モバイルコンピューティングと
ワイヤレス通信 24−31
（2003．3．7）

モバイル環境向けエージェント移動制御方式の実装

川上 憲治† 広重 一仁† 吉岡 信和‡ 本位田 真一‡＊

あらまし 近年，携帯電話や PDA などの携帯端末が広く普及し，利用場所を選ばずに，様々なアプリケーションを利用することが可能となった．しかしながら，携帯端末はバッテリの持続時間や小型化のため，アプリケーションで利用できる計算資源が限られるという制約がある．この制約を解決するために，筆者らはモバイルエージェントを基盤とするミドルウェアについて提案してきた．本稿では，筆者らが提案したモバイルエージェントを基盤とするミドルウェアの実装例について説明し，実装したシステムの動作結果をもとに，筆者らのアプローチの有効性について考察する．

キーワード 携帯端末，モバイルエージェント，移動制御

Implementation of Agent Migration Control Methods for Mobile Environments

Kenji KAWAKAMI† Kazuhito HIROSHIGE† Nobukazu YOSHIOKA‡ and Shinichi HONIDEN‡＊

Abstract In recent years, mobile terminals such as cellular phones and PDAs have come into widespread use and various applications have been used anytime and anywhere. However, applications executed on the mobile terminals lack sufficient computational resources because of limited battery lifetime and terminal miniaturization. In order to overcome such a limitation, we have proposed the mobile agent-based middleware. In this paper, we describe our implementation of the proposed middleware system. And we discuss the effects of our approach from the results of implemented system behavior.

Keyword Mobile Terminal, Mobile Agent, Migration Control

1. はじめに

近年，携帯電話や PDA などの携帯端末が広く普及し，利用場所を選ばずに，様々なアプリケーションを利用することが可能となった．また，個人利用に限らず，企業の情報システムにおいても携帯端末を利用する事例が増えており，携帯端末の重要性は年々高まりつつある．

しかしながら，携帯端末はバッテリの持続時間や小型化のため，アプリケーションで利用できる計算資源(CPU，メモリ，通信帯域幅など)が限られるという制約がある．これに対して，アプリケーションが他のコンピュータの豊富な計算資源を利用でき，また，計算資源や通信環境の変化にも適応できる機構があれば，ユーザは携帯端末の制約を意識することなくアプリケーションを利用できる．

筆者らはこのような機構を実現する手法として，モバイルエージェントを基盤とするミドルウェア(MA-based Middleware)について検討してきた[1][2]．ここで，モバイルエージェントはミドルウェアが提供するライブラリとして動作し，他の端末に移動してその資源を利用し処理を実行する．

本稿では，まずモバイル環境における課題を明らかにし，その課題に対する解決法である MA-based Middleware および提案システムに導入したエージェント移動制御方式について説明する．そして，マルチエージェントフレームワーク：Beegent[4]を利用した提案システムの実装例を示す．最後に，実装した提案システムの動作結果をもとに，筆者らのアプローチの有効性について考察する．

2. モバイル環境における課題と解決法

2.1. 携帯端末の制約

携帯端末では，バッテリの持続時間や携帯性のため，CPU やメモリといったアプリケーションから利用できる資源が制限される．今後，携帯端末に搭載される CPU の性能向上やデバイスの高機能化が進むことが予想されるが，固定の環境で利用されるコンピュータと比べて処理

†日本テレコム株式会社, Japan Telecom
‡国立情報学研究所, National Institute of Informatics
＊東京大学, The University of Tokyo

能力の本質的な差異は存在し続けるであろう．しかし，このような端末の処理能力差から利用できるデータやアプリケーションが限定されてしまうと，携帯端末の利便性は大きく損なわれることになる．

2.2. 課題解決の方針

携帯端末における計算資源の制約を克服する方法として，携帯端末内の限られた計算資源を効率的に利用する手法と，他のコンピュータの計算資源を利用する手法が考えられる．本稿では，後者の手法により携帯端末の限られた処理能力を補う手法について述べる．

