第II部

被災者支援安否情報登録検索システム

第2部
被災者支援安否情報登録検索システム

第1章　はじめに

WIDE プロジェクトでは、1995 年 1 月 17 日に起こった阪神淡路大震災を契機に、社会基盤としてのインターネットの役割、特に災害時のインターネットの役割を中心に検討を始めた。その活動を進めたのが Lifeline WG であり、社会情報基盤としてのインターネットを、技術的、運用的側面から見直し、その社会基盤としての可能性とそれを実現するための課題について研究を進めてきた。その研究の一貫として、IAA システムの研究開発に取り組んできた。

IAA システムは、被災者の安否情報をインターネット上の複数の拠点で収集・蓄積し、その情報の検索サービスを提供するためのシステムである。1996 年 1 月の第 1 回インターネット災害訓練では、訓練メニューの一つとして IAA システムを用いた安否確認訓練を実施し、2001 年までの計 6 回訓練を実施した。

ただし、Lifeline WG の研究対象は IAA システムだけではなく、社会基盤としてのインターネットの役割など広い範囲にわたっている。そこで課題を切り分け、目標を絞って研究を進める体制を作るため、Lifeline WG は 2001 年 3 月で発展的解消となった。そして、IAA システムに研究開発の対象を絞った本 WG (IAA-DEV WG) を立ち上げた。

本 WG では、現状の IAA システムを強化し、より堅牢なものとすることを目標に、議論と開発を進める。

1.1 研究活動の方針

IAA システムを災害時に利用されるアプリケーションととらえ、以下の 2 つの層に分けて考える。

上位層 ユーザが利用するアプリケーションとしての部分
- ユーザインターフェイス
- 事前登録モデル
- コミュニケーションモデル
- 社会システムとしての側面

下位層 災害時の状況に於てもロバストな情報伝達・転送、ネットワーク回りの仕組みとしての部分
- トランスポート
- リプリケーション機能

本報告では、「下位層」の新たな実装について述べる。

第2章　今年のアクティビティ

2.1 KANI の実装

本年度は IAA システムのうちでも下層の部分の研究開発に注力してきた。そこで本章では、本年度に実装した KANI システムについての設計と実装について述べる。

IAA98 システムのトランスポート部として NetNews システムを利用してきた。しかしこのトランスポートシステムとしての NetNews システムには、以下に示すような問題点があった。

- データ配送網 (feed link) を動的に変更できない。
- 様々なソフトウェア群から構成されており、設定が複雑かつ分散していて、管理・運用が煩雑である。
- ソフトウェアが巨大で、これ自体に変更を加えたり機能を追加するハードルが高い。

そこでこれらの問題点を解決し、かつ新たな機能を追加したトランスポートシステムとして KANI を開発した。

2.1.1 背景

本システムは、災害時に運用されることを前提としたシステムである。したがって、ネットワークの状態は激しく変化し、また分断などが断続的あるいは復旧不能なほどにおこるという前提を置かなけれ

ばならない。

また、災害時のような混乱した状態でも運用が行えるようにすることを考えると、その設定方法などは複雑であることは許容出来ないといえる。さらに運用をコンピュータの操作に詳しくないボランティアの人々のサポートを期待する場合にはなおさらである。

また、被災者情報という個人のプライバシに関する情報を扱うため、その扱いには慎重でなければならない。

以上のような背景より、次節に述べる設計ポリシに基づいて開発を行った。

2.1.2 設計ポリシ

KANI の設計ポリシを以下に列挙する。

- 頑健性に優れていること。
- 耐規模性に優れていること。
- できるだけ汎用的であること。
- 災害時の様な緊急時の運用に柔軟であること。
- 誰にでも使えること。
- セキュアであること。
- 移植性が高いこと。

KANI は IAA システムの一部として設計されてきた経緯から、災害時のようにネットワークの状態が不安定でも出来る限りデータの配送に努力するように設計する。

また、KANI は分散ソフトウェア環境を提供し、IAA システムはその上で動く、一アプリケーションであるように分離して設計されており、KANI の提供する分散網を利用する全く別のアプリケーションも容易に構築できるようにする。

KANI は、分散網へのクラスタの追加投入も視野にいれた動的な網制御機能を持っており、配送網自体もネットワークの状況に応じて最適と思われる経路でデータ配送を行う。

