ファイル細分化とノード集合化を用いた巨大ファイル配布用P2Pの提案

湯澤孝有[†] 服部晃和[†] 横田隆史[†]
大津金光[†] 馬場敬信[†]

近年，ブロードバンドの普及により，コンテンツ配信等の大規模データ転送が行なわれるようになった．これらは従来クライアントサーバ型ネットワーク (CS)により主に運用されているが，トラフィックの増大と，対処するためのコストの増加のために拡張性の限界が指摘されている．この解決方法としてP2P (Peer to Peer)の技術が注目されている．現在，様々なP2P技術が開発されているが，拡張性，信頼性，転送効率の全ての点を満たす技術は提案されていない．本稿では，Fragment Pool (FP)と呼ぶノード集合を導入することで上記問題点を解決するP2Pプロトコル FONet (Fragment pool Organizing Network)を提案する．FONetでは対象となるファイルを一定の大きさに細分化し，ネットワーク中に分散配置することで負荷分散を図る．ネットワークに参加しているノードの活動に伴いファイル断片の分布に偏りが生じるが，FONetでは，この偏りを基にファイルの存在確率の高いノードの集合(FP)を形成する．FPのメンバノードを中心にファイルの検索を行うことで，効率・拡張性共に優れたP2Pシステムを構築できる．提案方式についてシミュレーション評価を行った結果，CS，Gnutellaよりも効率・拡張性に優れる結果となり，FONetの有効性を示すことができた．

A proposal of an efficient P2P for huge file distribution using file division and node grouping

TAKATOMO YUZAWA,[†] AKIKAZU HATTORI,[†] TAKASHI YOKOTA,[†]
KANEMITSU OOTSU[†] and TAKANOBU BABA [†]

Spreading broadband network environment enables everyone to handle network contents with large-scale data transfer. Although these has been managed by Client-Server model, limitation of the model has been pointed out because of its less scalability caused by network traffic congestion. The P2P (Peer to Peer) technologies are focused as a solution of this problem. Although various P2P technologies has been developed, there has been no enough P2P technology that answers the requirements on scalability of its network size, reliability and efficient data transfer. In this paper, we propose a novel PureP2P protocol named FONet (Fragment pool Organizing Network) that has FP (Fragment Pool) to overcome these problems. In FONet, each file is divided to small fragments and these fragments are distributed in the network for load balancing. Deviation of distribution of fragments arises from actions of nodes in the network. FONet promotes the formation of nodes which is likely to have fragments of target file on the basis of this deviation. FONet achieves high collection efficienty and scalability by using FPs. Results of simulation reveal the effectiveness of FONet.

1. はじめに

近年，Linuxディストリビューションや動画ファイルのような巨大なファイルをインターネット上で配信する需要が高まっている．現在，インターネットでは主にクライアントサーバ型ネットワーク (CS)を用いてファイルの配布が行われている．しかし，巨大なファイルを配布するときアクセスが集中すると，サーバと通信路に負荷がかかり，ファイル転送に要する時間が増加し，配布効率が低下する．サーバと通信路の負荷の集中に対応するには，サーバの増強またはロードバランサを導入したクラスタサーバの構築，十分に広い帯域幅をもつ通信路の確保が必要となる．しかし，これらの実現にはコストが必要となる．

そこで，このようなコンテンツ配布のための技術としてP2P (Peer to Peer)が注目されている．P2Pならば，コンテンツ配布者は高価なサーバを用意しなくても，消費者のノードによる自律的な配布により，広くコンテンツを配布できる．これにより，サーバの構築や運用といったコストを削減できる．しかし，既存のP2Pには巨大なファイルの配布を対象としたものがいくつか有るが，単一故障点の存在，リソースの十分な活用が困難等の欠点がある．本稿では，ファイル細分化とノード集合化を用い，単一故障点を無くし，巨大ファイルの効率的な配布を実現するP2Pプロトコル FONet (Fragmet pool Organizing Network)[1]を提案する．

本稿では，3節でFONetの概念と基本動作を示し，4節

† 宇都宮大学工学部情報工学科
Department of Information Science, Faculty of Engineering, Utsunomiya University

ではシミュレータを用いてFONet, CS, Gnutella型P2Pの比較と, FONetの基礎評価を示す.

