

# SOXFire: XMPPに基づく都市センサ情報流通基盤

米澤 拓郎<sup>1,a)</sup> 伊藤 友隆<sup>1,b)</sup> 陳 寅<sup>1,c)</sup> 中澤 仁<sup>1,d)</sup>

**概要:** 本論文では都市センサ情報流通基盤 SOXFire を提案し、同基盤を用いこれまで藤沢市を中心として行ってきた都市情報収集・流通実験に関して述べる。動的に変化する都市状況に対応し、住民の安心安全や QoL を高め、効率的な都市運営を可能とするスマートシティでは、社会的に価値のあるセンサデータを膨大に収集し、必要とするサービスにオンタイムで届ける必要がある。本稿では都市センサ情報流通基盤として、XMPP を拡張した Sensor-Over-XMPP にもとづきセンサデータを流通可能とする SOXFire を提案する。XMPP を拡張することにより、SOXFire は膨大な都市センサ情報を、プライバシーを考慮したアクセスコントロールを行いながらスケーラブルに伝達可能なシステムとして設計された。我々は同基盤を用い、藤沢市の行政業務中に得られる様々な情報を効率よくかつ職員の負担なしで収集・流通する実験を2年以上に渡り行っている。本発表ではその概要を述べるとともに、同基盤とこれまでの実験に基づいたスマートシティの構築モデルに関して説明する。

## SOXFire: An Urban Sensor Data Distribution Platform based on XMPP

TAKURO YONEZAWA<sup>1,a)</sup> TOMOTAKA ITO<sup>1,b)</sup> YIN CHEN<sup>1,c)</sup> JIN NAKAZAWA<sup>1,d)</sup>

### 1. はじめに

現在、都市部への人口流入が地球規模で進んでおり、それに伴い都市における効率的なエネルギー管理、経済成長の必要性、住民の安全と生活の質向上など、今後数年間に対処しなければならない様々な課題に我々は直面している。これらの問題を解決するため、現在多くの研究者が都市のスマート化(=スマートシティ)に注目している。スマートシティには様々な定義が存在するが、総じていうと「ICT(情報通信技術)を駆使し、エネルギー、下水道、交通といった社会インフラを効率的に整備・運用する都市」であり、「都市運営を担う自治体を中心とし、その自治体を構成する様々な組織および組織が運用する自律分散システムの集合体(=System of Systems)」とされる。スマートシティを実現するためには都市の状況をリアルタイムに

把握し、その状況に応じて都市機能の変化や人々の行動変容を促す必要があり、そのためにはネットワーク接続されたセンサやアクチュエータ(=IoT)を都市中に設置し、得られた都市情報を様々な主体に流通させることが重要となる。

本研究ではスマートシティにおける情報流通基盤を実現するために、多種多様で複数の組織のセンサ設置主体が設置したIoTセンサ等から得られる都市センサデータを、様々な開発者がアクセス権限などを考慮した上で有効的に利用可能とさせるミドルウェアの構築を行う。都市センサデータの情報源はIoTセンサだけでなく、参加型センシングなどに代表される人から提供されるセンサデータ、SNS等Webから得られるセンサデータ等、多岐に渡るため、得られるセンサデータそのものだけではなく、そのメタ情報を表現できることが重要となる。更に、膨大なセンサデータの流通を実現するためには、流通基盤はスケーラビリティや拡張性を有する必要がある。また、他者に共有可能なセンサデータとそうでないセンサデータを区別した上で、他者に柔軟なアクセスを可能とさせる必要がある。本

<sup>1</sup> 慶應義塾大学大学院政策・メディア研究科  
Endo5322, Fujisawa, Kanagawa 252-0882, Japan

a) takuro@ht.sfc.keio.ac.jp

b) tomotaka@ht.sfc.keio.ac.jp

c) yin@ht.sfc.keio.ac.jp

d) jin@ht.sfc.keio.ac.jp

研究ではこれら都市センサデータを流通させるためにシステムが満たすべき要件を整理した上で、その要件を満たすミドルウェアシステム SOXFire を構築する。SOXFire は XMPP を拡張し Publish-Subscribe モデルに基づいたセンサデータの流通を可能とする Sensor-Over-XMPP プロトコルに従った情報流通を実現するとともに、上述した要件を満たすための機能拡張が施された XMPP サーバソフトウェアとして実装された。また、アプリケーション開発者が SOXFire を用いたサービス開発を容易に行うため、複数のプログラミング言語を対象としたライブラリを提供している。

本稿では、SOXFire の詳細を述べるとともに、同基盤を利用しこれまで神奈川県藤沢市を中心に運用してきたサービスについて紹介を行う。また、SOXFire のデータ流通に関するパフォーマンスを定量的に評価するため、センサデータ送信実験の結果を示す。本研究の貢献は、以下の3点である。

