

推薦論文

# モバイルエージェントを用いた格子状ネットワークを構成するユビキタスコンピュータ群の制御

國本 慎太郎<sup>1,a)</sup> 藤田 直生<sup>1,b)</sup> 佐野 渉二<sup>2,c)</sup> 寺田 努<sup>1,3,d)</sup> 塚本 昌彦<sup>1,e)</sup>

受付日 2012年7月17日, 採録日 2013年2月1日

**概要:** 本論文では, 格子状に配置されたユビキタスコンピュータ群に対して, それぞれのコンピュータが搭載する入出力デバイスをモバイルエージェントを用いて制御する手法を提案する. 隣接するユビキタスコンピュータに移動しながら処理が行えるモバイルエージェントを用いることで, ネットワークトポロジを考慮しながらコンピュータ群全体を容易に制御できる. 本研究ではモバイルエージェントのプログラムを実行すべきコマンドの羅列ととらえ, モバイルエージェントの移動, 並列処理, コンピュータの入出力制御などのコマンドを作成し, モバイルエージェントを用いてコンピュータ群を制御するための実行環境を構築した. さらに, コンピュータ数の変化への対応やコンピュータ群の動作変更を容易に行えることを示した.

**キーワード:** ユビキタスコンピューティング, モバイルエージェント, 格子状ネットワーク

## Controlling Ubiquitous Computers in Grid Topology Using Mobile Agents Programming

SHINTARO KUNIMOTO<sup>1,a)</sup> NAOTAKA FUJITA<sup>1,b)</sup> SHOJI SANO<sup>2,c)</sup> TSUTOMU TERADA<sup>1,3,d)</sup>  
MASAHIKO TSUKAMOTO<sup>1,e)</sup>

Received: July 17, 2012, Accepted: February 1, 2013

**Abstract:** In this paper, we propose a new programming model for controlling ubiquitous computers in grid topology. The programming style of using mobile agents that can migrate to neighboring computers enables to control whole computers considering network topologies. In this paper, we define mobile agent as a set of simple commands, which include migration, I/O control, and duplication. We have implemented a platform of our model, and confirmed that our approach could adapt to the change of the number of computer or network topology, and we could change the behaviors of computers by adding agents to the environments.

**Keywords:** ubiquitous computing, mobile agent, grid network topology

### 1. はじめに

近年, 情報機器の小型化や低価格化にともない, 小型のコンピュータが埋め込まれた機器が生活環境のいたるところに存在するユビキタスコンピューティング環境の実現が期待されている. ユビキタスコンピューティング環境では, 個々のコンピュータは小型で処理能力が低いため,

本論文の内容は 2011 年 7 月のマルチメディア, 分散, 協調とモバイル (DICOMO2011) シンポジウム 2011 にて報告され, ユビキタスコンピューティングシステム研究会主査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.

<sup>1</sup> 神戸大学大学院工学研究科  
Graduate School of Engineering, Kobe University, Kobe,  
Hyogo 657-8501, Japan  
<sup>2</sup> 公立はこだて未来大学  
Future University Hakodate, Hakodate, Hokkaido 041-8655,  
Japan  
<sup>3</sup> 科学技術振興機構さきがけ  
PRESTO, Japan Science and Technology Agency, Chiyoda,  
Tokyo 102-0076, Japan  
<sup>a)</sup> s-kunimoto@stu.kobe-u.ac.jp  
<sup>b)</sup> nfujita@port.kobe-u.ac.jp  
<sup>c)</sup> sano@fun.ac.jp  
<sup>d)</sup> tsutomu@eedept.kobe-u.ac.jp  
<sup>e)</sup> tuka@eedept.kobe-u.ac.jp

コンピュータ単体で行う処理よりもコンピュータ群全体で何ができるかが重要である。そのため、ユビキタスコンピュータ群全体を制御するメカニズムや、コンピュータどうしが自律的に連携しあい、ある目的に向かって協調して動作することが求められる。また、システムの使用範囲や環境内のコンピュータの数や配置は頻繁に変わることが想定されるため、コンピュータ数に対するスケラビリティが求められる。そこで本研究では、モバイルエージェントによってユビキタスコンピュータ群の制御を行う手法を提案する。提案手法ではモバイルエージェントを実行すべきコマンドの羅列であるととらえ、エージェントの移動、並列処理、コンピュータが持つ入出力デバイスの制御などのコマンドを作成した。すべてのコマンドを記号または大文字のアルファベット1文字で表すことで、コードを短縮すると同時にプログラム全体を把握しやすいようにした。本論文では格子状に配置したユビキタスコンピュータ群を想定し、それぞれのコンピュータに搭載された入出力デバイスをモバイルエージェントを用いて制御をするための実行環境を構築し、動作確認を行った。

以下、2章で関連研究について説明し、3章で提案するモバイルエージェントについて述べ、4章でモバイルエージェントの使用例について説明し、5章で考察を行い、最後に6章で本論文をまとめる。

## 2. 関連研究

ユビキタスコンピューティング環境実現のためのさまざまな研究が行われている。Teradaら[1]はイベント駆動型ルールに基づき入出力機器を制御するユビキタスコンピュータを提案している。MOTE[2]は、自発的にアドホックネットワークを形成し、マルチホップ機能を備えた小型デバイスであり、nesCと呼ばれるC言語を拡張したプログラミング言語を用いて制御プログラムを記述する。これらは個々のコンピュータごとに制御プログラムを記述するため、複数のコンピュータにまたがる処理を行う場合、それぞれの制御プログラムで整合がとれるように記述する必要があり、開発コストが大きい。

コンピュータ群を制御する研究として、多数のコンピュータを1つのコンピュータを扱うような記述で制御するマクロプログラミングがある。Kairos[3]は、コンピュータ群に対して、複数のコンピュータに及ぶ処理やコンピュータ間のトポロジを用いた処理を1つのプログラムで記述できる。Regiment[4],[5]は関数型プログラミングの概念を取り入れており、センサデータを扱った再帰的な処理を容易に行える。RuleCaster[6],[7]は各コンピュータもしくは複数のコンピュータにまたがる処理をルール形式で記述する。これらのシステムは個々のデバイスへのプログラミングの延長であり、各コンピュータのIDを指定しながら処理を記述する必要があったり、コンピュータの数や配置に

依存して各コンピュータ用プログラムを生成するためコンピュータ数やネットワーク構造が変化する場合にプログラム自体を更新する必要があったりする。

モバイルエージェントに関する研究の例として、Fokら[8],[9]の開発したAgillaでは、専用のAPIを利用して、センサデータを取得するプログラムをコンピュータに配備できる。Tsengら[10]やXuら[11]はワイヤレスセンサネットワークにおいてモバイルエージェントを用いてターゲットを追跡する手法を提案している。前者は、オブジェクトを発見するとモバイルエージェントがオブジェクトのローミングパスを追跡することで、通信とセンシングのオーバーヘッドを大幅に削減している。後者は無線センサネットワークにおけるトラッキングの問題を静的、動的、および予測動的なプランニングアルゴリズムによりシミュレーションして考察している。Ilarriら[12]はモバイルエージェントを用いて分散かつフォールトトレラントな監視手法を提案している。モバイルエージェントは、必要な場所に監視タスクを運搬し、容易にトラッキングを実現する。

