一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CS2020-67,IE2020-26(2020-11)

効率的なIoTデータ共有およびIoTサービス配備のためのIoTメタ情報管理に関する検討

高橋　圭介†　　中里　秀則††　　金井　謙治††

† 早稲田大学基幹理工学部　〒169–8555 東京都新宿区大久保 3–4–1
† 早稲田大学理工学術院総合研究所　〒169–8555 東京都新宿区大久保 3–4–1
E-mail: †saru56suke.nb1@akane.waseda.jp, ††nakazato@waseda.jp, †††k.kanai@aoni.waseda.jp

あらまし　本稿では、物理 IoT デバイスや IoT デバイス仮想化ソフトウェア (ThingVisor) が持つメタ情報の管理サービスである Catalog Server およびその各種プロトコルを提案する。本 Catalog Server を介することで、仮想 IoT システムへの ThingVisor の配備や生成される Virtual Things(vThings) の接続性までを自動的に管理することが可能となる。この機能を実現するために、1) 物理 IoT デバイスのメタ情報を Catalog Server へ登録するプロトコル、2)ThingVisor 作成者が作成した ThingVisor のメタ情報を Catalog Server に登録するプロトコル、3)ThingVisor の自動配備および vThings 取得に関するプロトコル、これら 3 つのプロトコルについて検討し、特に 3) における ThingVisor 自動配備から vThings 取得までにかかる遅延時間を初期評価した。

キーワード　IoT プラットフォーム、IoT システム仮想化、プロトコル

Proposal of IoT meta data management protocols for efficient IoT data sharing and IoT service deployment

KEISUKE　TAKAHASHI†, HIDENORI NAKAZATO††, and KENJI KANAI††

† School of Fundamental Science and Engineering, Waseda University　3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
† Waseda Research Institute for Science and Engineering　3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
E-mail: †saru56suke.nb1@akane.waseda.jp, ††nakazato@waseda.jp, †††k.kanai@aoni.waseda.jp

Abstract　This paper proposes Catalog Server and its various protocols for efficient IoT meta data management. The Catalog Server has capabilities of collecting and storing of physical IoT devices and IoT device virtualization software (ThingVisor). Through this Catalog Server, it is possible to automatically manage the deployment of ThingVisor to virtual IoT system (VirIoT) and solve connectivity of generated Virtual Things (vThings). To realize the functions, We desing the three following protocols: 1) protocol for registering meta data of physical IoT devices in Catalog Server, 2) protocol for registering meta data of ThingVisor created by ThingVisor creator in Catalog Server, and 3) automatic deployment of ThingVisor into VirIoT. We examine the protocols related to vThings acquisition and these three protocols, and in particular, we initially evaluated the delay for automatic deployment to vThings acquisition in 3).

Key words　IoT Platform, IoT Virtualization, Protocol

1. 背　　景

近年, Internet of Things (IoT) の普及に伴い, インターネットに接続される IoT デバイスが急増している。IoT サービスにおいても、交通、防犯、行政など多岐に渡っており、これらは独立した分野の課題解決に運用・適用されている。現状、これら IoT サービスの多くは IoT デバイスからアプリケーションまで抱き込む垂直型のサイロとして展開されている。これに対して、都市 OS [1] の参照アーキテクチャで議論されているように、スマートシティにおける IoT サービスの展開は、IoT サービス間のデータ連携、スマートシティをまたいだスマートシティ間の連携のように、水平方向への繋がりが期待されている。

このような背景を踏まえ、異なる分野の IoT サービスを水平方向に連携するためには、IoT デバイスが生成する生のデータだけでなく、各サービス固有のデータ処理後に生成される「知見」も各サービス間で共有することができるようなデータ共有基盤が必要になる。現在、上記で示したデータ共有基盤は、スマートシティ分野にて、FIWARE [2]，oneM2M [3] のような IoT プラットフォームとして様々なベンダーや標準化団体をはじめ、研究開発、実証試験が進められている。しかしながら、これら IoT プラットフォームでは独自のデータフォーマットが定義されており、IoT プラットフォームをまたいでデータ共有することまでは想定されていない。

