DHT におけるノードの動作履歴を用いたデータ管理手法の提案と実装

野口　悟[†1]　猪俣敦夫[†1]
藤川和利[†1]　砂原秀樹[†1,†2]

DHT による構造化オーバレイネットワークは優れた資源探索能力を有する反面、ノードのネットワークへの頻繁な参加／離脱（churn）に伴うネットワーク構成の変動に影響されやすく、一旦格納したデータの消失、ノード探索の失敗といった問題を引き起こす.

そこで、本論文では自ノードの参加／離脱動作の履歴 (セッション情報) を収集し、将来のセッション持続時間を予想することで相手ノードの動作に応じた動的な複製数設定を行い、データ消失耐性を向上させるとともにデータ維持コストを減少させる手法を提案する．提案手法は既存の DHT のルーティングアルゴリズムに付加する形で実装し、性能評価を行った．その結果、提案手法によって相手ノードのセッション持続時間に応じた動的な複製数設定を行い、複製配置に伴うデータ量低減、およびデータ消失耐性向上を図ることが出来ることを示した。

Efficient Data Management using the Session Log in DHT and its Evaluation

Satoru Noguchi,[†1] Atsuo Inomata,[†1] Kazutoshi Fujikawa[†1]
and Hideki Sunahara[†1,†2]

Distributed Hash Table (DHT) is a specialized distributed system that aims to lookup hashed identifiers efficiently in order to route messages to and from the corresponding nodes and objects. However DHTs are sensitive mechanism to churn(the continuous and interleaved process of arrival/departure of nodes). As a result, they often cause a losing for stored objects and a failure in the lookup. That is, we think it is a serious issue for all of overlay network. In this paper, we propose a dynamic replica location method using the session log in DHT. Our method can decide an effective number of replicas on session running period among anticipate nodes dynamically. Furthermore we implemented a prototype based on an existing DHT routing algorithm and evaluated on a side for performance. So we showed that our proposed method can be reduced a rate of the cost for the locating replication and the losing data.

1. はじめに

近年、大量のノード／データをスケーラブルに扱える方式として Peer-to-Peer（P2P）ネットワークが注目されている．P2P ネットワークは下位層、つまり実ネットワークのトポロジと異なる独自のトポロジを構成することから、このようなネットワークはオーバレイネットワークと呼ばれる.

オーバレイネットワークにはトポロジの制約の有無によって非構造化／構造化の区別がある．非構造化オーバレイネットワークはノードの隣接関係やデータの配置に制約が無く、探索をフラッディングによって行う．このためネットワーク構成が単純で障害耐性に優れる反面、探索の成功が保証されず、また規模拡張性に劣る．一方、構造化オーバレイネットワークは探索に数学的規則に基づく制約があり、規則的な探索を行うことによって確実に目的のノード，データを探索可能であり規模拡張性にも優れる．このため構造化オーバレイネットワークは大容量、多数のデータを広範囲に配布、格納、共有する基盤として注目されており、分散ストレージやコンテンツ配信アプリケーション用途について研究が盛んに行われている[3),5),8),11),12)]

代表的な構造化オーバレイネットワークとして DHT（Distributed Hash Table：分散ハッシュ表）がある．DHT はハッシュ表を分散保持する技術で、ノード数に対してスケーラブルなホップ数で資源探索を行うことができる．代表的なルーティングアルゴリズムとして Chord[15)],Kademlia[9)],Pastry[13)],Bamboo[10)] 等があげられる.

