社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－124（5）
2005－ GN － 57（5）
2005／9／21

XML 文書符号化方式"XEUS"の高圧縮化に関する一考察

小林 亜令　　村松 茂樹　　太田 慎司　　西山 智
（株）KDDI 研究所

XML 文書は拡張性・可読性の長所を持つ反面，テキスト形式のため冗長性が高い．このため携帯電話網での通信時にはデータ圧縮することが有利であると考えられる．筆者らはこれまでに携帯電話のための XML 文書符号化方式"XEUS"を提案し，データ圧縮率および符号化/復号処理負荷の面でその有効性を示した。しかし，XEUS は符号化対象文書内のノード類似性や要素値/属性値の特性を有効利用しておらず，圧縮率を改善する余地がある．そこで本稿では，XEUS の圧縮率を向上するための改良方式を提案する．性能評価実験の結果，データ圧縮率，符号化/復号処理時間，デコーダサイズ，システム全体処理時間の全てにおいて，提案手法が従来法に比べ，概ね優れた性能を持つことが分かった．

An improved method of XML document encoding with uniformed sheet “XEUS”

AREI KOBAYASHI , SHIGEKI MURAMATSU, SHINJI OTA, and SATOSHI NISHIYAMA
KDDI R&D Laboratories Inc.

XML data is extensible and easy to read from its nature, but it also has disadvantage in terms of transmission code length. We have already proposed a XML document encoding method, named “XEUS”. This method is effective, especially for transmission via cellular network where the bandwidth is limited. However it still has room for generating more compact encoded data, as it doesn't use the node similarity and the statistical character of element/attribute value. In this paper, we propose an improved encoding method using above character and we show the effectiveness of this method through performance evaluation.

1. はじめに

近年, PC を対象とした WWW ベースの情報システム構築の際には，開発コストの低減，汎用性の確保，拡張性の確保の観点から XML 形式のデータを採用するケースが一般的になってきている．XML は，拡張性・可読性の長所を持つ反面，テキストデータのためデータ格納効率が悪く, 伝送速度が低い携帯電話網ではデータ圧縮を行うことが望ましい．

従来の XML 文書を圧縮する方法として, http1.1 のようにプロトコルスタックで gzip を用いる手法に加えて，種々の XML 文書圧縮手法が提案されている（XMill[1]，xmlppm[2]，WBXML[3]）．しかしこれらの従来法では, 符号化により伝送符号量を低減できても, データ受信側でパース処理に加えて復号処理も必要とするため処理負荷が増大したり，任意の XML データ（スキーマ）を対象としていなかったりといった課題が存在する．

そこで筆者らは，この課題を解消するために，「XEUS(XML document Encoding with Uniformed Sheet, ゼウス)[4][5]」と呼ぶ符号化方式を提案した．XEUS は，「XEUS シート」と呼ばれる XML 形式の符号化テーブルを定義して符号化を行うことを特徴としており，任意の XML 文書を対象とし，そのスキーマ情報を利用した圧縮を行うだけでなく，符号化時にパース処理も行い, 受信側の処理負荷を低減できる．しかし，符号化対象文書内ノードや要素値/属性値の特性を有効利用していないため圧縮率の観点から性能改善の余地がある．

そこで本稿では，高圧縮化を目的とした XEUS の改良方式を提案する．また，次章で述べる携帯電話のための XML 文書符号化方式に対する４つの要求条件の観点から性能評価実験を行い，提案手法の有効性を示す．

2. 携帯電話のための XML 文書符号化方式に対する要求条件

まず携帯電話のための XML 文書符号化方式に対する要求条件を以下に示す．XEUS は，以下の課題を解決し，システム全体のパフォーマンスを向上させることを目的としている．

●データ圧縮

前節で述べたとおり，XML 文書はテキスト形式のためデータ格納効率が悪く，携帯電話網ではデータ圧縮を行い伝送符号量を低減する必要がある．

●符号化処理時間

符号化を導入することにより伝送符号量が低減されるが，XML 文書では必要なかった符号化処理が必要になる．この処理によってシステム全体の処理時間が増大してはならない．

