Using DHTs to Untangle the Web from DNS

Michael Walfish*
MIT Laboratory for Computer Science, mwalfish@lcs.mit.edu

1 Introduction

The marriage between DNS and the Web, while initially fruitful, is now seen by many as a mutually unhealthy union. DNS's original goal was practical and limited: allowing users to refer to machines with convenient mnemonics. As such, it has performed admirably. However, with the advent of the Web and the resulting commercial value of DNS names, profit has replaced practicality as the dominant force shaping DNS. Legal wrangling over domain ownership is commonplace, and the institutional framework governing the naming system (*i.e.*, ICANN) is in disarray. Commercial realities have transformed DNS into a branding mechanism, a task for which it is ill-suited.

A linked, distributed system such as the Web requires a Reference Resolution Service (RRS) to map from references, such as Web URLs, to actual locations, such as the IP address(es) where the referenced object is stored. In the current Web, references are URLs with a hostname/pathname structure and DNS serves as the RRS. The host-based nature of URLs—which ties references to specific locations and hard-codes the path component of the reference—makes content replication and movement hard.[1] Thus, the Web would be better served with a new RRS, one that does not embed location into the reference itself. The experiences with DNS suggest three basic design principles for any such RRS, which we now articulate.

Location-independent references: References should not contain information about the location of the content and should easily generalize to the case of widely replicated content. Along with many others, we argue that a reference, like any abstraction used for indirection, should remain constant as the referenced object moves. With current Web URLs, however, hyperlinks break when the referenced content migrates since the hostname and/or the directory path to the object changes, and no mechanism exists for updating referring hyperlinks.

Contention-free references: There is contention over domain names because they are used as the basis for URLs. Social problems include "name squatting", "typo squatting", and lawsuits over trademark infringement [6]. Moreover, there are many examples of misleading DNS-based URLs, such as `http://www.martinlutherking.org`, a Web site that, contrary to expectation, is decidedly hostile to Martin Luther King, Jr. All of these tussles result from DNS names having become branding elements. Rather than attempt to solve the problem of human contention over names (which is fundamentally insoluble, as years of trademark law attest), we are pushing the problem to another layer, one in which multiple, coexistent, competing solutions to the problem of names can exist. By making references human-unfriendly, we shield the core object location service from tussles, allowing it to focus on the technical goals of efficiency and reliability.

Minimal design: The use of links, or pointers, to refer to objects or content on other machines is not unique to the Web. Links are used in a variety of distributed systems for identifying objects and invoking remote code, for organizing data in sensornets, for locating devices, and for other purposes where one wants to refer to objects by name, not location. All of these systems require an RRS. Since reference resolution is a hard problem that requires delicate design and significant infrastructure, a sensible approach is providing a single RRS that can be shared among these various linked systems. However, to ensure the RRS can be shared, it should provide a minimal interface that does not impose any application semantics.

The purpose of this paper is to describe our design and implementation of a DHT-based RRS, called Semantic Free Referencing (SFR), that satisfies these three requirements, thereby providing a better RRS for the Web than DNS does currently. We will show that SFR permits functionality impossible in today's Web and that what at first appear to be insurmountable challenges for SFR—performance, fate sharing, security, and convenient naming—actually have reasonable solutions.

Of course, a natural question is: if SFR is used as an RRS for the Web and if references are human-unfriendly, how will humans retrieve content? The answer is, first, that today humans mostly use what we call *user-level names*, which translate user goals (*e.g.*, "find the website of Cornell University") into an object on the Web. Examples of user-level names include queries to search engines, AOL keywords, hyperlinks in documents, saved bookmarks, and links sent through e-mail. Each of these methods works well under SFR: search queries, for example, return candidate SFR references instead of DNS-based URLs.

