

## 制御ルールを考慮した センサアクチュエータネットワーク機構の構築

青木 崇行<sup>†1</sup> 桐原 幸彦<sup>†2</sup> 中澤 仁<sup>†3</sup>  
高汐 一紀<sup>†1,†3</sup> 徳田 英幸<sup>†1,†3</sup>

ネットワーク上に分散して存在するセンサやアクチュエータを協調させ、管理者の設定する制御ルールに基づいて制御を行うセンサアクチュエータネットワーク機構を実現するために、Spinning Sensors NW ミドルウェアと Sensor Actuator Network Markup Language (SANML) と呼ぶマークアップ言語を提案する。センサアクチュエータネットワークを実現するには、異なるホスト上で動作するセンサアクチュエータ間の通信の実現だけでなく、これらの協調動作を簡単に設定できる必要がある。これらの問題を、異種複数センサアクチュエータの異種分散ノード協調問題と協調動作記述設定問題として提示する。本論文では、センサやアクチュエータの協調動作について、SANML を用いた記述設定を実現し、その SANML を Spinning Sensors NW にて読み込み各センサやアクチュエータの動作設定を行い、センサアクチュエータネットワーク機構を実現した。これらのミドルウェアとマークアップ言語を用いた実験として空調管理アプリケーションとネットワークロボットアプリケーションを構築し、ミドルウェアの動作速度やマークアップ言語の記述力の評価を行い、本研究が実際のセンサやアクチュエータを用いた環境で有効に動作することを示した。本研究により、ネットワーク上に散在する異種複数のセンサやアクチュエータにおける異種分散ノード協調問題、協調動作記述設定問題を解決し、より簡単にセンサアクチュエータネットワークアプリケーションの構築が可能になる。

### A Sensor Actuator Network Architecture with Control Rules

SOKO AOKI,<sup>†1</sup> YUKIHIKO KIRIHARA,<sup>†2</sup> JIN NAKAZAWA,<sup>†3</sup>  
KAZUNORI TAKASHIO<sup>†1,†3</sup> and HIDEYUKI TOKUDA<sup>†1,†3</sup>

This paper presents a Spinning Sensors NW middleware which realizes a distributed sensor actuator network and a Sensor Actuator Network Markup Language (SANML) to write the controlling rules of networked sensors and

actuators. In the realization of sensor actuator network architecture, the programmer needs to coordinate sensors and actuators in the network and set the control rules of them. We have stated these problems as distributed coordination problem and coordination setting problem. The Spinning Sensors NW middleware provides the mechanism to coordinate multiple networked sensors and actuators. The SANML provides a format to express the control rules of sensors and actuators. We implemented a air conditioning application to test the performance of the middleware and the expressive power of the markup language. This paper overcomes both distributed coordination problem and coordination setting problem in sensor actuator network architecture and increase the flexibility and combinations of sensors and actuators.

#### 1. はじめに

ユビキタスコンピューティング技術の発達にともない、コンピュータや情報家電機器だけでなくセンサやアクチュエータもネットワークに接続されつつある。センサにはワイヤレスセンサノードのように無線で接続されるものから、工場内の特殊用途センサといった有線で接続されるものがある。アクチュエータも多種多様になっており、ネットワーク越しに制御可能なモータやシリンダといった単純な動きをするものから、空間内を移動可能なネットワークロボット等、複雑な制御が必要なアクチュエータ等が存在する。センサアクチュエータネットワークでは、これらのセンサとアクチュエータを相互に接続することにより新たなアプリケーションやサービスを創出する。センサアクチュエータネットワークを実現するにあたっての課題の1つに、これらを組み合わせて利用する際の通信・制御問題がある。たとえば、ある温度センサと湿度センサで取得されたデータを検証し、その結果をオフィス内のファンに通知するシステムでは、センサとファンの接続やデータフォーマット変換が必要になる。

センサアクチュエータネットワークに関する研究は、モバイルロボットノードとしてセンサとアクチュエータを組み合わせるネットワークロボットの研究<sup>1)</sup> や、ネットワーク上に分散するソフトウェアコンポーネント間にイベント配送機構を提供する研究<sup>2)-4)</sup> 等いくつか

<sup>†1</sup> 慶應義塾大学大学院政策・メディア研究科  
Graduate School of Media and Governance, Keio University

<sup>†2</sup> 株式会社トリプルダブル  
Triple Double Corporation

<sup>†3</sup> 慶應義塾大学環境情報学部  
Faculty of Environment and Information Studies, Keio University

行われている。しかし、既存の技術ではセンサとアクチュエータ間の組合せが固定的であったり、ネットワーク上に分散するヘテロジニアスなセンサとアクチュエータの協調動作を実現できない。また、組合せの設定に関しても、コンピュータ間や情報家電機器間における研究においては、GUI 等で機器間の接続はできるが、センサイベントの発生頻度、センサデータの多様性、多様な管理方法等といったセンサやアクチュエータに特有な機能を実現できていない。ここでは、前者を異種分散ノード協調問題、後者を協調動作記述設定問題として定義する。

本論文では、これらの問題を解決する機構として、Spinning Sensors NW ミドルウェアと Sensor Actuator Network Markup Language (SANML) を提案する。Spinning Sensors NW ミドルウェアでは、センサやアクチュエータといったハードウェアの抽象化を行うとともに、ネットワーク上に分散して存在しているセンサとアクチュエータを協調させ、センサアクチュエータネットワークアプリケーションを構築可能にする。また、センサとアクチュエータの協調に関する設定記述用のマークアップ言語として SANML を提案する。SANML を用いることで、センサとアクチュエータの組合せ、制御ルール、データ処理方法等の記述が可能になる。これらの機構を用いて、温度センサとファンを用いた空調管理アプリケーションとローテーションセンサとモータを用いたネットワークロボットアプリケーションを構築し、ミドルウェアのパフォーマンス測定を行うとともに、マークアップ言語の記述力の評価考察を行った。

本論文では、まず 2 章においてセンサアクチュエータネットワークについて述べる。3 章では、センサとアクチュエータの協調設定記述用マークアップ言語である SANML について説明する。そして、4 章では、SANML を用いたミドルウェアである Spinning Sensors NW ミドルウェアを提案し、5 章において、構築したサンプルアプリケーションの説明をする。さらに 6 章において評価を行う。7 章で関連研究を整理し、8 章で本論文をまとめる。

## 2. センサアクチュエータネットワーク

本論文では、周囲の環境情報を取得する機器をセンサ、物理空間上を移動したり、ディスプレイやスピーカといったデバイスを通して情報を出力したりする機器をアクチュエータとし、それらが相互に接続された環境をセンサアクチュエータネットワークと定義する。従来のセンサネットワークに対してアクチュエータが接続されることにより、センサデータを可視化できるだけでなく、そのデータによってアクチュエータを動作させることで、利用環境をより快適・便利にしつつ、地球環境に対しては低負荷なシステム実現が可能になる。な

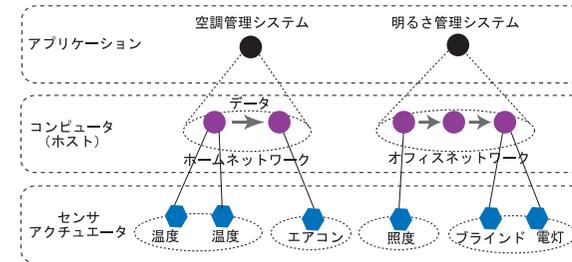


図 1 センサアクチュエータネットワーク

Fig. 1 Examples of sensor actuator network.

