M-050

Peer-to-Peer における Push 型情報共有を介したクラスタリング
Efficient clustering of P2P nodes by pushing messages.

安齋 嶺 † **寺田 実** ‡
Ryo Anzai Minoru Terada

1. はじめに

Unstructured P2P では検索を行う場合，有限な TTL(Time-To-Live) の範囲しか探索できない．そのため，ファイルを保持するノードの近くにいることが重要で，それを実現するためにノードの接続関係の変更によるトポロジの修正をここではクラスタリングと呼ぶ．例えば，ファイル共有ソフトでは回線の速度，所持するファイルや興味のあるファイルの種類などを条件にクラスタリングが行われている．

しかし，従来のファイル共有ソフトでは，クラスタリングを行う際に各ノードが 1 体 1 で情報交換を行っているため，クラスタが収束 (これ以上のノードの接続変更がなくなる) 状態になるまでに時間がかかってしまう．

そこで，ファイルを所持するノードが周囲に所持していることの情報を発信する Push 型の情報共有を介し，クラスタリングを行えば，収束までの時間を短縮することが可能となるはずである．

ただし，トポロジの修正には制約条件がある．P2P ソフトウェアではノードへの接続数には上限をおくのが一般的であるため，ファイルを所持するノードに接続が集中すると接続変更要求に応えることができなくなるのである．本研究では，こうした接続要求の集中を緩和する方式を提案する．

本研究では Gnutella 0.4 プロトコル [1] を拡張として，Push 型の情報共有を実現する手法を追加する．この拡張されたプロトコルを用いてクラスタリングのシミュレーションを行う．収束までにかかった時間とファイルを所持するノードを TTL の範囲内に含むノードの割合から有効性を検証する．

2. 提案手法

2.1 ノードの定義

以下で用いるノードの呼称をここで定義する．

- 同類ノード：興味が同じノードを指す．
- 所持ノード：ファイルを所持するノードを指す．
- 到達ノード：所持ノードの TTL の範囲にある同類ノードを指す．

2.2 Posh を利用したクラスタリング

Gnutella 0.4 プロトコルのデスクリプタのフォーマットに従い Posh デスクリプタ (通信されるメッセージ) を新たに定義する．Payload には発信元と所持しているファイルのメタ情報が記載される．

この Posh はファイルを所持するノードが以下の手順にしたがって送信する．

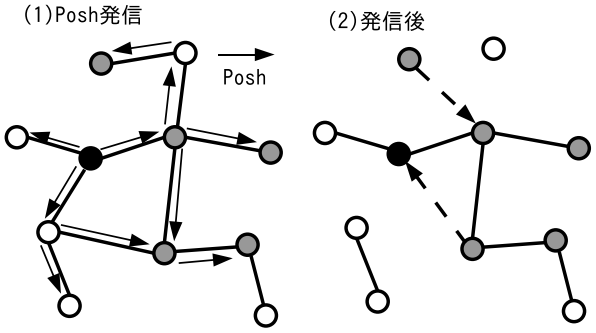

図 1: Posh を利用したクラスタリング：黒は所持ノード，灰は同類ノード，点線は新しい接続

1. 所持するファイルからひとつをランダムに選択する．
2. そのファイルに関するメタ情報を payload に記載する．
3. 発信元として自分のアドレスを同じく payload に記載する．

この Posh は Flooding で送信されるが，途中の転送は以下の手順に従う．

1. 受信した Posh のメタ情報に自分が興味があるか調べる．
2. **興味があれば，発信元に接続を要求し，payload の発信元のアドレスを自分のものに書き換える．**(図 1(2))
3. TTL を 1 減らし，Flooding を行う．

Posh の転送の結果として興味が同じノード同士が接続する．しかも，その Posh はファイルを所持するノードを起点として発信されているので，興味が同じで欲しいファイルが発見できるクラスタが生成されていると言える．

この手法は所持ノードへ接続 (要求) が集中せず，所持ノードの周囲にいる到達ノードへ分散される．

2.3 QueryHit を利用したクラスタリング

比較対象として既存の Query と QueryHit を利用したクラスタリングを実装した．動作は以下の通り．

1. ファイルを発見するために Query を発信し，Flooding によってファイルを所持するノードへ届く．
2. QueryHit を Query の発信元へ返す．(図 2(1))

