D_050

分散 Wiki による Web ページ間自律的関係創出システム

Autonomous Relation Construction between Web Pages by Decentralization Wiki

小瀬木 浩昭† 河木 孝治† 村田 大樹† 大矢 健太† 鎌田 浩嗣† 武田 正之‡
Hiroaki Ozeki Kouji Kawaki Hiroki Murata Kenta Ohya Hiroshi Kamata Masayuki Takeda

1. はじめに

1.1. 概要

これまで我々は，インターネット上に散在する情報同士の，話題単位のフラットな関係創出を可能とする機構に関して研究を行ってきた[1][2][3][4]．その本質は，(i)独立した複数の Wiki 同士を自動で相互連携する分散 Wiki ネットワークの構築，(ii)分散 Wiki と Blog との統合による，Blog 上に散在する記事の話題単位での相互接続，にある．通常それぞれが孤立している Wiki 同士を，Wiki のページ名である WikiName とその URL を共有することで，個別の管理権はそのままにひとつの巨大な分散 Wiki として振舞うことを可能にする．また分散 Wiki と Blog を統合し，WikiName をキーワードとして Blog の文章に自動的にリンク付けすることで，多種多様な小単位の話題が散在する Blog の特徴を活かした，話題単位での相互接続を可能にする．これにより，インターネット上の情報同士のフラットな関係創出が可能となる．

本稿では，以降 1.2 節で研究背景と本研究の位置づけを明らかにする．そして2章で[1][2][3][4]で提案したモデルの概要を述べ，3章で提案モデルを Wiki へ適用し Blog と連携させる．4章で実装と動作例の一部を紹介し，5章で類似システムからの負荷の傾向予測を行う．6章で考察を述べ，7章で関連研究と比較し本研究の新規性と有用性を裏付け，8章で本稿をまとめる．

1.2. 本研究の位置づけ

近年，Web2.0[5]という言葉が登場している．Web2.0 とは，コンテンツやサービスがサイトと言う枠を超え，インターネットというインフラの上でシームレスに連動し，新しいコンテンツやサービスを複合的に作り出す，そのような状態を指す言葉である．本稿では，動的に HTML が生成される Web サイトを対象として，話題単位で自律的な関係構築が行える環境を提案する．提案手法の位置付けを図 1 に示す．初期の静的な HTML で構成された Web（Web1.0）は，サイト内もサイト間もリンクは手動で行い関係構築の度合いが弱かった（図1左下）．CGI や CMS (Contents Management System)を積極的に取り入れ HTML の大半が動的に生成され，サイト内連携が密な Web（Web1.5）では，サイト内の関係構築が高まった（図1左上）．近年のトラックバックをサポートする Blog や一部の Wiki では，手動ではあるがサイト間の相互連携を支援しサイト間の連携が若干容易化された（図1中上）．それでも依然として検索エンジンの重要性が変わらないのは，サイト間の連携の機能が不十分だからである（図1右下）．

本研究では，独立した複数の Wiki 同士を自動で相互連携する分散 Wiki ネットワークを構築し，分散 Wiki と Blog との統合による，Blog 上に散在する記事を話題単位で相互接続することで，サイト間の連携を助力し，インターネット上の情報同士のフラットな関係創出を可能とする（図1右上）．

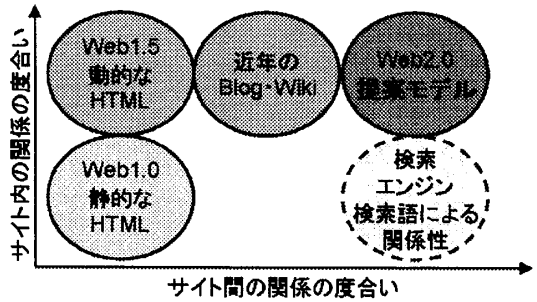

図1 サイト内とサイト間での関係構築可能性の度合い

†東京理科大学大学院 理工学研究科 情報科学専攻，
Graduate School of Science and Technology,
Tokyo University of Science
‡東京理科大学 理工学部 情報科学科，
Dept. of Information Sciences, Tokyo University of Science

