卒業論文　　2007年度（平成19年度）

# センサネットワークへの<br>透過的なアクセス機構の設計と実装

慶應義塾大学 環境情報学部

佐藤　龍

tatsurou@sfc.wide.ad.jp

指導教員

慶應義塾大学 環境情報学部

村井　純

徳田　英幸

楠本　博之

中村　修

高汐　一紀

重近　範行

湧川　隆次

Rodney　D.　Van　Meter　III

平成20年1月24日

卒業論文要旨　　2007 年度（平成 19 年度）

# センサネットワークへの透過的なアクセス機構の設計と実装

論文要旨

センサネットワークは，センサを用いて計測した実空間のある現象や状態のデータを伝播し，利用者へ提供するための通信ネットワークである．近年，センサ製造技術の向上により一般生活でのセンサネットワークの活用が可能となりつつある．そのため，センサデータを利用するシステムの開発者は，日常生活での多様な要求を満たすアプリケーションの開発が望まれている．しかし，センサネットワークを通じてセンサデータを利用するには，センサの仕様に応じてデータをリクエストした上で，多様なデータフォーマットを解釈しなければならない．そのため，実空間情報を利用するアプリケーションを実装する場合，利用するセンサネットワークの構成に依存した方法でしか利用できない．この問題には二つの要因が関係している．

一つ目の要因はデータ通信方式がセンサによって異なることである．データ通信方式がセンサによって異なると，センサデータを取得するために実装する必要のある処理が通信方式ごとに増える．さらに，差異を吸収するための処理も実装しなくてはならない．二つ目の要因はセンサデータフォーマットが統一されていないことである．データフォーマットがセンサの種類によって異なると，フォーマットを解釈する実装負荷が増えるのと同時に，使用するセンサを別の種類に変更する度に，新たなセンサに実装を合わせなくてはならない．これらの変更を全て，開発者がシステムごとに行うのは実装負荷が大変高い．

この問題を解決するプロキシモデルを取り入れたプロキシノードを設計，実装した．プロキシノードはアプリケーションによるセンサネットワークへのアクセスを代替し，センサデータフォーマットを共通の形式に変換して提供する機能を持つ．また，複数のアプリケーションからの多様な要求を集約してセンサネットワークへ伝える．そのため，プロキシノードを利用すると，センサネットワークを利用できるアプリケーション数が増加して効率が良い．

実装効率を検証するため，MicaMote と µPart のセンサで構成される二種類のセンサネットワークを利用するサンプルアプリケーションを実装し，プロキシノードの使用有無で場合を分け，実装効率を比較した．検証結果より，本実装を利用するとアプリケーションを少ないコストで実装でき，センサデータの利用効率を上げられることを評価した．

キーワード
1．センサネットワーク，2．プロキシ，3．実空間情報

慶應義塾大学 環境情報学部
佐藤 龍

**Abstract of Bachelor's Thesis**

Academic Year 2007

# The Design and Implementation of a Transparent Access Scheme for Sensor Networks

Summary:

A sensor network is a communication network to measure information of an environment and state of objects in real space, to distribute and offer sensor data provided by some sensors. Needs for activity who support, assist, and cover for a human using information of real space have become prominent and become essential to our life. Thus, developers of such kind of system are expected to use information on sensors to fulfill these expectations. However, it is necessary to request a query according to the specification of a sensor, then, decode various data formats to use real space information through the sensor network, thus it is difficult to use it effectively. This problem caused by two issues.

The first issue is the difference of communication methods between sensor implementations. The difference of communication methods forces developers to implement application for each protocol to acquire sensor data, to absorb the difference. The second issue is lack of generalized sensor data format. It is necessary to implement a format decoder when introducing another kind of sensor implementation. Developer is making these changes for each system, cause high implementation load.

To absorb these issues, we propose a proxy model. In this thesis, we designed a server program to adopt proxy model, and describe about a proxy node as implementation deliverable. Proxy node translate access to sensor network with application, and send sensor data with standardized format to application. In particular, proxy node aggregate requests from some applications. Thus, applications can utilize sensor networks even with multiple applications.

To evaluate effects of reducing implementation costs, we used two kinds of sensor networks, composed of MicaMote and µPart , and implemented application using them as a prototype. As a result, we belive that the proxy node is effective to reduce implementation costs.

Keywords:
1. Sensor Network, 2. Proxy, 3. Real Space Information

Keio University, Faculty of Environmental Information
Ryu Sato

# 目次

# 図目次

# 表目次

# 第1章　序論

本章では，本研究が焦点を当てる実空間情報の活用の事例を述べ，本論文が前提とする環境を概説するとともに，本論文の目的を述べる．

## 1.1　背景

本研究では，気象情報などの環境に関する情報や，環境内に存在する人や物に関するあらゆる情報を実空間情報と定義している．実空間情報を用いることで，現実に起こる出来事の状態の把握を必要とする活動をサポートできる．そのため，実空間情報は人の健康管理や行動の補助，商品の在庫数確認，動物の生態調査など幅広い分野での活用が期待されている．特に，人の健康管理や行動の補助など，日常生活に密接に関わる分野への活用は，新しい活用分野として強い関心が集まっている．

実空間情報の要素となる情報を取得するための機器をセンサと呼ぶ．センサは温度や照度，湿度などの物理的な値を，デジタル数値データに変換して出力できる．センサは，デバイス製造技術の進歩によって，小型化，低価格化，無線化し，設置しやすくなりつつある．また既存の研究では，複数のセンサを使って室内や構内の実空間情報を取得する活動が行われている [1, 2]．デバイス技術の進歩と既存の研究活動により，特定の環境内で実空間情報を取得する環境が整ってきたと言える．

センサを使うシステムで実空間情報を解析するには，複数の種類のセンサを多数利用する方法が効果的であると考えられる．複数のセンサを用いると，一つのセンサを用いる場合に比べて，実空間情報の詳細な解析に繋がる．センサはデバイスの周囲の物理量を数値化する．そのため，センサデータはセンサの近傍の情報を示す．センサを用いた観測では地理的な範囲が狭い．例えば，観測範囲の広い室内や構内などの実空間情報を取得するためには距離をあけて複数のセンサを設置し，全てのセンサデータの平均値を取る必要などがある．実空間情報は，空間内に設置されたセンサの密度が高い程に，少ない誤差で判断するのに役立つ．そのため，高精度に実空間情報を判断するには，複数のセンサが必要であると考えられる．

また，多種類のセンサを用いると，単一種類のセンサを用いる場合に比べて，実空間情報の詳細な解析に繋がる．例えば，温度センサを使うと人の体温が分かるため，健康管理に利用することができる．温度センサは，運動による一時的な高温と体調悪化による高温の違いを判断できない．この違いは，加速度センサを用いた運動状態の把握により，判断可能な場合があ

る．複数種類のセンサデータを合わせて用いる程，実空間情報を詳細に判断できる可能性が広がる．そのため，複数の種類のセンサを多数利用する方法は，実空間情報を高精度に把握するのに必要であると考えられる．

しかし，既存のセンサシステムは，センサデータの取得方式が統一的でない．複数種類のセンサデータを統合して用いることが困難である．この問題はセンサデータの取得方式が，センサシステムの構成に制限されていることに起因する．単一種類のセンサで多種類のセンサデータを統合する手段により，この問題は解決される可能性がある．しかし，広域の情報を知る上で，観測範囲とセンサの製造コスト，構成の柔軟性の観点から実現は難しいと言える．そのため，多種類のセンサネットワークを統合して利用するセンサシステムが効率的であると考えられる．

現状では，センサデータのフォーマットはセンサの種類によって異なり，すべてのセンサに共通するフォーマット規格は存在しない．一部のセンサシステムで共通に採用される規格は存在するものの，センサデータのフォーマットは製造ベンダごとに異なる．そのため，センサシステムのアプリケーション開発者に実装負担が大きくなっていると言える．

## 1.2 目的

本研究の目的は，多種類のセンサネットワークを統合的に用いるシステムにおいて，アプリケーション開発者の実装負担を軽減することである．現状ではセンサシステムにおけるアプリケーションを容易に実装できない．この問題は多種類のセンサデータを統一的に取得できる機構がないことに起因する．アプリケーション開発者は，センサデータ取得処理をそれぞれ実装する必要がある．その負担は，センサデータを統一的に取得できないことから，複雑な要求になるほど増加すると考えられる．さらに，取得したセンサデータをセンサの仕様に応じて解釈する必要があるため，取得したセンサデータを統合させるのが困難である．結果，システムで利用するセンサの構成を変更するには，構成に依存するアプリケーションに新たな実装が必要となり，非効率である．

本研究では，アプリケーション開発者の実装負担を軽減させるために，次の二点を解決すべき課題とする．

- アプリケーションによるセンサデータアクセス方式の統一
- 利用するセンサネットワーク構成の柔軟性の実現

## 1.3 用語定義

本論文で使用する用語の定義を以下に示す.

**実空間情報**
: 実世界における，気象情報などの環境の情報や人や物に関するあらゆる情報

**センサデバイス**
: 実世界における一種類の情報を計測するハードウェアデバイス

**センサデータ**
: 実世界における物理量などをセンサデバイスにより計測したデータ

**センサノード**
: 一つまたは複数のセンサデバイスで構成され，センサデータの送信機能を有するハードウェア機器

**センサネットワーク**
: 複数のセンサノードで構成されるネットワーク

**センサシステム**
: センサネットワークと，センサネットワークを利用するアプリケーションを含むシステム

## 1.4 論文の構成

本論文は全 10 章で構成される．まず，第 2 章では本研究の背景を述べる．第 3 章では本研究の目標とするセンサデータの利用環境を述べる．第 4 章ではセンサデータの利用環境の現状を述べる．現状と目標の利用環境を分析・比較することで，問題点を明らかにする．

第 5 章では本研究と関連する研究を挙げる．第 6 章では第 4 章で整理した現状の問題点を解決するためのアプローチを示し，実現性を議論する．第 7 章では第 6 章で示したアプローチにしたがった設計を示し，第 8 章ではシステムの実装を示す．第 9 章では既存のシステムとの比較によって評価する．設計に従って実装した本実装を，実際のセンサネットワークに適用した結果より本研究の有効性を示す．最後に，第 10 章で本論文の結論を述べる．

# 第2章　実空間情報の利用とセンサシステム

本章では本研究の背景である，実空間情報の解析とセンサデータ取得の関係を述べる．センサデータを使った実空間情報の解析はセンサシステムの構成と密接に関わっている．センサシステムの構成を分類し，課題解決のために有効な構成を考察する．

## 2.1　実空間情報

実空間情報を利用するには多種類のセンサデータが必要であると言える．本章では，実空間情報を解析するための情報処理の流れと利用例により，多種類のセンサデータの必要性を示す．

### 2.1.1　実空間情報の解析

実空間情報を利用すると，利用者は自身に有益な活動を行うための基本行為となる現状の把握を，コンピュータなどの計算機によって自動化できるため有益である．センサを用いて実空間情報の解析を行う仕組みを**センサシステム**と呼ぶ．

センサシステムは，センサネットワークと，センサネットワークを利用するアプリケーションの総称である．センサネットワークの詳細は第 2.2 節で述べる．図 2.1 にセンサシステムの構成を示す．図 2.1 における ADC(Analog to Digital Converter) はアナログ信号をデジタル信号に変換する回路である．

### 2.1.2　実空間情報の利用例

センサデータの精度はセンサの特性によって決まる．一方で，一種類のセンサデータを単に用いるだけではなく，複数の種類のセンサデータを組み合わせることにより，より高い精度で状態を特定できる可能性が広がる．そのため，利用者の要求する状態を特定するためには複数種類のセンサデータを用いるのが一般的である．多種類のセンサデータを使うシステムの例を表 2.1 に示す．ある人が寝ているのかどうかを知りたい場合など，体温が低下し，動きが鈍くなり（加速度がゼロに近い），血糖値が低下することから総合的に判断することができる．多種類のセンサデータを解釈して利用することで，一種類のセンサデータからでは感知することのできない，高度な判断を行うことができると言える．

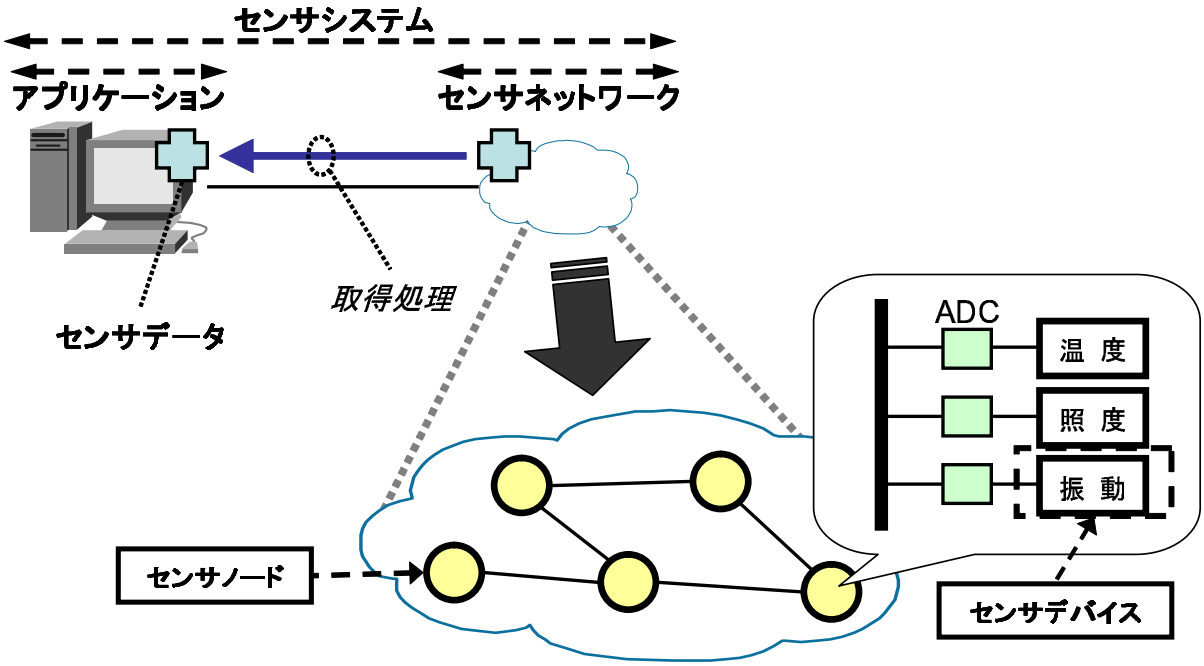


図 2.1　センサシステムの構成

表 2.1　実空間情報利用例

| 利用用途のジャンル | 利用例 | 使用センサ |
|---|---|---|
| 軍事利用 | Smart Dust [3] | その他センサ，バイタルセンサ |
| 人の健康管理 | CFH (Center for Future Health) [4] | バイタルセンサ |
| 動物の生態管理 | Great Duck Island [5] | バイタルセンサ |
| 物の状態管理 | Media Cup [6] | 一般感知センサ |
| 環境の状況管理 | Live E! [7] | 一般感知センサ |
| 人の行動確認 | Smart Kindergarten [8] | バイタルセンサ，一般感知センサ |

### 2.1.3　実空間情報の解析処理

実空間情報は多種類のセンサデータから解析できる．センサシステムにおいて，センサデータから実空間情報を解析するまでの情報処理の流れを図 2.2 に示す．実世界における物理量やエネルギ量を情報処理可能な値に変換し，その値を複数種類統合した結果をセンサデータと呼ぶ．

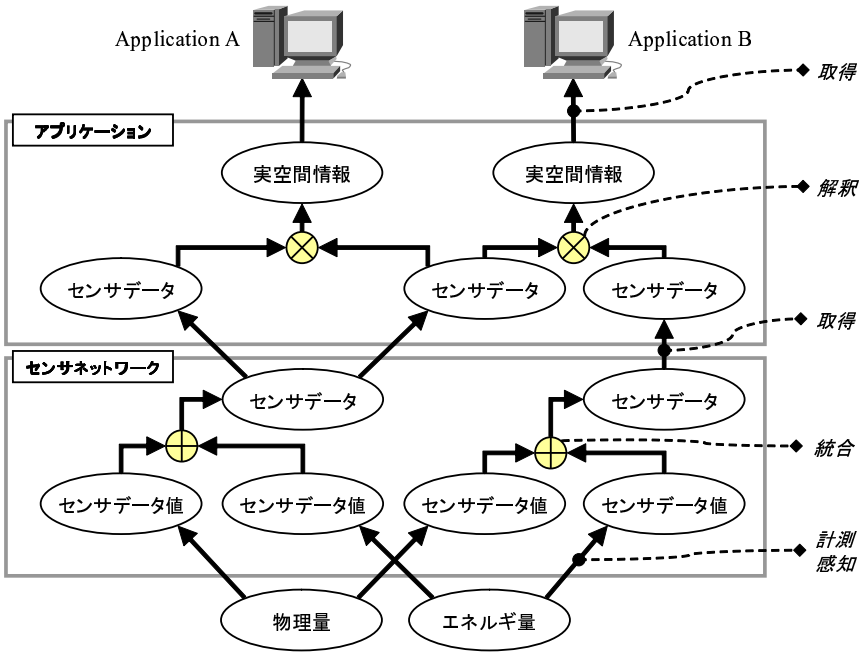


図 2.2　センサデータの統合利用による実空間情報の解析

### 2.1.4　センサとセンサデータ

センサはセンサデバイスを持つ機器である．センサの具体例として，Mote シリーズ（MicaMote，Mica2Dot，MICAz，IntelMote） [9]，µPart [10]，Smart Its [11]，$U^3$ [12] がある．

センサは物理量を計測，感知し，デジタル数値に変換する．計測，感知する対象の化学物質やエネルギの名称をセンサデータの種類と呼ぶ．センサが変換した数値をセンサデータと呼ぶ．センサデータの種類が温度であるセンサがセンサデータ値を計測する例を図 2.3 に示す．センサは設計限度により全ての変化を計測することはできないため，計測できる値は特定の値域内に制限される．図 2.3 の例では −10℃ から 30℃ までが計測値域である．計測した値は特定の精度でデジタルの数値に変換され，センサデータとなる．例では 20℃，30℃，20℃，29℃ の順に計測した温度が，値が 191，255，191，248 であるセンサデータに変換されている．これらの値域と精度はセンサの特性によって定まる．

