「マルチメディア，分散，協調とモバイル
(DICOMO2020)シンポジウム」 令和2年6月

CityFeder: 異種スマートシティ基盤を柔軟につなぐプログラマブル・フェデレーション機構

米澤 拓郎[1] 河崎 隆文[2] 吉田 拓人[1] 伊藤 友隆[2] 上津原 一利[3] 古城 篤[2] 中澤 仁[2] 河口 信夫[4]

概要：本論文では，異なるスマートシティ基盤間を社会状況に応じて動的に接続可能とするプログラマブル・フェデレーション基盤 CityFeder を提案する．動的に変化する都市状況に対応し，住民の安心安全や QoL を高め，効率的な都市運営を可能とするスマートシティでは，社会的に価値のある多種多様なセンサデータが収集される．一方，現実世界の問題は都市内に閉じることは稀であり，大規模災害や疫病などが発生した場合は，複数の都市で問題対処にあたる必要があり，スマートシティ間でデータを共有可能とするべきである．CityFeder はこの問題を解決するため，ビジュアルプログラミングおよび実行環境である Node-RED を拡張し，社会状況に応じたスマートシティ基盤間のデータ流通を実現する．本稿ではそのコンセプト，設計と実装を述べるとともに，具体的な応用シナリオを実現可能であることを示す．

CityFeder: A Programmable Federation Architecture for Adaptive Connecting of Heterogeneous Smart City Platforms

TAKURO YONEZAWA[1] TAKAFUMI KAWASAKI[2] TAKUTO YOSHIDA[1] TOMOTAKA ITO[2]
KAZUTOSHI UETSUHARA[3] ATSUSHI KOJO[2] JIN NAKAZAWA[2] NOBUO KAWAGUCHI[4]

1. はじめに

人口の都市集中，高齢人口や観光客の増加などによる急激な社会変化から，様々な社会問題が起こっている．また，台風や地震に代表される大規模災害が頻発し，2020 年には新型コロナウイルス（COVID-19）の流行も発生するなど，住民の生命に関わる深刻な出来事が起こっている．これらの社会問題は広域に渡って発生するため，複数の都市が連携して問題解決にあたる必要がある．近年，都市運用の効率化や諸問題の解決を ICT 技術により実現するスマートシティが注目されている．これまでスマートシティを実現するための様々なプラットフォームが提案されているが，統一化されたルールやプラットフォーム内部で利用するプロトコルも多々存在するため，異なるスマートシティ間の相互運用性が低いことが問題となる．上述した広域の社会問題への対応には複数の都市の連携が必須となるため，今後は社会状況に応じ，複数のプラットフォームの相互運用性を高め，データのリアルタイム流通や機能の相互利用を実現する必要がある．

本研究では，日々変化する社会状況に応じ，異なるスマートシティ基盤間で任意のデータの相互流通を可能とする CityFeder を提案する．個々の都市には多種多様なデータが存在し，平時から他都市と共有可能なデータもあれば，有事の際のみ共有が許されるデータも存在することが考えられる．すなわち，様々なデータに対するアクセス権を，社会状況に応じて変更可能とする必要がある（要件 1）．また，それぞれのプラットフォームは互いに異なるプロトコルを利用していることが想定されるため，プロトコルやデータフォーマットの相互変換が必要となる（要件 2）．これらの課題を解決することで平時・有事の様々な状況に対応した異種スマートシティ間の柔軟な連携が実現し，都市間横断型の様々なサービス開発に寄与することができる．

上述した 2 つの要件を満たしつつ，開発の容易性を高めるため，CityFeder はオープンソースソフトウェアの Node-RED を活用した設計・実装を行った．Node-RED は

1 名古屋大学大学院工学研究科
2 慶應義塾大学大学院政策・メディア研究科
3 フリーランス
4 名古屋大学未来社会創造機構

ノードという処理単位を GUI 上でつなぐ操作で簡単に一連の処理を開発・実行可能であり，豊富なプロトコルに対応している．CityFeder では Node-RED に拡張機能を施すことで，そのフローを通じ，社会状況の判別（要件 1）や，プロトコル・データ変換（要件 2）を実現する．CityFeder の全体概要を図 1 に示す．CityFeder は中央のサーバ（Web サイト機能と Node-RED 環境）と，各スマートシティ基盤上で動作する Node-RED 環境から構成される．CityFeder サーバでは任意の開発者から社会状況の登録を受け付けるとともに，それを API に変換し各都市に配信するサービスを提供する．各都市は Node-RED 環境内で用意されたアクセス権変更ノードやデータアクセスノードを用い，社会状況に応じたフェデレーションを実現する．また，本研究では災害時における活用など，具体的なアプリケーション・シナリオを提案し，その実現を行った．本研究の主な貢献は下記の 3 点である．

- 社会状況に応じたスマートシティ基盤間の動的な接続をプログラマブル・フェデレーションと定義し，その概念を提案すること
- プログラマブル・フェデレーションを実現するアーキテクチャ CityFder の設計と実装を行ったこと
- 実装した CityFeder を用い，災害時の物流支援シナリオの技術実証を行ったこと