- 都市センサデータ流通基盤が満たすべき要件を整理したこと
- 同要件を満たす SOXFire の提案
- SOXFire の実運用事例および定量的評価の提示

## 2. 都市センサデータ流通システムが満たすべき要件

### 2.1 スマートシティのレイヤリング

一般的にスマートなシステムとは、リアルタイムな状況把握と、その状況に対応する高い応答性を示すシステムを指す。よって、スマートシティとは、都市規模において、その状況把握 (Awareness) と応答性 (Responsiveness) を提供可能な都市である。都市は人、建物、車両、都市インフラ (道路、下水道など) など多様な実空間オブジェクトと、それらによって構成される多種多様な組織・社会 (家族や企業含む) の集合により成立している。よって、都市をスマート化する一つの手段は、その構成要素である実空間オブジェクトおよび組織・社会をスマート化し、そこで得られた情報を異種組織間で共有したり、都市レベルで共有することである。この構成要素を考慮し、スマートシティをレイヤー化されたモデルとして表現すると、図1のように示すことができる。図1中、最下位に位置する実空間オブジェクトに IoT センサを付与することで、実空間オブジェクト自身とその周辺の状況把握と応答能力を付与し、それを利用したアプリケーションが開発可能となる。また、実空間オブジェクトから得られた情報をそのオブジェクトが所属するコミュニティ内で共有することで、コミュニティの状況把握が可能となり、それを利用したサービスが実現できる。例えば、運送会社に所属する車両の位置情報を会社内で共有することで、スマートな運送計画立案が可能となる。更に、コミュニティ内で共有した情報の

うち、可能な情報を外部のコミュニティで共有し、最終的には様々なコミュニティ間 (=都市レベル) で共有することにより、都市の状況把握が可能となり、スマートシティの実現に寄与できる。このシナリオに基づく、スマートシティにおける都市センサデータ流通システムが満たすべき要件は次節のように整理できる。

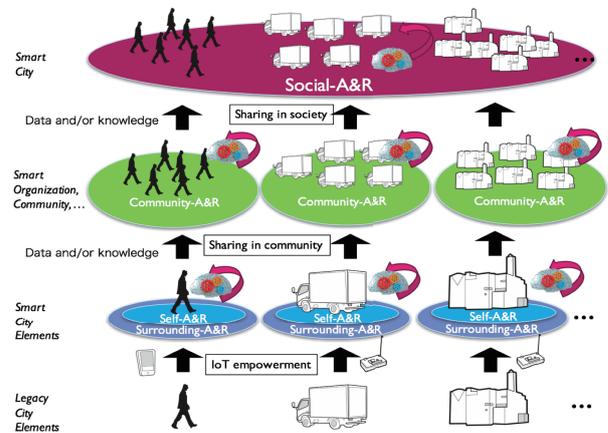


図1 スマートシティのレイヤモデル

### 2.2 要件

上述したスマートシティのモデルを考慮した場合、都市センサデータ流通基盤は、様々なデータ提供主体が設置した多種多様なセンサから得られたデータを、提供主体と異なる利用主体の存在を考慮した分散システムとして実現する必要がある。したがって、都市センサデータ流通基盤は下記の6つの要件を少なくとも満たす必要がある。

**要件1:** ヘテロニアスなセンサを対象とできること

物理的なIoTセンサだけでなく、参加型センシングやWEBセンシング等多種多様なセンシング手段を対象とできる必要がある。

**要件2:** 柔軟かつセキュアなデータアクセス方式

多種多様なデータ提供者・データ消費者が存在するため、センサデータにアクセスするためのアクセス権は、柔軟かつセキュアに設定可能であるべきである。

**要件3:** 多様な方法によるセンサデータ取得

アプリケーションによっては、継続したストリームデータとしてデータを取得したい場合もあれば、最新の1件のデータだけ取得したい場合もある。すなわち、多様な方法によってデータ取得ができるべきである。

**要件4:** 運用の容易性

実践的なシステムとして運用可能とするため、容易な運用性を実現する必要がある。センサデータの設置、消費、流通など、そのデータ生成・消費プロセスを細やかに支援可能であるべきである。

