修　士　論　文

# 広域センサネットワークの運用構造と多属性検索の実現

Operating Architecture and Multi-Attribute Search for Wide Area Sensor Networks

指導教員　江崎浩 教授

東京大学大学院　情報理工学系研究科
電子情報学専攻

氏名　66410　落合　秀也

提出日　2008 年 2 月 4 日

# 概要

近年，組込みコンピュータとインターネットの融合により，小型な設備で遠隔地の環境情報を観測することが可能になった．いまや多くのセンサがインターネットに接続され，結果として広域センサ網ができつつある．ところが実際には，これらのセンサシステムは運用組織ごと独自な設計で開発されているのが現状で，インターネットのアプリケーションとしては標準化が進んでいない．結局，多くのセンサデータがオンラインなのにもかかわらず，その一部しか同時には利用できない状況にある．それぞれのセンサデータは，気象観測，災害対策，農業など多くの場面で再利用できる可能性を秘めているが，これを実現するには，データ共有を妨げている上記の現状を打開しなければならない．独自な設計で開発される要因の一つには，利用目的に応じてセンサの表現方法や扱い方が多様であり，標準的な形を決定することが困難な点が挙げられる．本研究では，組織単位の運用による分散的な環境を踏まえた上で，アプリケーションに応じたスキーマを運用時に動的に更新可能にする．その上で，各種アプリケーション属性による広域検索を実現するための，多属性検索手法を提案する．アーキテクチャの提案においては，実際的な広域センサネットワークでの運用に求められる，自律分散性，データ提供先のアクセスコントロール，センサの管理認証，多言語によるセンサ表現，耐故障性の実現など，各種要求を考慮に入れている．本研究の段階では，広域センサネットワークの分散運用方式やその上での多属性検索を実現する基礎的な枠組みを提案することに焦点を当て，スケール性を実現する方法に関しては今後の課題としている．

本提案システムを実装し，検証することで，これらの要求を満足するシステムが実際に構築可能であることが判明した．組織名による検索の他，経緯度，移動体名，河川名などの各種属性をキーとした検索を実現できた．新規センサアプリケーションの導入に伴う新規属性をキーとした検索は，スキーマを変更することで柔軟に対応できた．センサの認証管理機能により，ゴミとして登録されるセンサを 49.8%から 3%にまで減少できた．組織 A にはデータ提供するが，組織 B にはデータ提供しないといった，データ提供先のアクセスコントロールも実現できた．

検索クエリの経路アグリゲーションへの挑戦，経路分岐の削減，ポーリング型とイベント通知型の共存などが検索パフォーマンスや通信形態の課題として残されている．運用上の課題としては，広域センサネットワークのトポロジや，プロファイルのスキーマの管理運用計画などがある．ユーザアプリケーションにとっての利便性や，データ処理や転送の効率を高めるため，データフィルタリングやデータアグリゲーションも必要である．センサ動作の監視システムのあるべき姿の模索，インターネットから孤立したセンサのデータを回収する技術や運用に関する検討なども残された課題である．

本研究で開発されたシステムは，Live E!広域センサネットワークとして，2008 年 1 月現在，東京大学，奈良先端科学技術大学院大学，Taiwan Network Information Center(TWNIC)，Prince of Songkla University(PSU; タイ，プーケット)の 4 拠点で，合計 9 台のサーバで分散運用されている．Live E! システムには，約 100 台のインターネット気象センサが登録されており，常時気象データが収集されアプリケーションへ提供されている．今後も広域センサネットワークの研究プラットフォームとして運用が継続される計画である．

# 目次

# 第１章　はじめに

近年，組込みコンピュータや通信デバイスの小型化，高性能化により，インターネットを利用して遠隔地の環境情報の観測が現実に行われてきている．様々な構成のセンサ機器や通信端末を用いたセンサがインターネットに接続され，結果としてかなりの数のセンサから環境情報が電子化され，インターネットを使って収集されていると推定される．ところが実際には，これらのセンサデータの収集システムはセンサの運用組織ごと独自な設計で開発運用されているのが現状で，システム間の相互接続性は確保できていない．多くのセンサデータがオンラインなのにもかかわらず，ユーザアプリケーションはその一部しか利用できない状況にある．それぞれの組織で集められたセンサデータは，気象観測，災害対策，農業など多くの場面で再利用できる可能性を秘めている．例えば，農場に設置された雨量センサが，洪水対策にも用いられることが考えられるし，小中学校の教材として設置される百葉箱で観測されたデータが，災害対策に加えて，ヒートアイランド現象の解析にも用いられるかも知れない．環境情報の流通が新たなビジネスを生み出す可能性もある．組織の枠を超えて環境情報の再利用を実現するには，組織間でのセンサシステムの相互接続性を確保する必要がある．一方で，実際的な広域センサネットワーク運用においては，データ提供先のアクセスコントロールが可能であることが要求される．アプリケーションに応じて広域に分散したセンサデータを種々観点から検索できるようになっている必要もある．本研究では，このような状況を踏まえた広域センサネットワークのアーキテクチャを提案する．

アーキテクチャの提案においては，実際的な広域センサネットワークでの運用に求められる，自律分散性，データ提供先のアクセスコントロール，センサの管理認証，多言語によるセンサ表現，耐故障性の実現など，各種要求を考慮に入れる．分散環境上での検索の実現においては，利用目的に応じて，スキーマを動的に更新できるようにしなければならない．

本研究で想定するセンサシステムは，遠隔地のセンサによって観測されたデータが，インターネットを介して，センサの指定するサーバに定期的に通知される方式のものである．センサ設置および運用は，複数の組織によって独立に行われ，それぞれの組織でデータ収集サーバが運用されている．実際的なセンサシステムにおいては，収集されたデータを，外部組織へ提供する際には，ユーザ(外部組織)の認証が必要な場合が多い．これは，設備投資をしてデータ収集の運用を行っている組織にとって自然なことであり，広域センサネットワークでは，この状況を想定しなければならない．

それぞれの組織のサーバが異なるスキーマで独自に構築されている場合，これらを相互接続するために Broker を導入する方法がよく知られている(Mediator もしくは Agent と呼ばれることもある)．この方式では，アクセスインタフェースやセンサデータの表現を統一するため，Broker と各サーバ間に Wrapper を作成する．Wrapper は，作成コストの他，

対象サーバに不足するデータ項目を付け足すなどの操作を伴うことがあり，運用コストの点で実際的でない．Peter Wegner の分類[1]における標準化方式でサーバ間の相互接続性を実現すれば，この問題は解消できる．本研究でもそのアプローチを取る．

ユーザ(外部組織)の認証と同時に，組織 A,B,C にはデータ提供するが，組織 D,E にはデータ提供しないといったアクセスコントロールを想定する必要がある．これは次に述べるようにデータの配置に制限を与える．アクセスコントロールを実現するには，それぞれのサーバで自組織のデータを管理し，ここでユーザ認証(組織認証)を行い，オンデマンドにデータ提供を行う形態を取れば，現実的なコストで運用できる．これは良く知られており，様々なサービスで実際に使われている．センサ運用者から離れたところにデータを集めて管理する広域センサネットワーク研究がいくつかある中で，本研究では，アクセスコントロールを実現する要求を想定し，センサ運用者からデータを離さずに管理する状況を設定する．

本研究では，センサをプロファイル部とデータ部で表現する．センサには固有の ID を割り当て，プロファイルおよびデータは，この ID で結び付けられる．プロファイル部には，センサに関する各種属性 *profile(n)*(e.g., *n* は，設置場所の緯度経度，住所，管理者名，センサ機種，設置環境，移動体名，関連河川名など)が記載され，ユーザアプリケーションは，この属性を参照することで，センサの分類や各種データ加工に利用する．データ部は，センサが通知してきたデータ集合 *data(t)*であり，最新値データから過去のアーカイブまで，理論的には，すべてが時刻 *t* を指定することで読出し可能であるとする．

センサ検索においては，一般に利用形態にあわせて，複数の属性が独立に検索キーとなる．地表面モデル(経緯度モデル)上での検索，特定の会社に属するセンサの検索(組織名検索)の他，移動体名や河川名を検索キーとするような状況が考えられる．本研究では，このような機能を実現する検索を多属性検索と呼ぶ．

本研究ではアクセスコントロールに関する要求や多属性検索を実現するため，センサデータを運用組織のローカルサーバで管理させた上で，SiteTable と IndexTable を用いた分散環境上の検索手法を提案する．検索キーの属性は，文字列型の場合もあれば，浮動小数点数型である場合もある．緯度，経度，河川などの属性群(スキーマ)は，システムを使うアプリケーションに応じて，唯一定義され，全体で共有されていなければならない．本研究で提案するアーキテクチャでは，これをシステムの最高権威が管理し，定期的に更新可能である．

本研究では上記システムの提案を主としているが，国際化に対応させるための多言語対応，アカウント発行によるセンサ管理方式，耐故障性を実現するための基本的な方針と工夫についても，どのようにシステムに埋め込むかを述べる．

本論文の中では，システムの実装状況や，現在の運用状況についても言及する．具体的には，ネットワーク環境や検索のパフォーマンスついて述べる．評価試験としては，システム構築の柔軟性，アカウント管理の有効性，新規アプリケーションへの対応，検索の機

能性，アクセスコントロールの観点から検証する．

以下，本論文の構成を示す．第 2 章で関連研究を述べ，第 3 章にて Live E!システムデザインを述べる．第 4 章で，実装と運用状況に言及し，第 5 章でシステムの評価する．第 6 章で考察し，第 7 章で今後の課題，第 8 章にて結論を述べる．

# 第 2 章 関連研究

## 2.1. Broker による独立データベースの統合

MetBroker[2,3]は，複数の組織で独立に構築されたセンサデータベースを統合し，統一的なインタフェースを，ユーザアプリケーションに提供する．各組織のデータベースと，Broker との間にそれぞれ Wrapper をプログラムすることで,アクセスインタフェースやデータ表現を統一する．具体的には，AMeDAS，和歌山県の砂防情報，米国の Oregon Integrated Plant Protection Center など，約 20 拠点のデータベースを統合している[3]．この方式のシステムでは Wrapper の作成および管理は人為的に行われる．Wrapper によって，不足している情報を追加する場合も考えられ，この場合は対応表のメンテナンスのために作業が発生する．

一般に，異なるスキーマや，異なるアクセスプロトコルで構築されたデータベース間を連携させる方式では，全体のスキーマやプロトコルなどを把握する必要があり，スケール性の点で課題が生じる．本研究での提案システムのように，サーバのデータスキーマ，アクセスプロトコルを標準化してしまえば，このような点でのスケール性は問題にならない．

Broker システムでは，設計開発を終え，運用段階まで進んでいるシステムを取り込むことができるメリットがある．対象とするシステムに変更を加える必要がなく，通常通り運用を続けながら取り込むことが可能である．

ユーザのアクセスコントロールに関して，MetBroker では，オリジンサーバによる認証を，ユーザに透過的に見せることで，実現している．ユーザはそれぞれのサーバからのデータ読出し権を設定してもらうことで，データの読出しが可能になる．この点は，本研究が提案するシステムと同じである．

## 2.2. GSN プロジェクト

Global Sensor Network(GSN)プロジェクト[4]では，仮想化と呼ばれる手法でセンサを標準形で表現し，複数のサーバ間での情報交換にこの表現を用いることで相互接続性を実現している．GSN プロジェクトでは，無線センサネットワークで収集されたデータをインターネット上の P2P オーバレイシステムで取り扱い，広域へセンサ情報を流通させることを可能にするものである．データのアーカイブや，ストリームによる配信／加工が可能だとされているが，検索等の方法も含め，具体的にどのように動作するかは明確になっていないものと思われる．

センサは Zeroconf 方式でシステムに取り込まれ，簡単に設置・利用ができるとされているが，誤登録の結果として残るゴミセンサや，アーカイブに残るゴミデータの扱いについては不明である．またユーザの認証やアクセスコントロールに関してもあまり研究されて

いないようである．

本研究で提案するシステムでは，データストリーム配信については扱ってはいないものの，データアーカイブ，検索，センサアカウント発行によりシステムへ登録する方式など，この辺りの課題について具体的な提案をしている．

## 2.3. 地理位置検索システム

明確には述べられていないが，Mill[5]や IrisNet[6]も，センサデータがシステム内で標準化された状況を想定している．

Mill は 2 次元空間を Z-ordering によって 1 次元 ID 空間に対応させた上で，Chord リング[7]でこれらの ID 空間を管理することで地理位置検索に関してスケール性を持たせている．ただ，本研究が想定する組織を単位としたアクセスコントロールや位置以外を検索キーにする場合については想定されていない．耐故障性やネットワーク遅延による影響などについても述べられていないため，実際的な環境で述べられているほどの性能が出るかはわからない．

IrisNet は，Organizing Agent(OA)により地理的な住所構成の木を作成し，その上にセンサデータを対応づける．XPath による検索を実現し，位置以外のものも検索キーに含むことが可能だが，Sensing Agent(SA)からのデータを複数の OA に通知する状況が想定されており，アクセスコントロールをどのように設定可能なのかが明確になっていない．本研究では，センサ運用組織にデータ公開権限を持たせ，センサ運用組織のポリシーでアクセスコントロール設定を行える状況を想定している．

## 2.4. Query-initiative 型／Data-initiative 型による分類

Hyo-Sang Lim et al[8]によると，本研究で対象としているシステムは，Query-initiative 型に分類される．予め分散的に蓄えられているデータに対して，クエリを発行し，データを取得する方式である．一方，予めクエリを発行しておき，発生するセンサデータを能動的にイベント情報として購読者に配信する Data-initiative 型のシステムもある．GSN[6]や，[9][10]は，このような型のシステムを研究している．

# 第 3 章 Live E! システムデザイン

Live E!プロジェクト[11]では, 本提案システムをデザインするまでの, 2005 年 5 月～2007 年 5 月の 2 年間に渡り，実際的な広域センサネットワークへの要求条件を調査した．調査結果のうち，本研究で取り組む要求条件を下記に挙げる．

- 標準形式による相互接続
- 組織単位のセンサプロファイル管理
- データ提供先のアクセスコントロール
- 分散環境におけるセンサ検索
- 複数の検索キーの同時サポート
- プロファイルスキーマの柔軟な更新

プロファイルスキーマは，プロファイルの属性名や属性型などの定義のことで，プロファイル記法および広域検索(分散環境におけるセンサ検索)のクエリ記法，検索処理を規定する．スキーマは，配布ソフトウェアに暗黙的に書込まれるべきではなく，システムが想定するアプリケーションの更新によって，定義を動的に変更できる必要がある．新規にプロファイル属性を追加する場合がこのような状況に相当する．本研究が対象とするシステムにおいては，運用経験上，スキーマの動的更新が 1 日以内に達成できれば十分である．

## 3.1 アーキテクチャ

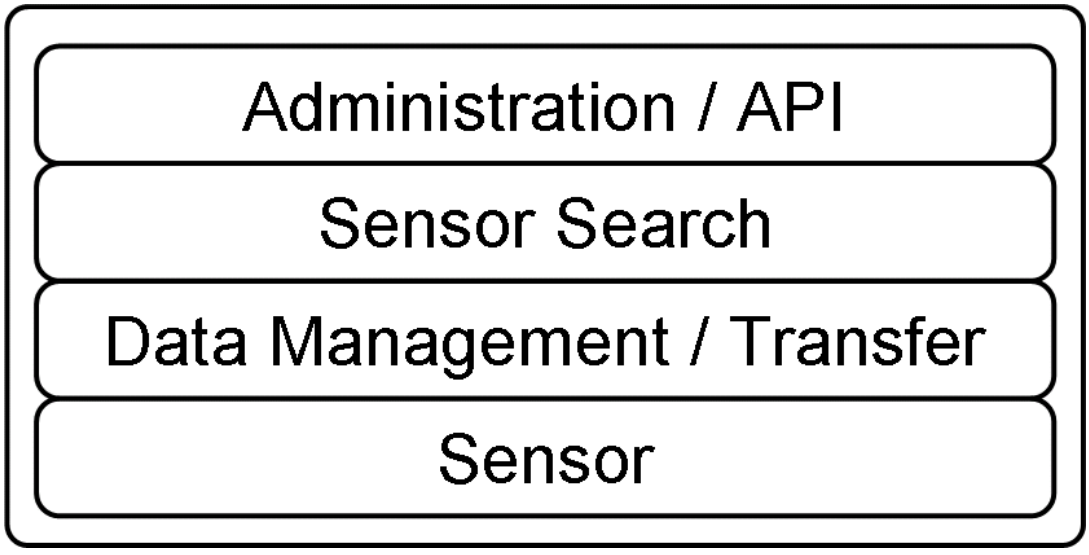



図 1: Live E! 広域センサネットワーク・層アーキテクチャ

Live E!広域センサネットワークとして，図 1 に示す層アーキテクチャを提案する．システム全体としてこの層構造を持つが，サーバ単独でもこの層構造を持っている．サーバ間はインターネット上のオーバレイとして結合され，ネットワークを形成する．結合されたリンクの上で，コントロール情報が交換され，全体として一つのシステムとして協調動作する．