2. 異種スマートシティ基盤間のプログラマブル・フェデレーション

2.1 問題

人口集中の進む都市の運営を効率化し，住民の QoL を高める多種多様なサービスを実現するため，スマートシティに関する取り組みが注目されている．一般的にスマートなシステムとは，リアルタイムな状況把握と，その状況に対応する高い応答性を示すシステムを指す．よって，スマートシティとは，都市規模において，その状況把握（Awareness）と応答性（Responsiveness）を提供可能な都市である．都市は人，建物，車両，都市インフラ（道路，下水道など）など多様な実空間オブジェクトと，それらによって構成される多種多様な組織・社会（家族や企業含む）の集合により成立している．よって，都市をスマート化する一つの手段は，その構成要素である実空間オブジェクトおよび組織・社会をスマート化し，そこで得られた情報を異種組織間で共有したり，都市レベルで共有することである．

一方で，我々は都市間を移動して生活を行っているし，大規模災害や新型コロナ問題等は複数都市，国レベルにまたがって発生する．よって，生活の様々な場面を支援したり，また有事において問題解決に複数の自治体で対応する際などは，スマートシティが収集する情報はぞれぞれの都市ごとに閉じるべきではなく，適切な都市間で共有されるべきである．都市が収集する情報にはプライバシに関するデータも存在するため，平時から共有可能なデータもあれば，有事の際にも共有すべきデータが存在する．従って，社会状況に応じた，柔軟な都市データへのアクセス権変更が必要となる（要件 1）．また，データを共有する際には，スマートシティに限らずデータにアクセスするための接続方式（プロトコル）と，データの表現形式（フォーマット）の差異をどう吸収するかが問題となる（要件 2）．これらの要件を同時に満たすシステムは，筆者らの知る限り存在せず，本研究はこれらの課題に取り組む．

2.2 目的

本研究の目的は，上述した課題を解決するために，社会状況に応じた柔軟なスマートシティ間のデータ共有を実現するアーキテクチャの設計とその実現を行うことである．本研究では，「社会状況に応じたスマートシティの相互接続」を，**プログラマブル・フェデレーション**と呼ぶ．プログラマブル・フェデレーションを実現すると，例えば「災害警戒レベル 3 が発令された場合は気象情報に関するあらゆるデータを他自治体へ開放する」「緊急事態宣言が発令された際には人口メッシュデータのデータを他自治体へ開放する」などの柔軟なデータ共有が可能となる．これにより，社会状況の変化に紐付いた柔軟かつセキュアなデータ共有が可能となり，多種多様なサービスの構築が期待される．このプログラマブル・フェデレーションを実現するアーキテクチャを CityFeder と呼び，設計と実装を行う．

2.3 ユースケース

下記に，CityFeder が実現するプログラマブルフェデレーションによって可能となるシナリオについて述べる．

2.3.1 災害時物流支援シナリオ

本ユースケースは，災害時が発生した際に，協力自治体が相互に様々なデータの共有を必要とするシナリオである．特に本研究では，被災地の避難所における物資情報が今後スマートシティ基盤へ集約されることを想定し，その情報を災害発生時に速やかに他自治体に伝達可能とするメリットについて着目する．被災地でどういう物資が足りており，どういう物資が足りていないか，といった情報は日々変化し，その状況に応じた支援物資を他自治体は送る必要がある．これまでは物資を必要とする自治体・避難所と，物資を送って支援したい自治体・組織との間での相互情報交換が不十分であり，ミスマッチがおきていた．本ユースケースでは，CityFeder が災害を検出すると，事前にデータ共有に合意していた自治体間で物資情報を共有し，在庫可視化サービスや支援物資需給交換サービスなどにより効率的な支援物資マネジメントを実現しようとするシナリオである．

2.3.2 人流データ連携シナリオ

本ユースケースは，大規模イベントが発生した際に，協

力自治体や協力民間企業が相互に様々なデータの共有を必要とするシナリオである．特に本研究では，大規模イベントが行われている地域やその周辺地域における滞在情報や交通移動情報が今後スマートシティ基盤に集約されることを想定し，その情報をイベント発生時に速やかに他自治体や他民間企業へ伝達可能とするメリットについて着目した．スポーツ大会や国際会議などの大規模イベント発生時において，多くの人が利用するホテルの予約情報や交通手段の混雑情報などは平常時より変化が激しく，快適な滞在・移動を行うためにはその状況に応じた情報共有が必要である．また，ホテル滞在時の相部屋可能情報や車移動時の相席可能情報などの平常時では公開されていない情報を大規模イベント時に公開することで快適な滞在・移動を可能にする．これまでは大規模イベント時における滞在者数，移動者数などの把握や相互情報交換が不十分であるためオーバーブッキングなどが発生し，滞在・移動に対する問題が生じていた．本ユースケースは，CityFeder が大規模イベントを検出すると，事前にデータ共有に合意していた自治体や民間企業間で平常時では公開されていないホテル空き情報，交通移動情報などの情報を共有し，予約情報や交通情報の可視化サービスを介して滞在時の効率的な情報マネジメントを実現しようとするシナリオである．

