情報処理学会論文誌 Vol. 50 No. 1 209–221 (Jan. 2009)

XEUS：携帯電話向けXML文書符号化方式

小林亜令[†1] 村松茂樹[†1] 西山智[†1]

本稿では，携帯電話環境における利用を想定したXML文書符号化方式“XEUS（Xml document Encoding with Uniformed Sheet）”を提案する．従来のXML文書符号化方式は，符号圧縮により伝送符号量を低減させても，受信側では復号処理に加えパース処理も行うため，逆に負荷が大きくなってしまい，携帯電話のように処理能力の限定された環境においては，必ずしも有効とはいえない．この課題を解消するために，本稿では，XML文書のスキーマ情報を含む符号化テーブルを，あらかじめ別ファイルとして定義し，符号化時にパース処理と符号圧縮処理の両方を行う方式“XEUS”を提案する．本方式は，伝送符号を高圧縮するだけでなく，従来復号側で処理していたパース処理やデータ型変換処理も符号化時に行うため，携帯電話上で，低消費メモリによる高速復号が可能となる．本稿では，性能測定実験により他方式と比較し本方式の有効性を考察するとともに，本方式を適用した商用事例についても紹介する．

XEUS: XML Document Encoding Method for Cellular Phone

Arei Kobayashi,[†1] Shigeki Muramatsu[†1]
and Satoshi Nishiyama[†1]

We propose an XML document encoding method for cellular phone. Conventional methods can compress XML document at the encoding server, but they need not only parsing but also decoding at the cellular phone, so processing time become longer. Our method uses the predefined encoding table for the schema of XML document as another file. Therefore, our method can not only compress XML document but also parse and translate data type, so we enabled higher compression ratio and faster decoding than other methods. Moreover, we introduce a commercial example using our method.

†1 株式会社KDDI研究所
KDDI R&D Laboratories Inc.

1. はじめに

近年，業務システム等，PCをクライアントとしたWeb情報システム構築の際には，開発コストの低減，汎用性や拡張性の確保を目的とし，XML形式のデータを流通データとして用いるケースが一般的となっている．XMLは，加工性，拡張性，可読性の利点を持つ反面，テキストデータのため冗長性が高く，携帯電話網のような伝送速度が限定された環境では符号圧縮を行うことが望ましい．

日本の携帯電話サービスは1987年に誕生して以来，急速に普及が進み2008年1月現在，1億台以上の端末が出荷されている[1)]．また近年では，携帯電話端末の高機能化が進みCPU，メモリの高性能化をはじめ，BREW[2)]やJavaといったアプリケーション実行環境の整備により，携帯電話を情報システムのクライアントとして利用可能になりつつある．そこで今後，携帯電話からPC向けに構築したWeb情報システムを利用するニーズが高まることが予想される．

従来のXML文書圧縮法は，符号圧縮により伝送符号量を低減させることは可能だが，受信側ではパース処理に加えて，復号処理も行う必要が生じるため，負荷が大きくなる．そのため符号圧縮によって伝送符号量を低減させても，システム全体のパフォーマンスが改善するとは限らない．

本稿では，この課題を解消するために，任意のXML文書を対象とし，そのスキーマ情報を含む符号化テーブルをあらかじめ別のXML文書として定義し，符号化時にパース処理と符号圧縮処理の両方を行う方式“XEUS（Xml document Encoding with Uniformed Sheet）[3),4)]”を提案する．本方式は，たとえば，RSS（RDF Site Summary）[5)]やSVG（Scalable Vector Graphics）[6)]といった1つの文書種別に対して1つの符号化テーブルを定義するため，同一文書種別においては，符号化対象文書に依存しない．また本方式は，要素の親子関係や属性の従属関係，値のデータ型で構成されるスキーマ情報を用いて，符号長や符号辞書を定義するため，符号化時に高圧縮するだけでなく，文節分解や構造解析，データ正当性検証といったパース処理や，受信側の処理に適したデータ型変換も行うことができ，受信側において，高速復号が可能となる．

本稿では，2章で関連研究について述べ，3章で携帯電話向けXML文書符号化方式に対する要求条件を述べる．次に4章で，提案方式の特徴やアーキテクチャを示し，5章で，性能測定システムの実装，6章で性能測定実験結果を示す．さらに7章で考察したうえで，本方式の適用事例を紹介し，8章で結論付ける．

2. 関 連 研 究

XML はネットワークを介した流通データの形式として利用されており，伝送や格納の効率化を目的として，HTTP1.1 の gzip のような汎用的な圧縮方式に加えて，XML 文書の特徴を利用したより効率の良い圧縮方式が提案されている．XMill [7] や XComp [8] は，(1) 要素名/属性名の符号化テーブル，(2) 木構造情報，(3) 要素値，属性値のコンテナに分解し，gzip により圧縮する．xmlppm [9] は，Prediction by Partial Match と呼ばれる手法で木構造を圧縮する．GPRESS [10] は，符号化対象は GML に固定されているが，座標値を効率良く圧縮することにより XMill 以上の圧縮率を実現している．

また，圧縮後のデータに対する高速なクエリ処理を目的として，XGrind [11]，XPRESS [12]，XQueC [13] および XQZip [14] が提案されている．これらの方式は，圧縮率についてはいずれも XMill と同等以下となっているが，圧縮後のデータに対して直接ランダムアクセスを行うことにより，高速なクエリ処理を実現している．

本稿の提案方式は，文書の圧縮による伝送の効率化のみを目的とするのではなく，携帯電話というリソースの限られた環境で動作するアプリケーションに対して，XML 文書の情報をツリー構造のルート要素から順に高速に提供することを目的とする．したがって，復号後に XML 文書をパースするという手順ではなく，復号結果をアプリケーションに提供可能な形式とする．また，圧縮後のデータに対するクエリ処理は目的としていない．

