XML on LDAP Network Database[1]

K.L. Eddie Law
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario,
Canada M5S 3G4

***Abstract*--This paper outlines a design that uses the eXtensible Markup Language (XML) to describe network database schema. The XML information will be encapsulated in the Lightweight Directory Access Protocol (LDAP) messages. LDAP has already established itself as a popular directory search protocol. Typically, the LDAP response message can carry unintelligible information at the LDAP layer. In the design in this paper, this information will be passed to an XML parser and translator for further processing. As a result, we can modify, update database information without taking down the whole system. This is the main objective in combining both XML and LDAP in a network model. We are investigating two aspects in this paper. The first objective is to use the flexibility and extensibility of XML that allow us to change the database contents dynamically. The second objective is to demonstrate that the combination of XML and LDAP can be a powerful tool in delivering dynamic database content to different recipients on-the-fly.**

***Index terms*—XML, LDAP, DTD, UIM, content, database.**

A. INTRODUCTION

In current networking information age with high network bandwidth availability, many large-scale network-wide applications can be implemented. Application-on-tap approach allows you to obtain services from Internet when you need to access and operate on your information and data stored somewhere in the networks. Therefore, there are no needs to have your data stored and administrated locally. Similarly, it is becoming more common for applications to be distributed across internal and external servers. We may need a directory service that helps users to locate network servers.

From the application or service information providers, it will be desirable if they can provide and adjust information according to the required data structure, according to the requirement of different end users. This is especially important if information can be delivered on-the-fly to different end users without re-constructing database structure at the provider side.

X.500 is originally designed to have a global and distributed directory service. It contains directory access mechanism and can hold a rich variety of information. However, it is too complicated to provide a full-blown X.500 service and implementation. The Lightweight Directory Access Protocol (LDAP) [1] is a very popular replacement of X.500 because of the simplicity of its network protocol and ease of database search, retrieval, modification and storage.

However, more diverse networked applications can be built with the rapid advancement of network technologies. Therefore, different data structures are required for different applications. It will be desirable to use one single search command and retrieve appropriate information that an end user is looking for. Even in the case of one single type of application, different vendors usually have different data structures and representation methods. Therefore in this paper, we are investigating and designing to use only one single directory service to handle information for multiple applications and multiple vendors. In this regard, we shall investigate if eXtensible Markup Language (XML) can make a good presentation language for extensible data representation.

Since the works being indicated in this paper are still under designed in research phase and testing in experimental phase, we shall need more experiments to verify the operations of this concept by combining both XML and LDAP together. In the following, we

[1] This paper is part of the UPM (Unified Policy-Based Management) project that is being supported by Nortel Networks.

shall use several paragraphs in describing both the LDAP and XML. Then we shall outline the related experimental work being carried out at the University of Toronto that merges XML and LDAP in the testbed.

B. LDAP AND XML

1) *LDAP Network Database*

LDAP provides directory access mechanism that allows globally distributed information to be stored uniquely in a database. The LDAP composes of two main portions. One part will be the network protocol standardization. Another part indicates the structure of the directory schema. The schema can be considered as a database structure. This is similar to the concept of management information base. Typically, the information is stored in a tree structure. Every entry has a unique Distinguished Name (DN). Given a base DN, we can use a Relative Distinguished Name for an entry with a pre-defined base directory tree. The following example shows an entry with a unique DN:

```
cn=Eddie Law, ou=Communications Group,
  ou=Department of Electrical and
  Computer Engineering, o=University of
  Toronto, c=Canada
```

where cn, ou, o and c represents comman name, organizaional unit, organization and country respectively.

The important design of the LDAP is on the directory access. The simple protocol LDAP structure enables a client to send requests to a directory and receive responses. In LDAP, there are several different types of requests. The central purpose of a directory service is to enable people or programs to search for information, thus the search request message is the an important LDAP operation. Currently, LDAP is assigned to use TCP port number 389 or 636 if using encryption.

In order to retrieve the content being stored in a LDAP database, we can follow the LDAP URL setup to inform a remote LDAP server from to search for a database entry. With the LDAP URL, we can use web browser to enable LDAP commands. Please see [1] for more information.

2) *XML*

XML [2] was originally designed to replace ASCII text in data representation and its design is based on SGML (Standard Generalized Markup Language). Similar to the case that LDAP is a simplified version of X.500, XML is a simplified version of SGML. The extensible idea developed in XML allows it to be widely and flexibly used in many different applications. XML is a trivial replacement of HTML which should be operable in all web browsers in future. With current development, the extensibility of XML can represent almost any type of documents, for example, a database or a directory structure.

