ないと考えられる．

6.3 遅延に関する評価

本小節では IWNR を用い経路における遅延の評価を行った．まず 802.11b のみの無線メディアを用いた RTT の測定を行ない，次に Bluetooth と 802.11b を用い RTT の計測を行なった．ただし全計測ともに，経路発見後に RTT の計測を行ったため，経路発見およびコネクション確立にかかる時間は含まれていない．

802.11b における遅延

この小節では 802.11b のみの無線メディアを使用し，前節と同様に端末 A を送信元とし，1 から 3 ホップまでの RTT を計測した．計測プログラムには ping を用い，50 回の平均 RTT を測定した．図 14 に結果を示す．

図 14 に示すグラフの横軸はホップ数を，縦軸は RTT (ミリ秒) を示している．IWNR は IWNR スタックを用

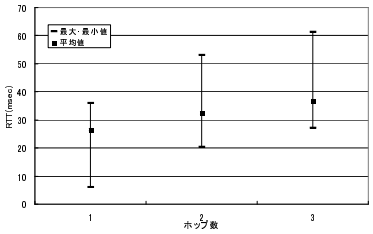

図 15 Bluetooth におけるホップ数と RTT の関係

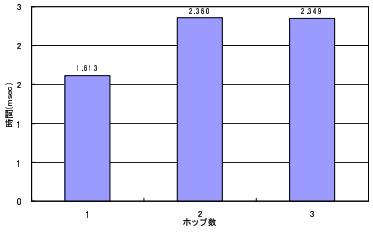

図 16 802.11b における経路発見時間

い計測した結果を，No IWNR は IWNR を用いず静的に経路表を書き換えマルチホップ環境を構築し，測定した結果である．

IWNR，No IWNR ともにホップ数が増加する毎に遅延が大きくなる．特に IWNR では，2 ホップから 3 ホップへの RTT 増加が著しく，10 ミリ秒差の遅延が発生し，No IWNR と 16.6 倍の差である．この原因は第 6.5 節のオーバヘッドにおいて説明を行う．

802.11b と Bluetooth における遅延

次に端末 A，B 間のリンクを Bluetooth し，他のリンクを 802.11b として経路を構築した．そのため 1 ホップの経路は Bluetooth，1 ホップ以外の経路は 802.11b の経路となる．測定プログラムとして l2ping プログラム [9] を用い，RTT の測定を行った．l2ping は通常の ping プログラムと同様に L2CAP 層に echo 要求を送信し，echo 要求を受信した端末は echo 応答を送信元へ返答するプログラムである．この l2ping を使用し，RTT の計測を 50 回行った．その結果を図 15 に示す．横軸はホップ数を，縦軸は RTT(ミリ秒) を示しており，グラフの値は RTT の最大，最小，平均値を表している．

802.11b のみ経路と同様，ホップ数が増加する毎に RTT が大きくなる．1 ホップから 2 ホップにおける RTT の差は 6 ミリ秒，2 ホップから 3 ホップにおける RTT の差は 3 ミリ秒と増加しているが，図 13 と同様に，Bluetooth のリンクがボトルネックとなってしまうため，802.11b のリンクが追加された場合でも大幅な RTT の増加は見られなかった．

6.4 経路発見時間

この小節では，経路発見時間の評価を行った．測定方法は rdtsc 命令を用い，CPU カウンタを表示させ測定を行った．まず 802.11b のみの経路を構築し経路のホップ数を増加させ，送信元である端末 A が経路要求を送信し，経路応答を受信するまでの時間を 10 回測定した．その結果の平均値を図 16 に示す．

図 16 の横軸はホップ数を，縦軸は RREQ を送信後 RREP を受信するまでの時間 (ミリ秒) を示している．1 ホップにおける経路発見時間は 1.61 ミリ秒であるが，2，3 ホップでは経路発見の時間が 1 ホップと比べ大きくなっている．2，3 ホップでは 2.3 ミリ秒となっており，1 ホップの経路発見時間と比較すると 0.7 ミリ秒差と時間がかかってしまっている．

表 1 Bluetooth における経路発見時間

処理	時間 (秒)	割合
inquiry 要求応答	9.34	0.89
接続要求応答	1.13	0.11
経路要求応答	0.05	0.005
合計	10.53	1

