モバイルコンピューティングと
ワイヤレス通信
高度交通システム
（２００１．９．７）
18－29
6－29

LDAP と XML を用いた ITS アドレス管理手法

杉山　敬三　　蕨野　貴之　　篠永　英之

(株)KDDI 研究所

本稿では、ITS における車両の位置情報や通信プロファイルなどのアドレス情報を管理する手法として、LDAP(Lightweight Directory Access Protocol)と XML(eXtensible Markup Language)を用いる方式を提案する。提案する方式は、LDAP をデータベースへのアクセスプロトコル及び情報モデルとして、XML をデータベースにおけるデータ格納形式として用いる。これにより、各種インターネットアプリケーションからLDAP経由でITSのアドレス情報へ容易にアクセスできるようになるとともに、XML で記述された情報を LDAP により流通させることが可能になる。また、プロトタイプシステムの実装を通じて、本方式の実用性を示す。

ITS Address Management Method

Using LDAP and XML

Keizo SUGIYAMA　Takayuki WARABINO　Hideyuki SHINONAGA

KDDI R&D Laboratories Inc.

Address is the information to uniquely identify objects to be communicated. It is necessary to manage address information like physical locations of vehicles and communication profiles in ITS environment. This paper proposes a novel method for ITS address management using Lightweight Directory Access Protocol (LDAP) and eXtensible Markup Language (XML). In this method, LDAP is for an access protocol and an information model, and XML is for a data store format. It allows us to facilitate the access to ITS address information from various Internet applications and spread ITS information described by XML via LDAP. The proposed method is shown as practically available through the implementation of a prototype system.

1. はじめに

通信は、人と道路と車両とを情報でネットワークするITS(Intelligent Transport System)の実現において、キーとなる重要な要素である。ETC(有料道路自動料金収受システム)やAHS(走行支援道路システム)では、DSRC(専用狭域通信)技術を使用して路車間通信を実現し、料金や走行支援に関する情報を授受している。

通信を確立する際には、通信する対象を一意に識別する情報が必要となる。このような識別情報やこれに付随した情報を一般に、アドレスと呼ぶ。車両の物理的な位置やDSRCのリンクID、IPアドレス、サービスプロファイルなどは、DSRCにおけるアドレス情報の例である。

ITSのアプリケーションは、アドレス情報を有効利用することで、状況に応じてカスタマイ

ズしたサービスを提供することが可能となる。例えば、車両の現在位置やサービスプロファイル等を取得することで、近くにあるガソリンスタンドや目的地までの道路情報を検索し、サービスプロファイルに応じて音声や画像によりその情報をドライバに提供するようなサービスが考えられる。

こういったサービスの実現には、アドレス情報の管理が効率的になされるとともに、各種ITSアプリケーションから容易にアドレス情報を検索できる必要がある。しかしながら、ITSにおけるアドレス管理手法については、これまで検討が行われていない。

本稿では、ITSのアドレス管理にLDAP(Lightweight Directory Access Protocol)[1]とXML(eXtensible Markup Language)[2]を用いる手法を提案する。まず、LDAPやXMLの概要について説明し、これらを組み合わせたアドレス管理の概念や有効性を示す。次に、本提案手法実現のポイントであるLDAP/XML変換方式について述べる。さらに、実装したプロトタイプシステムを用いて検索・更新性能を測定し、本方式が実用的に使用できることを示す。

2. LDAP 及び XML の概要

2.1. LDAP の概要

LDAPは、ITU-T(国際電気通信連合-電気通信標準化部門)勧告X.500[3]シリーズに規定されるOSI(開放型システム間相互接続)ディレクトリの概念をインターネット環境に適用するためにIETF(Internet Engineering Task Force)で開発されたものであり、現在はバージョン3である。LDAPは、TCP上で直接動作するとともに個々のデータ要素を単純な文字列形式で表現することで、OSIディレクトリに比べ軽量化を図っている。LDAPはサーバ・クライアントモデルに基づいており、サーバ・クライアント間のアクセスプロトコルが狭義のLDAPであるが、情報モデルやネーミングなども含めてLDAPと呼ぶことが多い。