センサデータの種類はセンサが感知する対象によって異なるため，多様な種類が存在する．表 2.2 にセンサの感知対象におけるセンサデータの種類を示す．一般感知センサは室内や構内などの空間を計測するために，物や壁に設置して利用する．また，バイタルセンサは人の状態

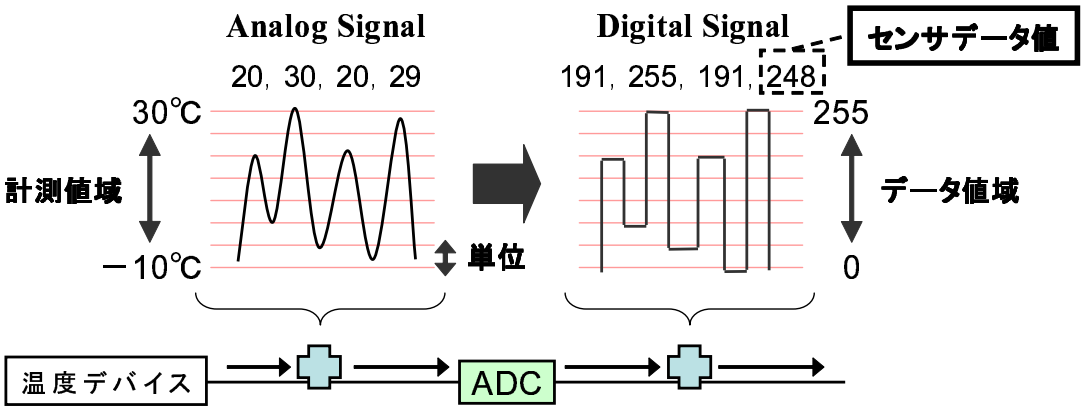


図 2.3　温度デバイスによるアナログデータからデジタルデータへの変換例

を計測することが目的であり，人が持ち運んで利用する場合がある．他にも，有毒ガスが発生しているかなどを感知するために，政策や産業利用といった特殊な目的のために利用するセンサがある．

表 2.2 から，センサは人につけるか空間の設置するかの違いがあることと，センサデータの種類が数多く存在することが分かる．例えば，温度センサは人につけると体温を示すが，空間に設置すると気温を示す．センサデータ種類は数多く存在し，実世界に存在する化学物質や物理現象の数と同じ数だけ存在しうるため，あらゆる計測を行える万能なセンサは存在しない．

表 2.2　センサの感知対象におけるセンサデータ種類

| 感知対象 | センサデータの種類 |
|---|---|
| 一般感知センサ | 温度，照度，湿度，圧力，加速度，方位，位置，人感 など |
| バイタルセンサ | 体温，血圧，脈拍，心拍数，血糖値，心電，筋電 など |
| その他 | 有毒ガス，地雷（金属感知） など |

## 2.2　センサネットワーク

センサネットワークは複数のセンサノードにより構成されるネットワークで，物理的な接続構成は大きく分けて二種類ある．一つ目は有線を用いてセンサノード間を接続したネットワークであり，主に家やビルなどの建造物で利用される構成である．二つ目は無線技術を用いてセンサノード間の通信を可能にするネットワークであり，有線での設置が困難な場所で利用される構成である．無線で通信するセンサネットワークを特にワイヤレスセンサネットワークと呼

ぶ．センサネットワークの構成における利点と欠点を表 2.3 に示す．表 2.3 から，センサネットワークの使用場所に応じて，センサネットワークの構成を使い分ける必要があることが分かる．そのため，一つのセンサネットワークは広域でなく，ある範囲内に構成するセンサノードが設置されていると言える．

表 2.3　センサの接続における利点と欠点

| 接続方式 | 利点 | 欠点 |
|---|---|---|
| 有線接続 | 安定した通信 | 配線が困難，センサ移動不可能 |
| 無線接続 | 配線不要，センサ移動可能 | 電源供給不安，無線ノイズの影響大 |

## 2.3　センサネットワークへのアクセス

本節では，センサネットワークへのアクセス方法について，データ通信方式，データ指定方式，データ流通モデル，ネットワーク構成の四点から考察し整理する．

センサネットワークから得られるセンサデータが示す情報は，地理的に広域な範囲でないことを第 2.2 節で述べた．しかし一方で，利用者は計測した場所とは物理的に遠く離れた場所から，PC や携帯電話などの端末を使ってセンサデータを知りたいという要求がある．この要求に対応するため，センサデータを取得できる場所を物理的な制限から開放できるインターネットが使われている．本研究ではインターネットを通じたセンサデータの取得環境を想定する．

### 2.3.1　データ通信方式

アプリケーションとセンサネットワークとの通信方式を以下に整理して示す．

**プル型**　同期通信方式．
　データを要求してから返答を受信するまでが同期的に行われる通信方式である．要求した側は返答を受信するまで待機する．

**プッシュ型**　非同期通信方式．
　データを順次要求しない通信方式である．要求の設定後，または通信の開始後にセンサネットワークから一方的にデータが送信される．

**イベントドリブン型**　非同期通信方式．
　データの要求から返答の受信までが非同期に行われる通信方式である．要求時に取得するデータの条件を指定でき，条件に合うデータのみを選択して送信される．

センサネットワークは三つの通信方式のうち，全てまたはいずれかの通信方式を利用することができる．しかし，すべてのセンサネットワークが三つの通信方式すべてに対応しているわけではない．例えば，XBridge と uPart で構成されるセンサネットワークはプッシュ型の通信方式のみに対応し，残る二つの通信方式には対応していない．この理由の一つとして，XBridge へセンサデータを集める µPart の計算資源が乏しく，アプリケーションからの要求に応じたセンサデータを送信できないことが関係している．

一方，センサデータの利用目的に応じて，アプリケーション実装効率の良い通信方式が異なる場合が多い．そのため，センサネットワークが対応していない通信方式を途中で解釈しなおすことで提供できれば，センサネットワークの利用効率が良くなることが見込まれる．

### 2.3.2 データ指定方式

アプリケーションがセンサデータを指定する方法は大きく分けて二つある．一つ目はノードセントリック方式であり，二つ目はデータセントリック方式である．以下にそれぞれの指定方式の特徴をまとめる．

**ノードセントリック方式**　対象ノードを識別子で指定する要求方式．

対象センサノードを指定する方式であるため，センサノードが持つセンサデバイスを把握しなければ，取得できるセンサデータの種類を判断することができない．センサ ID は同一種類のセンサネットワークでのみ共有されるため，標準規格を持たないセンサシステムを統合的に利用する目的では使えない．ID の指定方法が異なるセンサシステムを利用するためには，ID が重ならないように，ID 以外の識別子を利用する必要などがある．

**データセントリック方式**　センサデータ値の値域やセンサデータの種類を指定する要求方式．

識別子以外のデータを指定することでセンサデータを要求する．センサノードが持つセンサデバイスを把握しなくてよく，アプリケーションは ID の異なるシステムにおいても，センサデータを要求できるのが特徴である．指定する指標の例として以下のデータが挙げられる．

- センサデータ値
- センサデータの種類
- センサデータを生成した時刻
- センサノードの置かれた場所

ノードセントリック方式は ID 空間の制約から，センサシステムを統合的に利用する目的には不向きである．例えば，TinyDB は 8bit 分の ID によってセンサノードを区別し，µPart は 64bit 分の ID によってセンサノードを区別するため，ID が重なる可能性がある．特に，TinyDB によって二つのセンサネットワークを構成する場合，ID が重なる可能性が高いため，ID 空間を拡張するための仕組みが必要になるが，TinyDB にはその仕組みがない．同様の理由でノードセントリック方式を用いる研究は，ID 空間の拡張を考慮している場合は少ない．

一方で，複数種類のセンサネットワークを利用するヘテロジーニアスなセンサシステム構成では，利用しているセンサの種類による影響が少ない点で，データセントリック方式が適していると言える．異なる種類のセンサデータを区別せずに，値だけを利用することはできないため，センサデータの種類は少なくとも指定できる必要がある．種類を指定できるようになれば，アプリケーションは使用しているセンサの種類に影響せず，センサデータを取得することができる．

### 2.3.3 センサデータ流通モデル

センサネットワークのデータ流通モデルは主に二種類ある．一つ目はシンクノードモデルである．シンクノードモデルでは，センサデータはシンクノードへ集められる．二つ目は P2P モデルである．P2P モデルでは，全てのセンサノードが対等の役割を持ち，センサデータを自律分散的に保持する．二種類のモデルのネットワーク構成を図 2.4 と図 2.5 に示し，特徴を以下に示す．

**シンクノードモデル**　センサデータを特定のセンサノードへ集めるモデル．

センサノードは生成したセンサデータをシンクノードへ送る．集めたセンサデータはシンクノードを経由して提供されるため，センサデータを利用する方法に応じて，複数のアプリケーションが共有できる．一方で，センサノードの数とアプリケーションの数に応じて シンクノードの負荷は増加する．

**P2P モデル**　全てのセンサノードが対等の役割を持ち，センサデータを自律分散的に保持するモデル．

アプリケーションは各センサノードが持つ記憶領域に保持される．センサデータを流通させる動作に，特別な役割を持つセンサノードの存在を前提としないため，シンクノードモデルに比べて冗長性があるのが特徴である．

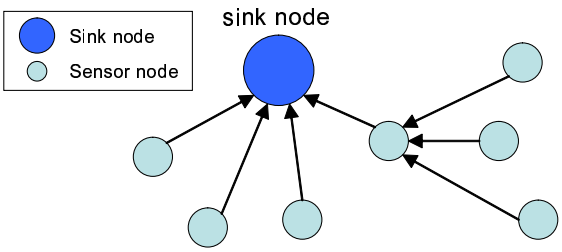


図 2.4　シンクノードモデル

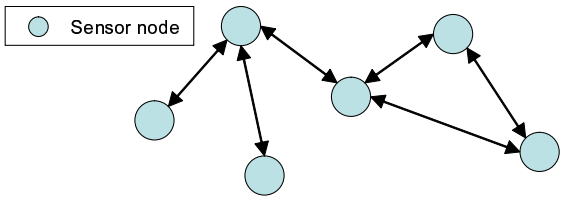


図 2.5　P2P モデル

### 2.3.4　ネットワーク構成

センサシステムのアプリケーションは，多種類のセンサデータを取得し，統合して高度な実空間情報を把握するソフトウェアである．多種類のセンサデータを取得するためには，センサネットワークを通じて多種類のセンサからデータを取得する必要がある．センサネットワークのネットワーク構成によって，センサデータを取得する方法を分類できる．一つ目は，多種類のセンサデバイスを持つセンサノードでセンサネットワークを構成し，そのネットワークからセンサデータを取得する場合である．このネットワーク構成を**ホモジーニアスノード型ネットワーク構成**と定義する．二つ目は，一つ，または複数種類のセンサデバイスを持つセンサによってセンサネットワークを構成し，それぞれからセンサデータを取得する場合である．このネットワーク構成を**ヘテロジーニアスノード型ネットワーク構成**と定義する．二つの場合において，センサシステムのアプリケーションを構築する時の利点と欠点を表 2.4 に示す．表 2.4 から，センサデータ取得環境はセンサネットワークのネットワーク構成に依存していることが分かる．センサシステムのアプリケーションを容易に実装できる機構を実現するためには，ネットワーク構成を考慮する必要がある．

表 2.4　ネットワーク構成の違いによるアプリケーション実装の利点と欠点

| ネットワーク構成 | 利点 | 欠点 |
|---|---|---|
| ホモジーニアスノード型 | 統一アクセス方式が利用可能，<br>データフォーマット解釈が容易 | センサ構成変更が困難 |
| ヘテロジーニアスノード型 | 構成変更が比較的容易 | 複数アクセス方式が必要，<br>複数データフォーマット解釈が必要 |

# 第3章　実現したい世界

センサデータを利用する研究の多くが複数種類のセンサデータを利用している現状と，インターネットを通じたセンサネットワークアクセスが効率的であることを第 2 章で述べた．本章では本研究の目標とするセンサデータの利用環境を述べる．図 3.1 に，本研究が目標とするセンサデータ利用環境の概要を示す．

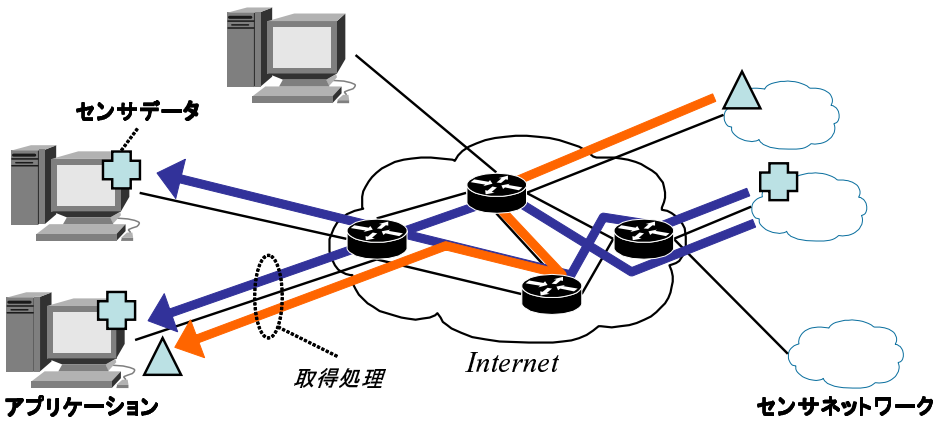


図 3.1　本研究が想定するセンサシステム構成の概要

## 3.1　統合利用可能なセンサデータ取得環境

本研究では，多様なセンサネットワークの統合利用を可能にし，センサデータを利用するアプリケーションを効率よく実装できる仕組みの実現を目標とする．本節では，異なるセンサネットワークを統合することにより実現できる利用環境を考察する．

### 3.1.1　ヘテロジーニアスなセンサネットワークへの対応

単一のセンサネットワークから取得できるセンサデータの種類は少ない場合が多いため，複数種類のセンサデータを取得するためには，複数のセンサネットワークからセンサデータを利用できることが望ましい．さらに，異なる種類のセンサネットワークを利用できれば，センサネットワークの取得方法に依存せずアプリケーションを実装できる．その結果，センサネットワークの構成がアプリケーションを考慮せずにすむとともに，アプリケーションの実装も容易

になると期待できる．

### 3.1.2 統一されたアクセス方式

アプリケーションの実装コストを上げず，センサシステムを構成するセンサの種類や，利用するセンサデータの種類を変更できるようにし，かつ，センサデータ取得のためのアクセス方式が統一的である仕組みが望ましい．アクセス方式が異なると，センサネットワークの仕組みごとにアクセスしなくてはならない．そのため，他のアプリケーションとデータ要求の共用や，センサデータの共有は難しい．

### 3.1.3 統一されたデータフォーマット

センサデータは単位や精度が様々である．さらに，複数種類のセンサデータを取得するためのセンサは，各種類のデータを計測するためのセンサデバイスを持ち，データ転送の効率化のため，それらを結合して一つのセンサデータとしてまとめる．まとまってしまったセンサデータを利用するには，再び各種類のデータに分割する必要があり，どのような種類のセンサデータが，どのように結合されたかを知らなくてはならない．データがどのように結合されたかを示す情報を，ここではデータフォーマットと呼ぶことにする．センサデータフォーマットがシステムごとに異なり，これを統一化する規格も提案されている [13]．しかし，一部のセンサにおいて実現されているものの，多くのセンサシステムにおいて，アプリケーションがフォーマットを解釈しなくてはならないのが現状である．データフォーマットの統一により，アプリケーションの実装コストを下げられると期待できるため，データフォーマットは統一されていることが望ましい．

## 3.2 柔軟なセンサシステム接続構成

センサデバイス技術の発展によって，工業的な利用や政策としての利用から，日常生活で利用するために必要な環境が整いつつある．例えば，会社での仕事の効率を高める目的での利用や，より一般的に，家庭内での生活を支援する目的での利用が想定される．アプリケーションが開発しやすいセンサシステム構成でなくては，将来的な利用活動に対応できないと想定される．多様な活用を支援するとともに，将来的な活用にも対応するためには，センサシステムを容易に利用できる仕組みが必要である．

容易に活用できない要因として，アプリケーションが利用できるセンサの種類が制限されていることがあげられる．この要因により，一度作ったセンサシステムの構成を拡張するのが難

しいと考えられる．そのため，センサシステムにおけるセンサノードの接続構成は柔軟に変更できるのが望ましい．

# 第4章　アプリケーション実装コストとセンサネットワーク管理

本章では現状のセンサシステムにおける問題点を目標環境と比較することで明示する．第 3 章で目標とするセンサデータ利用環境の要件を次の四点にまとめた．

- ヘテロジーニアスなセンサネットワークへの対応
- 統一されたアクセス方式
- 統一されたデータフォーマット
- 柔軟なセンサシステム接続構成

現状では目標環境を実現するための課題として，実装コストの問題とセンサネットワークの利用に必要な情報を管理する必要がある．以下に詳細を述べる．

## 4.1　実装コスト

センサデータを利用するアプリケーションを実装する際に必要となるコストはソースコードを記述する上で開発者が必要となる行動，またその結果を指す．具体的にはソースコードの行数と，ソースコードを記述するために必要となる知識であると考えられる．本章では，具体的に必要となる実装負担を述べる．

### 4.1.1　センサデータの解釈

センサを使うシステムが実空間情報を把握するためには，多種類のセンサを複数利用する方法が効果的であることを第 2 章で述べた．センサを複数利用するためにはセンサデータフォーマットを統一しなければならない．センサデータフォーマットが統一できなければ，利用する全てのセンサデータフォーマットを事前に把握し，アプリケーションで利用しやすい形式に変換するための解釈処理をしなくてはならないため，効率的に実装できない．一部のセンサシステムで共通して使用する規格は存在するが，使用するセンサの種類や，ハードウェア構成に依存しているため，現存するセンサシステムを統一的に扱えない．そのため，既存のセンサを含む複数のセンサデータを組み合わせて利用するのは難しい．

### 4.1.2 利用センサノードの構成変化による再実装

センサシステムにおけるアプリケーションは，使用しているセンサノードの種類に応じて実装する．そのため，使用するセンサノードの新たな追加や，別のセンサノードへの交換により，それに合わせてアプリケーションを再実装しなくてはならない．現状ではアプリケーション実装コストを上げることなく，利用するセンサノード構成を変更するのは難しいといえる．

