CA-EXI：運用コンテキストとEXI符号化を用いたIEEE1888メッセージ圧縮手法

落合　秀也[†a)]　　土井　裕介[††]　　江崎　浩[†]

CA-EXI: Context-Aware Efficient XML Interchange for IEEE1888 Message Compression

Hideya OCHIAI[†a)], Yusuke DOI[††], and Hiroshi ESAKI[†]

あらまし　機械間通信での3G回線利用は増えてきているが，XML通信をベースとする機械間システムではこれまで回線上の通信量が課題となることがあった．本研究では，その課題を解決するCA-EXIと呼ぶXMLメッセージの圧縮方式を提案し，IEEE1888通信システムにおける，その圧縮効率の評価実験を行った．その結果，CA-EXIは，幅広いフットプリントの領域(小さなIEEE1888XMLメッセージから大きなIEEE1888XMLメッセージまで)において，従来のEXI方式よりも強力かつGZIP方式に劣らないバランスの良いメッセージ圧縮率を達成した．特にメッセージサイズの小さい領域では，GZIP方式よりも強力な圧縮率を達成した．

キーワード　機械間通信，XML，EXI，圧縮，IEEE1888

1. まえがき

機械間通信 (M2M: Machine-to-Machine)，すなわち，センサ・アクチュエータ機器とクラウドサーバ間の通信に，第3世代移動体通信システム (通称3G回線) のようなキャリアネットワークを用いるケースが増加してきている．センサ・アクチュエータを遠隔地に設置する際に，同時に3G回線を引くことによって，既設ネットワークへの接続における諸問題（ファイヤウォール解放，MACアドレス届出，IPアドレス設定，物理的配線など）を一気に解決し，導入を容易にできるようになる．しかし，3G回線設備はネットワークキャリアによって維持されなければならず，回線費用が発生するという難点がある．回線の維持費用は通信量に左右されるところが多く，その実態は回線料金の契約形態にも反映される．

導入に関してこのように魅力的な3G回線であるが，M2Mの現実的な普及を考えた場合には，回線設備への負荷 (=通信の量) をどれだけ小さくできるかが，重要となってくる．一方で，M2Mシステム全体の開発効率性なども考慮しておく必要があり，単に通信量を減らすことだけを考えれば良いわけでもない．

そこで本研究では，開発効率性などの観点で設計されたHTTPとXMLによるM2Mシステムを想定した上で，狭帯域回線を通るメッセージを圧縮するContext-Aware Efficient XML Interchange (CA-EXI) 方式を提案する．具体的には，HTTPとXMLを用いるIEEE1888通信規格[1]を取り上げ，クライアントサーバ間の通信の中で，狭帯域回線部分の両端で圧縮 (Encode) と展開 (Decode) を行わせる運用形態に焦点を当てる．そして，その圧縮方式の一つとして，CA-EXIを提案する．

単に通信量を削減するのであれば，UDPデータグラムに必要な情報やデータをバイナリエンコードして送信すれば良い．きっと，フットプリントを小さくできる効率的なメッセージ・フォーマットがあるだろう．しかし，これだけではシステム全体のソフトウェア開発効率，メンテナンス効率などに課題を残すことになる．事実，M2Mシステムでの通信にはHTTPとXMLが使われるケースが多い (oBIX [2], BACnet Web Services [3] など) が，これはシステム開発や維

† 東京大学，東京都
The University of Tokyo, Tokyo, 113–8656 Japan
†† （株）東芝，川崎市
Toshiba Corporation, Kawasaki-shi, 212–8531 Japan
a) E-mail: ochiai@vdec.u-tokyo.ac.jp

持の効率性・汎用性を求めた結果なのである．

本研究で提案する CA-EXI 方式では，XML の圧縮符号化は，高効率 XML 交換方式 (EXI: Efficient XML Interchange) [4] をベースにしている．本研究では，EXI の本来の仕組みに加え，M2M システムならではの特性である下記 2 点に着目し，更なる圧縮率向上 (評価実験の結果は，EXI 方式に比べ，平均で 1/5.6 に，最大で 1/18.5 に圧縮した) を狙う．

- エンドノード近くのセンサ・アクチュエータやデータタイプの構成はほとんど変化しない（=同じ ID 表現や値表現が長期間に渡る複数メッセージに何度も登場する）
- エンドノード近くのセンサ・アクチュエータやデータタイプの構成は有限である (=登場する ID 表現や値表現の空間が有限である)

本研究では，このような特性に着目し，運用場面 (運用現場) という “コンテキスト” を作り，そのコンテキストを利用してより高度な圧縮を実現する．

本論文の構成は，次のとおりである．**2.** にて関連研究との関係を述べる．**3.** にて，本研究が想定するシステム構成を明確にし，**4.** 及び **5.** にて本研究で提案する CA-EXI 手法を説明する．**6.** で行った実験とその評価結果を述べる．**7.** にて一般的な M2M システムへの応用について考察をし，**8.** にて本論文を締める．

2. 関 連 研 究

インターネット上のシステム開発で，Web サービス (HTTP 及び XML) が主流になったことで，開発効率性やインターオペラビリティ向上の理由から，M2M システムの分野でも Web 技術を利用することが重要になっている．しかし，Web サービスは，通信量や処理の重さに課題があるとされており，それらを軽減するために，数多くの研究が行われている ([5]〜[8])．

