FogFlow：クラウドとエッジを通じたIoTサービスのオーケストレーション

Bin CHENG　Ernoe KOVACS　北澤 敦　寺澤 和幸　羽田 亨　武内 守

要 旨

スマートシティ、スマートインダストリー、コネクテッドビークル向けにさまざまなIoTサービスを提供するIoTインフラプロバイダは昨今、応答時間短縮の要請に応じて、地理的に分散したインフラを準備する必要に迫られ、それらの管理に要する手間と費用の問題に直面しています。FogFlowは、通信量及び応答時間の低減を実現するため、クラウドやエッジ上でIoTサービスの動的なオーケストレーションを実現する分散実行フレームワークです。高いスケーラビリティと信頼性を備えた自動化・最適化されたIoTサービスを提供することで、インフラプロバイダが運営コストを大幅に低減するための支援をします。また、FogFlowは、サービス開発者やシステムインテグレータが、安い開発コストと短い開発期間でIoTサービスを素早く実現するために必要なデータセントリックなプログラミングモデルと開発ツールチェーンを提供し、実際に、研究所だけではなく、NECソリューションイノベータのスマートシティプロジェクトでも実証されました。

KeyWords

Internet of Things（IoT）／エッジコンピューティング／サーバレスコンピューティング／サービスオーケストレーション／ダイナミックデータフロー

1. はじめに

Internet of Things（IoT）では、より多くのモノ（コネクテッドカーや飛行ドローン、住宅など）が接続され、それらのモノがさまざまなセンサーやデータソースから得たデータを利用することで状況をリアルタイムに感じ取り、それに応じて反応するハイパーコネクテッドな社会が可能になります。そして、これらのモノのスマートさは、バックエンドサービスとしての高度なデータ処理ロジックが実現するものです。制約を受けているIoTデバイスをスマート化するために解くべき課題の1つは、いかにバックエンドサービスロジックを、素早く簡単にそして効率的にオーケストレーションするかです。そうすることで、それらのデバイスが可能な限り素早く反応し、十分なデータを利用できるようになるのです。

従来のビッグデータ分析とは異なり、IoTサービスのオーケストレーションでは、以下に挙げる課題を考慮する必要があります。

（1）センサーは、時間の経過に伴う一貫したデータを生成しており、すべてのローデータ（生データ）を集中管理されたクラウドに送信することは、通信量と応答時間を上昇させるため、経済的に持続可能ではありません。データソースに近いエッジへ、より多くのデータ処理を動的に開放する必要があります。

（2）データ及びそこから得られた知見は、デバイス、サービス、アプリケーションそしてプラットフォーム間で共有・交換すべきです。このようなハイパーコネクテッドIoTシステムは、データ資源を共有する同じエッジ及びクラウド環境内で接続されている数千、数万ものIoTアプリケーションの実行を管理する必要があります。

（3）システムの作業負荷はより動的です。デバイス、サービス、エンドユーザーアプリケーションなどこれらが出現し、動き回り、再接続し、消滅しています。このため、作業負荷は常時変化します。

（4）多くのIoTサービスは、迅速な応答時間を必要とします。

（5）バックエンドインフラは、さまざまなレイヤにおいて、非常にヘテロジーニアス（異機種混在）でかつ地理的に分散されたエッジノードのリソースを管理する必要があります。

