社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－MBL－43 (4)
2007－ITS－31 (4)
2007／11／21

アドホックネットワークにおける効率的なデータ転送に関する一考察

篠原 昌子　　原 隆浩　　西尾 章治郎

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
〒565–0871 大阪府吹田市山田丘 1–5
E-mail: {sinohara.masako, hara, nishio}@ist.osaka-u.ac.jp

アドホックネットワークでは，データの利用性を向上させるため，移動体に他の移動体のもつデータの複製を作成することが有効である．移動体が相互接続する移動体から自身のもつデータ（複製）を要求された場合，通常，そのデータを要求した移動体までユニキャストを用いて転送する．この場合，頻繁に要求されるデータがネットワーク上を何度も転送されるため，移動体は電力を多く消費してしまう．そこで本稿では，データの可用性を損なわず，移動体の生存時間を長くするため，トラヒック削減を考慮した効率的なデータ転送について議論する．提案方式では，移動体が，複数の移動体から要求されたデータをマルチキャストを用いてまとめて転送することで，トラヒックを削減する．

A Study on Effective Data Transmission in Ad Hoc Networks

Masako SHINOHARA　　Takahiro HARA　　Shojiro NISHIO

Dept. of Multimedia Eng., Grad. Sch. of Information Science and Technology, Osaka Univ.
1–5 Yamadaoka, Suita-shi, Osaka, 565–0871 Japan
E-mail: {sinohara.masako, hara, nishio}@ist.osaka-u.ac.jp

In ad hoc networks, it is effective that each mobile host creates replicas of data items held by other hosts for improving data availability. When a mobile host receives a data request from another mobile host, the mobile host usually transmits the requested data item by unicast. However, if mobile hosts hold data items that are frequently requested by others, they have to transmit the data items many times and consume a large amount of power. In this paper, we discuss effective data transmission for not only keeping data availability but also prolonging the lifetime of mobile hosts. In our proposed method, each mobile host collects multiple requests to data items and transmits the requested data items by multicast, and thus reduces the data traffic.

1　はじめに

近年，無線通信技術の発展と計算機の小型化・高性能化に伴い，ユーザが携帯型の計算機（移動体）を持ち歩き，時間や場所に関係なくネットワークに接続できる移動体計算環境が普及しつつある．特に，ルータ機能をもつ移動体のみで一時的な無線ネットワークを形成するアドホックネットワークへの関心が高まっている [1, 7]．アドホックネットワークでは，既存の通信インフラを必要とせずに，移動体同士で自律分散的にネットワークを構築できるため，緊急災害時の救助活動やイベント時の情報共有への利用が期待されている．ここで，アドホックネットワークでは，移動体の移動によりネットワークが分断されるため，分断された部分ネットワーク内のデータにアクセスできず，データの可用性が低下してしまう．例えば，図 1 の中央の無線リンクが切断された場合，左側の 3 台の移動体はデータ D_2 に，右側の 3 台の移動体はデータ D_1 にアクセスできなくなる．アドホックネットワークにおけるアプリケーションには，移動体同士でデータを共有し，互いのもつデータにアクセスするものも多い．したがって，データの可用性を向上させるため，移動体が他の移動体のもつオリジナルデータの複製を作成することが有効であり，これまでにいくつかの複製配置方式が提案されている [3, 4, 6, 9, 11]．

ここで，アドホックネットワークでは，移動体の電力容量に制限があり，データ転送による消費電力が全体の大きな割合を占める場合が多い [2] ため，データ転送の機会が多い移動体は電力を早く使い

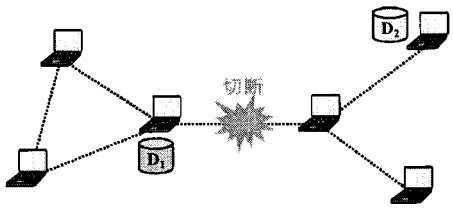


図 1: ネットワークの分断

果たしてしまう．このとき，電力を使い果たした移動体がネットワークから退出すると，その移動体のもつデータにアクセスできなくなる．さらに，ネットワークから退出した移動体が多くなると，ネットワークが疎になり，移動体が他の移動体と接続できる機会が減るため，データの可用性が低下してしまう．したがって，このような問題を改善するため，電力を使い果たす移動体の発生を抑制し，移動体の生存時間（電力を使い果たすまでの時間）をできるだけ長くすることは重要な研究課題である．

