ALEPHWEB: A SEARCH ENGINE BASED ON THE FEDERATED STRUCTURE

Gerard RODRIGUEZ, Leandro NAVARRO
gerard @ ac.upc.es, leandro @ac.upc.es
Universitat Politècnica de Catalunya UPC Barcelona Spain

1. INTRODUCTION

The volume of the information that navigates on internet is quickly growing up and finding a specific information is not an easy work. There are many information providers or web sites. Therefore, nobody is able to keep a list of interesting points. To solve this problem many search engines have been appearing. Anybody may connect to a search engine and find out the required link. The most common search engines, such as Lycos [December 94], are based on the quantity concept. It means that these search engines try to keep as much information as possible to keep track of the maximum number of information providers. However, sometimes it is not enough, because by applying only the quantity policy the final user can be overloaded by a large list with many irrelevant links.

On the other hand, systems such as Harvest [Bowman 91] and Discover [Sheldon 95] apply the quality policy. It means not to collect any kind of link, but to select the responses according to the requirements of users. But, when you try to find a specific information you get a "no-answers" response. So, at the end you come back to Lycos and at least you get some, maybe irrelevant, answers.

Our proposal is to combine these two strategies, quantity and quality. We think the solution is not to oppose quality against quantity, because without quantity the quality is just a "no-answers" response, and without quality you get a large list of irrelevant information which might oppress the final user and become ineffective.

In order to achieve our purposes we have built a new search engine based on the *federated structure* [Marques 93] [Petrie 92], which we have named AlephWeb. In a federated environment several entities decide to cooperate without loosing their independence. Each entity is represented by on AlephWeb server and the whole community of AlephWeb servers acts like a single search engine. The difference is that now the information (quantity), is distributed, and each information provider controls how its own information is indexed (quality), since who owns the information will be who better announces it. Each user's query is solved forwarding the query to the AlephWeb servers which have higher likelihood to better solve the request. In other words, the request is not answered against all the servers, but it is answered with who owns the information (query routing).

On the other hand, to navigate inside the federated structure we used a dynamic query routing strategy [Sheldon 94] based on a short description of what each information provider has got. These descriptions are stored in dynamic tables of partners and the user's query is guided according to the information that these tables have. Otherwise, these tables are not static and their contents are updated to reflect the real content of the information providers. Any change in the federated environment is reflected in these dynamic tables to better guide users'queries.

The rest of this paper will discuss more aspects about AlephWeb. The first chapter is a review of some search engines that you may find on the Web, giving you an idea of the concepts they are based on. The second chapter gives you more details about the main concepts of our design. And finally, the third chapter compares the current search engines against our concepts to give you an overview of our contribution.

2. RELATED WORK, A REVIEW OF THE CURRENT SEARCH ENGINES

There are three types of search engines : web-robots, subject specific directories and content routing based systems. The most popular web-robot systems, we have found, are the following : WebCrawler [Pinkerton 94], WWW Worm, Aliweb [Koster 94], RbseSpider [Eichmann 94], The Jump Station, Nikos, Lycos. These web-robot systems are based on the concept of a single and centralised database where all the information is stored. This information is gathered looking for web servers with robots, or by the direct submission of URLs from users [Connexions 95]. Once a set of URLs are gathered, the system builds an index to perform queries with the information stored. Some of these systems have several databases but in the same site. For instance, the WWW Worm has four databases to search by URL, original or citation, and to search by document, original or citation, but it keeps a centralised structure.

Every web-robot based system has its own index tool, its own query form and available choices, but all of them almost provide the same functionality, although each one has some remarkable features. For instance, WebCrawler checks the keywords of the user's query against as stop list to see if any keyword is too common. If the keyword is too common, this keyword is not applied because it might provoke a large list of irrelevant URLs.

Aliweb seems to be the most cooperative web-robot in the sense that users may send their own description of the information they wish to announce. It has a mechanism to periodically and automatically update the descriptions of the URLs it keeps to build indexes. Users can write descriptions of their services in a standard format [Deutsch 95] into a file on their Web server. Afterwards, Aliweb regularly retrieves all these files, and combines them into just one searchable database. Furthermore, this system offers the possibility of restricting the query to a certain domain, such the United Kingdom domain. Therefore, it provides some mechanisms to better select the environment where the query will be applied. This might be considered a simple query routing mechanism.