2.4 関連研究

これまで，スマートシティのための情報基盤を実現するため，センサやアクチュエータのためのミドルウェア [1], [2]，テストベット上のアプリケーション開発のためのツール群 [3], [4]，実際の都市へのデプロイメントの事例 [5] など，様々な研究が行われてきた．特に，FIWARE 基盤 [6] は欧州を中心として複数の都市のスマートシティ基盤として採用され，利活用が進められている．これらの基盤ではセンサデータ収集，流通，可視化など多種多様な機能が含まれているが，IoT データアクセスのためのミドルウェアとしては，RESTful 型のデータアクセスを基本とする HTTP プロトコルや CoAP [7] プロトコルの他，Publish-Subscribe 型を採用する MQTT[8] や ZeroMQ[9] などが広く利用されており，MQTT は Amazon 社が提供する AWS IoT プラットフォーム[*1]にも採用されている．スマートシティ基盤間でシームレスなデータ流通を行うためには，理想的にはプロトコルの統一化が望ましいが，それぞれのプロトコルには一長一短があり [10]，現実的ではない．よってこれらのプロトコルの差異を埋め，データの相互流通性を高めるインタオペラビリティの研究がなされている [11], [12]．一方，これらの研究はフレームワークの提案に留まっていたり，オープンソースであったとしても活発な開発が現状はなされておらず，様々なスマートシティ基盤が容易に扱える状況ではない．

また，本研究では社会状況に応じたアクセス権制御を行うため，社会状況をどう認識するかが重要となる．これまでユビキタスコンピューティング分野では社会状況やイベントを抽出する様々な研究がなされてきた．例えば Lee らはソーシャルネットワークデータを解析し，その増加率によるイベント検出の手法を提案している [13]．また，下坂らは携帯電話の位置情報履歴を用い，都市の活動人口予測手法を提案している [14]．これらの研究は，個別の社会状況を認識するための手法としては有用ではあるが，本研究が対象とするように多種多様な社会状況を認識するとともにそれを複数のスマートシティ基盤間で共有し，アクセス権変更の手段とするようなプロセスと一体化して検討されてはいない．

本研究では，上記の既存研究も踏まえ，多種多様な社会状況に基づき，スマートシティ基盤に対する動的なアクセス権変更を実現するとともに，異種プロトコル間でのデータ流通を可能とするアーキテクチャの構築を行う．

3. 設計

3.1 本研究が対象とするスマートシティ基盤

プログラマブル・フェデレーションを実現するために，本研究では 1）共通した社会状況を認識し，各スマートシティ基盤へ配信可能な社会状況認識機能，2）認識した社会状況に基づくダイナミックなアクセス権制御機能，の 2 つをコア機能とした基盤間接続モデルを考案した．現在，様々なスマートシティ基盤が存在し，提供されている API や機能も多々存在する．本研究の対象を明確化するため，下記の要件さえ満たせばプログラマブル・フェデレーションが利用できるとする．

本研究が対象とするスマートシティ基盤

スマートシティ基盤に (1) データアクセスのための API，および (2) アクセス権の変更が可能な API，の最低限 2 つの機能が備わっていれば，他の基盤と柔軟に接続が可能なプログラマブル・フェデレーションの対象とする．

3.2 CityFeder アーキテクチャ

CityFeder の全体概要を図 1 に示す．CityFeder は CityFeder サーバと，各都市 OS 毎に設置される Node-RED 環境の大きくわけて 2 種類の機能から構成される．CityFeder サーバは，各スマートシティ OS から登録される社会状況の定義とその社会状況に対応してアクセス権が付与されるデータのカタログを保持する．また，登録された社会状況を認識し，各都市 OS に伝達する機能を有する．CityFeder サーバの詳細は，4.1 節で詳しく述べる．また，

[*1] https://aws.amazon.com/jp/iot/

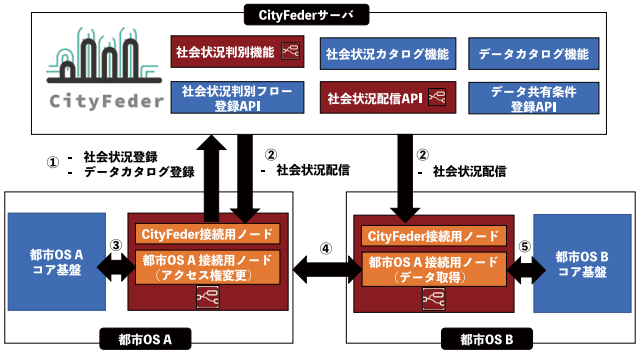

図 1　CityFeder のアーキテクチャ

各スマートシティ OS には 1）CityFeder および他スマートシティ OS とコミュニケーションをとるため，また 2）スマートシティ OS 内のデータ・機能に対するアクセス権を変更するための，Node-RED 環境が動作している．これらの機能は Node-RED のカスタムノードとして実装されており，その詳細は 4.2 節で詳しく述べる．上記の機能を利用し，CityFeder は下記の 4 段階の手順でデータの柔軟な相互流通を実現する．

