インデクスサーバの自律形成によるピアツーピアシステムの動的効率化

吉田　紀彦[†]　内田　良隆[††]　楢崎　修二[††]　瀬川　淳一[†††]
下川　俊彦[††††]

Dynamic Optimization of Peer-to-Peer Systems by Autonomous Index-Server Formation

Norihiko YOSHIDA[†], Yoshitaka UCHIDA[††], Shuji NARAZAKI[††], Jun'ichi SEGAWA[†††], and Toshihiko SHIMOKAWA[††††]

あらまし　コンテンツをネットワーク上の各ノード（ピア）に分散させるピアツーピア（P2P : Peer to Peer）システムでは，必要とするコンテンツの所在，すなわちインデクスを検索する方式の効率化が重要であり，例えば Napster ではインデクスを一元管理するサーバを置き，そこへのアクセス集中をまねいている．一方で，Gnutella では問合せをブロードキャストすることでインデクス検索を行い，ネットワークトラヒックの増大をまねいている．本研究では，インデクスサーバを動的，自律的かつ分散的に形成し，更に必要に応じて再配置していくことで，アクセス集中，トラヒックをともに軽減する P2P システム「AmorphicNet」を設計し，その有効性を実験で確認した．

キーワード　ピアツーピア，動的負荷分散，自律的ネットワーク再構成

1. まえがき

一般にネットワーク分散環境での情報共有は，中央サーバによる集中型から全ノードに複製を配置した完全分散型まで，様々な形態をとる．共有情報の参照や更新における複製間の一貫性を維持するためにノード内処理やノード間通信が必要であり，それらを最適化するようなネットワーク内の複製配置と（論理的な）ノード間接続関係，すなわちネットワークトポロジーの最適形は，ネットワーク性能，参照更新頻度，複製間の一貫性制約の強度などによって左右される．これらのパラメータが動的に確定する場合，変動する場合には，したがって最適なネットワークトポロジーも動的に変化する．

特定ノードへの負荷集中やネットワーク全体のトラヒックを軽減するためには，それらの状況を動的に監視し，ネットワークトポロジーを動的に再構成して最適化していく必要がある．その過程の解明と方式の構築を筆者らはこれまで進めてきており，

平たん構造：　各ノードが共有情報の（部分的な）複製を保持し，対等通信で管理する．

集中構造：　中央サーバで共有情報を一元管理する．

協調構造：　複数サーバで共有情報を協調管理する．

といったトポロジーの間で，パラメータの変化に追従して相転移する過程を明らかにし，実際にシステム設計と試作も行ってきている [1]～[3].

本研究ではこのネットワーク動的再構成を，ピアツーピア（Peer-to-Peer，以下 P2P）システムにおけるインデクスデータベースという共有情報の分散管理に適用する．

P2P システムは，インターネット上で主流のサー

† 埼玉大学工学部，さいたま市
Faculty of Engineering, Saitama Univeristy, Saitama-shi, 338–8570 Japan
†† 長崎大学工学部，長崎市
Faculty of Engineering, Nagasaki University, Nagasaki-shi, 852–8521 Japan
††† 株式会社東芝研究開発センター，川崎市
Corporate R&D Center, Toshiba Corporation, Kawasaki-shi, 212–8582 Japan
†††† 九州産業大学情報科学部，福岡市
Faculty of Information Science, Kyushu Sangyo University, Fukuoka-shi, 813–8503 Japan

バクライアントシステムにおけるサーバへのアクセス集中を軽減しするものとして（また，配信情報の匿名性を高める手段としても）注目されており，Napster, Gnutella などがよく知られている [4]．ただし，P2P システムはサーバクライアントシステムと相対立するものではなく，それぞれ分散情報共有の一形態ととらえるべきものである．P2P システムでは配信情報，すなわちコンテンツは各ノード（ピアともいう）に分散配置されており，したがってコンテンツの所在，すなわち，インデクスを獲得する方式の効率化が本質的に重要である．例えば，Napster ではインデクスデータベースを一元管理する中央インデクスサーバを置き，そのサーバへのアクセス集中をまねいている．一方で Gnutella では必要なたびに問合せをブロードキャストすることでインデクス検索を行い，ネットワークトラヒックの増大をまねいている．

本研究では，予見的・集中的・全域的な制御なしに自律的・分散的・適応的にインデクスサーバを形成し，しかも必要に応じて再配置することで，負荷一極集中，トラヒックともに軽減する P2P システムを設計した．本論文ではその「AmorphicNet」について，原理とシステム設計を述べ，プロトタイプを用いた実験の結果を示す．以下，**2.** は背景と関連研究，**3.** はシステムの構成原理，**4.** は設計の詳細，**5.** は実験と評価，**6.** はまとめと残された課題である．

