異種混在無線ネットワークにおける経路制御による仮想的な疑似デバイス構築手法

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

近年，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デバイスで構成され，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，終端端末間で無線デバイス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）を提案する．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デバイス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デバイスを切り替え，他の無線ネットワークへの接続を可能とする．また，無線デバイスが独自に持つユーザへの API を IWNR が提供することで，特定の無線デバイスに依存するアプリケーションの動作を可能とする．本論文では多様な無線デバイス，プロトコルに汎用的なソフトウェアになるよう本機構の実装を Linux カーネル上へ行い，無線デバイスとして IEEE 802.11b と Bluetooth を用い実環境において IWNR の性能を評価した．評価結果として IWNR は実環境において既存のアプリケーションが十分動作できる帯域供給が可能である透過的な無線ネットワークを構築できる．

A Routing-centric Approach for Construction of Virtual Transparent Device in Heterogeneous Wireless Networks

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

This paper presents a new network architecture for heterogeneous wireless networks. Although new systems of wireless networks have appeared, the ad hoc interconnection of different types of wireless devices have not been investigated. We propose Interchangeable Wireless Network Routing (IWNR) as a mean of connecting different wireless network. IWNR allows a user to access any wireless network. IWNR is based on mobile ad hoc networks (MANETs) and offers a virtual API through which an application on a device can access to a remote device with a different type of wireless networks. We have implemented IWNR for IEEE 802.11b and Bluetooth as a LKM for the Linux kernel. The evaluated results have shown that IWNR provides sufficient throughput for interconnecting IEEE 802.11b and Bluetooth devices.

1. はじめに

近年，無線技術の発達にともない，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 IEEE 802.11 a/b，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，無線端末の普及とともに機器の小型化が進み，Mote[1)] 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．

数多くの無線デバイスが存在する無線ネットワークでは各無線デバイスごと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デバイスを特定し，同一の無線デバイスによって通信をする必要がある．

† 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
†† 東京電機大学工学部情報メディア学科
Department of Information Systems and Multimedia, School of Engineering, Tokyo Denki University
††† 慶應義塾大学環境情報学部
Faculty of Environmental Information, Keio University

つまりユーザは通信に用いる無線デバイスを意識しなければならず，数多く無線デバイス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．無線デバイスの異なる端末どうしが集まった場合，ユーザへ終端端末間において無線デバイス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，無線デバイスを仮想的に同一に見せる具体的な構築手法として経路制御を使用した IWNR（Interchangeable Wireless Network Routing Protocol）[2),3)] を提案する．IWNR は MANET 技術を用い，経路制御によって送受信端末間の透過的な仮想デバイスを構築する．IWNR の長所として以下があげられる．

- 無線デバイス差異の吸収
  IWNR が無線デバイスの差異を吸収するため，ユーザは自分や送信相手が所持する無線デバイスを意識せず，データ通信が可能となる．
- 既存アプリケーションに対する柔軟性
  IWNR はカーネルモジュールとして実装されているため，既存アプリケーションを変更せずにそのまま使用できる．また無線デバイスが独自に提供する API を IWNR が提供し，無線デバイスに依存するアプリケーションを実際にその無線デバイスを所持していない端末上で動作できる．
- 端末移動に対する通信の持続性
  IWNR は MANET の経路制御機構を応用しているため，端末の移動に対して通信経路が途切れた場合でも，送信元は新たな経路を探索して通信の持続を図る．また，その場に存在する端末のみでネットワークが構築できるため，既存インフラストラクチャの存在が必要ない．

本論文の構成は，まず 1 章で研究背景である無線端末の普及について述べ，2 章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を提案する．次に，3 章で提案した手法を用いる関連研究を述べ，4 章で終端端末間の無線デバイスを仮想的に同一に見せる IWNR の設計，5 章で IWNR の実装について説明する．最後に，6 章で実機によって IWNR を動作させ評価の結果を報告し，7 章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

既存の無線ネットワークが持つ問題の大きな焦点は，多種多様な無線デバイスが混在する無線ネットワーク（異種混在無線ネットワーク）において，個々の無線ネットワークどうしの相互接続が不可能な点である．そのため異種混在無線ネットワークでは，Bluetooth，IEEE 802.11b の無線ネットワークなど，無線デバイスによってネットワークごとに壁が存在し，ユーザは端末が所持する無線デバイスを意識しなければならない．ユーザが端末の無線デバイスを意識せず，大きな枠組みの無線ネットワーク（透過的な無線ネットワーク）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る．

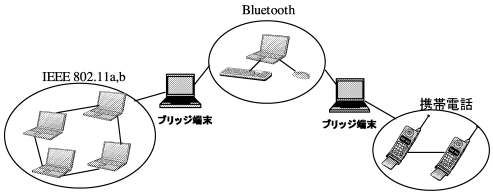

図 1 ブリッジ端末
Fig. 1 Bridge node.

異種混在無線ネットワークを抽象化した透過的な無線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デバイスを同一デバイスとして使用する手法があげられる．この手法を用いて構築された透過的な無線ネットワークは，Bluetooth，IEEE 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．

仮想的に無線デバイス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あげられる．たとえば，無線によるオーバ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，全無線デバイス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，既存アプリケーションをそれらの手法で構築した透過的な無線ネットワークで動作させようとした場合，アプリケーションの変更が必要となってしまう．そこで，本論文は仮想的に無線デバイス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による手法を用いた IWNR 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 1** 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，それぞれの無線デバイス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．透過的な無線ネットワーク構築のためには，端末が実デバイスを所持しない場合，ユーザは IWNR が提供する仮想的なデバイス（仮想デバイス）で通信を行い，下位層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，1，MANET の技術によるブリッジ端末までの動的な経路の構築，2，ブリッジ端末における適切なインタフェース切替えによる通信，3，実無線デバイスを所持しないユーザへの仮想デバイスの提供，を行い透過的な接続を可能にする．

3. 関 連 研 究

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，Lou らが提案する UCAN（A Unified Cellular and Ad-hoc Network Architecture）[4] があげられる．UCAN は，IEEE 802.11b の MANET と携帯電話のセルラ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にする．ただし，IEEE 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 IEEE 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混在無線ネットワーク間の透過的な接続は実現できない．

上位層に多種なプロトコルを用いても，汎用的にアドホックネットワークが構築できる経路制御プロトコルの実装アプローチとして，Draves らが提案した実装手法[5] があげられる．Draves らは経路選択の手法に改良を加えた DSR（Dynamic Source Routing）[6] を提案している．多くの MANET プロトコルは，宛て先までの経路が複数存在した場合ホップ数を指標として最短経路を選択する．Draves らは経路選択の指標として RTT などのホップ数以外の指標を加え，MANET を構築しそれぞれの経路選択指標の違いによる性能比較を行っている．実際に実機にて評価を行う際に，Draves らは経路制御プロトコルを実装している．このプロトコルは OSI における第 3 層で使用するネットワークプロトコルに依存しないよう，第 2 層に実装されているため，第 3 層へ多くのプロトコルが使用できる．しかし本論文で想定しているような複数の無線デバイスを想定した実装手法ではない．

4. IWNR の設計

IWNR は MANET の制御プロトコルを拡張し，ユーザが無線デバイス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デバイスで構築される異種混在無線ネットワークを，透過的な無線ネットワークとして提供できる．ユーザは宛て先が実際に搭載している無線デバイスを考慮することなく，ユーザの希望する無線デバイス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あらかじめ知っている必要がある．

既存のネットワーク機構において，ネットワークデバイスの差異を吸収するために第 3 層が存在する．しかし近年の無線デバイスは用途や用法により，様々な規格が存在しており IP アドレスをサポートしない，ブロードキャストやプロミスキャスモードが不可能である，また，リンクレベルでの接続を行わなければ上位層によるデータ送信が不可能な無線デバイスなどが存在し，これらの無線デバイスを使用する場合，第 3 層だけでは様々な無線デバイスの差異を吸収できない．そこで IWNR では，第 2 層によって無線デバイスの差異を吸収する方針をとる．

