Semantic-Free Referencing in Linked Distributed Systems

Hari Balakrishnan[a]
hari@lcs.mit.edu

Scott Shenker[b]
shenker@icsi.berkeley.edu

Michael Walfish[a]
mwalfish@lcs.mit.edu

[a]Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA
[b]ICSI Center for Internet Research (ICIR), Berkeley, CA

Abstract

Every distributed system that employs linking requires a Reference Resolution Service (RRS) to convert link references to locations. We argue that the Web's use of DNS for this function is a bad idea. This paper discusses the nature, design, and use of a scalable and dynamic RRS. We make two principal arguments about the nature of reference resolution: first, that there should be a *general-purpose* application-independent substrate for reference resolution, and second that the references themselves should be *unstructured* and *semantic-free*.

We observe that distributed hash tables (DHTs) provide an elegant and convenient platform for realizing these goals, and we present a general-purpose DHT-based Semantic-Free Referencing (SFR) architecture.

1 Introduction

The Web, whose links can direct readers to a vast array of remote documents, has revolutionized the nature of information dissemination by putting a global set of resources into the hands of each individual author and reader. However, the idea of using *links* to point to remote "objects" (of various types) is far more general than the Web. Links are used in a variety of distributed systems for identifying objects and invoking remote code [2, 4], for organizing data in sensornets [7], for locating devices [1], and for many other purposes where one wants to refer to objects by name, not IP address.

In general, we define a "link" as being composed of a *directive* and a *reference*. The reference tells the client where to find the *target* (*i.e.*, the object being linked to) while the directive tells the client how to process the target. For example, the Web link `<img src=http://www.mit.edu/ocw.jpg>` has a directive telling the client that the target of reference `http://www.mit.edu/ocw.jpg` is an image that should be rendered.

In order to use a link, clients need a way of resolving the reference to a location (*i.e.*, an IP address). Thus, every linked distributed system requires a *reference resolution service* (RRS). To be useful for many distributed Internet applications, an RRS should be *scalable*, performing well as the number of names and queries increases, and *dynamic*, able to accommodate fairly rapid changes in the binding between names and addresses.

This paper discusses the nature, design, and use of such a scalable and dynamic RRS. We make two principal arguments about the nature of reference resolution: first, that there should be a *general-purpose* application-independent substrate for reference resolution, and second that the references themselves should be *unstructured* and *semantic-free*.

After noting that distributed hash tables (DHTs) [5] provide an elegant and convenient platform for realizing these goals, we present a general-purpose DHT-based Semantic-Free Referencing (SFR) architecture and illustrate its use with some examples.

2 Shared and Semantic-Free

Every linked distributed application requires a scalable and dynamic method to translate references to locations. Since this is a complex problem requiring careful design and significant infrastructure, we should solve it exactly once with a shared substrate usable by all linked distributed systems.[1] Currently, no *general-purpose* RRS system exists. One might posit that Web URLs could fulfill this function. However, though the Web is wildly successful at providing its intended service, its DNS-based URLs are laden with application-specific semantics.

Today, because of these semantics, people often think of DNS names as branding mechanisms. As a result, there is tremendous legal contention for ownership of domain names [6]. Moreover, DNS's association between locations and well-known, delegated names makes it undesirable as a general-purpose RRS—while this association is merely inconvenient for Web publishers, who must acquire a DNS name before exposing content, it is clearly unnatural for linked systems in networks of sensors and devices in which administrative hierarchy and delegation are irrelevant, undesirable, or unavailable. Finally, because of their location-dependence, DNS-based Web URLs make it difficult to handle content replication, caching, and migration. Typically, new routing functionality for converting Web URLs to network locations requires baroque DNS hacks.

The argument for location-independent naming in linked systems is not new, and our position echoes the case made for Universal Resource Names (URNs) [8]. The various URN proposals advocated an architecture in which each linked system would have *its own* reference resolution service that would implement similar functionality in an application-specific manner. This contrasts with our contention that applications should use the *same* underlying method for reference resolution, and each linked system *share* the reference resolution infrastructure.

