マルチホップネットワークにおける送受信の同期による IoT デバイスの省電力化

河竹 純一[1]　杉本 一彦[1]　串田 高幸[1]

概要：マルチホップネットワークを使用してセンサデータをサーバに送信するとき，IoT デバイスは他の IoT デバイスのセンサデータを中継する役割を持つ．IoT デバイスの消費電力を削減するためには中継機能の起動時間を減らす必要があるが，IoT デバイスによってサーバまでのホップ数が異なるために中継機能の停止ができないという課題がある．理由は，センサデータを同時刻に送信する場合であってもそのセンサデータを中継する IoT デバイスに届くまでの時間が異なるためである．提案方式では，IoT デバイスのサーバまでのホップ数，内部処理時間，通信時間の 3 つの値を使って送信タイミングをサーバで算出する．送信タイミングをサーバまでのホップ数ごとに設定することにより，中継機能の起動時間を限定し，消費電力を削減する．提案手法を評価する実験として，2 台の ESP32 を ESP-NOW で無線接続し，16bytes のセンサデータ 100 個を 10 秒に 1 個の頻度でサーバに送信した．パケット到達率を調べた結果，33%であった．

1. はじめに

背景

センサを搭載した IoT デバイスがセンサから取得したデータ（以下，センサデータ）をサーバに送信する手法の一つとしてマルチホップネットワークがある．マルチホップネットワークを使用する例として，駐車場で自動的に車両を検出してユーザに空き状況を提示するスマートパーキングがある [1]．マルチホップネットワークにおいて IoT デバイスは一般的にバッテリで駆動している [2–4]．各 IoT デバイスは隣接する IoT デバイスのセンサデータを中継する役割があり，IP ネットワークに接続されたゲートウェイを通じてセンサデータを定期的にサーバに送信している．

マルチホップネットワークに関する研究では，バッテリで得られる電力に制限があることからネットワークにおけるノードの消費電力を削減することが必要とされている [5]．IoT デバイスの消費電力を削減することによって得られるメリットとしては，バッテリを交換する頻度の低減や長期的な運用が挙げられる [6, 7]．例えば出力電圧が 5V で容量が 10000mAh のバッテリで 1 つの IoT デバイスを 6 ヶ月間動作させるためには，10000mAh×5V/180 日 ≃278mWh/日に抑える必要がある．

[1] 東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻
〒 192–0982 東京都八王子市片倉町 1404-1

課題

マルチホップネットワークにおけるセンサデータの中継は，センサデータの送信，中継，受信の 3 つの処理と通信処理に分けられる．図 1 にセンサデータの中継の流れとそのときの処理を示す．図 1 では IoT デバイス A がセンサデータを取得したのちに IoT デバイス B を中継して IoT デバイス C にセンサデータを送信しており，それぞれ送信，中継，受信の役割を担っている．IoT デバイス A でのセンサデータの取得時から IoT デバイス C で受信が完了するまでの処理は，IoT デバイス A，IoT デバイス B，IoT

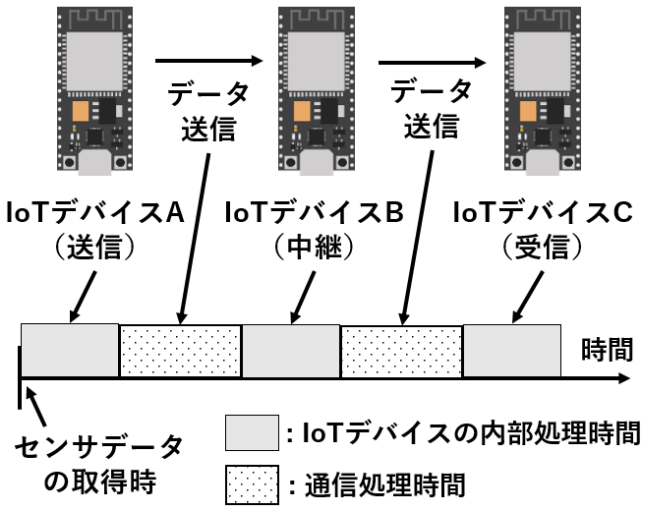

図 1　センサデータの中継の流れと処理

デバイス C の内部処理とデータ送信時の通信処理が交互に繰り返される．

サーバまでのホップ数が異なる複数の IoT デバイスが同時刻にセンサデータを送信すると，センサデータを中継する IoT デバイスにおいて受信するタイミングが一致しない．このことから，受信するタイミングが一致しない場合に中継機能の停止ができないため消費電力が削減できないという課題がある．ESP32 ではネットワークモジュールを起動するときの消費電力は停止しているときの消費電力と比べて 2.5 倍である [8]．中継機能を起動する時間が長くなると IoT デバイスの稼働時間が短くなるため，バッテリを交換する頻度が増える．

