pub/sub メッセージングにおける負荷分散性と低遅延性の適応的制御
Adaptive Management of Load Balance and Timeliness in Pub/Sub Messaging

坂野　遼平 *　　首藤　一幸 †
Ryohei Banno　　Kazuyuki Shudo

概要

pub/sub メッセージングは送受信者が broker を介してデータのやり取りを行う通信モデルである．これまで，broker への負荷集中を回避するため，多数の broker にスケールアウトさせる手法について研究が為されてきた．しかしながら，従来の手法では，スケールアウトにより高いスループットが得られる一方で，分散処理のオーバヘッドとして配送遅延が増大する課題があった．本研究では，トピックベース pub/sub を対象とし，スケーラビリティと低遅延性の両立を可能とする適応的制御手法を提案する．提案手法では，多数の broker によりオーバレイネットワークを構築し，高負荷のトピックでは関連する broker 群で配送木を形成してマルチホップ転送により負荷分散を図る．一方，低負荷トピックについては，関連する broker 群にシングルホップの局所オーバレイネットワークを形成させ，低遅延性を優先した配送を実現する．シミュレーション実験により，配送遅延の主要因子である転送経路長について，低負荷時には既存手法と比べ約 60% 以上，高負荷時においても約 20% 以上，短縮可能であることが明らかとなった．

1　はじめに

プログラム同士がデータをやり取りする通信モデルの一種として，pub/sub メッセージングがある．これは，送受信者の間に broker と呼ばれる中継サーバを配し，broker がデータ内容に応じて受信者を動的に決定して，データを転送する仕組みである．送受信者は通信相手を直接意識する必要が無いため，データの生成・消費に専念できるという高い疎結合性を有している [1]．加えて，broker から受信者へはプッシュ通信を行うため，リアルタイム性にも優れる．

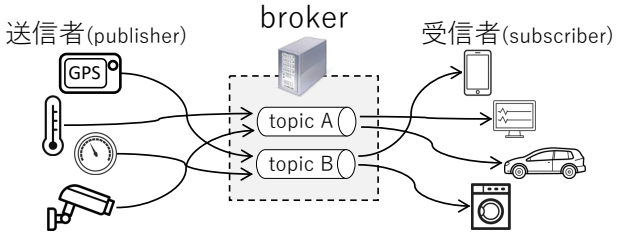

図 1　トピックベース pub/sub

本研究では，pub/sub メッセージングの中でも広く実用に供されている類型であるトピックベース pub/sub に着目する．トピックベース pub/sub では，送信者（publisher）と受信者（subscriber）がトピックと呼ばれる論理的なチャネルを介してやり取りを行う（図 1）．subscriber は broker に対して関心のあるトピックを事前申告（subscribe）しておくことで，当該トピックに対して発行（publish）されたメッセージを受信する．

前述の特性から，トピックベース pub/sub は非力なデバイスや送受信関係が流動的な状況に適する．このため，近年では特に IoT への応用で注目されており，IoT 向けの各種プラットフォームにおいて MQTT プロトコル [2] 等の採用が進んでいる他 [3]，Industry 4.0 の推奨規格である OPC UA においても対応が発表されている [4]．

多数の IoT デバイスから成る大規模システムへの適用を考えた場合，broker への負荷集中によるサービス停止や QoS 低下が問題となることから，複数 broker へのスケールアウトにより負荷分散等を実現する手法が提案されている [5, 6, 7, 8]．これら手法では，broker 群によりオーバレイネットワークを形成し，自律分散的な協調動作による broker の分散化を実現している．

* 東京工業大学　情報理工学院, banno@computer.org
† 東京工業大学　情報理工学院
This is an unrefereed paper.

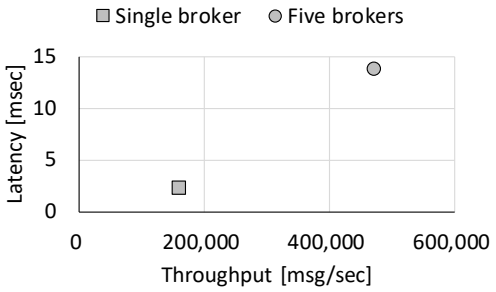

図 2　スケールアウトによる遅延増大

しかしながら，これらの既存手法では，スケールアウトにより高いスループットが得られる一方で，分散処理のオーバヘッドとして配送遅延が増大するという問題がある．複数 MQTT broker の連携を可能とする機構 ILDM [9] を用いた我々の予備的検証では，図 2 に示すスループットと配送遅延の実測値が得られた[1]．5 台の broker を連携させることにより，単体 broker と比べ最大スループットが約 3 倍に向上した一方で，遅延時間は約 6 倍に増大しており，スケーラビリティと低遅延性がある種のトレードオフ関係にあることが示されている．

