情報処理学会論文誌 Vol.61 No.2 326–338 (Feb. 2020)

SDNを活用するPub/Sub基盤における オーバレイネットワーク管理方式の改善手法

盛房 亮輔[1] 坂野 遼平[2] 安倍 広多[3] 寺西 裕一[4] 石原 真太郎[1] 秋山 豊和[1,a)]

受付日 2019年5月7日, 採録日 2019年11月7日

概要：トピックベースのPub/Sub通信モデルにおける配信基盤として，SDN技術を活用したPub/Sub基盤（SAPS）が提案されている．SAPSでは，2つの配送方式Application Layer Multicast（ALM）とOpenFlow Multicast（OFM）を併用し，前者のスケーラビリティと後者の配送効率の良さを相互補完的に活用することで，複数Broker間の効率的なメッセージ配送を実現する．SAPSにはALMからOFMへの切替方式として，一斉切替方式（EITHER方式）と逐次切替方式（BOTH方式）が存在している．前者は切替時間が短くOFMの配送効率の良さを享受しやすいがスケールしにくいという課題があり，後者はスケーラビリティに優れ大規模環境に適しているが，オーバレイ上で同じトピックをsubscribeしたメンバを2つのトピックに分けてグループ管理する必要があるため，メンバ不在トピックへのメッセージ転送やOFM切替の遅延時間が増大するという課題があった．そこで本論文では，BOTH方式においてメッセージ数や切替遅延を改善するオーバレイネットワーク管理手法を提案する．メッセージ数については，構造化オーバレイ上の強リレーフリー性を活用し，2トピック間でのメンバ不在トピックへの転送を防ぐ手法を提案した．また，切替遅延については，ノードのjoin/leaveをともなわず，ノードの内部情報を書き換えることで実質的な移行が完了するキー名書換え移行を提案した．実機を用いた検証により，これら提案手法によりメンバ不在トピックへのメッセージ転送がなくなり，かつ，OFM切替遅延を大幅に短縮できることを確認した．

キーワード：トピックベースPub/Sub，Application Layer Multicast，構造化オーバレイネットワーク

Improving Overlay Network Management in SDN Aware Pub/Sub Infrastructure

Ryosuke Morifusa[1] Ryohei Banno[2] Kota Abe[3] Yuichi Teranishi[4] Shintaro Ishihara[1] Toyokazu Akiyama[1,a)]

Received: May 7, 2019, Accepted: November 7, 2019

Abstract: SDN aware Pub/Sub infrastructure has been proposed as a delivery infrastructure in the topic-based Pub / Sub communication model. SAPS realizes efficient message delivery between multiple Brokers by using two delivery methods Application Layer Multicast (ALM) and OpenFlow multicast (OFM) complementarily in combination. SAPS has simultaneous switching method (EITHER method) and sequential switching method (BOTH method) as switching methods from ALM to OFM which establishes efficient delivery. While the former switches delivery method of all member nodes in a short period it has scalability issues. On the other hand, though the latter improves scalability in a large-scale environment, it is necessary to manage publishing/subscribing member groups for each delivery method in the overlay network. And current system design causes issues such as wasteful message transfers to an empty member topic and increase of delivery method switching period. In this paper, we propose a new overlay network management method to reduce the number of messages and the delay time of OFM switching in BOTH method. For the former, we exploit the property of strong relay freeness of structured overlay to prevent transfer to an empty member topic. For the latter, we propose key name rewriting transition in which substantial transition is completed by rewriting the internal information of the node without joining / leaveing of the node. As a result of experiments, it was confirmed that the proposed method eliminated the message transfer to an empty member topic and reduced the delay time of OFM switching drastically.

Keywords: topic based Pub/Sub, application layer multicast, structured overlay network

1. はじめに

SNS や IoT といったリアルタイムなメッセージ収集，分析，配送が要求されるアプリケーションでは，トピックベースの Pub/Sub 通信モデルが広く用いられている．また，IoT デバイスは計算資源が限られていることが多く，標準化された軽量な Pub/Sub プロトコルである MQTT [1] などが通信プロトコルとして利用されることが多い．MQTT をはじめとするトピックベースの Pub/Sub 通信では Publisher と Subscriber の間に存在する Broker がメッセージを中継することで非同期メッセージングを可能としている．しかしシステムの大規模化にともなう負荷分散や，地理的なスケーラビリティの観点から複数 Broker の分散配置を想定する場合，Broker 間通信ではスケーラビリティの向上と通信量の低減が求められるようになる．これに対し，著者らは Broker 間通信を効率化するための Pub/Sub 基盤として，SDN（Software Defined Network）技術を活用した Pub/Sub 基盤（以下 SAPS）[2] を提案している．

SAPS は Broker 間の配送方式として，2 つの配送方式「Application Layer Multicast（以下 ALM）」と「OpenFlow Multicast（以下 OFM）」を相互補完する形で併用し，効率的な配送を実現する（**図 1**）．ALM は IP マルチキャストの代わりにドメイン間サービスに適用できる [3]．オーバレイに join したノードは，ランダムで一意のキーが割り当てられ辞書順に並び，さらに同じトピックを subscribe するノードは，キーの接頭辞に同じトピック名を付与することで隣接するように挿入される．これによって，当該トピックに対応するキーの範囲に ALM でマルチキャストすることでリレーフリー [4] に publish できるようになっている．しかし ALM で物理的なトポロジを把握するには，下位層の情報を取得する仕組みが必要となるが，配送時に下位層のマルチキャストのようなネットワーク内でのメッセージ複製が実現できないため，配送経路が非効率になる場合がある．一方の OFM は，OpenFlow コントローラ（以下 OFC）が物理トポロジを把握したうえでネットワークを集中管理できる OpenFlow の特長を活かして，最適経路を用いた配信が可能である．OpenFlow はドメインごとに単一組織で運用し，OFC が API を利用者に提供することで，複数ドメインにまたがって OpenFlow ネットワークを制御することも可能である [5]．ところが OpenFlow スイッチ（以下 OFS）に登録可能なフローエントリ数に上限があり [6]，ノード数の増加に対するスケーラビリティに課題がある．そのため，SAPS では ALM と OFM を併用し，効率化が必要なトピックのみ OFM を活用している．

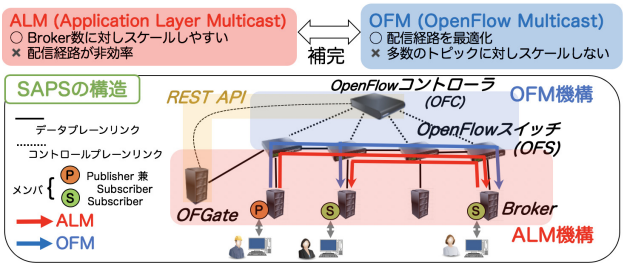

図 1 SAPS 概要図
Fig. 1 Overview of SAPS.

