# Webサーバ間での部分Webグラフ同期方式の提案

高砂　幸代[†]　　小林　亜樹[††]　　山岡　克式[††]　　酒井　善則[††]　　曽根原　登[†††]

†,†† 東京工業大学 大学院理工学研究科　〒152–8552 目黒区大岡山 2–12–1
††† 国立情報学研究所　〒101–8430 東京都千代田区一ツ橋 2–1–2
E-mail: †takasago@net.ss.titech.ac.jp, ††{koba,yamaoka,ys}@ss.titech.ac.jp, †††sonehara@nii.ac.jp

**あらまし**　Web グラフを利用した検索などへの応用を目的として提案している協調型 Web アーキテクチャ（CWA）は，Web サーバが各コンテンツの近傍リンク情報（部分 Web グラフ) の保持，配信機能などを持つ．本稿では，各部分 Web グラフ同期のため，まず，コンテンツ，ハイパーリンクをグラフのノード，エッジとみなした単純なモデルにおいて，ノードが保持範囲ノードに自身の変更を通知する方式を提案する．このとき，非同期的な通知の時間差や逆転により発生する管理グラフ情報の不整合に対処するため，管理範囲拡張時の再通知や，受信時の固有情報の判定によって整合性を確保した．次に，実際にはコンテンツは Web サーバに収容されていることを考慮したサーバモデルへと拡張し，同様の手法による通知方式を提案する．最後に，両モデルにおける通知数の差について簡単に議論する．
**キーワード**　Web とインターネット，メタデータの管理，協調型 DB，Web グラフ，協調型 Web アーキテクチャ

# A Synchronization Method of Web-Subgraphs of Web servers

Yukiyo TAKASAGO[†], Aki KOBAYASHI[††], Katsunori YAMAOKA[††], Yoshinori SAKAI[††], and Noboru SONEHARA[†††]

†,††Tokyo Institute of Technology　2–12–1 O-okayama Meguro-ku Tokyo, Japan, 152–8552
††† National Institute of Informatics　2–1–2 Hitotsubashi Tiyoda-ku Tokyo, Japan, 101–8430
E-mail: †takasago@net.ss.titech.ac.jp, ††{koba,yamaoka,ys}@ss.titech.ac.jp, †††sonehara@nii.ac.jp

**Abstract**　The Cooperative Web Architecture (CWA) has been proposed for relevant information retrieval using Web-Graph. In CWA, Web servers maintain and provide to users neighboring link information (Web-Subgraphs). When events that change the Web-Graph occur, the Web servers have to synchronize their Web-Subgraphs. In this paper, we first proposed the notification method for a simple model in which contents and hyperlinks are represented by nodes and edges. However, inconsistency problem is generated by time difference and reversal of notifications. We solved the problem by notifying old events when the management range was enhanced and judging direct notifications. Next, we enhanced the notification method to the server model considering that contents are accommodated by the Web server. At last, we discussed difference of the number of notifications in both models.
**Key words**　Web and Internet, Metadata Management, Cooperative DB, Web Graph, Cooperative Web Architecture

## 1. はじめに

HTML は他のコンテンツへのハイパーリンクを含んでおり，このリンクを辿っていくことで，ユーザは関連するコンテンツ間を移動することが可能である．このリンク情報はコンテンツをノードとし，ハイパーリンクを枝とする大きなグラフ空間（以下，Web グラフ）として考えることができる．リンク情報は意味ある情報として，Web グラフを利用する様々な研究が行われている．代表的なものとしては，被リンク数や，リンク数によりページをランキングする PageRank [1] や，関連する話題が密な Web グラフ構造になっていることを利用して Web コミュニティを発見する手法 [2]- [4] である．さらに，Web グラフにより形成される Web コミュニティを利用した検索ツール [5] や，Web グラフを可視化するツール [6]，Web グラフの構造を解明する研究 [7] なども行なわれている．

このように，Web グラフは様々な研究分野で利用されている有用な情報にも関わらず，簡単に入手することは困難とされている．そこで，簡単に入手するためにリンク情報を提供するサーバとして LINK データベース [8] が提案されている．しかし，これは Web 上の全てのリンク情報を 1 つのデータベース

で一括に保存しており，クローリングを行った後一括で更新を行う．そのため，頻繁なリンクの変化に対応できず，最新の情報を保持することが出来ない．A.Ntoulas らの調査 [9] によると，平均で毎週 25%の新しいリンクが作られていることがわかっている．このことからも，保持する Web グラフの頻繁な更新は必要不可欠だと考えられる．

Web グラフの応用を考えた場合，被リンクサイトの管理者が，サイト内容の変更や移動などをリンク元に通知したり，リンクでつながれたコンテンツ間には意味のつながりがあることが多いことを利用した近傍 Web グラフ空間内での検索が挙げられる．また，通常の閲覧時においても，正逆両方向の近傍リンク空間の様子を見ながらナビゲーションできれば，目的の情報を探し出したり，自分の閲覧履歴などを一目で理解できる．このように，多くの応用では，比較的近傍の Web グラフが得られればよい．また，Web リンク空間全体の Web グラフを構築しようとするとスケーラビリティの問題に突き当たるが，管理する情報を一部の近傍空間に限定することで，この問題を回避すると同時に，Web の超分散的な管理モデルにも合致させることができる．

そこで筆者らは，最新の Web グラフを Web サーバ側で維持，管理し，利用者や他サーバへ提供する仕組みとして，協調型 Web アーキテクチャ（Cooperative Web Architecture；以下，CWA）を提案している [10] [11]．CWA では，Web サーバは自身の保持するコンテンツの更新を確実に検知できる特性を利用して，Web サーバの付加機能としてサーバ内コンテンツの Web グラフを構築，管理すると同時に，各 Web サーバが近隣の最新の Web グラフ（以下，部分 Web グラフ）を保持することができるよう，近傍コンテンツの更新時に当該コンテンツを保持するサーバからの通知，およびリンク情報の提供を受ける仕組みを導入する．このような付加機能を導入した Web サーバのことを AWS（Advanced Web Server）と呼んでおり，アクセスした利用者に対してコンテンツだけでなく，部分 Web グラフ情報を提供することができる．リンクコンテンツ内での検索機能の提供や，周辺コンテンツを表示する Web ブラウジングなど様々な応用が期待される．

