実ネットワークの近接性を考慮した Vivaldi の実装と改良

木浦　正博

1　はじめに

近年，インターネットの発達とともに，ネットワーク負荷の増大が問題となっている．特に，ネットワーク上でやりとりされるデータの 60%以上が Peer-to-Peer(P2P) によるものである．

そこで P2P ネットワークは，実ネットワーク上に独自の仮想ネットワークを構築するためオーバーレイネットワークと呼ばれ，近年，注目されているファイル共有アプリケーションでは，目標ファイルを高速かつ低トラフィックで発見するために，現在までにさまざまなアルゴリズムが提案されてきた．

しかし，現在も P2P トラフィックは増加している．あるノードが遠方のノードにアクセスすることは，ネットワーク全体に対する負荷となるため，実ネットワークの近接性考慮が提案されている．

本研究では，その一手法である分散座標ネットワーク系アルゴリズムである Vivaldi[1] の実装を行う．また，Vivaldi は実ネットワークに近い距離をもつ座標系を構築するものの，通信遅延による三角不等式の不成立に対しては，正しい座標に収束できない可能性があるため，座標調節のための相手ノードの選択と，仮想ノードによる最適化を提案する．

2　P2P ネットワーク

2.1　P2P の概要

現在存在しているネットワークのほとんどが，組織や団体，国家に依存したネットワークである．ユーザーはプロキシサーバーや ISP を介してインターネットに接続している．P2P ネットワークはこのような既存のネットワークの上位に仮想ネットワークを構築し，ユーザーがサーバーや組織などを意識することなくネットワークにアクセスするための技術である．そのサービスはインスタントメッセージソフトウェアや近年，よく利用されるファイル共有ソフトウェアなどによって提供されるものが多い．

2.2　分散ハッシュテーブル

現在のファイル共有ソフトウェアの多くは，サーバーを必要とすることなく，クライアント同士が相互に通信を行うことで，データの送受信を行っている．そのため，特定のデータを持つクライアントを検索し発見するにはかなりの時間が必要である．分散ハッシュテーブルは，ハッシュ関数を用いてノードやファイルに固有の ID を付加することによって，検索を高速化するものである．特定のノードがもつファイルをノード ID の近接ファイル ID のものに限ることで，取得したいファイルを指定すれば，そのファイルの ID に近い ID を持つノードを発見することが目標ファイルを発見することにつながると言える．

しかし，これらの ID と実際のネットワークの近接性に相関はないため，発見するべき特定ノードの検索に非常に時間がかかってしまうことも考えられる．このような背景から，P2P，分散ハッシュの研究分野において実際のネットワークの近接性を考慮する Vivaldi が注目されている．

3　分散座標ネットワーク系 Vivaldi

3.1　Vivaldi の概要

Vivaldi では，物理的なばねの伸縮性を利用し，ノード間の遅延時間から仮想座標へのマッピングを行う．分散ハッシュテーブルにおけるノード同士の近接性はハッシュに基づく仮想的なものであった．しかし，データの検索の際には，この近接性だけではなく，実ネットワークの近接性を考慮しなければ，ネットワーク負荷を低減することはできない．Vivaldi には以下のような特徴がある．

- 分散型ネットワーク座標系
  各ノードが自律的に自身の座標を決定する．
- ばねの原理
  ユークリッド距離と実測値との誤差を徐々に修正し，最適化する．
- Piggyback
  アプリケーションレベルの通信に遅延測定のためのデータを付加するため，新しく通信を行う必要がなく，ネットワークトラフィックを減少できる．

Vivaldi は中央サーバーを用いることなく，各ノードが自律的に他のノードとの遅延時間から，2 乗誤差関数を利用して自身の位置を決定する．本章では，3.2 節と 3.3 節でそのアルゴリズムについて述べる．

3.2　Vivaldi のアルゴリズム

Vivaldi を実装した分散ハッシュテーブルライブラリである BambooDHT[2] を参考に解説を行う．
Vivaldi ではネットワークに参加しているすべてのノードにおいて，E を推定誤差，L_{ij} をノード ij 間の遅延時間，x_i をノード i の座標とした場合に，(1) 式によって求められる 2 乗誤差を最小にすることが目的である．

$$E = \sum_i \sum_j (L_{ij} - \|x_i - x_j\|)^2 \qquad (1)$$

これは物理的なばねの運動においてばね系全体のばねエネルギーを最小化することを模倣している．しかし，ネットワーク全体の推定誤差を求めることは困難であるため，各ノードの座標を分散最適化する．

