コネクテッドカーのための MQTT-Bridge を用いた 二重エッジアーキテクチャ設計方法の提案と評価

M2017SE012　宇野　聡将
指導教員　青山　幹雄

1 研究背景

コネクテッドカーにおいて大量データを処理するために，エッジコンピューティング（エッジ）が注目されている[1]．しかし従来のエッジアーキテクチャでは，コネクテッドカーの増加による負荷の増大に対応できず，エッジへの多様な品質要求を満たせなくなる．IoT での利用を想定した MQTT プロトコルは Pub/Sub アーキテクチャに基づいており，標準化されている．しかし，ブローカにメッセージが集中するためボトルネックとなり，スケーラビリティが課題となる．

本研究ではエッジコンピューティングに Pub/Sub を適用し，2 層のエッジにデータ配信処理を分散する二重エッジアーキテクチャを提案する．エッジ間でブリッジによるメッセージ共有を行い，エッジのスケールアウトを可能とする設計方法を提案する．提案アーキテクチャのプロトタイプを実装し，スケーラビリティを検証することで設計方法の有用性を示す．

2 研究課題

本研究の研究課題は以下の 3 点である．

(1) Pub/Sub を適用した二層のエッジにメッセージ配信処理を分散する二重エッジアーキテクチャの提案
(2) エッジ上で稼働するブローカ間でブリッジを用いたメッセージ共有でブローカをスケールアウトする設計方法の提案
(3) (1), (2)を車載システムに適用し，スケーラビリティを保証するエッジアーキテクチャの提案

3 関連研究

3.1 エッジコンピューティング

ネットワークのエッジにクラウドの機能の一部を配置して分散処理するアーキテクチャである[9]．エッジ上でメッセージを分散処理することで通信頻度や通信量が膨大なシステムのリアルタイム性の向上が可能となる．

3.2 コネクテッドカー

AECC(Automotive Edge Computing Consortium)[1]は従来のモバイルネットワークアーキテクチャにエッジを適用し，コネクテッドカーのデータ活用を推進している．

3.3 Publish/Subscribe アーキテクチャ

メッセージの非同期通信アーキテクチャであり，国際標準として MQTT がある[5, 6]．ブローカ間でメッセージを共有する方法が研究されており[2, 7, 8]，その方法の一つとしてブリッジ機能がある．

3.4 IoT システムのためのエッジアーキテクチャ設計方法論の提案と評価

アーキテクチャデザインパターンを用いて多様な非機能要求を満たすエッジアーキテクチャの設計方法が提案されている[10]．エッジ間スケールアウトに多段 Pub/Sub を用いる方法を提案している．

4 アプローチ

4.1 前提条件

本研究の前提条件として以下を設定する．

(1) Pub/Sub にエッジアーキテクチャを適用

Pub/Sub アーキテクチャのコンポーネントをエッジアーキテクチャの各レイヤに配置する．エッジ層にブローカを配置することでクラウドへの配信メッセージを削減でき，必要なネットワーク帯域を軽減可能である．

(2) 対象プラットフォームを MQTT

本研究では IoT 向けに実装された，軽量な通信プロトコルである MQTT の利用を前提とする．

(3) メッセージの定義

MQTT 通信にメッセージを用いる．メッセージはセンサデータ等を格納するデータと通信に必要な情報を格納するヘッダで構成される．ブローカによるメッセージの配信に必要なトピックはヘッダに格納される．

4.2 アプローチ

アプローチを図 1 に示す．エッジに Pub/Sub を適用し，ブローカをエッジに配置する．ブローカは 1 つのメッセージ受信に対し，複数のサブスクライバに同一のメッセージを配信する場合がある．そのためクライアントの増加によりメッセージ配信処理のスケーラビリティが問題となる．そこで，以下を満たすアーキテクチャ設計方法を提案する．

(1) エッジを分割して二層構造でメッセージ配信処理を分散しスケーラビリティを確保するアーキテクチャ
(2) Pub/Sub のブリッジを用いてエッジ間でデータを共有可能とするアーキテクチャ

