モバイルコンピューティングとワイヤレス通信　20－15
(2002.　3.　8)

XMLコンテンツの差分生成法とプッシュ型配信への応用

上野 英俊　鈴木 偉元　石川 憲洋　加藤 剛志　角野 宏光　高橋 修

株式会社NTTドコモ マルチメディア研究所
〒239-8536 神奈川県横須賀市光の丘3-5

E-mail: {hueno,hideharu,ishikawa,t_kato,sumino,osamu}@mml.yrp.nttdocomo.co.jp

あらまし 我々は，任意のXMLコンテンツの差分生成法を提案する．本論文では，DOMツリーを応用した差分抽出アルゴリズムと，差分情報を表現するためのXSLTファイルの生成方法について述べる．更新後のコンテンツは，更新前コンテンツとXSLTファイルで表現された差分情報から自動的に生成することが可能である．本論文では，XMLコンテンツの更新がDOMツリーのルートに近い位置に加えられた場合や，全体のコンテンツサイズと比較して更新部分のサイズが小さい場合において提案方式が有効であることを論ずる．最後に，本方式のプッシュ型配信サービスへの応用について説明し，特に移動通信網に適用した場合について考察する．

キーワード XSLT (Extensible Stylesheet Language Transformations), XML (Extensible Markup Language), プッシュサービス, XPath, DOM (Document Object Model)

Delta file creation method of XML contents and its application to push services

Hidetoshi Ueno, Hideharu Suzuki, Norihiro Ishikawa, Takeshi Kato, Hiromitsu Sumino, Osamu Takahashi

NTT DoCoMo, Inc. Multimedia Laboratories
3-5, Hikarino-oka, Yokosuka, Kanagawa, 239-8536, Japan

E-mail: {hueno,hideharu,ishikawa,t_kato,sumino,osamu}mml.yrp.nttdocomo.co.jp

Abstract We propose delta file creation method of arbitrary two XML contents. This paper describes delta-extracting algorithm by applying the DOM tree, and XSLT file creation method in order to express delta information. It is possible to create the updated content from an old content automatically by using the proposed method. This paper shows that the proposed method is superior when the renewals are added to the close positions to the root of a DOM tree or the size of the delta part is small enough compared with its entire content size. We also discuss an application of the proposed method to push type information delivery services, especially in case of adapting it to mobile networks.

Key words XSLT (Extensible Stylesheet Language Transformations), XML (Extensible Markup Language), push service, XPath, DOM (Document Object Model)

1. はじめに

XML(Extensible Markup Language)[1] は，XHTML(Extensible Hypertext Markup Language)等の記述言語から，通信プロトコルのSOAP(Simple Object Access Protocol)等への様々な応用が考えられる拡張性の高いメタ言語である．また，モバイル向けアプリケーション技術の標準化団体であるWAP(Wireless Application Protocol)フォーラムでは，記述言語の標準仕様としてXHTML Basic[2]を採用しており，XML技術を移動通信においても利用する動きが加速している．

しかしXMLは，コンテンツ表現に要素や属性等の制御的な情報を含める必要があり，CSV(Comma Separated Value Format)形式と比較して一般的に表現したいコンテンツサイズが大きくなる傾向にある．またXHTMLコンテンツは，HTMLコンテンツと比較して，XML宣言や要素の終了を明示的に記す必要があるなど，一般的にそのサイズが大きくなる傾向にある．移動通信網は，第三世代(3G)携帯電話方式の登場により高速化してきているが，急激な需要増加が見込まれるため，限られた無線資源の利用効率を高めることは非常に重要である．そこで筆者らは，XMLコンテンツの差分情報を生成し，プル型やプッシュ型配信においてこの情報を用いることで転送データサイズの削減を試みた．