格子状ネットワークについては、ノード間のリンクの有無を制御できるマルチホップ環境のテストベッドORBIT[13]など無線センサネットワークの分野で主に研究されている。Zhangら[14]は、単純なアルゴリズムをローカルにホップさせていくことでセンサネットワーク中でのルックアップアルゴリズムを小さなエネルギーで動作させている。一般的に、格子状ネットワークのような規則的で均一なコンピュータの配置は、各コンピュータが全体の中でどのような位置関係にあるかが分かりやすいため、その配置や接続関係を利用してコンピュータ群の制御を行うことが有効であると考えられる。

WAVEは、分散コンピュータ上のモバイルエージェント実行環境である[15]。特定の記号で表されたコマンドやリンク、ノード名などを用いてエージェントを移動させ、ノード情報の記録や出力を行う。しかし、WAVEは主にネットワーク構築やネットワーク管理に用いられる場合が多く、本研究とは目的が異なる。

## 3. モバイルエージェントを用いたユビキタスコンピュータ群の制御手法

### 3.1 アプローチ

本研究では、図1に示すように、隣接するコンピュータとの通信機能を持ち、入出力機器制御を行うコンピュータが数百から数千個程度格子状に接続されている環境において、コンピュータ群を制御する環境を想定する。この環境において、柔軟なユビキタスコンピューティング環境を実現するための要件として下記をあげる。

コンピュータ群の制御が容易：ユビキタスコンピューティング環境では、多数のコンピュータを制御する必要があり、

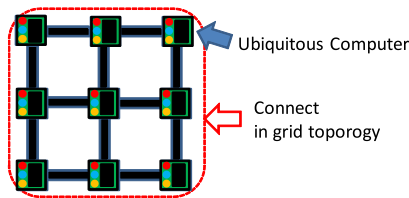


図 1 格子状ネットワーク  
Fig. 1 Network in grid topology.

また、それぞれのコンピュータは小型で処理能力が低いため、コンピュータを連携させることで群全体を制御することが重要である。そのため、簡易的な1つのプログラムで多数のコンピュータを協調制御することが求められる。

コンピュータ群のスケールの変化に対応可能：システム動作中に利用範囲を広げたり新機能を追加するなど、コンピュータの数や配置は頻繁に変化する。このため、コンピュータ数の増減やコンピュータが故障時の環境の再構築にかかる負担を小さくすることが求められる。

長期運用のための省電力化：使用されるコンピュータの多くはバッテリー駆動である。長期間運用するためには、通信量を削減し、省電力化することが求められる。

2章であげた従来手法はコンピュータ数の変化に対応する柔軟性がなかったり、環境の変化に対してプログラムを更新する場合、各コンピュータのファームウェアを更新するための通信量が大きなコストとなる。

筆者らはユビキタスコンピュータ群制御を各コンピュータを個々に制御する分散制御ととらえ、上述の要件を満たすには、各コンピュータのファームウェアを個別に作成するのではなく、コンピュータ群に対する1つのプログラムで各コンピュータが処理と通信を順次行うことで連係動作や一斉制御を達成することに着目した。そこで本研究では、モバイルエージェントによるユビキタスコンピュータ群制御手法を提案する。モバイルエージェントはコンピュータ間の移動性を持つプログラムで、あるコンピュータで実行している処理の状態を記録し、他のコンピュータで引き続いて処理を行える。そのため、時系列に沿って各ユビキタスコンピュータで行う処理、通信を順に記述するだけで複数コンピュータにまたがる処理をするプログラムを容易に作成できる。また、モバイルエージェントは非同期性を持つため、制御プログラムを更新する際、ユビキタスコンピュータのファームウェアを更新する必要がなく、新たなモバイルエージェントを実行すればよい。さらに、モバイルエージェントをコピーして複数のコンピュータ上で並列実行させることにより、大量のユビキタスコンピュータの制御を効率的に行わせることも可能となる。

特に、格子状ネットワークのコンピュータ群を制御するためには、規則的な処理を繰り返すことでコンピュータ群全体を制御することが有効であると考えられる。本論文では、モバイルエージェントの移動や入出力制御、繰り返し処

表 1 進行コマンドと進路変更コマンド  
Table 1 Migration commands.

F	進行方向にコマンド列を移動させる。
R	進行方向を右に変更する。
L	進行方向を左に変更する。
B	進行方向を後ろに変更する。

表 2 入出力制御コマンド  
Table 2 I/O control commands.

Ck	色を指定して、フルカラー LED を発光させる。
C(n <sub>1</sub> ;n <sub>2</sub> ;n <sub>3</sub> )	r 値, g 値, b 値を, それぞれ n <sub>1</sub> , n <sub>2</sub> , n <sub>3</sub> で指定して, フルカラー LED を発光させる。
Mkv	赤外光センサ, 可視光センサを k で指定して, そのセンサ値を v に代入する。

理など基本的な機能を持つコマンドを作成し、規則的な処理の繰り返しを容易に行える実行環境を構築する。なお、コマンドとその処理方法は、筆者らの研究グループで以前構築した WWW 上のアバタプログラミングを参考に行っている [16]。このアバタプログラムは、ブラウザ上で実行され、クッキーメカニズムを使用して他のサイトに移動する。アプリケーションの目的はまったく異なるが、通信負荷を考慮した短い言語の設計や処理メカニズムは類似している。

### 3.2 コマンド

本研究では、モバイルエージェントを実行すべきコマンドの羅列であるとして、ユビキタスコンピュータ群の入出力制御を行うために、さまざまな種類のコマンドを作成した。それらコマンドのすべてを大文字のアルファベットや記号で表すことで、プログラムが短くなるようにし、プログラム全体を把握しやすいようにした。プログラムは1文字ずつ処理される。以下、作成したコマンドの動作について説明する。ここで、 $n, m, l$  は整数、 $v, w$  はそれぞれ変数、配列、 $Z$  は任意のコマンド列とする。

#### 3.2.1 進行コマンドと進路変更コマンド

表 1 に進行コマンドと進路変更コマンドを示す。エージェントは進行方向として、前、後、左、右のいずれかの向きを持ち、これらのコマンドにより、エージェントの移動を制御する。進行コマンド「F」は「F」より後ろのコマンド列を現在の進行方向先のユビキタスコンピュータに移動させる。進路変更コマンドの「R」、「L」、「B」は、モバイルエージェントの進行方向を現在の進行方向に対してそれぞれ右、左、後ろに変更させる。

#### 3.2.2 入出力制御コマンド

表 2 に入出力制御コマンドを示す。現在は入力機器として、可視光センサ、赤外光センサ、出力機器として、フルカラー LED をコンピュータに搭載することを想定している。出力制御コマンド「C」はフルカラー LED を指定色に発光

表 3 繰り返しコマンド

Table 3 Repetition commands.

$m(Z)$	$Z$ を $m$ 実行する.
$*(Z)$	$Z$ を実行し続ける.

表 4 変数コマンド

Table 4 Variable commands.

$Svn$	$v$ を $n$ に設定する.
$Sv(Z)$	$v$ を $Z$ に設定する
$Swl:n$	$w$ の $l$ 番目を $n$ に設定する.
$Ivn$	$v$ を $n$ 増加させる.
$Dvn$	$v$ を $n$ 減少させる.
$Uv$	$Uv$ を変数 $v$ に書き換える.

