ファームウェア更新の時のリレーノード数と残存エネルギーにもとづく送信データ量の割り当てによる消費電力の平準化

井出 佑[1]　大沢 恭平[2]　串田 高幸[1]

概要：クラスタートポロジーのネットワークでは，クラスターヘッド (以後 CH と呼ぶ) が通信を中継する．これにより，CH は他のノードと比べて消費電力が大きく，ノード内で消費電力が偏る．これを解決するために，CH なることができる複数の IoT 機器の中で CH を交代する仕組みがある．複数の IoT 機器を使用するシステムでは分散配置された IoT 機器に対して無線によるファームウェア更新が行われる．課題は，ファームウェア更新時の更新データの配布を 1 台の CH が担当して消費電力が増加することにより，稼働時間が短くなることである．提案として，データの配布を担当できるノード (以後 RN とする) の台数で送信パケットの総数を等分割し，バッテリー残量の偏差から各 RN が送信するパケット数を分配する手法を提案する．実験では ESP32 を 6 台用意し，うち 3 台を RN として各ノードのファームウェア更新時の消費電力，更新時間，パケットロス率を計測した．比較はノード 1 台がデータの配布を担当した場合と提案手法を適用した場合を比較した．この時，送信したデータはファームウェアに見立てた 750[KB] のバイナリファイルである．結果として，クラスターの更新時間は送信を担当するノードが 1 台の場合は約 1392 秒，提案手法の場合は約 550 秒で約 883 秒短縮できた．消費電力の標準偏差は送信を担当するノードが 1 台の場合は約 99[mWh]，提案手法の場合は約 7[mWh] で約 92[mWh] 減少した．全てのノードの消費電力の総和は送信を担当するノードが 1 台の場合で約 872[mWh]，提案手法で約 505[mWh] であり，約 367[mWh] 削減できた．これは，提案手法によりクラスターの更新完了までの時間が短縮された結果，消費電力が削減量の総和が，増加量の総和を上回ったためである．RN から NN へファームウェアのパケットを送信した際のパケットロス率に関しては，3 台の NN の内，2 台の NN で最大で約 0.028[%]，最小で約 0.01[%] 高くなった．

1　はじめに

背景

農業分野では IoT 機器を使用して環境モニタリング，灌漑制御，病害予測が行われている．これにより，水，肥料，農薬の効率的運用が可能となり，収穫量の増大につながる [1].

このような IoT システムでは，多数の IoT 機器を農地内に分散配置し，それらから無線通信を使用しデータを伝送，収集する．農業で使用される IoT 機器はインフラが整っていない屋外で使用され，電源の確保が難しいためバッテリー駆動で運用される [2].

屋外の広範囲に IoT 機器を展開する場合，アクセスポイントが全ての IoT 機器との通信をカバーしきれない場合がある．この場合，マルチホップネットワークが使用される [3]．マルチホップネットワークとは，通信を行う際に複数の機器を経由してデータの送信を行うネットワークである．ここの機器はノードと呼ばれる [4]．これにより，ネットワークの通信距離の拡張，長距離通信による送信電力の削減，ルーティングによるネットワークの信頼性の向上が期待できる [5,6].

マルチホップネットワークのトポロジーの一つにクラスタートポロジーがある．クラスタートポロジーとは，ネットワーク内のノードをクラスターと言うグループ単位に分割し，クラスター内の代表のノードが他のクラスター内のノードのデータを収集し，サーバーに送信を行うトポロジーである．クラスタートポロジーでは代表のノードをクラスターヘッド (以後 CH とする)，それ以外のノードをクラスターメンバー (以後 CM とする) と呼ぶ．クラスタートポロジーのネットワーク構成を使用することで，CM からのデータ集約とサーバーへの送信を CH が担うことで，CM が直接基地局へ送信する際の負荷を削減できる．この

[1] 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
[2] 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 クラウド・分散システム研究室
〒 192–0982 東京都八王子市片倉町 1404-1

方式は通信距離の短縮化とデータの重複を削減する効果により全体の消費電力を抑制できる．CH には集約処理，転送処理，管理通信が集中するため，CM よりも消費電力が増加する [7,8].

また，クラスター内に CH になることができるノードが複数ある場合，CH は交代することができる．CH を交代することでノードのエネルギーを均等に使用することができ，特定のノードのバッテリーだけが早期に枯渇することを防ぐことができる [9–13]．CH の交代の判断は自身または交代先のノードのバッテリー残量をもとに行われる．交代の判断はラウンドごとに行われる [11–13]．ラウンドとは，CH の選出とクラスターの形成，データの送受信を 1 サイクルとして扱う単位である [10–12].

IoT 機器を使用したシステムでは物理的に分散配置された IoT 機器に対して，ファームウェア更新，機能追加，ソフトウェアのバグ修正を行うために，Over-The-Air(以後 OTA とする) アップデートを行う [14]．クラスタートポロジーのネットワークでの OTA アップデートでは，サーバーが CH に更新データを送信する．そして，CH がブロードキャストまたはマルチキャストにより CM に対して更新データを送信する．この時，パケットロスが発生した場合の再送信処理は CH が行う．再送信にはユニキャストが使用される [15–17].

課題

課題は，クラスタートポロジーのネットワークにおいてファームウェア更新を無線で行う時に，データの送信と再送信を全て 1 台の CH が引き受けると，その CH の消費電力が増加し稼働時間が減少することである．図 1に 1 台の CH でファームウェア更新を行った場合の例を示す.

CH の消費電力が増加すると，バッテリー残量をもとに CH を交代する場合に交代の頻度が増加する．これにより，ノード間の通信頻度が増加し，クラスターのネットワーク寿命が短くなる.

図 1では，ファームウェアの配布を旧 CH1 台が担当しているため，旧 CH のバッテリー残量が低くなっている．この場合，ラウンド終了後に CH を交代するため旧 CH は，新 CH を選出し新 CH に CH を交代することを知らせる．新 CH は自身が新 CH であることを知らせるために広告パケットを全ての CM に送信する．この処理が頻発することで，クラスター全体の消費電力が増加する.

各章の概要

第 2 章では関連研究について議論する．第 3 章では，課題に対しての提案方式について説明する. 第 4 章では，実装したソフトウェアについて説明する．第 5 章では，評価実験について説明する．第 6 章では，提案手法についての

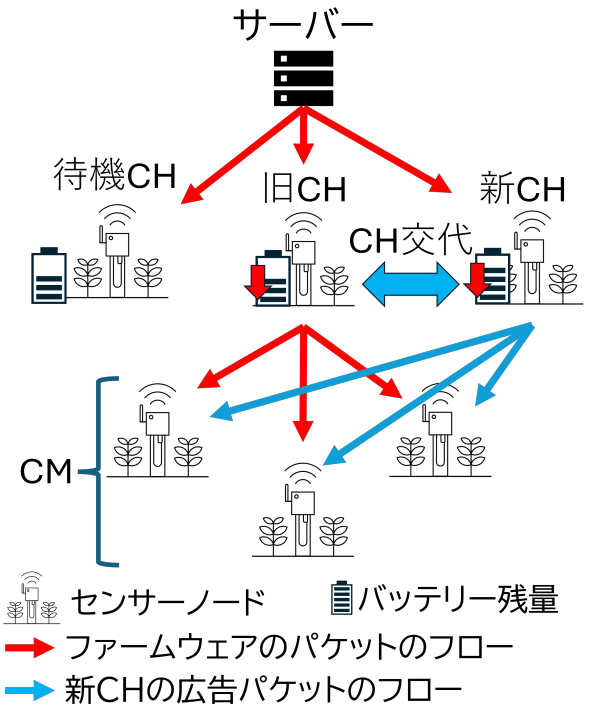

図 1: 1 台の CH でファームウェア更新を行った場合の例

議論をする．第 7 章では，本稿のまとめを行う.

