第 XVIII 部

環境情報の自律的な生成・流通を可能にするインターネット環境の構築

第 18 部
環境情報の自律的な生成・流通を可能にする インターネット環境の構築

第 1 章　はじめに

Live E! ワーキンググループでは気象センサを中心にセンサデータを誰でも自由に利用できる情報基板の構築を目指し、活動を展開している。

本年度は基盤システムの分散化を可能とする技術の開発・実装を行った。また国内外の研究機関と協力し、実際に分散環境を構築した。国際展開は昨年度よりもさらに加速し、アジアを中心にワークショップの開催やサーバおよび気象センサの設置なども進んでいる。

現在、アジアを中心にセンサ設置を進めている。本年度は台湾、インドネシアで新たにセンサが設置された。2008 年 1 月 7 日現在のセンサ設置状況を図 1.1、1.2 に示す。現在のセンサ登録実績は 200 程度で内 100 程度が現在稼働中である。

図 1.1.　センサ設置状況

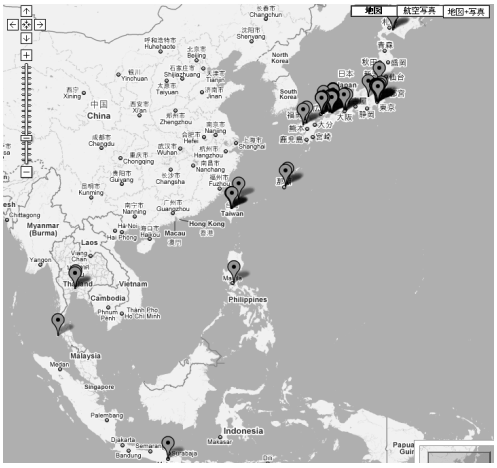

図 1.2.　センサ設置状況（アジア）

第 2 章　Live E! システムデザイン

Live E! プロジェクト [121] 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、本プロジェクト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き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。本プロジェクト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。

2.1 アーキテクチャ

Live E! 広域センサネットワークとして、図 2.1 に示す階層型アーキテクチャを提案する。システム全

体としてこの階層構造を持つが、サーバ単独でもこの階層構造を持っている。サーバ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 1 つのシステムとして協調動作する。

ネットワーク・トポロジは、図 2.2 に示すような階層構造になるように、管理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 [117] 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などが挙げられる。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。

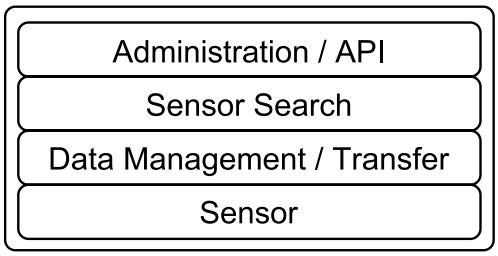

図 2.1.　Live E! 広域センサネットワーク・層アーキテクチャ

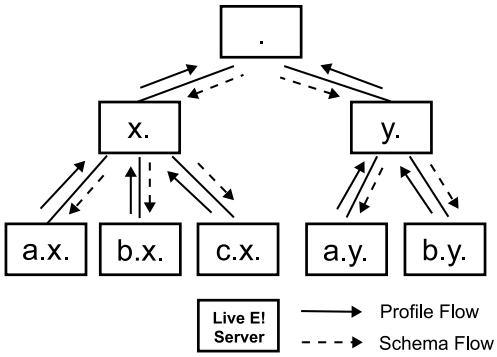

図 2.2.　階層構造による Live E! 運用ネットワーク

2.2 管理/API 層

2.2.1 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 1 つ管理し、ルートサーバから配信している。具体的には、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。図 2.3 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。

