2006－ICS－145（3）
2006／10／25

WisdomWeb: プッシュ型配信に基づく次世代コンテンツ配信について

大囿　忠親[†a)]　　新谷　虎松[†b)]

WisdomWeb: A Next Generation Web Contents Delivery System based on a Push Delivery Mechanism

Tadachika OZONO[†a)] and Toramatsu SHINTANI[†b)]

Abstract. 本論文では，プッシュ型配信に基づく新たなコンテンツ配信として時間保証型配信を提案する．本研究では，紙のように自由にかつ簡単に扱え，さらに智を備えた Web の実現を目指しており，そのような Web を WisdomWeb と呼んでいる．WisdomWeb の実現に向けて，Web ブラウザによる情報交換を直感的かつ効果的に行うソフトウェアとして WisdomWeb エンジンを開発した．WisdomWeb エンジンは，モバイルエージェントに基づくコンテンツ配信システムであり，Web ブラウザを紙のように扱うことを可能にする．本論文では，WisdomWeb エンジンのプッシュ配信機能を応用した時間保証配信についてのべ，その実装を示す．時間保証配信により Web に基づくコンテンツ配信の可能性が広がり，情報発信がより身近な存在になることが期待される．

Keywords. WWW，プッシュ型配信，Web エージェント

1. はじめに

一般社会へのインターネット，特に WWW（World Wide Web）の普及に伴い，WWW は生活インフラとして定着しつつある．例えば，インターネット広告の市場規模は，ラジオ広告の市場規模を超え，市場規模も前年度比 150 %以上の伸びで広がっている．WWW の単純さはその普及を促進したと考えられるが，今後の主要なメディアとしてさらなる発展を遂げるには多くの問題が残されている．

本研究では，紙のように自由にかつ簡単に扱え，さらに智を備えた Web の実現を目指しており，そのような Web を WisdomWeb と呼んでいる．WisdomWeb の実現に向けて，Web における次の 3 つの問題に取り組んでいる．一つ目の問題は Web による情報発信の困難さである．二つめの問題は，Web によるリアルタイムな情報配信の欠如である．最後の問題は，携帯端末上へのコンテンツ配信の制限である．本研究では，これらの問題を解決するための基盤的なソフトウェアとして，WisdomWeb エンジンを開発した．

WisdomWeb エンジンは，Web 上で動作するモバイルエージェントシステムである MiSpider [1], [2] の拡張である．MiSpider におけるエージェントは，ユーザが閲覧ページを移動した時に，セッション間で内部情報を保持したまま移動し，継続的にサービスを提供可能である．また，Web 上の任意のエージェント間で通信を行うためのメッセージ通信能力を持つ．このような，エージェントの機能を利用することで，ユーザにさまざまな Web ブラウジング支援を提供したり，新たな Web アプリケーションを構築することができる．本論文では，WisdomWeb エンジンとその応用システムとしてプッシュ型配信に基づく次世代コンテンツ配信システムを提案する．

以降，2. では従来の Web の問題点と，WisdomWeb について議論する．3. では，WisdomWeb エンジンとその要素技術である MiSpider について述べる．4. では WisdomWeb エンジンに基づく新たなコンテンツ配信である，時間保証型コンテンツ配信システムを提案する．5. では時間保証型コンテンツ配信システムの実装とその評価を示し，最後に 6. で本論文をまとめる．

† 名古屋工業大学大学院情報工学専攻
〒 466-8555 名古屋市昭和区御器所町
a) E-mail: ozono@nitech.ac.jp
b) E-mail: tora@nitech.ac.jp

2. WisdomWeb：紙のような Web

WisdomWeb は，紙のように自由にかつ簡単に扱え，さらに智を備えた Web である．ここで,「紙のような」とは，メディアリテラシの高くないユーザにとっても，操作が簡単で自由に読み書きでき，導入が容易で誰でも使え，かつ，予備知識を必要とせず敷居が低いという意味である．WisdomWeb では紙のような Web に知的なユーザ支援機能を加え，メディアリテラシの高くないユーザにも高いレベルの Web をサービスを提供できることが重要である．

本研究では，WisdomWeb の実現に向けて，Web における次の 3 つの問題に取り組んでいる．一つ目の問題は Web による情報発信の困難さである．ブログの普及により，以前に比べれば情報発信は容易になったといえるが，一般的なユーザが Web を紙のように自由に扱えるほど簡単にはなっていない．ブログではページ内のレイアウトに制限がある．一般的な Web ページ編集ソフトでは，ページの編集後にページをアップロードする必要がある．一般的なユーザにとって，このアップロードの困難さが壁となっている．
また，新聞などのメディアにおいて，単に記事を配信するだけでなく，その記事の使い方，例えば，何ページに表示するのかが適切か等をメタ情報として付加することが求められている．現状の，Web ページは，計算機による再利用を想定していない点が問題である．

