一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
NS2023-78(2023-10)

分散MQTTブローカにおける Server redirectionを用いたクライアント振り分け機構

吉村　佳祐†　　坂野　遼平†

† 工学院大学　〒192-0015 東京都八王子市中野町 2665–1

あらまし　近年 IoT デバイスが急速に普及している．IoT デバイスで多く使用されている通信プロトコルとして MQTT がある．MQTT は非常にシンプルなメッセージプロトコルで Publish/Subscribe モデルであることから疎結合性を備えているが， Broker や Subscriber に負荷が集中することが問題視されているため負荷分散が求められている．しかし，既存の負荷分散方法では Broker の負荷状況に応じた振り分けや Publisher の送信頻度に応じた振り分けが困難であり，単一障害点が発生する問題がある．本研究では既存手法の問題点の改善のため MQTT v5.0 で規定された Server redirection を利用した負荷分散手法を提案する．また提案手法の有効性を検証するため既存手法である DNS ラウンドロビンとの比較実験を行い評価指標として各 Broker のスループットと CPU 使用率を測定した．

キーワード　IoT，MQTT，Pub/Sub

Client Allocation Mechanism Using Server Redirection in a Distributed MQTT Broker

Keisuke YOHIMURA† and Ryohei BANNO†

† Kogakuin University　2665–1 Nakano-machi, Hachioji-shi, Tokyo, 192–0015 Japan

Abstract　MQTT is a very simple message protocol with a publish/subscribe model, which makes it loosely coupled. However, load balancing is required because of the problem of concentrated load on Brokers and Subscribers. However, existing load balancing methods have difficulty in distributing the load according to the Broker's load status and the Publisher's sending frequency, and there is a problem of single point of failure. In this study, we propose a load balancing method using server redirection specified in MQTT v5.0 to improve the problems of existing methods. In order to verify the effectiveness of the proposed method, we conducted comparison experiments with DNS round-robin, an existing method, and measured the throughput and CPU utilization of each Broker as evaluation indices.

Key words　IoT，MQTT，Pub/Sub

1. はじめに

近年 IoT デバイスが急速に普及しており，総務省の令和 4 年度版情報通信白書 [1] では 2024 年には 398.5 億台に拡大すると予想されている．IoT デバイスを用いた情報収集に使用されるプロトコルとして HTTP [2]，WebSocket [3] などが挙げられるがこれらのプロトコルは 1 対 1 のプロトコルであるために大規模な IoT 情報収集には不向きと言える．また，ヘッダサイズが大きいこともありデータサイズが大きくなることで通信帯域を圧迫してしまう可能性があった．その中で，近年では MQTT プロトコル [4] が注目を集めている．マルチキャストが可能で多対多の通信が行える．またヘッダサイズが小さいことから通信帯域を圧迫しないため，MQTT は HTTP などより大規模な IoT 情報収集に向いている．MQTT では Broker や Subscriber に負荷が集中することが問題視されており，負荷分散のために様々な手法が提案されている．しかし既存の負荷分散手法では Broker の負荷状況に応じた振り分けや Publisher の送信頻度に応じた振り分けが困難であった．

本研究では研究目的として既存の負荷分散手法に生じていた問題である，Publisher の送信頻度や Broker の負荷状況に応じた振り分けの実現と単一障害点の改善を目的とする．

本論文は全 5 章構成である．1 章では研究背景や目的を述べた．以降，2 章では既存研究と関連技術について解説する．3 章では提案手法について，4 章では実験環境から結果を述べ，その結果の考察をする．最後に 5 章では本論の総括と今後の展望，課題について述べる．

