情報処理学会論文誌 Vol.55 No.5 1555–1567 (May 2014)

推薦論文

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クエリルーティング手法

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)]

受付日 2013年9月20日, 採録日 2014年2月14日

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 個のスコアを持つデータを検索する Top-k 検索への関心が高まっている．本論文では，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 を提案する．CTR では，スコアが大きいデータを持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．

キーワード：Top-k 検索，ルーティング，クラスタリング，モバイルアドホックネットワーク

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

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

Received: September 20, 2013, Accepted: February 14, 2014

Abstract: Top-k queries, which retrieve k data items with the highest scores determined based on a query condition designated by a user, have been received much research interests. In this paper, we propose CTR which aims at performing top-k query processing by only nodes that contribute to acquisition of the exact answer in mobile ad hoc networks. In CTR, nodes holding data items with high scores become *ClusterHeads* (CHs), and top-k queries are transmitted between CHs via gateway nodes which belong to multiple clusters. Each CH maintains hop-counts between itself and nodes holding data items with high scores so that it can judge whether or not to transmit a query on the fly. As a result, CTR suppresses the overhead while keeping high accuracy of query result. The simulation results show that CTR functions well in terms of accuracy of query result, overhead, and delay.

Keywords: Top-k queries, routing, clustering, mobile ad hoc networks

1. はじめに

近年，ルータ機能を持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

[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, 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

本論文の内容は 2013 年 7 月のマルチメディア，分散，協調とモバイル（DICOMO2013）シンポジウムにて報告され，マルチメディア通信と分散処理研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

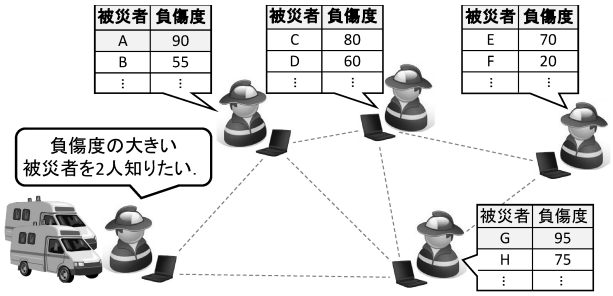


図 1 アドホックネットワークにおける Top-k 検索例
Fig. 1 Example of top-k query processing in a MANET.

（*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．そこで，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 検索を用いることが有効であると考えられる．たとえば，**図 1** のように，緊急災害が起きた現場に救急隊員が派遣され，MANET を形成し，情報共有やデータ検索を行うことが考えられる．このとき，限られた k 個の救援資源（たとえば救急車）を割り当てる際に，Top-k 検索を用いることが有効である．

様々なネットワーク環境において，Top-k 検索を行う手法が数多く提案されている．まず，ネットワーク内のすべてのデータを 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集中管理型の方法 [1], [8] 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 台の端末のみへのトラヒックの集中にともな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 [12], [17] や無線センサネットワーク [9], [10], [11], [18] 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，ネットワーク内のすべ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 と類似した特徴を持つ．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に，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手法が提案されている [9], [18]．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．MANET では，低トラヒックで，検索結果の取得に必要な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．

既存研究として，MANET におけるトラヒックの削減と検索結果の取得精度の低下抑止を目的とした Top-k 検索手法が，いくつか提案されている [7], [15], [16]．しかし，これらの手法では，フラッディングによるデータ検索を行い，ネットワーク内のすべての端末に検索クエリを送信していることから，上記の問題を十分には解決できていない．また，文献 [2], [3] では，経路表を用いて，上位 k 個のデータの取得に必要な隣接端末のみへ検索クエリを送信する手法が提案されている．これらの手法では，検索結果の取得に必要な端末のみに検索クエリを送信できている．しかし，各順位のデータに対して検索クエリの送信先である隣接端末を管理しているため，端末の移動速度が大きい場合，経路表の修正に大きなトラヒックが生じ，パケットロスの発生を招くことから，経路表，および検索結果の取得精度を維持できないという問題がある．

そこで本論文では，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 Routing）を提案する．CTR では，スコアが大きいデータを持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（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．

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 章で本論文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，MANET におけるクラスタを用いたルーティング手法，および Top-k 検索手法のうち，代表的な既存手法を紹介する．

2.1 クラスタを用いたルーティング

これまでに，MANET におけるクラスタを用いたルーティング手法は数多く提案されている．そのうち，CTR と同様に 1 ホップクラスタリングを用いた手法について紹介する．