3.3　CityFeder によるフェデレーションのプロセス

CityFeder によるプログラマブル・フェデレーションは，下記の手順に基づいて行われる．

ステップ 1: スマートシティ OS は，Node-RED フローによりモデリングされた社会状況の定義と，それに対応したデータのリストを CityFeder に登録する（図 1 中 1）

ステップ 2: CityFeder サーバは，登録された社会状況を監視し，その社会状況を認識した時点で，その社会状況を利用する各スマートシティ OS に伝達する（図 1 中 2）

ステップ 3: データを共有するスマートシティ OS は，Node-RED のカスタムノードを通じ，データへのアクセス権限を変更する（図 1 中 3）

ステップ 4: 他スマートシティ OS は，ステップ 3）で公開されたデータに対して Node-RED のカスタムノードを通じ，アクセスを行い（図 1 中 4），自身のスマートシティ OS 内で利用を行う（図 1 中 5）

CityFeder が上述したアーキテクチャ構成および手順でフェデレーションを行うこととした理由を，下記に示す．

CityFeder サーバの存在

(1) 一意な社会状況の認識・利用

異種スマートシティ基盤間で社会状況に応じたデータ共有の合意をとるためには，データ提供側，利用側双方が，共通した一意な社会状況を認識する必要がある．よって，CityFeder ではサーバサイドが社会状況を認識し，その状況を各スマートシティ OS に伝達を行うモデルとした．これにより，要件 1 を満たす．

(2) 社会状況・データカタログの集約

他スマートシティ OS がモデリングした社会状況を他スマートシティ OS が利用可能とした．これにより，社会状況を抽出するプログラム（実際は Node-RED フロー）の再利用性を高められる．また，どのスマートシティ OS がどういったデータを公開可能か，その際にどういう手順でアクセス可能かをカタログとして共有することで，様々なスマートシティ OS からデータアクセスを可能とする．

Node-RED の採用

(1) スマートシティ基盤の改変を必要としない

データ共有に対し，スマートシティ基盤自体に新たな API の実装を要求することは，多大なコストを要求することとなる．すでに多種多様なプロトコルに対する接続性を有したオープンソースソフトウェアの Node-RED を利用・カスタムノードとして必要な機能を実装することで，既存スマートシティ OS に対する変更を必要とすることなく，異なるプロトコル・フォーマットの差異を吸収したデータの相互利用を可能とする．これにより，要件 2 を満たす．

(2) 処理の見える化

Node-RED ではノードという小さな処理の単位をつなぎあわせ，フローとして一連の処理を記述する．本研究では，社会状況のモデリング自体を Node-RED で行うことで，そのフローを参照することによりどういった処理により社会状況がモデリングされているのか，第三者が理解することができる．よって，処理が見える化され，システム利用者間でのより深い相互理解が可能となる．

4.　実装

4.1　CityFeder サーバ

本章では，主に CityFeder サーバの機能について説明を行う．CityFeder サーバの機能は，1）社会状況およびその社会状況に対応した共有データカタログ機能，2）社会状況認識・配信機能，の大きく 2 点である．

4.1.1　社会状況およびその社会状況に対応した共有データカタログ機能

本機能は，各スマートシティ OS のユーザが利用できる社会状況リスト，社会状況に対応したデータリストの登録・閲覧を可能とする機能である．社会状況の定義は，任意の開発者が Node-RED のフローで行う．フローを記述する際の制約は，必ずそのフローの最後に true もしくは false を出力するノードを用意することである．フローの出力が true の場合，定義された社会状況が検出状態であることを示し，false の場合はそうでないことを示す．開発者によって定義されたフローの登録は，CityFeder サーバが

図 2　CityFeder WEB サーバ上で表示される登録済み社会状況と登録フォーム

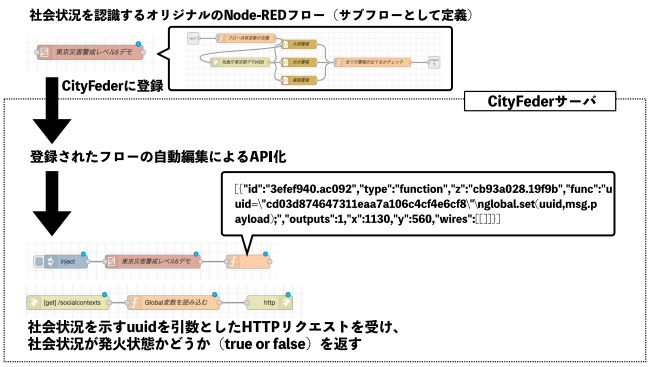

図 3　登録された社会状況フローの API 化

提供する WEB を通じて行う．図 2 に，登録された社会状況一覧のリストが表示されている様子と，社会状況の新規登録フォームを示す．新規登録を行う際は，社会状況の名前，説明，検索・分類のためのコンテキストタグ，社会状況に関連したエリアを代表する住所（簡易でも良い）と，社会状況を判別するための Node-RED フローを記載する．登録された社会状況はリストに表示され，それぞれのリストのリンクをクリックすると，登録された情報の詳細が表示される．表示内容は，上記で登録された内容に加え，CityFeder から配信される社会状況を各スマートシティ OS 基盤で動作する Node-RED 環境で受け取るためのノード定義が含まれる．この定義をコピーし，Node-RED 環境にインポートすることで，定義された社会状況が発火したタイミングで各スマートシティ OS の挙動の変化（アクセス権設定や他スマートシティ OS への接続）を行うフローを記述可能となる．