このような遅延時間の増大は，pub/sub メッセージング本来のリアルタイム性を損ない，適用範囲を狭める．ウェブコンテンツ表示における許容遅延の目安は 100 ミリ秒とされており [10]，E コマースにおいては 100 ミリ秒の遅延増が売上に大きく影響することも指摘されている [11]．IoT において機器制御等の高いリアルタイム性を要するサービスを構築する場合には，より厳しい遅延要件が想定される．図 2 は LAN 内における小規模な検証であり，実地ではより大きな遅延が想定される上に，十分なアプリケーション処理時間をも確保する必要があることを考えると，低遅延性の獲得は不可欠である．

本研究では，トピックベース pub/sub において，スケーラビリティと低遅延性の両立を可能とする適応的制御手法を提案する．前述の遅延時間増大の主要因として，broker 間のマルチホップ転送がある．負荷分散のためには broker 間で配信木を形成してデータをマルチホップ転送する必要があり，通信遅延や処理遅延の累積によって送受信者間の遅延時間増大が生じている．提案手法では，既存手法と同様に，多数の broker によりオーバレイネットワークを構築する．その上で，高負荷のトピック，すなわち多数の subscriber が集中しているトピックにおいては，関連する broker 群で配送木を形成して publish メッセージをマルチホップ転送することで負荷分散を図る．一方，低負荷のトピックについては，関連する broker 群にシングルホップのオーバレイネットワークを局所的に形成させ，低遅延性を優先した配送を実現する．

以降では，まず 2 章にてトピックベース pub/sub に関する既存のスケールアウト手法を概説する．続いて 3 章にて提案手法を説明し，4 章でシミュレーション実験による評価について示す．最後に 5 章でまとめと今後の課題について述べる．

2　関連研究

本章では，トピックベース pub/sub において broker を多数のノードから構成する既存手法について述べる．

Scribe [5] は，分散ハッシュテーブル（DHT）の一種である Pastry [12] を用いたマルチキャスト手法である．Scribe では，Pastry ネットワーク上にマルチキャストグループ毎の配送木を形成する．DHT では，あるキーを担当するノードが一意に定まることから，グループ名をキーとした際の担当ノードを，当該グループに関する配送木の根ノードとして用いる．グループへの参加ノードは，グループ名をキーとして Pastry ネットワーク上で探索を行う．探索経路上の各ノードは，探索クエリの送信元ノードを，当該グループに関する配送木の子ノードとして保持する．これにより，グループ毎に配送木が形成され，根ノード宛にメッセージを送ることで，参加ノードへのマルチキャストが可能となる．Scribe において，トピックをグループとして扱うことで，トピックベース pub/sub への応用が可能である．

Bayeux [6] は，Pastry と同じく DHT の一種である Tapestry [13] を用いた手法であり，DHT における探索経路を用いて配送木を形成するという点で Scribe と類似している．ただし，Scribe ではグループ参加者から根ノードに向かう経路が配送木として利用されるのに対し，Bayeux においては根ノードからグループ参加者へ向かう経路が用いられるという点が異なる．Scribe と比較し，根ノードにおいて全参加ノードの情報を保持する必要がある等のデメリットがある．

CAN-MC [7] も，DHT アルゴリズムをベースとした手法であり，CAN [14] を用いている．Scribe や Bayeux とは異なり，CAN-MC では，2 種類の CAN ネットワー

[1] 簡易的なトポロジで broker 間のマルチホップ転送を模擬して測定．なお，スループットや遅延は測定環境と実装品質に大きな影響を受けるため，あくまでも傾向を示す参考情報であることに留意されたい．

クを複数組み合わせてマルチキャストを実現している．すなわち，グループ探索に用いる全域 CAN ネットワークと，グループ内フラッディングに用いる小規模 CAN ネットワークである．マルチキャストトラフィックの発生がグループメンバーのみに限定されるといった利点がある一方，Scribe や Bayeux と比較しオーバレイネットワークの構築及び維持のコストが高いという短所を有している．

2.1 SG-TBPS

これら DHT ベースの既存手法は高いスケーラビリティを備えているものの，トピックベース pub/sub への適用において，いくつかの観点で課題を有している．まず，Scribe や Bayeux の場合，publisher から各 subscriber までの配送経路長が $O(\log N)$ となる．ここで，N は全ノード数である．subscriber 数が極めて少ない場合，例えば 1 の場合であっても，ノード数 N に依存した経路長を要することとなり，ネットワーク資源の浪費及び配送遅延の増大を招く．また，いずれの手法においても，subscriber の不在時に publish メッセージが転送され続ける問題がある．

これらの問題点は，特に IoT における価値密度の低いデータを想定した場合，顕著となる．IoT データは，特定の目的を伴わずに大量に生成され，一部のデータのみが多大な価値を有する傾向があるとされており，こうした特性は “exhaust data” とも呼ばれている [15]．従って，価値の低いデータ，すなわち subscriber が少数またはゼロであるようなトピックが多くを占めることが想定され，そのような状況下における非効率性は大きな問題となる．