させる。「 $Ck$ 」では  $k$  として r, g, b, m, y, c, w のいずれかを設定することで、それぞれ赤 (red), 緑 (green), 青 (blue), 紫 (magenta), 黄 (yellow), シアン (cyan), 白 (white) に発光する。「 $C(n_1;n_2;n_3)$ 」は r 値, g 値, b 値として、それぞれ  $n_1, n_2, n_3$  を設定して発光させる。

入力制御コマンド「 $M$ 」は可視光センサ, 赤外光センサのセンサ値を変数に代入する。 $Mk_1v$  の  $k$  として、可視光 (visible) センサを用いるときは  $v$ , 赤外光 (invisible) センサを用いるときは  $i$  を設定することで、そのセンサ値を変数  $v$  に代入する。

3.2.3 繰り返しコマンド

表 3 に繰り返しコマンドを示す。このコマンドはコマンド列の繰り返し処理を行う。整数  $m$  を用いて、 $m(Z)$  とすると  $Z$  を  $m$  回繰り返して実行し、 $*(Z)$  とすると  $Z$  を繰り返して実行し続ける。前者を有限繰り返しコマンド、後者を無限繰り返しコマンドと呼ぶ。

3.2.4 変数コマンド

表 4 に変数コマンドを示す。変数としては、エージェントが保持し、他のコンピュータに移動してもそのエージェントに付随するエージェント変数と、エージェントが保持せず、他のコンピュータに移動しないローカル変数を設けた。変数設定コマンド「 $S$ 」は「 $Svn$ 」, 「 $Svv_1$ 」, 「 $Sv(Z)$ 」とすることで、変数  $v$  の値をそれぞれ整数  $n$ , 変数  $v_1$ , コマンド列  $Z$  に設定する。「 $Swl:n$ 」で配列  $w$  の  $l$  番目のデータ (以降、 $w(l)$  とする) を  $n$  に設定する。「 $Sw:n$ 」とすると、配列  $w$  の最後尾のデータに  $n$  を新たに追加する。「 $Swv:v_1$ 」とすると、配列  $w$  の  $v$  番目のデータを変数  $v_1$  の値に設定し、「 $Swl:w_1l_1$ 」とすると、 $w(l)$  を  $w_1(l_1)$  に設定する。

変数増減コマンド「 $I$ 」, 「 $D$ 」は、「 $Ivn$ 」で変数  $v$  の値を  $n$  増加させ、「 $Dvn$ 」で変数  $v$  の値を  $n$  減少させる。「 $Iv$ 」, 「 $Dv$ 」とすると、変数  $v$  の値を 1 だけ増減させる。

変数書き換えコマンド「 $U$ 」は「 $Uv$ 」とすると、 $Uv$  を変数  $v$  の値に書き換える。変数  $v$  に  $n$  を設定していれば、「 $Uv$ 」は  $n$  に書き換えられ、コマンド列「 $Z$ 」を設定して

表 5 分岐コマンド

Table 5 Branch commands.

$(Z_1;Z_2;\dots;Z_i)$	コマンド列を移動可能な方向へ移動させる
$Evkn(Z_1;Z_2)$	$v$ が $kn$ で指定した条件を満たせば $Z_1$ を、満たさなければ $Z_2$ を実行する。

表 6 その他のコマンド

Table 6 Miscellaneous commands.

$Wm$	次のコマンドの処理を行うまで $m \times 100$ ms 待機する.
$Pm$	LED の発光時間を調整する.
:	コマンドを区切る.
$K$	エージェントの強制終了を行う.

いれば「 $Uv$ 」は  $Z$  に書き換えられる。

3.2.5 並列処理コマンド

並列処理コマンドはエージェントをコピーして分岐させることにより複数のコンピュータにおける並列処理を可能にする。「 $[Z_1;Z_2;\dots;Z_i] Z_{i+1}$ 」により  $Z_1Z_{i+1}, Z_2Z_{i+1}, \dots, Z_iZ_{i+1}$  を順に別コマンドとして実行する。

3.2.6 分岐コマンド

表 5 に分岐コマンドを示す。方向分岐コマンドは、進路変更コマンドを用いて指定した進行先にコンピュータが接続されている場合は移動する。「 $(Z_1;Z_2;\dots;Z_i)$ 」とした場合、 $Z_1$  中で設定された進行方向にコンピュータが接続されているかを調べ、進行可能なら  $Z_1$  を実行し、進行不可能なら  $Z_2$  が同様に実行可能かを調べる。 $Z_2$  が実行可能であれば、 $Z_2$  を実行し、進行不可能なら  $Z_3$  を評価する。以下、同様にどれか 1 つを実行するか、 $Z_i$  まで実行できなかったときに処理を終了する。

変数分岐コマンドは、 $Evkn(Z_1;Z_2)$  とすることで、変数  $v$  と  $n$  の値を  $k$  で設定した条件で評価し、評価結果に応じた処理を実行する。 $k$  は e, n, g, l, b のいずれかを設定する。e, n, g, l は、それぞれ  $v$  が  $n$  と等しい (equal),  $v$  が  $n$  と等しくない (not equal),  $v$  が  $n$  より大きい (greater),  $v$  が  $n$  より小さい (less) ことを示し、b のときは「 $Evbn_1:n_2(Z_1;Z_2)$ 」により、 $v$  が  $n_1$  より大きく、 $n_2$  より小さい (between) ことを満たせば  $Z_1$ , 満たさなければ  $Z_2$  を実行する。

3.2.7 その他のコマンド

表 6 にその他のコマンドを示す。待機コマンド「 $W$ 」は、たとえば  $Z_1WmZ_2$  のコマンド列を処理する際、 $Z_1$  の処理後、 $m \times 100$  ms 待機してから  $Z_2$  を処理する。

点減コマンド「 $P$ 」は、LED の発光時間を調節する。たとえば、 $CkPmZ_1$  とすると、LED は  $m \times 100$  ms 間発光して消灯し、さらに  $m \times 100$  ms 待機してから  $Z_1$  を処理する。

区切りコマンド「 $:$ 」は、複数の数字が並ぶときにそれらを区別するために用いる。たとえば、 $v$  を  $n$  に設定し、 $Z$

表 7 コマンドの処理

Table 7 Command processing.

$\frac{\text{Beforeconversion}}{\text{Afterconversion}}$	コマンド
$\frac{FZ}{Sv_1n_1 \cdots Sw_1l_1 : m_1 \cdots Z}$	進行
$\frac{YZ}{Z}$	進路変更
$\frac{YZ}{Z}$	入出力制御
$\frac{m(Z_1)Z_2}{Z_1m-1(Z_1)Z_2}$	有限繰返し
$\frac{*(Z_1)Z_2}{Z_1*(Z_1)Z_2}$	無限繰返し
$\frac{[Z_1; Z_2; \cdots; Z_s]Z_{i+1}}{Z_1Z_{i+1}, Z_2Z_{i+1} \cdots, Z_sZ_{i+1}}$	並列処理
$\frac{YZ}{Z}$	変数設定, 増減
$\frac{UvZ}{nZ \text{ or } Z_1Z}$	変数使用
$\frac{(Z_1; Z_2; \cdots; Z_s)Z_{i+1}}{Z_1Z_{i+1} \text{ or } Z_2Z_{i+1} \text{ or } \cdots \text{ or } Z_sZ_{i+1}}$	方向分岐
$\frac{Evkn(Z_1; Z_2)Z_3}{Z_1Z_3 \text{ or } Z_2Z_3}$	変数分岐
$\frac{YZ}{Z}$	待機, 点滅, 区切り
$\frac{KZ}{Z}$	終了