2. 本提案モデル

2.1. 提案モデルの概要と構成要素

提案モデルは，構成ノード（1つの Web サイト，以降ノードと呼ぶ）同士が全て対等な関係であるネットワーク構成をとる．提案モデルでは最大のグループとして，全ノードが参加するグループが最低1つはある（ただし，後述する，各サイトが定めるメッセージの最大伝達ホップ数の制限があるため，実際には全ノードの集合より小さなネットワークとなる）．

提案モデルの構成要素を図2に，キーワード作成から参照リンク形成までの流れを図3に示す．ここでキーワードとは，公開された Web コンテンツのリンク情報とそれに付随する情報を表す．

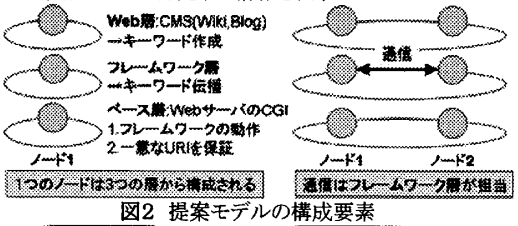

図2 提案モデルの構成要素

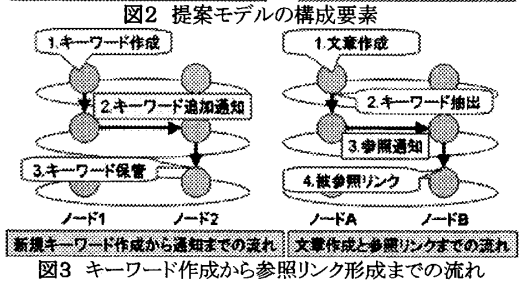

図3 キーワード作成から参照リンク形成までの流れ

提案モデルは，次の3層で構成される．

Web 層:Web サーバ上の CGI として動作している Wiki, Blog, CMS などからのキーワード作成と利用を行う層．

フレームワーク層:キーワード情報など情報の伝播を行う層．

ベース層:一意な URI を保証しフレームワークの動作とデータベースを保管する層．

新規キーワード作成の流れ (1)Web 層である Wiki や Blog 内でキーワードを作成する，(2)キーワードの情報はフレームワーク層によって他ノードに伝播される，(3)伝播された情報はベース層のデータベースに保存される．

参照リンク形成までの流れ (1)Web 層である Wiki や Blog に文章を執筆する，(2)その文章とベース層に保存されているキーワードの情報を比較し，一致するキーワードを抽出する，(3)抽出されたキーワードを作成したノードにフレームワーク層はキーワードへの言及が行われたことを通知する，(4)通知を受け取ったノードは言及した文章へのリンクを掲載する．

2.2. 本提案モデルの特徴

(1) ページ単位で関係を構築

Wiki や Blog はページごとに情報が断片化されて記録される特徴を持つ．ページ単位で関係を構築することで，断片化された情報の連携が円滑になり，情報の再構成と再利用性を高めることが可能となる．

(2) グループ内の全ノードの URI を保持

新規参加ノードが自身の URI を全ノードに向けて1回だけ送信（ただし，各サイトが定めるメッセージの最大伝達ホップ数により制限される）することで，グループ内の各ノードが，常に全ての他ノードの URI を保持していることを高い確率で保証する．これを用いてプッシュ型で情報伝播をすることで，速報性，網羅性を確保した柔軟な情報伝播が可能となる．

(3) ノード間でキーワードという情報を共有

各ノードは複数の「キーワード」を持っている．キーワードは，公開された Web コンテンツのリンク情報とそれに付随する情報である．

2.3. 伝達プロトコル

速報性と網羅性の概念を図4に，網羅・遅延型伝達プロトコルの処理例を図5に，限定・速報型伝達プロトコルの処理例を図6に示す．伝達プロトコルでは，各サイトが持つリストにより，例えるなら SNS(Social Network Service)の友達リストのように，親近度の近い順に優先してメッセージの伝達を行う．伝達プロトコルにより，負荷を考慮した柔軟な情報伝播が可能となる．