Second, the other way users access Web content today is by typing DNS-based URLs. In this case, DNS functions as a *canonicalization service*, *i.e.*, a mapping from well-known, invariant, human-readable names to Web objects. To permit equivalent functionality under SFR, DNS need not be the canonicalization source. Moreover, it might be desirable to have several such mapping services. So, rather than do what DNS-based Web URLs do—conflating the *reference*, which actually performs object routing, and the *user-level name*, which allows people to find what they are looking for—SFR separates the two functions and focuses on getting the object routing function right.[2]

*This is joint work with Hari Balakrishnan (MIT Laboratory for Computer Science, Cambridge, MA) and Scott Shenker (International Computer Science Institute (ICSI), Berkeley CA). This research is part of the IRIS project and supported by the National Science Foundation under Cooperative Agreement No. ANI-0225660

[1] One can certainly play DNS tricks to achieve replication; our point is that the DNS architecture hinders rather than helps replication.

[2] We address the question of bootstrapping in Section 4.4.

2 RRS Design Choices

There are three current proposals for a general-purpose RRS: DNS URLs, IP-based URLs, and URNs. However, none of these fulfills each of our three design goals. IP-based URLs would obviously be location-dependent. Current DNS-based URLs are neither human unfriendly, which is obvious, nor location-independent. A DNS-based URL ties an object to a specific administrative domain and hard-codes a path. If a piece of content moves and/or its path component changes, existing hyperlinks are invalidated, leading to messages like "Please update the referrer of this page"; this model violates the principle that someone should be able to maintain a pointer without knowing specifics of the pointer's target. The URN proposals, in contrast, are motivated by the goal of producing an RRS with persistent references that are independent of network location and current administrative domain, but in this model each linked system would have its own reference resolution system [8, 9].

Finally, one could also use an approach similar to the SFS framework [2]; hyperlinks would be links in SFS space to files on other machines. In this model, the references would contain hashes of public keys and so would not be human readable. But this approach fails our goal of full location-independence: these links in SFS space would still contain hard-coded paths, just as DNS-based URLs do. Moving a file to another administrative domain or changing the location of a file within a directory structure would require tedious and error-prone maintenance of symbolic links.

3 SFR Design and Implementation

Our definition of a *semantic-free* system for referencing is one in which references are *location-independent and human unreadable*; a semantic-free system would therefore satisfy at least two of our design principles. Turning now to the third principle, a shared, general purpose infrastructure that could be used in any number of systems should not make any assumptions about the structure of the namespace or about whether hierarchy is available. The three goals together, then, naturally lead to a flat, opaque, human unreadable namespace for references.

DHT technology [5] is perfectly suited to this task: at their core, DHTs map an unstructured key to a network location responsible for the key. This, then, is the SFR system: an application-independent, DHT-based object resolution infrastructure in which the object references, which we call tags, are unstructured, semantic-free keys in a DHT; lookups on the tag return object meta-data.

3.1 SFR Infrastructure

Our implementation of SFR associates a semantic-free 160 bit string, an `SFRTag`, to a simple data structure, the `o-record`. The `o-record` contains the object's meta-data and serves as a layer of indirection between the reference itself and the actual meta-data, such as the location of the object. The `SFRTag`s are keys in a DHT, and the data that the DHT associates with the keys is exactly the `o-record`. Note that it is only meta-data, and not objects themselves, that the DHT stores. Our implementation currently uses `chord/dhash` [10] as the underlying DHT, but the SFR architecture is modular and permits another DHT to be substituted with little effort.

Figure 1: The `o-record`

The SFR infrastructure provides application-independent semantics because, with the exception of the `location` field (a list of IP address and port pairs in our implementation), the `o-record` itself is application-independent. The `o-record` is illustrated in Figure 1. The application using the SFR infrastructure chooses the content and number of elements in each object's `location` field. Also under the application's control are the structure, content, and length of the `oinfo` field.

The interface to the SFR infrastructure is an API that allows an application to set `o-record` information as well as retrieve `o-record`s that are stored in the infrastructure. The infrastructure prevents collisions by rejecting insert requests and notifying clients if the `SFRTag` they request is already present. To prevent other clients from modifying their `o-record`s, entities can present a public key with their insert request. Our implementation of the SFR infrastructure stores this key and ensures that the current request and all subsequent modifications to the `o-record` have been signed with the corresponding private key.[3]

