Network Working Group
Request for Comments: 2276
Category: Informational

K. Sollins
MIT/LCS

January 1998

Architectural Principles of Uniform Resource Name Resolution

Status of this Memo

Copyright Notice

Abstract

This document addresses the issues of the discovery of URN (Uniform Resource Name) resolver services that in turn will directly translate URNs into URLs (Uniform Resource Locators) and URCs (Uniform Resource Characteristics). The document falls into three major parts, the assumptions underlying the work, the guidelines in order to be a viable Resolver Discovery Service or RDS, and a framework for designing RDSs. The guidelines fall into three principle areas: evolvability, usability, and security and privacy. An RDS that is compliant with the framework will not necessarily be compliant with the guidelines. Compliance with the guidelines will need to be validated separately.

Table of Contents

1. Introduction

The purpose of this document is to lay out the engineering criteria for what we will call here a Resolver Discovery Service (RDS), a service to help in the learning about URN (Uniform Resource Name) resolvers. The term "resolver" is used in this document to indicate a service that translates URNs to URLs (Uniform Resource Locators) or URCs (Uniform Resource Characteristics). Some resolvers may provide direct access to resources as well. An RDS helps in finding a resolver to contact for further resolution. It is worth noting that some RDS designs may also incorporate resolver functionality. This function of URN resolution is a component of the realization of an information infrastructure. In the case of this work, that infrastructure is to be available, "in the Internet" or globally, and hence the solutions to the problems we are addressing must be globally scalable. In this document, we are focussing specifically on the design of RDS schemes.

The Uniform Resource Identifier Working Group defined a naming architecture, as demonstrated in a series of three RFCs 1736 [1], 1737 [2], and 1738 [3]. Although several further documents are needed to complete the description of that architecture, it incorporates three core functions often associated with "naming": identification, location, and mnemonics or semantics. By location, we mean fully-qualified Domain Names or IP addresses, possibly extended with TCP ports and/or local identifiers, such as pathnames. Names may provide the ability to distinguish one resource from another, by distinguishing their "names". Names may help to provide access to a resource by including "location" information. In addition, names may have other semantic or mnemonic information that either helps human users remember or figure out the names, or includes other semantic information about the resource being named. The URI working group concluded that there was need for persistent, globally unique identifiers, distinct from location or other semantic information; these were called URNs. These "names" provide identity, in that if two of them are "the same" (under some simple rule of canonicalization), they identify the same resource. Furthermore, the group decided that these "names" were generally to be for machine, rather than human, consumption. Finally, with these guidelines for RDS's, this group has recognized the value of the separation of name assignment management from name resolution management.

In contrast to URNs, one can imagine a variety human-friendly naming (HFN) schemes supporting different suites of applications and user communities. These will need to provide mappings to URNs in tighter or looser couplings, depending on the namespace. It is these HFNs that will be mnemonic, content-full, and perhaps mutable, to track changes in use and semantics. They may provide nicknaming and other

aliasing, relative or short names, context sensitive names, descriptive names, etc. Their definition is not part of this effort, but will clearly play an important role in the long run.

URNs as described in RFC 1737 are defined globally; they are ubiquitous in that a URN anywhere in any context identifies the same resource. Given this requirement on URNs, one must ask about its implication for an RDS. Does ubiquity imply a guarantee of RDS resolution everywhere? Does ubiquity imply resolution to the same information about resolution everywhere? In both cases the answer is probably not. One cannot make global, systemic guarantees, except at an expense beyond reason. In addition there may be policy as well as technical reasons for not resolving in the same way everywhere. It is quite possible that the resolution of a URN to an instance of a resource may reach different instances or copies under different conditions. Thus, although a URN anywhere refers to the same resource, in some environments under some conditions, and at different times, due to either the vagaries of network conditions or policy controls a URN may sometimes be resolvable and other times or places not. Ubiquitous resolution cannot be assumed simply because naming is ubiquitous. On the other hand wide deployment and usage will be an important feature of any RDS design.

Within the URI community there has been a concept used frequently that for lack of a better term we will call a _hint_. A hint is something that helps in the resolution of a URN; in theory we map URNs to hints as an interim stage in accessing a resource. In practice, an RDS may map a URN directly into the resource itself if it chooses to. It is very likely that there will be hints that are applicable to large sets of URNs, for example, a hint that indicates that all URNs with a certain prefix or suffix can be resolved by a particular resolver. A hint may also have meta-information associated with it, such as an expiration time or certification of authenticity. We expect that these will stay with a hint rather than being managed elsewhere. We will assume in all further discussion of hints that they include any necessary meta-information as well as the hint information itself. Examples of hints are: 1) the URN of a resolver service that may further resolve the URN, 2) the address of such a service, 3) a location at which the resource was previously found. The defining feature of hints is that they are only hints; they may be out of date, temporarily invalid, or only applicable within a specific locality. They do not provide a guarantee of access, but they probably will help in the resolution process. By whatever means available, a set of hints may be discovered. Some combination of software and human choice will determine which hints will be tried and in what order.