2. 設計思想

P2PにおいてLinuxディストリビューションのような巨大なファイルを配布する場合, アクセスの分散とファイル送信元のノードの安定化が必要となる.

既存のP2Pでは, アクセス分散の手法として, (1) 受信したファイルを再配布対象とするもの, (2) 受信中のファイル (キャッシュ) を再配布対象とするもの, (3) ファイルを細分化するもの, がある.

(1) では, 複製されたファイルを再配布対象とする事で, 複製された数が増加すればする程アクセスの分散を図る事ができる. しかし, ファイルサイズが大きい場合, 複製が困難でありアクセスの分散に有効に働かない.

(2) では, ファイルを中継したノードが再配布可能となるキャッシュを保持するため, (1) と比較し生成されたキャッシュの分だけアクセスを分散させる事が期待できる. しかし, PureP2Pでは, 転送経路上のノードが常時存在することを保証できないため, 転送時間の長い巨大なファイルでは, 転送途中で失敗する可能性が高くなる. これに伴い, 転送中に作られていたキャッシュが未完成なままとなり, 期待した程の効果が得られない.

(3) は, 細分化したファイルを分散させることで, アクセスの分散が実現できる. 細分化したファイルは, 元のファイルと比較し1つ辺りのサイズが小さくなる. したがって, 元が巨大なファイルでも細分化したファイルの複製は容易であり, 十分なアクセスの分散を実現できる. また, 複製を数多く生成する事ができるため, 各細分化ファイルを所持するノードに冗長性を持たせる事ができ, この冗長性により送信元ノードの安定化を図る事も可能となる. しかし細分化したファイルが拡散すると, 収集効率が低下することになる. 既存のファイルの細分化を導入したP2Pでは, ファイルの拡散を防ぐためにインデックスサーバを用いるNapster[2]の様なネットワーク構成のHybridP2Pを利用しているが, 単一故障点が存在する.

これより, 巨大なファイルの配布の要件を考える.

- ファイルの細分化と, 細分化したファイルの分散配置
- ファイルの拡散の防御と, 単一故障点の除去

FONetではこれら2つの事項を, インデックスサーバを用いずにノード間のみで情報をやりとりするGnutella[3]と同じネットワーク形態のPureP2Pの採用と, ファイル細分化, ノード集合化概念を導入することで解決を図る.

3. FONet

3.1 概　　要

FONetでは, 負荷分散に対応し単一故障点を無くすために, PureP2Pを採用する. ファイルを断片と呼ぶ単位に細分化し, ネットワーク上に分散配置する. これより, 大規模システムに対応できるアクセスの分散を図る. FONetでは, 接続リストを作成し, このリストに他のノードへの接続情報を格納し, 接続情報を元に通信を行う. ただし, ノード数の増加に対して収集効率を維持するため, 上記接続リストを操作し, 論理的な集合を形成する.

ファイルを細分化して扱う方法では, 目的ファイルの断片を所持するノードに対し効率的に要求メッセージを伝えることが, 転送効率向上の鍵となる. 要求メッセージを効率的に伝える技術として分散ハッシュ表 (DHT) を用いる方法がある. しかし, 大規模P2Pネットワークにおいて家庭用計算機が頻繁に脱退参入を行う状況を考えると, DHTの維持管理によるコストの増加により, ファイルの配布効率が低下しかねない. そこで我々は, 任意の指標に基づくファイルの分類としてカテゴリを設定し, カテゴリに属すファイル断片を多く所持するノードの集合を考えた. カテゴリは, 例えばLinuxディストリビューションの種類毎に設ける. この集合は, 当該カテゴリのファイルを所持する確率の高い仮想的なノードを表す. この集合をFragment Pool (FP) と呼ぶ. システム中には, カテゴリに対応して複数のFPが論理的に形成される. 各ノードは, 当該カテゴリのファイル断片の所持状況により各FPに所属する度合を決定する. ノードは, ファイルの収集を行うとき, FPの所属ノードに対し要求メッセージを送ることで効率良く目的ファイルを収集できるようになる. 同時に, FP内では積極的に断片の複製が生成されることが期待でき, アクセスの分散と, 冗長性により断片発信元のノードの安定化が実現できる.