ネットワークを流れる転送データサイズを減らすための工夫として様々な方式が存在する．例えばHTTP(Hypertext Transfer Protocol)では，クライアントのキャッシュを有効活用するために，キャッシュが無効時のみデータ転送を行う機能を備える．キャッシュが無効な場合には，更新後データの全てを再転送する必要があるが，更新前後のコンテンツの変更は全体からみてわずかなデータ量であることから[3]，この差分情報のみを転送することでデータ量の減少が図れる．MPEG(Moving Pictures Expert Group)の画像データ圧縮や，TCP/IP及びRTP(Real-time Transport Protocol)のヘッダ圧縮がその応用例であり，その有効性は既に立証されている．HTTPにおいても同様の機能を提供するものとして，HTTPのデルタエンコーディング[4]が仕様化されており，旧コンテンツからの差分情報のみを転送するための機能が提供されている．XMLコンテンツにおいても，その差分情報を抽出し，HTTPデルタエンコーディング等を用いて差分情報のみを転送すれば，ネットワークを流れるデータ量の削減を図ることが可能であり，特に低速な移動通信網において非常に有効である．

本論文では，XMLコンテンツの差分生成法を提案し，特に移動通信網におけるプッシュ型配信に適用した場合について考察する．2節では，XMLコンテンツ差分生成における課題とその解決法を説明する．続いてXMLコンテンツの差分生成法の重要な部分であるXMLコンテンツの差分抽出法を3節に，差分情報ファイルの生成手順について4節に述べる．5節では，提案方式に関する評価と考察を行い，6節に提案方式の移動通信網におけるプッシュ型配信への応用ついて述べる．

2. XMLコンテンツ差分生成の課題と解決法

更新前後のXMLコンテンツの差分を生成するためには，1)差分情報の抽出方法，2)差分情報の表現フォーマット，3)差分情報から更新後XMLコンテンツの生成手順，という3つの課題がある．我々は，これらの課題を解決するにあたり，XSLT(Extensible Stylesheet Language Transformations)[5]を採用することにした．

XSLTは，XML文書の構造変換機能を提供するための仕様で，あるXMLコンテンツに対して特定範囲だけを指定した体裁で表示･出力することや，構成要素の並べ替えを行うことが可能である．この技術を応用することにより，更新前コンテンツから更新後コンテンツへの差分情報を，XSLTを用いた変換ルールとして記述することが可能である．そして，更新前コンテンツにこのファイルを適用するソフトウェアを用いて，自動的に更新後コンテンツを生成できる．

XSLTの採用により上記2)と3)の課題が解決できるため，本論文では，残る課題1)の解決法を提案する．提案するXMLコンテンツの差分生成法は，a)更新前後のXMLコンテンツの差分情報を抽出し，b)その抽出情報からXSLT差分情報ファイルを生成する，という2つの手順に分かれ，それぞれ3節と4節にて述べる(図1)．

なお，上記課題1)の簡単な解決法として，編集するXMLコンテンツに対して，ユーザの編集操作の記録を元に，差分情報を抽出する方法が考えられる．この方式では，XMLコンテンツそのものと，その編集記録を併せ持つ必要があり，あるコンテンツが2人のユーザによって別々に更新された時に，この

別々に更新された2つのコンテンツの差分を抽出することが不可能であるという問題がある．そこで本論文では，編集履歴を利用せずに，任意に与えられた2つのXMLコンテンツの差分生成法を提案する．これは，編集記録を保持する必要が無く汎用的である．

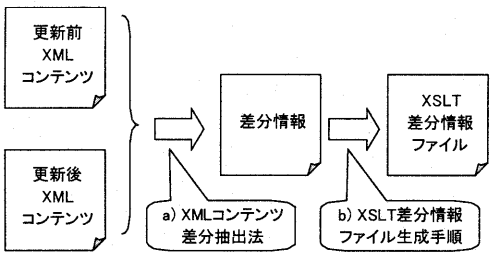

図1 XMLコンテンツ差分生成法の流れ

3. XMLコンテンツ差分情報抽出法

提案方式では，XMLをツリー(木)構造に例えて表現することが可能なDOM(Document Object Model)ツリー[6]を応用する．更新前後コンテンツのツリー(DOMツリーを以下では単にツリーと表現する)をルート(根)から順番に走査して比較することによって，更新前後のコンテンツの比較が可能である．