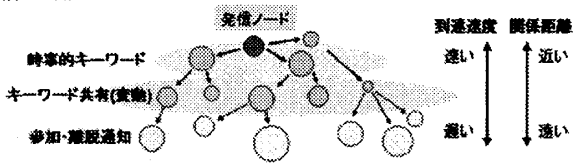

図4 伝達プロトコルの速報性と網羅性

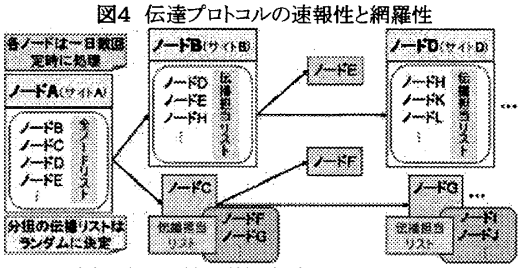

図5 網羅・遅延型伝達プロトコルの処理例

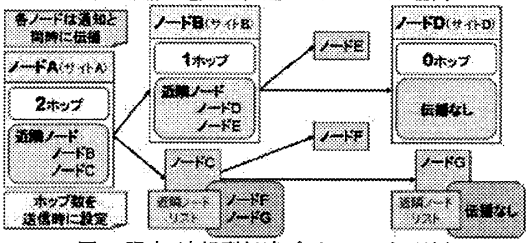

図6 限定・速報型伝達プロトコルの処理例

(a)網羅性が求められる情報伝播

グループへのノードの参加・離脱通知などは，速報性よりも，他の全てのノードへ通知するという高い網羅性が求められる．そのため，全ノードへ通知されるよう伝播を依頼していき，依頼された各ノードでは定期的(1日に数回から数日に1回程度の頻度)に処理がされるこれにより，広くゆったりとほぼ全てのノードへ情報が伝播される．

(b)速報性が求められる情報伝播

時事的なキーワードの追加などが要求される速報性の高い通知は，即時に処理・伝播させる代わりに，少ないホップ数かつ，一定期間に1つのノードが作成できる数を制限し，グループ全体にかかる負荷を抑える．

(c)伝播情報に応じた柔軟な情報伝播

通常のキーワードの追加通知などは，伝播される内容により，処理頻度や経由可能なホップ数を変化させる．即時性が低いキーワードは，伝播処理の頻度を低く抑える代わりに，経由可能なホップ数を多く設定し広範に伝播することを認める．即時性が高いキーワードは，経由可能なホップ数を少なく抑える代わりに，伝播処理の頻度を高く設定する．

3. 提案モデルの Wiki への適用

3.1. 概要

Wiki[6]とは Web ブラウザから利用できる簡便な Web コンテンツ管理システムである．1 つの Wiki サイトは複数の Wiki ページで構成され，1 つの Wiki ページは通常一意な URI を持ち，WikiName 形式(2 文字以上の大文字が含まれる英単語)のページ名とページ本文で構成される．

Web サーバ上の CGI で動作する Wiki を 1 つのノードとみなし，また本提案モデルにおけるキーワードを WikiName とそれに対応する URI とし，これを共有情報とする．これにより，通常 1 つの Wiki 内での閉じた WikiName の共有を複数の異なる Wiki 間で実現することが可能となる．

3.2. 動作の概要

システムの動作概要を図7に示す．なお，簡単のため，ページ名(キーワード)を「●」で表している．

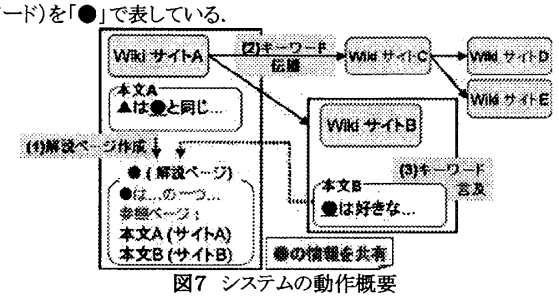

図7 システムの動作概要

(1)ノード A が，キーワード「●」に対応するページを新規作成すると，(2)そのキーワード情報が他のノードへ伝播する．(3)ノード B が●について自身の Wiki サイト内のページ中で言及すると，●に対応するページに自動的にリンク付けされ，また言及通知がノード A に送信され，ノード A 上の●に対応するページに，そのページの言及ページとしてノード B の該当ページへのリンクが掲載される．この仕組みにより Web ページ同士の自律的な関係構築が可能となる．