2 関連研究

クラスター内の各ノードで自身が CH になる確率にもとづいてランダムに CH になるかを決定し，CH をローテーションさせるプロトコルの LEACH を提案した研究がある [10]．これにより，全ノードの消費電力を均等化し，ネットワーク寿命を最大化することができる．この研究はアップリンクを想定したものであり，ダウンリンクを想定している本稿とは異なる.

クラスター内で各ノードの送信頻度が多く，初期エネルギーが少ないノードから CH に選出することで，最後のノードの稼働時間を最大化するアルゴリズムを提案した研究がある [18]．この研究は，ネットワーク寿命を最大化することを重きにおいており，従来行われてきたネットワーク全体の平均寿命を最大化する研究とは異なる．本稿では，一部のノードへの負荷の集中を回避すること行うため目的が異なる.

Wireless Sensor Network において遠隔でソフトウェア更新を行う時に，LEACH 方式を利用してネットワーク全体をクラスター化し，CH を中継としてノード全体にデータを配布する提案を行った研究がある [15]．この提案により，ソフトウェア更新時の消費電力と更新時間を削減することができる．この提案は，各クラスターに 1 台ずつ CH を配置して，そこからクラスター全体にデータを送信している．この研究は，ソフトウェア更新時に 1 台の CH に負

荷が集中することに対処できていないため，改善の余地がある．

3 提案

前提条件

本稿の提案を適用する上での前提条件を示す．

- ノードはセンサーデータを定期的に送る時に自身のバッテリー残量も送信しているため，サーバーは全てのノードのバッテリー残量を定期的に受信している
- 全てのノードとサーバーと通信が可能なノードを Relay Node(RN) としそれ以外を Non-Relay Node(NN) とする
- RN は全ての NN と通信が可能である
- RN と NN の数は固定である

提案方式

本稿では，無線を使用したファームウェアアップデートにおいて，特定の RN への負荷集中によるエネルギー消費の偏りを抑制することを目的とする．提案として，クラスター内の RN の台数とバッテリー残量から各ノードの NN へ送信するパケット数とシーケンス番号の範囲を決定し，各 RN が決められたシーケンス番号のパケットを配布する手法を提案する．RN と NN 間の通信では最初にブロードキャストでデータ送信を行った後，再送信をユニキャストでデータ送信を行う．提案方式について分割フェーズと送信フェーズの 2 つに分けて説明する．

分割フェーズ

このフェーズでは，各 RN が NN へ送信するパケットの数とシーケンス番号の範囲を決定する．まず，ノードに送信するファームウェアのファイルを RN から NN へ送信する際のパケットのペイロードサイズに分割する．この時のパケットの総数を P とする．その後，パケットの総数を RN の数である C で等分した P_{eq} を求める．C は必ず正の整数になる．本稿の実験では C は 1 もしくは 3 になる．割り切れずに余りが発生した場合はその値を P_{rem} とする．これを式に表したものを式 (1) に示す．

$$P_{eq} + P_{rem} = \frac{P}{C} \tag{1}$$

次に，各 RN を i として各 RN のバッテリー残量の偏差 $B_{dev,i}$ を求める．この時，求めた値の小数点以下は切り捨てる．偏差の単位は%である．等分したパケット数のうちの各 RN の偏差分のパケット数を，各 RN に加算するパケット数 $P_{add,i}$ とする．これを式に表したものを式 (2) に示す．

$$P_{add,i} = \lfloor P_{eq} \times \frac{B_{dev,i}}{100} \rfloor \tag{2}$$

等分したパケット数 P_{eq} とその RN に加算するパケット数 $P_{add,i}$ の和をその RN i が送信を担当するパケット数 $P_{send,i}$ とする．これを式に表したものを式 (2) に示す．

$$P_{send,i} = P_{eq} + P_{add,i} \tag{3}$$

もし，加算するパケット数 $P_{add,i}$ が小数の場合，少数点以下の値を切り捨てる．また，全ての RN の加算するパケット数 $P_{add,i}$ の総和が 0 以外の場合，P_{rem} と総和の差を算出する．これを式に表したものを式 (4) に示す．

$$P_{rem} = P_{rem} - \sum_{i=1}^{C} P_{add,i} \tag{4}$$

また，P_{rem} が 0 以外の場合，P_{rem} が 0 になるまでバッテリー残量が高い順に一つずつ P_{send} を振り分ける．この時各 RN のバッテリー残量の順位を $rank(i)$，余りを割り当てられる上位ノードの集合を R とする．これを式に表したものを式 (5) に示す．

$$P_{send} = \begin{cases} P_{send,i} + 1 & \text{if } P_{rem} > 0 \text{ and } rank(i) \in R \\ P_{send} & \text{otherwise} \\ P_{send,i} - 1 & \text{if } P_{rem} < 0 \text{ and } rank(i) \in R \end{cases} \tag{5}$$

送信フェーズ

このフェーズでは，分割フェーズで決定した各 RN が送信するデータのシーケンス番号の範囲に沿って，NN へデータを送信する．表 1に図 2と図 3における RN のバッテリー残量とパケットの総数が 6000 個の場合の送信パケット数を示す．

表 1: 図 2と図 3における RN のバッテリー残量とパケットの総数が 6000 個の場合の送信パケット数

ノード	バッテリー残量 (%)	送信パケット数 (個)
RN1	90	2080
RN2	85	1980
RN3	85	1940

まず，RN はサーバーからファームウェアを受信し保存する．その後，サーバーから自身が送信を担当するシーケンス番号の範囲と，保存したファームウェアと照らし合わせて，その範囲だけを読み込み NN へ送信する．この時，200[ms] から 300[ms] のランダムな時間間隔でブロードキャストを使用して送信する．送信間隔の最低値を 200[ms] にした理由は，実装で使用する通信プロトコルの ESPNOW

のデフォルトのビットレートが 1[Mbps] [*1] であり，これを余裕で下回る通信速度にすることで，パケットの取りこぼしを無くすためである．送信間隔の最大値を 300[ms] にした理由は，あるノードで最小値で送信するノードと最大値で送信するノードがあった場合でも，タイミングが被らないようにするためである．この送信が終了した後，RN は各 NN に対して，送信が終了したことを通知するパケットを送信する．

図 2に RN から NN へのデータ送信の例を示す．図 2では RN1 から順番に表 1の送信パケット数にもとづいて，シーケンス番号の範囲が割り振られている．

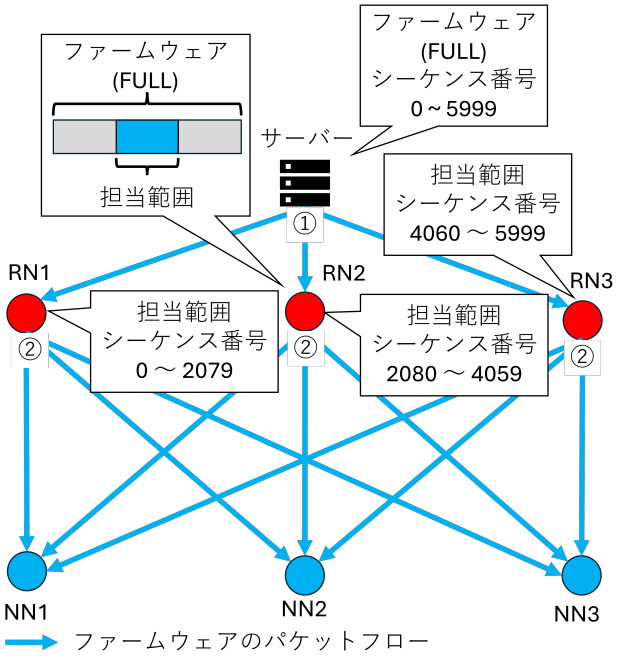

図 2: RN から NN へのデータ送信の例

NN は RN から送信完了のパケットを受信した後，受信できていないシーケンス番号を探す．もしロスがある場合は，パケットロスリスト (PL) をその範囲を担当した RN に送信する．これを受信した RN はユニキャストで再送信を行う．もし，NN で各 RN が担当したシーケンス番号の範囲ごとで，全てのパケットが揃ったら，その RN に対して更新が完了したことを通知するパケットを送信する．各 RN で全ての NN のデータ送信が完了したことを確認したら，サーバーに更新完了の通知を送信し，更新を終了する．図 3に NN の再送信要求と更新が完了した場合の例を示す．

