木構造を用いた識別子の動的な割当によるセンサーモジュールの一意識別

高木 優希[2,a)]　串田 高幸[2]

概要：近年, IoT の発展と共にその管理について注目されている. 特にデータ送信を行うセンサーモジュール (以降「センサー」と表記) の管理は物理的位置やデータ種別情報が必要となる. しかし, データのみを送信するセンサーのような識別情報を持たないノードはネットワーク上で認識が不可能である. そこで, 識別子を持たないセンサーに対して仮想的な識別子の付与を行い, 遠隔監視を実現する. 識別子管理を動的にするため, DB による各コンピュータボード情報およびセンサー識別子の管理を木構造で行う. これによって各ノードは 1 つ下位の識別子のみを管理するだけでよく, 最上位ノードが組織として管理できる識別情報が既存方式よりも格段に向上する. センサーの識別子は I2C 規格を基準とした 256 個を単一のコンピュータボードで扱える数として, 線形探索による割当済の管理を行う. その後, UUID を含む他の識別子との比較を行い, 識別精度や簡易性, セキュリティ, 管理の透過性に関する評価および議論を行う. この試みはコンピュータボードを用いた学習環境の整備や開発による IoT 利用可能分野の拡張を実現する.

1. はじめに

近年, IoT は発展し世界的な広がりを見せている [1]. Ericsson Mobility Report によれば 2019 年には 9 億デバイスが存在し, 2025 年には 24.9 億でデバイスになると予想されている [2]. 本レポートで対象とするものは short-range に関するものであるが, 19.5 億個になると予想されており, この膨大な数の IoT デバイスを管理する方法の確立は急務である. 日本国内においても IoT は発展をみせ, 自動車の自動運転といったモビリティ領域や, 都市や住宅をカバーするスマートシティ・スマートハウス領域, 健康的な生活を目指すウェルネス領域が注目されている [3]. これらは単一の領域内で完結せず, 相互に関係しあうことによって社会全体への効果が期待される. このように分野が異なっていても相互に関係しあうことを前提とすることで, これらすべてを一括で管理・運用可能にすることで, IoT は国内外問わず発展すると考えられる. そこで, 本研究による識別可能数の増加とプラットフォーム依存度の低下によって, IoT の包括的な管理を実現する.

1.1 背景

IoT は多くの分野に適応ができる便利なものだが, 前述したようにデバイス数が今後増加を続けていくことによって, IoT デバイスによって生成される大量のデータの保存, 安全なアクセス, データ処理の方法といった課題が発生する. Ciro Formisano らは 2 つの技術を組み合わせることで, これらの課題が解決できるとしている [4]. 1 つは, IoT 技術によるネットワーク内のオブジェクトを一意に識別することによってオブジェクトの相互接続を可能にすることである. もう 1 つは, クラウド技術による豊富なリソース上でアプリケーションを処理するための柔軟な仮想実行環境と, 使用状況によるスケールアウトおよびスケールダウンの提供である. これらによってセンサーやアクチュエータから情報を簡単に利用でき, 複雑なアクションが実行可能となる. また, この 2 つの技術を組み合わせることによって, システム設置のコスト削減と柔軟性といった経済的側面と, スマートシティの確立という社会的側面が実現されると述べている. 本研究では前者の IoT 技術によるオブジェクトの一意識別に関してより焦点をあて, IoT で用いられるセンサーに対して新たな識別子の提案を行う. この研究によって, 図 1 にようなビニールハウスの状態監視をはじめとする複数の位置に複数のセンサーを配置しなければならないような環境下で真価を発揮することが可能となる. 図 1 では, ビニールハウスの各棟に 4 つずつセンサーを配置し, 遠隔監視の状態を Web 上から確認した想定図を

[1] Tokyo University of Technology
Cloud and Distributed Systems Laboratory, Japan
[2] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[a)] C0117178

