情報処理学会論文誌 Vol. 50 No. 10 2556–2567 (Oct. 2009)

推薦論文

ワイヤレスセンサネットワークの設計開発支援環境 D-sense

森 駿介[†1] 梅津高朗[†1,†2] 廣森聡仁[†1,†2]
山口弘純[†1,†2] 東野輝夫[†1,†2]

ワイヤレスセンサネットワーク（WSN）のプロトコルやアプリケーションの開発においては，開発工程を複雑にする多くの要因がある．たとえば，プロトコル実装時には抽象度の高いアルゴリズム記述を低レベルのセンサノード実装言語を用いてコーディングする必要がある．また大規模な WSN ではテストベッドを構築し，多数のセンサノードを操作しながら性能評価を行うことも困難となる．そこで本論文では WSN プロトコル開発を支援する統合開発環境 D-sense の設計開発について述べる．D-sense では，多くの WSN プロトコルに共通する動作を API として提供することで，NesC などの低レベル言語での実装コード量を減少させる．それらの API を用いて記述された NesC の実装コードをシミュレータ用のコードに変換する機能を提供し，シミュレーション実験にともなう設計者の負担を低減させる．また，WSN アプリケーションのモニタリング，ロギング，デバッグといった支援機能も提供する．D-sense を用いて既存のいくつかのプロトコルを実装し，シミュレーションと実環境で性能評価を行うことで，D-sense の利便性を示した．

A Design Support Environment for Wireless Sensor Networks

Shunsuke Mori,[†1] Takaaki Umedu,[†1,†2]
Akihito Hiromori,[†1,†2] Hirozumi Yamaguchi[†1,†2]
and Teruo Higashino[†1,†2]

In order for designing wireless sensor network systems, we need to describe their low-level codes and analyze their performance. In addition, it is not easy to operate a number of nodes manually. In this paper, we propose a design support environment for wireless sensor networks. By providing algorithm-level APIs, we try to hide distributed, low-level operations in the NesC programming language. The algorithm-level APIs and other NesC codes can automatically be converted into simulator codes. The proposed support environment also provides functions for monitoring, logging and debugging of distributed programs. We have implemented some protocols in the proposed environment, and evaluated its usefulness.

1. まえがき

ワイヤレスセンサネットワーク（WSN）は，センサノードの通信，処理性能，電力容量といったアーキテクチャだけでなく，ネットワークの規模や対象アプリケーションなど，様々な面で従来の有線ネットワークと大きく異なる．また，利用環境やハードウェアも様々であることから，新規にプロトコルの設計開発が必要となることも多い．それゆえ，多くのプロトコルが様々な環境や目的を想定して設計されてきている[1)–8)]．しかし，WSN だけでなく，一般に分散システムの設計は様々な問題や困難をともなう．たとえば，プロトコルの設計段階はアルゴリズムレベルの振舞いのみに集中して設計を行えることが望ましいが，実際には対象となる環境に依存した実装レベルのコーディングも必要となる．また，プロトコルの性能分析や性能検証シミュレーションを行う際，多くの場合は実機用コードとシミュレータ用コードの互換性がないために改めてシミュレータ用にプロトコルの実装を行う必要がある．また，実環境実験は多数のセンサノードの設置，動作のロギング，実装の検証やデバッグのための操作が必要となるが，これらは煩雑であり多大な労力を要する．

そこで本論文では，WSN プロトコルの開発を支援する統合環境 D-sense を提案する．D-sense は，プログラミング言語 NesC を対象としているが，その概念は特定言語に依存せず，多くの WSN システムに対して適用が可能である．D-sense は WSN 開発における以下の支援を行う．

- 既存の WSN プロトコルを分析し，それらに共通する典型的な処理をモジュール化し，その API を提供する．開発者はこの API を用いることで，アルゴリズムに着目し本質

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†2 独立行政法人科学技術振興機構，CREST
Japan Science Technology and Agency, CREST
本論文の内容は 2008 年 7 月のマルチメディア，分散，協調とモバイル（DICOMO2008）シンポジウムにて報告され，同プログラム委員長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

的でない処理を隠蔽した実装が可能となる．

- シミュレーション実験と実環境実験のシームレスな連携を実現する．そのために NesC のコードを QualNet シミュレータ[9] 用のコードへ変換するトランスレータや，実環境実験でのノード配置や電波受信状況（電波受信強度）をシミュレータに反映させ，より実環境に近い環境をシミュレータで再現する機能を提供する．これにより，たとえば小規模 WSN での実験を実環境で行い，その結果をもとに中から大規模 WSN での実験をシミュレーションで継続するといったシームレスな連携が可能となる．
- センサノードのモニタリングや実行時の動作制御を可能とする機能を提供する．この機能による WSN プロトコルのテストや保守の支援の手法については 3.3 節で述べる．

D-sense の提供する API を用い，GPSR[1]，SPEED[2]，BIP[3]，Rumor Routing[4] といったプロトコルの実装を行った．特に SPEED について実環境とシミュレーションによる性能評価実験を行い，文献 2) で報告されている評価と比較することで，D-sense の実装の妥当性を示す．また記述コード量の解析を行い，D-sense が WSN プロトコル開発に要する労力の軽減に有効であることを示す．

2. 関 連 研 究

MoteLab[10] や CitySense[11] のような大規模テストベッドは無線端末へのプログラムの配布やインターネットを介した端末の共有の実現を目標としている．D-sense はプロトコルの設計から性能評価までを総合的に支援することを目指しているという点においてこれらと異なる．

TinyDB[12] や COUGAR[13] はセンサネットワークにおけるクエリ処理を行うプロトコル設計の支援を目的とし，イベントの取得や検索処理を実装する SQL ライクな API を提供している．MATE[14] はさらに一般的な用途のための API を提供しているが，たとえばイベントの検知，データのスタックへのプッシュ，データの送信といった実装レベルに限定されたものである．それらに対し D-sense は，既存の WSN プロトコルを分析および分類し，それぞれの分類に典型的な動作を抽出することで，実装レベルの処理を適切に隠蔽しプロトコルのアルゴリズム設計に集中できるような API 群を提供する点が異なる．たとえば，GPSR や SPEED のように Greedy Forwarding を行う位置情報ルーティングでは，(i) 隣接ノードの位置情報を得る，(ii) 自ノードから隣接ノードおよび目的ノードへの距離を計算する，(iii) 自ノードよりも目的ノードに近く，かつその目的ノードに最も近いノードを選択する，(iv) パケットを送信する，といった処理が共通している．D-sense はこれらのような一連の処理をパッケージ化し，それぞれを API として提供する．

