Vol.2013-CSEC-61 No.22
Vol.2013-IOT-21 No.22
2013/5/10

XMPPの効率的なEXI符号化を実現する動的文法合成方式

土井 裕介　佐藤 弓子

概要：マルチベンダな機器情報を管理する際に，名前空間の仕様を活用できる XML が効果的な場合が多い．特に，リアルタイム性と多様性がある M2M 通信を実現するためには，効率的にバイナリ化された EXI と XMPP の組み合わせによる効果は高いことが予想されている．一方，EXI と XMPP の組み合わせには，いくつかの問題が存在することが判明している．本発表では，EXI と XMPP の問題のうち，XMPP での通信中に動的に導入される XML スキーマ型について，文法の動的合成により効率的かつ妥当性が保証された符号化・復号を実現する方法について述べ，また必要となる実装量について，静的なスキーマ合成と比較評価する．

Dynamic Grammar Composition for Efficient XMPP Encoding in EXI

Abstract: XML can be effectively used for multivendor device information management. Especially, high efficiency of the combination of EXI and XMPP is expected in real-time heterogeneous M2M communication. However, there are several issues known to the combination of EXI and XMPP. This research describes how to manage dynamic schema-informed grammar synthesis that may happen in XMPP communication. In addition, a comparison with predefined combination schema in terms of the amount of implementation is also given.

1. 背景

エネルギーマネジメント技術・スマートグリッド技術のように，マルチベンダ・マルチシステムでのメッセージ交換を必要とするシステムにおいて，XML および XML Schema の持つメッセージ仕様記述性とその拡張性は有用である．また，XML を利用したリアルタイムなイベント通知には，XMPP[1] (Extensible Messaging and Presence Protocol) が有用であり，例えば OpenADR2.0[2] などで採用が検討されている．一方で，帯域や計算機資源の有効利用には XML のバイナリ版である EXI[3] が効果的であることがわかっており [4]，例えば SEP(Smart Energy Profile)2.0[5] などで採用されている．[6] で示されるように，XMPP における XML 利用に EXI を適応するには，(1) bit 単位の符号長を効果的に使うための NOP の創設 (2) 文法の動的な切り替えの 2 つが必要である．

本研究では，[6] における文法の切り替え方式の一方式を提案する．

1.1 XML Schema におけるワイルドカードによる文書型拡張

W3C XML Schema におけるワイルドカード要素の定義は以下のようになる．

`<xs:any processContents=(処理方法) namespace=(名前空間)/>`

`xs:any` で指定される箇所にあてはまる要素は明示的には定められないことになる．これにより XML 文書のスキーマ定義を変更することなく自由な拡張が可能となる．

名前空間の指示には，URI による明示的な名前空間の指定の他，現在定義している名前空間ではないもの (`##other`) という指示方法が存在する．この指示を行うことにより，将来どのような拡張であっても，他の独立した名前空間に定義されている限りにおいては受け付け，かつ，プロトコルの本体 (例えば XMPP の場合は `http://etherx.jabber.org/streams` など) は混乱させない，といった指定がある程度可能になる．

また，処理方法の指示においては，XML を解釈する際にこの要素をどのように検証するかの指示が可能であり，拡張要素として利用する場合はしばしば `lax` という指示子が用いられる．これは，その対象となる名前空間に対応す

[1] 株式会社東芝 研究開発センター

る定義を XML パーサが知っていた場合は検証を行い，そうでない場合は検証を行わず読み飛ばす，という意味に相当する．

1.2 XMPP の特徴

本節では，説明のため，XMPP の通信について簡単に説明する．XMPP は，ノード間で 1 つの TCP セッションを確立し，その上でメッセージをやりとりすることで通信を行う．例えば，あるクライアントが発行するメッセージの終点は，そのクライアントが接続しているサーバ，そのサーバに接続している他のクライアント，他のサーバ，他のサーバに接続しているクライアントの可能性があり，サーバはこれらを適切にルーティングする責任を持つ．