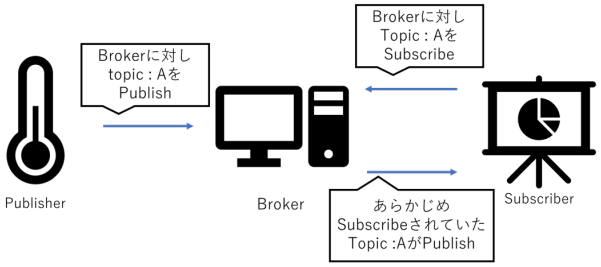

図 1　MQTT の構成

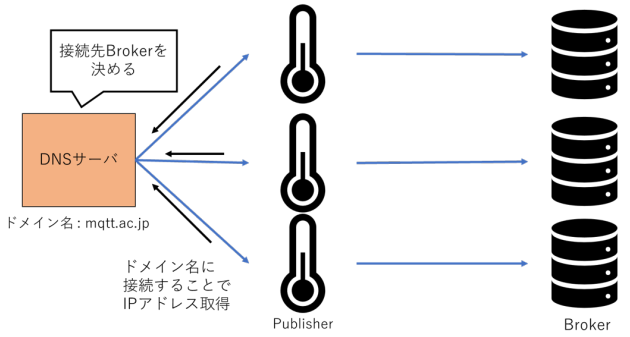

図 2　DNS ラウンドロビンの構成

2.　関 連 研 究

2.1　MQTT

MQTT は Publisher，Broker，Subscriber の 3 つの役割で構成される．また MQTT の特徴として非常にシンプルなメッセージプロトコルであり軽量なプロトコルであるため消費電力が小さく，通信帯域を圧迫しないこと，Publish/Subscribe モデルであることから疎結合性を持つことが挙げられる．MQTT の仕組みを図 1 に示す．MQTT では Publisher は Topic というメッセージに紐づけした情報を付与したまま Broker に送信し，Broker はそのメッセージを Sbscriber へ転送する．Subscriber は購読したいメッセージに対応する Topic を Broker にあらかじめ Subscribe する．Broker は Subscriber が購読した Topic のメッセージを受信した時に Subscriber に送信する．MQTT は大まかにこのような仕組みで動いている．

2.2　DNS ラウンドロビンと Shared Subscription を用いた負荷分散手法

Publisher から Broker にむけて大量にデータを送信する時遅延の増大やパケットの損失が生じる可能性がある．芳澤らの手法 [5] では DNS ラウンドロビンを用いて Broker の受信負荷を分散し，Shared Subscription を用いて Subscriber の受信負荷を分散させることで，高いスループットを生み出すことに成功していた．

2.2.1　DNS ラウンドロビン

DNS ラウンドロビンは製品としても出回っており [6]，メジャーな負荷分散システムの一つと言える．DNS ラウンドロビンの仕組みの概要を図 2 に示す．DNS ラウンドロビンでは DNS サーバがドメイン名と IP アドレスを対応付けた表を持ち，クライアントからの問い合わせに対し問い合わせられたドメイン名に対応する IP アドレスを返答する．クライアントは DNS サーバから返答された IP アドレスに接続することで負荷分散を行っている．メリットとしては実装が容易であり，またサーバの追加を行う際に DNS サーバの設定ファイルを変更するだけでよいなどが挙げられる．デメリットとしては DNS サーバはドメイン名に対応した表の順番通りにしか振り分けることができないため，Broker の負荷状況に応じたクライアントの振り分けができない．また，Broker に問題が生じた時，検知せずに通信を続ける．サーバの追加による DNS の設定変更の際にキャッシュによるタイムラグが発生するなどが挙げられる．

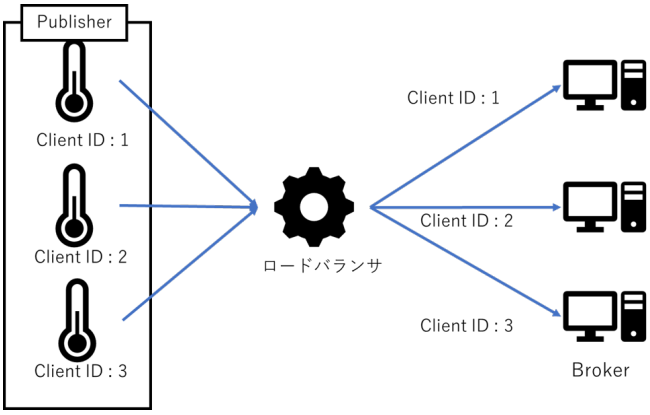

図 3　ロードバランサの構成

2.3　ロードバランサと Broker のクラスタ化を用いた負荷分散手法

前述の DNS ラウンドロビンなどの問題点の改善のため Jutadhamakorn らの手法 [7] では Publisher-Broker 間にロードバランサを使用し，Broker 同士をクラスタ化することでよりスケーラブルな負荷分散を実現した．

