解説

MARS (Multi-Agent-based Resource Sharing)

プラットフォームについて†

顧 優輝*1・菅原 研次*2

1. はじめに

近年、IoT 技術の世界的な発展と普及により、多種多様なモノがインターネット上の情報資源や情報処理基盤に対して接続することが可能となった。この接続可能となったモノのことを IoT デバイスと呼ぶ。この IoT デバイスは、現実空間の様々なデータがクラウドに蓄積することを可能とし、そのデータを処理し、自律的で適応的なサービスを提供する。例えば IoT の実現の一例として、駅にある自動販売機はインターネットと連携することでその場所や人に合わせた飲み物を提供している。

総務省の白書によれば、2022 年には世界の IoT デバイスの数が 348 億台を突破するであろうと予測されている¹。このことから、今後も日常生活で利用する様々なモノがインターネットに接続されることで、より多くのデータが収集されることとなり、ユーザの環境に応じたより細やかなサービスが提供される。しかしながら、それらの IoT デバイスを用いてユーザを支援するためのサービスを構築するためには、環境に対応したサービスを行うための IoT アプリケーションをそれぞれの環境に適応させて開発する必要がある。そのため、IoT デバイスをエージェントプログラムでラッピングし、エージェントの協調機能を用いて、IoT デバイスを適応的に組織化するエージェントアプローチが提案されてきた [1]。

しかしながら、既存のマルチエージェントプラットフォームでは、IoT アプリケーションを開発する際に IoT デバイスの制御や操作、そして IoT デバイスのネットワークの再利用のためのエージェントの知識記述に関して必要十分な機能を有していないため、我々は IoT に適応した新たなマルチエージェントプラットフォームである MARS (Multi-Agent-based Resource Sharing) プラットフォームの開発を行っている [2,3]。本稿では、IoT アプリケーション開発のためのエージェント技術の動向と、我々のグループで開発を進めている MARS プラットフォームについて述べる。

2. IoT とエージェント技術の動向

本章では、主な IoT のプラットフォームとエージェントの動向について述べる。

2.1 IoT に関するプラットフォームの動向

Internet of Things (IoT) という用語は、1999 年に Kevin Ashton が初めて用いたとされる [4]。現在までに IoT の概念を実現するための様々な取り組みが行われている。EU のプロジェクトである、Internet of Things Architecture (IoT-A)²では、IoT のアーキテクチャにおける参照モデルの作成と主な構成要素の定義の提案がなされた。また、ドイツの Industry 4.0 [5] や IoT-A のプロジェクト完了後の EU の Horizon 2020³等の取り組みにおいて IoT に関する研究開発の促進がなされた。さらに、IEEE においても IEEE P2413 というプロジェクトが発足し、標準化に向けての取り組みが行われている⁴。

そのような流れの中で、様々な IoT プラットフォームが開発・提供されている。オープンソースソフトウェア (OSS) による IoT プラットフォームとしては、Mainflux⁵、DeviceHive⁶ 等がある。また、商用の IoT プラットフォームの例として、Google Cloud IoT Core⁷や AWS IoT⁸、SORACOM⁹が挙げられる。Google Cloud IoT Core や AWS IoT では、エッジとクラウド側のどちらの環境もサポートしている。また、SORACOM は、IoT デバイスとクラウド処理基盤の間のネットワークの通信部分や IoT デバイスから得られたデータの前処理を行う機能等をサポートしている。

このような企業や研究機関で開発されている多種多様な IoT プラットフォーム間では、独自のプロトコルやデータのフォーマットにより通信が行われている。そのため、プラットフォーム間での相互接続ができない状況となっている。この問題はサ

† Multi-Agent-Based Resource Sharing (MARS) Platform
Yuki KAERI and Kenji SUGAWARA

*1 目白大学メディア学部
Faculty of Media Studies, Meiji University

*2 千葉工業大学情報科学部
Faculty of Computer and Network Science, Chiba Institute of Technology

1 <https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r02/html/nd114120.html>

