Vol.2012-GN-82 No.25
Vol.2012-CDS-3 No.25
2012/1/20

XML-Less EXI を搭載した 家電用通信アダプタの試作および評価

佐藤弓子† 土井裕介† 寺本圭一†

家庭内の機器に通信機能を搭載し各機器を協調制御させることにより，消費エネルギーの削減や創蓄エネルギーを組み合わせた最適化などに寄与するエネルギーマネジメントシステムが注目を集めている．

エネルギーマネジメント向けの通信規格の 1 つである ZigBee SEP2.0 のデータ形式として，XML をコンパクトに符号化した EXI の採用が検討されている．EXI 自体，メッセージサイズは小さいものの，符号化前や復号化後に自由度の高い XML データモデルを扱わなければならないため，メモリや CPU などの資源に厳しい制約のある組込み機器へ単純に搭載するには困難がある．

本研究では, 直接 EXI を操作可能にする XML-Less EXI を提案し, ZigBee SEP2.0 で利用するデータ形式に対応した通信アダプタを試作後，その有効性について評価する．

Implementation and Evaluation of Network Adapter with XML-Less EXI for Home Appliances

Yumiko Sato,† Yusuke Doi,† and Keiichi Teramoto†

There are many communication standards to control home appliances to reduce energy consumption. Among them, ZigBee Smart Energy Profile 2.0 (SEP2.0) is one of the communication standards for Home Energy Management System (HEMS).ZigBee SEP2.0 uses the recently adopted Efficient XML Interchange (EXI) standard for serializing XML messages. However, some home appliances and embedded devices do not have enough computing resource to handle complex XML-based data structure with high degree of freedom. In this research, we implemented and evaluated a communication adapter for home appliances that handles SEP2-stype data structure.

1. はじめに

家庭内の機器に通信機能を搭載し，各機器を協調制御させることにより消費エネルギーの削減や創蓄エネルギーを組み合わせた最適化などに寄与するエネルギーマネジメント（HEMS: Home Energy Management System）が注目を集めている．HEMS 向け通信規格は複数存在し, 国内向けには ECHONET[a][1], 北米向けには ZigBee[b] SEP2.0（Smart Energy Profile 2.0）[2]などが挙げられる．基本的に，機器間にて協調制御をするには，各機器が同じ通信規格に準拠している必要がある．これに対し，我々は機器側に異なる通信規格へ変換可能な通信アダプタを取り付けることによって，宅内ネットワーク内にて通信規格を合わせる方式を提案した．具体的には，ECHONET ミドルウェアアダプタを活用した，図 1 に示す ECHONET-ZigBee SEP2.0 通信アダプタを試作し，その動作を確かめた[3].

しかし，通信アダプタのような組込み機器は，メモリや CPU などの資源に厳しい制約を伴うことが多い．ECHONET は独自のデータ形式であり，データサイズが小さく処理も軽いため，組込みに向いている．一方，ZigBee SEP2.0 はデータ形式に XML（Extensible Markup Language）[4]または EXI（Efficient XML Interchange）[5]を採用することが検討されている．XML は汎用性が高い分，データが冗長であるため，データサイズが大きくなるばかりかその処理も軽いとはいえず，組込みには向いていない．EXI は XML をコンパクトに符号化したものであるためデータサイズは小さいが，符号化前や復号化後に XML と同等のデータモデル (DOM, SAX, StAX など) に関する処理が必要になる．

我々は，制約の高い組込み機器でも扱いやすい EXI として，負荷が高くなりがちな XML データモデルの処理を避け, 直接 EXI を操作する XML-Less EXI を提案している[6][7]．本研究では，実装として，ECHONET-ZigBee SEP2.0 通信アダプタに XML-Less EXI 実装し，その有効性について評価する．

