Ingrid: A Self-Configuring Information Navigation Infrastructure

Paul Francis *
Takashi Kambayashi
Shin-ya Sato
Susumu Shimizu

Abstract:

This paper presents Ingrid, an architecture for a fully distributed, fully self-configuring information navigation infrastructure that is designed to scale to global proportions. Unlike current designs, Ingrid is not a hierarchy of large index servers. Rather, links are automatically placed between individual resources based on their topic similarity in such a way that clusters of term combinations are formed. The resulting topology can potentially be searched and browsed by a robot efficiently. This paper describes the fundamentals of Ingrid--the topology design and the algorithms for creating and searching the topology. It discusses the scaling characteristics of Ingrid, and gives the scaling results of a limited experiment.

Keywords:

information, retrieve, navigate, browse, distributed, self-configure, infrastructure, Ingrid

Introduction

Current *browsing* on the Web consists of the traversal of 1) hypertext-style links between explicitly related documents, and 2) indexes and meta-indexes, which are usually structured according to organization, sometimes by topic, and are in any event almost always incomplete in their coverage. What is missing is general and complete *topic-level* browsing--that is, where all resources are linked according to topic.

Current *searching* on the Web consists of querying single-database search engines. While this method is effective, single-database search engines are necessarily (and usually intentionally) incomplete in their coverage. This is likely to become more rather than less true as the Internet grows. What is missing is complete internet-wide searching.

This paper describes Ingrid--a distributed, scalable, self-configuring information navigation infrastructure. The goal of Ingrid is to provide these two missing functions. In a nutshell, Ingrid works by automatically creating links between resources that are topically related. Resources are searched by "routing" (robot-style) from topic to topic until the appropriate resources are found. Browsing is accomplished in a similar way, except that movement from topic to topic is directed by the user. Browsing is truly topic to topic (versus document to document) because the browsing robot can efficiently traverse multiple links and produce a topic summary for the user.

The following describes one of many typical usage scenarios for Ingrid. An Ingrid "resource publishing" background process is running in conjunction with a mail archive. When new mail arrives, the Ingrid publisher automatically generates a profile of the mail (author, title, high-weight terms), and sends the profile to the Ingrid forward information server associated with the mail archive. The Ingrid forward information server "inserts" the profile into the Ingrid infrastructure by searching for and attaching links to similar profiles.

Later, a user wishes to find resources related to the topic of the previously inserted mail. Using an Ingrid browser, the user inputs keywords related to the topic. The Ingrid browser launches a robot that, by querying

various forward information servers, traverses links of the Ingrid infrastructure in search of resource profiles with matching terms. Because of the organization of the links, the robot is able to efficiently find better and better matches. The Ingrid browser presents the best matching resource profiles to the user, along with a set of related terms. The user then expands and focuses his/her search using some of the related terms.

In the following sections, Ingrid is contrasted with current Web practices. These sections serve both to review existing techniques (admittedly incompletely) and to give an overview of Ingrid.

Ingrid as Compared to HTML

Ingrid can be viewed as a web space parallel to (and complimentary to) the web space that exists by virtue of HTML links (see Figure 1). The two web spaces are similar in that they are both composed of (URL-type) links between resources. The similarity, however, stops there.

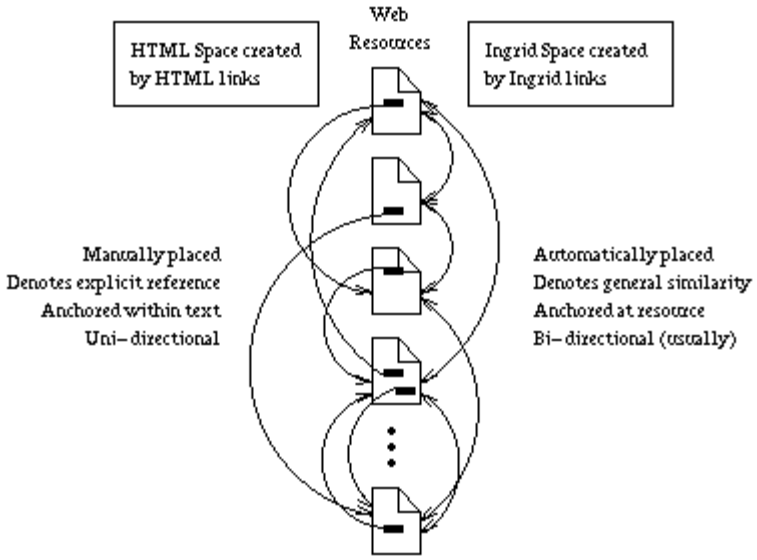

Figure 1: HTML Space Versus Ingrid Space

HTML links are manually placed (by and large) and denote an explicit reference from one resource to another. Ingrid links are automatically placed and denote a general topical similarity between two resources. HTML links individually have a strong local meaning, but collectively (index and meta-index documents notwithstanding) do not contribute well to the global organization of information. Ingrid links, on the other hand, have a somewhat weak local meaning, but collectively create a meaningful global organization of information.

Browsing in HTML space can be considered as having two forms. One is browsing through index documents, and the other browsing among leaf documents (see Figure 2). Navigation among leaves takes place at the "micro" level, and tends to be from individual document to individual document. Navigation through indexes is at the macro level, and seems to be most often oriented towards organizations. Sometimes it is oriented towards topic area, but to the extent that this is true, it is usually somewhat incomplete.

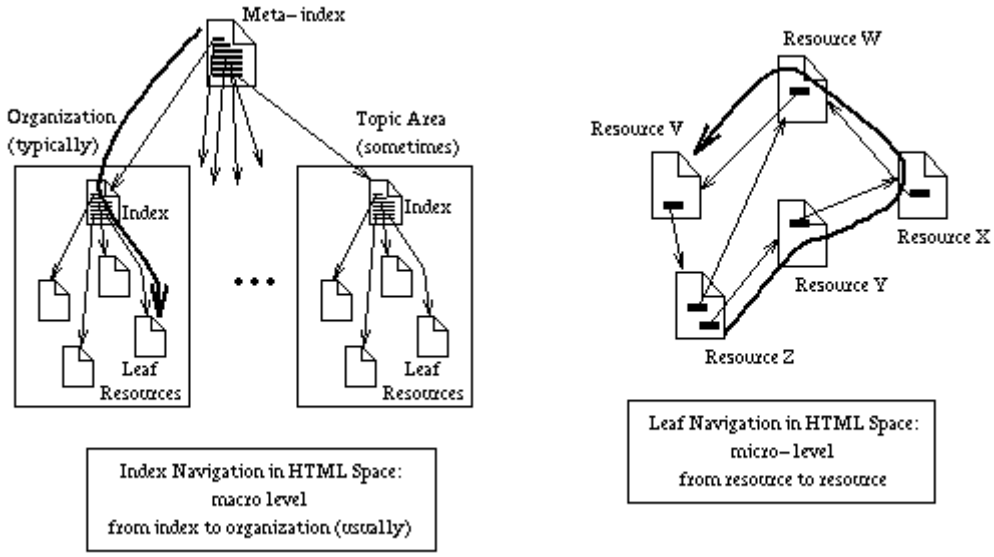

Figure 2: Navigation with HTML

Navigation in Ingrid space is envisioned to take place almost entirely at the "macro" level, from topic area to topic area (where a topic area is composed of a group of documents, see Figure 3). In addition, the topic areas are interconnected. Thus, one can browse among related topic areas. Because of these differences in the two web (or information) spaces, they are highly complimentary.