図 3では，各 RN に表 1に従ったシーケンス番号の範囲が割り振られている．また，NN1 はシーケンス番号 8 と 1000 をロスしている．この場合 RN1 に対してシーケンス番号 8 と 1000 を含む PL を送信している．また，他の RN に対しては完了通知を送信している．NN2 はシーケンス

[*1] https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-reference/network/esp_now.html

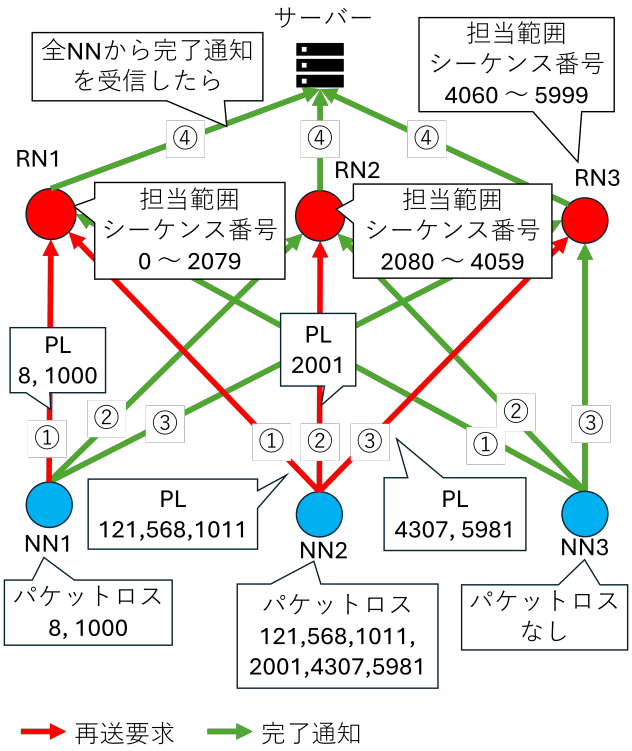

図 3: NN の再送信要求と更新が完了した場合の例

番号 121，568，1011，2001，4307，5981 をロスしている．この時，NN1 は RN1 に対して 121，568，1011 を含む PL を送信する．RN2 に対しては 2001 を含む PL を送信する．RN3 に対しては 4307，5981 を含む PL を送信する．NN3 はパケットロスが無いため，全ての RN に対して完了通知を送信している．もし，各 RN で全ての NN からの完了通知を受信したらサーバーに対して完了通知を送信する．

ユースケース・シナリオ

本稿のユースケースとして，スイカ畑で IoT 機器を使用した環境モニタリングシステムをクラスタートポロジーのネットワークで運用する場合を想定する [15]．図 4にスイカ畑のユースケースで提案を適用した場合の例を示す．この時，バグ修正，新機能の追加のためにファームウェアアップデートを行う．このユースケースでは，センサーの設置間隔が 5m である．このユースケースに本稿の提案を適用することで，ファームウェアアップデート時に特定の CH に負荷が集中して，消費電力が偏ることを抑える事ができる．

4 実装

提案ソフトウェアとしてサーバーに，RN の数とバッテリー残量をもとにファームウェアファイルを分割して各 RN に割り当てるソフトウェアである「Cluster base Data Segmenter(CDS)」を実装する．また，RN にサーバーで割り

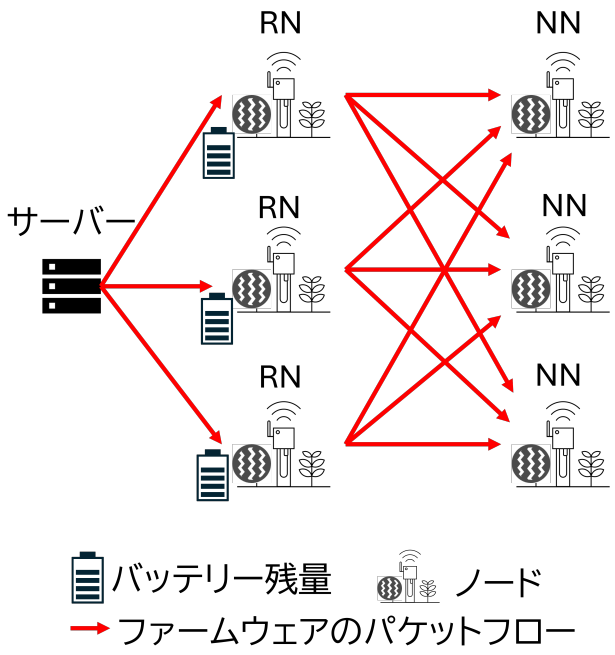

図 4: スイカ畑のユースケースで提案を適用した場合の例

当てられたシーケンス番号をもとにデータを読み込み，NN へデータを送信する「Cluster base Data Loader(CDL)」を実装する．そして，NN に RN からデータを受信し，パケットロスを検知して担当の RN に再送信要求を行う「Cluster base Data Receiver(CDR)」を実装する．CDS は Python で実装し，CDL と CDR は Micropython で実装する．サーバーと RN 間は Wi-Fi で通信し，RN と NN 間は ESP-NOW で通信する．更新の開始から更新終了までの流れを項目に分けて説明し，そこからユニキャストの処理について説明する．

更新開始から RN へのデータの送信

図 5に更新開始から RN へファームウェアを送信するまでの処理の流れを示す．

まず，管理者が更新開始の依頼をサーバーに送信する．これを CDS が受信すると，送信するファームウェアを RN から NN へ送信する際のペイロードのサイズに分割し，総パケット数を取得する．この時のペイロードサイズは 200 バイトである．そして，総パケット数，RN の数，各バッテリー残量から，各 RN が NN へ送信するパケット数を決定する．その後，RN をランダムな順番で選び，送信するシーケンス番号の範囲を番号の先頭から順番に割り当てる．その後，全ての RN の Config ファイルを作成する．Config ファイルの内容は，送信するシーケンス番号の範囲，総パケット数，ファームウェアのデータ容量，ファームウェアのハッシュ値である．Config ファイルを作成した後，クラスター内の全ての RN に対して，対応する Config ファイルを送信する．RN は Config ファイルを受信する

