エッジコンピューティングによる
緊急通報システムアーキテクチャ設計方法の提案と評価

2016SE002 安藤 有輝　2016SE051 柵木 哉信　2016SE059 長坂 卓朗
指導教員　青山 幹雄

1 はじめに

コネクテッドカーの普及とともに，自動車とクラウドが連携する新しいサービスの普及が進んでいる．そのサービスの一つに緊急通報システムがある．緊急通報システムは，事故が起きたときに緊急通報を迅速に行えるだけでなく，あおり運転の防止策にもなるとされている．しかし，実用化されているシステムは，通報するまでのプロセスで人手を介しているため，通報までに時間がかかる．警察や消防では，通報から現場到着までの時間短縮を課題としており，緊急通報システムにおいても時間短縮が必要である．

一方，今後はコネクテッドカーの普及により，クラウドと通信するデータ量が増加しレイテンシが増大する課題がある．この課題を解決する手段として，エッジコンピューティングが注目されている[1]．

本研究では，エッジコンピューティングを用いた緊急通報システムアーキテクチャ設計方法を提案する．プロトタイプを実装し，レイテンシとスケーラビリティを評価することで，提案方法の有用性を評価する．

2 研究課題

本研究では，以下の 2 点を研究課題とする．

(1) エッジコンピューティングを用いた緊急通報システムアーキテクチャ設計方法の提案
(2) 提案アーキテクチャ設計方法の有用性を評価

3 関連研究

3.1 緊急通報システム

緊急通報システムは，車両事故などの緊急時に位置情報などを自動で警察や消防に通知するシステムである．

欧州では，緊急通報システム eCall の新車着用が義務付けられている[3].

3.2 Publish/Subscribe アーキテクチャ

Publish/Subscribe(Pub/Sub)アーキテクチャは，ブローカを経由することで非同期メッセージ交換をするアーキテクチャである[9]. このアーキテクチャに基づく国際標準の一つに MQTT がある[7]. ブローカ間のメッセージ共有方法として，MQTT ブリッジがある[2].

3.3 エッジコンピューティング

エッジコンピューティングは，デバイスとクラウドの中間に，デバイスに物理的に近い位置に計算処理や通信機能を持った機器(エッジ)を配置するアーキテクチャである[11]. クラウドに送信するデータをフィルタリングする，デバイスへのデータ送信時間を削減するといった効果がある．

3.4 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

エッジを 2 層にし，MQTT のブリッジ機能を用いてメッセージ配信を分散させ，スケーラビリティを確保するエッジコンピューティングアーキテクチャが提案されている[10].

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) MQTT の利用
メッセージ配信プロトコルはブリッジが実装されており，ヘッダが軽量な MQTT を利用する．

(2) エッジ層に Pub/Sub のブローカを導入
上位層から順にクラウド層，エッジ層，デバイス層の 3 層とし，エッジ層にはブローカを配置する．

4.2 アプローチ

アプローチを図 1 に示す．現在提供されている緊急通報システムでは，緊急通報の発信元の近隣に緊急車両が存在しても，直接データ共有が行えない．そこで，緊急車両へ直接メッセージを配信するために，次の条件を満たすアーキテクチャを提案する．

(1) エッジを自動車の車載エッジ，車外エッジ，緊急車両の車載エッジに分割し，トピックによって経由するエッジを分散しスケーラビリティを確保する．
(2) MQTT ブリッジを用いてエッジ間で緊急メッセージを含むデータを共有可能とする．
(3) 緊急メッセージを自動車から緊急車両に車外エッジを経由することで共有できる．

