Vol.2014-IOT-24 No.18
2014/2/28

KiZUNA: P2P ネットワークを用いた 分散型マイクロブログサービスの実現

播磨 裕太[1,a)]　安倍 広多[1]　石橋 勇人[1]　松浦 敏雄[1]

概要：現在実装中の P2P ネットワークを用いた分散型マイクロブログサービス KiZUNA の設計について述べる．KiZUNA はサーバを必要としない Pure P2P 型のシステムとして実現する．メッセージの購読と配送には構造化 P2P ネットワークの 1 つである Skip Graph を用いた ALM(Application Level Multicast) を用いる．また，ハッシュタグ，全文検索，検索ストリーム，複製管理などの方式についても述べる．

キーワード：マイクロブログ，Pub/Sub, P2P，Skip Graph, Bloom Filter

KiZUNA: An Implementation of Distributed Microblogging Service over P2P Networks

HARIMA Yuta[1,a)]　ABE Kota[1]　ISHIBASHI Hayato[1]　MATSUURA Toshio[1]

Abstract: This paper describes a design of KiZUNA, a pure P2P-based microblogging service. Subscription and distribution of messages are implemented using ALM (Application Level Multicast) over a skip graph, a structured P2P network. Hashtags, full-text search, search stream, and replica management are also discussed.

Keywords: Microblogging, Pub/Sub, P2P, Skip Graph, Bloom Filter

1. はじめに

Twitter，Weibo などのマイクロブログサービスが情報伝達手段として普及している．このようなマイクロブログサービスでは，ユーザは短いメッセージを投稿できる．また，ユーザは興味がある他のユーザを「購読」することで，そのユーザの投稿をほぼリアルタイムに閲覧できる．先の東日本大震災では，被災者・自治体などの公的機関，地元メディア，ボランティアなどが Twitter を用いて様々な情報をリアルタイムに発信し，その有用性は広く認識された．

現在のマイクロブログサービスはサーバクライアントモデルで構築されている．ユーザはサーバにアクセスすることで投稿メッセージを送信し，また購読している他のユーザの新しいメッセージを取得する．この方式は，増加する投稿メッセージ数に対応してサーバやネットワークを増強する必要があり（以前は過負荷による Twitter の停止は珍しくなかった），運用にコストがかかる．また，単一故障点が存在するため，震災時における通信インフラとして信頼できるかどうか疑問が残る．

これらの問題点を解決するため，著者らは P2P ネットワークを用いた分散型マイクロブログサービス **KiZUNA** を開発している．大地震などによってネットワークやサーバが被災しても利用できるように，KiZUNA はサーバを必要としない Pure P2P 型で設計した．システムをネットワーク上で完全に分散させることで，耐故障性と高いスケーラビリティを確保する．

KiZUNA では，メッセージの購読と配送，ハッシュタグ，全文検索や検索ストリームなど，Twitter の機能のサブセットを実現する．また，メッセージ発信者の詐称防止

[1] 大阪市立大学大学院創造都市研究科
Graduate School for Creative Cities, Osaka City University
[a)] harima@sousei.gscc.osaka-cu.ac.jp

(発信者認証)，メッセージの複製管理機能などを備える．

なお，ユーザのメッセージを購読者に配送する処理はトピックベースの，またハッシュタグや検索ストリームはコンテンツベースの Publish/Subscribe 型通信と捉えられる．KiZUNA は P2P ネットワークにより 2 種類の Publish/Subscribe 型通信を実現する．

本稿では KiZUNA の設計について述べる．

2. 準備

KiZUNA では，メッセージの配送のために構造化オーバレイネットワークの 1 つである Skip Graph を，また検索のためには著者らが提案している BF Skip Graph を用いる．ここでは，これらについて簡単に説明する．BF Skip Graph は，同じく著者らが提案している集約 Skip Graph をベースとしている．また，Skip Graph を構成する分散双方向連結リストを維持管理するためには，DDLL プロトコルを用いる．これらについても説明する．