●復号処理時間

前節で述べたとおり，サーバ側でデータ圧縮することにより，携帯電話上の復号処理時間がパース時間より大きくなってはならない．

●デコーダサイズ

携帯電話上のアプリケーション実行用メモリ容量（ヒープサイズ）は限定されているため，データ圧縮を導入することにより，従来の XML パーサに比べてデコーダのサイズが増大して，ヒープサイズを圧迫してはいけない．

3. 従来の XEUS

3.1 符号化規則

XEUS は，3.2 節で述べる XEUS シートと呼ぶ XML 文書のノードのツリー構造と符号化テーブルをエンコーダとデコーダであらかじめ共有することを前提とし，エンコーダは符号化対象 XML 文書をパースして，ノードのツリー構造を符号化する．また要素名，属性名，要素値，属性値を XEUS シートに定義された符号化テーブルに則って符号化し，デコーダに送ることを基本とする．

送られるバイナリデータは，図 1 に示したとおりヘッダ部とボディ部に分けられ，ヘッダ部に符号化処理の際に参照した XEUS シート情報等を格納する．ボディ部には，XML 文書データの先頭（ルート要素）から要素の出現順に，各要素の符号化結果（要素名，属性名，属性値，要素値）を可変長で格納する．また各要素符号は図 2 の通り，要素名部，属性値部，要素値部の 3 つに分けられ，要素名部には，要素開始符号（各要素の符号開始識別符号），要素占有符号長符号（各要素符号が占める符号長），要素名符号（XEUS シートに定義された要素名符号），属性存在有無フラグ符号（XEUS シートに定義された各属性の存在有無識別フラグ）が格納される．属性値部/要素値部には，属性値/要素値符号長（各属性値/要素値符号が占める符号長），属性値/要素値符号（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属性値/要素値）が格納される．

3.2 XEUS シート

XEUS シートは XML 形式で記述され，符号化対象文書のノードのツリー構造と符号化テーブルを定義する．すなわち，各要素について，要素名，属性名，属性値，要素値，子要素名のリスト，親子関係と符号化テーブル，また属性値，要素値について，データ型（数値型，文字列型，選択型），符号長，個数を定義する．

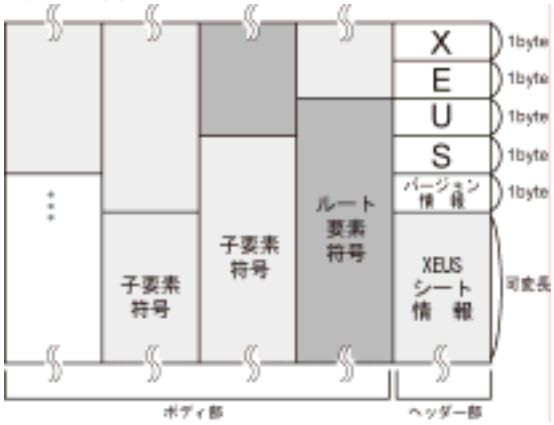

図 1：符号化後のバイナリデータ構造

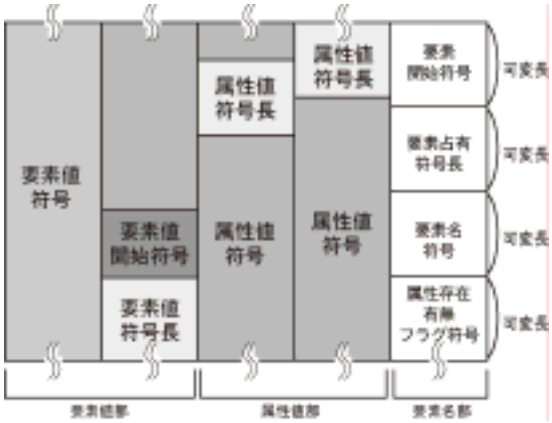

図 2：要素符号

4. 提案手法

4.1 従来の XEUS の課題

従来の XEUS は，符号化対象文書内ノードや要素値/属性値の特性を有効利用できていないという課題が残されている．

(1) 符号化対象文書内に出現するノード類似性

符号化対象 XML 文書内の各要素に付随する属性や子要素の出現パターンが類似する場合があり得る. 例えば下例では２つの polyline 要素について属性の各値は異なるが, 属性名の出現パターン，及び空要素であることは一致している．このようなケースにおいて従来の XEUS では，図 2 の要素開始符号，要素名符号，属性存在有無フラグ符号の３つの部分は全て同じ符号化結果になるにも関わらず, その都度同じ符号列を格納している（出現ノードの類似性を利用した符号化方式ではない）.

