一切合財RSSリーダ: RSSリーダへのWebリンク構造の融合

高橋　庸介† 　小林　亜樹†† 　山岡　克式††† 　曽根原　登††††

† 東京工業大学 情報工学科　〒 152–8552 目黒区大岡山 2–12–1
†† メディア教育開発センター　〒 261–0014 千葉市美浜区若葉 2–12
††† 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: †yosuke@net.ss.titech.ac.jp, ††aki@nime.ac.jp, †††yamaoka@ss.titech.ac.jp, ††††sonehara@nii.ac.jp

あらまし　ユーザが自分の興味を持ったコンテンツだけでなく，それに関連するコンテンツのRSSフィードも取得するにはRSSリーダへの登録に手間がかかる．またRSSリーダが，更新の有無を確認するだけのためにRSSフィード配信サーバに無駄な通信処理をさせる問題も指摘されている．本研究ではこの問題に対し，筆者らが提案している協調型Webアーキテクチャを用いたRSSフィード配信方式を提案する．本提案方式では，リンクで連結されたWebサイト間で更新情報の共有が，効率的な情報交換によって行われ，ユーザは興味のある1サイトのみをRSSリーダに設定すれば，リンクによって関連付けられたWebサイトのRSSフィードを一切合財取得することができる．また，RSSリーダにはリンク構造を基にRSSフィードを配置して表示する機能や，ユーザが無関心なRSSフィードの表示省略を簡単に行える機能などを持たせることで，大量のRSSフィードを整理してユーザに提供することができる．提案方式を試作システムとして実装し，その効果や問題点について事例を通じて考察する．

キーワード　RSS，RSSリーダ，Webグラフ，協調型Webアーキテクチャ

Issai-Gassai RSS Reader: RSS Reader amalgamated with link structures of the Web

Yosuke TAKAHASHI†, Aki KOBAYASHI††, Katsunori YAMAOKA†††, and Noboru SONEHARA††††

† Tokyo Institute of Technology　2–12–1 O-okayama Meguro-ku Tokyo, Japan, 152–8552
†† National Institute of Multimedia Education　2–12 Wakaba Mihama-ku Chiba, Japan, 261–0014
††† Tokyo Institute of Technology　2–12–1 O-okayama Meguro-ku Tokyo, Japan, 152–8552
†††† National Institute of Informatics　2–1–2 Hitotsubashi Chiyoda-ku Tokyo, Japan, 101–8430
E-mail: †yosuke@net.ss.titech.ac.jp, ††aki@nime.ac.jp, †††yamaoka@ss.titech.ac.jp, ††††sonehara@nii.ac.jp

Abstract　Making catalog of RSS feeds to RSS reader is a time and effort consuming task for users who want to obtain RSS feeds of contents they are interested in, as well as the ones relevant to their interests. Additionally the problem, that RSS readers cause unnecessary extra loads at feed distributing servers while checking updates, has been pointed out. In order to solve these problems, here we will propose a novel method to deliver RSS feeds using previously proposed Cooperative Web Architecture. In this new method, linked websites share their renewal information by means of efficient information exchange among them. Users registered to one website of their interest can easily obtain all RSS feeds of the associated websites. Furthermore, RSS reader can be arranged to have features like providing RSS feeds based on link structures or hiding specific feeds to users with no interest over them and thus, the rearrangment and mass distribution of RSS feeds can be made possible. Here, we will also introduce an implemented trial system based on the proposed method as well as its effects and problems using some case examples.

Key words　RSS, RSS Reader, Web Graph, Cooperative Web Architecture

1. 背　　景

Web 上の情報は日々著しい勢いで増大しており，ユーザが自分の嗜好にあった情報を入手することが困難になっている．このため，ユーザが膨大な情報を効率よく閲覧することを支援するためのツールとして RSS リーダが普及している．RSS リーダは Web サイトが発行する RSS 形式で書かれたデータ (RSS フィード) の取得を行い，ユーザが見やすい表示形態に変換して提供する [1]．RSS フィードには Web サイトの概要や更新情報などを記載することができる．これによりユーザはコンテンツの内容を把握し，興味を持ったものだけを実際に閲覧するということが可能になった．

現在の RSS リーダはユーザが登録した Web サイトの RSS フィードのみを取得する．RSS リーダへの登録は Web サイトのテーマとユーザの嗜好が一致したときに行われると考えられるため，Web サイトが発信する情報はほぼユーザの嗜好に沿っていると言える．しかし登録された Web サイトからの情報だけで，ユーザが満足する情報が得られるとは限らない．例えば，Google [2] における関連ページは検索にヒットしたコンテンツだけでなく，そのコンテンツに関連する情報も同時に提供している．このように単一の情報のみではなくそれと関連性を持つ情報を同時にユーザに提供することは，ユーザの要求に答えるために必要な機能であると言える．

Web における関連情報として、ハイパーリンクによって連結されている Web サイト群が考えられる．ハイパーリンクはコンテンツ間に関連性があることを示すものと言える．よってユーザが興味を持ったコンテンツのリンク先もまた，ユーザが興味を持つ内容である可能性は高い．ハイパーリンクがコンテンツ間の関連を表しているとして，意味的に類似したコンテンツ群 (Web コミュニティ) を Web リンク構造から抽出する研究は近年多く行なわれている [3]- [8]．これらが示すように，関連コンテンツを発見するためにはハイパーリンクが大きな役割を果たしている．ゆえに関連情報の収集にこれらを利用することは有効なことであると言える．また，リンクによる関連性はユーザの嗜好と合致しなくとも，別の側面からの視点としてユーザが興味を持つ可能性もある．つまり，ユーザが RSS フィードを要求したコンテンツだけでなく，ハイパーリンクによって関連付けられている他のコンテンツの RSS フィードもユーザにとって有益なものとなり得る．

