ルールに基づくユビキタスデバイスのための ネットワークトポロジ発見手法

岸野泰恵[†1] 寺田努[†1] 塚本昌彦[†2]
義久智樹[†3] 早川敬介[†4]
柏谷篤[†5] 西尾章治郎[†1]

本稿では，ユビキタスチップ間のネットワークトポロジを発見する手法を提案する．ユビキタスチップとは，ユビキタスコンピューティング環境を実現するために筆者らが提案している入出力制御デバイスであり，イベント駆動型ルールを組み合わせてその動作を記述する．ユビキタスコンピューティング環境においてはさまざまな通信手段が混在し，アプリケーション要求によって最適な通信手段やプロトコルが異なるため，ネットワークトポロジの発見においても，できるだけ柔軟な方式を採用する必要がある．そこで，本研究ではイベント駆動型ルールを用いた柔軟なネットワークトポロジ発見手法を実現する．提案手法はルールの組合せでトポロジ発見を実現するため，ルールを変更することで状況に応じた柔軟な処理が可能となる．さらに，シミュレータおよびユビキタスチップ実機上で提案するトポロジ発見手法を動作させ，提案手法が有効に働くことを確認する．

A Network Topology Discovery Mechanism for Rule-based Ubiquitous Devices

Yasue Kishino,[†1] Tsutomu Terada,[†1] Masahiko Tsukamoto,[†2]
Tomoki Yoshihisa,[†3] Keisuke Hayakawa,[†4] Atsushi Kashitani[†5]
and Shojiro Nishio[†1]

In this paper, we propose a new network topology discovery mechanism among ubiquitous chips. The ubiquitous chip is a rule-based, event-driven I/O (input/output) control device to compose ubiquitous computing environments. Since this device achieves the flexibility by describing its behavior in a set of rules, we employ a rule-based approach to discover a network topology. In ubiquitous computing environments, although we use various communication methods and application at the same time, our mechanism has enough flexibility to adopt such diversity in ubiquitous computing environments. Moreover, we have verified our algorithm by implementing it on a topology discovery simulator and actual prototype devices of ubiquitous chip.

1. はじめに

近年，コンピュータや半導体，センサなどの小型化により，ユビキタスコンピューティング環境が実現しつつある[4),9),13)]．筆者らの研究グループではこれまでに，イベント駆動型ルールで動作する入出力デバイスを用いた新しいユビキタスコンピューティングを提案している[12)]．このデバイスをユビキタスチップと呼び，将来的には家具や家電製品，壁や床などあらゆるものにユビキタスチップを埋め込むことで，人々の日常生活をサポートすることを想定している．

ユビキタスチップの動作はイベント駆動型ルールで記述する．ユビキタスチップは処理能力が低くメモリ容量も小さいため，ルールの言語仕様はシンプルなものになっている．しかし，ユビキタスチップに保存されたルールはシステム動作中に自由に変更でき，ユビキタスチップに接続されているデバイスもアプリケー

†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 神戸大学工学部
Faculty of Engineering, Kobe University
†3 京都大学学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University
†4 NEC エレクトロニクス第三システム事業本部
3rd Systems Operations Unit, NEC Electronics
†5 NEC インターネットシステム研究所
Internet Systems Research Laboratories, NEC Corporation

ション動作中に自由に変更できるため，複数のユビキタスチップを組み合わせた柔軟なアプリケーションが構築できる．また，動作やデバイス構成を動的に変化させられるため，ユビキタスチップの埋め込まれた家具を動かしたり後からユビキタスチップを追加するといった柔軟な運用が可能となる．

一方，複数のユビキタスチップが連携するようなアプリケーションを構築するためには，ユビキタスチップ間の接続関係であるネットワークトポロジを知ることが必要となる．そこで本稿では，ユビキタスチップのためのネットワークトポロジ発見手法を提案する．提案手法は，ネットワークトポロジの発見にイベント駆動型ルールを用いることで，状況に応じた手法の切替えなど柔軟なトポロジ発見を可能とする．

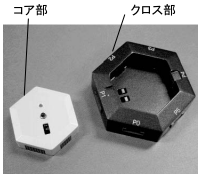
以下，2 章ではユビキタスチップの概要について述べ，3 章ではルールに基づくトポロジ発見手法について述べ，4 章で提案手法の実現について詳細に述べる．5 章では提案手法のシミュレータ上での実装と実機のユビキタスチップを用いた動作確認について述べる．6 章で提案手法について考察を行い，7 章で本稿をまとめる．

2. ユビキタスチップ

2.1 概　　要

ユビキタスチップ（**図 1**）は 5 ポートの入力ポートと 12 ポートの出力ポート，2 ポートのシリアル通信ポート，タイマ，ユビキタスチップ間のメッセージ送受信機能を持つ小型の入出力制御デバイスである．ユビキタスチップはコア部とクロス部に別れ，コア部がマイコンを含むユビキタスチップの主要部分であり，クロス部にはコネクタと充電式のバッテリが格納される．コア部ではマイクロチップテクノロジ社の PIC16F876（プログラムメモリ：8 KB，RAM：368 B，EEPROM：256 B）を 20 MHz で動作させて用いている．**図 2** に示すように，これらのポートにさまざまな入出力デバイスや他のユビキタスチップを接続する[12]．

ユビキタスチップの動作はイベント駆動型ルールで記述する．ルールにはイベント駆動型データベースの分野で利用されている ECA ルール[14] を単純化した形式を用いる．ECA ルールはイベント，コンディション，アクションの 3 つを一組にして動作を記述する言語であり，イベントとしては**表 1** に示すようにメッセージの受信，タイマの発火といった条件を，コンディションとしては入力状態と内部状態を，アクションとしては**表 2** に示すように出力やタイマの設定，メッセージやコマンドの送信などの操作を記述できる．



項目	仕様
動作電圧	2.9〜6 V
電池	4.3 V の充電池
入力端子	5 個
出力端子	12 個
通信ポート	2 個
使用マイコン	PIC16F876

図 1 ユビキタスチップ
Fig. 1 Ubiquitous chip.

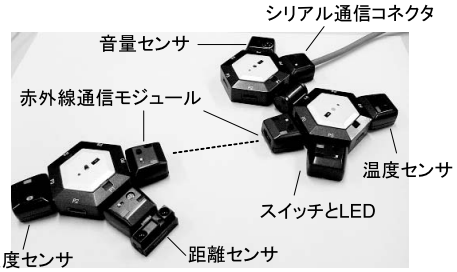


図 2 ユビキタスチップと周辺デバイスの接続例
Fig. 2 Attachments.

表 1 イベント
Table 1 Events.

名前	内容
RECEIVE_MESSAGE	メッセージの受信
RECEIVE_DATA	1 バイトのデータの受信
TIMER	タイマの発火
NONE	なし（コンディションのみの評価）

表 2 アクション
Table 2 Actions.

