IoT データにおける優先度を考慮した拡張 MQTT 法に関する提案

遠藤繁之[1] 内田法彦[2] 柴田義孝[3]

概要： 近年，Internet of Things(IoT)サービスが著しく普及し，2021 年には，世界中の IoT デバイス数は 447.9 億台程まで増加すると予想されている．そして，IoT デバイスによるデータ通信量も年々増加傾向にあり，M2M 接続のシェアは 2018 年の 33%から 2023 年には 50%に拡大し，146 億の M2M 接続が見込まれている．そこで，IoT/M2M 通信向けのシンプル・軽量かつ省電力な通信プロトコルとして MQ Telemetry Transport(MQTT)が注目されている．しかし，MQTT にはメッセージの適時性を保証する機能は定義されておらず, IoT デバイス数が増加傾向にある近年において，利用者が重要なメッセージを迅速に取得できない可能性が指摘されている．そこで，本研究では，MQTTv5.0 で新しく追加されたメタデータ領域を利用し，IoT データにおける優先度を考慮した拡張 MQTT 法を提案する．提案手法では MQTT パケットの送信者と受信者の仲介の役割を行う Broker に対し，分類・優先度キュー，・送信制御の 3 つの機能を付与することで優先度制御を行い，優先度レベルは本機能を利用するユーザによって任意に決定される．本機能を利用するユーザによって，任意に設定するものとする．提案手法の有効性を検証するため，プロトタイプシステムを作成し，3 つの優先度レベルからなる IoT データを持つ IoT デバイスが存在する環境を想定し，評価実験を行なった．その結果，優先度レベルの高いメッセージほど遅延時間の改善が見られ，最も優先度レベルが高いメッセージの伝送遅延は平均で 48.2%の削減が確認でき，その有効性が示された．

Proposal of the Enhanced MQTT Methods with the Priority of IoT Data

SHIGEYUKI ENDO[1] NORIKI UCHIDA[2] YOSHITAKA SHIBATA[3]

1. はじめに

近年，Internet of Things(IoT)サービスが著しく普及し，2022 年には，世界中の IoT デバイス数は 348.3 億程度になると予測されている[1]．そして，IoT デバイスによるデータ通信量も年々増加傾向にあり，cisco 社の調査によると M2M 接続のシェアは 2018 年の 33%から 2023 年には 50%に拡大し，147 億の M2M 接続が見込まれている[2].

このような IoT デバイスの著しい増加により，近年，IoT/M2M 向けの通信プロトコルとして低容量・高頻度の通信に特化した MQ Telemetry Transport(MQTT)が注目されている．MQTT は 2013 年に国際標準化団体(OASIS)によって標準化され[3]，処理能力や使用電力が限定された機器や環境でも動作するよう，Broker を介してメッセージのやりとりを行う Publish/Subscribe 型を採用しており, IoT 通信への応用が期待されている．

しかし，MQTT の現状の仕様において，メッセージの適時性を保証する機能は定義されておらず，IoT デバイスが増加傾向にある近年において，利用者が重要なメッセージを迅速に取得できない可能性が指摘されている[4].
そのため，本研究では MQTTv5.0[5]で新しく定義された User Property[6]を利用し, IoT データにおける優先度を考慮した拡張 MQTT 法について提案し, 評価する. 具体的には，送信者は IoT データを送る際，User Property と呼ばれるメタデータ領域に優先度情報を書き込み，中継者である Broker へ送信する．Broker は送信者から IoT データが含まれたメッセージを受けると大きく 3 つの機能に分けて処理を行う．

第一に，メッセージ内のメタデータ領域を参照し，メッセージの分類を行う機能．次に，分類されたメッセージを Priority Queue へ格納し，一番優先度の高いメッセージの選出を非同期に行う機能．そして，最後に，Priority Queue から取り出したメッセージを非同期にクライアントへ送信する機能である．

特に，非同期で動作する 2 つの機能に関しては，イベント駆動型で動作するため，余分な処理を行わないことによる Broker の処理速度の向上と, Broker のリソース軽減が期待できる. また, 非同期に送信制御を行うメリットとして，一度の大量のメッセージを受け取っても送信処理に与える影響を最小限にしつつ，受信者へメッセージの送信を行える．