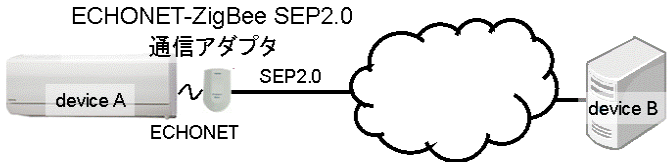

図 1. ECHONET-ZigBee SEP2.0 通信アダプタを含むホームネットワークの構成例

† 東芝研究開発センターネットワークシステムラボラトリー
Network Systems Laboratory Corporate Research & Development Center Toshiba Corporation

a) ECHONET は ECHONET コンソーシアムの登録商標
b) ZigBee は ZigBee Alliance, Inc.の登録商標

Vol.2012-GN-82 No.25
Vol.2012-CDS-3 No.25
2012/1/20

2. EXI の基本動作

ZigBee SEP2.0 は ZigBee Alliance で策定中の HEMS 向け規格の一つである．ZigBee や IPv6，TCP，HTTP など多層にわたるプロトコルと，緊急ピーク時間帯料金の表示や課金，デマンドレスポンスなどのファンクションセットからなるプロファイルである．ZigBee SEP2.0 で用いられる予定のデータ形式は XML または EXI である．

EXI は 2011 年 3 月に W3C にて勧告された XML のバイナリ符号化規格の 1 つである．符号化された XML データを EXI ストリームといい，EXI ストリームは XML データよりもデータサイズが格段に小さくなる．それは，終了タグのような出現頻度が高い情報には短い bit を割り当て，コメントのような出現頻度が低い情報には長い bit を割り当てているため，全体としてコンパクトになるからである．この出現する情報と bit 割り当て規則 (Grammar) には Build-in Grammar と Schema-Informed Grammar の 2 種類がある．Schema-Informed Grammar には，XML Schema に沿わないデータを許可するモード（non-strict モード）と，許可しないモード（strict モード）がある．

Build-in Grammar では XML Schema を利用せず, 読み込んだ XML データ／EXI ストリームから，出現候補を学習していく．どのような XML データでも符号化でき，Build-in Grammar で符号化された EXI ストリームを復号できる．要素や属性の名前を知らないため，要素名・属性名など，XML データの構造を表現するための文字列の情報を EXI ストリームに埋め込む必要があり，Schema-Informed で符号化したものに比べてデータサイズが大きくなる．また，学習によって Grammar が成長するため，実装時の使用メモリ量が予測できない．

一方 Schema-Informed Grammar では，XML Schema を利用しているため，出現候補ははじめからすべて分かっている．XML Schema で定義されている XML データであれば符号化でき，同じ XML Schema を利用して符号化された EXI ストリームを復号できる．strict モードの場合は同じ XML Schema を利用して符号化された EXI ストリーム以外は復号しないが，non-strict モードの場合は XML Schema から外れた部分は Built-in Grammar に移行して対応する．XML Schema によって要素や属性の名前が分かっているため，EXI ストリームに文字列を埋め込む必要がなく，Built-in Grammar での符号化に比べてデータサイズが小さくなる．また，Grammar は成長しないため，実装時の使用メモリ量は固定である．

以上に述べたなかで，よりデータサイズが小さく，予期しないメモリ量の増加が少ない Schema-Informed Grammar（strict モード）が最も組込み機器に適していると考えられる．

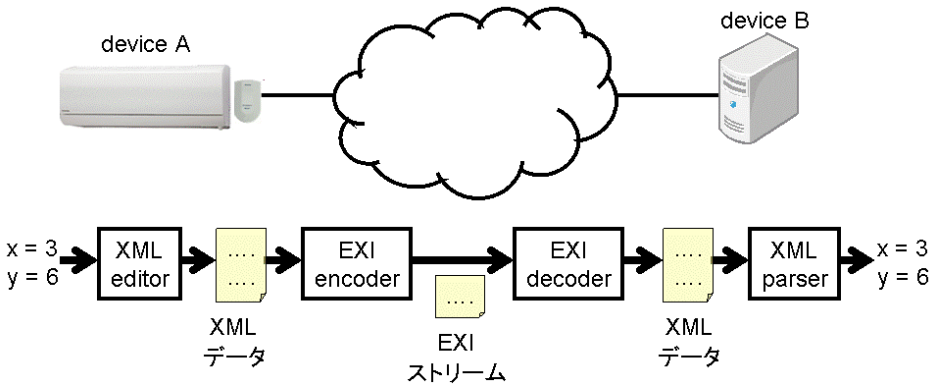

図 2. EXI の利用例

一般的に EXI は次のようなステップで利用される．（図 2）

<送信する場合>
1.値を XML (あるいは等価なデータモデル) に整形
2. XML データを EXI ストリームにエンコード
3. EXI ストリームを送信

