Vol.2015-MBL-77 No.22
Vol.2015-ITS-63 No.22
2015/12/3

多様なIoTデータストリームをクラウドレスで分散処理するミドルウェアの設計

中村 優吾[1,a)]　諏訪 博彦[1]　荒川 豊[1]　山口 弘純[2]　安本 慶一[1,b)]

概要：IoT デバイスが急速に進歩し，実世界のあらゆる情報をセンシングできるようになりつつある．それに伴い，様々なデバイスから時々刻々と生成される IoT データストリームの即時的な利活用が求められている．本研究では，生成されたデータストリームを，IoT デバイスが持つ計算資源を用いて，その場で素早く処理する「地産地処」をコンセプトに，IoT デバイス群の上位で動作する実時間処理ミドルウェアを設計する．本ミドルウェアには，(1) アプリケーションから要求されるタスクを分割しながら分散で実行する機能，(2) IoT データストリームを分散処理のためデバイス間でオンデマンドに流通する機能，(3) 複数のデータストリームをオンラインで分析する機能，(4) 異種センサ/アクチュエータのシームレスな連携を実現する機能を設計，実装する．本稿では，機能を限定したミドルウェアのプロトタイプを Raspberry Pi 上に実装し，動作検証した結果を報告する．

1. はじめに

近年，センサやアクチュエータといった小型デバイスをインターネットで繋ぐ Internet of Things（IoT）の概念が注目を集めている．IoT 技術の発展に伴い，環境に分散された多数の IoT デバイスから，実世界のあらゆる状況を示すセンサデータや動画像といった IoT データストリームを取得可能になりつつある．そのため，これらの IoT データストリームを戦略的に活用し，個人生活の質や地域社会の魅力を向上する状況適応型 IoT サービスの創造が求められている．

これまで，IoT やビックデータの分野では，多種多様なセンサデータを収集し，集約する技術，集約された大量のデータを効率良く処理する技術を中心に研究開発が行われてきた．そのため，既存の IoT システムの多くは，図 1 左上のように，クラウド中心のシステム形態を採用している．そして，IoT デバイスからのデータストリームは，クラウド側の大規模ストレージに集約，蓄積され，膨大な時間と強力な計算パワーを費やして分析された後に，実世界に還元されているのが一般的となっている．

[1] 奈良先端科学技術大学院大学　情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
[2] 大阪大学　大学院情報科学研究科
Graduate schoole of Information Science and Technology, Osaka University
[a)] nakamura.yugo.ns0@is.naist.jp
[b)] yasumoto@is.naist.jp

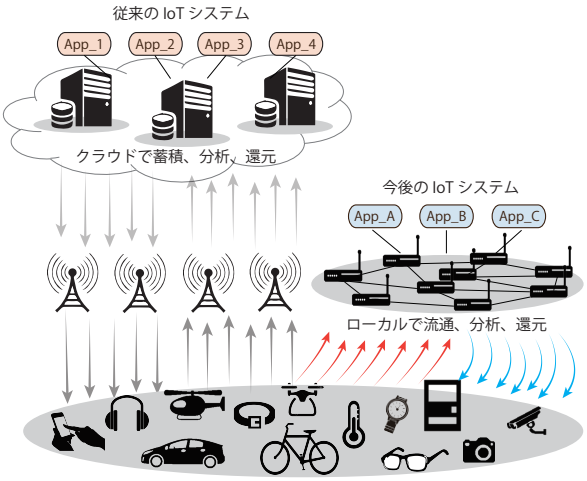

図 1　IoT システムのパラダイムシフト

しかし，このようなクラウド集中型のシステムは，通信頻度が高く，リアルタイム性が求められる IoT サービスを実現する場合に，逐次生成される大量のデータを遠隔に伝送する必要があり，遅延時間や通信効率，コストの観点で問題を抱えている．また，地域や個人に特化した IoT サービスを実現する場合には，IoT データストリームの中に，地域環境や地域住民に関する情報が含まれるため，そのようなデータはプライバシーやセキュリティの観点から特定の領域内に留めておくことが望ましい．

今後，IoT デバイスが更に発展し，デバイス自体の処理性能やメモリ容量が向上することを考えると，図 1 右上のようにローカルのデバイス上など，データの発生源により

近い場所で IoT ストリームデータを分散処理し，より効率的かつ効果的に活用することが求められる．これらの動向を踏まえて，情報流検討委員会 [1] では，情報流 [2] と呼ばれる IoT データストリームを発生源から効率的に分散処理し，データの流通と分析を最適化する IoT プラットフォーム IFoT(Information Flow of Things) を提言している [3]．

本研究では，IFoT に基づき，IoT デバイスの計算能力を活用し，その場で素早く処理する「地産地処」をコンセプトに，IoT デバイスの上位で動作し，IoT データストリームを実時間で処理する IFoT ミドルウェアの開発を行っている．IFoT ミドルウェアでは，(a) IoT デバイス群の上で複数のアプリケーションが計算リソースを共有しながら動作する環境，(b) 各アプリケーション内で生成された価値ある情報（コンテンツ）を 2 次利用，3 次利用（高次利用）できる環境の実現に向けて，(1) アプリケーションから要求されるタスクを分割しながら分散で実行する機能，(2) IoT データストリームを分散処理のためデバイス間でオンデマンドに流通する機能，(3) 複数のデータストリームをオンラインで分析する機能，(4) 異種センサ/アクチュエータのシームレスな連携を実現する機能を提供する．