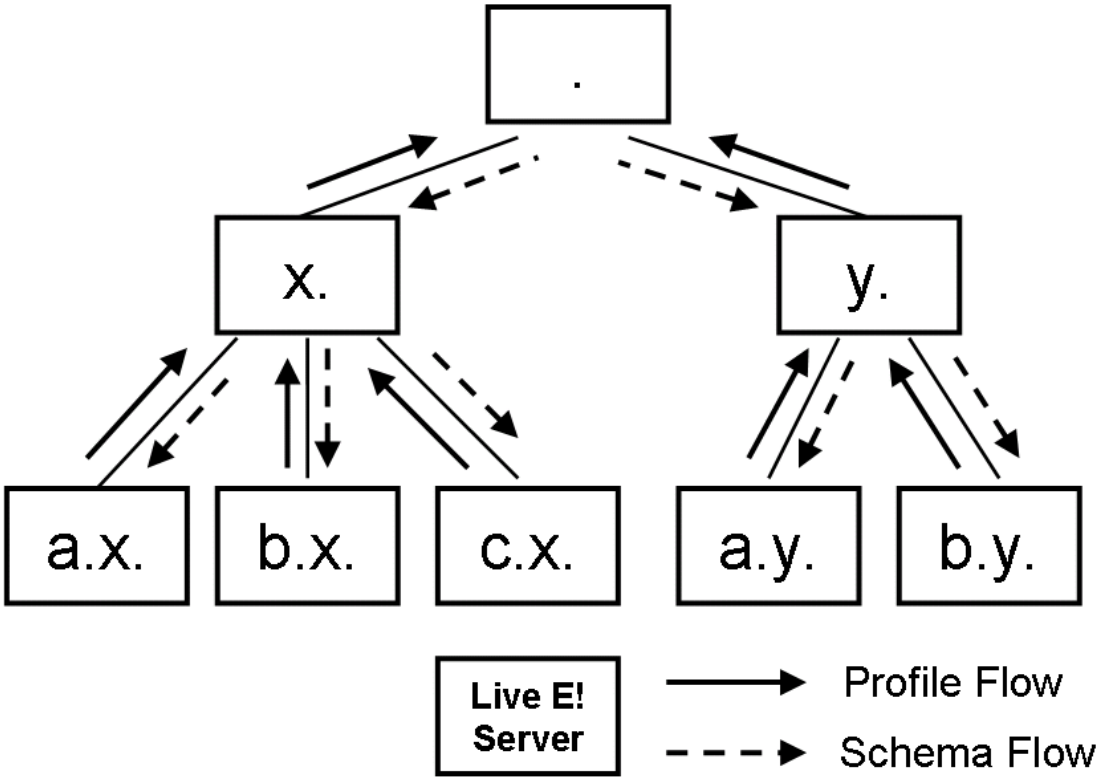



**図 2: 階層構造による Live E!運用ネットワーク**

ネットワーク・トポロジは，図 2 に示すような階層構造になるように，管理層(Administration 層)で静的に設定するものとする．静的なのは運用時に計画的なトポロジ設計を行えるようにするためである．サーバには DNS と同じ Variable-Depth 木[12]で名前を与える．トポロジに階層構造を採用した理由には，多くの組織の構造が階層的であること，プロファイルスキーマの配信が容易なこと，ルーティングプロトコルが簡素になることがある. サーバは DNS と同じ Master/Slave 方式で冗長構成をとることが可能である．

## 3.2. 管理／API 層

### 3.2.1. グローバル管理

管理層では，システム全体でプロファイルスキーマを一つ管理し，ルートサーバから配

信する．実際には，各サーバが上位階層サーバのスキーマを定期的に読み込むことで，ルートサーバによって発行されたスキーマを，システム全体に行渡らせることができる．図 3 にプロファイルスキーマの例を示す．属性名，属性型，属性の取りうる値，多言語を伴うか否かなどの情報が記述されている．ここで規定されるスキーマは，センサネットワークが対象とするアプリケーションを宣言するものである．スキーマに基づいてプロファイルやクエリが作成され，アプリケーションをサポートする．

本提案システムでは，スキーマをルートサーバで管理し，これをシステム全体に配信させることによって，システムが対象とするアプリケーションを柔軟に更新することを可能にしている．

```
<schema>
 <attr name=”location” type=”string” value=”.*” multilanguage=”true” />
 <attr name=”latitude” type=”float” />
 <attr name=”longitude” type=”float” />
 <attr name=”altitude” type=”float” />
 <attr name=”sensorVendor” type=”string” value=”Vaisala|Oregon” />
 <attr name=”sensorModel” type=”string” value=”WXT510|WM918|WMR968” />
 <attr name=”vehicle” type=”string” value=”.*” />
 <attr name=”river” type=”string” value=”.*” multilanguage=”true” />
    ・・・
</schema>
```

**図 3: プロファイルスキーマの例**

### 3.2.2. ローカル管理

各センサ運用組織の管理層では，下記の設定を行う．

- 近隣ノードとのリンク設定
- センサアカウント発行
- センサプロファイル発行
- データ提供先アクセスコントロール設定
- 各種パラメータ(キャッシュ時間など)の設定
- 基本設定(ログ出力先など)

近隣ノードとのリンク設定を静的に行い，管理可能なネットワークトポロジを構成していく．リンク設定は各サーバで行われるが，この設定されたリンクを通じてグローバルなネットワークが構築される．

センサエンティティをシステム内に作成するには，管理層で運用者によりアカウントを作成する．アカウントは，センサデータのアーカイブやプロファイルと結合される．その

ため，アカウントはアーカイブおよびプロファイルを削除するまでの永続的に有効なものである．作成されたアカウントをベースにセンサ認証を行う方式を採用することで，Zeroconf でセンサを取り込む方式と比べ，誤って登録されてしまうセンサを減らすことができる．

センサが各組織で設置され運用されることを考えると，そのプロファイルは組織ごとに発行されることが妥当である．ここでプロファイルのいくつかの属性は，自動生成することも可能だが，全属性の設定の自動化は難しいため，人の手で設定できる必要がある．GPS により経緯度情報は自動化できるし，センサ機器自身が機種情報を持つことでの自動化もできる．しかし，どのような設置環境(e.g.，路面，公園)かや，移動体名などの情報は人手による設定が実際的である．

アクセスコントロール設定は，組織のセンサ公開ポリシーを設定するものである．本提案システムでは，検索要求およびデータ取得要求は各サーバが発行する．要求の発行元組織名(サーバ名)を許可リストや拒否リストに追加することで設定する．<*.x.>を許可リストに加えれば，<a.x.>や<b.x.>などからのデータ取得要求は許可される．ここで，括弧<>でサーバ名を表す．アクセスコントロール設定は，データ転送層の Reference Monitor[13]に対して行われ，センサ検索後のオンデマンドなデータ取得要求に対して働く(3.4 で詳しく述べる)．

### 3.2.3. 言語ロケール

プロファイルの特定の属性(スキーマで multilanguage="true"が指定された属性)は，各言語での表記を掲載することができる．例えば，英語表記，日本語表記，タイ語表記など種々言語表記を，設置拠点名(location)に関連付けることが可能である．ユーザには，指定したロケールでの表記を生成的に取り出し提供する(図 4)．

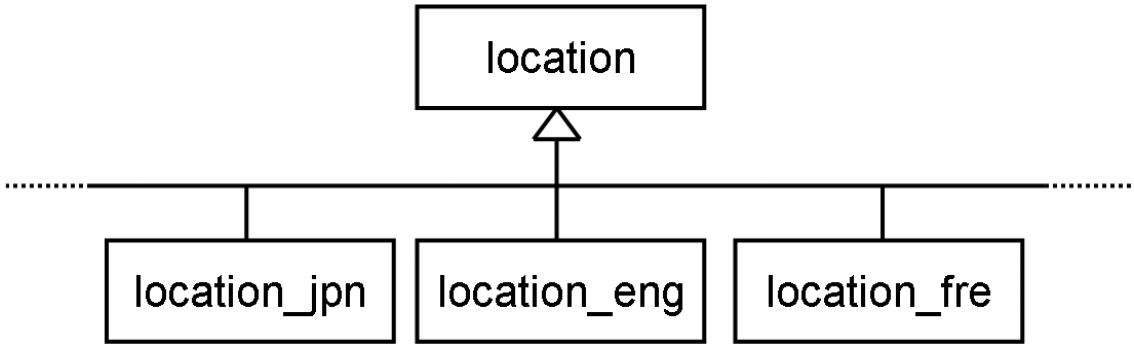



**図 4: 多言語の管理モデル**

言語ロケールが設定されているユーザには，location として提示されるが，プロファイルには location_jpn，location_eng，location_fre それぞれに，日本語，英語，フランス語で

の表記が記載されている．

### 3.2.4. Application Programming Interface (API)

API 層は，ユーザアプリケーションがセンサ検索／データ読出しを行うためのインタフェースを提供する．具体的にはセンサ検索を行うメソッドとして，

String search(String query)

がある．クエリの表記は 3.3 項で示すものである．このメソッドは，クエリに合致するセンサを含むサーバのリストを返すものであり，センサデータの読出しまでは行わない．センサデータの読出しを行うメソッドがこの上に定義され，ユーザアプリケーションへ提示される．

## 3.3. センサ検索層

センサ検索層では，管理層で設定された Live E!ネットワーク上での，センサ広域検索を自律分散的に実現する．2 種類の検索スキームを備え，組織名で検索する場合と，その他の一般属性で検索する場合との双方に対応させる．まず，一般的にクエリのルーティング手法を述べ，その後，組織名検索および多属性検索の実現手法について述べる．後半では，両者を複合的に利用した検索，Iterative 方式の採用，サーバリストのキャッシュについても解説する．

### 3.3.1. 検索クエリのルーティング

分散環境で検索を行うために，本研究では，検索クエリをルーティングする手法を採用する．ここでは，この方式を一般的に数式で表現する． $A_n$ を $n$ ホップ後にクエリが到達するサーバノードの集合，$s_0$ を検索開始サーバノードとする．このとき， $A_{n-1}$ から $A_n$ を算出する計算は漸化的に以下のように与えられる．

$$A_n = R(A_{n-1})Q, \quad A_0 = \{s_0\} \text{・・・数式 1}$$

ここで， $R$ は経路表の全体を表す． $R(s)$ をサーバノード $s$ での経路表とし， $R(A_{n-1})$ は $R(s), \forall s \in A_{n-1}$ を意味する． $Q$ はユーザから提示されたクエリである．

### 3.3.2. 組織名検索

ネットワークを構成する各サーバノードには，Variable-Depth 木[12]により名前の割り当てが行われている．これは DNS と同様の名前割り当て手法であり，検索対象がユニークであれば，O(d)での検索が可能である(d は，階層の深さ)．以下，数式 1 を念頭に，組織名検索の定義を述べる．具体的な状況として，サーバノード $s$ に，子ノード<jp.> <tw.> <th.>があり，検索クエリ $Q$ が<x.y.z.tw.>などと与えられる場合を考える．組織名検索は子ノードの文字列が， $Q$ の後半の文字列に合致すれば，その子ノードへクエリが転送されると定義

する．この状況下では<tw.>のみがクエリ<x.y.z.tw.>の後半に合致するため，$Q$は<tw.>へ転送される．この合致性を，<x.y.z.tw.> ⊂ <tw.>と表現し，<tw.>・<x.y.z.tw.>=1 と定義する．一般に，

$$u \subset v \quad \Rightarrow \quad u \bullet v = 1$$
$$u \not\subset v \quad \Rightarrow \quad u \bullet v = 0$$

と定義する．転送のために必要な下位通信レイヤでのアドレス識別子等の情報を$a_{<jp.>}$などと表現すれば，クエリ転送先$R \bullet Q$は，

$$R \bullet Q = \left(a_{<jp.>} < jp. > + a_{<tw.>} < tw. > + a_{<th.>} < th. >\right) \bullet Q$$

と表現できる．$Q =< hoge.jp >$，$Q =< ilan.th. >$，$Q =< psu.th. >$の場合それぞれについて計算してみると，

$$\left(a_{<jp.>} < jp. > + a_{<tw.>} < tw. > + a_{<th.>} < th. >\right) \bullet \begin{pmatrix} < hoge.jp. > \\ < ilan.tw. > \\ < psu.th. > \end{pmatrix} = \begin{pmatrix} a_{<jp.>} \\ a_{<tw.>} \\ a_{<th.>} \end{pmatrix}$$

となり，それぞれ$a_{<jp.>}$，$a_{<tw.>}$，$a_{<th.>}$へ転送されることになる．

本提案システムでの組織名検索クエリを，図 5 のように表記する．

```
<query domain=”admin” name=”*” />
```

**図 5: 組織名検索クエリの例**

domain は，アプリケーションの領域を示すもので，本研究では常に admin(運用管理ドメイン)である．name には，検索対象とする組織名(サーバ名)を記述する．<*>であれば，すべての組織を対象とし(i.e.，$\forall s \in$ サーバノード集合 $\Rightarrow$ $s \subset < * >$)，<*.wide.jp.>であれば，<wide.jp.>以下のすべての組織，<hongo.wide.jp.>であれば，<hongo.wide.jp.>を対象とする．

### 3.3.3. 多属性検索

多属性検索は，プロファイル情報を基にして様々な属性による検索を実現するものである．ここでは，数式 1 の原理に基づき，多属性検索を実現するための，R の構築および右辺の演算処理について述べる．R の構築は，クエリ経路表の構築で，右辺の演算処理はクエリの転送処理である．本研究では，クエリ経路表の構築を自律分散的に実現するために，プロファイルをネットワーク上で定期的に交換し合う．以下，管理層からスキーマが与えられたときに，どのようにして，異なるキーでの検索を並列に実現するかを示す．

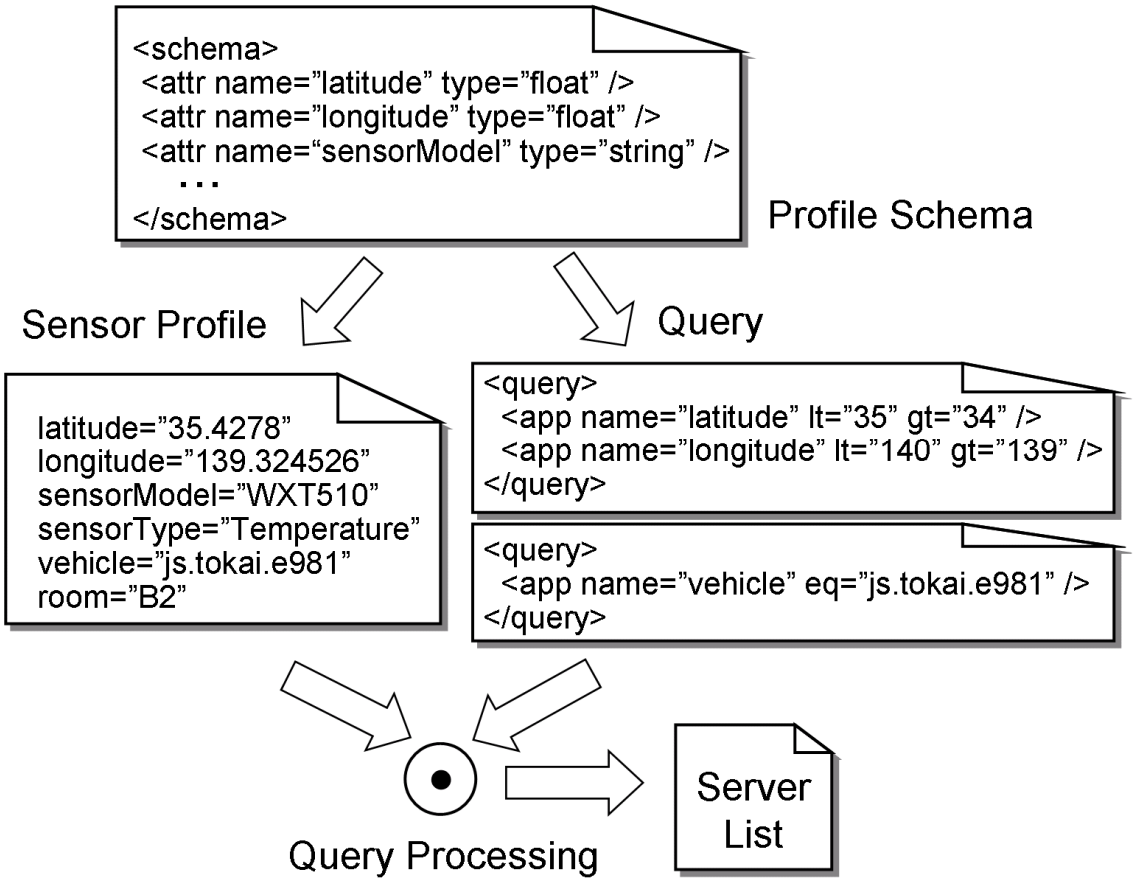



**図 6: スキーマ，プロファイル，クエリ，検索プロセスの関係**

図 6 にプロファイルスキーマ，プロファイル，検索クエリ，検索プロセスの関係を示す．スキーマが管理層から与えられ，各センサ運用組織は，センサに対して，このスキーマに適合するプロファイルを発行し，ローカルに登録してある．検索クエリもスキーマに適合する形で作成され，検索プロセスに利用される．ここでの検索プロセスとは，クエリに合致するプロファイルを含むサーバをリストアップする演算を意味する．ここでクエリの表記を図 7 に示す．

```
<query domain="admin" name="*" >
 <app name="latitude" lteq="40" gteq="30" />
 <app name="longitude" lteq="140" gteq="130" />
</query>
```