この問題は，センサノードの種類によって異なるセンサデータフォーマットを，扱えなくてはならない主体がアプリケーションであることに起因する．あらゆるアプリケーションの要求を満たせる精度と，種類の十分なセンサデバイスを持つセンサノードで，センサシステムを構成すれば，アプリケーションの様々な要求に対応できると考えられる．しかし，センサノードの製造コストはセンサデバイスの種類の数にほぼ比例して増加するため，種類の十分なセンサデバイスを持つセンサノードで構成するのは製造コストの面で現実的でないと言える．センサの標準フォーマットが存在しない現状において，センサデータフォーマットの問題を，利用センサノードが限定された構成を前提にして避けるべきでない．従って，利用するセンサを限定せずに，問題を解決する仕組みが必要であると考えられる．

## 4.2 センサネットワーク情報の管理

### 4.2.1 ネットワーク構成情報

センサネットワークにおいて，センサを取り付ける対象は，環境側と対象側の二つに分類できる．例えば室温を感知するセンサは環境側に取り付けられ，体温を感知するセンサは対象側に取り付けられる．対象側に取り付けられたセンサは，取り付けた対象と共に移動するため，一般的に対象側と環境側のセンサネットワークは別々に存在する．

インターネットを通じてセンサネットワークを利用する場合，どちらに分類されるセンサを利用しているのか知ることができない．分類情報を知ることができないと，同じ温度センサにおいて，室温を示すのか体温を示すのか分からず，センサデータを解析できない．そのため，センサネットワークで使用されるセンサの情報を管理する必要がある．

### 4.2.2 ネットワーク処理能力情報

インターネットを通じると，センサデータの活用範囲を拡張できるとともに，センサデータの要求数も増加すると想定される．多様な要求を受け付けることができる反面，センサネット

ワークを利用するアプリケーション数も増大することが予想される．

しかし，センサネットワークによっては，同時に利用できるアプリケーションの数が制限されている．TinyDB では同時に処理できるクエリ数が二つまでと仕様で定められている．例えば，三つ以上のクエリ処理を要求された場合，三つ目のクエリはシンクノードが受信せずにエラーとして処理しない．従って，三つ以上のクエリを処理させるためには，一つ目のクエリ処理を停止させてから三つ目のクエリを処理させなくてはならない．この問題により，複数のクエリを頻繁に受信すると，アプリケーションはセンサデータを受信することができなくなる．

想定環境において，アプリケーションの要求を最適化して，センサネットワークへ送信する必要がある．そのため，センサネットワークが対応する要求の処理能力などを適切に管理し，それに応じて利用する必要がある．

# 第5章　関連研究：センサネットワークを支える技術

本章では本研究が対象とするセンサネットワークを構成する技術を挙げる．研究分野を整理することで，本研究の実現する分野と既存研究との差異を明示する．

## 5.1　関連研究の分類

研究分野は大きく 3 つに分けることができる．図 5.1 に関連研究分野を分類して示す．図 5.1 には，実空間情報を利用するアプリケーション分野と，センサネットワークを構成するセンサネットワーク分野と，それぞれと通信するミドルウェア分野が示されている．本研究はセンサネットワークからセンサデータを取得し，アプリケーションへ提供するため，ミドルウェア分野に分類できる．

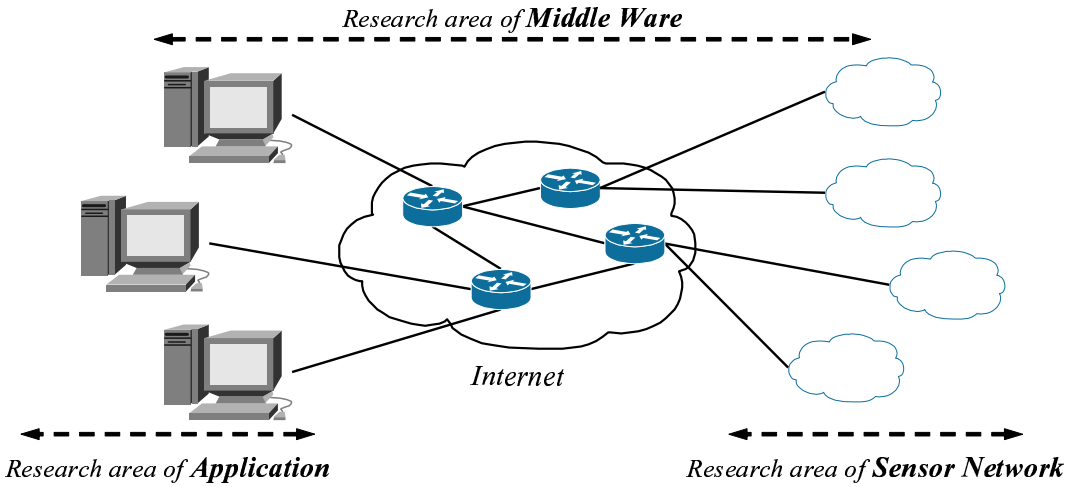


図 5.1　関連研究分野の分類

各研究分野における既存研究と機能例を表 5.1 に示す．以下の節において，本研究と最も関連するミドルウェア分野の既存研究を詳しく述べる．

表 5.1 関連研究分野の関連技術

| | 関連技術 | 機能 |
|---|---|---|
| Middle Ware | MANNA, LiveE! TinyDB, Cougar | データ集約，データ較正，データ破損対応など |
| Application | Sensor Web | データ可視化，異常状態検知・警告など |
| Sensor Network | TinyOS | ネットワーク構成，消費電力軽減，データ較正など |

## 5.2 MANNA

MANNA (A Management Architecture for Wireless Sensor Networks) [14] とは多種類のセンサネットワークを，一つのセンサシステムで利用するために必要な情報を管理するフレームワークである．管理する情報の例として，センサネットワークのセンサノード毎のバッテリ残量レベルや，搭載プロセッサ種類，通信の有向・無向などの情報を挙げている．予めデータを段階的に分け，各段階に一意の数値（コード）を対応させることで，管理情報の送信時における消費電力を抑える工夫をしている．

MANNA を利用したセンサシステムでは，いくつかのノードにエージェントとマネージャのいずれかの役割を予め設定する．エージェントの役割はノードごとの情報を保持し，マネージャへ提供することである．そして，マネージャの役割はエージェントを管理し，他のマネージャと情報を共有することである．エージェントとマネージャの二つの役割を設定することにより，それぞれがセンサネットワーク内外に接続されているトポロジに対応しているのが特徴である．

MANNA により，アプリケーションは多種類のセンサネットワークを利用するための情報を得ることができる．しかし，エージェントやマネージャを搭載できないセンサノードに対応できないため，アプリケーションが利用できるセンサネットワークの種類は限られる．さらに，アプリケーションからの多様な要求を最適化するなどの仕組みはなく，データフォーマットの統一もしない．そのため，多様な要求を持つアプリケーションの実装負担を減らすことはできない．

## 5.3 LiveE!

LiveE! [7] とは，世界中に分散した複数のセンサネットワークからセンサデータを収集するための基盤システムである．インタフェースをオープンにすることにより，多様なアプリケー

ションが情報を共有できるようにしている．この研究では，センサネットワークのデータを収集するために，WCN と PN の二つの役割を規定している [15]．WCN (Wireless Connected Network Node) はセンサネットワークと接続して，センサデータの情報を取得し蓄積する役割のノードである．また，PN (Patrol Node) は，WCN が物理的に設置された場所の近傍を巡回して，WCN からセンサデータを収集する役割である．PN の役割により，WCN はインターネットへの接続性を持つ必要がないため，設置場所を限定せずに適用範囲を広げられる利点を持つ．

LiveE!はセンサの物理的な設置場所の可能性を広げているため，アプリケーションの実用性を高め，多様な要求に応じることが期待できる．しかし一方で，使用するセンサは第 5.6 節で述べる LonWorks と互換性を持つセンサに限られる．そのため，多種類のセンサネットワークを連携できない．

## 5.4 TinyDB

同一センサネットワーク内で，センサの違いに依存せずにデータを取得するための研究として TinyDB [16] があり，同様の研究が盛んに行われている [17, 18]．TinyDB では，センサネットワーク内における，センサデータの発見を実現するための概念を提案している．TinyDB は TinyOS [19] 上で動作するアプリケーションである．

TinyOS はワイヤレスセンサノード用の OS である．ワイヤレスセンサネットワークのノード間のセンサデータ通信を容易に行えるようにするため，センサデバイスを操作してセンサデータを取得し，アプリケーションに統一的に提供する．TinyOS を使用できるセンサとして Mica や Telos などの Mote シリーズがある．

TinyDB は各センサに組み込まれ，要求に応じてセンサデータをシンクノードへ送信する．指定した任意の種類のセンサデータのみを取得することが可能であるため，センサデータを効率よく提供することができる．しかし，TinyOS 上で動くことを前提としているため，他のセンサシステムとの連携して多種類のセンサデータを利用することができない．

TinyDB は，データセントリック [20, 21] の概念を適用し，センサネットワークをセンサデータを格納するデータベースとして抽象化する概念を提案している．センサネットワークをデータベースとして抽象化することによって，データの利用と取得の機能を持つセンサネットワークを，データ取得の機能に単純化している．データの利用がネットワークの機能と分離されるため，アプリケーションを容易に変更できるメリットがある．

## 5.5 Cougar

Cougar [22] はデータセントリックの概念によって，センサネットワークをデータベースとして抽象化する手法を提案した研究である．センサネットワーク上でセンサデータ通信に加えて，データ要求をセンサネットワークに分散させるゲートウェイノードがある．ゲートウェイからデータ要求を受け取ったセンサノードは，要求に応じたセンサデータがあればゲートウェイに返答する．この二つの役割を持つノードの連携によって，センサネットワーク全体をデータベースとして抽象化している．従って，アプリケーションはセンサネットワークから統一的なアクセス方法でセンサデータを取得することができる．一方，この研究はセンサデータのフォーマットを統一しないため，センサネットワークの連携を行うのは難しい．

## 5.6 LonWorks

センサ情報を統合的に扱うシステムの一つとして，LonWorks [23] がある．LonWorks は消費電力の効率を高める目的で，電気，空調，照明，セキュリティ等の各種設備を一元的に制御，管理できるビルオートメーションシステムである．設備機器は，TCP/IP に準拠した通信プロトコルを内蔵した LSI を利用して，通信と制御を行う．各センサはセンサデータの取得とアクチュエータ制御の二種類の機能を対等に行う役割を持つ．二種類の機能はミューロンチップによって実現する．ミューロンチップは LonWorks が定めた通信プロトコルを使ってセンサ同士の通信を実現するためのセンサ埋め込み用マイクロチップである．この規格では，全てのセンサにミューロンチップを内蔵しなくてはならない．従って，現状ではチップを搭載していない Mote や µPart や $U^3$ など，既存のセンサノードを LonWorks に組み込むことは困難である．

# 第6章　アプローチ

第 4 章で，次の要件が問題解決に必要な課題であることを述べた．

- 実装コストの軽減
- センサネットワークの利用に必要な情報管理

本章では課題を実現するためのアプローチを述べる．具体的には，アプリケーションからセンサネットワークへのアクセス方法を統一するため，プロキシモデルの適用を提案する．

## 6.1　プロキシモデル

この課題を実現するために，プロキシモデルの適用を提案する．プロキシモデルの適用によるセンサデータ取得方法の変化を図 6.1 に示す．

プロキシモデルでは，アプリケーションとセンサネットワークの通信を中継する役割としてプロキシを定義する．プロキシはアプリケーションからのセンサネットワークアクセスを代替し，センサネットワークからのセンサデータを中継する．

アプリケーションはセンサデータ要求のためにプロキシと通信を行えばよいため，アクセス方法を統一できる．また，センサネットワークへのアクセスとセンサデータの解釈をプロキシが行うことで，センサネットワーク構成の拡張に応じてアプリケーションを再実装しないですむ．この利点により，アプリケーションの実装コストを軽減することができる．

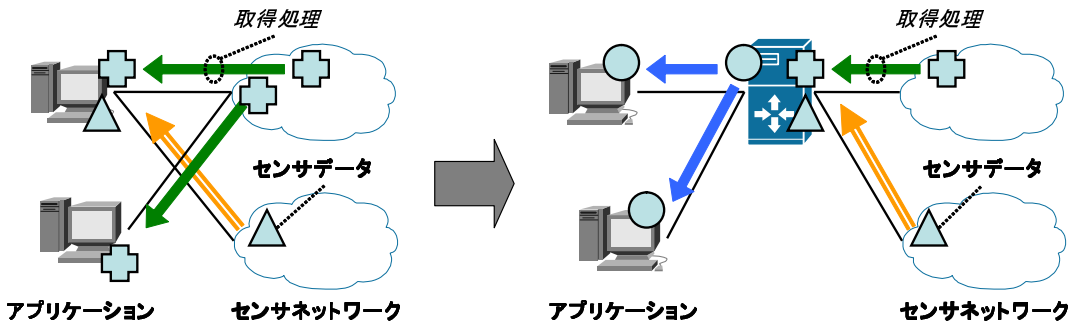


図 6.1　プロキシモデルの適用によるセンサデータ取得方法変化

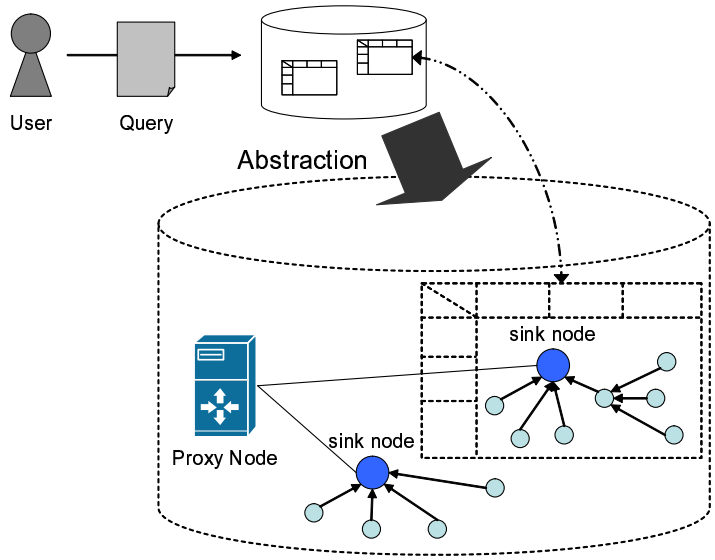


図 6.2 多種類センサネットワークをデータベースとして抽象化する概念モデル

## 6.2 センサネットワークの抽象化

プロキシモデルにより，アプリケーションからのデータアクセス方法を統一できれば，アプリケーションはセンサネットワークを多種類のセンサデータを保持するデータベースとして見ることができる． [24] 様々なセンサデータを統合するセンサシステムにおいても，各種類のセンサデータを共通のアクセス方法で取得できるため，アプリケーションは容易にセンサデータを利用することができる．

図 6.2 は多種類のセンサネットワークを データベースとして抽象化する概念を表す．各センサネットワークはデータベースにおけるテーブルに該当する．

センサネットワークをデータベースとして抽象化することで，センサデータの取得に関する操作を読み込み操作で一般化できる．そのため，センサデータのフォーマットと操作に関する情報の管理により，柔軟にセンサネットワークを利用できると考えられる．

## 6.3 センサデータ要求書式

データセントリック方式で記述できるセンサデータの要求書式として，SQL が挙げられる．一方，SensorML [25] や EPCIS [26] のクエリ操作モジュールなど，XML を用いて定義されるデータの要求書式が数多く存在する．本節では，センサデータの要求書式に関して議論する．

EPCIS (Electric Product Code Information Service) とは，IC タグの情報をアプリケーションで様々な目的で利用するためのサービスである．EPCIS の仕様は EPCGlobal [27] により策定されている．EPCIS ネットワークに参加するアプリケーションは，IC タグに書き込まれた識別子を基に EPCIS から取得できる．そのため，EPCIS で使用されるクエリ操作はセンサデータの要求に応用できる．しかし，すべてのデータ要求において XML を記述する必要があるため，XML を容易に記述できる実装が必須となり，新たな実装負担を強いると考えられる．従って，本研究における要求書式としては不適切である．

SQL はリレーショナルデータベースマネージメントシステムにおいて，データの要求や登録・削除・更新操作を記述するための言語である．SQL はコマンド形式の文字列で表されるため，マークアップ言語と比べ，データ通信量の面で要求書式として有利である．また，SQL はデータベースからあらゆるデータの要求を十分に行えるため，要求対象がデータベースである場合，要求書式として適しているともいえる．従って，SQL と同様の構造をセンサデータ要求書式に用いることで，アプリケーションからの多様な要求を満たし，センサノードの構成変更に柔軟に対応できると考えた．

# 第7章　設計

本研究では，多種類のセンサネットワークからセンサデータを取得するアプリケーションに対して，効果的な実装環境を提供するプロクシノードを設計し，実装した．本章ではプロクシノードの詳細な設計を述べる．図 7.1 はプロクシノードを利用したセンサシステムの構成例である．

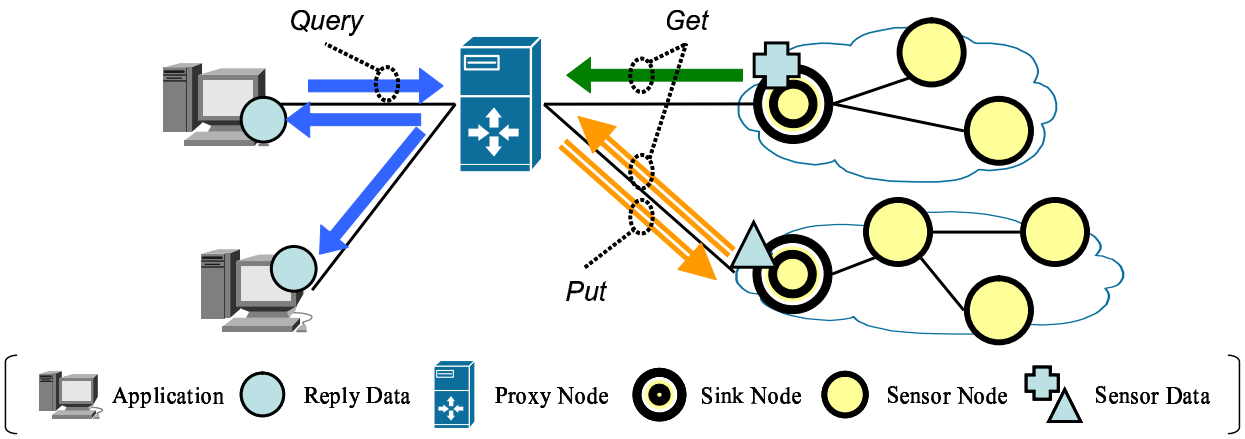


図 7.1　プロクシノードを使用したセンサシステム構成例

## 7.1　センサデータの要求受信

アプリケーションからセンサデータの要求を受信する機能を設計する．

アプリケーションごとに必要となるセンサデータは異なるが，他のアプリケーションと非同期に取得できることがセンサデータの利用効率の面から望ましい．複数のアプリケーションが本システムを同時に利用する場合，それぞれのアプリケーションが要求する条件を本システム内で記憶し，条件に合致したセンサデータのみ選択して返答できるようにする．