In XML document environment, an XML document has to be well-formed and valid. In order to parse an XML document, we require to have the rules to check the well-formness and validity of an XML data document, this document. This document is known as the Document Type Definistion (DTD). An XML parser needs to have a proper DTD in order to interpret the data arrangement in a given XML data file. There are only a few reserved words in XML, similar to HTML, XML is a text document with tags or keywords in angle brackets. The data schema is defined in the DTD file. One interesting aspect of XML is that even though XML data document itself is extensible, DTD is not. Hence, there are approaches people is investigating to provide extensibility on the DTD.

In order to handle different types of data formats for different applications, we need to have an engine that can send out different DTDs or the differences of DTDs. More explicit design is in the following section.

3) *XML on LDAP Network Database*

LDAP is already a popular choice in keeping directory information. It is quite common to combine LDAP with SQL server since LDAP mainly provides directory service with comparatively limited database keywords and mechanisms. SQL database is being used mostly together because it has been used for different applications for quite a long time already. However, in this paper, the main objective of this paper is to setup a database structure using XML with the LDAP simple network directory search and update capability. The extensibility of the XML provides an important part in providing flexibility in database schema construction. This means that no matter what the format of the data schema is required, we can adjust the DTD document and deliver it to the receiver side for proper XML translation.

To provide more flexible design on DTD, XMI (XML Metadata Interchange) can be investigated. In our lab experiment, we create simple difference, add and delete operations on DTD document. However, we shall not cover it in details in this investigation paper.

We have not yet described any system architecture designs. This is because XML on LDAP creates a tool that provides a unified information model (UIM) for different applications. UIM does not restricted itself on any system designs because we can always implement an XML translator with LDAP client implementations are independent to network architecture design.

C. Unified Information Model

The beneficial part after designing a unified inforamtion model is that we can always assume that data passing through the networks can be correctly interpreted by the receiver. However, we still have to determine where the UIM receiver should locate in a system design. There is one more problem we need to solve which is "we should put XML documents in what types of LDAP messages.".

There are two basic XML documents that may need to be passed through the networks, one is the XML data file and the other one is the associated DTD if in case we are not using a DTD difference file. Amongst all LDAP messages, LDAP search response message allows a result entry to be passed to a receiver and the LDAP client is not required to understand the message content of this response message. The LDAP client then sends the encapsulated information to an XML parser and interpreter which shall further process the message and pass the resultant information to the appropriate applications. Therefore, LDAP message is made extensible after combining with XML.

In the following, we provide a system architectural model that uses the proposed UIM. UIM is useful in mutli-tier network architecture. This means that any system using middleware approach can deploys this solution. The example shown below is an application running on a multi-tier network system design. UIM constitutes the back end control information storage and passing mechanism while the front end operations are decided after the control information is delivered from network server and gets interpreted by the middle-tier entity.

1) Example: Policy-Based Management in Internet

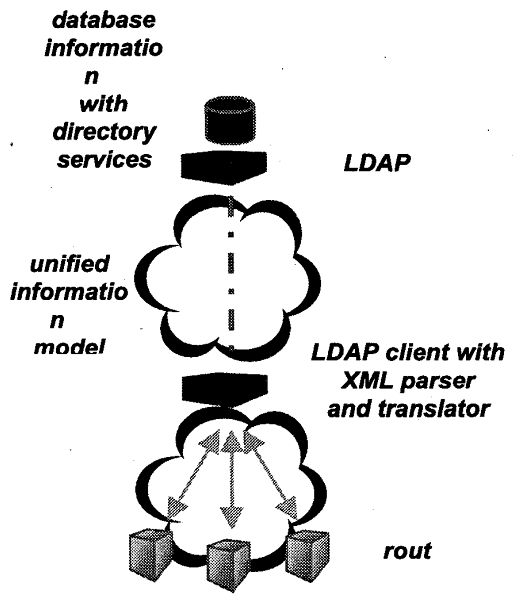
database informatio n with directory services
LDAP
unified informatio n model
LDAP client with XML parser and translator
rout

Figure 1: a multi-tier architecture.

This section indicates the experiments being carried out at the University of Toronto. The IP networks can be partitioned into many policy domains. In the laboratory, we currently have one policy domain and we have several routers in this domain. In policy-based management, there constitutes another protocol known as Common Open Policy Service (COPS) protocol but we will not investigate this protocol in this paper, and the system design is modified to use only LDAP operations in this paper. The original system design requires the policy server to run as a LDAP client.

In Figure 1, we show a three-tier network architecture in the network policy control example. The fundamental reason is that policy control is still on early draft version, the design of the policy rules and policy schemas are on the fast lanes and may change rapidly. However, many network products vendors at this stage are so eager to deliver products with their own proprietary schema structure.