本研究では，CWA において，各 AWS が部分 Web グラフを一貫性のある状態に保つための同期方式を提案する．部分 Web グラフの同期方式としては主に，あるタイミングで情報を収集するポーリング方式と，Web グラフの変更時のみに，変更点を通知するイベント通知方式が考えられる．ポーリング方式は逆リンク情報を入手することが困難であるため，イベント通知方式を採用することとする．

イベント通知方式では，イベントの通知を hop by hop で行う方法と，すべての通知対象に直接通知を行う方法が考えられるが，以下の理由から直接通知を採用する．

- 本来 hop by hop は通信コストの低減のために利用されることが多いが，Web グラフはオーバレイネットワークであるため，論理的な通信コストは hop by hop と直接通知で等しい．
- 直接通知は，hop by hop よりも通知メッセージを速く伝播できる可能性がある．
- 直接通知を行うことで，途中ノードの負荷や障害の影響を受けない点で有利である．
- 既存 Web サーバ混在の環境でも，直接通知は有利である．

以上より，Web グラフの同期方式として，直接通知を利用したイベント通知方式を採用し，検討を進める．

まず，簡単のため，各コンテンツを独立のノードとして表現した，コンテンツモデルにおける同期方式を提案する．イベント通知方式として，Web グラフ変更時に，単純に変更を通知するのみでは，複数の変更が同時に発生した場合に不整合の問題がおこる．その不整合の原因を整理し，不整合を解消する通知方式を提案し，その整合性を確認する．次にその方式を，サーバの存在を考慮した方式へと拡張を行い，サーバモデルでの通知方式を提案する．

## 2. 協調型 Web アーキテクチャ

昨今，いわゆるブログ（WebLog）システムにトラックバックと呼ばれる機能が盛り込まれていることが多いが，これは手動で逆リンクを生成する仕組みであるとも言える．このように，一方的にリンクを設定する行為に対して，逆方向に辿るサービスへの欲求は強いものがあり，HTTP の reference ヘッダによる情報をサーバで収集し，クライアントへ逆リンク情報を提供する手法の提案 [12] などもなされている．CWA では,「リンクによって参照されている」という情報を直接各 Web サーバ間で通知しあい，保持することで，リンクの双方向での参照が可能であるだけでなく，更新にも追従することができるなど，リンクの機能が強化されている．この近傍のリンク情報（部分 Web グラフ）を収集，維持し，それらの情報をユーザに対して提供することが，CWA における基本機能である．

拡張機能として，CWA では，近傍検索やナビゲーション支援のための機能のうち，サーバ側に実装したほうが効率的であると考えられる機能についてもサーバ側に付加する．基本的な付加機能として，各 Web サーバが検索キーとメタ情報との距離計算機能を持つ．そのため，各 Web サーバが近傍コンテンツ内での検索を行うことができる．検索処理は，部分 Web グラフとメタ情報を得たクライアント側で行うこともできるし，サーバ側で検索処理を行いその結果のみをクライアントに応答することもできる [13]- [16]．それらの配分は，その時点で実行可能な処理や検索モデルによって定められる．

試験的な実装では，HTTP の GET，HEAD メソッドに対して，当該コンテンツのみでなく，部分 Web グラフやリンクコンテンツのメタ情報も応答する．ユーザは，ブラウザの支援を前提に，そのメタ情報を参考にしてリンクを選択することで Web 上を自在に探索することができる．Web グラフやメタ情報は，コンテンツの更新時に変化する．コンテンツの更新・削除には，例えば，HTTP の PUT・DELETE メソッドを用いることで，Web サーバがコンテンツの更新を確実に検知することができる．Web サーバは更新を検知すると更新されたコンテンツに対して，メタ情報抽出動作を行うと同時に，このコンテンツを含む部分 Web グラフを保持している他の Web サーバへ変更を通知する．通知を受けたサーバはその通知に従って自サーバの

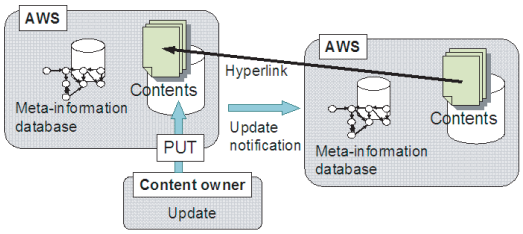


図 1　協調型 Web アーキテクチャ

保有する部分 Web グラフとメタ情報を更新する．それにより，協調型 Web システム全体でリンク情報やメタ情報の一貫性が保たれ，同期することができる．

しかし，コンテンツの更新がほぼ同時に発生した場合，通知の時間差により保持する部分 Web グラフ情報に不整合が生じることがある．本稿では，部分 Web グラフを収集，維持するという基本機能を実現するために必要な，整合性の保たれた同期方式を提案する．

## 3. コンテンツモデル

### 3.1 モデル化

実際のコンテンツは Web サーバに収容されているが，ここでは問題を簡単化するため，まず，コンテンツが独立に存在すると考え，リンク情報の変化をコンテンツ同士が通知するモデルを考える．その後 4 章でサーバを考慮したモデルに拡張する．