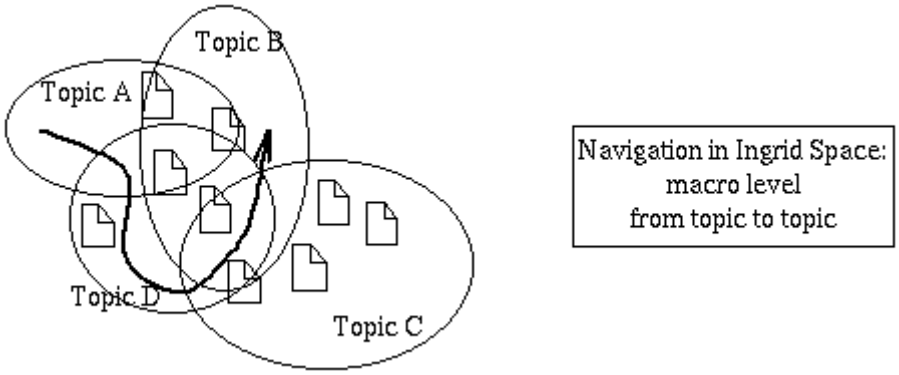

Figure 3: Topic-level Navigation

Ingrid as Compared to Single-Database Search Engines

Currently, the most effective way of searching the web is through one of the growing number of available single-database search engines. These search engines can be categorized into two kinds:

1) Those that attempt to index the entire web
2) Those that index a selected portion of the web

WAIS and Harvest are just two of many examples of the latter category. As illustrated in Figure 4, the goal of Ingrid is to allow searching of the whole web, but necessarily with less depth than can be achieved with a single-database search engine. Thus, the functionality of Ingrid is complementary with that of limited-coverage single-search engines.

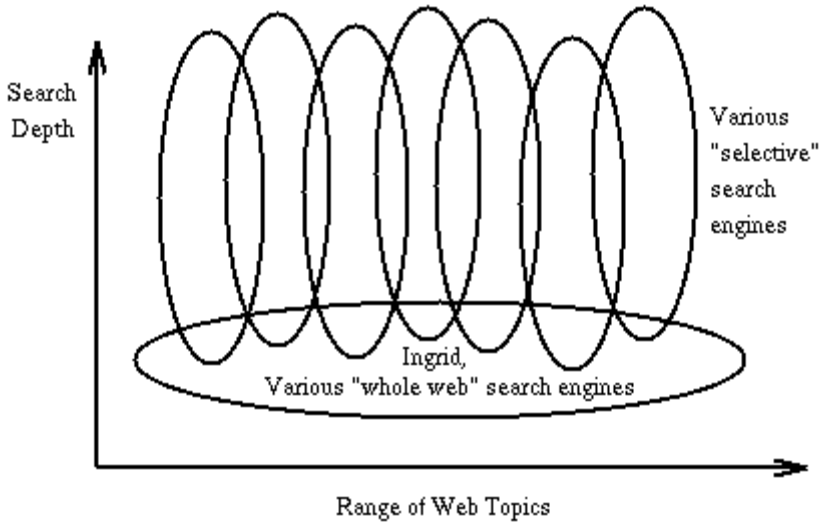

Figure 4: Comparison of Search Techniques

Two examples of search engines that attempt to index all web resources are Lycos and WWWW. As shown in Figure 4, these whole-web search engines and Ingrid are attempting to do roughly the same job, and are therefore essentially competing technologies. Thus, we wish to briefly justify the work of Ingrid in light of whole-web search engines.

The primary justification for work on Ingrid is scaling. It is not clear that the single-database approach will be able to keep pace with the growth of the web. So far, Lycos has apparently been able to keep pace, as it seems to consistently be indexing approximately 75% of the estimated 4 million (as of July 1995) total URLs. On one hand, 4 million documents barely scratches the surface of the total number of documents that can be expected to be available over the web in the future. On the other hand, Lycos has probably barely scratched the surface of what a "single" search-engine can do, given massive parallelism, huge memory farms, and the like.

In short, the ability of a single-database search engine to be able the index the entire web, and the associated costs, are unknown. Likewise, the ability of Ingrid to search the entire web is also unknown. Thus, it seems prudent to experiment with both methods.

Ingrid as Compared to Other Distributed Searching Techniques

There are many different (actual or proposed) distributed searching techniques. These range in functionality and complexity from strict traversal of a naming tree (DNS [3]), to automatic dispersion of terms over a mesh of forward information servers (Centroids). In this section, we limit ourselves to examples of distributed searching techniques that are administratively distributed and fully automatic. In particular, we consider Centroids and Fish-Search.

Table 1 compares Centroids and Fish-search (along with Ingrid and single-database engine search techniques). Centroids proposes building a super-structure of index servers above the existing search engines. These meta-search engines are able to forward queries to the appropriate leaf-level search engines and to other meta-search engines. (Note that the primary purpose of Ingrid is not to find search databases--Ingrid directly searches for and finds individual resources. To the extent that the contents of a search database can be summarized in a single document, however, Ingrid could be used to find search databases.)

Table 1: Characteristics of Various Search Techniques

	Required New Infrastructure	Pre-search Activity	Search-Time Activity
Single-Database Search Engines (Lycos, WWWW, etc.)	None	Gather Web Resources (Robot-style), Build (single-database) Index	Search Single-Database Search Engine
Mesh of Super-Indexes (Centroids)	Mesh of Index Servers	Summarize Databases, Build Index-Server Indexes	Route through Index Servers, Search Search Engines
Mesh of HTML links (Fish-search)	None	None	Gather Web Resources (Robot-style), Search Resources
Mesh of Local Indexes (Ingrid)	Topic-based resource links, Global Single-Term Servers	Summarize Resources, Create Links and Global Single-Term Server Entries	Occasionaly search Global Single-Term Server, Route through Ingrid Topology

Fish-search, on the other hand, traverses current HTML links, robot-style, until matching resources are found or until specified limits (time, number of resources traversed) are exceeded. Navigation in Ingrid is similar to Fish-search in that the Ingrid Navigator follows links in search of the desired terms. The primary difference is that the Ingrid Navigator follows links specifically created for the purpose of term-navigation, while Fish-search searches HTML links--a rather more ad hoc process.

Scaling of Fish-Search and Centroids

As with the single-database search engines, a critical issue with the distributed search mechanisms is scaling.

Fish-search, in our opinion, has little hope of scaling well. Because the organization of HTML links is ad hoc, the resources that match any given fish-search query may be far apart in the HTML topology. Thus, Fish-search may require a very large number of link traverses to find all of the matching resources.

The scalability of any super-structure of index servers depends entirely on how the information in the index servers is organized. The idea with Centroids is that a search-engine (or meta-search engine) automatically summarizes its contents by listing once each term that exists in any of its resources. (Note that these terms may include full-text or may be selected high-weight terms only.) These summaries are transmitted to meta-search engines, which may in turn summarize their contents and pass them on.

