複製による負荷分散を可能にしたP2Pプロトコルの評価

佐治　和弘†　　有次　正義††

† 群馬大学大学院工学研究科情報工学専攻　〒376-8515 群馬県桐生市天神町 1-5-1
†† 群馬大学工学部情報工学科　〒376-8515 群馬県桐生市天神町 1-5-1
E-mail: †kazuhiro@dbms.cs.gunma-u.ac.jp, ††aritsugi@cs.gunma-u.ac.jp

あらまし　P2P システムに対する効率的な負荷分散は重要な問題である．我々はこれまでに，SCOPE で提案されている複製の分散管理手法を BATON に組み合わせることで，人気データを持つノードのデータ送信負荷を，複製を持つノードに分散する P2P プロトコルを提案した．本稿では，人気の高いデータが存在する場合や，複製が存在するデータが頻繁に更新される場合を想定した環境のシミュレーションを実装し，評価する．提案手法と単純な複製の管理の手法との性能を比較することで，提案手法の有効性と問題点を検討する．
キーワード　ピアツーピア，複製，負荷分散

Performance Evaluation of a P2P Protocol with Load Balancing using Replicas

Kazuhiro SAJI† and Masayoshi ARITSUGI††

† Department of Computer Science, Graduate School of Engineering, Gunma University
1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan
†† Department of Computer Science, Faculty of Engineering, Gunma University
1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan
E-mail: †kazuhiro@dbms.cs.gunma-u.ac.jp, ††aritsugi@cs.gunma-u.ac.jp

Abstract　It is important to realize efficient load balancing mechanisms in P2P systems. We have proposed a P2P protocol by integrating a replica management method of SCOPE with BATON. Our protocol can achieve good load balancing for nodes having data with high request rates. In this paper, we evaluate our protocol under a simulation in some environments where there are some data with high request rates or high update rates by comparing with a simple replica management protocol for discussing the effectiveness and problems.
Key words　P2P, Replica, Load balancing

1. はじめに

Structured P2P システムの代表的な手法として，DHT を用いた手法がいくつか提案されており [1]，Chord [2]，Pastry [3] 等のシステムが開発されている．これらの DHT を用いた手法は，効率的なデータの配置方法と探索方法を提供する．また，BATON [4] では P2P のオーバーレイネットワークの構造に平衡 2 分木を用いており，DHT を用いた手法ではサポートできなかったレンジクエリのサポートが可能となっている．BATON では，各ノードは自律的に自身が受け持つ値の範囲を調節し，システム内の各ノードに掛かるクエリ数，アクセス数，データ数等の負荷を均一に保つことができる．しかし，多くのアクセスを受けるような人気データが存在する場合，人気データを管理するノードは過負荷状態となってしまい，値の範囲調節だけでは対応しきれなくなってしまう恐れがある．この問題の解決策の 1 つとして，人気のあるデータの複製を生成して他のノードに持たせ，負荷を分散する方法が考えられる．しかし複製が多数存在する場合を考えると，複製の管理は容易ではない．

また，eDonkey のトラフィックを計測した文献 [5] によると，download ストリームの平均サイズが 2.48M bytes なのに対し，non-download ストリームの平均サイズが 16.7K bytes と計測されており，データダウンロードに使われるデータのサイズはそれ以外のメッセージのサイズに比べて非常に大きく，総送信コストの大部分がデータ送信コストであることがわかる．このことから，他ノードに比べデータ送信量の多いノードや，更新情報データの送信量の多いノードが過負荷状態になることが考えられる．

我々は [6] で，BATON に SCOPE [7] で提案されている複製

の分散管理手法を組み合わせることで，効率的な複製の分散管理の手法と，複製を用いて人気データの送信負荷を分散する手法を提案した．データの更新が起きる場合でも, オリジナルデータと複製間の一貫性を保障することにより，複製を利用することが可能になる．さらに複製が多数存在するデータの更新が起きた場合でも，更新情報データの伝播を SCOPE で用いられている複製の分散管理手法を用いることで，更新情報データの送信コストが 1 ノードに集中することを避けることができる．

本稿では，我々が提案した手法の有効性と問題点を検討する．このために，人気の高いデータが存在する場合や，データが頻繁に更新される場合を想定した環境で，シミュレーションを行った．シミュレーションでは，データや更新情報データの送信と，それ以外のメッセージの送信とを区別する．これにより，データのサイズを意識しつつ，提案手法と単純な複製の管理の手法との性能の比較・評価を行う．さらに，提案手法において，データ送信負荷の分散や複製の一貫性の管理をより効率的に行うための実装方法の考察と，実験による比較・評価を行う．

2. 関 連 研 究

提案手法では，複製の位置情報を分散管理することで，更新処理の伝播を実現している．Gnutella や Winny は，Unstructured P2P システム上で，データを発見しやすくするために複製を利用しているが，複製の位置情報は管理しておらず，複製が更新されるような状況は想定されていない．Unstructured P2P システムにおいて，複製の位置情報が管理されていない場合，複製の更新をサポートするには，更新情報データのフラッディング等が考えられるが，複製を発見するためにトラフィックが増加してしまうことを考えると現実的ではない. [8] では，複製の位置情報の管理はしていないが，rumor spreading を基盤にした更新伝播アルゴリズムが提案されており，トラフィックの増加を抑えつつ，更新情報データの伝播を提供している．しかし，確率的な更新の伝播を提供しているため全ての複製が正しく更新される保障がなく，厳密な一貫性は提供していない．

