マルチメディア通信と分散処理　101－1
コンピュータセキュリティ　12－1
（2001．2．20）

組み込み機器用モバイルエージェントシステムの設計と実装

重安哲也 † 浦上美佐子 †,†† 三浦賢司 ††† 松野浩嗣 †††
† 山口大学大学院理工学研究科 †† 大島商船高等専門学校 ††† 山口大学 理学部

近年、「組み込み機器」と呼ばれる家電製品や業務用端末が、徐々にネットワーク機能を備えるようになってきており、分散ネットワークが形成されつつある。一般に、現在の分散ネットワークシステムでは、リモートプロシージャコールや、モバイルコードを採用しており、サービス中は接続回線を切断できない（通信トラフィックの増加）等の問題点がある。
そこで、モバイルエージェントの概念が提案されている。モバイルエージェントは、自律分散的にネットワーク管理やサービス管理を行うことができ、通信コストを低く押えることができる。組み込み機器では、その数が膨大となるため、さらなるトラフィックの効率化が望まれる。そこで我々は、組み込み機器ネットワークにモバイルエージェント技術を導入することを提案する。また、実際に組み込み機器用のモバイルエージェントシステムの開発を行った。本稿では、組み込み機器ネットワークについての考察を行い、本システムの設計と実装について報告する。

Design and Implementation of Mobile Agent System for Embedded Processor

Tetsuya Shigeyasu † Misako Urakami†,†† Kenji Miura††† Hiroshi Matsuno†††
† Graduate School of Science and Engineering, Yamaguchi University
†† Ohsima National College of Maritime Technology
††† Faculty of Science, Yamaguchi University

Recently, many electric products equip embedded processors for linking with other equipment in houses or offices, and these embedded processors structure distributed network systems. In general, since such system uses remote procedure call mechanism, these system must always keep communication lines between other systems.
On the other hand, the concept of mobile agent system has become widely accepted in the field of computer networks. Mobile agent has an ability to reduce the cost of communication.
This paper proposes to apply mobile agent technique to embedded processors and describes about the design and implementation of the system.

1　はじめに

組み込み機器とは、家電製品などの制御用に軽量のOSをROMなどに組み込んだ製品をさす。近年、組み込み機器の高機能、高性能化が進み、ネットワークにこれらの機器を接続し、運用を行うことを目的にさまざまな研究開発が行われている。

組み込み機器分野に限らず、現在の分散システムの通信は、RPC(リモートプロシージャコール)を基本としている。RPCでは、ネットワークに接続されている各端末に対して処理を行う場合、常にネットワーク回線の確保が求められる。また、RPCベースの通信ネットワークでは、呼び出し中のプログラムは返値を受け取るまで処理が停止し、分散システムの持つ並列性を十分に生かすことができない。組み込み機器ネットワークに移植性や汎用性を持たせることを目的に提案されているJTRON(Java Technology on ITRON)[1]では、ネットワーク越しにプログラムモジュールをダウンロードして実行する、モバイルコードの技術を応用することを提案している。モバイルコードはRPCをベースにした技術で、ネットワーク越しに処理を行う場合でも、RPCにおけるプロシージャに相当するプログラムを転送して実行する。しかし、呼び出しの度にダウンロードが行われ、データ転送には既存のRPCが使用されるため、回線を切断することはできない。

一方、モバイルエージェント技術と呼ばれるものが提案されている[3]。モバイルエージェント技術は、プログラムモジュールと共に実行状態も転送することができる。またその最大の特徴として、ネットワークを自律的に移動することで、ネットワーク管理やサービス提供を自律分散的に行うことができる。我々は、PC上におけるRPCベースのシステムと、モバイルエージェントベースのシステムの比較実験において、既にモバイルエージェントシステムが、ネットワークトラフィックの削減に貢献することを実証した[4]。そこで、我々は組み込み機器ネットワークにもモバイルエージェント技術を導入することを提案する。本稿では、

組み込み機器のネットワークを考察し、モバイルエージェント技術を用いることの有効性を説明する。そして、今回実際に開発した、組み込み機器用モバイルエージェントシステム (TINIAgent) の設計と実装について述べ、今後の研究課題を報告する。

