情報処理学会論文誌 Vol.56 No.1 273–283 (Jan. 2015)

# M2Mにおけるデバイスへの統一アクセス手法の提案と評価

中原 祥吾[1] 末田 欣子[1,2,a)] 多田 好克[1]

受付日 2014年5月7日, 採録日 2014年10月8日

**概要**：近年，通信技術の進歩によりM2M（Machine-to-Machine）通信が普及している．M2M通信とは，人間の手を介在せず，デバイスどうしを通信させる仕組みやコンセプトのことである．M2M通信を利用することで，自動化制御，高齢者医療監視，スマートホーム構築などといったM2Mアプリケーションの構築が可能となっている．しかし，M2Mアプリケーションを開発するためには，通信方式や制御方式が異なる様々なデバイスを扱うことを考慮しなければならない．そのため，デバイスごとに処理を用意する必要があり，M2Mアプリケーションが肥大化し，複雑になる要因となっている．そこで本研究では，M2Mアプリケーション開発の効率化を目的として，通信方式や制御方式の異なる様々なデバイスに対して，RESTアーキテクチャに基づき，デバイスをRESTサービス化することで，M2Mアプリケーションから共通の方法でアクセスできる仕組みを提案する．具体的には，デバイスへの統一的なアクセス手法を提供するため，OSGiにRESTアーキテクチャでアクセスできる仕組みをJAX-RSを利用して実現した．提案システムを実装し，ケーススタディを用いて評価を行った結果，M2Mアプリケーションが単純になり，仕様変更におけるM2Mアプリケーションへの影響が低減され，開発効率が向上することを確認した．

キーワード：M2M，REST，JAX-RS，OSGi，開発

# Proposal and Evaluation of a Unified Restful Access Method for M2M Devices

Syougo Nakahara[1] Yoshiko Sueda[1,2,a)] Yoshikatsu Tada[1]

Received: May 7, 2014, Accepted: October 8, 2014

***Abstract:*** Recent advances in communication technologies have accelerated the growth of machine-to-machine (M2M) communication. M2M communication is a concept and mechanism for communication among devices without human intervention. M2M communication makes it possible to build applications such as automated controls for sensors and medical monitoring devices for seniors, smart homes, and other M2M applications. However, it is necessary to consider how to deal with various kinds of devices with many different controls and communication methods in order to develop effective M2M applications. One approach is to use separate interface software for each device. However, this results in bloated and complicated software for M2M applications. In this paper, we propose a unified interface that enables a common access method to be used for different types of devices. Specifically, our system realizes a unified interface in the REST (REpresentational State Transfer) architecture using JAX-RS (Java API for RESTful Web Services) on OSGi (Open Service Gateway initiative). It is aimed at achieving more efficient M2M application development. We implemented our proposed system and evaluated it in a case study. The results indicated that the effect of the specification changes was reduced and the development efficiency improved.

***Keywords:*** M2M, REST, JAX-RS, OSGi, development

## 1. はじめに

様々なセンサがスマートフォンに搭載されているように

[1] 電気通信大学
The University of Electro Communications, Chofu, Tokyo 182–8585, Japan
[2] 日本電信電話株式会社ネットワーク基盤技術研究所
NTT Network Technology Laboratories, Musashino, Tokyo 180–8585, Japan
[a)] sueda.yoshiko@lab.ntt.co.jp

© 2015 Information Processing Society of Japan

ネットワークに接続されるデバイスが増加している．シスコのレポートでは，2020 年にはネットワークに接続されるデバイスの数が 500 億台に到達すると予測されている [1]．

また，人間の手を介在させず，デバイスどうしを通信させる仕組みやコンセプトを M2M（Machine-to-Machine）通信と呼んでいる．現在，M2M 通信を使用したスマートハウスの構築や機器制御の自動化といった M2M アプリケーションの開発が検討されている [2]．このような M2M 通信により集められたデータは，新しいビジネスやサービスに利用できる可能性があるため，非常に期待されている．

しかし，M2M アプリケーションを開発するためには，通信方式や制御方式の異なる様々なデバイスを操作する必要がある．そのため，複数の異なるデバイスを使用する M2M アプリケーションでは，デバイスごとに固有の処理を実装する必要がある．したがって，ソースコード量が肥大化し，複数のデバイスを組み合わせた処理が，非常に複雑になる可能性があるため，開発を困難にすることが考えられる．

そこで，このような課題を解決するために，複数のアプリケーション，ネットワーク，デバイスを統合的に扱うことができる水平統合型 M2M が検討されている [3]．**図 1** は，水平統合型 M2M アーキテクチャモデルを示したものである．このモデルでは，M2M プラットフォームが異なるネットワークやデバイスを統合し，共通のインタフェースを提供することで，M2M アプリケーションの構築を容易にしている．

そこで本研究では，水平統合型 M2M アーキテクチャモデルに基づき，M2M アプリケーションの開発が容易になるデバイスへの統一的なアクセス手法を提供することを目的とする．

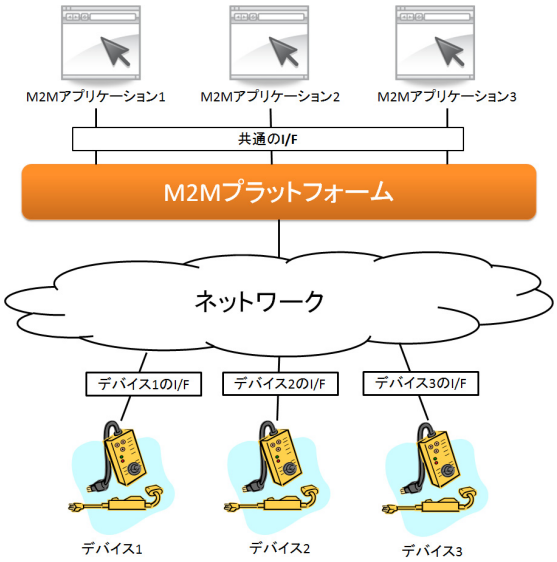


**図 1** 水平統合型 M2M アーキテクチャモデル
**Fig. 1** Horizontal integrated M2M platform.

本論文の構成は，以下に示すとおりである．まず，2 章で様々なデバイスに対して統一的なアクセスを行えるようにするための要求条件について述べる．次に，3 章で要求条件をもとに提案システムについて述べる．また，提案システムの有用性を検証するため，4 章で提案システムの評価について述べる．そして，5 章で既存研究との差異について述べる．最後にまとめと今後の課題について述べる．