P2P システムにおいて，複製の一貫性を提供しているシステムには [9] がある．ここではオリジナルデータを持つノードを根とし，複製を持つノードで更新伝播木という木構造を構築することにより複製を管理する．これにより，更新情報データ伝播にかかる負荷の分散と，複製更新にかかる遅延の減少を実現している．木構造のノードが全て複製を持つノードで構成されているので，複製の更新にかかる遅延が少なく，1 ノードが持つことができる子ノードの数を調整することによって，更新処理で 1 ノードにかかる負荷を調整することが可能である．しかし，複数の複製を持つノードが複数の更新伝播木の中間ノードに配置されると，更新伝播の負荷が偏ってしまう恐れがある．

本稿では，Structured P2P システムである BATON を基盤のオーバーレイネットワークとして利用している．Structured P2P システムでは，Unstructured P2P システムとは異なり，構造化されたオーバーレイネットワークを使用することによって，ネットワーク内にクエリにマッチするデータが存在するならば，そのデータを発見できることを保障する．Structured

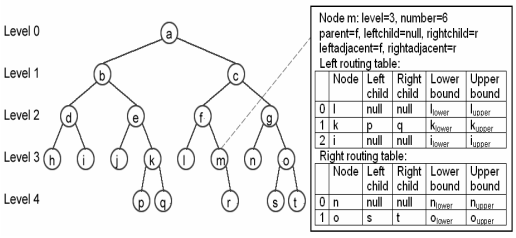

図 1　BATON の木構造とルーティングテーブル [4]
Fig. 1　BATON tree structure and routing table [4]

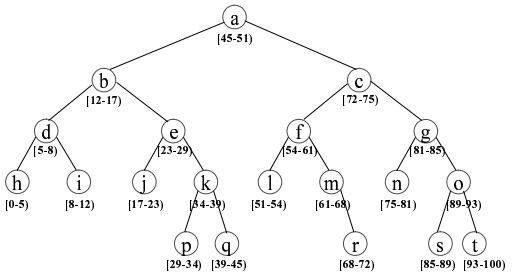

図 2　各ノードの値の範囲の例 [4]
Fig. 2　Example of range of value [4]

P2P システムで，複製を利用して性能の改善を目指している手法では，文献 [10] や文献 [11] が挙げられる．文献 [10] では複製を用いた Lookup 遅延の改善を目指し，Zipf 法則に基づいたクエリ分布に対して $O(1)$ の Lookup 性能を提供しており，データ更新時の複製の更新も提供している．文献 [11] ではファイルの人気度に応じた複製の作成やファイルの再配置を行うアルゴリズムを使用した DHT 基盤のファイル共有システムを提案している．しかし，どちらも検索効率を上げるための複製の配置方法を提供しているが，頻繁にデータの更新が起きるような環境を想定したものではなく，複製の厳密な一貫性の提供を考えたものではない．

3. 提 案 手 法

3.1 BATON の構造

提案手法では，オーバーレイネットワークに BATON [4] の平衡 2 分木の構造を用いる．図 1 に BATON の木構造の例とノード m が持つルーティングテーブルを示す．各ノードは親ノード，子ノード，左隣，右隣のノードへのリンクを持つ．また，各ノードは，右ルーティングテーブルと左ルーティングテーブルを持ち，同じ高さにおける左右の特定の位置のノードを管理する．

木構造内の全ての葉ノードと内部ノードには値の範囲を割り当てる．各ノードに割り当てられた値の範囲の例を図 2 に示す．各ノードは左隣のノードの最大値以上から右隣のノードの最小値未満の値の範囲を管理する．図中のノード e は，ノード j の最大値 23 以上，ノード p の最小値 29 未満の値の範囲を管理し，ノード q は，ノード k の最大値 39 以上，ノード a の最小値 45 未満の値の範囲を管理する．BATON のノードが，あるデータの識別子を自身の値の範囲に含んでいる場合，ノードはそのデータの Primary ノードであると言う．

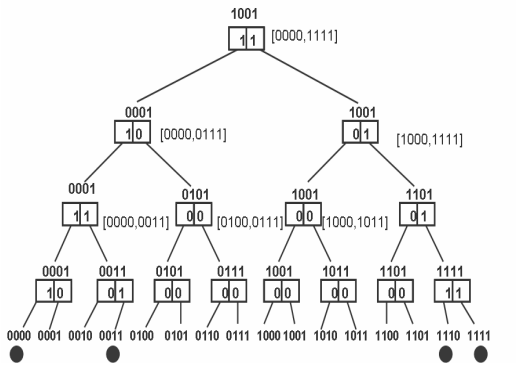

図 3　RPT の例 [7]
Fig. 3　Example of RPT [7]

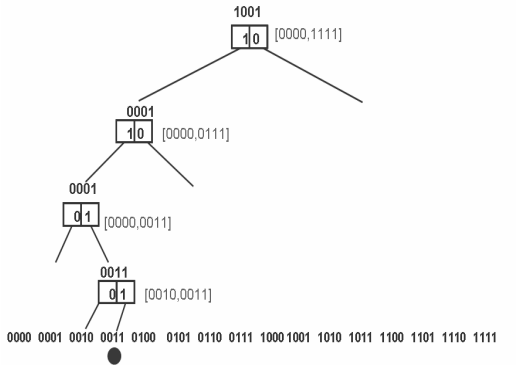

図 4　RPT の構築例 [7]
Fig. 4　Example of RPT construction [7]

