モバイルエージェントを用いたセンサネットワーク向けフレームワーク

梅澤 猛[†] 佐藤一郎[††] 安西祐一郎[†††]

本稿ではセンサネットワークの管理・運用のためのミドルウェアを提案する．このミドルウェアの特徴はセンシング対象となる環境の変化やアプリケーション要求に応じて，その構成や機能を変更できることにある．具体的には，センサ制御，データ変換，データ選別，データ統合，センサノード間通信などのセンサネットワーク管理・運用における基本機能からなる階層構造を持つミドルウェアであり，さらに各階層をモバイルエージェントをベースとするコンポーネントとして実現する．この結果，モバイルエージェントのコンピュータ間移動性を通して，ミドルウェアの機能を変更・拡張することが可能となり，その際に各コンポーネントに対して統一されたインタフェースを規定することにより，他の階層への影響を最小化することもできる．本稿では提案するミドルウェアのアーキテクチャを示すとともに，その実装に関して説明する．また，2つの応用事例を通して，提案するミドルウェアの有用性と柔軟性を議論する．

A Mobile Agent-based Framework for Configurable Sensor Networks

Takeshi Umezawa,[†] Ichiro Satoh[††] and Yuichiro Anzai[†††]

In this paper, we present a novel framework for self-configurable sensor networks. The framework enables sensor networks to be dynamically reconfigured to suit the requirements of applications and changes in the environments that is the object of their sensors. The framework provides a middleware system with an architecture that is structured as a collection of components organized in a hierarchy of five layers. Since these components are implemented as mobile agents and offer common interfaces in their layers, we can deploy the components at remote sensor nodes and dynamically replace them by other components designed for the same layer, without affecting the rest of the nodes. Therefore, the framework provides a powerful approach to the easy development of adaptive and application-specific software for sensor nodes. This paper describes the architecture of the framework and the implementation of its prototype, currently using Java as the implementation language along with a component description language. Two interesting applications are outlined to demonstrate the utility and flexibility of this framework.

1. はじめに

ユビキタスコンピューティング環境では，各種機器に計算と通信の能力が付加される．その中でも，センサネットワークは既存センサの持つ問題を解決する点でも重要である．一般に個々のセンサによる測定には限界があるが，多数のセンサから得られる測定値を統合して多くの単純なデータを集めることで複雑な結果を生み出すことができる．今後，室内外を問わず無数のセンサがネットワーク化されると思われるが，そのときセンサネットワークは1つの社会的インフラストラクチャとして位置づけられることになり，多くの人々が様々な目的に利用できることが必須の課題となる．

しかし，既存のセンサネットワークの多くは特定の用途を想定して設計されているため，それ以外の用途に利用することが難しい．これはネットワークを構成する各センサノードの処理能力，記憶容量，消費電力に制限があるため，想定された用途以外に対応するプログラムを保持・処理することができないことに起因する．また，センサノードの増減によるネットワーク特性の変化に加えて，センサネットワークを利用するアプリケーションの要求変化のために，センサノード

† 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†† 国立情報学研究所/科学技術振興事業団
National Institute of Informatics / Japan Science and Technology Corporation
††† 慶應義塾大学
Keio University

が提供するべき機能も事前に予測することはできない．たとえば，ビル内におけるユーザの位置検出では，所定の部屋やフロアにいるユーザを常時監視する場合と，特定のユーザがどこにいるかを発見する場合ではセンサネットワークに要求される機能は異なる．また，センサネットワークに新しいセンサノードを接続したときでも，新しいノードと既存のノードが扱える通信プロトコルやデータ単位などが異なるとデータの相互交換が困難になる．

そこで，本稿ではセンサネットワークを汎用的に利用する方法を提案する．これは各センサノードを制御・監視するソフトウェアをプラグインコンポーネントとして構成することで，センサネットワークの動的変更を可能にするミドルウェアを実現するものである．ただし，センサネットワークは集中制御を行うホストの存在を必ずしも仮定できず，また各ノードがネットワークの全域情報を保持できるとは限らない．さらに，センサノードの追加・削除が発生するため，分散化された管理方法と各種変化に柔軟に対応する能力が必要となる．そこで，コンポーネントをモバイルエージェントとして実現することで，センサノードの制御や通信プロトコルの処理を行うソフトウェアのセンサノードに対する追加・削除をエージェントの移動によって行う．こうした機能拡張ではネットワークやデバイスに関連した多くの属性値の設定なども必要となるが，モバイルエージェントはその移動時にプログラムコードと実行状態の両方を転送できることから，設定済みのソフトウェアを配布する手法として優れている．さらに，このミドルウェアではコンポーネントを階層的に構成することにより，コンポーネントの交換による変更を容易化する．

以下，本稿では，続く 2 章で関連研究と提案するミドルウェアの基本アイデアについて説明し，3 章ではミドルウェアのアーキテクチャについて解説する．4 章で実装について示し，5 章では 2 つの応用事例に基づいて提案するミドルウェアの有効性について議論する．6 章で結論を示す．

2. 方　　針

本稿で対象とするセンサネットワークは，住宅やオフィス，街などの空間に分散配置された各種センサ（たとえば人または物体の位置，方向，温度，明暗，騒音などの検出）をネットワークを通して相互接続し，その検出データを組み合わせることにより，測定精度の向上や環境情報の取得を実現するとともに，状況に応じた人間支援や環境監視を目的とする．その構成は