コンテンツをベースにした Web グラフの同期に関する問題のモデル化を行う．CWA では一定範囲の Web グラフ情報を一貫性のある状態に保つことが目的となる．Web グラフとは，Web コンテンツをそれぞれ独立のノードとして表現し，ハイパーリンクを有向リンクで表現した，有向グラフ空間である．この一定範囲内の Web グラフを部分 Web グラフと呼び，これはリンクの集合である．また，管理すべき Web グラフの範囲を管理範囲と呼ぶ．

ここで，$x, y, z$ をノードとしたとき，ノード間の論理的距離を定める距離関数 D は，距離の公理

（1） 非負性: $\mathrm{D}(x, y) \geq 0$

（2） 同一性: $\mathrm{D}(x, y) = 0 \Leftrightarrow x = y$

（3） 対称性: $\mathrm{D}(x, y) = \mathrm{D}(y, x)$

（4） 三角不等式： $\mathrm{D}(x, y) + \mathrm{D}(y, z) \geq \mathrm{D}(x, z)$

を満たすものとする．

また，リンクの向きを考慮しないリンクによる最短経路距離様の距離を用いることを念頭に，次のような条件を満たすものとする．

［条件 1］ $\mathrm{D}(x, y)$ が $\mathrm{link}(p \to q)$ の有無によって変化する場合、$\mathrm{link}(p \to q)$ が存在するとき，

$$\mathrm{D}(x, y) = \mathrm{D}(x, p) + \mathrm{D}(p, q) + \mathrm{D}(q, y).$$

と表される．

ただし，$\mathrm{link}(p \to q)$ は，ノード $p$ からノード $q$ へのリンクを表す．

すべてのノードは同じ距離関数 D を用いることとし，等しい距離 $d$ を管理範囲とすると仮定する．つまり，ノード $m$ の管理範囲内に存在するノードの集合 $[m]$ は次式のように表される．

$$[m] = \{\forall l \mid \mathrm{D}(m, l) \leqq d\}.$$

このとき，距離関数 D が対称性を満たし，かつ，全てのノードが等しい距離 $d$ を管理範囲とするため，以下の補題が成り立つ．

［補題 1］ ノード $x, y$ の管理範囲内に存在するノードの集合を $[x], [y]$ と表すとき，

$$x \in [y] \Longleftrightarrow y \in [x],\ x \notin [y] \Longleftrightarrow y \notin [x].$$

そのため，以下の定理が成り立つ．

［定理 1］ 整合性保持のための通知は，管理範囲内のノードに行われることが必要十分条件である．

また，前提条件と，全てのノードが等しい距離 $d$ を管理範囲とすることから，以下の補題が成り立つ．

［補題 2］ ノード $x, y$ の間に新しくリンクが追加され，$[x]$ と $[y]$ が接続した場合，

$$\forall v \in [x] \text{ について，}\ [v]^* \subseteqq [v] \cup [y].$$

$$\forall u \in [y] \text{ について，}\ [u]^* \subseteqq [u] \cup [x].$$

ただし，$[x]$ は，リンク追加前におけるノード $x$ の管理範囲内に存在するノード集合，$[x]^*$ は，リンク追加後におけるノード $x$ の管理範囲内に存在するノード集合を表す．

さらに，定理 1 と補題 2 から以下の定理が成り立つ．

［定理 2］ 複数の Web グラフが接続した場合，それらの Web グラフ内で発生した変更に関する通知は，接続された Web グラフの合併集合内のノードに行われれば十分である．

Web グラフ空間の変更を引き起こすものをイベントと呼ぶとき，イベントは，リンクの追加とリンクの削除のみを考えれば十分である．他にノードの追加・ノードの削除が考えられるが，これらはリンクの追加や削除の集合として捉えることができるため考慮しなくてよい．図 2 のように，イベントにより，追加もしくは削除されたリンクの出力元であるノードをイベントノードと呼び，リンクの追加により，新しく隣接となるノードを新隣接ノードと呼ぶ．

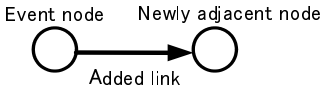


図 2　ノードの名称

### 3.2 単純方式

#### 3.2.1 単純方式の動作

1 章で述べたように，部分 Web グラフの同期方式として，直接通知を利用したイベント通知方式を採用する．ここでは，その中でも単純方式の動作を説明する．

基本的にイベントノードが，実際に保持している範囲（保持範囲）内の全ノードにイベントを通知する．しかし，リンクの追加イベントが発生した場合，新隣接ノードから繋がる Web グラフ情報を知る必要がある．また逆に，新隣接ノードは追加イベントノードから繋がる Web グラフ情報を知る必要がある．以下に詳しい動作を，イベントごとに説明する．通知に必要なメッセージの種類は 4 つのタイプに区別することができる．そ

表 1　通知の種類

| タイプ | 内容 |
|---|---|
| 1 | リンクの削除通知 |
| 2 | イベントノードから新隣接ノードへのリンクの追加通知 |
| 3 | 新隣接ノードからイベントノードへのリンクの追加通知 |
| 4 | タイプ 2，3 以外のリンクの追加通知 |

表 2　表 記 一 覧

| 表記 | 意味 |
|---|---|
| $a$ | ノード $a$ |
| $A$ | Web グラフ $A$ |
| link($a \to b$) | $a$ から $b$ へのリンク |
| $\langle a \rangle$ | $a$ の保持する Web グラフ |
| add: link($a \to b$) | $a$ から $b$ へのリンク追加イベント |
| rm: link($a \to b$) | $a$ から $b$ へのリンク削除イベント |
| notify[2] : $c \to d; \langle a \rangle$ ; | $c$ から $d$ へ $\langle a \rangle$ を通知する (タイプ 2) |

れらの通知タイプを表 1 にまとめる．また，ノードの動作の説明に用いる表記の意味は，それぞれ表 2 の通りである．

**3.2.2**　リンク追加イベント

リンクの追加イベントノードと新隣接ノードは，互いに保有している部分 Web グラフ情報を交換する．その後，この 2 つのノードから保持範囲の全ノードに，交換した部分 Web グラフ情報を付加して，リンクの追加イベントを通知する．