文献 [6] では，端末識別子を用いたクラスタリングアルゴリズムを提案している．この手法では，隣接端末中で，

最も識別子の大きい端末が CH となる．文献 [14] では，隣接端末の数に基づいたクラスタリング手法が提案されており，隣接端末の数が最大である端末が CH となる．文献 [4] では，端末の移動速度から CH を決定する手法を提案している．この手法では，クラスタのメンテナンスを小さくするため，隣接端末と自身の移動速度を比較し，相対的な移動速度が最も小さい端末が CH となる．また，これらのパラメータを重み付けした値から CH を決定する手法が，文献 [19] で提案されている．これらの手法では，ある端末へのデータ転送の効率化のためにクラスタリングを行っている．単純にこれらの手法を適応した場合にも，検索クエリのフラッディングは抑制できるが，端末が持つデータを考慮していないため，すべての CH にクエリを送信する必要があり，無駄なトラヒックが発生してしまう．CTR では，上位となるデータを持つ端末を CH とすること，および CH が上位データまでのホップ数を管理することにより，クラスタ内の端末，および隣接 CH への不要なクエリの転送を削減できる．

2.2 Top-k 検索

筆者らの知る限り，MANET における Top-k 検索手法は，文献 [2], [3], [7], [13], [15], [16] でのみ提案されている．文献 [13] での想定環境は，経済モデル（economic scheme）を採用しており，Top-k 検索を行う端末は，仮想的な通貨を支払うことによりデータを得たり，報酬や罰則を与えることによりデータの送信確率を決定したりするなど，本論文で想定する環境と本質的に異なる．また，文献 [13] で提案されている手法においても，検索クエリの伝搬はフラッディングを用いており，1 章で述べた問題を解決できない．

文献 [7], [15], [16] 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が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．

文献 [2], [3] では，Top-k 検索のためのルーティング手法が提案されている．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持つデータ，および検索クエリの転送先（宛先）を把握している．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）で転送する手法が提案されており，文献 [3] では，文献 [2] の手法を拡張し，検索クエリをマルチキャスト（複数経路）で転送する手法が提案されている．

MANET における既存の Top-k 検索手法では，文献 [3] の性能が最も高いと考えられる．しかし，文献 [2], [3] の手法では，Top-k 検索中にのみ経路表を修正しているため，トポロジ変化が激しい環境や検索クエリの発行頻度が小さい場合は，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，各端末は自身が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

本論文では，MANET を構成する各端末が，自身と他の端末の持つデータに対して Top-k 検索を行う環境を想定する．ネットワーク内には，n 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 m 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，すべ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（検索クエリ発行端末）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 個のスコアを持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，本論文における提案手法 CTR を説明する．まず，CTR の概要について述べ，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，自身の持つデータの順位を把握する．また，スコアが高いデータを持つ端末から順にクラスタヘッド（CH）となる 1 ホップクラスタリングを行う．これにより，上位のデータを保持する端末が 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が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，複数のクラスタに属するゲートウェイノード（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] で提案されている手法のように，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，その 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．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．

データのスコアが更新された場合には，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 M_p は，順位表（**表 1** 参照）を作成する．M_p は順位表作成メッセージをフラッディングし，このメッセージを初めて受信した端末は，メッセージの送信元端末を自身の親端末として記録する．また，スコアの高い r_{max} 個のデータのスコア，およびデータの保持端末識別子を親端末に送信する．ここで，スコア更新に対応するため，$r_{max} > k_{max}$ とする．この手順により，M_p は順位表を作成できる．M_p は，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，上位データの情報をすべての端末で共有する．

順位表を受信した，k_{max} 位以内のデータを持つ端末（データ所持端末と呼ぶ）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 を受信しなかったデータ所持端末以外の端末は，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，上位データを持つ端末が基本的に CH となるため，検索結果に含まれるデータを持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．ここで，α は，定数値をとるシステムパラメータである．

各端末が表 1 で表されるデータを保持している場合において，**図 2** を用いてクラスタを構築する例を説明する．黒色の丸は CH を示し，灰色の丸はデータ所持端末を示す．白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，高順位のデータを持つ端末ほど早くタイマが発火する．そのため，まず 1 位のデータを持つ端末 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 位のデータを持つ端末 M が，そ

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

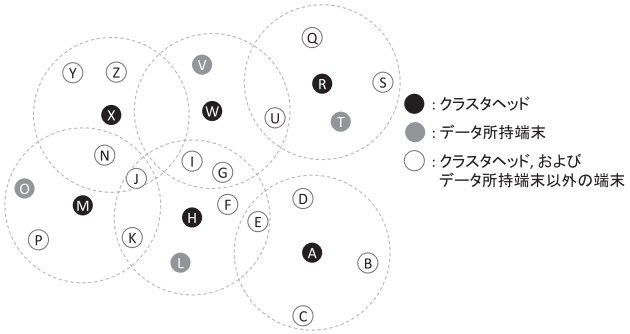


図 2 クラスタ構築の例
Fig. 2 Example of a cluster structure.