2. P2P システムとインデクス検索

P2P システムではコンテンツをサーバに集約せずに各ノードが分散的に保持する．したがって，それらを参照するには，何らかの方法でコンテンツのインデクスを知る必要がある．最も多用されている方式は，次の二つである．

（1） Hybrid P2P

Napster で導入された方式で，インデクスデータベースを一元管理するサーバを中央に置く．各ノードはそのサーバに問合せを行って，必要なコンテンツのインデクスを検索する．この方式では，サーバへの問合せアクセス集中というサーバクライアントシステムと同じ問題が発生する．これを解決する試みとしては，インデクスサーバのミラーサーバを置く FastTrack [5] があり，KaZaA [6] や Glokster [7] などに用いられている．この方式では，処理能力の大きいノードを「スーパーノード」とし，インデクスデータベースをもたせるようにしている．

（2） Pure P2P

Gnutella で導入された方式で，すべてのノードは対等に接続されている．各ノードは自身に接続している全ノードにインデクス問合せをブロードキャストし，それをネットワーク内で繰り返していくことで，目的とするコンテンツにいつかはたどり着くことを期待する．この方式では，ブロードキャストによって大きなトラヒックが発生する．TTL（Time To Live）というパラメータで検索ホップ数の最大値を指定して検索ノード範囲を制御することができるが，小さくすると検索能力がおさえられ，逆に大きくするとトラヒックが増大するため，本質的な解決にはならない．また，インデクスの転送を各ノードでキャッシュする改善策も考えられている．

以上とは別に，ネットワーク上に分散ハッシュテーブルを構成する方式もあり，Content-Addressable Network（CAN）[8]，Chord [9] などが提案されている．また，コンテンツの意味カテゴリーに従ってインデクスデータベースを分割し，範囲を絞って検索を効率化しようとする方式もあり，SIO Net [10] などが提案されている．

一方で，ネットワーク上のノード間接続関係を動的に変化させる研究としては，Small-World モデル [11] に基づくネットワークアルゴリズム [12] を，Freenet [13] の検索経路の再構成に適用した事例 [14] がある．そこでは，近隣ノードへの接続に加えてランダムな相手への接続を張ることで検索効率が向上することが示されている．しかし，存在するコンテンツに対する検索成功が保証されないという難点がある．

3. インデクスサーバの動的形成配置

インデクス検索において，Hybrid P2P の集中構造ではネットワーク負荷は小さいが，中央サーバノードの負荷が突出する．一方で，Pure P2P の平たん構造では全ノードの負荷は均一だが（実際には均一に大きくなる），ネットワーク負荷が大きい．したがって，その二つを両端とする軸のどこか中間に，ネットワーク負荷とノード負荷の両者について適切な値を与える点が存在する．そこでは，複数のインデクスサーバが分散的に配置され，インデクス検索が分散的に処理されることになる．これを図式的に図 1 に示す．

ここで，いくつのサーバをどのように配置するかを事前に静的に決定しておくことはできない．それらは検索トラヒックの分布状況から動的に決定しなければ

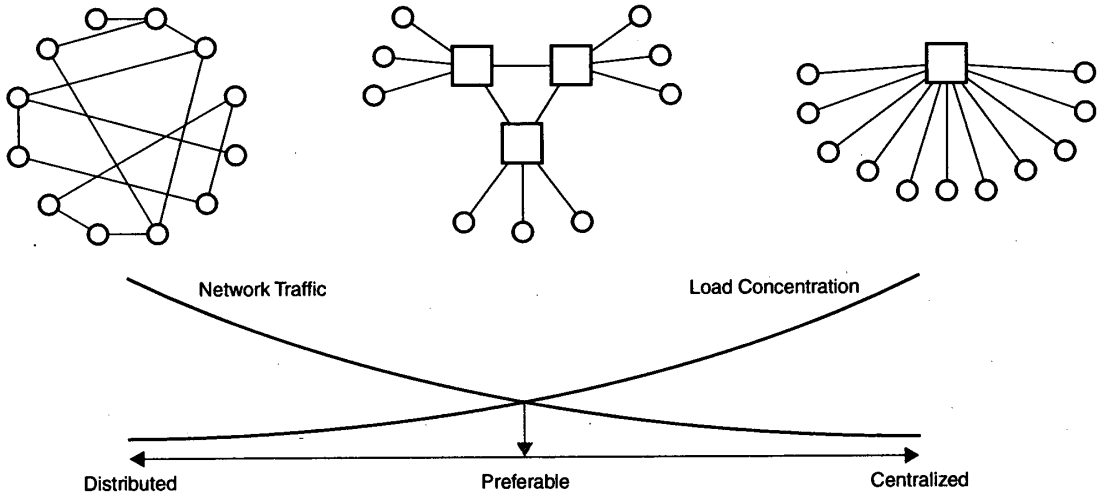


図 1　分散情報共有の最適化
Fig. 1　Optimization of distributed information sharing.

