Vol.2017-MBL-82 No.40
Vol.2017-UBI-53 No.40
2017/3/10

IoTデータ流を実時間で分散処理するための IoTデバイス向け共通ミドルウェアの設計と評価

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

概要：我々の研究グループでは，「地域で生成された IoT データ流を地域で処理して利活用する：地産地処」をコンセプトに，IoT データ流をデータの発生源に近い IoT デバイス群を活用して分散処理する新たなフレームワークとして IFoT(Information Flow of Things) を提案している．本研究では，IFoT の実現に向けた具体的なアプローチとして，IoT デバイスの上位で動作し，多種多様な IoT データ流の処理タスクをクラウドレスで分散実行する「IFoT ミドルウェア」を設計し，そのプロトタイプを開発する．IFoT ミドルウェアでは，異種の IoT データ流を統一的に取り扱うためにメタデータのフォーマットを策定すると共に，メタデータを付加して統一的な IoT データ流を生成するためのゲートウェイ機構を実現する．また，高次 IoT データ流を生成するための処理手順と，地理的エリアや時空間解像度，許容遅延が記述された設定ファイルの内容に従って，処理タスクを実時間で分散実行するための機構を実現する．IFoT ミドルウェアの動作検証として，小型の汎用デバイスである Raspberry Pi を用いて，各機構に関する性能評価実験を行った．実験の結果，統一的な IoT データ流生成機構に関しては，1 フレームが 10〜100Kbyte のデータ流に対して，素早くメタデータを付加し，実時間で流通可能であることがわかった．また，高次 IoT データ流生成タスクの分散実行機構に関しては，1 台のノードによるタスクの逐次処理と比べて，10 並列分散処理時に約 92.46%という高い並列化効率が得られた．また，タスクを分担する際に，低負荷なノードを選出することによって，高い並列分散処理パフォーマンスを得られることを確認した．

1. はじめに

Internet of Things (IoT) 技術の発展により，地球上で生成されるデータ量が指数関数的に増加している [1]．現状，IoT デバイスから周期的に逐次生成されるデータ (IoT データ流) は，ネットワークを介して遠隔に配置された大規模なデータセンタ（クラウド）に集約され，時間をかけて処理されている．数百億の IoT デバイスが実社会に導入される 2020 年に向けては，このようなクラウド一極集中型のアーキテチャでは，通信帯域の逼迫やクラウド側の計算・蓄積資源の枯渇といった問題が深刻化することが予測されている [2]．近年は，これらの問題意識から，通信インフラやクラウドで生じる負荷を軽減する概念として，小規模なデータセンタ（エッジサーバ）を近距離に配置するエッジコンピューティング [3] やフォグコンピューティング [4] の研究開発が活発化している．しかしながら，これらの概念は，クラウドコンピューティングの延長となる技術であるため，エッジサーバを利用する IoT デバイス数が想定を超えた場合に，通信帯域の逼迫やエッジ側の計算資源不足といった問題が発生する可能性がある．そのため，これらの問題を根本から解決する新たなアーキテクチャが求められている．

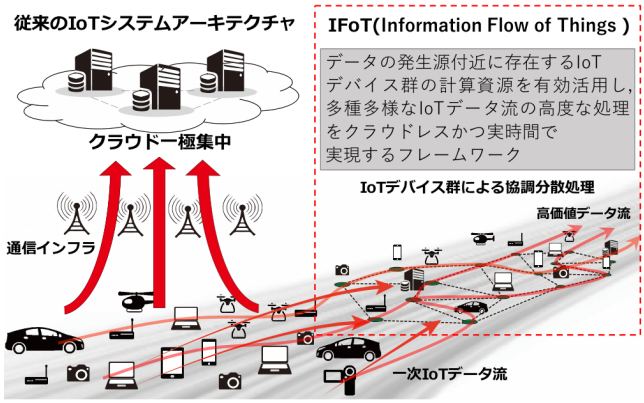


図 1　IoT システムのパラダイムシフト

これらの背景を踏まえて，我々の研究グループでは，通信インフラやクラウドで生じる負荷を分散し，IoT データ流の即時的な利活用を促進する次世代型の情報処理基盤として IFoT（Information Flow of Things） [5] (図 1 参照) を提案している．IFoT は，「地域で生成された IoT データ流を地域で処理して利活用する：地産地処」をコンセプトとして，IoT データ流をデータの発生源に存在する IoT デ

[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 School of Information Science and Technology, Osaka University
[a)] nakamura.yugo.ns0@is.naist.jp
[b)] yasumoto@is.naist.jp

バイス群を活用して分散処理し，ユーザやアプリケーションが求める高価値な情報を素早く実時間で提供することを目指している．IFoT の最大の特徴は，従来のクラウド依存なアーキテクチャが抱える問題に対する現実的な解決策として，今後，数千億〜数兆と実社会に展開されるであろう IoT デバイス群自体の計算資源を有効活用する点である．