の次に3位のデータを持つ端末Hが，Aと同様の操作を行う．これにより，E，J，およびKはGWとなる．ここで，データ所持端末でない端末R，S，T，およびUは一定時間内に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を送信し，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．このとき，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]を用いて保持する．検索クエリを初めて受信したCHは，受信した検索クエリに添付されている$List_{hop}$と，自身の$List_{hop}$を比較し，ホップ数が小さい順位のデータを検索する．これにより，無駄なデータの検索を抑制できるため，各CHは，不要に検索クエリを転送しない．ここで，クラスタ構築時において，CHが保持する$List_{hop}$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，この検索クエリを中継（転送）する端末までの，検索クエリの転送経路上に存在する端末の識別子のリストである．$Dest_{CH}$は，この検索クエリの送信先となるCHの識別子のリストである．そのため，CHが検索クエリを送信する場合，その検索クエリに含まれる$Dest_{CH}$は空である．$List_{hop}$は，この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アルゴリズム1，およびアルゴリズム2に示す．

添付されている$Dest_{CH}$に，自身の識別子が含まれる検索クエリを初めて受信したCHは，検索クエリの送信元端末を親端末（M_p.parent）として記録し，検索クエリに添付されている$RouteList$を記録する（アルゴリズム1，1–4行）．その後，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（アルゴリズム1，5–12行）．各CHは検索が必要なデータの順位（M_p.SearchRank）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．また，クラスタ構築後のTop-k検索では，

[*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2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 1 CHによるクエリ転送アルゴリズム

```
 1: if ∃M_p ∈ Query.Dest_CH then
 2:   if M_p receives for the first time then
 3:     M_p.parent ← Query.RouteList[|Query.RouteList| − 1]
 4:     M_p.RouteList ← Query.RouteList
 5:     for i = 0 to |Query.List_hop| − 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 ≠ holder of jth data then
 8:         M_p.SearchRank ← M_p.SearchRank ∪ {j}
 9:       else if M_p.List_hop[j − 1].hopCt = ∞ and
          Query.List_hop[i].hopCt = ∞ then
10:         M_p.SearchRank ← M_p.SearchRank ∪ {j}
11:       end if
12:     end for
13:     if M_p.SearchRank ≠ ∅ then
14:       Query.List_hop ← ∅
15:       for i = 0 to ⌈M_p.SearchRank| − 1 do
16:         j ← M_p.SearchRank[i]
17:         Query.List_hop ← Query.List_hop ∪
            {(j, M_p.List_hop[j − 1].hopCt)}
18:       end for
19:       Query.RouteList ← Query.RouteList ∪ {M_p}
20:       Query.Dest_CH ← ∅
21:       Broadcast query message
22:     else
23:       Perform Algorithm 3
24:     end if
25:   else
26:     if Query.RouteList[|Query.RouteList| − 2] = M_p then
27:       M_p.child ← M_p.child ∪ Query.Dest_CH
28:     else
29:       M_p.neighbors ← M_p.neighbors ∪
          Query.RouteList[|Query.RouteList| − 1]
30:     end if
31:   end if
32: end if
```

Algorithm 2 GWによるクエリ転送アルゴリズム

```
 1: /* 検索クエリの受信 */
 2: if M_p receives for the first time from a CH then
 3:   M_p.parent ← Query.RouteList[|Query.RouteList| − 1]
 4:   M_p.RouteList ← Query.RouteList
 5:   if ∀i of 0 ≤ i < |Query.List_hop|,
      Query.List_hop[i].hopCt≤1 and
      M_p.RankingTable[Query.List_hop[i].rank−1].node ∉
      M_p.List_CH then
 6:     Dest_CH.candidate ← ∅
 7:     if M_p is data holder then
 8:       Perform Algorithm 3
 9:     end if
10:   else
11:     Dest_CH.candidate ← M_p.List_CH −M_p.parent
12:     Set query timer
13:   end if
14: else if M_p has already received from a CH then
15:   Dest_CH.candidate ← Dest_CH.candidate − Query.Dest_CH

16:   if Query.RouteList[|Query.RouteList| − 2] = M_p then
17:     M_p.child ← M_p.child ∪
        Query.RouteList[|Query.RouteList| − 1]
18:   else
19:     M_p.neighbors ← M_p.neighbors ∪
        Query.RouteList[|Query.RouteList| − 1]
20:   end if
21: end if
22: /* 検索クエリの送信 */
23: if query timer expired then
24:   if Dest_CH.candidate ≠ ∅ then
25:     Query.RouteList ← M_p.RouteList ∪ M_p
26:     Query.Dest_CH ← Dest_CH.candidate
27:     Send query message 4
28:   else if M_p is data holder then
29:     Perform Algorithm 3
30:   end if
31: end if
```

CH が保持する $List_{hop}$ における，各順位のデータに対するホップ数は基本的に ∞ である．そのため，受信した検索クエリに添付されている $List_{hop}$ と自身の $List_{hop}$ において，同じ順位のデータに対するホップ数がともに ∞ である場合，その順位を M_p.SearchRank に追加する（アルゴリズム 1, 9–10 行）．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 CH（$Dest_{CH}$）を子 CH として記録する（アルゴリズム 1，26–30 行）．

