モバイルアドホックネットワークにおける クラスタを用いたTop-k検索のためのルーティング手法

天方 大地[1,a)] 佐々木 勇和[1,b)] 原 隆浩[1,c)] 西尾 章治郎[1,d)]

概要：ユーザが指定する検索条件に基づいてデータのスコアを決定し，上位 k 個のスコアをもつデータを検索する Top-k 検索への関心が高まっている．本稿では，モバイルアドホックネットワークにおいて，検索結果の取得に必要な端末のみで Top-k 検索を行うことを目指し，クラスタを用いた Top-k 検索のためのルーティング手法，CTR(Cluster-based Top-k query Routing) を提案する．CTR では，スコアが大きいデータをもつ端末がクラスタヘッドとなるクラスタリングを行い，複数のクラスタに属するノード（ゲートウェイノード）を介してクラスタヘッド間で検索クエリのルーティングを行う．各クラスタヘッドは，スコアが大きいデータまでのホップ数を管理し，自身が検索する必要のあるデータを自律的に判断する．これにより，取得精度を維持しつつ，不要な検索クエリの転送を抑止する．シミュレーション実験の結果から，提案手法は，高い取得精度を維持しつつ，低オーバヘッド，および低遅延を達成していることを確認した．

A Cluster-based Routing Method for Top-k Query Processing in Mobile Ad Hoc Networks

Daichi Amagata[1,a)] Yuya Sasaki[1,b)] Takahiro Hara[1,c)] Shojiro Nishio[1,d)]

1. はじめに

近年，ルータ機能をもつ移動端末のみで一時的な無線ネットワークを形成するモバイルアドホックネットワーク（*Mobile Ad hoc Networks*，以下 MANET）への関心が高まっており，災害地における救助活動などの様々な分散アプリケーションへの応用が期待されている．複数の端末が限られた通信帯域を共有する MANET では，必要なデータのみを効率的に取得する必要がある．その場合，ユーザが指定する検索条件に基づいてデータのスコアを決定し，上位 k 個のデータを検索する Top-k 検索が有効である．Top-k 検索方法として，まず，ネットワーク内の全てのデータを 1 台の端末が収集し，その端末にアクセスすることでデータを取得する中央処理型の方法 [1] が考えられる．しかし，データの更新が起きた際にデータを転送するオーバヘッドや，1 台の端末にのみへのトラヒックの集中に伴う頻繁なパケットロスにより，この方法は MANET では有効ではない．分散環境において効率的に Top-k 検索を行う手法が，P2P ネットワーク [11] や無線センサネットワーク [12] の分野において多数提案されている．しかし，P2P ネットワークでは，ネットワーク内の全ての端末と直接通信可能である点や，固定ネットワークを想定している点から，端末が移動し，動的にネットワークトポロジが変化する MANET には，これらの手法を単純に適応することはできない．一方，無線センサネットワークは，無線通信や，端末のバッテリ制限など，MANET と類似した特徴をもつ．そこで，消費電力や帯域の圧迫といった問題を解決するため，Top-k 検索を行う際，検索結果の取得に必要な端末のみでメッセージ処理を行うことが有効である．しかし，無線センサネットワークでは，固定端末，および 1 台のシンクを想定しているため，P2P ネットワークにおける手法と同様に，単純に MANET に適応することはできない．したがって，低トラヒックで，アクセスする必要がある端末のみにメッセージを送信すること，および端末の移動に対して動的に適応することが重要である．

そこで本稿では，検索結果の取得に必要な端末で Top-k 検索を行うことを目指し，クラスタを用いた Top-k 検索のためのルーティング手法 CTR(Cluster-based Top-k query

[1] 大阪大学大学院情報科学研究科マルチメディア工学専攻
Department of Multimedia Engineering Graduate School of Information Science and Technology Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
[a)] amagata.daichi@ist.osaka-u.ac.jp
[b)] sasaki.yuya@ist.osaka-u.ac.jp
[c)] hara@ist.osaka-u.ac.jp
[d)] nishio@ist.osaka-u.ac.jp

Routing) を提案する．CTR では，スコアが大きいデータをもつ端末がクラスタヘッド（*ClusterHead*，以下 CH）となるようにクラスタリングを行う．CH 間での検索クエリのルーティングを行うため，検索クエリは複数のクラスタに属しているゲートウェイノード（*GateWay-node*，以下 GW）を中継して送信される．CH は，スコアが大きいデータまでのホップ数を管理し，自身が検索すべきデータを自律的に判断する．これにより，取得精度を維持しつつ，不要な検索クエリの転送を抑制できる．シミュレーション実験の結果から，CTR は高取得精度を維持しつつ，低トラヒック，および低遅延を達成していることを確認した．

以下では，2 章で関連研究について述べ，3 章で想定環境について説明する．4 章で提案手法について説明し，5 章でシミュレーション実験の結果を示す．最後に 6 章で本稿の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，様々なネットワーク環境におけるデータルーティング手法，および Top-k 検索手法における代表的な既存手法を紹介する．

2.1 データルーティング手法

文献 [6] で提案されている手法では，無線センサネットワークにおいて，ネットワーク内で発生したイベント（データ）へのアクセス確率をブルームフィルタを用いて管理する．各端末は，隣接端末のもつブルームフィルタも保持し，データを要求する際，要求するデータへのアクセス確率が最大である隣接端末を，ブルームフィルタを用いて 1 台選択し，その端末にメッセージを送信する．これにより，ネットワーク内の全端末にアクセスすることなく，小さいオーバヘッドでデータを取得できる．しかし MANET において，常に全ての隣接端末の情報を管理するためには，全ての端末が定期的にメッセージを送信する必要があり，オーバヘッドが非常に大きくなる．

