適応的ピア発見による非構造 Peer-to-Peer システムの効率的な問い合わせ処理に関する研究

山田 太造

博士 (情報学)

総合研究大学院大学
複合科学研究科
情報学専攻

平成 16 年度
(2004)

2005 年 3 月

本論文は総合研究大学院大学複合科学研究科情報学専攻に
博士 (情報学) 授与の要件として提出した博士論文である。

審査委員：

高須 淳宏　(主査)
安達 淳　　国立情報学研究所
相澤 彰子
河野 浩之　　南山大学
丸山 勝巳
山田 茂樹

(主査以外はアルファベット順)

A Study on Efficient Query Processing For Unstructured Peer-to-Peer Systems Based on Adaptive Peer Discovery

Taizo Yamada

DOCTOR OF
PHILOSOPHY

Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)

March, 2005

A dissertation submitted to
the Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Advisory Committee:

Atsuhiro Takasu	(Chair)	
Jun Adachi		National Institute of Informatics
Akiko Aizawa		
Hiroyuki Kawano		Nanzan University
Katsumi Maruyama		
Shigeki Yamada		

(Alphabet order of last name except chair)

内容便覧

blog や Wiki 等に代表されるオープンな協調作業環境下で共有される文書は更新頻度が高く，システムを利用する各ユーザは全文書を必要とするわけではない．これらのシステムのユーザ数およびそこで共有される文書数は増加傾向にあり，今後も注目すべきデータ共有システムの 1 つである．オープンな協調作業環境システムで効率的なデータ共有を行うためにはシステムのスケーラビリティの確保およびデータ更新に対して柔軟に対応可能な仕組みを導入する必要がある．そこで自己組織化に優れる自律分散型システムである Peer-to-Peer（P2P）システムを利用することでこれらの問題を解決する．P2P システムと他の集中システムや分散システムと大きく異なる性質として，各ピアの自律性の高さから各ピアが接続対象を動的にかつ適応的に発見することが容易かつ自然に行うことが可能な仕組みを保持している点である．

本研究ではピア（ノード）の自律性およびシステムの柔軟性や自己組織化に優れる Gnutella や Freenet に代表される非構造 P2P システムを用いる．非構造 P2P システムでは，各ピアはシステム内の全ピアに対して積極的にコミュニケーションを行うのではなく，同様の嗜好であるピアに対して積極的にコミュニケーションを行い，それらのピアを動的かつ適応的に確保することが可能である．また，各ピアの嗜好や行動に変化があったとしても，その変化に柔軟に適応し，新たに積極的にコミュニケーションを行う対象を確保することも可能である．しかしながら，従来の非構造 P2P システムでは，問い合わせ処理の非効率性が指摘されており，P2P システムでのデータ共有を効率的に行うためには問い合わせ処理の性能を向上させる必要がある．本研究ではシステム上の各ピアの情報および動向に着目し問い合わせ処理の効率性を向上させることを目的とする．

P2P システムでの問い合わせ処理の処理効率を向上させるためには応答時間の向上および帯域幅消費量の低減を達成する必要がある．この問題の解決の手段として問い合わせメッセージのルーティングの方法および文書の配置方法の向上を挙げることができる．そこで本研究では，P2P ネットワーク上の各ピアが適応的にそのピアと同じ嗜好であるピアを発見可能である仕組みを提案する．この仕組みを用いると，同じ嗜好であるピアに対して積極的に問い合わせメッセージを送信するルーティング，およびそのようなピアに対して文書の複製を配置する仕組みを提供することが可能になる．

問い合わせメッセージのルーティングの目的はより少ない問い合わせメッセージで問い合わせを終了させることである．ルーティングの性能向上のためにはその指標を与える必

要がある．本研究ではその指標としてピアの “有用度” を導入し，各ピアはこれを利用して動的に P2P ネットワーク上にリンク（長距離リンク）を形成する．この仕組みにより，ネットワーク上の遠くに位置するピアに対しても直接問い合わせメッセージを送信することが可能になる．ピアの “有用度” はそのピアが所有する文書の “有用度” によって決定する．文書の “ 有用度 ” は文書の参照回数および文書の更新からの経過時間によって決定する．ピアの情報を維持管理するための仕組みが必要であり，ピア情報を格納するための分散インデックスである Direct Index（DI）を提案する．システム上の各ピアは DI 内のピア情報を伝播させ，インデックス更新メッセージを受信したピアはその内容に応じてその DI を更新させることで他のピアを動的に発見する．問い合わせ処理を行う場合，各ピアは適応的に長距離リンクを形成し，問い合わせメッセージをルーティングさせる．インデックス更新を行う場合，各ピアはその DI 内の情報を更新し，インデックス更新メッセージを伝播させる．DI を用いたインデックス更新では，文書の追加や更新等の個々の文書の情報の変化はその文書を所有するピア内では即時反映するが，他のピアに対しては敏感に反映させない．インデックス更新メッセージを受け取ったピアはそのメッセージがそのピアの DI 内の情報を変化させない場合，インデックス更新を停止するためである．インデックス更新の頻度を抑えることが可能になるため，インデックス更新のコストを低減することが可能になる．

P2P システムでは各ピアのネットワークへの接続状況は頻繁に変化させる．そのために，発見困難な文書やシステムから喪失してしまう文書が存在してしまう．そこで本研究では文書の複製を行う．文書複製のために，本研究では，複製の数をピアごとに動的に決定し，複製対象となる文書をその所有者であるピアが動的に決定する．各ピアはそのピアが生成できる複製の数をそのピアの “有用度” とその隣接ピアの “有用度” から相対的に決定する．複製する文書の決定は文書の “有用度” とその文書の既存の複製数に従う．“有用度” が高いものほど複製の対象となりやすいが既存の複製の数によってその文書の複製を抑制する方針である．文書複製を行う場合，複製の配置の対象を決定する必要がある．そのために，本研究では，動的にピアグループを形成し，そのピアグループに属するピアに対して複製を配置する手法を提案する．ピアグループの形成の方針はあるピアと関連するピアを発見し，それらのピア集合でグルーピングを行う．ピア間の関連を決定する方法がそれぞれのピアグループ形成の方針で異なる．

提案する分散インデックスおよび文書複製手法を評価するためにシミュレーションを行い，その効果を示し，提案した各手法の特色を示している．その結果，本研究での問い合

わせメッセージのルーティングおよび文書複製の手法は既存の非構造 P2P システムでの手法と比較して，ネットワークへのダイナミクスへ柔軟に対応し，帯域幅消費量の低減だけではなく，問い合わせ処理の応答時間の向上をも達成することが可能であった．本研究でのアプローチは非構造 P2P システムの特徴であるシステムの自己組織化および柔軟性の特徴を失うことがなかった．さらに本研究での手法は従来の非構造 P2P システムでの手法と異なる点は，各ピアが積極的にコミュニケーションを行う対象を単に隣接するピアだけではなくスケーラブルに選択することを可能にすることである．そのため，本研究で提案した適応的なピア発見を可能にした分散インデックスおよび複製配置の方法は問い合わせ処理の向上への寄与に有効な手段であることが分かった．

Abstract

An “open” cooperative working environment that is one of data sharing system, such as BBS, blog, Wiki and so on, is popular more and more. This system has a huge amount of shared data and users. Due to an efficiently of data sharing on the system, it should be considered to introduce the mechanisms for an improvement in scalability and data update. One of the solutions is to introduce a Peer-to-Peer (P2P) system which is one of a distributed computing system and has the feature of autonomy and self-organization.

The target of this paper is an unstructured P2P system such as Gnutella and Freenet that is superior autonomy, flexibility and self-organization. For effective document sharing on P2P systems, because rapid response time and scalability are requested by users, the query processing needs to be improved. However, the query performance on an unstructured P2P system is not efficient. In this paper, to solve the issues, new techniques of distributed indexing and replication are introduced.

The purpose of the query message routing is that a query can be stopped in the few number of query messages. Due to an improvement in the efficient routing, in this paper the “usefulness” of the peers is introduced to the decision that a peer forwards query messages to which peers. Using the usefulness of peers, a peer makes a link (a long link) on the P2P network dynamically, and sends query messages to useful peers who may locate far away from the peer. A peer needs to store the peer information. The *Direct Index* (DI) is a table within the peer information. The purpose of DI is to collect “useful”peer information. Because a peer stores information of another useful peer in its DI, it can send query messages to a useful peer directly. To evaluate a peer, “usefulness” is set to the peer. In this paper, the usefulness of the peer was assumed to compute by the usefulness and number of data which the peer has: the usefulness of peers is measured by the measure which is calculated based on the access frequency and publishing time of date. The usefulness of data is decided by the demand of it. The demand for data is judged according to the reference frequency. However, the demand for data changes at time, because the demand of data has few possibilities now even if there are comparatively many reference frequencies of the data: it is meant

that the data is no demand now though it had demand before. Therefore, usefulness of data is lowered by the passage of time. As a result, the usefulness of data can be set high if newly modified though it is not referred so much. Query processing using DIs have three advantages: a reduction in bandwidth, an improvement of query response time and scalability. These reasons are as follows: in the case of using DIs, a peer positively sends a query message to useful peer, and the peer can store useful peers at a position away from the peer in a DI of the peer. For an index update, DI is not sensitive against changing the usefulness of another peer, and a peer can limit to propagate the index update messages with useless peers. If a peer receives an index update message, the peer judges whether the message is important or not. If the message is not important, the index update processing may be stopped. The mechanism can achieve a reduction in the cost of the index update on the network.

An unstructured P2P system is used in this paper. However, the system suffers from reliability and scalability problems. For example, when a useful document is lost because a peer leaves the system, some users cannot access their documents as the system expands. The loss of a document may not be solved in any efficient query routing methods because of "breakaway of the document from the network". In this paper I discuss the necessity of improving the data sharing mechanism efficiency by using replication. When considering a replication mechanism, the efficiency of query processing must be considered. Data updating is an important factor in cooperative working environments. In this paper, I propose new methods of replication together with a new method of data updating in un structured P2P systems. In this paper, when a new document is registered at a peer, the peer replicates the document using the following steps: (1) Decide the number of replications, (2) Decide which documents are to be replicated, and (3) Decide which peers will be sent replications. The number of replications is based on the usefulness of the source peer. That is, a document held by a useful peer should be replicated more than those held by less useful peers. Due to decision of the replica location, a method of an adaptive and dynamic peer grouping is introduced. If a peer starts a replication (or update processing), the peer (re-)places the replica to peers who joins the peer group of the peer. The

decision of the peer group formation is used the peer preference such as querying, data exchanging and network status.

To validate the proposed methods: the distributed index and the replication, the effects of the methods and characteristics is shown by the several simulations. The experimental results show that the indexing and replication mechanisms can achieve a reduction in bandwidth consumption and an improvement in the response time against other techniques for an unstructured P2P system. Moreover, the proposed techniques don't lose the features of unstructured P2P systems such as the flexibility and the self-organization at all. A unique point of this paper is that a peer selects a communication target from not only "local" (its neighbors) but also peers away from the peer. The proposed indexing and replication mechanisms based on adaptive peer discovery are the efficient to improve query processing on an unstructured P2P network.

目次

表目次

図目次

第 1 章

序論

1.1　はじめに

我々は実社会にて様々な活動を行うために多くの情報を用いる．それらの情報は特定の場所や個人のみに留まるだけではなく，“流れ” に従って多くの個人に伝播される．この情報の流れによって，人々は情報を取得することが可能になる．さらに新たな情報を生成し，流通させる．我々は情報の取得の媒介として各種マスメディアや，友人，知人または親類等のネットワークを利用することができる．しかしながら生成された情報は膨大な量であるため，我々はいかなる情報を取得し利用するかを決定する必要がある．その判断は個々人にて行われるはずであり，情報の取得先も個人ごとに変化するはずである．ある事件に関する情報をマスメディアを通じて取得したある人はその情報を他人に伝播することが可能であり，さらにその事件の情報への個人の見解や他の媒介から得た情報等の関連する情報を付加してさらなる伝播を行うことも可能である．ここでの情報は全体像としての情報と個人レベルの情報の 2 つの情報が混在しているが，これらの情報の取捨選択を個々人が行う．また，マスメディアと個人では情報の保有量に差があることは明白である．マスメディアは多くの人々に伝播させるための仕組みを備えており，個々人が伝播させる情報よりもよりスケーラブルに情報を流通させることが可能である．しかしながら，マスメディアが伝播させる情報は世の中のすべてをカバーしているわけではない．また，マスメディアからの情報は不特定多数への伝播を行うため，情報の内容は一般的なものである．他方，個々人が伝播させる情報はその人が取得した内容に対して関心もしくは関連しそうであろう人へ伝達される傾向があるように思われる．個人が保有する情報のすべてをその人にとって既知である人々のすべてに対して伝播させるわけではない．個々人が伝播させる情報とマスメディアが伝播させる情報にはその性質に差異が生じる．

さて，インターネットの利用の拡大，拡充は電子化された文書を爆発的に生成させている．インターネット利用の目的として，ニュース配信，天気予報，情報公開・解説等の情報閲覧，航空券・新幹線の予約，ホテルの予約，オークション等の Web アプリケーションやファイルの公開・共有システム等様々である．インターネット上の “オープンな” 情報を利用するためには利用したい情報を得る必要がある．そのための仕組みとして Google の様なサーチエンジンを挙げることができる．サーチエンジンはインターネット上に分散された情報に対してインデキシングを行う．インデックス内には各サイトの Web ページの位置や内容，関連ページ等の情報を格納し，それらを用いて検索サービスを提供する．世界中の Web サイトの情報に対する検索サービスを中央サーバで提供する場合，その探索を簡単にすることができる．各ユーザは中央サーバに問い合わせるだけで取得したい情報に関する検索結果を得ることができる．インターネット上でオープンにされている情報に対する検索は中央サーバのインデックス内に格納された情報に対する検索にて終えることが可能になる．

しかしながらこのことを実現するためには問題がある．先ほどの各マスメディアと同様に，格納される情報には制限があることを挙げることができる．また，各テレビ局がその放送区域にサービスを提供するために電波塔を設置する必要があるように，サーチエンジンでも膨大なユーザに対してサービスを提供可能にする必要がある．これらの問題，すなわちスケーラビリティを確保するため，サーバの追加を行う必要がある．また，中央サーバの構造はその耐久性の問題も抱えている．その例として DoS（denial of service attack）攻撃がある．2000 年 2 月，Yahoo!がサンタクララにて抱えている数百台の計算機は “Yes, I heard you!” というメッセージを受け取った．問題はそのメッセージ数が 1,2 通ではなく数十億であった．ある 1 人のクラッカによって学校や研究所，各企業のサーバが乗っ取られ，それらの計算機が数千のメッセージを Yahoo!のサーバへ矢継ぎ早に送信したためである．このため，正規のユーザの Yahoo!に対する要求は数分間待たされる結果となってしまった．検索サービスを中央にて提供するためには非常に高コストになってしまう．

中央サーバのインデックス内に Web ページの情報を格納するためにはあらかじめロボット等によってインターネット上の情報を取得する必要があり，この作業を定期的に行っている．そのため，各サイトの更新を即時に中央サーバに反映することは困難である．更新が多発する例として BBS（掲示板システム; Bulletin Board System）を代表す

る “オープンな” 協調作業環境がある．BBS は任意のユーザが任意の議論を交わす場である．ユーザからの write を可能にし，その情報を即座に反映することが可能である．また，BBS では多人数間での情報共有を行うことができるため，その利用者は多い．1 つの文書を多人数で作成し共有するシステムとして最も普及しているシステムの 1 つであるといえる．各ユーザは自由に write 操作が可能であるため，write 操作が多発するに従って多くの文書が生成される．

また，blog および Wiki を代表とする CMS（Content Management System）が近年話題となっている．blog の利点は文書の管理にある．例えば，文書の生成，更新を FTP 等を意識して利用することなく行うことが可能である．また，全文検索以外にトピック検索を可能にし，各トピックに対する引用の自動化を行う．さらに trackback 機能により更新を的確に知るこることが可能である．blog は個人での利用（日記）だけではなく個々のユーザ間での意見のやりとりとして利用されるケースが多い．Wiki では版管理（versioning）を提供する．また，文書内のキーワードを自動抽出し，そのキーワードに対してそのサイト内でリンクを自動生成する．Wiki では多人数で文書を構成する場合に利用されることが多い．例としては，メモ帳，辞書の作成，ソフトウェア等のマニュアル作成等がある．オープンな協調作業システムではページの更新の問題解決のため，RSS（RDF Site Summary）を用いて更新情報を配信するサイトが増加している．多くの blog サイトでは RSS をヘッドライン配信の手法として利用している．RSS はニュースクリップのように簡単な文書の要約を XML 形式で作成したメタデータである．RSS を読みとるためにはそのサイト情報を記述する必要があり，また複数のサイト間での情報の重複も多く，ニュースを列挙しているだけの RSS もあり利用しやすいものばかりとは限らない．ユーザがキーワードを指定して自動的に分類できるものもあるが，ユーザが所望するものだけを選択し取得することは困難である．また，すべてのサイトで RSS を公開しているのではないため取得可能な情報には制限がある．RSS を公開しているサイトの増加に伴い，RSS のメッセージの氾濫によって引き起こされる帯域幅消費量の増加が新たな問題として指摘され始めている．

また各情報はその位置に依存する．サーチエンジン等での検索結果に基づいてある情報を手に入れるためにはその情報が配置されているサイトに実際に訪れる必要がある．特定のサイトへ訪れるユーザ数の増加は先ほどのサーチエンジンの例と同様の問題を抱えることになる．そこで分散システムを導入することでスケーラビリティの改善を行い，シ

ステムの効率性を向上させる方法がある．そのための方法の 1 つとして CDN（Content Delivery Network）がある．CDN では各サイトの情報をキャッシュするサーバ（エッジサーバ）を導入している．各ユーザからの問い合わせに応じて，そのユーザに “最も近い” エッジサーバの位置を返し，ユーザは指定されたエッジサーバから文書を取得することが可能になる．CDN を利用するためには，サイトごとに CDN サービスへ登録する必要がある．そのため，blog のようにサイト間で連携させる場合，各サイトが CDN に参加しなければ CDN の効果を発揮することができないかもしれない．また，これらのサイトではサイト間の連携は動的である．そのため BBS，Wiki または blog のサイトの多くが CDN サービスを利用することは現実的ではない．

オープンな協調作業環境下でやりとりされる情報の内容は，マスメディアを媒介として共有される情報とは異なる場合が多い．また，グループウェアの様に閉じた環境での内容とも異なる．電話のように完全に 1 対 1 でのやりとりとも異なる．ある特定の人々の間での情報を伝播できればよく，またその情報を必要とする人であれば特に断りなくその場で共有されている情報は取得することが可能であるべきである．このような情報の伝播は先ほどの個々人による情報の伝播と同様の方法で行うことができれば，効率的に行うことができるのではないであろうか．

本研究では情報の共有における効率性を考える．特に，個々人が生成し共有する情報の取得の効率化について考える．本研究は現在の中央サーバからもしくはマスメディアを媒介とする情報取得を批判しているわけではない．それらの中央にて取得可能な情報とは異なる性質をもつ情報，特に個々人にて生成され，流通される情報，不特定多数の人々に対してではなくある特定の人々に対して有用であると思われる情報をいかに効率的に共有するかを問題としている．ここではどこから目的とする情報を見つけるか，どこへ自分が知っている情報を伝播させるか，という問題がここでは発生する．ある特定の話題に関して関心を持つ人は常にその話題に関して関心を持つ保証はない．また，ある日突然のように（何らかのきっかけとなる事柄があるかもしれないが）ある話題に対して人々は関心を持つかもしれない．さらに現在の個々人が情報を伝播させる，させられる対象が今後もその対象となるとは限らないかもしれない．例えば，引っ越しや移転に伴い簡易に情報を伝播させる，させられる対象は変化する．すなわち，ネットワークにはダイナミクスが存在する．ネットワークのダイナミクスを考慮したシステムにて協調作業を行えば，中央サーバを用いたシステムよりもより “自然な” 形態で，個人間でのもしくはある関連する人々間での情報の共有を行えると考えている．

完全に分散化され，ネットワークのダイナミクスを考慮したシステムとして自律分散型システムである Peer-to-Peer（P2P）システムを挙げることができる．P2P システムとはシステムに参加している各ノード（ピア）の役割や機能は等しく，これらのピアによって構成される分散コンピューティングシステムである．P2P システムは近年，Napster[61] や Gnutella[33] 等に代表されるファイル共有システムとしてだけではなく，インスタントメッセージング（IM）[43] やグループウェア [37] に対する利用も行われている．オープンな協調作業環境に対して P2P システムを利用する利点は以下の通りである．

- 文書の検索は文書の位置に依存（URL に依存）しない．そのため P2P システムは位置透過性に優れたシステムであり，リンク切れや文書更新に伴う文書の位置の変化に対して効率的に対応可能である．
- システムのスケーラビリティに優れている．P2P システムでは，各ピアは他のピアを動的に発見し，それらのピア間での接続および切断を中央からの命令なしに行うことが可能である．そのため P2P システムでは各ピアの自律性は他の分散システムよりも高く，また自己組織化を行うことが可能である．
- 更新のある文書の効率的な共有が可能である．各文書の更新情報は自律的に伝播される．そのため機敏に文書更新の反映が可能な仕組みを P2P システムは提供する．

協調作業環境では効率的な問い合わせ処理が求められる．これは P2P システムを利用した場合は特に考慮すべき点である．それは以下の通りである．

- 帯域幅消費量の低減.
- 応答時間の向上.
- システムへ参加するピアの増加への対応.
- システムからのピアの離脱.

例えば，Gnutella ではフラッディング（flooding）による問い合わせ処理を行う．フラッディングでは問い合わせ処理を行うピアはその全隣接ピア（neighbor）に対して問い合わせメッセージを転送する．問い合わせメッセージを受け取ったピアはさらにその隣接ピアへ転送する．この仕組みでは，問い合わせメッセージは指数関数的に増加してしまう．そのためネットワーク全体での帯域幅消費量が増加してしまい，システムのパフォーマンスを低下させてしまう．また，Gnutella では中央にサーバが無いためにフラッディングを

行うが，この仕組みにより，問い合わせ処理の終了条件を満たすまで（ここでは検索結果を m 件以上集めることを目的とする）の応答時間が Client-Server システムよりも低下してしまう．Gnutella では問い合わせメッセージの転送される範囲をあらかじめ指定する．P2P システムでは他の分散システムよりもシステムに対するピアの出入りが頻繁に行われるため非常に動的なシステムである．P2P システムでのユーザの参加に従って，P2P ネットワークに参加するピアの数が増加すると問い合わせメッセージが転送されないピアが存在してしまう．そのため，システム全体での検索結果を得ることが困難になる．ピアがシステムから離脱するとそのピアが保持していた文書に対してアクセスすることが不可能になる．

1.2 適用例

本研究では図 1.1 に示すように共有される文書が更新され，またその文書を必要とするユーザが局所的である状況下でのシステム，オープンな協調作業システムをその適用として想定している．その適用範囲には，電子図書館システム，オープンソフトウェア開発，Wiki，blog および BBS 等がある．

たとえば電子図書館システムに適用された場合を考えてみる．電子図書館システムは，莫大な量の電子化された文書をデータベース化し，これとそれらの文書に対する情報に関するデータベースとを統合した情報検索システムである．また各文書に対して，ユーザや著者等によるコメントや，関連する文書やその他のその文書に関連する情報を基に各文書のメタデータを生成し，これを提供する．ある文書を取得する場合，その文書に関するメタデータを取得後，そのメタデータ内の情報に従って，各項目の情報をシステム内から取り出す．そこでメタデータをユーザに効率的に提供するためにメタデータの効率的な問い合わせ処理が必須である．メタデータ内には文書の識別子，題目，著者情報，トピック等の文書自体の情報，他の文書への参照情報，他の文書からの参照情報等の関連情報，著者や他のユーザからのコメント等の文書に対する付加情報から成り立つ．文書自体の情報には文書の識別子があるが，従来のシステムではこれは URI（Uniform Resource Identifier）を指し，実際には URL（Uniform Resource Locator）等に代表される，ネットワーク上での位置に依存した識別子が用いられることが多い．P2P システムを用いた場合，データは位置に依存しないため，文書の移動やメタデータの更新に伴うリンク切れの問題が解消される．これは参照情報等の関連情報やコメント等の文書に対する付加情報

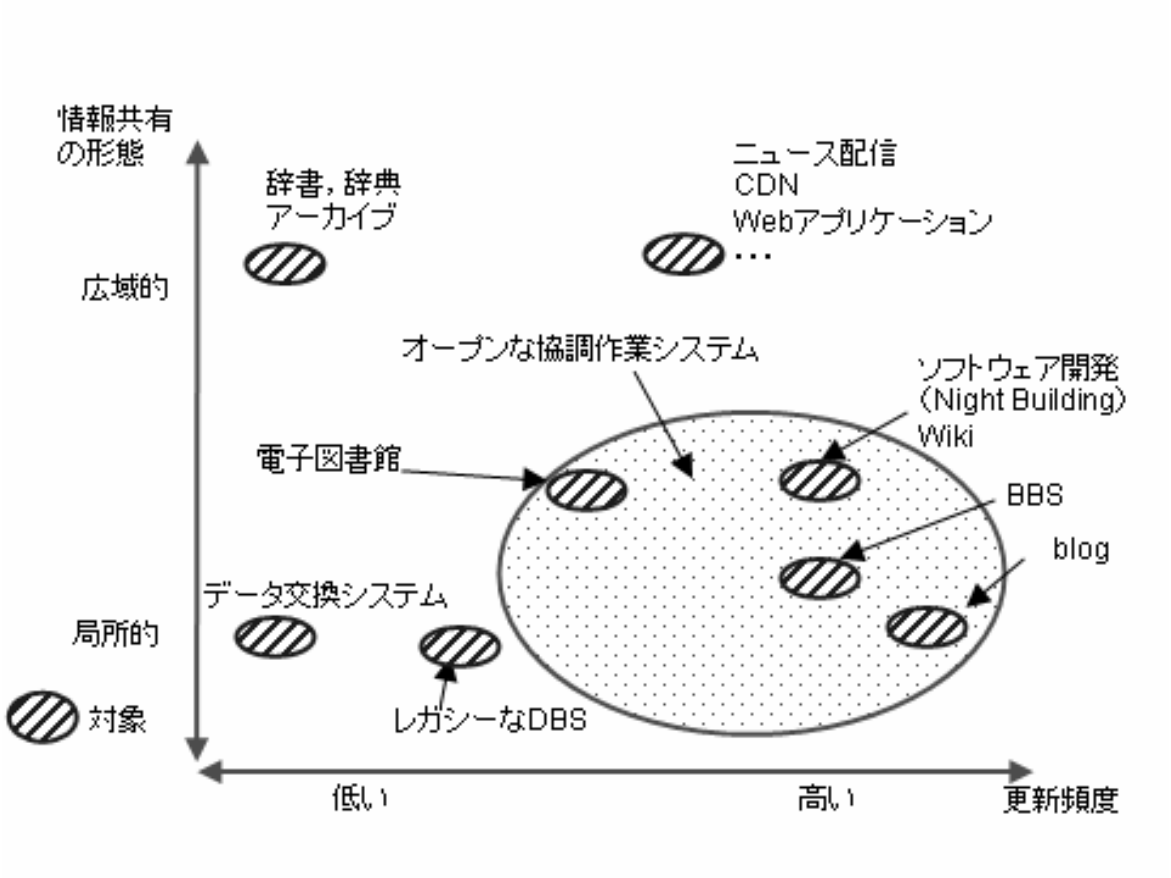


図 1.1 本研究の対象

にも同様のことがいえる．想定している P2P システム上にて電子図書館システムを構築する場合，付加情報はその発言を行ったピア上にそのコメントの実際の情報を配置する．そのコメントに対する識別子をメタ情報内に追加することで，メタデータは更新される．またメタデータ生成は各ピア上にて行い，そのピアがそのメタデータの所有者になる．そのため，各メタデータの管理はそれを所有しているピアにて行うため，メタデータの管理を中央にて行わない．

1.3 本研究の目的

本研究での目的は，P2P システムを利用したオープンな協調作業環境での文書の共有の効率性を向上させるために，P2P システムでの問い合わせ処理のパフォーマンスを向上させることである．そのためには帯域幅消費量の低減，応答時間の向上，参加するピアの増加・システムからのピアの離脱に対する対応を解決する必要がある．想定する P2P システムにおけるサービス層にて提供するルーティングおよび局所処理に関して新たなる

手法を導入し，P2P ネットワークでの帯域幅消費量の低減，応答時間の向上およびスケーラビリティのあるシステムを目指す．そのために問い合わせメッセージの経路を導くために新たな分散インデックスシステムを提案し，それに基づく問い合わせメッセージのルーティングの方法を示す．局所処理として文書の複製を行う．この文書複製方法として，動的にピアグループを形成し，そのグループ内で生成された複製を配置する方法を導入する．さらに，各種シミュレーションにてこれらの 2 つの提案事項の効率性を確かめる．

1.4　本論文の構成

本論文は 6 章で構成されている．

第 2 章では「関連研究」では，本研究と関連する分散システムや P2P システムでの問い合わせ処理に関する関連研究とを比較し，本研究の特色を示す．

第 3 章「Peer-to-Peer システム」では，本研究で用いた P2P システムでのシステムコア層の詳細，および各セッションでのプロトコルを示す．

第 4 章「問い合わせメッセージのルーティング」および第 5 章「複製配置」では導入する P2P システムでのサービス層に関して述べる．第 4 章では，本研究で提案する分散インデックスを導入し，その分散インデックスの詳細を述べ，提案する分散インデックスを用いた問い合わせメッセージのルーティングがいかに行われるかを示す．第 5 章では，本研究で提案する文書複製の方法および手順の詳細を述べ，さらに文書更新に対する手続きを示す．

第 6 章「実験」では，第 4 章および第 5 章で提案したルーティング方法および文書複製の効果を確かめるべくシミュレーションによる実験を行い，これらの提案手法の効果を示す．

第 7 章「結論」では，本研究での結論および課題について述べる．

1.5　本章のまとめ

オープンな協調作業環境の効率性を向上させるために，blog や Wiki を代表とする CMS（Content Management System）によって従来の BBS のようなシステムよりも文書の管理の機能は向上した．協調作業環境では各ユーザによる write 操作が多発するため，文書の更新頻度が高くなる傾向にある．そのため，更新に対して柔軟に対応可能な仕組みが必要である．また，オープンな協調作業環境を提供する各サイトでのユーザ数の増

加はシステムの管理コストを増加させてしまう．これらの問題に対して P2P システムを利用することはその解決の手段の 1 つである．しかしながら P2P システムでは問い合わせ処理に関して問題がある．これを解決するために，本研究では分散インデックスを導入し，文書複製・配置を行うシステムを提案する．提案する分散インデックスおよび文書複製手法を評価するためにシミュレーションを行い，その効果を示す．

第 2 章

関連研究

2.1 はじめに

本章では，本研究で用いる P2P システムのレイヤを示し，各 P2P レイヤに関連する研究を列挙し，本研究との比較を行い，本研究の特徴を示す．本研究は他の P2P システムだけではなく，他の分散システムとも関連する．特に，本研究ではサービス層に比重をおいている．そのため，サービス層に関連する研究を重点的に示す．また，サービス層にて提供される機能はその階層であるコア層を利用することで実現することが可能になる．そのため，サービス層で提供されるルーティング機能及び局所処理機能はコア層，特にシステムコア層に依存する．そこでサービス層を重点的に述べるが必要に応じてシステムコア層の内容を加える．

2.2 Peer-to-Peer システムレイヤ

オープンな協調作業に限らず，P2P システムを何らかのデータ共有システムに適用するためには，P2P システムがどのような機能を提供するかを定める必要がある．そのために P2P システムアーキテクチャを提供する必要がある．図 2.1 は本研究で用いた P2P システムのレイヤを示す．本研究で用いた P2P システムは下層よりコア（Core）層，サービス（Service）層，アプリケーション（Application）層の 3 層から成り立つ．

アプリケーション層ではその下層であるサービス層の機能を利用して構築することができ，電子図書館や Wiki，blog 等のオープンな協調作業環境を提供するようなコミュニティアプリケーションや，情報検索または情報融合，情報統合等の提供を行うアプリケーションを想定している．

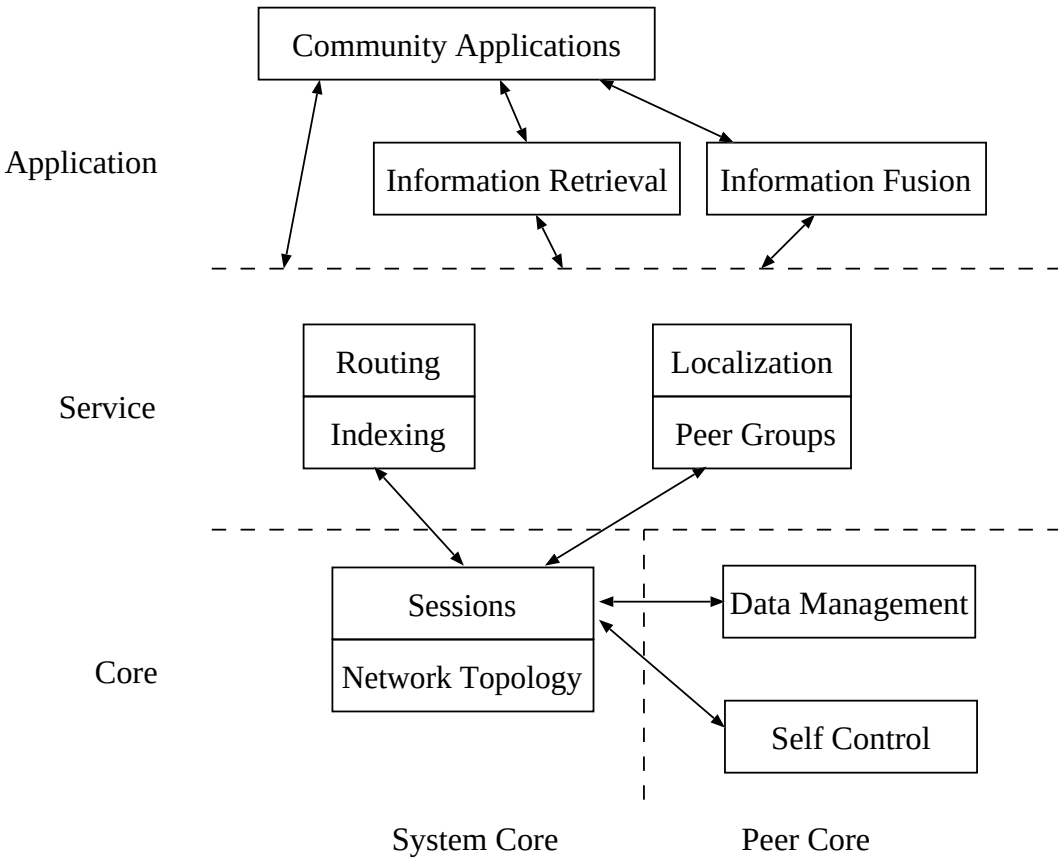


図 2.1　P2P システムレイヤ

コア層ではさらに，ピア間での通信をサポートするシステムコア層と各ピアの内部での処理をサポートするピアコア層の 2 層に分けられる．ピアコア層では各ピアによるデータ管理とピアの情報管理のための機能を提供する．データ管理の機能としては，各データのデータ名，生成時刻，更新時刻，参照回数，トピック情報等の管理がある．また，ピア情報管理の機能としては，ピア識別子，システムへの参加状況，隣接するピアの情報等の管理がある．各ピアはローカルにデータベースを持ち，ローカルインデックスを通して参照することができる．各ピアは問い合わせの内容を含む文書をローカルインデックスを用いて判断し，ローカルインデックスはその問い合わせにマッチする文書へのポインタを返す．システムコア層では，P2P ネットワーク上での各ピアの活動をサポートし，ネットワークトポロジおよびピア間での連携のための機能を提供する．与えられたネットワークトポロジに従って各ピアが他のピアに対して何らかの通信を行うとき，これをサポートする機能がセッション（session）である．他のピアに対する接続メッセージの転送，問い合わせメッセージの転送，データの転送を行うときに各ピアはセッションを利用して各メッセージをピア間でやりとりする．セッションはピアの活動に応じて以下の 3 つに分けられる．

- P2P システムに参加・離脱する，もしくは他のピアへの接続・切断する—接続セッション
- 問い合わせ処理を行う—問い合わせセッション
- データを転送する—転送セッション

ピアの各活動に従って，システムコア層を用いて問い合わせ処理やデータ転送等をサポートするのがサービス層である．サービス層ではルーティング（Routing）機能と局所処理（Localization）機能をサポートする．局所処理ではデータの転送に関する処理を提供する．データの転送は，各ピアが検索結果に応じてデータの転送要求を行ったとき，またはデータの複製を配置するときに用いられる．特に複製されたデータの配置を行う場合，複製の配置場所の候補としてピアグループ（Peer Group）を決定する．ルーティングでは各問い合わせメッセージの転送先の指標を与えるためインデキシング（Indexing）を行う．インデックスの更新は接続セッションを介したインデックス更新メッセージの伝播によって行われる．

このシステムのシステムレイヤは Jxta[70] で用いられているシステムレイヤに準じている．Jxta のシステムレイヤは多くの P2P システムで用いられている．また，Jxta はオープンソースの形態で提供されており，多くの大学等の研究機関やベンチャー企業によって Jxta プラットホームは広く利用されている．他の P2P システムアーキテクチャもほぼ Jxta と同様のシステムレイヤである場合が多い．Oceanstore[54] を利用する P2P システムでもほぼ同様のシステムレイヤである．Oceanstore 自体は本研究でのサービス層に相当するが，Oceanstore を用いたシステムでは，その上位層であるアプリケーション層およびその下層であるコア層が存在している．Oceanstore での基盤は Tapestry[83] を用いている．また，PIER（Peer-to-peer Information Exchange and Retrieval）プロジェクト [40] で用いられているシステムレイヤは Jxta でのアプリケーション層を分化させた形態であり，下層より DHT 層，PIER 層，アプリケーション層から成り立っている．ここで DHT 層は Jxta でのコア層，サービス層に相当し，ピア間のコミュニケーションに関する機能やルーティング等のサービス機能を提供する．PIER 層は情報検索の機能を提供する層であり，アプリケーション層は本研究でのコミュニティアプリケーションに相当する．このシステムも基本的に Jxta と同様のシステムレイヤの形態である．

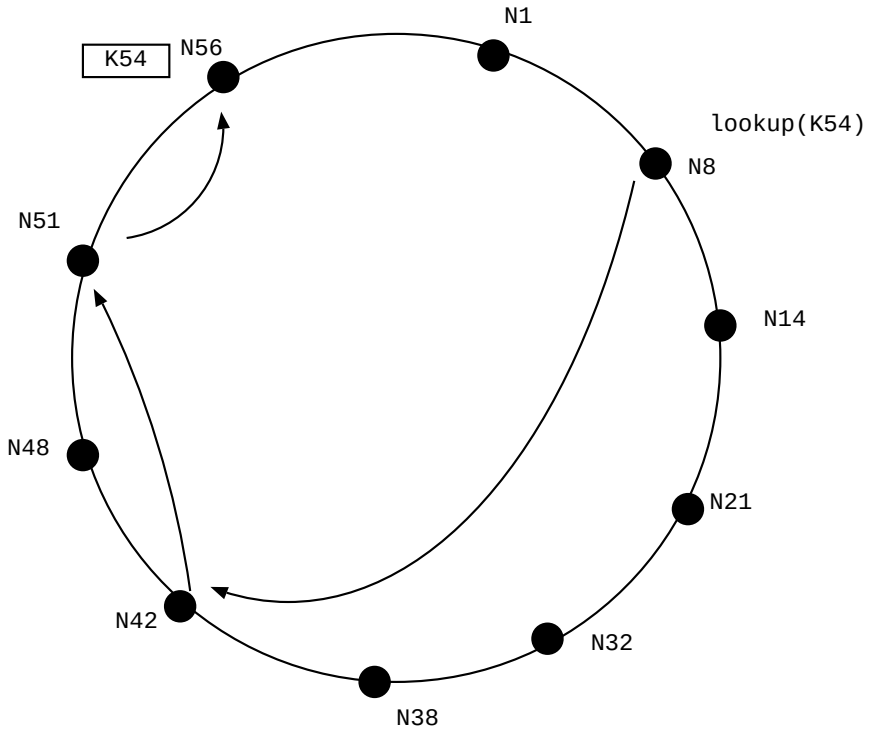


図 2.2　Chord の例

2.3　サービス層

2.3.1　ルーティング機能

ルーティング機能では問い合わせメッセージの伝達方法を提供する．そのためにはまず，メッセージをいずれのピア，ノードに転送するかを決定する必要がある．そのための指標としてルーティングテーブル等を利用したインデキシング機能を付随して提供する．

IP ネットワークにおいて経路情報を管理する手法として，スタティックルーティングとダイナミックルーティングの 2 種類が存在する．スタティックルーティングでは経路情報を各ルータ内に手動で設定する．この経路情報は基本的にルーティングテーブル内に格納する．一方，ダイナミックルーティングでは，RIP（Routing Information Protocol），OSPF（Open Shortest Path First）等のルーティングプロトコルによってルータが経路情報を動的に構築する．RIP では Distance Vector 方式と呼ばれる方式でルーティングテーブルを構築する．Distance Vector 方式 [39] では，各ルータはまず最初に自分のネットワーク名（ネットワークアドレス）をブロードキャストする．この結果，各ルータは自分に隣接するルータの情報（ルータの存在やそのルータまでの「距離」）を得ることがで

きる．次にこの情報をまとめてブロードキャストすると，隣接ルータのさらに隣のルータ情報が得られる．このようにして次々とルータ情報が伝播していく．この方式はインプリメントが容易という長所がある半面，情報伝播が遅いほか，ネットワークの規模が大きくなるとさまざまな運用上の問題が発生するという欠点がある．OSPF では Link State 方式 [16] と呼ばれる方式でルーティングを構築する．Link State 方式は自分の隣接ルータの情報を全ルータへ直接送信する方式である．受け取った側では，各ルータからの情報を基にしてネットワーク全体の構造を決定し，最短ルートを計算可能にする．この方式はインプリメントがやや複雑で，ルーティングテーブル情報も巨大になり，ルーティング情報を転送するための帯域幅消費量も大きくなるという欠点があるが，効率よく動的ルーティングを行なうことが可能である．これらの IP テーブルを用いたメッセージのルーティング方法は物理的なルータ間での方法であり，目的地に対してデータを送信するためのデータ転送における仕組みを提供する．これに対し，各分散システムでのメッセージのルーティングの方法では論理的なオーバレイネットワーク上での方法であり，各種サービスやアプリケーションの目的に応じたノード間の接続の方法を提供する．

分散データベースシステムでは 1 つのデータベース演算を並列化させるための方法としてデータベースを分割し，分割単位で処理を並列に行うことを可能にする．データの分割とともに問い合わせ処理の向上のためにはインデックスは重要な要素の 1 つである．$LH*$[57]，$RP*$[56] に代表される SDDS（Scalable Distributed Data Structure）は分散データベースでのインデックスメカニズムであり，厳密な問い合わせのマッチング，問い合わせの範囲および I/O クラスタリングを管理する．$RP*$ ではネットワークトポロジとして $B*-tree$ を用いる．各データはサーバ上に存在し，それらのデータのサーバへの分配方法はハッシュによる分割を行う．$RP*$ は $RP*_N$，$RP*_C$ および $RP*_S$ の 3 タイプ存在する．$RP*_N$ では各サーバがサポートするデータの範囲は同じ大きさに分割されるが，インデックスを持たない．クライアントが問い合わせを行う場合，各サーバに対してマルチキャストを行う．$RP*_C$ では各クライアントはそれ自身のキャッシュに対するインデックスを保持する．そして，問い合わせを行う場合，クライアントは保持しているインデックスに基づきマルチキャストを行う．$RP*_S$ では各サーバがインデックスを保持している．問い合わせ処理では，クライアントが問い合わせを任意のサーバに対して問い合わせメッセージを転送するとサーバはそれ自身のインデックスに基づきその問い合わせメッセージの目的となるサーバを特定し，point-to-point にて問い合わせメッセージを転送す

る．問い合わせメッセージを受け取ったサーバはその結果をクライアントに IAM（Image Adjustment Message）と呼ばれるメッセージで返す.

Akamai[28] に代表される CDN（Content Delivery Network）システムではコンテンツプロバイダのサーバからオリジナルデータを取得する代わりにユーザにとってネットワーク上で最も近い位置に存在するエッジサーバ（キャッシュサーバ）にキャッシュされたデータを見つけるような問い合わせ処理を行う．例として CDN Japan[11] ではコンテンツプロバイダサーバが用意されており，ユーザはエッジサーバ上のデータにアクセスする．ユーザは最初にコンテンツプロバイダサーバにアクセスを行い，認証を行う．コンテンツプロバイダサーバはユーザの認証を行うと，ネットワーク上でそのユーザから最も近い距離にあるエッジサーバへアクセスを振り分け，実際のコンテンツ配信は振り分けられたエッジサーバから行われる．エッジサーバにはあらかじめコンテンツプロバイダサーバよりコンテンツの転送が行われいる.