こうした問題に対し，構造化オーバレイネットワークの一種である Skip Graph [16] を用いた Topib-Based Pub/Sub 手法（以降，SG-TBPS と呼ぶ）が提案されている [8]．SG-TBPS では，広域ネットワークのエッジに多数の broker を配置し，それらが Skip Graph のトポロジを形成する．この時，同一トピックの publisher 同士および同一トピックの subscriber 同士が連結なサブグラフを構成し，またそれらサブグラフ同士も連結であるようにトポロジを作る（この条件を満たす性質を強 relay-free 性と呼ぶ）．これにより，

- 経路長がトピック参加ノード数 M に対し $O(\log M)$ となること
- publisher 群が subscriber の不在を検知し publish を停止可能であること

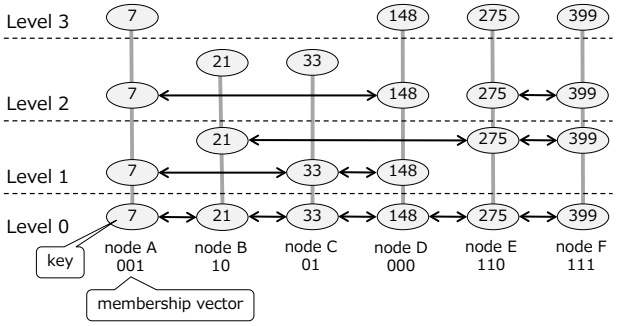

図 3　Skip Graph トポロジの例

を実現している．

3　提案手法

1 章で述べたように， 2 章の各既存手法は，スケールアウトにより高いスループットが得られる一方で，配送遅延が増大する問題がある．本章では，SG-TBPS をベースとして，スケーラビリティと低遅延性の両立を可能とする適応的制御手法について述べる．提案手法では，SG-TBPS において，経路表構築に FRT [17] の概念を適用することで，トピックの規模（参加ノード数）に追従したマルチホップとシングルホップの自律的切り替えを実現する．以降では，まず，要素技術となる Skip Graph，SG-TBPS，及び FRT について説明した後，提案手法の詳細を述べる．

3.1 Skip Graph

Skip Graph [16] は，構造化オーバレイネットワークの一種であり，範囲探索が可能という特徴を持つ．各ノードはキーとメンバーシップベクタと呼ばれるランダムな文字列を持つ．本稿では，メンバーシップベクタを構成する文字種の集合は $\Sigma = 0, 1$ であるものとする．Skip Graph では，メンバーシップベクタを用いて，Skip List [18] を多重化したトポロジを形成する．図 3 に例を示す．階層構造となっており，レベル 0 はキー順にソートされたノードの双方向リストとなっている．レベル i では，メンバーシップベクタの接頭 i 桁が一致するノード同士が双方向リストを形成する．キーの探索時は，探索対象のキーに最も近づくことができるレベルのリンクを選んでクエリを転送する．レベル 0 では全ノードが連なっているため，任意のノードを宛先として短い経路長で探索が可能である．ノード数が N の場合，探索の経路長は $O(\log N)$ となる．

Skip Graph ではノードがキー順にソートされて連なっているため，キーの範囲を指定した探索が可能

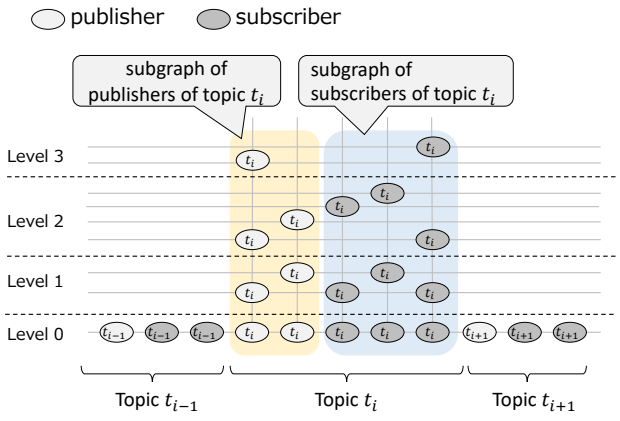

図 4　SG-TBPS のトポロジ

である．範囲探索クエリのルーティングについては，SFB [19] や MRF [20] 等の手法が知られている．

3.2　SG-TBPS

2.1 節で述べたように，SG-TBPS は Skip Graph を用いたトピックベース pub/sub 手法であり，広域ネットワークのエッジに配置した多数の broker がオーバレイネットワークを形成する．各ノードは，トピック名とノード種別（publisher または subscriber）をキーとした仮想ノードを保持し，Skip Graph のアルゴリズムにより図 4 に示すようなトポロジを形成する．ここで，Skip Graph ではノードがキー順でソートされることから，同一トピックの publisher 同士および同一トピックの subscriber 同士は連結なサブグラフを構成し，またそれらサブグラフ同士も連結となっている．これにより，トピック t_i について publisher と subscriber が存在する場合，subscriber のサブグラフとレベル 0 において隣接する publisher が唯一つ定まることが担保される．この publisher ノードは rendezvous point と呼ばれ，SG-TBPS では rendezvous point が subscriber の存在/不在を検出して同一トピックの publisher 群に対して通知をマルチキャストすることで，subscriber 不在トピックにおける publish の停止を可能としている．また，同一トピックの参加ノードによるサブグラフが連結であることから，経路長はトピック参加ノード数 M に対し $O(\log M)$ となり，全ノード数 N には依存しない特性が得られる．