**図 7: 多属性検索，クエリ表記**

app 属性は，多属性検索で用いる属性および条件を示す．図 7 の例では，latitude と

longitude が設定されている．複数設定されている場合は，それぞれの条件を満たすセンサを持つサーバを意味する(すなわち積集合)．条件の指定としては，より小さい(<; lt)，より大きい(>; gt), 等しいかより小さい(=<; lteq), 等しいかより大きい(=>; gteq), 等しい(==; eq), 等しくない(!=; neq)，含む(contains)などの演算を提供する．

本研究において，R はクエリ経路表と呼んでいるが，分散インデックス[14]と呼ばれることもある．ただし，分散インデックスという言葉には，クエリ転送処理を高速に実現するための仕組みが搭載されたクエリ経路表という意味が込められる．

以下，検索処理の実装を，クエリ経路表 R の構築，およびクエリ転送処理(i.e.，右辺の演算)に切り分けて考える．本研究では検索を，漸化式に抽象化することに主眼を置いているが，以下，提案システムで採用したクエリ経路表の構築手法，およびクエリ転送処理について述べる(ここではサーバのことを単にノードと表現する)．

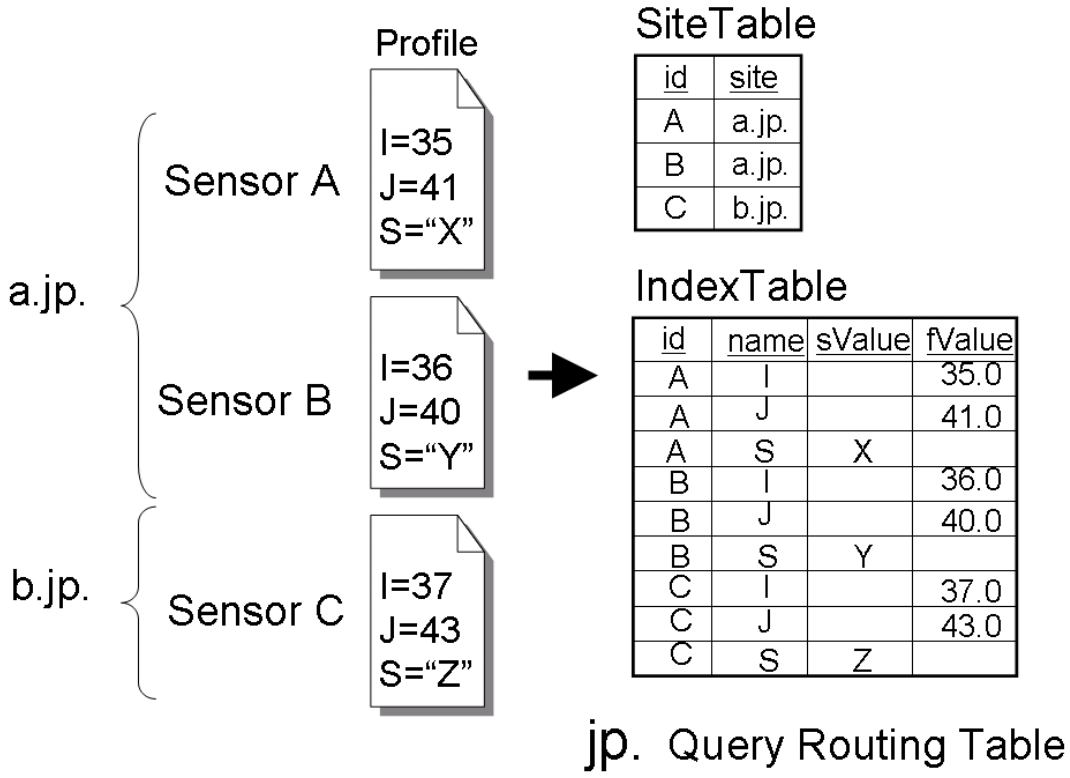



**図 8: クエリ経路表の構築**

**クエリ経路表の作成** 図 8 にノード<jp.>でのクエリ経路表の作成手順を示す．<jp.>には，子ノード<a.jp.>と<b.jp.>があり，<a.jp.>以下のノードには，センサ A, B，<b.jp.>以下のノードにはセンサ C が存在している．それぞれのセンサプロファイルは，図 8 に示す通りとする．ここでプロファイルの属性は，I および J(e.g.，緯度，経度)は浮動小数点数型，S が文字列型(e.g.，センサ機種)とするようにプロファイルスキーマで規定されている．<jp.>は，<a.jp.>および<b.jp.>から，センサプロファイルを受け取り，SiteTable と IndexTable

を次のように作成する．SiteTable では，センサ ID と子ノードを対応させ，IndexTable には，それぞれのセンサプロファイルを(センサ ID，属性名，文字列型の値，浮動小数点数型の値)で展開する．図の例では，I および J は浮動小数点数なので，fValue に値が格納され，S は文字列型なので，sValue に値が格納されている．次に述べるフォワーディング処理のため，(name, sValue)および(name,fValue)，それぞれに関して，検索インデックスを作成してある．

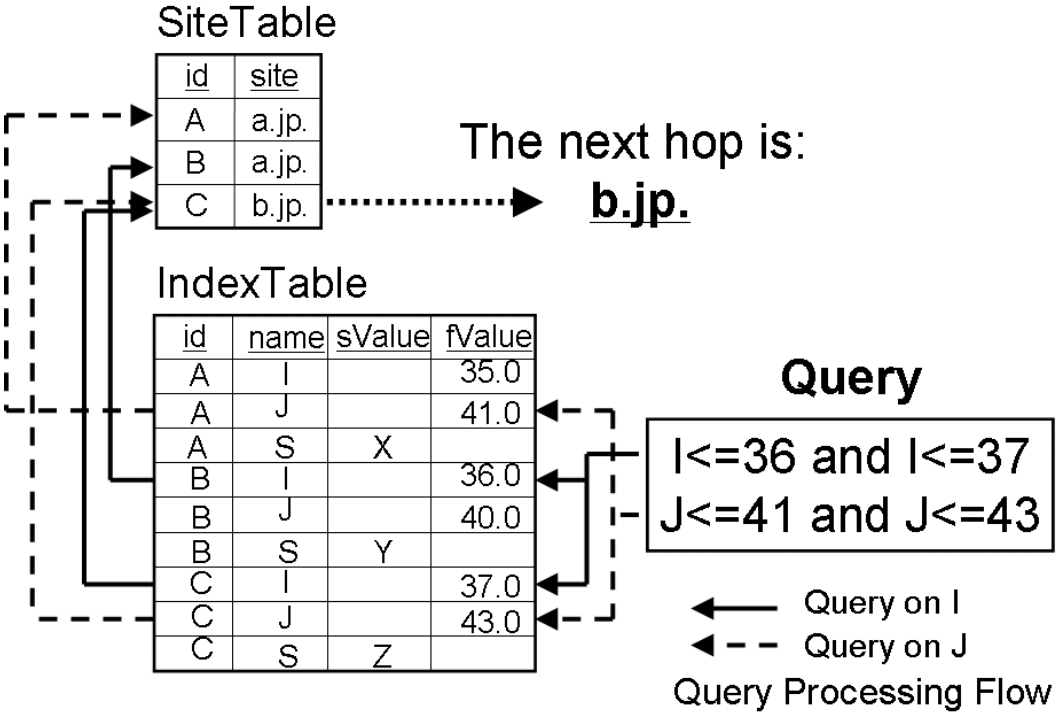



**図 9: クエリの転送処理**

**クエリ転送処理** いま<jp.>に対して，図 9 に示すような検索クエリが発行されたとする．まず検索クエリを，属性ごとの条件文に分割し，IndexTable において，それぞれ条件に適合するセンサ ID を考える(図 9 では，条件 I の場合 B と C，条件 J の場合 A と C)．SiteTable において，それらセンサ ID の積集合を取り，最終的に残ったセンサ ID を持つ子ノードをクエリの転送先として算出する(図 9 では，C が積集合として残り，<b.jp.>が解)．

この方式により，プロファイル属性を検索キーにしたノード検索が可能になる．検索対象領域が狭ければ，クエリ転送処理は多くの場合 $O(\log N)$以下で達成できる($N$はセンサ数，検索対象領域が狭い状況下ではインデックス検索コストが支配的)．この方式自体では，ノードの経路表は $O(N)$で増加する．

### 3.3.4. 複合検索

本研究で提案する多属性検索においては，センサ数の増大に伴い，上位ノードの経路表が増大し，同時に検索対象領域が広いクエリ転送処理には負荷がかかる．多属性検索においてスケール性を実現する研究は将来的な課題とし，本研究の段階では，検索は多属性検索の機能性を犠牲にすることでこの問題に対処することを考えている．ルートサーバから途中の階層まで組織名(ノード名)をキーとした検索を行なわせ，その先から，他のキーでの検索を行わせる方式である(図 10)．クエリ表記で記述すれば，図 11 のようになる．一定規模以上になった場合，上位ノードでの SiteTable, IndexTable の作成を諦めるか，検索対象領域が広いクエリの転送処理を禁止する．3.3.2 で述べた組織名をキーにした検索では，クエリ転送処理時間は事実上 *O(1)*になるため(理由: SiteTable, IndexTable での処理が不要なため)，クエリ転送処理に関しては，スケール性を持たせる事が可能である．

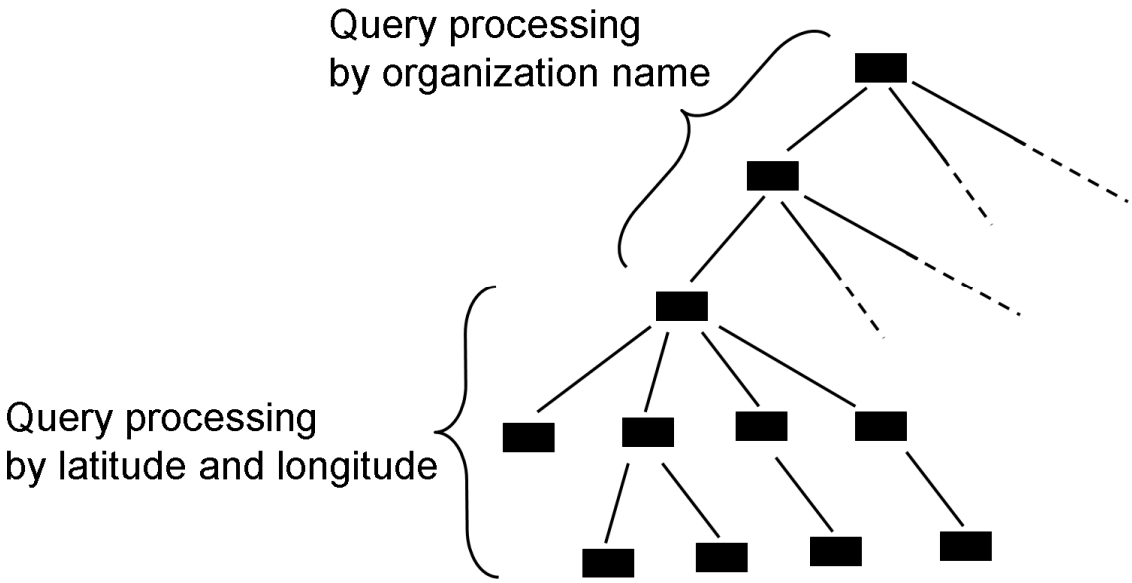



図 10: 複合検索

```
<query domain=”admin” name=”*.wide.jp.” >
 <app name=”latitude” lteq=”40” gteq=”30” />
 <app name=”longitude” lteq=”140” gteq=”130” />
</query>
```

図 11: 複合検索クエリの例

### 3.3.5. Iterative v.s. Recursive

検索の実装[15]には，Iterative 方式を採用している．Iterative 方式は，Recursive 方式

と比較しセッション時間が短いので，検索の上位ノードの負荷を小さく抑えることが可能であると同時に，タイムアウト閾値を低く抑えることが可能だと予測されるためである．

### 3.3.6. サーバリストのキャッシュ

本提案システムでは，ユーザにデータが提供されるまでに 2 段階のキャッシュが存在する．サーバリストのキャッシュと，データのキャッシュである．データのキャッシュは 3.4 項で述べる．検索要求に対して作成されたサーバリストは，センサデータに比べて更新される頻度が低い．センサデータは 1 分の間に複数の値が更新されるのに対し，サーバリストの生成に関わるプロファイルは，1 時間に 0.05%程度の確率で更新される(旧 Live E! システムのログ解析結果による)．したがって，キャッシュ時間を 1 時間ほどにしても大きな問題は起こらない．一方，クエリの種類を分類して結果をキャッシュするか否かを決定するようにし，発行頻度が高いクエリの結果のみをキャッシュできるようにしている．

## 3.4. データ管理／転送層

### 3.4.1. データ管理層

データ管理層は，各サーバのローカル環境において，センサ層から通知された認証後のセンサデータを扱う．データ管理や処理に関しては，ユーザからの要求条件に応じて，決定されるものであるが，本研究では，次のような要求を想定して，システムのデザインを行っている．

- ユーザはセンサの最新値に特に関心があり，頻繁に最新値のデータを読出す
- ユーザはアグリゲーション値(平均値，最大値，最小値など)に関心がある
- ユーザは場合によって過去の生データを必要とする

データ管理層に，最新値ファイル，アグリゲーション処理ファイル，過去データアーカイブファイルを設けて，それぞれにデータを格納することで，これらの管理をしている．

### 3.4.2. データ転送層

データ転送は，データ管理層で扱われているセンサデータのコピーを，外部サーバが読み込む操作である．ユーザからの要求を受け付けたサーバは，センサ検索層でリストアップされたサーバそれぞれに対して，データ取得要求を発行する．各サーバからはデータ取得のためのサービスが提供されており，このサービスを利用することで，データを取得することができる．図 12 に示すように，データ提供サービスのすぐ裏に，Reference Monitor [13]を設置し，提供先サーバの認証を行うようにして，データ提供先のアクセスコントロールを行う．Reference Monitor は，管理層での設定に従って，要求発行元のサーバにデータを提供すべきかどうか判断する．要求発行元のサーバ識別はデータ取得要求発行者が，自サーバ名を同時に提示させることで行う．本研究の対象外だが，要求身元の確認(ベリファイ)や改竄を防ぐ機構などはここに組み込まれる．

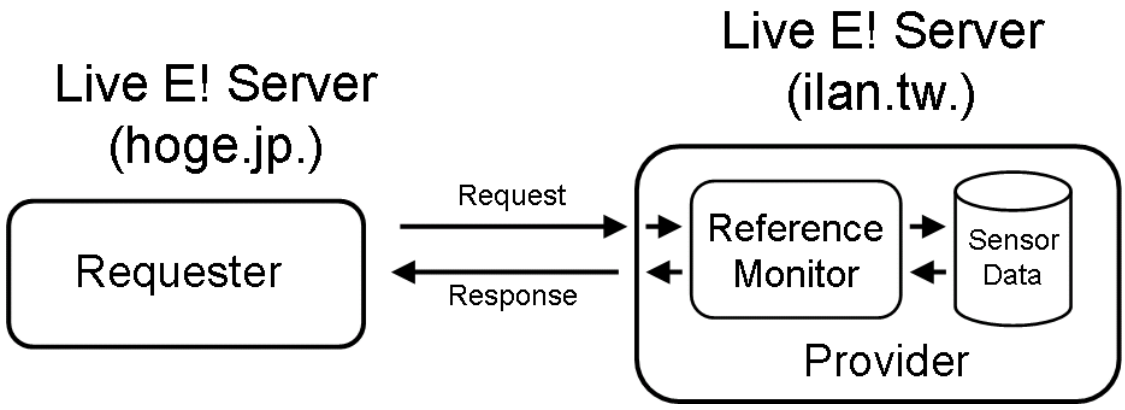



図 12: Reference Monitor によるデータ提供先のアクセスコントロール

複数のサーバから取得されたデータはマージされ，これが検索結果としてユーザに提供される．この取得されたデータはキャッシュされ，同一検索要求に対して高速に応答できるようになっている．

## 3.5. センサ層

センサ層は，インターネットを介してセンサとの通信を担当する．データ通知インタフェースは，システム内で標準化されており，そのインタフェースを使ってセンサがデータを通知してくる．通信時には，センサの認証も伴う．管理層で発行されたセンサアカウントに対応するセンサのデータのみシステム内に取り込み，それ以外のセンサに対してはエラーを応答する．センサは決められたサーバにのみデータを通知することができる．

## 3.6. サーバノードの複製と耐故障性

サーバノードは，メンテナンスやその他の原因で停止することが考えられる．特定のサーバノードの停止が，別システムを停止させたり，パフォーマンスを極端に停止させたりすることは極力避けるべきである．本研究で提案するシステムでは，それぞれ組織を代表するサーバを単位に，Master/Slave 方式で複製を作成し，冗長化効果によりサービス率の向上を計る仕組みにしている(図 13).