P2P システムでの問い合わせメッセージのルーティングは分散データベースの問題とも関連する [53]．しかしながら分散データベースでの各ノードはシステムへ長時間参加状態にあり，スタティックである．またシステムに参加しているノード数は P2P システムに比べかなり少数である．このことにより，分散データベースに用いられるインデックスの方法はそのまま P2P システムへ適応するのは必ずしも適切ではない.

Napster[61]，OpenNap[63] および KaZaA[48] に代表されるハイブリッド *P2P*（*Hybrid P2P* ; *HP2P*）システムは Client-Server システムにおけるサーバの様に振る舞う *super peer* と呼ばれるピアが存在する．super peer はシステムに参加している各ピアの情報（各ピアが所有しているデータのファイル名，サイズ，作成・更新日時，アクセス頻度等の各データの情報やそのピアの識別子，ネットワーク帯域幅等のピア自体の情報）を維持，管理し，この情報を利用してインデックスを構成し，問い合わせ処理のサービスを提供する．実際に共有されているデータの管理を行っているのは各ピアであり，super peer は実際のデータを保有していないため，データのやりとりは各ピア間にて直接行われる[*1]．Yang 等 [78] は HP2P ネットワークでの super peer 間でのインデックスの伝播の方法や一般のピアがネットワークへの参加の方法について各種方法を挙げ，それらを比較，検討している．しかし，このタイプのシステムは super peer が停止してしまうとシステム全

[*1] KaZaA では super peer は一般のピアより動的に決定される．そのため，super peer 自体にも共有データを保有している場合もある.

体も停止してしまう．また super peer への問い合わせの負荷が集中してしまう危険性が高い．さらにシステムのスケーラビリティ確保のためには super peer の数を増強を行う必要がある．そこで HP2P システムでは super peer の管理に関する問題があるため，その可用性（availability），スケーラビリティの点で注意を払う必要がある．

Gnutella[33] および Freenet[31] に代表されるピュア *P2P*（*Pure P2P*; *PP2P*）システムでは super peer は存在せず，各ピアは自律分散する．そのため PP2P システムでは特定の機能を有するピアが存在しないため，HP2P システムよりも可用性に関して優れる．ピアの情報はネットワーク内の一部分のピアによって管理され，問い合わせメッセージは各ピアが保持しているインデックス（分散インデックス）によってルーティングされる．PP2P システムはさらに構造化されたネットワークを用いるシステムと用いないシステムに分類される．

構造化されたネットワークを利用するシステムでは，主に *DHT*（*Distributed Hash Table*）を利用する．DHT では $lookup(key)$（問い合わせキー key に応じて問い合わせメッセージのルーティング）のみを提供する．各ピアはそのピアが保有しているデータ d とそのデータに対してあるハッシュ関数例えば SHA-1[22] によって求められるハッシュ値 $hash(d)$ で構成されるレコードを DHT に格納する．また，各ピアにはユニークな識別子（unique identifier; UID）が設定されており，あるピア P の UID に対するハッシュ値 $hash(P)$ が $hash(P) = hash(d)$ であるならばデータ d とそれを保有しているピアの IP アドレスをピア P に通知する．C というデータに関して問い合わせを行う場合，問い合わせを行うピアは $hash(C)$ を計算し，これを問い合わせキー key として $lookup(key)$ を実行する．DHT を用いた P2P システムでは問い合わせメッセージの効率的なルーティングを行うために構造化されたネットワークを用いる必要がある．このような仕組みの例として CAN[65]，Chord[67]，Tapestry[83]，P-Grid[1]，Pastry[66]，Oceanstore[54]，Symphony[59] 等がある．これらのシステムでのネットワーク構造は以下の通りである．

- リング構造 : Chord，Symphony
- N 次元トーラス : CAN
- 2 分木 : P-Grid
- メッシュ構造 : Tapestry，Pastry

Chord では図 2.2 に示すような 1 次元のリング構造のネットワーク構造を用いて問い合わせメッセージをルーティングする．Chord は Consistent Hashing[47] をスケーラビ

リティの点で改良した分散ハッシュシステムである．Consistent Hashing では高い巨大な 1 次元空間を想定する．Consistent Hashing に参加するピアはこの空間上 2^m でのランダムな ID が割り振られる（ここでは ID 空間のサイズを $m = 6$ とする.）．あるデータ a のハッシュキー K_a はハッシュ値 $hash(a)$ にて求められるものとして，K_a はその値で示される ID のピアに保持される．例えば，ハッシュキー $K54$ はピア $N54$ によって保持される．また，K_a で示される ID のピアがシステム内に存在しない場合，K_a が示す ID の位置から右回りに最初のピアによって保持される．例えば $K54$ は $N54$ および $N55$ を ID とするピアが存在せず $N56$ を ID をするピアが存在するならば $N56$ によって保持されうる．Consistent Hashing では各ピアは他のピア全体の存在をあらかじめ知っておく必要があるためスケーラビリティの点で問題がある．Chord では各ピアが保持しておくべきピア数は $O(\log N)$（ここで N はシステムに参加している全ピア数）である．各ピアは他のピア情報を finger table と呼ばれる分散インデックス内に格納する．各ピアによって保持すべきピアは $N_p + 2^{k-1} for 1 \leq k \leq m$ である．ここで N_p をピア p の ID とする．図 2.2 にてピア $N8$ が $K54$ に対する問い合わせを行う場合，$N8$ の finger table 内で $N54$ に最も近いピアは $N42$ である（$8 + 2^5 = 40$ であるため $N40$ を保持すべきであるが $N40$ はシステム内に存在しないため $N42$ を代わりに保持する）ため，$N42$ に問い合わせメッセージを転送し，その後問い合わせメッセージは $N42 \rightarrow N51 \rightarrow N56$ の経路で転送される．Chord では問い合わせメッセージ転送にて 1 ホップごとにハッシュキー k を保持するピアまでの残りの空間上での距離を半減させることを保証するため，問い合わせ処理での負荷は $O(logN)$ 程度に抑えることが可能である．効率的に問い合わせメッセージをルーティングすることができ，その伝播範囲はシステム全体であり，スケーラビリティの優れてたシステムを理論上構築可能である．特に Oceanstore ではピア数が 100 億のオーダでの参加している状況下での稼働を可能にするといわれている [54]．しかしながら，ネットワーク構造の維持のためには finger table を絶えず管理する必要があり，このコストは $O(\log N)$ であることが知られている．DHT を用いたシステムでは，効率的な問い合わせメッセージのルーティングが行われるが，そのためにはネットワークの構造のを維持コストの問題がある．

非構造ネットワークの例として Gnutella[33]，Freenet[31] 等が挙げられる．Gnutella では各ピアに隣接するピア（隣接ピア ; neighbor）の情報および各ピアのローカルでのデータ情報のみを用いて問い合わせ処理を行う．問い合わせメッセージは各ピアの全隣接

ピアへ転送され，さらにその隣接ピアへ転送される，フラッディング（flooding）による方法で伝播される．問い合わせメッセージには転送回数が事前に TTL（Time To Live）として決定されており，転送されるたびにこの値は 1 減り，$TTL = 0$ となるまで転送が行われる．Freenet でもフラッディングによる問い合わせ処理が用いられるが，ピア s の問い合わせ結果がピア t にある場合，$s \rightarrow t$ 間に位置するピア上に問い合わせ結果がキャッシュされる．そのため，実際に問い合わせ結果を保有するピアに到達する前に問い合わせ結果を得ることが可能である．しかしながら，キャッシュを用いて問い合わせメッセージの転送を効率的に行うためには，ある問い合わせが行われた経路上にて再び同様の問い合わせが行われなければならない．またキャッシュに存在しないデータに関して効率的に問い合わせを行えない．

local index[79] を用いた場合，以下の問い合わせ処理が行なわれる．各ピアは半径 r 以内に存在するピアの情報をインデックス内に格納する．ピア p が問い合わせを行う場合，まず p 自身のインデックスを使って問い合わせ処理を行う．ここで該当するデータがあればそのデータを結果として返す．また，返された結果が不十分な場合は，一定の規則に従って他のピア p' に問い合わせを転送する．p' はそのインデックスに対し問い合わせ処理を行う．これを繰り返す．転送規則の例として，問い合わせを行うピアと転送を打ち切るピアをホップ数で指定するものがある．例えば，問い合わせを行うピアは，問い合わせを発したピアから 1 および 5 ホップで到達できるピアであり，5 ホップを越えて問い合わせを転送しないといった規則が用いられる．P2P システムでは隣接ピアのネットワークへの参加・離脱やローカルデータベース内の更新等によってインデックスが更新されその更新が伝播する．この更新メッセージはピアの半径 r が大きければそれだけ増加することになる．

また，データの位置ではなく，問い合わせメッセージを転送の “方向” を示すインデキシングの方法も存在する．これらの方法には *Routing Index*（*RI*）[19]，*Adaptive Probabilistic Search*（*APS*）[74] 等の方法がある．RI では各ピアのデータをトピックごとに分類し，トピックごとにどれくらいのデータが存在するかという情報を伝播し，この情報を受け取った各ピアは保持している分散インデックス内に格納する．そのため，トピックごとにどの方向に問い合わせメッセージの転送を行えばよいかという判断をどのピアも行うことができる．このインデックスでは，P2P システムが扱うデータのカテゴリを定義し，このカテゴリごとにデータを多く有するピアの情報をインデックスとして用い

る．しかし，あらかじめ各データを分類する方法を考慮する必要がある．ネットワーク上での位置が離れているピアの情報は伝播されにくいためスケーラビリティに優れているとは言い難い．また，RI はインデックスの更新頻度が高いため，ピアが頻繁にシステムへの参加，離脱を繰り返すと帯域幅消費量が多くなってしまうという欠点がある．APS では過去の問い合わせ結果を問い合わせの内容ごとに記録しこれを用いて問い合わせメッセージをルーティングする．また APS は帯域幅消費量を低減することに特化した分散インデキシングの方法であるが，各ピア上で処理されたオブジェクト毎にそのピアのインデックスにレコードを追加するため，インデックスのサイズは肥大化してしまう傾向がある．

過去の検索結果を利用して問い合わせメッセージのルーティングを決定する方法がある．分散データベースシステムにて各クライアントが問い合わせに対する結果をキャッシュし，それを利用することで次回以降の問い合わせに利用する方法がある [25, 24]．P2P システムでも検索結果をキャッシュする方法がある．P2P システムでは問い合わせメッセージを伝播する．これを利用する方法がいくつか考案されている [64, 8]．Freenet では過去の検索結果をキーワードをキーとして各ピア上にキャッシュし，次回以降の問い合わせ処理にて利用する．また P2P システムにて検索結果をキャッシュするとき，キャッシュレポジトリは有限であるとした場合，そのキャッシュすべきキーワードは刻一刻と増加するためページングを必要とする．そのための方法として Yang 等は Most Recently Used（MRU ; もっとも古いキーをページアウトとする），Least Recently Used （LRU ; 最後に参照された時刻が古いキーをページアウトする），Most Files Shared（MFS ; 保持しているファイル数が最も少ないピアに関するキーをページアウトする），Most Results（MR ; もっとも検索結果の数が少ないキーをページアウトする）の各方法にてキーのページングに関する方法を比較し検討した [80].

PP2P システム，HP2P システムおよび DHT を用いたシステムの性能を比較したものを表 2.1 に示す.

2.3.2 局所処理機能

局所処理機能ではデータの転送に関する機能を提供する．データの転送は，検索結果に基づいたデータ転送，実際にあるデータを必要とするピアとそのデータを保有しているピアとの間でのデータ転送だけではない．分散システムでは，冗長性，信頼性および耐久性の観点からデータをネットワーク上のある場所へキャッシュもしくはデータの複製を配置することを行うシステムは多い.

表 2.1 各 P2P システムでの性能比較

	PP2P	HP2P	DHTs
代表例	Gnutella Freenet	Napster KaZaA	Chord Tapestry
処理効率	△	○	◎
規模	△（10-100 万）	○（100 万）	◎（100 億）
自律性	◎	○	△
柔軟性	◎	△	△
問い合わせの表現力	○	○	△

分散システムでのリモートオブジェクトのキャッシング [6] はサーバとクライアント間の通信のコミュニケーションのオーバヘッドを低減するための方法としてリモートオブジェクトキャッシングがある．リモートオブジェクトキャッシングの例として CORBA[15] や Java RMI[29] 等がある．これらのシステムでのキャッシュの方法では，サーバサイド上にあるオリジナルのデータをクライアント上もしくはクライアントに最も近い位置に存在する他のサーバ上にそのデータの複製を配置する．この方法によって，クライアントとサーバ間での通信の帯域幅消費量を低減することが可能である．リモートオブジェクトの複製はエンタープライズ分散システム間ではセキュリティの面から不可能である場合がある [71]．Web プロキシサーバは Web 上のリソースを URL をキーとしてプロキシサーバ上のキャッシュテーブルに格納する [10, 13, 73]．また Web ページキャッシングでのキャッシュの整合性の管理に関する研究もある [58, 10, 13, 73]．Gray 等 [34] は Web キャッシュに有効期間（“Leases”）を設定し，これを基にキャッシュの整合性の管理をサーバ駆動にて行う研究を行った．サーバサイドでの動的な Web データキャッシングはサーバ側での駆動によるキャッシュの整合性管理と関連する [12, 45]．Yin 等は様々なサーバ駆動での Web キャッシュ整合性のためのプロトコルを調査し，スポーツイベントに関する Web サイトにてそれらのプロトコルの効果をレポートした [81]．また Web キャッシュをクライアントによる “pull” とサーバからの “push” の組み合わせによって行う方法もある [27]．

P2P システムを利用した Web キャッシングの仕組みとして，Coral[18]，FreeCache[30] 等がある．Coral ではサイト単位でデータをキャッシュし，FreeCache ではファイル単位でデータをキャッシュする．そのため Coral では最初の呼び出しに時間がかかってしまう欠点がある．しかしながら，サイト間でのリンクを頻繁に呼び出す状況下では Coral は

FreeCache よりも効率的に処理を行うことが可能である．Coral ではあるキャッシュサーバ内に同一サイトのデータが存在するため，同一のサイトのデータを他のキャッシュサーバにて探す必要がない．

分散データベースシステムでは帯域幅消費量の低減や信頼性向上のために複製を生成し，これを配置する．データベース中の各レコードに対して更新処理が行われると，そのレコードの複製をいかに更新するかという問題がある．その方法は以下の通りである [35].

- Eager Replication : 各トランザクションでの write を伴う操作ではその更新を各複製を保持しているノード間で同期をとりながら行う．
- Lazy Replication : write の操作を伴う各トランザクションではその操作を行ったノードでのレコードに対して行い，その後複製に対して同様の write 操作を各複製を保持しているノードにて行う．そのため Lazy Replication では非同期に更新処理が行われ，2 相コミットを行う．

Lazy Replication はさらに次の 2 種類に分類される．

- Lazy Group Replication : 各レコードに対して更新処理を行うことが可能である．そのため，レコードの所有権は各複製を保持している各ノードにある．
- Lazy Master Replication : 各レコードには所有者が決定されている．更新はそのレコードの所有者のみに許されている．

Jim Gray 等は上記の複製手法では Lazy Master Replication を用いた場合，最もデッドロックおよびトランザクションのアボートの割合が低いことを示している [35]．Hwang 等は Lazy Replication の方が Eager Replication よりも問い合わせ処理の応答時間に関して優れていることを示している [41].

CDN での各キャッシュの配置および更新では（1）ユーザの要求によって初めてキャッシュ，(2）コンテンツプロバイダサーバでのデータ追加または更新に応じて各エッジサーバへ配置，および（3）定期的に更新，の 3 タイプある．これらは想定されるアプリケーションでのデータの更新頻度やアクセス頻度に応じて使い分けることができる．CDN ではキャッシュをどこに配置するか，いくつキャッシュを配置するかということを制御できない．そのため不必要なキャッシュも生成してしまう可能性があり，複製更新に関する帯域幅消費量も余分に消費してしまう．

P2P システムでの複製配置は分散データベースシステム（DDBS）で従来から研究されてきたが，P2P システムでは以下の点が異なっている [62].

- P2P システムではシステム上の各ピアは動的であり，任意に接続，切断を行うことができるが，DDBS での各ノードは静的である.
- P2P システムでは決まったスキーマを必要することは少なく，各ピアは動的であるために，あるピアでのスキーマは共有データへ反映しにくい．DDBS では各ノードは静的であるため，共有スキーマの知識を各ノードで保有することができる.
- P2P システムでは全ノードが接続しているとは限らないため，検索結果はシステム全体の結果であることはありえない．DDBS ではほとんどの場合，システム全体での完全な検索結果の集合を返すことが期待される.
- P2P システムでは伝言（word-of-mouth）方法で問い合わせメッセージをルーティングする．DDBS では各データの位置は厳密に特定されるため，問い合わせメッセージは直接共有データを保有するノードへ転送される.

このことは，DDBS での複製配置の手法は直接 P2P システムに適応できないことを意味する.

複製配置法は以下のように分類できる.

- No-Replication: この方法では複製の生産は行わない.
- Full-Replication: 複製配置を行おうとするピアは，あるピア集合に所属する全ピア上に複製を配置する.
- Partial-Replication: 複製配置を行おうとするピアは，システム上のいくつかのピア上に複製を配置する.
- Data Trading[17]: 各ピアの保有するデータをオークション形式で，システム上の他のピアへ配置する．これは主にディジタルアーカイブシステムとして用いることを想定した方法である.

Sun 等は HP2P システム上での Full-Replication 及び Partial-Replication に関する研究を行っており [68]，各ピアが保持しているデータの情報をどの super peer に対してエントリするかという問題を扱っている．DHT を用いたシステムでは以下の通りである．P-Grid ではデータの配置場所が各データのハッシュ値に応じて決定される．そのため各

データを保持するピアのグループ内で Full-Replication が行われる．P-Grid でのデータ更新は Chen 等 [14] によって行われている．この方法では Lazy Group Replication でのデータ更新を行う．各ピアはそのピアが保持すべきデータに対する更新情報の検索と更新を行ったピアがそのグループに所属するピアにその更新情報を伝播させる 2 フェーズでデータ更新の処理を行う．Datta 等 [23] は Tapestry でのデータ配置に関する研究を行っている．Tapestry 上での各ピアに親子関係を与え，各データの複製は各ピアの親子もしくは兄弟ピアにデータを配置する方法および各ピアから距離が r ホップで到達できるピアの負荷状況を把握しており，そのうち最も負荷が少ないピアに対して複製を配置する方法を提案している．Freenet ではデータ転送を要求したピアからデータを保持しているピアの経路上でデータ転送を中継するピアにもそのデータを LRU（Least Recently Used）でキャッシュする．OceanStore[54] では，可用性，耐久性のためエントリされたデータは 16 分割され，それぞれの破片は複製される．

2.4　アプリケーション層

アプリケーション層で提供される機能はサービス層で提供されるルーティング機能または局所処理機能を利用することで提供される．アプリケーション層にて実現されるコミュニティアプリケーションの例として，Napster，Gnutella および Freenet に代表されるデータ交換システムが有名である．これ以外にも ICQ[43] に代表されるインスタントメッセンジャや Groove[37] に代表されるグループウェア等，様々なアプリケーションが存在する．Oceanstore[54] では，グループウェア，E-mail 機能，電子図書館や科学データ等の多量なデータを格納するためのデータストレージ及びセンサーストリーミングに代表されるストリーミング機能等をその適用として想定している．

コミュニティアプリケーションはそのままサービス層の機能を用いることは可能である．しかしながら，サービス層で提供される機能を補完し，コミュニティアプリケーションとサービス層との間をつなぐ機能を別途提供する場合もある．その例として情報検索（Information Retrieval）がある．

非構造 P2P システムでは全文検索，キーワード検索およびトピック検索等を行うことが可能であり，柔軟性のある問い合わせが可能である．HP2P システムではシステム内のデータの情報は各ピアによって super peer に送信される．そのため，データ情報の生成は各ピアによって行われる．非構造 P2P システムと HP2P システムでは一見同様の検索

結果を示すように思われる．しかしながら，HP2P システムシステムでは参加しているピアが保持しているデータ全体から検索結果を一度に返す．そのため，HP2P システムでは従来の Client-Server システムと同様の検索結果を得られる可能性が高い．一方，非構造 P2P システムではシステムの一定範囲内での検索結果を返し，各検索結果の転送には遅延が生じる．このことは検索結果を m 件返すことを目的とする問い合わせを行う場合，問い合わせメッセージのルーティングによって検索結果には差異が生じることを意味する．

DHT を用いた P2P システムでは，求めるデータに対して，正確なデータ名を基にハッシュ値を求め，これをキーとする問い合わせを行う．そのため非常に厳密な問い合わせになる．ファイル名検索をサポートしたアプリケーションであれば，正確なファイル名を知っていることが前提条件になる．例えば，“foo.txt” というファイルを検索するためには “foo.txt” を基にハッシュ値を求める必要があり，“foo” を基にハッシュ値を求めたとしても所望する “foo.txt” を検索結果として得ることはできない．そのため，柔軟な問い合わせを行うことが困難である．問い合わせの柔軟性を解決するために DHT を用いたシステム上での情報検索の研究も行われている．pSearch[72] は CAN 上にて VSM（vector space model）や LSI（Latent Semantic Indeing）を行うことを目標としている．全文検索を目的としており，ハッシュキーの生成には文書からいくつかの特徴語を抽出し，重み付けしたものを CAN 空間にエントリする．PIER プロジェクト（P2P Information Exchange & Retrieval）でも CAN を用いている [40]．キーワード検索をサポートするためファイル名を n-gram 分解し，各トライグラム対してハッシュ値を求め，エントリする [38]．例えば，“Beethovens 9th” というファイル名であれば “Bee”，“eet”，“eth”，“tho”，“hov”，“ove”，“ven” “ens”，“ns%”，“s%9”，“%9t”，“9th” の 12 個のトライグラムに分解し，それぞれに対してハッシュ値を求め，エントリする．n-gram を用いた場合，エントリの数が増大してしまう欠点がある．PlanetP[21, 20] では，ピア p がトピック t の文書をいくつ持っているかという情報と t の文書をシステム内の全ピアの内どのくらいのピアが 1 つ以上持っているかという情報にてピアをランキングする．

2.5 本研究の特色

本研究では，P2P システムでの問い合わせ処理の効率性の向上を目的とする．問い合わせの柔軟性の観点より非構造 PP2P システムを用いる．非構造 PP2P システムではピアごとに発見されやすいデータと発見されにくいデータが存在することになる．そのため

ピアごとに検索結果が変更することが可能になる．この機能を利用し，各ピアが所望するデータを提示されやすい仕組みを整えるために各ピアのにとって利益が高くなるようなピアを発見する仕組みおよびピア情報を伝播させる仕組みが必要になる．そこでどのピアに問い合わせメッセージを転送するかを判断するために問い合わせメッセージのルーティングを，ピア情報を格納，伝播させるために分散インデックスを，有益なデータをより発見しやすくするためにデータ配置を考慮する必要がある．非構造 PP2P システムの問い合わせ処理では上記の関連研究より以下の事項が求められる．

1. 帯域幅消費量の低減
2. 応答時間の向上
3. 問い合わせメッセージの伝播範囲の広域性

これらをサポートする仕組みとして本研究では，非構造 P2P ネットワーク上で実現可能な有益なピアを収集し格納する新たな分散インデックスを導入し，これを用いた問い合わせ処理を行う仕組みを整える．また動的にピアグループを構成しそれらのピアグループ内で Partial-Replication を行う．また，P2P システムでは複製を保持しているピアが確実にシステム内に存在するとは限らない．さらに，更新手続きの複雑さを低減させるため Lazy Master Replication を行う．

2.6　本章のまとめ

本章では前章にて示したシステムレイヤに基づき，それぞれの機能に関連する従来の分散システムと P2P システムにて提供，提案された内容と本研究にて提供する内容とを比較し，本研究の特色について述べた．

第 3 章

非構造 Peer-to-Peer システムの概要

3.1 はじめに

一般的に，P2P システムは多く役割や能力が等しいピア（peer; ノード）によって形成される．各ピアは任意の時刻にシステムへ参加（join）し，また任意の時刻にシステムから離脱（leave）することができる．各ピアは少数のピアと接続する．これらのピアを隣接ピア（neighbor）と呼ぶ．例えば，図 3.1 のピア A の隣接ピアは実線で接続されているピア，すなわちピア C および D である．各ピアは隣接ピアを通じてシステム内の他のピアからの情報を得ることができ，また隣接ピアを通じてそのピアから他のピアへ何らかの情報を伝播することが可能になる．隣接ピアから得られた情報を用いて，問い合わせ処理を行い，実際のデータ転送を行うことが可能になる．

ここで問題となることは，システム内の各ピアはどのピアとコミュニケーションを行うのか，どのようなコミュニケーションを行うことが可能なのかを設定する必要がある．一般的な P2P システムではピア間の接続状況やピア間でのコミュニケーションは 2 章で与えた P2P システムレイヤにおけるシステムコア層の機能を用いて行われる．すなわち，この層では対象とする P2P システムでのネットワークトポロジ及び，ピア間でのコミュニケーションの媒介となるセッションを提供する．本章では本研究が対象とする P2P システムでのシステムコア層の機能および，本研究にて用いる P2P ネットワークの特徴を他の P2P システムでの方法を交えつつ述べる．

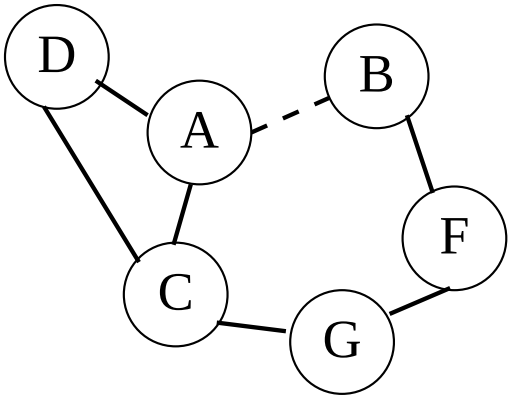


図 3.1　P2P ネットワークの例

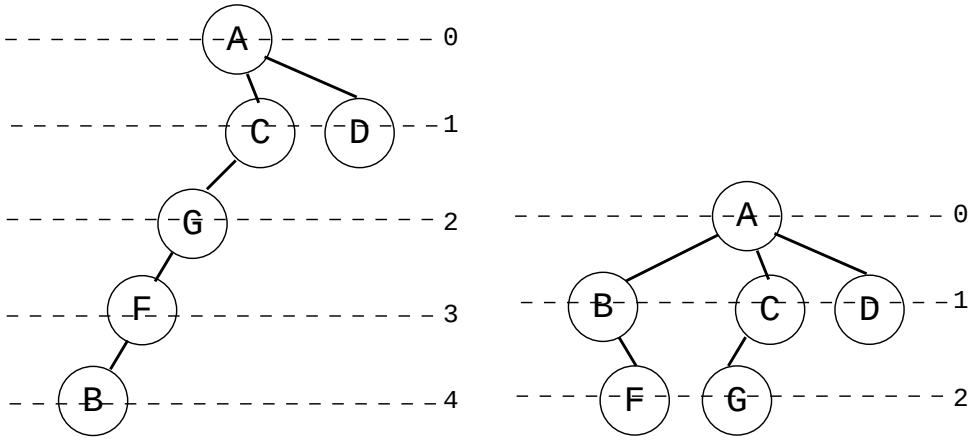


図 3.2　ピア A を根とする木構造の例

3.2　ネットワークトポトジ

本研究において、P2P ネットワークは木構造として表現する．各ピアはそのピアを根とする木構造としてネットワークを捉える．しかしながら，図 3.1 においてピア A を根とする木を構成する場合，サイクルが存在するため，木を構成することができない．そこで，A を根とする木を構成するために，A から任意のピアまでの最短経路を示す辺以外を除去する．図 3.1 の例であれば，図 3.2（左）に示すように CD 間の辺を除去することで除去することで木を構成することが可能になる．このとき，実際のネットワークにて CD 間の接続の切断を行うわけではない．ここで，任意のピア間の最短経路を示すために距離の決定する方法を示す必要がある．本研究では，ピア x を根とする木ネットワークにおいてピア x から別のピア y までの間の距離は x から y までの最短路長によって定義する．あ

るピア x に対してピア $i \rightarrow y \neq x$ が隣接するならば xi 間の $d(x,i) = 1$ と定義する．図 3.2（左）において，ピア A とピア C は隣接しているため $d(A,C) = 1$，AB 間の距離は 4 になる．各ピアはそれぞれ隣接するピアの最大数（ファンアウト数）が設定されており，これを F とする．各ピアはそのピアと隣接するピアに対して枝を形成する．この枝を短距離リンク（short link）と呼ぶことにする．

また，隣接するピアより他のピアの情報を得る．このとき，隣接以外の距離が 2 以上のピアに対して動的に枝を形成する．これを長距離リンク（long link）と呼ぶことにする．各ピアは他のピアのうち，m 個のピアに対して長距離リンクを形成する．長距離リンクはその対象となるピアを動的に変更することも可能である．各ピアがシステム内に存在する他のピアの情報を格納するインデックスを保持し，これを維持，管理する．各ピアは他のピアとの接続（join），隣接ピアからピアに関するインデックス更新（index update）が行われるときにシステム内の他のピア情報の取得または更新を行う．

P2P システムでは任意のピア間の接続状況は変化し続ける．ネットワークのダイナミクスに従ってネットワークのトポロジは変化するため，任意のピア間の距離も変動する．図 3.1 にて AB 間にて接続が行われた場合，図 3.2（左）から図 3.2（右）のように A が認識している P2P ネットワークの状態は変化する．AB 間の接続によって，AF 間の距離は 3 から 2 になる．

3.3 長距離リンクの指標

本研究で提案する P2P システムでは各ピアは隣接していないピアに長距離リンクを形成する．そのためには，システムに参加しているピアの情報を取得する必要がある．そこで各ピアはシステム内の他のピア情報を隣接ピアより取得する．ここで得られたピア情報を基に長距離リンクを形成することができるが，得られたピアの内どのピアへ長距離リンクを形成するかを決定する必要がある．本研究では長距離リンクを形成するための指標を**ピアの “有用度”（“usefulness” of peer）**として与える．ピアの “有用度” はそのピアが保有している文書によって決定する．まず，**文書の “有用度”（“usefulness” of document）**を決定する．本研究では，各文書の “有用度” はその文書への参照頻度と最終更新時刻からの経過時間によって決定する．各文書への参照頻度が高ければそれだけ多くのユーザによって必要とされたことを示す．しかしながら，その文書への参照は過去に多かった事象かもしれないし，現在はその文書が多くのユーザによって必要とされないか

もしれない．そこで，文書の更新からの経過時間に従ってその文書の “有用度” を低下させる．すなわち，その文書の新規性も考慮する．本研究では，ある文書への参照回数が多く，さらに新規であるものほど多くのユーザに必要とされると仮定する．ある文書へのアクセスはその文書の “有用度” を上げるが，時間の経過とともにその文書の “有用度” は低下する．これを以下の（3.1）式で表す.

$$E_{doc}(d) \equiv \frac{d_{ref}+1}{\log(d_{et})+1} \tag{3.1}$$

ここで d_{ref} を d の参照回数，d_{et} を d を公開時刻もしくは最終更新時刻からの経過時間とする．文書の “有用度” は文書の新規性を考慮している点に関して *Forgetting Factors* と関連する．石川等 [44] はオンライン上での文書クラスタの手法である F^2ICM(Forgetting-Factor-based Incremental Clustering Method) を提案している．文書間の類似性を計算するために，F^2ICM では時間経過と共に全文書の重みを減少させる Forgetting Factors を取り入れている．

次にピア p の “有用度” は p が保有する文書の “有用度” の総和によって決定することにした．これを（3.2）式に示す.

$$E_{peer}(p) \equiv \sum_{d \in D(p)} E_{doc}(d) \tag{3.2}$$

ここで，$D(p)$ はピア p の保有する文書の集合とする．$E_{peer}(p)$ をそのまま用いた場合，ネットワーク上の “有用度” が高い特定のピアに対して問い合わせメッセージの集中が起こる危険性がある．そこで，ネットワークでの位置に依存した “有用度” を導入し，ピア p_1 からピア p_2 へ渡される p_1 の “有用度” を以下のように定めた.

$$ER_{p_1}(p_1,p_2) = E_{peer}(p_1) + \sum_{p \in \mathcal{P}(p_1)-\{p_2\}} \frac{ER_{p_1}(p,p_1)}{F} \tag{3.3}$$

ここで $\mathcal{P}(p_1)$ は p_1 の隣接ピア集合，F は各ピアの隣接ピア数の最大値（ファンアウト数）である．この式によって p_1 と p_2 間の距離が遠くなるほど，A に渡された B の “有用度” は低く設定される．$ER_{p_1}(p,p_1)$ は p_1 のインデックス内に格納された p の “有用度” であり，DI の更新手続きに従って適宜更新されていくが，更新手続きの過程でタイムラグが生じることもある.

3.4 セッションプロトコル

本研究において，すべてのピア間のコミュニケーションはセッションを介して行われる．セッションはその利用によって，接続セッション（ピアのネットワーク上での状況の変化に関する通信），問い合わせセッション（問い合わせに関する通信），転送セッション（データの転送に関する通信）のいずれにか分類される．ここでは，各セッションでのピア間のコミュニケーションの方法を示す.

3.4.1 接続セッションプロトコル

接続セッションにおいて各ピアは他のピアの情報を伝播するために用いる．このセッションでは以下の作業を各ピアは行う.

- P2P ネットワークへの参加
- P2P ネットワークからの離脱
- ピア間の接続
- ピア間の切断

各ピアはシステムへ参加するためにはあらかじめ 1 つ以上の他のピア情報を知る必要がある．そのための方法の 1 つとして Point-to-Point という方法を挙げることができる．Point-to-Point では，各ピアはシステムの参加に際し，事前に P2P システム上の全ピアの情報を確保しており，それらのピアとのみ接続を行う方法である．明示的な Point-to-Point はピアを発見するための仕組みを不要にする方法である．しかしながら，P2P システムではシステムへ参加しているピアは動的に変化しつづけるため，新たなピアの参加に対して Point-to-Point の方法では対応することが困難である．そのため，動的にシステム上の他のピアの存在を発見することが可能な仕組みが必要となる．そこで，システム上の各ピアが動的に他のピアを発見する方法として以下の方法がある [69].

- ディレクトリサービスモデル
- ネットワークモデル
- マルチキャストモデル

ディレクトリサービスモデルの例として，OpenNap[63] や KaZaA[48] 等の HP2P システムではシステムに参加するときにログイン用サーバが用意されており，このサーバから他のピアの位置情報を取得する．そのため，各ピアがシステムへ参加するためにはログインサーバの位置情報を知ることだけでよい．この方法では各ピアはシステムの参加時にそのピアの位置情報をログインサーバへ送信する．その後，システムへ参加しているピアは super peer へそのピアの位置情報とともにそのピアが共有しているデータの情報を送信する．OpenNap ではログインサーバは super peer を兼ねている．そのため，上記のシステムへの参加の手順を 1 度に行うことができる．KaZaA ではログインサーバと super peer は異なる．そのため，ログインサーバから super peer の位置情報を得た後に super peer へそのピアの位置情報および共有されたデータの情報を送信する．ピアはシステムへ参加している間，super peer へ定期的に ping メッセージを送信する．そのため super peer は，一定期間，あるピアからの ping メッセージが送信されないピアはシステムから離脱している状態であると判断することが可能になる．しかしながら，この方法では各ピアがシステムへ参加するたびにログインサーバへアクセスしなければならない．そのため，ログインサーバの管理コストが別途必要になる．

JXTA[46] ではサブネットでのマルチキャストによって他のピアの存在を知ることができる．この方法ではディレクトリサービスモデルやネットワークサービスモデルとは異なり，システムへの参加に際し，システムへ参加している他のピア情報をあらかじめ知っておく必要はない．システムへ参加するピアは他のピアからの接続要求に応じられるように特定のポートを開くことでシステムへ参加の状態になり，そのポートを閉じることでシステムからの離脱の状態にすることができる．しかしながら，この方法を実現するためには物理的なネットワークでの問題を解決する必要がある．サブネット内でのマルチキャストを行う環境を整えるためには，ルータがこれをサポートしなければならない．

ネットワークモデルの例として，Gnutella や Freenet がある．多くの PP2P システムではこの方法が用いられている．Gnutella や Freenet では初期ピアリストをあらかじめ用意しており，最初はこのリストにあるピアへ接続する．Freenet や Gnutella では初期ピアリストを用いてシステムに参加し，他のピアと接続できれば，接続セッションや問い合わせセッションを介して他のピアの存在を新たに知ることができ，各ピアは独自にピアリストへ他のピアの位置情報を格納することが可能になる．Gnutella のクローンでは，初期ピアリストはそのアプリケーションとともに配布されていた．しかしながら，この方

法ではそのピアリスト内のピアへの負荷がシステムへ参加するピア数の増加とともに増大してしまい，システム全体のパフォーマンスが低下してしまう．そのために，Freenet では最近システムへ参加したピアの位置情報を記したピアリストを配布するためのサーバを用意し，このサーバへアクセスすることでピア情報の収集を可能にしている P2P アプリケーションも多い．ピアリストは各ピアによって保持され，管理される．各ピアはそれぞれが保持しているピアリストを用いてピア間の接続を行うことができる．ピア間での接続において，互いに機知であるピア集合を送信する機能を付加することで，それぞれのピアは新たに他のピアの存在を知ることができる．問い合わせ処理時に，問い合わせメッセージを送信する対象となるピアは問い合わせ処理の最中にも発見することができ，反対に，問い合わせメッセージを受信するピアは送信したピアの存在を新たに知ることも可能である．各ピア間の接続状況を検知するために，システムに参加しているピアは定期的に隣接ピアへ ping/pomg メッセージを送信する．このメッセージをやりとりを行えない隣接ピアはシステムから離脱したと判断し，そのピアは隣接ピア表もしくはピアインデックスから離脱した隣接ピア情報を削除する．

本研究では，ディレクトリサービスモデルでのログインサーバの管理コストの問題ややマルチキャストモデルでのルータ設定の問題の点から，ネットワークモデルを想定している．長距離リンク形成のために，本研究でのピア間の接続は以下のように行う．ピア間の接続ではそれぞれのピアがピア情報を交換し，このピア間で短距離リンクを形成する．さらにピア情報を受け取ったピアの中から上位 m 個の “有用度” が高いピアに対して長距離リンクを形成する．ピア間での切断は互いに受け取ったピア情報を所有のインデックス内から削除し，ピア間の短距離リンクを切断する．本研究において，インデックスから削除したピアに対して長距離リンクを形成している場合，その長距離リンクも切断する．そして，新たな長距離リンクの形成のためにインデックス内の “有用” なピアに対して長距離リンクを形成する．

3.4.2 問い合わせセッションプロトコル

問い合わせ処理ではあらかじめ与えられた停止条件を設定しておく必要がある．本研究では問い合わせの停止条件として，問い合わせの内容を含む文書を一定以上集めることであるとした．問い合わせ処理を行う場合，その問い合わせを行ったピアはその問い合わせがローカルで停止するかどうかを判定する．問い合わせが停止しない場合，問い合わ

せメッセージを他のピアへ転送させる．本研究では，各ピアは問い合わせメッセージを転送させるとき，短距離リンクもしくは長距離リンクを通じて転送させる．問い合わせメッセージを受信したピアはそのローカルでその問い合わせに対する処理をローカルで行い，問い合わせメッセージを転送したピアへ検索結果とともに他のピア情報を付加したメッセージを送信する．検索結果を受け取ったピアは問い合わせが停止するかどうかを判断する．もし停止しなければ以前より管理しているピア情報に新たに受け取ったピア情報を含めたピア集合の内，まだその問い合わせを送信していないピアを選び，そのピアに対して新たに長距離リンクを形成し，問い合わせメッセージを転送する．以上の処理をリンクを長距離形成する対象となるピアが存在しなくなるか，問い合わせが停止条件を満たすまで繰り返す．

検索結果を得たピアはその内容に応じて文書の転送を依頼し，文書のやりとりを行う．このとき文書を所有しているピアと転送要求を行っているピアの間で直接文書の転送を行う．

3.4.3　転送セッションプロトコル

各ピアは転送セッションを介してデータの転送を行う．データの転送は各ピアは検索結果を基にそのピアが所望するデータを所有しているピアへデータの転送要求のメッセージを送信し，転送可能であるならば，そのデータの転送を開始する．ここでデータの転送可能のための条件として，データの転送要求先のピアが以下の状況下にある場合を指す．

- システムに参加している．
- 単位時間あたりでのデータのアップロード可能な容量を超えていない．

また，データの転送要求したピアがデータ転送要求を行う条件としては，同時のデータのダウンロード容量を超えていないことである．

データの転送は上記のようなネットワーク上の各ピアが特定のデータを要求することで生じるが，複製されたデータの配置によっても引き起こされる．データ複製の詳細は 5 章にて示すが，データ複製の際でのデータ転送も転送セッションを介して行われている．この場合，先ほどのデータ転送の場合と同様の条件が求められる．

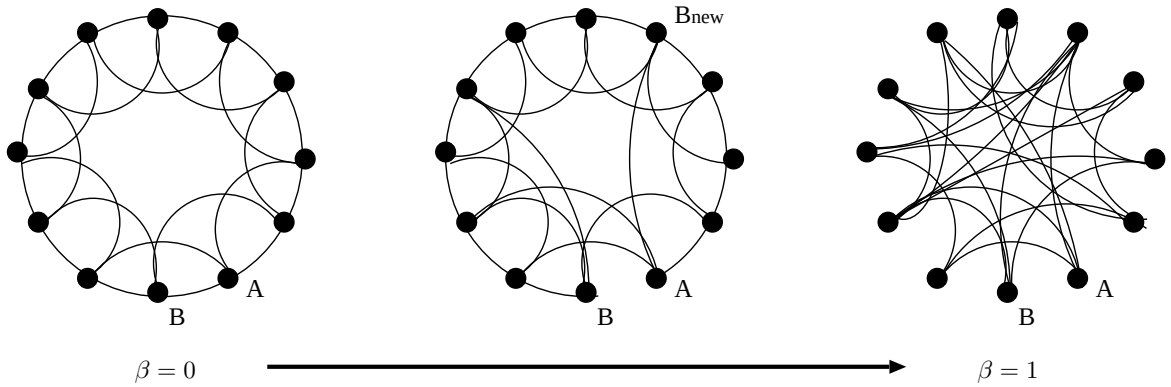
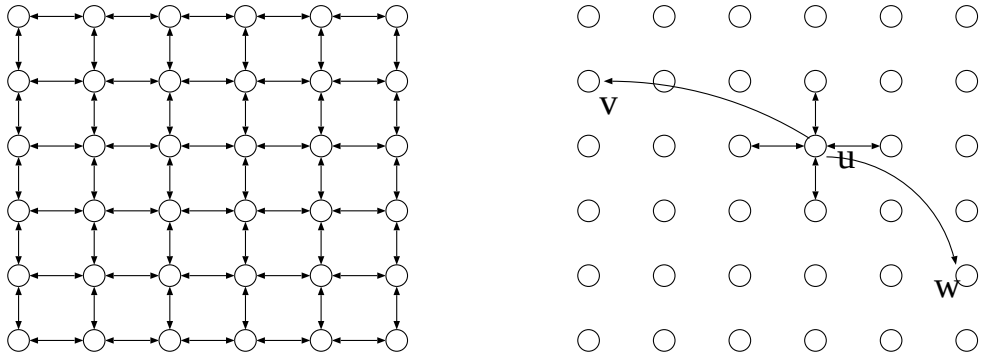


図 3.3　β グラフ：Watts 等のスモールワールド



（a）Kleinberg が用いた 2 次元ネットワーク　（b）短距離リンクと長距離リンク

図 3.4　Kleinberg のスモールワールド

3.5　本研究での P2P ネットワークモデルに関する考察

本研究で想定する P2P ネットワークは非構造である．そのため，各ピアはネットワークのどの位置に存在しなければならないという制約はない．各ピアは隣接ピアを自由に選択することが可能であり，自己組織化を行うため，高度にクラスタ化される可能性がある．しかしながら，ピア間のリンクは隣接ピア以外のピアに対して長距離リンクを形成するため，任意のピア間のパスの長さは短くなる可能性が高い．そのため，「ピア数が多い P2P ネットワークであっても，任意のピア間の平均パス長は短くすることができるネットワーク」を構成することが可能になる．この特徴を有するネットワークはスモールワールド（Small-World）と呼ばれる．

この「スモールワールド」という言葉は元々，社会学の分野において，友人・知人関係のネットワークに対して使われていた用語であり，社会心理学者である S. Milgram の実

験によってスモールワールド現象は広く知られるようになった．この実験では，ランダムに選んだネブラスカ州オハマ等の被験者数百人に手紙を直接は知らないボストン在住のとある人へその手紙を届けるように依頼した．その結果，届いた手紙は平均 6 人を経由して届いている，という結果を得た[*1][60]．この実験より，“米国のいかなる人も 6 人の向こう側にいる” という結果は “six-degrees of seperation” として有名になった．

その後 1998 年に D. Watts 等は，スモールワールドとはネットワーク上で隣接ノードは同じ隣接ノードを有する確率が高いのにもかかわらず，任意のノード間の平均パス長は短いという特徴を持つ，として定義した．Watts 等が用いた 1 次元リングの β グラフと呼ばれるネットワークモデルを図 3.3 に示す．このグラフ上の各ノードは近傍のノードに対して短距離リンクを保有するレギュラネットワーク（図 3.3 左）に対してある確率で各ノードのリンクをランダムに他のノードにつなぎ代える操作を行う．この確率が 1 のとき，ランダムグラフになる（図 3.3 右）．ここでノード A はその近傍にノード B を持つとする．ノード間のリンクのつなぎ換えの確率が 0 よりも大きい場合，A は B に対するリンクを他のノード B_{new} に対してリンクを接続する可能性がある．ここで，近傍以外のノードに対するリンクが長距離リンクになる．スモールワールドはノード間のリンクのつなぎ換えの確率が 0 から 1 の間であるときに，すなわちレギュラグラフとランダムグラフの中間に現れる．レギュラグラフでは各ノードは近傍ノードに対して必ず短距離リンクを形成する．そのため，あるノードの近傍のノード間で短距離リンクを形成してる確率がランダムグラフよりも高い．また，ランダムグラフでは長距離リンクを形成するため，レギュラグラフよりも任意のノード間の最短パス長が短くなる可能性が高い．Watts 等のスモールワールドはレギュラグラフでのクラスタ係数とランダムグラフでの任意のノード間の平均最短パス長の 2 つの特徴を兼ねたネットワークである [76]．Watts 等はスモールワールドでの特徴を決定する要素として

- L（characteristic path length）: グラフ中のすべてのノードの組についての最短パス長の平均.
- C（clustering coefficient）: 任意のノード v に関して，v の隣接ノードの集合を N_v とする．N_v 内での隣接するノード対の割合を NN_v とするとき，$C_v = \frac{|NN_v|}{|Nc|C_c}$ と定義する．この C_v の全ノードでの平均.