2 組み込み機器ネットワーク構築上の留意点

組み込み機器ネットワークを構築する場合には、現在の組み込み機器としての機能を損なってはならないということが要求される。そこで本稿では、以下の3つの項目に関して留意しながら組み込み機器ネットワークシステムを構築する。

1 高リアルタイム性

制御用システムであるため、リアルタイムに処理を行えなければならない。

2 ネットワーク協調性

既存のネットワーク資源を利用できなければならない

3 低コスト・コンパクト

ネットワーク機能の追加によるコストの上昇とサイズ増加は極力避けなければならない。

このような、組み込み機器ネットワークシステムにモバイルエージェント技術を導入することで、

a 移動先においてローカルに処理を実行

遠隔から処理を行うのではなく、ローカルに機器を制御できる (高リアルタイム性の維持)

b 機動性

ネットワーク上の全てのコンピュータに移動可能 (ネットワーク協調性の実現)

c TCP/IP を利用して実現

汎用の TCP/IP を通信プロトコルとして使用するので、既存のネットワーク資源を利用してネットワークを構築できる。(低コスト・コンパクト性の維持)

といった効果が期待できる。

3 組み込み機器用のモバイルエージェントシステムの設計

3.1 ネットワークモデル

組み込み機器ネットワークのモデルとして、既に提案されている JTRON の適用例 [2] を用いる。(図1) ネットワークに接続される構成要素は、ユーザからの要求を受け付けるインタフェース用端末と、そうでない個々の家電製品に搭載されている

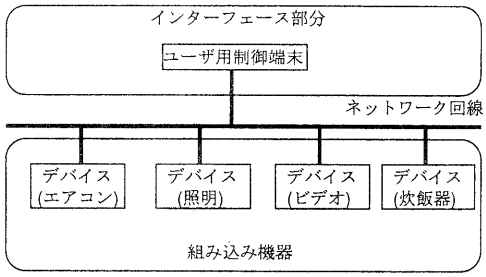

図 1: ネットワークモデル

組み込み機器の2種類に大別できる。インタフェース用端末は、ネットワークに最低1つは存在することになり、組み込み機器は制御するデバイスの数だけ存在することになる。

3.2 システムのプラットフォーム

組み込み機器のモバイルエージェントシステムの記述言語として、Java を用いる。Java は分散オブジェクトシステムにおいて、そのネットワーク関連機能の高さや非 OS 依存性が評価されている。

3.3 モバイルエージェントシステム

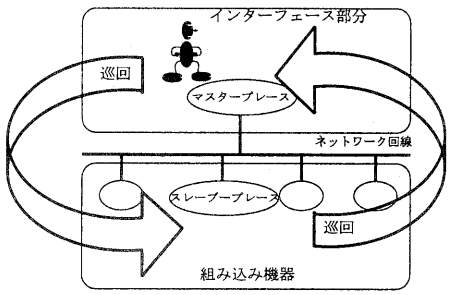

図 2: モバイルエージェントシステム

図1のネットワークモデルに対しモバイルエージェントシステムを導入する。(図2) モバイルエージェントの実行環境は各端末に1つずつ搭載する。本稿では、モバイルエージェントをはじめて提唱した Telescript[5] の定義に基づき、エージェントの実行環境をプレースと呼ぶことにする。そして、特にインタフェース用端末のプレースをマスタープレース、組み込み機器のプレースをスレーブプレースと呼ぶことにする。マスタープレースにおいて、ユーザからの要求を受けたエージェントは、その要求にしたがってスレーブプレースを巡回し処理を行う。このシステムは、インターネット回線に接続された場合も同様に考えることができる