二つめの問題は，Web によるリアルタイムな情報配信の欠如である．Web は新聞などの従来のメディアに比べれば圧倒的な即時性を持つが，情報更新が閲覧者の Web ページ読込動作に依存している．すなわち，閲覧者が閲覧中のページの内容が更新されたとしても，閲覧者が Web ページをリロードしない限り，閲覧者に更新内容が伝わらない．テレビで防災情報を放映すれば，直ちにその情報は視聴者の目にとまるが，Web の場合は閲覧者が Web ページをリロードするまで防災情報が伝わらない点が問題である．また，プッシュ配信を行った場合に，配信されたコンテンツの効果を測定する手法が未知でありさらなる研究が必要である．

最後の問題は，携帯端末上へのコンテンツ配信の制限である．携帯型の情報端末は，画面解像度，計算能力，通信速度，消費電力等の点で，パソコンに比べて能力が限定されており，パソコンのような情報端末と同じ手法でコンテンツ閲覧用システムを実現すると，使いづらくストレスの多いシステムとなる．携帯電話による Web ブラウジングの比率が高まっており，携帯電話による情報発信・閲覧に関して徹底的に技術開発することは，携帯情報端末分野におけるソフトウェアの国際的なイニシアティブを取るチャンスに繋がる．

本研究では，一般的な Web ブラウザ上でこれらの問題を解決するための基盤的なソフトウェアとして，WisdomWeb エンジンを開発した．WisdomWeb エンジンにより，誰もが Web ページを紙のように扱え，その情報がリアルタイムに更新可能になる．

本研究では，WisdomWeb エンジンの実現に向けて，紙のように扱える Web をブラウザ上で実現した [4]．ブラウザは最も普及しているアプリケーションであり，誰でも使えるという目的に合致しうる唯一の共通基盤である．Web を紙と同程度に簡単に扱えるメディアにするためには，追加ソフトウェアのインストールが大きな障害になる．ユーザが目の前にある Web ページを，実際にそこにある紙のように扱えることを可能にするソフトウェア技術を実現した．

プッシュ配信の普及には，Web ブラウザにプラグインソフトウェアを追加する必要のないプッシュ配信技術とそれに基づく次世代コンテンツ配信システムが必要である．さらにWWWの規模を考慮したスケーラビリティも必要である．Web に基づくプッシュ配信は，ポーリングに基づくメッセージ通信が必要となるため，サーバへの負荷が高くなる．本論文では，クライアントである Web ブラウザとサーバが協調することで，リアルタイム性を損なうことなく大規模なプッシュ配信を実現する手法として，番組表に基づくプッシュ配信を提案する．

コンテンツを限定された能力しかない情報端末用に向けてコンテンツを変換することで，計算負荷や通信負荷を軽減しつつ，通常の情報端末と同様なコンテンツ閲覧を実現するためのシステムが実現可能である．ここでは，そのコンテンツの操作に適したプログラムをコンテンツに付加することで軽量なコンテンツ配信を実現する．すなわち，ここでのコンテンツ配信は，コンテンツとそのコンテンツ専用のブラウザをペアで送信することに相当する．これにより受信側の端末に合わせて，配信するコンテンツやプログラムを最適化することが可能になる．これにより，携帯情報端末上でも，紙のように情報配信を行えるような技術が実現可能になる．

本論文では，Web によるリアルタイムな情報配信に焦点をあて議論を進める．

2.1 プッシュ型配信に基づく次世代コンテンツ配信

従来の Web ページは基本的には静的なコンテンツであり，閲覧者からの Web ページ配信要求時の内容が表示され続ける．Web ページの表示内容を変えるには，Web ブラウザのリロード機能などを用いて，再度，Web ページを Web サーバから読み込む必要がある．

従来のプッシュ配信では，RTSP(Real Time Streaming Protocol) のような，動画配信におけるプッシュ型の情報配信が主流であった [5]，動画配信の場合は専用の再生ソフトやプラグインを利用することで，プッシュ配信を実現している．本研究におけるプッシュ型配信は，動画だけでなく HTML で扱える全てのコンテンツのプッシュ配信を対象にする．

