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
IA2021-26(2021-10)

MQTT ブローカと共有サブスクリプションを用いたスケーラブルな IoT データ収集方式の検討

坂野　遼平†　　芳澤俊成 *†

† 工学院大学 情報学部 情報通信工学科
〒192-0015 東京都八王子市中野町 2665-1
E-mail: banno@computer.org

あらまし　スマートファクトリ等の IoT システムにおいては，センサ等から大量のデータを収集する必要がある．こうした用途に対し，Publish/Subscribe 型の通信プロトコルであり疎結合性に優れている MQTT が有望視されている．しかしながら，MQTT を用いる場合，ブローカがパフォーマンス上のボトルネックとなる．加えて，サブスクライバにおける受信処理についても高いスループットが求められる．本研究では，複数台の MQTT ブローカを用いたスケーラブルな IoT データ収集手法を提案する．提案手法では，複数ブローカによるパブリッシャからのデータ収集の並列化，および，共有サブスクリプション機能を用いたサブスクライバにおける受信処理の並列化を行い，アプリケーションが大量の IoT データを受信することを可能とする．負荷テストツール MQTTLoader を開発し，実機環境において提案手法の評価を行った．実験の結果，従来の方法と比べてスループットを向上できることを確認した．
キーワード　MQTT, IoT, Publish/Subscribe

A study of scalable IoT data collection method by shared-subscription with MQTT brokers

Ryohei BANNO† and Toshinori YOSHIZAWA†

† Department of Information and Communications Engineering, Kogakuin University
2665-1 Nakano-machi, Hachioji-shi, Tokyo, 192-0015 Japan
E-mail: banno@computer.org

Abstract　Internet of Things (IoT) systems like a smart factory require collecting a massive amount of data from sensors. MQTT is a promising protocol for such uses due to its loose-coupling nature by publish/subscribe messaging model. However, there is an issue that a broker could be a performance bottleneck. In addition, reception by a subscriber might not catch up with the amount of data from the broker. In this paper, we propose a scalable IoT data collection method with distributed MQTT brokers. By shared-subscription functionality, the proposed method enables an application to receive a massive amount of IoT data. We developed a load testing tool named MQTTLoader and evaluated the proposed method. Experimental results show that the proposed method can improve the throughput compared to conventional ways.

Key words　MQTT, IoT, Publish/Subscribe

1. はじめに

幅広い領域において IoT システムの展開が進んでいる．たとえば製造業においては，Industrie 4.0 の具現化として，産業用ロボットの状態を監視し，故障等を予測することでダウンタイムを削減するといった取り組みが進められている [1]．また，スマートシティにおいては，センサから環境データを収集し交通渋滞の解消や自然災害への予防的対策をおこなうといった取り組みが為されている．

∗ Present affiliation: VINX Corp.
This is an unrefereed paper.

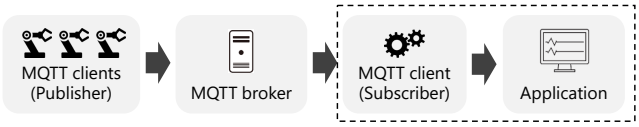

図 1　MQTT による IoT データ収集

IoT デバイスからデータを収集するための通信プロトコルとして，MQTT [2] が有望視されている．MQTT は，Publish/Subscribe モデル [3] による疎結合性を備えていることに加え，ヘッダサイズが小さく，トラフィック削減や省電力化の観点からも注目を集めている．

しかしながら，MQTT を用いた IoT データ収集においては，ブローカやサブスクライバがボトルネックとなる可能性がある．図 1 は，MQTT による IoT データ収集の典型的な構成を示している．パブリッシャから膨大なデータが送られる場合，ブローカが過負荷となり遅延の増大やデータのロスが生じる可能性がある．また，ブローカの過負荷が何らかの手段によって解消されたとしても，サブスクライバにおける受信処理が過負荷となり，やはりデータのロス等が生じる可能性がある．

これに対し，本研究では，複数台の MQTT ブローカを用いたスケーラブルな IoT データ収集手法を提案する．提案手法では，複数ブローカによるパブリッシャからのデータ収集の並列化を行うことに加え，MQTT version 5.0 [4] が備える共有サブスクリプション機能を用いてサブスクライバにおける受信処理の並列化を行い，アプリケーションが大量の IoT データを受信することを可能とする．