†電気通信大学 電気通信学研究科 情報通信工学専攻
‡電気通信大学 電気通信学部 情報通信工学科

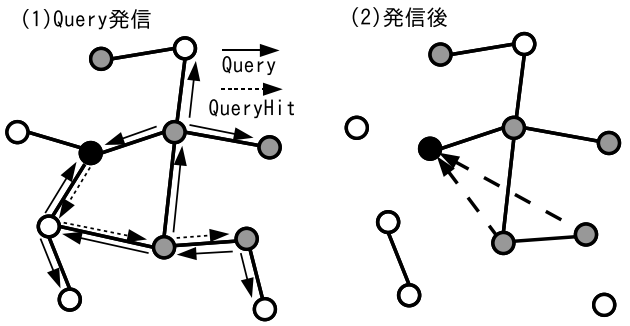

図 2: QueryHit を利用したクラスタリング：黒は所持ノード，灰は同類ノード，破線は新しい接続の要求．発信元は右下の同類ノード

3. **QueryHit の経路上で QueryHit の内容に興味のあるノードはファイルを所持するノードへ接続を要求する．**(図 2(2))

2.4 ランダムに接続するクラスタリング

もう一つの比較対象としてランダムに接続するクラスタリングを実装した．動作は以下の通り．

1. 近隣ノードのリストからランダムに 1 ノードを選択する．
2. 選択したノードに興味を判定するためのメタ情報を送信し，受け取ったノードは同じ興味を持っていれば接続する．

2.5 接続の切断

それぞれのクラスタリングの手法において，接続数に上限が設けてある場合，上限に達したら接続の切断を行う．

切断されるのは同類ノード (同じ興味を持つノード) ではないノードである．上限に達したノードが同類ノードではないノードと接続していない場合は，切断を行わず，接続要求を受け付けない．

なお，切断によって，どのノードとも接続していない非連結状態になることは考慮されていない．そのため，クラスタリングの途中で同類ノードではないものは非連結状態になることがある．

3. シミュレーションの設定

初期接続の生成は総ノード数と接続の総数を与えてランダムに生成している．また，シミュレーション中にノードの新規参加，離脱はない．

隣接ノードへの通信は 1 単位時間で完了することとした．Flooding の際，全隣接ノードへの送信も 1 単位時間で完了する．

実世界では興味やファイルは複数あるが，今回は 1 種類のみとした．また，シミュレーション中に所持するファイルや興味が変わらない．

所持ノードの数は 2, 5, 10, 20 個，同類ノードの数は 100, 200, 500 個を組み合わせ，それ以外のノードも含め，総ノード数は 10000 でシミュレーションを行った．

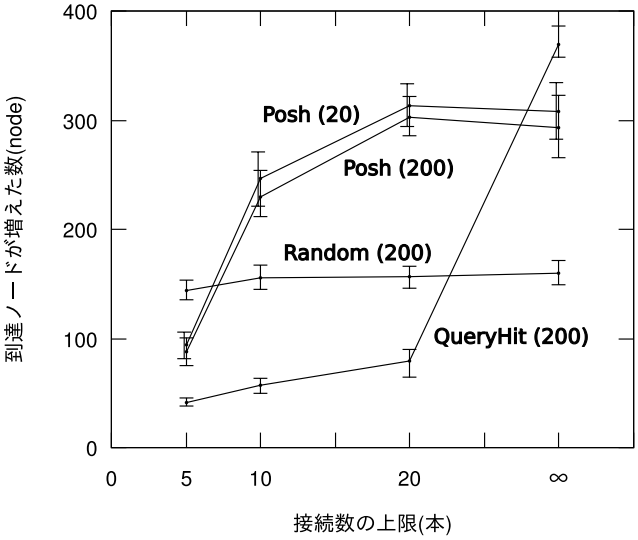

図 3: 接続数制限の上限と到達ノードの増加数 (所持ノード 10，同類ノード 500)

また，各デスクリプタを発信する間隔を 200 単位時間間隔のポアソン分布とした．ただし，Posh についてはデスクリプタを発信するノード数が少ないので 20 単位時間間隔でも実施した．

各ノードにおける接続数の制限は 5, 10, 20, または制限なしと設定されている．

それぞれのシミュレーションは 1 回につき 1000 単位時間と 10000 単位時間で実施した．

4. 結果・考察

4.1 クラスタリングの効果

まず，クラスタリングの効果を確かめるため，到達ノード (所持ノードの TTL 範囲内にある同類ノード) の増加量を測定した．

所持ノード数を 10，同類ノード数を 500 と固定したとき，1000 ステップ後の接続数制限の上限による到達ノードの増加量の違いを調べた．その結果は図 3 のようになった．エラーバーは 25%点と 75%点を示す．

ノードあたりの接続数に上限がある状況においては Posh は十分に有効であった．特に接続制限 20 本については QueryHit の 3 倍以上，Random の 2 倍程度になった．これは，QueryHit のように所持ノードへ集中させずに，到達ノードへ接続を分散させたためである．また，到達ノードと所持ノードの間にある関係の無いノードをショートカットできるように接続するため，ある程度離れた同類ノードも到達できるようになったといえる．