任意の2つのツリーから差分情報を抽出する方法として以下の2種類を検討した．

- 簡易的な差分情報の抽出方法(本論文では，「簡易差分抽出法」と定義)．既存方式であり，3.1節にて説明する．
- 最適化された差分を抽出する方法（本論文では，「最適差分抽出法」と定義)．本論文にて提案する方式であり，3.2節にて説明する．

3.1 簡易差分抽出法（既存方式）

簡易差分抽出法は，更新前後のツリーのルートから，更新前ツリーの各ノード(節)を基準に，更新後ツリーの対応ノードを順次比較し，更新前ツリーを走査し比較を進める過程で，ノードの不一致があったノードのサブツリー(部分木)を全て差分とみなす方式である．以下では，具体例を用いて簡易差分抽出法の処理手順を説明する．図2は，XMLの要素とその値だけで構成した，単純な更新前後のXMLコンテンツの例であり，図3は，そのツリー表現である．

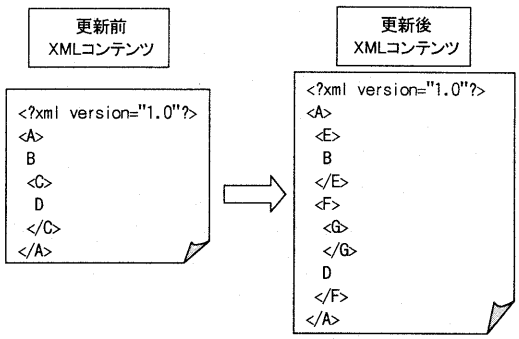

図2 XMLコンテンツ例

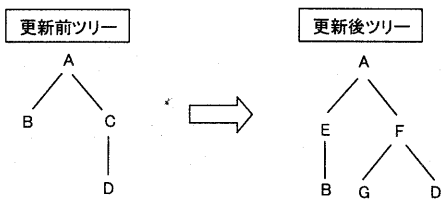

図3 XMLコンテンツのツリー表現

簡易差分抽出法では，図3の更新前コンテンツの(A,B)と更新後コンテンツの(A,E,B)の部分に着目すると，更新前コンテンツから(B)が削除され，新たに(E,B)が追加されたと判断する．つまりこの方式では，あるノードでの差分を検出すると，その子孫は全て差分とみなす．従って，ブランチ(分枝節)へのノードの挿入，削除や変更を正確に検出できず，図3の例では，(E)の挿入を検出できない．

文献[7][8]は，任意のXMLコンテンツの差分生成について説明しているが，いずれも簡易差分抽出法と同様に最適な差分情報の検出が不可能であり，いずれの論文においてもこの問題点については一切述べられていない．

3.2 最適化差分抽出法（提案方式）

最適化差分抽出法は，簡易的差分抽出法における課題であった差分情報の正確な抽出が可能な方式である．それは，ツリーをルートから順に比較し，あるブランチの更新を検出したとしても，その子孫に差分が無ければそのブランチに対してのみの更新であったことを検出することができる．以上のことから，本論文において最適とは，更新前後のツリーにおいて，検出した一致ノード数が最多であることと定義する．

以下では，最適化差分抽出法について具体例を用

いて説明する．その手順は以下の通り．

① ルートの解釈(3.2.1節)
② ノード組み合わせパターンの作成(3.2.2節)
③ 一致・差分ノードの確定(3.2.3節)

3.2.1 ルートの解釈

更新前後ツリーのルートは，必ず対応するものとして一致か変更のいずれかであると考える．実際には，削除や追加もあり得るがそれらも変更と見なす．図3では，いずれのルートも(A)で一致している．