本稿では，2 章で関連研究を述べ，3 章で IFoT ミドルウェアの応用と要件を示す．4 章で IFoT ミドルウェアの詳細設計およびプロトタイプの実装を示し，5 章でプロトタイプを動作検証した結果を示す．そして，最後に 6 章でまとめとする．

2. 関連研究

本章では，既存の IoT プラットフォームについて述べると共に，IFoT の実現に関連のあるストリーミング処理技術，データ分析技術，通信プロトコルについて示す．

2.1 IoT プラットフォーム

これまで，IoT サービスを実現する際のシステム構築コストの低減や，異種のデバイスやシステムの連携を促進することを目的として，様々な IoT プラットフォームが提案されている [4]．これらの IoT プラットフォームのほとんどが PaaS（Platform as a Service）型かつクラウドベースのアーキテクチャを採用している (Arkessa[5], Axeda[6], ThingSquare[7], Thingworx[8], WoTkit[9], Xively[10])．一方, 分散型の IoT プラットフォームとしては，OpenIoT[11], LinkSmart[12] が存在する．いずれのプラットフォームも，多様な IoT データストリームを集約する機能の提供やサービス構築の簡略化に留まっており，クラウドレスでの情報集約や実時間処理の実現には至っていない．また，多様な IoT デバイスを統括する IoT ゲートウェイの研究 [13][14][15] が存在するが，これらはクラウド上のサーバとの連携が前提となっており，集約したセンサデータの高度な分析処理をゲートウェイ上で行い，実環境に還元することは想定されていない．

我々の知る限り，エッジコンピューティング [16] やフォグコンピューティング [17] など，データを近くに集約して処理する概念は存在するものの，IFoT のように IoT デバイスそのものを計算リソースとみなし，データの発生源から，リアルタイムに分散処理することを目指した IoT プラットフォームは未だ存在していない．IoT デバイスに計算機能を分散できれば，ビル等の大型施設で数千台の IoT デバイスから生成されるデータストリームをエッジサーバに集約する場合にも，不要データのフィルタリングやメタデータの付加により，集約側の作業負荷を下げることが可能になる．

2.2 ストリーミング処理技術

連続的に発生するセンサデータストリームを入力として，イベントに応じて瞬時に処理を実行するストリーミング処理技術が研究されている．ストリーミング処理技術は，その性質からデータが入力されてから結果が返るまでの処理時間を短縮することが求められ，これまでいくつかの時間短縮手法が提案されている．これらの手法は，(1) 複数に分散したデータの発生源を一つに集約してから処理することで，個々の通信におけるオーバーヘッドを削減し，全体の通信にかかる時間を短縮する手法 [18][19] と (2) データを処理するデバイスを分散化し，処理負荷を分散して並列に実行することで全体の処理にかかる時間を短縮する手法 [20][21] という 2 種類のアプローチに大きく分類することが可能である．本研究で提案するミドルウェアは，データの発生源と処理ノードの両者が分散している環境で動作する．そのため，データ発生源の集約による通信時間の短縮と処理負荷の分散化による処理時間の短縮という双方のアプローチを適用し，IoT データストリームの処理を高速化を図る．

2.3 データ分析技術

ビックデータや IoT の注目に伴い，データ分析技術の需要が高まり，機械学習といった高度なデータ分析を実現する様々なフレームワークが研究されている．機械学習のフレームワークは，全てのデータをまとめて入力し，時間をかけて一括で処理するバッチ処理型と，発生するデータを 1 つ 1 つ入力し，即座に処理するオンライン処理型の二つに大きく分類することが可能である．バッチ処理型のデータ分析フレームワークとしては，Mahout[22]，WEKA[23] が存在する．一方，オンライン処理型のフレームワークとしては，SAMOA[24]，MLlib[25]，Jubatus[26]，Oryx[27] が存在する．本研究では，時々刻々と生成される IoT データストリームの即時的な分析の実現を目指し，オンライン学習と並列分散処理という二つの機能を兼ね備えた Jubatus を用いて IFoT ミドルウェアのプロトタイプを実装する．

2.4 通信プロトコル

IoT や M2M（Machine-to-Machine）の通信特性に合わせた軽量の通信プロトコルとして，出版/購読型メッセージングモデルを採用した MQTT（Message Queue Telemetry Transport）[28] や HTTP を簡略化した UDP ベースの CoAP（Constrained Application Protocol）[29] が提案されている．他にも，HTTP を双方に拡張した Websocket[30] や低消費電力 PAN において IPv6 通信を行うための 6LowPAN（IPv6 over Low power Wireless Personal Area Networks）[31] などが研究開発されている．本研究では，できる限り標準的なプロトコルを採用して，IFoT ミドルウェアの開発を行う．

3. ミドルウェアの要件

3.1 想定するアプリケーションと機能要件

IFoT ミドルウェアは，クラウドレスで動作し，センサデータストリームを蓄積することなく，発生源に近い場所で分散処理することを特徴としている．そのため，センシングからアクチュエーションのサイクルが早く，リアルタイム性の求められるアプリケーションとの親和性が高い．ここでは，IFoT ミドルウェアが想定するアプリケーション例を示し，その実現に向けて求められる機能を述べる．