†1 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science,Nara Institute of Science and Technology
†2 慶應義塾大学大学院 メディアデザイン研究科
Graduate School of Media Design,Keio University

しかしDHTを大規模サービスに適用する場合、ノードの頻繁な参加／離脱（churn）に伴うネットワーク構成の変動が問題となる．P2Pネットワークではノードはサービス利用者であると同時にサービス提供者でもあるため、あるノードの参加／離脱が他のノードのサービス提供に影響を及ぼす．具体的には、経路表上の情報と実際のノードの状態との乖離によるノード探索の失敗に伴う応答遅延、データ保持ノードの離脱に伴うデータ消失といった問題が起きる．このうち、ノード探索失敗については再探索を繰り返すことにより、遅延は増すものの求めるノードを発見することは出来る．一方でデータ消失が発生した場合、オーバレイネットワーク側でデータを復旧する手段はない．このためデータの消失を防止する策を講じる必要がある．

その対策として定期的にデータを再格納する手法、および一定量のデータの複製を複数ノードに配置する手法が一般的である．しかし既存の方式ではこれらの手法における設定（再格納の間隔／複製の数）が静的であり、各ノードのchurn頻度の違い、およびchurn頻度の変動に対応出来ない．churnの原因はユーザの操作によることが大きいため、その頻度は常に一様ではなくユーザによって異なっている．このような環境では画一的な設定を静的に行うことは適切ではない．

そこで本論文ではデータ消失の防止およびデータ管理コストの低減のため、ノードの動作履歴を元に個々のノードのchurn頻度を把握し、churn頻度の変化に動的に対応可能なデータ管理手法を提案する．以下、２節では現状のデータ管理手法の概要と問題点を述べ、３節では提案する複製管理手法について述べる。４節では評価について述べ、続く５節では関連研究について述べ、最終章で本論文をまとめる．

2. DHTにおけるデータ管理と問題点

本節ではDHTの概要、データ管理手法とデータ消失対策の必要性、およびその問題点について述べる．

2.1 DHT

DHTはデータの識別名をSHA-1等のハッシュ関数を用いてハッシュ化したID（key）とデータの実体（value）のペアをオーバレイネットワークに参加するノード間で分散保持するもので、Map型のデータ入出力インタフェースを提供する．アプリケーションはこのインタフェースを通してDHTを使用することでP2Pネットワークによるサービスを実現する[4]．DHTが提供する代表的な入出力機能を以下に示す．

- PUT(key,value)：key,valueペアをDHTに格納する
- GET(key)：keyに対応するvalueをDHTから取り出す

ノードについてもデータ同様に識別名をハッシュ化することで、データとノードを同一のID空間にマッピングする．ここで、データはそのIDに最も近いIDを有するノード（rootノード）が担当する．すなわち、ID空間上の領域をノード間で分割管理することでデータの分散保持を実現する．データの配置はID空間上の近接性に基づいて決定されるため、データの原本を保持するノード（DHTにPUTリクエストを行うノード）とPUTされたデータを格納するノードには関連がない．（IDが隣接していれば同一になることもある）．

2.2 データ消失対策の必要性と問題点

P2Pネットワークにおいて一般的にノードのネットワークへの参加/離脱には制限がないため、ネットワーク構成の変動（churn）が日常的に発生する．その際、ノードはサービス利用者であると同時にサービス提供者でもあるため、離脱したノードが担当するデータはネットワークから消失してしまう．そのため、(1) 複数ノードにデータの複製を配置しデータの消失を防止する、(2) 定期的にデータを再度格納する、という対処が必要になる．しかしながら現状、どちらの対処にも問題がある．以下、(1)(2) の概要と問題点を述べる．

2.2.1 複　　製

データ格納時、key,valueペアを本来担当すべきノード（rootノード）の他に複数のノードに配置することでデータ消失耐性を高めることができる[4),17)]．複製数は事前にシステムの動作環境（churn頻度）に適した値に設定しておく．しかし、この方式には下記に示す問題があげられる．

- 最適な複製数の設定が困難：サービス開始前にchurn頻度を予測し最適な複製数を設定することが困難である．churnの主な原因はユーザの操作であるため、事前に調査、計測を行うことが出来ない．
- 動作環境の変化に対応出来ない：P2Pネットワークではサービス提供者は個々のノードであるため、サービス開始後に設定調整を行うためには全ノードの設定を一括して変更する必要がある．しかし、広範囲に配布したアプリケーションの設定を任意のタイミングで全て変更するのは困難である．このため動作環境の変化に柔軟に対応出来ない．
- 最適な設定値がノードごとに異なる：ネットワークへの参加／離脱タイミングはノードによって異なる[6),14),16)]、すなわちchurn頻度はノードごとに

