6-3　IoT エッジコンピューティング技術
6-3　IoT Edge Computing Technology

寺西裕一　山中広明　河合栄治
Yuuichi TERANISHI, Hiroaki YAMANAKA, and Eiji KAWAI

将来の IoT（Internet of Things）においては、多数のモノ（センサー、家電、ロボット、クルマ等）が生成する膨大なデータをリアルタイムに収集・分析し、実世界にあるモノを制御する高度な情報サービスの実現が期待される。本稿ではこうした情報サービスの実現に適したアーキテクチャとして注目されるエッジコンピューティング技術や関連技術について概観するとともに、著者らが研究開発を進めている IoT エッジコンピューティング技術について紹介する。

In the future IoT (Internet of Things), advanced ICT services that control physical things according to the analysis results of collected real-time data generated by a large number of things (such as sensors, appliances, robots, vehicles). In this paper, we explore Edge Computing technology and related works that attract attention recently as fundamental technologies that are suitable for such future ICT services. We also introduce our currently ongoing projects on research and development of IoT Edge Computing Technology.

1　まえがき

スマートフォンやコンピュータのような専用端末のみならず、人々の身の回りに存在する「モノ」（センサー、家電、ロボット、クルマ等）がネットワーク接続し、安全で効率化された ICT サービスを街の至るところで提供可能とすることで、高い水準の社会生活を可能とするいわゆる IoT（Internet of Things）の実現に対する期待は大きい。

IoT においては、一般に、「モノ」をネットワーク接続し、データの収集や分析等をクラウドと連携しながらサービスを実現する形態がとられる。センサーネットワークとクラウドをつなぐ IoT を実現する上では、広域に配備されたモノが生成するデータをクラウドに送信するための「トラヒックの課題」、モノとクラウドとの間の距離が長いことに起因する「応答遅延の課題」があげられる。これらの課題を解決可能とするアーキテクチャとして、端末上、あるいは、端末に物理的に近いネットワーク装置（エッジ）等に、計算処理を実行可能な計算リソースを設け、クラウド上の処理の一部をそれらで実行する「階層化クラウドアーキテクチャ」が提唱され、様々な要素技術の研究開発が進められている。いわゆる『エッジコンピューティング（またはフォグコンピューティング）』と呼ばれるコンピューティング形態である。

本稿では、近年注目を集める IoT を活用したアプリケーション（以下、IoT アプリケーション）の例を概観するとともに、著者らが研究開発に取り組んできた IoT 向けエッジコンピューティング技術について述べる。

2　IoT アプリケーション

2.1　IoT アプリケーションの分類

図 1 は、IoT アプリケーションの大まかな分類を示している。

IoT サービスにおける最初の段階は、「可視化（Visualization）」である。センサー等から観測データを収集し、収集された観測データを必要に応じて分析したうえで、人が目で見て理解できる表示形式に変換し、提示する。あるいは、観測データの処理結果を付加情報として別の情報と組み合わせて提示する。

次の段階は、「アクチュエーション（Actuation）」である。ICT サービスとして、何らかの分析や判断に基づき、実世界にある端末に対する通知や、実世界にあるモノの制御を行う。アクチュエーションを含む

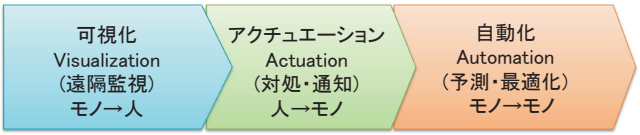

図 1　IoT アプリケーションの分類

IoT は、CPS（Cyber-Physical System）とも呼ばれ、研究開発が盛んに行われている。

更なる次の段階は、「自動化（Automation）」である。この段階は、人が介在することなく、モノ同士が自律的にやりとりを行い、街や家庭の至るところで安全性や効率性を保つよう、モノが自動的に動作する状態を指す。M2M（Machine-to-Machine）のコミュニケーションが行き渡った段階であり、いわゆるロボット化が進んだ世界であると言える。過去の蓄積に基づく予測や、何らかの知性に基づいたアクチュエーションにより、街が自律的・自動的に制御され、人々の活動の安全性向上や最適化がなされる。自動車の自動運転において、人が介在しない自動運転レベル（SAE J3016 におけるレベル 3 以上）は、この段階にあると言える。