文献 [7] では MANET において，データパケットのマルチキャストルーティング手法を提案している．データを要求する端末は，要求するデータをもつ端末までのメッセージ転送領域を設定しながらメッセージを転送し，自身から n ホップ以内に存在する端末をメッシュとよばれる役割を担う端末に指定する．メッシュ端末は，初めてデータパケットを受信した際，これをブロードキャストにより転送することで，トポロジ変化に対応し，データ要求端末までのデータパケットの転送率を向上する．本稿における提案手法では，Top-k 検索中にリンク切断が起きた場合においてもメッセージの転送率を維持するため，このアイデアを採用し，クラスタ内に存在する端末をメッシュ端末と見立てたメッセージの転送を行う．

2.2 Top-k 検索

様々なネットワーク環境において Top-k 検索手法が提案されている．ここでは，P2P ネットワーク，無線センサネットワークにおける代表的な Top-k 検索手法，および MANET における Top-k 検索手法について紹介する．

文献 [11] では，P2P ネットワークにおいて，代表ピアを用いた Top-k 検索手法を提案している．代表ピアは，論理リンクが存在するピアのもつデータの情報，および検索結果に含まれる可能性のあるデータの複製を管理する．また，ネットワーク内の代表ピアとデータ情報を交換することにより，検索結果に入るデータをもつ代表ピアを把握する．Top-k 検索の際は，それらの代表ピアにアクセスし，上位 k 個のデータを取得する．この手法は，任意の代表ピア間は直接通信可能であると想定している点で MANET への適応が難しい．また，固定端末を想定している点で MANET とは異なる．文献 [12] では，無線センサワークにおいて，通信量削減によりネットワークの高寿命化を目的とした FILA とよばれる手法を提案している．この手法では，各端末に上位 k 個に含まれるデータに関するフィルタを配布する．データの更新が起きた際，ネットワーク内のデータの k 番目までの順位に影響を及ぼすかどうかをフィルタを用いて確認し，影響がある場合，シンクまでデータの更新を通知する．しかし，この手法では，固定端末，およびネットワーク内で 1 台のシンクを想定している点で MANET とは異なる．

筆者らの知る限りでは，MANET における Top-k 検索手法は，文献 [5], [8], [9], [10] でのみ提案されている．文献 [8] での想定環境は，本稿で想定する環境と本質的に異なる．文献 [5], [9], [10] では，ネットワーク内のデータの k 番目のスコアを推定することにより，検索結果に含まれないデータの返信を抑制している．しかし，これらの手法では，ネットワーク内のすべての端末が検索クエリ，およびクエリ応答を送信しており，検索結果の取得に必要のない端末によるメッセージ転送が多く発生してしまう．そのため，不要なトラヒックの増加によるパケットロスや，検索時間の遅延増大などの問題が生じる．

そこで，筆者らは文献 [2], [3] において Top-k 検索のためのルーティング手法を提案した．これらの手法では，経路表を用いてネットワーク内の上位のスコアをもつデータ，および検索クエリの転送先（宛先）を把握する．Top-k 検索を行う端末は，経路表を用いて，上位 k 個のデータを取得するために必要な端末にのみ，メッセージを送信する．文献 [2] では，検索クエリをユニキャスト（単一経路）で転送する手法を提案した．この手法を拡張し，文献 [3] では，検索クエリをマルチキャスト（複数経路）で転送する手法を提案した．しかし，これらの手法では，Top-k 検索中にのみ経路表を修正していたため，トポロジ変化が激しい環境では，経路表における宛先端末とのリンク切断が頻繁に発生する．これにより，経路表修正のためのトラヒックが大きくなるため，パケットロスが頻繁に発生し，経路表の精度，および取得精度を維持できない．CTR では，CH のみが定期的にメッセージを送信するため，トラヒックを抑制しつつ，各端末は自身の属するクラスタを把握でき，トポロジ変化に大きく依存しない．また，メッセージの送信先となる端末とのリンク切断が生じた場合にも，経路探索を行わずにメッセージを転送するため，低オーバヘッド，および低遅延な Top-k 検索を実現できる．

3. 想定環境

本稿では，MANET を構成する各端末が，自身と他の端

末のもつデータに対して Top-k 検索を行う環境を想定する．ネットワーク内には，m 台の端末が存在し，各々が自由に移動する．また，ネットワーク内には n 個のデータが存在し，各々が特定の端末に保持されている．簡単化のため，全てのデータのサイズは等しく，各端末は複製を作成しないものとする．データのスコアは，検索条件とデータの属性値から決定し，何らかのスコアリング関数を用いて算出される．Top-k 検索を行う端末は，検索条件，および要求データ数 k を指定して検索クエリを発行し，ネットワーク内の上位 k 個のスコアをもつデータを取得することを目的とする．各端末は指定される最大の k の値（k_{max}）を把握しているものとする．

4. 提案手法

本章では，本稿における提案手法 CTR を説明する．まず，概要について述べ，その後，クラスタの構築，Top-k 検索，およびスコア更新時におけるメッセージ処理方法について説明する．

4.1 概要

提案手法では，まずネットワーク内のデータ情報を収集し，データの順位表を作成する．これをネットワーク内の端末間で共有することにより，自身のもつデータの順位を把握する．また，スコアが高いデータをもつ端末から順に CH となる 1 ホップクラスタリングを行う．これにより，上位データをもつ端末が CH となり，CH 間で検索クエリのルーティングを行うことで，検索結果を取得するために必要な端末のみによる Top-k 検索を実現する．1 ホップクラスタでは，クラスタ内の端末は CH とのリンクのみを把握するだけでよいため，文献 [2], [3] のように，多数の隣接端末とのリンクを管理する必要がなく，トポロジ変化に対して寛容になる．また，CH は一定時間ごとにメッセージ（$CHAnnounce$，以下 CHA）を送信し，これを受信した端末は自身の属するクラスタを確認できる．さらに，端末の移動によるトポロジ変化に対応するため，定期的にクラスタの再構築を行う．