例：

```
<svg>
  <polyline points=”0,0 10,10”
            stroke=”red” fill=”none”/>
  <polyline points=”30,10 10,20”
            stroke=”blue” fill=”red”/>
</svg>
```

(2) 要素値/属性値の出現頻度特性

符号化対象 XML 文書内に存在する数値，列挙型をデータ型とする要素値/属性値の出現頻度が偏る場合があり得る(2-a). さらに要素や属性のセマンティクスを用いることで, より高圧縮を行うことが可能である. 例えば上記例のように座標値が値として記述される場合には, 値を(x,y)の 2 値を組として扱い, 前座標値との差分値を符号化した方が, 符号圧縮率向上が期待できる場合がある. しかし XEUS ではこれらのセマンティクスを利用した圧縮は行えていない(2-b). 文字列においては，gzip による符号圧縮を行うが，符号化対象文字列値を連結し一箇所にまとめることにより gzip による符号圧縮効率の向上が期待できる(2-c). しかし従来の XEUS 符号化法では，これらのセマンティクスを考慮していない.

そこで上記の課題を解決し，符号圧縮率を向上させることを目的とした提案方式を次節に示す.

4.2 改良型符号化モデル (提案方式)

本節では，前述した課題を解決するために改良した符号化規則を提案する．図 3 に示すとおり，提案方式は，ボディ部のバイナリデータ構造が従来法と異なる．以下に特徴を示す.

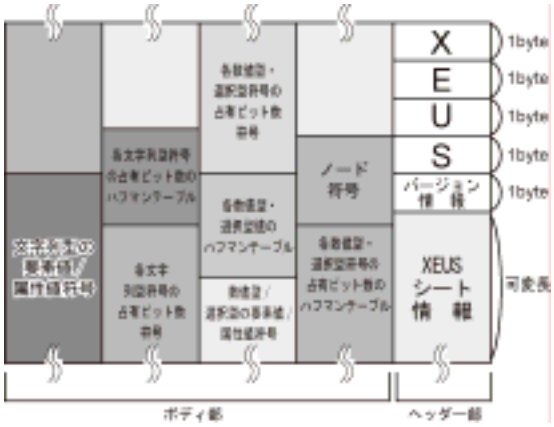

図 3：符号化規則

(1) 各ノードの属性名群と子ノード有無情報を組にして符号化

従来のボディ部には, XML 文書データの先頭（ルート要素）から要素の出現順に，各要素の符号化結果を格納していたが，提案方式では 4.1 節の課題(1)を解決するため，図 3 に示す通り，ノード符号と要素値/属性値符号を分離して符号化する．さらに各ノード符号は，各要素に対する属性名群，子ノード有無の情報を組にして符号化し，結果を格納する．（例：points,stroke,fill 属性が存在し，子ノードの無い polyline 要素に 0x00 の符号を割り当てる.）

(2) 要素値/属性値のもつデータ型毎に分離してハフマン符号化

4.1 節の課題(2-a)を解決するため，図 3 に示す通り，ノード符号や数値/列挙型のデータ型を持つ要素値/属性値に対してハフマン符号化を行う．ハフマンテーブルは XEUS シートに定義されたデータ型種別毎に生成する．本方式により，要素値/属性値の出現頻度特性を利用した符号化法となり，符号圧縮率向上が期待できる.

(3) データ型（座標値型，ユーザ定義型）の新設

提案方式では，課題(2-b)を解決するため，要素値/属性値のセマンティクスを用いた符号化規則を XEUS に組み込めるようにした．特に利用頻度が高いと考えられるセマンティクスとして，以下の 2 つ

については予め組み込んだ.

