ビザンチン障害耐性を有するキー順序保存型構造化オーバレイネットワークのためのマルチキャスト手法の提案

本多 徹[1,a)]　寺西 裕一[2,b)]　安倍 広多[3,c)]

概要：キー順序保存型構造化オーバレイネットワーク（KOPSON）では，キーの範囲を指定したマルチキャスト（キーベースマルチキャスト）が可能であり，分散 Pub/Sub システムなどで応用されている．しかし，ビザンチン障害耐性（BFT）を備えたキーベースマルチキャストを行える KOPSON は実現方法が知られていない．本稿では，BFT を有するキーベースマルチキャストの手法を提案する．著者らは先行研究で BFT を備える KOPSON を提案しているが，マルチキャストの実現方法は未定義であった．本提案は，この未定義部分を埋めるものである．メッセージ配送経路の冗長化とメッセージの検証によって BFT を備えたマルチキャストを実現する．本稿では提案手法の詳細と基礎的なシミュレーション結果について報告する．

キーワード：構造化オーバレイネットワーク，ビザンチン障害耐性，マルチキャスト，分散システム

1. はじめに

構造化オーバレイネットワークはアプリケーションレイヤのネットワークである．各ノードの自律的なルーティングにより指定されたキーを持つノードに効率的にメッセージを配送できる．そのうち，任意の全順序集合の要素をキーとし，キーの隣接関係がオーバレイネットワーク上でも保存されるものを，キー順序保存型構造化オーバレイネットワーク（Key-Order Preserving Structured Overlay Network，以下 KOPSON）と呼ぶ．KOPSON ではその構造から，指定したキー範囲内のすべてのノードへメッセージを送信する**キーベースマルチキャスト**を効率的に実行可能であり，分散 Pub/Sub システムや分散データベースなどへ応用されている [1][2].

KOPSON をインターネットなどのオープンな環境で運用する際は，規定のアルゴリズムに従わず，任意の動作を行うノード（以下 **faulty ノード**と呼ぶ）の存在を想定する必要がある．faulty ノードによって引き起こされる障害をビザンチン障害と呼び，ビザンチン障害に耐性があることをビザンチン障害耐性（BFT; Byzantine Fault Tolerance）と呼ぶ.

著者らは文献 [3] で、BFT を備える KOPSON とその上でユニキャストを行う方式を提案しているが，マルチキャストの方法は未定義であった．BFT のないオーバレイネットワークにおけるキーベースマルチキャストでは，faulty ノードによる規定のプロトコルを逸脱した動作が多くのノードに影響を及ぼす．例えば，faulty ノードが特定の相手宛の，あるいは特定の種類のメッセージを転送しないようにすることで，参加ノードの間で状態に一貫性がない状態を作り出すことができる．このような欠点は，例えばインターネット上で分散 Pub/Sub システムを用いたシステムを実用化するにあたって大きな障害となる．

著者らが知る限り，BFT を備えるキーベースマルチキャストの手法は知られていない．BFT を有するマルチキャストとして ByzCast[4]，RT-ByzCast[5] 等の既存研究があるが，これらはグループ内通信におけるマルチキャストを対象としていて，オーバレイネットワークに適用できるものではない．オーバレイネットワーク上の BFT を備えたマルチキャストとしては Kadcast[6] があるが，想定しているのはすべてのノードを対象とした全域マルチキャスト（ブロードキャスト）のみである．既存のキーベースマルチキャスト手法としては, Skip Tree Graph[7], Split-Forward Broadcasting [8] 等があるが，いずれも BFT を考慮してい

[1] 大阪市立大学工学部
Osaka City University, Department of Engineering
[2] 情報通信研究機構ネットワーク研究所
National Institute of Information and Communications Technology, Network Research Institute
[3] 大阪市立大学大学院工学研究科
Osaka City University, Graduate School of Engineering
[a)] a18tn041@vt.osaka-cu.ac.jp
[b)] teranisi@nict.go.jp
[c)] k-abe@osaka-cu.ac.jp

©2021 Information Processing Society of Japan

ない．

本稿では，文献 [3] で提案した KOPSON（以下**ベース KOPSON** と呼ぶ）上で，キーベースのマルチキャストを行うアルゴリズムを提案する．提案手法では，指定した範囲内のノードにマルチホップでメッセージを配送するが，このとき範囲内の各ノードに対し，複数のノードから同一メッセージを送信する（冗長化）．また，メッセージの検証により不正なメッセージを検出する．これらにより，BFT を有するマルチキャストを実現する．

2. 準備

本稿で提案するマルチキャストアルゴリズムは，ベース KOPSON の構造と経路表をそのまま利用する．ここではベース KOPSON について簡単に説明する．

ベース KOPSON は，メッセージ配送経路の冗長化とメッセージの検証によって BFT を有するオーバレイネットワークを実現する（本稿で提案するマルチキャストアルゴリズムでもこの方針を踏襲している）．BFT を有するオーバーレイネットワークの実現法としては，他に複数のノードで形成したグループ内で合意を取りながら動作する，Quorum ベースの方式が知られている（[9] など）．しかしこの方式は (1) 各グループに過半数の正常ノードが必要，(2) 合意のためのオーバーヘッドが大きく実装が複雑，といった問題があるため，採用しなかった．

