Vol.2013-IOT-20 No.25
2013/3/15

XMPPのSchema-Informed EXI利用における課題と解決

土井 裕介　佐藤 弓子　寺本 圭一

概要：一部のエネルギーマネジメント規格において，セッション管理に XMPP が，また，別の規格においては XML のコンパクト符号化に EXI がそれぞれ検討されている．これらを組み合わせることで高効率な M2M 通信を実現できる可能性があるが，現時点ではこれらの 2 つには仕様上の齟齬があり効率的に結合することは難しい．本稿では，この問題を EXI 側の仕様改善により解決できることを示し，それによる副作用について述べる．

Issues and Solutions for Use of Schema-Informed EXI on XMPP

Abstract: Energy management is a new and important application field of the network technologies. An energy management specification adopts XMPP as session management protocol. Another energy management specification adopts EXI as compact XML data representation. With the combination of XMPP and EXI, it is expected to enable highly efficient real time M2M communication platform. However, combination of XMPP and EXI is difficult because of some specification mismatches. In this paper, we describe the way to resolve the mismatches by changing EXI specification along with its side effects.

1. 背景

エネルギーマネジメント技術は，世界的な課題の一つである．特に，デマンドレスポンスと呼ばれる技術は，電力系統の安定性に需要家側が寄与することにより，より低コストで高い安定性の電力供給を可能とする技術である．

このためにさまざまな規格が討議されているが，米国 SGIP [*1]における Catalog of Standards に挙げられている規格に，OpenADR2.0 [1] ならびに SEP2.0[2] と呼ばれるものがある．これらは，それぞれ独立した課題から出発し，OpenADR2.0 は主に電力網側と需要家との間の電力マネジメント規格，SEP2.0 は需要家内での機器コントローラに対する情報伝達規格であるとされる．興味深い点は，これらは両方ともに XML をデータ符号化に XML を利用していることである．OpenADR2.0 はリアルタイム性を高めるために HTTP による通信の他に XMPP[3] (Extensible Messaging and Presence Protocol) をオプションとして検討しており，一方，SEP2.0 は無線メッシュネットワークを利用することが期待されることから，よりコンパクトな XML の符号化形式として EXI[4] (Efficient XML Interchange) をオプションとして採択している．

XML 技術であることから，XMPP と EXI を組み合わせることにより，リアルタイム性とコンパクトさを両立させることができるが，EXI のうち特に SEP2.0 が採用しているもっとも効率的なモードにおいては，これは困難である．本節では XMPP と EXI についてそれぞれ特徴を説明した後，節 2 で課題を詳細に述べ, 節 3 で EXI 仕様の拡張案を述べ，節 4 でこれらの解決案の副作用を，解決案を利用せずに EXI によるバイナリ符号化を実施した際と比較して説明する．

1.1 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

[1] 株式会社東芝 研究開発センター
[*1] Smart Grid Interoperability Panel

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．[*2] 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．

1.2 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．

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 [5]．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 による XMPP の運用

本節では，多数・多品種の組込機器に対するリアルタイム M2M 通信の枠組みとして，SI 文法による EXI による XMPP の運用における課題を検討する．

2.1 課題 1: 任意タイミングでのストリーム送出

節 1.1 で述べたように，XMPP は一つの TCP ストリームを一つの XML 文書とみなし，その中の要素を個別のメッセージとする形で，ポーリング等に比較してリアルタイム性の高い通信を実現している．メッセージを即時届けるためには，適切な要素の閉じタグの終了時点でフラッシュ (fflush および TCP PSH フラグ) により送信者から受信者にメッセージを届ける必要がある．

bit-pack オプションを利用した EXI の場合は，閉じタグの終了時点においてオクテット境界となるとは限らない．従って，ここでフラッシュを行うと，オクテット中の LSB 部分が不確定な状態でデータを送信してしまう．一方，受信側のデコーダは受信したオクテットの LSB が何であれ，これをデータとして EXI の解釈を進めてしまう．図 1 にこの様子を示す．

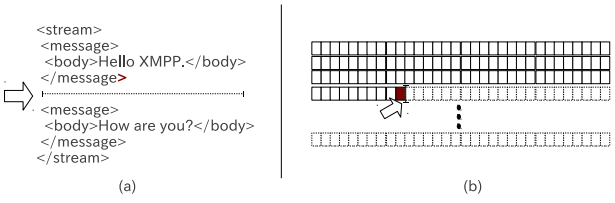

図 1 EXI におけるタグ境界とオクテット境界の関係例

図 1(a) において，XML 文書としては stream タグが閉じるところまでで一つの XML 文書として成立するが，XMPP はリアルタイムでのメッセージ送受信を行うため，例えば一つめのメッセージ送出時には message タグの閉じタグまで送出し，図中点線で示した場所でストリームが停止する．