通信路の負荷分散の要件を満たすためにPureP2Pとし, さらに, 上記のようにシステム中に適切にFPを構成することで効率の良いファイル収集を行い, ノード数の増加に対する高い拡張性を持つシステムFONetを提案する.

3.2 Fragment Pool実現方法

FONetの成否は, FPによるファイル収集効率の向上にかかっている. すなわち, 適切にFPを効率的に構成する方法, および目的のFPを効率良く検索する方法が求められる. 同時にこれらの方法は, PureP2Pの特性を活かし, PureP2Pの利点を損なわないようにしなければならない. 我々はこれを解決するために, FPパラメータ, 距離, 接続リスト構築法を導入した.

3.2.1 FPパラメータ

我々は, PureP2Pが接続リストを用いて通信を行う事に注目し, このリストを用いてFPを実現するために, FPパラメータ (FPP) を導入した. FPPは, カテゴリ数を次元とするベクトルで表現し, 各成分をカテゴリ毎の断片の所持率とする. 即ち, カテゴリ数を c, ノードが所持するカテゴリ i に属する断片数 N_i とするとき, 式 (1) のように表される.

$$FPP = (x_1, x_2, x_3 \cdots x_c) \quad (1)$$

$$x_i = N_i / \sum_k N_k \quad (2)$$

FONetでは, FPPを接続リストを操作するための最も基礎的な指標として用いる. なお議論の簡便化のため, 以降の議論では各ノードは自FPPの最大要素のFPに属するものとする.

3.2.2 距　　離

通常のPureP2Pでは, ノード間で接続情報を交換し,

接続リストを更新している．FONetでは各ノードは，接続リストを更新する際に互いのFPPを交換し，距離を算出する．距離とは，後述する距離関数により二つのノードのFPPを元に計算される値であり，この度合により接続リストへの情報の追加の可否を決定する．なお，算出された距離の値が小さい2つのノードは近いと表現し，距離の値が大きい2つのノードは遠いと表現する．

3.2.3 接続リスト構築法

接続リストの更新は，隣接ノード間で互いの接続リストを交換することにより行う．接続リスト用に一定の領域を割り当てておき，空きがあれば情報交換の結果得られたノードを追加する．FONetでは，接続リスト情報が割り当てられた以上の容量になった場合の取捨選択の方法(接続リスト構築法)を工夫することで前述のFPを実現する．

本稿では接続リスト構築法として，(1)最遠ノード削除，(2)FPP成分同調削除，を定義する．(1)最遠ノード削除は，上述の距離をもとに自ノードから遠い順に接続リストから削除して行く方法である．(2)FPP成分同調削除は，自FPPの各要素値の比と，接続リスト中のノードの所属FP毎の数の比が同じになるように調整する方法である．これらの接続リスト構築法による性能差を4.2.4で評価する．

FPは上記のように各ノードが持つ接続リストにより間接的に表す．ノード間の接続状況は，この接続リストで定義されるため，結局，特定のカテゴリの断片を多く含むノードが同一FPに属するとして互いに密な接続関係を作る．したがって，ノード間での接続情報の交換が進むにつれ，断片の所持確率の大きいノードが凝集する．即ち，時間の経過とともにFPの凝集度が高まり，ファイルの収集効率が高まる．所持率の高いノードが凝集している部分をFPの中心部とし，所持率の低いノード，新規に参加したノードが凝集している部分を外辺部と呼ぶ．