お、単一のセンサや単一のアクチュエータのことをセンサノードやアクチュエータノード、もしくは単にノードと呼ぶことにする。また、組み合わせて利用するセンサとアクチュエータをまとめて 1 つのフュージョンとして取り扱うことにする。センサアクチュエータネットワークのイメージを図 1 に示す。

### 2.1 センサアクチュエータネットワークの特徴

コンピュータネットワークや情報家電ネットワーク等の従来から存在するネットワークシステムと比較した、センサアクチュエータネットワークの特徴について、多様性、分散性、散発性の側面から説明する。

- 多様性

コンピュータや情報家電のネットワークでは機器相互を接続する規格、ネットワークプロトコル、フレームワークが多く開発されており、多様な機器が連携し始めている<sup>5)</sup>。しかしセンサアクチュエータネットワークにおいては MOTE<sup>6)</sup> といった多くの研究機関が利用している製品はあるものの、その他のハードウェアの多様な通信形式、機能、動作を相互に連携するようなフレームワーク研究はまだ少ない。

- 分散性

センサやアクチュエータがネットワーク接続機能を持つことにより、それらがインターネットからもアクセス可能になり、ユビキタスセンサアクチュエータネットワーク環境が構築されつつある。複数のセンサやアクチュエータが接続される際、各ノードからどんなデータが収集され、そのデータ処理をどのノード上で処理をするかが、システム管理や負荷分散の面で重要な検討要素になる。

- 散発性

センサでの取得値の変化(センサイバント)は,周辺環境の変化や生物の移動等により発生し,周期的な発生と非周期的な発生がある.周期的な場合には定期的にセンサデータを取得すればよいが,非周期的なセンサイバント発生においては,定期的なセンサの監視は効率が悪い.またセンサの精度や感度によっては,必要以上にセンサイバントが多く発生する可能性があり,その散発的な発生頻度を調整する必要が出てくる.

本研究ではこれらセンサアクチュエータネットワーク特有の問題に対応するミドルウェアとマークアップ言語の設計と実装を行った.

## 2.2 利用シナリオ

センサアクチュエータネットワークを用いた具体例として,3種類のシナリオを用意した.

### 2.2.1 オフィス空調管理

あるオフィスにおいて,快適かつ低消費電力を目指した職場環境実現のために,温度センサ,湿度センサ,外気を取り入れるファン,空調機器からなるセンサアクチュエータシステムを導入することにした.本システムでは,オフィス内外に設置した温度湿度センサの取得データによって,ファンと空調機器のいずれかを動作させてオフィス内を快適に保つ.

このシナリオを実現するには,機器それぞれがネットワークに接続されている必要がある.また,オフィスの外と中の気温と湿度について,複数センサからのデータ比較結果に応じてファンと空調から動作させる機器を決定し(条件分岐処理),実際にその制御を行う.

### 2.2.2 農場監視

ある畑において,作物の良好な発育を実現するために,温度センサ,土湿度センサ,および携帯電話による状況確認ディスプレイからなるセンサアクチュエータシステムを導入する.本システムでは1時間に1回温度と土湿度を管理者に送信し,畑の監視を実現する.

このシナリオを実現するには,機器それぞれがネットワークに接続されていてかつ,周期的(繰返し処理)にユーザにデータを送信する必要がある.

### 2.2.3 ネットワークロボット

ある精密機器工場を遠隔地から監視するために,移動型ロボットと操作系センサからなるセンサアクチュエータシステムを導入する.本システムでは,遠隔地の操作者がロボットの移動・動作パターンを入力し,工場内の監視ロボットを制御する.

このシナリオを実現するには,機器それぞれがネットワークに接続されていてかつ,遠隔地の操作者がロボットの移動や動作に関する一連の流れ(手順実行)を設定できる必要が

ある.

## 2.3 異種分散ノード協調問題と協調動作記述設定問題

利用シナリオで述べたようなセンサアクチュエータネットワークアプリケーションを構築するにあたって,本論文では異種分散ノード協調問題と協調動作記述設定問題の2種類の問題を提起し解決する.

異種分散ノード協調問題とは,ネットワーク上に分散して存在する異種複数のセンサとアクチュエータを接続して利用する枠組みがないことと定義する.たとえば既存の分散コンポーネント技術<sup>2)-4)</sup>ではネットワーク上のサービスを統合するフレームワークを提供しているものの,センサやアクチュエータに特有なデータ処理や,センサに特有なポーリングモデルとイベントドリブンモデルを考慮した通信機能がない.またセンサデータフュージョンミドルウェア<sup>7),8)</sup>においては,複数の同種のセンサノードを対象としており,異種のセンサとアクチュエータの組合せは考慮されていない.ロボットとしてモバイルセンサノードを実現している研究<sup>1)</sup>においては,同一ハードウェア上にセンサやアクチュエータが実装されており,ネットワーク越しの連携の実現やセンサとアクチュエータの組合せの変更が困難である.これら既存研究での課題をふまえ,本論文では Spinning Sensors NW ミドルウェアにおけるハードウェアの抽象化,協調処理,データ処理の機能を提供することにより解決する.特に協調処理の実現では,フュージョンという1つ以上のセンサとアクチュエータを組み合わせた仮想的なノードを提案する.

また,協調動作記述設定問題とは,異種複数のセンサとアクチュエータの協調動作を実現するにあたり,その協調方法の設定を簡単に設定できないことと定義する.たとえば先行研究であるセンサネットワークアプリケーション構築用のフレームワーク<sup>9)</sup>やプログラミングツールキット<sup>10)</sup>では,同種のセンサノードを対象としたアプリケーション構築は可能であるが,異種のセンサやアクチュエータの組合せ,またそれらの管理についての複雑な制御ルールの設定をする機能は実現されていなかった.これら既存研究での課題をふまえ,本論文ではXMLを用いたマークアップ言語での協調設定を実現し,センサアクチュエータ協調設定フォーマットであるSANMLを提供することにより解決する.