名前	内容
OUTPUT	出力ポートの制御
OUTPUT_STATE	状態変数の制御
TIMER_SET	タイマの設定
SEND_MESSAGE	メッセージの送信
SEND_DATA	1 バイトのデータの送信
SEND_COMMAND	コマンドの送信
HW_CONTROL	ハードウェア制御

ECA ルールは文献 12) で規定したフォーマットから数個のイベントとアクションを拡張したものに従って 16 進数に変換し，ユビキタスチップの EEPROM に格納する．コンディションとアクションの種類によってルールの長さが変わるが，最短のルールは 4 バイトであり，数十個のルールを格納できる．

2.2 通 信 機 能

ユビキタスチップは他のユビキタスチップとシリアル通信ポートを介して通信を行う．これまでに，有線の通信ケーブルだけでなく，シリアル通信ポートに接続して使用する赤外線通信モジュール，微弱無線モ

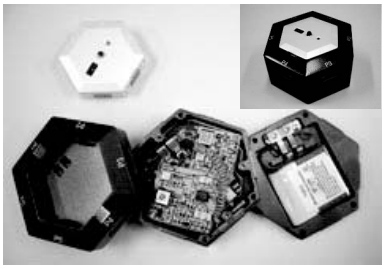


図 3 微弱無線通信モジュール
Fig. 3 Wireless communication module.

表 3 無線通信モジュールの仕様
Table 3 Specification of wireless communication modules.

種類	詳細	通信範囲
微弱無線	300 MHz 帯	数十 cm
Bluetooth	仕様 1.1	数 m
赤外線	IrDA 準拠のモジュール	30 度，約 1 m

ジュール，bluetooth モジュール[11] の試作を行っている．**図 3** に試作した微弱無線モジュールの概観を示し，**表 3** に各モジュールの仕様を示す．赤外線通信モジュール，微弱無線モジュールは無手順のシリアル通信を採用しているため，パケットのヘッダとフッタをチェックしエラーを防いでいる．ユビキタスチップでは，これらの通信モジュールを組み合わせ，それぞれの特性を活かしたアプリケーションの構築が可能である．

通信に関するアクション

ユビキタスチップは SEND_MESSAGE アクションと SEND_DATA アクションを用いて通信を行う．SEND_MESSAGE アクションは 8 種類のメッセージ（ID0 から 7）を送信し，SEND_DATA アクションはルールで指定した任意の 1 バイトのデータやアナログ入力ポートの値を送信する．前者は他のユビキタスチップとやりとりするために用い，後者はセンサの値を他のユビキタスチップやサーバに伝えるために用いる．さらに SEND_COMMAND アクションがあり，このアクションはユビキタスチップのルールの追加，削除，有効化，無効化といった操作を行うコマンドを送信する．**表 4** にコマンドの一覧を示す．これらのコマンドを用いることで，新たな ECA ルールの追加，削除，有効化・無効化といった操作を行う．これらのコマンドを他のユビキタスチップへ送信することで，ユビキタスチップどうしが互いの動作を制御することも可能である．

マルチホップの通信機能

ユビキタスチップの通信にはシングルホップモードとマルチホップモードの 2 種類のモードがある．シングルホップモードでは，送信元のユビキタスチップの通信範囲内にあるユビキタスチップすべてが通信を受信する．マルチホップモードでは，宛先および経路になるすべてのユビキタスチップの ID を順に指定し，宛先までデータを転送する．経路としてはすべてのユビキタスチップを意味する放送 ID を設定することもできる．マルチホップモードを利用することで，直接通信が行えない離れた位置にあるユビキタスチップとも通信できるようになる．

表 4 SEND_COMMAND で送信するコマンド
Table 4 Commands for the SEND_COMMAND.

名前	内容
ADD_ECA	ECA ルールの追加
DELETE_ECA	ECA ルールの削除
ENABLE_ECA	ECA ルールの有効化
DISABLE_ECA	ECA ルールの無効化
REQUEST_ECA	ECA ルールを要求

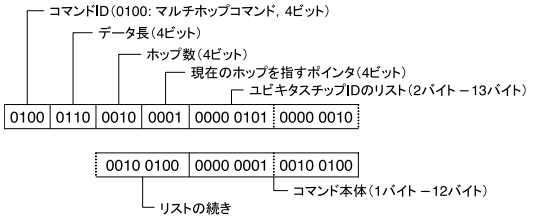


図 4 マルチホップパケットのフォーマット
Fig. 4 Format of a multihop packet.

マルチホップ通信を行う場合には，ユビキタスチップが送信するメッセージや 1 バイトのデータ，コマンドにマルチホップ通信に必要なヘッダを付け加える．マルチホップヘッダには，宛先および経路となるユビキタスチップの ID のリスト，ID の個数，現在の ID を指すポインタが含まれる．マルチホップパケットのフォーマットを**図 4** に示す．図 4 は，ID4 のメッセージを 2 ホップ先のユビキタスチップに送信する例としている．ID に特定の値（放送 ID）を指定した場合，この ID はすべてのユビキタスチップの ID を意味する．ユビキタスチップがマルチホップ通信を受信したときには，以下の手順で処理が行われる．

Step1 現在のポインタが指す ID を調べ，放送 ID であるか，自分自身の ID であれば次のステップへ進む．どちらにもあてはまらなければ，処理を終了する．

Step2 ポインタがリストの最後を指していない場合は，ポインタがリストの次の ID を指すようにヘッダを書き換え，次のユビキタスチップへ転送する．ポインタがリストの最後を指す場合は，次のステップへ進む．

Step3 リストの最後の ID と受信したユビキタスチッ

プの ID が一致するか，最後の ID が放送 ID であれば，自身が宛先のユビキタスチップであるため，メッセージやデータ，コマンドを処理する．

ユビキタスチップが送受信するパケットには最大のパケット長は 15 バイト以内であるという制限がある．たとえば，ADD_ECA コマンドを用いて 4 バイトのルールを送信することを考えた場合，1 バイトのマルチホップヘッダと 6 バイトの長さのコマンドが必要となるため，パケット長を 15 バイトに抑えるためには 1 バイトの送信元 ID を除くと，7 ホップ分（7 バイト分）の経路しか記述できない．このため，遠く離れたところにあるユビキタスチップへ直接ルールを書き込むことはできない．

3. ルールに基づくトポロジ発見手法

本研究では，**図 5** に示すように部屋のあらゆる場所に数十個のユビキタスチップが配置され，さまざまなセンサや出力機器と接続され，複数のアプリケーションが実現されている状況を想定する．ユビキタスチップがセンサからデータを収集し，メッセージをマルチホップの通信で送信し，受信したメッセージをもとに接続しているデバイスを制御し，LED やブザーで状況を知らせたり，アクチュエータを用いてドアを制御したりする．提案するトポロジ発見手法は以下のような場合に使用することを想定している．