異なるため、全てのノードに対して画一的な複製数設定を行う方式では通信相手のノードの churn 頻度に応じた適切な設定が行えない．

これらの要因によって不適切な複製数設定が行われた場合、複製数の不足に伴うデータ消失、過剰な複製による通信量増大/ストレージ浪費の問題が起きる．

2.2.2 再 格 納

データを保持するノードが定期的にデータの再格納を行うことで、データ消失の防止を図ることが出来ると考えられる．現在、再格納を (1) アプリケーション (データの原本を保有するノード)、または (2) DHT(データを格納するノード) が行う手法が提案されている．前者は PUT 時にデータの保管期限（TTL）を設定し期限内に再度データの格納を行うもので、再格納の動作は全てアプリケーションの判断で行われる．後者は DHT において PUT されたデータを保持するノード、つまり root ノードが一定間隔でデータの再格納を行うもので、アプリケーションが再格納の処理を意識する必要がない．しかし、一旦データを格納した後、再格納までの期間にデータが消失した場合、前者の方式では再格納が行われるまではデータ取得要求に対応出来ず、後者の方式では消失したデータの復旧自体が行えない．その対処としてデータの消失（ノードの離脱）を上回る頻度で再格納を行う方法があるが、妥当な頻度の決定が困難であり、また複製数設定と同様に適切でない頻度を設定した場合にデータ消失や過剰な再格納処理に伴う通信量の増大を招く．

これらの理由により、複製、再格納のいずれの場合においても、対象ノードの churn 頻度に応じた設定を行えないことが問題である．

3. 提 案 手 法

前節で述べたデータ消失、複製配置／再格納におけるデータ維持コスト増大問題の解決策として、本節では、各ノードの churn 頻度に応じて複製数を動的に調整することによって最適な複製数でデータの消失を低減した上で、消失するデータに応じた最小限の再格納周期を検知可能とする手法を提案する．

提案手法は 1)churn 頻度の把握、2) 複製配置、3) データの再格納の 3 ステップから構成される．詳細を以下に述べる．

3.1 churn 頻度の把握

はじめに、各ノードが任意の他ノードの churn 頻度を把握可能とする．ここで　churn 頻度の把握対象となるノードはデータの ID に最も近い ID を持つノード（root ノード)、すなわちデータの格納を担当するノードである．churn 状況の確認方法には以下の２つがある．

- 能動的手法:自ノードが対象ノードに対して PING メッセージ等の生存確認要求を定期的に送信することで、ノードの参加/離脱状況を観測する．
- 受動的手法：各ノードが自身の参加／離脱状況を観測し、通信相手のノードに通知する．

churn 頻度の把握を行うためには任意の１時点における生存確認だけではなく、一定の期間内で定期的に生存状況を確認する必要がある．これを前提とすると、前者の能動的手法では生存確認メッセージの送信を一定期間継続して行う必要があるが、対象ノードの特定は実際に PUT 要求を行うまで行えない．そのため、データ格納の際に迅速に対象ノードの churn 頻度を把握することは困難である．一方、受動的手法では各ノードが自身の churn 頻度を観測、保持するため、必要に応じて要求元に即座に churn 頻度を通知することが出来る上、新たな計測用メッセージを送信する必要がない．そこで、本提案では churn 状況の把握に後者を適用する．

3.1.1 セッション情報の収集

churn 頻度の把握は一定の計測期間内における自ノードの参加／離脱履歴（セッション情報）を収集することで実現する．セッションとは、ノードがオーバレイネットワークに参加してから離脱するまでの期間を指す．セッション情報の収集方法を以下に順に述べる．