複雑な設定を行わずに運用できるような設計を心掛け、設定ファイルはひとつのディレクトリに集め、簡便な記述フォーマットを心掛ける。

以上のようなことを念頭に KANI を設計した [180]。

2.1.3 実装について

今回は、実装のしやすさやポータビリティ、ライブラリの整備などの理由から、Java により実装した。

また、KANI 上でのアプリケーションの部分機能を共有できる、コンポーネントモデルを導入した。これは、KANI 上にコンポーネントと呼ばれるソフトウェアモジュールを組みたてていき、これを組み合わせることでひとつのアプリケーションを構成する、ビルディングブロック方式のモデルである。

KANI はこのコンポーネントを管理するコンポーネントマネージャと、データ配送管理を行うメッセージングマネージャから構成されており、この上でコンポーネントが動くことになる。

これを図に表したのが、図 2.1 である。

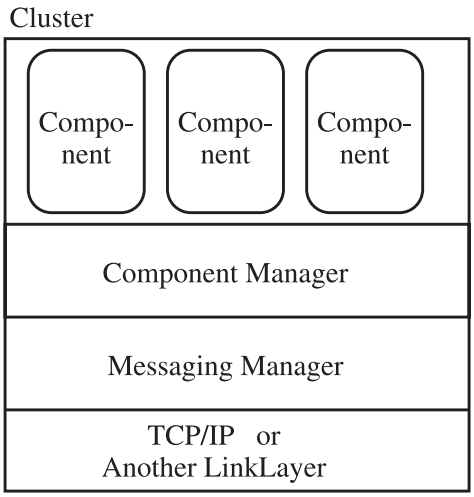

図 2.1.　コンポーネントモデル

コンポーネントは、サービスの単位毎に分離して作成することで再利用性を考慮し、様々なアプリケーションに再利用されることが可能である。アプリケーションはこれらのコンポーネントの集合であるといえる。

これを図に表したのが、図 2.2 である。

この図では、アプリケーション α はコンポーネント A および C から構成されており、アプリケーション β はコンポーネント B および C から構成されている。

次にコンポーネントマネージャについて説明する。コンポーネントマネージャは、コンポーネントを統合するのが仕事であり、以下の役割をになっている。

- ローカルコンポーネントの実行主体
- ローカルコンポーネント間の通信のサポート
- リモートコンポーネント呼び出しのサポート

コンポーネントマネージャが仲介する全てのコン

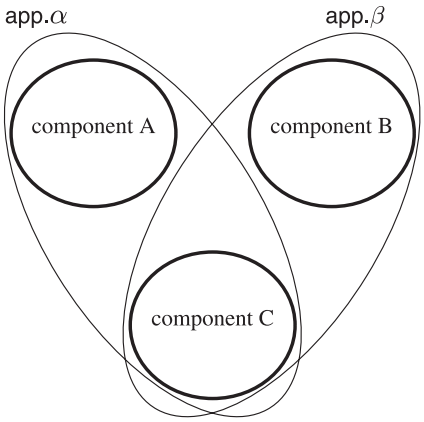

図 2.2. コンポーネントとアプリケーションの関係

ポーネント間の通信は、メッセージのやりとりとして抽象化されており、ローカルにあるコンポーネントとの通信とリモートにあるコンポーネントとの通信は、その違いを意識すること無く利用できるようになっている。

次にメッセージングマネージャについて説明する。メッセージングマネージャは、クラスタ間の通信を担うのが仕事であり、コンポーネントマネージャから見た場合にはコンポーネントのひとつである。

メッセージングマネージャは、IP アドレスなどのレイヤ 3 情報を隠蔽し、コンポーネントマネージャにはクラスタ内部での識別子であるノード ID を情報として渡している。これによって、レイヤ 3 が IP 網以外である場合でもメッセージングマネージャの変更のみで対応できるようになっている。

メッセージングマネージャはデータの配送に関して、以下のようなプライオリティで設計されている。

1. 可能な限りデータを届ける努力をする
2. 可能な限り早く伝達させる
3. 可能な限りトラフィックを減らす

つまり伝達速度よりは、確実にデータの配送が行われることが優先する。

次にメッセージングマネージャのデータ配送方法について説明する。

メッセージングマネージャは、各クラスタとフルメッシュの配送網を構築する。各クラスタへの出力キューにはプライオリティ付きの出力キューを持たせてあり、プライオリティが高いほど、頻繁にデータの送信を試みる。このプライオリティは自動的に変更されるようになっている。クラスタ間のデータのやりとりは、最初にデータの ID を相手に持っているかどうかを問い合わせる、いわゆる Ihave/Sendme 方式をもちいている。プライオリティは、相手がすでにデータを持っている場合には、プライオリティは下がるように、相手が持っていない場合はプライオリティがあがるようになっている。プライオリティが低くても必ずデータの送信は試みるようになっており、どこかひとつのクラスタと接続性が確保されていればデータは届くようになっている。