2.2.2 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）

```
<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 2.3.　プロファイルスキーマの例

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ため、管理層で運用者によりアカウントを作成する。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現実的である。

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。この要求を発行する組織名（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。アクセスコントロール設定は、データ転送層の Reference Monitor [162] 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。

2.2.3 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の言語表記を、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 2.4）。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。

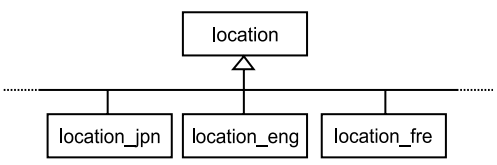

図 2.4. 多言語の管理モデル

2.2.4 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 節で示すものである。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（i.e., 内部で動いている search メソッドを利用して）定義され、ユーザアプリケーションへ提示される。

2.3 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、センサ広域検索を実現するし、2 種類の検索スキーム（i.e., 組織名検索、多属性検索）を備える。一般的なクエリのルーティング手法を記述し、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。さらに、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解説する。

2.3.1 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、本プロジェクトでは、検索クエリを Iterative 方式で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。

$$A_n = R(A_{n-1})Q \qquad A_0 = \{s_0\} \qquad \cdots\text{数式 } 1$$

ここで、R は経路表の全体を表す。$R(A_{n-1})$ は $R(s)$, $\forall s \in A_{n-1}$ を意味し、サーバノード s での経路表を表現している。Q はユーザから提示されたクエリである。

2.3.2 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 [117] 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、検索対象が単一であれば、$O(d)$ 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 には、子ノードとして、<jp.> <tw.> <th.> があり、検索クエリ Q は <x.y.z.tw.> などと与えられる。ここで、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されている。すなわち、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.> $\subset$ <tw.> と表現し、<tw.> $\bullet$ <x.y.z.tw.> $= 1$ と定義する。一般に、

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

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、

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

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

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

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 2.5 のように表記する。

domain は、アプリケーションの領域を示すもので、本プロジェクトでは常に 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。

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

図 2.5.　組織名検索クエリの例

2.3.3 多属性検索

多属性検索は、プロファイル情報を基にしてさまざま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。本プロジェクトでは、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 2.6 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、ローカルサーバに登録してある。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 2.7 に示す。

app 属性は、多属性検索で用いる属性および条件を示す。図 2.7 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。

本プロジェクトにおいて、R はクエリ経路表と呼んでいるが、分散インデックス [27] 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、

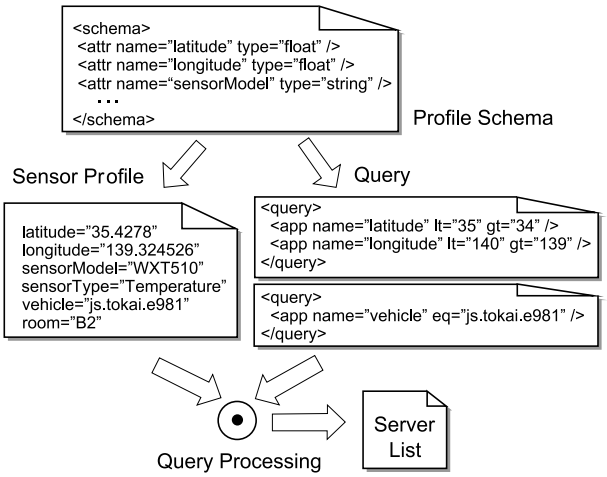

図 2.6. スキーマ、プロファイル、クエリ、検索プロセスの関係

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

図 2.7. 多属性検索、クエリ表記