[*2] http://xmpp.org/resources/schemas/

一方，bit-pack オプションによる EXI ではタグの終了が任意のビットに対応するため，図 1(b) に示すようにオクテットの途中の (LSB でない) ビットとなる場合のほうが多い．

従って，EXI で XMPP のような即時性のある XML メッセージングを実現するためには，byte-align オプションを利用するか，selfContained オプションを利用し，個別のメッセージを selfContained 要素として記述する必要がある．しかし，これらのオプションは符号の情報密度が低くなり，効率的でない．

2.2 課題 2: 複数のスキーマの扱い

節 1.1 で述べたように，XMPP は拡張可能なスキーマを定義することにより，拡張性の高いメッセージ構造を実現している．EXI により符号化することによりコンパクトな表現が期待できるが，現在の EXI 仕様はワイルドカード (`anyType`) で定義された要素については，UR-Type Grammar と呼ばれる動的なステートマシンを利用した符号化を行う仕様となっている ([4] Section 8.5.4.1.3.3)．

特に XMPP においては，図 2 に示すように，メッセージ中の多くの要素がワイルドカードにより導入されており，メッセージ中の UR-Type Grammar の利用比率は多くなる．その結果，オプションに至るまで厳格なスキーマを定義しているにも関わらず，SI 文法の利点を活用できない．

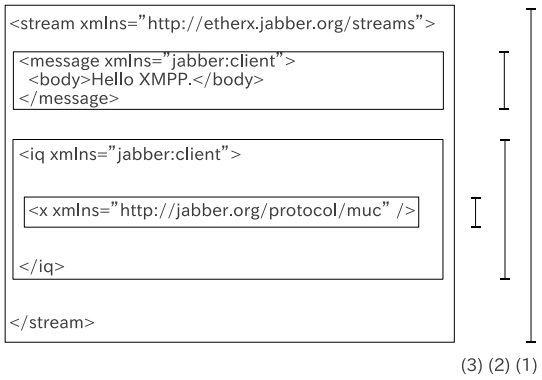

図 2 XMPP のストリーム中要素の再帰的定義の例: (1) がストリーム全体で，(2) が個々のメッセージおよび問い合わせ (`iq`) に対応し，(3) が個別拡張 (例では MUC: Multi User Chat) に対応する．

2.3 課題 3: ワイルドカードの取り扱い

EXI 仕様においては，XML Schema 仕様の 1.0 版を対象としている．一方で XMPP においては，XML Schema 仕様の 1.1 版を対象としている．XMPP においては拡張性を確保するために，ワイルドカード `anyType` を活用している．このワイルドカードの扱いにおいて XML Schema 1.0 と 1.1 で違いが存在し，XMPP の XML は EXI でそのまま符号化できない．

具体的にはこの問題は Unique Particle Attribution [*3] と呼ばれ，ワイルドカードの前後に存在しうる optional な要素が，ワイルドカードによって解釈することと，対応する明示的な要素定義によって解釈することとの 2 通りの解釈が可能となることに由来する問題である．XML Schema 1.1 において wildcard の解釈順位が弱められることによってこの問題は解決しており，その結果 XMPP はワイルドカードを自由に利用している．

この問題については EXI の次期仕様で解決する他なく，また XML Schema 1.1 は 2012 年の 4 月に Recommendation (W3C の標準化の最終段階) になったばかりであることから，本稿では範囲外とし，詳細は検討しない．

3. EXI 仕様の拡張案

前節で述べた EXI 仕様と XMPP の整合性問題のうち，節 2.1 で述べたオクテット境界の問題と，節 2.2 で述べた型の名前空間の問題について，本節で解決案を述べる．

3.1 解決のための基本方針

EXI 仕様の改善による，XMPP と EXI との整合性問題の解決方法であるが，以下の指針を持つこととする．まず，現時点での EXI 仕様を大幅に改変することは避けなければならない．また，可能な範囲で現在の EXI1.0 デコーダの仕様と整合性を取ることが望ましい．

節 2.3 で述べたとおり，参照している XML スキーマ仕様が異なるため実際には XMPP の XML を EXI にそのままエンコードしたとしても EXI1.0 と改良版の相互運用性を確保することは困難である．しかし，実装の大部分が再利用可能であれば，後方互換性の確保が容易となる．従って，生成文法のうち，他への影響が少ない部分に提案方式のための拡張規則を付加するというアプローチを取る．