3.2.2 ノードの組み合わせパターンの作成

更新前後ツリーにおける全ノードを各項目と考えた組み合わせの全パターンを求める．この時，選択対象にノードを一つも選択しない「選択無し」という項目を含め，さらにルートは項目から除外する．更新前のノード数を p，更新後のノード数を q とすると，それぞれのルートを除いた(p+q-2)個のノードから「選択無し」も含めた組み合わせパターンを求めるので，その数は以下の数式で表現される．図 3 の例では，p=4，q=6 であるので，全部で表 1に示す 256 通りのノード組み合わせパターンが得られる．

$$\text{ノードの組み合わせパターン数} = \sum_{k=0}^{p+q-2} {}_{p+q-2}C_k$$

表1 ノードの組み合わせパターンの一覧

項番	更新前ツリーのノード	更新後ツリーのノード
1	選択無し	選択無し
2		B
3		D
4		E
5		F
6		G
…		…
27		BDEF
28		BDEG
29		BDFG
30		BEFG
31		DEFG
32		BDEFG
33 ～ 64	B	…(16 通り)
65 ～ 192	(D,BC,BD,CD の 4 通り)	…(それぞれ 16 通り)
193 ～ 224	CD	…(16 通り)
225	BCD	選択無し
226		B
227		D
228		E
229		F
230		G
…		…
251		BDEF
252		BDEG
253		BDFG
254		BEFG
255		DEFG
256		BDEFG

3.2.3 一致・差分ノードの確定

ノードの組み合わせパターン(表 1)から一組を選択する．選択した組に含まれるノードを更新前後ツリーそれぞれから全て削除した新しいツリーを生成し，これが同じかどうかを比較する．図 4は，表 1 の項番 27 の情報を元に作成した新しいツリーである．図 4では，更新前ツリーは選択無しなので何も削除せず，更新後ツリーから(B,D,E,F)を削除したものとを比較した結果，一致していない例を示している．図 5は，更新前ツリーから(C)を削除したものと，更新後ツリーから(E,F,G)を削除したものが一致している例を示している．このように新しいツリーが一致した場合には，ノードの組み合わせパターンから選択したノードの集合が差分ノードであり，それ以外が一致ノードと考えることができる．図 5では，これを分かりやすく区別するため，一致ノードと差分ノードをそれぞれ○と□で囲んでいる．なお，この場合の一致ノード数は 3 である．

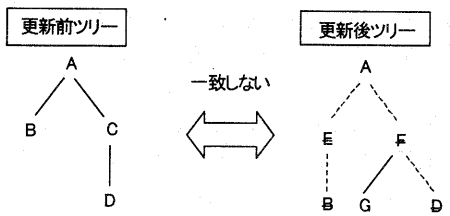

図4 新しいツリーが一致しない場合

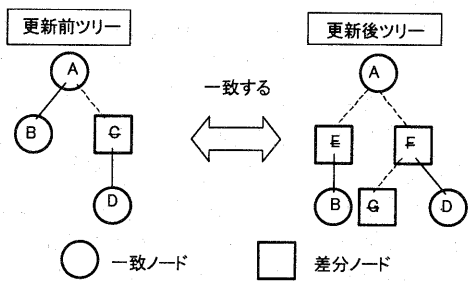

図5 新しいツリーが一致した場合

ところで，新しいツリーが一致するケースは複数存在する場合もある．例えば，更新前後ツリーのいずれもノードB以外をすべて削除するような組み合わせを選択した場合にも新しいツリーは一致するが，この場合一致ノード数は1であるため最適なものであるとは言えない．