Top-k 検索では，GW が CH の送信する検索クエリを転送することにより，CH 間のメッセージ転送を行う．CH は上位データまでのホップ数を管理し，検索クエリ送信元の CH よりもホップ数が少ないデータを検索することにより，検索クエリの受信端末が検索すべきデータを自律的に判断する．これにより，検索クエリの不要な転送を抑制する．また，文献 [2], [3] で提案した手法のように，検索クエリの転送先を一意に決定しないため，上位データまでの経路が固定されず，トポロジ変化に対応しやすい．さらに，複数の隣接 CH から検索クエリを受信できるため，単一経路でメッセージが転送される場合よりもメッセージの転送率が向上する．Top-k 検索中に CH とのリンク切断を検出した GW は，メッセージ（検索クエリ，またはクエリ応答）をブロードキャストし，リンク切断先のクラスタに存在する端末が CH にメッセージを転送することにより，メッセージの転送失敗を防ぐ．また，これにより，文献 [2], [3] のような経路探索を行う必要がなく，メッセージの転送遅延を抑制できる．

表 1 順位表の例
Table 1 Example of a ranking table

順位	スコア	保持端末
1	100	A
2	95	M
3	90	H
4	88	O
5	85	W
6	80	F
7	77	X
⋮	⋮	⋮
30	20	A

データのスコア更新が起きた場合には，小さいトラヒックでネットワーク内の全端末に通知を行うため，スコア更新メッセージを各 CH に転送し，このメッセージを受信した CH がスコア更新メッセージをブロードキャストする．これを受信した端末は，順位表を更新し，データの正確な順位を把握する．

4.2 クラスタの構築

本節では，検索結果の取得に必要な端末のみによる Top-k 検索を実現するためのクラスタリングについて説明する．

文献 [2], [3] と同様の方法により，ネットワーク内で初めて Top-k 検索を行う端末は，順位表（表 1 参照）を作成する．この順位表をフラッディングによりネットワーク全体に送信し，データの情報を全ての端末で共有する．

順位表を受信した，k_{max} 位以内のデータをもつ端末（$data\ holder$）は，データの順位が高いほど発火が早くなるようにタイマを設定する．タイマが発火した場合，CHA を送信する．CHA を受信した端末は，自身のクラスタヘッドリスト（$List_{CH}$）に，CH の識別子を格納する．ここで，タイマを設定している端末が，タイマの発火前に CHA を受信した場合，タイマを破棄する．一定時間内に CHA を受信しなかった data holder 以外の端末は，タイマを設定（端末識別子が小さいほど発火が早い）し，上記と同じ手順を行う．これにより，上位データをもつ端末が基本的に CH となるため，検索結果に含まれるデータをもつ端末間による検索クエリのルーティングを実現できる．CH は，一定時間（t 秒）ごとに CHA を送信することにより，各端末は自身の所属するクラスタの CH を把握できる．また，MANET では端末が自由に移動するため，ネットワークトポロジが動的に変化する．そのため，ネットワーク内の端末は，$(\alpha \times t)$ 秒ごとに上記と同様の操作を行うことにより，クラスタの再構築を行い，ネットワークトポロジの変化に対応する．

各端末が表 1 で表されるデータをもっている場合において，図 1 を用いてクラスタを構築する例を説明する．青色の丸は CH を示し，桃色の丸は data holder を示す．灰色の丸はそれ以外の端末を示す．また，点線の丸は，CH を中心とするクラスタの様子を示す．提案手法では，高順位のデータをもつ端末ほど早くタイマが発火する．そのため，まず 1 位のデータをもつ端末 A が CHA をブロードキャストし，これを受信した B，C，D，および E が $List_{CH}$ に A を加える．次に，2 位のデータをもつ端末 M が A と同様の操作を行う．これにより，E は GW となる．ここで，data

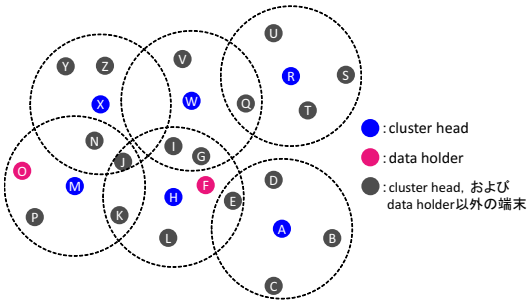


図 1 クラスタ構築の例
Fig. 1 Example of a cluster structure

holder でない端末 R，S，T，および U は一定時間 CHA を受信できないため，CHA の送信タイマを設定する．このとき，（辞書順において）識別子が最も小さい R が最短で CHA 送信し，CH となる．

4.3 Top-k 検索

本節では，提案手法 CTR における Top-k 検索のメッセージ処理方法について説明する．

4.3.1 検索クエリの転送

CTR では，検索結果を取得するために必要な端末間で検索クエリを処理するために，CH 間で検索クエリのルーティングを行う．この時，GW が CH の送信した検索クエリを中継する．CTR では，CH が上位データまでのホップ数をホップ数リスト（$List_{hop}$）[*1]を用いて保持し，検索クエリの送信 CH よりもホップ数が少ないデータを検索する．これにより，不要な検索クエリの転送を抑制できる．ここで，クラスタ構築時において，CH のもつ $List_{hop}$ における各データまでの初期ホップ数は ∞ とする．CH は，検索クエリの転送中にも，検索クエリに添付された情報，および順位表からデータまでのホップ数を把握することにより，$List_{hop}$ の更新を行う．