(1) 高齢者見守りアプリケーション

居住空間に複数配置された音センサやモーションセンサからのセンサデータストリームから高齢者の行動を認識し，異常がないかを常時モニタリングする．これにより，真に危険な状態（転倒後動けなくなるなど）を瞬時に検出する即時的なオンサイトトラッキングサービスを実現する．

このアプリケーションを実現するには，複数のセンサから生成される IoT データストリームを常に解析し，行動に異常がないかどうかを判定する必要がある．つまり，複数のデータストリームを複数のデバイス間で相互に流通しながら，実時間で高度なデータ分析を実現する機能が求められる．

(2) 空間状況に応じた快適な家電制御アプリケーション

オフィスや居住空間に複数配置された温湿度センサ，照度センサ，音センサ，人感センサから生成されるセンサデータストリームから空間の状況をモニタリングする．これにより，空間の微量な変化（室温や湿度，明るさ，人々の人数や行動）に応じて適切な空調や照明を制御するアプリケーションを実現する．

このアプリケーションを実現するには，複数のセンサから集められる情報を組み合わせて空間の状況を推定する必要がある．また，推定結果に応じて，照明や空調装置といった複数のアクチュエータを瞬時に制御する必要がある．つまり，相互の情報流通や高度なデータ分析に加えて，多種多様なセンサ/アクチュエータのシームレスな連携を実現する機能が求められる．

(3) 景観情報や混雑度に応じた移動支援アプリケーション

車載カメラやスマートフォン，環境に設置されている複数のセンサを用いて，景観の良さや群衆の移動，人流の騒音をセンシングし，街の景観が優れているスポットや各地点における現在の混雑状況を推定する．これにより，いま景観の良い場所やリアルタイムな混雑状況を考慮した移動ナビゲーションを実現する．

このアプリケーションを実現するには，桜センサ [32] や混雑センシング [33] など個々に独立しているアプリケーションが収集した価値のある情報を 2 次利用しながら，リアルタイムな情報をローカルで組み合わせて配信する必要がある．つまり，IoT デバイス群の上で複数のアプリケーションが同時に動作しながら，各アプリケーションで得られた価値のあるデータを相互に共有する機能が求められる．

3.2 課題と基本方針

これまでにも，センサデバイスが生成した情報をローカルのサーバで処理を行うクラウドレスなアプリケーションは，提案されている．しかし，それらの多くは，アプリケーションとデバイスが垂直に統合されたシステム形態となっており，異なるシステム間で，デバイスを共有したり，収集したデータを再利用することが困難であった．そのため，システム開発者は，センサ/アクチュエータといったデバイスの準備から取り掛かる必要があり，初期構築のコストが高いという課題がある．一方，無線センサネットワークの研究では，ローカルに存在するセンサデバイス群上で，複数のアプリケーションを実行可能なミドルウェア [34] やプラットフォーム [35] が提案されている．しかし，これらは，専用デバイス間での連携しか考慮されておらず，実行できる処理も情報のセンシングやデータ伝送などに限られており，センシングデータの異常値検出やオンラインクラスタリングといった高度な分析処理には対応していない．

IFoT ミドルウェアでは，環境に偏在する IoT デバイス自体の計算資源やそこから生成される IoT データストリームを共通のリソースとみなし，それらを多様なアプリケーションで共有できる水平分散型のプラットフォームの実現を目指す．そのために，タスクを分割しながら複数のデバイスで並列に実行する機能や，デバイスの要求に応じて情報をダイレクトに流通する機能を提供する．また，センサデータをオンラインで分析する機能や，異種センサ/アクチュエータを相互に連携するための機能を提供する．