[*1] 実際にボストン在住のとある人の元に行き着いた手紙は全体の 30% 程度（42/160）で，その 30% の到達成功事例を分析したところ，平均 6 回の仲介があった.

とした [77]．この L と C を用いて Watts 等はスモールワールドは以下の特徴を持つグラフであると定義した [75]：スモールワールドはノード数 N が大きな値であり，疎に結合したグラフである．すなわち $1 \ll k_{max} \ll N$ （ここで k_{max} は各ノードからのリンク数の最大値）である．また $L \approx L_{rand}$ および $C \gg C_{rand}$ （L_{rand}，C_{rand} はそれぞれランダムグラフでの L および C）をスモールワールドは満たす．

Amaral 等は長距離リンクがどのノードに対して形成しているかによってスモールワールドを 3 つのタイプがあることを示した [5]．

- scale-free ネットワーク：極端に多くのリンクを持つノードが存在する．これはべき分布（power-law）に従う．すなわち，リンク数が k 本であるノードの割合は $P(k) \approx k^{-r}$（ここで r は定数）に従う．
- broad-free ネットワーク：$P(k) \approx k^{-r}$ に基本的に従うが，ある k 以上であるノードは存在しない．
- single-free ネットワーク：各ノードのリンク数は k を平均値であるようなポアソン分布 $P(k) \approx e^{-k}$ に従う．極端に k が多いノードは現れない．

スモールワールドではグラフ中の任意のノードをグラフから除いた場合，L にはあまり影響がないという特徴を持っているが，scale-free ネットワークである場合，特定のノードの除去に対して，グラフへの影響が非常に大きいことも知られている [4]．本研究での P2P ネットワークは，ピアの “有用度” は位置に依存した値を伝播するため，特定のピアに対して圧倒的に多くのリンクが集中することはない．そのため broad-free ネットワークになる．特定のピアをシステムから離脱させたとしても，その影響を小さくすることが可能である．

Kleinberg は図 3.4 に示すような $n \cdot n$ の 2 次元のグリッドを用いて任意のノード間でのメッセージの転送ホップ数の平均に関して調べた [50, 49, 51]．各ノードは $\{(i,j) : i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,n\}\}$ としてネットワーク上の位置によって識別される．また，2 つのノード (i,j) と (k,l) 間の距離は $d((i,j),(k,l)) = |k-i| + |l-j|$ として “lattice step” として定義されている．また，あるノードとの距離が p 以下であるノードをそのノードの近傍として定義する．また，各ノードは近傍以外のノードに対して q 本の長距離リンクを保持する．ノード u がノード v に対して長距離リンクを保持する確率を $[d(u,v)]^{-r}$ に比例させる．ここで r は 0 以上の定数である．ここで，$p = q = 1$，$r = 2$

のとき任意の分散アルゴリズム A の任意のノードからノードへのメッセージの転送のためのホップ数は $O(\log^2 n)$ であることを示した．

上記より，Watts 等のスモールワールドでは各ノードがネットワーク上のどのノードに対して長距離リンク形成するかはランダムに決定されるが，Kleinberg のスモールワールドではノード間の距離に応じて決定されることがわかる．Milgram の実験では，手紙を受け取った各個人がその手紙の転送先を選択するとき，転送先の候補の中で職業や友人関係等の概念を用いたと考えられる．これらの概念は個人間の “距離” を見積もるための要素であり，自分と他者とを区別する要素でもあると考えることが可能であろう．Kleinberg は，Milgram の実験にて手紙を受け取った個人がいかにして次の手紙の転送先を決定していたかということに着目し，長距離リンクの形成を踏まえたスモールワールドモデルを示した，と考えることができる．ノード間の距離の概念を用いた長距離リンクの形成を行うのが Kleinberg のスモールワールドである．これに対し，Watts のスモールワールドである β モデルではノード間の距離を考慮していない．Watts 等のスモールワールドグラフのような座標のような距離的な概念が存在しないモデルにて探索を行う場合，問い合わせメッセージは基点を参照しながら転送することは困難である．Kleinberg は Kleinberg のスモールワールドモデルにて，長距離リンクの形成をランダムに行う場合（$r = 0$），すなわち Watts 等が用いた β モデルの 2 次元版の場合，問い合わせメッセージの期待されるホップ数は $O(n^{\frac{2}{3}})$ であることを示している．従って，任意のノード間でのメッセージの期待される転送時間はノード数に対して指数関数的に増加する [49, 51]．このとき，遠くに位置するノードに対する長距離リンクが多く形成されるかもしれないが，各ノードはどのリンクが目的地に対して最も有効であるリンクであるかをを特定することが困難になる．また，r の値が増大するにつれ，近くに位置するノードに対してのみ長距離リンクを形成することになる．Kleinberg は，$0 \leq r < 2$ のとき，Watts 等の β モデルと同様にメッセージ転送に対して有効なリンクが存在するのにも関わらず見つけることが不可能であり，$r > 2$ のとき，メッセージ転送の短縮に有効なリンクは存在せず，唯一 $r = 2$ のとき，目的地へのメッセージ転送に対して最も有効であると思われるノードを特定することが可能であり，このときの任意のノード間のパスの平均長が $O(\log^2 n)$ であることを示している．

Kleinberg はさらに木構造での場合に関しても調べた．これは基本的には 2 次元グリッドの場合と同様の設定を行っている．木の各ノード間の距離はノード間の深さによって定

義する．全ノード数 N で構成される $b-ary$ 木（各ノードの子ノードの数は b 個以下であるような木）を仮定する．ノード u がノード v への長距離リンクを保持する確率が d^{-1} に比例する場合，$u \rightarrow v$ でのメッセージの転送のためのホップ数は $O(\log N)$ であることを示した [52].

本研究にて想定しているネットワークにおいて，長距離リンクの対象となるピアは接続セッション以外にも問い合わせセッションを介して集められる．また，各ピアのファンアウト数は決定されるため各ピアの隣接ピア数は F 以下である．この条件によって，本研究で想定している P2P ネットワークにて，長距離リンクの対象となるピアが d^{-1} に比例すると仮定すれば，任意のピアからピアへ問い合わせメッセージを転送するとき，その期待されるホップ数は $O(\log N)$ になる．本研究で用いる P2P ネットワークと Small-World 性に関する議論の詳細は 6.4 で改めて行うことにする．

3.6 本章のまとめ

本章では，本研究でのシステムコア層での機能を示すため，想定する P2P ネットワークでのトポロジの詳細および各ピア間での各セッションでのプロトコルを示した．また本研究の P2P ネットワークでは，ネットワーク上の各ピアはそのピアの隣接ピア以外のピアに対してリンク（長距離リンク）を形成することを可能にしている．各ピアが長距離リンクを形成するための指標として，本研究ではピアの “有用度” を用いることにした．

第 4 章

問い合わせメッセージのルーティング

4.1 はじめに

この章では本研究で提案する Direct Index（DI）を用いた P2P システムでの接続セッションおよび問い合わせセッションについて述べる．また，DI の基本である *Normal DI*（*NDI*）では各ピアは，ピアの UID，ピアの “有用度” およびピアの位置等をピアの情報として格納する．DI を基に各ピアはピア情報を伝播し，このメッセージを他のピアは受け取ることで動的にピアを発見することが可能になる．動的に他のピアの情報を各ピアはその DI に格納し，静的ではなく動的に長距離リンクを形成することが可能になる．

実際には，NDI では各ピアの問い合わせ処理を効率的に行うような長距離リンクを形成することが可能になるとは限らない．問い合わせ処理の目的を，あるトピックに属する検索結果を一定以上集めることであると仮定する．この問い合わせ処理を行う場合，指定されたトピックに関する文書を所有するピア，またはそれらのピアを知っているピアに対して問い合わせメッセージを送信する方が望ましいと考えられる．しかしながら，NDI 内には文書に関する情報が少ない．そのため，NDI を用いた場合では特定のトピックの文書を保有するピアを発見することは困難になる．NDI をそのまま用いるのではなく，問い合わせ処理の目的に応じて NDI を拡張して用いるべきである．

そこで，トピック検索を可能にするために，NDI にトピックに関する情報を格納するように拡張した *Topic-based DI*（*TDI*）を導入する．本研究では文書のトピック情報として，トピックの “有用度”（topics of “usefulness” ）を用い，各ピアは TDI 内にトピックの “有用度” を格納する．TDI と用いた場合，長距離リンクを形成するときの指標として，

表 4.1　ピア A の Direct Index

UID	“usefulness”	location
A	6	local
I	60	D
J	45	D
F	21	B
B	18	neighbor
C	12	neighbor
G	6	C

各ピアはトピックの “有用度” を用いる．これにより，NDI をそのまま用いた場合よりも，効率的な問い合わせ処理の目的に応じた長距離リンクの形成が可能になると考えている．

P2P システムでのルーティングに関する各手法の目的を比較したものを図 4.1 に示す．Gnutella では問い合わせメッセージをフラッディングさせるため，システム全体での問い合わせメッセージ数は非常に膨大な数になってしまう．そのため，従来より PP2P システムでは帯域幅消費量の低減が重要視されている．他方 DHT を用いたシステムでの問い合わせ処理は構造化されたネットワークを用いることで帯域幅消費量の低減だけではなく，応答時間の向上に対しても積極的に対応している．従来の PP2P システムでは問い合わせ処理の応答時間の向上に対応した仕組みを提供する研究はあまり行われていない．DI は非構造 P2P システムで用いることが可能なインデックスであり，DI を用いたルーティングは帯域幅消費量の低減だけではなく，応答時間の向上も実現することが可能である．

本章では，システム内の各ピアが如何に他のピア情報を収集し，ピア情報を DI に格納し，維持，管理するかについて述べる．また，DI の基本である NDI および拡張 DI である TDI を用いた問い合わせ処理について述べる．

4.2　Direct Index

この節では DI の基本である *Normal DI*（*NDI*）について述べる．各ピア p は隣接するピア p' から “有用” な（“有用度” の高い）ピアの情報をピアの集合 $rec(p' \rightarrow p)$ として受け取ることによってインデックスを取得・更新する．インデックス情報の受取りは接続セッションで行われる．“有用” なピアの情報は各ピアごとに “有用度” に基づいてラン

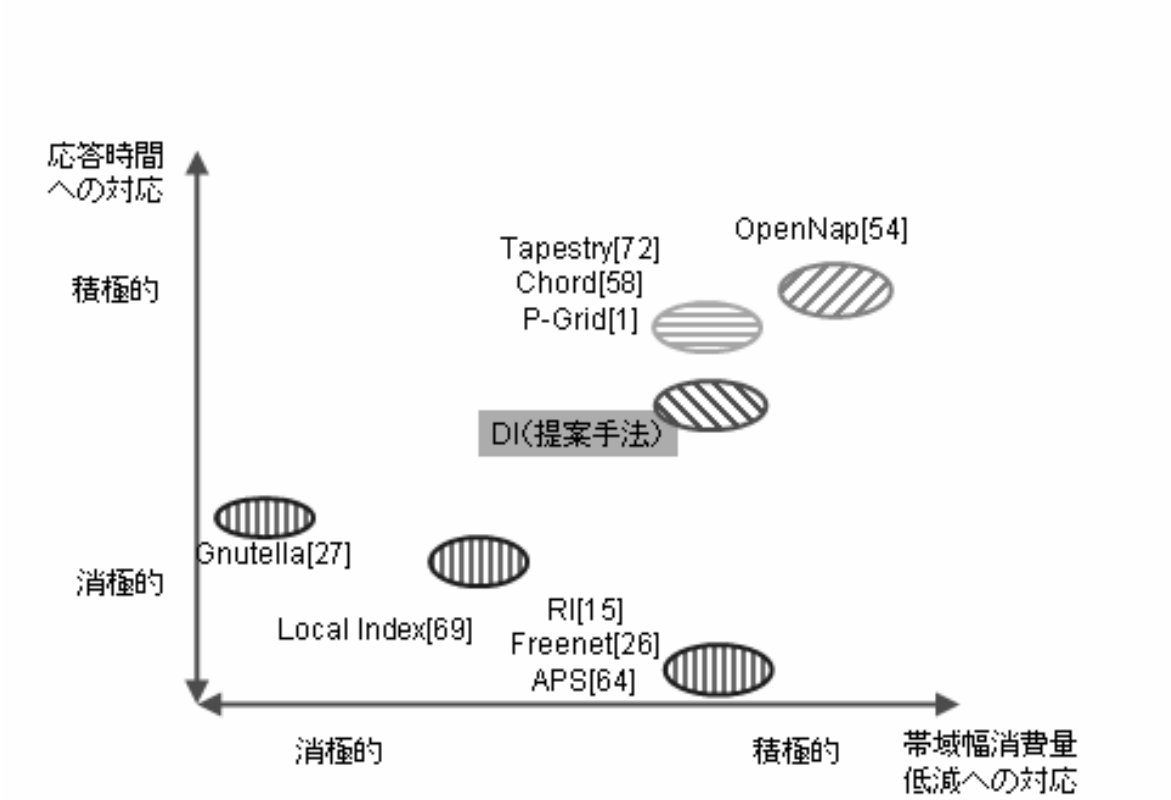


図 4.1 P2P システムでのルーティングに関する各手法の比較

キングされて格納されており，ピア p が問い合わせを行う場合は，そのうちの上位 n 個のピアに対して問い合わせを行う．問い合わせは，結果に満足するまで繰り返す．このとき，問い合わせを行ったピアから結果とともにピアの “有用度” の情報を検索結果と合わせて取得しインデックスを更新する．インデックスが更新されるとその情報が隣接ピアに伝播する．

例として図 4.2 のネットワークを考える．この図において各丸に囲まれたものをピア，そこに付随してある数値をそのピアの “有用度”，各ピア間の枝はそのピア間で接続の関係にあることを示す．このときピア A のインデックスは表 4.1 のように表される．このインデックスを *Direct Index*（*DI*）と呼ぶことにする．ピアの “有用度” は式（3.3）を用いて計算する．DI はピア情報を格納する．このピア情報はピアのネットワーク上の識別名，“有用度”，位置を 1 レコードとして格納される．ここでの位置はこの DI を保有するピアからどの方向に存在するかという情報である．この値は *local*（DI 保有のピア），*neighbor*（隣接ピア），*UID*（隣接ピアの方向に存在）のいずれかとなる．各ピアは DI を用いるこ

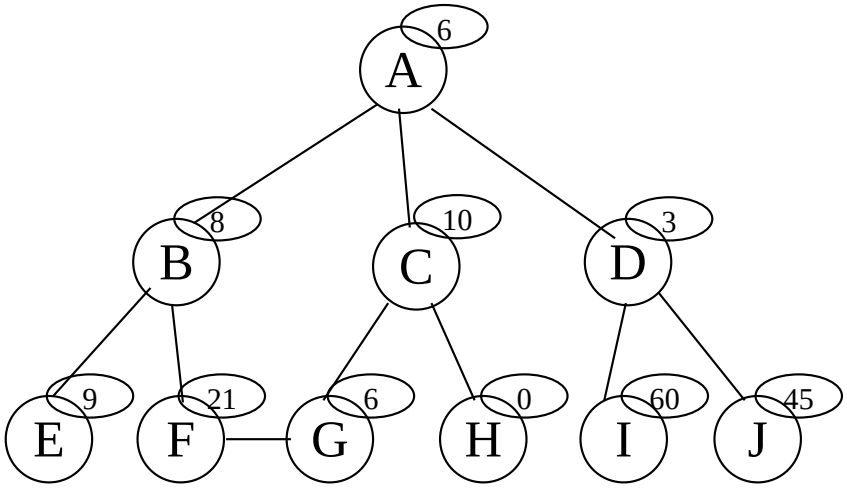


図 4.2　Direct Index の例

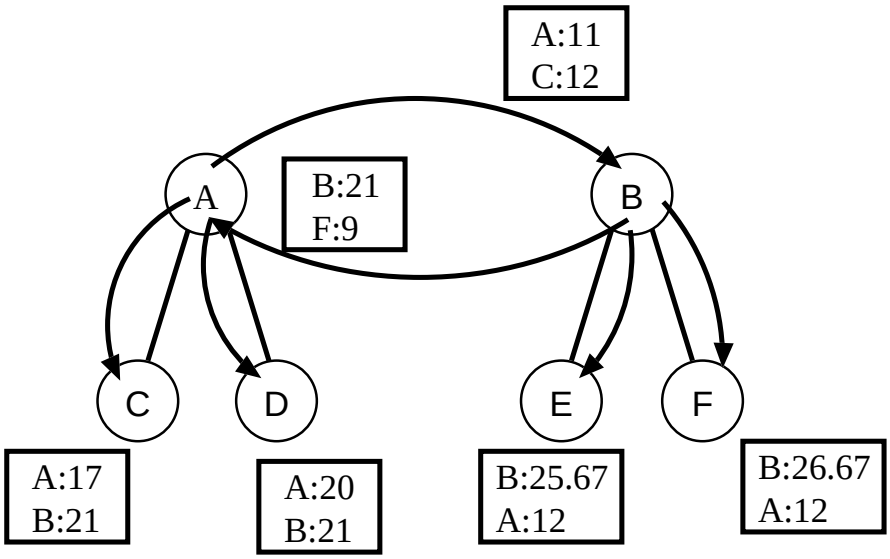


図 4.3　ピア間の接続の例

とで “有用度” の高いピアの情報を得ることが可能になり，それらのピアに対して直接問い合わせメッセージを送信することが可能になる．

4.2.1　接続セッション

NDI を用いる場合，各ピアはそのピアの DI 内に UID（ピアの識別子），“usefulness”（ピアの “有用度”）および location（そのピアが位置する方向）をピアの情報として格納する．接続セッションではピア間でのインデックス更新メッセージの伝播を行う．イン

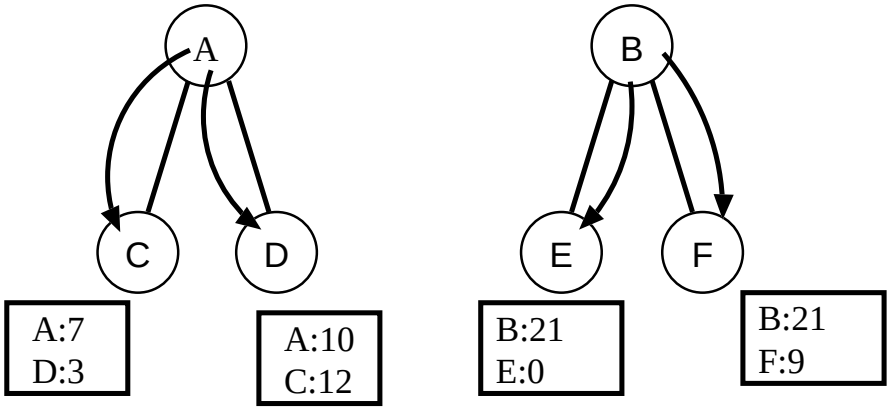


図 4.4 ピア間の切断の例

デックス更新メッセージによって各ピアは他のピアの情報を収集し，維持，管理する．問い合わせを行うとき，各ピアは DI に格納されているピアの “有用度” を利用する．そのため，DI を用いた問い合わせを行うためには DI の管理は重要である．

インデックス更新

インデックスの更新（index update）が必要な状況になると，ピア p は式（3.3）を計算する．ここで p_i は p の隣接ピアとする．この値と DI に格納されている各ピアのうちピアの “有用度” が上位 m 個を p が推奨するピア集合 $rec(p \rightarrow p_i)$ とする．$rec(p \rightarrow p_i)$ と p の P2P システム上での識別子を一組としたものをインデックス更新メッセージ $index\_update(message\_id, rec(p \rightarrow p_i))$ とする．ここで $message\_id$ はこのインデックス更新メッセージの識別子を示す．このメッセージを受け取ったピア p_i はその DI を更新する．さらに p_i は更新が必要になれば同様に更新の手続きを行う．

各ピアは以下の要因によってインデックスの更新を行う．

- 各ピアのローカルでの文書の状況の変化
- ネットワーク上での周囲の変動

各ピアのローカルでの文書の状況の変化には新規文書の追加，文書の更新，文書の削除等によってそのピアの “有用度” が変化する場合である．他方，ネットワーク上での周囲の変動とは，他のピアと接続，切断または他のピアからインデックス更新メッセージの受信を示す．ローカルにインデックスを更新したのち，そのピアは更なるインデックス更新が必要かどうかの判定を行う必要がある．各ピアはその隣接ピアへ最後に推奨したピア集

合を逐次記録する．このピア集合に変更があればインデックス更新メッセージを送信するが，前回の推奨ピアとの差異がなければ，メッセージを送信しない．例えば，図 4.2 においてピア E に更新が起こったとする．この更新により E はインデックス更新メッセージを B へ送信する．この更新によって B はその DI を更新する．このとき B の E 以外の隣接ピア A および F への推奨ピアが変更されるならば，さらに B は A および F にインデックス更新メッセージを送信する．メッセージを受け取った A および F はこの手順を繰り返す．B は E からのインデックス更新メッセージによって A および F へ送信すべき推奨ピア集合へなにも影響がないのであれば，このインデックス更新は B で終了することになり，更新メッセージの伝播はこれ以上行われない．

ピア間の接続，システムへの参加

次に，ピア間の接続（join）について述べる．接続は更新の一種であると考える．例えば，p_1 と p_2 の接続ではそれぞれ相手ピアへのインデックス更新メッセージを作成し，互いにそのインデックス更新メッセージの交換に成功すれば，接続に成功したとする．そして，p_1 と p_2 は各 DI を更新し，必要があればインデックス更新メッセージをそれぞれの隣接ピアへ送信する．

例えば，図 4.3 においてピア A とピア B の間での接続を行うとする．このとき A，B ともそれぞれ相手ピアへの更新メッセージを作成し，そのメッセージを交換する．このメッセージの交換に成功すれば接続に成功したこととなる．そして，この例ではこれ以上の更新が発生するため A は C と D へ，B は E と F へ更新メッセージを送信する．

ピア間の切断，システムからの離脱

ピア間の切断（leave）は接続と逆の動作である．切断も接続と同様に更新の一種であると考える．それぞれのピア間で切断を行う場合，それぞれのピアの DI から切断する方向にあるピア情報を削除すれば十分である．

図 4.4 においてピア A とピア B の間で切断が発生した場合を例に取る．まず，A と B の DI 内のそれぞれ相手の情報及びその相手方向にあるピアの情報も併せて削除する．更新が必要となれば更新を行う．A は C と D へ，B は E と F へそれぞれ新たな更新メッセージを送信する．これで切断は完了する．

ピアがシステムから離脱する場合，確実に明示的にシステムから離脱するとは考えられない．ユーザの操作によるシステムからの離脱だけではなく，各ピア自体の故障，ネット

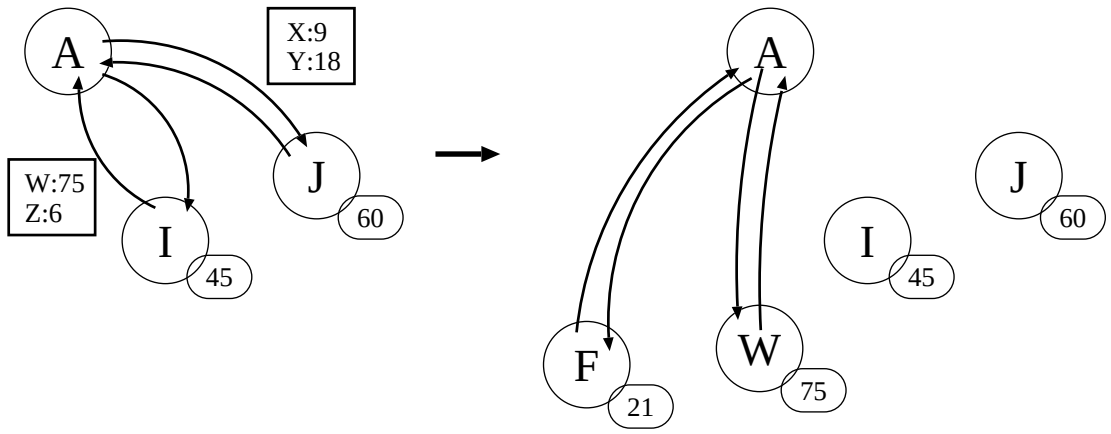


図 4.5 NDI を用いた問い合わせ処理の例

ワーク障害によってシステムから離脱する場合もあり得る．そこで，各ピアは隣接ピアがシステム上に存在しているかを判断しなければならない．システム上にて接続しているピア間では定期的に ping/pong メッセージをやりとりし，この方法にて各ピアは隣接ピアの存在を確認し，ピア間での接続の状況を確認する．このとき，pong メッセージを受信できなければ，その隣接ピアがシステムから離脱したと判断する．接続・切断によってインデックスの更新が行われるため，インデックス更新が引き続き行われる．

4.2.2 問い合わせセッション

問い合わせは以下のように処理される．問い合わせが発行されると，まずピア内にある文書に対して問い合わせ処理を行う．この問い合わせで十分な検索結果が得られれば処理を終了する．不十分な場合は問い合わせの転送を行う．このとき問い合わせを発行するピアの DI に格納されたピアで有用度が上位 n 以内にあるピアに対して問い合わせが行われる．問い合わせを受けたピアは，問い合わせ処理を行い，その結果とともにそのピアの推奨ピア集合も添付して送信する．問い合わせ結果を受け取ったピアは十分な文書が得られれば処理を終了する．不十分な場合は，問い合わせ結果とともに受け取った推奨ピア集合と DIs に存在するピア集合との和集合から問い合わせ送信済みのピア集合を除いたピア集合に対して有用度の上位 n 個のピアに問い合わせを発行する．この操作を十分な結果が得られるまで続ける．

例えば，図 4.2 におけるピア A が問い合わせを行う場合を考える．これを図 4.5 に示す．まず A のローカルにて解決しようとする．解決すれば，問い合わせは終了となる．解

決しなければ他のピアへ問い合わせを転送する．問い合わせの転送先の候補は A の DI でのランキングの上位 n 個内にあるピアとなる．$n = 2$ であれば，まず 2 つのピアへ問い合わせを行う．この例では転送先は I，J となる．I と J に問い合わせメッセージを送信し，問い合わせの結果を得る．このとき，問い合わせ結果とともに，I と J からそれぞれのピアが推奨するピアの情報を A は得ることになる．そして，A は結果に対して満足すれば問い合わせは終了する．満足しなければ，結果とともに得たピア情報と A の DI に格納されていてまだ問い合わせ転送を行っていないピアから転送先のピアを選出する．この場合，I，J からそれぞれピア X と Y，W と Z のピア情報を得たとする．これと A の DI に格納されているまだ送信していないピア集合から F と W が選出される．再び問い合わせの転送を行い，以上を繰り返すことで問い合わせ処理を行う．

この問い合わせ処理において，“有用” ではないピアにはメッセージは送信しない．また，各ピアは問い合わせを行う前には既知ではなかったピアに対しても問い合わせを行うことが可能になり，幅広くシステムに問い合わせメッセージを送信するが帯域幅消費量を抑えることも可能にする．

4.3　Direct Index の拡張

NDI を用いた問い合わせ処理では問い合わせメッセージの転送先は問い合わせを行っているピアの DI 内の “有用” なピアである．このとき，問い合わせメッセージの転送先のピアがその問い合わせに対する結果を多く保有しているとは限らない．NDI ではピアの “有用度” は格納しているが，格納されているピアがどのようなトピックの文書を所有しているか，という情報を格納しない．そのため NDI はシステム内で共有されている文書のトピックが多くない状況下にて用いることができる．すなわち，NDI 用いた問い合わせ処理を行う場合，その問い合わせに対するトピックに関する文書を積極的に保有しているピアを特定された後に，それらのピアの内で “有用” なピアを発見する仕組みが必要になる．そのために各ピアがあるトピックにどのくらい関連するかを指標する方法を設ける必要がある．そこで本研究ではピアとトピック間の関連性を示す**トピックの “有用度”** を導入する．各文書がトピックごとに分類され，トピックごとに文書の “有用度” を合計したものとしてトピックの “有用度” を決定する．さらにトピックの “有用度” を格納するように拡張した *Topic-based DI*（*TDI*）と呼ばれる DI を導入する．この節では TDI の構成，管理方法及びそれらを用いた問い合わせ処理の方法について述べる．

peer "A"						
UID	location	usefulness	categories			
			DB	IR	Network	OS
A	local	30	18	4	8	0
B	neighbor	50	30	0	12	0
C	neighbor	30	0	0	5	20

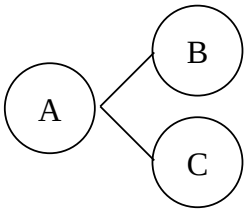


図 4.6 TDI の例

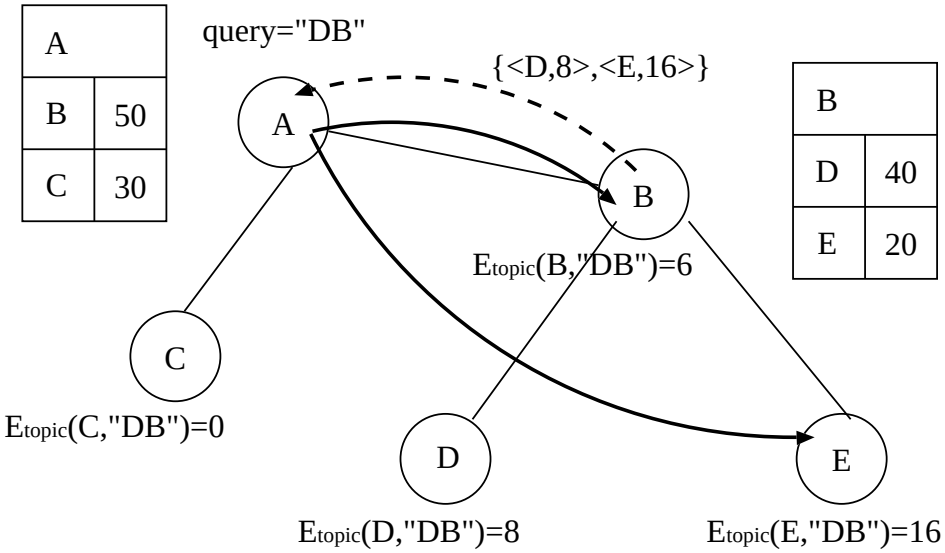


図 4.7 TDI を用いた問い合わせ処理

4.3.1 Topic-Based DI

TDI の構成と更新の方法

TDI は，各ピアの UID，ピアの “有用度” および方向以外に各ピアのトピックの “有用度”（*“usefulness” of topics*）も格納する．TDI では，ピア p のトピック t に対する “有用度” を（4.1）式のように p が保有するトピック t の文書の “有用度” の総和で表す．

$$E_{topic}(p,t) \equiv \sum_{d \in D(p,t)} E_{doc}(d) \tag{4.1}$$

ここで $D(p,t)$ は p が保有している，トピック t の文書の集合である．

図 4.6 は TDI の例である．各ピアは他のピアからインデックス更新メッセージを受け取り，TDI 内にピア情報を格納する．TDI のサイズに応じてインデックス更新メッセージのサイズも大きくなる．帯域幅消費量を低減するために，インデックス内の情報はもあ

まり大きくない方がよい．ピアの UID を 25 バイト，ピアの “有用度” を 4 バイト，ピアの方向を 25 バイトで表すとすれば 1 つのピアの情報として 54 バイト[*1] 必要である．隣接ピアの数の平均を f，各ピアが隣接ピアへ推奨するピアの数を m とすれば，NDI を用いる場合，各ピアは $54 \cdot f \cdot m$ バイト必要とする．各トピックの “有用度” を 4 バイトで表し，各ピアが所有している文書全体でのトピックの数が c 個であれば，TDI を用いる場合，各ピアは $f \cdot m \cdot (54 + 4 \cdot c)$ バイト必要とする．各ピアは，一定の範囲のトピックの文書を所有するため，TDI においてトピックに対する “有用度” の情報はあまり大きくならない．そのため，NDI と比較すると TDI のサイズは大きくなるが，際だって大きくなることはない．

TDI を用いた場合でのピア p_2 から隣接ピア p_1 へ推奨するピア $rec(p_2 \rightarrow p_1)$ の決定方法は p_1 と p_2 の DI 内のピア p 間の類似度をそれぞれのピアが所有している文書のトピックに関する類似度を用いて決定する．トピックに関するピア類似度 $tsim(p, p_1)$ は次のようにそれぞれのピアの各トピックの “有用度” での余弦によって決定する．

$$tsim(p, p_1) \equiv \frac{\sum_t E_{topic}(p,t) \cdot E_{topic}(p_1,t)}{\sqrt{\sum_t E_{topic}(p,t)^2} \cdot \sqrt{\sum_t E_{topic}(p_1,t)^2}} \tag{4.2}$$

p_2 はその DI 内のピア p に対して $tsim(p, p_1) \cdot ER_{p_2}(p, p_2)$ を計算し，この値に応じてランキングし，上位 m 個のピアを p_2 から p_1 への推奨ピア集合とする．インデックスの更新方法は NDI を用いた場合と同様である．しかしながら，トピックに対する “有用度” の変化によって推奨するピアを変更する可能性がある．そのため，TDI を用いた場合，NDI と比べ各ピアのインデックスの更新頻度が高くなり，帯域幅消費量は増加してしまう．

TDI を用いた問い合わせ処理

TDI を用いた問い合わせ処理ではピア p はその DI 内の各ピア p_i をランク付けするとき，$E_{topic}(p_i, t)$ を用いる．p が問い合わせ $q = \{q_1 \wedge q_2 \wedge \ldots \wedge q_k\}$ を行う場合について考える．ここで $q_r (r = 1 \ldots k)$ は問い合わせ q のキーワードを示す．q_r に対応する DI 内のトピックが t_r であるとき t_r の “有用度” を用いて

$$ER_p(p_i, p) \cdot \prod_r \frac{E_{topic}(p_i, t_r)}{ER_p(p_i, p)} \tag{4.3}$$

[*1] この値は各データ自体のサイズであり，実際に DI にこれらのピア情報をレコードとして格納する場合，ヘッダ等が必要になるためレコードのサイズは大きくなる．

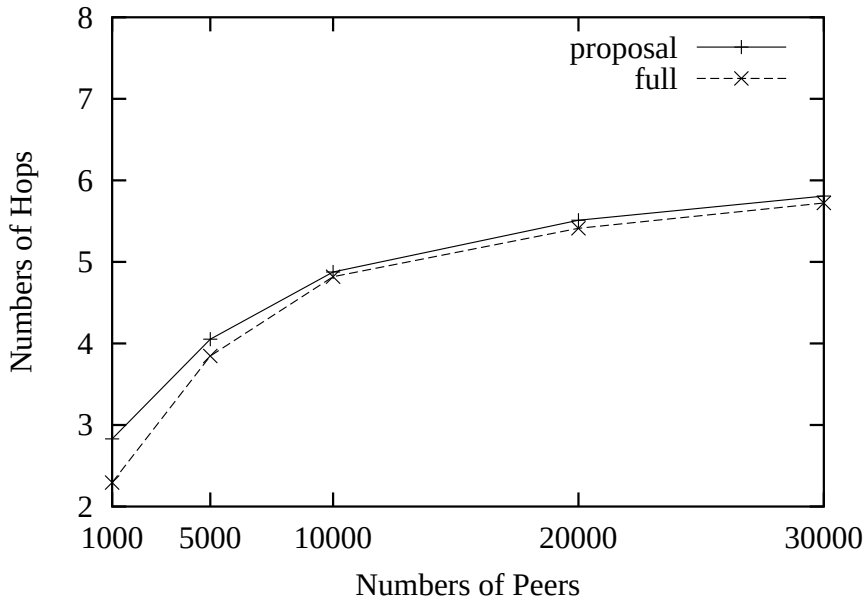
を計算し，この値に応じてピアをランキングする．ただし，$ER_p(p_i, p) = 0$ のとき，式（4.3）の値を 0 とする．式（4.3）はトピック $\{t_1 \wedge t_2 \wedge \ldots \wedge t_k\}$ の “有用度” の p_i の “有用度” での割合を基に，その期待値を求めている．p_i の隣接ピア p へ問い合わせメッセージを送信した場合，p はその検索結果とともにその問い合わせに関連するトピックの “有用度” に応じて推奨ピアを決定する．このとき，p の DI 内に格納されている各ピア p_j に対して式（4.3）を計算し，上位 n 個のピアを推奨ピアとする．もし問い合わせ q のタームが単一である場合，すなわち $q = \{q_0\}$ であれば q_0 に対応するトピック t_0 の “有用度” のみで問い合わせメッセージの転送先を決定する．

図 4.7 は TDI を用いた問い合わせ処理の例である．ピア A は問い合わせ “DB” を行う場合，A のローカルで問い合わせが終了しないならば A はその DI 内の各ピア p_i に対して $E_{topic}(p_i, \text{“}DB\text{”})$ （$p_i \in \{B, C\}$）を計算し，問い合わせメッセージの送信先を決定する．このとき $E_{topic}(B, \text{“}DB\text{”}) = 18$，$E_{topic}(C, \text{“}DB\text{”}) = 0$ であるため A は B に問い合わせメッセージを送信する．次に B はローカルに問い合わせを行い，$E_{topic}(p_j, \text{“}DB\text{”})$（$p_j \in \{D, E\}$）を計算する．このとき $E_B(D, B) = 40$，$E_B(E, B) = 20$ であり，$E_{topic}(D, \text{“}DB\text{”}) = 8$，$E_{topic}(E, \text{“}DB\text{”}) = 16$ である．B は推奨ピア集合 $\{< D, 8 >, < E, 16 >\}$ と検索結果を A に送信する．このとき D の “有用度” は E の “有用度” よりも高いが，A は D よりも E をより “有用” であると判断する．各ピアは TDI を用いることによって，問い合わせメッセージを送信先を修正することが可能になり，NDI と比較し問い合わせ処理性能の向上が期待できる．

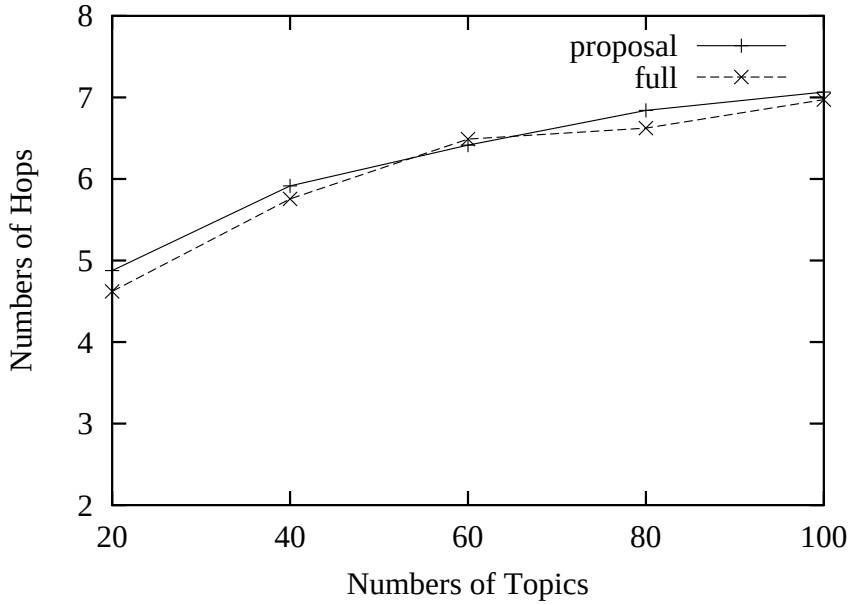
4.4 インデックス更新に関する議論

P2P システムが他の分散システムと大きく異なる点として，ネットワーク上の各ピアが非常に動的であること，非常に多くのピアが存在することである [19]．このことは，P2P システムでのインデックス内の情報の整合性を維持することを困難にする．

DI を用いた問い合わせ処理ではインデックス内の情報に応じて長距離リンクを形成し，問い合わせメッセージの転送先を決定する．そのため，インデックス内の情報が古い場合，そのピアは長距離リンク先の対象としてそのときのネットワークの状況にそぐわないピアを選択する可能性がある．さらに，すでにシステムから離脱したピアに対して長距離リンクを形成しようとする可能性もある．また，インデックス更新メッセージがネットワーク上の遠くに位置するピアまで伝播する可能性がある．この場合，あるピアがインデックス