検索クエリには，検索クエリ発行端末の識別子，検索条件，検索クエリ識別子，要求データ数 k，$RouteList$，$Dest_{CH}$，および $List_{hop}$ が含まれる．$RouteList$ は，検索クエリ発行端末から検索クエリ送信元端末までの端末識別子のリストである．$Dest_{CH}$ は，送信先 CH リスト（CH による転送時は $\emptyset$）であり，$List_{hop}$ は，検索クエリの送信端末が検索する，データの順位，およびそのデータまでのホップ数のリストである．

検索クエリを受信した端末 M_p が CH，および GW の場合の処理をそれぞれ Algorithm 1，および Algorithm 2 に示す．

CH は，受信した検索クエリに含まれる $List_{hop}$ と自身の $List_{hop}$ を比較しながら検索データの絞り込みを行う（Algorithm 1 5–12 行）．各 CH は検索が必要なデータを自律的に選択し，検索クエリの送信の必要性を確認するため，不要な検索クエリの送信を抑制できる．さらに，検索クエリに含まれる $RouteList$，および $Dest_{CH}$ から順位表に含まれる端末までのホップ数を把握でき，検索クエリ転送中に自身の $List_{hop}$ を更新できる．また，GW が転送した検索クエリが，自身が送信した検索クエリであった場合，その転送先のクラスタヘッドを子 CH として記録する（Algorithm 1 23–27 行）．

Algorithm 1 CH によるクエリ転送アルゴリズム

```
1:  if M_p is included as Query.Dest_CH then
2:    if M_p receives for the first time then
3:      parent ← sender node
4:      Record Query.RouteList
5:      for i = 0 to Query.List_hop.size-1 do
6:        j ← Query.List_hop[i].rank
7:        if M_p.List_hop[j-1].hopCt<Query.List_hop[i].hopCt
          and
          M_p≠jth data holder then
8:          M_p's searchRank ← M_p's searchRank ∪ j
9:        else if M_p.List_hop[j-1].hopCt=∞ and
          Query.List_hop[i].hopCt=∞ then
10:         M_p's searchRank ← M_p's searchRank ∪ j
11:       end if
12:     end for
13:     if M_p's searchRank≠ ∅ then
14:       for i = 0 to M_p's searchRank.size-1 do
15:         j ← M_p's searchRank[i]
16:         List_hop ← List_hop ∪ (j, M_p.List_hop[j-1].hopCt)
17:       end for
18:       Broadcast query message
19:     else
20:       Send reply message to parent
21:     end if
22:   else
23:     if   Query.RouteList[Query.RouteList.size-2]=nodeID
        then
24:       child ← child ∪ Query.Dest_CH
25:     else
26:       neighboring node ← neighboring node ∪ the last
          node identifier in Query.RouteList
27:     end if
28:   end if
29: end if
```

GW は基本的に検索クエリを CH に転送するのみであるが，検索する全てのデータまで残り 1 ホップ以下であり，転送先候補の CH が，そのデータの保持端末でない場合を把握できる（Algorithm 2 5–10 行）．これにより，不要な検索クエリの転送を抑制できる．

GW かつ CH でない端末が CH から検索クエリを受信した場合，検索クエリの転送は行わないが，自身が k 位以内のデータを保持していれば，送信元の CH を親として記録する．

ここで，MANET では端末の移動により，端末間のリンク切断が生じる場合がある．CH とのリンク切断が生じた場合，検索クエリの転送に失敗してしまう可能性がある．そのため，CH とのリンク切断を検出した端末は，グリーディクエリをブロードキャストする．グリーディクエリに添付されている情報は検索クエリと同様であるが，$Dest_{CH}$ には，自身とのリンク切断が生じた CH が含まれる．グリーディクエリを受信した端末は，$Dest_{CH}$ に含まれる CH のクラスタに属しており，その CH からの検索クエリを受信していない場合，その CH に検索クエリを送信する．また，リンク切断を検出した端末は，その CH を $List_{CH}$ から削除する．これにより，CH とのリンク切断が生じた場合においても，クラスタに属する端末が検索クエリを中継することにより，CH へ検索クエリを転送できる．

図 2 を用いて，検索クエリの転送例を説明する．端末 B が $k = 2$ の Top-k 検索を行うものとし，図中の吹き出しは各端末が保持する $List_{hop}$ を示す．B は自身が属するクラスタの CH である A を $Dest_{CH}$ とした検索クエリを送信

[*1] 特定の CH に検索クエリを転送するために必要な GW 数をホップ数とする．つまり，図 1 において，A から H，B，および X までのホップ数は，それぞれ 1，0，および 2 となる．

Algorithm 2 GW によるクエリ転送アルゴリズム

```
/* 検索クエリの受信*/
if M_p receives for the first time from CH then
    parent ← sender node
    Record Query.RouteList
    if ∀i of 0 ≤ i <Query.List_hop.size,
    Query.List_hop[i].hopCt≤1 and
    RankTable[Query.List_hop[i].rank-1].node ∉ M_p.List_CH
    then
        Dest_CH.candidate ← ∅
        if M_p is data holder then
            Send reply message to parent
        end if
    else
        Dest_CH.candidate ← M_p.List_CH - sender node
        Set query timer
    end if
else if M_p has already received from CH then
    Dest_CH.candidate ← Dest_CH.candidate -
    Query.Dest_CH
    if Query.RouteList[Query.RouteList.size-2]=nodeID
    then
        child ← child ∪ the last node identifier in
        Query.RouteList
    else
        neighboring node ← neighboring node ∪ the last node
        identifier in Query.RouteList
    end if
end if
/* 検索クエリの送信*/
if query timer expired then
    if Dest_CH.candidate≠ ∅ then
        Dest_CH ← Dest_CH.candidate
        Send query message
    else if M_p is data holder then
        Send reply message to parent
    end if
end if
```