ベース KOPSON では，faulty ノードによる以下の攻撃に対処する．faulty ノード同士が結託することも想定する．

Random Next Hop　faulty ノードがアルゴリズムに従わず，本来転送すべきでないノードや不正なレベルにメッセージを送信する攻撃．

Eclipse Attack　複数の faulty ノードが結託することで標的ノードの経路表エントリを占有し，ネットワーク上で孤立させる攻撃．

Sybil Attack　攻撃者が多数の faulty ノードをネットワークに参加させ，正常な動作を妨害する攻撃．

2.1 Authority

ベース KOPSON では，信頼の起点となる Authority の存在を前提とする．各ノードは公開鍵と秘密鍵のペアを保持しており，Authority はその公開鍵に署名を付与しておく．各ノードはオーバーレイネットワークに参加する際，Authority にアクセスし，key，TMV，および ⟨key, TMV⟩ に対する Authority の署名を取得する（これらをまとめて**認証済参加情報**と呼ぶ）．key はオーバーレイネットワーク上で各ノードを識別するためのキー，TMV（Trustable Membership Vector）は各ノードの隣接関係を決定するための α 進数の乱数である．ノード間で通信路を確立する際に，（Authority によって署名された）公開鍵と認証済参加情報を交換し，お互い署名を検証することで Authority によって承認されていないノードとの通信を防ぐ（ノード間で通信路を確立しない場合は，各メッセージに適切な署名を付けて同等の検証を行う）．

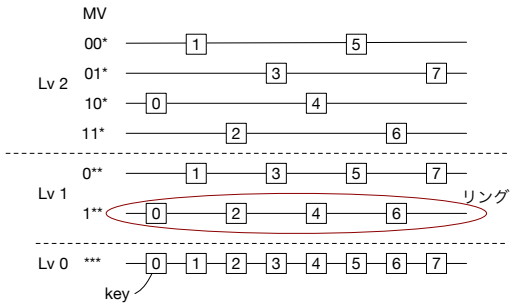


図 1　Skip Graph の構造

Authority は，1 ユーザが取得できる認証済参加情報の数に制限を加えることで，攻撃者が多数の faulty ノードを参加させることを困難にし，Sybil Attack を防ぐ．

2.2 オーバレイネットワークの構造

ベース KOPSON は Skip Graph[10] と同様の構造を持つ．Skip Graph は**図 1** に示す階層構造を持つ KOPSON である．階層構造の各レベルは，キーの順序に従ってソートされたノードからなる**リング**から構成されている． リングは環状であり，最大のキーを持つノードの正方向の隣接ノードは最小のキーを持つノードとなる（負方向も同様）．なお，以下右方向を正方向，左方向を負方向とする．各ノードは十分な桁数の α 進数の乱数（MV; Membership Vector）を生成する．レベル i $(i \geq 0)$ では，MV の先頭 i 桁と一致するノード同士がリングを形成する．このため，レベル i には最大 α^i 個のリングが存在する．なお，レベル 0 のリングは全てのノードが所属する．

ベース KOPSON では，オーバーレイネットワークの構造の決定に，MV ではなく Authority より与えられた TMV を用いる．これにより，faulty ノードは各レベルの隣接ノードを自由に選択することが出来ず，Eclipse Attack が困難になる．

2.3 経路表

ベース KOPSON では，冗長化に関するパラメータ k を用いる．k は，k 個のノードを無作為に選んだときに，少なくとも 1 つは正常なノードが含まれる確率が十分高くなるように選ぶ．

各ノード v は Skip Graph と同様，階層化された経路表を持つ．v の経路表のレベル i は，TMV が v の TMV と先頭 i 桁が一致するノードのうち，キーが範囲 $[u, w]$ に含まれるノードのリストである．ただし，u（あるいは w）は v からレベル i リングを左方向（あるいは右方向）にたどったときに，TMV が $i+1$ 桁一致するノードの $k-1$ 個目のノードのキーである．レベル i のノードを，u から w

©2021 Information Processing Society of Japan

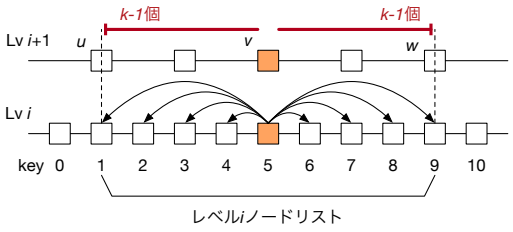


図 2 経路表の構造（$k = 3$）

の方向に並べたものを**レベル i ノードリスト**と呼び，U_i で表す．なお，U_i には v 自身も含まれる．例えば，**図 2** において $k = 3$，v をノード 5 とすると，u はノード 1，w はノード 9 となる．このため，v のレベル i ノードリストは，ノード 1 からノード 9 までのノードのリストとなる．

レベル i ノードリストの各要素（経路表エントリ）は，当該ノードのロケータ（IP アドレスなど），認証済参加情報，公開鍵から構成される．

レベルが上がるほど，リングに所属するノード数は減少する．一方，各ノードのレベル i ノードリストには，左方向，右方向とも少なくとも $k-1$ 個のノードが入る．このため，いずれ左方向，右方向でノードリスト内に重複が生じることになる．このレベルを経路表の最大レベル h とする．経路表の高さは $h+1$ である．

3. 提案手法

本稿で提案するマルチキャストアルゴリズムでは，2 章で述べたベース KOPSON の構造と経路表をそのまま利用し，その上でキーベースマルチキャストを行う．以下，特徴を述べる．

マルチキャストの範囲は，任意のキー a，b に対して区間 $[a, b)$ とし，この区間にキーが含まれるノードにメッセージを高確率で配送する．なお，$[50, 10)$ のような，ラップアラウンドした範囲も許容する．メッセージの到着順序は考慮しない．

提案手法は，上位レベルから下位レベルへ向かってマルチホップでメッセージを配送する．その際，「経路の冗長化」と「メッセージの検証」によって想定する攻撃に対処する．

3.1 経路の冗長化

提案手法では，あるノード v に対してメッセージを送信するノード（**上流ノード**と呼ぶ）を複数（k 個）にすることで配送経路を冗長化する．ここで k は 2.3 節で述べた冗長度のパラメータである．k 個のノードすべてが faulty である確率は十分低いため，v は高確率でメッセージを受信できる．1 つのマルチキャストに対して，各ノードは複数の上流ノードから同一のメッセージを受信する可能性があるため，マルチキャストに対して固有の ID（クエリ ID）を割