Search queries are sent to meta-search engines. These engines refer the searcher to those engines that contain all of the terms in the query. These engines are in turn queried, and so on.

In some cases, the referal may be false. For instance, consider a query with terms A, B, and C. Because Centroids lists each term once, and gives no information about how terms relate, a Centroids summary does not distinguish between one resource with terms A, B, and C, and three resources each with only one of the terms. The primary scaling issue, then, is that of how many false referals (and subsequent fruitless queries) occur. (Note that memory size of the meta-search engines is not a scaling issue, precisely because information about how terms relate is suppressed. Thus, memory scales linearly with the size of the global vocabulary, which we believe to be manageable.)

This issue remains open, and we believe can't be answered without significant experience. Ingrid of course has its own scaling issues, and so again we believe that it is appropriate to experiment both with Ingrid and Centroids (and of course any other proposals that appear promising).

Basic Components of Ingrid Infrastructure

As shown in Figure 5, the basic component of the Ingrid Infrastructure is the so-called *Resource Profile*. The Resource Profile is essentially a (text) summary of the resource that it represents (though the resource itself can be anything--an audio/video file, a physical object, or a service). The two required elements of the Resource Profile are:

1) Terms (or rather, *term combinations*) that describe the resource, and
2) A pointer (URL) back to the resource.

Other information may be available, such as resource title, authors, date, size, type, etc.

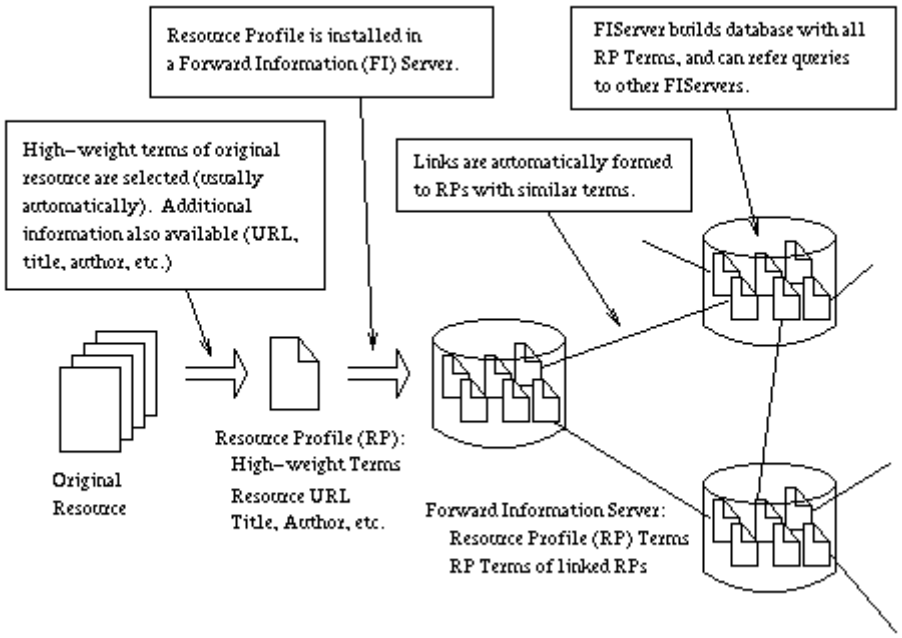

Figure 5: Resource Profile

The term combinations in a given Resource Profile characterize the resource. These are the terms that will match-up against the terms in a search query.

One or more Resource Profiles are installed in a Forward Information Server (*FIServer*). The FIServer adds the term combinations of the Resource Profile to its searchable database. The Resource Profile is also linked with a selected set of similar other Resource Profiles (possibly in other FIServers). Thus, the Resource Profile essentially becomes a node in a mesh network of Resource Profiles. This network is called the *Ingrid Topology*.

The practical effect of the link is that the Resource Profile's FIServer will add the terms of the neighbor Resource Profiles to its searchable database. Thus, the FIServer is able answer queries in two ways:

1) By listing its own matching Resource Profiles, and
2) by referring the querying system to other FIServers that have matching Resource Profiles.

This allows an *Ingrid Navigator* (searcher or browser) to "route" itself through the Ingrid Topology (or, more accurately, through the mesh of FIServers) and find relevant Resource Profiles. In this sense, Ingrid's

FIServer is similar to the meta-search engines of Centroids discussed in the last section. One practical difference, however, is that the FIServer in Ingrid contains only terms for its own Resource Profiles and a very selected set of other (similar) Resource Profiles. Thus, every document server (FTP, gopher, WAIS, HTTP, etc.) can potentially become an FIServer. This can be contrasted with Centroids, where even the lowest-level meta-search engine indexes the entire databases of multiple search engines.

Ingrid will in general have a larger number of smaller FIServers than a scheme like Centroids. This has its advantages and disadvantages. The primary advantage is that we can leverage the enormous aggregate latent computing resources (CPU, memory, bandwidth) of every computer that holds a resource (or, more accurately, holds the Resource Profile, since they need not be on the same computer). In other words, if a computer is capable of storing and transmitting a network-retrievable resource, then for some extra (CPU, memory, bandwidth) cost, it can contribute to the overall navigation infrastructure. (What this extra cost is remains to be seen.)

The major disadvantage of distributing navigation across many small servers is that the overall control of the navigation infrastructure is diffused, thus making it difficult to control, for instance, performance or correctness. On the other hand, if this turns out to be an insurmountable problem, it is always possible to simply store all the Resource Profiles in a relatively small number of machines, thus effectively making those machines large, dedicated meta-search engines.

Finding the First FIServer

As described above, an Ingrid Navigator will "route" from FIServer to FIServer as it searches for the best matching resources. In some cases, however, an FIServer may not initially know of any FIServers that contain desired terms. Especially given the potentially large number of FIServers in Ingrid, we assume that it would be too expensive to randomly query successive FIServers in the hope of finding one with one or more desired terms.

To prevent this from happening, we require the existence of a special type of forward information server called a Global Single-Term Server (GSTServer) (see Figure 6). Each GSTServer contains one entry for every term in all of Ingrid space. Each entry points to a small number (perhaps three, for robustness) of FIServers that contain that term at least once. The GSTServer is only used to "bootstrap" an Ingrid Navigator's search. Once an Ingrid Navigator finds at least one FIServer with a given term, it can be referred to others without having to query the GSTServer again.