2.1 Skip Graph

Skip Graph の構造を図 1 に示す．縦方向の線で繋がれた四角はノードで，その中の数字はキーを表している．また，各ノードはメンバシップベクタと呼ばれる基数 w の乱数を持つ（本稿では $w = 2$ とする）．Skip Graph は複数の階層（レベル）で構成され，レベル i では最大 $w^i = 2^i$ 個の双方向連結リストが存在する．レベル 0 ではすべてのノードが同じ連結リストに所属し，レベル $i\,(> 0)$ では，メンバシップベクタの上位 i 桁が一致するノード群が同じ連結リストに所属する．キーは全順序集合の要素であり，各連結リストではノードはキーの昇順に接続される．

Skip Graph の連結リストは分散双方向連結リストであり，各ノードは左右のノードへのポインタ（IP アドレスなど）を持つ．また，この連結リストは，左端と右端のノードが接続された循環双方向連結リストでもある．ノード u のレベル i での左ノードを $u.left[i]$, 右のノードを $u.right[i]$ と表記する．

Skip Graph では，任意のキーの検索を $O(\log n)$ ホップで実行できる．また，Skip Graph ではキーが昇順に並ぶため，指定したキーの範囲内のすべてのノードから値を取得する範囲検索が可能である．また，範囲検索を応用することで，指定したキーの範囲内のノードにメッセージを送信する ALM (Application Level Multicast) も実現できる．範囲検索もしくは ALM にかかるホップ数も $O(\log n)$ である．

2.2 DDLL

2.1 節で述べたように，Skip Graph は複数の分散双方向連結リストによって構成される．著者らは，ノードの並行挿入や削除がある環境でも一貫性を維持しながら分散双方向連結リストを維持管理する方式として，DDLL プロトコルを提案している [1]．DDLL は分散排他制御を行わないため，ノードの離脱などの障害に強いという特徴を持つ．KiZUNA では Skip Graph の分散双方向連結リストとして DDLL を用いる．紙数の関係上 DDLL の詳細については割愛するが，提案方式と密接に関連する近隣ノード集合についてはここで説明する．

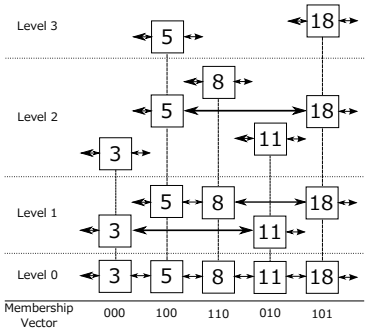

図 1 Skip Graph の例
Fig. 1 Example of a Skip Graph.

双方向連結リスト上のあるノード X が突然離脱あるいは故障した場合を考える．このとき，X の右ノードと左ノードの間にリンクを張って連結リストを修復しなければならない．この目的のために，各ノードは自ノードの左側の直近 k 個のノードのキーとポインタを維持する（これらを近隣ノード集合と呼ぶ）．各ノード u は u の左ノードが突然離脱あるいは故障した場合，u の近隣ノード集合を用いて u の左側で最も近い，生存しているノード q を探し，u と q の間でリンクを張ることで連結リストを修復する．

近隣ノード集合の更新はアクティブに行う．すなわち，ノードの挿入，削除およびリンクの修復を行った場合，その都度変更のあったノードの右側 k 個のノードの近隣ノード集合を更新する．

2.3 集約 Skip Graph

著者らは，Skip Graph の構造を利用して，指定したキーの範囲における集約値（平均，合計，最大，最小など）を効率良く求める集約 Skip Graph を提案している [2]．集約の対象は，各ノードがキーとは別に保持する任意の値 (value) である．value はキーとは無関係の値であり，変化しても良い．集約 Skip Graph では，ノード u はレベル $i\,(i > 0)$ において $[u.left[i+1], u.left[i])$ の範囲に含まれるノードの value の集約値を保持する[*1]．この集約値は定期的に更新する．ある範囲の集約値は，範囲内の 1 つ以上のノードが保持する集約値を集めることで計算する．