2 IoT-A (Internet of Things Architecture), EU Integrated Project,
<http://www.meet-iot.eu/iot-a-paper.html>

3 <https://ec.europa.eu/digital-single-market/en/research-innovation-iot>

4 <https://standards.ieee.org/standard/2413-2019.html>

5 <https://www.mainflux.com/>

6 <https://devicehive.com/>

7 <https://cloud.google.com/iot-core>

8 <https://aws.amazon.com/iot/>

9 <https://soracom.jp/>

イロ化 (silo-ed) と呼ばれている。Web の技術を用いてこれらの IoT の問題点を解決することを目指して、Web of Things というソリューションの標準が検討され、2020 年には勧告がなされている¹⁰。

2.2 マルチエージェントシステムを開発するためのマルチエージェントフレームワーク及びプラットフォーム

マルチエージェントシステムの開発を支援するために、様々なマルチエージェントフレームワークやマルチエージェントプラットフォームが存在する。

また、マルチエージェントシステムの開発のための標準化を行う組織として、Foundation for Intelligent Physical Agents (FIPA)¹¹があり、各マルチエージェントプラットフォーム間の相互運用のために FIPA 標準が作成された。エージェントは Agent Communication Language (ACL) と呼ばれる言語及びプロトコルを用いて、エージェント間の通信を行う。ACL には KQML [6] や、前述の FIPA が標準化し、KQML を拡張した FIPA-ACL¹²等がある。

Java Agent DEvelopment Framework (JADE) は、この FIPA 標準に準拠したエージェントフレームワークとして Java で実装され、開発が続いている [7]。また、Open Multi-Agent System (OMAS) は、独自の知識ベース持ち、KQML を拡張した ACL による通信を行うことができるマルチエージェントプラットフォームである [8]。日本国内では、ADIPS フレームワークの概念に基づくフレームワークである DASH や、その開発支援環境である IDEA が提供されている [1]。ADIPS フレームワークとは、クラウド上のサーバプログラム、ネットワークにおける通信のためのミドルウェア、エッジにおけるリソースをエージェント化し、エージェントの持つ協調機能により、それらの要素を連携することで、アプリケーションを動的に構成することが可能である [1,9]。

2.3 IoT アプリケーション開発におけるエージェント

Savaglio らによるエージェント型 IoT に関する研究のサーベイ [10] によれば、エージェント型の開発を行うことで IoT のリソースと通信インタフェースによる技術的な相互運用性が得られ、メッセージ形式 (ACL) が同一なことによるメッセージ構文上の相互運用性が実現できることが指摘されている。また、認知的で自律的な IoT システムを実現可能であることが指摘されている。

エージェント型 IoT における開発が可能なミドルウェアとしては、Smart Object という概念を用いた ACOSO [11] がある。ACOSO では、JADE や Jadex をエージェントのフレームワークとして利用している。

また、別の開発可能なエージェントプラットフォームとして、Radigost [12] がある。Radigost は、JavaScript や HTML5、

WebSocket 等の Web の技術を利用して実装され、Web ブラウザ上動作する。さらに、Siebog [13, 14] というエージェントミドルウェアでは、前述の Radigost をクライアント側で使い、サーバ側で J2EE に移植した Jason インタプリタを備えた XJAF というエージェントミドルウェアを用いている。このプラットフォームでは、クライアントとサーバ間の通信に WebSocket と HTTP が用いられる。

2.4 IoT に対する既存のマルチエージェントプラットフォームの問題点

JADE や OMAS, そして DASH/IDEA 等の既存のマルチエージェントプラットフォームには、以下のような問題があり、エージェントを用いて IoT アプリケーションを構築することが難しい状況にある。

まず 1 つ目の問題点 (P1) は、既存マルチエージェントシステムを構成するエージェント要素が、異なるファイアウォールの内側に存在する場合、ファイアウォールによりブロックされるため、エージェント間のメッセージ通信ができないことである。すなわち、既存のマルチエージェントプラットフォームではエージェント間通信はグローバルな IP アドレスを用いた TCP や UDP のブロードキャストのようなプロトコルを利用していることが、エージェント間通信がブロックされる原因となっている。このため、エージェントアプリケーションごとにファイアウォールを超えるための機能を作成する必要がある。