XML 文書の圧縮に関しては，EXI が登場する以前は，Xmill [9]，xmlppm [10], [11]，Fast Infoset，WBXML などが提案され，Christian Werner らの研究による分析 [12] では Xmill や WBXML が良いとされた．ただし，この検証には EXI は盛り込まれておらず，W3C が行った評価 [13] では，EXI が最も良いという結果となっている(注1)．その後，W3C にて EXI の仕様化が進み，現在は EXI が圧縮方式の主流となりつつある．

なお，アプリケーション・オブジェクトのシリアル化（符号化）の観点から考えた場合，XML は単に一つのシリアル化方式であり，EXI も別のシリアル化方式であると言える．Doi の研究 [14] ではこの特性を利用して，組込みデバイスが直接 EXI でやり取りする方式を提案している．

これまで，このように XML の圧縮手法が研究されてきたが，ほとんどの研究は，単一の XML 文書を圧縮することのみを考えていて，何度も登場する似たような XML 文書に対する圧縮は十分に検討されてはいない．しかし，実際には，先に述べた M2M の特性 (エンドノード近くのセンサ・アクチュエータやデータタイプの構成はほとんど変化しない，また，その取りうる構成は有限である) があるのであり，これを想定すれば，更に圧縮が可能なはずである．我々の研究は，この部分に着目し，検証を行ったものである．

3. システム構成

本研究では，図 1 に示すように，IEEE1888 のクライアントノードとサーバノードの間に狭帯域回線がある場合を想定し，その両端でメッセージを圧縮・展開するものとする．IEEE1888 では，遠隔手続き呼出し (RPC: Remote Procedure Call) 方式で，クライアントからリクエストメッセージをサーバに対して送り付け，サーバは処理結果をレスポンスとして回答する．実際の運用では，例えば，クライアントノード側に，電力メータや環境センサがあり，その計測値をクラウド側のサーバに対して送り付け，その成否結果がレスポンスとして返ってくるという形になる (あくまで一

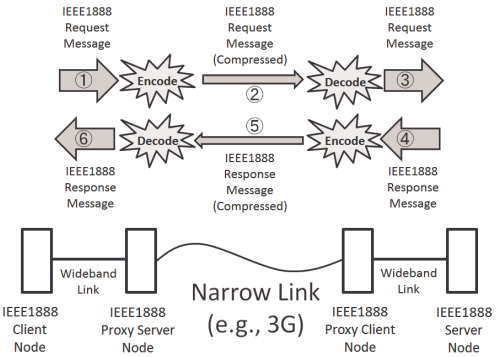

図 1　システム構成と通信の全体像
Fig. 1　Overview of a target system and communication flow.

(注1)：Efficient XML emerges as the best performer for nearly all test documents in all application classes とある．

例である)．本研究ではクライアントサーバ間にプロキシノードを用意し，このプロキシノード間の通信を圧縮することを考える．つまり，TCP のレベルでは，クライアントノードは，まず，プロキシサーバノードに対して，IEEE1888 のリクエストメッセージを送る．すると，本研究で提案する CA-EXI 手法によりメッセージが圧縮され，対向にあるプロキシクライアントノードに送信される．プロキシクライアントノードでメッセージを復元し，IEEE1888 サーバノードに送信する．レスポンスメッセージは，プロキシクライアントで結果が圧縮され，狭帯域回線を伝わった後，プロキシサーバで展開され，IEEE1888 クライアントノードに届けられる．

ここで，圧縮手法としては，古典的には GZIP [15] があり，近年は EXI が使われようとしている．本研究で提案する CA-EXI 手法は，EXI 方式をベースとしつつも，2 段階圧縮を行うことによって，EXI 方式よりも高い圧縮率を実現する．

4. Context-Aware Efficient XML Interchange (CA-EXI)

本提案手法では，図 2 に示すように，二段階のフェーズによって，バイナリコードに圧縮したメッセージを生成する．IEEE1888 の XML メッセージを，途中，圧縮効率のよい中間 XML コードに変換し，その中間 XML コードを EXI によりバイナリ符号化 (圧縮) する．バイナリ化されたメッセージが，デコーダまで配送され，逆の演算処理を経て，元の IEEE1888 の XML メッセージが復元される．

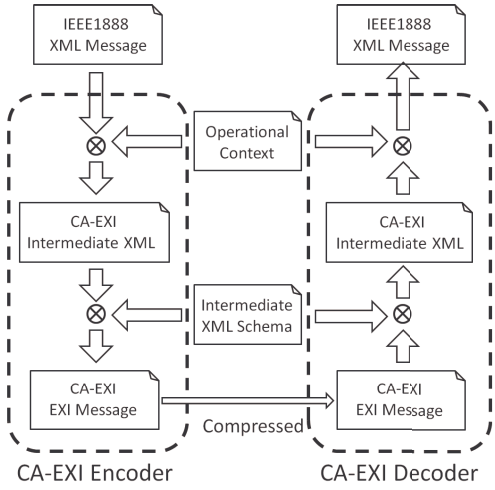

図 2　CA-EXI 圧縮・展開のフロー
Fig. 2　CA-EXI compression and infration.

