CANに基づくP2Pネットワークのルーティングテーブル更新方法について

CAN Based Updating Method of Routing Table in P2P Network

木下 俊吾 † 川村 尚生 †† 菅原 一孔 ††

Shungo Kinoshita† Takao Kawamura †† Kazunori Sugahara ††

† 鳥取大学 大学院 工学研究科 知能情報工学専攻 †† 鳥取大学 工学部 知能情報工学科

1 はじめに

ピュアP2Pネットワークにおいて分散しているコンテンツを効率よく確実に検索するために，ネットワークを構造化する手法として分散ハッシュテーブル（以下DHT）がある．CAN[1]もその一つであるが，ルーティングテーブルの整合性を保つ具体的な方法が提案されていないため，CANを用いた実用的なシステムは存在していない．

そこで本稿ではCANのルーティングテーブル更新方法として2つのアルゴリズムを提案・実装する．

2 CANによるP2Pネットワークの構築

CANは仮想座標空間を持ち，この空間に$(key, value)$のペアをハッシュ関数を用いてマッピングする．keyはコンテンツに対応しており，$value$には一般的にコンテンツの位置情報（IP）が格納されている．CANに参加するコンピュータ（以下ノード）は座標空間の一部の領域を割り当てられ，CANから離脱するノードは領域を他のノードへ引き渡す．この時，各ノードが自身の管理する領域にマッピングされた全ての$(key, value)$を管理することでコンテンツの位置情報を分散管理している．ここでCANの持つ仮想座標空間はN次元に対応しているが，今後はモデルを簡単化して説明を行うため$[0,0] \times [1,1]$の2次元座標空間に限定する．

CANが構造化されたP2Pネットワークを構築するために，各ノードは以下の情報を管理する．

- ノード情報（自身のIPと管理領域のセット）
- ルーティングテーブル（隣接ノードのノード情報）

ここで隣接ノードとは自身の管理領域と領域切片が接しているノードを指す．コンテンツの検索はこのルーティングテーブルを用いて行う．図1はA〜Eの5つのノードがCANに参加した際の各ノードが管理する領域の状態とルーティングテーブルを示している．このとき図中の各ノードのノード情報は表1に示される通りである．ノードDを例に挙げるとルーティングテーブルとして隣接ノードB，C，Dのノード情報を管理している．

CANでは各ノードが管理するルーティングテーブルの整合性を保つために，各ノードが自身のノード情報とルーティングテーブルを含む更新要求メッセージを隣接ノードへブロードキャストする．ブロードキャストは自身の領域が変化した場合，もしくは一定周期で行われる．そして更新要求を受信したノードは受信した情報をもとにルーティングテーブルの更新を行う．ノードの参加・離脱により隣接ノードが常に変化している状態において，常に全てノードのルーティングテーブルの整合性を保つことはできないが，一定周期で送信される更新要求により，各ノードのルーティングテーブルの整合性を高めることができる．

各ノードのルーティングテーブルの整合性を高めるためには，各ノードが隣接ノードの最新のノード情報をルーティングテーブルに登録する必要がある．しかし，[1]ではルーティングテーブル整合性を保つための具体的な更新アルゴリズムについて述べられていない．そこで次章ではCANにおける具体的なルーティングテーブル更新アルゴリズムを設計する．

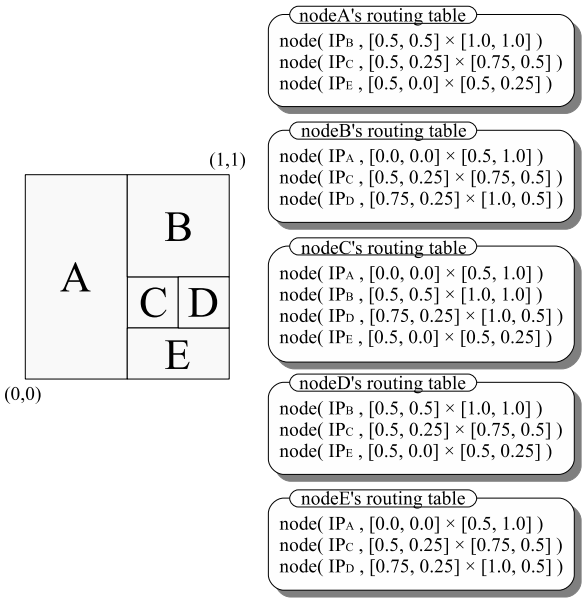

図 1: CANにおけるP2Pネットワーク構築の様子

3 ルーティングテーブル更新アルゴリズム

この章ではCANを実装する上で生じる問題点を考察し設計を行う．また更新要求メッセージに含む情報として，[1]で提案されている「ノード情報」と「ルーティングテーブル」を用いる方式を方式1,「ノード情報」のみを用いる方式を方式2として設計する．