これらのことを基に，ユーザが RSS フィードを要求したコンテンツ (中心コンテンツ) からハイパーリンクにより数ホップで到達できるコンテンツ (近傍コンテンツ) の RSS フィードを，中心コンテンツの関連情報として取得するサービスが考えられる．しかし従来の RSS フィード配信方式で中心コンテンツと近傍コンテンツの RSS フィードを取得するには，RSS リーダに多くの Web サイトを登録する必要があるが，これには問題がある．1 つはユーザの RSS リーダへの登録手間問題で，ユーザは複数の RSS フィードを取得するために，ハイパーリンクを辿っては RSS リーダに Web サイトを登録するという作業を繰り返す必要がある．これはユーザにとって非常に負担となる．もう 1 つはサーバのアクセス処理問題で，ユーザが要求する RSS フィードが増加する分 RSS フィード発信サーバへのアクセスが増加し，サーバへの負担が大きくなる恐れがある [9]．もともと一定時間置きにアクセスを行なう機能 (polling) を持つ RSS リーダが与えるサーバへの負荷は無視できるものではなく，実際にサーバがダウンし，RSS フィードの内容を縮小せざるを得なかった例もあるため [10]，現在の RSS フィード配信方法で先ほど述べたサービスを行なうことは好ましくない．

本稿では筆者らの研究室で提案している協調型 Web アーキテクチャ [11] を用いて，上記の問題を解決しつつサービスを実現する RSS フィード配信方式を提案する．またこの方式に対応した機能を持つ RSS リーダについても提案する．

また本提案方式の実装を行ない，提案方式の有効性，拡張性について議論する．

2. 提案 RSS フィード配信モデル

2.1 概　　要

本章では，背景で述べたサービスを可能とし，かつ問題点を解決する RSS フィード配信モデルを提案する．

RSS フィードは RSS 形式のファイルとして，他のコンテンツ同様 Web 上に独立に存在しているが，本来これはコンテンツの情報を示すものであるためコンテンツと一体で管理されるべきである．本稿では RSS フィードをコンテンツのメタデータの 1 つとして扱い，コンテンツ自身にアクセスすることで得られるデータとする．

本提案方式では，筆者らの研究室で提案されている協調型 Web アーキテクチャ(Cooperative Web Architecture，以下 CWA) を用いる．CWA についての詳細は次節で説明する．CWA を用いることにより，サーバは自身が保持しているコンテンツの RSS フィードだけでなく，その近傍コンテンツの RSS フィードも保持できるようになる．

RSS リーダがコンテンツに対して polling によるアクセスを行うと，対象となったコンテンツを持つサーバは対象コンテンツの RSS フィードと共に，その近傍コンテンツの RSS フィードも同時に RSS リーダに提供する．RSS リーダがそれらの情報をユーザに提供することで，ユーザは中心コンテンツと近傍コンテンツの RSS フィードを全て閲覧することができる．

2.2 協調型 Web アーキテクチャ

CWA とはユーザの近傍検索や目的コンテンツへのナビゲーションのために，ユーザにコンテンツの周辺リンク状況とメタデータを提供する仕組みである．CWA においてサーバは，Web のリンク構造 (Web グラフ) を分散的に管理する．具体的にはサーバが，自身に保存されている Web コンテンツファイルと共に，当該コンテンツを中心とし，そこからリンクの向きに関係なくあるホップ数で到達できる範囲 (管理範囲) の部分 Web グラフ (近傍 Web グラフ) を管理する．さらにサーバは近傍 Web グラフ中のコンテンツのメタデータも同時に管理する．このような機能を持つサーバを AWS と呼ぶ．

コンテンツ内容の更新などに伴い，コンテンツに関するメタデータが更新される．CWA において各 AWS に保持されてい

る共通のコンテンツに関するメタデータは，同一かつ最新のものである必要がある．これに対応するため，AWS には自身が保持するコンテンツのメタデータの更新を検知したとき，そのメタデータを持つ他の AWS に更新を伝える機能を持つ．

上記のメタデータ配信を行うためには，あるコンテンツを管理する AWS が，当該コンテンツのメタデータを保持する AWS を把握している必要がある．ここで CWA では，中心コンテンツからの管理範囲となるホップ数はどのコンテンツにおいても等しく，コンテンツ a の管理範囲内コンテンツであるコンテンツ b にとって，コンテンツ a は必ず管理範囲内コンテンツとなる．よって AWS は自身の保持するコンテンツのメタデータの更新を検知したとき，当該コンテンツの管理範囲内コンテンツを保持する AWS に更新されたメタデータを送信することで，更新メタデータを持つ AWS に更新内容を通知できる [12]．

2.3 配信モデル

提案 RSS フィード配信方式を説明するために RSS フィード配信をモデル化する．モデルにおいてはコンテンツをノード，ハイパーリンクをリンクとすることで，Web をグラフとして扱う．コンテンツは RSS フィードをメタデータとして持つが，この様子をノードが自身の RSS フィードを保持していると表現することとする．ノードは自身が表すコンテンツだけでなく，複数コンテンツの RSS フィードを保持できる．