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。本プロジェクトでは検索を、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）。

・クエリ経路表の作成　図 2.8 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 2.8 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

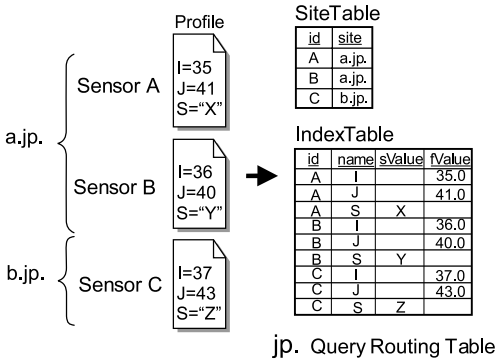

図 2.8. クエリ経路表の構築

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。

・クエリ転送処理　いま <jp.> に対して、図 2.9 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 2.9 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 2.9 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。

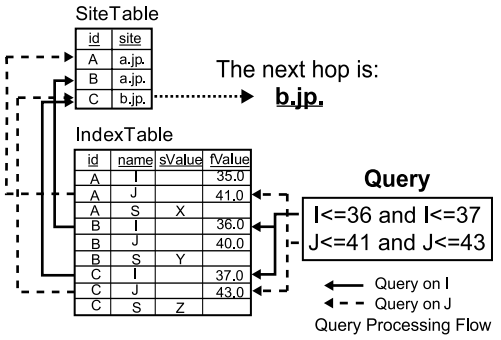

図 2.9.　クエリの転送処理

2.3.4 複合検索

本プロジェクト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 2.10）。クエリ表記で記述すれば、図 2.11 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。2.3.2 項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）、スケール性を持たせる事が可能である。

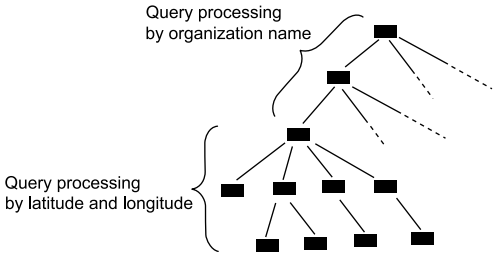

図 2.10.　複合検索

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

図 2.11.　複合検索クエリの例

2.3.5 Iterative v.s. Recursive

検索の実装 [163] 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推測されるためである。

2.3.6 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 節で述べる。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（Live E! 運用経験による）。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、同一クエリの発行確率が高いもののみをキャッシュできるようにしている。

2.4 データ管理/転送層

2.4.1 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、本プロジェクトでは、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関心があり、頻繁に最新値のデータのみ読み出す
- 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、これらの管理を行っている。

2.4.2 データ転送層

データ転送は、あるサーバの管理層で扱われているデータのコピーを、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 2.12 に示すように、データ提供サービスのすぐ裏に、Reference Monitor[162] を設置し、提供先サーバの認証を行うようにすれば、データ提供先をアクセスコントロール可能である。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時に、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。

複数のサーバから取得されたデータをマージし、これを結果としてユーザに応答する。この応答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。

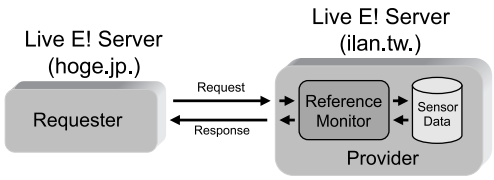

図 2.12. Reference Monitor によるデータ提供先のアクセスコントロール

2.5 センサ層

センサ層は、インターネットを介した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、センサの認証もともなう。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返す。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。

2.6 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低下させたりすることは極力避けるべきである。本プロジェクト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、冗長化によりサービス率の向上を図る仕組みにしている（図 2.13）。

システムの停止のほか、ディスクなど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ほか、組織で管理するデータそのものも含まれる。Master、Slave 1、Slave 2 いずれか 1 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、問い合わ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。

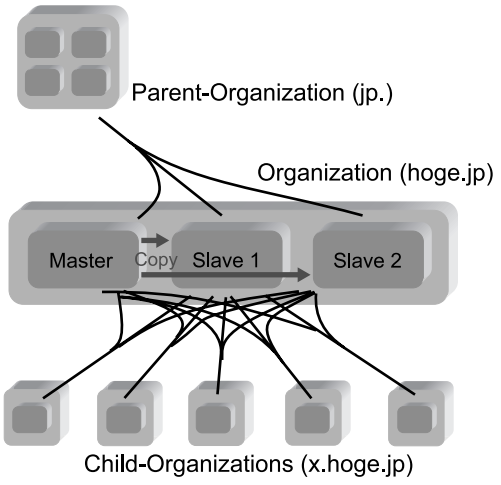

図 2.13. 冗長化による対故障性の向上

第 3 章　実装/運用状況

3.1 実装

図 3.1 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。SOAP エンジンの実装には、Axis1.4 を使用している。

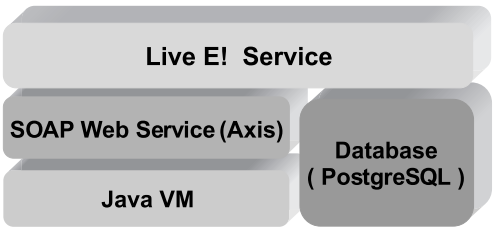

図 3.1.　Live E! サーバの実装構成

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ほか、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。このテーブルを作成する SQL コマンド列を図 3.2 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 3.3