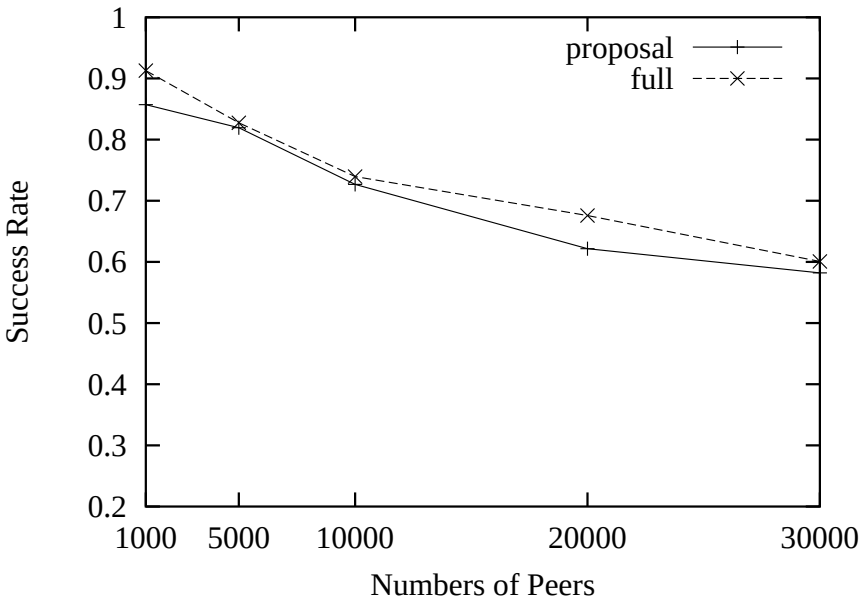


（a）ピア数変動

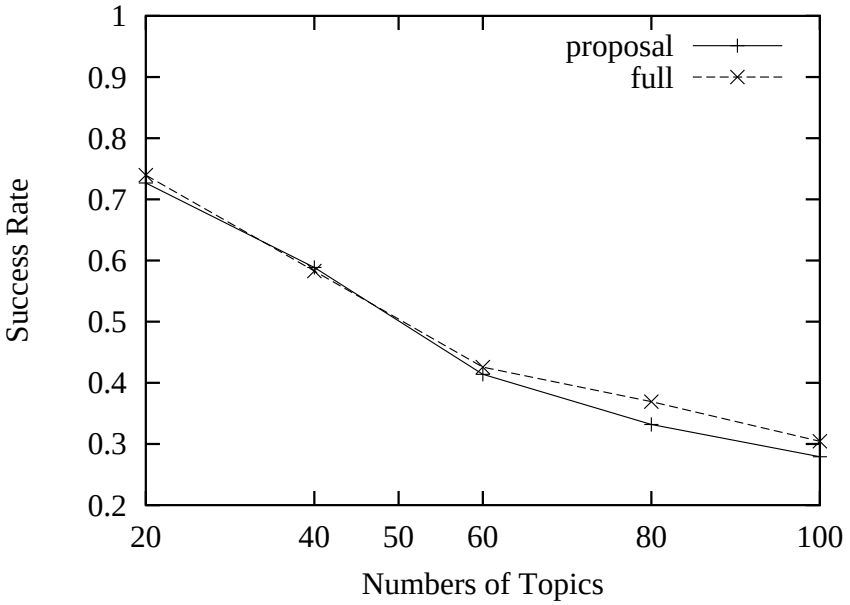


（b）トピック数変動

図 4.8　NDI の更新と応答時間

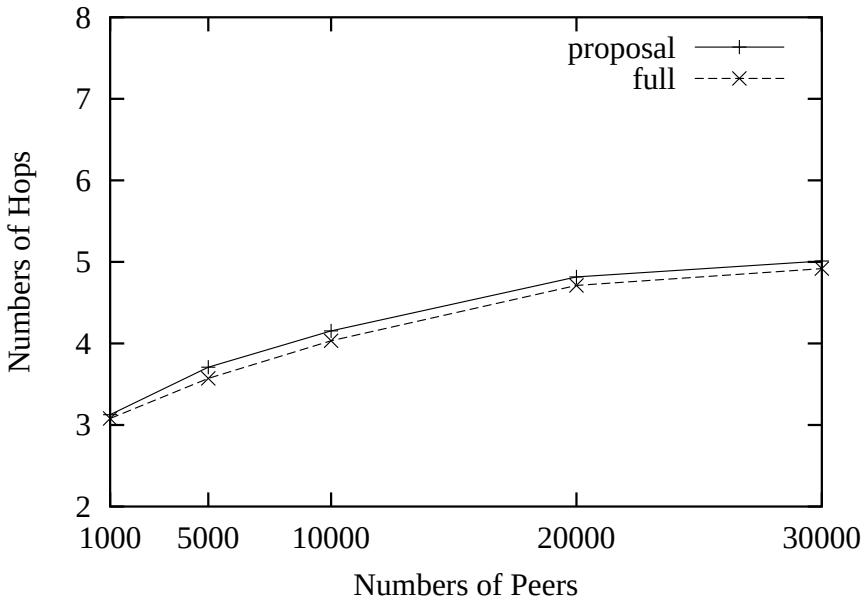


（a）ピア数変動

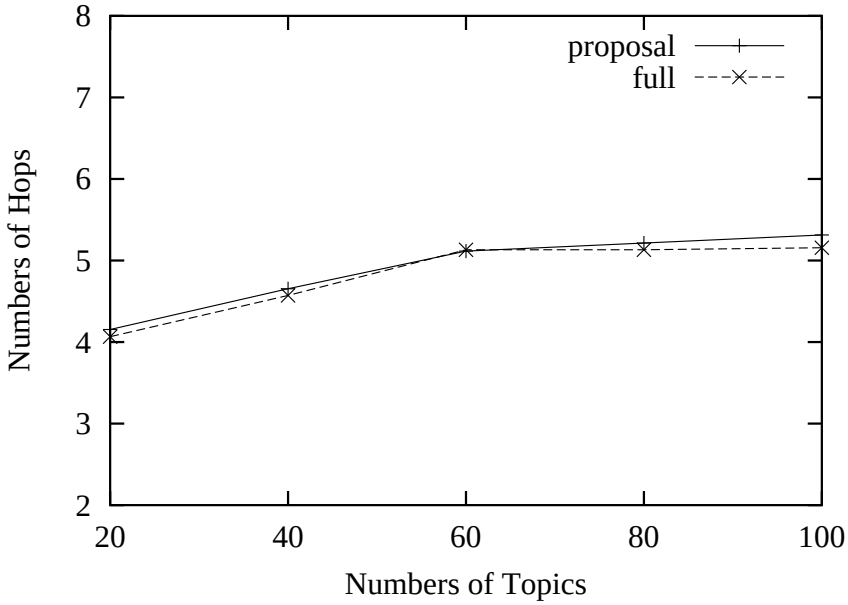


（b）トピック数変動

図 4.9　NDI の更新と成功率

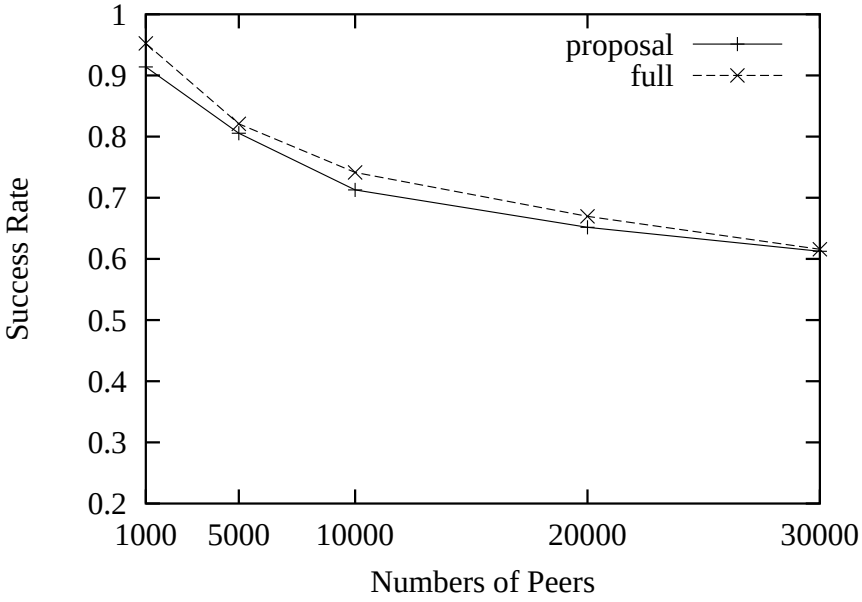


(a) ピア数変動

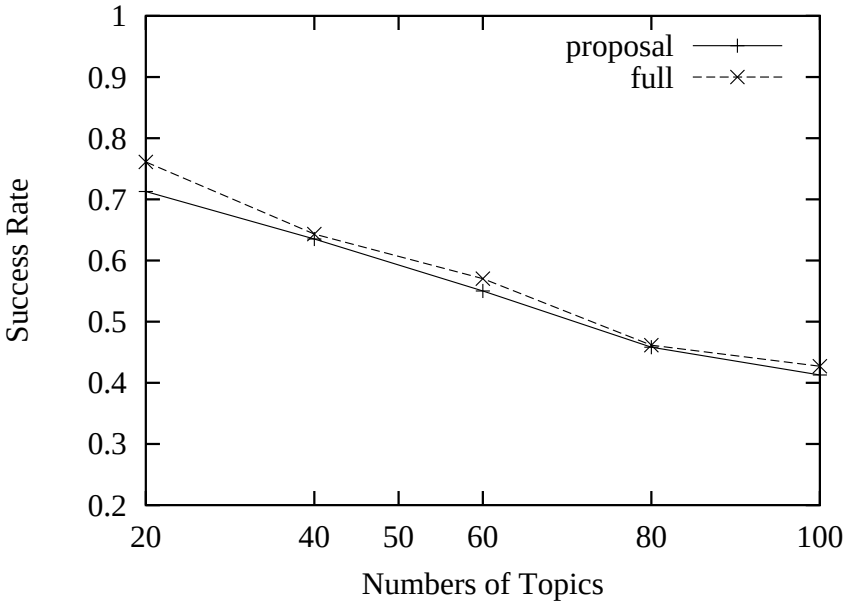


(b) トピック数変動

図 4.10　TDI の更新と応答時間



（a）ピア数変動



（b）トピック数変動

図 4.11　TDI の更新と成功率

更新メッセージを受信したとき，インデックス更新の起源となったピアは，すでに新たなインデックス更新を行っているかもしれない．インデックス更新メッセージが非常に遠くに伝播させられるのであれば，各ピアはネットワークの広範囲に渡って長距離リンクを形成することが可能になるが，インデックス更新メッセージの内容はピア間の距離に応じて不確かな情報になってしまう可能性が高くなってしまう．

そのため，インデックスの整合性を可能な限り維持する方が望ましい．このことは分散インデックスを用いた問い合わせ処理は特に気を使うべきである．しかしながら，頻繁なインデックス更新はインデックス更新メッセージ数を増加させ，帯域幅消費量を増加させてしまう可能性がある．また，ネットワーク上の遠くに位置するピアに対するインデックス更新メッセージの伝播は頻繁なインデックス更新を行うことでは解決できない．文書の内容をベースとしたインデックスではインデックスの整合性が保たれなければ，効率的な問い合わせ処理は非常に困難なる．たとえば，文書の内容をインデックスに格納する Routing Index（RI）[19] では以下の 2 つの方法を考案している．1 つめはインデックスの伝播範囲（horizon）をあらかじめ決定し，その範囲以外には伝播させない方法である．2 つめは，インデックス更新メッセージが伝播する距離に応じてその効力を低下させる方法である．2 つの方法ともネットワークの局所にインデックス更新を通達する方法である．よって，ピア間の距離が遠いピアに対してインデックス更新メッセージを伝播する必要がない．しかしながら，このインデックス更新の方法では各ピアはそのピアから一定範囲以内の情報のみしか知ることができないため，スケーラビリティの改善を行う必要がある．

本章で示した NDI および TDI のインデックス更新の方法はピアの接続状況のや推奨するピアの情報の変化をインデックス更新のタイミングとし，フラッディングによるインデックス更新メッセージを伝播させる．そのため，インデックス更新の頻度は文書の内容をベースとするインデックスでの更新に比べ低く抑えることが可能であるが，インデックスの整合性の維持に対してあまり考慮していない．この本研究でのインデックス更新の方法を用いた場合ではどのくらいの整合性が保たれるのであろうか．そこで，本研究でのインデックス更新の方法（proposal）とシステム上の全ピアのインデックスの整合性が完全に保たれている場合（full）を比較することで示す．インデックスの内容は問い合わせ処理での一定以上の検索結果を得るまでのホップ数とその成功率に大きな影響を与えると考えられるため，これらの結果を示すことでインデックス更新の方法の性能を示す．また，システムの状況を変化させるため，ピア数および文書のトピック数を変動させた．この状

況を図 4.8，4.9，4.10，4.11 に示す．これらの結果では，本研究でのインデックス更新の方法での結果は proposal，システム上の全ピアのインデックスの整合性が完全に保たれている場合での結果は full として示している．また，ピア数は 10,000，文書のトピック数は 20 を基本値としている[*2]．

図 4.8，4.10 は NDI および TDI を用いたときの 20 件の検索結果を得るのに必要であったホップ数を示している．NDI を用いた場合では，本研究でのインデックス更新の方法はピア数の変動，トピック数を変動させた場合でも完全に整合性が保たれている場合とあまり変わらない結果を得た．この傾向は TDI でも同様であった．また，図 4.9，4.11 はそれぞれ NDI，TDI を用いたときの 20 件以上の検索結果取得の成功率を示す．この結果も NDI，TDI にかかわらず先ほどの結果と同様であった．

本研究では長距離リンクの指標としてインデックスを用いる．その指標はインデックスの内容が完全でなくても DI の効率性はあまり低下しないことが上記の実験結果から得ることができる．

4.5 本章のまとめ

本章では，非構造 PP2P システムでピア情報を格納する Direct Index（DI）を提案した．また DI の構造，ピア情報の伝播の方法，および DI を用いた問い合わせメッセージのルーティングの方法を示した．さらに，DI を各ピアのトピック情報を格納するように拡張した Topic-based DI（TDI）を提案し，TDI の構造および TDI を用いた問い合わせ処理の方法を示した．DI を用いた問い合わせ処理では以下の利点がある．

- 問い合わせメッセージ先は隣接ピアにだけではなく，ネットワーク上にて離れて位置するピアもそのターゲットとすることが可能になる．無駄な問い合わせメッセージの転送を低減することが可能になるため帯域幅消費量および応答時間の向上を期待することができる．
- 問い合わせ処理の最中に新たなピア情報を取得することが可能である．そのため動的に問い合わせメッセージの転送先を選択可能である．

[*2] 実験設定の詳細は第 6 章を参照のこと．

第 5 章

文書複製

5.1 はじめに

前章では問い合わせ処理の向上のためにピア情報に関する分散インデックスである Direct Index（DI）を導入した．しかしながら分散インデックスでは解決できない問題がある．DI では “有用” なピアを動的に発見するが，問い合わせ処理が行われているときのシステムに参加しているピアのみを問い合わせメッセージの転送先とする．P2P システムではピアは動的にシステムへの参加およびシステムからの離脱を行うため，システムから離脱しているピアが保持していた文書がシステムから “消えて” しまう．また “有用” な文書がより少ないホップ数で発見されやすくするための仕組みは DI だけでは解決できない．この 2 つの問題を解決し，問い合わせ処理のさらなる向上を図るために，本研究では文書を複製し，他のピアへ配置する方法を考案した．本研究では複製の数を制限し，複製処理に関するピア間のやりとりを極力抑制するために Partial-Replication を行う．そこで文書の複製のために

1. 複製対象となる文書の選択
2. 複製回数の決定
3. 複製配置の場所の選択

の 3 つの決定を行い文書の複製処理を行う．また，文書が更新された場合，その文書に複製が存在するならば，複製の再配置を行う必要がある．文書更新のメカニズムを簡略化するために Lazy Master Replication を行う．

図 5.1 は P2P システムでの文書複製の各手法の目的を示している．P2P システムを利用したグループウェアである Groove[37] では，静的に生成されたピアグループ内で共有

されている文書はそのピアグループに所属している全ピアに対してその文書の複製を配置する．そのため，ピアグループ内の全文書の永続性は高い．Bid Trading[17] はその応用として電子図書館を想定している．そのため，全文書の永続性を積極的に考慮している．Oceanstore[54] は DHT である Tapestry[83] を用いた P2P アーキテクチャである．Oceanstore ではその応用例の 1 つとして，科学技術データや電子図書館でのアーカイブを想定していて，共有される全文書の永続性を積極的にサポートしている．他方，Partial Lookup[68] は HP2P システムにて，文書をどの super peer にエントリさせるかを考慮している．

本研究での文書複製の手法は “有用度” が高い文書を積極的に複製する．そのため，“有用度” が高い文書の永続性は積極的にサポートする．しかし “有用度” が低い文書の永続性は考慮しない．各ユーザにとってそのとき発見したいもの，利用したいものを極力複製し，その発見のサポートを行うことを本研究の文書複製の目的としている．さらに本研究では応答時間の向上も考慮している．応答時間の向上のための 1 つの方法として，ある文書を必要とするピア，もしくはそのピアの近くに複製を配置する方法を挙げることができる．しかしながら，各ピアはあるとき必要とした文書をその後もその文書を必要とするとは限らない．P2P システムでは各ピアは頻繁にそのネットワークへの接続状況を変化させる．そのため，あるピアが文書の複製を配置するときに，複製配置の対象とするピアがいつもシステムに参加しているとは限らない．そこで本研究では，ピアグループを動的に，適応的に形成し，そのピアグループに所属するピアから複製配置の対象を選択しそのピアに対して複製を配置する方法を提案する．ピアグループの形成ためにはあるピアグループにどのピアが所属するを判断するための基準を設定する必要がある．また複製配置のために，ピアグループに所属するピアのうちどのピアからその対象として選ぶかを決定する必要がある．

本章での以降の節では複製対象となる文書及びその複製回数，複製の配置方法，文書の更新方法について述べる．

5.2　複製対象文書とその複製回数の決定

複製対象となる文書およびその複製回数を決定するために，本研究では文書の “有用度” を利用し，“有用” な文書ほど多く複製する．本研究では非構造 PP2P システムを取り扱うためサーバが存在しない．そのためシステム全体の文書の情報を集約することは困

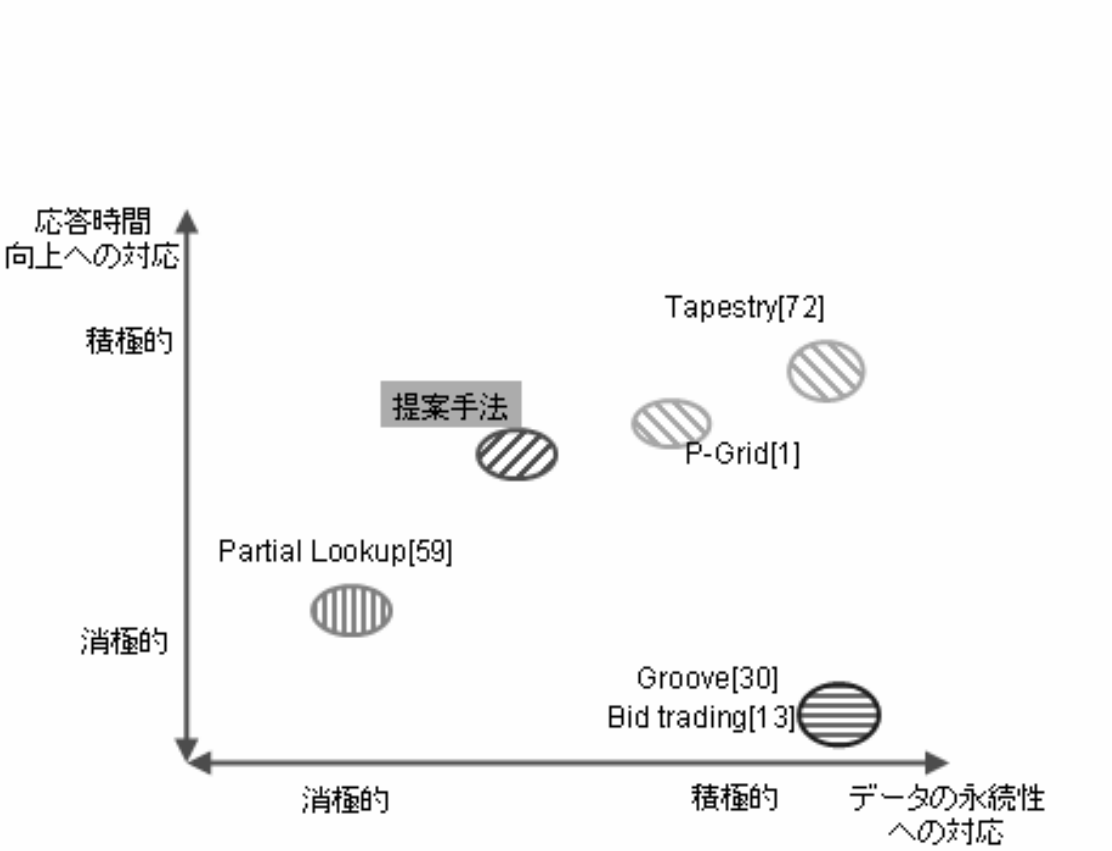


図 5.1 P2P システムでの文書複製の各手法の比較

難である．また，各文書の有用度は参照回数や時間の経過により非常に変動しやすい．そのため，文書の “有用度” で文書の複製数を決定するためには文書の有用度の情報を頻繁に交換しできるだけ多くの文書の情報を得る必要があるため，文書の “有用度” のみを用いて文書の複製数を決定することはシステムのパフォーマンス低下を招く要因になる可能性がある．そこでピア p とその各隣接ピアの “有用度” によって各ピアの複製数を相対的に決定することにした．p とその隣接ピアの集合を $\mathcal{P}$，1 ピアあたり c 個の複製を生成する場合，p とその隣接ピア全体での複製の合計数は $c \cdot |\mathcal{P}|$ となる．このとき p の複製数 $P_{peer}(p)$ は次の式にて決定した．

$$P_{peer}(p) = c \cdot |\mathcal{P}| \cdot \frac{E_{peer}(p)}{\sum_{p_i \in \mathcal{P}} E_{peer}(p_i)} \tag{5.1}$$

ピアの “有用度” は頻繁に変動する値である．そのため $P_{peer}(p)$ を頻繁に再計算する必要があるように思える．しかしながら，頻繁に再計算を行うのは効率上問題があり，また実際には $P_{peer}(p)$ の変化はあまりない．$P_{peer}(p)$ が変化するときは，その隣接ピアの “有

用度” が変化するとき，もしくはピア p が所有する文書の更新や新規文書の追加が行われたときである．そこで，隣接ピアに変化があるとき，すなわち，隣接ピアからの接続もしくは切断の処理があった場合に再計算を行うようにすることで，頻繁な再計算を抑えることができる．4 章で提案した DI とともに用いる場合では，インデックス更新メッセージを受け取ったときに再計算する．

ピア p は図 5.2 に示すの手続きにて複製対象となるオリジナル文書を選択し，その複製を他のピアへ配置する．この手続きでは，ある文書を複製対象と選択する場合，その文書の “有用度” とその文書の既存の複製の数を考慮している．すなわち，“有用度” が高い文書ほど複製の対象になりやすいが，その文書が複製されるたびに複製の対象になりにくくする．ピア p が所有する文書 d_1，d_2，d_3 の “有用度” がそれぞれ 9，4，1 であり，$P_{peer}(p) = 4$，また p は，まだいずれの文書も複製を行っていないとする．複製対象となる文書を選択するとき，まず < 文書の “有用度”> / < 文書の複製の数 > を計算する．このとき p は d_1 を複製の対象として選択する．次の複製の対象を選択するとき，文書 d_1，d_2，d_3 の < 文書の “有用度”> / < 文書の複製の数 > を計算するとそれぞれ 4.5，4，1 であるので，またもや d_1 が複製の対象になる．$P_{peer}(p) = 4$ であるので，この手続きを 4 回繰り返す．その結果，文書 d_1，d_2，d_3 の複製の数はそれぞれ 3，1，0 になる．この複製対象の選択の手順では，文書の “有用度” が高いものほど選択されやすいが，常に文書の “有用度” が高いものが選択されることにはならない．

5.3　文書の配置方法

次に複製文書配置の場所を決定する方法が必要になる．本研究ではピアグループを動的に形成し，そのピアグループに所属するピアへ複製を配置する．複製文書配置の方法として以下の 4 つの方法を検討する．

5.3.1　PN（Placement on Neighbors）方式

ピアはその隣接ピアを同じピアグループであると見なす．そして隣接ピアのうちもっとも “有用度” が低いピアへの複製を試みる．もし，隣接ピアの複製のレポジトリには空きがあるならば，複製は配置可能である．もし，レポジトリに余裕がないために複製が配置できないのであれば，次に “有用度” が低い隣接ピアへ複製を配置しようとする．この方法では，複製文書を配置するピアに 1 ホップで到達できるため，文書を配置しているピアの状態をすばやく把握することが可能である．

```
//replicated document selection phase
function select_replicated_docment (peer p){
  int ndr = <the sum of replicas p placed>;
  list dl = <a list of documents p has>;
  while(p_peer(p)<ndr){
    document rd;
    int n = dl.size();
    //seclect a replicated document
    foreach (i:dl){
     document d = dl[i];
     double ud = <a usefulness of d>;
     int d_rep = <numbers of replicas of d>;
     rd = max(rd,ud/(d_rep+1));
    }
    //placement the replica
    place_replica(rd);
  }
}
```

図 5.2 複製対象となる文書を選択の手順

5.3.2 PIR（Placement on Inverted References）方式

P2P システムでは，ある文書を取得するときに直接ピア間で文書を転送する．ここで，ある文書 d へ参照を行う場合，参照を行ったピアは d に対して関心があり，d を保有するピアと d を参照したピアの間には関連性があると考えられる．あるピア p_1 が保有する任意の文書が他のピア p_2 より参照されたことがあれば，この方式では，p_1 は複製を p_2 上へ配置しようとする．各ピアはそのピアへ文書参照のスタックを用意し，どのピアが参照したかという参照履歴を記録する．そのスタックに蓄えられたピアをそのスタックを所有しているピアは同じピアグループに所属すると見なす．複製を配置する場合，ピアをスタックから取り出しそのピアへの複製配置を試みる．

5.3.3 PRP（Placement on Related Peers）方式

各ピアはそのピアと類似の問い合わせを発行するピア上へ複製を配置する．例えば，ピア p_1 と p_2 はともに “database” というキーワードでの問い合わせを行うことが多い場

合，p_1 と p_2 の関連性は高く，p_1 が複製を配置する場合，p_2 は複製配置の対象となる．p_1 と p_2 の類似度はそれぞれピアによる過去 l 件の問い合わせのキーワードでの余弦によって決定されることにする．すなわち，

$$qsim(p_1, p_2) = \frac{\sum_k tf(q_k^1) \cdot tf(q_k^2)}{\sqrt{\sum_k tf(q_k^1)^2} \cdot \sqrt{\sum_k tf(q_k^2)^2}} \tag{5.2}$$

ここで $q^i = \{q_1^i, \ldots q_l^i\}$ はピア p_i が行った問い合わせのキーワードを示し，$tf(q_k^h)$ はキーワード q_k^h の出現頻度を表す．この方式では各ピアは類似度が 0 以上であるピアが同じピアグループに所属していると判断する．PRP 方式を用いて複製を配置する場合，複製を配置しようとするピアともっとも類似度が高いピアへ複製配置を試みる．文書配置を行うピアはそのピアが既知であるピアから配置対象となるピアを選択するが，この文書配置方式を用いる場合，各ピアは他のピア情報を得るときに，他のピアの存在以外にも問い合わせの状況情報も得る必要がある．

5.3.4　ランダム方式

各ピアは，そのピアが既知である任意のピア上へランダムに複製を配置する．ランダム方式以外の各文書配置方式において文書配置対象となるピアを見つけ出すことができない場合がある．その場合，ランダム方式を用いて文書を配置する．

システムに参加している各ピアは他のピアの情報を次のように得ることができる．

- システム上のピアが接続を行うときに，そのピア間にてそれぞれのピア間で既知であるピアの情報を交換する．
- 問い合わせ処理を行うときに，問い合わせメッセージを送信したピアと受信したピアは互いの情報を交換する．

5.4　文書の更新

オリジナル文書の更新が発生するとき，その文書に複製が存在すれば，同期処理を行う必要がある．そこで，各オリジナル文書を保有するピアはその複製を配置したピアへ文書の再配置を行う．しかしながら，各ピアが文書を同時にアップロードできる数は有限であるため，更新された文書の複製を同時に再配置できるとは限らない．さらに P2P システ

ムでは，ある文書の複製を所有しているピアが常にシステムに参加しているとは限らないため，完全にピア間で同期をとりながら文書の更新を行う Eager Replication を用いることは困難である．そのため，本論文では非同期に文書の更新を行う Lazy Replication を用いる．非同期に更新を行う場合，更新の遅延が発生してしまう．更新の遅延が発生しているときに問い合わせ処理が行われてる場合を考えてみる．再配置が行われている文書を必要としない問い合わせであればその更新はその問い合わせ処理に対してあまり影響を与えない．しかしながら，再配置中の文書を必要とする問い合わせ処理が行われている場合，再配置前の情報を基に検索結果を返すことになる．文書 d が更新されその複製の再配置する場合，再配置中であることを明示するために，d の複製の再配置前に d の複製を非同期の状態にするメッセージをその複製を保持しているシステム内の全ピアに対して送信する．そして，新たな複製を配置したときに再び同期の状態にする．複製が非同期である状態を短期間にするためには文書更新の遅延をなるべく抑える必要がある．そこで本論文では，図 5.3 に示す複製再配置の手順を用いる．この複製の再配置では，新たな複製の再配置を完了したピアがその複製の再配置を手助けを行う．この仕組みによって文書更新の遅延を抑制することができる．オリジナルの文書を所有しているピア p はその文書の複製の再配置を終えており文書配置を行うことが可能なピアへ複製配置対象のピアのリストを渡す．このとき，d の再配置を終え文書配置に協力可能なピアの集合を fp，文書更新を終えていないピアの集合を nfp とすると，p は，nfp のリストを $min(|nfp|, |fp|)$ 個の互いに異なる部分リストに分割し，この個数だけ $R_f(d)$ からピアを無作為に選び（図 5.3 : $select\_peers$ 関数），そのピアへ部分リストを送信する．次に，複製配置対象のピアのリストを受け取ったピアは複製対象ピアの再配置の状況を確認する．このとき，複製対象ピアをはシステム上に確認できれば，新文書を再配置する．そしてまだ文書再配置が行われていないピアのリストを p へ送信する．以上の手続きを繰り返し，文書の再配置を完了する．これにより，更新の遅延を抑えることが可能になる．

また，P2P システムは非常に動的なシステムであるため，文書の更新情報が複製配置をおこなったピアに到達しない場合もある．例えば，複製を保有するピアが離脱状況にあるときに，文書の更新が行われる場合もあり得る．このため，再び参加状況になったときに所有している複製の更新状況を調べる．更新情報を発見したとき，改めて文書の再配置を依頼することで文書の更新を行う．オリジナル文書を保有しているピア p が，以前複製を配置したピア p_i へ文書を再配置する際に，文書更新を行い文書を再配置しようとした

```
//select target peers
function select_peers(document d, list tp, list fp){
  return a part of peers who don't finish
  to place the replica of d(randomly).
}
//push phase
function place_replica(document d){
  list tp = <a list of peers who have replica of d>;
  list fp;
  while(fp.size()!=tp.size()){
    //placement new replica
    foreach(i:fp){
         list nfp = select_peers(d,tp,fp);
       send_peer_list(i,d,nfp);
    }
    list pl = select_peers(d,tp,fp);
    push_replica(p,d,pl);
    while (true){<wait for a while>}
    fp.add(<peers who finished to place new replica>);
  }
}
//replica place procedure
function push_replica (peer p, docuemt d, list tp){
  //tp is a list of target peers for replica placement.
  list fp;
  foreach (i:tp){
    peer rp = tp.get(i);
    //replica placement
    if !fp.contain(rp) & success-place-replica(rp,d)
       fp.add(rp);
  }
  //send fp to peer p.
  send_list(p,fp);
}
```

図 5.3　複製配置の手順

時刻から時間 t_{ml} 経過しても p は p_i をシステム上に確認できない場合，p_i は長期間離脱している状況であると呼ぶことにする．複製を配置したピアが長期にわたり離脱の状態にあるために，文書の更新を行うことができない．このとき，オリジナル文書所有のピアは時間 t_{ml} を経過しても複製配置対象のピアの存在を確認できなければ他のピアに文書配置を行うことで長期にわたり離脱しているピアによる弊害を抑えることができる．

表 5.1　各トランザクションの特徴

	ロック	時刻印	多版
デッドロック	発生する	発生しない	発生しない
アボート	○	×	△
並列性	×	○	○

5.5　文書更新に関する議論

本研究ではオープンな協調作業環境下での文書共有を想定しているため，文書の更新が行われる場合がある．そこで文書更新の手法を述べた．しかしながら，上記の文書更新の手法は 1 トランザクションあたり 1 つの文書の write 操作までしか対応していない．アプリケーションによっては分散トランザクションでは 1 トランザクションで複数の文書を更新することを求められる．たとえば，blog ではある文書 1 つを更新する際に，その文書が他の文書を参照しているのであれば参照先の文書の backtrack 情報を更新する必要がある．このようにユーザにとってはある文書 1 つを更新するような操作であっても，実際には複数の文書の更新が必要になるかもしれない．複数の文書を更新する場合，1 つのピア上でその操作が完了するとは限らない場合もある．あるトランザクションで更新される文書はその文書を所有するピアとは別のピアが所有する文書を更新しなければならない場合もある．

分散トランザクションのモデルは UNIX 関連技術の標準化団体 X/Open(The Open Group) によって定義されている．このモデルでは分散トランザクション処理を行うために使用されるのために使用されるソフトウェアの種類，役割および連携方法を示している．このモデルは，トランザクションの開始および終了を管理し，複数のトランザクションを受け付けを管理するトランザクションマネージャ，操作対象となる文書に対するロック管理，もしくは時刻印の管理を行うスケジューラ，および実際の文書の read または write 操作を実行するデータマネージャの 3 階層の構造である．多くの商用データベースでは X/Open の分散トランザクションモデルを用いている．トランザクションでは複数のトランザクションが並列して実行した場合でも ACID 特性と呼ばれる性質が成り立つことが求められる．

- 原子性（Atomicity）: トランザクションが実行された結果は，そのトランザクションが完全に実行された状態か，もしくは全く実行されていない状態かのどちらかの

状態でその処理を終えなければならない．トランザクションの内容が完全に反映されるときコミット（commit）するといい，全く反映されないときアボート（abort）するという．

- 整合性（Consistency）: トランザクションを実行した後，その整合性のとれた状態を保たなければならない．
- 隔離性（Isolation）: 各トランザクションが行う処理の内容が他のトランザクションの影響を受けないように処理する必要がある．すなわち，複数のトランザクションが並列して処理されても，逐次トランザクションを処理した場合と一致しなければならない．
- 耐久性（Durability）: 一度コミットしたトランザクションの操作内容は障害が発生しても取り消されてはならない．すなわち，コミットされた内容は取り消すことはできない．

複数のトランザクションが並列して処理される場合，トランザクションの ACID 特性を満たすための制御は同時実行制御（concurrency control）と呼ばれる．同時実行制御は大きく分けるとロックを用いる方法と時刻印を用いる方法の 2 つに分けることができる．

ロックを用いる同時実行制御は捜査対象となるデータをロックする．ロックには 2 種類あり，read を行う場合に用いる共有ロック（shared lock ; S ロック）と write を行う場合に用いる専有ロック（exclusive lock ; X ロック）の 2 種類がある．ロックを用いる方法では，他のトランザクションが並列して処理していることを意識しないで処理することが可能である．ある文書に対して S ロックがかけられている場合には write 操作ができなくなり，また X ロックがかけられている場合には read 操作もできなくなる．そのため，あるトランザクションが他のトランザクションをロックによってブロックする期間が長くなると並列性が悪くなる．また，データベースの各タプル単位でロックするような場合，すなわち小さい単位でロック場合，多くのロックが必要となりデッドロックが起きる可能性が高くなる．また，隔離性のためには各トランザクションを逐次実行した場合と同様の結果を得る性質である直列可能性（serializability）を確保するために 2 相ロック（two-phase lock）が広く用いられている．2 相ロックでは 1 トランザクション内の各演算に必要なロックを順次獲得して行き，処理を進め 1 つでもロックを解放したならそれ以降ロックを行わない手法である．直列可能性を確保するためにコミットもしくはアボートするまでロックを解除しない 2 相コミットである厳密な 2 相ロック（strict two-phase lock），

もしくはそれを拡張したものが多くのデータベースマネージメントシステム（database managemment system ; DBMS）で採用されている [32].

時刻印を用いる同時実行制御では多くの方法がある．その基本的な方法は各トランザクション，そこでの各操作もしくは各データに時刻印を付加する．その代わりに各データに対するロック処理を行わない．楽観的同時実行制御（optimistic concurrency control）[55] ではコミットするときに write 操作のみを探知し，その操作を行う対象となるデータに対する時刻印を確認し，もし競合するトランザクションの操作がなければ処理をコミットする．もし競合するトランザクションが存在するのであれば，あるトランザクション T のコミットするときの確認時の時刻印が他のトランザクションの開始の時刻印とコミット時の確認の時刻印の間の値である様なすべての T を探し，もし T がその条件を満たす場合，T の write 操作の対象となるデータに対して read 操作を行っているトランザクションがあれば T はアボートする．時刻印を用いる各方法では write 操作が頻繁に行われる場合，ロックを用いる同時実行制御に比べ，アボートするトランザクションが多くなるが，トランザクションの並列性が非常に高くなる．

他の時刻印を用いる方法には多版同時実行制御（multiversion concurrency control）という方法もある．この方法では，あるデータが更新されるたびに版（version）を作成する．この版は時刻印もしくはそれに代わるものとその版のデータによって識別される．Bhattccharya 等 [9] は IBM DB2 Content Manager で用いられている多版同時実行制御の 1 つの方法である loosely coupled トランザクション [42] を改良した方法を用いている．この方法での read 操作の対象はもっとも新しい版になる．また write 操作を行う場合，その操作対象となるデータの最後に更新を行ったときの版 $V(commit)$ を取得し，そのデータの操作によって生成される版 $V(token)$ を抽出する．このトランザクションをコミットするときに，$V(commit) \neq V(token)$ である場合は，このトランザクションを実行している間に新たな版が生成されるため，このトランザクションはアボートされる．また，最後にコミットした版の時刻印 $Tm(commit)$ を取得し，トランザクションをコミットする直前に対象となるデータの最終更新の時刻印 $Tm(access)$ を取得する．もし $Tm(commit) \neq Tm(access)$ であるときも同様にアボートさせる．この方法では並列性の高い同時実行制御を可能にするが，アボートするトランザクションの数は 2 相ロックを用いる方法よりも多くなってしまう．

以上の同時実行制御の各手法の性能を表 5.1 に示す．本研究ではオープンな協調作業システムに P2P システムを適用することを想定しており，そのため更新頻度が高い文書

を扱うことになる．そのため write 操作が多発する状況下で用いることを想定していない楽観的同時実行制御は不向きであると考えられる．しかしながら，ある文書 d に対して write 操作を行うピアが決定している場合，もしくは限られたピアのみが d に対して write 操作を行うようなアプリケーション，たとえばニュース配信システムであれば，楽観的同時実行制御を用いることは非常に有効であると考えられる．そのため，CDN をその応用としている Oceanstore では楽観的同時実行制御を行う Bayou system[26] で用いられているデータ更新の手法を取り入れている [54].

他方ロックを用いる方法では広く商用 DBMS にて用いられている方法である．この方法は頻繁に write 操作が行われる状況下でもトランザクション処理のアボートを他の手法よりも少なくすることが可能である．本研究にてこの手法を用いる場合，ロックは文書単位で行われる．しかしながら，並列して処理されるトランザクション数が多い場合，ロックによってその並列性は失われることになるかもしれない．版を用いる同時実行制御では，ロックを用いる手法と楽観的同時実行制御の中庸な性質を持つ同時実行制御の手法であるといえる．すなわち，アボートするトランザクションの数は楽観的同時実行制御よりも少なく，トランザクションの並列性はロックを用いる同時実行制御よりも高くなる．P2P システムでは各ピアのネットワークでの状況は他の分散システムよりも遙かに変動しやすい．そのため，ロックを用いる同時実行制御は他の方法よりも 1 トランザクションの処理に費やす時間どうしても長くなってしまう．その回避を行うために，短時間でトランザクションを終えるための仕組みが必要になる．その仕組みの 1 つとしてタイムアウトを用いる方法を考えることができ，タイムアウトの時間を調整することでトランザクションの最適化を行うことが可能になると考えられる．

版を用いた同時実行制御ではロックを行わない．また，各トランザクションが write 操作を行うときに，他のトランザクションの干渉をまったく受けない．ロックを用いる方法では read または write の対象となるデータにロックがかけられてる場合一時的にトランザクションを中断しなければならない．また，楽観的同時実行制御では他のトランザクションの開始やデータ確認のときの時刻印を確認する必要がある．PP2P システムでトランザクション処理を行う場合，トランザクションマネージャはそのトランザクションを実行する各ピアであるため，1 つピア上でシステム全体のトランザクションを管理することはできない．そのため，あるトランザクションが操作対象としているデータを他のトランザクションが操作しているかを見知することが困難になってしまう．以上よりオープンな

協調作業環境を P2P システムに適用する場合，多版同時実行制御を用いることが最良の方法であるのではないかと考えられる．しかしながら，多版同時実行制御では更新するたびにそのデータの版が生成される．そのため，頻繁にデータが更新されると多くの版が生成されるため，他の同時実行制御よりも多くの記憶領域を必要としてしまう．P2P システムでは各データは中央に集中して格納されていない．そのため，多くの版が生成されたとしてもそれらはシステム全体に分散して格納されるため，それほどシステム全体の致命的な欠点にはならないと考えている．

各トランザクションでは write 操作の対象となる文書が複数のピアにまたがる場合，一度のコミット命令（one-phase commit ; 1 相コミット）では不都合な場合がある．あるトランザクション T が文書 d_1，d_2 に対して write 操作を行うとする．また，d_1 および d_2 はそれぞれピア p_1，p_2 が所有していると仮定する．T をコミットする場合，p_2 ではコミットを終えたが，p_1 ではコミットできなかった場合，このコミットをロールバックさせる処理を行わなければならない．しかしながら，p_2 ではすでにコミット済みであるため，d_2 をロールバックさせることは不可能である．そのため，整合性および耐久性に問題が生じる．そこでコミット処理を 2 相に分ける 2 相コミット（two-phase commit）[7] が広く分散データベースや分散ファイルシステムにて用いられている [82].

P2P システムにて 2 相コミットを適用する場合にあるトランザクションをコミットさせるときは以下のように行うことができる．その対象となるデータを所有する各ピアに対して，vote メッセージを送信する．このメッセージではコミット処理が完全に終えるまで操作対象となるデータをブロックする．そしてそのトランザクションを取り仕切る調整者（coordinator）に対してコミット可能である状態を知らせるメッセージ（vote-commit）もしくはアボートさせることを知らせる（vote-abort）のどちらかを各ピアは送信する．もし 1 件でも vote-abort メッセージを調整者が受信した場合，対象の全ピアに対して global-abort メッセージを送信し，トランザクションをアボートさせる．vote-abort メッセージが 1 件も無い場合，対象となる全ピアに対して global-commit メッセージを送信し，各ピアは対象のデータコミットし，このトランザクションをコミットし終了させる．このコミット処理では vote メッセージを送受信する投票層とそれを基にトランザクション全体の決定を通達する決定相の 2 相から成り立っている．2 相コミットでは投票相で操作対象となるデータに対して他のトランザクションのいかなる操作もブロックしてしまう．また，調整者がコミット処理中に離脱してしまうと，最終決定が行えない等の問題がある [82].

そこで，調整者がいなくてもトランザクションを終了させるために，トランザクションの終了の “見込み（presumption）” を行う方法 [36, 2] が考案されている．また，コミット処理に用いる見込みの方法を動的に決定する方法 [82] も考案されている．また，1 相コミットと 2 相コミットを動的に使い分けるような仕組みを導入したコミット処理の方法 [3] も考案されている．

上記に示した 2 相コミット処理を実際に P2P システムで処理可能することは可能であろうか．調整者がいなくてもトランザクションを終結させる方法は上記に示したように考案されている．そのため，2 相コミットを P2P システムでのトランザクション処理を終結させる仕組みとして導入することは可能であると思える．検索結果に応じて文書の取得を行い，その後文書を更新するのであれば文書の更新の処理を行う．文書を更新するタイミングに更新対象となる文書を所有するピアがまだネットワーク上に存在することが 2 相コミットを P2P システムで行うための条件になる．そのため，多くのピアで処理することが必要であるような複雑なトランザクションの処理は非常に困難になると予想することができる．そのため，1 トランザクションあたりの write 操作の対象となる文書の数およびその文書を所有するピアがどれくらいであれば可能であるのかを見極める必要がある．

5.6　本章のまとめ

本章では分散インデックスでは解決できない問題である文書消失の防止および “有用度” の高い文書の発見の助力のため文書複製の方法を示した．提案した文書複製の方法では Partial-Replication を行う．そこで文書の複製のために（1）複製対象となる文書の決定,（2）複製回数の決定，および（3）複製配置の場所の決定の方法を示した．複製配置場所の決定のために，動的にピアグループ形成し複製を配置する，PN 方式，PIR 方式および PRP 方式を提案した．さらに文書更新のメカニズムを簡略化するために Lazy Master Replication を行う文書更新の方法を示した．

第 6 章

実験

6.1 実験の概要

本章では提案した分散インデックスである DI および PN，PIR，PRP の各複製配置方法による問い合わせ処理へ与える効果をシミュレーションによって示す.

提案する分散インデックスである NDI 及び TDI の効果を明らかにするために，各ピアの文書の情報を格納する Routing Index(RI)[19] を用いた検索方法及びフラッディングによる検索方法（Gnutella 検索）と比較した．また，文書複製の配置の効果を確かめるため，5 章で与えた各複製配置方法及びデータ配置を行わない No-Replication の 5 つの方式を比較する．各手法で用いられるパラメータを表 6.2 に示す.

本章で行ったシミュレーションモデルは以下の通りである．用いたネットワークトポロジはサイクルの存在する木構造とした．各ピアの隣接ピアの数の上限 F を 4 とする．初期データ分布として偏在のあるデータ分布を用いた．実際にはシステム全体の 80% のデータをシステム上の 20% のピアが保有するデータ分布を用いた．この実験では全文書のサイズは同じであると仮定している．検索モデルとしてブーリアンモデルを用いた．P2P システムの問い合わせ処理では，同一の文書を異なるピアから受け取ることがある．本研究では同一の文書を複数個受け取った場合，1 つの結果を得たものとした.