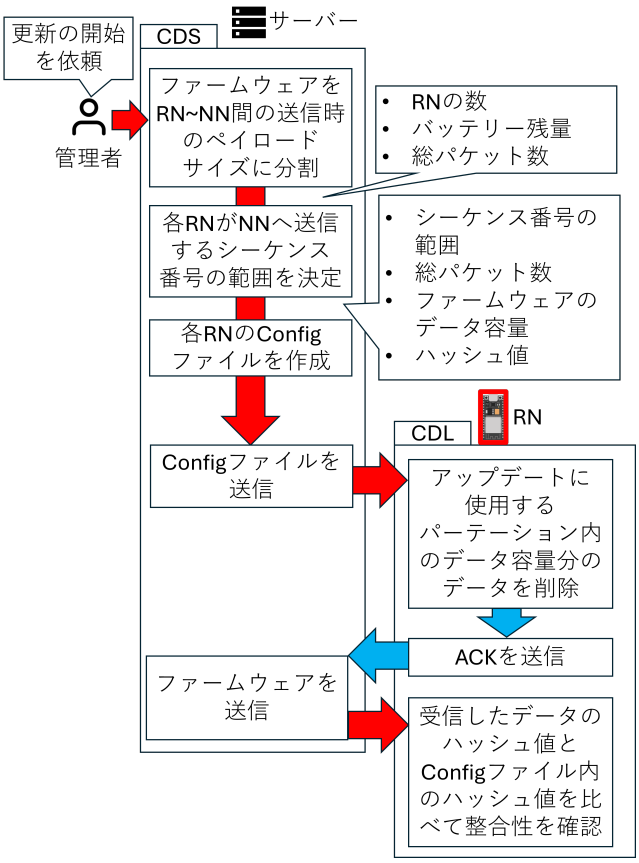

図 5: 更新開始から RN へデータを送信するまでの処理の流れ

と，更新に使用するパーテーション内のファームウェアのデータ容量分のデータを削除する．削除が完了した後，RN はサーバーに ACK を返す．ACK を受信したサーバーは ACK を送信してきた RN に対してファームウェアを送信する．ファームウェアの受信が終了した後，RN は受信したファームウェアのハッシュ値を計算し，Config ファイル内のハッシュ値と比較して，ファイルの整合性を確認する．

RN から NN へのデータ送信から更新完了

図 6に RN の NN へのデータ送信から更新完了までの処理の流れを示す．

まず，RN は全ての NN に対して，ファームウェアのデータ容量，ファームウェアのハッシュ値，自身が送信を担当するシーケンス番号の範囲をユニキャストで送信する．これを受信した RN は，アップデートに使用するパーテーション内のファームウェアのデータ容量分の領域のデータを削除する．削除が完了した後，ACK を RN に送信する．RN は全ての NN から ACK を受信したら，担当範囲のデータをシーケンス番号の順番に読み込み，NN へブロードキャストで送信する．この時，200[ms] 間隔でデータを送信する．NN はデータを受信するたびに，受信したパケットのシーケンス番号を記録する．RN が担当範囲のデータを全

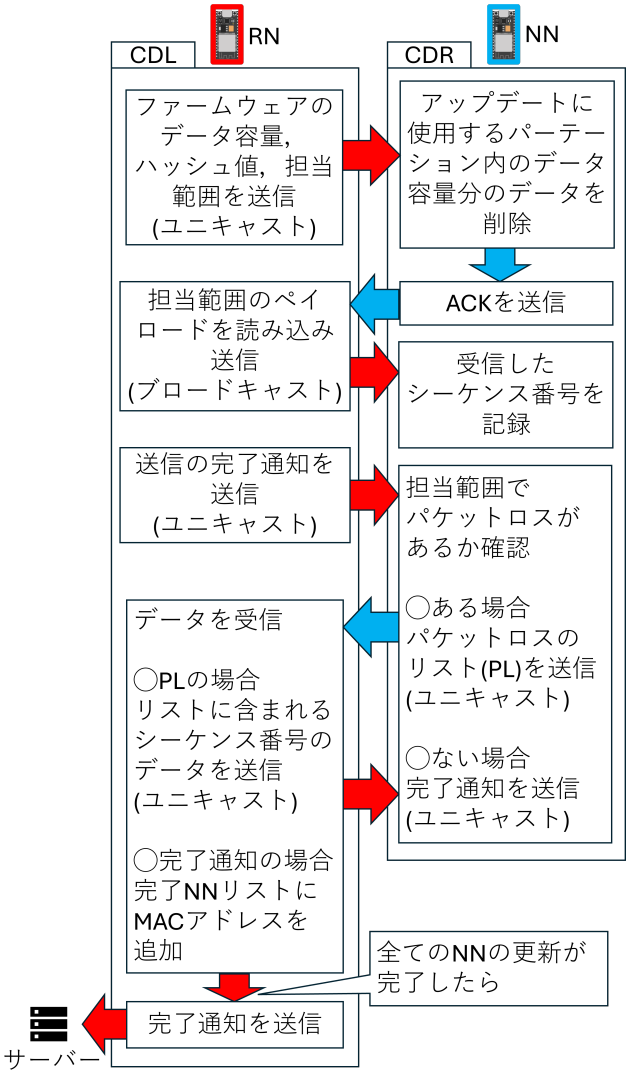

図 6: RN の NN へのデータ送信から更新完了までの処理の流れ

て送信した後，送信が完了したことを通知するパケットを全ての NN にユニキャストで送信する．このパケットを受信した NN は，パケットを送信してきた RN が送信を担当するシーケンス番号の範囲の受信したシーケンス番号の記録から，ロスしたパケットのシーケンス番号を探し，パケットロスリスト (PL) に記録する．パケットロスがある場合，PL をその範囲を担当する RN に送信する．PL を受信した RN は，ユニキャストで PL に含まれるシーケンス番号のパケットを送信する．もし，パケットロスがない場合，その範囲を担当する RN に完了通知を送信する．完了通知を受信した RN は完了 NN リストに，完了通知の送信元 NN の MAC アドレスを追加する．もし，全ての NN から完了通知を受信したら，サーバーに完了通知を送信する．

ユニキャストの処理

ESP-NOW には再送信の処理が無いため，ユニキャストを行うにあたり，ACK の返答と，ACK が返答されないまたは届かなかった場合の再送信の機能を実装した．図??に RN から NN に対してユニキャストを行う際の処理の流れの例を示す．

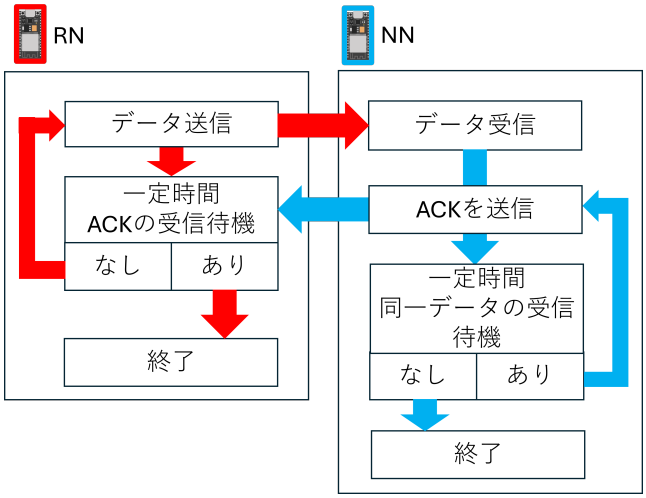

図 7: RN から NN に対してユニキャストを行う際の処理の流れの例

まず，RN は NN にデータを送信する．その後 NN から ACK の返答を待つ．もし，1 秒経っても ACK が受信できなければ再送信を行う．1 秒に設定した理由は，TCP の初期の再送信の間隔が 1 秒だからである [19]．もし，1 秒以内に ACK を受信したら，通信を終了する．データを受信した NN は RN に ACK を返す．NN は RN からデータを受信したら ACK を返す．その後，同一のデータが再送信されるかを確認するために 5 秒間，受信待機をする．5 秒間待機する理由は，1 秒間隔の再送信が何回かロスして，ACK が送信先に到達したと判定を誤らないようにするためである．もし同一のデータを受信したら，ACK を再送信する．受信しなかった場合，通信を終了する．この処理は送受信が RN と NN で入れ替わった場合も同様の処理を行う．

5 評価実験

評価方法は，1 台のノードがデータを配布してファームウェアアップデートを行なった場合と，提案ソフトウェアを使用した場合のノードの消費電力を比較する．1 台のノードがデータを配布する場合を提案なしとする．提案なしのデータフローを図 8に示す．