2 つ目の問題点 (P2) は、マルチエージェントシステムから IoT リソースの集合を制御及び管理する機能がないことである。すなわち、エージェント間メッセージとデバイスの IoT リソースの API のメッセージ形式の間に差があり、その差を埋めるための機能が提供されていない。このため、IoT デバイスを制御し、管理するためのエージェントをそれぞれ作成する必要があるため、エージェント型 IoT アプリケーションの開発者の負担になっている。

MARS プラットフォームは、これらの問題点を同時に解決し、IoT に対応した新たなマルチエージェントプラットフォームの構築することを目的とする。

3. MARS プラットフォームの概要

本章では、我々が開発を行っている MARS プラットフォームのアプローチについて概説する。

3.1 IoT リソースとリソースコネクタ

現実空間に設置可能なデバイス (カメラ、マイク、加速度センサ、ディスプレイ、スピーカ等のセンサやアクチュエータ) のことをユビキタスデバイスと呼ぶ。また、デジタル空間のデータ及びそれら処理・分析するためのプログラムのことをデジタルリソースと呼ぶ。さらに、ユビキタスリソースとデジタルリソースをあわせて情報リソース (以下、リソース) と呼ぶ [15, 16]。本稿では、IoT におけるリソースのことを IoT リソースと呼ぶ。

エージェント間メッセージと IoT リソースの API を繋ぐ機能を持つエージェントプログラムをリソースコネクタと呼ぶ。

10 <https://www.w3.org/TR/wot-architecture/>

11 <http://www.fipa.org/>

12 <http://www.fipa.org/repository/aclspecs.html>

図1 MARS プラットフォームの概要

すなわち、リソースコネクタは、エージェントメッセージを解釈して、IoT リソースの API の制御コマンドに変換し、IoT リソースからのデータ形式をエージェントメッセージに組み込む動作を行う。リソースコネクタの詳細については以降の節にて述べる。

3.2 MARS プラットフォームにおけるリソースの共有

図1にMARSプラットフォームの概要図を示す。MARSプラットフォームは、リソース空間、リソースコネクタ空間、エージェント空間により構成される。図の左部分には、Application レイヤ、Network レイヤ、そして Perception (Device) レイヤによって構成される代表的な IoT の3層モデルを示している。MARS プラットフォームでは、それぞれのレイヤ内のデバイス・プログラム・データを IoT リソースとみなすことで、リソースコネクタを経由してエージェント空間に配置されたエージェントによる制御が可能となる。

MARS プラットフォーム上のエージェントは、他のエージェントと通信する際、ネットワーク接続における複雑な設定等を必要とせず、透過的にエージェントメッセージを用いた通信が可能である。そして、リソースコネクタを経由したエージェントからの指示により、各レイヤ内の IoT リソースを必要に応じて動的に連携させることができる。この連携により、IoT リソースを組み合わせることで、IoT アプリケーションを自律適応的に構成することができる。

また、Resource Connector Repository 機能により、IoT アプリケーションを動的に構成する際に必要な IoT リソースを制御するためのリソースコネクタをスタンバイした状態で保持し、必要に応じて有効/無効化を行うことで IoT リソースごとに対応するリソースコネクタの管理を可能とする。

さらに、Agent Repository 機能は、ADIPS フレームワークのように動作し、IoT アプリケーションを構成するために必要なエージェントの組み合わせを Agent Repository 内に保存する。そして、必要に応じてその組み合わせは再利用され、エージェント空間に展開され、サービスが提供される。

これらの機能を備えた MARS プラットフォームを実現することで、まず、透過的なメッセージ通信の実現により、問題点 P1 の解決を行う。そして、Resource Connector Repository 機能や Agent Repository 機能、そして、リソースコネクタ等の機能により、IoT リソースの管理及び制御を可能とすることで、