3.3. 実現システムの詳細

3.3.1. 表記方法

主体 X の公開鍵を $P(X)$ ，$P(X)$に対応する主体 X の秘密鍵を $S(X)$ と表記する．証明書の内容を<>で括って表現し，その証明書に $S(X)$で電子署名が施されていることを，$<...>S(X)$と表現する．主体 Y のネットワーク上での識別子を $URI(Y)$と表記する．Wiki ページ *page* のネットワーク上での識別子を $URI(page)$と表記する．通知文を[通知内容]と表記する．また，紙面の都合上，$\{a \in A,\ b \in B;\ (a,b)\}$を略記して$\{a,b\}$と表記しているところがある．

3.3.2. 各ノードの保持する情報

全てのノードの集合を U, U の部分集合を $G \subseteq U$ とする．G は U のほとんど全ての要素を含む部分集合とする．共有キーワードの集合を K とする 各ノードはそれぞれ以下の情報を保持する．

(1) *Node* リスト: $\{\forall x \in G; URI(x), P(x)\}$

グループ G に含まれるノードの URI，そのノードの公開鍵，の組みのリスト．*NList* と略記する．自身のノード情報はこのリストへ含めない．

(2) *Keyword* リスト: $\{\forall k \in K, \exists x \in G; k, URI(page), URI(x)\}$

共有情報であるキーワード，そのキーワードに対応するページの URI，その対応するページを持つノードの URI，の組みのリスト．1つのキーワードに対して複数の対応するページが存在する場合もある．*KList* と略記する．

(3) *Keyword* 参照リスト: $\{\exists k \in K; k, URI(page)\}$

自サイトから参照を行っているキーワード(自身が作成したキーワードも含む)，そのキーワードへ参照を行っている自サイトのページの URI，の組みのリスト．自身が作成したキーワードへの参照もこのリストへ含める．*KRefList* と略記する．

(4) *Keyword* 被参照リスト: $\{\exists k \in K; k, URI(page)\}$

参照が行われている自サイトのキーワード，そのキーワードへ参照を行っているページの URI(自サイト内のページの URI も含む)，の組みのリスト．自サイト内のページから自サイトのキーワードへの参照もこのリストへ含める．*KRefdList* と略記する．

3.4. 基本プロトコル

各通知は，ヘッダ部とメッセージ部から構成され，ヘッダ部は(*URI*(差出ノード)，*URI*(宛先ノード))から構成される．

3.4.1. ノードの参加

(1) 新規にグループに参加するノード A は，既に参加しているいずれかのノード(ここではノード B とする)から *NList*, *KList*, $P(B)$を受け取り，*NList* に$(URI(B), P(B))$を追加する．

(2) 参加通知として，<ヘッダ部;[参加通知], $URI(A)$, $P(A)$>を，$\forall node \in NList$ に送信する．

(3) 参加通知を受け取った各ノードは，*NList* にノード A に関する情報$(URI(A), P(A))$を追加する．

3.4.2. ノードの脱退

(1) グループから脱退するノード C は，脱退通知として，<ヘッダ部; [脱退通知], $URI(C)>S(C)$を，$\forall node \in NList$ に送信する．

(2) 脱退通知を受け取った各ノードは，受け取った電子署名を保管し

である $P(C)$ を用いて検証し，正当性を確認した後，$NList$ からノード C に関する情報 $(URI(C), P(C))$ を削除する．以下，各通知は同様に正当性の検証を行うものとする．

3.4.3. キーワードの追加

(1) キーワード $NewKeyword$ の追加を行うノード D は，自サイト内に，$NewKeyword$ と本文から構成されるページを作成する．この $NewKeyword$ はページ名(WikiName)となる．

(2) 自身の $KList$ に$(NewKeyword, URI(NewKeyword), URI(D))$を追加する．

(3) キーワード追加通知として，<ヘッダ部; [キーワード追加通知], $NewKeyword$, $URI(NewKeyword)>S(D)$を，$\forall node \in NList$ に送信する．

