Ingrid: A Self–Configuring Information Grid

Paul Francis

NTT Software Laboratories
3–9–11 Midori–cho, Musashino–shi, Tokyo, 180 Japan
+81 422 59 3843
francis@slab.ntt.jp
(This document is at http://www.ntt.jp/ntt/soft–labs/ingrid/overview.html)

1. Quick Overview of the Ingrid Project

This paper gives an overview of Ingrid and the Ingrid project at NTT's Software Laboratories. Ingrid is a system for navigating (searching *and* browsing) globally distributed information. It works by organizing information into a certain mesh topology (the *Ingrid Topology*) that can be efficiently navigated.

Our purpose with this paper is to convince you that Ingrid is *plausible* (we're not yet convinced ourselves that it will definately work) and that it deserves large–scale experimentation.

The goals of Ingrid are:

- Medium–quality ("best effort") keyword–style searching of *any* text resources stored *anywhere*.
- High–quality browsing of the text resources (or, non–text resources described by text).
- Fully automatic operation (works in the absence of keyword or attribute selection and requires no manual placement of links).
- Fully distributed navigation (no dedicated single–database search engines are required).
- No limitations on the global number of resources or hosts.
- Adding value to resources by placing them near other relevant resources in Ingrid space.

The *non*–goals of Ingrid are:

- Full–text searching (not).
- URL–URN resolution (not).
- Traditional white–pages (X.500) or DNS directory service (at least, Ingrid wouldn't be the most efficient way to do this).
- Replace current HTML links (not). Ingrid is meant as an enhancement to, not a replacement of, the current Web.

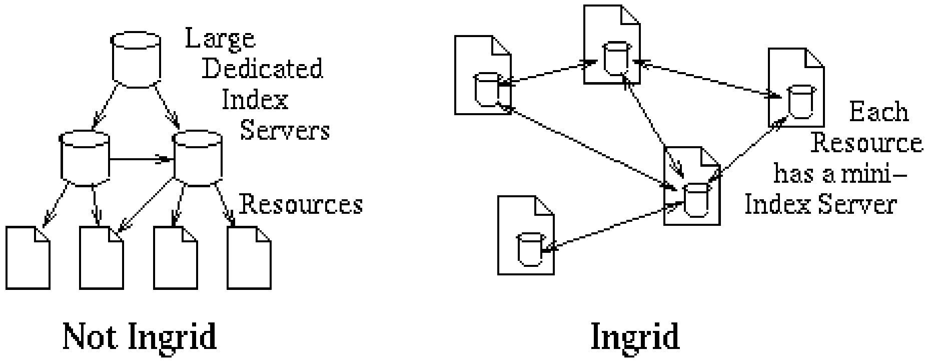
2. Some Basic Design Principles

2.1 Ingrid can be Fully Distributed

The major distinguishing characteristic of Ingrid is that it does not require any kind of dedicated "index server" (to borrow a term from WHOIS++) infrastructure (well, with one exception, discussed later). Rather, every resource acts as a kind of "mini–index server" in the sense that:

1) each resource is linked to (has "forward knowledge" about, to borrow again from WHOIS++) a small number of (similar) *peer* resources, and
2) using the forward knowledge, each resource can answer queries from navigation servers.

Figure 1: Comparison of Ingrid and Traditional Index Structure (.ps)



This approach 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, actually, holds the *Ingrid node* that represents the resource). 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.)

Another advantage is that, by eliminating the need for a dedicated infrastructure up front, we make the initial buy−in small, and increase the probability that Ingrid will get off the ground at all.

The major disadvantage to this approach 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 Ingrid nodes in a relatively small number of machines, thus effectively making those machines dedicated index servers.

2.2 Ingrid can be Fully Automatic

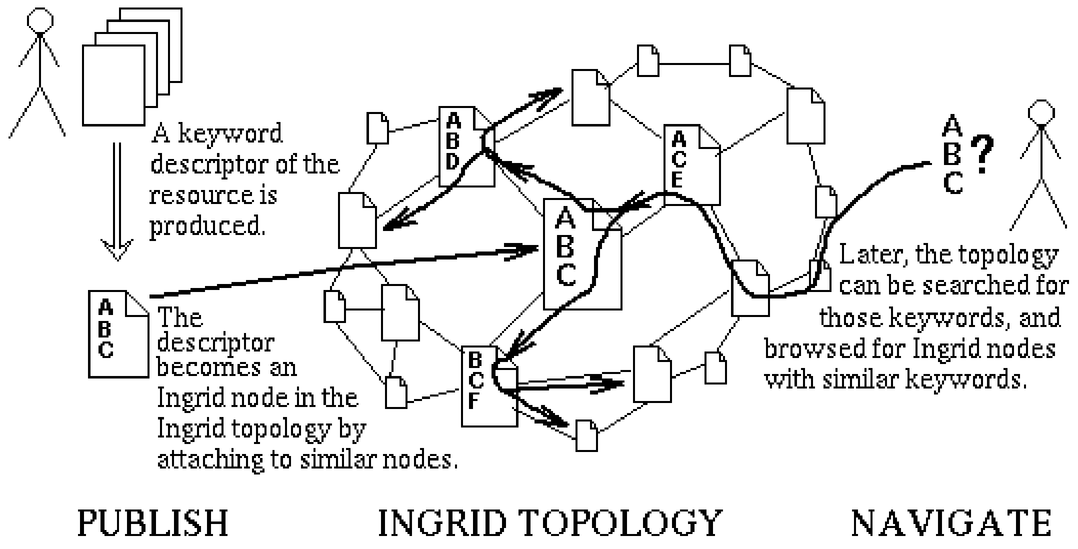
Another factor in getting off the ground is buy−in in terms of human effort. Even if we can leverage the resources of every information server for navigation, it won't fly if it requires a large configuration effort.