### 7.1.1　データ通信方式の統一

第 2.3.1 節でセンサネットワークとのデータ通信方式を，**プル型**，**プッシュ型**，**イベントドリブン型**の三つに分類した．本実装では，シンクノードが対応しない通信方式を補完し，アプリケーションが効果的にシンクノードを利用できるようにする．

通信方式を統一させるために実装すべき機能を整理し，表 7.1 に示す．表 7.1 は縦軸に示した通信方式を実装するために必要な機能を，センサネットワークが対応する通信方式により場合分けして示している．機能の詳細を以下に示す．表 7.1 より，アプリケーションへ全ての通

表 7.1 データ受信方式の統一化に必要な機能

| | | センサネットワークが対応する通信方式 | | |
|---|---|---|---|---|
| | | **プル型** | **プッシュ型** | **イベントドリブン型** |
| **本実装が対応すべき通信方式** | **プル型** | $(A)$ | $(A)$ | $(A)$ $(C)$ |
| | **プッシュ型** | $(A)$ $(D)$ | $(A)$ | $(A)$ $(C)$ |
| | **イベントドリブン型** | $(A)$ $(B)$ $(D)$ | $(A)$ $(B)$ | $(A)$ |

信方法を提供するためには，五つの機能が必要であることが分かる．

$(A)$ 要求の転送機能

要求転送機能とは，アプリケーションからの要求をシンクノードへ転送する機能である．シンクノードが実現させたい機能に対応している場合，シンクノードへのアクセスが可能であれば新たな機能を追加する必要はない．そのため，アプリケーションからの要求をシンクノードへ転送する機能があればよい．

$(B)$ 指定条件によるセンサデータの選択機能

センサデータがアプリケーションの要求条件に当てはまるかを比較する機能である．シンクノードがイベントドリブン型である場合，条件を比較する機能を実装しているため，条件比較機能を実装する必要はない．

$(C)$ 全てのセンサデータを選択可能な条件の指定機能

全てのセンサデータを要求可能な条件を指定する機能である．センサネットワークごとに静的に条件を設定できればよい．

$(D)$ 逐次的な要求の送信機能

逐次要求機能とは，一定の周期でシンクノードに要求を送信する機能である．

### 7.1.2 データ通信書式

アプリケーションとプロキシノードとの通信書式として，**データ要求書式**と**返答データフォーマット**の二点を定める．以下に詳細を述べる．

**データ要求書式**

センサデータを指定する方法を定める．本システムはアプリケーションが指定したセンサデータのみを返答するため，アプリケーションが受け取るセンサデータは，センサデータを指定する方法で決まる．アプリケーションが，いくつかのセンサデータを統計的に計算し，利用者に表示するのが目的である場合，いくつかのセンサデータをまとめた結果や，特定の条件を満たすデータが必要になる．センサデータ要求書式が条件を柔軟に表現可能であれば，全てのセンサデータを集めるという実装負担を減らせる．さらに利用者へ提供する情報を多様に表現するためには，対象となるセンサデータの粒度は細かく，より利用者へ提供する情報を生成しやすい形式で指定できる必要がある．

センサデータの要求言語として SQL ライクな要求言語が適していることを第 6.3 節で述べた．センサデータを要求するために必要となる機能を考察し，SQL の構文と対応する機能を定める．センサデータの要求は，データの取得操作であるため SELECT 文を用いる．条件を設定する場合は，WHERE 句を用いて，条件式を SELECT 文内に記述する．条件を設定するかしないかはアプリケーションの目的に応じて異なるため，WHERE 句は省略できることとする．

**返答データフォーマット**

センサデータの返答を示すデータフォーマットを定める．シンクノードとの通信方式を統一するためには，表 7.1 に示した機能を実装する必要があることを第 2.3.1 節で述べた．非同期型の通信方式において，アプリケーションが複数の要求を送信した場合，返答データの順序は，要求を送信した順序と一致することが保障されない．そのため，返答データがどの要求の返答であるのか知ることができない．アプリケーションが要求したデータを利用するためには，要求と返答データの対応を識別する機能が必要である．

識別方法として二つの方法が考えられる．一つ目は要求クエリに識別子を付け，返答データに同じ識別子をつける方法である．そして，二つ目は返答データに要求したデータの識別情報を付加する方法である．一つ目の方法は要求クエリを一意に定めるための機構が，アプリケーション側かプロキシノード側に実装する必要になる．アプリケーション側に実装する場合，他のアプリケーションとの相互判断する上での識別機構が必要になる．また，プロキシノード側に実装する場合，プロキシノードから要求クエリの識別子を取得するための通信が必要となり，アプリケーションへの実装負担を強いることとなる．アプリケーションを容易に実装できるよう，本実装では二つ目の方法を利用する．二つ目の方法では，返答データは一つで完結するため，全ての返答データを一様に処理でき，実装負担は軽減できる．

図 7.2 に返答データフォーマットを示す．複数の返答データは改行で区切り，一つの返答

データ内における複数のセンサデータはカンマで区切る．一つのセンサデータは，スペック情報に設定したデータ種類と値の対をイコールで接続する．

データ種類 = 値 [, データ種類 = 値 [, · · · ]] ↵

図 7.2 返答データフォーマット

### 7.1.3 センサデータ要求の最適化

アプリケーションからの要求を，そのままセンサネットワークへ伝えるのは望ましくないことを第 3 章で示した．センサネットワークからセンサデータを受け取ることのできるアプリケーション数をできる限り増やして，利用効率をあげられる機能を設計する．

アプリケーションからのクエリを一つに集約することで，センサネットワークからのデータ通信量を減少させる仕組みをプロキシノードに実装する．プロキシノードは複数のアプリケーションからのクエリを受け取ることを想定しているため，全てのアプリケーションからのクエリを，そのままセンサネットワークへの問い合わせるのはセンサのデータ通信量の面から効率的ではない．センサのデータ通信量が増加すれば，消費電力は増加し，バッテリ寿命が短くなる．そのため，問い合わせ要求を集約することで，センサネットワークから効率よくセンサデータを取得できる．

## 7.2 センサデータフォーマットの解釈

異なるセンサデータフォーマットを持つセンサネットワークからセンサデータを取得し，アプリケーションへ提供するためには，センサデータフォーマットをセンサネットワークごとに解釈する必要がある．本節では，センサデータフォーマットを解釈するためにスペック情報を定義し，スペック情報をプロキシノードへ設定する方法を定める．

### 7.2.1 スペック情報

スペック情報とは，センサネットワークから取得するセンサデータに含まれる，**センサデータ値の種類**，**データ型**，**データ長**，**データ単位**の定義である．付録 A の図 A.1 において，スペック情報を C 言語上で実装した際の定義を示す．センサデータ値の種類とは，センサノードが固有に持つ情報とセンサデバイスが計測した対象を示す情報である．例えば，センサノードの識別子や温度，湿度である．データ型とは，センサデータ値が整数値，小数点，ASCII 文

字列，バイナリ文字列のいずれの型であるかを示す情報である．そして，データ長はセンサデータ値が何バイトのサイズであるかを示し，データ単位はセンサデータ値の単位を示す情報である．

プロキシノードでの処理のため，センサネットワーク内で利用するセンサのセンサデータに関する情報をスペック情報として記述する．プロキシノードはスペック情報を扱うことで，センサデータ取得操作に拡張性を持たせられるため，利用するセンサの構成変更に対応できる．

スペック情報により，データ型とデータ長に応じて複数のセンサデータを含むデータパケットを分割できる．結果，分割したセンサデータごとにセンサデータの種類とデータ単位を関連付けることができ，アプリケーションが要求した条件と比較できる．

### 7.2.2 スペック情報の設定

スペック情報はセンサネットワークを構成するセンサに変更がなければ更新する必要はない．そのためスペック情報は更新頻度が低く，プロキシノードの実装コード内に設定されていても動作に支障はないと言える．しかし，センサの変更を容易にするためには更新の利便性を考慮する必要があるため，センサネットワークの管理者が直接変更できる方法も別に実装する必要がある．具体的には，プロキシノードとセンサネットワークの管理者がアクセス可能なファイルやデータベースにスペック情報を蓄積し，センサデータを解釈する前にスペック情報を読み込む方法である．

### 7.2.3 センサネットワークの登録

センサネットワークごとに定めたスペック情報を読み込むためには，センサデータ取得先のセンサネットワークを識別し，スペック情報と対応付ける必要がある．センサデータ取得先のセンサネットワークをスペック情報と対応付ける動作を，センサネットワークの登録と呼ぶ．

センサネットワークを登録する方法は，登録する主体により三種類の方法に分けて考えられる．一つ目の方法はシンクノードが登録する方法である．シンクノードが自身が構成するセンサネットワークの識別情報を，通信の開始時に個別で，またはヘッダとしてプロキシノードへ送信することでセンサネットワークを登録できる．しかし，全てのセンサネットワークに対する通信プロトコルの規定による仕様の制約は，利用するセンサ構成を制限することとなるため，拡張性の観点からは適していない．二つ目の方法はプロキシノードが登録する方法である．三つ目の方法はプロキシノードとシンクノード以外の主体が登録する方法である．例えば DNS などの IP アドレスと名前を対応づけられるシステムとの連携が考えられる．

本実装では，センサネットワークをプロキシノードが登録するよう実装する．センサネットワークにおけるセンサの構成変更に対応でき，通信プロトコルの制約や動作に必要な計算資源を制限が無い点で，他の方法に比べて有利であると考えられる．

## 7.3 センサネットワークへのアクセス

プロキシノードからセンサネットワークへのアクセス方法を設計する．本節では，アクセス方法の設計に必要な，センサネットワークの識別，センサネットワークに対する操作関数，データフォーマットについて述べる．

センサネットワークは IP アドレスによって識別する．本研究が想定する環境は，多種類のセンサネットワークが IP ネットワークを通じて接続されている環境であることを第 3 章で述べた．すなわち，シンクノードは IP ネットワークとセンサネットワークとの境界に存在するため，プロキシノードとシンクノードは IP ネットワーク上のノードである．結果，本システムはシンクノードを IP で識別することができる．シンクノードはセンサネットワークからセンサデータを取得することができるため，シンクノードを識別することでセンサネットワークを識別することができる．

本システムがセンサネットワークに行う操作として get と put を規定する．シンクノードを通して取得できるデータはセンサネットワークごとに異なる．さらに，センサデバイスの変更に伴ってデータフォーマットも変わる．よって，センサデバイスの変更によるアプリケーションの再実装コストを軽減するためには，データフォーマットの解釈を本システム内で行う必要がある．以上の要件を満たすために，センサネットワークから取得するデータは一種類，または多種類のセンサデータを保持しているとする．get は取得したデータをビット列として受け取り，各センサネットワークのデータフォーマットに従って各種類のセンサデータの値を分割し，センサデータの種類，単位を意味づけする機能を実装する．

## 7.4 要件整理

以上の機能を実現するプロキシノードを設計した．スキーマは，通信対象により実装する機能をインタフェースとして分担し，三つの概念構造で構成されるように設計した．図 7.3 に設計したプロキシノードのスキーマを示す．アプリケーションとの通信に必要な機能をアプリケーションインタフェースとし，センサネットワークとの通信に必要な機能をセンサネットワークインタフェースとした．そして，インタフェース動作を連結する概念をインタフェースアダプタとした．

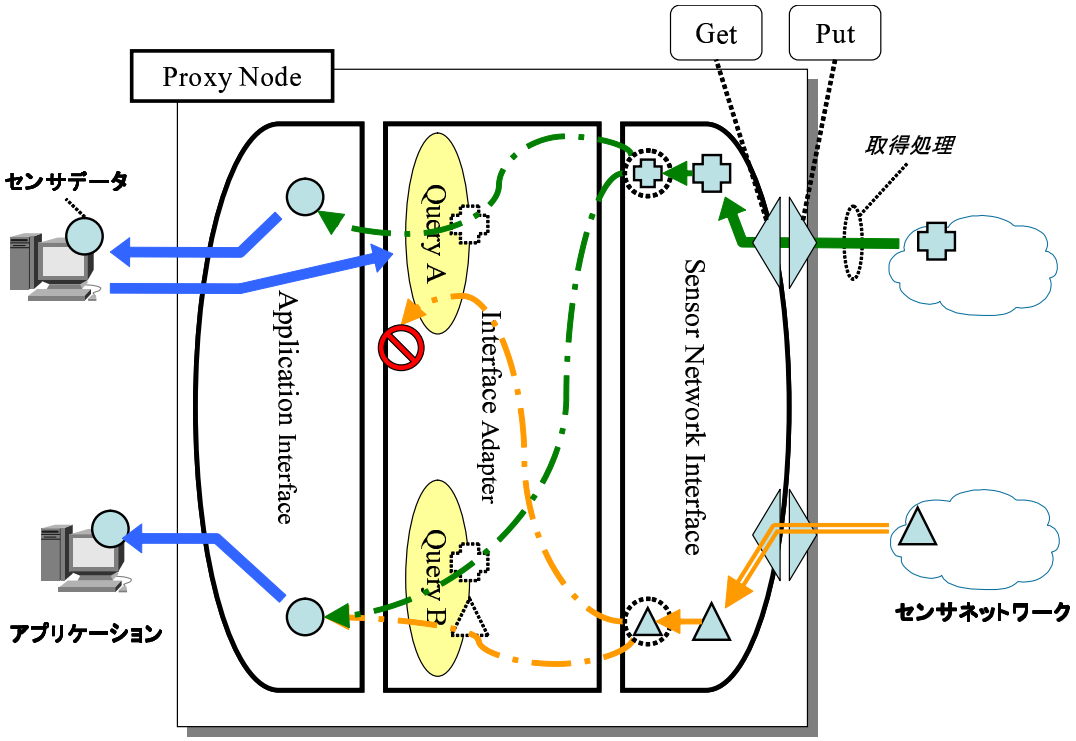


図 7.3 プロキシノードのスキーマ設計

# 第8章　実装

本章では第 7 章の設計に従って実装したプロキシノードの詳細とともに，プロキシノードを利用するサンプルアプリケーションとして実装した室内環境モニタリングアプリケーションの詳細を述べる．プロキシノードとアプリケーションの実装環境を表 8.1 に示す．

表 8.1　実装環境

| | OS | 言語 | コンパイラ | その他 |
|---|---|---|---|---|
| プロキシノード | Debian 4.1 | C | gcc 4.1.2 | bison 2.3 (Yacc/Lex) |
| 室内環境モニタリング<br>サンプルアプリケーション | WindowsXP SP2 | Java | javac 1.6.0 | なし |
| simuPart | Debian 4.1 | C | gcc 4.1.2 | なし |

## 8.1　対応センサネットワーク

プロクシノードを二種類のセンサネットワークに対応して実装した．一つ目は TinyDB を実装した MICAz で構成されるセンサネットワークである．二つ目は XBridge と µPart で構成されるセンサネットワークである．

プロクシノードの動作を検証するために，同じ二種類のセンサネットワーク環境を実機を用いて構築した．また，同じ種類のセンサネットワークにが実空間の広域に設置された環境におけるプロクシノードの動作を確認するため，多数のセンサノードにより構築されるセンサネットワーク環境を，シミュレータで一部を代替して構築した．

実機を用いたセンサシステムのトポロジを図 8.1 に示し，センサネットワークをシミュレータで代替したセンサシステムのトポロジを図 8.2 に示す．

図 8.1 におけるセンサネットワークを構成するセンサノードは MICAz と µPart と XBridge である．MICAz，µPart ，XBridge の写真とセンサネットワークにおけるトポロジを，それぞれ図 8.3 と図 8.4 に示す．各センサネットワークのシンクノードがプロキシノードと接続されている．MICAz で構成されるセンサネットワークにおけるシンクノードは，ID が 0 の MICAz である．シンクノードの MICAz は，対外ネットワークとの物理接続インタフェースとしてシリアルしか持たないため，プロクシノードと常に直接的な接続ができない場合がある．プロクシノードが IP アドレスでセンサネットワークを識別するために，IP 通信とシリア

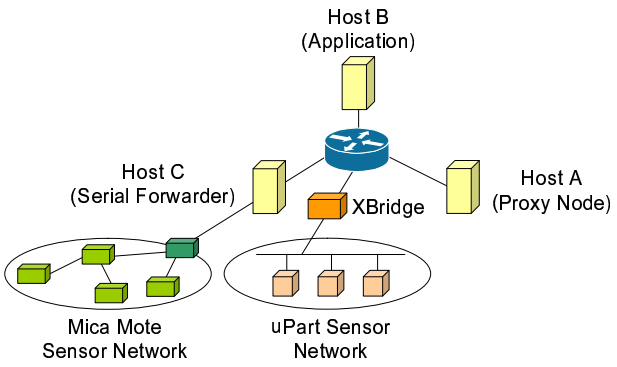


図 8.1 動作検証環境のトポロジ

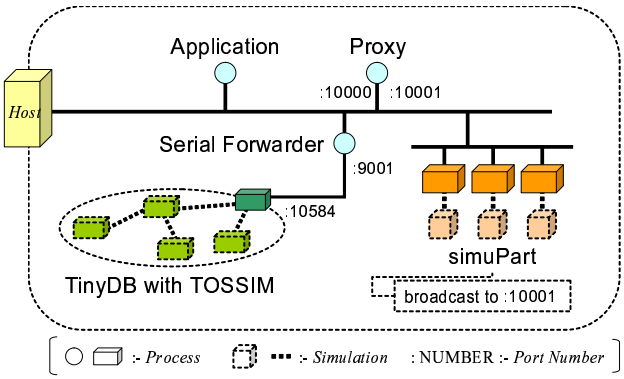


図 8.2 シミュレート環境トポロジ

ル通信をブリッジするプログラムの Serial Forwarder を使用した．また，µPart と XBridge で構成されるセンサネットワークにおけるシンクノードは XBridge である．