2.1.1. 情報モデル

検索や更新などの操作で扱う情報の単位をエントリと呼ぶ。LDAPでは、図１に示すようにエントリはDIT(Directory Information Tree)と呼ばれる木構造として管理される。個々のエントリは、DITのルートノードから該当エントリまでの各枝に付与された相対識別名(RDN)を連結した識別名(DN)により、一意に識別される。エントリには複数の属性が含まれ、個々の属性は一つの型と一つ以上の値を持つ。例えば、personというエントリはsn(surname)やcn(common name)、telephoneNumberといった属性を含む。

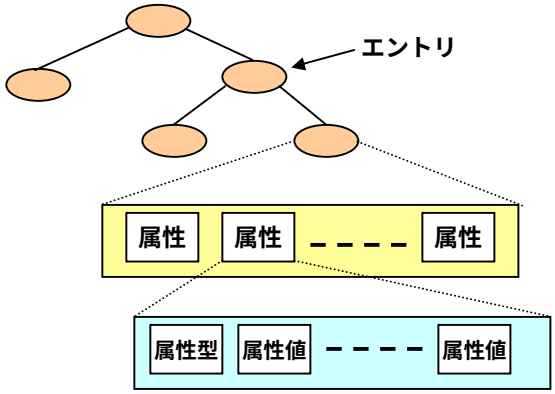

図1　エントリ、属性、型及び値の関係

2.1.2. LDAP 操作

上述のように格納された情報は、表1に示す９つの操作を用いてクライアントからアクセスされる。個々の操作には、それを制御するパラメタが複数存在する。

表1　LDAP操作の一覧

bind	ディレクトリと結合する
unbind	ディレクトリとの結合を解放する
search	指定されたエントリの情報を検索する
compare	指定されたエントリの値と提示値を比較する
add	エントリを新規に作成する
delete	既存のエントリを削除する
modify	既存のエントリの属性値を変更する
modifyRDN	既存のエントリの相対識別名を変更する
abandon	既に行った操作要求を破棄する

2.1.3. スキーマ定義

スキーマとは、属性型定義、オブジェクトクラス定義、並びに、エントリの属性に対し、どのようにフィルタや属性値宣言を適用するか、追加や変更操作を認めるかどうか、等を決定するためにサーバが用いる情報の集合である。

図2に、LDAPにおけるスキーマの定義例を示す。これら3つのうち、countryはRFC(Request For Comment)で定義される標準的なオブジェクトクラス、dsrcProviderと

dsrcSubscriberProfileは後述する性能評価のために独自に定義したものである。

```
①country オブジェクトクラス
objectClass ( 2.5.6.2 NAME 'country' SUP top
STRUCTURAL MUST c MAY ( searchGuide $ description ) )
attributetype( 2.5.4.6 NAME 'c' SUP name
SINGLE-VALUE )
attributetype( 2.5.4.14 NAME 'searchGuide' SYNTAX
1.3.6.1.4.1.1466.115.121.1.25 )
attributetype( 2.5.4.13 NAME 'description'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{1024} )
②dsrcProvider オブジェクトクラス
objectClass ( 1.2.3.1 NAME 'dsrcProvider' SUP top
STRUCTURAL MUST dsrcProviderId )
attributetype( 1.2.4.1 NAME 'dsrcProviderId' SUP
name SINGLE-VALUE )
③dsrcSubscriberProfile オブジェクトクラス
objectClass ( 1.2.3.2 NAME 'dsrcSubscriberProfile'
SUP top STRUCTURAL MUST vehicleId MAY ( ipHostNumber
$ dsrcGatewayId $ dsrcBeaconId
$ dsrcSubscriberProfile ) )
attributetype( 1.2.4.2 NAME 'vehcileId' SUP name
SINGLE-VALUE )
attributetype( nisSchema.1.19 NAME 'ipHostNumber'
DSEC ‘IP address as a dotted decimal, eg.
192.168.1.1, EQUALITY caseIgnoreIA5Match SYNTAX
‘IA5String{128}’ )
attributetype( 1.2.4.3 NAME 'dsrcGatewayId' SUP
name SINGLE-VALUE )
attributetype( 1.2.4.3 NAME 'dsrcGatewayId' SUP
name SINGLE-VALUE )
attributetype( 1.2.4.4 NAME 'dsrcSubscriber
Profile' SYNTAX INTEGER )
```

