A Taxonomically Distributed Data Retrieval Model

M. Marquina R. Ramos Pollán A. Taddei

CERN, IT Division
CH 1211 Genève 23, Switzerland

As many arising directory services are being implemented and a variety of collections of data are being made available, it becomes increasingly important for users the need of guidance through such a network of heterogeneous data. At CERN, different Physics Collaborations are setting up systems to provide data related to themselves (people, experiments, applications configurations, etc.). With this variety of services, when a generic CERN user wants to find a very determined piece of data (for instance "event number 24971 temperature conditions"), which service should he ask to? We present here a model aimed at solving this Data Retrieval problem. This is done by proposing a protocol for computers to exchange information about the data they offer and setting up the basic mechanisms to build up, maintain and query a distributed taxonomy of data. This taxonomy will be used to find a set of locations where a given user's query is likely to retrieve significant information.

Key words: data retrieval; WWW; taxonomy; terminology; directory services

1 Introduction

With the growth and popularity of Internet based services a huge amount of data has been made accessible by virtually everyone. Many kinds of data can potentially be offered in such a way, from people directories (*white pages*) to repositories of books and catalogued documentation, etc. This raises two problems. On the user (or *client*) side (s)he must know **where** to search (this is, what server should be asked) and **how** to search. On the data provider side the problem is that about terminology [8] (i.e. fields or concepts which are semantically identical are not called the same by different providers) and structure (which services are more general and how related services could be advertised homogeneously to the user).

Preprint submitted to Elsevier Preprint *29 April 1997*

The model discussed in this paper [1] pretends to establish the basic mechanisms to promote and easy up collaborative cooperation towards achieving commonly accepted terminology and structure in the data offered to the community in a local and (possibly) global way. To achieve this we introduce the concept of a **Metadata Advertising** mechanism which offers a common ground for (1) data providers to structure and unify the services they provide and (2) helping users find a set of places where their search is highly probable to give satisfactory results. Also we present a prototype implementation of these ideas.

2 Model Architecture

The model it not aimed at imposing a universal terminology or hierarchy of the services which may exist. We cannot pretend to possess all the criteria and knowledge enough to create a commonly accepted terminology and classification of ideas held by other people. Many times only the experts in one field are the ones able to make a practical organisation of it and achieve a common terminology.

The main idea if that of a **Metadata Advertising Protocol** (MAP) which links clients and data providers to retrieve and publish information about the data available, with mechanisms to stimulate understanding and agreement among the participants.

2.1 Model Components

The model is intended to be applied mainly for directory and structured data scattered across the net. Within this context there is a need to formalise the elements which come into play in order to offer a coherent model architecture: the network of nodes within which the model operates and the data it handles.

2.1.1 Network Nodes

From the logical point of view, initially there are two kinds of already existing nodes: **Directory Service Providers** (DSPs), which provide data as it is

[1] This paper deals with the **data retrieval** [2] problem which should not be confused with the **information retrieval** problem although they may share common tools and protocols. This later one has a broader sense and deals with highly unstructured data. *Data providers* are also referred throughout the paper as *Directory Services Providers* or *Repository Providers*.

described below. **Clients**, which access DSPs through queries and using some kind of method or protocol that they **must** share (like HTTP, FTP, LDAP, ACAP...).

The model introduces another kind of nodes, known as the **Metadata Advertising Nodes** (MANs), which maintain information (*metadata*) about DSPs' contents using a **Metedata Advertising Protocol** (MAP).

It is important to make a distinction between a logical node and its physical location (or machine). A machine can hold any number of logical nodes, offering different functionalities to the community. This is, a machine could hold several DSPs, a MAN and run client applications. Interactions among the three kind of nodes is described in the next section.

2.1.2 Data Model

The model is not designed to cope with all the information available in any form, but with structured data. So it won't be suitable (although may be useful) for users searching things like: *I want to learn how Aleph is designed* but for more specific queries like *I want to know where I could look for somebody's address.*

In our first approach, DSPs included in this model are those providing **collections** of **objects**, each being described by a set of `attribute:value` pairs. A collection has a set of descriptive labels, optionally belongs to a hierarchical category and must only hold one kind of objects. This is, objects within a collection must share a minimum set of attributes (although it is not required that all objects in a collection share ALL their attributes). One DSP can only provide one collection, although different DSPs may live in the same physical location.