本実験では，ピアの操作をシステムへの参加状況の変化（システムへの参加・離脱および他のピアへの接続・切断），問い合わせ処理，文書取得・文書配置，文書の更新・作成とし，各ピアは，1 単位時間に上記の操作を指定された確率で行うものとした．各ピアはシステムへ再び接続を行うとき，最後にシステムへ参加していたときの隣接ピア，これらのピアに対して接続できない場合は類似の問い合わせメッセージを出していたピア，その

表 6.1　実験で用いたネットワークおよびピアに関するパラメータ

定数名	説明	標準値
	システム	
F	ファンアウト	4
T	ネットワークトポロジ	Tree + Cycle
num_peer	ピア数	10,000
$NumDoc$	文書数	30,000
DD	文書の分布	80/20
num_topic	トピック数	20
$MaxUp$	文書の最大アップロード数	4
$MaxDown$	文書の最大ダウンロード数	4
	ピア	
$join\_ratio$	接続の確率	0.20
$leave\_ratio$	切断の確率	0.10
$join\_net\_ratio$	ネットワークへの参加の確率	0.006
$leave\_net\_ratio$	ネットワークからの離脱の確率	0.006
QM_ratio	問い合わせ，文書更新を行う確率	0.2
$query\_ratio$	QM_ratio での問い合わせを行う確率	0.8
$modify\_ratio$	QM_ratio での文書修正を行う確率	0.16
$create\_ratio$	QM_ratio での文書作成を行う確率	0.04
	問い合わせ処理	
$StopCondition$	問い合わせの終了条件	20
$query\_size$	問い合わせメッセージの平均サイズ	82B
$RetrievalType$	検索モデル	ブーリアンモデル

他のピアの順序で接続を行おうとする．また切断を行う場合，各ピアは過去の問い合わせにおいて平均検索結果数が少なかったピアに対して切断する．また各ピアには嗜好するトピックを持っており，問い合わせを行う場合，そのトピックに関する問い合わせを行う確率は 0.6 とし，文書を生成する場合は常にそのトピックに関する文書を生成するものとした．

各ピアがシステムへ参加する手順は次の通りである．まず，システムへ参加するピアが何らかの文書の複製を所有している場合，そのピアが所有している全複製を非同期の状態

表 6.2 各分散インデックスおよび複製配置に関する実験で用いたパラメータ

Diret Index		
TTL	Time To Live （TTL）	8
m	推奨ピア数	4
$UpdateSize_{NDI}$	NDI のインデックス更新メッセージ平均サイズ	89B
$UpdateSize_{TDI}$	TDI のインデックス更新メッセージ平均サイズ	108B
Routing Index		
TTL	Time To Live	40
$Type\_RI$	RI の種類	ERI
$minUpdate$	インデックス更新のための閾値	1%
$UpdateSize_{RI}$	RI のインデックス更新メッセージ平均サイズ	105B
Gnutella		
TTL	Time To Live	8
複製配置		
$search\_type$	検索方法	Gnutella
$replica\_repository$	最大複製保有数	20
c	1 ピアあたりの平均複製数	8
$wait\_leave$	切断状況の待ち時間	10
ref_log	参照ピア用スタックの大きさ（PIR）	10
$query\_log$	問い合わせメッセージ記録件数（PRP）	10

にする．次に，隣接ピアを既知であるピアから選択する．もしその選択したピアと隣接することが可能であるならば，各インデックスを用いた場合でのピア間の接続のための手続きを行い，そのピアと隣接する．各ピアがあるピア p と隣接することが可能であるためには，p が

- システムに参加している
- F 未満の隣接ピアを保持している

状態である必要がある．以上の手続きによってそのピアはシステムに参加している状態になる．システムに参加したピアはそのピアが何らか複製を所有している場合，各複製の複製情報があるかを確認し，もしある複製に対する更新があれば更新された文書の複製を再配置する．

各ピアはあるピアと隣接する場合，そのピアのピアリストから隣接ピアを選択するが，その手順は次の通りである．まず，前回システムに参加していたときに隣接していたピアを隣接ピアの候補として選択する．もし，前回システムに参加していたピアを隣接ピアとして選択できない場合，そのピアの過去の検索結果の情報を用いる．過去の検索結果の情報とは，いままでにどのピアから 1 問い合わせあたり何件の検索結果を得たかという情報である．そして，あるピアは過去の検索結果からもっとも検索結果数を返すピアを選択する．もし，あるピア p はいままでにピア p_1 から 5 件の検索結果を取得，ピア p_2 から 10 件の検索結果を取得していたのであれば，p はまず p_2 を隣接ピアとして選択しようとする．もし，候補が複数ある場合はランダムに決定する．

あるピアがそのピアの隣接ピアと切断する場合，その対象となるピアを選択する基準として隣接してからの検索結果の情報を用いる．この検索結果の情報ではそのピアと隣接してからの情報である．この情報は今までの全検索結果の情報ではなく，あるピアと隣接したとき以降での検索結果の情報である．隣接ピアと切断する手順は次の通りである．あるピア p の隣接ピアが p_1，p_2，p_3 であるとする．p はそれぞれの隣接ピアと接続してからの 1 問い合わせあたり検索結果数がそれぞれ 3，5，4 であれば p は p_1 を切断するピアとして選択する．

表 6.1 は実験全体でのパラメータを示している．この表に示されている各変数の値はその変数を固定した場合の値である．これらのパラメータは主に 2 つのカテゴリーに分けることができる．1 つは，システム状況を変化させるパラメータであり，ピア数およびトピック数がある．問い合わせ処理を行う場合，これらの値を変えることで，システム全体の文書数を増減させ，問い合わせ処理の状況を変化させる．もう 1 つは，ピアの各メソッドの実行頻度である．ピアの各メソッドの実行頻度を変化させることによってシステム全体のピア数の増減や，ネットワークの安定性，システム全体の文書数等の問い合わせ処理状況を変化させた．本研究では接続，切断の頻度を高く設定した場合，文書作成以外の他のメソッド実行頻度を相対的に低下するように設定した．また，問い合わせ処理の実行頻度を高く設定した場合，文書の更新の頻度を相対的に低く設定した．

6.2　実験結果

この節では，本研究で第 4 章で提案した分散インデックスである DI，第 5 章で提案した複製配置の手法（PN，PIR および PRP），およびそれらの組み合わせが問い合わせ処

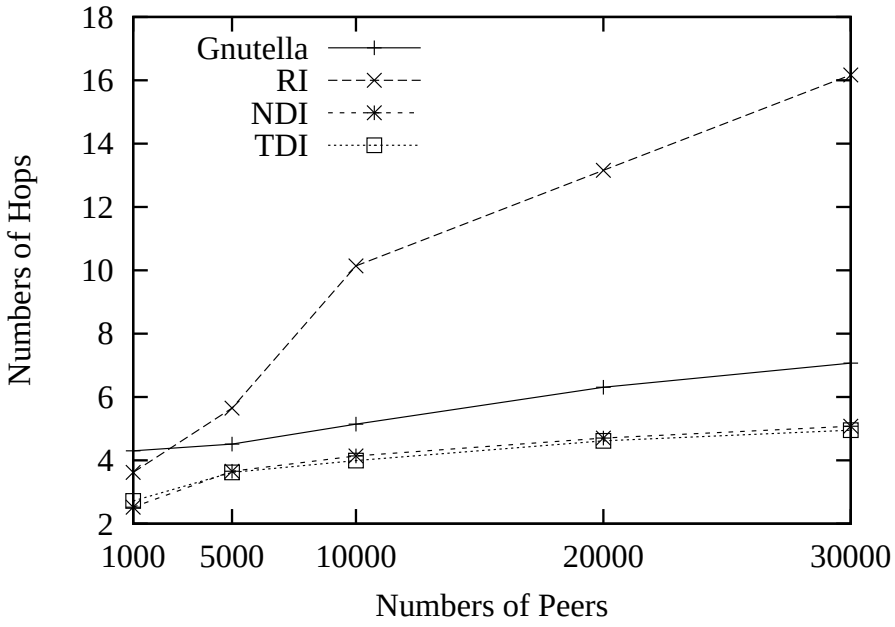
理に与える効果を示す．以下の各実験では変動させる変数はただ 1 つである．変動させない変数の各値は表 6.1 および表 6.2 を用いる．

6.2.1 分散インデックスの効果

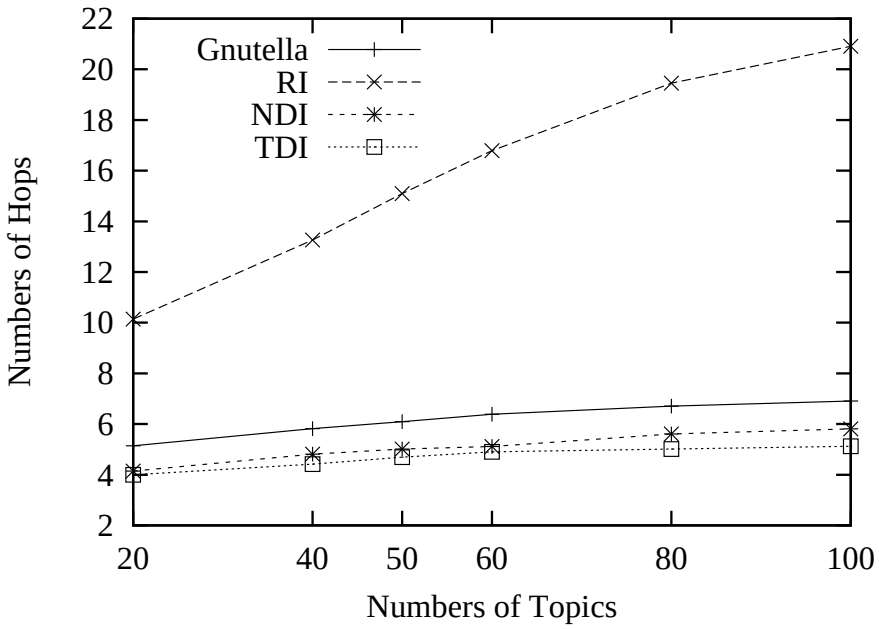
DI の効果を効果を確かめるために応答時間，帯域幅消費量および問い合わせの成功率に注目してシミュレーションを行う必要がある．そこで分散インデックスの効果を確かめるために以下の項目に注目した．

- 問い合わせ処理終了までのホップ数
- 問い合わせメッセージ数
- 問い合わせの成功率
- インデックス更新メッセージ数

PP2P システムは HP2P システムよりも特定の各ピアへの問い合わせメッセージの集中は起こりにくい．問い合わせ処理での応答時間はピア上での処理時間とピア間での問い合わせメッセージの転送時間で計算する．しかしながら，ピア上での処理時間はピア間での問い合わせメッセージの転送時間よりも著しく少ないため，問い合わせ処理の応答時間はネットワーク上での問い合わせメッセージの転送回数，すなわち問い合わせメッセージのホップ数によって代替することができる．そのため，問い合わせメッセージに関する帯域幅消費量はそのメッセージ数に比例する．そこで問い合わせに関する帯域幅消費量の計測は問い合わせメッセージ数の計測に代替する．また，インデックス更新メッセージの帯域幅消費量に関しても同様にインデックス更新メッセージ数にて代替する．問い合わせの成功率は問い合わせを終了できた割合で示すことにした．本実験では問い合わせの終了条件（*StopCondition*）を 20 件以上の検索結果の取得と設定した．PP2P システムではシステムの中央にサーバや super peer が存在しない．そのため，PP2P システムにはシステム全体の様子を管理や関知しているサーバや super peer が存在しない．PP2P システムでは，ピア間のコミュニケーションによって問い合わせメッセージの伝播は行われる．ピア数が増加された場合，各ピアからの問い合わせメッセージがスケーラブルに伝播されることが重要である．そこで，ピア数の増減に関して注目することにした．システム内のユーザ数の増加や議論の幅が広げられたとき，または新規文書の追加，文書の修正等の要因にてトピック数が増加すると考えられる．そこでトピック数の増減に関しても注目することにした．



(a) ピア数変動の影響



(b) トピック数変動の影響

図 6.1　各分散インデックスでの応答時間

応答時間

初期の文書数は固定しているため，ピアの増加に伴い各手法のパフォーマンスは低下する．図 6.1（a）はピア数を変動させたときの各分散インデックスを用いた場合での一定以上の検索結果を得るのに必要であった問い合わせメッセージのホップ数を示す．Gnutella はフラッディングすなわち幅優先探索を行う．そのため問い合わせメッセージが 1 ホップすると F^h 個のピアへ転送される（ここで F はファンアウト数，h はホップ数を示す）．問い合わせメッセージの転送対象のピア数が多いため，応答時間は短縮される．RI は深さ優先探索を行う．そのため，各ホップでの問い合わせメッセージの転送先のピアの数は一定数であり Gnutella と比べて非常に少ない．そのため RI を用いた場合，多くのピアへ問い合わせメッセージを転送するために多くのホップ数を必要とする．DI を用いた場合でも各ホップでの問い合わせメッセージの転送先のピア数は一定数である．しかしながら，NDI 及び TDI はピア数が増加しても，応答時間をフラッディングによる方法の約半分程度に低減することが可能であった．NDI 及び TDI を用いた場合，各ピアは多くの “有用” な文書を保有しているピアに対して直接問い合わせメッセージを送信することが可能であるため，ピアはフラッディングによる問い合わせ処理に比べ応答時間の短縮が可能である．またこれらの DI を用いた場合，遠くに位置する “有用” なピアに対しても問い合わせメッセージを送信することが可能であるため，スケーラブルに問い合わせを行うことができる．ピア数が 1,000 であるとき NDI と TDI のパフォーマンスにあまり差はないが，ピア数の増加に伴い，TDI は NDI よりも優れたパフォーマンスを示す．これは TDI は問い合わせメッセージの送信先をトピックに対する “有用度” を用いて修正することが可能であるためである．

次にシステムのトピック数を変化させた場合での応答時間の結果を図 6.1（b）に示す．トピック数を増加させた場合，各トピックあたりの文書数は減少するため，問い合わせ処理のパフォーマンスは低下してしまう．Gnutella のフラッディングによる問い合わせ処理は，総当たりに問い合わせメッセージを伝播させるため応答時間はトピックスの影響をさほど受けない．RI を用いた場合，問い合わせを行ったピアから近い位置にあるピアに問い合わせとマッチする検索結果があれば効果的なパフォーマンスを示すが，各トピックあたりの文書数は減少するため，より多くのホップ数で問い合わせを終了しなければならなかった．NDI を用いた場合，トピックの増加の影響を多大に受けてしまう．NDI を用いた場合では問い合わせメッセージを転送したピアはその転送先の候補となるピアが問い合

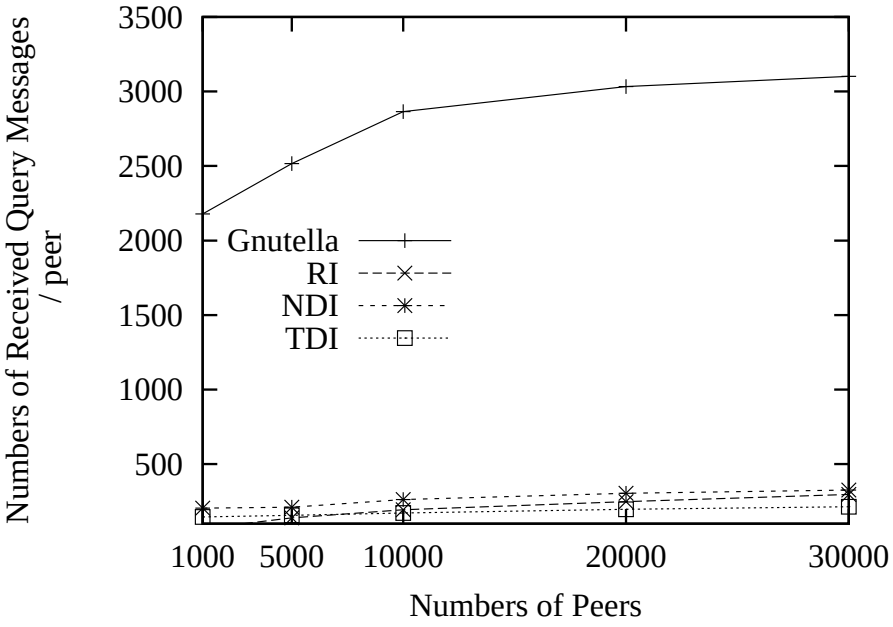
わせにマッチする文書を持っているかどうかを判断できない．そのため，NDI のパフォーマンスは悪化し Gnutella のパフォーマンスに近い結果を示した．TDI はトピックに関して最適であると思われるピアの位置を示すことが可能である．しかも，隣接していないピアの位置であっても示すことができる．そのため，トピック数の影響をあまり受けずに検索結果を一定以上集めることが可能になる．

問い合わせメッセージ数

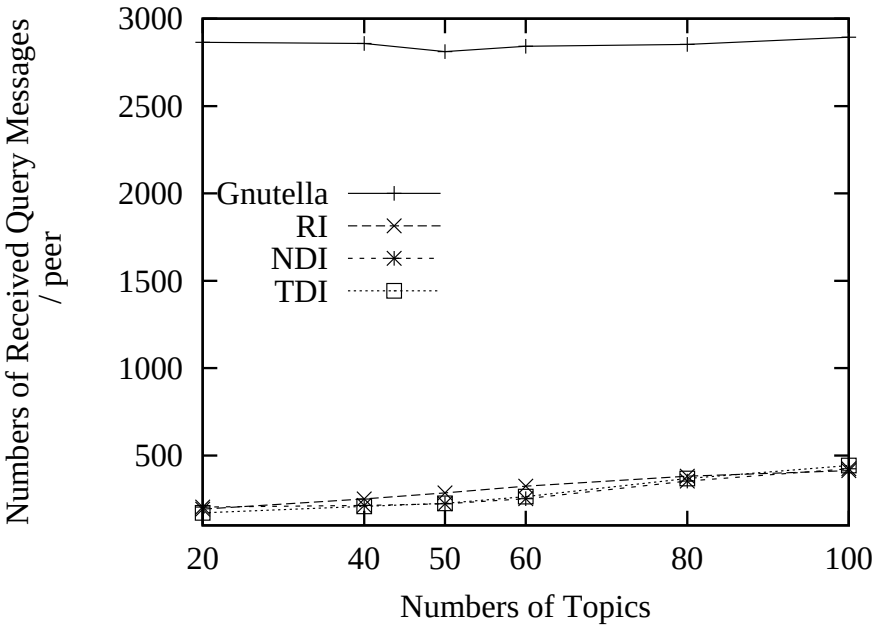
図 6.2（a）は 1 ピアあたりの受信した問い合わせメッセージ数を示す．Gnutella では TTL で指定された範囲に存在する全ピアへ問い合わせメッセージが伝播される．そのため Gnutella での検索方法では問い合わせメッセージを各ピアは非常に多く受信する．これに対し，RI，NDI および TDI は問い合わせメッセージ数を低減するすることが可能であった．これらを用いた問い合わせ処理では各ピアはおのおののインデックスを用いることで，問い合わせメッセージの転送先を選択し，無意味だと思われる問い合わせメッセージの転送を低減することが可能である．

図 6.1（a）および図 6.2（a）から以下のことが分かる．Gnutella に代表される幅優先探索は問い合わせを行っているピアから一定範囲内の全ピアに対して問い合わせメッセージを伝播させる．そのため応答時間を短縮することができるが，無用な問い合わせメッセージが多く存在してしまう．Freenet や RI に代表される深さ優先探索では問い合わせメッセージをマルチキャストしないため，一定以上の検索結果を得るために必要なホップ数が多くなるが，無用な問い合わせメッセージを低減することが可能である．これらの探索方法に比べ，DI を用いた問い合わせ処理では応答時間および問い合わせメッセージ数の低減が可能になる．しかも，ピアの増加に対してそのパフォーマンスを減少することが無かった．それは，ネットワーク上のどの位置にあるピアにもそのピアが問い合わせにて必要とする他のピアに関する情報を得ることができる仕組みを提供し，遠く離れたピアにも問い合わせメッセージを転送することができるからである．

図 6.2（b）はトピック数を変動させた場合での 1 ピアあたりの問い合わせメッセージ数を示す．トピック数の変動があまり影響されなかった Gnutella であるがこの傾向は問い合わせメッセージ数でも同様である．Gnutella でのフラッディングの問い合わせ処理はその問い合わせ処理を停止させる方法は唯一 TTL のみであるため，Gnutella での問い合わせメッセージ数はトピック数の影響は皆無である．トピック数の影響を受けやすい NDI であるが，トピック数を増加させてもその問い合わせメッセージ数は RI および TDI



(a) ピア数変動の影響



(b) トピック数変動の影響

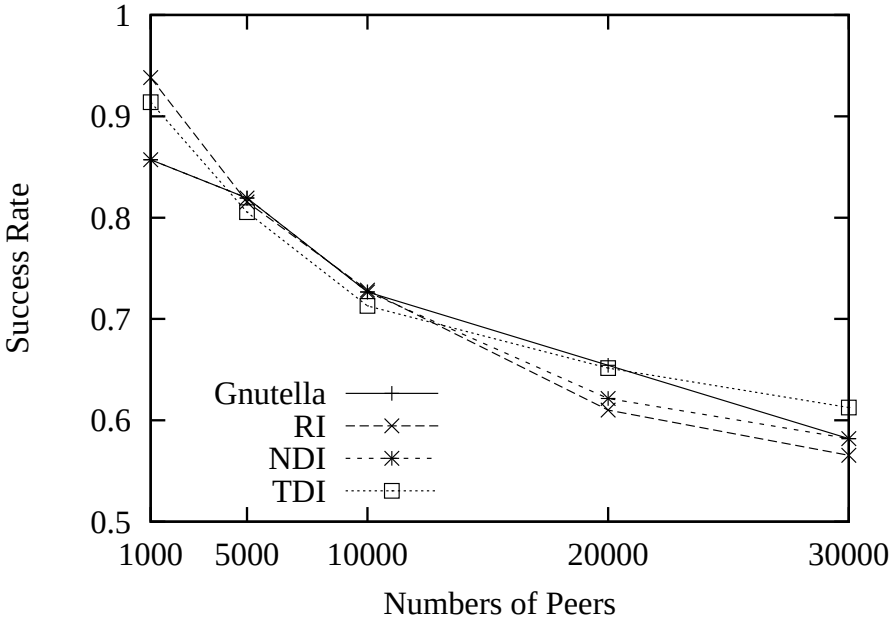
図 6.2 各分散インデックスでの 1 ピアあたりの問い合わせメッセージ数

とあまり変わらない．DI を用いた問い合わせ処理では，問い合わせを行うピアは各ホップにて転送対象となるピアの情報を新たに得るが，新規に得られる “有用” なピア情報が必ずしも得られていないと考えられる．そのため，各ホップで転送対象となるピアの数が F よりも小さくなってしまうと考えられる．そのため，RI や TDI と同程度の問い合わせメッセージ数を各ピアは受け取るが，応答時間は TDI よりも劣る結果を示したと考えられる．TDI は問い合わせの内容に従って，問い合わせの転送先を修正することが可能である．そのため，NDI を用いた場合よりも効率的に “有用” であるピア情報を得られることが可能であったと考えられる．

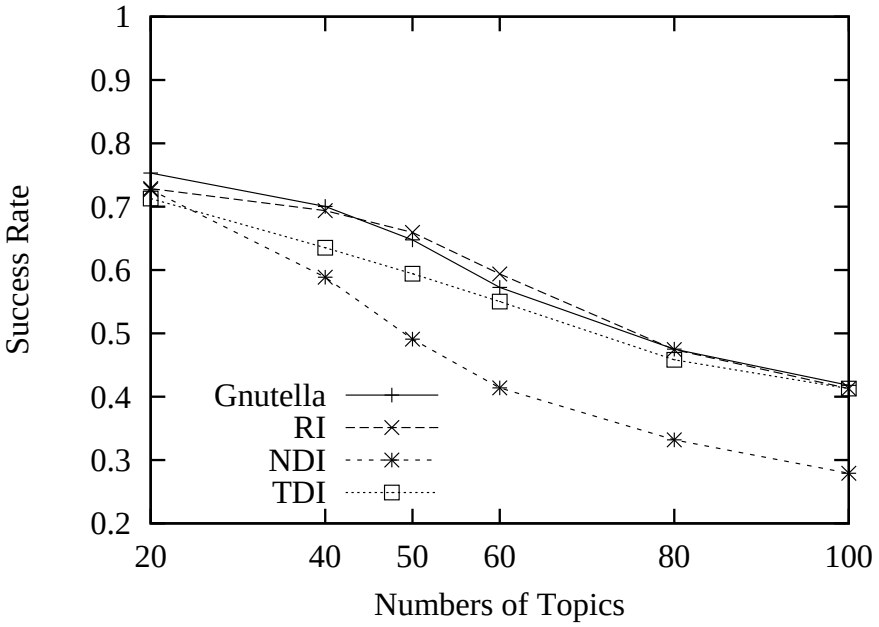
問い合わせの成功率

図 6.3（a）はピア数の増加に対する問い合わせの成功率を示す．NDI 及び TDI はピア数の増加に伴いそのパフォーマンスは低下するが，RI や Gnutella とあまり変わらないパフォーマンスを示す．ピア数が 30,000 であるとき，TDI を用いた問い合わせでは Gnutella や RI よりも成功率が高いことが分かった．TDI での問い合わせメッセージが伝播される範囲が Gnutella や RI でのそれよりも広いことが分かる．Gnutella では各ピアは問い合わせを行ったピアから一定範囲内に位置する全ピアへ問い合わせメッセージを伝播するが，その範囲よりも遠くのピアには問い合わせメッセージを伝播することはできない．RI では伝播されるインデックス更新メッセージは距離が遠くなるほどその効力が減少する．そのため，遠い位置に存在するピアに問い合わせを転送することが困難になる．ピア数が多いが 1 ピアあたりの保有する文書数が少ない状況下（ここではピア数が 30,000 のとき）ではスケーラブルに問い合わせメッセージが伝播可能な仕組みが必要であり，このための仕組みを DI は提供することが可能なる．

“有用” なピアの情報を格納する NDI であるが，トピック数が増加すると，各ピアとそのピアが DI に格納しているピアの嗜好するトピックが同じである確率は低下する．そのため格納しているピアが所有している文書をそのピアを DI に格納しているピアが嗜好する確率も低下する．そのため，図 6.1（b）および図 6.2（b）で示したように応答時間及び問い合わせメッセージ数に関するパフォーマンスは低下してしまった．問い合わせ処理中に “有用” ではあるがその問い合わせに関してはあまり類似した文書を所有していないピアの情報も NDI は得てしまうため，トピック数が増加した場合では NDI での問い合わせの成功率も低下してしまう結果を得た．これを図 6.3（b）に示す．トピック数の増加に伴い，各トピックに属する文書数は減少してしまう．そのため，各手法とも問い合わせの成



(a) ピア数変動の影響



(b) トピック数変動の影響

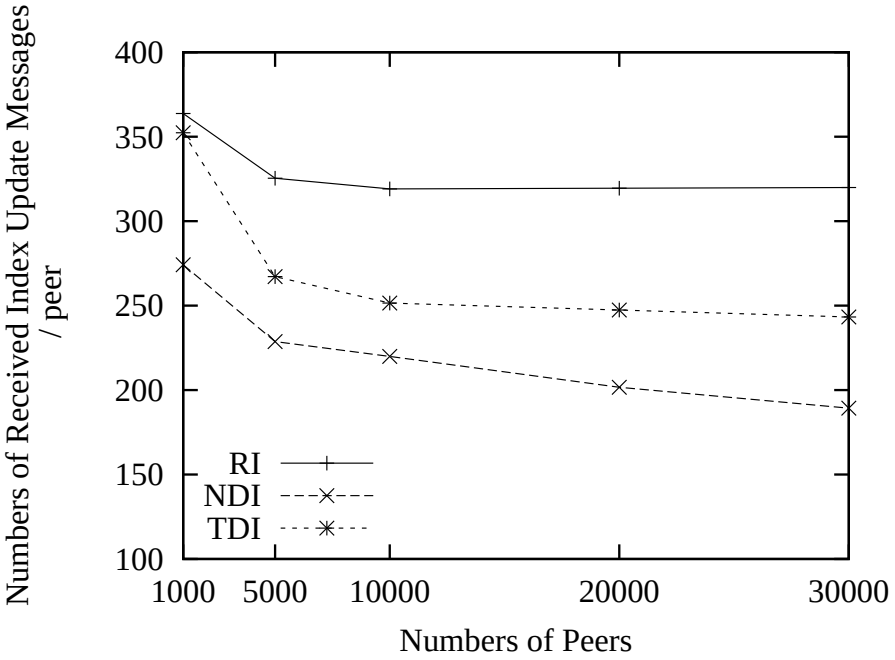
図 6.3 ピア数の変動に対する各種分散インデックスでの問い合わせの成功率

功率は低下してしまう．特に上記の理由より，NDI を用いた場合では Gnutella，RI および TDI に比べそのパフォーマンスを大きく低下させてしまう．また，インデックス更新の頻度も高くなってしまう結果を得た．各ピアは一定の確率で他のピアと切断する．TDI を用いた場合では，Gnutella や RI と同程度の問い合わせの成功率であった．これは，問い合わせメッセージの転送先の修正が TDI によって修正可能であったことが分かる．

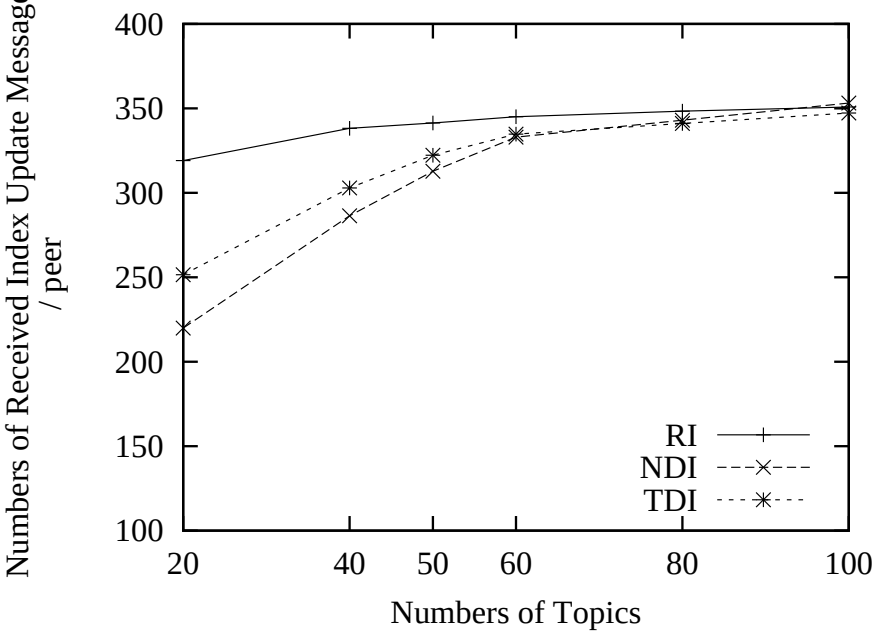
インデックス更新メッセージ数

図 6.4（a）はピア数を変動させた場合での 1 ピアあたりの受け取ったインデックス更新メッセージ数を示す．Gnutella ではインデックス更新の必要がない．そこで RI，NDI，TDI で比較した．NDI を用いた場合，最も更新メッセージ数が少なかった．NDI はトピックに対する情報を格納しない．そのため RI や TDI と比較した場合，NDI はインデックスの更新に対してあまり敏感ではないといえる．対照的に，RI は NDI に対しインデックスの更新に敏感であることが分かる．RI ではあるピアのインデックス更新は前回のそれと比べ 1% 以上の差異が生じればインデックスの更新を行い，さらにインデックス更新メッセージを送信する．TDI は各トピックの “有用度” の変化に対して敏感であるが，“有用” ではないピアのインデックス更新情報の伝播をあまり行わないため，RI と NDI の中間的な更新頻度であった．

図 6.4（b）はトピック数を変動させた場合での 1 ピアあたりの受け取ったインデックス更新メッセージ数を示している．前述したように RI はインデックス更新に対して非常に敏感である．RI を用いた場合，各ピアがインデックス更新の情報を受信した場合にそのメッセージによって更新される値が更新前に比べどのくらい変化するかによってさらなるインデックス更新が必要になるかどうかを判定する．文書数が一定でトピック数だけ増加させると各トピックに属する文書数は減少するためにインデックス更新によるインデックス内の各トピックの値（“*goodness*”）の変動はトピック数の増加に応じて頻発するようになる．たとえば，あるピアのトピック A の値が 100 であった場合に A の値が $+1$ される場合と，A の値が 10 であった場合にその値を $+1$ するのであれば，後者の方が A の値の変化率は高い．そのためトピック数の増加によってインデックス更新の頻度はさらに高くなるはずである．しかしながら RI のインデックス更新では各ピアの情報は広範囲に伝播しない．それは RI でのインデックス更新時に送信される各トピックの値は 1 ホップごとに小さくするように設定されているためである．たとえトピック数の増加によって各トピックの値が頻繁に変化がしたとしても，その影響は広範囲に伝播しないため，RI の



(a) ピア数変動の影響



(b) トピック数変動の影響

図 6.4 ピア数の変動に対する各種分散インデックスでの 1 ピアあたりのインデックス更新メッセージ数

インデックス更新の仕組みはトピック増加の影響をあまり受けない．TDI では各ピアはその隣接ピアごとに推奨するピアの情報を適宜記録し，インデックス更新の判断は推奨するピアが変化すればインデックス更新メッセージを送信する．そのためトピック数の増加は TDI でのインデックス更新を誘発する．そのためトピック数が増加した場合，そのインデックス更新頻度は高くなる．NDI はトピックに関する情報を格納しないが，そのインデックス更新の頻度はトピック数の変動に影響されうるという結果を得た．NDI を用いた場合，*leave_ratio* に従って隣接ピアに対して切断処理を行う．このとき，最も検索結果が良くないピアに対して切断する．各ピアが他のピアに対して切断を行う場合，各ピアは過去の検索結果を下に切断対象となるピアを選択する．この状況下で NDI を用いた場合，“有用度” の高いピアへ問い合わせメッセージを送信したとしても，多くの検索結果が得られるとは限らないため，“有用度” の高いピアを推奨したピアに対して切断する可能性が高い．よって，インデックス更新の頻度は高くなってしまう．NDI および TDI でのインデックス更新では “有用” ではないピアの情報はあまり伝播しない．これはトピック数の変動に影響されない．そのため，インデックス更新メッセージの伝播は制限されるものが多いため，トピック数が 100 の状況下でも RI と同程度のインデックス更新メッセージ数であった．

6.2.2 複製配置の効果

ここでは複製配置の効果を明らかにするために，以下の観点から実験を行った．

- 1 問い合わせメッセージあたりの検索結果数：
  フラッディングによる問い合わせ処理によって得られる検索結果数を計測した．このとき，各問い合わせメッセージの伝播は TTL によって制限した．
- 問い合わせのレスポンスタイム：
  一定以上の検索結果を得るために要したホップ数を用いた．以下の理由のため，本実験では，50 件以上の検索結果数を取得するために要したホップ数を計測した．この実験では任意のピア間での通信速度は一定であると仮定しており，また，P2P システムでの問い合わせ処理の多くの時間はネットワーク上の通信時間である．

検索結果数および応答時間に着目する．6.2.1 節と同様に応答時間を一定以上の検索結果を得るために必要であった検索結果にて代替する．この実験では各複製配置の特徴を示す

ため，以下のパラメータに着目した．この節ではこれらのパラメータの変動が問い合わせ処理にどれだけ影響を与えるかを示す．

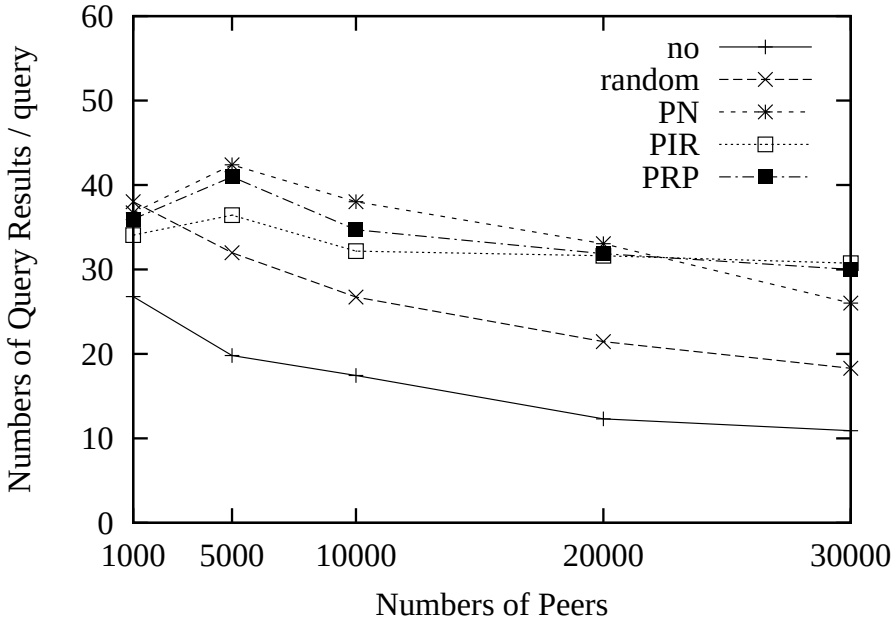
- ピア数：num_peer
- トピック数：num_topic
- 1 ピアあたりの複製配置数：c
- ピアの接続・切断処理の頻度：$join\_ratio$，$leave\_ratio$
- 問い合わせ処理の頻度：$query\_ratio$

ピア数の変動の影響

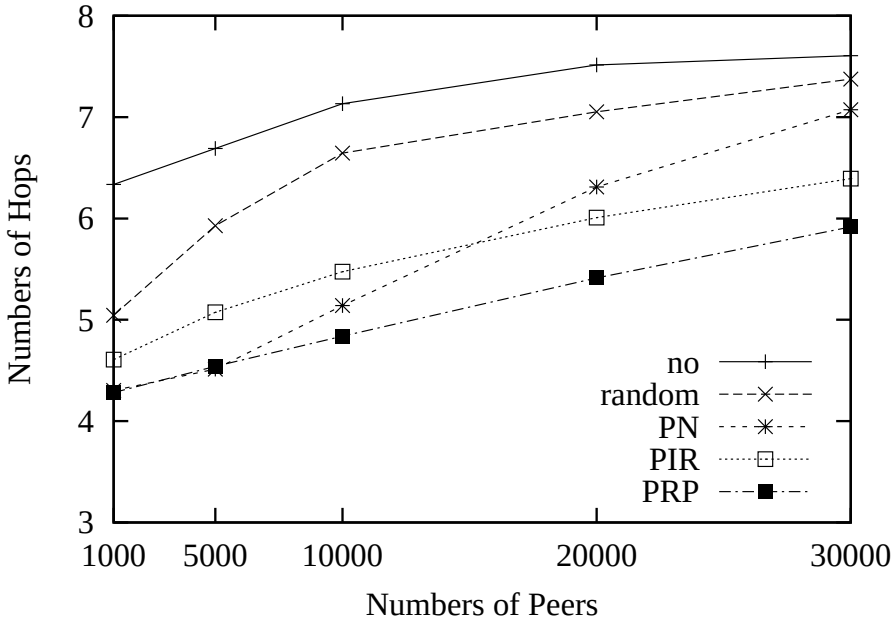
図 6.5 は，ピア数の変動による問い合わせ処理への影響を示す．ピア数の変動はシステム全体の文書数に変化がないため，各配置方式のパフォーマンスは，ピア数の増加に伴い低下する傾向にある．ただしピア数が 1,000 のとき，No-Replication 以外の配置方式では，ピア数が 5,000 の場合に比べ検索結果数でのパフォーマンスは低下している．各文書の複製数はピアごとに設定されるため，“有用度” が高い文書であっても，その文書が複製されない可能性がある．すなわち，ピア数が少ないために，効率的な文書配置が行われなかったと考えられる．

PN 方式を用いた場合，ピア数が少ない場合に対して有効であるが，ピア数を増加するに従って，そのパフォーマンスは低下した．本実験では，各ピアは同じ嗜好の問い合わせメッセージを送信するピアに対して接続をしようとした．PN 方式を用いた場合，複製配置対象となるピアは隣接ピアであるため，各ピアが嗜好する文書が近い距離に位置するピア上に存在する傾向がある．これはピア数が少ない場合には非常に有効である．PN 方式では，各ピアから遠い距離に位置するピアに配置されている文書がそのピアによって嗜好される確率は低い．各文書は局所的に配置されるため，ピア数の増加に従って，各ピアによって発見できる文書の数は減少する．よって PN 方式はスケーラビリティに問題があることが分かる．ランダム方式は，問い合わせメッセージを送信したピアから遠い距離に位置するピア上にも嗜好する文書がいくらか配置されるが，ピア数の増加に伴い，嗜好する文書が配置されている確率は低くなるため，そのパフォーマンスは低下する．

PN 方式及びランダム方式に対し，PIR 方式及び PRP 方式では，ピア数の影響をあまり受けなかった．その要因として，これらの方式では，PN 方式と比較した場合，各ピアから離れた距離に位置するピア上にも，そのピアによって嗜好される文書が配置されるた



（a）検索結果数



（b）応答時間

図 6.5　*num_peer* の変動による問い合わせ処理への影響

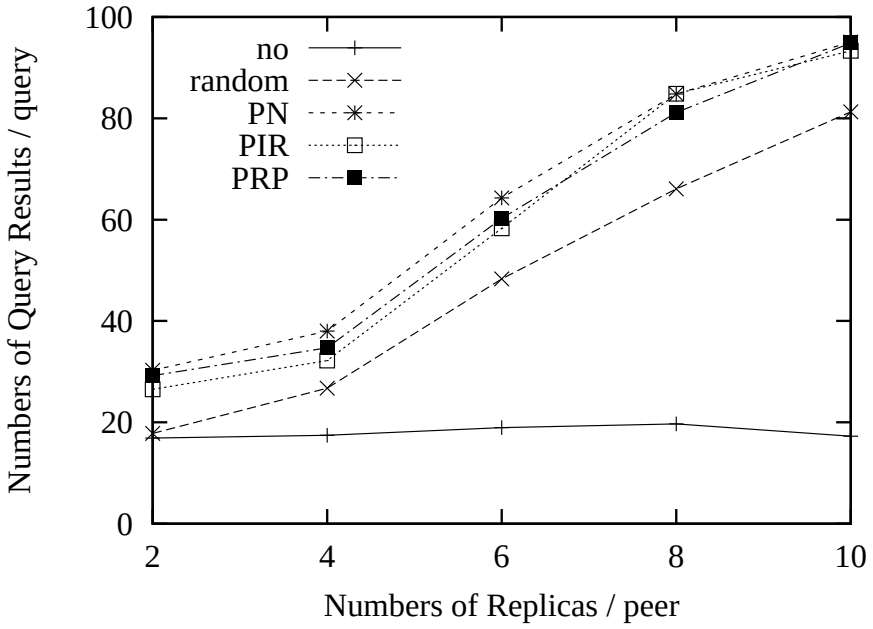
めと考えられる．PIR 方式は，ピア数の増減に関わらず，PRP 方式よりも低いパフォーマンスであったが，両方式はともに同様の傾向を示した．ピア数の増加に伴い，PRP 方式と PIR 方式のパフォーマンスの差は，わずかではあるが縮小する．PRP 方式を用いた場合，ピア p から離れたピア p_i 上に p によって嗜好される文書が配置されている確率が高いが，p と p_i の距離が遠くなるに従って，p が嗜好する文書が p_i に配置されている確率は低くなると予想される．PIR 方式を用いた場合も同様であると予想されるが，各ピアから離れて位置するピアにそのピアによって嗜好される文書が配置されている確率は，PRP 方式の場合よりも高いと予想できる．これは以下の理由のためである．ピア数が多い状況下では図 6.6（a）より PIR 方式を用いた場合と PRP 方式を用いた場合では 1 問い合わせあたりの検索結果数にはあまり差がない．しかしながら，図 6.6（b）から PIR 方式を用いた場合よりも PRP 方式を用いた場合の方が 0.4 ホップ早く一定以上の検索結果を取得することが可能であることがわかる．

複製数の変動の影響

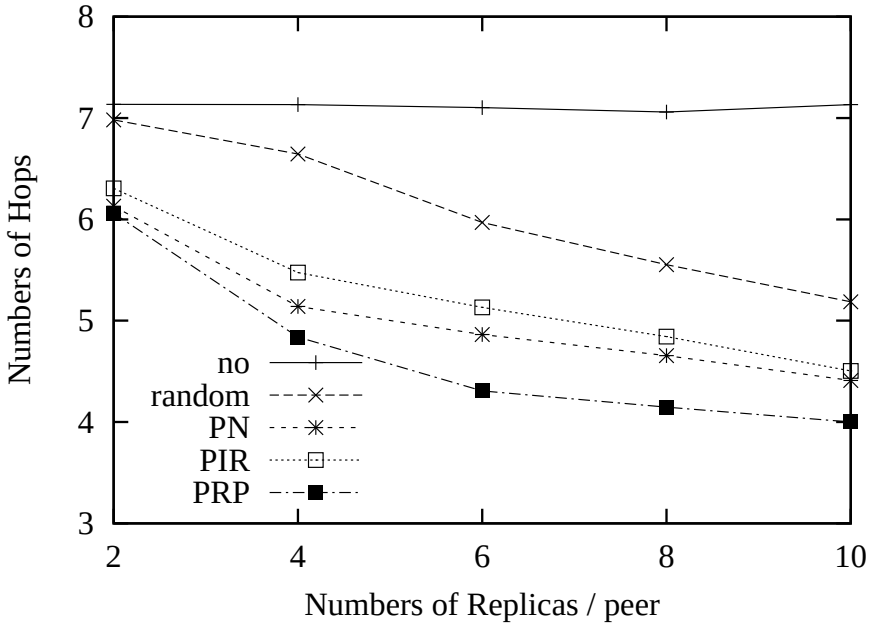
図 6.6 は 1 ピアあたりの複製数の問い合わせ処理への影響を示す．1 ピアあたりの複製数を増加させた場合，システム全体の文書数は増加する．システム上に多くの複製が存在する場合，提案した 3 方式ともそのパフォーマンスが優れていることが分かった．ただし，PN 方式を用いた場合，各ピアと近い距離に位置するピアに多くの同一の複製が配置されている．そのため，PN 方式を用いた場合の応答時間は PRP 方式よりも劣る．

