通信障害時に迂回経路でデータ転送をする方式

柴智瀚[1,a)]　串田 高幸[1]

概要：サーバとデバイスが通信をしている時に通信障害が発生し, データを引き続き送信するのが難しくなって, データが時々無くなってしまう課題がある. このような問題を解決するために, デバイス付近にあるサーバと繋がっている別のデバイスを探索し, BLE(Bluetooth Low Energy) を用いて一時的に近隣のデバイス経由して継続的にデータを送信する方法を提案する. この提案によって, IoT デバイスが自分自身とサーバへの接続状態を監視し, 通信障害が起こった際に周囲に転送リクエストをブロードキャストする. リクエストを受信したデバイスが相互の BLE 通信状況を評価し, プリセット値を超え, 経由ノードとしての通信時間が最も短いデバイスを介してデータの転送を開始する. BLE 通信状況に対する評価方法は, IoT デバイス間のパケット配信率 (Packet Delivery Ratio:PDR) を評価することである. 提案に対する評価方法は, IoT デバイスが直接サーバへ送信する時間と転送ノードを経由して送信する時間を比較して結論を出す.

1. はじめに

背景

近年では, IoT(Internet of Things) が大きな研究の注目を集めている. 将来のインターネットの一部と見されており, 何十億の知能な「モノ」で構成される規模になると見込まれている [1]. この膨大な数を持っているインターネットを構築する際には無線通信が不可欠な存在だと考えられている. インストールコストを削減できることは, 有線通信に対する無線通信の主な利点の一つである [2]. 徐々に話題になってくる IoT では常に百台, 千台以上のデバイスが設置されている. この場合, 千台のデバイスを有線で繋がると相当な費用がかかると予想される. したがって既存の IoT デバイスは無線通信を使ってデータの送受信をやっているケースが大多数である. IoT は機器の監視手段として多く扱われて, 特定の場所に IoT デバイスを設けていれば, この場所の温度, 湿度をリアルタイムで監視すうことが可能である, サーバで監視したデータを取得して分析を行い, 異変と異常を検知することで問題を早めに解決するのが大きな特徴である.

既存の無線通信は多種多様であり, Wi-Fi, Zigbee と Bluetooth がよく使われている. 特に Bluetooth は低消費電力, 低コスト, 高可用性と高精度のため室内の超低電力短距離通信 IoT 配置に揺るぎない位置を占めている [3]. Bluetooth は低電力通信要件を満たし, データ伝送のエネルギー効率の方が Wi-Fi より 30%の増幅が見られており, ほとんどのモバイルデバイスで利用できると言った様々な利点を持っている [4]. 短距離通信と言ったら最初に頭に浮かぶのは必ず Bluetooth といっても過言ではない. しかし, その複雑な検出メカニズムのため, IoT の膨大なデバイス数に向いていない [5]. これらの制限に直面して, BLE は IoT 主導の通信方法として従来の BL に徐々に取って代わった [6]. BLE は, ペアリングを排除し, Bluetooth に固有の複雑な検出を簡素化すると同時に, 短いデータ交換をサポートするように設計されていた [7][8]. 本稿では BLE を用いて, サーバとの通信障害があるデバイスに対する通信を維持させる迂回経路を設計する.

課題

IoT デバイスとサーバへの通信が電波干渉や金属による通信障害が時々発生する. また移動中のデバイスが通信範囲外に辿り着いて, 従来の Wi-Fi と接続できなくてデータを送信できないジレンマに陥ってしまう. 通信手段によるデータが正確にサーバへ伝送できたかどうかがわからない. できない場合はデータの損失とデータの信頼性が低下してしまう恐れがある. IoT とサーバへの送受信が一旦切れたら, リアルタイムでデータを送ったり, データを分析したりすることも不可能になってしまう. これらの問題を解決するために, 次の課題を直面している. 迂回回路の時にどの経路が最良なのかを検討しなければならない. 図 1 は経路選択問題を示す. このような複数の IoT デバイスが存在する時, ESP32A が ESP32C と ESP32D を経由してもいいが, ESP32B と ESP32C のような 2hop でデータを転送しても