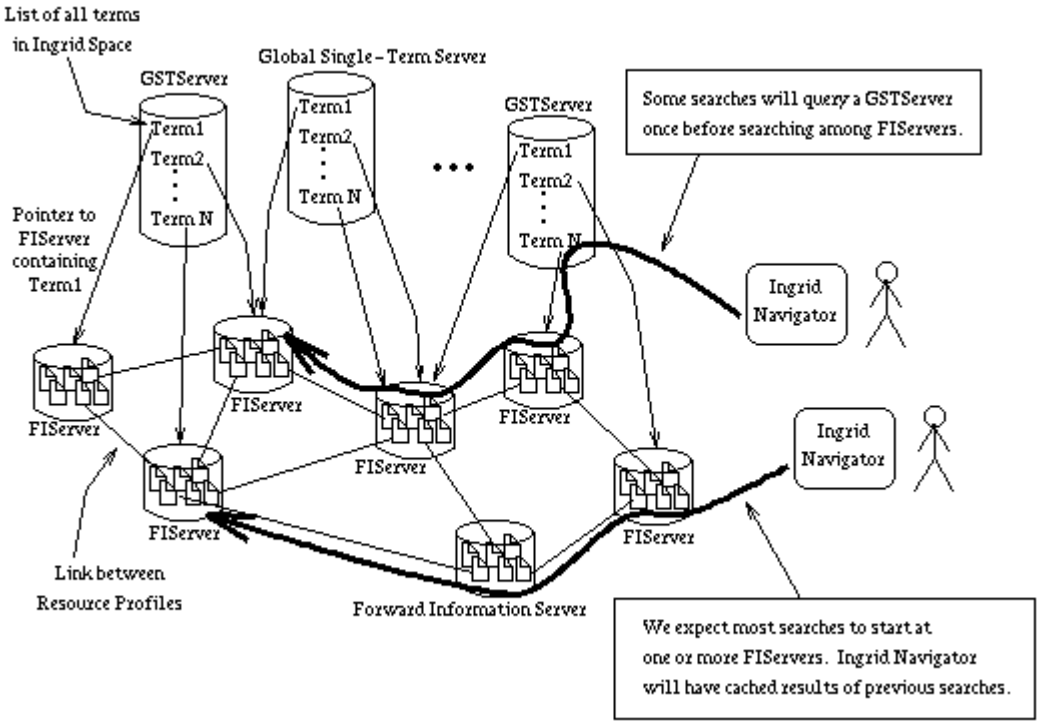

Figure 6: Global Single-Term Servers

We expect the GSTServers to scale in terms of memory because the list size is that of the global vocabulary, which is bounded (at some millions of terms, but bounded none-the-less) and manageable. The GSTServers should scale in terms of number of queries because we expect most searches to already know of relevant FIServers from the start. This is because Ingrid Navigators cache the results of previous searches.

Description of Ingrid

Nothing said in this paper up to now is particularly novel. The functional components described above are little different from what has already been proposed, and are in fact only minor extensions to what already exists. Indeed, the primary reason for the above is simply to set the context for describing what is unique about Ingrid.

Two things make Ingrid unique and potentially feasible:

1) the logical organization of the Ingrid Topology (and how it leads to efficient navigation), and
2) the algorithm for automatically building the Ingrid Topology.

They are described in the following sections.

The Ingrid Topology

The definition of the Ingrid Topology is actually quite simple. Assume a set of Resource Profiles, each with a term combination (set of terms). Each Resource Profile is a node in the Ingrid Topology. Define a *cluster* as a connected sub-topology. That is, there is a path between any two nodes in a cluster that contain only nodes in that cluster. The Ingrid Topology is a mesh topology whereby for every combination of terms, the Resource Profiles that contain those terms are connected such that they form a cluster.

For instance, Figure 7 shows 13 Resource Profiles connected in a mesh topology. Each of the Resource Profiles has one or more of terms A, B, and C. All Resource Profiles that have the same term or set of terms form a cluster. For instance, all Resource Profiles with term C (those numbered 7 through 13) form a connected cluster. Likewise those with both terms A and C (10 through 13) form a cluster, as do those with all three terms (11 through 13).

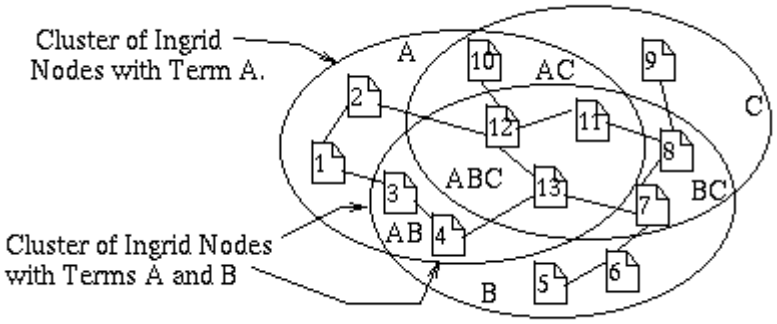

Figure 7: Example Ingrid Topology

Within the limitations of the above definition, we try to keep the Ingrid Topology sparse. That is to say, we strive to minimize the number of links each Resource Profile has. (A fully connected graph--every node connected to every other--strictly speaking satisfies the above definition, but is obviously of little benefit.)

Keeping the topology sparse has two performance benefits. First, the amount of information that must be stored by each FIServer is minimized. Second, the amount of information that must be searched by the FIServer when it is queried by an Ingrid Navigator is minimized.

A possible downside of a sparse topology is that the overall network diameter is bigger. In practice, however, we believe for a number of reasons that this will have little negative effect. One reason is that the node degree is expected to be rich enough that diameter will be small in any event. Another reason is that, in order to know when a search is exhausted, it is necessary to query all FIServers that contain Resource Profiles in the cluster. Thus, the cost of the search is dependent more on the number of FIServers in a cluster, and less on the diameter.

Searching the Ingrid Topology

One of the major open questions about Ingrid is the efficiency of its search (how many queries are required, how long it takes, and so on).

Before discussing searching, it is worth reminding the reader that the search is directed by the Ingrid Navigator, not by the FIServers it queries. That is, the Ingrid Navigator queries FIServers, gets back some forward information about neighboring FIServers, and then decides itself which FIServers to query next. In this sense, searching in Ingrid is similar to current web robots, particularly Fish-search, and different from network-layer routing, where the intermediate nodes themselves manage the routing of the packet.

First some nomenclature. Assume three terms, A, B, and C. It what follows, we denote an FIServer that contains at least one Resource Profile that contains all three terms A, B, and C, as an *ABC FIServer*. We denote the set of all ABC FIServers as an *ABC Cluster*. If an Ingrid Navigator knows of at least one ABC FIServer, then it is said that the Ingrid Navigator is *in* the ABC Cluster. Note that the ABC Cluster is said to be a sub-cluster of the AB Cluster (because all FIServers in the ABC cluster are, by definition, also in the AB Cluster).

The basic idea behind searching the Ingrid Topology is to successively find a sub-cluster of the already-found cluster until either 1) the desired cluster if found, or 2) the search is exhausted. In this sense, searching the Ingrid Topology is hierarchical in nature.

For instance, assume that an Ingrid Navigator is searching for a resource with terms A, B, C, and D. Further, assume that an AB FIServer has already been found. The Ingrid Navigator queries the AB FIServer for any Resource Profiles with the terms A, B, C, or D. The answer will, at a minimum, list the neighbor AB FIServers. (Because of caching, the answer may contain other FIServers as well. This is further discussed later.)

The Ingrid Navigator then queries the neighbor AB FIServers, and so on. Eventually, either all AB FIServers will be queried, or one of the answers will contain an FIServer with additional matching terms (that is, an ABC, ABD, or ABCD FIServer). In the latter case, subsequent queries are limited to the FIServers with the additional matching terms. The search continues in this fashion until either

1) all useful FIServers have been queried,
2) some search limits, such as length of search or maximum number of queries, has been reached, or
3) a fully matching (ABCD) FIServer has been found.

