マルチメディア通信と分散処理 107－16
（２００２．３．２９）

ファイル発見のためのネットワーク負荷を抑える転送先学習アルゴリズム

川西智也
京都大学工学部情報学科

中村素典
京都大学総合情報メディアセンター

美濃導彦
京都大学総合情報メディアセンター

概要　現在ネットワークを介したファイルの共有への需要が高まっている．これを実現するために，ピアツーピア環境でファイル発見を行うという手法がある．これには導入の容易さや，耐障害性など利点もあるが，ネットワーク負荷が高くなるという欠点を持つ．これはファイル発見用のパケットをすべての隣接ノードに転送するため，パケットの数が爆発的に増加するからである．そこで本稿では確率的転送と学習アルゴリズムを利用し，転送先とする隣接ノードの数を減らすことで，ネットワーク負荷を軽減する手法を提案する．本手法の有効性を検証するためシミュレーション実験を行った結果，同程度の負荷で 6 倍程度の応答が返ることを確認した．

An Algorithm of Learning Next Hop to Reduce Network Load For File Discovery

KAWANISHI Tomoya
Dept. of Information Science, Kyoto University

NAKAMURA Motonori
Center for Information and Multimedia Studies, Kyoto University

MINOH Michihiko
Center for Information and Multimedia Studies, Kyoto University

Abstract　There is a growing demand for sharing files via network now. Discovering files in Peer-To-Peer network is used as one way to share files. This is easy to set up and hard to fail. But network load is heavy because a packet to discover files is forwarded to all neighboring nodes and the number of packets increases explosively. To reduce network load, we present a method of probabilistic forwarding and learning algorithm to reduce the number of nodes which a packet is forwarded to. Simulation experiments show that our method can discover six times more files under same network load.

1　はじめに

ネットワークを介して各ノードがファイルを提供・共有の両方を行う方式には，ファイルの発見をすべてのノードが協働することで実現する分散的な方式とファイル発見をサーバで一元的に行う集中的な方式がある．

分散的な方式は，集中的な方式と比べて導入が容易であることや耐障害性があるといった長所がある．これらの点から，本稿では分散的な方式に着目する．しかし一方，この方式にはネットワーク負荷が高いという短所がある．

ネットワーク負荷が高い理由は，ファイル発見を行うとき問い合わせを行うパケットを過剰に多くのノードに転送していることにある．ノードにより提供するファイルの質・量は偏在しているが，現在の実装ではその点が全く考慮されていない．

本稿では，この点を考慮して発見処理を改善した手法を提案する．本手法では転送を確率的に行い，その確率を学習アルゴリズムによって更新することにより，ネットワーク負荷を軽減する．

2　ピアツーピア型ファイル発見

この章ではそのサービスに参加するすべてのノードが協働して，ファイル発見する仕組みについて説明する．これをピアツーピア型ファイル発見と呼ぶ．この方式では IP 層のネットワーク構造の上に独自の論理ネットワークを構築する．そのネットワーク上で，あるノードと論理的な接続関係を持つノードのことを隣接ノードという．

この方式として著名な Gnutella のファイル発見の手順をまとめると以下のようになる．

1. ファイルを発見しようとするノードは，Query パケットを生成する．

2. Query パケットはすべての隣接ノードに対して転送される．
3. Query パケットを受信したノードは，そのすべての隣接ノードにさらに転送するという処理を繰り返す．(ブロードキャスト)
4. Query パケットを受信したノードに求められているファイルがあれば，QueryHit というパケットを生成する．
5. QueryHit パケットは，Query パケットが通った経路を逆向きにたどるように転送され，その Query パケットを生成したノードに QueryHit パケットが返される．

QueryHit パケットが Query パケットの通った経路を逆向きにたどるための仕組みは非常に簡単である．ノードは Query パケットを転送するときに，それがどの隣接ノードから来たのかを覚えておく．そのために Query パケットには一意な識別子が割り当てられている．QueryHit パケットは対応する Query パケットと同じ識別子になるようになっており，ノードは QueryHit パケットを受信すると，対応する Query パケットが送られてきた隣接ノードに転送する．