CH からの検索クエリを初めて受信した GW は，CH と同様に，親端末と *RouteList* を記録する（アルゴリズム 2，2–4 行）．GW は基本的に検索クエリを CH に転送するのみであるが，検索するすべてのデータまで残り 1 ホップ以下であり，この検索クエリの転送先となる CH の候補（$Dest_{CH}$.candidate）が，そのデータの保持端末でない場合を把握できる（アルゴリズム 2，5–9 行）．これにより，不要な検索クエリの転送を抑制できる．また，検索クエリを転送する必要がなく，上位 k 個に含まれるデータを保持している（data holder である）場合，親端末にクエリ応答を返信する（アルゴリズム 2，8 および 27 行）．

GW および CH のいずれでもない端末が 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 を親として記録し，クエリ応答を返信する．

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，検索クエリをブロードキャストする．この検索クエリに添付される $Dest_{CH}$ には，自身とのリンク切断が生じた CH の識別子が含まれる．この検索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，その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 3 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 CH 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

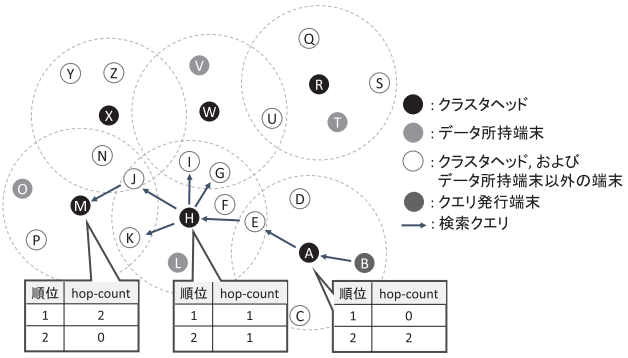


図 3 検索クエリの転送例
Fig. 3 Example of query routing.

$Dest_{CH}$ 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 位のデータを持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 位のデータを保持し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 では上位 k 個のデータ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分かる．

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，その転送先となる端末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，*DataList*，および *ReplyCH_List* が含まれている．*DataList* は，データ，そのデータの保持端末の識別子，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，これまでにクエリ応答を転送してきた CH の識別子のリストである．

4.3.1 項から，検索クエリを受信する端末 M_q は，CH である端末，GW である端末，および，CH および GW のいずれでもないデータ所持端末である．クエリ応答の送信，および受信の処理は，それぞれアルゴリズム 3，およびアルゴリズム 4 に従う．以下に，クエリ応答を送信する M_q と，このクエリ応答を受信した端末の動作について説明する．

- CH および GW のいずれでもないデータ所持端末の場合
  順位表を参照し，自身が持つ k 位以内のデータを添付したクエリ応答を，自身の親端末に送信する（アルゴリズム 3，5 行）．
- CH または GW の場合
  (1) 自身の子 CH がいない端末 M_q が，CH の場合，または，データ所持端末である GW の場合，自身

Algorithm 3 クエリ応答送信アルゴリズム

```
1: Reply.ReplyCH_List ← M_q.ReplyCH_List //クエリ応答を受信
   していない場合，M_q.ReplyCH_List= ∅
2: if M_q is a CH then
3:     Reply.ReplyCH_List ← Reply.ReplyCH_List ∪ {M_q}
4: end if
5: DataList ← ∀{data, M_q, 0} of its own data,
   where data.score ≥ M_q.RankingTable[k − 1].score
6: Reply.DataList ← DataList ∪ M_q.DataList //クエリ応答を受
   信していない場合，M_q.DataList= ∅
7: Send reply message to M_q.parent
```

Algorithm 4 クエリ応答受信アルゴリズム

```
1: M_q.ReplyCH_List ← Reply.ReplyCH_List
2: Update each hop_count in Reply.DataList
3: M_q.DataList ← M_q.DataList ∪ Reply.DataList
4: if M_q receives replies from its all children || predetermined
   time limit has elapsed then
5:     Perform Algorithm 3
6: end if
```

の親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，クラスタ内のデータ所持端末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（アルゴリズム 3，3 行）．

(2) すべての子 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，自身の親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 位以上のスコアを持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 CH の存在しない端末からクエリ応答の返信を開始し，検索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 では端末の移動にともなうリンク切断により，親端末にクエリ応答を返信できない場合がある．その場合，アルゴリズム 5 およびアルゴリズム 6 の手順に従ってクエリ応答を返信する．このときに送信されるクエリ応答の要素は，通常のクエリ応答の要素に加え，*nextCH* および *nextGW* を含む．*nextCH* は，このクエリ応答の次の転送先となる CH の識別子であり，*nextGW* は，このクエリ応答の次の転送先となる GW の識別子である．親端末とのリンク切断を検出した端末は，転送先となる端末の識別子（*nextCH*，および *nextGW*）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，指定された転送先の端末を把握している（たとえば指定された CH のクラスタに属している）場合，その端末に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，指定された転送先の端末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させることができ，取得精度を維持できる．