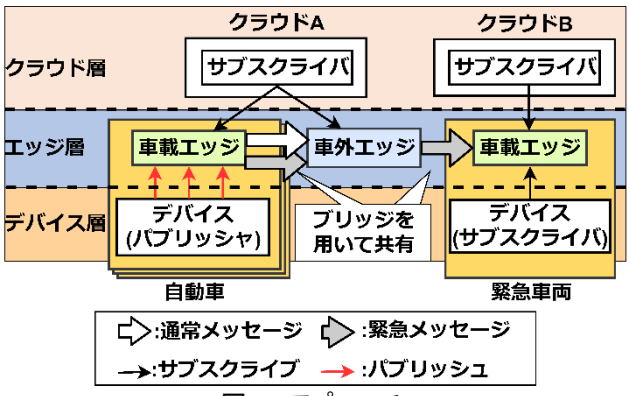

図 1 アプローチ

5 アーキテクチャ設計方法

5.1 設計プロセス

アプローチに従い，システムアーキテクチャを設計する．設計プロセスを以下に示す．

(1) 論理アーキテクチャ設計
ブリッジを用いてメッセージを共有できるアーキテクチャを設計する．

(2) メッセージの定義
メッセージの種類を通常メッセージと緊急メッセージに分け，それぞれのトピックを設計する．

(3) 物理アーキテクチャ設計
論理アーキテクチャに基づき，各機能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.

5.2 論理アーキテクチャの設計

本研究で提案するシステムアーキテクチャを図 2 に示す. まず自動車で生成されたメッセージはブローカ A に送信される. 負荷分散のため, 一部のメッセージはブローカ B にブリッジによって共有される. 緊急メッセージが送信された場合, まずブローカ A がメッセージを受け取り, ブローカ B に共有する. さらに, ブローカ B はブローカ C にブリッジによってメッセージを共有する. これにより, クラウドを中継せずに緊急指令室, 緊急車両にメッセージを配信できる.

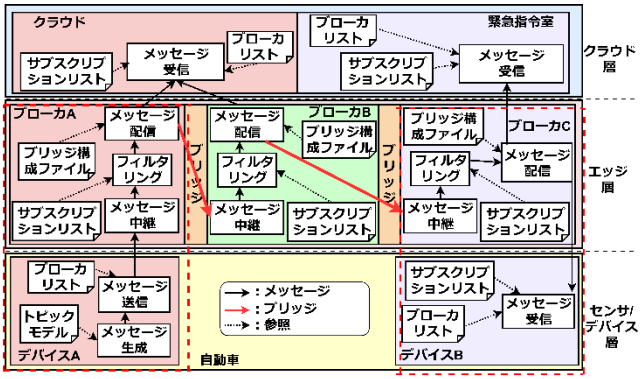

図 2 論理アーキテクチャ

5.3 メッセージの設計

5.3.1. メッセージトピックの設計

メッセージトピックの構造を図 3 に示す. 送信するメッセージのトピックは 4 階層で構成される. 第 2 階層で通常メッセージと緊急メッセージの識別, 第 3 階層でデータ概要の識別, 第 4 階層でメッセージを一意に識別する構造とする.

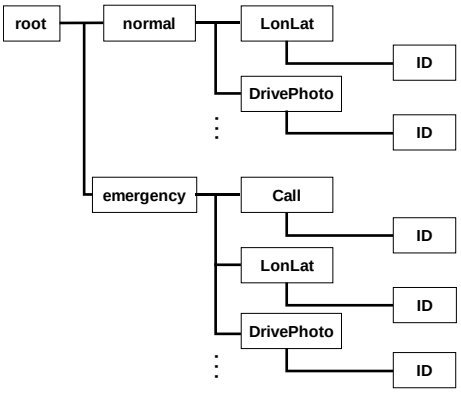

図 3 トピックの構造

5.3.2. メッセージ設計

送信するメッセージは,「接続機関における自動車からの緊急通報の取扱いに関するガイドライン」に基づいて定義する[8]. メッセージのトピックごとにレイテンシを計測するためメッセージは JSON 形式とし, メッセージの生成時刻をデータとともに送信する.

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. 物理アーキテクチャを図 4 に示す. 自動車, 緊急車両にはデバイスと車載エッジが配置されている. それぞれの車載エッジは車外エッジとブリッジによってメッセージの共有を行う. さらにクラウド層では, 通常メッセージを収集するクラウド, 緊急メッセージを収集する緊急指令室が配置されている.