[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
[a)] C0118185

いい．しかし実際の状況では唯一のリンクで通信を行うのが一般的なので，どの経路が一番良いのかが決まらなければならない．

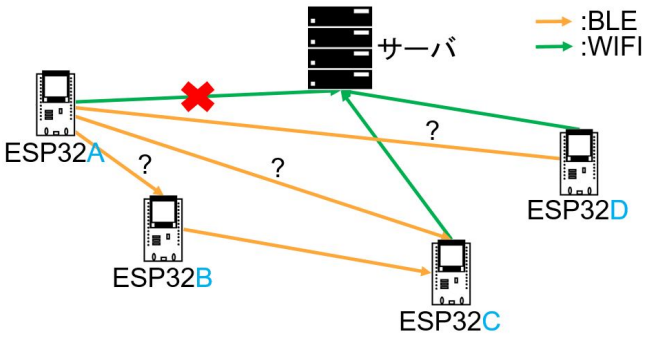

図 1　経路選択問題

今回は BLE のブロードキャストを使い，通信範囲内における全てのデバイスにリクエストを発信することができる．これで経路選択問題を乗り超えてサーバと自動的に迂回経由路で繋がることが期待できる．概念図は図 2 で示す．解決方法としてはデバイス B のような既にサーバと通信しているノードが通信障害があるデバイス A のリクエストを受信した後に，各デバイス B が通信状況を評価する．最初に設定された値を評価の基準として，それを上回ると，デバイス B が転送ノードとして通信障害があるデバイス A の代わりにデータを転送するスキームを想定している．

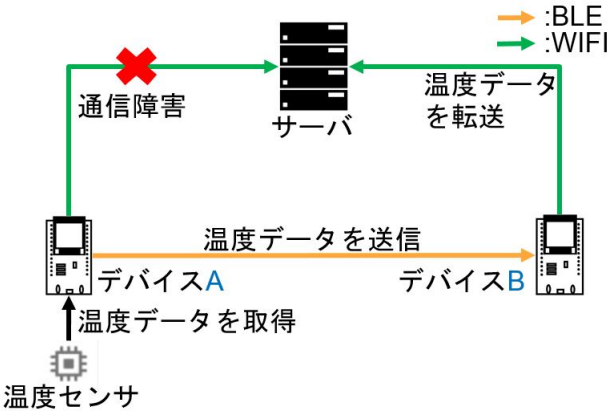

図 2　概念図

各章の概要

第 2 章以下の各章の概要について説明する．第 2 章では本稿の関連研究について紹介する．第 3 章では前述した課題を解決する方法を具体的に提案する．第 4 章では実際の実装と実験を紹介する．第 5 章では本稿が提案した方法の評価と分析である．第 6 章では本稿が出来なかった点と不足のところを議論する．最後に第 7 章で結論を出す．

2. 関連研究

この章では既存研究の内容を借りてどのような位置づけにあるのか，また，どのような不足があるのか関連研究を交えて述べる．

Huang-Chen Lee らは LoRA PHY メッシュネットワークを基づいて，広範囲にある IoT センサーを監視するシステムを構築した．LoRA GW を一つだけでも 800 × 600 のキャンパスにおいて各デバイスのデータ送信頻度，信号強度をゲートウェイで見えるようにした．特に通信の質が遠ければ遠いほど悪くなると指摘された [9]．Huang-Chen Lee らの研究はデータを遠距離でどのように効率の高いデータを伝送するのに焦点を当てたが，通信障害があるデバイスの対処法は検討されていなかった．

経路選択問題の解決方法について Sanjit Biswas らは ExOR というアドホックモードを使っており広範囲にある実験を行った．この研究では，マルチホップワイヤレスネットワークにおける大規模なユニキャスト送信のスループットを向上させることができる統合型ルーティング方式を目標とする．38 個ノードの 802.11b テストベッド (test-bed) である実装の測定は，ExOR が従来のルーティングと比較してほとんどのノードペアのスループットを向上させることを示している [10]．SanjitBiswas らはアドホックモードを使ってネットワークのノード同士の通信方法を提案したが．アドホックモードを使えない状況では使えないのが課題である．

同じく通信範囲から抜けた IoT デバイスについて研究を行った Syed Rafiul Hussain らは，複数 BLE ゲートウェイが存在する時に，動いている IoT デバイスが一旦 BLE ゲートウェイの通信範囲から抜けたら近所の他の BLE ゲートウェイにシームレスに接続させる方法を提案した [11]．Syed Rafiul Hussain らの提案において，ゲートウェイと IoT デバイス間のペアリングごとにすべてのゲートウェイを同期するには，かなりの時間と帯域幅が必要であり，通信のオーバーヘッドが大きくなる問題がある．

