2004－UBI－5　(8)
2004／6／21

# イベント駆動型入出力制御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† 寺田 努† 塚本昌彦† 義久智樹† 早川敬介‡ 柏谷 篤∗ 西尾章治郎†

† 大阪大学大学院情報科学研究科
‡ NEC エレクトロニクス 第三システム事業本部
∗ NEC インターネットシステム研究所

**あらまし：** 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み合わ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 Detection Mechanism for Rule-based I/O Control Devices

Yasue Kishino†, Tsutomu Terada†, Masahiko Tsukamoto†, Tomoki Yoshihisa†,
Keisuke Hayakawa‡, Atsushi Kashitani*, and Shojiro Nishio†

† Graduate School of Information Science and Technology, Osaka University
‡ 3rd Systems Operations Unit, NEC Electronics
∗ Internet Systems Research Laboratories, NEC Corp.

**Abstract :** In this paper, we propose a new network topology detection mechanism four ubiquitous chips. The ubiquitous chip is a rule-based, event-driven I/O (input/output) control device we have proposed for constructing ubiquitous computing environments. Since this device archives flexibility by describing its behavior as a set of rules, we employ a rule-based approach to detect a network topology. In ubiquitous computing environments, there are various communication methods, and the optimum one is different according to application requirements. Therefore, our flexible detection mechanism works well in ubiquitous computing environments. Moreover, we have verified our algorithms by implementing them on a topology detection 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 [6, 8, 10]．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 [9]．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埋め込むことで，人々の日常生活をサポートするさまざまなサービスを実現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

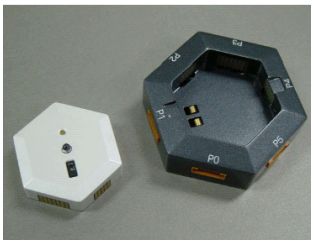


図 1: ユビキタスチップ

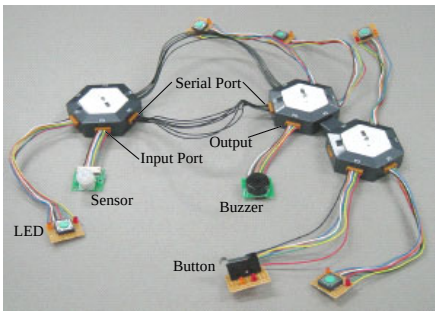



図 2: ユビキタスチップと周辺デバイスの接続例

ワークトポロジの発見にルールを用いることで，状況に応じた手法の切り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 章では提案手法の詳細を述べる．4 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．5 章で提案手法について考察を行い，6 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もつ入出力制御デバイスである．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．

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．

表 1: イベント

| 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

| 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 |

## 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，赤外線通信ユニット，微弱無線ユニット，bluetooth ユニットの試作を行っている．図 3 に試作した微弱無線ユニットを示す．このようにユビキタスチップでは，さまざまな特徴をもつ通信ユニットを組み合わせ，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

ユビキタスチップの基本的な通信機能としては SEND_MESSAGE アクションと SEND_DATA アクションがある．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を制御するために用い，後者はセンサの値など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．

### コマンド

表 3 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 ルールの追加や，ID を指定した ECA ルールの削除や有効化・無効化といった操作を行う．ユビキタスチップは SEND_COMMAND アクションでこれらのコマンドを他のユビキタスチップへ送信できるため，ユビキタスチップ同士が互いにルールを制御できる．

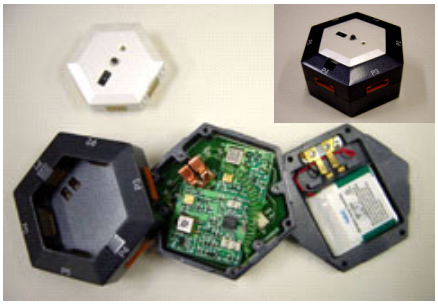


図 3: 微弱無線通信ユニット

表 3: SEND_COMMAND で送信するコマンド

| 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 |

### 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 を指定し，マルチホップで宛先のユビキタスチップまでデータを転送する．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．

マルチホップ通信を行なう場合には，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．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．