D-sense は WSN プロトコルの開発において，アルゴリズム設計から実装レベルのコーディング，シミュレータと実環境端末のシームレスな連携，およびネットワークを介したデバッグやモニタリングまでを対象とした包括的な支援を行うという既存のアプローチには見られない新規性を有し，また実機実験による有用性も示している．

3. D-sense の設計

ユーザは D-sense の提供する実装支援 API を利用して，アルゴリズムレベルのプログラムを記述するだけで，シミュレーション，実環境の双方で WSN プロトコルを評価することができる．加えて，管理支援 API を利用してシナリオを記述することで，デバッグも容易に行うことができ，モニタリングや保守などの運用に利用することもできる．以降では D-sense の提供する機能について述べる．

3.1 実 装 支 援

D-sense の提供する主な機能の 1 つが，アルゴリズム設計の支援である．開発者は D-sense の実装支援 API を利用してアルゴリズムレベルでコードを記述し，API translator が API を実装コードに置き換えることで NesC によるプロトコルの実装を得られる．可能な限り多様なプロトコルの実装を支援するため，既存の典型的なプロトコル，特にルーティングプロトコルの分析および分類を行い，それに基づいた実装支援 API を設計した．この分析および分類は，文献 15) において行われている，ネットワークの構造（**図 1** の “Routing

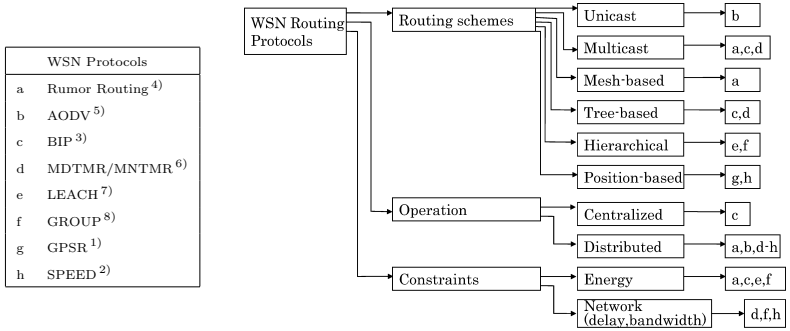


図 1　既存 WSN ルーティングプロトコルの分類
Fig. 1　Classification of WSN protocols.

schemes”），およびプロトコルの動作（図 1 の “Operation”）の 2 つの観点に基づいた分類を参考に，ネットワークに対する制約（図 1 の “Constraints”）に基づいた分類も加えるなど，WSN プロトコルの特徴を考慮して行った．この分類例を図 1 に示す．たとえば，GPSR（図 1 中の “g”）は位置情報を用いるルーティングであり，GHT [16] など位置情報ベースのデータ管理手法においても利用されるが，このプロトコルの実装には “position-based routing” を対象とした API を利用できる．同様に，Rumor Routing [4]，AODV [5]，BIP [3]，MDTMR [6]，LEACH [7]，GROUP [8] および SPEED [2] といった代表的なプロトコルも図 1 のように分類できる．

このような分類に基づき，その分類特有の動作や，各分類に共通して利用される汎用的な動作，さらにそれらの組合せによる複雑な動作を指定可能な API を設計した．たとえば，多くのプロトコルで利用される汎用的な動作である隣接ノードの ID を取得する機能を API として設計した．一方，位置情報ベース（図 1 の “Position-based”）のプロトコルやアプリケーション特有の動作である，指定ノードの位置を取得する機能，および，いくつかのクラスタベース（図 1 の “Hierarchical” に属する）のプロトコルで用いられる，クラスタ内で電力残量が最も大きいノード ID を得るような処理などの機能も API として提供する．これらのような主な API を**表 1** に示す．API の詳細は Web サイト [17] で公開している．

4 章では，これらの API を用いた既存プロトコルの実装例をいくつかあげて，支援の有用性を示す．

3.2　シミュレーションと実環境実験のシームレスな連携による実験支援

D-sense はシミュレーションと実環境実験との間の連携を行い，性能評価を支援するための機能の提供を行う．連携機能として，(1) 2 者間でのコード共有，(2) アニメータによる実環境実験結果の可視化，および (3) 実環境のログをもとにしたシミュレーションのパラメータ設定，があげられる．

コード共有機能により，API translator により生成された NesC のコードはそのまま Mica MOTE 上で実行可能であるうえ，NesC translator によって QualNet シミュレータ [9] 用の C++コードを生成することも可能である．可視化機能は，実環境のセンサノード群の動作や状態を QualNet アニメータで表示することができる（**図 2**）．より分かりやすく表示するため，バッテリの電力残量や LED の点灯状態（Mica MOTE に対応）を表すグラフィックを提供する（**図 3**）．この機能は，実環境で動作させる Mica MOTE に搭載するプログラムに管理用コンポーネントを追加し，環境依存のメッセージ送受信などのイベント，センサノードの電力残量や位置などの内部状態のログの取得や集約を，それぞれのノード上のコンポーネントが協調して行うことにより実現する．

表 1　実装支援 API の一部
Table 1　Example of D-sense design APIs.

Generic APIs
Get the IDs of the one hop neighbor nodes
-> get_neighbors(IDs[],len_IDs)
Send a packet to the designated node
-> send_unicast_packet(ID,pkt)
Position based protocol APIs
Get the position of the designated node
-> get_position(ID)
Compute the distance between the designated two nodes
-> get_distance(ID,ID)
Hierarchical protocol APIs
Get the IDs of the cluster members
-> get_cluster_nodes(IDs[],len_IDs)
Get the cluster head of the neighbor cluster
-> get_neighbor_clusterhead()
Energy Constraint Protocol APIs
Get the residual battery of the designated node
-> get_residual_energy(ID)
Get the energy consumption to send a packet
-> get_transmission_energy()
Network Constraint Protocol APIs
Get the delay between the designated two nodes
-> get_delay(ID,ID)
Get the packet loss ratio at the designated node
-> get_packet_loss_rate(ID)
Functional (Combined) API
Get the ID of the maximum residual battery node in a cluster
Get the minimum delay node which has the specific data
Get the packet loss ratio in the tree

また，パラメータ設定機能は，実環境で得たノードの位置情報や受信メッセージの電波強度のログをもとに，実環境での設定や電波状況を継承したシミュレーション環境へのシームレスな移行を実現する．これにより，小規模ネットワークでは実環境で評価し，中規模から大規模ネットワークでは同様の設定でシミュレーションにより評価するといったことが容易に行える．

