DEWS2004 I-12-05

階層型Bloom Filterを用いた分散ファイル管理

三橋　孝平† 　森本　亮† 　三浦　孝夫†

† 法政大学 工学部電気電子工学科　〒 184–8584 東京都小金井市梶野町 3-7-2
E-mail: †{c9943120,c9943128,miurat}@k.hosei.ac.jp

あらまし　本稿では、インターネットを介した Peer-to-Peer 環境におけるファイル探索のための新しい機構を提案し、これにより位置透過な方式でのファイル探索が行えることを述べる。その実現のため、各サイトのファイル構造に対して**階層型** *Bloom Filter* を用いた手法を導入し、これにより部分構造を指定したファイル探索質問を可能にする。我々は本提案手法を JXTA を用いて試作することでサーバなしマルチキャスト環境での通信を実現し、また、実験によりその有効性を実証する。

キーワード　ファイル探索、P2P 分散システム、JXTA、 ブルームフィルタ

Looking up Files in Peer-to-Peer Using Hierarchical Bloom Filters

Kohei MITSUHASHI†, Ryo MORIMOTO†, and Takao MIURA†

† Dept.of Elect.& Elect. Engr., HOSEI University　KajinoCho 3-7-2, Koganei, Tokyo, 184–8584 Japan
E-mail: †{c9943120,c9943128,miurat}@k.hosei.ac.jp

Abstract　In this investigation, we propose a new mechanism for looking up files in Peer-to-Peer over internet environment so that we can look up files in a location-transparent manner. To do that, we introduce *hierarchical Bloom filters* to local file structures which allow us to query full/part of the structures. We show some experimental system with major features of JXTA where we put attentions on serverless and multi-casting communication, and we examine empirical results on the system.

Key words　File Look-up, P2P Distributed System, JXTA, Bloom Filter

1. 前　書　き

近年インターネット環境の普及により、広範囲なデータの管理に対する必要性がますます高まっている。しかし、精密なシステムが大規模環境で使用されるほど、性能および整合性を保つことは困難であり、協調的で自律性を保つ新たな管理機構の必要性が求められている。

例えば、東京やニューヨーク、フランクフルトなどに支社を持つ会社の資材部を考える。そこでは、インターネットによる通信が絶え間なく行われており、定期的に会議を行う。このような状況下では、各サイトおよび各ファイルは自律的にネットワークへの参加・離脱をするべきであり、ファイルは利用者がその位置を意識しなくてもよい様な位置透過な方式で管理されるべきである。コンピュータ間での通信については効率的な管理をすべきであり、パケットにおけるブロードキャストやループは発生させるべきではない。さらに、外部に対して厳しい機密管理を行いながら、反対に資材部の内部では各 サイト は同等に機能すべきである。各サイトに置かれたファイルは毎日更新されるかもしれず、そして、それらのファイルがいつサイト間を移動するとも限らないため複製やキャッシュを用いた手法は適さない。

本稿では Peer-to-Peer（P2P）ネットワークにおけるファイル探索のための新たな方法を提案し、これにより拡張性が高く位置透過な（マイグレーションを効率よく行える）ファイル管理機構を確立できることを述べる。とくに、ファイル構造を用いたファイル探索を効率よく行うため、各 サイト のファイル構造に対して**階層的** *Bloom Filter* を用いた技法を導入する。また、ここでは、JXTA [5] による**サーバなし、マルチキャスト環境**での通信網を想定する。このようにして、位置透過な方式において、少ない情報量での自律的なファイル管理を実現する。

2 章では P2P 環境下での情報管理について論じ、3 章では Bloom Filter を使用したファイル探索について論じる。また、4 章では JXTA の概要と本提案システムの構成について論じる。5 章では実験結果を示し、6 章で結びとする。

