ゲートウェイクラウドを用いたサービス指向 IoT アーキテクチャの提案

2012SE029 濱野 真伍　2012SE128 久間 一輝
指導教員 青山 幹雄

1　研究背景と目的

近年，IoT の進展によって，インターネットに接続されるデバイスの数が増加している．それに伴い，既存のデバイスの制御と新規デバイスの追加を管理する技術が求められる．これに対処するために oneM2M アーキテクチャなどが提案されている．しかし，膨大な数のデバイスを管理できるスケールアウト可能なアーキテクチャは提案されていない．本稿ではスケールアウト可能な IoT アーキテクチャを提案する．

2　研究課題

本稿では，以下の 2 点を研究課題とする．

(1) デバイスを動的に管理するスケールアウト可能なアーキテクチャ

IoT ではデバイスの数が多くそれらが常にネットワークに接続される訳ではない．デバイスが計測データを送信したい時に任意に接続でき，デバイスの接続台数に応じてスケールアウト可能なシステムが必要である．

(2) デバイスのデータをタイミングによらず取得できるアーキテクチャ

デバイスの状態に関係なくインフラサーバがデバイスのデータを非同期かつタイミングに関係なく受信でき，デバイスが計測データをインフラサーバの状態やタイミング関係なく送信できるアーキテクチャが必要である．

3　関連研究

(1)　IoT とは

IoT(Internet of Things)とは，識別できるデバイスがインターネットに接続され，ネットワーク通信を行うシステムである．IoT の進展により，インターネットに接続されるデバイスの数が 2012 年に約 87 億個であったのに対し，2020 年には 約 500 億個になるとも言われている[7].

(2)　oneM2M アーキテクチャ

1)　oneM2M 論理アーキテクチャ

oneM2M 論理アーキテクチャは，アプリケーションとミドルウェアの 2 階層で構成される[5,7]．また，主にゲートウェイとデバイスから構成されるフィールドドメインとサーバから構成されるインフラストラクチャドメインの 2 つに区分される．

2)　oneM2M 物理アーキテクチャ

oneM2M 物理アーキテクチャは，インフラサーバ，ミドルノード，デバイスノードの 3 層からなる[5,7]．しかし，スケールアウトの考慮はない．

(3)　Publish/Subscribe アーキテクチャ

Publish/Subscribe アーキテクチャとは，パブリッシャが特定のサブスクライバに対して非同期にメッセージを配信するメッセージアーキテクチャである[6].

(4)　クラウドコンピューティング

クラウドコンピューティングとは，リソースの共有プールに，ネットワーク経由でオンデマンドにアクセス可能なモデルである．

(5)　エッジ/フォグコンピューティング

クラウドコンピューティングの概念をネットワークのエッジにまで拡張し，クラウドとエンドデバイスの間にゲートウェイなどの処理を行うサーバを配置するアーキテクチャである．　エッジコンピューティングの一例として，フォグコンピューティング[1]がある．

4　アプローチ

本稿では oneM2M 物理アーキテクチャに基づき，図 1 に示すようにゲートウェイとインフラサーバの間にゲートウェイクラウドを導入することでスケールアウト可能なサービス指向に基づいた IoT アーキテクチャを提案する．

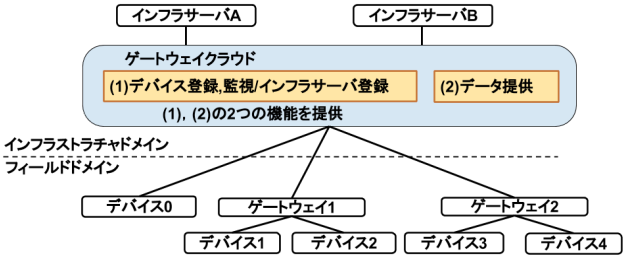

図 1 アプローチ図

ゲートウェイクラウドでは(1)デバイス登録，監視/インフラサーバ登録，(2)データ提供の 2 つの機能を標準インタフェースを介して提供する．

5　提案アーキテクチャ

本稿では，提案アーキテクチャを，論理アーキテクチャ，物理アーキテクチャ，配置アーキテクチャで定義する．

5.1　論理アーキテクチャ

論理アーキテクチャを図 2 に示す．デバイスノードではセンサデータの計測を行い，ゲートウェイクラウドに送信する．ゲートウェイクラウドノードでは，デバイス登録，監視/インフラサーバ登録，データ提供，計測データの保存を行う．インフラサーバノードではデータ受信と活用を行う．各機能の詳細を以下に定義する．

(1) データ計測機能：センサデータを計測する．
(2) 計測データ送信機能：計測したセンサデータを構造化しデータ送信機能へ送信する．
(3) 送信先分配機能：システムが分散する際の計測データの送信先を指定する．
(4) データ配信機能：計測データ送信機能から送信されたデータをサブスクリプションに合わせてデータ受信機能へ配信する．
(5) データ受信機能，データ登録機能：すべての計測データを受信し計測データベースへ登録する．
(6) 計測データベース：計測データを保存する．
(7) データ検索機能：データ活用機能からのリクエストに

合わせて計測データベースを検索する.