システムの停止の他，場合によってはディスク等が破損してしまい，データの復旧が不可能になってしまう場合も考えられる．そこで，複製するものとしては，データ提供／サーバ検索等のサービスの他，組織で管理するデータそのものも含まれる．

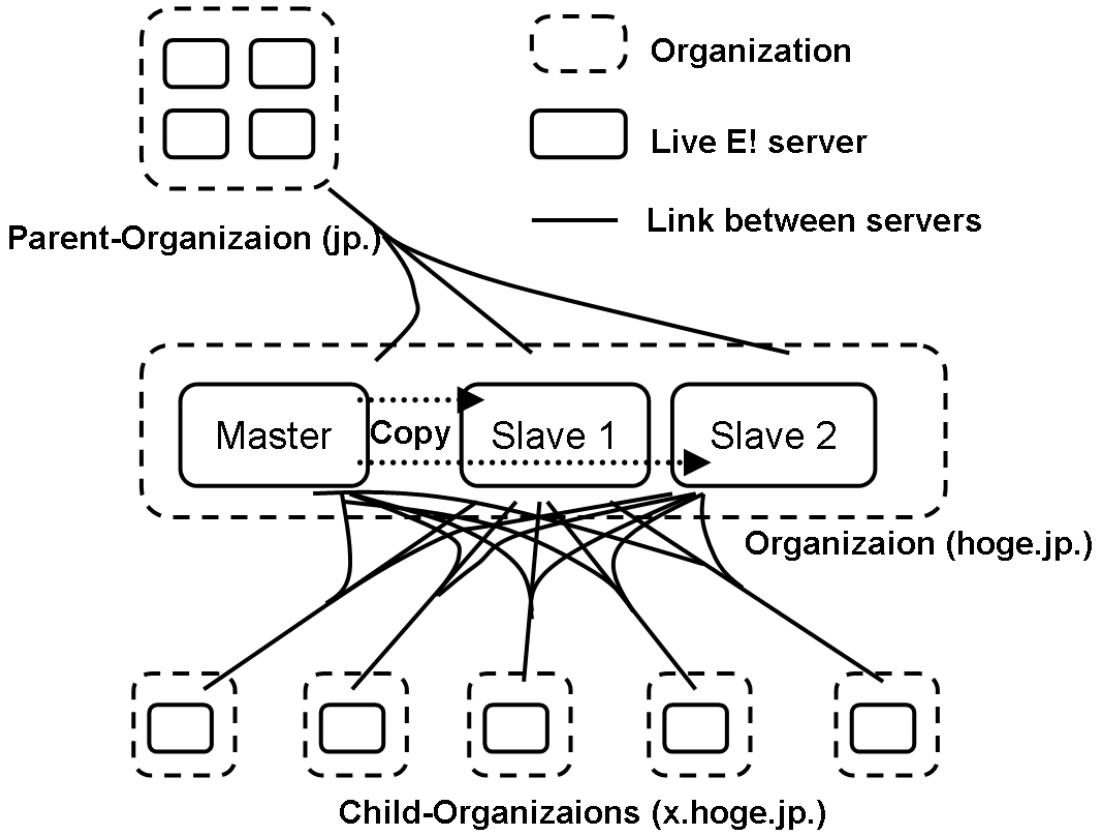



図 13: 冗長化による耐故障性の向上
Master, Slave 1, Slave 2 いずれか一台が動いていれば,
ネットワークとして動作が継続できる

クエリ転送処理において，転送先のサーバ群のうち，問合せ先のサーバが停止していた場合，次のサーバ候補へ問合せを繰り返すが，この停止の判断に数秒程度の検索遅延を生み出すことになる．そこで，本提案システムでは，各サーバノードが周辺のサーバの動作状態を定期的にモニタリングし，リダイレクト先として提示するサーバリストにその動作状態を付加させる．このようにすることで，検索クエリが停止しているサーバを自動的に避けるように検索プロセスが達成される．

# 第 4 章 Live E! 実装／運用状況

## 4.1. 実装

### 4.1.1. サーバの実装アーキテクチャ

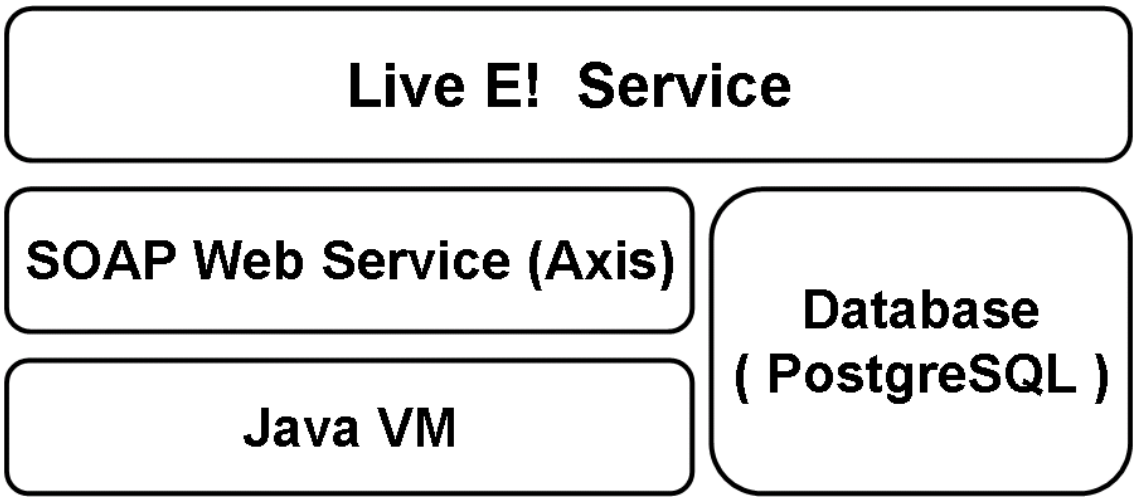



図 14: Live E! サーバ実装アーキテクチャ

図 14 に典型的な Live E!サーバの実装形態を示す. Live E!のサービス(ver.0.9.9)は, Java2 ver.1.5 で実装され，その規模はクラス数にして 291 個である．インターネットを介したすべての通信(i.e.，センサ・サーバ間，ユーザ・サーバ間，サーバ・サーバ間)は，SOAP Web サービスで行われ，交換されるデータは XML 上に表現されている(具体的なサービスは付録 A に掲載する)．SOAP エンジンの実装には，Axis1.4[16]を使用している．データベースに PostgreSQL を使用し，センサデータのアーカイブを管理する他，検索クエリの経路表もデータベースのテーブルとして実装されている．

### 4.1.2. 検索機能の実装

クエリ経路表のテーブルを作成する SQL コマンド列を図 15 に示す．SiteTable を admin.combined テーブル，IndexTable を admin.combined_index テーブルとして実装した．データ型は文字列，浮動小数点数の他，整数型，DateTime 型にも対応させてある．図 16 に検索クエリの SQL での表現を示す．この SQL コマンドを発行することにより，PostgreSQL がクエリ転送先の計算を実行する．

```
CREATE TABLE admin.combined (
id varchar PRIMARY KEY,
site varchar REFERENCES admin.neighbor(name) ON DELETE CASCADE,
);

CREATE TABLE admin.combined_index (
id varchar REFERENCES admin.combined(id) ON DELETE CASCADE,
name varchar NOT NULL,
sValue varchar,
fValue float8,
iValue integer,
tValue timestamptz,
PRIMARY KEY(id,name)
);

CREATE INDEX combined_index_sValue_index
            ON admin.combined_index(name,sValue);

CREATE INDEX combined_index_fValue_index
            ON admin.combined_index(name,fValue);

CREATE INDEX combined_index_iValue_index
            ON admin.combined_index(name,iValue);

CREATE INDEX combined_index_tValue_index
            ON admin.combined_index(name,tValue);
```

図 15: クエリ経路表のデータベースへの実装

```
SELECT site FROM admin.combined WHERE
  id IN (
    SELECT id FROM admin.combined_index
      WHERE
        name='latitude' AND
        fValue>=30 AND
        fValue<=40
  ) AND
  id IN (
    SELECT id FROM admin.combined_index
      WHERE
        name='longitude' AND
        fValue>=130 AND
        fValue<=140
  )
GROUP BY site;
```

図 16: クエリ転送先を算出する SQL の例

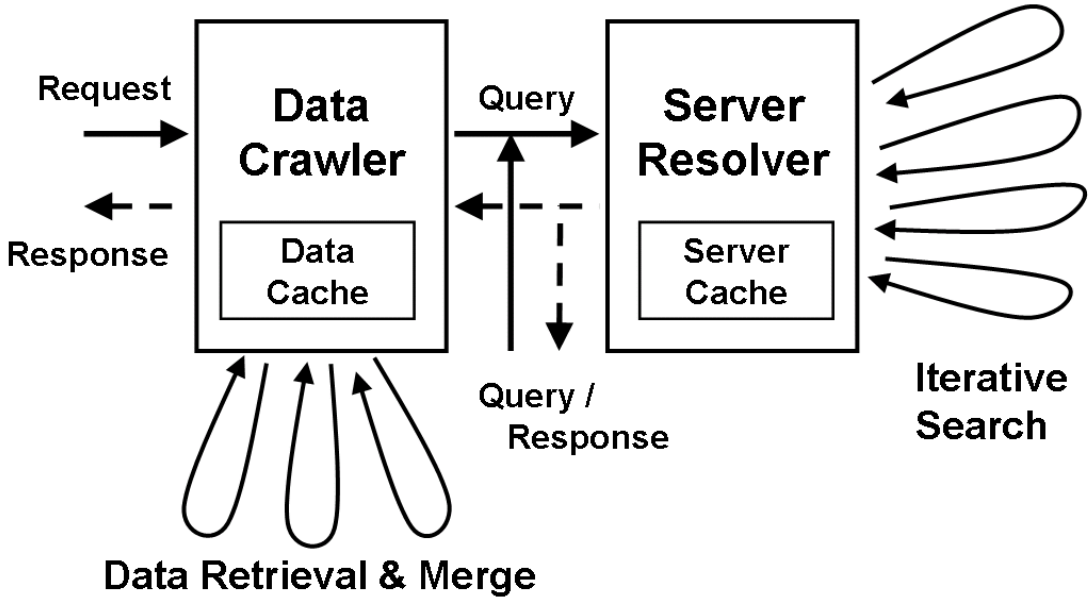



図 17: 検索エンジンの構成

検索エンジン(図 17)は，サーバ検索部(Server Resolver)と，データ抽出部(Data Crawler)とに分けて実装した．ユーザからクエリ要求があったときに，以下のような動作をする．

1. データ抽出部に入る
2. クエリ要求に対する有効なサーバリストがキャッシュにあれば，それを応答し終了，なければ次の処理を続行
3. サーバ検索部に入る
4. クエリ要求に対する有効な解がキャッシュにあればそれを応答(7.へ)，なければ次の処理を続行
5. シングルスレッドで Iterative にクエリを発行し，サーバ検索を行う
6. 検索結果リストをキャッシュし，データ抽出部へ結果を応答する
7. リストに記載されたサーバすべてに対してシングルスレッドでデータを問い合わせる
8. 収集されたデータをマージし，結果とする
9. 結果をキャッシュ後，応答し終了

サーバ検索，およびデータ抽出は，すべてシングルスレッドで行われる．またサーバ検索が完了してからデータ抽出が行われ，サーバ検索中にデータ抽出プロセスを開始する実装にはなっていない．

### 4.1.3. センサの表現

Live E!では，センサを図 18 のように表現する．センサユニットには，種々の観測センサが搭載されている場合が少なくない．センサユニットを表現するボックスを考え，その中

に，各種センサを表現する．各センサが観測したデータを value として，それらのセンサ内に書込むという形態である．ここで value は複数あっても良い．センサ ID はグローバルユニークであることが要求される．センサからのデータアップロードおよび，ユーザへのデータ提供には，すべてこの形式によるセンサ表現を用いている．

センサユニット, および各センサにはプロファイルを XML の属性として記述することができる．図 18 の例では，センサユニットに対し，location_eng，location_jpn，latitude，longitude，sensorVendor，sensorModel 属性でプロファイル情報が記載され，各センサには，sensorType 属性がプロファイル情報として記載されている．

```
<sensorGroup class="combined"
    id="hongo.wide.ad.jp/WXT510/u-tokyo/elab/"
    location_eng="The University of Tokyo"  location_jpn="東京大学"
    latitude="35.712194" longitude="139.76775"
    sensorVendor="Vaisala" sensorModel="WXT510" xmlns="http://live-e.org/DataType/2007/03/">
  <sensor id="hongo.wide.ad.jp/WXT510/u-tokyo/elab/Temperature" sensorType="Temperature">
    <value time="2008-01-24T00:00:00.0000000+09:00">25.5</value>
  </sensor>
  <sensor id="hongo.wide.ad.jp/WXT510/u-tokyo/elab/Humidity" sensorType="Humidity">
    <value time="2008-01-24T00:00:00.0000000+09:00">68.4</value>
  </sensor>
  <sensor id="hongo.wide.ad.jp/WXT510/u-tokyo/elab/Pressure" sensorType="Pressure">
    <value time="2008-01-24T00:00:00.0000000+09:00">1021.9</value>
  </sensor>
  <sensor id="hongo.wide.ad.jp/WXT510/u-tokyo/elab/RainFall" sensorType="RainFall">
    <value time="2008-01-24T00:00:00.0000000+09:00">0.0</value>
  </sensor>
  <sensor id="hongo.wide.ad.jp/WXT510/u-tokyo/elab/WindDir" sensorType="WindDir">
    <value time="2008-01-24T00:00:00.0000000+09:00">325</value>
  </sensor>
  <sensor id="hongo.wide.ad.jp/WXT510/u-tokyo/elab/WindSpeed" sensorType="WindSpeed">
    <value time="2008-01-24T00:00:00.0000000+09:00">0.6</value>
  </sensor>
</sensorGroup>
```

**図 18: センサの表現**

### 4.1.4. インターネットセンサの構成

図 19 に現在の Live E!が使用している典型的なインターネットセンサの構成を示す．組込みコンピュータである Armadillo[17]に，気象センサ WXT510(Vaisala 社製)や，WM918(AmbientWeather 社製)が RS232C で接続され，解析されたデータがインターネットを介して Live E!のサーバに定期的に送信される仕組みとなっている．