IoT デバイス間のコミュニケーション問題を解決ために Paul らは SMRP(Secure Multi-Hop Routing Protocol) というマルチホップルーティングプロトコルで三つの IoT デバイス (PhidgetSBC 1072 single board computers: SBC) を持って実験を行った [12]．実験に関しては，まず IoT デバイスで構築されたネットワークの安全性をテストした，次に RTD(round-trip-delay) と帯域幅のパフォーマンスを測定することにより，OLSR[13] アルゴリズムをもとに SMRP との比較実験を行った．Paul らの研究には前述のアドホックモードを使ったと同じく，アドホックモードが使えなければ実行できない課題がある．

エッジデバイスの通信状況に対する検測及び評価対策は Bharath Sudharsan らの論文で研究を行った [14]．この論

文には BLE と Wi-Fi 共に RSS の測定データベースを採用したが, それぞれ各自の参照データベースが異なっている. Wi-Fi の方は kthrss データベースを選択し, BLE の方は BLE RSS データベースを選択した. Bharath Sudharsan らの論文により彼らのモデルで Wi-Fi と BLE の RSS をそれぞれ 94.14%と 92.55%で予測できる. 本稿との違いは評価指標である. Bharath Sudharsan らの論文は RSS で検測したが, RSS はメトリックとしてどれほどの確信度があるのか説明されていなかった.

3. 提案

提案としては通信障害があるデバイスが BLE 通信範囲にある他のデバイスを経由して、最終に取得したデータをサーバへ転送する. このため, 各自 IoT デバイスに送信用プログラム, デバイスとサーバへの Wi-Fi 通信状況を監視プログラムとデバイス相互の BLE 通信状況を評価するプログラムが内蔵されなければならない. 図 3 はこれを示す. 一般的にデータを取得するデバイスを伝送デバイスとする. Wi-Fi 通信状況を監視するプログラムは伝送デバイスとサーバの通信障害に異常を検知していない場合は, いつもの通り何の変更もないままである. 一方, 伝送デバイスがサーバとの通信状況が異常が出る場合は, BLE モードを使い, BLE 通信範囲内における転送デバイスを探して, 転送デバイスを経由ノードとして通信ルートを変更させる. 図 4 を参考する. これで, 通信障害を受け入れたデバイスにデータ送信を維持させることができる.

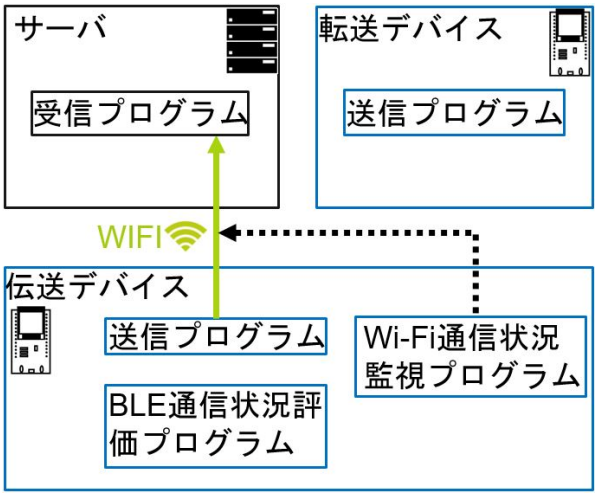

図 3 アーキテクチャ図 1

前述の通りこのようなことを実現するため, どのデバイスを経由という経路選択問題を先に解決しなければならない. Raul Rondon らの論文により BLE 通信のデバイスには、ペリフェラル (Peripheral)、セントラル (Central)、ブロードキャスター (Broadcaster)、オブザーバー (Observer) の 4 つの異なる役割を割り当てることができる. ブロード