現状の SAPS には，オーバレイネットワークの管理方式に問題があり，無駄なメッセージ配送によって帯域を圧迫し，スケーラビリティへ悪影響を及ぼす可能性があるため，本研究では管理方式の改善手法を提案する．

本論文の貢献は以下のとおりである．

(1) オーバレイネットワーク上で，ALM/OFM それぞれの配送方式を利用するノードグループに対するメッセージ配信において，強リレーフリー特性を活用することで異なるグループ間での無駄なメッセージ送信が削減できることを確認した．
(2) オーバレイネットワーク上で，ALM/OFM それぞれの配送方式を利用するノードグループ間で，1 度に多数のグループメンバを移動する場合に，更新メッセージの伝搬方式を変更することで，移動時間を短縮できることを確認した．
(3) 上記の改良によって SAPS での配送方式の切替時間を大幅に削減できることを確認した．

以降，2 章では，既存の Pub/Sub 基盤の提案と SAPS について比較し，本研究の位置付けを明らかにする．3 章でより具体的な SAPS の仕組みとその課題について述べ，4 章では SAPS の従来の実装における 2 つの切替方式の比較について述べる．5 章では提案手法の概要ならびに実装について述べる．そして 6 章では従来手法と提案手法の比較評価について述べて，最後に 7 章でまとめと今後の課題を述べる．

2. 関連研究

これまでにシステムの大規模化にともなう負荷分散や，地理的なスケーラビリティの観点から複数 Broker の分散配置を行う分散 Pub/Sub 基盤が提案されている．文献 [7] では，オーバレイ上のキー値に物理的なネットワークの情報として階層的なクラスタ ID を付与することで，クラス

1 京都産業大学大学院先端情報学研究科
Division of Frontier Informatics, Graduate School of Kyoto Sangyo University, Kyoto 603–8555, Japan
2 東京工業大学情報理工学院
School of Computing, Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
3 大阪市立大学大学院工学研究科
Graduate School of Engineering, Osaka City University, Osaka 558–8585, Japan
4 情報通信研究機構（NICT）
National Institute of Information and Communications Technology, Koganei, Tokyo 184–8795, Japan
a) akiyama@cc.kyoto-su.ac.jp

タをまたがる非効率なメッセージ配信を低減する方式を提案している．文献 [8] では，複数の ZeroMQ インスタンスにおけるトピックの Subscribe 状況を Zookeeper により共有する方式で実装し，Publish/Subscribe 通信の特性について調査している．文献 [9] では，コントローラによって MQTT Broker の Bridging Table を集中管理することで，Broker 間連携を実現している．文献 [10] でも述べられているが，文献 [8] で活用されている Zookeeper は集中管理型でありスケーラビリティに限界がある．また文献 [9] のアプローチも同様にコントローラによる集中管理が想定されている．そのため，地理的に分散した環境においては，遠隔拠点で動作するインスタンスに依存することになり，遅延による性能低下や，拠点間通信障害によるローカル処理の停止といった課題をかかえている．また，集中管理部分がボトルネックとなり，Churn が発生する環境への対応が困難になる可能性がある．これに対し，文献 [7] は P2P ベースのため，状態更新時の他ノードへの依存は周辺ノードに限定される．また，文献 [7] では下位層も考慮した隣接関係を構築できるため，遠隔拠点への依存も低減できる．P2P ベースであるため，Churn への対応も容易になると考えられるが，文献 [7] で用いられていたオーバレイネットワークである SkipGraph に対し，本論文でも活用したより Churn 耐性の高い Suzaku [11] も開発されていることも考慮すると，文献 [8], [9] と比較して，広域分散環境での運用に適していると考えられる．一方で，文献 [7] では，下位層のネットワークにおいて距離が近いノードを階層化されたクラスタとすることで，クラスタ間の無駄な通信を削減しているが，クラスタ内やクラスタ外への通信で下位層の同一ネットワーク機器上を複数回通過する無駄な通信が発生する場合がある．また，メッセージの複製が Broker 上で行われるため，配送先 Broker が多くなると，Broker の利用可能帯域を消費してしまう．

これに対し PLEROMA [12] では，SDN を活用することで動的に下位層のネットワークを制御し，最適な配送経路で Pub/Sub 通信を実現している．しかし，ハードウェアの OpenFlow スイッチでは，TCAM のような高コストなデバイスを用いて高性能な配送性能を実現しているため，フローエントリ数の上限に制約がある [6]．フローエントリ数の一例としては，文献 [13] などに記載がある．

そのため，すべてのトピックを SDN で配送するのではなく，subscriber 数が多く，publish 頻度の高いトピックは SDN で配送し，他のトピックは P2P オーバレイネットワークで配信する方式として，筆者らの研究グループでは SAPS [2] を提案している．SAPS では，OpenFlow を用いた下位層による配送方式である OFM と，文献 [7] で提案した P2P オーバレイを用いた配送方式である ALM をハイブリッドに切り替えて活用する．しかし，詳細は後述するが，SAPS で提案している 2 つの配送方式の切替方式である EITHER 方式と BOTH 方式について，配送方式の管理に必要なメッセージ数については文献 [2] で比較していたが，切替時間の違いによる遅延の影響について調査できていなかった．本論文ではこの影響を調査し，そこでの課題の解決を目指す．以降ではまず SAPS の仕組みと課題について述べる．

3. SAPS の仕組みと課題

現状の SAPS では，トピックごとに利用する配送方式を ALM から配送効率の良い OFM へフローエントリが登録できる限り切り替えるという方針での運用が想定されており，その切替方式として一斉切替方式（EITHER 方式）と逐次切替方式（BOTH 方式）が存在する．OFC と通信を行う特殊な Broker である OFGate は，OFM のための配信木（OFM 木）を構築する際，OFM で配信するための特別な IP アドレスとポート番号の組（OFM アドレスと呼ぶ）を発行する．EITHER 方式では publish ごとに OFGate へ OFM アドレスの有無を問い合わせて，ALM か OFM のどちらか（= EITHER）で配送する（**図 2**）．このため EITHER 方式では，OFM 構築後に即座に移行が完了し切替効率が良くなる．一方で，4.3 節で詳しく述べるが，OFGate への Publish/Subscribe にともなう問合せ要求が後述の BOTH 方式と異なり，Broker 配下にある Publisher/Subscriber 数に比例して増大する点が課題となる．これに対して BOTH 方式は，publish ごとに OFGate へ OFM アドレスを問い合わせることはせず，つねに ALM および OFM の両方（= BOTH）で配送する．特徴は OFM 移行前のメンバ（以降トピックの Subscriber と Publisher を総称してメンバと呼ぶ）と，OFM 移行後のメンバをオーバレイ上のトピック「ALM トピック」と「OFM トピック」に分けてグループ管理し，ALM でのメッセージ配送や，OFM アドレスの周知，OFM アドレスのキャッシングなどに利用している点である．「OFM トピック」は，OFM 構築後に OFM 経由の publish（以下 OFM publish）を受信可能となったメンバが「ALM トピック」から移行したグループのことである（**図 3**）．OFM アドレスを知っているメンバが publish する場合，ALM に残っているメンバに対しては「ALM トピック」へのオーバレイ経由の publish

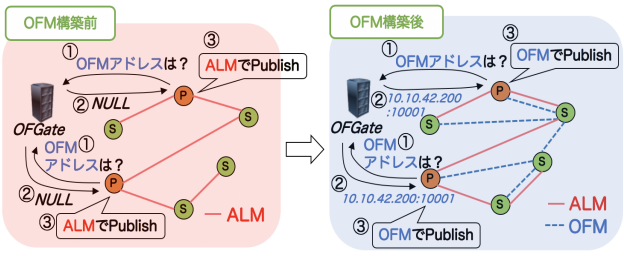

図 2 EITHER 方式での publish
Fig. 2 Publish with EITHER method.

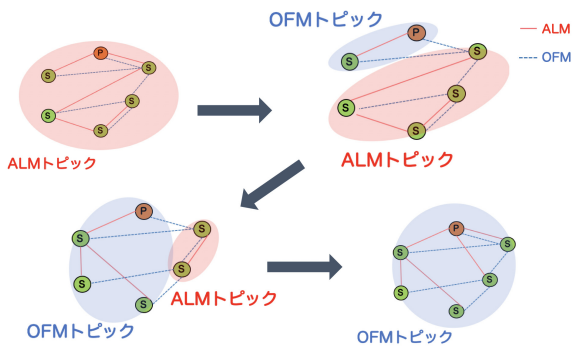

図 3 BOTH 方式におけるトピック移行の流れ
Fig. 3 Topic migration flow in BOTH method.