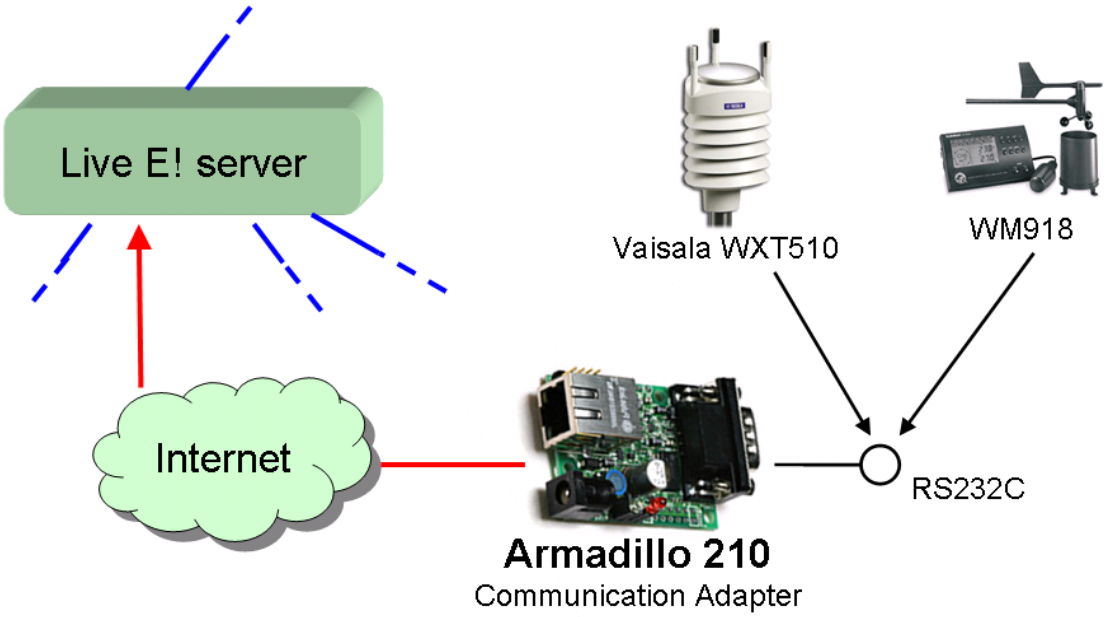



図 19: Live E!で使用しているインターネットセンサの構成

## 4.2. 運用状況

### 4.2.1. トポロジとネットワーク環境

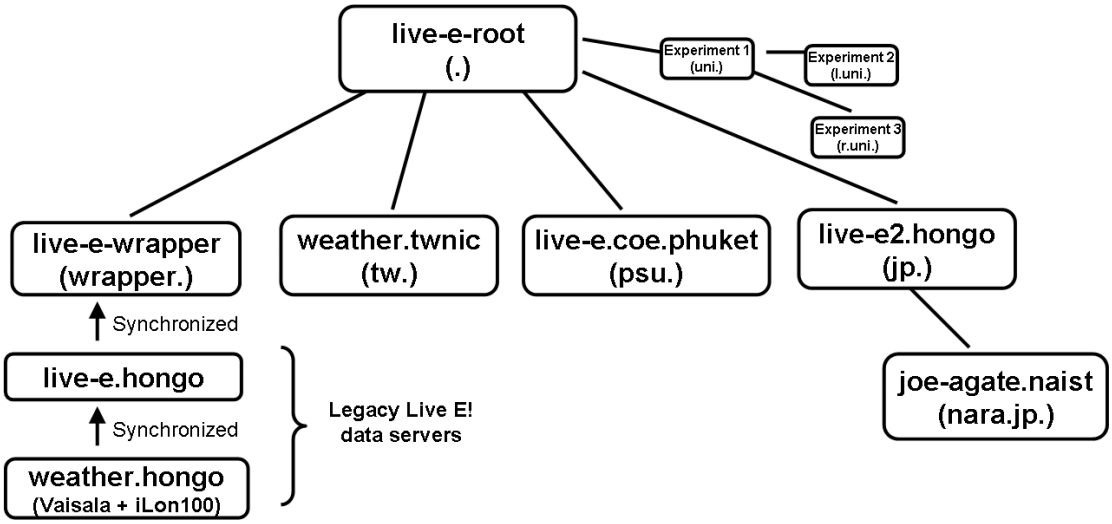



図 20: 2008 年 1 月現在の運用状況

2008 年 1 月現在, Live E!ネットワークの運用構成は, 図 20 となっている. weather.hongo および live-e.hongo は，現在の Live E!システムが開発される以前から運用されていた集中管理型サーバシステムで，live-e-wrapper により，データが現在の Live E!システムに提供されている．live-e-root(<.>)，live-e-wrapper(<wrapper.>)，live-e2.hongo(<jp.>)は東京大

学(UT)で運用され，joe-agate.naist(<nara.jp.>)は奈良先端科学技術大学院大学(NAIST)，live-e.coe.phuket(<psu.>)はタイ，PSU 大学の Phuket キャンパス，weather.twnic(<tw.>)は，Taiwan Network Information Center(TWNIC)に設置されたサーバである．図 21 に，UT から各拠点へのネットワーク上の Round Trip Time (RTT)の 1 日の変化の様子を示す．UT と PSU 間は，時間帯によって 150ms から 1100ms の間を推移し，UT と TWNIC 間は 150ms 程度，UT と NAIST 間は 8ms 程度で安定している．UT から各拠点への Traceroute の結果は付録 B に掲載する．

各サーバが持つセンサ数は次の通りである．ルートサーバ<.>で管理されているセンサは 0 個，<wrapper.>では 80 個，<jp.>では 4 個，<nara.jp.>では 2 個，<psu.>では 1 個，<tw.>では 6 個，合計 93 個である．

2008 年 1 月現在の Live E!システムに登録されたセンサの設置状況を付録 C に掲載する．

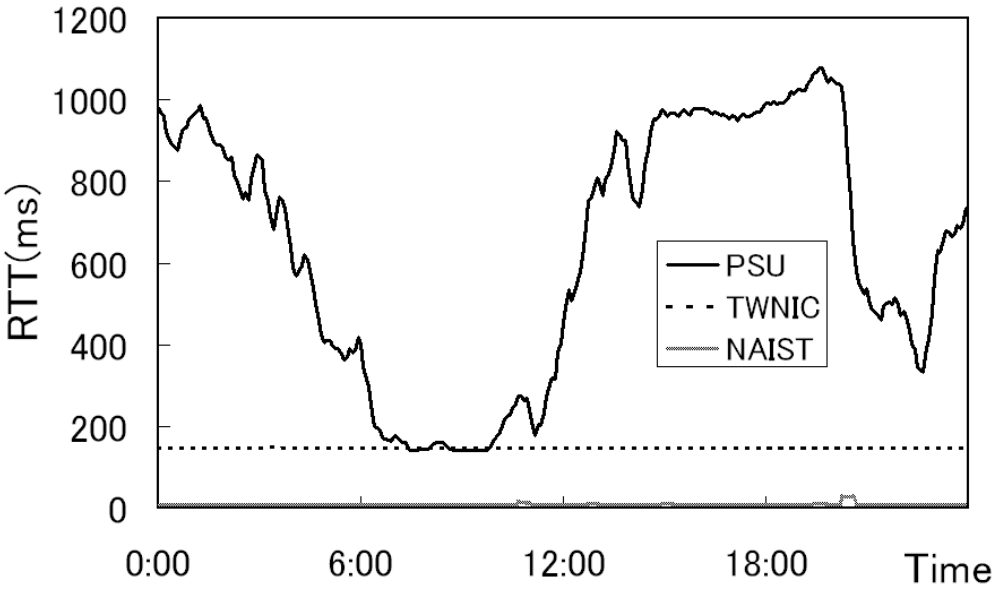



**図 21: UT から各拠点への RTT (2008 年 1 月 19 日の結果)**

### 4.2.2. プロファイルスキーマ

図 22 に 2008 年 1 月時点での Live E!プロファイルスキーマを示す．経緯度，センサベンダ，センサモデル，河川名，設置点名，地理的な住所，カメラへのアクセス方式などの情報を記載することが可能になっている．

```
<?xml version="1.0" encoding="UTF-8"?>
<profileSchema xmlns="http://live-e.org/Schema/2007/03/">
  <schema class="combined|element" name="longitude" type="float" value=".*"/>
  <schema class="combined|element" name="sensorVendor" type="string"
    value="Vaisala|AmbientWeather|Davis|Ubiteq|MatsushitaDenko|TriState"/>
  <schema class="combined" name="vehicle" type="string" value=".*"/>
  <schema class="value" name="time" type="time" value=".*"/>
  <schema class="combined" name="river" type="string" value=".*"/>
  <schema class="element" name="accuracy" type="float" value=".*"/>
  <schema class="combined|element" name="altitude" type="float" value=".*"/>
  <schema class="combined" name="ipCamera" type="string" value=".*"/>
  <schema class="element" name="sensorType" type="string"
    value="Temperature|Humidity|Pressure|DayRainFall|RainFall|WindSpeed|
           WindDir|CO2|Illuminance|AccelerationX|AccelerationY|Soil_Moisture|
           Soil_Temperature|Solar_Radiation"/>
  <schema class="element" name="error" type="float" value=".*"/>
  <schema class="combined|element" name="sensorModel" type="string“
    value="WXT510|WM918|WM928|WMR968|VantagePRO|WSN-100X|FS-Va-01|PICNICv12"/>
  <schema class="combined|element" name="latitude" type="float" value=".*"/>
  <schema class="combined" multilanguage="true" name="address" type="string" value=".*"/>
  <schema class="combined" multilanguage="true" name="location" type="string" value=".*"/>
  <schema class="combined|element" name="gAltitude" type="float" value=".*"/>
</profileSchema>
```

図 22: プロファイルスキーマ

### 4.2.3. 検索パフォーマンス

全サーバ検索はバックグラウンドで実行されキャッシュされるように運用しているため，ユーザには全サーバ検索結果は 10ms 以内で提供される．図 23 には，キャッシュ更新のためにバックグラウンドで定期的に実行される全サーバ検索時間の推移を示す．検索時間は，サーバ間の RTT に強く依存している．現在のネットワーク環境においては，UT と PSU 間の RTT と正の相関を示している(図 24).

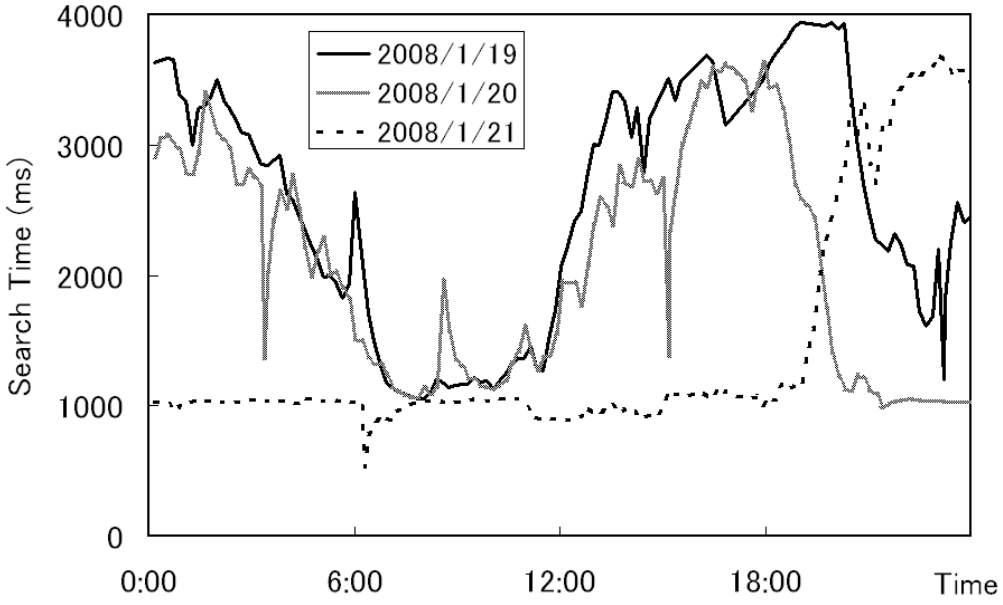



図 23: 全サーバ検索時間（2008 年 1 月 19 日～21 日の結果）

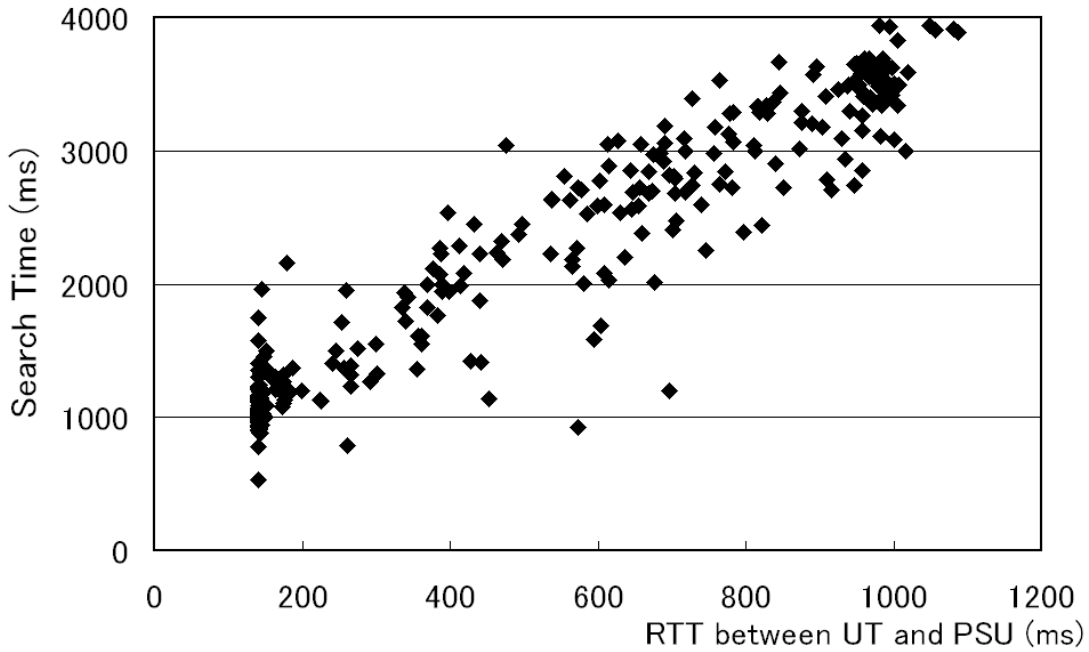



図 24: UT，PSU 間の RTT と，全サーバ検索時間の関係
2008 年 1 月 19 日～21 日のデータを使用

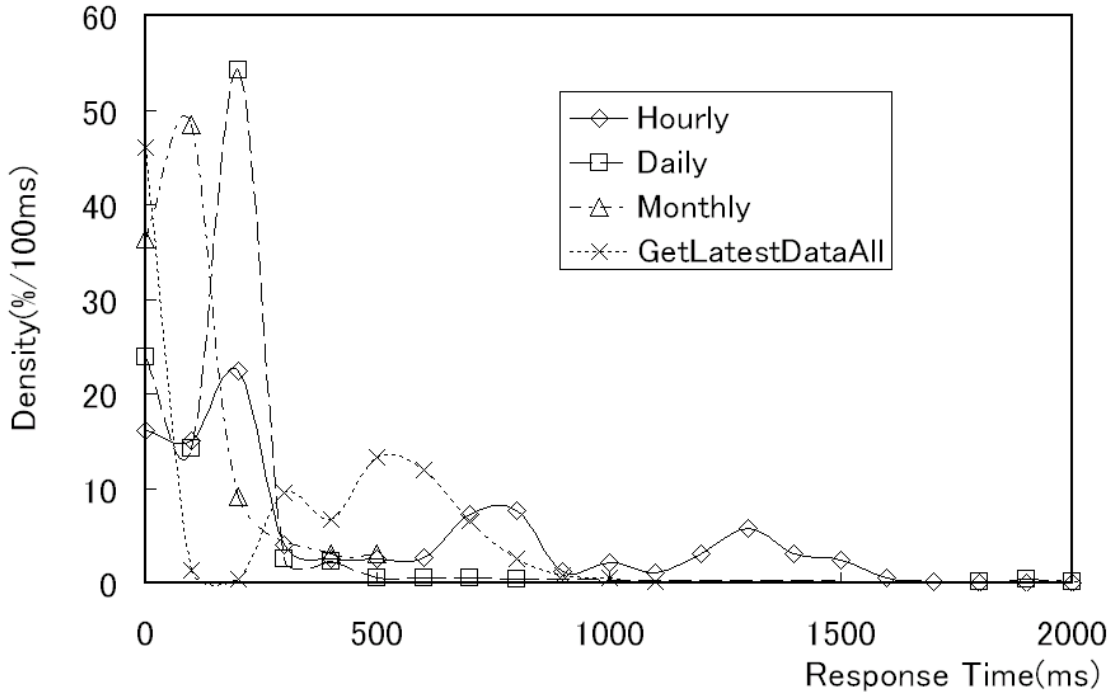



図 25: ユーザリクエスト処理時間分布(サーバ台数 7 台以内; RTT 直径 10ms 以内の場合)

図 25 に，<jp.>で運用開始時から 2007 年 12 月までに観測された「ユーザリクエストの処理時間分布」を示す．この期間においては，サーバの台数は 7 台以内で，サーバ間の RTT 直径はほとんどの場合 10ms 以内であった．リクエストごと扱うデータ量が異なるため，単純な比較はできないが，平均処理時間は 200ms～400ms 程度であった．