図 2 に示すように，無線デバイスとして IF1 のみを所持する端末である送信元が，無線デバイスとして IF2 を所持する宛て先までデータ通信を行うとする．IWNR では，送信元が IF1，IF2 の無線デバイスを両方所持するブリッジ端末まで動的に経路を構築し，ブリッジ端末がデータの宛て先により IF1，IF2 を適切に切り替えることで，異種混在無線ネットワークへ透過的な接続を提供できる．

IWNR は MANET の制御プロトコルの 1 つである DSR を拡張し設計行った．IWNR は DSR による MANET 機能を用い，ブリッジ端末および宛て先までの経路の発見，データの転送を行い，端末の移動により経路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る必要がある．次節で透過的な無線ネットワーク下で起こりうる問題の具体的なシナリオとともに，必要となる新たな機能に関して述べる．

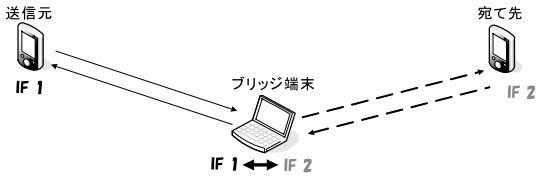

図 2 ブリッジ端末によるインタフェース切替え
Fig. 2 Change interface in bridge node.

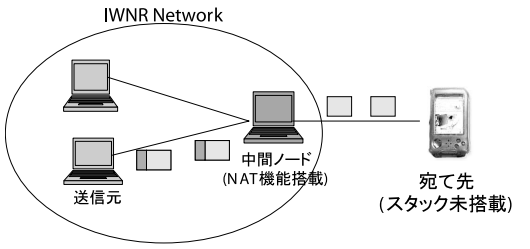

図 3 IWNR スタック未搭載端末による IWNR ネットワークへの接続
Fig. 3 Connect to IWNR network for node without IWNR stack.

4.1 IWNR のシナリオモデル

プロトコ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デバイスは，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，その端末は 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．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 をサポートする各端末は**図 3** に示すような NAT（Network Address Translation）機能を保持するものとする．図 3 では IWNR ネットワークと IWNR が未搭載の端末である宛て先とを接続し，経路の途中に存在する端末（中間ノード）が存在する．中間ノードが IWNR のプロトコルスタック（IWNR スタック）を未搭載な端末と IWNR スタックを搭載した端末とを接続している場合，IWNR のヘッダの追加，削除，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．ただし，プロトコルスタック改変が不可能な端末が IWNR ネットワークへ参加できる条件は，IWNR ネットワークのエッジに端末単体で存在する場合のみに限られる．

デバイスに依存するアプリケーション

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 を利用して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．

たとえば，Bluetooth のプロトコルスタックである Bluez[7] 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 に依存するアプリケーションはドメインとして PF_BLUETOOTH を呼び出す．そのため Bluetooth に依存するアプリケーションは，Bluetooth が搭載されていない端末上では正常に動作しない．そのような状況下で IWNR を利用し Bluetooth デバイスのみを搭載した送信元と，IEEE 802.11b のみを搭載した宛て先への接続を行うとする．送信元のアプリケーションが Bluetooth に依存し 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．そこで無線デバイスが独自に提供している API を IWNR が仮想的な API（仮想 API）として提供し，その仮想 API を通じ送信データを IWNR が仮想デバイスとして送信することで，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，プロトコル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 を仮想 API として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．以上をふまえたうえで 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) 無線デバイス切替え機能
(3) NAT 機能
(4) 仮想 API の提供

4.3 動 作 手 順

IWNR は DSR を拡張した設計となっているため，IWNR の動作手順は DSR の動作手順とほぼ同一である．

経 路 発 見

図 4 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

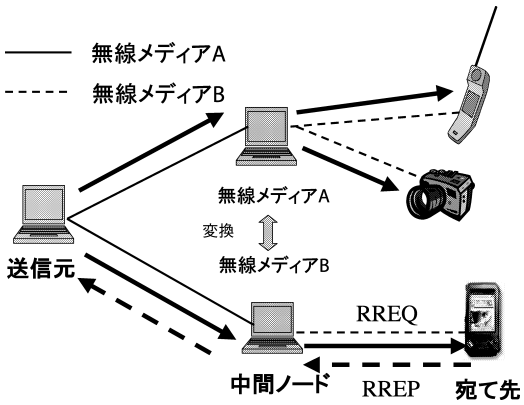

図 4 IWNR の経路発見
Fig. 4 Route discovery for IWNR.

ケット（RREQ）に一意な識別子（RREQ ID），宛て先アドレス，送信元アドレスを格納し，ブロードキャストによってネットワークすべての端末に対して送信する．RREQ を受信した端末は RREQ に格納してある経路情報，インタフェース情報を経路表に登録し，自らのアドレス，インタフェース情報を RREQ へ格納し，再び RREQ をブロードキャストによって他の端末へ転送する．もし RREQ を受信し，かつ無線デバイスを複数所持する端末はすべての無線デバイスに対して，RREQ を転送する．また，RREQ を受信した端末は送信した宛て先アドレスと RREQ ID の組合せを調査する．もし端末が以前に同一な組合せの RREQ を受信した場合，その RREQ を転送せずに破棄する．これによりネットワークにおける RREQ のループを防ぐ．

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，ユニキャストによって RREP を送信元へ送信する．RREP を受信した中間ノードは 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デバイスを使用し，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．

データ転送

RREP を受信した送信元は宛て先までの経路が判明した場合，データ送信を開始する．送信元は通常データのヘッダに IWNR が構築する独自のヘッダ（IWNR ヘッダ）を付随させる．IWNR ヘッダにはデータの送信元，宛て先アドレス，転送元の端末アドレス，転送する次の端末アドレス，上位データ情報，データシーケンス，経路の識別子が格納され，IWNR ヘッダを付随したデータが送信される．転送データを受信した中間ノードは宛て先アドレスをキーとして経路表に問合せを行い，次の転送先とインタフェース情報を取得する．中間ノードはそれをもとに IWNR ヘッダの情報更新を行い，適切な無線デバイスによりデータを転送する．IWNR では DSR の Flow State を用いてデータ転送を行うため，IWNR ヘッダ内にすべての転送端末アドレスを格納する必要はない．

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，経路上の中間ノード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，端末は転送データの IWNR ヘッダからデータシーケンスを取得し，このデータシーケンスを含む ACK パケットを生成して，転送元へ送信する．各端末は各データの送信時刻を記録しており，それに対応するデータの ACK を受信すると，次の転送先端末までの RTT（Round Trip Time）を計算し，記録する．もし転送元が転送先から測定された RTT に基づいて決定された再送タイムアウト時間内に転送先からの ACK を受信できない場合，転送先にデータの再送を行い，ある一定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ことを検知する．タイムアウトの計算方法および再送回数は DSR と同一であり，タイムアウトの計算は RFC793 による TCP タイムアウトの計算法を用いる．また IWNR の再送回数も DSR と同様に 2 回とする．転送元による再送が失敗し，使用経路が途切れたことを検知した際，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すべて削除する．

アドレス変換

IWNR はプロトコル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，データへの IWNR ヘッダの付加（カプセル化）および削除（カプセル解除）を行う．IWNR が動作する端末（IWNR 端末）が RREQ を転送後，IWNR 端末は各デバイスごとに IWNR を使用せず，もとの無線デバイスのプロトコルスタックへ 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

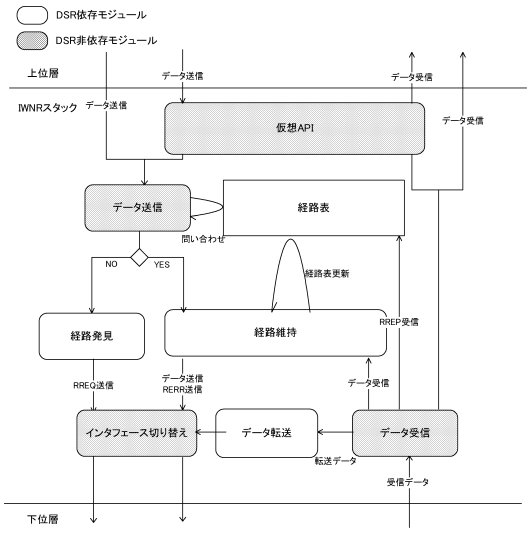

図 5 IWNR のモジュール構成図
Fig. 5 IWNR modules.