2.2　IoT アプリケーションの具体例

前記の各過程において、今後本格的な商用化が期待される将来 IoT アプリケーションの例を以下にいくつか挙げる。

- AR、MR

AR（Augumented Reality）は、現実世界の映像に対し、新たな付加情報等を重畳させて表示するアプリケーションである。センサー情報や、カメラに映った映像情報を分析することで、現実世界にマーカ等を設定することなく、現実世界の映像に対して仮想的な映像を重畳させて表示する AR はマーカレス AR と呼ばれる。典型的なアプリケーションは、案内情報をヘッドマウントディスプレイやカメラつきスマートフォン上のライブ映像に重畳提示するナビゲーションである。実世界に存在する端末へ、端末や実世界の状態に応じた情報提示が必要なアプリケーションであり、正確な情報提示位置を維持するには高度な精度の分析と、高い応答速度が必要である。米 Microsoft 社が MR（Mixed Reality）と呼んでいる、現実世界と仮想世界を混合させる体験を提供するプラットフォームも既に登場しており、商用化は始まっていると言える。AR、MR は、前記分類における Visualization がリアルタイムに実行される形態ととらえられる。

- スマート家電

スマートフォンやスマートスピーカーと連携して、遠隔操作をすることができる家庭内電化製品がいくつか商品化されている。例えば、音声コマンドによって点灯状態や明るさを制御できる照明等がある。また、利用者がイベントに応じて実行すべきコマンドをあらかじめ定義可能な IFTTT のようなサービスも登場している。これにより、例えば、人感センサーで人が検出されたときに自動的に照明をつける、カーテンを開けるといった動作も定義可能となる。このようなスマート家電は Actuation の典型的なアプリケーションであるといえる。

- スマート交通

IoT による交通の高度化は、大きな期待が寄せられる分野である。運転者により制御された、または、自律移動する移動体（自転車・車・公共移動手段等）において、街を歩く人の安全を守るためのナビゲーションや運転制御を状況に応じて行う。例えば、スマートフォンの位置センサーや街のセンサー等から、歩行者がそのままの動きをした場合に事故が起こると予見される場合、移動体の運転者への安全運転情報の提供や、移動体そのものの減速制御等を行う。こうした動作は、前記分類における Actuation、Automation に相当する。

- 自走型ロボット

自律的に移動することができる自走型ロボットは、Automation の典型的なアプリケーションと言える。例えば、通路脇に設置されたカメラ映像と、ロボットが持つセンサーによって、ロボットの周囲の状態を認識し、適切な方向・速度でロボット自身が移動制御を行う。ロボットが移動するか否かの判断は、フィールドの最新状態を計測・分析しつつロボット間で協調しながら決定する。

- セキュリティ

実世界との接点を持つ IoT にとって、物理的・ソフトウェア的な不正や攻撃を排除し、正しい動作制御を行うための信頼性の確保は大きな課題となる。社会システムを健全な状態に保つには、不正や攻撃等が検知されたとき、他への影響を最小化し、対処を素早く実行することの重要性が高い。素早い対処には、人の介在を待つことなく、不正な動作が検出された端末、デバイス、ネットワークについて自動的に排除・フィルタする等の処理を行う必要がある。こうした動作は、前記分類における Actuation、Automation に相当する。

3　エッジコンピューティング

3.1　エッジコンピューティングの概要

2.2 で示した IoT アプリケーションは、例えばセンサーデバイス単体で完結するものではなく、ネットワークを介して通信が必須となるものが多くを占める。高度な処理を必要とするアプリケーションでは、処理全体あるいは一部をクラウド上で実行するコンピューティング形態が多くとられる。例えば、AR、MR では、アプリケーションの高度化による情報処理の複雑化に伴い、端末単体で処理すると、消費電力が大きくなる。

モバイルが基本となる IoT においてはバッテリーの持続時間は長いことが望ましく、消費電力は小さくしたい。また、例えば他端末との位置関係を表示する必要がある場合、全ての処理が端末のみでは完結しない。こうした場合、処理全体あるいは一部を端末上ではなくクラウド上で実行する形態を取ることになる。

このとき、まず、デバイスが生成するデータをクラウドに送信する際に生じるトラヒックが課題となる。大量数のセンサーを利用した広域のデータ分析が必要なアプリケーションでは、一つひとつのデータ量は小さくとも、クラウドにデータを収集する際、トラヒックの集中により、ネットワーク帯域の不足が生じ得る。また、例えば自動車で映像を取得し、分析するアプリケーションでは、上流ネットワークが無線公衆網となり、取得された映像を全て送信するには通信帯域が足りない場合があり得る。一方、ネットワーク帯域の問題のみならず、クラウドへのデータ送信そのものが問題となることもある。例えば、パーソナルデータを扱うアプリケーションでは、プライバシーポリシー上、統計処理を行わねば組織の外へ送信することが許可されないことがある。この場合、クラウドへセンサーデータをそのまま送信できない。