## 3. センサアクチュエータ協調設定マークアップ言語: SANML

複数のセンサやアクチュエータの協調設定を設定ファイルとして記述し,協調を容易に実現するために,センサアクチュエータ協調設定マークアップ言語である Sensor Actuator Network Markup Language (SANML)を開発する.本マークアップ言語では,アプリケー

ションで用いるセンサとアクチュエータを指定するだけでなく、センサデータの取得やアクチュエータの起動についてのルール設定や、センサとアクチュエータ間でやりとりされるデータの処理方法の指定等を可能にする。

### 3.1 言語要件

2.1 節のシナリオから導出される機能要件は以下のとおりである。複数のセンサやアクチュエータを組み合わせて利用する。また、条件分岐や繰返しといったルールが使われており、センサとアクチュエータ間でのデータの処理も実現される必要がある。したがって、センサアクチュエータネットワークアプリケーションを記述するマークアップ言語の言語要件として、組合せ設定、ルール設定、データ処理設定をあげる。

#### 組合せ設定

アプリケーションにおいて使用するセンサとアクチュエータについて、その名前や IP アドレスを記述する。ここでのセンサとアクチュエータの組合せがミドルウェアにおいてフュージョンとして実現される。なおセンサとアクチュエータの数は 1 対 1 だけでなく、多対 1、1 対多、多対多の関係を想定している。

#### ルール設定

ルール設定では、各センサからフュージョンへデータを送信する際の条件や、フュージョンでデータを統合した際に実際にそれらを用いてアクチュエータへの制御を行うかを判断する条件等の設定を可能にする。

#### データ処理設定

データ処理設定では、センサからアクチュエータに送られるセンサデータについて、数値演算やフォーマット変換が必要なときに、ここに記述する。またフュージョンにおけるセンサデータの集約処理についても、データ処理設定として記述する。

上記言語要件において、特にルール設定やデータ処理設定はセンサデータの取扱やアクチュエータの駆動に関する記述を実現していることから、センサアクチュエータネットワーク特有の要件になっている。

### 3.2 マークアップ言語の設計と実装

SANML の設計と実装を XML Schema をベースとして行った。SANML の基本構造例を図 2 に、また SANML 内で用いている各タグやそれらの保有要素の一覧を表 1 に示す。SANML は、ルート要素である `< sanml >`、センサ、アクチュエータ、フュージョンを定義する子要素である `< sensor >`、`< actuator >`、`< fusion >` から構成される。`< sensor >` や `< actuator >` においては、id として実装クラス名を、また port として同一コンピュー

```

<!--ルート要素 -->
<sanml>
  <!--子要素：入力（センサ）設定 -->
  <sensor name="SensorName">
  </sensor>

  <!--子要素：出力（アクチュエータ）設定 -->
  <actuator name="ActuatorName">
  </actuator>

  <!--子要素：フュージョン（組み合わせ）設定 -->
  <fusion>
    <!--孫要素：データ処理設定 -->
    <process type='for'>
      <!-- 曾孫要素：ルール設定 -->
      <sensor ref='SensorName' symbol='rule' value='x'>
      </process>
      <actuator ref="ActuatorName">"
    </fusion>
  </sanml>

```

図 2 SANML 基本構造

Fig. 2 Basic composition of SANML.

タ上に複数のセンサやアクチュエータが存在する場合において、その実装クラスが使用するネットワークポート番号を指定する。また、`< fusion >` においてはセンサやアクチュエータの組合せ設定とともに、センサの保有要素として条件分岐や繰返しといったルール設定や、`< process >` タグを用いて複数センサからのデータの合計、平均、最大、最少、分散等を計算するデータ処理設定の記述が可能になっている。フュージョンで設定可能な機能の詳細は表 2 に示す。

### 3.3 SANML 記述例

SANML を用いた具体的な記述例を、2.2 節であげた 3 つのシナリオを用いて記述した。図 3 ではオフィス空調管理例を用いて、外気の気温により 2 種類のアクチュエータを使い分ける条件分岐について、`< rule type = ' ifelse ' >` タグを用いて記述している。ここでは外気温が 25 度以上ならファンを、25 度未満ならエアコンを制御するようになっている。また、図 4 では農場監視例を用いて、農場の土の温度と湿度を 60 分ごとに農場管理者の携帯電話に送る繰返し処理を `< rule type = ' for ' >` タグを用いて記述している。土の温度については、`symbol = ' step '` 要素を用いることで、0.1 度ずつの変化を取得している。この例では、2 種類のセンサからのデータを 2 種類の描画アクチュエータに送る設定になっている。さらに図 5 ではネットワークロボット例を用いて、遠隔地からの制御を記述している。

表 1 SANML の定義  
Table 1 Definition of SANML.

SANML タグ	要素：内容	保有要素	機能
sanml	ルート要素	sensor , actuator , fusion	SANML の使用宣言
sensor	子要素：	id	センサ実装クラスの指定
	入力設定	port	同一ノードに複数ポートがある場合のポート指定
actuator	子要素：	id	アクチュエータ実装クラスの指定
	出力設定	port	同一ノードに複数ポートがある場合のポート指定
fusion	子要素： フュージョン設定	sensor	組合せ設定：sensor , actuator を羅列
		actuator	ルール設定：条件を symbol と value で指定
		process	データ処理設定：合計・平均・最大・最小等の指定
		rule	ルール設定：条件分岐 (ifelse) や繰返し (for) の指定

表 2 SANML Fusion における各種設定一覧  
Table 2 Definition of SANML fusion setting.

fusion タグ内に記述した sensor タグ, actuator タグに付随するルール設定		
項目	SANML タグ	内容
四則演算	symbol = '+,-,*,/'	取得データと symbol に続いて記述する value の値を計算する
比較演算	symbol = '{<,<=,=,>,>}'	取得データと value の値を比較して条件内ならデータ出力
時間演算	symbol = 'interval'	取得データを指定時間ごとに出力 (例：value=1000 ならデータが 1 秒ごとに出力)
時間演算	symbol = 'wait'	実行後に指定時間待つ (例：value=1000 ならその処理を実行後 1 秒待つ)
その他	symbol = 'step'	取得データを指定精度で出力 (例：value=10 ならデータが 10 変わるごとに出力)
process タグで設定可能なデータ処理設定		
項目	SANML タグ	内容
標準機能	type = {sum,ave,max,min,var}	複数の取得データの合計, 平均, 最大, 最小, 分散の値を出力
ユーザ設定	type = "filename"	ユーザ指定の計算・データ処理を外部ファイル (filename) で指定する
rule タグで設定可能なルール設定		
項目	SANML タグ	内容
条件分岐	type = 'ifelse'	sensor の値が条件内ならば actuator を制御する
繰返し	type = 'for'	sensor のデータ取得や actuator の制御を繰り返す

ここではセンサを用いずに、ロボットの制御手順を < process > タグ内に羅列することで、アクチュエーションの手順を表現している。

#### 4. センサアクチュエータ分散協調ミドルウェア：Spinning Sensors NW

本章では、センサアクチュエータ分散協調ミドルウェアについてその機能要件を述べ、開発を行った Spinning Sensors NW ミドルウェアについて、その設計、実装、動作手順を説明する。

#### 4.1 機能要件

センサアクチュエータネットワークを実現するミドルウェアとしての機能要件について、ハードウェアの抽象化、ネットワーク対応、フュージョンの構築の 3 点について説明する。ハードウェアの抽象化

センサアクチュエータネットワークを構成するセンサやアクチュエータの開発元、機能、制御方法は多岐にわたる。ミドルウェアにおける汎用性を維持するために、ハードウェアの仕様によらないミドルウェアの実装が必要になる。

```

<!--子要素：入力（センサ）設定 -->
<sensor name="OutsideTempSensor">
  <id>ssnw://sensor/OutTempImpl/192.168.1.2/sensor1/</id>
</sensor>

<!--子要素：出力（アクチュエータ）設定 -->
<actuator name="FanActuator">
  <id>ssnw://actuator/FanImpl/192.168.1.4/act1/</id>
</actuator>
<!--子要素：出力（アクチュエータ）設定 -->
<actuator name="AirconActuator">
  <id>ssnw://actuator/AirconImpl/192.168.1.5/act1/</id>
</actuator>

<!--子要素：データ処理設定 -->
//外の気温が 25 度以上なら Fan, 25 度未満なら Aircon の実行
<fusion>
  <rule type='ifelse'>
    <sensor ref='OutsideTempSensor' symbol='>' value='25'>
      <actuator ref='FanActuator'></actuator>
    </sensor>
    <sensor ref='OutsideTempSensor' symbol='<' value='25'>
      <actuator ref='AirconActuator'></actuator>
    </sensor>
  </rule>
</fusion>

```

図 3 オフィス空調管理 SANML

Fig. 3 Air conditioning management written by SANML.