これに対し、我々は、スマートシティにアプリケーションに拡張性と相互運用性をもたらす仮想 IoT・クラウド連携基盤の研究開発 (Fed4IoT) [4] を進めている。Fed4IoT プロジェクトでは、異なるスマートシティ間、異なる IoT プラットフォーム間における IoT デバイスや生成されるデータを共有するための基盤を提供し、各 IoT プラットフォームに対して拡張性と相互運用性を実現するとともに、スマートシティにおけるアプリケーションの開発及び展開の簡便化の実現も目的としている。これに向けて、筆者らはこれまで IoT デバイスが生成するデータを効率的に共有できる仮想 IoT システム（VirIoT）[5] を提案してきた。この仮想 IoT システムはスマートシティに配置されている物理 IoT デバイスを仮想化させる IoT デバイス仮想化ソフトウェア（ThingVisor)、仮想 IoT デバイスが生成する Virtual Things（vThings）をテナント（ユーザ）へ仲介する仮想サイロ（Virtual Silo（vSilo)）によって実現されている。この ThingVisor および vSilo を利用することで、テナントは IoT デバイスを容易に共有することや地理的に様々な場所への IoT アプリケーションを容易に展開することが可能となる。

提案している仮想 IoT システムそのものは、ThingVisor および vSilo のネットワーク内への効率的な展開や MQTT や HTTP といったプロトコルで相互接続する機能に主眼を置いており、動的な ThingVisor の開発や ThingVisor や vThings の効率的な管理までは検討していない。動的な ThingVisor の開発については、すでに筆者らは [6] にて ThingVisor Factory として提案している。この ThingVisor Factory では、ThingVisor の設計においてネットワーク仮想化技術の一つであるサービスファンクションチェイニング技術を適用している。サービスファンクションチェイニングでは、IoT データの処理機能をサービスファンクションと定義し、ネットワーク内でサービスファンクションを結合する技術を意味する。ThingVisor Factory は、テナントや ThingVisor 開発者が独自の ThingVisor を設計しネットワーク内に自動配備する機能を提供する。しかしながら、ThingVisor Factory では、スマートシティが有する利用可能な物理 IoT デバイスや利用可能なサービスファンクション（あるいは ThingVisor）は予め登録されている（既知）という前提があり、物理 IoT デバイスをはじめ、動的に構築される ThingVisor や生成される vThings を、どのように効率的に収集・管理するかは課題のままである。

そこで本稿では、上記の課題を解決するために、物理 IoT デバイスや ThingVisor が持つメタ情報の管理サービスである Catalog Server およびその各種プロトコルを提案する。Catalog Server は、ThingVisor Factory の構成要素の一つに位置付けられ、テナントや ThingVisor 開発者に対する仮想 IoT システムへのインターフェースの側面も持つ。つまり、本 Catalog Server を介することで、仮想 IoT システムへの ThingVisor の配備や vThings の接続性までを自動的に解決することが可能となり、テナントは、Catalog Server から提示される ThingVisor や vThing を指定するだけで要求する IoT データを利用することが可能となる。この機能を実現するために、1) 物理 IoT デバイスのメタ情報を Catalog Server へ登録するプロトコル、2)ThingVisor 作成者が作成した ThingVisor のメタ情報を Catalog Server に登録するプロトコル、3)ThingVisor の自動配備および vThings 取得に関するプロトコル、これら 3 つのプロトコルについて検討し、特に 3) における自動配備から vThings 取得までにかかる遅延時間を初期評価したので報告する。

2. 関 連 研 究

2.1 IoT プラットフォーム

Fed4IoT では、新たな IoT プラットフォームを開発して運用するのではなく、FIWARE，oneM2M のような既存の IoT プラットフォームを活用することを想定している。そのため、ここでは、IoT プラットフォームの代表として、FIWARE [2] と oneM2M [3] について紹介する。

FIWARE [2], [7] はヨーロッパの大手企業が多く携わり開発している IoT プラットフォームの一つであり、欧州、日本を含む様々なスマートシティで利用が検討されている。FIWARE は Generic Enabler(GE) というモジュールの仕様を規定しており、GE はコンテキストブローカーと IoT ディスカバリの二つの機能を持つ。コンテキストブローカーと IoT ディスカバリはそれぞれ、IoT データのやり取りを行うインターフェース NGSI-10、データの所在のやり取りを行うインターフェース NGSI-9 を持つ。そのため、コンテキストブローカーが IoT デバイスと IoT データの流通を行う役割を持ち、IoT ディスカバリがデータのやり取りを行うべき IoT デバイスを決定するといった管理の役割を持つと言える。このようなインターフェースにより、異なる分野のサービス間でも FIWARE が持つデータを共有することが可能となる。また、FIWARE で用いるデータは、標準化されたデータモデルで管理されており、このデータモデルと上記で示した NGSI により、高度なデータ検索を行うことができる。FIWARE のオープン実装としては、Orion Context Broker [8] が挙げられる。