## 2. 要求条件

本章では，水平統合型 M2M アーキテクチャモデルに基づき，様々なデバイスに対して統一的なアクセスを可能にするための要求条件について述べる．

### (1) 様々なデバイスの操作・制御

M2M アプリケーション開発では，様々な通信方式や制御方式によって，異なる複数のデバイスを操作する必要がある．また，似たようなデバイスでも機能が異なるデバイスが存在する．そのため，検討するインタフェースでは，どのようなデバイスが使用されても共通の方法で操作・制御を行えることが必要である．

### (2) 仕様変更への追随

M2M アプリケーション事例では，新しいデバイスの追加や不要になったデバイスを除去することなどで仕様変更が起こる可能性がある．インタフェースをデバイス固有の処理に依存して定義していた場合，仕様変更時には，インタフェースに大きな影響がでる．もちろん，M2M アプリケーションもインタフェースの変更に対応していく必要があるため，M2M アプリケーションにも仕様変更のコストがかかる．

したがって，検討するインタフェースは，デバイスの追加や除去に柔軟に対応できると同時に，デバイス固有の処理への依存性を抑えて，仕様変更時には，M2M アプリケーションに与える影響を小さくできることが必要である．

### (3) 開発効率の向上

インタフェースが複雑な場合，デバイス固有の処理よりもインタフェースの実装に，よりコストがかかる可能性がある．また，インタフェースを利用したデバイス間の通信も複雑になり，開発効率が低下する恐れがある．したがって，検討するインタフェースは，シンプルな通信設計が行え，実装が容易である必要がある．

### (4) リアルタイム性の実現

M2M アプリケーションでは，定められた時間内に処理を完了させることや逐次デバイスからセンシングする必要のある事例が考えられる．

欧州の標準化団体である ETSI [4], [5] などで検討されているサービスユースケースをもとに考えると，人の行動や移動交通などの用途には，秒オーダのレスポンスが期待される．対して，メータリングや家電制御などの場合は，分オーダのレスポンスが必要となる．増尾らの研究では，

© 2015 Information Processing Society of Japan

M2M の代表的なユースケースである家電を操作するための通信に基づく，性能評価例を示しており，リクエスト処理頻度は，2,000 世帯で 2,500 リクエスト/秒程度である [6]．また，現状の AT&T の実施している Digital Life [7] において家庭で利用可能なセンサは十数個である．

つまり，M2M のユースケースにおいては，1〜10 リクエスト/秒程度が期待されているため，家庭内デバイスをとりまとめる HGW Server に数十のセンサがつながることを考えると今後センサが増加してもリクエスト処理頻度として，100〜200 リクエスト/秒が実現できることが目標になる．

**(5) 低パフォーマンス環境での動作**

M2M アプリケーションの事例では，人間が入っていけないような危険な場所や家の中のように非常に限られた場所にあるデバイスを対象とする場合がある．このような場所では，比較的パフォーマンスの低い計算機を使用して，デバイスを管理していることが多い．

たとえば，ホーム ICT では，モバイル端末のような HGW Server を使用している [8]．様々な M2M アプリケーションの事例に対応するためには，限られた場所でデバイスを管理するために使用されている HGW Server のような比較的パフォーマンスの低い計算機でデバイスを管理する必要がある．Kuroda らによる OSGi（Open Services Gateway initiative）ベースの HGW における評価では，CPU が 3.30 GHz，メモリが 2 GB（RAM）の PC を利用しているが，OSGi でのメモリ消費量とは別に提案している手法での総メモリ消費量が 30 MB 程度である [9]．このことから，同程度のスペックの HGW Server を用いて提案システムを動作させた場合，OSGi を含む総メモリ量が 30〜50 MB 程度を目標とする．

## 3. デバイスへの統一的アクセス手法

本章では，要求条件を満たすための技術の適応検討について述べ，提案システムおよびケーススタディについて述べる．

### 3.1 要素技術の適応検討

要求条件 (1)，(2)，(3) を満たすための要素技術の適応検討について述べる．なお，要求条件 (4) および (5) を満たしているか確認するためには，提案システムを実装し，評価する必要がある．

#### 3.1.1 REST アーキテクチャの適応検討

要求条件 (1) と (3) を満たすために，REST（Representational State Transfer）[10] の利用を検討する．

REST とは，Web サービスのような分散システムにおいて複数のソフトウェアを連携させるのに適した設計方法のことである [10], [11]．近年では，Twitter [12] や DropBox [13] といった Web サービスにアクセスするための API として活用されている．

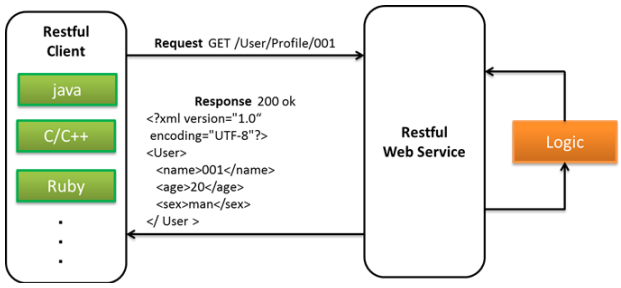


**図 2** REST アーキテクチャ
**Fig. 2** REST architecture.

REST は，HTTP の GET/POST/PUT/DELETE などのような CRUD（Create/Read/Update/Delete）に結び付けることができるメソッドでリソースを操作することによって，そのリソースの状態を取得する通信設計を行う．

**図 2** は，特定の人物の情報を得るための REST アーキテクチャ例である．Client は，「User/Profile/001」というリソースに対して GET リクエストを送信し，「001」という人物の情報を得ている．

上記の考えを適用し，デバイスを REST サービス化することができれば，リソースを参照するだけで，デバイスにどのような機能があるのかを知ることができ，デバイスの役割を定義しやすくなるといった利点がある．これにより，様々なデバイスの操作・制御を共通的に実現できる．REST による通信は非常にシンプルであり，HTTP の実装を行うだけで，容易に開発することが可能である．固定されたメソッドのみを使用するため，どのようなデバイスを使用しても共通の方法で操作・制御可能であることから，開発効率の向上が期待できる．