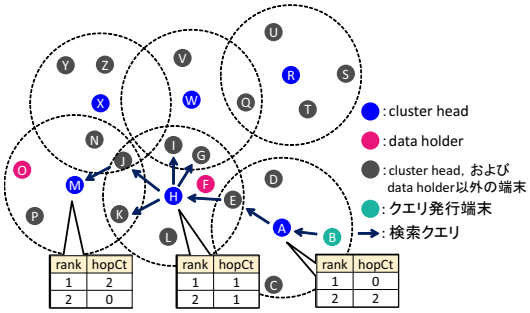


図 2 検索クエリの転送例
Fig. 2 Example of query routing

し，これを受信した A は検索クエリをブロードキャストする．この検索クエリに含まれる $List_{hop}$ は (2 位, 2 ホップ) で構成されている．これを受信した異なるクラスタにも属している E は，そのクラスタの CH である H を $Dest_{CH}$ とした検索クエリを送信する．このとき，A は，E の送信した検索クエリを傍受することにより，H を子 CH に追加する．H からの検索クエリを受信した I，および G は W が 1 ホップ先の 2 位のデータをもつ端末でないと判断できるため，検索クエリを転送しない．J は M が 2 位のデータをもっていることを把握できるため，M に検索クエリを転送するが，X には転送しない．また，J の検索クエリを傍受した K も転送を停止する．以上の例により，CTR では検索結果の取得に必要な端末のみによる Top-k 検索を行えることがわかる．

4.3.2 クエリ応答の返信

クエリ応答は，検索クエリの転送経路を用いて返信する．さらに，リンク切断時には転送先を指定したクエリ応答をブロードキャストし，転送先を把握している端末が自律的にクエリ応答を転送する．

クエリ応答には，検索クエリ識別子，送信端末の識別子，*nextCH*，*nextGW*，*DataList*，および *ReplyCH_List* が含まれている．*nextCH* は，転送先 CH の識別子（ブロードキャスト時のみ）であり，*nextGW* は，転送先 GW の識別子（ブロードキャスト時のみ）である．*DataList* は，データ，そのスコア，保持端末の識別子，およびそのデータの返信を開始した CH からのホップ数のリストであり，*ReplyCH_List* は，クエリ応答を転送してきた CH のリストである．

検索クエリの転送が完了し，クエリ応答の返信を開始する端末 M_q と，これを受信した端末の動作について説明する．

- CH および GW でない data holder の場合
  順位表を参照し，自身のもつ k 位以内のデータを添付したクエリ応答を検索クエリの送信元に送信する．
- CH または GW の場合
  (1) 自身の子 CH がいない端末 M_q は，検索クエリの送信元端末にクエリ応答を送信する．このとき，順位表を参照し，k 位以内のデータをクエリ応答に添付する．M_q が CH の場合，クラスタ内の data holder から受信したデータもクエリ応答に添付し，自身の識別子を *ReplyCH_List* に追加する．
  (2) 全ての子 CH からクエリ応答を受信するか，検索クエリを受信してから一定時間経過した端末 M_r は，クエリ応答を作成し，検索クエリの送信元端末にクエリ応答を送信する．このとき，クエリ応答には受信データ，および，自身の保持する k 位以上のスコアをもつデータを添付する．さらに M_r が CH の場合，*ReplyCH_List* に自身の識別子を追加する．

以上の動作により，子の存在しない CH からクエリ応答の返信を開始し，クエリ発行端末までクエリ応答を集約しながら返信できる．さらに，順位表を参照することにより，不要なデータの返信は生じない．また，受信したクエリ応答中の *DataList* から，データまでのホップ数を把握できるため，CH はこれを用いて自身の $List_{hop}$ を更新することができる．

MANET では端末の移動に伴うリンク切断により，検索クエリの送信元端末にクエリ応答を返信できない場合がある．その場合，Algorithm 3 の手順に従ってクエリ応答を返信する．検索クエリの送信元端末とのリンク切断を検出した端末は，転送先を指定したクエリ応答をブロードキャストする．これを受信した端末は，指定された転送先を把握している（例えば指定された CH のクラスタに属している）場合，クエリ応答を転送する．リンク切断が起きた場合のみ，クエリ応答をブロードキャストし，これを受信した端末が，指定転送先にクエリ応答を転送することにより，データの転送率を向上でき，取得精度を維持できる．

以上の動作により，検索クエリ送信元端末とのリンク切

Algorithm 3 リンク切断時の処理

/* リンク切断を検出 */
1: /* Cluster Head Procedure */
2: **if** neighboring node≠ ∅ **then**
3: parent ← one of nodes included in neighboring node
4: neighboring node ← neighboring node - parent
5: Send reply message to parent
6: **else**
7: nextCH ← *RouteList*[*RouteList*.size-2]
8: nextGW ← original parent
9: Broadcast reply message
10: **end if**
11: /* Gateway Node Procedure */
12: nextCH ← parent
13: nextCH ← ∅
14: Broadcast reply message

/* クエリ応答を受信 */
1: **if** reply.nextCH is included in $List_{CH}$ **then**
2: Send reply message to nextCH
3: **else if** reply.nextGW is included in neighboring node **then**
4: Send reply message to nextGW
5: **end if**

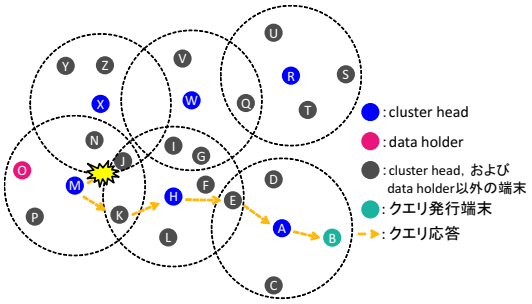


図 3 リンク切断時のクエリ応答の返信例
Fig. 3 Example of reply in case of link disconnection

断が生じた場合にもクエリ応答を返信できる．このとき，各 CH は子端末として GW でなく，CH を管理することにより，不要に GW からのクエリ応答を待つ必要がない．これは，クエリ応答の *ReplyCH_List* からクエリ応答を返信してきた CH を把握できるためである．