The SFR infrastructure also prevents an attack when there is a partition; the attack occurs if two malefactors on opposite sides of the partition conspire to eavesdrop on a legitimate request and then insert the corresponding `SFRTag` into the SFR infrastructure on the opposite side of the network partition, leading to the possible expulsion of the legitimate content when the partition heals. To prevent this attack, the infrastructure forces random key choice by requiring that the `SFRTag`, when first created, is a secure hash of the client's public key and a client-chosen nonce. Updating the public key does not invalidate the reference since the SFR infrastructure only requires that the relationship between the `SFRTag` and the public key is satisfied when the tag is *first* inserted. After that, the SFR infrastructure ensures that updates to the public key or to the content have been signed with the existing public key. We note that this scheme also has the benefit of preventing human readability.[4]

We observe that a deployed system would use infrastructure nodes running the SFR and DHT software, not "peers" behind cable modems. We do not claim to know this infrastructure's economic model—it could be centrally financed or "donated", like DNS.[5] Our current design presumes that clients trust the infrastructure; in Section 4.3 we consider a different trust model, namely using a system like SkipNet as the underlying DHT [4].

Although SFR is designed to be useful for any application requiring reference resolution, the remainder of this paper primarily focuses on our implementation of one application: the

[3] We plan soon to implement a facility by which entities can update the public key associated with an `o-record` by supplying a password.

[4] If the user does not supply a public key, then the infrastructure can enforce that the `SFRTag` is the hash of something, yielding randomness in tag choice but no protection against overwriting.

[5] How one organizes and finances a DHT infrastructure is an important open question but not one we address here.

Web over SFR.

3.2 The Web over SFR

After first describing our design and implementation, we then show how functionality that is difficult or impossible with DNS-based URLs is easily available in a version of the Web that uses the SFR infrastructure.

Because both the administrative domain (*i.e.*, the DNS name) and the path on the Web server are hard-wired into today's URLs, it is difficult for content providers to change the logical path component of the URL or to move objects from one administrative domain to another without breaking the existing hyperlinks that use this URL. Today, if a Web object changes location but stays within its domain, a Web server can, with difficulty, accommodate this movement by continuing to accept URLs with path components that do not correspond to the file system on the Web server. However, this requires extra configuration effort on the part of the content provider and is not the norm. Moreover, it requires administrative control over the Web server, which is not something that normal users have. If an object moves from one administrative domain to another, HTTP redirects are necessary to avoid broken links, but redirects are not only clumsy, they require the cooperation of the originating Web server.

As a result, in the current Web, when an object moves, hyperlinks generally break, and a human browsing the Web reads "please update your links" or "please notify the referrer of this page". The idea that pointers have to change when objects move violates the principle that a reference should be independent of the current location of the item being pointed to.

With the Web over SFR, in contrast, we take advantage of the location-independent routing given by the SFR infrastructure and create a fully general layer of indirection for Web objects. We use an `SFRTag` as an abstract reference to a Web object. The `location` field in the `o-record` associated to the `SFRTag` indicates the current IP address of a Web server that can fulfill HTTP requests for the object. And the `oinfo` field of the `o-record` indicates the logical path on the Web server. In the Web over SFR, URLs have the following form:

```
sfr://fbcd1234abc.../<optional path>
```

To resolve a URL like this one, a Web browser[6] strips out the first portion of the URL, treats it as the `SFRTag` and submits it to the SFR infrastructure. The browser then receives the matching `o-record` from the SFR infrastructure and submits a standard HTTP request to the IP address and port in the `o-record`'s `location` field. The path that goes in the HTTP request is the contents of the `oinfo` field concatenated with the `<optional path>` from the original URL.

This design preserves the semantics of HTTP. From Web servers' perspective, the SFR infrastructure is invisible; Web servers continue to receive HTTP GET requests with paths. We also note that this framework naturally allows dynamic content because embedded links can contain semantic-ful paths. Without the `<optional path>`, Web browsers would be unable to form and submit URLs that had never existed before (since there would be no way to get an `SFRTag` to resolve to something that had never been inserted into the SFR infrastructure).

[6] A proxy server performs this function in our implementation.

3.3 Features

3.3.1 Robust Linking

In contrast to the current method of Web linking, the SFR approach provides a layer of indirection to abstract not only the machine location (as occurs in DNS) but also the logical path on the Web server. This permits a general mobility solution: if a piece of content, currently referenced by an `SFRTag`, moves to another Web server at a different path, the content provider need only change the `location` and `oinfo` fields in the `o-record` in order to permit the correct reference resolution to occur for Web clients. Web pages linking to the object can continue to maintain the same references.