#### 3.1.2 OSGi の適応検討

要求条件 (2) を満たすために OSGi（Open Services Gateway initiative）[8] の利用を検討する．

OSGi とは，Java モジュールの動的追加や実行を管理するための基盤システムの仕様である．**図 3** のように Bundle と呼ばれるモジュールを動的に追加や削除することが可能である．また，Equinox [14]，Apache Felix [15]，Knoplersh [16] などといったオープンソースの実装があり，誰でも使用することが可能である．さらに OSGi は，ホーム ICT として家電やセンサネットワークなどを管理するためにも使用されている．

デバイスの REST サービスを Bundle 化することができれば，REST サービスのみを必要に応じて追加や除去することが可能になる．REST による通信は，固定されたメソッドのみで行うため，デバイスの仕様変更により，REST サービスが追加や除去されても M2M アプリケーションに与える影響は小さいと考えられ，仕様変更に追随できると考えられる．

© 2015 Information Processing Society of Japan

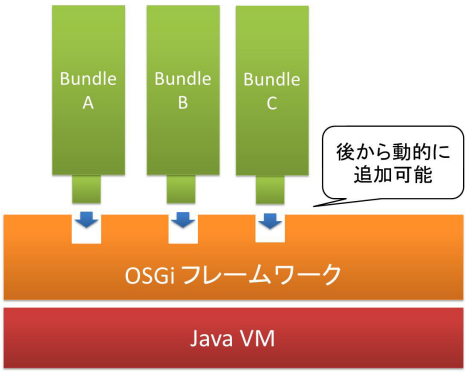


図 3 OSGi の概要
**Fig. 3** Mechanism of OSGi framework.

表 1 アノテーション例
**Table 1** Example of annotations.

| アノテーション | 概要 |
|---|---|
| @GET/POST/PUT/DELETE | リソースに対する操作を設定 |
| @PATH | リソースパスを設定 |
| @ENCODED | エンコードタイプを設定 |
| @PRODUCES | レスポンスの形式を設定 |
| @CONSUMES | ユーザから送られてくるリクエストに対するパラメータ形式の設定 |
| @FORMPARAM | リクエストに対するパラメータを取得 |

### 3.1.3 JAX-RS の適応検討

REST アーキテクチャと OSGi を組み合わせて，要求条件 (1)，(2)，(3) を満たすために JAX-RS（Java API for RESTful Web Services）の利用を検討する．REST サービスを実現するための一手段として，Java サーブレットが利用されていたが，JAX-RS アノテーションを利用することで，M2M アプリケーション開発者はリソースとデータ・オブジェクトに集中できるようになり，サーブレットを用いて通信レイヤーを作成する必要もなくなる．つまり，JAX-RS を利用することで，Java のクラスやメソッドに対して，**表 1** に示すアノテーションを利用することができ，容易に REST サービス化することが可能である．

**図 4** は，温度センサがあった場合に JAX-RS で REST サービスを作成した例である．たとえば，Device/Sensor/Temperature リソースを method（ObjectGetSensorValue）で操作する．具体的には，リソースに対して GET リクエストを送るとセンサから温度を取得することができる．PUT と DELETE リクエストの場合，センサの ON/OFF に対応付けすることが可能である．POST リクエストの場合，センサの設定（取得周期など）を変更する操作に対応付けることが可能である．このように JAX-RS を用いることで簡単にデバイスを REST サービス化できる．

```
@PATH ( "Device/Sensor/" )
@ENCODED ( CCEncoded.UTF8 )
public class CCSensorRestService
{
    @GET
    @PRODUCES ( CCMediaType.XML )
    @PATH ( "temperature" )
    public String GetTemperature () ;

    @POST
    @PRODUCES ( CCMediaType.XML )
    @CONSUMES ( CCMediaType.XML )
    @PATH ( "temperature" )
    public String SetupTemperature (
        @FORMPARAM ( " usr_input " ) String strInputValue ) ;

    @PUT
    @PRODUCES ( CCMediaType.XML )
    @PATH ( "temperature" )
    public String TemperatureSwitchON () ;

    @DELETE
    @PRODUCES ( CCMediaType.XML )
    @PATH ( "temperature" )
    public String TemperatureSwitchOFF () ;
}
```

図 4 JAX-RS による REST サービス例
**Fig. 4** REST service using JAX-RS.

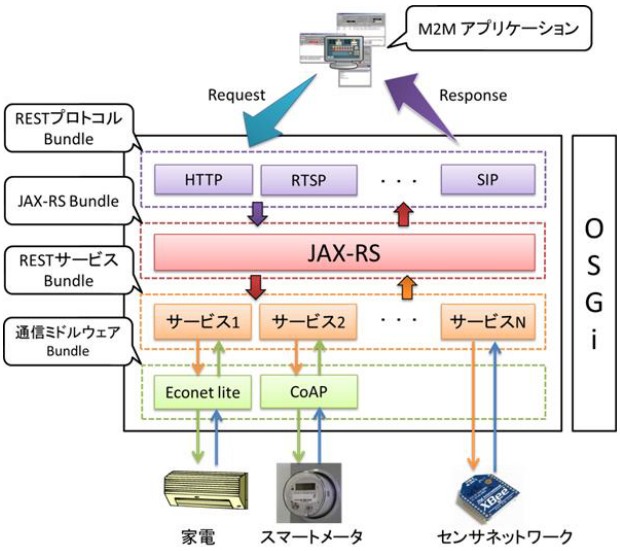


図 5 提案システム
**Fig. 5** Proposed system.

### 3.2 提案システム

提案システムでは，デバイスへの統一的なアクセス手法を提供するため，3.1 節で述べたとおり，OSGi に REST アーキテクチャでアクセスできる仕組みを JAX-RS を利用して実現することが特徴である．具体的には，既存の OSGi でデバイス管理などを行っている Bundle を REST により操作可能にする．Bundle として REST アーキテクチャを実装し，OSGi の種類を問わず，Bundle を REST により操作可能であることを目指す．

**図 5** は，提案システムのイメージを示したものである．

© 2015 Information Processing Society of Japan

提案システムは，REST プロトコル Bundle，JAX-RS Bundle，REST サービス Bundle，通信ミドルウェア Bundle で構成される．提案システムでは，M2M アプリケーション開発者が自由に REST 対応のプロトコルを選択でき，REST サービスやデバイス固有の処理を自由に追加/除去可能にするために，個々の処理を Bundle として実装する．