ネットワーク上のコンピュータの計算資源を利用する代表例として，クライアント／サーバ型コンピューティングが挙げられる．しかしながら，固定的なクライアント／サーバ型モデルでは，携帯端末がネットワークに接続できない状況では，クライアント側で処理を継続することができない．モバイル環境では，携帯端末がネットワークに接続できない状況においても，携帯端末上でオフラインに処理を実行できる機構が必要となる．

また，ネットワーク上のコンピュータを利用する場合，負荷集中やデータ転送量が多い場合には，必ずしもネットワーク側で処理することが処理の高速化にはつながらない．したがって，状況に応じて他の端末の豊富な計算資源を柔軟に利用する機構が必要となる．

2.3. Mobile Agent の利用

筆者らは，他の端末の計算資源を柔軟に利用するための方法として，モバイルエージェント(以下，MA と略す)を用いたアプローチを採用した．ここでは，アプリケーションが必要とするライブラリを MA として構成し，このライブラリが実行に適した環境(端末)に移動して，その計算資源を利用する．

MA を用いたアプローチの利点としては，アプリケーションはライブラリが提供するサービス(機能)だけを知っていればよく，ライブラリに適した実行環境を知る必要がない．ライブラリ自身が，実行時点での環境の中で自分に適した実行環境を選択して移動するので，アプリケーションはライブラリに対して実行要求を投げるだけでよい．これは，エージェントの自律性によって実現される．

また，本稿では「計算資源の制限」に焦点を当て，以下の内容では「実行に適した ＝ レスポンスがよい」と定義するが，ライブラリの種類によって実行に適した環境も多様である．単にライブラリのプログラムコードだけを移動するのではなく，実行に適した環境を評価する基準を MA の内部に定義し，MA の移動と共に持ち運ぶことにより，ライブラリの多様性にも柔軟に対応できる．

さらに，MA の状態を保持したまま移動できるという特徴を利用すれば，ライブラリを複数回実行した後に移動した場合でも，他の端末でその続きから実行を再開できる．

2.4. MA-based Middleware

上記の MA を利用したアプローチを実現するシステムとして，筆者らは MA-based Middleware を提案した[1][2]．提案システムはライブラリである MA と MA が動作するための Middleware Platform(以下，MWP と略す)の 2 つに大きく分けられる(図 1)．MA はライブラリとしてアプリケーションに対して特定のサービス(機能)を提供する．MWP は MA を制御するための基盤であり，アプリケーションが MA を透過的に利用するための機能を提供する．具体的には，MA 移動判断のための環境情報収集や移動の手続きを MA の代理で行ったり，アプリケーションからの MA 実行要求を移動先の MA に転送したりといった機能を提供する．

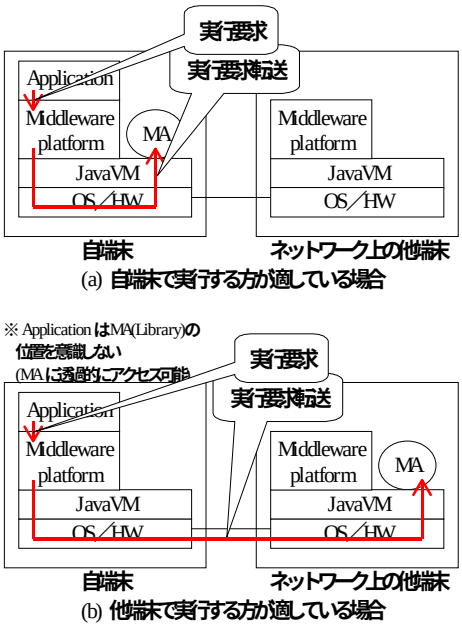

図 1 MA-based Middleware

3. 移動制御方式

3.1. 移動制御方式の構成

提案システムの中では，MA の移動先を決定するための環境情報収集のために，エージェント間のタスク割当てによく用いられる契約ネットプロトコル[3]を用いている．契約ネットプロトコルは "タスク告示", "入札", "落札"の 3 段階のステップで構成される．