This approach is flexible about how much of the reference functions as an abstraction. An `SFRTag` can refer to a machine (in this case, the `<optional path>` is the same as it is with today's URLs), to a Web object (in this case, the `SFRTag` abstracts the entire URL and the `<optional path>` is empty), or to a directory structure (in this case, the `SFRTag` abstracts the entire URL up until the root of the directory structure, and the `<optional path>` is everything underneath the directory structure). For example, a researcher might have a large collection of publications in one directory and might wish to abstract only the location of the collection. In this case, the `SFRTag` would abstract the IP address of the Web server as well as the path on the server up until the document collection. The individual publications would retain the same names. So the SFR Web references would have the following form:

```
sfr://fbcd1234/pub1.ps
sfr://fbcd1234/pub2.ps
```

If the researcher's affiliation then changes, he alters the `o-record` corresponding to `fbcd1234` so that it contains a different Web server and a different path on the new server. A referring Web page that embeds `sfr://fbcd1234/pub1.ps` can safely be ignorant of the move.

3.3.2 Democratic Replication

The Web over SFR also permits functionality that is inaccessible to ordinary Internet users with today's DNS-based URLs. As several content distribution networks have demonstrated, it is possible to implement replication solutions by using today's DNS [1]. But this requires a massive worldwide network and a custom DNS implementation; most Internet users cannot afford to implement, or even be customers of, a system of this scale. With our version of the Web on SFR, however, individuals can receive replication services by agreeing to host each other's content. We call this kind of replication *democratic replication*, and it works very simply.

Individuals reference their content with `SFRTag`s and place into the corresponding `o-record` multiple locations and paths. Each (location,path) pair represents a valid location of the Web object. One of the pairs is the source of the content, and the other pairs represent replicated versions on other Web servers. In the simplest model, users pair up with friends and mirror each other's websites with `rsync`. Each friend communicates to the other the IP address of a Web server and path on that Web server where she is storing her friend's content, and each friend then updates her own `o-record` with this information. Now, clients doing a lookup receive an `o-record` with multiple locations

and choose which Web server to contact via standard server selection methods.

Implementing this functionality in today's Web would be impossible for ordinary users (and very inconvenient for administrators). In the current Web, the only way to refer to content on two different Web sites is to use an abstract domain name (and either hack the DNS or statically configure the abstract domain name to return the IP addresses of the two servers.) This hypothetical URL has to work on two different Web servers with two different directory structures, meaning the Web servers would have to be configured to recognize both the abstract domain name and the path component of the hypothetical URL. This configuration would require administrative control over the Web servers. A benefit of the SFR approach is that ordinary users need control only, say, their home directory; they then communicate a path under their home directory to the individual for whom they are replicating.

3.3.3 Extensibility

Because of the `o-record`'s simplicity and generality, the Web over SFR (and, indeed, any application built on SFR) can easily accommodate new features. For example, the `oinfo` field could additionally store a certificate indicating that a site is safe for children [7]. After a user clicked on an SFR hyperlink pointing to the site, the Web browser would retrieve the `o-record` and verify that the `o-record` contained a certificate indicating the site was safe.

4 Challenges

Because of DNS's hierarchical, delegated structure, there are a number of features that it naturally provides that might at first appear difficult for SFR. In this section, we show how SFR can address these issues.

4.1 Performance

Although lookups in Chord, the substrate in our implementation, use $O(\log n)$ hops, we can in practice reduce the number of overlay hops per lookup to one or two, making the latency associated with an SFR lookup roughly equivalent to a DNS lookup. We simply expand Chord's location cache. Indeed, our experiments and simulations show that a relatively small location cache (with room for approximately 20% of the nodes in a Chord ring) usually yields one or two lookups. The reason for this is as follows: if the originating node, O, has not cached the target T, O is likely to know about a node, N, *near* T, and N is likely to know about T. In the domain of caching for DHTs, "being close counts". Caching 20% of the nodes in the ring is reasonable because a deployed SFR infrastructure would be bounded.[7]

This bound also permits us to use a substrate specifically optimized for one hop lookups. Gupta *et al.* prove that by exchanging state in the background, any DHT with a ring structure (such as Chord) can guarantee one hop lookups almost all the time [3]. Sufficient conditions for this scheme are enough bandwidth between DHT nodes and a bound (well above what we expect for SFR) on the DHT size. We plan to use an implementation of this scheme in the future.