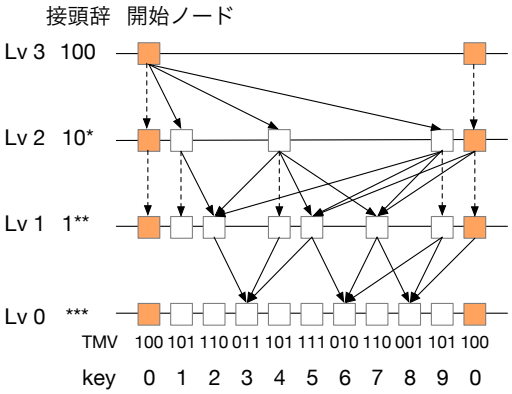


図 3 全域マルチキャストの経路例 ($k = 3$)

り当てる．各ノードは受信したメッセージのクエリ ID と開始ノードのキーのペアを記録し，同一ペアが付与されたメッセージを再び受信した場合はこれを無視する．クエリ ID と開始ノードのキーをペアで扱うのは，faulty ノードが正常なメッセージのクエリ ID を付与した偽のメッセージを拡散することで，正常なメッセージ配送が妨害されるのを防ぐためである．なお，クエリ ID と開始ノードのキーのペアには開始ノードによる署名を付与し，偽造を防ぐ（3.2 節参照）．

3.1.1 メッセージ転送の概要

マルチキャストの開始ノードから送信されたメッセージを，どのように冗長化しながら範囲内の各ノードに配送するかについて述べる．まず，単純な全域マルチキャスト（すべてのノードに対するマルチキャスト）について述べ，次にキーの範囲を限定した場合のマルチキャストについて述べる．ここでは，すべてのノードは正常ノードとする．

3.1.1.1 全域マルチキャスト

図 3 は，マルチキャストの開始ノードをノード 0 とし，ノード 0 が所属しているリングのみを示している（いまは矢印は無視すること）．すなわち，レベル i に含まれるノードの TMV はノード 0 の TMV（接頭辞は 100）と先頭 i 桁以上が一致する．各ノードはリング状に並んでいるため，ノード 9 の右方向の隣接ノードはノード 0 となる（ここではノード 0 を左右 2 箇所に示している）．

提案手法の基本的な考え方は，この図において，あるレベルで新しく現れるノード v に対し，1 つ上のレベルの k 個のノードから同一のメッセージを冗長に送信するというものである．

開始ノードは，経路表の最大レベル l のすべてのノードにメッセージを送信する．

レベル i (> 0) でメッセージを受信したノード u は，レベル $i-1$ のある範囲内（これをレベル $i-1$ の **forward range** と呼ぶ）で，レベル $i-1$ に新しく現れるノード v にメッセージを送信する．ここでレベル $i-1$ の forward range は，レベル i で u の左 k_L 個目のノードのキーから，右 k_R

©2021 Information Processing Society of Japan

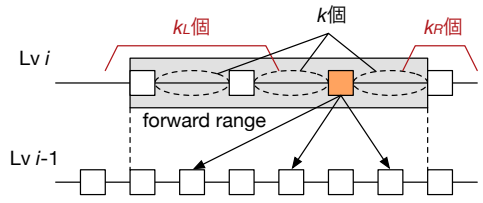


図 4 レベル i のノードが配送する範囲（$k=3$）

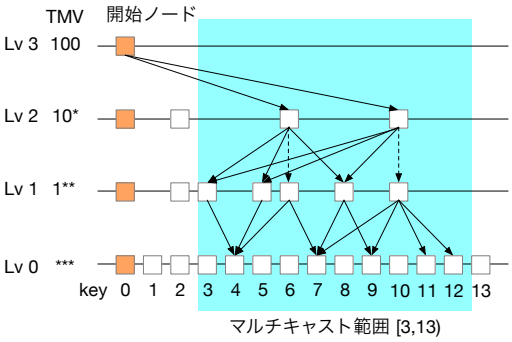


図 5 上流ノードが不足する例（$k=3$）

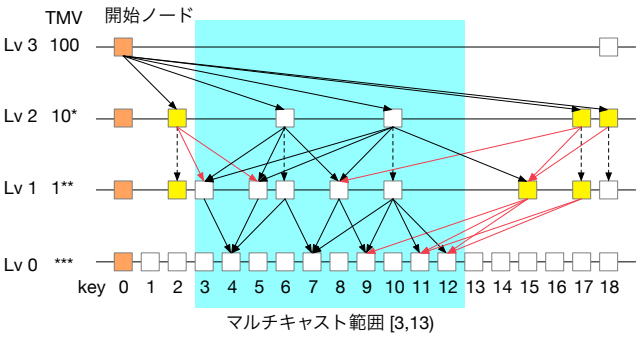


図 6 指定したキー範囲に対するマルチキャストの例（$k=3$）

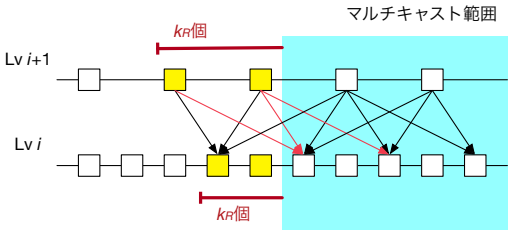


図 7 補助ノード（$k=4$）

個目のノードのキーの範囲である．ただし，$k_L=\lceil k/2\rceil$，$k_R=\lfloor k/2\rfloor$ とする．この範囲はレベル i のノードにより分割される k 個の区間を含むことに注意せよ（**図 4**）．レベル i の各ノードがこの方法でレベル $i-1$ で新しく現れるノードにメッセージを送信することで，レベル $i-1$ で新しく現れるノードは，k 個のノードから冗長にメッセージを受信することになる．u はレベル $i-1$ への送信を行なった後，レベル $i-2$ 以下のレベルでも同様に処理を行う（図 3 では破線で示している）．