Natchetoi ら[15] は，ビジネスアプリケーションを想定し，クライアントがあらかじめ XML 文書の構造を知っていることを利用した圧縮方式を提案している．この方式は，XML 文書のサイズが小さい場合でも圧縮率が高く，簡単な復号によりアプリケーション用のデータを得ることが可能だが，符号化対象文書の文書構造が一定でない場合には，文書ごとに符号化辞書を送る必要が生じるため，符号圧縮率の点で効率的ではない．OMA [16] で策定された WBXML [17] は符号化時にパース処理も行うことが可能な方式であるが，エンコーダ，デコーダとは別に符号化テーブルを定義することができないため，XML 文書種別ごとにエンコーダ，デコーダを用意しなければならず，任意の XML 文書を対象とした符号化方式とはいえない．本稿の提案方式は，任意の XML 文書を符号化対象としており，その種別ごとに，そのスキーマ情報を用いた符号化テーブルをあらかじめ別ファイルとして定義する．その結果，任意の XML 文書を対象とした高圧縮，高速復号が可能であり，同様の方式はこれまで提案されていない．

3. 携帯電話向け XML 文書符号化方式に対する要求条件

携帯電話網を利用したデータ伝送や，携帯電話でデータを受信して，パース処理や数値演算，描画等のアプリケーション処理を行うために，携帯電話向け XML 文書符号化方式は，以下の要求条件を満たすことが求められる．

条件①：伝送符号量の低減

前述のとおり，XML 形式のデータはテキストデータのため，冗長性が高く，伝送効率が低いため，携帯電話網のような伝送速度が限定された環境では符号圧縮を行い，伝送符号量を低減させることが望ましい．

条件②：データ受信側処理時間の低減

符号化方式を導入することにより，携帯電話上の復号処理時間が増大してはならない．

条件③：復号処理時のメモリ使用量の低減

携帯電話は，実行時に使用可能なメモリ量が限定されているため，復号処理時のメモリ使用量が増大してはならない．

後述する性能評価実験では，上記要求項目に関して従来法との性能比較を行っている．

4. 提 案 方 式

4.1 提案方式の概要

本章では，前章の要求条件を満たし，課題を解消する方式 “XEUS” を提案する．XEUS は，任意の XML 文書を対象とした符号化方式であり，XEUS シートと呼ぶ，あらかじめ別の XML 文書として定義された符号化テーブルを用いることを特徴とする．

XEUS シートは，符号化対象 XML 文書種別における要素の親子関係や各要素に付随する属性リスト，要素値の有無，値のデータ型といったスキーマ情報と，各要素名/属性名に対する符号語，要素値/属性値の符号長/符号語といった符号化テーブルの両方を定義可能であり，本シートを用いることにより，符号化対象文書を，各要素や属性を符号化するために必要最低限の符号長で符号化できる．また，XEUS シートは同じ文書種別内では符号化対象文書に依存しないため，符号化結果に XEUS シートを添付する必要がない．そのため，データの冗長性が低くなり，高圧縮が期待できる（条件①）．さらに，符号化時に XEUS シート内のスキーマ情報を用いて，パース処理やデータ型変換処理を行うことが可能なため，復号側の処理時間（条件②）やメモリ使用量（条件③）の低減が期待できる．

また，XEUS シートを XML 形式とすることにより，XEUS シート自体も XEUS 符号化

することができるため，デコーダは，あらかじめ XEUS 符号化された XEUS シートを用いる．これにより，XEUS シートの復号処理と，符号化結果の復号処理を共有することができるため，デコーダの構造の複雑化やコードサイズの肥大化を抑えることが可能となる．

一方，XEUS が符号化対象とするデータモデルは XML Information Set [18] であり，符号化前の XML 文書を受信側で正確に復元することはできない．これは，XEUS が想定するアプリケーションは SVG や RSS といった XML 文書の高速処理（描画等）であり，XML 文書を空白，改行，属性順等まで含めて正確に伝送することは目的でないからである．

4.2　提案方式の処理フロー

XEUS は，符号化対象文書の文書種別ごとに，XEUS シートを符号化対象文書とは別の XML 形式のファイルとして定義する．そしてエンコーダ，デコーダの双方は，XEUS シートを，あらかじめ保持していることを前提とする．なお，デコーダの XEUS シートはあらかじめ XEUS 符号化しておく．

図 1 のとおり，エンコーダは，まず，符号化対象文書をパースし，符号化対象文書の名前空間を識別した後，対応する XEUS シートを参照しながら，ノードのツリー構造と要素名，属性名，要素値，属性値の符号化を行う．その結果，エンコーダは，本方式独自のバイナリデータを符号化結果として生成し，携帯電話網を通じて伝送する．デコーダは，符号化結果を受信し，そのヘッダ部から符号化時に用いた XEUS シートを識別し，同じシートを参照しながら復号処理を行う．その結果，元の XML 文書を出力するのではなく，データの正当性検証，文書構造解析，アプリケーション（ブラウザ等）が処理しやすいデータ型への変換を行った後のデータ（API）を出力する．

4.3　要素名，属性名の符号化規則

本方式は，符号化対象文書種別（名前空間）のスキーマ情報に適した符号長，符号語に則って，要素名，属性名の符号化を行う．以下の例を用いて具体的な符号化規則を示す．ある符号化対象文書種別が，**図 2** に示す DTD の構造を持つとすると，本方式では，**表 1** のような符号化テーブルを定義する．

スキーマ情報を用いない符号化方式と異なり，本方式では，子要素を符号化するために必要最短の符号長定義を行い，子要素内に限定した一意の符号を割り当てる．表 1 の例では，<要素 C> に 1 bit，<要素 F><要素 G><要素 H> に 2 bit 割り当て，<要素 F> に 00，<要素 G> に 01，<要素 H> に 10 を割り当てる．その結果，表 1 の <要素 C> と <要素 F> のように，文書内の別の要素名に対して同じ符号語が割り当てられるが，符号化テーブルを参照することにより，正しく復号することができる．