**要件5:** スケーラビリティと拡張性

数百一数万一数億のセンサを対象とするため、システ

ムはスケーラビリティを有する必要がある。また、各組織によって有効に活用可能なシステムへと進化させるため、その拡張性も重要である。

#### 要件 6: 複数組織間での接続性

センサデータは各組織内で生産されるため、それらの組織間でのデータ流通を可能とする必要がある。

### 2.3 関連研究

これまで、スマートシティのための情報基盤を実現するため、センサやアクチュエータのためのミドルウェア [1], [2]、テストベット上のアプリケーション開発のためのツール群 [3], [4]、実際の都市へのデプロイメントの事例 [5] など、様々な研究が行われてきた。これら先駆的な取り組みはスマートシティのための基盤の事例として位置づけられるが、主に IoT やそこから得られるオープンデータを対象としており、本研究のように参加型センシングや WEB センシングなどより多様なデータソースを考慮していない。また、これらの研究は基本的にクラウドを利用した中央集権的なアーキテクチャに基づいており、本研究が目的とする、異なる様々な運用組織が存在しつつ、それらの間でセンサデータのやり取りを効果的に行うなどの考慮が十分でない。クラウドコンピューティングに基づいた CityHub [6] は、データのオープン性を高めるため、CKAN\*1 と接続し、メタ情報を含んだデータカタログおよびデータへのアクセスを提供している。一方、CKAN は RESTful なアクセスを前提としており、Publish-Subscribe モデルなど動的なデータに対するアクセスが考慮されていない。また、IoT データアクセスのためのミドルウェアとしては、MQTT [7] や CoAP [8] などが広く利用されており、MQTT は Amazon が提供する AWS IoT プラットフォーム\*2 にも採用されている。一方、これらのプラットフォームは基本的にセンサノード設置主体と利用主体が同一であり、こういったセンサが設置されているか予め組織内で把握されていることを前提としており、本研究が対象とするオープンなデータ流通には必ずしも適さない。これらのことから、本研究で示す要件を満たすスマートシティのための都市センサデータ流通基盤はこれまで部分的にしか存在していないため、本研究では上述した要件を満たす基盤構築を行う。

## 3. SOXFire

### 3.1 Sensor-Over-XMPP

上述した要件を満たすために、我々は eXtensible Messaging and Presence Protocol (XMPP) を活用することとした。XMPP は、伝統的にチャット通信に使用される XML 形式のインターネットプロトコルである。XMPP には、ユーザーアカウントとセキュアな認証機能、

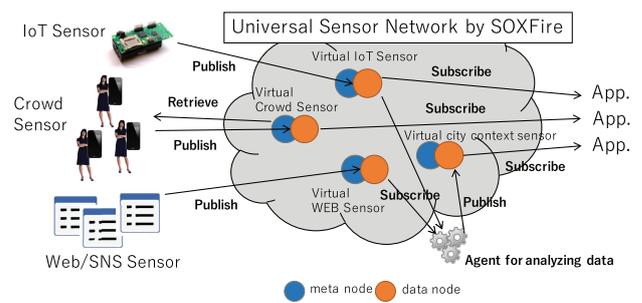


図 2 SOXFire における仮想センサノードモデル

柔軟なアクセス制御メカニズム、パブリッシュサブスクライブ (pubsub) イベントモデル、スケーラブルなクラスタリングとフェデレーション機能など、いくつかの機能を備えている。さらに、XMPP は XML を使用するため、開発者が独自のスキームを独自に定義できるように拡張性を有する。MQTT や CoAP のようないくつかの IoT プロトコルが存在し、それらを実際に活用することもでき、各プロトコルは特定の目的のために最良の選択であるケースも存在する。しかし、我々は現在、XMPP がソーシャルビッグデータ流通の観点からは、上述した要件を満たすための最も実用的なプロトコルの 1 つであると考えられる。特に本課題では、Sensor Andrew プロジェクトで提案された Sensor-Over-XMPP (SOX) 仕様 [9] に着目した。SOX は、センサデータ共有のための pubsub モデルを活用しており、センサデータを配信するためのシステムの基盤として設計されている。しかし、オリジナルの SOX では、物理的な IoT センサに焦点を当てており、多種多様なデータリソースの活用を対象としていない。よって、本課題では、スマートな都市のさまざまな層のユーザ間で、多種多様なセンサデータを共有するために XMPP プロトコルおよび SOX を拡張したシステムの構築を行なった。

### 3.2 SOX における仮想センサモデル

前述のように、本課題におけるセンサモデルは Sensor Andrew プロジェクトで提案された SOX 仕様を拡張している。SOX 仕様では、物理センサは pubsub イベントノードとして対応する仮想センサとして表現される。pubsub モデルを使用することは、MQTT と同じように、IoT データを大量に配布するための理想的な方法である。一方 SOX の利点の 1 つは、センサのメタ情報を表現可能なスキームとなっていることである。これにより、ユーザは必要なセンサを理解/発見することができる。これは、マルチコミュニティ環境間でソーシャルビッグセンサデータを共有するための重要な機能である。SOX では、物理センサに対応する仮想センサは、SensorName\_meta と SensorName\_data という 2 つの PubSub ノードに関連付けられている。図 2-1-1 に、SOX における実世界センサと仮想センサの関係の概要を示す。Postfix に \_meta を持つ PubSub ノード

