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

河竹 純一[1]　杉本 一彦[1]　串田 高幸[1]

概要：無線 LAN のマルチホップネットワークにおける IoT デバイスはバッテリ交換頻度の削減や長期的な監視のために省電力化が求められている．センサデータは IoT デバイスを中継してサーバに送信される．課題は，中継する IoT デバイスが中継機能を停止できないことによって消費電力が削減できないことである．中継機能を停止できない理由は IoT デバイスによってサーバまでのホップ数が異なり，受信するタイミングが一致しないためである．提案手法として，IoT デバイスのサーバまでのホップ数と中継するときの処理時間及び通信時間からセンサデータを送受信するタイミングをサーバで決定し，中継機能の起動時間を減らす．送受信するタイミングは，タイムアウト時刻までに送信されたセンサデータを全て受信し，次のホップ先が同一の IoT デバイス全てが送信可能な状態になるような時間とする．実験では ESP32 を 6 台使用し，屋外でセンサデータを送信する実験を行う．評価では IoT デバイスの消費電力と稼働時間について，中継機能を常に起動する手法と提案手法を比較する．

1. はじめに

背景

スマートパーキングにおいて，自動で車両を検知して空き状況をユーザに知らせる際に無線 LAN のマルチホップネットワークが使用される．無線 LAN のマルチホップネットワークはバッテリで駆動する複数の IoT デバイスで構成されている．ここでの IoT デバイスとは，センサからのデータ（以下センサデータ）の取得と，ワイヤレス通信が可能なマイクロコントローラのことである．本稿ではマイクロコントローラとして Espressif Systems 社の ESP32 を使用する．スマートパーキングにおけるマルチホップネットワークでは，ネットワークを構成する IoT デバイスが定期的にセンサデータを取得し，ゲートウェイを通じてサーバに送信している．また，各 IoT デバイスは通信可能範囲内にある IoT デバイスのセンサデータを転送する役割がある．

マルチホップネットワークの消費電力に関する研究では，一般的にネットワーク全体の寿命を長期化すること，すなわち各 IoT デバイスの消費電力を可能な限り削減することが求められている [1]．スマートパーキングにおいては，IoT デバイスの消費電力を削減することによって，バッテリ交換頻度の削減や長期的な監視が可能になる．例えば 1 つの IoT デバイスにつきバッテリ容量 10000mAh，出力電圧 5V のモバイルバッテリを搭載すると，6 か月間動作させるためには 1 日当たりの消費電力量を約 277mWh に抑える必要がある．

課題

無線 LAN のマルチホップネットワークにおける課題は，中継する IoT デバイスが中継機能を停止できないことによって消費電力が削減できないことである．中継機能を停止できない理由は，同時刻にセンサデータを取得した IoT デバイスが直ちにセンサデータを送信すると，中継する

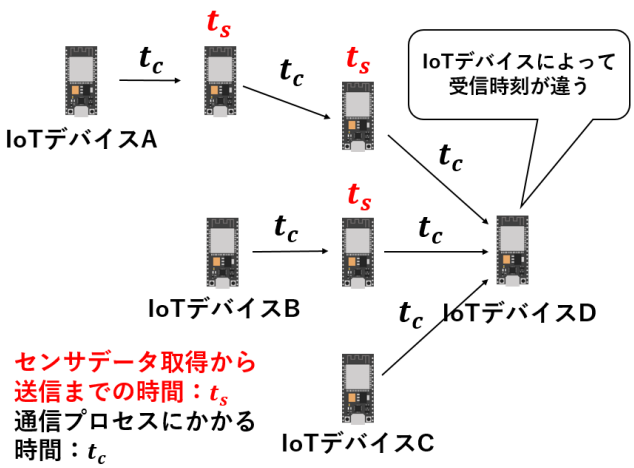

図 1　センサデータの受信タイミングが一致しないときの例

[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

IoT デバイスがセンサデータを受信するタイミングが一致しないためである．ESP32 の場合，中継機能を起動すると中継機能を停止しているときと比べて 2.5 倍の電力を消費する [2]．したがって消費電力が削減できないことによって IoT デバイスの稼働時間が短くなるため，長期的な監視ができなくなる．

図 1 に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示す．図 1 は IoT デバイス A，IoT デバイス B，IoT デバイス C がそれぞれ IoT デバイス D に対してセンサデータを送信する状況を表している．その際，IoT デバイス A は 2 ホップ，IoT デバイス B は 1 ホップを要し，IoT デバイス C は直接 IoT デバイス D と通信する．また，センサデータは同時刻に取得するものとし，センサデータの受信から送信までの時間を t_s，通信プロセスにかかる時間を t_c とした．t_s と t_c について ESP32 を用いて 1000 回の計測をした結果，最頻値として t_s が 29ms，t_c が 24ms という結果が得られた．これを例として t_s を 24ms，t_c を 29ms とすると，IoT デバイス D における IoT デバイス A のセンサデータの受信時刻は $3 \times 29ms + 2 \times 24ms = 135ms$ 後，IoT デバイス B のセンサデータの受信時刻は $2 \times 29ms + 24ms = 82ms$ 後，IoT デバイス C のセンサデータの受信時刻は $29ms$ 後である．したがって，IoT デバイスによって受信時刻が異なることがわかる．また，ホップ数が増えると t_s と t_c 処理の回数が比例して増えるため，それにかかる時間も増加する．なお，t_s の値は IoT デバイスの CPU の周波数や処理速度によって，t_c の値は IoT デバイス間の距離，媒質，遮蔽物の有無，通信プロトコルによって決定される．以上のことから中継する IoT デバイスは中継機能を停止することができず，消費電力を削減することができない．