XMPP のコネクションは，1 つの大きな XML 文書となっており，その中に `iq`，`presence`，`message` などのタグが含まれる．プログラムはこのメッセージを受信したらイベントドリブンに処理を行う．例えば `presence` タグが閉じたらプレゼンス情報を取り扱うハンドラが呼ばれ，`message` タグが閉じたらメッセージ情報を取り扱うハンドラが呼ばれる，等が一般的な実装方法である．

また，XMPP においては，XML スキーマと名前空間により標準的なメッセージだけでなく拡張メッセージも厳密に定義されている．[*1] これは，XML Schema によるエレメント定義のうち，ワイルドカード (`xsd:any`) によるものであり，特に，`processContents="lax"`という指定が行われている部分については，対象となる要素の定義を解釈側が知らなければ，そのままスキーマのない XML として処理して (あるいは無視して) 良いという定義によるものである．

XMPP は以上のような XML の特徴を利用することにより，特定のノードのみが解釈可能なメッセージを破綻なくやりとりすることが可能となっている．

1.3 EXI の特徴

EXI は，エンコーダとデコーダで同期した文法 (ステートマシン) を利用し，文法構造をコンパクトに符号化し，内包されるデータについては，型情報が利用できる際は効率的なバイナリ表現により符号化する仕様である．文法の生成には，XML スキーマを用いる方法と用いない方法がある．XML スキーマを用いる方法 (Schema-Informed Grammar，以後 SI 文法と呼ぶ) では，スキーマに由来する構造情報を文法が内包するため，デコードと妥当性検証を同時に行える．一方，スキーマを利用しない場合 (Built-in Grammar) は，XML に記述可能な表現であればどのような情報でも記述できるが，効率性においてやや劣り，スキーマに対する妥当性検証は別途行う必要がある．

[*1] `http://xmpp.org/resources/schemas/`

また，EXI にはさまざまなオプションが存在する．ここでは，本研究に関係するオプションとして符号長に関するオプション (bit-pack, byte-align) および自己完結要素 (self-contained) に関するオプションについて説明する．

符号長について，8bit 単位での符号長にするオプション (byte-align) と，1bit 単位でコンパクト化するオプション (bit-pack) が存在する．当然のことながら，8bit 単位での符号長としたほうが EXI 単体での効率は劣る．なお，既存の圧縮アルゴリズム (DEFLATE) との組み合わせによる圧縮が可能な場合は，8bit 単位での符号化のほうがより高い圧縮率となる場合が多い．

また，要素の透過なコピーを可能とするための，selfContained オプションが存在する．selfContained オプションは，特定の要素を selfContained，つまり独立した要素として記述することを可能にする．selfContained 要素以下はコピー可能になるように設計されている．これは，通常の EXI においては繰り返し登場する文字列について二度目以降は参照とすることにより符号の効率化を図っているのに対し，selfContained 要素に関しては再度文字列をストリームに含める必要があることを意味する．副次的効果として，bit-pack オプションを使った場合であっても，selfContained 要素は必ずオクテット境界から開始し，オクテット境界で終了するように padding が付加される．

SI 文法による bit-pack オプションを用い，かつ DEFLATE 等の圧縮アルゴリズムを利用しない EXI は，組込機器が多い環境に適している [4]．DEFLATE アルゴリズム等を利用しないことから，省メモリにデコード処理が可能である．一方，符号は十分にコンパクトであり，またデコード処理も静的なステートマシンにより可能となるため，動的メモリ確保を排除でき，より堅牢なデコーダとなる．