<受信する場合>
1. EXI ストリームを受信
2. EXI ストリームを XML あるいは等価なデータモデルにデコード
3. XML データをパースし値を抽出

EXI の利点は，データを圧縮できることであり，結果，通信路のトラフィック量を少なくすることができる．これは，IEEE 802.15.4 のように送信フレームサイズが小さな無線通信系においては大きな利点である．

ただし，組込み機器で従来用いられてきた ECHONET のようなフォーマットと比較すると，XML のデータモデルは複雑であり，限られた計算資源で完全な処理を行うことは難しい場合がある．

Vol.2012-GN-82 No.25
Vol.2012-CDS-3 No.25
2012/1/20

3. XML-Less EXI の設計

本研究では，XML 処理を避け，EXI を直接操作可能にする，組込み機器向けの XML-Less EXI を構築する．本稿では，Schema-Informed Grammar（strict モード）版 EXI に基づいた XML-Less EXI を設計，実装する．

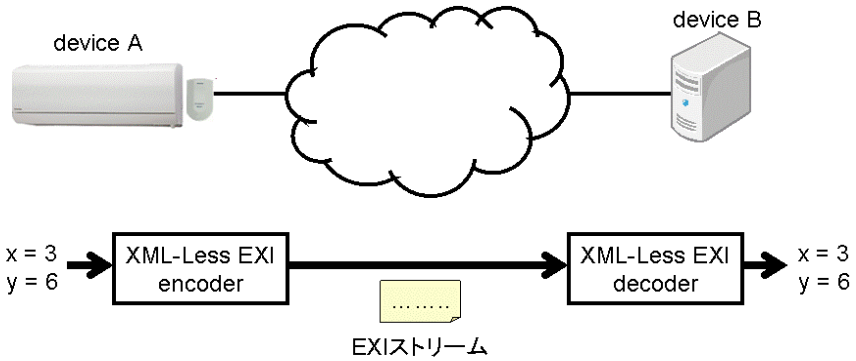

図 3. XML-Less EXI の利用例

図 3 に示すように，XML-Less EXI を次のようなステップで利用できるように設計する．

<送信する場合>
1. デバイスの値を EXI に変換し（XML-Less EXI encoder），雛形に埋め込む
2. EXI ストリーム送信

<受信する場合>
1. EXI ストリームを受信
2. EXI ストリームを解析して値を抽出（XML-Less EXI decoder）

このように，XML-Less EXI では XML データのパース処理などが不要となり，直接 EXI ストリームとデバイス値との変換が可能となる．
以降では XML-Less EXI encoder および decoder の設計について述べる．

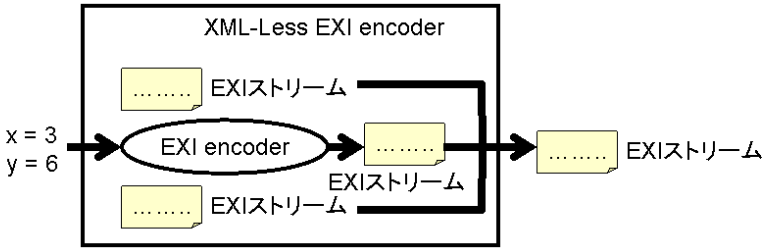

図 4. XML-Less EXI encoder の構成と動作

データ形式に XML を用いているほとんどの機器において，XML データの構造はあまり変わらない．また，XML データの各要素の値もすべてが変わるわけではない．たとえば，データを作成した時間を示す要素の値は，データが作成されるたびに変わり，データを作成した機器を示す要素の値は，その機器においては変わることはない．このような XML データを EXI ストリームに符号化する場合，構成も値も変化しない部分は，EXI ストリームも変化せず，構成や値が変化する部分は，EXI ストリームも変化する．こうした特徴に基づいて XML-Less EXI encoder（図 4）を作成する．XML-Less EXI encoder は，XML データの変化しない部分を前もって符号化していた EXI ストリームと，変化する部分を毎回 EXI ストリームに符号化する EXI encoder から構成される．XML-Less EXI encoder の動作は，機器から得た値を符号化した EXI ストリームと，前もって符号化していた EXI ストリームとをあわせて一つの EXI ストリームとして出力する，というものである．

