IoT機器の共同通信による障害情報の開示およびMQTTクライアント切断遅延の短縮

高橋 祐之介[1]　串田 高幸[1]

概要：IoT 機器のメッセージ交換方法として MQTT プロトコルを用いる事がある．IoT 機器の死活監視方法として MQTT ブローカーのコネクション確立状況を使用する方法がある．しかし，MQTT プロトコルは接続継続の為の Keep-Alive メッセージの最終到着時間を使用し，Keep-Alive-Time のタイムアウト時間を超えないと MQTT クライアントを切断しない．これにより，IoT 機器の障害が発生してから切断までに遅延が発生してしまう問題がある．そこで，MQTT，ICMP パケットを使用したエコーリプライの有無，Bluetooth 電波発信状況を使用したハイブリッドな原因特定方法を提案する．同一セグメント上の複数の IoT 機器が障害の発生した IoT 機器と通信を行う事で，障害発生時にネットワーク，ソフトウェア，ハードウェアまたは電源の何に障害が発生しているか切り分けを行う．MQTT ブローカーは Keep-Alive-Time のタイムアウト時間に加え，切り分け内容を切断判断の材料とする事で切断遅延を短縮した．評価として，提案手法と既存の MQTT プロトコルによる IoT 機器切断遅延時間を比較した．提案手法は最大遅延時間，最小遅延時間共に既存手法よりも半分以下の時間で切断処理を行う事ができた．また，障害の発生した IoT 機器の状況に応じた，適切な障害内容を表示する事が出来た．

1. はじめに

背景

IoT 機器はインターネットを介して通信を行い，人の手を借りずに情報を取得する機器の事である [1]．IoT 機器は直接インターネットに接続し，センサーデータをクラウドへ送る役割を担い，センサーから収集されたデータをリアルタイムで処理している [2]．Ericsson Mobility Report によれば，IoT 機器の台数は 2021 年に 14.6 億の IoT 機器が存在し，2027 年には 30.2 億台以上を超えると予測されている [3]．

IoT システムで使用されるプロトコルの 1 つとして MQTT がある．MQTT はリソース制約の多い M2M 通信用に設計された出版-購読型メッセージ伝達プロトコルである．IBM によって 1999 年に開発された [4]．MQTT は TCP を使用して接続を行う為，メッセージの確実な伝達が必要とされる IoT システムの要件に対して使用される．HTTP によるメッセージの伝達よりも，パケットのオーバヘッドとメッセージサイズが小さい為，リソース制約の多い IoT デバイスで使用される [5]．

MQTT プロトコルを使用して IoT 機器が別の IoT 機器にメッセージを伝達したい場合，MQTT クライアントにあたる IoT 機器はトピック識別名を指定してメッセージを MQTT ブローカーに送信する．MQTT ブローカーはメッセージのトピック識別名を購読している別の MQTT クライアントに受け取ったメッセージを転送する事で伝達できる．

MQTT クライアントが MQTT ブローカーに接続した場合，MQTT クライアントにあたる IoT 機器は Ping メッセージを定期的に送信している．この Ping メッセージの役割は MQTT ブローカーと MQTT クライアント間の接続を維持する為に送信する，Keep-Alive メッセージである [6]．

MQTT プロトコルは接続継続の為の Keep-Alive-Time を使用し，MQTT クライアントの切断判断をしている [7]．その為，MQTT クライアントである IoT 機器の電源を遮断してから，Keep-Alive-Time のタイムアウト時間を超えるまで MQTT クライアントを切断しない．その間，ユーザーや管理者は MQTT クライアントが正常に動いているという判断をする．IoT デバイスと接続して，監視・制御を行えるクラウドサービス Azure IoT hub や AWS IoT core も MQTT ブローカーのコネクション確立状況を死活監視に使用している．

また，IoT 機器の障害にはハードウェアの障害，電源の遮断，ネットワーク障害，人為的なミスがある [8]．障害を

[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

迅速に復旧するには，障害の詳細情報を特定する必要がある [9].

課題

MQTT プロトコルは IoT 機器に障害が発生しても，コネクションの切断までに遅延が発生するという課題がある．また，障害が発生した IoT 機器がメッセージを送信できない状態になる事で，IoT 機器の状況を監視サーバに伝えられない．その為，IoT 機器の詳細情報がわからない課題がある．

基礎実験

既存の Keep-Alive-Time のタイムアウト時間を使用した MQTT クライアント切断までの遅延時間を確認する．

IoT 機器に SoC のマイクロコントローラである ESP32 を使用し，電源を遮断してから MQTT クライアント切断までの遅延時間を計測する．IoT 機器の Keep-Alive-Time は 30 秒と 60 秒に設定し，それぞれ計測を行う．IoT 機器の電源を投入し，MQTT ブローカーに接続してから電源を遮断するまでの時間を 0 秒から 140 秒まで 1 秒ずつ変更して，MQTT クライアント切断までにかかる遅延を計測する．

計測結果を図 1，図 2 に示す．

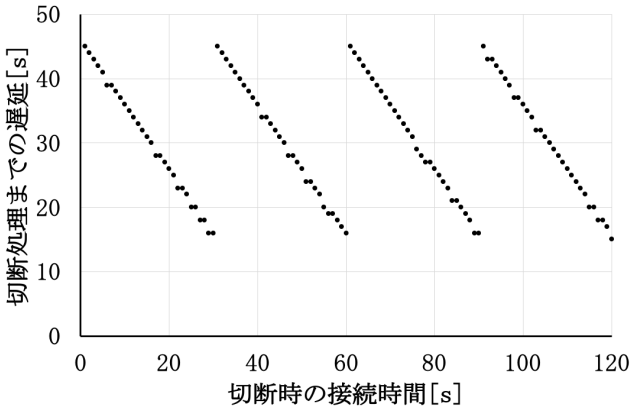

図 1　Keep-Alive-Time 30 秒の検出遅延

グラフの横軸は IoT 機器が MQTT ブローカーに接続してから電源を遮断するまでの時間で，縦軸は IoT 機器の電源を遮断してから MQTT クライアント切断までの遅延時間を表す．