（以下 ALM publish）で配送され，OFM に移行したメンバに対しては OFM publish で配送される．OFM アドレスを知らないメンバは，同様に「ALM トピック」に対して ALM publish で配信するが，OFM には配送できないため，「OFM トピック」に対して OFM publish の代わりに ALM publish を行う．また，「OFM トピック」への publish の応答として OFM アドレスを取得する．このように BOTH 方式では，OFGate への問合せ以外にメンバ間で OFM アドレスを取得できるため，OFGate への問合せ負荷を軽減できる．EITHER 方式と BOTH 方式で OFGate に対する問合せメッセージ数については文献 [2] で比較されており，比較したすべてのケースにおいて BOTH 方式が優れていることが確認できている．しかし OFM 構築後に行われる ALM トピックから OFM トピックへの移行（以下トピック移行）は，Churn 回避のためノードごとにランダムな待ち時間後に実行される．なお，4 章で述べる計測では，最小値 500 ms，最大値 5000 ms の待ち時間を一様分布の乱数を用いて生成した．この遅延により，すべてのメンバのトピック移行が完了するまでの間，publish を実施するメンバは，ALM publish と OFM publish を二重配送する必要があるため，EITHER 方式と比較して切替効率が悪い．

これら両方式の違いが性能に与える影響の比較調査が行われていなかったため，本論文ではまず，BOTH 方式のスケーラビリティについて EITHER 方式と比較するために，publish 時の OFGate への問合せ頻度の違いによる「publish に必要なメッセージ数」として「下位層のスイッチ間の物理ホップ数の総計*1」（以下総ホップ数）の違いを調査した．調査結果の詳細は 3 章で後述するが，ALM から OFM へ移行する過程の publish に必要な物理ホップ数の推移を比較したところ，BOTH 方式ではメンバ不在トピックへの転送やトピック移行中に ALM と OFM で二重にメッセージ転送される影響により，計測中のほとんどの期間で EITHER 方式よりも多くのメッセージ転送が発生することが分かった．

そこで本論文では，構造化オーバレイ上の強リレーフリー性という性質に着目し，2 トピック間でお互いのトピックを監視してメンバ不在トピックの検知および転送を制御する手法を検討する．またメンバのトピック移行における Churn を回避するために生じていた遅延を短縮するために，ノードの join/leave をともなわずにメンバの実質的な移行が可能になるキー名書換え移行の検討を行う．

4. 既存実装での配送方式切替手法の比較

4.1 SAPS の実装

SAPS では ALM を実現するミドルウェアとして，構造化オーバレイネットワークとエージェント機構を持つ P2P Interactive Agent eXtensions（以下 PIAX）[14] を用いている．PIAX では，ID Transport（IDT）と Locator Transport（LT）が分離されていることから，Broker は ALM と OFM を異なるポートで待ち受け，同時に受信することが可能である．また下位層では，ALM で用いる通常の IP 伝送用のフローエントリと OFM 用のフローエントリを OFS のフローテーブルへ両方登録することが可能なため，ALM と OFM の配送を併用できる．SAPS では PIAX に実装されている Skip Graph 拡張 MKSG（Multi-Key SkipGraph）[15] を用いて構造化オーバレイを構築しており，ノード間のデータ構造として，DDLL [16] を用いている．OFM 木の構築機能は SDN 技術の 1 つである OpenFlow のコントローラである Ryu [17] を用いて構築されている．

OFM 木の構築は，該当トピックに関する閾値を設定し，定期的に行われるトピックの各メンバによるモニタリングで配信メッセージ数を監視し，閾値を超えると該当トピックのメンバの依頼で行われる．閾値としては，トピックのメンバ数や累計 publish 回数など，Subscriber 側で集計可能な情報を活用し，たとえば Publish 頻度の高いトピックとして 5 秒間に 3 回以上 Publish メッセージを受信した場合といったシンプルなものから，文献 [18] のように ALM や OFM の通信コストを考慮したもの，さらに，トピックへの出入りの頻度なども考慮したものなど，様々な設定方法が考えられる．現時点ではあらゆる場合で最適な設定方法は明らかにできていないが，設定する閾値は，OFM で使用可能なフローエントリ数に上限があることを考慮し，できるだけ使用頻度の高いトピックから切り替えられるようにする．ALM 機構と OFM 機構との連携は “OFGate” と呼ばれる特殊な Broker が担う．OFGate は REST API 経由で OFC に OFM 木の構築/更新を要求したり（**表 1**），OFM 木ごとの参加メンバや OFM 木を識別したりするための OFM アドレスを管理している．

*1 オーバレイ上における仮想ノード間のメッセージ転送を論理ホップ，下位層のスイッチ間でのメッセージ転送を物理ホップと呼ぶ．以降では，特に言及がない限り，「ホップ数」は送信されたすべてのメッセージの物理ホップ数の合計となる総ホップ数を指すものとする．

表 1 OFM 木構築/更新要求のために REST API で使用する URL
Table 1 REST API URLs for OFM tree construction/update requests.

要求内容	URL
OFM 構築	http://example.com/api/setupOFM
OFM 参加更新	http://example.com/api/addOFMMembers/10.10.42.200:10001
OFM 脱退更新	http://example.com/api/removeOFMMembers/10.10.42.200:10001

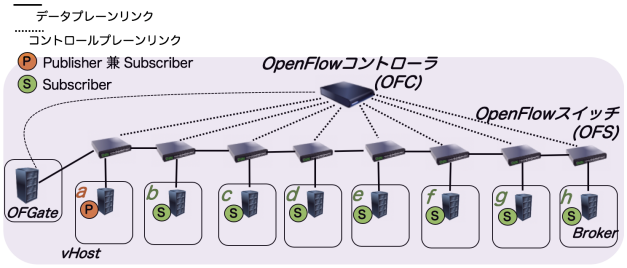

図 4 計測トポロジ
Fig. 4 The topology used in the measurement.

4.2 比較項目について

EITHER 方式は切替効率が良いが，大規模環境での運用を想定した場合，スケーラビリティの観点から BOTH 方式がより重要になる．そこで既存実装を用いて BOTH 方式のスケーラビリティについて EITHER 方式との比較調査を行う．

まず EITHER 方式で運用上ボトルネックとなりうる OFGate への問合せ負荷を比較しようとしたが，既存実装では Broker の転送負荷が OFGate での制御負荷と比較して高かったため，OFGate に負荷を集中させる以前に，問合せメッセージを送出する側の Broker が処理限界に達してしまった．そのため，問合せ負荷による比較については見送ることとした．なお Broker の配送性能に関して，本論文で検証に用いた PIAX に実装されている MKSG の性能は，PIAX に実装された Suzaku を用いた構造化オーバレイにおいて改善しており，別途配送性能の改善についても取り組んでいるが，本論文ではまずは負荷を用いた評価方法ではなく，両方式で OFGate への問合せに生じるメッセージ数として総ホップ数の差異で比較する．

4.3 計測環境

計測トポロジは，**図 4** に示すように OFS が 8 台接続されたライントポロジとし，OFS に Broker がそれぞれ接続されていると想定する．以下ではまずこのような計測構成を採用した理由について述べる．

文献 [2] では，スイッチの故障や各ポートの帯域の制約を考慮しない環境で，ツリートポロジとライントポロジを用いて，ALM と OFM の性能評価を行っている．ツリートポロジはライントポロジと比較してスイッチ間のホップ数を低減できるため，下位層を把握できないオーバレイネットワークの通信によって生じる無駄な配送が少なくなる．なお，故障や帯域の制約を考慮しない場合，データセンタで発生するコアや集約スイッチでのポート数の制約がなくなるため，一般的にデータセンタで利用されている Leaf-Spine などの構成は，ツリートポロジで同等な評価結果が得られると考えられる．文献 [2] では，ライントポロジとツリートポロジを比較し，結果としてライントポロジにおいて OFM がよりメッセージ数を低減できることを確認している．一方で，ライントポロジはオーバレイネットワークから見ると，無駄に物理スイッチを往復する可能性がある不利なトポロジとなる．本論文で行う評価はオーバレイネットワークでの制御トラフィックの評価であり，評価はオーバレイネットワークに不利な環境で，かつ，SAPS の利用価値が高い OFM にとって有利な環境において行うのが妥当であると考えられる．このような理由から以降の計測ではライントポロジを用いる．