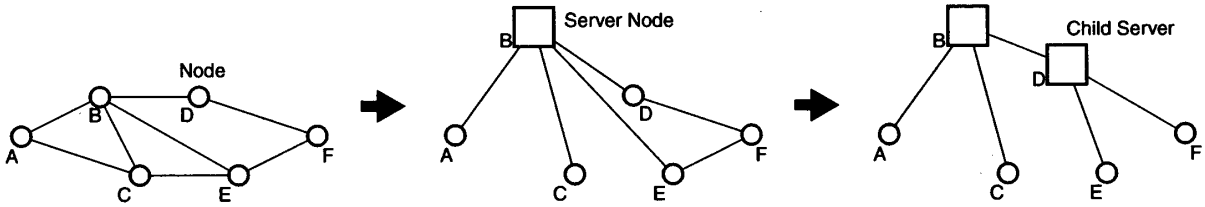


図 2　サーバの動的形成配置
Fig. 2　Dynamic organization of servers.

ならない．また，P2P 自体がコンテンツの分散共有をめざすこともあり，最適化を中央集中制御なしに自律的に，かつできるだけ分散的に行うことが望ましい．

本研究で構築している P2P システム「Amorphic-Net」は，図 2 に示すように，Pure P2P と同じくインデクスサーバのない対等なノードからなる平たん構造を初期状態とし，各ノードは自らの周辺のトラヒックを監視しつつ，必要に応じて自らをサーバに格上げし，更に複数サーバ間で再構成を行っていく [15]～[17]．一方で，トラヒックや負荷が減少した場合には，逆のサーバ格下げによって階層構造から平たん構造への再構成も行う．

以下，システムの概略を述べる．トラヒックと負荷に応じたサーバの自律形成を効果的に実現するための基本設計を，項目ごとにまとめる．

（1）　Pure P2P と Hybrid P2P の融合

本システムでは，Pure P2P 型の平たん構造と Hybrid P2P 型の階層構造が混在する．また，従来の Hybrid P2P と異なり，サーバはインデクスの完全なデータベースをもつのではない．したがって検索は，まず Hybrid P2P 的にサーバへの問合せを行い，失敗したならば Pure P2P 的なブロードキャストに切り換える．

（2）　サーバ形成の契機

サーバはネットワークトラヒックの集積点に形成することで，その効果をより発揮できる．すなわち，あるノードの近傍のトラヒックが過大になったならば，そこにサーバを形成すべきである．

（3）　トラヒックの推定

一般にネットワークトラヒックの測定ないし推定には様々な手法がある．本研究では，各ノードが自らの近傍の局所的な情報を必要とすることから，自らを通過するパケットの数量でトラヒックを近似する．

（4）　インデクス情報のサーバへの集積

新たに形成されたサーバは，サービスすべきインデクスデータベースを動的に獲得する必要がある．本研究では，サーバの周辺ノードからインデクス情報を集積することで，これを構築する．

（5）　サーバの増殖

サーバの負荷が過大になったならば，更にサーバを

増殖させて負荷分散を行う．サーバ負荷は検索パケット数，または自らがサービスする周辺ノードの個数などで近似できる．なお，この際に，(i) インデクスデータベースを複製することでミラーサーバとしてアクセス分散を図る，(ii) インデクスデータベースをインデクスの値に従って分割することで検索効率化を図るなど，いくつかの選択肢が考えられる．

（6） サーバの消失

上と逆に，負荷が減少してサーバが不要になったならば，サーバを消失させて負荷集中を軽減する．

（7） 障害対策

一般に Pure P2P は，障害に対する耐性が高い．(モバイル環境などでの) サーバの切離し，ないし何らかの障害によってサーバが利用できなくなった場合には，Hybrid P2P 型から Pure P2P 型に移行すればよい．

（8） システムの収束性

例えば，ネットワークがサーバのみになるような際限ないサーバの形成増殖，形成増殖と消失を繰り返すトポロジー振動などを回避し，一定の状況のもとでは安定的なトポロジーに収束する必要がある．本研究では，形成と消失にヒステリシスをもたせるほか，必要な手立てを導入する．