\*1 <http://ckan.org>

\*2 <https://aws.amazon.com/jp/iot/>

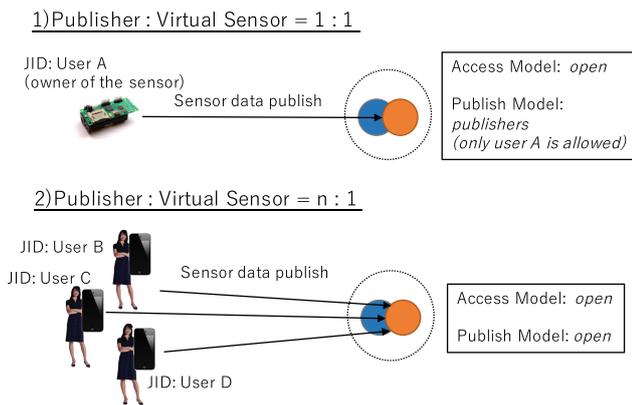


図 3 センサデータの配信パターン

は、センサデバイス名 (mySensor など)、デバイスタイプ (屋外天候など)、デバイスに搭載されたセンサ (温度や湿度など)、単位 (例えば、摂氏とパーセント) などを保持する。また、Postfix に `_data` を有する PubSub ノードは、実際のセンサデータを配信するために使用される。SOXFire が提供する SOX ライブラリでは、このメタデータノードとデータノードは隠蔽されノードを活用可能となっているため、メタ/データノードの違いを意識することなく、メタ情報と実際のデータの両方に簡単にアクセス可能である。SOXFire では、XMPP の PubSub 機能およびアクセス権<sup>\*3</sup>を活用し、ノードを購読するための「公開」、「許可」または「ホワイトリスト」、ノードにデータを公開するための「公開」、「発行者」、「加入者」などの、pubsub ノードに対する柔軟なアクセス制御を設定可能となる。アクセス制御を使用することにより、SOX に次のような異なるセンサパターンを提供することが可能である。

#### パターン 1 パブリッシャ : 仮想センサ = 1 : 1

このパターンでは、物理エンティティが仮想センサノードに 1 対 1 で関連付けられていることを意味する。これは、IoT センサノードの基本モデルである。たとえば、加速度センサと温度センサを備えたセンサノードが、対応する仮想センサにそれらのセンサデータを公開することだけが許可されているとする。仮想センサに公開されたデータは、それを購読するどのユーザにも配布される。この場合、公開モデルを「パブリッシャ」として、特にセンサノードの所有者のみに設定し、アクセスモデルを誰にとっても「オープン」に設定することでアクセス権が設定できる (図 3 の 1 参照)。

#### パターン 2 パブリッシャ : 仮想センサ = n : 1

このモデルは、誰でも (または許可されたユーザ) も仮想センサノードのセンサデータパブリッシャとして機能できることを意味する。これは、参加型センシングのための基本モデルとなっている。この場合、公開

モデルとアクセスモデルの両方を「公開」に設定する必要がある (図 3 の 2 参照)。

### 3.3 要件を満たすためのアプローチ

XMPP および Sensor-Over-XMPP を採用することで、2.2 節で掲げた要件のいくつかは解決できると考えられる。一方で、既存の XMPP の仕様ではサポートされない要件も存在するため、その要件を次のようなアプローチで満たす XMPP サーバ SOXFire を本研究では提案する。

#### 要件 1: ヘテロジニアスなセンサを対象とできること

本要件は、Sensor-Over-XMPP の仕様を拡張し、物理的な IoT センサだけでなく、参加型センシングや WEB センシングなどのセンサ種類の宣言を可能とさせる。参加型センシングでは、通常、都市の特定の出来事を報告するようにユーザに依頼する (例えば、写真付きの道路被害報告など)。参加型センシングのためのメタデータ表現として、'参加型' などのいくつかのセンサタイプを追加し、'singleChoice'、'multipleChoice'、'freeFormText' または 'image' などのメタ表現を定義した<sup>\*4</sup>。このメタデータは、パブリッシャのためのレポート作成インタフェースの自動生成にも使用可能である。

#### 要件 2: 柔軟かつセキュアなデータアクセス方式

本要件は、3.2 節でも述べたように、XMPP が提供する柔軟なユーザモデル、アクセスモデルにより実現される。一方 XMPP では仕様として、フェデレーションを行っている XMPP サーバ間では、匿名ユーザでのサブスクライブ・パブリッシュを行うことができない。SOXFire ではデータアクセスの利便性を高めるため、匿名ユーザを利用したフェデレーション先の仮想センサノードのアクセスを実現する。

#### 要件 3: 多様な方法によるセンサデータ取得

本要件は、XMPP が提供する PubSub 機能、およびサーバ側で任意の個数キャッシュされたデータに対する明示的なデータ問い合わせにより、プッシュ通信およびプル通信によるセンサデータ取得を可能とすることで満たす。一方で、過去の膨大なセンサデータをすべてキャッシュすることは現実的でないため、過去の任意のデータに関しては外部サービス化を行う必要がある。

#### 要件 4: 運用の容易性

本要件を満たすため、提案する Sensor-Over-XMPP の拡張に基づく仮想センサノード管理 (生成、削除、検索など)、センサデータの取得などを容易に行えるライブラリを提供する。