もう 1 つの大きな課題は、モノとクラウドとの間の物理的距離が長いことに起因する応答遅延の課題である。例えば、AR では 40 ms 以内の応答時間が望ましいとされているが、一般にクラウドでは、国内では 100 ms 以内、国外では通常 100 ～ 200 ms 程度の伝搬遅延があり、処理遅延を含まない通信遅延だけでも要求される応答時間を超えてしまう。物理的な制御が関わる Actuation や Automation のアプリケーションでは、イベントの発生から実際の制御までの遅延によって、致命的な事故も発生し得る。また、伝搬遅延が大きいと、TCP のように送受信確認がある通信プロトコルでは通信帯域が減少する。

こうした課題を解決可能とする方策として、端末上、あるいは、端末に物理的に近いネットワーク装置周辺（エッジ）に、計算処理やデータ保存等を行うリソースを設置し、クラウド上の処理の一部を実行する「階層化クラウドアーキテクチャ」（以下、階層化クラウド）が提唱され、様々な要素技術の研究開発が進められている。図 2 は、階層化クラウドのモデルを示している。このようにエッジで計算処理を行うコンピューティング形態はエッジコンピューティング（またはフォグコンピューティング）と呼ばれる。エッジコンピューティングにより、フィルタリング処理や統計処理等を端末に近い位置で行うことが可能となり、クラウドに至る経路で生じるトラヒックの選別や削減が可能となる。また、端末に近い位置にあるストレージや計算リソースを利用できれば、伝搬遅延が削減され、通信帯域や応答性能が向上する。

ETSI（European Telecommunications Standards Institute）では MEC（Multi-access Edge Computing）の標準化が進められており、公衆無線の基地局に計算処理を行うサーバ等を設置する設計がなされている。2021 年には米国で 16 万以上の 5 G 向け MEC 設備が配備され、2026 年にはその数は 3 倍以上に増加するとの予測もある [1]。国内では、NTT 未来ねっと研究所が EAW エンジン（Edge Accelerated Web Engine）

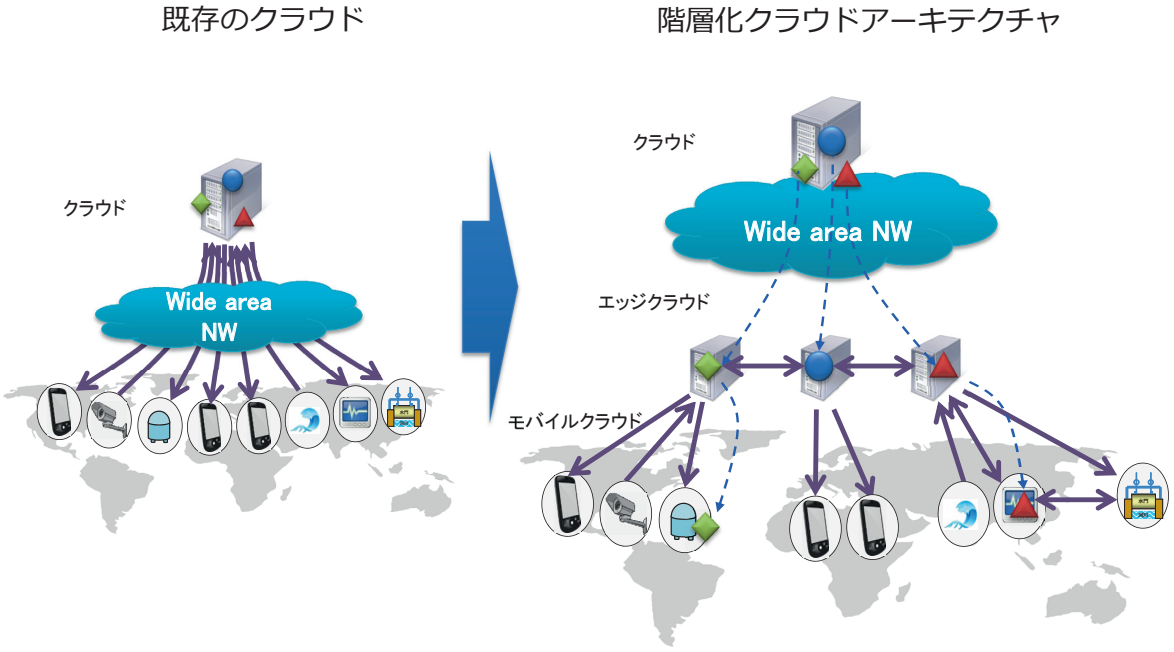

図 2　階層化クラウドアーキテクチャ

と呼ばれるエッジ実行型のウェブアプリ実行エンジン等を開発しており [2]、エッジコンピューティングの商用化を進めている。計算リソースの配置場所としては、家庭内ルータや、エッジルータ、ISP ルータなど複数の候補があげられている。

