社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－AVM－62（5）
2008/9/25

ビジュアルセンサネットワークのためのリソース指向アーキテクチャ

岩谷 周† 中塚 正之† 高柳 雄太郎† 甲藤 二郎†

†早稲田大学大学院基幹理工学研究科 〒169-0072 東京都新宿区大久保 3-4-1 55N-06-09

E-mail: †{iwatani, nakatsuka, takayanagi, katto}@katto.comm.waseda.ac.jp

あらまし 近年活発に研究が行われているセンサネットワークであるが、ここ数年カメラやオーディオなどマルチメディアセンサーを用いたネットワークの構築が行われている．このように多種のセンサを組み合わせることでより正確かつ詳細な情報が環境から得られるが、現状まだ限定的な事例での利用が中心である．そこでセンサネットワークから得られるセンサデータ値を Web サービスとして提供し、それらを広く Web の世界へ提供することを提案する．また Web サービス化手法としてリソース指向アーキテクチャでの構築を行い、様々なデータ形式をリソースという統一された形式での提供を目指す．

キーワード センサネットワーク，リソース指向アーキテクチャ，センサフュージョン，Web サービス

Resource Oriented Architecture for Visual Sensor Networks

Hiroshi IWATANI† Masayuki NAKATSUKA† Yutaro TAKAYANAGI† and Jiro KATTO†

† Graduate School of Fundamental Science and Engineering, Waseda University 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-0072 Japan

E-mail: †{iwatani, nakatsuka, takayanagi, katto}@katto.comm.waseda.ac.jp

Abstract Sensor Network has been a hot research topic for the past decade and has moved its' phase into using multimedia sensors such as cameras and microphones. Combining many types of sensor data will lead to more accurate and precise information of the environment. However the use of the sensor network data is still limited to certain circumstances. Thus we propose the deployment of sensor data as web services. In order to unify different types of sensor data and also to support heterogeneous client applications, we used Resource Oriented Architecture.

Keyword Sensor Networks, Resource Oriented Architecture, Sensor Fusion, Web Service

1. はじめに

1.1. 研究背景

21 世紀になりユビキタスネットワーク社会の実現に向けてセンサネットワークを利用したシステムやアプリケーションの研究が盛んに行われてきた．これまでのセンサは主に簡単な数値を計測するもの(i.e.光量，音量，温度，湿度など)が中心として用いられていたがセンサ技術の進化，低消費電力化，安価化，が進むとともにマルチメディアセンサ(カメラやマイク)を利用したセンサもセンサネットワークのセンサとして研究されるようになった．このように様々な種類のセンサがセンサネットワークに導入されてはいるものの限定的な事例での利用例が多く，まだ一般に普及することが見られない．

センサネットワークの研究の多くは構築するセンサネットワーク間のローカルの配置・通信方法・意味のあるデータの抽出など，個々のセンサネットワークで克服するべき課題に挑戦することが盛んである．しかしそれらはその構築されたネットワーク内での事例にとどまってしまう.そこで本研究ではセンサデータを Web サービス化することにより，様々なクライアントに提供することを実現する．Web サービス化手法としてはリソース指向アーキテクチャ(以下 ROA)を用い，異種のセンサから得られる様々なデータ形式の統一を行っていく．次の節では関連研究について述べ，2 章で提案内容について説明する．そして 3 章では実装とその実装結果についての考察，最後 4 章でむすぶ．

1.2. 関連研究

センサーネットワークは、物理層からアプリケーション開発に至るまで、世界中できわめて多種多様な研究開発が行われている。著者らもまたセンサネットワークを利用した具体的なアプリケーションの例として[3]のように電波強度を利用した人の混雑度推定やカメラセンサを利用した人物の位置推定[4]の検討を進めている．これらは別種のセンサであるがもしこれらを複合的に利用すればどちらの研究においてもさらに精度を上げることが可能となるであろう．このように限定的な事例でのセンサネットワークの利用においても，さらに複数種のデータがあるに越したことはなく，またせっかく取得したセンサデータをローカル内の推定のみに使用するだけではもったいない．センサから得られたデータは広く世の中に公開されるべきであり，それによって初めていつでも，どこでもというキーワードを満たすユビキタス社会の実現に近づくことが出来る．