(1) 事前に計測周期を設定する．

(2) 初回のサービス起動時、現在時刻を周期開始時刻として記録し、計測を開始する．

(3) セッション開始時すなわちオーバレイネットワークへの参加時、現在時刻および計測周期内における相対時間をセッション開始時刻として記録する．

(4) 一定期間ごとに自ノードの参加／離脱状態を確認し、セッションが継続している場合は現在時刻を当該セッションの最終確認時間として記録または更新する．確認の間隔は事前に設定する．

(5) オーバレイネットワークから離脱後、次回参加する際に前回のセッション持続時間を確定し、セッション情報を生成する．１つのセッション情報はセッション開始時間（計測周期内における相対時間)、セッション持続時間から構成される．以降、計測周期終了まで同様の処理を行う．

(6) 計測周期が終了したとき、前回周期までに累積したセッション情報と今回周期に収集したセッション情報を統合する．その後、計測周期開始

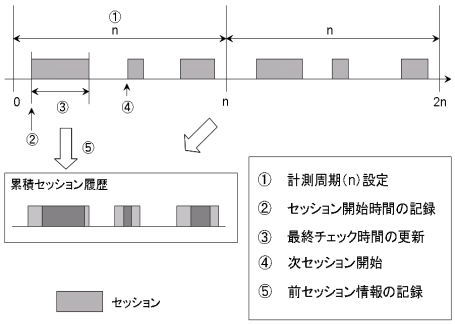

図 1 セッション情報計測の概要
Fig. 1 Overview of measuring the node's session information.

時刻を現在時刻に更新し、次回周期の計測を開始する．

一定の計測周期を設定する根拠は churn の原因がユーザの操作にあり、各ユーザごとに参加／離脱タイミングに偏りが存在すると考えられるためである．

実際に 6)，14)，16) は、P2P ファイル共有ネットワークにおいてノードのセッション時間は均一ではなく、少数のノードが長期間生存し、大半のノードが短期間で離脱する、つまりノードによってセッション時間に差異があるという観測結果を示している．このため、ユーザの動作周期に合致した計測周期を設定することで計測周期内の任意のタイミングにおけるセッション持続時間を予想することが出来ると考えられる．計測周期の設定はユーザの動作周期を考慮し、1 日間や 7 日間とすることが考えられる．

また、セッション情報の統合は新たに計測したセッション情報と前周期の同時間帯のセッション情報との平均を算出することによって行う．このため、周期的に動作するノードにおいて計測周期を反復することによってセッション情報の精度を高めることが出来る．計測周期とセッション情報計測の概要を図 1 に示す．

3.1.2 セッション情報の提供

上記の手順により収集した自ノードのセッション情報を使用し、以降の周期におけるセッション持続時間を予想し、相手ノードに通知する．処理方法を以下に述べる．

(1) オーバレイネットワークへの参加時、現在時刻が計測周期のどのタイミングに該当するか確認する．
(2) 前節の手順で収集したセッション情報を参照し、現在時刻に合致するタイミングにおけるセッション持続時間を取得する．
(3) 現在時刻および得られたセッション持続時間を、現セッションの開始時刻および予想セッション持続時間として保持する．
(4) ルーティング要求受信時、自ノードの現セッション情報を応答メッセージに付加する．
(5) 応答メッセージを受信したノードは、経路表上の該当ノードのエントリにセッション情報を追加／更新する．

これらの処理によって、任意のノードの離脱タイミングを予想することが可能となる．

3.2 複製の配置と維持

データ格納時、root ノードから順に ID が近接したノード順にデータの複製を配置していく．このとき、予想セッション持続時間が一定の閾値（閾値 1）以上であるノードを発見するまで複製数を増加させる．また、予想セッション持続時間が前記と別の一定の閾値（閾値 2）以下であるノードには複製配置を行わない．