本論文では，プッシュ型配信に基づく次世代コンテンツ配信として，時間保証型コンテンツ配信を提案する．ここで，時間保証型コンテンツ配信とは，Web コンテンツの内容をテレビのように時間に従い変化させる配信である．すなわち，コンテンツの表示される時間帯を保証するコンテンツ配信を実現する．今後，テレビや携帯端末などの，従来のパソコンとは異なり入力が困難な閲覧機器による Web ページの閲覧が普及すると考えられている．このような閲覧機器を用いた場合，閲覧者の積極的な操作に基づく Web コンテンツの配信では不十分であり，閲覧者が操作をしなくてもコンテンツを配信できることが重要になる．一般的な情報配信者と閲覧者にとって，時系列に従った情報配信はテレビやラジオなどの既存のメディアによって慣れており，受け入れることに抵抗感が少ないと考える．時間保証型コンテンツ配信を，Web ページ上でも実現することにより，これまでに様々なメディアで培われてきたコンテンツ配信の技術や知見を応用できる可能性がある．

放映中のテレビ番組と Web コンテンツの内容を同期させるためにプッシュ型配信を利用する報告もある [6]．文献 [6] では，閲覧者のパソコン上にブラウザ以外の特別なアプリケーションを導入し，プッシュとプルを適切に使い分けることで，効率の良いコンテンツ配信を提案しているが，特別なアプリケーションを導入させる必要がある点が，課題である．誰にでも簡単に使えるような Web という観点から，通常の Web ブラウザに追加ソフトウェアを導入せずに，動作することが好ましい．携帯端末などの様々な HTML 表示可能なデバイスには，追加ソフトウェアの導入が容易ではないものも少なくない．

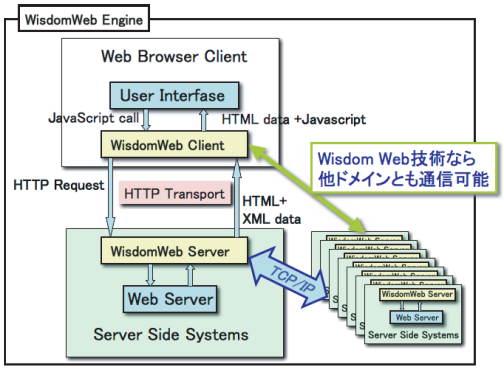

図 1　WisdomWeb エンジンの構成

3. WisdomWeb エンジン

WisdomWeb エンジンは，一般的な Web ブラウザ上において次の 4 つの機能，1) Web ブラウザ上に自由に書き込める機能，2) Web ブラウザに対するプッシュ配信機能，3) 多様な閲覧用デバイスのためのコンテンツ変換機能， 4) 効果測定のためのデータ収集機能，を実現するための基盤である．これらの機能を，一般的な Web ブラウザに対して特別なソフトウェア(注1)の追加インストールをすることで，広い範囲のユーザに対するサービスの提供が可能になる．図 1 に，WisdomWeb エンジンの構成図を示す．WisdomWeb エンジンは，WisdomWeb クライアントと WisdomWeb サーバから構成される．WisdomWeb クライアントと WisdomWeb サーバ間では，HTTP に基づくメッセージ通信が可能である．以降，4 機能について述べる．

1) の Web ブラウザ上に自由に書き込める機能 [4] では，通常の Web ブラウザで既存の Web コンテンツの内容を編集できるだけでなく，Web ブラウザ上に手書きができ，さらにユーザにデータの保存を意識させないので，ユーザは Web オーサリングソフトやアップロードなどについて知らなくても情報を発信できるようになり，情報発信に対する敷居を低くすることによる情報発信の広がりに期待できる．

2) の Web ブラウザに対するプッシュ配信機能は，Web ページに対して任意のタイミングで情報を配信することを可能にする．従来の Web では，閲覧中の Web ページに対する更新が，ユーザのリロード時にしか反映されなかった．AJAX [9] と呼ばれるプログラミング技術を利用することで，同様な機能を実現可能

(注1)：プラグインなど

ではある．WisdomWeb エンジンの特筆すべき点は，AJAX ではできない，複数のサーバとのプッシュ配信をも可能にする点である．1) と 2) の機能を組み合わせることで，複数ページ間の内容をリアルタイムに同期させることも可能であり，グループウェアへの応用が期待できる．

3) の多様な閲覧用デバイスのためのコンテンツ変換機能は，表示デバイスに対して最適な形式にコンテンツを変換することで表示や操作方法のみならず通信方法までをも最適化することを可能にする．例えば，携帯電話用に Web ページを送るときに画像サイズの最適化だけでなく，その操作プログラムの最適化までも可能になる．