Ingrid automatically installs and maintains its links. In addition, we have implemented traditional term−weighting techniques so that even keywords are automatically chosen. (In what follows, we will generally use the word "term" rather than "keyword".) This is possible in part because Ingrid does not require attribute−value type objects. While automatic term selection is never perfect, our experience so far indicates that it is effective, and can be improved.

3. A Functional View of Ingrid

Ingrid has three major functional components; *publishing*, *navigating*, and *Ingrid* *topology*. (Well, there is a fourth, but I don't want to confuse things by discussing it yet.)

Figure 2: Ingrid has Three Major Functional Components (.ps)



Inside every Ingrid node are a set of terms that describe the resource associated with the Ingrid node. The terms can be manually or automatically generated, and the resource they describe can be anything; a document, a video clip, a physical object, or a person. The Ingrid node is then linked to other Ingrid nodes that have, to the extent possible, identical terms. The act of generating these terms, and of inserting the resulting Ingrid node into the Ingrid topology is called *publishing*.

A user searches the Ingrid topology for terms that (s)he believes best describes the resources the user wants. As with any information retrieval system, sometimes the user will know exactly what terms to use, but other times the user must guess. This search will produce Ingrid nodes that contain the search terms (or a superset of the terms). If no such Ingrid nodes exist, the search will produce the Ingrid nodes that best match the search terms. In any event, after the user has found a set of nodes, the user can browse around the neighborhood for resources similar to the ones (s)he has found. Because Ingrid nodes are attached to other nodes with similar terms, the user may often find relevant resources even though the user did not start with the terms that describe the relevant resource.

3.1 What is an Ingrid Link?

Note that the links between Ingrid nodes are similar to but *not the same as* the links between HTML, or other hypertext, documents. They are similar in that they are URLs (or, perhaps, some day, when our children are retired, URNs). But they are different in that they are not anchored in the text of the resource per se. Rather, they point from a resource in total to another. They are also different in that, while like

HTML links each Ingrid link is uni−directional, Ingrid links almost always come in pairs. That is, Ingrid nodes almost always point to each other.

But most importantly, note that Ingrid links are completely orthogonal to their HTML counterparts. The existence of an HTML link from one resource to another does not imply the existence of an Ingrid link between the same two resources. Nor do we have any plan to or see any particular benefit to relating Ingrid links and HTML links. (This is of course not to say that some smart browser can't simultaneously take advantage of both kinds of links, in addition to WHOIS++ centroids and WAIS databases and everything else that is out there.)

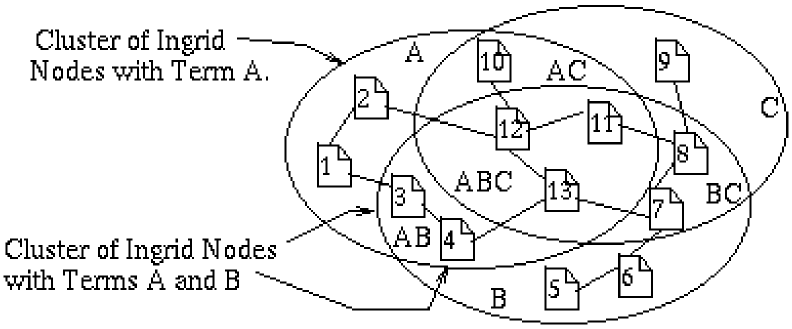
4. What Does the Ingrid Topology Look Like?

Nothing said up to now is particularly novel. Most of what we've described so far could just as well be said about the current Web. What is different is that with Ingrid we propose to be able to efficiently search *the entire global Ingrid topology* (potentially the entire Web) for *all resources* that are characterized by a given set of terms. This level of searching is impossible with the current Web.