Open Edge Computing Initiative [3] では、米カーネギーメロン大学を中心に Elijah と呼ばれる Cloudlet（エッジコンピューティングのためのボックス型サーバ）の開発を行い、Living Edge Lab と呼ばれるオープンテストベッドを構築して実験を進めている。

3.2　IoT エッジコンピューティングの技術的課題

2 で述べたとおり、エッジコンピューティングの商用化に向けた動きは活発化しているが、IoT 向けのエッジコンピューティングの本格的な普及には様々な技術的課題がある。以下に技術課題の例をあげる。

1. 設備コストの課題

まず、エッジに配備するリソース（エッジリソース：計算サーバ、ストレージ等）の設置・運用コストが課題となる。小さい伝搬遅延を保ってひとつのエッジリソースが受け持つことができる地理的範囲は限られる。したがって、あるサービスプロバイダが広いサービスエリアをカバーするには膨大数のエッジリソースの設置・運用を行う必要があり、金銭的・管理上のコストは大きくなる。このような設備では、複数のサービスプロバイダが共通の物理的インフラストラクチャを共有し、仮想化されたエッジリソースとして利用可能な形態が理想的である。IoT におけるエッジリソースは、通信事業者等が提供するネットワーク設備内のみならず、WiFi の基地局のように、店舗等に設置されるアプライアンスが付加価値として内蔵する展開も想定されるため、このようなエッジリソース部分まで含めた仮想化が必要となる。

2. 分散化に伴う課題

エッジリソースは、必ずしもネットワークやデータセンターに精通した運用者が運用を行うわけではない。したがって、24 時間の連続稼働を保証することはできず、停電や電源操作等により任意のタイミングでリソースが利用できない状態となる可能性を考慮する必要がある。分散エッジリソースが自律的に運用される状況の下、情報サービスとして可用性を高く保った運用をいかに行うかは課題である。

3. 動的な状況変動の課題

IoT では、データソースが自動車やスマートフォン等の移動体である場合も考えられ、エッジリソースあたりの処理すべきデータソース数や処理対象数は時間に応じて変化する。また、例えば、映像に映った人を分析するアプリケーションでは、カメラに映る人の増減に応じて、データ量や処理量が大幅に変化する。エッ