We must assume that most resolutions of URNs will be provided by the use of locally stored hints, because maintaining a database of globally available, completely up-to-date location information is infeasible for performance reasons. There are a number of circumstances in which one can imagine that hints become invalid, either because a resource has moved or because a different URN resolver service has taken over the responsibility for resolution of the URN. Hints may be found in a variety of places. It is generally assumed that a well engineered system will maintain or cache a set of hints for each URN at each location where that URN is found. These may have been acquired from the owner of the resources, a recommendation of the resource, or one of many other sources. In addition, for those situations in which those hints found locally fail, a well engineered system will provide a fall-back mechanism for discovering further hints. It is this fall-back mechanism, an RDS, that is being addressed in this document. As with all hints, there can never be a guarantee that access to a resource will be available to all clients, even if the resource is accessible to some. However, an RDS is expected to work with reasonably high reliability, and, hence, may result in increased response time.

The remainder of this document falls into three sections. The first identifies several sets of assumptions underlying this work. There are three general assumptions:

* URNs are persistent;
* URN assignment can be delegated;
* Decisions can be made independently, enabling isolation from decisions of one’s peers.

The next section lays out three central principles Resolver Discovery Service design. For each of these, we have identified a number of more specific guidelines that further define and refine the general principle. This section is probably the most critical of the document, because one must hold any proposed RDS scheme up against these principles and corollary guidelines to learn whether or not it is adequate. The three central principles can be summarized as:

1) An RDS must allow for evolution and evolvability;
2) Usability of an RDS with regard to each of the sets of constituents involved in the identification and location or resources is paramount;
3) It is centrally important that the security and privacy needs of all constituents be feasibly supported, to the degree possible.

Each of the three major subsections of the guidelines section begins with a summary list of the more detailed guidelines identified in

that section.

The final section of the document lays out a framework for such RDSs. The purpose of this last section is to bound the search space for RDS schemes. The RDS designer should be aware that meeting the guidelines is of primary importance; it is possible to meet them without conforming to the framework. As will be discussed further in this last section, designing within the framework does not guarantee compliance, so compliance evaluation must also be part of the process of evaluation of a scheme.

2. Assumptions

Based on previous internet drafts and discussion in both the URN BOFs and on the URN WG mailing list, three major areas of assumptions are apparent: longevity, delegation, and independence. Each will be discussed separately.

The URN requirements [2] state that a URN is to be a "persistent identifier". It is probably the case that nothing will last forever, but in the time frame of resources, users of those resources, and the systems to support the resources, the identifier should be considered to be persistent or have a longer lifetime than those other entities. There are two assumptions that are implied by longevity of URNs: mobility and evolution. Mobility will occur because resources may move from one machine to another, owners of resources may move among organizations, or the organizations themselves may merge, partition, or otherwise transforms themselves. The Internet is continually evolving; protocols are being revised, new ones created, while security policies and mechanisms evolve as well. These are only examples. In general, we must assume that almost any piece of the supporting infrastructure of URN resolution will evolve. In order to deal with both the mobility and evolution assumptions that derive from the assumption of longevity, we must assume that users and their applications can remain independent of these mutating details of the supporting infrastructure.

The second assumption is that naming and resolution authorities may delegate some of their authority or responsibility; in both cases, the delegation of such authority is the only known method of allowing for the kind of scaling expected. It is important to note that a significant feature of this work is the potential to separate name assignment, the job of labelling a resource with a URN, from name resolution, the job of discovering the resource given the URN. In both cases, we expect multi-tiered delegation. There may be RDS schemes that merge these two sets of responsibilities and delegation relationships; by doing so, they bind together or overload two distinctly different activities, thus probably impeding growth.

The third assumption is independence or isolation of one authority from another and, at least to some extent, from its parent. When one authority delegates some of its rights and responsibilities to another authority, the delegatee can operate in that domain independently of its peers and within bounds specified by the delegation, independently of the delegator. This isolation is critically important in order to allow for independence of policy and mechanism.

This third assumption has several corollaries. First, we assume that the publisher of a resource can choose resolver services, independently of choices made by others. At any given time, the owner of a namespace may choose a particular URN resolver service for that delegated namespace. Such a URN resolver service may be outside the RDS service model, and only identified or located by the RDS service. Second, it must be possible to make a choice among RDS services. The existence of multiple RDS services may arise from the evolution of an RDS service, or development of new ones. Although at any given time there is likely to be only one or a small set of such services, the number is likely to increase during a transition period from one architecture to another. Thus, it must be assumed that clients can make a choice among a probably very small set of RDSs. Third, there must be independence in the choice about levels and models of security and authenticity required. This choice may be made by the owner of a naming subspace, in controlling who can modify hints in that subspace. A naming authority may delegate this choice to the owners of the resources named by the names it has assigned. There may be limitations on this freedom of choice in order to allow other participants to have the level of security and authenticity they require, for example, in order to maintain the integrity of the RDS infrastructure as a whole. Fourth, there is an assumption of independence of choice of the rule of canonicalization of URNs within a namespace, limited by any restrictions or constraints that may have been set by its parent namespace. This is a choice held by naming authorities over their own subnamespaces. Rules for canonicalization will be discussed further in the framework section below. Thus, there are assumptions of independence and isolation to allow for delegated, independent authority in a variety of domains.

The modularity assumptions of delegation and isolation imply independence of decision and implementation, leading to a decentralization that provides a certain degree of safety from denial of service. Based on these these assumptions in conjunction with that of longevity and those for URLs and URNs as detailed in RFCs 1736 and 1737, we can now turn to the guidelines for an RDS.