図2　スキーマの定義例

2.2. XML の概要

2.2.1. XML と DTD

XMLは、WWW関連の標準化を行っているW3C(World Wide Web Consortium)によって規定されたマークアップ言語であり、HTML(Hyper Text Markup Language)のようなシンプルなフォーマットで文書構造を記述でき、独自にタグを定義できる。XMLはアプリケーション非依存であり、タグをネストさせることでデータの階層的な管理が可能である。これにより、XML文書はデータとして処理ができる。

XML文書におけるスキーマはDTD(文書型定義)であり、これに従ってXML文書が作成される。図3に、XML文書の例を示す。図3で、”<!DOCTYPE”から”]>”までがDTDであり、それ以降をXMLインスタンスと呼ぶ。

```
<?xml version=”1.0” encoding=”Shift_JIS”?>
<!DOCTYPE 社員名簿 [
<!ELEMENT 社員名簿 （社員）+>
<!ELEMENT 社員 （部署、氏名)>
<!ELEMENT 部署 (#PCDATA)>
<!ELEMENT 氏名 (#PCDATA)>
]>
<社員名簿>
<社員>
<部署>無線通信グループ</部署>
<氏名>蕨野貴之</氏名>
</社員>
<部署>営業開発グループ</部署>
<氏名>杉山敬三</氏名>
</社員>
</社員名簿>
```

図3 XML文書の例

2.2.2. データ操作言語

上述したようにXMLはデータとして処理ができるため、データベースへ格納して操作言語を共通化することにより、各種のアプリケーションからの利用が容易になる。このようなXMLデータベースにおけるデータ操作言語として、XQLやXPath、XUpdateが開発されている。

(1)XQL(XML Query Language)[4]

XQLは、XMLの属性やプログラミング言語中に埋め込んで使えるシンプルな問合わせ言語として開発された。XQL式の評価対象となるノードのリストをコンテクストと呼び、'/'でXML要素の階層を、'//'で全ての子孫ノードを、'@'でXML属性を表す。'[ ]'はフィルタ演算子を表す。例えば図3のXML文書に対して、

/社員名簿/社員/氏名

というXQL式を実行すると、

<氏名>蕨野貴之</氏名>
<氏名>杉山敬三</氏名>

という結果になる。

(2)XPath(XML Path Language)[5]

XPathは、XML文書の特定の部分を指定するパス指定言語である。XQLの成果を反映した文法やデータモデルとなっており、①ルートノード、②要素ノード、③テキストノード、④属性ノード、⑤名前空間ノード、処理命令ノード、⑦コメントノードの7つのノードからなる木構造としてXML文書を扱う。また、XQLと同様に、フィルタ演算子を使用することもできる。

(3)XUpdate[6]

XUpdateは、XML文書中のデータの更新を行うための言語であり、XPath式で更新対照を

指定し、それに対してappend、update、remove等の要素を使用して更新する。例えば、図3のXML文書の最後に社員を追加する場合、以下のような式を発行する。

```
<xupdate:append select="/社員名簿" child="last()">
<xupdate:element name="社員">
<部署>無線通信グループ</部署>
<氏名>篠永英之＜/氏名＞
</xupdate:element>
</xupdate:append>
```

3. LDAP と XML を用いた ITS アドレス管理手法

3.1. LDAP と XML を用いた ITS アドレス管理手法の提案

2.1で述べたように、LDAPはインターネット環境で既に電子メールなどのアドレス情報の管理に使用されている。したがって、これらインターネットアプリケーションとの親和性を考慮すると、ITSにおけるアドレス管理手法としてLDAPを適用することが考えられる。しかしながら、通常LDAPは一旦格納された情報に対する検索系のアクセスが殆どであり、ITSにおける車両の位置情報のように頻繁な更新操作が発生する場合には、効率的に処理できない可能性がある。