4. システム設計の詳細

4.1 初期状態

AmorphicNet の初期状態は Pure P2P と同一である．各ノードはいくつかのノードと接続関係のリンクをもち，そのノードの存在を知っているとともに，情報をやり取りできる．リンク相手のノードを「近隣ノード」と呼ぶ．

なお，ネットワークに新たに加入しようとするノードは，既に加入しているノードを最低一つ，何らかの方法で知り，それらとの間にリンクを張ることで加入する．

4.2 インデクスのキャッシュ

初期状態では，インデクス検索は近隣ノードすべてに問合せパケットを送出することで行う．近隣ノードから問合せパケットを受けたノードは，自らがそのコンテンツをもつか，または該当インデクスをキャッシュ内にもつならば，そのインデクスを，検索キーワードと所在情報の対からなる検索結果パケットとして，逆向きに送り出す．もたない場合は，その問合せパケットを近隣ノードすべてに転送する．

検索結果パケットもノード間で転送されていき，検索を開始したノードがそれを受けたら，インデクスからコンテンツを保持するノードを知り，そこにコンテンツを要求する．この検索結果の転送の際に，各ノードは自らを通過する検索結果パケットからインデクスを自らに取り込んでキャッシュする．

なお，キャッシュは容量に制限はなく，一定の生存期間（TTL，ただし，Gnutella の TTL とは異なる）をもち，それを超えたインデクスは消失，揮発する．この生存期間の制限はインデクス更新に対応するためである．

このインデクスキャッシュの導入だけでもトラヒック軽減に寄与するのに加え，本システムでは更に，ノードを通過するトラヒック，すなわち自らを通過するパケットの数量を近似的に把握するのにキャッシュの大きさを用いる．

4.3 サーバノードへの格上げ

トラヒック（キャッシュ数）が一定値を超えたノードは，自らのコンテンツのインデクスとキャッシュ内容のインデクスをデータベースとしてもつサーバに自らを格上げする．ただし，サーバとして機能するに足る一定の計算性能とネットワーク性能をもつノードに限り，また各ノードは自らのサーバ格上げを最初から禁止する選択肢ももつ．

サーバは以下の方法で集積したインデクスデータベースをもち，問合せに対して該当インデクスがデータベース内にあれば，それを回答する．ない場合に近隣ノードへ転送することはせず，「所在不明」の検索失敗パケットを返答する．また，次項で説明するサーバ階層木の根サーバとなる．

一方，サーバでないままのノードを「平ノード」と呼ぶ．サーバの近隣となった平ノードは，問合せをブロードキャストせずにサーバのみに 1 対 1 送信する．「所在不明」の検索失敗パケットが返された場合は，改めてサーバ以外の近隣すべてに問合せをブロードキャストし，初期状態と同じ検索を行う．

また，障害対策として，サーバからの返答が一定時間内に得られずにタイムアウトした場合には，平ノードはサーバに何らかの障害が発生したものと判断して，初期状態と同じブロードキャスト探索に戻る．

（1） 新サーバの初期処理

（i） 近隣の平ノードに，サーバになったことを通知．

（ii） 近隣になった平ノードから送られてくるインデクスセットを自らのインデクスデータベースに追加．

（2） 新サーバの存在を知らされた平ノードの初期処理

(iii) 自らの近隣の平ノードに，次回の問合せへの返信に併せて，新サーバの存在を通知．

(iv) サーバの近隣ノードとなり，次の問合せからはサーバのみに送信する．

（v） 自らのコンテンツのインデクスとキャッシュ内容のインデクスのセットをサーバに送信．

(vi) 自らのサーバ格上げを禁止．

以上の (i) と (iii)，(iv) によって，サーバの存在は徐々にネットワーク内に知れ渡っていき，サーバの近隣ノードも増加していく．一方で (vi) によって，サーバの際限ない増加を抑制する．また，(ii) と (v) によって，インデクスがサーバのデータベースに集積されていく．

以上の手順で，いくつかのノードが互いに独立にサーバになることもありえる．しかし，それらの間でデータベースの一貫性を確保することはしていない．

4.4 インデクスサーバの動的再構成

サーバの近隣ノードが増加して負荷が高まった場合，近隣の平ノードの一つにサーバ格上げの要請を行い，自らのインデクスデータベースを複製して渡す．これによってサーバ負荷分散を図る．このように，現在の設計では，最も基本的な方式として，ミラー的な複数サーバ（ただし後述のように，厳密に完全な複製ではなくなる場合がある）を形成するものとしている．これらサーバ群は親子関係に従って階層を構成する．これは後述のように，インデクス更新に効率的に対応するためである．