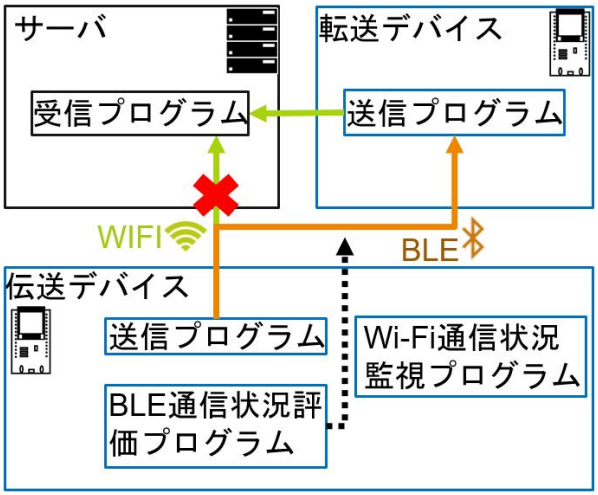

図 4 アーキテクチャ図 2

キャスターとオブザーバーの役割を持つデバイスは、アドバタイジングチャネルを利用して、それぞれブロードキャストデータを送受信する [15]. BLE は近距離通信技術として開発され, 30 メートル以上の距離を設定することは可能ではあるが、実際には 5 メートル程度にまで短くされる. このため, 通信帯域が圧迫せずに BLE 通信範囲内にある全てのデバイスへブロードキャストしてパケットを送信することが可能となっている. その後, パケットを受信したデバイスが受け取ったパケットをサーバへ転送して, サーバで最終に到着したデータ量を評価する. ここに評価する基準は PDR(パケット配信率) を採用する. その評価結果が予め設定された値を達すると, サーバから転送デバイスを経由して伝送デバイスまでかかった時間を計算して, 時間の短い順で, 一番かかった時間が短いリンクから転送デバイスを選択する. このリンクの転送デバイスを経由ノードとして二つのデバイスの接続を確立し, データを転送し始める.

一つのノードを経由してアクセスできるシングルホップパターンと複数のノードを経由するマルチホップパターンがある. シングルホップであれば一つのノードを経由してデータを送信することが可能する, これを図 2 で示す. 図 5 であるマルチホップの方は更に複雑になる. 通信障害があるノードが BLE を使って, 通信範囲内における全てのノードがサーバへ接続できないであれば, 又はスリープモードの場合だと, 範囲内におけるノードをもう一回 BLE 通信範囲内におけるサーバへ接続できるノードを探索が必要がある.

4. 実装と実験環境

実装の概念図とするソフトウェア構成図を図 6 に示す. 図 6 であるそれぞれ番号の内容は後で述べた.

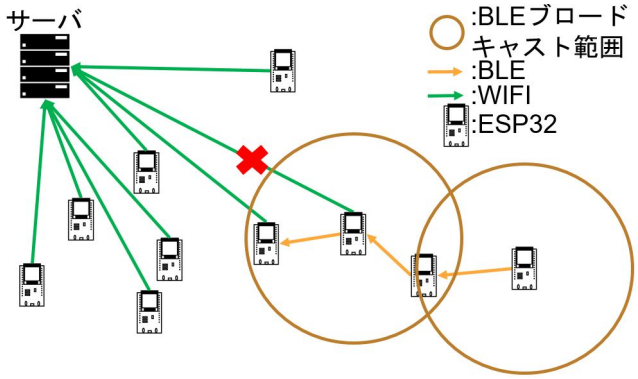

図 5　マルチホップ方式

4.1　実装

今度は ESP32 を二つ用意して実験を行った．VM の Ubuntu で Ubuntu18.04 をインストールしたもので実験用のサーバを作成した．図 6 の通り使用しているデバイスは ESP32 が ESP32A と ESP32B, サーバが一つある．各 IoT デバイス (ESP32) が同じなプログラムが内蔵されておる．それぞれはデータを伝送する sender, データ伝送状況を監視する monitor, BLE を使ってリクエストブロードキャストする ble_broadcast と IoT デバイス間の BLE 通信状況を評価する status_evaluation である．サーバではデータの取得と時間を計算する receiver がある．一般的な状況では ESP32A の sender が順調にサーバにある receiver にデータを伝送する．もし ESP32A の monitor はその間の Wi-Fi 通信が異常を検知したら, ESP32A の blesearch を起動して今の BLE 通信範囲内における全てのデバイスへ発信リクエストをブロードキャストする．ブロードキャストされたパケットを受信した ble_central が受信したデータを ESP32B の sender へ伝送して, 最終にデータを ESP32B の sender から発信する．途中に ESP32A の status_evaluation がその BLE 通信状況を評価して, ESP32B にある ESP32B とサーバの通信状況監視する monitor の結果をプラスして最初に設定した基準値以上であれば, ESP32B からサーバまでかかった時間と ESP32A から ESP32B までかかった時間を加算して, 時間が一番短い場合に引き続き ESP32B を経由ノードとしてデータを転送する．そうでない場合は, 接続を切断して最終の評価を満たすまで上述の流れを繰り返す．