ここで，EXI の文法の構造を説明する．EXI の文法は，個々の生成規則に対し順位が与えられており，生成規則に対応する符号 (イベントコード) はその順位によって異なる長さの符号が割り当てられる．このイベントコードは，上位の生成規則については短いものを利用し，下位の規則は上位の規則のイベントコードに下位のイベントコードを追記した形になる．特に，例外的な生成規則は下位に位置付けられ，長い符号となる．例えば，ある文法状態に対応する生成規則の集合が，図 3 に示すようにあったとき，CM(コメント) に対応するイベントコードは，カンマを結合 (concatenation) として，`1,3,0` となる．

この構造は，利用頻度が低い生成規則に関して長い符号を割り当てることにより，全体の符号長を小さくする，というものである．同時に，EXI の bit-pack オプションの利用時は，同順位の文法数を表現するのに必要十分なビット

[*3] `http://www.w3.org/wiki/UniqueParticleAttribution`

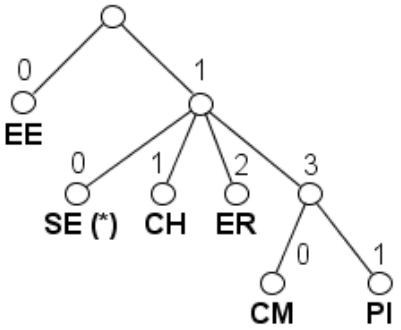

図 3 生成規則の順位とイベントコードの関係 ([4] Figure 8-1 より引用)

長の整数によってイベントコードを符号化するため，最上位の生成規則はなるべく小さくしておくことが望ましい．例えば，前述のイベントコード 1,3,0 は，実際には二進数で表現すると 4 ビットの 1110(1,11,0) となる．

EXI の仕様においては，文法上の特殊ケース (selfContained など) については第二位以下の生成規則としている．本稿でも同様に，解決に必要な生成規則がある場合は，これを第二位の生成規則として追記することをもって解決する．

3.2 NOP によるオクテット境界

節 2.1 で述べたオクテット境界の問題については，NOP (No Operation) の規則の導入によって行う．EXI エンコーダを利用するアプリケーション側から明示的にフラッシュの要求があった時に，オクテット境界をまたぐまで副作用のない NOP を記述する．エンコーダ・デコーダは，NOP に対してはデコーダの状態を変化させず，また NOP はいかなる XML 要素にも対応しない．

NOP の記述には，第二位の生成規則に副作用のない NOP 生成規則を追加する必要がある．

本方式の利点は，簡単な文法の変更で問題を解決できる点である．一方，本方式には以下に述べる 2 つの欠点がある．

(1) 同じ情報が同じ形に符号化されるとは限らない
(2) 冗長な符号が挿入されるため，サイズが長くなる

欠点 1 については，EXI の場合は元々 XML が同一であっても利用する生成規則が異なる場合は異なる符号になる場合がある (詳細は省略)．従って，EXI においては，元々符号レベルでの比較が難しく，同一符号であることをもって同一内容であるとする実装戦略をとることは難しい．以上から，実質的には問題にならないと考えている．欠点 2 については，節 4 で検討する．

3.3 拡張スキーマの導入

節 2.2 で述べたように，XMPP においてはワイルドカードにより拡張要素を導入している．XMPP では個々の拡張要素に明確なスキーマが割り振られているので，これを SI 文法で記述する方法を検討する．

まず，既存の SI 文法においては，文法生成時に対応するスキーマと，これに由来する型文法 (Type Grammar) が確定していることを前提としている．これを拡張するために，以下の手続きを考える．以下，EXI ストリーム開始時のスキーマを基本スキーマ (XMPP で言う `http://etherx.jabber.org/streams` 名前空間に対応するもの) とし，これに対して拡張機能に対応し，動的に参照されるスキーマを拡張スキーマ (XMPP において，例えば `http://jabber.org/protocol/muc` に対応するもの) と呼ぶことにする．また，拡張スキーマにはそれぞれ ID が割り当てられるものとする．

エンコーダは，XML 文書をエンコード中，拡張スキーマによって定義された要素を発見したら，これを UR-Type Grammar でエンコードするか，拡張スキーマによってエンコードするかを決定する．拡張スキーマによるエンコードを指示するために，SI 文法の第二位もしくは第三位に，拡張要素を終端記号とする生成規則を追加する．

このとき，デコーダは拡張スキーマを知っているかもしれないし，知らないかもしれない．そこで，拡張スキーマ利用時の拡張要素は以下の内容から構成することとする．

- 拡張要素ストリーム長 (Integer): bit-pack オプション時はビット数，byte-align オプションの場合はオクテット数
- 拡張スキーマ ID (String): 拡張スキーマの ID
- 拡張要素の EXI ストリーム