図 2の例を用いて，MA の移動先を決定する流れを説明する．まず，MA はコーディネータに自身の情報(評価基準を含む)を登録し，他端末の情報収集を要求する(①)．この要求を受けて，コーディネータは情報収集の要求を他端末にマルチキャストする(②)．ここで，マルチキャストされる範囲は予め定められたエリアに限定される．各端末のコーディネータは自端末の状況を調査し(③)，その結果を要求元のコーディネータに返却する(④)．要求元のコーディネータは登録された"評価基準" (例えば, 予測される処理性能や通信コストなど)に基づいて最も適切な移動先端末を選択し(⑤)，その結果を選択した端末を含む全ての端末に報告する(⑥)．これにより, 適切な端末に MA が移動する(⑦)．

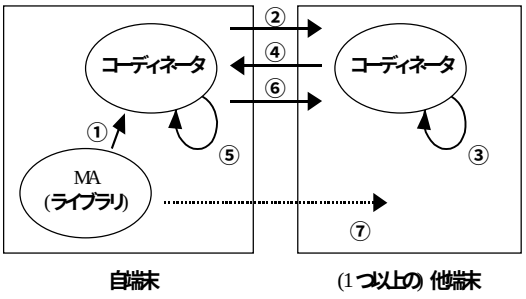

図 2 移動先決定のための交渉の流れ

表 1 評価式の項目と関連するパラメータ

評価式の項目	主なパラメータ
実行時間	・端末の最大性能 ・端末の負荷状態 ・MA のタスクサイズ
通信時間	・端末間の通信状態 ・端末の負荷状態 ・通信データサイズ
移動時間	・端末間の通信状態 ・MA のデータおよび コードサイズ

MA を，単に計算資源が豊富な他端末に移動させても，MA を利用するアプリケーション自体の処理効率が必ずしも向上するとは限らない．例えば，MA とアプリケーション間の通信処理や MA の移動処理などのオーバヘッドが発生するような場合には，他端末の計算資源を利用しても処理効率の向上が望めないことがある．そのため，これらのオーバヘッドを考慮して評価基準を定義する必要がある．アプリケーションの処理性能の向上を目的とする場合，例えば以下のような評価式 *fn* (*n* は端末を表す添え字)を用いる．

fn = MA の実行時間の推定値
　＋MA のデータ通信時間の推定値
　＋MA の移動時間の推定値　　(1)

評価式は(1)に示すように大きく 3つの項目で構成されている．各項目に影響を与える主なパラメータを表 1に示す．端末の最大性能は CPU やメモリといった物理的な計算資源によって決まる固定的なパラメータである．端末の負荷状態は，情報収集を行った時点での各端末内の負荷の大きさを表す．MA のタスクサイズは基本となる端末で無負荷の状態で実行した場合の実行時間で表現される．端末間の通信状態は端末が接続されているネットワークの状態である．通信データサイズは，MA の実行に必要となるデータの通信量である．MA のデータおよびコードサイズは，MA が移動する際に必要となる総データ転送量である．

4. MA-based Middleware の実装

4.1. 実装例

今回，筆者らは JavaVM 上で動作するマルチエージェントフレームワーク：Beegent[4]を用いて, 提案システムを実装した. 具体的には, PC 上の JVM には JDK 1.1.8，PDA 上の JVM には Jeode (Personal Java 1.2 準拠), Beegent は Ver. 1.0 を利用した．

Beegent は，分散システムを，アプリケーションとそれをエージェント化するエージェントラッパー，エージェントラッパーを利用する仲介エージェントの 3 種類の構成要素で構築できる．エージェント間の通信は，XML フォーマットのエージェント間通信言語 ACL によって行う．仲介エージェントは移動可能なエージェントである．Beegent フレームワークでは，複数の主体(エージェントラッパーと仲介エージェ

ント)が対話(メッセージ交換)による相互作用を行うことで，目的とする処理を実行する．この各主体の動作を定義するものがインタラクションプロトコルである．Beegent の基本となる構成要素を図 3に示す．