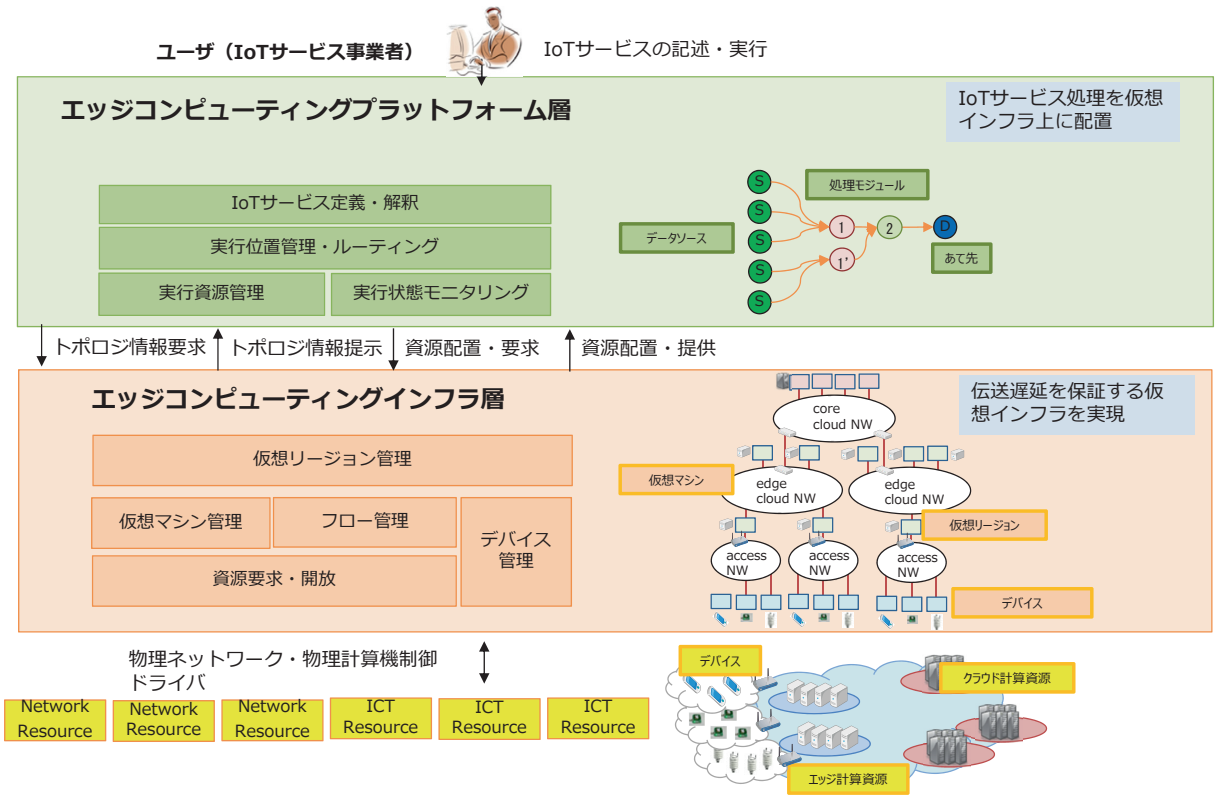

図 3　IoT エッジコンピューティングのアーキテクチャ

ジコンピューティングシステムとしては、こうした状況変動に応じてリソースを階層化クラウド上にいかに確保し、サービスを継続するかが課題となる。

4 IoT エッジコンピューティング

我々は、前節で示した技術課題に対処し、IoT 向けのエッジコンピューティングを実現するための基盤となる技術の研究開発を進めている。本節では、我々が提案するアーキテクチャ及び提案アーキテクチャを構成する要素技術について述べる。

4.1 IoT エッジコンピューティングのアーキテクチャ

我々は、分散配置された多数のデバイスからのデータを低遅延で処理するための仮想的なエッジコンピューティング環境を同一のハードウェア環境上に複数同時に提供可能な基盤を実現するインフラ層と、そのインフラ層を用いて上位 IoT サービスに必要なデータ処理を実行するプラットフォーム層から成る 2 階層のアーキテクチャを提案している。

図 3 は、それぞれの階層が持つ大まかな機能要素と、プラットフォームレイヤ PaaS（Platform-as-a-Service）とインフラストラクチャレイヤ IaaS（Infrastructure-as-a-Service）の構成を示している。

4.2 IoT エッジコンピューティングインフラストラクチャ

本節では、インフラストラクチャレイヤに相当する IaaS の機能要素について述べる。前記、エッジコンピューティングにおける「設備コストの課題」の解決を図るものである。

IoT サービス事業者にとっては、物理インフラ所有者から取得した仮想化されたエッジリソース（以下、VM）について、許容できる遅延内で無線基地局を通してエンドユーザデバイスと通信可能であることが要求である。一方、物理インフラ所有者にとっては、特に物理サーバの消費電力を抑え、インフラ運用コストを最小化することが要求である。これらを実現するには、無線基地局から許容遅延内で通信可能なエッジサーバの空き状況に応じて、IoT サービス事業者の顧客となるエンドユーザデバイスのサービスに必要最低限の VM 台数を見積もり、配置することが必要である（図 4）。

既存研究におけるアプローチの 1 つ文献 [4] では、インフラ所有者が物理ネットワークトポロジとエッジサーバの設置場所を公開した上で、IoT サービス事業者が設置場所ごとに要求 VM 台数を直接指定するアプローチが取られている。このアプローチでは、インフラ所有者が IoT サービス事業者に開示する情報が膨大になる。さらに、省電力性の観点で望ましくない場所への VM 要求を回避するため、IoT サービス事業者とインフラ所有者の間で複雑なネゴシエーションが必要になる。このため、IaaS-PaaS 間のインタフェースでは、膨大かつ複雑な情報交換を強いられる問題がある。既存研究における別のアプローチ [5] で は、無線基地局とエッジサーバの組合せをあらかじめ決定し、動作中に変更することはない。無線基地局に接続するユーザ数予測やネットワーク上でのエッジサーバ設置コスト、許容遅延等を考慮して、無線基地局とエッジサーバ設置場所の組合せを決定する。無線基地局に接続したエンドユーザデバイスに対しては、常に組合せになっているエッジサーバを用いてサービスを提供する。このため、IoT サービス事業者とインフラ所有者のやり取りは簡素にできる。ただし、このアプローチでは、想定する許容遅延を小さくすると、できるだけ無線基地局の近くにエッジサーバを設置する必要があるため、無線基地局とエッジサーバ設置場所が 1 対 1 の組合せになる。このため、全体の稼働 VM 数が多くなり、消費電力が大きくなる問題がある。