図 3 の矢印は $k=3$，$k_L=2$，$k_R=1$ の場合のメッセージ配送経路を示している．レベル 1 でノード 5 がメッセージを受信した場合，レベル 0 の forward range は $[2,7]$ となる．同様にノード 7，ノード 9 のレベル 0 の forward range はそれぞれ $[4,9]$，$[5,0]$ となる．各ノードはこの範囲内で，レベル 0 で新しく現れるノードにメッセージを送信する．例えばノード 6 はこれらのノードから合計 $k=3$ 個のメッセージを受信することが確認できる．

3.1.1.2 指定したキー範囲に対するマルチキャスト

ここまで，マルチキャストの範囲を全域としてきたが，指定したキーの範囲内のノードにマルチキャストを行うように拡張することを考える（キーベースマルチキャスト）．単純には，上記の手法において，各ノードの forward range をマルチキャスト範囲内に限定することが考えられる（**図 5**）．

しかし，この方法では，マルチキャスト範囲の端に近いノード v では，v の上流ノードの一部または全部がマルチキャスト範囲に含まれない場合がある．例えば，図 5 のノード 3（初めて現れるレベルは 1）の上流ノードはノード 2，6，10 であるが，ノード 2 はマルチキャスト範囲外である．

そのような場合に対処するため，提案手法ではマルチキャスト範囲の外側のノードを利用する（**補助ノード**と呼ぶ）．マルチキャスト範囲内のノードと同様，補助ノードもメッセージの配送に関与する．**図 6** に例を示す．黄色のノードが補助ノードであり，赤色の線は補助ノードによるメッセージ送信を表す．

マルチキャスト範囲内のあるレベル i (≥ 0) の左端のノードでは，左側で最大 k_R 個の上流ノードが不足する可能性がある．このためレベル $i+1$ で，マルチキャスト範囲の左端より左側の k_R 個のノードを補助ノード（**左補助ノード**）とする（**図 7**）．同様に，マルチキャスト範囲の右端より右側の k_L 個のノードを補助ノード（**右補助ノード**）とする．レベル $i+1$ の補助ノードは，レベル i のマルチキャスト範囲内のノードの上流ノードとなる．

補助ノードへのメッセージ送信も k 個の上流ノードにより冗長化する必要がある．しかし，単純に forward range の限定範囲をマルチキャスト範囲の左 k_R 個先および右 k_L 個先のノードまで拡張するだけでは，補助ノードの上流ノードを k 個確保できない場合がある（例えば図 7 のレベル i の左端の補助ノードは，$k=4$ にも関わらず上流ノードが 3 個しか存在しない）．このため，次のように forward range を拡張する．**図 8** で説明する．図は，レベル $i+1$ のマルチキャスト範囲左端付近のノードがレベル i のノードにメッセージを送信する場合を示している（$k=4, k_L=k_R=2$）．レベル i の左補助ノードが k 個の上流ノードを確保するために，レベル $i+1$ の左補助ノードに加え，レベル $i+1$ のマルチキャスト範囲内で，左端から k_L 個目までのノード u がレベル i の補助ノードに送信する（正確には，u がレ

©2021 Information Processing Society of Japan

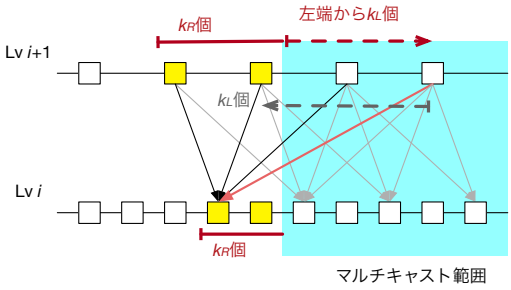


図 8 補助ノードへのメッセージ送信 ($k = 4$)

ベル $i+1$ でマルチキャスト範囲の左端から j ($\leq k_L$) 番目のノードである場合，u はレベル $i+1$ の左 $k_R+(j-1)$ 個目のノードからマルチキャスト範囲左端までの範囲に存在する，レベル i の補助ノードに送信する）．右端側についても同様である．図 8 でレベル $i+1$ のノードから伸びている赤い矢印は，上記のルールによって新たに追加される経路を示す．2.3 節で述べたように，各ノードはレベル i において，レベル $i+1$ で左右両方向に $k-1$ 個先のノードまでのポインタを保持しているため，拡大された forward range は経路表の範囲内に収まる．

3.1.2 開始レベル

ここで，マルチキャストの開始ノードが最初にメッセージを送信するレベル（開始レベル）について述べる．

開始レベル l は，開始ノードの経路表において，条件（レベル l の最左ノードから最右ノードまでの範囲がマルチキャスト範囲と補助ノードをカバーする）を満たす最も下のレベルである．なお，経路表の最上位レベルはすべてのキー範囲をカバーしているため，上の条件を満たす． 開始レベルでは補助ノードのみが存在し，マルチキャスト範囲内にはノードが存在しない場合もある．各レベルはリング構造となっているため， 左右の補助ノードが同一のノードとなる場合や，補助ノードが存在しない場合（あるレベルのすべてのノードがマルチキャスト範囲に含まれる場合）があるが，このようなレベルも開始レベルとする．**図 9** ではオレンジのノードが開始ノードであり，矢印によって経路表に含まれるノードを示している．レベル i では補助ノードをカバーしていないが，レベル $i+1$ で補助ノードをカバーするため，この場合は $i+1$ を開始レベルとする（図では開始ノードがマルチキャスト範囲内に存在する場合を示しているが，範囲外の場合でも同様）．