2. 関 連 研 究

MQTT においてはブローカがパフォーマンス上のボトルネックとなり得るため，その性能向上を図る研究が様々に為されている．

muMQ [5] はブローカ単体の性能向上を図ったものである．マルチコアの活用，および，DPDK を用いたカーネルオーバヘッドの回避により，性能向上を実現している．ただし，複数ブローカによるスケールアウトには対応しておらず，また DPDK に対応したハードウェアを要するという制約がある．

MQTT-ST [6] や ILDM [7], [8] では，複数の MQTT ブローカを相互に接続し配送木を形成する手法が提案されている．これらの手法では，パブリッシャおよびサブスクライバの配置に局所性がある場合，すなわち同一トピックのパブリッシャおよびサブスクライバが同じブローカに接続している場合に，優れた効率性を発揮する．一方で，図 1 に示したような全パブリッシャからのデータ収集を想定した場合，負荷分散の効果を得ることは難しく，ブローカ間のマルチホップにより遅延が増大するという問題がある．

Detti ら [9] は，各サブスクライバが複数ブローカに接続することで，ブローカ間の通信量を削減する手法を提案している．この手法は，アプリケーションがブローカ群に対し複数の接続を持つという点で，本研究における提案手法と類似している．しかしながら，同一ブローカからのメッセージ受信を複数のサブスクライバで分担することについては考慮されていない．

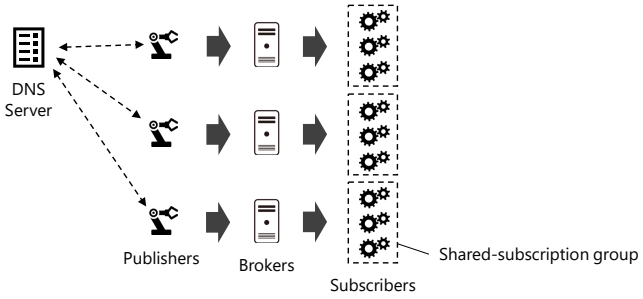

図 2　提案手法の概要

3. 複数 MQTT ブローカを用いたスケーラブルな IoT データ収集手法

本研究では，少数のアプリケーションが多数のパブリッシャからデータを収集して活用する状況を想定し，複数台の MQTT ブローカを用いたスケーラブルな IoT データ収集手法を提案する．図 2 に提案手法の概要を示す．提案手法では，複数台のブローカを用いて負荷分散を図ることに加え，各アプリケーションが複数のサブスクライバを用いて IoT データを並列に受信する．

各パブリッシャは，ブローカ群のうちいずれかひとつに接続する．接続先ブローカは，DNS ラウンドロビンによって定める．提案手法においてパブリッシャに求められる要件は，ブローカの指定にドメイン名を用いること，および適切な DNS サーバを参照することのみであり，それ以外の点については通常の MQTT クライアントと同様に振る舞う．

アプリケーションは，ブローカ数以上のサブスクライバを内部に保持し，各サブスクライバはブローカ群のうちいずれかひとつに接続する．本稿では，各サブスクライバの接続先ブローカについては，アプリケーションの設定としてユーザが事前指定することを想定している．その際，アプリケーションごとに，すべてのブローカにひとつ以上のサブスクライバが接続するよう指定されるものとする．

あるアプリケーションが有する複数のサブスクライバが，同一のブローカに接続する場合，それらサブスクライバは共有サブスクリプショングループを形成する．共有サブスクリプションは，MQTT version 5.0 [4] において規定された機能であり，あるトピックのメッセージを共有サブスクリプショングループ内で分担受信することを可能とする．すなわち，当該グループが購読しているトピックのメッセージは，グループ内のいずれかひとつのサブスクライバへと配送される．これにより，サブスクライバによる受信の負荷分散が可能となる．

なお, 共有サブスクリプションと類似した機構として, Apache Kafka [10], [11] におけるコンシューマグループが挙げられる．Kafka と MQTT にはそれぞれ特色があるが，本研究では，OASIS [4] および ISO [12] によって標準化されており IoT で広く用いられている MQTT を対象としている．