(8) データ受信機能:データ配信機能へサブスクリプションを送信し, 計測データを受信する.

(9) データ活用機能:サービスに応じて決定される機能.

各ノード間の通信は共通インタフェースを用いて行う.

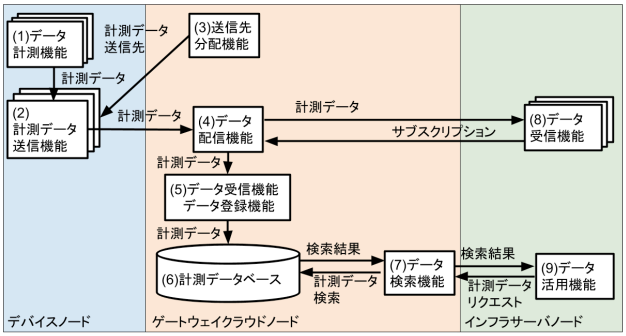

図 2 論理アーキテクチャ

5.2 プロセス定義

(1) デバイス登録プロセス

デバイス登録プロセスを図 3 に示す.

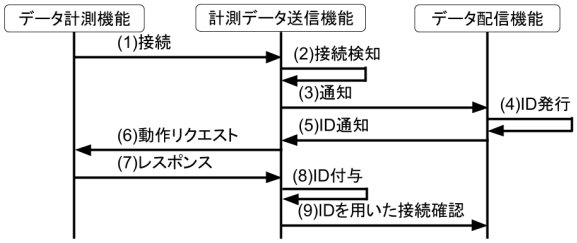

図 3 デバイス登録プロセス

データ計測機能がデータ送信機能に接続された際, データ配信機能に通知を行い, データ配信機能はデータ計測機能ごとにユニークな ID を発行し通知する. その後, 計測データ送信機能は通知された ID を用いてデータ配信機能への接続確認を行う.

(2) 受信データ決定プロセス

受信データ決定プロセスを図 4 に示す.

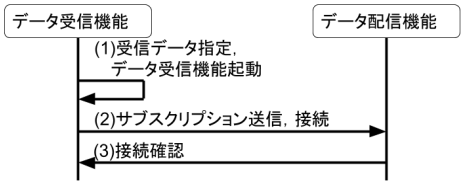

図 4 受信データ決定プロセス

データ受信機能は受信したいデータのコンテンツもしくはトピックを指定しデータ配信機能に対してサブスクリプションを送信し接続を行う. その後データ配信機能から接続確認をデータ受信機能に送信する.

1) コンテンツ指定:計測データの変数と値を範囲指定することでデータを指定する方法.
2) トピック指定:計測データに付与されるトピック情報を指定する方法. トピックは階層構造になっており"#"をワイルドカードとして用いることができる.

サブスクリプションを指定しなかった場合, データ配信機能に送信されるすべての計測データを受信する.

(3) 計測データ登録プロセス

計測データ登録プロセスを図 5 に示す.

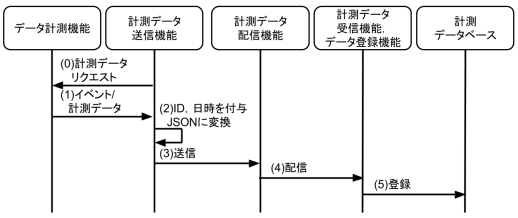

図 5 計測データ登録プロセス

計測データ送信機能からのリクエストもしくは, データ計測機能におけるイベントで発生した計測データを計測データ送信機能で ID や日時を付与し JSON 形式に構造化する. その後, 計測データ配信機能に計測データを送信し, 計測データ受信機能によって受信した計測データをデータ登録機能が計測データベースに登録する.

(4) データ配信プロセス

本稿ではデータ配信に REST の要求/応答を用いた配信と Publish/Subscribe ブローカを用いた配信の 2 つの方法を用いる. これにより, リアルタイムなデータ配信とタイミングに依存しないデータ配信を提供ができる.