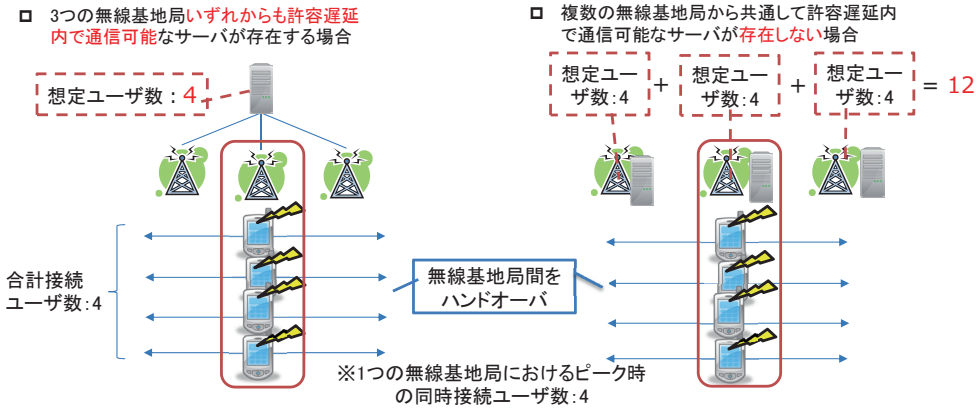

図 4　VM 要求に必要な想定ユーザ数の例

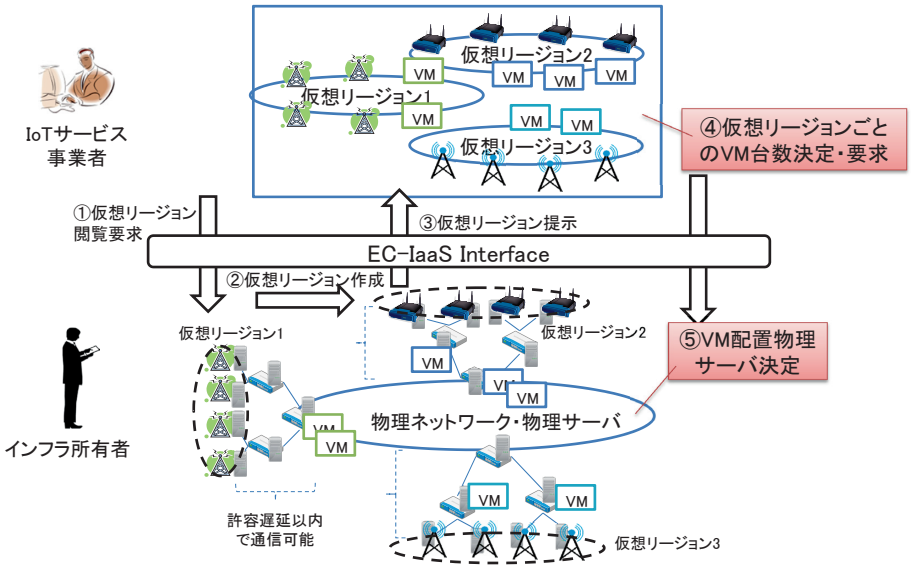

図 5　仮想リージョンに基づく VM 取得手順

著者らはこの問題に対応するため、インフラ資源配置を抽象化する「仮想リージョン」と階層間インタフェースの提案を行った（図 5）。「仮想リージョン」は具体的には、許容遅延内で通信可能で、利用可能なエッジサーバを共通に持つ無線基地局のグループである。IoT サービス事業者の許容遅延とエッジサーバの空き状況を基に、インフラ所有者が作成する。IoT サービス事業者は、ネットワークのトポロジやエッジサーバの配置場所を参照する代わりに、仮想リージョンを参照することで、グループに属する無線基地局の接続ユーザ数のみに基づき、要求 VM 台数を決定することができる。仮想リージョンごとに要求された VM 台数に基づき、インフラ所有者は、省電力性も考慮した VM 配置を行う。提案方式により、従来手法では両立できなかった、階層間の制御メッセージ量の削減と省電力性が両立できることを確認した [6]。

4.3　IoT エッジコンピューティングプラットフォーム

本節では、プラットフォームレイヤに相当する PaaS の機能要素について述べる。前記、エッジコンピューティングにおける「分散化に伴う課題」、「動的な状況変動の課題」に対応するものである。本プラットフォームは IoT のデータフローを処理するための機能要素について主に扱っている。

4.3.1　データフローフレームワーク

IoT のアプリケーションを簡単に作成することを目指し、実行するデータ処理の流れを、「データフロー」のパラダイムによって定義するフレームワークがいくつか提案されている [7][8]。これらのフレームワークには、データフローを構成する処理のコンポーネント