```
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 3.3.　クエリ転送先を算出する SQL の例

```
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 3.2.　クエリ経路表のデータベースへの実装

```
<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 3.4. センサの表現

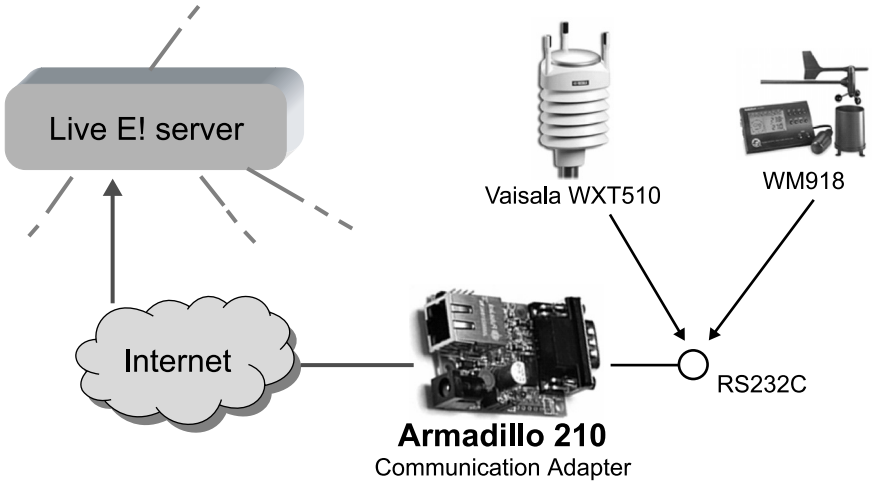

図 3.5. Live E! で使用しているインターネットセンサの構成

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。

Live E! では、センサを図 3.4 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ことも少なくない。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き込むという形態である。ここで value は複数あってもよい。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。

図 3.5 に現在の Live E! プロジェクト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 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。