4. 提案するミドルウェア

4.1 ミドルウェアの動作環境

本研究で想定する IFoT ミドルウェアの動作環境を図 2 に示す．本稿では，(a) 実環境の情報を収集するデバイス

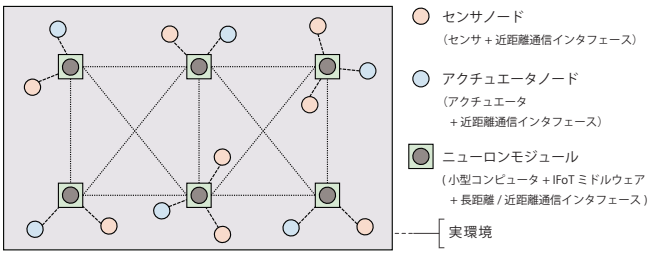

図 2 想定する動作環境

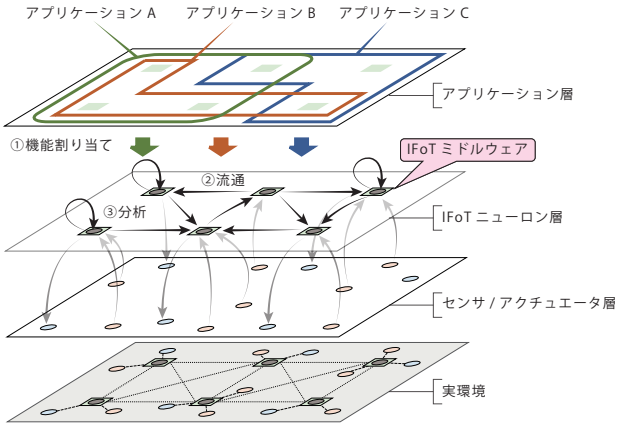

図 3 システムの階層構成

をセンサノード，（b）実環境の人，モノ，空間に作用するデバイスをアクチュエータノード，（c）提案する IFoT ミドルウェアを組み込んだ小型のコンピュータを**ニューロンモジュール**と定義する．このように，本システムは，複数台のセンサ/アクチュエータノードとデータの処理を行うニューロンモジュールが実環境に存在するものとする．ニューロンモジュールは，インターネットと接続するための長距離通信インタフェース（WiFi, LTE など）に加えて，多様なセンサ，アクチュエータと接続するための近距離通信インタフェース（Bluetooth Low Energy, Enocean, Zigbee など）を搭載していることを前提とする．また，センサ/アクチュエータは，ニューロンモジュールと接続するための近距離通信インタフェースを搭載している．各ニューロンモジュール間および，直接接続しているセンサ/アクチュエータとニューロンモジュール間は，相互にデータの送受信が可能であるものとする．

4.2 基本コンセプト

IFoT ミドルウェアの動作環境を階層化したものを図 3 に示す．このように，システムは，アプリケーション層，IFoT ニューロン層，センサ/アクチュエータ層の 3 つに階層化される．提案する IFoT ミドルウェアは，複数に分散されたニューロンモジュール上で動作し，アプリケーション層の要求に応じて，相互に連携，協調しながら，同時並行的に生成されるセンサデータストリームを流通，処理する役割を担う．IFoT ミドルウェアの機能を以下に記す．

（1）タスクの分散実行

空間に分散された IFoT ニューロンモジュール上で，複数のアプリケーションがリソースを共有しながら動作することを想定している．そのため，IFoT ミドルウェアでは，各アプリケーションの要求に従って，各ノードにタスクを割り当て，分散で実行する機能を提供する．

（2）オンデマンドな情報流通

複数のセンサノードから同時多発的にセンサデータが生成され，それらのデータは，ストリームとなって上位の各 IFoT ニューロンモジュールに送信される．IFoT ミドルウェア上では，そのセンサデータストリームを上位から割り当てられたタスクに応じて，各モジュール間で相互に流通する機能を提供する．

（3）オンラインな情報分析

複数のセンサノードから逐次生成されるセンサデータストリームの即時的な活用を実現するためには，リアルタイム性のある分析が求められる．そのため，IFoT ミドルウェアは，各モジュール間で相互に流通するストリームデータを蓄積することなく流れのままに逐次分析する機能を提供する．

（4）異種センサ/アクチュエータの統合

想定環境には，インタフェースや通信規格が異なる多種多様なセンサ/アクチュエータが存在する．そのため，IFoT ミドルウェアは，それらの差異を吸収し，データを中継する機能を提供する．

4.3 アーキテクチャ

本ミドルウェアの論理アーキテクチャを図 4 に示す．ここで，アプリケーションの仕様を示した設定ファイルを**レシピ**と定義する．具体的には，図 5 のように，湧き出るセンサデータストリームをどのように組み合わせて，どのタイミングで分析するかといった処理手順を示すタスクグラフが記述されたファイルを想定する．以下に，IFoT ミドルウェアの要素となる各機構の説明を述べる．

（1）機能割り当て機構

機能割り当て機構は，レシピ分解クラスとタスク割当クラスから構成される．レシピ分割クラスは，各アプリケーションのレシピを読み込み，一連の処理を分散実行可能なタスク単位に分割する．タスク割り当てクラスは，レシピ分解クラスが分割したタスクを各 IFoT モジュールに分配する．

（2）情報分析機構

情報分析機構は，学習クラス，判定クラス，管理クラスから構成される．学習クラスは，流通するセンサデータを逐次的に学習し，分析モデルの構築，更新を行う．判定クラスは，学習クラスが構築した分析モデルにしたがって，センサデータを分析し，状況の判定