デコーダが未知の拡張スキーマ ID を発見した場合は，データ長分だけデータの読み込みをスキップすれば良い．この構造により，エンコーダはデコーダが持っている拡張機能の有無によらず，拡張機能を SI 文法で符号化することができる．

本方式の欠点は，拡張要素ストリーム長・スキーマ ID の記述分だけストリームが長くなることである．節 4.2 にて検証する．

4. 評価

評価においては，動的な評価の環境が整っていないため，静的な評価による各方式によるオーバーヘッドの評価を行う．対象として，XMPP サーバとの間のチャット・マルチユーザチャットのログを tcpdump で取得し，ペイロード XML ならびに XMPP のスキーマを修正したものを利用した．なお，再構成は，一部スキーマに合致しない要素があった点について，順序の変更を行い修正を行った点と，Unique Particle Attribution 問題を解決するため，XML スキーマの wildcard 定義を一つコメントアウトしたものである．最終的な XML は 3601 オクテットとなり，

なお，EXI エンコーダは EXIficient [*4]，XMPP サーバは jabberd2 2.0 [*5]，クライアントは pidgin 2.7.3 [*6] を利用した．

なお，別途記述のない限り，SI 文法を生成する XML スキーマは，XMPP の stream スキーマを基礎とし，ここから直接 import 関係があるものについて，schemaLocation をローカルファイルに書き換えたものであり，8 つの名前空間からなる．これをスキーマ A と呼ぶ．

また，現時点で実現できる XMPP のバイナリストリームにもっとも近い，byte-align オプションを利用したスキーマ A による符号化したものをベースライン (ストリーム B) とし，節 3 で提案する方式によるものをそれぞれ提案ストリーム p_n とし，これを算出するために実際に EXIficient により符号化したストリームを理想ストリーム i_n とする．ストリーム i_n は EXI による符号化という観点から見た理想状態であり，実際には節 2 で述べた問題により XMPP の符号化には利用できない．

4.1 NOP によるストリーム長の変化

節 3.2 で述べた NOP による flush は，XMPP のやりとりの単位となるタイミングで NOP の生成規則を用いることで副作用のないビットを EXI ストリームに挿入し，flush すべき要素を含むオクテットをストリームで埋めることにより実現する．

ここで，理想的な NOP は 1 ビット単位で導入できるとすれば，平均した NOP によるオーバーヘッドは 1 メッセージあたり 4bit と言える．しかし，実際には文法構造によって 3 ビットから 5 ビット程度が単位となるため，ここでは保守的に，1 回の flush によって 8 ビットのオーバーヘッドがあると仮定する．この仮定のもとで，byte-align オプションを利用した EXI ストリーム (BASE) と NOP により flush を実現した bit-pack オプションを利用した EXI ストリーム (p_1) とのサイズを比較する (表 1)．ここで，flush は tcpdump によるトレースから，PSH フラグを設定した数により推測し，25 回の flush が必要だとする．これを，スキーマ A を利用し bit-pack オプションにより符号化を行ったストリーム (i_1) のサイズに加算することで，p_1 のサイズを推測した．

また，selfContained オプションにより符号化したものを SC として示した．selfContained 要素においては，繰り返し登場する文字列の圧縮が行えないことから，圧縮率はストリーム BASE に劣る．

4.2 サブスキーマ導入によるストリーム長の変化

サブスキーマ導入によるストリーム長の変化は見積りが難しい．節 3.3 で述べたように，現時点では XMPP の EXI による符号化が仕様上困難であることによる．そこで，本稿では，以下の手順で簡易な評価を行った．

表 1 NOP によるストリーム長の変化 (スキーマ A)

ストリーム	長さ (オクテット)	BASE 比 (%)
BASE	1358	100
i_1	1201	88
p_1	1226	90
SC	1589	117

まず，サブスキーマが存在しない，現状の EXI 仕様による XMPP の EXI 符号化を想定する．

全ての要素をスキーマ情報を利用した SI 文法で符号化できたと仮定する．現状の EXI 符号化においては，SI 文法作成時に，利用する全て要素に対応する型文法を含め，また，ワイルドカードについて processContents="strict" と指定し，かならず型情報を参照して validate するよう指示することで対応できる．本稿ではこれをスキーマ B と呼ぶ．今回の評価ケースでは，スキーマ B は 22 の名前空間から成る．スキーマ B に由来する SI 文法を用い，byte-align オプションにより符号化されたストリームをストリーム i_2，bit-pack オプションにより符号化されたストリームをストリーム i_3 とする (表 2)．