一方、昨今ITSの分野において、XMLがデータフォーマットとして普及し始めている。例えば、MOSTEC(モバイル標準化検討委員会)により位置情報を端末間で授受するために開発された POIX(Point Of Interest exchange language)[7]や、北海道開発庁北海道開発局開発土木研究所により道路情報を記述するために開発された RWML(Road Web Markup Language)[8]などがある。

LDAPの実装は通常、フロントエンドのプロトコル処理部と、バックエンドのデータベース部の２つに分けることができる。このバックエンドにXMLデータベースを適用することにより、更新系の操作を効率的に処理できるようになるとともに、XMLにより交換されている各種の情報をLDAPで流通させることが可能となると考えられる。そこで本稿では、クライアントがアドレス情報にアクセスするためのプロトコルにLDAPを、アドレス情報の管理技術としてXMLを使用することを提案する。図4に、LDAPとXMLを用いたアドレス管理の概念を示す。図4の左側では、ITSアプリケーションやインターネットアプリケーションがLDAPによりアドレス情報を検索して利用しており、右側ではXMLデータベースを中心として、各種XMLフォーマットによりアドレス情報の交換が行われていることを示している。

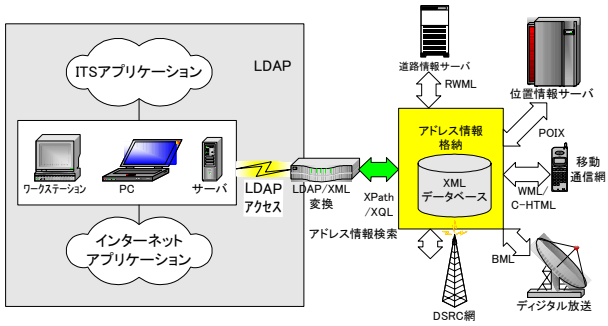

図4 LDAPとXMLによるアドレス管理の概念

本提案方式の実現にあたっては、図5に示すようにLDAPとXMLの間で変換を行う機能が必要となる。ここでは、この機能を実現するモジュールをLDAP/XMLコンバータと呼ぶ。

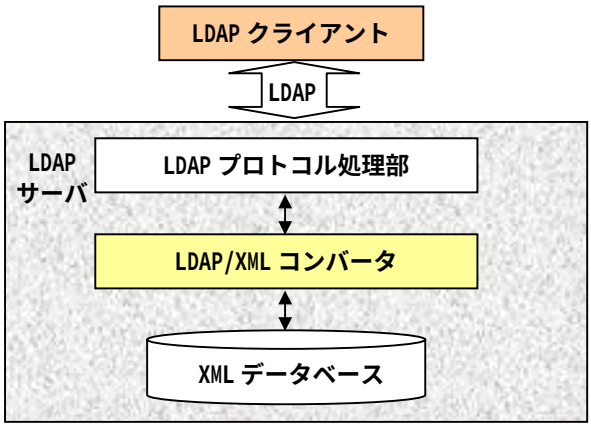

図5 LDAP/XMLコンバータの概念

3.2. LDAP/XML 変換方式

LDAP/XML変換では、LDAPのエントリの属性値をXML要素に変換する情報の変換とLDAPの検索操作をXQLなどのXML操作言語に変換する操作の変換が必要となる。

3.2.1. 課題

LDAPもXMLも、データモデルが木構造である。したがって、LDAPのスキーマをXMLの

DTDに対応付ける際には、インピーダンスミスマッチがないよう、階層構造を反映して変換する必要がある。

また、LDAPでは、属性の出現回数はスキーマで規定される。例えば、LDAPオブジェクトクラス定義において、MUSTと定義された属性は1回以上、MAYと定義された属性は0回以上出現する。また、LDAP属性型定義で、SINGLE-VALUEと定義された場合は単一の属性値を、MULTI-VALUEと定義された場合は複数の属性値を持つことができる。したがって、DTDにもこれらの情報を対応付ける必要がある。