Algorithm 5 クエリ応答のブロードキャスト

```
1: /* 端末 M_q がリンク切断を検出 */
2: /* CH の場合 */
3: if M_q.neighbors ≠ ∅ then
4:     M_q.parent ← ∃M_p' ∈ M_q.neighbors
5:     M_q.neighbors ← M_q.neighbors − M_q.parent
6:     Send reply message to parent
7: else
8:     nextCH ← M_q.RouteList[|M_q.RouteList| − 2]
9:     nextGW ← original parent of M_q
10:    Broadcast reply message
11: end if
12: /* GW の場合 */
13: nextCH ← M_q.parent
14: nextGW ← ∅
15: Broadcast reply message
```

Algorithm 6 ブロードキャストされたクエリ応答を受信

```
1: /* 端末 M_q' がブロードキャストされたクエリ応答を受信 */
2: if ∃ Reply.nextCH ∈ M_q' List_CH then
3:     Send reply message to Reply.nextCH
4: else if ∃ Reply.nextGW ∈ M_q' neighbors then
5:     Send reply message to Reply.nextGW
6: end if
```

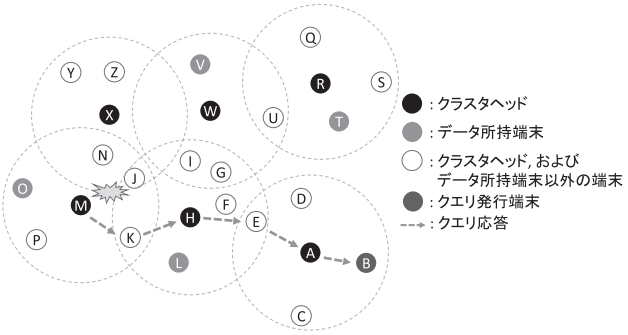


図 4 リンク切断時のクエリ応答の返信例

Fig. 4 Example of reply in case of link disconnection.

以上の動作により，親端末とのリンク切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，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 4 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 3 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 位以上のスコアを持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4のように返信され，検索クエリ発行端末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，各端末の持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持つ端末ほど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，ネットワーク内のすべ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n台の端末が存在する．各端末はランダムウェイポイントモデル [5] 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（3.9842 [dBm]）でデータを送信する．各端末はそれぞれ，128[bytes] のデータを20個保持するものとした．データのスコアは正規分布に従い，とりうる値は [1, 999] の範囲に含まれる整数とした．また，各端末の持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．本実験におけるパラメータは**表2**のとおりである．

Top-k検索手法として，CTR，CTR without $List_{hop}$，既存研究で最も性能の高い文献 [3] の手法，およびフラッディングによるデータ検索を行う単純手法を用いた．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，すべてのCHが検索クエリをブロードキャストする．文献 [3] で提案されている手法は，2.2節で述べたとおり，経路表を用いてクエリを転送する．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位以内のデータを保持している場合，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．各手法において，順位表で管理する順位数r_{max}を100とし，k_{max}を50とした．CTRにおいて，クラスタリングの際に用いるパラメータαおよびtは，予備実験から，基本値（表2）における取得精度が最も高くなる値を用いた．さらに，文献 [3] で提案されている手法における，経路探索時のTTLを2とした．

表2 パラメータ設定
Table 2 Configuration of parameters.

パラメータ	意味	基本値	範囲
k	要求データ数	30	1–50
v	速度 [m/sec]	1.0	0–3
I_q	クエリ発行周期 [sec]	3	2–10
n	端末数	100	100–300

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，I_q [sec] ごと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（シミュレーション時間は$I_q \times 400$ [sec]）際の以下の評価値を調べた．

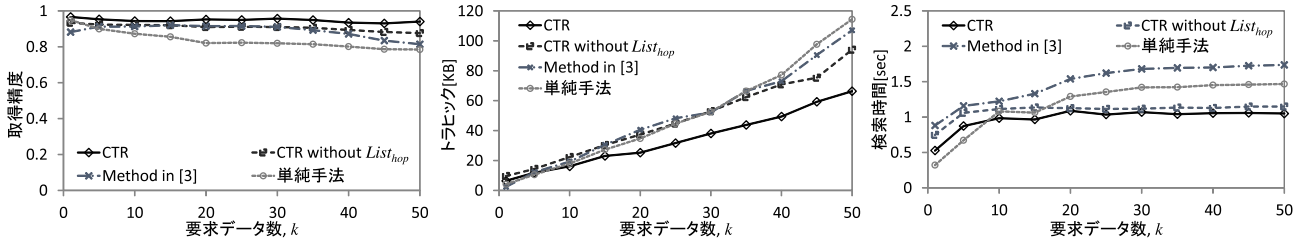
- 取得精度：順位付き検索結果の性能を測るMAP（Mean Average Precision）の値を取得精度とする．正解集合は，ネットワーク全体に存在するデータのうち，スコアが最も高いk個のデータの集合である．MAPは，各クエリの平均精度AP（Average Precision）を平均化したものであり，APおよびMAPは以下の式で求める．

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

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

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（つまり，400）を示す．また，eは以下のように定義される．

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



(a) 取得精度 (b) トラヒック (c) 検索時間

図 5 要求データ数 k の影響
Fig. 5 Impact of k.