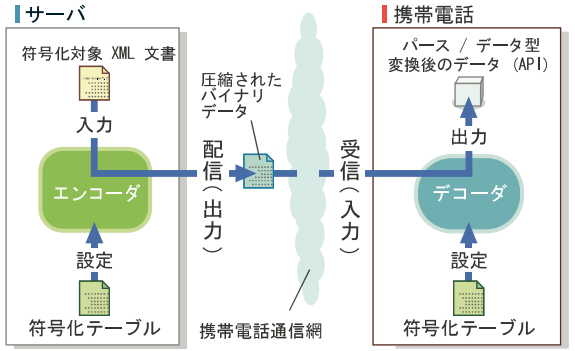

図 1　提案方式の概要図
Fig. 1　Outline of proposed method.

```
<!ELEMENT 要素 A (要素 C)*>
<!ATTLIST 要素 A 属性 b CDATA #IMPLIED>
<!ELEMENT 要素 C (要素 F | 要素 G| 要素 H)*>
<!ATTLIST 要素 C 属性 d CDATA #IMPLIED>
<!ATTLIST 要素 C 属性 e CDATA #IMPLIED>
<!ELEMENT 要素 F EMPTY>
<!ELEMENT 要素 G ENPTY>
<!ELEMENT 要素 H EMPTY>
<!ATTLIST 要素 H 属性 i CDATA #IMPLIED
                 属性 j CDATA #IMPLIED
                 属性 h CDATA #IMPLIED>
```

図 2　文書構造例（DTD）
Fig. 2　Example of document structure (DTD).

属性については，XML1.1 の仕様上，1 つの要素に対して，同じ属性が複数存在した場合，最初に記述された属性のみを要素に紐付け，2 番目以降に記述された属性は無視されることと，1 つの要素における異なる属性の出現順は影響しないことから，1 bit による符号化が可能となる．表 1 の例では，全属性名の符号長が 1 bit であり，存在する場合 0，存在しない場合 1 の符号語を割り当てる．

さらに，この符号化テーブルは符号化対象文書の名前空間内で共通であり，同じ名前空間

表 1　要素名，属性名の符号化テーブル例
Table 1　Example of encoding table for element name and attribute name.

要素名，属性名	符号長[bit]	符号語
<要素 A>	0	-
属性 b	1	0 | 1
<要素 C>	1	0
属性 d,e	1	0 | 1
<要素 F>	2	00
<要素 G>	2	01
<要素 H>	2	10
属性 i,j,k	1	0 | 1

内であれば符号化対象文書に依存しない．よって符号化テーブルを別ファイルに分離することが可能であり，符号化結果に添付する必要がない．

4.4　要素値，属性値の符号化規則

本方式は，あらかじめ符号化テーブルに定義された要素値，属性値のデータ型，符号長，セパレータに則って符号化を行う．データ型は，大きく数値型，列挙型，文字列型の 3 種類を定義できる．4.3 節に示した例の構造をもとに生成した符号化対象文書例を用いて，具体的な符号化規則を示す（**図 3**）．

●数値型

- unsigned integer（符号なし整数型）
- integer（整数型）
- fixedpoint（固定小数点型）
- double（浮動小数点型）
- BCD（2 進符号化 10 進数符号型）
- coodinates（座標値型）

の 5 種類のデータ型を定義可能である．さらに，値の符号化に適した符号長と個数を符号化テーブルに定義する．たとえば属性値 b のような場合は，unsigned integer 型の 4 bit で個数は 1 と定義する．属性値 d のように，値が整数列の場合は，integer 型の 8 bit で個数は任意（implied）とし，セパレータとしてカンマを定義する．属性値 e のように，値が先頭に 0 を許容する数値（例：電話番号，カード番号）の場合は，1 つの数値を 4 bit で符号化する BCD（Binary Coded Decimal）型と定義する．属性値 k のように，値が整数の座標値

```
<要素 A　属性 b="1">
  <要素 C　属性 d="1,2,3"
           属性 e=" 0000111122223333">
    <要素 H 属性 k="0,8 16,12"
            属性 i="item1">aaa</要素 H>
  </要素 C></要素 A>

各属性値の特性：
・属性値 b は 1 個の 1〜10 の整数
・属性値 d は整数-100〜100 の羅列
・属性値 e は数の羅列（カード番号）
・属性値 k は座標値の羅列
・属性値 i は item1 か item2 のどちらか
```

図 3　符号化対象文書例
Fig. 3　Example of encoding document.

列の場合は，前座標値からの差分値（2 つ前の数値からの差分）を符号化する coordinates 型と定義する．

●列挙型

属性値 i のように，値が列挙型（item1，item2 のどちらか）の場合は，データ型を列挙型とし，選択肢を符号化するための必要最低限の符号長（属性値 i の場合は 1 bit）と符号語を割り当てる．

●文字列型

要素値 H のように，値が文字列の場合は，データ型を文字列型とし，文字コード（要素値 H の場合は Shift_JIS）と符号長（要素値 H の場合は implied）を定義する．

●要素値，属性値の符号長の符号化規則

要素値，属性値の符号化テーブルで，数値の個数や文字列の符号長が任意と定義された値（属性値 d，e，k，要素値 H）は，符号長が符号化対象文書に依存するため，符号化結果に値の符号長を添付する．符号長が符号化テーブルにより固定化できるもの（属性値 b，i）については，値の符号長を符号化結果に添付しない．

4.5　XEUS シートの構造

前述した符号化規則に則った符号化に必要な符号化テーブルとスキーマ情報を XEUS シー

```
<xeus version="2.0" xmlns="#xeussheet-sample" >
<head><root name="要素 A"/></head>
<body>
 <element name="要素 A">
  <attlist><attr name="属性 b" ><value>
    <number data="UI" bit="4" qt="1" />
  </value></attr></attlist>
  <children bit="1">
   <child_element name="要素 C" code="0"/>
  </children>
 </element>
  …..
 <element name="要素 H">
  <attlist><attr name="属性 k">
   <value separator=", ">
    <number data="coordinates" bit="8" qt="implied"/>
   </value></attr>
   <attr name="属性 i">
    <value><choice bit="1" qt="1">
     <list code="0">item1</list>
     <list code ="1">item2</list>
    </choice></value></attr></attlist>
  <elemet_value>
   <char encoding="Shift_JIS" length="implied"/>
  </element_value></element></body></xeus>
```

図 4　XEUS シート例
Fig. 4　Example of XEUS sheet.