IEEE1888 の XML メッセージから中間 XML コードを生成する際や，中間 XML コードから元の XML メッセージに復元する際に，“運用コンテキスト” と呼ぶ情報を利用する．この運用コンテキストは，その狭帯域回線の両端の運用形態を反映した情報であり，自動抽出させることが可能である．その抽出方法は次章にて解説する．

中間 XML コードのスキーマは，IEEE1888 で使われる XML のスキーマを表 1 のように対応付けしたものである．IEEE1888 ではグローバル性や汎用性を高めるために anyURI や string という形で抽象定義されていた部分を，運用コンテキストの配下で使われる中間 XML では，よりプリミティブな型に対応付けるようにしている．

IEEE1888 の XML を，中間 XML コードに対応付けする際に利用する運用コンテキストは，図 3 に示す構造となっている．

図 3 の運用コンテキストの場合，図 4 の IEEE1888 の XML メッセージは，図 5 の中間 XML コードに変換される．

図 4 のオリジナル XML は，URI 型や文字列型を多く含んでいるため，EXI 符号化による圧縮効率はあまり期待できないが，図 5 の中間 XML コードでは，

表 1　中間 XML のスキーマ構造
Table 1　Schema for intermediate XML.

IEEE1888 XML Schema			Intermediate XML Schema		
Element	Attr	Type	Element	Attr	Type
key	id	anyURI	key	id	uint
key	eq	string	key	eq	datetime
key	neq	string	key	neq	datetime
key	lt	string	key	lt	datetime
key	gt	string	key	gt	datetime
key	lteq	string	key	lteq	datetime
key	gteq	string	key	gteq	datetime
pointSet	id	anyURI	pointSet	id	uint
point	id	anyURI	point	id	uint
value	content	string	ivalue	content	integer
			rvalue	content	double
			evalue	content	uint
			svalue	content	string

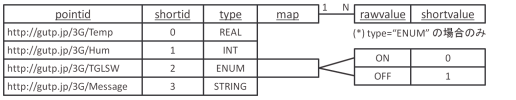

図 3　運用コンテキストの例
Fig. 3　Operational context example.

```
<transport>
 <body>
  <point id="http://gutp.jp/3G/Temp">
   <value time="2013-01-30T00:00:00+09:00">25.5</value>
  </point>
  <point id="http://gutp.jp/3G/Hum">
   <value time="2013-01-30T00:00:00+09:00">58</value>
  </point>
  <point id="http://gutp.jp/3G/TGLSW">
   <value time="2013-01-30T00:00:00+09:00">OFF</value>
  </point>
  <point id="http://gutp.jp/3G/Message">
   <value time="2013-01-30T00:00:00+09:00">Hello</value>
  </point>
 </body>
</transport>
```

図 4　IEEE1888 XML メッセージ
Fig. 4　IEEE1888 XML message.

```
<transport>
 <body>
  <point id="0">
   <rvalue time="2013-01-30T00:00:00+09:00">25.5</rvalue>
  </point>
  <point id="1">
   <ivalue time="2013-01-30T00:00:00+09:00">58</ivalue>
  </point>
  <point id="2">
   <evalue time="2013-01-30T00:00:00+09:00">1</evalue>
  </point>
  <point id="3">
   <svalue time="2013-01-30T00:00:00+09:00">Hello</svalue>
  </point>
 </body>
</transport>
```

図 5　中間 XML コード
Fig. 5　Intermediate XML code.

これらが数値に対応付けされていたり，型明示されていたりするため，圧縮効果を期待できる．中間 XML コードにも文字列型が用意されているが，**6.** の評価実験で明らかなように実際のセンサネットワークでは文字列型でしか扱えない値表現は多くはない．

5. 運用コンテキストの抽出と共有

5.1 抽出アルゴリズム

CA-EXI 圧縮に必要な運用コンテキストは，実際にやり取りされる複数の IEEE1888 XML メッセージから動的に抽出することができる．基本的には，CA-EXI のエンコーダに次々と到着する XML メッセージに，(1) 新規の IEEE1888 ポイント ID があれば，運用コンテキストに新規の ID レコードを追加し，(2) 値の中身は常にチェックし，適切なデータ型を選択すれば良い．その抽出アルゴリズムを，本研究では以下のように定義する．

現在の運用コンテキストを C と置き，C がもっているコンテキストの 1 レコードを c とする ($c \in C$ と表現する)．そして，$c.pointid$，$c.shortid$，$c.type$，$c.map$ をそれぞれ IEEE1888 のポイント ID，中間 XML コードでの対応番号，値の型 (NULL, INTEGER, REAL, ENUM, STRING のいずれか)，値表現の数値対応集合 (ENUM 対応表) とする．値表現の数値対応集合 $c.map$ の一要素を $e(\in c.map)$ とする場合，$e.rawvalue$ で IEEE1888 XML での値表現，$e.shortvalue$ で中間コードでの値表現をするものとする．

いま，到着する IEEE1888XML メッセージの集合を M とし，その M の一要素を m とする ($m \in M$ と表現する)．このとき，$m.points$ をメッセージ m に記載されている IEEE1888 でのポイントまたはポイントセットの集合とする (point 要素または pointSet 要素に対応する)．各ポイント (セット) を $p(\in m.points)$ と表現したとき，$p.id$ でポイント ID を，$p.values$ で p がもつ値 (value 要素) の集合をもつものとする．実際の値は，$v \in p.values$ と置いた場合に，$v.content$ として取り出せるものとする．