### ネットワーク対応

協調させるべきセンサとアクチュエータはネットワーク上の別ノード（別 IP アドレス）として存在する場合がある。本ミドルウェアではそのようなネットワーク越しのセンサとアクチュエータの通信を実現する。

### フュージョンの構築

センサとアクチュエータの協調を実現するフュージョンについて、プログラム内にプログラミング言語で記述するのではなく、より簡便に外部設定ファイルとして記述できる必要がある。

### 4.2 ミドルウェアの設計と実装

センサやアクチュエータを協調して利用するための、センサアクチュエータ分散協調ミドルウェアである Spinning Sensors NW (Network) を開発する。本ミドルウェアは Spinning Sensors ミドルウェア<sup>11)</sup> を拡張して実現している。Spinning Sensors NW では、Spinning

```

<!--子要素：入力（センサ）設定 -->
//SoilTempSensor と SoilMoistSensor の設定をここに書く

<!--子要素：出力（アクチュエータ）設定 -->
//CellphoneActuator1 と CellphoneActuator2 の設定をここに書く

<!--子要素：データ処理設定 -->
//60 分ごとにセンサデータを携帯電話に送信
<fusion>
  <rule type='for'>
    <sensor ref='SoilTempSensor'
      symbol='interval' value='3600' symbol='step' value='0.1'>
      <actuator ref='CellphoneActuator1'></actuator>
    </sensor>
    <sensor ref='SoilMoistSensor' symbol='interval'
      value='3600'>
      <actuator ref='CellphoneActuator2'></actuator>
    </sensor>
  </rule>
</fusion>

```

図 4 農場監視 SANML

Fig. 4 Farm monitoring written by SANML.

```

<!--子要素：出力（アクチュエータ）設定 -->
//NetworkedRobot1 の設定をここに書く

<!--子要素：出力（アクチュエータ）設定 -->
//NetworkedRobot2 の設定をここに書く

<!--子要素：出力（アクチュエータ）設定 -->
//NetworkedRobot3 の設定をここに書く

<!--子要素：データ処理設定 -->
//以下の 3 種類のアクチュエーションを 5 秒ごとに実行する
<fusion>
  <actuator ref='NetworkedRobot1' symbol='wait' value='5'>
  <actuator ref='NetworkedRobot2' symbol='wait' value='5'>
  <actuator ref='NetworkedRobot3' symbol='wait' value='5'>
</fusion>

```

図 5 ネットワークロボット SANML

Fig. 5 Farm monitoring written by SANML.

Sensors では不可能だったネットワーク越しのセンサとアクチュエータの協調の実現や、記述フォーマットを用いた協調の設定等を実現したところに差異がある。本ミドルウェアでは、別 IP アドレスである複数のセンサやアクチュエータを統合させて相互の通信を可能にするとともに、3 章で説明した SANML を併用することにより複雑な協調動作を簡便に実現で

きるようにする．Spinning Sensors NW の機能は，Spinning Sensors の機能を包含する．

Spinning Sensors NW の実装は Java 言語<sup>12)</sup> を用いて行った．ユーザは本ミドルウェアをライブラリとしてインストールし，パスの設定をするだけでアプリケーション開発を行える．Spinning Sensors NW のソフトウェア構成を図 6 に示す．センサやアクチュエータに対応するクラスはそれぞれハードウェアに特化したコードを記述する実装クラスとハードウェアによらずに共通の機能を提供する抽象クラスの 2 層に分かれている．フュージョンクラスでは，外部ファイルとして用意される SANML を読み込む．

センサ，フュージョン，アクチュエータ間の通信については，センサにおける値の変化があった際に各センサ実装クラスで継承している Observable クラスを用いてフュージョンへ変更が通知されるイベントドリブンシステムを採用した．また，センサデータの定期的な収集モデルであるポーリングモデルについては，SANML における `< rule type = ' for ' >` タ

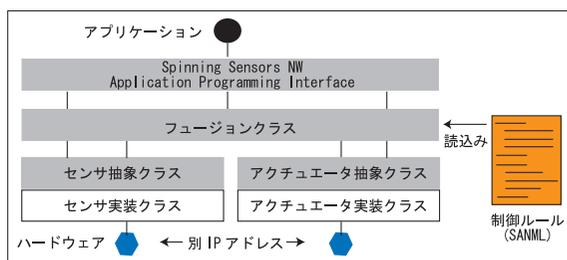
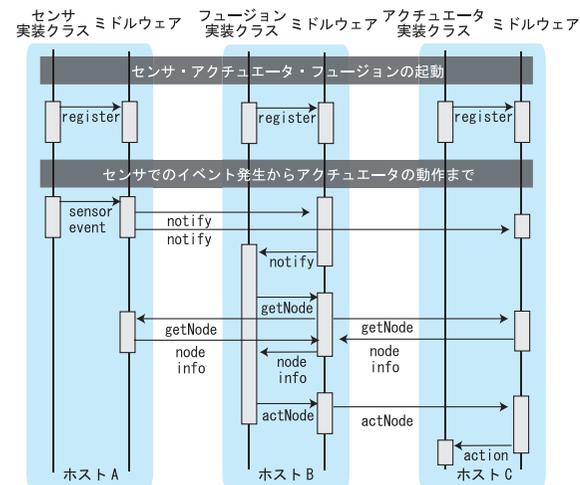


図 6 Spinning sensors NW ソフトウェア構成  
Fig. 6 Spinning sensors NW software architecture.

グを使用して実現可能である．現在 Spinning Sensors NW に対応しているセンサアクチュエータ一覧を表 3 に示す．

### 4.3 動作手順

Spinning Sensors NW を用いたセンサアクチュエータネットワークアプリケーションの動作手順を図 7 と以下に示す．まず初めにネットワークに接続された各コンピュータホス



※acknowledgement はシーケンス図からは省略

図 7 シーケンス図  
Fig. 7 Spinning sensors NW sequence chart.

表 3 サポートされているセンサとアクチュエータの例  
Table 3 List of supported sensors and actuators.

