4T-07

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

宇野　聡将†　　井ノ口　仁也†　　青山　幹雄†

南山大学　理工学部　ソフトウェア工学科†

1. はじめに

自動車における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].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．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

研究課題は以下の 2 つとする.

(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と活用が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 MQTT(MQ Telemetry Transport)

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[3].

3.2 IoT/M2M

センサを含むデバイス同士またはデバイスとクラウド上のサーバとをネットワークを介して接続し，収集されたデータを送受信することにより様々なサービスを提供するシステムである．IoT アーキテクチャはデバイス，ゲートウェイ，ネットワーク，IoT サービスで構成されている[4].

3.3 エッジコンピューティング

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[5].

4. アプローチ

エッジサーバにフィルタリングブローカを配置し，外部システム上でセンサデータを効率的に活用可能にするブローカアーキテクチャを提案する．システム間連携には， MQTT を用いる．また，エッジコンピューティング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する．センサ/デバイス層とエッジ層は車内環境にあると仮定する.

An Edge Computing Architecture using MQTT for Automotive Systems

†1 Toshimasa Uno, Jinya Inoguchi, Mikio Aoyama, Dep. of Software Engineering, Nanzan University.

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.

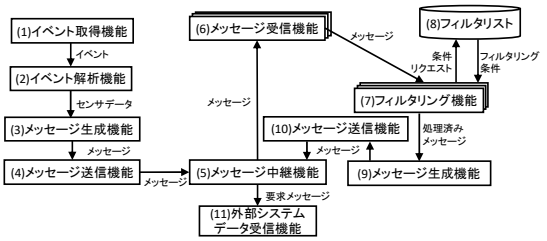

図 1 機能アーキテクチャ

5.2 マルチブローカアーキテクチャ

MQTT ブローカのスケールアウトを実現するためのマルチブローカパターンの物理アーキテクチャを提案する(図 2).

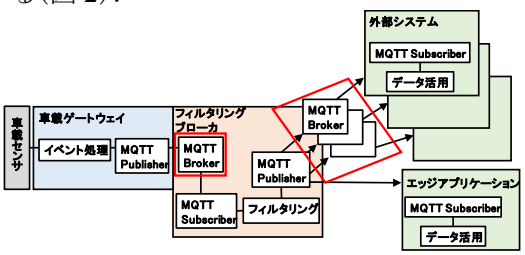

図 2 マルチブローカアーキテクチャ

5.3 フィルタリング

一次フィルタリングとしてトピックベースフィルタリングをメッセージ受信時に行う．その後，トピックストリングに基づき動的にフィルタリング条件を変更させメッセージのコンテンツに着目した二次フィルタリングを実行する.

6. プロトタイプの適用と実行結果

6.1 プロトタイプ構成

マルチブローカでの構成を以下に示す(図 3).

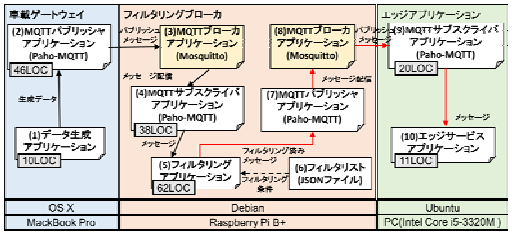

図 3 マルチブローカパターン

Copyright ©2017 Information Processing Society of Japan.
All Rights Reserved.

プロトタイプは単一ブローカとマルチブローカの 2 パターンで実装を行う．単一ブローカは MQTT ブローカを一つのみ配置し車内外全てのメッセージ中継を行う．両パターンの処理性能を比較しマルチブローカパターンの有用性を示す．

6.2 プロトタイプ適用

車載システムへ異常検知を適用する．生成したデータをパブリッシャへ送信し，メッセージを生成後フィルタリングブローカへ送信する．条件に基づき動的フィルタリングを実行し異常値データを検出し，エッジアプリケーションへ通知する．

6.3 実行結果

図 4 では，センサ値にトピックが割り当てられメッセージが生成されている．図 5 では，フィルタリングブローカ上での結果を示す．true が正常値，false が異常値を示す．図 6 が示すエッジアプリケーション上では異常値の false 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．メッセージ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た．

```
**************************************************************
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 4 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 5 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 6 エッジアプリケーション

6.4 比較検証

2パターンのアーキテクチャを実行し，遅延時間の比較を行う．計測開始はメッセージ送信時，計測完了はエッジアプリケーションでのメッセージ受信時とする．テストケースとして，パブリッシャは 60msec と 100msec の送信周期で，200 件のメッセージの送信した．また，同ケースを2回行い計測の妥当性を確認する．以下に遅延時間比較のグラフを示す（図 7）．

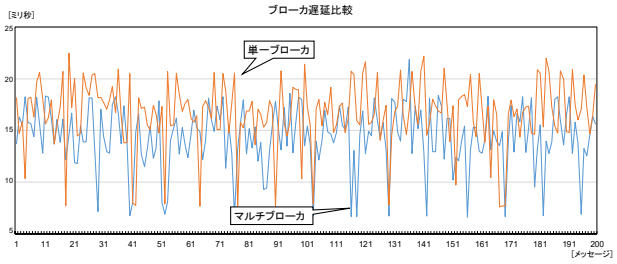

図 7 遅延時間比較

遅延時間を単一ブローカを T_1，マルチブローカを T_2，T_2 でのブローカ単体の遅延時間を T_{21} として定義する．

表 1 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	－
単体削減率(%)	50	44	42	－

平均値，T_1=17.1，T_2=14.6 より，処理全体で T_2 の遅延時間が平均 2.5msec，約 15% 短縮できた．ブローカ単体の遅延時間も T_1 と T_{21}＝7.3msec を比較すると約 42%短縮できた．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．

7. 提案アーキテクチャの評価

(1) 動的フィルタリング：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：MQTT を適用させたことで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となった．例題に適用し評価を行ない，相互運用性を確認した．

(3) 実装パターンの比較：マルチブローカを実装し，処理性能の優位性を示した．また，MQTT ブローカをスケールアウトによりメッセージ処理時間を短縮可能であることが示せた．

8. 考察

MQTT を用いて車内外でセンサデータを共有し，動的フィルタリングを実行することでセンサデータの外部システムでの効率的活用を可能にした．MQTT においてメッセージの内容に着目し，フィルタリングを行うことで，従来のトピックベースよりも細かく受信条件を設定が可能となり，車内外においてセンサデータの活用がより効果的に行えるようになると考えられる．

9. まとめ

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．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が可能になった．プロトタイプを実装し例題へ適用しシステム間の相互運用性とスケールアウトを評価した．

参考文献

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

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

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

[4] 富田 二三彦，山崎 徳和，MCPC，M2M/IoT 委員会 M2M/IoT 教科書，インプレス，2015.

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

Copyright ©2017 Information Processing Society of Japan.
All Rights Reserved.