2.3.1　ロードバランサ

ロードバランサは製品としても出回っている [8] メジャーな負荷分散システムの一つである．ロードバランサの仕組みの概要を図 3 に示す．ロードバランサではクライアント接続時にクライアント ID をハッシュとして保存しクライアント ID ごとに Broker へ割り振る．その後クライアントから送られるメッセージはロードバランサを経由して先ほど割り振られた Broker に届けることで負荷分散を行っている．メリットとして DNS ラウンドロビンより高性能な負荷分散が規定できる．また，ロードバランサ以下の機器の異常によるシステムの停止が起きないことが挙げられる．デメリットとして全通信がロードバランサを経由するためロードバランサが単一障害点となっている．また，大規模なネットワークであるほど高性能なロードバランサであることが求められる．Publisher-Broker 間に中継地点が増えるため遅延が増大してしまうことが挙げられる．

3.　提 案 手 法

提案手法の概要を図 4 に示す．Server redirection を実装したディスパッチャ (以下 MQTT Dispatcher と呼称) を用意する．MQTT Dispatcher は各 Broker から CPU 使用率を取得し，負荷

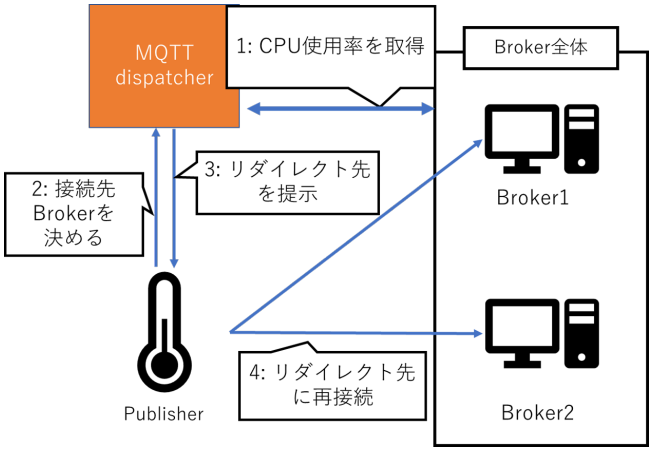

図 4　提 案 手 法

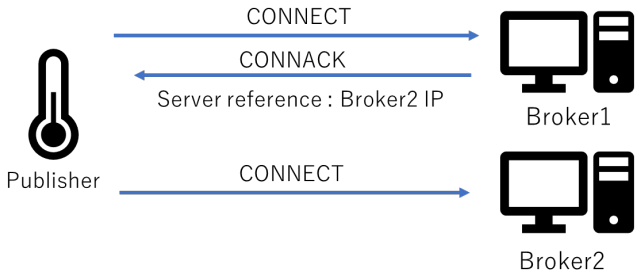

図 5　Server redirection の構成

表 1　マシンスペック

	Publisher，Broker
CPU	Celeron N3350
Memory	4GB
OS	Ubuntu20.04

表 2　Publisher のパラメータ

	Publisher1	Publisher2
台数	9 台	1 台
ペイロードサイズ	600byte	600byte
送信頻度	2000msg/sec	10000msg/sec
動作時間	100 秒	100 秒

