P2P ネットワークにおける 経路選択アルゴリズムの提案と実装

川口 忠伸[1]、山之上 卓[2]

概要

分散システム上のグループ内のノード間通信を行うための経路選択アルゴリズムを、P2P技術と2分木の性質を用いてJavaで実装し、評価実験を行った。ノード間を完全2分木状に結合することにより、わずかな整数演算と文字列比較だけで、経路表の作成や交換などをすることなく、2分木上の経路選択が可能となる。グループに参加しているノード数をnとすると、ノード間通信の最大遅延時間はO(log n)である。

Suggestion and Inplementation of a Routing Algorithm for a P2P Network

Tadanobu Kawaguchi[1], Takashi Yamanoue[2]

Abstract: An implementation and an evaluation of a routing algorithm for communication between two nodes in a group on a distributed system are shown. This algorithm uses a P2P technology and a property of the binary tree. If the connection of the nodes has the shape of a binary tree, the shortest path between any two nodes is selected using a small amount of integer calculations and string comparisons. The algorithm does not need to construct routing tables or to exchange routing information. The time complexity of the largest latency of the communication between two nodes is O(log n) where n is the number of the nodes.

1 はじめに

近年、たくさんのPCを端末として利用する分散システムが、企業や大学などで導入されている。これらの登場にあわせて、さまざまな分散アプリケーションが登場している。

P2Pマルチキャストを用いた電子黒板システム[1]などでは、分散システム上のノードのグループ内で一斉送信を効率よく行うために、木構成を用いることがある。この方式であれば、ノードの増加に伴う送信者端末の負荷を抑えることができる。

しかし、ノードのグループ内では、全ノードにデータを送信するだけではなく、ある特定のノード間のみで通信を行う場合もある[2]。多くの分散システムでノード間通信を行う場合、IPルーティングやMacアドレス学習スイッチのように、通信を行うノード以外の場所に、経路選択を行うための経路表を用意したり、Ethernetのダムハブ(マルチポートリピータ)のように、個別送信・全送信の区別なくデータを全ノードに転送し、そのデータが不要なノードにおいてデータを破棄したりする方法が取られている。しかしながら、このような物理的なネットワークインフラは、動的に参加ノードが変化する論理的なグループ内のノード間通信には対応していない。また、これらはグループ内でマルチキャストを行うとき、不要なデータをグループ外のノードに配信する場合があり、無駄が多く、特にノード数が増えればその負荷は無視できなくなる。

1 九州工業大学情報工学部, Faculty of Computer Science and System Engineering ,Kyushu Institute of Technology
2 鹿児島大学学術情報基盤センター,Computing and Communications Center,Kagoshima University

そこで、本論文では、分散システム上のグループ内のノード間通信を行うための経路選択アルゴリズムを、P2P技術と2分木の性質を用いてJavaで実装したことと、その評価実験結果について述べる。まず、2章でシステムの構成とアルゴリズムを示し、3章で実装と評価実験について述べ、4章で関連研究との比較を行い、5章でまとめを行う。

2 ノードの結合とアルゴリズム

2－1 ノードの結合

ノードはn個あり、各ノードは1以上n以下の自然数である識別番号(以下IDと述べる)を持つものとする。

また、ノードを図1のような完全2分木状に結合する。この木においては、ノードは図の数字の順番に木に挿入される。 この木において、親ノードのIDを*P*、左の子のIDを*L*、右の子のIDを*R*とすると、以下の関係式が成り立つことが知られている。[3]

```
L = P*2;
R = P*2+1;
```

この木においては、ノードがトラブルなどで欠落した場合を除いては、根から各葉までの高さの差が2以上になることはない。このため、隣同士のノードの通信遅延が一定であれば、もっとも遠くにあるノードへデータを送る場合、O(log(n))の遅延で送ることができる。

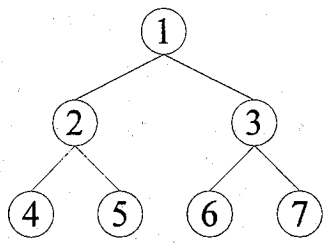


図1:ノードの構成

2－2 完全2分木の作成

新たにノードが参加する場合は、幅優先方式によって、図1の数字の順番にノードを追加していく。これにより、根から各葉までの高さの差は1以内に抑えられる。

特定のノードが離脱する場合は、IDが最大のノード(＝必ず葉である)と離脱するノードを入れ替える操作を行う。これにより木の分割を防ぐ。また、この操作により、移動されたノードのIDは、離脱したノードのものに変更される。