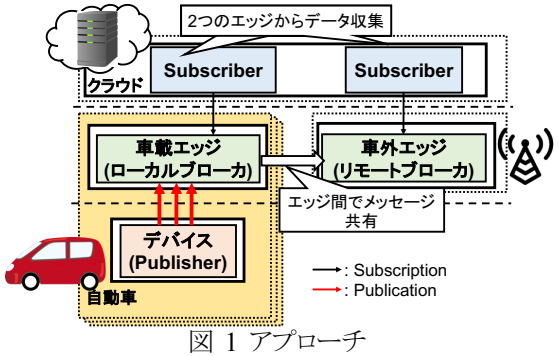

図 1 アプローチ

5 アーキテクチャ設計方法

5.1 提案プロセス

アプローチに従い，コネクテッドカーに適用する二重エッジアーキテクチャを設計する．提案プロセスを以下に示す．

(1) 論理アーキテクチャ設計

ブリッジを用いてエッジ間でメッセージを分散するアーキテクチャを設計する．

(2) ブリッジを用いたメッセージング設計

ブリッジのメッセージ分散に用いるトピックモデルの設計とブリッジの振る舞いを定義する．

(3) アーキテクチャの具体化

論理アーキテクチャに基づいたエッジ間でのメッセージ分散の実現に必要なコンポーネントを示す物理アーキテクチャを設計する. また, デバイスからクラウドへのメッセージ配信のモデルを設計する.

5.2 二重エッジアーキテクチャ設計

本研究で提案する二重エッジアーキテクチャを図 2 に示す. 車載エッジはメッセージを集中的に収集するが, 単一のエッジであるため負荷の増大が問題となる. メッセージ集中によって車載エッジの負荷が増大した場合, 二層構造を実現する車外エッジへメッセージを分散して処理する. 各エッジは独立して稼働するため, メッセージ共有には複数回の Pub/Sub 通信を行う必要がある. このとき車載エッジのプロセスの増加が問題となる. そこで, 車載エッジが車外エッジのクライアントとして振る舞い, メッセージ共有を可能とするブリッジを用いることでメッセージ分散プロセスの負荷を軽減する.

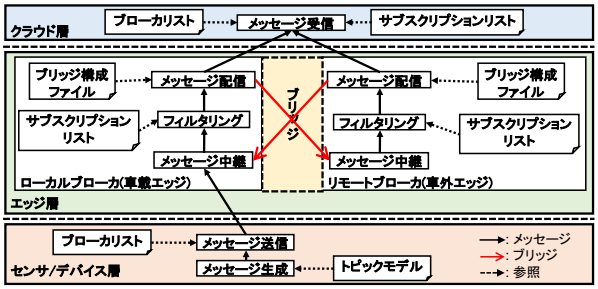

図 2 論理アーキテクチャ

5.3 ブリッジを用いたメッセージング方法

5.3.1. トピック設計

ブリッジによるメッセージ共有の振る舞いはブリッジ構成ファイルで決定する. トピック構造の例を図 3 に示す. トピックはメッセージに含まれたデータに基づいて決定するリソース部とメッセージを一意に識別可能とする ID 部で構成される. またトピックは階層構造をとることから, リソース部を上位層から下位層に詳細化するよう決定する. ブリッジでメッセージ配信を分散する際に, エッジ毎に異なる上位のトピックをワイルドカード# を用いることで下位のトピックを抽象化して指定可能である. これにより, 複雑なトピック指定を必要としない効率的なメッセージが分散可能となる.

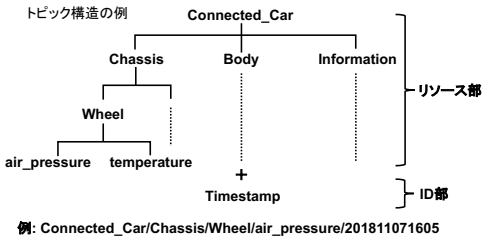

図 3 トピック設計

5.3.2. ブリッジによるメッセージ分散プロセス

本研究でのブリッジについて以下に定義する. また, ブリッジを有効化したブローカの挙動を図 4 に示す.