oneM2M [3], [9] は複数の標準化団体や組織が参画し、開発された水平統合型のプラットフォームである。oneM2M のアーキテクチャではサービス側のアプリケーションを示す Application Entity(AE)、IoT プラットフォームにおける共通機能群である Common Service Entity(CSE) が規定されている。AE 同士でのデータの通信は認められておらず、AE と CSE 間、CSE 同士での通信といった CSE を中継した通信が義務付けられている。CSE は、ゲートウェイに配備され、データ管理、デバイス

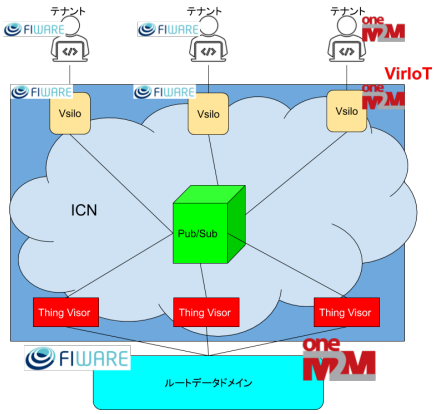

図 1　Fed4IoT システムの全体図

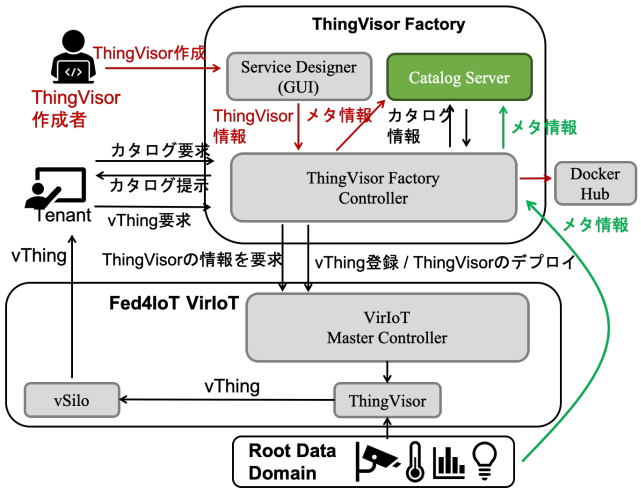

図 2　提案する Catalog Server の位置づけ

管理、セキュリティなど 12 種類の共通機能を持つ。これらの機能は外部から操作できるようリソースとして定義されており、生成、取得、更新、削除、通知の 5 つの操作のみによって構成されるプロトコル設計になっている。さらに、oneM2M は通信方式を標準化しており、通信方式を複数プロトコル、複数のシリアライゼーションから選択できる仕様になっている。これにより、システム開発時に柔軟に通信方式を選択できる。oneM2M のオープン実装として、OpenMTC [10] や OM2M [11] など多く挙げられている。

3. Fed4IoT が提案する仮想 IoT システム VirIoT の構成

本章では Fed4IoT が提案する仮想 IoT システムの構成 [4], [12] について概説する。図 1 に Fed4IoT が提案する仮想 IoT システム「VirIoT」の全体図を示す。図のルートデータドメインは、VirIoT の外に位置しており、FIWARE や oneM2M といった IoT プラットフォームが管理している外部アクセス可能なデータブローカーや物理 IoT デバイスそのもので構成されており、VirIoT から見たときの「物理」IoT データの供給源を指す。本来、物理 IoT データへのアクセスについてはプロトコルや API の差異はあるが、ここでは抽象度を上げルートデータドメインとして総称する。VirIoT の構成要素として、ThingVisor, vSilo が挙げられる。ThingVisor は、ルートデータドメインからのデータを取得し、必要なデータ処理を施した後、中立データフォーマット（例えば NGSI-LD）へ変換し、vThings としてデータを発行する機能を持つ。この vThings は物理 IoT デバイスと対比し仮想 IoT デバイスとして扱われ、この仮想 IoT デバイスを生成するソフトウェアを ThingVisor として呼称する。vSilo は、テナントから見たときに、仮想 IoT デバイスがテナント間で独立して扱えるようにする（サイロ化する）役目を持ち、各 vSilo はテナント固有の IoT プラットフォームのデータブローカー機能を持つ。具体的な機能として、vSilo は ThingVisor から発行される vThings の取得、中立データフォーマットから各 IoT プラットフォーム固有のデータフォーマットへの変換およびテナントへのデータ提供機能で構成される。これらの機能の制御は、vSilo 内に vSilo と呼ばれる vSilo controller が司る。この ThingVisor と vSilo の間を結ぶため、VirIoT では、MQTT に代表される Pub/Sub モデル型の通信プロトコルを採用している。