以上のような転送の仕組みの他に，際限なく転送が繰り返されることを防ぐため，TTL(Time To Live) と複数回受信パケットの破棄の 2 つの仕組みが用意されている．

Query パケットは TTL の回数だけ転送されると，それ以上転送が行われず破棄される．その結果 Query パケットは生成したノードから TTL の回数以内のホップで到達可能な範囲にしか到達しない．

複数回受信パケットの破棄とは，同じノードが同じ識別子の Query パケットを 2 回以上受信した場合，2 回目以降に受信されたパケットはそれ以上転送処理されず，捨てられることである．これにより，同一の Query パケットが異なる経路を通って同一のノードに到達した場合でも一回しか転送しなくなる．

3　ネットワーク負荷の軽減手法

ピアツーピア型ファイル発見において，現在実装に使われている Query パケットをブロードキャストによって転送するという手法では Query パケットの数が指数関数的に増加する．Query パケットの数が増加すると，ネットワーク帯域や計算資源が消費され，同時使用している他のアプリケーションに悪影響を与えることになる．実際，Gnutella を使用するとき，165kbps のネットワーク帯域が平均して占有されることが報告されている [2]．これは広く利用されている ISDN 回線の 64kbps よりも大きく，より広帯域な ADSL 回線にとっても常時この帯域が使用されてしまうことは大きな負担である．

そこで，Query パケットの増加を抑えることによって，ネットワークの負荷を軽減する手法を提案する．

基本的なアイデアは「すべての隣接ノードに Query パケットを転送する」代わりに「より多くより早く発見できる隣接ノードに Query パケットを限定的に転送する」ことである．転送先が少なくなればその分 Query パケットの数の増加は抑えられることになる．

このような方法が適用できる理由は，ノードが保有するファイルの質・量には偏りが大きいからである．例えば，何もファイルを提供しないがそのサービスに参加しているノードが Gnutella では 70%程度存在している [3]．一方，一部のノードが大量のファイルを提供している．例えば上位 1%のノードが 37%ものファイルを提供している [3]．その結果，ノードによって応答する確率には大きな違いがある．

応答が返りやすい隣接ノードに対して優先的に転送するために，本稿では転送する確率を隣接ノードに割り当て，その転送確率を 3.2. 節で示す学習アルゴリズムによって逐次更新，改善する手法を提案する．より早く応答が返る隣接ノードに転送する確率をより高くすることによって，転送先を限定しネットワーク負荷を軽減できると考える．

3.1.　転送先の決定と転送確率の更新

ここでは，本手法でどのようにしてファイル発見を行うのかについて詳説する．

3.1.1.　転送先の決定

ファイル発見を要求するノードにおいて Query パケットが生成した時や他からの Query パケットを受信した転送するときはすべての隣接ノードの中から転送先隣接ノードが確率的な機構により選ばれて，転送される．

転送処理を行うときは，その転送処理を行った時間を記憶しておく．これは，3.2. 節で示す学習を行うときに使用する．

転送するときは次の作業を N 回だけ繰り返す．N は隣接ノードを選択する回数である．その結果，同じ隣接ノードが複数回選択されたとしても，実際には一回しか転送しない．パラメータ N は受信した 1 つの Query パケットをどれだけ多くの隣接ノードに

転送を行うかを決めるパラメータで，N を多くすると，転送確率が低い隣接ノードに対しても転送する可能性が高くなる．

転送するときの隣接ノードの選び方には，次の 2 とおりあり，そのどちらになるのかは探求確率 P_e という確率で決定される．

- P_e の確率で転送先をランダムに決定する．
- $(1-P_e)$ の確率で，隣接ノードごとの転送確率にしたがって転送する．

開始直後の状態では，転送確率はすべての隣接ノードに対して等確率とする．これによりほぼランダムに転送されることになる．

学習が進むと，発見されるファイルの数が多く，発見するのにかかる時間が短い隣接ノードに転送される確率が次第に高くなり，そうでないノードへ転送する確率は次第に低くなる．学習が進むと上の操作を N 回行ったとしても，少数の隣接ノードにのみ転送する状態になることが期待される．