ユーザが RSS リーダを利用する方法は大きく 2 つに分けられる．1 つは，ブラウザやメーラなど，ユーザのローカル環境にインストールして利用する方法である．これをクライアント型 RSS リーダと呼ぶことにする．もう 1 つは Bloglines [13] のような RSS フィード収集サイトが持つ RSS リーダを利用する方法である．これを Web サーバ型 RSS リーダと呼ぶことにする．どちらも登録されたコンテンツに polling を行なうことで RSS フィードを取得するが，Web サーバ型 RSS リーダはさらに ping の受信も行なうことができる．今回は通信時のモデルとして，RSS リーダはノードに polling を行うことで，ノードが保持する RSS フィードを入手できるとする．

RSS リーダはユーザがノードの登録を行うことで，当該ノードに polling を開始する．また，Bloglines のような RSS フィード収集サイトは RSS リーダの機能を持つコンテンツと言えるので，RSS リーダを持ったノードと見なせる．Web サーバ型 RSS リーダの場合，ユーザは要求する RSS フィードを持つノードを RSS フィード収集サイトノードを通し，RSS リーダにノードの登録を行う．ユーザはノードを見ることでコンテンツの情報を，RSS リーダを見ることで RSS リーダが取得した RSS フィードを把握することができる．

2.3.1 通信要素

ここでモデルで用いる通信要素の特徴をまとめる．表 1 は以下を整理したものである．

- 既存方式の通信要素
  - polling

RSS リーダが RSS フィードを取得する際に行う通信．コンテンツに更新がなくても一定時間置きにノードにアクセスしてしまうため効率が悪い．

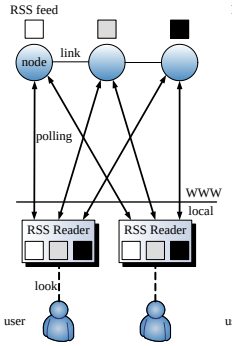

図 1 クライアント型 RSS リーダ

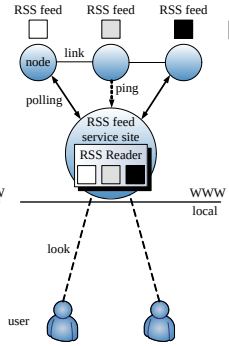

図 2 Web サーバ型 RSS リーダ

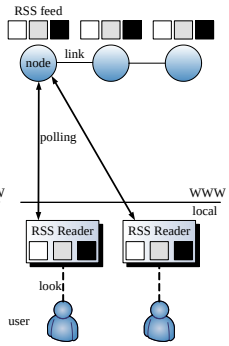

図 3 提案 RSS フィード配信方式

  - ping

ノードが RSS フィード収集サイトに自身のコンテンツの更新を通知する際の通信．更新のときにのみ発生する通信なので効率が良い．しかし RSS フィード配信サイトが RSS フィード収集サイトの存在を認知している必要があるため，配信サイト側の負担が大きくなるという欠点もある．

- 提案方式で新たに用いる通信要素
  - ノード間メタデータ通知

提案方式で用いる，CWA による RSS フィードを含むメタデータ配信時に起こる通信．当通信は各ノードの管理範囲が拡大した場合や，コンテンツの内容が更新されたのを管理範囲ノードに通知する際に発生する．このように更新通知が必要なときにのみ行われる通信なので効率が良い．

2.3.2 既存方式配信モデル

既存方式の場合，各ノードが保持する RSS フィードは，そのノードが示すコンテンツに関する RSS フィードのみである．

図 1 はクライアント型 RSS リーダを用いて RSS フィードを収集するモデルである．それぞれのユーザがある範囲の全ノードの RSS フィードを入手するため，ユーザの RSS リーダは全ノードに polling を行っている．

図 2 は Web サーバ型 RSS リーダを用いて RSS フィードを収集するモデルである．RSS フィード収集ノードが，ある範囲のノードを全て取得するために RSS リーダを用いているモデルである．中央のノードは RSS フィード収集ノードへ ping サーバ登録をしており，更新のたびにその内容を送信する．

2.4 提案方式

提案方式では CWA を用いる．CWA において AWS は自分の管理するコンテンツに関する RSS フィードだけでなく，近傍コンテンツに関する RSS フィードも取得，管理する．これにより，中心コンテンツを持つサーバにアクセスするだけで近傍コンテンツに関する RSS フィードを手に入れられる．またそれに伴い，RSS リーダに登録すべきコンテンツは中心コンテンツのみでよいことがわかる．

以下ではこれをモデルを用いて説明する．

2.4.1 配信手法

図 3 は各ユーザがクライアント型 RSS リーダを用いて，ある範囲に存在する全ノードの RSS フィードを取得するモデル

	通知元	通知先	通知方法	通知先の認識	図の矢印
polling	ノード	RSS リーダ	Pull	不要	⟷
ping/notification	ノード	RSS フィード収集ノード	Push	手動	⇢
ノード間メタデータ通知	ノード	ノード	Push	自動	

表 1　メタデータをやり取りする通信の特徴

である．

提案方式では CWA によりノード間メタデータ通信が行われる．この通信を用いて各ノードは自分の近傍ノードに，自身の RSS フィードを配信する．これにより各ノードは自分の近傍ノードの RSS フィードを得ることができる．すなわち，各ノードは近傍ノードの最新 RSS フィードを常に保持している状態となる．

RSS リーダがノードに polling を行うとノードが保持する RSS フィードを全て取得できるので，ノード自身に関する RSS フィードだけでなく，近傍ノードの RSS フィードも同時に取得できることになる．図 3 の場合，各 RSS リーダには左のノードのみが登録されており，そのノードだけに polling 行う．しかし提案方式により，RSS リーダは他の 2 ノードの RSS フィードも取得できている．ユーザはこれを見ることにより，これら全ての RSS フィードを閲覧できる．