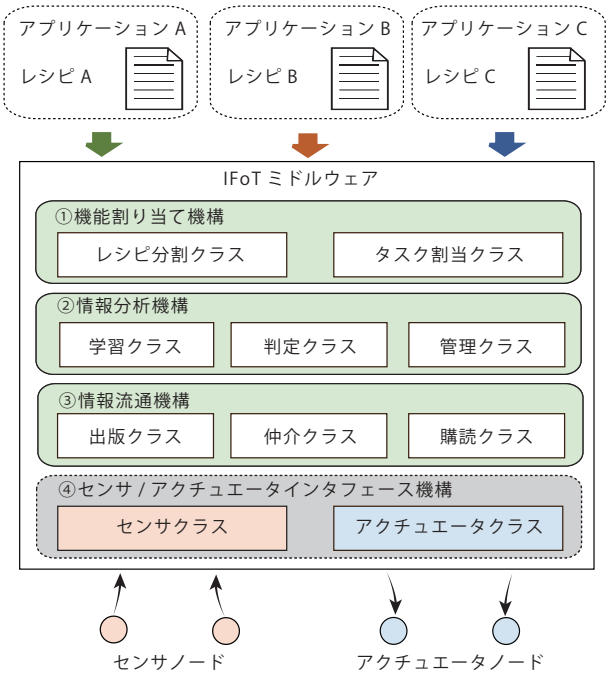

図 4　ミドルウェアの論理アーキテクチャ

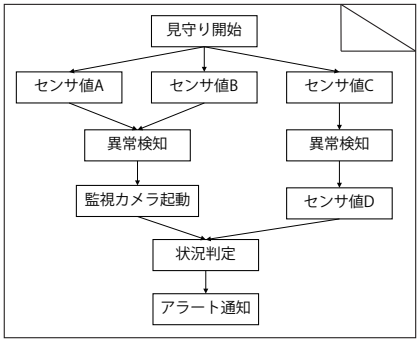

図 5　レシピのイメージ

を行う．管理クラスは，学習や判定といった処理を分散化する場合の協調動作など相互の連携を管理する．

(3) 情報流通機構

情報流通機構は，出版クラス，仲介クラス，購読クラスから構成される．本ミドルウェアでは，IFoT ノード間の情報流通に出版/購読型モデルを採用し，疎結合かつスケーラブルなメッセージングの実現を目指す．出版クラスは，センサデータの送信側となり，購読クラスは，センサデータの受信側となる．仲介クラスは，出版クラスと購読クラスの中間に位置し，購読クラスの指定トピックに基づきデータの流通を管理する．

(4) センサ/アクチュエータインタフェース機構

センサ/アクチュエータインタフェース機構は，センサクラスとアクチュエータクラスから構成される．それぞれのクラスは，対応するセンサ/アクチュエータノードのハードウェアや通信インタフェースを抽象化し，情報流通機構に対して共通のインタフェースを提供する．

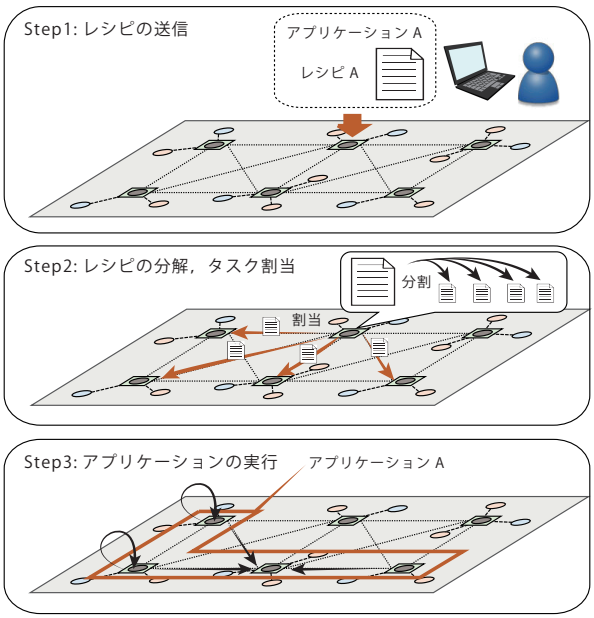

図 6　アプリケーションの構築手順

4.4　アプリケーションの構築手順

IFoT ミドルウェアを用いたアプリケーション構築手順のイメージを図 6 に示す．このように，Step1 : アプリケーション構築者は，アプリケーションのレシピを記述し，その設定ファイルを IFoT モジュールに送信する．Step2 : レシピを受信した IFoT モジュールは，ファイルを読み込みレシピを分散可能なタスク単位に分割する．そして，センサの接続状況やリソース状況に応じて，各タスクを他の IFoT モジュールに割り当てる．Step3 : タスクを割り当てられた各 IFoT モジュールは，その内容に従ってクラスを起動し，相互に協調，連携しながらアプリケーションとしての動作を振る舞う．

4.5　アプリケーションの実行例

ここでは，3 台のセンサからのストリームデータをオンライン分析し，異常を検出した場合にアクチュエータに通知するという簡易的な応用シナリオを例に，複数台のニューロンモジュール上で分散実行されるクラス間の相互連携の流れついて述べる．クラス間の連携イメージを図 7 に示す．

(1) 流通フェーズ

モジュール A，B，C 内において，センサクラスは，センサノードから生成されるセンシングデータ（data）を出版クラスに中継する．次に，出版クラスは，仲介クラスが起動されているモジュール D に対して，トッピックというチャンネルを指定して，受信したセンシングデータを出版（Pub(Topic, data)）する．これらのクラスは，センサノードからデータが生成される度に，以上の動作を繰り返す．

(2) 分析フェーズ

モジュール E，F 内において，購読クラスは，モ

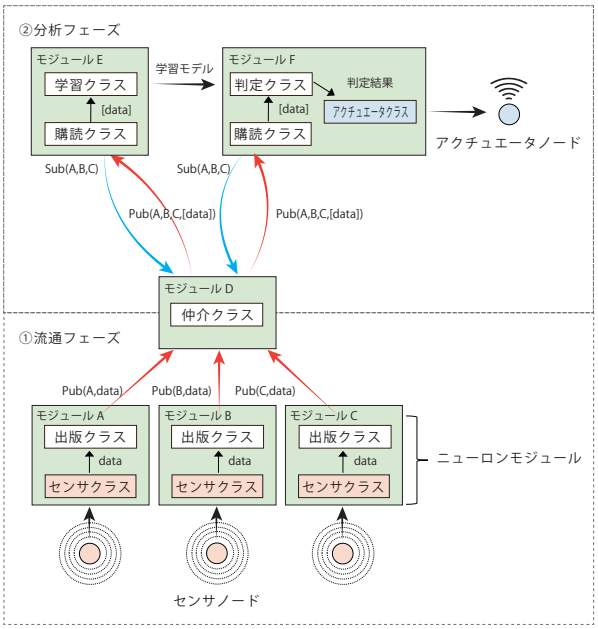

図 7　クラス間の連携イメージ

ジュールDの仲介クラスに対して，購読（Sub(Topic)）メッセージを送り，購読したいトピックを登録する．購読メッセージを受理した仲介クラスは，登録されたトピックに応じて，次々と出版されるセンシングデータを転送する．次に，モジュール E の学習クラスは，購読クラスが受信したデータを受けとって，逐次追加学習を行い学習モデルを更新する．モジュール F 内では，購読クラスがセンシングデータを受信する．判定クラスは，学習クラスが更新した学習モデルに基づき，センシングデータが異常値かどうかを判定する．そして，異常値を検出した場合は，アクチュエータクラスを介して，アクチュエータノードに通知する．