各章の概要

第 2 章 関連研究では，課題に関連する研究とその比較をする．第 3 章 提案では，課題を解決するための提案方法とそのユースケースシナリオについて詳しく述べる．第 4 章 実装と実験方法では，提案方式をもとに開発するソフトウェアやハードウェアの実装と実験方法について説明する．第 5 章 評価と分析では，実験の結果から提案内容を評価する方法とその分析方法について説明する．第 6 章 議論では，本稿の提案，実験，評価について議論すべき内容について述べる．最後に，第 7 章 おわりにで本稿のまとめと今後の展望について述べる．

2. 関連研究

Rahman らは，マルチホップネットワークにおいてデータパケットを複数のノードに集約するためのハイブリッドデータ集約スキーム QADA を提案した [3]．この提案によって消費電力やトラフィックの負荷の削減を実現している．この研究では消費電力とトラフィックの負荷のトレードオフの関係について述べているが，IoT デバイスの送受信のタイミングについては言及されていない．

Karan らは，マルチホップネットワークに低電力通信規格である Bluetooth Low Energy を適用し，ネットワークトポロジーをスター型とメッシュ型のハイブリッドとする手法を提案した [4]．これにより従来の NRF24LOIP と Zigbee を使用するプラットフォームと比較して消費電力を大幅に削減した．ただし，Bluetooth Low Energy の通信は定期的なアドバタイズ通信とスキャンが必要になるため，1 回の通信にかかる消費電力が低いとしても通信回数が増えるという課題がある．

El-Hoiydi らは，マルチホップネットワークにおける省電力な MAC プロトコルとして送信と受信のタイミングを合わせる手法 WiseMAC を提案した [5]．同等の遅延時間のもとで WiseMAC を Zigbee プロトコルと比較した結果，WiseMAC は約 85%の消費電力を削減した．ただし，この研究ではダウンリンクのみに焦点を当てており，アップリンクについては言及されていない．

Xu らは，データ集約型のマルチホップネットワークで一般的に用いられている LEACH プロトコルを改良した E-LEACH プロトコルを提案した．E-LEACH アルゴリズムでは、ネットワーク負荷の均衡を保つためにセンサノードの残りの電力を考慮し、最適なクラスターサイズに応じてラウンドタイムを変更している。これにより従来の LEACH プロトコルよりもネットワークの寿命を 40%延ばした．ただし，この研究は 100 個のノードをランダムに配置するシミュレーションによって結果を提示しており，実際のユースケースに即した環境での実験が不足している．

3. 提案方式

提案手法として，サーバまでのホップ数ごとにセンサデータを送信するタイミングをサーバで決定する手法を確立する．提案手法の目的は中継する IoT デバイスがセンサデータを受信するタイミングを揃えることによって中継機能を起動する時間を減らすことである．これにより中継機能を停止する時間が増えるため，消費電力が削減される．なお，提案手法ではホップ数をもとにセンサデータを送信するタイミングを決定するため，IoT デバイスの移動や追加があるとホップ数や経路が経過時間に伴って変化することから送信タイミングが定まらない．そこで，提案手法の前提条件を以下とする．

- IoT デバイスの位置は固定し，追加及び撤去はされない．
- ネットワークから孤立している IoT デバイスは存在しない．
- 送信先となるサーバは 1 つとする．

基礎実験

センサデータを送信するタイミングを決定する手法を確立するために，基礎実験としてセンサデータの受信から送信までの時間 t_s と通信プロセスにかかる時間 t_c を計測した．この実験の目的は，t_s，t_c にかかる時間と平均値からのばらつき具合を調べることである．

まず通信プロセスにかかる時間 t_c を計測した．実験は ESP32 を 2 台用意し，1 台に高精度な外部 RTC モジュールである DS3231 を取り付けた．DS3231 の誤差範囲は 0°C∼40°C の環境において ±2ppm である．実験方法は，DS3231 を備えた ESP32 からもう一方の ESP32 に対してデータを送信するプログラムを作成し，プログラムにおいて送信するコマンドの前後で時刻を計測した．そのときの時刻の差を t_c として，1000 回計測した．送信したデータはスマートパーキングで車両を検出するときに使われる距離センサのセンサデータを想定し，32bit の単精度浮動小数点とした．ただし，ESP-NOW での通信を実装するうえで浮動小数点を文字列型にキャストする必要があったため，パケットペイロードのサイズは 9byte となった．