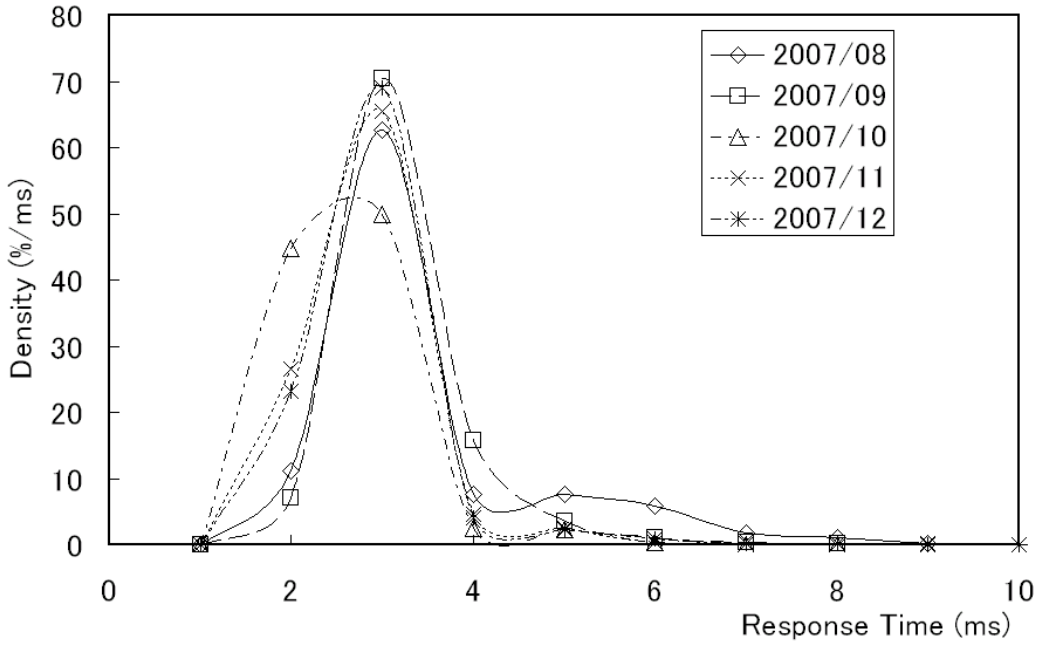



**図 26: ルートサーバでのクエリ転送処理時間分布**

図 26 に，「ルートサーバにおけるクエリ転送処理時間分布」を月ごとに示す．クエリ転送は 2～4ms でほぼ完了している．各月の総クエリ数は下記であった．

- 8 月 1026 回
- 9 月 814 回
- 10 月 3500 回
- 11 月 3992 回
- 12 月 8133 回

# 第 5 章 システム評価

## 5.1. 相互接続性と運用管理

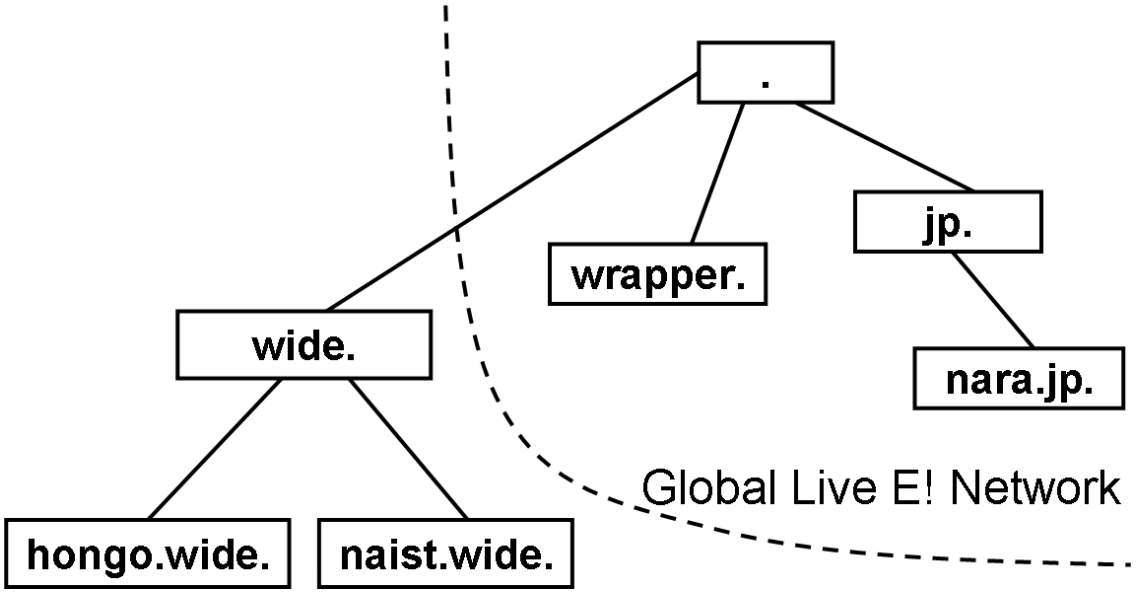



図 27: "wide"の Global Live E! Network への参加実験

センサ運用組織 wide と，その子組織 hongo，naist が独自にセンサを管理運営しており，Live E!の他組織とのデータ共有を行う状況を想定する．それぞれの組織で Live E!サーバをインストールし，図 27 のように，グローバル Live E!ネットワークに接続した(管理層で各リンクを設定した)．動作開始の数秒後，リンクの接続状態がアクティブになった．グローバル Live E!ネットワークで管理されているプロファイルスキーマが，これら 3 台のサーバに配布される様子が観測された．上位サーバからのスキーマ読込み周期を，300 秒に設定すると，10 分以内に，<hongo.wide.>や<naist.wide.>までスキーマが到達した．

こうして構築されたネットワーク上で，それぞれの組織の管理下で，独立にセンサ登録やプロファイル管理を行えることを確認した．登録したプロファイルは，上位のサーバに伝播することも確認した．伝播周期を 300 秒に設定した結果，<hongo.wide.>に登録したセンサのプロファイルは，10 分以内に<.>にまで到達することが確かめられた．ここで 10 分間は，実運用上ほとんど問題にならない時間だと考えている．従来型方式の場合，相互接続のための Wrapper 作成に，日単位での時間を要していた．つまり，$10^{-4}$ 程度の時間短縮を実現できたことになる．

同様の結果は，2007 年 8 月 30 日に中華人民共和国，西安で開催された APNG Camp での Live E! workshop(構成を付録 D に掲載)，および 2008 年 1 月 10 日にタイ，チェンマイで開催された Thai UniNet での Live E! workshop においても実証された．

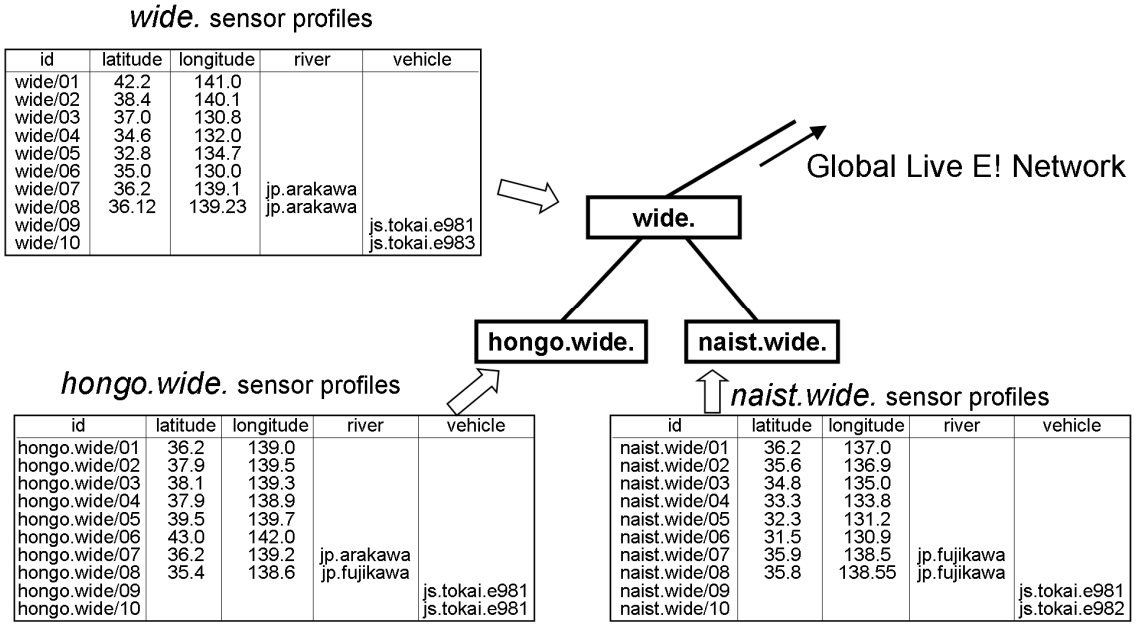
## 5.2. センサのアカウント管理による効果

旧来の Live E!システムにおいて，センサのアカウント管理をしない Zeroconf 方式で，センサ登録を行う場合と，アカウント管理をした場合とで，どの程度の ID が使われなくゴミとして残ってしまうかを調査した．

調査は以下の方法で行った．旧 Live E!システムのセンサデータアーカイブ，約 4 億 4870 万件すべてに対して，記録に残っている ID に対して，その ID に関連づけられたデータアップロードトランザクション数を算出する．そのトランザクション数が，10000 以下の ID は，一時的に運用されただけのゴミとして残っているセンサとして判断する．ゴミの ID の割合を計算する．ここで，このアーカイブへは誰でも認証は不要で書込みができるようになっている．一方，プロファイル登録に関してはゴミが明らかに増大の一途をたどった経緯があり，アカウントによるセンサ管理が行われている．

データアーカイブに記録として残っているセンサ ID の数は 1548．そのうち，一時的に運用されただけでゴミとして残っているセンサと判断された ID の数は 771．アカウント管理されているセンサ ID の数は 957 で，その中で同様の条件でゴミと判断された ID の数は 29 であった．ここから，アカウント管理を行わない場合は，49.8%の ID がゴミとして残り，アカウント管理を行う場合は，ゴミの量は 3%に抑えることが可能であったことがわかる．

## 5.3. 検索




| id | latitude | longitude | river | vehicle |
|---|---|---|---|---|
| wide/01 | 42.2 | 141.0 | | |
| wide/02 | 38.4 | 140.1 | | |
| wide/03 | 37.0 | 130.8 | | |
| wide/04 | 34.6 | 132.0 | | |
| wide/05 | 32.8 | 134.7 | | |
| wide/06 | 35.0 | 130.0 | | |
| wide/07 | 36.2 | 139.1 | jp.arakawa | |
| wide/08 | 36.12 | 139.23 | jp.arakawa | |
| wide/09 | | | | js.tokai.e981 |
| wide/10 | | | | js.tokai.e983 |

| id | latitude | longitude | river | vehicle |
|---|---|---|---|---|
| hongo.wide/01 | 36.2 | 139.0 | | |
| hongo.wide/02 | 37.9 | 139.5 | | |
| hongo.wide/03 | 38.1 | 139.3 | | |
| hongo.wide/04 | 37.9 | 138.9 | | |
| hongo.wide/05 | 39.5 | 139.7 | | |
| hongo.wide/06 | 43.0 | 142.0 | | |
| hongo.wide/07 | 36.2 | 139.2 | jp.arakawa | |
| hongo.wide/08 | 35.4 | 138.6 | jp.fujikawa | |
| hongo.wide/09 | | | | js.tokai.e981 |
| hongo.wide/10 | | | | js.tokai.e981 |

| id | latitude | longitude | river | vehicle |
|---|---|---|---|---|
| naist.wide/01 | 36.2 | 137.0 | | |
| naist.wide/02 | 35.6 | 136.9 | | |
| naist.wide/03 | 34.8 | 135.0 | | |
| naist.wide/04 | 33.3 | 133.8 | | |
| naist.wide/05 | 32.3 | 131.2 | | |
| naist.wide/06 | 31.5 | 130.9 | | |
| naist.wide/07 | 35.9 | 138.5 | jp.fujikawa | |
| naist.wide/08 | 35.8 | 138.55 | jp.fujikawa | |
| naist.wide/09 | | | | js.tokai.e981 |
| naist.wide/10 | | | | js.tokai.e982 |

図 28: 組織名，緯度経度，河川名，移動体名による検索実験環境

次の状況を想定し，組織名，経緯度，移動体名，河川名，それぞれをキーとした検索が可能であることを確認した. (1)hongo は, 東日本エリアを重点的にセンサ展開する. 河川情報に強く関係した水位センサや，列車に取り付けられたセンサも管理している. (2)naist は，西日本エリアを重点的にセンサ展開する．水位センサや，列車のセンサも管理している．(3)wide は，hongo や naist に属しない wide メンバが持つセンサを管理する. 日本全体にセンサを展開するが，河川関連のセンサや，列車についたセンサも管理している.

サーバのトポロジ上では，センサは組織で分類されているが，経緯度，移動体，河川は，複数の組織にまたがって存在するといった現実に起こりうる状況を再現している. 例えば, hongo と wide は, *river="jp.arakawa"*に関連するセンサをそれぞれで持っている.

以上の状況を想定し，図 28 のようにプロファイル登録した．検索実験と結果を，表 1 に示す．この結果からわかるように，複数の異なる検索キーで，それぞれ検索可能であった.

**表 1: 検索実験結果**

| 検索キー | 条件 | 平均検索時間(ms) | 結果 |
|---|---|---|---|
| 組織名 | * | 354 | ., jp., nara.jp., wrapper., wide., naist.wide., hongo.wide. |
| | *.wide. | 167 | wide., naist.wide., hongo.wide. |
| | naist.wide. | 58 | naist.wide. |
| 経緯度 | [36,40]×[139,140] | 248 | wrapper., wide., hongo.wide. |
| | [30,35]×[130,133] | 224 | wrapper., wide., naist.wide. |
| 河川名 | jp.arakawa | 157 | wide., hongo.wide. |
| | jp.fujikawa | 234 | naist.wide., hongo.wide. |
| 移動体名 | jp.tokai.e981 | 208 | wide., naist.wide., hongo.wide. |
| | jp.tokai.e982 | 149 | naist.wide., |
| | jp.tokai.e983 | 118 | wide. |
| 組織名+経緯度 | *.wide.+ [36,40]×[139,140] | 119 | wide., hongo.wide., |

この検索実験の結果から，問い合わせサーバ数と平均検索時間の相関をグラフにした. 図 29 にそのグラフを記す. 検索時間は, 問い合わせられるサーバ数にほぼ比例し, 約 50[ms/サーバ数]であることが読み取れる.

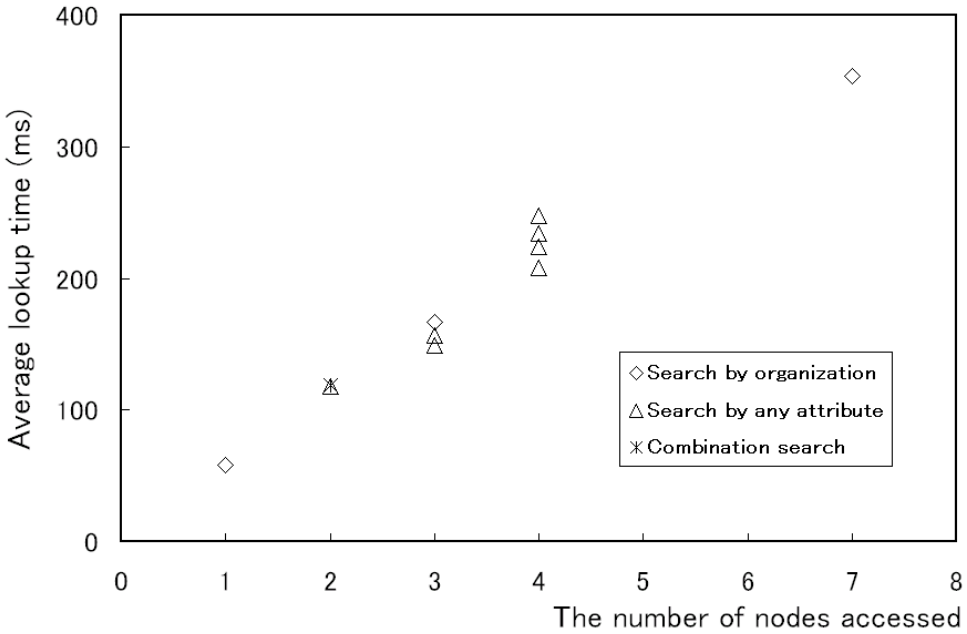



図 29: 発見サーバ数と平均検索時間の関係

## 5.4. 新規センサ分類への対応

スキーマを管理することで，新しい分類に対応できることを確かめた．具体的には，センサの分類に，道路や鉄道などの路線名を新たに加える状況を設定した．このような分類は，交通の安全確保アプリケーションにおいて，路面状態を把握する上で，今後，必要になってくると想定される．

評価実験では，ルートサーバにおいて，スキーマに下記(図 30)を追加した．

```
<schema name=”road” name=”string” value=”.*” />
```

図 30: スキーマに追加した項目

結果，このスキーマ情報は，1 時間以内に，すべての Live E!サーバに配信されることが確かめられた．<hongo.wide.> において，hongo.wide/06 に *road=”jp.route.6”*のプロファイルを付加した結果，<naist.wide.>，<jp.>などのサーバから *road=”jp.route.6”*を検索キーに，このセンサを検索可能になった．

## 5.5. センサデータの公開制御

図 27 の Live E!ネットワークトポロジにおいて，wide(hongo, naist を含む)の特定のセンサデータは，wide 内だけで閉じて，外部に公開すべきでない状況を考えた．hongo で収集しているデータは，wide 内での共有により組織全体の活動に貢献できるが，不特定多数の

外部組織への公開は不利益を生じうる場合にこのような要求が発生する．

hongo において，このセンサに対するデータ公開ポリシーを<*.wide.>のみに設定した．結果，外側の組織<wrapper.>, <jp.>, <nara.jp.>からの<hongo.wide.>のデータ読出しは不可能となり，<wide.>, <naist.wide.>, <hongo.wide.>からのみ読出し可能になった．こうして目的のアクセスコントロールを実現できていることが確かめられた．