MQTT クライアント切断までの遅延時間を表 1 に表す．

表 1　切断までの遅延時間

Keep-Alive-Time	最小遅延時間	最大遅延時間
30s	15s	45s
60s	30s	90s

基礎実験から，MQTT プロトコルによって接続された IoT 機器の電源を遮断してから MQTT クライアント切断

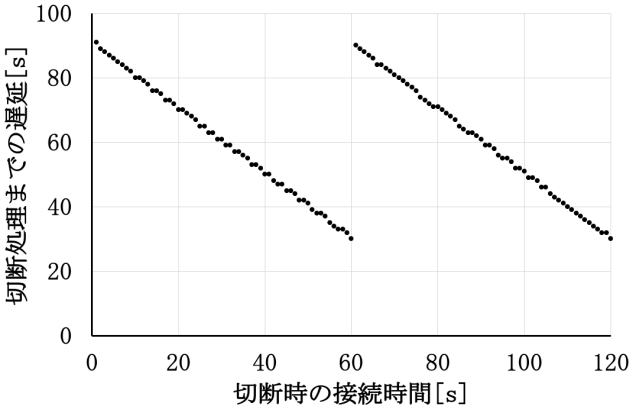

図 2　Keep-Alive-Time 60 秒の検出遅延

までに遅延が発生する事が確認できた．

MQTT の仕様書を確認すると，Keep-Alive-Time のタイムアウト時間は Keep-Alive-Time を 1.5 倍した時間になる．

Keep-Alive メッセージの送信直後に電源が遮断された時の切断遅延の関係を図 3 に示す．Keep-Alive メッセージの送信直後に電源が遮断された時，遅延時間は最大遅延時間に近くなる．

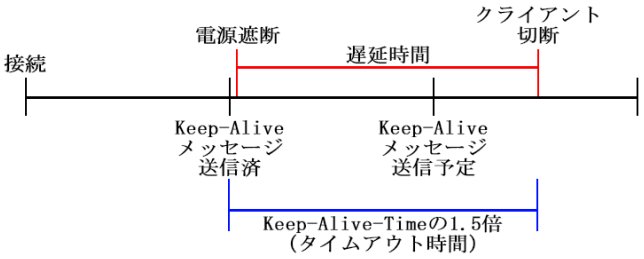

図 3　Keep-Alive メッセージの送信直後の電源遮断

Keep-Alive メッセージの送信直前に電源が遮断された時の切断遅延の関係を図 4 に示す．Keep-Alive メッセージの送信直前に電源が遮断された時，遅延時間は最小遅延時間に近くなる．

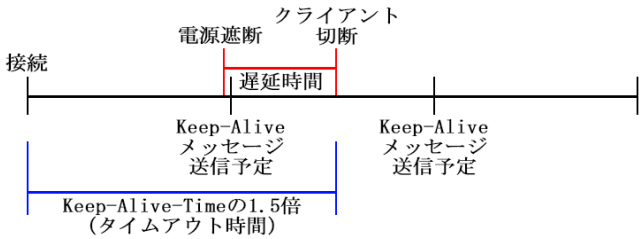

図 4　Keep-Alive メッセージの送信直前の電源遮断

Keep-Alive-Time を短くすれば MQTT クライアント切断までの遅延時間を短縮する事ができる．しかし，無線 LAN 接続が不安定な状況下で MQTT クライアントが通信を続ける必要がある状況でもブローカサーバは MQTT クライアントのコネクションを切断してしまう問題が発生してしまう．障害検出時間の短縮と誤検知の可能性はトレー

ドオフの関係がある．その為，Keep-Alive-Time を短い時間に設定する事も難しい [10].

各章の概要

2. 関連研究

IoT 機器の死活監視に関する論文で，IoT 機器の障害を時系列データから検知する方法がある [11]．IoT 機器のデータや数値が一時的に普段とは違うものになり，短期間で元の状態に戻った場所がないか確認して障害検知を行う「Point Anomaly」と呼ばれる方法，過去の数値から予測したパターンとは外れた場所がないか確認して障害検知を行う「Contextual Anomalies」と呼ばれる方法，一定間隔のデータを 1 つの集合として，規則的に動いていたパターンから外れた場所がないか確認して障害検知を行う「Collective or Pattern Anomalies」の考え方に沿って障害検出を行う方法が記述されている．Keep-Alive-Time のタイムアウト時間 以外の情報を使用して，障害の検知ができる為，障害の検出を早くできる利点がある．しかし，IoT 機器の障害検知後，障害が発生した箇所や原因を見つけ出す事は出来ない為，MQTT クライアント切断の判断材料には使用できない．

IoT 機器の死活監視に関する論文で，IoT 機器に接続されたセンサー値の相関を事前計算し，相関ないデータを取得した際に障害と検知する方法がある [12]．リアルタイムでデータの相関関係をチェックする為，障害の検出を高速化する事ができる．しかし，IoT 機器の障害検知では無く，センサーの障害検知をしている為，MQTT クライアント切断の判断材料には使用できない．

関連研究の内容では，障害が発生した IoT 機器の詳細情報を取得できない問題が解決できておらず，IoT 機器の電源を遮断してから，MQTT クライアント切断遅延が発生してしまう．

障害の検出の為に，ログファイルを分析する方法が研究されている [13]．しかし，IoT 機器がサーバに対して，メッセージを送信できない状態になった場合は，障害が発生した後の情報を伝えられなくなる．そのため，障害発生前の分析しかできない．

3. 提案

MQTT ブローカーが IoT 機器の切断を行なう際の判断材料を増やす為に，MQTT 通信，ICMP パケットを使用したエコーリプライの有無，Bluetooth 電波発信状況を使用したハイブリッドな原因特定方法を提案し，特定した原因を元に IoT 機器の切断判断をおこなう．IoT 機器は電源が通電されている際，常に Bluetooth 電波を発信する．本提案方式を使用する際の前提条件として，IoT 機器の Bluetooth 電波を同一セグメント上に配置された IoT 機器が受信できる場所に設置する必要がある．Bluetooth 電波発信状況は，同一セグメント上にある他の IoT 機器が対象の IoT 機器の Bluetooth 電波を受信できるかで判断を行う．Bluetooth 電波の識別には Bluetooth MAC アドレスを使用する．