トピック数の影響

図 6.7 はトピック数の変動による問い合わせ処理への影響を示す．トピック数を増加させることによって，1 トピックあたりの文書数は低下する．PN 方式及びランダム方式は，トピック数が少ない場合に，優れたパフォーマンスを発揮するが，トピック数の増加に伴い，そのパフォーマンスは低下した．これに対し，PIR 方式及び PRP 方式では，PN 方式及びランダム方式と比較して，そのパフォーマンスはあまり低下しなかった．トピック数が増加すると同じ嗜好であるピアの数は減少するため，各ピアは同じ嗜好であるピアを隣接ピアとして選択することが困難になる．そのため，PN 方式ではトピック数の増加に伴い，隣接ピアが同じ嗜好を持つ確率が低下し，各ピアから近い距離に位置するピアに配置されている文書がそのピアによって嗜好される確率は低下してしまう．これに対し，PIR 方式及び PRP 方式では，各ピアから離れたピア上にも嗜好する文書が配置されてい

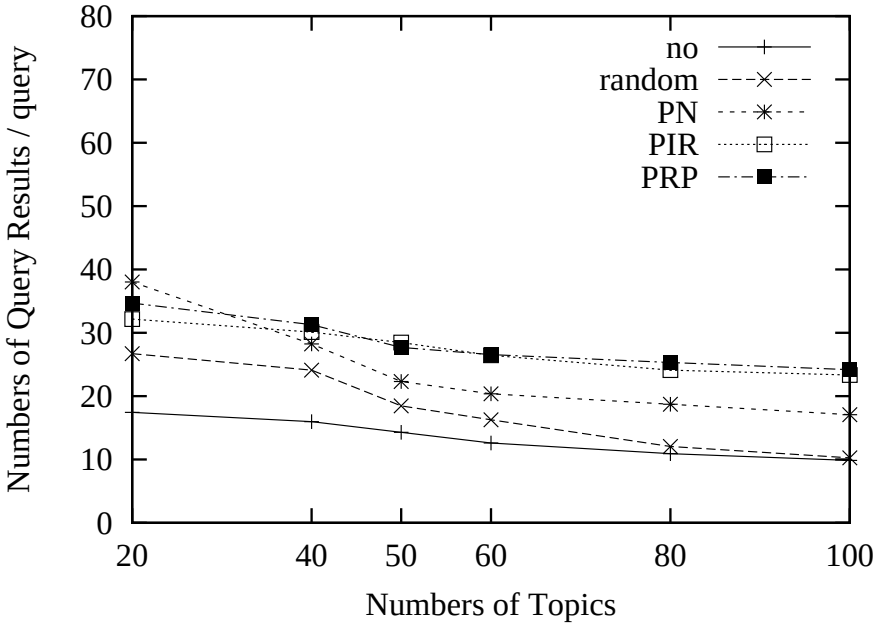


（a）検索結果数

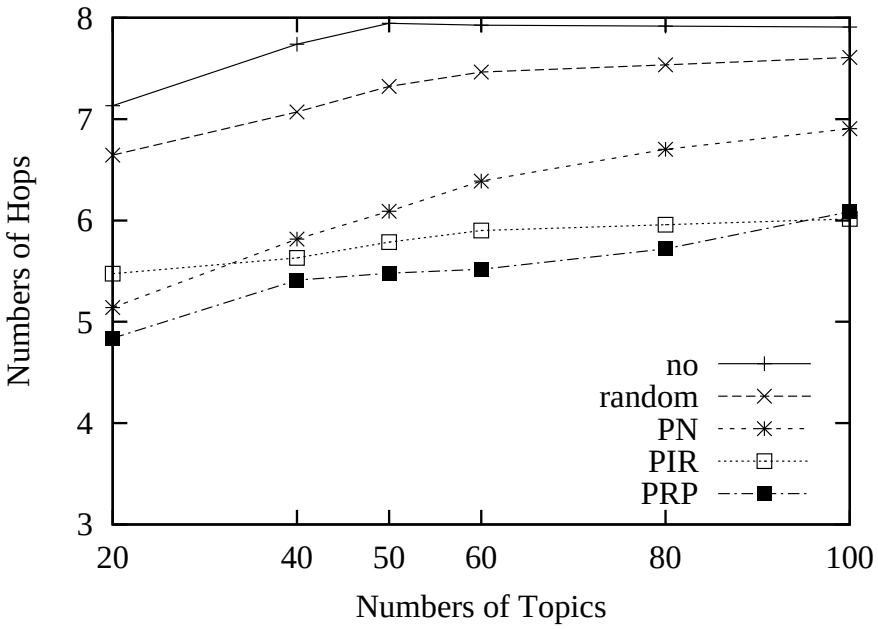


（b）応答時間

図 6.6　1 ピアあたりの平均複製数の変動による問い合わせ処理への影響



（a）検索結果数



（b）応答時間

図 6.7 *num_topic* の変動による問い合わせ処理への影響

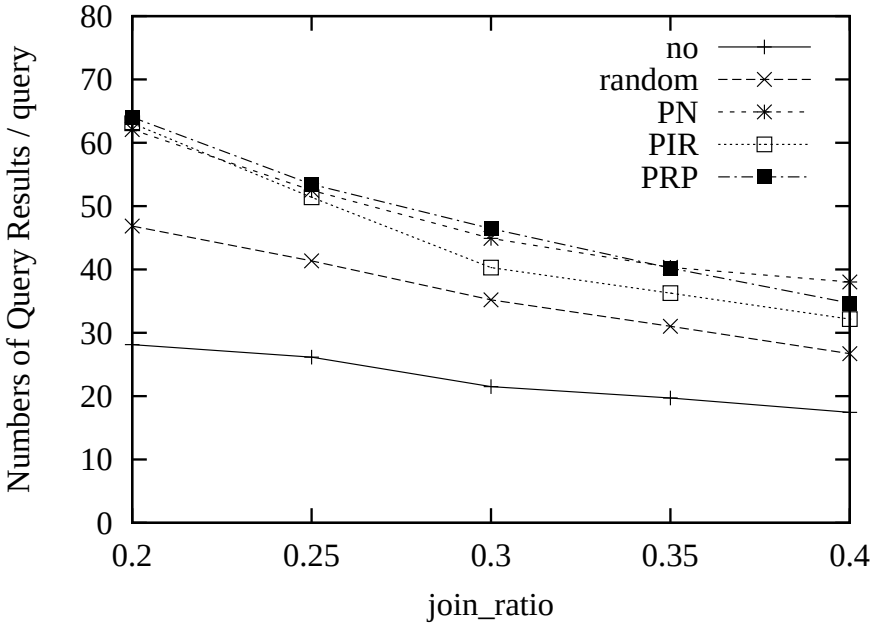
る確率が PN 方式よりも高いと予想できる．このことにより，PN 方式を用いた場合に比べ，PIR 方式及び PRP 方式を用いた場合ではトピック増加に伴う影響を低減することが可能になる．またトピック数が多い場合では，PIR 方式と PRP 方式でのパフォーマンスの差はあまり見られなくなった．この状況下では，両提案方式が実際には同様のピアを選択し配置していると考えられる．

ピアの接続・離脱の頻度の影響

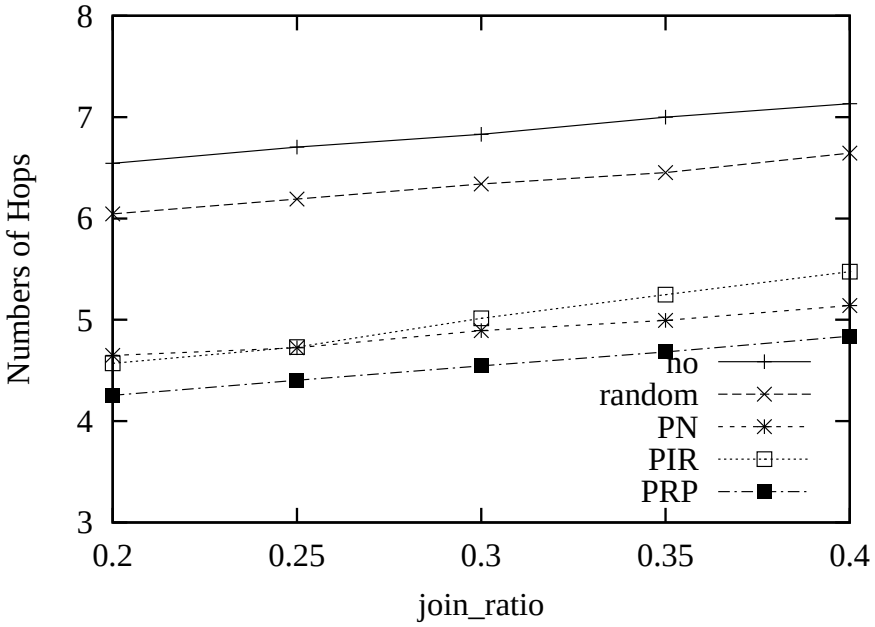
図 6.8 にピアによる接続，切断の実行頻度による問い合わせ処理への影響を示す．この実験では，$join\_ratio$ を変動させた場合，この頻度と同様に $leave\_ratio$ を変動させている．$join\_ratio$ の頻度が低いほど安定したネットワークになり，この頻度が高いほど不安定なネットワークになる．$join\_ratio$ の頻度が低い場合，提案した 3 つの配置方式ともランダム方式よりも良いパフォーマンスを示した．PIR 方式及び PRP 方式は $join\_ratio$ の頻度に関わらず，同様な傾向を示し，わずかではあるが，PIR 方式に比べ PRP 方式は不安定なネットワークでのパフォーマンスが優れている．PN 方式を用いた場合，安定したネットワーク環境下では他の提案方式とあまり変わらない結果を示したが，高頻度の $join\_ratio$ に対して他の提案方式よりもパフォーマンスの低下を抑えることが可能であった．PN 方式を用いた場合，あるピア p の隣接ピア p_i が切断したとしても，p_i の隣接ピアには p_i が保有している文書の複製を保有している可能性が高く，$join\_ratio$ の影響を軽減することが可能である．

問い合わせの頻度の影響

$query\_ratio$ の変動が問い合わせ処理へ与える影響を図 6.9 に示す．$query_ratio$ が低い場合，文書更新を行う確率が高くなる．文書更新を行う場合，本研究での文書更新では更新対象となる文書の複製の再配置を行う．そのとき，まず，更新対象の複製を非同期にし，その後に複製の再配置を行う．文書更新の頻度が高い場合，文書の再配置の処理を行うための遅延が生じるため，非同期である文書が増加し，システムへの悪影響を与える．しかしながら，文書更新時の文書再配置の遅延の発生にもかかわらず，$query\_ratio = 0.5$ のとき，No-Replication を用いた場合以外の方式では多くの検索結果を得ることができた．本研究では，各ピアは一旦複製を他のピアへ配置した後に，文書が更新されるまで，文書の再配置を行わない．そのため，ピアの切断に伴いシステム全体の文書数は減少してしまう．一度切断したピアが再び接続を行うとき，以前の隣接ピアと再び隣接するとは限

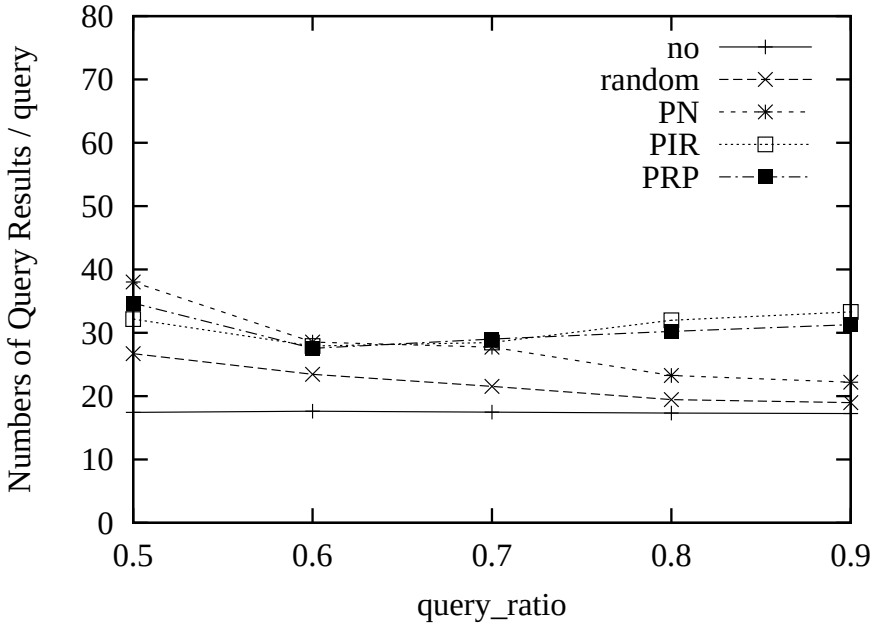


(a) 検索結果数

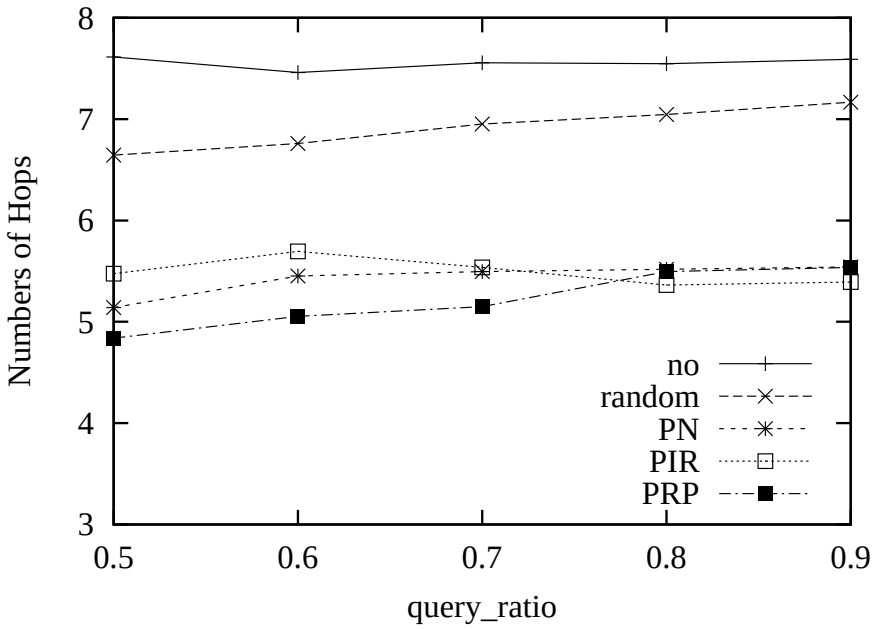


(b) 応答時間

図 6.8 *join_ratio* の変動による問い合わせ処理への影響



（a）検索結果数



（b）応答時間

図 6.9　*query_ratio* の変動による問い合わせ処理への影響

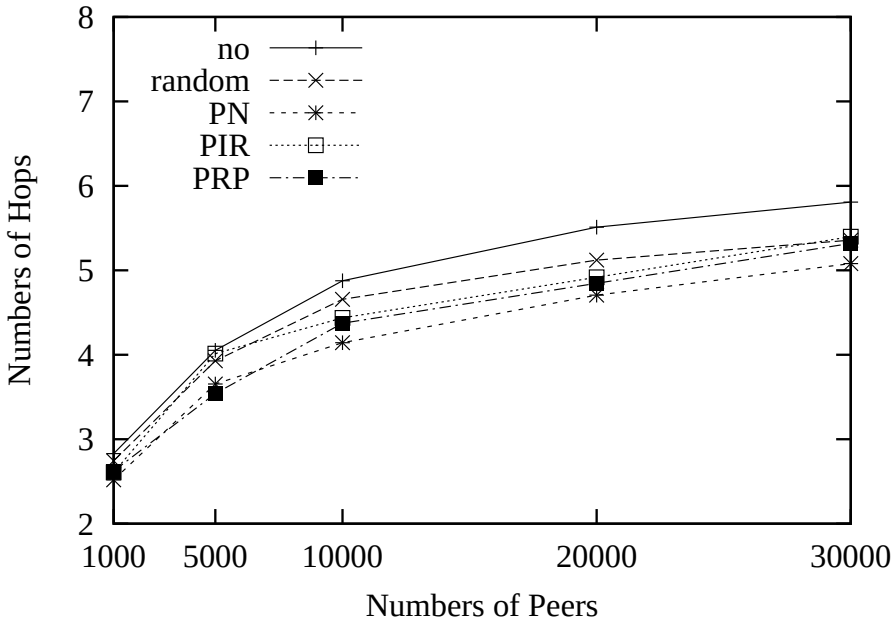
らない．そのため PN 方式では，文書更新の頻度が低い場合では，ピア p から近くに位置するピアは p が嗜好する文書を保有する確率は低下する．また，文書更新頻度が低くなるほど，PN 方式での文書分布はランダム方式の文書分布に近づくと考えられる．そのため，文書更新が程良く発生することにより，問い合わせ処理のパフォーマンスは向上する．PIR 方式では文書更新の頻度が低い状況下でも，他の複製配置方式と比較して，その影響を受けにくい結果を得た．PIR 方式では，各ピアから離れた距離に位置するピアにも複製を配置するため，文書分布は文書更新の頻度にあまり影響されないと予想できる．それは，PIR 方式を用いた文書複製配置は文書の更新頻度の変動によってその配置対象となるピアを変化させることが少ないためである．これは PRP 方式でも同様の傾向になると考えられる．

6.2.3 分散インデックスと複製配置を組み合わせた効果

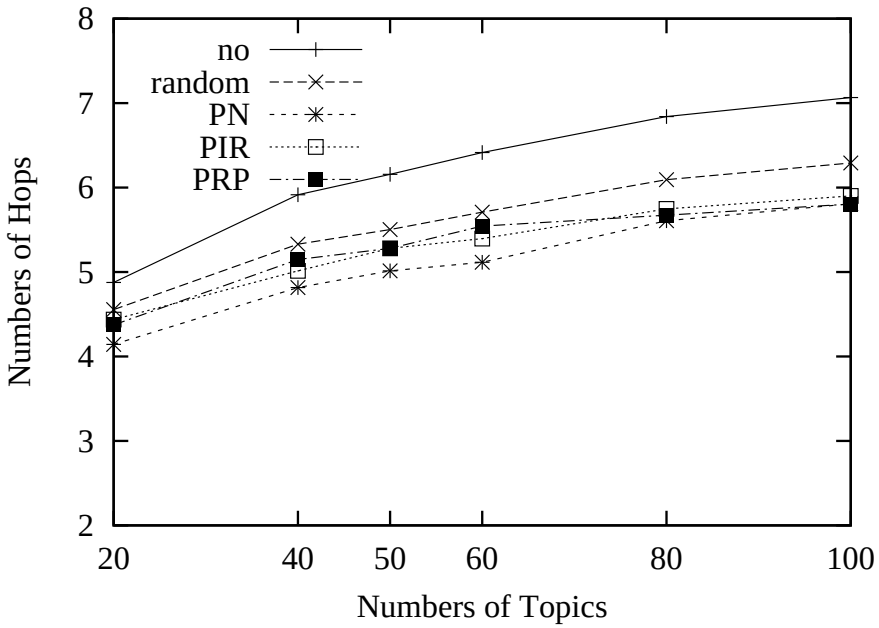
分散インデックスと複製配置の組み合わせはそれぞれを単独で用いるよりも問い合わせ処理への効果は優れることは予想の範囲である．しかしながら，それぞれの分散インデックスにとってどの複製配置を組み合わせればより高い効果が得られるかは調べる必要がある．そこでここでは各分散インデックスと各複製配置方式を組み合わせたときの応答時間，問い合わせメッセージ数，問い合わせの成功率およびインデックス更新メッセージ数へ与える影響を示すことにする．

応答時間への効果

NDI，TDI および RI に対して各種複製配置方式を組み合わせた場合でのシステムでのピア数またはトピック数を変動させたときの応答時間の状況をそれぞれ図 6.10，図 6.11 および図 6.12 に示す．ピア数もしくはトピック数の変動に対して NDI および TDI では各複製配置方式を組み合わせることでおよそ 1 ホップ程度改善する結果を得た．これに対し，RI に各複製配置方式を組み合わせることで多大な改善を行うことが分かった．RI に各複製配置方式を組み合わせる場合，ピア数が 30,000 である場合では応答時間をおよそ 1/3 程度に短縮することができた．またトピック数が 100 の場合では，その応答時間を半減させることが可能であった．これは RI はシステム全体での文書数が多くなければ効率的に問い合わせ処理が行えないことに他ならない．RI に対し，NDI および TDI を用いた場合ではシステム全体の文書数が少ない状況下でも効率的に問い合わせ処理を行うこと

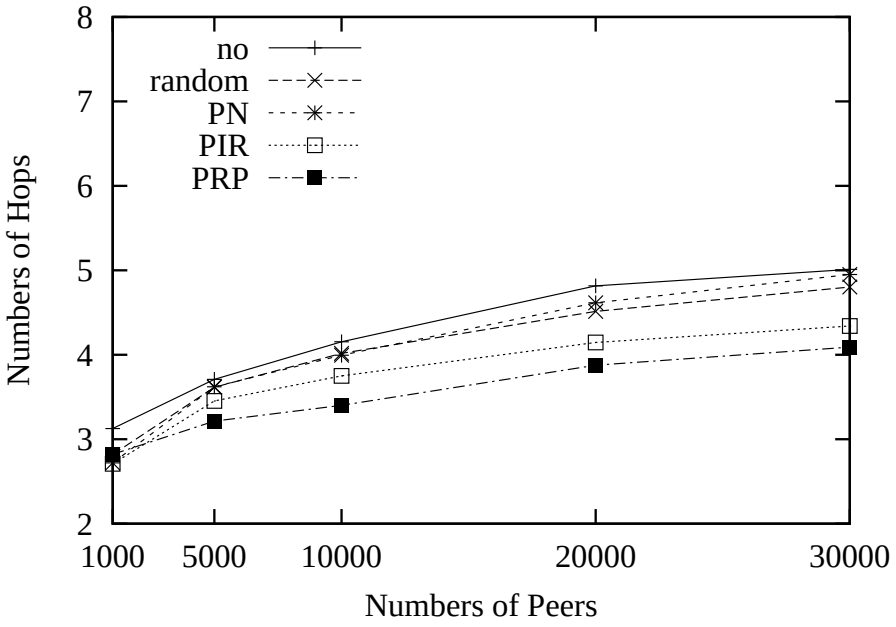


(a) ピア数変動の影響

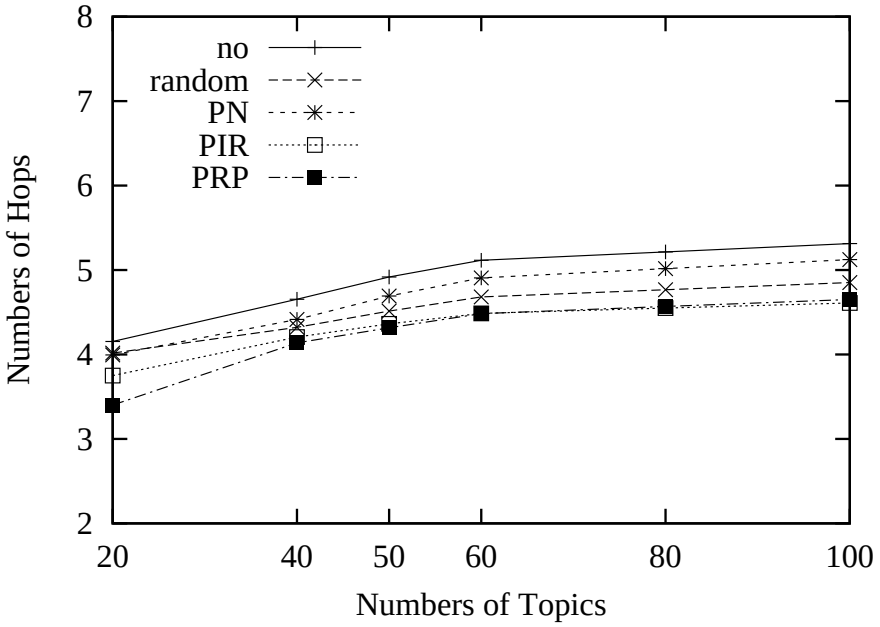


(b) トピック数変動の影響

図 6.10　NDI+ 各複製配置の応答時間

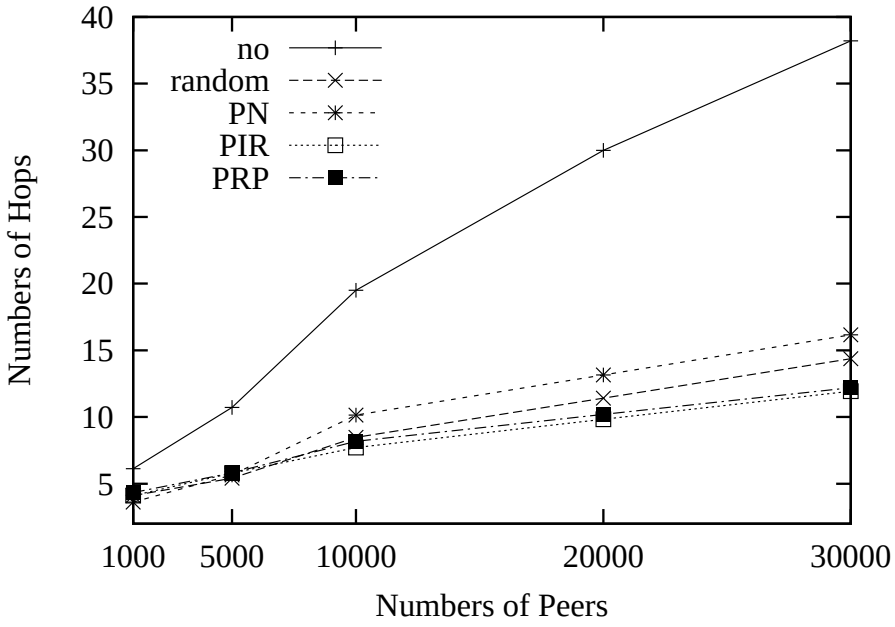


(a) ピア数変動の影響

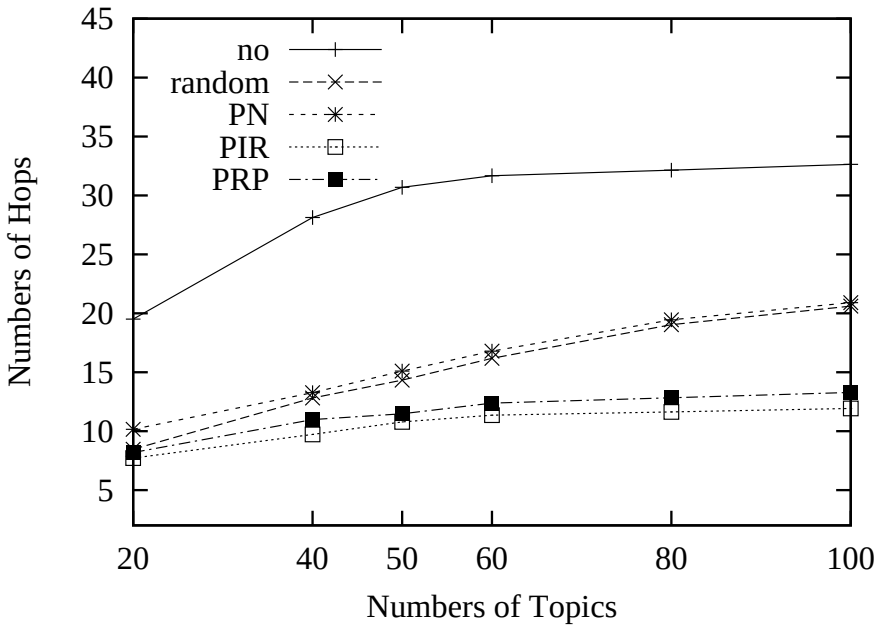


(b) トピック数変動の影響

図 6.11 TDI+ 各複製配置の応答時間



(a) ピア数変動の影響



(b) トピック数変動の影響

図 6.12　RI+ 各複製配置の応答時間

が可能であることを示した．RI は “有用” であるピアが問い合わせを行うピアの近くに多く位置することが効率的な問い合わせ処理を行うことの重要な要素の 1 つであることを示すと考えられる．NDI および TDI は “有用” であるピアが問い合わせを行うピアから離れた位置にある場合でも効率的に問い合わせ処理を行うことが可能であると考えられる．

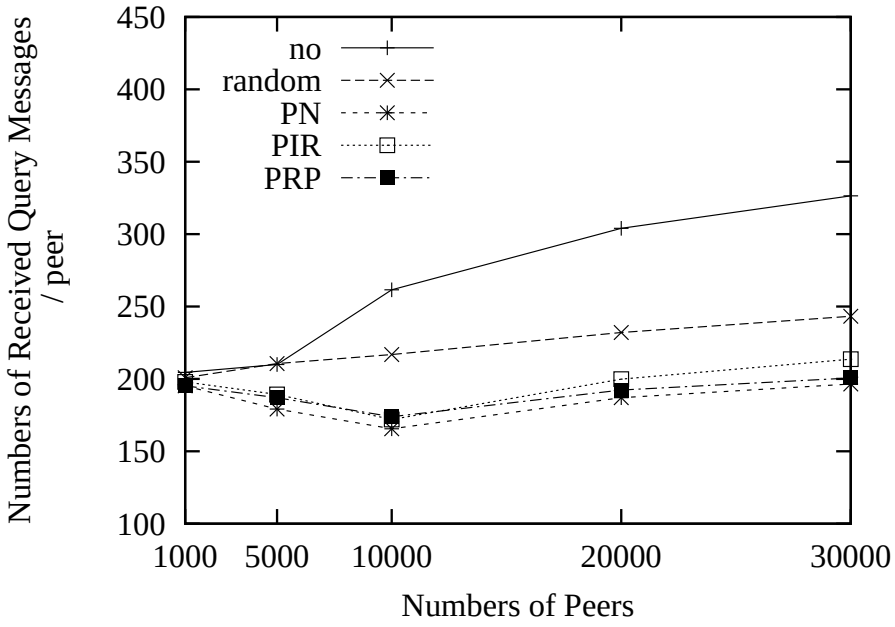
NDI では特に PN 方式と組み合わせた場合がもっともよい応答時間を示した．PRP 方式および PIR 方式と組み合わせた場合はランダム方式とあまり変わらないパフォーマンスを示した．これは NDI を用いた場合での切断するピアの決定に依存すると考えられる．ピア間の接続および切断の繰り返しによって各ピアはその隣接ピアの対象として最も検索結果を返したピアを選ぼうとし，“有用” である隣接ピアであっても検索結果が芳しくないピアに対して切断しようとする．NDI を用いた場合の自己組織化によって，各ピアは隣接ピアの対象を類似度が高いピアから選ぼうとする．そのため単に隣接ピアへ複製を配置する PN 方式がもっとも効果的であったと考えられる．

TDI では PRP 方式がもっともよい応答時間を示した．これは各ピアはその TDI 内にトピックの “有用度” を格納するためである．PRP 方式では最もトピックに関する類似度が高いピアへ複製を配置しようとする．TDI では問い合わせの転送先をトピックの “有用度” がもっとも高いピアから選ぶ．そのため PRP 方式と TDI の親和性が高くなり，この組み合わせは応答時間の改善に対して最も効果的であったと考えらえる．また，トピック数が多くなると PIR 方式と PRP 方式は実際は同様のピアへ複製を配置しようとしていた．そのためトピック数が 100 の場合では TDI と PIR 方式との組み合わせは PRP との組み合わせとほぼ同様のパフォーマンスを示したと考えられる．

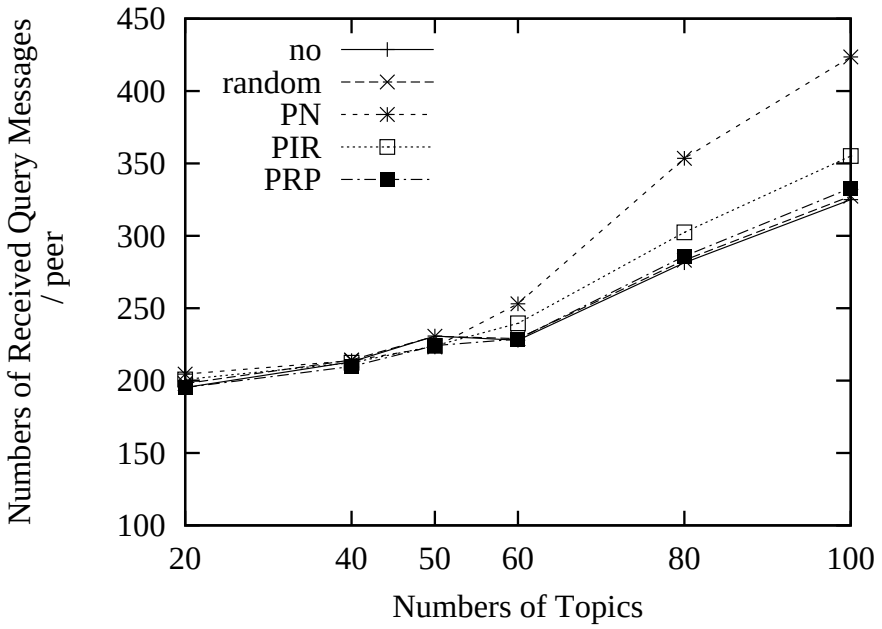
問い合わせメッセージ数への効果

ピア数およびトピック数を変動させた場合での各分散インデックスと各複製配置方式の組み合わせが問い合わせメッセージ数へ与える影響を図 6.13，図 6.14 および図 6.15 に示す．RI に複製配置を組み合わせたとしてもあまり効果がないように思える．しかしながら，RI ではわずかな問い合わせメッセージ数で一定以上の検索結果を得ることが可能である．ピア数が 30.000 である状況下では複製を配置しない場合に比べ PIR 方式を用いた場合ではおよそ 50 メッセージの違いがある．このメッセージ数の低減が RI と複製配置を組み合わせたときの応答時間の短縮を達成したと考えられる．

NDI に複製配置方式を組み合わせると，ピア数を変動させた場合では問い合わせメッセージ数を 2/3 程度低減させることが分かった．また，ピア数が 10,000 である場合，

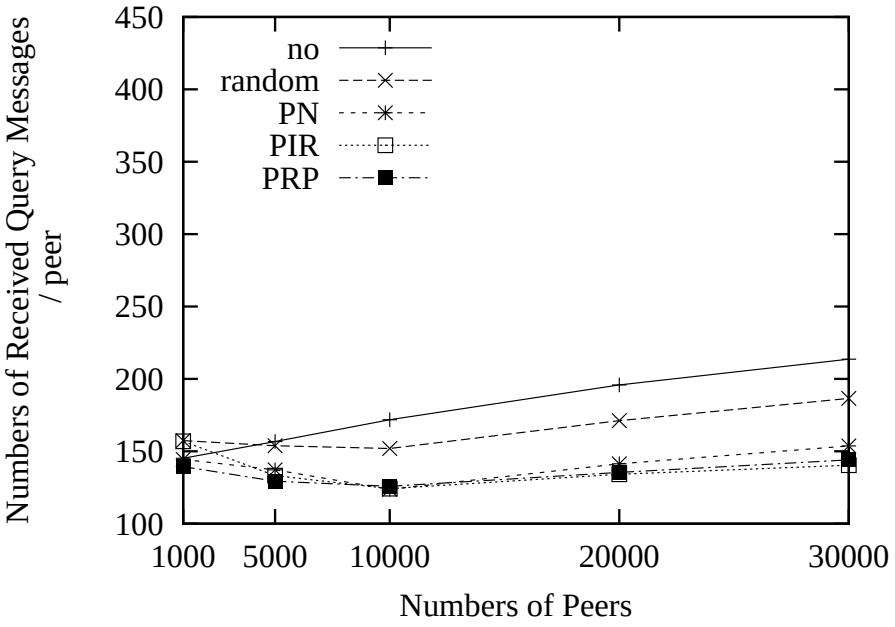


（a）ピア数変動の影響

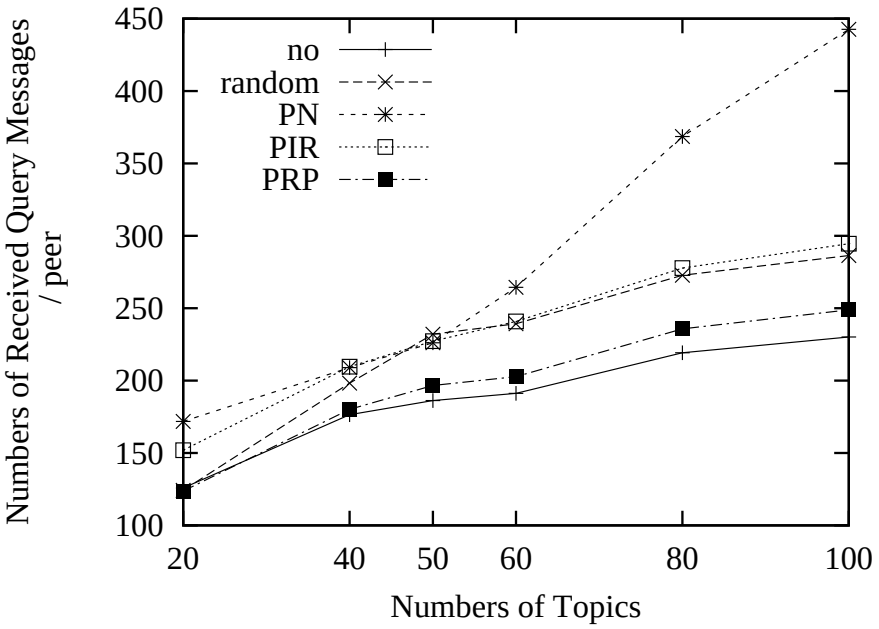


（b）トピック数変動の影響

図 6.13　NDI+ 各複製配置の問い合わせメッセージ数

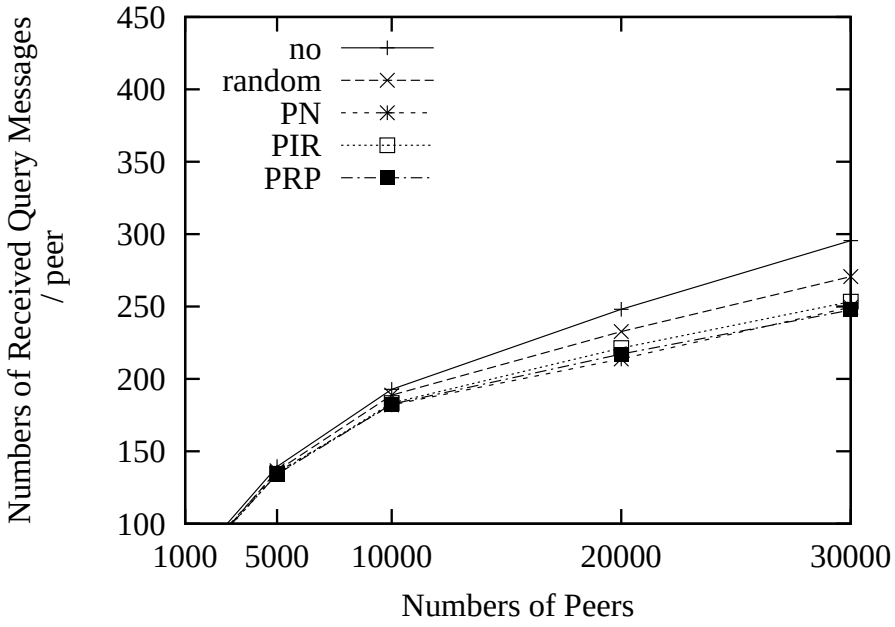


(a) ピア数変動の影響

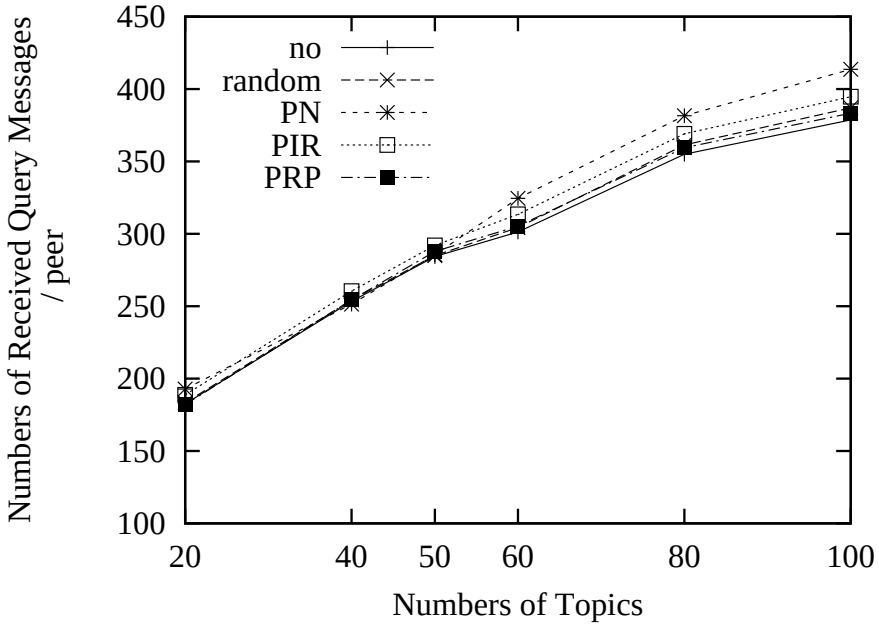


(b) トピック数変動の影響

図 6.14 TDI+ 各複製配置の問い合わせメッセージ数



（a）ピア数変動の影響

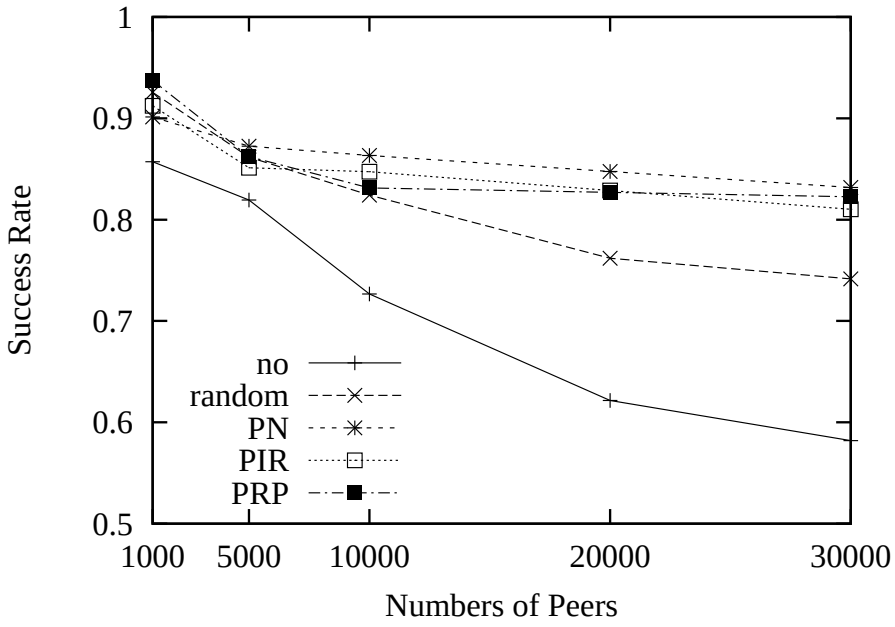


（b）トピック数変動の影響

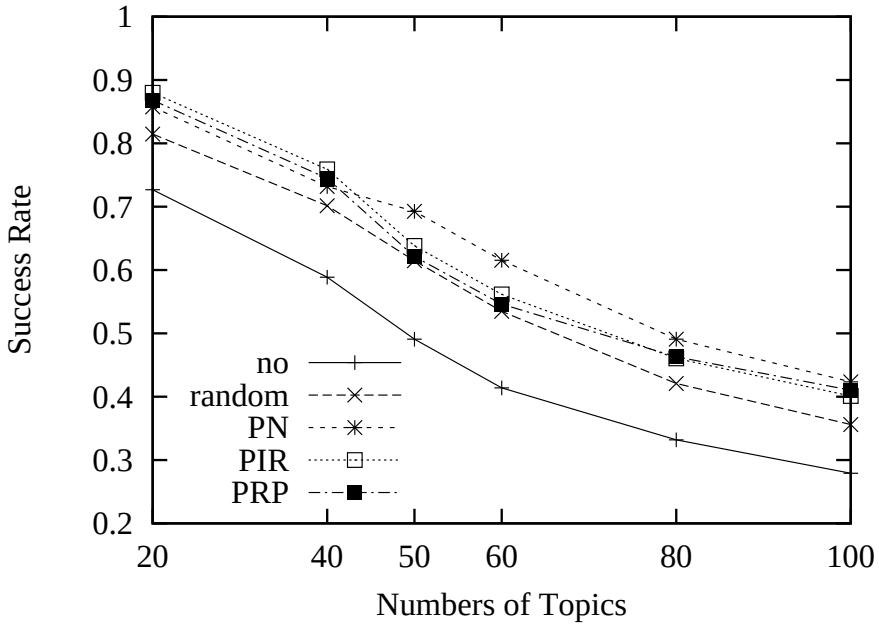
図 6.15　RI+ 各複製配置の問い合わせメッセージ数

NDI と PN 方式，PIR 方式および PRP 方式を組み合わせると問い合わせメッセージ数が 10,000 よりも少ない場合に比べ少なかった．これは，問い合わせメッセージを受信しないピアが他の状況下に比べ多く存在することを示すと考えられる．トピック数の変動に対して効果的ではなかった NDI であるが複製を配置することで非常に多くの問い合わせメッセージ数を低減することが可能であった．この理由として NDI での自己組織化によって隣接ピアで多くの検索結果を得られる確率が高くなったためであると考えられる．