- ユーザが PDA を部屋に持ち込み，部屋のアプリケーションをカスタマイズする．
- ユーザが PDA だけでなくユビキタスチップも持って部屋に入ったときには，ユーザの持っているユビキタスチップに接続されたブザーや振動モータが部屋で稼動中のアプリケーションと連携して動作するようにルールを書き換える．ユーザの持ち込んだ PDA がネットワークトポロジを発見し，ルールの書き換えを行う．
- ユビキタスチップの埋め込まれた新しい家具が部屋に持ち込まれたときには，業者などが部屋のユビキタスチップの動作を変更する．
- ネットワークのトポロジが頻繁に変化するときは，部屋にトポロジを管理し，維持するためのサーバを設置する．

このような状況で，以下のような 3 つの例を考える．

場所に応じたトポロジ構築

建物に埋め込まれた多数のユビキタスチップとセンサが連携して建物制御システムを実現している状況を想定する．ここでは，各ユビキタスチップに取り付けられたセンサの種類をシステムが把握しているとする．

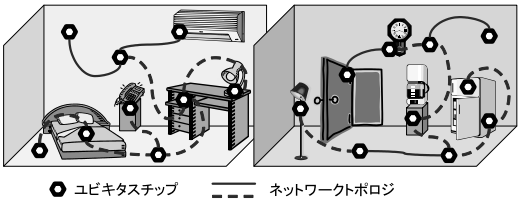


図 5 想定環境のイメージ図
Fig. 5 Image of our assumed environment.

このような状況で，周囲に重要なセンサがある部分ではすべてのネットワークの接続関係を把握することで確実に重要なセンサのデータを利用できるようにし，特に重要なセンサがない部分では最低限のネットワークの接続関係の把握にとどめることで，メッセージ量を削減でき，電力の消費を抑える．たとえば人の出入りの管理が重要な建物ではドア付近のセンサが重要となり，温度調節が必要な建物では温度センサが重要となる．

信頼性に基づくトポロジ発見

ネットワークの信頼性が場所によって異なる環境を想定すると，メッセージが受け取れないなどのエラーがよく起こる部分では，メッセージの送信を繰り返すなどして確実にメッセージが届くようにする必要がある．しかし，信頼性の高い部分ではメッセージの送信を繰り返す必要はない．トポロジの発見においても，このようにネットワークの信頼性によって動的にメッセージの送信方法を変更すると効率良くトポロジの発見が行える．特に有線のネットワークと無線のネットワークが混在する状況ではこのような機能が重要となる．

センサに基づくトポロジ発見

壁面やドアなどにユビキタスチップが埋め込まれ，ユビキタスチップが埋め込まれた物が出し入れされるような倉庫を想定する．このような状況では，現在収納されている物の種類によって，特に詳細に状況を確認するべき部分が変わってくる．たとえば，あるときは温度が高い部分の状況を把握する必要があるが，別のものが収納されているときには日の当たる部分の状況を把握する必要があるといった場合が想定される．このような環境では，詳細な状況を知る必要がある部分ではすべてのユビキタスチップのトポロジを発見して詳細にデータを収集し，そうではない部分では最低限のユビキタスチップのみでトポロジ構築することで，サーバの計算機のリソースやユビキタスチップのバッテリを節約できる．

このようにユビキタスコンピューティング環境にお

けるアプリケーションを実現する際には，柔軟なネットワークトポロジの発見・構築手法が必要となる．具体的には以下のような特徴が求められる．

(1) **動的な振舞いの変更：**ユビキタスコンピューティング環境においてネットワークトポロジを発見する際には，周囲の状況やアプリケーションの要求によって，必要とされる接続関係が異なる．そのため，トポロジの発見機能には，周囲の状況の変化やアプリケーションからの要求をトリガとして，動的にトポロジの発見手法を切り替える機能が必要となる．これまでに提案されている手法[2),3),7),8)]では，トポロジを発見している途中にトポロジ発見手法自体を変更することは困難であった．

(2) **多数の手法の混在：**上記のような状況を考慮すると，単にトポロジの発見を効率良く行うための手法だけでなく，センサの値などと関連させた無数の手法が存在する可能性がある．たとえば，トポロジを発見している様子をユーザが LED で確認できる手法や，マイクをユビキタスチップに接続し，音が聞こえる範囲内のみでトポロジを発見するなどといった特殊なトポロジ発見手法も存在しうる．このような多数の手法に対しては，パラメータの変更のみで対処することは不可能であり，これらのすべての要求に対処できるだけの手法を前もってファームウェアに実装しておくことも難しい．さらに，将来新しいトポロジ発見手法が開発されたときにもこれを導入できる必要がある．

(3) **貧弱なデバイスでも実装可能：**想定環境では多様なトポロジ発見手法が必要となるが，ユビキタスチップのような計算能力の乏しいデバイスでそれらを実現するためには，高い計算能力を必要とせず，さらにデバイスのその他の機能を妨げないことが求められる．また，ルーティングテーブルを保持するほどのメモリを持たないデバイス上でも動作可能であり，複数の通信手段が混在する状況にも対処できる必要がある．

本研究では，ECA ルールを用いてこのような要件を満たすトポロジ発見手法を実現する．ルールでトポロジ発見手法を実現すると，ルールの有効化/無効化/追加/消去を行うことで，動的にトポロジ発見手法を変更することが可能となる．また，パラメータの変更のみでは実現できない大幅な手法の変更についても，ファームウェアを書き換えることなく，ルールを差し替えるのみで対処できる．さらにルールによってトポロジ発見手法を記述すると，機能のモジュール化が容易になるため，このような手法の変更もスムーズに行える．さらに本研究では，ユビキタスチップがすでに持つ機能のみを利用してトポロジ発見手法を実現しているため，ユビキタスチップの既存の機能と並存できる．

4. ユビキタスチップのためのトポロジ発見

4.1 想 定 環 境

ユビキタスチップはすでにルール処理エンジンを持っており，ユビキタスチップの通信に関する ECA ルールを用いてトポロジ発見を実現する．実際のアプリケーションでは，サーバはユビキタスチップのルールを書き換え，アプリケーションが動作するモードからトポロジの変化をチェックするモードにユビキタスチップの動作を定期的に切り替える．このようなモードの変更は，ENABLE_ECA コマンドと DISABLE_ECA コマンドを用いてルールの有効化/無効化を行うことで実現できる．

図 6 にトポロジ解析の例を示す．ユビキタスチップが図 6 (a) に示すように配置されているとき，サーバは図 6 (b) に示すようなツリー構造でユビキタスチップ間のネットワークトポロジを管理する．ツリーのそれぞれのノードは，図 6 (c) に示すような，自身の ID と親，子の ID の一覧とその他の通信可能なユビキタスチップの ID の一覧を持つ．