これらの技術上の課題はすべて、インフラプロバイダがIoTサービスを管理する際の、新しい課題と複雑な問題と

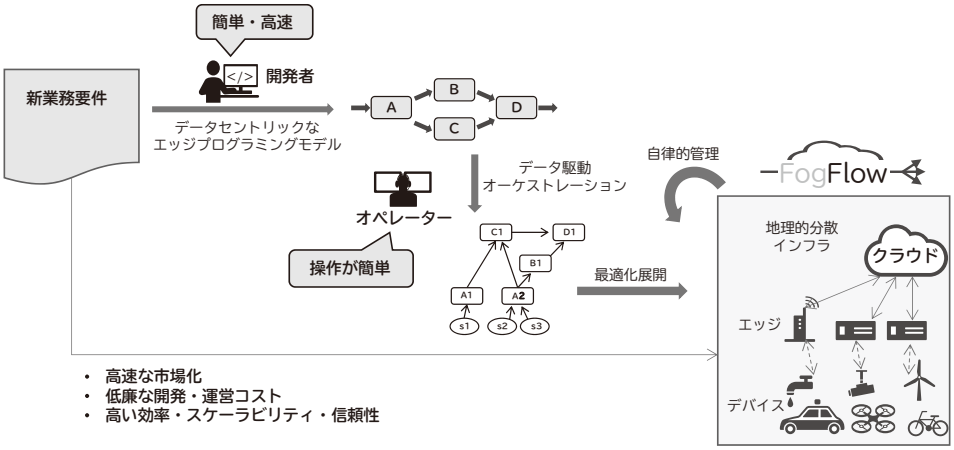


図1　FogFlowを使った各種の提案

なります。FogFlowは、これらの問題を抱えるすべてのインフラオペレーターが、複雑な課題を取り扱い、地理的に分散された共有環境内での各種IoTサービスを自動的にかつ効率的に管理できるよう設計されています。

FogFlowは、**図1**に示すように、IoTプロバイダがさまざまな業務需要のために簡単・迅速にサービスを提供できるよう、標準ベースのデータセントリックなエッジプログラミングモデルを提供しています。また、データ駆動で最適化されたサービスオーケストレーション機構を利用して、都市規模のIoTサービスの提供に必要な数千、数万ものクラウド及びエッジノードを、インフラプロバイダが自動的・効率的に管理し、最適な性能を実現することをサポートします。これによりスマートシティやスマートファクトリのような大規模IoTプロジェクトでの開発及び運営コストを節約し、生産性を向上し、市場化までの期間を短縮し、更にはスケーラビリティと安定性を増大することが可能となるのです。

2. FogFlow

FogFlowにおけるIoTサービスの定義は、リンクされたオペレーターのデータ処理フローであり、それらはグラフによって表されます。各オペレーターはIoTデバイスまたは上流にある処理フローから入力を得て、IoTサービスの業務ロジックを実行し、中間結果を次のオペレーターに渡します。各オペレーターはDocker（ドッカー）のアプリケーションとして実現され、FogFlowによって専用

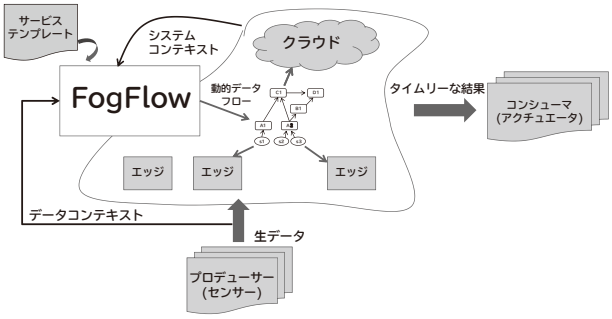


図2　FogFlowの高レベルモデル

Dockerコンテナ内部のタスクとして実行されるようにインスタンス化されます。タスク同士は実行時に入力と出力のデータ従属性に基づいてリンクされます。

図2に示すように、IoTサービスは必要なデータ処理を行うために、プロデューサー（例えばセンサー）とコンシューマ（例えばアクチュエータやアプリケーション）の間の動的データ処理フローとしてオーケストレーションされます。以下に示す手順は、IoTサービスのオーケストレーションを実施するために設計されたものです。

はじめに、サービス開発者は、サービスロジックを定義するためのサービステンプレートを作成します。その際、Webベースのグラフィックフローエディタである FogFlowタスクデザイナーを使用します（**図3**）。サービステンプレートはIoTサービスの抽象的・静的なデータ処理ロジックを表しており、そこには、ある種別の出力を生成するために必要な入力を獲得するためにどのオペレー