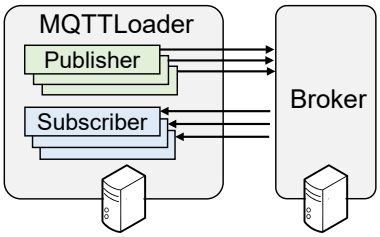

図 3　単一ホストによる MQTTLoader の利用

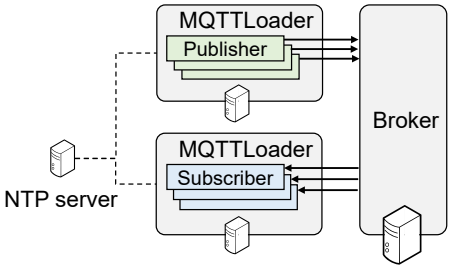

図 4　複数ホストによる MQTTLoader の利用

```
Measurement started: 2020-11-04 19:18:38.169 JST
Measurement ended: 2020-11-04 19:18:49.925 JST

-----Publisher-----
Maximum throughput[msg/s]: 76122
Average throughput[msg/s]: 57142.86
Number of published messages: 400000
Per second throughput[msg/s]: 41128, 64410, 76122, 71158, 75101, 71518, 563

-----Subscriber-----
Maximum throughput[msg/s]: 83989
Average throughput[msg/s]: 57142.86
Number of received messages: 400000
Per second throughput[msg/s]: 11244, 55569, 76514, 70669, 74768, 83989, 27247
Maximum latency[ms]: 895
Average latency[ms]: 512.50
```

図 5　MQTTLoader の出力例

4.　負荷テストツール MQTTLoader

2 章にて述べたように，MQTT ブローカはパフォーマンス上のボトルネックとなり得るため，その性能特性を定量的に知ることは重要である．MQTT v5.0 に準拠したブローカの性能特性を明らかにし，また提案手法の定量的な評価を行うために，MQTT ブローカの負荷テストツールとして MQTTLoader を開発した [13],[14]．MQTTLoader は Java 言語で実装されており，バイナリ，ソースコード，およびドキュメントは GitHub で公開されている [13].

図 3 は MQTTLoader の概要を示している．MQTTLoader を用いることで，複数の MQTT クライアントを同時に立ち上げ，ブローカに対して様々な構成にて負荷をかけることができる．代表的なパラメータを表 1 に示す．表 1 に記載したパラメータ以外にも，QoS レベル，Retain フラグ，ペイロードサイズ，パブリッシャの送信間隔，ランプアップ/ランプダウン時間等，様々な設定が可能である．

MQTTLoader は単一ホスト上での動作に加え，図 4 に示すように複数ホスト上で動作させることも可能である．パブリッシャとサブスクライバを同一ホスト上で動作させた場合，サブスクライバによる受信負荷がパブリッシャのスループットを抑制するといった相互作用が生じる可能性がある．異なるホスト上でパブリッシャとサブスクライバをそれぞれ動作させることにより，そうした相互作用を回避することができる．

表 1　代表的なパラメータ

パラメータ	説明
broker	ブローカの IP アドレスまたはドメイン
mqtt_version	MQTT プロトコルのバージョン
num_publishers	パブリッシャ数
num_subscribers	サブスクライバ数
shared_subscription	共有サブスクリプションの ON/OFF
ntp	NTP サーバの IP アドレスまたはドメイン

MQTTLoader は，パラメータ指定された一定時間あるいは一定量の負荷をかけ終えた後，平均スループット，平均レイテンシ等を算出する．図 5 に例を示す．ここでレイテンシは，パブリッシャによるメッセージ送出からサブスクライバによる受信までの所要時間である．パブリッシャが送出する各メッセージのペイロードにはタイムスタンプが埋め込まれており，サブスクライバにおける受信時刻との差分からレイテンシが計算される．MQTTLoader を複数ホストで動作させる場合，パラメータ設定にて同一の NTP サーバを指定しておくことで，ホスト間の時刻のずれを補正してレイテンシの計算が行われる．

5.　評　　価

提案手法を評価するため，MQTTLoader を用いて以下の各指標を測定した．

スループット (パブリッシャ側)
パブリッシャとブローカ間の平均スループット
スループット (サブスクライバ側)
ブローカとサブスクライバ間の平均スループット
レイテンシ
パブリッシャからサブスクライバまでの平均遅延

パブリッシャおよびサブスクライバには，MQTTLoader v0.7.2 を用いた．MQTT ブローカについては様々な製品が存在するが [15]〜[17]，本実験では，特に広く用いられており優れた性能で知られる Mosquitto (v1.6.9) を用いた．

パブリッシャ数は 8 とし，各パブリッシャは 600 バイトのペイロードを持つメッセージを送信間隔ゼロにて送出する設定とした．測定は 60 秒間行い，各測定を 3 回繰り返して平均を算出した．