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

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クエリ発行回数（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．
- 各端末の送受信トラヒック：各端末が，シミュレーション中に送信，および受信したメッセージの総バイト数をクエリ発行回数（400）で割った値．

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 5** 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 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) 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] で提案された手法は取得精度が低下していることに対して，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させている．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．

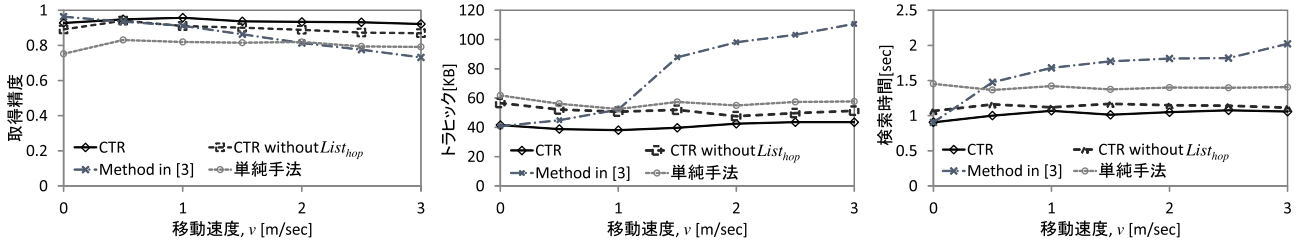
図 5 (b)，および図 5 (c) より，CTR は低オーバヘッド，および低遅延を達成していることが分かる．CTR では，各 CH が $List_{hop}$ を用いて検索の不要なデータを判断することにより，検索クエリ転送範囲の拡大を抑制しているため，トラヒックを削減し，検索時間を抑制している．また，リンク切断の際も，経路探索を行わずにメッセージを転送できることから，遅延を抑制できる．CTR without $List_{hop}$ では，ネットワーク内のすべての CH に検索クエリが転送されるため，CTR よりもトラヒック，および遅延が増加している．文献 [3] で提案された手法は，k 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，文献 [3] で提案された手法よりも検索時間が短い．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 トポロジ変化の影響

端末の移動速度 v，およびクエリ発行間隔 I_q 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．v が大きい場合，隣接端末間の距離が短時間で長くなるため，リンク切断が起きやすく，ネットワークトポロジが大きく変化する．I_q が大きい場合，検索クエリが発行されていない間に，CTR では CH と GW 間の距離が，文献 [3] で提案された手法では経路表で管理している隣接端末との距離が長くなるため，これらの端末間でリンク切断を検出する機会が増加する．

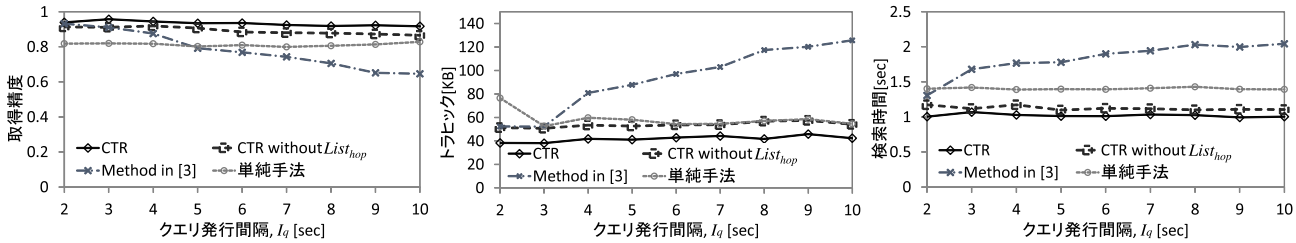
v，および I_q を変えた場合の各手法の性能をそれぞれ**図 6**，および**図 7** に示す．図 6 において，横軸は移動速度 v を表し，縦軸は図 6 (a) では取得精度，図 6 (b) ではトラヒック，および図 6 (c) では検索時間を表す．また，図 7 において，横軸はクエリ発行間隔 I_q を表し，縦軸は図 7 (a) では取得精度，図 7 (b) ではトラヒック，および図 7 (c) では検索時間を表す．

図 6 (a) および図 7 (a) から，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分かる．CTR，および CTR without $List_{hop}$ では，CH



(a) 取得精度 (b) トラヒック (c) 検索時間

図 6 速度 v の影響
Fig. 6 Impact of v.



(a) 取得精度 (b) トラヒック (c) 検索時間

図 7 クエリ発行周期 I_q の影響
Fig. 7 Impact of I_q.