図3　FogFlowタスクデザイナー

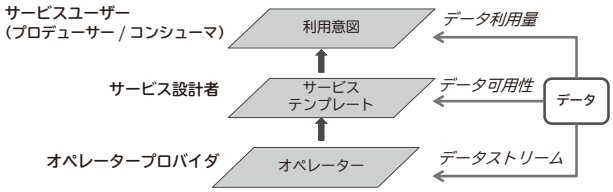


図4　FogFlowでのさまざまな役割

ターが利用されているか、及びオペレーターをトリガーするタイミングと方法などの詳細が含まれています。サービスプロバイダは、他者の登録したオペレーターを再利用することも、自分でオペレーターを作成することも可能です。

サービステンプレートを登録すると、FogFlowのランタイムは利用可能なデータのコンテキストを監視し、登録されたサービステンプレートをインスタンス化すべきタイミングと方法を判定します。例えば、IoTゲートウェイにデータが到着した時点でフローを開始することも可能です。FogFlowは各オペレーターにいくつのタスクインスタンスが必要か、または各タスクインスタンスに必要な詳細な構成を決定します。すべてのデータフローのデータ構造は、NGSI[1)]と呼ばれる同一の標準化データモデルに基づいて記述されています。これにより、FogFlowはどのエッジノードでどのような種別のデータが作成されているかを知ることができ、更には登録されたコンテキストメタデータの利用可能性に応じて各エッジノードのデータ処理フローをトリガーし起動することが可能となります。

最終的に、FogFlowは、いくつかの最適化アルゴリズムを使用して、各エッジノードで利用可能なリソース数、データソースの位置、予測された作業負荷の経時変化などを含むリアルタイムシステムコンテキストに応じて、どのタスクインスタンスをどこに展開するかを決定します。

Azure IoTエッジやAmazon Greengrassなど他のエッジコンピューティングフレームワークと比較して、FogFlowは以下に挙げる独自の特徴を備えています。

データセントリックなプログラミング：FogFlowはデータセントリックなプログラミングモデルを提供し、各種の役割がよりデータ処理をベースとし、直感的な方法で各レベルでの目標を表現可能にすることで、IoTサービスの設計と利用を容易にしています。

図4に示すように、オペレーターレベルでは、オペレーターの作成者は自己のオペレーターが取り扱えるデータの種別、提供する機能、及び生成すべき結果の種別を指定することのみが必要です。サービスレイヤでは、サービス設計者はさまざまなオペレーターを利用するサービステンプレートを数分で構築可能です。その後の実行時間中、FogFlowはサービスユーザーの定義した高レベルのデータ利用意図に基づいて自動的にデータ処理フローのオーケストレーションを行います。

サービスユーザーは、データのプロデューサーまたは生成されたデータのコンシューマのどちらかになります。プロデューサーの立場からは、生成したデータに適用すべきサービスロジックの種別を指定でき、コンシューマの立場からは生成してほしいデータの種別を指定できます。FogFlowが行うことは、それらの高レベルの利用意図を具体的なデータ処理フローに翻訳し、続いてそれらをクラウド及びエッジノード上にシームレスに展開・保守することです。より重要なことは、このデータセントリックなプログラミングモデルでは、標準データモデルに基づいて内在するデータ処理フローを多数のサービスやユーザー間で共有し最適化できるという点です。FogFlowは、インフラプロバイダがデータ駆動のエコシステムと経済性を求めて移行するために必要な独自のベースを提供します。

自律的管理：FogFlowは、分散した自律的手法でIoTサービスのオーケストレーションの決定を実現します。これは、FogFlowの各エッジノードは、ローカルなコンテキストの視点のみに基づいて独自の決定を行うことが可能であることを意味しています。大多数の作業負荷はこのようにしてエッジで直接、中央のクラウドに頼ることなく常時処理できるのです。この「クラウドレス」アプローチでは、FogFlowは応答時間の高速化を可能にするだけでな

く、高いスケーラビリティと信頼性を実現します。

最適化展開：FogFlowでは、タスクの構成とデータ処理フローの展開は、さまざまな目標に合わせてクラウド/エッジ環境に最適化されています。例えば、クラウドノードとエッジノード間の内部データトラフィックの最小化、期待される利用可能な結果をローデータから生成するための応答時間の最小化、生成した結果の信頼性の最大化などです。データ処理フローの最適化は、あるサービスの起動の最初に行われるだけではなく、サービスの寿命全体にわたって続行されます。このような最適化行動には、コネクテッドカーや飛行ドローンのような移動体に対する応答時間を最小に保つことを目的とした、あるエッジから別のエッジへのタスクの移行があります。