本研究では，IFoT の実現に向けた具体的なアプローチとして，以下に示す 2 つの課題に焦点を当て，IoT デバイスの上位で動作し，多種多様な IoT データ流の処理タスクをクラウドレスで分散実行するための共通ミドルウェアとして「IFoT ミドルウェア」の設計/開発を行う．

課題 1：統一的な IoT データ流の生成

あらゆるモノがセンサ化される IoT 環境では，フォーマットや通信規格の異なる IoT データ流が生成されることが想定されている．そのため，これらの雑多な IoT データ流を統一的に扱うための仕組みが求められている．

課題 2：IoT データ流の実時間処理

多くの IoT デバイスが実世界の状況を反映した IoT データ流を生成しているため，情報の鮮度が落ちる前に，様々なサービスとして活用できることが望ましい．そのため，アプリケーションの要求に応じて，IoT データ流を素早く処理する仕組みが求められる．

IFoT ミドルウェアでは，課題 1 の解決に向けて，多様な IoT デバイスから生成される異種のデータ流を共通の枠組みで処理できるようにするためのメタデータのフォーマットを策定する．また，策定されたメタデータを異種のデータ流に付加すると共に，共通のネットワーク上で流通させるためのゲートウェイ機構を実現する．課題 2 の解決に向けては，アプリケーションが要求する高価値な情報（高次 IoT データ流）を生成するための処理手順と，地理的エリアや時空間解像度，許容遅延などの品質を要求クエリとして記述するための言語を策定すると共に，要求クエリの内容に従って，IoT デバイス群上で処理タスクを分散実行するための機構を実現する．

IFoT ミドルウェアの有効性を検証するために，小型の汎用デバイスである Raspberry Pi を用いて，各機構（A : 統一的な IoT データ流生成機構, B : 高次 IoT データ流生成タスクの分散実行機構）に関する性能評価実験を行った．実験の結果，統一的な IoT データ流生成機構に関しては，1 フレームが 10〜100Kbyte のデータ流に対して，素早くメタデータを付加し，実時間で流通可能であることがわかった．また，高次 IoT データ流生成タスクの分散実行機構に関しては，1 台のノードによるタスクの逐次処理と比べて，10 台のノードを用いた並列分散処理時に，約 92.46 %という高い並列化効率が得られた．また，タスクを分担する際に，低負荷なノードを選出することによって，高い並列分散処理パフォーマンスを得られることを確認した．

2. IFoT のユースケースと要件

本章では，多種多様な IoT デバイスから逐次生成さる雑多な IoT データ流を活用した IoT アプリケーションの具体例として，3 種類のユースケースシナリオを示し，そのシナリオを実現するための要件を示す．

2.1 高齢者向け生活見守りアプリケーション

音センサ，モーションセンサ，カメラなどのセンシングデバイスを搭載した複数の IoT デバイスが設置されている居住空間を想定する．この居住環境では，IoT デバイスから生成される複数の IoT データ流を組み合わせて処理することにより，高齢者が現在何をしているのかといった生活行動を示す高次の IoT データ流が生成されている．家族や介護士などは，この高次 IoT データ流を日頃からモニタリングすることによって，高齢者の生活習慣の異変などを早期発見することが可能になる．また，高齢者が転倒して動けなくなるなどの真に危険な状態が発生した場合は，自動的に異常が検知され，家族や介護士，病院等に対して素早く状況が通知される見守りサービスが提供されている．

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 データ流を常に解析し，高次 IoT データ流，すなわち行動認識の結果を生成すると共に，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 IoT データ流を複数のデバイス間で相互に流通しながら，実時間で異常がないかを判断する機能が求められる．

2.2 快適かつ省エネな空調制御アプリケーション

温湿度センサ，照度センサ，CO2 センサやサーモカメラなどのセンシングデバイスを搭載した複数の IoT デバイスが設置されているオフィス空間を想定する．このオフィス空間では，その空間の温湿度や照度といった環境情報に加えて，オフィスワーカそれぞれの体温などが示された IoT データ流が生成されている．空調制御アプリケーションは，これらの IoT データ流を活用することによって，生産性向上に向け快適に空調環境を制御することができる．また，空間の人の密度や，会議室の空き状態を表す IoT データ流も生成されており，より省エネに照明装置や空調装置を制御するアプリケーションも動作している．

このアプリケーションを実現するには，オフィス空間の環境情報やオフィスワーカの体温などに相当する異種の IoT データ流を組み合わせて処理し，オフィス空間の状況とオフィスワーカの状況を日頃から監視できる必要がある．この場合，空間の人の密度に応じて処理が必要なデータ量が変動することから，計算資源の容量に空きがある IoT デバイスの処理能力を活用して，人の密度が高い場所の IoT データ流を処理できることが望ましい．