2.4.2　ユーザの手間の削減

従来方式では，上で述べたように各ノードに自身のコンテンツの内容を示す RSS フィードしか保持されておらず，polling を行うことで得られる情報は各 polling の対象ノードに関する RSS フィードのみである．よってユーザはリンクをたどり，関係があると思われるノードを見つけて 1 つ 1 つ RSS リーダに登録を行うことになる．これはユーザにとって大きな負担になると言える．

提案方式では各ノードが自身の管理範囲すべてのノードの RSS フィードを保持している状態であるので，polling を行うことでこれらのノードの RSS フィードをすべて取得できる．よってユーザはわざわざリンクをたどりノードの登録を行う必要はなく，ユーザの負担を軽減することができる．

図で説明すると，RSS リーダに登録されているノード数は RSS リーダへの polling，ping 矢印の数と等しいと見なすことができるが，図 1，2 において RSS リーダへの矢印は 3 本必要なのに対し，図 3 では 1 本となっている．

2.4.3　通信負荷の削減

- 通信回数の比較

通信負荷問題の改善についてクライアント型 RSS リーダと提案 RSS フィード配信方式のモデルを用いて検証した．今回は通信負荷を比較するためにノード間や，ノードと RSS リーダ間の通信回数を求める．これにより複数ユーザが興味のあるコンテンツ群の RSS フィードを取得する際に，クライアント型 RSS リーダよりも提案 RSS フィード配信方式の方が通信回数が少ない条件を求める．通信に際して発生するトラヒック量は RSS フィードや近傍 Web グラフのデータ量に依存し，比較が困難なため本稿では議論しないこととする．

RSS リーダがコンテンツにアクセスする頻度を l[回/day]，各コンテンツの平均更新頻度を r[回/day] とする．$l < r$ の場合，RSS リーダは更新ごとに RSS フィードをユーザに提供することができず，コンテンツの更新が生じたときにユーザに通知できるという RSS リーダの目的を十分達成することができない．よってここでは $l > r$ と仮定する．

他のパラメータを定義する．あるノード C の管理範囲ノードの数を N，その内ユーザが興味を持つコンテンツの割合を p，それらのコンテンツの RSS フィードを取得したいとするユーザの数を U とする．

まず，クライアント型 RSS リーダの場合の通信回数を考える．ノード C の管理範囲に注目したとき．各ユーザは自分の興味のあるノードを RSS リーダに登録するため，RSS リーダに登録されるノード，すなわち polling を行うノードの数は Np である．よって全 RSS リーダにより発生する通信の回数は $NplU$[回/day] と表せる．

次に提案方式の場合の通信回数を考える．各ユーザはノード C に登録すれば，興味のある Np のノードの RSS フィード全てを取得できることになる．これにはノード間メタデータ通知により，ノード C 以外の管理範囲内ノードからノード C へ，ユーザが興味を持たないノードからの通知も含め，合計 $(N-1)r$[回/day] の通信が行われる．また，RSS リーダがノード C にアクセスすることによって初めてユーザに RSS フィードを提供できるが，それには lU[回/day] の通信回数が必要になる．よって提案方式での通信回数は $(N-1)r + lU$[回/day] と考えられる．

これによって導かれる，提案方式の通信回数がクライアント型 RSS リーダの通信回数よりも小さい条件を考えると，

$$p > (1 - \frac{1}{N})\frac{r}{lU} + \frac{1}{N} \tag{1}$$

となる．ここで管理範囲内ノードが十分多い環境を考え，$1/N$ を無視することができると仮定すると，

$$p > \frac{r}{lU} \tag{2}$$

のように式を置き換えられる．これによって示されることは r/lU で示される割合よりも，提案方式で RSS フィードを取得できるノードの内，ユーザが興味を持つものの割合が大きければよいということになる．

これは例えば，あるノードを登録しているユーザが 1 人いる場合を考えたとき，そのノードの管理範囲内ノードが全て 1 日平均 1 回程度更新される日記風のブログであり，かつ RSS リーダが 1 時間に 1 度更新確認のために polling を行うような環境であるならば，$U = 1$，$r = 1$，$l = 24$ となり，ユーザは取得した RSS フィードのうち 1/24 以上を気に入るならば，提案方式の方が優れていると言える．

また，あるノードが表すコンテンツの更新頻度と，RSS リー

ダのノードへのアクセス頻度がほぼ同じであるとした場合でも，そのノードを登録しているユーザが 100 人いるならば $r = l$, $U = 100$ となり，ユーザが取得した RSS フィードのうち 1/100 以上を気に入れば提案方式の方が優れていると言える．

以上のような環境は現実的にも十分考えられる環境であり，提案方式による通信負荷の改善が見込まれると考えるが，コンテンツの更新頻度や，RSS リーダの polling 頻度，ユーザが管理範囲内ノードの内，興味を持つ割合など具体的な数値が示せないので明確に示すことができない．また，サーバ型 RSS リーダについては不明確なパラメータがさらに増えるため，まだ検証を行えていない．これらの検証は今後の課題である．

- 人気コンテンツへのアクセスの分散

Web コンテンツの中で多くのユーザに興味を持たれるコンテンツは人気コンテンツとされる．人気コンテンツが RSS フィードを配信すると，多くのユーザが当該コンテンツを RSS リーダに登録するため大量の polling アクセスが集中してしまう．さらに，この大量の polling アクセスは一定時間おきに生じるので，サーバにとって大きな負担となる．