図 3 を用いて，リンク切断時のクエリ応答の返信例を説明する．図 2 の転送経路に従って検索クエリを転送後，M-J 間でリンク切断が生じた場合を想定する．端末 M は子となる CH が存在しないため，クエリ応答の返信を開始する．M は表 1 における 2 位以上のスコアをもつデータをクエリ応答に添付する．ここで，M は K を隣接端末として認識していないと仮定する（認識している場合，K にクエリ応答を返信する）．J とのリンク切断を検出した M は *nextGW* に J を，*nextCH* に H を格納したクエリ応答をブロードキャストする．これを受信した K は H のクラスタに属しているため，H にクエリ応答を転送する．H は子端末として J を記録せず，M を記録しているため，J からのクエリ応答を待たず，すぐにクエリ応答を送信できる．その後，クエリ応答は図 3 のように返信され，クエリ発行端末 B は，ネットワーク内の上位 2 個のデータを取得できる．

4.4 スコア更新への対応

本節では，各端末のもつデータのスコアが更新された場合の処理について説明する．

CTR では，スコアが大きいデータをもつ端末ほど CH になりやすく，また，定期的にクラスタを再構築する．クラスタの構築では，ネットワーク内の端末間でデータの正確な順位を共有する必要があるため，スコア更新を全端末に通知する．単純に，スコアの更新をフラッディングを用いて全端末に送信することが考えられるが，1 章で述べたように，不要なメッセージ転送が多く行われてしまい，実行中の Top-k 検索に干渉し，取得精度が低下することが考えられる．そこで CTR では，Top-k 検索時の検索クエリと同様の方法でスコア更新メッセージを転送する．この際，$List_{hop}$ を用いず，CH はスコア更新メッセージをブロードキャストし，これを受信した端末は自身の順位表を修正する（CH は自身の $List_{hop}$ も更新する）．また，GW は，このメッセージを送信元以外の CH に転送する．これにより，メッセージの転送端末数を抑制しつつ，ネットワーク内の全ての端末がスコア更新メッセージを受信できる．

5. シミュレーション評価

本章では，提案手法の性能評価のために行ったシミュレーション実験の結果を示す．本実験では，ネットワークシミュレータ Qualnet6.1[*2]を用いた．

5.1 シミュレーション環境

800[m]×500[m] の 2 次元平面状の領域に 100 台の端末が存在する．各端末はランダムウェイポイント [4] に従い，v[m/sec] の速度で移動する．停止時間は 60[秒] とした．各端末は，IEEE802.11b を使用し，伝送速度 11[Mbps]，通信伝搬距離が 100[m] 程度となる送信電力でデータを送信する．ネットワーク内には，128[Byte] のサイズのデータが 2,000 個存在するものとし，各端末はそれぞれ 20 個のデータをもつものとした．データのスコアは正規分布に従い，取り得る値は [1, 999] の範囲に含まれる整数とした．また，各端末のもつデータのスコアは，300 秒から 600 秒の間で更新されるものとした．

Top-k 検索手法として，CTR，CTR without $List_{hop}$，文献 [3] で提案した手法，および単純手法（Naïve Method）を用いた．CTR without $List_{hop}$ は，基本的に CTR と同様のメッセージ処理を行うが，検索クエリ転送時に $List_{hop}$ を用いない．つまり，スコア更新メッセージと同様の手順で検索クエリが転送され，全ての CH が検索クエリをブロードキャストする．単純手法は，まず CTR と同様に順位表を作成する．検索クエリ，およびスコア更新メッセージはフラッディングを用いて転送し，親子関係を構築せず，検索クエリを受信した端末が k 位以内のデータをもっている場合，検索クエリの送信元端末にすぐにクエリ応答を送信する．各端末は，クエリ応答に初めて受信したデータが添付されている場合，検索クエリの送信元端末にクエリ応答を送信する．また，各手法において，順位表で管理する順位数を 100 とし，k_{max} を 50 とした．さらに，文献 [3] で提案した手法における，経路探索時の TTL を 2 とした．

本実験では，要求データ数 k は基本的に 30 とし，5.2.1 項では 1~50 の間で変化させた．また，端末の移動速度 v は基本的に 0.5[m/sec] とし，5.2.2 項では 0~3[m/sec] の間

[*2] Scalable Network Technologoes: Creaters of QualNet Network Simulator Software,
<URL: http://www.scalable-networks.com/>.

で変化させた．以上のシミュレーション環境において，各端末の初期位置をランダムに決定し，3[sec] 毎にランダムに選ばれた端末が Top-k 検索を行うという処理を 400 回繰り返した際の以下の評価値を調べた．

- 取得精度：順位付き検索結果の性能を測る MAP(Mean Average Precision) の値を取得精度とする．MAP は，各クエリの平均精度 AP(Average Precision) を平均化したものである．AP および MAP は以下の式で求める．

$$AP_i = \frac{1}{k}\sum_{j=1}^{k}\frac{x}{j}\cdot e \tag{1}$$

$$MAP = \frac{1}{querynum}\sum_{i=1}^{querynum} AP_i \tag{2}$$

AP_i は i 番目のクエリの平均精度である．x は，取得した解の上位 j 個のうち正解集合の j 位以内である解の個数である．$querynum$ はクエリの発行数を示す．また，e は以下のように定義される．

$$e = \begin{cases} 1 & (j \text{ 番目の解が正解集合に含まれる}) \\ 0 & (j \text{ 番目の解が正解集合に含まれない}) \end{cases} \tag{3}$$

したがって，MAP は，より上位のデータを取得できているほど高い値となる．

- トラヒック：シミュレーション中に送信されたメッセージの総バイト数を試行回数（400）で割った値．
- 検索時間：全検索クエリに対する，検索クエリ発行端末が検索クエリを発行してから検索結果を取得するまでの平均時間．