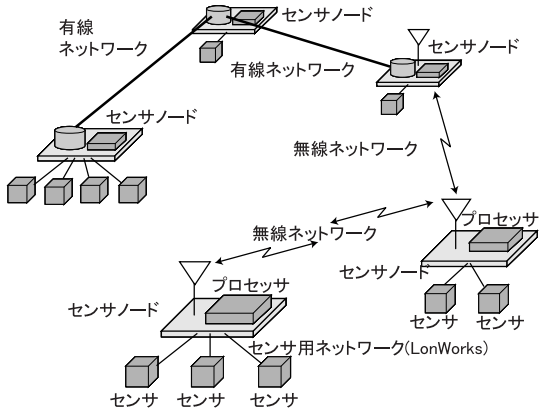

図 1 センサネットワークの例
Fig. 1 A typical sensor network.

図 1 に示すように，計算能力と通信能力を持つネットワークノード（センサノードと呼ぶ）が 1 つ以上のセンサを制御・監視する形態をとる．そして，各センサノードは有線または無線通信ネットワークを通して Peer-to-Peer 方式により接続される．たとえば，室内の床に埋め込まれた圧力センサや壁に備え付けられた赤外線による動体センサを複数のセンサノードで統合し，各ノードの検出情報を集約することにより人または物体の位置や移動方向を検出することができる．

2.1 要 求 事 項

各センサノードは多様なセンサを扱うとともに，他のセンサノードとデータを円滑に交換する必要がある．これらをアプリケーションに対して代行する方法として，センサネットワークを管理・運用するためのミドルウェアが必要となる．なお，このミドルウェアはセンサネットワークの特性から次のような要求を満足する必要がある．

汎用性： 前述のように各センサノードが持つ計算能力や記憶容量などへの制限が多い一方で，センサネットワークはあらかじめ想定した用途だけでなく，開発時に未知な用途を含む多様な用途に利用できることが求められる．このため，各センサノードが必要なソフトウェアを必要なときだけ保持・実行することが求められる．

適応性： 測定対象やアプリケーションの要求が変化した際には，該当するセンサノードが提供する測定方法やデータ選別などの機能を動的に変更することが求められる．

スケラビリティ： ノード数増加によるセンサノードやネットワークへの影響を最小化するため，非集中制御が求められる．センサが収集するデータは多量となる傾向があるが，その一方で各センサノー

ドは無線通信などで接続されることが多く，その通信帯域は広いとはいえない．したがって，必要なデータだけを転送する機構が求められる．

独立性：センサの種類は多様であり，そのデータ形式も多様である．センサの制御方法においてもセンサ自身がデータを常時または変化時に出力するもの，外部からの問合せに応じてデータを出力するものなどがある．ミドルウェアはこうした様々なセンサを扱える必要がある．また，OS やハードウェアに独立であることが望まれる．

2.2 関連研究

本稿で提案するミドルウェアを説明する前に既存研究との比較を述べる．

近年，センサネットワークを対象としたアルゴリズムやミドルウェアがいくつか研究されており，センサネットワークを非集中制御で管理し，そのトポロジが変化するネットワークを対象としている．しかし，既存研究の多くはセンサノード間の経路制御やネットワーク管理に関するものである．たとえば，MIT による Directed Diffusion[5] は無線通信で接続されたセンサノード間転送経路の発見機構を実現する．また，ETH による Smart Context-Aware Packets（sCAP）[11] はデータ転送用のパケットにそのパケットが通過したノードの識別子を保持することにより，効率的なデータ転送を実現する．ただし，これらはネットワーク構成の変化を通信方法に反映することができるが，前述の要求事項として議論したような測定対象やアプリケーション要求の変化に対する適応性は提供していない．一方，本稿で示すミドルウェアは個々の通信制御を前提にするものではなく，センサの制御や検出データの選別，ノード間通信などを動的に変更する枠組みを提案するものである．

なお，本稿と同様にモバイルエージェントをセンサネットワークへ応用した事例もいくつか存在するが，それらは通信トラフィックの削減を目的としている．一般にセンサによって測定されるデータ量は大きくなるが，その一方でセンサネットワークは無線通信が前提となったり消費電力の制約を受けたりすることなどから，通信帯域に対する制約が多い．これを解消する方法として，アプリケーション要求に合致する測定データのフィルタリング用プログラムコードをモバイルエージェントにより各ノードへ運び，そのコードを実行することにより必要なデータだけをノード間で交換・選別する方法が知られており，たとえば Qi らのセンサネットワーク[12] ではモバイルエージェントによる通信トラフィックの低減および測定データの合成手法を提案している．一方，本稿で示すミドルウェアは既存手法と同様にモバイルエージェントを通してセンサネットワークのスケラビリティを向上させると同時に，モバイルエージェントを移動・置換可能なコンポーネントとして導入することにより，ミドルウェアの汎用性や適応性，独立性を高めることが目的となる．

なお，著者の 1 人は階層モバイルエージェントによるコンポーネント手法[10] を提案しているが，これはモバイルエージェントによる大規模アプリケーションの開発とエージェント転送プロトコルの動的変更を目的としたものであり，センサネットワークを想定したものではない．

