Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2008-064 October 29, 2008

Recursively invoking Linnaeus: A Taxonomy for Naming Systems

Karen R. Sollins

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Recursively invoking Linnaeus: A Taxonomy of Network Naming Systems

Karen R. Sollins
MIT Laboratory for Computer Science
sollins@lcs.mit.edu

March 1, 2002

Abstract[1]

Naming is a central element of a distributed or network system design. Appropriate design choices are central. This paper explores a taxonomy of naming systems, and engineering tradeoffs as an aid to the namespace designer. The three orthogonal components of the taxonomy are the characteristics of the namespace itself, name assignment, and name resolution. Within each of these, we explore a number of distinct characteristics. The position of this paper is that engineering design of naming systems should be informed by the possibilities and tradeoffs that those possibilities represent. The paper includes a review of a sampling of naming system designs that reflect different choices within the taxonomy and discussion about why those choices were made.

1. Introduction

Carl von Linné, also known as Linnaeus, the 18th century scientist is widely recognized as the father of taxonomy, specifically a taxonomy of living beings. A taxonomy is a scheme for classifying and naming entities. Linnaeus' scheme was based on two principles, a hierarchical categorization and a binomial naming scheme based on genus and species epithet. The hierarchy describes characteristics, while the binomial name provides a unique identifier among all living entities for all time. Linnaeus' scheme has been modified and updated, but remains the core of the naming scheme used in biology. In this paper, we will explore a taxonomy for naming used in networks, in contrast with Linnaeus' subject of living beings.

Defining a naming taxonomy for networking is not new. Two of the early seminal works on this subject were by Shoch [Sho78] and Saltzer [Sal 78, Sal82].[2] Suffice it say that we have been considering the design of naming systems for almost 25 years. Thus, one must ask why do it again. This work attempts to bring together much of the previous work and to bring order to it. Naming is central to any discourse, especially one as distributed and heterogeneous in requirements as networking has become. Furthermore, because it is so central, understanding how to decide among alternatives in naming schemes and their related mechanisms has an enormous impact on the functioning of those systems. Hence, the position of this paper is that engineering design of naming systems should be informed by the possibilities and tradeoffs that those possibilities represent.

It is commonly held that we cannot conceive of concepts without naming them. We certainly cannot identify, discuss, or share concepts or resources without being able to

[1] This effort was sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0553.

[2] Those early works focused on distinguishing the entities to be named and the resolution or translation from the names for one group to the names for another, but not on the actual names themselves.

give them names or identifiers.[3] In other words, names or identifiers are critical to being able to use, share, and exchange objects and resources in cyberspace.

There are three basic functions of names: equality, access or reference, and meaning or mnemonics. If names are assigned in a unique (one-to-one and onto) relationship with the entities they name, they allow us to ascertain whether or not two entities are in fact one and the same, without having to access the entity itself. Names can also be used as handles providing access or reference to named entities. To do this, names may need some resolution mechanism. Finally, in some cases, names or identifiers also have meaning relevant to the named entities. They may describe characteristics such as the owner, the language in which a program is written, or function or purpose. For example, a host might be named *router1.cs.univ1.edu*, identifying it as a router owned by the computer science department of University 1. In some cases, names intentionally have no meaning at all. An example of such an identifier is one that is generated using a hash function run over the bit representation of the original version of a document, simply to generate a probabilistically unique identifier.

In automated systems, there are a number of aspects of the system to be designed. The design of the set of names or namespace from which names can be selected will have a significant impact on other aspects of the naming system. Additionally, there are questions of who does the assignment of names for which entities and where and how names are resolved.

This paper provides a taxonomy of naming: the objective is to clarify and simplify the process of designing namespaces and their related mechanisms appropriately. Section 2 describes characteristics of namespaces. Namespace resolution alternatives are explored in Section 3. With these in mind, in Section 4 we consider the tradeoffs in deciding how to design an appropriate namespace for particular situations. Finally, in Section 5, we review some existing examples in light of the taxonomy and ensuing discussion. The paper is summarized in Section 6.

[3] In this work, we use the terms *name* and *identifier* interchangeably to connote a handle for an entity.

2. Characterizing naming systems

There are a number of dimensions across which we can characterize names and naming. We consider three major categories of characteristics: those of namespaces themselves, those of name assignment, and those of name resolution. We identify here those that we believe are particularly important. This is not intended to be a complete list as much as a list of key characteristics.