4) の効果測定のためのデータ収集機能は，プッシュ配信により配信された Web コンテンツの効果を測定するための機能である．単に配信回数を計測するだけでなく，Web ページに対するユーザの挙動を考慮することで，より厳密な効果測定を可能にする．例えば，Web コンテンツは，配信されても画面上に表示されるとは限らない．本システムでは，実際に画面上に表示されたかどうかすらも考慮した効果測定を可能にする．

本論文では，2) と 4) について議論し，WisdomWeb エンジンに基づく時間保証型プッシュ配信を提案する．

3.1 Web エージェント MiSpider

WisdomWeb エンジンのコア技術である MiSpider について説明する．MiSpider は，Web ページ上で動作する Javascript 言語に基づくエージェントシステムとしてである [1], [2]．Web ページ上の知的な支援をエージェントとしてカプセル化することで，Web ページ閲覧中にサービスを動的にプッシュしたり，単一ユーザの活動を支援するだけではなく，複数のユーザ間の作業のより高度な支援を可能にする Web ページの実現も可能になる．

ユーザが複数のページを閲覧する場合，それらのページ間でも継続的なサービスが提供されることが好ましい．Potter [3] は，Web 閲覧におけるセッションを記憶媒体に保存する手法を提案しているが，Javascript の状態保存は考慮していない．ブラウザとサーバ間は自由に通信できない，かつ，ユーザのページ遷移等のタイミングは予測困難なので，Javascript の継続的なサービスの実現が困難であった．MiSpider では，ユーザによる非同期な要因によるサービス実行中断においても，サービスの継続を可能にする．

MiSpider は Web ブラウザ上で永続性，メッセージ通信，GUI などの API を提供する Web エージェントである．開発者は，提供された API を利用して，容易にエージェント開発を行うことができる．

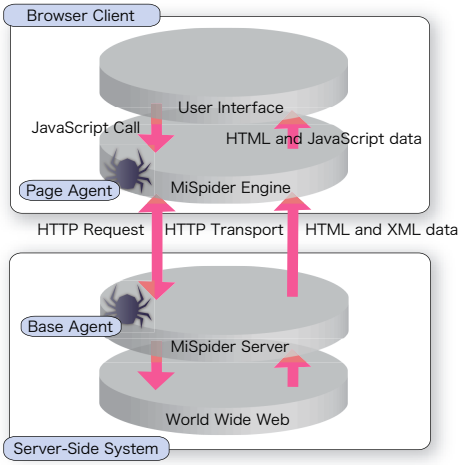

図 2　MiSpider の構成

処理の記述には Javascript を用いる．Javascript は Web 上におけるユーザの操作に応じた処理を記述できるため，Web 上におけるエージェント開発に適している．

図 2 は，MiSpider の構成図である．MiSpider は，ベースエージェント（図 2 中の下の四角）とページエージェント（図 2 中の上の四角）から構成される．ベースエージェントはサーバ上で動作するエージェントであり，ページエージェントは Web ページ上で動作するエージェントである．ページエージェントとベースエージェントは，互いに通信可能である．ベースエージェントは，サーバ上で動作しているので，計算機資源に関する制約がページエージェントよりも緩い．例えば，CPU の実行速度，ネットワーク通信帯域，または 2 次記憶装置などに関して，ページエージェントよりも圧倒的に豊富な資源を用いて処理を実行可能である．ページエージェントは，ユーザとリアルタイムにインタラクションを行えるという利点がある．例えば，ベースエージェントは，Web ページ内の情報を表示中に変更したり，表示されている情報の表示時間を計測することができない．ネットワークの遅延により，ベースエージェントはユーザとのインタラクションに関して不利である．ページエージェントは，Web ブラウザ上で動作するという都合から，インタプリタ言語による処理の遅さや，Web ブラウザ上のセキュ

リティに関連した制約により，実行可能な処理が非常に限られている．例えば，任意のサーバと通信することすら困難であり，ベースエージェントとしか通信できない．互いの利点と欠点を補うために，ベースエージェントとページエージェントが協調して，ユーザに対してサービスを提供することが必要になる．例えば，ページエージェントが遠隔地で動作するセンサ及びアクチュエータで，ベースエージェントはそれを遠隔操作する知的なソフトウェアという形態が考えられる．