**(1) REST プロトコル Bundle**

REST アーキテクチャに対応することができるプロトコルを処理する Bundle である．REST に対応できるプロトコルには，HTTP や RTSP（Real Time Streaming Protocol），SIP（Session Initiation Protocol）などがある．プロトコルを Bundle 化することにより，ユーザは必要に応じて上記プロトコルを選択可能である．

**(2) JAX-RS Bundle**

REST プロトコル Bundle からの内容をもとに REST サービス Bundle を呼び出すための Bundle である．REST プロトコル Bundle と REST サービス Bundle を連携させるために 1 つの Bundle として実装する．また，REST サービスの動的な着脱を可能とする．

**(3) REST サービス Bundle**

JAX-RS を利用して作成された REST サービスクラスを Bundle 化したものである．REST サービスクラスを Bundle 化することで，ユーザはデバイスごとにサービスを記述できる．REST サービス Bundle は，JAX-RS Bundle から呼び出され，サービス内容に応じて適切な通信ミドルウェア Bundle を呼び出す役目がある．

**(4) 通信ミドルウェア Bundle**

この Bundle は，デバイス固有の処理をまとめた Bundle であり，REST サービス Bundle によって呼び出される．これらの Bundle は，デバイスやセンサを提供するデバイス提供者によって用意されることを想定している．

### 3.2.1 Bundle 間通信

各々の Bundle は，OSGi の Bundle 間通信によって接続される．Bundle 間通信とは，各 Bundle が提供する機能を OSGi が持つ機能管理表に登録しておき，この表を参照することで，各 Bundle は他の Bundle が持つ機能を利用することができる．また OSGi には，新しい機能が登録されたとき，各 Bundle に通知する機能もある．

### 3.2.2 M2M アプリケーションからのリクエスト時の動作

Bundle 間通信により，各 Bundle 間の接続が完了した場合，M2M アプリケーションからのリクエストをもとに JAX-RS を使用して，デバイスの REST サービスを呼び出すことが可能である．

**図 6** は，リクエスト時の動作シーケンスを示したものであり，以下はその流れである．

(1) M2M アプリケーションが HTTP リクエストを送信．
(2) HTTP Bundle が JAX-RS Bundle に HTTP リクエストを送信．

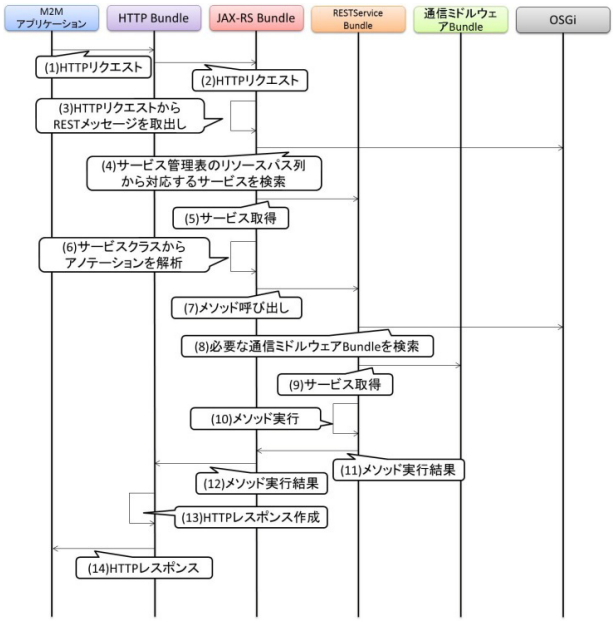


**図 6** リクエスト時の動作シーケンス図
**Fig. 6** Sequence for request.

(3) HTTP リクエストから REST メッセージ（HTTP メソッドやリソースパスなど）を取得．
(4) リソースパスを使用して OSGi のサービス管理表から REST サービス Bundle を検索．
(5) REST サービス Bundle 内に存在する JAX-RS によって作成された REST サービスクラスを取得．
(6) (3) の REST メッセージを使用して，取得した REST サービスクラスのメソッドに付加されたアノテーションを解析して実行すべきメソッドを取得．もし実行すべきメソッドが見つからなかった場合，レスポンスコード 404 を M2M アプリケーションに返答．
(7) 取得した REST サービスクラスのメソッドを JAX-RS Bundle が実行．
(8) REST サービスクラスのメソッドを実行する際に必要となる通信ミドルウェア Bundle を OSGi から検索．
(9) REST サービスクラスのメソッドを実行するために必要な通信ミドルウェア Bundle のメソッドを取得．
(10) 通信ミドルウェア Bundle のメソッドを実行．
(11) 通信ミドルウェア Bundle のメソッド実行結果を JAX-RS Bundle で実行中の REST サービスクラスのメソッドが受信．これにより，REST サービスクラスのメソッドの実行が完了．
(12) JAX-RS Bundle が HTTP Bundle に REST サービスクラスのメソッド実行結果を送信．
(13) REST サービスクラスのメソッド実行結果をもとに HTTP レスポンスを作成．
(14) M2M アプリケーションに HTTP レスポンスを返信．

© 2015 Information Processing Society of Japan

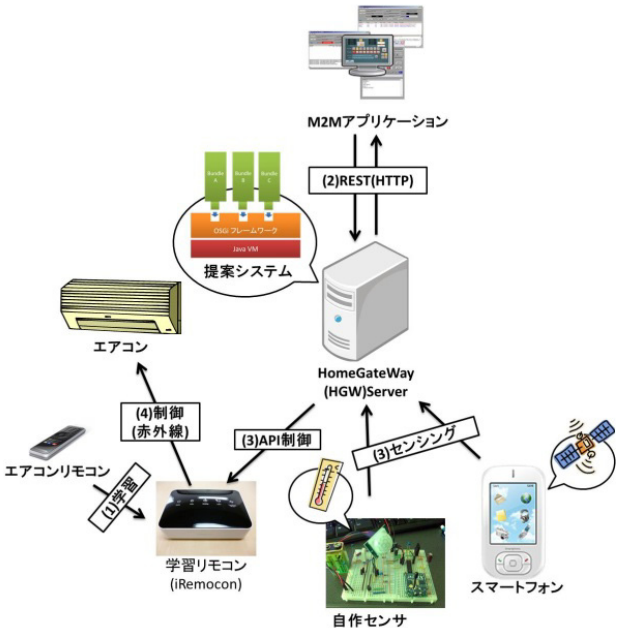


図 7 ケーススタディ（エアコンの自動制御）
**Fig. 7** Case study.