XML-Less EXI encoder は，固定部分の XML データの構成や値を変更したい場合には作成しなおす必要がある．変更があるたびに開発者が毎回 XML-Less EXI encoder を作成するのは時間や手間がかかる．これを軽減するため，XML-Less EXI encoder を自動生成する XML-Less EXI encoder generator（図 5）を作成する．XML-Less EXI encoder generator は XML データの雛形と XML Schema と設計ファイルをもとに XML-Less EXI encoder を作成する．設計ファイルには機器が扱う変数（型と変数名）と，EXI ストリームに符号化する際の型と XML の構造（要素）の対応づけを記述しておく．XML-Less EXI encoder generator は，まず XML Schema から Grammar を生成し，次に Grammar と XML データの雛形から固定部分の EXI ストリームを生成し，最後に Grammar と設計ファイルから EXI encoder を生成する．

Vol.2012-GN-82 No.25
Vol.2012-CDS-3 No.25
2012/1/20

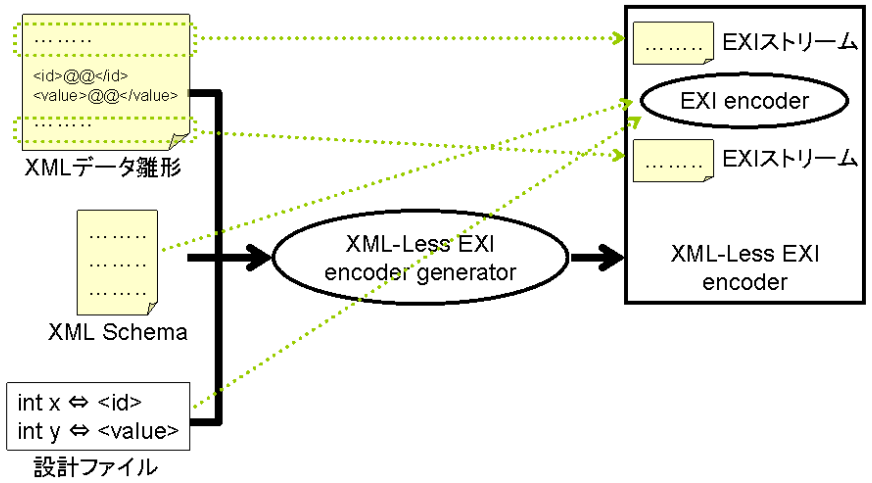

図 5. XML-Less EXI encoder generator の入出力

開発者が EXI ストリームから値を取り出してアプリケーションで利用する場合，すべての値を取り出す必要はなく，抽出したい値だけが取り出せればよい．こうした特徴に基づいて XML-Less EXI decoder（図 6）を作成する．XML-Less EXI decoder は，EXI ストリームを復号する EXI decoder と，復号した結果が抽出したい値であった場合に取り出す関数から構成される．XML-Less EXI decoder の動作は，読み込んだ EXI ストリームを EXI decoder で復号し，復号した後に抽出したい値を取り出し，それ以外の値を読み飛ばす，というものである．