2. Schema-Informed EXI の動的文法合成

本節では，Schema-Informed EXI の動的文法合成手法について述べる．

2.1 問題の定義

XMPP の符号化において，あるスキーマ S_1 によって定義される SI 文法 G_1 を利用して符号化を開始した EXI 文書の途中で，S_1 には含まれない名前空間において定義されるスキーマ S_2 が動的に導入される．ここで，(1)S_2 が導入された後の文脈において適切にスキーマによる符号化が可能であること，(2)S_2 の導入がスキーマから文法を再導出する，あるいは，再導出された文法全体を保持する必要があるような高コストなものでないこと．この 2 つの条件を充足する SI 文法 G_x を構築する手法を定義する．

G_x として，スキーマ S_2 にもとづく文法 G_2 をそのまま利用することはできない．これは，S_2 にもとづく要素が導入された文脈において，S_1 は以前存在し，S_2 にもとづく

要素から S_1 の定義を参照することが可能であることによる．従って，G_x は何らかの手法で合成される必要がある．この合成手続き全体を，本稿では $G_x = G_1 \ll G_2$ と定義する．なお，本稿では「$\ll$」を「G_1 のコンテキスト上に G_2 を合成する」，という意味で定義し，慣用的に利用される $\ll$ 演算子とは関係がない．

2.2 SI 文法の構成要素

本節では，文法を合成するにあたり，文法の構成要素を明確にする．EXI における SI 文法のうち，静的な要素は，以下の 3 要素から成ることが [3] からわかる．

- D :「ドキュメントグラマー」 [3] section 8.5.1 に定義される．ある XML 文書のルート要素を規定する文法であり，ある SI 文法に必ず一つ含まれる．実体は一組の生成規則．なお，ここでは簡単のため，section 8.5.2 に定義される，ドキュメントフラグメントグラマーはドキュメントグラマーと同質のものであることから，同一として扱う．
- T :「エレメントタイプグラマー」 [3] section 8.5.4 に定義される，ある文書に含まれる要素とその型を定義する文法であり，スキーマによって定義された型に対応して一組の生成規則が定義される．スキーマでは複数の型を定義できることから，ある SI 文法に含まれるエレメントタイプグラマーは，個々の型名でラベル付けされた生成規則の集合である．
- S :「ストリングテーブル初期化ベクトル」 [3] appendix D に定義される，ストリングテーブルの初期化に使われる要素である．仕様上は文法に含まれないが，スキーマに由来する要素であるとして，本研究では文法の一部として取り扱う．

以上から，$G_x = G_1 \ll G_2 = (D_x, T_x, S_x)$ であるとし，以下に，$D_x T_x, S_x$ を定義する．

2.3 ドキュメントグラマーの合成

ドキュメントグラマーは，XML 文書のルート要素に相当する生成規則である．W3C XML Schema においては明示的に定義されている要素は全てルート要素になりうるので，単純にルート要素を列挙する生成規則となる．

文法 G_1 と G_2 があったとき，それぞれのドキュメントグラマー D_1 と D_2 の合成は，列挙された要素を単純に結合することにより実施できる．このとき，EXI 仕様 [3] が指示するような形で各要素を再度整列してもよいが，任意の文法合成に対してできるだけ RAM を利用せずに合成演算を完了するためには，単純に D_1 の後に D_2 を (あるいは D_2 の後に D_1 を) 結合するのがよい．

2.4 エレメントタイプグラマーの合成

エレメントタイプグラマーは，個々の文法要素として参照される型定義であり，一つの型が一組の生成文法 (あるいは状態機械) に対応する．エレメントタイプグラマーの生成文法はそれぞれ QName[*2]でラベル付けされた辞書型に等しい．従って，これらのラベルをもとに辞書を統合することにより T_x を生成できる．