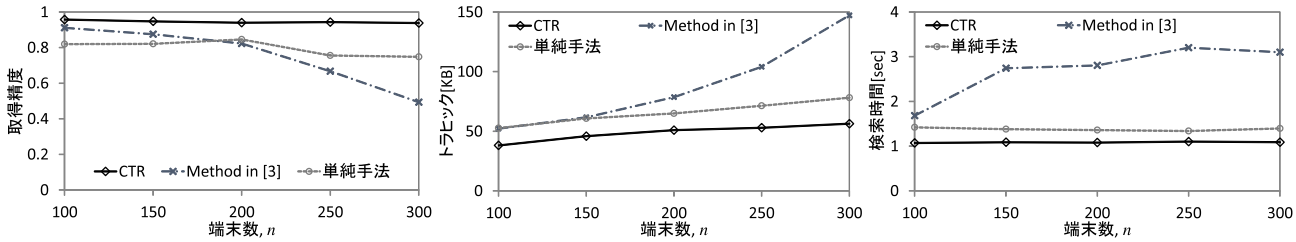
が一定時間ごとに CHA を送信することにより，各端末は CH とのリンクを確認することができ，定期的にクラスタを再構築している．そのため，文献 [3] で提案した手法よりも，検索クエリ，およびクエリ応答の転送率が高い．さらに，リンク切断が起きた場合においても，リンク切断を検出した端末はメッセージをブロードキャストし，これを受信した端末が，受信メッセージの送信元端末とのリンク切断が起きた端末へ，自律的にそのメッセージを転送する．これにより，検索クエリ，およびクエリ応答のメッセージ転送失敗を防いでいる．文献 [3] で提案された手法では，トポロジ変化が小さい場合（v が小さい場合），リンク切断が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 6 (b)，図 6 (c)，図 7 (b)，および図 7 (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] で提案され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] で提案された手法では，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．

5.2.3 端末数 n の影響

端末数 n を変化させた場合の各手法の性能を**図 8** に示す．これらの図において，横軸は端末数 n を表し，縦軸は図 8 (a) では取得精度，図 8 (b) ではトラヒック，および図 8 (c) では検索時間を表す．5.2.1 項，および 5.2.2 項から，CTR without $List_{hop}$ の性能は CTR 以下であることが分かるため，CTR without $List_{hop}$ の結果は割愛する．

図 8 (a) より，CTR の取得精度は端末数にかかわらず，一定して高い値であることが分かる．CTR では，検索結果の取得に必要な端末のみにより Top-k 検索を行っているため，メッセージを送信する端末数は，ネットワーク全体の端末数によらず，ほぼ一定である．そのため，検索時間も一定になる（図 8 (c)）．一方，文献 [3] で提案されている手法の取得精度は，端末数が増加するとともに低下している．ネットワーク内の端末数が増加すると，上位 k 個のデータを取得するために必要な端末数の割合は低下する．文献 [3] で提案されている手法では，Top-k 検索中にのみ経路表を修正しているため，検索クエリを受信しない端末は自身の経路表を修正する機会を失い，その精度を保つことができない．また，端末数が多いとき，単純手法の取得精度も低下する．これは，検索クエリをフラッディングによ



(a) 取得精度 (b) トラヒック (c) 検索時間

図 8 端末数 n の影響
Fig. 8 Impact of n.

表 3 端末の送受信トラヒック [KB]
Table 3 Traffic transmitted and received by nodes.

手法	平均	分散	最大
CTR	0.83	0.08	1.54
CTR without $List_{hop}$	0.90	0.13	1.95
Method in [3]	1.01	0.35	3.03
単純手法	1.08	0.70	3.65

り転送しているため，多くの端末がクエリ転送を行い，クエリ応答とのメッセージ衝突が頻繁に生じるためである．

図 8 (b) より，CTR のトラヒックは，端末数が増加するとともにわずかに増加することが分かる．CTR では，CH とクエリを転送する GW とのリンク切断が生じた際，クラスタ内に存在する端末が受信したメッセージを CH に転送している．ネットワーク内の端末数が多い場合，クラスタに含まれる端末も増加するため，リンク切断の際に多くの端末がメッセージを転送し，トラヒックが増加する．文献 [3] で提案されている手法のトラヒックが増加するのは，多くの端末が自身の経路表を修正する機会を失うことに起因する．つまり，多くの端末が，自身が検索クエリを発行した際に経路探索メッセージを送信するため，トラヒックが増加し，検索時間が長くなる（図 8 (c)）．単純手法のトラヒックが増加するのは，フラッディングにより検索クエリを転送しているためである．しかし，k 位以内のデータを持つ端末のみクエリ応答の返信を開始することから，クエリ応答を返信する端末数は基本的に一定であり，検索時間が増加しない（図 8 (c)）．

5.2.4 各端末の送受信トラヒック

CTR では，CH 間のメッセージ転送を行うため，CH にトラヒックが集中することが考えられる．そこで，消費電力に影響のある，各端末の送受信トラヒックを**表 3** に示す．

表 3 から，CTR は比較手法よりも送受信トラヒックの平均，最大が小さいことが分かる．さらに分散も小さいことから，負荷分散ができていることも分かる．CTR では，GW が検索クエリを傍受することにより，不要に CH に検索クエリが転送されることを抑制している．また，端末が移動していること，および定期的にクラスタを再構築していることから，CH や GW となる端末が固定されないため，一部の端末に集中して負荷がかかることはない．

6. おわりに

本論文では，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持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させている．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．