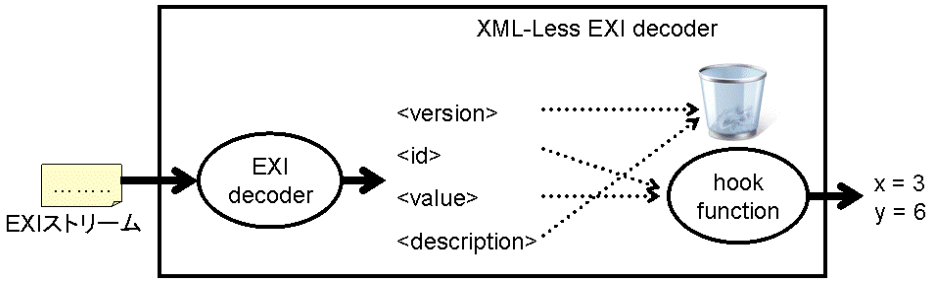

図 6. XML-Less EXI decoder の構成と動作

XML-Less EXI decoder は，抽出したい値を変更したい場合には作成しなおす必要があり，開発者の時間と手間がかかる．これを軽減するため，XML-Less EXI decoder を自動生成する XML-Less EXI decoder generator（図 7）を作成する．XML-Less EXI decoder generator は XML Schema と設計ファイルをもとに XML-Less EXI decoder を作成する．設計ファイルには機器が扱うデータ形式（構造体型と各メンバ名）ならびに，EXI ストリーム中のデータ型と XML の構造（要素）の対応づけを記述しておく. XML-Less EXI decoder generator は, まず XML Schema から Grammar を生成し，次に Grammar から EXI decoder を生成し，最後に Grammar と設計ファイルから値を取り出す関数(hook function)を生成する．

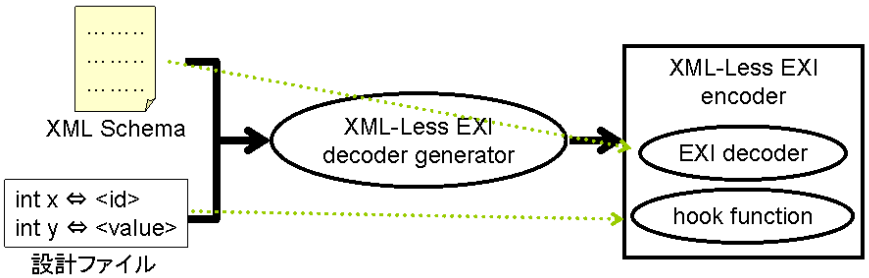

図 7. XML-Less EXI decoder generator の入出力

4. 通信アダプタへの XML-Less EXI の実装

これまでに示した内容に基づき，ECHONET-ZigBee SEP2.0 通信アダプタ上に XML-Less EXI を実装する．ここでは，ZigBee SEP2.0 コントローラみなしの PC に XML-Less EXI encoder を実装し，[3]で試作した ECHONET-ZigBee SEP2.0 通信アダプタに XML-Less EXI decoder を実装する．アプリケーションは ZigBee SEP2.0 デマンドレスポンスとする．デマンドレスポンスは端末がコントローラから消費エネルギーの削減依頼を受け，それに対応する機能である．

XML-Less EXI encoder の実装に先立ち，雛形となる XML データと XML Schema，設定ファイルを用意する．空調の設定温度を変更することで，指定した時間，エネルギーを削減するデマンドレスポンスに対応する XML データの雛形を次に示す. なお，次のデータはあくまで例であり，ZigBee Alliance の定める XML Schema に沿ったものではない．

Vol.2012-GN-82 No.25
Vol.2012-CDS-3 No.25
2012/1/20

```
<?xml version="1.0" encoding="UTF-8"?>
<devicecontrol>
    <:devicecategory>thermostat</devicecategory>
    <scheduled>
        <duration>3600</duration>
        <start>1234567890</start>
    </scheduled>
    <setpoint>
        <type>heater</type>
        <value>20</value>
    </setpoint>
</devicecontrol>
```

要素 devicecontrol がルートタグとなり，その子要素に devicecategory や scheduled, setpoint がある．要素 devicecategory は端末の種類を，要素 scheduled はエネルギー削減の開始時刻と継続時間を，要素 setpoint は設定モードと設定温度を示す．動作確認のための実装として，本研究では要素 scheduled の子要素 start（開始時刻）の値と要素 setpoint の子要素 value（設定温度）の値のみを変化させることとする．