各ノードの動作は以下のようになる．

**add:link**($x \to y$) **が発生**

- イベントノード:$x$
  1. notify[2] : $x \to y; \langle x \rangle$ ,link($x \to y$);
  2. 通知タイプ [3] の $\langle y \rangle$ を受信したら，
  2.1 notify[4] : $x \to$nodes($\langle x \rangle$); $\langle y \rangle$ ,link($x \to y$);
  2.2 $\langle x \rangle = \langle x \rangle + \langle y \rangle +$link($x \to y$)
- 新隣接ノード:$y$
  1. 通知タイプ [2] の $\langle x \rangle +$link($x \to y$) を受信したら，
  1.1 notify[3] : $y \to x; \langle y \rangle$ ;
  1.2 notify[4] : $y \to$nodes($\langle y \rangle$); $\langle x \rangle$ ,link($x \to y$);
  1.3 $\langle y \rangle = \langle y \rangle + \langle x \rangle +$link($x \to y$)
- 受信ノード:$\forall v \in$nodes($\langle x \rangle$)

  通知タイプ [4] の $\langle y \rangle$ ,link($x \to y$) を受信したら，

  $\langle v \rangle = \langle v \rangle + \langle y \rangle +$link($x \to y$)
- 受信ノード:$\forall u \in$nodes($\langle y \rangle$)
  1. 通知タイプ [4] の $\langle x \rangle$ ,link($x \to y$) を受信したら，

  $\langle u \rangle = \langle u \rangle + \langle x \rangle +$link($x \to y$)

**3.2.3**　リンク削除イベント

リンクの削除イベントノードが，保持範囲内の全ノードにリンクの削除イベントを通知する．各ノードの動作は以下のようになる．

**rm:link**($x \to y$) **が発生**

- イベントノード:$x$
  1. $\langle x \rangle = \langle x \rangle -$link($x \to y$)
  2. notify[1] : $x \to$nodes($\langle x \rangle$);link($x \to y$);
- 受信ノード:$\forall z \in$nodes($\langle x \rangle$)

  通知タイプ [1] の link($x \to y$) を受信したら，

  $\langle z \rangle = \langle z \rangle -$link($x \to y$)

**3.2.4**　通知受信時の部分 Web グラフの更新

通知を受信したノードは，保持する部分 Web グラフを受信した通知をもとに更新するが，この時受信した情報のうち管理範囲内の情報のみを入手し，管理範囲外の情報を破棄する．

### 3.3　単純方式の問題点

単純方式のみを用いる更新では，保持する部分 Web グラフが実際の Web グラフと異なってくる場合がある．この不整合問題を解決するため，まず原因について整理する．

**3.3.1**　部分 Web グラフの不整合

イベントが発生してから，通知を受信するまでには，処理や伝播にある程度の時間がかかる．この通知を受信するまでの間，通知受信前のノードが保持する部分 Web グラフは，一時的に実際の Web グラフと矛盾している状態となる．この一時的な矛盾が生じている間に，他のイベントが発生し，矛盾の生じているノードが通知を行うと，矛盾が他のノードに伝播し，その矛盾は恒久的な状態として保持され，いわば矛盾が確定する．以下では，部分 Web グラフの矛盾が確定した状態のことを部分 Web グラフの不整合と呼ぶ．

つまり，部分 Web グラフの不整合は，同時期に異なるイベントが複数発生することで生じる．この不整合は，通知の範囲に起因する問題と，通知の順序に起因する問題に区別されるため，以下ではこれらに区別して，不整合の原因を詳しく説明する．

**3.3.2**　範囲に起因する問題

- 通知範囲の不十分問題

イベント発生時に，イベントノードは保持範囲内の全ノードに対して通知を直接行う．しかし，この保持範囲は，実際にノードが保持している範囲であり，常に実際の Web グラフに対して正確な範囲を保持しているわけではない．定理 1 より，通知すべき範囲は，本来管理すべき管理範囲内のノードであるため，保持範囲と管理範囲が一時的に矛盾している場合に通知を行うと，通知範囲が不十分となることがあり，結果として不整合が生じる．

具体例として，図 3 のように，add:link($g \to d$) とほぼ同時に rm:link($c \to e$) が発生した場合を取り上げる．rm:link($c \to e$) が発生したことで，Web グラフ $X$ は $X'$ に変化し，イベント発生後に，各ノードが管理すべき範囲は $X' + Y$ となる．不整合が生じる原因を表しているのが図 4 である．各矢印が，通知を表しており，矢印の上段が通知のタイプ，下段が通知内容を表している．ノード $c$ が追加通知を受ける前に，rm:link($c \to e$) が発生しているため，削除イベントの通知は Web グラフ $X$ 内のノードのみに行われ，Web グラフ $Y$ 内のノードには通知されない．そのため，Web グラフ $Y$ 内のノードは古い Web グラフ $X$ を持ち続ける事になり，実際のリンク情報との不整合が生じる．

