MQTT を用いた車載エッジコンピューティングアーキテクチャの提案と評価

2013SE053 井ノ口 仁也 2013SE233 宇野 聡将

指導教員 青山 幹雄

1. 研究背景と目的

車載システムと車外システムとの連携が進められ，車載システムで発生したセンサ情報のシームレスな共有が望まれる[1][3].そのため，車載ゲートウェイで車載ネットワークを介したセンサデータを活用するアーキテクチャが必要となる．車載システムで発生するイベントを，車載ブローカを用いて処理し外部との連携を行うための Publish/Subscribe アーキテクチャが提案されている[5]．Publish/Subscribe アーキテクチャの実装として MQTT（MQ Telemetry Transport）が ISO で標準化され，IoT への応用が期待されている[4]．本稿では，MQTT を用いた外部システムとの連携を行うアーキテクチャを提案する．

2. 研究課題

研究課題として以下をあげる．

(1) 車載システムで発生するイベントを外部システム上で共有及び活用可能なアーキテクチャ
(2) 車載センサデータの動的フィルタリング

3. 関連研究

3.1. Publish/Subscribe アーキテクチャ

Publish/Subscribe アーキテクチャとは，メッセージを送信するパブリッシャ，メッセージを受信するサブスクライバ，メッセージの中継を行うブローカで構成される非同期メッセージ配信アーキテクチャである．メッセージフィルタリングにはトピックベース，コンテンツベース及びそれらを組み合わせたタイプベースがある[2].

3.2. MQTT (MQ Telemetry Transport)

MQTT とは Publish/Subscribe アーキテクチャの実装として ISO で標準化されている通信プロトコルである．メッセージフィルタリングはトピックベースが利用可能である[4].

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，IoT を構成するデバイスとクラウド間にあるネットワーク機能を持つエッジサーバにクラウドの機能の一部を配置する．エッジサーバ上で処理を行うため高いリアルタイム性を要求され，通信頻度や通信量も膨大となるアプリケーションへの適用が可能となる．エッジコンピューティングアーキテクチャは下位層からセンサ/デバイス層，エッジコンピューティング層，インターネット層，クラウド層から構成される[5].

4. アプローチ

本稿では，車載センサで発生するデータをエッジサーバ上で処理するフィルタリングブローカを設置することで外部システムにおいてセンサデータの効率的な活用を可能にするブローカアーキテクチャを提案する．大量発生するセンサデータの送受信には Publish/Subscribe アーキテクチャを適用し各システム間の連携を実現する．Publish/Subscribe アーキテクチャの実装として MQTT を用いる．

エッジコンピューティングアーキテクチャに基づき車載センサをセンサ/デバイス層，車載ゲートウェイ及びフィルタリングブローカ，エッジアプリケーションをエッジ層，外部システムをクラウド層へ適用させる．この時，センサ/デバイス層とエッジ層は車内環境にあると定義する．

5. 提案アーキテクチャ

5.1. 機能アーキテクチャ

図 1 に機能アーキテクチャを示す．車載ゲートウェイではイベントの取得と解析を行い，メッセージをフィルタリングブローカへ送信する．フィルタリングブローカではメッセージのフィルタリングを行い，結果を外部システムへ送信する．外部システムはセンサデータの受信と活用を行う．

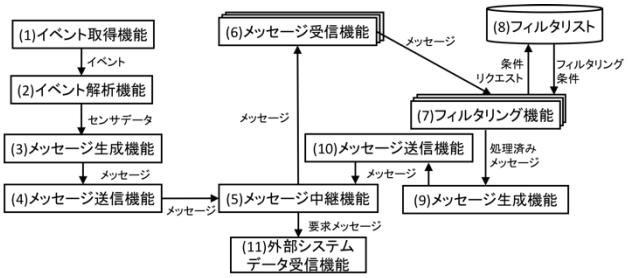

図 1 機能アーキテクチャ

(1) イベント取得機能：イベントを取得する．
(2) イベント解析機能：イベントを解析し，イベント情報を取得する．
(3) メッセージ生成機能（車載ゲートウェイ）：イベント情報からトピック及びメッセージの生成を行う．メッセージはセンサ値，発生時刻を持つ．
(4) メッセージ送信機能（車載ゲートウェイ）：メッセージをデータ中継機能へと送信する．
(5) メッセージ中継機能：メッセージをトピックに基づいてメッセージ受信機能へと配信する．
(6) メッセージ受信機能（フィルタリングブローカ）：サブスクライブ要求を行なっているトピックのメッセージを受信する．受信後フィルタリング機能へ送信する．
(7) フィルタリング機能：システムが設定した条件に基づきメッセージのフィルタリングを行う．対象メッセージはメッセージ生成機能へ送信する．
(8) フィルタリスト：センサ値によるフィルタリングを行うための条件を記述したファイル．フィルタリング機能によって参照される．