実体であるLDAPのエントリとXMLインスタンスを変換する場合、LDAPには識別名により各エントリが一意に識別できるがXMLにはXML要素を識別する標準的な方法は存在しないため、XMLに識別情報を付加する必要がある。ただし、値はどちらもテキストで表現するため、変換処理は殆ど不要となる。

3.2.2. 前提

XML文書には、DTDに基づいて検証された妥当な(Valid)XML文書と、DTDに基づく妥当性は検証されないがXMLとしてのタグ付け規則に従っているかのみを検証した整形式(well-formed) XML文書がある。ここでは、妥当なXML文書を扱うものとする。具体的には、以下の項目を前提として変換を行う。

- XML文書は、XMLインスタンスに加えて、XML宣言とDTDを必須とする。
- DTD中では、要素型宣言と属性リスト宣言がなされている。
- XMLインスタンスでは、DTDに従ってタグ付けを行う。

3.2.3. LDAP/XML 変換処理のフローチャート

LDAP/XMLコンバータの処理は、図6に示すように、事前に登録されるエントリの情報を変換する処理と、LDAPクライアントから操作要求を受信した時に発生する処理に分けられる。

事前の処理では、スキーマ及び実体の対応付けを行い、XMLデータベースに登録する。また、LDAP操作を受信するたびにXML操作言語を生成してXMLデータベースにアクセスし、その結果をLDAPクライアントに応答することを繰り返す。

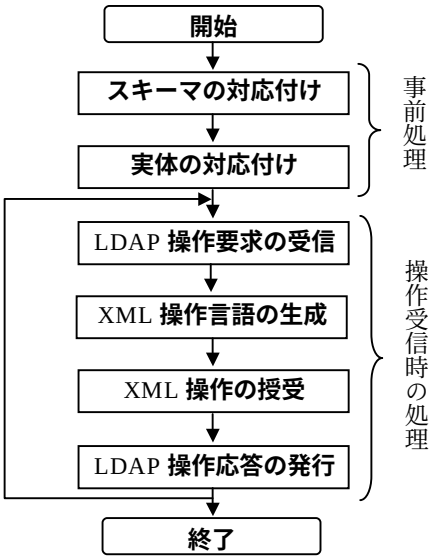

図6 LDAP/XMLコンバータのフローチャート

3.2.4. 情報の変換規則

(1)LDAP スキーマから DTD への変換

①一つのLDAPスキーマから一つのDTDを作成する。複数のXML文書から参照可能とするため、DTDは外部サブセットとして保持する。

②STRUCTUALなLDAPオブジェクトクラス、すなわちLDAPエントリになりうるLDAPオブジェクトクラスは、一つの要素型宣言と一つの属性型宣言に対応付ける。LDAPオブジェクトクラス名をXMLの要素名とする。

③STRUCTUALなLDAPオブジェクトクラスの全ての属性は、LDAPオブジェクトクラス名に対応するXML要素の子要素とする。すなわち、LDAP属性名をXML子要素名として要素型宣言を行い、個々の要素型宣言はカンマで区切って列挙する。

④DITの階層関係は、XML要素の包含関係に対応付ける。該オブジェクトクラスの属性に対応する要素型宣言の最後に、DITでの直接下位オブジェクトクラス名の要素型宣言を'*'で修飾して列挙する。DITのルートのオブジェクトクラスは、DTDのルートの要素名とする。

⑤LDAP属性の出現回数は、表2に示すようにXML要素の出現回数に対応付ける。

表2　LDAP属性の出現回数の対応付け

LDAPオブジェクトクラス定義	LDAP属性型定義	XML要素修飾子
MUST	SINGLE-VALUE	なし
MUST	MULTI-VALUE	+
MAY	SINGLE-VALUE	?
MAY	MULTI-VALUE	*

(2)LDAPエントリからXMLインスタンスへの変換

①STRUCTUALなLDAPオブジェクトクラスにおいて、識別名の情報は属性型宣言に対応付ける。具体的には、dnというXML属性名を定義し、その値としてLDAPエントリの識別名を大文字に正規化して持たせる。

②LDAPの属性値は、該XML要素のテキスト部分にそのまま対応付ける。ただし、漢字コードはどちらもUTF-8であるため、基本的に変換は不要であるが、XMLにおいて特別な意味を持つ特殊文字は、表3のように変換する。