2. P2P 環境下での情報管理

この章では、なぜ P2P 下での情報管理が必要なのか、そして、どのような特徴を実装すべきなのか、について論じる。

P2P 環境では主に**データ共有**、**分散計算**、**自律システム**の 3 つの応用分野を狙いとしている。**データ共有**については、とくに質問評価技法、拡張性の問題、効率的なデータ構造など、データベースにおける側面を考える。**分散計算**については、多くのコンピュータの資源へ負荷を分散させるための並行処理に興味がある。そこでは非集中化、非階層的、動的な結合が注目すべき点である。**自律システム**については、他のサイトのいかなる通知もなしで、いつでもネットワークへの参加・離脱が行えることを意味する。ここでの主な関心はシステムの**耐久性**である。

P2P 環境におけるデータ検索では**どの粒度で**データを管理するかが重要なポイントとなる。仮に**データレコード**単位での管理をする場合、レコードごとに持つ**キー**を 1 つの場所にマップしなければならない。その典型的な手法が *Distributed Hash Tabl*（DHT）[1] である。そこでは、データのキー値と IP による位置指定情報との対応付けを行い、そこからレコードを見つけ出す。これは明らかに、自律的であるという要件に適さない。なぜなら我々が想定する状況下では自身のレコードを保持し、それらのマイグレーションを行うことを望むからである。この問題はデータを管理する粒度が小さ過ぎることによるものと思われる。そこで、管理・格納・検索の観点から**ファイル**単位での管理を行う。また、ファイルをその**名前**によって管理することから、その環境下での**ファイル探索**機構について論じる必要がある。

伝統的にいくつかのタイプのファイル探索機構が P2P システムの中で提案されている。最も単純な手法が**サーバ方式**である。そこでは、中央のコンピュータがシステムの機能において非常に重要な役割を果たし、そして利用者が ⟨ ファイル名、サーバ ⟩ による探索を行う。例えば、*Napster* はファイルに索引付けを行うことで、効率的なアクセスを提供している。これでは明らかに、P2P 処理における要件を満たすことができない。2 番目の手法はサーバ同士の**階層方式**である。そこでは、要求メッセージは自動的に適切なサーバに送られ、そして利用者はそれらがどのサーバを経由するかを気にしない。典型的な例は *Domain Name System*（ DNS ）である。このアプローチでは多くの要求がより上位のサーバに行く傾向がある。しかし、複製とキャッシュを利用することでこの問題を解消している。3 番目の手法はすべてのサーバが管理用のサイトなしに接続されている**対称方式**である。そこでは、事実上それぞれの質問がすべてのサーバにブロードキャストされる。機構全体のスループットを減らすためのその通信が莫大な量となってしまう。例えば、*Gnutella* は多くの通信パケットを使い、そして **パケットループ**を引き起こす。そこでは通常、全体の統制のために**管理サイト** (super peer) を導入するが、それは P2P の概念を否定してしまう。また、素早くファイル探索を行うための何らかの機構についても検討するべきであるが、それはほとんどの場合、通信システムとは独立した問題である。最後の手法は *DHT* **に基づく**ファイル名管理であり、そこではすべてのファイルが静的にいくつかのサイトに対応している。多くのファイルを読み出しのみであると想定してはいるが、この手法では、各サイト間でのファイルの移動を行いたいという目的に適さない。

ここで、 ディレクトリ / ファイルによる探索、および情報量の削減、という 2 つの点に注意を払って P2P アプローチをとる。前の章で述べたように、我々は効率的かつ自律的、さらに動的に働くファイル探索機構を必要とする。

そこでは、それぞれのサイトが同じ原則に基づいてファイル探索を行うことを想定する。すなわち、どのサイトも他のサイトに対してあらゆるファイル探索要求を行えるが、同時に他のサイト からのいかなる要求も受け入れて働くべきである。

本試作において、本稿の主題であるファイル探索のための *Bloom Filter* 、そして P2P 通信のための *JXTA* **に基づくパケット管理**という 2 つの基本アイデアについて論じる。後者については、最初の章で述べたように、広範囲なネットワーク間での**中規模**程度の通信量を想定する。