集約 Skip Graph において指定範囲の集約値を求めるために必要なホップ数とメッセージ数は $O(\log n)$ である．

[*1] 集約する範囲が文献 [2] と左右逆になっているが，本稿では実装に合わせている．

2.4 BF Skip Graph

BF Skip Graph は，複数のノードが分散して保持するテキストに対する全文検索を効率的に行う仕組みである [3]．集約 Skip Graph と同様の方法を用いて各ノードが（value として）保持する Bloom Filter を集約する．Bloom Filter は，ある要素が集合に含まれるかどうかを定数時間で判定する確率的データ構造である．

各ノードは，保持するテキストを形態素解析や n-gram などにより単語単位に分解し，Bloom Filter に登録する．各ノード u はレベル $i(i>0)$ において $[u.left[i+1], u.left[i])$ の範囲に含まれるノードの Bloom Filter を（ビット OR により）集約した Bloom Filter を保持する．この範囲内のノードが保持するテキストに単語 q が含まれている場合，集約した Bloom Filter にも q が含まれる．q が含まれるテキストを検索するには，各レベルで集約した Bloom Filter の中に q が含まれているかを調べ，含まれている場合はそれぞれ下位の適切なノードにクエリを転送する．BF Skip Graph では，複数の語の AND 検索も容易に実行できる．

Bloom Filter の集約は各ノードが定期的に行う．ローカルな Bloom Filter の変更が全ノードの（集約した）Bloom Filter が反映されるには $O(\log n)$ 回の集約処理が必要であるため，新規に追加したテキストを検索できるようになるにはある程度の時間を要する．

なお，BF Skip Graph は Skip Graph の機能も備えているため，キーによる検索も可能である．

3. 関連研究

P2P システムによりマイクロブログサービスを実現する先行研究として，Megaphone[4] と Cuckoo[5] がある．

3.1 Megaphone

Megaphone は Pure P2P 方式で実現されているマイクロブログサービスである．メッセージの配送には Pastry を用いた ALM の一種である Scribe を使用している．各ユーザは公開鍵証明書によって識別されるが，公開鍵証明書を取得するには外部のサービス（Web や電子メールなど）を必要とする．また，ハッシュタグや全文検索の機能は提供されていない．

3.2 Cuckoo

Cuckoo はハイブリッド P2P 方式で実現されているマイクロブログサービスである．メッセージの配送方法は 2 通りあり，購読者が少ないユーザは直接購読者にメッセージを送信するのに対し，購読者が多いユーザは Gossip プロトコルを用いてメッセージを送信する．Cuckoo はハイブリッド P2P 方式であるため，単一故障点を完全に排除できていない．また，Megaphone と同様，ハッシュタグや全文検索の機能は提供されていない．

4. 設計

ここでは KiZUNA の設計について述べる．

4.1 主な仕様

Twitter と同様，KiZUNA のユーザは英数字からなるハンドルネームを持つ．KiZUNA では Twitter が提供している機能のサブセットを提供する．以下は KiZUNA が提供する主な機能である．

- メッセージの購読と配送
- ハッシュタグ
- 全文検索
- 検索ストリーム

メッセージの配送先は Twitter の仕様に準ずる．まず，あるユーザが発信したメッセージは，そのユーザの購読者に配送される．また，メッセージに “@”+ハンドルネームが含まれる場合は，当該ハンドルネームを持つユーザにもメッセージが配送される（この機能をメンション（Mention）と呼ぶ）．また，“@”+ハンドルネームをメッセージの先頭に含むメッセージは返信として特別に扱う．メッセージの発信者を p，当該ハンドルネームを持つユーザを q としたとき，このようなメッセージは q および，p と q の双方を購読しているすべてのユーザに配送される．

ハッシュタグは，“#”+キーワード 形式の文字列であり，発信するメッセージ中に埋め込むことにより，当該ハッシュタグを登録しているすべてのユーザにメッセージが配送される．これにより，購読しているユーザに関係なく特定のテーマに対する発言を参照することが容易になる．

検索ストリームは任意の文字列を登録することで，システム内を流れるすべてのメッセージで，当該文字列を含むものが配送される仕組みである．

全文検索は，過去一定期間のすべてのメッセージの中から指定した単語を含むものを検索する機能である．

また，KiZUNA を Pure P2P 方式で実現する関係で，次の機能も備える．

- 発信ノードがオフラインの状態でもメッセージを取得できるように，複数のノードがメッセージの複製を維持する．
- メッセージの偽造を防止するため，電子署名により発信者認証を行う．