4.1.2　社会状況認識・伝達機能

CityFeder は登録された社会状況を抽出する Node-RED フローをサーバ内の Node-RED 環境にコピーし，社会状況の監視を行い，その社会状況が検出された際にその状況を各スマートシティ OS に伝達する機能を有する．本研究では登録された社会状況毎に HTTP のエンドポイントを自動で用意し，社会状況が配信された場合にはそのエンドポイントから true が返される仕様とした．これを，社会状況定義フローの API 化と呼ぶ．この仕様を実現するため，CityFeder は登録されたフローに対し，ユニークな

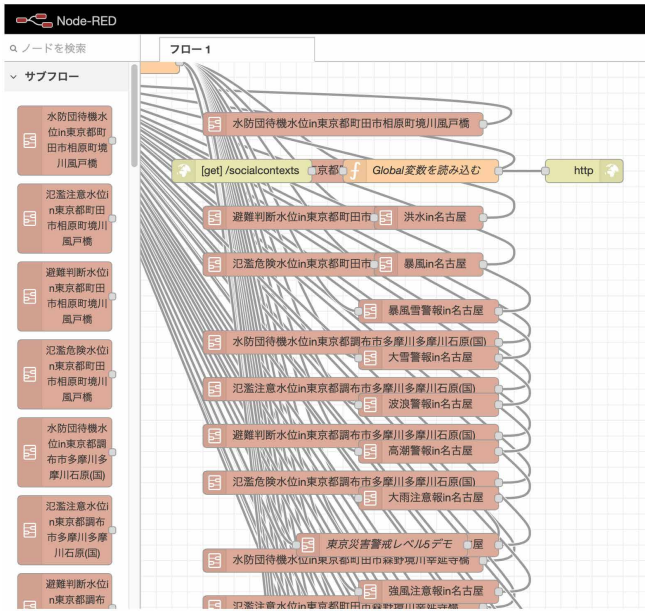

図 4　CityFeder サーバ内で API として動作する Node-RED フロー群

HTTP エンドポイントを設定するための uuid を付与し，CityFeder サーバの Node-RED 環境内の共有変数に社会状況の状態（true or false）を保存する．Node-RED 環境内には社会状況の問い合わせを受けるための HTTP リクエスト用フローが存在し，指定された uuid に紐付いた社会状況を true or false で返答し，社会状況を伝達する（3）．上述した処理に基づき，CityFeder サーバは登録された全ての社会状況フローを自動で変換し，社会状況の監視と配信を行っている．図 4 に，50 以上の社会状況が登録・API 化され，動作状態にある CityFeder サーバ内の Node-RED のパレットの様子を示す．

4.2　CityFeder および各スマートシティ基盤間の接続

本研究では，実際に相互接続を可能とするスマートシティ基盤として，名古屋大学が中心となり開発を進めている Synerex 基盤 [15]，慶應義塾大学が中心となり開発を進めている SOXFire 基盤 [16]，および欧州が中心となり開発を進めている FIWARE 基盤の 3 つの異なる基盤を対象とした実装を行った．3 つの基盤を，CityFeder サーバと接続し，受信した社会状況に基づいたスマートシティ基盤のアクセス権変更や，他スマートシティ基盤への接続しデータの受信・利用を可能とさせるため，Node-RED のカスタムノードを実装した．Node-RED 環境を通して CityFeder サーバおよび他スマートシティ基盤と接続することで，プロトコルやデータフォーマットの差異を吸収し，統一的なデータの取り扱いが可能となる．本研究では，3 つの基盤それぞれに対して，コネクション管理，データアクセス（GET および Subscribe 形式），データ送信（PUT および Publish 形式），アクセス権変更などのカスタムノードを実

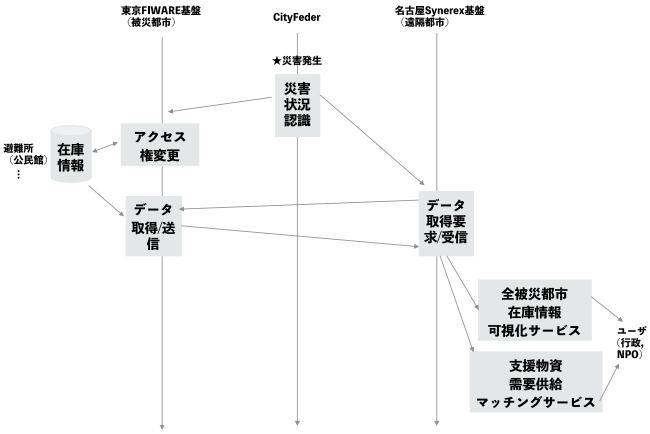

図 5　実証シナリオ

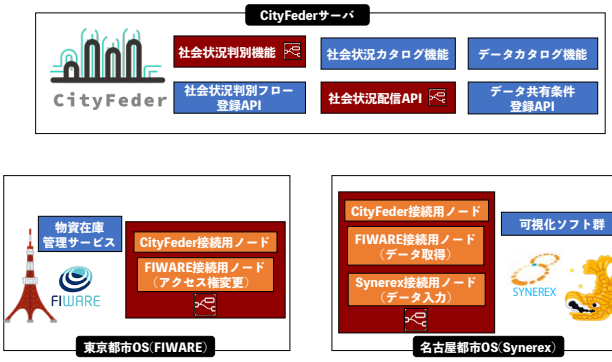

図 6　ユースケース実証のシステム構成

装した．Synerex，SOXFire，および FIWARE の対応したカスタムノードの実装は Github で公開している他，npm 経由で直接 Node-RED 環境からインストールが可能となっている．