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（未搭載端末）として扱われる．IWNR ネットワークと未搭載端末との中間に存在する IWNR 端末はアドレスの変換，未搭載端末から IWNR 端末への送信データのカプセル化，IWNR 端末から未搭載端末への送信データのカプセル解除を行う．また未搭載端末より希望する端末へデータ送信が行われる場合，未搭載端末に接続をしている IWNR 端末が代わりに宛て先に対する経路要求を行い，宛て先までの経路を確立する．

4.4 IWNR モジュール

IWNR モジュール全体の構成図を**図 5** に示す．DSR に依存するモジュールは経路発見，経路維持，データ転送となっており，その他のモジュールは DSR に非依存なモジュールとなる．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，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 RREP や 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，データ受信モジュールはデータ転送モジュールへ転送データを渡し，経路維持を行うため転送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，経路維持モジュールは RERR の送信を行い，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，データ受信モジュールは IWNR 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．そこで無線デバイス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（IWNR アドレス）を用い，端末の認識を行う．そのため，IWNR は経路要求を行う RREQ の宛て先アドレスはデバイス依存のアドレスを用い，転送を行うためのソースルーティングは IWNR アドレスを用いる．

5. IWNR の実装

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）をサポートしている．本機構ではユーザが簡単に本機構を使用できるよう LKM を用いて実装を行う．

IWNR がサポートする無線デバイスとして，IEEE 802.11b，Bluetooth を選択する．Bluetooth は通常の TCP/IP プロトコルを用いず，独自のプロトコル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．将来的には今回実装に用いた IEEE 802.11b，Bluetooth のみだけではなく，多くの無線デバイスも IWNR へ対応する予定である．また本実装では前章で示した 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デバイスの対応を予定している．IWNR が新たな無線デバイスをサポート

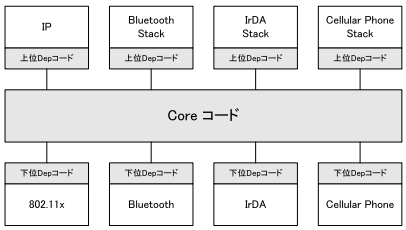

図 6 IWNR における実装のポリシ
Fig. 6 Policy of IWNR implementation.

するために，IWNR の大幅なコードの追加や変更は望ましくない．無線デバイスに依存する部分であるごく一部のコードを IWNR へ追加すれば，IWNR へ新たな無線デバイスが対応できるよう実装すべきである．そこで IWNR の実装方針として，**図 6** に示すように無線デバイスに依存する部分のコード（Dep コード）と，無線デバイスに依存せずに使用でき，IWNR の機能において中核となる部分のコード（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無線デバイスに特化した Dep コードを IWNR へ追加することで，簡単に新たな無線デバイスを IWNR へサポートできる．また Dep コードは上位層と接続する部分（上位 Dep コード）と下位層で無線デバイスに接続する部分（下位 Dep コード）に区分される．Dep，Core コードの必要な機能と対応する，図 5 のモジュールを以下にあげる．

- Dep コード
  - 無線デバイスに依存する情報の管理（経路表）
    下位，上位 Dep コード：IWNR アドレスと実アドレスのマッピングやデータリンク層でのコネクション管理など無線デバイスに依存する情報の管理を行う．
  - データ受信機能（データ受信モジュール）
    下位 Dep コード：実無線デバイスからデータを受信した場合，無線デバイスに依存する情報の記録を行い，データリンク層のヘッダを削除し Core コードへデータを渡す．
    上位 Dep コード：Core コードから渡された自分宛てのデータを上位層へ渡す．
  - データ送信機能（データ送信，インタフェース切替えモジュール）
    下位 Dep コード：Core コードから渡された送信データへデータリンク層のヘッダを付随させ実デバイスへデータを送信する．
    上位 Dep コード：上位層からデータ送信の要求が発生した場合，宛て先までの経路の有無を調べ，Core コードへ送信データを渡すもしくは経路発見要求を行う．
  - 仮想 API の提供（仮想 API モジュール）
    上位 Dep コード：無線デバイス独自の API を仮想 API として提供する．
- Core コード
  - 経路管理機能（経路表）
    RREQ，RREP，RERR を受信した際，経路表へ経路情報の更新を行う．
  - データ送信機能（データ送信，インタフェース切替え機構）
    上位 Dep コードより渡された送信データへ経路情報が格納された IWNR ヘッダを付随させ，適切な下位 Dep コードのデータ送信機能へ送信の要求を行う．
  - データ受信機能（データ受信，データ転送モジュール）
    下位 Dep コードより渡された受信データを解析し，受信データの転送，破棄および，適切な上位 Dep コードのデータ受信機能へ受信の要求を行う．
  - 経路発見機能（経路発見モジュール）
    上位 Dep コードより経路発見の要請があった場合，RREQ を作成し，全無線デバイスにおける下位 Dep コードのデータ送信機能へ送信要求を行う

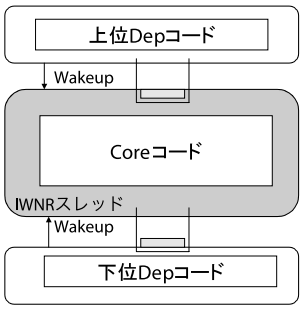

図 7 IWNR のカーネルスレッド
Fig. 7 A kernel thread of IWNR implementation.

また**図 7** 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 Dep コードから渡されるデータはすべてキューを使用し Core コードのスレッドに渡される．

5.3 Dep における実装

IEEE 802.11b における実装

IEEE 802.11b の実装では，IWNR スタックが**図 8** に示すようにデータリンク層とネットワーク層に位置

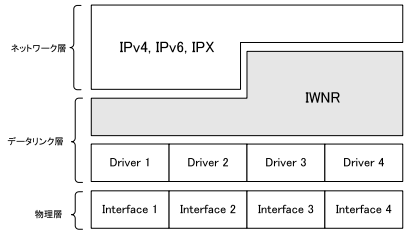

図 8 無線 LAN における IWNR の実装
Fig. 8 IWNR implementation for wireless LAN.