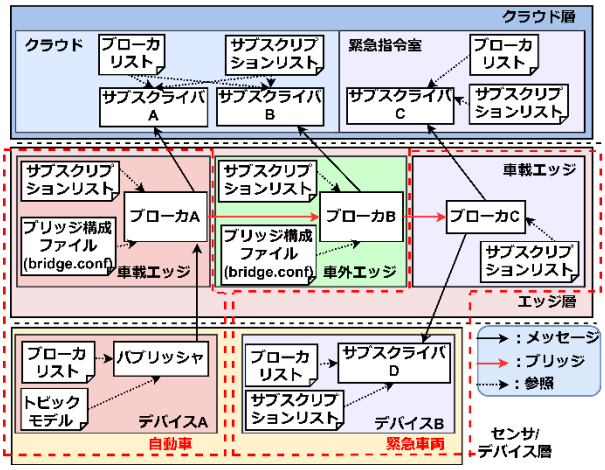

図 4 物理アーキテクチャ

6 プロトタイプの実装

6.1 プロトタイプ実装の目的

プロトタイプの実装の目的は次の 3 点である.

(1) 提案アーキテクチャの妥当性の確認

(2) 多段 Pub/Sub アーキテクチャ[6]と比較し, 二重エッジアーキテクチャ設計方法の優位性の確認

(3) 有効なメッセージ分散パターンの確認

6.2 プロトタイプ実装

プロトタイプの実装にあたり, メッセージ配信プロトコルには MQTT を用いる. ブローカの実装は Eclipse Mosquitto[4], クライアントの実装は Eclipse Paho[5]を用いる.

6.3 実行環境

プロトタイプの実行環境を表 1, 表 2 に示す.

表 1 実行環境

システム名	デバイス	エッジ	クラウド, 緊急指令室
ハードウェア	Raspberry Pi 4 B	Raspberry Pi 3 B+	FMVWWD2S8
OS	Raspbian 10	Raspbian 9.4	Ubuntu 18.04
プロセッサ	Cortex-A72,1.5GHz	Cortex-A53,1.4GHz	Intel Core i7-6700,3.40GHz
メモリ	4GB	1GB	16GB

表 2 ソフトウェア環境

ソフトウェア	実装範囲	バージョン
Eclipse Mosquitto	ブローカ	1.6.8
Eclipse Paho	クライアント	1.4.0

6.4 実行パターン

本プロトタイプは 4 つの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. 実行パターンを図 5 に示し, それぞれのパターンについて以下に詳細を示す.

(1) ブリッジパターン: 自動車の車載エッジと車外エッジをブリッジ接続し, 通常メッセージを分散してクラウドに配信するパターン. さらに車外エッジと緊急車両の車載エッジをブリッジ接続し, 緊急メッセージを緊急車両および緊急指令室に配信する.

(2) 多段 Pub/Sub パターン: メッセージ配信シーケンスは(1)と同様で, ブリッジ接続でなく多段 Pub/Sub によって配信する.

(3) 単一ブリッジパターン: 車外エッジのブローカを用いずメッセージ配信を行う. 緊急メッセージを受け取った車載エッジのブローカが緊急車両のブローカにブリッジを用いて緊急メッセージを共有することで緊急メッセージを緊急車両および緊急指令室に配信する.

(4) 単一多段 Pub/Sub パターン: メッセージ配信シーケンスは(3)と同様で，多段 Pub/Sub によって配信する．

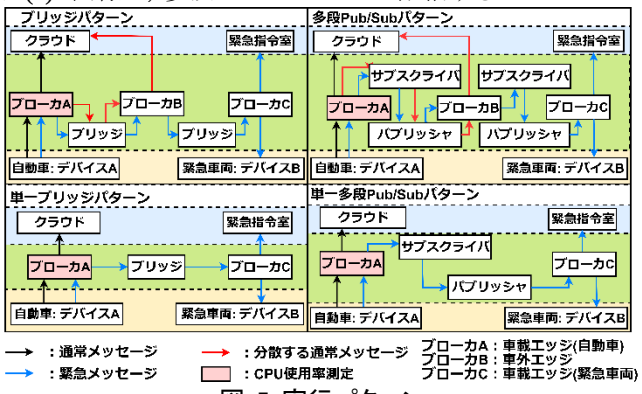

図 5 実行パターン

6.5 実行シナリオ

ブリッジパターンの実行シナリオを図 6 に示し，以下に詳細を示す．

(1) デバイス A 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．

(2) 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．

(3) 同様にブローカ B はブリッジ構成ファイルに基づきブローカ C 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 C に緊急メッセージを送信する．