1) REST の要求/応答を用いた配信

REST の要求/応答を用いた計測データ配信プロセスを図 6 に示す.

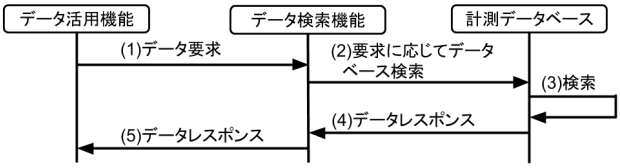

図 6 REST の要求/応答を用いた配信プロセス

データ活用機能はデータ検索機能にデータ要求を送信し, データ検索機能はその要求に従って, データベースを検索し, データ活用機能にデータをレスポンスとして送信する.

2) Publish/Subscribe ブローカを用いた配信

Publish/Subscribe ブローカを用いた計測データ配信プロセスを図 7 に示す.

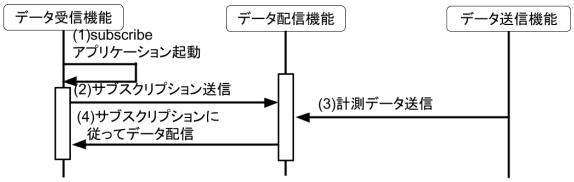

図 7 Publish/Subscribe ブローカを用いた配信プロセス

データ受信機能はサブスクリプションを送信した後ポートを開放し続けることでデータ配信機能からの計測データ受信が可能にする. データ配信機能はサブスクリプションに従って計測データを配信する.

(5) 接続先変更プロセス

接続先変更プロセスを図 8 に示す. 新しくデバイスが追加され, データ配信機能 A が処理能力や通信能力の制限で新しいデバイスのデータ送信機能が接続できない場合, 送信先分配機能はスケールアウトした他のデータ配信機能 B の接続先をデータ送信機能に通知する. データ送信機能は送信先をデータ配信機能 B に変更することで, 計測データの送信が可能になる. このプロセスによりゲートウェイクラウド全体のデバイス管理とスケールアウ

トが可能になる.

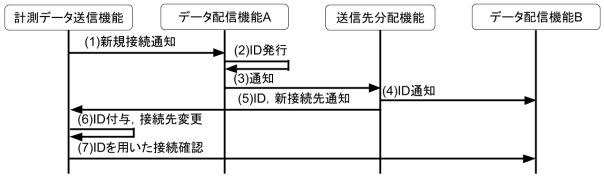

図 8 接続先変更プロセス

5.3 物理アーキテクチャ

(1) ゲートウェイクラウドを用いた拡張アーキテクチャ

拡張アーキテクチャを図 9 に示す.

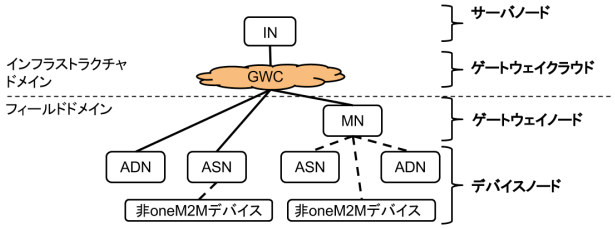

図 9 拡張アーキテクチャ

本稿では oneM2M 機能アーキテクチャの構成に基づき, ゲートウェイノードとサーバノードの間にゲートウェイクラウドを設置するアーキテクチャを提案する. 図 9 では Web の通信を実線で示し, LAN の通信を破線で示す.

(2) 物理アーキテクチャ

物理アーキテクチャを図 10 に示す.

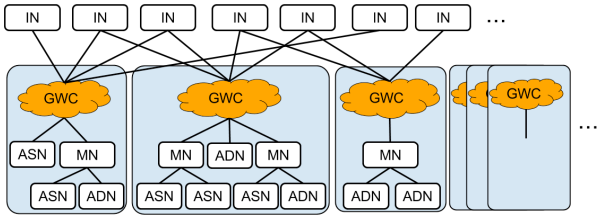

図 10 物理アーキテクチャ

本アーキテクチャは膨大な数のデバイスに対応するために, ゲートウェイクラウドをスケールアウトする方法を用いる. 1 つのゲートウェイクラウドに接続されるデバイス数はクラウドの処理能力によって異なる.

(3) 分散データベース

本アーキテクチャのデータベース配置を図 11 に示す.