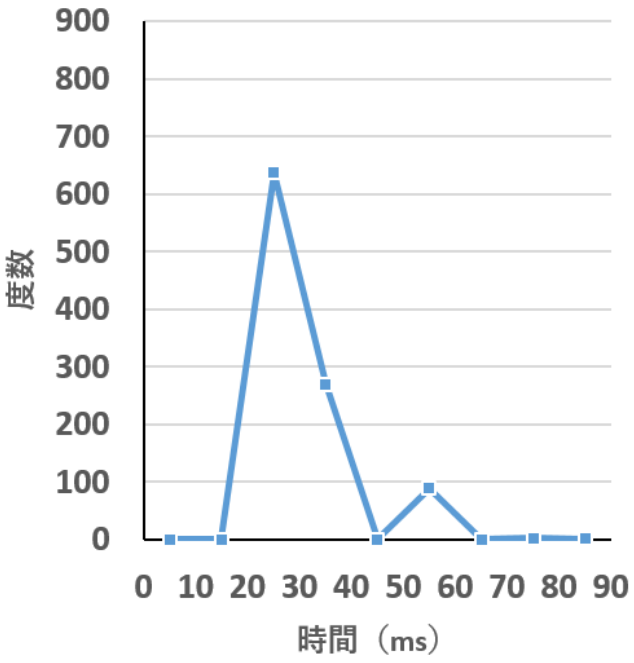

図 2　t_c の度数分布

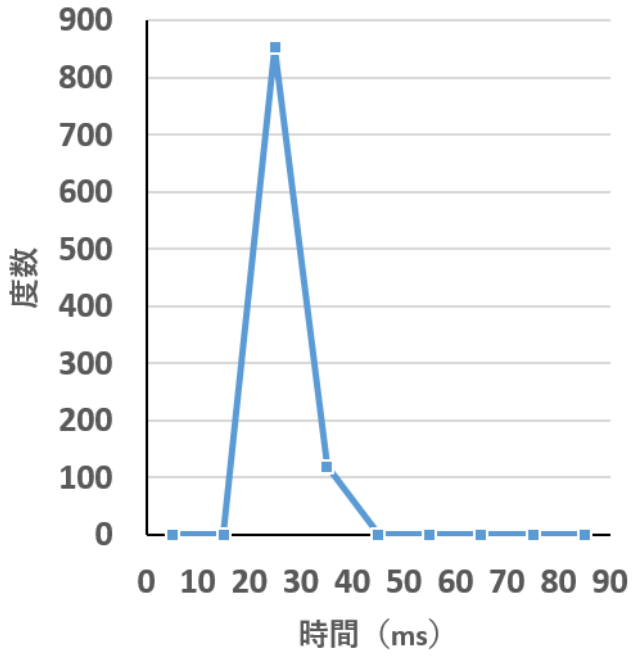

図 3　t_s の度数分布

プログラミング言語処理系として MicroPython1.18 を使い，通信方法として ESP-NOW[*1]を使った．ESP-NOW は Espressif Systems 社が開発した最大 20 台のピアツーピア通信が可能な通信プロトコルであり，通信速度は最大で 1Mbps である．図 2 に t_c の度数分布を示す．最も度数が高い階級は (20, 30] であり，その度数は 637 であった．また，平均値は 32.44ms，最頻値は 29ms，中央値は 30ms であった．さらに標準偏差を調べたところ 7.81 であったが，実際の値は図 2 からわかるように (40, 50] の階級の度数が 0 であるのに対して (50, 60] の階級に 89 個分布しているように，断続的な値になっている．これは再送処理が発生したことによって，1 回の通信あたりの時間の整数倍の数値が出ていることが推測される．

次に，センサデータの受信から送信までの時間 t_s を計測した．実験は ESP32 を 3 台用意し，1 台に DS3231 を取り付けた．実験方法は，DS3231 を備えた ESP32 から 1 台の ESP32 を中継して別の ESP32 に対してデータを送信するプログラムを作成した．中継する ESP32 のプログラムでデータを受信してから送信するまでの時間を t_s とし，t_c と同様に 1000 回計測した．図 3 に t_s の度数分布を示す．なお，この実験で得られたデータの総数は 971 個であった．これは，1000 回の通信のうち 29 回の通信が失敗したことが原因である．最も度数が高い階級は (20, 30] であり，その度数は 850 であった．また，平均値は 26.74ms，最頻値は 24ms，中央値は 26ms であった．標準偏差は 3.02 であり，t_c と比べるとばらつきが少ない．以上の結果から，t_c は再送処理の有無によって変化し，t_s は t_c に比べて変化が少ないことがわかった．

提案手法ではセンサデータを送信するタイミングを決定する手法について確立するが，通信を成功させるためには受信する IoT デバイスが事前に中継機能を起動している必要がある．したがって，中継機能を起動する時間として通信を待ち受ける時間を設定する必要がある．なぜなら，IoT デバイスが時刻を参照する内部 RTC に誤差があり，送受信のタイミングが前後するためである．そこで ESP32 の内部 RTC の誤差がどの程度生じていてどのように変動するのかについて高精度な RTC モジュールと比較するこ