5.　ユースケース実証

本研究では，2.3 節で述べた 2 つのユースケースのうち，特に 1 つ目のユースケースについて，CityFeder がその実現が可能であることの技術実証を行った．CityFeder は，ある社会状況が発生した際に，異なる複数のスマートシティ基盤を連携可能とさせるが，本ユースケースにおいては，災害警戒レベルが 5 の場合，遠隔自治体（名古屋）から被災自治体（東京）の避難所における物資情報を自動で取得可能とする，というシナリオとなっている．本シナリオのシーケンス図を図 5 に示す．CityFeder サーバにより災害の発生を認識次第，その状況を東京の FIWARE 基盤と，名古屋の Synerex 基盤に伝達する．東京 FIWARE 基盤では，在庫情報に対するアクセス権を変更し，名古屋ユーザからのアクセスを許可する．同時に，名古屋ユーザは FIWARE 基盤に対してデータ取得要求を送信し，FIWARE 基盤からデータが送信される．Synerex 基盤では，取得した情報を可視化する．このシナリオを技術的に達成することで，CityFeder の有効性を実証する．

本実証のシステム構成を図 6 に示す．本シナリオは，

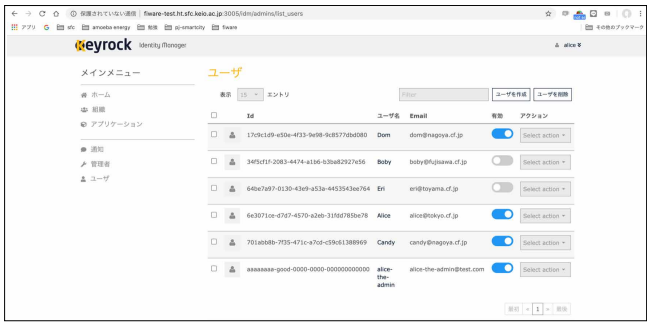

図 7　FIWARE 上のユーザ管理画面

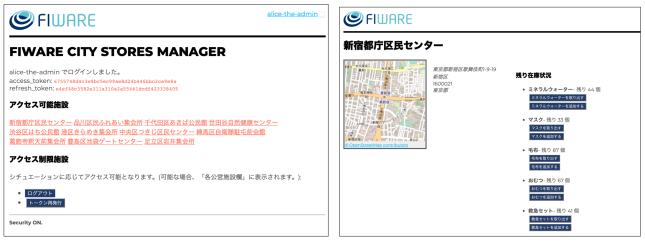

図 8　FIWARE 上で動作する在庫物資管理サービス

CityFeder サーバ，東京都市 OS（FIWARE），名古屋都市 OS（Synerex）の大きく 3 つのシステムが連携して実現される．CityFeder サーバは 2.4.1 で述べた機能を有しており，また東京都市 OS には避難所の在庫物資管理サービスが動作しており，名古屋都市 OS には Synerex に流れるデータの可視化ソフト群が動作している．また，東京都市 OS，名古屋都市 OS のそれぞれに対応した Node-RED 環境が動作している．

5.1　FIWARE で動作する在庫管理システム

CityFeder の汎用性を示すため，提案者ら以外によって開発される FIWARE を東京都市 OS と模して，本シナリオの技術実証を行う．FIWARE は欧州を中心に開発が進むスマートシティ基盤であり，同基盤を対象としうることを示すことは，CityFeder の汎用性を示す上でも重要であると考えた．本実証では，FIWARE 上で動作するアプリケーションおよびデータへのアクセス権を動的に変更させる機能を CityFeder/Node-RED で実現する．まず，FIWARE サーバ上の WEB ページでデータおよびアプリケーションへのアクセス制御を行うためのユーザ登録を行う．そして，社会状況に応じて組織単位でデータへのアクセス制御を行うため，各ユーザを該当の組織に登録する．データの公開対象外のユーザを無効化することでデータへのアクセスを禁止することができる．そして，社会状況に応じて，組織単位でユーザの有効化を行う（図 7）ことで，アクセス制御を実現する．これらのアクセス制御を，Node-RED 環境を介して行えるカスタムノードにより自動化が行われる．この FIWARE 環境では，東京の避難所ごとに Web ページが用意しており，それぞれのページには保有される物資の