図2 リソースコネクタエージェントの構造

問題点 P2 の解決を行う。

4. マルチエージェントプラットフォーム MARS の開発

4.1 リソースコネクタの設計

図2にリソースコネクタエージェントの構造を示す。リソースコネクタエージェントは、対象のリソース及びエージェントの特性を記述したプロファイル (Resource Profile, Agent Profile)、プラグインの機構を持つ入れ替え可能な対象のリソースの制御部 (Resource Plugin)、他のエージェント (Ag) と ACL によってコミュニケーションを行う通信部 (Agent Communicator)、そして、各プロファイルによりリソースコネクタエージェントの動作を制御するコア (Resource Connector Core) により構成される。リソースコネクタエージェントにより、各 IoT リソースをエージェント化することで、他のエージェントからはエージェントメッセージを介して IoT リソースの状態の把握や操作、協調を行うことが可能となる。また、リソースコネクタエージェントは、Resource Plugin の機構により、IoT リソースの制御プログラムをエージェント空間内のエージェントの記述言語を用いて実装する必要がない。そのため、エージェントの開発者以外がプラグイン部分のみ開発することが可能であり、開発者の負担を軽減できる。

さらに、IoT リソース同士の通信の接続開始・終了等の制御についても IoT リソースの API を制御することで実現可能である。後述する MARS プラットフォームの応用例においては、カメラやディスプレイ等の制御以外にもビデオチャットのプログラムの制御 (複数拠点間の接続の開始・終了等) を行っている。

4.2 MARS エージェントの設計

図3にMARSエージェントの構造を示す。MARS エージェントは、他のエージェント (Ag) と ACL によってコミュニケーションを行う通信部 (Agent Communicator)、エージェントの特

図3 MARS エージェントの構造

図4 Coterie の構造

性と状態が記述されたプロフィール (Agent Profile)、エージェントのワーキングメモリ (Working Memory)、エージェントが自身の持つスキルを実行する際のタスクを管理するタスク制御部 (Task Manager)、Agent Profile や Working Memory の状態を参照して各モジュールを制御するコア制御部 (MARS Agent Core)、そして、エージェントが持つスキルの集合 (Skills) から構成される。

4.3 Coterie の設計

図4にCoterieの構造を示す。Coterieは、エージェント空間を実現するために、Publish/Subscribe モデルにおける Broker としてエージェントメッセージをやり取りする Agent Message Broker (AMB)、AMB に接続された複数の MARS エージェントと Resource Connector、エージェントメッセージを記録・再利用するための Agent Log Manager とログの保存先のデータベースである Agent Message DB、エージェントを管理するための Agent Manager、Coterie 内のエージェントが共有メモリとして利用可能な Blackboard を管理する Blackboard Manager、そして、他の Coterie との接続を行う通信エージェントである XA から構成される。

Coterie は、XA によって複数の Coterie 間を接続することが可能であり、スケールアウト可能な構造を持つ。さらに、AMB 及び XA により、MARS エージェント間で ACL によるメッセージをやり取りする際、Coterie に対して接続がなされているすべてのエージェントと透過的にメッセージのやり取りが

可能となる。そのため、エージェントの開発者が、接続対象の PC やサーバの IP アドレス等の物理的な情報を意識せずとも、エージェント同士のメッセージ通信の実装が可能である。この機構により、透過的なエージェントメッセージ通信を実現することで、問題点 P1 の具体的な解決を図る。

4.4 MARS プラットフォームの実装

MARS プラットフォームの主な実装は、Python を用いて行った。また、エージェントがユーザとインタラクションを行うユーザインタフェース (UI) 部分では、リアルタイムなインタラクションを実現するために C#, Unity や .Net Framework 等の開発用フレームワーク、そして、Web ブラウザ経由のインタフェースでのやり取りのために JavaScript を実装に用いている。これらの言語や開発用フレームワークは、多種多様なデバイスやサーバでの実装を行う上での互換性の確保や、各種ディープラーニングライブラリ (TensorFlow, PyTorch 等) を用いたエージェントの拡張を可能とするために選択した。

さらに、Unity 上で動作する MARS エージェントも実装しており、簡単に Unity 上のオブジェクトをエージェント化することが可能である。この機能を用いることで、例えばスマートグラスの UI が非常に簡便に実装可能となった。また、VR 空間に存在する何らかのオブジェクトに対してエージェントがインタラクションすることも可能となった。