3.2　SCOPE の分散管理手法と BATON への適用

提案手法では複製の分散管理を行うために先に説明した BATON の平衡 2 分木の他に，SCOPE [7] で用いられている RPT(replica partition tree) を使用する．SCOPE では RPT を用いて複製の管理を多数のノードに分散して行うことができるため，複製管理のための更新情報データの伝播処理や，複製の位置情報の維持コストを複数のノードに分散することが可能となる．以降に，複製の分散管理手法を RPT の説明と共に述べる．

4 ビットの識別子空間における，データ 9 = 1001 の複製が識別子 0000，0011，1110，1111 のノードに配置された時の，RPT の例を図 3 に示す．RPT は 1 データに対して 1 つ生成される．RPT の各ノードを代表ノード，ノードが持っている 2 ビットの値をパーティションベクトルと呼ぶ．SCOPE では識別子空間をパーティションに分割し，パーティションごとに代表ノードを決める．代表ノードの決定方法を，RPT の構築方法と共に説明する．以下 1 回の分割で得られるパーティションの数をパーティションサイズと呼ぶ．パーティションベクトルの長さはパーティションサイズと等しくなる．

例として図 4 に，パーティションサイズを 2 とし，データ 9 = 1001 の複製が識別子 0011 のノードに複製が配置された時の RPT の構築方法を示す．また，各代表ノードが管理するパーティションの範囲を代表ノードの右側に表す．まず，複製を持つノードは RPT の葉ノードとなる代表ノードを探す．識別子空間が m ビットで表現される時，各ノードとデータは m ビットの識別子で表現される．パーティションサイズを 2^n とした時，複製を持つノードは，上位 $(m-n)$ ビットを複製が配置された識別子，下位 n ビットにオリジナルデータの識別子を付けることで，RPT の葉代表ノードの識別子を得る．例では複製は 0011 に配置されたので，0011 の上位 $3(4-1)$ ビットを固定し，下位 1 ビットをデータの識別子 1001 と同じにすることで，葉代表ノードの識別子 0011 が得られる．この代表ノードは，識別子 0010〜0011 における複製の有無をパーティションベクトルを使って管理しており，パーティションベクトルの右側のビットに 1 を立てることで，識別子 0011 に複製が存在していることを示している．次に代表ノードは自身の識別子の上位 2 ビットを固定し，下位 2 ビットをデータの識別子 1001 と同じにすることで，RPT の次のレベルの代表ノード 0001 を得る．この代表ノードは，識別子 0000〜0011 における複製の有無をパーティションベクトルを使って管理しており，パーティションベクトルの右側のビットに 1 を立てることで，識別子 0010〜0011 に複製が存在していることを示している．同様の処理を繰り返し，最終的には RPT のルートノード 1001 を得ることができる．RPT のルートノードである 1001 は，オリジナルデータと同じ識別子を持ち，識別子空間全体における複製の有無をパーティションベクトルを使って管理している．例では，左側のパーティションベクトルに 1 を立てることで識別子空間 0000〜0111 に複製が存在することを示している．

RPT のルートに近い代表ノード程広い範囲における複製の有無を管理しており，葉ノードに近い代表ノード程小さな範囲における複製の有無を管理している．RPT の葉代表ノードのパーティションベクトルは，各識別子における複製の有無を直接示している．各パーティションベクトルは，代表ノードの識別子に従って，基盤である BATON の木構造に割り当てられる．図 4 のパーティションベクトルを BATON の各ノードに割り当てた例を図 5 に示す．RPT には図 4 の識別子 0001 のように，複数のパーティションレベルにおいて代表ノードとなる識別子が存在する．これにより，同一データに関する複数のパーティションベクトルを，1 つの識別子に割り当てられる状況が起きるため，図 5 の BATON のノード d のようにパーティションベクトルを複数持つようなノードが現れる．これにより，パーティションベクトルによるストレージコストの偏りが起き，ノード d のようなノードは，複製更新時に他の代表ノードよりも多くの更新情報データの送信を行う可能性がある．しかし，このような識別子を持つノードは，ノード間の通信を行わずに複数のパーティションベクトルを参照できるメリットも持つ．また，RPT はデータごとに生成されるため，複数のデータの RPT が生成されることを考えると，複製管理のための負荷は各ノードに分散されると考えられる．

3.3　オペレーション

以下，提案手法で使われるオペレーションのうち，本稿の議論に関係するものについて説明する．詳細は文献 [6] を参照されたい．

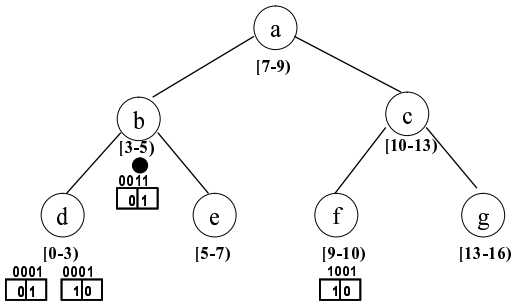

図 5　BATON の木構造とパーティションベクトルの対応
Fig. 5　BATON tree structure and its partition vectors