3. Bloom Filter を用いたファイル探索

この章では P2P 環境でどのようにファイル探索を行うべきか論じる。基本のアイデアは分散データベース処理 [3] のために考案された *Bloom Filter* を用いることである。

現在までに、ファイル探索についての、いくつかの調査が実験的な研究により成されている [4]。最も単純な探索では、すべてのファイルを徹底的に調べるため、無駄な情報が大量に生じてしまう。通常、**探索の対象となるファイルを削減する**ための何らかの技術を必要とする。しかし、単純なビットマップ Filter（後に述べる *Bloom Filter* と呼ばれるもの）は、一般的な質問処理には非常に効率的であるが、ディレクトリ構造を扱うにはあまりに単純なものである。圧縮 Filter はそのどれもがサイト間での分散質問のためのものであり、ローカルサイト内で直接用いられるものではない。集約 Filter は各ローカルサイトにあるすべてのファイルを要約することを目的とするが、部分構造を扱うためのものではない。

本稿で想定するファイル管理機構では、すべてのファイルは階層的に管理されている。そして、根から**パス**を指定することによってすべてのノードに到達することが可能である。

［EXAMPLE 1］ 一般的にファイルを管理するための階層構造は Windows、Unix、MacOS、IBM メインフレーム OS などプラットホーム独立な各サイトの中で保持される。以下の例で述べるように、1 章で例に挙げた会社のケースでいくつかのファイルを探索する。

(1) `MaterialDept/PartTimeEmp/Address`
資材部のパートタイム従業員すべての住所録を意味する。

(2) `/Tokyo//Address/`
東京オフィスの従業員すべての住所録を意味する。先頭の `/Tokyo` は”Tokyo”がトップの項目として現れて

いることを記述している。

(3) `/Tokyo/*/Address`

”Tokyo”から 1 レベルをはさんで下にあるすべての住所録を意味する。*はその 1 レベルにとんなファイル名があってもよいというワイルドガードであり、//はそこに何レベルの構造が入ってもよいというワイルドカードである。

□

階層的なディレクトリ構造物に取り組むために、基本的に単一レベルと集約レベルの両方の Filter を持った *Bloom Filter* のアプローチをとる。

Bloom Filter は **ビットベクトル**（ビット列）であり、それは **シグネチャコーディング**によって得られる。すなわち、収集してきたファイル名に対して 1 つ 1 つ**ハッシュ**関数を使ってビット列へと変換し、それらの**論理和**操作を行う。あるディレクトリが持っているすべてのファイルをそのファイル名によってシグネチャコーディングした値を *Local Bloom Filter*（LBF）と呼ぶ。すべてのディレクトリファイルは名前を持ち、そしてシグネチャの一部としてコード化されることに注意が必要である。一方、あるノードについて、*Global Bloom Filter*（ GBF ）とは、そのノードに含まれるすべてのディレクトリの GBF から形成され、それらすべての論理和操作を行う。GBF は再帰的に定義され、葉ノード（ディレクトリを 1 つも含まないノード）では空の GBF となる。

各ノードは LBF と GBF のペアを保持しており、各サイトは最高階層の LBF と GBF を管理する。あるサイトに送られてきた質問については、このペアを調べることで求められたファイルがこのサイトにあるのかどうかを効率よく決定することができる。我々が開発したこの機構を**階層型** *Bloom Filter* と呼ぶ。［EXAMPLE 2］ 図 1 の階層構造に対しての質問 `C//xyz` を例証する。

`C` と `xyz` の シグネチャコーディングした値をそれぞれ `BF(C)` と `BF(xyz)` で表現する。同じく `A` の GBF の値ならば `GBF(A)` というようにを表現していくとする。もし `GBF(A)` が `BF(C)` と `BF(xyz)` 含んでいるなら、このサイトは部分構造を含んでいるかもしれない。そこで、`LBF(A)` が `BF(C)` 含んでいるかどうかたずねる。