3. Guidelines

The guidelines applying to an RDS center around three important design principles in the areas of evolvability, usability, and security and privacy. At its core the function of an RDS is to provide hints for accessing a resource given a URN for it. These hints may range in applicability from local to global, and from short-lived to long-lived. They also may vary in their degree of verifiable authenticity. While it may be neither feasible nor necessary that initial implementations support every guideline, every implementation must support evolution to systems that do support the guidelines more fully.

It is important to note that there are requirements, not applicable specifically to an RDS that must also be met. A whole URN system will consist of names in namespaces, the resolution information for them, and the mapping from names in the namespaces to resolution information (or hints). URNs themselves must meet the requirements of RFC 1737. In addition, namespaces themselves must meet certain requirements as described by the URN Working Group [4]. Although all these requirements and guidelines are not described here, they must be supported to provide an acceptable system.

Each section below begins with a summary of the points made in that section. There is some degree of overlap among the areas, such as in allowing for the evolution of security mechanisms, etc., and hence issues may be addressed in more than one section. It is also important to recognize that conformance with the guidelines will often be subjective. As with many IETF guidelines and requirements, many of these are not quantifiable and hence conformance is a judgment call and a matter of degree. Lastly, the reader may find that some of them are those of general applicability to distributed systems and some are specific to URN resolution. Those of general applicability are included for completeness and are not distinguished as such.

3.1 Evolution

The issues in the area of the first principle, that of evolvability, are:

1.1) An RDS must be able to support scaling in at least three dimensions: the number of resources for which URNs will be required, the number of publishers and users of those resources, and the complexity of the delegation, as authority for resolution grows and possibly reflects delegation in naming authority;
1.2) A hint resolution environment must support evolution of mechanisms, specifically for:
* a growing set of URN schemes;
* new kinds local URN resolver services;
* new authentication schemes;
* alternative RDS schemes active simultaneously;
1.3) An RDS must allow the development and deployment of administrative control mechanisms to manage human behavior with respect to limited resources.

One of the lessons of the Internet that we must incorporate into the development of mechanisms for resolving URNs is that we must be prepared for change. Such changes may happen slowly enough to be considered evolutionary modifications of existing services, or dramatically enough to be considered revolutionary. They may permeate the Internet universe bit by bit, living side by side with earlier services or they may take the Internet by storm, causing an apparent complete transformation over a short period of time. There are several directions in which we can predict the need for evolution. At the very least, the community and the mechanisms proposed should be prepared for these.

First, scaling is a primary issue in conjunction with evolution. The number of users, both human and electronic, as well as the number of resources will continue to grow exponentially for the near term, at least. Hence the number of URNs will also increase similarly. In addition, with growth in sheer numbers is likely to come growth in the delegation of both naming authority and resolution authority. These facts mean that an RDS design must be prepared to handle increasing numbers of requests for inclusion, update and resolution, in a set of RDS servers perhaps inter-related in more complex ways. This is not to say that there will necessarily be more updates or resolutions per URN; we cannot predict that at this time. But, even so, the infrastructure may become more complex due to delegation, which may (as can be seen in Section 4 on the framework) lead to more complex rules for rewriting or extracting terms for staged resolution. Any design is likely to perform less well above some set of limits, so it is worth considering the growth limitations of each design alternative.

Second, we expect there to be additions and changes to the mechanisms. The community already understands that there must be a capacity for new URN schemes, as described in [4]. A URN scheme will define a set of URNs that meet the URN requirements [2], but may have further constraints on the internal structure of the URN. The intention is that URN schemes can be free to specify parts of the URN that are left opaque in the larger picture. In fact, a URN scheme may choose to make public or keep private the algorithms for any such "opaque" part of the URN. In any case, we must be prepared for a growing number of URN schemes.

Often in conjunction with a new URN scheme, but possibly independently of any particular URN scheme, new kinds of resolver services may evolve. For example, one can imagine a specialized resolver service based on the particular structure of ISBNs that improves the efficiency of finding documents given their ISBNs. Alternatively, one can also imagine a general purpose resolver service that trades performance for generality; although it exhibits only average performance resolving ISBNs, it makes up for this weakness by understanding all existing URN schemes, so that its clients can use the same service to resolve URNs regardless of naming scheme. In this context, there will always be room for improvement of services, through improved performance, better adaptability to new URN schemes, or lower cost, for example. New models for URN resolution will evolve and we must be prepared to allow for their participation in the overall resolution of URNs.

If we begin with one overall plan for URN resolution, into which the enhancements described above may fit, we must also be prepared for an evolution in the authentication schemes that will be considered either useful or necessary in the future. There is no single globally accepted authentication scheme, and there may never be one. Even if one does exist at some point in time, we must always be prepared to move on to newer and better schemes, as the old ones become too easily spoofed or guessed.

In terms of mechanism, although we may develop and deploy a single RDS scheme initially, we must be prepared for that top level model to evolve. Thus, if the RDS model supports an apparently centralized (from a policy standpoint) scheme for inserting and modifying authoritative information, over time we must be prepared to evolve to a different model, perhaps one that has a more distributed model of authority and authenticity. If the model has no core but rather a cascaded partial discovery of information, we may find that this becomes unmanageable with an increase in scaling. Whatever the model, we must be prepared for it to evolve with changes in scaling, performance, and policy constraints such as security and cost.