(1) 2 つのブローカ間で, 一方が受信したメッセージを他方に送信して共有可能とする通信である.

(2) ブリッジ構成ファイルでトピックを指定することで, 共有するメッセージを限定可能とする.

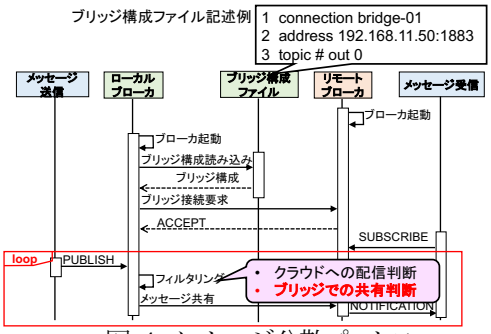

図 4 メッセージ分散プロセス

5.4 物理アーキテクチャ

5.2 節で示した論理アーキテクチャに基づき物理アーキテクチャを設計した(図 5). 提案アーキテクチャの各層間の Pub/Sub 通信に MQTT を適用する. デバイスおよび車載エッジは自動車を単位とした単一のノードに配置され, 路上などに配置される車外エッジと MQTT ブリッジでメッセージ共有を行うことでメッセージ配信処理を分散する.

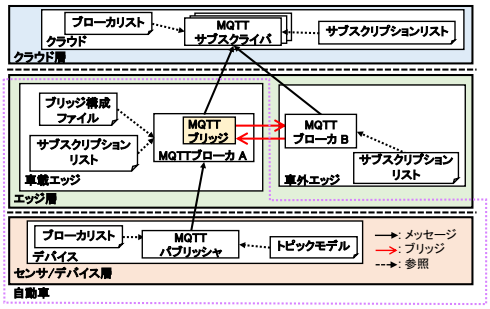

図 5 物理アーキテクチャ

5.5 メッセージ配信モデル

物理アーキテクチャに基づいたメッセージ配信モデルを図 6 に示す. 車載エッジの MQTT ブローカは MQTT ブリッジを起動し, 車外エッジの MQTT ブローカに接続, クライアントとなる. クラウド層の MQTT サブスクライバは車載エッジおよび車外エッジに配置された MQTT ブローカにサブスクライブする.

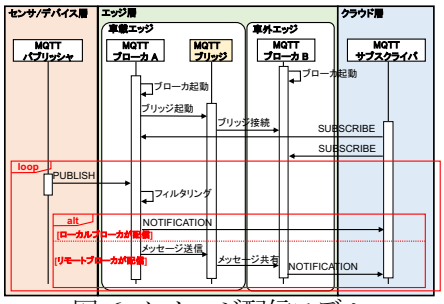

図 6 メッセージ配信モデル

6 プロトタイプの実装

6.1 プロトタイプ実装の目的

提案アーキテクチャのスケーラビリティを検証するためにプロトタイプを作成する. 目的を以下に示す.

(1) ブリッジ処理のブリッジプロセスの増加によってローカルブローカにかかる負荷がパフォーマンスに影響を及ぼすか検証し, ブリッジを用いることの妥当性を確認する.

(2) 多段 Pub/Sub との比較

ブローカ間でメッセージを共有する方法として, 多段 Pub/Sub がある. ブリッジを用いた場合との CPU 使用率を比較し, 本提案アーキテクチャの優位性を示す.

6.2 プロトタイプ実装

プロトタイプの実装にあたり，Publish/Subscribe アーキテクチャの実装である MQTT を用いる．また，ブローカの実装として Mosquitto[3]，MQTT クライアントの実装として Paho-mqtt[4]を用いる．本プロトタイプは2つのブローカをブリッジ接続することでメッセージを分散してクラウドに配信する二重エッジパターンとブローカ単体でメッセージ配信を行う単一エッジパターン，多段 Pub/Sub でメッセージ共有を行う多段 Pub/Sub パターンを実行することで提案アーキテクチャの効果を検証する．