3.3.1　Subscribe/Unsubscribe

複製を持つノードをそのデータの Subscriber と呼ぶ．Subscribe オペレーションは，Subscriber が複製の位置を Primary ノードに知らせるために行う処理である．本稿ではデータを受信した際に，検索要求のクエリを発行したノードに複製を配置する．この手法はオーナ複製法と呼ばれており [12]，Gnutella でも用いられている方式である．Subscriber ノードは自身の識別子の下位 n ビットをオリジナルデータの識別子と一致させることで，RPT の葉ノードとなる代表ノードを探す．図 4 において，識別子 0010 がデータ 1001 の複製を持って，新たに Subscriber となった時，ノード 0010 は葉代表ノードであるノード 0011 に Subscribe オペレーションを送る．オペレーションを受け取った代表ノード 0011 は，パーティションベクトルの左側のビットに 1 を立てる．Subscribe 処理は，代表ノード 0011 のような既に 1 の立っているパーティションベクトルを持つ代表ノード，または RPT のルート代表ノードに辿り着くまで，親代表ノードを辿って続けられる．ノード 0010 が複製を消去する時は Unsubscribe 処理を実行する．ノード 0010 は代表ノード 0011 を探索し，ノード 0011 はパーティションベクトルの左側のビットを 0 にする．Unsubscribe 処理は，パーティションベクトルに 2 つ以上 1 が立っている代表ノード，または RPT のルート代表ノードに辿り着くまで，親代表ノードを辿って続けられる．

3.3.2　Exact Match Query

データ v を探索する Exact match query を発行，または受け取ったノードは BATON の Search exact アルゴリズムに従ってデータ v の探索を行う．探索は，v の値を範囲内に含むノード，もしくは v の複製を持つノードに辿り着くまで行われる．システム内のノード数が N の時，探索に必要なステップ数は $O(\log N)$ である．探索後，ノードはデータ送信要求を発行したノードに，要求されたデータを返す．ここで Primary ノードが過負荷状態になっている時は，データの送信作業を Subscriber に頼む．Primary ノードはまずシステム内におけるデータの複製の有無をパーティションベクトルから確認し，複製が存在すれば下位の代表ノードにデータ送信要求を転送する．要求を受けた代表ノードは，パーティションベクトルから Subscriber の位置を判断し，下位の代表ノードにデータ送信要求を送る．データ送信要求は RPT を辿り Subscriber まで届けられる．

各代表ノード間のデータ送信要求の送信には Search exact を使って，$O(\log N)$ ノードを経由して送られる．RPT を辿って Subscriber を見つけるために，Search exact を送信する回数は平均で $O(\log N)$ なので，Subscriber 発見のために必要となるホップ数は，平均で $O(\log^2 N)$ となる．Exact match query による探索が，データの複製を見つけないまま Primary ノードに辿り着いた場合，Primary ノードはデータ送信要求を Subscriber に送るか，自身が要求されたデータの送信を行うか判断する．Subscriber に送信要求を送る場合，複製発見までに多くのノードを経由する必要があるので，Primary ノードが過負荷状態である場合以外では，Primary ノードがデータの送信を行うのが望ましい．本稿では単位時間あたりのデータ送信回数が閾値 A を越えた Primary ノードは，A + 1 回目以降のデータの送信処理の a % を Subscriber に行わせることで負荷を分散する．

代表ノードのパーティションベクトルに複数 1 が立っている時は，RPT の高さに注目し，複数ある下位の代表ノードの中から，最も下位へ進める代表ノードを選ぶ．図 3 の RPT において，BATON のノードが識別子 1001〜1011 を値の範囲に持っていて，Primary ノードから Subscriber を探す場合，自身の持つ 5 つのパーティションベクトルの中から，ビット 1 が立っていて，最も下位にある，範囲 [1000,1111] を管理する代表ノード 1001 のパーティションベクトルを参照し，下位の代表ノード 1101 にデータ送信要求を送る．これにより，Subscriber までのホップ数を最小限に抑えることができる．ここで，複製を持つ Subscriber の負荷も考慮したデータ送信負荷の分散を考慮する．上で挙げた RPT の高さに注目した Subscriber の探索方法では，RPT を効率的に辿ることが可能である．しかし，毎回同じ Subscriber に送信要求を送ることになるので，1 つの Subscriber に複製の送信負荷が集中してしまう．本稿では，Subscriber は一定回数以上複製を送信したらその複製を削除し，負荷が集中することを避けることとした．また，本稿では RPT の高さに注目した Subscriber の探索方法の他に，パーティションベクトルに 1 が複数立っていたら，次に進む代表ノードをランダムに選ぶ手法を実装し，両手法の評価，比較も行った．ランダムに代表ノードを選ぶ手法では，RPT の高さに注目した手法に比べて経由するノード数が増えるが，負荷が特定の Subscriber に集中するのを避けることができる．

3.3.3　Update

更新処理は，Primary ノードがデータを更新した時，または Primary ノードが Subscriber からデータの更新要求を受け取った時に始まる．更新情報データは Primary ノードから RPT を辿り，全ての Subscriber に送られる．要求を受け取った各レベルの代表ノードはパーティションベクトルの値に応じて下位レベルの代表ノードに要求を送る．RPT を使うことにより Primary ノードは複製の位置の管理コストをパーティションベクトル維持のみに抑えることができ，更新情報も 1 レベル下位の代表ノードに送るだけで済むため，Primary ノードにかかる更新情報データの送信負荷を分散することができる．ここで，各代表ノード発見のために行われる探索は，Search exact を

使ってメッセージのやり取りのみで行われる．メッセージに比べ，データサイズの大きい更新情報データの伝播は，代表ノードが発見された後，代表ノード間で直接やり取りされる．これにより更新情報データの伝播コストを代表ノード間のやり取りだけで抑えることができる．