3.3 距離関数

FONetでは，距離を用いてFPへの参加の度合を決定する．この距離を計算する関数として距離関数を定義する．ここでは，所属FPP要素差分距離，全FPP要素差分距離，ユークリッド距離，重み付け全FPP要素差分距離，重み付けユークリッド距離を示す．ノードA，BのFPPを$FPP_A = (a_1, a_2 \cdots a_n)$，$FPP_B = (b_1, b_2 \cdots b_n)$とする．

所属FPP要素差分距離では，ノードAがFPiに所属しているとし，式(3)の様にノードAとノードBのFPPのi番目の差を距離としている．計算量も少なく，断片の所持の大まかな特徴を算出できるものと期待できる．

$$\text{affiliation distance} = |a_i - b_i| \tag{3}$$

全FPP要素差分距離では，式(4)の様に全要素の差分をとり，この差分の絶対値の総和を距離とする．ユークリッド距離では，FPPを位置ベクトルとみなし，式(5)の様にFPP間のユークリッド距離を求める．両距離関数は，所属FP差分距離とは異なりFPP成分全体の比較を行う．これらの距離関数では，計算量は多少増加するが単一のFPだけではなくFPP全体の特徴を抽出することで異FP間での連結を柔軟にすることが期待できる．

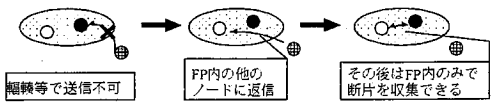

図1 断片集積

$$\text{difference distance} = \sum_k |a_k - b_k| \tag{4}$$

$$\text{euclid distance} = \sqrt{\sum_k (a_k - b_k)^2} \tag{5}$$

また，式(4)，(5)を改変した距離関数に，ノードAが所属するFPに該当するFPP成分の差分に対し重み付けを行うことで所属するFP内での接続の強化を図る重み付け全要素差分距離と，重み付けユークリッド距離を定義する．

なお，これら距離関数による性能差を4.2.3で評価する．

3.4 FPの検索

前記のように，各ノードは自身が保持する断片の状況によりFPを形成する．目的のファイルが同じFPに属していれば，前記の方法で高効率な収集ができるが，異なるFPに属する場合には，目的FPの情報(参加ノードなど)を効率的に得る方法が必要となる．このためFONetでは，以下の操作により目的のFPの検索を行なう．

ノードは，自己のFPPの要素のうち，収集ファイルが属するカテゴリの要素を一時的に高め，収集ファイルが属するカテゴリに対応したFPに所属しようとする．例として，ノードがカテゴリiに属するファイルの収集を試るとする．このとき，ノードはFPPに対し，要素x_iを高めるための重みwをつける．$FPP' = (x'_1, x'_2, x'_3 \cdots x'_c)$の$j$番目の要素について，式(6)のようになる．

$$x'_j = \begin{cases} \dfrac{wN_j}{wN_i + \sum_{k \neq i} N_k} (j = i) \\ \dfrac{N_j}{wN_i + \sum_{k \neq i} N_k} (j \neq i) \end{cases} \tag{6}$$

3.2.3に示したようにFPはFPPをもとにに形成されるため，この操作により目的のFPの接続情報が多く集まることになる．これは，FPPの操作により目的のFPの外辺部あるいは外部にあるノードが一時的に中心部に向かって移動することを意味する．これに伴い，目的の断片の収集が進むと，その分だけ当該カテゴリのFPPが増加する．その結果，ノードは更にFPの中心部に向かい，収集効率が向上する．

3.5 断片の集積

細分化されたファイルが分散配置されている状況では，必ずしも全てのファイル断片がFP内部のノードに保持され，効率の良いファイル収集が行われるとは限らない．また，たとえFPが正しく検索できても，FP外辺部にある断片が収集できないために，結果として収集効率が著しく悪化する可能性や，ファイル収集中は要求しているノードに対し集中的な通信が発生するために，CSとは逆に受信側で輻輳が発生する可能性もある．このため我々は断片集積と呼ぶ機能を導入し解決を図る．