2.3 動的拡張性を持つミドルウェア

本稿で提案するミドルウェアの基本構成および機能について概説する．前述のセンサネットワーク向けミドルウェアに対する要求事項を満足するため，各センサノード上で必要とされる各種機能をモバイルエージェントによるプラグインコンポーネントとして実現する．ここで，モバイルエージェント[4),8)] はコンピュータ間を移動できる自律的なプログラムであり，その移動に際してはプログラムコードだけでなく，変数などの実行状態もデータ化して転送できるため，移動先では移動直前の実行状態から処理を継続できる．この結果，従来のプログラムコードを主体としたコンポーネントと比較して，コンポーネントに対する各種設定を行った状態での遠隔コンピュータへの配布に適している．これは，センサや通信制御などにおいて多くの設定が必要となるセンサネットワークでは有用な特徴となる．また，このコンポーネントはモバイルエージェントの移動性や自律性を継承することから，コンポーネント自身が配布先のセンサノードを発見し，そのノードにそれ自身または複製を移動させることで，遠隔センサノードへのコンポーネントの配布・稼働を実現できる．

コンポーネント拡張・置換では依存関係によるミドルウェア全体への影響を考慮する必要がある．ところで，センサフュージョンの処理は，(1) センサを制御・監視し，その測定データに対する (2) 変換と (3) 選別を行い，各センサからの選別済みデータを (4) フュージョン（統合）するという手順で実現されることが多い[3)]．そこで，提案するミドルウェアではこれらの 4 つの処理に加えて，統合済みデータを遠隔ノードに転送する (5) 通信機能を含む 5 つの機能に分けて構成する．さらに，これらの機能は測定環境やアプリケーション，ネットワークの変化に応じて変更が要求されることから，その機能をモバイルエージェントによるプラグインコンポーネントとして実現し，拡張・置換

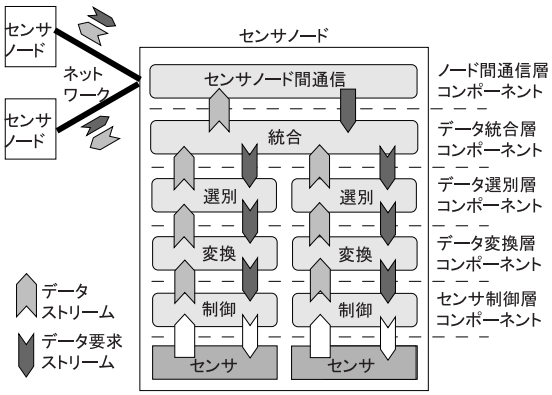

図 2 コンポーネントアーキテクチャ
Fig. 2 Component architecture.

を可能とする．なお，これらの 5 つの機能はデータ処理の流れ（つまり各機能の番号順）に応じた依存関係を持つことから，それに対応した**図 2** ような階層モデルとしてコンポーネントを構成する．さらに階層間インタフェースを定めることにより，コンポーネントの拡張・置換時の影響を最小化・明確化できるようにする．

- **センサ制御層**のコンポーネントは，ノードに接続されているセンサの制御・監視を行うものであり，センサに対して所定の間隔でデータ要求を行い，計測データを収集する．
- **データ変換層**のコンポーネントは，センサ制御層のコンポーネントから受け取ったセンサ依存表現データをアプリケーションによる処理に適した表現データ形式に変換する．データの正規化を行うことで，これより上の階層ではセンサの相違によるデータ表現の違いを意識せずに処理が可能となる．
- **データ選別層**のコンポーネントは，特定の閾値を超えるセンサデータや，一定期間内に大きな変動があったデータのように，ある与えられた条件に合致するもののみを選び，選別済みデータとして出力する．
- **データ統合層**のコンポーネントは，ノードに接続された複数のセンサから集められたデータをフュージョン（統合）することで，データの精度向上や，相違な測定対象による複合データを生成し，統合済みデータとして出力する．
- **ノード間通信層**のコンポーネントは，ノード間の通信プロトコルを定義し，他のノードとのデータ交換や，特定のホストへのデータ送信を行う．なお，このコンポーネント自身がその移動性を利用してデータを収集・移送することも可能である．

たとえば，センサノード間の通信プロトコルを変更するときは通信対象となるノード上のノード間通信層のコンポーネントを，またセンサの出力する単位系を変えるときはデータ変換層のコンポーネントを，そして収集すべきデータを変えるときはデータ選別層のコンポーネントを入れ替える．なお，この入れ替えは特定サーバまたは他のセンサノードから必要なコンポーネントを複製・移動させることにより実現する．また，各層は他の層に非依存な単一の機能を提供するとともに，階層間には統一されたインタフェースを提供することから，各階層のコンポーネントを置換・拡張した際のの影響はその層または依存関係がある下位層に限定される．

3. 設　　計

本稿で提案するミドルウェアは各センサノードに用意され，コアシステムとコンポーネント，そしてコンポーネントの実行・移動を実現するモバイルエージェントのランタイムシステムの 3 つの部分からなる．**図 3** にミドルウェアの基本構成を示す．