これにより、できるだけ確実に早く、さらにトラフィックをできるだけ抑えたデータ配送を行っている。

現在の KANI をつかった IAA システムは図 2.3 のような構成になっている。

図でみるとわかるように IAA システムは、LWLLDB, PgServer, MIAADecoder という 3 つのコンポーネントから構成されている。以下に、そ

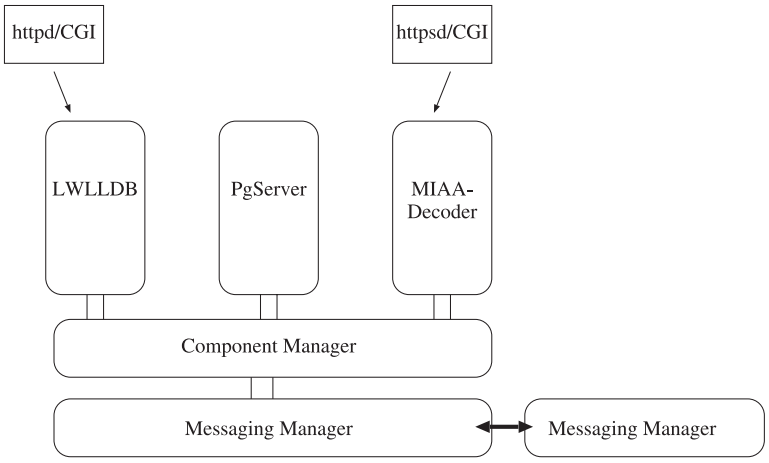

図 2.3. KANI 上に構築された IAA システム

れぞれのコンポーネントの役割を示す。

LWLLDB

LightWeightLLDB プロトコルによる検索、登録要求を処理するコンポーネント。データのパースや正規化処理もここで行っている。

PgServer

PostgreSQL のフロントエンドエンジン。Java のオブジェクトを引数にもった登録要求メッセージや、検索要求メッセージに対する処理を行う。

MIAADecoder

次章で説明する、他組織の IAA システムとのデータ交換を行うためのコンポーネントで、他組織の IAA システムから XML 形式にて送られてきたデータを処理し、被災者情報データとして登録するためのコンポーネント。

2.1.4 従来の IAA と比較した場合の優位点

IAA98 と KANI 上に構築した IAA システムとの比較を以下に示す。

配送網のうち動的な評価ルーチンによって優先配送を行うようになった。これにより、網の現状に即した配送経路を選択し、かつバックアップの配送もおこなえるようになり、より配送に関する堅牢性が増したといえる。

また、データの配送においては SSL を用いて配送するようにし、従来の DES 暗号方式を撤廃。クラスタ毎に鍵を持たせ、その鍵に基づいた公開鍵暗号方式を用いたデータの暗号化をおこなった。これにより、全てのクラスタでひとつの秘密鍵を共有する必要がなくなった。また、Java のシリアライザブルデータのストリーミング機能をもちいることで、アプリケーションのもつデータ構造を保持したままデータ転送できるようになった。これにより KANI はより汎用なデータ配送メカニズムとして利用できることを意味している。また、実装言語を従来の Perl から Java に移行することで、セキュリティ面での配慮や開発行程の短縮、より高いポータビリティを確保した。従来の Perl 版は、Perl モジュールに高く依存しており、各種モジュールのバージョンによっては、動作に不具合が生じることがあった。

またコンポーネントモデルを採用し、アプリケーションと配送部分を分離することで、IAA 以外のアプリケーションの開発が容易になった。

2.2 フリーの実装とそのリリース

今後、この KANI ベースの IAA システムを WIDE プロジェクト内部で公開し、そのフィードバックによって改良を行っていく予定である。また、その後には、フリーソフトウェアとしてリリースしていく予定でいる。

第 3 章　他のシステムとの連携インタフェースの提案

IAA システム以外の被災者情報データベースがいろいろと開発されてきた昨今、これらのデータベース間でのデータ交換の方法について議論がなされている。現在でもこの議論は継続中であるが、今までに提案された事柄をここに述べる。