ルールを用いたトポロジ発見手法は単純にトポロジを発見する手法だけでなくセンサの値と連携した手法などさまざまな手法が考えられるが，本稿では例として基本的な手法と，トポロジの維持をルールで行う手法，ネットワークの信頼性によってルールを切り替えながらトポロジ発見を行う手法を取り上げ，詳細を述べる．

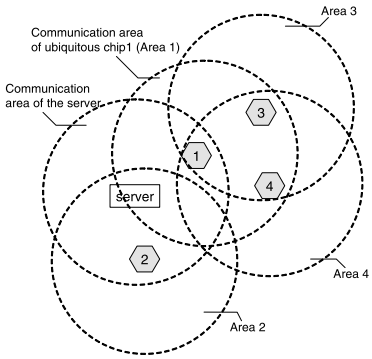
4.2 基本的な手法

トポロジを発見する手法としては，メッセージをマルチホップで送信した際の送信元 ID を用いる手法（直接法）と，メッセージを用いて固有の ID を送信する手法（間接法）の 2 通りを実現した．直接法では少ないメッセージ数でトポロジを発見でき，間接法ではより遠いホップのユビキタスチップを発見できる．実現した手法では，サーバが環境内にあるユビキタスチップに ECA ルールを書き込み，ユビキタスチップに問合せを送信し，その返信からネットワークトポロジを解析する．以下で両手法の詳細を述べる．

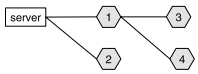
4.2.1 直 接 法

直接法でネットワークトポロジの発見に使用する ECA ルールを**表 5** に示す．サーバは，このルールを用いて以下のような手順でメッシュ構造のネットワークトポロジを発見する．

ここで，すべてのユビキタスチップには WAIT_UCID



(a) 実空間でのユビキタスチップの配置
(a) Allocation of ubiquitous chips in the real world



(b) ツリー構造を用いたユビキタスチップ間の
ネットワークトポロジの表現
(b) Tree topology of network among ubiquitous chips

(c) ツリー構造の各ノードが持つ情報
(c) Content of each node

ID	親	子	その他
server	—	1, 2	—
1	server	3, 4	—
2	server	—	—
3	1	—	4
4	1	—	3

図 6 トポロジ解析の例
Fig. 6 An example of the topology discovery.

表 5 直接法で用いる ECA ルールの一覧
Table 5 ECA rules for direct method.

WAIT_UCID
E:
C: フラグが ON
A: ID に比例した時間のタイマを設定し，フラグを OFF
REPLY_MESSAGE
E: REQUEST_UCID メッセージの受信
C:
A: フラグを ON
E: タイマの発火
C:
A: REPLY_UCID メッセージの送信
REQUEST_UCID：ユビキタスチップの ID 要求
REPLY_UCID：サーバへの応答

ルールと REPLY_MESSAGE ルールの 1 つ目のルールが書き込まれているとする．

Step1 サーバから直接通信できるユビキタスチップに REPLY_MESSAGE ルールの 2 つ目のルールを格納するための ADD_ECA コマンドを送信する．

Step2 ルールを書き込んだユビキタスチップへ REQUEST_UCID メッセージを送信する．

Step3 REQUEST_UCID メッセージを受信したユビキタスチップからの返信である REPLY_UCID メッセージを受信する．

Step4 受信したメッセージの送信元 ID を記録し，トポロジのツリーへ追加する．

以上の操作により 1 ホップ目に存在するユビキタスチップを発見できる．以降，同様に 2 ホップ目以降の発見を行う．

Step5 前のホップで発見したユビキタスチップの中で最小の ID を持つユビキタスチップに注目する．

Step6 現在注目しているユビキタスチップを通って 2 ホップで通信を行えるユビキタスチップへ REPLY_MESSAGE ルールの 2 つ目のルールを追加する．これは，サーバが送信するコマンドのマルチホップヘッダの 1 ホップ目には注目しているユビキタスチップの ID を，2 ホップ目には放送 ID を設定することで実現する．

Step7 Step6 と同じ宛先に REQUEST_UCID メッセージを送信する．

Step8 ユビキタスチップからの REPLY_UCID メッセージを受信する．

Step9 メッセージの送信元 ID をトポロジのツリーへ追加する．ID がすでに受信したことのある ID であれば，注目しているユビキタスチップの通信可能ユビキタスチップとして記録し，そうでなければ，注目しているユビキタスチップの子として記録する．

Step10 REPLY_UCID メッセージの重複受信を避けるため，現在注目しているユビキタスチップの親と親が発見される以前に発見されたユビキタスチップから REPLY_MESSAGE ルールを無効化する．

Step11 前のホップで発見されたユビキタスチップのうちまだ注目されていないものがあれば，次にそれに注目し，Step6 へ戻る．すべてのユビキタスチップがすでに注目されていれば，次のステップへ進む．

Step12 前のホップで最初に注目したユビキタスチップの最小の ID を持つユビキタスチップを次に注目するユビキタスチップに選択し，2 ホップ目と同様にして 3 ホップ目以降の操作を行う．

Step13 ホップ数がコマンド長の制限を超えるか，すべてのユビキタスチップを発見し終えれば，ネットワークトポロジの発見を終了する．

ツリー構造のみを発見する場合も同じルールを使用する．メッシュ構造の場合と異なるのは，ルールの無効化と削除のタイミングのみであり，注目しているユビキタスチップを変更するときに，それまでに発見したすべてのユビキタスチップの WAIT_UCID ルール

表 6 間接法で使用する ECA ルールの一覧
Table 6 ECA rules for indirect method.

SEND_MESSAGES
E: C: フラグが ON A: ID に比例した時間待った後に, REPLY_UCID_M メッセージを送信し, フラグを OFF
REPLY_MESSAGES
E: REQUEST_UCID_M メッセージの受信 C: A: フラグを ON
RELAY_MESSAGES
E: REPLY_UCID_M メッセージの受信 C: A: REPLY_UCID_M メッセージの送信
REQUEST_UCID_M：ユビキタスチップの ID 要求 REPLY_UCID_M：ユビキタスチップの ID を表す 複数のメッセージの返信

と，REPLY_MESSAGE ルールを無効化する．

4.2.2 間 接 法

間接法の ECA ルールを**表 6** に示す．間接法では，ユビキタスチップが複数のメッセージを送信して自身の ID をサーバへ知らせる．たとえば，57H (01010111) というユビキタスチップ ID をサーバへ通知する場合，ID を 01，01，01，11 という 4 つのメッセージに分割して送信する．送信元 ID を利用してトポロジを発見する直接法では，ユビキタスチップからサーバへのすべてのホップを指定した REPLY_MESSAGE ルールを書き込むが，パケット長の制限があるため 4 ホップ以上離れた場所にあるユビキタスチップには REPLY_MESSAGE ルールを書き込めない．間接法では，ID をメッセージで表現し，途中のユビキタスチップでメッセージを転送するため，長いホップを指定したルールを書き込む必要がなく，サーバはより遠いホップまでユビキタスチップを発見できる．