探求確率 P_e でランダムに転送する手続きはネットワークの変化に対応しやすくするために用意されている．これが小さいと硬直的な転送となり，大きいとネットワーク負荷を抑える効果が小さくなる．

3.1.2. 発見した時の処理

ファイルを発見したときは， QueryHit パケットを生成する．QueryHit パケットは対応する Query パケットと同じ識別子を持つ．

3.1.3. 応答と転送確率の更新

QueryHit パケットが生成され，対応する Query パケットが生成されたノードにまで返されるとき，経由するノードで Query パケットの転送確率を更新する．このときの経路は対応する Query パケットと同じで向きが逆となっている．

QueryHit パケットが転送されるとき，それがやってきた隣接ノードに対する Query パケットの転送確率を大きくし，それ以外の隣接ノードに対する転送確率を小さくする．このようにすることで，発見をしやすくすることができる．転送確率の更新はその Query パケットを生成したノードだけではなく，QueryHit パケットを中継する途中のノードに対しても行う．

転送確率の更新は QueryHit パケットが早く返ってくればくるほど強く学習を行うようにする．そうすると，早く応答が得られるようになると期待される．

3.1.4. キューにおける待ち

キューとは，パケットをすぐに処理できない場合に，一時的にそれを保存しておき，パケットを処理できるような状態になってから処理を行うために溜めておくバッファのことである．

キューでは以下の順番でパケットを処理する．

1. QueryHit パケットが到着するとすでにキューにある Query パケットより優先され，その Query パケットより前に挿入される．先に待っている QueryHit パケットがある場合はその最後にまわされる．
2. Query パケットが到着するとキューの一番最後に待たされる．

QueryHit パケットを優先的に転送することによって，応答が早くなるだけでなく，学習がより速く進む．

3.2. 転送先学習アルゴリズム

3.2.1. AntNet

IP ルーティングにおける非常にロバストでかつ効率的な手法として，AntNet という手法が提案されている [4]．AntNet は強化学習の１つであり，アリの巣でのアリの挙動に触発されて着想された IP ルーティングアルゴリズムである．AntNet は様々な環境で実験された結果，既存のアルゴリズムと比べて，遅延や負荷の少ない転送を実現している．

本方式ではこの AntNet を参考にして，転送アルゴリズムを設計した．その理由は，AntNet に，以下のような利点があったからである．

- IP ネットワークに対して実験した結果，良い結果が得られた実績があったこと．
- 自律分散型の学習アルゴリズムであること．
- アルゴリズムそのものが単純であること．
- Gnutella と対応をつけやすいこと．
- 保持する必要があるデータ量が少ないこと．

AntNet では新しいルートを探索するエージェントとして働くパケットと，それによって発見されたルートを今後通りやすくするためのパケットによって，転送確率を更新し，パケットの効率的な転送を実現する．本提案手法では Gnutella での Query パケットを前者に，QueryHit パケットを後者に対応づけることによって，AntNet[4] の学習アルゴリズムを適用する．

3.2.2. 転送先学習アルゴリズムの詳細

Query パケットをある隣接ノードに転送する確率は転送先学習アルゴリズムによって更新・決定される．転送先学習アルゴリズムは次の 3 つの値から Query パケットの転送確率を更新する強さを決定する．

- 応答時間 T．Query パケットがそのノードを通り，対応する QueryHit パケット が，そのノードに戻ってくるまでの時間．
- そのノードでの応答時間の平均 μ．
- そのノードでの応答時間の標準偏差 d．

これらの値を取得するために外部のノードと情報交換の必要がなく，自律的なアルゴリズムである．

また，これらの値しか使用しないため，学習のために必要な記憶域は少ない．Query パケットを転送した時刻を受信した Query パケットの識別子の数，それに平均応答時間とその標準偏差である．転送した時刻は受信した Query パケットの数の識別子の数だけ必要だが，従来手法でもその Query パケットがすでに通ったことがあるかどうかを同じ数だけ記憶する必要がある．