In the third case, all ABCD FIServers can then be queried to find all matching Resource Profiles.

This process is illustrated in Figure 8. This figure shows 5 clusters as 5 circles, for terms A, B, C, D, and E. The various sub-clusters are indicated by the overlap of the circles. Assume that the Ingrid Navigator knows of at least one FIServer for each single term (the solid small circles in the figure). This is always possible using the Global Single-Term Server (GSTServer), though in general the Ingrid Navigator should already have good starting points based on previous searches.

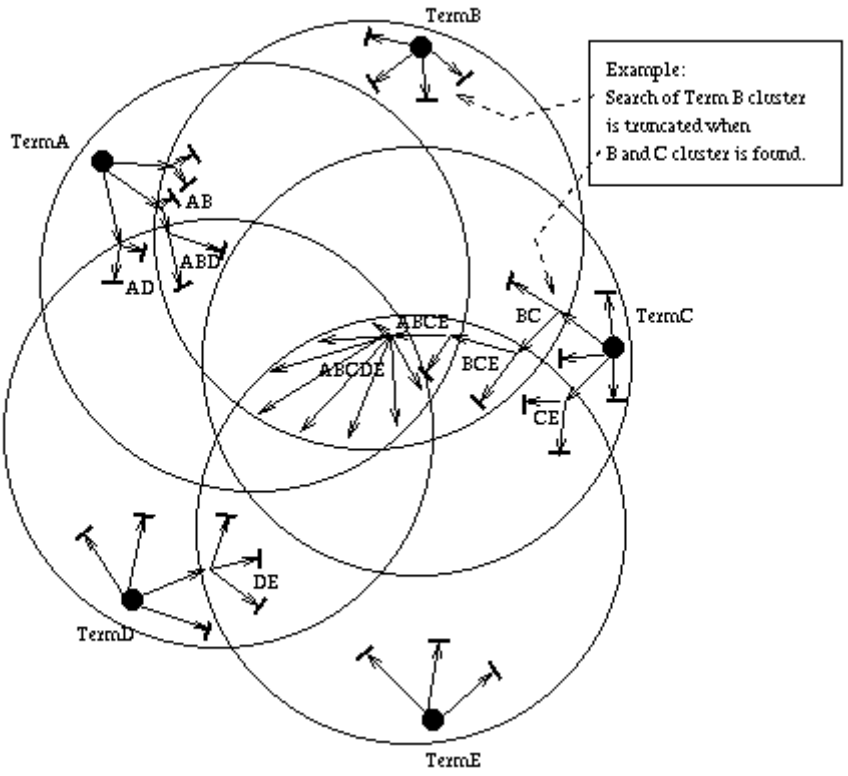

Figure 8: Example of Search Procedure

The Ingrid Navigator explores each of the 1-term clusters until it finds one or more FIServers with two terms. This searching process is denoted by the arrows. In the case of Figure 8, we show the Ingrid Navigator finding AB, AD, BC, CE, and DE FIServers. At this point, any continued searching of 1-term clusters is discontinued (as indicated by the line across the arrow tip). This is because searching a 2-term cluster is, in most cases, more likely to yield good results than a 1-term cluster. (The exception being, for instance, a 1-term cluster with a very rare term versus a 2-term cluster with 2 common terms.)

Figure 8 shows searching branching out within the 2-term clusters until two 3-term clusters are found, ABD and BCE. Any further 2-term cluster searching is halted, and the 3-term clusters are searched. Next we show that a 4-term cluster ABCE is found. This cluster is explored until ABCDE Cluster is found. Finally, all of the FIServers in ABCDE Cluster are queried, thus finding all Resource Profiles with terms A, B, C, D, and E, and therefore all resources for which those five terms are considered high-weight terms.

Taking advantage of cached forward information

While the searching process as described above has the nice property of continuously narrowing the scope of the search, this in and of itself does not guarantee an adequately efficient search. A cluster for a common term (such as "computer") may include hundreds of FIServers and many thousands of Resource Profiles. Searching such a cluster for additional terms is likely to be unacceptably costly.

There are a number of techniques that might be used to increase the efficiency of the search. We are hoping that one in particular, simple caching of forward information, will prove adequate. The caching strategy we describe in what follows is, as of this writing, being implemented in the so-called alpha-version of the Ingrid prototype. (We hope that, as of this reading, it has already been implemented. :-)

Figure 9 shows two kinds of forward information, Persistent Forward Information, and Cached Forward Information. Persistent Forward Information is that already discussed in the context of the Ingrid Topology--it is the forward information that gets installed as a result of inserting a Resource Profile into the Ingrid Topology. This forward information is persistent in that it remains as long as the Resource Profile stays valid (or, normally, as long as the resource itself stays valid).

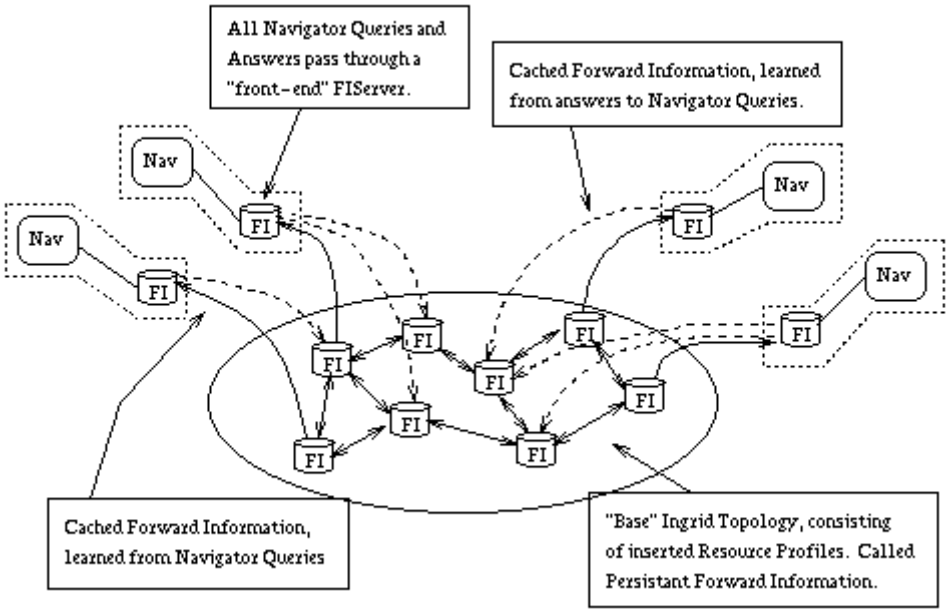

Figure 9: Forward Information Caching Architecture

The Persistent Forward Information can be thought of as forming a stable base topography over which terms can, with high probability, be successfully searched (even if, perhaps, inefficiently).

The Cached Forward Information, on the other hand, is derived from the results of previous searches. Since it is the Ingrid Navigators themselves that learn the results of searches, not the FIServers, one question is that of how to transfer the results of recent searches from the Ingrid Navigators to the FIServers.