There is another system that it is not exactly a web-robot system, it is the Simon Index. The main difference is that it uses the concept of subject space to group resources. A subject space is a set of names where each name represents a resource. Moreover, a resource into a subject space can be a link to another subject space with more resources or other subject spaces. With the concept of subject space, users can build a graph of environments with several rooms, and they can group resources into subject spaces according to their type or the organisational context of the user. In order words, the Simon prototype provides a mechanism for users to submit their URL, such as Aliweb, but it also offers the concept of subject spaces to group submissions of users in contexts. However, it has not any mechanism to guide a forcing user navigating inside a subject space set.

The second group of search engine is the subject specific directory such as Planet Earth. It offers a set of subject specific web pages with links to many web sites related to the subject. It groups the information into thermatic rooms according to the type of document each URL offers.The main problem with this type of search engine is that the information is maintained manually so it becomes really difficult to keep up-to-date.

Finally, the third group is the content routing based systems. In this group we have clustered systems such as Harvest and Discover. These systems are based on two concepts: not centralised structure and query routing. AlephWeb can be classified in this group.

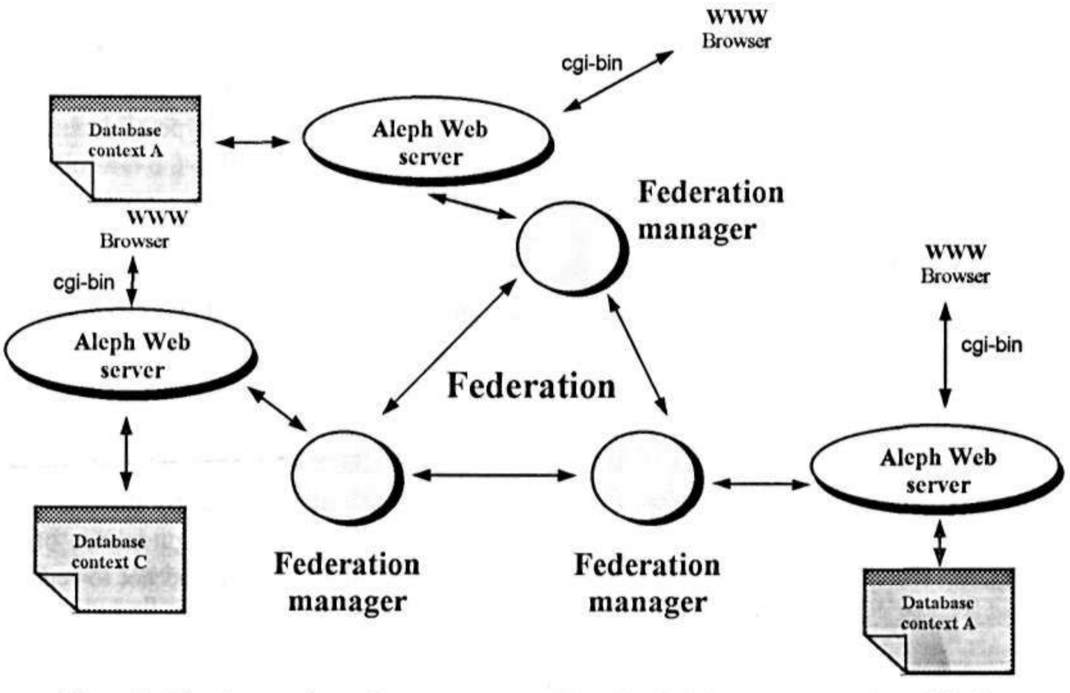
An important feature that they have in common is their structure. They do not use a centralised structure neither do they have a unique database where the information is stored. Discover and Harvest have a distributed structure but with slightly differences. Discover has a hierarchical structure meanwhile Harvest follows the guidelines of a federated structure but with a central authority to enhance the behavior and not to repeat tasks [Hardy 95].

Harvest has four types of agents. These are the gatherer that builds indexes and creates object summaries (SOIF), the broker that provides a network accessible database formed

by collecting SOIFs from gatherers and other brokers, the replication manager, and the object cache that makes local copies of the most popular objects. On the other hand, Discover has two types of agents. They are the content routers, to lead the query, and the information providers, or WAIS servers [Kahle 91], where the documents are stored.
Furthermore, these systems apply the query routing concept, although each one differently implements it. The query routing concept [Sheldon 94] is the ability to lead the user's query through all the servers of information, via content routers in the Discover system and via brokers in the Harvest system. This guidance is carried out according to a description of every server and the system tries to select the servers that will better match the user's requirements.