表3　XMLにおける特殊文字の変換

文字	XMLインスタンスでの標記
<	<
>	>
&	&
‘	'
“	"

(3)変換例

図2のLDAPスキーマを、XMLのDTD及びXMLインスタンスに変換した例を図7に示す。

```
① DTD
<!DOCTYPE country SYSTEM “aaa.dtd”[
<!ELEMENT country (c, searchGuide*, description*,
dsrcProvider*)>
<!ATTLIST country dn CDATA #REQUIRED>
<!ELEMENT c (#PCDATA)>
<!ELEMENT searchGuide (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT dsrcProvider (dsrcProviderID,
dsrcSubscriberProfile*)>
<!ATTLIST dsrcProvider dn CDATA #REQUIRED>
<!ELEMENT dsrcProviderID (#PCDATA)>
<!ELEMENT dsrcSubscriberProfile (vehicleID,
ipHostNumber?, dsrcGatewayId?, dsrcBeaconId?,
dsrcServiceProfile*)>
<!ATTLIST dsrcSubscriberProfile dn CDATA
#REQUIRED>
<!ELEMENT vehicleID (#PCDATA)>
<!ELEMENT ipHostNumber (#PCDATA)>
<!ELEMENT dsrcGatewayId (#PCDATA)>
<!ELEMENT dsrcBeaconId (#PCDATA)>
<!ELEMENT dsrcServiceProfile (#PCDATA)>
]>
② XMLインスタンス
<?xml version=”1.0” encoding=”UTF-8”?>
<county dn=”c=jp”>
<c>jp</c>
<description>日本</description>
<dsrcProvider dn=”DSRCPROVIDERID=KDDI, C=JP”>
<dsrcProviderId>kddi</dsrcProviderId>
<dsrcSubscriberProfile dn=”VEHICLEID=KS1,
DSRCPROVIDERID=KDDI, C=JP”>
<vehicleId>ks1</vehicleId>
<ipHostNumber>1.2.3.4</ipHostNumber>
<ipHostNumber>5.6.7.8</ipHostNumber>
</dsrcSubscriberProfile>
<dsrcSubscriberProfile dn=”VEHICLEID=KS2,
DSRCPROVIDERID=KDDI, C=JP”>
<vehicleId>ks2</vehicleId>
<dsrcBeaconId>BS1</dsrcBeaconId>
</dsrcSubscriberProfile>
</dsrcProvider>
</country>
```

図7　情報の変換結果

3.2.5. 操作の変換規則

LDAPの操作は、表1に示した９種類が存在する。このうち、LDAPのSearch及びCompareはXPathやXQLのようなXMLの検索言語に対応付け、LDAPのModify、ModifyRDN、Add、Deleteといった更新系の操作はXUpdateのような更新系の言語に対応付ける。BindやUnbind、AbandonはXMLデータベースとのセッションの確立・解放等に該当するが直接対応付けられるXML言語はなく、使用するXMLデータベースに依存する。

XML操作は、①操作対象となるXML要素の指定、②指定されたXML要素配下における情報の検索または更新、の2つのステップに分けることができる。以下、各々について記述する。

①では、図2のスキーマの階層構造に対応した形でパスを指定するとともに、フィルタとしてパス毎に識別名を設定する。具体的には、図2のスキーマにおいて、以下の操作を発行する。

```
/country[@ndn='C=JP']/dsrcProvider[@ndn='
DSRCPROVIDERID=KDDI1,C=JP']/dsrcSubscriberP
rofile[@ndn='VEHICLEID=KS10000,DSRCPROVIDER
ID=KDDI1,C=JP']
```

②では、SearchやCompareにおける属性値の取得は①と同様にパス指定を行い、必要な属性を取得するためのフィルタを最後に設定することで実現する。Modifyは、まず該エントリの属性値取得操作を行い、次に変更する属性値をテキスト部分に設定したXML要素を持つXML更新操作を発行する。Addは全属性値をModifyすることに等しいが、その前にModify追加対象のエントリ及びその親エントリの存在を確認するXML操作を発行する。Deleteでは、削除対象のエントリの子エントリが存在しないことを確認するXML操作を発行する。