以下，ユーザ u が使用する計算機をノード u と表記する．また，ユーザ u とノード u を特に区別せずに使用することがある．

4.2 P2P ネットワークの構成

KiZUNA は，以下の 4 つの構造化 P2P ネットワークを用いて実現する．

SG_{msg}　メッセージ配送 (Skip Graph)

SG_{stream}　検索ストリームとスクリーンネームによる配送 (BF Skip Graph)
SG_{full}　全文検索 (BF Skip Graph)
DHT　分散ハッシュテーブル (Skip Graph)

KiZUNA では利用するすべての P2P ネットワークを Skip Graph ベースとすることで，実装コストを下げている（KiZUNA は P2P 基盤ソフトウェア PIAX[6] 上に実装しているが，PIAX の DHT は Skip Graph を用いて実装している）．

4.3 ユーザ

Twitter では各ユーザは一意のハンドルネーム（スクリーンネーム）を持つが，サーバを用いない KiZUNA では一意のハンドルネームを発行することが困難である．このため，KiZUNA ではユーザを識別するために，乱数によって生成した，衝突する可能性を無視できる程度に長いビット列（**UID** と呼ぶ）を用いる．

ユーザにとっては UID は不便であるため各ユーザは英数字によって構成されるスクリーンネームも持つ．スクリーンネームは識別には用いないが，他のユーザと同一のスクリーンネームが存在することは好ましくないため，アカウントを生成するときには，指定したスクリーンネームが使われていないかどうかをチェックする（詳細は 4.11 節で述べる）．ただし，このチェックは完全ではない（同時に同じスクリーンネームを登録しようとした場合など）．以下，ユーザ u の UID を $u.uid$ と表記する．

メッセージに電子署名を付加するため，各ユーザは公開鍵と秘密鍵のペアを保持する（詳細は 4.5 節と 4.13 節参照）．

4.4 メッセージ

各ユーザが発信するメッセージにはシーケンス番号を付与する．ユーザ u がメッセージを発言（発信）するとき，P2P ネットワークを用いて下記のデータを配送する．

- u の UID とハンドルネーム
- シーケンス番号
- タイムスタンプ
- メッセージ本体
- 電子署名

4.5 ユーザデータの格納

各ユーザ u に関するデータは，以下のように u が使用するノード（ノード u）と DHT に格納する．

ノード u はローカルに以下の情報を保持する．

- $u.uid$ とハンドルネーム
- u の公開鍵と秘密鍵のペア
- u が発信した過去すべてのメッセージ（過去ログ）
- u が購読しているユーザの UID と公開鍵

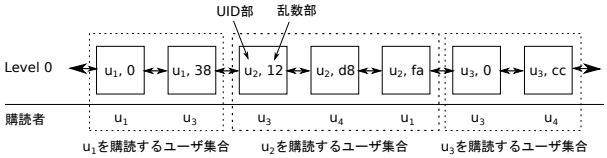

図 2　SG_{msg} の構造（レベル 0）
Fig. 2　Structure of SG_{msg} (Level 0).

- u が購読しているユーザから受信した過去一定期間のメッセージ（複製として用いる）

DHT には $u.uid$ をキーとして，u のハンドルネームと公開鍵を格納する．

4.6 購読/配送アルゴリズム

メッセージを購読者に配送する方法について述べる．

ユーザによっては多数のユーザから購読される場合がある (数万〜数千万)．このため，KiZUNA では，Skip Graph (SG_{msg}) 上で ALM によるマルチキャストを用いることでメッセージを効率的に配送する．

2.1 節で述べたように，Skip Graph の ALM では指定したキーの範囲にメッセージを送信できる．これを利用し，KiZUNA では，(1) ユーザ u を購読するすべてのユーザは SG_{msg} 上で連続した範囲（u の配信範囲と呼ぶ）にキーを挿入し，また (2)u がメッセージを発信する場合は，u の配信範囲に対して ALM によりマルチキャストすることにより，u を購読するユーザにメッセージを配信する（この方法は文献 [7] の手法と類似している）．