（1） 既存サーバの再構成処理

（i） 近隣の平ノードのうちから任意に候補を選び，格上げ要請．

(ii) 承諾されたらインデクスデータベースの複製を譲渡，拒否されたら別の候補を選択．

(iii) アクセス分散のために，自らの近隣ノードの半数に新たな子サーバへの変更を通知．

（2） 新しい子サーバの初期処理

(iv) 格上げ要請を受けたら，可否を判断して返答．

（v） インデクスデータベースの複製を親サーバから受領．

（3） サーバ変更通知を受けた平ノードの初期処理

(vi) 次回からの問合せの送信先を新たな子サーバに変更．

なお，アクセス数の減少に伴いインデクスが揮発して減少するか，または何らかの理由で近隣ノードが減少し，複数サーバが不要になった場合は，自らを平ノードに格下げする．すなわち，近隣ノードに，問合せ送信先のサーバを自分から親サーバに切り換えるように要請する．

このようなサーバ消失の際にはサーバ木も，祖母と孫を新たな親子として関係づけ直すなどの再構成を行う．

4.5 問合せの転送と回答の中継

サーバ木内で根サーバ以外のサーバは，問合せに自らインデクスを返答できない場合は，その問合せを親サーバに転送し，親からの回答を中継する．そして，その回答のインデクスを自らのデータベースにも取り込む．ただし，このインデクスも前述の TTL によって生存期間が制限され，揮発していく．

以上から，検索時に他ノードから問合せを受けると，各ノードでは次の処理を行うことになる．

- 平ノード（サーバの存在を知らない） ⇒ 近隣ノードすべてに問合せを送る．
- 平ノード（サーバの存在を知っている） ⇒ サーバに問合せを送る．検索失敗なら，改めてサーバ以外の近隣ノードすべてに問合せを送る．
- サーバノード（木の根ではない） ⇒ 自ら返答する．検索失敗なら親に問合せを転送する．
- 根サーバノード ⇒ 自ら返答する．検索失敗ならその旨を返答する．

4.6 サーバのインデクスの更新

コンテンツの削除・変更・追加があったノードは，そのことをインデクスサーバに通知する必要がある．そのノードは，サーバの存在を知らないならば，何もしない．知っているならば，そのサーバに削除・変更・追加を通知する．通知を受けたサーバは，自らの属するサーバ木の根サーバにもそれを通知する．このインデクスの削除・変更・追加は上述のサーバ木内転送中継の処理によって，いずれサーバ木全体に反映する．

この方式では，特に削除・変更について，それがサーバ木全体に反映するまでは一時的に，各サーバのもつインデクス情報の間の一貫性が損なわれる．しかし，一貫性管理のための特別なサーバ間メッセージが不要なので，ここでは一貫性管理の負荷をおさえることを優先している．その根拠として，第1に，一貫性の破綻（矛盾）は比較的短時間である．第2に，サーバ上のインデクス情報とそれに対応するコンテンツとの間の矛盾は，一般にキャッシュとオリジナルとの間でオ

リジナルの削除・変更の際に発生する矛盾に相当し，例えばキャッシュ付き Gnutella でも同様に発生する．そして，本方式ではコンテンツの削除・変更がサーバに通知されるので，キャッシュ付き Gnutella に比べて改良になっている．

5. プロトタイプ実装と評価

本研究では，まず小規模なローカルネットワーク上でプロトタイプの実装を行い，システムとしての実現性の確認と，有効性の検証実験を行った．具体的には，ノード：SPARC10 × 20 台，接続：100base-TX のネットワークをプラットホームとした．実装には Java を用いている．

(1) 時間測定

まず，上記プロトタイプでの時間的な測定結果を下に示す．日常的に教育研究に運用されている計算機ネットワーク上に構築したことから，有効数字は 1 けたにとどまっている．

問合せ検索：	約 3 ms
サーバ格上げ：	約 3 ms
インデクスセット転送：	約 200 ms

他アプリケーションなどによるマシン負荷やネットワーク負荷に従って値の変動があるが，このように，サーバ格上げ処理の時間は問合せ検索処理と同等で，インデクスデータベース構築のためにインデクスセットを集積する時間が支配的である．

(2) 効率評価

同じプラットホーム上に Hybrid P2P の Napster, Pure P2P の Gnutella，及びキャッシュ機構を付加した Gnutella の三つも Java でクローンを実装して比較を行った．その実験結果を以下に示す．ここでは，サーバ格上げの契機：キャッシュ数 16，サーバ増殖の契機：近隣ノード数 7 とし，問合せは各ノードで（0〜120 秒の間隔で）ランダムに生成した．