これまでに筆者らは，データの可用性の向上と移動体間の消費電力の均一化を目的として，移動体が，他の移動体からアクセスされる回数を均一化しつつ，自身や周囲の移動体が頻繁にアクセスするデータの複製を優先的に配置する複製配置方式を提案した [9]．また，電力を使い果たす移動体の発生を抑制することを目的として，電力残量の多い移動体からなる経路でデータを転送するデータアクセス方式を提案した [10]．しかし，これらの方式では，移動体が要求されたデータを要求した移動体までユニキャストを用いて転送するため，頻繁にアクセスされるデータは結局，ネットワーク上を何度も転送され，消費電力が大きくなるという問題があった．

そこで本稿では，データの可用性を損なわず，移動体の生存時間を更に長くするため，トラヒック削減を考慮した効率的なデータ転送について議論する．提案方式では，移動体が，複数の移動体から要求されたデータをマルチキャストを用いてまとめて転送することで，トラヒックを削減する．

以下では，2 章で関連研究について紹介し，本研究との比較を行う．3 章で想定環境について述べる．4 章でデータの転送方式について述べ，5 章で提案方式の特徴について考察する．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

近年，アドホックネットワークにおける複製配置やキャッシングに関する研究が盛んに行われている．

筆者らは，文献 [3] において，移動体の各データへのアクセス頻度とネットワークトポロジを考慮した複製配置方式を提案している．また，文献 [4] では，文献 [3] の方式を拡張し，データ更新の発生を考慮した複製配置方式を提案している．これらの方式は，移動体が多種類のデータにアクセスできるように，隣接移動体間や安定度の高いグループの移動体間でデータ（複製）の重複を解消して，データの可用性を向上させる．文献 [6] では，配置する複製の数を抑制することで，複製配置のコストを削減しながら，データ転送による遅延を軽減する方式を提案している．文献 [11] では，各移動体が他の移動体のアクセス状況を管理し，頻繁にアクセスされるデータの複製を配置することで，アクセス遅延の軽減とデータ転送によるトラヒックの削減を実現するキャッシング方式を提案している．

これらの複製配置方式は，複製を配置することで，データ転送に要するホップ数が小さくなるため，ネットワーク全体のトラヒックを削減できる．しかし，データを必ずユニキャストで転送するため，トラヒックを十分に削減しているとは言えない．一方，本研究では複製配置については特に想定していないため，これらの方式のアプローチを組み合わせて適用することが有効と考えられる．

一方，ルーティングに関する研究では，マルチキャストを用いたルーティング方式が数多く提案されている．文献 [8] では，AODV[7] を拡張し，マルチキャストに対応したツリー型のルーティング方式 MAODV(Multicast Ad Hoc On-Demand Distance Vector Protocol) を提案している．この方式では，マルチキャストグループごとに共有木を形成し，移動体が新たに参加する場合や，移動体の移動や離脱によってリンクが切断した場合には，木を再構成することで，グループ内の経路を常時維持する．また，文献 [5] では，マルチキャストグループ内の移動体およびそれらの移動体間の経路上の移動体を一つの転送グループとして扱う，メッシュ型のルーティング方式 ODMRP(On-demand Multicast Routing Protocol) を提案している．この方式では，明示的なメッセージなしにグループから離脱できるよう

に，経路生成やグループ内の移動体の管理をオンデマンドで行うことで，制御トラヒックを削減する．

これらのルーティング方式は，マルチキャスト通信を用いることで，トラヒックを削減できる．しかし，データ転送の前にマルチキャストグループを作成，管理しておく必要がある上，アプリケーションレベルでの効率化は行われていない．