以降，本稿では，第 2 章で MQTT の概要および MQTT の QoS 制御機能に関する既存研究について述べ，第 3 章では，提案手法である IoT データ優先度を考慮した拡張 MQTT 法について，主な 3 つの機能である Classification, Priority Queue, Priority Control について具体的に説明する．第 4 章では，提案手法の評価実験を行い，評価結果を述べる. そして最後に第 5 章で, まとめと今後の課題を述べる．

1 福岡工業大学大学院
Graduate School of Communication and Information Networking, Fukuoka Institute of Technology
2 福岡工業大学
Fukuoka Institute of Technology
3 岩手県立大学
Iwate Prefectural University

2. 既存研究

2.1 M2M/IoT 向け通信手法

近年，M2M/IoT 通信が求めるシンプル・軽量かつ省電力な通信プロトコルとして AMQP, CoAP, MQTT 等のプロトコルが注目されている[7][8][9]．その中でも MQTT は，HTTP のようなリクエスト・レスポンス型を用いず，Broker を介して通信を行う Publish/Subscribe 型を採用しており，IoT 分野において広く採用されている．

Publish/Subscribe 型は，図 1 左側のメッセージ送信者である「Publisher」が，中央のメッセージ中継者である「Broker」に対し，Topic と呼ばれる階層構造の識別子を付与したメッセージ配信を行う(PUBLISH)．Broker は右側のメッセージ受信者である「Subscriber」から Topic Filter と呼ばれる受信したい Topic の情報を予め登録しているため(SUBSCRIBE)，対応する Subscriber に対しメッセージを転送するといったフローをとる．例えば図 1 では，publisher で生成され転送された A/temp という Topic が，2 つの Subscriber からそれぞれ A/temp と B/temp という Topic で SUBSCRIBE された時，A/temp のみ PUBLISH する．

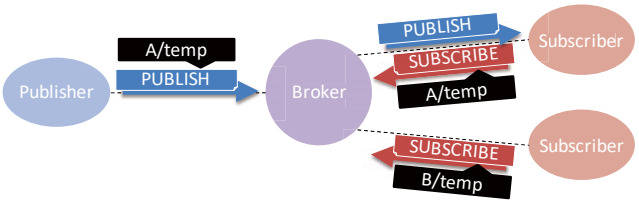

図 1. Publish/Subscribe 型の概要

また，MQTT 独自の特徴として，固定ヘッダーが最小 2[Byte]と小さいことから高頻度のショートパケット通信に適していることや，図 2(a), (b), (c)に示すような，メッセージの重要度によってメッセージの到達保証を行う QoS の機能が存在し，メッセージの重要度や帯域の状況を加味してメッセージ配信の QoS をクライアントが設定できることなどから，M2M/IoT 向け通信プロトコルとして広く活用されていると考えられる．