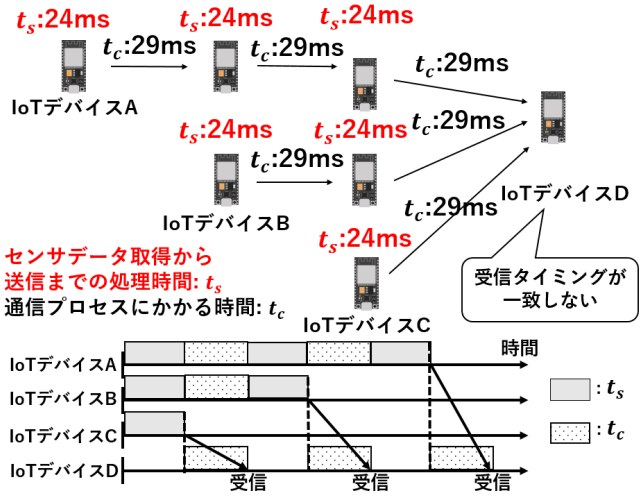

図 2 ホップ数の違いによりセンサデータの受信タイミングが一致しないときの例

ホップ数の違いによりセンサデータの受信タイミングが一致しないときの例を図 2 示す．IoT デバイス A，IoT デバイス B，IoT デバイス C はセンサデータを定期的に送信し，IoT デバイス D にマルチホップネットワークによって送信されるものとする．また，IoT デバイス A，IoT デバイス B，IoT デバイス C のホップ数はそれぞれ 2 ホップ，1 ホップ，0 ホップ（直接送信）であるとする．このときホップ数の数に比例して内部処理時間と通信時間が増加するため，IoT デバイス D における受信タイミングが一致しない．次に，具体的な内部処理時間と通信時間について図 3 と図 4 に示す．図 3 と図 4 は IoT デバイスとして ESP32 を用いたときの内部処理時間（t_s）と通信時間（t_c）について 1000 回計測したときの頻度分布である．この計測において最頻値を算出した結果，t_s が 24ms，t_c が 29ms であった．これを例として t_s を 24ms，t_c を 29ms とすると，図 2 における IoT デバイス D の受信時刻は，IoT デバイス A のセンサデータが $24ms \times 3 + 29ms \times 3 = 159ms$ 後，IoT デバイス B のセンサデータが $24ms \times 2 + 29ms \times 2 = 106ms$ 後，IoT デバイス C のセンサデータが $24ms + 29ms = 53ms$ 後となる．したがってホップ数に伴って処理時間が増えるため到着にかかる時間が増加する．このことから，送信先までのホップ数が異なる IoT デバイスにおいてセンサデータの受信時刻が異なる．以上のことから中継機能の停止ができず，消費電力の削減ができない．

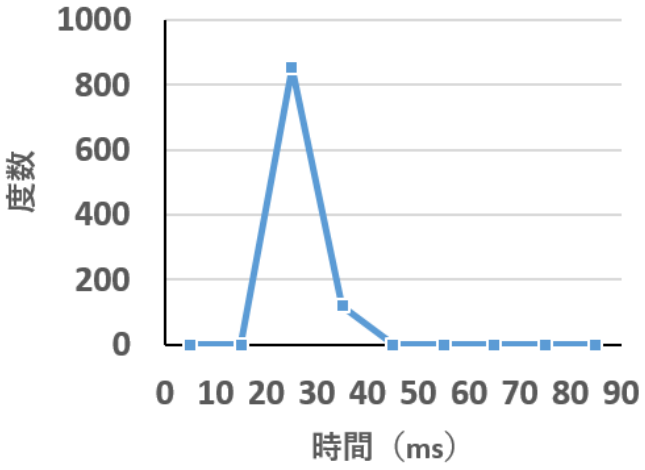

図 3 t_s の頻度分布

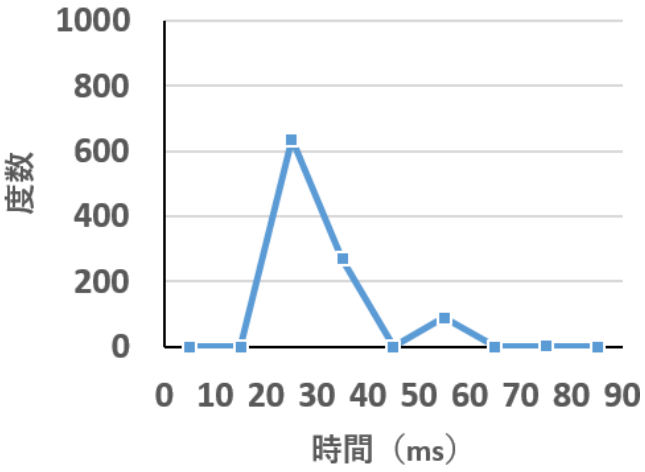

図 4 t_c の頻度分布

各章の概要

第 2 章 関連研究ではマルチホップネットワークの省電力化に関する研究について説明する．第 3 章 提案では課題に対する提案手法とそのユースケースについて説明する．第 4 章 実装では開発したソフトウェアの構成について述べる．第 5 章 実験と分析では提案手法を評価する実験の方法と結果の分析方法について述べる．第 6 章 議論では提案手法について議論すべき内容について説明する．第 7 章 おわりにでは本稿の課題，提案，評価について簡潔に述べる．