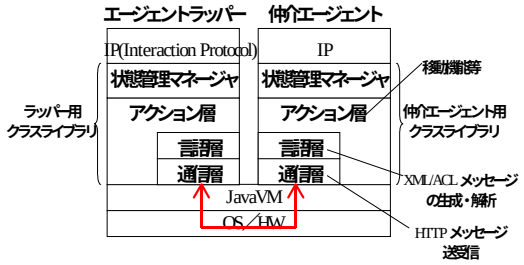

図 3 ラッパーと仲介エージェントのレイヤ構成

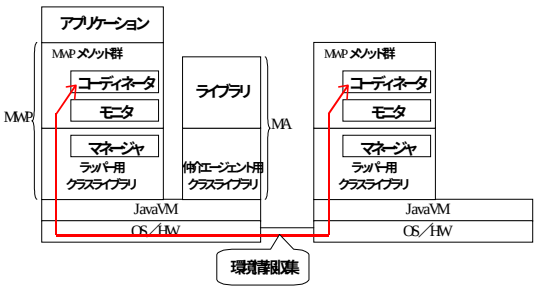

図 4 MA-based Middleware の構成

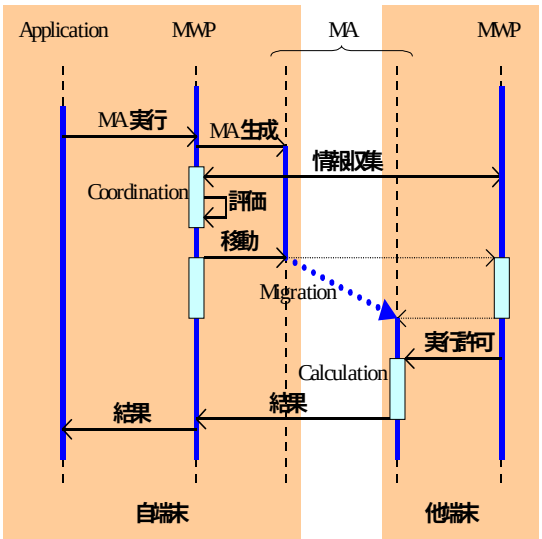

図 5 MA 実行のシーケンス

図 4に Beegent を利用して実装した提案システムの構成を示す．まず，アプリケーションに機能を提供するライブラリを移動可能な MA として構成するために，仲介エージェントのクラスを利用した．次に，MA の動作環境を提供するミドルウェアプラットフォーム(MWP)を構築するために，エージェントラッパーのクラスを用いた．アプリケーションは，MWP が提供するメソッドを利用することで，MA を透過的に実行することが可能となる．MA の移動制御を行うために MWP に必要な機能は，自端末の状態を監視するモニタ機能，他の端末の情報を収集して移動先を決定するコーディネータ機能，MA の移動や位置管理を行うマネージャ機能である．今回の実装では，自端末の状態(ノードの最大性能および負荷状態)をファイルに保存し，モニタ機能が状態を取得するように実装した．コーディネータ機能の通信処理は，Beegent が提供する XML/ACL のクラスライブラリを利用して実装した．マネージャ機能は，MA の移動や位置管理を行うが，これは Beegent の機能をそのまま利用した．

また，アプリケーションがライブラリの実行要求を MWP に送ってから，結果を受け取るまでの流れを図 5に示す．

4.2. 実験環境

ホームネットワーク等のローカルなネットワーク内に性能の異なる複数の端末が接続されている状態を想定する．具体的には，PDA，Note PC，デスクトップ PC が各 1 台ずつ，無線 LAN に接続している環境を用意した．図 6に今回の実験システムのネットワーク構成を，表 2に今回使用する各端末のスペックを示す．