2.5 ストリングテーブル初期化ベクトルの合成

ストリングテーブル初期化ベクトルは，EXI の通信で利用する文字列リテラルを効率的に表現するために，スキーマなどに含まれるリテラルや XML 処理において特別な意味を持つ URI などを列挙した表の集合である．URI パーティションと呼ばれる，URI の一覧を含めた一つの表と，各々の URI に対し，利用されるローカル名の集合を定義した表 (ローカルネームパーティション) の集合からなる．ここで，URI パーティションを U，個々の URI を $u \in U$ としたとき，ローカルネームパーティション L の特定の URI に対応する表を $L[u]$ とする．このとき，$S_x = S_1 \ll S_2 = (U_x, L_x)$ は，以下のように定義する．

$$U_x = U_1 \oplus U_2 \tag{1}$$

$$L_x = [L_1[u] \oplus L_2[u] | u \in U] \tag{2}$$

なお，ここでは，$p \oplus q$ は，要素の重複がない順序集合 p および q があったとき，各々の順序を維持したまま，重複を排除して結合を行う．例えば，$(1,2,3) \oplus (6,4,2) = (1,2,3,6,4)$ となる．また，$L[u]$ が EXI 文法上存在しない場合は空集合であるとする．

重複を排除してストリングテーブル初期化ベクトルを合成し，これをストリングテーブルの初期状態として利用することにより，合成状態の文法における URI ならびに QName の，文字列によらず識別子による指定が可能になる．

3. 評価

文法結合方式の評価を行うため，我々が持つ EXI 実装 (EIGEN[4]) のうち EXI 文法生成を行うプログラムを利用して文法サイズおよび差分の評価を行った．この文法生成プログラムは，XML スキーマを入力とし EXI 文法に対応する文法構造を生成する．内部的にエンコーダおよびデコーダ生成に利用しているが，文法構造をそのままテキストに出力する機能があるため，これを利用した．

3.1 評価の条件

評価のために，簡単なスキーマ群を作成した．まず，拡張の基本になるスキーマ (base) を図 1 に示す．ある氏名を記述するための XML スキーマであり，`<first/><family/>` 以下，本スキーマが対象としている名前空間以外であれば，

[*2] XML で利用する，一意性のある識別子であり，URI とローカル名の組み合わせからなる．

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://example.org/base"
 targetNamespace="http://example.org/base"
 elementFormDefault="qualified">
 <xs:element name="person" type="root" />
 <xs:complexType name="root">
  <xs:sequence>
   <xs:element name="first" type="xs:string" />
   <xs:element name="family" type="xs:string" />
   <xs:any processContents="lax" namespace="##other"
    minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

図 1 拡張の基礎となるスキーマ: base.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://example.org/ext"
 targetNamespace="http://example.org/ext"
 elementFormDefault="qualified">
 <xs:element name="prefix" type="xs:string" />
 <xs:element name="address" type="AddressType" />
 <xs:complexType name="AddressType">
  <xs:sequence>
   <xs:element name="street" type="xs:string" />
   <xs:element name="city" type="xs:string" />
   <xs:element name="zip" type="xs:string" />
   <xs:element name="country" type="xs:string" />
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