ブローカおよびサブスクライバについては，図 6 に示す 4 パターンを測定し比較した．

SGL-SGL
ブローカ数 1 (B_1)，サブスクライバ数 1 (S_1)
SGL-SHD
ブローカ数 1 (B_1)，サブスクライバ数 2 (S_1，S_2)．共有サブスクリプション有り．
MLT-SGL
ブローカ数 2 (B_1，B_2)，サブスクライバ数 2 (S_1，S_2)．共有サブスクリプション無し．

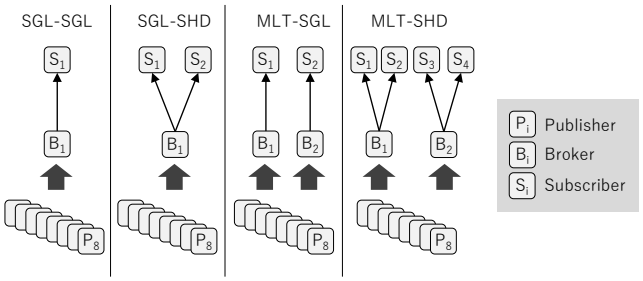

図 6　実 験 構 成

表 2　マシンスペック

	パブリッシャ	ブローカおよびサブスクライバ
CPU	Core i9 10900K	Celeron N3350
Memory	64GB	4GB
OS	Ubuntu 20.04	Ubuntu 20.04

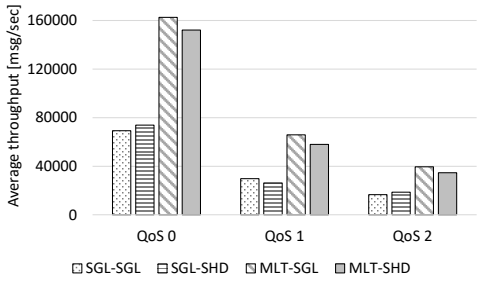

図 7　スループット (パブリッシャ側)

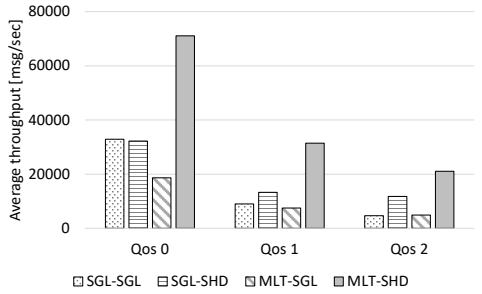

図 8　スループット (サブスクライバ側)

MLT-SHD
ブローカ数 2 (B_1，B_2)，サブスクライバ数 4 (S_1，S_2，S_3，S_4)．共有サブスクリプション有り．S_1 と S_2，S_3 と S_4 がそれぞれ共有サブスクリプショングループを形成．

表 2 は実験に用いたホストマシンのスペックを示している．なお，S_1 と S_2，および S_3 と S_4 は，同一ホストマシン上で動作させている．

5.1　スループット

図 7 および図 8 は，それぞれパブリッシャおよびサブスクライバのスループットを示している．全体として，パブリッシャ側のスループットがサブスクライバ側のスループットよりも大きくなっている．このため，ブローカにおいては過負荷が生じている状態と考えられる．

サブスクライバ側のスループットに着目すると，提案手法で

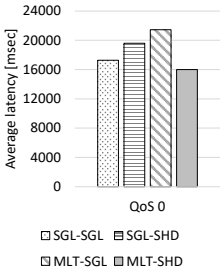

図 9　レイテンシ (QoS level 0)

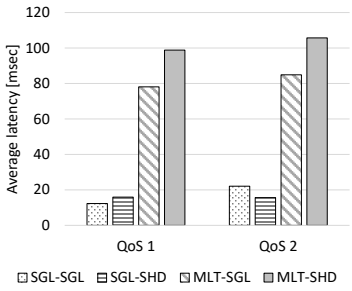

図 10　レイテンシ (QoS level 1 or 2)

ある MLT-SHD が最も高いスループットを得ている．この結果より，複数ブローカおよび複数サブスクライバを用いたデータ収集がスループット向上に有効であることがわかる．一方，図 7 では，MLT-SGL が MLT-SHD がよりも高いスループットとなっている．原因のひとつとして，各ブローカ内における共有サブスクリプションの処理がブローカにおける受信処理に影響を及ぼしていることが考えられる．