分類	機器名	内容	実装クラス名
Sensor	TECO uPart <sup>13)</sup>	light, temperature, movement	UpartSensorImpl
Sensor	LEGO Mindstorms <sup>14)</sup>	light, sound, ultrasonic, touch	MindstormsSensorImpl
Sensor	Phidgets <sup>15)</sup>	temp, light, rotation, slider	PhidgetsSensorImpl
Sensor	Phidgets RFID <sup>15)</sup>	RFID reader and tags	PhidgetsRFIDImpl
Robotic Actuator	LEGO Mindstorms	motor	MindstormsActuatorImpl
Robotic Actuator	Phidgets	motor	PhidgetsActuatorImpl
Service Component	Aviosys IPPower	power control	IpPowerImpl
Service Component	JFreeChart	graph viewer	DataViewerImpl
Service Component	Apple Quicktime	movie player	VideoControllerImpl

```

public class NodeSender implements Serializable{
private static NodeSender instance;
private ObjectOutputStream output;
private ConfigManager config;
//関連ノードを送信する
public synchronized NotifyMessage send(NotifyMessage msg){
Socket socket = null;
try {
//関連ノード取得依頼が来た場合に、通信ソケットの生成
if(msg.getCommand().equals(CoreConstants.REGISTER_NODE)){
socket = makeSocket(config.getServerId(),
config.getServerPort());
}
catch(){//エラー処理省略
}
//ノード情報をソケットに書き込む
output = new ObjectOutputStream(socket.getOutputStream());
output.writeObject(msg);
output.flush();
//書き込み終了
output.close();
socket.close();
}
}
public synchronized Socket makeSocket(String ip,int port){
//socket 開始処理
}
public synchronized void closeSocket(String ip){
//socket 終了処理
}
}

```

図 8 関連ノードの取得部サンプルコード  
Fig. 8 Sample code of spinning sensors NW.

ト上において、センサ、アクチュエータ、フュージョンを起動する。この起動においては、各ホストで動作するミドルウェアに対してそれぞれのセンサ、アクチュエータ、フュージョンの情報を登録するだけで、まだネットワーク通信は発生しない。ユーザが SANML を用いてフュージョンを構成するセンサとアクチュエータを記述し、そのフュージョンを起動してミドルウェアに登録することを、フュージョンの構築と呼ぶ。

センサでイベントが発生し、それに応じてアクチュエータを動作させるシーケンスは以下の 3 手順である。

#### 1. センサイベントの通知

センサにおいてセンサデータに変化があった場合、その変化をまず同ホスト上で動作

しているミドルウェアに通知した後、そのミドルウェアから別ホスト上で動作しているミドルウェアに通知する。通知を受信した各ホスト上のミドルウェアでは、そのセンサデータが自ホストで登録されているフュージョンやアクチュエータと関連するかを判定した後、関連する場合はその実装クラスにデータを通知する。

#### 2. 関連ノードの取得

フュージョン実装クラスにおいて、自分の関連ノードとして設定されているセンサ実装クラスとアクチュエータ実装クラスのインスタンスを取り寄せる。ここでは、SANML に記述してある実装クラス名や IP アドレスから構成される識別子を基に、インスタンスの取得を行う。図 8 に関連ノードを送信する箇所のサンプルコードを示す。

#### 3. アクチュエータの制御

アクチュエータの制御は、送られてきたセンサデータに基づいて行う。送られてきたセンサデータは、フュージョンにおいて SANML に記述してあるデータ変換処理に基づいて処理をされ、アクチュエータが受信可能な新たな制御コマンドとしてアクチュエータに送信される。

#### 5. サンプルアプリケーション

Spinning Sensors NW ミドルウェアと SANML を用いて、空調管理アプリケーションとネットワークロボットアプリケーションの構築を行った。

##### ● 空調管理アプリケーション

空調管理アプリケーションでは、Phidgets の温度センサと Aviosys の電源管理ユニットを経由した扇風機を利用して、温度がある閾値以上に上昇したら扇風機のスイッチを入れ、閾値以下の場合にはスイッチをオフにするという条件でセンサアクチュエータアプリケーションを構築した。Phidgets のセンサはホスト A に USB で接続されており、また Aviosys の電源管理ユニットはシリアル接続でホスト B に接続されている。ホスト A と B はそれぞれ Ethernet によりネットワーク接続されている。

##### ● ネットワークロボットアプリケーション

ネットワークロボットアプリケーションでは、Phidgets のつまみ型ローテーションセンサと Phidgets のステッピングモータを利用して、ローテーションセンサを回転させた分だけ、ステッピングモータの角度を変更するという条件でアプリケーションを構築した。空調管理アプリケーションと同様に、ローテーションセンサとステッピングモータはそれぞれ別のホストに USB で接続されており、それらのホストは Ethernet によ

りネットワーク接続されている。

これらのサンプルアプリケーションの構築では、Java のプログラムの加筆や変更を必要としておらず、SANML を用いたフュージョン設定ファイルを空調管理用とネットワークロボット用の 2 種類を作成することで実現している。また、両アプリケーションともに、ネットワーク越しのセンサとアクチュエータの協調動作になっており、物理的に離れたセンサとアクチュエータを協調できることを検証した。

空調管理では使用したセンサが温度センサのためにセンサイベントの発生頻度は低かったが、ネットワークロボットでは、つまみを 1 回転させると値が 0 から 999 まで変わるローテーションセンサを使用したために、頻繁にセンサイベントが発生した。また、ステッピングモータ側でもコマンドを受けてから実際にその角度に移動するまでに数百ミリ秒かかっており、この間に新たなセンサデータを受付けた場合には、1 つ目の角度にした直後に再度回転を開始するという挙動になっている。このようなセンシティブなセンサや、動作に時間のかかるアクチュエータを使用する場合は、フュージョン設定におけるデータ変換を利用して（たとえば、0 から 999 までとる値を 0 から 100 までになるような正規化処理を行うようにする）、発生頻度を調整することが有効になる。

## 6. 評価

Spinning Sensors NW ミドルウェアと SANML の評価として、実際のハードウェアとミドルウェアを用いた実験評価、SANML を用いたセンサアクチュエータアプリケーションの制御ルールの記述力評価、そしてセンサアクチュエータネットワーク用ミドルウェアとしての考察評価を行った。

### 6.1 実験評価

実験評価として、ミドルウェアの動作速度を計測するパフォーマンス測定と、実際にセンサデータを収集した結果からその動作を評価するセンサデータ収集評価の 2 種類を実施した。

#### 6.1.1 定量的評価：パフォーマンス測定

ミドルウェアのパフォーマンス測定として、図 9 に示す 2 通りの実験環境にて動作速度測定実験を実施した。実験 1 では、センサとフュージョンをノード A で、アクチュエータをノード B で実行し合計 2 ノード用いた。実験 2 ではセンサをノード A、フュージョンをノード B、アクチュエータをノード C として、合計 3 ノード用いた。表 4 にそれぞれの実験で用いた計算機環境を示す。本実験でのフュージョンでは、5 章で述べたネットワークロボットアプリケーションを用いており、動作時間測定としてセンサイベントの発生からアクチュエータへの制御コマンド送信までを計測した。