表している．

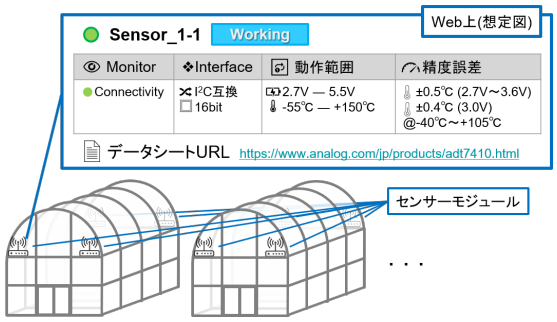

図 1　ユースケース図 (ビニールハウス)

1.2　課題

IoT を管理する上で最も重要なことは, オブジェクトが一意に識別ができるということである [4][5][6]. センサーには識別子を元から保持しているとそうでないものが存在している. 次の章で詳しく説明をするが, 識別子を新たに付与するアプローチでは, オブジェクト識別子やクライアント ID を付与する, あるいはリソースの形式やデバイス ID を用いるといった解決方法が存在している. しかし, 新たに付与した識別子や元から識別情報を保持しているものの多くが提供元企業のデバイスやプラットフォームに依存しており, 相互に提供することや連携することが困難である. また, これらの多くが IPv4 同様の 43 億以上の膨大な数の想定がされておらず, 今後も IoT デバイス数が増加することを考慮すると, 識別可能上限数の上昇も必要である. これらから, 依存度が低いオブジェクト識別子をベースとして, 木構造による識別子情報の管理を行う.

1.3　各章の説明

この章では研究の背景や課題について具体例を交えて言及した. また, 課題解決によって得られる恩恵についても触れた. 2 章では本研究に関連した先行研究について触れる. それぞれの研究の紹介と本研究との関連性について, 同質の点や未解決の課題について言及する. 3 章では本研究の提案について述べる. モデル図およびテーブル情報やデータフロー図を示しながら説明を行う. 4 章では, 実装について, 具体的なソフトウェア構成や作成したプログラムを交えて説明を行う. 5 章では評価について述べる. 実験環境と評価項目および今後期待する結果について説明を行う. 6 章ではこれまでの章を元に議論を行う. 7 章では今後の研究の方向性と展望について述べる.

2.　関連研究

この章では本研究と関連した先行研究について取り上げ, その関連性について述べる.

先述したように, IoT は農業での利活用の試みは既に行われている. Mahammad らはバッグボーンとしてクラウドコンピューティングを使用した農業 IoT のセンサー監視について調査した. また, この論文においても一意の ID が必要になると言及されている [7]. 現状では IoT は多くのノードで構成されることが想定されており, 本研究においても合計で 43 億以上の IoT デバイスが可能ということを想定としている. IPv4 プロトコルでは, 4 バイトのアドレスを通じて各ノードを識別する方法があるが, IPv4 のアドレスが枯渇した今では上限数に問題があるといえる. また, IPv6 プロトコルを用いた方法も考えられているが, 適切性はアーキテクチャに依存するため注意が必要と述べられている.

識別子の付与やノードの一意識別は先行研究においても課題として挙げられている. Jahoon らは, IoT プラットフォームで動作するデバイス ID の相互運用性に関する分析を行った [5]. オブジェクト識別子をベースとする oneM2M および GS1 Oliot やクライアント ID ベースの IBM Watson IoT, リソースの形式やデバイス ID をベースとする OCF IoTivity について, フォーマットやメタデータが言及されている. ここでのオブジェクト識別子とは, Abstract Syntax Notation One(ASN.1) で指定されたツリー構造を持つものを指す. 本研究で扱うものに関してもオブジェクト識別子ベースに類似するため, oneM2M および GS1 Oliot との比較も評価で行う. この論文で述べられている様に, プラットフォームに依存した識別子も存在するため, 識別 ID を統合する仕組みも必要となってくると考えられる.