まず，2 つのノード ij において，ノード i が調節すべき方向ベクトルは単位ベクトルを $u(x_i - x_j)$ として，(2) 式によって調節する値を決める．

$$F_{ij} = (L_{ij} - \|x_i - x_j\|) \times u(x_i - x_j) \tag{2}$$

次に方向ベクトルと現在の座標からノード i は F_i を最小とする方向に移動する．あるノードとその相手ノードとの距離が遅延時間よりも長ければ，徐々に座標を変更し距離を短くとる．逆に，遅延時間が長ければ，距離を長くとるように座標を調節する．しかし，その収束性は通信先のノードによるため座標が発振してしまう可能性がある．BambooDHT では定数 δ を時間の経過とともに減少させることにより，収束をはかる．(3 式)

$$x_i = x_i + F_i \times \delta \tag{3}$$

P2P ネットワークでは，一度にすべてのノードが参加するのではなく，ある程度の間隔をあけてノードが参加するため，あるノードが参加する際には，ネットワーク内のノードの座標は最適化されているものと考える．

3.3 高さベクトルの適応

インターネットにおける高速回線のみを用いてネットワークを構築すれば，ノード間の遅延時間は，物理的な距離との相関がある一方で，低速回線環境では，物理的な距離と遅延時間に相関がない．これは，低速回線においてインターネットコアノードにパケットがキューイングされる時間がより多くかかるためである．そのため Vivaldi では，ユークリッド距離ではなく独自の距離を適応する．Fig. 1 に示すように，コアノードを平面座標

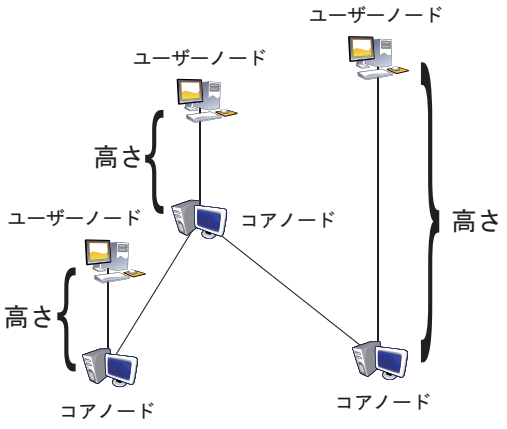

Fig. 1　独自距離の導入 (出典：自作)

上に配置し，実際に通信を行う各ノードをコアノードの上方に配置する．このモデルでは，ノード間の距離は，2 つのノードそれぞれの高さとそれらのノードのコアノード間のユークリッド距離の和となる．

先に挙げたモデルから Vivaldi プログラムではユークリッドベクトル演算を (4)-(6) 式のように拡張している．

$$[x, x_h] - [y, y_h] = [(x - y), x_h + y_h] \tag{4}$$

$$\|[x, x_h]\| = \|x\| + x_h \tag{5}$$

$$\alpha \times [x, x_h] = [\alpha x, \alpha x_h] \tag{6}$$

4 提案手法

4.1 Vivaldi の問題点

実ネットワークは，低速回線を考慮すればユークリッド距離を適応できないことが問題となる．Fig. 2 に示すように三角不等式が成り立たない場合を考えれば明らかである．Dabek らは，King data set[3] を用いて Vivaldi

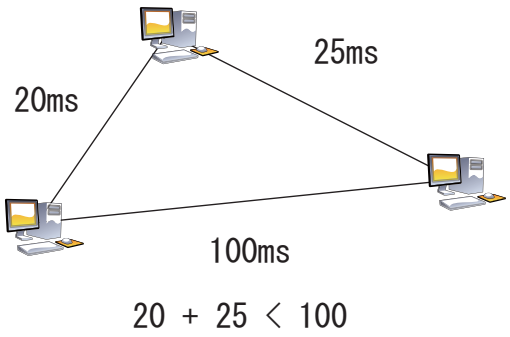

Fig. 2　三角不等式の不成立 (出典：自作)

の実験を行い，ノード間の遅延時間による三角不等式不成立は全体の 4.5%であったと述べているが，King data set は世界中の DNS 間の遅延時間であり，安定性の高い高速回線を用いているものがほとんどであると考えられる．そのため，低速回線が多いエンドユーザー間の通信においては，三角不等式の不成立が多いことが考えられる．また，Vivaldi ではばねモデルを採用しているため，三角不等式が成り立たない影響は全ノードの座標決定に影響を与える可能性がある．