The third evolutionary issue is even more mechanical than the others. At any point in time, the community is likely to be supporting a compromise position with respect to resolution. We will probably be operating in a situation balanced between feasibility and the ideal, perhaps with policy controls used to help stabilize use of the service. Ideally, the service would be providing exactly what the customers wanted and they in turn would not request more support than they need, but it seems extremely unlikely. Since we will almost always be in a situation in which some service provision resources will be in short supply, some form of policy controls will generally be necessary. Some policy controls may be realized as mechanisms within the servers or in the details of protocols, while others may only be realized externally to the system. For example, suppose hint entries are being submitted in such volume that the hint servers are using up their excess capacity and need more disk space. Two suggestions for policy control are pricing and administrative. As technology changes and the balance of which resources are in short supply changes, the mechanisms and policies for controlling their use must evolve as well.

3.2 Usability

To summarize, the usability guidelines fall into three areas based on participation in hint management and discovery:

- 2.1) The publisher
  - 2.1.1) URN to hint resolution must be correct and efficient with very high probability;
  - 2.1.2) Publishers must be able to select and move among URN resolver services to locate their resources;
  - 2.1.3) Publishers must be able to arrange for multiple access points for their location information;
  - 2.1.4) Publishers should be able to provide hints with varying lifetimes;
  - 2.1.5) It must be relatively easy for publishers to specify to the management and observe their hint information as well as any security constraints they need for their hints.
- 2.2) The client
  - 2.2.1) The interface to the RDS must be simple, effective, and efficient;
  - 2.2.2) The client and client applications must be able to understand the information stored in and provided by the RDS easily, in order to be able to make informed choices.
- 2.3) The management
  - 2.3.1) The management of hints must be as unobtrusive as possible, avoiding using too many network resources;

2.3.2) The management of hints must allow for administrative controls that encourage certain sorts of behavior deemed necessary to meet other requirements;
2.3.3) The configuration and verification of configuration of individual RDS servers must be simple enough not to discourage configuration and verification.

Usability can be evaluated from three distinct perspectives: those of a publisher wishing to make a piece of information public, those of a client requesting URN resolution, and those of the provider or manager of resolution information. We will separately address the usability issues from each of these three perspectives. It is important to recognize that these may be sitautions in which interests of some of the participants (for exampel a use and a publisher) may be in conflict; some resolution will be needed.

It is worth noting that there are two additional sorts of participants in the whole naming process, as discussed in the URN WG. They are the naming authorities which choose and assign names, and the authors who include URNs in their resources. These two are not relevant to the design of an RDS and hence are not discussed further here.

3.2.1 The Publisher

The publisher must be able to make URNs known to potential customers. From the perspective of a publisher, it is of primary importance that URNs be correctly and efficiently resolvable by potential clients with very high probability. Publishers stand to gain from long-lived URNs, since they increase the chance that references continue to point to their published resources.

The publisher must also be able to choose easily among a variety of potential services that might translate URNs to location information. In order to allow for this mobility among resolvers, the RDS architecture must support such transitions, within policy control bounds. It is worth noting that although multiple listing services are available in telephone books, they are generally accompanied by a fee. There is nothing preventing there being fees collected for similar sorts of services with respect to URNs.

The publisher must be able to arrange for multiple access points to a published resource. For this to be useful, resolver services should be prepared to provide different resolution or hint information to different clients, based on a variety of information including location and the various access privileges the client might have. It is important to note that this may have serious implications for caching this information. For example, companies might arrange for

locally replicated copies of popular resources, and would like to provide access to the local copies only for their own employees. This is distinct from access control on the resource as a whole, and may be applied differently to different copies.

The publisher should be able to provide both long and short term location information about accessing the resource. Long term information is likely to be such information as the long term address of a resource itself or the location or identity of a resolver service with which the publisher has a long term relationship. One can imagine that the arrangement with such a long term "authoritative" resolver service might be a guarantee of reliability, resiliency to failure, and atomic updates. Shorter term information is useful for short term changes in services or to avoid short lived congestion or failure problems. For example, if the actual repository of the resource is temporarily inaccessible, the resource might be made available from another repository. This short term information can be viewed as temporary refinements of the longer term information, and as such should be more easily and quickly made available, but may be less reliable. Some RDS designs may not distinguish between these two extremes.

Lastly, the publishers will be the source of much hint information that will be stored and served by the manager of the infrastructure. Despite the fact that many publishers will not understand the details of the RDS mechanism, it must be easy and straightforward for them to install hint information. This means that in general any one who wishes to publish and to whom the privilege of resolution has been extended through delegation, can do so. The publisher must be able not only to express hints, but also to verify that what is being served by the manager is correct. Furthermore, to the extent that there are security constraints on hint information, the publisher must be able to both express them and verify compliance with them easily.

3.2.2 The Client

From the perspective of the client, simplicity and usability are paramount. Of critical importance to serving clients effectively is that there be an efficient protocol through which the client can acquire hint information. Since resolving the name is only the first step on the way to getting access to a resource, the amount of time spent on it must be minimized.