(1)　BMP280 から温度データの取得．

(2)　Monitor がサーバと ESP32 の通信状況を ICMP を送って定期的に監視する．

(3)　通信障害に異常を検知した時に転送リクエストをブロードキャストで発信するプログラムを起動する．

(4)　BLE 通信を使って, 通信範囲内における全てのデバイスと接続して, テスト用データを送る．

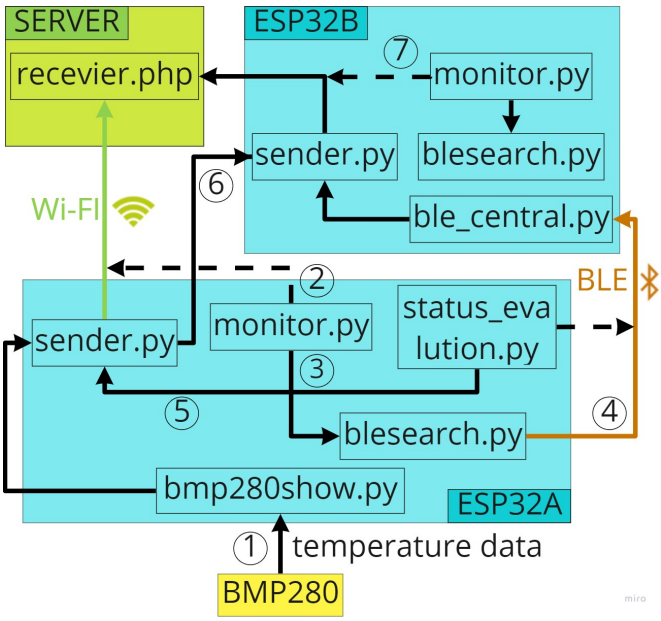

図 6　ソフトウェア構成図

(5)　転送リクエストを受信した後に BLE 通信評価プログラムは通信障害があるノードへの BLE 通信状況を評価する．

(6)　通信障害があるノードとの ESP32 はパケット配信率と応答速度が極めて悪い場合は次のノードをリクエスト．そうではない場合には受信した ESP32 を転送ノードとする．

(7)　全ての ESP32 が監視用プログラムが内蔵しているので, データを転送と同時に自体監視も行っている．

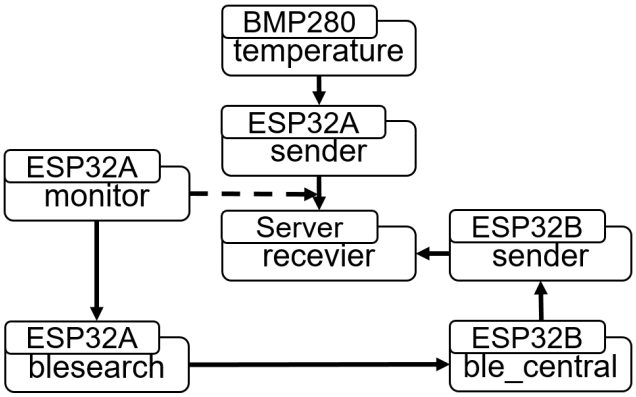

図 7　フロチャート

4.2　実験環境

実験を行うために ESP32A と ESP32B 二つのデバイスを用意して, サーバは VM の Ubuntu で Ubuntu18.04 を構築したものを使用する [16]．本研究では通信状況が異常がある時しか機能しないため, 実験する際に ESP32A の Wi-Fi 接続を事前に切断する必要がある．具体的にはデバイスの

Wi-Fi モードを off にする後に, ESP32A が自動的に BLE 通信モードに変更されたかどうかを確認する. BLE 通信手段でどのぐらいのデータを転送したのかを PDR で表示する. 実験をする際に, 通信障害を持っている ESP32A にある送信するプログラム sender が発送する時点の時間を記録して, 温度データと共にパケットをまとめて BLE で ESP32B に内蔵している受信用プログラム ble_central に送信する. 受信した後に ESP32B にある sender からデータをサーバへ送信し, サーバにある receiver がデータを受信すると共にデータを受信した時の時間と発信する時の時間を比較して評価する. 最後に発信時間と受信時間の差を ESP32B の sender に返信して表示する. 図 8 はこれを示す.