上記のように定義するとき，IEEE1888XML メッセージの集合 M から，運用コンテキスト C を次の方法で抽出させる．なお，C の初期値は Φ(空集合) とする．

– 運用コンテキスト抽出アルゴリズム (ここから)–

Forall $m \in M$
 Forall $p \in m.points$
  If $\neg$ exists $c \in C(c.pointid = p.id)$
   c :=new()
   $c.pointid := p.id$
   $c.shortid$:=count(C)
   $c.type$:=NULL
   $C := C \cup \{c\}$
  ElseIf
   c :=arg$\{c.pointid = p.id\}$
  EndIf
  Forall $v \in p.values$
   If $c.type$ =NULL
    $c.type$:=type($v.content$)
    $c.map := \Phi$
   ElseIf $c.type$ =INTEGER
    If type($v.content$)=REAL
     $c.type$:=REAL

```
      c.map := Φ
     ElseIf type(v.content)=ENUM
      c.type :=ENUM
      c.map := Φ
     EndIf
    ElseIf c.type =REAL
     If type(v.content)=ENUM
      c.type :=ENUM
      c.map := Φ
     EndIf
    ElseIf c.type =ENUM
     If count(c.map)<MAX_ENUM
      If ¬ exists e∈c.map(e.rawvalue = v.content)
       e :=new()
       e.rawvalue := v.content
       e.shortvalue :=count(c.map)
       c.map := c.map ∪ {e}
      EndIf
     ElseIf
      c.type :=STRING
      c.map := Φ
     EndIf
    EndIf
   EndForall
  EndForall
EndForall
```

– 運用コンテキスト抽出アルゴリズム (ここまで)–

ここで，各定数・オペレータ・関数は，次の意味である．

- MAX_ENUM ENUM 型の最大要素数設定値
- = 両辺の値が等しいかどうかの判定
- < 左辺の値が右辺の値より小かどうかの判定
- := 右辺を左辺に代入する
- new 新しい要素の生成
- count 要素数を求める
- type この値のデータ型を求める

このアルゴリズムによって，通信路上を流れるIEEE1888XML メッセージから，運用コンテキストを自律的に抽出することが可能になる．

定数である MAX_ENUM は，適用する現場の設備の組み合わせから ENUM 要素数 (=*c.map* 要素数) を推測し，それを超える値を設定する．それよりも小さい値を設定すると，本来 ENUM 型として判定される値が，STRING 型と判定されてしまい，通信時には毎回文字列を送ってしまうことになる．一方で，MAX_ENUM の値を限りなく大きくすることもできるが，本来 STRING 型と判定される ID レコードが常に ENUM 型となってしまい兼ねない．その場合，新しい文字列が登場するたびに，*c.map* 要素を追加することになり，単に運用コンテキストのフットプリント (*c.map* 要素数も考慮に入れたもの) が，大きくなってしまう．しかし，メモリ容量に余裕があれば，多少 MAX_ENUM の設定値が大きくても，弊害はなく，単に，多くの文字列パターンが *c.map* 表により，数値 *e.shortvalue* で表現されることになる．

5.2 共有アルゴリズム

運用コンテキストは，(1) レコードの追加，(2) 型の変更，(3) ENUM 型の場合は値要素の追加，のタイミングで変化が発生する．これらのタイミングで，エンコーダ側からデコーダ側に，変化の種別 (レコード追加，型変更，ENUM 値要素の追加) と，それに付随するパラメータを渡すことで運用コンテキストの共有を行うことができる．

- “レコード追加” の場合は，*c.shortid*，*c.pointid*，*c.type* (*c.type* = *ENUM* の場合は *e.shortvalue* と *e.rawvalue* のペアも) をパラメータとして渡す．
- “型変更” の場合は，*c.shortid*，*c.type* (*c.type* = *ENUM* の場合は *e.shortvalue* と *e.rawvalue* のペアも) をパラメータとして渡す．
- “ENUM 値要素追加” の場合は，*e.shortvalue*，*e.rawvalue* のペアを，パラメータとして渡す．

上記の共有アルゴリズムは，差分情報を伝達するだけなので，実装時には，エンコーダ・デコーダ間での運用コンテキストの不一致検出，運用コンテキストの全転送などの仕組みも併せて必要となるかもしれない．しかし，本研究では，上記のものを共有アルゴリズムとして考えることにする．

6. 評　　価

本研究では，実際の通信路を流れた IEEE1888 通信メッセージを用い，(1) 運用コンテキストの抽出に関する評価，(2) 圧縮率に関する評価，(3) 通信の総量に関する評価を行った．以下が，その実験方法と結果である．

6.1 データセットと実験方法

東大グリーン ICT プロジェクト及び Live E!プロジェクトで運用している 6 台の IEEE1888 サーバが

やり取りする IEEE1888 メッセージを，それぞれの IEEE1888 サーバの根元でキャプチャし，これを実験のデータセットとした．データの収集期間，生データのフットプリント，ポイント ID 数を表 2 に示す．