2.3 リアルタイム観光支援アプリケーション

様々な観光スポットを網羅的にキャプチャすることが可能なカメラや，気候を観測するためのセンシングデバイスを搭載した IoT デバイスが多数設置されている都市エリアを想定する．広域をカバーする多数のカメラから生成される一次 IoT データ流を解析処理することにより，街の混雑度が高いスポットや桜や紅葉など景観が良いスポットを示した高次の IoT データ流が生成されている．観光客は地域ネットワークに接続している自身のモバイル端末から，これらの IoT データ流を確認することができ，街のリアルタイムな状況を確認することができる．また，混雑が嫌いだけど景観の良いスポットを訪れたいといった観光客には，混雑度と景観情報など複数の高次 IoT データ流を組み合わせて，いま空いている景観スポットを推薦するなど観光客の嗜好に合わせた観光支援アプリケーションを提供することも可能である．

このアプリケーションを実現するには，桜センサ [24] や混雑センシング [25] など個々に独立しているアプリケーションが生成した価値のある IoT データ流を 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. IFoT ミドルウェア

本章では，1 章で示した課題および 2 章で示した要件に対応するための IFoT ミドルウェアについて述べる．

3.1 IFoT ミドルウェアのコンセプト

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をセンサノード，(b) 提案する IFoT ミドルウェアを組み込んだ IoT デバイスを IFoT スレーブノード，(c) IFoT ミドルウェアを搭載し，複数の IFoT スレーブノードを管理する役割を担うデバイスを IFoT マスタノードと定義する．実環境には，対象空間を網羅的にセンシングするために複数台のセンサノードと IFoT スレーブノードが設置されている．さらに，これらのノードを統括するデバイスとして IFoT マスタノードが一台設置されている．各センサノードと IFoT スレーブノードは，有線か無線の通信インタフェースで直接接続されているものとする．IFoT ミドルウェアでは，IFoT スレーブノードと IFoT マスタノードは，センサノードとしての機能を持たせることが可能である．そのため，IFoT スレーブノード，IFoT マスタノードのそれぞれが一台づつ存在する環境を最も基本的なシステム構成とする．また，計算資源が豊富なセンサノードを IFoT スレーブノードや IFoT マスターノードとして選出することも可能である．

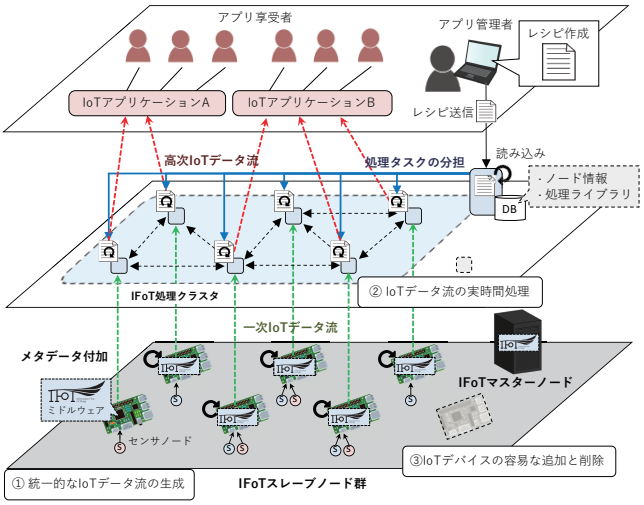


図 2 IFoT ミドルウェアの動作環境

IFoT ミドルウェアでは，生のセンサデータに対してメタデータが付加された統一的な IoT データ流を一次 IoT データ流と呼ぶ．また，IoT アプリケーションが求める高価値な IoT データ流を高次 IoT データ流と呼ぶ．IFoT ミドルウェアでは，要求された高次 IoT データ流を素早く流通するために，IFoT 処理クラスタを活用した実時間処理の実現を目指している．IFoT 処理クラスタとは，ローカルに存在する IFoT スレーブノードの計算資源をひとまとまりのグループと捉えた処理クラスタである．アプリ管理者は，高次 IoT データ流の仕様を規定した設定ファイルを記述し，IFoT マスタノードに送信することで IFoT 処理クラスタに対して，高次 IoT データ流の生成処理を依頼することができる．高次 IoT データ流の仕様を規定した設定ファイルをレシピと呼ぶ．レシピには，高次 IoT データ流をどのように処理して生成するかを示したタスクグラフに加えて，地理的エリアや時空間解像度，許容遅延といった要求品質が記述されている．レシピを受け取った IFoT マスタノードは，要求品質から，タスクグラフを分割し，分散処理に向けた処理タスクを生成する．そして，自身のデータベースに存在する IFoT スレーブノード情報を照合し，どのノードにどの処理タスクを分担するかを決定すると共に，処理タスクを各ノードに送信する．処理タスクを分担する際には，その処理に必要となるプログラムである処理ライブラリも一緒に送信される．処理タスクを受け取った IFoT スレーブノードは，その内容に従って IoT データ流の処理を行い，高次 IoT データ流を生成/流通する．

3.2 IFoT ミドルウェアの設計

図 3 に IFoT ミドルウェアの構成を示す．IFoT ミドルウェアは，IFoT マスタノードと IFoT スレーブノードという二つから構成されている．IFoT マスタノードは，IFoT スレーブノードとなる IoT デバイスのノード情報を管理するとともに，アプリ管理者から送られてくるレシピを読み