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，および検索遅延をさらに抑制しつつ，高い取得精度を保証できると考えられる．そのため，今後は，複製配置を考慮したメッセージ処理手法について検討する予定である．

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 postision 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.42–49 (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*, pp.251–256 (2013).
[4] Basu, P., Khan, N. and Little, T.D.C.: A mobility based metric for clustering in mobile ad hoc networks, *Proc.*

Int. Conf. on ICDCS, pp.413–418 (2001).
[5] 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).
[6] Ephremides, A., Wieselthier, J.E. and Baker, D.J.: A design concept for reliable mobile radio networks with frequency hopping signaling, *Proc. IEEE*, Vol.75, No.1, pp.56–73 (1987).
[7] 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).
[8] Hristidis, V., Koudas, N. and Papakonstantinou, Y.: Prefer: A system for the efficient execution of multi-parametric ranked queries, *Proc. Int. Conf. on SIGMOD*, pp.259–270 (2001).
[9] Jiang, H., Cheng, J., Wang, D., Wang, C. and Tan, G.: Continuous multi-dimensional top-k query processing in sensor networks, *Proc. Int. Conf. on INFOCOM*, pp.793–801 (2011).
[10] Liu, X., Xu, J. and Lee, W.-C.: A cross pruning framework for top-k data collection in wireless sensor network, *Proc. Int. Conf. on MDM*, pp.157–166 (2010).
[11] Malhotra, B., Nascimento, M. and Nikolaidis, L.: Exact top-k queries in wireless sensor networks, *IEEE Trans. Knowledge and Data Engineering*, Vol.23. No.10, pp.1513–1525 (2011).
[12] Michel, S., Peter, T. and Weikum, G.: KLEE: A framework for distributed top-k query algorithms, *Proc. Int. Conf. on VLDB*, pp.637–648 (2005).
[13] 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).
[14] Parekh, A.K.: Selecting routers in ad-hoc wireless networks, *Proc. Int. Telecommunications Symposium*, pp.420–424 (1994).
[15] 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).
[16] 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).
[17] 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).
[18] Wu, M., Xu, J., Tang, X. and Lee, W.-C.: Top-k monitoring in wireless sensor networks, *IEEE Trans. Knowledge and Data Engineering*, Vol.19, No.7, pp.962–976 (2007).
[19] Yang, W.-D. and Zhang, G.-Z.: A weight-based clustering algorithm for mobile ad hoc network, *Proc. Int. Conf. on Wireless and Mobile Communications*, p.3 (2007).

推薦文

本論文では，MANET を対象に，クラスタを用いた Top-k 検索のためのルーティング手法が提案されている．また，シミュレーション実験の結果から，提案手法が従来手法の性能を上回ることが示されている．既存の問題に対して新たな解を示した有用性の高い研究であり，本研究会からの推薦に値する．

（マルチメディア通信と分散処理研究会主査　勝本道哲）

天方 大地　（学生会員）

2012 年大阪大学工学部電子情報工学科卒業．現在，同大学大学院情報科学研究科博士前期課程在学中．分散環境におけるクエリプロセシング技術に興味を持つ．日本データベース学会の学生会員．

佐々木 勇和　（学生会員）

2009 年大阪大学工学部電子情報エネルギー工学科卒業．2011 年同大学大学院情報科学研究科博士前期課程修了．現在，同大学院情報科学研究科博士後期課程在学中．モバイル環境における検索技術に興味を持つ．日本データベース学会の学生会員．

原 隆浩　（正会員）

1995 年大阪大学工学部情報システム工学科卒業．1997 年同大学大学院工学研究科博士前期課程修了．同年同大学院工学研究科博士後期課程中退後，同大学院工学研究科情報システム工学専攻助手，2002 年同大学院情報科学研究科マルチメディア工学専攻助手，2004 年より同大学院情報科学研究科マルチメディア工学専攻准教授となり，現在に至る．工学博士．1996 年本学会山下記念研究賞受賞．2000 年電気通信普及財団テレコムシステム技術賞受賞．2003 年本学会研究開発奨励賞受賞．2008 年，2009 年本学会論文賞受賞．モバイルコンピューティング，ネットワーク環境におけるデータ管理技術に関する研究に従事．IEEE，ACM，電子情報通信学会，日本データベース学会の各会員．

西尾 章治郎 （フェロー）

1975 年京都大学工学部数理工学科卒業．1980 年同大学大学院工学研究科博士後期課程修了．工学博士．京都大学工学部助手，大阪大学基礎工学部および情報処理教育センター助教授を経て，1992 年大阪大学工学部教授，2002 年大学院情報科学研究科教授となり，現在に至る．その間，大阪大学サイバーメディアセンター長，大学院情報科学研究科長，理事・副学長を歴任．データベースシステムにおけるデータおよび知識管理に関する研究に従事し，紫綬褒章，立石賞功績賞等を授与される．日本学術会議会員．本会では理事を歴任し，論文賞，功績賞を受賞．IEEE，電子情報通信学会フェロー．