する．IWNR スタックは仮想デバイスとして存在するためデータリンク層となる．データリンク層に実装することで，IWNR スタックの上位層となるネットワーク層で使用されるプロトコルによらず IWNR が実装できるためデータリンク層に実装を行った．

IEEE 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．将来的には IWNR アドレスを MAC アドレスより自動生成することを想定している．

以下に IEEE 802.11b の Dep コードの機能に関して説明する．

- 無線デバイスに依存する情報の管理
  下位 Dep コード：隣接端末の IWNR アドレスとそれに対応する MAC アドレスをエントリとして持つテーブル（MAC アドレステーブル）を所持する．MAC アドレステーブルは端末が IEEE 802.11b よりデータ送信を行う際に IWNR アドレスから MAC アドレスを解決するために使用され，データを受信した際に IWNR，MAC ヘッダが解析され，MAC アドレステーブルのエントリが更新される．
- データ受信機能
  下位 Dep コード：受信パケットのイーサタイプが IWNR を示すタイプならば，実デバイスドライバからデータ受信機能へパケットを渡す．その後，受信された IWNR パケットは Core コードに存在するデータ受信機能へキューを通じて渡される．
  上位 Dep コード：Core コードより渡された自分宛ての受信パケットから IWNR ヘッダを取り除き，適切な上位層へ受信パケットを渡す．
- データ送信機能
  下位 Dep コード：Core コードから渡された送信データを実無線デバイスへ送信する．その際 MAC アドレステーブルより IWNR アドレスと対応する MAC アドレスを取得し，MAC ヘッダの作成後，実デバイスに対し送信要求を行う．
  上位 Dep コード：上位層からデータ送信要求が発生した場合，Core コード内にある経路表より宛て先までの経路があるかを問い合わせる．もし，宛て先の経路が存在した場合 Core コード内のデータ送信機能にデータを渡す．Core コード内の経路表に経路が存在しなければ，Core コード内の経路要求機能を呼び出し，経路要求を行う．
- 仮想 API の提供
  上位 Dep コード：本実装は IWNR をイーサネットの仮想デバイスドライバとして実装しているため，AF_INET ドメインソケットに対する仮想 API の提供は必要ない．

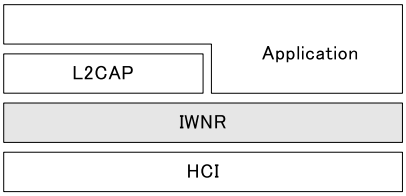

図 9 Bluetooth における IWNR の実装
Fig. 9 IWNR implementation for Bluetooth.

Bluetooth における実装

Bluetooth の実装では，**図 9** で示すように HCI（Host Control Interface）と L2CAP の中間に IWNR スタックが存在する．L2CAP から送信されるデータは HCI に送信される以前に IWNR スタックを通過し，HCI で受信した ACL データはすべて IWNR スタックに受信される．以下に Bluetooth の Dep コードに関する機能を述べる

- 無線デバイスに依存する情報の管理
  下位 Dep コード：隣接ノードの IWNR アドレスとそれに対応する HCI コネクションをエントリとして持つテーブル（コネクションテーブル）を所持する．端末が Bluetooth 端末へデータを転送する際，IWNR アドレスからコネクションテーブルより対応する HCI コネクションを取得してデータ送信を行う．コネクションテーブルは HCI 層においてコネクションの状態が変化した際に更新される．
  上位 Dep コード：相手先の Bluetooth アドレスとそのアドレスに対応する IWNR アドレスをエントリとして持つテーブル（Bluetooth アドレステーブル）を所持する．Bluetooth アドレステー

ブルはデータ送信時に上位 Dep コードによって Bluetooth アドレスから IWNR アドレスに変換する際に使用される．また Bluetooth アドレステーブルのエントリは RREQ，RREP，RERR を受信した場合に更新される．

- データ受信機能
  下位 Dep コード：HCI より受信されたデータから HCI ヘッダを取り除き，Core コードのデータ受信機能へキューを通じデータを渡す．
  上位 Dep コード：Core コードから渡された受信データが L2CAP 層のデータである場合，そのデータを受信した HCI コネクションが必要になるため，対応するコネクションをコネクションテーブルより取得し，受信データに含まれる IWNR ヘッダを取り除き L2CAP 層にデータを渡す．
- データ送信機能
  下位 Dep コード：Bluetooth にはブロードキャストという機能がないため，Core コードから渡された送信データが RREQ ならば，HCI において inquiry の送信を行い，応答したすべての Bluetooth 端末へコネクションを確立し，各コネクションに対して RREQ を送信する．RREQ 以外であればそのデータはユニキャストで送信されるため，コネクションテーブルより対応するコネクションを宛て先の IWNR アドレスから取得し，実デバイスに対しデータの送信要求を行う．
  上位 Dep コード：Bluetooth はデータの送受信を行う前に必ず，宛て先もしくは転送先に対し HCI でのコネクションを確立する必要がある．L2CAP において HCI コネクション確立の要求が行われた際，下位 Dep コード内のコネクションテーブルへコネクションの問合せを行い，もし宛て先もしくは転送先までのコネクションが確立し，かつ，次の転送先が無線メディアとして Bluetooth を使用するならば，IWNR アドレスより対応するコネクションを返す．次の転送先が Bluetooth でない場合，下位 Dep コード内で作成された疑似コネクションを L2CAP へ返す．もし宛て先および転送先に対応するコネクションがコネクションテーブルに存在しない場合，宛て先までの経路が決定されていないため，Core コード内にある経路要求機能を用い経路要求を行う．
- 仮想 API の提供
  上位 Dep コード：Bluez における独自の API は，AF_BLUETOOTH ドメインのソケットから呼び出される関数であるため，それらの関数が仮想 API として IWNR で提供される．仮想 API は送信データの前へ L2CAP ヘッダを追加し，上位 Dep コード内のデータ送信機能にデータを渡す．

5.4 Core コードにおける実装

Core コードの実装は，下位の無線デバイスおよび上位のプロトコルに依存せず処理を行える部分を示している．また，Core コード内ではそれぞれの端末情報はすべて 32 ビットの識別子である IWNR アドレスによって扱われ，処理するデータも IWNR ヘッダのみとなり，上位，下位層のデータ処理は行わない．

Core コードは以下に示す機能を有する．

- 経路管理機能
  経路管理機能は Core コード内の経路管理を行う機能であり，デバイスもしくは上位層に依存するアドレスや情報は Core コード内の経路表には存在しない．経路表はエントリとして経路の宛て先，無線インタフェース，全中間ノードのアドレス，経路の識別子を持ち，転送先のアドレスやデータ送信時の使用無線デバイスの情報取得のために使用される．また，経路表のエントリは RREP，RREQ 受信時に更新される．
- データ受信機能
  IWNR ヘッダより受信データの種類を判別し，それに対応した処理を行う．自分自身が宛て先でないパケットならば，IWNR ヘッダの更新を行いデータを転送するため，下位 Dep コードの送信機能にデータを渡す．もし自分宛ての受信パケットならば，適切な上位 Dep コードのデータ受信機能にデータを渡す．
- データ送信機能
  データ送信機能は，上位 Dep コードに存在するデータ送信機能より，経路表に宛て先までの経路が存在した場合に呼び出される．まず渡された送信データの前へ，必要な情報を格納した IWNR ヘッダを付随させ，下位 Dep コードに存在するデータ送信機能にデータを渡す．
- 経路発見機能
  上位 Dep コードのデータ送信機能より，宛て先に経路がない場合に呼び出され，RREQ の作成を行う．作成された RREQ は無線デバイス分だけ複製され，全無線デバイスのデータ送信機能へ RREQ が渡される．

6. IWNR の評価

IWNR は異種混在無線ネットワークにおいて透過的な無線ネットワークを構築するための機構である．そ

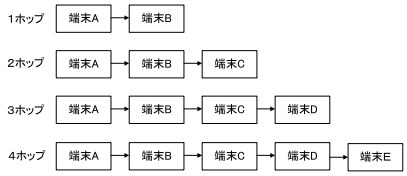

図 10 実験環境（直線型トポロジ）
Fig. 10 Evaluation environment (line topology).