3 想定環境

本稿では，協調作業における作業の効率化を図るため，数十から百台程度の移動体で構成される中規模なアドホックネットワークを利用して，モバイルユーザが情報共有を行う環境を想定する．例えば，イベントや展示会の会場では，企業などの団体がブースを出展し，来場者に製品やサービスを展示する．この場合，各団体のブースにある計算機は，製品やサービスに関する情報，ブースでのイベントスケジュール，および会場内で配布するファイル（壁紙やムービーなど）をデータとして保持する．また，来場者は自身のもつ小型端末を用いてこれらのデータにアクセスしながら，自身の行動を決定する．このような環境では，各データをユニキャストで転送すると，膨大なトラヒックによって帯域や電力を消費してしまう．

想定環境の詳細を以下に示す．

- アドホックネットワークを構成する移動体として，s 個の固定サーバ（識別子：$S_1, S_2, \ldots, S_s$）と，c 個の自由に移動するクライアント（識別子：$C_1, C_2, \ldots, C_c$）が存在する．上記の情報共有の環境例では，ブースにある計算機がサーバとなり，来場者のもつ小型端末がクライアントとなる．
- d 個のデータ（識別子：$D_1, D_2, \ldots, D_d$）が存在し，各々が特定のサーバにオリジナルデータとして保持されている．簡単化のため，全てのクライアントは，各サーバの識別子とその保持しているデータを把握しているものとする．
- サーバは，クライアントからのデータ要求に対して自身の保持するオリジナルデータを送信する．
- クライアントは，サーバの保持するデータにアクセスする．アクセス対象のデータを保持するサーバと相互接続している場合，データの取得時間のデッドラインを設定して，データを要求する．なお，本稿では，1 ホップ以上の無線リンクで接続し，互いに通信可能な状態を相互接続と定義する．

  また，簡単化のため，クライアントは複製配置を行わないものとする．なお，クライアントが複製を配置する環境では，5.4 節で述べるように提案方式を拡張し，要求されたデータをサーバと同様に転送する．

4 効率的なデータ転送

クライアントがサーバにデータを要求した場合，最も単純な方法として，サーバが要求されたデータをユニキャストを用いて即座に転送することが考えられる．しかし，この場合，サーバは頻繁に要求されるデータを何度も転送することになり，トラヒックが増加してしまう．

そこで本稿では，サーバは，クライアントからのデータ要求を一旦保留しておき，後からマルチキャストを用いて複数のクライアントにまとめて転送する．ここで，クライアントからのデータ要求にはデッドラインが設定されているため，サーバは，クライアントがデッドラインの時間内にデータを取得できるよう，データ転送を開始する時刻を決定する必要がある．また，サーバは，同じデータを要求している複数のクライアントにデータをまとめて転送できるよう，データ転送木を作成する必要がある．

以下では，まず，本稿でのデータアクセスの手順について説明する．次に，サーバがクライアントから要求されたデータの転送開始時刻を決定する方法，およびデータを複数のクライアントにまとめて転送するためのデータ転送木を作成する方法について説明する．最後に，作成したデータ転送木を用いたデータ転送の手順について説明する．

4.1 データアクセス

本節では，クライアントがサーバの保持するデータにアクセスする手順について説明する．

クライアントは，アクセス対象のデータを保持するサーバと相互接続していれば，データの取得時間のデッドラインを設定し，データを保持するサーバにデータ要求パケットを送信する．このパケット

には，自身の識別子，データを保持するサーバの識別子，要求したデータの識別子，データを要求した時刻，および設定したデッドラインが含まれる．また，クライアントは，そのログをデータ要求履歴表に記録する．この表は，要求したデータの識別子，データを要求した時刻，および設定したデッドラインを含む．

クライアントが，デッドラインの時間内にサーバからデータを取得できた場合，そのアクセスは成功となり，データ要求履歴表から該当するログを削除する．時間内に取得できなかった場合，そのアクセスは失敗となり，データ要求履歴表から該当するログを削除する．なお，データを保持するサーバと相互接続していない場合，データ要求は行わず，そのアクセスは即座に失敗となる．