また，データの人気に偏りがある時，オーナ複製法により人気データの複製が大量に生成されてしまう可能性がある．複製が増えた時に，更新が頻繁に発生する状況を考えると，複製の一貫性の管理は容易ではない．本稿では，全ての複製に更新情報データを送ることで複製の一貫性を管理する手法と，データの更新が発生したら，一部の複製は更新し，残りの複製を無効化する方法を考える．SCOPE の分散管理手法では，システム全体の複製の数は把握できないので，Update が起きた際に，代表ノードのパーティションベクトルに 1 が立っていたら，一定確率で Update 処理を Deletion 処理に変換し，現在の代表ノードより下位の代表ノードに Deletion 処理を伝播することで，一部の複製を削除する．これにより，更新の際に一部の複製を削除し，複製数の増加を防ぐ．複製が削除されたら，その位置から Unsubscribe 処理を実行し，対応するパーティションベクトルを 0 にする．

3.3.4 Deletion

削除対象となるデータの発見には，BATON のオペレーションを使用する．データの削除が行われた時，Primary ノードはネットワーク内に存在するデータの複製を削除する必要があるので，削除命令を送信する．Primary ノードは RPT を辿り，全ての Subscriber に通知を送る．通知を受け取った各レベルの代表ノードはパーティションベクトルの値から複製の位置を判断し，下位レベルの代表ノードに削除命令を送信する．削除命令を受け取った Subscriber は，複製を削除した後，Unsubscribe 処理を実行する．

4. 評価実験

提案手法の有効性を示すためにシミュレータを用いて評価実験を行い，提案手法で示した複製を用いた手法，BATON で提案されている複製を用いない手法，単純な複製管理手法におけるパフォーマンスを比較した．

4.1 実験環境

実験では，0〜1023 の識別子空間中に 500 個のノードと 1000 個のデータを配置した．各データには 0〜1023 のいずれかの識別子をランダムに割り当て，データの識別子に重複はないものとする．また，各ノードが所持できる複製数の上限は 10 とし，上限を超えた場合は，LRU によって最も参照されていない複製を削除するものとする．データサイズについて説明する．メッセージのサイズが 1 であるのに対し，各データのサイズは 1〜20 の一様分布で，更新情報データのサイズは 1〜50 の一様分布で決まるものとした．以上の状態で Exact eatch query を発生させる．データ k に対する Exact match query の発生確率 Q_k は Zipf 法則 [13] に従うものとした．式を以下に示す．

$$Q_k = \frac{k^{-\alpha}}{\sum_{m=1}^{1000} m^{-\alpha}}$$

実験では $\alpha = 1$ に設定した．以下，各手法の説明とパラメータを示す．

4.1.1 単純な複製管理手法

単純な複製管理手法を説明する．複製の配置には提案手法と同様に，オーナ複製法を用いる．単純な複製管理手法では，Primary ノードが Subscriber の IP アドレスを保持することで，全ての複製の位置を管理する．Primary ノードは自身のデータを送信したら，送信先ノードの IP アドレスを Subscriber のアドレスとして登録する．また，Subscriber がデータの送信を行った場合は，Primary ノードにデータ送信先ノードの IP アドレスを送り，新たに複製が配置されたことを報告する．

実験では，人気データを持つノードが過負荷状態になることを避けるため，Primary ノードは 5 回目以降のデータ送信処理の 75% を Subscriber に任せることとした．Subscriber にデータ送信処理を任せる場合，Primary ノードは，自身の持つ Subscriber の IP アドレスのセットの中からランダムに 1 つを選び，データ送信依頼のメッセージを送る．更新が起きた場合は，Primary ノードから全ての Subscriber へ更新情報データの伝播が行われる．単純な複製管理手法では，Primary ノードから Subscriber へは 1 ホップで移動できるため，複製発見のためのホップ数を低く抑えることができるが，複製が多数存在する場合は，Primary ノードに負荷が集中する可能性がある．

4.1.2 提案手法

単純な複製管理手法と同様に，Primary ノードは 5 回目以降のデータ送信処理の 75% を Subscriber に任せる．Subscriber の発見は RPT を用いて行われる．RPT のパーティションサイズは 4 とした．ここで，RPT の高さに注目した Subscriber の探索方法と，パーティションベクトルに 1 が複数立っていたら，次に進む代表ノードをランダムに選ぶ Subscriber の探索方法を実装し，両手法の比較・評価も行った．さらに，提案手法では送信負荷の偏りを防ぐため，10 回以上送信処理に使われた複製は削除するものとした．

4.2 データ送信量の比較

各手法で Exact match query を 5000 回発生させた時の，各ノードが送信したデータサイズの総量を図 6，7，8，9 に示す．図 6 は複製未使用時に各ノードが送信したデータ送信量である．人気データを持つノードのデータ送信量が他に比べ極端に大きくなっていることがわかる．これに対して複製を用いた 3 手法によるデータ送信量は，Subscriber を利用してデータ送信作業を分散することにより，ノード間のデータ送信量の偏りが解消されていることがわかる．また単純な複製管理手法と提案手法では，どちらも 5 回目以降のデータ送信の 75 % を Subscriber に任せるため，データ送信量の偏りに関しては，ほとんど差が見られないが，提案手法の間では，同じ Subscriber に毎回送信要求が送られる RPT の高さに注目した手法に比べ，RPT をランダムに辿る手法では，全ての Subscriber に均等に送信要求が送られるため，送信量の偏りに多少の改善が見られた．