それぞれのデータセットには，IEEE1888 サーバへのリクエストと IEEE1888 サーバからのレスポンスの全てが含まれている．

抽出に関する評価では，この通信メッセージを順に処理して運用コンテキストの抽出をサーバごとに行わせ，その際に発生する運用コンテキストの共有に必要な通信量の推移，及び，コンテキスト情報の大きさの推移を評価した．コンテキスト情報の大きさは，ここでは ID 数と ENUM 対応表の数の和とした．また，抽出結果の ID 数，データタイプの内訳についても評価した．なお，抽出アルゴリズムでの定数である MAX_ENUM は，100 に設定した(注2)．

通信量を調べる際には，**5.2** におけるアルゴリズムに付随する各情報量を，

- **変化の種別: 1 byte**
- **c.shortid: 2 byte**
- **c.pointid: c.pointid 長+2 byte**
- **c.type: 1 byte**
- **e.shortvalue: 1 byte**
- **e.rawvalue: e.rawvalue 長+2 byte**

と置いて，発生した情報量の総和を計算した．この実験では，実装に依存する各種オーバヘッド (IP ヘッダの大きさや，TCP 通信のやり取り) は除外した．

圧縮率に関する評価では，IEEE1888 メッセージを，そのまま (1) EXI 方式で圧縮する場合，(2) GZIP 方式で圧縮する場合，(3) 提案方式の CA-EXI 方式で圧縮する場合の 3 通りの方法を比較した．比較においては，フットプリントが小の領域 (2000byte 未満) での圧縮効率，フットプリントが中の領域 (2000byte〜10000byte) での圧縮効率，フットプリントが大の領域 (10000byte 以上) での圧縮効率をグラフにプロットした．1 メッセージあたりのフットプリントの分布 (確率密度関数，累積密度関数) も分析し，どの領域を重点的に圧縮すると効果的かについても評価した．ここで，確率密度関数 (PDF: Probability Distribution Function) の計算は，各サーバでやり取りされた通信メッセージ量の密度関数を，100byte 単位で区切って求め，サーバごとの平均値を取った．累積密度関数 (CDF: Cumulative Distribution Function) の計算は，求められた PDF を積分することで求めた．本評価においては，**5.1** のアルゴリズムに基づき，到着する XML メッセージごとに動的に運用コンテキストを抽出しながら，その XML メッセージを圧縮し，その圧縮量を評価している．運用コンテキストの共有に必要な通信量は，この分析では，切り出されている．

通信の総合量に関する評価では，全てのデータセットに対し，総合的に考えてどの程度圧縮できたかを評価した．すなわち，まず，CA-EXI について，運用コンテキストの共有のための通信量と圧縮データ通信量の和を考え，それらの関係を分析した．その上で，CA-EXI，EXI，GZIP での全通信量を比較評価した．

表 2　評価実験で使用したデータセット
Table 2　Data set for evaluation.

Server	Duration	Traffic	Number of IDs
cloud	81.5hour	446 MByte	122
cloud2	81.5hour	516 MByte	110
dev	81.5hour	884 MByte	2353
live-e	81.5hour	570 MByte	1996
sandbox	16.5hour	17.6 GByte	887
storage	81.5hour	22.6 GByte	3634

(注2)：実験で用いたデータには，エアコンの動作モード，スマートタップの設定モード，特殊設備の状態値など，各種実験で用いている様々なデータ表現が含まれていた．その内容を考えつつ，多少大きい分には弊害がないことを踏まえ，安全を見て 100 に設定した．

6.2　運用コンテキストの抽出の推移

図 6 及び図 7 に，運用コンテキストの共有に使った通信量 (総和) の推移，及び，コンテキスト情報の大きさの推移の結果を示す．横軸は，通信開始時から到着した IEEE1888 メッセージの数を対数スケールで表示したものである．コンテキスト情報の大きさは，エンコーダ・デコーダで保有すべきメモリ容量の大きさに関係する．

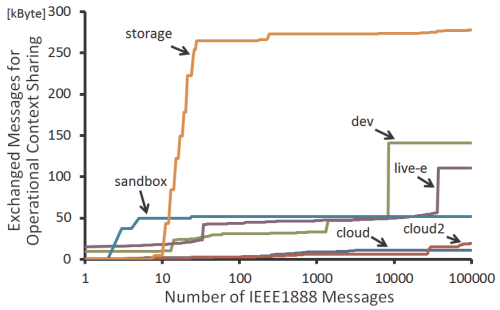

図 6　運用コンテキストの共有に必要な通信量 (情報量) の推移
Fig. 6　Transition of the total messages for operational context sharing.

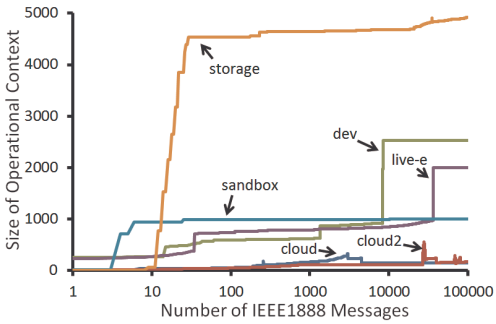

図 7　コンテキスト情報の大きさ (ID レコード数と *c.map* 要素数の総和) の推移
Fig. 7　Transition of context size.