The URN proposals also assumed that achieving scalability required both hierarchy and namespace delegation. One of our goals, however, is to eliminate these for the reasons mentioned above: using semantic-free references avoids human contention and an unstructured namespace is more natural and general-purpose, given the variety of applications requiring access to this shared infrastructure.

[1] There may be cases, such as in isolated sensornets, where the use of a common Internet-based RRS infrastructure is not possible. However, even in cases where a separate infrastructure is needed, we believe that the presence of a general-purpose RRS *design* would be of significant benefit.

These aspects of the system imply that the RRS must support a "flat" namespace, making it difficult to design a hierarchical resolution method. Indeed, DNS scales very well in large part because of hierarchy based on administrative delegation, and until recently, there was no known RRS that could scale well without such hierarchy.

Fortunately, the recently developed DHT techniques offer a solution to the problem of designing a scalable, dynamic, and unstructured RRS, because they provide a general mapping between an unstructured key and a network location responsible for the key. The rest of this paper describes the design and use of a DHT-based RRS. We start by discussing the general concerns that any distributed linking infrastructure must address.

3 Link Infrastructure Components

We identify four basic, distinct concerns that must be addressed by any linking infrastructure and discuss how the Web implements them today.

Reference routing: This is fundamental to a linking infrastructure: Given a reference, the system must locate the target. Currently, the Web uses DNS to convert a URL to an IP address and then point-to-point IP unicast routing for client-server communication.

Reference integrity: The system must prevent the same reference from being used for two different intended targets,[2] a problem that becomes interesting when the set of link creators is distributed. Most previous systems have addressed this by embedding semantics in references, creating a reference namespace; the Web, for example, relies on DNS's administrative delegation for achieving reference integrity.

User-level names: Users need to be able to translate their goals (*e.g.*, "reach American Airlines") and context into an appropriate reference. In some cases, links are both generated and used by programs that require no outside context. But in many cases, the links will be exposed to human users, and for them the system must expose useful handles.

On the Web, URLs themselves are an important user-level naming mechanism; URLs are usually human-readable and occasionally memorable. As a consequence, corporations sometimes fight for

[2] It is of course possible and sometimes desirable for two different references to point to the same target.

"choice" DNS names. However, an increasingly common and reliable way for discovering document locations on the Web is to use search engines.[3]

Confidence and authentication: In most cases, users of a linking infrastructure would like some assurance that the reference they followed has taken them to the appropriate object. Formal methods can provide cryptographically-derived verification of server and content authenticity, but in many cases more informal *confidence-building* hints could satisfy users that they have reached their intended target.

Strict server (and content) authenticity on the Web is verified via certificates issued by trusted authorities. We note that the certificate infrastructure is largely independent of the Web architecture and can be used by any general link infrastructure. However, most information on the Web is not authenticated in the strict sense of the term. Instead, users often rely on informal assurances that they have found the correct reference (*e.g.*, seeing the DNS name of the content provider at the top of a browser page gives the user some confidence in the displayed data). While it is unclear to us precisely how important this notion of "confidence" is, it appears to provide something useful to Web users.

We note that the Web, instead of cleanly separating the four necessary features described above, uses DNS and the name registries to implement each of them: Reference routing occurs via DNS's name-to-IP address translation; DNS name registration helps achieve reference integrity; and some aspects of both the user-level naming and confidence features derive from the readability of DNS names.

As we described in Section 2, the human friendliness of DNS-based URLs, as well as its static associations between names and locations, results in consequences that are undesirable for a generic linking infrastructure. The next section details our proposal for a system free of those problems.