[7] An over-estimate of the number of SFR nodes is the number of DNS servers, which is on the order of 10^5.

4.2 Fate Sharing

By delegating the namespace, DNS naturally permits a feature that is highly desirable in an RRS: *fate sharing*. For example, if the computers inside `cornell.edu` become disconnected from the rest of the Internet, those machines can continue to resolve names ending in `cornell.edu`, since the authoritative name server for `cornell.edu` is on their side of the partition.

At first blush, it seems difficult for SFR to provide this feature because references, and their associated meta-data, are stored on random nodes in the SFR infrastructure. However, the solution is not complicated. In the current implementation of SFR, content providers inserting `o-record`s do so by contacting a nearby member of the infrastructure running `sfr_server` software; the purpose of this module is to abstract the underlying DHT. To achieve fate sharing and scoping, organizations would set up proxy `sfr_server`s. This proxy would appear as an `sfr_server` to hosts within the organization and as a standard SFR client to the global SFR infrastructure.

Clients wishing to insert `o-record`s would submit a standard SFR insert request to the proxy. The proxy would store the associated `o-record` and would then insert the `o-record` into the global DHT on behalf of the client. This approach clearly results in fate sharing: if a disconnect occurs, the proxy continues to serve `o-record` data for content that originated inside the organization.

The proxy could also be enhanced by making it mirror the global DHT, *i.e.*, by making it into a cache. This need not require a large investment in infrastructure or machines, since the mirror could be a DHT consisting of one node. Finally, a natural solution for scoping arises in this framework: when clients within the organization wish to limit their meta-data to members of their organization, the proxy (or mirror) simply stores the `o-record` locally, without forwarding it on to the global DHT. Implementing a proxy that provides fate sharing, caching and scoping is the subject of future work.

4.3 Security and Management

We discussed in Section 3 how SFR ensures the integrity of `o-record`s. SFR also has to worry about misbehaving nodes; this is not a problem in DNS because content providers are responsible for their own meta-data. For now, in SFR, our implementation assumes that it is running on trusted nodes that behave (though, of course, it assumes nothing about the intent of the clients of the infrastructure).

We emphasize that SFR's current assumption of a well-behaved underlying DHT could be relaxed by changing the substrate from Chord to a DHT like SkipNet [4] that is aware of administrative boundaries between organizations.[8] In this model, the content provider would have substantial control over where his `o-record` is stored, eliminating potential security issues.

A DHT like SkipNet also quite obviously yields an alternate solution for fate-sharing, since objects' meta-data would be stored close to the objects being referenced. As an aside, we note that either a centrally managed infrastructure or a SkipNet-like DHT incorporating administrative domain addresses problems from loading attacks and hotspots. For now, the implementation of SFR ensures the integrity of `o-record`s and protects

[8] Although SkipNet proposes DNS names to identify administrative regions, this choice is arbitrary and could instead use different strings.

against misbehaving clients, but it does not address the problem of misbehaving DHT nodes.

4.4 Canonicalization

Right now, the benefit that DNS gives is associating to Web content a set of occasionally memorable and usually transcribable handles. From our informal observations, this mnemonic benefit is not crucial to human users, but it is sometimes extremely convenient to be able to type in a Web browser `www.cnn.com` if one is looking for CNN's home page. In this case, DNS functions as a *canonicalization service*—it provides a name that is persistent, well-known, and sometimes memorable.

Although SFR does not explicitly incorporate a canonicalization service, it does permit any number of such services to coexist. If SFR becomes popular, there is every reason to believe that Web service providers with the appropriate expertise would compete to provide such a service. Instead of offering detailed conjecture about the design of canonicalization services, we note 1) that there are two obvious models already in existence: AOL keywords, and the paper yellow pages and 2) that SFR is modular enough to permit a wide spectrum of services.