#### 要件 5: スケーラビリティと拡張性

<sup>\*3</sup> <https://xmpp.org/extensions/xep-0060.html#accessmodels>

<sup>\*4</sup> <http://sox.ht.sfc.keio.ac.jp/schema/schema.html>

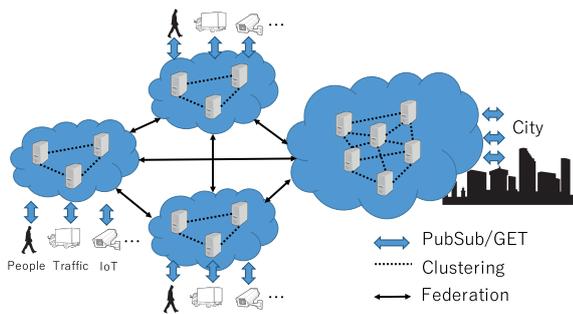


図 4 SOXFire アーキテクチャ

本要件は、XMPP サーバにおいて実績のあるクラスタリング機能を利用することで満たすことができる。また、XMPP は XML を用いているため、拡張性を有する。

#### 要件 6: 複数組織間での接続性

本要件は、XMPP のサーバ間フェデレーション機能を利用することで、満たすことができると考えられる。フェデレーションとは、独立した XMPP ドメインを持つサーバ間で相互通信を可能とする方法である。また、クライアントアプリケーションからフェデレーション越しのサブスクリプションを容易に行えるよう、クライアントライブラリを構築する必要がある。

## 4. 実装

上述した要件を満たす SOXFire サーバソフトウェア、クライアントソフトウェアの実装を行った。以下で説明するソフトウェアは、全て我々の Web サイト<sup>\*5</sup>から入手可能となっている。

### 4.1 SOXFire サーバ

XMPP サーバの実装を行うにあたって、いくつかのオープンソースの XMPP サーバソフトウェアの調査を行った。調査対象として ejabberd, tigase, Openfire などを挙げ、実際にインストールして挙動の確認と、ソースコードの調査を行った。その中で、我々は Apache license でオープンソース化されている Openfire を選択した。Openfire は 2002 年より開発されている Java 言語で書かれた XMPP サーバソフトウェアであり、継続して開発がなされている。Openfire はインストールが容易であり、管理ツールの機能やクラスタリングのためのプラグインなどもコミュニティにより豊富に提供されている。一方、3.3 節で述べた要件を全て満たしてはならず、我々は Openfire のソースコードを修正・拡張することにより都市センサデータの流通に適した SOXFire を実装した。主に、下記の変更を行った。

- 匿名サブスクリプションの有効化

Openfire ではサブスクリプションは登録ユーザのみ

に許可されており、アンサブスクリプションを行わない限りその情報は永続化され内部データベースに保存される。しかしながら、都市センサデータの利用・共有を促進し多様なアプリケーションを実現可能とするためにはユーザ登録をしなくともオープンなデータにはアクセスできるべきであり、SOXFire では匿名ユーザのサブスクリプションを可能としている。

- サーバ間フェデレーション時の PubSub ノードアクセス

Openfire はサーバ間フェデレーション機能を備えている。しかし、Openfire ではフェデレーション先の Pubsub ノードに対するアクセスは匿名ユーザ・登録ユーザ両方行えない仕様となっていた。SOXFire ではこれを可能とし、自分が所属する XMPP ドメインの外部にある XMPP サーバ内のノードに対して、サブスクリプションおよびパブリッシュを可能とした。これにより、クラスタリングとフェデレーションを組み合わせたスケーラブルな情報流通基盤を実現可能であると考えられる (図 4 参照)。

- データベーストランザクションの効率化

Openfire では XMPP が提供する様々な機能を実現するために、ユーザ情報、PubSub ノード情報、パブリッシュデータの保存、サブスクリプション情報などをデータベースに書き込み利用を行う。SOXFire では多数のセンサノードに対する多数のデータ配信を想定しているため、高頻度のデータベースアクセスを抑えるよう、データベースへのトランザクション回数を減らす工夫を行っている。また、匿名ユーザのサブスクリプションに関して、匿名ユーザから生存確認応答がなくなった場合、その情報をデータベースから自動的に削除することによりストレージ使用容量を抑えるよう工夫を行っている。

上述した機能を有する SOXFire は Apache License によって公開を行っている。最新の SOXFire は、Openfire バージョン 4.2.3 を修正して実装されている。現在、我々は複数のプロジェクト、都市において異なるドメインの SOXFire を複数台運用しており、それぞれはフェデレーション機能によって相互接続がなされている。

### 4.2 SOXFire クライアントライブラリ

SOXFire が提供する Sensor-Over-XMPP に基づくセンサデータアクセスをアプリケーション開発者が容易に行えるよう、XMPP プロトコルや XML での通信を隠蔽するクライアントライブラリを実装した。特に、Sensor-Over-XMPP ではセンサノードのメタ情報とセンサデータを区別するために、2つの Pubsub ノード (`_meta` と `_data`) の組み合わせで動作するが、本ライブラリではその区別を意識せず利用可能としている。現在、安定バージョンが

\*5 <http://sox.ht.sfc.keio.ac.jp>

Java, Javascript, Objective-C の各言語で提供されており、Python 言語で一部の機能が実装されている。下記に、Java 言語においていくつかの機能の実装例を示す。

● SOXFire サーバへの接続

```
SoxConnection con =
    new SoxConnection("server-name.org");
```

● センサノードのリスト取得

```
//get the list on connected server
List<String> nodeList = con.getAllDeviceList();

//get the list on federated server
List<String> nodeList2 =
    con.getAllDeviceList("other-server.org");
```