**3.3.3**　順序に起因する問題

複数の通知を受信する場合，受信する順序により不整合が生じることがある．順序が入れ替わることによる不整合には，伝

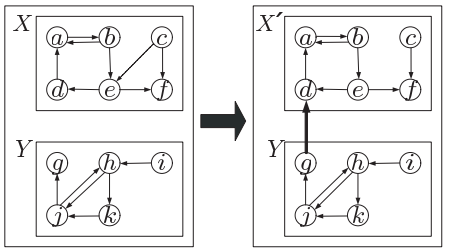


図 3　イベントの同時発生

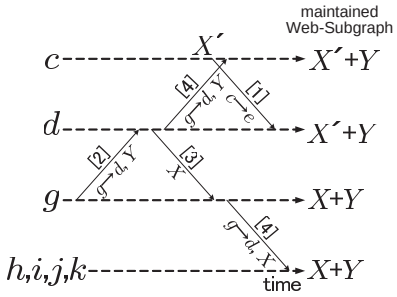


図 4　通知範囲の不十分問題

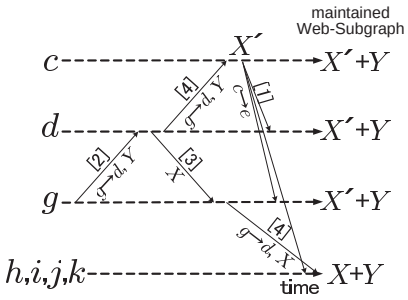


図 5　伝聞情報による上書き問題

聞情報による上書き問題と，通知の破棄問題がある．

- 伝聞情報による上書き問題

通知する内容には，固有情報と伝聞情報が含まれる．自ノードから，順方向に一段先までのリンク情報のことを固有情報と呼び，固有情報以外のリンク情報を伝聞情報と呼ぶ．伝聞情報は，自ノード以外の情報であるため，実際の Web グラフと矛盾している可能性がある．通知方式では，追加通知において，伝聞情報を通知する．そのため，通知の逆転により，後から通知される誤った伝聞情報により上書きされ，不整合が生じることがある．

具体例として，同様に図 3 のように add:link($g \to d$) とほぼ同時に rm:link($c \to e$) が発生した場合を取り上げる．伝聞情報の上書き問題が生じる原因を表しているのが図 5 である．この例ではノード $c$ は，保持範囲 $X + Y$ 内のノードに削除イベントを通知している．しかし，ノード $h, i, j, k$ は，削除通知の後にノード $g$ からの伝聞情報を含むタイプ 4 の追加通知を受信している．この追加通知は，Web グラフ $X$ を通知しており，この $X$ は削除された link($c \to e$) を含んでいる．そのため，rm:link($c \to e$) の情報を上書きされ，Web グラフ $X$ を持ち続けることとなり，実際のリンク情報との不整合が生じる．

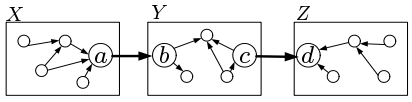


図 6　追加イベントの同時発生

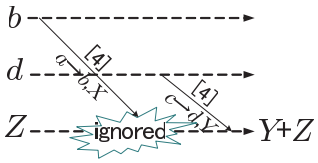


図 7　通知の破棄問題

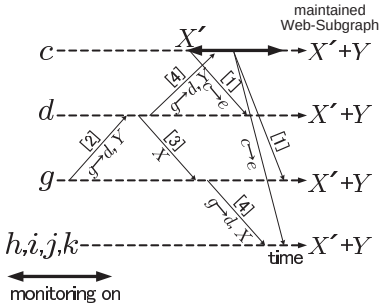


図 8　範囲に起因する問題の解決

- 通知の破棄問題

部分 Web グラフを更新する際，受信ノードは，通知受信時に管理範囲外である Web グラフ情報は破棄する．そのため，実際には管理範囲内の Web グラフ情報を，通知受信時に管理範囲外と判定し，破棄してしまう場合がある．

図 6 のように add:link($a \to b$) とほぼ同時に add:link($c \to d$) が発生し，Web グラフ $X$，$Y$，$Z$ が連結したとき，通知の破棄問題が生じる場合のシーケンス図が図 7 である．Web グラフ $Z$ 内のノードは，ノード $b$ から link($a \to b$) と Web グラフ $X$ の情報を，ノード $d$ から link($c \to d$) と Web グラフ $Y$ の情報を通知される．この順序が，図 7 のように，ノード $b$ からの通知を先に受信すると，受信時にはまだ Web グラフ $Y$ の情報を保持していないために，Web グラフ $X$ を管理範囲外として破棄してしまい，実際のリンク情報との不整合が生じる．

### 3.4 提案方式

各イベント＝トランザクションとして，トランザクションを分離する等の不整合を避ける方法が考えられる．しかし，ロックなどを使う方式は Web には適さないため用いない．提案方式では，単純方式に以下の動作を追加する．

#### 3.4.1 範囲に起因する問題の解決

イベント発生後，時間 $T$ の間，イベントノードと新隣接ノードは監視モードに入る．監視モード時に追加ノードの通知を受けたら，それらの追加ノードのうち，管理範囲内のノードであり，かつ未通知のノードに対して過去に発生したイベントを通知する（図 8）．

#### 3.4.2 順序に起因する問題の解決

受信した通知を通知管理時間 $S$ の間管理しておき，通知受信時には，管理している過去の通知と，受信した通知をもとに部

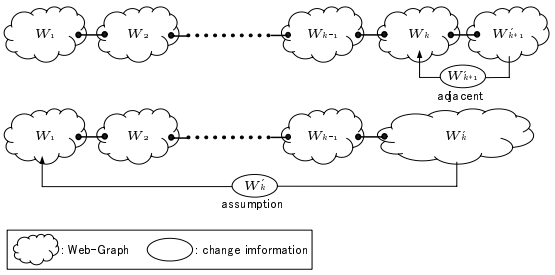


図 9　多連結する部分 Web グラフでの伝播

分 Web グラフを更新する．それらの通知間で矛盾が生じた場合は固有情報による通知を採用する．

### 3.5　提案方式の整合性

提案方式の整合性が確保されることを示す．不整合はイベント同時発生時に生じるため，イベント同時発生時に通知が全ノードに行われ，かつ，通知の順序が入れ替わっても整合性が確保されることが示されれば，提案方式の整合性を示すことができる．そこで，通知範囲の正当性と通知順序への依存性について議論する．