4. 提案する IoT メタ情報管理プロトコル

本章では、物理 IoT デバイスや ThingVisor が持つメタ情報の管理サービスである Catalog Server およびその各種プロトコルを提案する。

1 章で説明したように、現状の VirIoT は、ThingVisor と vSilo をネットワーク内に配備し接続する機能を持つのみで、ルートデータドメインが持つ IoT デバイスのメタ情報や ThingVisor Factory を通して作成された ThingVisor のメタ情報の管理機能は持たない。そのため、VirIoT の利便性をより高めるために、物理 IoT デバイスや ThingVisor のメタ情報管理やテナントと VirIoT のインターフェース機能を Catalog Server により実現する。以降にて、上記のそれぞれの機能を実現するためのプロトコルについて個別に説明する。なお、提案する Catalog Server は図 2 に示すように、ThingVisor Factory の構成要素の一つに位置することとする。

- 物理 IoT デバイスのメタ情報を Catalog Server へ登録するプロトコル
- ThingVisor 作成者が作成した ThingVisor のメタ情報を Catalog Server に登録するプロトコル
- ThingVisor の自動配備および vThings 取得に関するプロトコル

各プロトコルの簡単なシーケンスについては図 2 にて色分けして図示している。緑は物理 IoT デバイスの情報を Catalog Server へ登録するプロトコルを、赤は ThingVisor 作成者が作成した ThingVisor のメタ情報を Catalog Server に登録するプロトコルを、黒はテナントが Catalog Server を介した ThingVisor の自動配備および vThings の取得に関するプロトコルをそれぞれ示している。

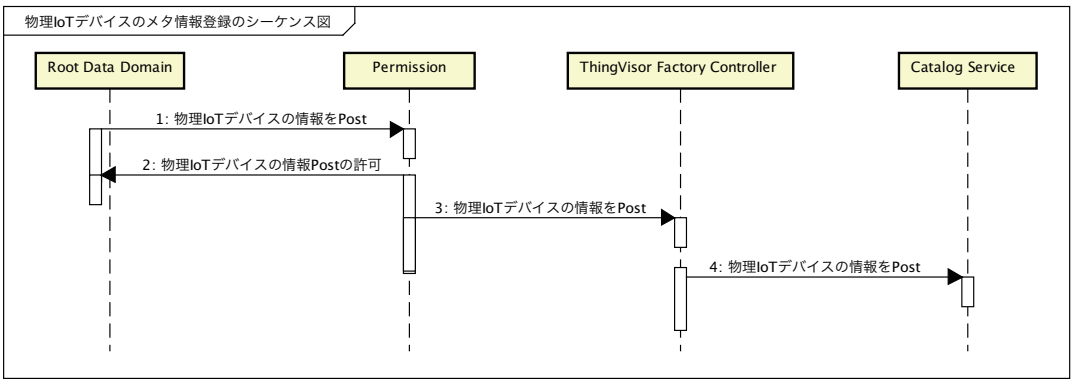

図 3　物理 IoT デバイスのメタ情報登録のシーケンス図

4.1　物理 IoT デバイスのメタ情報を Catalog Server へ登録するプロトコル

ThingVisor Factory にて作成者やテナントの要求に応えるような多様な ThingVisor を作成するためには、ルートデータドメインが持つ物理 IoT デバイスのメタ情報を共有する必要がある。そのため、Catalog Server では、これらメタ情報を保存、検索といった機能を持つ。

そのため、本稿では、その初期検討として、図 3 に示すような物理 IoT デバイスのメタ情報を登録するためのプロトコルを検討する。図に示すように、ルートデータドメインから物理 IoT デバイスの情報を Thing Visor Factory 経由で Catalog Server に登録する。この時、ルートデータドメインと VirIoT は独立して管理されているため、物理 IoT デバイスの情報を登録するにはルートデータドメインは VirIoT 側からの許可が必要となる。そのため、ThingVisor Factory は VirIoT 側で許可した物理 IoT デバイスのメタ情報を受け取り、その情報を Catalog Server に登録することができる.