● センサノードのインスタンス化（上：接続先 SOXFire ノード上のセンサノードを対象とする場合、下：フェデレーション先を指定する場合）

```
//associate to virtual sensor on connected server
SoxDevice sd = new SoxDevice(con, "sensor_name");

//associate to virtual sensor on federated server
SoxDevice sd2 =
    new SoxDevice(con, "sensor_name",
        "other-server.org");
```

● .meta ノードからメタ情報の取得

```
Device deviceInfo = sd.getDevice();
```

● .data ノードから最後にパブリッシュされたデータの取得

```
Data data = sd.getLastPublishData();
```

● センサノードのサブスクライブおよびパブリッシュされたデータに対するイベントハンドラ

```
sd.subscribe();
sd.addSoxEventListener(this);
...
public void handlePublishedSoxEvent(SoxEvent e){
    List<TransducerValue> values =
        e.getTransducerValues();
    //then write processing of the data
}
```

## 5. プロトタイピングツール

SOXFire を利用したアプリケーションのプロトタイプ構築を容易に行えるよう、Node-RED から SOXFire 内の仮想センサノードに対するサブスクライブおよびパブリッシュを可能とする Node の実装も提供されている\*6。Node-RED は IBM によって開発がなされ、現在は Apache License2.0 で配布されているフロー・ベースの開発ツール

\*6 <https://www.npmjs.com/package/node-red-contrib-sox>



図 5 Node-RED における SOX ノードおよびその設定画面

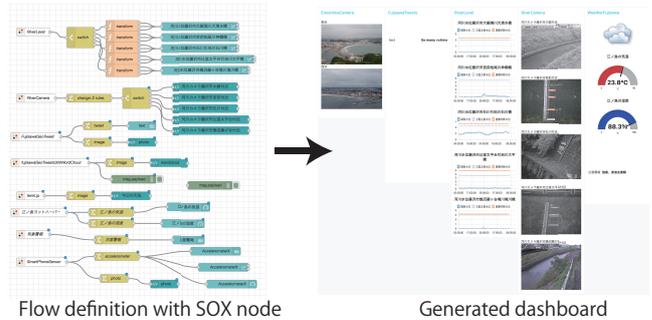


図 6 Node-RED におけるフロー定義例と情報ダッシュボード

であり、Node と呼ばれる様々なコンポーネントをワイヤで配線することにより、様々な機能の I/O 連携を視覚的に構成可能とする。図 5 に、SOX Node（図 5 左）を利用する際の、サブスクライブ対象となる仮想センサノード名の設定画面（図 5 中央）と、SOXFire へのログイン設定画面（図 5 右）を示す。SOX Node で受信したセンサデータはワイヤで連結されたノードに JSON オブジェクトとして受け渡され、任意の処理を記述することができる。図 6 は、SOX Node と Dashboard Node を組み合わせ、受信したセンサデータを分解し、ダッシュボード形式で表示した例を示している。この例では、SOXFire サーバ内に存在する河川監視カメラ、河川水位センサ、気象情報センサ、Twitter センサなど異種データ源からデータを受信し、折れ線グラフやテキスト、画像などでデータの可視化を行っている。Node-RED から SOXFire の仮想ノードにアクセス可能とすることで、様々なコンポーネント連携を容易に実現できる。

## 6. 運用実験

本研究で実現した SOXFire を用い、神奈川県藤沢市を中心とした運用実験をこれまで行ってきた。本章では、代表的な事例を 2 点紹介する。

### 6.1 清掃車を用いた環境センシング

日本国内には、環境省が設置した大気汚染常時監視測定局が約 2,000 局存在し、そのうち 5 局が藤沢市に存在する。一方上述の通り同市には約 43 万人の住民が暮らしていることから、8 万人以上に 1 局ほどしか存在しないということも言え、各住民の観点では、自分の吸っている空気がど

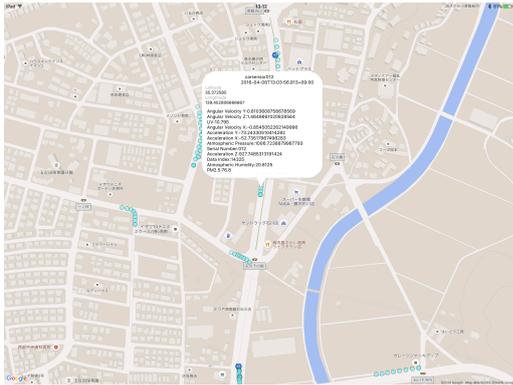


図 7 センシングデータの可視化例：連続する小円は直近 10 秒間の清掃車の運行経路，中央吹出部は当該地点のセンサデータ値

れだけ綺麗なのか？ということも、正確にはほとんど知り得ないということになる。そこで筆者らは、この空白を埋めるため、毎日市内をくまなく走行する清掃車を IoT 化してそこにセンサを取り付け、ゴミや資源だけでなく環境情報も集めるシステムを開発し、運用している。