提案なしの場合，RN の中から最もバッテリー残量が多い RN がデータの配布を担当する．図 8では最もバッテリー残量が多い RN2 がデータの配布を担当している．提案なしとありでは，データをブロードキャストで送信した後，ユニキャストでパケットロスを確認して再送信を行う処理は共通している．提案なしではブロードキャスト時のデー

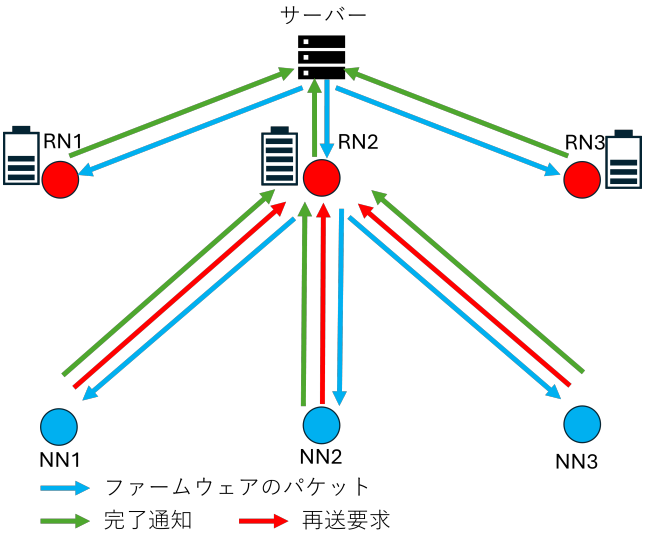

図 8: 提案なしのデータフロー

タ送信の間隔は 200[ms] であり，ランダムな時間だけ待つ処理は行わない．RN は直接サーバーからファームウェアを受信する．その後，自身に送信を担当するパケットが無い場合または，NN へ担当するパケットを送信した後に，全ての NN から完了通知を受信したらサーバーに完了通知を送信する．各ノードで最初のパケットを受信した時点から完了通知を送信するまでの間の消費電力を測定する．消費電力の計測間隔は 10[ms] とする．送信するデータとして 750[KB] のバイナリファイルをファームウェアに見立てて送信する [20].

実験環境

図 9に中継機と RN3 台と NN3 台を実際に配置した時の写真を示す．実験は東京工科大学八王子キャンパスの片柳記念ホール前の広場で行った．図 9の赤丸で囲まれた 3 台のノードが RN で，青丸で囲まれた 3 台のノードが NN，緑の丸で囲まれた機器が中継機である．ノードは隣接するノード間の距離を約 2[m] 離して設置した．

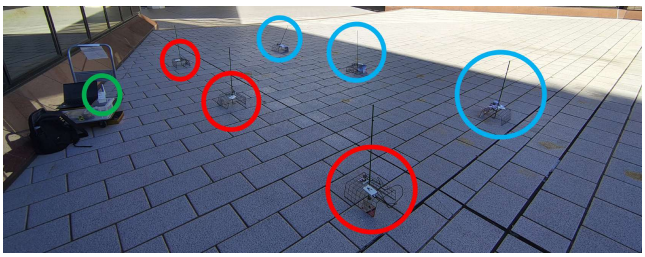

図 9: 中継機と RN3 台と NN3 台を実際に配置した時の写真

図 10に設置したノードの写真を示す．図 10の赤丸で囲まれているものが ESP32 である，青丸で囲まれているものがモバイルバッテリーである．モバイルバッテリーは 4000[mAh] のものを使用している．ESP32 は地面から約 115[mm] 離れている．

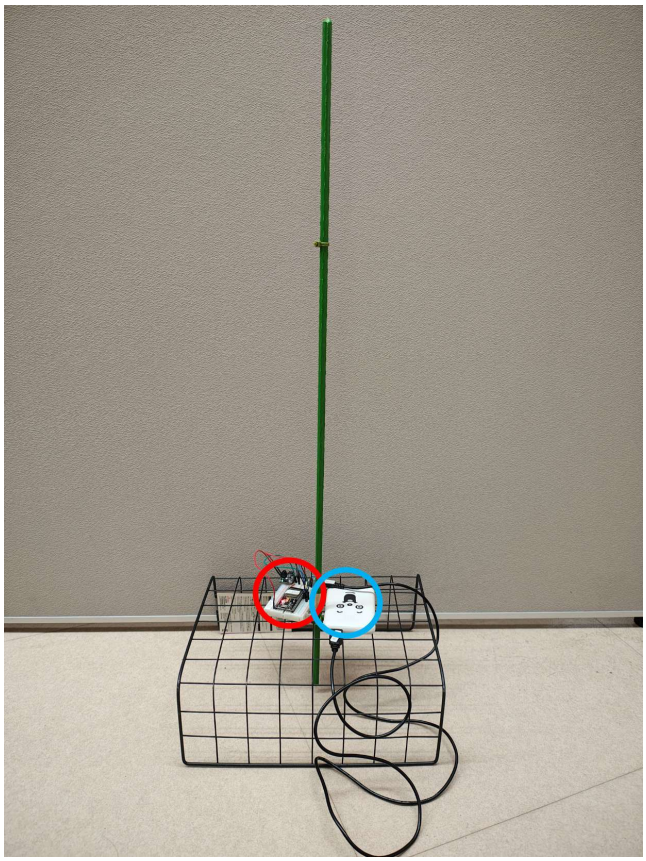

図 10: 設置したノードの写真

図 11に実験環境の構成図を示す．構成図の構成要素について項目に分けて説明する．

ノード

ノードには，Espressif Systems 社のマイクロコントローラーである ESP32 を使用した．ESP32 には事前に，Micropython のサイトにある OTA サポート版の ESP32 用のファームウェアである「ESP32_GENERIC-OTA-20250809-v1.26.0.bin」をインストールした．また，電流と電圧を計測することができる INA219 が接続されている．今回の実験では 6 台を用意し，提案ありの場合は，RN を 3 台，NN を 3 台とした．ノードは縦横の間隔を約 2m 離して設置した．

サーバー

サーバーは, Broadcom 社の VMware 製品である VMware ESXi 上に仮想マシンを作成し，Ubuntu24.04LTS をインストールして，Python 3.12.3 の仮想環境を構築したものを使用した．サーバーでは Flask サーバーを動作させた．

また，更新に使用するファームウェアのダミーデータとして「firmware_750k.bin」を置いた．

ルーター

Wi-Fi ルーターは，ASUS 社の「ASUS TUF-AX5400」を使用した．サーバーとルーターは 1Gbps 対応の LAN ケーブルで接続されている．

中継機

ルーターの中継機として，TP-Link 社の「RE605X AX1800 Wi-Fi 6 中継器」を使用した．ルーターと中継機は 2.4GHz 帯の Wi-Fi で無線接続されている．

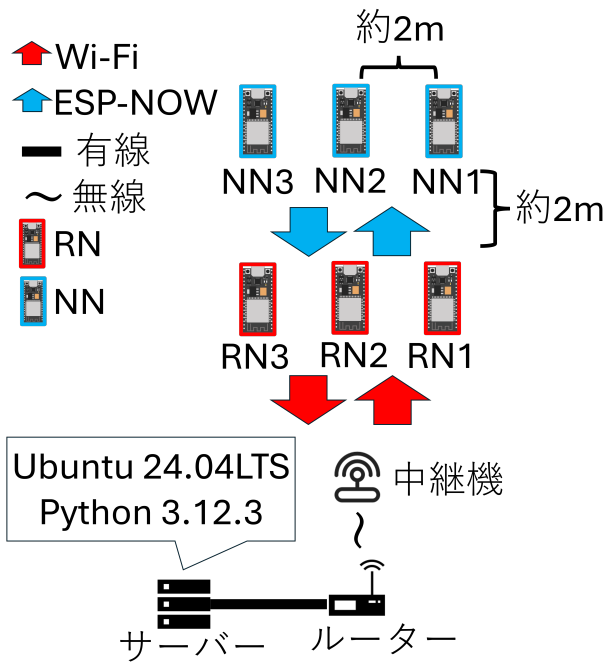

図 11: 実験環境の構成図

実験結果と分析

提案ありとなしの実験を行った際の初期状態について表 2に示す．

表 2: 提案ありとなしの初期状態

実験	ノード	バッテリー残量 (%)	送信パケット数
提案あり	RN1	99	1284
	RN2	98	1272
	RN3	99	1284
提案なし	RN1	99	0
	RN2	99	3840
	RN3	99	0