[*1] https://micropython-glenn20.readthedocs.io/en/latest/library/espnow.html

とによって調べた．まず，ESP32 を 1 台用意し，DS3231 を取り付けた．実験方法としては，ESP32 の内部 RTC と DS3231 の両方で 10 分間の始まりと終わりを計測し，式 1 によって誤差 e を算出して記録する．式 (1) において T_{DS} は DS3231 による 10 分間の秒数，T_{ESP} は DS3231 で 10 分間が経過したときの ESP32 が示した経過秒数である．

$$e[ppm] = \frac{T_{DS} - T_{ESP}}{T_{DS}} \times 10^6 \qquad (1)$$

この操作を 100 回繰り返すプログラムを MicroPython 1.18 で実装した．その結果を以下に示す．

図 4 に実験回数に対する時刻の誤差の散布図を示す．時刻の誤差は-54.21ppm から-12.05ppm の範囲で変動した．これは DS3231 の誤差範囲から考えても有意な誤差であることがわかる．

図 5 に時刻の誤差の度数分布を示す．平均値，最頻値，中央値をそれぞれ算出した結果，平均値は-37.28，最頻値は-36.14，中央値は-36.14 となった．これらの値は全て-40～-35 の階級に位置しており，-40～-35 の階級を境に，離れた階級ほど度数が低くなっている．また，標準偏差 σ は

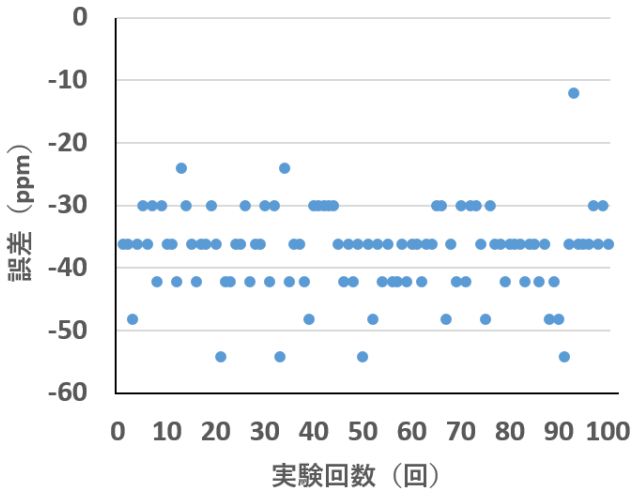

図 4　実験回数に対する時刻の誤差の散布図

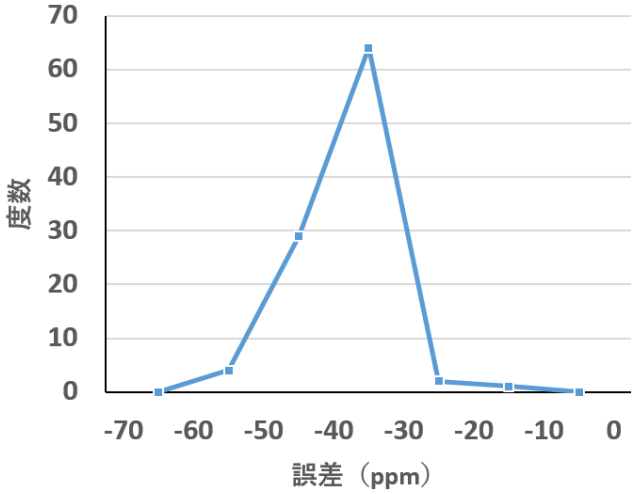

図 5　時刻の誤差の度数分布

6.83 であり，100 回の実験のうち 99 回の時刻の誤差が「平均値 $\pm 3\sigma$」の範囲に存在していた．これらの結果から，提案手法で用いる IoT デバイスの時刻の誤差は，平均値 $\pm 3\sigma$ の範囲とする．

提案方式

図 6 に提案手法の概要図を示す．図 6 は 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 がそれぞれサーバにセンサデータを送信している状況であり，サーバまでのホップ数ごとに色で領域を分けた図である．青色の領域がホップ数 1，赤色の領域がホップ数 2，黄色の領域がホップ数 3 の領域を示している．提案手法ではそれぞれのホップ数ごとにセンサデータを送信するタイミングを設定しているため，IoT デバイス A と IoT デバイス D が IoT デバイス B に送信するタイミング，IoT デバイス B と IoT デバイス E が IoT デバイス C に送信するタイミング，IoT デバイス C と IoT デバイス F が Wi-Fi ルータを介してサーバに送信するタイミングがそれぞれ一致する．したがってホップ数ごとに送受信のタイミングが一致する．