同システムでは、清掃車の屋根の上にセンサボックスを取り付け、加速度、角速度、方位、気圧、湿度、温度、紫外線 (UV-A)、照度、PM2.5、および GPS の各値を毎秒 100 回測定している。センサボックスは、清掃車内に設置した小型計算機 OpenBlocks IoTBX1<sup>\*7</sup>と USB ケーブルで接続されており、測定データを同計算機から最大通信速度 200kbps の携帯電話網を通じて、やはり毎秒 100 回、SOXFire に送信される。測定データは藤沢市役所内や藤沢市環境事業センター内で図 7 に示すように可視化され、清掃車のリアルタイム位置把握や環境情報の把握に供されている。特に大規模災害時等に、これらの車両の存在位置を即時把握することで、行政による初動に役立てられる可能性が高いと、藤沢市より期待されている。

## 6.2 自治体職員参加型センシング

落書きや不法投棄物といった事象は、落書きとそうでない絵の区別、あるいは不法投棄物と合法的に排出されたゴミの区別が必要で、それらは現状のセンサでは不可能である。このような事象は一般的には、住民や行政職員が発見した際に、電話等で通報され、担当部署へ場所等が伝達されて対応される。この連絡過程で、場所が曖昧であったり大きさが不明であったりするなど、情報が不完全である場合や、住民等による通報から担当課への連絡までに複数の担当者を介す必要があり、情報の伝達に時間を要する場合がある。これらは行政の業務効率化の観点で課題であり、藤沢市でも大きな問題となっていた。

そこで我々は、自治体職員を対象として、上記のような不具合事象に関する情報を担当課等と共有できる参加型セ

<sup>\*7</sup> [http://openblocks.plathome.co.jp/products/obs\\_iot/bx1/](http://openblocks.plathome.co.jp/products/obs_iot/bx1/)

ンシングシステム「藤沢みなレポ」を開発し、2016 年 10 月から運用している [10]。同システムでは、行政職員が、スマートフォンで不具合事象を撮影した画像にラベルを付与して投稿でき、投稿画像を SOXFire を介して担当課の職員等へ即時に共有できる。投稿された画像とその位置情報等は、ウェブ画面上で閲覧できる他、全投稿画像にフィルタを適用して特定のラベルを持つもののみを地図上に示すこともできる。これらによって、特定の不具合事象に関する詳細な情報を正確に把握することや、様々な不具合事象の地理的分布を網羅的に把握することが容易になっている。現在本システムは、不具合事象として落書き、不法投棄、ゴミの出し間違い、道路の陥没や道路標示のかすれ、街灯の不具合等に対応している。

## 7. センサデータ配信実験

SOXFire は XMPP のクラスタリングやフェデレーション機能を利用することでスケラブルに情報の流通が可能であるが、1 台の SOXFire サーバによりどの程度の遅延でセンサデータの配信が行えるか、実験を行った。

### 7.1 実験環境

実験は、研究室内のネットワークに接続された 3 台のコンピュータを利用し行った。SOXFire は Mac Pro 内で稼働するバーチャルマシン (VMWare/Ubuntu16.04LTS) 上にインストールされ、同バーチャルマシンは 3.5GHz Intel Xeon E5 の CPU プロセッサが 2 コアおよび 8GB のメモリが割り当てられている。センサデータ配信に利用するコンピュータは 2.7GHz 12 コア Intel Xeon E5 の CPU で、64GB のメモリを搭載しており、Java で実装された SOXFire クライアントライブラリによりデータ送信を行う。配信されるデータは、下記の通り、XML 形式で 1 配信あたり 483 バイトである。

```
<iq to='pubsub.soxfujisawa.ht.sfc.keio.ac.jp'
id='r2nuf-40' type='set'><pubsub xmlns=
'http://jabber.org/protocol/pubsub'>
<publish node='testNode_data'><item><data>
  <transducerValue id='temperature' timestamp=
  '2018-08-08T20:37:25+09:00' rawValue='35'>/>
  <transducerValue id='latitude' timestamp=
  '2018-08-08T20:37:25+09:00' rawValue='35.23456'>/>
  <transducerValue id='longitude' timestamp=
  '2018-08-08T20:37:25+09:00' rawValue=
  '139.4316725'>/>
</data></item></publish></pubsub></iq>
```

SOXFire から該当のセンサノードをサブスクリプトし、データを受信するコンピュータは 2.9GHz Intel Core i7 の CPU を搭載し、メモリは 16GB である。送信プログラム同様、受信プログラムも Java 言語用のライブラリを利用した。SOXFire サーバおよび配信コンピュータはギガビット Ethernet の有線 LAN で接続され、受信コンピュータは

802.11nの無線WiFiネットワークに接続されている。

実験はまず、1台あたりの最大配信可能データ数を調査するため、配信データ数を増加させながら、サブスライバが1台の際に受信した際の遅延(送信時のタイムスタンプと受信時のタイムスタンプを比較)を計測した。次に、配信データ数を毎秒10回および、毎秒100回に固定しつつ、受信を行うサブスライバの数をマルチスレッドで増加させながら、遅延への影響を計測した。