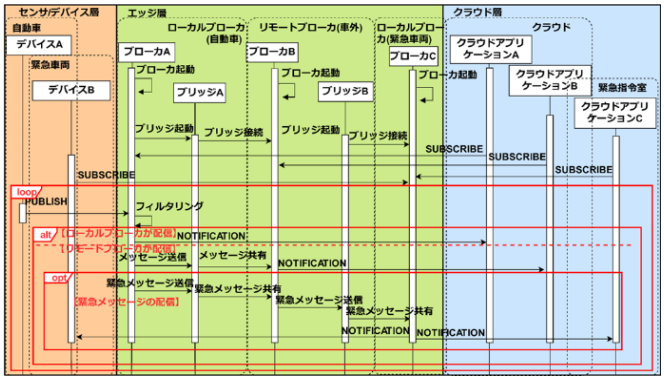

図 6 ブリッジパターンの振る舞い

6.6 適用事例

適用事例として危険運転の通報を用いる(図 7)．ドライバーが自車に対して危険行為をする自動車を見つけ，車内の緊急通報ボタンを押下する．押下により自動車のデバイスで生成したデータをブローカに送信し，緊急メッセージとして緊急車両に送信する．また，通常時は車両の速度や位置情報などのセンサ情報を常時クラウドに送信している．

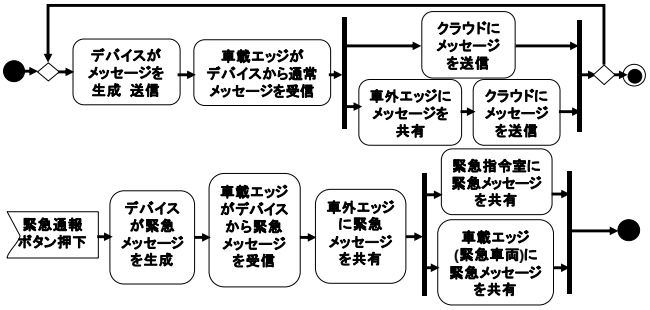

図 7 アクティビティ図

7 評価

7.1 評価方法

通常メッセージと緊急メッセージとしてそれぞれ 100KB，300KB，500KB の画像と 5.3.2 節で示した文字列をまとめて配信する．まずデバイス A から 0.1 秒ごとに通常メッセージを配信する．メッセージ送信数が 200 になった時点から通常メッセージに加えて緊急メッセージも配信する．レイテンシはパブリッシャがパブリッシュした時刻とサブスクライバがサブスクライブした時刻の差として測定する．また，ブローカ A の CPU 使用率を 0.1 秒間隔で測定する．6.4 節で示した 4 つの実行パターンで 3 回測定を行う．車外エッジが存在するパターンでは，次の 2 つのシーケンスを実行する．

(1) ブローカ A でフィルタリングした画像データをブローカ A からクラウドへ配信

(2) ブローカ A でフィルタリングした画像データをブローカ B に送り，さらにブローカ B からクラウドへ配信

7.2 実行結果

7.2.1. レイテンシ

実行結果を表 3 に示す．

表 3 緊急車両のレイテンシの平均値(ms)

実行パターン	シーケンス	100KB	300KB	500KB
ブリッジ	(1)	42.6	110.7	183.1
	(2)	44.2	120.1	184.4
多段 Pub/Sub	(1)	58.6	137.4	216.2
	(2)	59.5	133.8	215.3
単一ブリッジ		39.2	94.9	162.9
単一多段 Pub/Sub		49.9	140.1	198.2