なお，実際には節 3.3 で述べたように，これらに加えて個別の拡張要素毎に拡張要素のヘッダ (データ長およびスキーマ ID) を含むことになる．スキーマ ID はアプリケーション毎に自由に決定できるため，長さを仮定することは難しいが，合理的に考えて，URI のような長いスキーマ ID を採択することは考えづらい．ここでは仮に拡張スキーマ ID を平均 3 オクテット (うち 1 オクテットは文字列長)，データ長を 1 オクテット (EXI の整数は 8 ビットを単位とする可変長)，拡張要素の生成規則の符号を，bit-pack オプションでは 0.5 オクテット (4 ビット)，byte-align オプションでは 1 オクテットで表現できたとする．XML の要素のうち，スキーマ A に含まれない型が出てくるたびに拡張要素ヘッダを記述する．評価対象の XML においてはこのような拡張型は 20 回登場することから，ストリーム i_2, i_3 にそれぞれ対応するオーバーヘッドを付加したものをストリーム p_2, p_3 とする (表 2)．

表 2 スキーマ B によるストリーム長

ストリーム	長さ (オクテット)	BASE 比 (%)
i_2	895	66
p_2	995	73
i_3	719	53
p_3	809	60

4.3 考察

表 1 および表 2 のうち，提案方式に関わるもののスト

[*4] http://exificient.sourceforge.net/
[*5] http://jabberd2.org/
[*6] http://www.pidgin.im/

リーム長を図 4 にまとめた．

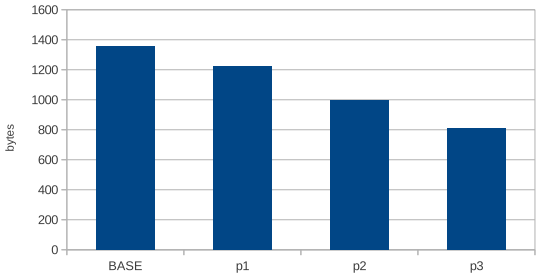

図 4 ベースラインと各提案方式のストリーム長の比較

まず，現状の仕様のうち，課題 3(節 2.3) のみを解決したものがベースライン (BASE) である．課題 1(節 2.1) を NOP 生成規則により解決したものはスキーマ A によって符号化されたストリーム p_1 に相当し，課題 2(節 2.2) を拡張スキーマの導入により解決したものは，スキーマ B によって符号化されたストリーム p_2 に，課題 1,2 の両方を解決したものは同じくスキーマ B によって符号化されたストリーム p_3 に相当する．それぞれ，ベースライン比が $90\%, 73\%, 60\%$ となり，個別の課題解決よりも組合せ課題の解決のほうがより効果的であることがわかる．

5. 結論と今後の課題

本研究では，スマートエネルギー関連技術に求められる XMPP と EXI という 2 つの XML 技術について，仕様の齟齬を明確化した．XMPP と EXI の組み合わせにより，組み込み機器を含めた幅広い機器間で，明確な仕様に裏打ちされた拡張性の高いメッセージを交換可能になり，Internet of Things / 機器間通信 (M2M) の可能性・可用性・生産性が大きく高まると期待している．

具体的には，EXI の仕様の拡張により XMPP を EXI の SI 文法により利用するための障壁として，オクテット境界の問題，拡張スキーマの問題，ワイルドカードの取り扱いの問題の 3 点を述べ，うちワイルドカードの取り扱いを除く 2 つについて解決案を提示した．

EXI については，仕様が複雑であることから，実装が複雑になりがちであること，また，EXI を用いた XMPP の実際の実装においては XML ストリーム API によるやりとりが中心となることから，実装の組み合わせが事実上存在しない．今回は通常の XML を静的に EXI で符号化し，ここから仕様変更によるオーバーヘッドの期待値を加え，評価した．その結果，本稿で述べた課題 1, 2 の解決により，ベースライン比 60%までストリームサイズを圧縮できることが推定できた．

本技術は第一にスマートエネルギー関連技術を対象とするものであるが，分散環境における機器間通信に対して有効な組み合わせであると考える．今後は，本問題提起を契機に本提案の仕様への取り込みに向けた活動や，実装にもとづく評価を行いたいと考えている．

参考文献

[1] The OpenADR Primer, OpenADR Alliance White Paper.
[2] Smart Energy 2.0 DRAFT 0.9 Public Application Profile, ZigBee Alliance Draft for Public Review (2012).
[3] Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core, IETF RFC 6120 (2011).
[4] Schneider, J. and Kamiya, T.: Efficient XML Interchange (EXI) Format, W3C Recommendation (2011).
[5] 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).