Network Working Group
Request for Comments: 1727
Category: Informational

C. Weider
P. Deutsch
Bunyip Information Systems

December 1994

A Vision of an Integrated Internet Information Service

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This paper lays out a vision of how Internet information services might be integrated over the next few years, and discusses in some detail what steps will be needed to achieve this integration.

Acknowledgments

Thanks to the whole gang of information service wonks who have wrangled with us about the future of information services in countless bar bofs (in no particular order): Cliff Lynch, Cliff Neuman, Alan Emtage, Jim Fullton, Joan Gargano, Mike Schwartz, John Kunze, Janet Vratny, Mark McCahill, Tim Berners-Lee, John Curran, Jill Foster, and many others. Extra special thanks to George Brett of CNIDR and Anders Gillner of RARE, who have given us the opportunity to start tying together the networking community and the librarian community.

1. Disclaimer

This paper represents only the opinions of its authors; it is not an official policy statement of the IIIR Working Group of the IETF, and does not represent an official consensus.

2. Introduction

The current landscape in information tools is much the same as the landscape in communications networks in the early 1980's. In the early 80's, there were a number of proprietary networking protocols that connected large but autonomous regions of computers, and it was difficult to coalesce these regions into a unified network. Today, we have a number of large but autonomous regions of networked information. We have a vast set of FTPable files, a budding WAIS network, a budding GOPHER network, a budding World Wide Web network,

etc. Although there are a number of gateways between various protocols, and information service providers are starting to use GOPHER to provide a glue between various services, we are not yet in that golden age when all human information is at our fingertips. (And we're even farther from that platinum age when the computer knows what we're looking for and retrieves it before we even touch the keyboard.)

In this paper, we'll propose one possible vision of the information services landscape of the near future, and lay out a plan to get us there from here.

3. Axioms of information services

There are a number of unspoken assumptions that we've used in our discussions. It might be useful to lay them out explicitly before we start our exploration.

The first is that there is no unique information protocol that will provide the flexibility, scale, responsiveness, worldview, and mix of services that every information consumer wants. A protocol designed to give quick and meaningful access to a collection of stock prices might look functionally very different from one which will search digitized music for a particular musical phrase and deliver it to your workstation. So, rather than design the information protocol to end all information protocols, we will always need to integrate new search engines, new clients, and new delivery paradigms into our grand information service.

The second is that distributed systems are a better solution to large-scale information systems than centralized systems. If one million people are publishing electronic papers to the net, should they all have to log on to a single machine to modify the central archives? What kind of bandwidth would be required to that central machine to serve a billion papers a day? If we replicate the central archives, what sort of maintenance problems would be encountered? These questions and a host of others make it seem more profitable at the moment to investigate distributed systems.

The third is that users don't want to be bothered with the details of the underlying protocols used to provide a given service. Just as most people don't care whether their e-mail message gets split up into 20 packets and routed through Tokyo to get to its destination, information service users don't care whether the GOPHER server used telnet to get to a WAIS database back-ended by an SQL database. They just want the information. In short, they care very much about how they interact with the client; they just don't want to know what goes on behind.

These axioms force us to look at solutions which are distributed, support multiple access paradigms, and allow information about resources to be handed off from one system (say Gopher) to another (say WWW).

4. An architecture to provide interoperability and integration.

The basic architecture outlined in this paper splits up into 4 levels [Fig. 1].

At the lowest level, we have the resources themselves. These are such things as files, telnet sessions, online library catalogs, etc. Each resource can have a resource transponder attached [Weider 94a], and should have a Uniform Resource Name (URN) [Weider 94b] associated with it to uniquely identify its contents. If a resource transponder is attached, it will help maintain the information required by the next level up.

At the next level, we have a 'directory service' that takes a URN and returns Uniform Resource Locators (URLs) [Berners-Lee 94] for that resource. The URL is a string which contains location information, and can be used by a client to access the resource pointed to by the URL. It is expected that a given resource may be replicated many times across the net, and thus the client would get a number of URLs for a given resource, and could choose between them based on some other criteria.