また ROA では公開するリソースに URI を割り当てるのであるが，こちらはセンサデータ毎に個別のものを割り振ることとなり，UbiquitousID[6]のような ID の割付と関連付けられる．UbiquitousID では ucode という独自の ID をありとあらゆるモノに割り振り，それらの情報や関係などをまとめ・管理することを目的としている．そしてモノだけではなく空間などにも ucode を割り振ることで，RFID 対応端末や無線通信端末を持ったユーザがその空間に近づくだけで，関連する情報が自動で取得できるシステムに利用することが可能となっている．センサネットワークから得られる情報もこれに関連づけられるが，しかし ucode は主に企業や団体などが取得し，利用することが考えられているので，本研究の目的(センサネットワークから得られる情報を広く様々な利用者から利用可能とすること)とは異なる．

2. 提案内容

2.1. リソース指向アーキテクチャ

従来 Web サービス化手法では REST(REpresentational State Transfer)対 SOAP(Simple Object Access Protocol)という言葉での論争があった[5]．しかしこれらの用語はそもそもアーキテクチャではないため，言葉のみが一人歩きし混乱が生じていた．REST とはそもそも Roy Fielding が論文[2]でアーキテクチャを判断するための指標として定義した言葉である．この REST という指標で高く評価される事を目的として確立されたアーキテクチャがリソース指向アーキテクチャであり，こちらは[1]の作者である Richardson によって明確なアーキテクチャとしてまとめられている．この節では ROA について簡単な説明を行う．詳細な説明は[1]を参照するとよい．

まず SOAP というテクノロジを利用したリモートプロシージャコール(RPC)や，REST という言葉が一人歩きし、中途半端に取り入れられて生まれてしまった REST－RPC ハイブリッドなどから説明する．

~RPC~

RPC での Web サービス化ではプロシージャというように関数を公開することとなる．例えばあるブログサイトにおいてブログエントリの中からクエリでの検索を行ったり，指定されたブログの作成や削除を行うサービスを構築することを考える.この場合 get_blogentry(var)などといった関数を公開することとなる．削除する場合は delete_blogentry()，作成する場合 post_blogentry()などとなるであろう．このように処理毎にサービスを公開しなければならない点と，またブログサイトは数多くあり，それぞれが独自の関数を提供した場合(getblog(var)，get_blog(var)，get_article(var1，var2)...)その数は膨大となる．特定のサイトが提供するサービスのみを利用したい場合はそれほど手間にならないと思われるが，例えば様々なブログサイトの検索サービスを利用したまとめサイトを作成するとなると，ひとつひとつの仕様を確認しテストする事となる．

~REST-RPC ハイブリッド~

上記のような手間を軽減する手法，そして REST という言葉が部分的に理解された結果生まれてしまったものが REST－RPC ハイブリッドである．こちらは Web サービスと言うにふさわしく，新たなサービスを公開するときは URI を提供することとなる．先ほどの例ならば http://example.com/get_blogentry や http://example.com/delete_blogentry となる．引数は URI に続けて?q1=A&q2=B…と続けることで渡し、HTTP 通信の GET メソッドを選択することでリクエストを送信する．レスポンスは主に XML 形式で格納された結果が得られることとなる．これによりブラウザからの確認も可能であり，前述のブログのまとめサイトの作成の手間も軽減する.しかしながらこの手法では問題点がある．それは HTTP 通信にそもそもメソッドがいくつか用意されているという点に起因する．代表的なものは GET と POST であるがその他 PUT や DELETE もある．このハイブリッド手法では例えブログを削除するときも http://example.com/delete_blog?…に GET リクエストを送ることとなる．これは明らかな矛盾である．この矛盾をなくすために ROA が確立された．

~ROA~

まず公開するものは URI という点で REST-RPC ハイブリッドと同じであるが，その内容は主に名詞・形容詞で構成することとなる．例えばブログエントリなら

ば http://example.com/blog/2008_8_7 などである．そしてそれに対して HTTP の 4 つのメソッド(GET, POST, PUT, DELETE)に対応させる．これだけで ROA が成立する．もう少し詳細に説明すると，Web サービスとして公開するものは URI が割り振られた何かしらのデータ(例で言えば 8 月 7 日のブログ記事)でありそれに対して HTTP の 4 つのメソッドに操作を加えることが出来るということである．そしてこの何かしらのデータがリソースであり，Web サービスを公開する際にリソースを中心に考えるアーキテクチャが ROA となる．