特定した IoT 機器の情報は Web ブラウザから閲覧できるようにする．

提案手法の処理の流れを図 5 に示す．

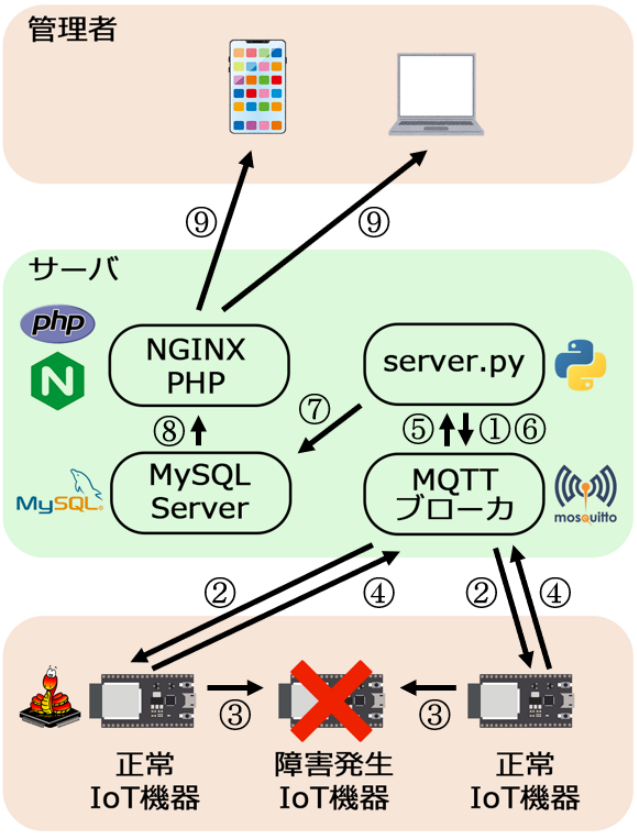

図 5 原因特定方法による処理の流れ

(1) IoT 機器の障害発生を感知し，障害が発生した IoT 機器と同一セグメント上のすべての IoT 機器に対して，状況確認メッセージを送信する．

(2) MQTT ブローカーは障害が発生した IoT 機器と同一セグメント上の IoT 機器すべてに対して，メッセージを転送する．

(3) 要求を取得した IoT 機器は，障害が発生した IoT 機器に対して，ICMP パケットを使用したエコーリクエストを送信する．Bluetooth 電波のスキャンも行う．

(4) 同一セグメント上のすべての IoT 機器は MQTT ブローカーに障害が発生した IoT 機器からのエコーリプライの有無と Bluetooth 電波取得状況のメッセージを送信する．

(5) MQTT ブローカーは監視ソフトウェアにメッセージを転送する．

(6) 監視ソフトウェアは IoT 機器からの情報を元に，障害

が発生した IoT 機器の障害内容の切り分けを行う．障害切り分け内容に応じて，IoT 機器の切断処理の要求を行う．

(7) 切り分け終了後，ログ管理用データベースに切り分け内容を追加する．

(8) ログ管理用データベースの内容を元に，HTML ファイルを作成し，Web ブラウザから閲覧できるようにする．

(9) Web ブラウザから監視サーバにアクセスし，障害が発生した IoT 機器の障害内容を確認する．

本提案方式を使用する事で MQTT ブローカーは IoT デバイスの電源が遮断されている事を確認してから切断処理を行なう事ができる為，Keep-Alive-Time のタイムアウト時間に達する前に MQTT クライアント切断を行なう事ができるようになる．

原因特定を行う前に，障害を検知しなければならない．MQTT メッセージを MQTT ブローカーに送信できない状態を障害とする．その為，IoT 機器には Keep-Alive 送信間隔加え，Heart-Beat 送信間隔を設定する．Heart-Beat-Time は Keep-Alive-Time よりも短い時間で設定し，設定した間隔毎に Heart-Beat メッセージを送信する．Heart-Beat-Time を 1.5 倍した時間を Heart-Beat-Time のタイムアウト時間とする．Heart-Beat メッセージの最終到着時間が Heart-Beat-Time のタイムアウト時間を超えた場合，機器に障害が発生したと判断する．

障害が発生した IoT 機器の障害内容の切り分けを行う為に，監視サーバは同一セグメント上に配置された他の IoT 機器へ情報の収集を行なう．監視サーバは障害が発生した機器の Bluetooth 電波発信状況と ICMP パケットを使用したエコーリクエストに対するエコーリプライの有無の確認を同一セグメント上に配置された他の IoT 機器すべてに要求する．要求を受け取った IoT 機器は障害が発生した IoT 機器と直接通信を行い，Bluetooth 電波発信状況と ICMP パケットを使用したエコーリプライの有無を監視サーバに送信する．

同一セグメント上の IoT 機器増加による，IoT 機器情報取得の遅延が発生する事を考慮し，IoT 機器への要求と情報の取得は MQTT プロトコルを使用し非同期で通信を行う．

提案アルゴリズム

監視サーバは障害が発生した IoT 機器と同一セグメントに配置された IoT 機器からの情報を使用して，障害内容の切り分けを行う．監視サーバは障害内容の切り分けを行なう際に，図 6 に示すアルゴリズムを使用する．

障害内容切り分けアルゴリズムでは，初めに，同一セグメント内のネットワーク全体に障害が発生しているか，一部の IoT 機器に障害が発生しているかの切り分けを行う．

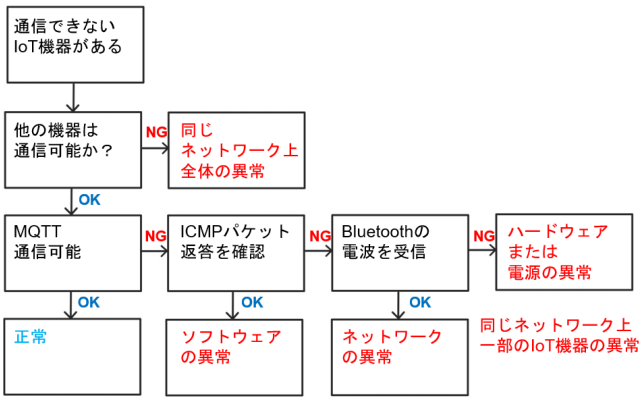

図 6 障害内容切り分けアルゴリズム

この切り分けは，障害が発生した機器以外で同一セグメント上に正常動作している機器があるか確認する．1 台でも正常動作している機器がある場合は，一部の IoT 機器に障害が発生していると判断する．

次に，障害が発生した IoT 機器と同一セグメント上にあるすべての IoT 機器が，障害が発生した IoT 機器に対して，ICMP パケットを使用したエコーリクエストを送信する．障害が発生した IoT 機器からのエコーリプライを確認した機器が 1 台でもある場合，ソフトウェア異常の可能性と判断する．

次に，障害が発生した IoT 機器と同一セグメント上にあるすべての IoT 機器が，周囲の Bluetooth 電波のスキャンを行う．障害が発生した IoT 機器と同じ Bluetooth MAC アドレスの電波を IoT 機器が 1 台でも確認できた場合，ネットワーク異常の可能性と判断する．

障害が発生した IoT 機器と同一セグメント上のすべての IoT 機器が ICMP パケットを使用したエコーリプライと Bluetooth 電波を確認できない場合，ハードウェアの異常または電源の異常の可能性と判断する．

ハードウェアの異常または電源の異常の可能性と判断された場合は MQTT ブローカーに対し，MQTT クライアントの切断要求を行う．それ以外はネットワークの再接続や，ソフトウェアの再起動によって再度パケットが送信される可能性がある為，Keep-Alive-Time のタイムアウト時間が経過するまで MQTT クライアントの切断を行わない．

ユースケース・シナリオ

提案手法は室内，屋外問わず，同一ネットワーク上に複数の IoT 機器を配置する状況での使用を想定している．サーバルームや美術館などの常に温度管理が必要な場所や，病院や自宅で使用できる緊急通報ボタンなど，常に IoT 機器が正常に動作している必要のある場合に使用できる．MQTT プロトコルを使用した緊急通報システムの例を図 7 で示す．