(9) メッセージ生成機能(フィルタリングブローカ):外部システムへ送信するメッセージを生成する. メッセージはセンサの種類, フィルタリング結果, センサ値, タイムスタンプを持つ.
(10) メッセージ送信機能(フィルタリングブローカ):外部システムへのメッセージ送信を行う.
(11) 外部システム受信機能:要求を行なったメッセージを受信する. センサ値を利用する実行環境を持つ.

5.2. 物理アーキテクチャ

機能アーキテクチャ基づき, MQTT ブローカのスケールアウトを実現するための物理アーキテクチャを定義する. 本稿では単一ブローカとマルチブローカの 2 パターンの構成を提案, 比較を行いマルチブローカアーキテクチャの有用性を検証する. 図 2 にマルチブローカとして 2 つの MQTT ブローカを用いて構成した物理アーキテクチャを示す.

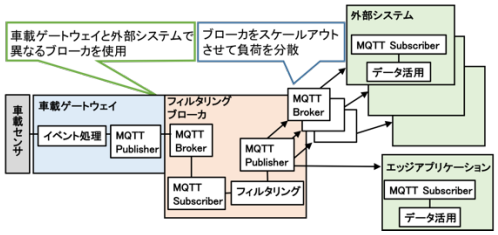

図 2 マルチブローカパターン

5.3. イベントモデル/トピックモデル

本稿で用いるイベントモデルとトピックモデルを定義する.

5.3.1. イベントモデル

提案アーキテクチャは車載ネットワーク上にある車載センサで発生するイベントを処理する構造を持つ. イベントの処理はイベントモデルに基づき, 車載センサ単位で発生するイベントを用いてフィルタリングブローカ送信時に使用するトピックを決定する. 図 3 にイベントモデルを示す.

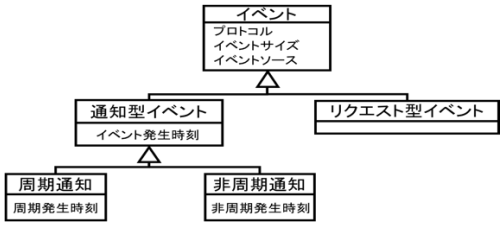

図 3 イベントモデル

イベントには「通知型」, 「リクエスト型」が存在する. 本稿において発生するイベントは車載システム上にある車載センサ単位で発生するイベントとして定義する. したがって, 提案アーキテクチャではイベントモデルに基づき, 「通知型」イベントとしてイベントを扱う.

5.3.2. トピックモデル

本稿で用いるトピックモデルを定義する. トピックは root/Level2/Level3 の 3 階層をトピックストリングと定義する.
(1) root:トピックツリー第 1 階層の root にはフィルタリングブローカ及び, 外部システムへのパスを記述する.
(2) Level2:トピックツリー第 2 階層の level2 にはフィルタリングブローカ上で起動するサブスクライバへのパスを記述する.
(3) Levle3:トピックツリー第 3 階層の level3 にはフィルタリングブローカ上で実行するフィルタリング処理時に参照するフィルタリストへの識別子として記述する.

5.4. イベント処理

イベントの取得及び MQTT 通信に使用するメッセージ生成処理を図 4 に示す.

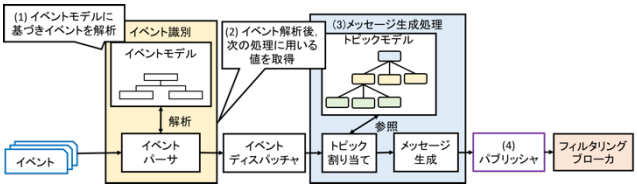

図 4 メッセージ生成

メッセージ生成の過程を以下に述べる.
(1) イベント識別:イベントモデルに基づいてイベント解析.
(2) 値の取得:イベントの保持する情報を取得.
(3) メッセージ生成:イベント情報を元にトピックを割り当て, イベント情報を付与したメッセージを生成.
(4) パブリッシュ:トピックストリングに基づきメッセージ送信.