表 1: 図 1 における各ノードのノード情報

ノード名	ノード情報
A	$node(IP_A, [0.0, 0.0] \times [0.5, 1.0])$
B	$node(IP_B, [0.5, 0.5] \times [1.0, 1.0])$
C	$node(IP_C, [0.5, 0.25] \times [0.75, 0.5])$
D	$node(IP_D, [0.75, 0.25] \times [1.0, 0.5])$
E	$node(IP_E, [0.5, 0.0] \times [0.5, 0.25])$

3.1 方式 1

3.1.1 ノード情報の拡張：タイムスタンプの付与

前章で述べた通り [1] にはルーティングテーブルを最新に保つ方法は示されていない．例えば，1 つのノードが複数のノードから更新要求メッセージを受信し，メッセージに “IP は同じだが領域情報が異なる” ノード情報がそれぞれ含まれていた場合，更新要求を受信したノードはどのノード情報を登録すべきか分からない．これは受信した順番が最後だからといって受信したメッセージに含まれるノード情報が最新とは限らないからである．

この問題を解決するために，各ノードのノード情報を拡張子し，領域情報に生成された時間をタイムスタンプとして付与する．よってタイムスタンプを基準に最新のノード情報をルーティングテーブルに登録可能となる．

3.1.2 更新要求メッセージの転送

[1] ではルーティングテーブルの整合性を保つため，各ノードが管理領域が変化したときもしくは一定周期で更新要求メッセージをブロードキャストすることが述べられている．しかし実際には隣接ノードだけにメッセージをブロードキャストするだけではルーティングテーブルの整合性を保てないことを例で示す．

図 2 はノード A，B が CAN に参加している状態から，ノード C，D が新たに参加した直後のルーティングテーブルの状態を示している．ノード C はノード A から，ノード D はノード B より同時に領域を分け与えられたものとする．この瞬間ノード C，D は隣接ノードでありながら互いに存在を知らない状態となってる．このような状態で隣接ノードに更新要求メッセージをブロードキャストしてもノード C，D はお互いを登録することができない．例えばノード C のノード情報をノード D のルーティングテーブルに登録する場合を考えると，ノード C をルーティングテーブルに登録しているノードのブロードキャストによる間接的な伝達，もしくはノード C のブロードキャストによるノード D への直接伝達，という 2 つの伝達方法が考えられる．しかし前者の場合，ノード A がノード C を登録していても，ノード D がノード A の隣接ノードでないためノード A による間接的な伝達はありえない．またノード B にとってノード C は隣接ノードではないためノード B のルーティングテーブルにノード C が登録されることはない．つまりノード B による間接的な伝達もありえない．後者はノード D のノード情報がノード C のルーティングテーブルに登録されていることが前提だが，ノード D のノード情報もノード C と同様前者による伝達ができないので，やはりありえない．よってルーティングテーブルに不整合が生じてしまう．

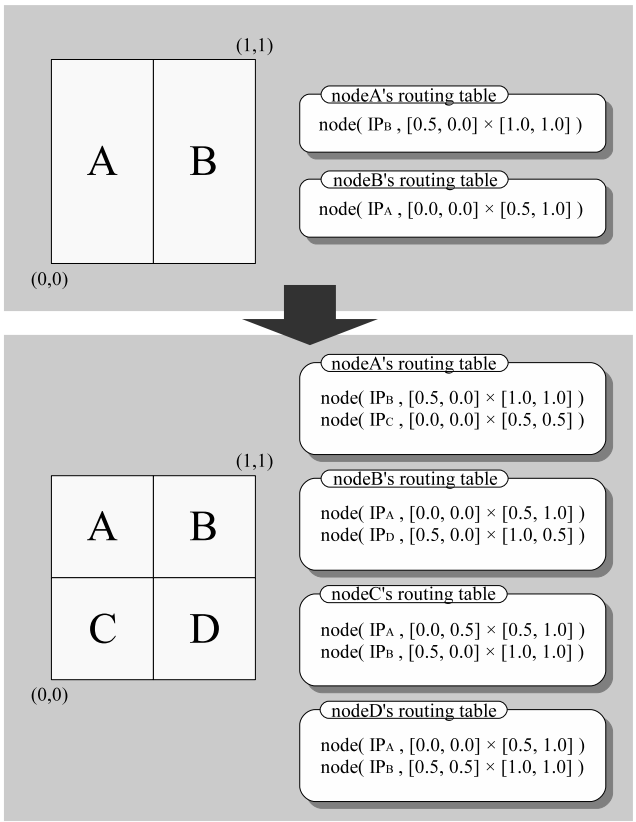

図 2: 複数ノードの同時参加によるルーティングテーブルの不整合

この問題に対し，更新要求を受信したノードがさらにそのメッセージを転送することで解決した．つまりブロードキャストによる更新要求を受信したノードは，その要求に含まれるノード情報をさらにブロードキャストによって隣接ノードに転送する．図 2 を例にして考えると，転送回数 2 回でノード C のノード情報をノード D に登録することができる．例えばルーティングテーブルにノード C を含むノード A のブロードキャストをノード B が受信すると，ノード B はその受信した情報を更にブロードキャストすることによってノード D にノード C のノード情報を伝達することができる．またノード C がノード B をまだ隣接ノードとして認識している状態でブロードキャストすることで，受信したノード B がその情報を更にブロードキャストし