具体的な各モジュールの実装では、まず Coterie を実装するために、プロトコルとして多くの IoT プラットフォームで採用されておりメッセージのやり取りに利用されている MQTT プロトコルを用いた。通信には TLS を用いた暗号化を行うことでセキュリティを担保している。MQTT プロトコルを用いる利点としては、Google Cloud IoT、AWS IoT、Azure IoT Hub¹³ 等の様々な IoT 向けクラウドサービスが提供されており、それらを利用することで商用のクラウドサービス基盤においてもスケールアウトが可能である点が挙げられる。エージェントメッセージのやり取りを行う AMB の Broker は、MQTT サーバである VerneMQ¹⁴を用いて実装を行った。また、MQTT のメッセージの永続的な保存のために、MariaDB¹⁵と Redis¹⁶を組み合わせ Agent Log Manager の実装を行った。また、Coterie の各要素を実現するために、Python を用いて実装を行った。

エージェントが動作する全ての場所においてエージェント同士の透過的なメッセージのやり取りを実現するためには、エージェントの通信が接続されたネットワークのファイアウォールを超える必要がある。そのため、Coterie の中心である AMB の Broker サーバへは、インターネットに接続可能な環境においてほぼ開放されている 443 番のポートを用いて接続する形式としている。

エージェントの開発を行うための統合開発環境 (IDE) は Web ブラウザ上で UI が動作するように実装した。エージェン

13 <https://azure.microsoft.com/ja-jp/services/iot-hub/>

14 <https://vernemq.com/>

15 <https://mariadb.com/>

16 <https://redis.io/>

トの処理の実行状況とメッセージのやり取りを確認するための機能を実現するためには、UI がエージェントの状態をリアルタイムに反映する必要がある。そのため、ブラウザ上の UI と Web サーバとの通信には WebSocket を用いた。Web サーバ側を実装するためのライブラリは Flask を用いている。

4.5 MARS プラットフォームにおける IoT リソースのオーケストレーション機能

IoT リソースのオーケストレーションとは、様々なユーザ（IoT サービス提供者、IoT サービス利用者等）の要求に応じて、IoT リソースを自律的に連携させ IoT アプリケーションを構成することを指す。

多くの IoT リソースを利用する環境においては、必要な IoT リソースを必要な IoT アプリケーションに割り当てることが難しいため、MARS プラットフォームでは、エージェントが IoT リソースのオーケストレーションとその割当を自律的に適応的に行う。このオーケストレーションでは、ADIPS フレームワークを参考としたエージェントによる IoT リソースのオーケストレーションを行う。

5. MARS プラットフォームによるマルチエージェントシステムの開発例

5.1 分散型ブレインストーミング会議支援システムの開発実験

我々は MARS プラットフォームを用いて、遠隔地間のブレインストーミング会議の支援を行うためのシステムの開発を行った [2,3]。

遠隔地に分散した拠点間においてブレインストーミング会議を実施するためには、ネットワークを介して同期可能なブレインストーミングのための UI を持つアプリケーションや、ビデオ会議アプリケーション等の複数の IoT アプリケーションを組み合わせる必要がある。また、それらの IoT アプリケーションでは、各拠点のローカル環境に配置された多くの IoT デバイス（カメラ、マイク、ディスプレイ、マルチタッチインタフェース等）や、マルチメディアな情報を通信するための機能、そして、拠点間でやりとりされる会議データの保存と再利用を行うためのクラウド上の機能等の IoT リソースが必要となる。

また、このような会議支援システムにより多種多様なブレインストーミング会議の実施を支援するためには、様々な環境やユーザの要求に従って必要な機能やシステムの構成を動的に変更する必要がある。

そこで我々は、チャンネルという概念を提案し、MARS プラットフォームを用いてシステムを実装することで、会議支援システム上の全ての IoT リソースをエージェントから操作可能とした。具体的には、チャンネルの概念を用いて通信部分を含めた IoT リソースの制御をエージェントから行い、エージェントの指示により IoT アプリケーションの動的構成を行った。図 5 に実装したシステムにおける各チャンネルの例（Video Channel, Note Channel）を示す。

図 5 分散型ブレインストーミング会議支援システム [3]