3. エッジプログラミングモデル

現在、FogFlowは各種のワークロードパターンをサポートするため、以下２種類のプログラミングモデルを提供しています。

3.1 サービストポロジー

第一のワークロードパターンは、コンシューマが出力データをサブスクライブした場合にのみ、同じ出力データを生成するために必要な処理フローを起動することです。このパターンに基づいてIoTサービスを定義するには、サービスプロバイダは、リンクされたオペレーターで構成されかつ各オペレーターが特定の粒度でアノテーション（注釈付け）されているようなサービストポロジーを定義する必要があります。FogFlowは各オペレーターの粒度を考慮に入れ、当該オペレーターがインスタンス化すべきタスクインスタンスの数を、利用可能なデータに基づいて決定します。サービストポロジーは、いずれかのコンシューマまたはアプリケーションが発行する要件オブジェクトに従って明示的に起動されます。この要件オブジェクトは定義されたサービストポロジーの処理ロジックのどの部分を起動すべきかを定義しており、またオプションとして起動された処理ロジックを適用するためのデータソースをフィルタするための特定のジオスコープ（地理的範囲）を定義することもできます。詳細は、著者らの論文[2)]及びオンラインチュートリアル[3)]を参照してください。

3.2 Fog関数

第二のワークロードパターンは、ストリーム処理手順の正確なシーケンスについて、サービス設計者が先験的には知らないというシナリオのために設計されたものです。代わりに、このような設計者は所与の種別の情報を取り扱う特定のオペレーターを含むFog関数を定義できます。これにより、FogFlowはすべてのFog関数の記述に基づいて処理フローのグラフを作成できます。

サービストポロジーとは異なり、Fog関数はオペレーターの数は１のみという非常に単純なトポロジーを持ち、入力データが利用可能になった時点でトリガーされます。FogFlowは新規データに対して、複数のFog関数を用いたり、それらを自動連鎖することが可能であるため、データの着信と消滅に従い、FogFlowのランタイムは常時変化する実行グラフを自動的にトリガーし管理することができます。

設計の視点からは、Fog関数は、とあるIoTサービスの全体的処理ロジックを新しい業務要件のために変更する必要が生じた場合には、Fog関数を追加または削除することで簡単にロジックの経時変更が行えるため、サービストポロジーよりもフレキシブルです。FogFlowは、Fog関数プログラミングモデルを使って、クラウド - エッジベースの環境に適したサーバレスコンピューティングをサポートします。

4. ユースケース

第４章では、サービスプロバイダがどのようにFogFlowプログラミングモデルを利用して、さまざまなスマートIoTサービスを実現できるかを説明します。以下では、サービストポロジーに基づいたケースとFog関数に基づいたケースの２通りの具体的なユースケースを紹介します。

4.1 迷子ファインダー

これは、FogFlowによって可能となるエッジコンピューティングを使って、迷子をできるだけ迅速に発見するためのものです。例えば、あるスマートシティ内に、そしてそのシティ内の各スタジアム内にも多数のカメラが展開されているとします。各種のIoTサービスをサポートするため、IoTゲートウェイのようなエッジノードが、この都市のインフラの一部としてさまざまな場所に数多く展開され、