この転送先学習アルゴリズムでは，まず次のように r_0 を算出する．

$$r_0 = \begin{cases} \frac{T}{c\mu} & \text{if} \quad \frac{T}{c\mu} < 1 \\ 1 & \text{otherwise} \end{cases}$$

こうして算出される r_0 は，その応答時間 T がどれだけ良かったかを示す無次元の値である．応答時間が短ければ，r_0 は小さくなる．つまり，r_0 は小さいほど良い．ここで，μ は応答時間の平均，c はそれを補正するパラメータである．1 を超えた場合は 1 にする．この個所はオリジナルの AntNet と同じ処理である．

次に，その応答時間 T がどれだけ信頼できるかという観点で，次の式で算出される値 U を用いて，r_0 を調整する．e は自然対数の底である．

$$\begin{aligned} U &= \begin{cases} \mathrm{e}^{-a\frac{d}{\mu}} & \text{if} \quad r < t \\ \mathrm{e}^{-a'\frac{d}{\mu}} - 1 & \text{if} \quad r > t \end{cases} \\ r_1 &= r_0 - U \end{aligned}$$

U は標準偏差 d に関して単調減少の関数であり，標準偏差 d が小さくなると，大きな値になる．U を引くことで r の値を小さくする．ここはオリジナルの AntNet に比べ若干簡略化している．

ここで，閾値 t よりも r_0 が小さいとき (良い値のとき)，U は正であり，閾値 t よりも r_0 が大きいとき (悪い値のとき)，U は負の値である．これによって，短い時間で応答が返ってきた場合に特に強く学習することにできる．

次に，$r \in [s, 1]$ となるように調整する．s は非常に小さな値で，過剰に学習することを防ぐために用意されたパラメータである．$s = 1.0 \times 10^{-20}$ に設定している．これはオリジナルの AntNet では $s = 0$ となっている．

$$r_2 = \begin{cases} 1 & \text{if} \quad 1 < r_1 \\ r_1 & \text{if} \quad s < r_1 < 1 \\ s & \text{if} \quad r_1 < s \end{cases}$$

最後にこの r_2 をべき乗する．

$$r_3 = r_2^h$$

こうして得られた r_3 を用いて転送確率を次のように更新する．ここで，P_f は QueryHit パケットが転送されてきた隣接ノードへの転送確率で P_n はそれ以外の隣接ノードへの転送確率である．

$$\begin{aligned} P_f &= r_3 P_f + 1 - r_3 \\ P_n &= r_3 P_n \end{aligned}$$

QueryHit パケットを転送してきた隣接ノードの先には応答を返したノードがある．その隣接ノードに転送する確率を上げ，それ以外の隣接ノードへの転送確率を下げることによって，転送した結果応答が返る確率を高めている．

4 実験と評価

3.2. 節で提案した手法が Gnutella のように，すべての隣接ノードに Query パケットを転送するような方法 (ブロードキャスト方式) に比べて，ネットワーク負荷，発見できた数といった観点で優れていることを示すために実験を行った．

4.1. 実験での目標と評価基準

評価基準にはネットワーク負荷の評価基準として転送回数，応答結果の良さの評価基準として応答数を採用した．

ネットワーク負荷の評価基準

- **滞留数**　ネットワーク内にあるノードのキューで待っているパケットの数の平均．
- **転送回数**　生成された 1 つの Query パケットが転送されることで TTL がゼロになったり同じノードに複数回到達したりして，その Query パケットの子孫がすべて消滅するまでに転送される回数の平均．

応答結果の良さの評価基準

- **応答数**　生成された 1 つの Query パケットに対して返される QueryHit パケットの数.