3.1 モバイルエージェント

前述のように，このミドルウェアではセンサネットワークを制御・管理する主要機能を Java 言語で実装されたモバイルエージェントによるコンポーネントとして実現する．モバイルエージェントは独立した計算実体として導入され，モバイルエージェントの合成や置換などの機能は提供しない．これらの機能をコアシステムおよびフレームワークライブラリとして実現することにより，多様なモバイルエージェントシステムを拡張することなく利用できるようにする．なお，センサネットワークの制御・監視は移動先コンピュータのリソースを利用することから，エージェントの移動前や終了前には獲得したリソースの解放，また生成先や移動先では必要なリソースの獲得などの処理を明示的に行う必要がある．したがって，このミドルウェアから利用可能なモバイルエージェントシステムは，セキュリティ機構が許す限りリソースを利用できることと，エージェントの実行モードが遷移した前後に明示的にエージェントにそれを通知する機能が必要である．また，後述するようにエージェントの移動手法は弱移動性を前提とする．ただし，これらは既存の多くの Java 言語モバイルエージェントシステムの利用を制限するものでない．

3.2 コアシステム

コアシステムは各階層間にインタフェースを規定し，

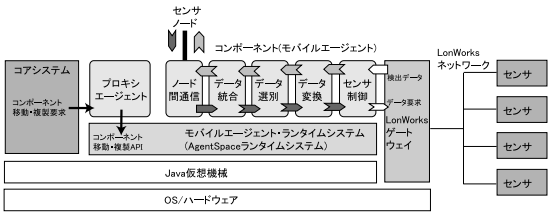

図 3 ミドルウェアの構成
Fig. 3 Structure of middleware.

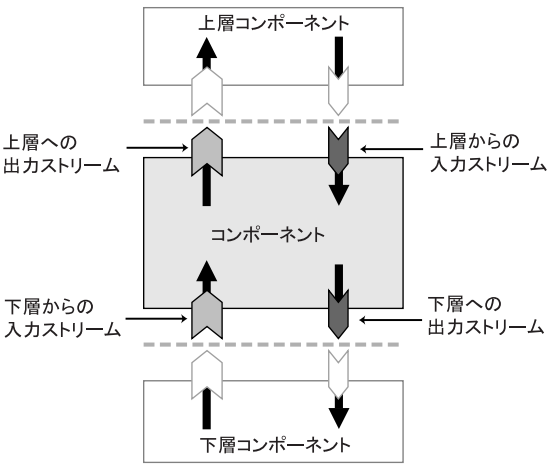

図 4 コンポーネント間インタフェース
Fig. 4 Interfaces between components.

測定環境やアプリケーション要求の変化に応じて必要なコンポーネントの配布・置換を制御する．

階層コンポーネントのインタフェース

このミドルウェアではコンポーネントを**図 4** のように階層的に構成する．この階層間インタフェースは隣接する階層のコンポーネントに対する 1 つ以上の入出力ストリームとして与えられる．上層から下層へのインタフェースは制御命令の転送用ストリームとなり，下層から上層へのインタフェースはセンサによって検出されたデータの転送用ストリームとなる．各ストリームは Java 言語の I/O Stream を利用したものであり，その接続・切断はコアシステムにより制御される．

階層コンポーネントの配布

コンポーネント配布には 3 つの方法がある．1 つ目はあらかじめコンポーネントを配布しておく方法である．2 つ目はコンポーネントを実現するモバイルエージェントの自律性を利用して，コンポーネント自身が移動・複製を行うことにより，遠隔センサノードにそれ自身を配布する方法である．3 つ目は測定環境やアプリケーション要求の変化に応じて必要なコンポーネントを配布・取得する方法である．2 つ目の方法は 5 章で概説することから，以下では 3 つ目の配布方法について具体的な手順を示す．

(1) センサノードが測定環境やアプリケーション要求，自身のシステム構成の変化を検出したとき，隣接するノードまたは所定のサーバにメッセージを送信する．このときのメッセージには，その変化の検出に必要なコンポーネントすべての識別子を含んでいる．
(2) コンポーネントの要求メッセージを受け取ったノードまたはサーバはそのコンポーネントを保持していれば複製を作って送り返す．保持していない場合はその要求メッセージを他のサーバまたはノードに転送する．なお，メッセージの増加を避けるため転送回数は制限可能である．

以上により，あらかじめ想定していなかった機能であっても，コンポーネントとして必要に応じて導入することができる．また，各コンポーネントは TTL（Time-To-Live）フィールドを持ち，有効期間を明示的に制限することもできる．したがって，所定の時間内に利用されなかったコンポーネントは自動的に削除または所定のサーバに帰還させることもできる．

階層コンポーネントの置換

このミドルウェアの目的の 1 つは動的にセンサノードの機能を変更することにある．各ノード内のコンポーネント置換手順を示す．

(1) センサノードに新しいコンポーネントとなるエージェントが到着する．
(2) ミドルウェアはそのコンポーネントの入るべき階層を特定し，該当する階層の既存コンポーネントにおける上層および下層とのストリームを切断する．
(3) 新しいコンポーネントが同じ階層の既存コンポーネントと排他的な関係になるときは既存コンポーネントを取り除く．
(4) ミドルウェアは新しいコンポーネントの入出力ストリームを上層および下層のコンポーネントと接続する．

なお，現在の実装ではコンポーネント置換過程に検出されたデータの補完は行っていない．ストリーム内部にバッファリングし，接続後に補完することも可能であるが，センサによる検出データは即時性が要求されることに加えて，一般にセンサネットワークではセンサの検出失敗やエラーが頻繁に起きるため，それを利用するアプリケーションは検出ミスを前提に構成されていることが多いこと，本稿の対象は室内外に設置されたセンサによる人間支援にあることから，データ