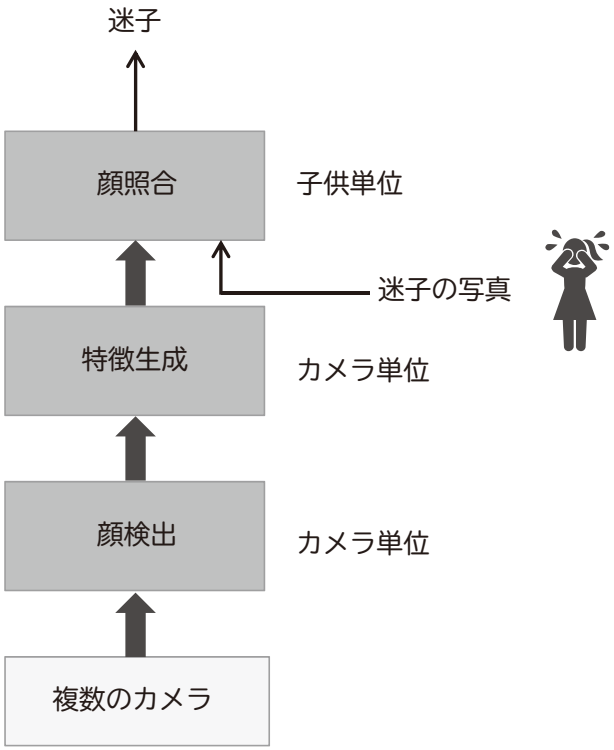


図5　迷子ファインダー

FogFlowによって管理されています。

FogFlowを利用した、容易に実現可能なサービスには、**図5**に示すサービストポロジーを使用した迷子発見サービスがあります。設計されたサービストポロジーは、カメラのビデオストリームから顔画像を認識し抽出する顔検出、検出した個々の顔について特徴ベクトルを算出する特徴生成、検出した顔を迷子の顔の特徴ベクトルと比較する顔照合、これら3種類のオペレーターで構成されています。サービストポロジー内の各オペレーターには別々の粒度が定義されており、例えば顔照合は子供単位でインスタンス化されていますが、他の2種のオペレーターはカメラ単位でインスタンス化されています。

このサービストポロジーを起動するために、外部のアプリケーションは要件オブジェクトを発行し、顔照合オペレーターの出力結果をサブスクライブします。外部アプリケーションは、この要件内に定義したジオスコープを変更することでFogFlowをコントロールし、ジオスコープの変動に合わせてサービストポロジーをオーケストレーションすることができます。こうして、最初は小さな範囲で迷子を捜し、見つからない場合には探索範囲を段階的に広げていくことが可能となります。

このユースケースでは、変動するジオスコープに対してデータ処理フローの動的オーケストレーションを行うという複雑な問題はFogFlowが扱うため、コンシューマとしての外部アプリケーションは非常に単純になります。更に、著者らの実験によれば、クラウドベースのアプローチと比較すると、通信量は95%の低下が達成されました。

4.2 スマートパーキング

これは、プロジェクトパートナーであるムルシア大学とともに、ムルシアの街の実際のシーンに基づいて実施したものです。ムルシアには、市政府が運営する規制駐車ゾーン（ここでは毎日の駐車スポットの利用履歴情報が得られる）と、私企業が運営する私有駐車施設（ここでは駐車スポットの利用可能性情報がリアルタイムで得られる）の2種類の駐車施設があります。NECのスマートパーキングサービスは、これら2施設からのデータソースとその他の公共輸送情報を利用して、個々の運転者にリアルタイムでそれぞれの運転者に合わせた駐車場の推奨情報を提供します。

このユースケースにサービストポロジーを適用するのは簡単ではありませんが、必要とされるデータ処理ロジックの実現はFog関数を利用することで容易になります。**図6**に示すように、必要なことは、当該ユースケースに関与する個々の物理的オブジェクトに、専用のFog関数を設計し実行することだけです。例えば、個々の公共施設において利用できる駐車スポットの数を、過去の履歴情報に基づ