また，図 8 のような管理会社や管理人が遠隔で IoT 機器

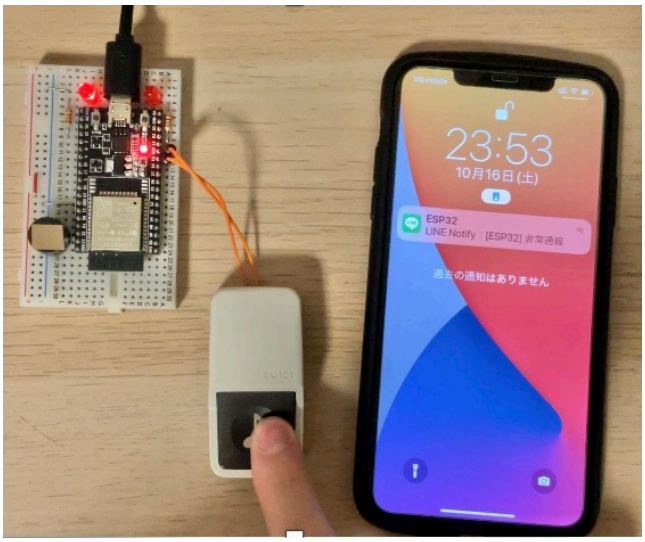

図 7 IoT 機器を使用した緊急通報システムの例

の管理を行う必要のある HVAC 空調システムや農業用ビニールハウス内センサーにも使用できる．

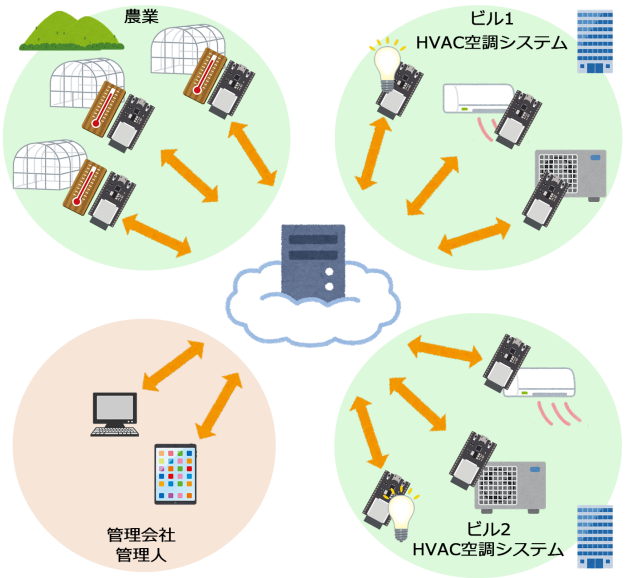

図 8 IoT 機器の遠隔監視

この提案方式を使用する事で，IoT 機器に障害が発生した際に，IoT 機器の障害内容を取得できる．その為，障害復旧作業を行う前に，事前調査が可能になり，早期復旧につながる．

4. 実装と実験方法

実装

監視サーバに Python を使用し，IoT 機器には Micro Python を使用して提案方式を実装する．IoT 機器同士や IoT 機器とサーバ間の通信には MQTT プロトコルを使用し通信を行う．

監視サーバに MySQL サーバをセットアップし，接続，切断，障害，詳細情報のログを格納するテーブルを作成する．nginx と PHP を使用し，インターネットブラウザ上から IoT 機器に関するログを閲覧できるようにする．

監視サーバは IoT 機器管理用クラスを使用する．以下，接続機器クラスという．IoT 機器接続時にインスタンス生成を行い，接続管理用配列に格納する．この接続管理用配列を使用して IoT 機器の接続管理を行う．切断時には接続管理用配列からインスタンスを削除する．

接続機器クラスには以下の内容を格納するインスタンス変数を用意する．

- グローバル IP アドレス
- ローカル IP アドレス
- IoT 機器名
- Bluetooth MAC アドレス
- 無線 LAN MAC アドレス
- Heart-Beat メッセージ送信間隔
- Keep-Alive メッセージ送信間隔
- ソフトウェアバージョン
- 機器の状態
- Bluetooth 電波取得状況格納配列
- ICMP パケットを使用したエコーリプライ返答の有無格納配列

IoT 機器は Heart-Beat メッセージの送信と接続時の情報送信以外は監視サーバの要求に応じて処理を行なう．要求メッセージの送信や，要求に対する情報の返却は MQTT プロトコルを使用して MQTT メッセージで通信を行なう．

トピック識別名

管理サーバからの要求は，処理内容をトピック識別名に含み，IoT 機器が処理に使用するデータをメッセージに格納して送信する．IoT 機器はトピック識別名とメッセージの内容に応じた処理を行い監視サーバに処理結果を送信する．

監視サーバと IoT 機器が購読するトピック識別名を表 2 に示す．

表 2 購読トピック識別名

トピック識別名	購読機器
s/#	監視サーバ
c/IoT 機器の無線 LAN MAC アドレス/#	IoT 機器
c/IoT 機器のグローバル IP アドレス/#	IoT 機器

監視サーバと IoT 機器が購読するトピック識別名の関係を 9 に示す．

トピック識別名に含まれる#はマルチレベル・ワイルドカードである．記号で指定した階層以下すべてのメッセージを購読する．特定の IoT 機器にメッセージを送信する場合は c/IoT 機器の無線 LAN MAC アドレス/ から始まるトピック識別名を使用し，同一セグメント上の機器すべて