欠損に寛容であることが多いことなどからデータの補完は行っていない．また，取り除かれたコンポーネントは将来の利用に備えてコアシステム内のキャッシュに所定時間だけ保持することもできる．

3.3 コンポーネント階層

提案するミドルウェアでは前述のようにセンサノードの機能をセンサ制御，データ変換，データ選別，データ統合，ノード間通信の5つの機能に分割する．それぞれの階層コンポーネントが提供する機能を以下に説明する．

センサ制御

数多くのセンサが存在するため，それらの制御方法は多種多様となる．この層ではセンサを含めたハードウェアと低レベルソフトウェアに関する処理を行い，こうしたセンサによる相違を吸収する．たとえば，センサにはデータを受動的に取り出すものと，能動的に出力するものがある．前者のセンサには問合せコマンドを所定間隔や上層から要求に応じて送信して測定データを取得する．一方，後者のセンサはセンサ自身が所定間隔やデータ変化時にデータを送信してくることから，各センサにバッファを設けて最新のデータを格納することで，上層からのデータ取得要求時にただちに結果を返せるようにする．

データ変換

センサはその制御方法だけでなく，測定されるデータの表現方法も相違する．たとえば温度センサではその出力がセ氏，華氏，絶対温度など様々な単位系がある．また，実際の測定値を定数倍することで小数値を整数値として出力するセンサも多い．たとえば，“298”という測定値を得た場合，これは “29.8°C” を示すこともあれば，“298K（=25°C）” を示すこともありうる．この層では，精度や単位などについてデータ表現の変換を行う．この層のコンポーネントは，下層のセンサ制御層のコンポーネントから1本の入力ストリームで結ばれ，測定データを受け取ると自動的にデータ表現を変換し，その変換結果を上層のコンポーネントに出力ストリームを介して送る．なお，上層からの制御命令はそのまま下層に転送する．コンポーネントはモバイルエージェントとして実装されているので，各センサノードはアプリケーションや他のノードの要求に応じて，このデータ変換用コンポーネントを取得・置換することができる．

データ選別

センサノードを利用するアプリケーションはセンサの測定したデータのすべてを必要とはしないことが多い．必要なデータだけを選別することにより，ノード間で交換されるデータ量を減らすことができる．ただし，選別方法はアプリケーション要求や環境の変化によって変わる．したがって，選別を行うコンポーネントを用意し，下層のコンポーネントから受け取ったデータから必要なものだけを上層のコンポーネントに転送させる．なお，他層と同様にコンポーネントはモバイルエージェントとなることから，そのノード間移動を通じて配布・置換を行うことができる．また，データ選択ポリシは入力ストリームから出力ストリームへのフィルタプログラムとして定義され，コンポーネントは下層のコンポーネントに対して所定の間隔でデータを要求することもできる．

データ統合

1つのノードに複数のセンサが接続されている場合，それぞれのセンサから得られたデータを統合することで，データの精度向上や条件の絞り込みなどが可能である．この層におけるコンポーネントは上層に対する出力ストリームは1つであるが，1つ以上の下層コンポーネントと接続するための複数の入力ストリームとそこから得たデータを統合するためのプログラムを持つ．また，このデータ統合用コンポーネントは後述のノード間通信用コンポーネントに入力ストリームをつなぐことができ，その場合，他のセンサによる測定データの統合にも利用できる．

ノード間通信

センサネットワークでは，多数のセンサノードから得たデータを特定のホストへ集めて処理を行うアプリケーションが多くみられる．また，他のノードからデータを得ることで，データの補完や統合を行うこともある．この層では，これらのホストやノード間でデータ交換を行う機能を提供する．各コンポーネントは，下層のコンポーネントに接続された入力ストリームを持ち，これを介して遠隔ノードとデータ交換を行う．AODV[7]やDSR[2]のように，無線センサネットワーク向けに多くのルーティングやフォワーディングのアルゴリズムが提案されており，他ノードとの通信プロトコルとして，各コンポーネントにこれらを定義することでシステムの柔軟性を高めることができる．なお，モバイルエージェントの移動性や自律性を利用することにより，コンポーネント自身がデータを他のセンサノードに移送することもできる．この場合，プログラム自身による経路制御ができるため，柔軟な経路制御が可能となる．

4. 実　　装

本稿で提案するミドルウェアはJava言語[1]のJava

```
public abstract class Agent
    implements Serializable {
  // エージェント生成後のコールバックメソッド
  void create();
  // エージェント消滅時のコールバックメソッド
  void destroy();
  // エージェント移動時のコールバックメソッド
  void leave(URL dst);
  // エージェント到着時のコールバックメソッド
  void arrive();
  // エージェント複製後のコールバックメソッド
  void clone();
  ....
  // 自分自身を移動
  void go(URL dst) throws
    NoSuchLocationException ... { ...}
  // 自分自身を複製
  URL duplicate(URL dst) throws
    NoSuchLocationException ... {...}
  // ライフタイムを設定する
  void setTimeout(int time);
  ....
}
```

図 5 リスト 1: Agent クラス
Fig. 5 List 1: Agent class.