これが正しいので、次のステップに進む。`GBF(C)` は `BF(xyz)` を含んでいるが、`LBF(C)` は `BF(xyz)` を含んでいない。そこで今度は `GBF(D)` が `BF(xyz)` 含んでいるかどうか、そして `GBF(E)` が `BF(xyz)` 含んでいるかどうか吟味する。前者のケースでは答え NO であるが、後者のケースでは答えは YES である。YES と答えた E に対して、さらに、 `LBF(E)` が `BF(xyz)` を含んでいるかたずねる。

注目すべき点は、たとえ `LBF(E)` が `BF(xyz)` 含んでいるとしてもそこにファイル `xyz` がないということがあり得ることである。それは **フォールスポジティブ**と呼ばれている。しかし、もしそこにファイルが本当にあれば Bloom Filter は必ず YES という答えを返す。□

4. 試作システム

本稿では、サーバなし、パケットループなし、マルチキャスト環境での通信を少ないオーバーヘッドで可能にする完全 P2P システムを必要とする。そこで、 JXTA を基本的なホスト通信機構として利用する。また、どのように試作システムを実装すべきかを論じる。

4.1 JXTA プロトコル

ここでは簡単に JXTA 技術について述べる [2]。より多くの詳細は [5] に掲載されている。

JXTA 技術の目的は、共通の基本的な機能を提供することにあり、これを用いて様々な応用が構築可能である。とくに、様々な P2P システム間の相互運用性を達成することができる。JXTA の基本概念は、*UUID*, *advertisement*, *peer*, *message*, *pipe*, *peer group* の 6 つから成り立っている。

UUID とは、各サイト内で識別可能な JXTA 用の ID であり、128 ビット長である。これにより、peer，advertisement，service などを特定する。この番号は大域性を有しないため、他の資源を識別する機構と連動して使用されねばならない。

advertisement は XML 構造の文書であり、資源の存在を主張し、名前や詳細を与える。対象となるのは、peer，peer group，pipe および service である。JXTA 技術においては advertisement の基本集合を定めており、XML スキーマを用いて部分型を定義できる。

peer とは通信に要する規約を主張できる単位であり**サイト**と対応すると考えてよい。processor，process，machine，user などを宣言し、この利用に関する規約を有している。ただし、peer はすべての規約を理解する必要は無く、必要に応じて処理が出来ればよい。

message とは、非同期、非信頼、一方向トランスポートのレベルで使用できる。message は、この意味で datagram とみなすことができ、封筒と宛名の付いたメッセージである。封筒には、ヘッダ、内容要約、発信、受信などが記載され、発信・受信は URI で洗い出された論理実体である。通常、これは物理アドレスに変換され処理される。対象となるデータ内容は可変長のバイト列である。

peer group とは仮想的な実体であり、peer group 規約に関する記述を扱うだけの働きをする。通常 peer group は複数の peer

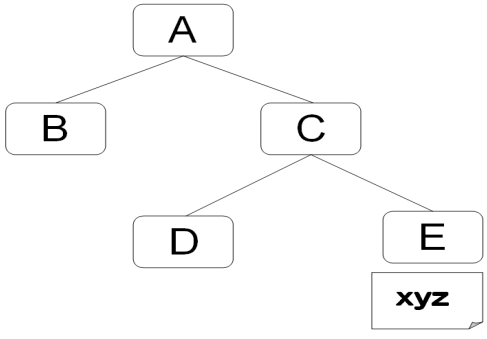

図 1　階層型 Bloom Filter

からなり、共通のサービスを提供・利用する。JXTA 仕様では具体的な記述・規定は何もなく、 group の作成やメンバ制約などもない。また入れ子メンバなどの規定もない。ただし、主たる定義は、peer group をどのようにして発見するか、という規約にある。

pipe は JXTA が提供する基本的な通信経路であり、非同期、一方向 (入出力の区別あり) のメッセージ路である。pipe は peer とは動的に結合し、どの peer といつどの様に結合するかの規定はない。*point-to-point pipe* はちょうど 2 つの peer を結び、送信者からは出力 pipe に、受信者には入力 pipe として扱われる。*propagate pipe* は複数の peer を同時に結合する。一つの出力 pipe として扱われた message はすべての入力 pipe に伝播して送信される。