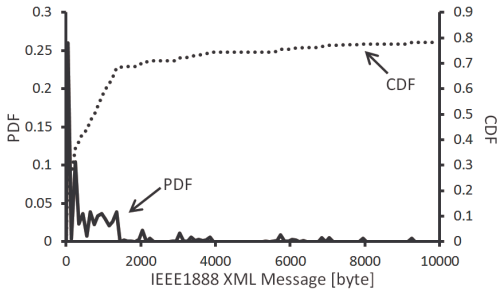

図 8　IEEE1888 メッセージ (1 個) あたりのフットプリントの分布 (確率密度関数 PDF，累積密度関数 CDF)
Fig. 8　Distribution of data size for one IEEE1888 message.

表 3　データタイプの内訳
Table 3　Data type distribution.

Server	NULL	INTEGER	REAL	ENUM	STRING
cloud	2	41	73	2	4
cloud2	2	12	64	16	16
dev	451	640	1086	176	0
live-e	1133	71	792	0	0
sandbox	439	186	135	117	0
storage	401	872	1498	929	1

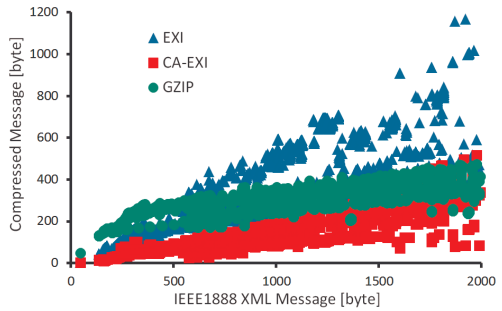

図 9　フットプリントが小 (0〜2000byte) の領域での圧縮量
Fig. 9　Compression rate of IEEE1888 messages (0 - 2000 byte).

この図から，主に通信開始時に運用コンテキストの共有が行われ，その後は，時折，変化に伴う共有のための通信が発生していることがわかる．コンテキスト情報の大きさも，通信開始時には急速に変化するが，その後は，ほぼ変化せずに推移していることがわかる (横軸は対数表示であり，通信開始初期が特に引き延ばされていることに注意)．なお，コンテキスト情報の大きさは，時折減少しているが，これは c.type=ENUM から c.type=STRING に移行したタイミング (その *c.map* 対応表がクリアされたため) で発生している．

この結果より，通信開始時には頻繁に運用コンテキストの抽出・共有処理を行うことになるが，その後は変化が発生しにくい傾向にあることがわかる．つまり，一度コンテキストを抽出・共有してしまえば，そのコンテキストを利用して CA-EXI による IEEE1888 の XML メッセージ圧縮に専念することが可能になる，と言える．

6.3　運用コンテキストの抽出結果

表 3 に，抽出によって分類された各データタイプの内訳を示す．数値は，その型と判定された IEEE1888 ポイントの個数を表す．NULL になっているものは，値が存在しなかった (あるいはポイントセットである) ため，値の型判定ができなかったことを意味する．この表からわかるように，NULL を除けば，REAL 型，INTEGER 型，ENUM 型が大半を占めている．このように，実際の運用では，STRING 型は，ほとんど見受けられないことが分かった．

6.4　圧縮データのフットプリント

図 8 に，IEEE1888 の XML メッセージ 1 個が，どのくらいのフットプリントをもっているかを表す確率密度関数 PDF 及び累積密度関数 CDF を示す．この結果から，この実験で用いたデータセットにおいては，

- 69.0%のメッセージが小領域
- 9.1%のメッセージが中領域
- 21.9%のメッセージが大領域

に属することがわかる．

図 9，10，11 にそれぞれ IEEE1888XML メッセージのフットプリントが小さい場合，中くらいの場合，大きい場合の EXI，CA-EXI，GZIP 方式での圧縮されたメッセージのフットプリントを示す．

図 9 の結果より，フットプリントが小の領域では，提案手法 (CA-EXI) が EXI と GZIP 方式に比べて優

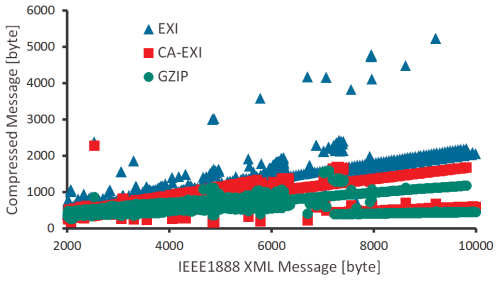

図 10　フットプリントが中 (2000～10000byte) の領域での圧縮量

Fig. 10　Compression rate of IEEE1888 messages (2000 - 10000 byte).

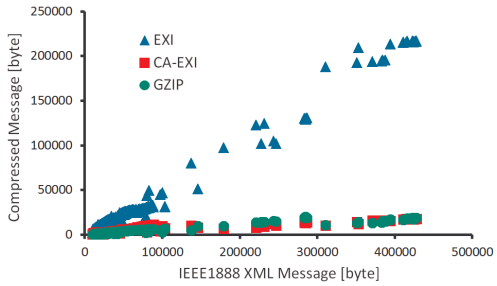

図 11　フットプリントが大 (10000byte～) の領域での圧縮量

Fig. 11　Compression rate of IEEE1888 messages (10000byte -).