特に電波強度に関して，すべての方位に対して均等に減衰が起こる一般的な電波モデル

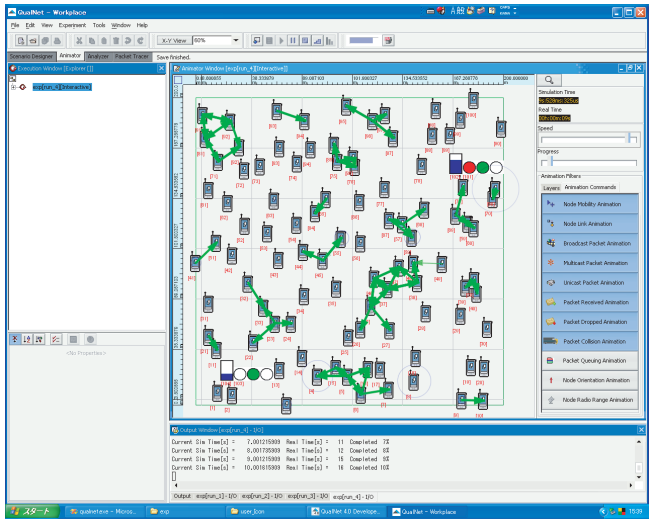


図 2　QualNet シミュレータによるセンサノード状態の表示
Fig. 2　A snapshot from QualNet Simulator (sensor node status is visualized).

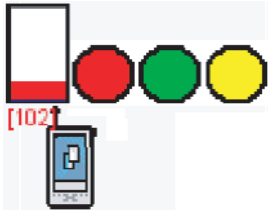


図 3　Mica MOTE の状態を可視化するための LED・バッテリ残量表示機能
Fig. 3　Visualization of LED and battery for Mica MOTE.

によるシミュレーションでは，実機から得られる性能との間に大きなギャップが生じることが確認されている[18]．この問題に対して，同文献では方位によって減衰率が変動する RIM（Radio Irregularity Model）を提案している．これは単位角度ごとの最大変位 DOI（Degree of Irregularity）をパラメータとして持ち，適切な値を与えることで実機とシミュレーションのギャップを埋めることを目的としている．D-sense では，実環境上のいくつかのノードのログをもとに DOI 値を導出し，これを反映した RIM を適用したシミュレーションを実行できる機能を提供する．DOI の導出は次のように行う．各ノード i に対し，そのノードからほぼ等距離にある隣接ノード群 $N(i)$ を，ノード i が持つ各隣接ノードとの位置関係の情報をもとに求める．ノード i からのメッセージのブロードキャスト送信に対し，それを受信した $N(i)$ に含まれる各ノードは受信電波強度を記録する．それらの値をもとに，方位に対する電波強度の差異を反映した適切な DOI 値を設定する．

3.3　プロトコルのデバッグおよび管理支援

ここでは WSN プロトコルのデバッグおよび管理の支援について述べる．この目的のため，それぞれのノード上ではデバッグエージェント，基地局ではデバッグコントローラと呼ばれるモジュールを常駐動作させる．デバッグコントローラとデバッグエージェントとの協調により分散環境でのデバッグを実現する．

デバッグ支援では，開発者が，どのノードでどのような条件が成立した場合にセンサネットワーク全体でプログラム実行を停止したいか，また停止後にはどのノードがどのような処理を行うべきかについて記述する．この記述をデバッグシナリオと呼ぶ．具体的には，デバッグシナリオは文の集合からなり，1 つの文は複数のノードの内部変数やそれらのノードにおいて発生するイベントについての条件式と，その成立時に実行する処理（アクションと呼ぶ）を記述した動作記述から構成される．

条件式は，基本的にノード名とそのノードで成立判定が可能な部分条件（これを局所条件と呼ぶ）の論理結合の形で記述する．これに対し，各ノード上のデバッグエージェントは，そのノードの局所条件が成立した場合には，基地局のデバッグコントローラへその旨を通知する．基地局上のデバッグコントローラは，条件全体が成立するかを各ノードからの通知に基づきつねに判断している．

ただし，WSN では通常データセントリックの設計概念が適用される場合が多い．すなわち，個々のノードは特定の識別子を持たず，各ノードが自律的に役割を決定し目的とするデータ処理を行う．この概念により設計されたプロトコルのデバッグに柔軟に対応するため，デバッグシナリオではノードを指定する方法を工夫している．まず，任意のノードを表すシンボル $ を利用できるものとする．ノード名として$が指定された局所条件は，すべてのノードがそれぞれその条件判定を行い，成立した場合はその事実をデバッグコントローラに通知する．デバッグコントローラはその通知の送信者を $ に割り当て，アクションを実行する．さらに，ノード名の代わりに，ある条件を満足するノードの集合を指定することもできる．この場合は，各ノードがそのノード集合に自身が含まれるか否かを自身のみで判定できることが前提となる．たとえば，ある正方領域に存在する各ノードを局所条件の判定

表 2　管理支援 API の一部
Table 2　Example of D-sense Debug APIs.

ノード指定のための API
2 点（絶対座標で指定）を対角線とする正方形領域内のノードを指定.
-> region_square(position, position)
1 点（絶対座標で指定）から距離 n 以内の領域にあるノードを指定.
-> distance_within(position, distance)
あるノードから n ホップ圏内にあるノードを指定.
-> hop_num_within(ID, hops)
あるノードと同じクラスタに所属するノードを指定.
-> cluster_member(ID)
条件指定のための API
ループを検出した時に真
-> on loop detected()
動作指定のための API
一定時間経過後から実行
-> start_timer(msec)
一定時間経過後まで実行
-> end_timer(msec)

ノードとして指定することもできる．この例では，各ノードが位置情報を保持していれば，その判定は自身で可能である．また，ノードの位置関係，たとえばあるノードからのホップ数制限を利用したノード集合指定を行う場合は，位置関係（この例ではホップ数）をあらかじめ何らかの方法で把握していることが前提となる．

また，条件式は各ノード上の NesC コード中の変数や関数名を参照して記述することができる．たとえば，条件式中に NesC コード中の関数名を記述すると，その関数の実行時に値が真となり，ノードの動作状況に応じたデバッグシナリオの実行が可能となる．