4.2 仮想ノードの追加

4.1 節で挙げた問題を解決するため，座標系の改良を提案する．通信上問題となる遅延はあるノードに接続した複数のノードの中で低速回線のもの，すなわち遅延時間の長いものであり，ノードが自身の座標決定のために他のノードと通信する際には，なるべく遅延時間の短いノードと通信し，座標決定をすることが望ましい．

座標が最適化されたノードが低回線ノードと通信を行った場合，座標空間における距離と遅延時間の差が大きいことがわかるが，その場合には 2 つのノードの間に，仮想ノードを挿入し，仮想ノードの座標を調節することにより，2 ノード間の距離と遅延時間の差を減少させる．(Fig. 3)

5 評価実験

5.1 実験方法

実験では，ノード間の平均遅延時間をあらかじめ与え，それらの遅延時間を元に，Vivaldi における各ノードの座標の変化をシミュレートする．

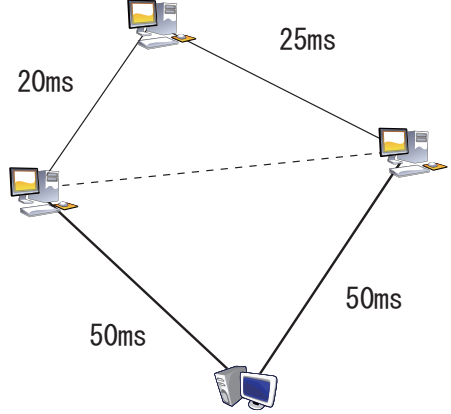

Fig. 3　仮想ノードの追加 (出典：自作)

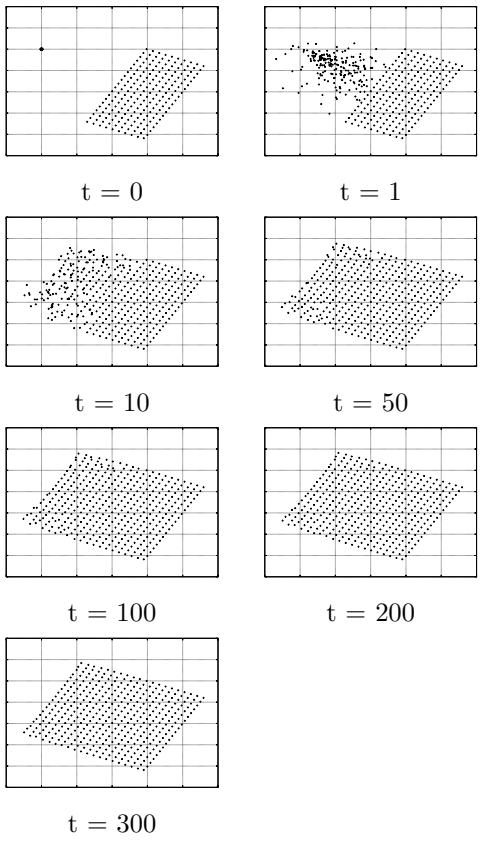

Fig. 4　実験結果 (出典：自作)

5.2　評価データ

評価データに関しては，400 ノードのうち，200 ノードが最適な座標に収束している状態から，新たに 200 ノードを追加し，それらのノードの収束性を確認する．

収束すべき 200 ノードは，遅延時間が 1000ms 以内のノードと通信することで座標修正を行い，三角不等式に違反するような長い遅延時間は無視する．

なお，仮想ノードの挿入に関しては，今回は行わない．

5.3　実験結果

200 ノードが等間隔に並んでいる状態から，新たに 200 ノードを原点に配置し，網上に並ぶ 200 ノードの内，遅延時間の短いノードをランダムに選択して座標を調節した．

経過時間 t の増加に伴う各ノードの座標を Fig. 4 に示す．

6　まとめと今後の課題

本研究では，ネットワークにおける遅延時間から仮想座標にノードを配置する Vivaldi アルゴリズムを実装した．また，低速回線によって三角不等式に違反するノードを含んだ上で，それらを無視して座標を最適化できることも確認した．

今回の実験では，実装をシミュレートするまでに止まったが，今後は，Vivaldi のパラメータを調整するとともに，実際に仮想ノードを挿入してノード同士が保有するルーティングテーブルから，最適な経路を選択し目標ノードにクエリを到達させる Agent を作成したい．

参考文献

1)　Frank Dabek, Russ Cox, FransKaashoek, and Robert Morris.: Vivaldi: A Decentralized Network Coordinate System In the Proceedings of the ACM SIGCOMM '04 Conference, Portland, Oregon, August 2004

2)　BambooDHT http://www.bamboo-dht.org/

3)　King data set http://pdos.csail.mit.edu/p2psim/kingdata/