5.5. フィルタリング処理

フィルタリングブローカ上でのメッセージフィルタリング過程を図 5 に示す. MQTT におけるフィルタリングにはトピックベースフィルタリングが実装されている. 提案フィルタリングプロセスでは, トピックベースでのフィルタリングをメッセージ受信時に一次フィルタリングとして実行する. その後, トピックストリング情報に基づき, 動的に条件を変更させメッセージのコンテンツ(内容)に着目した 2 次フィルタリングを行う.

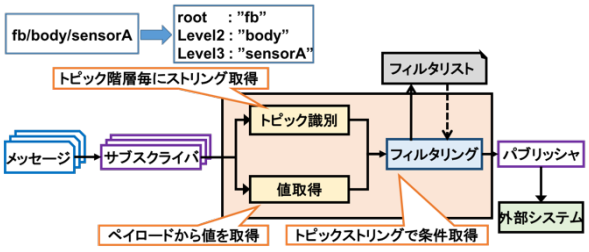

図 5 フィルタリング処理

6. プロトタイプ実装

6.1. 実行シナリオ

作成するプロトタイプの実行シナリオはデータ生成アプリケーションから生成したセンサデータをメッセージ生成アプリケーションへ送信し, その後フィルタリングブローカへのメッセージを生成しフィルタリングアプリケーションへメッセージを送信する. フィルタリングアプリケーションでフィルタリングを行いその結果をエッジアプリケーションへ送信する. 図 6 に実行シナリオを示す.

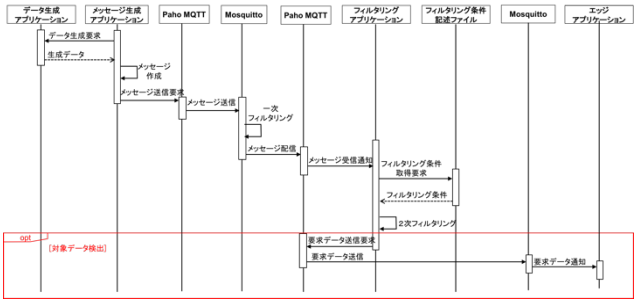

図 6 実行シナリオ

6.2. 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	車載ゲートウェイ	フィルタリングブローカ	エッジアプリケーション
ハードウェア	MacBook Pro	Raspberry Pi	LIFEBOOK S762/F
OS	OS X El Capitan10.11.6	Debian 7.11	Ubuntu 12.04 LTS
プロセッサ	2.9 GHz Intel Core i5	ARM1176JZF-S700MHz	Intel Core i5-3320M CPU @ 2.60GHz x 4
メモリ	16GB	256MB	4GB
ストレージ	256GB SSD	16GB SD カード	32.7GB
実装言語	python 2.7.10	python 2.7.3	python 2.7.3

表 2 ソフトウェア情報

ソフトウェア	バージョン
Mosquitto	1.4.10
Paho-MQTT client	1.2

6.3. プロトタイプの構成

プロトタイプを提案物理アーキテクチャに基づき MQTT ブローカを単一で起動させた場合と, 2 つの MQTT ブローカを並列起動させる場合の 2 つのパターンを実装する(図 7).

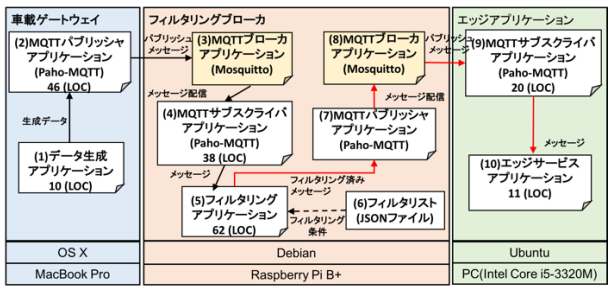

図 7 プロトタイプ構成:マルチブローカ

実装する 2 パターンのプロトタイプに, 例題を適用させシステムの一連の処理時間を計測, 比較検証を行う.