次にネットワークの規模について，後述するようにソフトウェアスイッチを物理スイッチと想定して利用しているが，スイッチの各ポートの帯域制限を想定しておらず，評価に利用した PC で，十分に Pub/Sub メッセージが送受信できる規模で評価を行うことになるため，8 台での評価とした．より大規模な環境での評価は今後の課題とする．一方で，地理的な分散を考慮して SAPS の Broker を配置する場合，たとえばクラウドサービスの Availability Zone（AZ）ごとに Broker を配置する方法などが考えられる．現在の SAPS の実装では，同一 Broker 内で複数の Subscriber が同一トピックを Subscribe した場合，オーバレイネットワーク上に仮想ノードは追加されず，個々の Publisher/Subscriber の状態は Broker 内で管理されるため，本論文での BOTH 方式での Publish に関する評価内容は Broker 配下に同一トピックに対する多数の Publisher/Subscriber が配置された場合と，オーバレイネットワーク上の動作は同じになる．この場合，各 Broker が Publisher/Subscriber をどの程度ホストできるかが課題になるが，文献 [19] で示されているとおり，PIQT では一般的なデスクトップ PC 上の単体プロセスで 80 万メッセージ/秒のスループットでロスなし転送が行えることが確認できており，1 秒単位で Publish するようなアプリケーションにおいては，相応の PC を準備できれば，AZ の担当範囲内のクライアントをカバーできるのではないかと考えられる．Amazon Elastic Compute Cloud では文献 [20] に示されているように，アジアパシフィックで 7 拠点の AZ を提供しており，たとえば本論文で実施した実験を，このエリアをカバーする形での配置を想定した実験ととらえることもできると考えられる．文献 [7] で提案している階層化クラスタでは，アジア・パシフィックを 1 つのクラスタと定義することができるが，クラスタ内での通信，たとえばシドニーからソウルを経由してシンガポールに行くなど，非効率なパスが選択されてしまう可能性があり，SAPS により通信を効率化できる可能

表 2 両方式のフェーズごとの publish に必要なホップ数の違い
Table 2 The number of hops required to publish a message in each phase of both delivery methods.

切替方式	移行前	トピック移行中	移行後
EITHER 方式	$M_{ALM} + M_Q$	-	$M_{OFM} + M_Q$
BOTH 方式	$M_{ALMt} + M_{OFMt}$	$M_{ALMt} + M_{OFM}$	$M_{ALMt} + M_{OFM}$

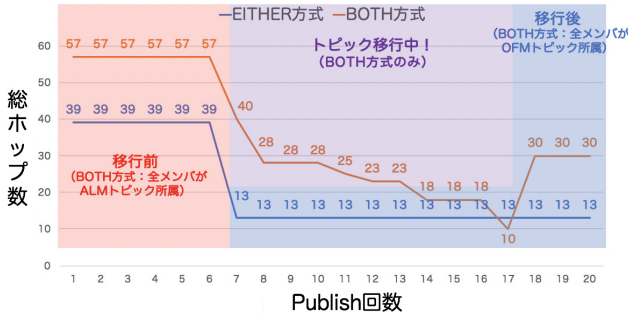

図 5 総ホップ数推移の比較結果の一例
Fig. 5 An experimental example of changes in total number of hops.

性はあると考えている．以上のようなことから，8 台の評価でも現実的なネットワークに適用可能な結論が得られると考えている．

図 4 の構成で，単一サーバ上に Mininet [21] を用いて仮想ネットワークを構築し，リソースを分離した各仮想ホスト上で Broker を起動している[*2]．

4.4 計測結果

4.3 節で述べた構成で，同じトピックを subscribe した a〜h のメンバに対して，a が 1 秒間隔で publish したときの，publish ごとの総ホップ数の推移を示す．

BOTH 方式では publish ごとに OFGate への問合せが発生することがないため，publish するために必要なホップ数は，トピック移行期間中以外は BOTH 方式が下回ると予想したが，BOTH 方式では 3.2 節で述べるとおりメンバ不在トピックへの転送が発生するため，逆に EITHER 方式を上回る総ホップ数になることが分かった．つまり，移行前後のフェーズごとの総ホップ数は**表 2** の式のようになる．ただし表中の M_{ALM} は ALM publish，M_{ALMt} と M_{OFMt} は BOTH 方式特有の ALM/OFM トピック宛の ALM publish，M_{OFM} は OFM publish，M_Q は OFGate への問合せメッセージにおける総ホップ数を表す．

計測結果の一例を**図 5** に示す．BOTH 方式の移行前の M_{OFMt} および移行後の M_{ALMt} へのメンバ不在トピックへの転送により，BOTH 方式のホップ数が EITHER 方式を上回ることが確認できる．ただし BOTH 方式では Churn 回避のため，メンバごとの移行開始時間は約 2 秒間隔となっていること，ならびに，この例では特に総ホップ数が多くなる場合として，b〜h，最後に a の順でトピック移行を実施していることから，トピック移行中のホップ数が比較的多く示されている．この点を考慮して，増加した総ホップ数を差し引いたとしても，BOTH 方式の総ホップ数が，EITHER 方式を下回ることはなく，BOTH 方式によるメッセージ数削減効果のメリットが得られないことが確認できた．

4.5 BOTH 方式のメッセージ配送における問題点

BOTH 方式のホップ数が増加した一番の問題点は，メンバ不在トピックへの転送である．ALM publish の実行にともないオーバレイ上で 1 度トピック宛の範囲検索が実行されると，たとえそのトピックが存在しなくてもその範囲にキーが一番近いノードまでメッセージが転送される．結果は当然空集合となり，無駄なメッセージ転送となるが，メンバが publish 前にトピックメンバの有無を把握していなければ，配送を制御することは難しい．またトピックが増加した際の課題として，同時多発的にトピック移行が起きることで移行がさらに遅延する可能性が考えられる．

5. オーバレイネットワーク管理方式の改善手法

本論文では，SAPS の BOTH 方式のメッセージ配送における課題を解決するため，オーバレイネットワーク管理方式の改善手法を検討する．また現在 SAPS プロジェクトでは，分散型 MQTT Broker として利用可能な PIQT（MQTT on PIAX）[22] への移植が進められていることから，本論文でも SAPS の移植に合わせて，PIQT 内部の PIAX に実装された Suzaku を改良する形で，オーバレイネットワーク管理方式の改良手法を実装する．Suzaku によって SkipGraph よりも構造が単純で高速なキー順序保存型オーバレイネットワークを構築できる[*3]．ノード間のデータ構造として，SkipGraph と同様に DDLL を用いている．

5.1 Subscriber 不在時のメッセージ転送防止手法

まず「メンバ不在トピックへの転送」に関しては，相手トピックメンバの有無を監視し，メンバの publish を制御するために「強リレーフリー性」を活用した Subscriber 不在時のメッセージ転送防止手法を提案する．

強リレーフリー性とは，ネットワークリソースの浪費を改善するために文献 [23] で提案された構造化オーバレイ上の特性で，V をノード集合，T をトピック集合とすると以下のように定義される．ノード $v \in V$ とトピック $t \in T$ を入力として，ブール値関数 Sub(v,t) と Pub(v,t) を定義する．Sub(v,t) = true の場合，ノード v はトピック t の

*2 計測に用いた機器やソフトウェアのバージョンについては付録を参照されたい．

*3 3.2 節で述べた BOTH 方式のメッセージ配送における課題は，Suzaku でも存在する．

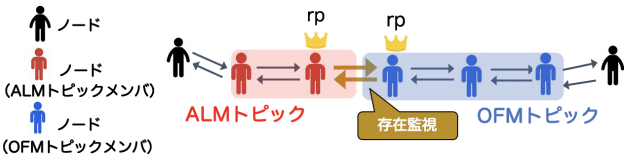

図 6 強リレーフリー性を活用した Subscriber 不在時のメッセージ転送防止手法

Fig. 6 A prevention method of unnecessary message transfer without subscribers utilizing strong relay-free characteristics.

subscriber であり，Pub(v,t) = true の場合，ノード v はトピック t の publisher である．以下の 3 つの条件すべてが ∀t∈T について満たされる場合，グラフ G は強リレーフリー性を満たす．