・座標値型

これは数値型（integer,unsigned integer,double,fixedpoint）の拡張（例：unsigned integer の座標値型）であり，２つの数値を組として，前組からの差分値を算出して符号化する．SVG 文書のように要素値/属性値に座標値(x,y)が記述される場合に有効である．

・ハフマンテーブル ID 付データ型

前述の通り，要素値/属性値のハフマンテーブルは，データ型毎に生成されるが，ノード種別（要素名/属性名）に依存した値の出現特性がある場合には, 同じデータ型であってもハフマンテーブルを分けた方が符号圧縮効率が向上する．そこで提案手法では, 同じデータ型であってもハフマンテーブルを複数用意し，その識別用 ID を付加したデータ型を定義することができる．

（例：points という ID のハフマンテーブルを用いて符号化する integer 型）

(4) 文字列型のデータ型を持つ要素値/属性値の連結処理

課題(2-c)を解決するため，従来法と異なり，文字列値のデータ型を持つ要素値/属性値を連結し一箇所にまとめることにより gzip による符号圧縮効率の向上を図る．

これらの手法により, 符号化対象文書内に出現するノードの類似性や要素値/属性値の出現頻度特性を考慮した符号化方式が可能となり, 圧縮率の向上が見込める．

4.3 エンコーダとデコーダの処理フロー

4.2 節の符号化規則に従ったバイナリデータを生成するための符号化処理フロー及びバイナリデータの復号処理フローを図 4 と図 5 に示す．エンコーダでは，まず初期化処理を行い(1)，XEUS シートの読み込み，解析を行う(2)．その後，符号化対象文書のパース処理，符号化処理を行い，ボディ部の生成を行う(3-4)．その後，要素値/属性値のハフマン符号化を行う(5)．最後に文字列の gzip 処理を行い，出力する(6).

デコーダは，まず初期化処理を行い(1)，XEUS シートの解析を行う(2)．次に XEUS バイナリデータを読み込み gunzip 処理を行う(3)．次にボディ部の復号処理を行い，SAX イベントを発行する(4-6)．最後に endDocument イベントを発行した時点で復号処理終了となる(7).