のため，本機構は実際に機器どうしの接続が可能となることを目標としており，既存のシミュレータでは異種混在無線ネットワークは想定していないため，実機による評価を行った．また本論文では経路維持モジュールを実装しておらず，送信端末が経路の途切れを検知することが不可能であり，同じ経路を用いてデータ送信を行う．しかし今回の評価環境ではメッシュネットワークのように端末が静止し，経路の途切れる状況が発生しないため未実装でも問題ない．

本機構はシミュレータでなく実環境での使用を前提としている．実際に IWNR を使用した場合，既存のネットワークアプリケーションが通常の動作環境と変わらず使用できることが重要である．そこで，本評価では使用アプリケーションに十分な帯域や遅延の供給が可能であるかを評価するため実機によってマルチホップの環境を構築し，転送性能と遅延を評価した．次に IWNR の遅延や転送性能に影響を与える，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

IWNR を Red Hat Linux-2.4.20 に LKM として実装し，Bluetooth および IEEE 802.11b を無線デバイスとして用い評価を行った．

測定を行う実験環境は**図 10** に示すように，端末 A を送信元として固定し，端末 A，B，C，D，E からなる 1 から 4 ホップの経路を構築した．また**図 11** に示すように，Bluetooth と IEEE 802.11b が混在するスター型のトポロジを構築した．評価を取得するために用いたラップトップの性能および無線デバイスの詳細を**表 1**，**表 2** に示す．

6.2 スループット

IEEE 802.11b のみの無線デバイスを用いて TCP スループットの測定を行い，次に IEEE 802.11b と Bluetooth および Bluetooth のみでネットワークを構築し，Bluez のツールとして公開されている l2test プログラム[11] を用い，L2CAP 上で転送性能の評価を行った．

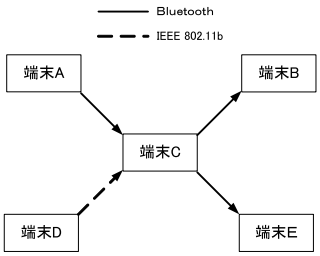

図 11 実験環境（スター型トポロジ）
Fig. 11 Evaluation environment (star topology).

表 1 評価端末
Table 1 Node's specs for evaluations.

ノード名	PC
端末 A	Panasonic Let's Note CF-M2XR メモリ：64 MB，CPU: Intel Pentium 3 650 MHz
端末 B	Panasonic Let's Note CF-B5R メモリ：64 MB，CPU: Intel Pentium 3 600 MHz
端末 C	IBM ThinkPad T41p メモリ：1 GB，CPU: Intel Pentium M 1.70 GHz
端末 D	IBM ThinkPad T41p メモリ：1 GB，CPU: Intel Pentium M 1.70 GHz
端末 E	IBM ThinkPad T41p メモリ：1 GB，CPU: Intel Pentium M 1.70 GHz

表 2 無線デバイス
Table 2 Wireless devices.

デバイス名	製品名
IEEE 802.11b	Buffalo Melco WLI-PCM-L11 [8]
Bluetooth A	3 COM 3 CREB96 [9]
Bluetooth B	Brainboxes BL-554 [10]
Bluetooth C	ThinkPad T41p 内蔵 Bluetooth
Bluetooth D	ThinkPad T41p 内蔵 Bluetooth
Bluetooth E	ThinkPad T41p 内蔵 Bluetooth

IEEE 802.11b におけるスループット

IEEE 802.11b のみを無線デバイスとして使用し，端末 A を送信元とし，経路が決定した後，TCP スループットを測定した．まず，MAC アドレスでフィルタリングを行い疑似的に 1 から 3 ホップのネットワークを構築し，予備的な評価を行った．NetPerf [12] 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 は通常の IP スタックに存在する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ごとにスループットが減少しており，平均 0.86 倍の性能劣化が見られた．IWNR は NO-IWNR に比べ，各パ

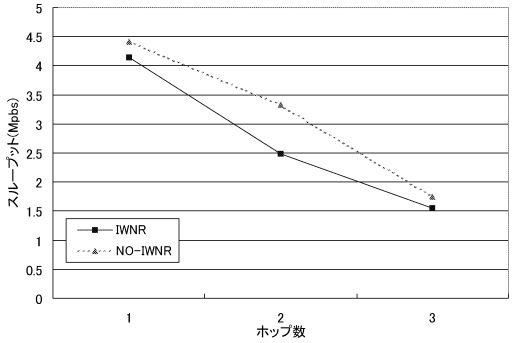

図 12 IEEE 802.11b におけるスループット（フィルタリングあり）
Fig. 12 Throughput in IEEE 802.11b (with filtering).

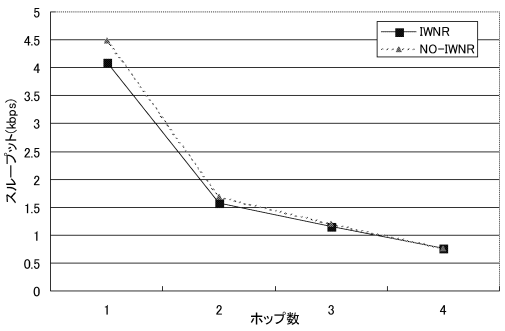

図 13 IEEE 802.11b におけるスループット
Fig. 13 Throughput in IEEE 802.11b.

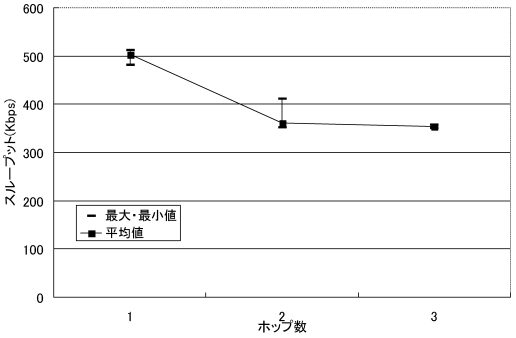

図 14 Bluetooth，IEEE 802.11b におけるスループット
Fig. 14 Throughput in Bluetooth, IEEE 802.11b.

ケットに対して IWNR ヘッダが 32 byte 分増分し，パケットサイズのみの算出で約 2%ほどスループットが減少する．また NO-IWNR が行う通常の処理に加え，IWNR は IWNR 独自の処理を各端末上で行う必要があるため遅延が高くなり，NO-IWNR に比べ最低でも 2%以上 TCP スループットが減少する．Padhye らの TCP スループット推定式[13] より実測値の具体的な値を考察する．Padhye らの推定式はパケットロス率，パケットサイズ，RTT より TCP スループットの推定値を算出できる．ただしロス率が 5%以下の状況で TCP スループットの実測値と推定値が非常に近似することが知られている．一番ロス率が低いと考えられる，1 ホップの NO-IWNR におけるスループット，遅延から推定式を用いてロス率を求めると約 0.004 となる．このロス率からパケットサイズが 1,468 byte（$1,500 - 32 = 1,468$），1 ホップ時の IWNR の RTT（0.54 msec）より算出される TCP スループット推定値は 4,065 kbps となり，実測値の値である 4,146 kbps とさほど差がない．そのため実測値から求めた約 0.86 倍という性能劣化値は本評価実験環境において信頼性がある．

次に IEEE 802.11b のみを無線デバイスとして使用し同様な評価実験を行った．ただし MAC アドレスによるフィルタリングを行わず，物理的にノード間の距離を離し，1 から 4 ホップまでの TCP スループットを測定した．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 13** に示す．フィルタリングを用い疑似的にトポロジを構築した評価と同様にホップ数の増加にともない，スループットが低下する．しかしノード間の距離を離すことで各リンクの無線品質が劣化しパケットロスが増加するため，スループットの劣化が激しい．

IEEE 802.11b と Bluetooth におけるスループット

次に IEEE 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 IEEE 802.11b 上でスループットを測定した．IEEE 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．使用無線デバイスは，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 IEEE 802.11b となる．ただし IEEE 802.11b のリンクでは MAC アドレスによるフィルタリングを行った．また，データ送信を行う端末は Bluetooth のみを所持する端末 A とした．

図 14 は l2test を用い，パケットサイズが 672 byte のデータを 10 秒間送受信し 10 回測定した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 ホップ以外はすべて IEEE 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，スループットが