2. 関連研究

Rahman らの研究ではマルチホップネットワークにおけ

るデータ集約スキームである QADA が提案されている [9]．この研究は，データパケットを一部のノードに集約することによって省電力化やトラフィック負荷を軽減している．ただし，IoT デバイスにおける送受信の同期に関しては言及されていない．

Karan らの研究では BLE（Bluetooth Low Energy）を使用したネットワークとしてスター型ネットワークとメッシュ型ネットワークを複合したハイブリッド型のマルチホップネットワークが提案されている [10]．この研究では従来の Zigbee を使用したマルチホップネットワークよりも大幅に電力を削減している．一方で Bluetooth を使用する場合には定期的なアドバタイズ及びスキャンの処理が必要なため，通信回数が増加するという課題がある．

El-Hoiydi らの研究では省電力なマルチホップネットワークを実現するための MAC プロトコルである WiseMAC が提案されている．WiseMAC では送受信の同期を取っており，Zigbee を使用したマルチホップネットワークと比較して消費電力を 85%削減している．ただし，この研究ではダウンリンクの通信のみに焦点が当てられている

Xu らの研究では，無線センサネットワークのアルゴリズムにおいて一般的に知られている LEACH を改善した E-LEACH が提案されている．E-LEACH はセンサノードのバッテリ残量を参照することでネットワーク負荷のバランスを取っており，クラスタの規模に応じてクラスタヘッダの役割を担う時間を動的に変更している．ただし，この研究ではシミュレーションによる実験のみに留まっており，ユースケースに即した環境での電力測定は行われていない．

3. 提案

マルチホップネットワークにおける中継機能の停止ができないという課題に対して，IoT デバイスのサーバまでのホップ数と t_s，t_c の値からサーバで送信タイミングを算出するスケジューリング手法を提案する．提案の目的はセンサデータを送信するタイミングをサーバまでのホップ数ごとに設定することにより IoT デバイスの中継機能の起動時間を限定することである．中継機能の起動時間を限定することで中継機能の停止が可能な時間が増加するため，IoT デバイスの消費電力の削減につながる．ただし，IoT デバイスの移動や追加，撤去がある場合にサーバまでのホップ数が変化してしまい，送信タイミングを常にサーバに問い合わせなければならなくなる．したがって提案手法の前提条件として以下を設定する．

- 設置する IoT デバイスは位置が固定されており，追加，撤去がされない．
- 構築するマルチホップネットワークにおいて IoT デバイスが孤立していない．
- センサデータを送信するサーバは 1 つである．

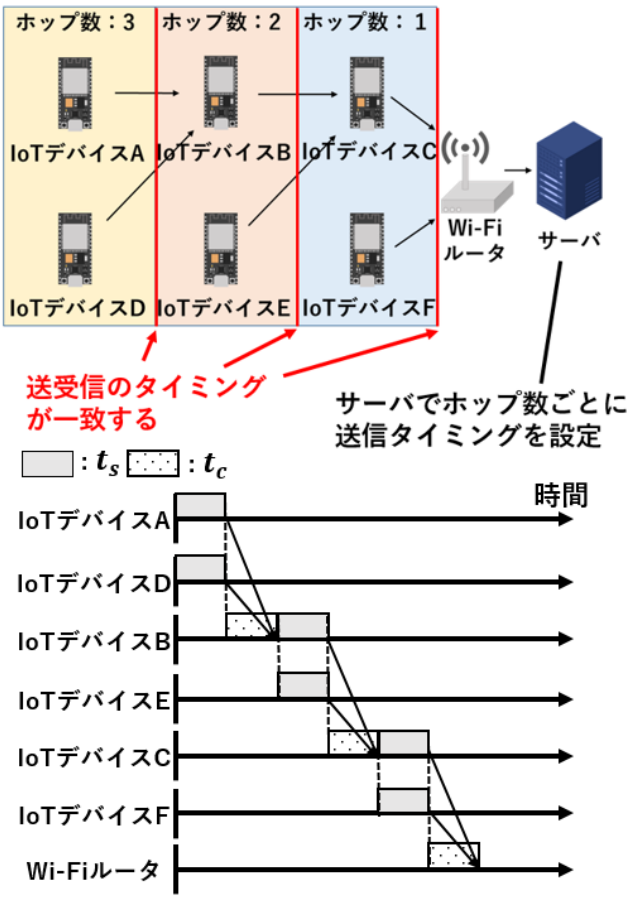

図 5 提案手法の概要図

提案手法の概要図を図 5 に示す．図 5 において，IoT デバイス A，IoT デバイス B，IoT デバイス C，IoT デバイス D，IoT デバイス E，IoT デバイス F は同じ時刻にセンサデータを取得し，サーバにセンサデータを送信する．このとき，IoT デバイス A と IoT デバイス D のセンサデータは IoT デバイス B と IoT デバイス C を中継して送信し，IoT デバイス B と IoT デバイス E のセンサデータは IoT デバイス D を中継して送信するものとする．提案手法ではホップ数ごとに送信するタイミングと受信するタイミングの同期を取り，図 5 における黄色の領域（ホップ数 3），赤色の領域（ホップ数 2），青色の領域（ホップ数 1）において同じタイミングで送信する．これにより IoT デバイス B で IoT デバイス A と IoT デバイス D のセンサデータを受信する時刻と IoT デバイス C で IoT デバイス B と IoT デバイス E のセンサデータを受信する時刻がそれぞれ一致する．以上のスケジューリングによって中継機能の起動時間を短縮し，消費電力を減らす．