次章で詳しい説明を行うが, 本研究では識別子の管理として木構造モデルを採用している. Huansheng らは本研究と同様の IoT における非 ID オブジェクトのツリーコードモデリングとアドレス管理について定義と主な重要性について述べた [6]. しかしながら, 異種センシングデータの統合やアドレッシング時間の制御, リアルタイム性といった課題が残っていると述べている. 本研究ではプラットフォーム依存を解決するため, この論文で述べているモデリングやアドレス指定技術とさせることで, より汎用性の高さを維持しつつ識別可能上限数の多い IoT システムの実装が可能になる.

3.　提案

この章では, 本研究の提案内容について詳細に述べる. 本研究は, 各ノードに対して新規に生成した識別子を付与し, それらを木構造として各上位ノードで管理することで, ノードの識別可能数を IPv4 の上限である 43 億以上に拡張することと, 既存方式や先行研究での他の識別子との連携を可能にすることでプラットフォーム依存の低下を実現することが目的である.

まず, ネットワーク内で全てのセンサーノードが一意に識別可能とするために, 各ノードに対して識別子を付与す

る．この時，遠隔監視を行うサーバーを上位ノードとし，コンピュータボードを中間ノード，センサーを下位ノードとした木構造モデルを提案する．また，組織の管理用として，最上位ノードを設置する．これらのモデル図を図 2 に示す．

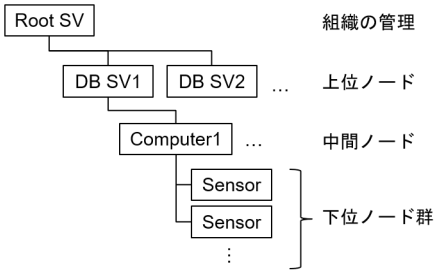

図 2　木構造による識別子管理のモデル図

中間ノードは下位ノードの識別子の付与と管理を行い，上位ノードは中間ノードの識別子の付与と管理を行う．最上位ノードは上位ノードから下位ノードまでを 1 つの組織として管理し，組織の統合や解除を行う．各ノードの動きをまとめたものを図 3 に示す．

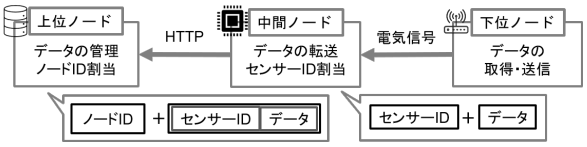

図 3　各ノードの役割

下位ノードは一般にセンサーであるため，データの取得および送信を行う．中間ノードでは受信したデータに下位ノード識別子であるセンサー ID を付与して上位ノードへ転送する．上位ノードは受信したデータと中間ノードに割り当てたノード ID を紐づけて保存する．つまり，センサーから取得したデータは中間ノードおよび上位ノードへと 1 つ上のノードへ転送されるたびに識別情報が追加付与されることによって多くの下位ノードを管理することが可能となる．上位ノードは中間ノードのみを管理し，中間ノードは下位ノードのみを管理するため，1 つ上のノードや 1 つ以上下のノードを考慮する必要がない．これによって 43 億以上の下位ノードを管理することが可能となる．最上位ノードの動きを図 4 に示す．

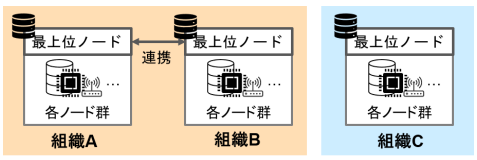

図 4　識別子テーブルのイメージ図

最上位ノードでは，基本的には組織内の管理として，各上位ノードとの疎通確認を行う．しかしながら，組織を統合させる，あるいは分割したい場合に最上位ノード同士による連携を行い，仮想的なグループを作成する．図 4 では，組織 A と組織 B が連携して橙色のグループとなり，組織 C はどの組織とも連携していないため，組織 C のみの青色グループである．ここでの連携とは，各組織の上位ノードに対して相互にアクセスができる状態のことを指し，中間ノードおよび下位ノードは除外される．この目的は，組織 A を管理するユーザが組織 B の監視を行う場合やその逆を行う場合に用いるためである．