滞留数が増えるとノードでの待ち時間が長くなるため，応答が返るまでの時間が長くなる．

転送回数や応答数は TTL や隣接ノード数によって変化する．TTL や隣接ノード数を増加させると，転送回数が増加し，Query パケットを生成したときに到達可能となるノード数が増加する．そうして，到達できるノード数が増加すると，応答が返る数が増えることが予測される．これは，ユーザにとって選択肢が増えることになる．

転送回数が多いとパケットを転送するため，ノードの計算資源や，ネットワーク帯域をより消費することになる．そうすると，それらの資源を利用する，他のアプリケーションの動作を圧迫することになる．

本稿での目標としては滞留数の増加が抑えられていることと，同じ転送回数で数多く発見することとした．

4.2. 実験で想定する環境

シミュレーション実験を行うにあたり，次のような仮定を置いた．

1. 離散時間でシミュレーションを行い，1 つのパケットを 1 つの隣接ノードに転送するのにかかる時間を，1 単位時間とした．これは，あるノードが単位時間あたりに転送できるパケットの数は 1 個であり，転送にかかる時間は転送先となる隣接ノードの数に比例してかかるということである．したがって，あるノードがパケットを 4 つの隣接ノードに転送するには 4 単位時間が必要となる．IP ネットワークでユニキャストを繰り返すことによって，隣接ノードに転送するためこのような仮定にした．
2. パケットがキューで待つ個数に限界があるとした (本方式では 35，ブロードキャスト方式では 100)．キューからあふれたパケットは破棄され，それ以上の転送処理を行わない．これによって，トラフィックが爆発してしまった場合でも正常な処理ができる．
3. ファイルを提供するノードの配置や接続関係はランダムとした．
4. ノードは 0.01 の確率でファイルを提供するノードであるものとした．ノード数 10000 の場合は約 100 個のノードがファイルを提供していることになる．

実験においては，パケット生成率，TTL，隣接ノード数を変化させながら実験した．パケット生成率とは，各ノードが 1 単位時間に，どれくらいの確率で 1 個の Query パケットを生成するかを示した確率である．ノード数が 10000 でパケット生成率が 0.0001 の場合，平均して 1 ターンに 1 個のパケットが生成される．

さらに転送先学習アルゴリズムでの各種パラメータは $c = 2$，$t = 0.45$，$a = 10$，$a' = 9$，$s = 1.0 \times 10^{-20}$，$h = 0.05$ とした．また，転送処理の際のパラメータである探求確率 P_e は 0.05，隣接ノードの選択を行う回数 N は隣接ノード数の 1.5 倍に設定した．

4.3. シミュレーション結果

この節ではシミュレーションを行った結果について述べる．

4.3.1. 滞留数の変化

この項ではノード数は 10000，隣接ノード数は 4，TTL は 7 として，時間によって滞留数がどのように変化するかについてシミュレーションした結果を示す．図 1 は，ブロードキャスト方式の場合に，図 2 では，本方式の場合にどのように変化したのかを示している．両者ともパケット生成率は 0.0001 から 0.001 の間で変化させたものをプロットしている．

この図の横軸 (turn) は，シミュレーション開始からの単位時間を，縦軸は滞留数 (Average Packets In Queue) を示している．滞留数は少ないほどネットワークに対する負荷の観点から望ましい．プロットするときには，全体の傾向を見るために，100 単位時間の間の平均をとっている．

ブロードキャスト方式では，パケット生成率が高くなるにつれ，滞留数は急激に増加する．

一方，本方式では，滞留数は時間が経つとともに大きく減っていくことが分かる．これは学習が進むことにより転送先となる隣接ノードが限定され，Query パケットの増加が抑えられたことの結果である．