図6は，本節に説明した一致・差分ノード確定までのフローを示す．

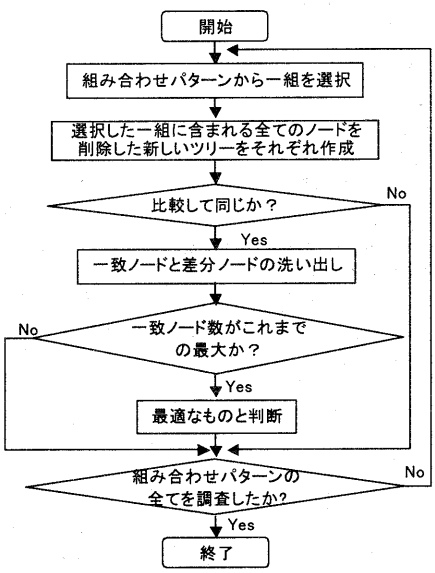

図6 一致・差分ノード確定までのフローチャート

以上の手順に従って求められた一致ノードと差分ノードは，簡易差分抽出法と異なり，あるブランチに対する更新を検出しても，そのブランチの子孫の差分情報を正しく抽出できるため，最適な差分情報といえる．

4. XSLT差分情報ファイル生成手順

本節では，3.2節で抽出された差分情報から，XSLT差分情報ファイルを生成する方法について説明する．生成手順は以下のとおり．

① 差分情報の生成(4.2節)
② 差分情報のマージ(4.3節)
③ XSLT差分情報ファイルの生成(4.4節)
④ XSLT差分情報ファイルの決定(4.5節)

4.1 差分情報項目の整理

3.2節で抽出した個々の差分ノードは，削除，変更，追加・挿入の3種類の差分種別で表現可能である(表2)．それぞれの項目は，単一ノードの場合と部分ツリー等の複数ノードの集合である場合がある．なお，追加とはリーフ(葉)の子ノードの追加であり，挿入はブランチの子ノードへの挿入のことを示し，変更の位置によって呼び方が異なる．また，XSLT差分情報ファイルを生成するために必要な情報は，この他に基準ノード，相対位置，ノード情報がある．これらは差分種別によって必要な情報が異なる(表2)．

表2 差分種別と必要な情報項目

差分種別	基準ノード	相対位置	ノード情報
削除	削除位置	基準ノードと同一	(不要)
変更	変更位置	基準ノードと同一	変更情報
追加・挿入	追加・挿入の位置	基準ノードからの相対関係	追加・挿入情報

4.2 差分情報の生成

3.2節で抽出した個々の差分ノードそれぞれついて，4.1節に説明した差分種別に当てはめ，差分情報を生成する．差分情報の生成は，まず i)一致ノードで挟まれた差分ノードの差分情報を生成し，その次に ii)一致ノードで挟まれていないノードに対する差分情報の生成を行う．

i)では，差分ノードを全て削除した一致ノードのみで構成するツリーを新たに作成し，その親子ノードのペアに着目して以下の照合を行う．まず更新前後ツリー両方に対して，この親子ペアの間に削除した差分ノードが存在していたかどうか調べる．削除した差分ノードが更新前後ツリーの両方に存在していた場合には，この差分ノードは異なる値を持つはずなので，更新前ツリーの差分ノードから，更新後の差分ノードへの変更であると判断する．いずれか片方のみに削除した差分ノードが存在していた場合には，この差分ノードは更新前ツリーからの削除，もしくは更新後ツリーへの挿入であると判断する．

例えば，図5の例において，更新前後ツリーの(A,D)の部分に着目すると，更新前ツリーにおける差分ノード(C)と，更新後ツリーにおける差分ノード(F)が更新前後いずれのツリーにも存在するため，これは(C)から(F)への変更であると判断できる．同様に，(A,B)の部分については，更新前ツリーには何も存在しないため，更新後ツリーにノード(E)が挿入されたものと判断できる．以上により得られた差分情報を表3に示す．表中の基準ノードを表現するために用いている XPath[9]は，XML 文書中の特定位置を指定す