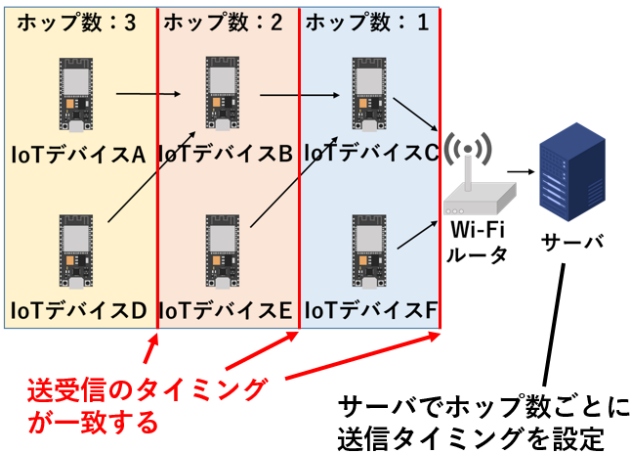

図 6　提案手法の概要図

次に，アルゴリズム 1 に送信タイミングの算出アルゴリズムを示す．送信タイミングは前回にセンサデータを取得した時刻からの秒数を指定する．ここで入力する値は，IoT デバイスのホップ数 n_h，ネットワークの最大ホップ数 N_h，センサデータの受信から送信までにかかる処理時間 t_s，通信プロセスにかかる時間 t_c，センシング間隔 i_s である．センシング間隔とは，IoT デバイスがセンサデータを取得する間隔のことであり，ユースケースとなるアプリケーションやユーザによって決まる．センサデータを出力される値は送信タイミングの時間 τ である．以下にアルゴリズム 1 の処理の流れを説明する．

まず，τ を i_s として初期化する．i_s で初期化する理由は，送信タイミングの時間を前回にセンサデータを取得した時

アルゴリズム 1 送信タイミングの算出アルゴリズム

Input: n_h: ホップ数, N_h: 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, i_s センシング間隔
Output: τ: 送信タイミング

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

刻からの秒数としているため，センシング間隔となる時間が必ずかかるためである．次に，$N_h - n_h \geq 0$ の条件を満たす場合に while 文の処理を実行する．while 文の処理としては，まず τ にセンサデータの受信から送信までの時間である t_s を足す．これはホップ数ごとに処理が発生するためである．次に $N_h - n_h \neq 0$ であるとき，すなわち末端の IoT デバイスでないときに通信プロセスにかかる時間である t_c を τ に足す．これは末端の IoT デバイス以外では受信するときの通信プロセスの時間がかかるためである．while 文の最後では n_h に 1 を足すことで次のループに移行する．関数の戻り値は計算結果である τ としている．

次に，提案手法の手順の概要を示す．

① 経路の確立
② t_s，t_c の取得
③ 送信タイミングの決定
④ 送信タイミングの取得
⑤ 中継機能を起動する時間の設定

提案方式の手順の目的と詳しい説明を以下に示す．

① 経路の確立
この手順の目的は IoT デバイスが経路表を取得することとサーバが IoT デバイスのホップ数を把握することを目的としている．まず，IoT デバイスは経路制御パケットをブロードキャストによってフラッディングさせる．経路制御パケットは送信元の IoT デバイスの MAC アドレス，経由した IoT デバイスの MAC アドレス，経由するたびに 1 増加するホップカウントを含む．サーバは経路制御パケットを受け取ると経路表を応答として IoT デバイスに返す．サーバは IoT デバイスから受け取った経路制御パケットのホップカウントの中で最小となる値をその IoT デバイスのホップ数として登録する．

② t_s，t_c の取得
この手順の目的はセンサデータを中継するときに発生する処理時間と通信時間を把握することである．センサデータの受信から送信までの時間 t_s と通信 プロセスにかかる時間 t_c を調べるために，送信時にかかる時間と中継するときにかかる時間を計測する．これを 1000 回計測したのちに平均化した t_s，t_c を算出する．t_s，t_c は各 IoT デバイスからサーバに送信する．

③ 送信タイミングの決定
この手順の目的は IoT デバイスから参照される送信タイミングをサーバで算出することである．サーバは t_s，t_c と IoT デバイスのサーバまでのホップ数から送信タイミングを決定する．送信タイミングは，タイムアウト時刻までに送信されたセンサデータを全て受信し，次のホップ先が同一の IoT デバイス全てが送信可能な状態になるような時間を算出する．送信タイミングは前回にセンサデータを取得した時刻からの秒数を指定する．ただし，ホップ先の IoT デバイスが複数存在する場合にセンサデータの送信時にパケットが衝突することがある．そこで，実際に送信する時刻は算出した送信タイミングからタイムアウト時刻までの時間の中でランダムに決定する．これにより送信する時刻がずれるため，パケットの衝突を防ぐことができる．また，再送処理が発生した場合には再びタイムアウト時刻までのランダムな時刻に再送することでパケットの衝突を防ぐ．

④ 送信タイミングの取得
この手順の目的は，IoT デバイスがセンサデータを送信するタイミングを把握することである．送信する IoT デバイスはサーバに送信タイミングを問い合わせて自身の送信タイミングを取得する．IoT デバイスはセンサデータを取得した直後からタイマーを起動し，サーバに問い合わせた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．また，センサデータの送信時には受信する IoT デバイスの時刻データの要求をし，返答として送信先の IoT デバイスからも時刻データを受け取ることで時刻の同期をする．