ここで Y 軸の目盛りを見ると本方式の方が滞留数が圧倒的に少ないことが分かる．

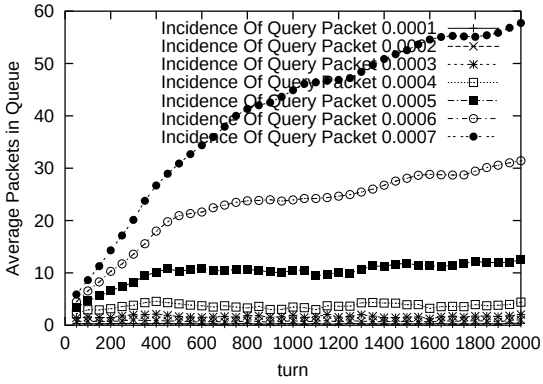

図 1: ブロードキャスト方式

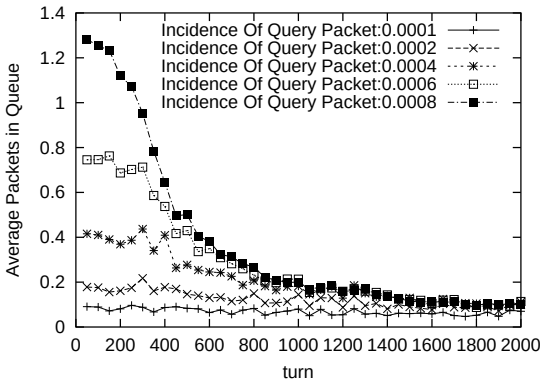

図 2: 本方式

4.3.2. 転送回数と応答数

ノード数を変化させたときにどれだけネットワーク負荷つまり転送回数と応答数との関係がどう変化するのかについて示す.

転送回数も応答数も TTL と隣接ノード数によって決定される．そのため，TTL や隣接ノード数を様々な値に設定し，それによって得られた転送回数や応答数でプロットしたものが図 3 である.

横軸は転送回数 (Network Load)，縦軸は応答数 (The Number Of Discovery) とした．ノード数を 1000，4000，15000 に設定したときの転送回数と応答数の関係を示している．本方式の点群は ANT，ブロードキャスト方式の点群には GNU と印字している.

図 3 を見ると，本方式で行った場合は，ノード数の増加に対して，返る応答が大きく増えることが分かる．それに対して，ブロードキャスト方式の場合は，ノード数が増えても，返る応答数は微増にとどまっている．ノード数が多くなるほど，本方式の優位が強まっていることが分かる.

さらに，ノード数が少ないときでも，常にブロードキャスト方式よりも少ない転送回数で同じ応答数が返るようにできていることが分かる．ノード数が少ないときでも本方式で行う方が有利である.

ノード数が 15000 のときを比較すると，転送回数が 400〜2500 のとき 6〜8 倍程度，多く発見できることが分かった (最高は転送回数が約 800 のとき).

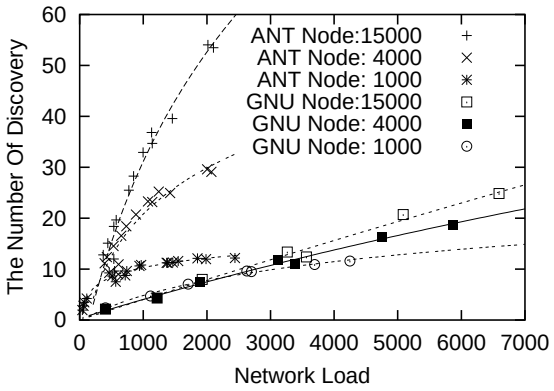

図 3: 転送回数と応答数

5 結論

本報告書では，Gnutella のような純粋型ピアツーピア環境において少ないネットワーク負荷で多くのファイルの発見を実現する転送先学習アルゴリズムを提案した．本方式は，自律的であり，ノードに必要とされるメモリも少なく，情報交換のために多くのパケットを必要とすることもない．このことを実際にシミュレーションすることで確認した.

今後の課題としては，ノードの応答確率の違いや，位置的または時間的に突発的に多く Query パケットを生成された場合について，調べる必要がある.

参考文献

[1] *Gnutella Protocol Specification v0.4* http://www.clip2.com/GnutellaProtocol04.pdf

[2] Jason Wang *Gnutella Bandwidth Usage* http://resnet.utexas.edu/trouble/p2p-gnutella.html(2001)

[3] Eytan Adar and Bernardo A. Huberman *Free Riding on Gnutella* http://www.hpl.hp.com/shl/papers/gnutella/Gnutella.pdf (2000)

[4] Gianni Di Caro, Marco Dorigo *AntNet: A Mobile Agents Approach to Adaptive Routing* http://aepia.dsic.upv.es/revista/numeros/12/baran.pdf (1997)