[EXAMPLE 3] 図 2 で、JXTA による通信機構と制御の流れを述べる。

長方形は peer を意味し、それらは 5 つの peer が 1 つのサービスを共有するような peer group に属している。中央の円はホスト通信のための pipe を意味している。message が `peer0` から入ってくると、それらは propagate pipe で他のすべての peer に伝播される。`peer3` は `peer0` への応答を返す際、`peer0` への point-to-point pipe を用いる。このように `peer0` は他のサイトからの messege を聞くために 2 種類の通信手段を持っている 。□

JXTA 技術とは、実際にはこのような概念を用いて構築された規約の集合である。プラットホーム独立であり、トランスポート独立でもある。たとえば、TCP/IP を使って実現されてもかまわない。ここで扱われる規約は次の 6 つである。

(1) Peer Discovery Protocol
(2) Peer Resolver Protocol
(3) Peer Information Protocol
(4) Peer Membership Protocol
(5) Pipe Binding Protocol
(6) Endpoint Routing Protocol

Peer Discovery Protocol とは、peer がどの様にして他の peer の advertisement を発見するかに関するもので、peer、peer group、advertisement についても同様に働く。標準的に用意されているものは、極めて単純であり、原理的にはすべての advertisement を見つけようとする。

Peer Resolver Protocol とは、peer が peer、peer group、pipe などを散策するための質問を送信・受信することを規定しており、通常は、データファイルなどを利用できるサイトだけで利用される。

Peer Information Protocol によって peer は他の peer の機能や状況を知ることが出来る。

Peer Membership Protocol は requirement の検索、加入、変更などを行うための規約で、認証などの機密管理を扱う。

Pipe Binding Protocol とは、peer が pipe advertisement を pipe に結合するための規約であり、message が実際に pipe で処理される状況を指定する。pipe を待ち行列とみなせば、待ち行列操作に対応する。

Peer Endpoint Protocol とは、peer が peer router に対して message を相手に送信できるかどうかを問うための規約をいう。ここでは NAT、firewall なども含めて考えており、peer router とよばれるゲートウェイの役割を担うかどうかの指定も含まれる。

4.2 システム構成

本提案システムの基本的な構成は、JXTA を用いた P2P 通信機構 、GUI ベースのユーザインターフェイス、そして、階層型 Bloom Filter を用いた分散ファイル管理により成り立っている。前節で述べているように、JXTA はプラットホーム（オペレーティングシステム）独立な P2P 様式におけるマルチキャスト的通信機構を可能にする。我々はその通信機構の上にファイル管理機構を設計し、そして実装する。

前にも述べているように、効率的なファイル探索を達成するために LBF / GBF 機構を導入する。Bloom Filter は、本試作システムでは 1024 ビット長とし、これを 32 ビット整数配列 32 個で表現する。ディレクトリごとに、そこに含まれるすべてのファイル名についてこれを 4 分割する。ファイル名全体をハッシュコード化してどれか 1 つの部分だけを選び、選ばれたものをハッシュコード化して Bloom Filter のシグネチャと**論理和**操作を行う。これを 128 ビットずつ 8 回繰り返し、すべてのファイル名から LBF を生成する。これにより、ファイル名の一部分として出現しやすい文字列への偏りを避けることができる。部分ディレクトリがあれば、その GBF を集めて論理和をとり、当該ディレクトリの GBF とする。また、他のサイトの最高階層の集約 Filter のキャッシュを各 サイト が持つことで探索質問を送る相手を限定することもできる。この場合、キャッシュを保持しマルチキャスト制御を行うことの負荷と、それによって得られる適合性とのトレードオフを調べる必要がある。