今回の実験で送信するファームウェアは 3840 パケットに分割された．提案なしの時，RN2 が NN へのデータの配布を担当した．また，提案ありのバッテリー残量は RN1 が 99[%]，RN2 が 98[%]，RN3 が 99[%] であった．各 RN に割り当てられたパケット数は，RN1 が 1284 個，RN2 が 1272 個，RN3 が 1284 個だった．

図 12にファームウェアの配布をノード 1 台で行った場合と提案手法で行った場合の各ノードの消費電力を示す．縦軸は更新処理中における消費電力で単位は [mWh] である．

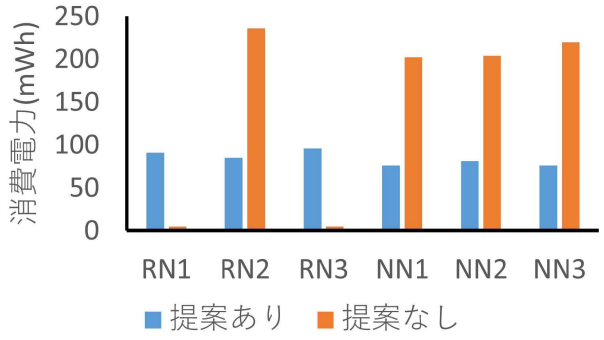

図 12: ファームウェアの配布をノード 1 台で行った場合と提案手法で行った場合の各ノードの消費電力

提案なしの各ノードの消費電力は，RN1 が約 5[mWh]，RN2 が約 236[mWh]，RN3 が約 5[mWh]，NN1 が約 202[mWh], NN2 が約 204[mWh], NN3 が約 220[mWh] だった．RN2 が最も消費電力が高く，RN1 と RN3 が最も消費電力が低くなった．RN2 の消費電力が最も高くなった理由は，全ての NN の更新が完了したことを確認してから，最後に更新を完了するためである．RN1 と RN3 の消費電力が最も低くなっている理由は，サーバーからファームウェアを受信した後，NN へのデータ送信は行わず，即座に更新を完了するためである．提案なしの標準偏差は約 99[mWh] であった．また，全てのノードの消費電力の総和は約 872[mWh] である．提案ありの各ノードの消費電力は RN1 が約 91[mWh]，RN2 が約 85[mWh]，RN3 が約 96[mWh]，NN1 が約 76[mWh]，NN2 が約 81[mWh]，NN3 が約 76[mWh] だった．提案ありの標準偏差は約 7[mWh] だった．全てのノードの消費電力の総和は約 505[mWh] である．提案なしとありを比較すると，標準偏差が提案により約 92[mWh] 減少したことがわかる．また，図 12のグラフを見ると，提案なしは消費電力が RN2 に偏っているのに対し，提案ありではノード全体の差が小さくなっていることがわかる．また，全てのノードの消費電力の総和は提案により，約 367[mWh] 削減することができた．

図 13にファームウェアの配布をノード 1 台で行った場合と提案手法で行った場合の各ノードの更新時間を示す．縦

軸は更新にかかった時間で単位は秒である．

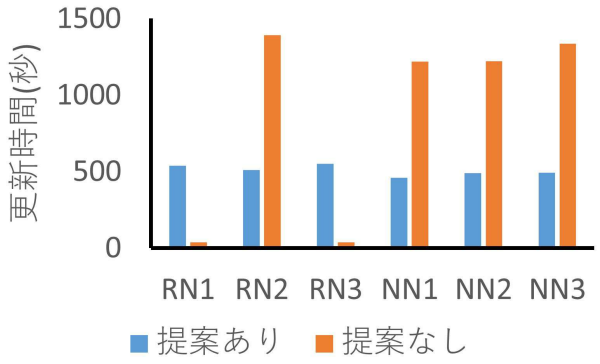

図 13: ファームウェアの配布をノード 1 台で行った場合と提案手法で行った場合の各ノードの更新時間

提案なしの各ノードの更新時間は，RN1 が約 38 秒，RN2 が約 1392 秒，RN3 が約 38 秒，NN1 が約 1220 秒，NN2 が約 1221 秒，NN3 が約 1337 秒であった．RN2 が最も更新時間が長く，RN1 と RN3 が最も更新時間が短かった．RN2 の更新時間が最も長くなった理由は，全ての NN の更新が完了したことを確認してから，最後に更新を完了するためである．RN1 と RN3 の更新時間が最も短くなった理由は，サーバーからファームウェアを受信した後，NN へのデータ送信は行わず，即座に更新を完了するためである．提案なしの更新時間の標準偏差は約 594 秒である．提案ありの各ノードの更新時間は，RN1 が約 538 秒，RN2 が約 509 秒，RN3 が約 550 秒，NN1 が約 460 秒，NN2 が約 490 秒，NN3 が約 493 秒であった．RN3 が最も更新時間が長く，NN1 が最も更新時間が短かった．RN3 の更新時間が最も長い理由は，RN の中で最も多くパケットを割り当てられたことと，RN の中で最も遅くサーバーからファームウェアをダウンロードしたからである．提案なしの更新時間の標準偏差は約 30 秒である．提案なしと提案ありを比較すると，RN1 と RN3 を除いて更新時間を短縮できたことがわかる．しかし，RN1 と RN3 は他のノードと同程度まで更新時間が長くなっている．RN1 と RN3 以外で短縮できた時間は，RN2 で約 883 秒，NN1 で約 760 秒，NN2 で約 731 秒，NN3 で約 844 秒であった．また，標準偏差は提案により約 564 秒減少した．これらのことから，提案手法によりクラスター全体の更新時間が短くなり，ノードごとの更新時間の差を小さくすることができることはわかった．

図 14にファームウェアの配布をノード 1 台で行った場合と提案手法で行った場合の NN のファームウェアパケットのロス率を示す．縦軸はパケットロス率で単位はパーセントである．