条件成立後のノード動作（アクション）の記述は，同様にアクションを実行すべきノード名とその動作を記述する．アクションは NesC に基づく記述が可能で，表 1 で例をあげた実装支援 API を用いることもできる．また，アクションの開始時間の指定などを記述するための API も用意する．シナリオの記述に用いることのできる管理支援 API の一部を**表 2** に示す．

3.4　デバッグ・管理支援の例

本節では，D-sense のデバッグ支援機能を用いたデバッグや管理がどのように行われるかについて，2 つの例を用いて述べる．

以下は，ルーティングにおけるループ検出を行うシナリオ例である．

```
$.on_loop_detected ->|
  nodeIDs = $.received_packet->recent_visit_nodes;
  foreach u in nodeIDs {
      u.refresh_table();
  }
```

このデバッグシナリオでは，1 行目が条件を表し，2 行目以降が条件成立後のアクションを表す．前述したように，各条件およびアクションでは，変数および関数の前にノードの指定を記述することで，それらを判定および実行するノードを明記する．なお，指定なしの変数や関数は基地局上のデバッグコントローラで保持されていることを意味する．このシナリオは，“$” で記されるノード（すなわち任意のノード）において関数 on_loop_detected が呼び出された場合（この関数はルーティングメッセージが同じノードを経由する場合に呼び出される），“$” ノードの保持する構造体 received_packet に含まれるノードのリスト recent_visit_nodes を，基地局の持つリスト nodeIDs に代入する．そして，nodeIDs に含まれる各ノードに関数 refresh_table() を実行させる．これは，ルーティングプロトコルにおけるアサーションや事後条件の指定に用いることができる．

以下の例はシステムのモニタリングや保守を目的としたものである．

```
$.region_square(a,b) && ($.residual_energy < 0.2) -> |
      $.beacon_interval = $.beacon_interval * 2;
```

このシナリオは，地点 a，b を対角線上の 2 頂点とした正方形の領域内に含まれるノード（それらのノードは “$region\_square(a, b)$” により得られる）であり，その電力残量 residual_energy の値が 20% 以下であれば，ノードの持つ変数 beacon_interval の値を 2 倍にする．beacon_interval の値に応じてビーコンの発信間隔が決定されるものとすれば，このシナリオの導入によってその領域のネットワークの寿命を延ばすことができる．

これらのシナリオは，デバッグコントローラとデバッグエージェントによって実現するが，データや通知の交換や処理実行のプロトコルはさらに検討の余地がある．1 つ目のシナリオを例として，**図 4** にエージェントとコントローラによる分散処理の様子を示した．たとえば，動作指定に含まれる次の部分を例にあげる．

```
nodeIDs = $.received_packet->recent_visit_nodes;
```

nodeIDs はコントローラ上で管理する変数であるが，右辺の変数は “$” ノード内で保持されている．どのノードも条件を満たせば “$” ノードとなるため，各ノード上のエージェントは関数 on_loop_detected が実行された場合，つねにコントローラへ通知を行う．通知を

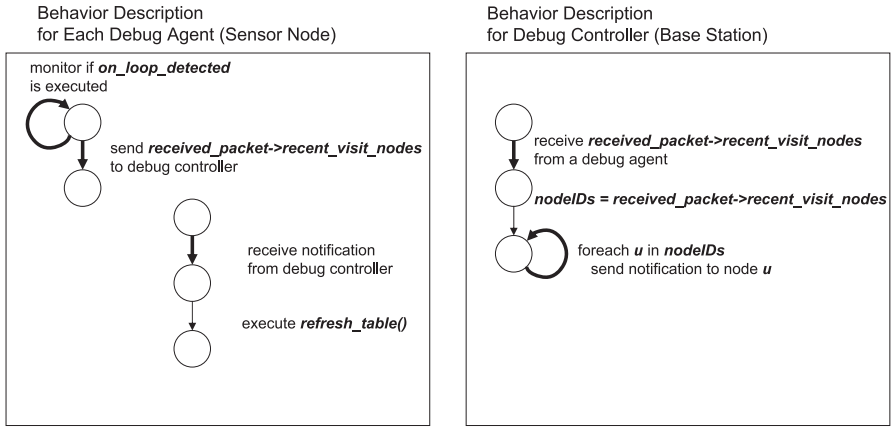


図 4　デバッグシナリオの分散的実行例
Fig. 4　Distributed execution of debug scenario.

受け取ったコントローラは nodeIDs の値を更新する．現在はこの例のように基地局の集中管理型による実現を行っているが，基地局が離れていたり，局所条件が頻繁に成立および不成立を繰り返したりする場合，基地局への通知のためのトラフィックが増大する．したがって可能な場合はノード間協調で条件判定するなど，トラフィックオーバヘッドを考慮してどのノードが何をすべきかが自動的に決定できることが望ましい．我々は現在，文献 19) などのサービス自動分散化技術を応用してこれらを実現する方法を検討中である．

4. WSN プロトコルの実装

この章では既存の WSN プロトコル GPSR，SPEED，BIP，Rumor Routing の実装支援 API を用いた実装例を示す．

GPSR[1] は，平面的グラフを用いて目的地に最も近い隣接ノードに対してパケットの転送を行う位置情報ベースのプロトコルである．**図 5** (a) は，よく知られた平面的グラフの 1 つである Relative Neighborhood Graph（RNG）を生成するアルゴリズムの実装である．たとえば，隣接ノードのリストを取得する “get_neighbors”（02 行目）やノード間の距離を取得する “get_distance”（07 行目）といった API を用いることで，アルゴリズムは簡潔に実装されている．図 5 (b) は GPSR のルーティング処理の実装例である．各ノードは隣接ノードのリストを作り（01 行目），パケットを目的地に最も近いノードへと転送する（05–09 行目）．

```
01: get_planar_graph(graph g){
02:     len = get_neighbors(neighbor_IDs, sizeof(neighbor_IDs));
03:     for (i = 0; i < len; i++){
04:         nodeID = neighbor_IDs[i];
05:         for (j = 0; j < len; j++){
06:             nodeID' = neighbor_IDs[j];
07:             if (get_distance(myID,nodeID)
                    > max(get_distance(myID,nodeID'),
                          get_distance(nodeID,nodeID')))
08:                 g.remove_edge(myID,nodeID);
09: }
```

(a) RNG Generation

```
01: len = get_neighbors(neighbor_IDs, sizeof(neighbor_IDs));
02: forwardID = get_my_ID();
03: for (i = 0; i < len; i++){
04:     nodeID = neighbor_IDs[i];
05:     if(get_distance(nodeID,targetID)
            < get_distance(forwardID,targetID))
06:         forwardID = nodeID;
07: if(forwardID != myID) // forward toward a nearer node
08:     send_unicast_packet(forwardID,packet);
09: else peremeter_mode == true; // perimeter mode (omitted)
```

(b) Greedy Forwarding

図 5　GPSR 実装例
Fig. 5　Example implementation of GPSR.