143 kbps と大きく減少している．ブリッジ端末における Bluetooth から IEEE 802.11b への無線デバイスの切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．転送先が同一無線デバイスの場合，パケットを受信した際に追加される無線デバイスの MAC ヘッダサイズは変わらず，受信したパケットへそのまま転送先の MAC ヘッダを追加できる．しかし転送先の無線デバイスが異なる場合，受信したパケットへ追加する無線デバイスの MAC ヘッダサイズが異なる．特に Bluetooth での MAC ヘッダサイズは 4 バイトであるのに対し，IEEE 802.11b は 14 バイトになる．Linux はバッファを受信パケットサイズで取得するため，受信パケットへ新たなデータを追加できない．そのため，Bluetooth デバイスから IEEE 802.11b へ転送を行う場合，転送パケットごとに新たな大きさのバッファを確保し，転送データのコピーをする必要がある．このバッファの確保とデータコピーのオーバヘッドがデバイス切替えのオーバヘッドである．

次に図 11 のようなスター型トポロジを構築し，同様に l2test を用いスループットを測定した．図 11 で示すように，端末 D，端末 C 間のみが IEEE 802.11b のリンクとなり，他のリンクはすべて Bluetooth となる．またデータ送信を行う送信元は 2 台存在し，端末 A から C を経由し端末 E へデータ送信を行うデータフローと，端末 D から C を経由し端末 B に送信を行うデータフローが存在する．結果を**図 15** に示す．縦軸は前回と同様，l2test を用い 10 秒間のデータ送受信を 10 回測定した最大値，最小値，平均値を示している．横軸の IEEE 802.11b・Bluetooth は，端末 D，C，B におけるスループット，Bluetooth は端末 A，C，E におけるスループットを示し，No-Cross-Traffic は送信しているノードが 1 台のみ，つまりデータフローが自分自身しかいない状況であり，Cross-Traffic は送信ノードが 2 台存在する，つまりデータフローが自分自身のほかにもう 1 つ存在する状況である．IEEE 802.11b・Bluetooth および Bluetooth のスループットは No-Cross-Traffic と比べ，Cross-Traffic では急減に劣化している．端末 C は無線デバイスとして IEEE 802.11b，Bluetooth の両方を持つブリッジ端末であり，ブリッジ端末へデータフローが集中した場合スループットが劣化してしまう．

Bluetooth におけるスループット

次に Bluetooth のみを用い，図 10 に示すように 1 から 4 ホップまでの経路を構築し，l2test にてスループットを測定した．**図 16** に結果を示す．図 16 における縦軸は l2test を用いパケットサイズが 672 byte のデータを 10 秒間送受信し，10 回測定した平均値を，横軸はホップ数を示す．2 ホップから 3 ホップにかけてスループットが非常に劣化しているが，これは 3 ホップ目のノードである端末 C が 2 つのピコネットに所属しており，2 台のマスタノードからデータ送信の制御を受けているためだと考えられる．

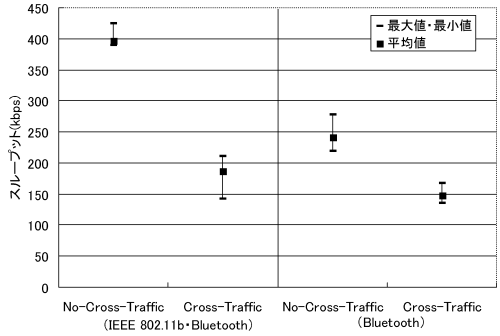

図 15 Bluetooth，IEEE 802.11b のスター型トポロジにおけるスループット
Fig. 15 Throughput in Bluetooth, IEEE 802.11b for star topology.

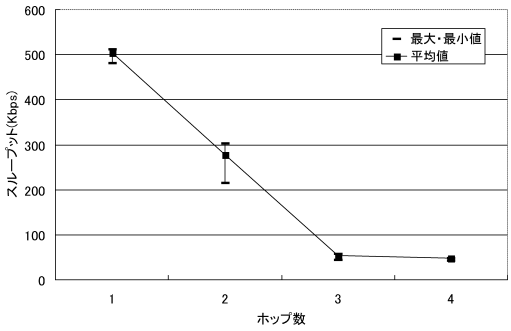

図 16 Bluetooth におけるスループット
Fig. 16 Throughput in Bluetooth.

6.3 遅延に関する評価

まず IEEE 802.11b のみの無線デバイスを用いた RTT の測定を行い，次に Bluetooth と IEEE 802.11b および Bluetooth のみを用い RTT の計測を行った．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．

IEEE 802.11b における遅延

IEEE 802.11b のみの無線デバイスを使用し，前節と同様に端末 A を送信元とし RTT を計測した．計測プログラムには ping を用い，データサイズを 64 byte として 50 回の平均 RTT を測定した．

まず MAC アドレスによるフィルタリングを行い，

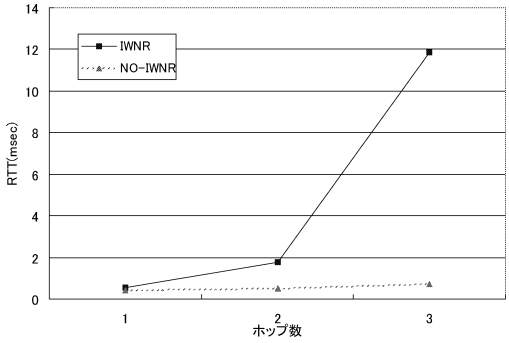

図 17 IEEE 802.11b における RTT（フィルタリングあり）
Fig. 17 RTT in IEEE 802.11b (with filtering).

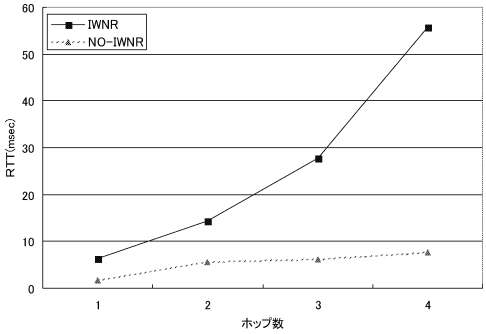

図 18 IEEE 802.11b における RTT（フィルタリングなし）
Fig. 18 RTT in IEEE 802.11b (without filtering).

疑似的なマルチホップのトポロジを構築した評価結果を**図 17** に示す．図 17 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, NO-IWNR ともにホップ数が増加するごと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 倍の差となる．また，IWNR と NO-IWNR における RTT の差が非常に大きいにもかかわらず，図 12 におけるスループットに差がない．これらの原因は 6.5 節のスタックの処理時間において説明を行う．

次に MAC アドレスによるフィルタリングを行わず，実際にノード間の距離を離し 1 から 4 ホップまでのネットワークを構築し RTT の測定を行った．**図 18** に示すグラフは前回と同様に横軸がホップ数を縦軸が RTT を示す．NO-IWNR，IWNR ともにホップ数が増加するごとに RTT が増加する．また，NO-IWNR と IWNR の差は非常に大きくなってしまっている．

IEEE 802.11b と Bluetooth における遅延

次に端末 A，B 間のリンクを Bluetooth，他のリンクを IEEE 802.11b として経路を構築し RTT を測定した．ただし IEEE 802.11b のリンクでは MAC アドレスによるフィルタリングを行った．測定プログラムとして l2ping プログラム[14] を用い，RTT を測定した．l2ping は通常の ping プログラムと同様に L2CAP 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．送信端末を端末 A としデータサイズを 20 byte に設定して RTT の計測を 50 回行った．その結果を**図 19** 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 の平均値を表している．

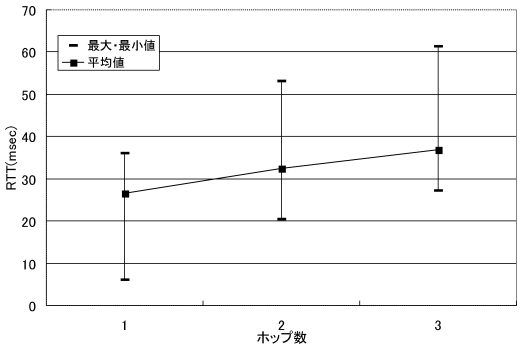

図 19 IEEE 802.11b と Bluetooth におけるホップ数と RTT の関係
Fig. 19 Relationship between RTT and number of hops in IEEE 802.11b and Bluetooth.