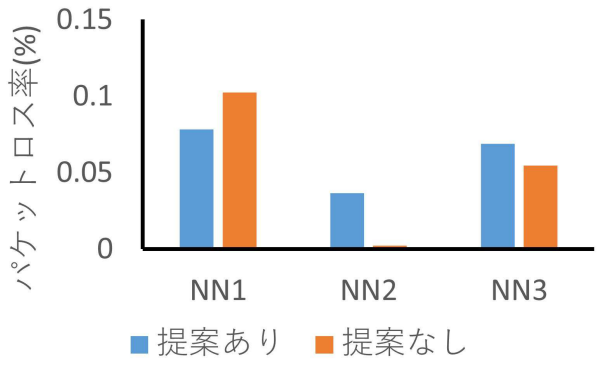

図 14: ファームウェアの配布をノード 1 台で行った場合と提案手法で行った場合の NN のファームウェアパケットのロス率

提案なしの NN のパケットロス率は，NN1 が約 0.1[%]，NN2 が約 0.002[%]，NN3 が約 0.05[%] である．NN1 が最もパケットロス率が高くなり，NN2 が最もパケットロス率が低くなった．この様な結果になった理由として通信距離の違いがある．図 14を見ると，NN2 のパケットロス率が NN3 と比べても極端に低くなっていることがわかる．この時，データの配布を行っているノードは RN2 であり，NN2 との距離は約 2[m] で NN1 と NN3 は約 2.8[m] になり，NN2 よりも NN1 と NN3 の方が通信距離が長くなる．提案ありのパケットロス率は，NN1 が約 0.07[%]，NN2 が約 0.03[%]，NN3 が約 0.06[%] である．NN1 が最もパケットロス率が高くなり，NN2 が最もパケットロス率が低くなった．この様な結果になった理由は，提案なしと同様に通信距離の違いがある．提案なしとの相違点として，通信距離がより長くなっている場合がある．NN1 は RN1 との通信距離が約 2[m]，RN2 との通信距離が約 2.8[m]，RN3 との通信距離が約 3.4[m] である．NN2 は RN1 との通信距離が約 2.8[m]，RN2 との通信距離が約 2[m]，RN3 との通信距離が約 2.4[m] である．NN3 は RN1 との通信距離が約 3.4[m]，RN2 との通信距離が約 2.4[m]，RN3 との通信距離が約 3.4[m] である．このように，各 NN で RN との通信距離が異なり，提案なしよりも通信距離が長くなることがある．また，NN2 は他の NN と比べて通信距離が短い．提案なしと提案ありを比較すると，NN2 と NN3 は提案ありの方がパケットロス率が高くなっている．このことから，提案によりパケットロス率が高くなることが分かった．

消費電力，更新時間，パケットロス率の結果から，提案手法により，消費電力と更新時間は低くなるが，パケットロス率が高くなることがわかった．

6 議論

本提案では，パケットの衝突回避のために 200[ms] から 300[ms] の間のランダムな時間待ってからデータを送信している．現状では 200[ms] から 300[ms] に設定している根拠が乏しく，適切な値に設定できていない．この待機時間の範囲を適切に設定するにあたり，既存手法である CSMA/CA に則った値の決め方にすることが良いと考える．CSMA/CA では待機時間が 0 からコンテンションウィンドウまでのランダムな値とスロット時間の積によって求められる．コンテンションウィンドウの最小値は 15 で，最大値は 1023 である．コンテンションウィンドウは，再送信の回数に応じて倍増させる [21]．スロット時間に関しては，受信側の取りこぼしを避けるためにデータを受信してから次の受信待機に移行するまでの時間を計測し，それをスロット時間とすることが良いと考える．

本提案では，ユニキャスト時の ACK の待ち時間を 1 秒，ACK 送信後の再送信の待ち時間を 5 秒にしている．現状この時間に設定している根拠が乏しい．この値を適切に設定するにあたり，既存手法の TCP を参考にすることが良いと考える．理由として，TCP は無線通信において広く使用されており，信頼性の高い再送信の機能を持つプロトコルだからである [22]．TCP では，RTT が計測されるまでは待ち時間を 1 秒にする．その後，RTT を計測できたら，その RTT を $SRTT$，RTT を二分の一した値を $RTTVAR$ として，$SRTT$ にクロック粒度 G と，$RTTVAR$ と定数 K の積を比較し大きい方を加算した値を RTO としている．この時，G の値は 100ms 以下が良いとされている．また K はデフォルトで 4 である．RTO を求める式を式 (6) に示す [19].

$$RTO = SRTT + max(G, K \times RTTVAR) \qquad (6)$$

本提案では，パケットの衝突を回避するために，データを送信するときに，ランダムな時間待っている．しかし，これだけではパケットの衝突を完全に無くすことは不可能である．よりパケットが衝突する可能性が低い手法に，Time Division Multiple Access(TDMA) がある．これは，各ノードに送信タイミングを割り当て，順番にデータを送信させることで，同一チャネル上での通信衝突を防止する方式である．これを導入することによりパケット衝突を避けることができ，再送信頻度が削減されることによって，消費電力の削減になる [23–25].

また，再送信の削減の手法として，冗長符号化を使用する手法がある．これは，送信側で送信する元データに対して冗長情報を付与し，受信側で冗長情報を使用して誤りの検知，訂正をする手法である．これを行うことで，送信される総パケット数の内の一定数だけ受信できれば，誤り訂正により元のデータを復元できる．これにより，再送信頻度を削減できる [26].

本提案は前提条件として「サーバーは全ノードのバッテリー残量を定期的に受信している」という条件がある．これにより，全ノードはバッテリー残量を定期的にサーバーに送信する必要があり，これにより，送信するデータ量が増え，この処理自体がオーバーヘッドになる．この条件をなくす手法として，アップデート時に，RN のバッテリー残量が前回送信時から変化した場合にのみ，その値を送信する手法を提案する．現状の提案ではバッテリー残量が正の整数になることを前提にしているため，値が変化した時のみバッテリー残量を送信する方法により，バッテリーの消費が 1[%] 未満ならバッテリー残量を送信しなくて良くなる．また，ファームウェアアップデート時のみ送信することにより，定期的にバッテリー残量を送信することがなくなりオーバヘッドが解消される．

本提案の他の前提条件として「全てのノードとサーバーと通信が可能なノードを RN とする」という条件がある．全てのノードと通信が可能という条件は，無線通信において障害物と電波干渉の影響，通信距離による電波の減衰から条件を満たすノードは限られるため現実的ではない．そのため，実際には一部のノードとしか通信ができないノードが出てくる．この条件をなくす手法として，サーバーと直接通信が可能なノードを RN として，RN と通信が可能な NN を経由して RN と通信できない NN にデータを送信する手法を提案する．この提案では，RN が直接通信できない NN のリストを保持している．また，NN は直接通信が可能なノードのリストを保持している．RN は通信可能な NN にその RN と通信ができない NN と通信が可能であるかを問い合わせる．問い合わせを受信した NN は RN に通信の可否と通信可能な場合は自身のバッテリー残量を送信する．全ての NN から問い合わせの応答を受信した RN は，バッテリー残量が最も多い NN を経由して，RN が直接通信できない NN へデータを送信する．これにより，RN の通信可能な距離に関係なく割り当てられたパケットをノード全体に行き渡らせることができる．

本提案の他の前提条件として「RN と NN の数は固定である」という条件がある．しかし，実際に IoT 機器を運用する場合，バッテリー切れ，機器の故障，機器の追加によりノードの数が変動することがある．この条件をなくす手法として，ファームウェア更新時に各ノードが通信可能なノードを確認する手法を提案する．この手法では，まずサーバーがノードが稼働しているかを確認するパケットをブロードキャストで送信する．それを受信した RN または新規のノードはサーバーに ACK を返す．直近で通信できた RN からの ACK を受信できなかった場合，ユニキャストでパケットを送信する．RN はノードが稼働しているか

を確認するパケットをブロードキャストで送信した後，直近で通信ができた NN で ACK を受信できなかった NN に対してユニキャストで同じパケットを送信する．これを受信した NN または新規のノードは RN に ACK を返す．その後，NN はブロードキャストでノードが稼働しているかを確認するパケットを送信した後，直近で通信ができた NN で ACK が受信できなかった NN に対して同じパケットをユニキャストで送信する．これを受信した NN または新規のノードは送信元の NN に ACK を返す．ACK が帰ってきたノードに関しては稼働しているノードとしてその後の処理を進める．また，新規のノードはサーバーからのパケットを受信できれば自身を RN とし，サーバーからパケットを受け取れず RN または NN からパケットを受信できた場合は自身を NN とする．この手法により，ノードの数に変動があっても正常に提案手法が動作する．