を  $m$  回繰り返すようなコマンド列は「 $Svnm(Z)$ 」となり、 $nm$  を 1 つの整数  $t$  とした  $Svt(Z)$  と区別できない。この場合、「 $Svn:m(Z)$ 」とすると、意図した動作になる。

終了コマンド「K」は、それ以降のコマンド列を消去し、エージェントを消滅させる。

### 3.3 コマンドの処理方法

各コンピュータはコマンド列を格納するコマンドメモリを持ち、格納されたコマンド列を表 7 のようにコマンド列を書き換えながら実行する。以下、それぞれのコマンドの処理方法について説明する。

#### 3.3.1 進行コマンドと進路変更コマンドの処理

進行コマンド F の処理方法について示す。メモリに FZ が存在するとき、F を認識すると Z は進行方向先のメモリに格納される。この際、エージェント変数があれば、コマンド列 Z に付与して移動させる。エージェント変数  $v_1, \dots, v_i, v_{i+1}, \dots, v_h, w_1(n_1), \dots, w_j(n_l)$  の値が  $n_1, \dots, n_i, Z_1, \dots, Z_h, m_1, \dots, m_j$  に設定されている場合、コマンド列は  $Sv_1n_1 \cdots Sv_in_i Sv_{i+1}(Z_1) \cdots Sv_h(Z_h) Sw_1n_1 : m_1 \cdots Sw_jn_l : m_j$  と書き換えられる。進路変更コマンドは、進路変更コマンドを Y として YZ を実行する場合、Y を認識するとエージェントの方向を変更して Y を削除する。

#### 3.3.2 入出力制御コマンドの処理

入出力制御コマンドを Y として、メモリに YZ が存在する場合、Y を実行してから削除する。

#### 3.3.3 繰返しコマンドの処理

有限繰返しコマンド  $m(Z_1)Z_2$ 、無限繰返しコマンド  $*(Z_1)$  の処理を示す。整数が先頭にあるコマンドは有限繰返しコマンドであるため、整数  $m$  を認識するとコマンド列から  $Z_1$  を探し、 $m(Z_1)Z_2$  を  $Z_1m-1(Z_1)Z_2$  と書き換える。無限繰返しコマンドは  $*$  を認識するとコマンド列から  $Z_1$  を探し、 $*(Z_1)$  を  $Z_1*(Z_1)$  と書き換える。

#### 3.3.4 変数コマンドの処理

変数設定・変数増減コマンドを Y として、メモリに YZ が存在する場合、Y を実行して削除する。変数使用コマンド  $UvZ_1$  がメモリに存在した場合、 $Uv$  を変数  $v$  の値で書き換える。

#### 3.3.5 並列処理コマンドの処理

並列処理コマンド  $[Z_1; Z_2; \cdots; Z_i] Z_{i+1}$  の処理は、 $[$  を認識すると、コマンド列の中から  $Z_1$  と  $Z_{i+1}$  を探して  $Z_1Z_{i+1}$  を実行し、 $Z_1;$  を削除する。

#### 3.3.6 分岐コマンドの処理

方向分岐コマンド  $(Z_1; Z_2; \cdots; Z_i) Z_{i+1}$  の処理は、 $($  を認識すると、 $Z_1$  を探し、 $Z_1$  の中で指定された進行方向にコンピュータが接続されているかを調べる。つながっていれば、 $(Z_1; Z_2; \cdots; Z_i) Z_{i+1}$  を  $Z_1Z_{i+1}$  に書き換え、つながっていなければ  $(Z_2; \cdots; Z_i) Z_{i+1}$  に書き換える。変数分岐コマンド  $Evkn(Z_1; Z_2)Z_3$  の処理は、E を認識したときに変数  $v$  の値と  $n$  の値を  $k$  の条件で判定し、条件を満たせば  $Z_1Z_3$  に書き換え、満たさなければ  $Z_2Z_3$  に書き換える。

#### 3.3.7 その他のコマンドの処理

待機コマンド、点滅コマンド、区切りコマンドを Y とすると、Y を認識すると該当の処理を実行して Y を削除する。終了コマンド K に関しては、KZ がメモリに存在する場合、K を認識するとそれ以降のコマンド列 Z をコマンドメモリから消去する。このときローカル変数は消去しない。

### 3.4 実装

前節で述べたコマンドを扱える実行環境を構築した。図 2 にシステム構成を示す。制御対象であるユビキタスコンピュータ、コンピュータの接続線、コンピュータへのプログラム書き込み機（ライター）、シリアル通信用アプリケーションは筆者らの研究グループで作成したものである。図 3 に示すユビキタスコンピュータの大きさは 20mm 四方で、MCU としては Atmel 社の ATMEGA644P を備え、フルカラー LED (RGB 各色 8bit PWM 制御)、可視光センサ、赤外光センサを搭載し、通信線で隣接するユビキタスコンピュータと 4 方向までシリアル通信 (TTL UART 115.2kbps) が行える。ユビキタスコンピュータにはコマ

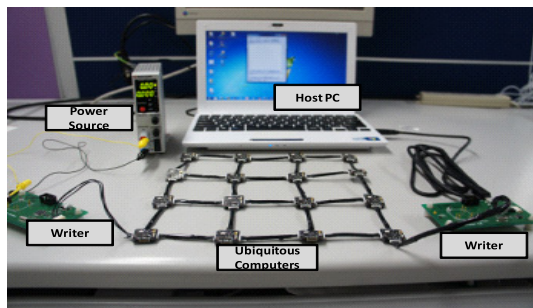


図 2 システム構成

Fig. 2 The system configuration.

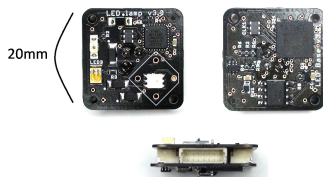


図 3 制御対象の小型デバイス

Fig. 3 Our developed ubiquitous computer.

ンド列の処理を行うエンジンを持つ。メモリの制約を考慮し、3.2 節で述べた整数  $n, m, l$  はそれぞれ 1Byte の符号なし整数, 2Byte の符号なし整数, 0 から 9 の整数で, 変数  $v$  はエージェント変数として  $g$  から  $w$  の 17 種類, ローカル変数として  $x, y, z$  の 3 種類, 配列  $w$  はエージェント変数として  $a, b, c$  の 3 種類, ローカル変数として,  $d, e, f$  の 3 種類とした。エンジンのコード量は 22,692 バイトとなった。

ホスト PC とユビキタスコンピュータ群は、ライタを経由して接続する。PC とライタは USB 接続である。ユビキタスコンピュータはバッテリーを搭載していないため、電源を同じく別のライタ経由でユビキタスコンピュータ群と接続する。PC からコマンドを送信すると、PC と接続されているコンピュータのコマンドメモリに格納される。このとき、ユビキタスコンピュータが下の通信ポートからモバイルエージェントを受信するとき、モバイルエージェントは上方向の向きとされる。

#### 4. モバイルエージェントの利用例

前述のコマンドを組み合わせたモバイルエージェントの使用例を示す。以降、説明のために図 4 のようにコンピュータに ID を振る。

コンピュータ 11 の LED を青色に発光させる。

[例] FFLFFCb

[説明] コンピュータ 1, 2, 3, 7, 11 の順にエージェントを移動させ、コンピュータ 11 の LED を青色に発光させる。図 5 に処理の流れを示す。コンピュータ 1 から FF により 2 つ進み、コンピュータ 3 へ移動する。その後 L により左に進路を変更し、FF によりコンピュータ 11 に到達する。

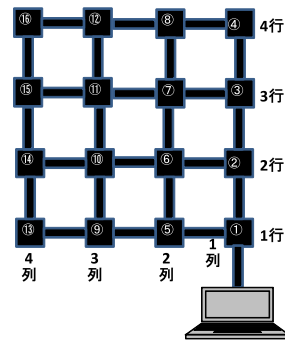


図 4 各コンピュータの ID 振り分け

Fig. 4 Assignment of ID for each computer.