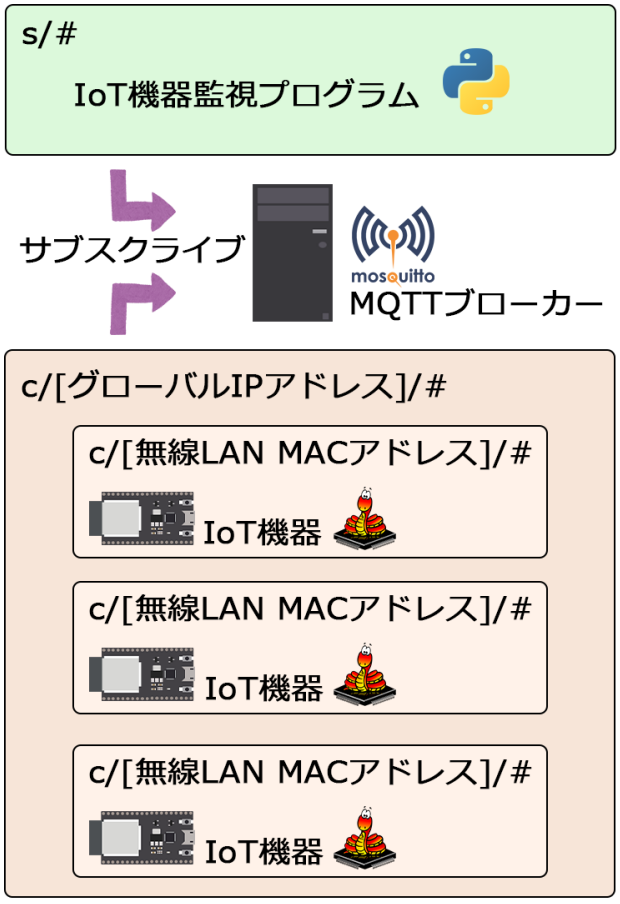

図 9　購読するトピック識別名の関係

にメッセージを一括して送る場合は c/IoT 機器のグローバル IP アドレス/ から始めるトピック識別子を使用して送信する．

今回の実装で使用するトピック識別名を表 3 に示す．

表 3　使用するトピック名

トピック名	送信機器	受信機器
s/join	IoT 機器	監視サーバ
s/disconnect	IoT 機器	監視サーバ
s/ping	IoT 機器	監視サーバ
c/無線 LAN MAC/wont_join	監視サーバ	IoT 機器
c/グローバル IP/search_bt	監視サーバ	IoT 機器
s/return_bt	IoT 機器	監視サーバ
c/グローバル IP/want_ping	監視サーバ	IoT 機器
s/return_ping	IoT 機器	監視サーバ

今回使用するトピック識別名と処理内容は以下の通りである．

s/join

IoT 機器に電源が投入され，新たに接続する際に，IoT 機器が監視サーバにメッセージを送信する．以下，接続通知という．メッセージには機器名，ソフトウェアバージョン，無線 LAN MAC アドレス，Bluetooth MAC アドレス，グローバル IP アドレス，ローカル IP アドレス，Keep-Alive 送信間隔，Heart-Beat 送信間隔を json 形式で送信する．監視サーバはメッセージを受け取った際に接続管理用配列にインスタンス生成しメッセージ内の IoT 機器に関する情報をインスタンス変数に格納する．インスタンス生成する際，機器の状態に”正常”を格納する．

s/disconnect

IoT 機器がネットワークから切断する際に，IoT 機器が監視サーバにメッセージを送信する．メッセージには無線 LAN MAC アドレスを入れて送信する．監視サーバはメッセージを受け取った際に接続管理用配列から無線 LAN MAC アドレスと同一機器のインスタンスを削除する．

s/ping

IoT 機器が Heart-Beat メッセージを監視サーバに送信する．メッセージには無線 LAN MAC アドレスを入れて送信する．監視サーバはメッセージを受け取った際に接続管理用配列から無線 LAN MAC アドレスと同じ機器の Heart-Beat メッセージの最終到着時間を現在時刻に変更する．

c/IoT 機器の無線 LAN MAC アドレス/wont_join

監視サーバから特定の ESP32 に接続通知の再送を要求する．メッセージには何も含まない．IoT 機器はメッセージを受け取った際に再度，接続通知を行う．

c/IoT 機器のグローバル IP アドレス/search_bt

監視サーバからトピック識別名内で指定したグローバル IP アドレスと同じ複数の IoT 機器に送信する．以下，Bluetooth 電波確認要求という．メッセージには Bluetooth 電波の発信状況を確認する IoT 機器の Bluetooth MAC アドレスと無線 LAN MAC アドレスを入れて送信する．IoT 機器はメッセージを受け取った際に周囲の Bluetooth 電波のスキャンを行い，スキャン結果の中にメッセージに含まれる Bluetooth MAC アドレスからの電波発信の有無を監視サーバに返却する．返却の際のトピック名は/s/return_bt を使用する．

s/return_bt

IoT 機器から監視サーバに Bluetooth 電波受信の有無を送信する．メッセージにはスキャンを行った IoT 機器の Bluetooth MAC アドレス，無線 LAN MAC アドレス，Bluetooth 電波受信の有無，を入れて送信する．監視サーバはメッセージを受け取った際，接続管理用配列から無線 LAN MAC アドレスと同じ機器の Bluetooth 電波取得状況配列に受け取った情報を追加する．

c/IoT 機器のグローバル IP アドレス/want_ping

監視サーバからトピック識別名内で指定したグローバル IP アドレスと同じ複数の IoT 機器に送信する．以下，エコーリプライ応答確認要求という．メッセージには ICMP パケットを使用した，エコーリクエストの送信とエコーリプライの有無を確認する IoT 機器のローカル IP アドレスと無線 LAN MAC アドレスを入れて送信する．IoT 機器は

メッセージを受け取った際にメッセージ内に含まれる IoT 機器のローカル IP アドレスに対して ICMP パケットを使用し，エコーリクエストを送信する．障害が発生した機器からのエコーリプライの有無を監視サーバに返却する．返却の際のトピック名は s/return_ping を使用する．

s/return_ping

IoT 機器から監視サーバに ICMP パケットを使用した，エコーリクエストに対するエコーリプライの有無を送信する．スキャンを行った IoT 機器のローカル IP アドレス，無線 LAN MAC アドレス，ICMP パケットを使用したエコーリプライの有無を入れて送信する．監視サーバはメッセージを受け取った際，接続管理用配列から無線 LAN MAC アドレスと同じ機器のエコーリプライ返答の有無を格納する配列に受け取った情報を追加する．

監視サーバの動作

サーバソフトウェア起動時，100ms 間隔で接続管理用配列に含まれる各インスタンス内の Heart-Beat メッセージの最終到着時間が Heart-Beat-Time のタイムアウト時間を超えていないか確認する．

IoT 機器の情報の要求

Heart-Beat-Time のタイムアウト時間を超えている IoT 機器が存在した場合，障害が発生している為，IoT 機器の状態を”正常”から”確認中”に変更する．その後，障害が発生している IoT 機器と同一セグメント上の IoT 機器にエコーリプライ応答確認要求と Bluetooth 電波確認要求を行う．

IoT 機器の状態が”正常”である，同一ネットワーク上に接続されている IoT 機器の情報がすべて集まるまで待機する．例えば，同一ネットワーク上に 8 台の IoT 機器が接続されていており，2 台の IoT 機器の状態が”確認中”の場合，5 台の IoT 機器からの情報が集まってから情報切り分けアルゴリズムを使用する．

情報の切り分け

障害が発生した IoT 機器のインスタンス変数である，Bluetooth 電波取得状況格納配列， ICMP パケットを使用したエコーリプライ返答の有無格納配列の内容を使用して，障害内容の切り分けを行う．