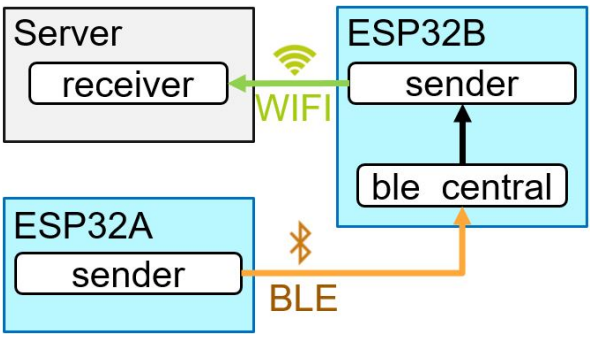

図 8　実験図

5.　評価と分析

実験を三つの段階を分けて評価を行う. まずは実験を行い, 人為的に Wi-Fi モードを off した後に IoT デバイスとサーバへの通信異常を検知するまでかかった時間を算出する. 次に, 通信異常を検知するした後に BLE モードとして動作するまでかかった時間を測る. 最後に転送ノードとして最終にデータをサーバへ転送するまでかかった時間を計測する.

- 通信異常の検知
- BLE モードの変更
- 転送データの到着

異常な通信がない場合, 発信元の IoT デバイス ESP32A から受信元の Server までデータを直接伝送する平均時間と今度の実験で転送ノード ESP32B を経由して ESP32A から Server までデータを間接に転送する平均時間を分けて計算する. 最終的な計算結果を比較して結論を導き出す. 流れは図 9 を示す. 実際に実験を行い, IoT デバイスから直接サーバへ送受信続けて, 平均時間を百回計測して 0.545 秒を得られた. 転送デバイスを経由する場合も百回計測して平均時間は 1.823 秒である数値を取得した. これによって, 正常にデバイスから直接サーバへ送信するより, 通信障害のあるデバイスが転送デバイスを BLE で経由して間接に

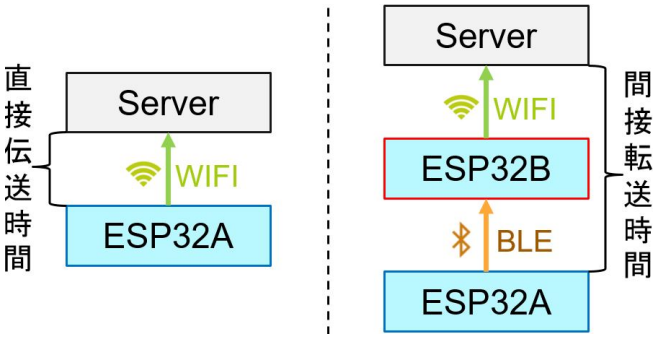

図 9　評価概念図

送信する方が約 1.2 秒長くかかることが分かった.

IoT デバイスから直接にサーバへ送信する場合を正常状態で, 通信障害を受けたと仮定して転送デバイスを経由することでサーバへデータを送信する場合は今度の提案方式である. それぞれの実験回数による時間コスト遷移グラフを図 10 で作成した. 図 10 のように, 正常状態での直接伝送時間コストは計測回数と関係なく, ずっと 0.5 秒左右に維持されている. 一方で近隣の IoT デバイスを経由する場合には時間コストが計測回数と共に周期的のように変わっていくことが発見した. 特にピークの前後における 2 つの値はほぼ 2 秒の巨大な差がある. 時間コストは一旦ピークになったら迅速に減っていく傾向を持っていることも見られる.