Furthermore, it will be important to be able to build simple, standard interfaces to the RDS so that both the client and applications on the client’s behalf can interpret hints and subsequently make informed choices. The client, perhaps with the

assistance of the application, must be able to specify preferences and priorities and then apply them. If the ordering of hints is only partial, the client may become directly

involved in the choice and interpretation of them and hence they must be understandable to that client. On the other hand, in general it should be possible to configure default preferences, with individual preferences viewed as overriding any defaults.

From the client's perspective, although URNs will provide important functionality, the client is most likely to interact directly only with human friendly names (HFNs). As in direct human interaction (not computer mediated), the sharing of names will be on a small, private, or domain specific scale. HFNs will be the sorts of references and names that are easy to remember, type, choose among, assign, etc. There will also need to be a number of mechanisms for mapping HFNs to URNs. Such services as "yellow pages" or "search tools" fall into this category. Although we are mentioning HFNs here, it is important to recognize that HFNs and the mappings from HFNs to URNs is and must remain a separate functionality from an RDS. Hence, although HFNs will be critical to clients, they do not fall into the domain of this document.

3.2.3 The Management

Finally, we must address the usability concerns with respect to the management of the hint infrastructure itself. What we are terming "management" is a service that is distinct from publishing; it is the core of an RDS. It involves the storage and provision of hints to the clients, so that they can find published resources. It also provides security with respect to name resolution to the extent that there is a commitment for provision of such security; this is addressed in Section 3.3 below.

The management of hints must be as unobtrusive as possible. First, its infrastructure (hint storage servers and distribution protocols) must have as little impact as possible on other network activities. It must be remembered that this is an auxiliary activity and must remain in the background.

Second, in order to make hint management feasible, there may need to be a system for administrative incentives and disincentives such as pricing or legal restrictions. Recovering the cost of running the system is only one reason for levying charges. The introduction of payments often has an impact on social behavior. It may be necessary to discourage certain forms of behavior that when out of control have serious negative impact on the whole community. At the same time, any administrative policies should encourage behavior that benefits

the community as a whole. Thus, for example, a small one-time charge for authoritatively storing a hint might encourage conservative use of hints. If we assume that there is a fixed cost for managing a hint, then the broader its applicability across the URN space, the more cost effective it is. That is, when one hint can serve for a whole collection of URNs, there will be an incentive to submit one general hint over a large number of more specific hints. Similar policies can be instituted to discourage the frequent changing of hints. In these ways and others, behavior benefitting the community as a whole can be encouraged.

Lastly, symmetric to issues of usability for publishers, it must also be simple for the management to configure the mapping of URNs to hints. It must be easy both to understand the configuration and to verify that configuration is correct. With respect to management, this issue may have an impact not only on the information itself but also on how it is partitioned among network servers that collaboratively provide the management service or RDS. For example, it should be straightforward to bring up a server and verify that the data it is managing is correct. Although this is not a guideline, it is worth nothing that since we are discussing a global and probably growing service, encouraging volunteer participants suggests that, as with the DNS, such volunteers can feel confident about the service they are providing and its benefit to both themselves and the rest of the community.

3.3 Security and Privacy

In summary, security and privacy guidelines can be identified as some degree of protection from threats. The guidelines that fall under this third principle, that of security, are all stated in terms of possibilities or options for users of the service to require and utilize. Hence they address the availability of functionality, but not for the use of it. We recognize that all security is a matter of degree and compromise. These may not satisfy all potential customers, and there is no intention here to prevent the building of more secure servers with more secure protocols to suit their needs. These are intended to satisfy the needs of the general public.

3.1) It must be possible to create authoritative versions of a hint with access-to-modification privileges controlled;
3.2) It must be possible to determine the identity of servers or avoid contact with unauthenticated servers;
3.3) It must be possible to reduce the threat of denial of service by broad distribution of information across servers;
3.4) It must be possible within the bounds of organizational policy criteria to provide at least some degree of privacy for traffic;

3.5) It must be possible for publishers to keep private certain information such as an overall picture of the resources they are publishing and the identity of their clients;
3.6) It must be possible for publishers to be able to restrict access to the resolution of the URNs for the resources they publish, if they wish.

When one discusses security, one of the primary issues is an enumeration of the threats being considered for mitigation. The tradeoffs often include cost in money and computational and communications resources, ease of use, likelihood of use, and effectiveness of the mechanisms proposed. With this in mind, let us consider a set of threats.

Voydock and Kent [5] provide a useful catalog of potential threats. Of these the passive threats to privacy or confidentiality and the active threats to authenticity and integrity are probably the most important to consider here. To the extent that spurious association causes threats to the privacy, authenticity, or integrity with respect to information within servers managing data, it is also important. Denial of service is probably the most difficult of these areas of threats both to detect and to prevent, and we will therefore set it aside for the present as well, although it will be seen that solutions to other problems will also mitigate some of the problems of denial of service. Furthermore, because this is intended to be provide a global service to meet the needs of a variety of communities, the engineering tradeoffs will be different for different clients. Hence the guidelines are stated in terms of, "It must be possible..." It is important to note that the information of concern here is hint information, which by nature is not guaranteed to be correct or up-to-date; therefore, it is unlikely to be worth putting too much expense into the correctness of hints, because there is no guarantee that they are still correct anyway. The exact choice of degree of privacy, authenticity, and integrity must be determined by the needs of the client and the availability of services from the server.