**Step2** ポインタがリストの最後をさしていない場合は，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，自身があて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．

## 3 トポロジ発見手法

本研究では，各部屋といった 1 つの空間に，数十のユビキタスチップとそれに接続したセンサや出力デバイスがさまざまな場所に埋め込まれている状況を想定している．ユビキタスチップは無線通信でセンサから収集したデータをやり取りし，そのデータをもとに出力デバイスを制御する．このようなユビキタスチップ間の通信を利用した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ネットワークトポロジを解析することが必要となる．

空間には，PC 程度の処理能力とメモリがあり，ユビキタスチップとメッセージやコマンドを送受信できる機能を備えたサーバが設置されていることを想定している．このサーバがユビキタスチップのネットワークトポロジを管理し，トポロジの変化を検出したときや，メッセージ量を調節する必要があるときに，自動的にルールを書き換え，通信経路を変更する．さらに，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解析する際には，以下のような理由からその手法自体に柔軟性が求められる．

- ユビキタスチップの電池を節約するためには，通信量を最小にする必要があるため，サーバはネットワークトポロジの必要な部分のみを解析できる必要がある．
- 必要とされるネットワーク構成が周囲の状況やアプリケーションによって異なる．たとえば，確実な通信が求められるアプリケーションでは，ユビキタスチップ間のすべての接続関係を発見し，何通りもの経路を確保する必要があるが，そうでないアプリケーションでは，すべての関係を発見せず，それぞれのユビキタスチップについて一通りの経路が確保できれば十分である．

さらに，ユビキタスコンピューティング環境においては有線や無線などさまざまな通信手段が混在することが予想され，提案手法はこのような状況にも対応できる必要がある．

このようなネットワークトポロジを解析する手法は，近年，アドホックネットワークの分野で数多く提案されている．アドホックネットワークは，無線通信機能をもつ複数の端末が一時的にネットワークを構築するネットワークであり，モバイル端末間のトポロジが無線通信

によって解析される．代表的なトポロジ解析，ルーティング手法としては，DSDV[2] や，OLSR[4]，AODV[3]，DSR[5] 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取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．

本研究では，ユビキタスチップ間のネットワークトポロジを解析するための新たな手法を提案する．提案する手法はルールで記述できるため，解析手法を柔軟に変更できる．さらに，ユビキタスチップへルールを書き込むだけで手法が実現できるため実装が容易である．本稿では，直接法と間接法の二種類の手法を提案する．さらに，それぞれの手法についてすべてのユビキタスチップ間の接続関係を発見するメッシュ構造と，サーバから各ユビキタスチップへの経路を一つずつ発見するツリー構造の発見方法を提案する．この二種類の構造の発見方法は，ルールを削除・無効化するタイミングを変更することで実現する．サーバは，ユビキタスチップのルールを書き換え，問い合わせメッセージを送信し，返答メッセージを受信することでネットワークトポロジを解析する．

図 4 にトポロジ解析の例を示す．ユビキタスチップが図 4(a) に示すように配置されているとき，サーバは図 4(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 4(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 の一覧をもつ．

以下で，二種類の手法の詳細を述べる．

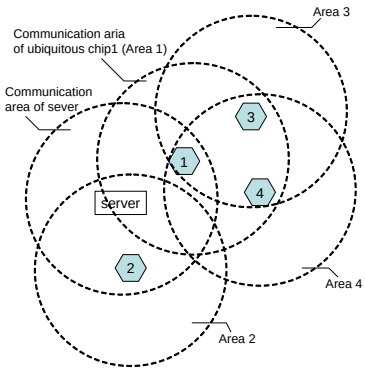
## 3.1 直接法

直接法でネットワークトポロジの発見に使用する ECA ルールの一覧を表 4 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の発見を行う．

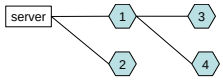
**Step1** すべてのユビキタスチップに前もって WAIT_UCID ルールを書き込む．