4.6　プロトタイプの実装

本研究では，IFoT ミドルウェアにおける情報流通機構と情報分析機構の 2 つに機能を限定し, IFoT ミドルウェアのプロトタイプを開発した．プロトタイプの実装は，Python 言語を用いて行った．本プロトタイプでは，情報流通機構の実現にあたり，IFoT ニューロン間のデータ送受信に，ブローカー・ベースの軽量な出版/購読型メッセージ・プロトコルである MQTT（Message Queuing Telemetry Transport）を採用した．そして，MQTT サーバに，MQTT Mosquitto[36]，MQTT クライアントに，Paho[37] を使用し，各クラスを実装した．また，情報分析機構では，逐次的な学習および分析の実現に向けて，並列分散処理に対応可能なオンライン機械学習フレームワークである Jubatus を活用した．

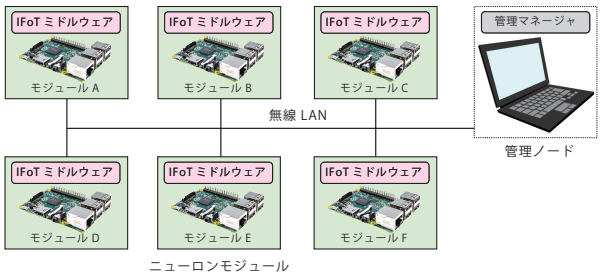

図 8　評価システム構成

表 1　評価システムの機器仕様

ニューロン モジュール	使用機器	Raspberry Pi 2
	OS	Raspbian
	CPU	ARM Cortex-A7 900MHz
	Memory	1GB
管理 ノード	使用機器	Think pad x250
	OS	Ubuntu 14.04
	CPU	Core i5-5200U 2.2GHz
	Memory	8GB

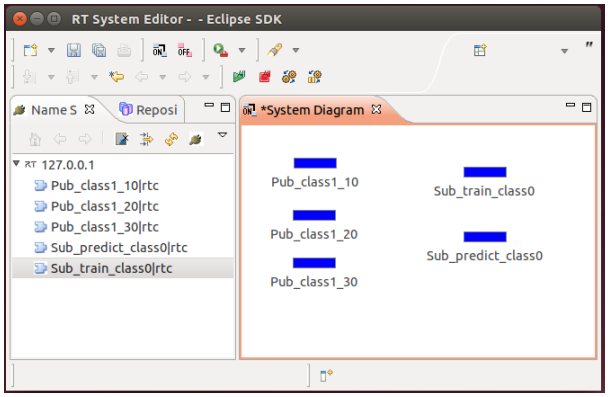

図 9　管理マネージャ画面

5.　動作検証実験

5.1　実験の概要

本研究では，IFoT ミドルウェアの実現可能性を検証することを目的として，4.6 節で述べた IFoT ミドルウェアのプロトタイプを 6 台の Raspberry Pi 上に実装し，動作検証実験を実施した．

実験では，4.5 節で述べた応用シナリオ基づいて，ニューロンモジュール上の IFoT ミドルウェアを駆動し，データの流通，分析という一連の処理にかかる動作時間を測定する．そして，データの生成レートを変化させた場合の処理遅延（センサデータの生成から学習，分類処理が完了するまでの遅延時間）の変化を確認する．

実験に使用した評価システム構成を図 8 に示し，機器の仕様を表 1 に示す．本システムは，IFoT ミドルウェアのプロトタイプが導入された 6 台のニューロンモジュール（Raspberry Pi）と実験用に設置された 1 台の管理ノード