3.3　Flexible Routing Tables

Flexible Routing Tables（FRT）[17] は，構造化オーバレイネットワークにおいて経路表を柔軟に構築するための概念である．一般的な構造化オーバレイネットワーク（ [21, 12, 13] 等）では，経路表構築に際し，一定の条件を満たすノードを探索して経路表エントリとして追加する．これに対し，FRT では，以下の 2 ステップにより経路表構築を行う．

1. entry learning: 経路表にエントリを無条件に追加する
2. entry filtering: 所与の経路表サイズ L を超えた時，経路表エントリをひとつ選択して削除する

entry learning は，例えばランダムな宛先に向けて探索クエリを発行しても良いし（能動的学習），他ノードからのクエリを中継する際に得られるノード情報を用いても良い（受動的学習）．entry filtering では，sticky entry と呼ばれる必須保持エントリを除外したエントリ群から，削除するエントリをひとつ選択する．この時，経路表間の順序関係 $\leq_{FRT}$ をあらかじめ定義し，エントリ削除によって構成し得る全経路表パターンの中で最大のものを選び，削除エントリを決定する．$\leq_{FRT}$ は経路表間の優劣を表すものであり，例えば $E \leq_{FRT} F$ の時，経路表 F は経路表 E よりも優良な経路表であることを意味する．用途に応じて $\leq_{FRT}$ を適切に定義することで，様々な特性を持つ構造化オーバレイネットワークを柔軟に設計することが可能である．

FRT の大きな利点のひとつとして，シングルホップ動作とマルチホップ動作を一貫して扱う構造化オーバレイネットワークを実現できる点がある．一般的な構造化オーバレイネットワークでは，経路表に追加するエントリの条件が定められているため，ノード数が少ない場合であっても，条件を満たさないノードは経路表に追加されず，結果としてマルチホップ動作を強いられる．一方，FRT に基づく経路表の場合，経路表サイズ L 以下のノード数の場合は経路表に全ノードを載せることが可能となり，シングルホップを実現できる．

3.4　負荷分散性と低遅延性の両立

提案手法では，SG-TBPS における経路表構築に FRT の概念を導入する．具体的には，同一トピックの subscriber ノードを含む経路表をより優良なものとして，経路表間の順序関係 $\leq_{FRT}$ を定義する．これにより，subscriber 数が経路表サイズ L よりも十分に小さいトピックにおいては，関連する broker 群がフルメッシュのサブグラフを形成することが期待できる．一方，subscriber 数が L よりも大きいトピックについては，従来の SG-TBPS と同様のマルチホップ転送が為される．これにより，低負荷時には低遅延性を，高負荷時には負荷

分散性を優先するトポロジへと自律的に切り替わることとなり，負荷分散性と低遅延性を両立することができる．

順序関係 $\leq_{FRT}$ については，FRT-Skip Graph [22] における定義をベースとし，同一トピックの subscriber ノードを優先的に保持するよう拡張する．具体的には，以下の各ポリシーを記載順の優先度で満たす経路表を優良とみなす．なお，Skip Graph のレベル 0 隣接ノードに相当するエントリは sticky entry とする．

1. Skip Graph の各レベル隣接ノードに相当するノードをエントリとして保持
2. 同一トピックの subscriber ノードをエントリとして保持
3. 各レベルで同数のエントリを保持
4. 高レベルのノードをエントリとして保持
5. 自ノードに近いキーを持つノードをエントリとして保持

上記を満たす順序関係 $\leq_{FRT}$ を定義するため，まず，幾つかの記法を定義する．

$l_{max}(E)$ 経路表 E における最大レベル

$EMP(E)$ 経路表 E において，$l_{max}(E)$ 個のレベルのうち，エントリ数が 0 であるレベルの集合．$EMP(E) = \{l_i \mid |E_{l_i}| = 0\}$ である．

