

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

宇野 聡将[†] 青山 幹雄[†]

概要: 本稿は、エッジ上でトピックベースとコンテンツベースのブローカを接続し、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アーキテクチャを提案する。提案アーキテクチャのプロトタイプを実装し、提案アーキテクチャの有効性を示す。

キーワード: IoT, エッジコンピューティング, MQTT

A Multi-Broker Edge Architecture for Content-Based Filtering

TOSHIMASA UNO[†] MIKIO AOYAMA[†]

1. 研究背景と課題

IoT の普及によりネットワークでやりとりされる情報が増大している。例えば、車載システムの故障診断ではクラウドに車載センサーデータを収集することで、故障の予測や通知を可能とする。このようなデータはコンテンツベースで非同期な収集が必要である。本稿では、エッジ上でトピックベースとコンテンツベースのブローカを接続し、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アーキテクチャを提案する。提案アーキテクチャのプロトタイプを作成し、有効性を示す。

2. 関連研究

2.1 MQTT

MQTT は Publish/Subscribe アーキテクチャの実装である[3]。ブローカベースのため、パブリッシャとサブスクライバの追加/削除が容易である。メッセージのフィルタリングはトピックベースである。実装として Apache Mosquitto[1], Paho mqtt[2]がある。

2.2 エッジコンピューティング

IoT を構成するデバイスとクラウド間に配置されたネットワーク機能を持つサーバである[4]。デバイスが高頻度でデータをクラウドに送信する場合、エッジサーバ上でのフィルタリングより、必要なデータのみを収集することが可能である。

3. アプローチ

ブローカを接続したマルチブローカアーキテクチャを提案する。エッジにフィルタリングブローカを配置する。フィルタリングブローカは多段階 Publish/Subscribe 通信によってデータに基づくコンテンツベースフィルタリングを可能とし、ブローカへのメッセージ集中による遅延を軽減する。フィルタリング条件はトピックとデータに基づいた決定木によって定義する。

4. マルチブローカアーキテクチャ

4.1 マルチブローカアーキテクチャの構成

提案アーキテクチャの構成を図 1 に示す。センサ/デバイスは

データを収集し、パブリッシャとしてメッセージをエッジに送信する。エッジ上のフィルタリングブローカをトピックブローカ(TB)とコンテンツブローカ(CB)に分割する。TB は MQTT ブローカと同様のトピックベースフィルタリングを行う。コンテンツベースフィルタリングが必要なメッセージは CB に中継される。CB はメッセージ内のデータに基づきコンテンツベースフィルタリングを行う。TB と CB は Publish/Subscribe アーキテクチャにより接続する。これによって、トピックとコンテンツでのフィルタリングが分散できる。フィルタリング後の要求メッセージはエッジ上の他サーバか外部システムのサーバがサブスクライバとして受信可能である。

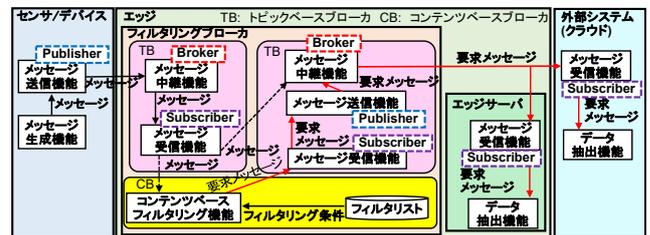


図 1 提案アーキテクチャ

4.2 段階的フィルタリング

提案アーキテクチャは TB と CB による多段階のフィルタリングを行う。CB におけるフィルタリングプロセスを図 2 に示す。フィルタリストは外部システムとエッジサーバがフィルタリングブローカに登録したトピックとフィルタリング条件の組を保持する。CB はトピックに基づきフィルタリング条件をフィルタリストから取得し、コンテンツベースフィルタリングを行う。



図 2 コンテンツベースフィルタリングプロセス

4.3 フィルタリング条件の抽出方法

フィルタリング条件の抽出プロセスを図 3 に示す。MQTT におけるトピックは階層構造であるので、メッセージからトピックを階層ごとにストリングとして抽出する。このストリングを用いてフィ

[†] 南山大学大学院 理工学研究科 ソフトウェア工学専攻

ルタリング条件を決定する。トピックによるフィルタリング条件は決定木として定義する。ルートからの深さごとに意味を定義する。

(1) ルート: メッセージ宛先

サブスクリバはサブスクリプションにワイルドカードの後方一致(#)を用いることでメッセージを取得する。サブスクリプション例として、”自身のID”/# と指定する。

(2) 深さ2: データ発生源(ネットワーク)