複製を用いることによるシステムの変化を表 1 にまとめる．ここでは，複製未使用時を N，単純な複製管理手法を S，提案手法の，RPT の高さに着目した手法を H，RPT をランダム

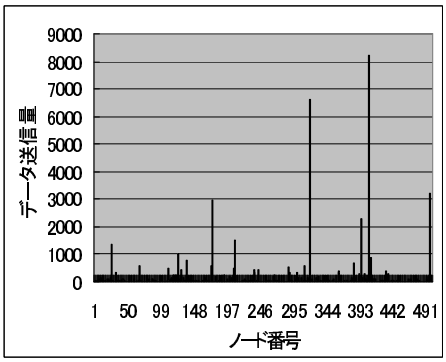

図 6　複製未使用時の各ノードのデータ送信量
Fig. 6　Amount of data sent by each node in BATON

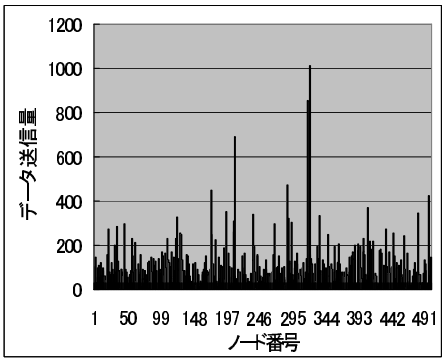

図 7　単純な複製管理手法における各ノードのデータ送信量
Fig. 7　Amount of data sent by each node in simple replication

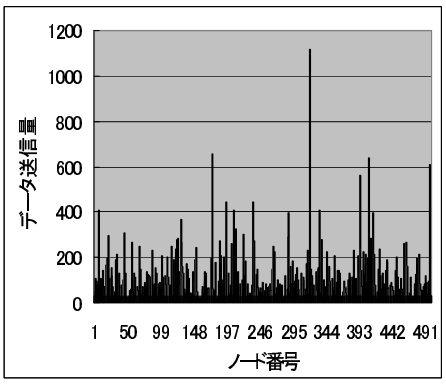

図 8　提案手法における各ノードのデータ送信量 (RPT の高さに注目した時)
Fig. 8　Amount of data sent by each node in our proposal(minimal hop traversal)

に辿る手法を R とし，単純な複製管理手法での複製アドレス数の平均，提案手法でのパーティションベクトル数の平均をストレージ平均に示した．複製を利用する手法では，途中で複製を持つノードを偶然見つけるケースがある．しかし，Search exact オペレーションが 複製を見つけないまま Primary ノードまで辿りついた時，Primary ノードが過負荷状態ならば，さらに Subscriber を探しに行く．ここで Subscriber まで 1 ホップで移動できる単純な複製管理手法では，ホップ数の最大値はそれほど変化せず，ホップ数の平均値は他の手法に比べて小さく

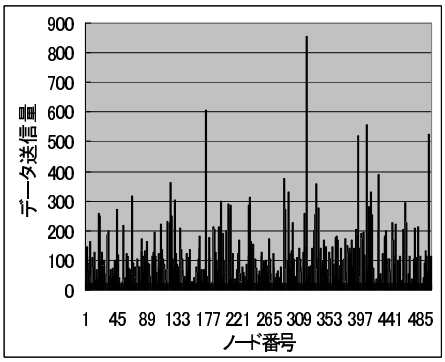

図 9　提案手法における各ノードのデータ送信量 (RPT をランダムに辿った時)
Fig. 9　Amount of data sent by each node in our proposal(random traversal)

表 1　実 験 結 果
Table 1　Experimental results

	N	S	H	R
ホップ数平均値	4.50	3.40	4.37	4.52
ホップ数最大値	14	14	28	28
ストレージ平均	0	7.86	18.1	18.0

なった．これに対して提案手法では，RPT を辿って Subscriber を探索する必要があるので，ホップ数の最大値が他の手法に比べて大きくなっており，さらにランダムに RPT を辿る手法では，RPT の高さに注目した手法に比べて経由するノード数が増えるため，ホップ数の平均値が高くなった．表 1 のストレージ平均から，複製未使用時には必要のない管理コストが新たに発生していることがわかる．また，単純な複製管理手法では，提案手法のように，複数のノードに分散しないで複製のアドレスを直接管理するため，全体におけるストレージコストの平均は低く済む．次にストレージコストの偏りについて考える．単純な複製管理手法と提案手法における，各ノードが管理する複製アドレス数と，パーティションベクトル数の分布を図 10，11 に示す．オーナ複製法では，人気データの複製が多数生成される．単純な複製管理手法では，多数の複製の位置を 1 ノードで管理するため，負荷に偏りが起きてしまう．これに対し，提案手法ではパーティションベクトルによって複製の位置を分散管理するため，ストレージコストに偏りがなくなっていることがわかる．

4.3　更新情報データの伝播

以降，提案手法には RPT の高さに注目した Subscriber の探索手法を用いる．データの更新が起きた時のパフォーマンスを提案手法と単純な複製管理手法で比較する．Exact match query50 回おきに，データの更新を 1 回発生させ，合計 5000 回の Exact match query と，100 回の Update オペレーションを実行する．データ k に対する Update の発生確率は $\alpha = 1$ の Zipf 法則に従うものとした．各ノードがデータ更新のために，送信した更新情報データの総量を図 12，13 に示す．単純な複製管理手法において，最も更新情報データ送信量の多かったノードの送信量が 88448 なのに対し，提案手法では，最も