(4) キーワード追加通知を受け取った各ノードは，$(NewKeyword, URI(NewKeyword), URI(D))$を $KList$ に追加する．

3.4.4. キーワードの削除

キーワードの削除はそれに対応するページを持つノードのみが行えるとする．

(1) キーワード $RmKeyword$ の削除を行うノード E は，自身の $KList$ から，$(RmKeyword, URI(RmKeyword), URI(E))$を削除する．

(2) $\exists page, (RmKeyword, URI(page)) \in K$ Re $fList$ ならば，そこから $(RmKeyword, URI(page))$を削除する．

(3) 同様に自身の $KRefdList$ からも$(RmKeyword, URI(page))$を削除する．

(4) キーワード削除通知として，<ヘッダ部；[キーワード削除通知]，$RmKeyword$，$URI(RmKeyword)>S(E)$を，$\forall node \in NList$ に送信する．

(5) キーワード削除通知を受け取った各ノードは，$(RmKeyword, URI(RmKeyword), URI(E))$を $KList$ から削除する．

(6) $\exists page, (RmKeyword, URI(page)) \in K$ Re $fList$ ならば，そこから $(RmKeyword, URI(page))$を削除する．

3.4.5. キーワード参照追加

ノード F が $page$ を編集するとき，$\exists Keyword \in page \wedge KList$ ならば，次のことを行う．

(1) $Keyword$ に対応するページ $URI(Keyword)$へのリンク付けを行う．この $Keyword$ に対応するページは複数ある場合もある．

(2) $(Keyword, URI(page))$を $KRefList$ に追加する．

(3) キーワード参照追加通知として，<ヘッダ部; [キーワード参照追加通知], $Keyword$, $URI(page)>S(F)$を，$\exists node \in NList$, $(Keyword, URI(Keyword), URI(node)) \in KList$ に送信する．1 つのキーワードに対して，複数のキーワードに対応するページが存在した場合には，その全てへ通知を送信する．

(4) キーワード参照追加通知を受け取った各ノードは $KRefdList$ に $(Keyword, URI(page))$を追加する．

3.4.6. キーワード参照削除

ノード G が $page$ を削除するとき，$\exists Keywoed \in page \wedge KList$ ならば次のことを行う．

(1) $(Keyword, URI(page))$を $KRefList$ から削除する．

(2) キーワード参照削除通知として，<ヘッダ部; [キーワード参照削除通知], $Keyword$, $URI(page)>S(G)$を，$\exists node \in NList$, $(Keyword, URI(Keyword), URI(page)) \in KList$ に送信する．1 つのキーワードに対して，複数のキーワードに対応するページが存在した場合には，その全てへ通知を送信する．

(3) キーワード参照削除通知を受け取った各ノードは $KRefdList$ から$(Keyword, URI(page))$を削除する．

4. 実装・動作例（実行例）

4.1. 処理系

情報伝播のフレームワークは Perl5 を利用して実装した．Web サーバには Apache1.3.29 を利用した．情報伝播のフレームワークと連携する Wiki は PukiWiki1.4.5を，連携する Blog は tDiary 2.0.2をベースにした．

4.2. 動作例

動作例として実行画面のスクリーンショットを図8，図9，図10に示す．図8，図9では，サイト外への自動リンク機能が「[0]」と表されている．1つのキーワードに対応するページが複数存在すれば，[0]，[1]...とリンク付けされる．また，JavaScript と CGI を組み合わせて，本フレームワークに対応していない，一般の Blog でのキーワード共有のための自動リンクの仕組みを実現した(図10)．これは，キーワードリストを本フレームワーク対応のサーバから取得し，取得したキーワードリストと，Blog 中に登場する単語の合致を判定し，リンクの自動推薦を実現する．