が、この場合マスタープレースはインターネット上のどこにでも存在することができるものとする。

3.4 エージェントの転送方式

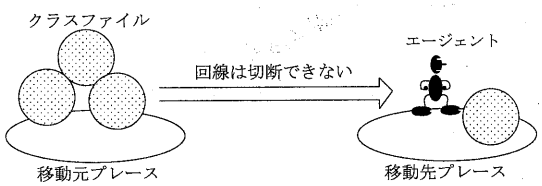

図 3: オンデマンド転送方式

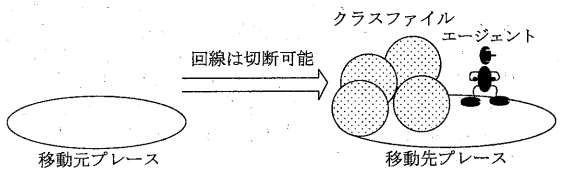

図 4: 一括転送方式

モバイルエージェントシステムにおいてエージェントとして転送されるべきものとして、エージェントの実行コードと内部状態の 2 つがある。しかし、Java 言語の仕様では、プログラムは実行されるまで、必要となるクラスファイルが特定できないと。そのためクラスファイルの転送方式は Aglets[8] に代表されるオンデマンド転送方式 (図 3) と、AgentSpace[3] に代表される一括転送方式 (図 4) の 2 種類が提案されている。その特徴は、

i オンデマンド転送方式

エージェントの実行に必要となるクラスファイルは必要に応じて転送する。

ii 一括転送方式

エージェントの実行に必要なクラスファイルは全て移動時に転送する。

である。回線を切断してエージェントを実行させる場合、オンデマンド転送方式では、実行が中断してしまう可能性がある。そこで本稿では AgentSpace の提案する一括転送方式を採用することにする。

3.5 クラスファイルのキャッシュ制御

PC 上にて実現されている既存のモバイルエージェントシステムはエージェントの実行コードであるクラスファイルをシステムにキャッシュするという方式をとる。こうすることで一度プレースを訪問したエージェントが再び訪れた場合、キャッシュされているクラスファイルを利用することにより、システムの高速化が実現できる。しかし、計算機資源の乏しい組込み機器においてはキャッシュ制御を行い貴重なメモリを消費してしまうこととなる。そこで本研究はキャッシュ制御を行わないシステムを設計することにした。

3.6 システムの設計

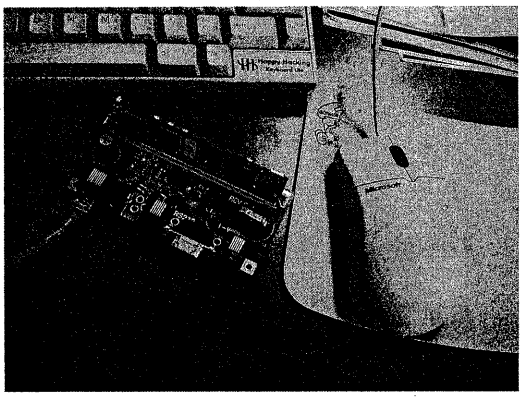

図 5: TINI

本研究では、Java 組み込み機器の TINI[6](図 5) をモデルとしてモバイルエージェントシステム (以下、TINIAgent) を設計する。TINI は SIMM サイズの非常にコンパクトなワンボードマイコンで、リアルタイム OS を搭載している。そして、1-wire デバイスと呼ばれる電子デバイスを接続することで測定機器の制御用としての使用もできる。また、Ethernet もサポートしており、インターネットへの接続も可能である。

3.6.1 プレース

TINIAgent は Java の実行環境である、Java Virtual Machine(以下、JVM) 上で動作し、エージェント実行を管理する。その概要は図 6 に示すとおりである。プレースは、その他のプレースへエージェントを転送することができ、転送されたエージェントは転送先のプレースにおいて、再び実行可能な状態にされる。プレースはエージェントの実行や送受信を管理するエージェントサーバと呼ばれる部分と、個々のエージェント情報についての管理を行うエージェントマネージャと呼ばれる部分の 2 つから構成されるが、本稿では特に区別せずにプレースと総称して説明する。