トに定義する．**図 4** に 4.3〜4.4 節の例をもとにした XEUS シートの一部を示す．

XEUS シートの <head> でルート要素を定義し，<body> で要素，属性の符号化テーブルを定義する．<element> の name 属性で要素名を定義し，その子要素 <attlist> 以下で出現しうる属性群を定義する．

<attr> の name 属性で属性名を定義し，数値型の属性値については，<value> の子要素

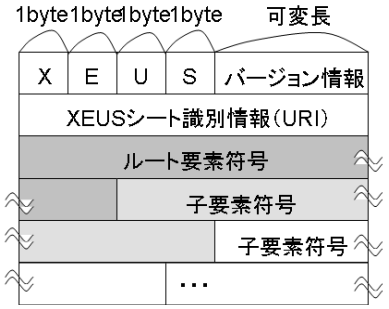

図 5　符号化結果のデータ構造
Fig. 5　Data structure of encoding result.

<number> で，データ型（data），符号長（bit），個数（qt）を定義する．列挙型の属性値については，<value> の子要素 <choice> で符号長（bit），個数（qt）を定義する．さらにその子要素 <list> で値の候補と符号語（code）を定義する．文字列型の属性値については，<value> の子要素 <char> でエンコード方式（encoding），文字列の符号長（length）を定義する．

また <element_value> で，要素値の符号化テーブルを定義し，<children> の bit 属性で子要素名の符号長を定義し，<child_element> の name 属性で子要素名を，code 属性で符号語を定義する．

4.6　符号化結果のデータ構造

前述した符号化規則に則り，符号化を行った結果，生成されるバイナリデータの構造を以下に示す．

符号化結果のデータ構造は，**図 5** のとおり，ヘッダ部とボディ部に分けられ，ヘッダ部に符号化時に参照した XEUS シート識別情報として URI を格納する．ボディ部には，符号化対象文書のルート要素から要素の出現順に，各要素における要素名，属性名，属性値，要素値の符号化結果を，可変長の要素符号として格納する．

要素符号を詳細化した構造を**図 6** に示す．要素符号は，要素名・属性名部，属性値部，要素値部の 3 つに分けられ，要素名・属性名部には，各要素の符号開始識別符号を示す要素開始符号，各要素符号が占める符号長を示す要素占有符号長符号，XEUS シートに定義された要素に対する符号語を示す要素名符号，XEUS シートに定義された各属性に対して 1 bit の存在有無識別フラグを示す属性存在有無フラグ符号が格納される．

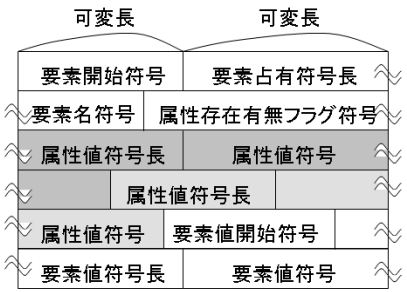

図 6　要素符号のデータ構造
Fig. 6　Data structure of element code.

表 2　符号化結果例
Table 2　Example of encoding result.

符号	内容
00000000	要素 A の開始符号
00011001	要素 A の占有符号長(27byte)
10000000	要素 A の属性有無判定フラグ符号
00000001	属性 b=”1”
00000000	要素 C の開始符号
00010111	要素 C の占有符号長(23byte)
11000000	要素 C の属性有無判定フラグ符号
00000011	属性値 d の属性値符号長(3byte)
0000…0011	属性 d=”1,2,3”
00000100	属性値 e の属性値符号長(4byte)
0000…0011	属性 e=”0000111122223333”
00000001	要素 H の開始符号
00000111	要素 H の占有符号長(11byte)
10100000	要素 H の属性有無判定フラグ符号
00000000	属性値 i=”item1”
00000100	属性値 k の属性値符号長(4byte)
0000…100	属性 k=”0,8 16,12”
00000011	要素値符号長(3byte)
01100001x3	要素値 H=”aaa”

属性値/要素値部には，各属性値/要素値符号が占める符号長を示す要素値/属性値符号長，XEUS シートに定義されたデータ型や符号長に則って符号化された符号語を示す属性値/要素値符号が格納される．

4.4 節の符号化対象文書を，4.5 節の XEUS シートにより，符号化した結果を**表 2** に示す（表 2 の例では，各符号に対してバイト整列を施している）．

符号化対象文書のサイズが 143 byte，符号化結果のサイズが 29 byte（どちらもヘッダ部は除く）であり，提案方式によって符号量が低減されることが分かる．

5. 評価システムの実装

5.1 XEUS エンコーダ

XEUS エンコーダは，C++言語でコーディングを行い，Linux 上で動作可能なソフトウェアとして，実装した．XEUS エンコーダの処理フローを**図 7** に示す．まず，①エンコーダ起動時限定の処理として，符号化対象文書の名前空間に対応した XEUS シートを読み込む．次に符号化対象文書を SAX パース[19] し（Xerces-C 2.6.0[20] を利用），XEUS シートを参照しながら，②要素ごとに，③要素名，④属性存在有無判定フラグ，⑥属性値，⑦要素値の符号化を行う．その後，⑧子要素が存在すれば，②に戻り，子要素の符号化処理を行う．子要素が存在しなければ，⑨要素占有符号長の算出を行う．全要素の符号化を完了したところで，全体を gzip で圧縮し，処理を終了する．

5.2 XEUS デコーダ

XEUS デコーダは，C 言語でコーディングを行い，携帯電話向けアプリケーション実行環境である BREW3.1.2Ja 上で，動作可能なソフトウェアとして実装した．XEUS デコーダの処理フローを**図 8** に示す．

まず，①デコーダ起動時の限定の処理として，復号対象文書の名前空間に対応した XEUS シートを読み込む．この際に，XEUS シートも XML 文書であるため，XEUS シートの名前空間に該当する XEUS シートを，本方式の仕様として固定化し，XEUS シートも XEUS で符号化したデータとして読み込むことにより処理の高速化を図っている．復号処理は，受信データの一部を読み込み，gunzip により解凍し，③要素開始符号の出現順に，XEUS シートを参照しながら，④要素名，⑤属性名，⑦属性値，⑧要素値の復号処理を行う．手順⑧が終了するごとに，手順④で復号された各要素の占有符号長と復号した符号長との比較を行い，「手順④の占有符号長 > 復号したデータの符号長」であれば，次要素は子要素であり，「手順④の占有符号長=復号したデータの符号長」であれば，次要素は兄弟要素であると判