3.2 MiSpider によるサービス提供

ユーザが MiSpider によるサービスを受けるには 2 つの方法がある．1 つ目の方法は，ユーザが，Web ブラウザを用いて本システムにログインする方法である．ログイン時に利用するエージェント名，パスワード，および閲覧する Web ページの URL を入力する．もう 1 の方法は，閲覧中のページ上で，ブックマークレットと呼ばれる Javascript を実行することである．本ブックマークレットは，MiSpider をサーバからダウンロードして実行するための方法が記述されており，これにより MiSpider が対象 Web ページ上で動作可能になる．これらの操作を行うことで，ユーザは，指定したエージェントの機能をブラウジング中に利用することができる．

前者の方法では，Web ブラウザからの閲覧は，MiSpider サーバ上の Web プロキシ（CGI として実装される）を経由して達成される．Web ページへのアクセスは，MiSpider 経由で行われる．これにより，Web プロキシを通過する時に，ベースエージェント が閲覧予定の Web ページに，MiSpider のページエージェントを埋め込むことが可能になる．ページエージェントが埋め込まれたページは，ユーザに送信されブラウザ上で評価されることで，MiSpider によるサービス提供が可能になる．

3.3 永 続 性

MiSpider では，セッション間における継続的なサービス提供を実現している [8]．一般的な Javascript では，ユーザが閲覧ページを移動した際に，実行情報が初期化される．閲覧ページを移動した後も継続的にエージェントに処理を行わせるためには，前のページを閉じる時（ページを移動することも含む）に，MiSpider エージェントの実行状態を保存・退避し，新しいページを読み込む時に初期化された MiSpider エージェントに実行状態を復帰するため手法が必要である．

MiSpider では，プロキシを通して Web ページにアクセスする．プロキシでは，Javascript の埋め込みやリンク先の書き換えなどを行う．埋め込まれた Javascript ファイルには，ページ読み込み時の処理と，閲覧ページ移動時の処理を記述する．そのための手続きは次の通りである．Web ページを読み込む時に，エージェントの内部状態を記述した XML ファイルを同時に読み込む．取得した XML 文章を構文解析して，Javascript オブジェクト kbase に変換する．kbase から得られた実行状態をエージェントの実行状態として再設定した後に，エージェントの処理を再開する．ページを閉じる時には，kbase を XML 文章に変換し，内部状態を XML ファイルとして保存する．

```
[JavaScript]
kbase.hoge_site.items[0].title = "item0";
kbase.hoge_site.items[0].url = "http://www.hoge.co.jp/item0.html";
kbase.hoge_site.items[1].title = "item1";
kbase.hoge_site.items[1].url = "http://www.hoge.co.jp/item1.html";
[XML]
<kbase>
 <knowledge>
  <name>hoge_site</name>
  <items>
   <item>
    <title>item0</title>
    <url>http://www.hoge.co.jp/item0.html</url>
   </item>
   <item>
    <title>item1</title>
    <url>http://www.hoge.co.jp/item1.html</url>
   </item>
  </items>
 </knowledge>
</kbase>
```

図 3　Javascript のオブジェクトと XML の対応例

エージェントの内部状態の表現に XML を用いることで，Javascript のオブジェクトや配列などのデータ構造を表現できる．XML の木構造の節はタグ名が複数形の時は配列，単数系の時はオブジェクトに対応する．木構造の葉は文字列として扱われる．Javascript のオブジェクトと XML の対応関係の例を図 3 に示す．図 3 の上側が Javascript によるオブジェクトの表現例で，下側が XML による表現例である．例えば，Javascript 上のオブジェクト kbase は，XML のルートノードを表している．Javascript 上の hoge_site は，XML 上の<name>hoge_site</name>に基づいて生成されている．XML 上の items タグは，item タグ内の内容を要素とする配列として Javascript 上で実現している．それぞれの item タグ内の内容 title, url は，Javascript 上のオブジェクトの持つ変数としている．

4. 時間保証型コンテンツ配信

コンテンツを時間に伴い配信するためにはコンテン

ツと配信時間の関係を管理する必要がある．本研究では，コンテンツとコンテンツの制御の対を番組と呼び，番組 p 次の 3 つ組で定義する．

$$p = \langle c, t, a \rangle$$

c はコンテンツを表し，t は $t = \langle t_s, t_e, t_c \rangle$ と表される．t_s と t_e は，それぞれ表示の開始時間と終了時間を表す．t_c は，表示時間に関する制約を表し，例えば，曜日による表示の可否を制約づけるために利用される． a は c の表示方法を制御するためのプログラムやデータを表す．コンテンツ配信者は，番組表 $P = \{p_1, p_2, p_3, ...\}$ を WisdomWeb サーバに渡すことで，コンテンツの表示時間や表示方法を制御する．

p に従って，WisdomWeb クライアントと WisdomWeb サーバ間で，コンテンツに関する制御をやりとりし，WisdomWeb クライアントが p に従い時間毎にコンテンツを切り換えることで，コンテンツ配信者が指定した時間にコンテンツが配信されているように見える．