3.1.3 特殊な場合

上位のレベルでは，**図 10**(a) のように左右の補助ノードが重複したり，(b) のように全く存在しない場合がある．(a) の場合，重複した補助ノードは左右両方の補助ノードとして送信する．(b) の場合は，マルチキャスト範囲内の右端（あるいは左端）に近いノードが，左端（あるいは右端）の補助ノードの役割を果たす．

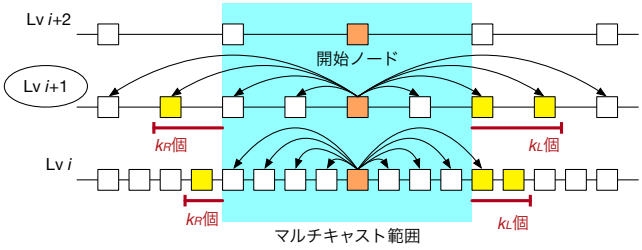
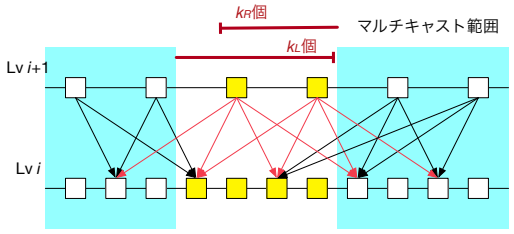
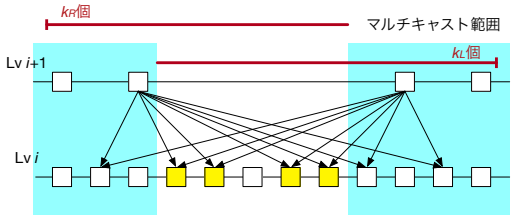


図 9 マルチキャスト開始レベル（$k = 3$）



(a) 補助ノードが重複する場合



(b) 補助ノードが存在しない場合

図 10 特殊な場合（$k = 4$）

3.2 メッセージの検証

マルチキャストの開始ノードを s とする．s は以下の情報をメッセージ m_s に格納する（**図 11** 参照）．

- メッセージ種別（マルチキャストメッセージを表す定数：multicastOp）
- マルチキャスト範囲 r
- クエリ ID q
- 開始レベル l
- s の認証済参加情報
- s の公開鍵
- アプリケーションで使用するペイロード
- 以上の情報（被署名部分）に対する s の秘密鍵による署名

m_s の受信ノード v は，s の認証済参加情報と s の公開鍵に Authority の署名があること，被署名部分に対する s の署名を検証する．

v が別のノード w にメッセージ m'_s を送信するときは，以下の情報をメッセージに格納する．

- m_s
- 転送レベル l
- v の認証済参加情報
- v の公開鍵
- 以上の情報（被署名部分）に対する v の秘密鍵による

©2021 Information Processing Society of Japan

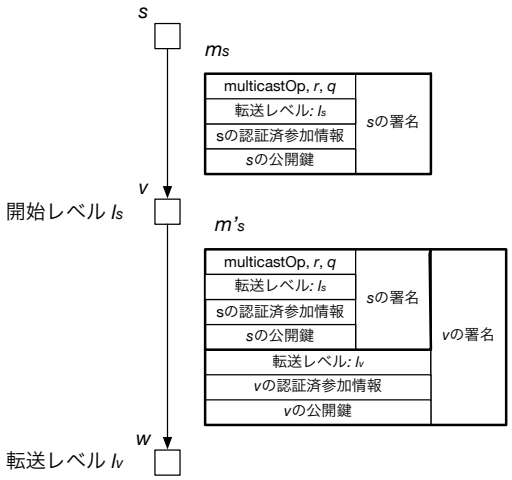


図 11 メッセージの構造

署名

m'_s を受信した w は，v と同様 m_s の検証を行う．また同様に v の認証済参加情報と v の公開鍵に対する Authority の署名，および被署名部分に対する v の署名を検証する．さらに，以下の検証を行う．

- 転送レベル l が，v と w の TMV の先頭一致桁数に等しいこと．これにより l の改ざんを防止する．なお，s から直接受信するノード（1 ホップ目のノード）では，この条件は成立しないためこのチェックは行わない．
- w が，マルチキャスト範囲 r 内に含まれる，あるいはレベル l における補助ノードになっていること．これは w の経路表を参照することで検証可能である．これにより，マルチキャストと無関係なノードに送信する Random Next Hop 攻撃を防止する．

faulty ノードが不正を検出されずに送信できる相手ノードは，あるレベル i のマルチキャスト範囲内あるいは補助ノードで，faulty ノードと TMV が i 桁一致するノードに限られる．このようなノードは本来別の経路で同じマルチキャストメッセージを受信するノードであり，また，m_s 部分は改ざんできないことから faulty ノードからメッセージを受信しても配送に失敗することはない．