このため、各ノードは、新たに親ノードへつなぎかえる操作を外部から行うことができ、作業の途中でもIDを変えることができる仕様になっているものとする。

2－3 経路選択

各ノードはデータ転送の際に、データのあて先により経路の選択を行う。あて先は1つ以上のノードID、または全員への送信を意味する文字列「broadcast」である。

経路選択のために、ノードIDから「位置文字列」を求める。これはrootノード(IDが1のノード)を「O」とし、その左の子を「L」、右の子を「R」とする。それ以降のIDのノードには親のノードの位置文字列に「L」または「R」を付加したものである。

この位置文字列は、IDを用いて以下の手順で求められる。

(1) IDを2進数に変換し、先頭1桁を除く
(2) 0をL、1をRとする。

この文字列は、rootノードからの経路を意味する。したがって同じ親を持つノードであれば、位置文字列は前方一致する。例えば位置文字列LRRLRを持つノードの親は、LRRLであり、左の子はLRRLRL、右の子はLRRLRRである。

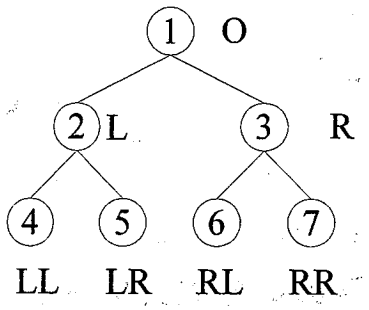


図2:経路選択文字列の振り方

各ノードは、隣接ノードから転送されてきたデータを受け取った場合、あるいは自分のノードからのデータ送信がある場合は、次のアルゴリズムによって、隣接するどのノードにデータを転送すべきかを判断する。

(1) あて先が「broadcast」の場合は、隣接するすべてのノードに転送する。とともに、自身もデータを受け取る。ただし、データの送信側(データを送ってきた側)の隣接ノードへデータを転送してはならない。
(2) あて先のIDが自身のIDであった場合、転送を行わず、自身でデータを受信する。
(3) あて先のIDから位置文字列を求める。
(4) 文字の前方から順に文字を比較し、1文字でも違えば、親ノードへ転送。
(5) 自身のIDで前方一致する場合は、その次の文字が示すノード(Lならば左の子、Rならば右の子)へデータを転送する。

以上の操作を各ノードで行うことで、データは目的のノードへ配信される。ノードは木状に接続されており、データが同じノードを2回以上経由することはないため、送信元のノードから目的のノードへの経路は1つしかなく、かつ、最短となる。

3 実装と評価

今回、分散システム上のグループ内でノード間通信を行うシステムを、2章で述べたアルゴリズムを使い、Javaで実装した。ここで隣接したノード間はTCP/IPを用いて通信が行われる。

このシステムは、実際にデータの送受信を行うノードと、ノードの構成を管理するグループマネージャで構成されている。グループマネージャは、ノードの新規参加、および離脱に伴う処理を自動的に行うサーバであり、ノードの通信中、ノードの新規参加や離脱がない場合は、グループマネージャは一切通信を行わない。

ノードが新規にグループに参加する場合は、グループマネージャに問い合わせを行い、接続すべき親ノードのIPアドレスとポート番号、ならびに、自身に割り当てられるIDを得る。IDはグループマネージャによって管理される。離脱の際は、グループマネージャからの指示により、最大のIDを持つノードと離脱するノードとの入れ替え操作を行い、その後離脱する。

経路選択に使用するためにあて先IDや送信元IDをデータに付加することが必要であるが、本システムでは、データを一定の単位に区切り、そのデータの先頭にテキスト形式のヘッダをつけて送ることにした。

本システムを評価するため、15台の計算機をノードとして用いて以下のような実験を行った。この計算機には(株)ミントウェーブのVID端末システムを利用した[4]。OSとしてMicrosoft (R) Windows2000 Professional、Javaの実行環境として、SunMicroSystemsのJ2SDK1.4を用いた。各端末は100BASE-TXのEthernetで、1台のSwitchに接続した。

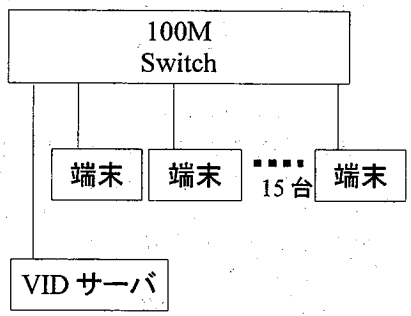


図3:実験に用いた端末の構成

この実験は、以下のように、グループ内のノード間で100バイトの文字列を往復させるものである。これを指定した回数繰り返し、その経過時間を計測する。反復回数は最大１０００回行った。