To avoid confusion it is valuable to highlight the meanings of terms that have different meanings in other contexts. In this case, the term "authoritative" as it is used here connotes the taking of an action or stamp of approval by a principal (again in the security sense) that has the right to perform such an act of approval. It has no implication of correctness of information, but only perhaps an implication of who claimed it to be correct. In contrast, the term is often also used simply to refer to a primary copy of a piece of information for which there may also be secondary or cached copies available. In this discussion of security we use the former meaning, although it may also be important to be able to learn about whether a

piece of information is from a primary source or not and request that it be primary. This second meaning arises elsewhere in the document and is so noted there.

It is also important to distinguish various possible meanings for "access control". There are two areas in which distinctions can be made. First, there is the question of the kind of access control that is being addressed, for example, in terms of hints whether it is read access, read and modify access, or read with verification for authenticity. Second, there is the question of to what access is being controlled. In the context of naming it might be the names themselves (not the case for URNs), the mapping of URNs to hints (the business of an RDS), the mapping of URNs to addresses (not the business of an RDS as will be discussed below in terms of privacy), or the resource itself (unrelated to naming or name resolution at all). We attempt to be clear about what is meant when using "access control".

There is one further issue to address at this point, the distinction between mechanism and policy. In general, a policy is realized by means of a set of mechanisms. In the case of an RDS there may be policies internal to the RDS that it needs to have supported in order to do its business as it sees fit. Since, in general it is in the business of storing and distributing information, most of its security policies may have to do with maintaining its own integrity, and are rather limited. Beyond that, to the degree possible, it should impose no policy on its customers, the publishers and users. It is they that may have policies that they would like supported by the RDS. To that end, an RDS should provide a spectrum of "tools" or mechanisms that the customers can cause to be deployed on their behalf to realize policies. An RDS may not provide all that is needed by a customer. A customer may have different requirements within his or her administrative bounds than outside. Thus, "it must be possible..." captures the idea that the RDS must generally provide the tools to implement policies as needed by the customers.

The first approach to URN resolution is to discover local hints. In order for hints to be discovered locally, they will need to be as widely distributed as possible to what is considered to be local for every locale. The drawback of such wide distribution is the wide distribution of updates, causing network traffic problems or delays in delivering updates. An alternative model would concentrate hint information in servers, thus requiring that update information only be distributed to these servers. In such a model the vulnerable points are the sources of the information and the distribution network among them. Attackers on the integrity of the information stored in a server may come in the form of masquerading as the owner or the server of the information. Wide replication of information

among servers increases the difficult of masquerading at all the locations of the information as well as reducing the threat of denial service. These lead us to three identifiable guidelines for our security model:

* ACCESS CONTROL ON HINTS: It must be possible to create an authoritative version of each hint with change control limited only to those principals with the right to modify it. The choice of who those principals are or whether they are unlimited must be made by the publisher of a hint.

* SERVER AUTHENTICITY: Servers and clients must be able to learn the identity of the servers with which they communicate. This will be a matter of degree and it is possible that there will be more trustworthy, but less accessible servers, supported by a larger cluster of less authenticatable servers that are more widely available. In the worst case, if the client receives what appears to be unvalidated information, the client should assume that the hint may be inaccurate and confirmation of the data might be sought from more reliable but less accessible sources.

* SERVER DISTRIBUTION: Broad availability will provide resistance to denial of service. It is only to the extent that the services are available that they provide any degree of trustworthiness. In addition, the distribution of services will reduce vulnerability of the whole community, by reducing the trust put in any single server. This must be mitigated by the fact that to the extent trust is based on a linked set of servers, if any one fails, the whole chain of trust fails; the more elements there are in such a chain, the more vulnerable it may become.

Privacy can be a double-edged sword. For example, on one hand, an organization may consider it critically important that its competitors not be able to read its traffic. On the other hand, it may also consider it important to be able to monitor exactly what its employees are transmitting to and from whom, for a variety of reasons such as reducing the probability that its employees are giving or selling the company’s secrets to verifying that employees are not using company resources for private endeavor. Thus, although there are likely to be needs for privacy and confidentiality, what they are, who controls them and how, and by what mechanisms vary widely enough that it is difficult to say anything concrete about them here.

The privacy of publishers is much easier to address. Since they are trying to publish something, in general privacy is probably not desired. However, publishers do have information that they might like to keep private: information about who their clients are, and information about what names exist in their namespace. The

information about who their clients are may be difficult to collect depending on the implementation of the resolution system. For example, if the resolution information relating to a given publisher is widely replicated, the hits to _each_ replicated copy would need to be recorded. Of course, determining if a specific client is requesting a given name can be approached from the other direction, by watching the client as we saw above.

There are likely to be some publishers publishing for a restricted audience. To the extent they want to restrict access to a resource, that is the responsibility of the repository providing and restricting access to the resource. If they wish to keep the name and hints for a resource private, a public RDS may be inadequate for their needs. In general, it is intended for those who want customers to find their resources in an unconstrained fashion.