6.3 実行シナリオ

二重エッジパターンを図7に示す．デバイスアプリケーションで生成したメッセージをエッジ層に配置したブローカ A が受信する．ブローカ A はブリッジ構成ファイルに基づきブローカ B にメッセージ共有するか判断する．共有する場合はブリッジを経由してブローカ B にメッセージを送信する．クラウドアプリケーションは異なるトピックをサブスクライブするため2つ稼働させるが，1つのクラウドサーバ上で稼働しているものとする．単一ブローカパターンはメッセージ共有を行わず，図 7 のブリッジ構成ファイル，ブリッジ，ブローカ B，クラウドアプリケーション B が存在しない状態でクラウドへメッセージを送信する．

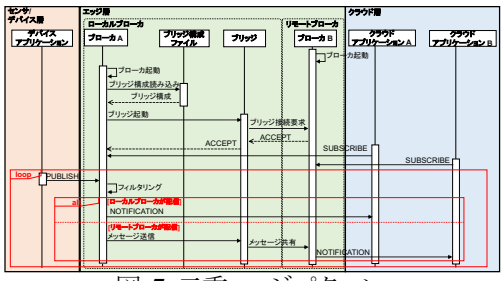

図 7 二重エッジパターン

多段 Pub/Sub パターンを図 8 に示す．ローカルブローカでサブスクライバとパブリッシャを稼働する．ブローカ A が受信したメッセージをサブスクライバが配信を受け，トピックとペイロードを変更せずパブリッシャに渡してリモートブローカにパブリッシュする．サブスクリプションで指定するトピックはブリッジ構成ファイルで指定するトピックと同一である．

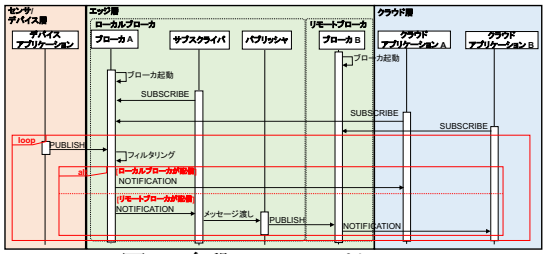

図 8 多段 Pub/Sub パターン

6.4 実行環境

本プロトタイプの実行環境および実装に用いたソフトウェア情報を表 1，表 2 に示す．

表 1 実行環境

システム名	デバイス	ローカルブローカ	リモートブローカ	クラウド
ハードウェア	Raspberry Pi 3 B+	Raspberry Pi B+	Raspberry Pi 3 B+	MacBook Pro
OS	Debian 9.6	Debian 9.6	Debian 9.6	OS X El Capitain
プロセッサ	Cortex-A53	RM1176JZF-S	Cortex-A53	Intel Core i5

表 2 ソフトウェア情報

ソフトウェア	実装範囲	バージョン
Mosquitto	ブローカ	1.4.10
Paho-MQTT client	クライアント	1.4.0

6.5 プロトタイプ構成

6.5.1. プロトタイプの構成

二重エッジパターンのプロトタイプの構成を図 9 に示す．

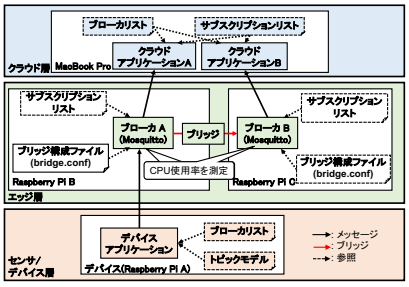

図 9 二重エッジパターンプロトタイプ構成

6.5.2. 構成要素の説明

(1) デバイスアプリケーション

10 ミリ秒間隔でメッセージを生成し，ブローカ A に送信する．ペイロードはタイムスタンプと乱数で生成した実数値である．

(2) ローカルブローカ(ブローカ A)

受信したメッセージをクラウドアプリケーションに配信する．また，ブローカ起動時にブリッジ構成ファイルを読み込むことでブリッジを起動する．ブリッジ接続中は，ブリッジ構成ファイルに従ってリモートブローカにメッセージを共有する．

(3) リモートブローカ(ブローカ B)