メッセージの複製管理を行うノードを分散させるために (詳細は 4.10 節で述べる)，u の配信範囲の中で，u を購読するノードをランダムに並べる．このために，u を購読するノードは $(u.uid, rnd)$ というペアを 1 つのキーとして SG_{msg} に挿入する．ここで，rnd は $1 \leq rnd \leq$ RNDMAX を満たす乱数である．2 つのキー (uid_1, rnd_1) と (uid_2, rnd_2) の大小関係は，uid_1 と uid_2 が等しくない場合はこれらの大小関係によって決定し，等しい場合は rnd_1 と rnd_2 の大小関係によって決定する．また，u の配信範囲は $[(u.uid, 1), (u.uid, \text{RNDMAX})]$ となる．

ノード u はキー $(u.uid, 0)$ を挿入する．これにより，(1) u のメッセージを u の購読者に配送する際に u を購読していないノードを経由することを避けられる．また，(2) u が保持する u のすべての過去ログを全文検索の対象にできる (4.11 節参照)．

SG_{msg} の例を図 2 に示した（レベル 0 部分のみ）．u_1 を u_3 が，u_2 を u_3 と u_4 と u_1 が，u_3 を u_4 が，それぞれ購読している状態を示している．破線で囲まれている領域が，それぞれの配信範囲を表している．なお，$(u_2, 0)$ が存在しないため，u_2 はオフラインであることが分かる．

各ノードは，オンラインになったときに自身が購読する

すべてのユーザ q に対し，SG_{msg} にキー $(q.uid, rnd)$ を挿入し，オフラインになったときにそれらのキーを削除する．

4.7 メンションと返信の配送

メンションおよび返信の配送について述べる（配送ルールについては 4.1 節で述べた）．

ユーザ u が発信したメッセージ m が “@” $+h$ を含む場合（つまり，ユーザ h へのメンションあるいは h への返信の場合），u は u の購読者と h にメッセージを送信する．（返信の場合，本来は u と h の双方を購読しているユーザへのみメッセージを配送すればよいが，u は u と h の双方を購読しているユーザを簡単には把握できないため，u の購読者すべてに送る．u の購読者で h を購読していないユーザでは，当該メッセージの表示を抑制する．）

u が h にメッセージを送るためには SG_{stream} 上でキー h に対して送信する．h がオンラインの場合，SG_{stream} にキー h を挿入しているので，これにより h は m を受け取ることができる．この時点で h がオフラインだった場合，m を受け取ることができないが，h がオンラインになった時点で “@” $+h$ を全文検索することで m を取得する（全文検索については 4.11 節参照）．

4.8 ハッシュタグの登録と配送

ハッシュタグの登録と配送も SG_{msg} を用いて行う．ユーザ h がハッシュタグ t を登録する場合，h はキーとして “#” $+t$ を挿入する（t を登録するすべてのノードが同一のキーを挿入する）．なお，ハッシュタグのためのキーと通常の配送のためのキー $(u.uid, rnd)$ との大小関係は実装で定める．ユーザがハッシュタグを含むメッセージ m を発信する場合，含まれるハッシュタグ t に対して，SG_{msg} 上で範囲 [“#” $+t$, “#” $+t$] に対して ALM を行う．m の配送時に h がオフラインだった場合は m を受け取ることができないが，メンションの場合と同様，オンラインになった時点で “#” $+t$ を全文検索することで m を取得できる．

4.9 メッセージの削除

メッセージ m を削除する場合，削除用に m のシーケンス番号を含んだ擬似的なメッセージ m_{del} を，m の配送先（メンションや返信などがある場合は，そのための宛先を含む）に配送する（m_{del} には通常のメッセージと同様，新規のシーケンス番号を付与する）．m_{del} を受信したノードは，m の内容を削除する．m_{del} は通常のメッセージと同じように複製管理を行うことで，m_{del} の配信時にオフラインであったノードもオンラインになった時点で m の内容を削除できるようにする．