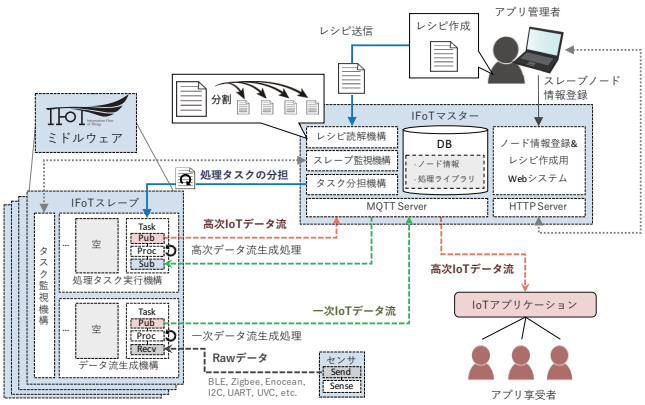


図 3 IFoT ミドルウェアの設計

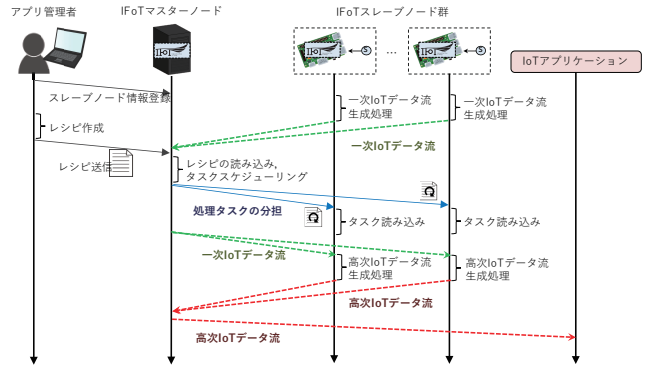


図 4 IFoT ミドルウェアの処理手順

込み，処理タスクを IFoT スレーブノードに分担する役割を担っている．IFoT スレーブノードは，接続しているセンサノードからのセンサデータを統一的な IoT データ流に変換して流通すると共に，IFoT マスタノードから振られた処理タスクを実行し，高次 IoT データ流を生成する役割を担っている．なお，IFoT ミドルウェアでは，IoT データ流の通信プロトコルとして，近年，IoT 向け軽量プロトコルとして注目を集めている MQTT を採用している．MQTT は Pub/Sub（Publish/Subscribe）型のデータ配信モデルで，メッセージ発行元とメッセージ購読者を MQTT サーバがトピックごとにつなぐ仕組みである．IFoT ミドルウェアでは，IFoT マスタノードが MQTT サーバの役割を担う．IFoT スレーブノード同士は，IFoT マスタノードを介して IoT データ流のやり取りを行う．

IFoT ミドルウェアの処理手順を図 4 に示す．まず，アプリ管理者は，対象空間に設置する IoT デバイスの情報を IFoT マスタノードに登録する．登録のためのインタフェースは，IFoT マスタノード上の Web システムで提供されている．設置された IFoT スレーブノードは，データ流が発生する位置情報やデータの種類を記すメタデータを付加し，一定周期で一次 IoT データ流をパブリッシュする．メタデータの記述ルールは，次節で詳述する．次に，アプリ管理者は IoT アプリケーションで必要となる高次 IoT データ流の生成レシピを記述し，IFoT マスタノードに送信す

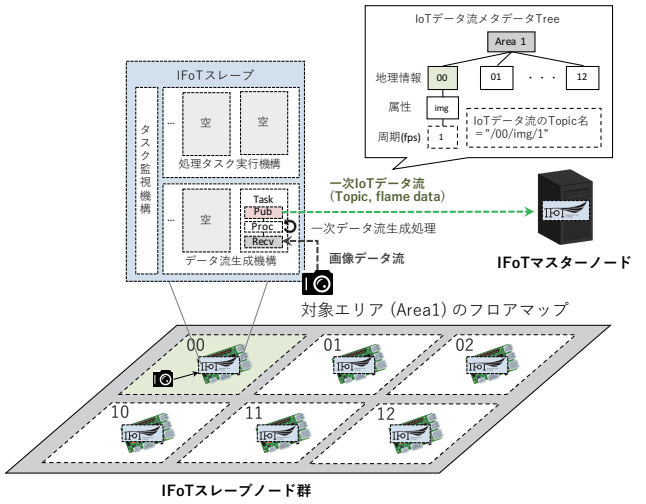


図 5 一次 IoT データ流の生成イメージ

る．IFoT マスタノードは，レシピを読み込み，どのタスクをどのノードに割り当てるのかをスケジューリングする．そして，処理タスクが IFoT スレーブノードに分担される．IFoT スレーブノードは，受信した処理タスクを読み込み，指定された IoT データ流の加工処理を行い，高次 IoT データ流を生成する．IoT アプリケーションは，この高次 IoT データ流をサブスクライブすることによって，高次 IoT データ流を定期的に受信することができる．