Every server description contains information about which kind of documents or resources every server keeps. Harvest has called this description as SOIF (Structured Object Interchange Format). Harvest uses the SOIFs to select the set of servers to better address the query but Harvest does not support interaction with multiple servers, neither does it merge the results. However, it supports the ability to select a broker to perform a query instead of automatically forwarding the request. Meanwhile, Discover has named the same concept as content labels and they are a bit more complete than in the case of Harvest. Moreover, Discover supports query routing in the execution time of the query and it is able to merge the result coming from several content routers and information providers.

3. THE ALEPHWEB COMPUTATIONAL MODEL :



Figue 1 : This figure shows the components of the AlephWeb computational model: the AlephWeb servers, the federation managers and the databases, and their relationship.

The AlephWeb computational model is organized by contexts or environments. Each environment represents an organizational context, or a physical environment, or a thematic area. The information of each environment is stored in its own database, and the elements of the AlephWeb architecture provide the affordances to make it available to the whole community.

The AlephWeb architecture in each environment has three main components : the AlephWeb servers, the federation managers and the databases. These elements can be grouped according to their functionality in search oriented objects and federation oriented objects. The AlephWeb servers and the databases form the search oriented objects. These objects keep track of all the links of each information provider in their environment, and perform the search in their local database. Meanwhile, the federation oriented objects, which are the federation managers, interconnect several AlephWeb servers from different environments. Moreover, the federation managers carry out the required tasks to maintain the federated structure :

- interconnect différent AlephWeb servers to solve a user's request.
- guide the query to the most valuable set of AlphWeb servers (query routing), to discard the irrelevant sources of information (quality policy).
- update the information used to guide the query according to the changes in the environment.
- update the query routing data whenever the membership of the federated structure is modified.

To summarise, joining the functionality of the search oriented objects and the federation oriented objects, the whole community of AlephWeb servers provide a quality-quantity search engine service.

3.1. THE FEDERATED STRUCTURE AND QUERY ROUTING

AlephWeb has been designed with large environments in mind. In this sort of environments scalability and heterogeneity are important issues. By scalability we mean that the system must be able to grow without affecting the current state of the AlephWeb servers already working. On the other hand, heterogeneity is closely related to the diversity of large social and computational settings that rule on every entity. We mean that every information provider, it is a Web server installed, is under the control of a set of people with their specific point of view of the organisational setting. This diversity of viewpoints might decrease the ability to cooperate or share documents among several entities. [Schmidt 93].

In order to address the previous drawbacks and achieve a real and easy distribution of the information, we have taken as schema the federated structure. In this structure distinct entities from different environments resolve to cooperate without decreasing their independence. A central authority does not exist. Communications take form between anyone that is inside the federation without the cooperation of a central entity. Thus, how entities are completely independent and how central entity does not exist, then boundary troubles provoked by the dynamism of the structure will increase. The element that deals with this dynamism is the federation manager.

The federation manager offers the whole federation environment to each entity. We might see the federation manager as the door to the outside space. If an entity needs to connect to the outside, it means with other AlephWeb serves, it must cooperate with the federation manager. It will do all the required tasks to make an efficient use of the federation, such as to choose the best foreigner entities to address the user's query, merge the results coming from the federation, maintain the list of partners, and it should hide the complex structure of the federation to the entity as much as possible.

The federation manager uses a short description of the content of other AlephWeb servers to lead the query to the most worthy set of them. We have named this functionality as dynamic *query routing*. Whenever the federation manager has to choose a set of partners to forward the user's query, it looks in these descriptions and it decides which are the most relevant. The query is not solve against all the information providers, but the query is addressed to the most valuable of them, so the result will be restricted to the information that best matches the user requirements (quality policy). These short descriptions are represented with IAFA templates [Deutsch 95], and they are stored in every federation manager into the *dynamic tables*.