仮想マシン（JDK1.1 以上，Personal Java を含む）上で稼働する．以下ではそのモバイルエージェントシステムとして AgentSpace[9] を利用している場合について概説する．なお，Aglets[6] などの他の Java 言語によるモバイルエージェントシステムを利用しても容易に実現できると考えられる．

4.1 モバイルエージェント

前述のようにコンポーネントはモバイルエージェントとして実装され，その実行や移動はモバイルエージェントのランタイムシステムによって実現される．各エージェントは Java オブジェクトであり，Agent クラスのサブクラスとして定義される（**図 5**）．

ここでメソッド create()，destroy()，leave(URL dst)，arrive()，clone() はコンポーネントの生成直後，終了直前，移動直前，移動直後，複製直後にモバイルエージェントのランタイムシステムにより呼び出されるモバイルエージェント側のコールバックメソッドである．エージェントはこれらのメソッドにそれぞれの実行モード遷移に必要な処理を定義することになる．ところで，コアシステムはモバイルエージェントとして実現されたコンポーネントの移動や複製を行う際に，モバイルエージェントのランタイムシステムを制御する必要がある．ただし，ランタイムシステムへの拡張を回避するため，コアシステムの代理となるエージェントを用意し，コアシステム制御要求を受け取ると対応したランタイムシステム上の対応する API を呼び出すこととした．

なお，モバイルエージェントの移動では，移動対象となるエージェントプログラムの範囲に応じて強移動性と弱移動性[4),8)] に分かれるが，本稿のミドルウェアではコンポーネントは弱移動によって移動してよいとする．この結果，移動時にインスタンス変数は保存されるが，ローカル変数やプログラムカウンタは対象外となる．ただし，センサネットワークでは各コンポーネントはセンサを含む多様な計算資源を獲得・解放する必要があり，仮に強移動性を用いたとしても，移動透過の実現は困難となることから，弱移動による制限は少ない．また，Java 言語による多くのモバイルエージェントシステムは弱移動に基づいていることから，結果として多くのモバイルエージェントシステムが利用できることになる．エージェントの認証やセンサノードの保護はモバイルエージェントシステムおよび Java のセキュリティ機構を適用している．

4.2 コンポーネント

各階層のコンポーネントは Agent クラスのサブクラスとなり，ここではデータ変換層とデータ選別層コンポーネントの定義を宣言する AbstractComponent 抽象クラスを**図 6**（リスト 2）に示す．具体的なコンポーネントは同クラスの実装となる．

AbstractComponent クラスでは，下位層および上位層へのインタフェースとして 4 つの入力または出力ストリームを持ち，下位層からの入力ストリーム inputLower からデータの読み出し，それぞれのデータ処理を行った結果を上位層への出力ストリーム outputUpper に書き込む．そして，引数なしの connect メソッドを呼び出すことにより階層間接続を実現するが，このメソッドはコンポーネント到着後にコアシステムにより自動的に呼び出される．なお，引数付き connect メソッドは接続するストリームを明示的に指定するものである．また，connect と同様に移動直前にはストリームの切断処理を行うメソッドが自動的に呼び出される．

5. 評　　価

5.1 基本性能評価

ミドルウェアの実装は Java（JDK1.1）互換の実行環境で動作し，OS やハードウェア，ネットワークプロ

```
public abstract class AbstractComponent
    extends Agent {
  // 上層からの入力ストリーム
  protected PipedInputStream inputUpper;
  // 上層への出力ストリーム
  protected PipedOutputStream outputUpper;
  // 下層からの入力ストリーム
  protected PipedInputStream inputLower;
  // 下層への出力ストリーム
  protected PipedOutputStream outputLower;

  public void connect() {
    // 上層に接続される入力ストリームを開く
    inputUpper = new PipedInputStream();
    // 上層に接続される出力ストリームを開く
    outputUpper = new PipedOutputStream();
    // 下層に接続される出力ストリームを開く
    inputLower = new PipedInputStream();
    // 下層に接続される入力ストリームを開く
    outputLower = new PipedOutputStream();

    // エージェントの参照を取得し接続を行う
    AgentContext ac = getAgentContext();
    Enumeration agents =
       ac.getAgents(lowerAgent);
    if(agents.hasMoreElements()) {
      AgentIdentifier aid
      = (AgentIdentifier)agents.nextElement();
      ac.send(aid, "connect",
       inputLower, outputLower);
    }
    ....
  }
  public void connect(PipedInputStream in,
          PipedOutputStream out)
  throws IOException{
    try {
      // 隣接する 2 層のストリームを接続する
      inputUpper.connect(out);
      outputUpper.connect(in);
    } catch (IOException ex) {
      throw(ex);
    }
  }
}
```

図 6 リスト 2：AbstractComponent クラス
Fig. 6 List 2 : AbstractComponent.