なお、root ノードに近接するノードの予想セッション持続時間が一様に短い、または非常に高い閾値を設定した場合に複製数が無制限に増大する恐れがあるため、配置可能な複製数、および探索する root 近接ノードの範囲に上限を設ける．従来の複製配置手法では root ノードから ID が近接しているノード順に複製を上限値まで、常に一定数配置するが、提案手法では閾値 1、閾値 2 により、予想セッション持続時間が長いノードのみに選択的に複製を配置する事が出来る．このため、root ノード付近に予想セッション時間が長いノードが存在する場合、従来手法と比較して少ない複製数でデータ消失を回避することが可能となると考えられる．

また、仮に複製数の上限に達しても要求する予想セッション持続時間を満たすノードを発見できないデータがあっても、配置したデータの予想生存時間、すなわちデータ保持ノードの予想セッション持続時間を元に、それらのノードが離脱する前に再格納を行うことでデータ消失耐性を向上させることができる．

3.3 実装上の優位性

予想セッション持続時間を取得することにより、個々のノードの churn 頻度に応じた複製数調整を行うことが可能となる．また、データの格納先となったノードの予想セッション持続時間を確認することでデータ再格納の周期を決定する基準となる情報提供を行うことが出来る．これにより、複製の不足／データ再格納の遅延に伴うデータ消失の防止、複製の過剰／データ再格納の過剰に伴う通信量増大を防止できる．また、セッション時間の予測は各ノードが単独で行い、予測

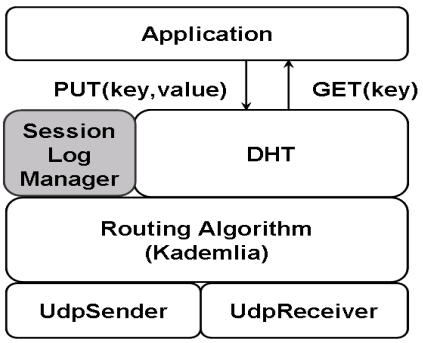

図 2 システム構成
Fig. 2 Structure of system arcitecture.

したセッション情報は既存のメッセージに付加することから、新たなメッセージを発生させることなく処理を行うことが可能である．更に、いずれの処理も DHT における ID 距離を基準とした経路選択基準を大きく変更するものではないため、既存のルーティングアルゴリズムに追加する形で実装することができる．

4. 評　　価

本節では、提案手法の有効性を評価するために実装したシステムについて概観し、評価実験について述べた後、実験結果を考察する．

4.1 実 装 環 境

機能の拡張性、再利用性を考慮し、Java SE6 を用いて提案手法の実装を行った．DHT のルーティングアルゴリズムとして Kademlia をベースとし、トランスポートプロトコルに UDP を使用した．また Dabek らの KBR[4] に従って機能を抽象化し、セッション情報の収集機能は DHT、ルーティングアルゴリズムとは別のサービスとして KBR における高レベル層に追加した．セッション情報に基づいたデータ配置機能については DHT の PUT 機能を拡張する形で実装した．システム構成を図 2 に示す． また、計測した評価指標を以下に示す．

- データ取得（GET）成功率：要求する key を取得出来たときデータ取得成功とする．
- 通信量：各ノードが送出したメッセージの総数及び総バイト数を計測する．メッセージの種別は区別せず、全てのクエリで送信されるメッセージを計測対象とする．
- 複製数：データ格納時に作成する複製数の平均値を計測する．

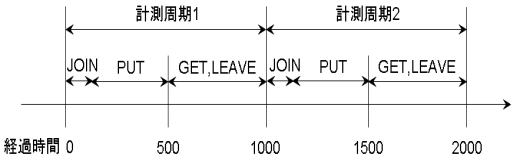

図 3 計測の流れ
Fig. 3 Flow of the experiment.