るための仕様であり，本提案では，基準ノードの位置を指定するために用いている．

表3 一致ノードにはさまれたノードの差分情報

差分種別	基準ノード	相対位置	ノード情報
変更	ノードCのXPath	基準ノードと同一	タイプ：要素 名前：F
挿入	ノードAのXPath	第一子ノード	タイプ：要素 名：E

上記 ii)では，上記 i)で差分情報を生成しなかった残りのノードに対する差分情報を生成する．残りのノードとは，すなわち一致ノードで挟まれていないノードのことを示すので，ここで生成される差分情報は，リーフの削除か追加のみである．

図 5の例では，ノード(G)が追加されたものとして判断されるため，表 4に示す差分情報が得られる．

表4一致ノードにはさまれていないノードの差分情報

差分種別	基準ノード	相対位置	ノード情報
追加	ノードCのXPath	第一子ノード	タイプ：要素 名前：G

4.3 差分情報のマージ

4.2節で得られた差分情報は，差分ノード毎に生成されているが，これを基準ノードに着目することにより差分情報のマージが可能である．マージすることにより XSLT のテンプレートに同一のものが使えることになるため，その数を減らすことができる．例えば，表 3と表 4のノード(C)に対する差分情報をマージして表 5に示す差分情報の生成ができる．

表5 マージ後の差分情報

基準ノード	差分種別	相対位置	ノード情報
ノード C の XPath	変更	基準ノードと同一	タイプ：要素 名前：F
	追加	第一子ノード	タイプ：要素 名前：G

4.4 XSLT 差分情報ファイルの生成

4.2節と4.3節で得た差分情報を基に XSLT 差分情報ファイルを生成する．表 3，表 4及びそれらをマージして得た表 5の差分情報から実際に生成した XSLT 差分情報ファイルの例を図 7に示す.以上により更新前XMLコンテンツから更新後XMLコンテンツへの差分情報を表現した XSLT 差分情報ファイルの作成が完了した．