SPEED[2] もまた位置情報ベースのルーティングプロトコルである．SPEED では Stateless Non-deterministic Geographic Forwarding（SNGF）アルゴリズムによって目的地により近く，かつパケット転送のトラフィックが小さいノードを選択している．**図 6** は SNGF の実装例である．パケットを受信したノードは自分の位置よりも目的地に近いノードを発見し（02–07 行目），2 つのグループへ分類する．ノードの伝送効率が閾値 *set_point* より大きい場合にはグループ *FS_first*，その他のノードはグループ *FS_second* へ割り振る（08–13 行目）．伝送効率は，伝送の経路長や輻輳のレベルなどを基準として決定する（15–26 行目）．ともに位置情報ベースのルーティングプロトコルである GPSR や SPEED の実装は，同様の API を用いて行うことができる．

BIP[3] はセンサノードをツリー状のトポロジとして構築する集中管理型のプロトコルである．ソースノードを根としてツリーを構築し，最小の送信電力でパケットが届くノードを 1 つずつ順に加えていくことにより，全ノードでツリーを構築する．また，祖先ノードの無

```
01:  SNGF(message){
02:     my_length = get_distance(message->target_ID, myID);
03:     len = get_neighbors(neighborIDs, sizeof(neighborIDs));
04:     num_FS_first = 0; num_FS_Second = 0;
05:     for (i = 0; i < len; i++){
06:        nodeID = neighbor_IDs[i];
07:        diff = my_length
             - get_distance(message->target_ID, nodeID);
08:        if(diff > 0){
09:           if(diff / get_delay(myID,nodeID) > set_point)
10:              FS_first[num_FS_first++] = nodeID;
11:           else
12:              FS_second[num_FS_Second++] = nodeID;
13:        }
14:     }
15:     if(num_FS_first > 0){
16:        forwarding_probability = 0;
17:        forwarding_nodeID = FS_first[0];
18:        for(i = 0;i < num_FS_first; i ++){
19:           nodeID = FS_first[i];
20:           fp = get_probability
                (get_distance(myNodeID,nodeID)
                ,get_queue_size(nodeID));
21:           if(fp > forwarding_probability){
22:              forwarding_probability = fp;
23:              forwarding_nodeID = nodeID;
24:           }
25:        send_unicast_packet(nodeID, message);
26:     }
27:     else // forward toward the nodes in FS_second
28:  }
```

図 6　SPEED 実装例
Fig. 6　Example implementation of SPEED.

線範囲に子孫ノードが含まれているような場合，その間の中継ノードは不要となるため，そのようなノードを除外するといった動作も行う．

集中制御のプロトコルでは，トポロジの構築やルーティングにおいて任意のノードやノード間の情報を利用できるため，集中制御のルーティングプロトコル向け API ではこれらの情報を管理しユーザに提供する．BIP の実装例を**図 7** に示す．具体的には，まずソースノードを根としてツリー構築を開始し（01 行目），ツリーに含まれないノードのうち，ツリーに含まれるいずれかのノードに対して最小電力でパケットが到達するノードから順に選択し，ツリーに追加していく（06–14 行目）．その際，祖先ノードの無線範囲に子孫ノードが含まれている場合，その中継ノードを削除する（17–24 行目）．

Rumor Routing [4] はメッシュ型トポロジベースのルーティングプロトコルであり，データの蓄積や検索を目的として設計されている．イベントテーブルの管理をエージェントを用いて行うのが特徴で，ノードがエージェントを受信したときには，イベントまでのホップ数，およびエージェントの送信元ノードからの方向とホップ数といった，エージェントが保持している情報をイベントテーブルに加える．それと同時に，ノードはイベントテーブルに記録された情報をエージェントに渡し，エージェントが長時間訪れていない他のノードへ転送する．ノードがクエリを受信したときに，そのノード自身が目的のイベントを持っている場合はクエリの処理を実行し，その他の場合はクエリを転送する方向を，クエリの情報をもとに検索する．そのような情報がない場合は，クエリが一定の時間内に訪問していないノードへ転送する．

```
01: num = get_tree_nodes(root_node_ID,
                  treeNodes,sizeof(treeNodes));
02: while (num < N){
03:    parentNode = NULL;
04:    childNode = NULL;
05:    energy = MAX_VALUE;
06:    for (i = 0; i < num; i++){
07:       node = treeNodes[i];
08:       min_ID = get_minimum_energy_node(node->ID);
09:       if(energy > get_energy_consumption(
                        node->ID,min_ID)){
10:          parentNode = node;
11:          childNode = get_node(min_ID);
12:          energy = get_energy_consumption(
                        node->ID,min_ID);
13:       }
14:    }
15:    if(parentNode) add_child(parentNode,childNode);
16: }
17: num = get_tree_nodes(root_node_ID,
                  treeNodes,sizeof(treeNodes));
18: for(i = 1; i < N;i++)
19:   for(j = 1; j < N; j++)
20:    nodeI = get_node(i); nodeJ = get_node(j);
21:    if(has_children(nodeI,nodeJ)){
22:      len = get_children(nodeJ,childrenIDs, sizeof(childrenIDs));
23:      if(is_neighbours(nodeI,childrenIDs))
24:            set_parent(childrenIDs,nodeI);
```

図 7　BIP 実装例
Fig. 7　Example implementation of BIP.