7.2.2. CPU 使用率

ブローカ A の CPU 使用率の平均値は，図 5 で示す自動車が通常メッセージと緊急メッセージの両方を送信している時とし，表 4 に示す．

表 4 ブローカ A の CPU 使用率の平均値(%)

実行パターン	シーケンス	100KB	300KB	500KB
ブリッジ	(1)	5.1	8.3	9.5
	(2)	4.1	6.6	8.7
多段 Pub/Sub	(1)	17.1	16.4	21.1
	(2)	14.6	16.3	22.8

7.3 評価

7.3.1. 自動車から緊急車両への配信のレイテンシの比較

図 5 に示した各パターンでの緊急車両の表 3 に示すレイテンシの線形近似を図 8 に示す．

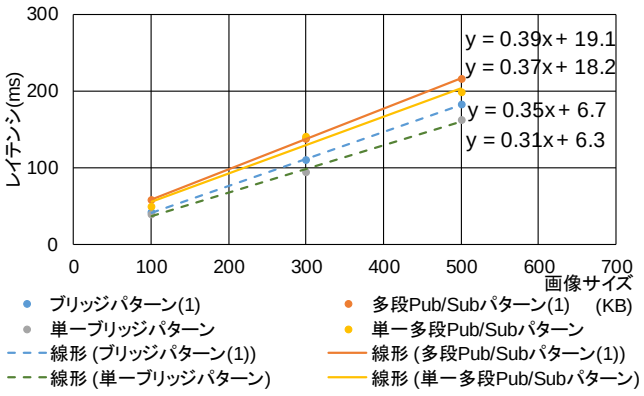

図 8 緊急車両のレイテンシ

ブリッジパターン(1)は多段 Pub/Sub パターン(1)と比較し，レイテンシは最大で 33.08 ミリ秒，線形近似の傾きは 11.4%の減少が確認できた．このことから，レイテンシにおいてブリッジパターンは多段 Pub/Sub パターンに比べ優位であることが確認できた．

また，ブリッジパターン(1)と単一ブリッジパターンを比較すると，線形近似の傾きは 12.9%増大していた．ネットワーク上でデータの送受信を行う際に最適な経路を割り出すルーティングを考慮すると傾きの増大は妥当であると言える．

図 5 に示すデバイス A が通常メッセージと緊急メッセージを送信している状態と通常メッセージのみ送信している状態でのレイテンシの線形近似を図 9 に示す．近似の傾きは画像データで 25%，軽量文字列データで 42%増大している．このことから，ブローカを経由するメッセージ数の増大に比例してレイテンシが増大することが確認できた．

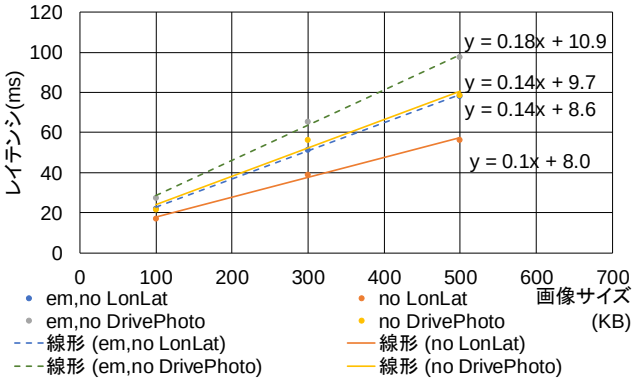

図 9 クラウドのレイテンシ

7.3.2. CPU 使用率の比較

図 5 で示すデバイス A が通常メッセージと緊急メッセージの両方を送信している状態での CPU 使用率の線形近似を図 10 に示す．多段 Pub/Sub パターンでは近似の傾きが 51.5%減少しており，ブリッジパターンでは 6.9%減少している．このことから，CPU 使用率は送信データ量の増大に比例することが確認できた．