3.6.2 エージェントの移住

エージェントの移住は TINIAgent の機能の中でもっとも大きな機能である。その概要を図 7 に示す。本システムでは、エージェントが遠隔地に転

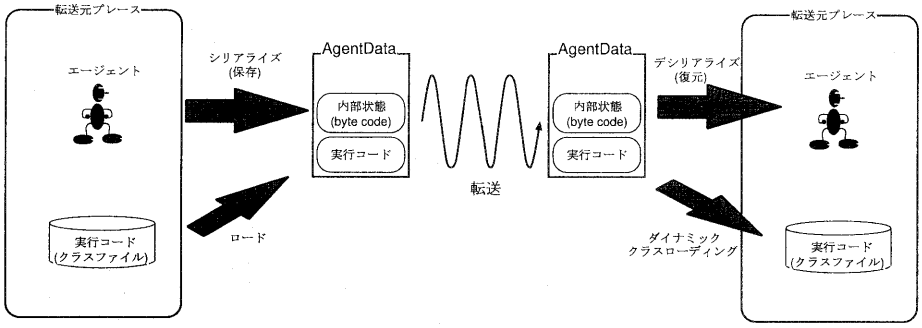

図 7: エージェントの移住

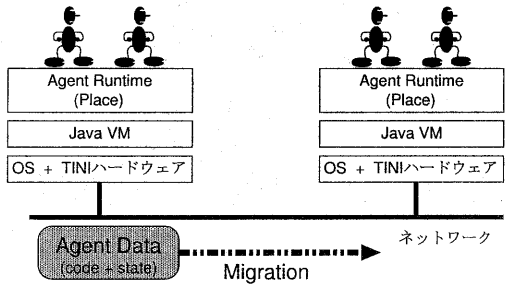

図 6: TINIAgent

送される場合、エージェントの内部状態とエージェントの実行コードの２つを転送する。本システムではこれら２つを１つのオブジェクトにして転送をおこなう。エージェントの転送に関わる流れを、以下に詳しく説明する。

エージェントの実行の中断　エージェントの実行の中断はJavaのシリアライゼーションを利用することにより実現する。まず、エージェントの転送要求が発生した場合、プレースはエージェントをバイトデータにシリアライズする。このときシリアライズされるのは、Javaの仕様によりエージェントの状態に限られる。その後、実行コードであるエージェントのクラスファイルを併せて、AgentDataと呼ぶオブジェクトとして転送を行う。

エージェントの実行再開　エージェントが移動を完了すると、再び実行再開するために内部状態が復元 (デシリアライズ) される。内部状態を復元するためにはその型枠となる実行コード (Java ではクラスファイル) を読み込まなくてはならない。TINIAgent ではそのために、Javaのダイナミッククラスローディングを利用する。JVM はダイナミッククラスローディングのために標準でクラスローダを実装している。しかし、このクラスローダは CLASSPATH と呼ばれる環境変数に定義された、ファイルシステム上のクラスしか参照することができない。しかし、エージェントの転送のたびにエージェントのクラスファイルを組み込み機器へファイルとして、追加するのでは、実行時間と、ハードウェアリソースの確保の両点からみて無駄である。そこで、TINIAgent ではメモリ上に存在するクラスファイルも参照し実行することができるように、標準で実装されているクラスローダを再設計する。

3.6.3　エージェント管理

エージェントはネットワーク上に複数存在することができる。これは、複数のエージェントが協調しながら効率的に処理を遂行できるようにするためである。実際にこれら複数のエージェントはそれぞれの位置するプレースにより管理される。プレースはエージェントの実行状態、実行コード、識別子（ID）等を把握し、エージェントに対しての絶対的な支配権をもつ。たとえば、エージェントが実行を継続することが不可能となった場合でも、プレースは強制的にそのフリーズ状態にあるエージェントを排除することが可能である。

3.7　エージェント間通信

エージェントがネットワーク上に複数存在する場合、互いに強調することで、より分散システムの並列性が増す。このため、エージェント間において、なんらかの通信方法が必要となる。TINIAgent では、エージェント間通信を行う場合、プレースを介して行う。

3.8　コマンドラインのユーザインタフェース

TINI に限らず組み込み機器は高度な入出力機器を持たない。したがって、搭載する JVM にも GUI を構成するための API は含まれない。そのため、我々は TINIAgent の操作にコマンドライン方式を採用する。(図 9 にその実行画面を示す。)