図1に断片集積の概念を示す．図中●は断片を要求しているノードを，網かけ○は断片送信元ノード，○は断片

を集積するノードを表す．受信側で輻輳が発生し，要求ノードへの転送が長時間ブロックされる状況が起きた場合，送信元ノードが図1のように要求元ノード自身ではなく，接続リスト中の返信先ノードと同じFPに属する他のノードに送ることを許す．断片を受信したノードはそのまま当該断片を所持する．要求元ノードはファイル収集が完了するまで要求メッセージを送信しファイルの検索を続けるが，目的の断片が論理的に近い位置に移動しているため収集効率が高まることが期待できる．このように断片を集積することにより，FP内のファイル存在確率をより高め，ファイル収集効率を向上させる．

3.6 ルーティング

FONetでは，FP中心部のノードがより多くの断片を所持している．各ノードは接続リストの更新の際に隣接ノード間でFPP情報の交換も行う．この情報を利用し，FP内部での要求メッセージの転送方法(ルーティング)を工夫することで更に断片の収集効率の向上が期待できると考えた．本稿では，以下に示すようにランダム転送，押し上げ転送，押し下げ転送，引き上げ転送，引き下げ転送の4種のルーティング方法を考え，4.2.2節で評価する．

ランダム転送とは，メッセージの転送先のノードを接続リスト中の対象となるFPに属するものからランダムに選択する方法である．

図2　押し上げ転送

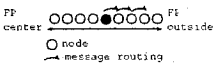

図3　押し下げ転送

図2に押し上げ転送の概念図を，図3に押し下げ転送の概念図を示す．以降の図では，黒円は要求メッセージを送信するノードを表す．白円は，黒円のノードが欲するファイルのカテゴリに対応したFPに所属するノードを表し，左側のノード程FPの中心に位置する．押し上げ転送とは，要求メッセージを常にFPの中心部方向の最も近いノードに対し転送する方法である．中央部に要求メッセージが集中し，断片を発見しやすくなるが，アクセスの集中が予想される．押し下げ転送とは，要求メッセージを常にFPの外辺部方向の最も近いノードに対し転送する方法である．外辺部に送信するため，断片を発見しにくくなるが，しかしその分アクセスの分散が期待できる．

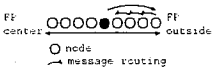

図4　引き上げ転送

図5　引き下げ転送

図4に引き上げ転送の概念図を，図5に引き下げ転送の概念図を示す．引き上げ転送では，最初要求メッセージを接続リスト中の最もFP外辺部に位置するノードに送信し，その後は押し上げ転送と同様に転送する方法である．これより，要求メッセージの転送回数は増えるが，押し上げ転送よりもアクセスの分散が期待できる．引き下げ転送では，最初要求メッセージを接続リスト中の最もFP中心部に位置するノードに送信し，その後は押し下げ転送と同様に転送する方法である．これより，アクセスが集中するが，断片の発見確率の増加を期待できる．

4. 評　　価

提案したFONetの有効性を検証するため，ノード数，FPの内外への要求メッセージの送信比率，距離関数，接続リスト構築法，ルーティング方式による平均ファイル収集時間の変化を比較する．平均ファイル収集時間 (average file gathering time) とは，ファイルの収集を開始してから終了するまでにかかる時間で，断片の検索と転送の時間の総和となる．評価のためにFONet及びCS，Gnutellaを模擬するソフトウェアシミュレータを作成し，平均ファイル収集時間を測定した．

4.1 評価方法

FONetシミュレータでは，以下の動作を導入する．

- ノードは，全ファイルのカテゴリ，名前，分割数を既知
- あらかじめ断片を無作為に選んだノードに分配
- 最初に各ノードに無作為に数ノードの接続情報を付与
- 全ノードは接続情報を交換し，接続リストを更新．更新後，0.5秒待機した後に再度リストを更新
- ファイル収集ノードはFP内外に一定比率で要求メッセージを送信．FPがファイル収集に有効ならば，この比率は収集効率に影響すると予測