3.2.1 統一的 IoT データ流のメタデータ記述ルール

雑多で非構造な IoT データ流に対して，共通の枠組みで処理できるようにするためのメタデータ記述ルールを定義する．IoT データ流の生成間隔ごとに，データが発生した位置情報やデータの種類を付加することにより，IoT データ流の検索や加工処理の容易化を目指す．IFoT ミドルウェアでは，データの振り分けを効率化するために，IoT データ流の通信プロトコルとして MQTT [23] を採用している．そこで，IoT データ流のメタデータを MQTT パケットの Topic 部分に記述することとする．まず，IFoT ミドルウェアでは，図 5 に示すように，IFoT スレーブノードの登録時に IoT データ流メタデータ Tree を構築する．IFoT スレーブノードは，IoT データ流メタデータ Tree に基づいて，IFoT マスタノードが存在するエリアからの相対パスで，地理情報，データの属性，生成周期を Topic とする一次 IoT データ流を生成する．

3.2.2 高次 IoT データ流生成レシピのフォーマット

アプリケーションが要求する高次の IoT データ流を生成するためには，様々なデータ流の収集，加工，統合といった処理を伴う．IFoT ミドルウェアおける**「レシピ」**とは，JSON フォーマットをベースとした設定ファイルであり，高次データ流を生成するための処理手順や，地理的エリアや時空間解像度，許容遅延などの要求品質を記述することが可能である．レシピの基本構造を図 6 に示す．レシピ内で用いる要素の一覧を表 1 に示す．レシピは，if_spec

```
{
  "if_spec":{
    "if_name":"IFNAME",
    "drive_time": DRIVE_TIME,
    "target_area":"AREA",
     "sampling_period":SPS,
    "mesh_width":MESH
  },
  "if_dag":{
    "if_task":{
      "library_id":"LIBRARY_ID",
      "ex_delay": DELAY_TIME,
      "in_topic":"IN_TOPIC",
      "out_topic": OUT_TOPIC"
    },
    ...
  }
}
```

図 6　レシピの基本構造

と if_dag という要素で構成されている．if_spec には，IoT データ流の要求仕様を記述し，if_dag には，高次 IoT データ流を生成するためのタスクグラフを記述する．if_spec は，高次データ流の名前（if_name），タスクを投げてから終了するまでの制限時間（drive_time），対象とするエリア（target_area），データ流の生成周期（dt），センサの設置間隔であるメッシュ幅（dw）から構成される．これにより，30 分間，キッチンの温湿度データ流を 1fps で，メッシュ幅 1m 毎に収集するというような仕様を記述可能である．

if_dag は，複数の if_task で構成され，if_task は，タスクグラフにおける 1 タスクに相当する．if_task は，その処理タスクを実行するのに必要なライブラリの ID（library_id），その処理の許容遅延（ex_delay），入力データ流の Topic 名（in_topic），出力データ流の Topic 名（out_topic）で構成される．例えば，画像データ流から映った人の顔を抽出する処理タスクの場合，「library_id = face_ditection, ex_delay = 10, in_topic = /img, out _topic = /face_img」のように記述する．その顔画像が誰なのかを認識したい場合には，in_topic を/face _img とした顔認識処理タスクを追加することになる．現状，処理タスクに必要となるライブラリは，事前に IFoT マスタノードのデータベースに登録されているものしか取り扱うことができない設計となっている．今後は，クラウドと連携し，他のアプリ管理者が作成した処理ライブラリを共有できるような仕組みを導入する予定である．また，IFoT マスタノード上ではレシピ作成用 Web システムが起動しており，アプリ管理者は，Web のフォームを介してレシピを記述することが可能な設計である．

3.2.3　高次 IoT データ流生成処理の分散実行機構

IFoT ミドルウェアでは，レシピに記されている要求品質とタスクグラフの内容に応じて，高次 IoT データ流を生成するための処理をローカル環境の IoT デバイス群を用いて分散実行するための機構を実現する．シンプルな例として，前節で示した，顔検出処理を題材に，IFoT ミドルウェアによる並列分散処理のイメージを図 7 に示す．

まず，アプリ管理者は，自身のアプリケーションで必要

表 1　レシピの要素

JSON Key	型	説明
if_spec	配列	IoT データ流の要求品質
if_name	文字列	IoT データ流の名前
drive_time	数値	タスクの生存時間 (min)
target_area	文字列	ターゲットエリア
dt	数値	時間解像度：データ流の生成周期 (ms)
dw	数値	空間解像度：メッシュ幅 (m)
if_dag	配列	IoT データ流のタスクグラフ
if_task	配列	処理タスク
library_id	文字列	処理ライブラリ ID
ex_delay	数値	許容遅延 (s)
in_topic	文字列	入力の Topic 名
out_topic	文字列	出力の Topic 名