さてこのアーキテクチャで前述の例を確認すると．まずブログを削除する場合は削除したい記事の URI (例えば http://example.com/blog/2008_8_7)に HTTP メソッド DELETE でリクエストを送信する．ハイブリッドと違い矛盾がない．また検索は http://example.com/blog?q1A=&q2=B…に対して GET メソッドのリクエストを送ればよい．大事なことは URI を見ることでリソースが識別できることと、処理に関しては HTTP メソッドに委ねる点である．

続いて HTTP の 4 つのメソッドを簡単に説明する．GET は取得，POST は新規作成，PUT が更新，DELETE が削除である．本来 HTTP には PUT という更新用のメソッドが用意されていたのであるが，ブラウザが GET と POST のみしか対応していないものがほとんどでありあまり使われて来なかった．この状況を憂いていた Roy Fielding が REST という概念を述べた[2]のであるが，Fielding 自身が HTTP の仕様策定に大きく関わっていたという背景がある．

2.2. センサデータと ROA

前節で ROA について簡単に説明したが，この ROA をセンサネットワークにおけるセンサデータに応用することを考えたい．ユビキタスネットワーク社会の実現に必要不可欠であるセンサ網であるが，このセンサから得られる情報というものを限定的な事例のみで用いることは勿体ないことであり，無駄が生じる．警備のための監視カメラを設置してある場所は多々あると思われるが，それらが広く一般に使用出来るよう公開されていればそれらを用いて人の数の把握などなんらかのアプリケーションに利用できるはずである．その為にＷｅｂサービス化を行うことは有用であり，センサネットワークを利用したアプリケーション作成を容易にするであろう．さらにセンサデータの提供ということは極めてシンプルな処理であり，まさにリソース指向に適している．ROA でサポートする 4 つのメソッド GET, POST, PUT, DELETE をセンサデータに適用することを考えた場合：

GET：センサデータの取得．クライアントアプリケーションがセンサデータ値を取得したい時に用いる．

POST：センサデータの新規作成．新たなセンサを環境中に配置したとき，そのセンサに URI を割り当て新規作成する．(つまりセンサデータを提供する人が行う)

PUT：センサデータの更新．POST で作成したリソースに PUT によるデータ更新を行い続けることでセンサデータ値をリアルタイムに提供することが出来る．

DELETE：センサデータの削除．設置したセンサを撤去した場合など，データの提供を停止するときにリソースを削除する．

上記のようになり，これで十分センサデータの Web サービスはその機能を満たしている．

続いてセンサデータをリソースとして提供する上で URI の命名規則を考えなくてはならない．ROA では名詞・形容詞で構成されることと，分かりやすい規則性を持たせることが重要である．この 2 点を考慮し，また今回センサという物理的なモノは必ずどこかの位置に設置されなければならないという点([1]の影響も強く受け)を考慮し，http://example.com/sensordatas/経度/緯度/センサタイプ　とした．センサタイプとは光量や音量，電波強度やビジュアルなどセンサの種類を示すこととする．例えば経度 139.77048055555556，緯度 35.67777222222222，センサタイプが光量だとすると http://exmaple.com/sensordatas/139.77048055555556/35.67777222222222/light となる．このようにすることで URI を見るだけで何を示しているのかわかるはずである．http://example.com/sensordatas/sensor_1 などとした場合では，すっきりしているが，URI だけではどこの何のデータかを判断することが出来ない．

ベースは上記のような命名規則であるが，例外もあるであろう．例えば GPS センサである．どこに設置してあるか分かっている GPS センサのセンサデータ値を取得したい人はいない．そのため GPS センサの場合は http://example.com/sensordatas/GPS/GPS_ID とする．GPS_ID に単純に数値を入れてしまうと URI を見ても何の GPS 情報であるか分からないので，GPS センサを付帯しているモノを表現する言葉が適している．人であれば名前，車であれば車体番号などが想定される．

その他に提供すべきリソースがある．これまでに述べたリソースでは予め用いたいセンサデータが決まっているアプリケーションでは有用であるが，ユーザの位置が変わる毎にその位置に一番近いセンサデータを扱いたい場合も想定される．むしろそのような場合のほうが多々考えられる．例えば近年当たり前のように