3.3 アルゴリズムの詳細

ここでは，配送アルゴリズムの詳細を述べる．メッセージの署名や検証処理については割愛する．

3.3.1 マルチキャスト開始処理

ノード s が範囲 r に対するマルチキャストを開始する処理を**図 12** に示す．5 行目の if 文では，s のレベル l ノードリストが，r と補助ノードを加えた範囲をカバーする，もしくは l が最上位レベルかどうかを判定している．($s.U_l[-k_L]$ は，s のレベル l ノードリストの末尾から k_L 番目の要素を表す)．

extended_range(r, l) は，$l = 0$ の場合はマルチキャスト範囲 r を返し，$l > 0$ の場合は経路表のレベル l で見えている範囲で，r をレベル l の補助ノードまで拡大した範囲（l の最左補助ノードから最右補助ノードの範囲）を返す関数である．ただし，最左（あるいは最右）補助ノードがレベル l の経路表に載っていない（見えていない）場合は，レベル l の最左（あるいは最右）ノードで代替する．extended_range(r, l) の詳細は割愛する．

```
1  // r: multicast range
2  s.start_multicast(r) {
3    // h: max level
4    for l ← 0 to h {
5      if (r ⊆ (s.U_l[k_R], s.U_l[−k_L]) ∨ (l = h − 1)) {
6        start_level ← l
7        break
8      }
9    }
10   q = random() // query ID
11   range ← extended_range(r, start_level)
12   foreach x ∈ s.U_start_level {
13     if (x ∈ range) {
14       send ⟨multicastOp, r, start_level, q, s⟩ to x
15     }
16   }
17 }
```

図 12 マルチキャストの開始ノード s における処理

```
1  // r: multicast range
2  // l: received level
3  // q: query ID
4  // s: key of source node
5  upon receiving ⟨multicastOp, r, l, q, s⟩ by v:
6    if ((q, s) is already seen) {
7      return // ignore duplicated messages
8    }
9    register (q, s) as an already seen message
10   do {
11     recurse = false
12     if (l = 0) {
13       send the message to app
14       return
15     }
16     // list of ranges that v should send to
17     ranges ← forward_range(r, l − 1)
18     foreach x ∈ v.U_{l−1} {
19       if ((one of ranges contains x) ∧
20         common_prefix_length(v.tmv, x.tmv) = l − 1) {
21         if (x = v) {
22           recurse = true
23         } else {
24           send ⟨multicastOp, r, l − 1, q, s⟩ to x
25         }
26       }
27     }
28     l ← l − 1
29   } while (recurse)
```

図 13 ノード v におけるメッセージ転送処理

3.3.2 メッセージ転送処理

マルチキャストメッセージを受信したノード v の処理を**図 13** に示す．

宛先のレベル l が 0 ならば，アプリケーションにメッセージを渡す．そうでなければ，次に述べる forward_range 関数によってレベル $l-1$ における forward range を求め，レベル $l-1$ で新しく現れるノードにメッセージを送信する．ただし，送信先ノードに v 自身が含まれる場合は，レベルを 1 つ下げて同じ処理を再帰的に繰り返す．

forward_range(r, i) は，マルチキャスト範囲 r に対して，レベル i の forward range を求める関数である．この際，

©2021 Information Processing Society of Japan

3.1.1.2 節および 3.1.3 節で述べた条件を考慮する．

(1) レベル $i+1$ ノードリスト U_{i+1} に基づいて，範囲 $U=[u_1,u_2]$ を求める．u_1 は，レベル $i+1$ で r の左端から j ($\leq k_L$) 個目までのノードでは，U_i の中で v から左 $k_R+(j-1)$ 個目のノードのキー，それ以外の場合は左 k_L 個目のノードのキーとする．また，u_2 は，r の右端から j ($\leq k_R$) 個目までのノードでは，U_i の中で v から右 $k_L+(j-1)$ 個目のノードのキー，それ以外の場合は右 k_R 個目のノードのキーとする．

(2) 範囲 $L=\text{extended_range}(r,i)$ とする（3.3.1 節参照）．

(3) U と L の共通部分が求める範囲である．図 10(b) のような場合，共通部分は 2 つの範囲に分割される．

forward_range 関数の詳細は割愛する．

4. 考察

4.1 最大ホップ数

マルチキャスト範囲内の各ノード v は複数のメッセージを受信するが，その中で最小のホップ数を v のホップ数とし，範囲内のすべてのノードにおけるホップ数の最大値を，マルチキャストの最大ホップ数とする．

提案手法では，転送レベルはマルチキャスト開始レベルから 1 ホップごとに少なくとも 1 下がる．そのため，最大ホップ数は最悪の場合でマルチキャスト開始レベル +1 となる（+1 は開始ノードから送信する最初のメッセージの分）．

文献 [3] で示した経路表の最大レベルの期待値（h）から，（マルチキャスト範囲が全域となる場合など）転送レベルが最大レベルから始まる場合の，最大ホップ数の最悪値は以下の式で近似できる（n は全ノード数）．

$$h+1 \approx \log_{\alpha} \frac{n}{2(k-1)} \tag{1}$$

なお，この式は上界を与えるもので，実際の最大ホップ数はこれより小さくなる場合が多い．これについては 5.1 節で述べる．

4.2 メッセージ数

faulty ノードが存在しない環境における，マルチキャストに要するメッセージ数について考察する．マルチキャスト範囲内の各ノードと各レベルの補助ノードは基本的に k 個のメッセージを受信する．このうち，補助ノードに送信するメッセージがオーバーヘッドとなる．また，開始ノードから直接メッセージを受信するノードは（k 個ではなく）1 メッセージのみ受信するが，これはメッセージ数を減らす要因となる．トータルでのメッセージ数はシミュレーションによる評価（5.1 節）に委ねる．

4.3 メッセージ到達率

メッセージ到達率（マルチキャスト範囲内の正常ノードにおける平均メッセージ受信確率）について考察する．

マルチキャスト実行時に，各ノードは確率 f で故障しているとする．ここではストップ故障を想定する（下流ノードへのメッセージ送信が行われる保証がないという点で，ビザンチン故障と同一視できる）．開始ノードは故障していないものとし，また，通信路の故障確率は無視できるものとする．開始ノードから i ホップ目でメッセージを受信するノードを v_i，v_i がメッセージを受信する確率を p_i とする．