5.2　レイテンシ

図 9 および図 10 はレイテンシの測定結果を示している．4 パターンのいずれについても，QoS 0 では極めて大きなレイテンシが生じている．QoS 1, 2 と比べ，パブリッシャからブローカに対してバースト的な負荷がかかっていることが要因と考えられる．QoS 1, 2 においては，MLT-SHD が最もレイテンシが大きくなっている．なお，本実験においては，前述のとおりブローカが過負荷となっており，レイテンシについては時間経過とともに増大していく状態であったと考えられ，定常的な値では無いことに留意する必要がある．

6.　お わ り に

本研究では，複数台の MQTT ブローカを用いたスケーラブルな IoT データ収集手法を提案した．提案手法では，複数ブローカによるパブリッシャからのデータ収集の並列化に加え，共有サブスクリプション機能を用いてサブスクライバにおける受信処理の並列化を行う．負荷テストツール MQTTLoader を開発し，実機環境において提案手法の評価を行った．ブローカとして Mosquitto を用いた実験の結果，従来の方法と比べてスループットを向上できることを確認した．一方で，レイテンシについては増大する傾向があることがわかった．今後の課題としては，実験結果の詳細な分析や，レイテンシを改善する仕組みの検討等が考えられる．

謝辞　本研究は JSPS 科研費 19K20253 の助成を受けたものです.

文　　献

[1] FANUC Corp., “Field system,” https://fanuc.co.jp/en/product/field/index.html (accessed Sep. 17, 2021).
[2] MQTT, https://mqtt.org/ (accessed Sep. 17, 2021).
[3] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The Many Faces of Publish/Subscribe,” ACM Computing Surveys, vol.35, no.2, pp.114–131, 2003.
[4] OASIS Standard, “MQTT Version 5.0,” 2019.
[5] W. Pipatsakulroj, V. Visoottiviseth, and R. Takano, “muMQ: A lightweight and scalable MQTT broker,” Proc. IEEE International Symposium on Local and Metropolitan Area Networks, pp.1–6, 2017.
[6] E. Longo, A.E. Redondi, M. Cesana, AndrésArcia-Moret, P. Manzoni, “MQTT-ST: a Spanning Tree Protocol for Distributed MQTT Brokers,” Proc. IEEE International Conference on Communications, pp.1–6, 2020.
[7] R. Banno, J. Sun, M. Fujita, S. Takeuchi, and K. Shudo, “Dissemination of edge-heavy data on heterogeneous MQTT brokers,” Proc. IEEE International Conference on Cloud Networking, pp.1–7, 2017.
[8] R. Banno, J. Sun, S. Takeuchi, and K. Shudo, “Interworking Layer of Distributed MQTT Brokers,” IEICE Transactions on Information and Systems, vol.E102.D, no.12, pp.2281–2294, 2019.
[9] A. Detti, L. Funari, and N. Blefari-Melazzi, “Sub-Linear Scalability of MQTT Clusters in Topic-Based Publish-Subscribe Applications,” IEEE Transactions on Network and Service Management, vol.17, no.3, pp.1954–1968, 2020.
[10] J. Kreps, N. Narkhede, and J. Rao, “Kafka: A distributed messaging system for log processing,” Proc. International Workshop on Networking Meets Databases, pp.1–7, 2011.
[11] Apache Kafka, https://kafka.apache.org/ (accessed Sep. 17, 2021).
[12] ISO Central Secretary, “Information technology — Message Queuing Telemetry Transport (MQTT) v3.1.1,” Standard ISO/IEC 20922:2016, International Organization for Standardization, 2016.
[13] MQTTLoader, https://github.com/dist-sys/mqttloader (accessed Sep. 17, 2021).
[14] R. Banno, K. Ohsawa, Y. Kitagawa, T. Takada, and T. Yoshizawa, “Measuring Performance of MQTT v5.0 Brokers with MQTTLoader,” Proc. IEEE Consumer Communications & Networking Conference, pp.1–2, 2021.
[15] R.A. Light, “Mosquitto: server and client implementation of the MQTT protocol,” Journal of Open Source Software, vol.2, no.13, p.265, 2017.
[16] HiveMQ, https://www.hivemq.com/ (accessed Sep. 17, 2021).
[17] EMQ X, https://www.emqx.io/ (accessed Sep. 17, 2021).