### 3.3 ケーススタディ

提案システムの動作検証と提案システムが要求条件を満たしているか評価するため，家庭用エアコンの自動制御をケーススタディとして実装した．

使用デバイスとして，学習リモコンに iRemocon [17] を使用した．iRemocon とは，赤外線リモコンを学習し，赤外線を発信することができるデバイスである．文字列ベースのコマンド API を用いて Ethernet 経由で操作可能である．次に，温度/照度/湿度が取得できるセンサを自作した．このセンサは，Zigbee 経由で HGW Server にデータを送信する．HGW Server 側に接続する Zigbee モジュールは，RS232C で接続する．なお現在，IETF で策定中の M2M 専用プロトコルである CoAP（Constrained Application Protocol）[18] で操作できるように実装した．さらに GPS や加速度といったコンテキスト情報をエアコン制御に利用するために，スマートフォンも使用した．

ケーススタディでは，JAX-RS を使用して，これらのデバイス用の REST サービスを作成し，OSGi 上で動作する Bundle とした．これにより，M2M アプリケーションからの REST リクエストによる各デバイスの制御が可能である．

**図 7** は，エアコンの自動制御例を示したものであり，以下は，その流れである．

(1) 学習リモコンに家庭用エアコンリモコンを学習させる．
(2) M2M アプリケーションが REST リクエストを提案システムに送信する．
(3) (2) のリクエスト内容をもとにセンシング要求や学習リモコンに制御命令を発行する．
(4) 学習リモコンにより，エアコンが制御される．

## 4. 評価

本章では，提案システムの有用性を検証する．2 章で述べた要求条件を満たすかを確認する評価項目を示し，その結果について述べる．

### 4.1 評価項目

ケーススタディの M2M アプリケーションをもとに，要求条件を満たしているかの評価を行う．

要求条件 (1) を満たすか確認するためにデバイスのリソース定義を行い，様々なデバイスの操作・制御が M2M アプリケーションから可能であるか，確認する．

要求条件 (2) を満たすか確認するために REST と他の技術の比較を行い，デバイスの追加や除去が起こった場合に発生する仕様変更がアプリケーションに与える影響を評価する．

要求条件 (3) を満たすか確認するために，提案システムを介する場合と介さない場合でケーススタディのソースコード量を比較し，開発効率について評価する．

要求条件 (4) を満たすか確認するために，提案システムを介することで発生する通信ボトルネックが M2M アプリケーションの動作に影響を与えないか評価する．

要求条件 (5) を満たすか確認するため，通信ボトルネック調査と提案システムが動作している計算機のパフォーマンスを評価する．

### 4.2 リソース定義の評価

M2M アプリケーションからデバイスを簡単に利用する手法として，デバイスの REST サービス化を提案した．これにより，統一的な操作・制御を実現される．REST サービスを作成するためには，まずデバイスが持っている機能を抽出し，リソースの定義を行う必要がある．そこで，ケーススタディで使用したデバイスのリソース定義を行った．

実際には，デバイスドライバをデバイス提供者が用意する現状と同様に，リソース定義や通信ミドルウェア Bundle の作成はデバイスの機能や仕組みを熟知したデバイス提供者が行うことが開発効率が良いと考える．デバイス提供者は，M2M アプリケーション開発者に自身のデバイスを利用してもらうためにリソースの定義や Bundle の作成を行うことで利用しやすいデバイスとしてアピールできる．

M2M アプリケーションの開発者は，どのようなリソース定義が存在するか知る必要があるため，HEADER メソッドを使用して，「http://127.0.0.1/rest/services」にリクエストを送信する．これにより，定義されているリソースの一覧を取得することが可能であるものとする．

**表 2** は，デバイスのリソース定義例である．リソースを定義しやすい仕組みとして，表 2 のようにメソッドとリソースを定義するのみの実現方式を提供している．この

© 2015 Information Processing Society of Japan

表 2 リソース定義例
**Table 2** Definitions of temperature sensor resources.

| メソッド | リソース | 概要 |
|---|---|---|
| GET | /sensor/temperatrue | 温度取得 |
| PUT | /sensor/temperatrue | 温度センサ ON |
| DELETE | /sensor/temperatrue | 温度センサ OFF |
| PUT | /iRemocon | iRemocon ON |
| DELETE | /iRemocon | iRemocon OFF |
| GET | /iRemocon/signal/{Idx} | 指定された番号の信号を発信 |
| PUT | /iRemocon/signal/{Idx} | 指定された番号に信号を学習 |
| DELETE | /iRemocon/signal/{Idx} | 指定された番号の信号を削除 |
| GET | /android/gps | 緯度経度取得 |
| PUT | /android/gps | GPSON |
| DELETE | /android/gps | GPSOFF |
| POST | /android/gps/distance/{int} | 何 m 変化すると通知するか設定 |

定義に基づき，JAX-RS で REST サービスを作成することが可能である．このように，リソース定義さえできていれば，提案システムでデバイスを利用可能である．ケーススタディの M2M アプリケーションから定義されているリソース 1 つ 1 つにリクエストを送信したところ，適切な REST サービスが実行され，iRemocon を介してエアコンの操作・制御が可能であった．

### 4.3 仕様変更時の影響評価

REST アーキテクチャは，RPC（Remote Procedure Call）や SOAP（Simple Object Access Protocol）とよく比較される．RPC や SOAP では，サービス提供側が独自に定義したインタフェースをクライアントに提供することで，サービスが利用可能となる．**図 8** は，RPC の例である．クライアント側スタブで Profile メソッドを実行するとサーバに引数が送信される．サーバ側スタブは，引数を受信すると，対応するメソッドを実行してクライアント側スタブに返信し，あたかもクライアントで Profile メソッドが実行されたかのように見える仕組みである．

このとき，サーバ側スタブでメソッドの変更があった場合，クライアント側スタブのメソッドも変更する必要がある．たとえば，Profile メソッドの名前を GetUesrData などに変更した場合，クライアントも変更する必要がある．

SOAP についても RPC と同様に，クライアントに変更が生じる．

REST の場合，GET/POST/PUT/DETELE などの固定されたメソッドを使用するため，RPC や SOAP よりもクライアントに与える影響は少ない．

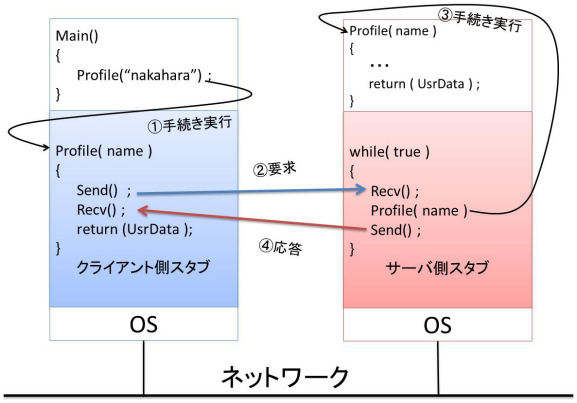


図 8 RPC の概要
**Fig. 8** Remote procedure call.