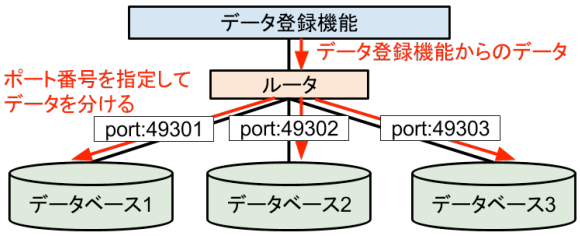

図 11 分散データベース配置

図 12 では例として, 3 つのデータベースへ分散する配置を示す. それぞれのデータベースへ計測データを保存するためにポート番号を指定する. データ登録機能から送信される各データに付随するキーによって分散配置される. 本データベースにデータを保存する際には, シャーディングとレプリケーションを組み合わせ, データの一貫性を保つ. また, データベースの数を増やしでスケールアウトすることで, 増加するデバイスの計測データを全て保存することができる.

5.4 配置アーキテクチャ

配置アーキテクチャを図 12 に示す.

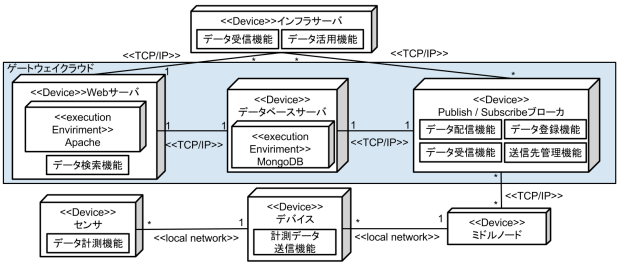

図 12 配置アーキテクチャ

ゲートウェイクラウドノードに Web サーバ, データベース, Publish/Subscribe ブローカを配置する. また, Publish/Subscribe アーキテクチャに基づき, 本アーキテクチャでは, デバイス, ミドルノードを Publisher として扱い, インフラサーバを Subscriber とする. ゲートウェイクラウドに Publish/Subscribe ブローカを配置することにより, インフラサーバとデバイスノードの間を非同期かつ独立となるように仲介する. そのため, デバイスに対してインフラサーバから直接通信が行われることはない.

6 プロトタイプの実装

6.1 実行シナリオ

プロトタイプの実行シナリオは Raspberry Pi に接続した温度センサの計測データをゲートウェイクラウドに送信し, 計測データベースへの登録とインフラサーバでの受信を確認する. Raspberry Pi からの送信周期は処理能力の限界値である 6 ミリ秒ごととする. また, インフラサーバから REST の要求/応答を用いて, 計測データベースから任意のデータを受信可能であることを確認する.

6.2 実装環境

プロトタイプのゲートウェイクラウド実装環境を表 1, デバイスの実装環境を表 2 に示す.

表 1 ゲートウェイクラウドの実装環境

コンポーネント	仕様
OS	Ubuntu 14.04 LTS
CPU	Intel® Core™2 Duo CPU E7300 @ 2.66Ghz×2
メモリ	2GB
Publish/Subscribe サーバ	Mosquitto 1.4.3[3]
Web サーバ	Apache 2.4.7
データベース	MongoDB 3.0.7[2]

表 2 デバイスの実装環境

コンポーネント	仕様
マシン	Raspberry Pi B+
温度センサ	MCP9700 -E/TO
コンバータ	MCP3008-8Channel 10Bit

6.3 プロトタイプの構成

プロトタイプの構成を図 13 に示す.

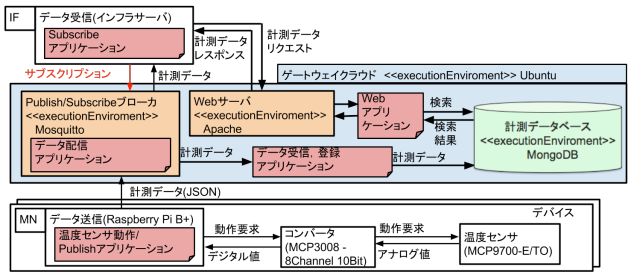

図 13 プロトタイプの構成

7 プロトタイプの適用

Raspberry Pi で計測される温度センサのデータを図 14, インフラサーバにおける受信確認を図 15, 計測データの MongoDB における登録状況を図 16 に示す.

```
2016-01-06 13:59:03.041
temp :    23.55
2016-01-06 13:59:03.047
temp :    23.87
2016-01-06 13:59:03.053
temp :    25.81
2016-01-06 13:59:03.058
temp :    23.87
```