4.2 実 験 環 境

計算機 1 台に 1000 ノードを起動するエミュレーション環境で前項の性能指標を計測した．実験期間を 2000 秒、セッション履歴計測周期を 1000 秒とし、同一の動作を 2 周期分実施することでセッション履歴に基づく動的な複製配置の効果を確認した．すなわち、1 周期目はセッション履歴が存在しないことから、予想セッション持続時間に基づく複製配置を行わない．これに対して 2 周期目は 1 周期目のセッション履歴を元に複製配置を行うことが可能である．システム動作条件を**表 1** に示す．

また、計測の流れを図 3 に示し、以下順に述べる．

(1) 全てのノードを起動する．
(2) 0.01 秒間隔で全てのノードをオーバレイネットワークに参加させる．
(3) 1 秒間隔で 500 組の key,value ペアを各 1 回 PUT する．対象ノードはランダムに選択する．
(4) 1 秒間隔で上記の key,value ペアを各 1 回 GET する．対象ノードはランダムに選択する．また、この期間中に任意のタイミングで全てのノードを順次離脱させる．
(5) 全ての GET が完了し、かつ全てのノードが離脱した後、再度全てのノードを参加させる．
(6) 以後、上記と同様に新たな 500 組の key,value ペアの PUT,GET を行う．離脱も同様に行う．

ノードのセッション持続時間の長さに応じた複製配置を行う手法の効果を示すため、各ノードのセッション持続時間に区分を設けた．最長のものは 1 回の計測周期中、常にセッションを継続し、最短のものは PUT 期間終了後、すみやかに離脱する．実験条件を単純化するためセッション時間の区分は 4 区分とし、それぞれの区分におけるノード数は、セッション持続時間が長いノードほどノード数が少なくなるよう設定した．複数のファイル共有ネットワークにおける計測結果において、基本的なセッション時間長およびノード数は異なるものの、それぞれの分布については上記同様の

傾向を有することが示されている[6),14),16)]．また、各ノードが周期的に動作すると仮定することからノードのセッション時間は 1 周期目、2 周期目とも同様とする．ノードのセッション時間とその割合を**表 2** に示す．

4.3 実験結果

実験結果を**表 3** に示す．実験は同一環境で 3 回実施し、その平均値を評価に用いた．表中の前半／後半とは計測周期 1 周期目 (0-1000 秒) および 2 周期目 (1001-2000 秒) の期間に該当することを示しており、複製数とは PUT 時に作成した複製の平均値を示している．データ数、データサイズは各ノードが送信したメッセージの総数及び総バイト数を示す．GET 成功率は GET の総数に対して要求する key を取得した割合を示す．

4.4 考　　察

複製数に関して、前半 5 に対して後半 4 となっており対象ノードの予想セッション持続時間に応じて複製数が動的に調整されていることがわかる．しかしながら通信量に着目すると、前半に対して後半の方が減少しているものの、その差は総数と比較して小さな割合に留まっている．この原因はノード探索時のルーティングアルゴリズムの処理を (1)root ノード及び root ノードに ID が隣接する複数ノードを探索、(2) 探索した root ノード群から各処理において必要な任意の数のノードにメッセージ送信、としていることから、PUT 時に配置する複製数に無関係に (1) において一定数のノードと通信するためである．すなわち、複製数調整の影響を受けるのは (2) の処理であるが、(1) と比較して通信対象ノード数が少ないことから通信量の低減が軽微なものとなる．ただし、ノード探索及び GET(要求時) と異なり、PUT 時には格納するデータの実体を送信する．このため扱うデータのサイズが増大すれば複製数低減の効果がより大きくなる．

GET 成功率については前半の 55%に対して後半では 70%が成功しており、複製数調整の効果が現れている．しかしながら、効果は上がっているものの成功率としてはまだ高いとは言えない．GET 成功率はデータ保持ノードにおける PUT 後のセッション持続時間（データの生存期間）、および PUT 後から GET 発生までの時間に依存する．つまり、前者の時間が最長かつ後者の時間が最短となるとき、最も GET 成功率が高くなる．しかし実験では GET 対象の key がランダムに選択されることから、後者の時間を変化させることは出来ない．GET 成功率を向上させるにはデータの生存期間を延長させることが最適である．