#### 3.5.1　通知範囲の正当性

本来通知すべき管理範囲の全ノードに通知が行われるかを考える．定理 2 より，複数の Web グラフが接続した場合，通知が Web グラフの合併集合内のノードに行われれば充分であった．そこで，複数の Web グラフが接続した場合に，通知が Web グラフの合併集合内の全ノードに伝播するかを考える．

（1） 隣接する部分 Web グラフでの伝播

追加イベントにより隣接となった部分 Web グラフ間で，変更が必ず伝播されるかを考える．1 つの追加イベントにより二つの部分 Web グラフ $W_1$ と $W_2$ が連結される場合を想定する．部分 Web グラフ $W_1$ 内でイベントが発生し，$W_1'$ となった場合に $W_2$ に $W_1'$ が伝播されればよい．

$W_1$ を $W_1'$ に変化させたイベントに関わる，イベントノードもしくは新隣接ノードは $W_1$ 内に存在する．また，$W_2$ の情報は必ず $W_1$ に通知される．よって，$W_2$ の情報を通知された，監視モード中のイベントノードもしくは新隣接ノードは，$W_2$ に対して $W_1'$ を通知する．以上より，隣接する部分 Web グラフに関する変更は伝播されるといえる．

（2） 多連結する部分 Web グラフでの伝播

次に多数の追加イベントにより，複数の部分 Web グラフが連結される場合について考える．隣接する部分 Web グラフに関する変更は伝播されることを前提とする．リンクの追加イベントが $k-1$ 個同時発生し，部分 Web グラフ $W_1$ から $W_k$ が連結した時，$W_k$ の変更 $W_{k'}$ は $W_1$ まで伝播すると仮定する．$W_{k+1}$ が $W_k$ にさらに同時に連結すると，$W_{k+1}$ の変更 $W_{k+1}'$ は隣接である $W_k$ には伝播する．それにより，$W_k$ は $W_k'$（$= W_k + W_{k+1}'$）となる．この変更は仮定により $W_1$ に伝播することになる．つまり，$W_{k+1}'$ は $W_1$ まで伝播することとなる．以上より，帰納的に，複数の部分 Web グラフが連結する場合にも，変更が伝播されるといえる．

このように，追加イベントが複数同時発生した場合も，Web グラフの変更を正しく伝播し，イベントは管理範囲内の全ノードに通知される．

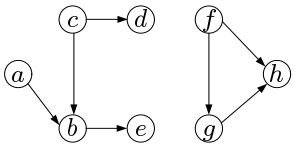


図 10　初期トポロジ

#### 3.5.2　通知順序への依存性

次に，整合性が通知を受信する順序に依存しないことを示す．通知管理時間 $S$ 以内に受信したすべての通知を元に，部分 Web グラフを更新するため，順序は無関係であり，問題は $S$ 以内に受信した通知間において矛盾が発生した場合である．矛盾が発生した場合に固有情報による通知を採用することが正当であるかを考える．

リンクの追加は，固有情報として通知される場合と，伝聞情報として通知される場合があるが，リンクの削除は必ず固有情報として通知されるため，1 つのリンクに関して矛盾する通知の組み合わせは，以下の 2 通りのみである．

（1） 追加の固有情報と削除の固有情報

（2） 追加の伝聞情報と削除の固有情報

固有情報は情報が正しいことが保証されている．よって，(1) のように共に固有情報として通知されたにも関わらず，通知間に矛盾が発生した場合は，後から受信した情報を採用する．この状況は，あるノードが短時間で追加と削除を行った場合であり，そのイベントノードから追加と削除の 2 種類の通知を受信したことを意味する．同ノードからの通知の順序が逆転することはないと考えることが自然である．

(2) のような状況で，追加を採用することは，あるイベントノードが削除通知を行った保持範囲内のノードが，直後に同じリンクを追加した時には保持範囲外になりイベントノードから通知されないことを意味する．さらに，他のノードから受信した追加通知に含まれる伝聞情報を採用することを意味する．イベントノードにとって保持範囲外であるということは，矛盾を受信したノードにとって，イベントノードは同様に保持範囲外となるため，議論のリンクも保持範囲外となる．それにも関わらず，範囲内として伝聞情報を採用することは矛盾している．短時間で削除されたリンクが，追加される場合，(1) のように通知はどちらも固有情報として通知される．そのため，(2) のような場合，削除の固有情報を採用することは正しいと言える．

以上より，(1) のように同一ノードからの通知の順序は入れ替わらないことを前提とした場合に，通知管理時間 $S$ 以内においては，通知の順序が入れ替わっても整合性は確保される．

### 3.6　シミュレーション

シミュレーションにより，単純方式での不整合の発生具合と，提案方式により不整合が生じないことを実験的に示す．初期トポロジは図 10 であり，シミュレーションの条件を表 3 の条件 1 にまとめる．また，距離関数 D として，リンクの方向を無視し，各リンクの距離を 1 とした場合の最短経路距離を求める

表 3　シミュレーション条件

| | 条件 1 | 条件 2 |
|---|---|---|
| ノード数 | 8 | 20 |
| 初期リンク数 | 7 | 21 |
| 初期トポロジの分割数 | 2 | 4 |
| 管理範囲 | 7 段以内 | 7 段以内 |
| 発生追加イベント | add:link($e \to g$), add:$link$($d \to f$) | ランダムに 5 個 |
| 発生削除イベント | rm:link($b \to c$) | ランダムに 5 個 |
| 監視モード時間 $T$ | 無限 | 無限 |
| 通知管理時間 $S$ | 無限 | 無限 |
| 実行回数 | 1000 回 | 1000 回 |