次に，送信タイミングの算出アルゴリズムについてアルゴリズム 1 に示す．アルゴリズム 1 の関数 CALC_SEND_TIME が送信タイミングを算出する関数であり，関数の引数はホップ数 hop_num，ネットワークの最大ホップ数 max_hop，t_s，t_c，センシング間隔 $interval$

アルゴリズム 1 送信タイミングの算出アルゴリズム

Input: hop_num: ホップ数, max_hop: ネットワークの最大ホップ数, t_s: センサデータの受信から送信までの時間, t_c: 通信プロセスにかかる時間, $interval$: センシング間隔
Output: $send\_time$: 送信タイミング

```
1: function CALC_SEND_TIME(hop_num, max_hop, t_s, t_c, interval)
2:     Initialize:
3:         send_time ← interval
4:     while max_hop − hop_num ≥ 0 do
5:         send_time ← send_time + t_s
6:         if max_hop − hop_num ≠ 0 then
7:             send_time ← send_time + t_c
8:         end if
9:         hop_num ← hop_num + 1
10:    end while
11:    return send_time
12: end function
```

であり，戻り値として送信タイミング $send\_time$ を算出する．センシング間隔はセンサデータを取得する頻度を表す周期であり，ユーザによって決められる値とする．以下でアルゴリズム 1 の処理について説明する．

最初に，ユーザが決定するセンシング間隔 $interval$ の値で送信タイミング $send\_time$ を初期化する．センシング間隔の値で初期化する理由は，IoT デバイスが最後にセンサデータを取得してからの時間を送信タイミングに設定するためである．次に，$max\_hop - hop\_num \geq 0$ の条件式が True の場合に While 文を実行する．この条件式は IoT デバイスのホップ数が最大ホップ数以下である場合に True となる．次に While 文の中の処理で t_s と t_c を $send\_time$ に加算する．ただし，$max\_hop - hop\_num \neq 0$ の条件式を満たさない場合（ホップ数が最大ホップ数と等しい場合）には t_c の値を加算しないようにしている．理由はネットワークの末端の IoT デバイスにおいては受信時の通信プロセスの時間がかからないためである．最後に hop_num に 1 を加算し，While 文の条件判定に戻る．While 文の条件判定で False になった場合にはループから抜けて $send\_time$ の値を関数の戻り値として返す．

次に，提案手法の手順とその目的について詳しく述べる．提案手法の手順は以下の 5 つである．

① 経路の確立（IoT デバイスがどの IoT デバイスを中継してサーバにセンサデータを送信するかを決める）
② t_s, t_c の取得（センサデータの送信の前段階として内部処理時間と通信処理時間を計測し，サーバに送信する）
③ 送信タイミングの決定（IoT デバイスの送信タイミングをサーバで算出する）
④ 送信タイミングの取得（サーバで算出した送信タイミングを IoT デバイスが問い合わせる）
⑤ 中継機能を起動する時間の設定（送信タイミングと時刻の誤差をもとに，IoT デバイスで中継機能を起動する時間を設定する）

①の手順の目的はセンサデータを送信する際の中継先を IoT デバイスが把握することと IoT デバイスのホップ数をサーバが把握することである．まず，IoT デバイスが通信可能範囲内の IoT デバイスに対して経路制御パケットを送信する．経路制御パケットはセンサデータを取得した IoT デバイスの MAC アドレス，中継した IoT デバイスの MAC アドレス，ホップ数を表すホップカウントを含んでいる．サーバが経路制御パケットを受け取ったとき，最も小さいホップカウントの値をその IoT デバイスのホップ数として登録し，中継した IoT デバイスの MAC アドレスから経路表を作成して IoT デバイスに応答として返す．

②の手順の目的はサーバで送信タイミングを算出するために t_s と t_c をサーバが把握することである．t_s と t_c はそれぞれ 1000 回計測したのちに最頻値を算出する．計測が終了したのち，それぞれの IoT デバイスが t_s と t_c の最頻値をサーバに送信する．

③の手順の目的はサーバが送信タイミングを算出し，IoT デバイスから参照できるようにすることである．送信タイミングは IoT デバイスが最後にセンサデータを取得してからの時間とし，アルゴリズム 1 の CALC_SEND_TIME 関数によって算出される．

④の手順の目的はそれぞれの IoT デバイスがサーバで算出した送信タイミングにセンサデータを送信することである．IoT デバイスは送信タイミングをサーバに問い合わせて取得する．取得した送信タイミングにしたがって最後にセンサデータを取得した時点からタイマーを起動し，送信タイミングの時間が経過したときにセンサデータを送信する．