図 14 Raspberry Pi おける計測データ

```
tempture/room1 {"date": "2016-01-06 13:59:03.041", "D_id": "D01", "id": "D01_121", "temp"
: 23.55}
tempture/room2 {"date": "2016-01-06 13:59:03.044", "D_id": "D02", "id": "D02_19", "temp":
 30.0}
tempture/room1 {"date": "2016-01-06 13:59:03.047", "D_id": "D01", "id": "D01_122", "temp"
: 23.87}
tempture/room2 {"date": "2016-01-06 13:59:03.052", "D_id": "D02", "id": "D02_20", "temp":
 29.68}
```

図 15 インフラサーバにおける受信状況

```
> use admin
switched to db admin
> db.temp_test.find()
        中略
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc02"), "temp_id" : "D01_121", "topic" : "temptur
e/room1", "date" : "2016-01-06 13:59:03.041", "tempture" : 23.55, "Device_id" : "D01" }
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc03"), "temp_id" : "D02_19", "topic" : "tempture
/room2", "date" : "2016-01-06 13:59:03.044", "tempture" : 30, "Device_id" : "D02" }
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc04"), "temp_id" : "D01_122", "topic" : "temptur
e/room1", "date" : "2016-01-06 13:59:03.047", "tempture" : 23.87, "Device_id" : "D01" }
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc05"), "temp_id" : "D02_20", "topic" : "tempture
```

図 16 MongoDB 登録状況(一部抜粋)

図 16 に示すように配信と同時にデータベースへの保存も完了している. 次に, REST を用いた要求/応答型のデータ収集例を図 17 に示す. GET メソッドを用いることで, 計測データベースに保存してあるデータから, 必要なデータをリソースとして取得できる.

図 17 REST を用いた計測データ受信例

図 17 に示すようにインフラサーバからデータベースを検索し, 条件に当てはまっているデータがリソースとして一覧表示されていることが確認できた.

8 アーキテクチャの評価と考察

(1) 評価

本アーキテクチャの評価として以下の 2 点を挙げる.

1) Publish/Subscribe ブローカによるスケーラビリティの評価

Publish/Subscribe ブローカを用いることでデバイスからのイベントを待ち合わせなく, 発生順にインフラサーバで受信可能になった.

2) インフラサーバのデータの完全性の評価

Publish/Subscribe ブローカと, 要求/応答型の REST のサーバをゲートウェイクラウドで実装することで, インフラサーバはデバイスデータのリアルタイムな収集と, データベース検索によるタイミングによらない収集が可能になる. その結果, 計測データの収集の自由度が高まり, インフラサーバの計測データ収集の完全性を保証する.

(2) 考察

本稿では oneM2M の機能アーキテクチャの構成に基づきスケールアウト可能なゲートウェイクラウドを設置したアーキテクチャを提案し, IoT アーキテクチャにおいて考慮されていないスケーラビリティの問題点を解決した.

9 今後の課題

(1) 共通データモデルの作成

デバイスごとのデータモデルを共通データモデルに変換しデータベースに保存するためのデータモデルの作成.

(2) コンテンツベースの実装

プロトタイプにおいてコンテンツベースでの実装を行うことで, より詳細な条件を指定した Publish/Subscribe ブローカの実装.

(3) 膨大なデバイスでの実装

プロトタイプにおいて, 膨大な数のデバイスを接続した場合のスケールアウト可能なクラウドにおける実装.

10 まとめ

本稿では, ゲートウェイクラウドと Publish/Subscribe ブローカを用いた IoT アーキテクチャを提案した. デバイスの計測データを目的に合わせて収集するために REST を用いた要求/応答と Publish/Subscribe ブローカ を併用する方法を用いた. プロトタイプでは温度センサを用いて, デバイスの数が動的に変化可能であることを示した. また, インフラサーバで目的に合わせたタイミングによらないデータの収集が可能であることを示した. その結果, 本アーキテクチャの動作を保証した.

11 参考文献

[1] F. Bonomi, et al., Fog Computing and Its Role in the Internet of Things, Proc. of MCC '12. pp. 13-16.

[2] MongoDB, https://www.mongodb.org/.

[3] Mosquitto, http://mosquitto.org/.

[4] 日経 BP 出版局(編), クラウド大全 第 2 版, 日経 BP 社, 2010.

[5] oneM2M, Functional Architecture, Jan. 2015.

[6] S. Tarkoma, Publish/Subscribe Systems: Design and Principles, Wiley, 2012.

[7] 富田二三彦, 山崎徳和, MCPC, M2M/IoT 委員会, M2M/IoT 教科書, インプレス, 2015.