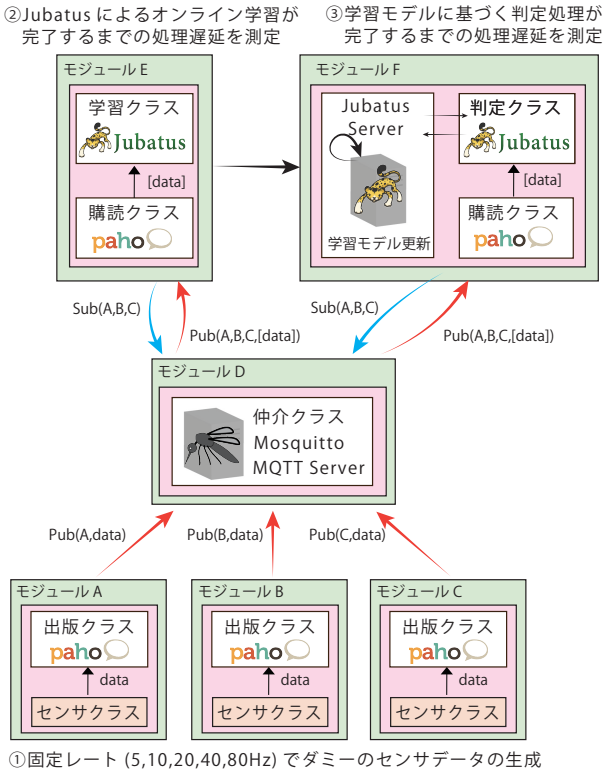

図 10 評価システム構成

で構成されている．全てのモジュールは，共通の無線 LAN に接続している．システム評価者は，管理ノード上の管理マネージャ（図 9）から各モジュール上で起動するクラスを選択し，モジュール間の連携動作を設定することが出来る．なお，管理マネージャの実装には，OpenRTM-aist[38] を用いた．

5.2 実験方法と結果

本実験では，図 10 に示すように，3 台のニューロンモジュール上で生成したダミーのセンサデータ（32byte）を Mosquitto Broker を介して逐次流通しながら，Jubatus によるオンラインの（1）学習処理，（2）判定処理が完了するまでにかかる処理時間を測定した．そして，センサデータの生成レート（5, 10, 20, 40, 80Hz）を変化させた場合の処理遅延の変化を確認した．なお，メッセージの流通を管理する MQTT サーバはモジュール D で起動し，センサ値の学習および分類処理を行う Jubutus サーバは，モジュール F で起動する．このように，本実験では，各モジュールが担う処理を並列分散化せずに，1 つのタスクが 1 つのモジュール内で完結する構成での処理遅延を測定する．また，MQTT の QoS（Quality of Service）モードは，3 段階中（0, 1, 2）サーバへの負荷が一番少ない 0（1 度限りのメッセージ送信，再送処理なし）を選択する．

表 2，3 にセンサデータ生成レートに応じた学習処理および判定処理における遅延時間の測定結果を示す．結果より，学習，判定，共に，5, 10Hz のような生成レートが低い

表 2 評価結果 (センサデータ生成-学習)

Sampling rate(Hz)	Time(ms)	
	Ave.	Max
5	58.969	357.619
10	60.904	360.761
20	232.944	419.513
40	1123.317	1482.500
80	1636.907	1913.752

表 3 評価結果 (センサデータ生成-判定)

Sampling rate(Hz)	Time(ms)	
	Ave.	Max
5	58.969	346.142
10	59.020	334.501
20	74.747	373.992
40	744.535	819.748
80	1144.580	1249.122

センシングに関しては，遅延の少ないリアルタイムな処理を実現できることが分かった．また，20，40Hz で，遅延時間が増え始めることが確認できる．そして，80Hz を超える高い生成レートの場合には，遅延時間が更に増加し，リアルタイム性が損なわれるということが判明した．また，学習処理より判定処理の遅延時間が少ない点に関しては，判定クラスが動作しているモジュール F 上で，Jubatus サーバが起動しているためだと考えられる．今回の実験は，センサデータを生成するノード数が 3 台かつ，ダミーセンサデータのパケットサイズが 32byte という小規模な環境で実施した．今後，ストリームデータを生成するノード数が多い環境や，画像や動画のように一つのパケットサイズの大きな環境で，実時間処理を実現するためには，各処理タスクの並列分散化によりリアルタイム性とスケーラビリティを追及する必要がある．

6. おわりに

本研究では，IoT デバイスの計算能力を活用し，その場で素早く処理する地産地処をコンセプトに，IoT デバイスの上位で動作し，IoT データストリームの実時間処理を実現する IFoT ミドルウェアを提案した．IFoT ミドルウェアでは，(a) IoT デバイス群の上で複数のアプリケーションが計算リソースの共有しながら動作する環境，(b) 各アプリケーション内で生成された価値ある情報（コンテンツ）を 2 次利用，3 次利用（高次利用）できる環境の実現に向けて，(1) アプリケーションから要求されるタスクを分割しながら分散で実行する機能 (2) IoT データストリームを分散処理のためデバイス間でオンデマンドに流通する機能，(3) 複数のデータストリームをオンラインで分析する機能 (4) 異種センサ/アクチュエータのシームレスな連携を実現する機能を提供する．

本稿では，IFoT ミドルウェアの詳細設計を示し，一部

に機能を限定した IFoT ミドルウェアのプロトタイプを試作した．また，IFoT ミドルウェアの実現可能性を検証することを目的として，IFoT ミドルウェアのプロトタイプを 6 台の RaspberryPi 上に実装し，動作検証実験を実施した．実験の結果，小規模な動作環境では，リアルタイムな流通，分析処理を実現可能であることが分かった．

今後は，より大規模な環境に適応するために，クラウドとの一部連携も考慮しながら，IFoT 上で IoT データストリームの流通，分析処理を並列分散実行するための仕組みを検討し，IFoT ミドルウェアのリアルタイム性とスケーラビリティを追及する．また，レシピを記述するための言語やインタフェースを策定すると共に，ニューロンモジュールの動的な参加離脱に対応した，ノード検索機構を開発し，複数のアプリケーションが IFoT ミドルウェア上で動作する環境を実現する．そして，複数のデバイスが遍在する IoT 環境での実証実験を行い，IFoT ミドルウェアの有効性を評価する．