3.2. THE FEDERATION MANAGER

The federation manager is the core of the federated structure and it carries out the required tasks to maintain the federated structure. The federation manager has to provide functions for:

- choosing which entities will cooperate to serve a request. Each request is just relevant to a group of entities inside the whole federation, and each request can not be served communicating with all the federated entities because it would break the quality policy. So, the federation manager should address the request to the group of entities which are the most relevant according to the content of the request. *(Partner identification).*
- dealing with changes in membership : either new entities joining or leaving the federation *(registration functionality).*

3.2.1. PARTNER INDENTIFICATION

Partner identification refers to the mechanisms to choose the best set of partners to serve a request according to its content [Duda 94]. It is required to select a set of partners because there can be a lot of entities (Federation managers acting as their contact point) in a federated environment. Thus, a user's request cannot be addressed contacting with all the existing entities. This would provoke an excessive number of interactions and irrelevant information. Therefore, a mechanism to guide the process of looking for these potential partners is required. In our computational schema these mechanisms are *dynamic tables* and *federation policies*. :

Dynamic tables hold information about each potential partner (AlephWeb server). This information is compared to the requirements of a request to select which will be involved in carrying out it. Each federation manager keeps its own dynamic table and for each entry it has the following information :

l FederationQmanagerQid : a unique identifier of the entity in the federated environment or a contact point (i.e. its federation manager).

l ContentQrouting : which requests are more relevant to the entity, which kind of information each AlephWeb server provides.

l Address : information to establish a session with the foreign federation manager. It means machine, socket number, protocol, etc. (e.g. a URL).

l Rate : The *rate* indicates the quality of the cooperation that each federation manager offers. This information reflects the history of previous cooperations and changes accordingly. This rate will be higher for federation managers that have successfully served previous requests. This is useful to maintain an updated view of the federation based on dynamic information.

Federation policies deal with how to use the information about content routing held in dynamic tables. These policies contain rules to manage all interactions with foreign entities. These rules are the following :

- Modification of the rates. The rules that modify the rate item are usually a set of formulas combining history with other values to calculate a new value for the rate item. For instance, one rule could be : # successful Requests / #submitted Requests.
- Maintenance of the dynamic tables. The federation manager needs a set of rules to maintain the dynamic tables alive and useful. It means pruning useless entries (with lower rate than a threshold) and what to do whenever a new entity joins the federated structure.
- Interpretation of the content routing item. For instance, partners might be selected depending on either the cost of the interaction or the quality of the information to be served. In other words, depending on which quality parameters are more relevant for the user's query.

3.2.2. REGISTRATION FUNCTIONALITY

Changes in membership are more difficult to address in a federated environment because there is not central authority,and information about entities is known by several federation managers. We have addressed this drawback using a water-fall protocol to set up the dynamic table of a newcomer and to announce itself to the rest. This protocol takes the following steps:

1. The new entity establishes a communication with one member of the federation (fed-X). It does not matter which one it chooses, since all of them are equivalent.
2. The new entity sends its own information, the address item, the content-routing item, the rate item, the federationQmanagerQid item, to fed-X in order to announce itself.
3. Fed-X returns to the newcomer the content of its own list of known federation managers including itself, which represent known entities inside the federation.
4. The new entity repeats the process (1st to 3rd) for each entity Fed-X has provided until all the federation managers have been checked out.

This water-fall protocol guarantees that the federated structure does not require a central authority and increases openness since no structural restrictions are implicit in the protocol for newcomers. Registration policies may be introduced to impose organisational restriction to limit the membership.

Whenever an entity leaves the federation, it is not required to announce this event to the rest, because the process of calculating the rate item would discard that entity automatically. When a entity leaves the federation, all the requests to it would fail. Therefore, the rate item would decrease and the federation policies will remove that entity from the dynamic tables. Again, a central authority is not required.

4. WHICH ARE OUR CONTRIBUTIONS COMPARED WITH THE CURRENT SEARCH ENGINES ?

Once our concepts and the current search engines have been reviewed, the next step will be to compare the current search engine features against the concepts we propose in this paper. It is done to clarify what we think the weak points of the current search engines are, and to extract the contributions that our prototype might bring. The next table summarises the features of most of the search engines against our concepts.