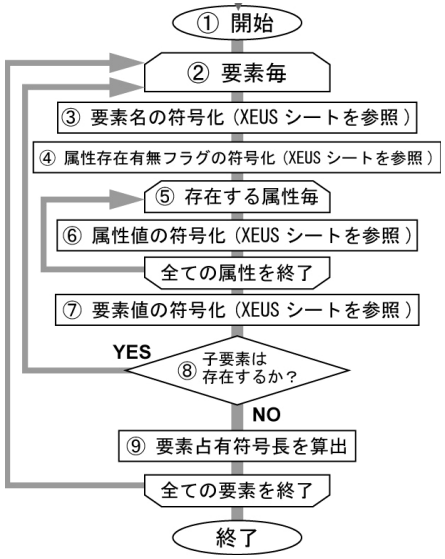

図 7　XEUS エンコーダの処理フロー
Fig. 7　Processing flow of XEUS encoder.

断し，ノードのツリー構造を復号する．全要素の復号処理を完了したところで，処理を終了する．

本方式は，本来データ受信側で処理する必要があった，パース処理や，外部アプリケーションの処理に適したデータ型変換を，符号化時に処理するため，復号処理負荷の低減が期待できる．

6. 性能評価実験結果

評価項目は，符号圧縮率，符号化処理時間，復号処理時間，復号処理時のメモリ使用量，の 4 項目とした．

6.1 評価環境

評価データには，SVG，RSS の 2 種類のデータ群（10 KB-1 MB，240 サンプル）を用いた．**表 3** のとおり，SVG は数値，RSS は文字列の比重が高い．

また本評価では，XEUS エンコーダ，PC 上で動作する XEUS デコーダの比較対象方式として，gzip，XMill（bzip2 モード），xmlppm を使用し，携帯電話上で動作する XEUS デコーダの比較対象方式として，gunzip+expat [21] を使用している．デコーダの比較対象に XML パーサを用いる理由は，携帯電話上で実行可能な他方式のデコーダが存在しないことと，他方式では，復号後にパース処理を必要とすることである．

XEUS エンコーダ，gzip，XMill，xmlppm は，CPU: Pentium4 2.8 GHz，Memory: 2 GB，OS: RedHat Enterprise Linux ES ver.3 の PC 上で動作させた．また，XEUS デコーダ，expat は，前記 PC に加えて，au 携帯電話 W41T（ARM9 150 MHz，BREW3.1.2Ja）上で動作させた．

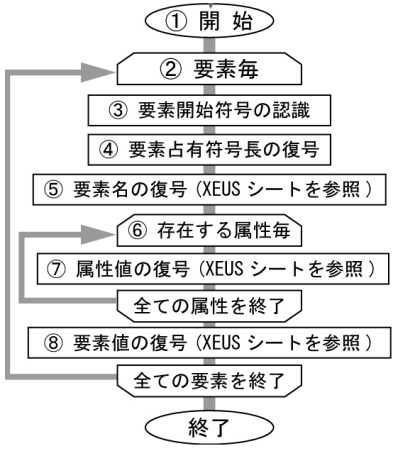

図 8　XEUS デコーダの処理フロー
Fig. 8　Processing flow of XEUS decoder.

表 3　実験データの特性
Table 3　Feature of experiment data.

	マークアップ[%]	数値[%]	文字列[%]
SVG	26.3	69.5	4.2
RSS	25.6	2.5	72.0

6.2 符号化処理性能測定結果

符号圧縮率を**図 9**，**図 10** に示す．符号圧縮率は，以下の式で算出した．

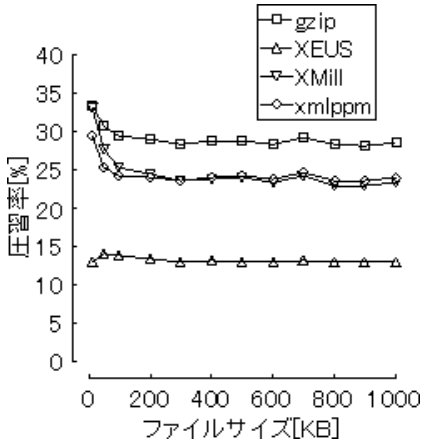

図 9　符号圧縮率の比較（SVG）
Fig. 9　Comparison of compression ratio (SVG).

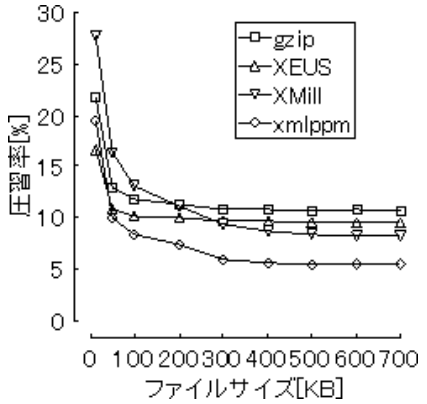

図 10　符号圧縮率の比較（RSS）
Fig. 10　Comparison of compression ratio (RSS).

$$\text{符号圧縮率 [\%]} = \frac{\text{符号化後のバイナリサイズ}}{\text{符号化前の XML 文書サイズ}} * 100$$

図 9 から，SVG については，提案方式が gzip の約 2 倍の圧縮性能を持ち，他方式と比べても 10%程度の性能優位性が見られる．これは，符号化対象文書のスキーマ情報に適した符号化テーブルを定義できるためと考えられる．特に SVG では座標値がデータに占める割合が大きいため，coordinates 型が優位性につながっている．また，提案方式は，10 KB 程度の軽量な符号化対象文書に対しても，圧縮率が低下しない傾向がある．これは，符号化テーブルを別ファイルとして定義し，符号化結果に添付する必要がないことが要因と考えられる．一方 RSS は，数値が少なく，文字が大部分を占めるため，ファイルサイズが大きい場合には，xmlppm に対して性能が劣るが，一般的な RSS のファイルサイズである 50 KB では他方式とは同等レベル以上の性能であるといえる．

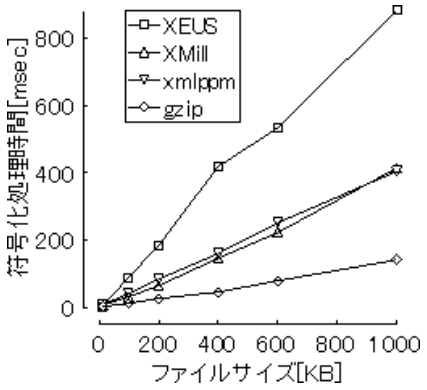

図 11　SVG の符号化処理時間
Fig. 11　Encoding process time of SVG.