7 おわりに

課題は，ファームウェア更新時の更新データの配布を 1 台の CH が担当して消費電力が増加することにより，稼働時間が短くなることである．提案として，データの配布を担当できるノード (以後 RN とする) の台数で送信パケットの総数を等分割し，バッテリー残量の偏差から各 RN が送信するパケット数を分配する手法を提案する．実験では ESP32 を 6 台用意し，うち 3 台を RN として各ノードのファームウェア更新時の消費電力，更新時間，パケットロス率を計測した．比較はノード 1 台がデータの配布を担当した場合と提案手法を適用した場合を比較した．この時，送信したデータはファームウェアに見立てた 750[KB] のバイナリファイルである．結果として，クラスターの更新時間は送信を担当するノードが 1 台の場合は約 1392 秒，提案手法の場合は約 550 秒で約 883 秒短縮できた．消費電力の標準偏差は送信を担当するノードが 1 台の場合は約 99[mWh]，提案手法の場合は約 7[mWh] で約 92[mWh] 減少した．全てのノードの消費電力の総和は送信を担当するノードが 1 台の場合で約 872[mWh]，提案手法で約 505[mWh] であり，約 367[mWh] 削減できた．これは，提案手法によりクラスターの更新完了までの時間が短縮された結果，消費電力が減少した削減量の総和が，消費電力が増加した増加量の総和を上回ったためである．RN から NN へファームウェアのパケットを送信した際のパケットロス率に関しては，3 台の NN の内，2 台の NN で最大で約 0.028[%]，最小で約 0.01[%] 高くなった．

参考文献

[1] Duguma, A. L. and Bai, X.: How the internet of things technology improves agricultural efficiency, *Artificial Intelligence Review*, Vol. 58, No. 2, p. 63 (2024).

[2] Jawad, H. M., Nordin, R., Gharghan, S. K., Jawad, A. M. and Ismail, M.: Energy-Efficient Wireless Sensor Networks for Precision Agriculture: A Review, *Sensors*, Vol. 17, No. 8 (online), DOI: 10.3390/s17081781 (2017).

[3] Navarro, M., Davis, T. W., Villalba, G., Li, Y., Zhong, X., Erratt, N., Liang, X. and Liang, Y.: Towards Long-Term Multi-Hop WSN Deployments for Environmental Monitoring: An Experimental Network Evaluation, *Journal of Sensor and Actuator Networks*, Vol. 3, No. 4, pp. 297–330 (online), DOI: 10.3390/jsan3040297 (2014).

[4] van den Berg, H., Roijers, F., Castro, M., Gomez, C., Paradells, J., Staub, T., de Oliveira, R., Karlsson, J., Avallone, S., Hurni, P. et al.: Multi-hop wireless networks, *chap*, Vol. 5, pp. 1–68 (2009).

[5] Leenders, G., Callebaut, G., Ottoy, G., Van der Perre, L. and De Strycker, L.: An Energy-Efficient LoRa Multi-Hop Protocol through Preamble Sampling for Remote Sensing, *Sensors*, Vol. 23, No. 11 (online), DOI: 10.3390/s23114994 (2023).

[6] Elmonser, M., Alaerjan, A., Jabeur, R., Chikha, H. B. and Attia, R.: Enhancing energy distribution through dynamic multi-hop for heterogeneous WSNs dedicated to IoT-enabled smart grids, *Scientific Reports*, Vol. 14, No. 1, p. 30690 (2024).

[7] Kaur, S., Kour, S. and Singh, M.: EECH HEED an adaptive hybrid clustering protocol for energy efficient soil monitoring in heterogeneous wireless sensor networks, *Scientific Reports*, Vol. 15, No. 1, p. 35548 (2025).

[8] Liu, X.: A Survey on Clustering Routing Protocols in Wireless Sensor Networks, *Sensors*, Vol. 12, No. 8, pp. 11113–11153 (online), DOI: 10.3390/s120811113 (2012).

[9] Pachlor, R., Shrimankar, D., Nagwanshi, K. K. and Paliwal, M.: Cluster-Head Rotation Approaches in Sensor Networks: A Review (2022).

[10] Heinzelman, W., Chandrakasan, A. and Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks, *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pp. 10 pp. vol.2– (online), DOI: 10.1109/HICSS.2000.926982 (2000).

[11] Liu, J.-L. and Ravishankar, C. V.: LEACH-GA: Genetic algorithm-based energy-efficient adaptive clustering protocol for wireless sensor networks, *International Journal of Machine Learning and Computing*, Vol. 1, No. 1, p. 79 (2011).

[12] Qing, L., Zhu, Q. and Wang, M.: Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks, *Computer Communications*, Vol. 29, No. 12, pp. 2230–2237 (online), DOI: https://doi.org/10.1016/j.comcom.2006.02.017 (2006).

[13] Younis, O. and Fahmy, S.: Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach, *IEEE INFOCOM 2004*, Vol. 1, p. 640 (online), DOI: 10.1109/INFCOM.2004.1354534 (2004).

[14] Chien, H.-Y. and Wang, N.-Z.: A Novel MQTT 5.0-Based Over-the-Air Updating Architecture Facilitating Stronger Security, *Electronics*, Vol. 11, No. 23 (online), DOI: 10.3390/electronics11233899 (2022).

[15] Jeong, H. and Ahn, B.: A Software Update Method using Clustering WSNs, *Proceedings of the International Conference on Wireless Networks (ICWN)*, The Steering Committee of The World Congress in Computer Science, Computer ⋯, p. 1 (2014).

[16] Wang, X., Wang, J. and Xu, Y.: Data dissemination in wireless sensor networks with network coding, *EURASIP*

Journal on Wireless Communications and networking, Vol. 2010, No. 1, p. 465915 (2010).

[17] Yu, J. and Zhang, X.: A Cross-Layer Wireless Sensor Network Energy-Efficient Communication Protocol for Real-Time Monitoring of the Long-Distance Electric Transmission Lines, *Journal of Sensors*, Vol. 2015, No. 1, p. 515247 (online), DOI: https://doi.org/10.1155/2015/515247 (2015).

[18] Lewandowski, M. and Płaczek, B.: A Cluster Head Selection Algorithm for Extending Last Node Lifetime in Wireless Sensor Networks, *Sensors*, Vol. 25, No. 11 (online), DOI: 10.3390/s25113466 (2025).

[19] Sargent, M., Chu, J., Paxson, D. V. and Allman, M.: Computing TCP's Retransmission Timer, RFC 6298 (2011).

[20] Becker, B., Oberli, C., Zobel, J., Steinmetz, R. and Meuser, T.: ESP-NOW Performance in Outdoor Environments: Field Experiments and Analysis, *2025 20th Wireless On-Demand Network Systems and Services Conference (WONS)*, pp. 1–8 (2025).

[21] : IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379 (online), DOI: 10.1109/IEEESTD.2021.9363693 (2021).

[22] Ogola, A.: A Survey Paper on TCP Congestion Control Algorithms (2024).

[23] Samara, G.: Wireless Sensor Network MAC Energy - efficiency Protocols: A Survey, *2020 21st International Arab Conference on Information Technology (ACIT)*, pp. 1–5 (online), DOI: 10.1109/ACIT50332.2020.9300065 (2020).

[24] Sinde, R., Begum, F., NJAU, K. and Kaijage, S.: Lifetime improved WSN using enhanced-LEACH and angle sector-based energy-aware TDMA scheduling, *Cogent Engineering*, Vol. 7 (online), DOI: 10.1080/23311916.2020.1795049 (2020).

[25] Gajjar, S., Choksi, N., Sarkar, M. and Dasgupta, K.: LEFT: A latency and energy efficient flexible TDMA protocol for wireless sensor networks, *International Journal of Computer Network and Information Security*, Vol. 7, No. 2, p. 1 (2015).

[26] Du, W., Liando, J. C., Zhang, H. and Li, M.: Pando: Fountain-Enabled Fast Data Dissemination With Constructive Interference, *IEEE/ACM Trans. Netw.*, Vol. 25, No. 2, p. 820–833 (online), DOI: 10.1109/TNET.2016.2614707 (2017).