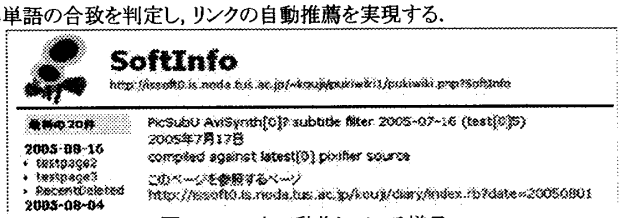

図8 Wiki 上で動作している様子

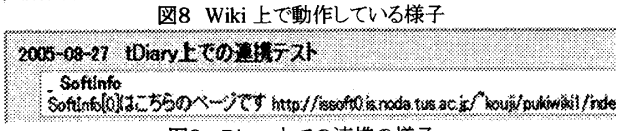

図9 tDiary 上での連携の様子

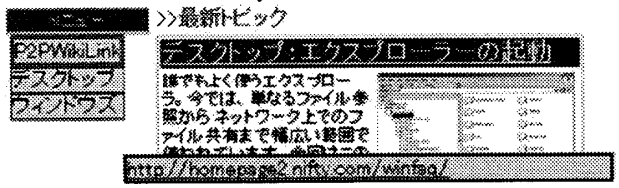

図10 JavaScript を利用した一般サイトへの自動リンク付け

5. 提案システムにおける負荷の傾向予測

提案システムに類似する，はてなダイアリー[10]を調査して得られたアクセス傾向のデータから，提案システムにおける負荷の傾向を予測する．1日記を1 Node，キーワード言及数をキーワード参照数とする．

5.1. 通信における負荷

各 Node 間で通信が発生するのは，任意の Node の参加・脱退，任意の Node のキーワード作成・削除，自 Node のキーワード参照追加・削除，自キーワードの参照追加・削除，の各処理の際である．本ネットワーク全体の1日の通信回数は，1日の各処理の発生回数と処理1回あたりの通信回数の積の和である．

5.2. 類似システムの傾向データ

表1 はてなダイアリーの調査結果

日記	新日記 / 日	キーワード	新キーワード / 日	キーワード言及/記事	新記事 / 日記・日
19万個	283.95個	12万6千個	172.84 個	4.05 個	1.31 個

実現システムの通信頻度の計算におけるパラメータを決定するため，はてなダイアリーについて調査した(2005.9)．公開されているデータ，および最近更新された日記を 50 件おきに調べた結果によると表1のようになる．なお日記数とキーワード数は，最近 16 ヶ月で共にほぼ線形に増加している．また，日記や記事，キーワードの減少数は不明だが作成件数に比べれば無視できる程度の数と考えられるため，今回は考慮から外した．

5.3. 負荷傾向の推定

表2 各処理の通信回数

	新 Node 参加	キーワード作成	キーワード参照追加	キーワード被参照追加	計
発生回数/日	283.95	172.84	$5.32\times n$	$5.32\times n$	
通信回数/1 処理	n	$2\times(n\text{-}1)$	1	1	
はてな規模(n=19万)の時（万回）	5395.1	6567.9	101.08	101.08	12165.16

Node 数と 1Node あたりの平均通信回数の関係は図5のようになった．各処理に伴う通信の発生回数は表2のようになった．1 Node あたりの平均通信回数は1日に 640.27 回，つまり平均 2 分に 1 回程度の通信頻度になると推定される．多くのキーワードはコンスタントに言及されており，例えば，はてなダイアリーの 2005 年 9 月 24 日現在の最も言及されているキーワードである「orz」は1日約 750 回の言及数であった．一方，時事的なキーワードとして言及数の多い「台風」は，過去3ヶ月間のうちでの最大が 4500 件，「地震」は 3800 件の言及数であった．仮に最大1日に 5000 回の言及数のあるキーワードを持った場合，通常の負荷と合わせると1日に 5640 回，平均で 1 分間に 4 回程度の通信頻度となるが，こ

の程度の負荷であれば通常の利用の範囲内と考えられるため，通常のアクセス負荷の範囲内であるといえる．

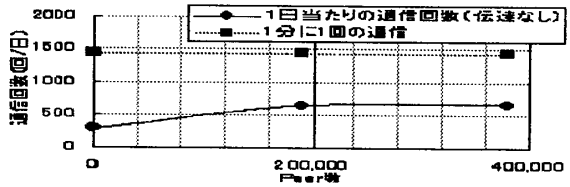

図5　Node(Peer)数と1 Node(Peer)あたりの通信回数の関係

6. 考察

6.1. 従来のロボット型検索手法と提案手法との比較

従来のロボット型の検索手法と提案手法を比較する(表3)．ある程度古い文章に対して，従来手法は特に優れているが，クロール頻度より新しい文章を検索することはできない．提案手法はそういった新しい文章の多くを検索することができる．負荷に関しては従来手法，提案手法ともに実際に検索されるか分からない情報についてもデータを保持する必要がある．検索結果の適合性に関しては提案手法では元々解説されていることが分かっているキーワードについて検索するため，特に優れている．再現性に関しては Web 全体では古い文章の方が多いため従来手法が優れている