GPS 機能付き携帯電話が使用されているが，GPS＋現状の気候，によるお薦めレストラン情報提供サービスを作りたいとする．これは従来のレストラン検索サービスの検索結果を付近の状況(温度や湿度)に応じて表示順を変えるものをイメージしている．この場合ユーザの位置情報は GPS により取得できるが，その GPS 情報と一致するセンサデータが得られるとは限らない．むしろ一致しない場合がほとんどであると考えられる．そのような場合に備え，http://example.com/sensordatas/GPS から得た経度/GPS から得た緯度/temperature/nearest というリソースを用意しておく必要がある．http://example.com/sensordatas/search_nearest などのようにしてしまうと前節で述べた通り混乱のもととなるので気をつけなければならない．

さらに想定しなければならない点はセンサ自体が動いていることも考えられる．例えば遠隔操作ロボットにセンサを配備している研究も行われている．この場合そのロボットや動くもの自体に GPS センサが搭載さていることが前提となり，センサデータを更新するのと同様経度・緯度情報も更新すればよい．しかしながら，GPS センサの誤差を考慮するとそのリソースをピンポイントで使用したい場合見つけられない可能性が高い．そのため GPS センサ同様 http://example.com/sensordatas/dynamic/ID/センサタイプも提供することとする．ID には GPS 同様数値ではなくセンサが配備されているモノと関連する言葉とする．

3. 実装

3.1. 実装環境

実装に Ruby on Rails[7]を用いた．ネットワークアプリケーションを作成するためのフレームワークであるが，2007 年末に出た新バージョンではより Restful な設計(ROA に近い)が行いやすく変更された．また緯度・経度情報に必要不可欠な位置情報は Yahoo!地図[8]とのマッシュアップを図りユーザビリティを向上させた．データベースには MySQL[10]を用い，センサデータテーブルとしては ID，データ値，センサタイプ，緯度,経度,コメントという情報を格納することとした．またビジュアルセンサの提供の実現には Ustream[9]を用いた．これは Web カメラさえあれば誰でもリアルタイム映像を世界に配信できるサービスである．センサタイプをカメラとした場合今回はこのサービスで公開した動画のリンクをセンサデータ値として保持させる形で実現した．

実装結果例を図１～３に示す．図３のように http://localhost:3000/sensordatas/経度/緯度/light.xml という URI に GET リクエストを送ればセンサデータが取得できる．

センサデータの更新を行うプログラムも実装した．まず予め Web ブラウザからセンサを設置する位置と種類に対応するリソースを作成しておく．例えばセンサタイプが電波強度ならば http://localhost:3000/sensordatas/経度/緯度/rssi である．その上でセンサプログラムを作成し java の HTTP 通信で，先ほどの URI にセンサデータとともに PUT メソッドを送信することで実装を行った．なお用いたセンサは MOTE[11]の MICAz である．