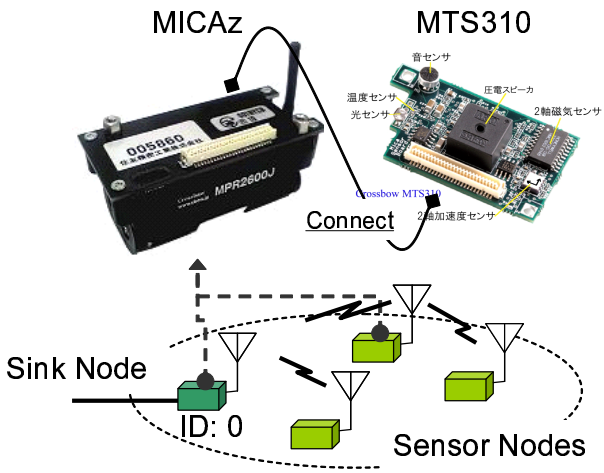


図 8.3 MICAz and MTS310

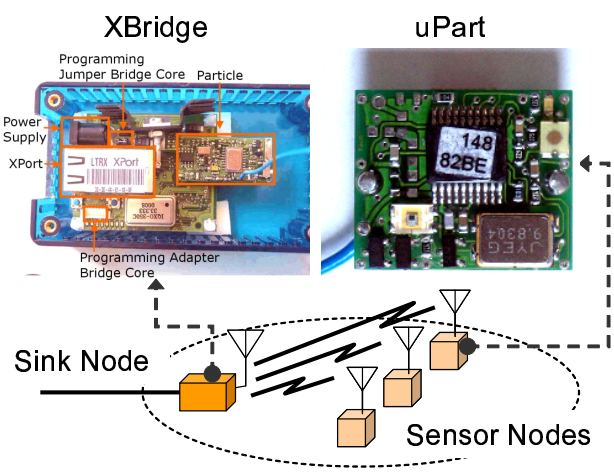


図 8.4 XBridge and µPart

図 8.2 におけるセンサネットワークはシミュレータで代替した．MicaMote の動作は TOSSIM [28] シミュレータで代替した．また，XBridge と µPart の動作をシミュレートする simuPart を C 言語で実装し，センサネットワークの動作を代替した．

TOSSIM は TinyOS のシミュレータである．TinyDB などの TinyOS 上で動作するアプリケーションは TOSSIM を用いて動作をシミュレートできる．TOSSIM は MicaMote 間の無線通信をシミュレートし，電波干渉による無線通信の可否をシミュレートする．

simuPart は XBridge と同じく，µPart からのセンサデータを UDP パケットでブロードキャストする．センサデータ値はランダムに設定し，ランダム値の範囲を指定できる．ランダ

表 8.2 XBridge とシミュレータの動作の違い

| シミュレートしない点 | XBridge とシミュレータの動作の違い |
|---|---|
| 電波干渉 | データ破損，パケットロス |
| 伝送遅延 | センサデータ生成時刻と受信時刻のずれ |
| 電波到達距離 | パケットロス |
| 電波強度 | パケットロス |

ム値は ANSI C 標準の rand 関数を用いて，生成時刻を 32bit ランダムシードとした結果を利用した．また，simuPart は µPart と XBridge 間の無線通信はシミュレートしないため，実機の動作とは表 8.2 に示す点が異なる．

パケットロスはプロキシノードにセンサデータが到達しないことを示す．データ破損は，センサデータが生成した時の値と異なることを示す．この動作の違いにより，シミュレータで代替して構築したセンサネットワーク環境では，アプリケーションに到達するパケットの到達率と，データ破損率を検証することはできない．しかし一方で，プロキシノードがセンサデータをアプリケーションに中継する処理速度は，パケットロスとデータ破損に影響しないため，シミュレーション環境においても十分に検証できる．

## 8.2 プロキシノード

新しい種類のセンサネットワークからセンサデータを取得するためには，センサデータ解析処理と通信処理をプロキシノードに実装しなくてはならない．この処理を容易にするため，センサネットワークごとの処理部分をアプリケーション通信部分と分けて実装した．本体と分けた処理部分をセンサネットワークプラグインと呼ぶ．プラグインによって通信方式と，get 関数，put 関数を定め，アプリケーションからの要求に応じてそれぞれの関数を呼び出す．新たな種類のセンサネットワークへは，センサネットワークプラグインを追加することで対応できる．

**Socket Manager** Socket Manager は IP を使った通信のコネクションを概念として導入し，通信をコネクション名で管理するための機能である．IP で識別したセンサネットワークにアクセスするための実装である．

IP の通信に対応するため，TCP と UDP のどちらのプロトコルも使用できる．TCP の場合，通信先のアドレスとポート番号を指定すると，サーバとして動作するための listen 状態で

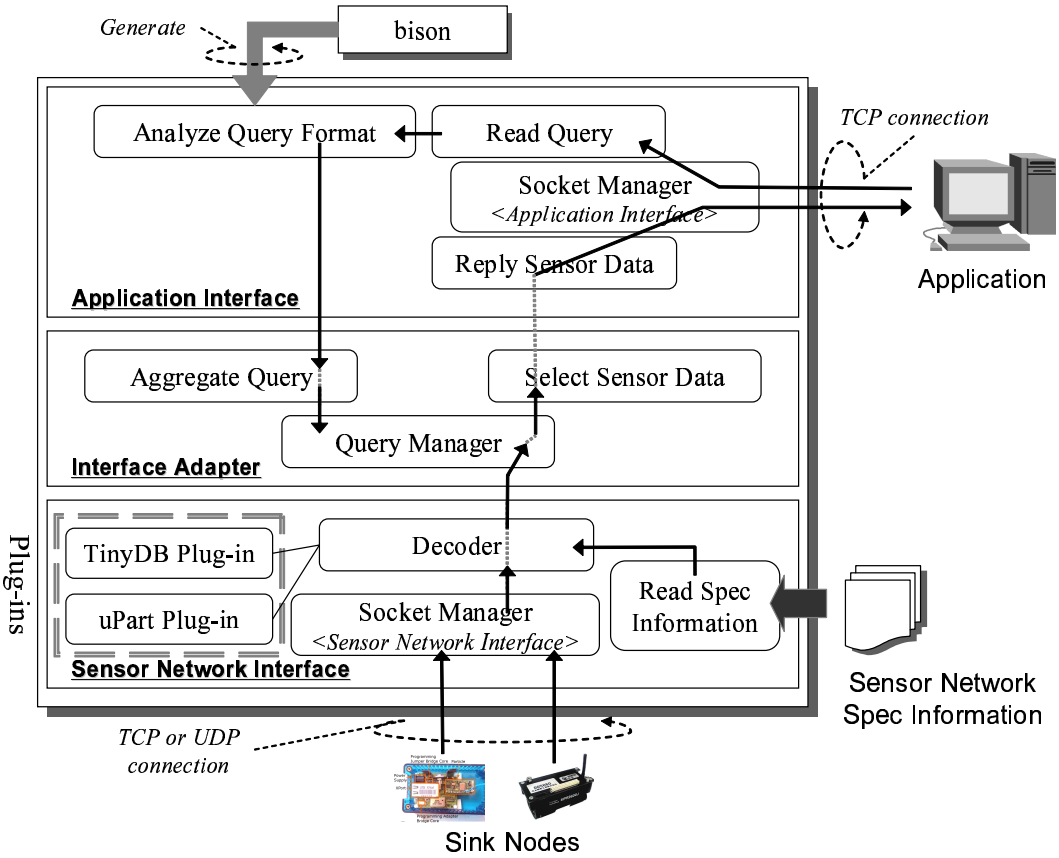


図 8.5 ProxyNode の動作概要図

開いたソケットを作成する．UDP の場合，通信先のアドレスを指定すると，指定したアドレスからの UDP パケットを受信するためのソケットを作成する．

**Read Query** Read Query は，アプリケーションから TCP/IP を用いて，センサデータ要求クエリを読む機能である．受け取ったクエリを Analyze Query Format 解析器に渡す．

**Reply Sensor Data** Reply Sensor Data は，クエリで指定した条件に合うセンサデータを TCP/IP を用いて返答する．返答データは図 7.2 で示したフォーマットに従って CSV 形式で返す．

**Analyze Query Format** Analyze Query Format は bison により生成した解析器である．SQL ライクなセンサデータ要求クエリを構文解析する．受け取るセンサデータを条件と比較するために QueryManager へデータを渡す．

**Aggregate Query** Aggregate Query は，複数のアプリケーションからの要求クエリを集

約する機能である．TinyDB を例とする一部のセンサネットワークから効率的にセンサデータを取得するための機能である．

**Select Sensor Data** Compare Query は，集約したクエリとセンサデータを比較するための機能である．比較のために，センサデータの種類を変数とし木構造により条件の要素を保存する．各要素を接続する条件を再帰的に比較して，センサデータが条件と合っているかを判定する．

**Query Manager** Query Manager は，いつでもセンサデータと条件を比較できるように保存する機能である．

**TinyDB Plug-in** TinyDB Plug-in は TinyDB を用いた MicaMote により構成されたセンサネットワークから，センサデータの受信をサポートする機能である．

**µPart Plug-in** µPart Plug-in は XBridge と µPart により構成されたセンサネットワークから，センサデータの受信をサポートする機能である．

**Decoder** Decoder はシンクノードに応じて適切なプラグインを用いてセンサデータを受信し，スペック情報に応じてセンサデータを解析する機能である．センサデータを解析するためにスペック情報をプラグインに渡し，スペック情報の変更に対応する．

**Read Spec Info** Read Spec Info はスペック情報を解析し，構造体変数に格納する機能である．

### 8.2.1 センサネットワークプラグイン

今回は TinyDB と µPart 用のセンサネットワークプラグインを実装し，各センサネットワークが持つセンサのスペック情報を，予めソースコード内に埋め込んだ．プロキシノードがセンサデータを取得するまでの処理の流れを図 8.6，図 8.7 に示す．

TinyDB は TinyOS 上で動作するため，TinyOS を実装した MICAz センサを用いて TinyDB を動作させた．MICAz は無線通信規格に IEEE802.15.4 を使い，MOTE と Zigbee の二種類の無線通信規格を切り替えて使用することができる．MICAz のセンサデバイスは二軸加速度・二軸磁気・光・温度・音センサを持つ MTS310 を使った．µPart は，振動，光，温度センサを持つ．

TinyDB はプル型とイベントドリブン型のデータ送信方式を持つため，TinyDB プラグインはアプリケーションからの要求があった場合にセンサデータ要求を発行する．µPart はプッ

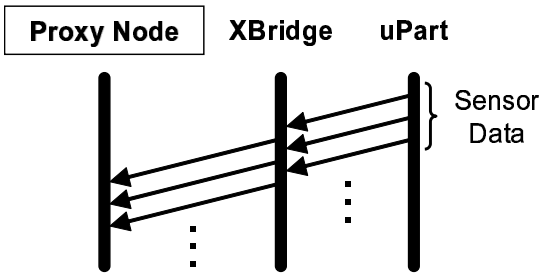


図 8.6 µPart からのからのセンサデータ取得の流れ

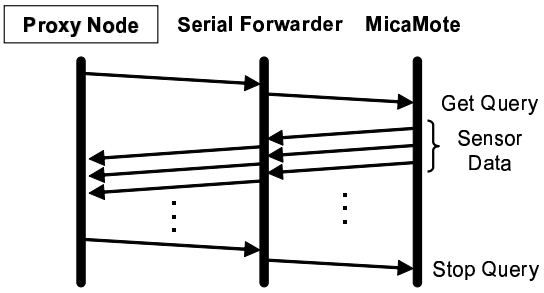


図 8.7 Mote からのセンサデータ取得の流れ

シュ型のデータ送信方式を持つため，µPart プラグインは常にセンサデータを取得し続ける．取得したセンサデータは，予め記録したスペック情報に従ってセンサデータを取り出し，スペック情報と関連付ける．保持しているセンサデータはアプリケーションの要求と比較し，要求と一致しているセンサデータのみをアプリケーションへ提供する．

### 8.2.2 アプリケーションインタフェース

アプリケーションとの通信における，センサデータの要求クエリ受信と返答データの送信は TCP/IP を用い，ポート番号は 10000 番を用いた．クエリ文は SQL ライクな記述言語で表す．例えば，温度データを取得したい場合は，図 8.8 に示すクエリを記述する．

| | |
|---|---|
| クエリ | SELECT temperature ; |
| 返答例 | temperature=27 ↵ |
| | temperature=28 ↵ |
| | temperature=27 ↵ |
| | ⋮ |

図 8.8 温度データ取得クエリと返答例

例えばアプリケーションは，センサデータの値が普段と異なる場合に処理を行うためには，全てのセンサデータを受け取って値の比較を行わなくてはならない．処理の効率をあげるため，センサデータを値によって条件指定することにより，特定の値のみを返答できる機能を実装した．例えば，図 8.9 のクエリにより，50 度以上のセンサデータのみ受信ができる．

プロキシノードが提供するクエリ要求書式において使用できる演算子と定数を図 8.10 にまとめて示す．より詳細な記述書式の定義は付録 D の図 D.1 に示す．

| | |
|---|---|
| クエリ | SELECT nodeid, temperature WHERE temperature > 50 ; |
| 返答例 | nodeid=12340001,temperature=51 ↵<br>nodeid=12340002,temperature=51 ↵<br>⋮ |

図 8.9　50 度以上の温度データ取得クエリ

| | |
|---|---|
| 比較演算子 | $<$　$>$　$<=$　$>=$　$<>$ |
| 論理演算子 | $AND$　$OR$　$NOT$ |
| 算術演算子 | $+$　$-$　$/$　$*$ |
| 整数型定数値 | $-2,147,483,647 \sim +2,147,483,647$ |

図 8.10　アプリケーションインタフェースにおけるクエリ要求書式

### 8.2.3　インタフェースアダプタとクエリの最適化

インタフェースアダプタにおいて，アプリケーションからの要求を集約する機能を実装した．さらに，センサネットワークが複数のクエリを処理できるかどうかによって集約動作を分類し，クエリの更新を連続的に行う動作を実装した．この動作を図 8.11，図 8.12 に示す．図 8.11，図 8.12 は，クエリ a の発行後，クエリ b が発行された場合を例として，センサネットワークが複数クエリを処理できる場合と，できない場合による更新動作の違いを示している．

プロキシノードはクエリ b が発行された後，センサネットワークが複数のクエリを同時に処理できない場合は，クエリ a を停止させた後に，クエリ a，b を集約したクエリによる問い合わせを送信する．複数クエリを同時に処理できる場合は，クエリ a，b を集約したクエリによる問い合わせをシンクノードに送信する．クエリ a，b の返答を受信すると，クエリ a を停止する．この場合，複数クエリを処理できない場合に比べ，クエリ a の返答をアプリケーションへ送信できない時間がなくなるため，返答を連続して行うことができる．

## 8.3　室内環境モニタリングサンプルアプリケーション

センサデータを利用するアプリケーションとして実装した環境モニタリングアプリケーションの画面のスナップショットを図 8.14 に示し，トポロジを図 8.13 に示す．このアプリケーションは室内の温度変化と照度変化を管理する利用例である．センサデータ要求クエリの

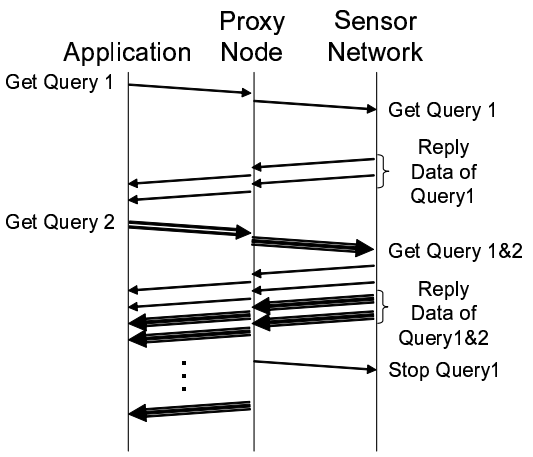


図 8.11 クエリ集約動作 a（複数クエリ同時処理対応）

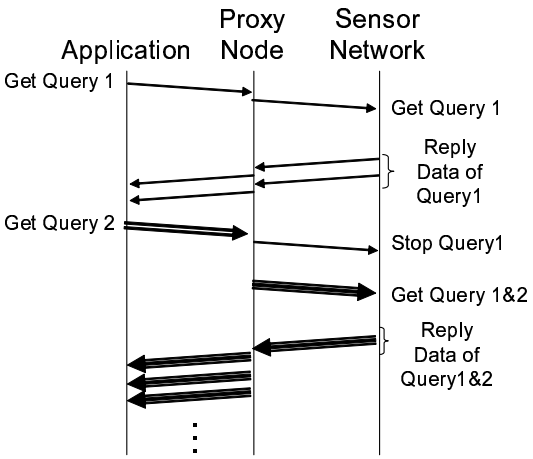


図 8.12 クエリ集約動作 b（複数クエリ同時処理未対応）

SQL 文を図 8.15 に示す.