5.2 評価結果

5.2.1 要求データ数 k の影響

要求データ数 k を変化させた場合の各手法の性能を図 4 に示す．これらの図において，横軸は要求データ数 k を表し，縦軸は図 4(a) では取得精度，図 4(b) ではトラヒック，および図 4(c) では検索時間を表す．

図 4(a) より，CTR は他の手法よりも高い取得精度を維持できていることが分かる．これは，クラスタリングによるトポロジ変化への対応，およびリンク切断時におけるメッセージ処理により，検索クエリ，およびクエリ応答の転送率が他の手法よりも高いためである．特に k が大きい場合，文献 [3] で提案した手法は取得精度が低下しているのに対して，CTR，および CTR without $List_{hop}$ では取得精度の低下を抑止している．k が大きい場合，クエリ応答のメッセージサイズが大きくなり，パケットロスが発生しやすい．このとき，CTR，および CTR without $List_{hop}$ では，ブロードキャストを用いたクエリ応答の返信により，CH までクエリ応答の返信を行うことで，検索結果に含まれるデータの転送率を向上している．単純手法は，クエリ応答を集約せずに，初めて受信したデータを返信する．そのため，クエリ応答の送信回数が増加し，パケットロスが発生しやすく，高い取得精度を維持できない．

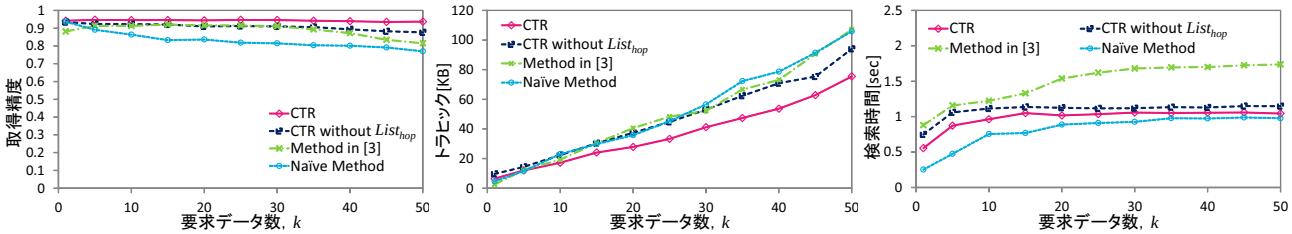
図 4(b)，および図 4(c) より，CTR は低オーバヘッド，および低遅延を達成していることが分かる．CTR では，各 CH が $List_{hop}$ を用いて検索の不要なデータを判断することにより，検索クエリ転送範囲の拡大を抑制しているため，トラヒックを削減し，検索時間を抑制している．また，リンク切断の際も，経路探索を行わずにメッセージを転送できることから，遅延を抑制できる．CTR without $List_{hop}$ では，ネットワーク内の全ての CH に検索クエリが転送されるため，CTR よりもトラヒック，および遅延が増加している．文献 [3] で提案した手法は，k が大きい場合，検索時間が長い．これは，k が大きい場合，検索クエリの転送先端末が増加するため，リンク切断の機会が増加することから，経路探索を行う端末が増加し，検索クエリ，およびクエリ応答の転送遅延が増加するためである．単純手法は，親子関係を構築せず，k 位以内のデータはすぐに検索クエリの送信元端末に送信されるため，検索時間が短い．しかし，上述のように，クエリ応答の送信回数が増加するため，トラヒックが大きい．

5.2.2 端末の移動速度 v の影響

端末の移動速度 v を変化させ，ネットワークトポロジの変化の影響を調べた．このときの各手法の性能を図 5 に示す．これらの図において，横軸は移動速度 v を表し，縦軸は図 5(a) では取得精度，図 5(b) ではトラヒック，および図 5(c) では検索時間を表す．

図 5(a)，CTR はトポロジ変化の影響をそれほど受けず，高い取得精度を維持できていることがわかる．CTR，および CTR without $List_{hop}$ では，CH が一定時間ごとに CHA を送信することにより，各端末は CH とのリンクを確認することができ，定期的にクラスタを再構築している．そのため，文献 [3] で提案した手法よりも，検索クエリ，およびクエリ応答の転送率が高い．さらに，リンク切断が起きた場合においても，クラスタ内の端末をメッシュ端末 [7] と見立てたメッセージ転送を行うことにより，検索クエリ，およびクエリ応答のメッセージ転送失敗を防いでいる．文献 [3] で提案した手法では，トポロジ変化が小さい場合（v が小さい場合），リンク切断が起こる頻度が小さいため，安定したメッセージ転送を実現しており，取得精度が高い．しかし，トポロジ変化が激しい場合（v が大きい場合），経路表における宛先端末，および Top-k 検索中の親端末とのリンク切断が頻繁に生じるため，経路探索に失敗する機会が増加し，取得精度が低下してしまう．