謝辞 本研究の一部は，奈良先端科学技術大学院大学 CICP（Creative and International Competitiveness Project ：想像力と国際協力を育む情報科学教育コア「プロジェクト型研究」）の助成によって行われたものである．ここに記して謝意を表します．

参考文献

[1] InfoFlow(情報流) プロジェクト, http://www.infoflow.org.
[2] 安本慶一，山口弘純 ：多数のデータストリームを実時間で融合・編纂し利活用するための次世代「情報流」技術，情報処理学会誌，Vol. 10, No. 2, pp. 287-302 (2014).
[3] K. Yasumoto, H. Yamaguchi, H. Shigeno ： Survey of Real-time Processing Technologies of IoT Data Streams, to appear in Journal of Information Processing (2016).
[4] Mineraud J, Mazhelis O, Su X, Tarkoma S ： Contemporary Internet of Things platforms. arXiv Preprint arXiv:150107438 (2015)
[5] Arkessa, http://www.arkessa.com/
[6] Axeda, http://www.axeda.com/
[7] ThingSquare, http://www.thingsquare.com/
[8] Thingworx, http://www.thingworx.com/
[9] M. Blackstock and R. Lea ： IoT mashups with the WoTKit, in Proceedings of IEEE Internet of Things (IOT), pp. 159-166 (2012).
[10] Xively, http://xively.com
[11] J. Kim and J.-W. Lee ：OpenIoT: An open service framework for the Internet of Things,” in IEEE World Forum on Internet of Things (WFIoT), pp. 89-93 (2014)
[12] P. Kostelnik, M. Sarnovsk, and K. Furdk ： The semantic middleware for networked embedded systems applied in the internet of things and services domain, Scalable Computing: Practice and Experience, vol. 12, pp. 307-315 (2011)
[13] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin ： IoT gateway: bridging wireless sensor networks into internet of things, in Proceedings of IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC), pp.347-352 (2010).
[14] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta ： The internet of things has a gateway problem, ACM HotMobile'15 , pp. 27-32 (2015)
[15] Sunyanan Choochotkaew, Hirozumi Yamaguchi ,Teruo Higashino ,Megumi Shibuya ： Data Prioritization at Multi-user IoT Gateway with Multiple Sensor Data Attributes, マルチマディア通信と分散処理ワークショップ論文集, Vol. 5, pp.53-61 (2015)
[16] A. Davis, J. Parikh, W. E. Weihl ： Edgecomputing: extending enterprise applications to the edge of the internet, Proc. of the 13th international World Wide Web conference on Alternate track papers and posters, (2004).
[17] Bonomi, Flavio, et al.: ： Fog Computing and Its Role in the Internet of Things, Proc. of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12), pp. 13-16 (2012).
[18] S. Dutta, A. Narang, and S. K. Bera :Streaming Quotient Filter: A Near Optimal Approximate Duplicate Detection Approach for Data Streams, Proc. of the VLDB Endowment, Vol. 6, No. 8, pp. 589-600 (2013).
[19] Q. Zhang, J. Liu, and W. Wang ： Approximate Clustering on Distributed Data Streams, Proc. of the International Conference on Data Engineering (ICDE2008), pp. 1131-1139 (2008).
[20] C. Lei, E. A. Rundensteiner, and J. D. Guttman ： Robust Distributed Stream Processing, Proc. of the InternationalConference on Data Engineering (ICDE2013), pp. 817-828 (2013).
[21] 義久 智樹，原 隆浩 : IoT 環境におけるストリーミング処理時間短縮手法, マルチマディア通信と分散処理ワークショップ論文集, Vol. 5, pp.62-70 (2015)
[22] Apache Mahout，http://mahout.apache.org/
[23] Weka，http://www.cs.waikato.ac.nz/ml/weka/index.html
[24] Apache SAMOA，https://samoa.incubator.apache.org/
[25] Spark MLib，http://spark.apache.org/mllib/
[26] Jubatus，http://jubat.us/ja/
[27] Cloudera Oryx, https://github.com/cloudera/oryx
[28] MQTT，http://mqtt.org/
[29] CoAP，http://coap.technology/
[30] WebSocket，https://www.websocket.org/
[31] Z. Shelby and C. Borman ：6LoWPAN: The Wireless Embedded Internet, (2009)
[32] Shogo Maenaka, Shigeya Morishita, Daichi Nagata, Morihiko Tamai, Keiichi Yasumoto, Toshinobu Fukukura, and Keita Sato. ：SakuraSensor: a system for realtime cherry-lined roads detection by in-vehicle smartphones, Proc. of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2015), pp. 695-705 (2015)
[33] 西村友洋, 樋口雄大, 山口弘純, 東野輝夫 ：スマートフォンを活用した屋内環境における混雑センシング, 情報処理学会論文誌, Vol. 55, No. 12, pp. 2511-2523 (2014).
[34] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K.Pister ：System architecture directions for networked sensors, Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 93-104, 2000.
[35] J. Hill and D. Culler, ：MICA: A wireless platform for deeply embedded networks, IEEE Micro, vol. 22, pp. 12-24, Dec. 2002.
[36] Mosquitto, http://mosquitto.org/
[37] Paho, http://www.eclipse.org/paho/
[38] N. Ando, T. Suehiro, K. Kitagaki, and T. Kotoku ： A Software Platform for Component Based RT-System Development: OpenRTM-Aist, Proc. of SIMPAR 2008, pp.87-98, Nov. 2008.