⑤の手順の目的は隣接する IoT デバイスがセンサデータの送信より前に中継機能を起動することである．中継する IoT デバイスは隣接する IoT デバイスの送信タイミングを参照し，送信される時刻よりも前に中継機能を起動する必要がある．そこで，中継機能を起動する時刻は送信される時刻よりも「センシング間隔 $\times 3\sigma$」だけ早い時刻とする．なお，3σ は IoT デバイスの時刻の誤差の標準偏差を 3 倍した数値であり，例えば ESP32 の場合には約 30ppm である [11]．センサデータの受信後は中継機能を停止する．

図 6 に提案手法のセンサデータの送受信の流れを示す．ホップ数 3 の IoT デバイスからホップ数 2,1,0 の IoT デバイスを中継してサーバに向けて送信するとき，まずホップ数 2 の IoT デバイスがホップ数 3 の IoT デバイスの送信タイミングよりも「センシング間隔 $\times 3\sigma$」だけ早い時刻に中継機能を起動する．次に，ホップ数 3 の IoT デバイスの送信タイミングになったらホップ数 3 の IoT デバイスがホップ数 2 の IoT デバイスに向けてセンサデータを送信し，ホップ数 2 の IoT デバイスが受信した時点で通信を終

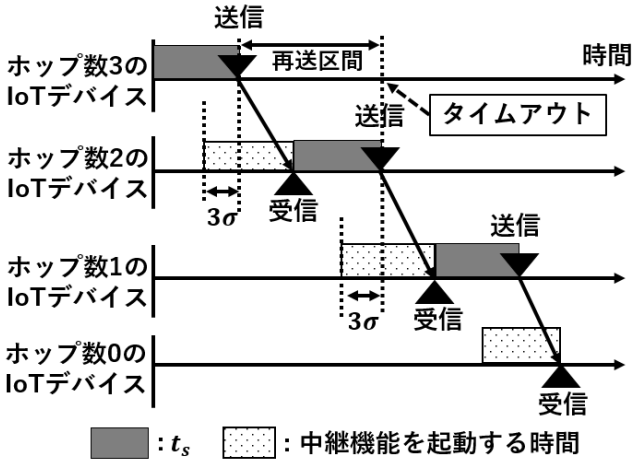

図 6　センサデータの送受信の流れ

了する．送信に失敗した場合はホップ数 2 の送信タイミングまでを再送区間としてセンサデータを再送する．再送区間で送信できなかった場合にはタイムアウトとして処理を終了し，送信できなかったセンサデータを次の送信タイミングに繰り越す．以上の動作を繰り返し行うことによってセンサデータをサーバに送信する．

次に再送区間における再送処理について述べる．ツリー構造のマルチホップネットワークにおける親子関係にある IoT デバイスをそれぞれ親ノード，子ノードとする．図 7 に複数の子ノードを持つ場合のスケジューリング手法と再送処理について示す．図 7 において IoT デバイス A，IoT デバイス B，IoT デバイス C は IoT デバイス D の子ノードであるため，同一のタイミングでセンサデータを送信するとパケットが衝突して受信できない．そのため送信タイミングをずらした時間を予めサーバで算出し，1 台ずつ順番に送信タイミングを設ける．また，送信に失敗した場合の再送処理も同様に 1 台ずつ順番に再送の機会を設ける．ただし，タイムアウト時刻になった場合と再送の処理が必要ない場合には再送しない．

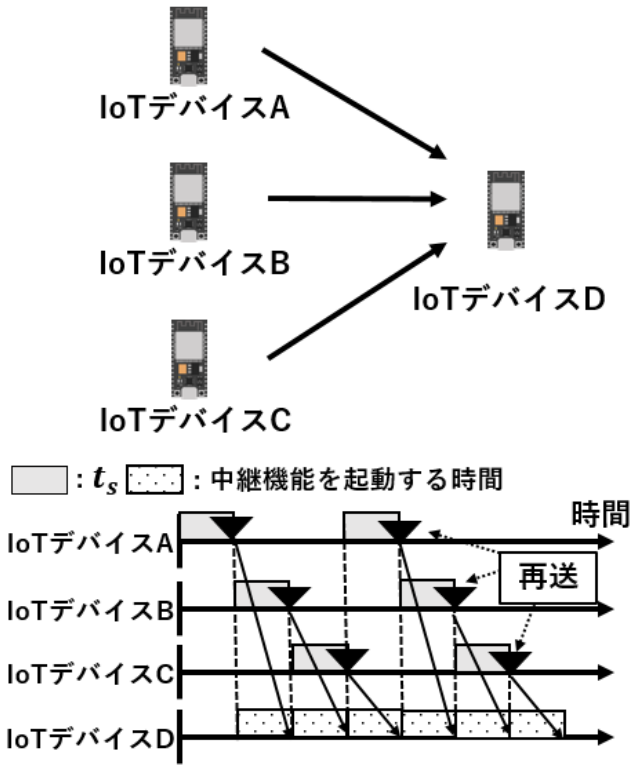

図 7　複数の子ノードを持つ場合のスケジューリング手法と再送処理

ユースケース・シナリオ

提案手法のユースケースとして，駐車場の空き状況を提供するスマートパーキングがある [12–15]．図 8 にスマートパーキングのユースケース図を示す．スマートパーキングはマルチホップネットワークを利用して車両検出センサのデータをサーバに送ることで空き状況を把握しユーザに表示するアプリケーションである [1]．IoT デバイスの距離間隔を 3m としているのは，車両の横幅を 2.5m と想定しているためである．