基本的な操作は直接法の場合と同様である．まず，メッシュ構造を発見する場合の操作について述べる．ユビキタスチップには前もって SEND_MESSAGES ルールが書き込まれている．直接法での REPLY_MESSAGE ルールが追加される部分の操作は行わない．また直接法と異なるのは，ユビキタスチップに注目された際に RELAY_MESSAGE ルールが書き込まれる点である．このルールは REPLY_UCID_M メッセージを中継するために使用される．前もって SEND_MESSAGES ルールが書き込まれた時点では周囲のユビキタスチップの ID が分からないため，このルールでのメッセージの送信はマルチホップでない SEND_MESSAGE アクションを用いて記述される．このメッセージを注目されているユビキタスチップが受け取ると，メッセージは RELAY_MESSAGE ルールによってサーバまで転送される．また，この RELAY_MESSAGE ルールは，メッセージの重複受信を避けるため，次のユビキタスチップが注目されたときに無効化され，自分自身の子が最初に注目されたときに再び有効化される．

ツリー構造のみを発見する場合も同じルールを使用する．メッシュ構造の場合と異なるのは，ルールを無効化するタイミングのみであり，注目しているユビキタスチップを変更するときに，それまでに発見したすべてのユビキタスチップの SEND_MESSAGES ルールを無効化する．

4.2.3 両手法の比較

両手法を比較すると，直接法では ID をサーバに知らせる際に必要なメッセージは 1 であるのに対して，間接法では 4 つのメッセージが必要となるため，直接法の方が少ないメッセージ量でトポロジを発見できる．トポロジを発見可能な範囲で比較すると，直接法で発見できる最大のホップ数は 3 であり，間接法では 7 ホップとなる．

4.3 トポロジの維持

ユビキタスチップの埋め込まれた物の出入りが多い場合など，ネットワークトポロジの維持が必要な場合には，トポロジの維持を必要とする部分のユビキタスチップにそのためのルールを書き込むことで，ネットワークトポロジ維持を実現する．本節では新しいユビキタスチップの検出と，ユビキタスチップの退出に対するネットワークトポロジ維持について述べる．

表 7 は新しいユビキタスチップを検出するためのルールである．サーバはネットワークトポロジの発見が終了した時点で NOTIFY_DISCOVER ルールをネットワーク上のユビキタスチップに書き込む．新しくネットワークに参加しようとするユビキタスチップは ADVERTISE_MESSAGE ルールによって NEW_UC メッセージを周囲のユビキタスチップへ繰り返し送信する．このメッセージを NOTIFY_DISCOVER ルールが書き込まれたユビキタスチップが受信すると，新しいユビキタスチップへ STOP_NEW_UC メッセージを送信し，同時に自身の ID をサーバに知らせる．新しいユビキタスチップでは STOP_ADVERTISE ルールが発火し NEW_UC メッセージの送信を停止する．サーバは ID の通知を受けると，最初に受け取った ID のユビキタスチップに注目し，間接法でトポロジ発見を行った場合と同様にして，新しいユビキタスチップの ID を調べる．ただし，このときメッセージの重複を防ぐため，ID を通知したが注目されなかったユビ

表 7 新たなユビキタスチップの発見に用いるルールの一覧
Table 7 ECA rules for discovery of a new ubiquitous chips.

ADVERTISE_MESSAGE（新たなユビキタスチップ）
E: C: タイマ用フラグが OFF A: タイマ（リピート）を設定しタイマ用フラグを ON
E: タイマの発火 C: A: NEW_UC メッセージの送信
STOP_ADVERTISE（新たなユビキタスチップ）
E: STOP_NEW_UC メッセージの受信 C: A: タイマを停止
NOTIFY_DISCOVER（その他のユビキタスチップに追加）
E: NEW_UC メッセージの受信 C: A: タイマ 1 とタイマ 2 を設定
E: タイマ 1 の発火 C: A: STOP_NEW_UC メッセージの送信
E: タイマ 2 の発火 C: A: WAIT_UCID ルールと SEND_MESSAGES ルールを発火させるためのフラグを ON
NEW_UC：新たなユビキタスチップが周囲に存在を知らせるメッセージ STOP_NEWUC：NEW_UC を止めるためのメッセージ

表 8 ユビキタスチップの退出を発見するためのルール一覧
Table 8 ECA rules for discovery of disappearing ubiquitous chips.

CHECK_NEXT（親のユビキタスチップに追加）
E: C: フラグ 1 が OFF A: タイマ 3（リピート）を設定し，フラグ 1 を ON，フラグ 2 を OFF
E: タイマ 3 の発火 C: フラグ 2 が OFF A: CHECK_ALIVE メッセージを子のユビキタスチップへ送信しフラグ 2 を ON
E: REPLY_ALIVE メッセージの受信 C: A: フラグ 2 を OFF
REPLY_CHECK（子のユビキタスチップに追加）
E: CHECK_ALIVE メッセージの受信 C: A: REPLY_ALIVE メッセージの送信
NOTIFY_DISAPPEAR（親のユビキタスチップに追加）
E: タイマ 3 の発火 C: フラグ 2 が ON A: WAIT_UCID ルールと SEND_MESSAGES ルールを発火させるためのフラグを ON
CHECK_ALIVE：子のユビキタスチップへ繰り返し送信されるメッセージ REPLY_ALIVE：CHECK_ALIVE メッセージに対する応答

キタスチップの RELAY_MESSAGE ルールを一時的に無効にする．

表 8 は 2 つのユビキタスチップ間の通信を定期的に確認することで，ユビキタスチップのネットワークからの退出を検出するためのルールである．

2 つのユビキタスチップのうちサーバへのホップ数が近い方を親，遠い方を子とする．このルール例では子の退出を検出できる．親は CHECK_NEXT ルールによって CHECK_ALIVE メッセージを子に向けて定期的に送信する．子ではこのメッセージを受信すると REPLY_CHECK ルールが発火し REPLY_ALIVE メッセージを返信する．親が一定時間以内に返信を受信できないまま，タイムアウト用のタイマが発火すると NOTIFY_DISAPPEAR ルールが発火し，サーバに自身の ID を通知することで子の退出を通知し，サーバはトポロジを修正する．

4.4 通信エラーを考慮した手法

本節では，ネットワークの信頼性が場所によって異なるような環境を想定する．通信エラーやパケットのロストが発生するような状況では，メッセージを繰り返し送信することでより正確なトポロジの発見が可能になると考えられるが，エラーがほとんど発生しないような状況ではこのような繰返しは電源やネットワーク帯域を無駄に消費する．

本節では間接法を拡張することでこのような状況に対応したネットワークトポロジ発見手法について述べる．ここでは，ネットワークの信頼性を以下のように 3 つに分類する．