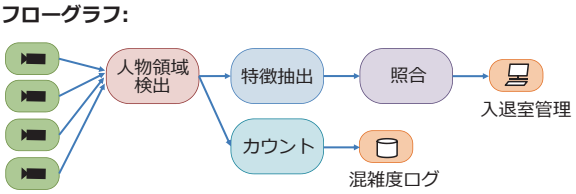

図 6　データフローの例

の詳細を知らずとも、一連の処理をフローとして定義できる利点がある。

しかし、既存のデータフローフレームワークは、一度データフローを定義すると、基本的に次にデータフロー定義者が更新するまで、フローの構造は変化しない。したがって、エッジコンピューティング環境において想定すべき動的な状況変動に対応するには、データフロー定義者がデータフローの定義を更新するか、ピーク時に必要なネットワークや計算機の資源を常に確保し、それらを用いるよう、あらかじめデータフローを定義しなければならない。

関連技術として、クラウド上で連続的に生成されるストリームデータを分散処理するフレームワークもいくつか提案されている [9][10]。これらは、専用プログラムの記述が必要であり、データフローフレームワークのように個々の処理の詳細に立ち入らずに全体の処理フローを定義できない。また、基本的にマスタスレーブ型でシステムが構成されるため、データ送信時にクラウド上のサーバへの問い合わせが必要となり、伝送遅延によって配信遅延が大きくなる問題や、問い合わせの集中に伴い負荷が増大してしまう問題がある。

著者らはこれら課題に対処すべく、データの処理を行うコンポーネント間の接続を Topic-Based Pub/Sub

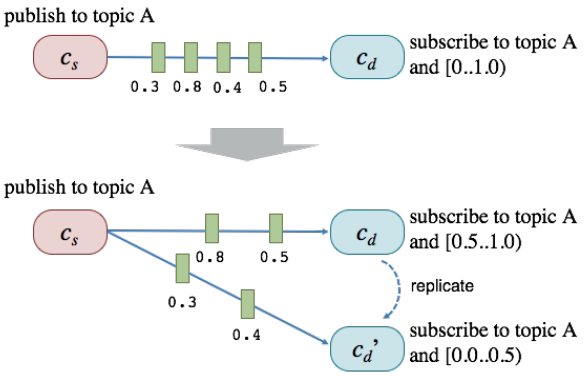

図 7　データフローグラフの更新動作

(TBPS) によって定義する前提のもと、データフローの定義を変更することなく、ネットワークや計算機の過負荷状態に応じてデータフローの構造を動的に変化させる動的データフロー処理プラットフォームを提案している [11]。

4.3.2　動的データフロー処理

図 6 は、データフローの例を示している。図は、2 つのアプリケーションを含むグラフの例である。このグラフは、映像から人の領域を切り出す「人物領域検出」のコンポーネントがビデオカメラからの入力を受信し、「特徴抽出」のコンポーネントで特徴を取り出し、「照合」のコンポーネントがマッチングを行い、特徴が合致した場合に通知を行うというアプリケーション (入退室管理) と、「人物領域検出」の出力数を「カウント」のコンポーネントで数え、時間ごとに記録するアプリケーション (混雑度ログ) を含んでいる。

データフローを実行する際、動的な状況変動に対応するには、プロセスの実行位置の変更やプロセスを実行するリソース数の増減が必要となる。こうした変更にデータフローを対応させるため、提案プラットフォームでは、TBPS の送受信データに『 index 』と呼ぶ属性を追加する。index は一次元のスカラー値である。

publisher は、トピックに加えて index を付与してデータを publish する。subscriber は、topic と index の範囲を指定して subscribe する。

図 7 は、データフローのグラフの更新動作を示している。この例では、topic A が c_s から c_d へのデータ送信に対応している。この例ではデータが、0.3, 0.8, ... を index として持っている。図上では、c_d は [0, 1.0) に subscribe しているため、全てのデータは c_d に到達する。図下では、処理リソース数の変化によって、あらたにあて先コンポーネント c_d' が追加され、subscribe する index が [0, 0.5) と [0.5, 1.0) に分割されている。これによって処理が振り分けられる。

これら index の割当てはプラットフォームのレベルで実行されるため、データフローを定義する利用者は、動的な状況変動に対応したデータフロー構造の変化を意識する必要はない。また、提案プラットフォームでは、このルーティング機構を構造化オーバレイを用いて実装することで、各処理のコンポーネントにおけるあて先を、クラウドに問い合わせることなく自律的かつ効率的に実行可能とした。