Search engine	Query routing	Structure	Other Important features
WebCrawler	Not provided	Centralised.	nil
WWW Worm	Not provided	Centralised.	nil
Aliweb	Not provided	Centralised. but it offers the possibilities of mirroring	It can group the resources into few types. It can limit the searches to a certain domain, such as "es".
RBSE Spider	Not provided	Centralised	nil
The SIMON Index	Not provided	Centralised.	It offers to users the ability to group resources in subject spaces depending on their knowledge.
LYCOS	Not provided	Centralised.	It has a rather complete query tool.
Planet Earth	Not provided	Centralised and it is updated manually.	The information is clustered in thematic rooms.
Harvest	It creates and keeps descriptions about the resources, but il cannot merge the results.	Distributed with a central authority. The functionality is split among its agents. It has a static federated structure among the brokers.	Users can customise their brokers via sripts to adapt them to their organisation.
Discover	It provides a query routing tool based on descriptions (content labels). It is able to merge the result and the query is led matically.	Distributed structure with content routers and information providers. The structure of its agents is hierarchical (tree).	It has the ability to group documents which are similar in collections. Provides post response query leading facilities.

The first group of search engines, from WebCrawler to Planet Earth, has the same problem: They use a centralised structure. It means they try to collect all the information in just one site. Therefore, these systems might overload the server since all users must connect with it. Some systems, such as Aliweb or Nikos, can replicate the information, but this solution does not solve the problem of irrelevant information (quality policy).

The solution is to build search engines with the distributed architecture in mind. Some search engines such as Harvest or Discover already use this structure. Our proposal takes a slightly similar concept, it is a distributed structure as a backbone of the system, but we propose a step ahead. We propose a structure, the federated structure, that will be able to increase the freedom of each environment. Each environment has its own AlephWeb server and all the servers cooperate to build a global net of search engines. The cooperation of servers is freely carried out applying the federation policies.

Regarding Harvest, we might conclude that it uses a static federated structure instead of the dynamic federated structure we propose. It is static because the brokers have the ability to gather information from other brokers, but each broker must replicate the information that it has collected from partners instead of dynamically share the data.

On the other hand, the structure of the agents of Discover is hierarchical and it clearly limits the capability to adapt them to the changes of the environment. This hierarchical structure, for instance, might provoke problems such as how to decide where a new agent has to be placed in. With our federated structure whenever a new agent comes in, it only has to make it public to the others and it will not be placed under the responsability of another agent.

Another problem of most of the search engines is that just work according to a quantity policy, and they do not use a quality policy. It means that they try to gather the maximum number of object references regarding all the subjects, and afterwards they return all these object references regarding all the subjects, and afterwards they return all these object references to the user in a large list. We think this quantity policy does not suit to the current and future state of the global information community. Users might lose inside those large lists of information without any help to extract the most relevant references. Therefore, we propose to combine the quality policy with the quantity policy. The quality policy would try not to forward the user's query to the useless servers of information, which would generate irrelevant references.

There are some systems which already take quality policy into account. These are Harvest, that provides query routing, and Discover, which provides query routing and a post-response query refinement.However, Harvest limits the ability of its query routing mechanism because it does not seem to support interaction with multiple servers. On the other hand, we think the hierarchical structure of Discover clearly limits its features.

5. CONCLUSIONS

It this paper we have presented a new computational model to implement a global search engine. In this model we propose to combine the quantity policy (i.e. gather as much information as possible) with the quality policy (i.e.discard the irrelevant responses) to provide a better search engine service. The quality policy is provided by means of splitting the search engine database and distribute its parts in several environments. To distribute the parts we have chosen the federated structure since it does not limit the independence of the entities and does not limit the dynamism of the structure. Furthermore, the dynamic query routing policy is required to guide the user's request to the most worthy set of parts, which are managed by AlephWeb servers. Moreover, a relevant feature of our dynamic query routing policy is its ability to re-configure itself according to changes in the federation.

By the time this paper was written, there were two AlephWeb servers already working. The former is a link to a mirror of the Aliweb database, and the latter is the AlephWeb server of the Catalan Chapter of Internet Society (1). Future issues are concerned with feedback studies of its use, and studying the suitability of applying the interception functionality to provide affordances for working across boundaries [Marques 93].

Références Bibliographiques

[December 94] John December. *New Spiders Roam the Web* Computer-Mediated Communication Magazine. Volume 1, Number 5, Septembre 1994.