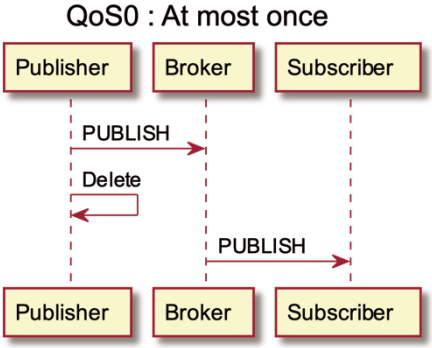

図 2(a). QoS0 の通信フロー

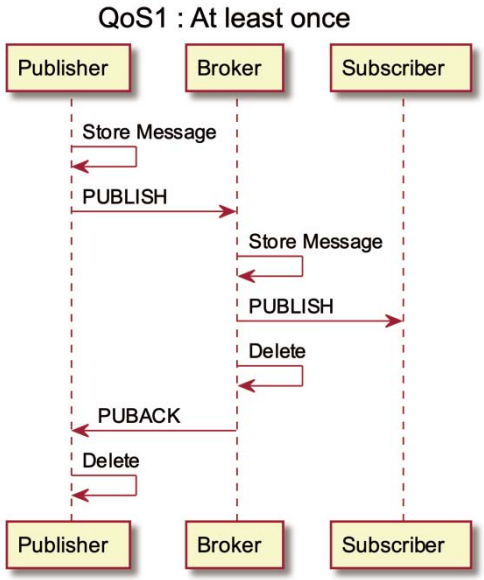

図 2(b). QoS1 の通信フロー

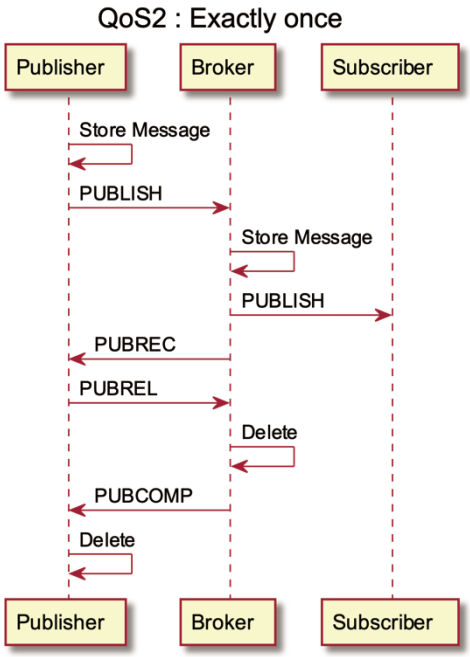

図 2(c). QoS2 の通信フロー

図 2(a)に示す QoS0 は，再送制御を行わない最大一回の送信となるため，通信エラー等で転送が失敗すると，メッセージが損なわれる可能性がある．また，図 2(b)に示す QoS1 は，Broker から PUBACK と呼ばれるメッセージが届いたことを確認するパケットを受け取るまで送信側に対し再送を行うため，最低でも一回のメッセージ送信が行われる．しかし，QoS1 では PUBACK が受信できない場合は再度送信を行うため，メッセージが重複する可能性がある．そのため，QoS2 では PUBACK に該当する PUBREC の確認応答として PUBREL を送信することで，送信できたことが確認できるので，Publisher に対し PUBCOMP を返し，保証

性を高める設計となっている．

しかし，図 2(a)(b)(c)の MQTT における QoS 機能はメッセージの到達保証しか行わず，メッセージの重要度に応じた順序保証などは行われない．そのため，IoT デバイスが急増する近年において，重要度の低い IoT トラヒックで帯域を使用し，重要なメッセージを適切なタイミングで送信できない可能性がある．

そのため，近年では MQTT に優先度制御機能を付与する研究が行われてきた．

2.2 p-MQTT・P-MQTT

これまでも MQTT の QoS 機能について注目し，MQTT に優先度制御機能を付与する関連研究がいくつか行われている．第 1 に，Yong-Seong Kim らによる，MQTT Control Packet を用いた，MQTT に優先度制御機能を付与する p-MQTT[10]が提案している．これは，MQTTv3.1.1 時点で予約済みとなっていた 2 つのフィールドをそれぞれ Urgent, Critical パケットと設定することで，メッセージ分類を行なっていた．しかしながら，この提案手法では，MQTT のバージョンアップによって，予約済みだったフィールドが認証機能の追加で使用されたため，再現することが不可能となった．

第 2 に，内山らは Topic に優先度 Topic と呼ばれる，優先度 Topic 識別用文字列と優先度の段階を示す数値を通常の Topic に追記することで，使用者が任意に優先度レベルを設定できる P-MQTT[11]を提案している．具体的には，図 3 に示すように，通常の Topic「A/temp」に優先度 Topic を追記することで，「A/temp/priority/1」という記述となる．

しかし，P-MQTT は Broker-Subscriber 間で優先度 Topic がオーバーヘッドになることが懸念されており，その解決法として，図中で示すように，Broker 内で優先度 Topic が強制的に取り外されて Subscriber へ転送する設計となっている．そのため，Subscriber がどのメッセージが優先度の高いものだったのか判断することが難しく，多様な環境が想定される IoT アプリケーションへの十分な対応ができているとは言えない．