4 SFR Architecture

This section outlines our proposed SFR architecture in terms of the four link components. We emphasize that there are several open problems in each part. In designing the architecture to support semantic-free references, we seek to implement the four components as independently as possible (in contrast to the Web's overloading of DNS with responsibilities for which it was not intended and for which it is not well-suited). This gives us the benefits of modularity: tailoring components to their tasks, permitting different versions of components for different applications, and freeing applications to use only those components necessary for them.[4]

Only one of the components—reference routing—is *essential*, and it forms the central piece of our proposed SFR architecture. Traversing links requires clients to locate remote targets, and so all distributed applications need this functionality. Because our solution is free of application-level semantics, we expect global sharing of this component by *all* SFR applications.

To implement scalable **reference routing**, the SFR architecture uses DHTs, which provide a mapping between a 160 bit reference, which we call an *SFRTag*, or simply—to emphasize its semantic-free nature—a *tag*, and an application-defined object-record (`O-record`). Our intent is for the `O-record` to provide a general-purpose scalable object location method that: (1) allows applications to embed semantics into objects while being oblivious to those semantics and (2) handles object replication, object mobility and object updates. The `O-record` has the following fields:

```
class O-record {
   UniqueID SFRTag;    // 160-bit id
   IP:Port Location;   // current locn.
   ObjectInfo O-info; // app-specific
 }
```

When the application creates an object, it also creates a new `O-record` and inserts it into the reference routing infrastructure (*i.e.*, the DHT) by calling `SFR_Insert(SFRTag,O-record)`. (We address the issue of how `SFRTag` is generated later in this section.) In addition to the network location of the object, the `O-record` contains object information, defined and consumed only by the application. Examples of this information include the object's name in the application's namespace, the retrieval method for this object (*e.g.*, HTTP, SMTP),

[3] In fact, some say Google has supplanted URLs as the user-level naming method of choice [9].

[4] For example, a system without human involvement does not need a user-level naming component.

and a timestamp indicating last-modified-time. The `O-info` could even contain code that the searching client can use to retrieve the object. It is important to note that, although the `O-info` can be quite sophisticated, the reference routing machinery remains general-purpose because it operates below the application's namespace.

Once an `O-record` is in the SFR infrastructure, users who wish to retrieve the corresponding object first call `SFR_Lookup(SFRTag)`. (The user-level naming component, which we discuss below, will allow the user or the user's application to obtain the `SFRTag` corresponding to her particular goals.) The infrastructure then returns the `O-record` of the object. The `O-record`'s `Location` and `O-info` fields together have enough information for the searching client to retrieve the object. In some cases, the contents of the `O-info` field may signal to the client that object retrieval is unnecessary; a timestamp field, for example, would allow the client application to decide if a cached copy is current enough.

While many of the DHT proposals were in the context of pure P2P systems on arbitrary hosts, we envision a decentralized but managed collection of machines in different administrative domains, somewhat akin to the DNS infrastructure, with reasonable stability and trustworthiness. Many groups are working to make such a DHT-based system a core component of the Internet's future infrastructure, and there are many open research questions (such as caching, replication, and security) that require further work. We don't address those research questions here and instead presume the presence of a working DHT infrastructure in our design.

We note that by using a DHT to resolve references, we are losing some of the *fate-sharing* that exists in current RRSs, such as DNS. Currently, with DNS, if an administrative domain is partitioned from the network, individuals in that administrative domain can often resolve references local to that domain (*e.g.*, they can successfully browse internal Web sites). Providing a form of fate-sharing within the SFR insfrastructure is an area of ongoing investigation for us.

Reference integrity must be ensured by the application that creates a new object and inserts the corresponding SFRTag. Generation of unique tags is completely up to applications (which are free to employ a variety of techniques, including choosing random tags or hashing the contents of the `O-record`). The important points here are twofold: first, whatever method the application chooses, the SFR infrastructure will attempt to prevent the same tag from being inserted twice (updates and deletions are of course permitted). Second, the application can *check* whether the candidate tag has already been claimed (by calling `SFR_Lookup()` on the candidate) and then *reserve* its candidate. Because the namespace is massive, one need not worry that an adversarial application or user could preemptively reserve a significant chunk of potential identifiers. If an adversary reserved one new identifier each nanosecond for ten years, this would result in approximately 2^{58} identifiers being reserved, which, assuming the identifiers are 160 bits, represents a tiny fraction—only $1/2^{102}$—of the entire namespace.