	端末 1	端末 2	端末 3
種類	PDA	PC	Note PC
機種	東芝 GENIO e550GX	富士通 FMV 717G TX7	IBM ThinkPad i Series 1620
CPU	PXA250 400MHz	Pentium4 1.7GHz	Mobile PentiumIII 600MHz
メモリ	128 MB	1 GB	192 MB
OS	Pocket PC 2002	Windows 2000	Windows Me
JVM	Jeode 1.9.1	JDK 1.1.8	JDK 1.1.8

表 2 各端末のスペック

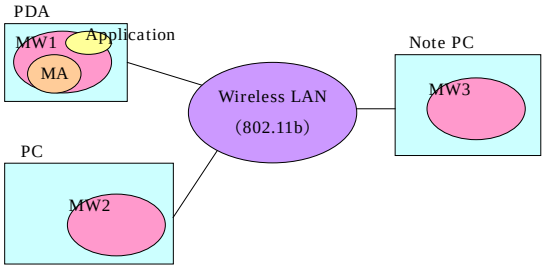

図 6 実験システムのネットワーク構成

今回は，ミドルウェア利用の一例として，MA が画像フィルタリング(Low Pass Filter：以下 LPF と略す)のライブラリとして動作し，PDA 上のアプリケーションがそのライブラリを利用して画像を変換するというシナリオを想定している．このようなシナリオにおいて，PC, NotePC の負荷状態が異なる場合の提案方式の動作について調べる．

4.3. 実験結果と考察

4.3.1. 基本性能

まず，MA を用いたアプローチの有効性を分析するために，3×3，5×5，7×7 マスクを用いた各 LPF を用いて PDA 上で処理を実行する場合と PC(または Note PC)上に LPF ライブラリを移動して実行する場合との処理時間を比較する．図 7，図 8，図 9は，LPF のマスクの種類ごとに各端末上で実行した場合の処理時間とその内訳を示したものである．すべての LPF において，PDA 上で実行した場合はライブラリの実行時間が大半を占めるのに対して，PC および Note PC 上で実行した場合はデータ通信時間が大きな割合を占め，実行時間は極めて短いことがわかる．また，計算アルゴリズムが複雑になるに従い実行時間は長くなる(入力画像は同じなのでデータ通信時間は一定)ため，MA を用いて処理時間を改善するという手法がより効果的に働く．

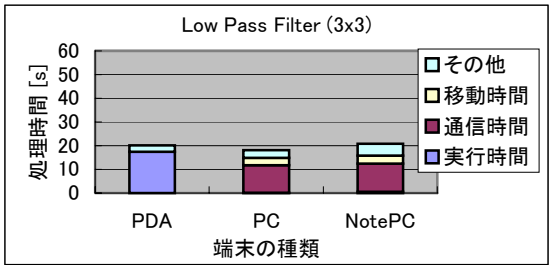

図 7 基本性能測定(3×3 マスク LPF)

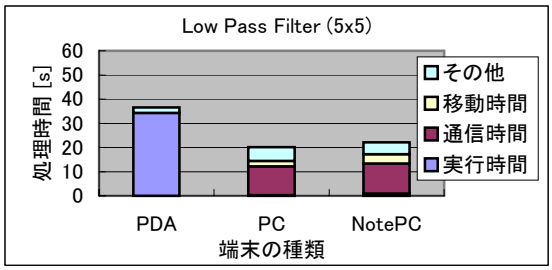

図 8 基本性能測定(5×5 マスク LPF)

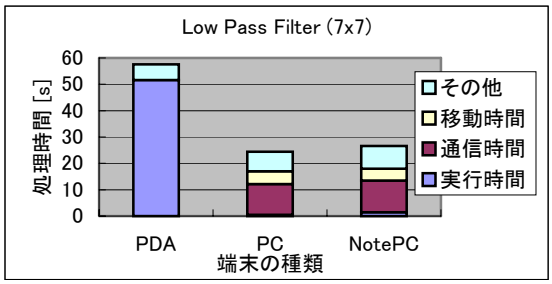

図 9 基本性能測定(7×7 マスク LPF)