機器が扱う変数（型と変数名）と，EXI ストリームに符号化する際の型と XML の構造（要素）の対応づけを指定する設計ファイルを次に示す．

```
<?xml version="1.0"?>
<struct>
 <name>devicecontrol</name>
 <target>devicecontrol</target>
 <entries>
  <entry>
   <key>start</key>
   <type>unsigned int</type>
   <convfunc>conv_uint32</convfunc>
   <target>/scheduled/start/text()</target>
  </entry>
  <entry>
   <key>value</key>
   <type>unsigned int</type>
   <convfunc>conv_uint32</convfunc>
   <target>/setpoint/value/text()</target>
  </entry>
 </entries>
</struct>
```

要素 struct が設定ファイルのルートタグとなる．struct の子要素 name は機器が用意する値を格納する構造体の名前に相当する．子要素 target と entries では雛形のどの要素に対して EXI 符号化するかを指定する．entries の子要素 entry では具体的に変数（型と変数名）や符号化するための関数を指定する．これらのファイルから XML-Less EXI encoder を生成する．開発者は，構造体メンバ devicecontrol.start と devicecontrol.value に値を与え，固定部分の EXI ストリーム書き出し関数と値の符号化関数を呼び出すことによって EXI ストリームを生成できる．

XML-Less EXI decoder の実装に先立ち，XML Schema と設定ファイルを用意する．XML Schema は XML-Less EXI encoder と同一のものである．以下に示す設計ファイルも，符号化するための関数が復号化するための関数に変わっている以外は XML-Less EXI encoder の設定ファイルと同様である．

```
<?xml version="1.0"?>
<struct>
 <name>target_data</name>
 <target>/devicecontrol</target>
 <entries>
  <entry>
   <key>start</key>
   <type>unsigned int</type>
   <convfunc>read_uint</convfunc>
   <target>/scheduled/start/text()</target>
  </entry>
  <entry>
   <key>value</key>
   <type>unsigned int</type>
   <convfunc>read_uint</convfunc>
   <target>/setpoint/value/text()</target>
  </entry>
 </entries>
</struct>
```

これらのファイルから，XML Schema に従う EXI ストリームを復号する XML-Less EXI decoder を生成する．EXI ストリームを復号するにつれ，コールバック関数を経由して start と value の値を取得できる．開発者は，これらの値を直接利用するアプリケーション部分のみを記述すれば EXI 対応のプログラムを実装することができる．

Vol.2012-GN-82 No.25
Vol.2012-CDS-3 No.25
2012/1/20

5. 評価

本評価にて使用した ECHONET のメッセージサイズと EXI ストリームのサイズ，XML メッセージのサイズを図 8 に示す．XML および EXI の値は ZigBee SEP2.0 デマンドレスポンスの削減要求の内容を示す devicecontrol の大きさを示す．ZigBee SEP2.0 では命令の記述粒度が異なるため，ECHONET のメッセージサイズは等価な命令に相当する 2 回分のメッセージサイズを示す．図 8 から，XML メッセージのサイズに比べ，EXI ストリームのサイズは 10 分の 1 以下であり，ECHONET のメッセージサイズに近く, ECHONET-ZigBee SEP2.0 通信アダプタにおいて EXI を用いることは有効であると考えられる．