For instance, the people directory service at CERN, is a collection of **people** (which are the objects) who work at CERN in the particle physics, computer science and engineering fields. One **person** is described by its *id, surname, firstname, email, office location, phone number, division* and *computer accounts.*

2.2 Governing Mechanisms

As we can see in figure 1 at the core of the model there is a *Metadata Advertising Protocol* (MAP), which basically maintains and organises in a distributed manner a table of information (the **metadata**) about the DSPs existing in the net. This is stored in the MANs and each one of them contains, in principle, the same metadata. The overall mechanism works like this: DSPs use the MAN network to distribute information about the objects they provide. When a new service is offered, the new DSP should tell a MAN about it. The MAN network organises and spreads out such information, taking care that DSPs providing the same kind of information should do it in the same terms. When a client wants to find a particular object, (s)he should first search or navigate through the metadata provided by any MAN and select a set of DSPs where to make a query following their guidelines.

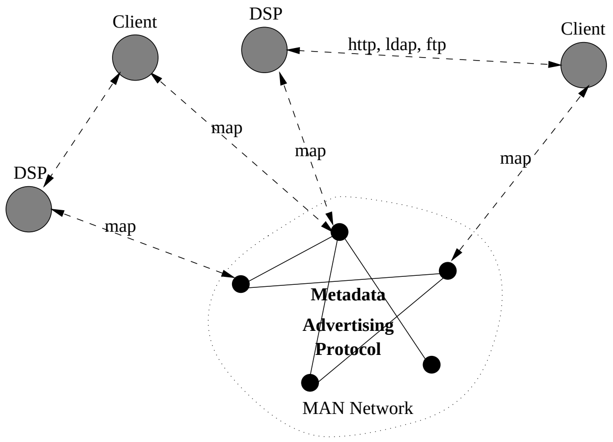

Fig. 1. Governing Principles

3 The Metadata Advertising Protocol (MAP)

In order to achieve this, the must exist a MAP which should (1) maintain and globally distribute the metadata, (2) organise the information for an easy navigation of users, (3) provoke agreements on DSPs providing the same kind of data and (4) provide a common language for DSPs to specify the contents and clients to understand it, this is, a language to write the metadata. Here we settle down the main designing principles of the MAP protocol. These principles are the buildings blocks for the MAP specification which will be outlined later.

3.1 Globally Available Metadata

First, there is a need to maintain and globally distribute the metadata about the services available. This is totally carried out by the MANs and, in this aspect, MAP principles are very similar to NNTP [6]. MANs behave like NNTP servers. Just like NNTP servers maintain a globally available amount of news articles, MANs maintain a table of DSPs which is continuously fed to among MANs. New entities in the metadata (advertising new DSPs) are spread out just like news articles, so that they reach every MAN. This makes clients and DSPs independent from location and access point in the MAN network within a delay of a few days (like news articles). Considering that the dynamics of DSPs (creation and deletion) are much less changing, this is a reasonable approach.

3.2 A Metadata Language

We need a common language for DSPs to organise and tell MAN about themselves, and for clients to make an effective use of the information about DSPs. There are some independent efforts within the Internet community to provide some kind of *contents file format* or *metadata format* and mechanisms to manipulate it within different contexts ([1], [9], [3]). We have initially taken a very simple approach following the ideas in the RDM/SOIF model ([4], [5]) in order to make a prototype implementation so that we can test our ideas. See section 5 for possible lines regarding this issue.

In our initial implementation, the metadata existing in the MAN network is made of three kinds of entities and each entity has some attributes describing it. See the example below for an illustration of this. Following the concepts introduced in section 2.1.2 these entities are:

- **DSPs**. An specification of an DSP must include some data about the objects it contains (type and attributes), some description, and information about how to search it and for its classification. The usage of these later ones will be explained in the next section.
- **Object types**. An object type basically defines a standard set of attributes for objects which can be part of collections offered by DSPs. Object types are provided as means to unify data in different DSPs which is semantically similar. See point 4.3 for more information.
- **Categories**. Which are used to create a tree to hold DSPs. Under certain circumstances, DSPs offering related objects can associate and create a common designation for the kind of services that they offer. This way they may create a new category which must also be advertised. Categories are arranged in a hierarchical way so that each category has a parent but the top one. This way we support a `taxonomy` of the data. Each DSP should belong to a category, so that they are arranged hierarchically.

An example of each entity can be seen here [2] :

[2] At the time of writing the final format and structure of the MAP protocol and its components is not yet defined. Although some aspects may differ in the near future from what is being stated here, the main principles and mechanisms will remain

```
Entity:         dsp
Name:           CERN's combined directory
Location:       http://consult.cern.ch/xwho
Admin:          Phone.Master@cern.ch
Objecttype:     person
Status:         not classified, object type normalised
Category:       white_pages.science.physics
Attributes:     firstname, surname, email, phone, division

Entity:         category
Name:           white_pages
Parent:         .

Entity:         category
Name:           science
Parent:         white_pages

Entity:         category
Name:           physics
Parent:         white_pages.science

Entity:         objecttype
Name:           person
Attributes:     firstname, family name, address, phone, email
Synonyms:       surname:family name, telephone:phone
Content:        people, individuals, particular
```

Here we have the description of a DSP (CERN's phone book) which contains objects of type `person` and belongs to the `white_pages.science.physics` category. A small category tree is defined by the three **category** entities. And last, the **objecttype** `person` is defined, declaring its standard attributes and some synonyms to them. For instance, there is an attribute called `family name`, but DSPs can refer to it also as `surname` (like CERN's phone book), and users accessing the metadata won't encounter any ambiguity.