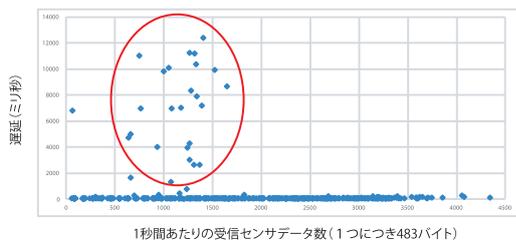


図8 1秒あたりの受信データ数と遅延の関係

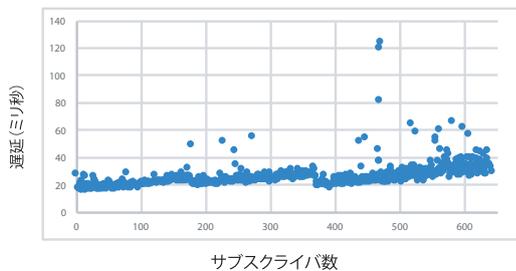


図9 毎秒10回のデータ配信時のサブスライバ数と遅延の関係

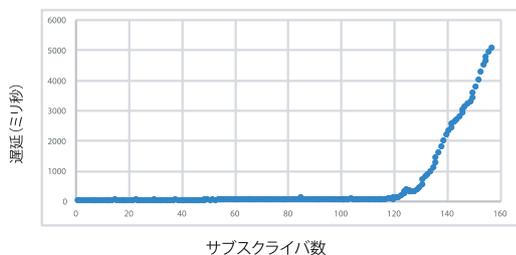


図10 毎秒100回のデータ配信時のサブスライバ数と遅延の関係

## 7.2 実験結果

図8に、1秒あたりの配信データ数の増加に伴う遅延時間の結果を記す。1秒あたりの送信データ数が3000(約1.5メガバイト/秒)あたりまでは遅延時間はほとんど変化しなかったが(およそ50ミリ秒前後)、それを越えたあたりから遅延が発生し、それにともない1秒あたりの受信データ数も低下してしまった(図8中、赤で囲った部分が該当する)。次に、配信データを毎秒10回に固定しつつ、サブスライバの数を変化させていった結果を図9に示す。毎秒10回に固定した場合は、大きな遅延の増加は認められないまま、OSのマルチスレッド数の制限に達した。一方、毎秒100回のデータ配信時においてサブスライバ数を増加

させた場合は、サブスライバの数が120を超えたあたりから遅延が発生した(図10)。これらの結果はコンピュータのスペックやネットワークに依存するが、都市に存在するセンサノード数が1分間に1回程度のデータ配信であれば、数千規模のノードから数百人に対して遅延なくセンサデータの配信が行えることを示唆している。今後、より大規模な環境での評価実験や、クラスタリング時のパフォーマンスの評価を行っていく。

## 8. まとめ

本稿では、複数の異なる組織が内包され、それぞれの組織でのデータ利用と異種組織間でのデータ共有を柔軟かつセキュアに行うための、都市センサデータ流通基盤SOXFireを提案した。SOXFireはXMPPプロトコルを拡張し、Sensor-Over-XMPPに基づくデータ配信を可能としつつ、都市センサデータ流通のための要件を満たす機能を提供する。今後は、複数の都市での運用実験を進め、様々な実環境での評価を行っていきたいと考えている。

## 謝辞

本研究の一部は、国立研究開発法人情報通信研究機構に支援頂いた。

## 参考文献

- [1] Raffaele Gialfreda. *iCore: A Cognitive Management Framework for the Internet of Things*, pages 350–352. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [2] Internet of things - architecture. <http://www.iot-a.eu>.
- [3] Citysdk. <http://www.citysdk.eu>.
- [4] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth, et al. Spitfire: toward a semantic web of things. *Communications Magazine, IEEE*, 49(11):40–48, 2011.
- [5] Luis Sanchez, Luis Mu noz, Jose Antonio Galache, Pablo Sotres, Juan R. Santana, Veronica Gutierrez, Rajiv Ramdhany, Alex Gluhak, Srdjan Krco, Evangelos Theodoridis, and Dennis Pfisterer. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61:217 – 238, 2014. Special issue on Future Internet Testbeds.
- [6] R. Lea and M. Blackstock. City hub: A cloud-based iot platform for smart cities. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 799–804, Dec 2014.
- [7] MQTT. <http://mqtt.org>.
- [8] CoAP. <http://coap.technology>.
- [9] A. Rowe, M. E. Bergeés, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett, J. M. F. Moura, and L. Soibelman. Sensor andrew: Large-scale campus-wide sensing and actuation. *IBM J. Res. Dev.*, 55(1&2):66–79, January 2011.
- [10] 米澤 拓郎, 伊藤 友隆, 坂村 美奈, 竹内 孝, 納谷 太, 上田 修功, and 中澤 仁. 自治体職員参加型センシングによる業務効率化と都市理解の向上. マルチメディア, 分散, 協調とモバイルシンポジウム, pages 320–329, 2018.