$SUB(E)$ 経路表 E において，自ノードと同一トピックの subscriber エントリの集合

$MAX(E)$ 経路表 E において，最もエントリ数が多いレベルの集合

$SUCC(E)$ 経路表 E のうち，右方向のエントリ集合

$PRED(E)$ 経路表 E のうち，左方向のエントリ集合

$KEY(E)$ 経路表 E のエントリをキー順にソートしたエントリ列

これらを用いて，以下の命題を定義する．

$$
\begin{aligned}
B_1 \Leftrightarrow\ & l_{max}(E) < l_{max}(F) \\
& \vee (l_{max}(E) = l_{max}(F) \\
& \quad \wedge |EMP(E)| > |EMP(F)|) \\
B_2 \Leftrightarrow\ & l_{max}(E) = l_{max}(F) \\
& \wedge |EMP(E)| = |EMP(F)| \\
B_3 \Leftrightarrow\ & B_2 \wedge |SUB(E)| < |SUB(F)| \\
B_4 \Leftrightarrow\ & B_2 \wedge |SUB(E)| = |SUB(F)| \\
B_5 \Leftrightarrow\ & B_4 \wedge |MAX(E)| > |MAX(F)| \\
B_6 \Leftrightarrow\ & B_4 \wedge |MAX(E)| = |MAX(F)| \\
B_7 \Leftrightarrow\ & B_6 \wedge \\
& min(MAX(E)) < min(MAX(F)) \\
B_8 \Leftrightarrow\ & B_6 \wedge \\
& min(MAX(E)) = min(MAX(F)) \\
B_9 \Leftrightarrow\ & B_8 \wedge \\
& KEY(SUCC(E)) >_{dic} KEY(SUCC(F)) \\
B_{10} \Leftrightarrow\ & B_8 \wedge \\
& KEY(PRED(E)) >_{dic} KEY(PRED(F))
\end{aligned}
$$

ここで，$>_{dic}$ は辞書式順序であり，

$$\{a_i\} <_{dic} \{b_i\} \Leftrightarrow a_k < b_k \quad (k = min\{i \mid a_i \neq b_i\})$$

である．上記命題を用いて，提案手法における経路表 E と F 間の順序関係は以下のように定義される．

$$E \leq_{FRT} F \Leftrightarrow B_1 \vee B_3 \vee B_5 \vee B_7 \vee B_9 \vee B_{10}$$

3.5 提案手法の詳細動作

提案手法では，経路表の維持管理のために表 1 に示す各種クエリを用いる．なお，送信元情報及び転送元情報は，当該ノードの IP アドレスやメンバーシップベクタ，キー等の情報を含むものとする．以下，本節では，これらクエリを用いた各ノードの動作詳細について述べる．SG-TBPS と同様の動作となる点については説明を省いていることに注意されたい．また，以下では，レベル i 隣接ノードとは Skip Graph におけるレベル i の隣接ノードに相当するノードを指すものとする．

3.5.1 範囲探索

提案手法において，後述する publish メッセージの配送やノード離脱時の通知には，SFB [19] による範囲探索を用いる．ただし，提案手法の経路表では，FRT により各レベルに複数の隣接ノードを持ち得ることから，低レベルの隣接ノードが高レベルの隣接ノードよりも遠方に位置する場合がある．通常の SFB における，高レベルから低レベルへと順次対象範囲を分割していくプロトコルを適用できないため，提案手法においては，キーが遠方の隣接ノードから順に参照して部分範囲の委譲を行う．

3.5.2 ノード追加時

あるトピックに対し新たなノードが subscriber として追加される場合，当該ノードは，まず自身の挿入位置を探索し，レベル 0 の両隣接ノードを sticky entry として経路表に追加する．次に，左右それぞれのレベル 0 隣接ノードに対し，探索レベルを 0 に設定した REXP クエリを発行する．これにより，Skip Graph の経路表に相当するエントリの収集を図る．

表 1　経路表管理に用いられるクエリ

クエリ名称	説明	運搬する情報
REXP	定型的なエントリ探索（Regular EXPloration）のクエリ	送信元情報，転送元情報，既存エントリリスト，探索レベル
EEXP	臨時的なエントリ探索（Extra EXPloration）のクエリ	送信元情報，転送元情報，既存エントリリスト，不足エントリ数
PUB	publish メッセージの発行クエリ	送信元情報，トピック，publish データ，転送元情報
UNSUB	ノードのトピックからの離脱（UNSUBscribe）を通知するクエリ	送信元情報，転送元情報
NEW	新たな経路表エントリ候補を通知するクエリ	送信元情報

一定時間経過後，経路表のエントリ数が経路表サイズ L 未満の場合，同一トピックの subscriber のサブグラフを宛先とした範囲探索により EEXP クエリを発行する．さらに一定時間が経過した後，エントリ数が L 未満の場合には，ランダムな宛先に対する探索として EEXP クエリを発行することを繰り返す．

3.5.3　ノード離脱時

あるトピックの subscriber ノードが離脱する場合，当該ノードは，そのトピックの publisher のサブグラフを宛先とした範囲探索により UNSUB クエリを発行する．これは，publisher ノードにおいて，無効となる経路表エントリを削除するために行われる．FRT の性質上，提案手法の経路表構築は非対称に行われるため，各ノードは自身を経路表に持つノードを把握しない．しかしながら，後述するように publish メッセージの配送には SFB [19] を用いており，ノード離脱をタイムアウトにより判断するとリアルタイム性が損なわれることから，UNSUB クエリを用いて publisher ノードへの離脱通知を行っている．