3.3 Organising the Information and Achieving Common Terminologies

So how is a global taxonomy and the terminology for the different types of objects defined? Initially, DSPs are registered in their own terms, this is, they are unclassified and the attribute names of the objects it contains are just what the DSP manager defined. At the same time a few MANs must analyse the information held in the metadata searching for possible similarities among DSPs and the objects they provide.

MANs detecting this kind of similarities should call the involved DSPs for a possible agreement on (1) the object names of the entities that they offer, (2) the objects they might share, their attributes and names and/or (3) a creation of a new category or their inclusion in an already existing one. Should the called DSP reach an agreement on some of the issues the corresponding changes in their entries must be updated and adequate `objecttype` and `category` records should be created or updated.

The use of aliases and synonyms is intended to make understanding among DSPs easier. Clients searching DSPs should construct their search using their terminology, translating synonyms as needed. Also DSPs should do some kind of lookup of unknown attributes when answering requests from clients. DSPs considering they offer the same type of objects don't have to sacrify their internal structure for the shake of compatibility. A definition of a common set of attributes (with an `objecttype` entity) is inherently attached to a definition of commonly accepted synonyms. DSPs are free to use synonyms as they wish and don't have to modify their databases schema in order to reach compatibility.

4 Proof of Concept

We have prototyped the ideas underlying this model in a very simple implementation of a MAP server and a client. The server basically provides a place to hold metadata as showed in section 3.2 and a few commands to access and search it.

4.1 MAP Experimental Server Specification

MAP is based on a simple request/response client/server model. A MAN implements a MAP server. Requests are just queries on the metadata, and responses are the list of entities which match the query. As mentioned before, the metadata is made of three types of entities: `dsps`, `objecttypes` and `categories`. Each entity is characterized by a set of fields. This specification makes no assumption on how a MAP server stores or uses the metadata.

4.1.1 Requests

There are the following set of possible requests:

– `GET <entitytype> <field> <regexp>`: Will make the server give a response with all the entities of `entitytype` having their `field` attribute

matching the `regexp` regular expression. Thus all entities given in the response are of the same type. The meaning of the fields is the following:
  · `entitytype`: The kind of entity to return. It must be `category`, `dsp` or `objecttype`.
  · `field`: The attribute of `entitytype` to match `regexp` against. You can see a list of fields for each category in the example in the previous section. Non existing fields will cause a zero match (no entities will be returned).
  · `regexp`: The regular expression to match `fields` in `entitytype`. Must be constructed following PERL rules. This is made in order to ease up the design to the experimental server which is also made in PERL.
- `SEARCH <regexp>`: Will make the server give a response with all the entities where any of its fields match the `regexp`. `regexp` has the same semantics as in the GET request. Entities given in the response won't necessarily be of the same type.
- `WHATSNEW <date>`: Will make the server give a response with all the entities which have been added after the specified date. This command is implemented in order to enable MANs to synchronise their contents.

4.1.2 Responses

Responses are given a list entities separated among them by a blank line. Each entity consists on a set of lines describing its fields, the first field being "entity" containing the entity type which it is about. Each line contains one field. Fields are specified with their field name followed by a set of blanks and by the field value. This is described by the following grammar:

```
response :: entity
response :: entity blank-line response

entity      :: entity-head "\n" entity-body
entity-head :: "entity " entity-name
entity-body :: field "\n"
entity-body :: field "\n" entity-body

entity-name :: "dsp" | "category" | "objecttype
field       :: fieldname blank fieldvalue

blank-line :: "\n"
blank      :: "space" | "tab" |
```

4.1.3 An Example

There is a first implementation of the server running at the `map.cern.ch` machine at port 2345. This is a transcript of a sample session. If you type:

```
telnet map.cern.ch 2345
```

You get

```
Trying...
Connected to hpcernix.cern.ch.
Escape character is '^]'.
Welcome to the MAP experimental server v0.1
```

if you now type:

```
GET dsp objecttype ^person$
```

you will get all the dsps which offer repositories of people. Following the example of the metadata in the last section you would get something similar to this:

```
entity dsp
name cern's combined directory
status not classified, object type normalised
attributes firstname, surname, email, phone, beep, division
location http://consult.cern.ch/xwho
objecttype person
category white_pages.science.physics

entity dsp
name us switboard white pages service
status not classified, object type not normalised
attributes firstname, lastname, city, state
location http://www.switchboard.com/bin/cgiqa.dll?mg=
objecttype person
category white_pages.countries.united_states
```

4.2 A Client Model

Users requiring a piece of data should be able to use the metadata to find locations (DSPs) where such data is likely to be stored. Users should have a way to navigate through the metadata. Thus, we need a *Client Interaction Model* which should access the MAP server and offer the user a significative picture of the metadata.