次に符号化処理時間の測定結果を**図 11**，**図 12** に示す．図から，提案方式は他方式と比べ，符号化処理時間が長いことが分かる．これは，本方式が符号化時に，本来受信側で行うパース処理やデータ型変換処理を行っているためと考えられる．

6.3　復号処理性能測定結果

本方式が対象とするデバイスは携帯電話であるが，携帯電話上で動作可能な方式は gunzip+expat のみであるため，他方式との相対的な性能比較を行うために，まず PC を用いた復号処理時間の測定を行った．測定用には，要素数，属性数，数値の個数，文字列数をカウントする外部アプリケーションを作成し，復号処理時間は，復号対象データを読み込み，復号（パース）を行い，カウントを完了するまでとした．測定結果を**図 13**，**図 14** に示す．図から，提案方式の復号処理性能が，他方式よりも相対的に高いことが分かる．ただ，どの方式についても 1 秒以下と処理時間が短く，gunzip+expat については，提案方式と比べ

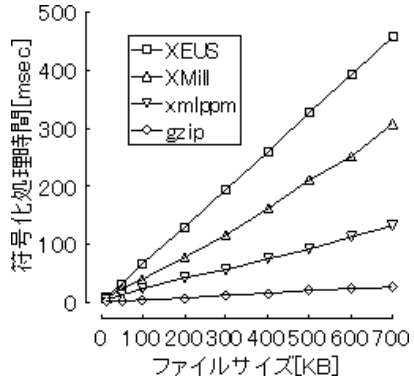

図 12　RSS の符号化処理時間
Fig. 12　Encoding process time of RSS.

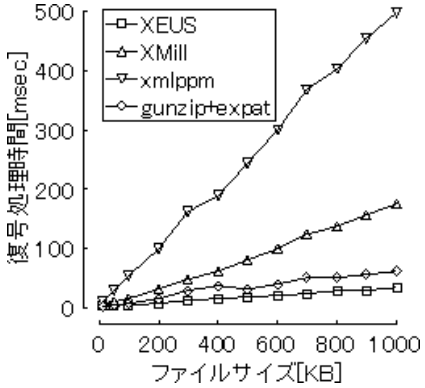

図 13　PC における復号処理時間の比較（SVG）
Fig. 13　Comparison of decoding process time using PC (SVG).

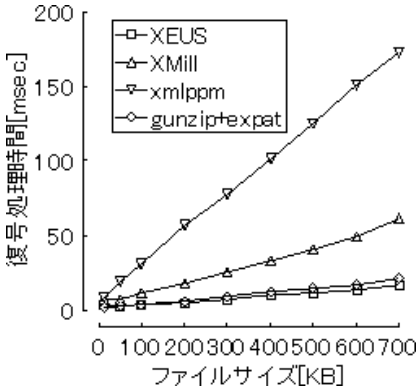

図 14　PC における復号処理時間の比較（RSS）
Fig. 14　Comparison of decoding process time using PC (RSS).

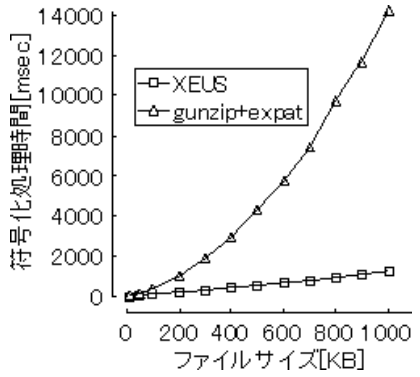

図 15　携帯電話における復号処理時間の比較（SVG）
Fig. 15　Comparison of decoding process time using cellular phone (SVG).

て，大きな性能差は見られていない．これは CPU の処理やヒープメモリのアクセス速度が高速であるため，提案方式の特徴であるパース処理やデータ型変換の処理負荷低減が，処理時間に大きな影響を与えないためと考えられる．

次に携帯電話を用いた復号処理時間の測定結果を**図 15**，**図 16** に示す．XMill，xmlppm については，携帯電話上で動作不可能であることから，gunzip+expat のみを比較対象とした．図から，100 KB 以上のデータの場合において，本方式は他方式と比べ，優位性が見られる．特にファイルサイズが大きい場合に，処理時間の差が大きくなる．これは，携帯電話は PC と異なり，CPU の処理速度やヒープメモリのアクセス速度が低速であるため，提案方式の特徴が大きな処理時間の差につながっていると考えられる．

また，復号処理時のメモリ使用量の計測結果を，**図 17**，**図 18** に示す．メモリ使用量は，

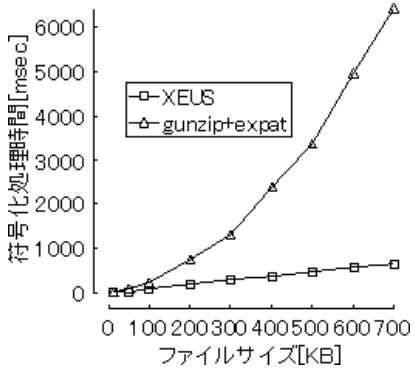

図 16　携帯電話における復号処理時間の比較（RSS）
Fig. 16　Comparison of decoding process time using cellular phone (RSS).