図 2 で、構成の骨組みと制御の流れについて述べる。`peer0` が `(1)` のように共通の pipe に要求メッセージを送る。それから要求は同じ peer group のすべての peer に `(2)` のように伝播される。プロセス全体は JXTA のマルチキャスト的通信手段によって管理される。それぞれの peer が要求メッセージを受け取ると、階層型 Bloom Filter の技法を使用して処理が行われる。いずれかの peer 、例えば `peer3` が何かしら要求されたファイルを見つければ、peer は `(4)` のように答えを直接 `peer0` に送る。最終的に要求元の peer である `peer0` は `peer3` にファイル転送を求める。本試作システムでは、FTP を利用してファイルのダウンロードを行う。

図 3 で、`report//data1.els` という要求とそれに対する他の peer からの答え例示する。操作の手順は、はじめに左上のボックスに要求を書き Search ボタンを押す。応答が帰ってくれば、リストの中から欲しいファイルを表示した行を選び、DownLoad ボタンを押す。

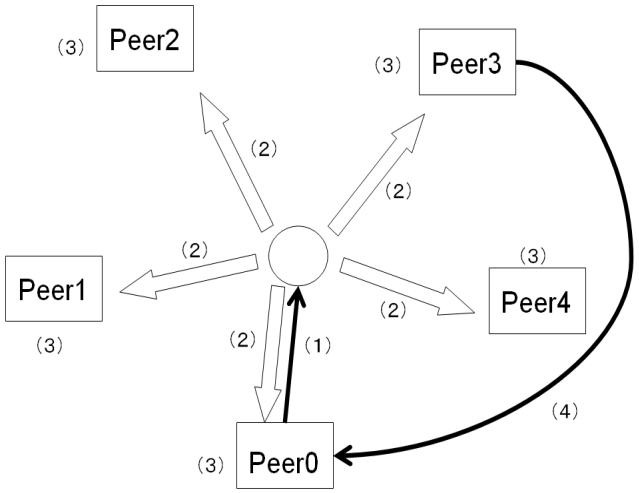

図 2　システム構成

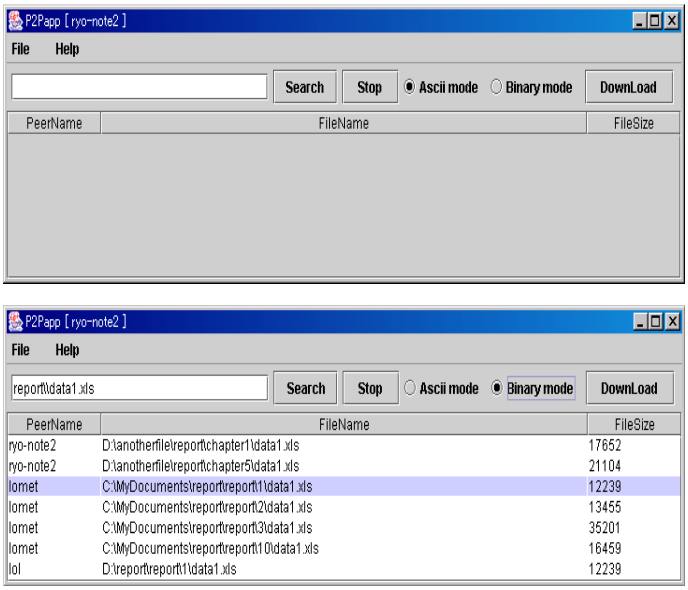

図 3　ユーザインターフェイス

5.　実　　験

本稿では試作システムの設計を行い、3 台の同じ性能のコンピュータに実装する。その環境は Pentium Ⅳ 1.8GHz 、メモリ 256MB 、100MB FastEther LAN 、 FreeBSD 4.6.2 、JAVA SDK 1.3.1 である。また、探索対象は 77845 個のファイル、800MB とし、複製は利用しない。ここに要約した情報を示す（ **ファイル**は **ディレクトリ**を含んでいる）。

	レベル 1	レベル 2	レベル 3	レベル 4	レベル 5
CPU1. ファイル	31	385	4257	6050	1505
(ディレクトリ)	24	211	496	147	1
CPU2. ファイル	6	136	1755	14568	17356
(ディレクトリ)	6	98	630	1404	432
CPU3. ファイル	8	130	1063	2276	9559
(ディレクトリ)	8	66	219	722	801
ファイル合計	45	651	7075	22894	28420
(ディレクトリ合計)	38	375	1345	2273	1234