既存手法では複製配置ノードは ID 距離のみに基づいて決定されるので、データの生存期間に影響するパラメータは各ノードのセッション持続時間の分布および複製数であり、そのうち調整可能なパラメータは複製数のみである．一方、提案手法では各ノードのセッション持続時間を予測可能であることから、生存期間に影響するパラメータは上記に加えて複製配置対象および対象外となるノードの比率、およびそれらのノードを決定する基準すなわち閾値 1、2 となる．複製配置ノードの選択は閾値の設定によって行うため、適切な閾値設定を行うことが重要となるが、仮に最適な設定でない場合でも格納したデータの予想生存時間を知ることが出来ることから、予想生存時間が経過する前にデータを再格納するという対応を行う余地があり、既存の静的な複製配置手法と比較してデータ消失耐性の低下を防止することが可能である．

5. 関連研究

Dabek らは KBR において複製配置用 API について言及しており、複製配置手法の一例として Chord において root ノードの successor ノード近傍に複製配置を行うことを提案している[4)]．

首藤はルーティングアルゴリズムに依存しない churn 対策として root ノード近傍への複製配置、新規参加ノードへのデータ委譲、root ノード近傍の複数ノードに対するデータ要求、データ格納ノードによるデータの自動再格納の手法を提案しており、Overlay Weaver[18)]

表 1　実験条件：システム動作条件
Table 1　System specifications in the experiment.

項目	設定
複製数上限	5
計測周期	1000 秒
計測間隔	10 秒
閾値 1	500 秒
閾値 2	200 秒

表 2　実験条件：ノードのセッション時間（単位：秒）
Table 2　Session length of nodes in the experiment.

ノード数	セッション時間
100	1000
200	750
300	600
400	500

表 3　実験結果
Table 3　Results in the expeiment.

計測期間	複製数	データ数	データサイズ	GET 成功率
前半	5	132648	200131818	55 %
後半	4	130547	198131817	70 %

を用いたエミュレーションによって自動再格納が最もGET 成功率を高めるとの結果を示している[17]．なお、本稿の提案手法においてもノード探索時に root ノードだけでなく root ノード近傍の複数ノードを探索する手法を用いている．

Binzenhofer らは churn 頻度を把握するために隣接ノードに自身の参加／離脱情報を通知することを提案している[1]．この提案は churn 状況の把握について述べたものであり、データの配置に関しては言及していない．

Godfrey らは churn が起きる環境におけるノード選択について、選択時点での起動時間が長いノードを優先的に選択することで churn 耐性を高められるとのシミュレーション結果を示している[7]．この手法は簡便ではあるが、セッション時間が均一でないノードを有効に利用することが出来ないと考えられる．

Chun らは分散ストレージにおける複製の生成/消滅過程をマルコフモデルに当てはめ、複製数の変化を予測することでノードの一時離脱時に不要な複製を増加させない手法を提案している[2]．彼らはデータの可用性と耐性を別個に考慮すべきであるとしており、ノードの一時離脱を可用性の低下、永続的な離脱を耐性低下と捉え、前者においては複製数を増加させず、後者の場合のみに対処を行う必要があると主張している．この手法は離脱したノードが離脱前の情報を保持したまま再参加することが前提になるため、対象サービスが限定される．

6. おわりに

6.1 まとめ

本論文では DHT におけるデータ管理において問題となるデータ消失、データ維持コストに着目し、ノードの参加／離脱履歴の収集／分析に基づく適応的なデータ管理手法の提案を行った．さらに提案手法を既存のルーティングアルゴリズムである Kademlia に対して実装し、性能評価の結果、参加／離脱履歴から求めたノードの予想セッション持続時間に基づいて複製数を動的に調整し、通信量の増大を抑えつつデータ取得成功率を高めることが出来るという結果を得た．ノードの参加／離脱が周期的に発生する環境において、計測周期を繰り返しセッション情報を蓄積することで高精度のセッション持続時間の予測が可能になると考えられる．

また、提案手法は既存の DHT における経路選択手法を大きく変更するものではないため、他の DHT のルーティングアルゴリズムに対しても適用することが可能である．しかしながら、提案手法ではノード探索に伴うクエリは低減させることが出来ないため、扱うデータのサイズが小さい場合には通信量低減の効果が小さいことも判明した．