Of course, the problem of bootstrapping exists, namely how users get pointers to directory services. There are a number of possibilities here, ranging from browsers shipping with pointers, to links sent in e-mail from friends, to applications running on the local host that populate a local database of useful sites.

4.5 Confidence

DNS provides another convenience for the Web that might appear difficult under SFR: human-level confidence in the authenticity of content. For example, humans browsing the Web who see a URL beginning with `www.nytimes.com` are confident that the content they are viewing is owned by the newspaper they think of as *The New York Times*. This level of confidence is actually quite weak; it entirely depends on whether the "correct" company owns a given domain name. In many cases, such as in the earlier example of `www.martinlutherking.org`, this "correctness" is not only hard to achieve, it is also ill-defined. For example, there are many different businesses in the same industry with the same name.

Under SFR, there are a number of ways to achieve this kind of human-level confidence in content. Rather than give lengthy detail on possible solutions, we outline just one: hyperlinks on Web pages optionally embed a `taginfo` object alongside the `SFRTag`. This object contains cryptographic statements of the form "Entity E says that this tag is CNN", where E is a Web service provider that users trust. Users' browsers would inform them about who is certifying the link. We plan to build strawman prototypes for the Confidence and Canonicalization services.

4.6 Transition Strategy

The goal of SFR is not to provide equivalent functionality to DNS, which users ought to continue to use for its original purpose of hostname translation. The goal is rather to provide a more attractive alternative for the subclass of applications, like Web referencing, that require an RRS.

If SFR becomes popular and widely deployed, client machines will need to discover a reachable SFR server. Clients today find out about available DNS servers via DHCP or via hardcoding; we envision an identical process for informing clients of SFR servers. Providing client access to an SFR server would be one of the services offered by an ISP or large institution.

5 Conclusion

In this paper, we have knowingly adopted an extreme view, namely that all references should be devoid of human-readable semantics. It is entirely possible that the referencing system of the future will be either somewhere in between DNS and SFR or a combination of DNS and SFR. Indeed, from a usability perspective, each system offers something the other does not. DNS offers convenience publicizing and composing content while SFR makes it easy to implement mobility, democratic replication and traditional replication and to avoid broken links.

Ultimately, our goal is to achieve the full potential of the Web as a medium in which anyone can publish, in which objects can freely migrate, and for which the infrastructure is simple, cheap, robust, and accessible. Attaining these characteristics, we believe, would be easy and natural under SFR.

6 Project Status

Our implementation of the SFR infrastructure is complete. It contains: an SFR client library (which must be pointed to an available `sfr_server`), the `sfr_server` described above, and a modified version of `dhash/chord`. The Web over SFR uses the client SFR library and permits robust linking, democratic replication and the security benefits we have outlined. Users access the Web over SFR by pointing Web browsers to a custom proxy Web server that uses SFR to handle HTTP requests like `http://fa12213ba123/`. We have also studied caching performance in simulation. Our future work is building a proxy `sfr_server` to address fate sharing, incorporating solutions for caching (including the one hop DHT techniques [3]), building prototypes for canonicalization and confidence services, and using the `o-record` to implement extensions to current Web functionality.

References

[1] Akamai Technologies, Inc. `http://www.akamai.com`.

[2] K. Fu, M. F. Kaashoek, and D. Mazières. Fast and secure distributed read-only file system. *ACM Transactions on Computer Systems*, 20(1):1–24, Feb. 2002.

[3] A. Gupta, B. Liskov, and R. Rodrigues. One hop lookups for peer-to-peer overlays. In *Proc. of 9th Workshop on Hot Topics in Operating Systems (HotOS IX)*, Lihue, Hawaii, May 2003.

[4] N. J. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, Seattle, WA, March 2003.

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

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

[7] S. Savage. Personal communication, Feb 2003.

[8] K. Sollins. Architectural principles of uniform resource name resolution, Jan 1998. RFC 2276.

[9] K. Sollins and L. Masinter. Functional requirements for uniform resource names, Dec 1994. RFC 1737.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. ACM SIGCOMM*, San Diego, CA, Aug. 2001.