図 3 にネットワーク内の総パケット数，図 4 にアクセス最大ノードのアクセス数，図 5 に検索に要する平均ホップ数について，それぞれ時間的な推移を示す．いずれも 3 分ごとの累積値，最大値，平均値であり，複数回の試行を平均した結果である．ネットワークトポロジーはおおむね下記のような傾向を示している．

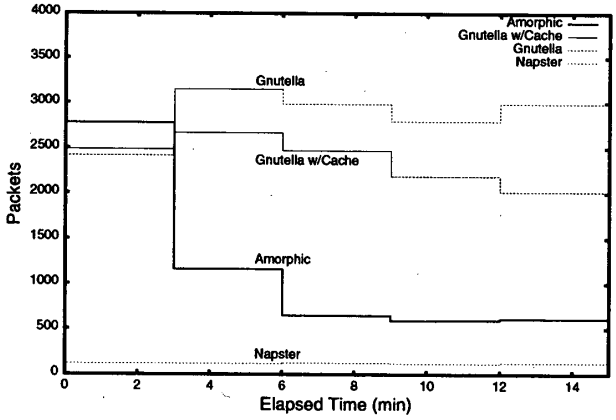


図 3 総パケット数の推移
Fig. 3 Transition of packet amounts.

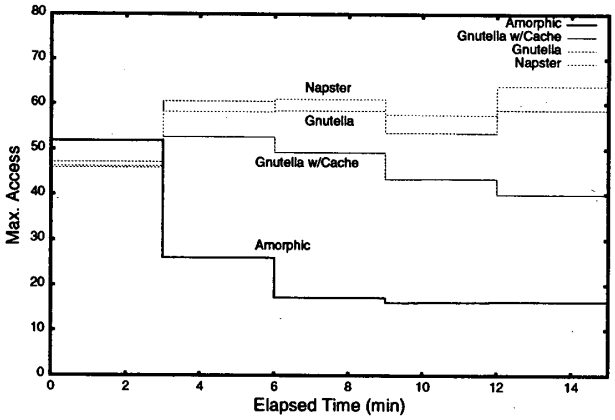


図 4 最大アクセス数の推移
Fig. 4 Transition of maximum access per node.

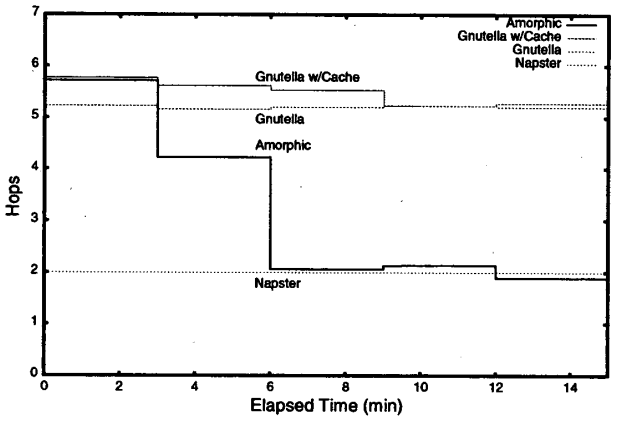


図 5 平均ホップ数の推移
Fig. 5 Transition of averege hop counts.

3〜6 分後：	最初のサーバ形成．一部，複数の独立なサーバ形成も．
6〜9 分後：	子サーバ形成，すべての平ノードがいずれかのサーバに接続．
15 分後：	サーバ 4〜5 個．

参考値として，15分後には，サーバのインデクスデータベースの平均サイズ：340，平ノードのキャッシュの平均サイズ：52，となっている．

なお，Gnutellaでは，問合せパケットがブロードキャストの繰返しによってネットワーク内に「充満」するため，最大アクセス数は一極集中型のNapsterと大差ない．キャッシュを導入すると，コンテンツを保持するノードよりも遠くにあってキャッシュを保持するノードも応答を返すようになるため，平均ホップ数が増える．また，自らがもつコンテンツの検索はホップ数0のため，平均がNapsterの2（往復各1）を下回ることもありえる．

このように，トラヒックを表す総パケット数，アクセス集中度を表す最大アクセス数，検索効率を表す平均ホップ数のいずれも，時間経過に従ってPure P2Pから改善されていくこと，特に最大アクセス数は最終的に他のすべての方式を下回ること，平均ホップ数は最終的にHybrid P2Pと遜色なくなることを確認した．

（3） 動的適応性

次に，ネットワークの動的変化，すなわちノードの加入や離脱，また障害に対する適応性に関して行った実験の結果を示す．なお，障害については，ここではノードが停止する障害，すなわち停止障害及びオミッション障害を考え，したがってノードの予告なしの離脱と同様に扱うことができる．