動作時間について、実験 1 では約 650 ミリ秒、実験 2 では約 850 ミリ秒かかっている。この所要時間において、特に時間を要しているのは、オブジェクト化したセンサやアクチュエータのデータ（IP アドレス、実装クラス名、各機器が持つ機能のハッシュテーブル等）をシリアライゼーションし、ネットワーク経由で別のコンピュータに送信する箇所（フュージョンによる関連ノードの取得）や、ネットワーク越しのイベント送信（アクチュエータへのコマンド送信）であり、1 回のネットワーク通信に約 200 ミリ秒要している。実験 2 の方が実験 1 よりネットワーク越しの通信が 1 回多いために、200 ミリ秒の差が出ている。ネットワーク越しの通信の回数に応じて 100 ミリ秒のオーダーで所要時間が増えているため、高

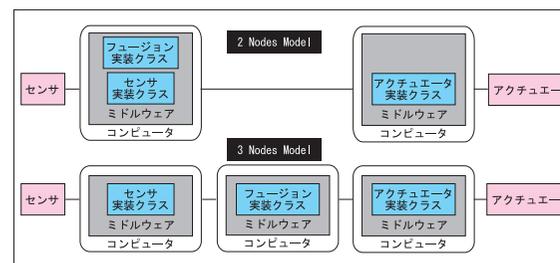


図 9 定量的評価における機器構成

Fig.9 Hardware composition in performance evaluation.

表 4 測定環境

Table 4 Evaluation environment.

ホスト	メーカー	CPU	Memory	Network	OS
ホスト A	Lenovo Thinkpad X60	Intel Core 2 1.83 GHz	1.5 GB	1 Gbps	Windows XP SP3
ホスト B	Lenovo Thinkpad T60	Intel Core 2 2.00 GHz	3.0 GB	1 Gbps	Windows XP SP1
ホスト C	Lenovo Thinkpad R40	Intel Core 2 1.46 GHz	1.0 GB	1 Gbps	Windows XP SP2

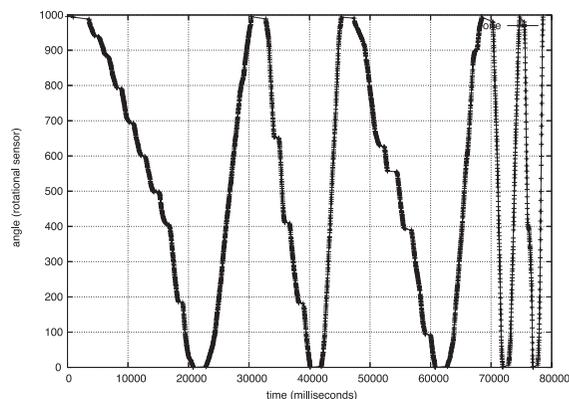


図 10 センサデータ収集評価 (刻み幅 1)

Fig. 10 Evaluation of collected sensor data (1/1).

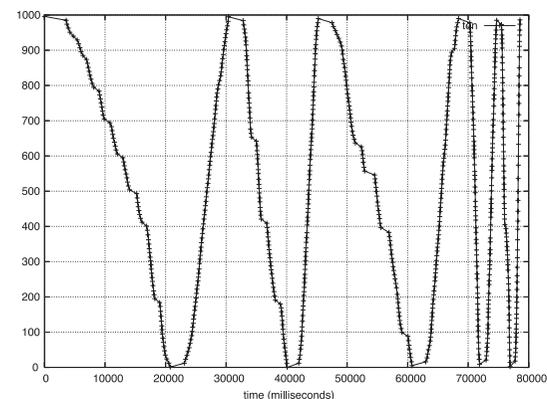


図 11 センサデータ収集評価 (刻み幅 10)

Fig. 11 Evaluation of collected sensor data (1/10).

速性を求められるロボットアプリケーション等では、ネットワーク通信が少なくなるよう各実装クラスを同ホスト上に配置して対応する。空調管理のような気温変化のモニタリングでは、数秒のオーダでの変化検知で十分であるため、本モデルウェアの実行速度で要件を満たしている。

### 6.1.2 センサデータ収集評価

Spinning Sensors NW におけるセンサデータの散発性への対応を評価するために、ローテーションセンサを用いたセンサデータ収集実験を行った。本実験では、5章のサンプルアプリケーションで用いたローテーションセンサを人間が指で回転させ、そのセンサデータを描画した。図中の横軸を時間、縦軸をローテーションセンサの値で示した。図 10、図 11、図 12 では、それぞれ人間の操作は同じだが、センサでのイベント発生条件を変えている。図 10 ではセンサのハードウェアの仕様に合わせ 0 から 1000 の値をそのまま発生させ、図 11、図 12 ではそれぞれ 10 刻みずつ、100 刻みずつ発生させている。つまり、図 11、図 12 ではセンサイイベントの発生数を減少させることで、ネットワークへの負荷を軽減する効果がある。

センサから発生するイベント数は、図 10 では合計 2165 回のところ、図 11 では合計 711 回、そして図 12 では合計 82 回になった。センサイイベント発生条件を 100 刻みにするとセンサイイベント数は抑えられるものの、描画した結果の波形図を比較すると波形細部の表現精度が低くなることが分かった。したがって、大まかなセンサデータの変動が必要な空調管理

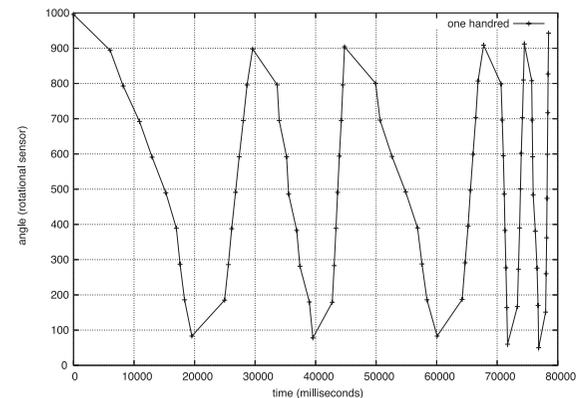


図 12 センサデータ収集評価 (刻み幅 100)

Fig. 12 Evaluation of collected sensor data (1/100).

例のようなアプリケーションにおいては刻み幅を大きくし、細かなデータ遷移が必要なネットワークロボットのようなアプリケーションでは刻み幅を小さくし、より詳細にデータを取得するように変更すべきである。これらの設定変更は SANML 内に *symbol = 'step'* 要素を記述することで、Spinning Sensors NW モデルウェアに反映される。

表 5 記述力評価

Table 5 Evaluation of descriptive power.