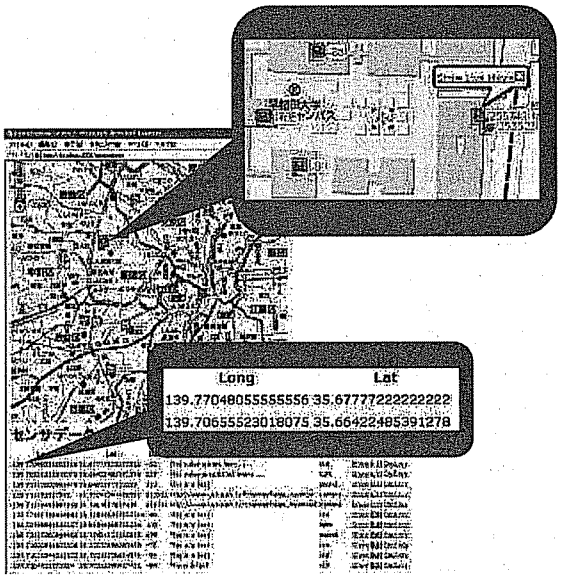

図１実装結果

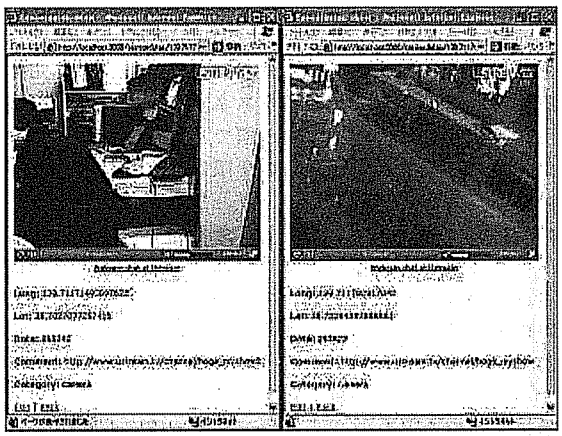

図２カメラセンサ

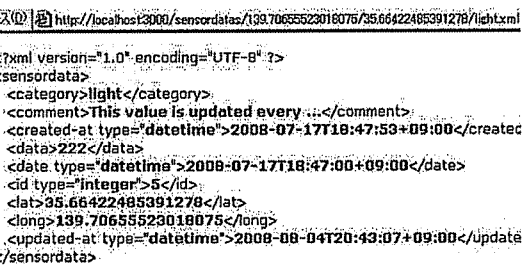

図 3 光量取得リクエストへのレスポンス

3.2. 考察

前節の実装を行った結果の考察事項をまとめる：

・センサデータの Web サービス化を ROA で行う事は広く様々なクライアントアプリケーションに自由にセンサデータ値を使用可能とする手段として以下のような有用点がある：

①分かりやすく，URI を見ただけでどこで何のデータか分かる．

②HTTP 通信さえ出来ればプログラミング言語や OS やハードに依存することなくデータを取得することが可能．

③Web ブラウザから値を確認できる．

・しかしながら考えなければいけない点もある：

①HTTP 通信はリクエストからレスポンスまで一定の時間を要する．遅延要求が厳しく，リアルタイムで更新し続けなければいけないようなデータを扱う場合は他の手段の検討も必要である．

②何千何万ものセンサを配備するようなネットワークの場合それらを全てリソースとして公開するのは大変な手間となる．こちらはセンサノード群である程度結果をまとめた上で公開することが必要である．

③位置によるリソースの識別を行ったが，緯度経度だけでは情報が足りない状況がある．例えば建物の各階に設置されたものの場合がある．この場合 URI に高さ情報を含めることで解決する．その他に多くのセンサ値はそのセンサが設置された位置を中心として一定の半径内の環境に関する情報を取得するが，例えば電波強度などは 2 つのノード間のデータであり，またカメラなどは設置した方向のみのデータを取得する．こういった情報もうまく組み込むことが必要である．

上記に加えセキュリティ面であるが，こちらは ROA での基本的な手法同様となる．ROA の 4 つのメソッドでは GET に制限をかける必要はない．しかしながら残りの三つは余計な情報や嘘の情報が載せられないよう規制をかける必要がある．こちらもユーザ名ないしはコミュニティによるセンサデータ値を提供する人の認証を行った上での POST, PUT, DELETE に対応する形式を構築しなければならない．

4. むすび

多種センサデータのリソース指向アーキテクチャによる Web サービス化の提案と実装を行った．ROA に基づいた URI の組み立て，HTTP の 4 メソッドへの対応，地図サービスとのマッシュアップなどを行い，その有用性を確かめた．様々な種類のセンサデータを統一されたリソースとして提供することにより異種類のデータを統一したデータ形式での提供が実現した．検討事項も多々あるが，このように Web サービスとして公開すれば，これまでセンサネットワークに関わって来なかった人でも関わることが可能となり様々な新しいアイディア・アプリケーションが生まれてくることが期待される．

文　献

[1] L. Richardson, S. Ruby. “RESTful Web Services.” O’Reilly, 2007.

[2] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* Ph.D. thesis, Dept. of Information and Computer Science, University of California at Irvine, 2000.

[3] M. Nakatsuka, J. Katto: A Study on Passive Crowd Density Estimation, ICMU2008 Tokyo, JAPAN

[4] 高柳雄太郎，甲藤二郎："人物のリアルタイム三次元位置推定実験に関する一検討" 信学技報，Vol.108, No.127, pp.73-76, Jul.2008.

[5] Michael Muehlen. *Developing Web Services Choreograph Standards—the case of REST vs. SOAP.* Decision Support Systems 40, 2005.

[6] Ubiquitous ID Technologies
http://www.uidcenter.org

[7] Ruby on Rails
http://www.rubyonrails.org/

[8] Yahoo!地図
http://developer.yahoo.co.jp/map/

[9] Ustream
http://www.ustream.tv/

[10] MySQL
http://www.mysql.com/

[11] Crossbow
http://www.sensor-network.net/