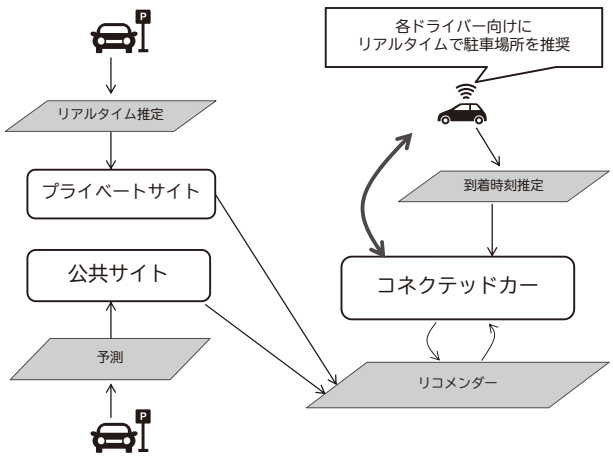


図6　スマートパーキング

いて10分ごとに予測するためには、1つのFog関数を設計・実行し、また、個々のコネクテッドカーについては、道路の交通状況に応じて到着時刻を推定するためのFog関数と、到着時に運転者はどの駐車施設の駐車スポットを使用できるかを計算するためのFog関数の2つを作成・実行します。これらのFog関数インスタンスの展開はそれらの入力データソースに近いエッジノードに置かれるので、FogFlowを利用することで通信量は50%低下でき、また各運転者に対するリアルタイムでの推奨駐車場情報も提供できます。

5. FIWAREとの相互作用

2017年11月から、FogFlowはFIWARE（ファイウェア）[4]コミュニティにおいて、Generic Enabler（GE）のインキュベーテッドオープンソースとして奨励されています。このコミュニティにおいてFogFlowはクラウドエッジオーケストレータという独自の地位を保ち、クラウドとエッジを通じて動的データ処理フローのシームレスな起動と管理することで、データの採集・変形、更には高度分析をも行っています。**図7**に示すように、FogFlowは他のFIWARE GEと協働し、以下に示す2つのレイヤにおいてFIWAREベースのIoTプラットフォームの強化が可能です。

上層レイヤでは、FogFlowはOrion Brokerと標準化NGSIインタフェースを経由して、以下に示す2つの方法で協働します。第一の方法は、Orion BrokerをあるFogFlow IoTサービスの生成するコンテキスト情報の目的地と見なすことです。この場合、外部アプリケーションかFogFlowがNGSIのサブスクリプションをするか、またはFogFlowダッシュボードがFogFlowに要求されたコンテキスト更新を特定のOrion Brokerに転送することを要求しなければなりません。第二の方法は、付加的情報を提供するデータソースとしてOrion Brokerを見なすことです。この場合、必要な情報をフェッチするための単純なFog関数をFogFlowシステムに作成することができます。どちらの方法でも既存のFIWAREシステムに変更を加える必要はありません。したがって、この種の統合は高速にかつほとんど労力を使わずに実行することができます。

下層レイヤでは、MQTT、COAP、OneM2M、OPC-UA、LoRaWANなどのNGSIがサポートしていないデバイスと統合するため、FogFlowは既存のIoTエージェントを再使用し、それらをFog関数プログラミングモデルに基づいてFogFlowアダプタに変換します。また、FogFlowはこれらのアダプタを使用して、エッジでの直接のデバイス統合のために必要なアダプタを動的に起動します。このようにして、FogFlowは広範囲なIoTデバイスとの対話を可能にします。

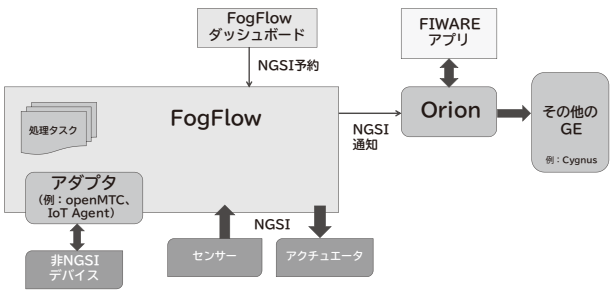


図7　他のFIWARE GEとの統合

6. おわりに

FogFlowは、IoTサービスの動的データ処理フローをクラウドとエッジ上にてシームレスでスケーラブルに管理し、IoTサービスのオーケストレーションを実現する分散型実行フレームワークです。本稿では、その設計目的、主要な技術特徴、業務上の価値提案を紹介しています。また、オンデマンドデータ処理のためのサービストポロジーやサーバレスエッジコンピューティングのためのFog関数を含むそのプログラミングモデルの概要も述べました。またこれら2つのプログラミングモデルを、実際の都市スケールのIoTサービスに使用する例として2つのユースケースを提示しました。