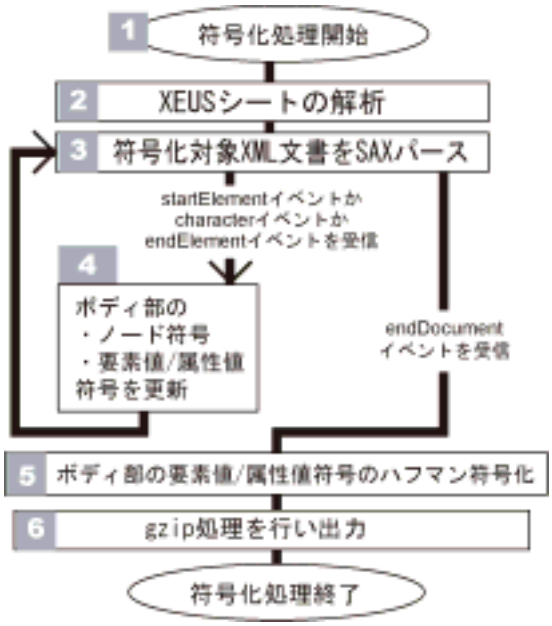

図 4：符号化処理フロー

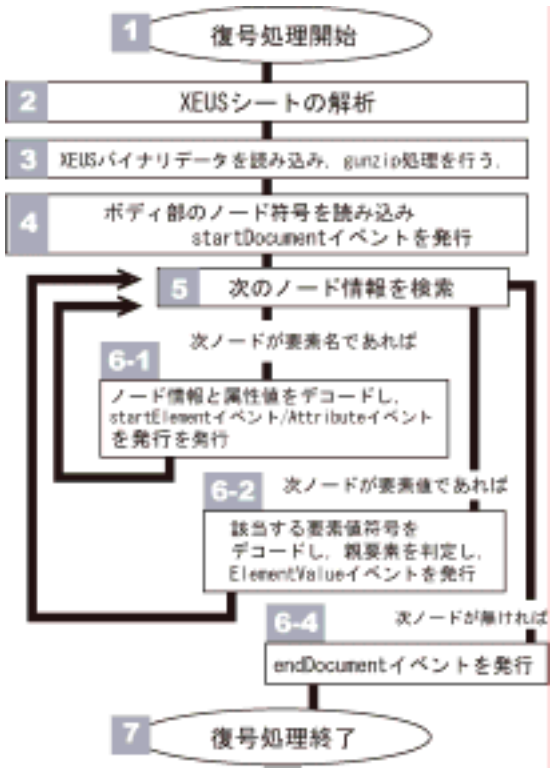

図 5：復号処理フロー

5. 提案手法の性能評価

4 章で述べた提案方式についてエンコーダ，デコーダの実装を行い，2 章で述べた要求条件に基づいて性能評価実験を行った．

5.1 評価コンテンツ

評価コンテンツには，XMark[6]，SVG[7]，RSS[8]の 2 種類（10KB と 100KB）を用いた．表 1 に示した各々の特性から XMark は文字列，SVG は数値，RSS はマークアップ部分（要素名，属性名）の比重が高いことが分かる．5 章で行う全ての実験は本コンテンツ群を使用している．

表 1：評価コンテンツの特性

	マークアップ[%]	数値[%]	文字列[%]
XMark	17.4	1.8	**80.8**
SVG	26.9	**65.2**	7.9
RSS	**75.2**	5.3	19.5

5.2 データ圧縮率

表 2 に提案方式, 従来の XEUS 及び代表的な圧縮方式である gzip のデータ圧縮率を示す．圧縮率は以下の式にて算出した．

$$\text{圧縮率}[\%] = \frac{\text{符号化後(バイナリ)のデータサイズ}}{\text{符号化前}(XML)\text{のデータサイズ}} \times 100$$

文字列の多い XMark と数値の多い SVG については，提案方式は他方式に比べ 5.0%～8.5%優れている．これは提案方式により XML 文書内に出現するノードの類似性や要素値/属性値の出現頻度特性を考慮したことが要因と考えられる．ただし RSS に対しては，従来の XEUS とほぼ同等の性能である．これは，RSS に数値型のデータ型を持つ要素値/属性値が少ないことと，RSS は RDF 準拠のため属性が存在せず, 提案方式の効果が現れなかったことが原因と考えられる．

表 2：圧縮率の比較結果

実験コンテンツ		gzip	従来の XEUS	**提案方式**
XMark	10KB	43.9%	45.1%	**38.5%**
	100KB	37.7%	38.3%	**33.3%**
SVG	10KB	37.3%	28.5%	**21.2%**
	100KB	32.0%	26.3%	**17.8%**
RSS	10KB	16.4%	12.7%	**13.3%**
	100KB	11.3%	9.3%	**9.4%**

5.3 符号化処理時間

表 3 に, 提案方式と従来の XEUS のエンコーダの符号化処理時間を示す．エンコーダの動作環境は以下の通りである．

○OS ：RedHat Enterprise Linux ES ver.3
○CPU：Pentium4 2.8GHz
○Memory：2GB
○実行環境：Apache2.0 Module

初期化処理後, 符号化対象文書の読み込みを開始した時点から XEUS バイナリデータ出力を完了した時点までを符号化処理時間として計測した．表 3 の通り，従来の XEUS 方式と比較すると，XMark,SVG では，3%～57%優れているが，RSS についてはデータサイズが大きくなると若干劣ってしまう．これは提案方式のノード情報符号化規則が複雑化した結果，ノード情報の多い RSS ではノード符号化時間が大きくなったためと考えられる．

表 3：エンコード処理時間

実験コンテンツ		従来の XEUS[msec]	**提案方式 [msec]**
XMark	10KB	14.0	**13.5**
	100KB	101.1	**64.6**
SVG	10KB	24.8	**21.0**
	100KB	202.5	**139.2**
RSS	10KB	22.8	**18.1**
	100KB	187.1	**209.4**

5.4 復号処理時間

表 4 に，提案方式，従来の XEUS の復号処理時間を示す．表 4 では，XEUS がない場合本来必要となる XML 文書のパース時間も比較のため示した．XEUS デコーダ及び XML パーサ（Reaxion 製 XML Parser[9]）の動作環境は以下の通りである．

○仕様端末：携帯電話(au W21S)
○デコーダ動作環境：BREW 2.1 Ja

計測方法は，要素名数，属性名数，要素値数，属性値数をカウントするようなアプリケーションを実装し，XEUS バイナリデータの読み込み開始時点からカウント完了時点までを復号処理時間として計測した．表 4 から分かるとおり，提案方式の復号処理時間は, 従来の XEUS と比較すると, XMark, RSS では 15%～40%の性能向上が見られるが, SVG では 9%～10%の性能劣化が見られる．これは SVG には