次に，データの扱い方について説明を行う．上位ノード及び中間ノードにデータベースを構築し，それぞれの識別情報や中間ノードおよび下位ノードの情報を格納する．各テーブルのイメージ図を図 5 に示す．

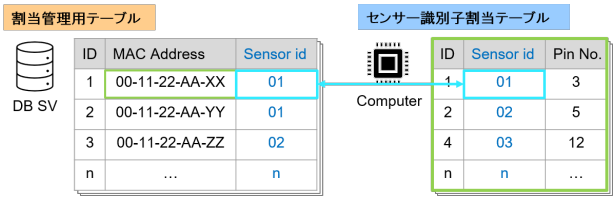

図 5　識別子テーブルのイメージ図

中間ノードでは，センサー情報を一定間隔ごとに上位ノードに送信するエージェントソフトウェアを作成し，配置する．これによって，中間ノードのテーブルの更新に伴って上位ノードのテーブルも更新が可能となる．また，中間ノードおよび上位ノードは識別子用のテーブルだけでなく，センサー情報に関しても保存を行い，図 1 の Web 上表記のような遠隔監視を実現する．

これらによって，先述した通り莫大な数のセンサーを動的に管理し，運用することが可能となる．図 1 のビニールハウスを例に挙げれば，複数のビニールハウスに対して複数のセンサーを複数箇所配置し，状態を監視しているとする．1 つのセンサーから異常値を検知した場合やデータ送信がなかった場合，そのセンサーを特定し対処しなければならない．しかし，識別情報を持たないセンサーではどのビニールハウスのどの区画に配置したどのセンサーかということが分からないのである．この課題を本研究が解決することで，IoT の適用分野がより広がると同時に IoT 管理に対して貢献ができると考えられる．

4.　実装について

この章では，3 章を元に具体的な実装について言及する．まず，最終的に遠隔監視を実現するため，識別子の情報だけでなくコンピュータボードのマシン情報やセンサーのデータシート情報が必要となる．図 1 における必要となるデータシート情報としては，センサーの使用可能な規格や動作範囲，精度の誤差である．マシン情報の自動的な取得および格納を実現するため，図 5 の他にセンサー情報を格納する

テーブルを作成する．センサー情報管理テーブルのイメージ図を図 6 に示す．

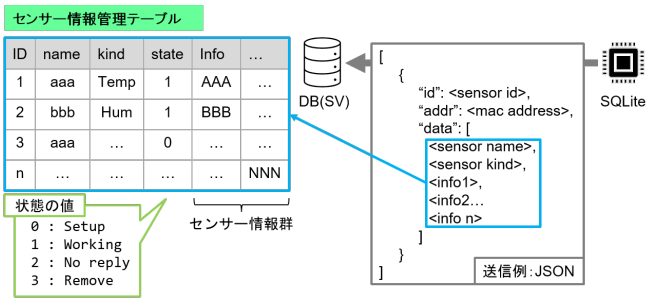

図 6　センサー情報管理テーブルのイメージ図

図 6 では，例として JSON 形式でコンピュータボードからデータを送信するものを挙げた．送信するデータは事前にコンピュータボード上の SQLite データベースに保存しておく．センサーのデータシートの情報はセンサー接続時に，その他の情報は一定間隔でサーバーへ送信される．テーブルでは，センサーに付与した ID をプライマリーキーとして，センサーの種類やデータシートの情報およびセンサーの状態を保存する．センサーの状態は Setup, Monitoring, No Reply, Remove の 4 種類である．この 4 種類に関する説明を表 1 に示す．