4.3.3　実装

本研究では、実装に筆者らが提案している双方向キー順序保存型オーバレイネットワーク Suzaku [12] を用いて index に基づく振り分け機構のプロトタイプを実装した。Suzaku は、多数の連続したキーが挿入・削除された場合も最大検索ホップ数がほぼ $O(\log_2 n)$ に抑えられ (n はノード数)、スケーラビリティ、Churn 耐性に優れている。Suzaku はオーバレイフレームワーク PIAX [13] 上に実装されており、プロトタイプも PIAX を用いて実装した。プロトタイプは、基本機能の実装を完了している。Suzaku は、ベースとなるノードの追加削除のアルゴリズムとして DDLL[14] を用いており、定期的な更新によるスタビライズ動作を必要とせず、短時間で構造変更を行うことができる。この特徴は、データフローの構造変更に対する追従性の実現に適している。

7　おわりに

本稿では、近年注目を集めている IoT アプリケーションを概観し、それらに適した新たな将来コンピューティングアーキテクチャとして注目されるエッジコンピューティング技術や関連技術について述べた。また、将来の IoT エッジコンピューティングに向けた要素技術の研究開発活動の取組を紹介した。

本稿で示したインフラストラクチャ、プラットフォームは、NICT が提供するテストベッドに実装し、様々なアプリケーションに適用して有効性を検証していく予定である。

【参考文献】

1　GillottResearch,"The Business Case for MEC in Retail: A TCO Analysis and its Implications in the 5G Era," GillottResearch Inc., June 2017.
2　N. Takahashi, H. Tanaka and R. Kawamura, "Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing," Mobile Cloud 2015, pp.233-234, March. 2015.
3　Open Edge Computing Initiative, "Open EDGE Computing," http://openedgecomputing.org/, 参照, Aug.7.2018.
4　D. Krishnaswamy, R. Kothari, and V. Gabale, "Latency and policy aware hierarchical partitioning for nfv systems," NFV-SDN 2015, pp.205-211, Nov. 2015.
5　A. Ceselli, M. Premoli, and S. Secci, "Cloudlet network design optimization," 2015 IFIP Networking Conference, pp.1-9, May 2015.

6 H. Yamanaka, E. Kawai, Y. Teranishi, H. Harai, “Proximity-Aware IaaS for Edge Computing Environment,” Proc. of IEEE International Conference on Computer Communication and Networks (ICCCN 2017), pp.1-10, July 2017.
7 Node-RED project,“Node-RED,” http://nodered.org/, 参照, Aug.7.2018.
8 A. Pintus, D. Carboni and A. Piras, “The anatomy of a large scale social web for internet enabled objects,” The Second International Workshop on Web of Things, pp.1-6, June 2011.
9 M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker and I. Stoica, “Spark: Cluster computing with working sets,” HotCloud, 10, 10-10, pp.1-7, June 2010.
10 P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi and K. Tzoumas, “Apache flink: Stream and batch processing in a single engine,” http://asterios.katsifodimos.com/assets/publications/flink-deb.pdf, 参照, Aug.7.2018.
11 Y. Teranishi, T. Kimata, H. Yamanaka, E. Kawai and H. Harai, “Dynamic Data Flow Processing in Edge Computing Environments,” Proc. of IEEE Computer Software and Applications Conference (COMPSAC 2017), pp.935–944, July 2017.
12 安倍 広多, 寺西 裕一,“高い Churn 耐性と検索性能を持つキー順序保存型構造化オーバレイネットワーク Suzaku の提案と評価,” 信学技報, vol.116, no.362, pp.11–16, Dec. 2016.
13 Y. Teranishi, “PIAX: Toward a Framework for Sensor Overlay Network,” Proc. of 6th IEEE Consumer Communications and Networking Conference 2009 (CCNC 2009), pp.1-5, Jan. 2009.
14 K. Abe, M. Yoshida, “Constructing Distributed Doubly Linked Lists without Distributed Locking,” Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P 2015), pp.1-10, Sept. 2015.

寺西裕一　（てらにし　ゆういち）

ネットワークシステム研究所
ネットワーク基盤研究室／
総合テストベッド研究開発推進センター
テストベッド研究開発運用室
研究マネージャー
博士（工学）
分散システム、オーバレイネットワーク、センサーネットワーク、応用システム

山中広明　（やまなか　ひろあき）

総合テストベッド研究開発推進センター
テストベッド研究開発運用室／
ネットワークシステム研究所
ネットワーク基盤研究室
研究員
博士（情報科学）
エッジコンピューティング、ネットワーク仮想化、Software-Defined Networking

河合栄治　（かわい　えいじ）

総合テストベッド研究開発推進センター
テストベッド研究開発運用室
室長
博士（工学）
エッジコンピューティング、IoT、ICT テストベッド