提案方式は，障害が発生した IoT 機器と同一セグメント上のすべての IoT 機器と通信できない場合は情報の切り分けができない．障害が発生した IoT 機器と同一セグメント上の IoT 機器の状態がすべて”確認中”の場合は同一セグメント上の機器すべてに障害が発生している事をログ表示用データベースに追加し，障害内容の切り分けを終了する．

初めに障害が発生した IoT 機器の ICMP パケットを使用したエコーリプライの有無を確認する．エコーリプライを確認した機器が 1 つでもある場合，ソフトウェアに障害が発生した可能性があると判断し，ログ表示用データベースに追加して，障害内容の切り分けを終了する．

次に，障害が発生した IoT 機器の Bluetooth 電波発信状況を確認する．対象機器の Bluetooth の電波を受信できた機器が 1 つでもある場合，ネットワークに障害が発生した可能性があると判断し，ログ表示用データベースに追加して，障害内容の切り分けを終了する．

最後に，Bluetooth 電波と ICMP パケットを使用したエコーリプライを 1 台も確認できなかった場合は IoT 機器の電源が遮断された可能性とハードウェア自体に障害が発生した可能性があると判断し，ログデータ表示用データベースに追加して障害内容の切り分けを終了する．

障害が発生した IoT 機器のエコーリプライか Bluetooth 電波をどちらか 1 つでも確認できた場合はネットワークの再接続や，ソフトウェアの再起動によって再度パケットが送信される可能性がある．その為，Keep-Alive-Time のタイムアウト時間が経過するまで切断を行わない．

電源が遮断された可能性とハードウェア自体が故障した可能性がある場合は，MQTT ブローカーに対象 MQTT クライアントの切断を要求し，監視サーバの接続管理用配列から IoT 機器のインスタンスを削除する．

実験環境

本稿では以下の環境で評価実験を行った．

IoT デバイス

- Hardware: ESP-WROOM-32
- Micro Python Version: 1.14
- 使用台数: 8 台

監視サーバ

- Hardware: 研究室内仮想マシン
- CPU: 1 コア
- Memory: 2GB
- OS: Ubuntu Server 20.4.3
- Python Version: Python 3.8.10

IoT デバイスとして，低消費電力 SoC のマイクロコントローラである ESP-WROOM-32 を使用した．以下，ESP32 という．ネットワークは研究室内のローカルネットーワークを使用した．

実験に使用するシステム構成を図 10 に示す．

実験では研究室内に ESP32 を 8 個配置し，実験を行った．実験で使用する ESP32 の配置を図 11 に示す．赤色の線で囲まれた ESP32 の状況を変更し，実験を行った．状況の変更は，電源の遮断，無線 LAN の切断，ソフトウェア上で例外の発生を行った．

ログ用データベースには MySQL を使用し，監視サーバと同じ仮想マシンに MySQL データベースをインストールした．ログ用データベースには日時，詳細情報，ログ表示色を格納できるテーブルを用意した．監視サーバは IoT 機器の接続，障害発生，障害内容の切り分け，切断処理が行われた際，ログ用データベースに日時と内容を記録した．

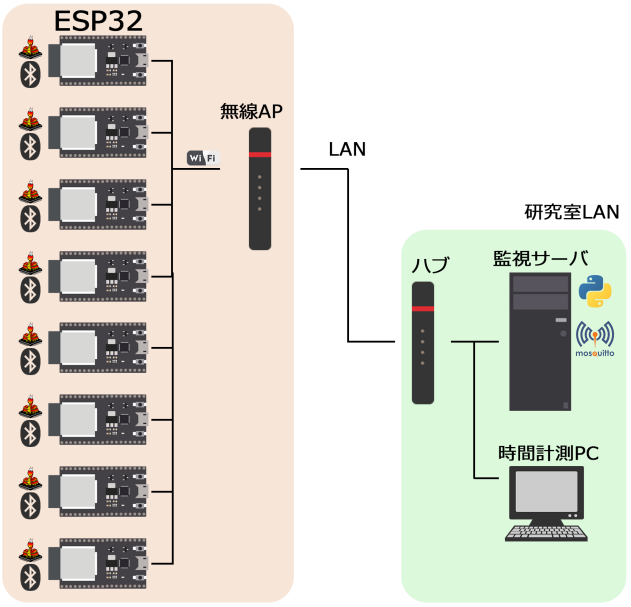

図 10　システム構成

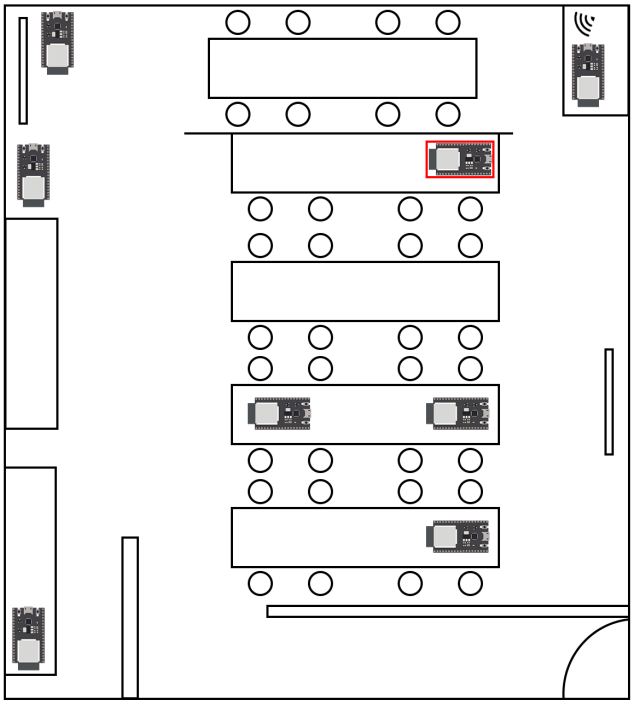

図 11　デバイスの配置図

ログの内容を Web ブラウザから閲覧できるように監視サーバと同じ仮想マシンに nginx と php をインストールし，ログ用データベースの内容を Web ブラウザから閲覧できるようにした．

すべての IoT 機器が接続通知を行った後，正常に ESP32 が動作をしている場合のログ表示を図 12 に表す．

Web ブラウザに表示されるログの色は表 4 のようになっている．

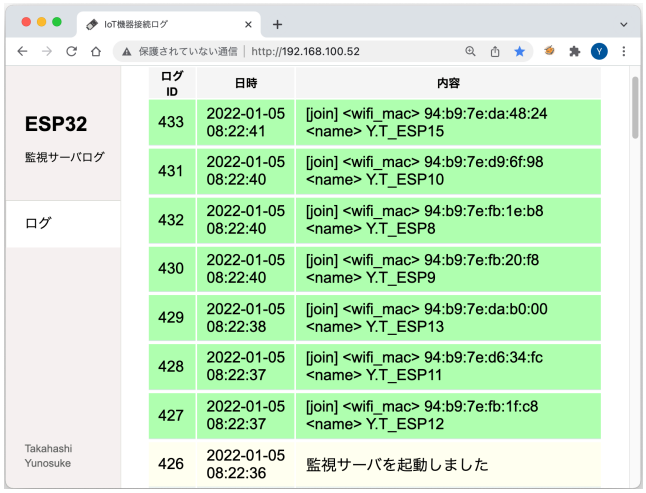

図 12　接続通知が行われた直後の Web ブラウザ画面

表 4　ログ表示の色

色	意味	追加タイミング
緑	接続時	接続通知時
黄	障害切り分け内容の表示	情報切り分け終了時
橙	障害発生時	Heart-Beat タイムアウト経過時
赤	切断処理時	切断処理実行時