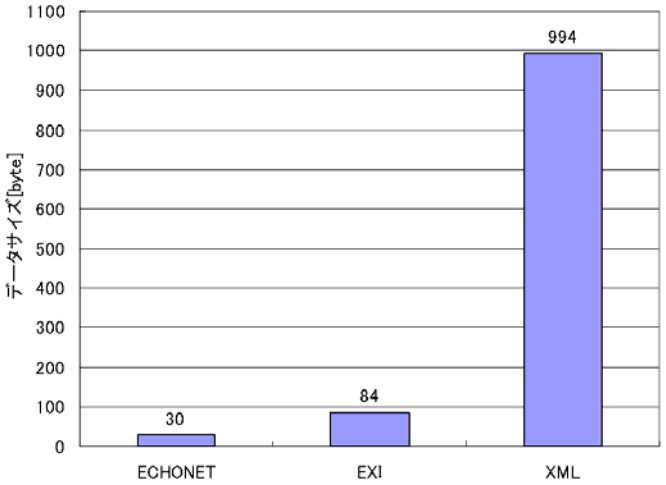

図 8　ECHONET と EXI と XML のメッセージサイズの比較

XML-Less EXI の実装において，実装の容易さについて評価する．XML-Less EXI encoder generator および decoder generator では，必要な設計ファイルを用意するだけで XML-Less EXI encoder および decoder の API が自動生成され，開発の時間が短縮される．開発者がアプリケーションで XML-Less EXI encoder を用いる場合，固定部分の EXI ストリームを出力する API と，値を与えた後に，変化部分の EXI ストリームを出力する API を使用するだけである．さらに EXI における型を意識せずに利用できる．開発者がアプリケーションで XML-Less EXI decoder を用いる場合，取り出された値を処理する関数を書くほかは EXI ストリームを解析する API を使用するだけである．このように, 組込みに対応した実装でありながら, その実装が容易であることが言える．

XML-Less EXI がコンパクトに実装できることを確認するために実行ファイルサイズについて評価する．通信アダプタ用に C 言語で実装した ZigBee SEP2.0 デマンドレスポンスの XML-Less EXI encoder の実行ファイルサイズを図 9(a)に，XML-Less EXI decoder の実行ファイルサイズを図 9(b)に示す．C 言語で実装された EXI のオープンソースは XML Schema に対応不可であったため, 参考として C 言語で実装された XML のオープンソース expat2.0.1[8]を用いた XML 解析プログラムの実行ファイルサイズを図 9(c)に示す．XML-Less EXI encoder は devicecontrol の EXI ストリームを出力するプログラム，XML-Less EXI decoder は EXI ストリームから devicecontrol の start と value の値を取り出すプログラム, XML 解析は devicecontrol の XML データから start と value の値を取り出すプログラムである．図 9 の値はそれぞれのプログラムを動的リンクでコンパイルしたときのサイズである．XML 解析のサイズには，プログラムの実行ファイルのほかにライブラリ libexpat.so.1 のサイズが含まれる．

図 9 から，XML 解析に比べ，XML-Less EXI decoder の実行ファイルサイズは 5 分の 1 以下であり，EXI ストリームを直接操作するほうがコンパクトな実装が出来ることが分かる．このことから，ECHONET-ZigBee SEP2.0 通信アダプタのような組込み機器にも適していると考えられる．また，XML-Less EXI encoder は 8[KB]と非常に小さく実装できるため，定期的に値を送信するセンサのような組込み機器には適していると考えられる．

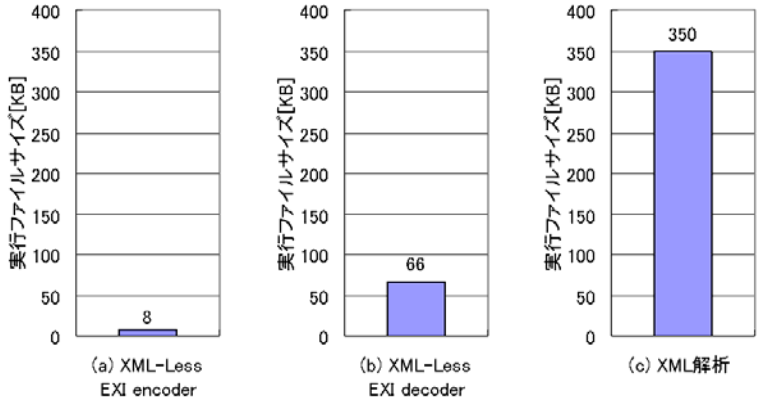

図 9.　XML parser と XML-Less EXI の実行ファイルサイズ

Vol.2012-GN-82 No.25
Vol.2012-CDS-3 No.25
2012/1/20

6. 標準化活動との関係

以上に述べたように，EXI の活用は計算資源の限られた組込み機器において，XML データモデルの利用方法として有効であることがわかった．一方で，ZigBee SEP2.0 の標準化活動において，このような EXI の利点は必ずしも広く認知されていなかったと言える．実際，ZigBee SEP2.0 について 3 月の時点で公開されたアプリケーション仕様[9]は，EXI や軽量アプリケーションプロトコルである CoAP (Constrained Application Protocol) [10]が仕様策定の場で利点が十分認知されず，規格案から削除されていた．