[Bowman 94] C.M. Bowman, P.B. Danzing, D.R. Hardy, U. Manber, and M.F. Schwartz. *The harest information discovery and access system* In Proc. of the Second International World Wide Web Conference, Chicago, Illinois, October 1994.
[Sheldon 95] Mark A. Sheldon, Andrzej Duda. Ron Weiss, David k. Gifford. *Discover : a Resource Discovery System based on Content Routing.* Computer Networks and ISDN systems. Volume 27, Number 6, April 1995. ISSN 0169-7552.

[Marques 93] J.M. Marques, L. Navarro, M. Sarmiento. *From small to large scale.* Comic Deliverable D.1.1. ISBN 0-901-800-55-4.

[Sheldon 94] M.A. Sheldon, ed altres. *A content routing system for distributed information servers.* In Proc. Fourth International Conference on Extending Database Technology, March 1994.

[Pinkerton 94] B. Pinkerton. *Finding what people want : Experiences with the WebCrawler.* In Proc. of the First International Conference on the World Wide Web, Geneva, Switzerland, May 1994.

[Koster 94] M. Koster. *Aliweb archie-like indexing in the Web.* In Proc. of the Fisrt International Conference on the World Wide Web, Geneva, Switzerland, May 1994.

[Eichmann 94] D. Eichmann. *The RBSE Spider, balancing effective search against web load.* In Proc. of the Fisrt International Conference on the World Wide Web, Geneva, Switzerland, May 1994.

[Connexions 95] Martijn Koster, Nexor. *Robots in the Web : threat or treat ?* Connexions Magazine, Volume 9, Number 4, April 1995.

[Deutsch 95] P. Deutsch, A Emtage, Bunyip, M. Kester, Nexor, M. Stumpf. *Publishing information on the Internet with Anonymous FTP.* Internet Draft. March 1995. ftp://nic.merit.edu/documents/internet-drafts/draft-ietf-iir-publishing-2.txt.
[Hardy 95] Darren R. Hardy, Michael F. Schwartz, Duane Wessels. Harvest. *Effective Use of Internet Information.* Technical Report CU-CS-743-94, April 1995, Department of Computer Science, University of Colorado, Boulder, Colorado 80309-0430.

[Kahle 91] B. Kahle and A. Medlar. *An information system for corporate users* : Wide Area Information Servers. Technical Report TMC-199, Thinking Machines, Inc. April 1991, Version 3.

[Shmidt 93] Eds. K. Schmidt, L. Bannon. *Issues of Supporting Organizational Context in CSCW Systems.* 1993. ISBN 0-901800-28-27.

[Duda 94] A. Duda, M.A. Sheldon. *Content routing in networks of WAIS servers.* International Conference on Distributed Computing Systems Proceedings, IEEE, June 1994.

[Petrie 92] C.J. Petrie, *Enterprise Integration Modelling.* Proceedings of the First Intenational Conference, MIT Press, 1992.

(1) http:/aleph.ac.upc.es:8000/alephweb.html

7. Appendix

Why should we change the current search engines procedure ? The following message posted to the comp.sys.isis newsgroup by John Fereira on the 5th of July of 1995 will give you some important reasons...

"In article <3teo42$jlp@news.fsu.edu> Mark A Brenneman writes :
>
>I was juste wondering what isis provider even is ?
>Mark

There seems to be some confusion about what this newsgroup is about. As Ken said, it is for discussing the products of Isis Distributed Systems.
I'm not suprised that people get confused however. If on does a URL search using Lycos (a popular WWW search engine) specifying only the keyword "isis" you will find a variety of hits. The first one on the list describes ISIS as "the worlds brightest spallation neutron source".

It seems to be a facility in England for physics studies.
Next on the list is a page dedicated to ISIS/Draw, a PC drawing package out of a site in Germany. In fact, the web page is written in German.

The third entry is a page to the "Isis" internet provider, specifically it's staff. The 12th through 14th entries are also from "isisnet".

Fourth on the list is a white paper from the UK that makes a few references to ISIS, the neutron source. The fifth through the ninth all make references to ISIS in the UK as well.

Fifteenth on the list, and the last entry in the default number of hits is something called ISIS International. It was impossible to tell what this actually is.

The next entry that isn't related to any of the previous hits is something at NASA that seems to be an image processing system but the link is unavailable. That's hit #24.

Finally, the 27th hit in the search points to the Isis/Horus page at Cornell, the *real* Isis.

...followed by the 28th entry, the Information System and Insect Studies (ISIS) laboratory.

It's no surprise people are confused about the purpose of this newsgroup".