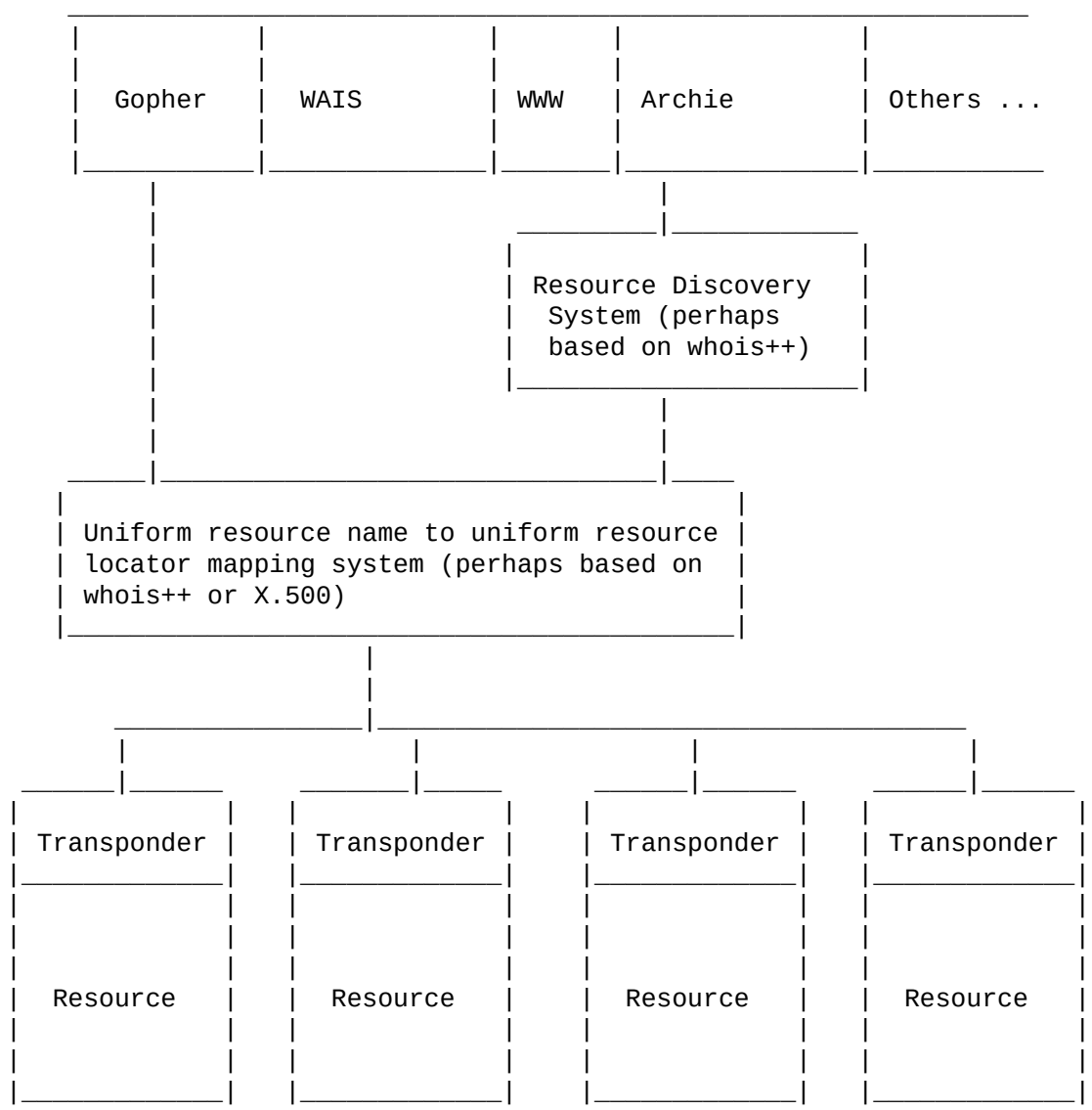

Figure 1: Proposed architecture of an integrated information service

The third level of the architecture is a resource discovery system. This would be a large, distributed system which would accept search criteria and return URNs and associated information for every resource which matched the criteria. This would provide a set of URLs which the information service providers (GOPHER servers, etc.) could then select among for incorporation.

The fourth level of the architecture is comprised of the various information delivery tools. These tools are responsible for collating pointers to resources, informing the user about the resources to which they contain pointers, and retrieving the resources when the user wishes.