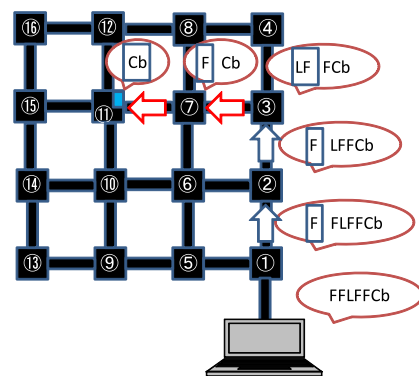


図 5 コンピュータ 11 を発光させる処理の流れ

Fig. 5 Flow of lighting LED on the computer 11.

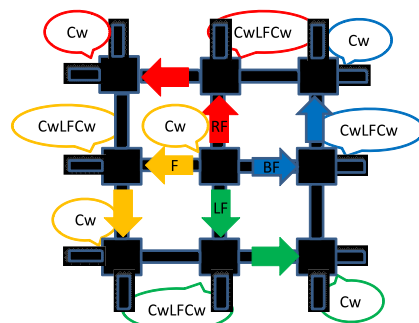


図 6 暗い部分とその周辺を明るくする処理の流れ

Fig. 6 Flow of lighting LEDs around dark area.

コンピュータ 11 では Cb で LED を青色に発光させる。このとき、コンピュータ 11 への経路以外のコンピュータは通信も処理も行わないため、全体として通信量や消費電力が小さくなる。

暗い部分とその周辺を明るくする。

[例] FFLFFMiyEyg200([RF;LF;BF;CwF]CwLFCw;K)

[説明] コンピュータ 11 の赤外光センサの値を読み取り、コンピュータ 11 とその周辺を明るくする。図 6 に処理の流れを示す。まず FFLFF によりコンピュータ 11 に移動し、Miy により赤外光センサの値を読み取って変数  $y$  に格納する。Eyg200100 で、変数  $y$  の値が 200 より大きければ [RF;LF;BF;CwF]CwLFCw によりユビキタスコンピュータ 11 の 4 方向に CwLFCw を移動させることで、コンピュ

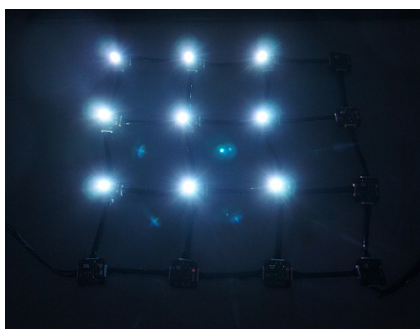


図 7 実行結果

Fig. 7 Result of processing.

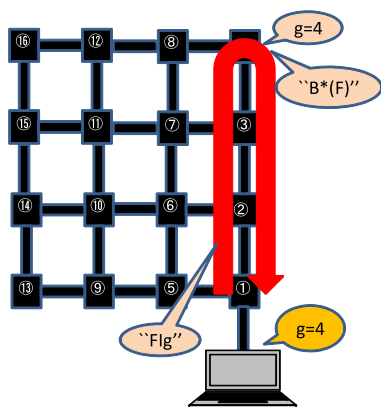


図 8  $Sg1*((FIg;B*(F)))$  の処理

Fig. 8 Process of  $Sg1*((FIg;B*(F)))$ .

タ 11 の周りも LED を発光させる。変数  $y$  の値が 200 より小さければ  $K$  により処理を終了する。実行例を図 7 に示す。このように、条件分岐によりエージェントを複製移動させて複数のコンピュータの入出力制御を同時に行える。ユビキタスコンピュータの数をカウントする。

[例]  $Sg1*((FIg;B*(F)))$ ,

$[FFF;FF;F;:]LSg1*((FIg;B*((F;RF))))$

[説明] エージェントが前方に移動するたびにエージェント変数を増加させることでユビキタスコンピュータの数を計測する。図 8 に処理の流れを示す。このエージェントによって返ってきた  $g$  の値によって格子状ネットワークが何行で構成されているか分かる。まず、 $Sg1$  でコンピュータ 1 で  $g$  を 1 に設定し、 $*((FIg;B*(F)))$  において前進できるため  $FIg*((FIg;B*(F)))$  によりコンピュータ 2 へ移動する。コンピュータ 2 にはエージェント変数が付随した  $Sg1Ig*((FIg;B*(F)))$  が移動し、同様にコンピュータ 3 には  $Sg2Ig*((FIg;B*(F)))$ 、コンピュータ 4 には  $Sg3Ig*((FIg;B*(F)))$  が移動していく。コンピュータ 4 ではこれ以上移動できないため、 $(FIg;B*(F))$  の  $B*(F)$  が実行され、進路を後ろに変更し、ホスト PC へ戻る。戻る際は  $Ig$  を実行しないため  $g$  は増加せず、エージェントが  $g$  の値を持ってホスト PC に戻る。ホスト PC ではエージェントの処理系が組み込まれていないため、コマンド処理が

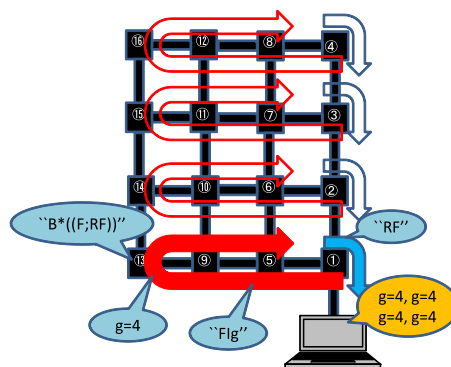


図 9  $[FFF;FF;F;:]LSg1*((FIg;B*((F;RF))))$  の処理

Fig. 9 Process of  $[FFF;FF;F;:]LSg1*((FIg;B*((F;RF))))$ .

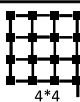
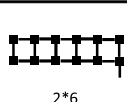
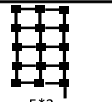
Scale	 4*4	 2*6	 5*3
Result of $``Sg1*((FIg;B*(F)))``$	$g=4$	$g=2$	$g=5$
Part of $X_k$ $``X_kLSg1*((FIg;B*((F;RF))))``$	$[FFF;FF;F;:]$	$[F;:]$	$[FFFF;FFF;FF;F;:]$
Result of $``X_kLSg1*((FIg;B*((F;RF))))``$	$g=4, g=4$ $g=4, g=4$	$g=6$ $g=6$	$g=3, g=3$ $g=3, g=3, g=3$

図 10 3つのネットワークへの実行結果

Fig. 10 Results of processing programs in three environments.