# 6. 考察

サーバ間のデータ表現やアクセスインタフェースを標準化することで，ネットワークのトポロジに柔軟性を持たせることが可能になった．設定されたトポロジの上で，プロファイル情報やスキーマなどが交換され自律分散的に広域検索を可能にした．このような点において自律分散性を持たせたことで，ユーザは検索経路の設定を行わなくても良くなっている．

一方で，本提案システムではネットワークトポロジ設計や，スキーマ設計，アカウント発行，アクセスコントロールをユーザに任せることで，計画的な運用を可能にしている．アカウント発行による管理運用方式を導入することにより，Zeroconf 方式では 49.8%の登録 ID がゴミ ID として残ってしまうという現象を削減することが可能になった．アーカイブに登録された ID から算出した結果であるが，プロファイルにゴミとして残ってしまう現象を削減できることを意味している．これはシステムとしての信頼性を向上させる上で重要なことである．

実験により，組織名をキーとした検索の他，様々な属性を検索キーにして広域検索が実現できることを確認した．具体的には，経緯度による地表面の領域を指定してのセンサ検索，特定の河川に関連が深いセンサの検索，移動体に結合されたセンサの検索が実現できることを確認した．特定のモデルに依存しないで，異なるタイプのキーで検索可能なことを示している．本研究の段階では，スケール性を実現するための仕組みについては，扱っていない．

現在の運用環境では，サーバ間のネットワーク的な遅延に検索パフォーマンスが大きく依存している．キャッシュ技術により，検索パフォーマンスの低下をユーザには見せないようにしているが，キャッシュにはデータ一貫性の課題がある．また，ユーザからの要求が稀なものであれば，満足するキャッシュが存在しない状況も考えられる．キャッシュを大きくし，キャッシュ時間を長く設定することは，ユーザへ検索パフォーマンスを良く見せることができるが，他の側面(i.e.，リアルタイム性，キャッシュのための記憶容量)での質が低下することが予想される．

検索の実験において，検索プロセスでアクセスするサーバ数に検索時間が正比例することが確かめられた．これはシングルスレッドによる検索のため予想されていたことだが，マルチスレッドで検索したとしても，対象数が多い状況下では正比例に近い形になることは予測できる．一般に，検索対象となるサーバ数が増加すると，正比例に検索が終了するまでの時間がかかる．検索対象となるサーバの数を低く抑えることが検索効率の向上のために重要であると考察できる．

ネットワーク遅延に関する現象の調査，および検索時にアクセスするサーバ数の実験から，現在の実装システムにおいて，検索時間は次の式で与えられることが予測される．

$Serverlookuptime = 3\sum RTT_n$　ただし $RTT_n$ は，ノード $n$ との通信遅延を表す．

# 7. 今後の課題

## 7.1. 運用スキーム

実際的な運用にあたっては，ネットワークのトポロジ設計を計画的に行う必要がある．例えば，地域単位の検索が想定される状況では，国ごと地域ごとに運用サーバを計画的に配置することで，検索に伴う分岐が減少し，検索効率を上げることが可能になると考えられる．計画的な割り当てによって，システム全体のパフォーマンスが向上するとする見解は RFC2050[18]でも述べられている．どのような検索モデルが頻繁に利用され，その検索パターンがどのようになるかを予測，あるいはプロトタイプシステム上で観測した上で，トポロジ設計における割り当てポリシーを設定することが望ましいだろう．

計画的な運用には，プロファイルスキーマの管理も挙げることができる．プロファイルスキーマを発行するルートサーバの役割は，インターネットでの Internet Assigned Numbers Authority(IANA)[19]に相当する．スキーマを動的に配信する仕組みを導入したことで，システム全体に渡ったプロファイル記法や言語表記，そして検索キーの管理を可能にしたが，具体的なプロファイルスキーマのモデルをどのように設定するかは運用上の課題として位置づけられる．

広域センサネットワークにおいては，ユーザへの検索効率を向上させるために，センサデータのキャッシュは欠かすことができないと思われる．ここでデータの一貫性を向上させるためにはキャッシュ時間を小さくすることで対応し，検索効率を向上させるためにキャッシュ時間を長くすることで対応できる．運用段階において，データの一貫性に関する許容範囲を明確にした上で，検索効率をできるだけ高く保つことが可能なキャッシュ時間を設定しなければならない．センサの場合，データは連続的に変化する値であり，許容条件の設定次第では，キャッシュの有効さを時間基準にすることが妥当かどうかという考え方もありうる．

本研究では，センサデータに対してユーザのアクセスコントロールを行うメカニズムを導入している．センサデータを提供可能な組織名，提供不可能な組織名を，サーバの運用者が設定することが可能であり，これは比較的小さなコストで運用できると考えている．本研究では，センサデータの管理はそのセンサの運用サーバ(オリジンサーバ)にあり，観測されたセンサデータが，許可された組織にのみ，その設定の期間中提供されるようになっている状況を想定している．ただし，この状況下では，データをおおむねオリジンサーバから取得せざるを得ないので，検索およびデータの読み出し効率を高く保つことができないと考えられる．そこで，将来的には，現時点で観測されたセンサデータが，今後，永遠に自由に流通されることを許可する状況を考え，その状況下で効率よく検索できるようなシステムをデザインする必要もあるだろう．

## 7.2. 検索効率

本研究では，種々検索キーでの広域検索を実現するために，センサプロファイルを上位側サーバに読み込ませることで，検索クエリを広域にわたってルーティングさせる手法を提案し，一定の規模においては実現可能であることが明らかとなった．本研究で扱った規模であれば，集中インデックスサーバを設置し，そこに問い合わせるだけでクエリに合致するすべてのノードを応答する検索方式も可能かもしれない．あえてクエリ・ルーティングにより検索を実現する手法[14]を採用したのは，将来的に様々なルーティング実装を組込み可能にするためである．

具体的には経路アグリゲーションによって得られる効果を期待する場合が考えられる．下位サーバノードが保持するセンサプロファイルをすべて上位に転送することでクエリ経路表を作成したが，ここでプロファイルデータのアグリゲーションを行うことにより，経路の粒度を粗くし，経路表を小さくすることが可能になる．経路表を小さくすることは，たいていの場合，それぞれのノードでのクエリ転送時間を小さくすることにつながる．一方で，無駄なノードにまでクエリの転送が分岐してしまうことによる損失も考えられ，実際的な環境でどのようにアグリゲーションを行うのが妥当かは今後の検討課題である．

本研究で提案した広域センサネットワークにおいては，Iterative 方式でのサーバ検索を行っている．Iterative 方式は，Recursive 方式と比べ，問い合わせセッション時間を小さく抑えることができるため，サーバノードの負荷軽減や，タイムアウト時間を低めに設定することが可能だといった利点を持つ．一方で，検索の分岐が増えた場合に，検索時間が *O(2^d)* で増加する他(*d* は検索の深さ), IP ネットワーク上で離れたノードセットを順にめぐる場合に，ネットワーク遅延による検索効率の低下が顕著に現れる[17]という欠点もある．検索エンジンによる Iteration をマルチスレッド化したり，検索とデータ読出しをパイプライン化したりすることで検索効率の向上が見込まれるが，これらの手法のみでは根本的な解決にはならず，検索の分岐をどれだけ小さくすることが可能か，が重要な指標になると考えている．

検索効率は, 下位層である IP ネットワークの特性に影響される. ネットワーク的に遠く，通信遅延(RTT)が大きい状況下では，検索に大きな時間を要する．ネットワーク的に遠い環境が存在する状況下でも，バックグラウンド検索や，結果のキャッシュ，そして検索頻度を調整するなどして，ユーザに提供する検索効率を向上させることは可能だと考えられる．

## 7.3. Query-initiative 型と Data-initiative 型の共存

Query-initiative 型システムは，データを分散させておき，クエリを発行することで，目的のデータを取得する方式のシステムである．Data-initiative 型システムはクエリを予め発行し分散させておくことで，発生するデータを転送しユーザに提供する方式のシステム

である[8]．Query-initiative 型システムはユーザによるポーリング型システムで，Data-initiative 型システムはユーザへのイベント通知型システムと見ることもできる．

センサから発行されたデータが，ユーザに提供されるまでのシステムデザインは，下記の観点で分類することができると考えられる．

- 検索要求の頻度
- データのリアルタイム性

実際的なシステムにおいては，これらの性質や要求条件が広くわたることが考えられる．それらの条件に適合できるようにシームレスに結合されたシステムデザインの研究が求められる．

サーバ・サーバ間の通信において，ユーザ数が少ない等の理由で，検索頻度が低い状況であれば，ポーリング手法によりオンデマンドにデータを取ってくる方式が考えられるだろう．このような状況下では，キャッシュの効果はあまりないため，キャッシュが存在しなくても良い．

ユーザ数が増えてくると，同一要求による検索頻度が増大してくる．ただし，これらの要求は，必ずしもデータのリアルタイム性を必要としているわけではない．通常，対向サーバの検索や，データの読出しにいくらかの時間がかかるため，ユーザ側のサーバで，対向サーバの resolve 情報やセンサデータを定期的にキャッシュしておくことで検索コストの削減と，応答パフォーマンスの向上を計ることができる．

対向サーバからのデータ読出しコストが大きい状況下(e.g., ネットワーク遅延が大きいなど)では，キャッシュ時間を長く設定することが普通である．しかし，有効期限方式によるキャッシュの場合は，データソースとの間で，一貫性にずれが生ずる場合がある．特にユーザがデータのリアルタイム性を求める場合，長い時間に設定されたキャッシュはユーザの要求を満足しない状況を生み出しかねない．そこでデータに更新があった場合に，その情報をキャッシュに通知し，更新する必要が生じてくる．これがイベント型配信である．イベント型配信において，どのような状況を更新の必要な状況と判断するかは，別途研究の余地がある．タイムスタンプの更新，値の変動，値の大きな変動，特定の値が 0 から別の値に更新された場合など，様々な状況が，更新の必要な状況の候補として考えられる．データ配信のコストとの兼ね合いが課題である．

上記のように，ユーザからの要求によって様々なシステムデザインが考えられる．これらをシームレスに結合し，動的に動作形態を選択していくことが可能なシステムを今後研究していく必要がある．

## 7.4. センサデータ処理

本研究で開発したシステムでは，特定の時間領域の中での，センサデータアグリゲーションも扱った．最小値，平均値，最大値，和，数を計算し，アグリゲーションテーブルに

保存することで，実際的なアプリケーションでの利便性を向上させている．しかし，センサデータにはノイズが含まれる場合がある他，値を比較する際には，高度による補正が必要な場合も存在する．さらには，データの補完を行うことで，より正確に意味のある値を計算する必要もある．これらの計算処理は，システムから生データを取り出した後で，アプリケーションで処理することも不可能ではないが，よく利用される処理結果は保存しておき，これを利用させるようにすることで，応答速度の向上とデータ転送量の削減が期待できる．

ノイズフィルタには，単純に値を閾値判定するものから，値の変動を計測することで，学習的にフィルタするものなど，様々なものが考えられる．気圧や温度などのデータは，プロファイル情報などを参考にし，海面補正を実行することで，平均値を意味のあるものにすることができる．アグリゲーションは，時間領域の中のみで行われるべきではなく，空間的な領域の中で行われることもありうる．アグリゲーションの領域の大きさは，1 時間，1 日，1 ヶ月などの基本的な単位のみではなく，2.5 時間，5.9 時間など特定の領域を単位に行われるような状況も考えられる．本提案システムを磨き上げ，実用的なシステムに持っていくためには，上記に挙げた様々な要求を処理することが可能であることが求められる．

## 7.5. 監視システム

センサシステムを運用する者にとって，センサ等の動作状態を把握することは大切なことである．監視システムとしては，(1)センサの生命監視(動作検証レベル)と，(2)アプリケーションレベル監視システムが考えられる．

センサの生命監視システムは，センサの動作を常に監視し続け，センサが停止したとき，不可能な値を生成したとき(e.g.，湿度 160%)，タイムスタンプが極端に過去もしくは未来のものであるとき，その他の挙動が異常になったときに，警告メッセージとして管理者に伝えるものである．今後，異常状態のパターンを調査し，その検出アルゴリズムについて提案していく必要がある．

アプリケーションレベル監視システムは，センサの動作が正常で信頼性があると判断された上で，センサデータの挙動から何らかの事象を検出するシステムである．例えば，雨が降り出したとか，風が強くなってきているとか，そのような情報を抽出し，イベントとして取り出すシステムが考えられる．具体的にアプリケーションのレベルで，どのような監視システムが作られるかは今後の運用で明らかになってくると思われる．同時に，広域センサネットワークに要求される事象が明らかとなり，取り組むべき課題が見つかるだろう．

## 7.6. データ収集の技術

本提案システムでは，センサからのデータ通知インタフェースを定義し，センサからサーバへ直接的にインターネットを使ってデータ通知をしてくる状況を考えている．センサが送信したデータが，非同期であっても良いので最終的にサーバに転送されてくればよいのであって，必ずしもセンサがインターネットにつながっている必要はない．山中の農場に，センサ設置のためのインターネット回線を提供するのではなく，付録 E に示したように MANET 無線センサノードを設置し，自動車などの移動体が巡回してデータを回収し，最終的にサーバに送信できれば良い状況も多々存在する．このような状況下では，センサから直接はデータをサーバに送信できないので, インターネット上にデータ送信用の Proxy サーバを設置することになるだろう．課題としては，センサがデータをアップロードするにはアップロード先のサーバが指定され，認証が必要なことである．これらのセッション情報や認証情報が透過的にやり取りできなければならない.

# 8. おわりに

本研究では，インターネット上のセンサデータを自律分散的に共有可能にする Live E!広域センサネットワークの提案を行った．提案においては，プロトタイプシステムによる 2 年間にわたる運用経験から，実際的なシステムに求められる要求条件を抽出，整理することから始め，それを実現するアーキテクチャの提案を行った．具体的には，階層化アーキテクチャを考え，管理/API 層，検索層，データ管理/転送層，センサ層にシステムを分割した．管理/API 層では，グローバルスキーマの管理，トポロジの管理，ユーザアプリケーションへのインタフェースを提供する. 検索層では, サーバ名(組織名)をキーとした検索の他，プロファイル情報をネットワーク内で定期的に交換することにより，各種センサの属性をキーとした検索を実現する．データ管理/転送層では，センサデータの保存，データ提供先のアクセスコントロール，外部サーバへのデータ提供を行う．センサ層では，インターネットセンサから送信されてくるセンサデータを受信するが，この際にセンサ認証を行うことで，システムとしての統制を高める．

本提案システムにより，分散的に運用されるセンサシステム間での相互接続性が実現された．最高権威組織がスキーマを運用することで，新規にセンサの分類を追加することが可能になった. センサプロファイルの交換により, サーバ名での検索に加えて, 各種属性(e.g., 経緯度，河川名，移動体名)をキーにした検索が可能になった．これは，管理情報であるスキーマや，経路情報としてのプロファイルが自律的にネットワーク内で交換されることで実現されている．組織のサーバに運用するセンサデータを保存するように制限を設けることで，ユーザ認証(i.e., データ提供先のアクセスコントロール)を実現した．検索パフォーマンスについては，ネットワーク遅延や検索時にアクセスされるサーバ数の方が，クエリ転送処理時間よりも大きな影響を与えることが明らかになった．

将来的な課題として，運用計画に関する課題，検索パフォーマンスや通信形態上の課題，センサデータ処理に関する課題が発生した．運用上の課題としては，トポロジ設計やサーバ名の割当てに関する方針，スキーマ管理方針が挙げられる．検索パフォーマンスや通信形態上の課題としては，経路アグリゲーションへの挑戦，検索分岐の少ないデータ配置手法の考案，効率的なキャッシュ技術，Data-initiative 型と Query-initiative 型の共存などがある．システム内部で行われるべきセンサデータ処理に関する課題には，誤データのフィルタリング，補正，補完，データアグリゲーション，監視機能などがある．

# 謝辞

本研究は，江崎浩教授が主査を務める Live E!プロジェクト(2005 年 5 月発足)を研究活動の基礎においている．筆者は，Live E!プロジェクトに 2005 年 11 月から関わり，この場で，本研究を進めるための土台を提供してもらった．本研究でシステムを開発するまでの意見交換においては，WIDE プロジェクトの多くの方々にもお世話になった．特に，奈良先端科学技術大学院大学のインターネット・アーキテクチャ講座のたくさんの学生および砂原秀樹教授，松浦知史氏には大変お世話になった．2007 年 8 月に，本提案システムの運用開始後，Asia Pacific Network Group(APNG)に後押ししてもらい，Live E! ワークショップを開催，実証実験を兼ねた研究成果発表を行った．2007 年 12 月からは，Taiwan Network Information Center (TWNIC)や Prince of Songkla University(PSU) と交流を持ち，Live E!サーバの運用を開始してもらうことができ，本研究を充実させるための努力を注いでもらっている．PSU の Sinchai 教授には，2008 年 1 月に ThaiUniNet へ招待してもらっている．

東京大学情報基盤センターの中山雅哉准教授にも，海外における活動や，研究発表等，様々な面からサポートしてもらった．筆者の所属する江崎研究室においては，先輩の山本先生，岡部氏，吉田氏，藤田氏と，長い間研究生活を共にし，多くの面でアドバイスをもらった．同輩の田中氏，安本氏，山口氏，王氏，賈氏とは，互いに助け合い，後輩にあたる杉山氏，阪本氏，大口氏，Sergio 氏，姜氏，肥村氏，浅井氏には，研究などにおいて，いろいろとお手伝いをしてもらった．活動に伴う旅費などの支援や，事務的な処理において，江崎教授や秘書の高橋さん，田坂さんにはいつもお世話になった．

江崎先生には，研究の方法や方向性，技術的な側面，活動に対する激励，チームリーダーとしての在り方など，すべてにおいてお世話になった．特に，Live E!プロジェクトの場や，海外の研究機関との交流の場をたくさん紹介してもらい，筆者がより多くの経験をできるようにと，積極的に取り計らってもらった．

東京大学入学から現在までの 6 年間，静岡で暮らす両親には，筆者が東京で暮らし，安心して勉学や研究に努めるための支援をしてもらってきた．

本研究は，多くの方々のサポートの結果として，現在の段階を迎えており，筆者はそのありがたさを改めて感じここに記すと共に，今後も研究を進め，社会全般にとって役に立つものを生み出していきたいと考えている．

# 参考文献


[1] P. Wegner: “Interoperability”, ACM Computing Surveys, 28, 1, pp. 285—287 (1996).