**Step2** サーバから直接通信できるユビキタスチップに REPLY_MESSAGE ルールを格納するための ADD_ECA コマンドを送信する．

**Step3** ルールを書き込んだユビキタスチップへ




(a): 実空間でのユビキタスチップの配置




(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

(c): ツリー構造の各ノードがもつ情報

| ID | 親 | 子 | その他 |
|---|---|---|---|
| server | - | 1, 2 | - |
| 1 | server | 3, 4 | - |
| 2 | server | - | - |
| 3 | 1 | - | 4 |
| 4 | 1 | - | 3 |

図 4: トポロジ解析の例

REQUEST_UCID メッセージを送信する．

**Step4** 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．

**Step5** 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．

以上の操作が終わった後，サーバは 2 ホップで通信が行えるユビキタスチップの発見の操作を開始する．2 ホップ目以降の操作は 1 ホップ目と同様である．

**Step6** 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 をもつユビキタスチップに注目する．

**Step7** 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 ルールを追加する．このとき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 を設定し，REPLY_MESSAGE ルール中の REPLY_UCID メッセージ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 ホップ目にはサーバの ID を設定する．

**Step8** Step7 と同じ ID が指定されたマルチホップヘッダを用いて REQUEST_UCID メッセージを送信する．

表 4: 直接法で用いる ECA ルールの一覧

| WAIT_UCID |
|---|
| E:<br>C: フラグが ON<br>A: ID に比例した時間のタイマを設定 |
| REPLY_MESSAGE |
| E: REQUEST_UCID メッセージの受信<br>C:<br>A: フラグを ON |
| E: タイマの発火<br>C:<br>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<br>REPLY_UCID：サーバへの応答 |

**Step9** ユビキタスチップからの REPLY_UCID メッセージを受信する．

**Step10** 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．

**Step11** REPLY_UCID メッセージの重複した受信を避けるため，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 ルールを消去する．

**Step12** 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，Step7 へ戻る．すべてのユビキタスチップをすでに注目していれば，次のステップへ進む．

**Step13** 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 をも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，同じように 3 ホップ目以降の操作を行う．

**Step14** ホップ数がコマンド長の制限を越えるか，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 ルールを無効化し，REPLY_MESSAGE ルールを削除する．

## 3.2 間接法

間接法でネットワークトポロジの発見に使用する ECA ルールの一覧を表 5 に示す．

表 5: 間接法で使用する ECA ルールの一覧

| SEND_MESSAGES |
|---|
| E:<br>C: フラグが ON<br>A: ID に比例した時間待った後に，<br>REPLY_UCID_M メッセージを送信 |
| REPLY_MESSAGES |
| E: REQUEST_UCID_M メッセージの受信<br>C:<br>A: フラグを ON |
| RELAY_MESSAGES |
| E: REPLY_UCID_M メッセージの受信<br>C:<br>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<br>REPLY_UCID_M：ユビキタスチップの ID をあらわす複数のメッセージの返信 |

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 ID のユビキタスチップが自身の ID をサーバ知らせるときには，ID を 01，01，01，11 という 4 つの ID のメッセージに分割して送信する．サーバは受信した複数のメッセージを再構築することでユビキタスチップの ID を知る．このため，サーバはマルチホップヘッダから送信元 ID を利用して ID を知る直接法に比べて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おり，直接法で REPLY_MESSAGE ルールが追加される部分で間接法では REPLY_MESSAGES ルールが追加される．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

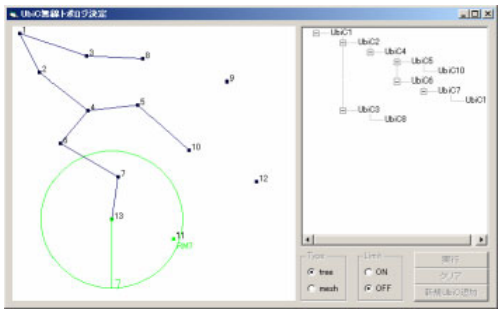


図 5: トポロジ発見のシミュレータ

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 実装

提案する二つの手法の動作をシミュレータを用いて確認した．図 5 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左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．