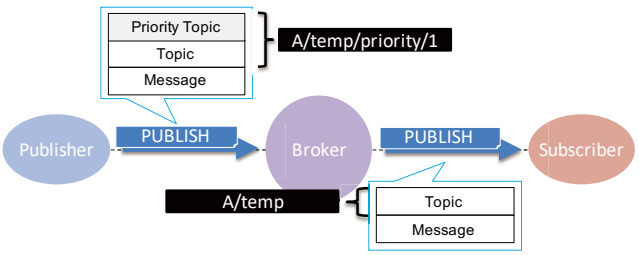

図 3. P-MQTT の概要

そこで，本研究では MQTTv5.0 でメタデータ領域として新しく追加された User Property を用いてメッセージの分類分けを行い，後述する優先度制御を行う手法を提案する．

3. 提案手法

3.1 データ優先度を考慮した拡張 MQTT 法

本研究では，IoT デバイスが増加傾向にある近年において，MQTT を利用するサービスに対し，IoT データの持つ優先度に着目し，データの重要度や優先度に応じて順序保証を行う IoT データにおける優先度を考慮した拡張 MQTT 法を提案する．

システム環境としては，温湿度センサや照度センサを搭載した複数の IoT デバイスが設置されている環境を想定する．この空間では，温湿度や照度といった情報に加え，時間情報などが記された IoT データが発生しており，これらの IoT データを活用することで生産業向けの空調環境を制御することができる．このようなアプリケーションに提案する拡張 MQTT 法の適用する際のシステム全体像を図 4 に示す．Publisher としては Raspberry PI のようなセンサーデバイスを想定し，Broker としては高処理能力を有するサーバ機，Subscriber としては PC や携帯電話のようなモニタリングシステムを想定している．また，想定システムでは，空間の広さや規模に応じて，処理するデータ量が変動することが予想され，1 つの異常発生によって，一部のエリア全てが異常値を検出することが考えられる．つまり，発生するデータの緊急性やアプリケーションの要求によって処理や転送フローを柔軟に拡張できる機能が求められる．

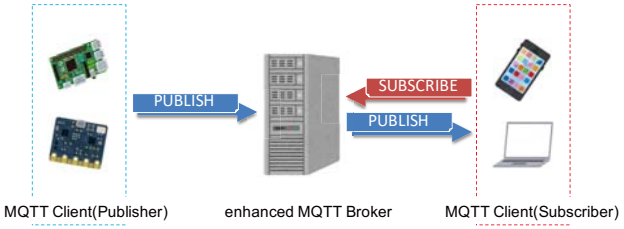

図 4. 提案手法適用例

そして，本提案手法を実現するために，Publisher が PUBLISH する際にメタデータ領域に優先度情報を書き込むことに加え，Broker に対して以下に示す 3 つの拡張機能を提案する．

① Classification
メタデータ領域内の優先度情報を基にメッセージの分類を行う機能

② Priority Queue
上記の機能から転送されてきたメッセージを対応する優先度別 Queue へ格納する機能

③ Priority Control
Priority Queue からのメッセージの取り出し，及び Subscriber への送信処理を行う機能

図 5 に，拡張 MQTT 法を採用した場合における動作フローを示す．MQTT Broker は，Publisher から PUBLISH メ

ッセージを受け取ると，Classification メソッド内で User Property の値に基づいた優先度情報とメッセージ情報を Priority Queue へ転送する．Priority Queue メソッドでは，Classification メソッドから受け取った情報を基に，対応するキューへの格納処理と，Priority Queue 内のキューイング処理を非同期に行う．その後，Priority Control メソッド内で，Priority Queue から，その時点で一番優先度の高いメッセージを取り出す処理から，Subscriber に対しての PUBLISH 処理までを非同期に行う．

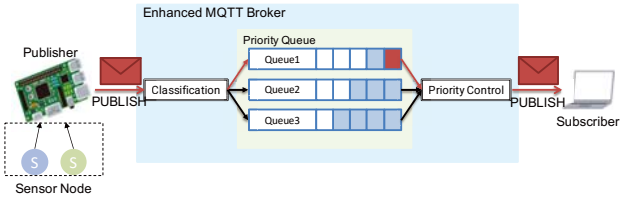

図 5. 提案手法動作フロー

各機能の詳しい動作については，3.2 節から順に詳細を述べる．

3.2 Classification

Broker に追加する 1 つ目の機能は，Publisher から送信されてきたメッセージ内の User Property を基にメッセージの分類を行う機能である．User Property は「key」と「value」の 2 つの値の組を持つ UTF-8 String Pair 型で構成されるため，Publisher は図 6 に示すように，User Property 内に priority が key 値となった優先度情報を付与することで，Broker はその値を参照してメッセージを分類し，後述する Priority Queue へメッセージを転送する処理を行う．