なお，FP内外への要求メッセージ送信比率を変化させた実験を4.2.1で行う．

各ノードには識別子として，自然数が割り当てられる．観測ノードとして識別子が1から100の100ノードを選び，これらの動作を観測する．これらの観測ノードは断片が無作為に配布されており，所属するFPが無作為となる．観測ノードは，シミュレーションの開始から0〜10秒の中で無作為に決めた時間だけ経過した後にファイルの収集を開始，この収集時間を測定する．これらの収集時間の平均を平均時間 (average time) と呼ぶ．その他ノードも観測ノードと同様にファイルの収集を開始する．また，各シミュレータでは断片の転送の際に，0.01秒毎に接続状況を確認し，通信速度を計算する．このため，計算量が多くなっており，ノード数が増加した場合に実験が困難になる．このファイル転送の部分を簡略化し，常に最悪の通信速度とした最悪時間 (worst time) を用いてノード数が増加したときの実験を行う．

CSシミュレータでは，以下の動作を導入する．

- 高速通信路(クライアントの10倍)を持つ，常に安定したサーバが存在
- 全クライアントはサーバの所在，所持ファイルを既知

FONetシミュレータと同様に，クライアントの平均ファイル収集時間を測定する．

Gnutellaシミュレータでは，以下の動作を導入する．

- 基本的な動作はFONetシミュレータと同じ
- FPを作らず，ファイルの細分化を行わない
- 実際のGnutellaと異なり，要求メッセージの転送の際にこれを複製，複数送信する動作は導入しない
- 転送中のファイルを再配布する

表1　シミュレータのパラメータ

FP(カテゴリ)数	3
ファイル数	18 (各カテゴリ毎に 6)
最大メッセージ転送回数	7
1発呼メッセージ数	1
バンド幅	2048[Khyte/s]
ファイルサイズ	512[Mbyte] (断片は 1024 [Khyte])
断片数	512
ノード参入脱退確率	0.005秒毎に0.001

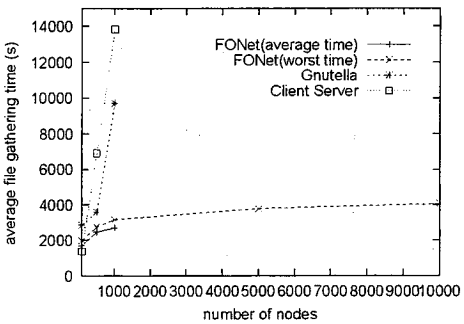

図6　FONetとCS, Gnutellaにおける平均ファイル収集時間

FONetシミュレータと同様に，観測ノードの平均ファイル収集時間を測定する．

各シミュレータの共通事項として，ノード (サーバを除く) は0.005秒毎にノード参入脱退確率によりオンラインのノードはオフラインとなり，オフラインのノードはオンラインに復帰する．これにより，一般のネットワークにおける状況を再現する．また同時に，オンライン上のファイルを収集していないノードは一定確率でランダムに選定したファイルの収集を開始する．いずれのシミュレータにおいてもノードはファイルを収集している最中はそのファイルを収集し終るまで新たなファイルの収集を開始しない．また，ファイルを収集しているノードがオフラインに移行した場合，その収集活動を中止する．オンラインに復帰した際に以前収集中のファイルがある場合，そのファイルの収集活動を行う．

主なシミュレーションパラメータを**表1**に示す．

4.2 実験結果

以下に各実験結果を示す．なお，4.2, 4.2.1におけるFONetシミュレータでは，ルーティングはランダム転送，接続リスト構築法は最遠ノード削除，距離関数は所属FP-P要素差分距離を採用している．また，4.2.2, 4.2.3, 4.2.4では，ノード数1000, FP内外への送信比率を5:1とする．

図6にFONet, CS, Gnutellaシミュレータにおいて各々のノード数を100, 500, 1000, 5000, 10000(5000と10000ノードはFONetの最悪時間における実験結果)と変化させた時の平均ファイル収集時間を示す．グラフでは，縦軸が平均ファイル収集時間を表し，横軸はノード数を表す．CSとGnutellaについて注目をすると，どちらもノード数の増加に比例して平均ファイル収集時間が増加している．これは，CSでは，サーバへのアクセスの集中