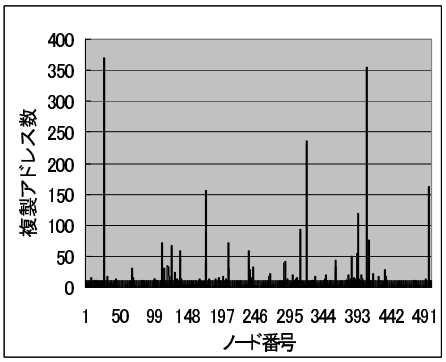

図 10　単純な複製管理手法における各ノードの複製アドレス数
Fig. 10　Number of replica addresses in simple replication

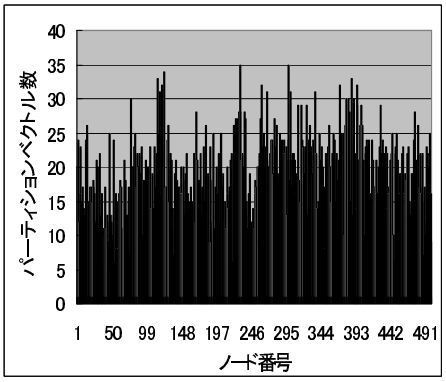

図 11　提案手法における各ノードのパーティションベクトル数
Fig. 11　Number of partition vectors in our proposal

多くの送信を行ったノードでも送信量が 4653 と送信量の偏りが改善されていることがわかる．これは，単純な複製管理手法では，オリジナルデータの Primary ノードが，全ての複製の Subscriber に対して更新情報データを送信するのに対し，提案手法では，Primary ノードは RPT の下位の代表ノードに更新情報データを送信するだけで済むため，更新情報データの送信負荷を RPT の各代表ノードに分散することができるためである．その反面，単純な複製管理手法においては各ノードの平均送信量が 412 なのに対し，提案手法は各ノードの平均送信量は 789 である．提案手法では複製に更新情報データを伝播する際に，複数の RPT 代表ノードを経由するため，システム全体を見るとトラフィックの量が増加してしまう．

提案手法の更新情報データの伝播では，全ての複製に更新情報を伝播するために，RPT を辿って移動する必要がある．Primary ノードから 1 つの Subscriber に辿りつくまでに経由したノードの数を計測した所，平均が 13.1 ノードで，最大で 34 ノードとなった．単純な複製管理手法が，直接 Primary ノードと Subscriber ノードで更新情報データのやりとりができるのに対して，提案手法では，伝播のために多くのノードを経由する必要があることがわかる．しかし，Primary ノードから Subscriber に辿りつくまでに行われた，更新情報データの伝播回数は，平均で 3.6 回，最大でも 5 回となった．提案手法では，代表ノードの発見はメッセージのやり取りのみで行い，サイズの大きい更新情報データのやり取りは，代表ノード間で行うため，メッセージを経由するノード数は多くなるが，実際に更新情報データの送信を行うノードの数は低く抑えることができる．

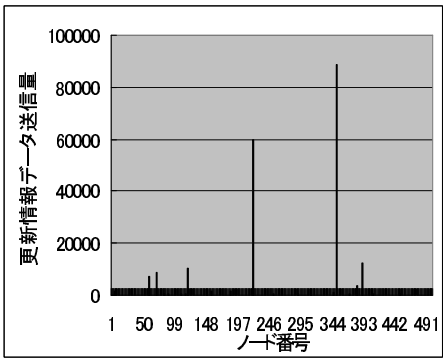

図 12　単純な複製管理手法における各ノードの更新情報データ送信量
Fig. 12　Amount of update data sent by each node in simple replication

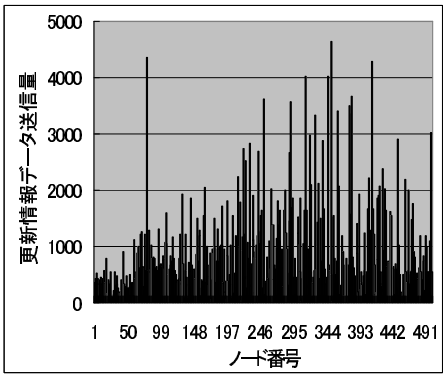

図 13　提案手法における各ノードの更新情報データ送信量
Fig. 13　Amount of update data sent by each node in our proposal

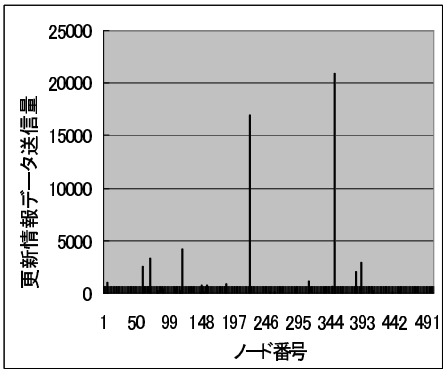

図 14　一部の複製を削除した時の単純な複製管理手法における各ノードの更新情報データ送信量
Fig. 14　Amount of update data sent by each node in simple replication (partial replica update)

本稿では，複製を置くことで，Primary ノードにかかるデータ送信負荷の分散と，人気データの探索効率の向上を実現した．しかし，オーナ複製法により人気データの複製が大量に生成されてしまった時，データが頻繁に更新される環境を考えると，多くの複製に対して更新情報データを送らなければならないため，複製の一貫性管理は容易ではない．ここではデータの更新