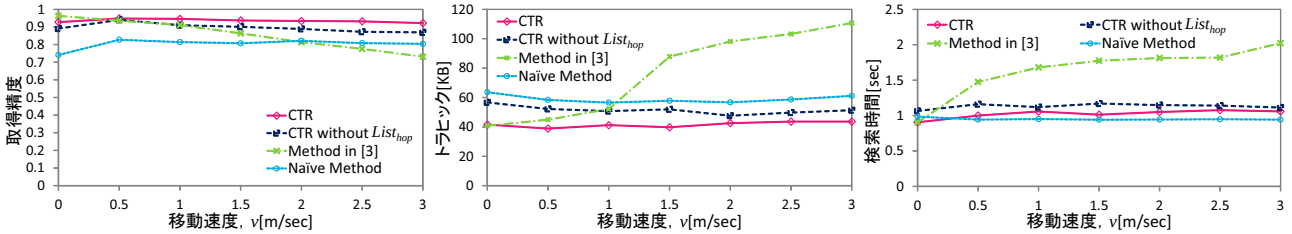
図 5(b)，および 5(c) より，トポロジ変化が激しい場合においても，CTR は低オーバヘッド，および低遅延を達成していることが分かる．これは，5.2.1 項で述べた理由と同様である．CTR では，$List_{hop}$ により検索クエリの不要な転送を抑制しているため，検索クエリの転送範囲拡大を抑制し，CTR without $List_{hop}$，および文献 [3] で提案した手法よりもトポロジ変化の影響を受けにくい．また，CTR，および CTR without $List_{hop}$ では，GW ではなく，CH を子端末として記録している．子である CH からのクエリ応答を受信できた場合，すぐにクエリ応答を返信するため，GW と CH 間のリンク切断の影響を受けにくく，遅延を抑制できる．文献 [3] で提案した手法では，トポロジ変化が激しい場合，多くの端末が経路探索を頻繁に実行す



(a) 取得精度　(b) トラヒック　(c) 検索時間

図 4　要求データ数 k の影響

Fig. 4　Effects of k



(a) 取得精度　(b) トラヒック　(c) 検索時間

図 5　速度 v の影響

Fig. 5　Effects of v

るため，トラヒック，および遅延が増加する．また，トポロジ変化が激しいとき，リンク切断の影響により，子端末からクエリ応答を受信できない場合が多く，検索クエリの中継端末はクエリ応答をすぐに返信できず，遅延が増加してしまう．

6. おわりに

本稿では，MANET において，検索結果の取得に必要な端末のみによる Top-k 検索を実現する，クラスタを用いた Top-k 検索のためのルーティング手法として CTR を提案した．CTR では，スコアの大きいデータをもつ端末をクラスタヘッドとするクラスタリングを行い，クラスタヘッド間で検索クエリのルーティングを行うことにより，検索結果を取得するために必要な端末のみでの Top-k 検索を実現している．また，クラスタヘッドによる定期的なメッセージ送信，およびクラスタの再構築によりトポロジ変化に対応し，トポロジ変化が激しい場合でも高い取得精度を維持できる．リンク切断の際には，経路探索を用いず，クラスタ内の端末がメッセージをクラスタヘッドに中継することにより，転送遅延を抑制しつつ，メッセージの転送率を向上している．シミュレーション実験の結果から，提案手法は従来手法の性能を上回ることを確認した．

本稿では，クラスタを用いてトポロジ変化に対応し，高い取得精度を維持した．ここで，クラスタ内に存在する端末に，検索結果に含まれるデータの複製を配置することで，検索クエリの転送範囲の拡大を抑制でき，トラヒック，および検索遅延を抑制しつつ，高い取得精度を保証できると考えられる．そのため，複製配置を考慮したメッセージ処理手法について，今後取り組む予定である．

謝辞　本研究の一部は，文部科学省研究費補助金・基盤研究 S（21220002），および基盤研究 B（24300037）の研究助成によるものである．ここに記して謝意を表す

参考文献

[1] Akbarinia, R., Pacitti, E., and Valduriez, P.: Best position algorithm for top-k queries, *Proc. Int. Conf. on VLDB*, pp.495–506 (2007).

[2] Amagata, D., Sasaki, Y., Hara, T., and Nishio, S.: A routing method for top-k query processing in mobile ad hoc networks, *Proc. Int. Conf. on Advanced Information Networking and Applications*, pp.161–168 (2013).

[3] Amagata, D., Sasaki, Y., Hara, T., and Nishio, S.: A robust routing method for top-k queries in mobile ad hoc networks, *Proc. Int. Conf. on MDM*, (2013 to appear).

[4] Camp, T., Boleng, J., and Davies, V.: A survey of mobility models for ad hoc network research, *Wireless Communications and Mobile Computing*, Vol.2, No.5, pp.483–502 (2002).

[5] Hagihara, R., Shinohara, M., Hara, T., and Nishio, S.: A message processing method for top-k query for traffic reduction in ad hoc networks, *Proc. Int. Conf. on MDM*, pp.11–20 (2009).

[6] Li, X., Xu, J. and Wu, J.: HR-SDBF: An approach to data-centric routing in WSNs, *Int. Journal. High Performance Computing and Networking*, Vol.6, No.3, pp.181–196 (2010).

[7] Mendez, M.R., and Aceves, G. -L.J.J.: An interest-driven approach to integrated unicast and multicast routing in MANETs, *Proc. Int. Conf. on Network Protocol*, pp.248–257 (2008).

[8] Padhariya, N., Mondal, A., Goyal, V., Shankar, R., and Madria, K.S.: EcoTop: An economic model for dynamic processing of top-k queries in mobile-P2P networks, *Proc. Int. Conf. on DASFAA*, pp.251–265 (2011).

[9] Sasaki, Y., Hara, T., and Nishio, S.: A top-k query method by estimating score distribution in mobile ad hoc networks, *Proc. Int. Conf. on Advanced Information Networking and Applications Workshops*, pp.944–949 (2010).

[10] Sasaki, Y., Hara, T., and Nishio, S.: Two-phase top-k query processing in mobile ad hoc networks, *Proc. Int. Conf. on NBiS*, pp.42–49 (2011).

[11] Vlachou, A., Doulkeridis, C., Nørvåg, K., and Vazirgiannis, M.: On efficient top-k query processing in highly distributed envirinments, *Proc. Int. Conf. on SIGMOD*, pp.753–764 (2008).

[12] Wu, M., Xu, J., Tang, X. and Lee, -C.W.: Top-k monitoring in wireless sensor networks, *IEEE Trans. Knowledge and Data Engineering*, Vol.19, No.7, pp.962–976 (2007).