Let's take a look in greater detail at each of these levels.

4.1 Resource layer

The resources at this layer can be any collection of data a publisher wishes to catalog. It might be an individual text file, a WAIS database, the starting point for a hypertext web, or anything else. Each resource is assigned a URN by the publisher, and the URL is derived from the current location of the resource. The transponder is responsible for updating levels 2 and 3 with the appropriate information as the resource is published and moves around.

4.2 URN -> URL mapping

This level takes a URN and returns a number of URLs for the various instantiations of that resource on the net. It will also maintain the URN space. Thus the only functionality required of this level is the ability to maintain a global namespace and to provide mappings from that namespace to the URLs. Consequently, any of the distributed 'directory service' protocols would allow us to provide that service. However, there may be some benefit to collapsing levels 2 and 3 onto the same software, in which case we may need to select the underlying protocol more carefully. For example, X.500 provides exactly the functionality required by level 2, but does not (yet) have the functionality required to provide the level 3 service. In addition, the service at level 2 does not necessarily have to be provided by a monolithic system. It can be provided by any collection of protocols which can jointly satisfy the requirements and also interoperate, so that level 2 does appear to level 3 to be universal in scope.

4.3 Resource discovery

This is the level which requires the most work, and where the greatest traps lurk to entangle the unwary. This level needs to serve as a giant repository of all information about every publication, except for that which is required for the URI -> URL mapping. Since this part is the least filled in at the moment, we will propose a mechanism which may or may not be the one which is eventually used.

When a new resource is created on the network, it is assigned a URN determined by the publisher of the resource. Section 4.1 discusses in more detail the role of the publisher on the net, but at the moment

we can consider only 2 of the publisher's functions. The publisher is responsible for assigning a URN out of the publishers namespace, and is responsible for notifying a publishing agent [Deutsch 92] that a new resource has been created; that agent will either be a part of the resource location service or will then take the responsibility for notifying an external resource location service that the resource has been created. Alternatively, the agent can use the resource location service to find parts of the RLS which should be notified that this resource has been created.

To give a concrete example, let's say that Peter and Chris publish a multi- media document titled, "Chris and Peter's Bogus Journey", which talks about our recent trip to the Antarctic, complete with video clips. P & C would then ask their publishing agent to generate a URN for this document. They then ask their publishing agent to attach a transponder to the document, and to look around and see if anyone a) has asked that our agent notify them whenever anything we write comes out; or b) is running any kind of server of 'trips to Antarctica'. Janet has posted a request that she be notified, so the agent tells her that a new resource has been created. The agent also finds 3 servers which archive video clips of Antarctica, so the agent notifies all three that a new resource on Antarctica has come out, and gives out the URN and a URL for the local copy.

4.4 Information delivery tools

One of the primary functions of an information delivery tool is to collect and collate pointers to resources. A given tool may provide mechanisms to group those pointers based on other information about the resource, e.g. a full-text index allows one to group pointers to resources based on their contents; archie can group pointers based on filenames, etc. The URLs which are being standardized in the IETF are directly based on the way World Wide Web built pointers to resources, by creating a uniform way to specify access information and location for a resource on the net. With just the URLs, however, it is impossible without much more extensive checking to tell whether two resources with different URLs have the same intellectual content or not. Consequently, the URN is designed to solve this problem.

In this architecture, the pointers that a given information delivery tool would keep to a resource will be a URN and one or more cached URLs. When a pointer to a resource is first required (i.e. when a new resource is linked in a Gopher server), level 2 will provide a set of URLs for that URN, and the creator of the tool can then select which of those will be used. As it is expected that the URLs will eventually become stale (the resource moves, the machine goes down, etc.) the URN can be used to get a set of current URLs for the resource and an appropriate one can then be selected. Since the cost

of using the level 2 service is probably greater than the cost of simply resolving a URL, both the URN and the URL are cached to provide speedy access unless something has moved.