Figure 9 shows that each Ingrid Navigator is behind a kind of "front-end" FIServer. All interactions with the Ingrid infrastructure pass through the FIServer. In particular, queries and query answers pass through the FIServer. These FIServers cache the query answers before passing them on to the Ingrid Navigators.

In addition, "regular" FIServers (we use quotes because in practice FIServers can serve both roles) cache received queries. This results in the following behaviour. An Ingrid Navigator X conducts a search for some terms, using FIServer X as a front-end. In the process, a number of FIServers cache the queries from FIServer X. In addition, FIServer X caches the found Resource Profiles.

Later, another Ingrid Navigator Y conducts a search for some of the same terms. In the process of the search, Navigator Y queries one of the FIServers previously queried by Navigator X. The FIServer returns forward information about FIServer X. Navigator Y queries FIServer X, and learns of the Resource Profiles previously found by Navigator X. In other words, Navigator Y takes advantage of the work already carried out by Navigator X.

Note that, alternatively, we could have designed the caching scheme such that front-end FIServers are not required. Rather, after a search is completed, Ingrid Navigators explicitly inform the FIServers they previously queried about the Resource Profiles they found. Thus, all cached forward information refers directly to Resource Profiles (rather than indirectly referring to Resource Profiles by first referring to Navigators).

Without getting into a long discussion about the pros and cons of the two schemes, we point out simply that the presence of front-end FIServers has other advantages, and so is likely to be an architectural component in any event. These advantages include:

1) acting as a proxy forward information cache service on behalf of the Navigators behind it,
2) managing certain Ingrid Topology maintenance (garbage collection) functions, thus simplifying Navigator implementation, and
3) acting as a firewall between internal Navigators and external Ingrid components.

There are many questions about the effectiveness and cost of this caching scheme, and about searching efficiency in general. These questions can only be answered through real experience. Our expectations, however, is that typical operating parameters for forward information will be 10s of Persistent Forward Information "links" per Resource Profile, and 10s to 100s of Cached Forward Information links per Persistent Forward Information link.

Algorithm for Automatically Building the Ingrid Topology

This section discusses how to build the "base" Ingrid Topology--that is, how to install the Persistent Forward Information associated with each Resource Profile. In general, the term Ingrid Topology refers only to the Persistent Forward Information.

The basic principle behind installing Persistent Forward Information is simple: When a new Resource Profile needs to be installed, it searches for itself, and then connects to whatever it finds.

That having been said, let's discuss it a little further. Each Resource Profile is associated with a single FIServer. Each Resource Profile has a set of terms. From these terms, a (combinatoric) number of term combinations can be generated. For each such term combination, there may (or may not) exist a corresponding cluster. To fully join the Ingrid Topology, then, the Resource Profile must add a link to every existing such cluster.

A search of the Ingrid Topology will retrieve a set of Resource Profiles whose terms match as many of the term combinations in the joining Resource Profile as possible. By adding links to these Resource Profiles, the new Resource Profile effectively joins the appropriate clusters. The mechanism for creating a link is to add Persistent Forward Information both to the FIServer of the new Resource Profile, and to the FIServer of the neighbor Resource Profile.

In order to keep the topology sparse, the Resource Profile adds links to as small a number of other Resource Profiles as possible while still trying to join as many clusters as possible. So, for instance, if new Resource Profile ABCDE finds Resource Profiles ABCD and ABC, it will add a link to only Resource Profile ABCD, since a link to both would be redundant, and a link to Resource Profile ABC alone would result in fewer clusters being joined.

In many (most?) cases, a new Resource Profile may have to make a trade-off between the number of links created and the number of clusters joined. We don't know how this will play out, but one observation is that a Resource Profile only need join those clusters for which it "expects" to be searched. For instance, it may be possible for a Resource Profile to, over time, monitor how it has been searched, and to change clusters, or even change the weighting of its own terms, correspondingly.

Maintaining connected clusters

No system in Ingrid maintains any explicit state about clusters. This is because there are a combinatoric number of clusters, and Ingrid could not scale if it had to maintain any kind of per-cluster information. For instance, cluster boundaries are not labeled, and the only system that may ever explicitly know the full membership of a given cluster is a Navigator that recently fully explored a cluster. FIServers do not know the full membership of a cluster. They only know their neighbor Resource Profiles.

This raises the question of how it is known if a cluster is connected or partitioned. In fact, it is not known. When a Resource Profile X attaches to another Resource Profile Y, the assumption is that, for each term combination that can be generated from Y's terms, Y has successfully joined the corresponding cluster. There is no explicit information, however, that indicates whether or not this is true. Thus, clusters will on occasion be partitioned. How frequent this is in practice, what the practical consequences of it are, and what can be done about it if it is a problem, remains to be seen.

One rule that is required to help maintain cluster connectivity is: A Resource Profile can only attach to Resource Profiles that are "older" than it. This is to prevent dependency loops. For instance, consider the case where, for a given term combination, Resource Profile Z attached to Resource Profile Y, which had attached to X, which had attached to W.

Z-->Y-->X-->W

Assume that later, Resource Profile X discovered that Resource Profile Z was a much better match, and that it could replace a number of links, including the one to W, by one link to Z. Z's path connectivity to W, however, depends on X. If X were to remove its link to W and attach to Z, a dependency loop would form (Z-Y-X-Z), and the cluster would become partitioned.

The mechanism for labeling the "age" of Resource Profiles is as follows. Every Resource Profile has a single Join Sequence Number. When a Resource Profile attaches to the Ingrid Topology, it sets its Join Sequence Number to be one higher than that of the neighbor with the highest Join Sequence Number. Subsequently, the Resource Profile cannot attach to another Resource Profile with a higher Join Sequence Number (though Resource Profiles with higher Join Sequence Numbers can of course attach to it).

A note on the Global Single-Term Server (GSTServer)

A new Resource Profile may have term combinations that no other Resource Profile contains. When this happens, there is obviously no existing cluster with that term combination for the new Resource Profile to join. However, because of the new term combination, the corresponding new cluster is created by virtue of the new Resource Profile joining the Ingrid Topology (though the new cluster has only one member). The new Resource Profile has therefore created new points in Ingrid space (or, new locations in the Ingrid topography, however you like to think about it).

It also may happen that a new Resource Profile has an individual term that no other Resource Profile has. The new Resource Profile will know that it is a new term because the term will not be listed by the GSTServer. Because the new term will exist after the new Resource Profile joins the Ingrid Topology, the GSTServer must be updated with the new term. It is the responsibility of the FIServer that owns the new Resource Profile to insure that the GSTServer obtains the new term.

Essentially, the FIServer tells a GSTServer to add a Forward Information link for that term pointing back to it. The GSTServer will then update its neighbor GSTServers as to the new term, and they will update their neighbors, and so on until all GSTServers have the new Forward Information. (Note that the network of GSTServers will, according to current thinking on the topic, be manually configured as a sparse topology. This is similar to the scheme used by Harvest.)