We have implemented a sample client application which offers two interaction models. It is developed using Java (which was chosed for its portability advantages) and can be reached at `http://wwwmap.cern.ch/client`. It connects to our experimental map client running at the `map.cern.ch` machine at port 2345.

It interactively makes requests to the MAP server as you browse or search the information by:

- Navigating through the hierarchy of categories existing in the server. You can see what DSPs are in each category and eventually jump to it by pressing a button. It is an example of using the GET request.
- Searching any regular expression through the entities existing in the server. You can see what entities match the search and eventually jump to DSPs. It is an example of the SEARCH request.

Our goal with these interaction models is to illustrate how rich a client can be with a very simple MAP specification.

5 Future Work

MAP is in its startup phase and there are some issues which must be addressed in order to follow its evolution.

- There is a need to stablish a metadata language. Much of the usefulness of the model will be given by how well can we maintain and organise an entire taxonomy. The lines of work in this aspect are being based on *Terminological Knowledge Representation Systems* and *Descriptive Logics*. They may give the theoretical background to a (possibly) simplified implementation of a metadata language.
- There is a need to find algorithms which determine which DSPs may be offering the same kind of information. *Machine learning*, *knowledge discovery*, *data mining* and *clustering techniques* are the lines of work which we are following in order to provide a set of suited approaches. However, there is no need to have a standard or universal algorithm. Each MAN can run their own one. The only requisite is that it calls up the DSPs which may have the same things and then they agree on what entities to create or modify (agreeing on field names, categorisation, etc.).

Also from the user interaction model point of view there are some aspects to address. Although decisions in this point may not have entirely a technical basis, the basic ideas are:

- Users should navigate somehow through the metadata to find DSPs.
- Users can use the specification of DSPs in order to query them in their own terms. This is, a generic enough client would be able to query any DSP just by reading their specification from the metadata.

Thus, our current work is first aimed at providing a metadata language and a user interaction model (which makes the model feasible at least from a given locality, say one laboratory) and then at organising metadata coming from different sources (this is, providing algorithms as explained above).

6 Conclusions

Although the model presented here is initially intended to Directory Services Providers, nothing prevents other specialised information servers to benefit from it. For instance, WWW servers can add their own entities as long as the collection of documents they provide can fit within a category, be globally described with few words and the site is searchable. From our point of view, the most important idea introduced by the model is that of **collaboration** and **cooperation** to build up a common terminology and taxonomy from **bottom up** without imposing a global structure from **top down** and avoiding providers from major reestructuration in their sites.

You can always check the current status of the project in [7] where you can also find information about the **client interaction model** (which provides users and client applications with the appropiate tools to navigate through the metadata) and the initial repository of metadata is maintained so that DSPs willing to join can find some criteria on how to describe their site. In simple words, the model can be seen as a motivation for users and providers to organise their visions of themselves and a round table where groups of people offering the same kind of services know each other, sit and agree on a common language.

References

[1] R.J. Barchman and J.G Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, pages 171–216, 1985.

[2] William Frakes and Ricardo Baeza-Yates. *Information Retrieval. Data Structures and Algorithms.* Prentice Hall, New Jersey, 1992.

[3] R.V. Guha. Meta content framework. Research report, Apple Computer, 1997.

[4] Darren Hardy. Resource description messages (rdm). Technical report, Netscape Corporation, 1996.

[5] Darren R. Hardy, Michel F. Schwartz, and Duane Wessels. Harvest v1.4 user's manual: Effective use of internet information. Technical report cu-cs-743-94, University of Colorado at Boulder, 1994.

[6] Phil Lapsley and Brian Kantor. Network news tranfer protocol. Rfc 977, U.C. Berkeley and U.C. San Diego, 1986.

[7] M. Marquina, R. Ramos, and A. Taddei. The metadata advertising protocol. http://wwwmap.cern.ch, CERN, 1997.

[8] Eduardo Mena, Vipul Kashyap, Arantza Illarramendi, and Amit Sheth. Managing multiple information sources through ontologies: Relatioship between vocabulary hetereogeneity and loss of information. Technical report, Facultad de Informatica, Universidad del Pais Vasco and LSDIS Lab, University of Georgia, 1995.

[9] Mark A. Sheldon, Andrzej Duda, Ron Weiss, James W. O'Toole, and David K Gifford. Content routing for distributed information servers. Research report, Programing Systems Research Group, MIT and Bull-IMAG Systemes, 1994.