## 4.1 直接法

表 6 に直接法の実装に用いたルールを示す．初期状態では，ユビキタスチップには WAIT_UCID ルールのみが格納されており，REPLY_MESSAGE ルールは Step2 と Step7 で無線通信によって追加される．ユビキタスチップが ID7 のメッセージ（REQUEST_UCID）を受信したときに，State0 が 0 であると State0 に 1 を設定する．この状態の変化によって WAIT_UCID ルールが発火すると，タイマが設定され State0 は 0 に設定される．タイマが発火するとユビキタスチップは ID6 のメッセージ（REPLY_UCID）をサーバまでマルチホップ通信で送信する SEND_MESSAGE アクションを実行する．サーバがこのメッセージを受信すると，マルチホップヘッダから送信元 ID を取り出し，ネットワークトポロジに追加する．

## 4.2 間接法

間接法についても，直接法と同様にしてルールを作成し，動作を確認した．ルールの一覧を表 7 に示す．初期

表 6: 直接法の実装に用いたルールの一覧

| | |
|---|---|
| WAIT_UCID | |
| E: | |
| C: S0=1 | |
| A: S0=0, T(ID × 100ms) | |
| 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


(a): トポロジ 1　　(b): トポロジ 2

図 6: 直接法の動作確認に用いたトポロジ

状態では，ユビキタスチップには，SEND_MESSAGES ルールのみが格納されており，REPLY_MESSAGES ルールと RELAY_MESSAGE ルールはサーバから ADD_ECA コマンドを受信したときに追加される．ID5 のメッセージ（REPLY_UCID_M）を受信したときにすでに REPLY_MESSAGE ルールが追加されていれば，ユビキタスチップ内では REPLY_MESSAGE ルールによって SEND_MESSAGES ルールが発火し，ユビキタスチップの ID を表す 4 つのメッセージが順に送信される．また，これらのメッセージを中継するために受信したユビキタスチップでは，RELAY_MESSAGE ルールのいずれかが発火し，次のユビキタスチップかサーバへメッセージが転送される．

## 4.3 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 6 に動作確認に利用した二つのトポロジの例を示す．動作確認では，微弱無線通信モジュールを用い，電波の反射の影響を避けるため，WAIT_UCID ルール中のタイマの長さを表 6 の二倍の長さに設定した．

図 6(a) に示すトポロジでは最初のホップに ID6 と 8 の二つのユビキタスチップがある．二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 ルールのタイマの時間が

表 7: 間接法の実装に用いたルールの一覧

| REPLY_MESSAGES |
|---|
| E: RM(5) |
| C: S1=0 |
| A: S1=1 |

| RELAY_MESSAGES (4 rules) | | | |
|---|---|---|---|
| E: RM(0) | E: RM(1) | E: RM(2) | E: RM(3) |
| C: | C: | C: | C: |
| A: SM_M(0) | A: SM_M(1) | A: SM_M(2) | A: SM_M(3) |

| SEND_MESSAGES (5 rules) | | |
|---|---|---|
| E: | E: Timer | E: Timer |
| C: S1=1 | C: S3=1, S4=1 | C: S3=0, S4=1 |
| A: S1=0, T(ID × 500ms) | A: S3=0, S4=1, SM(D) | A: S3=1, S4=1, SM(C), T(100ms) |
| E: Timer | E: Timer | |
| C: S3=1, S4=0 | C: S3=0, S4=0 | |
| A: S3=0, S4=0, SM(B), T(100ms) | A: S3=1, S4=0, SM(A), T(100ms) | |

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，メッセージをサーバか，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

異なり，一つずつ応答が返ってくるため，サーバは二つのユビキタスチップを発見できた.

図 6(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.

# 5 考察

## 5.1 二つの手法の比較

メッセージ量の観点から二つの手法を比較すると，直接法は，1 つのメッセージでユビキタスチップの ID をサーバへ伝えるため，メッセージ量が少なくなる．間接法を用いた場合は 4 つのメッセージでサーバへ ID を伝えるため，直接法よりも通信量が多くなり，発見に必要な時間も 4 倍になる．通信遅延のない状況を想定し，14 個のユビキタスチップとサーバからなる 3 ホップまでの二分木のツリー構造を両手法で発見するのに必要な通信量と時間を試算したところ，直接法では 1822Byte と 56 秒，間接法では 2780Byte と 224 秒となった．この時間は，ユビキタスチップのタイマ設定の最小単位を変更することで高速化できる．また試算では，ユビキタスチップ ID の最大値は 39，2 つのタイマを用いると仮定した.