4.3.2. 移動制御

移動制御の実験に用いた各種パラメータを表 3に示す．これらの値はすべて，実測値をもとにして設定した概算値である．例えば，端末の最大性能は，実際に MA を実行したときの実行時間を測定し，その逆数の比として設定し，端末ごとにファイルで持たせた．なお今回は，各端末の負荷状態と MA の実行時間との関係をわかりやすくするために，負荷(対象となる MA を含む)が逐次的に実行されるモデルを想定し，MWP にて負荷の逐次実行を制御した．

表 3 実験に用いた MA のパラメータの値

パラメータ	値
端末の最大性能	PDA：1 (基準) NotePC：37 PC：110
MA のタスクサイズ	34 (PDA 上で 34 秒)
MA の通信時間	19 秒
負荷のタスクサイズ	34 (PDA 上で 34 秒)
負荷の通信時間	1 秒
MA の移動時間	3 秒
データ通信量	256×256 個の integer 値

本稿では，評価式として以下の式を用いた．

fn = MA のタスクサイズ／端末の最大性能
＋MA のデータ通信時間
＋負荷のタスクサイズ／端末の最大性能
＋負荷のデータ通信時間
＋MA の移動時間　　(2)

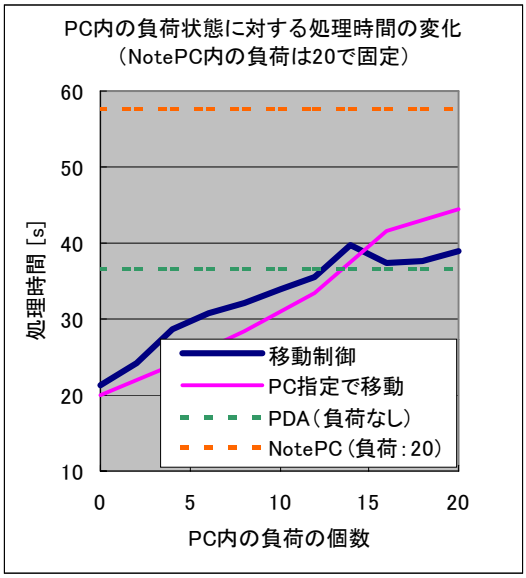

図 10 移動制御方式の動作(1)

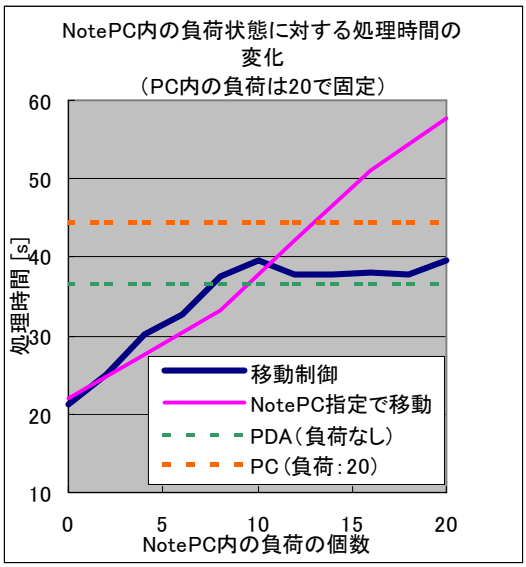

図 11 移動制御方式の動作(2)

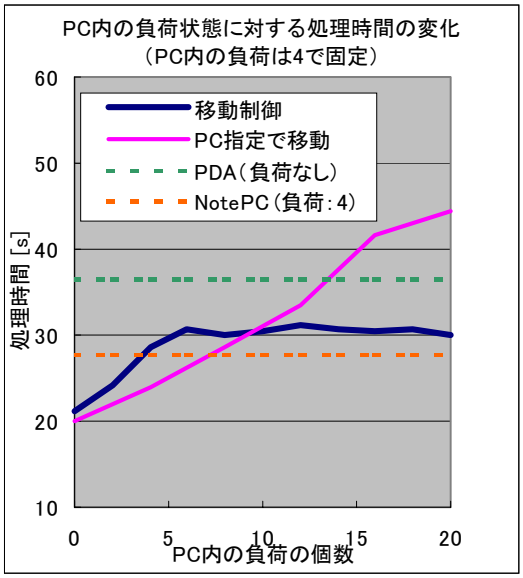

図 12 移動制御方式の動作(3)