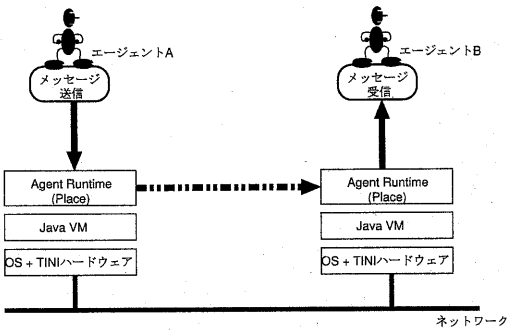

図 8: エージェント間通信

```
[sigeyasu@genome ~/TINIAgent]$ java Main
os name:Linux
os arch:i386
os  ver:2.2.14-1v16smp
os  classpath.:/home/sigeyasu/jdk1.2.2/jre/lib:/home/si
geyasu/TINIAgent/:/home/sigeyasu/system/
Version.......ver1.00a1
Scince.......'00.8.29
last modified.'00.10.17
set the SendPort = 5000,ReceivePort = 5001;
**** TINI Agent System Control Panel ****
version[ver1.00a1]
************ Command List ************
[n]ew agent                [a]gent status
[l]ist of agent            [m]essage send
[k]ill agent               [s]erver status
[q]uit server
```

図 9: システムのユーザインタフェース

3.9 セキュリティ機能

組み込み機器ネットワークに限らず、インターネット接続を考慮するシステムの場合、そのセキュリティ機能が問題となる。我々は、エージェントが組み込み機器に直接的にアクセスすることに対して制限を加える。具体的には、システムに悪影響を及ぼすようなAPIを使用できないようにするために、前述のエージェント用に再定義を行うクラスローダで、危険度の高いAPIのロードを禁止する。どうしても、必要なAPIはエージェントからプレースへと実行依頼を行うようにして、エージェントからの直接的な実行は避けるようにする。

3.10 軽量なシステムの構築

TINIはその仕様用途として、測定機器の制御用に使用されることが提案されている [7]。そのため、測定用に必要なハードウェアリソースをできるだけ多く確保するために、TINIAgentは必要最小限の軽量なシステムとしなければならない。そのため、今回開発したシステムはこれまで本節で述べた機能に的を絞ったものとする。

4 システムの実装

今回システムは、LinuxOSを搭載するPC上において、JDK1.2を用いて開発を行った。しかし、TINIに搭載されるJVMには以下の機能制限があるため搭載することができなかった。

TINI搭載のJVMの機能制限

- シリアライゼーションの未サポート
- ダイナミッククラスローディングの未サポート

しかし、TINI用の次期JVMリリースにより、上記2つの正式サポートが表明されている [9]。TINIAgentはその他の機能においてTINIで動作することを確認しており、次期JVMリリースがあればすぐにTINIAgentをTINI上に搭載することが可能である。以下は、PC上において実装を行った結果を示す。

4.1 システムの安定性

モバイルエージェントの安定性を確かめるために、我々は、本システムを搭載する2台のPC間を巡回するようにエージェントをプログラムし、実験を行ったが、24時間のあいだシステムは停止することなく安定して動作を行った。

4.2 関連システムとの比較

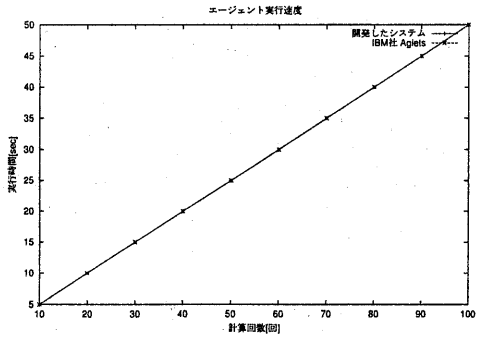

図 10: エージェント実行時間

次に関連システムとして、IBMのAglets[8]との実行時間の比較を行った。AgletsはPC上のJVMでのみ動作するモバイルエージェントである。比較は、エージェントの実行速度と、転送速度の2点について行った。結果については、図10,11に示すとおりである。実行速度は簡単なプログラムを繰り返して行わせる実験で、エージェントの転送は行っていない。この結果を見る限り、両システム共に実行速度の違いをみることはできない。