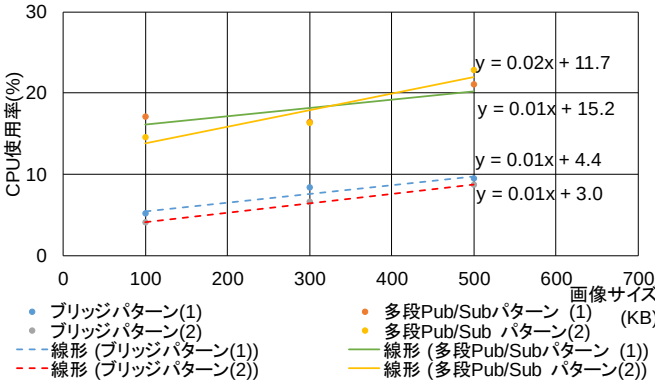

図 10 CPU 使用率

8 考察

8.1 緊急通報システムアーキテクチャへの適用

実現されている緊急通報システムでは，人手を介しているため，通報までに時間がかかる．提案システムアーキテクチャでは，通報までのプロセスを自動化することにより，迅速な通報が可能になる．さらに，近隣の緊急車両に緊急通報を配信することにより，緊急指令室からの人手による指令を受信するプロセスを削減できる．これらによって，緊急車両が現場に到着するまでの時間を短縮できると考えられる．

8.2 二重エッジアーキテクチャのスケーラビリティ

本研究では，二重エッジアーキテクチャをさらに拡張し，3 つのエッジ間でのメッセージ共有を行うアーキテクチャを提案した．プロトタイプを実装し，レイテンシと CPU 使用率を測定した．実行結果より二重エッジアーキテクチャは多段 Pub/Sub アーキテクチャに対しレイテンシを短縮できることを確認した．さらに，エッジの数が増大しても，二重エッジアーキテクチャでは CPU 使用率とレイテンシは比例して増大する．これらのことより，二重エッジアーキテクチャはスケーラビリティがあると言える．

9 今後の課題

今後の課題は次の 2 点である．

(1) 大量データでの性能評価

LIDAR を用いたセンサデータの活用や該当車両近隣の車両と連携することにより，大量データに対しても二重エッジアーキテクチャの性能を評価する必要がある．

(2) 緊急通報システム以外のサービスへの適用

スケーラビリティ，リアルタイム性が求められるサービスに二重エッジアーキテクチャが適用できるか検討する必要がある．

10 まとめ

緊急通報システムにエッジコンピューティングを適用したシステムアーキテクチャ設計方法を提案した．MQTT-Bridge を用いた二重エッジアーキテクチャを用いて，スケーラビリティの高いシステムアーキテクチャを実現した．提案アーキテクチャのプロトタイプを実装し，レイテンシと CPU 使用率を測定した．異なるエッジアーキテクチャを適用した場合との比較から，提案アーキテクチャが先行研究の多段 Pub/Sub アーキテクチャよりもレイテンシと CPU 使用率を低減できることを確認した．

参考文献

[1] AECC, General Principle and Vision White Paper Version 2.1.0, Dec. 2018, https://aecc.org/wp-content/uploads/2019/04/AECC_White_Paper_v2.1_003.pdf.

[2] 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.

[3] CEN-EN 16072: 2015, Intelligent Transport Systems–ESafety-Pan European eCall-Operating requirements.

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

[5] Eclipse Paho, https://eclipse.org/paho/.

[6] 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.

[7] ISO/IEC 20922:2016, Information Technology - Message Queuing Telemetry Transport (MQTT), V3.1.1, 2016.

[8] 警察庁 消防庁 国土交通省, 接続機関における自動車からの緊急通報の取扱いに関するガイドライン, May 2018, http://www.mlit.go.jp/common/001234105.pdf.

[9] P. T. Eugster, et al., The Many Faces of Publish/Subscribe, ACM Computing Survey, Jun. 2003, pp. 114-131.

[10] 宇野 聡将 他, 二重エッジアーキテクチャ設計方法の提案と MQTT-Bridge を用いたプロトタイプによる評価, 情報処理学会第 81 回全国大会講演論文集, Mar. 2019, pp. 215-216.

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