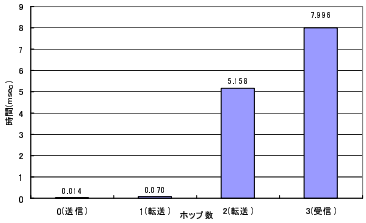

図 17 3 ホップ時のスタック処理時間

次に，Bluetooth における経路発見時間の測定を行った．Bluetooth の測定は Bluetooth のリンク 1 ホップのみで計測した．Bluetooth はコネクション型の無線デバイスのため，隣接ノード全てにデータを配送するブロードキャストができない．そこで IWNR は周囲の Bluetooth を探し，発見した Bluetooth 端末それぞれに接続を確立し，RREQ の送信を行っている．

Bluetooth における経路発見時間を 10 回計測しその平均値を表 1 に示す．inquiry 要求応答時間は，送信元が周囲の Bluetooth 端末を検索するために HCI 層で inquiry を送信し，その要求を受信した Bluetooth 端末から自分の Bluetooth アドレスおよび提供するサービスの応答を受信するまでの時間である．接続要求応答は隣接端末より Bluetooth アドレスを取得後，HCI 層でコネクションの要求を送信し，宛て先から応答があり，コネクションを確立するまでの時間を示す．経路要求応答は宛て先へ HCI 層での接続確立後，IWNR における RREQ を送信し，宛て先からの RREP を受信するまでの時間を示している．

宛て先が inquiry の送信から RREP を受信するまで約 11 秒もかかってしまっている．しかし，その時間の割合は inquiry の要求応答が 0.83，コネクション確立が 0.16 と，ほぼ Bluetooth のデバイスの挙動によって多く占められてしまっている．IWNR スタック自体の処理時間が占める割合は 0.01 と少ない．送信の要求が IWNR へ渡され，実際にデータが送信されるため 11 秒もかかってしまうため，アプリケーションによってはタイムアウトを発生してしまうものもあった．

6.5 スタックの処理時間

この小節では，IWNR スタック処理時間の測定および解析を行った．前節の遅延測定では，図 14 のグラフが示すように 3 ホップのにおける RTT の増分が際だっていた．そこで本小節では 3 ホップ時の IWNR スタック処理時間に焦点を絞り評価を行った．図 17 に結果を示す

図 17 は 3 ホップの経路構築後，ping を行い rdtsc 命令により IWNR のスタック処理時間を 10 回測定した平均値を示している．グラフの縦軸は IWNR スタックにパケットが渡され，下位層および上位層にパケットを送信するまでの処理時間 (ミリ秒) を表しており，横軸は各端末における IWNR の処理を示している．送信元である 0 ホップの端末は送信処理となり，2，3 ホップの端末は転送処理，宛て先端末である 3 ホップは受信処理の時間と

表 2　3 ホップ経路時の受信処理時間

処理	時間 (ミリ秒)	割合
デバイスからの受信処理	0.00189	0.0002
スレッド起床	**7.171873**	**0.999**
キューからのデータ取得	0.00169	0.0002
ヘッダ処理	0.003	0.0004
合計	7.17853	1

なっている．グラフでは，2 ホップ目の転送処理が 5 ミリ秒，3 ホップの受信処理に約 8 ミリの時間がかかっており，主なオーバヘッドとして受信処理が大きな割合を占めていることが分かる．

しかし受信処理はホップ数が増加した場合でも，上位層にパケット渡す IWNR スタックの処理は変化しないため，図 14 に示すように 3 ホップの経路にのみ大幅な RTT の増加が発生する理由とならない．

次に 3 ホップの受信処理時間を詳細に解析した．3 ホップ時の受信端末において，IWNR スタックの処理関数に rdtsc 命令を追加し，処理時間の詳細を 10 回測定した．その平均値を表 2 に示す．

表 2 では，それぞれの処理時間および全体の割合を示している．デバイスからの受信処理は，デバイスドライバから渡されたデータをキュー渡す処理である．スレッド起床では Core コードのプロセスである，図 8 に示す IWNR 独自のカーネルスレッドを起床させるまでの時間となる．次の，キューからのデータ取得はプロセスの起床後，キューからデータを取得し，パケット毎の処理関数を呼ぶまでの時間を，ヘッダ処理では処理関数によりヘッダを処理後，上位層へパケットを渡すまでの時間となっている．