提案方式では人気コンテンツの RSS フィードが，リンク的に近傍にあるコンテンツを RSS リーダに登録することでも取得可能である．このため従来方式では人気コンテンツに集中していた RSS リーダによる polling アクセスを，他のコンテンツに分散させることができる．この結果，人気コンテンツを持つサーバの処理するアクセスが減少し，大量アクセスの集中によるサーバのダウンなどの事態の回避につながると考えられる．

3. 一切合財 RSS リーダ

3.1 概　　要

従来の RSS リーダは登録されたコンテンツの RSS フィードを取得するのみであった．この章で提案する一切合財 RSS リーダは前章の提案 RSS フィード配信方式に対応した RSS リーダであり，登録されたコンテンツの RSS フィードだけでなく，リンク構造的にその近傍に存在するコンテンツの RSS フィードも同時に取得し，ユーザに提供する．

3.2 RSS フィードの選別

提案方式により RSS リーダに登録したノードに polling を行うだけで，その近傍に存在するコンテンツの RSS フィードを全て取得できるようなるが，コンテンツ間のハイパーリンクが常にユーザの意図する関係を表しているとは限らず，不必要な RSS フィードも多く取得してしまう．よって必要な RSS フィードと不必要な RSS フィードが混在してしまうことで，ユーザが本当に入手したい情報を探すのに非常に手間がかかると考えられる．ゆえに，よりユーザが目的の RSS フィードを見つけやすくする機能を付加した RSS リーダが求められる．以下ではその機能を提案する．

- 木構造表示

リンクで連結されたノードは何かしらの関係があると見なすことができ，リンクをたどることで関連コンテンツを発見できる．しかし途中でユーザが興味を持たないノードにたどり着いた場合，それ以降のホップでたどり着けるノードはそのノードに関係するノードであると考えられ，ユーザが興味を持つ可能性は低い．

そこで RSS リーダが取得した RSS フィードを，指定したノードを根とした木構造状に表示する．ユーザが必要のないと感じたノードを非表示設定にすると，そのノードを根とする部分木がすべて非表示となる．これにより不必要であると考えられる RSS フィードをまとめて非表示にできる．

この方法は Web リンク構造が複雑化しても，ユーザが自分で非表示にする範囲を指定して，複数の RSS フィードをまとめて非表示にできるという点が利点となるが，ユーザは Web リンク構造をある程度眺めて取捨選択を行なう必要があるため，やや手間がかかる．

- ホップ数制限

指定ノードからのホップ数が大きいノードほど，指定ノードとの関係が薄れ，ユーザにとって興味のないものとなる可能性が高い．そこでユーザがホップ数を指定することで，RSS リーダが取得した全 RSS フィードの内，指定ノードからのホップ数がユーザの指定したホップ数以内になるもののみを表示させる．

この方法では木構造表示によるものと違い，大量の RSS フィードを一度に非表示にすることができるためユーザの手間は少ないが，Web リンク構造に対して柔軟ではないので，ユーザが興味を持つはずの RSS フィードも非表示としてしまう可能性が高い．

- Web コミュニティの抽出

Web リンク構造等を基に Web コミュニティの抽出を行う．そして抽出した Web コミュニティごとに RSS フィードをまとめてからユーザに提供する．こうすることで取得した RSS フィードがある程度ジャンル分けされ，見やすい形でユーザに提供される．現在提案されている方法では，Web コミュニティ抽出の方法は完全に自動で行なわれるものが多く，ユーザの手間を必要としない．

今回は実験的に木構造表示による RSS フィード閲覧機能を採用する．Web コミュニティ抽出も併用することでよりユーザの閲覧を支援することができるが，Web リンク構造によって効果が様々に異なってくることが予想されるため，今回は比較的効果が判断しやすい木構造表示のみを採用し，検証を行う．

3.3 関 連 研 究

3.2 節で述べたように，本提案方式によって Web グラフ上での近傍 Web サイトの RSS フィードが得られることと，意味的なフィルタリング手法との本質的な関連はない．しかし実用上の観点からは，大量の情報から必要な情報を取捨選択する本質的な問題であると言える．ここでは，他の大量の RSS フィードをフィルタリングしてユーザに提供する研究について触れる．

文献 [14] では，ユーザが過去に閲覧した RSS フィードからユーザの嗜好をプロファイリングし，新たに取得した RSS フィードの要素とユーザの嗜好を比較することで，この RSS フィードをユーザに推薦するかどうかを判定できるツールを提案している．この方式では，推薦される RSS フィードはユーザが RSS リーダに登録した Web サイトから提供されるものであるため，ユーザは自分の興味のある情報を高い頻度で配信し

てくれる Web サイトを発見するか，とりあえず大量の Web サイトを登録する必要がある．よってユーザが Web サイトを発見する手間は残される．しかし本提案 RSS フィード配信方式では，登録 Web サイトと関連する Web サイトの RSS フィードを一切合財収集してくれるため，そのような手間が削減されると言える．よってこの方式は大量の RSS フィードを処理するのに適しており，筆者らの提案する RSS フィード配信方式に利用できると考えられる．

4. 試作システム

4.1 システム

提案方式を実現するため，Web サーバに Web グラフと RSS フィードの管理機能を追加する形で実装する．また，RSS リーダ機能についても Web サービス型として実装する．

Web サーバは Web グラフ管理機構，RSS フィード管理機構，RSS フィード提供機能の各モジュールから構成され，それぞれ Web グラフの管理，必要な RSS フィードの管理，クライアントリクエストに対する応答を主に担当する．

RSS リーダはコンテンツ登録機能，RSS フィード表示機能，RSS フィード取得機能により構成され，各部は RSS フィードを取得するコンテンツの登録，取得している RSS フィードの表示，各 Web サーバへのリクエスト送信を行なう．