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match= "/A" >
  <xsl:copy>
   <E>
    <xsl:apply-templates>
   </E>
  <xsl:copy>
 </xsl:tempale>
 <xsl:template match= "/A/C" >
  <F>
   <G>
   </G>
   <xsl:apply-templates>
  </F>
 </xsl:tempale>
 < xsl:template match = " @* | node() " >
  < xsl:copy >
   < xsl:apply-templates select = " @* | node() " />
  </ xsl:copy >
 </ xsl:template>
</xsl:stylesheet>
```

図7 XSLT 差分情報ファイルの例

4.5 XSLT 差分情報ファイルの決定

3.2節にて一致ノード数が最大となるツリーを求めた結果，一致ノード数が最大なものが複数存在する場合がある．例えば(X)という同じ値を持つノードが多数使われていたコンテンツの場合に，どれを一致ノードとみなすかによって複数の選択肢が得られる場合がある．この場合，いずれを選んでも一致ノード数が最大であるので，本論文での「最適」という定義は満たすが，以下に述べる 3 つの点に着目することで，さらに最適な 1 つの XSLT 差分情報ファイルに絞り込むことができる．

- 差分情報の複雑度に着目

差分情報が複雑である程，差分情報から生成する XSLT のテンプレートとその生成手順が複雑なものとなる．従って，差分情報の複雑度を定義し，この値が大きいほど差分情報が複雑であるとし，複雑度が最も小さいものを選択する．本論文では，追加・挿入の場合，3 種類の情報が必要であることから，この複雑度を 3 と定義する．削除や変更についても同様に，複雑度をそれぞれ 1，2 とする(表 6).

表6 差分情報の複雑度

分類	複雑度	必要な情報
削除	1	・XSLT テンプレートの適用 (<xsl:apply-templates />)
変更	2	・変更後のノード内容 ・XSLT テンプレートの適用
追加・挿入	3	・基準ノードをそのまま出力する内容 ・追加・挿入するノードの内容 ・XSLT テンプレートの適用

- コンテンツサイズに着目

生成した XSLT 差分情報ファイルのサイズが最小のものを最適とする.

- XSLT のテンプレート数に着目

差分情報のマージ(4.3節)を実施した結果，テンプレート数が最小であるものを最適とする.

5. 提案方式の評価と考察

本節では，提案方式による効果を明確にし，主にコンテンツサイズに着目した場合の提案方式の評価と考察を行う.

5.1 提案方式による効果

提案方式は，既存方式と異なり，ブランチへの挿入や削除，変更等の更新に対処可能である(表 7). 一般に XML コンテンツの更新は，XML の任意の位置において発生する可能性があるので，提案方式は，ブランチの更新に対応するという点で優れている.

表7 既存方式と提案方式の比較

分類	リーフ		ブランチ	
	既存	提案	既存	提案
追加(リーフ)・挿入(ブランチ)	○ (対応可)	○	× (対応不可)	○
削除	○	○	×	○
変更	○	○	×	○

5.2 コンテンツサイズの評価

XML コンテンツは，あらゆるパターンが考えられるため，本アルゴリズムの有効性をあらゆるコンテンツにおいて証明することは難しい. しかしいくつかの例に対して提案方式を適用し，その有効性の傾向を確かめることは可能である. 我々は，様々なコンテンツ例において本方式を適用した結果，本提案アルゴリズムが有効なコンテンツには以下の特徴があることを確認した.

- ツリーのルートに近い位置で更新があるデータ
- 全体のコンテンツサイズが大きくそれに対して更新部分がわずかなデータ.

以下具体例を用いて説明する. ある 449 バイトの XHTML Basic のコンテンツ(図 8)において，<p>要素の親要素として新たに<h2>要素を挿入した場合について考える. このコンテンツに提案方式を適用した場合，304 バイトの XSLT 差分情報ファイル(図 9)が生成され，元のコンテンツと比較してサイズが減少している. また，簡易差分検出法を用いた場合には，471 バイトの XSLT 差分情報ファイル(図 10)が生成され，元のコンテンツと比較して逆にサイズが増加している. 簡易差分検出法を用いた場合には，挿入された<h3>要素の下位の<p>要素やその下位の<a>要素が削除され，同じものが新たに追加されたものとして判断されるためである.

以上の例では，単純なコンテンツ例を用いたが，ツリーのレベル(深さ)がより深いコンテンツを用いた場合や，全体のコンテンツサイズがさらに大きく，それと比較して更新部分サイズが十分に小さい場合には，得られる差分情報ファイルと更新後コンテンツとのサイズ差がさらに大きくなることが考えられる. 逆にリーフの更新のように提案方式による利点が出にくい場合には，提案方式と簡易差分抽出法により得られるデータサイズの差が小さい場合もある. また大幅なデータ更新が加えられた場合には，更新後コンテンツのサイズよりも XSLT 差分情報ファイルサイズが逆に大きくなってしまう場合もある. 以上のことから，更新後コンテンツのファイルサイズと XSLT 差分情報ファイルサイズ比較することで，更に適したコンテンツを選択することができる.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-
basic10.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml"
              xml:lang="en" >
<head>
 <title>XHTML Basic Content Exapmle</title>
</head>
<body>
 <p>
  <a href= "http://www.example1.com/inde.html" >This is the
link for the big money! Cliek!</a>
  <a href= "http://www.example2.com/inde.html" >This is the
link for the big fortune! Cliek!</a>
 </p>
</body>
</html>
```