前提より $p_1=1$ である．また，v_2 の k 個の上流ノードのうち，どれか 1 つが正常であれば v_2 はメッセージを受信できる．したがって，$p_2=1-f^k$ である．同様に考えると

$$p_i=(1-f^k)^{i-1} \tag{2}$$

となる．最悪の場合のメッセージ到達率は，ホップ数が最大のときである．最大ホップ数は経路表の高さ $h+1$ と等しいため，このときのメッセージ到達率は

$$p_{h+1}=(1-f^k)^h \tag{3}$$

となる．

故障率が高い環境においても，k を大きくすることで高いメッセージ到達率が得られる．ただし，マルチキャストに要するメッセージ数は増加する．また，経路表サイズが増加するため（詳細は文献 [3] 参照），ノードの参加・離脱などのコストが増大するというデメリットもある．

5. 評価

提案手法をシミュレーションによって評価した．ここでは $\alpha=2$ とした．

5.1 最大ホップ数

以下の条件で最大ホップ数の平均値を求めた；$k=2,4,6$ について n を 100 から 3,200 まで変化させてオーバーレイネットワークを生成し，マルチキャストを 300 回実行した．開始ノードは乱数で決定した．最大ホップ数はマルチキャスト範囲の大きさに依存し，範囲が全域（ブロードキャスト）となる場合に最大となるため，マルチキャスト範囲は全域とした．オーバーレイネットワークは 5 つ生成し，最大ホップ数の平均値を求めた．各ノードは故障しないものとした．

結果を**図 14** に示す．破線は最悪の場合の最大ホップ数の理論値（式 (1)）である．グラフから最大ホップ数は対数オーダで増加することが確認できる．

最大ホップ数の実測値は式 (1) よりも小さい．これは，式 (1) はレベル l でメッセージを受信したノードはレベル $l-1$ のみで配送する前提としているのに対し，実際のアルゴリズムではレベル $l-1$ からレベル 0 までのレベルで配送するためである．これらのレベルの近傍ノードにはあ

©2021 Information Processing Society of Japan

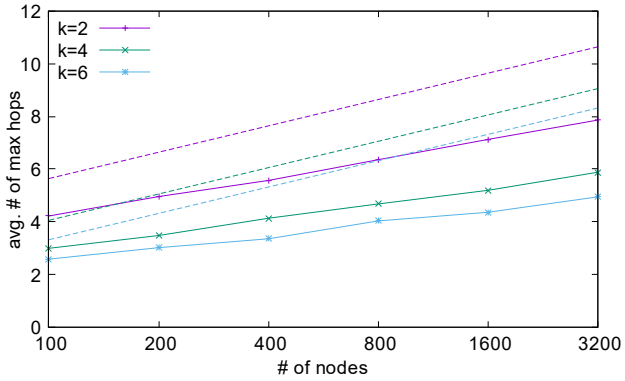


図 14 最大ホップ数（$\alpha = 2$，$n_r/n = 1$）

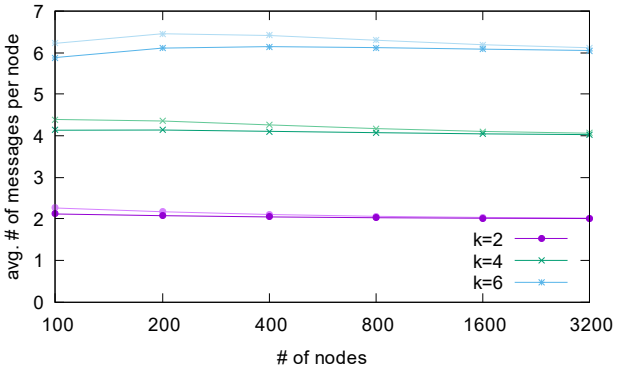


図 15 メッセージ数（マルチキャスト範囲内の 1 ノードあたり．$\alpha = 2$，$n_r/n = 0.5$）

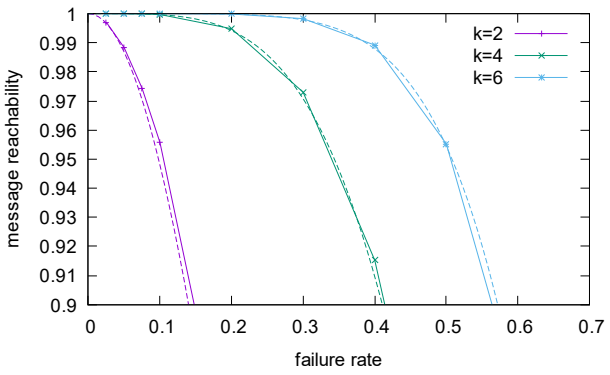


図 16 メッセージ到達率（$\alpha = 2$，$n_r/n = 0.5$）

と 1 ホップで到達できる．k が大きいほど各レベルで送信するメッセージ数は増加するため，最大ホップ数は式（1）に比べて小さくなる．

5.2 メッセージ数

マルチキャストに要するメッセージ数を測定した．5.1 節と同様の実験を行い（ただし，マルチキャスト範囲の幅は $n_r/n = 0.25$ および 0.5 となるように乱数で決定した．ここで n_r はマルチキャスト範囲内のノード数），補助ノードを含めたすべてのノードが受信するメッセージの総数を n_r で割ることで，範囲内の 1 ノードあたりの平均メッセージ数を求めた．結果を**図 15** に示す．薄い色のプロットは $n_r/n = 0.25$，濃い色のプロットは $n_r/n = 0.5$ の場合の結果である．