技術名	組合せ設定 (家電)	組合せ設定 (センサアクチュエータ)	データ処理設定	ルール設定	競合ルール防止
USDL <sup>9)</sup>	対応済み	アクチュエータ未対応	対応済み	未対応	未対応
ECPAP <sup>16)</sup>	汎用性あり	汎用性あり	対応済み	未対応	対応済み
InterPlay <sup>4)</sup>	対応済み	未対応	未対応	対応済み	未対応
SANML	未対応	対応済み	対応済み	対応済み	未対応

## 6.2 定性的評価：制御ルールの記述力

SANML を用いた制御ルールの記述力評価として、プログラミング言語内での記述との比較と、制御ルールにおける記述可能項目について評価を行う。

センサやアクチュエータの組合せを設定する方法として、プログラミング言語内にコードで記述する方法がある。この記述法では、その言語の記法やミドルウェアの API 等を理解している必要があった。一方で SANML ではマークアップ言語であるので、プログラム内での記述と比較して、センサアクチュエータネットワークを設定する開発者・管理者に要求する技術的難易度は低い。またプログラム内に記述する方法では、フュージョンの設定を変えるたびに再コンパイルが必要になるが、設定を SANML で外部ファイルとして保持することで、その必要はなくなる。たとえば、センサからのデータ取得間隔を変更するだけでもプログラム全体の再コンパイルが必要であった。SANML の設計にあたっては 2 章で示したシナリオをベースに、3 章の図 3、図 4、図 5 で示すような一般的なアプリケーション記述が可能になるようにし、より複雑なセンサデータを用いた学習アルゴリズム記述等の処理についてはプログラム内で記述するようにした。

SANML での記述可能な制御ルールは、組合せの設定だけでなく、ルールの設定やデータの受け渡しにおけるデータ処理まで可能である。たとえば、複数のセンサとアクチュエータを羅列することにより協調設定の記述が可能となる。ルール設定では、センサでのセンサイベント発生に関するルール設定や、アクチュエータの制御の前段階で利用する条件分岐等の設定が可能である。これらの記述により、SANML を用いて、センサアクチュエータネットワークで活用される代表的なデータ配送モデルであるイベントドリブンモデルとポーリングモデルの両方も実現可能になった。また表 5 に示すように、関連マークアップ言語研究では、センサアクチュエータネットワークに必要な組合せ設定、データ処理設定、ルール設定の 3 種類を満たした先行研究はなかった。USDL<sup>9)</sup> はセンサや情報家電に、また InterPlay<sup>4)</sup> は情報家電に特化しており、ルール設定の記述力を有していない。また、汎

用的なルール設定言語である ECPAP<sup>16)</sup> では、厳密なルール設定は可能なものの、データ処理設定等のセンサとアクチュエータを連携する機能に欠けている。

## 6.3 定性的評価：ミドルウェアについての考察

Spinning Sensors NW ミドルウェアでは、センサやアクチュエータにおいて抽象クラスを提供し、ネットワーク上に分散して存在するヘテロジニアスなセンサやアクチュエータを取り扱えるようにしたことに、従来のホモジニアスなセンサネットワークと比較した際の優位性がある。また、新たにフュージョンという概念を提案し、各ノードがネットワーク上で分散して配置されても動作することで、負荷分散に備えるだけでなく、物理的に離れた位置に存在するセンサやアクチュエータの協調動作を実現できる。制御ルールについては、SANML を用いて組合せ、ルール、データ処理の各設定を外部ファイルとして実現し、複雑なヘテロジニアスセンサアクチュエータネットワークアプリケーションの設定を簡便にできるようにした。

先行研究である Spinning Sensors<sup>11)</sup> と本論文の差異は、まず、先行研究ではセンサがアクチュエータ上に接着され、センサ自身がアクチュエータにより移動可能になる単一ノードを想定していた。これに対して本論文では、センサとアクチュエータが別ノードとして、かつネットワーク経由で協調する環境を想定している。このネットワーク対応のために、Spinning Sensors NW ミドルウェアでは、分散マルチスレッドシステムとして実装されている。さらに先行研究ではセンサとアクチュエータの組合せや制御ルールはプログラム内に記述する方式だったところを、本論文ではより複雑化するそれら設定を、SANML として外部化したところに特徴がある。

本論文で未対応なセンサやアクチュエータの検索・発見機能や、ネットワークの運用にネットワークアドレス変換機能 (NAT) を用いている環境への対応については、ユビキタスサービス発見プロトコル<sup>17)</sup> や NAT 対応のセンサ共有フレームワーク<sup>9)</sup> 等の先行研究を利用していく。

## 7. 関連研究

ネットワーク上の分散コンポーネントを統合するミドルウェアを紹介する。まず、ECPAP<sup>16)</sup>ではイベント駆動型プログラミングの記述規則である ECA (event-condition-action) 規則を拡張した ECPAP (event-condition-precondition-action-postcondition) 規則を提案し、同ルールにおける複数イベントの発生を防ぐイベント駆動型プログラミングを構築した。また、InterPlay<sup>4)</sup>では特に情報家電機器を対象とし、擬似センテンスを用いた協調制御を実現している。これらの先行研究では発生しているイベントの解析や、簡便な協調制御用のインタフェースの提供は実現されているものの、対象が汎用的すぎたり、情報家電機器に特化したりしており、センサやアクチュエータに特有な多様性や散発性への対応や、またセンサとアクチュエータ間のデータ処理への対応ができない。

センサネットワークのアプリケーション構築キットとしては、SNACK<sup>18)</sup>がある。SNACKでは、Crossbow社のMOTE<sup>6)</sup>をターゲットとして、無線センサネットワーク環境で特に重要なメモリリソースや電源リソースの低減のために、コンフィギュレーション言語を用意している。SNACKでは、MOTEでの開発言語であるNesCの非効率さに着目しており、本論文のように多種のセンサやアクチュエータを扱うことはできない。また、ロボット用ミドルウェアとしては、ロボット制御に関するソフトウェアのモジュール化を推進するRT Middleware<sup>19)</sup>がある。RT MiddlewareはCORBAをベースにした分散コンポーネント技術であるが、使用するにあたってCORBAの動作環境を構築する手間がかかる。

センサネットワークを対象とした宣言的センサネットワーク技術として、DSN<sup>10)</sup>やDeclarative Networking<sup>20)</sup>等がある。これらはセンサデータの取得方法として、宣言的なSQL言語を用いることで、より分かりやすいセンサネットワーク管理を実現している。しかしながら、MOTEを対象とした実装であり多種多様なセンサアクチュエータに対応しておらず、またデータの受け渡しに関してリアルタイム性の考慮もなされていない。

## 8. まとめ

本論文では、既存のセンサアクチュエータネットワークシステムにおける異種分散ノード協調問題と協調動作記述設定問題について提示した。そしてこれらを解決するセンサアクチュエータネットワーク機構であるSpinning Sensors NWミドルウェアと制御ルール記述フォーマットであるSANMLを提案し、その有効性を示した。

Spinning Sensors NWとSANMLを使用することで、ネットワーク上に別ノードとして

存在する複数のセンサノードとアクチュエータノードをフュージョンとして設定し、センサアクチュエータネットワークアプリケーションを実現できる。本論文ではまずセンサアクチュエータネットワーク環境での問題点を整理した。そして、センサとアクチュエータの協調制御を記述するためのマークアップ言語であるSANMLを提案した。SANMLを用いることで、プログラミング言語でのセンサアクチュエータアプリケーション構築に比べて、より簡便にそれらアプリケーションを記述できることを示した。また、SANMLを設定ファイルとして読み込み、実際にセンサアクチュエータネットワークアプリケーションを構築するSpinning Sensors NWミドルウェアを提案した。本ミドルウェアにより、ネットワークに接続されたセンサとアクチュエータを組み合わせたアプリケーションを実現できることを示した。また評価としてその実行速度が現実的な時間内で行われていること、SANMLの記述力について、センサアクチュエータネットワーク環境に適しており、先行研究と比較して独自性があることを示した。