表 3 提案システムを介す場合のコード割合
**Table 3** Percentage of necessary code for the proposed system.

| 概要 | 割合 |
|---|---|
| REST リクエスト送信処理(HTTP 通信) | 5.3% |
| REST レスポンス解析処理(JSON 解析) | 6.8% |

また，OSGi 上で処理を管理しているため，仕様変更にも柔軟に対処することが可能である．たとえば，自作センサ操作用の CoAP 作成/解析処理に変更があったと仮定する．提案システムを介さない場合，自作センサを利用しているすべての M2M アプリケーションの修正が必要となる．しかし，提案システムを介していた場合，REST サービスに合わせて処理 Bundle を修正することで，すべての M2M アプリケーションに修正を反映させることが可能である．したがって，仕様変更時でも M2M アプリケーションに与える影響を小さくすることができる．

### 4.4 開発効率の評価

M2M アプリケーション開発の容易性を評価するために，以下ではケーススタディとして実装した M2M アプリケーションの総コード量（総ライン数）をもとに割合で示す．なおコードは，一般的なコーディング規約をもとに Java で作成した．

まず，ケーススタディとして実装した M2M アプリケーションのコードをもとに，提案システムを介する場合に必要となるコードと介さない場合でも必要となるコードに分類した．**表 3** に示すとおり，提案システムを介する場合にのみ必要となるコード量の割合はこれら 2 つの和で全体の約 12%となる．残りの約 88%のコードは，提案システムを介さない場合にも必要となるコードであり，それぞれのコード割合を**表 4** に示している．新しいデバイスを使用するたびに通信およびコマンド作成/解析などの処理を追加する可能性がある．

また，使用するデバイスによってコード量も変化する．たとえば，表 4 に示す自作センサ操作用の CoAP 作成/解

© 2015 Information Processing Society of Japan

表 4 提案システムを介さない場合のコード割合
Table 4 Percentage of necessary code commonly required.

| 概要 | 割合 |
|---|---|
| iRemocon 用の通信処理 | 13.2% |
| iRemocon 用のコマンド作成/解析処理 | 9.9% |
| 自作センサ用の通信処理 | 4.1% |
| 自作センサ操作用の CoAP 作成/解析処理 | 40.1% |
| スマートフォンとの通信処理 | 20.6% |

析処理は，他の処理に比べてコード量が多くなっている．これは他の処理と比較して複雑な処理であることを示している．このように使用するデバイスによっては，処理が複雑なものがあり，M2M アプリケーションが肥大化する可能性があることを考慮しなければならない．

提案システムでは，表 4 に示す各デバイスへの通信およびコマンドや CoAP 作成/解析処理は，OSGi 上に Bundle として作成した．これにより，表 3 に示す OSGi に対して REST リクエストを送信する処理と REST レスポンスを解析する処理が必要となった．しかし，表 3 の REST リクエスト送信処理や REST レスポンス解析処理を再利用することにより，新たなデバイスを追加してもデバイスごとに固有の通信方式を実装する必要はない．つまり，Bundle の再利用により M2M アプリケーションへの影響を抑えることができる．しかし，REST レスポンス解析処理は，レスポンスの形式により，コード量が肥大化する可能性がある．そこで，今回作成した REST サービスは，JAX-RS の JSON Parser を使用して単純な JSON 形式に変換したものをレスポンスとしている．したがって，JAX-RS の JSON Parser を使用する場合には，共通の JSON 形式を解析することになるため，REST レスポンス解析処理を再利用することが可能である．

M2M アプリケーションには，表 3 に示す処理のみを実装すればよく，コードのシンプル化やコード量の削減ができ，効率的な開発が実現できた．また，表 4 に示す処理は M2M アプリケーションに依存せず，共通的に利用することが可能であるため，試験工程の短縮や再利用による開発効率の向上に寄与することができた．

### 4.5 通信ボトルネックの評価

提案システムを介すことで発生する通信が M2M アプリケーションの動作に影響しないか評価する．そのために，JMeter を使用して提案システムに対して 1 時間おきに 1 秒間で処理しなければならないリクエストを 100 ずつ増加させながら HTTP リクエストを 5 時間送信した．なお，実験環境は**表 5** に示したとおりであり，**図 9** は，通信ボトルネックの評価結果である．これによれば，100～400 リクエスト/秒のとき平均応答時間は，約 10～20 ms 程度だった．しかし，500 リクエスト/秒のとき，平均応答時間は，約 200 ms 程度となった．

表 5 実験環境
Table 5 Experiment environment.

| | |
|---|---|
| CPU | Celeron(1.2GHz) |
| メモリ | 4GB(OSGi のヒープサイズは 1GB) |
| OSGi | Apache Felix |

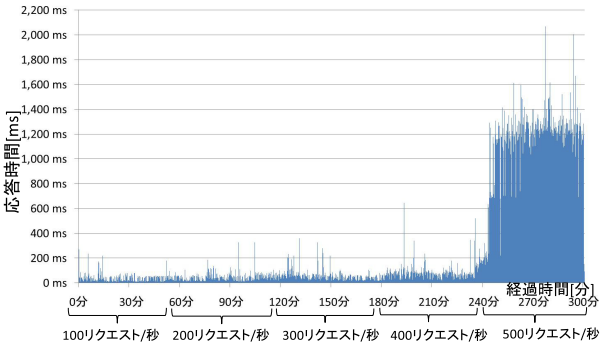


図 9 通信ボトルネックテスト結果
Fig. 9 Result of bottleneck test.