The final privacy issue for publishers has to do with access control over URN resolution. This issue is dependent on the implementation of the publisher's authoritative (in the sense of "primary) URN resolver server. URN resolver servers can be designed to require proof of identity in order to be issued resolution information; if the client does not have permission to access the URN requested, the service denies that such a URN exists. An encrypted protocol can also be used so that both the request and the response are obscured. Encryption is possible in this case because the identity of the final recipient is known (i.e. the URN server). Thus, access control over URN resolution can and should be provided by resolver servers rather than an RDS.

4. The Framework

With these assumptions and guidelines in mind, we conclude with a general framework within which RDS designs may fall. As stated earlier, although this framework is put forth as a suggested guide for RDS designers, compliance with it will in no way guarantee compliance with the guidelines. Such an evaluation must be performed separately. All such lack of compliance should be clearly documented.

The design of the framework is based on the syntax of a URN as documented in RFC-2141 [6]. This is:

URN:<NID>:<NSS>

where URN: is a prefix on all URNs, NID is the namespace identifier, and NSS is the namespace specific string. The prefix identifies each URN as such. The NID determines the general syntax for all URNs within its namespace. The NSS is probably partitioned into a set of

delegated and subdelegated namespaces, and this is possibly reflected in further syntax specifications. In more complex environments, each delegated namespace will be permitted to choose the syntax of the variable part of the namespace that has been delegated to it. In simpler namespaces, the syntax will be restricted completely by the parent namespace. For example, although the DNS does not meet all the requirements for URNs, it has a completely restricted syntax, such that any further structuring must be done only by adding further refinements to the left, maintaining the high order to low order, right to left structure. A delegated syntax might be one in which a host is named by the DNS, but to the right of that and separated by an "@" is a string whose internal ordering is defined by the file system on the host, which may be defined high order to low order, left to right. Of course, much more complex and nested syntaxes should be possible, especially given the need to grandfather namespaces. In order to resolve URNs, rules will be needed for two reasons. One is simply to canonicalize those namespaces that do not fall into a straightforward (probably right to left or left to right) ordering of the components of a URN, as determined by the delegated naming authorities involved. It is also possible that rules will be needed in order to derive from URNs the names of RDS servers to be used in stages.

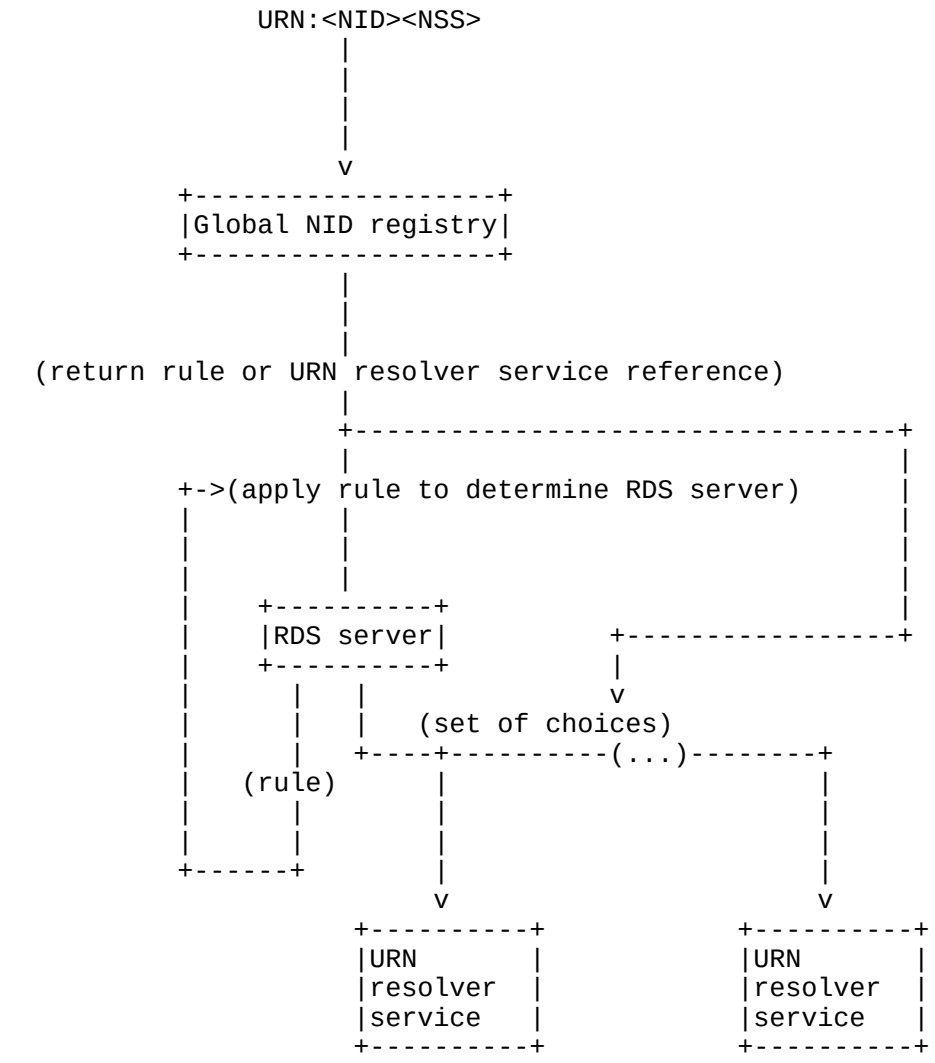

Figure 1: An RDS framework

The NID defines a top level syntax. This syntax will determine whether the NID alone or in conjunction with some extraction from the NSS (for the top level naming authority name) is to be used to identify the first level server to be contacted. At each stage of the lookup either a new rule for generating the strings used in yet another lookup (the strings being the identity of another RDS server and possibly a string to be resolved if it is different than the original URN) or a reference outside the RDS to a URN resolver service, sidestepping any further use of the RDS scheme. Figure 1

depicts this process.