4.2　ThingVisor 作成者が作成した ThingVisor のメタ情報を Catalog Server に登録するプロトコル

次に、ThingVisor 作成者が ThingVisor を作成し、そのメタ情報を登録するプロトコルについて説明する。図 4 に示すように、はじめに ThingVisor 作成者は、Catalog Server から ThingVisor Factory Controller 経由で利用可能な IoT デバイスや ThingVisor の情報を取得する。この情報をデータフロープログラミングにおけるノード（ブロック）として表示することで、ThingVisor 作成者は、GUI プログラミングにより直観的に ThingVisor を作成する。この時、作成した ThingVisor を ThingVisor Factory Controller に送信することで、ThingVisor Factory 側で ThingVisor の Docker イメージを生成し Docker Hub に登録する。Docker イメージの登録が完了した後、作成した ThingVisor のメタ情報を Catalog Server に登録する。

4.3　ThingVisor の自動配備および vThings 取得に関するプロトコル

最後に、テナントが本 Catalog Server を介して vThings を選択し、必要な ThingVisor を VirIoT へ自動配備および vSilo と接続するプロトコルについて紹介する。本プロトコルによって VirIoT への ThingVisor の配備をテナントから見て自動化できることから、VirIoT へのインターフェースという側面も持つ。図 5,6 より、はじめにテナントは Catalog Server に登録されている vThings の一覧を取得する。そこで、テナント固有の vSilo に追加したい vThings を選択し、ダッシュボードから ThingVisor Factory Controller へと vThings を要求する。ThingVisor Factory Controller は vThings を要求されると、要求された vThings を出力する ThingVisor が起動しているかどうか VirIoT の Master Controller に問い合わせる。もし起動しているならば、図 5 より、ThingVisor Factory Controller は vSilo に vThings を追加するよう Master Controller に命令する。起動していない場合は、図 6 より、ThingVisor Factory Controller が Master Controller に ThingVisor を起動させる命令を送り、起動させる。その後上記のように vSilo に vThings を追加するよう Master Controller に命令する。最後に、Master Controller が ThingVisor Factory Controller へ vThing 追加完了の旨を送信する。

5.　VirIoT 上における vThing 取得にかかる遅延時間の評価

5.1　実験環境及び実験シナリオ

本稿で提案しているプロトコルの性能評価として、特にクリティカルになると予想される 3 つ目のプロトコルについて実際に VirIoT 上に ThingVisor が自動配備され vThing の取得までにかかる遅延時間について評価を行う。その際、図 5 と図 6 に示されているように、ThingVisor が配備済みかどうかによって大きく遅延時間が変わると予想されることから、これらのシーケンスに注目して遅延評価を行う。

まず実験環境について説明する。本評価実験では、仮想マシン VitualBox 上で、ゲスト OS は Ubuntu 18.04.5 で行なった。また仮想マシンは一台で行っており、VirIoT、Catalog Server、ThingVisor Factory Controller は全て同一のマシンで起動することとする。

ThingVisor は WeatherAPI から気象情報を取得する ThingVisor（weather-thingvisor）を用いており、世界の都市の気象情報を vThings として発行する。vSilo については、IoT プラットフォームである oneM2M のオープンソース実装の一つである Mobius ブローカーを利用しているものを採用する。