6.2 今後の課題

提案手法では周期性なく参加／離脱を行うノードや、設定した計測周期に合致しない周期で動作するノードについては予想セッション持続時間の信頼性が低くなり、適切な複製配置を行えない恐れがある．今後の課題として、予想セッション持続時間の正当性を検証し、計測手法をノードの周期に合わせて動的に変更する機構が必要である．また、本論文ではセッション時間予測の効果を確認するために実験条件を単純化し、周期的に動作するノードのみを用いて性能評価を行ったが、今後は定量的な分析を通して提案手法の評価を実施し、かつ周期性を持たないノードが含まれる等、より実用に近い環境においても評価を行い、提案手法の更なる性能改善に取り組んで行く必要がある．

参考文献

1) Binzenhöfer, A. and Leibnitz, K.: Estimating Churn in Structured P2P Networks, *20th International Teletraffic Congress (ITC20)*, Ottawa, Canada (2007).
2) Chun, B.-G., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, M. F., Kubiatowicz, J. and Morris, R.: Efficient replica maintenance for distributed storage systems, *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design and Implementation*, USENIX Association, pp. 4–4 (2006).
3) Dabek, F., Kaashoek, M.F., Karger, D., Morris, R. and Stoica, I.: Wide-area cooperative storage with CFS, *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, ACM, pp.202–215 (2001).
4) Dabek, F., Zhao, B.Y., Druschel, P., Kubiatowicz, J. and Stoica, I.: Towards a Common API for Structured Peer-to-Peer Overlays, *In Proceedings of IPTPS2003*, pp.33–44 (2003).
5) DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W.: Dynamo: amazon's highly available key-value store, *SIGOPS Oper. Syst. Rev.*, Vol.41, No.6, pp.205–220 (2007).
6) Falkner, J., Piatek, M., John, J.P., Krishnamurthy, A. and Anderson, T.: Profiling a million user dht, *IMC '07: Proceedings of the 7th*

ACM SIGCOMM conference on Internet measurement, ACM, pp.129–134 (2007).
7) Godfrey, P. B., Shenker, S. and Stoica, I.: Minimizing churn in distributed systems, *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM, pp.147–158 (2006).
8) Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C. and Zhao, B.: OceanStore: an architecture for global-scale persistent storage, *In Proceedings of ASPLOS-IX*, ACM, pp.190–201 (2000).
9) Maymounkov, P. and Mazières, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, *In Proceedings of IPTPS2002*, pp.53–65 (2002).
10) Rhea, S., Geels, D., Roscoe, T. and Kubiatowicz, J.: Handling churn in a DHT, *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, USENIX Association, pp.10–10 (2004).
11) Rhea, S., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I. and Yu, H.: OpenDHT: a public DHT service and its uses, *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM, pp.73–84 (2005).
12) Rowstron, A. and Druschel, P.: Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility, *SIGOPS Oper. Syst. Rev.*, Vol. 35, No. 5, pp. 188–201 (2001).
13) Rowstron, A. I. T. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Springer-Verlag, pp.329–350 (2001).
14) Steiner, M., En-Najjary, T. and Biersack, E. W.: A global view of kad, *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ACM, pp.117–122 (2007).
15) Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications, *SIGCOMM Comput. Commun. Rev.*, Vol.31, No.4, pp.149–160 (2001).
16) Stutzbach, D. and Rejaie, R.: Understanding churn in peer-to-peer networks, *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ACM, pp.189–202 (2006).
17) 首藤一幸：下位アルゴリズム中立な DHT 実装への耐 churn 手法の実装，情報処理学会論文誌. コンピューティングシステム， Vol.49, No.2, pp. 1–9 (2008).
18) 首藤一幸，田中良夫，関口智嗣：オーバレイ構築ツールキット Overlay Weaver，情報処理学会論文誌. コンピューティングシステム， Vol.47, No.12, pp.358–367 (2006).