Browsing the Ingrid Topology

It is well known that keyword-style searching is only a part of the overall searching process. "Relevance feedback" is an important part of any searching process, whether or not it is directly supported by the search mechanism or not. Relevance feedback is where the user indicates to the search system which (key-word)

matching resources are in fact good matches. The system then finds resources strongly related to the good matches.

Another useful function of a typical searching system is to suggest alternative terms that the user may use in subsequent searches.

Because related resources are near each other in the Ingrid Topology, both of the above features can be efficiently provided.

For the case of relevance feedback, an Ingrid Navigator can efficiently traverse the links surrounding a selected Resource Profile, returning to the user those Resource Profiles that have the most terms in common with the selected Resource Profile.

For the case of suggesting alternative terms, an Ingrid Navigator can traverse the links in the clusters that match the user's search terms, and return to the user those terms that appear most often in the Resource Profiles found.

Note that in neither of the above cases is the user explicitly aware of the actual Ingrid links. The Ingrid Navigator automatically traverses nearby links on behalf of the user and returns the results. Actually, the first version of the Ingrid Navigator we developed early in 1995 allowed the user to follow Ingrid links (similar to the operation of HTML browsers). We quickly found, however, that this was of limited, and sometimes misleading, use. One reason is that it didn't allow the user to see the overall structure of information. Another reason is that, since Ingrid links are automatically placed, and since they are not anchored in the text of the resource, it is often not clear to the user why the link exists or if it should be followed. The second version of the Ingrid Navigator eliminated link-level browsing altogether.

The Ingrid Navigator under development for the alpha-release of Ingrid will have both of the above feedback features. When a search is executed using the Ingrid Navigator, it will display both:

1) the set of matching resources (usually by title), in order of best match, and
2) the set of related terms, in order of most related.

The user will be able to tag resources and terms as being either particularly relevant or particularly irrelevant. This will result in an updated display, and may result in additional browsing (if the user so indicates). The user may of course modify the set of search terms and execute another search.

Scaling

The main scaling concern is the amount of Persistent Forward Information required to maintain a correct (or adequately correct, by some measure of adequate) Ingrid Topology.

For a fully correct Ingrid Topology, every Resource Profile must be connected to a cluster for each of its possible term combinations. From our (limited) experience in automatic term weighting, we found that from 10 to 15 terms were adequate to describe a medium-size document (around 10 pages). Assume a Resource Profile with 20 terms. In the theoretical worst case, it is possible to generate 184,756 different term combinations, none of which are subsets of any others (this is the case where each term combination has exactly 10 terms). Thus, in the worst cast, such a Resource Profile would require 184,756 links.

For a lower bound, if there exists another Resource Profile with the same 20 terms, then the given Resource Profile can join all possible clusters with a single link. (Or, if no other Resource Profile exists with any of the 20 terms, no links are required.) Note that FIServers require no explicit labeling of cluster boundaries. FIServers know only what terms their own Resource Profiles have, and what terms their neighbor Resource

Profiles have. Therefore, for the above case, even if the single link to another Resource Profile with the same 20 terms allows the Resource Profile to belong to roughly one million (2**20) clusters, a simple list of the 20 terms is sufficient to effectively "label" all the clusters.

Somewhere between these two bounds--a single link and thousands of links--lies reality. The actual number of links cannot be predicted, because the number depends on what the collection of Resource Profiles is. There are, however, two positive characteristics that lead us to believe that the actual number of links will be manageable.

Consider a given Resource Profile X. All other Resource Profiles in the world can be put into one of two groups:

1) those that share no terms with Resource Profile X (the Zero Group), and
2) those that have terms in common with Resource Profile X (the Shared Group).

The first positive characteristic is that the total number of Resource Profiles in the Zero Group has absolutely no effect on the number of X's links. There is never a reason for Resource Profile X to attach to a Resource Profile with no shared terms. Practically speaking, this means that the large majority of resources in the world have no effect on a given resource (or, for that matter, on an Ingrid Navigator). They are as invisible.

The second positive characteristic is that, after a certain point, the more Resource Profiles there are in the Shared Group, the fewer links Resource Profile X requires. The graph of Figure 10 shows how increasing the total number of Resource Profiles effects the number of links per Resource Profile. As the number of Resource Profiles increases from zero, the number of links per Resource Profile increases. However, at some point, the number of links per Resource Profile begins to decrease with continued increase in the total number of Resource Profiles.

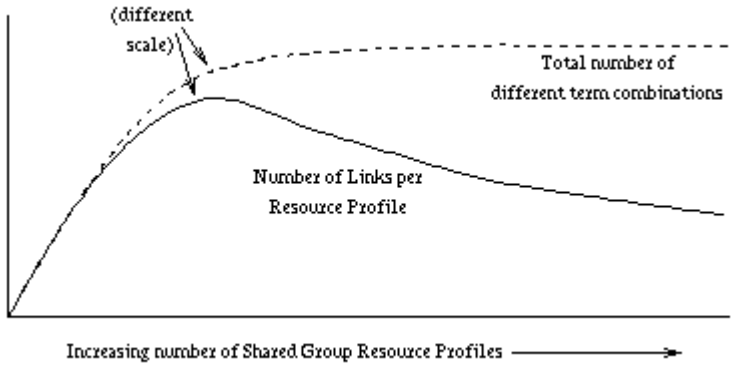

Figure 10: Growth of Links Versus Resource Profiles

The reason for this can be seen from the line that plots the total number of different term combinations against increasing Resource Profiles. Because all of the Resource Profiles in the shared group can be said to come from the same "topic area", the total number of different terms is bounded. As a result, as the number of Resource Profiles grows, the total number of term combinations represented begins to saturate. New Resource Profiles start tending to have the same term combinations as existing Resource Profiles. Correspondingly, the increase in the total number of clusters also slows down. But, since the number of Resource Profiles continues to grow faster than the number of clusters, the task of maintaining connected clusters is spread across more and more Resource Profiles, with the result that each individual Resource Profile requires fewer links.

Even given this phenomenon, it is unknown where the peak of the curve lies. Therefore a scaling problem may still exist. This is one of the two main reasons that only high-weight terms are used to form the Ingrid Topology. By limiting the number of terms, we also limit the number of links required between Resource Profiles. The other reason is that the "quality" of the Ingrid Topology may be lowered if less than high-weight terms are used. The number of terms used for a Resource Profile is a trade-off between search quality and quantity. It is not clear whether the number of terms chosen will be limited by a desire for quality over quantity, or limited by scaling.

Note also that it is not necessarily necessary (or even possible, practically speaking) to maintain a fully correct Ingrid Topology. An imperfect Ingrid Topology means simply that a search for a given set of terms will not yield every matching resource. It may be adequate for, say, 90% or 80% of the total number of Resource Profiles with a given term combination to actually be in the same cluster.

A reasonable strategy for inserting Resource Profiles into the Ingrid Topology is to have an upper bound on the average number of links per Resource Profile (within a given FIServer). An FIServer can add and delete links according to observed search patterns to maximize search quality.

Experimental Results

A small experimental Ingrid Topology was built in April of 1995. It consisted of roughly 4000 related documents (RFCs, IETF Internet Drafts and meeting reports, and the archives of a few IETF mailing lists). Terms were automatically weighted using term frequency by inverse document frequency style weighting [1]. The average number of terms selected per Resource Profile was approximately 12. The total set of selected terms resulted in around 5000 distinct terms.