We envision that various directory services will provide mappings between **user-level meanings** and the associated tags. Just as search engines now provide a mapping between keywords and URLs, in the future we envision that a plethora of directory services, each with its own intended audience and its own economic model, will provide a similar mapping service between keywords and other descriptions into tags. Far from being an unfortunate consequence of the semantic-free restriction, we believe that directory services are a more robust, extensible, reliable, and convenient method of advertising resources than non-permanent, forgettable URLs.

It is important to note that these directory services are not part of the core SFR architecture. Furthermore, their listings would not be decided at standards meetings or by official bodies. Some directory services could be funded by user contributions and would be designed to offer responses closest to the user's intent. Other directories could be funded by businesses seeking to have their name associated with certain keywords (payments would assure that a given airline would be one of the first items on the list of responses to a search for airlines, for example). Universities and other research non-profits could band together to offer directory services providing reliable listings in the research arena.

There are two aspects to **confidence and authentication**: first, only the "owner" or other authorized party should be allowed to modify a particular `O-`

`record`. One way to implement this is to permit the entity inserting `O-records` to present a public key along with its `SFR_Insert()` request. If the entity presents a public key, then the SFR infrastructure ensures that the insert message and all modification messages have been signed with the corresponding private key. We note that the tag routing infrastructure should offer this level of authentication to applications, but it should not require them to take advantage of it.[5]

The second aspect to confidence and authentication is that clients (or their end-systems) should be able to verify the authenticity of references, and for this function, we envision different services using different mechanisms. Some services would use a trusted certificate authority and certificate chains much like the Web certificate infrastructure does today. However, the design of more informal confidence-building measures in an SFR infrastructure is an interesting problem, since references no longer convey semantic information to human users. For certain applications like the Web, a reasonable approach might be to rely on information provided by a search engine (such as a cryptographic hash) to validate content. We note that our proposal neither precludes any of these approaches nor mandates an authentication scheme.

5 Using Semantic-Free Links

The previous section's SFR architecture overview presented a preliminary design that, with many details to be fleshed out, will be the subject of future work. To give a more concrete idea of how the SFR architecture might be used in practice, we now describe two possible applications.

5.1 Web

One possible SFR-based redesign of the Web would be to use search engines as essentially the only way of reaching targets. This elevates search engines, already important, to a critical position in how documents are accessed. Search engines will continue to index documents by their content but will associate documents with tags, instead of URLs. Since the reference in the link exposes no semantics, showing the corresponding bitstring when the user hovers over a link isn't particularly useful. We propose to handle this user-interface issue by instead displaying the meta-data corresponding to the target.[6]

In this SFR-based Web, two traditionally difficult issues—content replication and mobile Web servers—are handled without significant additional effort or specialized infrastructure because of the layer of indirection provided by the DHT. To handle replication, a tag (mapping to a DHT key) would have multiple values corresponding to the different nodes that contain some content. To handle server mobility, any change in the IP address/port of a target can be implemented using an `O-record` update mechanism.

Authentication is separate from link naming and would use a trusted certificate infrastructure as the Web does today. Search engines that produce the tags would give some degree of confidence, but exposing additional confidence-building hints is a trickier issue; one might envision a scheme in which each document has meta-data descriptions that are authenticated using signatures.

5.2 Directory services

Directory services are an important primitive in many distributed systems, since they allow participants to rendezvous with each other (*e.g.*, clients with servers). Invocations to a directory service often take the form of a remote procedure call (RPC) in a program—for instance, a program that has a link requiring email to be sent might invoke a DNS `gethostbyname`, or discovering a nearby color printer might require a suitable call to INS [1] or a SOAP call encoded in XML syntax [2]. The problem of finding suitable nodes that can answer an RPC is common to these applications, and can be implemented using SFR. Recent work has shown how this can be done in DNS [3] and in INS, using mappings between intentional names and SFRTags [1]. The same approach can be adapted to SOAP RPC, eliminating the need for HTTP encodings of SOAP.