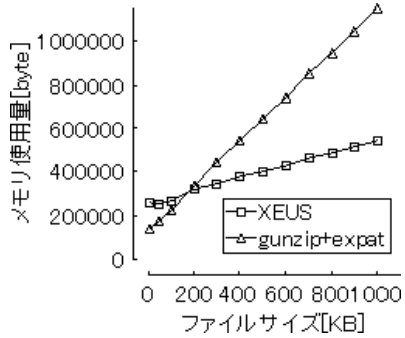

図 17　SVG のメモリ使用量
Fig. 17　Memory consumption volume of SVG.

デコーダ（パーサ）の実行コードサイズと復号処理時の最大メモリ使用量の和とした．提案方式の実行コードサイズは 37.2 KB であり，gunzip+expat の実行コードサイズは 107.9 KB である．提案方式の実行コードサイズが小さいのは，gunzip+expat は，XML パースの機能を実装する必要があるのに対して，提案方式は，XML パースを符号化時に実行するため，デコーダに実装する必要がなく，より軽量な XEUS の符号化規則を実装するのみであることが主な要因と考えられる．

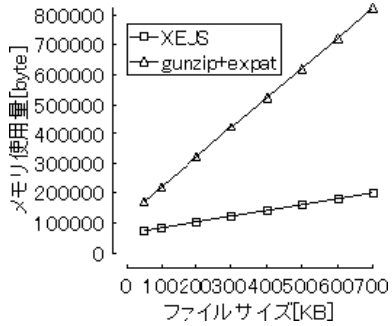

図 18　RSS のメモリ使用量
Fig. 18　Memory consumption volume of RSS.

また，図から，本方式のメモリ使用量は，gunzip+expat と比べ，概ね優位性が見られ，SVG の場合は，1,000 KB のデータで 2 倍程度の差が得られ，RSS の場合は，700 KB のデータで 4 倍程度の差となっている．これらの結果は，本方式が，パース処理やデータ型変換処理を符号化時に行っていることが主な要因と考えられる．

XEUS デコーダにおける XEUS シートの解析処理は，デコーダ起動時に 1 回のみ実行される処理であるため，図 15，図 16 の復号処理時間には含めていない．また，SVG の XEUS シート解析時間は 590 msec，RSS の XEUS シート解析時間は 40 msec であり，復号処理性能を大きく低下させる処理ではないと考える．

6.4　システム全体の処理時間の比較

携帯電話を用いた XML 文書の伝送システムは，①携帯電話から XML 文書取得要求を発行する，②XML 文書配信サーバが XML 文書の符号化を行う，③符号化された XML 文書を伝送する，④携帯電話上で受信したデータを復号する，の 4 つの処理で構成される．そこで，これら 4 つの処理時間の合計値を測定することにより，システム全体の処理時間の比較を行った．比較結果を**図 19**，**図 20** に示す．図から，提案方式を用いることにより，gunzip+expat と比べて，4–5 倍程度高速であることが分かる．図 11，図 12 で示したとおり，提案方式は，他方式と比べ，符号化処理時間が長くなるが，図 15，図 16 で示したとおり，復号処理時間が大幅に短いため，システム全体の処理時間の低減につながっている．また，図 13，図 14 に示したとおり，xmlppm や XMill の復号処理時間は，gunzip+expat と比べ，相対的に長いことから，提案方式は xmlppm や XMill と比べ，システム全体の処理時間について，優位性があると推測できる．

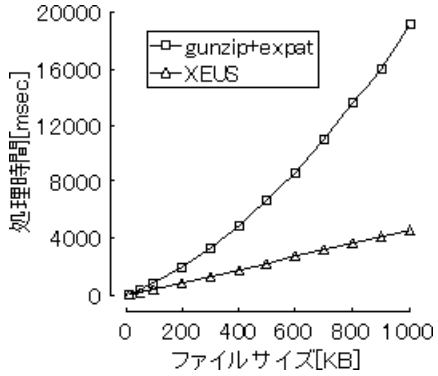

図 19　システム全体の処理時間（SVG）
Fig. 19　Processing time of whole system (SVG).

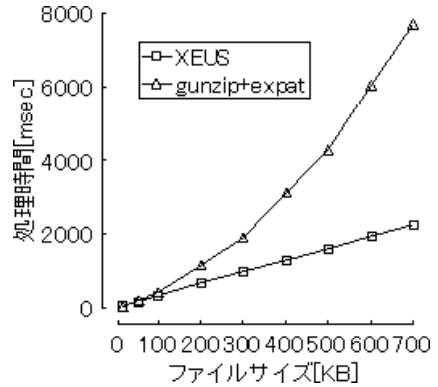

図 20　システム全体の処理時間（RSS）
Fig. 20　Processing time of whole system (RSS).

7. 考察と適用事例

前章から，本方式の優位性は示されたが，他にも概念的な有効性が存在する．

まず，符号化時に XEUS シートをサブセット化し，符号化対象文書のうち，XEUS シートに定義されているノードのみを符号化することができる．これは，サーバ側でマスタデータを管理し，クライアントにはその一部分のみを開示したいケースに有効である．

その反対に，符号化時に用いた XEUS シートから，サブセット化された XEUS シートを復号処理用として渡すことにより，文書の一部分のみを復号することができる．これは，閲覧権限等によって，復号範囲を変えたいケースに有効である．また，復号時に，値が固定化されている等，内容を解析する必要がない要素については，読み飛ばすことも可能であるため，復号処理の高速化につながる．

さらに，符号化時と復号時の符号化テーブルを別のタグセットに変えることにより，符号化前の XML 文書と異なる文書構造を持つ XML 文書に復号することができる．これは，多言語対応等のケースに有効である．

また，名前空間を用いて，既存のスキーマ A からスキーマ A+B に仕様拡張された場合，本方式では，XEUS シートも同様に，スキーマ A 向けの XEUS シートは改変せず，名前空間を用いて，スキーマ A+B の XEUS シートを定義することができる．そして，本方式は，スキーマ A 向けの XEUS シートで符号化された文書も，スキーマ A+B 向けの XEUS シートで符号化された文書も復号処理可能である．

これらの利点は，符号化テーブルが XML 文書であることと，復号時に要素の占有符号長を用いて，読み飛ばすことが可能であること，が要因である．

提案方式は，すでに「EZ ニュースフラッシュ[22)]」という au 携帯電話向け RSS コンテンツのマルチキャスト配信サービスに商用利用されている（**図 21**）．本サービスは，限られた伝送帯域の中で，大量の RSS コンテンツ配信を行う必要があるため，符号圧縮率が非常

図 21　EZ ニュースフラッシュ（画面例）
Fig. 21　EZ news flash (Captured image of screen).