行われず処理を終了する。ホスト PC 上では戻ってきた  $g$  の値が確認でき、このネットワークが 4 行で構成されていることが分かる。

次に  $LSg1*((FIg;B*((F;RF))))$  の処理の流れを図 9 に示す。このエージェントが返す  $g$  の値によって格子状ネットワークが何列で構成されているかが分かる。先ほどのエージェントによりネットワークが 4 行構成であることが分かったため、 $LSg1*((FIg;B*((F;RF))))$  の先頭に  $[FFF;FF;F;:]$  を加え、各行のコンピュータに  $LSg1*((FIg;B*((F;RF))))$  のコマンド列を移動させる。あとは上記と同様のパターンで、コンピュータ 1, 2, 3, 4 から出発したエージェントはコンピュータ 13, 14, 15, 16 に達したときそれぞれ変数  $g$  を 4 に設定している。コンピュータ 13, 14, 15, 16 では左方向に移動できないため、 $(FIg;B*((F;RF)))$  の  $B*((F;RF))$  により進路を後ろに変更し、ホスト PC へ戻る。この際、 $(F;RF)$  により前進と右折の方向分岐させているため、コンピュータ 1, 2, 3, 4 に戻ってきた際、そこから右折してホスト PC の方向へ進むことになる。戻ってきた  $g$  の値から、この格子状ネットワークにあるコンピュータの数は  $4 \times 4$  で 16 であると分かる。

[実験] このエージェントを 3 つのネットワークで実行した結果を図 10 に示す。1 段階目のエージェントはネットワーク構造によって実行結果が変わるため、2 段階目のエージェントにはそれによって最初の並列部分を変える必要がある。3 つすべてでコンピュータ数を正しく計測でき、

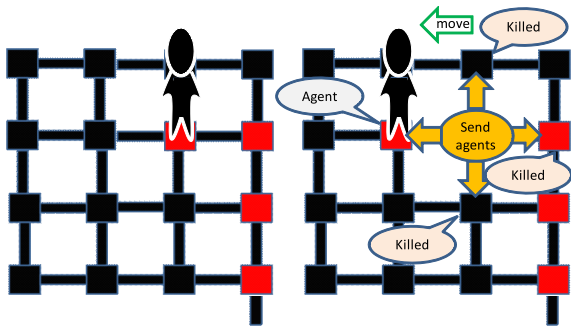


図 11 処理の流れ  
Fig. 11 Flow of object tracking.

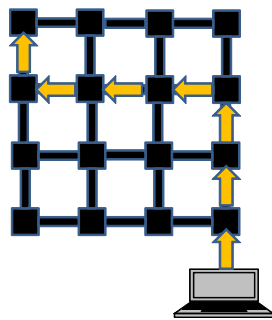


図 12 移動経路  
Fig. 12 Migration path of the object.

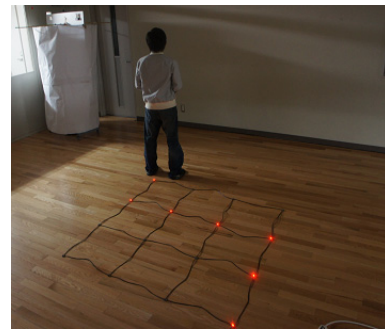


図 13 トラッキング結果  
Fig. 13 Result of object tracking.

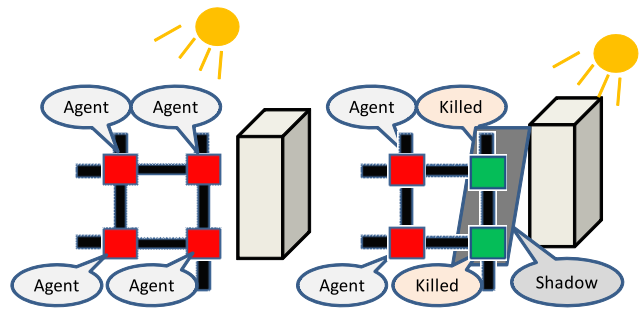


図 14 日照時間測定プログラムの処理  
Fig. 14 Flow of sunshine duration measurement.

エージェントを用いればネットワーク構造や規模の変化に柔軟に対応できることが分かった。

オブジェクトトラッキング

[例]  $*(MiyEyg200(W10Cr;[B;R;L;:]FMiyEyg200(;K)))$   
 [説明] 追跡対象者の真下が暗くなることを利用し、対象者を追跡する。図 11 に処理の流れを示す。もし対象オブジェクトがコンピュータの真上にいれば、そのコンピュータでは MiyEyg200 が満たされ、W10Cr により 1 秒間隔で LED が赤く発光する。対象が移動して MiyEyg200 が満たされなくなると、 $[B;R;L;:]F$  により 4 方向へそれぞれ MiyEyg200(;K)\* $(MiyEyg200(W10Cr;[B;R;L;:]FMiyEyg200(;K)))$  を移動させる。移動先に対象がいなければ MiyEyg200 が満たされず、K により処理が終了する。対象が存在すればコマンド列  $*(MiyEyg200(W10Cr;[B;R;L;:]FMiyEyg200(;K)))$  の処理を続ける。

[実験] このエージェントの動作確認を行った。床に 500mm 間隔にデバイスを配置し、対象者には図 12 の経路を歩かせた。図 13 に示すように移動経路上のユビキタスコンピュータの LED のみ赤色に点灯させることができた。

日照時間測定

[例]  $*(MizEzg200(CgK;CrIxExe60(Sx0Iy;:)W600))$   
 [説明] 赤外光センサを用いてその部分が日向か日陰かを判断し、日照時間を測定する。図 14 に処理の流れを示す。LED を点灯させているのは処理の継続、終了を視覚的に確認するためである。もしコンピュータが日向にあれば、その部分は明るくなるため MizEzg200 は満たされず、CrIxExe60(Sx0Iy;:)W600 が実行される。これは x を 1 増やし、x が 60 でなければ W600 により 1 分間待機し、x が 60 ならば y を 1 増やして x を 0 に設定し 1 分間待機する。この処理により、x で分、y で時をカウントできる。コンピュータが日陰にあればその部分は暗くなるので CgK により処理を終了する。最後に各コンピュータの x、y の値を確認する。



(a) 45 分後の結果



(b) 65 分後の結果

図 15 日照時間測定プログラムの実行結果

Fig. 15 Result of sunshine duration measurement.