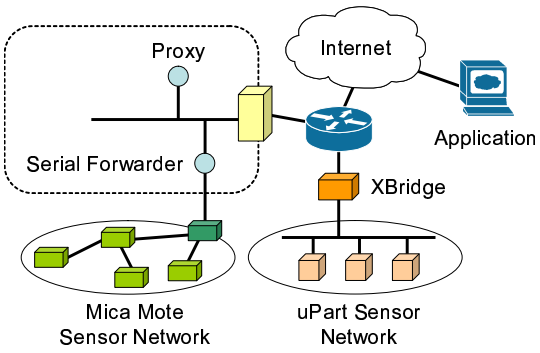


図 8.13 室内環境モニタリングアプリケーション動作トポロジ

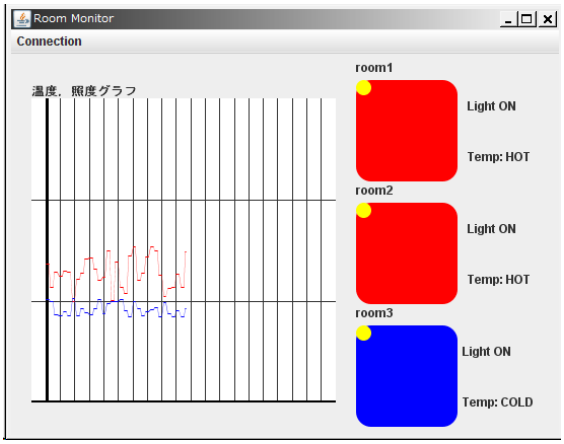


図 8.14 室内環境モニタリングアプリケーション

アプリケーションは室内の温度と照度をモニタリングする．室内に設置された全ての温度センサと照度センサの値を平均して，それぞれ室温と照明の明るさとみなす．室温は 20℃ 未満は青色，20℃ 以上 25℃ 未満は黄色，25℃ 以上は赤色を使って表す．照明の明るさは $100lux$ 未満は無点灯状態，$100lux$ 以上は点灯状態を表す．また，過去の室温と照明の明るさの記録をグラフで表す.

```
SELECT id, temperature where temperature < 20 ;
SELECT id, temperature where temperature >= 20 and temperature < 25 ;
SELECT id, temperature where temperature >= 25 and temperature < 30 ;
SELECT id, light where light < 100 ;
SELECT id, light where light >= 100 ;
SELECT id, light, temperature ;
```

図 8.15　室内環境モニタリングアプリケーションが発行するクエリ

# 第9章　評価

本論文で実装したプロキシノードの有効性を示すため，課題を実現するための要件実現における関連研究との比較を定性的に評価し，シミュレーション実験により処理性能を評価した．本章では実験環境を示すとともに，評価の詳細を述べる．

## 9.1　関連研究との比較

プロキシノードは，異なるセンサネットワークを統合利用するアプリケーションを容易に実装するための機能を実装した．本節ではプロキシノードと関連研究とを機能面において比較するとともに，規模性，柔軟性の性質において比較し，定性評価としてまとめる．表 9.1 は関連研究と比較をまとめた結果である．以下に項目の詳細を述べる．

表 9.1　関連研究との比較

| | | プロキシノード | TinyDB | MANNA | LiveE! |
|---|---|---|---|---|---|
| 機能 | イベントトリガ ($A$) | ◯ | ◯ | (◯) | × |
| | データ集約 ($B$) | × | ◯ | × | ◯ |
| | クエリ集約 ($C$) | ◯ | × | × | × |
| 性質 | 柔軟性 ($D$) | ◯ | × | ◯ | △ |
| | 規模性 ($E$) | △ | × | △ | △ |

機能比較 ··· ◯：実装している，×：未実装である，(◯)：規定している
性質比較 ··· ◯：優れている，△：部分的に優れている，×：劣っている

($A$)　イベントトリガ

イベントトリガ機能によるアプリケーションの実装量変化について検討する．本実装ではイベントトリガ機能を実装した．

この機能を実装したことで，アプリケーションへの不要なセンサデータの送信を減らせるため，アプリケーションは受信するセンサデータを取捨選択する処理を少なくできる．設定する条件の内容によっては，他のクエリの返答と重ならず，各クエリごとに独立した集合の返答を受け取るように指定できる．

例えば，第 8.3 章で実装した環境モニタリングアプリケーションのように，温度データを三つの状態に分類し，状態に応じて視覚化する内容を変更したいという要求において，各クエリの返答は独立である．複数の独立したクエリによる返答処理は，複数のプロセスで並列に処理することができるため，他のプロセスによる処理結果を考慮せずに実装できる．従って，この機能は並列処理の実装を容易にする機能であると評価できる．

(*B*) データ集約

データ集約機能によるセンサデータ転送のトラフィック量変化について検討する．本実装ではデータ集約機能を実装していない．

センサネットワークから取得したセンサデータの最大値・最小値・総数・総和・平均・中央値を算出する機能がデータ集約機能であり，転送するトラフィック量を減らすことができる．削減できるトラフィック量は指定したクエリ条件に応じて変化し，多数のセンサノードからセンサデータを集める場合に高い効果が期待できる．

TinyDB はデータ集約機能を実装しており，センサネットワーク内トラフィック量の減少と，アプリケーションとセンサネットワークの間におけるトラフィック量の減少が同時に期待できる．同様に，LiveE!もこの機能を実装している．

一方，本実装でデータ集約機能を実装する場合，アプリケーションとの通信におけるトラフィック量の減少が期待できる．しかし，イベントトリガ機能により，アプリケーションとの通信におけるトラフィック量の減少が同様に期待できる．従って，この機能を実装することによるトラフィック量変化は，データ集約機能を実装している他の研究と比べて，大きく劣る結果にはならないと評価できる．

(*C*) クエリ集約

クエリ集約機能によるセンサデータ取得時間の変化について検討する．この機能を実装したことで，アプリケーションは他のアプリケーションのクエリを考慮することなく，自身のクエリを発行することができるようになる．この機能が存在しなかった場合，アプリケーション同士でクエリを把握するための仕組みや，ネットワーク内のノードに対して，常に全てのセンサデバイスの情報を取得し続けなければならないため，アプリケーションの実装コストとセンサネットワーク内のトラフィック量を減少できたと評価できる．

(*D*) 柔軟性

新しい種類のセンサネットワークを利用する場合において，アプリケーションに加える必要のある変更について検討する．本実装を利用するとセンサデータ取得方法とセンサデータフォーマットを統一して提供できるため，アプリケーションは該当する機能を変更しなくてよ

い．さらに，新しく追加するセンサのスペック情報を設定し，センサネットワークプラグインを作成すれば，本実装を拡張できる．TinyDB は MicaMote 以外のセンサネットワークを追加できないため，新しい種類のセンサネットワークへの柔軟性はないと評価できる．同様に，LiveE!は LonWorks ネットワークに対応していないセンサネットワークを追加できない．

従って，新たな種類のセンサネットワークを追加できる点と，追加におけるアプリケーションへの実装負担が少ないことから，本実装は関連研究と比べて柔軟性に優れていると評価できる．

(*E*) 規模性

本実装はアプリケーションに提供するセンサデータを全て中継するため，利用するセンサノード数の増加と比例して処理負荷が増える．また，中継に必要な処理を一元化して行うため，単一障害点となりうる．しかし一方で，アプリケーション数の増加に対しては，クエリを集約できるため処理負荷を抑えることができる．

TinyDB はセンサノード数の増加に比例して，センサデータの到達時間が増える．さらに，センサデータの要求数が二つまでであるため，アプリケーションの増加に対応できない．MANNA はセンサノードの管理に階層構造を用いているため，センサノード数の増加に対応できるが，アプリケーションの増加に対する処理負荷を考慮していない．同様に，LiveE!もアプリケーションの増加に対する処理負荷を考慮していない．

従って，本実装はセンサノードの増加に対応できないが，アプリケーションの増加に一定の範囲で対応できるため，他の研究と比較して，規模性が部分的に優れていると評価できる．

## 9.2 センサデータ待機時間の最適化性能

センサデータ要求のクエリに返答するプル型の通信方式のセンサネットワークにおいて，センサネットワークからクエリの返答を受信するまでの時間は，アプリケーションがセンサデータの返答を待つ時間の一部である．そのため，あるアプリケーションが，他のアプリケーションと同じセンサデータを要求する場合，クエリの集約機能により，アプリケーションがクエリの返答を待つ時間が短縮する．しかし，新たなセンサデータを要求する場合，センサネットワークからクエリの返答を待つ時間が必要となる．プル型の通信方式のセンサネットワークにおいて同様である．

本節ではアプリケーションがクエリの返答を待つ時間を短縮する機能を実験結果から評価する．実験では，プル型の通信方式の TinyDB での短縮時間を計測した．結果を考察し，同様の通信方式のセンサネットワークにおける，短縮機能の性能を示す．

TinyDB はネットワーク内のセンサノード数の増加に応じて，クエリがネットワーク内を行き渡るまでの時間が増えるため, 返答を待つ時間は増加する．この時間を, TOSSIM を用いて, ノード数の増加に対して計測した結果を図 9.1 に示す．送信したクエリは SELECT id ; である．センサネットワークからの返答時間は，プロキシノードがクエリを送信してから返答を受信するまでの時間を計測した．ID を要求するクエリは，指定するセンサデータの種類が一種類であり，条件指定がないため全てのセンサノードからセンサデータを取得できる要求である点で，最も処理負荷の少ない単純なクエリであるため，返答受信時刻を計測するのに適切なクエリであると考えた．図 9.1 の結果から，ノード数 $n$ に対して，返答時間が $R(n) = 4.6434n - 34.049$ で示せるよう近似できた．

一方，クエリ集約機能を実装した場合，クエリの発行順序によって返答を受け取るまでの時間が異なることを図 8.11 と図 8.12 に示した．例えば，クエリ $Q_a$ を発行した後，クエリ $Q_b$, $Q_c$ を 1 秒間隔で発行する．すると，プロキシノードは $Q_a$，$Q_{a+b}$ を発行した後， $Q_{a+b+c}$ をバッファする．その後，$Q_{a+b}$ の返答を受け取った後に，$Q_{a+b}$ を停止させ，$Q_{a+b+c}$ を発行する．結果，$Q_{a+b}$ の応答時間は $R(n)$ で，$Q_{a+b+c}$ の応答時間は $R(n)+1$ で表すことができる．以上から，応答時間は，クエリ $q$ を発行した時刻と，前のクエリ $q-1$ を発行した時刻の経過時間を $T(q)$ とすると，9.1 式で表せる．

$$
\begin{aligned}
R(n) &\simeq 4.6434n - 34.049 \\
R(n)\prime &= \begin{cases} 2 \cdot R(n) - T(q) & (T(q) < R(n)) \\ R(n) & (T(q) \geq R(n)) \end{cases}
\end{aligned} \tag{9.1}
$$

$$
\begin{aligned}
T(1) &= 0 \\
R_{reduce}(n) &= \begin{cases} R(n) - T(q) & (T(q) < R(n)) \\ 0 & (T(q) \geq R(n)) \end{cases}
\end{aligned} \tag{9.2}
$$

9.1 式から, 応答時間の短縮分 $R_{reduce}(n) = R(n)\prime - R(n)$ は 9.2 式で表せる．$R_{reduce}(n) \geq 0$ より，9.2 式からアプリケーションの返答待ち時間を全体的に短縮することができたと言える．

## 9.3 センサデータ処理性能

センサデータの処理性能を，シンクノードがセンサデータを送信してからアプリケーションがセンサデータを受信するまでの時間（**センサデータ処理時間**）における，プロキシノードがセンサデータを受信してからアプリケーションへ送信するまでの時間（**センサデータ処理時**

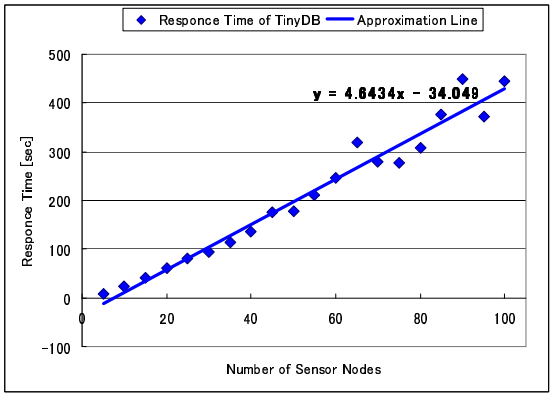


図 9.1　センサノード数変化における TinyDB のクエリ応答時間

**間**）の比率で示し，評価する．図 9.2 において，センサデータ処理時間を $T_A$ として示し，センサデータ到達時間を $T_B$ として示す．

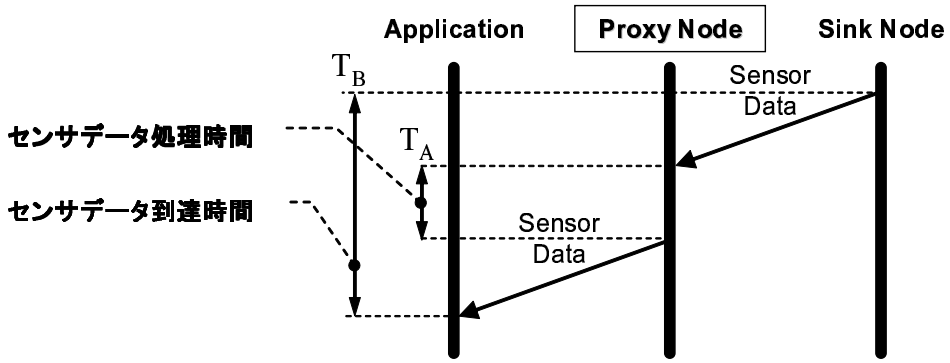


図 9.2　センサデータ処理時間とセンサデータ到達時間の定義

### 9.3.1　実験の目的

プロキシノードを使用すると，センサデータのフォーマット解釈処理により，センサデータ到達時間が増加する．増加する時間によってはセンサデータ取得の即時性がやや失われるため，アプリケーションが機能しなくなる場合がありうる．そのため，センサデータ処理性能を計測し，プロキシノードの実用性を評価する必要がある．

### 9.3.2 実験動作環境

実験は一台の PC において，本実装とセンサネットワークのシミュレータを動作させて行った．使用した PC の性能と，利用したシミュレータの詳細を表 9.2 に示す．センサノードのサンプリングレートは，uPart の最頻値 288ms と TinyDB を利用した MicaMote の最頻値 250ms を考慮し，250ms に設定した．

表 9.2 実験動作環境

| | 実験動作環境 |
|---|---|
| PC | CPU：Pentium 4 1.3GHz，Memory：256MB，OS：Debian 4.1 |
| TOSSIM | Version：tinyos 1.1.15Dec2005cvs-2，記述言語：nesC 1.1.2b-2 |
| TinyDB | サンプリングレート 250ms |
| simuPart | サンプリングレート 250ms |

### 9.3.3 計測結果

センサデータ処理時間は，図 8.2 で示したトポロジにおいて，プロキシノードが出力するログの時刻を基に計測した．プロセススイッチなどが結果に大きく関わらないようにするため，センサデータ処理をプロキシノード内部で 100 回繰り返し，$T_A = \frac{1}{100}(T_{END} - T_{BEG})$ として算出した．図 9.3 にログに出力した時刻とセンサデータ処理時間の関係を示す．一方，センサデータ到達時間は tcpdump コマンドで観測した通信ログから算出した．

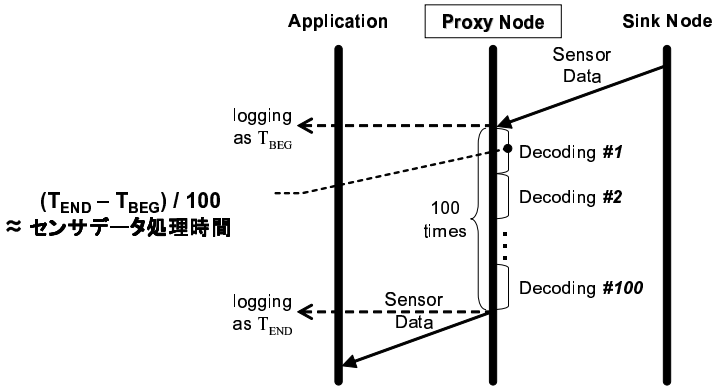


図 9.3 センサデータ処理時間の計測時のプロキシノード処理動作

センサデータの受信時間を計測した結果を図 9.4，図 9.5 に示し，まとめた結果を表 9.3 に

示す．図 9.5 はセンサネットワークプラグインのセンサデータ処理時間を表し，図 9.3 はセンサデータ処理時間とセンサデータ到達時間の比率を表す．図 9.5 より，µPart プラグインと TinyDB プラグインのセンサデータ処理時間は，それぞれ係数 0.0103 と 0.0029 の，一次関数式で近似できたことが読み取れる．そして，図 9.5 より，センサデータ到達時間に対して，センサデータ処理時間が十分に短いことが読み取れる．

計測結果により，センサノード数増加に対する処理時間増加率は低く，中継に必要となる処理時間は全体に対して十分に低いことが分かり，プロキシノードは実用性があると評価できる．

表 9.3　センサデータ処理時間の計測結果

| | センサデータ処理時間 ($T_A$) | センサデータ到達時間 ($T_B$) | 比率 ($T_A/T_B$) |
|---|---|---|---|
| µPart | 13.70 $\mu sec$ | 1510.38 $\mu sec$ | 0.91% |
| TinyDB | 5.87 $\mu sec$ | 750.33 $\mu sec$ | 0.79% |

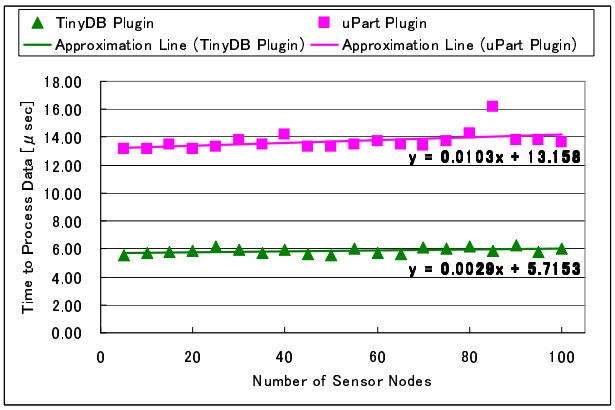


図 9.4　プラグインのセンサデータ処理時間

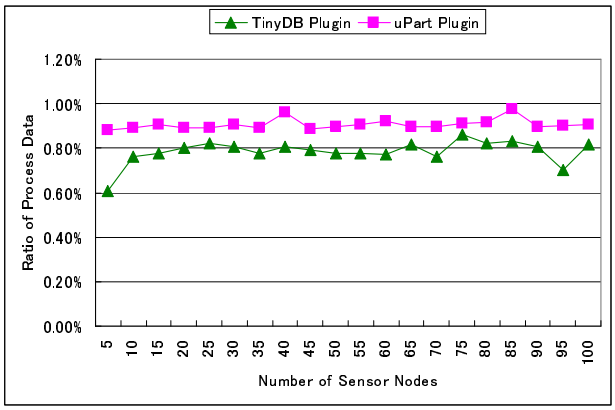


図 9.5　センサデータ処理時間とセンサデータ到達時間の比率

## 9.4　クエリ要求書式の記述表現力

第 7 章で定めたセンサデータ要求言語が，アプリケーションからの多様な要求を柔軟に記述できることを検討し，評価する．

本実装の SQL ライクな言語は，手続き型言語と比べて複雑な処理を記述できない．そのため，データマイニングなど，統計処理によってセンサデータから高次な情報を解析する目的に

は向いていないと言える．一方で，センサデータ値と整数値の四則演算，比較演算，論理和，論理積の論理演算が可能である．センサノードは，計算能力が潤沢に無い場合が多いため，ほとんどのセンサノードにおいて，センサデータを小数で表さない．そのため，小数を使わないセンサデータ値による条件指定は十分に記述できると考えられる．