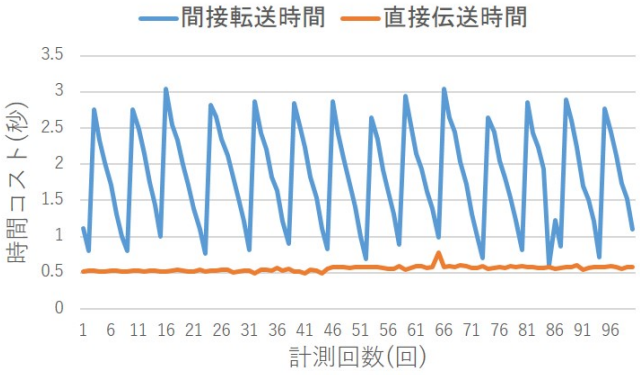

図 10　正常状態と提案方式の時間コスト比較

6.　議論

本研究で通信障害があるデバイスの対処法について研究を遂行し, 参考になれる micropython に基づく BLE 研究を拝読しようと考えていたが, 実際に BLE のモジュールを ESP32 に導入するのが未だに未熟であるため, 参考できる記事が極めて少ない. しかし, ラズパイと比べてより小さいサイズを持っているので, 拡張性を強調する IIoT(Industry Internet of Things) にとって開発価値がある. 関連研究としても BLE 通信に関する研究は ESP32 を用いて行った実験もほとんど掲載されていないことが事実である. これか

ら ESP32 の開発価値を着目し, 研究を進め, 将来に IIoT の主流な IoT デバイスの一つになると可能性がある.

マルチホップ方式の解決方法をここで軽く述べる. 前述のほとんどがシングルホップ方式だと思うが, 実際のマルチホップ方式はシングルホップの角度から考えてもらえば難しくないはず. ここで図 2 と図 5 を合わせて解説する. まずは図 2 のようなデバイス A がデバイス B を経由してデータを転送するシングルホップ方式から考えよう. この時点で, デバイス B が自体 Wi-Fi でサーバと繋がっていないなら, デバイス B をデバイス A としてもう一歩デバイス B を探すのが一般的だが, マルチホップの構成は逆の方である. 実際に, デバイス A が転送リクエストをブロードキャスト際には, 既にマルチホップでもしくはシングルホップでサーバとの通信が接続されていないデバイスしか経由デバイスにならない. つまりデバイス B が経由ノードとする前に自体がデバイス A にもなったことがある可能性がある. これで既にデバイス A になったことあるデバイス B は自動的にマルチホップ方式になれる.

BLE に関する研究は Beacon[14] を関連するケースがあるが, 実際にビーコンを使用するかどうかは、実装内容によって異なる. Beacon は BLE の通信モジュールでデータ発信の役割を担っているものの, データ受信とデータの応用に向かっていない欠点がある. また, Beacon の導入と共に必ず多少のコストがかかってしまうので, 純粋な ESP32 で構成された本研究においては適用していない.

または今度の実験で BLE を長時間使うと, ブロードキャスターであれオブザーバーであれ, ESP32 に膨大な負荷をかかってしまうことが発見した. API との反応が遅くなるのはともかく, プログラムが機能しないケースも時々起こる. したがって, 使用しないときは BLE をオフにするのが最善である. 全体のソフトウェアを評価する時に, bmp280 から取った温度変数をタイムスタンプに変えた後で, 明らかにサーバから返信されたデータの間隔が遅くなっていた. これになった理由はタイムスタンプがウェブサイトから取得してきた可能性がある.

評価するの時間変数に関して, 今回の実験では時間差を取るためにタイムスタンプを採用した. しかしながら, ESP32 の時間変数には二つの問題がある. 一つ目は ESP32 の時間変数はミリ秒が取得できないである. 既存のライブラリの中の localtime も現地時間ではない. 最後に worldtimeapi というウェブサイトで現地時間を取得することで解決した. 二つ目は ESP32 のタイムスタンプ関数は従来の 1970 年ではなく 2000 から計算されているので, サーバにおける従来のタイムスタンプを一致させるもしくは 30 年の空白を埋めるために, 自分で 1970 年から 2000 年までの秒数と現地時間との差を計算して補足させた.

7. おわりに

本研究では Wi-Fi 通信範囲外にあるか, 電波干渉により通信障害が発生してしまう IoT デバイスに焦点を当てた. 通信障害の検出については, 今回は ICMP を送ることで監視する. 通信障害を受けた IoT デバイスの解決策は迂回回路で近隣の IoT デバイスを経由ノードとしてデータを転送し, データの送受信を維持させる手法を提案した. 実験のところに IoT デバイス間の BLE 通信状況を評価するため, PDR という指標を基に評価方法を提出した.