4.1 番組表に基づくプッシュ配信

同時に WisdomWeb サーバにアクセスする WisdomWeb クライアント数が増えると，p を交換するための通信量が問題となる．HTTP や現在のインターネットの制約により，WisdomWeb サーバから WisdomWeb クライアントに対して接続要求を発することはできない．WisdomWeb サーバから WisdomWeb クライアントへの通信には，WisdomWeb クライアントから WisdomWeb サーバに対するポーリングが必要となる．このようなポーリングを多数の WisdomWeb クライアントから発した場合，WisdomWeb サーバは大量の接続要求を処理する必要があり，効率的ではない．

WisdomWeb エンジンで配信するコンテンツを制御するのではなく，番組表を WisdomWeb クライアントに送信し，コンテンツの配信制御を WisdomWeb クライアントで全て行うことで，WisdomWeb サーバと WisdomWeb クライアント間での通信を減らすことが可能になる．この場合，番組表の更新状態を WisdomWeb サーバと WisdomWeb クライアント間で共有する必要があるが，WisdomWeb クライアントからのポーリング時に更新状態を交換すればよい．これにより，番組表に特に更新がなければ，正確なプッシュ配信が可能になり，番組表に更新がある場合は最悪でもポーリング間隔の長さ程度の誤差でプッシュ配信を制御可能になる．

番組表を用いたプッシュ配信において，WisdomWeb サーバ上で行っていた処理を WisdomWeb クライアントに行わせることが可能になり，WisdomWeb サーバのスケーラビリティが向上する．

4.2 効果測定

効果測定を行うために，閲覧者の特定が有益である．閲覧者の特定のための ID などを番組表と同時に送ることが好ましい．番組表を拡張し，WisdomWeb クライアントに次のデータ $Q = \langle P, R \rangle$ を送ればよい．ここで，P は番組表，Q は効果測定用のデータで $Q = \langle id_{user} \rangle$ である．id_{user} は閲覧者の ID である．

コンテンツが閲覧されたことを計測する手法として，コンテンツを配信した回数をサーバ側で計測する手法が主流である．この方法では，Web ブラウザやプロキシサーバ上のキャッシュが閲覧された場合に，閲覧回数をカウントできないという欠点や，Web ブラウザ上に表示されたかどうかが不明であるという欠点がある．実際にはコンテンツが届かなかった場合や，Web ページの画面外の領域にコンテンツが表示された場合にもカウントされてしまう点が問題であり，正確な効果測定を妨げている．

WisdomWeb クライアントは，コンテンツが実際に表示されたときを検出して閲覧回数をカウントできる点が優位である．Web ブラウザ上で閲覧回数をカウントする欠点は，カウントした閲覧回数をサーバに送る必要がある点である．閲覧回数をサーバに送る前に，Web ブラウザが終了したり，別のページに切り替わると，閲覧回数をサーバに送ることができない．これを回避するために，閲覧回数を即座にサーバに送信するとサーバの負荷が問題となる．本研究では，WisdomWeb クライアントの継続的実行機能を利用することで，適切に閲覧回数をサーバに送ることが可能になる．WisdomWeb エンジンを利用することで，より正確な閲覧状況を確実にサーバに送信でき，正確な効果測定が可能になる．

5. 時間保証型コンテンツ配信システムの実装

WisdomWeb クライアントは Javascript で実装し，WisdomWeb サーバは Java で実装した．高頻度のポーリングを効率よく処理するために，パイプライン型の Web サーバ [10] として実装した．図 4 は，WisdomWeb エンジンのパイプライン構成を表している．