IEEE 802.11b のみ経路と同様，ホップ数が増加するごとに 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 ミリ秒と増加しており，スループットと同様にブリッジ端末におけるオーバヘッドより 1 ホップから 2 ホップへの RTT が大きく増加している．また，図 17 において IEEE 802.11b での 1 ホップから 2 ホップへホップ数が増加すると平均約 2 ミリ秒増加し，図 19 において 2 ホップから 3 ホップへ増加した場合，平均約 3 ミリ秒 RTT が増加している．これより，図 19 における 2 ホップから 3 ホップの RTT は IEEE 802.11b が 1 ホップ追加された増分とほぼ一致しており，ブリッジ端末ではない中間端末が 1 つ増えるごとに 3 ミリ秒程度 RTT が増加することが予想される．

Bluetooth における遅延

次に Bluetooth のみを用い，図 10 に示すように 1 から 5 ホップまでの経路を構築し，データサイズ 20 byte とし l2ping にて RTT を測定した．**図 20** に結果を示す．図 20 における縦軸は 50 回の平均 RTT を横軸はホップ数を示す．Bluetooth のみでネットワークを構築した場合のスループットと同様に，ホップ数が増加するごとに急激に RTT が劣化する．またスループッ

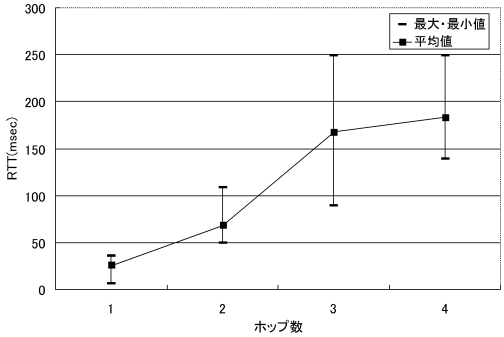

図 20 Bluetooth における RTT
Fig. 20 RTT in Bluetooth.

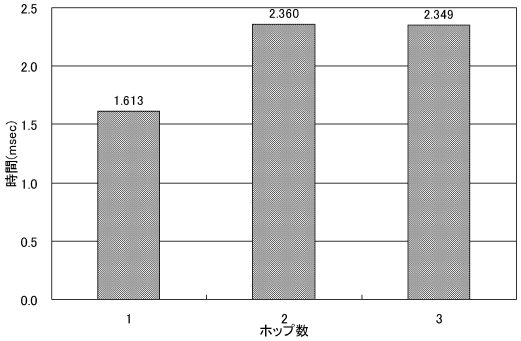

図 21 IEEE 802.11b における経路発見時間
Fig. 21 Latency of route discovery in IEEE 802.11b.

トと同様に 3 ホップ目の端末は 2 つのピコネットに所属するため，急激に RTT が劣化する．

6.4 経路発見時間

IWNR における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した．本評価で用いた CPU は 650 MHz であり測定分解能は約 1.5 ナノ秒となる．まず MAC アドレスのフィルタリングにより IEEE 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 21** に示す．

図 21 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 ミリ秒差となっている．

次に，Bluetooth における経路発見時間の測定を行った．Bluetooth はコネクション型の無線デバイスのため，隣接ノードすべ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**図 22** に示す．横軸はホップ数を縦軸は経路発見時間 (秒) を示す．ホップ数が増加するごとに経路発見時間が増加する．Bluetooth では隣接ノードの探索を行う 1 回の inquiry 要求，応答に平均約 9 秒もかかっており，さらに HCI 層でのコネクションを確立するまで約 1 秒かかる．そのため，4 ホップ先のノードまで RREQ を送信し応答が返るまで 43 秒も待つ必要があり，IWNR ネットワークにおいて Bluetooth 端末がボトルネックとなってしまう．

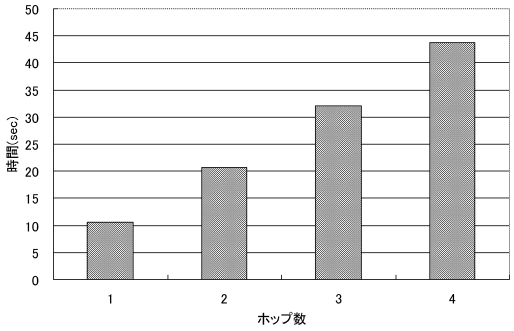

図 22 Bluetooth における経路発見時間
Fig. 22 Latency of route discovery in Bluetooth.

表 3 3 ホップ時の IWNR スタック処理時間
Table 3 Processing time in 3 hop route.

測定端末（処理）	時間 (ミリ秒)
0（送信）	0.014
1（転送）	0.07
2（転送）	5.158
3（受信）	7.996

6.5 IWNR スタックの処理時間

6.3 節の RTT 測定では，図 17 のグラフが示すように 3 ホップ目における RTT の増分が際だっていた．そこで本節では図 17 における実験環境と同様にすべて無線デバイスとして IEEE 802.11b を用い，MAC アドレスによるフィルタリングを行い疑似的な 3 ホップの経路上で IWNR スタック処理時間に焦点を絞り評価を行った．**表 3** に結果を示す．

表 3 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．測定端末は測定した端末を示しており，時間は 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) を表している．表 3 の送信元で

表 4 3 ホップ経路時の受信処理時間
Table 4 Processing time for data receive in 3 hop route.

処理	時間（ミリ秒）	割合
デバイスからの受信処理	0.00189	0.0002
スレッド再開	**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

ある 0 ホップ目の端末での処理の内訳は送信処理のみであり，1，2 ホップ目の端末での処理の内訳は転送処理，宛て先である 3 ホップ目の端末での処理の内訳は受信処理となっている．表 3 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 ミリ秒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，上位層にパケットを渡す IWNR スタックの処理は変化しないため，図 17 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 4** に示す．表 4 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下位 Dep コードがキューに入れる処理である．スレッド再開は下位 Dep コードより Core コードが動作しているスレッドである，図 7 に示す IWNR 独自のスレッドへ再開要求を送信し，実際にカーネルスレッドが再開するまでの時間となる．キューからのデータ取得はスレッドの再開後，下位 Dep と Core コード間に存在するキューからパケットが取得され，パケットの処理関数が呼ばれるまでの時間を，ヘッダ処理は処理関数によりパケットのヘッダを処理後，上位 Dep コードが上位層へパケットを渡すまでの時間となっている．

表 4 に示す測定結果から，受信処理の 0.99 以上は下位 Dep コードがパケットをキューに格納後，Core コードのスレッドに再開要求を発行し実際にスレッドが再開するまでの時間である．すなわち，休眠しているスレッドに対して再開要求が送られても，すぐにスレッドが再開し実行スレッドとしてスケジュールが行われないため，処理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原因は Linux のプロセススケジュールとのタイミングで，再開までに時間がかかってしまう状態で受信

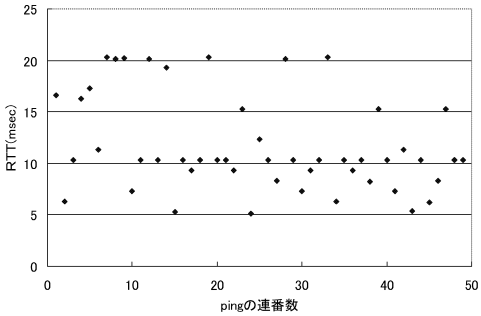

図 23 3 ホップ時の ping による RTT の分散
Fig. 23 Variance of RTT for ping in 3 hop route.