ここで求めた経過時間は、それぞれ３回ずつの測定を行い、その平均値をとった。

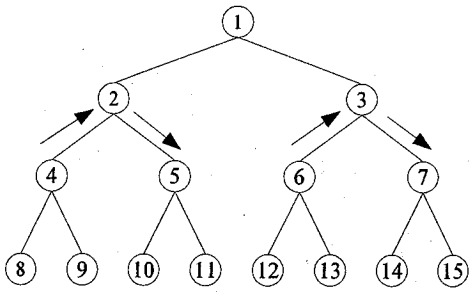
ノードは図４のように結合した。この図において、丸の中の数字がノードの番号(ID)を示す。

(1) 全体のノード数が1～3までの3個の場合、および1～15までの15個の場合において、同じノード間(ノード1とノード2の間)でデータを往復させ、経過時間を比較した。また、同じことを回数を変えて行った。(表1)

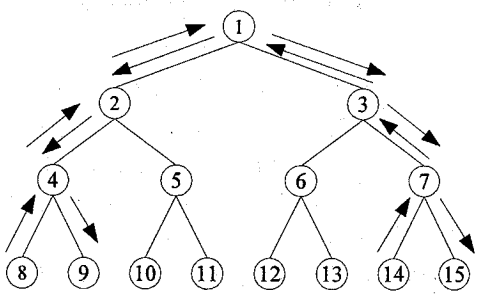
(2) ネットワーク上にある、ホップ数が2であるそれぞれ異なる2ノードで、経過時間と木における位置の関係を調べた。(表2)

(3) アトランダムに2ノードを選択して通信を行い、ホップ数による経路上の経過時間の違いを調べた。(表3)

(4) 経路選択が正しく動作し、他の場所の通信に影響を及ぼしていないかどうかを調べるために、経路が重複しない、ネットワーク上の2つのノードの対の間で同時にデータを往復させ互いに影響を及ぼさずに通信できるかどうかを調べた。 また、経路が重複する2経路において同時にデータを往復させた場合と、個別に送信テストを行った場合を比較し、互いの通信にどのような影響を及ぼすのかを調べた。(表4)



経路の重複なし(4-5,6-7)



経路の重複あり(8-15,14-9)

図4:実験に用いた経路の例

表2:木の場所の違いと経過時間(単位: Sec)

区間	経過時間(1000往復)
2→3	19.4
6→7	32.3
14→15	31.6

表3:距離の違いと経過時間(単位: Sec)

場所	経過時間(1000往復)
2→5	33.9
4→5	34.9
4→11	32.6
8→15	31.4

表1:ノード数と応答時間、
および送信回数と応答時間の関係(単位: Sec)

ノード1-2間の往復回数(回	10	50	100	500	1000	2000
ノード数が3の場合	0.234	1.370	2.660	14.600	28.800	58.600
1回あたりの応答時間	0.023	0.027	0.027	0.029	0.029	0.029
ノード数15の場合	0.230	1.420	2.840	14.500	30.600	59.900
1回あたりの応答時間	0.023	0.028	0.028	0.029	0.031	0.030

表4:経路の交錯の有無と1000往復の経過時間(単位: Sec

経路の交錯	経路	個別 平均	同時 平均
交錯なし	4→5	32.26	32.75
	6→7	31.99	32.82
交錯なし	8→11	33.35	35.80
	12→15	31.97	32.05
交錯あり	8→15	31.83	93.94
	14→9	31.52	75.20
交錯あり	14→9	30.06	63.95
	5→14	31.79	56.42

以上の各表より、次のことが考えられる。

(1) 表1の実験において、データの往復回数と経過時間について、ネットワーク全体のノード数、および送信テストの反復回数に関わらず、ノード間でデータを往復させたときの平均経過時間は同じである。

(2) 表2の実験は、本来、はネットワーク内の各所にて、設計上速度の差がないことを示す実験であったが、root ノードを経由する経路では、若干高速であることが示された。これは、経路選択において、root ノードでは一部の処理が省略できるため、その結果がここに出たものではないか、と考えられる。

(3) 表3の実験においてはノード数間の距離に関わらず経過時間がほぼ同じであることが示された。これは、送信元、および返信元のノードにおける、画面表示などの処理に比べて、中途のデータ転送が十分に高速であることを示すものと考える。