4.10 複製管理

ユーザ u が発信したすべてのメッセージは，ノード u がすべて保持するが（4.5 節参照），u がオフラインの間にも他のノードからメッセージを取得できるようにするために，u 以外のノードが u のメッセージの複製を維持する．

u が発したメッセージは常に u の購読者に配送されているため，u の購読者が複製を維持する．これにより，複製を作成するための通信コストを省略できる．u の過去のメッセージを取得するには SG_{msg} 上で範囲 [$(u.uid, 0)$, $(u.uid,$ RNDMAX)] に対して範囲検索すれば良い．

記憶容量の圧迫を避けるため，購読者ノードが複製を維持する期間は過去一定期間に限定する．なお，発信者本人はすべてのメッセージを保持しているため，発信者がオンラインの間はすべての過去のメッセージを取得できる．

u を購読する v は，オンラインになったときに（SG_{msg} にキー $(v.uid, rnd)$ を挿入する際に），隣接ノードから，v がオフラインの間のメッセージを取得する．

さて，上記の方法では，メッセージの複製数はオンラインの購読者数と等しいため，オンラインの購読者が減少すると複製数が減少してしまう．このため，オンラインの購読者数が一定数 r を下回った場合，ランダムに選択したノードに u を強制的に購読させる（これを複製のための購読と呼ぶ）．複製のために購読しているノードでは，メッセージは受信するが表示は行わない．

u のオンラインの購読者数（すなわち複製数）が r を下回っているかどうかを判定するには，次のように DDLL の近隣ノード集合を用いる．SG_{msg} において，u の配信範囲の右端のノード（p とする）のレベル 0 の近隣ノード集合は，p の左側の直近 k 個のノードへのポインタを含んでいる（2.2 節参照）．このため，p は近隣ノード集合内でキーの値が $(u.uid, any)$ であるものを数えることで u のメッセージの複製数が r を下回るかどうか判定できる（そのためには $r \leq k$ となるように k と r を選ぶ）．この処理は近隣ノード集合が変更された契機で行う．u の配信範囲の右端のノード（p）を複製管理ノードと呼ぶ．

複製数が不足する場合，複製数を維持するために p は何らかの方法（乱数で生成した UID に最も近いノードを SG_{msg} から検索するなど）で u を購読していないノード（q とする）へのポインタを取得し，q に購読要求を送信する．購読要求を受信した q は，u を（複製のために）購読する．複製のための購読は q がオフラインになると停止し，次にオンラインになった時には購読しない．

複製管理ノードは複製数を維持する責任を持つ．ノードが SG_{msg} に挿入する際のキーとして，UID と乱数のペアを用いている理由は，複製管理ノードがなるべくランダムに選ばれるようにするためである．

4.11 全文検索

メッセージの全文検索のために，1 つの BF Skip Graph（SG_{full}）を用いる．各ノードが保持するメッセージを形態

素解析や n-gram によって単語単位に分解し，Bloom Filter に登録する．これを BF Skip Graph を用いて集約することによって全文検索を実現する．

オフラインのノードが発信したメッセージも検索できるようにするために，各ノードが保持している（他のノードから受信した）メッセージの複製も全文検索の対象とする．ただし，ユーザ u のすべての購読者が保持している u のメッセージの複製は基本的に同一であるため，u のすべての購読者が SG_{full} に参加することは無駄である．

KiZUNA では SG_{msg} において，u の配信範囲の左端の t 個のノードが SG_{full} に参加する．自ノードが配信範囲の左端の t 個のノードに含まれるかどうかの判定は，SG_{msg} のレベル 0 の近隣ノード集合を参照すればよい（4.10 節で述べた複製数の判定と同様．$t \leq k$ となるように k を選ぶ）．t 個のノードに含まれる場合，SG_{full} に対し，u のスクリーンネームをキーとして SG_{full} に挿入し，u のメッセージの複製を Bloom Filter に登録する．スクリーンネームをキーとする理由は，SG_{full} を，メンションを含むメッセージの配送にも用いるためである（4.7 節参照）．

ユーザ u がオフラインでも，SG_{full} 上で u のスクリーンネームを検索できる．このため，アカウント作成時に，指定したスクリーンネームが使われているかどうかを SG_{full} を使って判定できる（4.3 節参照）．