4.5 Using the architecture to provide interoperability between services

In the simplest sense, each interaction that we have with an information delivery service does one of two things: it either causes a pointer to be resolved (a file to be retrieved, a telnet session to be initiated, etc.) or causes some set of the pointers available in the information service to be selected. At this level, the architecture outlined above provides the core implementation of interoperability. Once we have a means of mapping between names and pointers, and we have a standard method of specifying names and pointers, the interoperation problem becomes one of simply handing names and pointers around between systems. Obviously with such a simplistic interoperability scheme much of the flavor and functionality of the various systems are lost in transition. But, given the pointers, a system can either a) present them to the user with no additional functionality or b) resolve the pointers, examine the resources, and then run algorithms designed to tie these resources together into a structure appropriate for the current service. Let's look at one example (which just happens to be the easiest to resolve); interoperation between World Wide Web and Gopher.

Displaying a Gopher screen as a WWW document is trivial with these pointers. Every Gopher screen is simply a list of menu items with pointers behind them (we'll ignore the other functionality Gopher provides for a moment), so is an extremely simple form of a hypertext document. Consequently with this architecture it is easy to show and resolve a Gopher screen in WWW. For a WWW to Gopher map, the simplest method would be that when one accesses a WWW document, all the pointers associated with links off to other documents are brought in with the document. Gopher could then resolve the links and read the first line of each document to provide a Gopher style screen which contains everything in the WWW document. When a link is selected, all of the WWW links for the new document are brought in and the process repeats. Obviously we're losing a lot with the WWW -> Gopher mapping; some might argue that we are losing everything. However, this does provide a trivial interoperability capacity, and one can argue that the 'information content' has been preserved across the gateway.

In addition, the whole purpose of gatewaying is to provide access to resources that lie outside the reach of your current client. Since all resources are identifiable and accessible through layers 2 and 3, it will be easy to copy resources from one protocol to another since

all we need to do is to move the pointers and reexpress the relationships between the pointers in the new paradigm.

4.6 Other techniques for interoperability

One technique for interoperability which has recently received some serious attention is the technique of creating one client which speaks the protocols of all the information delivery tools. This approach has been taken in particular by the UNITE (User's Network Interface To Everything) group in Europe. This client would sit on the top level of the architecture in Figure 1. This technique is best exemplified by the recent work which has gone into Mosaic, a client which can speak almost all of the major information services protocols. This technique has a lot of appeal and has enjoyed quite a bit of success; however, there are several practical difficulties with this approach which may hinder its successful implementation.

The first difficulty is one that is common to clients in general; the clients must be constantly updated to reflect changes in the underlying protocols and to accommodate new protocols. If the increase in the number of information services is very gradual, or if the underlying protocols do not change very rapidly, this may not be an insuperable difficulty. In addition, old clients must have some way of notifying their user that they are no longer current; otherwise they will no longer be able to access parts of the information mesh.

The second problem is one which may prove more difficult. Each of the currently deployed information services provides information in a fundamentally different way. In addition, new information services are likely to use completely new paradigms for the organization and display of the information they provide. The various clients of these information services provide vastly different functionality from each other because the underlying protocols allow different techniques. It may very well prove impossible to create a single client which allows access to the full functionality of each of the underlying protocols while presenting a consistent user interface to the user.

Much of the success of Mosaic and other UNITE tools is due to the fact that Gopher, WWW, and other tools are still primarily text based. When new tools are deployed which depend more on visual cues than textual cues, it may be substantially more difficult to integrate all these services into a single client.

We will continue to follow this work and may include it in future revisions of this architecture if it bears fruit.

5. Human interactions with the architecture

In this section we will look at how humans might interact with an information system based on the above architecture.

5.1 Publishing in this architecture

When we speak of publishing in this section, we are referring only to the limited process of creating a resource on the net, assigning it a URN, and spreading the information around that we have created a new resource.

We start with the creation of a resource. Simple enough; a creative thought, a couple of hours typing, and a few cups of coffee and we have a new resource. We then wish to assign it a URN. We can talk to whichever publishing agent we would like; whether it is our own personal publishing agent or some big organization that provides URN and announcement services to a large number of authors. Once we have a URN, we can provide the publishing agent with a URL for our local copy of the resource and then let it do its thing. Alternatively, we can attach a transponder to the resource, let it determine a local URL for the resource, and let it contact the publishing agent and set up the announcement process. One would expect a publishing agent to prompt us for some information as to where it should announce our new resource.