- **ケース 1**（エラーがほとんどない場合）：間接法を用いる．
- **ケース 2**（エラーが少しある場合）：新たに発見されたユビキタスチップは REPLY_UCID_M メッセージの返信を複数回繰り返す．変更するルールを**表 9** (a) に示す．繰り返す回数はネットワークの状況によって変更するものとする．
- **ケース 3**（エラーが多い場合）：新たに発見されたユビキタスチップは REPLY_UCID_M メッセージの送信を，注目されているユビキタスチップから返信のメッセージを受け取るまで繰り返す．表 9 (b) にルールを示す．

ネットワークの信頼度は表 9 (c) に示すルールを用いて予測する．CHECK_NETWORK ルールを前もってすべてのユビキタスチップに書き込み，新たなユビキタスチップに注目するときに CHECK_NETWORK メッセージをこのユビキタスチップと直接通信可能な範囲のユビキタスチップに送信する．サーバは返信の REPLY_CHECK メッセージを受信した個数から信頼

表 9 通信エラーを考慮したトポロジ発見のルール一覧
Table 9 ECA rules for the dynamic method.

(a) ケース 2 のルール
(a) ECA rules in case 2

SEND_MESSAGES（間接法から書き換え）
E:
C: フラグが ON
A: ID に比例した時間待った後，フラグを OFF に設定し，REPLY_UCID_M を 1 つずつ数回繰り返し送信

(b) ケース 3 のルール
(b) ECA rules in case 3

SEND_MESSAGES（間接法から書き換え）
E:
C: フラグが ON
A: ID に比例した時間待った後，フラグを OFF に設定し，REPLY_UCID_M メッセージを 1 つずつ送信し続ける
E: STOP_REPEAT メッセージの受信
C:
A: 次の REPLY_UCID_M メッセージを送信するようにフラグを変更
STOP_MESSAGES（注目されているユビキタスチップに追加）
E: REPLY_UCID_M メッセージの受信
C:
A: STOP_REPEAT メッセージの送信

(c) ネットワークの信頼性を求めるためのルール
(c) ECA rules to detect network reliability

CHECK_NETWORK
E: CHECK_NETWORK メッセージの受信
C:
A: REPLY_CHECK メッセージを数回送信
CHECK_NETWORK：新たなユビキタスチップへメッセージの送信を依頼するメッセージ REPLY_CHECK：CHECK_NETWORK メッセージに対する応答

度を決定する．

このようにルールを動的に選択することで，効率良くネットワークトポロジを発見できる．さらに，アプリケーションの中でもこのネットワークの信頼性の情報を利用できる．この手法もすべてルールの変更のみで実現しており，新しい手法を導入するときもファームウェアを変更する必要がない．

5. 実　　装

5.1 シミュレータ

提案する 2 つの手法の動作をシミュレータを用いて確認した．**図 7** にシミュレータのスクリーンショットを示す．左側の領域にユビキタスチップを配置し，解析したネットワークトポロジを右側の領域にツリー構造で表示している．シミュレータ上では，ユビキタスチップの無線通信範囲は固定とした．

表 10 と**表 11** に実際のルールを示す．このルール

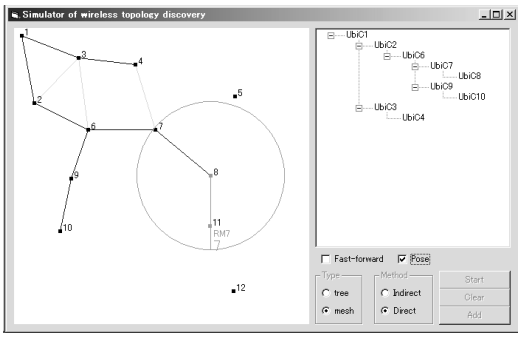


図 7 トポロジ発見のシミュレータ
Fig. 7 Simulator of topology discovery.

表 10 直接法の実装に用いたルールの一覧
Table 10 Rule set for direct method.

WAIT_UCID	
E:	
C: S0 = 1	
A: S0 = 0, T (ID × 100 ms)	
REPLY_MESSAGE（2 個のルール）	
E: RM(7)	E: Timer
C: S0 = 0	C:
A: S0 = 1	A: SM_M(6)

T：ID に比例した時間待つためのタイマ
RM：RECEIVE_MESSAGE イベント
SM_M：SEND_MESSAGE（マルチホップモード）アクション（メッセージをサーバまで送信するための経路が記述されている）
Message 7：REQUEST_UCID メッセージ
Message 6：REPLY_MESSAGE メッセージ
S0 (State 0)：タイマを設定するためのフラグ

を用いて**図 8** に示すトポロジの発見を行った結果を**表 12** に示す．結果より，直接法は間接法に比べて約半分のメッセージでトポロジを発見できるが，4 ホップ以上のユビキタスチップを発見できない．3 ホップまでは直接法を用い，それ以降は間接法を用いる複合的な手法はメッセージ量も間接法に比べて少なく，すべてのユビキタスチップを発見できている．

5.2 実機での動作確認

さらにシミュレータだけでなく，実際にユビキタスチップを用いて直接法の動作の確認を行った．**図 9** に動作確認に利用した 2 つのトポロジの例と動作確認の様子を示す．動作確認では，微弱無線通信モジュールを用い，電波の反射による影響を避けるため，WAIT_UCID ルール中のタイマの長さを表 10 の 2 倍の長さに設定した．

図 9 (a) に示すトポロジでは最初のホップに ID6 と 8 の 2 つのユビキタスチップがある．2 つのユビキタスチップへ向けて同時に REQUEST_UCID メッセージを送信しているが WAIT_UCID ルールのタイマの

表 11 間接法の実装に用いたルールの一覧
Table 11 Rule set for indirect method.

REPLY_MESSAGES	RELAY_MESSAGES（4 個のルール）			
E: RM(5)	E: RM(0)	E: RM(1)	E: RM(2)	E: RM(3)
C: S1 = 0	C:	C:	C:	C:
A: S1 = 1	A: SM_M(0)	A: SM_M(1)	A: SM_M(2)	A: SM_M(3)

SEND_MESSAGES（5 個のルール）		
E:	E: Timer	E: Timer
C: S1 = 1	C: S3 = 1, S4 = 1	C: S3 = 0, S4 = 1
A: S1 = 0, T (ID × 500 ms)	A: S3 = 0, S4 = 0, SM(D)	A: S3 = 1, S4 = 1, SM(C), T (100 ms)
E: Timer	E: Timer	
C: S3 = 1, S4 = 0	C: S3 = 0, S4 = 0	
A: S3 = 0, S4 = 1, SM(B), T (100 ms)	A: S3 = 1, S4 = 0, SM(A), T (100 ms)	

T：ID に比例した時間待つためのタイマ
RM：RECEIVE_MESSAGE イベント
SM：SEND_MESSAGE（シングルホップモード）アクション
SM_M：SEND_MESSAGE（マルチホップモード）アクション（間接法では，サーバか次に中継するユビキタスチップまでの経路が記述されている）
Message 0 – 3：REPLY_UCID_M メッセージ
Message A – D：ユビキタスチップの ID を表す 4 つのメッセージ（メッセージ ID0 から 3 の 4 種類のメッセージで 1 バイトのユビキタスチップ ID を表す）
Message 5：REQUEST_UCID_M メッセージ
S1 (State 1)：タイマを設定するためのフラグ
S3, S4 (State 3, State 4)：メッセージ A から D を順に送信するための変数

表 12 トポロジ発見の結果
Table 12 Result of topology discovery.