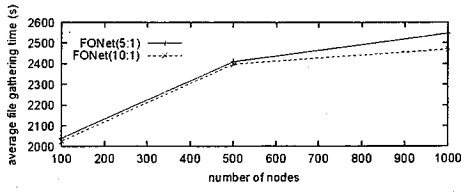

図7　FONetにおけるFP内外への送信比率の変化

表2　ルーティングによる比較

ルーティング方式	平均ファイル収集時間 (s)
ランダム転送	2757.73
押し上げ転送	3400.02
押し下げ転送	4234.31
引き上げ転送	2866.66
引き下げ転送	4409.50

により効率が落るためであり，Gnutellaでは，ノード数が増加したためファイルを所持するノードを発見することが困難となったためと考えられる．FONetにおいては，ノード数が増加した時，CS, Gnutellaと比較し平均ファイル収集時間の増加は少ない．これは，ファイルの細分化によるアクセスの分散と，FPの構築によるノード間での連携により，効率の低下を防げたためと考えられる．また，FONetでの平均時間と最悪時間について差があまり無かったことについて考える．最悪時間ではFONetにおいて，ノードが通信路を常時全て使用している仮定でシミュレーションを行っている．つまり，平均時間におけるシミュレーションでも通信路を占有する割合が多く，結果として，最悪時間と似通った状況になったためと考えられる．

4.2.1 FP内外への要求メッセージ送信比率での評価

図7にFONetにおけるFP内外への要求メッセージ送信比率を(内:外) =(5:1), (10:1)と変化させた時の平均ファイル収集時間を示す．グラフより，(5:1)の送信比率よりも，(10:1)の送信比率のシミュレーションの方が平均ファイル収集時間を短縮することがわかる．これにより，FPを利用することで断片を収集する効率が向上することを示せ，FPの有効性を示せる．

4.2.2 ルーティングの評価

表2では，各ルーティング方式について，それぞれ距離関数，接続リスト構築法を変化させ，その平均をとったものである．この表より，ランダム転送が最も良く，続いて引き上げ転送，押し上げ転送となり，押し下げ転送，引き下げ転送は平均収集時間が長くなった．各方式について考察すると，引き上げ転送では，期待した通りにアクセスの分散と，高確率での断片の発見が実現できたためと考えられる．押し上げ転送では，高確率な断片の発見を実現できたが，アクセスの分散において引き上げ転送に劣るため，その分結果が悪くなったと考えられる．押し下げ転送では，FP外辺部にメッセージを送信することから，目標の断片の発見確率が低く，結果が悪くなったと考えらる．引き下げ転送では，仮にFP中心部で目標の断片を所持するノードを発見しても，他のノードも同様に要求メッセージを送信するため，すぐにそのノードの接続

表3　距離関数による比較

距離関数	平均ファイル収集時間(s)
所属FPP要素差分距離	3662.04
全FPP要素差分距離	3457.59
ユークリッド距離	3503.21
重み付け全FPP要素差分距離	3526.85
重み付けユークリッド距離	3518.54

表4　接続リスト構築法による比較

接続リスト構築法	平均ファイル収集時間(s)
最遠ノード削除	3470.12
FPP成分同調削除	3597.17

数が限界に達する．したがって，実質押し下げ転送と変わらず，結果が悪くなったと考えられる．

4.2.3　距離関数の評価

表3では，各距離関数について，それぞれルーティング，接続リスト構築法を変化させ，その平均をとったものである．表より，所属FP要素差分距離が最も悪く，全FPP要素差分距離が最も良いものとなっている．ユークリッド距離，重み付け全要素差分距離，重み付けユークリッド距離ではほとんど差が無い．

4.2.4　接続リスト構築法の評価

表4では，各接続リスト構築法について，それぞれルーティング，距離関数を変化させ，平均をとったものである．表より，最遠ノード削除が若干良い結果となっている．

なお，これらルーティング，距離関数，接続リスト構築法の中で最も効率が良かった組合せは(ランダム転送，全FPP要素差分距離，FPP成分同調削除)の組合せであり，最も悪かった組合せは，(引き下げ転送，所属FPP要素差分距離，FPP成分同調削除)となることがわかった．