3.1 データ交換ポリシ

様々な IAA システムとデータの交換を行うためのプロトコルは、現在まだ規定されていない。LLDB プロトコルは、あくまでも WIDE IAA システムとそのクライアントとの通信プロトコルであり、様々な IAA システムとの間での取決めは、なされていない。また、その取決めを策定する際に基本となるデータ交換のポリシが必要となる。以下に、我々が提案する基本ポリシを挙げる。

- 送信元ではデータを送信することのみに責任を負い、データ送信正常終了後は送信先システムに送信したデータの扱いに関する権限を委譲する。
- 受信側ではデータの表示項目、キー項目を含め自由に扱うことができる。
- データ交換の際は SSL などのセキュア通信を必須とする。

このポリシはデータを送信したらその先での扱いは制御できないというスタンスに立ったものである。逆にいうならば扱いを制御したい情報は相手に送ってはいけないということである。そこで、現在は表 3.1 にあるような被災者を特定するに必要な最小限のデータのみを交換し、データ送信先におけるユーザには、オリジナルデータへのポインタを適宜提供することを提案する。

表 3.1. 交換するデータ項目

項目名				意味	必/任	フォーマット
iaa_data	registered_time			登録日時	必須	UTC(10 進数) の文字列
	disaster_id			災害 ID	必須	文字列
	personal_id			個人 ID	必須	文字列
	name	iname		国際名	必須	文字列 (ASCII 文字列)
		localname	lyomi	姓の読み	任意	文字列 (カタカナ)
			fyomi	名の読み	任意	文字列 (カタカナ)
			lname	姓の漢字	任意	文字列
			fname	名の漢字	任意	文字列
	state			状況	必須	数字 (1 桁)
	additional_info			備考	任意	文字列
	iaa_id			IAA_ID	必須	数字
	reference			リファレンス	必須	数字
	system_info			システム情報	必須	数字

3.2 データフォーマット

交換するデータフォーマットも規定しなくてはならない。我々は広く一般的にデータ記述言語として用いられている XML による記述を検討している。図 3.1 にしめすような DTD によって被災者情報を記述する方法が提案されている。

実際の XML データは図 3.2 のようになる。

このようなデータを SSL のようなセキュアなチャネルを通してやりとりすることで、別組織間での被災者情報の流通を行うことを検討中である。

3.3 ID 付け

世界中でおこる災害の被災者を特定する際に、なんらかの ID を付与し、それを使った情報流通を行うことで、効率的な処理や、プライバシの保護に役立つと考え、現在 IAA-ID とよばれるいくつかの ID を検討している。以下に検討中の ID について述べる。

3.3.1 設計ポリシ

ID の設計に際しては、以下の点に留意する。

- ID の中にどのような情報を持たせるべきか
- ID を構成する名前空間をどうわけるべきか
- ID の運用方法

現在、災害 ID, 個人 ID, IAA-ID の 3 つを検討している。災害 ID は、ある地域に起った災害を指し示す。災害が起った地域と時期がわかるようにする。個人 ID は、ある被災者情報を特定する。しかし、ID だけで個人のプライバシに関する情報がわからないようにする。また ID を生成したものがわかるようにする。IAA-ID は IAA システムを指し示す。

ID を構成する名前空間に関しては、階層的に扱えるようにし、下位組織に権限を委譲できるようにする。つまり全体の一元管理は行わない。また、最終的にユニークになるようにしなければならないが、その方法までは定義しない。ただし、ID 全体の大きさ (長さ) に制限を設ける必要はある。

また、ID の運用方法については、ID を構成する名前空間の階層毎に役割を定め、独立して運用できるようにする。

3.3.2 ID の設計

災害 ID と個人 ID を URN であらわす。URN は、情報を一意に特定するための仕組みであるのと同時に、階層的に扱うことが容易な仕組みである。したがって、上記ポリシの実現にふさわしいものと考えられる。