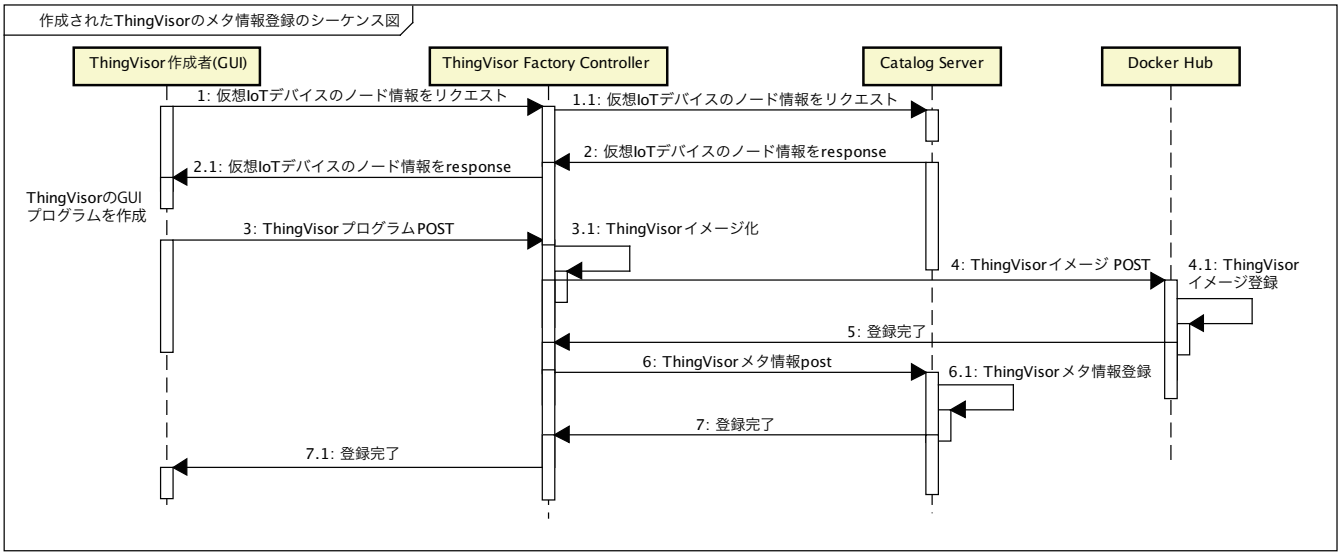

図 4　作成された ThingVisor のメタ情報登録のシーケンス図

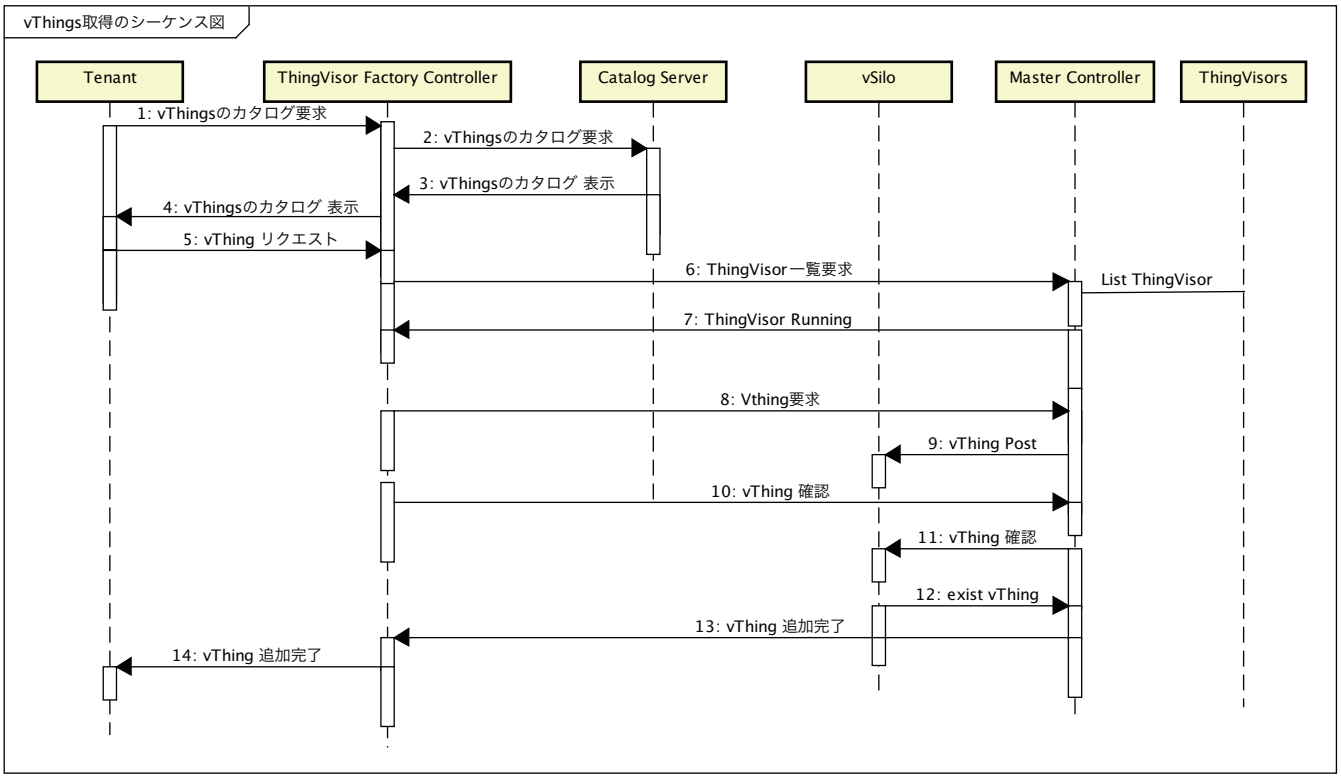

図 5　ThingVisor 起動時の vThings 取得のシーケンス図

ThingVisor と vSilo 間のデータ接続は MQTT ブローカーを利用する。

また、図 6 のシーケンス番号 1 から 20 までの処理を行えるよう ThingVisor Factory Controller と Catalog Server を含めプロトコル実装し、シーケンス番号 5 から 20 までの vThing 取得にかかる遅延時間を二つの条件下で測定し、遅延評価を行なう。なお、本実験の試行回数は以下の条件それぞれに対して 10 回行い、平均値をとることとする。

二つの条件を以下に示す。

- ThingVisor が起動しているときの vThing 取得にかかる遅延時間
- ThingVisor が起動していないときの vThing 取得にかかる遅延時間

5.2　評価結果

表 1 より、ThingVisor が起動している時よりも ThingVisor が起動していないときの方が IoT サービス配備遅延時間が大きくなっており、ThingVisor を起動させるのに大きな遅延が生じることがわかる。

表 1　vThing 取得における遅延時間

TV の条件	TV の作成時間 [S]	IoT サービス配備の遅延時間 [S]
TV が起動している	ー	1.619681621
TV が起動していない	8.347873259	9.589337516