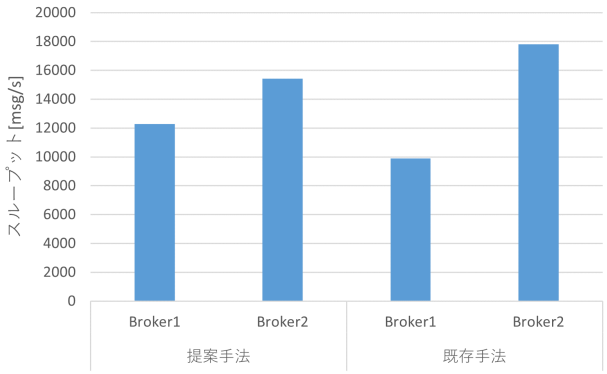

図 6　Publisher-Broker 間のスループット

が低い Broker と接続するように Server redirection を利用して Publisher の接続先を誘導する．各 Broker から得た CPU 使用率はソケット通信を用いて MQTT Dispatcher に送信し，MQTT Dispatcher 上で 5 秒間平均を求める．クライアントから接続があった際には，各 Broker のうち直近の平均値が最も低い Broker を Server redirection のリダイレクト先に設定する．

3.1　Server redirection

Servr redirection は MQTT version 5.0 [9] にて規定された新機能である．Server redirection の仕組みの概要を図 5 に示す．Server redirection とはクライアントが Broker に接続した際の応答メッセージに他の Broker の情報を記載する仕組みである．

4.　実　　験

4.1　実 験 環 境

本実験で使用したマシンのスペックを表 1 に示す．Publisher は 2 種類用意し合計 10 台，Broker は 2 台使用し，Subscriber の実装はしないものとした．また測定ツールとして MQTTLoader v0.8.2 [10] を使用した．Broker の実装には Mosquitto [11] を使用した．また CPU 使用率の取得には自作のプログラムを使用した．Broker は Linux ベースの Ubuntu 上で実装しているので Linux の/proc/stat の出力情報を使用して 1 秒ごとに CPU 使用率を算出した．

4.1.1　Publisher のパラメータ

用意した 2 種類の Publisher のパラメータ設定を表 2 に示す．

4.2　評 価 指 標

本実験では Publisher-Broker 間のスループットと各 Broker の CPU 使用率の推移の評価を行った．

4.3　実 験 方 法

本研究では提案手法と DNS ラウンドロビンを用いた手法（以下既存手法と呼称）の計 2 手法で比較した．提案手法の実験構成は図 4 と同様，既存手法の実験構成は DNS ラウンドロビンで行われるような振り分け方になるように Publisher の接続先を予め設定して行った。．また，Publisher は約 5 秒間隔で追加し追加順はランダムに決定した．

4.4　実 験 結 果

Publisher-Broker 間の平均スループットを図 6 に示す．各 Broker の CPU 推移の結果のうち Publisher2 を 2 番目に加わったケースを図 7 と図 8 に，4 番目に加わったケースを図 9 と図 10 に示す．スループットに関して提案手法は既存手法と比べ，Broker 間のスループットの差が少ない結果となった．

4.5　実験結果を踏まえた考察

Publisher-Broker 間のスループットに注目すると提案手法の各 Broker 間のスループットの差は約 3000msg/s ほどであるのに対し既存手法は約 8000msg/s であることが分かる．これより各 Broker にかかる負荷が既存手法より均一になっているので提案手法のほうが効率的な負荷分散ができたと言える．

Publisher2 を 2 番目に加えたものの CPU 使用率の推移に注目すると提案手法のほうが既存手法より計測時間全体で CPU 使用率の差が少ないことが分かる．つまり Broker にかかる負荷が

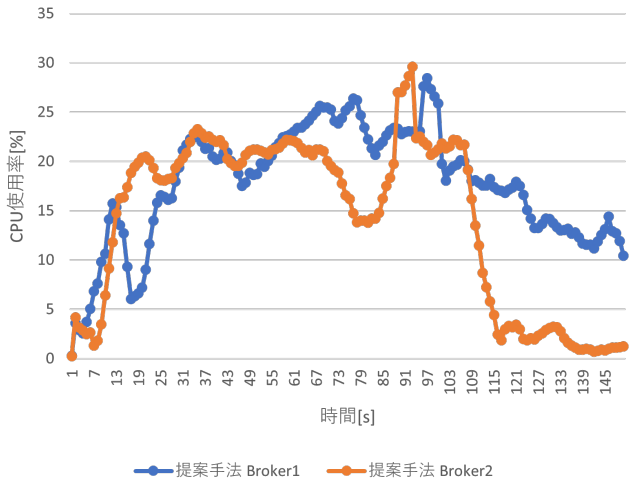

図 7　2 番目に加えた場合の CPU 使用率 (提案手法)