There are several points worth noting about the RDS framework. First, it leaves open the determination of the protocols, data organization, distribution and replication needed to support a particular RDS scheme. Second, it leaves open the location of the computations engendered by the rules. Third, it leaves open the possibility that partitioning (distribution) of the RDS database need not be on the same boundaries as the name delegation. This may seem radical to some, but if the information is stored in balanced B-trees for example, the partitioning may not be along those naming authority delegation boundaries (see [7]). Lastly, it leaves open access to the Global NID Registry. Is this distributed to every client, or managed in widely distributed servers? It is important to note that it is the intention here that a single RDS scheme is likely to support names from many or all naming schemes, as embodied in their NIDs.

One concept that has not been addressed in Figure 1 is that there may be more than one RDS available at any given time, in order to allow for evolution to new schemes. Thus, the picture should probably look more like Figure 2.

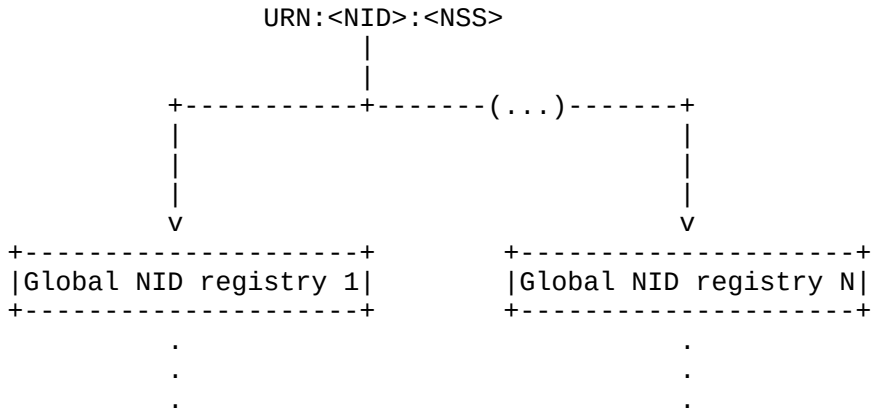

Figure 2: More than one co-existing RDS scheme

If we are to support more than one co-existing RDS scheme, there will need to be coordination among them with respect to storage and propagation of information and modifications. The issue is that generally it should be assumed that all information should be available through any operational RDS scheme. One cannot expect potential publishers to submit updates to more than one RDS scheme. Hence there will need to be a straightforward mapping of information from one to the other of these schemes. It is possible that that transformation will only go in one direction, because a newer RDS service is replacing an older one, which is not kept up to date, in order to encourage transfer to the newer one. Thus, at some point, updates may be made only to the newer one and not be made available to the older one, as is often done with library catalogs.

This framework is presented in order to suggest to RDS scheme designers a direction in which to start designing. It should be obvious to the reader that adherence to this framework will in no way guarantee compliance with the guidelines or even the assumptions described in Sections 2 and 3. These must be reviewed independently as part of the design process. There is no single correct design that will conform to these guidelines. Furthermore, it is assumed that preliminary proposals may not meet all the guidelines, but should be expected to itemized and justify any lack of compliance.

5. Acknowledgments

Foremost acknowledgment for this document goes to Lewis Girod, as my co-author on a preliminary URN requirements document and for his insightful comments on this version of the document. Thanks also go to Ron Daniel especially for his many comments on my writing. In addition, I recognize the contributors to a previous URN framework document, the "Knoxville" group. There are too many of you to acknowledge here individually, but thank you. Finally, I must thank the contributors to the URN working group and mailing list (urn-ietf@bunyip.com), for your animated discussions on these and related topics.

6. References

[1] Kunze, J., "Functional Recommendations for Internet Resource Locators", RFC 1736, February 1995.

[2] Sollins, K., and L. Masinter, "Functional Requirements for Uniform Resource Names", RFC 1738, December 1994.

[3] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[4] URN Working Group, "Namespace Identifier Requirements for URN Services," Work in Progress.

[5] Voydock, V. L., and Kent, S. T., "Security Mechanisms in High-Level Protocols", ACM Computing Surveys, v. 15, No. 2, June, 1983, pp. 135-171.

[6] Moats, R., "URN Syntax", RFC 2141, May 1997.

[7] Slottow, E.G., "Engineering a Global Resolution Service," MIT-LCS-TR712, June, 1997. Currently available as <http://ana.lcs.mit.edu/anaweb/ps-papers/tr-712.ps> or <http://ana.lcs.mit.edu/anaweb/pdf-papers/tr712.pdf>.

7. Author’s Address

Karen Sollins
MIT Laboratory for Computer Science
545 Technology Sq.
Cambridge, MA 02139

Phone: +1 617 253 6006
EMail: sollins@lcs.mit.edu

8. Full Copyright Statement