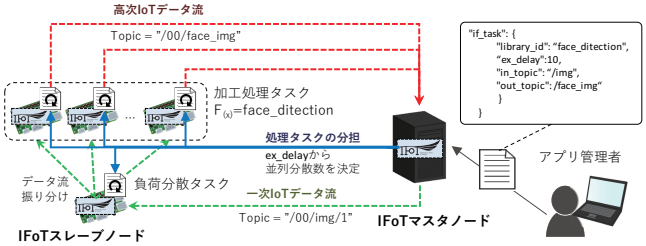


図 7　レシピに基づく並列分散処理の例

となる顔画像データ流を生成するためのレシピを記述する．作成したレシピは，IFoT マスタノードに送信され，レシピ読解機構で読み込まれる．スレーブ監視機構は，処理クラスタ内の各 IFoT スレーブノードに割り当てられたタスクを把握し，各 IFoT スレーブノードの計算資源空き状況を監視する役割を担っている．これにより，高負荷な状態の IFoT スレーブノードを避けて，低負荷な IFoT スレーブノードに優先的にタスクが分担するようなタスクスケジューリングが実施される．タスク分担機構では，各タスクの許容遅延（ex_delay）の値に基づいて，そのデバイスの並列分散数を決定し，処理タスクが分割される．なお，データベースに蓄積されている処理ライブラリには，その処理の負荷レベルがあらかじめ明らかになっているものとする．タスクは，基本的に，データ流を各分散ノードに分配する負荷分散タスクと IoT データ流を処理ライブラリに基づいて処理する加工処理タスクに分割される．ここで，負荷分散タスクとは，逐次生成される IoT データ流を受信し，加工処理タスクが割り当てられた IFoT スレーブノードを順次選択して，1 フレーム毎に送信することで負荷分散する役割を持つ．各 IFoT スレーブノードに対して処理タスクと処理ライブラリが送信された後は，処理タスクの内容に従って，IFoT スレーブノード群が処理を始める．

4.　評価実験

本章では，提案する IFoT ミドルウェアの基本処理性能を評価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用いた評価実験の内容を記す．

Vol.2017-MBL-82 No.40
Vol.2017-UBI-53 No.40
2017/3/10

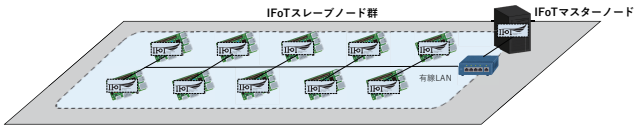


図 8 IFoT ミドルウェア評価システム

表 2 評価実験

	IFoT スレーブ× 10	IFoT マスタ × 1
CPU	ARM Cortex-A53 1.2 GHz	Intel core i7-2600 3.4GHz
Memory	1GB	8GB
DISK	64GB	500GB
OS	Raspbian jessie 7.10	Ubuntu 16.04

4.1 実験環境

実験に使用する評価システムの構成を図 8 に示す．本システムは，IFoT ミドルウェアを搭載した IFoT スレーブノード 10 台と IFoT マスタノード 1 台で構成されている．各ノードの計算機スペックを表 2 に記す．各ノードは，有線 LAN で接続されており，相互に通信が可能な状況である．

4.2 統一的 IoT データ流の生成機構の評価

4.2.1 実験内容

統一的な IoT データ流生成機構では，受信した IoT データ流に対して素早くメタデータを付加すると共に，MQTT プロトコル対応のパケットに変換し，実時間で再流通することが求められる．そこで本実験では，10byte から 100Mbyte まで 2 の乗数単位のサイズに設定したダミーデータ流を生成し，それぞれのサイズに関して，1 フレームにかかる一次データ流生成処理の時間を測定した．

4.2.2 実験結果

表 3 に，フレームサイズによる処理時間の変化を記す．表に示すように，1 フレームのサイズが 10byte から 10Kbyte までは若干の増減が見られるものの処理時間がほぼ変わらず，実時間で統一的な IoT データ流を生成できていることがわかる．100Kbyte から徐々に処理時間が増加しており，10Mbyte になると処理時間が 941.67(ms) となることが確認できる．処理時間を増加させる要因として，1 フレームのデータサイズが大きいことから読み込み処理と，送信処理に時間がかかっていることが考えられる．この結果からデータ流の 1 フレームが 10Mbyte 以下のデータ流に関しては，実時間で再流通可能であることが判明した．

4.3 高次 IoT データ流生成タスクの分散実行機構の評価

4.3.1 実験内容

高次 IoT データ流生成タスクの分散実行機構では，レシピに記された高次データ流生成要求を満たすために，レシピに記された処理タスク (if_task) に記されている許容遅延

表 3 フレームサイズによる処理時間の変化

	処理時間 (ms)	
Frame size	Ave.	Max.
10byte	15.13	15.59
100byte	15.13	15.57
1Kbyte	15.21	15.74
10Kbyte	15.57	16.38
100Kbyte	22.13	43.59
1Mbyte	105.97	119.54
10Mbyte	941.75	945.16