4.2 vAWS モジュール

本来 AWS の機能はサーバの機能と一体で動作するべきものであるが，導入のしやすさから virtualAWS モジュール (以下 vAWS) を用いる．vAWS は自サーバが保持している HTML コンテンツから周囲 n ホップの Web グラフを維持，管理できるモジュールである．また，ディレクトリ単位で Web リンク情報を管理できる機能を持つ．CGI/perl が動作する Web サーバにおいて，perl による CGI プログラムを適当なディレクトリに配置し，これを設定するデータファイルを vAWS を導入したいディレクトリに配置することで CWA の機能を実現できる．1 つのサーバに複数の仮想的な AWS を表現することも可能になっている．vAWS が保持する機能は以下の 4 つである．

- HTML コンテンツのアップロード時に HTML からリンク情報を抽出し，Web グラフに反映させる機能．
- HTML コンテンツの削除時に，それにともなうリンク構造の変化を Web グラフに反映させる機能．
- 他 vAWS から送られてくる Web リンク情報を，保持している Web グラフに反映させる機能．
- 自サーバの Web グラフ更新時にリンク情報の変更を必要な他サーバに送信する機能．

vAWS はコア機能である Web グラフの管理機能のみが動作する状態であったので，これに CWA におけるメタデータ管理，配信機能の実装として RSS フィード管理，提供機能を追加した．追加した機能を以下に示す．

- 他 vAWS から Web グラフと共に送られてくる RSS フィード群を読み込み，データベースへ送る機能．
- 自サーバの RSS フィードの更新機能．
- 自サーバの RSS フィード更新時に，必要な他サーバに RSS フィードを通知する機能．

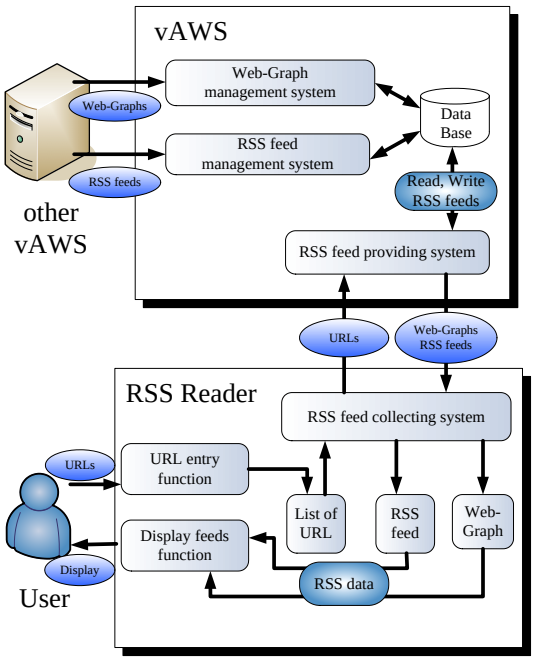

図 4　システムブロック図

これにより，各サーバが保持コンテンツの近傍コンテンツの RSS フィードを保持することができるようになる．以下では vAWS の機能を，図 4 のシステムブロック図を基に具体的に説明する．

4.2.1 Web グラフ管理機構

Web グラフの維持，管理機能を受け持つ部分である．Web グラフはデータベースに保持されており，必要なときに追加，削除，更新が行われる．このデータベースに管理されている各コンテンツの管理範囲を基に RSS フィードの通知範囲が決められる．

4.2.2 RSS フィード管理機構

RSS フィードの維持，管理を行う機能を受け持つ．サーバが保持する RSS フィードは，Web グラフの構造変化によって新たに管理範囲にノードが追加されたときや，管理範囲内の RSS フィードが更新されたときに追加される．これらの得られた RSS フィードはデータベースに保存され，Web グラフの同期時や，提案 RSS リーダが自サーバの RSS フィードを取得する際に呼び出される．

4.2.3 RSS フィード提供機構

指定したコンテンツの管理範囲内コンテンツの RSS フィードを提供する機能を受け持つ．RSS リーダが必要な RSS フィードを一切合財入手するときに使われる．自サーバの持つコンテンツの URL を入力とし，データベースにアクセスし，そのコンテンツの近傍 Web グラフと近傍コンテンツの RSS フィードを返す．

4.3 RSS リーダモジュール

次に RSS リーダモジュールを図 4 を基に説明する．このモジュールを適当なディレクトリに配置しブラウザでアクセスすることで Web サーバ型 RSS リーダと同様の動作を行える．