表3　ロボット型検索手法と提案手法との比較

	古い文章	新しい文章	負荷	適合率	再現率
従来	◎	△	△	○	○
提案	○	◎	△	◎	△

6.2. セキュリティ機構

各ノードは専用の PKI に基づく公開鍵と秘密鍵の組を生成，保持する．各通知には送信ノードの電子署名を付けることでなりすまし，改ざんを防ぐ．各ノードはノードリストの中に公開鍵を保持しているので独自に検証することが可能である．そのため，スパムなどの問題を起こす Node に対して個別に対応することが可能である．

6.3. ノードによる選別

伝播情報を受け取った各ノードがその内容を確認し，その内容が有益でないと判断した場合には，その伝播を取り止める，ノードによる選別を考える．これにより，各ノードの判断で洗練された有益な情報のみが情報伝播の過程で生き残っていく．また，ノードが伝播を阻止する確率を p，各ノードが通知を受け取るまでの経由ホップ数の平均を h とすると，1 つのノードが受け取る伝播情報は p^{h-1} に減少する．同様の手法で，通知内容の評価を行うこともできる．

6.4. カテゴリとランクの導入

はてなダイアリーキーワード[10]では，キーワードに「カテゴリ」を導入し，利用者にその言葉の属するカテゴリを明示することで，同音異義語など表記法の違う同じ意味を指す言葉について利用者に選択させる仕組みを用意している．提案手法でも同様にそれぞれのキーワードについてカテゴリを設けることが可能である．また，キーワードは利用者が自由に作成できるため，同じキーワードが複数の Node で作成されてしまうことがある．利用者により淘汰され解説の充実したページが残るのを待つ方法もあるが，キーワードページに投票フォームを設け，正当性の保証に PKI 鍵を用い，利用者によるランク付けを可能とすることで，良質なページを推薦して提示することが可能となる．

6.5. ソーシャル・ブックマークサービスへの応用

ソーシャル・ブックマークサービス(SBS)への提案モデルの適用を考える．各 Node で，ブックマークしたサイトの URI と内容を表すタグを共有情報とすることで分散 SBS が実現できる．また，本稿の適用例で挙げたキーワード共有サービスと組み合わせることにより，キーワードの解説ページと同時に，そのキーワードをタグに持つ SBS に登録されたサイトを提示することも可能となる．

7. 関連研究

[7]は，EcAgent 実行環境というサーバを用いた Hybrid P2P で構成されるが，提案手法は特定のサーバに依存しない．なお，サーバを取り入れればEcAgentと同様の機能が追加でき，また併用も可能である．

[8]では，JXTA を基盤とし，ブローカレスな P2P ネットワークを構築して，意味情報・嗜好情報から類似の Peer を発見する手法を述べている．提案モデルを P2P システムととらえて比較した場合，[8]では各 Peer が PC などの豊富な CPU とメモリを持つことを前提とし，データマイニング，嗜好情報・意味情報活用，メタ情報の類似度の活用などを議論しているのに対し，提案手法は，Web サーバ上の CGI など限られたマシン資源上で，定期処理を基本とし，即時処理を組み合わせ，シンプルな仕組みで大規模な運用にも耐えられることを重視している点が異なる．

Ingrid[9]では，Web サイト毎にインデックスを保持し，複数のサイトにまたがった分散検索を実現する手法を提案している．このような従来の検索がプル型で過去の情報を検索対象とするのに対し，提案手法は未来の登場語に対するプッシュ型検索ということができる．つまり，「キーワード」を設置することで，未来に対する登場語の逐次観察が可能である．未来の登場語に対しては，その都度全文検索する従来手法より，登場したサイトから通知を受ける提案手法の方が効率的である(表3)．