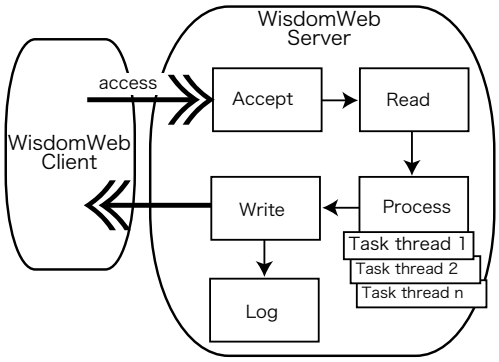

図 4　WisdomWeb エンジンのパイプライン構成

図 4 の四角はパイプラインの段を表しており，矢印の順にデータが流れる．WisdomWeb エンジンのパイプラインは 5 段階から構成されており，WisdomWeb クライアントからの接続要求は，前から順番に Accept, Read，Write，Process，Log 段階で処理される．Accept と Read と Write は，それぞれ接続要求の受付，通信受信，通信送信である．Process は WisdomWeb エンジン内の処理内容（WisdomWeb クライアントの配信，番組表更新チェック，そして効果測定用データなどの処理）によってスレッド処理により並行処理を行っており，この点が Choi らのアプローチ [10] とは異なる．Log は処理の実行ログを記録する．WisdomWeb サーバの CPU 個数の増加に伴いパイプラインの数を増やせば性能が向上する．

5.1　番組表更新機構

4.1 節で示したように，WisdomWeb エンジンと WisdomWeb クライアント間でのメッセージ通信を利用して番組表を交換することで，時間保証型のコンテンツ配信が達成される．番組表の更新チェックは，番組表に付与されたタイムスタンプを利用する．WisdomWeb クライアントは，ポーリング時に，WisdomWeb サーバへ番組表のタイムスタンプを送信する．

番組表が更新されていた場合，WisdomWeb サーバは，最新の番組表を WisdomWeb クライアントに送信する．WisdomWeb クライアントは，現在持っている番組表と新しい番組表を比較する．比較の結果，削除されたコンテンツを Web ページから削除する．

WisdomWeb クライアントは，ポーリングに対する WisdomWeb サーバからの反応が無い場合，ポーリング間隔を長くする．例えば，ポーリングに失敗した場合に，現在の 2 倍の間隔でポーリングする．これは，WisdomWeb サーバの過負荷を軽減させる効果がある．ポーリングの成功が続けば，ポーリング間隔を徐々に元に戻す．

5.2　効果測定機構

WisdomWeb クライアントは，収集した効果測定用のデータを定期的に WisdomWeb サーバに送信する．送信されるデータには，コンテンツの表示回数や表示時間などが含まれる．WisdomWeb クライアントは，コンテンツの表示回数や表示時間に関して，セッション中に過度にならない程度の頻度で更新する．

WisdomWeb サーバは，一定時間毎に効果測定用データを集計する．集計処理では，受信した表示回数や表示時間に関するデータから重複を取り除き，これまでのデータに加算する．データの重複除去処理を効果測定用データを受信するたびに行うのは無駄が多いので，一定時間毎にまとめて処理している．

5.3　評価：WisdomWeb エンジンの性能

WisdomWeb クライアント数と WisdomWeb サーバの処理能力の関係を調べた．本試験により時間保証の精度を見積もることができる．

1 台の計算機上に WisdomWeb サーバを実装し，10 台のパソコン上に WisdomWeb クライアントを実装し，これらの計算機をイーサネットや無線 LAN で相互接続した．サーバ用計算機は，PowerMac G5（2 個の PowerPC G5 2GHz と 1G のメモリ）である．クライアント用計算機は，iMacG5 1.6GHz(有線), PowerBookG4 1.5GHz (有線), eMac G4 1GHz (有線), eMac G4 1GHz (有線), iMac G4 1GHz(有線), PowerBook G4 1.3GHz (無線), iMac G5 1.6GHz(有線), eMac G4 1GHz (有線), and iBook G4 1.2GHz (無線) である．クライアント計算機上では複数の WisdomWeb クライアントを同時実行させた．

同時に実行させる WisdomWeb クライアントを 100 ずつ増やしながら，WisdomWeb サーバの処理性能を計測した．同時実行させる WisdomWeb クライアントの数は，10 台のクライアント用パソコン上で均等にした．それぞれの WisdomWeb クライアントは，ポーリング間隔を 0 秒で実行した．

図 5 は，WisdomWeb エンジンにおけるメッセージ通信の性能のグラフである．図 5 の横軸は，WisdomWeb クライアント数で，縦軸は WidomWeb サーバの性能で 1 秒あたりのメッセージ処理数を表している．WisdomWeb クライアント数が 100 の時に，最高性能 9647.8 回/秒のメッセージを処理している．Wis-