図 8 は Rumor Routing の実装例であり，1–6 行目がエージェント受信時の動作，8–16 行目がクエリ受信時の動作の記述である．エージェント受信時は，エージェントの持つ，イベントまでのホップ数 *num_hops*，およびエージェントの送信元のノード *source_ID* からの

方向とホップ数の情報をイベントテーブルに加える処理を行い（02 行目），イベントテーブルに記録された情報をエージェントに載せ（03 行目），そのうえでエージェントが長時間訪れていないノードへ転送する（04–06 行目）．クエリ受信時に，ノード自身が目的のイベントを持っている場合はクエリの処理を実行を行い（10–11 行目），イベントを持たない場合はクエリの情報をもとに方向を検索し，転送を行う（12–13 行目）．クエリに関する情報がない場合は，クエリが過去一定期間に訪問していないノードを選択し，転送を行う（14–15 行目）．

また，実装支援 API の有用性を示すため，SPEED について，(1) API 使用時，(2) QualNet シミュレータ用 C++，(3) MOTE 端末用 NesC，の 3 つの実装のコードの行数を比較した．**表 3** にそのコード行数を示す．API を用いた 200 行程度の実装行った場合，API を用いない場合の 1,000 行程度に該当する実装が得られた．これより，API により大幅に実装の労力が削減されていることが分かる．

```
01:  on_agent_received(source_ID,agent_packet){
02:      event_table = set_event_distance(
                 agent_packet->num_hops,source_ID);
03:      agent_packet = set_event_info(
                 agent_packet, event_table);
04:      if(agent_packet->ttl-- > 0)
05:          send_unicast_packet(
               get_not_visited_neighbor(agent_packet),
               agent_packet);
06:  }
07:
08:  on_query_received(source_ID,query_packet){
09:      query_packet->ttl--;
10:      if(get_num_hops(event_table,query_packet->data)==0)
11:          doQuery(query_packet)
12:      else if (get_hops(query_packet->data) > 0)
13:          send_unicast_packet(query_packet,
           get_forwarding_direction(event_table,query_packet))
14:      else
15:          send_unicast_packet(
               get_not_visited_neighbor(query_packet),
               queryPacket)
16:  }
```

図 8　Roumor Routing 実装例
Fig. 8　Example implementation of Rumor Routing.

表 3　SPEED プロトコルのコード行数
Table 3　Lines of code of SPEED.

	(1) 実装支援 API	(2) C++	(3) NesC
行数	221	1,044	1,147

さらに，SPEED の実装に用いた API を用いて，SPEED と同様に位置情報ベースのルーティングプロトコルである GPSR の実装も行った．SPEED および GPSR は，ともにパケットの送信先として目的地への距離が最も近くなるノードを選択する Greedy Forwarding に関する処理が含まれるため，API 非使用時は 200 行程度の実装が必要であるが，API を利用した場合は 70 行程度で実装が可能であった．このことは，プロトコルの分類に特徴的な処理を抽出した API の再利用性を示している．

5. 性能評価実験

D-sense の実装の妥当性と有用性を示すため，シミュレーションおよび実環境において D-sense を用いて SPEED プロトコルの性能評価を行い，シミュレーションの性能を文献 2) の報告と比較した．

実験シナリオは文献と同様のものを使用した．このシナリオは SPEED プロトコルの輻輳回避の性能のテストを目的としている．実験環境は**表 4** に示す正方領域で，ノードはランダムに配置されている．ストリームの送信元として領域の左側のいくつかのノードからランダムに送信ノードを選び，ストリームの送信先は領域の右側の基地局となる．送信ノードは 1 packet/sec の一定のビットレートで送信を行う．また，輻輳を発生させるために領域の中央付近から 2 ノードをランダムで選択し，実験時間 150 秒のうち 75 秒から 150 秒の間，その 2 ノード間でフローを発生させる．輻輳度合いの推移による輻輳回避を評価するため，輻輳発生用のフロー（以下，輻輳フロー）は，0 から 100 packets/sec まで 10 刻みの各フローレートを用いて数回ずつのシミュレーションを実行し，基地局へのデータパケットの遅

表 4　実験環境
Table 4　Experimental environment.

	既存報告	シミュレーション	実環境
PHY & MAC	802.11	802.11	802.15.4
帯域	200 Kb/s	200 Kb/s, 250 Kb/s	250 Kb/s
ペイロードサイズ	32 Bytes	32 Bytes	32 Bytes
領域	(200 m, 200 m)	(200 m, 200 m), (20 m, 20 m)	(20 m, 20 m)
ノード数	100	100, 25	25
配置	均一	均一	均一
無線範囲	40 m	40 m, 8 m	8 m

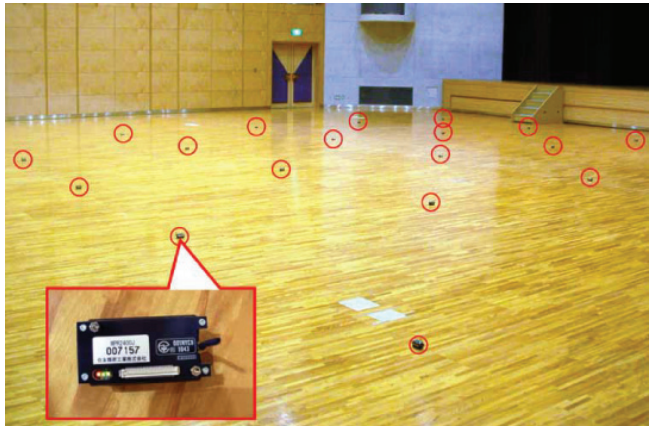


図 9　実環境実験の MOTE 配置
Fig. 9　Arrangement of MOTEs in real environment.

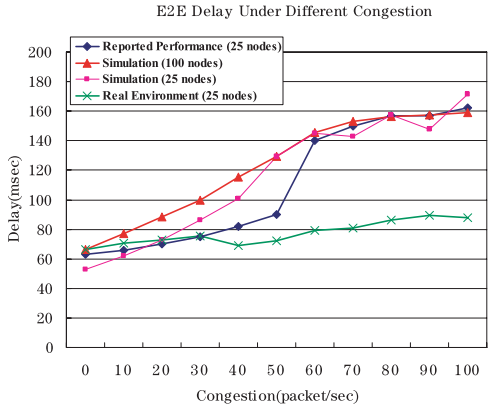
延とロス率を評価する．

実験環境を表 4 に示す．実環境実験では MOTE 端末数に制限があったため，25 ノードによる評価を行った．シミュレーション実験は，比較のために文献 2) による報告，実環境実験の両方それぞれと同様の設定で行った．MOTE とシミュレータの無線範囲の設定は，それぞれ領域のサイズに合わせた値に設定し，シミュレータではまず揺らぎのない理想的な電波到達環境を仮定した．**図 9** は MOTE 端末を均一に配置した実環境実験の様子である．