3.5.4　publish 時

範囲探索により PUB クエリを subscriber ノードへ配送する．この時，経路表のエントリ数が経路表サイズ L 未満の場合は，EEXP クエリをあわせて発行する．これにより，ノード離脱により経路表エントリが減少した際，同一トピックの subscriber ノードを新たなエントリとして迅速に収集することを可能としている．

3.5.5　定期的処理

各ノードは一定時間毎に，左右それぞれのレベル 0 隣接ノードに対し，探索レベルを 0 に設定した REXP クエリを発行する．これにより，Skip Graph の経路表に相当するエントリの更新を図る．

3.5.6　クエリ受信時

REXP クエリを受信した場合，既存エントリリストに自身が含まれていなければ，送信元ノードに対し NEW クエリを発行する．受信した REXP クエリの探索レベルが i の時，自身が送信元ノードのレベル $i+1$ 隣接ノードに該当する場合は，探索レベルをインクリメントし，レベル $i+1$ の隣接ノードに REXP クエリを転送する．自身が送信元ノードのレベル $i+1$ 隣接ノードに該当しない場合は，レベル i の隣接ノードに転送する．該当する転送先が存在しない場合は，処理を終了する．

EEXP クエリを受信した場合，既存エントリリストに自身が含まれていなければ，送信元ノードに対し NEW クエリを発行する．不足エントリ数をデクリメントし，その結果が 1 以上であれば，次の転送先へと EEXP クエリを転送する．

UNSUB クエリを受信した場合，送信元ノードを経路表から削除する．

いずれのクエリにおいても，受信時，送信元ノード及び転送元ノードが自身の経路表に含まれていない場合は，エントリとして追加する．ただし，UNSUB クエリにおける送信元ノードは追加しない．

4　評価

提案手法について，シミュレーションによる評価を行った．シミュレーションプログラムは Java 言語により実装した．

物理ノード数を $10,000$ とし，各ノードが平均 5 つのトピックに publisher または subscriber として参加する構成とした．あるトピックについて，publisher 数を 1 とし，subscriber 数を 10 から 5120 まで変化させながら，平均経路長及び最大経路長を測定した．測定は 3 回行い，平均値を算出した．