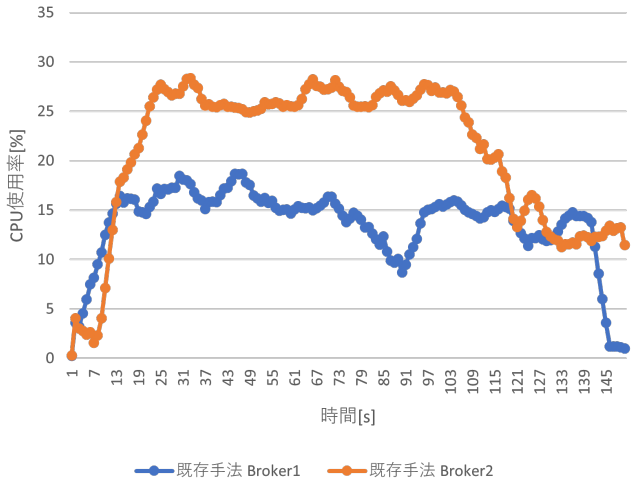

図 8　2 番目に加えた場合の CPU 使用率 (既存手法)

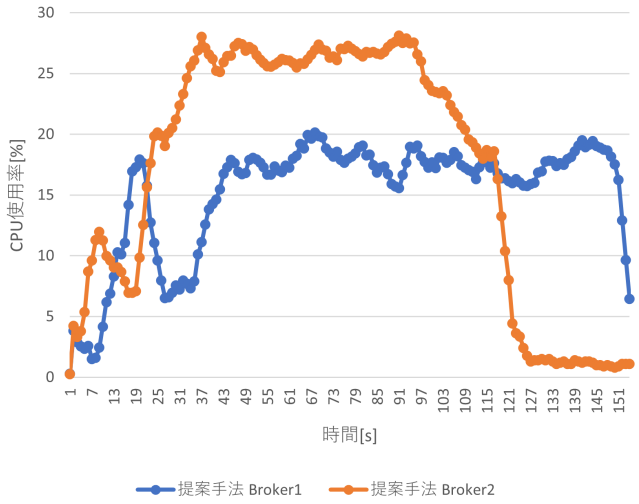

図 9　4 番目に加えた場合の CPU 使用率 (提案手法)

既存手法より均一化できたと言える．これの理由として高負荷の Publisher の追加が早かったため，高負荷の Publisher が追加された時にできた各 Broker 間の差を後に追加された Publisher

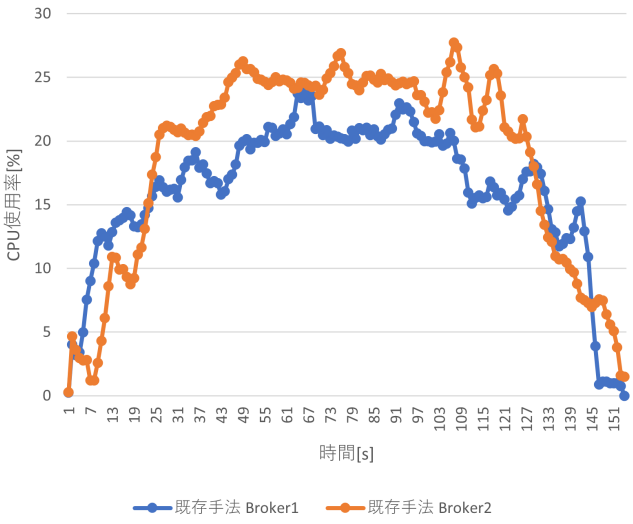

図 10　4 番目に加えた場合の CPU 使用率の推移 (既存手法)

で埋めることができたからだと考える．

Publisher2 を 4 番目に加えたものの CPU 使用率の推移に注目すると既存手法のほうが計測時間全体で CPU 使用率の差が少ないことから提案手法のほうが Broker にかかる負荷をより均一化出来たと言え，提案手法の有効性を示せていないように見える．