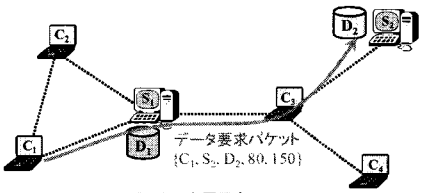
一方，データ要求パケットを受信したサーバは，4.2 節の方法に従ってデータ転送開始時刻を求める．その後，パケットに含まれるクライアントの識別子，データの識別子，要求時刻およびデッドラインに加え，求めたデータ転送開始時刻とパケットの転送経路を要求受付表に記録する．なお，パケットの転送経路は，IP の traceroute オプションなどを利用することで，パケットに付与されているものとする．

図 2 は，時刻 80 において，クライアント C_1 がデッドラインを時刻 150 に設定して，サーバ S_2 の保持するデータ D_2 を要求する動作を示す．各表は，データ要求パケットを送受信した後の，C_1 のデータ要求履歴表と S_2 の要求受付表に追加されるログを示す．

4.2 データ転送開始時刻の決定

サーバは，クライアントからデータ要求パケットを受信した場合，クライアントがデッドラインの時間内にデータを取得できるよう，データの転送を開始する時刻を決定する．サーバが転送を開始するまでの時間が長いほど，その時間内に同じデータを要求するクライアント数は増加すると考えられる．したがって，マルチキャストを用いたデータ転送によりトラヒックを削減する場合，データ転送開始時刻はできる限り遅い方が望ましい．

ここで，デッドラインまでの時間が長い場合，サーバがデータ要求を長時間保留できるため，デッドラインまでの時間が短い場合に比べて，データ転送開始時刻を遅く設定できる．また，データを要求したクライアントまでの経路のホップ数が短い場合や，要求されたデータのサイズが小さい場合，データの転送に要する時間が短いため，データ転送開始時刻を遅くできる．



クライアントのデータ要求履歴表

	データID	要求時刻	デッドライン
C_1	D_2	80	150

サーバの要求受付表

	クライアントID	データID	要求時刻	デッドライン	データ転送開始時刻	転送経路
S_2	C_1	D_2	80	150	128	$C_1 \rightarrow S_1 \rightarrow C_3 \rightarrow S_2$

図 2: データアクセス

そこで本稿では，次式を用いて，データ転送開始時刻 T_{st} を求める．

$$T_{st} = T_{dl} - \frac{1}{BW} \times size \times hop - T_{free}. \quad (1)$$

ここで，T_{dl} はデータ要求パケットに設定されたデッドライン，$size$ は要求されたデータのサイズ，hop はデータを要求したクライアントからサーバまでのホップ数を示す．また，BW は隣接する移動体（サーバおよびクライアント）の間で 1 秒間に転送可能なデータサイズ（平均転送効率）を示し，T_{free} はサーバのハードディスクアクセスや CPU 計算，4.4 節で用いる制御パケットの送受信，およびトポロジ変化への対処などに用いるための余裕時間を示す．つまり，式 (1) の右辺第 2 項は，サーバがクライアントにデータを転送するのに要する時間の予測値を示し，T_{st} はクライアントのデータアクセスが成功するために，サーバがデータを転送する最遅開始時刻を示す．

図 2 を用いて，データ転送開始時刻を決定する例を説明する．データ D_2 のサイズを $1[MB]$，$BW = 2[Mbps] = 0.25[MB/sec]$，$T_{free} = 10[sec]$ とすると，C_1 から S_2 までのホップ数は 3 ホップであるため，データ転送開始時刻 T_{st} は次のようになる．

$$\begin{aligned} T_{st} &= 150 - \frac{1}{0.25} \times 1 \times 3 - 10 \\ &= 128. \end{aligned}$$