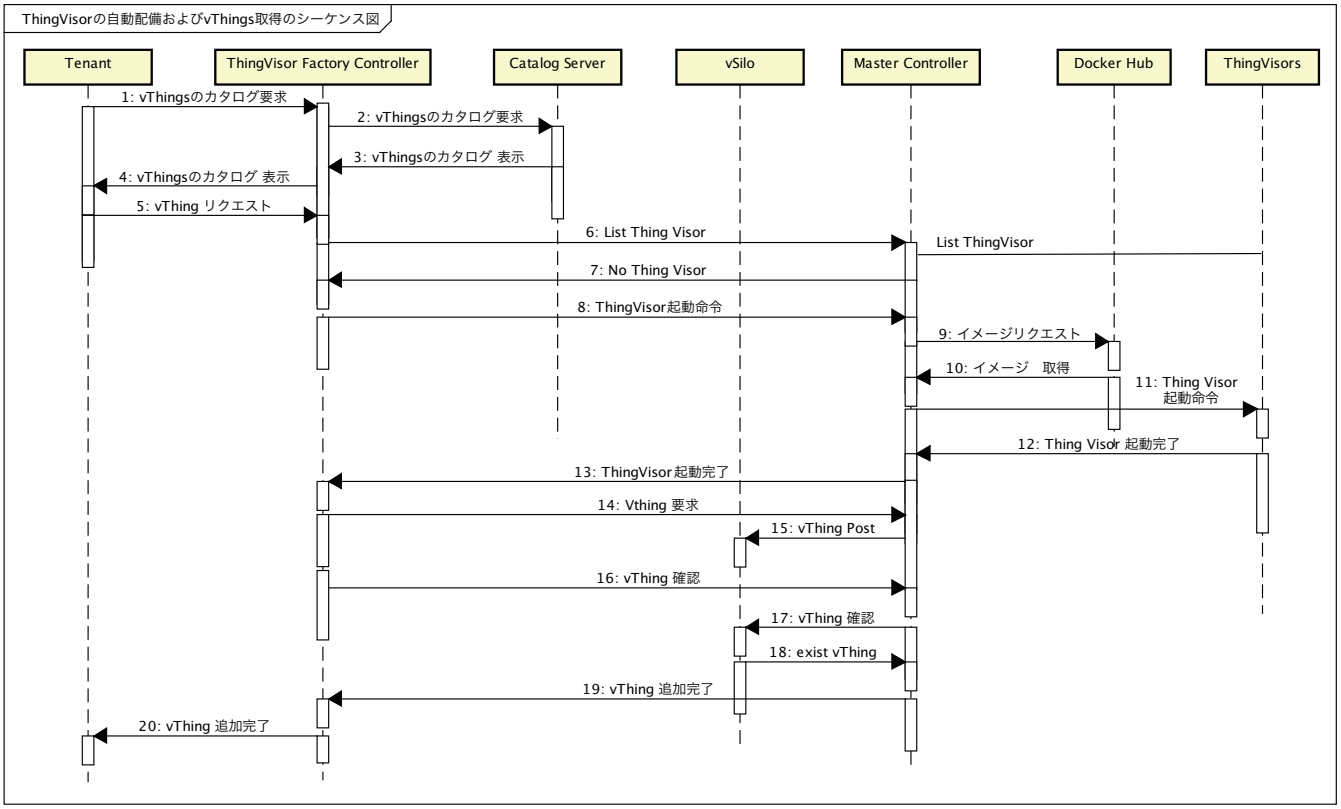

図 6　ThingVisor の自動配備および vThings 取得のシーケンス図

6. まとめと今後の展望

本稿では、Fed4IoT が提案している仮想 IoT システム（VirIoT）の利便性を高めるため、物理 IoT デバイスや ThingVisor が持つメタ情報の管理サービスである Catalog Server およびその各種プロトコルを提案した。具体的には、1) 物理 IoT デバイスのメタ情報を Catalog Server へ登録するプロトコル、2)ThingVisor 作成者が作成した ThingVisor のメタ情報を Catalog Server に登録するプロトコル、3)ThingVisor の自動配備および vThings 取得に関するプロトコル、これら 3 つのプロトコルについて検討し VirIoT への ThingVisor の配備や vThings の接続性までを自動的に解決することを確認とした。また、3) における ThingVisor の自動配備から vThings 取得までにかかる遅延時間の評価を行なった。

現状、単一のテナントである想定であるが、今後は、テナントを複数人想定し、複数のテナントが同じ ThingVisor を要求する場合において、ThingVisor が出力する vThing を共有し vSilo で分離（Isolation）できるかどうかを検証する予定である。さらに、複数テナントがそれぞれ地理的に異なる場所から要求する場合や ThingVisor の配置ノード先によって vThing の取得時間の変化を評価する予定である。

謝辞　本研究の成果は，総務省の（平成 30 年度）戦略的情報通信研究開発推進事業（国際標準獲得型）【JPJ000595】「スマートシティアプリケーションに拡張性と相互運用性をもたらす仮想 IoT-クラウド連携基盤の研究開発 (Fed4IoT)」によるものである。

文　献

[1] 戦略的イノベーション創造プログラム，“都市 OS,” スマートシティリファレンスアーキテクチャ，第 7 章，内閣府，3 月 2020.
[2] FIWARE, “FIWARE web page,” [Online]. Available: https://www.fiware.org.
[3] oneM2M, “oneM2M web page,” [Online]. Available: https://onem2m.org.