図 2 拡張スキーマの例: ext.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://example.org/combo"
 targetNamespace="http://example.org/combo"
 elementFormDefault="qualified">
 <xs:import namespace="http://example.org/base"
  schemaLocation="base.xsd" />
 <xs:import namespace="http://example.org/ext"
   schemaLocation="ext.xsd" />
</xs:schema>
```

図 3 静的合成スキーマの例: combo.xsd

<xs:any/>要素により任意の拡張を許す構成になっている．

これに対し，拡張要素は例えば図 2 のように定義する．拡張要素には 2 つの定義がされており，簡単な定義として prefix(Dr. 等)，複雑な要素として address(住所) の定義を許している．

本評価では，複数のスキーマから動的に合成した文法を生成する (postcombo) が，比較のため，静的に合成したスキーマ combo を図 3 のように定義する．

以上のスキーマに対し，提案方式が優れた拡張性，特に拡張時に文法構造の変化を最小限におさえる仕組みを備えることを示す．

3.2 文法量の評価

我々の実装では，EXI の文法のうち，ドキュメントグラマーとエレメントタイプグラマーは同一のテーブルの参照により実現されている．テーブルは，一行が一つの型に対応する型テーブル・一行が一つの状態に対応する状態テーブル・一行が一つの遷移に対応する遷移テーブルの 3 段階から成り，本質的な文法サイズはほぼ遷移テーブルの要素数で決定される．これは，型テーブル・状態テーブルはインデックスを 1 つ格納するだけの構造であるのに対し，遷移テーブルは文法の情報を全て格納する必要があるためもっともデータ量が多いことに加え，一般的に遷移数はもっとも多くなることによる．従って，以下簡単のため文法量は遷移数で比較することとする．

節 3.1 で導入した各文法の遷移数を表 1 に示す．一見して明らかなように，この例では静的合成文法 (combo) と動的合成文法 (postcombo) は同一の複雑さで実現できている．

表 1 各文法の遷移数

文法	遷移数
base	9
ext	10
combo	14
postcombo	14

3.3 差分量の評価

前節で，動的合成による文法量が静的な合成と差がないことを遷移数により確認した．次に，静的文法と動的文法の差分を比較する．ここで，差分がより大きなブロックに固まり，順序変化が少なければ少ないほど，合成文法の記述が簡単である，という仮定を置く．

文法 base と combo, postcombo それぞれの文法のテキスト形式出力の間を，diff コマンドにより差分を取った内容を表 2 に示す．なお，本実装におけるテキスト形式出力は，一つひとつの遷移に対応する行を含む文法解釈に必要な各テーブル (ストリングテーブルを含む) を 1 行 1 行出力するものである．従って，文法間の差分の大きさは単純に diff コマンドにより計測可能である．表 2 では，差分行数 および うちマイナス (diff -u の出力のうち行頭が-ではじまるもの，削除を意味する) の数を示す．削除を別途カウントしたのは，文法合成においてはスキーマの追加のみを定義しているにも関わらず削除が必要になるというのは，要素の順序替えを意味しており，これは実装の複雑さ[*3]に直結するからである．

表 2 からわかるように，postcombo(提案方式) は，静的合成方式 (combo) に比べて文法上の差分が少なく，かつ全

[*3] 単純な追加であればリスト構造で良い

表 2 文法合成方法による差分数の比較

比較	差分行数	マイナス数
combo	66	8
postcombo	21	0

てが追加であることから，実装がよりシンプルになることが期待できる．

4. 結論と今後の課題

本研究では，厳密なスキーマ定義が求められがちな業務系/監視系センサネットワークやスマートグリッド向けプロトコルとして適している XMPP について，より効率的なメッセージングを実現するための文法合成手法について提案し，XML スキーマの段階で合成する方式に比べて提案方式による EXI 文法合成のほうが文法差分量が少なく，評価に用いた例では 1/3 以下となることを示した．

本方式は現時点では独自方式の一つであり，実際には広く利用可能な符号化方式ではない．現在の EXI には独自拡張符号を導入する余地がないため，既存の EXI に対する独自拡張部分として利用することも困難である．従って，今後本方式を広く利用可能とするための働き掛けを，W3C を含めた関連標準化団体にて行う予定である．

参考文献

[1] Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core, IETF RFC 6120 (2011).
[2] The OpenADR Primer, OpenADR Alliance White Paper.
[3] Schneider, J. and Kamiya, T.: Efficient XML Interchange (EXI) Format, W3C Recommendation (2011).
[4] Doi, Y., Sato, Y. and Teramoto, K.: XML-Less EXI with Code Generation for Integration of Embedded Devices in Web Based Systems, *In Proc. of IoT2012 (The Third International conference on the Internet of Things)* (2012).
[5] Smart Energy 2.0 DRAFT 0.9 Public Application Profile, ZigBee Alliance Draft for Public Review (2012).
[6] 土井裕介, 佐藤弓子, 寺本圭一：XMPP の Schema-Informed EXI 利用における課題と解決, 情報処理学会研究報告 2012-IOT-20 (2013).