一方，間接法では長いホップのマルチホップ通信を行わずにメッセージを中継することでユビキタスチップの ID をサーバへ伝えられるため，直接法よりも柔軟で拡張性に優れる．特に，試作したユビキタスチップはコマンド長に 15 バイトの制限があるため，間接法を用いると 7 ホップ離れた場所に設置されたユビキタスチップを発見できるが，直接法では 5 ホップ以上離れているユビキタスチップを発見できない.

## 5.2 ルールによるトポロジ発見の利点

すべてルールを用いて二つの手法を実装しているため，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考えられる．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．このような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．既存の手法は柔軟性に欠けているが，提案手法では，ECA ルールを変更することで容易にこのような手法を実現できる．さらに，新たな手法が考案されたときには，ルールを書き換えることで容易に新たな手法に切り替えられ，ファームウェアを書き換える必要がない.

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なえる．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なうため，解

析の速度が遅くなったり，メッセージ量が増加する可能性がある．

### 5.3 動的なトポロジ変化への対応

本稿で提案した手法は，動的なトポロジ変化には対応していないが，ユビキタスコンピューティング環境においては，動的にネットワークトポロジが変化することが予想される．このような動的なトポロジの変化も，新たにユビキタスチップの追加や削除を検出するための ECA ルールを追加することで実現できる．また，実際のアプリケーションではサーバにアプリケーションモードとトポロジ維持モードの二種類のモードを設けることを想定している．サーバは定期的にトポロジ維持モードへ遷移し，トポロジの変化がないか調査する．このような遷移も ENABLE_ECA コマンドや DISABLE_ECA コマンドを用いることで実現できる．

また，すでに間接法の場合については，新しいユビキタスチップを発見するための ECA ルールを考案しており，シミュレータ上でその動作を確認した．新しいユビキタスチップはメッセージの送信を繰り返し，周囲のユビキタスチップがこのメッセージを受信するとサーバまでメッセージをルールで転送する．サーバがこのメッセージを受信すると 3.2 節で説明した方法と同様にして，新しいユビキタスチップの ID を問い合わせ，ネットワークトポロジに追加する

### 5.4 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 [7]. 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．さらに，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．

## 6 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，新規ユビキタスチップの追加や削除など，ネットワークトポロジの変化に対応した機能の実現が挙げられる．さらに，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を提案することも課題である．

## 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」によるものである．ここに記して謝意を表す．

## 参考文献

[1] M. Beigl and H. Gellersen: Smart-Its: An Embedded Platform for Smart Objects, Smart Objects Conference (sOc) 2003, 2003.

[2] C. E. Perkins and P. Bhagwat: Highly Dynamic Destination-Sequenced Distance-Vector Routing(DSDV) for Mobile Computers, in Proc. of SIGCOMM 1994, 1994.

[3] C. E. Perkins and E. M. Royer: Ad-hoc On-Demand Distance Vector Routing, in Proc. of WMCSA1999, 1999.

[4] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot: Optimized Link State Routing Protocol for Ad Hoc Networks, in Prod. of IEEE INMIC 2001, 2001.

[5] D. B. Johnson and D. A. Maltz: Dynamic Source Routing in Ad Hoc Wireless Networks, in Proc. of Mobile Computing 1996, 1996.

[6] J. Kahn, R. Katz and K. Pister: Mobile Networking for Smart Dust, In Proc. of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom99), pp.271–278, 1999.

[7] MICA, http://www.xbow.com/products/Wireless_Sensor_Networks.htm.

[8] K. Sakamura: TRON: Total Architecture, In Proc. of Architecture Workshop in Japan'84, pp. 41–50, 1984.

[9] T. Terada, M. Tsukamoto, K. Hayakawa, T. Yoshihisa, Y. Kishino, A. Kashitani, and S. Nishio: Ubiquitous Chip: a Rule-based I/O Control Device for Ubiquitous Computing, in Proc. of Pervasive2004 , 238–253, 2004.

[10] M. Weiser: The Computer for the Twenty-first Century, Scientific American, Vol. 265, No. 3, pp. 94–104, 1991.