3.2 運用状況

3.2.1 トポロジとネットワーク環境

2008 年 1 月現在、Live E! ネットワークの運

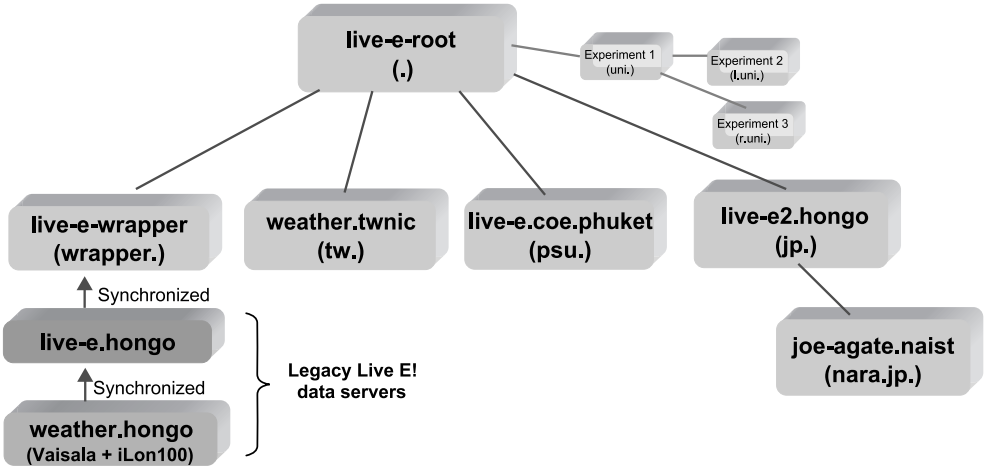

図 3.6.　2008 年 1 月現在の運用状況

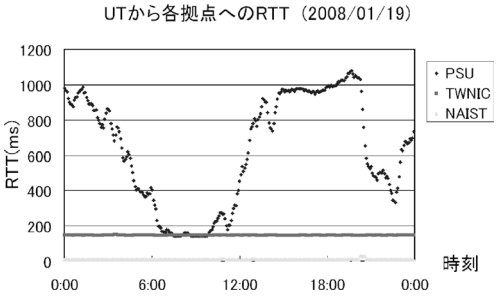

図 3.7.　UT から各拠点への RTT（2008 年 1 月 19 日）の結果

用構成は、図 3.6 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

```
<?xml version="1.0" encoding="UTF-8"?>
<profileShema xmlns="http://live-e.org/Shema/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=".*"/>
</profileShema>
```

図 3.8.　プロファイルスキーマ

設置されたサーバである。図 3.7 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 150 ms から 1100 ms の間を推移し、UT と TWNIC 間は 150 ms 程度、UT と NAIST 間は 8 ms 程度で安定している。

3.2.2 プロファイルスキーマ

図 3.8 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。

3.2.3 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 10 ms 以内で提供される。図 3.9 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、UT と PSU 間の RTT と正の相関を示している。

図 3.10 に、<jp.> で観測された「ユーザリクエストの処理時間分布」を示す。リクエストごとに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 200 ms〜400 ms 程度である。

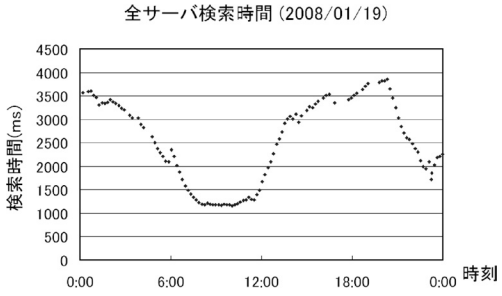

図 3.9. 全サーバ検索時間（2008 年 1 月 19 日の結果）

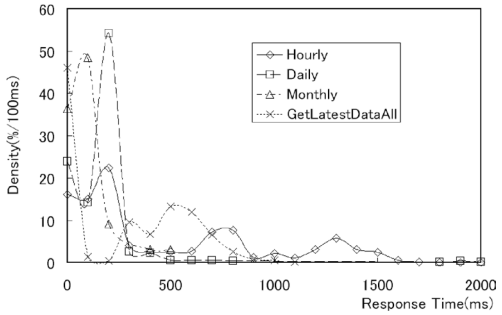

図 3.10. ユーザリクエスト処理時間分布（2007 年 11 月までのデータより）

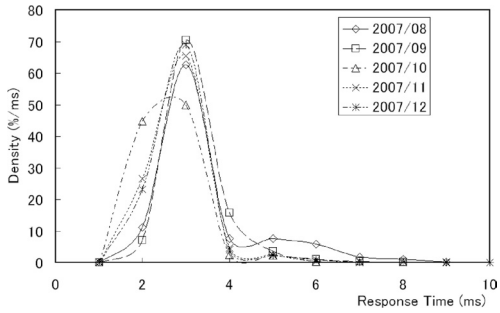

図 3.11. ルートサーバでのクエリ転送処理時間分布

図 3.11 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 2〜4 ms でほぼ完了している。各月の総クエリ数は下記のとおりであった。

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

第 4 章　今後の展開

国際展開にともなって Web や各種ドキュメントなどの英語化に注力した事もあり、海外でもセンサ設置が徐々に進みだした。その中でユーザビリティに関する要望も頻繁に聞かれるようになってきている。

基盤システムの改善・拡張およびセンサデータの信頼性確保や無線やモバイル技術との連携など多くの研究課題が残されている。その一方でユーザビリティ改善のために運用面での努力が急務である。来年度は研究が先行してユーザの視点を見過ごしてしまわぬよう、センサ登録の簡素化、各種 API の改善、メタデータの整備および自動付与などにも注力していきたい。