実際の P2P ソフトウェアの多くは接続数に制限がある．Posh は十分に有効であるといえる．

接続数に制限がない場合には，QueryHit では QueryHit の経路上の同類ノードはすべて所持ノードに隣接するようにトポロジが変更されるので，その同類ノードを経由した大幅な到達ノード数の増加が可能となることが読み取れる．

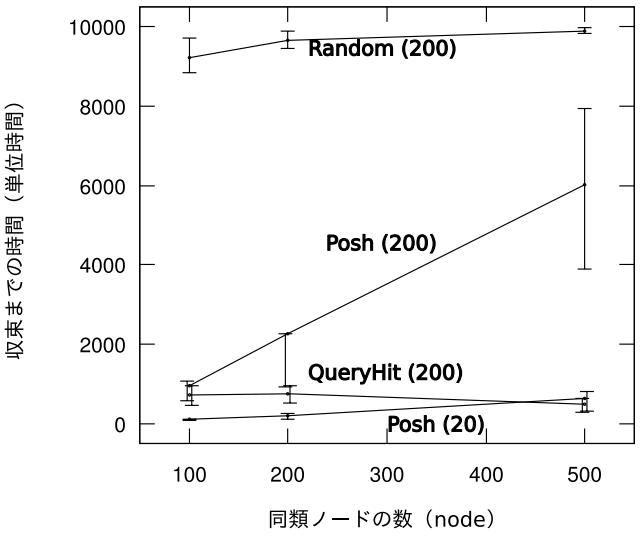

図 4: 同類ノードの数と収束までの時間 (所持ノード 10, 接続数制限の上限 10)

4.2 収束までの時間

次に，所持ノードを 10，接続数制限の上限を 10 としたとき，収束までにかかった時間を調べた．ただし，時間までに収束しなかった試行については 10000 単位時間で収束したものとして算出している．その結果は図 4 のようになった．図 4 で Posh の後につけた () 内の数字は Posh デスクリプタの発信間隔である．エラーバーは図 3 と同様である．

同類ノード数が 100, 200 では Posh(20) が他の手法よりも早く収束している．その後は QueryHit が早く収束するようになっている．

同類ノードの少ない状況においては Posh は QueryHit の手法と同程度か早く収束することが可能である．

5. 先行研究

5.1 Gnutella 0.4

Gnutella は Pure P2P のファイル共有ソフトウェアである．このプロトコルはデスクリプタとそのルーティングの方法を定義している．プロトコル仕様にはクラスタリングに関する言及はなく，実装に任されている．

5.2 Gnutella 0.6[2]

このプロトコルで Ultra Peer という処理能力の高いノードを中心に木構造のようなクラスタリングを行う．しかし，そのクラスタ内に欲しいファイルを所持するノードがいるとは限らない．

5.3 pushare[3]

pushare は Push 型の情報共有を実現するファイル共有ソフトである．デスクリプタは提案手法と同じような手法で転送される．ただし，発信元のアドレスを書き換えるのはファイルを所持している場合に限られる．そのため，所持ノードへの接続が集中しやすい．

6. 今後の課題

6.1 複数の興味での実施

実ネットワーク上では複数の種類の興味が存在する．これを考慮した場合，受信したデスクリプタに対して，興味があると判定するのが難しくなる．あるノードが持つ複数の興味について，ひとつでも該当すれば興味ありとすることもできるし，全てを該当して興味ありとすることもできる．その判定によっては，うまくクラスタが構成できない可能性があるかもしれない．それを確かめる必要がある．

6.2 総ノード数を増やしての実施

実ネットワーク上では相当な数のノード数を保有するソフトウェアは少なくない．より，現実に近づけるために総ノード数を今回の 10000 ノードから 100 万ノード程度まで増やして実施し，提案手法の動作を確かめる必要がある．

6.3 トラフィックの比較

今回，考慮しなかったもののひとつとしてトラフィックがある．

どんなに，シミュレーション上で収束までの時間が短縮できたとしても，トラフィックがあまりにも大きいと，実ネットワーク上ではうまく作用できない可能性がある．

Gnutella をベースで行う場合，その接続を維持するためのデスクリプタ (Ping, Pong) などもあり，実際にかかる負荷を検証する必要性がある．

参考文献

[1] The Annotated Gnutella Protocol Specification v0.4, `http://rfc-gnutella.sourceforge.net/developer/stable/index.html` .

[2] RFC-Gnutella 0.6, `http://rfc-gnutella.sourceforge.net/developer/testing/index.html` .

[3] pushare, `http://fuktommy.com/pushare/` .