に重視される．また，端末起動中は，定期的に配信される RSS コンテンツを常時待ち受け画面上に提示する必要があるため，高速復号，低消費メモリといった点も重視される．よって，本サービスにおける RSS コンテンツ符号化方式に提案方式が採用され，2008 年 7 月現在，1000 万人以上のユーザに利用されている．

8. おわりに

本稿では，携帯電話で復号処理を行うことを想定した XML 文書符号化方式 “XEUS（Xml document Encoding with Uniformed Sheet）” を提案した．本方式は，符号化対象文書のスキーマ情報に適した符号化テーブルを別の XML 文書として定義することを特徴としており，名前空間内で共通化できる部分を符号化テーブルに定義し，符号化対象文書に依存する部分を符号化結果に格納することによって，高圧縮が可能となった．さらに，符号化時にパース処理やデータ型変換を行うことにより，受信側（携帯電話）の復号処理負荷を低減することが可能となった．本稿では，性能評価実験により他方式と比較し，優位性を確認した．本方式は，概念的な有効性も備えており，様々な適用先が想定され，商用事例も存在する．今後は，国際標準化等普及促進に向けた取り組みを行う予定である．

謝辞　日頃ご指導いただく KDDI 研究所秋葉所長に深謝いたします．また，熱心に議論してくださった ATR 認知情報科学研究所井ノ上所長，KDDI 高木氏，田中氏，KDDI 研究所松本氏に感謝いたします．本研究の一部は，独立行政法人情報通信研究機構からの委託研究「ユビキタス ITS」に基づき行われたものである．

参考文献

1) TCA. http://www.tca.or.jp/japan/database/daisu/yymm/0801matu.html
2) BREW. http://brew.qualcomm.com/brew/ja/
3) 小林亜令，高木　悟，村松茂樹，井ノ上直己：XML 文書汎用符号化方式「XEUS」，信学技報 DE2001-9, Vol.101, No.110, pp.65–72 (2001).
4) 小林亜令，松本一則，井ノ上直己：汎用 XML 文書符号化方式「XEUS」の性能評価，情報科学技術フォーラム，pp.99–100 (2003).
5) RSS. http://web.resource.org/rss/1.0/spec
6) SVG. http://www.w3.org/Graphics/SVG/
7) Liefke, H. and Suciu, D.: XMill: An efficient compressor for XML data, *Proc. 2000 ACM SIGMOD*, pp.153–164 (2000).
8) 井川甲作：DTD を用いた XML 文書圧縮アルゴリズムに関する研究，東京工業大学工学部情報工学科卒業論文 (2000).
9) Cheney, J.: Compressing XML with Multiplexed Hierarchical Models, *Proc. 2001 IEEE Data Compression Conference*, pp.163–172 (2001).
10) Guan, J. and Zhou, S.: *GPress: Towards Effective GML Documents Compresssion*, pp.1473–1474 ICDE (2007).
11) Tolani, P. and Haritsa, J.R.: XGRIND: A Query-friendly XML Compressor, *18th Int. Conf. on Data Engineering*, pp.225–234 (2002).
12) Min, J.K., Park, M.J. and Chung, C.W.: XPRESS: A queriable compression for XML data., *SIGMOD'03*, pp.122–133 (2003).
13) Arion, A., Bonifati, A., Costa, G., d'Aguanno, S., ManoLescu, L. and Pugliese, A.: XQueC: *Pushing queries to compressed XML data*, VLDB (2003).
14) Cheng, J. and XQzip, W.Ng.: Querying compressed XML using structural indexing., *EDBT'04*, pp.219–236 (2004).
15) Natchetoi, Y., et al.: A Context-Dependent XML Compression Approach to Enable Business Applications on Mobile Device, *Euro-Par* (2007).
16) OMA. http://www.openmobilealliance.org/
17) Wireless Application Protocol Forum: Binary XML Content Format Specification Ver.1.3 (2001).
18) XML Information Set. http://www.w3.org/TR/xml-infoset/
19) SAX. http://www.saxproject.org/
20) Xerces. http://xerces.apache.org/xerces-c/
21) expat. http://www.jclark.com/xml/expat.html
22) EZ ニュースフラッシュ．http://www.au.kddi.com/ezweb/service/news_flash/

(平成 20 年 3 月 27 日受付)
(平成 20 年 10 月 7 日採録)

小林　亜令（正会員）

1973 年生．1996 年北海道大学工学部電子工学科卒業．1998 年北海道大学大学院工学研究科修士課程修了．同年 KDDI 株式会社（当時 KDD）入社．現在，（株）KDDI 研究所特別研究員．XML，SVG，通信放送融合技術，センサデータ解析技術，ITS 等の研究開発に従事．情報処理学会学会誌 SWG 幹事．

村松　茂樹（正会員）

1997 年東京大学工学部電気工学科卒業．1999 年東京大学大学院工学系研究科電子情報工学専攻修士課程修了．同年 KDDI 株式会社（当時 KDD）入社．現在,（株）KDDI 研究所 Web データコンピューティンググループ研究主査．XML，SVG，ウェブマッピング，ITS 等の研究開発に従事．

西山　　智（正会員）

1961 年生．1984 年東京大学工学部電気工学科卒業．同年国際電信電話（現 KDDI）（株）入社．1991 年米国テキサス大学オースチン校計算機科学科修士課程修了．現在,（株）KDDI 研究所 Web データコンピューティング G グループリーダ．この間，データベース，ネットワーク管理，ITS，エージェント通信，ユビキタス通信システムの研究に従事．2005 年度本学会山下記念賞受賞．電子情報通信学会会員．