表 1　ステータス一覧表

ステータス	説明
Setup	初期セットアップ/センサー情報の参照
Working	正常動作中/定期的なデータ送信
No Reply	応答なし/中間ノードと物理的接続有り， 上位ノードへデータ送信無し
Remove	センサー取り外し/中間ノードとの物理的接続無し

また，それぞれの状態に対する遷移図を図 7 に示す．

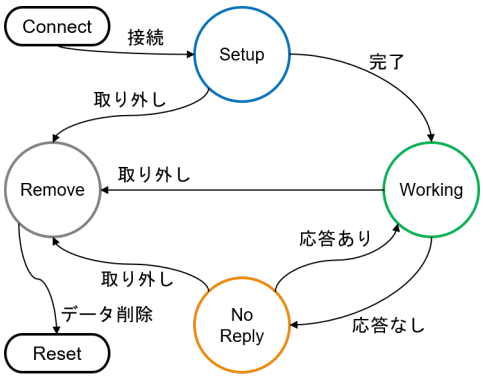

図 7　ステータスの状態遷移図

センサーが接続された際を図 7 の Connect とした時，センサー情報を SQLite DB に格納する状態を Setup，正常動作時を Working，センサーが物理的に接続されているがデータの送信がない場合を No Reply，センサーが取り外された場合を Remove としている．状態が Remove になった際，全てのデータベースからセンサーの全情報を再帰的に削除する．これを Reset としている．基本的な動作としては，応答の有無による Working と No Reply の状態が繰り返し変化すると考えられる．

次に，センサーの識別子の割当に関する説明を行う．I2C のアドレス情報を基準とし，0x00 から 0xff の間の 256 個の範囲内で順に割り当てを行う．割り当ての方法を疑似コードとして表し，図 8 に示す．

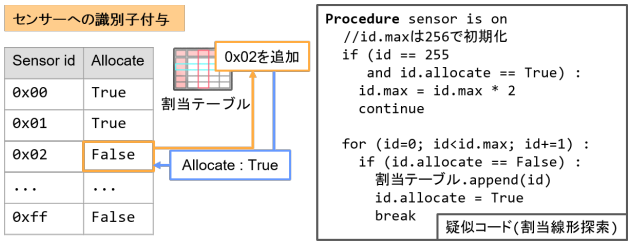

図 8　疑似コードとその流れのイメージ図

識別子が割り当て済みかどうかは線形探索によって判断を行う．図 8 のテーブルのカラムである Allocate が False のものを線形探索によって探し，割り当てを行う．もし，すべての識別子が割り当てられた場合は，識別子テーブルを拡張する．

5.　評価について

本研究では各ノードに対して識別子を付与し木構造による管理を行っている．そのため，識別精度や識別可能範囲，他の識別子との特徴比較を行う予定である．また，最上位ノードの動作確認を行う予定である．

5.1　実験環境

本研究で用いた機器を表 2 に示す．

表 2　使用機器表

ノード種別	表記名	使用 OS/使用機材
最上位ノード	Root SV	Ubuntu20.04LTS
上位ノード	DB SV	Ubuntu18.04LTS + MySQL
中間ノード	Computer	Raspbian4.19 + SQLite
下位ノード	Sensor	ADT7410/DHT11/BMP180

本研究では，フリーの Linux ディストリビューション 1 つである Ubuntu を主に使用し，中間ノードでは Raspberry Pi，下位ノードでは上記のセンサーを使用した．中間ノードのデータ取得を行うプログラムおよびエージェントソフトウェアは Python3 によって記述している．

5.2 評価方法

評価項目として以下の 4 点を実施予定である.