How we propose to do this has everything to do with the structure of the Ingrid topology. Indeed most of what's been said up to now is just to lay the context for what comes next−−−the description of the Ingrid topology.

The definition of the Ingrid topology is actually quite simple. Assume a set of Ingrid nodes, each with a set of terms. 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 Ingrid nodes that contain those terms form a cluster.

Figure 3: Ingrid Topology with Three Terms, A, B, and C, and various combinations (.ps)



For instance, Figure 3 shows 13 resources connected in a mesh topology. Each of the resources has one or more of terms A, B, and C. All resources that have the same term or set of terms (that is, the same *term combination*) form a cluster. For instance, all resources 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).

Within the limitations of the above definition, we try to keep the Ingrid topology sparse, so that each resource has a small number of links.

4.1 Some Advantages of the Ingrid Topology

The Ingrid topology has a number of potential advantages.

Efficient Searching

First, from a searching perspective, you only need to find one resource with a given term combination in order to easily find all resources with that term combination. Once you find one, you simply browse among the immediate neighbors (robot−style) until all such resources are retrieved. Thus, if you already know of a relevant resource, finding more is efficient.

Furthermore, finding the first relevant resource (that is, finding at least one resource that contains a user−specified set of terms) *may* also be efficient. This is because the search can successively find clusters with more terms, each time narrowing the search significantly. In this sense the Ingrid topology is hierarchical. The set of resources with terms A, B, C, and D are nested within the set of resources with terms A, B, and C, which are in turn nested within the set of resources with terms A and B. (More on this below .)

High−quality Browsing

Second, from a browsing perspective, once you find a relevant resource, everything nearby is potentially relevant in that it shares terms with your resource. Thus, browsing is high quality. In fact, while browsing Ingrid, you are not necessarily or often aware of the Ingrid topology per se (you do not explicitly traverse links). Rather, you are able to activate a certain set of terms, and see

1) a (possibly partial) list of resources with the active terms, and
2) other terms that are related to the active set.

By clicking on various terms, you can increase or decrease or change the scope of your view (or, in terms of Ingrid mechanism, change the active cluster). We call this *term−level browsing* (as opposed to the usual *resource−level browsing*).

The Topology Is the Information

The value of a given resource may be increased by its inclusion in the Ingrid topology. In the extreme, a resource that has almost no intrinsic value on its own may become valuable in the Ingrid topology.

An example of this is the following application, which I call "Top Ten Favorites". With this application, users can query Ingrid to find CDs, books, movies, etc., that they will probably enjoy. The query itself consists of a list of their own favorites. This query itself becomes a node in the Ingrid topology, thus connecting to nodes that have favorite–lists with significant overlap. Items in nearby favorite–lists that are not in the user's own favorite–list are likely to be things the user would also like.

Any one of these lists has little value of its own (or, more accurately, it is of no value to most people but highly valuable to people with similar but not identical lists). Because users are able to find similar lists through Ingrid (and in so doing provide their lists to other users), the information value of each list is greatly increased.

5. Why We Think Ingrid Will Scale

We're not sure it will–––we need to experiment. We have, however, a few principles that give us hope.

5.1 The Ingrid Topology Itself

The above drawing of the Ingrid topology (Figure 3) is something of a misnomer in that it explicitly shows cluster boundaries. In Ingrid clusters are not explicitly labeled, nor does any system need to keep a list of the Ingrid nodes in any cluster. Rather, the information at each Ingrid node consists of:

1) the Ingrid node's own terms,
2) the Ingrid node's own links (URLs for its neighbor nodes),
3) the terms of its neighbor nodes, and
4) any other relevant information about the resource represented by the Ingrid node, such as resource title, resource URL, and so on.

Thus, the size of a cluster or number of clusters is not a scaling issue. Navigators can determine the "boundaries" of the clusters they are interested in simply by observing which neighbors have or don't have the terms of the cluster.

In the limited experimentation we've done so far, we find that 10 to 15 terms is adequate to describe a typical (10,000 word) RFC. So the number of terms is not a scaling issue.

We believe that Ingrid will also scale well in terms of the number of links. This derives from the fact that Ingrid is, to the extent possible, a sparse topology, and from the fact that multiple Ingrid nodes *share* the load of maintaining the Ingrid topology.

For instance, consider again the Ingrid topology of (Figure 3) . Note first that Node 13 is not attached to Node 11. It is enough that Node 13 is attached to Node 12, which is attached to Node 11. Nodes 11, 12, and 13 share the load of maintaining cluster ABC. This includes sharing the load of links going to other clusters (Nodes 12 and 13 each have two links going outside of cluster ABC, and Node 11 has another.) Moreover, Node 13 derives all of its connectivity to cluster A (7 nodes) through only two links−−−those with Node 4 and Node 12.