加えて，平均応答時間以上の値の平均値を平均遅延時間として，遅延の調査も行った．その結果，100～400 リクエスト/秒のとき平均遅延時間は 80～100 ms 程度だった．しかし，500 リクエスト/秒のとき，平均遅延時間は，1,200 ms 程度かかり，リクエストの失敗も見られたため，実験環境のスペックでの限界であると考えられる．

以上により，リアルタイム性の実現として目標としたリクエスト処理頻度は 100～200 リクエスト/秒を満たしていることが確認できた．また，平均応答時間は，8～10 ms 程度であり，提案システムを介しても十分な処理が可能であることが確認できた．

### 4.6 パフォーマンスの評価

提案システムが，パフォーマンスが低い計算機上でも動作するか確認するために，通信ボトルネックテストと同時に JConsole を使用して提案システムの CPU 使用率とメモリ使用量の調査を行った．

**図 10**，**図 11** は，調査結果である．これによれば，CPU 使用率はリクエスト数が増加するたびに，段階的に上昇しているが最大でも 60～72%程度であった．

しかし，メモリ使用量については，100～300 リクエスト/秒では，平均で 50 MB 程度であったが，400 リクエスト/秒では 350 MB 程度使用した．さらに 500 リクエスト/秒では，大幅にガベージコレクションが動作する結果となった．この原因として，OSGi のヒープサイズが不足していたことがあげられる．図 11 に示されるようにリクエストが頻発した場合，ヒープサイズが不足しており，新しいインスタンスが作成できず，応答できないことから通信

© 2015 Information Processing Society of Japan

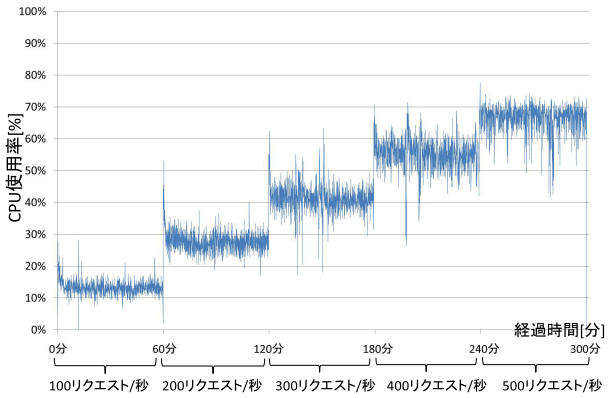


図 10 CPU 使用率
**Fig. 10** Result of CPU utilization.

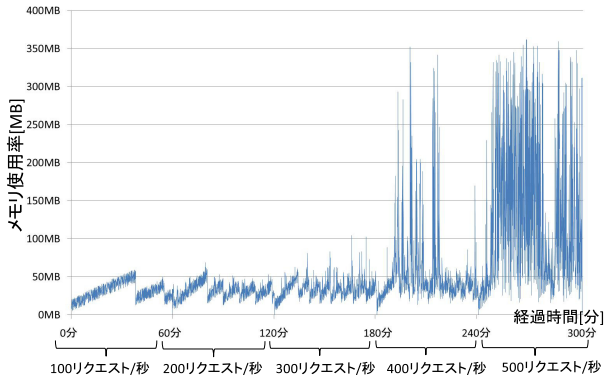


図 11 メモリ使用率
**Fig. 11** Result of memory usage.

ボトルネックとなっていると考えられる．

以上により，低パフォーマンス環境では，リクエスト処理頻度 100〜300 リクエスト/秒において，目標値とした総メモリ量が 30〜50 MB 程度を満たしている．このことから，HGW Server のような低パフォーマンス環境での動作に関し，十分な処理が可能であることが確認できた．

## 5. 関連研究

本章では，既存のデバイス制御方法や M2M プラットフォームについて関連の深い研究を 5 件抽出し，本研究との差異を示す．

1 件目は，ホームコンピューティングミドルウェアの 1 つである SENCHA（Smart Environment for Controlling Home Appliances）[19] で REST アーキテクチャを実装してデバイスを制御する研究が行われている．SENCHA では，ユーザが持っているパーソナル端末にゲートウェイを設置し，同様に SENCHA が導入されている周辺デバイスを SOAP で操作する．このとき，SENCHA を REST アーキテクチャで操作できるようにすることで，デバイスを操作するアプリケーションを容易に開発可能である．しかし，あるデバイスの情報をもとに別のデバイスを操作するアプリケーションを開発する場合，イベント通知接続を利用しなければならない．イベント通知接続とは，デバイスが指定した状態のとき，自動的に別のデバイスに制御リクエストを送信する仕組みである．そのため，デバイスごとにイベント通知を行う REST サービスを実装する必要があり，複数のデバイスを使用する M2M アプリケーションの開発への適用は難しい．

2 件目は，Android 端末に OSGi を実装し，センサを Bundle 化することで M2M アプリケーションの開発を容易にするためのプラットフォームの研究である [20]．この研究では，Bundle を追加や除去することで，必要に応じてセンシングするデータを選択することが可能である．センシングデータは，XMPP（eXtensible Messaging and Presence Protocol）と呼ばれるプレゼンスプロトコルでセンシングデータを収集する計算機に送信される．センシングデータを収集する計算機にも OSGi が実装されており，センシングデータを解析する処理を Bundle として提供している．M2M アプリケーション開発には，XMPP および OSGi のバンドル開発の両方に関する知識が必要であり，実現環境としても Android 端末とセンシングデータを収集する計算機の両方に OSGi を準備する必要がある．

3 件目の研究は，Open MTC というミドルウェアを用いて構築されたネットワーク上で管理されているデバイスに対して，REST アーキテクチャで制御するための API を提案している [21]．この研究では，REST アーキテクチャを提供するために Fokus Broker と呼ばれるソフトウェアを使用している．Fokus Broker は，アプリケーションと OpenMTC により構築されたネットワークの仲介役として必要である．また，提案されている API は，OpenMTC で構築されたネットワークのみでしか使用することができない．

4 件目の研究は，SOA アーキテクチャである Service Mix を用いて，デバイスを制御するためのサービスを作成し，REST や SOAP などでアクセスできるプラットフォームを提案している [22]．Service Mix には，OSGi が組み込まれており，ユーザが新しいサービスの追加や除去を行うことを助けることができる．一方，一般的にパフォーマンスが低い HGW Server は，Dalvik VM [23] のような制限された環境の上で動作するように最適化された OSGi を利用している．しかし，Service Mix に組み込まれた OSGi は，Dalvik VM のような環境で動作するように最適化されていない．