この処理により、x で分、y で時をカウントできる。コンピュータが日陰にあればその部分は暗くなるので CgK により処理を終了する。最後に各コンピュータの x、y の値を確認する。

[実験] このエージェントを実験環境で動作させた。エージェントを各コンピュータで実行するためにコマンド列の先頭に  $[FL;L][FF;F;:]$  を加えている。コンピュータを 500mm 間隔に配置し、45 分後にコンピュータ 1, 2, 65 分後にコンピュータ 3, 4 が陰になるように板で覆った。90 分後、移動した先のコンピュータの変数を順番に配列 a, b に格納しホスト PC に戻ってくるエージェント  $L*(Sa:xSb;y(F;RF))$  により、各コンピュータの変数を取得した。実行結果を図 15 に示す。この際、各コンピュータの測定時間は、ID1



から順に 44 分, 44 分, 65 分, 65 分, 90 分, 90 分となり, エージェントが想定どおり動作したことが分かる。

## 5. 考察

本研究のように隣接間通信を行う小型デバイスが格子状に接続された環境に対し, ユビキタスコンピュータごとに記述されたファームウェアをそれぞれのコンピュータに組み込んで処理を行う手法 (従来手法 1), Kairos のようにユビキタスコンピュータ群全体に対して記述されたプログラムから個々のコンピュータ用のプログラムを生成し, 個々のコンピュータにプログラムを配信する手法 (従来手法 2) と比較しながら提案手法の有効性を考察する。従来手法 1 については通信を介してファームウェアを更新することを想定する。

### 5.1 ユビキタスコンピュータ群の制御方式

ユビキタスコンピューティング環境では, 多数のコンピュータを容易に制御できることが求められる。そのため, 「各ユビキタスコンピュータの ID の把握がいらぬ」「コンピュータどうしの連携がとりやすい」「コンピュータどうしの連携がとりやすい」などの特徴が必要になる。また, 複数のタスクの並列実行が容易であることで, コンピュータ群を効率的に利用できる。

従来手法 1 では, 個々のコンピュータに対して, ファームウェアを作成する必要がある。コンピュータ間で連携を行う場合には, システム全体で整合をとりながらメッセージ通信を行う必要があり, 労力と時間に関するコストが大きい。また, コンピュータの ID とその配置場所を把握しておく必要もある。複数のタスクを実行する場合には, それぞれのタスクを個々のコンピュータで行う処理に分解して記述しなければならず, 1 つのプログラム上に複数のタスクについて記述するため, バグが生じやすい。従来手法 2 では, 複数のコンピュータにまたがる処理であっても実行タスクについて記述するだけでよく, 個々のファームウェアを生成する際に, 連携のためのメッセージ処理も生成されるため, 従来手法 1 に比べ, 開発コストが小さい。しかし, ID を指定してユビキタスコンピュータの処理を記述することや 1 つのプログラム上に複数のタスクについて記述することの困難さは同様である。

提案手法では, コンピュータ群で行う処理を単純なコマンドを組み合わせた 1 つのプログラムで実現する。プログラムが移動するため, 複数コンピュータにまたがる処理も進行コマンドを組み合わせることで実現でき, コンピュータ間でのメッセージ通信やコンピュータの ID を意識せずに, コンピュータ間で連携をとった処理を行える。コンピュータの配置やトポロジを把握する必要があるが, 4 章で示したようにコンピュータの位置取得やトポロジ発見をプログラムで行うことで, 遠方にあるコンピュータの制御

も進行コマンドと繰返しコマンドを組み合わせることで簡単に記述できる。さらに, 複数のタスクを実現する場合は, タスクごとにコマンド列を記述して, 複数のエージェントを実行するだけでよく, 複数タスクが混在するシステムも容易に実現できる。

### 5.2 スケーラビリティ

ユビキタスコンピューティング環境では, 使用目的や場所に応じてコンピュータの数や配置が変化するため, それに応じてシステムを修正する必要がある。そのため, コンピュータ数が増えた際やコンピュータが故障した際のシステム再構築にかかる負担を小さくすることが求められる。

従来手法 1 では, システムを拡張するために増やしたコンピュータの処理がそれまでのシステムに影響を与える場合には, 新たに追加したコンピュータだけでなくシステム全体のコンピュータのファームウェアを更新する必要がある。開発コストが大きい。従来手法 2 では, 記述したプログラムから各コンピュータのファームウェアを生成するため開発コストは小さくなるが, プログラムはコンピュータの数や配置に依存しているため環境全体のコンピュータ上のプログラムを更新しなければならない場合が生じる。

提案手法では, 環境に追加するコンピュータにエージェントの処理エンジンを組み込む必要があるが, これは環境に依存しないため, 開発コストはほぼない。また, 4 章の例で示したように, コンピュータ数が増えたり減ったりしても, 実行したいプログラムに移動コマンドなどを加えるだけで容易にアプリケーションを修正できたり, 「オブジェクトトラッキング」の例のように, そのまま同じプログラムが動作する。

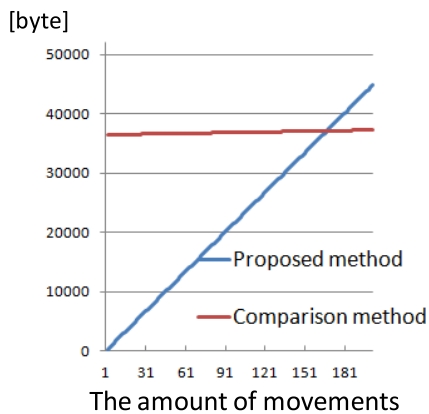
### 5.3 ユビキタスコンピュータの通信量

ここでは 4 章で示したオブジェクトトラッキングを例に従来手法と通信量を比較する。従来手法では, 小型デバイスである Arduino を用い, Arduino に LED, 赤外光センサを装着したものを格子状に 16 個並べることを想定する。図 16 は, 上述のようなトラッキングを行うために

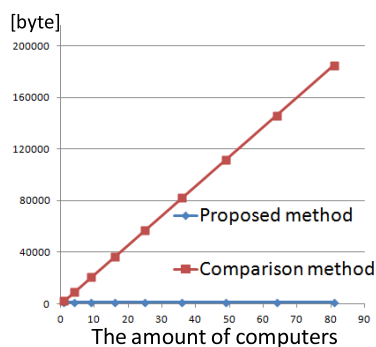
```
int x; int y;
void setup(){Serial.begin(9600);}
void loop(){
  if (Serial.available() > 0) {
    x = Serial.read(); //Receive data(message)
    y = analogRead(pin); //Read sensor value
    if(y > 200){
      digitalWrite(pin, HIGH); //Make LED emit light
      while(1){
        y = analogRead(pin); //Read sensor value
        if(y < 200){
          Serial.write(data); //Send data(message)
          break;
        }
      }
    }
  }
}
```

図 16 オブジェクトトラッキングを達成するプログラム

Fig. 16 A program for object tracking.



(a) トラッキング数に対する通信量の比較



(b) コンピュータ数に対する通信量の比較

図 17 従来手法と提案手法の通信量比較

Fig. 17 Traffic comparison between two methods.

Arduino に組み込むプログラムの入出力に関する部分を抜粋したものである。

総通信量  $D$  は  $D_0$  を生成プログラムのサイズ,  $D_1$  をメッセージ量,  $x$  をコンピュータ数,  $y$  を移動回数とすると次のように表される。

$$D = D_0x + 4D_1y \text{ [byte]} \quad (1)$$

従来手法では図 16 に示すプログラム量を書き込むため  $D_0$  は 2,126 byte,  $D_1$  は 1 byte なので,

$$D = 2126 * 16 + 4 * 1y = 34016 + 4y \text{ [byte]} \quad (2)$$

となる。

提案手法では, モバイルエージェントが追跡対象とともに移動していくため, 最初ホスト PC に送るプログラムは「\*(MiyEyg200(Cr;[B;R···])」であり  $D_0$  は 43 byte, メッセージ量は移動先のコンピュータを制御するプログラム「MiyEyg200(;K)\*(MiyEyg200(Cr;···)」が必要なため,  $D_1$  は 56 byte となる。したがって,  $D$  は

$$D = 43 + 4 * 56 * y = 43 + 224y \text{ [byte]} \quad (3)$$

となる。図 17 (a) に移動回数に対する通信量の変化を示す。長期運用をした場合, 提案手法よりも従来手法のほうが通信量を抑えられることが分かる。しかし, ユビキタス