手法	メッセージ個数	メッセージ量（バイト）	未発見個数	未発見経路
直接法	93	302	8	22
間接法	647	2,849	0	0
間接法（3 ホップ以内のみ）	181	837	8	22
複合手法	559	2,105	0	0

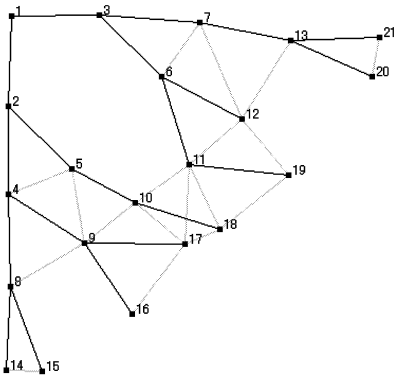
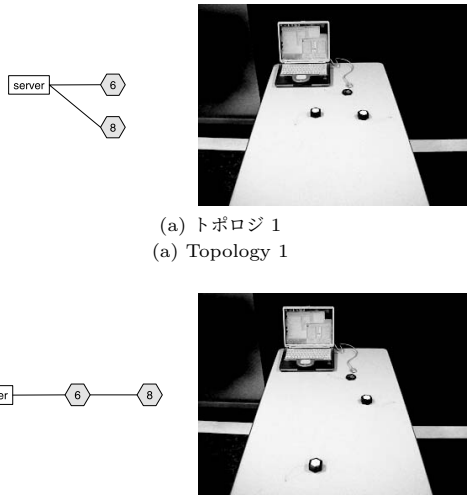


図 8 トポロジ発見のシミュレーションで用いたトポロジ（黒の実線がトポロジのツリー，灰色の実線がツリー以外のトポロジを意味する．四角が各ユビキタスチップを意味し，数字はユビキタスチップの ID を意味する．また，ID1 のユビキタスチップをサーバとする）
Fig. 8 An example of topology for simulation.



(a) トポロジ 1
(a) Topology 1

(b) トポロジ 2
(b) Topology 2

図 9 動作確認に用いたトポロジと動作確認の様子
Fig. 9 Topologies for the verification and snapshots.

時間が異なり，1 つずつ応答が返ってくるため，サーバは 2 つのユビキタスチップを発見できた．

図 9 (b) に示すトポロジでは，最初のホップに ID6 のユビキタスチップが，次のホップに ID8 のユビキタスチップが設置されている．ID6 のユビキタスチップを発見した後，このユビキタスチップに注目し，ID6 のユビキタスチップを介したマルチホップ通信を行う

ことで ID8 のユビキタスチップを発見できた．

6. 考　　察

6.1 ルールに基づくトポロジ発見の利点

提案手法の機能はすべてルールを用いて実現されているため，サーバに近い部分では直接法を用い，この手法では発見できない距離では間接法を用いるという複合的な手法も実現できる．また，提案手法ではルールを消去・無効化するタイミングを調節することで，ツリー構造のみを発見するか，メッシュ構造を発見するかを変更できる．既存の手法はこのような柔軟性に欠けているが，柔軟なネットワークトポロジの解析手法はユビキタスコンピューティング環境においてアプリケーションを構築するために必要となる．また，新たな手法を実装する際には，ルールを書き換えるだけでよくファームウェアの書き換えを必要としない．

一方，ファームウェアでトポロジの解析手法を実現するような既存の方法では，その手法に特化した最適な方法で実装を行える．このような手法と比較すると，提案手法はルールを介して解析を行うため，解析の速度が遅くなったり，メッセージ量が増加したりする可能性がある．

6.2 新たな手法の開発

一般的にルールを用いて複雑なシステムを実現しようとすると，逐次的なプログラミング言語を用いる場合と比較して，たいへんな労力を必要とする場合が多い．しかし，トポロジ発見に利用しているルールは，ユビキタスチップがそれほど高機能なデバイスではないため，比較的シンプルなルールの組合せで動作することが予想され，システムの開発にもそれほど労力をとられないと予想している．さらに，新たなネットワークトポロジの発見手法を開発する際には，シミュレータを用いてデバッグを行うことを想定している．トポロジ発見手法は，ユビキタスチップの動作を記述するための ECA ルールをそのまま用いて記述するため，これまでに実現したシミュレータを用いて動作を確認することが可能である[10]．さらに，メッセージ量の確認といった機能をトポロジ発見のシミュレータで実現しているが，このような機能も将来的にはユビキタスチップのシミュレータに統合する予定である．

6.3 関 連 研 究

MOTE は無線センサネットワークのためのプラットフォームである[6]．MOTE では，ネットワークトポロジを自動的に発見でき，センサ情報の収集が可能である．しかし，ネットワークトポロジを発見する方法を動的に変更することは考慮されていない，ユビキタスチップを用いると，より柔軟なアプリケーションの構築が可能となる．

Smart-Its はさまざまなものに埋め込むための小型のコンピュータである[1]．Smart-Its においても無線通信はすでに実現されているが，本稿で提案するような柔軟なネットワークトポロジの発見手法は提案されていない．また，本稿で提案した手法を Smart-Its デバイスの上に実装することも可能であると考えている．

ネットワークトポロジを解析する手法としては，近年，アドホックネットワークの分野で数多くの方式が提案されている．代表的なトポロジ解析，ルーティング手法としては，DSDV[7] や，OLSR[2]，AODV[8]，DSR[3] などがあるが，いずれもルーティングを維持するためのメッセージをやりとりするための処理能力や，ルーティング情報を格納するための記憶容量がモバイル端末に必要となる．このため，これら既存の手法はユビキタスチップには向かない．さらに，これらの手法は静的に用いられるものであり，トポロジを解析する手法自体を柔軟に変更することは想定されていない．

7. ま と め

本稿では，ユビキタスチップのためのルールに基づいたネットワークトポロジの発見手法を提案した．提案する手法では，ECA ルールを書き換えることで動的に解析手法を切り替えることが可能であり，ユビキタスコンピューティング環境における柔軟なアプリケーションの構築が可能となる．さらに本稿では，シミュレータおよびユビキタスチップ実機を用いて，提案手法の動作確認を行った．

今後の課題としては，提案したネットワークトポロジの情報を利用してアプリケーションの動作を動的に変更する枠組みやシミュレータなど開発環境の整備があげられる．

謝辞 本研究の一部は，文部科学省 21 世紀 COE プログラム「ネットワーク共生環境を築く情報技術の創出」，および基盤研究（A）（17200006）の研究助成によるものである．ここに記して謝意を表す．

参 考 文 献

1) Beigl, M. and Gellersen, H.: Smart-Its: An Embedded Platform for Smart Objects, *Smart Objects Conference* (*sOc*) *2003* (2003).
2) Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A. and Viennot, L.: Optimized Link State Routing Protocol for Ad Hoc Networks, *Proc. IEEE INMIC 2001* (2001).