Because the global vocabulary is more−or−less bounded (big, but bounded), and because terms do not combine in arbitrary ways (it is more likely a resource with the term "computer" will contain the term "software" than the term "bladder"), an increase in the number of nodes implies increased sharing, and thus fewer links per node. In fact, we did a simulation where we increased the number of nodes without increasing the vocabulary, and found indeed that more nodes translates into fewer links per node. But, this is just simulation, so it is of course no proof.

5.2 Searching the Ingrid Topology

This is the part we're the least sure about. Though we've got a few fancy ideas on searching heuristics, basically we're hoping that simple caching will make searching efficient.

When we say searching the Ingrid topology, we mean finding at least one resource with a given set of terms. The basic approach to searching the Ingrid topology is to successively find subclusters within the cluster already found, thus adding additional terms to those already found. So, say for instance that we are looking for a resource with terms A, B, and C (Figure 3) , and have already found Node 1 (with just term A). From resource 1, we search nodes with term A (that is, we *search cluster A*) until we find an additional term. In this case, we find resource 3, which additionally has term B. Now we search cluster AB until we find a resource with term C. In this case, we go to Node 4 and then find Node 13. (Note that this search isn't being forwarded from node to node by the nodes themselves. Rather, a robot is doing this search by querying nodes for their knowledge and then picking the next best node.)

The problem of course is that there may be many thousands of nodes with a given term combination. Thus, finding the few with an additional searched−for term may be inefficient.

Our initial approach to this problem is caching. When an *Ingrid navigator* (robot) searches, it caches what it learns along the way. If the Ingrid navigator shares a computer with or is otherwise directly associated with an *Ingrid server* (a computer that stores Ingrid nodes and answers queries), the Ingrid server also caches the intermediate results. Further, the Ingrid servers that it queries can cache the source of the query (the Ingrid navigator), and refer future similar queries to that Ingrid navigator.

Thus, an Ingrid navigator will usually already know of a number of relevant Ingrid servers when it starts a search, and the search will therefore already be well−narrowed before it even starts. Further, the Ingrid servers will have cached information about the more common searches, so usually a search will benefit from cache hits.

Indeed, a cache entry is just another kind of link. The Ingrid links already described form the basic infrastructure and are especially useful for browsing, while the cache "links" point to popular term combinations and make searching more efficient.

5.2.1 Finding the First Term

The enquiring reader may be wondering how it is that an Ingrid navigator finds an Ingrid node with any relevant terms at all if the navigator knows of no such nodes at the start of a search. This is particularly a problem if the navigator is looking for a single or small number of rare terms.

To prevent the navigator from visiting thousands of Ingrid nodes trying to find even one term, we propose a mesh of index servers, called *global term servers* (this is the exception we referred to earlier).

Each global term server maintains an entry for every individual term (not term combination) in the global vocabulary. Each entry points to just a few (say three, just for robustness) Ingrid nodes. When a navigator has reasonably explored all other alternatives and still can't find an Ingrid node with a given term, it can query a global term server.

The global term servers themselves scale in terms of memory because they scale according to the size of the global vocabulary, which is bounded (at some millions of terms, but bounded none−the−less). The global term servers should scale in terms of number of queries because of caching−−−it will be unusual for a navigator to know absolutely nothing about the terms it is searching for.

5.3 Adding Nodes to the Ingrid Topology

The basic approach to adding new nodes to the Ingrid topology is to first search the Ingrid topology for the terms of the new node, and then attach to what was found. The new node tries to "satisfy" as many combinations of its own terms as it can. That is, it tries to attach to existing nodes that, taken together, already have every term combination that can be generated from the new node's terms.

If there already exist Ingrid nodes with many similar terms to the new node, then each of these existing nodes will satisfy a large percentage of the new node's term combinations, and the new node will need to attach to only a few existing nodes. If on the other hand, the existing Ingrid nodes each share only a few terms with the new node, then the new node may have to attach to many such existing nodes to satisfy its term combinations. In so doing, the new node essentially creates a new point in Ingrid space.

In any event, adding new nodes to the topology scales the same as searches, which we've already hand−waved about.

6. Lots More

Of course there are many many issues not discussed in this short overview (nor mentioned in this final section). Among these are issues of topology dynamics−−−how to handle resources that are deleted, how to update the global term server, and so on. Our general approach is to 1) require no "pinging" anywhere, but let everything be event driven, and 2) be very tolerant of "incorrect" topology, such as "one−way" Ingrid links and partitioned clusters.

Another set of issues have to do with security. We are proposing that, when adding an Ingrid node, the existing Ingrid node's that it attaches to should form links back to the new Ingrid node. This no doubt presents all sorts of new opportunities for abuse. We do not yet have much of a handle on this and related problems.

While we will try to solve as many problems as we can in the lab, we will ultimately need to put Ingrid on the net and see how it goes. We hope you will join us in this experiment.