(4) 表4の実験によって、理論上、経路が交錯する場合には遅延が発生し、交錯しない場合には、遅延が発生しないことを示した。この実験により、経路選択においてこのアルゴリズムが十分に機能しており、実際に通信を行っているノード以外のノードへのデータの漏れがないことを示したと考える。

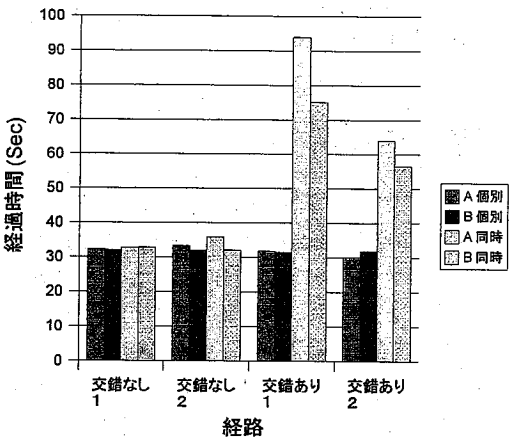


図5:経路の交錯と経過時間

※交錯なし1:経路 A 4→5、B 6→7
交錯なし2:経路 A 8→11、B 12→15
交錯あり1:経路 A 8→15、B 14→9
交錯あり2:経路 A 14→9、B 5→14
繰り返し回数は1000回。

4 関連研究

P2P-Cache[5]では、個別ノード間通信において、毎回セッションを張る方式を実装している。この方式は、比較的簡単に実装できるが、ノード間通信の間隔が一定ではなく、通信の相手がたくさんのノードにあたる場合は非常にオーバーヘッドが大きい。また、ノードを複数指定してデータを発信する場合も、セッションの開始が大変面倒になる。

RelayCast[6]では、本実装と同様のApplication Layer Multicastを、ミドルウェアとして実装している。特にネットワークの構成と、マルチキャストのための経路木を分離し、論理ネットワーク上にマルチキャストツリーを構成している。また、実装方式として、ローカルプロクシ方式を用いているので、API で実現する方式より、既存のアプリケーションとの親和性が高いが、グループ内のノード間通信については配慮されていない。

本研究のような木の構成方法では障害時の復旧が課題となる。これを解決する方法の一つと

して、三浦[7]らによる二重配送路トポロジ制御方式などがある。

5 おわりに

本論文では、2分木構成を持つネットワークにおいて、経路表の作成や交換が不要な経路選択アルゴリズムについて述べた。この方式であればわずかな整数演算と文字列の比較だけで、自律的に経路を選択することができる。

また今回実装した分散システムをは、先に挙げた電子黒板システムのほか、PC端末やディスクレス端末などを一元管理するシステム、街頭のプラズマモニタなどに静止画情報を配信するシステムなどに応用可能である。

現在本システムのノード間で交換されるデータはテキストのみである。バイナリ形式のデータに関しては、BASE64 などのアルゴリズムを用いてテキストに変換してから送信する機構を実装する予定である。本システムで Firewall をまたいで木を構成するとき、子ノードからの接続を受け付けられなくなる可能性がある。今後、このような問題を解決する必要もある。

謝辞

本論文で使用した実験環境を利用させていただいた、株式会社ミントウェーブ様に感謝いたします。

参考文献:

[1] 平原 貴行、山之上 卓、安在 弘幸、有田五次郎「TCPを用いた分散環境のための電子黒板システムとその性能評価」,情報処理学会研究報告,2002-DSM-27-10,2002
[2] 山之上 卓「P2P 技術を応用した分散システムの排他制御機構の試作」情報処理学会研究報告 2003-DSM-29-3,2003
[3] 石畑 清 著「アルゴリズムとデータ構造」岩波講座ソフトウェア科学シリーズ3,岩波書店,1989
[4] 株式会社ミントウェーブ http://www.mintwave.co.jp/
[5] 柿原聡「Peer-to-Peer システムにおける動的な木構造の生成による検索の高速化」,東京工業大学平成13年度学士論文 http://www.csg.is.titech.ac.jp/paper/dice-bachelor2002.pdf
[6] 三村 和, 中内 清秀, 森川 博之, 青山 友紀: "RelayCast: ピアツーピア型ストリーム配信のためのミドルウェア," 電子情報通信学会技術報告, IN2002-42, July 2002.
[7] 三浦、花野、牛島、三上「P to P 型高品質ライブ配信技術」NTT 技術ジャーナル 2003.8 pp.12-15, 2003.
[8] 権藤克彦 著「Java によるプログラミング入門」(株)サイエンス社,2000
[9] 結城 浩 著「Java 言語で学ぶデザインパターン入門」(株)ソフトバンク,2001