4. LDAP/XML 変換方式の評価

4.1. 実装

図５の構成に対応したプロトタイプシステムを実装した。動作環境はWindows2000™の動作するパソコン(PentiumⅢ1GHz、メモリ512Mbyte)である。

4.1.1. LDAP クライアント

LDAPクライアントからLDAPサーバに発行する一連の操作の手順をシナリオとして事前にファイルに登録できるようにした。シナリオでは、LDAP操作要求の他に、LOOP（指定個所を指定回数繰り返し実行する）、WAIT（指定操作の実行を指定時間遅延させる）、RESULT（非同期のLDAP操作要求に対する結果を取得する）コマンドを設けた。このシナリオファイルを読み出して実行し、各操作に要した時間を表示するとともに、特定の操作の実行結果を詳細に表示できるようにした。

4.1.2. LDAP プロトコル処理部

LDAPプロトコル処理部には、LDAPのフリーソフトとして著名なOpenLDAP™をWindows2000に移植して使用した。

4.1.3. XML データベース

XMLデータベースには、eXcelon™を用いた。eXcelonは、オブジェクト指向データベースであるObjectStoreをDBMS（データベース管理システム）として使用している。eXcelonでは、更新操作に独自の言語を用いている。

4.1.4. LDAP/XML コンバータ

OpenLDAPのフロントエンドは、LDBM互換バックエンドと呼ばれるインデックス型のデータベース(BerkleyDB)に対応している。そこで、フロントエンドからLDBM互換バックエンドを呼び出している関数はそのまま使用し、その中で実際のデータベースにアクセスしている部分をeXcelonのAPIに置き換えることで実現した。

4.2. 測定結果

LDAPクライアントから各種の操作を発行した場合の応答時間を表4に示す。測定では、country、dscrProvider、dsrcSubscriberProfileのエントリ数を各々1、1、10000とし、検索等の操作は10000番目のエントリに対して発行した。値は、5回測定した結果の平均値である。なおInitはLDAPクライアント内のローカルな処理であり、Unbindはサーバからの応答がない一方向の操作である。

ここでは、高速化のため、eXcelonのインデックス機能を利用し、識別名を保持する属性にインデックスを設定した。また、比較のため、①LDAPクライアントとサーバは同一マシン上で実行しデータベースはeXcelonを使用、②LDAPクライアントとサーバは同一マシン上で実行しデータベースはBerkleyDBを使用、③LDAPクライアントとサーバは100BASE-TXで接続された異なるマシン上で実行しデータベースはeXcelonを使用、の３つの場合を測定した。

表4　測定結果(単位：秒)

LDAP操作	①	②	③
Init	0.001	0.001	0.003
Bind	0.029	0.014	0.039
Unbind	0.002	0.002	0.003
Search	0.039	0.001	0.041
Modify	0.113	0.090	0.132
Add	0.985	0.153	1.000
Delete	1.352	0.136	1.335
ModifyRDN	1.557	0.248	1.581
Compare	0.021	0.001	0.022

4.3. 考察

(1)　処理時間について

アドレス管理において通常使用されるLDAP操作は、位置情報のようなアドレス情報を検索するSearch操作と更新するModify操作であると考えられる。

表4において、10,000エントリ登録時、最後のエントリのSearch操作には、提案方式（パターン①）で約40ms、元のOpenLDAPの場合（パターン②）で1ms程度である。また、Modify操作の場合には、パターン①で約110ms、パターン②で約90msである。

30mセルのDSRCスポットを車両が180km/hで走行することを考えると、その通過時間は600msであり、その数分の一の時間で情報の検索及び更新が完了する必要がある。このことから、提案方式をITSのアドレス管理に適用した場合においても、検索・更新処理性能とも実用上は問題ないといえる。

また、Add、Delete、ModifyRDNなど、XML要素の追加・削除が行われる場合は、提案方式では1秒以上要している。したがって、新規車両の登録のようなエントリの追加や削除操作は、オフラインでXMLデータベースにアクセスする等で対処するのが望ましい。