ネットワークの動的変化が発生する基本的な状況としては，以下がある．

（i） ノードの加入
（ii） 平ノードの離脱
（iii） 検索結果転送中の中継ノードの離脱
（iv） サーバの離脱に伴う平ノードの動作変更
（v） サーバの離脱に伴うサーバ木の再構成

これらのうち，(ii)はHybrid P2P，Pure P2Pを問わず，(iii)はPure P2Pでも全く同様に発生する．また，(i)は，新ノードは平ノードとして加入するので，Pure P2Pと同様である．そこで，(iv)と(v)，すなわちサーバの予告なしの離脱に対する適応性について，先の実験と同じプロトタイプのうえで実際に変化を発生させて，検証を行った．

［実験1］ サーバとその近隣の平ノード複数という状況で，サーバを強制停止させる．平ノードはいずれもサーバが切断されたことを検出して，自動的にブロードキャスト検索への切換えを行う．

［実験2］ 複数サーバが2段以上の木を構成している状況で，末端（葉）以外のサーバを強制停止させる．停止したサーバの子サーバは自分の親が切断されたことを検出して，自動的に自らが根サーバになり，自らの木に属する子サーバ以下にその変更を伝達していく．停止したサーバの親サーバは，切断された子を抜きに処理を続行する．

以上の挙動を確認した．なお，いずれの対応処理も，先の問合せ検索，サーバ格上げとほぼ同じ時間で実行できている．

6. むすび

本研究で設計，実装したAmorphicNetでは，コンテンツの所在を表すインデクスデータベースを保持するサーバを動的，自律的かつ分散的に形成配置することによって，従来のPure P2Pに比べてネットワークトラヒックの削減，Hybrid P2Pに比べて一極集中の回避，更に検索効率の向上が達成できることを確認した．サーバの配置はトラヒックに応じて自律的に定まるため，事前に静的に定めておく必要がないのも本システムの長所である．

現在は第1に，しきい値などのパラメータ設定に伴うトポロジー再構成の挙動変化について詳細な解析を行いつつあり，その一助とすべく，大規模ネットワークのシミュレーションシステムも開発中である．

第2に，いくつかの改良に着手している．例えば，サーバ格上げ時のキャッシュ収集に時間を要するので，これをバックグラウンド処理化して問合せ処理と並行に進める，などの効率化である．

第3に，サーバ格上げに関して，現在の実装ではサーバとして機能するためのコードを全ノードが最初から潜在的に内包しており，多くのノードは平ノードのままなことから，むだが多い．そこで，サーバをモバイルスレッドとして構成し，サーバ格上げやサーバ増殖をスレッドの移送や複製として処理する実装を，Aglets [18], [19] を参考にして進めている．これによって，何らかの理由でサーバノードがネットワークから離脱する場合も，そのサーバスレッドをデータベースごと別ノードに移送することで，サーバ機能を継続できるようになるのも利点である．

第4に，CANやChordで有効性が示されている分散ハッシュテーブルとの融合を進めている．すなわち，サーバ増殖の際に，インデクスデータベースを単に複製するのでなく，インデクスのハッシュ値に従って分割する．そして，平ノードはハッシュ値に従ってサー

バを選択するようにする．これによって更なる検索の効率化をめざす．

最後に，P2Pにおける障害対策は基本的に，ノードが停止する障害を想定していることが多いが，ノードが（悪意によるものも含めて）正しくない応答をするコミッション障害まで考える必要がある．そこで，ネットワーク故障診断理論に基づく分散協調型のコミッション障害の検出及び復旧に関する筆者らの過去の研究成果[20]の組込みも始めている．

謝辞 初稿に対して貴重な御意見を頂戴した査読者の方々に感謝する．

文　献

[1] S. Narazaki, H. Yamamura, and N. Yoshida, "Strategies for Selecting Communication Structures in Cooperative Search," Int'l J. of Cooperative Information Systems, vol.4, no.4, pp.405–422, Dec. 1995.

[2] S. Narazaki, N. Yoshida, H. Hamachi, T. Shimokawa, and K. Ushijima, "Dynamic Copy Allocation Scheme for Distributed Resource Sharing Based on Meta-level Computation," Proc. 1998 Int'l Conf. on Parallel and Distributed Processing Techniques and Applications, pp.829–834, Las Vegas, U.S.A., July 1998.

[3] T. Shimokawa, S. Narazaki, H. Hamachi, and N. Yoshida, "Dynamic Multi-Server Reconfiguration Using Meta-Level Computation in Distributed Information Sharing," Proc. Conf. on Systems, Cybernetics and Informatics 1999, vol.5, pp.274–281, Orlando, U.S.A., Aug. 1999.