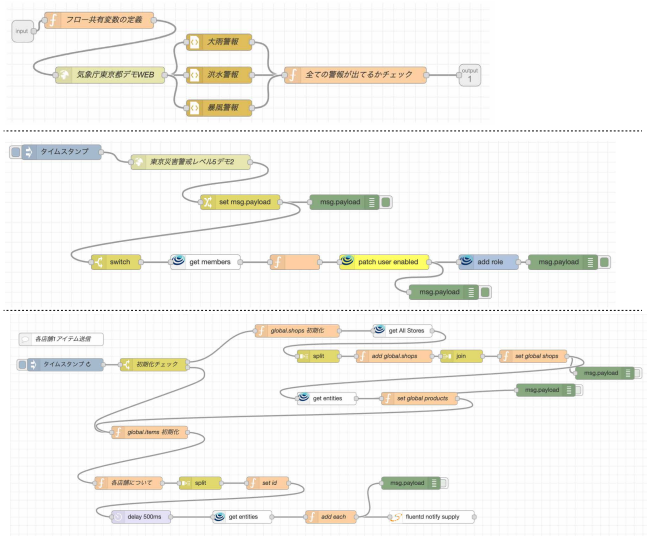

図 9　実証シナリオを構成する Node-RED フロー（上段：CityFeder サーバ内で動作，中段：FIWARE サーバ側で動作，下段：Synerex サーバ側で動作）

数が表示されている（図 8）.（※なお，これらの避難所は今回の技術実証のために仮想的に設定したもので，実際には存在しない避難所であるため注意されたい.）通常，これらの Web ページおよび物資データにアクセス可能なのは限られた東京都市 OS のユーザのみであるが，本実証では災害警戒レベル 5 が発生したタイミングで名古屋都市 OS のユーザがアクセス可能となるように，予め CityFeder にポリシーが登録されることとなる.

5.2　プログラマブル・フェデレーションのプロセス

2.3 で示したように，CityFeder は 4 つのステップでプログラマブル・フェデレーションを実現する．本実証において，シナリオを実現するためにその 4 つのステップをどう経ているか，下記に述べる.

5.2.1　ステップ 1 ：災害警戒レベル 5 の検出

災害警戒レベル 5 を認識するフローを作成し（東京都市 OS 開発者もしくは社会状況をモデリングする任意の開発者），CityFeder サーバに登録する．本実証では，災害警戒レベル 5 を気象庁の警報・注意報掲載 Web ページを参照することで認識する．任意のタイミングでテストが行えるようにするため，実際には気象庁の Web ページを模した Web ページ を用意した．この Web ページはボタンをクリックすることで，大雨・洪水・暴風警報を表示することができるため，Node-RED 側でこの Web ページを監視し，3 つの警報が全て表示されれば災害警戒レベル 5 が発生した，と判断するようにした．図 9 の上段に，フローの定義を示す.

5.2.2　ステップ 2 ：検出された社会状況の伝達

ステップ 1 で定義した社会状況（災害警戒レベル 5）を検知し，各都市 OS に伝えるため，登録されたフローを API

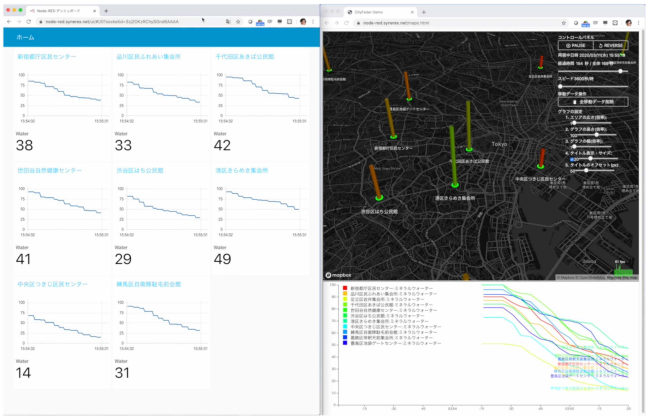

図 10　在庫物資の可視化

化し，動作させておく．また，事前に名古屋都市 OS ユーザと東京都市 OS ユーザ間で，この社会状況が発生した際に物資情報に関するアクセスが許可されることを相互に合意しておく．そのため，予め名古屋都市 OS ユーザのアカウントを，東京都市 OS（FIWARE 環境）内に作成しておく必要がある．最も，データを誰にでも公開する場合は匿名アカウントや無記名ユーザでのアクセスで良いため，その場合はアカウント登録は必要ないことが想定される.

5.2.3　ステップ 3 ：アクセス権の変更

東京都市 OS は，災害警戒レベル 5 の状況が発生した場合，東京都市 OS 内の物資データへのアクセスを名古屋都市 OS ユーザへ許可するフローを事前に構築しておく．図 9 の中段に，実際のフローを示す.

5.2.4　ステップ 4 ：データの取得と利用

データアクセスに対するアクセス権が許可された際，名古屋都市 OS は Node-RED 環境内に提供された FIWARE 接続ノードを利用し，東京都市 OS のデータにアクセスし，物資データを取得，Synerex 基盤に送信するとともに，可視化する．具体的には，FIWARE 基盤が提供する Orion Context Broker からデータを受信し，Synerex 内のデータ送受信用チャネルにデータ Publish を行い（図 9 下段のフローにより，実現），可視化サービス Harmoware-VIS で表示すると同時に，ダッシュボードやグラフで表示を行う（図 10）.