図8 XHTML Basic で記述したコンテンツ例

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match= "/html/body" >
    <h3>
      <xsl:apply-templates>
    </h3>
  </xsl:tempale>
  < xsl:template match = " @* | node() " >
   < xsl:copy >
    < xsl:apply-templates select = " @* | node() " />
   </ xsl:copy >
  </ xsl:template>
</xsl:stylesheet>
```

図9 最適化差分抽出法による XSLT 差分情報ファイル

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match= "/html/body" >
    <h3>
     <p>
      <a href= "http://www.example1.com/inde.html" >This is
the link for the big money! Cliek!</a>
      <a href= "http://www.example2.com/inde.html" >This is
the link for the big fortune! Cliek!</a>
     </p>
    </h3>
  </xsl:tempale>
  < xsl:template match = " @* | node() " >
   < xsl:copy >
    < xsl:apply-templates select = " @* | node() " />
   </ xsl:copy >
  </ xsl:template>
</xsl:stylesheet>
```

図10 簡易差分抽出法による XSLT 差分情報ファイル

6. プッシュ型配信への応用

筆者らは，オリジンサーバにおけるコンテンツの更新を契機として，プッシュ型配信によりクライアントにコンテンツを提供する方式を提案している[10]．この方式は，即時的なコンテンツ提供が可能であり，さらに更新された差分情報のみをプッシュするためにネットワーク利用効率にも優れている．

プッシュ型配信サービスでは，プッシュするコンテンツがクライアントにとって本当に必要な情報かどうかサーバ側での判断が難しいという問題がある．コンテンツがクライアントにとって必要とされなかった場合には，ネットワークリソースの浪費となるため，プッシュデータのサイズをできる限り小さくすることが課題となる．さらに移動通信網に適用した場合には，先述したようにネットワークを流れるデータ転送量をできるだけ小さくすることが重要であるため，本提案方式は特に移動通信網において有効であるといえる．また以上の観点から，4.5節においてコンテンツサイズが一番小さい XSLT 差分情報ファイルを選択することは，移動通信網で利用する場合の最良のコンテンツ選択であるといえる．

7. まとめ

本論文では，任意の XML コンテンツの差分生成法を提案し，これまでの方式よりも優れた差分情報を生成することが可能であることを示した．提案方式は，コンテンツの更新が XML ツリーのルートに近い位置に加えられた場合や，全体のコンテンツサイズと比較して更新部分のサイズが小さい場合において有効であるという傾向を，XHTML Basic のコンテンツ例を挙げて示した．さらに本研究では，提案方式をプッシュ型配信に応用する場合について，特に移動通信網に適用した場合にはそのコンテンツサイズが重要であることを説明した．今後の研究では，一般的な XML コンテンツにおいて本提案の有効性を定量的に証明することが課題となる．

文　献

[1] Tim Bray, et al., "Extensible Markup Language (XML) 1.0 (Second Edition)," W3C Recommendation, Oct 2000.

[2] Mark Baker, et al., "XHTML Basic," W3C Recommendation, Dec 2000.

[3] Jeffrey C. Mogul, et al., "Potential benefits of delta-encoding and data compression for HTTP," Proceeding of SIGCOMM 97, Sep 1997.

[4] Jeffrey C. Mogul, et al., "Delta Encoding in HTTP," RFC3229, Jan 2002.

[5] James Clark, "XSL Transformations (XSLT) Version 1.0," W3C Recommendation, Nov 1999.

[6] Arnaud Le Hors, et al., "Document Object Model (DOM) Level 2 Core Specification Version 1.0," W3C Recommendation, Nov 2000.

[7] Robin La Fontaine, "A Delta Format for XML: Identifying Changes in XML Files and Representing the Changes in XML," XML Europe 2001, May 2001.

[8] F. P. Curbera, D. A. Epstein, "Fast Difference and Update of XML Documents", XTech'99, March 1999.

[9] James Clark, et al., "XML Path Language (XPath) Version 1.0," W3C Recommendation, Nov 1999.

[10] 上野英俊 他, "移動通信におけるプッシュプロトコルの提案と評価," 情報処理学会マルチメディア，分散，協調とモバイルシンポジウム(DICOMO2001),Vol. 2001, No.7, page 217-222, June 2001.