トコルなどのプラットフォームには依存しない．現在の実装は必ずしも最適化されたものではないが，その基本性能として 2 つのセンサノード間のコンポーネントを移動・置換する際のコストを示す．センサノード間が 100BASE-T Ethernet による有線ネットワークにより接続されている場合のコストは 61 ms となり，IEEE802.11b 無線ネットワークにより接続されている場合は 64 ms となる．ここで，転送対象は空のコールバックメソッドから構成された最小化されたコンポーネントであり，各センサノードは Windows2000 および JDK1.1 が稼働する PC（PentiumIII-750 MHz）である．また，コンポーネントの実現に用いるモバイルエージェントシステムとして前述の AgentSpace を用いている．また，センサノード間転送は TCP/IP を用いており，上述の 2 つのコストはそれぞれコンポーネントの直列化（整列化），TCP コネクションの確立，直列化エージェントの転送，転送確認，復号化（逆直列化），型検証，安全性の確認処理が含まれる☆．両コストは AgentSpace システムのエージェント転送コストに対して，それぞれ 10%以下となり，ミドルウェアによるオーバヘッドは小さいといえる．なお，本稿が対象としている室内外に設置されたセンサによる人間支援ににおいてはコンポーネントの移動・置換は頻発しないことから，十分な性能を持っているといえる．また，測定に用いたコンポーネントのサイズは 8 KB となり，実際の運用時にはコンポーネントの処理に応じてサイズが増加することになるが，一般にセンサが検出するデータ量と比較すると十分に小さく，適切にデータを選別することによりネットワークトラフィックを軽減することが可能となる．

5.2 応 用 事 例

センサノードに接続されたセンサは，各コンポーネントによって管理されるため，提案するミドルウェアは，温度，圧力，湿度，位置，移動，方向，騒音レベル，照明の状態などの監視を含め，幅広いセンサタイプをサポートできる．以下，温度センサと位置センサを用いた 2 つのミドルウェアの応用事例を示し，システム構成や物理的環境の変化にミドルウェアが適用する様子を概説する．

室内温度測定システム

1 つ目の事例は室内における温度センサから構成されるセンサネットワークである．各センサノードは Pentium 互換プロセッサを搭載し，**図 7** のように LonWorks ネットワーク経由で 1 つ以上の温度センサと接続され，コアシステムは LonWorks を制御するためのゲートウェイ機能を提供している．

ここでは各センサノードの構成や要求変化を想定したいくつかの実験を行った．まず，センサの増設実験では各センサはそれの属性，たとえばセンサ制御層コンポーネントの URL，データタイプ，データ形式，測定精度，測定間隔などを示すプロファイルが用意され

☆ エージェントの転送では直列化した実行状態に加えて，エージェントを構成するコードも一括して転送している．このとき Java 言語のソケットライブラリを利用し，Java RMI は利用していない．

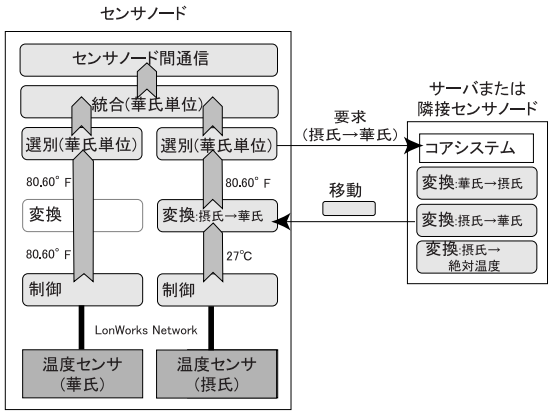

図 7 温度測定システム
Fig. 7 Reconfigurable temperature sensing system.

ていることから☆，センサノードは接続された温度センサのプロファイルを入手し，そこに示された URL に従ってそのセンサに対応したセンサ制御層コンポーネントをサーバから移動させることが確認された．そのコストは LonWorks ネットワークを介したセンサ発見処理のコストとサーバへの移動依頼メッセージ転送処理を除くと前節に示したコンポーネントの移動・置換コストとほぼ一致した．なお，モバイルエージェントを含む移動プログラムは移動先の計算リソースの取得が難しくなることが多いが，ここでは制御層コンポーネントはつねに LonWorks を介してセンサの制御・監視するだけでよく，計算リソース取得上の障害は最小化される．また，コンポーネント側処理は LonWorks の通信制御となり，そのソフトウェア開発もセンサ制御・監視も容易となる．

このほか，接続するセンサとして温度検出データをセ氏で出力するものと，絶対温度で出力するものの 2 種類を混在して接続した場合，上位層となるデータ選別層コンポーネントの要求に応じてデータ変換層コンポーネントを置換するだけでよく，その際に他の階層に影響を与えない．さらに，データ選別層コンポーネントをセ氏出力を要求するものと，絶対温度出力を要求するものを変更した場合も下位層コンポーネントの置換が要求されるだけで上位層には影響がない．これはこのミドルウェアにおける階層構造がセンサネットワークにおける各種制御・監視機能の依存関係に対応していることを示している．

モーショントラッキング

次に測定環境変化に応じたデータ選別方法の変更について概説する．これは床上の圧力センサおよび壁や天井に設置された赤外線による動体検出センサをセンサネットワークにより結合して，人間の位置や移動方向を監視するものである．

このアプリケーションは人間の存在地点およびその近傍のセンサノードだけからデータを検出するものである．人間の存在が検出された場合，センサ制御層やデータ選別層コンポーネントは近傍のセンサを制御・監視するセンサノードへそれ自身の複製を配置し，逆に人間の存在検出が一定時間ないときはそれ自身を削除する．センサネットワークは各センサノードの計算能力，記憶領域，ネットワークの通信帯域に制限が多いことから，その負荷を最小にすることが要求されるが，この方法により人間の存在地点およびその近傍を管轄するセンサノード以外にはコンポーネントの保持・稼働が不要となり，不必要なコンポーネントの稼働を避けることができる．またネットワークを介して交換される検出データも最小化される．なお，各センサノードの対応範囲が一般のオフィスビルや自宅の部屋程度であれば，人間の歩行速度に対してコンポーネントの複製と移動は十分小さいことから人間の移動に追随し，また近傍を管轄するセンサノードには必要なコンポーネントが配置されていることから，人間が移動した場合も検出時のデータ欠損を防ぐことが可能となっている．