4.3 データ転送木の作成

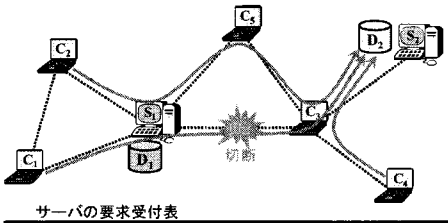
サーバは，あるデータへの要求がデータ転送開始時刻になった場合，同じデータを要求している他のクライアントにもデータをまとめて転送できるよう，自身を根とし，データ要求を行ったクライアントおよび同じデータを要求している他のクライアントを内部節点とするデータ転送木を作成する．データ転送木は，データ転送開始時刻となった要求の転送経路を初期状態とし，以下の手順によって，他の要求の転送経路を部分木として統合することで作成する．

1. サーバは，要求受付表から，未選択でデータ転送開始時刻までの時間が最も短い要求を選択する．
2. 選択した要求の転送経路上の移動体（サーバまたはクライアント）を，要求したクライアントから順に，データ転送木から探索し，最初に発見した移動体を重複移動体とする．
3. 重複移動体がサーバ自身でない場合，クライアントから重複移動体までの経路をデータ転送木に統合する．

   重複移動体がサーバ自身である場合，選択した要求の転送経路は，データ転送木に含まれるどの経路とも重複していない．このような要求は，まとめて転送してもトラヒックが変化しないため，この要求の処理を終了して手順 1 に戻る．
4. 重複移動体からサーバまでの経路が，データ転送木と選択した要求の転送経路において異なる場合，それぞれの経路を転送経路に含む要求の要求時刻を比較し，時刻の遅い方の経路に更新する．これは，移動体の移動によってネットワークトポロジが変化する場合，要求時刻からの時間が短い要求の転送経路の方が，最新のネットワークトポロジを反映している可能性が高いためである．

   また，この手順によって，データ転送木に統合されている他のどのデータ要求とも転送経路が重複しない要求が発生した場合，その要求の転送経路をデータ転送木から削除する．

図 3 と図 4 を用いて，サーバ S_2 がデータ転送木を作成する動作を説明する．図 3 の表は S_2 の要求受付表を示し，S_2 がクライアント C_1，C_2 および C_4 からデータ D_2 を要求されていることを示す．以下では，C_1，C_2 および C_4 からのデータ要求をそれぞれ要求 C_1，要求 C_2，および要求 C_4 と呼ぶ．



サーバの要求受付表

	クライアント ID	データ ID	要求時刻	デッドライン	データ転送開始時刻	転送経路
S_2	C_1	D_2	80	150	128	$C_1 \rightarrow S_1 \rightarrow C_3 \rightarrow S_2$
	C_2	D_2	115	180	152	$C_2 \rightarrow S_1 \rightarrow C_5 \rightarrow C_3 \rightarrow S_2$
	C_4	D_2	100	175	157	$C_4 \rightarrow C_3 \rightarrow S_2$

図 3: データ転送木の作成

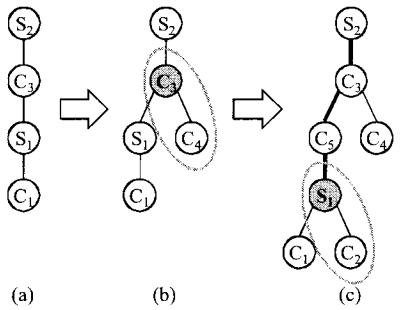


図 4: 転送経路の統合とデータ転送木の更新

S_2 は，時刻 128（要求 C_1 のデータ転送開始時刻）になると，図 4(a) のように，要求 C_1 の転送経路をデータ転送木の初期状態とする．まず S_2 は，データ転送開始時刻までの時間が最も短い要求 C_4 の転送経路から重複移動体 C_3 を発見し，クライアントから重複移動体までの経路 $C_4 - C_3$ をデータ転送木に統合する．図 4(b) は，要求 C_4 の処理が終わった後のデータ転送木を示し，灰色の丸は重複移動体，点線で囲まれた部分は統合された経路を示す．同様に，S_2 は要求 C_2 の転送経路から重複移動体 S_1 を発見し，経路 $S_1 - C_2$ を統合する．ここで，重複移動体 S_1 から S_2 までの経路が，データ転送木と要求 C_2 の転送経路において異なるため，より要求時刻の遅い要求 C_2 の転送経路 $S_1 - C_5 - C_3$ に更新する．図 4(c) は，要求 C_2 の処理が終わった後のデータ転送木を示し，太線は更新された経路を示す．