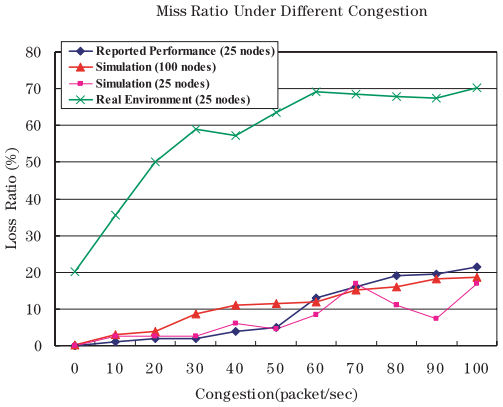
図 10 (a) にエンドノード間のパケット到達遅延を示す．100 ノードの実験では，シミュレーション性能と既存報告性能は輻輳フローが 40 packet/sec 時に若干の差が見られるものの，同様の傾向を示していることが分かる．25 ノードの実験では，シミュレーションでは 100 ノード時と同程度の遅延が観測されたが，実環境実験ではシミュレーションに比べ遅延が小さいことが確認できる．

図 10 (b) はパケットのロス率（基地局に到達しなかったパケット数）を示している．100 ノードの実験では，シミュレーションの性能と既存報告性能はほぼ同程度であることが分かる．25 ノードの実験でも，ほぼ同等の値を示している．

図 10 で見られるように，実環境とシミュレーションの結果を比較すると，実環境の方が遅延が小さく，パケットロス率が高くなっている．これは，実環境では電波の揺らぎが大きいことが原因であると考えられる．実環境では，ノードはより遠くのノードからのビーコン



(a) E2E Delay

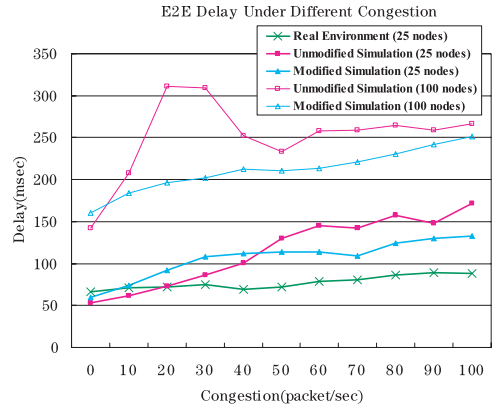


(b) Packet Loss Ratio

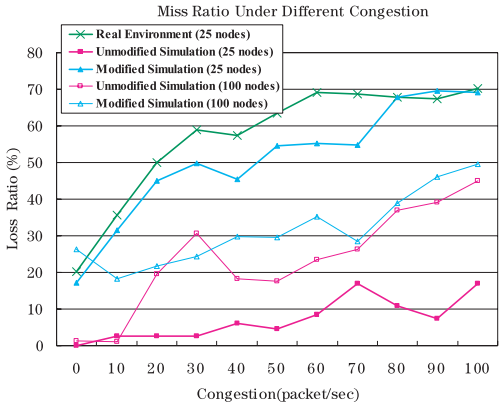
図 10　SPEED プロトコルの性能
Fig. 10　Performance of SPEED protocol.

を受信でき，その発信ノードを隣接テーブルに登録する．そのため，より遠くのノードを選択してパケット送信を行うために遅延が小さくなるが，パケットロスの確率は高くなったと考えられる．

また，このような実環境実験の結果と分析をふまえ，実環境実験に合わせたパラメータ設定を行った．具体的には，機器を直に床に設置していたことをふまえ，アンテナ高を初期値



(a) E2E Delay



(b) Packet Loss Ratio

図 11　設定改善後のシミュレーション結果
Fig. 11　Result of modified simulation.

の 1.5 m から実際の値である 0.1 m に変更し，10 m のノード間距離であってもある程度のパケット受信を確認したことを反映し，通信範囲が 8 m となる初期値 −23.5 dBm から，距離が 10 m であっても同様に受信可能となる 0 dBm へと電波出力を変更した．さらに，パスロスモデルを Flat model から実環境に近い Two Ray へ変更した．また，SPEED プロトコルではパケットの伝送速度に対する閾値を設け，パケットの転送に要する時間や転送先のノードまでの距離などを用いて速度を算出し，閾値を上回るような転送先を選ぶ．この際に用いる値を，実機におけるパケット転送の処理速度を考慮し，100 m/s から 500 m/s へと設定し直した．この設定を用いて，表 4 の実環境実験と同様の配置による評価を行った．その結果，**図 11** のように実環境実験により近づいたシミュレーションの結果を示すことができた．さらに，同様のノード間距離で 40 m 四方の領域に配置した 100 ノードのシミュレーションについても，実環境実験に合わせたパラメータ設定を行った場合と行わなかった場合との比較を行った．その結果，図 11 のように，25 ノード時に大きな差があったパケットロス率の差が小さくなった．これは，25 ノード時はネットワークを構成するノード数が小さく，各ノードの隣接ノードが少ないため，実環境に合わせた設定を行い電波到達距離が大きくなった場合，その干渉により輻輳が起こっている中央付近を回避する経路を得られないという状況が起こっていたと考えられるが，100 ノード時には十分な迂回路が得られるために電波到達距離の差の影響を受けにくくなったためと考えられる．このように，D-sense を用いて実環境実験とシミュレーション実験を並行して行うことで，実環境をより良く再現したシミュレーション環境での評価を行える．また，その環境をより大規模なシミュレーションに適用することにより，環境の差異が様々な規模のネットワークに与える影響を評価できる．

以上の性能評価から，D-sense の実装の妥当性を示すことができた．加えて，実環境特有の問題とその原因の発見，その対処方法についての考察，およびそれらを考慮することでシミュレーションの現実性の改善ができ，実環境で WSN プロトコルを実装し評価することの重要性や，D-sense によるこのような性能評価支援の有用性が示された．

6. まとめと今後の課題

本論文では，WSN プロトコルの設計開発を支援する統合環境 D-sense を提案した．D-sense はプロトコル設計の総合的な支援のため，アルゴリズムレベルの設計を支援する API を提供するとともに，シミュレーションと実環境実験のシームレスな連携，実環境実験におけるデバッグ支援機能を提供する．また，D-sense の効果を示すため，既存の WSN プロトコル GPSR，SPEED，BIP，Rumor Routing の実装を行い，さらに SPEED プロトコルをシミュレーションおよび実環境の両方で性能評価を行った．今後は実装支援 API，管理支援 API や各種機能の充実を図り，一般公開することを目指す．