- ノード $\{ v \in V |$ Sub(v,t) = true $\}$ によるサブグラフ G_s が接続されている．
- ノード $\{ v \in V |$ Pub(v,t) = true $\}$ によるサブグラフ G_p が接続されている．
- G_s，G_p によるサブグラフが接続されている．

たとえば IoT のデータ収集におけるセンサとアクチュエータのように，トピックの Publisher あるいは Subscriber だけの役割をするノードの範囲を分け，キー空間において，トピック名より低い優先度で順序関係を与え，それぞれの範囲を隣接させることで満たされる性質となっている．強リレーフリー性での不在トピックへの転送防止では，階層型のキー順序保存型オーバレイネットワークにおいて，ノードが隣接ノードを必ず把握できることに着目し，Subscriber に隣接する Publisher のノードを rendez-vous point（以下 rp）として，Subscriber の存在監視をさせる．Subscriber の有無が変わったら，rp は他の Publisher へ通知することで，publish の中断あるいは再開を制御し，Subscriber 不在時のメッセージ転送を防ぐ．

本論文では，SAPS の BOTH 方式における ALM トピックおよび OFM トピックの 2 トピック間において，相手トピックメンバの不在検知を実現する必要がある．SAPS では，Publisher が Subscribe することを前提としているため，提案手法では ALM トピックと OFM トピックの範囲を隣接させ，境目にそれぞれ rp を配置することで，お互いに存在監視させる（**図 6**）．

5.2 キー名書換え移行の導入

BOTH 方式のトピック移行における従来の join/leave 移行は移行所要時間が長くなるため，OFM が構築されてからトピック移行が完了するまでの間に，OFM publish に加えて ALM publish で二重に配送されるメッセージが増加してしまう．これを削減するためには，移行所要時間の短縮が必要であるが，複数ノードの移行開始をいっせいに行うと Churn が発生し，移行完了が極端に遅れるノードが

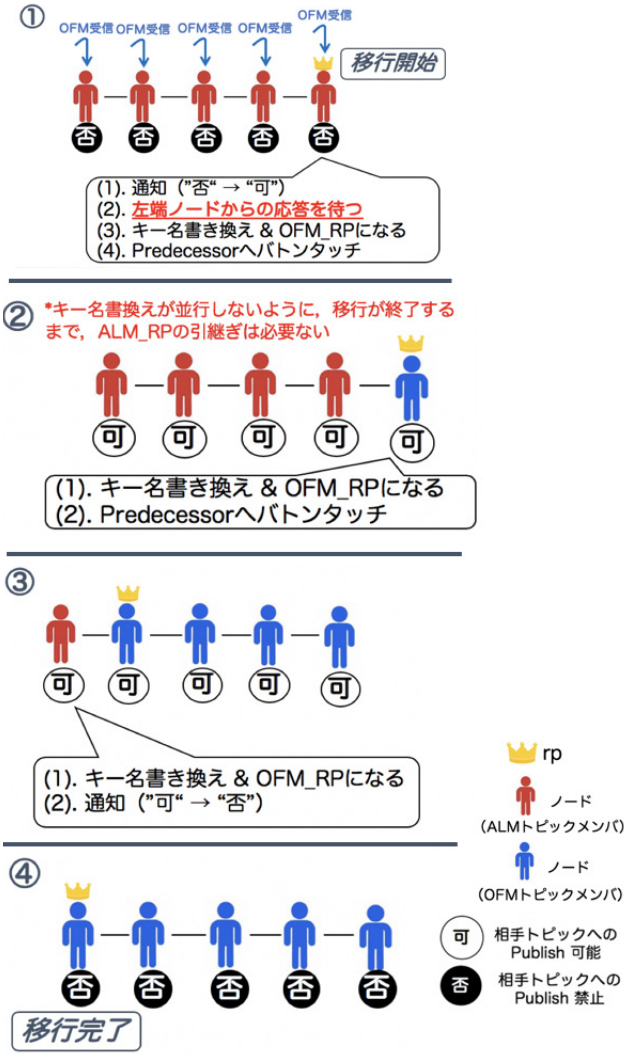

図 7 キー名書換え移行の流れ

Fig. 7 Sequence of migration by key-name-change.

発生するなど移行所要時間は安定せず，Broker の処理負荷も大きくなる．そこで，ノードの join/leave をともなわずにノードの「キーのトピック部分の書き換え」だけで実質的なトピック移行を完了させるキー名書換え移行の導入を提案する．キー名はトピック名の次に Broker 起動時にオーバレイへ join する際にランダムで一意に与えられた PeerID が続く文字列である．同じトピック名を持つキーどうしは隣接して挿入され範囲を形成し，PeerID の大きさで各ノードの並びが決まる．

キー名書換え移行も 2 トピック間で強リレーフリー性が満たされることを想定している．各ノードが右方向にキーの値が大きくなる循環双方向リストを形成している前提で，書換え後の並びでキーのトピック部分の辞書順（ALM トピック ＜ OFM トピック）と不整合を起こさないように，トピックの右端に位置する ALM トピックの rp（以下 ALM_RP）からトピックの左端に位置するノードに向かって逐次的にキー名書換えを実施する．キー名書換え移行の単純な流れを**図 7** に示す．以降，図 7 について述べる．

- 図 7 の①：
  トピック移行のトリガは OFM publish 受信だが，キー名書換え移行では，その際右端の ALM_RP だけが移行処理を始める．以後，トピック移行中は両トピックのメンバが存在することになるため，(1) および (2) で ALM_RP は OFM トピックへの publish を許可する旨を ALM トピックメンバへ通知する[*4]．次に ALM_RP はその通知が届いたことを確認してから (3)，キー名書換えを実施し ALM_RP から OFM トピックの rp（以下 OFM_RP）になる (4)．最後に隣接左ノード（以下 Predecessor）へ次の移行を促して移行処理は完了する．
- 図 7 の②：
  (1) 隣接右ノード（以下 Successor）から移行処理のバトンを受けたノードは，キー名書換えと OFM_RP の引継ぎを行い，(2) 最後に Predecessor へ次の移行を促す．
- 図 7 の③：
  ALM トピックの左端の位置にいるノードは，Successor から移行処理のバトンを受けた後，②の (1) と同様な処理を行う．ノードは Predecessor が ALM トピックメンバではないことを確認すると，(2) OFM_RP として ALM トピックへの publish を中断するよう OFM トピックメンバに対して通知を送信し，それがすべてのメンバに同期されることで，すべてのキー名書換え移行は完了する（図 7 の④）．

5.3 キー名書換え移行の適用条件と導入効果について

トピックは一意に順序を決定するために，キー名に加えて PeerID でソートされるため，トピック移行後に ALM トピックに join するノードの PeerID は，その他すべての OFM トピックメンバの PeerID より小さくなければ，キー名書換え移行後の並びに不整合が生じてしまう．このためキー名書換え移行を実施できるのは，OFM メンバが存在しない状態での移行に限られる．これに対しては，キー名書換え移行中に，同一トピックの OFM_RP のキーを参照し，それより小さいキーになるように PeerID を設定することで，初回移行時に限らずトピック移行時全般でキー名書換え移行を使うことができる．しかし，PeerID の変更により新たにノード間でのメッセージ交換が必要となるため，従来の join/leave に対するメリットが減少する．また，SAPS では，トラフィック量などの通信コストの削減数が多い（= メンバ数の多い）トピックから OFM 木を構築して切り替える方針のため，従来の join/leave 移行では，初回のトピック移行こそが Churn が発生しやすく遅延時間も長くなる．このため，現時点では実装をシンプルにすることを優先し，初回移行時のみの実装となっている．なお，初回移行以外の場合は，トピック移行に関わるノード数によって移行時間が変化するが，本論文では実装していないため，その調査は今後の課題とする．

[*4] ALM トピックメンバが publish したメッセージを OFM トピックメンバへ送信できるようにするため，キー名書換えの前に相手トピックへの publish を許可させる通知が必要．