FogFlowはオープンソースFIWARE GEとして奨励されており、他のFIWARE GEとの協働も容易に実行できます。FogFlowは、標準ベースのデータ駆動エッジコンピューティングフレームワークとして、インフラプロバイダが自己のインフラをデータ駆動のエコシステムと経済性を追求してオープン化することを可能にするための独自のベースを提供します。

参考文献

1) FIWARE-NGSI v2 Specification
http://fiware.github.io/specifications/ngsiv2/stable/
2) Bin Cheng et al. ：FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities, IEEE Internet of Things Journal, Volume 5, Issue 2, pp.696-707, 2018.4
https://ieeexplore.ieee.org/document/8022859/
3) FogFlow Tutorial
http://fogflow.readthedocs.io
4) NECプレスリリース：NEC Develops a FIWARE-based Fog Computing Framework for Edge-based IoT Services, 2017.11
https://uk.nec.com/en_GB/press/201711/20171127_01.html

執筆者プロフィール

Bin CHENG
NEC Laboratories Europe
Senior Researcher

Ernoe KOVACS
NEC Laboratories Europe
Senior Manager

北澤 敦
NECソリューションイノベータ株式会社
プロフェッショナルフェロー

寺澤 和幸
未来都市づくり推進本部
シニアマネージャー

羽田 亨
NECソリューションイノベータ株式会社
IoT基盤ソリューション事業部
エグゼクティブエキスパート

武内 守
NECソリューションイノベータ株式会社
IoT基盤ソリューション事業部
マネージャー

NEC 技報のご案内

NEC技報の論文をご覧いただきありがとうございます。
ご興味がありましたら、関連する他の論文もご一読ください。

NEC技報WEBサイトはこちら

NEC技報（日本語）　NEC Technical Journal（英語）

Vol.71 No.1　データを活用した持続可能な都市経営特集

データを活用した持続可能な都市経営特集によせて
データ利活用型スマートシティの始動

◇ 特集論文

データを活用した都市経営のビジョン
世界のデータ利活用型スマートシティ開発動向
持続可能な社会に向けた都市経営へのパラダイムシフト

データ利活用型スマートシティの実証・実装事例
データを活用した都市経営の海外事例
FIWAREを活用したスマートシティ向け共通プラットフォームの構築（高松市事例）
豊島区における「群衆行動解析技術」を活用した総合防災システム
訪日外国人向けのおもてなしサービスの高度化と地域活性化への取り組み事例
自治体データ活用事例 ～財務・子育て・地域振興などのさまざまなデータ活用～

シティマネジメント技術
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
FogFlow：クラウドとエッジを通じたIoTサービスのオーケストレーション
スマートシティIoTに求められるセキュリティ要件と技術
欧州におけるスマートシティとSociety 5.0の実現へ向けての標準化の動向
都市評価指標標準とその活用

地域共創
地域共創基盤としての「スマートシティたかまつ推進協議会」
枠を超えた共創活動「せとうちDMO」の立ち上げ
包括連携協定による地域共創
「新たな行政サービス共創研究会」が創るこれからのあたりまえ

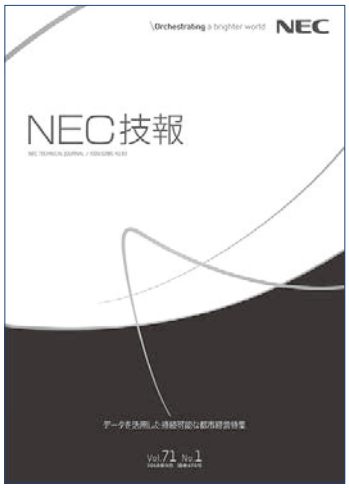
◇普通論文

スピン流熱電変換 ～インフォマティクスを活用した材料開発と適用領域～
NanoBridge-FPGAによるIoTデバイスの低電力・高性能化
IoTデバイス応用に向けたナノカーボンの材料開発
Hyperledger Fabric 1.0を用いた金融領域におけるブロックチェーン技術検証

◇ NEC Information

C&Cユーザーフォーラム＆iEXPO2017　Orchestrating a brighter world
基調講演
展示会報告

NEWS
2017年度C&C賞表彰式開催



Vol.71 No.1
（2018年9月）

特集TOP