TDI と各複製配置方式を組み合わせたとき，ピア数が変動する状況下では問い合わせメッセージはほぼ一定である結果を得，ピア数の変動にあまり影響されないパフォーマンスを TDI は示した．特に PRP 方式との組み合わせは，他の複製配置方式との組み合わせよりもトピック数の変動の影響を緩和することが分かった．TDI+PRP は各ピアが嗜好しているトピックの “有用度” をより高くする．そのため，トピック数が増加した状況下でも効率的に問い合わせメッセージの転送先を修正することが可能であったと考えられる．



(a) ピア数変動の影響

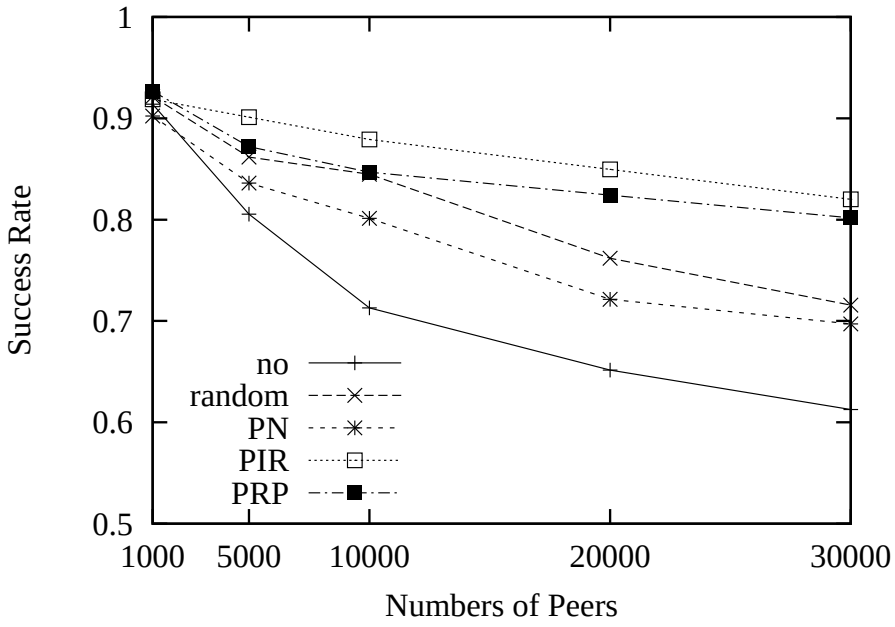


(b) トピック数変動の影響

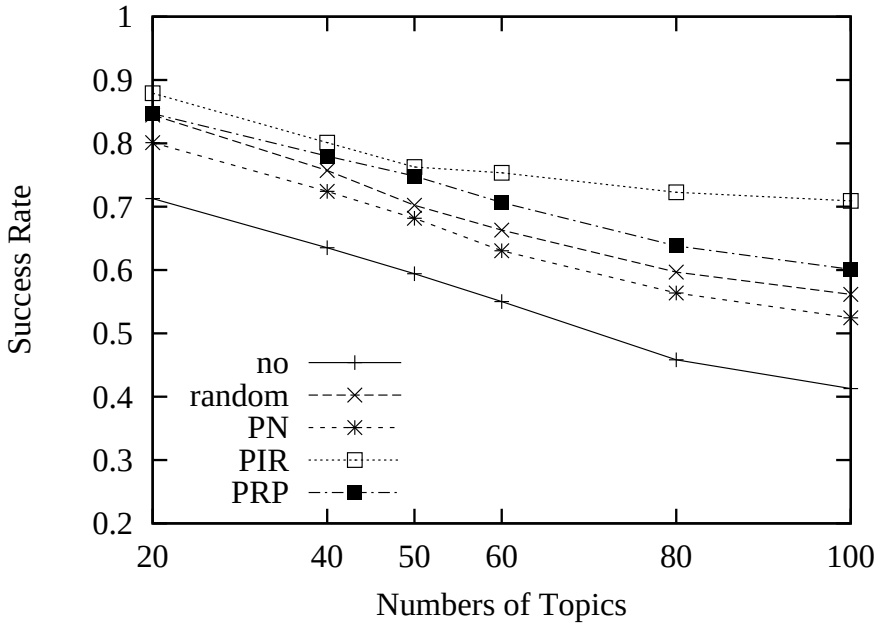
図 6.16　NDI+ 各複製配置の問い合わせ成功率

問い合わせの成功率への効果

図 6.16，図 6.17 および図 6.18 は問い合わせの成功率を示す．複製を配置することでシステム内の文書数が増加する．そのため問い合わせの成功率は著しく向上する．ピア数が増加した状況下で，PIR 方式および PRP 方式は TDI または RI を用いた場合にてその効果が他の複製配置方式よりも優れていた．NDI でも PIR 方式および PRP 方式は優れたパフォーマンスを示したが，PN 方式との組み合わせはより効果的であった．これも NDI での自己組織化がその要因であろうと思われる．トピック数の増減に対して NDI+PN の組み合わせはある程度問い合わせの成功率が改善された．TDI+PIR または TDI+PRP は問い合わせの成功率を著しく改善した．この理由としてはそれらの組み合わせの親和性であろうと思われる．

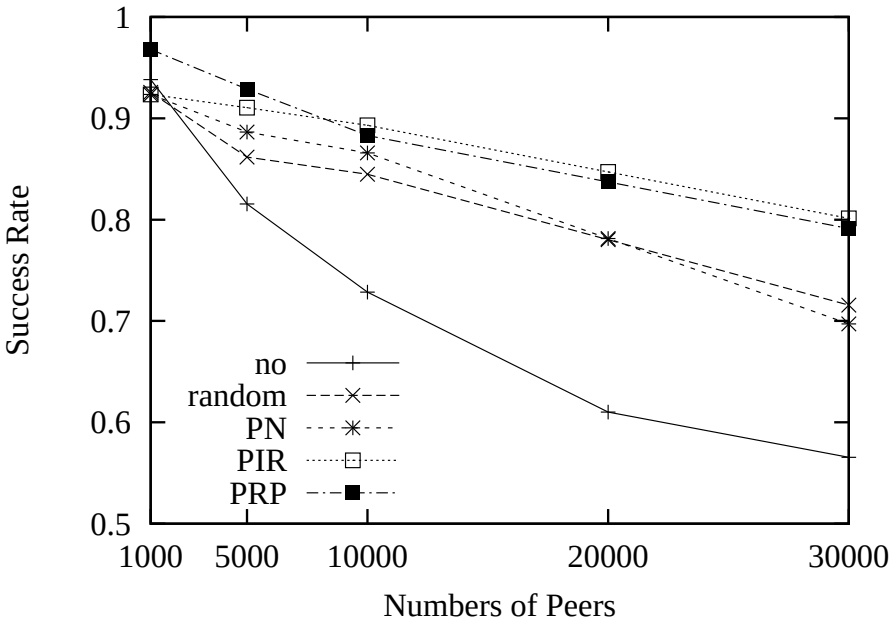


（a）ピア数変動の影響

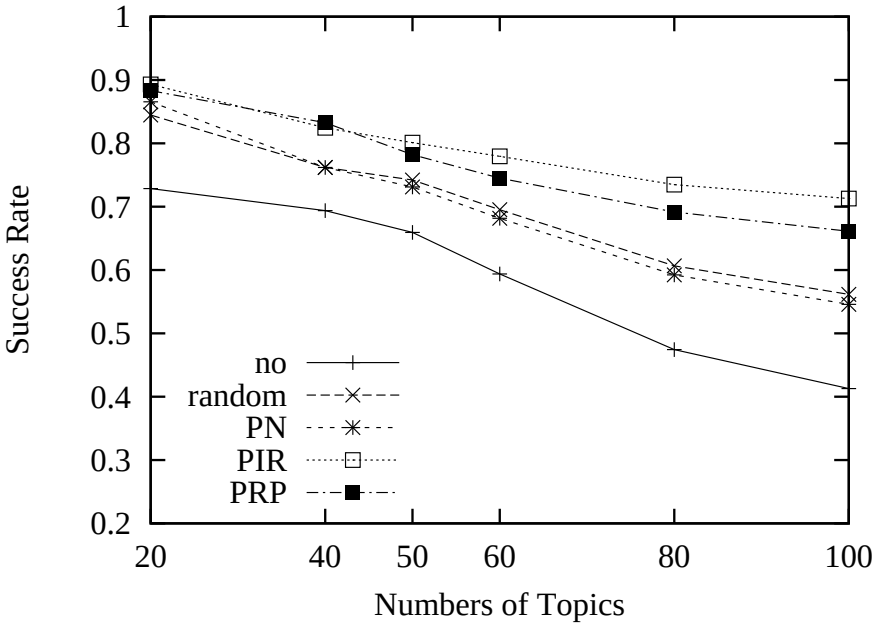


（b）トピック数変動の影響

図 6.17　TDI+ 各複製配置の問い合わせ成功率

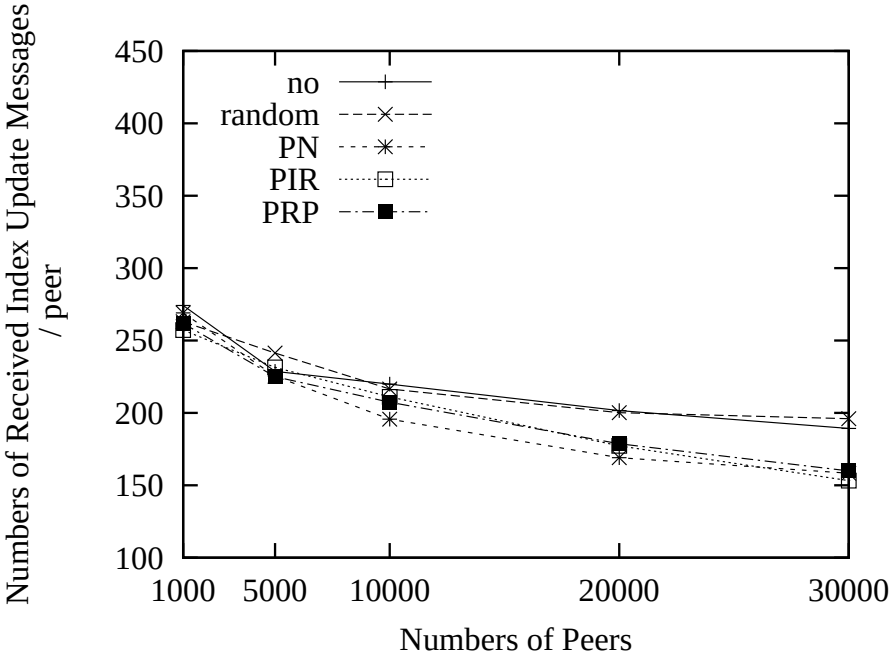


(a) ピア数変動の影響

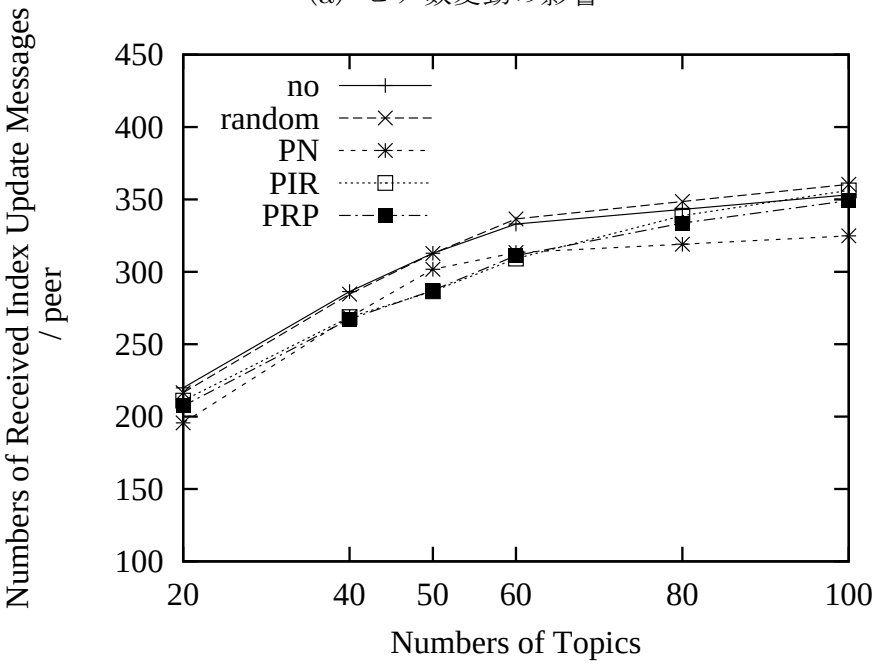


(b) トピック数変動の影響

図 6.18 RI+ 各複製配置の問い合わせ成功率

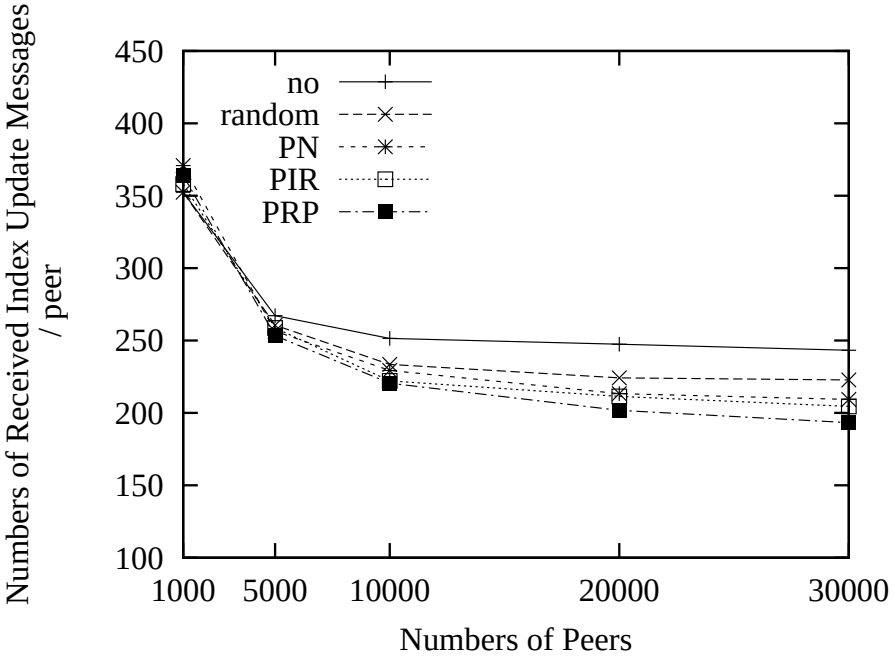


（a）ピア数変動の影響

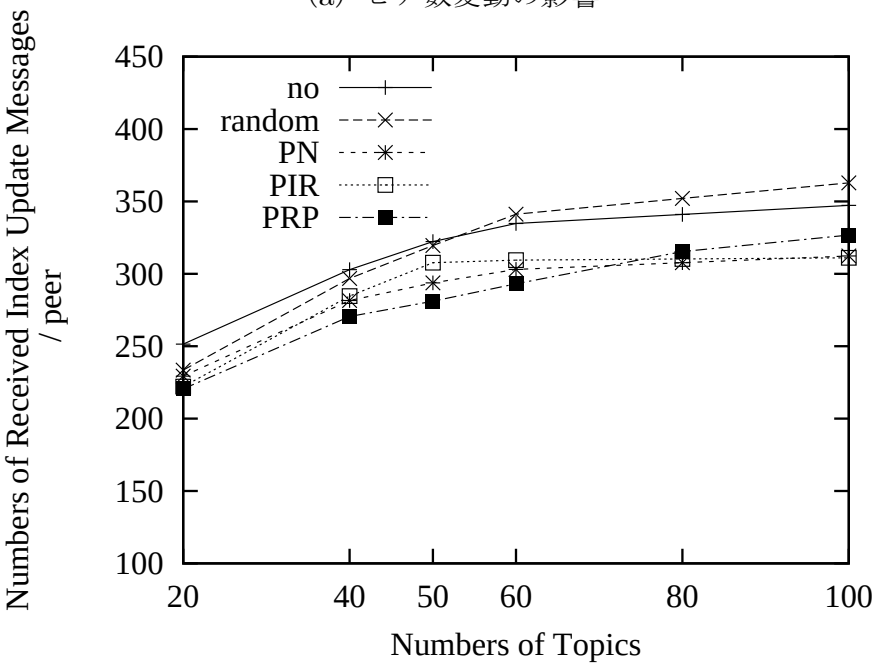


（b）トピック数変動の影響

図 6.19　NDI+ 各複製配置のインデックス更新メッセージ数



(a) ピア数変動の影響



(b) トピック数変動の影響

図 6.20 TDI+ 各複製配置のインデックス更新メッセージ数

インデックス更新メッセージ数への効果

図 6.19，図 6.20 および図 6.21 はインデックス更新のメッセージ数を示す．複製配置は各ピアのローカルデータベースを更新させるため，インデックス更新に敏感である RI ではそのインデックス更新を誘発する．この傾向はピア数およびトピック数の変動に影響され，複製配置方式との組み合わせは影響しない．トピック数が変動する状況下では NDI および TDI も RI と同様の傾向を示した．しかしながら，ピア数が増加した状況下での NDI および TDI では PN 方式，PIR 方式または PRP 方式との組み合わせの場合，1 ピアあたりのインデックス更新メッセージ数は減少する．それは “有用” であるピアのインデックス更新のみ伝播される．そのため，各ピアの DI 内での推奨するピアの順位に変動がないならばインデックス更新メッセージを送信しないためである．これらの複製配置方式はシステム内の各ピアと類似度が高いピアの “有用度” を高めるような複製配置を行うため，DI 内に格納しているピアの “有用度” は実際には変動しているが，推奨するピアの順位の変動をより少なくすると考えられる．

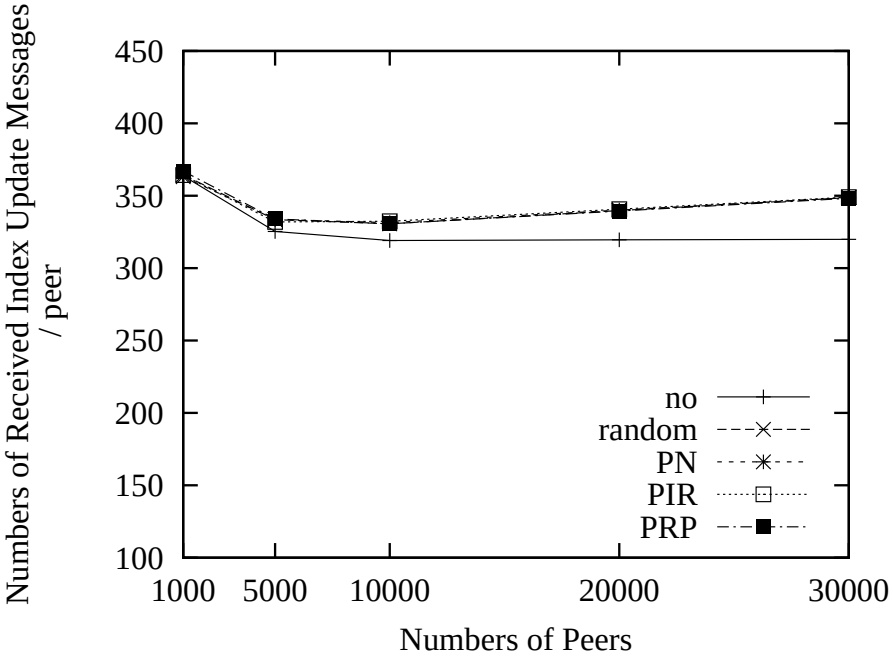
6.3 問い合わせの特性と処理の効率性に関する実験

前節の実験では，各ピアはそのピアが嗜好するトピックに対して 0.6 の確率で問い合わせを行うように設定していた．さらに，各トピックに属するキーワードの数はどのトピックでも同数であり，あるキーワードは特定のトピックに属するように設定してあった．他の実験設定のとき，システムはどのような挙動を示すのであろうか．そこで，本節では各ピアが実行する問い合わせ処理の内容をそのピアが嗜好するトピックである確率を変動させた場合のシステムの挙動を探ることにする．

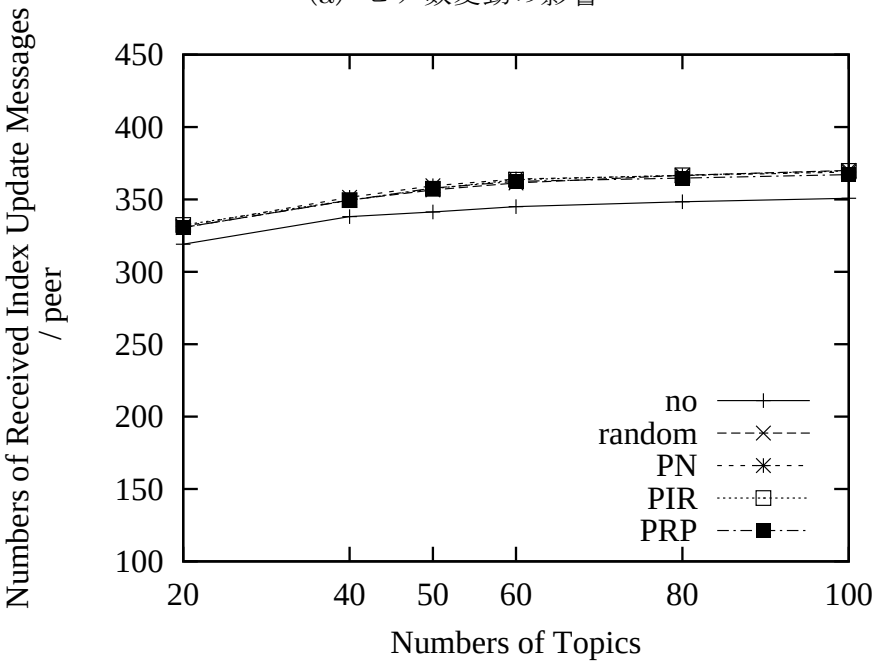
6.3.1 実験概要

本節と前節との実験設定の差異は以下のとおりである．

- キーワードの分布：各キーワードがどのくらいのトピックに属するかを設定するためにトピックに対するキーワードの分布の確率分布を与えた．その確率分布は正規分布 $N(\mu, \sigma^2)$ に従うように設定した．ここで，μ はキーワードが属するトピック数の母平均，σ^2 はその母分散を示す．本節の実験では $\mu = num\_topic \cdot 0.1$，$\sigma = num\_topic \cdot 0.08$ とした．また，各トピックに属する文書数はそのトピックに



(a) ピア数変動の影響



(b) トピック数変動の影響

図 6.21 RI+ 各複製配置のインデックス更新メッセージ数

属するキーワード数に比例させた．あるトピック t_i の文書数 $NumDoc(t_i)$ は次のように決定した．

$$NumDoc(t_i) = NumDoc \cdot \frac{|K(t_i)|}{\sum_i |K(t_i)|} \tag{6.1}$$

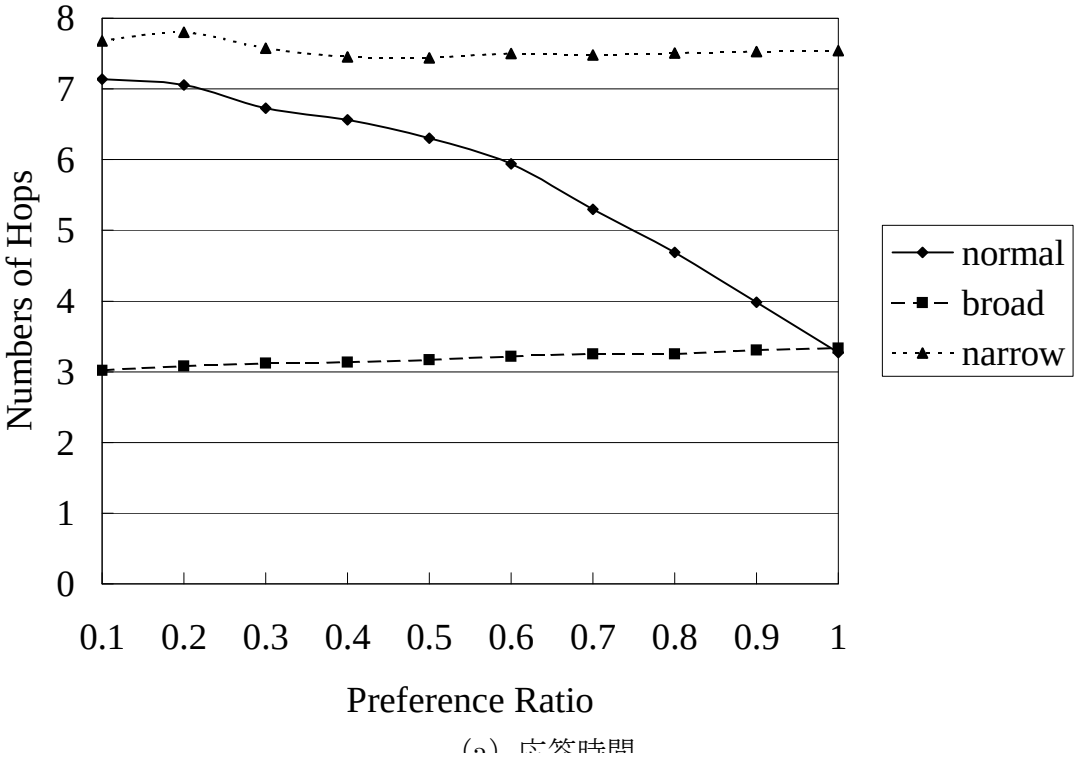
ここで，$K(t_i)$ はトピック t_i に属するキーワードの集合を示す．

- ピアの振る舞い : 各ピアがそのピアが嗜好するトピックに対する問い合わせ処理を行う確率（*perference ratio*）を変動させた．その範囲は [0.1 . . . 1.0] とした．さらに，各確率でピアが振る舞うときに，システム上の各ピアが，もっとも多くのトピックに属するキーワードで問い合わせを行った場合（broad），および 1 つのトピックにしか属していないキーワードで問い合わせを行った場合（narrow）での問い合わせ処理の状況を示した．

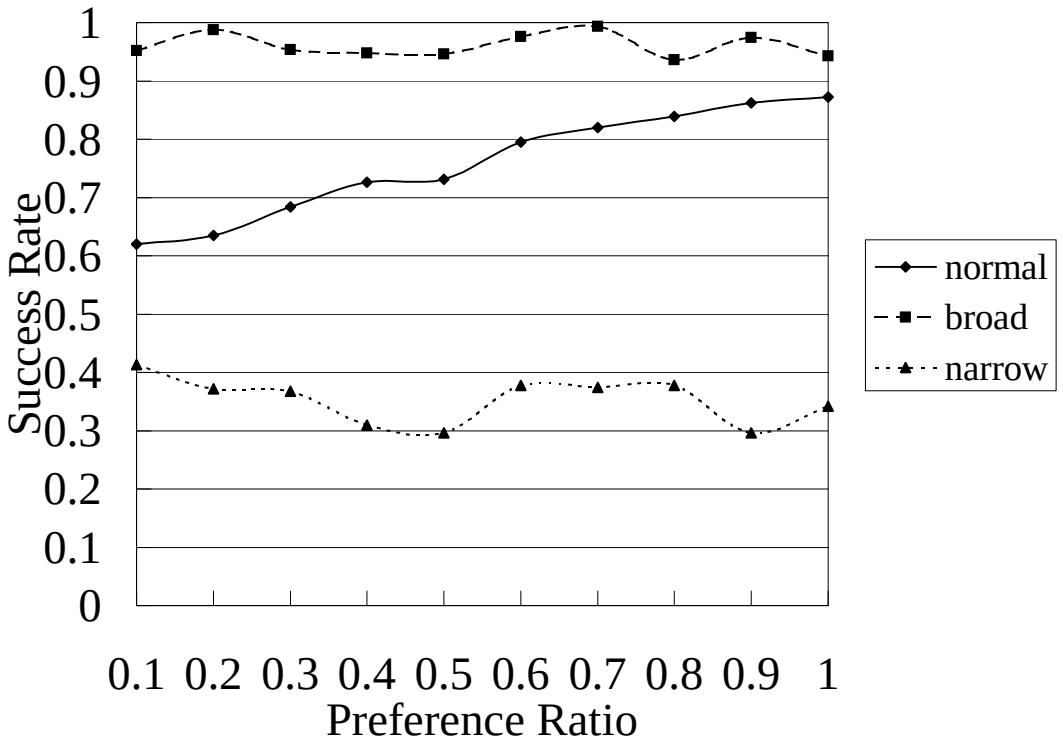
6.3.2 実験結果

図 6.22 は Gnutella 検索を用いたときの問い合わせ処理の応答時間およびその成功率を示す．各ピアの嗜好とそのピアの問い合わせ処理の状況が一致しない，すなわちあるピアがそのピアが嗜好するトピックに対する問い合わせ処理をあまり行わない場合，Gnutella のような検索方法ではその問い合わせ処理の応答時間および成功率は低下する結果を得た．この結果は図 6.22 での “normal” で示した．各ピアはその隣接ピアを選択するときに，過去の検索結果の情報を用いる．そのため多くのピアはそのピアの振る舞いと嗜好が一致する問い合わせ処理を行う確率が低い場合，隣接ピアと嗜好が同じである確率は低下し，ランダムに隣接ピアを選択することになる．反対に，ピアの振る舞いと嗜好が一致する問い合わせ処理を行う確率が高い場合，各ピアはその隣接ピアを同じ嗜好であるピア集合の中から選択する確率が高くなる．この状況下では各ピアが行う問い合わせにマッチする文書は，ネットワーク上でのそのピアの近くに位置することになる．そのため問い合わせ処理の応答時間および成功率は高い．

多くのトピックに属するキーワードで問い合わせを行った場合では，ピアの振る舞いを変動させたとしても，ほぼ問い合わせ処理の応答時間およびその成功率に変化はあまりみられなかった．ただし，各ピアがそのピアが嗜好するトピックに対して問い合わせを行う確率が高くなるにつれ，わずかではあるが応答時間が悪化する．その理由は各ピアがその隣接ピアを選択するときに過去の検索結果の情報を用いるためである．各ピアがそのピア



(a) 応答時間

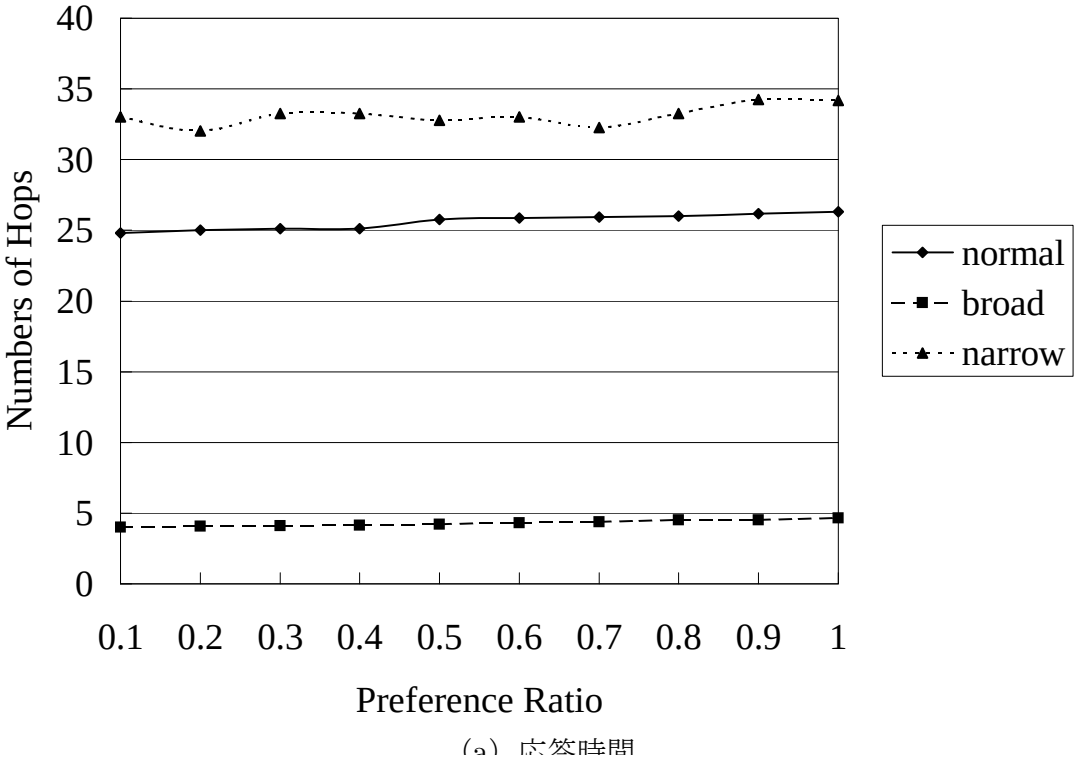


(b) 成功率

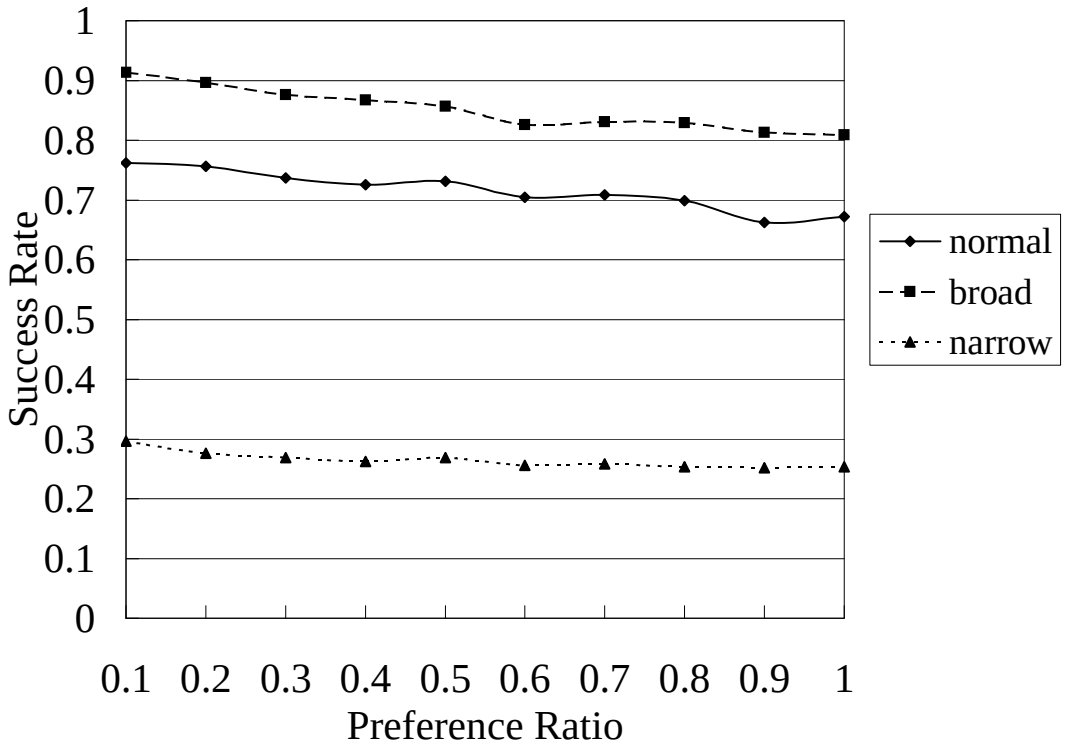
図 6.22　ピアの振る舞いを変動させた場合での Gnutella の問い合わせ処理効率

が嗜好するトピックのみに対する問い合わせ処理を行っている場合，各ピアは同じ嗜好のピアを隣接ピアとして選択する確率が高い．反対に，各ピアがそのピアが嗜好するトピックに対する問い合わせ処理をあまり行わない場合，隣接ピアとして同じ嗜好のピアを選択することをあまり行わなくなる．あるピアのネットワークの近い位置にそのピアと同じ嗜好のピアが多い場合について考えてみる．あるピアが多くのトピックに広範囲に分布しているキーワードで問い合わせを行う場合，そのピアが嗜好するトピックにそのキーワードが属しているのであれば，その問い合わせ処理の応答時間は早いが，嗜好するトピックにそのキーワードが属していない場合，そのピアから離れた位置に求める文書が配置されていることになる．広範囲に分布しているキーワードでの問い合わせ処理の成功確率は高いが，その文書はネットワーク上に偏って分布することになる．一方，各ピアがそのピアが嗜好しないトピックに対する問い合わせ処理である確率が高い場合について考えてみる．もっとも多くのトピックに属するキーワードを含む文書は，各ピアとそのピアが選択する隣接ピアの嗜好が同じである確率が低下するため，ネットワーク上に偏らずに分布することになる．そのため，システム上の任意のピアが多くのトピックに属するキーワードでの問い合わせ処理を行う場合，あまり偏って分布していない状況下の方がその問い合わせ処理の応答時間は向上する．しかしながら，あるピアからネットワークの一定範囲内にもっとも多くのトピックに属するキーワードを含む文書は発見しやすいため，そのキーワードに対する問い合わせ処理の成功率はあまり変化がなかった．

RI を用いた場合での各ピアの振る舞いがそのピアの嗜好と一致する確率を変動させた場合の結果を図 6.23 に示す．RI を用いた場合，各ピアが行う問い合わせ処理とそのピアの嗜好の一致する確率の変動はあまり問い合わせ処理の応答時間および成功率に影響を与えないことがわかった．この傾向はキーワードの分布に影響されないこともわかった．RI に格納される情報は各トピックの情報を幅広く格納するため，問い合わせ処理ごとにそれにマッチする文書が多く存在するであろうと思われる方向を RI は示すことができる．そのため RI を用いた場合，いかなるトピックに対する問い合わせ処理であってもその方向をシステム上の任意のピアの RI は示すことが可能になる．しかしながら，嗜好するトピックに対する問い合わせ処理を行う確率が高くなるにつれ，問い合わせの応答時間および成功率がわずかではあるが低下することがわかった．また，多くのトピックに属するキーワードでの問い合わせでも，ごく少数のトピックにしか属していないキーワードでの問い合わせでも同様の結果を得た．Gnutella の場合と同様に，ピアの振る舞いとピアの



(a) 応答時間



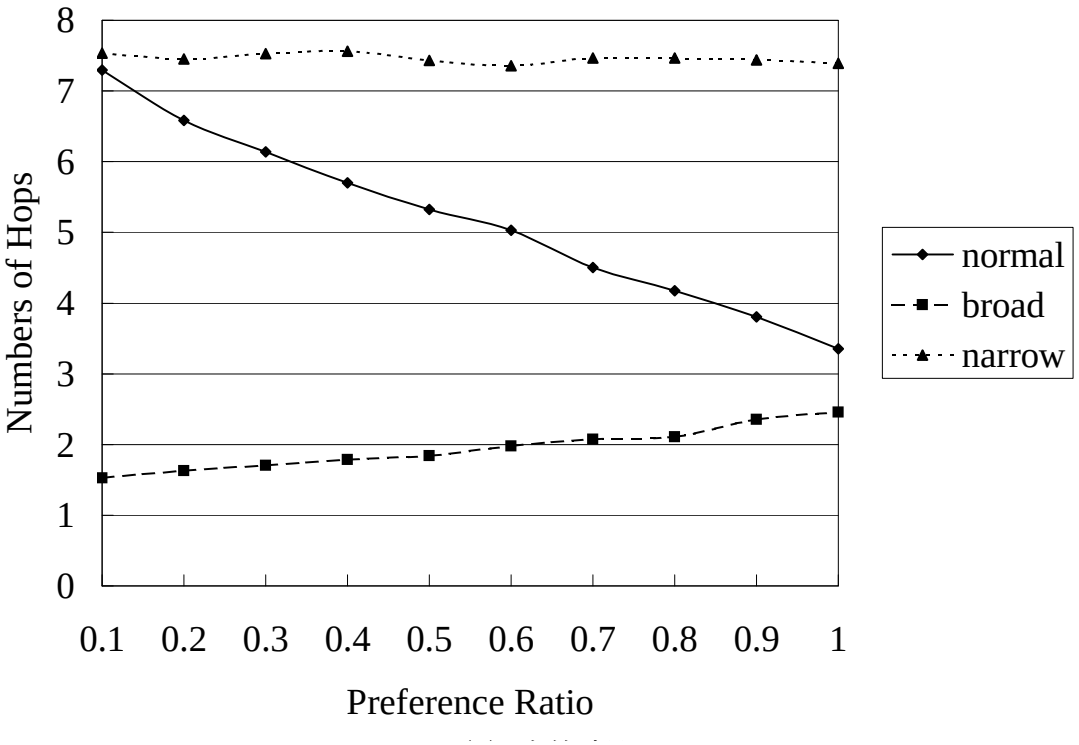
(b) 成功率

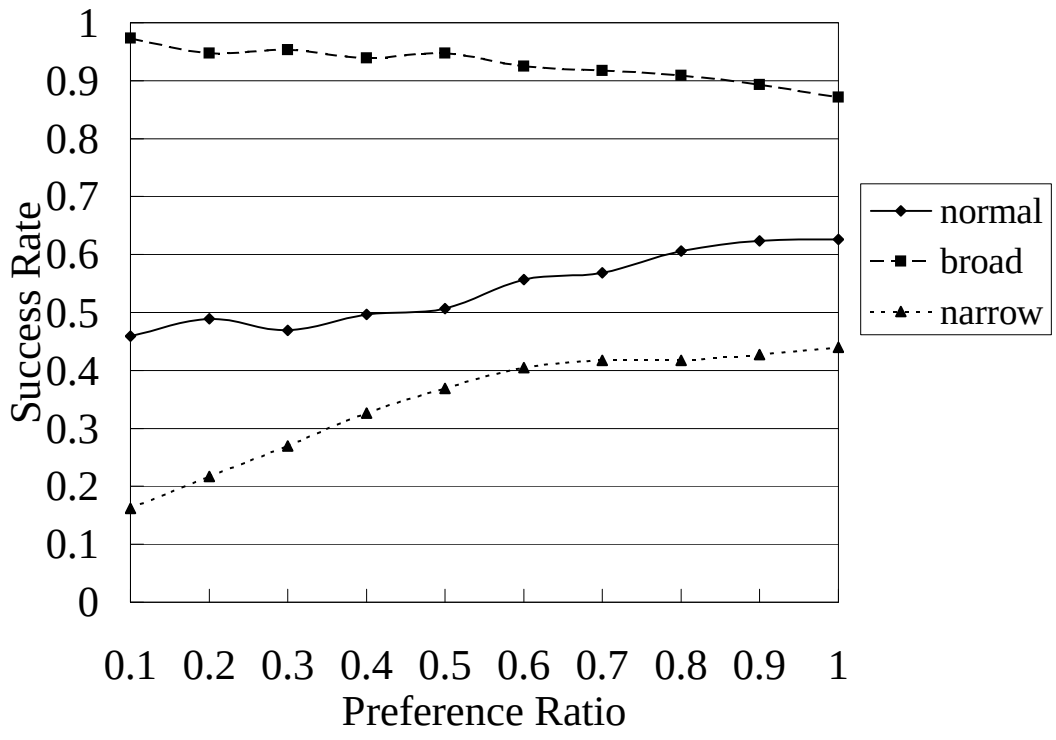
図 6.23　ピアの振る舞いを変動させた場合での RI の問い合わせ処理効率

嗜好が一致する場合，各ピアはそのピアと同じ嗜好であるピアを隣接ピアとして選択する．また，RI は深さ優先探索を行う．そのため RI で示された方向に確かに多くの文書は存在するが，何ホップ向こうにいくつの文書があるかという情報までは RI では示すことができない．そのため，非常に近い位置に多くの嗜好するトピックに属する文書は多くなるが，ネットワーク上での距離が多くなるにつれ，求める文書が位置している確率は低くなる．RI を用いた場合，探索場所を戻すためには，これ以上のピアが存在しないとき，またはその問い合わせにマッチする文書がその方向には確実に存在しないときである．高度にクラスタ化されたネットワークでは，浅い部分に多くの求める文書が位置しているが深い位置には求める文書があまり位置していない．この状況下で RI は効率的な問い合わせ処理を行うことが困難になることがわかった．