[5] It is possible that for certain kinds of objects, the entity inserting an `O-record` would want any other user to be able to update the contents of the `O-record` without bothering with public and private keys.

[6] We emphasize that there are certainly many other interface questions to be worked through in this proposal for the Web; for example, we will have to develop a user-interface to HTML composition that would allow content providers to embed tags in their document while protecting them from unwieldy bitstrings.

6 Discussion

DNS is an excellent system for identifying static nodes in the Internet, and, at the time the Web was invented, it was also the best choice for reference routing. However, it has been over a decade since the Web was first deployed, and a system that statically maps well-known names to locations is not a good choice for the key component of a general linked infrastructure. We believe that DNS should continue to perform its original function but that there are now three reasons to reconsider the design of reference resolution systems: (1) the importance of links as a general concept in distributed systems is better understood, (2) we are more familiar with the weaknesses of embedding semantics in references, and (3) DHTs promise scalable, dynamic reference routing. Our thesis is that a general-purpose linking infrastructure will empower a variety of distributed services, and that *semantic-free* reference routing is the organizing principle for designing it.

We conclude with an analogy between virtual memory in traditional operating systems and our proposal. Most programming languages today support typed objects that have references to other objects. The compiler for the language converts all references to virtual (rather than physical) addresses, which are in turn translated to physical addresses by a language-independent OS and memory management unit.

Now consider a typical distributed linked application. It has objects, which have references to other objects, defined in an application-specific syntax. However, despite the ubiquity of remote linking, there is currently *no* notion of virtual addressing for linked applications.

Our SFR proposal aims to fill this void. For linked distributed applications it provides a general virtual-to-physical translation that allows application-specific links to be layered over the abstraction. We believe that an SFR infrastructure is timely, viable, and worthy of further research.

Acknowledgments

We thank Frank Dabek, Mark Handley, and Karen Sollins for useful discussions. We thank the anonymous reviewers for their comments.

This research was conducted as part of the IRIS project (`http://project-iris.net/`), supported by the National Science Foundation under Cooperative Agreement No. ANI-0225660.

References

[1] BALAZINSKA, M., BALAKRISHNAN, H., AND KARGER, D. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In *Proc. International Conf. on Pervasive Computing* (Zurich, Switzerland, Aug. 2002).

[2] BOX, D., ET AL. Simple Object Access Protocol (SOAP) 1.1. `http://www.w3.org/TR/SOAP`, May 2000. W3C Note.

[3] COX, R., MUTHITACHAROEN, A., AND MORRIS, R. Serving DNS using a Peer-to-Peer Lookup Service. In *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)* (Cambridge, MA, Mar. 2002).

[4] HORSTMANN, M., AND KIRTLAND, M. DCOM Architecture. `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom%/html/msdn_dcomarch.asp`, July 1997.

[5] Infrastructure for Resilient Internet Systems. `http://www.project-iris.net/`, 2002.

[6] MUELLER, M. *Ruling the Root: Internet Governance and the Taming of Cyberspace*. MIT Press, Cambridge, MA, May 2002.

[7] SHENKER, S., RATNASAMY, S., KARP, B., GOVINDAN, R., AND ESTRIN, D. Data-Centric Storage in Sensornets. In *Proc. Hotnets-I* (Oct. 2002).

[8] SOLLINS, K. Architectural Principles of Uniform Resource Name Resolution, Jan 1998. RFC 2276.

[9] WIGGINS, R. The Effects of September 11 on the Leading Search Engine. *First Monday: Peer-Reviewed Journal on the Internet 7*, 10 (Oct. 2001). Available from `http://www.firstmonday.org/issues/issue6_10/wiggins/index.html`.