7. プロトタイプの適用と実行結果

7.1. プロトタイプ適用

プロトタイプ構成に車載システム上の異常検知を適用する. 車両の状態を表現する値を生成, メッセージを作成して送信を行う. さらにフィルタリングブローカに上でメッセージのフィルタリングを行い, 異常値を持つメッセージを検出する. そして, 異常値を持つメッセージをエッジアプリケーションに送信する. エッジアプリケーションは異常値データのみを取得する.

7.2. 実行結果

プロトタイプを実装し, 例題へ適用, 実行を行った. 以下にその実行結果として, 車載ゲートウェイ上でのメッセージ送信状況を図 8 に, フィルタリングブローカ上での受信状況を図 9 に, エッジアプリケーション上でのメッセージ受信状態を図 10 に示す.

```
************************************************************
topic:fb/body/a pub_time:2016-12-28 13:09:20.042976 value:61
************************************************************
topic:fb/info/a pub_time:2016-12-28 13:09:20.043258 value:-30
************************************************************
------------------------------------------------------------
topic:fb/vc/d pub_time:2016-12-28 13:09:20.043559 value:45
------------------------------------------------------------
```

図 8 車載ゲートウェイ

```
-------------------------message information-------------------------
topic: fb/vc/e payload: {"date": "2016/12/28 13:09:20", "value": -35}
---------------------------------------------------------------------
False
Found Abnormal_Value
new topic:ea/vc/e
new payload:{u'date': u'2016/12/28 13:09:20', 'flag': False, u'value': -35}
```

図 9 フィルタリングブローカ

```
ea/body/a "{u'date': u'2016/12/28 13:09:18', 'flag': False, u'value': -50}"
receive time:2016/12/28 13:09:19
ea/body/d "{u'date': u'2016/12/28 13:09:19', 'flag': False, u'value': -43}"
receive time:2016/12/28 13:09:19
ea/info/a "{u'date': u'2016/12/28 13:09:19', 'flag': False, u'value': -59}"
receive time:2016/12/28 13:09:19
ea/vc/e "{u'date': u'2016/12/28 13:09:19', 'flag': False, u'value': -54}"
receive time:2016/12/28 13:09:19
```

図 10 エッジアプリケーション

図 8 からセンサ値にトピックが割り当てられていることがわかる. 図 9 では, フィルタリングブローカ上でのメッセージ受信状態とフィルタリング結果が示されている. 結果は, true が正常値, false が異常値を示している. 図 10 で, エッジアプリケーション上で異常値に関する false のフラグを保持するメッセージのみが正常に受信できていることが示せた. また, メッセージの保持する情報としてセンサ値, タイムスタンプ, 異常個所の表現が実現できていることが分かる.

7.3. 比較検証

プロトタイプを実行し, その処理性能を比較する. 計測方法を図 11 に示す. 単一ブローカパターンの遅延時間を T_1, マルチブローカパターンの遅延時間を T_2 とし, マルチブローカパターンでのブローカ単体の時間をそれぞれ T_{21}, T_{22} として定義する.

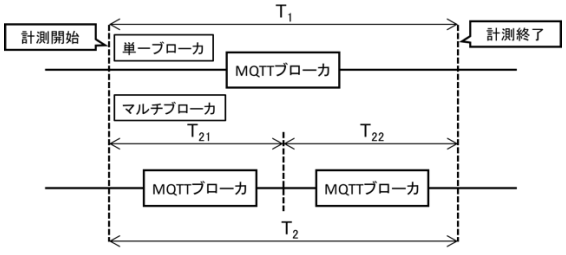

図 11 計測方法

イベントは 200 件発生させその処理の遅延時間を計測した．メッセージ送信周期は，60msec と 100msec の 2 パターンで計測した．上記ケースを 2 回行い，検証材料として用いる．図 12 にパターン毎に遅延時間を表したグラフを，表 3 にそれぞれの最大値と最小値，及び平均値と標準偏差を，図 13，図 14 に処理時間の散布図を示す．