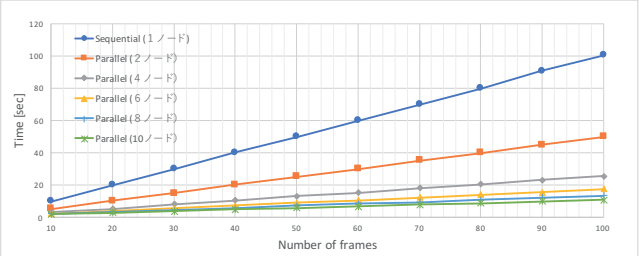
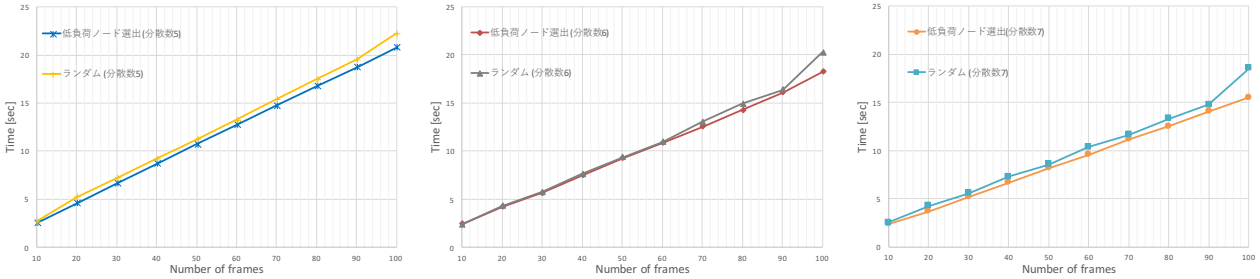


図 9 逐次処理と分散処理の処理時間の比較

(ex_delay) に基づき，処理の並列分散数を決定する．このとき，高次 IoT データ流が生成されるまでの許容遅延を満たすためには，IFoT マスタノードが決定した並列分散数の IFoT スレーブノードを用いて許容遅延以下で高次 IoT データ流の生成処理を完了する必要がある．また，高い並列分散処理パフォーマンスを得るために，最適なノードを選択する必要がある．本実験では，画像データ流から顔画像を検出する処理タスクを題材として扱い，高次 IoT データ流生成タスクの分散実行機構の処理性能を評価する．実験で用いる画像データ流は，あらかじめ撮影された動画データを使用し，1 フレームのデータサイズは 100Kbyte であり，生成レートは 10fps とする．負荷分散タスクを割り当てられた IFoT スレーブノードは，処理対象となる一次データ流を 1 フレーム毎に，加工処理タスクが割り当てらた IFoT スレーブノードに振り分ける役割を担う．今回の評価実験は，分散処理の対象となる IFoT スレーブノードが全て同じ処理能力であるため，データ流の振り分け方式は，ラウンドロビンを採用する．

まずはじめに，全 IFoT スレーブノードにタスクが分担されていない初期状態での分散処理性能を確認するために，IFoT スレーブノード 1 台を用いた逐次処理と IFoT スレーブノード 2〜10 台を用いた並列分散処理の性能比較実験を行う．次に，IFoT スレーブノードの半数が他のレシピによって既に使用されている状況で，許容遅延 (ex_delay) 10 秒で顔画像データ流の生成処理タスクを分担するシチュエーションを想定し，IFoT マスタノードが保有する情報に基づき低負荷なノードを選出する場合とタスクを分担するノードをランダムに選出する場合の処理性能比較実験を行う．



(a) 並列分散数 = 5　(b) 並列分散数 = 6　(c) 並列分散数 = 7

図 10　低負荷ノード選出とランダムなノード選出における処理時間の比較

4.3.2　実験結果

図 9 に，IFoT ミドルウェアによる逐次処理と並列分散処理の処理時間の比較を示す．フレーム数が増えるに従って，並列分散処理の方が逐次処理に比べて処理時間が短くなっていることが確認出来る．なお，100 フレーム 10 並列処理時に，約 92.46 %という高い並列化効率が得られている．

並列分散数 5, 6, 7 で行う顔画像データ流生成処理に関して，IFoT マスタノードが保有する情報に基づき選出した低負荷なノードを用いた場合と，ランダムなノード選出により選ばれたノードを用いた場合の処理時間の比較をそれぞれ図 10 に示す．

これらの結果より，並列分散数が 5, 6, 7 のいずれの場合も，ランダムなノード選出より低負荷なノードを選出した場合の方がより早く処理を完了していることが確認出来る．このことから，高い並列分散処理パフォーマンスを得るためには，タスクを分担する時点で処理負荷が低いノードに対して優先的にタスクを分担する手法が有効であることが示された．

5.　関連研究

5.1　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 [6]．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 [7], Axeda [8], ThingSquare [9], Thingworx [10], WoTkit [11], Xively [12])．一方，分散型の IoT プラットフォームとしては，OpenIoT [13], LinkSmart [14] 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．国内においては，IoT や参加型センシングにより生成される情報ストリームを取得し融合することで新たなサービスを実現する基盤として ClouT [15] の研究が存在する．ClouT では，IFoT のように，IoT デバイス自身の計算資源を活用した，クラウドレスでの分散処理を実現することは想定しておらず，高次 IoT データ流を複数融合させたリアルタイム活用も取り扱っていない．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 [16] [17] 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．