(2) XMLデータベースの性能について

LDAPサーバにおいて、検索処理時間は次式で求められる。

検索処理時間＝ base エントリの情報取得 + 候補エントリの検索　+ base エントリ下 (base 含む) の全エントリの情報取得

使用した情報モデルでは、アドレス情報はdsrcSubscriberProfileに含まれており、検索対象エントリの識別名は既知であるという条件下では、2項目の候補エントリの検索処理は不要である。また、XMLデータベースにおいて、単一エントリの情報取得に要する時間を別途測定した結果、約17ms要した。したがって、上記の式にあてはめると、XMLデータベース部では検索処理に17×2　34ms要する。これから、LDAPプロトコルハンドラ及びLDAP/XMLコンバータのオーバーヘッドは数ms程度であり、XMLデータベースの処理時間が大半である。

次に、更新処理時間は次式で求められる。

更新処理時間＝自エントリの情報取得＋自エントリの情報更新

10000番目のエントリの情報を修正する場合、別途測定したXMLデータベース単体の処理時間は各々17.0msと97.9msであり、合わせると115msとなってほぼ表4の値に近く、やはりXMLデータベースの処理時間が大半である。

OpenLDAP標準のBerkleyDBは、特に検索系の処理時間を小さくするためにインデックスの持たせ方などを工夫している。一方、今回使用したXMLデータベースであるeXcelonはDBMSにObjectStoreを使用しており、XMLの木構造をオブジェクト指向モデルにそのままマッピングできるという利点がある。しかしながら、ObjectStoreは汎用DBMSとしてトランザクション管理等の様々な処理を行っており、その負荷が大きいと思われる。

このように、今回の実験ではXMLデータベースの処理時間が支配的であるため、XMLデータベース自体の性能が向上すれば本方式の性能も向上すると言える。また、更新操作は検索操作の場合ほど、パターン①と②の処理時間に差がないことから、一般のLDAPサーバは検索系の操作に向いていることがわかる。

(3) LAN接続のオーバーヘッドについて

表4で、InitやUnbindのように確認を要さない操作は数msで終了している。また、パターン①とパターン③において、LAN経由の場合とそうでない場合の処理時間の差異は殆どなく、LAN接続のオーバーヘッドはほぼ無視できると言える。ただし、ネットワーク機器の処理遅延はルータの段数などで変化するため、実際のネットワーク構成においては、別途検討する必要がある。

5. おわりに

本稿では、ITSにおける車両の位置情報や通信プロファイルなどのアドレス情報を管理する手法として、LDAPとXMLを用いる方式を提案し、そのポイントとなるLDAP/XML変換規則を示した。本方式により、各種インターネットアプリケーションからITSのアドレス情報へ容易にアクセスできるようになるとともに、XMLで記述された情報をLDAPにより流通させることが可能になる。また、プロトタイプシステムを実装し、本方式がITSのアドレス管理において実用的に使用できることを示した。なお、本研究は通信・放送機構(TAO)による委託研究の一環として実施している。最後に日頃ご指導頂く(株)KDDI研究所浅見所長、松島副所長、水池執行役員、加藤執行役員に感謝します。

参考文献

[1]:RFC2251, “Lightweight Directory Access Protocol (v3)”, 1997
[2]:W3C Recommendations, “Extensible Markup Language(XML) 1.0 (Second Edition)”, http://www.w3.org/TR/2000/REC-xml-20001006
[3]:ITU-T Rec. X.500, “The Directory – Overview of Concepts, Models and Services”, 1998
[4]:W3C, “XML Query Language”, http://www.w3.org/TandS/QL/QL98/pp/xql.html
[5]:W3C,”XML Path Language Version 1.0”, http://www.w3.org/ TR/XPath
[6]W3C, “XUpdate - XML Update Language”, http://www.xmldb.org/xupdate
[7]:モバイル標準化検討委員会, “Point Of Interest eXchange language”, “http://mostec.aplix.co.jp”
[8]:北海道開発局開発土木研究所, “Road Web Markup Language”, http://rwml.its-win.gr.jp/