5.4 提案手法の実装について

DDLL にはリモートノードへ指定の処理を実行させるためのイベント機構が存在し，1 方向イベントあるいは Request/Reply 型のイベントを定義し，メッセージとして送信できる．受信ノードのキューへ enqueue されたイベントは逐次処理される．

rp による相手トピックメンバ有無の状態変化の検知は，ノードの join/leave にともなうリンク更新のメッセージを受信した rp が，更新されたリンク先のキー名をチェックすることで，検知できるようにしている．その状態変化をトピックメンバへ通知するためには，通知用のメッセージを rp からトピックの端のノードまでリレーするように DDLL の既存イベントを拡張実装することで対応した．ノードの join/leave にともなう rp の引継ぎについては，各ノードがリンク更新のメッセージを受信するたびに，「自分が rp かどうか」をチェックし，状態変数を更新することで非同期で引き継がれる．

キー名書換えを実施するノードによる自身のキー名および関連データの書き換えは，他のノードとの通信なしで行われる．書き換え後，Predecessor へのイベントのリレーに加えて，書き換えによって生じた一時的な不整合を解消するために，影響を受ける他のノードの経路表の値も更新する．

6. 評価

Suzaku を構造化オーバレイに用いたシステム上に従来手法と提案手法を実装して比較を行う．移行時に ALM によって発生する無駄なメッセージがどの程度削減されたかを比較したいため，図 4 から OFC と OFGate を除いた環境で，ALM メッセージのみを計測する．トピック移行のトリガは特殊なペイロードを含む ALM publish で代用しており，移行期間中のメンバの増減や故障の発生は想定しないものとする．

6.1 トピック移行の所要時間比較

まずは，従来手法と提案手法の単純なトピック移行所要時間を比較した．従来手法も Suzaku を構造化オーバレイに用いたシステム上に実装し直しているため，Suzaku により join/leave の性能が改善している．従来手法のトピック移行所要時間としては，最初にトピック移行を開始したメンバの処理開始時間と最後にトピック移行を終了したメンバの処理終了時間の差分を示す．提案手法のトピック移行所要時間としては，その時間に加えてトピック移行前後の

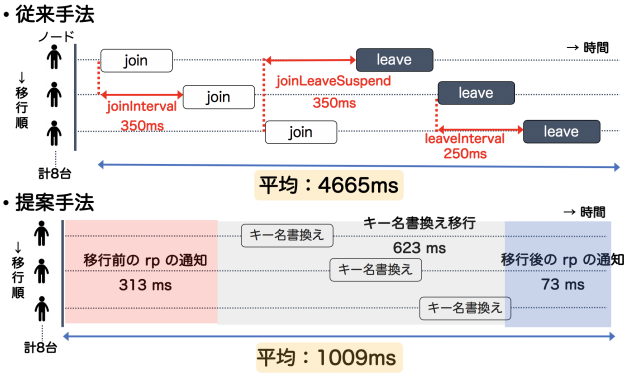

図 8 トピック移行の所要時間比較

Fig. 8 Comparison of time required for topic migration.

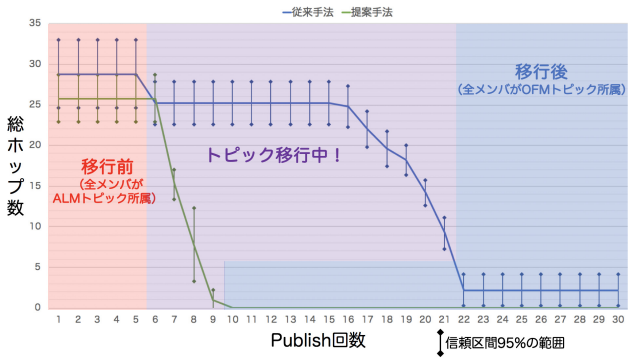

図 9 総ホップ数推移の比較結果（10 回平均）

Fig. 9 Comparison of changes in total number of hops (average of 10 trials).

rp の通知に要する時間も含める．結果の概要として，計測 10 回の平均値として，従来手法 4,665 ms ±47 に対し，提案手法 1,009 ms ±100 と 1/4 以下の短縮を確認した（図 8）．以下で結果の詳細について述べる．

本論文での評価環境は仮想化環境でソフトウェアスイッチを用いてホスト間を接続する環境を用いている．この環境では，処理の重複が多くなると負荷が高くなり，処理のタイムアウトや，移行時間が長くなる状態が起こる．そこで，本評価環境において，join/leave の実装が正常に動作する状態を再現できるよう，図 8 に示したような joinInterval（メンバの join 開始間隔），joinLeaveSuspend（最後の join 開始から最初の leave 開始までの間隔），leaveInterval（メンバの leave 開始間隔）の 3 つのパラメータを設定した．これらのパラメータの値を変化させながら十分な回数の事前計測を行った結果，タイムアウトが発生せず所要時間が安定する最小値として，joinInterval および joinLeaveSuspend は 350 ms，leaveInterval は 250 ms を用いた場合，トピック移行の所要時間平均（10 回）は 4,665 ms ±47 となった．一方，今回の環境では Broker の負荷増大によって他の Pub/Sub 処理などに影響を生じるため実用的ではないが，これを無視して joinInterval を 100 ms まで下げた場合，メンバごとの join 処理の所要時間自体が長くなるため，重複低減のために joinLeaveSuspend は 700 ms へ延ばす必要があるが，トピック移行の所要時間は下げられる．それでも 3,400 ms 程かかるため，提案手法では 1/3 以下に所要時間を短縮できることが分かる．

対して，提案手法のトピック移行における所要時間の内訳をみると，移行前後の通知に 4 割程度の時間がかかることが分かる．一方，各メンバのキー名書換え自体は 100 ms を下回ることが多く，キー名書換え自体が他のノードとの通信なしで済むことことから，所要時間を短縮できたと考えられる．なお，6.3 節で述べるように，従来手法である join/leave では，leave 処理の遅延が発生するが，簡単のため，図 8 および図 9 では leave 処理遅延が発生しなかった結果のみ示している．

表 3 従来手法と提案手法の publish に必要なホップ数の違い（M_Q と M_{OFM} を考慮しない場合）

Table 3 The number of hops required to publish a message in the former and the proposed method (excluding M_Q and M_{OFM}).

BOTH 方式に取り入れる手法	移行前	トピック移行中	移行後
従来手法	$M_{ALMt} + M_{OFMt}$	M_{ALMt}	M_{ALMt}
提案手法	M_{ALMt}	M_{ALMt}	-

6.2 両手法の総ホップ数推移の比較

続いて従来手法および提案手法における，publish に必要な総ホップ数の推移を比較する．従来手法の joinInterval，joinLeaveSuspend，leaveInterval には，引き続き同じ値を設定している．Suzaku を構造化オーバレイに用いたシステム上への移植により，join/leave 移行の所要時間が短縮されたため，a の publish 間隔も（3.1 節での実験における）2 秒から 0.2 秒へと短くした．計測方法は，基本的に 3.1 節と同じだが，M_Q と M_{OFM} を考慮しないため，フェーズごとの publish に必要な総ホップ数の式は**表 3** のようになる．また，提案手法では強リレーフリー性の拡張によってメンバ不在トピックへの転送を防ぐため，移行前の M_{OFMt} および移行後の M_{ALMt} は式に含まない．

計測結果を図 9 に示す．図 9 では，両手法でランダムに生成したオーバレイネットワークを 10 組用意し，各 publish における総ホップ数の平均値および信頼区間 95%の範囲の比較結果を示している．移行前および移行後のフェーズでは，メンバ不在トピックへの配送阻止により，提案手法の総ホップ数が下回っている．またトピック移行時間の短縮により，提案手法では ALM での配送がより早くストップしていることが確認できる．従来手法では join の後に leave するため，総ホップ数の減少開始がさらに遅くなっている．

そして提案手法を取り入れた BOTH 方式では，メンバ不在トピックへの転送を防ぐことが可能なので，トピック移行中の publish 以外は EITHER 方式を下回ると考えられる（**表 4**）．

表 4 EITHER 方式と提案手法の publish に必要なホップ数の違い
Table 4 The number of hops required to publish a message in EITHER and the proposed method.