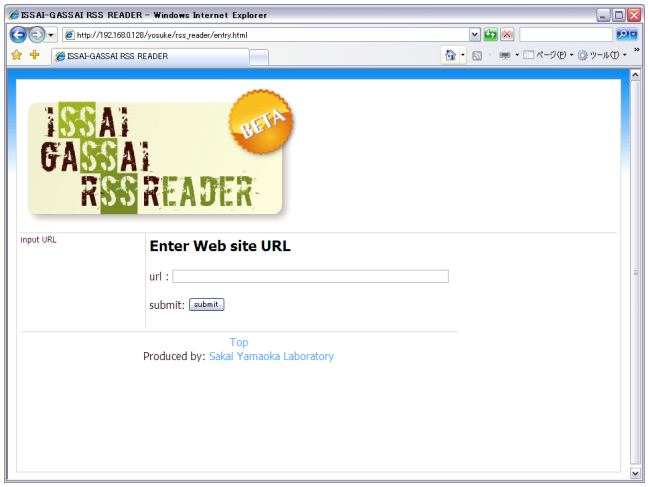

図 5　RSS フィード登録画面

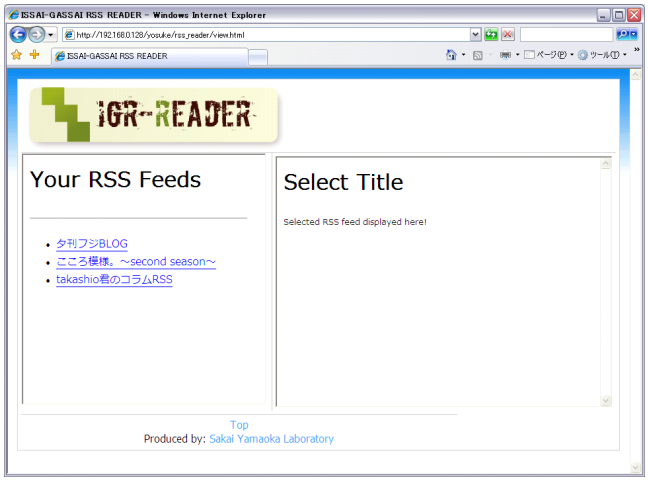

図 6　RSS フィード選択画面

4.3.1　コンテンツ URL 登録機能

ユーザが RSS リーダに RSS フィードを取得したい URL を登録する機能を受け持つ．登録された URL のリストは保存され，これを基に RSS リーダは polling を行う．

4.3.2　RSS フィード取得機能

vAWS から RSS フィードをまとめて取得するための機能を受け持つ．vAWS が持つ RSS フィード提供機能部に URL を指定してアクセスすることで，指定コンテンツの近傍 Web グラフと，近傍コンテンツの RSS フィードを一切合財取得する．得られた Web グラフと RSS フィードは保管され，ユーザに提供する際に呼び出される．

4.3.3　RSS フィード表示機能

ユーザに RSS フィードを提供する機能を受け持つ．ユーザは画面を見て得られた RSS フィードを確認できる．指定したコンテンツの近傍の RSS フィードを一切合財表示するため，ユーザにとって不必要な RSS フィードも表示される可能性が高い．よって大量の RSS フィードをフィルタリングする必要がある．今回の実装で用いる方法，効果は次章で述べる．

4.4　機能事例

実際の Web のリンク構造の一部を vAWS 上に再現し，RSS フィードの取得を行なう様子を示す．今回のシステムの使用例として 2007 年 1 月 5 日における「夕刊フジ」(http://www.yukan-fuji.com/) の最新記事公開サイトを中心ノードとし，記事に関するトラックバックなどをリンクとして Web からリンク構造抽出し，vAWS 上に再現した．

RSS リーダはリンク構造を基に RSS フィードを表示するが，途中 RSS フィードを配信しないサイトが含まれると RSS リーダはノードに対応するデータを持たないので Web リンク構造を完全に表現できない．これに関しては，ノードと RSS フィードが同時に管理されることを利用し，RSS フィードを配信しないサイトに関しては HTML の title タグの部分を RSS フィードの内容として扱うなどすることで解決できると考えられる．また，なんでも RSS [15] のように HTML 内の情報を分析して RSS フィードを作成する方法もある．今回は簡単のために RSS フィードを配信できるサイトのみをデータとして扱った．これには夕刊フジ最新記事公開サイト，トラックバック先のブログ，ブログと同カテゴリに属する他のブログ最新記事公開サイトなどが含まれる．

ユーザは RSS フィードを取得したいコンテンツの URL を RSS リーダに登録する (図 5)．RSS リーダはその URL を登録コンテンツリストに書き込む．polling による通信を開始するとき，RSS リーダはそのリストに載っている各 URL のコンテンツを保持するサーバにアクセスする．vAWS の RSS フィード提供機能によって Web グラフと RSS フィードを取得した RSS リーダは，自分のデータベース部分にそれぞれを保管しておく．ユーザは閲覧したい RSS フィードを選択すると (図 6)，画面右側に RSS フィードの更新情報が表示され，画面左に近傍 RSS フィードが木構造状に表示される (図 7)．この画面において，未読は太字，既読は標準のフォントで表示される．

図 7 における近傍 RSS フィードは主に「夕刊フジ」の記事に関する意見などを書いたブログである．1 ホップ目の全 RSS フィードは根ノードである「夕刊フジ」のいずれかの記事に関連している．例のとしては,「takashio 君のコラム RSS」と「あおのり世相をぼやく」の最新情報に，中心コンテンツの記事の 1 つである「ケータイのキー音を消さない人たち」に関するヘッドラインが含まれていた．このようにお互い関連のあるサイトの RSS フィードをまとめて入手できていることがわかる．

4.5　非表示設定の効果

前章のようにニュース記事を中心コンテンツとして設定した場合，様々な内容の RSS フィードを同時に取得してしまっても，それはユーザの意図したものであるので不利益なものではなかった．ここで，例えば「こころ模様。～second season～」を中心コンテンツとし，このブログに関係する記事のみを取得したいとする．このとき，リンク先である「夕刊フジ」の RSS フィードはブログと関係の無いものがほとんどであり,「夕刊フジ」とリンクしている他のブログも含め大量の不要な RSS フィードが混ざってしまう (図 8)．これによりユーザの RSS フィード閲覧を妨げてしまう．