要求書式の表現拡張性においては，SQL と同様に関係論理と関係代数に関する操作を単純な問い合わせで表現できる可能性を持つ．このため，本実装で述べたセンサデータを選択して要求する機能を，WHERE 句による単純な記述で表現できた．選択機能の他に，関係代数における射影と結合によって，複数のセンサネットワークを横断した要求を，単純な記述で表現できる可能性を持つ．その結果，携帯電話や PDA などの PC に比べて計算資源が乏しい機器においても，プロクシノードを通じて，多様にセンサデータを利用できる可能性があり，将来的な応用分野の発展が期待できる．

本論文において，潤沢な計算資源を持たないノードを，システムを構成するノードとして考慮した．そのため，シンクノードは本実装を利用するアプリケーションとなりうる．シンクノードがアプリケーションとして動作できれば，センサデータの値によって自身が構成するセンサネットワークの動作を動的に変更するなどの応用が可能となる．その点で SQL ライクな要求言語が持つ表現力は，センサネットワークを統合利用する場合において，手続き型言語を用いる場合と比べて柔軟な要求に対応できる可能性があると考えられる．

## 9.5 アプリケーション実装コストの軽減

室内環境アプリケーションの開発コストをソースコードの記述行数で示し，プロキシノードを使用した場合と，使用しなかった場合で比較評価する．利用したアプリケーションは第 8.3 節で述べた Java プログラムである．同じアプリケーションを二通りの場合で実装し，空行，コメント行を除いた行数を比較した．表 9.4 にソースコードの記述行数を示す．アプリケーションは複数の Java ソースコードファイルから構成され，センサデータの取得処理以外は内容が全く同じであるため，これらのファイルを除いて比較した．プロクシノードを使用した場合の動作を図 9.6 に示し，図 9.7 にプロクシノードを使用しなかった場合の動作を，アクティビティ図で示す．図から，プロキシノードを使用することで，センサデータの取得に関わる処理を減らすことができたことが分かる．

実装したアプリケーションは，プロキシノードが提供する基本的な機能を使用した．また，使用したセンサネットワークの数は二種類である．そのため，より多くの種類のセンサネットワークからセンサデータを取得し，より多くの要求を持つ場合，表 9.4 の結果以上にソースコードの記述量を減らすことができるのは明らかである．

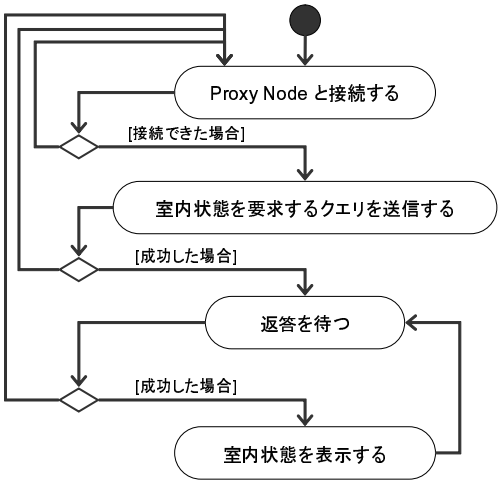


図 9.6 アプリケーションアクティビティ図（プロキシ使用）

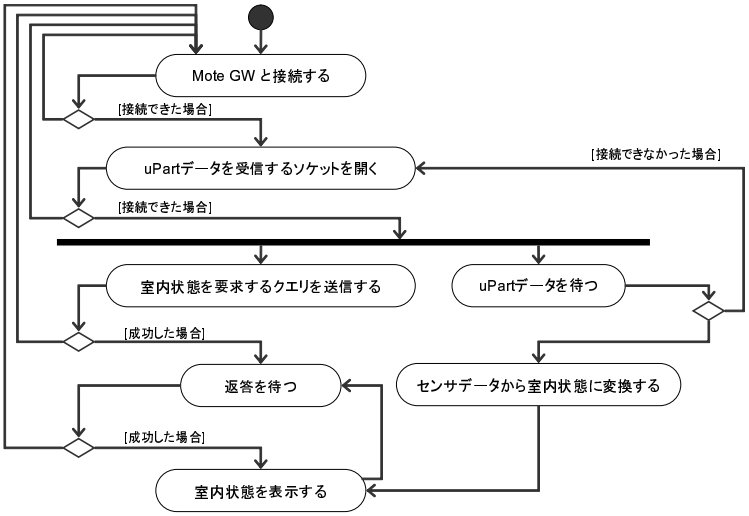


図 9.7 アプリケーションアクティビティ図（プロキシ未使用）

表 9.4 センサデータ取得処理におけるソースコード記述行数の比較

| プロキシノード使用 ($L_A$) | プロキシノード未使用 ($L_B$) | 減少割合 ($(L_B - L_A)/L_B$) |
|---|---|---|
| 280 (421*1) | 325 (466*1) | 13.8% |

*1 アプリケーション全体のソースコード記述行数

# 第10章　結論

本章では本論文の結論をまとめ，本研究の追加課題を検討する．

## 10.1　本論文のまとめ

本論文では多種類のセンサネットワークからセンサデータを取得するプロキシモデルを提案した．提案したプロキシモデルにおいて要件を整理し，設計，実装を行い，豊富な種類のセンサデータを利用するアプリケーションの容易な実装を可能にした．単純な要求を行う，一つのアプリケーションの開発コストを減らせたことから，インターネットを通じた複数センサネットワークを統合利用するアプリケーションの開発コストを，全体的に減らせると期待できる．

本実装ではアプリケーションとセンサネットワークの通信を仲介するプロキシノードを IP ネットワーク上に構築した．プロキシノードはアプリケーションインタフェースとセンサネットワークインタフェースの二つのインタフェースと，インタフェース動作を連結するインタフェースアダプタを持つ．アプリケーションインタフェースはアプリケーションに対し，豊富な種類のセンサデータを統一的に取得する機能を提供する．センサネットワークインタフェースはセンサネットワークからセンサデータを取得する機能を提供し，センサネットワークの種類ごとにセンサネットワークプラグインとして実装される．インタフェースアダプタは，アプリケーションからのセンサデータ要求を集約して，センサネットワークから効果的にセンサデータを取得する．センサネットワークをプラグインとして分けて実装できるよう設計し，プロキシノードに拡張性を持たせた．そして，センサデータを要求する言語として SQL ライクなクエリ言語を定義し，返答するセンサデータを条件式によりフィルタリングする機能を実装した．

本実装が，多種類のセンサネットワークを統一的に利用するアプリケーションを容易に実装できることを実証するため，二種類のセンサネットワークを利用するアプリケーションを，本実装を利用した場合と，利用しない場合の二通りで実装し，ソースコード記述行数を比較した．センサネットワークは µPart センサと XBridge で構成される種類と，TinyDB を実装した MicaMote で構成される種類の二種類の実験環境を構築した．アプリケーションは，二種類のセンサネットワークに共通して利用可能な温度と照度のセンサデータを取得し，室内の状態を判別して表示する室内環境モニタリングアプリケーションを実装した．

実証した結果，本実装を利用した場合と利用しなかった場合で，センサデータの取得処理に

関する実装コードに 13.8% の差があった．実証実験から，本実装により多くの実装コードが必要であったセンサシステムアプリケーションが容易に実装できることを確認した．さらに，提案したプロキシモデルでは，アプリケーションはセンサデータを統一的に取得できるため，新たなセンサネットワークの追加により再実装しなければならない処理は少ない．そのため，新規開発による実装負担と，利用するセンサ構成の変化による再実装負担を軽減でき，利用するセンサ構成の拡張性があると評価できる．

複数のアプリケーションが本実装を利用する場合において，機能を追加しつつ，効率を落とさずにセンサデータを中継できることを検証するため，シミュレータを用いてプロキシノードの処理性能を計測した．計測には，TinyDB を実装した MicaMote で構成されるセンサネットワークをシミュレートする TOSSIM と，µPart と XBridge で構成されるセンサネットワークをシミュレートする simuPart の二つを用いた．TOSSIM と simuPart は，センサネットワークがセンサデータを送信する動作を，データフォーマットと返答に要する時間に関して擬似的に行い，センサノード数を変更できる．プロキシノードによるセンサデータ中継の実用性を検証するため，センサネットワークプラグインのセンサデータ処理時間を，センサノード数変化に対して計測した．

計測した結果，TinyDB プラグインと µPart プラグインはそれぞれ，センサノード変化に対する応答時間は一次関数増加し，関数の係数は，TinyDB プラグインが 0.0029 に近似でき，µPart プラグインが 0.0103 に近似できた．そして，センサネットワークからアプリケーションまでのセンサデータ送信時間に対するセンサデータ処理時間は，TinyDB プラグインが 0.79% であり，µPart プラグインが 0.91% であり，共に 1% 未満であることを確認した．計測結果により，センサノード数増加に対する処理時間増加率は低く，中継に必要となる処理時間は全体に対して十分に低いことが分かり，プロキシノードの実用性を確認できた．

プロキシノードを既存研究と比較し，拡張性に優れるとともに，規模性にやや優れていると評価した．プロキシノードを利用することで，拡張性と規模性を必要とする環境モニタリングアプリケーションの実装負担を，特に軽減できることが期待できる．将来，本研究を応用することで，アプリケーション開発に特別な技術が必要なくなり，実空間情報を提供する活動が促進する結果，より多くの利用者が自身に有益な活動の補助などへの活用が期待できる．

## 10.2 今後の課題

本節では，本研究における今後の課題を述べる．本実装には，単一障害点の解決と直感的なセンサデータの指定が実現できていないという課題がある．課題を解決するための要件を以下に詳しく述べる．

### 10.2.1 単一障害点の解決

本実装は，アプリケーションとセンサネットワークとの通信を中継することでアプリケーションの実装を容易にしている反面，プロキシノードがアプリケーションの動作に不可欠となる点で単一障害点となりうる．シンクノードもセンサネットワークからセンサデータを取得する機構における単一障害点となるが，複数のシンクノードを提供できるセンサネットワーク技術によって補える．現状で本実装が対応したセンサネットワークは一つのシンクノードを前提としたが，将来的には複数のシンクノードを持つセンサネットワーク技術に対応できることが望ましい．

この課題は，障害検知と障害回避または障害復旧が可能な構成により，プロキシノードの機能を冗長化することで解決できると考えられる．障害を検知する方法には，正常動作状態を示す Hertbeat Message を通知し続け，一定時間メッセージが来ない場合を障害が起こったと判断する方法がある [29]．また，障害を回避する方法には，等しい機能を持つ複数のノードによりシステムを構成し，あるノードが停止した場合に他のノードで代替する方法がある．さらにプロキシノードとアプリケーションとの通信におけるネットワーク障害を回避する方法も研究されている [30]．これらの機能を本実装と連携させるか，同等の機能を実装することで，この課題を解決し，アプリケーションへ提供する機能を冗長化できるだろう．

### 10.2.2 直感的なセンサデータの指定

センサデータの記述方法として，人が指定しやすい直感的な指標であることは，本研究の想定する環境において重要である．本論文では，基本的な指標として，センサデータの種類で指定する方法を採用した．しかし，この方法では，対象となるセンサデータを十分適切に指定できているとは言えない．例えば，温度データを指定した場合，プロキシノードと接続された全てのセンサネットワークから温度データを収集するため，広範囲に構築されたセンサネットワークが接続されている場合，大量のセンサデータを取得することとなる．そのため，新たな指標の追加により指定するセンサデータ数を今以上に絞り込む必要がある．新たな指標は直感的であることが利便性の観点から望ましい．

新たな指標の例として，センサネットワークやセンサノードの設置場所を指定する方法がある．アプリケーションの要求を考えると，センサデータの種類以外に，設置場所を大まかなに知っていると想定することができるため，設置場所は今後考慮すべき指標である．しかし，プロキシノードで接続された全てのセンサネットワークやセンサの設置場所を記憶して，検索を

行うのは大変な処理負荷となる．また，設置場所を示す値として，絶対座標，相対座標，名前の内，どの値を使うのかによって，設置場所記憶時のデータ量，管理の容易さ，識別可能になる情報量などが変わるため，様々な議論が必要である．

# 付録A　スペック情報の定義

```
enum column_type {  /* センサデータ値の情報 */
  COLUMNT_ID,COLUMNT_LOCATION,COLUMNT_BATTERY,COLUMNT_TEMPERATURE,COLUMNT_LIGHT,
  COLUMNT_HUMIDITY,COLUMNT_PRESSURE,COLUMNT_MOTION_1,COLUMNT_MOTION_2,COLUMNT_MOTION_3,
  COLUMNT_NONE,COLUMNT_NULL,COLUMNT_MAX
};
enum unit_type {    /* センサデータ値の単位 */
  UNITT_VOLTAGE,UNITT_CELCIOUS,UNITT_NM,UNITT_GRAVITY,UNITT_RH,UNITT_MBAR,UNITT_TIMES,
  UNITT_NONE,UNITT_NULL,UNITT_MAX
};
enum tlv_type {     /* センサデータ値のデータタイプ */
  TLVT_INTEGER = 0x8000, TLVT_CHAR    = 0x4000, TLVT_STRING  = 0x2000,
  TLVT_BIT     = 0x1000, TLVT_USER    = 0x0000, TLVT_NULL    = 0xFFFF
};

struct datum_spec { /* スペック情報構造体 */
  enum column_type column_type;
  enum tlv_type tlv_type;
  enum unit_type unit_type;
  int len;
};
```

図 A.1　スペック情報の定義

| スペック情報の項目 | 項目のとりうる値 | 値の説明 |
|---|---|---|
| column_type | センサデータ値の情報 | |
| | COLUMNT_ID | 識別子 |
| | COLUMNT_LOCATION | 位置 |
| | COLUMNT_BATTERY | バッテリ残量 |
| | COLUMNT_TEMPERATURE | 温度 |
| | COLUMNT_LIGHT | 照度 |
| | COLUMNT_HUMIDITY | 湿度 |
| | COLUMNT_PRESURE | 圧力 |
| | COLUMNT_MOTION_1 | 加速度 x 軸 |
| | COLUMNT_MOTION_2 | 加速度 y 軸 |
| | COLUMNT_MOTION_3 | 加速度 z 軸 |
| | COLUMNT_NONE | 情報なし |
| | COLUMNT_NULL | 不正値 |
| unit_type | センサデータ値の単位 | |
| | UNITT_VOLTAGE | ボルト ($V$) |
| | UNITT_CELCIOUS | 摂氏 (℃) |
| | UNITT_NM | ナノメートル ($nm$) |
| | UNITT_GRAVITY | 重力 ($G$) |
| | UNITT_RH | 相対湿度 (%) |
| | UNITT_MBAR | ミリバール ($mbar$) |
| | UNITT_TIMES | 単位なし (回) |
| | UNITT_NONE | 単位なし |
| | UNITT_NULL | 不正値 |
| tlv_type | センサデータ値のデータタイプ | |
| | TLVT_INTEGER | 整数 |
| | TLVT_CHAR | 文字 |
| | TLVT_STRING | ASCII 文字列 |
| | TLVT_BIT | バイナリ |
| | TLVT_USER | ユーザ指定 |
| | TLVT_NULL | 不正値 |
| 〈$int$ 型〉 len | センサデータ値のデータ長 | |

図 A.2　スペック情報の定義詳細

# 付録 B　評価用スペック情報の詳細

```
struct datum_spec mica_spec[] = {
  {COLUMNT_ID,          TLVT_INTEGER, 2, UNITT_NULL},     // node id
  {COLUMNT_LIGHT,       TLVT_INTEGER, 2, UNITT_NM},       // illuminance
  {COLUMNT_TEMPERATURE, TLVT_INTEGER, 2, UNITT_CELCIOUS}, // temperature
  {COLUMNT_MOTION_1,    TLVT_INTEGER, 2, UNITT_GRAVITY},  // motion x
  {COLUMNT_MOTION_2,    TLVT_INTEGER, 2, UNITT_GRAVITY},  // motion y
  {COLUMNT_HUMIDITY,    TLVT_INTEGER, 1, UNITT_RH},       // % of relative humidity
  {COLUMNT_PRESSURE,    TLVT_INTEGER, 2, UNITT_MBAR}      // barometric pressure
};
```

図 B.1　TinyDB(MTS310) のスペック情報

```
struct datum_spec upart_spec[] = {
  {COLUMNT_NULL,        TLVT_NULL,    1, UNITT_NULL},     // version
  {COLUMNT_LOCATION,    TLVT_BIT,    44, UNITT_NULL},     // location
  {COLUMNT_ID,          TLVT_BIT,     8, UNITT_NULL},     // id
  {COLUMNT_NULL,        TLVT_NULL,    1, UNITT_NULL},     // sequence
  {COLUMNT_NULL,        TLVT_NULL,    2, UNITT_NULL},     // concom
  {COLUMNT_NULL,        TLVT_NULL,    1, UNITT_NULL},     // length
  {COLUMNT_NULL,        TLVT_NULL,    3, UNITT_NULL},     // config
  {COLUMNT_BATTERY,     TLVT_INTEGER, 1, UNITT_VOLTAGE},  // battery
  {COLUMNT_MOTION_1,    TLVT_INTEGER, 1, UNITT_TIMES},    // motion
  {COLUMNT_TEMPERATURE, TLVT_INTEGER, 1, UNITT_CELCIOUS}, // temperature
  {COLUMNT_LIGHT,       TLVT_INTEGER, 1, UNITT_NM},       // illuminance
};
```