切替方式	移行前	トピック移行中	移行後
EITHER 方式	$M_{ALM} + M_Q$	-	$M_{OFM} + M_Q$
BOTH 方式（提案手法）	M_{ALMt}（$= M_{ALM}$）	$M_{ALMt} + M_{OFM}$	M_{OFM}

6.3 評価結果に関する考察

本論文での評価を通して，提案手法および join/leave に対する課題が確認できた．以下ではこれらの課題について述べる．

6.3.1 提案手法における課題

提案手法における課題としては，rp からの通知やキー名書換え移行時に発生する DDLL イベントをノード間でリレーしている間のノードの join/leave や故障発生時の遅延があげられる．これらの事象には，たとえばイベントを Request/Reply 型で定義し，送信メッセージのタイムアウト検出や Nak 受信時に隣接ノードを再計算し，リクエストを再送するといったリカバリ機能を実装する方法が考えられる．しかしリカバリが完了するまでの間，リレー進行の遅延は防ぐことができない．以上のような点から，提案手法はノードの出入りが頻繁なトピックに対して適用できない可能性がある．以下では想定するアプリケーションや障害によって発生するノードの出入りの頻度とその影響について述べる．

SAPS では，OpenFlow ネットワークでのマルチキャストを活用して，Pub/Sub の性能を改善するため，基本的には特定の配信元から多数の受信者に配信する 1 対多通信に有効な Pub/Sub 基盤となっている．1 対多通信が有効な例としては，Twitter のような SNS において，著名人のツイートをフォローするケースや，電力の需給バランスを維持するためのデマンドレスポンスなどがあげられる．たとえば Twitter の場合，Socialbakers 社が提供する統計 [24] では，著名人として最もフォロワ数の多いバラク・オバマ氏のフォロワ数は 1 億人程度となっており，また，OMNICORE 社が提供する統計 [25] では，Twitter ユーザ全体に対して 46%程度の人が毎日プラットフォームを利用していると述べられている．粗い推定となるが，これらからオバマ氏のフォロワが毎日 1 回プラットフォーム上で Subscribe, Unsubscribe すると考えると，毎秒 500–600 件の Subscribe/Unsubscribe が発生することになる．文献 [26] では，データセンタで用いられている機器における障害の発生頻度について調査しており，図 4 には機器ごとの障害発生率，表 4 には 1 日あたりの障害発生件数が示されている．これらの結果から，障害による join/leave の発生頻度は Twitter におけるフォロワの Subscribe/Unsubscribe に比べると十分に小さいと考えられる．

Twitter の例で最も出入りが激しい状況になった場合，Broker 上で同じ頻度の join/leave が発生すると，提案手法のトピック移行期間に多数の join/leave が発生する可能性があるが，4.3 節で述べたとおり，現実的なネットワークに適用する際には，Broker にはある程度の担当範囲を想定しているため，Broker 配下にあるトピックの Publisher/Subscriber 数が 0 になったとき，あるいは，Publisher/Subscriber が 0 から 1 になったときにオーバレイネットワーク上の join/leave が発生することになる．そのため，上述のような非常に Subscriber 数が多いトピックでは，オーバレイネットワーク上の join/leave 発生数は Publisher/Subscriber の join/leave に対して十分に小さくなると考えられる．

一方で，Subscriber 数の少ないトピックでは，Pub/Sub クライアントの join/leave がそのまま Broker 単位での join/leave となる可能性が高く，join/leave 頻度も高くなると考えられる．文献 [2] で述べられているように，BOTH 方式で，あるトピックに新規参加した Publisher/Subscriber は，たとえ OFM 木が構築されていたとしても，直接 OFM 木に追加されるのではなく，ALM トピックに参加して，ALM で Publish/Subscribe することになる．そのうえで，OFM 移行の条件を満たした際に，ALM に残っているノードが再度移行処理を行うことになる．そのため，Unsubscribe 時に OFGate への問合せが発生することを除いては，基本的には Suzaku によって ALM トピックへの join/leave が行われるため，文献 [7] などで述べられているオーバレイネットワークベースの Pub/Sub 基盤に近い性能を実現することができると考えられる．以上のようなことから，ノードの出入りの頻度については配備方法を調整することで，ある程度低減できる可能性がある．なお，5.3 節で述べたとおり，初回移行時以外のトピック移行は join/leave 方式しか実装していないため，その性能評価は今後の課題とする．

6.3.2 join/leave に関する課題

他の課題としては，現状の join/leave の実装自体が本来の性能を出せていない点があげられる．本論文では同一の実装上で比較を行っているため，比較結果自体は処理内容の複雑さを比較できていると考えているが，本来は join/leave の処理時間も含めてより少ない時間で処理できることが望ましい．より高性能な評価環境を整備したうえで，join/leave の性能を評価する必要がある．ほかにも，DDLL のノード故障検出アルゴリズムとの競合により，leave 処理でさらに遅延が生じる可能性がある．本論文では，各ノードで定期的に実行される故障チェックが終わるまで leave 処理が開始できない実装を用いていたため，たとえば Predecessor の leave と故障チェック実施ノードによる Predecessor への故障確認メッセージの送信が被るとタイムアウトとなり，その間故障チェック実施ノードの leave 処理は中断する．この点についても，今後実装改良により競合発生時の遅延時間を短縮する予定である．さら

に，これらの修正を経たうえで Churn 発生や運用規模の違いによる影響調査を行う必要があると考えている．

7. おわりに

本論文では，SAPS の BOTH 方式における publish 時の不必要なメッセージ配送を改善するための手法について検討した．メンバ不在トピックへのメッセージ転送に関しては，構造化オーバレイネットワーク上の性質である強リレーフリー性を拡張し，相手トピックのメンバ不在時の publish 中断を実現することで，これを改善した．さらに，トピック移行時以外の publish に必要な総ホップ数で EITHER 方式を下回ることが可能になった．トピック移行所要時間に関しては，逐次的にノードのトピックを書き換えていくキー名書換え移行を提案し，従来の join/leave 移行と比較して，OFM 切替え時のトピック移行所要時間を大幅に短縮した．一方の join/leave 移行では，Churn 回避のための遅延に加えて，DDLL の故障チェックとの競合による leave 処理遅延が懸念される．leave 処理遅延に関しては，たとえば各ノードが故障チェックされるタイミングを把握しこれを避けて leave することで，故障チェックのタイムアウトとともに故障チェック実施ノードの leave 処理遅延も回避できると考えられる．トピック移行所要時間の短縮に関して，本論文では BOTH 方式を対象としてキー名書換え移行を提案したが，join/leave 移行はたとえばネットワーク移動（Broker 自体の移動）にともなうオーバレイ上での任意のグループ間移動を必要とするアプリケーションなどに活用できる可能性があり，並行して検討する必要があると考えている．

今後の課題としては，評価環境ならびに join/leave の実装を改善し，Churn や運用規模の違いによる影響調査を行っていく必要がある．

謝辞 本研究開発の一部は JSPS 科研費 17K00143, 19K20253 の助成を受けたものである．

参考文献

[1] mqtt (online), available from ⟨http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html⟩ (accessed 2019-05-06).

[2] Akiyama, T. et al.: Scalable Pub/Sub system using OpenFlow Control, *Journal of Information Processing*, Vol.24, No.4, pp.635–646 (2016).

[3] Hosseini, M. et al.: A survey of application-layer multicast protocols, *IEEE Communications Surveys & Tutorials*, Vol.9, No.3, pp.58–74 (2007).

[4] Chockler, G. et al.: Constructing Scalable Overlays for Pub-Sub with Many Topics, *ACM Symposium on Principles of Distributed Computing*, pp.109–118 (2007).

[5] 新納和樹ほか：下位層での配送を考慮した分散 Pub/Sub 基盤における複数管理ドメインにまたがるメッセージ配送手法の検討と評価，電子情報通信学会総合大会 (Mar. 2019).