In the network, the policy request information is typically sent through the network router to the LDAP server. Upoin receiving information from the LDAP server, the middle-tier entity (ME) can interpret the database information, finalize and carry out the policy-related control decision. One main result we are investigating to provide a unified information model for policy-based management in Internet. This section describes a system design that will be highly beneficial upon using this proposed XML/LDAP design.

In this network model, the middle-tier entity processes many intelligent operations. It receives proprietary message requests which request for certain types of traffic quality. Upon receiving these policy control query messages and through the LDAP client process, ME sends a LDAP message to the LDAP server to obtain database information for those routers under the control of this specific ME. An example of the LDAP URL is:

```
ldap://ldap.Policy.server/o=University%
  20of%20Toronto,c=Canada??sub?(cn=Switc
  h*)
```

This message informs the LDAP server to search for the control information regarding the Switch at the University of Toronto. We may obtain more than one entry information with this kind of non-explicit LDAP request message. Therefore, it will be the role of ME to identify the correct entry information. And from this information, ME passes proprietary control message to the router to carry out appropriate action control.

When the LDAP server receives a request, this LDAP message will be translated and converted to XML search. For example, we may have only one search result obtained and they are shown in Figure 2.

These simple XML documents retrieved from the database by the LDAP server and they are sent back to ME. These documents will be encapsulated in the LDAP response message. The LDAP client will pass the content to the XML parser and get translated according to the vendor's control code. DTD and XML files are separate files and new DTD is required if its associated database schema is modified. On the other hand, as long as the new requesting database schema does not change, DTD files are not required to send again to the receiver side.

```
<!DOCTYPE Switch [
  <!ELEMENT Switch (Name, Interface+)>
  <!ATTLIST Switch Admin CDATA
  #REQUIRED>
  <!ELEMENT Name (#PCDATA)>
  <!ELEMENT Interface (#PCDATA)>
  <!ATTLIST Interface id ID #REQUIRED>
]>

<?xml verison="1.0"?>
<Switch Admin="Someone">
  <Name>edge1</Name>
  <Interface id="1">Only X can go
  through.
  </Interface>
  <Interface id="2">IP interface
  w.x.y.z.
  </Interface>
</Switch>
```

Figure 2: Examples of the DTD and XML Files

One project on the side being investigated is relating to the creation of a DTD update. In that project, we shall not send the whole new DTD file again and again. Based on the concept of incremental modification on network database, we shall need to create difference, add and delete messages to the XML parser at the receiving end. This concept is very similar to XMI; however, we do not want and do not need to have a full-fledge implementation of XMI specification. We only want to create simple message passing between server and client host. As a result, we can further improve the dynamicity of database construction. At the current phase, we are implementing different database files with different DTD constructs. Upon the completion of this new DTD updating part, we can combine all database file into one single file, and we only need to send the differences between the old and new DTDs.

The reader may also have noticed that, in the UIM design, we requires to provide MEs with intelligence to do proper control message translation. This is because different vendors have different control message formats and structures. That means that we still have to create several translating modules at the ME. However, the construction of these translating modules are outside the scope of this paper. We are investigating to use Java as a programming language to build these translation modules. This is because Java does not require compiler to operate on a Java program. If there are Java interpreter, such as inside a web browser, then we can get these modules running on-the-fly. Currently, the designs of these modules are still under investigation.

D. CONCLUSIONS

This paper introduces the concept of using XML data with DTD schema stored inside the LDAP directory tree. The extensibility of XML creates the flexibility in setting up data content according to different applications and/or for different vendors.

The combination of XML and LDAP create the concept of UIM. Under this model, the information content is totally transparent to the whole hardware configuration architecture. Therefore, we do not need to change system architecture or take down any hardware components in order to upgrade the system and modify the database schema. Hence this model allows incremental improvements.

There is also a topic that we have not addressed in this paper. This is to design the UIM with load sharing and balancing mechanisms, especially when the loads being applied to the LDAP server are very heavy. In reference [3], there proposed a system architecture for load balancing and load sharing in WWW scenario to enhance the server performance. This approach can also be investigated in applying similar concept to this LDAP server case.

REFERENCE

[1] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)," IETF RFC 2251.

[2] Extensible Markup Language (XML) 1.0, W3C Recommendation REC-xml-19980210, Feb. 10, 1998.

[3] K.L.E. Law, B. Nandy, A. Chapman, "A Scalable and Distributed WWW Proxy System," Proc. IEEE Conf. Multimedia Computing Systems (ICMCS) '97, pp.656-571, Ottawa, 1997.