れていることがわかる．CA-EXI 方式は，平均すると，EXI 方式と比べて 1/3.6 に圧縮し，GZIP 方式と比べて 1/2.8 に圧縮できていた．

図 10 の結果より，フットプリントが中の領域では GZIP の方が，圧縮効率が良い場合があることがわかる．しかし，EXI 単体の圧縮に比べれば，提案手法 (CA-EXI) の方が圧縮効率は良い．

図 11 の結果より，フットプリントが大の領域では，EXI に比べて，提案手法 (CA-EXI) の方がはるかに圧縮できていることがわかる．その圧縮量は GZIP とさほど変わらない．

全体的には，CA-EXI 方式は，EXI 方式と比べて平均 1/5.6，最大で 1/18.5 に圧縮していた．また，GZIP 方式と比べて，平均で 1/1.16 に圧縮していた．

以上の結果を総合すると，CA-EXI は，

- 幅広いフットプリントの領域 (小さな IEEE1888XML メッセージから大きな IEEE1888XML メッセージまで) において，十分な圧縮を実現した．
- 特に出現頻度の高い「フットプリントが小の領域」では，EXI や GZIP 方式よりも強力な圧縮率を達成した．

と評価することができる．

表 4　CA-EXI での圧縮データ総量と運用コンテキスト通信総量の比較

Table 4　Size comparison of compressed data and messages for operational context sharing in CA-EXI.

Server	Total	Data	Context	Ratio
cloud	62.3MByte	62.3MByte	13.1KByte	0.00021
cloud2	70.7MByte	70.7MByte	26.7KByte	0.00038
dev	61.5MByte	61.4MByte	141KByte	0.0023
live-e	56.6MByte	56.6MByte	51.8KByte	0.00091
sandbox	823MByte	823MByte	111KByte	0.00013
storage	1.03GByte	1.03GByte	278KByte	0.00027

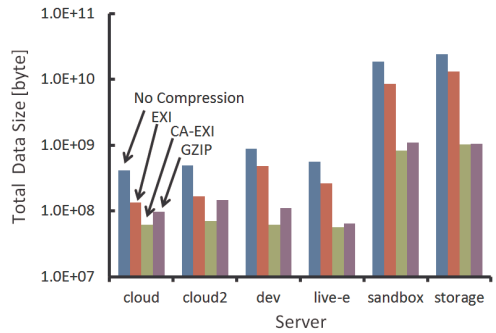

図 12　全データセットでの通信総量の比較

Fig. 12　Compressed data footprint for each data set.

6.5　通信総量の評価

まず，CA-EXI での圧縮データの通信量と，運用コンテキスト情報の通信量を総合した際の，これらの総量及び比率を表 4 に示す．この結果から，運用コンテキストの通信総量は，圧縮されたデータ量の総量と比較してほぼ無視できるほど小さいと言うことができる．また，ID 数及び ENUM 要素数が多いケースほど運用コンテキストの通信総量は大きくなるが，対象とするデータの量が多ければ多いほど，全体に占める割合は小さくなる傾向にある．

図 12 は全ての通信がそれぞれの手法でどの程度まで圧縮できたかを示している．この結果から，CA-EXI は (運用コンテキストの共有のための通信量も含めて) 全てのケースにおいて，EXI や GZIP よりも優れる結果となったことが読み取れる．

7.　考　　察

本研究では，M2M システムでの通信メッセージ量の削減を実現する際に，IEEE1888 通信にフォーカスを当てて，その圧縮手法として CA-EXI を提案し，そ

の効果の有効性を示した．この CA-EXI は，XML スキーマにおいて，anyURI 形式や string 形式となっている部分 (すなわち EXI 符号化であまり小さくならない部分) を “運用コンテキスト” の考え方で，プリミティブな int 形式，double 形式などに対応付けることが基本原理となっている．つまり，この CA-EXI 方式自体は，IEEE1888 以外の XML 形式 (oBIX [2], BACnet Web Service [3]，ZigBee SEP 2.0) などにも応用可能であると考えられる．

今回の実験では，EXI よりも GZIP の方が良い圧縮率を示すことがあったが，これは今回用いた IEEE1888 の XML メッセージの特徴を反映した結果と言える．EXI では，文字列型は完全に一致する場合のみ圧縮し，同一 XML ドキュメントに同じ文字列が何度も登場するときに効果的である．一方，IEEE1888 には一つのドキュメントに複数の ID が登場するが，個々の ID は末尾がそれぞれ異なるため，EXI では圧縮できない．したがって，IEEE1888 の XML メッセージをそのまま EXI 化しても圧縮率は高くない．一方の GZIP 方式では，文字列としては相互に似ている ID が何度も登場するため，IEEE1888 のドキュメントを強力に圧縮できる．この IEEE1888 の XML ドキュメントの性質が，EXI と GZIP の圧縮率の差を生み出したと考えられる．

1. で述べたように，IEEE1888 の XML ドキュメントは，主に，センサ・アクチュエータの情報をやり取りするため，通常の運用では，登場する ID や値の表現が限られてくる．一つのドキュメントでは再登場しない ID が，一つの運用コンテキストでは再登場する．そのため，一度，運用コンテキストを抽出してしまえば，それ以降，その変化はほとんど起きない (=無視できるレベルになってくる) と考えてよい．今回の実験は，このような特徴を裏付けつつ，そのような場合に，CA-EXI が強力に機能することを明らかにした．