図 B.2　µPart のスペック情報

# 付録C　評価実験結果データ詳細

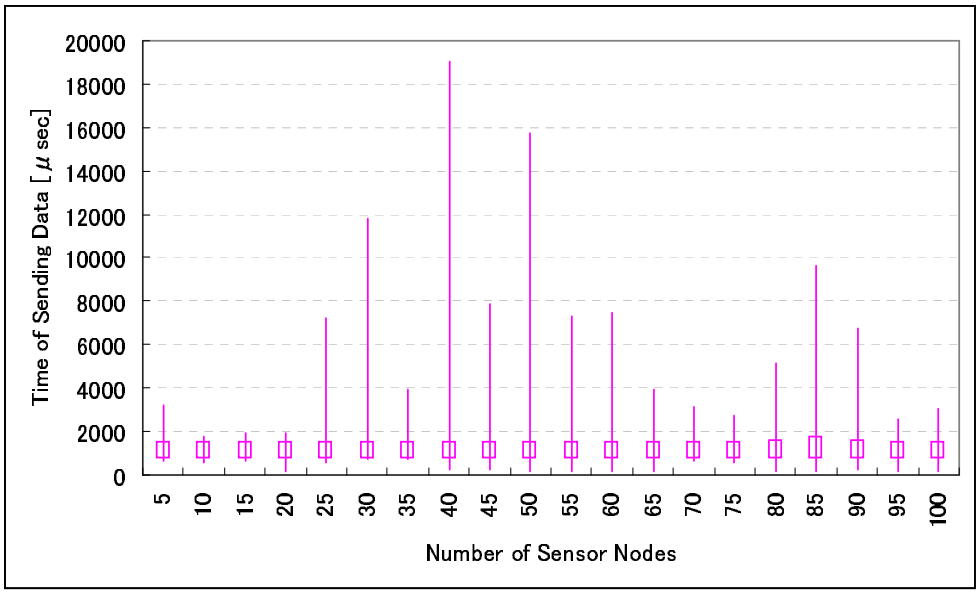


図 C.1　μPart のセンサデータ到達時間

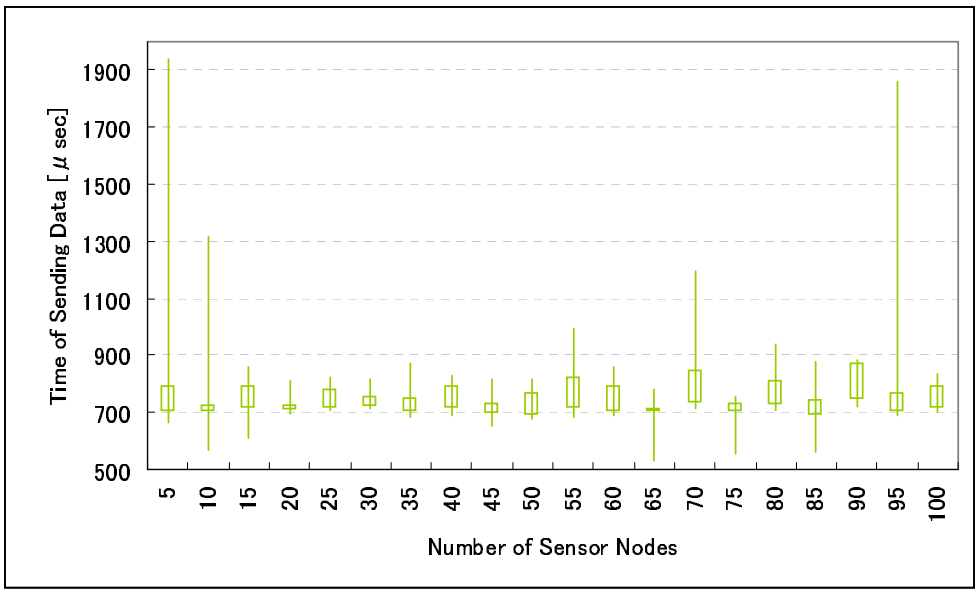


図 C.2　TinyDB のセンサデータ到達時間

| Number of Nodes | Time of Decoding Sensor Data (µsec) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MICAz Plug-in | | | | | µPart Plug-in | | | | |
| | Min[*1] | Max[*2] | Avg[*3] | Med[*4] | Std Deviation[*5] | Min[*1] | Max[*2] | Avg[*3] | Med[*4] | Std Deviation[*5] |
| 5 | 5 | 6 | 5.5833 | 6 | 2.5442 | 12 | 37 | 13.239 | 13 | 3.8305 |
| 10 | 5 | 7 | 5.6875 | 6 | 2.5186 | 13 | 28 | 13.223 | 13 | 3.8072 |
| 15 | 5 | 6 | 5.8182 | 6 | 2.4192 | 13 | 18 | 13.535 | 13 | 3.7917 |
| 20 | 5 | 6 | 5.8333 | 6 | 2.3887 | 12 | 18 | 13.241 | 13 | 3.8043 |
| 25 | 6 | 7 | 6.1818 | 6 | 2.3394 | 13 | 21 | 13.321 | 13 | 3.7686 |
| 30 | 5 | 6 | 5.9091 | 6 | 2.3321 | 13 | 25 | 13.839 | 14 | 3.5811 |
| 35 | 5 | 6 | 5.6923 | 6 | 2.4448 | 12 | 39 | 13.542 | 13 | 3.7018 |
| 40 | 5 | 7 | 5.9286 | 6 | 2.3688 | 12 | 3501 | 14.161 | 13 | 3.9440 |
| 45 | 5 | 6 | 5.6364 | 6 | 2.5347 | 12 | 191 | 13.333 | 13 | 3.8304 |
| 50 | 5 | 6 | 5.5833 | 6 | 2.5454 | 12 | 158 | 13.331 | 13 | 3.8400 |
| 55 | 6 | 6 | 6.0000 | 6 | 2.2883 | 12 | 994 | 13.518 | 13 | 3.8520 |
| 60 | 5 | 7 | 5.7273 | 6 | 2.5602 | 12 | 1020 | 13.690 | 13 | 3.8856 |
| 65 | 5 | 7 | 5.6000 | 5 | 2.6573 | 12 | 1004 | 13.513 | 13 | 3.8227 |
| 70 | 6 | 7 | 6.0833 | 6 | 2.3070 | 12 | 1003 | 13.380 | 13 | 3.8555 |
| 75 | 5 | 7 | 6.0000 | 6 | 2.3924 | 12 | 2079 | 13.707 | 13 | 3.8676 |
| 80 | 6 | 7 | 6.2000 | 6 | 2.3527 | 13 | 144 | 14.254 | 14 | 3.6538 |
| 85 | 5 | 7 | 5.8333 | 6 | 2.4844 | 15 | 1018 | 16.205 | 16 | 4.0086 |
| 90 | 6 | 7 | 6.3000 | 6 | 2.4056 | 12 | 989 | 13.767 | 13 | 3.8560 |
| 95 | 5 | 6 | 5.7692 | 6 | 2.4035 | 13 | 95 | 13.775 | 13 | 3.7551 |
| 100 | 5 | 7 | 6.0000 | 6 | 2.4253 | 12 | 2058 | 13.584 | 13 | 3.8498 |

[*1]Minimum, [*2]Maximum, [*3]Average, [*4]Median, [*5]Standard Deviation

表 C.1　シミュレート環境におけるセンサデータ処理時間

| Number of Nodes | Time of Receiving Sensor Data (µsec) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MICAz Plug-in | | | | | µPart Plug-in | | | | |
| | Min[*1] | Max[*2] | Avg[*3] | Med[*4] | Std Deviation[*5] | Min[*1] | Max[*2] | Avg[*3] | Med[*4] | Std Deviation[*5] |
| 5 | 666 | 1942 | 853.3 | 698.5 | 16.875 | 1413 | 3182 | 1359.2 | 1475.0 | 0.014 |
| 10 | 565 | 1320 | 699.5 | 709.0 | 9.743 | 1392 | 1794 | 1266.3 | 1456.0 | 0.035 |
| 15 | 610 | 859 | 675.5 | 734.0 | 5.625 | 1404 | 1948 | 1088.0 | 1472.0 | 0.055 |
| 20 | 698 | 809 | 665.8 | 720.0 | 0.002 | 188 | 1921 | 1095.1 | 1454.0 | 5.625 |
| 25 | 710 | 826 | 681.0 | 717.0 | 0.002 | 593 | 7210 | 1253.9 | 1468.0 | 5.625 |
| 30 | 713 | 816 | 679.0 | 727.0 | 0.002 | 1445 | 11846 | 1078.5 | 1497.0 | 0.080 |
| 35 | 683 | 872 | 674.5 | 706.0 | 0.002 | 1423 | 3973 | 762.0 | 1488.0 | 0.093 |
| 40 | 691 | 832 | 690.7 | 716.5 | 0.002 | 214 | 19044 | 1152.9 | 1456.0 | 25.778 |
| 45 | 655 | 818 | 653.3 | 706.0 | 0.002 | 253 | 7837 | 689.5 | 1471.0 | 5.626 |
| 50 | 675 | 819 | 669.9 | 702.5 | 0.002 | 183 | 15717 | 468.2 | 1460.0 | 5.626 |
| 55 | 684 | 996 | 709.4 | 734.0 | 9.743 | 154 | 7295 | 494.5 | 1464.0 | 7.956 |
| 60 | 686 | 859 | 683.9 | 719.0 | 0.002 | 181 | 7505 | 685.2 | 1465.0 | 25.778 |
| 65 | 529 | 781 | 628.5 | 704.0 | 5.625 | 180 | 3940 | 593.7 | 1476.0 | 12.579 |
| 70 | 716 | 1195 | 748.9 | 734.0 | 12.578 | 624 | 3145 | 391.4 | 1465.0 | 0.106 |
| 75 | 555 | 755 | 640.7 | 712.0 | 5.625 | 670 | 2695 | 286.1 | 1469.0 | 0.096 |
| 80 | 707 | 936 | 700.1 | 735.0 | 0.002 | 169 | 5136 | 328.7 | 1556.0 | 13.779 |
| 85 | 558 | 879 | 536.1 | 699.5 | 0.003 | 136 | 9606 | 430.0 | 1720.0 | 38.152 |
| 90 | 719 | 887 | 722.5 | 777.0 | 5.625 | 226 | 6745 | 199.1 | 1500.0 | 0.087 |
| 95 | 692 | 1860 | 763.2 | 703.0 | 16.875 | 133 | 2587 | 288.4 | 1519.0 | 15.911 |
| 100 | 700 | 838 | 677.4 | 737.0 | 0.002 | 154 | 3026 | 345.4 | 1489.0 | 14.883 |

[*1]Minimum, [*2]Maximum, [*3]Average, [*4]Median, [*5]Standard Deviation

表 C.2　シミュレート環境におけるセンサデータ到達時間

# 付録Ｄ　センサデータ要求書式詳細

$$
\begin{aligned}
\{\text{ 問い合わせ指定 }\} ::= & \ SELECT \left[ * \middle| \{\text{ センサデータの種類 }\} \right] \left[ \{WHERE \text{ 句 }\} \right]; \\
\{\text{ センサデータの種類 }\} ::= & \ \text{スペック情報で定義されたセンサデータの種類} \\
\{WHERE \text{ 句 }\} ::= & \ WHERE \{\text{ 選択条件 }\} \\
\{\text{ 選択条件 }\} ::= & \ \{\text{ ブール項 }\} \Big| \{\text{ 選択条件 }\} \ OR \{\text{ ブール項 }\} \\
\{\text{ ブール項 }\} ::= & \ \{\text{ ブール因子 }\} \Big| \{\text{ ブール項 }\} \ AND \{\text{ ブール項 }\} \\
\{\text{ ブール因子 }\} ::= & \ \left[ NOT \right] \{\text{ 比較述語 }\} \\
\{\text{ 比較述語 }\} ::= & \ \{\text{ 値式 }\} \{\text{ 比較演算子 }\} \{\text{ 値式 }\} \\
\{\text{ 値式 }\} ::= & \ \{\text{ 項 }\} \Big| \{\text{ 値式 }\} + \{\text{ 項 }\} \Big| \{\text{ 値式 }\} - \{\text{ 項 }\} \\
\{\text{ 項 }\} ::= & \ \{\text{ 因子 }\} \Big| \{\text{ 項 }\} * \{\text{ 因子 }\} \Big| \{\text{ 項 }\} \ / \{\text{ 因子 }\} \\
\{\text{ 因子 }\} ::= & \ \left[ + \middle| - \right] \{\text{ 一次子 }\} \\
\{\text{ 一次子 }\} ::= & \ \{\text{ 整数値 }\} \Big| \{\text{ センサデータの種類 }\} \\
\{\text{ 比較演算子 }\} ::= & \ < \ \Big| \ > \ \Big| \ <= \ \Big| \ >= \\
\{\text{ 整数値 }\} ::= & \ -2,147,483,647 \sim \ +2,147,483,647
\end{aligned}
$$

図 D.1　アプリケーションインタフェースにおけるセンサデータ要求書式 (BNF 記法)

# 謝辞

本論文執筆にあたり，ご指導いただきました慶應義塾大学環境情報学部教授 村井純博士，並びに同学部教授 徳田英幸博士，同学部教授 中村修博士，同学部准教授 楠本博之博士，同学部准教授 高汐一紀博士，同学部専任講師 重近範行博士，同学部専任講師 湧川隆次博士，同学部専任講師 Rodney D. Van Meter 博士に感謝致します．

また，研究を進めるにあたり，ご指導とご助言をいただきました慶應義塾大学環境情報学部准教授 三次 仁博士，並びに慶應義塾大学大学院政策・メディア研究科助教 鈴木茂哉氏，同学科講師 南政樹氏，同学科助教 稲葉達也氏，同学科助教 中根雅文氏，同学科講師 羽田久一博士，慶應義塾大学 SFC 研究所訪問研究員 徳増理氏に感謝致します．特に，鈴木茂哉氏には本論文の執筆の際に，ご指導ご鞭撻いただきました．ご多忙な予定の中，親身にご指導いただけたことを深謝致します．氏なしには卒業論文は完成し得ませんでした．また，南政樹氏には研究生活の二年間に渡り，様々な活動を通してご指導ご鞭撻いただきました．広い見聞により手厚くご指導いただけたことを深謝いたします．

そして，慶應義塾大学環境情報学部徳田・村井合同研究室の研究生である小原泰弘氏，川喜田佑介氏，奥村佑介氏，苧阪浩輔氏，金井瑛氏，神谷尚保氏，空閑洋平氏，榊原寛氏，中井彦一郎氏，水谷正慶氏，山本彰氏，江村桂吾氏，尾崎隆亮氏，金仙麗氏，佐藤泰介氏，波多野敏明氏，福井達士氏，藤本俊佑氏，山口修平氏，山田真弘氏に感謝します．特に，小原泰弘氏には研究発表，論文執筆の際に様々な御助言を幾度と無くいただきました．また，空閑洋平氏には本論文の執筆の際に様々な助言をいただきました．そして，榊原寛氏，中井彦一郎氏には，本研究の実験への御助力をいただきました．

研究の場として OB/OG の諸先輩方が築き上げてきた研究団体の SING/IA*，RING，Auto-ID，MOVE!に感謝します．

最後に，私生活を支え，学びの場を与えてくれた父と母，そして，安らぎの場を築いてくれた祖父，祖母，姉，兄，友人への感謝を持って謝辞といたします．

2008 年 1 月 24 日

佐藤 龍

# 参考文献


[1] Beluga Project. *http://beluga.sfc.keio.ac.jp/*, May 2007.

[2] SSLab: Smart Space Laboratory Project. *http://www.ht.sfc.keio.ac.jp/SSLab/*, May 2007.

[3] SMART DUST. *http://robotics.eecs.berkeley.edu/~pister/SmartDust/*, January 2008.

[4] About the Center For Future Health. *http://www.futurehealth.rochester.edu/*, January 2008.

[5] Great Duck Island 2003. *http://www.coa.edu/html/greatduckisland2003.htm*, January 2008.

[6] mediacup@teco. *http://mediacup.teco.edu/*, January 2008.

[7] Live E! ～活きた地球の環境情報～. *http://www.live-e.org/*, January 2008.

[8] Smart Kindergarten - Home -. *http://nesl.ee.ucla.edu/projects/smartkg/*, January 2008.

[9] D.E. Hill, J.L. Culler. Mica: a wireless platform for deeply embedded networks. *Micro, IEEE*, Vol. 22, pp. 12– 24, Nov/Dec 2002.

[10] Michael Beigl, Christian Decker, Albert Krohn, Till Riedel, and Tobias Zimmer. µParts: Low Cost Sensor Networks at Scale. *Ubicomp*, September 2005.

[11] SMART-ITS — HOME. *http://www.smart-its.org/*, January 2008.

[12] 永原崇範, 鹿島拓也, 猿渡俊介, 川原圭博, 南正輝, 森川博之, 青山友紀, 篠田庄司. ユビキタス環境に向けたセンサネットワークアプリケーション構築支援のための開発用モジュール U3（U-cube）の設計と実装. 電子情報通信学会信学技報 情報ネットワーク (IN), Mar 2003.

[13] IEEEP1451. IEEE Std P1451.0/D6.04, 2006.

[14] Linnyer Beatrys Ruiz, José Marcos Nogueira, and Antonio A. F. Loureiro. MANNA: a management architecture for wireless sensor networks. *Communications Magazine, IEEE*, Vol. 41, No. 2, pp. 116–125, Feb 2003.

[15] *Collecting Adaptive Data for Isolated Wireless Sensors with Patrol Nodes in Live E!*, May 2006.

[16] Sam Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB:

An Acqusitional Query Processing System for Sensor Networks. *ACM TODS*, 2005.

[17] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, Vol. 31, No. 3, pp. 9–18, 2002.

[18] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, August 2000.

[19] UC Berkeley. TinyOS Community Forum —— An open-source OS for the networked snesor regime. *http://www.tinyos.net/*, May 2007.

[20] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*, pp. 186–201, New York, USA, 1999. ACM Press.

[21] Mike Esler, Jeffrey Hightower, Tom Anderson, and Gaetano Borriello. Next century challenges: data-centric networking for invisible computing: the Portolano project at the University of Washington. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 256–262, New York, USA, 1999. ACM Press.

[22] Philippe Bonnet, J. E. Gehrke, and Praveen Seshadri. Towards Sensor Database Systems. *Proceedings of the Second International Conference on Mobile Data Management*, January 2001.

[23] ECHELON Japan, Lonworks. *http://www.echelon.co.jp/products/lonworks.html*, May 2007.

[24] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, Vol. 36, No. SI, pp. 131–146, 2002.

[25] Open Geospatial Consortium. OpenGISR Sensor Model Language (SensorML): Request for Public Comments — OGCR. *http://www.opengeospatial.org/standards/requests/31/*.

[26] EPC Information Services (EPCIS) Version 1.0.1 Specification. *http://www.epcglobalinc.org/standards/epcis/epcis_1_0_1-standard-20070921.pdf*, September 2007.

[27] EPCGlobal. *http://www.epcglobal.com/*, December 2007.

[28] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 126–137, New York, USA, 2003. ACM.

[29] Bidirectional Forwarding Detection (bfd) Charter. *http://www.ietf.org/html.charters/bfd-charter.html*, January 2008.

[30] Victor C. Zandy and Barton P. Miller. Reliable network connections. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pp. 95–106, New York, USA, 2002. ACM.