しかし，提案手法では高負荷の Publisher を加えたと考えられる 20 秒付近から Broker2 の負荷が急激に上昇し加えられた Publisher の計測時間が終了するであろう 120 秒付近では急激に減少しその後停滞することから高負荷の Publisher が加えられた後に新規 Publisher が追加されていないことが分かる．さらに高負荷の Publisher が加えられたことを CPU 監視システムが反映していることが分かり，Publisher の送信頻度を CPU 使用率で疑似的に表現できていると言える．つまりここでは研究目的であった Publisher の送信頻度や Broker の負荷状況に応じた負荷分散が行えたと言える．

CPU 使用率の推移の結果を比べるとどちらの順番でも既存手法の Broker の 50 秒から 100 秒の間は実質同程度の負荷がかかっているはずであるが CPU 使用率の推移が大きく異なる．ここから CPU 使用率のみの負荷指標では一定した負荷指標とするには不足していると考えることもできる．

また本研究の弱みとしては複数 Publisher による同時多発的な dispatcher への接続がウィークポイントとなってしまっているため Publisher の追加をする際に 1 台ごとに時間を開けなくては振り分けの際に最も有効的な結果が得られなくなる可能性があることである．

5. おわりに

5.1 まとめ

本研究では Broker の負荷状況や Publisher の送信頻度に応じた負荷分散のために、MQTT v5.0 にて規定された Server redirection を使用した負荷分散手法を提案した．この手法の有効性の検証として既存手法である DNS ラウンドロビンと比較実験

を行い Publisher-Broker 間のスループットと各 Broker の CPU 使用率を測定することで既存手法より提案手法による負荷分散が優れている点を確認できた．

5.2 今後の課題・展望

Publisher にかかる負荷を表す負荷指標に差が生じているため，一定した負荷指標の実現のために CPU 使用率のほかの負荷指標の設定，または調整が必要であると言える．さらに今回 Broker の負荷状況の監視に使用したシステムは自作のものだったため実用性，正確性を高めるためオープンソースの監視システム [12] などとの連携が必要と言える．最後に今回 Server redirection の実装をしたのは Publisher からの CONNACK パケットのみの実装であったため，Subscriber や DISCONNECT パケットのなどにも実装することでよりスケーラブルな負荷分散を目指す必要があると言える．

謝辞 本研究は JST さきがけ JPMJPR21P8 の支援を受けたものです．

文　　献

[1] 総務省, “令和 4 年版　情報通信白書,” https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r04/pdf/index.html (accessed May 19, 2023).
[2] “Http documentation,” https://httpwg.org/specs/ (accessed May 19, 2023).
[3] “Rfc 6455 - the websocket protocol,” https://datatracker.ietf.org/doc/html/rfc6455/ (accessed May 19, 2023).
[4] “Mqtt - the standard for iot messaging,” https://mqtt.org/ (accessed May 19, 2023).
[5] R. Banno and T. Yoshizawa, “A scalable iot data collection method by shared-subscription with distributed mqtt brokers,” EAI International Conference on Mobile Networks and Management, pp.218–226, 2021.
[6] “Adguard dns — ad-blocking dns server,” https://adguard-dns.io/en/welcome.html (accessed May 19, 2023).
[7] P. Jutadhamakorn, T. Pillavas, V. Visoottiviseth, R. Takano, D. Kobayashi, “A scalable and low-cost mqtt broker clustering system,” 2017 2nd International Conference on Information Technology(INCIT), pp.••–••, 2017.
[8] L.C. ofF5, “Nginx plus for the iot : Load blancing mqtt,” https://www.nginx.com/blog/nginx-plus-iot-load-balancing-mqtt/ (accessed May 19, 2023).
[9] OASIS, “Mqtt version 5.0,” https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html (accessed May 19, 2023).
[10] “Mqttloader,” https://github.com/dist-sys/mqttloader/ (accessed May 19, 2023).
[11] “Eclipse mosquitto,” https://mosquitto.org/ (accessed May 19, 2023).
[12] “Zabbix ;; the enterprise-class open sourse network monitoring solution,” https://www.zabbix.com/jp/ (accessed May 19, 2023).