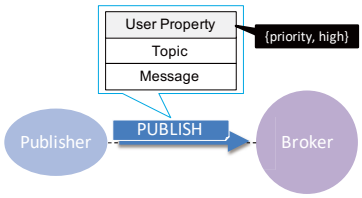

図 6. メタデータ適用例

また，このとき User Property に値が設定されていないメッセージは一番優先度の低い Queue へ格納し，{priority, clear}といったメッセージを受信した場合は図 7 に示すように User Property の中身を取り外し，User Property による Broker-Subscriber 間の通信量を増やさない設定も利用者によって任意に選択できる．

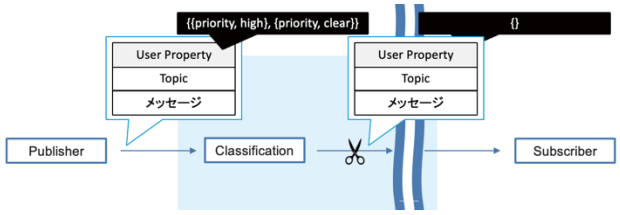

図 7. メタデータ領域を転送しない場合のフロー

3.3 Priority Queue

Broker に追加する 2 つ目の機能は，前述の Classification 機能により，分類・転送されたメッセージを各 Queue へ格納する機能である．本機能の特徴として， Queue にメッセージが格納された際に MQTT Broker を制御するプロセスとは別プロセスで，イベント駆動型でキューイング処理を行い，一番優先度の高いメッセージの特定を行う．これにより，送信制御に要する時間の短縮と，Broker のリソース軽減が期待できる．

3.4 Priority Control

Broker に追加する 3 つ目の機能は，Priority Queue への問い合わせから Subscriber への送信処理までを非同期に行う機能である．本機能もキューイング処理と同様に，イベント駆動型で非同期に動作するため，前節の Priority Queue の動作を含めると図 8 に示すようなフローとなる．

① 分類されたメッセージを対応するキューへ格納
② 最も優先度の高いメッセージの選出
③ 選出されたメッセージを Priority Control へ渡す
④ Subscriber への送信制御

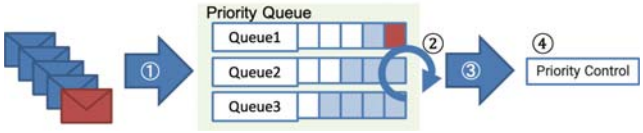

図 8. イベント制御フロー

このような非同期の制御フローを取ることで，PUBLISH 等の受信処理が大量に発生した場合においても送信処理に影響を与えずに，Subscriber への PUBLISH 処理を行うことができる．

4. 実装・評価実験

4.1 実験環境

拡張 MQTT 法の有効性を検証するため，プロトタイプシステム実装を行い，実験と評価を行った．

実験環境を図 9 に示す．実験では Publisher, Broker, そして Subscriber は Raspberry Pi3 Model B 上で Golang で実装し，各機器は同一 LAN 内の有線ネットワークで接続されている．また，本実験に用いるシステム環境を表 1 に示す．Publisher 端末は同一の端末上にマルチスレッドで 30~150 台の Publisher Client を発生させ，各 Client はセンサデータや優先度情報を含めた，90 [Byte]のメッセージを一回ずつ発生する．この時，優先度の種類は高・中・低の 3 種類とし，最終的に優先度の比率が 1: 1: 1 になるよう調整する．

そして，Publisher がメッセージの送信開始を行った時刻と Subscriber がメッセージを受信した時間差を伝送遅延時間として，従来手法と提案手法の計測結果を比較する．