グラフより，範囲内の 1 ノードあたりの受信メッセージ数は n によらず概ね k 個となることが確認できる．$n_r/n = 0.25$ の結果が $n_r/n = 0.5$ の結果を上回っているのは，n_r が減少することで補助ノードが送受信するメッセージ数の影響が相対的に大きくなったためと考えられる．$n_r/n = 0.5$，$n = 100$ のとき，$k = 2$ および $k = 4$ では平均メッセージ数は k をわずかに上回っているが，$k = 6$ では下回っている．これは，k が大きくなるにつれて開始ノードが直接送信するノード数が増加し，補助ノードによるオーバーヘッドを打ち消すためと考えられる（4.2 節参照）．いずれのプロットでもマルチキャスト範囲が大きくなるにつれてメッセージ数は k に収束している．これは，範囲が大きくなるにつれて補助ノードのオーバーヘッドおよび開始ノードが直接送信することによる削減効果が無視できるようになるためと考えられる．

5.3 メッセージ到達率

メッセージ到達率をモンテカルロ法により測定した．$k = 2, 4, 6$ について，$n = 1{,}000$ のオーバレイネットワークを 5 つ生成し，その上でマルチキャストを 100 回ずつ行った．マルチキャストの範囲は $n_r/n = 0.25$ および $n_r/n = 0.5$ となるように乱数で決定した．マルチキャストの際，配送経路を示すグラフを作成し，このグラフを用いて，配送経路に含まれる各ノード（ただし始点ノードを除く）を故障率 f に従ってランダムに故障させたときの，各ノードのメッセージ到達率（全試行に対するメッセージを受信した回数の割合）の平均値を求めた．モンテカルロ法の試行回数は 1,000 回とした．

結果は，$n_r/n = 0.25$ の場合と 0.5 の場合でほぼ同一となったため，ここでは $n_r/n = 0.5$ の結果のみを示す（**図 16**）．破線は，予備実験で求めた $n_r/n = 0.5$ における最大ホップ数の平均値を用いて式 (2) より求めたメッセージ到達率の理論値である．実測値は理論値と概ね等しいか，若干上回った．上回る理由は，実際のホップ数が理論値の計算で用いた最大ホップ数の平均値より小さい場合が多いためである．図より，故障率が高い環境でも k の値を大きくすることにより，高いメッセージ到達率が得られることが確認できる．

6. おわりに

本稿では，ビザンチン障害耐性（BFT）を備えるキー順序保存型構造化オーバレイネットワークのためのマルチキャスト手法を提案した．提案手法は，マルチキャスト範囲内の各ノードに k 個の異なるノードがメッセージを送信することで経路を冗長化し，またメッセージを検証するこ

©2021 Information Processing Society of Japan

とで，BFT を備えたマルチキャストを実現する．実験により，ノードの故障率が高い環境でも k を大きくすることで高いメッセージ到達率が得られることを示した．

提案した冗長化されたキーベースマルチキャスト手法は，一般的なストップ障害に対する対策としても有効である．既存の手法では上流ノードが 1 つしかないため，あるノードの上流ノードが停止すると，下流ノードすべてに影響が及ぶ．この場合，オーバーレイネットワークを修復して故障した上流ノードを取り除いた後，マルチキャストを再送するといった対策が必要であり，配送遅延が大きくなるが，提案手法ではこのような場合も高確率で再送することなくメッセージを配送できる．

謝辞 本研究は JSPS 科研費 JP20H04186 の助成を受けたものである．ここに記して謝意を表す．

参考文献

[1] Banno, R., Takeuchi, S., Takemoto, M., Kawano, T., Kambayashi, T. and Matsuo, M.: 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).

[2] Shao, X., Jibiki, M., Teranishi, Y. and Nishinaga, N.: An efficient load-balancing mechanism for heterogeneous range-queriable cloud storage, *Future Generation Computer Systems*, Vol. 78, No. 3, pp. 920–930 (2018).

[3] 寺西裕一，秋山豊和，安倍広多：ビザンチン障害耐性を備えるキー順序保存型構造化オーバーレイネットワークの実現に向けて，情報処理学会研究報告，Vol. 2021-DPS-186, No. 54, pp. 1–8 (2021).

[4] Coelho, P., Junior, T., Bessani, A., Dotti, F. and Pedone, F.: Byzantine Fault-Tolerant Atomic Multicast, *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 39–50 (2018).

[5] Kozhaya, D., Decouchant, J. and Esteves-Verissimo, P.: RT-ByzCast: Byzantine-Resilient Real-Time Reliable Broadcast, *IEEE Transactions on Computers*, Vol. 68, No. 03, pp. 440–454 (2019).

[6] Rohrer, E. and Tschorsch, F.: Kadcast: A Structured Approach to Broadcast in Blockchain Networks, *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pp. 199–213 (2019).

[7] González-Beltrán, A., Milligan, P. and Sage, P.: Range queries over skip tree graphs, *Computer Commun.*, Vol. 31, No. 2, pp. 358–374 (2008).

[8] Banno, R. and Shudo, K.: An Efficient Routing Method for Range Queries in Skip Graph, *IEICE TRANSACTIONS on Information and Systems*, Vol. E103–D, No. 3, pp. 516–525 (online), DOI: 10.1587/transinf.2019FCP0008 (2020).

[9] Young, M., Kate, A., Goldberg, I. and Karsten, M.: Towards Practical Communication in Byzantine-Resistant DHTs, *IEEE/ACM Transactions on Networking*, Vol. 21, No. 1, pp. 190–203 (2013).

[10] Aspnes, J. and Shah, G.: Skip graphs, *ACM Trans. on Algorithms*, Vol. 3, No. 4, pp. 1–25 (2007).

©2021 Information Processing Society of Japan