本実験では二つ ESP32 を IoT デバイスとして, 通信障害を受けた際に提案によるどれぐらい時間がかかるのを測定した. その結果, 時間コストは経由デバイスを探索, 確立, データを転送するまでの回数に伴って, 周期的に大きく変動し. 平均で約 1.2 秒増えていることが分かった.

参考文献

[1] Shancang Li, Li Da Xu and Shanshan Zhao:The internet of things: a survey *Springer Science+Business Media New York 2014*, DOI 10.1007/s10796-014-9492-7(2015).

[2] Kanitkorn Khanchuea and Rawat Siripokarpirom:A Multi-Protocol IoT Gateway and WiFi/BLE Sensor Nodes for Smart Home and Building Automation: Design and Implementation *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, DOI 10.1109/ICTEmSys.2019.8695968(2019).

[3] Cheng Zhou, Jiazheng Yuan, Hongzhe Liu and JingQiu:Bluetooth Indoor Positioning Based on RSSI and Kalman Filter *Wireless Personal Communications*, 96, pages4115–4130(2017).

[4] Guntur Dharma Putra, Azkario Rizky Pratama, Alexander Lazovik and Marco Aiello: Comparison of energy consumption in Wi-Fi and bluetooth communication in a Smart Building *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, DOI 10.1109/CCWC.2017.7868425(2017)

[5] Albert F. Harris III, Vansh Khanna, G ¨ uliz Tuncay, Roy Want, and Robin Kravets: Bluetooth Low Energy in Dense IoT Environments *IEEE Communications Magazine*, DOI 10.1109/MCOM.2016.1600546CM(2016).

[6] Johanna Nieminen, Carles Gomez, Markus Isomaki, Teemu Savolainen, Basavaraj Patil, Zach Shelby, Minjun Xi, and Joaquim Oller: Networking Solutions for Connecting Bluetooth Low Energy Enabled Machines to the Internet of Things *IEEE Network*, DOI 10.1109/MNET.2014.6963809(2014)

[7] Bluetooth 4.2 Core Specification *tech. rep., Bluetooth*, (2014).

[8] Roy Want, Bill Schilit, and Dominik Laskowski: Bluetooth LE Finds Its Niche *IEEE Pervasive Computing*, DOI 10.1109/MPRV.2013.60(2014).

[9] Huang-Chen Lee , Senior Member, IEEE, and KaiHsiang Ke:Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation *IEEE Transactions on Instrumentation and Measurement*, DOI 10.1109/TIM.2018.2814082(2018).

[10] Sanjit Biswas and Robert Morris:ExOR: Opportunistic

Multi-Hop Routing for Wireless Networks *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, (2005).

[11] Syed Rafiul Hussain, Member, IEEE, Shagufta Mehnaz, Member, IEEE, Shahriar Nirjon, Member, IEEE, and Elisa Bertino, Fellow, IEEE: Secure Seamless Bluetooth Low Energy Connection Migration for Unmodified IoT Devices *IEEE Transactions on Mobile Computing*, 24, pages278–290(2017).

[12] Paul Loh Ruen Chze, Kan Siew Leong Communications and Networks Group, School of Engineering Nanyang Polytechnic Singapore: A Secure Multi-Hop Routing for IoT Communication *2014 IEEE World Forum on Internet of Things (WF-IoT)*, DOI 10.1109 / WF-IoT.2014.6803204(2014).

[13] T. Clausen and P. Jacquet: RFC3626: Optimized Link State Routing Protocol (OLSR) *RFC Editor United States*, (2003).

[14] Bharath Sudharsan, John G. Breslin and Muhammad Intizar Ali: Adaptive Strategy to Improve the Quality of Communication for IoT Edge Devices *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, DOI 10.1109/WF-IoT48130.2020.9221276(2020).

[15] Raul Rondon, Mikael Gidlund and Krister Landernas: Evaluating Bluetooth Low Energy Suitability for Time-Critical Industrial IoT Applications *International Journal of Wireless Information Networks volume*, DOI 10.1109/TMC.2017.2739742(2018).

[16] Agus Kurniawan: *Internet of Things Projects with ESP32*, (2019).