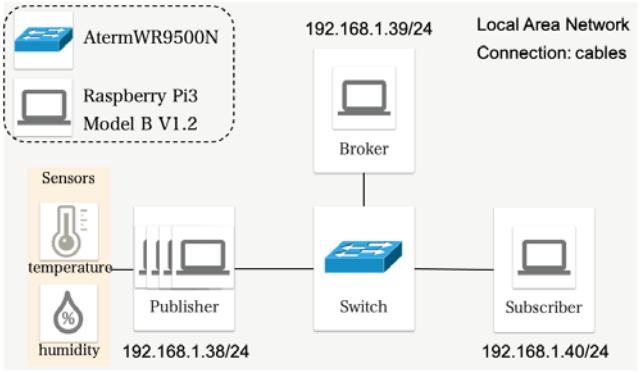

図 9. 実験構成

表 1. 実験パラメータ

Parameter	Value
Number of publishers	30-150
Data rate	100Mbps
Size of message	90Byte
Total number of messages per publisher	1
Number of priorities	3
Publish QoS level	1

4.2 実験・評価

実験では Publisher は同一の端末上にマルチスレッドで 30-150 台の Publisher を発生させ，Broker に対し同時に接続を確立した後，一斉に PUBLISH し，Subscriber が受信するまでの伝送遅延時間を計測した．また，提案手法ではメッセージ優先度の高: 中: 低を，1: 1: 1 の比率となるよう PUBLISH し，既存の MQTT 法と比較した．

そして，優先度別の伝送遅延時間について，優先度高のグラフを図 10(a)に，優先度中のグラフを図 10(b)に，優先度低のグラフを図 10(c)に示す．

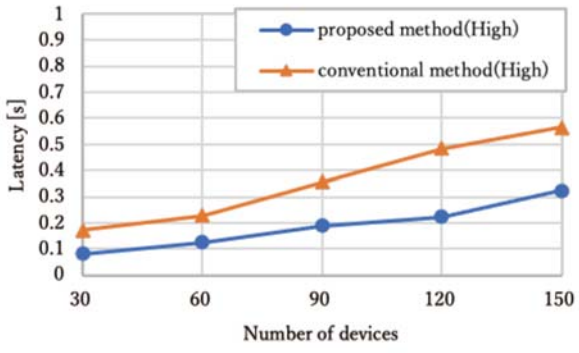

図 10(a) 優先度高メッセージの伝送遅延

図 10(a)では，Publisher30 台の場合において，提案手法と従来手法の伝送遅延の差は約 0.1[s]に対し，Publisher150 台の場合では伝送遅延の差が約 0.2[s]と，Publisher の数が増えるにつれて伝送遅延の差が広がっているため，提案手法が Publisher 数の増加とともに効果を発揮することが分かる．

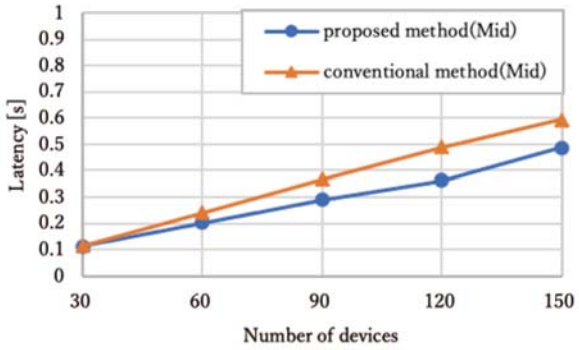

図 10(b) 優先度中メッセージの伝送遅延

また，図 10(b)の場合においても同様に，Publisher30 台の場合では伝送遅延の差はほとんど見られないが，Publisher150 台の場合では伝送遅延の差が約 0.1[s]と，Publisher 数の増加に従って伝送遅延の差が広がっていることが確認できる．