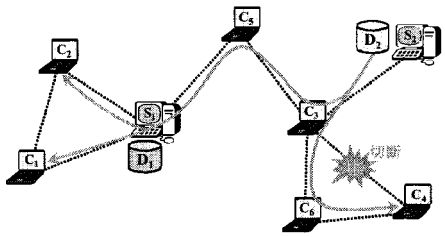


図 5: データ転送

4.4 データ転送

本節では，サーバが，4.3 節で作成したデータ転送木を用いて，データを転送する手順について説明する．

アドホックネットワークでは，移動体の移動によってネットワークトポロジが変化するため，データ要求パケットの転送経路がデータ転送時においても有効とは限らない．そこで，サーバは，データ転送木に含まれる経路が有効か否かを調べるため，データ転送木において自身の子となっている移動体（サーバまたはクライアント）に確認パケットを送信する．確認パケットには，自身の識別子と作成したデータ転送木が含まれる．

確認パケットを受信した移動体は，パケットに含まれるデータ転送木における自身の子に確認パケットを転送する．このとき，ネットワークトポロジの変化により送信先の移動体と接続していなければ，送信先の移動体を宛先とする経路探索を行う．経路を発見できた場合，データ転送木における自身から送信先の移動体までの経路を発見した経路に更新して，確認パケットを送信する．一方，経路を発見できなかった場合，データ転送木から送信先の移動体を根とする部分木を削除する．

データ転送木における自身の子がいない場合，および部分木の削除によりいなくなった場合，確認パケットに含まれるサーバの識別子およびデータ転送木を含む確認応答パケットを返信する．移動体は，全ての子から受信した確認応答パケットを受信すると，パケットに含まれるデータ転送木をまとめ，自身の親となる移動体に送信する．

サーバは，確認応答パケットを受信すると，データ転送木に沿ってデータを転送する．

図 5 は，図 6(a) に示すデータ転送木を作成したサーバ S_2 が，データを転送する動作を示す．ここで，クライアント C_3 は，データ転送木において自身の子となっている C_4 と直接接続していないため，データ転送木を経路探索によって発見した経路 $C_3 - C_6 - C_4$ に更新する．図 6(b) は，C_3 が経路探索によって経路を更新した後のデータ転送木を示し，太線は更新された経路を示す．

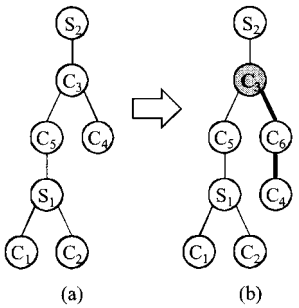


図 6: データ転送木の更新

5 考察

本章では，提案したデータ転送方式の考察を行う．

提案方式では，サーバが，設定されたデッドライン，データのサイズ，および転送経路のホップ数を用いて，データ転送開始時刻を決定することで，クライアントはデッドラインの時間内にデータを取得できる．また，サーバは，同じデータを要求している複数のクライアントを内部節点とするデータ転送木を作成することで，これらのクライアントにデータをまとめて転送して，トラヒックを削減できる．さらに，サーバがデータを転送する前に，データ転送木に含まれる無効な経路の検出と経路探索による更新を行うことで，無駄なトラヒックの発生を抑制できる．

以下では，まず，デッドラインの設定方法について考察する．次に，データ転送開始時刻の決定方法およびデータ転送木の作成方法の拡張について考察する．最後に，クライアントが複製を配置する環境における提案方式の拡張について考察する．

5.1 デッドラインの設定

本稿では，クライアントが，データの取得時間のデッドラインを設定して，サーバにデータを要求することを想定したが，デッドラインの設定方法については，特に限定していない．例えば，クライアン