近年，通信インフラやクラウドで生じる負荷を軽減する概念として，小規模なデータセンタ（エッジ/フォグサーバ）を近距離に配置するエッジコンピューティング [3] やフォグコンピューティング [4] が注目を集めている．また，その概念を拡張し，P2P のような他の分散システムにクラウドサーバの処理タスクを委任する Edge-Centric Computing [18] の概念も提唱されている．エッジコンピューティングの実現に向けては，複数のコンピューティングノードが抱える様々な異種性を吸収するためのミドルウェアとして Tasklets [19] が研究されている．これらに対して，IFoT の特色は，従来のクラウド依存なアーキテクチャが抱える問題に対する現実的な解決策として，これらの概念を拡張し，データ発生源に近い IoT デバイス群自体を計算資源として活用する点である．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．

5.2　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 [20] [21] 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 [22] 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 データ流処理の高速化を図る．

6. おわりに

本稿では，IoT デバイスの上位で動作し，多種多様な IoT データ流の処理タスクをクラウドレスで分散実行する「IFoT ミドルウェア」の設計について述べた．IFoT ミドルウェアの動作検証を行うために，Raspberry Pi を用いて，各機構に関する性能評価実験を行った．実験の結果，統一的な IoT データ流生成機構に関しては，1 フレームが 10〜100Kbyte のデータ流に対して，素早くメタデータを付加し，実時間で流通可能であることがわかった．また，高次 IoT データ流生成タスクの分散実行機構に関しては，1 台のノードによるタスクの逐次処理と比べて，10 台のノードを用いた並列分散処理時に，約 92.46 %という高い並列化効率が得られた．また，タスクを分担する際に，低負荷なノードを選出することによって，高い並列分散処理パフォーマンスを得られることを確認した．

今後は，100 台規模の IoT デバイスを用いたフィールド実験に向けて，IFoT ミドルウェアの機能拡張を行うとともに，実時間性やスケーラビリティ，QoC (Quality of Computation) といった評価基準のもと，IFoT ミドルウェアの有用性を評価する．

謝辞 本研究の一部は，JSPS 科研費 16H01721 の助成によって行った．ここに記して謝意を示す．

参考文献

[1] IDC, The Digital Universe of Opportunities : Rich Data and the Increasing Value of the Internet of Things, https://www.emc.com/collateral/analyst-reports/idc-digital-universe-2014.pdf, (2014).
[2] [2] D. Evans, The Internet of Things—How the next evolution of the internet is changing everything, (2011).
[3] 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).
[4] 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).
[5] K. Yasumoto, H. Yamaguchi, H. Shigeno: Survey of Real-time Processing Technologies of IoT Data Streams, Journal of Information Processing, Vol. 24, No. 2, pp.195-202, 2016.
[6] Mineraud J, Mazhelis O, Su X, Tarkoma S : Contemporary Internet of Things platforms. arXiv Preprint arXiv:150107438 (2015)
[7] Arkessa, http://www.arkessa.com/
[8] Axeda, http://www.axeda.com/
[9] ThingSquare, http://www.thingsquare.com/
[10] Thingworx, http://www.thingworx.com/
[11] M. Blackstock and R. Lea : IoT mashups with the WoTKit, Proc. of IEEE Internet of Things (IoT), pp. 159-166 (2012).
[12] Xively, http://xively.com
[13] 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)
[14] 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)
[15] ClouT : Cloud of Things for empowering the citizen clout in smart cities, http://clout-project.eu/jp/
[16] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin : IoT gateway: bridging wireless sensor networks into internet of things, Proc. of IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC), pp.347-352 (2010).
[17] 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)
[18] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere : Edge-centric computing: Vision and challenges, ACM SIGCOMM Computer Communication Review, Vol.45, No.5, pp.37-42 , (2015).
[19] D. Schafer, J. Edinger, J. M. Paluska, S. VanSyckel, and C. Becker. Tasklets : "better than best-effort" computing, IEEE In 25th int'l conf. on Computer Communication and Networks (ICCCN), pp .1-11, (2016).
[20] 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).
[21] Q. Zhang, J. Liu, and W. Wang : Approximate Clustering on Distributed Data Streams, Proc. of the int'l conf. on Data Engineering (ICDE2008), pp. 1131-1139 (2008).
[22] C. Lei, E. A. Rundensteiner, and J. D. Guttman : Robust Distributed Stream Processing, Proc. of the int'l conf. on Data Engineering (ICDE2013), pp. 817-828 (2013).
[23] MQTT，http://mqtt.org/
[24] 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 int'l Joint conf. on Pervasive and Ubiquitous Computing (UbiComp 2015), pp. 695-705 (2015)
[25] T. Nishimura, T. Higuchi, H. Yamaguchi, T. Higashino: Detecting smoothness of pedestrian flows by participatory sensing with mobile phones, Proc. of the 2014 ACM int'l Symposium on Wearable Computers (ISWC 2014), pp.15-18, (2014).