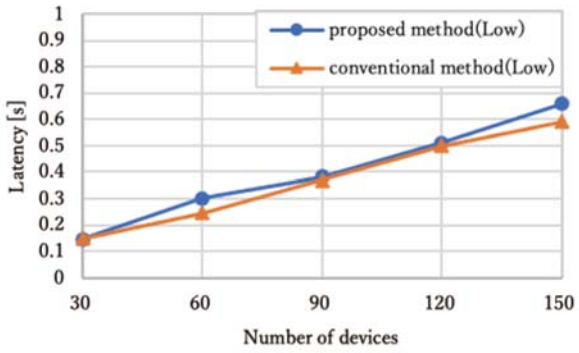

図 10(c) 優先度低メッセージの伝送遅延

そして，図 10(c)の優先度が低いメッセージに関しては，既存手法より伝送遅延が若干大きいものの，ほとんど差はなかった．したがって，提案手法では，優先度高・中のメッセージ伝送を効率的に行なっただけでなく，優先度低のメッセージ伝送に関してもほとんど違いは見られなかったことが示されている．これは，Priority Queue がメッセージを受け取った際に発生するキューイング処理がオーバーヘッドとなっているものの，処理時間が微小であったため，以上のような結果になったと考えられる．

以上の結果から，本提案手法は既存手法と比較して，優先度の高い IoT データの遅延時間を減少させることを示し，効果的に，優先度の高いデータを到達することが可能であると考えられる．また，本提案手法では，メタデータ内の優先度情報を消去する機能を用いることで，Broker-Subscriber 間の帯域を効率的に使用することができるため，更に伝送遅延を短くすることも可能であったことが考えられる．

5. おわりに

本研究では，IoT デバイスが増加傾向にある近年において，順序保証が行われないという MQTT の QoS の課題に対し，IoT データにおける優先度を考慮した拡張 MQTT 法の提案をした. さらに, 提案手法の有効性を検証するため, Golang を用いてプロトタイプシステムを実装し，評価を行った．その結果，Publisher 数 30-150 台と変化させつつ，優先度{High, Mid, Low}の比率が同じという状況において, 優先度高・中の情報を持つメッセージは Publisher 数が増加するに従って従来手法との伝送遅延の差が広がっていることが確認でき，その有効性が示された．また，優先度低の場合においても，Publisher 数に関係なく，同程度の性能を発揮することを確認した.

今後の課題としては，優先度比率を変化させた場合の評価や，Publisher 数を増加させた場合，通信障害を想定した際のデータ到達率の評価についても計画している．また，より複雑なネットワーク状況を想定した実験や，複数の Publisher が継続的に大量の PUBLISH メッセージを発生する状況，帯域に制約がある条件下といったシナリオでの評価実験を検討している.

参考文献

[1] 総務省：情報通信白書令和2年版，総務省(オンライン)，入手先<https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r02/html/nb000000.html> (参照2021-1-20).

[2] Cisco Systems G.K. : Cisco Annual Internet Report (2018–2023) White Paper，Cisco Systems(オンライン)，入手先<https://www.cisco.com/c/ja_jp/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html> (参照 2021-1-20).

[3] OASIS Standard Incorporating Approved Errata 01 : MQTT Version 3.1.1 Plus Errata 01，OASIS Open(オンライン)，入手先<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqttv3.1.1.pdf> (参照 2021-1-8).

[4] Dipa Soni, Ashwin Makwana: A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT), INTERNATIONAL CONFERENCE ON TELECOMMUNICATION, POWER ANALYSIS AND COMPUTING TECHNIQUES (ICTPACT - 2017).

[5] OASIS Standard : MQTT Version 5.0，OASIS Open(オンライン)，入手先<https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html> (参照 2021-1-8).

[6] Florian Raschbichler : User Properties - MQTT 5 Essentials Part 6，HiveMQ(オンライン)，入手先<https://www.hivemq.com/blog/mqtt5-essentials-part6-user-properties> (参照 2021-1-20).

[7] AMQP，AMQP.org(オンライン)，入手先<https://www.amqp.org> (参照 2021-1-8).

[8] CoAP，CoAP.org(オンライン)，入手先<https://coap.technology/> (参照 2021-1-8).

[9] MQTT，MQTT.org(オンライン)，入手先<https://mqtt.org/> (参照 2021-1-8).

[10] Yong-Seong Kim, Hwi-Ho Lee, Jung-Hyok Kwon, Yong Sin Kim, Eui-Jik Kim: Message Queue Telemetry Transport Broker with Priority Support for Emergency Events in Internet of Things, Sensors and Materials, Vol.30, No.8.

[11] 内山 仁, 嶋野博史: 優先度を考慮した送信制御が可能な P-MQTT の開発と評価，マルチメディア，分散，強調とモバイルシンポジウム(DICOM2019).