For example, we may just wish a local announcement, so that only people in our company can get a pointer to the resource. Or we may wish some sort of global announcement (but it will probably cost us a bit of money!)

Once the announcement has been made, the publishing agent will be contacted by a number of pieces of the resource location system. For example, someone running a WAIS server may decide to add the resource to their index. So they can retrieve the resource, index it, and add the indexes to their tables along with a URI - URL combination. Then when someone uses that WAIS server, it can go off and retrieve the resource if necessary. Or, the WAIS server could create a local copy of the resource; if it wished other people to find their local copy of the resource, it could provide the URI -> URL mapper with a URL for the local copy. In any case, publication becomes a simple matter.

So, where does this leave the traditional publisher? Well, there are a number of other functions which the traditional publisher provides in addition to distribution. There are editorial services, layout and design, copyright negotiations, marketing, etc. The only part of the traditional role that this system changes is that of distributing the resource; this architecture may make it much cheaper for publishers

to distribute their wares to a much wider audience.

Although copying of resources would be possible just as it is in paper format, it might be easier to detect republication of the resource in this system, and if most people use the original URN for the resource, there may be a reduced monetary risk to the publisher.

5.2 A librarian role in this architecture

We've been in a number of discussions with librarians over the past year, and one question that we're frequently asked is "Does Peter talk this rapidly all the time?". The answer to that question is "Yes". But another question we are frequently asked is "If all these electronic resources are going to be created, supplanting books and journals, what's left for the librarians?". The answer to that is slightly more complex, but just as straightforward. Librarians have excelled at obtaining resources, classifying them so that users can find them, weeding out resources that don't serve their communities, and helping users navigate the library itself. None of these roles are supplanted by this architecture. The only differences are that instead of scanning publisher's announcements for new resources their users might be interested in, they will have to scan the announcements on the net. Once they see something interesting, they can retrieve it (perhaps buying a copy just as they do now), classify it, set up a navigation system for their classification scheme, show users how to use it, and provide pointers (or actual copies) of the resource to their users. The classification and selection processes in particular are services which will be badly needed on a net with a million new 'publications' a day, and many will be willing to pay for this highly value added service.

5.3 Serving the users

This architecture allows users to see the vast collection of networked resources in ways both familiar and unfamiliar. Bookstores, record shops, and libraries can all be constructed on top of this architecture, with a number of different access methods. Specialty shops and research libraries can be easily built, and then tailored to a thousand different users. One never need worry that a book has been checked out, that a CD is out of stock, that a copy of Xenophon in the original Greek isn't available locally. In fact, a user could even engage a proxy server to translate resources into forms that her machine can use, for example to get a text version of a Postscript document although her local machine has no Postscript viewer, or to obtain a translation of a sociology paper written in Chinese.

In any case, however the system looks in three, five, or fifty years, we believe that the vision we've laid out above has the flexibility

and functionality to start tying everything together without forcing everyone to use the same access methods or to look at the net the same way. It allows new views to evolve, new resources to be created out of old, and for people to move from today to tomorrow with all the comforts of home but all the excitement of exploring a new world.

6. References

[Berners-Lee 93] Berners-Lee, T., Masinter, L., and M. McCahill, Editors, "Universal Resource Locators", RFC 1738, CERN, The Xerox Corporation, University of Minnesota, December 1994.

Deutsch, P., Master's Thesis, June 1992.
Available for anonymous FTP as
<ftp://archives.cc.mcgill.ca/pub/peterd/peterd.thesis>.

[Weider 94a] Weider, C., "Resource Transponders", RFC 1728, Bunyip Information Systems, December 1994.

[Weider 94b] Weider, C. and P. Deutsch, "Uniform Resource Names", Work in Progress.

Security Considerations

Security issues are not discussed in this memo.

7. Authors' Addresses

Chris Weider
Bunyip Information Systems, Inc.
2001 S. Huron Parkway #12
Ann Arbor, MI 48104

Phone: +1 313-971-2223
EMail: clw@bunyip.com

Peter Deutsch
Bunyip Information Systems, Inc.
310 Ste. Catherine St. West, Suite 202
Montreal, Quebec, CANADA

Phone: +1 514-875-8611
EMail: peterd@bunyip.com