すべての過去のメッセージを Bloom Filter に登録すると，Bloom Filter に登録する要素数が多くなり，Bloom Filter の偽陽性確率が上昇してしまう．このため，Bloom Filter に登録するメッセージは，過去一定期間のメッセージに絞る．

4.12 検索ストリーム

文献 [3] では，全文検索のための P2P ネットワークとして BF Skip Graph を提案したが，テキストと検索クエリの関係を入れ替えることで検索ストリーム（4.1 節参照）も BF Skip Graph で実現できる．

各ノードが保持する検索文字列を表す Bloom Filter を作成し，これを集約する．ユーザがメッセージ m を発信するとき，m を単語単位に分解し，各単語が最上位レベルの集約した Bloom Filter に含まれる場合は下位レベルのノードにメッセージを転送する．これを最下位レベルまで繰り返すことで，各ノードに，それぞれの検索文字列を含むメッセージが配送される．

検索ストリームのための BF Skip Graph は専用に用意する（SG_{stream}）．各ノード u は，SG_{stream} に対し $u.uid$ をキーとして挿入する．

4.13 発信者認証

P2P システムではメッセージは容易に偽造できるため，すべてのメッセージに電子署名を付与することで発信者を認証する．メッセージの検証のために必要な公開鍵は，各ユーザが自身の UID をキーとして DHT に登録しておく．

しかし，一般の DHT ではだれでもデータを上書きできるため，公開鍵の改ざんが容易に行えるという問題がある．このため，DHT の put 要求にも電子署名を付与することで，既に DHT に登録されているユーザの情報を他人が勝手に書き換えられないように制限する．

5. 実装

現在, KiZUNA を P2P 基盤ソフトウェア PIAX[6] 上に実装中である．これまでに，PIAX に対して，DDLL，DDLL をベースとした Skip Graph，さらに集約 Skip Graph と BF Skip Graph を実装した．また，KiZUNA としてはメッセージの購読と配送，発信者認証機能が動作している．

6. おわりに

本稿では，P2P ネットワークを用いた分散型マイクロブログサービス KiZUNA の設計について述べた．KiZUNA ではメッセージの購読や配送，全文検索や検索ストリーム，発信者認証といった機能を構造化 P2P ネットワークを用いて実現することで，耐障害性の高い，スケーラブルなマイクロブログサービスを実現する．今後は，未実装機能の実装と定量的な性能評価を行う予定である．また，DTN (Delay/Disruption Tolerant Network) のサポートも予定している．

謝辞　本研究は JSPS 科研費 24500089 の助成を受けている．

参考文献

[1] 安倍広多ほか：構造化オーバーレイネットワークに適した分散双方向連結リスト DDLL，情処研報，Vol. 2010-DPS-144, No. 1, pp. 1–8 (2010).
[2] Abe, K., et al.: Aggregation Skip Graph: A Skip Graph Extension for Efficient Aggregation Query over P2P Networks, *International Journal on Advances in Internet Technology*, Vol. 4, No. 3, pp. 103–110 (2012).
[3] 岩本大記ほか：P2P ネットワークにおける Skip Graph と Bloom Filter を用いた効率的な複数キーワード検索手法の提案，情処研報，Vol. 2011-DPS-146, No. 28, pp. 1–8 (2011).
[4] Perfitt, T. et al.: Megaphone: Fault Tolerant, Scalable, and Trustworthy P2P Microblogging, *Proc. of 2010 5th Intl. Conf. on Internet and Web Applications and Services*, IEEE, pp. 469–477 (2010).
[5] Xu, T. et al.: Cuckoo: Towards Decentralized, Socio-Aware Online Microblogging Services and Data Measurement, *Proc. of 2nd ACM Intl. Workshop on Hot Topics in Planet-scale Measurement*, ACM, pp. 4:1–4:6 (2010).
[6] PIAX Inc.: PIAX. `http://www.piax.org/` (2013 年 12 月 19 日確認).
[7] 坂野遼平ほか：ストリームデータの配送に向けた分散トピックベース Pub/Sub 手法の提案，信学技報，Vol. 113, No. 364, pp. 41–46 (2013).