図 6 ファイアウォールを超えたりリソースの共有

5.2 分散型ブレインストーミング会議支援システムの開発実験の結果

日本の千葉工業大学（CIT）フランスのコンピエーニュ工科大学（UTC）において分散型ブレインストーミングの開発を行い、稼働実験を行った。図 6 に稼働実験の際のシステムの状態を示す。実験の結果、各拠点におけるエージェントは、自拠点及び他拠点のファイアウォールの影響を受けることなく、図 6 に示した通り、各拠点間の複数のチャンネルの接続がなされた。よって 2.4 節で示した問題点 P1 は、MARS プラットフォームにより解決できたと言える。

また、本実験では、各チャンネルを MARS プラットフォームによって実装し、動作させた。その結果、各チャンネルにおける IoT リソースが MARS プラットフォームによって管理されることで、チャンネルの追加と削除をエージェントの制御によって可能となった。そのため、IoT アプリケーション開発者が必要に応じてチャンネルを手動で追加する必要がなくなり、開発者の負担が軽減できた。よって 2.4 節で示した問題点 P2 も解決できたと言える。

5.3 プラットフォームによるエージェントの永続性の確保

エージェントはその特性として、自律性、自発性、反応性、社会性、永続性を持つ。MARS プラットフォームでは、特に既存のマルチエージェントプラットフォームが機能として支援していることが少ない永続性に着目し、プラットフォームにエージェントの永続性を持たせるための機能を備えることとした。そのため、プラットフォームが以下の機能と構造を持つこととした。

- クラウドサーバ上にマルチエージェントプラットフォーム上でやり取りされた全てのエージェントメッセージを永続的に保持可能とする。必要に応じて、記録されたメッセー

ジの削除も可能とするが、削除の操作に関するメッセージは残すこととする。

- エージェントが異常終了したタスクが中断した場合、保持されたエージェントメッセージからエージェントの再立ち上げとタスクの復旧を行う。

5.4 JSON 形式を用いた MARS-ACL

MARS プラットフォームでは、様々なプログラミング言語や IoT システムで取り扱いやすい JavaScript Object Notation (JSON) をベースとした MARS-ACL を開発し、利用している。この MARS-ACL では、FIPA-ACL や KQML に互換性を持つため、他のエージェントプラットフォームとの接続が可能である。

6. おわりに

本稿では、IoT 向けマルチエージェントプラットフォームである MARS プラットフォームについて説明を行った。現在の MARS プラットフォームにおいて、以下の 4 つの課題があり、今後対応していきたいと考えている。

- 複数の IoT アプリケーションから同じ 1 つの IoT リソースに対して制御を行う際、エージェントとリソースコネクタで排他制御を行う必要がある。その際の動作を予め MARS プラットフォーム側でテンプレートとして準備し、エージェント開発者が容易にそれらの事象に対応できるようにしたいと考えている。
- Coterie に保存されたエージェントメッセージは永続的に記録される設定にしておき、データの削除のためのコマンドは存在するものの、削除に関する規定は特段設けていない。個人情報保護等の観点からも、より現在の社会に即した実装を検討していきたいと考えている。
- 複数の IoT アプリケーションが同時に動作している状況で、IoT アプリケーション自体のオーケストレーションを行う場合、複数の Coterie を複合的に扱う必要がある。現在の MARS プラットフォームには、複数の Coterie を動的に組み合わせる際に支援する機能が備わっていないため、検討の上、実装する予定である。
- MARS プラットフォーム上で動作するエージェントすべての行為においてパーミッションのような実行権限設定を導入したいと考えている。既存のマルチエージェントプラットフォームでは、エージェントが特権ユーザとして動作していることが多い。そのため、より安全なシステムを目指すために、多くの機能で権限設定を可能としたいと考えている。

さらに、MARS プラットフォームに関するドキュメントの整理を行い、MARS プラットフォームをオープンソースソフトウェアとして GitHub 等から公開していきたいと考えている。