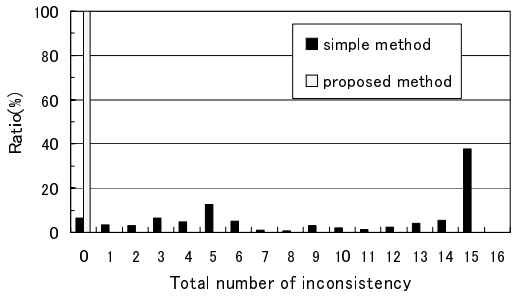


図 11　シミュレーション結果

関数を用いた．不整合が生じるイベントの同時発生時においてシミュレーションを行うため，監視モード時間やイベント管理時間を無限とした．イベントにより発生する通知をランダムに実行した．一回の実行結果として，各ノードで最終的に保持している部分 Web グラフの矛盾しているリンクの本数の合計が，不整合の合計数として出力される．結果が図 11 のグラフである．横軸に不整合の合計をとり，その時の実行回数の割合を縦軸にとる．単純方式では，不整合の発生しない場合が 10% に満たなかったのに対し，提案方式では全ての場合において不整合が生じなかった．

次にイベントをランダム生成した場合のシミュレーションを行った．シミュレーション条件は，表 3 の条件 2 の通りである．リンクの追加とリンクの削除をそれぞれ 5 個ずつランダム生成し，10 個のイベントを同時発生させ，同様に通知をランダムに実行した．その結果，すべての実行回において提案方式では不整合が発生しなかった．

以上より，提案方式により不整合が生じないことを実験的に示した．

## 4.　サーバモデルへの拡張

### 4.1　サーバモデル

3 章では，コンテンツモデルでの通知方式を提案し，整合性が確保されることを示した．実際のコンテンツはサーバ内に複数存在しているため，この事実を利用するサーバモデルに拡張を行うことで，実装に近い通知モデルとなる．コンテンツモデルの整合性は確保されたため，コンテンツモデルを順に拡張することでサーバモデルを提案する．

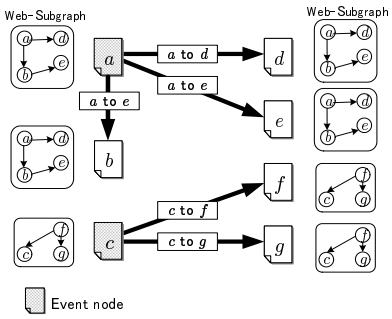


図 12　ノード間通知

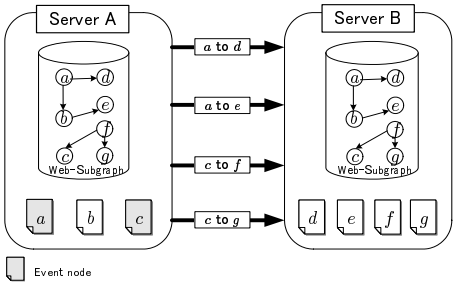


図 13　サーバ間通知

サーバモデルでは，サーバがそれぞれ独立に存在し，その中に複数のノード（Web コンテンツ）が存在しているものとする．また，各サーバは一定範囲内の Web グラフ情報を一貫性のある状態に保つことが目的となるが，この時の範囲は，サーバ内の全ノードの管理範囲の和集合となる．

### 4.2　サーバ間でのイベント通知

まず，イベント発生時の通知をサーバ間で行うように拡張を行う．コンテンツモデルにおいて，(ノード → ノード）で送受信した通知メッセージを（図 12），サーバモデルでは（ノードを保有するサーバ → ノードを保有するサーバ），と変更して送受信ことで，サーバ間で通知を行う（図 13）．ただし，通知元ノードと通知先ノードが同じサーバ内に存在している場合は，通知を行わなわず，サーバで内部処理するものとする．

変更点は通知メッセージの送受信をサーバが行う点のみで，通知メッセージ内容はコンテンツモデルと同様であるため，整合性が保たれることは明らかである．

### 4.3　サーバ集約

サーバ間で通知を行うことにより，同サーバ内のコンテンツに関する通知を集約することができる．サーバ集約は，通知先のノードを集約する着サーバ集約と，通知元のノードを集約する発サーバ集約に分類することができる．

#### 4.3.1　着サーバ集約

サーバは，１つのイベントが発生すると，イベントノードの保持範囲内の全ノードに対して，直接通知を行うが，このときの通知先のノードが，同一のサーバに存在する場合がある．このとき，同一サーバ内のノードに対する通知を一括して行う．通知先が集約されるので，１つの通知メッセージにおいて，通知先が複数のノードとなるが，通知するイベントは１つである．

#### 4.3.2　発サーバ集約

コンテンツ制作において，複数のページを同時に更新するこ

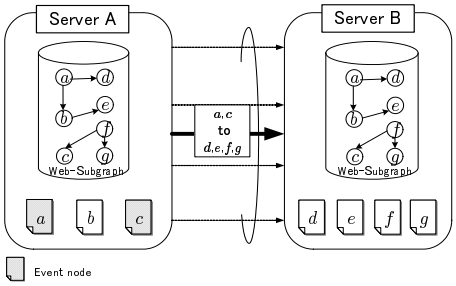


図 14　発着サーバ集約

とが一般的である．そのため，同サーバ内の複数のノードが，連続的にイベントを発生する可能性は高いと考えられる．そこで，同時に発生する複数のノードからの通知を一つの通知メッセージに集約して通知を行う．通知元が集約されるので，1つの通知メッセージにおいて通知元が複数のノードとなり，通知するイベントも複数となる．

現実的には，処理待ちキュー内のイベント処理を，一旦，通知メッセージの構築でバッファリングし，実際の通知はイベントキューが空になってから行うことで，同じコンテンツへの通知を集約する．また，管理者が更新ボタンを押すといった，意味的に人が更新を定義することで，さらに長時間での発サーバ集約を行うことも考えられる．