The namespace

Scope of namespace - Global vs. local: the question here is whether a namespace, the set of names from which an individual name can be drawn is large enough to handle all such requests globally. We can contrast two distinct situations. If a namespace is *global*, it provides a single, shared pool of names from which to select. In contrast, if a namespace is *local*, it provides names that may also be provided by another local namespace. If one designs a global namespace, but the space runs out (the names are all used), then either names will no longer be available from that namespace, or names will need to be reused. Reuse can occur in the time domain, by reassigning names over time or in the space domain by creating local namespaces. A related issue is that if the possible number of names is comparable to the number of entities to be named within that namespace, name assignment will be dense. There may be implementation implications of a dense namespace.

Syntax of namespace: there are several aspects to the syntax of a namespace. One is

the size of individual names. In some cases, names will be of fixed size; in others, names will be unbounded. There are advantages and disadvantages of each, one of which is the utility of the names to the client community. Machines often deal more efficiently with fixed size names or identifiers, whereas humans often find variable and unbounded length names to be more useful. Another is whether or not the names have internal structure or not. There are three well-known structures; at one extreme there is a ***flat namespace***, in which names have no structure and are simply selected from a large space. At the other extreme are very organized namespaces, in particular ***hierarchies***, in which a name is composed of elements selected from successive subspaces; such namespaces may be singly or multiply rooted.[4] In the middle there are ***composite names*** that are less organized. Attribute based naming is such an example.[5] A third characteristic is the character set available; this may have an impact on who can use or reproduce a name

Finally, it is worth noting that some composite or hierarchical names include punctuation or special characters to distinguish components and others do not. This may be intended for ease of use for potential users of the names. In some cases, there may be different representations for the name under different conditions. An example is the IP address, which may be represented as dotted quads or bit strings.

Name assignment

Authority for assignment of names: distinct from the structure, but often reflected in it, is the authority for assigning names. The DNS reflects the delegation of authority for naming in the hierarchical structure of the names themselves; authority for naming in subspaces of the DNS follow its hierarchical structure. In the telephone numbering system within the US, authority for assigning telephone numbers is delegated to telephone companies, by giving them sets of 10,000 numbers. There is no reflection of that delegation in the numbers themselves. IP addresses are delegated in different sized sets throughout the address space.

Persistence: Some names may be assigned permanently, while others may be available for re-use over time. This is distinct from how global or universal a name may be. Thus, for example, the DNS may be considered global, but an assignment is not persistent. A DNS name may be reassigned to something completely different. Some DNS names are leased for a stated period of time and, if not renewed, available to any other customer. Social Security Numbers within the US are intended to persist significantly longer than a human lifetime, although even they may be re-used. One can imagine a namespace based on an algorithm such as MD5 [Riv95] applied to a combination an entity, the owner's name and a current timestamp to generate an id or name that is probabilistically unique for all time. The probability of uniqueness can be increased to any desirable degree by increasing the size of the result. Such an identifier might continue to be used for the entity or resource regardless of modifications to the entity, because the only purpose was to generate an identifier.

Uniqueness: in some cases, it is important that there be a one-to-one mapping between names and entities; thus, each entity named in the namespace would have exactly one name and no two entities could have the same name. This provides an extremely simple set of equality tests using names. If two names are the same, the identify the same entity; furthermore, if the names are different the entities are distinct from each other. In other cases, a name will not be assigned to more than one entity, but an entity may have more than one name; this provides a less strict uniqueness. One can imagine the other form of less strict

[4] The Domain Name System or DNS [Moc87a, Moc87b] is an example of a singly rooted hierarchy.

[5] We find an example in the Ingrid system [FKS+95], in which names are simply a set of attribute value pairs in any order.

uniqueness, although it is not generally found.