NDI を用いた場合でのピアの振る舞いとそのピアの嗜好が一致する確率を変動させた場合での問い合わせ処理の状況を図 6.24 に示す．NDI を用いた場合，ピアの振る舞いとその嗜好が一致しない場合，効率的な問い合わせ処理を行うことが非常に困難になることがわかった．特にピアが嗜好するトピックに対する問い合わせ処理を行う確率が 0.8 よりも低くくなると，問い合わせ処理の成功率は低下しはじめ，0.6 よりも低くなると，大幅に問い合わせ処理の成功率は低下することがわかった．これは，NDI 内には文書の情報が無いため，問い合わせの内容ごとに問い合わせメッセージの転送先を選択することが困難であるためである．しかしながら，ピアの振る舞いとその嗜好があまり一致しない状況下で，多くのトピックに属するキーワードによる問い合わせ処理を行った場合，その応答時間および成功率は非常に優れていることがわかった．NDI を用いた場合，各ピアが DI 内に格納しているピアの情報はピアの “有用度” が高いが，それらのピアの多くが多くのトピックに属しているキーワードを含む文書を所有している可能性が高い．そのため，多くのトピックに属する文書に対する問い合わせ処理を行う場合，NDI は効率的な問い合わせ処理を行うことができる．

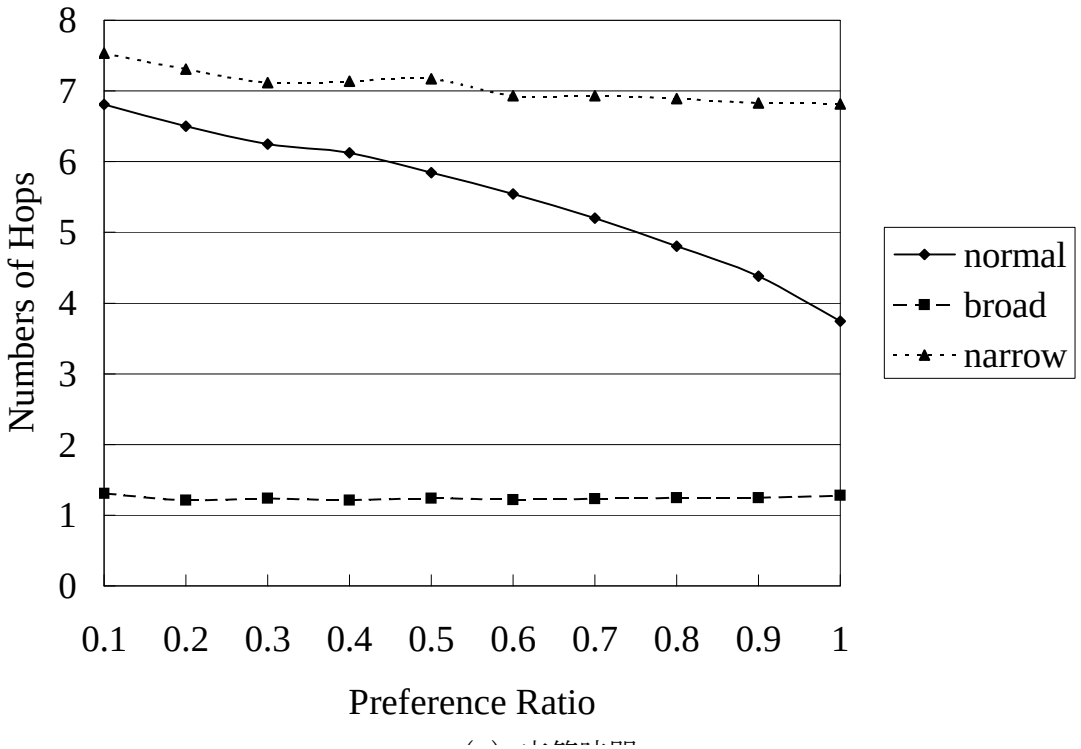
TDI を用いた場合でのピアの振る舞いとそのピアの嗜好が一致する確率を変動させた場合での問い合わせ処理の状況を図 6.25 に示す．TDI は NDI にトピック情報を格納するように拡張した DI であるため，TDI を用いた場合もその傾向はほぼ NDI を用いた場合と一致し，ピアの振る舞いとその嗜好が一致する確率が 0.7 以下になると，問い合わせ処理の成功率は低下しはじめ，0.3 以下になるとその成功率は大幅に低下してしまう．しかしながら，NDI を用いた場合と比べて，TDI を用いた場合での問い合わせの成功率は非

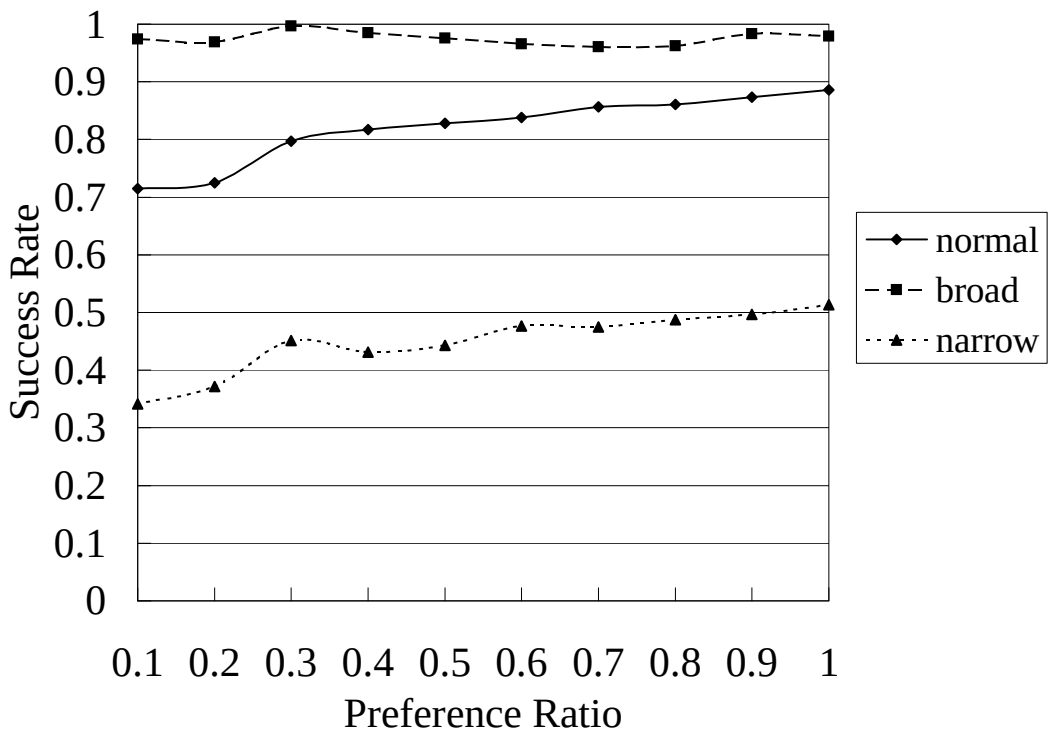




（b）成功率

図 6.24　ピアの振る舞いを変動させた場合での NDI の問い合わせ処理効率





(b) 成功率

図 6.25　ピアの振る舞いを変動させた場合での TDI の問い合わせ処理効率

常に高い結果を得た．あるピア A の DI が図 4.6 に示すような状態である場合に，A がトピック “OS” に対する問い合わせを行う場合，NDI では単にピアの “有用度” が A の DI 内でもっとも高いピア B に問い合わせメッセージを送信してしまう．しかしながら TDI を用いた場合，ピア “OS” の “有用度”A の DI 内でもっともが高い C に対して問い合わせメッセージを送信することが可能になる．一見，TDI と RI は同じ深さ優先を行うようにみえる．TDI は問い合わせメッセージを求める文書のトピックの “有用度” が問い合わせ処理を行うピアの DI 内でもっとも高いピアに直接送信することが可能である．RI は問い合わせに対する “方向” を示すが，TDI は問い合わせに対する “方向” ではなく “位置” を示す．そのため，TDI では RI でみられる探索位置を戻すという手間を省くことが可能になる．これが TDI と RI との差異である．

共有される文書が頻繁に更新される状況下では素早く検索結果を得る必要がある．この点で TDI は RI よりも有効である．RI は，ピアの嗜好とその振る舞いが一致する確率の変動をあまり影響されない．一方，TDI は NDI ほどピアの嗜好とその振る舞いが一致する確率の変動の影響を受けないが，RI よりもその影響を受けやすいことがわかったが，TDI を用いた場合，この確率が 0.3 よりも大きい状況下で効率的な問い合わせ処理を行うことが可能であることもわかった．以上の実験から以下の状況下で TDI は効率的な問い合わせ処理を行うことができると結論づけることができる．

- ある程度そのピアの嗜好パターンに従った問い合わせ処理を行う．
- 更新される文書を共有している．
- 非常に動的なネットワークでの問い合わせ処理を行う．

6.4 実験結果に関する議論

本実験での標準の設定では，単位時間あたりに全ピアの 30% が他のピアへの接続または切断を行う．さらに全ピアの 0.012% がシステムへの参加または離脱を行う．すなわち，各ピアによるシステムへの参加及び離脱または他のピアに対する接続及び切断の確率を頻繁に行うように設定していた．このことは，ネットワークの状況が刻一刻と変化させていたことを示す．その設定にも関わらず，DI を用いた場合では，RI を用いた場合よりも少ないインデックス更新頻度にて Gnutella よりも効率的な問い合わせ処理の応答時間を示した．本研究で用いた P2P ネットワークでは 3 章で述べた Small-World 現象が現れる．

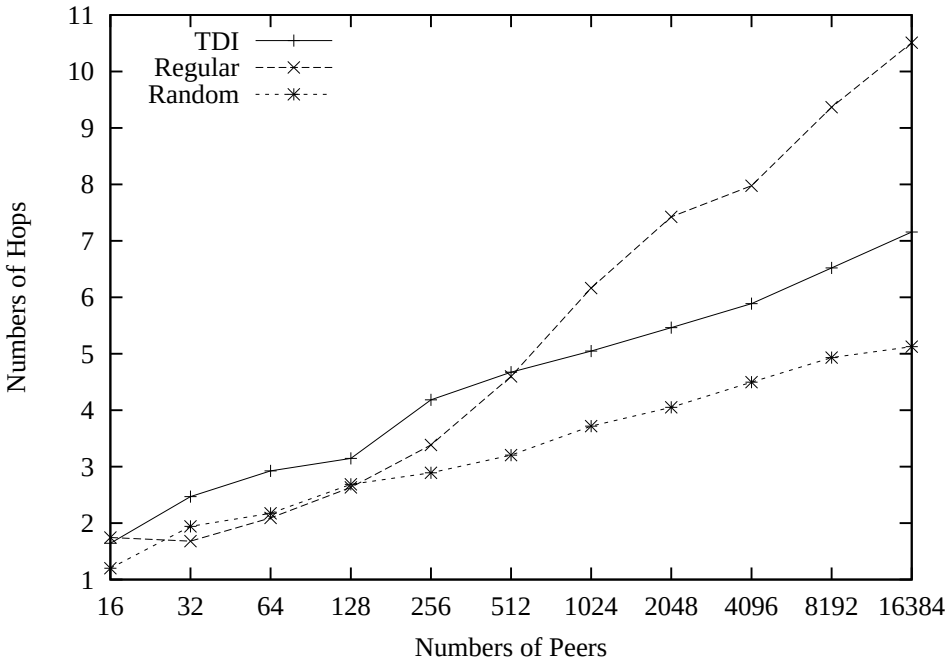


図 6.26　任意のピア間の平均パス長

この現象が現れることによってインデックス更新メッセージ数を低減することが可能であるのにもかかわらず，問い合わせ処理の応答時間も向上することができるであろうと考えている．本節では本研究で用いた P2P ネットワークで Small-World 現象が現れることを示し，Small-World 性を満たす P2P ネットワークが問い合わせ処理に与える影響について議論する．

まず，本研究で用いた P2P ネットワークが Small-World 性を満たすことを示す．3.5 節で述べたように，ネットワークが Small-World の性質を満たすためには平均パス長 L およびクラスタ係数 C を求める必要がある．そこで，本研究で用いた P2P ネットワーク，レギュラグラフもしくはランダムグラフの性質を満たすグラフで L および C を求めることにする．Small-World では各ピア間の距離を決定する必要がある．そこで，本研究では各ピアが実行する問い合わせのキーワードで類似度を計測し，これを用いてピア間の距離を定義し，ピア間の類似度は式（5.2）で決定することにした．レギュラグラフのネットワークを作成するためには各ピアが隣接ピアを選択する方法を決定する必要がある．そこで各ピアはその隣接ピアを選択するときに，そのピアが既知であるピアでもっとも類似度が高いピアから選択することにした．また，ランダムグラフのネットワークを作成する場合，各ピアが既知であるピアからランダムに他のピアを選択し，そのピアに対して接続する．

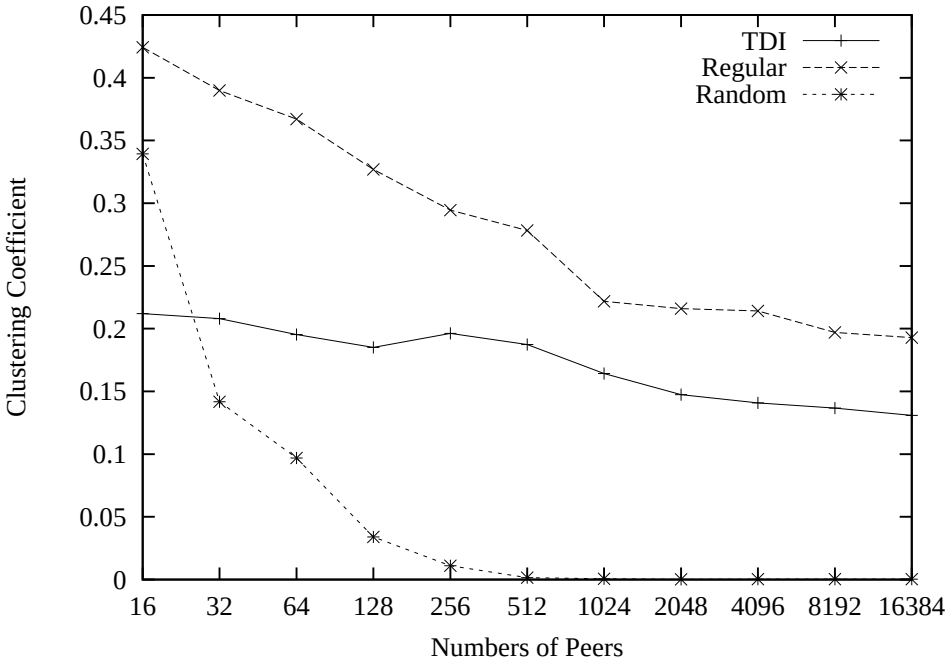


図 6.27 クラスタ係数

以上の 3 つのネットワークでピア数を変動させたときの L の結果を図 6.26 に示す．この図では x 軸を log スケールで表示している．ランダムグラフでは任意のピア間の平均パス長は $O(\log N)$（N は全ピア数）であることが知られている [49]．そのため，ランダムグラフでの任意のピア間の平均パス長は log でスケールする．また，本研究で用いた P2P ネットワークでの結果も log のオーダでスケールすることがこの結果からわかった．そのため，ランダムグラフでの L と本研究で用いた P2P ネットワークでの L は近似することがわかる．

次にピア数を変動させた場合でのクラスタ係数の結果をみることにする．この結果は図 6.27 に示している．ランダムグラフでは任意のピア同士を隣接させるため，ランダムグラフでの C の値は非常に小さくなる．また，レギュラグラフでは同じような嗜好であるピアに対して積極的に接続しようとするため，ピア p の隣接ピアの隣接ピアが p と隣接する確率は他の場合よりも高くなる．本研究で用いた P2P ネットワークはレギュラグラフほどクラスタ係数は高くないがピア数が増加すると，2 つのネットワークでの C の値の差は少なくなる．さらに，ランダムグラフでの C よりもはるかに大きな値であった．以上の 2 つの実験から本研究で用いた P2P ネットワークでは Small-World 性を満たしているといえる．

次に Small-World 性を満たすことが P2P ネットワークに寄与する事項について述べる．Small-World 性を満たす場合，各ピアが送信するインデックス更新メッセージの多くはそのメッセージの起源であるピアと距離が近いピアが受信することになる．そのため，あるピアはそのピアと距離が近いピアと頻繁にコミュニケーションを行う．そのため，ピア間の距離はコミュニケーションの頻度によっても定義することができるかもしれない．またそのメッセージは距離が遠いピアに対しても送信されることがある．ピア間の距離が近くないのにもかかわらず，伝播するインデックス更新メッセージは “有用度” が高いピアの情報に他ならない．そのため，ピア間の距離が近くないピアに伝播される情報は他のトピックに対する問い合わせを行うための “ブートストラップ” としての役割を担うことが可能になる．この現象が寄与する内容の例としては，6.3 節で TDI を用いた場合ではピアの振る舞いとそのピアの嗜好が一致する確率があまり高くない状況下でも，問い合わせの成功率をあまり低下させなかったことを挙げることができる．TDI を用いる場合，長距離リンク先は “有用度” が高いピアであるため，そのリンク先のピア，もしくはそのピアに近いピアの情報を知っているピアである．TDI を用いた場合では長距離リンクが他のトピックに対するブートストラップとして働き，これを利用することで，他のトピックに対する問い合わせも行うことが可能になる．

また，4.4 節でインデックスの整合性が保たれていないが，問い合わせのパフォーマンスをほとんど低下させなかった理由も Small-World 性によるものであると考えられる．Small-World 性によって C が高いため，インデックス更新メッセージの伝播対象の多くがそのインデックス更新の起源に近いピアに対して行われる．そのため，多くのインデックス更新メッセージは起源から非常に近い距離のピアにのみ伝播することがしばしばである．一方，重要なインデックス更新メッセージの伝播は広範囲に伝播する．この情報に対する整合性が保ちにくいが，多くのピア（そのピアとは嗜好が異なるピア）はそのピアとは異なるトピックの問い合わせに用いられることになる．

ネットワーク上のダイナミクスはネットワークの状況を変化させるだけではなく，問い合わせ処理の効率性のために行われている仕組みを低下させる傾向がある．そこで P2P システムでの問い合わせ処理ではいかにその影響を低減させるかが問題になる．これは他の分散システムと P2P システムとの大きな差分になりうる．ネットワークのダイナミクスを考慮するためには，より少ない情報を簡単な仕組みにてネットワーク上に伝播させる必要があるが，少ない情報を元に問い合わせ処理を効率的に行うべきである．これ

を実現するために Small-World の性質を P2P ネットワークが満たすことは非常に有効である．そのため今後，非構造 P2P ネットワークを評価するための 1 つの方法として Small-World 性に関する議論を行うことになるのではないか，と考えている．

さらに，本研究で用いた P2P ネットワークではピアはそのピアと近い距離のピアの状況に関して詳しく知ることが可能である．DI を用いた場合では長距離リンクを形成するため，Gnutella の場合よりも広い範囲の状況を各ピアは知ることが可能であるが，その範囲はネットワーク全体ではない．ある程度の範囲内の状況を各ピアは把握しているがその範囲以外の状況の詳細は把握することができない．しかしながら，詳細を把握していない場所の情報は長距離リンク先のピアを通じて知ることが可能になる．そのため，各ピアは身近な状況は “透明に”，それ以外のネットワークの状況を “半透明” に理解することが可能になる．“半透明” な場所の情報を知るためには，あるピアにとっては “半透明” な場所もその場所を把握しているピアが存在するため，把握したい場所に位置するピアに通知すれば知ることが可能である．これは，実社会にて各個人はその住居範囲のことを知っているがその範囲を超えた場合の情報はやがて不確かなものになるが，その不確かな場所へのたどり着くための経路を知っていれば，その場所に行き，その場所の状況を把握することができる，ということに類似している．このとき，行き先の場所までの途中の経路に対して，次にどのように行けばたどり着くかを把握していない場合，例えばどの電車に乗り継ぐかを把握していない場合，では行き先までのたどり着くことを困難にする．目的地までの完全な経路を各ピアが把握しているのではなく，次にいずれのピアに聞けば目的地に近づくのかを判断できる状況を生み出すことを可能にする．

6.5 本章のまとめ

TDI を用いた場合，他のインデックスを用いた場合に比べトピック数の増加に対応しているといえる結果を得た．トピック数の増加に対して，問い合わせの成功率は低下し問い合わせメッセージ数は増加してしまうが，ホップ数はあまり変化が無かった．この状況下にて TDI と複製配置を組み合わせた場合，TDI のパフォーマンスは飛躍することも分かった．問い合わせ成功率に関して TDI+PIR または TDI+PRP の組み合わせは NDI を用いた場合と比較した場合，大幅なパフォーマンス向上を確認することができた．DI を用いた場合，各ピアは問い合わせメッセージをあまり受け取らないピアが存在する．PRP では他のピアの問い合わせを利用した文書複製の方法である．そのため，多くの問

い合わせメッセージを受け取るピアが文書複製を行う場合，PRP 方式によって効果的な文書複製の配置が期待される．しかしながら，あまり問い合わせメッセージを受け取らないピアが PRP 方式による文書複製配置を行う場合，PRP 方式の効率性を発揮できないためそのパフォーマンスを低下させてしまうと考えられる．

第 7 章

結論

Peer-to-Peer（P2P）システムとはシステムに参加している各ノード（ピア）の役割や機能は等しく，これらのピアによって構成される分散コンピューティングシステムである．オープンな協調作業環境に対して P2P システムを利用する利点として，文書の検索は文書の位置に依存しない，自己組織化に優れている，更新のある文書の効率的な共有が可能である等が挙げられる．しかしながら，協調作業環境では効率的な問い合わせ処理が求められ，そのためには帯域幅消費量の低減，応答時間の向上，システムへ参加するピアの増加への対応，システムからのピアの離脱等が求められる．

オープンな協調作業環境の効率性を向上させるために，まず本研究にて想定した P2P システムを示し，システムレイヤにて問い合わせ処理に関して向上させる要因を挙げた．さらに本研究での P2P ネットワークにおける各ピア間での各セッションでのプロトコルを示した．想定する P2P ネットワークでは非構造 P2P システム上にてピアの “有用度” を用いて長距離リンクを形成する．

そして，問い合わせ処理を向上させる方法としてピアの動向に着目した分散インデックスおよび文書複製の手法を提案した．分散インデックスでは非構造 PP2P システムで導入可能なピア情報を格納する Direct Index（DI）を提案した．また DI の構造，ピア情報の伝播の方法，および DI を用いた問い合わせメッセージのルーティングの方法を示した．DI を実際に利用するためには問い合わせ処理の目的に応じて拡張する必要がある．そこで通常の DI（Normal DI ; NDI）を各ピアのトピック情報を格納するように拡張した Topic-based DI（TDI）を提案し，TDI の構造および TDI を用いた問い合わせ処理の方法を示した．TDI は文書のトピック情報が扱うことが可能であるという条件の下で，あ

表 7.1　DI と従来の非構造 P2P システムの性能比較

	Gnutella	RI	NDI	TDI
応答時間	○	△	○	◎
問い合わせメッセージ数	×	◎	○	◎
インデックス更新	−	○	◎	○
成功率	○	○	×	○
スケーラビリティ	×	○	○	◎

るトピックに関して一定以上の検索結果を取得することを問い合わせ処理の目的とした場合に用いるように DI を拡張したものである．

また分散インデックスでは解決できない問題である文書消失の防止および “有用度” の高い文書の発見の助力のため文書複製のを行い，動的にピアグループ形成し複製を配置する方法（PN 方式，PIR 方式及び PRP 方式）を提案し，さらに文書更新の方法を示した．

提案する分散インデックスおよび文書複製手法を評価するためにシミュレーションを行い，その効果を示した．その結果，本研究で提案した TDI はトピック増加に対して柔軟に対応し，応答時間の向上および帯域幅消費量の低減に役立つことが示された．また，TDI と動的なピアグループ形成を基にした複製配置方法を組み合わせることで問い合わせ処理の成功率も向上させることが示された．特に TDI と PRP 方式または PIR 方式の組み合わせは，TDI の効果のさらなら向上に有効であった．本研究での実験結果に基づいた，従来の非構造 P2P システムでの問い合わせメッセージのルーティングの各手法と DI を用いた場合との性能比較を表 7.1 に示す．さらに本研究での手法は従来の非構造 P2P システムでの手法と異なる点は，各ピアが積極的にコミュニケーションを行う対象を単に隣接するピアだけではなくスケーラブルに選択することを可能にすることである．

以上より本研究で提案した適応的なピア発見を可能にした分散インデックスおよび複製配置の方法は問い合わせ処理の向上への寄与に有効な手段であることが分かった．

本研究では，オープンな協調作業のためのシステムを想定した P2P システムアーキテクチャの一部分の機能向上のための方法を提唱したにすぎない．実際の協調作業のアプリケーションでの問題は実際にシステムを運用しなければわからないことも多いかもしれない．そこで，実際には以下の内容を考慮する必要があると考えている．

- 実際のネットワークでの実験

- トピック抽出

本研究で提案したルーティングや文書複製の手法の評価のための実験はシミュレーションによるものである．これはオーバレイネットワーク上での評価である．しかしながら，実際にシステム運営での問い合わせ処理を評価するためには実際の物理的なネットワーク考慮した上での評価を行うべきであると考えられる．オーバレイネットワーク上では 1 ホップであったとしても実際にはルータをいくつも経由している場合も多いかもしれないためである．実際のシステムでの応答時間や帯域幅消費量と，本研究での実験の結果ではどのくらい差があるかは現在では計測していない．

本研究で行った実験において，本研究で提案した TDI を用いた問い合わせ処理の効率性を示した．実際に TDI を用いるためには，トピック情報を用いることが前提である．本研究では各文書のトピック情報を利用可能であるという仮定の下に，TDI を提案し，実験を行った．しかしながら，P2P システム上で文書からトピックを抽出する方法に対する手法の提案または議論を行っていない．そこで，実際にアプリケーションにて TDI を利用することを可能にするために，P2P システム上でのトピック抽出に対する手法を考慮する必要があると思われる．

謝辞

高須淳宏教授に於かれましては，総合研究大学院大学複合科学研究科情報学専攻博士後期課程の 3 年間の長きにわたりまして，ご指導を賜りました．先生には，研究の過程，内容のあり方，進め方等に関しましてもご指導頂き，また研究内容以外にも研究生活でのサポートをいただきました．心より感謝いたします.

丸山勝巳教授に於かれましては，私の研究の状況に対して貴重なご意見をいただきまして，心より感謝いたします.

相澤彰子教授に於かれましては，研究への貴重なご意見，ご指摘をいただきました．心より感謝しております.

安達淳教授に於かれましても，3 年間にわたりご指導をいただきました．研究の要所に関しまして，その都度，貴重な助言をいただきまして深く感謝いたしております.

山田茂樹教授に於かれましては，私が気づくことの無かった事項に関して大変貴重なご意見をいただきました．心より感謝しております.

河野浩之教授に於かれましては，私の博士審査に快く引き受けていただき，また，貴重なご意見をいただきました．心より感謝しております.

相原健郎助教授に於かれましては，2 年間大変ご貴重な時間をいただき，様々なご指導をいただきました．また，公私にわたり貴重なご意見を賜りました．深く感謝いたします.

最後になりましたが，研究生活以外にも様々な点で支えて頂きました総合研究大学院大学複合科学研究科情報学専攻の同輩，また友人家族に深く感謝しております.

参考文献

[1] Aberer, K.: P-Grid : A self-organizing access structure for P2P information systems, *Ninth International Conference on Cooperative Information Systems (CoopIS 2001)*, Trento, Italy (2001).

[2] Al-Houmaily, Y. J. and Chrysanthis., P. K.: Two-Phase Commit in Gigabit-Networked Distributed Databases, *The* 8^{th} *ISCA International Conference on Parallel and Distributed Computing Systems*, pp. 554–560 (1995).

[3] Al-Houmaily, Y. J. and Chrysanthis, P. K.: 1-2PC: the one-two phase atomic commit protocol, *The 2004 ACM Symposium on Applied Computing (SAC'04)*, Nicosia, Cyprus (2004).

[4] Albert, R., Jeong, H. and Barabasi, A.: *Error and attack tolerance in complex networkd*, Nature 406 , 378 (2000).

[5] Amaral, L. A. N., Scala, A., Barthelemy, M. and Stanley, H. E.: Classes of small-world networks, *The National Academy of Siences*, Vol. 97, No. 21, pp. 11149–11152 (2000).

[6] Bal, H. E., Bhoedjang, R., Hofman, R., Jacobs, C., Langendoen, K., Ruhl, T. and Kaashoek, M. F.: Performance evaluation of the Orca shared-object system, *ACM Transactions on Computer Systems (TOCS)*, Vol. 16, No. 1, pp. 1–40 (1998).

[7] Bernstein, P. A., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).

[8] Bhattacharjee, B., Chawathe, S., Gopalakrishnan, V., Keleher, P. and Silaghi, B.: Efficient Peer-To-Peer Searches Using Result-Caching, *The 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, USA (2003).

[9] Bhattacharya, S., Brannon, K. W., Hsiao, H., Mohan, C., Narang, I. and Subramanian, M.: Coordinating Backup/Recovery and Data Consistency Between Database and File Systems, *The 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD'2002)*, Wisconsin, USA (2002).

[10] Cao, P. and Irani, S.: GreedyDual-Size: A Cost-Aware WWW Proxy Caching Algorithm, *the 1997 USENIX Symposium on Internet Technologies and Systems(USITS'97)*, Monterey, USA (1997).

[11] CDNJapan: http://www.cdn-japan.com/.

[12] Challenger, J., Iyengar, A. and Dantzig, P.: A Scalable System for Consistently Caching Dynamic Web Data, *The IEEE Infocom '99 Conference*, New York, USA (1999).

[13] Chankhunthod, A., Danzig, P., Neerdaels, C., Schwartz, M. F. and Worrell, K. J.: A Hierarchical Internet Object Cache, *The 1996 USENIX Annual Technical Conference(USENIX'96)*, San Diego, USA (1996).

[14] Chen, Y., Katz, R. and Kubiatowicz, J.: Dynamic Replica Placement for Scalable Content Delivery, *The 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Cambridge, USA (2002).

[15] Chockler, G., Dolev, D., Friedman, R. and Vitenberg, R.: Implementing Caching Service for Distributed CORBA Objects, *17th IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing(Middleware 2000)*, New York, USA (2000).

[16] Clausen, T. and Jacquet, P.: Optimized Link State Routing Protocol (OLSR), IETF RFC 3626 (2003).

[17] Cooper, B. F. and Garcia-Molina, H.: Bidding for storage space in a peer-to-peer data preservation system, *The 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria (2002).

[18] Coral: http://www.scs.cs.nyu.edu/coral/.

[19] Crespo, A. and Garcia-Molina, H.: Routing Indices For peer-to-peer Systems, *The 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria (2002).

[20] Cuenca-Acuna, F. M. and Nguyen, T. D.: Text-Based Content Search and Retrieval in ad hoc P2P Communities, *The International Workshop on Peer-to-Peer Computing(co-located with Networking 2002)*, Pisa, Italy (2002).

[21] Cuenca-Acuna, F. M., Peery, C., Martin, R. P. and Nguyen, T. D.: PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, *The 12th International Symposium on High Performance Distributed Computing (HPDC-12)*, Seattle, USA (2003).

[22] D. Eastlake, r. and Jones, P.: US Secure Hash Algorithm 1 (SHA1), IETF RFC 3174 (1992).

[23] Datta, A., Hauswirth, M. and Aberer, K.: Updates in Highly Unreliable, Replicated Peer-to-Peer Systems, *The 23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, Rhode Island (2003).

[24] Degenaro, L., Iyengar, A., Lipkind, I. and Rouvellou, I.: A Middleware System Which Intelligently Caches Query Results, *17th IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing(Middleware 2000)*, New York, USA (2000).

[25] Delis, A. and Roussopoulos, N.: Performance and Scalability of Client-Server Database Architectures, *18th International Conference on Very Large Data Bases (VLDB'92)*, Vancouver, Canada (1992).

[26] Demers, A., Petersen, K., Spreitzer, M., Terry, D., Theimer, M. and Welch, B.: The Bayou architecture: Support for data sharing among mobile users, *IEEE Workshop on Mobile Computing Systems & Applications (WMCSA1994)*, California, USA (1994).

[27] Deolasee, P., Katkar, A., Panchbudhe, A., Ramamritham, K. and Shenoy, P.: Adaptive Push-Pull: Disseminating Dynamic Web Data, *Tenth International World Wide Web Conference (WWW10)*, Hong Kong (2001).

[28] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R. and Weihl, B.: Globally Distributed Content Delivery, *IEEE Internet Computing*, Vol. 6, No. 5, pp. 50–58 (2002).

[29] Eberhard, J. and Tripathi, A.: Efficient Object Caching for Distributed Java RMI Applications, *18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany (2001).

[30] FreeCache: http://www.archive.org/web/freecache.php.

[31] FreenetProject: http://freenet.sourceforge.net/.

[32] Garcia, J. C. and Ferreira, P.: Concurrency control for distributed cooperative engineering applications, *The 2002 ACM Symposium on Applied computing (SAC2002)*, Madrid, Spain (2002).

[33] Gnutella: http://www.gnutella.com/.

[34] Gray, C. G. and Cheriton, D. R.: Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency, *12th ACM Symposium on Operating Systems Principles(SOSP'89)* (1989).

[35] Gray, J., Helland, P., O'Neil, P. and Shasha, D.: The dangers of replication and a solution, *The 1996 ACM SIGMOD international conference on Management of data*, Montreal, Canada (1996).

[36] Gray, J. and Reuter, A.: *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann Publishers, Inc. (1993).

[37] GrooveNetworks: http://www.groove.net/.

[38] Harren, M., Hellerstein, J. M., Huebsch, R., Loo, B. T., Shenker, S. and Stoica, I.: Complex Queries in DHT-based Peer-to-Peer Networks, *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Cambridge, USA (2002).

[39] Hedrick, C.: Routing Information Protocol, IETF RFC 1058 (1988).

[40] Huebsch, R., Hellerstein, J. M., Lanham, N., Loo, B. T., Shenker, S. and Stoica, I.: Querying the Internet with PIER, *29th International Conference on Very Large Data Bases (VLDB 2003)*, Berlin, Germany (2003).

[41] Hwang, S., Lee, K. S. and Chin, Y. H.: Data Replication in a Distributed System: A Performance Study, *The 7th International Conference and Workshop on Database and Expert Systems Applications (DEXA 1996)*, Zurich, Switzerland (1996).

[42] IBMContentManager: http://www-306.ibm.com/software/data/cm/.

[43] ICQ: http://web.icq.com/.

[44] Ishikawa, Y., Chen, Y. and Kitagawa, H.: An On-Line Document Clustering Method Based on Forgetting Factors, *5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL2001)*, Darmstadt, Germany (2001).

[45] Iyengar, A. and Challenger, J.: Improving Web Server Performance by Caching Dynamic Data, *The USENIX Symposium on Internet Technologies and Systems (USITS'97)*, Monterey, USA (1997).

[46] JXTATechnology: http://wwws.sun.com/software/jxta/.

[47] Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M. and Lewin, D.: Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web, *The twenty-ninth annual ACM symposium on Theory of computing (STOC'97)*, El Paso, USA (1997).

[48] KaZaA: http://www.kazaa.com//.

[49] Kleinberg, J.: The small-world phenomenon: An algorithmic perspective, Technical report, Cornell Computer Science Technical Report 99-1776 (1999).

[50] Kleinberg, J.: Navigation in a Small World, Nature 406 (2000).

[51] Kleinberg, J.: The small-world phenomenon: An algorithmic perspective, *32nd ACM Symposium on Theory of Computing (STOC 2000)*, Portland, USA (2000).

[52] Kleinberg, J.: Small-World Phenomena and the Dynamics of Information, *Neural Information Processing Systems 2001 (NIPS*2001)*, Vancouver, Canada (2001).

[53] Kossmann, D.: The State of the Art in Distributed Query Processing, *ACM Computing Surveys*, Vol. 32, No. 4, pp. 422–469 (2000).

[54] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C. and Zhao, B.: OceanStore: An Architecture for Global-Scale Persistent Storage, *The 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, Cambridge, USA (2000).

[55] Kung, H. and Robinson, J.: On optimistic methods for concurrency control, *ACM Transactions on Database Systems*, Vol. 6, No. 2, pp. 213–226 (1981).

[56] Litwin, W. and Neimat, M.-A.: RP* : A Family of Order-Preserving Scalable Distributed Data Structures, *20th International Conference on Very Large Data Bases (VLDB 1994)*, Santiago, Chile (1994).

[57] Litwin, W., Neimat, M.-A. and Shneider, D.: LH*: Linear Hashing for distributed Files, *ACM Conference on Management of Data (SIGMOD1993)*, Washington, DC, USA (1993).

[58] Liu, C. and Cao, P.: Maintaining Strong Cache Consistency in the World-Wide Web, *17th International Conference on Distributed Computing Systems (ICDCS '97)*, Baltimore, USA (1997).

[59] Manku, G. S. and Bawa, M.: Symphony: Distributed Hashing in a Small World, *4th USENIX Symposium on Internet Technologies and Systems (USIT'03)*, Seattle, USA (2003).

[60] Milgram, S.: *The Small World Problem*, Psychology Today (1967).

[61] Napster: http://www.napster.com/.

[62] Ng, W. S., Ooi, B. C., Tan, K. and Zhou, A.: PeerDB: A P2P-based System for Distributed Data Sharing, *The 19th International Conference on Data Engineering (ICDE 2003)*, Bangalore, India (2003).

[63] OpenNap: http://opennap.sourceforge.net/.

[64] Patro, S. and Hu, Y. C.: Transparent Query Caching in Peer-to-Peer Overlay Networks, *17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, Nice, France (2003).

[65] Ratnasamy, S., Francis, P., Handley, M., Larp, R. and Shenker, S.: A scalable content-addressable network, *ACM SIGCOMM 2001*, San Diego, USA (2001).

[66] Rowstron, A. and Druschel, P.: Pastry: Scalable, distributed object location and Routing for large-scale peer-to-peer systems, *18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany (2001).

[67] Stoica, I., Rorris, R., Karger, D., Kaashoek, M. and Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications, *ACM SIGCOMM 2001*, San Diego, USA (2001).

[68] Sun, Q. and Garcia-Molina, H.: Partial Lookup Services, *The 23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, Rhode Island, USA (2003).

[69] Sundsted, T.: How practice of peer-to-peer computing: Discovery, http://www-106.ibm.com/developerworks/library/j-p2pdisc/.

[70] SunMicrosystems, I.: Project JXTA: An Open, Innovative Collaboration, Technical report, Sun Microsystems, Inc., Project JXTA General Papers http://www.jxta.org/project/www/docs/OpenInnovative.pdf (2001).

[71] Takase, T. and Tatsubori, M.: Efficient Web Services Response Caching by Selecting Optimal Data Representation, *24th International Conference on Distributed Computing Systems(ICDCS 2004)*, Tokyo, Japan (2004).

[72] Tang, C., Xu, Z. and Mahalingam, M.: pSearch: Information Retrieval in Structured Overlays, *First Workshop on Hot Topics in Networks (ACM HotNets-I)*, New Jersey, USA (2002).

[73] Tewari, R., Dahlin, M., Vin, H. and Kay, J.: Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet, *The 9th International Conference on Distributed Computing Systems (ICDCS'97*, Austin, USA (1999).

[74] Tsoumakos, D. and Roussopoulos, N.: Adaptive Probabilistic Search for Peer-to-Peer Networks, *Third International Conference on Peer-to-Peer Computing (P2P'03)*, Linkoping, Sweden (2003).

[75] Watts, D.: *Networks, Dynamics, and the Small-World Phenomenon*, The American Journal of Sociology, Vol.105 (1999).

[76] Watts, D.: *Small Worlds: The Dynamics of Networks Between Order and Randomness*, Princeton Studies in Complexity (1999).

[77] Watts, D. and Strogatz, S.: Collective dynamics of 'small-world' networks, Technical report, Nature. 1998 Jun 4;393(6684):440-2 (1998).

[78] Yang, B. and Garcia-Molina, H.: Comparing Hybrid Peer-to-Peer Systems, *27th International Conference on Very Large Data Bases (VLDB 2001)*, Roma, Italy (2001).

[79] Yang, B. and Garcia-Molina, H.: Improving Search in peer-to-peer Networks, *The 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria (2002).

[80] Yang, B., Vinograd, P. and Garcia-Molina, H.: Evaluating GUESS and Non-Forwarding Peer-to-Peer Search, *The 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, Tokyo, Japan (2004).

[81] Yin, J., Alvisi, L., Dahlin, M. and Iyengar, A.: Engineering Server Driven Consistency for Large Scale Dynamic Web Services, *Tenth International World Wide Web Conference (WWW10)*, Hong Kong (2001).

[82] Yu, W., Wang, Y. and Pu, C.: A Dynamic Two-Phase Commit Protocol for Self-Adapting Services, *Services Computing, 2004 IEEE International Conference on (SCC'04)*, Shanghai, China (2004).

[83] Zhao, B., Kubiatowicz, J. and Joseph, A.: Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing, Technical report, U. C. Berkeley Technical Report UCB//CSD-01-1141 (2000).

研究発表一覧

雑誌論文

1. 山田太造, 相原健郎, 高須淳宏, 安達淳: Peer-to-Peer システム上での効率的なデータ配置による問い合わせ処理とロードバランスへの寄与, 情報処理学会論文誌：データベース（TOD22）, Vol. 45, No. SIG5, pp. 93–101 (2004).

レター

1. 山田太造, 相原健郎, 高須淳宏, 安達淳: peer-to-peer システム上での評価の高い peer の発見, 日本データベース学会 Letters (DBSJ Letters), Vol. 2, No. 1, pp. 135–138 (2003).

国際会議（査読有り）

1. Yamada, T., Aihara, K., Takasu, A. and Adachi, J.: A Distributed Index System for Efficient Query Processing in Peer-to-Peer Networks, *2003 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'03)*, Victoria, Canada (2003).
2. Yamada, T., Aihara, K., Takasu, A. and Adachi, J.: An Index System for Dynamic Peer-to-Peer Network Based on Document Usefulness, *1st International Workshop on P2P Data Management, Security and Trust (PDMST'04) in conjunction with the 14th International Conference on Database and Expert Systems Applications (DEXA'2004)*, Zaragoza, Spain (2004).

3. Yamada, T., Aihara, K., Takasu, A. and Adachi, J.: Replica Placement for Effective Document Sharing Mechanisms in Peer-to-Peer Networks, *The 8th IASTED International Conference on INTERNET & MULTIMEDIA SYSTEMS & APPLICATIONS (IMSA 2004)*, Hawaii, USA (2004).
4. Yamada, T., Aihara, K., Takasu, A. and Adachi, J.: A Topic-Based Index Mechanism Using Usefulness of Peers in Unstructured Peer-to-Peer Networks, *The Twenty-Third IASTED International Conference on Databases and Applications (DBA 2005)*, Innsbruck, Austria (2005).

全国大会

1. 山田太造, 相原健郎, 高須淳宏, 安達淳: プライベートネットワークでの peer-to-peer システムを利用したデータ共有, 第 2 回情報科学技術フォーラム（FIT2003）, 江別, 北海道 (2003).

研究会等（査読有り）

1. 山田太造, 相原健郎, 高須淳宏, 安達淳: Peer-to-Peer システム上での効率的なデータ配置による問い合わせ処理とロードバランスへの寄与, データベースと Web 情報システムに関するシンポジウム (DBWeb2003), 東京 (2003).
2. 山田太造, 相原健郎, 高須淳宏, 安達淳: peer-to-peer システム上での評価の高い peer の発見, 第 14 回データ工学ワークショップ (DEWS2003), 北陸・片山津温泉 (2003).
3. 山田太造, 相原健郎, 高須淳宏, 安達淳: 非構造 Peer-to-Peer システム上でのピアの有用性に基づいた問い合わせ処理, 夏のデータベースワークショップ (DBWS 2004), 松山・奥道後温泉 (2004).