**4.3.3**　発着サーバ集約と整合性

着サーバ集約と発サーバ集約を同時に行うことで，短時間でのサーバ間の通知を一回で行う (図 14)．

コンテンツモデルにおける複数の通知が，発着サーバ集約により1つの通知メッセージにまとめられるが，それぞれの通知の通知元・通知内容・通知先は変化しないため，通知漏れといった通知範囲の問題は発生しない．通知順序の問題では，同一ノード発の通知について保存されているか，保存されたのに等しい最終的な情報が通知されれば良く，これは同一サーバ内における処理であり集約時に，容易に実現できる．以上より，発着サーバ集約を行うサーバモデルでの通知方式は整合性が確保されるといえる．

### 4.4　通知数の比較

コンテンツモデルでは，イベント発生時の理論的な通知数は，管理範囲内のノード数 $N$ に等しい．リンクコンテンツが自サーバ内である確率を $P_{own}$ とすると，自サーバ内コンテンツに通知する必要のないサーバモデルでは，通知数が $N(1-P_{own})$ となる．さらに，リンクコンテンツが $k$ 個ある場合に，$k$ 個目のコンテンツが $k-1$ 個までのコンテンツと異なるサーバに存在する確率を $p_k$ とすると，着サーバ集約を行うことで，理論的に通知数は $\sum_{k=1}^{N(1-P_{own})} p_k$ に集約されると考えられる．

## 5.　ま　と　め

本稿では，CWA を実現するために必要不可欠な，部分 Web グラフ間での同期を保つための通知方式を提案した．問題を単純化したコンテンツモデルにおいて，通知の範囲や順序に起因する問題を解決する方式を提案し，整合性を確認した．次に，サーバの存在を考慮したサーバモデルへと拡張を行い．サーバモデルにおける通知量の簡単な検討を行った．

本稿では，管理範囲を定める論理距離が距離の公式を満たすこと，最短経路距離様の距離を用いることを念頭においた条件の 2 点のみを仮定し，管理範囲の決定法には言及しなかった．実際の Web 空間では，人気コンテンツが存在し，それらのコンテンツは多数の着リンクをもつ．このような，着リンク多数のノードが存在した場合，そのノードを管理範囲内とするノードは，それら全てのリンクを管理する必要がある．しかし，実際のサーバには能力の差が存在し，これらのリンクを管理することが出来ないサーバも存在する．サーバ能力も考慮した管理範囲の適切な決定法は今後の課題である．

## 文　　献


[1] L. Page:“PageRank: Bringing order to the Web”, Stanford Digital Digital Libraries working paper, 1997.

[2] Jeffrey Dean, and Monika R. Henzinger:“Finding related pages in the World Wide Web”, In Proc. of 8th WWW Conference, pp.1467-1479, 1999.

[3] R.Kumar, P.Raghavan, S.Rajagopalan, and A.Tomkins: “Trawling the web for emerging cyber-communities”, In Proc. of 8th WWW Conference, pp.1481-1493 1999.

[4] 村田 剛志: “参照の共起性に基づく Web コミュニティの発見”, 人工知能学会論文誌 Vol.16 No.3, pp.316-323, 2001.

[5] 久我昌崇, 中所武司: “Web コミュニティの知識に基づく情報検索手法の評価”, 情報処理学会研究報告 DBS123-6, IPSJ, 2001.

[6] Nahum Gershon:“Moving happily through the world wide web”, In *IEEE Computer Graphics and Applications*, pp. 72–75, 1996.

[7] Andrei Broder, Ravi Kumar, and Marzin Maghoul:“Graph structure in the web”, In Proc. of 9th WWW Conference, pp.309–320, 2000.

[8] K.Randall, R.Stata, R.Wickremesinghe, and J.Wiener:“The LINK database: Fast access to graphs of the Web”, Research Report 175, Compaq Systems Research Center, 2001.

[9] A.Ntoulas, J.Cho, and C.Olston:“What’s New on the Web? The Evolution of the Web from a Search Engine Perspective”, In Proc. of 13th WWW Conference, pp.1-12, 2004.

[10] Aki Kobayashi, Katsunori Yamaoka, Yoshinori Sakai: “Cooperative Web Architecture for Search and Navigation Ass istance”, In Proc. of ICWI2002 IADIS WWW/Internt 2002, pp.726-729, 2002.

[11] Aki Kobayashi, Kuangmin Tan, Katsunori Yamaoka, Yoshinori Sakai: “Relevant information retrieval for cooperative Web architecture”, In Proc. of ICWI2004 IADIS WWW/Internet 2004, pp.1125-1128, 2004.

[12] Soumen Chakrabarti, David A. Gibson, Kevin S. McCurley: “Surfing the Web Backwards”, In Proc. of 8th WWW Conference, pp.1679-1693, 1999.

[13] 小林 亜樹，新名 崇，内藤 清一郎，酒井 善則，山岡 克式：“PIRCS: リンク情報の自律的収集による Web 情報探索”，電子情報通信学会技術報告，SSE2000-116， pp.7–12, 2000.

[14] 樋山 大輔, 内藤 清一郎, 小林 亜樹, 山岡 克式, 酒井 善則：“超分散型 Web 検索システム PIRCS の試作”, 電子情報通信学会技術報告，SSE2000-239 IN2000-195，pp.25–32, 2001.

[15] Aki Kobayashi, Hidetomo Miyahara, Kaito Tochihara, Hengjiang Wang, Katsunori Yamaoka, Yoshinori Sakai: “PIRCS: A Link Context Based Search on The Web,” In Proc. of IEEE PACRIM’03 S11-1, pp514-517, 2003.

[16] Hidetomo Miyahara, Aki Kobayashi, Katsunori Yamaoka, Yoshinori Sakai: “Visualization of Web Link Space for Neighboring Search”, In Proc. of ICWI2004 IADIS WWW/Internet 2004, pp.1135-1138, 2004.