ところで，コンポーネントの複製や移動などの配置方法はそのコンポーネント自身で定義することから，各センサノードが隣接するセンサノードのネットワークアドレスを知っていれば十分であり，センサネットワーク自体には特別な機構を用意する必要がない．これはモバイルエージェントの自律性や移動性が多用途化や環境変化への適応性を持つセンサネットワーク構築に有用であることを示している．

6. おわりに

本稿では，センサネットワークを対象としたミドルウェアを提案した．これはセンサネットワークの管理・運用における 5 つの基本機能（センサ制御，データ変換，データ選別，データ統合，センサノード間通信）に対応した階層型アーキテクチャとなる．そして各階層はモバイルエージェント技術を利用したプラグインコンポーネントとして実現され，階層間にはストリーム通信を基礎とする統一的インタフェースを提供する．この結果，測定対象となる外部環境やアプリケーション要求の変化に応じて，対応するコンポーネントを交換・拡張することができ，またその際他の階層に対する影響も最小化することができる．

☆ 実際の実装では XML 形式で記述されている．

最後に今後の課題を述べる．本稿で示したミドルウェアは特定のアプリケーションに限定したものでなく，汎用的な枠組みである．すでに位置センサや温度センサなどによる環境測定に利用しているが，広範なアプリケーションに応用することを検討している．また，セキュリティ機構として，現在の実装ではモバイルエージェントシステム側のセキュリティ機構を利用しているが，センサネットワークに最適化した保護手法が望まれる．ところで，現在の実装では各階層コンポーネントはセンサノード内で動作するが，その一部を遠隔コンピュータで稼働することにより処理の分散化が可能となる．今後は相違なコンピュータ上のコンポーネントどうしを接続する機構についても検討したい．

参 考 文 献

1) Arnold, K. and Gosling, J.: *The Java Programming Language*, Addison-Wesley (1998).
2) Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C. and Jetcheva, J.: A Performance Comparison of Multi-Hop Wireless Ad-Hoc Network Routing Protocols, *Proc. International Conference on Mobile Computing and Networking* (*Mobicom'98*) (Oct. 1998).
3) Brooks, R.R. and Iyengar, S.S.: *Multi-Sensor Fusion*, Prentice Hall (1998).
4) Fuggetta, A., Picco, G.P. and Vigna, G.: Understanding Code Mobility, *IEEE Trans. Softw. Eng.*, Vol.24, No.5 (1998).
5) Intanagonwiwat, C., Govindan, R. and Estrin, D.: Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, *Proc. Conference on Mobile Computing and Networking* (*Mobicom'2000*) (Aug. 2000).
6) Lange, B.D. and Oshima, M.: *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley (1998).
7) Perkins, C.E.: Ad-hoc On-Demand Distance Vector (AODV) Routing, IETF MANET, Internet Draft (Dec. 1997).
8) 佐藤一郎：モバイルエージェントの動向，人工知能学会論文誌，Vol.14, No.4, pp.598–605 (1999).
9) 佐藤一郎：AgentSpace：モーバイルエージェントシステム，*MACC'98* 日本ソフトウェア科学会 (1998).
10) Satoh, I.: MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, *Proc. International Conference on Distributed Computing Systems* (*ICDCS'2000*), pp.161–168, IEEE Computer Society (Apr. 2000).
11) Michahelles, F., Samulowitz, M. and Schiele, B.: Detecting Context in Distributed Sensor Networks by Using Smart Context-Aware Packets, *Proc. International Conference on Architecture of Computing Systems*, LNCS Vol.2299, pp.34–50, Springer (2002).
12) Qi, H. Iyengar, S. and Chakrabarty, K.: Distributed Multiresolution Data Integration Using Mobile Agents, *Proc. IEEE Aerospace Conference* (2001).

(平成 14 年 7 月 8 日受付)
(平成 14 年 12 月 3 日採録)

梅澤　猛

昭和 51 年生．平成 11 年慶應義塾大学理工学部電気工学科卒業．平成 13 年同大学大学院計算機科学専攻修士課程修了．現在同大学院開放環境科学専攻博士課程在学中．モバイルエージェントの研究に従事．

佐藤　一郎（正会員）

平成 3 年慶應義塾大学理工学部電気工学科卒業．平成 8 年同大学大学院理工学研究科計算機科学専攻前期博士課程修了．平成 8 年同専攻後期博士課程修了，博士（工学）．平成 13 年より国立情報学研究所助教授．この間平成 11〜14 年まで科学技術振興事業団さきがけ研究 21 研究員．分散システム，ミドルウェアに関する研究に従事．電子情報通信学会，IEEE，ACM 等会員．

安西祐一郎（正会員）

昭和 21 年生．昭和 49 年慶應義塾大学大学院博士課程修了．昭和 63 年より慶應義塾大学理工学部教授．平成 13 年より慶應義塾長．この間昭和 56 年〜57 年カーネギーメロン大学客員助教授．計算機科学，認知情報処理過程の研究に従事．工学博士．電子情報通信学会，日本認知科学会，ACM，IEEE 等会員．