[2] S. Ninomiya, T. Kiura, A. Yamakawa, T. Fukatsu, K. Tanaka, H. Meng and M. Hirafuji: “Seamless integration of sensor network and legacy weather databases by MetBroker”, In Proceedings of IEEE/IPSJ SAINT2007 workshop, p.68 (2007).

[3] MetBroker, http://www.agmodel.net/agmodelJa/projects/metbroker.html

[4] K. Aberer, M. Hauswirth and A. Salehi: “Infrastructure for data processing in large-scale interconnected sensor networks”, In Proceedings of IEEE MDM2007 (2007).

[5] S. Matsuura, K. Fujiwara and H. Sunahara: “Mill: A geographical location oriented overlay network managing data of ubiquitous sensors”, IEICE Transactions on Communications, E90-B, 10, pp. 2720—2728 (2007).

[6] P. B. Gibbons, B. Karp, Y. Ke, S. Nath and S. Seshan: “IrisNet: an architecture for a world wide sensor web”, IEEE Pervasive Computing, 2, 4, pp. 22—33 (2003).

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan: “Chord: a scalable peer-to-peer lookup service for internet applications”, In Proceedings of ACM SIGCOMM, pp.149—160 (2001).

[8] H. S. Lim, J. G. Lee, M. J. Lee, K. Y. Whang and I. Y. Song: “Continuous query processing in data streams using duality of data and queries”, In Proceedings of ACM SIGMOD (2006).

[9] D. J. Abadi, W. Lindner, S. Madden and J. Schuler: “An integration framework for sensor networks and data stream management systems”, In Proceedings of the 30th VLDB Conference, pp.1361—1364 (2004).

[10] H. Ochiai, Z. Wang, R. Oguchi, A. Sugiyama, Y. Sakamoto, S. Ishida and H. Esaki: “Application of content-based network for sensor data distribution system”, In Proceedings of IEEE/IPSJ SAINT2007 workshop, p.74 (2007).

[11] M. Nakayama, S. Matsuura, H. Esaki and H. Sunahara: “Live E! project: sensing the Earth”, LNCS, 4311, pp.61—74 (2006).

[12] P. V. Mockapetris and K. J. Dunlap: “Development of the domain name system”, In Proceedings of ACM SIGCOMM, pp.123—133 (1988).

[13] A. S. Tanenbaum and M. V. Steen: “Distributed Systems”, chapter General Issues in Access Control, pp. 414—418, Peason Education, Inc., second edition (2006).

[14] H. Crespo and A. Garcia-Molina: “Routing indices for peer-to-peer systems”, In Proceedings of IEEE Distributed Computing Systems, pp.23—32 (2002).

[15] A. S. Tanenbaum and M. V. Steen: “Distributed Systems”, chapter Implementation of Name Resolution, pp.205—209, Peason Education, Inc., second edition (2006).

[16] Apache Web Services Project, http://ws.apache.org/

[17] AtmarkTechno, http://www.atmark-techno.com/

[18] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg and J. Postel: “RFC2050: internet registry IP allocation guidelines” (1996).

[19] IANA, http://www.iana.org/

# 付録 A – Live E! サーバの提供するサービス

| | | |
|---|---|---|
| **サービス名** | **Admin200703** | |
| サービス概要 | サーバ間でコントロール情報を交換する | |
| メソッド | getProfileAll | プロファイルを経路情報として提供 |
| | queryService | このサーバノードにあるサービスの解決 |
| | getNeighbor | 近隣ノード情報の提供 |
| | getSchema | このサーバノードが持つプロファイルスキーマ |
| | replicate | リンクコントロール情報のコピーを作成(冗長化用) |
| | search | クエリ処理(e.g., クエリ応答, クエリ転送) |

| | | |
|---|---|---|
| **サービス名** | **ProfileManagement200703** | |
| サービス概要 | プロファイルの操作(e.g., 読出し, 登録, パスワード更新) | |
| メソッド | getProfileSchema | プロファイルスキーマの提供 |
| | getAvailableLocales | 利用可能な言語ロケール |
| | getProfileAll | このサーバノードが管理する全プロファイル |
| | getProfile | 指定したセンサのプロファイル読出し |
| | setProfile | 指定したセンサのプロファイル登録 |
| | updatePassword | 指定したセンサのアクセスパスワード更新 |

| | | |
|---|---|---|
| **サービス名** | **DataUpload200703** | |
| サービス概要 | センサデータのアップロードを受付 | |
| メソッド | uploadElement | センサの最小単位でデータをアップロード |
| | uploadCombined | センサユニット単位でデータをアップロード |
| | uploadCollection | 複数ユニットのデータをまとめてアップロード |

| | | |
|---|---|---|
| **サービス名** | **DataManagerReplication200703** | |
| サービス概要 | Master サーバノードのセンサデータレプリカ作成 | |
| メソッド | replicateProfile | プロファイルのコピーを作成 |
| | replicateLatestData | 最新センサデータのコピーを作成 |
| | replicateArchiveData | 過去データ(アーカイブ)のコピーを作成 |

| サービス名 | DataProvider200703 | |
|---|---|---|
| サービス概要 | サーバノードのローカルセンサデータを提供 | |
| メソッド | getProfileSchema | プロファイルスキーマ |
| | getLatestDataAll | 全センサの最新データ |
| | getLatestData | ID 指定したセンサの最新データ |
| | getLatestDataByAreaRect | 領域指定したセンサの最新データ |
| | getArchiveCombinedData | ID 指定したセンサの過去(生)データ |
| | getDataHourlyAggregated | 時単位で集約したセンサデータ |
| | getDataDailyAggregated | 日単位で集約したセンサデータ |
| | getDataMonthlyAggregated | 月単位で集約したセンサデータ |
| | getProfileAll | 全センサのプロファイル |
| | getProfile | ID 指定したセンサのプロファイル |
| | getProfileByAreaRect | 領域指定したセンサのプロファイル |
| | getArchiveProfile | プロファイルの更新履歴 |

| サービス名 | GlobalDataProvider200703 | |
|---|---|---|
| サービス概要 | グローバル Live E!ネットワークのセンサデータを提供 | |
| メソッド | getProfileSchema | プロファイルスキーマ |
| | getLatestDataAll | 全センサの最新データ |
| | getLatestData | ID 指定したセンサの最新データ |
| | getLatestDataByAreaRect | 領域指定したセンサの最新データ |
| | getArchiveCombinedData | ID 指定したセンサの過去(生)データ |
| | getDataHourlyAggregated | 時単位で集約したセンサデータ |
| | getDataDailyAggregated | 日単位で集約したセンサデータ |
| | getDataMonthlyAggregated | 月単位で集約したセンサデータ |
| | getProfileAll | 全センサのプロファイル |
| | getProfile | ID 指定したセンサのプロファイル |
| | getProfileByAreaRect | 領域指定したセンサのプロファイル |
| | getArchiveProfile | プロファイルの更新履歴 |
| | search | クエリによるサーバ検索 |

# 付録 B – Live E! サーバ間の IP 層経路状況

```
traceroute to live-e.coe.phuket.psu.ac.th (202.12.73.172),
15 hops max, 40 byte packets
 1  203.178.135.1  0.185 ms  0.122 ms  0.140 ms
 2  203.178.138.243  0.432 ms  0.410 ms  0.410 ms
 3  203.178.133.141  0.466 ms  0.454 ms  0.451 ms
 4  203.181.248.109  11.368 ms  0.552 ms  0.547 ms
 5  192.203.116.148  115.996 ms  115.967 ms  115.965 ms
 6  207.231.240.148  116.012 ms  115.995 ms  116.034 ms
 7  202.28.212.45  965.905 ms  953.450 ms  938.711 ms
 8  202.28.218.17  956.153 ms  978.884 ms  950.657 ms
 9  202.28.218.14  964.105 ms  930.868 ms  943.251 ms
10  202.28.212.166  238.571 ms  238.409 ms  238.247 ms
11  202.28.216.10  926.585 ms  933.289 ms  963.742 ms
12  202.28.221.22  923.573 ms  935.958 ms  978.315 ms
13  * * *
14  * * *
15  * * *
```

UT から PSU に設置された Live E!サーバ(psu.)への Traceroute 結果

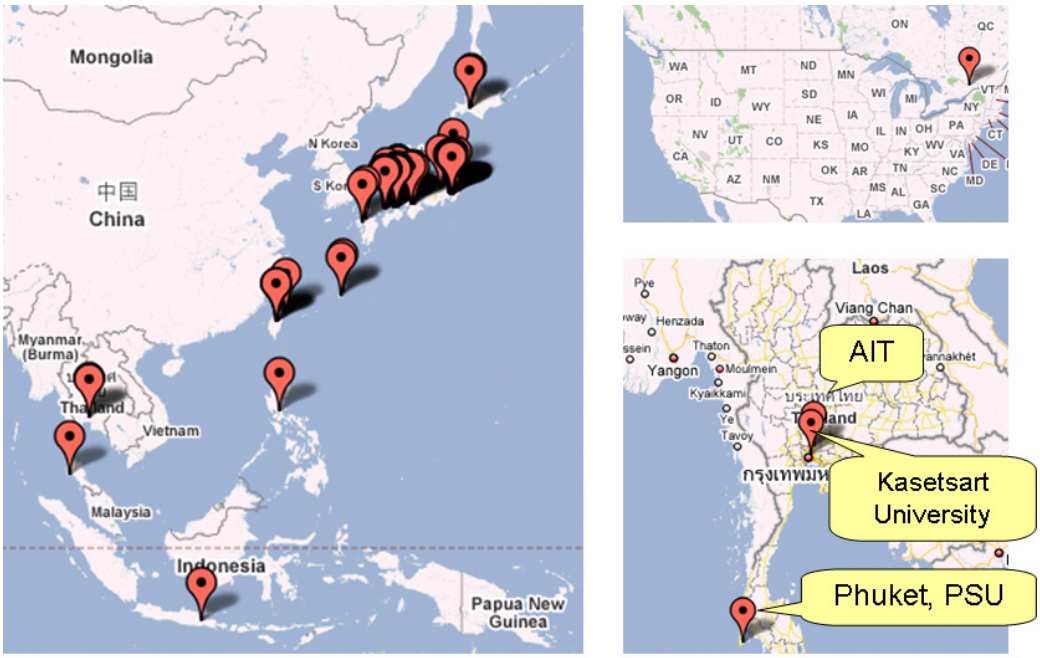
```
traceroute to weather.twnic.net.tw (211.72.210.101),
 30 hops max, 40 byte packets
 1  203.178.135.1  0.205 ms  0.130 ms  0.142 ms
 2  203.178.138.243  0.454 ms  0.411 ms  0.407 ms
 3  203.178.141.141  0.376 ms  0.319 ms  0.309 ms
 4  192.50.36.49  109.665 ms  109.677 ms  110.281 ms
 5  157.130.206.41  124.427 ms  124.781 ms  125.273 ms
 6  152.63.49.26  124.609 ms  125.042 ms  128.720 ms
 7  152.63.55.126  134.476 ms  124.949 ms  124.719 ms
 8  152.63.113.21  126.818 ms  128.125 ms  125.943 ms
 9  152.63.144.98  125.933 ms  128.358 ms  129.578 ms
10  63.65.130.118  126.300 ms  127.261 ms  126.146 ms
11  202.39.83.189  126.887 ms  126.052 ms  127.033 ms
12  211.72.108.162  256.865 ms *  261.990 ms
13  220.128.3.50  152.733 ms  153.026 ms  153.895 ms
14  220.128.1.141  152.754 ms  152.577 ms  153.903 ms
15  203.75.188.67  146.428 ms  146.613 ms  146.477 ms
16  211.72.234.13  156.765 ms  161.734 ms  161.064 ms
17  211.72.210.251  167.063 ms  165.341 ms  159.586 ms
18  211.72.210.101  160.991 ms  152.717 ms  161.134 ms
```

UT から TWNIC に設置された Live E!サーバ(tw.)への Traceroute 結果

```
traceroute to joe-agate.naist.jp (163.221.170.54),
 15 hops max, 40 byte packets
 1  203.178.135.1  0.213 ms  0.128 ms  0.145 ms
 2  203.178.138.243  0.429 ms  0.401 ms  0.411 ms
 3  203.178.136.238  7.831 ms  7.821 ms  7.819 ms
 4  203.178.136.171  10.763 ms  10.646 ms  10.702 ms
 5  163.221.1.9  8.671 ms  8.615 ms  8.623 ms
 6  163.221.5.67  8.702 ms  8.651 ms  8.646 ms
 7  163.221.5.227  9.053 ms  8.992 ms  8.999 ms
 8  163.221.170.54  8.673 ms  8.635 ms  8.636 ms
```

UT から NAIST に設置された Live E!サーバ(nara.jp.)への Traceroute 結果

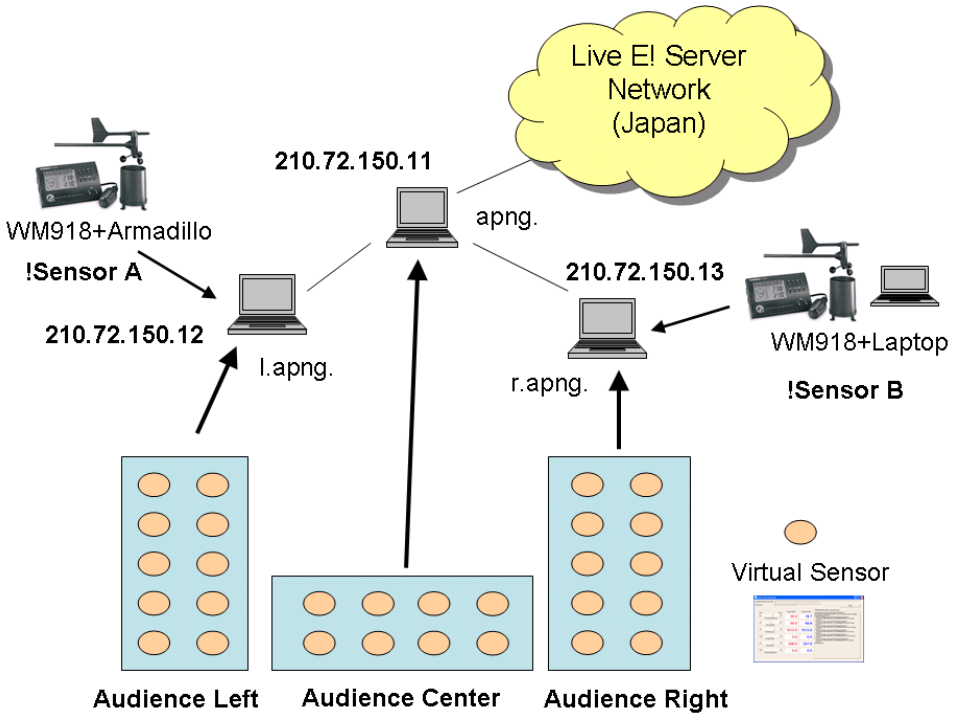
# 付録 C – Live E! センサの展開状況




http://www.map-asp.net/Spatial_Gateway/pl/Gate_100.html

# 付録 D – 西安で行われた実証実験

～ APNG Camp Live E! workshop にて ～



# 付録 E – 無線センサパッケージの構成