[4] 中里秀則, “スマートシティのインターオペラビリティ　—日欧共同研究 Fed4IoT—,” ITU ジャーナル，vol.49，no.9，pp.9–12，9 月 2019.
[5] A. Detti, G. Tropea, G. Rossi, J.A. Martinez, A.F. Skarmeta, and H. Nakazato, “Virtual IoT systems: Boosting IoT innovation by decoupling things providers and applications developers,” 2019 Global IoT Summit (GIoTS), pp.1–6, June 2019.
[6] 金井謙治，中里秀則，金光永煥，“Things as a Service を実現する ThingVisor Factory と IP/ICN 間の Things 共有アーキテクチャの提案,” 信学技報 vol. 119, no. 424, CS2019-101, pp. 19-24，電子情報通信学会，2 月 2020.
[7] 竹内　崇，寺澤和幸，“データ利活用型都市経営を実現する情報プラットフォーム：FIWARE,” NEC 技報，vol.71，no.1，pp.29–32，9 月 2018.
[8] FIWARE, “Orion conext broker web page,” [Online]. Available: `https://github.com/telefonicaid/fiware-orion`.
[9] 原田　恵，前大道浩之，山崎育生，“水平統合型 IoT プラットフォーム標準規格 oneM2M の最新動向,” NTT 技術ジャーナル，vol.130，no.2，pp.69–72，2 月 2018.
[10] open mtc, “OpenMTC web page,” [Online]. Available: `https://www.openmtc.org/`.
[11] The Eclipse Foundation, “Eclipse OM2M,” [Online]. Available: `https://www.eclipse.org/om2m`.
[12] 金井謙治，吉田英聖，金光永煥，中里秀則，横谷哲也，向井宏明，中村健一，上杉　充, “Things as a Service を実現する Fed4IoT プラットフォームの研究開発,” 信学技報 vol. 119, no. 256, CS2019-69, pp. 39-44，電子情報通信学会，10 月 2019.