各パラメータには表 3の値を用いる．5×5 マスクの LPF を用いた場合の動作を図 10, 図 11, 図 12に示す．図の横軸は PC 内の負荷を表し，縦軸はアプリケーションの処理時間を表す．MA を移動させずに PDA 内で実行する場合には，PC(または NotePC)内の負荷に無関係に処理時間は一定である．図 10は，NotePC の負荷を高い状態で一定とし，PC の負荷を変えて実行した結果である．PC に移動することをあらかじめ指定して実行する場合，PC 内の負荷が高くなるほど，アプリケーションの処理時間は遅くなり，ある状態以上では PDA 内で MA を実行する場合よりも処理時間が遅くなる．これに対して提案方式を用いた場合には，PC 内の負荷が低い状態では MA は PC に移動して実行されるが，PC 内の負荷が高い状態では移動せずに PDA 内で実行される．図 11は，逆に PC の負荷を高い状態で一定とし，NotePC の負荷を変えて実行した結果である．この場合も，移動制御方式は図 10の場合とほぼ同様の動作となる．図 12は，NotePC の負荷を低い状態で一定とし，PC の負荷を変えて実行した結果である．この場合は，PC 内の負荷が低ければ MA は PC 内に移動して実行されるが，負荷が高くなると MA は PDA よりも処理を速く実行できると推定される NotePC の方に移動して実行される．これらの結果より，提案する移動制御方式は，ネットワーク内の端末の状態から MA を処理するのに適した端末を決定するという動作を適切に行えてい

ることが分かる．

最後に，評価式とパラメータ設定について考察する．本稿では，評価式およびパラメータは実測値に基づいて設定した．現状では，本研究のアプローチを利用するためには，ライブラリの設計者がライブラリの性能を測定し，ライブラリの内部にその情報を定義することを前提としている．将来的には，ライブラリのコードから評価式を自動生成したり，ライブラリが繰り返し実行される間に統計情報を蓄積していく等の方法を取ることにより，ライブラリ設計者の負担を軽減し，移動制御の精度を向上することが可能であると考える．

5. まとめ

本稿では，筆者らが提案したモバイルエージェントを基盤とするミドルウェア(MA-based Middleware)の実装例について説明した．そして，画像フィルタリングを実行するライブラリを用いて，PDA 上のアプリケーションからローカルに実行する場合と，ライブラリを PC 上に移動してリモートで実行する場合について，処理時間を比較し，MA を用いて処理を PC 上に移動させるアプローチの有効性について示した．さらに，提案した移動制御方式が，移動先の候補である PC の負荷状態が低い場合は MA を移動して処理を実行し，高い場合は移動せず PDA 上で実行するという動作を示すことを確認した．これによって，提案システムがモバイル環境の課題に対する解決法を提供することを，実際のアプリケーション利用の観点から示すことができた．

謝　辞

日頃ご指導頂く当社情報通信研究所ワイヤレスシステム部 藤井輝也 部長に感謝いたします．

文　献

[1] 川上 憲治，広重 一仁，佐々木 宏，岡宅泰邦，本位田 真一，"モバイル環境向けエージェント移動制御，" 情報処理学会 MBL 研究報告，2002 年 3 月．

[2] 川上 憲治，広重 一仁，吉岡 信和，本位田 真一，"モバイル環境向けエージェント移動制御 (その 2)，" 情報処理学会 MBL 研究報告，2002 年 10 月．

[3] R. G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," IEEE Transactions on computers, vol.c-29, no.12, 1980.

[4] Bee-gent: コミュニケーションをエージェント化するマルチエージェントフレームワーク，http://www2.toshiba.co.jp/beegent/