フィルタリングブローカは異なる発生源で同種のセンサで収集したデータをフィルタリングすることを想定している。一意にフィルタリング条件を決定可能とするため、決定木にデータ発生源とネットワークを定義する。

(3) 深さ3...n: センサタイプ

フィルタリングの対象となるセンサタイプをノードとする。同一のセンサタイプで異なるフィルタリング条件を指定可能とするため、木の深さを制限しない。

(4) 終端ノード: フィルタリング条件

コンテンツベースフィルタリング条件を終端ノードに配置する。

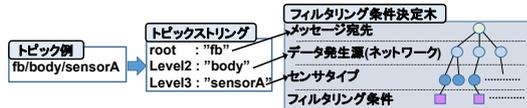


図 3 フィルタリング条件抽出プロセス

5. プロトタイプの実装

5.1 目的

提案アーキテクチャの有効性を評価するため、プロトタイプを実装した。プロトタイプは単一ブローカとマルチブローカの 2 パターンを実装し、処理性能を比較した。

5.2 プロトタイプの構成

プロトタイプの構成を図 4 に示す。本プロトタイプはエッジまでを実装範囲とし、全てのメッセージを TB, CB, TB の順に中継した。図 1 に基づき2つのMQTTブローカを配置、接続した。

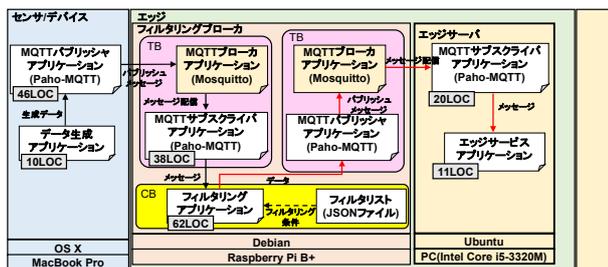


図 4 プロトタイプ構成

6. プロトタイプ評価

6.1 評価方法

プロトタイプを実行し、その処理性能を比較した。計測方法を図 5 に示す。メッセージ送信から受信までの所要時間を遅延時間と定義する。単一ブローカの遅延時間を T_1 、マルチブローカの遅延時間を T_2 とし、マルチブローカでのブローカ単体の遅延時間をそれぞれ T_{21} , T_{22} として定義する。

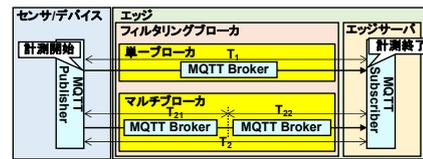


図 5 遅延計測方法

6.2 実行結果

メッセージを周期的に、合計 200 件送信して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。60 msec 周期の遅延時間を表 1 に示す。単一ブローカとマルチブローカの遅延時間の推移を図 6 示す。

表 1 遅延時間の測定結果

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	-
単体遅延短縮率(%)	50	44	42	-

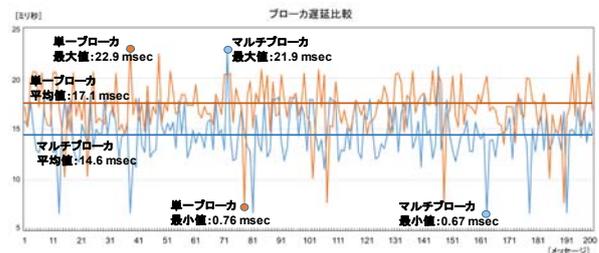


図 6 遅延時間の推移

6.3 評価

表 1, 図 6 から平均遅延時間はブローカを単一で配置した場合に 17.1 msec, マルチブローカは 14.6 msec であり, 2.5msec, 14.6%短縮できた。この結果からマルチブローカが有効であるといえる。

7. 今後の課題

- (1) フィルタリング処理時間を考慮した評価
- (2) フィルタリング条件登録/削除プロセス
- (3) 組み合わせフィルタリング

8. まとめ

本稿では、エッジ上でブローカを接続し、遅延を軽減可能とするマルチブローカアーキテクチャを提案した。提案アーキテクチャはエッジ上のフィルタリングブローカを TB と CB に分割し、Publish/Subscribe アーキテクチャにより接続した。これにより、トピックとコンテンツでのフィルタリングを分散した。プロトタイプを 2 パターン実装し、比較検証を行うことでマルチブローカの有効性を示した。

参考文献

- [1] Apache, Mosquitto, <https://mosquitto.org/>.
- [2] Eclipse, Paho, <http://www.eclipse.org/paho/>.
- [3] ISO/IEC 20922, Information Technology – Message Queuing Telemetry Transport (MQTT) v3.1.1, ISO, 2016.
- [4] W. Shi, et al., The Promise of Edge Computing, IEEE Computer, Vol. 49, No. 5, May 2016, pp. 78-81.