コンピューティング環境下では, 周囲の状況や実行するタスクが変わったときに, コンピュータの動作を変更するために制御プログラムを更新する必要があり, 従来手法では, コンピュータそれぞれのファームウェア更新に多くの通信量を要する。提案手法では, ファームウェアを更新する必要がなく, 新たなモバイルエージェントを実行すればよい。したがって, ファームウェア更新の頻度によってどちらのアプローチが優れているかが変化することになる。

環境内のコンピュータ数が変化した場合, オブジェクトトラッキングを行うには, トラッキング数を 20 回とすると従来手法では,

$$D = 2126 * x + 4 * 20 = 2126x + 80 \text{ [byte]} \quad (4)$$

となる。提案手法では, エージェントは追跡対象者とともに移動していくため環境内のコンピュータ数に依存せず, 従来手法のように各コンピュータのファームウェアを更新する必要がない。よって提案手法の通信量は

$$D = 43 + 4 * 56 * 20 = 4523 \text{ [byte]} \quad (5)$$

となる。図 17 (b) にコンピュータ数に対する通信量の結果を示す。このように提案手法を用いることでコンピュータ数が多い環境では従来手法と比べ非常に小さい通信量でアプリケーションが実現できる。

## 6. おわりに

本論文では, 格子状に配置したユビキタスコンピュータ群に対して, それぞれのコンピュータに組み込まれた入出力デバイスを制御するためのプログラミングを想定し, モバイルエージェントによる制御を行う環境を実現した。モバイルエージェントを実行すべきコマンドの羅列であるととらえ, エージェントの移動, 並列処理, 入出力制御などの機能を持つコマンドを作成した。プロトタイプとして, 筆者らの研究グループで作成した小型デバイスを使用したエージェント実行環境を実装し, さまざまな実例をあげながら提案手法が実環境で動作することを確認した。

今後の課題としては, このシステムを用いた実環境でのユビキタスシステムの作成, 評価が考えられる。

謝辞 本研究の一部は, 文部科学省科学研究費補助金基盤研究 (A) (23240010) によるものである。ここに記して謝意を表す。

## 参考文献

- [1] 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. International Conference on Pervasive Computing (Pervasive 2004)*, pp.238-253 (2004).
- [2] Warneke, B., Last, M., Liebowitz, B. and Pister, K.: Smart Dust: Communicating with a Cubic-Millimeter Computer, *Proc. IEEE Computer Magazine*, Vol.34,

- No.1, pp.44-51 (2001).
- [3] Gummadi, R., Gnanawali, O. and Govindan, R.: Macro-Programming Wireless Sensor Networks Using Kairos, *Proc. International Conference on Distributed Computing in Sensor Systems (DCOSS2005)*, pp.126-140 (2005).
  - [4] Newton, R., Morrisett, G. and Welsh, M.: The Regiment Macroprogramming System, *Proc. 6th International Conference on Information Processing in Sensor Networks (IPSN2007)*, pp.489-498 (2007).
  - [5] Newton, R. and Welsh, M.: Region Streams: Functional Macroprogramming for Sensor Networks, *Proc. 1st Int. Workshop on Data Management for Sensor Networks (DMSN2004)* (2004).
  - [6] Urs, B. and Gerd, K.: RuleCaster: A Macroprogramming System for Sensor Networks, *Proc. 21st Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA2006)* (2006).
  - [7] Urs, B. and Gerd, K.: A State-Based Programming Model and System for Wireless Sensor Networks, *Proc. 3rd International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS2007)*, pp.261-266 (2007).
  - [8] Fok, C.-L., Roman, G.-C. and Lu, C.: Rapid development and flexible deployment of adaptive wireless sensor network applications, *Proc. 25th IEEE International Conference on Distributed Computing Systems (ICDCS2005)*, pp.653-662 (2005).
  - [9] Fok, C.-L., Roman, G.-C. and Lu, C.: Mobile agent middleware for sensor networks: An application case study, *Proc. 6th International Conference on Information Processing in Sensor Networks (IPSN2005)*, pp.382-387 (2005).
  - [10] Tseng, Y.-C., Kuo, S.-P., Lee, H.-W. and Huang, C.-F.: Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies, *Proc. Computer Journal*, Vol.47, No.4, pp.448-460 (2004).
  - [11] Xu, Y. and Qi, H.: Mobile agent migration modeling and design for target tracking in wireless sensor networks, *Ad Hoc Networks*, Vol.6, No.1, pp.1-16 (2008).
  - [12] Ilarri, S., Mena, E. and Illarramendi, A.: Using cooperative mobile agents to monitor distributed and dynamic environments, *Information Sciences*, Vol.178, No.9, pp.2105-2127 (2008).
  - [13] Raychaudhuri, D.: Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols, *Proc. Wireless Communications and Networking Conference (WCNC2005)*, Vol.3, pp.1664-1669 (2005).
  - [14] Zhang, C. and Herman, T.: Localization in Wireless Sensor Grids, *Computers and Their Applications*, pp.388-393 (2006).
  - [15] Sapaty, P.-S.: Mobile processing in open systems, *Proc. 5th IEEE High Performance Distributed Computing (HPDC1996)*, pp.182-191 (1996).
  - [16] Loh, Y.-H., Ogawa, T., Tsukamoto, M. and Nishio, S.: Avatar Programming in a Virtual Space on the World Wide Web, *Proc. 6th International Workshop on Multimedia Information Systems (MIS 2000)*, pp.42-51 (2000).

推薦文

本論文は格子状に接続された入出力を持つ小型コンピュー

タどうしを自己組織的に制御する手法について論じており、その新規性や有効性といった研究上の貢献が大きいため推薦する。

(ユビキタスコンピューティングシステム研究会主査  
 椎尾一郎)



國本 慎太郎

2011年神戸大学工学部電気電子工学科卒業。同年より同大学院工学研究科電気電子工学専攻博士前期課程，現在に至る。



藤田 直生 (正会員)

2002年奈良工業高等専門学校卒業，2004年同専攻科修了。2006年大阪大学大学院工学研究科博士前期課程修了，2010年同経済学研究科博士前期課程修了，2011年神戸大学大学院工学研究科博士課程退学。同年4月神戸大学工学研究科学術推進研究員となり現在に至る。工学および経営学修士。ユビキタス，ウェアラブルコンピューティングの研究に従事。電子情報通信学会，土木学会等の各会員。



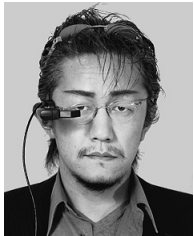
佐野 渉二 (正会員)

2004年神戸大学工学部電気電子工学科卒業。2006年同大学院自然科学研究科博士前期課程修了。2012年同大学院自然科学研究科博士後期課程修了。同年4月より神戸大学大学院工学研究科学術推進研究員，同年11月より公立ほこだて未来大学特別研究員となり，現在に至る。博士(工学)。スマートシティはこだてプロジェクトに関する研究に従事。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興味を持つ。IEEE，ヒューマンインタフェース学会の各会員。



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。2007年神戸大学大学院工学研究科准教授、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、2005年には同機構事務局長を兼務。2004年には英国ランカスター大学客員研究員を兼務。博士(工学)。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, ヒューマンインタフェース学会の各会員。



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。