ZigBee SEP2.0 のような応用において．EXI のメッセージがコンパクトであること，省メモリであることに加えて，一般に XML の利点のうち「ベンダ拡張」が自由に行えること，そのために名前空間による拡張部分の明確化を行うことが重要であると考えられる．EXI の省メモリ性を維持するため，以下のような構成が適切であると考える．

- λ 省メモリ性を追求するために，Built-in Grammar の利用を行わない
  組込み文法は動的に成長するため，メモリ利用量の上限を事前に規定することが難しい．また，特に堅牢性の観点から，特定の組込み環境においては動的メモリ確保を嫌う傾向があり，これに対応するためでもある．
- λ XML Schema 拡張は仕様で定義される xsi:type による派生型指定を用いる
  EXI の導入の際，重要な論点として，将来の拡張性を確保すること，という論点が明示された．Built-in Grammar を排した，strict モードによる Schema-infomred Grammar は，通常の方法では拡張が困難である．特に，複数の拡張に対応する XML Schema に対応するためには，複数の XML Schema に対応する文法を ROM に格納する必要があり，組込み機器においては実装コストとなりうる．ところが，XML Schema 規格において，xs:type による派生型の直接指定が可能であることが EXI にも記述されており，これを用いた拡張を行うことで，大半のデータ型において共通部分と派生部分といった形での Grammar の記述が可能になる．これにより，strict モードによる Schema-infomred Grammar においても，高い拡張性を維持しながら Grammar のサイズに対する影響を最小限に留められる．

ZigBee SEP2.0 の正式版において，どのような形で EXI が採用されるかは執筆時点では不明確である．しかし，ZigBee SEP2.0 が本来ターゲットとしている組込み向けシステムへの XML 適用において，本研究の成果を活かせるよう，標準化の場において適切な形でのフィードバックを進めている．

7. おわりに

ECHONET-ZigBee SEP2.0 通信アダプタ上に制約のある組込み機器でも扱いやすい XML-Less EXI を実装し，その有効性について評価した．

メッセージを EXI ストリームにすることによって，データ削減の効果を得ることができた．XML-Less EXI によって，ZigBee SEP2.0 デマンドレスポンスのデータを EXI で取り扱う機能を通信アダプタにコンパクトに実装することができた．また，開発者は，XML-Less EXI による簡単な API を用いることで EXI を利用できるため，開発コストを削減することができる．

仮に ZigBee SEP2.0 以外のメッセージ形式であっても，それが EXI を利用したものであれば，設計ファイルの更新にてある程度柔軟に適応可能であると考えられる．

今後は，処理系の更なる軽量化を推し進めると共に、Schema-Infromed Grammar の non-strict モードおよび Built-in Grammar の対応などの機能拡充を目指す予定である．

参考文献

1) ECHONET コンソーシアム - http://www.echonet.gr.jp/
2) ZingBee ZigBee Smart Energy - http://www.zigbee.org/Home.aspx
3) 佐藤弓子，土井裕介，寺本圭一：通信規格の異なる家電機器と家電コントローラを相互接続可能にするアダプタの実装方法，情報処理学会第 1 回 CDS 研究会研究報告(2011).
4) Extensible Markup Language (XML) 1.1 - http://www.w3.org/TR/xml11/
5) Efficient XML Interchange (EXI) Format 1.0 - http://www.w3.org/TR/exi/
6) 土井裕介，佐藤弓子，寺本圭一：XML-Less EXI: EXI 利用プロトコルの家電・組込機器への実装手段の検討，情報処理学会第 31 回 UBI 研究会研究報告(2011).
7) Y. DOI, Y. SATO, and K. TERAMOTO, "EIGEN: XML-Less EXI with Code Generation for Smart Energy Home Appliances", IEEE Transaction of Consumer Electronics (2012).
8) Expat XML Parser - http://sourceforge.net/projects/expat/
9) ZigBee Smart Energy 2.0 DRAFT 0.7 Public Application Profile - http://zigbee.org/Standards/ZigBeeSmartEnergy/ZigBeeSmartEnergy20PublicApplicationProfile.aspx
10) Constrained Application Protocol (CoAP)- http://tools.ietf.org/html/draft-ietf-core-coap-03