ノード D に情報を伝達することも考えられる．このように転送回数を 2 回とすることでルーティングテーブルの整合性を保つことが可能となる．

3.2 方式 2

続いて更新要求メッセージに「ノード情報」のみを含む方式 2 を設計する．

[1] ではブロードキャストする更新要求メッセージに，送信者の「ノード情報」と「ルーティングテーブル」を含めるとしている．また上でも述べたように，この更新要求メッセージは 2 回転送しなければルーティングテーブルの整合性を保てない．

しかし 2 回転送する必要があるのであれば，更新要求メッセージは送信者の「ノード情報」のみを含んでいればルーティングテーブルの整合性を保てると予想される．先ほど図 2 で示した例のように隣接ノードでありながら互いの存在を知らない場合においても，送信元は「ノード情報」のみを 2 回転送すれば隣接ノードにノード情報を伝達することができるからである．

また方式 1 におけるメッセージの大きは隣接ノード数＋ 1（自身のノード情報）であるのに対し，方式 2 におけるメッセージの大きさは 1（自分のノード情報）となる．平均隣接ノード数が N である場合，方式 2 はメッセージの大きさの平均がが方式 1 の $\frac{1}{(N+1)}$ となる．そのため方式 2 は方式 1 に比べメッセージの大きさが大幅に小さくなるメリットがある．

方式 2 のアルゴリズムは更新要求メッセージに含まれる情報以外は方式 1 と同様である．

4 実装環境

この章では先ほど設計した 2 つの方式の実装環境について述べる．実装には本研究室で開発されたモバイルエージェントフレームワーク Maglog[2] を用いた．Maglog はネットワーク上を自由に移動できるエージェントを Prolog 記法で記述することができる．但しエージェントはエージェントサーバと呼ばれるアプリケーション上でのみ動作する．またエージェント間の通信はエージェントサーバに用意されたフィールドを用いるが，これはインターネットにおける電子掲示板におけるやりとりになぞらえることができる．各エージェントはエージェントサーバに用意されたフィールドを常に監視しており，あるエージェントがフィールドに宛先を記したメッセージを書き込めば，自分宛のメッセージをフィールドから読み取る事ができる．

今回の CAN に基づく P2P ネットワーク構築を行うために，各ノードにノードエージェントと呼ばれるエージェントを用意する．ノードエージェントは

- ノード情報の管理
- ルーティングテーブルの管理
- 参加・離脱要求の受け付け
- コンテンツ要求メッセージのルーティング

などを行う．

P2P ネットワークに参加する各ノードはエージェントサーバを立ち上げノードエージェントを動作させる．ノードエージェント同士がノード間でメッセージ通信を行いネットワークを構築する．

5 実験

5.1 動作確認

動作確認では，ハッシュテーブルの次元数を 2 として，4 つのノードの一斉参加・離脱を行って検証した．検証は更新要求メッセージに含まれる情報が「ノード情報」と「ルーティングテーブル」を含む方式（以下方式 1）と「ノード情報」のみの方式（以下方式 2）に分けて行った．このとき両方式とも領域情報にタイムスタンプを付与するものとする．

転送回数を 2 回として動作確認を行ったが，様々な状況において両方式ともルーティングテーブルの整合性を保つことに成功した．図 2 と同様の状況も再現し整合性が保たれていることが確認できた．

5.2 転送量の比較

両方式における転送量の測定を行った．8 つのノードをシステムに参加させ，全てのノードのルーティングテーブルの状態が正しく更新されてから，1 分間ノード間のパケット転送量を測定した．結果は方式 1 の転送量が方式 2 の約 1.5 倍となった．この時平均隣接ノード数が 3 であったためメッセージの大きさの平均は約 4 倍となる．この結果は転送量の比とメッセージの大きさの比は一致しない事を示している．これは今後より詳しく検証していく必要がある．

6 おわりに

CAN に基づいたルーティング更新アルゴリズムを新たに提案・実装し，新方式において CAN ルーティングテーブルの整合性を保つのに必要な転送回数の検証と，動作確認を行った．実装した両手法ともに転送回数を 2 回とすることでルーティングテーブルの整合性を保つことができたが，今回検証できなかった様々な状況においても整合性を保てるかどうかは保障できない．

今後はネットワークの規模，参加・離脱の頻度などの条件を変更し実験・検証を行うと共に，正しく情報が行き渡るために必要なメッセージの転送回数の同定を行う．またネットワークに及ぼす負荷が方式の違いによってどれほど変化するかなどを検証する．

参考文献

[1] Ratnasamy, S. et al.: A Scalable Content-Addressable Netwrok, *Proc. of the ACM SIGCOMM 2001 Tech. Conf.*, pp. 161–172 (2001).

[2] Motomura, S., Kawamura, T. and Sugahara, K.: Logic-Based Mobile Agent Framework with Concept of Field, *IPSJ Journal*, Vol. 47, No. 4 (2006). will be published.