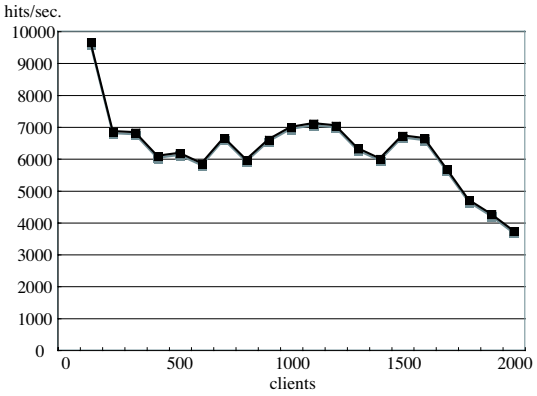

図 5　WisdomWeb エンジンのメッセージ通信性能

domWeb クライアント数が 200 から 1700 では，6000 〜7000 回/秒であり十分な性能であるといえる．それ以上多くなると性能が減衰していくが，既存の Web サーバの負荷分散技術により，適切に負荷を分散できる．本試験により，適切にサーバを分散すれば，ポーリング間隔を 2 秒程度に縮めても問題ないといえる．コンテンツ配信における時間保証は，2 秒以内の誤差で実現可能である．今回評価に用いた計算機は，サーバ用計算機ではないので，より高速な計算機をサーバ用に利用することでより正確な時間保証も可能になる．

6. おわりに

本論文では，紙のように自由にかつ簡単に扱え，さらに智を備えた Web を実現するための WisdomWeb エンジンを提案し，その応用として時間保証型配信システムを示した．WisdomWeb エンジンは，Web 上で動作するモバイルエージェントシステムである MiSpider を基盤として，Web ページに自由に書き込める機能などが拡張されている．WisdomWeb エンジンの特筆すべき点は，閲覧者にとって通常の Web ブラウザだけで動作する点であり，誰でもより簡単かつ直感的に情報発信できるようになる．

時間保証型配信においてコンテンツの表示時間を正確に制御するための，番組表に基づくプッシュ型コンテンツ配信機構を提案した．提案方式により，サーバの負荷をクライアントに分散することが可能になり，サーバのスケーラビリティを高めることが可能になった．実験により，Java による実装でも十分な性能の実装が可能であることを示した．また，プッシュ型配信のための効果測定について議論し，コンテンツの表示回数に関して実際に表示された回数を計測する手法を示した．これにより，より現実に即した形でのコンテンツの効果測定が可能になる．

今後の課題としては，プッシュ型配信を世界規模で行うためのアルゴリズムの開発が挙げられる．全世界規模でのプッシュ型配信を実現することで，従来とは本質的に異なる Web コンテンツの配信が可能になる．

文　献

[1] Y. Fukagaya, T. Ozono, T. Ito and T. Shintani, ”MiSpider: A Continuous Agent on Web Pages ,” WWW2005, pp.1008-1009, May. 2005.

[2] 大囿忠親　新谷虎松　深萱裕二郎　平岡佑介, “Web エージェント MiSpider におけるエージェント間協調,” 合同エージェントワークショップ&シンポジウム 2005(JAWS 2005), pp.386-393, 2005.

[3] S. Potter, J. Nieh, “WebPod: Persistent Web Browsing Sessions with Pocketable Storage Devices,” WWW2005, pp. 603-612. 2005.

[4] Tashiro, N., Hattori, H., Ito, T., and Shintani. T. 2004. Implementing a MiSpider agent based writable web for a dynamic information sharing system. In the Proc. Of the 13th World Wide Web Conference (WWW-2004), pp. 256-257, 2004.

[5] Julie E. Kendall and Kenneth E. Kendall, “Information delivery systems: an exploration of Web pull and push technologies,” Communications of the AIS, Vol. 1, No. 4es, pp. 1-43, 1999.

[6] H. Tanaka, E. Kawai, S. Yamaguchi, and H. Yamamoto, “Implementation issues of a push event delivery mechanism for web content service synchronized with TV programs.” In The 8th International Conference on Advanced Communication Technology (ICACT2006), 2006.

[7] Chen-Tung Chen and Wei-Shen Tai, “An information push-delivery system design for personal information service on the internet,” Information Processing and Management: an International Journal, Vol. 39, No. 6, pp. 873-888, 2003.

[8] 大囿 忠親, 深萱 裕二郎, 伊藤 孝行, 新谷 虎松, “Web エージェントシステム MiSpider における継続的実行について，” 日本ソフトウェア科学会第 22 回全国大会, CD-ROM, 2005.

[9] Ajax: a new approach to web applications http://www.adaptivepath.com/publications/essays/archives/000385.php

[10] G. S. Choi, J.-H. Kim, D. Ersoz, and C.R. Das, A Multi-Threaded PIPELINED Web Server Architecture for SMP/SoC Machines, In the Proc. of the 14th World Wide Web Conference (WWW2005), pp. 730-739, 2005.