次に、エージェントの転送速度に関する実験である。エージェントは、2台のPC上を繰り返し移

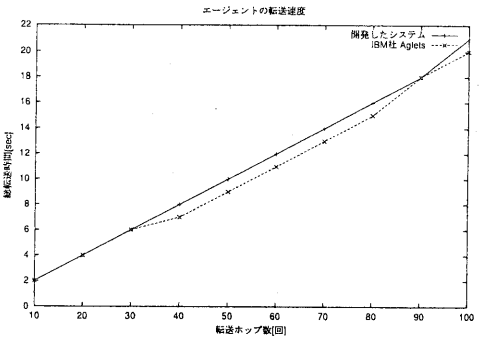

図 11: エージェント転送時間

動する。このときに、移動以外の処理は一切行っていない。この実験の結果をみると、本システムが Aglets に比べ僅かながら劣っている。これは、Aglets はもともと、PC 用に設計されているためクラスファイルのオンデマンド転送を行っているからだと考えられる。Aglets はエージェントの実行ファイルをシステム内のキャッシュに保存する。このため、エージェントが転送してきてもキャッシュにクラスファイルが存在する場合は、クラスファイルの転送は行われない。逆に TINIAgent では、前述のように組み込み機器へ搭載するという前提に設計されているため、ハードウェアリソースを減少してしまうキャッシュ制御は行なっていない。このため、常にクラスファイルの転送が行われる。我々は、クラスファイル転送分の時間が、結果として現れているものだと考える。また、本システムより転送速度のよい、Aglets のオンデマンド通信ではあるが、Java 言語の仕様によって、クラスファイルの転送要求がどこで起こるのかを予測することはできない。これは、回線切断の有効性を掲げて研究を行ってきた、我々のコンセプトから大きく外れるものになる。よって、転送速度が劣るからといって、本システムの組み込み機器ネットワークに対する妥当性は何ら問題ないものと考えられる。

4.3 TINIAgent の応用分野

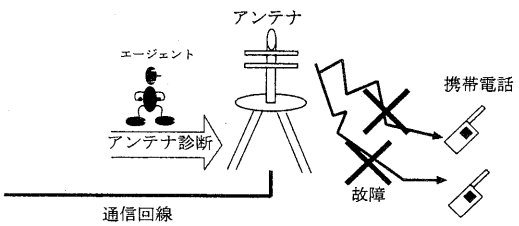

図 12: アンテナ診断エージェント

TINIAgent はその応用分野として、小型設備等への導入が予想される。例として、携帯電話のアンテナ設備を考える。日頃アンテナは端末間の通信を制御する。しかし、システムに不都合が起こった場合は原因を特定しなければならない。市街地にあれば、すぐに人間が診断に行くことも可能であるが、山中にあってすぐに直行できないといったことも予想できる。こうした場合、機器診断機能を持ったエージェントならば、アンテナをすぐに診断することができる。エージェントではなくても、設備自体に診断機能を備えることによって、機器を診断することは可能であるが、日頃は頻繁に使用しない機能まで常に機器に搭載してしまうと、機器のコストの上昇を招いてしまう。モバイルエージェントならば、提供するサービスを動的に入れ換える事が可能である。

5 今後の研究課題

以上、Java 組み込み機器へのモバイルエージェントシステムの導入の提案と実際のシステムの開発について述べた。本システムはその実験結果からシステムとしての安定性を確認ができた。今後は、TINI 用の次期 JVM リリースを待って TINIAgent を実際に TINI に搭載するとともに、TINI のネットワークを構成しての TINIAgent の評価を行う方針である。

参考文献

[1] 根本 忍, JTRON 仕様の実装とその現状, 情報処理, Vol.40,No.3, 情報処理学会, 1998.

[2] TRON プロジェクト, JTRON 仕様書, http://tron.um.u-tokyo.ac.jp/TRON/JTRON/

[3] 佐藤一郎, モバイルエージェントの動向, 人工知能学会誌, Vol.14, no.4, pp.598–605, 1999.

[4] 市村美佐子, 重安哲也, 鶴田祐紀, 松野浩嗣, WWW 情報検索システムのモバイルエージェントによる効率化, 平成 11 年度電気・情報関連学会中国支部連合大会講演論文集, pp310-311,1999

[5] James E.White 著, 山崎重一郎／津田宏 編訳, Telescript 言語入門, アスキー出版社, 1996.

[6] Dllas Semiconductor 社, Introducing TINI, http://www.ibutton.com/TINI/

[7] 渡辺伸吾, TINI AMEDES, http://www.tini.prug.or.jp/partech2000/

[8] Danny B.Lange,Mituru Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley 出版, 1998.

[9] Mailing list ARChives at PCC http://marc.theaimsgroup.com/