今後の課題としては、より大規模かつ高密度なセンサアクチュエータネットワークアプリケーションにおける実証実験や、センサアクチュエータに限らず情報家電機器やウェブサービス等の各種コピキタスサービスとの連携等を実現したい。

## 参考文献

- 1) LaMarca, A., Brunette, W., Koizumi, D., Lease, M., Sigurdsson, S., Sikorski, K., Fox, D. and Borriello, G.: Making Sensor Network Practical with Robotics, *Proc. 1st International Conference of Pervasive Computing* (Aug. 2002).
- 2) Nakazawa, J., Tokuda, H., Edwards, W.K. and Ramachandran, U.: A bridging framework for universal interoperability in pervasive systems, *The 26th IEEE International Conference on Distributed Computing Systems* (2006).
- 3) Iwai, M. and Tokuda, H.: Evaluation of a robust middleware for enormous distributed task handling, *The 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications* (2005).
- 4) Messer, A., Kunjithapatham, A., Sheshagiri, M., Song, H., Kumar, P., Nguyen, P. and Yi, K.H.: Interplay: A middleware for seamless device integration and task orchestration in a networked home, *IEEE International Conference on Pervasive Computing and Communications*, Vol.0, pp.296-307 (2006).
- 5) Sun Microsystems, Inc.: Jini Architecture Specification Version1.1 (2000). <http://www.sun.com/jini/specs/jini1.1.pdf>
- 6) Hill, J. and Culler, D.: Mica: A wireless platform for deeply embedded networks, *IEEE Micro*, Vol.22, No.6, pp.12-24 (2002).

- 7) Kumar, R., Wolenetz, M., Agarwalla, B., Shin, J., Hutto, P., Paul, A. and Ramachandran, U.: Dfuse: A framework for distributed data fusion, *The 1st ACM Conference on Embedded Networked Sensor Systems (Sensys)* (2003).
- 8) Li, S., Son, S. and Stankovic, J.: Event detection services using data service middleware in distributed sensor networks, *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN)* (2003).
- 9) Nakazawa, J. and Tokuda, H.: A middleware framework for sharing sensor nodes among smart spaces, *4th International Conference on Networked Sensing Systems, 2007. INSS'07*, pp.171–178 (June 2007).
- 10) Chu, D., Popa, L., Tavakoli, A., Hellerstein, J.M., Levis, P., Shenker, S. and Stoica, I.: The design and implementation of a declarative sensor network system, *Sensys'07: Proc. 5th international conference on Embedded networked sensor systems*, New York, NY, USA, pp.175–188, ACM (2007).
- 11) Aoki, S., Nakazawa, J. and Tokuda, H.: Spinning sensors: A middleware for robotic sensor nodes with spatiotemporal models, *The 14th International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA)* (2008).
- 12) Gosling, J., Joy, B. and Steele, G.: The java language specification (1999).
- 13) Beigl, M., Decker, C., Krohn, A., Riedel, T. and Zimmer, T.: uParts: Low Cost Sensor Networks at Scale, *Demonstration Proceedings of 7th International Conference of Ubiquitous Computing* (Sep. 2005).
- 14) LEGO Group: LEGO Mindstorms NXT (2006). <http://mindstorms.lego.com/>
- 15) Phidgets Inc.: Phidgets Analog Sensors and Servo Motors (2003). <http://www.phidgets.com/>
- 16) Shankar, C. and Campbell, R.: Ordering management actions in pervasive systems using specification-enhanced policies, *PERCOM'06: Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications*, pp.234–238, IEEE Computer Society (2006).
- 17) Lee, C. and Helal, S.: A multi-tier ubiquitous service discovery protocol for mobile clients, *1st ACM International Conference on Mobile Systems and Applications*, pp.701–711 (May 2003).
- 18) Greenstein, B., Kohler, E. and Estrin, D.: A sensor network application construction kit (snack), *The 2nd ACM Conference on Embedded Networked Sensor Systems (Sensys)* (2004).
- 19) Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T. and Yoon, W.: Rt-component object model in rt-middleware — distributed component middleware for rt (robot technology), *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)* (2005).
- 20) Loo, B.T., Condie, T., Hellerstein, J.M., Maniatis, P., Roscoe, T. and Stoica, I.:

Implementing declarative overlays, *SIGOPS Oper. Syst. Rev.*, Vol.39, No.5, pp.75–90 (2005).

(平成 20 年 10 月 3 日受付)

(平成 21 年 4 月 15 日採録)



青木 崇行 (正会員)

2003 年慶應義塾大学大学院政策・メディア研究科修士。2003 年から 2006 年にかけてソニー株式会社にて画像信号処理の研究に従事。2009 年慶應義塾大学大学院政策・メディア研究科博士課程単位取得退学。現在、慶應義塾大学大学院政策・メディア研究科特別研究助教。主に、センサネットワーク、ロボティックセンサノード、ユビキタスシステム、画像処理等の研究に従事。



桐原 幸彦

2003 年慶應義塾大学大学院政策・メディア研究科修士。同年ソニー株式会社入社。ミドルウェアの研究に従事。2006 年に株式会社トリプルダブル設立。主に、センサネットワーク、ミドルウェア、携帯用ソフトウェア等の研究に従事。



中澤 仁 (正会員)

2000 年慶應義塾大学大学院政策・メディア研究科修士。2003 年慶應義塾大学大学院政策・メディア研究科博士。2004 年から 2005 年にかけてジョージア工科大学訪問研究員。現在、慶應義塾大学大学院政策・メディア研究科講師。主に、分散オブジェクト指向システム、オブジェクト移送ミドルウェア、ホームネットワーク等の研究に従事。



高汐 一紀 (正会員)

1992年慶應義塾大学大学院工学研究科修士(工学). 1995年慶應義塾大学大学院工学研究科博士(工学). 電気通信大学助手, 慶應義塾大学大学院政策・メディア研究科助教授を経て, 現在, 慶應義塾大学環境情報学部准教授. 主に, 分散システム, 実時間システム, ユビキタスシステム等の研究に従事. IEEE, ACM, 日本ソフトウェア科学会各会員.



徳田 英幸 (正会員)

1977年慶應義塾大学大学院工学研究科修士. 1983年ウォータールー大学 Ph.D. (Computer Science). 同年カーネギーメロン大学計算機科学科勤務. 1990年同学科研究准教授. 現在, 慶應義塾大学環境情報学部学部長. 主に, 分散リアルタイムシステム, マルチメディアシステム, 超並列・超分散システム, ユビキタスシステム等の研究に従事. IEEE, ACM, 日本ソフトウェア科学会各会員.

---

日本ソフトウェア科学会各会員.