5.　評価手法と分析手法

障害の発生と本提案による障害内容切り分け

電源遮断

停電や誤って電源プラグを抜いてしまう事を想定した実験を行う．本実験時は，プログラム動作中の IoT 機器の電源を遮断する事で電源の障害を発生させた．障害が発生し，Heart-Beat-Time のタイムアウト時間経過後，同一セグメント上に配置された他の IoT 機器から送信され，障害内容の切り分けアルゴリズムで使用した情報を示す．

```
[2022−01−05 08:23:00] [system]
return_ping 94:b9:7e:d9:71:b8 <ng>7 <ok>0
[2022−01−05 08:23:03] [system]
return_bt 94:b9:7e:d9:71:b8 <ng>7 <ok>0
```

障害が発生した IoT 機器の Bluetooth 電波を受信できた IoT 機器はなく，ICMP パケットを使用したエコーリプライを受信した他の IoT 機器も無かった．

IoT 機器の障害内容を含むログを図 13 に示す．

Heart-Beat-Time のタイムアウト時間を経過すると，オレンジ色のログが追加され，同一セグメント上の IoT 機器へ情報の要求が行われる．黄色のログは障害内容の切り分けが終了すると追加され，障害内容が表示される．

障害内容切り分けアルゴリズムにより”電源の遮断またはハードウェア以上の可能性”という適切な判断が行われた．

日時	内容
2022-01-05 08:23:03	[disconnect] <wifi_mac> 94:b9:7e:d9:71:b8 <msg> --
2022-01-05 08:23:03	[system] Y.T_ESP14同一セグメント上のIoT機器がBTとPINGを受信できません，電源遮断またはハードウェア異常の可能性．切断処理を行います.
2022-01-05 08:22:59	[warning] <wifi_mac> 94:b9:7e:d9:71:b8 <msg> HeartBeat timeout

図 13 電源遮断時のログ表示

ネットワーク障害

ネットワーク機器の設定誤りや故障を想定した実験を行う．本実験時は，プログラム動作中の IoT 機器の無線 LAN を切断する事で人工的にネットワークの障害を発生させた．

障害が発生し，Heart-Beat-Time のタイムアウト時間経過後，同一セグメント上に配置された他の IoT 機器から送信され，障害内容切り分けアルゴリズムで使用した情報を示す．

```
[2022−01−05 08:16:50] [system]
return_ping 94:b9:7e:d9:71:b8 <ng>7 <ok>0
[2022−01−05 08:16:52] [system]
return_bt 94:b9:7e:d9:71:b8 <ng>1 <ok>6
```

障害が発生した IoT 機器の Bluetooth 電波を IoT 機器 7 台中 6 台が受信し，ICMP パケットを使用したエコーリプライを受信した他の IoT 機器は無かった．

IoT 機器の障害内容を含むログを図 14 に示す．

日時	内容
2022-01-05 08:18:11	[disconnect] <wifi_mac> 94:b9:7e:d9:71:b8 <msg> KeepAliveTime timeout
2022-01-05 08:16:53	[system] Y.T_ESP14のBT電波を他の機器が受信,ネットワークに異常発生の可能性
2022-01-05 08:16:49	[warning] <wifi_mac> 94:b9:7e:d9:71:b8 <msg> HeartBeat timeout

図 14 ネットワーク障害時のログ表示

障害内容切り分けアルゴリズムにより”ネットワークに異常発生の可能性”という適切な判断が行われた．

ソフトウェア障害

IoT 機器で動作するソフトウェアにバグが含まれており，ソフトウェアが停止してしまう事を想定した実験を行う．本実験時は，プログラム内に 0 除算を行なう処理を追加する事で人工的にソフトウェアの障害を発生させた．障害が発生し，Heart-Beat-Time のタイムアウト時間経過後，同一セグメント上に配置された他の IoT 機器から送信され，障害内容切り分けアルゴリズムで使用した情報を示す．

```
[2022−01−05 06:58:46] [system]
return_ping 94:b9:7e:d9:71:b8 <ng>0 <ok>7
[2022−01−05 06:58:48] [system]
return_bt 94:b9:7e:d9:71:b8 <ng>0 <ok>7
```

障害が発生した IoT 機器の Bluetooth 電波を IoT 機器 7 台すべてが受信し，ICMP パケットを使用したエコーリプライも 7 台すべて受信した．

IoT 機器の障害内容を含むログを図 15 に示す．

日時	内容
2022-01-05 07:00:08	[disconnect] <wifi_mac> 94:b9:7e:d9:71:b8 <msg> KeepAliveTime timeout
2022-01-05 06:58:46	[system] Y.T_ESP14のPINGを他の機器が受信,ソフトウェアに異常発生の可能性
2022-01-05 06:58:45	[warning] <wifi_mac> 94:b9:7e:d9:71:b8 <msg> HeartBeat timeout

図 15 ソフトウェア障害時のログ表示

障害内容切り分けアルゴリズムにより”ソフトウェアに異常発生の可能性”という適切な判断が行われた．

実験結果から IoT 機器に電源，ネットワーク，ソフトウェアの障害を発生させた際に適切な IoT 機器の障害内容がブラウザ上に表示された．

本提案の接続台数に応じた切断遅延時間の推移

本実験では ESP32 を 2 台から 8 台まで 1 台ずつ増やし，電源遮断から切断までの遅延時間を測定する．IoT 機器の Hart-Beat-Time は 5 秒で固定し，接続してから電源を遮断するまでの時間を 0 秒から 20 秒まで 1 秒毎に測定し，測定結果の平均を取得する．結果を表 16 に示す．