(1) 下位ノードを上位ノードから監視した際の識別精度
(2) 識別可能範囲の検証
(3) 既存方式・先行研究における他の識別子と本研究との特徴比較
(4) 最上位ノードを用いた組織の統合および分割の検証

(1) では, 図 1 のような遠隔監視が実現可能かどうか, さらに下位ノードの識別が正常に行われているか中間ノードとの比較検証を行い評価を行う. (2) では, 実際の環境下でシミュレーションを行い検証を行う. 理論値として, MySQL の INT 型を符号なしで 0 から 4294967295 の 4294967296 個扱うことができ, センサー ID の 256 種類と乗算し, 1 兆以上の識別が可能ということである. しかしながら, 下位ノードを莫大な数用意することは非現実的であるため, 乱数によって生成されるデータが下位ノードから送られてきたデータだと仮定し, 中間ノードと上位ノードの動作を確認する. これによって実際に扱うことのできる識別可能上限数を調査し, その結果を評価する. (3) では, 既存方式や先行研究で提案されている識別情報と本研究で扱う識別子の違いを評価する. また, 本研究の識別子と他の識別情報を連携させる方法について検証および結果に対する評価を行う. (4) では, 最上位ノードによる組織間の統合・分割処理が正常に動作するかを検証する. また, 統合後および分割後の (1) と (2) の動作についても検証する. これらの結果について評価を行う.

6. 議論

この章では, これまでの章についての議論を述べる. 本研究の提案部分では, 現状ノード ID に MAC アドレスを用いており, ネットワーク外のノードに対して識別が難しくなる点や機器によっては MAC アドレスが変更できるものもあるため, より強力な識別子が必要となる. そのため, 一般に世界的規模で扱うことのできる UUID との比較検証を行うと同時に, 新たな識別子の生成方法・付与方法を考慮する必要がある. また, 2 章にもあったように IPv6 プロトコルを用いた方法も存在している. しかし, アーキテクチャによっては使用が困難な場合もあり, IPv6 プロトコルが本研究に対して有効なものかどうか, 有効であれば他の識別情報との違いが何かを調査および検証する必要がある.

7. おわりに

最後に, 本レポートのまとめを述べる. 本研究では, 2 つの課題に対して着目を行った. 1 つは, 識別情報を持つノードと識別情報を持たないノードが存在し, これらの一括管理を行うことが難しいということである. もう 1 つは, 識別情報を付与する際に, アプリケーションやプラットフォームに依存しやすいという点である. これら 2 つの課題に対して, ノードの識別情報の有無に関わらず動的な管理を行うことで解決を行う. 既存方式や先行研究と比較して, 識別精度を維持しつつ, 識別可能上限数の拡張とプラットフォーム依存の低下を実現する. これらを実現するために, 各ノードが 1 つ下のノードの識別子を管理する木構造のアーキテクチャを採用した. 本研究によって, IoT 管理が可能な分野の範囲が格段に広がると同時に, 発展と増加を続ける膨大な数の IoT デバイスの管理が容易となる.

参考文献

[1] Novo, O.: Blockchain Meets IoT : A Architecture for Scalable Access Management in IoT, *IEEE Internet of Things Journal*, Vol. 5, No. 2, pp. 1184–1195 (2018).
[2] Naidoo, N., der Westhuizen, P. V., Lartey, E. and Lee, J.: Ericsson Mobility Report, Technical report, Ericsson (2019).
[3] of Internal Affairs, M. and Communications: *Information and Communications in Japan* (2019).
[4] Formisano, C., Pavia, D., Gurgen, L., Yonezawa, T., Galache, J. A., Doguchi, K. and Matranga, I.: The Advantages of IoT and Cloud Applied to Smart Cities, *2015 3rd International Conference on Future Internet of Things and Cloud* (2015).
[5] Koo, J. and Kim, Y.-G.: Interoperability of device identification in heterogeneous IoT platforms, *2017 13th International Computer Engineering Conference (ICENCO)* (2017).
[6] Ning, H., Yang Fu, S. H. and Liu, H.: Tree-Code modeling and addressing for non-ID physical objects in the Internet of Things, *Telecommunication Systems* (2014).
[7] : A Survey: Smart agriculture IoT with cloud computing, *2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)* (2017).