⑤ 中継機能を起動する時間の設定
この手順の目的は，センサデータが送信されたときに確実に受信できるように IoT デバイスの中継機能を起動することである．受信する IoT デバイスは，自身よりもホップ数が 1 大きい IoT デバイスの送信タイミングから中継機能を起動する時刻を設定する．中継機能を起動する時刻は送信タイミングよりも「前回の送信時から次の送信タイミングまでの時間 $\times 3\sigma$」早い時刻とする．また，センサデータを受信したあと直ちに中継機能を停止し，受信したセンサデータを送信するための処理に入る．

図 7 に提案手法のセンサデータの送受信の流れを示す．

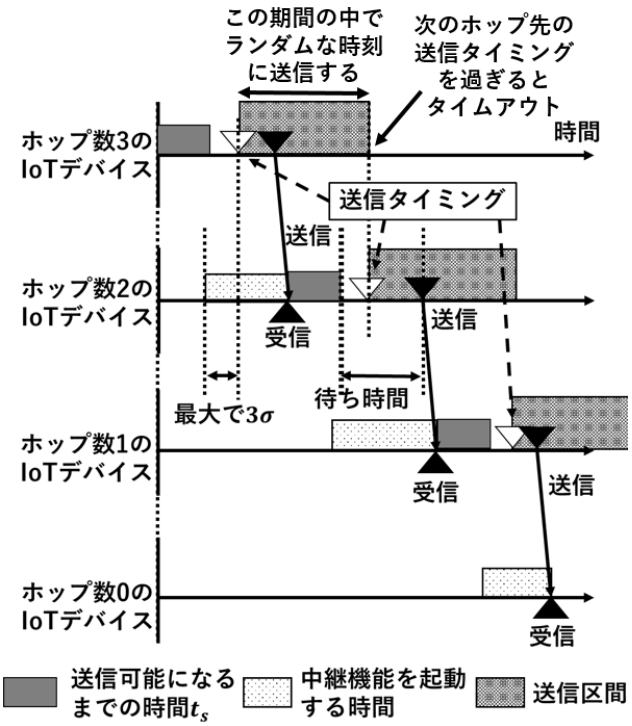

図 7 センサデータの送受信の流れ

図 7 ではサーバまでのホップ数が 0，1，2，3 の IoT デバイスがあり，末端の IoT デバイスであるホップ数 3 の IoT デバイスからサーバに対して順にセンサデータを送っている．また，図 7 の時間軸の始まりは IoT デバイスが同時にセンサデータを取得するタイミングである．まず，ホップ数 3 の IoT デバイスはセンサデータを送信することが可能になった後，提案手法の送信タイミングまで待機する． 次に，ホップ数 2 の IoT デバイスはホップ数 3 の IoT デバイスの送信タイミングに合わせて受信するための中継機能を起動する時間を設定する．その後ホップ数 3 の IoT デバイスは送信タイミングからタイムアウト時刻までの期間でランダムな時刻にセンサデータを送信する．センサデータを受信したホップ数 2 の IoT デバイスは，ホップ数 3 の IoT デバイスがセンサデータを送信するときと同様にホップ数

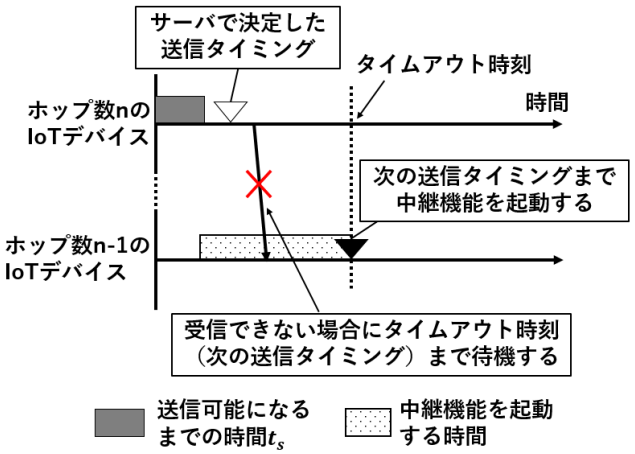

図 8 タイムアウト時の流れ

1 の IoT デバイスに対してセンサデータを送信する．この動作を繰り返すことによってセンサデータがサーバに到達する．このときホップ数ごとに送信タイミングを設けていることによって受信タイミングが一致する．したがって中継機能を起動する時間以外の時間で中継機能を停止することができるため，消費電力が削減される．

次に，センサデータが送信タイミングに送信されなかった場合に，受信する IoT デバイスが中継機能を起動する時間を制限するためのタイムアウト時刻について説明する．タイムアウト時刻を設定することによって中継機能が起動し続けることを防ぐ．図 8 にタイムアウト時の流れを示す．図 8 はホップ数 n の IoT デバイスからホップ数 n-1 の IoT デバイスにセンサデータを送信する状況である．ここでホップ数 n の IoT デバイスから送信されるはずのセンサデータがホップ数 n-1 の IoT デバイスで受信できなかった場合，タイムアウト時刻まで中継機能を起動する．タイムアウト時刻はホップ数 n-1 の次の送信タイミングとする．