[6] あきみち，宮永直樹，岩田　淳：マスタリング TCP/IP（OpenFlow 編），オーム社 (2013).

[7] Teranishi, Y., Banno, R. and Akiyama, T.: Scalable and Locality-Aware Distributed Topic-Based Pub/Sub Messaging for IoT, *2015 IEEE Global Communications Conference* (*GLOBECOM*), pp.1–7 (2015).

[8] Khare, S. et al.: Scalable Edge Computing for Low Latency Data Dissemination in Topic-Based Publish/Subscribe, *2018 IEEE/ACM Symposium on Edge Computing* (*SEC*), pp.214–227 (2018).

[9] Rausch, T., Nastic, S. and Dustdar, S.: EMMA: Distributed QoS-Aware MQTT Middleware for Edge Computing Applications, *2018 IEEE International Conference on Cloud Engineering* (*IC2E*), pp.191–197 (2018).

[10] Glendenning, L. et al.: Scalable consistency in Scatter, *Proc. 23rd ACM Symposium on Operating Systems Principles* (*SOSP '11*) (2011).

[11] Abe, K. and Teranishi, Y.: Suzaku: A Churn Resilient and Lookup-Efficient Key-Order Preserving Structured Overlay Network, *IEICE Trans. Communications* (Sep. 2019). DOI: 10.1587/transcom.2018EBT0001.

[12] Tariq, M.A. et al.: PLEROMA: A SDN-based high performance publish/subscribe middleware, *Proc. 15th International Middleware Conference* (*Middleware '14*), pp.217–228, (Dec. 2014).

[13] 後谷浩輔ほか：OpenFlow における経路切り替えコストを考慮したネットワーク制御手法の一検討，第 80 回全国大会講演論文集，Vol.2018, No.1, pp.279–280 (Mar. 2018).

[14] PIAX (online), available from ⟨https://github.com/piax/piax⟩ (accessed 2019-03-16).

[15] 小西佑治ほか：単一ノードに複数キーを保持可能とする SkipGraph 拡張，情報処理学会論文誌，Vol.49, No.9, pp.3223–3233 (2008).

[16] Abe, K. and Yoshida, M.: Constructing distributed doubly linked lists without distributed locking, *2015 IEEE International Conference on Peer-to-Peer Computing* (*P2P*) (2015).

[17] Ryu (online), available from ⟨https://github.com/osrg/ryu⟩ (accessed 2019-03-16).

[18] 藤田雅浩，秋山豊和：SDN aware Pub/Sub System における通信方式の通信コストを考慮した切り替え決定についての検討，電子情報通信学会総合大会，B-16-6 (Mar. 2017).

[19] 福井浩貴，秋山豊和，寺西裕一：分散型 MQTT broker の基礎評価とプロファイリングに基づく性能改善の検討，信学技報，Vol.117, No.187, IA2017-17, pp.31–35, (2017).

[20] Amazon Elastic Compute Cloud リージョンとアベイラビリティーゾーン（オンライン），入手先 ⟨https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-available-regions⟩（参照 2019-08-27）.

[21] Mininet (online), available from ⟨http://mininet.org/⟩ (accessed 2019-05-06).

[22] PIQT (online), available from ⟨http://piqt.org/⟩ (accessed 2019-03-16).

[23] Banno, R. et al.: 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 (Mar. 2015).

[24] Statistics of Twitter Profiles provided by Socialbakers (online), available from ⟨https://www.socialbakers.com/statistics/twitter/profiles/⟩ (accessed 2019-08-27).

[25] Twitter by the Numbers: Stats, Demographics & Fun Facts (online), available from ⟨https://www.omnicoreagency.com/twitter-statistics/⟩

(accessed 2019-08-27).
[26] Gill, P., Jain, N. and Nagappan, N.: Understanding network failures in data centers: Measurement, analysis, and implications, *SIGCOMM Computer Communication Review*, Vol.41, No.4, pp.350–361 (2011).

付　　録

A.1 開発・評価環境

4章で利用したPIAX ver 3.0.0rc1では，オーバレイネットワークとしてMKSGを用いた．また，6章で利用したPIAX ver 4では，オーバレイネットワークとしてSuzakuを用いた．DDLLはPIAX ver 3.0.0rc1およびPIAX ver 4で共通の実装となっており，MKSGおよびSuzakuにおける双方向連結リストの管理に用いた．分散MQTT BrokerであるPIQTについては，すでにPIAX ver 4への移行が完了しているが，論文執筆時は移行途中であったため，そのままの記載としている．また，4章，6章の評価では，評価内容に直接影響しないため，Publisher/SubscriberからBrokerへのアクセスは従来のSAPSのREST APIを用いた．現在は移行後のPIQTと統合し，MQTTによるPub/Subが可能となっている．実装・評価に用いた機器環境を**表 A·1**にまとめる．

表 A·1 実装・評価に用いた機器環境
Table A·1 An environment used for the implementation and the evaluation.

OS	Ubuntu 16.04.5
Server	DELL PowerEdge R515
CPU	AMD Opteron(TM) Processor 4133
コア数	8
メモリ量	32GB
PIAX（4章の実験）	ver. 3.0.0rc1
PIAX（6章の評価）	ver. 4
Java	ver. 1.8.0_181
Mininet	ver. 2.3.0d1
Ryu	ver. 4.13
Open vSwitch	ver. 2.5.4
OpenFlow	ver. 1.3
Python	ver. 2.7.12

盛房 亮輔

2017年京都産業大学コンピュータ理工学部卒業．2019年同大学大学院博士前期課程修了．

坂野 遼平　（正会員）

2010年北海道大学工学部情報エレクトロニクス学科卒業．2012年同大学大学院情報科学研究科修士課程修了．2018年東京工業大学大学院情報理工学研究科博士課程修了．博士（理学）（2018年9月，東京工業大学）．2012年日本電信電話（株）入社，未来ねっと研究所勤務．2018年より東京工業大学情報理工学院研究員，現在に至る．主に広域分散処理技術とその応用についての研究に従事．2015年度本会論文賞を受賞．IEEE，IEICE各会員．

安倍 広多　（正会員）

1992年大阪大学基礎工学部情報工学科卒業．1994年同大学大学院博士前期課程修了．同年日本電信電話（株）入社．1996年大阪市立大学学術情報総合センター助手．2000年同講師．2003年同大学院創造都市研究科講師．2005年同助教授．2012年同教授，2018年より同大学大学院工学研究科教授，現在に至る．博士（工学）（2000年2月，大阪大学）．分散システム，基盤ソフトウェア等に興味を持つ．2013年度本会論文賞を受賞．IEEE，IEICE各会員．

寺西 裕一　（正会員）

1993年大阪大学基礎工学部情報工学科卒業．1995年同大学大学院基礎工学研究科博士前期課程修了．同年日本電信電話（株）入社．2005年大阪大学サイバーメディアセンター講師．2007年同大学大学院情報科学研究科准教授．2008年より情報通信研究機構専攻研究員，招へい専門員を兼任．2011年より情報通信研究機構研究マネージャおよび大阪大学サイバーメディアセンター招へい准教授，現在に至る．分散システム，オーバレイネットワーク，センサーネットワークおよびその応用システムに関する研究開発に従事．2011年度本会論文賞を受賞．博士（工学）（2004年3月，大阪大学）．IEEE，IEICE各会員．

石原 真太郎

2016 年京都産業大学コンピュータ理工学部卒業，2018 年同大学大学院博士前期課程修了．現在，同大学院博士後期課程在学中．

秋山 豊和 （正会員）

1999 年大阪大学大学院工学研究科修士課程修了，2000 年同大学院博士課程中退後，同大学サイバーメディアセンター助手を経て，2005 年同センター講師．2008 年京都産業大学コンピュータ理工学部講師，2011 年同大学准教授を経て，2018 年同大学情報理工学部教授．分散システム，IoT アプリケーション等に興味を持つ．博士（工学）（2003 年 9 月，大阪大学）．IEEE CS，IEICE 各会員．