トが開始時間の決まっているイベントの情報を要求する場合，デッドラインは，イベントの開始時間によって設定され，要求したクライアントや時刻には依存しない．一方，恒常的なイベントの情報を要求する場合，クライアントが要求を待機できる時間をデッドラインまでの時間と設定するため，デッドラインは要求したクライアントや時刻に依存する．

ここで，デッドラインは，4.2 節で説明したデータ転送開始時刻に大きく影響するため，デッドラインの設定方法によって適切なデータ転送開始時刻の決定方法が異なる可能性がある．そこで，今後は，様々なデッドラインの設定方法に対して，提案方式を評価し，必要があれば，転送開始時刻の決定方法を動的に変更するよう提案方式を拡張する．

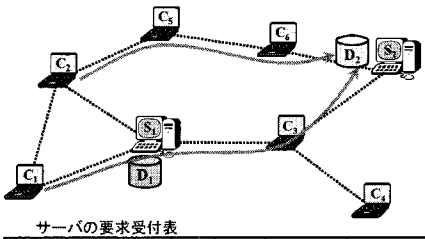
5.2 データ転送開始時刻の決定

提案方式では，クライアントのデータアクセスが成功するために，サーバがデータを転送する最遅転送開始時刻を予測し，これをデータ転送開始時刻とした．この時刻は，ネットワークトポロジの変化によって転送経路が無効となった場合に，4.3 節や 4.4 節の処理によって経路を更新するための余裕時間 T_{free} を含む．しかし，経路更新に要する時間やデータ転送に要する時間の増加分が，余裕時間を越えてしまった場合や，新たな経路を発見できなかった場合には，デッドラインの時間内にクライアントがデータを取得できず，アクセスが失敗してしまう．

ここで，移動体（サーバとクライアント）の移動速度が速い場合，時間が経過するほどネットワークトポロジが変化し，経路更新やデータ転送に要する時間が増加する可能性や，データを取得できない可能性が高くなる．一方，移動体の密度が高い場合，ネットワークトポロジが変化しても，新たな経路を発見できる可能性が高い．そこで，今後は，移動体の移動速度や密度を考慮した最遅転送開始時刻をデータ転送開始時刻とすることを考える．また，本稿で提案したデータ転送開始時刻の決定方法と組み合わせることで，トラヒックを削減しつつ，データアクセスの失敗を減らすデータ転送開始時刻を決定することを考える．

5.3 データ転送木の作成

本稿では，データ要求の転送経路が，データ転送木に含まれるどの経路とも重複していない場合のみ，データ転送木に統合しない．このようなデータ要求は，その要求がデータ転送開始時刻になったとき，改めてデータ転送木を作成する．ここで，転送経路が作成中のデータ転送木と最初の数ホップしか重複していないデータ要求は，作成中のデータ転送木に統合せず，データ転送開始時刻になったときに改めてデータ転送木を作成する方が，データ転送時のトラヒックを削減できる可能性がある．



サーバの要求受付表

	クライアント ID	データ ID	要求時刻	デッドライン	データ転送開始時刻	転送経路
S_2	C_1	D_2	80	150	128	$C_1 \rightarrow S_1 \rightarrow C_3 \rightarrow S_2$
	C_2	D_2	115	180	152	$C_2 \rightarrow C_3 \rightarrow C_6 \rightarrow S_2$

図 7: 位置情報を利用した経路統合

そのため，データ転送木へ統合するか否かは，経路の重複の有無だけでなく，経路の重複度合いも含めて検討する必要がある．また，データへの要求頻度やデッドラインなどを考慮して，データ転送木を改めて作成する場合に削減できるトラヒック量を予測する方法について検討する必要がある．

一方，データ要求パケットに，移動体の位置情報や隣接している移動体の情報を付加することで，トラヒックをより削減するデータ転送木を作成できる可能性がある．例えば，図 7 において，クライアント C_1 と C_2 からのデータ要求の転送経路は一切重複していない．そのため，C_1 からのデータ要求を初期状態として，本稿で提案したデータ転送木の作成方法を用いても，C_2 からの要求は統合されない．ここで，位置情報や隣接移動体情報によって，S_2 と C_2 の接続を検出できれば，S_2 を重複移動体とし，経路 $S_2 - C_2$ をデータ転送木に統合できる．