ユースケース・シナリオ

図 9 にユースケースシナリオの概要を示す．本稿のユースケースは，駐車場の空き状況を監視するスマートパーキングである．スマートパーキングでは車両を検出するセンサを備えた複数の IoT デバイスによって構成されるマルチホップネットワークを活用しており，サーバに送られるセ

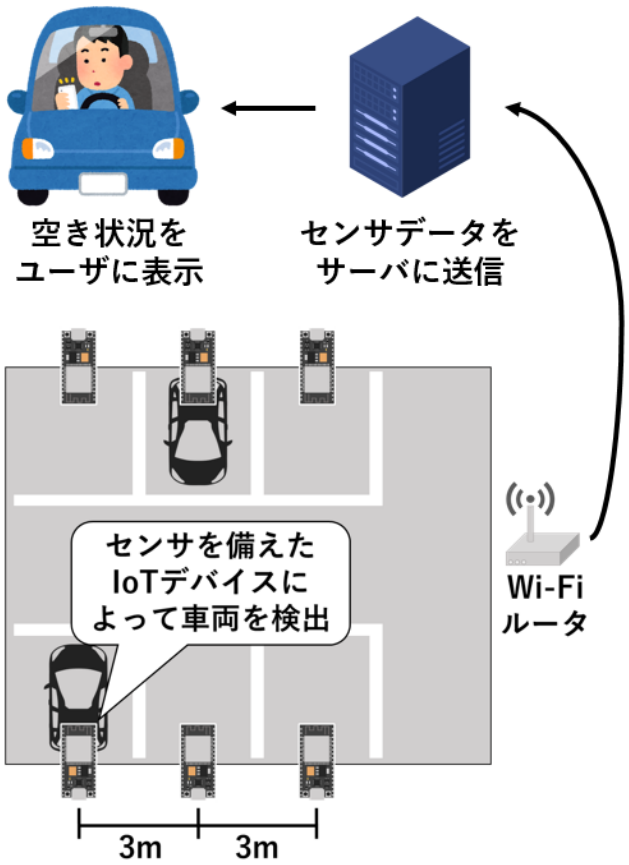

図 9 ユースケースシナリオの概要

ンサデータを参照することによって空き状況をユーザに表示することができる [6]．図 9 では，車両の横幅を 2.5m として IoT デバイスが 3m 間隔で並んでいることを想定している．このユースケースに本稿の提案手法を適用することによって消費電力が削減され，IoT デバイスに電力供給をしているバッテリの交換頻度が少なくなる．

4. 実装と実験方法

実装

IoT デバイスのソフトウェアは MicroPython で実装し，サーバのソフトウェアは Python で実装する．また，IoT デバイスとして ESP32 を使用する．図 10 にソフトウェアの構成図を示す．IoT デバイスのソフトウェアは，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 3 つを備えている．また，サーバのソフトウェア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 つを備えている．これらのプログラムについて以下で説明する．

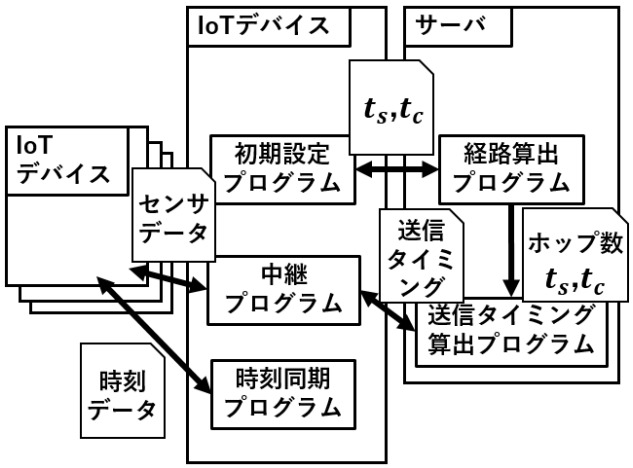

図 10 ソフトウェア構成図

初期設定プログラム

初期設定プログラムはサーバまでのホップ数とセンサデータ取得から送信までの時間 t_s，通信プロセスにかかる時間 t_c をサーバに登録するためのプログラムである．IoT デバイスは経路制御パケットをサーバの経路算出プログラムに対して送信し，応答を待つ．

経路算出プログラム

経路算出プログラムは初期設定プログラムから送られた経路制御パケットをもとに，経路表を応答として返すためのプログラムである．また，初期設定プログラムから送られた t_s，t_c を受け取る．

中継プログラム

中継プログラムは主にセンサデータの中継と取得したセンサデータの送信をする．初回のみ送信タイミングをサーバに問い合わせ，以降はその送信タイミングにしたがって送信する．また，初期設定プログラムからサーバ宛てに送信された経路制御パケットや t_s，t_c のデータの中継も行う．なお，本稿のユースケースとしているスマートパーキングでは車両の有無を検出したときのみセンサデータを送信する．したがって中継機能は毎回起動されるが，センサデータが届かない場合はタイムアウト時刻に中継機能を停止する．