表 2 に示す測定結果から，受信処理の 0.99 以上はパケットをキューに格納後，スレッドに起床要求を発行し実際に処理するためのカーネルスレッドが起動するまでの時間である．つまり，休眠しているプロセスに対し起床要求の発行を行っても，すぐにプロセスが起床し，実行のプロセスとしてスケジュールが行われないため大幅に時間がかかっている．3 ホップの経路においてのみ，起床までの時間がかかってしまったのは Linux のプロセススケジュールとのタイミングで，起床までの時間に時間がかかってしまう状態で受信パケットが到着してしまったのが原因であると考えられる．現状ではカーネルスレッドの優先度は他のプロセスと同一なものとなっているため，解決策としてカーネルスレッドのスケジュール優先度を上げ，他のプロセスより優先的にスケジュールされるよう，優先度を変更するなどの方法が考えられる．

7. まとめと今後の課題

本論文のまとめと今後の課題について述べる．

7.1 ま と め

本論文は異種無線メディアで構成されるネットワークにおいて，ユーザが透過的に無線ネットワークへ接続できるフレームワークである IWNR を提案した．IWNR は各無線メディアが異なる場合，異なる無線ネットワーク同士による透過的な接続を行うことを目的し，従来の無線ネットワークでは解決できなかった，無線メディアの壁という問題を解決した．

また実際に実機に IWNR を実装し，Bluetooth と 802.11b を無線メディアとして用い，転送性能，遅延の性能を評価した．転送性能はホップ数が増加する毎に劣化し，遅延も同様にホップ数が増加する毎に増加した．また本機構を組み入れず静的に経路表を変更し，マルチホップの経路を構築した環境との転送性能の比較において，IWNR では性能の劣化は見られたが，No IWNR と比較すると平均 0.8 倍となり大きな性能劣化は見られなかった．

本機構を用いることで．異なる無線メディア同士による透過的なネットワーク接続が可能となった．

7.2 今後の課題

本小節では今後の課題について述べる．

- 経路維持の実装
  今回の実装では経路維持機構の実装はされていないため，端末の移動による経路の途切れが判明できない．経路維持を実装し，再び評価を行い，IWNR の性能を調査する．
- 評価環境における規模の拡大
  今回の評価では 3 ホップの経路を構築し，計 4 台のノードでしか測定を行わなかった．今後評価ノードの台数を増やし評価環境の規模を拡大し，IWNR の評価を詳細に行う．
- 多種の無線メディアへの対応
  本論文は IWNR が使用する無線メディアとして Bluetooth と 802.11b のみの実装を行った．今後，IrDA やセンサデバイスなどの他の無線メディアへの対応を行う．
- IWNR のアドレス割り当て
  IWNR を用いる際，上位層，下位層から IWNR を独立させるため，IWNR 独自のアドレスを用いる．今回は手動でアドレスを割り当てているが，このアドレス割り当ての自動化を行う．
- セキュリティ
  IWNR スタックが存在する端末は IWNR ネットワークに誰でも参加ができてしまう．今後，自分のネットワークと他のネットワークを論理的に区別し，他のネットワーク端末は通信が不可能なセキュリティ機構を導入する．

参 考 文 献

1) Lou, H., Ramjee, R., Sinhan, P., Li, L. and Lu, S.: UCAN: A Unified Cellular and Ad-Hoc Network Architecture, *Proceedings of ACM MobiCom '03* (2003).
2) Kawadia, V., Zhang, Y. and Gupta, B.: System Services for Ad-Hoc Routing: Architecture, Implementation and Experiences, *Proceedings of ACM MobiSys '03*, pp. 99–112 (2002).
3) Perkins, C.: Ad hoc On-Demand Distance Vector (AODV) Routing (2003). RFC 3561.
4) Chakeres, I. D. and Belding-Royer, E. M.: AODV Routing Protocol Implementation Design, *Proceedings of IEEE WWAN '04*, pp. 698–703 (2004).
5) Maltz, D. A.: On-Demand Routing in Multi-hop Wireless Ad Hoc Networks (2001). Ph.D. Thesis, School of Computer Science, Carnegie Mellon University.
6) Broch, J., Johnson, D. and Maltz, D.: The Dynamic Source Routing Protocol for Mobile Ad Hoc Netoworks (2003). IETF Internet-Draft [Work in Progress].
7) l2test: http://www.bluez.org/download.html.
8) Netperf: http://www.netperf.org.
9) l2ping: http://www.bluez.org/download.html.