	レベル 6	レベル 7	レベル 8	合計
CPU1. ファイル	8	0	0	12236
(ディレクトリ)	0	0	0	879
CPU2. ファイル	5538	148	0	39507
(ディレクトリ)	7	0	0	2577
CPU3. ファイル	9953	2750	363	26102
(ディレクトリ)	207	14	0	2037
ファイル合計	15499	2898	363	77845
(ディレクトリ合計)	214	14	0	5493

ここでは、ランダムにレベル 2 とレベル 5 のファイルをそれぞれ 5 個づつ選び、それらへの探索をテストケースとして行う。レベル 2 のファイルについては`/dir/file` の形式で、レベル 5 のファイルについては `dir1/file` ， `/dir2//file` の 2 つの形式（「レベル 5（1）」と「レベル 5（2）」で示す）で、5 回づつ探索を行う。また Bloom Filter の処理時間と、JXTA 通信時間を計測し、それぞれのファイル毎の平均 (BF，JXTA で示す) と標準偏差 (BF(stdev)，JXTA(stdev) で示す) を表およびグラフに示す。

レベル 5（1）	BF	JXTA	BF(stdev)	JXTA (stdev)
ファイル 1	1104.4	214	17.81291666	75.62076434
ファイル 2	204.8	193.2	2.588435821	7.395944835
ファイル 3	311.6	202.2	1.140175425	16.51363073
ファイル 4	200.2	223.4	2.167948339	67.98014416
ファイル 5	301	214.8	3	46.55856527

レベル 5（2）	BF	JXTA	BF(stdev)	JXTA (stdev)
ファイル 1	624.4	260.6	2.792848009	76.34985265
ファイル 2	771	196.2	5.099019514	3.346640106
ファイル 3	744.2	194.4	3.1144823	5.504543578
ファイル 4	770.6	269.4	4.159326869	71.34633838
ファイル 5	832	262.2	46.00543446	109.369557

レベル 2	BF	JXTA	BF(stdev)	JXTA (stdev)
ファイル 1	308.6	313.2	2.701851217	48.03852621
ファイル 2	168.4	291.8	1.341640787	43.39585234
ファイル 3	304.6	340.4	2.073644135	94.43145662
ファイル 4	126.2	352.2	0.447213596	118.9735265
ファイル 5	232.6	303.4	11.78134118	38.97178467