このようなスマートパーキングの事例の一つとして，株式会社 NTT ドコモが開発した「Smart Parking Peasy」がある[*1]．Smart Parking Peasy は車の入出庫を検出するスマートパーキングセンサとセンサデータを管理サーバに送信するゲートウェイで構成されている．スマートパーキン

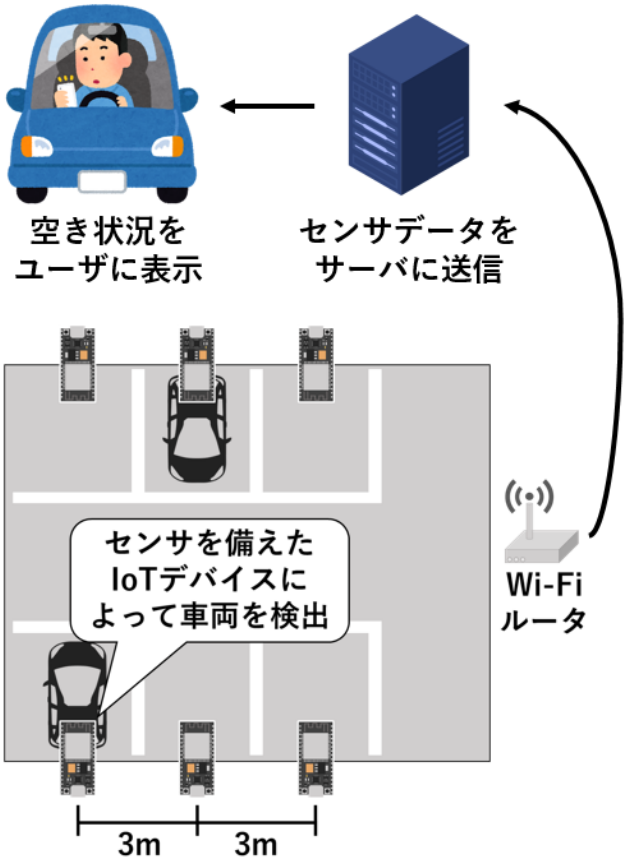

図 8　スマートパーキングのユースケース図

[*1] NTT DOCOMO,INC., https://smartparking.peasy.jp/ (2022/12/21)

グセンサはバッテリで動作しており，ゲートウェイは AC 電源に接続されている．また，スマートパーキングセンサは定期的にセンサデータを取得している．以上のことから，スマートパーキングのユースケースシナリオにおいて提案手法を取り入れることによって消費電力が削減される．

4. 実装

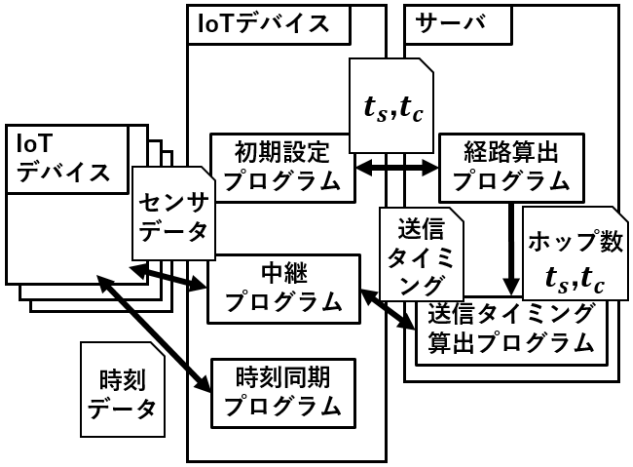

図 9 ソフトウェア構成図

実装は IoT デバイスのハードウェアとして ESP32 を使用し，ソフトウェアを MicroPython で実行する．サーバは Python で API を実装する．ネットワークプロトコルは ESP-NOW を使用する．ソフトウェア構成図を図 9 に示す．IoT デバイスのソフトウェアには初期設定プログラム，中継プログラム，時刻同期プログラムの 3 つがあり，サーバのソフトウェアには経路算出プログラムと送信タイミング算出プログラムの 2 つがある．各プログラムの内容について下記に述べる．

初期設定プログラム

初期設定プログラムは IoT デバイスの t_s と t_c を測定しサーバに送信する役割とサーバから経路表を取得する役割がある．IoT デバイスは最初に経路制御パケットを送信し，サーバから経路表を取得する．その後経路表に基づいて t_s と t_c を測定し，サーバに送信する．

経路算出プログラム

経路算出プログラムは IoT デバイスから送信された経路制御パケットのペイロードを解析し，センサデータを取得した IoT デバイスの MAC アドレスと中継した IoT デバイスの MAC アドレスのリストからセンサデータを送信するための経路を算出する．経路算出プログラムは REST API であり，HTTP の GET リクエストと POST リクエストを受け付ける．