本稿の提案モデルにおけるキーワード共有は，はてなダイアリーキーワード[10]と見かけ上似ているが，C/S で完全集中型のそれに対して，提案手法は，分散環境下で，特定のホスティングサーバに依存せず，[10]と同様の機能を実現可能である．

ソーシャル・ネットワークサービス(SNS)の技術的な本質は，利用者同士の親近度に応じたアクセス制御にある．分散型 SNS の試みとして Affelio[11]などがあげられるが，提案手法は，分散型 SNS と併用可能で，例えば提案手法を適用した Wiki や Blog にさらに SNS によるアクセス制御を付与することで容易に SNS 化することが可能である．

Blog360[12]は，専用の検索システムを用いてブログで話題のキーワードを収集し，ランキングサービスを提供している．ランキングサービスは，全 Node でのキーワードの使用状況を集積する必要がある．提案システムと併用することで利便性の向上が期待できる．例えば提案システムで人気のキーワードを知るためにランキングは有用である．

Technorati[13]はサイトから更新通知を受け，速報性の高い検索を実現している．提案手法におけるキーワード参照通知も同様に，速報性の高いリンク生成を実現し，キーワードのため，検索語句を考える必要がないのが優位な点である．

8. 結論

本稿では，インターネット上に散在する情報同士の，話題単位のフラットな関係創出を可能とするモデルを提案した．その具体化として，独立した複数の Wiki 同士を自動で相互連携する分散 Wiki ネットワークを構築し，分散 Wiki と Blog との統合による，Blog 上に散在する記事を話題単位で相互接続することで，インターネット上の情報同士のフラットな関係創出を可能とするシステムを実現した．本稿で実現したシステムは具体例であるが，提案したモデルは，特定のサービスやアーキテクチャを前提としない汎用性の高いモデルであり，提案モデルの適用範囲は広い．今後の課題は，内部処理の効率化，通信の効率化を図り，大規模なシステムとして動作した場合の定量的な評価，実装システムの公開に向けたプログラムの洗練化である．

参考文献

[1] 小瀬木浩昭，河木孝治，大矢健太，鎌田浩嗣，村田大樹，武田正之: Web ページ間自律的関係構築機能の実現とその評価，データベースと Web 情報システムに関するシンポジウム 2005(DBWeb2005), pp.17-24 (Nov. 2005).
[2] 小瀬木浩昭，河木孝治，大矢健太，鎌田浩嗣，村田大樹，武田正之: Web ベース P2P 情報伝播モデルの提案と自律的リンク構築機構の実現，合同エージェントワークショップ＆シンポジウム 2005, pp.127-134 (Nov. 2005).
[3] 河木孝治，小瀬木浩昭，大矢健太，鎌田浩嗣，村田大樹，武田正之: P2P 型 Web 情報伝播モデルの提案と自律的リンク構築機構の実現，インターネットコンファレンス 2005(IC2005), pp.87-96 (Oct. 2005).
[4] 大矢健太，小瀬木浩昭，河木孝治，鎌田浩嗣，村田大典，武田正之: Web ページ間自律的関係構築機構の実装とその評価，分散システム／インターネット運用技術シンポジウム 2005(DSM2005), pp.31-36 (Dec. 2005).
[5] http://en.wikipedia.org/wiki/Web_2.0
[6] Wiki Wiki Web. http://c2.com/cgi/wiki?WikiWikiWeb
[7] 中村隆幸，井上知洋: アドホックコミュニティを発見する EcAgent 実行環境の実装，情報処理学会論文誌， Vol.45, No.1, pp.94-102 (Jan. 2004).
[8] 中沢実，服部進実: 意味情報と嗜好情報に基づく P2P システムの提案と実装，情報処理学会論文誌, Vol.44, No.3, pp.2666-2677 (Mar. 2003).
[9] Ingrid. 論文リスト http://www.ingrid.org/ja/docs/paper.html
[10] はてなダイアリー. http://d.hatena.ne.jp/
[11] Affelio. http://affelio.jp/
[12] Blog360. http://blog360.jp/
[13] Technorati. http://www.technorati.com/