表 1　レベル 2 およびレベル 5 のファイル

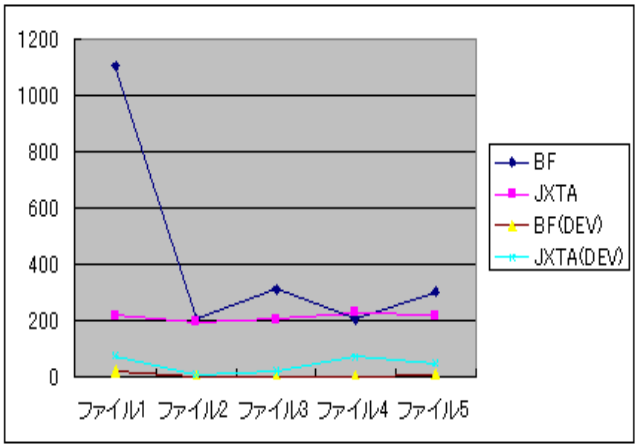

図 4　レベル 5 のファイル（1）

表とグラフから、ファイル探索にかかる BF の値は変化しており、JXTA の値は変化のない状態を保ちながらもその標準偏

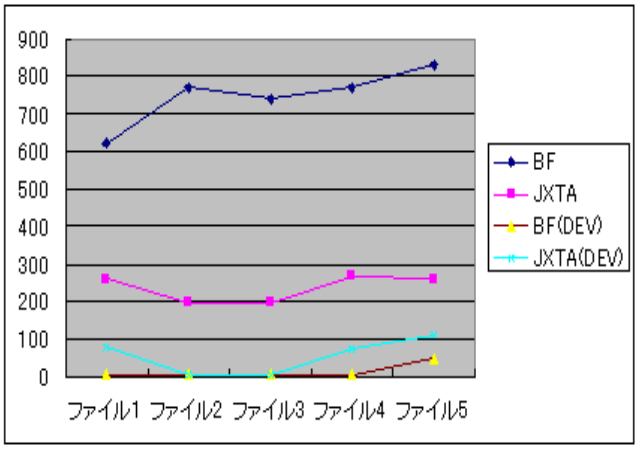

図 5　レベル 5 のファイル（2）

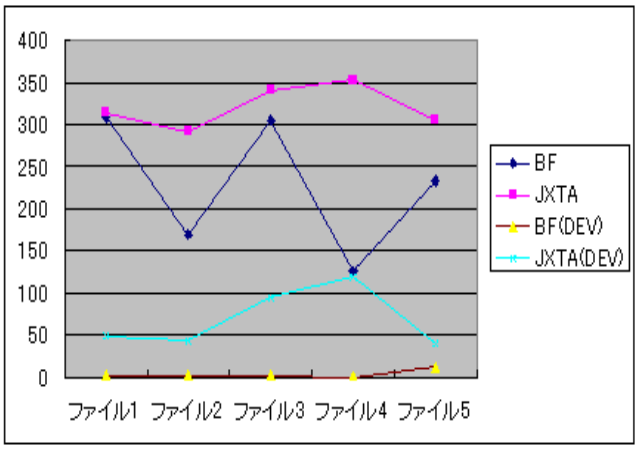

図 6　レベル 2 のファイル

差は変化していることが分かる。BF の値については、いくつかのファイルの探索を行い、そして、その結果がファイルの置かれた場所、キャッシュバッファリングやその他の様々な状態に依存することから、むしろ変化すべきである。しかし、その標準偏差の値は非常に小さいものとなっている。

一方、1 つのファイルについて見た場合、JXTA は小さい値をとることもあるが、それは BF と比較してひどくばらつきがある。これは Pipe Discovery Service の遅れが原因かもしれない。

今回の実験により、どのサイトもネットワークへの参加・離脱が自由に行えることを確認した。また、本稿での主な目的が提案システムの実現可能性を見ることであるから、研究室内において、システムの評価を行った。

6. 結　　び

本稿では、P2P ファイル探索機構を提案し、JXTA 通信パッケージに基づく試作システムを実装した。

また、ファイル探索要求を調べて、処理効率と同様、P2P が持つ有効性の実現可能性を評価した。この実験によって、階層型 Bloom Filter が P2P 下でのファイル探索において重要な役割を果たすことを示した。不幸にも、現在の JXTA の実装では、Pipe Discovery Service のために多くの時間を要してしまうことが頻繁にあり、そのため、リアルタイム処理には適さない。

また、本稿で設計したシステムは我々の研究室のホームページによりフリーウェアとして公開予定である。(http://www.dbl.k.hosei.ac.jp)

謝　　辞

本研究の一部は文部科学省科学研究費補助金 (課題番号 14580392) の支援による.

文　　献

[1] Balakrishnan,H. et al.: Looking Up Data in P2P System, *C.ACM* 46-2, pp.43-48, 2003

[2] Gong, Li: Industry Report – JXTA: A Network Programming Environment, IEEE *INTERNET COMPUTING*, 2001 June

[3] Kossman, D.: The State of the art in Distributed Query Processing, ACM Comp.Survey 32-4, pp.422-469, 2000

[4] Ledlie, J., Serban, L. and Toncheva, D.: Scaling Filename Queries in a Large-Scale Distributed File System, Harvard Univ. TR03-02, 2002

[5] 倉骨彰, 佐野元之訳：JXTA のすべて – P2P Java プログラミング, 日経 BP 社