The Ingrid Topology built was fully or near-fully correct. We say near-fully because, while the insertion software was designed to connect to all possible clusters, we did not in fact examine all clusters for full connectedness. Rather, we examined a small number of arbitrarily selected clusters, mainly for the purpose of checking correctness of the software.

The average number of links per Resource Profile was roughly 10. The Resource Profile with the most links had 50. The minimum was two links (which occurred when two versions of the same paper were inserted, with the result that one would find the other and attach only a single link).

While this experiment was way too small to be considered definitive, we were pleased by the results we did get. Ten links per Resource Profile is quite a low overhead. This is especially true when considering that the collection we chose is strongly related, as evidenced by the fact that 4000 resource produced only 5000 distinct terms. In other words, the collection truly represents a single topic area. The Resource Profiles of this collection would be strongly inter-connected even in a "real" (large and diverse) Ingrid Topology. Therefore, these links may represent a significant proportion of the total number of links the Resource Profiles would have in a global Ingrid Topology.

Global Single-Term Servers

Another scaling concern is that of the GSTServers. In the current design, each GSTServer contains three entries for each term in all of Ingrid space. (We say "current" design, because if this proves excessive, it is always possible to create a hierarchy of GSTServers, at the cost of extra complexity.) This section provides some rough back-of-the-envelope calculations for GSTServer size.

Assume that each entry requires, on average, 200 bytes (one term plus three host names plus overhead). A good dictionary for any given language has on the order of 200,000 words (including proper names). As a

worst-case estimate, assume that the most common 300 languages are fully represented. This requires 12 gigabytes of memory. Double this to (roughly) account for technical, scientific, and other specialized terms. 24 gigabytes is still well within current technology.

To continue the worst-case assessment, assume that each entry requires updating once per week. (This implies that the average lifetime of a Resource Profile is one week--a conservative estimate.) According to the current protocol for updating an entry, three packets are required:

1) one to inform the GSTServer of the old and new entries,
2) one to request verification of the new entry,
3) one to receive verification of the new entry.

This makes roughly 600 packets per second. Assuming that each packet is 200 bytes, we get around 100kbytes per second. Again, within current technology, though a system that can manage 200 updates per second of a 24 gigabyte database and still have processing leftover to handle queries is no small machine. On the other hand, it will be some time before 300 languages are fully represented.

Project Status and Goals

For an up-to-date project status, please consult our Ingrid project home page at http://www.ntt.jp/ntt/soft-labs/ingrid/.

We implemented a demo-version of Ingrid in spring of 1995, and built a small Ingrid topology (4000 resources) spread over four machines, and a simple navigator (see Section 4.1). Nothing in that experiment led us to believe Ingrid would fail. However, we didn't build anywhere near big enough a topology to really test Ingrid.

Therefore, we are now working towards a medium-scale test of Ingrid. Our target is to start a limited alpha-test towards the end of 1995. By limited, we mean approximately 100 hosts and 100,000 resources. This should give us some preliminary estimates on the scaling and performance of Ingrid. Depending on how the alpha-test goes, we hope to release a beta-version sometime around mid-1996.

We are committed to providing full public-release software of all components of Ingrid, including the infrastructure components (the FIServer and GSTServer), a Navigator, and Resource Profile creation and insertion software in Japanese and English (plus hooks to easily add other languages). This having been said, we have applied for patents on the basic Ingrid topology and related algorithms (basically, what is covered by the Section labeled *Description of Ingrid*). If the patents are awarded, our policy is that no royalties will be applied against the distribution of non-commercial Ingrid software, or against the use of Ingrid for non-profit activity. We will reserve the right, however, to charge royalties on the sale of commercial implementations of Ingrid, the use of Ingrid to advertise for-profit products.

(We recognize that a technical proceedings is not normally an appropriate place to make this kind of statement. However, given the short path from research experiment to commercial product typical on the web, and given the frequent misunderstandings over patents, we feel that such a statement is not inappropriate.)

Last Comments

This paper describes the fundamental aspects of a scalable, fully distributed, fully self-configuring information navigation infrastructure. It compares it with both competing and complementary technologies, presents some arguments as to why it may scale, and gives some limited experimental results on scaling.

There are many issues that we have given considerable thought that are not discussed in this paper. These include (but are not limited to) implementation issues (our implementation is fully connectionless, running over UDP), topology maintenance issues (rather than do constant pinging, we discover and repair errors during normal use), coordination of distributed resource profiling (when different people profile and insert the same resource), quality control of resources, other applications (such as a resource referral service like the HOMR Music Recommendation Service), multi-lingual issues (the Ingrid Infrastructure uses only the Mule [2] MULtilingual Enhancement to GNU Emacs 19 character set), private Ingrid (firewalls and encryption), interaction with single-database search engines, and richer search semantics (boolean, partial matching).

We feel that we have reasonable answers for most of these issues. The two issues of greatest concern at this point (besides scaling) are 1) how to not be overwhelmed by irrelevant or garbage resources (understanding that one man's garbage is another man's gold), and 2) how to deal with the security problems created by the fact that, in the general case, any FIServer can request any other FIServer to add Persistent Forwarding Information.

In any event, implementation and experience will be required to better understand these issues.

Acknowledgements

We would like to thank Jeff Smith for his review of this paper, and for many interesting and fruitful discussions of Ingrid. We would like to thank Shigeki Goto, Ryouichi Hosoya, Masaki Itoh, Katsunori Kon, Minoru Koyama, Kenichiro Murakami, Yutaka Ogawa, Hitoaki Sakamoto, Jeff Smith, and Kenji Takahashi (all of NTT) for their support of or contributions to the Ingrid project.

List of URLs

Centroids, http://services.bunyip.com:8000/products/digger/digger-main.html

Fish-Search, http://www.win.tue.nl/win/cs/is/reinpost/www94/

Harvest, http://harvest.cs.colorado.edu/

HOMR, http://homr.www.media.mit.edu/projects/homr/

Lycos, http://query2.lycos.cs.cmu.edu/

Robots, http://web.nexor.co.uk/mak/doc/robots/robots.html

WWWW, http://www.cs.colorado.edu/home/mcbryan/WWWW.html

References

1. Salton, G. *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

2. Mule, anonymous FTP at ftp.iij.ad.jp:/pub/misc/mule.

3. Mockapetris, P. "Domain names - implementation and specification," RFC 1035, anonymous FTP at ds.internic.net:/rfc/rfc1035.txt.

About the Authors

Paul Francis [http://www.ntt.jp/people/francis/]
NTT Software Labs
francis@slab.ntt.jp

Takashi Kambayashi [http://www.ntt.jp/people/kam/]
NTT Software Labs
kam@slab.ntt.jp

Shin-ya Sato [http://www.ntt.jp/people/sato/]
NTT Network Service Systems Labs
sato@sphere.csl.ntt.jp

Susumu Shimizu [http://www.ntt.jp/people/shimizu/]
NTT Software Labs
shimizu@slab.ntt.jp

This document can be found at http://www.ntt.jp/ntt/soft-labs/ingrid/)