5. 関連研究

関連研究として，ファイルの検索効率を高めるために分散ハッシュ表(DHT)を利用したP2PであるChord[4]，CAN[5]と，FONetと同じくファイルの細分化によりファイルの取得を速くしようとするBittorrent[6]，eDonkey[7]について，巨大なファイルの配布の観点から比較を行う．

DHTを利用したP2PのChord，CANは，ファイル検索の高速化を目的とする．よって，これらのP2Pで巨大なファイルの配布を行う場合，別途ファイル配布元のノードの安定化，アクセスの分散を導入する必要があり，現在のものでは巨大なファイルの配布に向いていない．

高速なファイル配布を目的とするP2PのBittorrent，eDonkeyはファイル細分化の概念を導入している．Bittorrentでは，構造上ファイルの公開元のノードであるSeederの存在が欠かせない．Seederがネットワークから脱退した場合，ファイルを転送することが不可能となる．つまり，Seederが単一故障点となる．また，Bittorrentでは，ダウンロード中のノードを利用して高速化を行っている．しかし，ネットワーク上にはダウンロードに失敗したことで細分化したファイル(bit)をいくつか持っているノードが存在することが予想でき，これらのノードからもbitを収集することができる．Bittorrentでは，構造上これらのノードからbitを収集することは困難である．つまり，リソースを無駄にしていることになる．

eDonkeyは，分散ハッシュ表の概念とファイルの細分化の概念を組み合わせることで，ファイル配布の高速化を実現するP2Pである．eDonkeyでは，DHTにより最適化されたOvernetサーバを利用したHybridP2P型のネットワーク構造となっている．したがって欠点として，インデックスサーバを用意するコストがかかる，インデックスサーバが単一故障点となる，インデックスサーバにアクセスが集中するといった欠点を持つ．

FONetでは，これらのP2Pと比較し，特定のノードに依存しないため単一故障点を持たず，その上で巨大なファイルの配布の高速化を実現できている．また，ネットワーク上に存在するいずれのノードからも断片を収集可能であり，複製された断片を無駄にする事がない．

6. おわりに

本稿では，PureP2Pにおける巨大ファイルの配布の効率化手法としてFONetを提案し，その構築方法，基本的な動作を示した．また，シミュレーションによりCS，Gnutellaと比較し，FONetの有効性を示した．更には，FONetの基礎評価として，FP内外への要求メッセージの送信比率による変化と，接続リスト構築法，距離関数，ルーティングの評価を示した．その上で他のP2Pと比較し，FONetは，従来の技術よりも巨大なファイルの配布を対象としたP2P手法としての優位性を確保できている事を示した．

今後，FONetの更なる最適化手法を追加しシミュレータで評価する．最適化の方法として，FPの階層化，断片の自動複製，断片の自動削除などを検討する予定である．

謝辞　本研究は，一部日本学術振興会科学研究費補助金(基礎研究(B)14380135，同(C)16500023，若手研究14780186)の援助による．

参考文献

1) 湯澤 孝有，服部 晃和，横田 隆史，大津 金光，馬場 敬信，“ファイル細分とノード集合化概念を用いたP2Pファイル転送手法の提案，” 先進的計算基盤システムシンポジウムSACSIS2004論文集，pp131-132，2004年
2) Napster.com. http://www.napster.com/
3) Welcom to Gnutella. http://welcome.to/gnutella
4) Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications,” ACM SIGCOMM 2001, SanDeigo, CA, August 2001, pp149-160.
5) Sylvia Ratnassamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, “A scalable content-addressable network,” In Proceedings of ACM SIGCOMM, San Diego, CA(August2001)
6) The Official BitTorrent Home Page. http://bitconjurer.org/BitTorrent/
7) eDonkey2000-Overnet. http://www.edonkey2000.com/