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，経路算出プログラムから得られるホップ数と t_s，t_c の値から提案手法に則って送信タイミングを算出するためのプログラムである．算出した送信タイミングは中継プログラムから要求があった際に応答として返す．

時刻同期プログラム

時刻同期プログラムは隣接している IoT デバイスと時刻同期を行うプログラムである．時刻は受信タイミングと同期するためにサーバに近い IoT デバイスに合わせて同期する．したがって，時刻同期プログラムから要求を受け取った IoT デバイスが時刻データを応答として返す．

実験環境

図 11 に実験環境の図を示す．実験環境として，東京工科大学八王子キャンパスの屋外駐車場で IoT デバイス 6 台を無線接続し，IoT デバイスが Wi-Fi ルータを介してサーバと通信できるような環境を想定する．IoT デバイス間の距離は，本研究のユースケースとしているスマートパーキングに倣って 3m とする．IoT デバイス 6 台は図 11 の実線で示す経路で接続する．

実験方法は，図 11 の電流を測定する IoT デバイス 4 台に電流を測定するセンサを取り付け，電流を測定すること

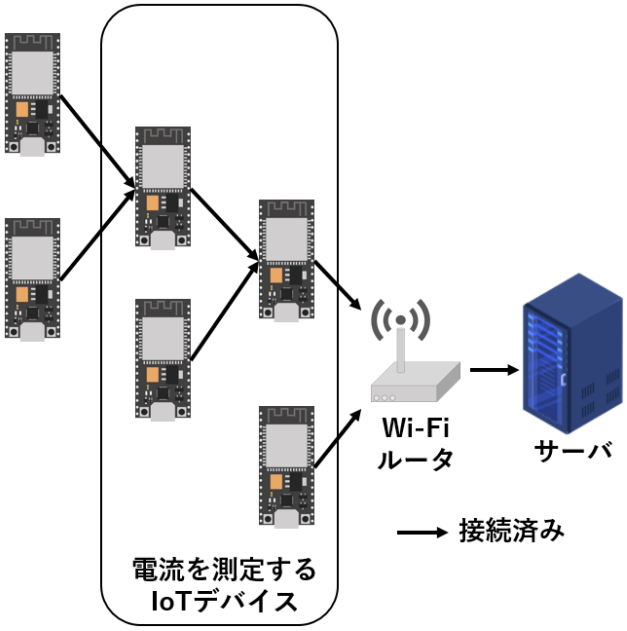

図 11 実験環境の図

によって消費電力を算出する．この 4 台は中継機能を有する IoT デバイスであり，本稿の提案手法によって消費電力が削減される．したがって 6 台のうち 4 台で電流を測定し評価する．消費電力は IoT デバイスの動作電圧が一定（3.3V）であるものとして電流値と乗算する．

5. 評価手法と分析手法

IoT デバイスの消費電力と稼働時間について，中継機能を常に起動する手法と提案手法を比較することによって評価する．また，横軸を実験開始時からの経過時間，縦軸を消費電力とした折れ線グラフを示すことによって IoT デバイスの消費電力について分析する．さらに，バッテリ容量を 10000mAh としたときの IoT デバイスの稼働時間についてもグラフを示し，分析する．

6. 議論

提案手法では，通信プロセスにかかる時間 t_c が変わらないものとして，最初に設定した値のみを用いていた．しかし，実際のネットワークの環境においては t_c の値が経過時間に伴って変化する場合がある．そこで，動作中の IoT デバイスからも t_c の値を取得し，更新することによって環境が変化したときにも対応できる．

また，提案手法ではネットワーク全体における末端の IoT デバイスから順にセンサデータが送信される．しかしリンクによって最大ホップ数に差がある場合，IoT デバイスはセンサデータが到着するまで送信を待機しなければならない．その解決のため，マルチホップネットワーク全体を Wi-Fi ルータを親とした木構造と捉える．そこで，部分木ごとに送信タイミングを個別に設定する．これによりホップ数が異なる IoT デバイスのセンサデータの到着を待たずともセンサデータの送信が可能になる．

7. おわりに

本稿の課題は中継する IoT デバイスが中継機能を停止することができないことによって消費電力が削減できないことである．提案手法として，サーバまでのホップ数ごとにセンサデータを送信するタイミングを決定する手法を確立する．提案手法によって中継する IoT デバイスがセンサデータを受信するタイミングがそろうため，中継機能を起動する時間を減らすことができる．IoT デバイスの消費電力と稼働時間について，中継機能を常に起動する手法と提案手法を比較することによって評価する．

参考文献

[1] 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).

[2] 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).

[3] 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).

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

[5] El-Hoiydi, A. and Decotignie, J.-D.: WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks, *Proceedings. ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No.04TH8769)*, Vol. 1, pp. 244–251 Vol.1 (online), DOI: 10.1109/ISCC.2004.1358412 (2004).

[6] 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).