[4] A. Oram, ed., Peer-to-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly and Associates, 2001.

[5] http://www.fasttrack.nu/

[6] http://www.kazaa.com/

[7] http://www.grokster.com/

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," Proc. ACM SIGCOMM 2001, pp.161–172, San Deigo, U.S.A., Aug. 2001.

[9] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and Hari Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," Proc. ACM SIGCOMM 2001, pp.149–160, San Deigo, U.S.A., Aug. 2001.

[10] 星合隆成，柴田　弘，酒井隆道，小柳恵一，"意味情報ネットワークアーキテクチャ：SION アーキテクチャ，" NTT R&D, vol.50, no.3, pp.157–164, March 2001.

[11] S. Milgram, The Individual in a Social World: Essays and Experiments, McGraw-Hill, 1992.

[12] J. Kleinberg, "The Small-World Phenomenon: An Algorithmic Perspective," Proc. 32nd ACM Symp. on Theory of Computing, pp.163–170, Portland, U.S.A., May 2000.

[13] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," Lecture Notes in Computer Science, vol.2009, pp.46–67, Springer, 2001.

[14] H. Zhang and A. Goel, "Using the Small-World Model to Improve Freenet Performance," Proc. IEEE Infocom 2002, 10 pages, Los Angeles, U.S.A., June 2002.

[15] 内田良隆，"ピア・ツー・ピア・システムにおけるインデクス情報管理の動的再構成，" 長崎大卒論，Feb. 2002.

[16] 内田良隆，吉田紀彦，楢崎修二，下川俊彦，瀬川淳一，"AmorphicNet：インデクスサーバを動的生成するピアツーピア分散情報共有システム，" 信学技報，vol.102, no.276, pp.1–6, Aug. 2002.

[17] 内田良隆，吉田紀彦，楢崎修二，"インデクスサーバを動的生成配置するP2Pシステム AmorphicNet，" 信学情報システムソサエティ/情処学情報技術レターズ，vol.1, pp.217–218, Sept. 2002.

[18] D.B. Lange and M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, 1998.

[19] http://aglets.sourceforge.net/

[20] 溝口博三，下川俊彦，吉田紀彦，"分散協調型の故障診断と秩序再構成，" 長崎大工学部研報，vol.31, no.57, pp.55–60, July 2001.

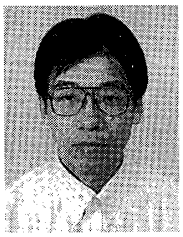
（平成14年11月29日受付，15年3月12日再受付）

吉田　紀彦　（正員）

昭56東大大学院工学系研究科計数工学専攻修士課程了．（株）三菱総合研究所，九大助手，同助教授，長崎大教授などを経て，平14より埼玉大・工・情報システム工学科教授．平5〜6スタンフォード大客員研究員．博士（工学）．プログラミング方法論，並列分散協調処理などの研究に従事．IEEE Computer Society Asia-Pacific Acitivity 委員，各種国際会議プログラム委員・運営委員などを歴任．International Journal of Cooperative Information Systems 常任編集委員．本会情報システムソサエティ/情報処理学会第1回情報科学技術フォーラム論文賞受賞．情報処理学会，日本ソフトウェア科学会，ACM，IEEE各会員．

内田　良隆

平14長崎大・工・情報システム卒．同大大学院工学系研究科情報システム工学専攻博士前期課程在学中．分散処理の研究に従事．本会情報システムソサエティ/情報処理学会第1回情報科学技術フォーラム論文賞受賞．



楢崎　修二

平 2 九大大学院工学研究科情報工学専攻修士課程了．同年日本電信電話（株）入社．平 5 九大工学部助手．平 8 九工大工学部講師．平 11 より長崎大・工・情報システム助教授．博士（工学）．プログラミング言語，並列分散協調処理の研究に従事．本会情報システムソサエティ/情報処理学会第 1 回情報科学技術フォーラム論文賞受賞．情報処理学会，日本ソフトウェア科学会，人工知能学会，ACM 各会員．

瀬川　淳一

平 11 九大大学院システム情報科学研究科知能システム学専攻修士課程了．同年（株）東芝入社．XML 関連の研究に従事．

下川　俊彦

平 4 九大大学院工学研究科情報工学専攻修士課程了．同年（株）東芝入社．平 9 九大大学院システム情報科学研究科助手，平 12 同システム情報科学研究院助手．平 14 九州産業大情報科学部助教授．博士（情報科学）．広域分散協調処理，インターネットなどの研究に従事．情報処理学会会員．