3) Johnson, D.B. and Maltz, D.A.: Dynamic Source Routing in Ad Hoc Wireless Networks, *Proc. Mobile Computing 1996* (1996).
4) Kahn, J., Katz, R. and Pister, K.: Mobile Networking for Smart Dust, *Proc. ACM/IEEE International Conference on Mobile Computing and Networking* (*MobiCom 99*), pp.271–278 (1999).
5) Levis, P., Madden, S., Gay, D., Polastre, J., Szewczyk, R., Woo, A., Brewer, E. and Culler, D.: The Emergence of Networking Abstractions and Techniques in TinyOS, *Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation* (*NSDI 2004*) (2004).
6) MICA. http://www.xbow.com/products/Wireless_Sensor_Networks.htm
7) Perkins, C.E. and Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, *Proc. SIGCOMM 1994* (1994).
8) Perkins, C.E. and Royer, E.M.: Ad-hoc On-Demand Distance Vector Routing, *Proc. WMCSA 1999* (1999).
9) Sakamura, K.: TRON: Total Architecture, *Proc. Architecture Workshop in Japan '84*, pp.41–50 (1984).
10) 相良亮平，義久智樹，岸野泰恵，寺田　努，塚本昌彦，祐成光樹，田口大悟，西尾章治郎：イベント駆動型小型デバイス AhroD のためのアプリケーション開発環境，電子情報通信学会 2005 年総合大会，p.362 (Mar. 2005).
11) 祐成光樹，義久智樹，田口大悟，寺田　努，塚本昌彦，柏谷　篤，西尾章治郎：イベント駆動型小型デバイス AhroD のための Bluetooth 無線通信ユニットの開発，電子情報通信学会 2005 年総合大会，p.359 (Mar. 2005).
12) Terada, T., Tsukamoto, M., Hayakawa, K., Yoshihisa, T., Kishino, Y., Kashitani, A. and Nishio, S.: Ubiquitous Chip: a Rule-based I/O Control Device for Ubiquitous Computing, *Proc. Pervasive 2004*, pp.238–253 (2004).
13) Weiser, M.: The Computer for the 21st Century, *Scientific American*, Vol.265, No.3, pp.94–104 (1991).
14) Widom, J. and Ceri, S.: *Active Database Systems: Triggers and Rules for Advanced Database Processing*, Morgan Kaufmann Publishers (1996).

(平成 17 年 5 月 19 日受付)
(平成 17 年 11 月 1 日採録)

岸野　泰恵（学生会員）

2002 年大阪大学工学部電子情報エネルギー工学科情報システム工学科目卒業．2004 年同大学院情報科学研究科マルチメディア工学専攻博士前期課程修了．現在，同専攻博士後期課程に在籍．ユビキタスコンピューティング，ヒューマンインタフェースに興味を持つ．

寺田　　努（正会員）

1997 年大阪大学工学部情報システム工学科卒業．1999 年同大学院工学研究科博士前期課程修了．2000 年同大学院工学研究科博士後期課程退学．同年より大阪大学サイバーメディアセンター助手．2005 年より同講師．現在に至る．2002 年より同大学院情報科学研究科マルチメディア工学専攻助手，2005 年より同講師を併任．2004 年より特定非営利活動法人ウェアラブルコンピュータ研究開発機構理事，2005 年より同機構事務局長を兼務．工学博士．アクティブデータベース，ウェアラブルコンピューティング，ユビキタスコンピューティングの研究に従事．IEEE，電子情報通信学会，日本データベース学会の各会員．

塚本　昌彦（正会員）

1987 年京都大学工学部数理工学科卒業．1989 年同大学院工学研究科修士課程修了．同年シャープ（株）入社．1995 年大阪大学大学院工学研究科情報システム工学専攻講師，1996 年同専攻助教授，2002 年同大学院情報科学研究科マルチメディア工学専攻助教授，2004 年神戸大学電気電子工学科教授となり，現在に至る．2004 年より特定非営利活動法人ウェアラブルコンピュータ研究開発機構理事長を兼務．工学博士．ウェアラブルコンピューティングとユビキタスコンピューティングの研究に従事．ACM，IEEE 等，8 学会の会員．

義久 智樹（正会員）

2002 年大阪大学工学部電子情報エネルギー工学科卒業．2003 年同大学院情報科学研究科マルチメディア工学専攻博士前期課程修了．2005 年同専攻博士後期課程修了後，京都大学学術情報メディアセンター助手となり，現在に至る．博士（情報科学）．ユビキタスコンピューティング，ウェアラブルコンピューティングに興味を持つ．電子情報通信学会，日本データベース学会各会員．

早川 啓介（正会員）

1996 年筑波大学大学院理工学研究科修士課程修了．同年日本電気株式会社に入社．2004 年 NEC エレクトロニクス（株）へ移籍，現在に至る．ユビキタスコンピューティングに興味を持ち，主に RFID システムとセンサネットの研究開発に従事．ヒューマンインタフェース学会，日本バーチャルリアリティ学会各会員．

柏谷 篤

1989 年 NEC 入社．中央研究所にて画像入力端末の研究開発に従事．1998 年より NEC インターネットシステム研究所でユビキタス端末/サービス基盤の研究開発に従事．現在，NEC インターネットシステム研究所ユビキタスシステム・テクノロジーグループ研究部長．

西尾章治郎（正会員）

1975 年京都大学工学部数理工学科卒業．1980 年同大学院工学研究科博士後期課程修了．工学博士．京都大学工学部助手，大阪大学基礎工学部および情報処理教育センター助教授，大阪大学大学院工学研究科情報システム工学専攻教授を経て，2002 年より同大学院情報科学研究科マルチメディア工学専攻教授となり，現在に至る．2000 年より大阪大学サイバーメディアセンター長，2003 年より大阪大学大学院情報科学研究科長を併任．この間，カナダ・ウォータールー大学，ビクトリア大学客員．データベース，マルチメディアシステムの研究に従事．現在，Data & Knowledge Engineering 等の論文誌編集委員．本会理事を歴任．電子情報通信学会フェローを含め，ACM，IEEE 等，8 学会の会員．