ここで RSS リーダの非表示機能を用いる．非表示機能とは，ある RSS フィードを選択すると，その RSS フィードを配信し

図 7　RSS フィード閲覧画面

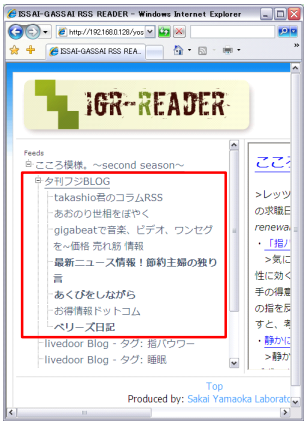

図 8　非表示設定前

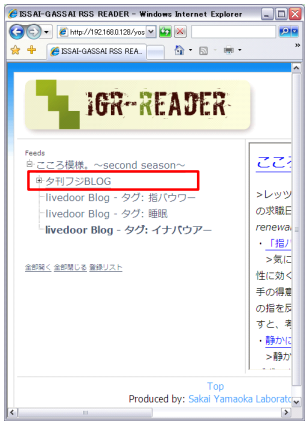

図 9　非表示設定後

ているサイト以降のホップの RSS フィードを非表示にすることができる (図 9)．これによりユーザが必要としている部分のリンク構造がより簡潔に示されるようになった．

5. 結　　論

本稿ではまず，特定のコンテンツだけでなくその近傍の RSS フィードを一切合財入手することの有効性を示した．

次に，それを既存の RSS フィード配信方式で行う際の問題点を挙げ，それを解決するための RSS フィード配信方式を提案した．さらに提案方式に対応した RSS リーダについて述べ，大量の RSS フィードを扱うための方法に対して検討を行った．

最後に試作システムを実装し，実際の Web リンク構造を再現し，RSS フィード取得を行って効果を確認した．また RSS リーダの表示を操作し，効率的に RSS フィードを閲覧できることを確認した．

今回の試作システムでは木構造表示のみで RSS フィードを整理したが，その他にも RSS フィードを整理する機能を RSS リーダや RSS フィードを提供するサーバに付与することによって，さらにユーザが利用しやすい仕様へと改善できる．

RSS フィードを整理するために利用できるものとしてリンクが考えられる．すべてのリンクを同等に見るのではなく，そのリンクが持つ情報を用いて RSS フィードを差別化することで RSS フィードの整理をすることができると考えられる．例えば双方向にリンクを張っているサイトを発見する手法が考えられる．このようなサイトは，一方向のみのリンクで接続されたサイト以上に関連性が強いと考えられる．さらにサイト内でのリンクの出現順序も利用することができる．出現順序が近いと，それらのリンク先サイトは関連性が強いことが考えられる．以上のようなサイトを明示的に示したり，表示のときに近くに配置することによってユーザの閲覧を補助することができる．

また，サイトの管理者の嗜好やユーザの嗜好を反映させて RSS フィードを表示することも，ユーザの閲覧を補助する方法として考えられる．サイト管理者は，そのサイトの近傍にあるサイトすべての RSS フィードを単純に提供するのではなく，管理者の嗜好に合うように RSS フィードを選択して提供することで，より管理サイトの特徴を表現できる．他にも bookmark サイトやお気に入りサイトなどを明示的に表示させることで，これらが他よりも管理者の嗜好の反映されたサイトであることをユーザに示すことができる．また，クライアント側でもユーザの閲覧履歴などを記録，分析してユーザが興味を持つと予測されるものを優先的に表示させるという方法も考えられる．

以上のように，よりユーザが利用しやすいインターフェースとするための拡張は多く考えられ，それらの機能の追加と効果の検証は今後の課題である．

文　　献

[1] IPSJ Magazine, Vol.46, No.1, 2005
[2] Google, http://www.google.co.jp.
[3] Jon M. Kleinberg. “Authoritative sources in a hyperlinked environment,” In Proc. of the ninth annual ACM-SIAM symposium on Discrete algorithms, pp. 668–677, 1998.
[4] Jeffrey Dean and Monika R. Henzinger. “Finding related pages in the World Wide Web,”In Proc. of the 8th WWW Conference, 1999.
[5] 豊田正史. “WWW における関連コミュニティ群の発見,” 情処研報 DBS122–40, IPSJ, 2000.
[6] 村田剛志. “参照の共起性に基づく Web コミュニティの発見,” 人工知能学会誌 Vol.16 No.3, pp. 316–323, 2001.
[7] 原田昌紀, 風間一洋, 佐藤進也. “参照共起分析の Web ディレクトリへの適用,” IPSJ 研究報告 NL–142–007, 2001.
[8] 浅野泰仁, 吉田雄介, 西関隆夫, 豊田正史, 喜連川優. “サイト間グラフの最小カットを用いたウェブ上のコミュニティ発見法,” アルゴリズム研究会情処研報, AL–095–8, pp. 51–58, 2004.
[9] M.Hicks. “RSS comes with bandwidth price tag,” eWeek, Sept, 2004
[10] “Windchase: Microsoft blog の事件にみる，RSS のスケーラビリティの問題”, http://windchase.jp/blog/27.html.
[11] Kobayashi Aki, Yamaoka Katsunori, Sakai Yoshinori. “Cooperative Web Architecture for Search and Navigation Assistance,” In Proc. of ICWI2002, IADIS, 2002.
[12] 高砂 幸代, 小林 亜樹, 山岡 克式, 酒井 善則. “Web サーバ間での部分 Web グラフ同期方式の提案,” DEWS2006 7C–o2, 2006.
[13] Bloglines, http://www.bloglines.com.
[14] 向井 誠, 青野 雅樹. “RSS に基づく個人向け内容型情報推薦プロトタイプシステム”, IPSJ 2005–NL–169, 2005.
[15] 南野 朋之, 奥村 学, “なんでも RSS-HTML 文章からの RSS 自動作成”, In Proc. of JSAI, 1A4–01, pp. 1–4, 2005.