中継プログラム

中継プログラムは他の IoT デバイスから送信されたセンサデータを中継してサーバに向けて送信するためのプログラムである．サーバで算出された送信タイミングをもとに，センサデータを中継するときの中継機能の起動と停止を行う．中継プログラムは初期設定プログラムによって送信される経路制御パケットと t_s，t_c の中継も行う．

送信タイミング算出プログラム

送信タイミング算出プログラムは提案手法に基づいて IoT デバイスの送信タイミングの算出をするプログラムである．算出した送信タイミングは API を介して IoT デバイスが取得できるようにする．送信タイミングの算出に必要な入力の値は経路を算出したときの値を使用する．

時刻同期プログラム

時刻同期プログラムは他の IoT デバイスと時刻を同期し，時刻の誤差を縮めるためのプログラムである．同期する時刻はサーバを基準とし，サーバに近い IoT デバイスに合わせて時刻を調整する．したがってサーバに近い IoT デバイスに向けて時刻データの要求をし，時刻データ要求されたときは自身の時刻データを応答として返す．

5. 実験と分析

実験環境

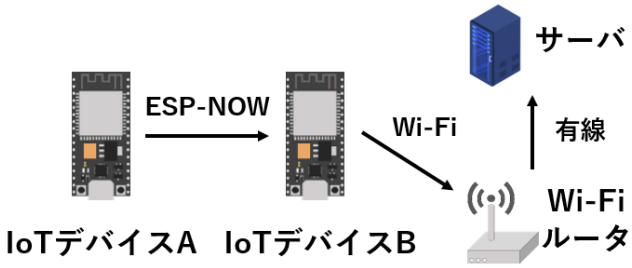

図 10 実験環境の図

実験環境の図を図 10 に示す．東京工科大学のクラウド・分散システム研究室で IoT デバイスとして 2 台の ESP32（IoT デバイス A と IoT デバイス B）を ESP-NOW で無線接続し，Wi-Fi ルータを介したサーバへの送信を可能にした．送信するセンサデータはタイムスタンプ，送信元の IoT デバイス名，車両を検出するための距離センサの距離データに見立てた 32bit の浮動小数点とした．また，センサデータ 1 個あたりのデータ量は 16bytes である．

実験方法としては，IoT デバイス A が IoT デバイス B と Wi-Fi ルータを中継して 10 秒に一回の頻度でサーバにセンサデータを 1 個送信し，これを 100 回繰り返す．このときセンサデータを中継する IoT デバイスにおいてパケット到達率を計測する．パケット到達率は実験開始後の経過時間までに受信したパケット数を送信パケット数で割ったときの百分率とする．1000 秒後に全ての送信が完了した時点で実験を終了する．

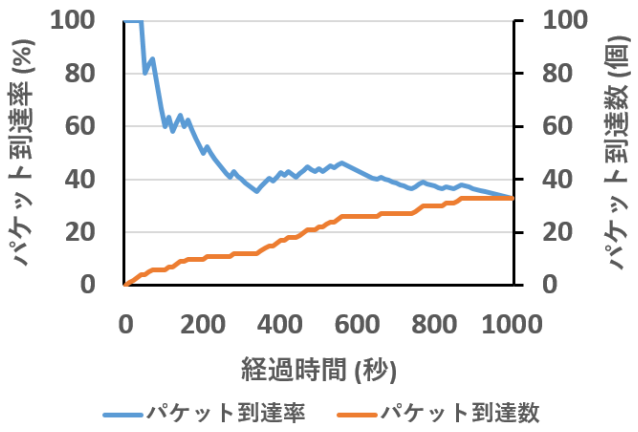

図 11 実験開始後の経過時間に対するパケット到達率とパケット到達数

実験結果と分析

提案手法の評価指標は IoT デバイスのパケット到達率とする．図 11 に実験開始後の経過時間に対するパケット到達率とパケット到達数を示す．図 11 より，パケット到達率は実験開始後からおよそ 400 秒後まで減少傾向にあり，その後 40%付近を横ばいで推移している．また，最終的なパケット到達率は 33%であり，これは 100 個送信したセンサデータのうち 33 個がサーバに到達したことを表している．パケット到達率が 33%と低い値になった理由としては，消費電力の削減のためにセンサデータを 1 回送るたびに通信の初期化をしていたためである．通信が初期化されることにより IoT デバイス B が受信するときに準備ができていないまま IoT デバイス A からセンサデータが送信されたことに起因する．

6. 議論

提案手法では送信タイミングとして IoT デバイスが最後にセンサデータを取得してからの時間をタイマーでセットする方法を取っている．しかし，より消費電力を削減するために DeepSleep を使用するとタイマー自体がリセットされてしまうために送信タイミングにセンサデータが送信できない．解決するための方法としては，IoT デバイスが DeepSleep の状態から起動するために要する時間を予め測定し，DeepSleep の時間+起動にかかる時間の分を送信タイミングの時間から差し引き，起動した時点からのタイマーとその時間を比較する方法がある．