5.4 クライアントによるデータ転送

本稿では，データを保持するサーバのみが，クライアントから要求されたデータを転送することを想定した．しかし，クライアントが，取得したデータの複製を自身のデータ領域に作成している場合，他のクライアントから要求されたデータをサーバと同

様に転送することで，転送に要する時間やトラヒックを削減できるものと考えられる．

このとき，サーバとデータ（複製）を保持するクライアントが，同じデータを重複して転送しないよう，提案方式を拡張する必要がある．例えば，複製を保持するクライアントが，4.1節のデータ要求におけるデータ要求パケットや4.4節のデータ転送における確認パケットを転送しないことや，データを要求したクライアントが，データ取得時に，他のサーバやクライアントに記録されている要求受付ログの削除を依頼することが考えられる．また，データを要求したクライアントに近いサーバやクライアントが，データをを転送できるよう，転送開始時間を決定することも考えられる．

6 まとめ

本稿では，データの可用性を損なわず，移動体の生存時間を長くするため，トラヒック削減を考慮した効率的なデータ転送について議論した．具体的には，移動体が，複数の移動体からのデータ要求を一旦保留しておき，後からマルチキャストを用いてまとめて転送することで，トラヒックを削減する．提案方式では，移動体がデータをデッドラインの時間内に取得できるよう，データ転送を開始する時刻を決定する．また，同じデータを要求している複数の移動体にデータをまとめて転送できるよう，データ転送木を作成する．

今後は，5章で考察した事項について検討を加え，より効率的なデータ転送について検討する予定である．また，本稿で提案した方式の性能評価をシミュレーション実験によって行う予定である．

謝辞 本研究の一部は，文部科学省科学研究費補助金・基盤研究 (A)(17200006)，財団法人国際コミュニケーション基金，および特別研究員奨励費 (19-2204) 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] J. Broch, D. A. Maltz, D. B. Johnson, Y. -C. Hu, and J. G. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," Proc. IEEE/ACM MOBICOM'98, pp. 85–97, 1998.

[2] L. Feeney, and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," Proc. IEEE INFOCOM'01, pp. 1548–1557, 2001.

[3] 原 隆浩, "アドホックネットワークにおけるデータ利用性向上のための複製配置," 電子情報通信学会和文論文誌 B, Vol. J84-B, No. 3, pp. 632–642, 2001.

[4] T. Hara, "Replica Allocation Methods in Ad Hoc Networks with Data Update," ACM/Kluwer Journal on Mobile Networks and Applications (MONET), Vol. 8, No. 4, pp. 343–354, 2003.

[5] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-Demand Multicast Routing Protocol," Proc. IEEE WCNC'99, pp. 1298–1304, 1999.

[6] P. Nuggehalli, V. Srinivasan, and C.-F. Chiasserini, "Energy-Efficient Caching Strategies in Ad Hoc Wireless Networks," Proc. MobiHoc'03, pp. 25–34, 2003.

[7] C. E. Perkins, and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," Proc. IEEE WMCSA'99, pp. 90–100, 1999.

[8] E. M. Royer, and C. E. Perkins, "Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol," Proc. IEEE/ACM MOBICOM'99, pp. 207–218, 1999.

[9] 篠原昌子, 林 秀樹, 原 隆浩, 西尾章治郎, "アドホックネットワークにおける消費電力を考慮したデータ複製配置方法," 情報処理学会論文誌, Vol. 47, No. 1, pp. 15–27, 2006.

[10] 篠原昌子, 原 隆浩, 西尾章治郎, "モバイルアドホックネットワークにおける消費電力を考慮したデータ複製へのアクセス方式," 情報処理学会論文誌, Vol. 48, No. 12, 2007, 採録決定.

[11] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," Proc. IEEE INFOCOM'04, 2004.