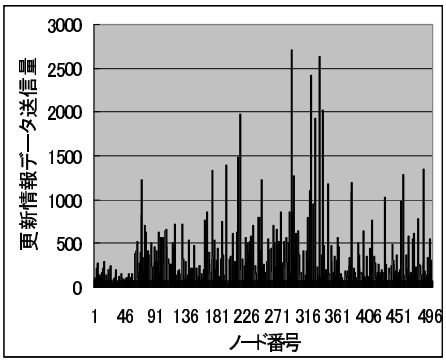

図 15　一部の複製を削除した時の提案手法における各ノードの更新情報データ送信量
Fig. 15　Amount of update data sent by each node in our proposal(partial replica update)

頻度に応じて，複製を減らす方法を考え，実装を行った．単純な複製の管理手法では，データの更新が起きる度に 6 割の複製には更新情報データを送信し，4 割の複製は削除する．また，RPT を用いた提案手法では，更新情報データを代表ノードが受け取ったら，10 % の確率で Update 処理を，Deletion 処理に変換し，その代表ノードが管理しているパーティションの複製を削除する．実験では RPT の高さは 5 なので，各代表ノードで 10 % の確率で削除命令への変換を行うと，複製が削除されずに更新される確率は，0.9^5 となり，残りの約 4 割の複製が削除される．以上の方法で実験を行った時の，各ノードが送信した更新情報データの送信量を図 14，15 に示す．全ての複製の更新を行っていた図 12，13 に比べ，1 部の複製を削除すると，更新情報データ伝播にかかる負荷を小さく抑えることが可能となる．このように，複製の削除を行えば，複製の一貫性管理のための負荷を抑えることができるが，その反面，複製数の総量が減るため，データ発見のためにかかるホップ数の増加等が起こってしまう．これらのバランスを最適にするため，今後はデータの人気度，データのサイズ，更新頻度，更新情報データのサイズ等から，データ要求に対するレスポンスと各ノードにかかる複製管理のための負荷が適当になるような，最適な複製数について考察する必要がある．

5. まとめと今後の課題

本稿では，我々が提案した BATON に SCOPE の分散管理手法を適用する手法の有効性と問題点を検討するために，人気の高いデータが存在する場合や，複製を持つデータが頻繁に更新される場合を想定した環境で，シミュレーションによる評価を行い，提案手法と単純な複製の管理の手法との性能を比較した．比較の結果，提案手法では，複製の更新情報の伝播にかかる更新情報送信コストや，複製を維持するために必要なストレージコストを各ノードに分散し，人気データを持つノードに偏る負荷を解消できることが確認できた．しかし，提案手法では，更新情報の伝播のためにシステム全体にかかるトラフィック量が増加してしまう問題点がある．今後はトラフィック量の増加を抑えるための，最適な複製数の検討が必要となる．

さらに，今後の課題として複製の配置方法と，複製の置き換え方法の改善が挙げられる．現在は，オーナ複製法と LRU を用いた単純な複製の配置や複製の入れ替えしか行っていないが, [10] や [11] では，Lookup の性能を上げるための，DHT 上における複製の配置や置き換え方法について検討されている．今後はこれらの手法と，複製の分散管理手法との組み合わせを考え，より効率的に複製を利用することを検討する必要がある．

謝　　辞

本研究の一部は，科学研究費補助金基盤研究 (C)(18500073) により行なわれた．

文　　献

[1] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris and I. Stoica: “Looking up data in P2P systems”, CACM, **46**, 2, pp. 43–48 (2003).
[2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan: “Chord: A scalable peer-to-peer lookup protocol for internet applications.”, IEEE/ACM Trans. Networking, **11**, 1, pp. 17–32 (2003).
[3] A. I. T. Rowstron and P. Druschel: “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems”, Middleware, pp. 329–350 (2001).
[4] H. V. Jagadish, B. C. Ooi and Q. H. Vu: “BATON: A balanced tree structure for peer-to-peer Networks”, VLDB, pp. 661–672 (2005).
[5] K. Tutschku: “A measurement-based traffic profile of the eDonkey filesharing service”, Proc. PAM2004, LNCS 3015, pp. 12–21 (2004).
[6] 佐治, 有次：“複製による負荷分散を可能にした P2P プロトコルの提案”, 日本データベース学会論文誌 (DBSJ Letters), **5**, 2, pp. 9–12 (2006).
[7] X. Chen, S. Ren, H. Wang and X. Zhang: “SCOPE: Scalable consistency maintenance in structured P2P systems”, INFOCOM, pp. 1502–1513 (2005).
[8] A. Datta, M. Hauswirth and K. Aberer: “Updates in highly unreliable, replicated peer-to-peer systems”, ICDCS, pp. 76–85 (2003).
[9] 中通, 内田, 原, 前田, 西尾：“Peer-to-Peer ネットワークにおける木構造を用いた複製更新の伝播について”, 電子情報通信学会第 15 回データ工学ワークショップ（DEWS 2004） (2004).
[10] V. Ramasubramanian and E. G. Sirer: “Beehive: O(1) lookup performance for power-law query distributions in peer-to-peer overlays”, NSDI, pp. 99–112 (2004).
[11] J. Kangashrju, K. W. Ross and D. A. Turner: “Adaptive content management in structured P2P communities”, First International Conference on Scalable Information Systems (2006).
[12] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker: “Search and replication in unstructured peer-to-peer networks.”, SIGMETRICS, pp. 258–259 (2002).
[13] G. K. Zipf: “Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology”, Addison-Wesley (1949).