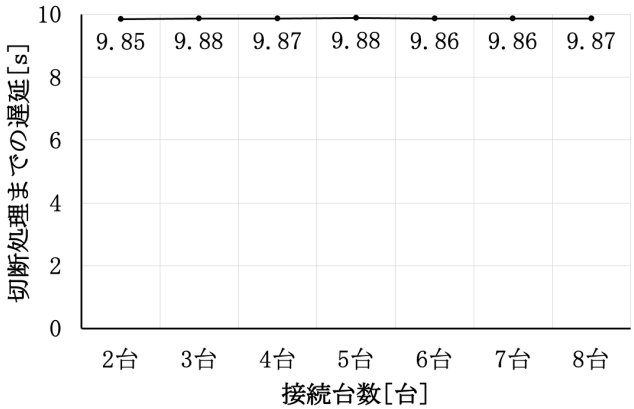

図 16 IoT デバイス台数毎の切断時間

本提案方式では，IoT 機器の接続台数が増加しても，IoT 機器の電源が遮断されてから，MQTT クライアント切断までの時間は増加しなかった．

本提案と既存手法の切断遅延時間の比較

本実験では既存の Keep-Alive-Time のみを切断判断に使用した，接続管理方法と本提案の IoT 機器電源遮断時の切断処理までの遅延時間を比較する．

既存の手法は Keep-Alive-Time を 30 秒に設定する．提案方式では IoT 機器の Hart-Beat-Time を 5 秒に設定する．1 台の IoT 機器の電源を遮断する．接続してから電源を遮断するまでの時間を 0 秒から 120 秒まで 1 秒毎に測定し比較を行う．

結果を図 17 に示す．

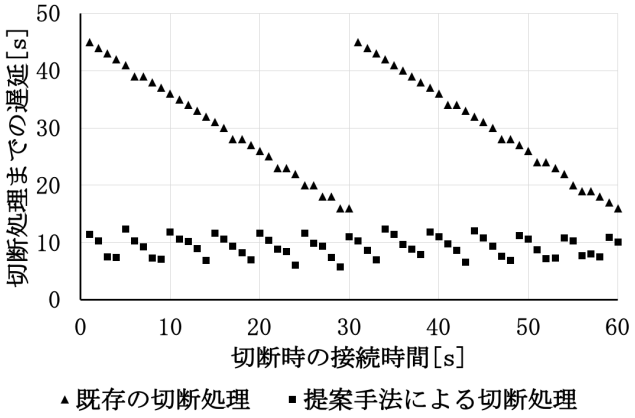

図 17 既存手法と提案手法の切断遅延時間の推移

遅延時間を表 5 に表す．

表 5 既存手法と提案手法の切断遅延時間

	既存手法	提案手法
最大遅延時間	45s	12s
最小遅延時間	15s	5s

提案手法は最大遅延時間，最小遅延時間共に既存手法よりも半分以下の時間で MQTT クライアントの切断を行う事ができた．

6. 議論

同一ネットワーク上の IoT 機器の台数が増加しても，検出遅延時間が増加しなかったのは，他の IoT 機器への情報の要求と取得を非同期で処理している為である．しかし，非同期処理は同一セグメント上の IoT 機器が一時的に情報を返却できなくなった場合，返却できなくなった IoT 機器の Hart-Beat-Time により，障害内容の切り分け終了までの時間が増加してしまう可能性がある．本提案方式では情報の返却ができない IoT 機器の Hart-Beat-Time が長い場合，検出遅延時間以外にも，情報取得待機時間が増加してしまう．現在，IoT 機器が接続する際の Keep-Alive-Time や Heart-Beat-Time を手動で設定しなければならない．今後は IoT 機器の設定値を自動的に適切な設定にする為に，センサーの情報送信間隔を活用した設定値決定アルゴリズムを作成していきたい．

7. おわりに

本研究では MQTT プロトコルを使用した，IoT システムが IoT 機器の電源遮断後，MQTT クライアント切断までに多くの遅延が発生する点と，障害が発生した際の IoT 機器の障害内容がわからない点を課題としている．この課題を解決する為に，MQTT 通信，ICMP パケットを使用したエコーリプライの有無，Bluetooth の電波発信状況を使用したハイブリッドな原因特定方法を提案した．これにより，MQTT ブローカーは障害が発生した IoT 機器の障害内容の切り分けを行い，障害内容を切断の判断材料として使用できる．提案手法は最大遅延，最小遅延共に既存手法よりも半分以下の時間で切断処理を行う事ができた．これによって，IoT 機器の障害を早期発見する必要のあるシステムに貢献可能である．

参考文献

[1] Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions, *Future Generation Computer Systems*, Vol. 29, No. 7, pp. 1645–1660 (2013).

[2] Granat, J., Batalla, J. M., Mavromoustakis, C. X. and Mastorakis, G.: Big Data Analytics for Event Detection in the IoT-Multicriteria Approach, *IEEE Internet of Things Journal*, Vol. 7, No. 5, pp. 4418–4430 (2020).

[3] Cerwall, P.: *Ericsson Mobility Report November 2021* (2021).

[4] Thota, P. and Kim, Y.: Implementation and Comparison of M2M Protocols for Internet of Things, *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD)*, pp. 43–48 (2016).

[5] Naik, N.: Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP, *2017 IEEE International Systems Engineering Symposium (ISSE)*, pp. 1–7 (2017).

[6] van der Westhuizen, H. W. and Hancke, G. P.: Practical Comparison between COAP and MQTT - Sensor to Server level, *2018 Wireless Advanced (WiAd)*, pp. 1–6 (2018).

[7] Palmieri, A., Prem, P., Ranise, S., Morelli, U. and Ahmad, T.: MQTTSA: A Tool for Automatically Assisting the Secure Deployments of MQTT Brokers, *2019 IEEE World Congress on Services (SERVICES)*, Vol. 2642-939X, pp. 47–53 (2019).

[8] Ni, K., Ramanathan, N., Chehade, M. N. H., Balzano, L., Nair, S., Zahedi, S., Kohler, E., Pottie, G., Hansen, M. and Srivastava, M.: Sensor Network Data Fault Types, Vol. 5, No. 3 (2009).

[9] Roughan, M., Griffin, T., Mao, M., Greenberg, A. and Freeman, B.: Combining Routing and Traffic Data for Detection of IP Forwarding Anomalies, Vol. 32, No. 1 (2004).

[10] Zhuang, S., Geels, D., Stoica, I. and Katz, R.: On failure detection algorithms in overlay networks, *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, Vol. 3, pp.

2112–2123 vol. 3 (2005).
[11] Cook, A. A., Mısırlı, G. and Fan, Z.: Anomaly Detection for IoT Time-Series Data: A Survey, *IEEE Internet of Things Journal*, Vol. 7, No. 7, pp. 6481–6494 (2020).
[12] Choi, J., Jeoung, H., Kim, J., Ko, Y., Jung, W., Kim, H. and Kim, J.: Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction, *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 610–621 (2018).
[13] Liu, Z., Qin, T., Guan, X., Jiang, H. and Wang, C.: An Integrated Method for Anomaly Detection From Massive System Logs, *IEEE Access*, Vol. 6, pp. 30602–30611 (2018).