5 件目の研究は，EPC（Electronic Product Code）と呼ばれる一意に識別可能なコードを物理センサに付加し，HTTP を使用してセンサからデータを取得することが可能なプラットフォームを提案している [24]．この研究では，EPC と HTTP を利用して複数の物理的なセンサを組み合わせた仮想センサを構築することが可能である．また，仮想センサは，物理センサと同様に EPC が割り当てられ，

© 2015 Information Processing Society of Japan

HTTP でセンシングデータの取得を行うことが可能である．しかし，センシングデータは，1 分に 1 度しか取得することができないことから，秒オーダで情報を取得しなければならない M2M アプリケーション事例への適用は難しい．

## 6. おわりに

本研究では，M2M アプリケーションの開発を容易にすることを目的にしている．そのために，OSGi 上に JAX-RS を実装し，デバイス用 REST サービスを構築し，M2M アプリケーションから容易にデバイスを利用可能にした．これにより，M2M アプリケーションは，REST を利用することでデバイスの操作が可能で，直接デバイスの操作処理を実装するよりも，ソースコード量を抑えることができることを示した．また，提案システムを介した M2M アプリケーションの評価結果より，リクエスト処理頻度 100〜200 リクエスト/秒のとき，応答時間は 8〜10 ms 程度であり，リアルタイムの実現に十分な処理能力を確認できた．さらに，HGW Server のようなパフォーマンスの低い計算機上でも動作することを示した．

しかし，本研究では M2M アプリケーションの開発を容易にすることを優先させて評価したため，認証機能については検討していない．そのため，HGW Server のあるネットワーク上から誰でもアクセス可能であるが，M2M アプリケーション事例によっては，アクセスを制限したい場合もある．今後は，認証機能について検討する予定である．

### 参考文献


[1] いよいよ本格化する Internet of Things，CISCO in Spire ISSUE10 (Jan. 2013), 入手先 ⟨http://cisco-inspire.jp/issues/0010/coverstory.html⟩.

[2] Boswarthick, D., Elloumi, O. and Hersent, O.: *M2M Communications: A Systems Approach*, WILEY (2012).

[3] 藤田隆史，後藤良則，小池　新：M2M アーキテクチャと技術課題，電子情報通信学会誌，Vol.96, No.5 (2013).

[4] ETSI TR 102 898 - Machine-to-Machine communications (M2M); Use cases of Automotive Applications in M2M capable networks v1.1.1 (2013-04).

[5] ETSI TR 102 691 - Machine-to-Machine communications (M2M); Smart Metering Use Cases v1.1.1 (2010-05).

[6] 増尾　剛，中村二朗，松岡茂登ほか：リアルタイム Web 技術による HEMS サービスクラウド化の検討，電子情報通信学会技術報告，Vol.112, No.350, NS2012-117, pp.1–6 (2012).

[7] AT&T: Digital Life (2014), available from ⟨http://www.att.com/shop/digital-life.html⟩.

[8] 勝田光弘：OSGi の概要と最新情報 ホームゲートウェイ（HGW）における最新動向と適用例，Java Developer Workshop - Client platforms (2011).

[9] Kuroda, Y., Yamasaki, I., Kondo, S., Katayama, Y. and Mizuno, O.: A memory isolation method for OSGi-based home gateways, *CBSE '14 Proc. 17th International ACM Sigsoft Symposium on Component-based Software Engineering*, pp.117–122 (2014).

[10] 山本陽平：Web を支える技術-HTTP，URI，HTML，そして REST，技術評論社 (2010).

[11] Leonard Ricbardson and Sam Ruby: RESTful Web サービス，オライリージャパン (2007).

[12] Twitter (2013), available from ⟨https://twitter.com/⟩.

[13] Dropbox (2013), available from ⟨https://www.dropbox.com/⟩.

[14] Equinox (2013), available from ⟨http://www.eclipse.org/equinox/⟩.

[15] Apache Felix (2013), available from ⟨https://felix.apache.org/⟩.

[16] Knopflerfish (2013), available from ⟨http://www.knopflerfish.org/⟩.

[17] 株式会社グラモ：iRemocon (2013)，入手先 ⟨http://i-remocon.com/⟩.

[18] Constrained Application Protocol (2013), available from ⟨http://tools.ietf.org/html/draft-shelby-core-coap-01⟩.

[19] Ishikawa, H., Ogata, Y., Adachi, K., et al.: Building Smart Appliance Integration Middleware on the OSGi Framework, *Object-Oriented Real-Time Distributed Computing, IEEE International Symposium*, pp.139–146 (2004).

[20] Kuna, M., Kolaric, H., Bojic, I., et al.: Android/OSGi-based Machine-to-Machine Context-Aware System, *Proc. 11th International Conference on Telecommunications*, pp.95–102 (2011).

[21] Elmangoush, A., Magedanz, T., Blotny, A., et al.: Design of RESTful APIs for M2M Services, *16th International Conference on Intelligence in Next Generation Networks*, pp.50–56 (2012).

[22] Busemann, C., Gazis, V., Gold, R., et al.: Enabling the Usage of Sensor Networks with Service-Oriented Architectures, *7th International Workshop on Middleware Tools* (2012).

[23] Dalvik: how Google routed around Sun's IP-based licensing restrictions on Java ME, available from ⟨http://www.betaversion.org/~stefano/linotype/news/110/⟩ (accessed 2014-07).

[24] 小澤みゆき，三次　仁：物理/仮想統合センサメッセージングシステム，電子情報通信学会技術報告，Vol.113, No.360, NS2013-156, pp.125–131 (2013).



**中原 祥吾**

2014 電気通信大学大学院博士前期課程修了．以来，RTOS のスケジューリング方式に関する研究に従事．


© 2015 Information Processing Society of Japan

**末田 欣子** （正会員）

1995 東京電機大学大学院博士前期課程修了．同年日本電信電話株式会社入社．以来，ネットワークソフトウェア，ネットワーク基盤技術の研究に従事．現在，同社ネットワーク基盤技術研究所主任研究員．電気通信大学客員准教授．博士（工学）．電子情報通信学会会員．

**多田 好克** （正会員）

1985 年東京大学大学院工学系研究科情報工学専門課程博士課程修了．工学博士．同年電気通信大学電子情報学科着任．1992 年より電気通信大学大学院情報システム学研究科．並列・分散システムの記述法に興味を持ち，オペレーティングシステムをはじめとするシステムソフトウェアの実現法に関する研究に従事．ACM，電子情報通信学会各会員．

© 2015 Information Processing Society of Japan