参 考 文 献

- [1] 原英樹, 藤田茂, 菅原研次, 木下哲男, 白鳥則郎: “ADIPS フレームワークのためのエージェント開発支援環境,” 情報処理学会論文誌, Vol.40, No.11, pp. 4030-4040, 1999.
- [2] Y. Kaeri, C. Moulin, K. Sugawara, and Y. Manabe: “Agent-based system architecture supporting remote collaboration via an internet of multimedia things approach,” *IEEE Access*, Vol.6, pp. 17067-17079, 2018.
- [3] Y. Kaeri, K. Sugawara, C. Moulin, and T. Gidel: “Agent-based management of support systems for distributed brainstorming,” *Advanced Engineering Informatics*, Vol.44, 101050, 2020.
- [4] K. Ashton: “That ‘internet of things’ thing,” *RFID J.*, Vol.22, No.7, pp. 97-114, 2009.
- [5] M. Hermann, T. Pentek, and B. Otto: “Design principles for industrie 4.0 scenarios,” *2016 49th Hawaii Int. Conf. on System Sciences (HICSS)*, pp. 3928-3937, 2016.
- [6] T. Finin, R. Fritzson, D. McKay, and R. McEntire: “KQML as an agent communication language,” *Proc. of the 3rd Int. Conf. on Information and Knowledge Management (CIKM’94)*, Nov. 29-Dec. 2, Gaithersburg, p. 456-463, 1994.
- [7] F. Bellifemine, A. Poggi, and G. Rimassa: “JADE-A FIPA-compliant agent framework,” *Proc. of the 4th Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 99)*, Vol.99, Apr. 19-21, London, p. 33, 1999.
- [8] J. P. A. Barthès: “Omas – a flexible multi-agent environment for CSCWD,” *Future Generation Computer Systems*, Vol.27, No.1, pp. 78-87, 2011.
- [9] 菅原研次, 木下哲男, 顧優輝: “IoT アプリケーションを開発するためのエージェント指向技術,” 知能と情報, Vol.30, No.3, pp. 148-156, 2018.
- [10] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, and G. Fortino: “Agent-based internet of things: State-of-the-art and research challenges,” *Future Generation Computer Systems*, Vol.102, pp. 1038-1053, 2020.
- [11] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou: “Agent-oriented cooperative smart objects: From IoT system design to implementation,” *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, Vol.48, No.11, pp. 1939-1956, 2018.
- [12] D. Mitrović, M. Ivanović, Z. Budimac, and M. Vidaković: “Radigost: Interoperable web-based multi-agent platform,” *J. of Systems and Software*, Vol.90, pp. 167-178, 2014.
- [13] D. Mitrović, M. Ivanović, M. Vidaković, and Z. Budimac: “The siebog multiagent middleware,” *Knowledge-Based Systems*, Vol.103, pp. 56-59, 2016.
- [14] M. Ivanovic, M. Vidaković, Z. Budimac, and D. Mitrović: “A scalable distributed architecture for client and server-side software agents,” *Vietnam J. of Computer Science*, Vol.4, pp. 127-137, 2017.
- [15] 顧優輝, 真部雄介, 菅原研次: “ユビキタスデバイスとデジタルリソースを論理エージェントに接続するためのエージェントプラットフォームの開発,” 情報処理学会論文誌, Vol.56, No.1, pp. 284-294, 2015.
- [16] Y. Kaeri, Y. Manabe, K. Sugawara, and C. Moulin: “An IoT Application Connecting Edge Resources and Cloud Resources using Agents,” *Int. J. of Energy, Information and Communications*, Vol.8, No.1, pp. 1-14, 2017.

(2020 年 10 月 31 日 受付)

[問い合わせ先]

〒161-8539 東京都新宿区中落合 4-31-1

目白大学

顧 優輝

E-mail: yk.kaeri@gmail.com

—— 著 者 紹 介 ——

かえり ゆうき
顧 優輝 [非会員]

2017 年千葉工業大学大学院情報科学研究科博士後期課程単位取得退学。2018 年千葉工業大学・博士（工学）取得。同年、目白大学メディア学部メディア学科助教。現在に至る。マルチエージェントシステムと IoT に関する研究に従事。情報処理学会、IEEE 各会員。

すがわら けんじ
菅原 研次 [非会員]

1977 年東北大学大学院修士課程修了。1980 年千葉工業大学助手。現在、千葉工業大学情報科学部情報ネットワーク学科教授。工学博士。1992 年日本工業教育協会功績賞。1994 年情報処理学会山下記念賞。1996 年情報処理学会論文賞。2000 年電子情報通信学会業績賞。2008 年電子情報通信学会フェロー。