比較対象として，2 章で述べた Scribe 及び SG-TBPS

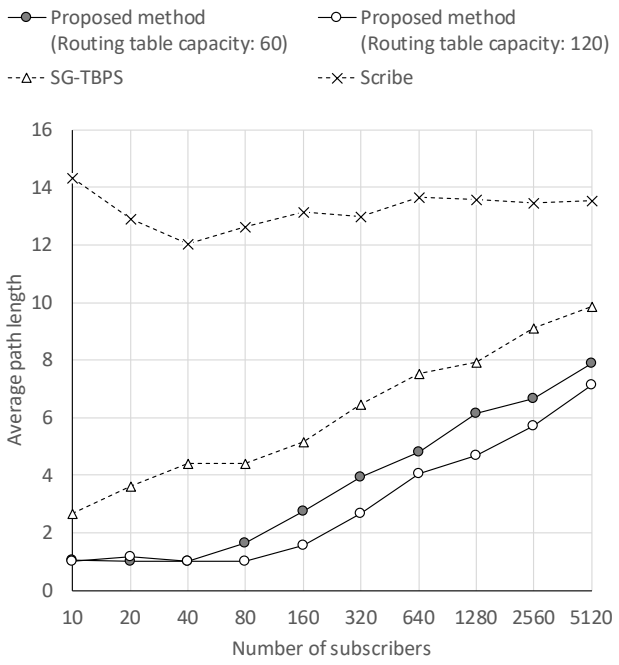

図 5　平均経路長

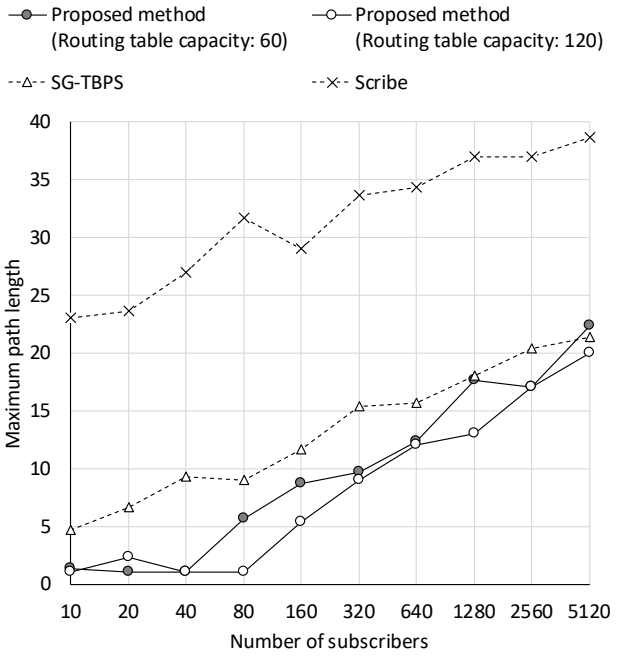

図 6　最大経路長

を用いた．提案手法における経路表サイズ L の値としては，60 及び 120 の 2 パターンを用いた[2]．

図 5 は平均経路長の測定結果である．提案手法の場合，subscriber 数が少ない場合には経路長が 1 となっていることがわかる．経路表サイズ 120 のパターンでは，subscriber 数 80 まで，シングルホップによる配送を実現できている．経路表サイズ 60 における SG-TBPS に対する平均経路長の削減率を算出したところ，subscriber 数 80 以下の低負荷時には約 63% から 77%，subscriber 数 160 以上の高負荷時には約 22% から 47% であった．

また，図 6 は最大経路長を示している．SG-TBPS では，平均経路長の概ね倍程度となっており，Scribe においても平均経路長よりも 1.5 倍から 2.5 倍程度大きな値となっている．一方，提案手法の場合，subscriber 数が少ない場合には最大経路長も 1 となっている．

4.1　遅延時間に関する考察

提案手法の効果を考察するために，簡易的なモデル化を用いて配送遅延の比較を行った．ここでは，publish メッセージの配送において，各ノードの内部で固定的に必要となる処理遅延を 1 ミリ秒，ノード間の通信遅延を 10 ミリ秒，マルチホップ時の子ノード数を 10，子ノード毎に要する処理遅延を 0.1 ミリ秒として，前述の平均経路長から配送遅延を算出した[3]．

図 7 に結果を示す．なお，参考情報として単体 broker のケース，即ち subscriber 数がそのまま子ノード数となるケースについても推定値をプロットしている．

グラフからわかるように，Scribe 及び SG-TBPS の場合，subscriber 数が多い状況下では単体 broker と比べ遅延を抑えられているものの，subscriber 数が少ない状況下では単体 broker に劣る遅延性能となっている． 1 章にて述べたように，IoT データにおける価値密度の低さを考慮すると，subscriber 数が少ない状況下が支配的となるケースも考えられることから，こうした性能特性は実用上望ましくない．一方，提案手法では，subscriber 数が少ない状況下では単体 broker に伍する低遅延性を持ち，subscriber 数が多い時にも SG-TBPS と同様の傾向で遅延を抑えることができていることがわかる．

[2] 予備実験として SG-TBPS における仮想ノード毎の経路表エントリ数を調査したところ，最大 58 であった．Skip Graph の経路表エントリ数は乱択アルゴリズムに基づき定まることから，各ノードとしては最悪値を許容することが求められる．このため，SG-TBPS と同程度の許容経路表サイズとして 60 を用い，またより大きな経路表を許容可能であるパターンとして 120 を用いることとした．

[3] 実際の各処理遅延は，実装方式，例えばノード間通信を同期的に行うか非同期的に行うか等によって大きく異なってくる．また，マルチホップ時の子ノード数も，いずれの手法においても $O(\log N)$ に抑えられるものの，実際にはノード毎に異なり，本文に記載のとおり簡易的なモデル化であることに注意を要する．ここでは，大局的な傾向を示す情報として記載しており，より詳細な検証は今後実施予定である．

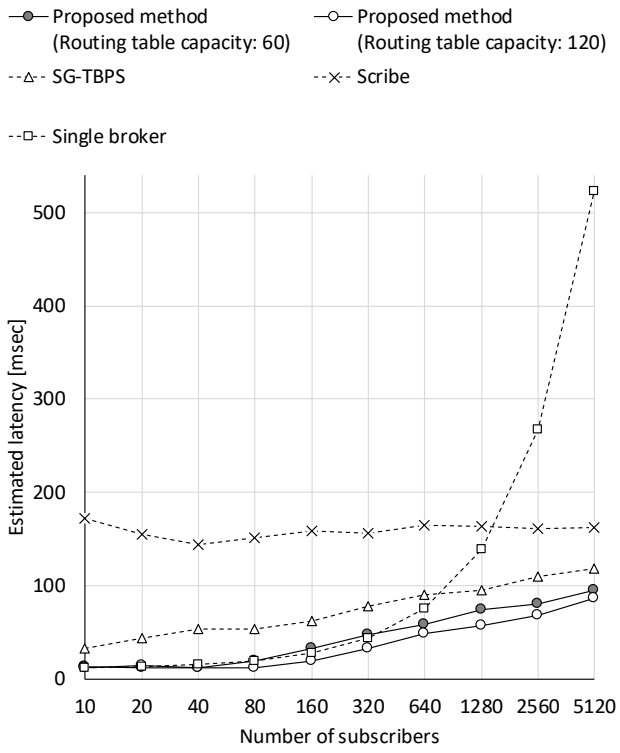

図 7　簡易モデルによる推定遅延時間

5　おわりに

本稿では，トピックベース pub/sub において，スケーラビリティと低遅延性の両立を可能とする適応的制御手法を提案した．SG-TBPS に FRT の概念を導入することで，subscriber 数の多寡に応じた broker 間のシングルホップ動作とマルチホップ動作の自律的な切り替えを実現した．シミュレーション実験により，既存手法と比べ経路長を短縮可能であること，また，subscriber 数が少ない状況下では経路長が 1 となることを示した．