以上の 4 ステップにより，事前の社会状況モデリング，それに基づく CityFeder サーバによる状況検出・伝達，東京都市 OS 内データへの動的なアクセス権変更，名古屋都市 OS 側のデータの受信・可視化，が達成され，本研究で対象とするシナリオが技術的に実証可能であることが確認された.

6.　まとめ

本研究では，社会状況に応じた異種スマートシティ間の動的なデータ・機能流通を実現するプログラマブル・フェデレーションの概念を新たに提案し，その実現を行った.

本研究においてプログラマブル・フェデレーションは，社会状況を認識・伝達するとともに，データのカタログ及びアクセス方法が記載されたカタログ機能を有する CityFeder サーバと，各スマートシティに付随する Node-RED 環境に提供される基盤接続のためのカスタムノードを利用した設計・実装が行われた．また，データを様々な観点で理解するための複数の可視化機構を構築し，それらを利用した災害時物流支援サービスのシナリオに適用した．結果，CityFeder が考案したシナリオを技術的に実現可能とすることを示した．これにより，図 43 に示す定量的な評価を達成し，本研究の当初の目的を超える成果を得たと結論づけられると考える．

一方，本研究で構築した CityFeder を社会実装し，実用化していくためには，少なくとも下記の課題に取り組む必要がある．

- CityFeder サーバ運用主体の明確化
  CityFeder は中央集権的な構造で，一意な社会状況を安定して監視し，配信する設計となっている．この機能を中央集権的に運用することは，社会状況の共通理解と利用のためには必要であるが，どういった組織が実際に運用するか，その信頼性をどう担保するか，は今後具体的な議論が望まれる．
- 様々な都市 OS を対象とした Node-RED 環境用接続ノードの実装
  本研究では Synerex 基盤，SOXFire 基盤，FIWARE 基盤の 3 つの基盤に対する接続ノードの実装を行ったが，スマートシティ基盤の候補は他にも様々存在するため，それらに対する接続アダプタをどう揃えていくか，は今後の課題である．一方，本研究では Node-RED をミドルウェアとして用いているため，豊富な既存資産を有効活用できると考えている．この点では，Node-RED コミュニティとの連携を推進していくことが重要であると考えられる．

謝辞

本研究の一部は，内閣府総合科学技術・イノベーション会議の「戦略的イノベーション創造プログラム（SIP）第 2 期／ビッグデータ・AI を活用したサイバー空間基盤技術」（管理法人：NEDO），総務省 SCOPE，科研費基盤研究 C（19K11945）の支援を受けたものです．

参考文献

[1] Raffaele Giaffreda. *iCore: A Cognitive Management Framework for the Internet of Things*, pages 350–352. Springer Berlin Heidelberg, 2013.
[2] Internet of things - architecture. `http://www.iot-a.eu`.
[3] Citysdk. `http://www.citysdk.eu`.
[4] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth, et al. Spitfire: toward a semantic web of things. *Communications Magazine, IEEE*, 49(11):40–48, 2011.
[5] Luis Sanchez, Luis Mu noz, Jose Antonio Galache, Pablo Sotres, Juan R. Santana, Veronica Gutierrez, Rajiv Ramdhany, Alex Gluhak, Srdjan Krco, Evangelos Theodoridis, and Dennis Pfisterer. Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*, 61:217 – 238, 2014. Special issue on Future Internet Testbeds.
[6] The Open Source Platform for Our Smart Digital Future FIWARE. `https://www.fiware.org`.
[7] CoAP. `http://coap.technology`.
[8] MQTT. `http://mqtt.org`.
[9] ZeroMQ. `https://zeromq.org`.
[10] N. Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7, 2017.
[11] Jae-young Ahn, Jun Seob Lee, Hyoung Jun Kim, and Dae Joon Hwang. Smart city interoperability framework based on city infrastructure model and service prioritization. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 337–342, 2016.
[12] Eclipse sensiNact. `https://projects.eclipse.org/projects/technology.sensinact`.
[13] Ryong Lee and Kazutoshi Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, LBSN ’10, page 1–10, New York, NY, USA, 2010. Association for Computing Machinery.
[14] 下坂正倫, 早川裕太, and 坪内孝太. 携帯端末位置履歴を用いた階層ディリクレ混合回帰モデルに基づく活動人口予測. In **情報処理学会研究報告ユビキタスコンピューティングシステム**, number 15, nov 2018.
[15] 河口信夫, 米澤拓郎, and 廣井慧. Synerex: 超スマート社会を支える需給交換プラットフォームの設計コンセプトと機能. In **情報処理学会研究報告ユビキタスコンピューティングシステム**, number 49, feb 2020.
[16] Takuro Yonezawa, Tomotaka Ito, Jin Nakazawa, and Hideyuki Tokuda. Soxfire: A universal sensor network system for sharing social big sensor data in smart cities. In *Proceedings of the 2nd International Workshop on Smart*, SmartCities ’16, New York, NY, USA, 2016. Association for Computing Machinery.