ローカルブローカからメッセージ配信処理を分散するためブリッジ経由でメッセージを受信する．ローカルブローカと同様にクラウドアプリケーションにメッセージを配信する．

(4) クラウドアプリケーション

ブローカにあらかじめ決定したトピックでサブスクライブする．本プロトタイプではローカルブローカとリモートブローカから受信するメッセージに重複が起こらないトピックを設定する．

7 例題への適用

7.1 適用事例

コネクテッドカーのエッジ連携に提案アーキテクチャを適用する．二重エッジパターンとしてローカルブローカはデバイスが生成したメッセージをあらかじめ定めた割合であるメッセージ分散率に基づいてリモートブローカに分散する．ッセージ分散率 0%のとき，ローカルブローカのみでメッセージを配信し，リモートブローカにメッセージを共有しないため単一エッジパターンと同等する．ローカルブローカの CPU 使用率を測定し，ブリッジによって得られるスケーラビリティを検証する．

8 評価

8.1 評価方法

6.2 節で示した 3 つの実行パターンを 3 回ずつ実行した．デバイスから 10 ミリ秒ごとにメッセージを送信し，ローカルブローカが最初のメッセージを受信してから 10 秒経過した後，CPU 使用率を 1 秒間隔で 150 秒間測定した．二重エッジパターンと単一エッジパターンおよび多段 Pub/Sub パターンの CPU 使用率の平均，推移を比較する．

8.2 評価

8.2.1. 単一エッジパターンとの比較

単一エッジパターンと二重エッジパターンを比較する(図 10)．単一エッジパターンの CPU 使用率の平均は約 11%だった．二重エッジパターンでメッセージを分散しながらメッセージ配信

した場合，CPU 使用率の平均はメッセージ分散率が 50%のときに単一エッジパターンとほぼ同値であった．ブリッジプロセスが最も増加するメッセージ分散率 75%の場合でも CPU 使用率は約 12%であり，最大で 1.56%の増加であることからローカルブローカに負荷をかけずにメッセージ分散が可能である．

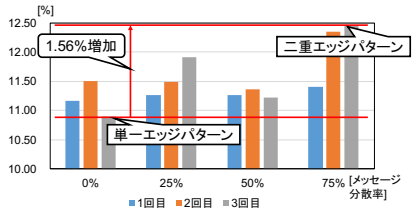

図 10 CPU 使用率の平均値 二重エッジパターン

単一エッジパターンと多段 Pub/Sub パターンを比較する(図 11)．多段 Pub/Sub の場合，メッセージ分散率の増加に伴い CPU 使用率が大幅に上昇した．Pub/Sub 処理が CPU 使用率に大きく影響していることを確認した．

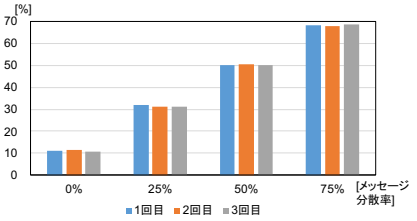

図 11 CPU 使用率の平均値 多段 Pub/Sub パターン

8.2.2. 多段 Pub/Sub との比較

二重エッジパターンと多段 Pub/Sub パターンを比較する(図 12)．メッセージ分散率が 25%と 75%のときに着目する．多段 Pub/Sub パターンでは CPU 使用率が 31%から 68%にまで大幅に増加したことに対し，二重エッジパターンでは CPU 使用率は 11%から 12%までと 1%の増加であった．同じメッセージ分散率での CPU 使用率は最大でメッセージ分散率が 75%のとき約 82%負荷を削減することができた．よって二重エッジパターンによってエッジに負荷をかけることなくメッセージ分散を行えることを確認した．