パケット到達率が 33%と低いことに対しての解決策としては，最初に通信の疎通確認をする制御パケットを送信することである．制御パケットを送信して通信が可能であることを確認したうえでセンサデータを送信することにより，パケット到達率が向上する．ただし，制御パケットを送り続けてしまうと送信する IoT デバイスにおいて消費電力が増加してしまうため，制御パケットを送る回数に制限を設ける必要がある．実験結果のパケット到達率は 33%であり，平均しておよそ 3 回に 1 回は到達していることを考慮すると，制御パケットを送る回数は 3 回 ~4 回が適切である．

7. おわりに

本稿の課題は，サーバまでのホップ数の違いにより中継する IoT デバイスの中継機能を停止することができないことである．そこで送受信のタイミングを一致させるために送信タイミングをサーバで算出するスケジューリング手法を提案した．実験として 2 台の ESP32 を無線接続し，パケット到達率によって提案手法を評価した結果，33%であった．

参考文献

[1] Chinrungrueng, J., Sunantachaikul, U. and Triamlumlerd, S.: Smart Parking: An Application of Optical Wireless Sensor Network, *2007 International Symposium on Applications and the Internet Workshops*, pp. 66–66 (online), DOI: 10.1109/SAINT-W.2007.98 (2007).

[2] Radunovic, B. and Le Boudec, J.-Y.: Rate performance objectives of multihop wireless networks, *IEEE transactions on Mobile computing*, Vol. 3, No. 4, pp. 334–349 (2004).

[3] Cai, Z. and Chen, Q.: Latency-and-coverage aware data aggregation scheduling for multihop battery-free wireless networks, *IEEE Transactions on Wireless Communications*, Vol. 20, No. 3, pp. 1770–1784 (2020).

[4] Iqbal, S., Qureshi, K. N., Kanwal, N. and Jeon, G.: Collaborative energy efficient zone-based routing protocol for multihop Internet of Things, *Transactions on Emerging Telecommunications Technologies*, Vol. 33, No. 2, p. e3885 (online), DOI: https://doi.org/10.1002/ett.3885 (2022).

[5] Fourati, L., El-Kaffel, S., Mnaouer, A. B. and Touati, F.: Study of Nature Inspired Power-aware Wake-Up Scheduling Mechanisms in WSN, *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 2154–2159 (online), DOI: 10.1109/IWCMC48107.2020.9148433 (2020).

[6] Jayakumar, H., Lee, K., Lee, W. S., Raha, A., Kim, Y. and Raghunathan, V.: Powering the internet of things, *Proceedings of the 2014 international symposium on Low power electronics and design*, pp. 375–380 (2014).

[7] Iannacci, J.: Microsystem based Energy Harvesting (EH-MEMS): Powering pervasivity of the Internet of Things (IoT)–A review with focus on mechanical vibrations, *Journal of King Saud University-Science*, Vol. 31, No. 1, pp. 66–74 (2019).

[8] 杉本一彦，串田高幸：”電池残量にもとづく無線中継機能の ON/OFF によるデータ受信期間の延長”，研究報告モバイルコンピューティングと新社会システム (MBL)，Vol. 2022, No. 6, pp. 1–8 (2022).

[9] Rahman, H., Ahmed, N. and Hussain, M. I.: A hybrid data aggregation scheme for provisioning Quality of Service (QoS) in Internet of Things (IoT), *2016 Cloudification of the Internet of Things (CIoT)*, pp. 1–5 (online), DOI: 10.1109/CIOT.2016.7872917 (2016).

[10] Nair, K., Kulkarni, J., Warde, M., Dave, Z., Rawalgaonkar, V., Gore, G. and Joshi, J.: Optimizing power consumption in iot based wireless sensor networks us-

ing Bluetooth Low Energy, *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 589–593 (online), DOI: 10.1109/ICGCIoT.2015.7380533 (2015).

[11] 河竹純一, 杉本一彦, 串田高幸: ”無線 LAN のマルチホップネットワークにおける送受信のタイミングの一致による IoT デバイスの省電力化”, クラウド・分散システム研究室, Vol. CDSL-TR-109 (2022).

[12] Benson, J. P., O'Donovan, T., O'Sullivan, P., Roedig, U., Sreenan, C., Barton, J., Murphy, A. and O'Flynn, B.: Car-park management using wireless sensor networks, *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, IEEE, pp. 588–595 (2006).

[13] Urdiain, L. O., Romero, C. P., Doggen, J., Dams, T. and Van Houtven, P.: Wireless sensor network protocol for smart parking application experimental study on the arduino platform, *2nd International Conference on Ambient Computing, Applications, Services and Technologies*, Citeseer (2012).

[14] Lin, T. S., Rivano, H. and Le Mouël, F.: Performance comparison of contention-and schedule-based mac protocols in urban parking sensor networks, *Proceedings of the 2014 ACM international workshop on wireless and mobile technologies for smart cities*, pp. 39–48 (2014).

[15] Wenzhi, C. and Bai, L.: A smart roadside parking navigation system based on sensor networks for ITS, *2006 IET International Conference on Wireless, Mobile and Multimedia Networks*, IET, pp. 1–4 (2006).