参 考 文 献

1) Karp, B. and Kung, H.T.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, *Proc. 6th Annual International Conference on Mobile Computing and Networking* (*MobiCom 2000*), pp.149–160 (2000).
2) He, T., Stankovic, J.A., Lu, C. and Abdelzaher, T.: SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks, *Proc. 23rd International Conference on Distributed Computing Systems* (*ICDCS2003*), pp.46–55 (2003).
3) Wieselthier, J.E., Nguyen, G.D. and Ephremides, A.: On the Construction of energy-Efficient Broadcast and Multicast Trees in Wireless Networks, *Proc. 19th Annual Joint Conference of the IEEE Computer and Communications Societies* (*INFOCOM2000*), pp.585–594 (2000).
4) Braginsky, D. and Estrin, D.: Rumor routing algorthm for sensor networks, *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications* (*WSNA2002*), pp.22–31 (2002).
5) Perkins, C. and Royer, E.: Ad-Hoc On-Demand Distance Vector Routing, *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications* (*WMCSA1999*), pp.90–100 (1999).
6) Wei, W. and Zakhor, A.: Multiple Tree Video Multicast Over Wireless Ad Hoc Networks, *IEEE Trans. Circuits and Systems for Video Technology*, Vol.17, No.1, pp.2–15 (2007).
7) Heinzelman, W.R., Chandrakasan, A. and Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks, *Proc. 33rd Annual Hawaii International Conference on System Sciences* (*HICSS-33*), pp.1–10 (2000).
8) Liyang, Y., Neng, W., Wei, Z. and Chunlei, Z.: GROUP: A Grid-Clustering Routing Protocol for Wireless Sensor Networks, *Proc. 2nd International Conference on Wireless Communications, Networking and Mobile Computing* (*WiCOM2006*), pp.1–5 (2006).
9) Scalable Network Technologies, Inc.: QualNet Simulator. http://www.scalable-networks.com/
10) Werner-Allen, G., Swieskowski, P. and Welsh, M.: MoteLab: A wireless sensor network testbed, *Proc. 4th International Symposium on Information Processing in Sensor Networks* (*IPSN 2005*), pp.483–488 (2005).
11) CitySense project: CitySense – An Open, Urban-Scale Sensor Network Testbed. http://www.citysense.net/
12) Madden, S., Franklin, M., Hellerstein, J. and Hong, W.: TinyDB: An Acquisitional Query Processing System for Sensor Networks, *ACM Trans. Database Syst.*, Vol.30, No.1, pp.122–173 (2005).
13) Bonnet, P., Gehrke, J. and Seshadri, P.: Towards sensor database systems, *Proc. 2nd International Conference on Mobile Data Management* (*MDM2001*), pp.3–14 (2001).
14) Levis, P. and Culler, D.: Mate: A Tiny Virtual Machine for Sensor Networks, *Proc. 10th International Conference on Architectural Support for Programming Languages and Operating Systems* (*ASPLOS-X2002*), pp.85–95 (2002).
15) Al-Karaki, J.N. and Kamel, A.E.: Routing techniques in wireless sensor networks: A survey, *IEEE Trans. Wireless Communications*, Vol.11, No.6, pp.6–28 (2004).
16) Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R. and Shenker, S.: GHT: A Geographic Hash Table for Data-Centric Storage in SensorNets, *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications* (*WSNA 2002*), pp.78–87 (2005).
17) D-sense Web: D-sense: An Integrated Environment for Algorithm Design and Protocol Implementation in Wireless Sensor Networks. http://www-higashi.ist.osaka-u.ac.jp/software/WSN/D-sense/
18) Zhou, G., He, T., Krishnamurthy, S. and Stankovic, J.A.: Models and solutions for radio irregularity in wireless sensor networks, *ACM Trans. Sensor Networks*, Vol.2, No.2, pp.221–262 (2006).
19) Yamaguchi, H., El-Fakih, K., Bochmann, G.V. and Higashino, T.: Deriving protocol specifications from service specifications written as predicate/transition-nets, *Computer Networks*, Vol.51, No.1, pp.258–284 (2007).

(平成 21 年 1 月 15 日受付)
(平成 21 年 7 月 2 日採録)

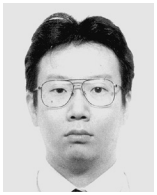
推 薦 文

本論文では実環境およびシミュレーション環境の双方で利用可能なワイヤレスセンサネットワークで使われるプロトコルの設計・開発ツールが提案されている．アルゴリズムレベルで記述できる API を提供することにより，低レベル言語による実装の負荷を抑え，シミュレーションのための実装負荷も抑えている．複数の著名なプロトコルを実際に実装してみて，その性能を実環境およびシミュレーション環境の双方で評価し，その有効性を示しており，ツールとしての完成度は高い．実際に API が公開されれば多くの研究者に有用になると考えられ，ワイヤレスセンサネットワークの研究の発展に大きく貢献する有用性の高いものであり，論文誌の推薦論文としてふさわしい．

（マルチメディア，分散，協調とモバイル（DICOMO2008）シンポジウム
プログラム委員長　串間和彦）

森　　駿介（学生会員）

平成 20 年大阪大学基礎工学部情報科学科卒業．同年同大学大学院情報科学研究科博士前期課進学．ワイヤレスセンサネットワークの設計開発支援に関する研究に従事．



梅津　高朗（正会員）

平成 13 年大阪大学大学院基礎工学研究科博士前期課程修了．同年同大学大学院博士後期課程進学．平成 14 年同大学院博士後期課程退学後，同大学院情報科学研究科助手．平成 19 年より同研究科助教．博士（情報科学）．高度交通システム，アドホックネットワーク用ミドルウェアや開発環境の研究に従事．

廣森　聡仁（正会員）

平成 16 年大阪大学大学院基礎工学研究科博士後期課程修了．平成 17 年株式会社エヌ・ティ・ティ・ドコモ入社．平成 20 年より大阪大学大学院情報科学研究科助教．博士（工学）．モバイルアプリケーションやモバイルネットワークの設計および性能評価に関する研究に従事．IEEE 会員．

山口　弘純（正会員）

平成 6 年大阪大学基礎工学部情報工学科卒業．平成 10 年同大学大学院基礎工学研究科博士後期課程修了．同年オタワ大学客員研究員．平成 11 年大阪大学大学院基礎工学研究科助手．平成 14 年同大学院情報科学研究科助手．平成 19 年より同研究科准教授．博士（工学）．分散システムや無線通信プロトコルの設計および実装に関する研究に従事．IEEE，電子情報通信学会各会員．

東野　輝夫（フェロー）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院基礎工学研究科博士後期課程修了．同年同大学助手．現在，同大学大学院情報科学研究科教授．工学博士．分散システム，通信プロトコル，モバイルコンピューティング等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．本会フェロー．