名前空間識別子として「iaa」を使った例を示す。正式には IANA による承認と登録が必要である。

災害 ID は、「iaa」の次の階層の識別子として「d-id」を災害 ID をあらわすものとして使い、その次の階層に地域をあらわす識別子として ccTLD を用いる。最後に年月をあらわす 6 桁の数字と「-」で繋いだ後にその年月で一意になる文字列を配置する。

```
<!ELEMENT iaa (iaa_data+)>
<!ELEMENT iaa_data (
     registered_time,
     disaster_id,
     personal_id,
     name,
     state,
     additional_info?,
     iaa_id,
     reference,
     system_info
     )>
<!ELEMENT registered_time   (#PCDATA)>
<!ELEMENT disaster_id   (#PCDATA)>
<!ELEMENT personal_id   (#PCDATA)>
<![%basic; [
     <!ELEMENT name (iname)>
]]>
<![%include_jp;[
     <!ELEMENT name (iname, localname?)>
     <!ELEMENT localname (lyomi?, fyomi?, lname?, fname?)>
     <!ELEMENT lyomi    (#PCDATA)>
     <!ELEMENT fyomi    (#PCDATA)>
     <!ELEMENT lname    (#PCDATA)>
     <!ELEMENT fname    (#PCDATA)>
     <!ATTLIST localname locale CDATA #FIXED "JP">
]]>
<!ELEMENT iname (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT additional_info  (#PCDATA)>
<!ELEMENT iaa_id (#PCDATA)>
<!ELEMENT referrence (#PCDATA)>
<!ELEMENT system_info (#PCDATA)>
<!ATTLIST iaa version NMTOKEN #REQUIRED>
```

図 3.1. DTD

フォーマットは以下のようになる。

urn:iaa:d-id:ccTLD:YYYYMM-xxxx

たとえば、2002 年 5 月の日本における災害は、以下のように表現できる。

urn:iaa:d-id:jp:200205-1

個人 ID は、「iaa」の次の階層の識別子として「p-id」を個人 ID をあらわすものとして使い、その次の階層に地域をあらわす識別子として ccTLD を用いる。次の階層には、この個人 ID を生成したベンダをあらわす識別子 (VENDER-ID) を配置する。最後にそのベンダによってユニークにつけられた文字列が続く。

フォーマットは以下のようになる。

urn:iaa:p-id:ccTLD:VENDER-ID:xxxx

例.

urn:iaa:p-id:jp:MoonMicrosystems:1234567890

```
<?XML version="1.0" ?>
<!DOCTYPE iaa SYSTEM "dtd.txt" [
<!ENTRY % basic "IGNORE">
<!ENTRY % include_jp "INCLUDE">
]>
<iaa>
    <iaa_data>
        <registered_time>9999999999</registered_time>
        <disaster_id>urn:iaa:d-id:jp:200205-1</disaster_id>
        <personal_id>urn:iaa:p-id:jp:MoonMicrosystems:A12345678</personal_id>
        <name>
            <iname>Taro Saigai</iname>
            <localname locale="JP">
                <lyomi>ヤマダ</lyomi>
                <fyomi>タロウ</fyomi>
                <lname>山田</lname>
                <fname>太郎</fname>
            </localname>
        </name>
        <state>1</state>
        <additional_info>とくになし</additional_info>
        <iaa_id>iaa://moon.iaa.jp/urn:iaa:d-id:200205-1/urn:iaa:p-id:jp:
Moonmicrosystems:A12345678</iaa_id>
        <reference>http://moon.iaa.jp/search.cgi<referece>
        <system_info>IAA-System Version2002</system_info>
    <iaa_data>
</iaa>
```

図 3.2. 被災者情報の XML サンプル

IAA-ID に関しては現在検討中である。

今後、事前登録モデルも視野にいれながら、ID の構造とその運用に関して検討をすすめていく。また、IAA システムを運用している組織間レベルでの連絡の方法や手順などについても IAA アライアンスなどと協力しながら確立していこうと考えている。

第 4 章　今後の予定

4.1 フリー版 IAA システム (IAA98) のリリース

2003 年 3 月に、現在の IAA システムの実装である IAA98 をフリーソフトウェアとしてリリースする。

IAA システムへのインタフェースである LLDB プロトコルはすでに公開されている。IAA98 は、この LLDB プロトコルのリファレンス実装として公開し、同時に IAA98 ベースの開発を終了する。

4.2 他のシステムとの連携インタフェースの仕様公開

類似の IAA システム間でデータを交換する方法については、今後も本 WG での議論を進めていく。特に交換するデータについては、メタデータ技術の活用も検討していく予定である。

また、議論の結果として連携インタフェースの仕様を作成および公開し、類似システムを開発している関係者とも議論を進めていく。

4.3 KANI ベース IAA システムのフリー版の開発

KANI ベースの IAA システムは、WIDE プロジェクト内部での公開を経て、フリーソフトウェアとしても公開していく。KANI の実装のみでなく、先に公開する予定の「他のシステムとの連携インタフェースの仕様」に基づく実装も含む予定である。