今後の課題としては，ノードの出入りを含めたより実際的な実験設定による評価等を行う予定である．

謝辞

本研究は JSPS 科研費 19K20253 の助成を受けたものです．本研究は東京工業大学 末松基金の支援を受けたものです．本研究は（公財）セコム科学技術振興財団 一般研究助成の支援を受けたものです．

参考文献

[1] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.M. Kermarrec, “The Many Faces of Publish/Subscribe,” ACM Computing Surveys, vol.35, no.2, pp.114–131, 2003.
[2] MQTT. https://mqtt.org/ (accessed 2019-06-20).
[3] Amazon Web Services, “Designing MQTT Topics for AWS IoT Core,” May 2019.
[4] OPC Foundation, “OPC Foundation announces OPC UA PubSub release as important extension of OPC UA communication platform.” https://opcfoundation.org/news/press-releases/opc-foundation-announces-opc-ua-pubsub-release-important-extension-opc-ua-communication-platform/ (accessed 2019-06-20), March 2018.
[5] M. Castro, P. Druschel, A.M. Kermarrec, and A. Rowstron, “Scribe: A large-scale and decentralized application-level multicast infrastructure,” IEEE Journal on Selected Areas in communications, vol.20, no.8, pp.1489–1499, December 2002.
[6] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J.D. Kubiatowicz, “Bayeux : An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination,” Proc. International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp.11–20, June 2001.
[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Application-Level Multicast using Content-Addressable Networks,” Proc. International COST264 Workshop on Networked Group Communication, pp.14–29, November 2001.
[8] R. Banno, S. Takeuchi, M. Takemoto, T. Kawano, T. Kambayashi, and M. Matsuo, “Designing Overlay Networks for Handling Exhaust Data in a Distributed Topic-based Pub/Sub Architecture,” Journal of Information Processing, vol.23, no.2, pp.105–116, 2015.
[9] R. Banno, J. Sun, M. Fujita, S. Takeuchi, and K. Shudo, “Dissemination of edge-heavy data on heterogeneous MQTT brokers,” Proc. IEEE International Conference on Cloud Networking (CloudNet), pp.1–7, September 2017.
[10] Google, “Measure Performance with the RAIL Model.” https://developers.google.com/web/fundamentals/performance/rail/ (accessed 2019-06-20).
[11] R. Kohavi and R. Longbotham, “Online Experiments: Lessons Learned,” IEEE Computer, vol.40, no.9, pp.103–105, September 2007.
[12] A. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems,” Proc. IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, pp.329–350, November 2001.
[13] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz, “Tapestry: A resilient global-scale overlay for service deployment,” IEEE Journal on Selected Areas in Communications, vol.22, no.1, pp.41–53, January 2004.
[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable Content-Addressable Network,” ACM SIGCOMM Computer Communication Review, vol.31, no.4, pp.161–172, October 2001.
[15] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A.H. Byers, Big data: The next frontier for innovation, competition, and productivity, McKinsey Global Institute, 2011.
[16] J. Aspnes and G. Shah, “Skip Graphs,” ACM Transactions on Algorithms (TALG), vol.3, no.4, pp.37:1–37:25, November 2007.
[17] H. Nagao and K. Shudo, “Flexible Routing Tables: Designing Routing Algorithms for Overlays Based on a Total Order on a Routing Table Set,” Proc. IEEE International Conference on Peer-to-Peer Computing, pp.72–81, August 2011.
[18] W. Pugh, “Skip Lists : A Probabilistic Alternative to Balanced Trees,” Communications of the ACM, vol.33, no.6, pp.668–676, June 1990.
[19] R. Banno, T. Fujino, S. Takeuchi, and M. Takemoto, “SFB: A Scalable Method for Handling Range Queries on Skip Graphs,” IEICE Communications Express, vol.4, no.1, pp.14–19, January 2015.
[20] Y. Konishi, M. Yoshida, S. Takeuchi, Y. Teranishi, K. Harumoto, and S. Shimojo, “An Extension of Skip Graph to Store Multiple Keys on Single Node,” Journal of Information Processing Society of Japan (in Japanese), vol.49, no.9, pp.3223–3233, September 2008.
[21] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications,” SIGCOMM Computer Communication Review, vol.31, no.4, pp.149–160, October 2001.
[22] M. Hojo, R. Banno, and K. Shudo, “FRT-Skip Graph: A Skip Graph-style structured overlay based on Flexible Routing Tables,” Proc. IEEE Symposium on Computers and Communication (ISCC), pp.657–662, June 2016.