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．**図 23** では 3 ホップ時の RTT の分布を示している．横軸は ping の連番数を縦軸はそれぞれの RTT を示す．このグラフでは標準偏差が 4.66 となり RTT の分散が大きく，RTT の急激な増加はスケジュールタイミングによるものであると考えられる．

表 3 の各端末における処理時間を合計すると約 13 ミリ秒もかかっており，RTT の往復処理時間は約 26 ミリ秒となる．しかし図 12 における 3 ホップの RTT は約 12 ミリ秒となっており RTT の値が一致しない．この原因は表 3 においてスタック処理時間を測定するために，各端末に rdtsc 命令および CPU カウンタを表示させる printf をコードに追加し，それらの測定用命令呼び出しのオーバヘッドに加えスレッド再開のタイミングが変化した現象が予測される．カーネルスレッドによる処理時間の影響は表 3 の 1（転送），2（転送）より明らかである．表 3 の転送処理では 1 ホップ目の端末，2 ホップ目の端末ともに IWNR の処理はまったく同一であるが，処理時間の差は 5 ミリ秒以上も開いており，スレッド再開のタイミングが処理時間に大きな影響を与えることが分かる．

上記の IWNR スレッド再開までに大幅な時間がかかってしまう事象から図 12，図 13 において，IWNR，NO-IWNR の RTT に大きな差があるにもかかわらず，TCP スループットの差が小さい原因が特定できる．現状の IWNR の実装では，Core コードのスレッドは下位 Dep コードによってキューに蓄積された受信データをすべて取得し，1 度に処理を行う．つまりキューに複数の受信パケットが蓄積されていた場合，1 度のスレッド再開で複数のパケットが処理される．RTT の計測では ping プログラムを用いており，1 秒に 1 回の echo 要求パケットを送信するため，1 回のスレッド再開に対し 1 つのパケットが処理される．しかし，TCP データ送受信ではバースト的にデータが

送信されるため，下位 Dep コードと Core コード間でデータ受渡しを行うキューへ多くのパケットが蓄積される．そのため，IWNR は 1 回のスレッド再開に対し多くの送受信パケットを処理でき，IWNR スタック内の処理が効率良く動作する．これら処理動作が原因で図 12，図 17 において，IWNR，NO-IWNR の RTT に大きな差があるにもかかわらず，TCP スループットの差が小さい現象が発生した．

7. まとめと今後の課題

7.1 ま と め

本論文は異種無線デバイス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 は異なる無線ネットワークどうしの透過的な接続を実現する．また IWNR は多種な無線デバイスをサポートする観点から，IWNR を無線デバイスに依存する部分の Dep コードと無線デバイスに依存せず IWNR の機能の中心である Core コードへ明確に分け，実機によって実装を行った．実機上の評価として，Bluetooth と IEEE 802.11b を無線デバイスとして用い，転送性能，遅延の性能を評価した．IWNR は遅延性能において NO-IWNR と比較し大幅な劣化がみられたが，転送性能においては NO-IWNR とほぼ性能が等しく，IWNR が構築するネットワークの転送性能はアプリケーションが動作する環境の許容範囲である．

7.2 今後の課題

今後の課題として以下があげられる．

- 評価環境における規模の拡大
  今回の評価では 4 ホップの経路を構築し，計 5 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て評価環境の規模を拡大し，IWNR の評価を詳細に行う．
- 多種の無線デバイスへの対応
  本論文は IWNR が使用する無線デバイスとして Bluetooth と IEEE 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デバイスへの対応を行う．
- IWNR のアドレス割当て
  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，このアドレス割当ての自動化を行う．

謝辞 本研究の一部は NTT DoCoMo「ユビキタスネットワークにおける機器間の協調と制御に関する研究」，三菱電機株式会社情報技術総合研究所「Federated Sensor Network：自律型センサネットワーク研究」，文部科学省「デジタルメディア・コンテンツ総合研究機構」の下に行われています．

参 考 文 献

1) Hill, J. and Culler, D.: A Wireless Embedded Sensor Architecture for System-level Optimization, Technical report, U.C. Berkeley (2001).
2) 高橋ひとみ，斉藤匡人，間 博人，徳田英幸：異種無線ネットワーク環境における透過的な接続を実現する経路制御機構，情報処理学会システムソフトウェアとオペレーティングシステム研究会，pp.73–80 (2005).
3) Takahashi, H., Saito, M., Aida, H. and Tokuda, H.: A Routing-Centric Approach for Covering Heterogeneous Wireless Networks, *Proc. ICST CONWIN'05* (2005).
4) Luo, H., Ramjee, R., Sinha, P., Li, L.E. and Lu, S.: UCAN: a unified cellular and ad-hoc network architecture, *Proc. ACM MobiCom'03*, pp.353–367 (2003).
5) Draves, R., Padhye, J. and Zill, B.: Comparison of Routing Metrics for Static Multi-Hop Wireless Networks, *Proc. ACM SIGCOMM '04*, pp.171–182 (2004).
6) Broch, J., Johnson, D. and Maltz, D.: The Dynamic Source Routing Protocol for Mobile Ad Hoc Netoworks (2004). IETF Internet-Draft [Work in Progress].
7) BlueZ. http://bluez.sourceforge.net
8) Buffalo Melco WLI-PCM-L11. http://buffalo.melcoinc.co.jp
9) 3COM 3CREB96. http://www.3com.com
10) Brainboxes BL-554. http://www.brainboxes.com
11) l2test. http://www.bluez.org/download.html
12) Netperf. http://www.netperf.org/
13) Padhye, J., Firoiu, V., Towsley, D. and Kurose, J.: Modeling TCP Throughput: A Simple Model and its Empirical Validation, *Proc. ACM SIGCOMM '98*, pp.303–314 (1998).
14) l2ping. http://www.bluez.org/download.html

(平成 17 年 5 月 18 日受付)
(平成 17 年 11 月 1 日採録)

高橋ひとみ

2003 年慶應義塾大学環境情報学部卒業．2005 年同大学大学院政策・メディア研究科修士課程修了．現在，同大学院政策・メディア研究科後期博士課程に在学中．無線ネットワーク，モバイルアドホックネットワークの研究に従事．

斉藤　匡人（学生会員）

2004 年慶應義塾大学大学院政策・メディア研究科修士課程修了．現在，同大学院政策・メディア研究科後期博士課程在学中．ユビキタスアドホックネットワークにおけるセキュリティ基盤技術，ネットワーク情報の三次元視覚化等の研究に従事．日本学術振興会特別研究員（DC2）．ACM，電子情報通信学会各学生会員．

間　　博人（学生会員）

現在慶應義塾大学大学院政策・メディア研究科後期博士課程在学中．センサネットワーク，無線通信，通信プロトコルの研究に従事．IEEE．

戸辺　義人（正会員）

1984 年東京大学工学部電気工学科卒業．1986 年同大学大学院修士課程修了．同年株式会社東芝入社．1997〜2002 年慶應義塾大学にて研究員および特別研究助教授．2000 年博士（政策・メディア）．2002 年から東京電機大学．現在，同大学教授．ユビキタスコンピューティング，センサネットワークの研究に従事．IEEE，ACM，電子情報通信学会，計測自動制御学会各会員．

徳田　英幸（正会員）

慶應義塾大学より工学修士．カナダ，ウォータールー大学より Ph.D.（Computer Science）．現在，慶應義塾大学大学院政策メディア・研究科委員長，同大学環境情報学部教授．分散リアルタイムシステム，マルチメディアシステム，通信プロトコル，超並列・超分散システム，モバイルシステム等の研究に従事．IEEE，ACM，日本ソフトウェア科学会各会員．