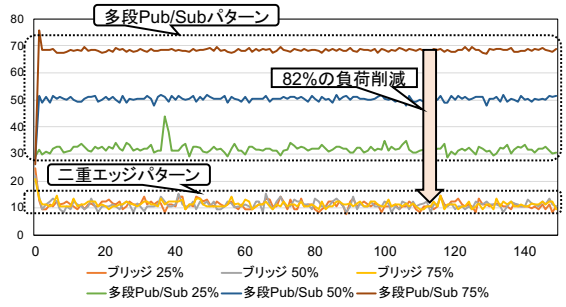

図 12 CPU 使用率の推移

9 考察

9.1 先行研究との比較

3.4 節で述べた先行研究である IoT システムのためのエッジアーキテクチャ設計方法論の提案と評価[11]ではエッジ間でのメッセージ分散方法として複数回の Pub/Sub 通信を行う多段 Pub/Sub パターンをとっている．本研究では多段 Pub/Sub でメッセージを共有するパターンと提案設計方法であるブリッジを用いてメッセージを共有するパターンのプロトタイプを実装した．2つのパターンで同じ条件でのメッセージ送受信を行い，CPU 使用率を測定した結果，ブリッジを用いる方法がスケーラビリティについて優位である．

9.2 例題適用

AECC[1]のモバイルネットワークアーキテクチャにエッジコンピューティングを適用したアーキテクチャを提案している．しかしエッジは単一の構成であり，エッジのスケーラビリティが課題となる．本研究では車載と車外の二層のエッジでメッセージ配信処理を分散するアーキテクチャを提案した．プロトタイプの評価から，本提案アーキテクチャは AECC のアーキテクチャと比較してスケーラビリティについて優位である．またエッジ間スケールアウトで，コネクテッドカーの増加に伴う収集データ量の増大に対処可能だと考えられる．

10 今後の課題

今後の課題として以下の 2 点を挙げる．

(1) 移動体の特性を考慮したブローカ接続方法

自動車は移動体であるため，接続するエッジを変更する必要がある．その場合でもサービス提供を継続可能とするため車外エッジへの接続モデルを設計する必要がある．

(2) 遅延時間での評価

自動運転など，コネクテッドカーへのサービス提供にはシステムにリアルタイム性を求めるものがある．ブリッジと多段 Pub/Sub でのエッジ間メッセージ共有の遅延時間を測定することで，提案した設計方法の妥当性を評価する必要がある．

11 まとめ

二層のエッジにメッセージ配信処理を分散して負荷を軽減する二重エッジアーキテクチャを提案した．3 パターンのプロトタイプを実装し，メッセージ送受信時のローカルブローカの CPU 使用率を測定した．先行研究との比較から，本研究で提案した二重エッジアーキテクチャによりエッジのスケーラビリティを確保可能であり，設計方法の有用性を示した．

参考文献

[1] AECC, General Principle and Vision, V. 2.0.0, Apr. 2018, https://aecc.org/wp-content/uploads/2018/02/AECC_White_Paper.pdf.

[2] M. Collina, et al., Introducing the QEST Broker: Scaling the IoT by Bridging MQTT and REST, Proc. of PIMRC '12, IEEE, Sep. 2012, pp. 36-41.

[3] Eclipse Mosquitto, https://mosquitto.org.

[4] Eclipse paho, https://eclipse.org/paho/

[5] ISO/IEC 20922:2016, Information Technology – Message Queuing Telemetry Transport(MQTT) V.3.1.1, 2016.

[6] OASIS, MQTT Version 3.1.1 Plus Errata 01, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[7] 坂野 遼平 他, スケーラブルな MQTT ブローカの実現に向けた分散連携プロキシの提案, DICOMO2016 論文集, 情報処理学会, Jul. 2016, pp. 541-547.

[8] A. Schmitt, et al., Dynamic Bridge Generation for IoT Data Exchange via the MQTT Protocol, Procedia Computer Science, Vol. 130, Apr. 2018, pp. 90-97.

[9] W. Shi, et al., The Promise of Edge Computing, IEEE Computer, Vol. 49, No. 5, May 2016, pp. 78-81.

[10]濱野 真伍 他, IoT システムのためのエッジアーキテクチャ設計方法論の提案と評価, 第 198 回ソフトウェア工学研究会, Vol. 2018-SE-198, No. 12, 情報処理学会, Mar. 2018, pp. 1-8.

[11]宇野 聡将 他, コンテンツベースフィルタリングを可能とするマルチブローカエッジアーキテクチャの提案と評価, ウィンターワークショップ 2018 論文集, 情報処理学会, Jan. 2018, pp. 60-61.