数値が多く，ハフマン符号化による圧縮率向上を実現できている分，復号処理負荷が大きくなることが要因と考えられる．

表 4：復号（パース）処理時間の比較

実験コンテンツ		XML パーサ [msec]	従来の XEUS [msec]	**提案方式 [msec]**
XMark	10KB	316	20	**12**
	100KB	2573	142	**92**
SVG	10KB	382	30	**33**
	100KB	3796	180	**197**
RSS	10KB	873	19	**16**
	100KB	7922	130	**100**

5.5 デコーダサイズ

表 5 に，デコーダのサイズを示す．表 5 から分かるとおり，提案方式のデコーダのサイズは，他の XML パーサや従来の XEUS のサイズとほぼ等しい．

表 5：デコーダ（パーサ）のサイズ

	XML パーサ	従来の XEUS	**提案方式**
デコーダ（パーサ）のサイズ[byte]	22,012	20,224	**21,528**

注）BREW アプリケーション実行形式(mod)で比較

5.6 システム全体性能評価

表 6 に，提案方式，従来の XEUS 及びそれを用いない場合（XML パーサと記述）のシステム全体性能を示す．本実験では，提案方式と XML パーサ，従来の XEUS について，http リクエストを発行してから復号（パース）処理が完了するまでの時間を計測した．ネットワーク環境は，cdma2000 1x EV-DO 網を用いている．XEUS サーバとコンテンツサーバは同一サーバ内に存在し，XML パーサの評価実験時には XEUS サーバ（中継サーバ）を介さず，直接コンテンツサーバにアクセスし，gzip 処理された XML ファイルをダウンロードする．表 6 の結果から分かるとおり，商用通信網を利用した実験であるためデータ取得時間にばらつきがあるものの，伝送符号量，符号化処理時間，復号時間に差がある分，システム全体性能は提案方式が優れている傾向がある．

表 6：システム全体評価 [msec]

		XML パーサ	従来の XEUS	**提案方式**
XMark	10KB	1083 (774/309)	857 (837/20)	**735 (723/12)**
	100 KB	4092 (1599/2493)	1737 (1595/142)	**2289 (2197/92)**
SVG	10KB	913 (540/373)	694 (664/30)	**520 (487/33)**
	100 KB	6042 (2323/3719)	1729 (1549/180)	**1560 (1363/197)**
RSS	10KB	1374 (507/867)	775 (756/19)	**552 (536/16)**
	100 KB	8899 (1038/7861)	1331 (1201/130)	**1246 (1146/100)**

注）括弧内はデータ取得時間/デコード(パース)時間

6. まとめ

本稿では，XML 文書符号化方式”XEUS”の高圧縮化を目的とした符号化規則を提案し，2 章で述べた要求条件に基づいて行った性能評価実験結果を示した．その結果，データ圧縮率については，SVG と XMark で 5.0%-8.5%，符号化処理時間については，XMark,SVG で 3%～57%，復号処理時間については，XMark,RSS で 15%-40%の性能向上を確認した．またデコーダサイズについても大きな増大は見られなかった．各実験において性能が下回ったケースも見受けられたが，僅かな差であった．その結果，システム全体性能について比較した結果，提案方式の性能が他方式よりも優れていることが分かり，有効性が示された．

参考文献

[1] H.Liefke and D.Suciu. Xmill:an efficient compressor for XML data.In Proceedings of the 2000 ACM SIGMOD,pp.153-164,2000

[2] James Cheney, Compressing XML with Multiplexed Hierarchical Models, inProc. of the 2001 IEEE Data Compression Conference, pp. 163-172

[3] Bruce Martin and Bashar Jano:http://www.w3.org/TR/wbxml

[4] 小林,松本,井ノ上 ” XML 文書汎用符号化方式「XEUS」”, 信学技報 DE2001-9

[5] 小林他, ” 汎用 XML 文書符号化方式「XEUS」の性能評価 ” ,FIT2003.

[6] http://monetdb.cwi.nl/xml/index.html.

[7] http://www.w3.org/TR/SVG11/

[8] http://web.resource.org/rss/1.0/spec

[9] http://www.reaxion.com/