なお，本研究では，End-to-End の通信は，IEEE1888 の XML メッセージを交換することを前提とし，狭帯域回線を通るときだけメッセージを圧縮するというアプローチを取った．しかし，エンコーダ及びデコーダを，エンドデバイスやサーバ側にもたせることも当然，応用としては可能である．またコンテキストの抽出処理は，動的に行われるものを対象としていたが，静的にコンテキスト情報をサーバ側にもたせておき，エンドデバイスは中間 XML の EXI をそのまま理解させるようにシステムを設計することも応用としては可能だろう．

8. むすび

我々は，XML をベースとする M2M システムが狭帯域回線 (e.g., 3G 回線) を利用する際に，その通信フットプリントをどのように圧縮したらよいかを研究し，具体的手法として CA-EXI 方式を提案した．CA-EXI は，運用現場の構成を反映したコンテキストを利用して，効率的な通信メッセージの圧縮を行う．評価実験で分かったように，コンテキストの抽出処理は頻繁に起こるものではなく，コンテキスト抽出及びコンテキスト情報のエンコーダ・デコーダ間での共有の負担は大きくはない．本研究のデータセットを用いた場合，高効率 XML 交換方式である EXI 手法での圧縮量に比べて，更に平均 1/5.6，最大 1/18.5 に圧縮した．また，70%近い通信メッセージが取りうるフットプリント 2000 バイト未満のメッセージに関しては，平均すると，GZIP 方式と比べて 1/2.8 に，EXI 方式と比べて 1/3.6 に圧縮できていた．

文　献

[1] 落合秀也，IEEE1888 プロトコル教科書，インプレスジャパン，2012.

[2] “Open building information exchange.” http://www.obix.org/

[3] “Bacnet - A data communication protocol for building automation and control networks.” http://www.bacnet.org/

[4] “Efficient XML interchange (EXI) format 1.0.” http://www.w3.org/TR/exi/

[5] N.B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, “Tiny web services: Design and implementation of interoperable and evolvable sensor networks,” ACM SenSys, 2008.

[6] D. Schall, M. Aiello, and S. Dustdar, “Web services on embedded systems - A performance study,” IEEE PERCOM, 2010.

[7] C. Werner, C.B.Y. Brandt, and S. Fischer, “Compressing soap messages by using pushdown automata,” IEEE ICWS, 2006.

[8] R. van Engelen, “Code generation techniques for developing light-weight XML web services for embedded devices,” ACM SAC, 2004.

[9] H. Liefke and D. Suciu, “XMill: An efficient compressor for XML data,” ACM SIGMOD, 2000.

[10] J. Cheney, “Compressing XML with multiplexed hierarchical PPM models,” IEEE DCC, 2001.

[11] J. Kangasharju, S. Tarkoma, and T. Lindholm, “Xebu: A binary format with schema-based optimizations for XML data,” 6th International Conference on Web Information Systems Engineering, vol-

ume 3806 of Lecture Notes in Computer Science, pp.528–535, 2005.
[12] C.J. Augeri, D.A. Bulutoglu, B.E. Mullins, R.O. Baldwin, and L.C. Baird, “An analysis of XML compression efficiency,” ACM ExpCS, 2007.
[13] “Efficient XML interchange measurements note.” http://www.w3.org/TR/exi-measurements/
[14] Y. Doi, Y. Sato, M. Ishiyama, Y. Ohba, and K. Teramoto, “XML-less EXI with code generation for integration of embedded devices in web based systems,” IEEE Internet of Things, 2012.
[15] J. loup Gailly and M. Adler, http://www.gzip.org/

（平成 25 年 1 月 31 日受付，5 月 28 日再受付）

落合　秀也　（正員）

平 18 東大・工・電子情報卒．平 20 同大大学院情報理工学系研究科修士課程了．平 23 同大学院同研究科博士課程了．同年同大学大規模集積システム設計教育研究センター・助教，現在に至る．博士 (情報理工学，東京大学)．設備ネットワーク，広域センサネットワーク，遅延耐性ネットワーク研究の他，IEEE1888，ASHRAE の設備ネットワーク標準化活動に従事．

土井　裕介　（正員）

平 12 慶應義塾大学大学院政策・メディア研究科修士課程了．平 23 年 3 月東京大学大学院情報理工学系研究科博士後期課程単位取得退学．同年 9 月，博士 (情報理工学，東京大学) 取得．平 12 より，(株) 東芝研究開発センターで分散システム，ホームネットワーク，電子タグ，組み込み向け XML 処理系等の研究開発，Smart Energy Profile 2.0 の標準化等に従事．

江崎　　浩　（正員）

昭 62 九州大学大学院・工・電子修士課程了．同年 (株) 東芝入社．平 2 米国ニュージャージ州ベルコア社．平 6 コロンビア大学・客員研究員．平 10 東京大学大型計算機センター・助教授．平 13 同大学大学院・情報理工学系研究科・助教授．平 17 同大学大学院・同研究科・教授，現在に至る．博士 (工学，東京大学)．MPLS-JAPAN 代表，IPv6 普及・高度化推進協議会専務理事，WIDE プロジェクト代表，JPNIC 副理事長．