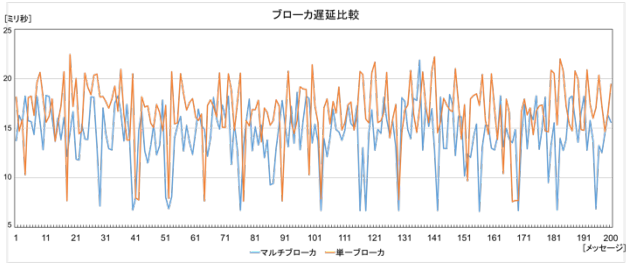

図 12 遅延時間比較のグラフ

表 3 遅延時間

遅延時間(msec)	最大値	最小値	平均値	標準偏差
T_1	22.9	7.5	17.1	2.53
T_2	22.7	6.6	14.6	2.48
T_{21}	11.4	3.3	7.3	—
単体向上率 (%)	50	44	42	—

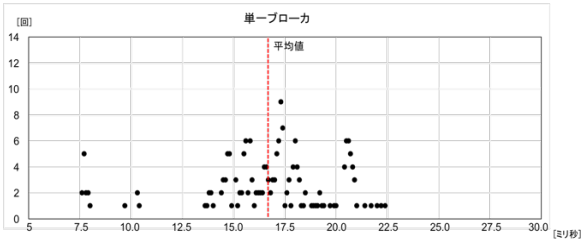

図 13 単一ブローカ散布図

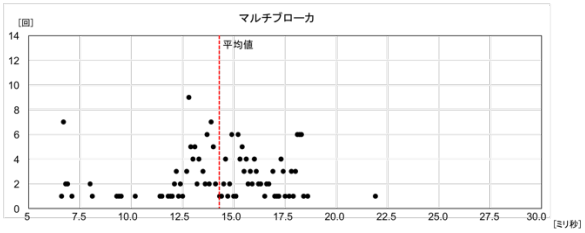

図 14 マルチブローカ散布図

遅延時間の平均値，T_1=17.1，T_2=14.6 より，処理全体を通して T_2 の遅延時間が平均 2.5msec，約 15% T_1 よりも短縮できた．また，ブローカ単体での遅延時間も T_1 と T_{21}＝7.3msec の遅延時間を比較すると約 50%短縮することができた．この結果から，マルチブローカパターンでの処理能力の優位性を示すことができた．

8. 提案アーキテクチャの評価

(1) フィルタリング

動的にフィルタリング条件を変更可能にすることで，多様なイベントに対してのフィルタリング処理が可能となった．

(2) 外部システムとの相互運用性

提案アーキテクチャでは Publish/Subscribe アーキテクチャの実装として MQTT を適用させたことで外部システムと車載システム間での車載センサデータの共有及び活用が可能になり，例題に適用しその検証を行なった．その結果，提案アーキテクチャにおける相互運用性を満たすことができたと考えられる．

(3) 実装パターンの比較

マルチブローカを用いた実装パターンにおいて処理性能の優位性を示すことができた．また，MQTT ブローカをスケールアウトすることでメッセージの処理時間を短縮可能であることが示せた．

9. 今後の課題

(1) リクエスト型イベントの対応

外部システムからの要求をリクエスト型イベントとし，フィルタリングの仕様変更等のリクエストに対応可能にする．

(2) 組み合わせフィルタリング

より多様なフィルタリングを実行するため，センサデータを組み合わせたフィルタリング処理の実行を可能にする．

(3) 処理イベントの優先度決定

イベントを発生順に処理を行わず，イベントに対して優先度を決定しフィルタリング処理を行う仕組みの提案．

10. まとめ

本稿では，車載システム上にセンサデータのフィルタリングを行うフィルタリングブローカを設置したブローカアーキテクチャを提案した．フィルタリングにおいて，動的にフィルタリング条件を変更可能にすることで 多様なイベントに対しての処理の実行が可能になった．また，プロトタイプを実装し例題へ適用させてシステムの相互運用性とブローカスケールアウトの評価をおこなった．

参考文献

[1] R. Coppora, et al., Connected Car: Technologies, Issues, Future Trends, ACM Computing Surveys, Vol. 49, No. 3, Article 46, Oct. 2016.

[2] デンソーカーエレクトロニクス研究会，図解カーエレクトロニクス (下) 要素技術編，日経 BP 社，2012.

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

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

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