Resolution of names

Resolution of namespace - Universal vs. relative: orthogonal to the issue of the scope of the namespace from which a name has been chosen, there is a question of resolution of that name. Within the scope of the namespace, it may be the case that from different places in the network, a name may be resolved differently. For example, if a service is replicated in several places, distinct clients may find the name of the service resolved to different servers. It may well be that for the average client, such distributed service may be invisible, except for enhanced response, but for more sophisticated users or more complex interactions, it may not be invisible.[6]

Existence and persistence of resolution: the issue here is whether or not it is required that a name be resolvable at all times. To the extent that a name may be viewed as a placeholder for an entity that does not yet exist, or may only exist at certain times, the name may not be resolvable at all times. An example is a statically assigned address for a machine that is not always turned on. Another example is a "future" in Multilisp[Hal85], which will only be evaluated as needed in the future. This concept exists in other programming languages as well.

Timing of resolution: a second question related to resolution is when it might occur, early or late. A source route can be considered an early resolution or binding of identifier to route. Routing based on a globally consistent set of routing tables, that occurs one hop at a time might be considered late, or at least time delayed, and spread over a period. Piecemeal resolution is another potential aspect of resolution. A more sophisticated scheme is exemplified by Vadhat in his "Active Names" [VDA+99]. In this work, a name is resolved one component at a time by sending the name to a named service, which may perform some sophisticated function on part of the name and send the remainder of the name on to another service. We will discuss this work further below.

To summarize, we have three major groups of characteristics of name systems: characteristics of the namespace itself, the name assignment process, and the name resolution mechanisms.

3. Translation: aliasing or resolving names

There is another issue worth addressing here, with respect to the translation of names. To do this, let us return to the early work on naming in which Shoch [Sho78] proposed a three-way distinction among a *name* indicating "what we seek", an *address* indicating "where it is", and a *route* indicating "how to get there." Thus, Shoch proposed two tiers of resolution, from name to address and from address to route. Saltzer [Sal82] then proposed an expanded yet still constrained resolution model, but hinted at a generalization in which name resolution is viewed as repeated resolutions from one namespace to another.[7]

In fact, with respect to translation, there are two classes of translation that may occur. We can call one ***aliasing*** and the other ***resolution***. Aliasing is translating a name into another name from the same namespace

[6] In an example from the phone system, Sears has a single nationwide 800 telephone number for customers to learn about a scheduled home delivery time. The client types in the phone number of the delivery location to access the delivery time information. But, the database is partitioned, so that an incoming call is routed to the appropriate regional database based on the area code of the originating phone. Hence if a call to the 800 number is made from outside the delivery region, the entry is unavailable. In this case, the resolution to a particular database is a locally defined function.

[7] Saltzer suggested a similar scheme specifically for operating systems in [Sal78].

and abstraction space. Thus, a nickname for a human might be considered an alias; DNS MX records are a second example. In contrast, resolution is the operation of translating from one namespace to another, for example from human name to email address. An email address can be considered to be a name for a service that will collect, store, and deliver email messages to a human. In the same way, one can consider a DNS name the name of a host in the Internet, and an IP address as the name of a location in the topology of the network.

4. Making choices in the net

With all this in mind, we can now discuss the issues of making choices with respect to naming in a networked environment. There are several key issues influenced by a combination of distribution and scalability. The tradeoffs arise from the different roles in naming: name assignment, resolvers, named entities, and locations.

In name assignment, i.e. the selection of a particular name, there may be both technical and policy issues. Technically, if names are needed in both administratively and physically disparate locations where there are performance constraints on the acquisition of names, the assignment of names may need to be distributed. This may imply a need for coordination of name assignment, although there are solutions, which avoid that, such as hierarchically dividing the namespace into subspaces, or using a hash algorithm to generate names that are probabilistically unique to any degree necessary. Alternatively, if the delays of a centralizing are acceptable, neither of these schemes is necessary.

In addition, there may be technical reasons to use disparate namespaces having to do with syntax. For example in the original definition of 32-bit IP addresses, 8-bits were assigned to each of four components, which forced subspaces to be defined, where they otherwise may not have been important, but generally the 8-bit components were easier for programs to handle than many other alternatives. The choices made to meet these requirements will need to be balanced against creating and managing such namespaces and the translation of them. There may also be policy reasons, such as either paying for names for distributing name assignment as in the DNS today, or security, i.e. keeping the knowledge of names private.

As suggested in the previous section, in order to support both increased numbers of requests for resolution and increases in the namespaces being resolved, resolution may be partitioned or staged. One example of this is found in the DNS [Moc87a, Moc87b][8] in which the information a resolver contains can be cached where needed to distribute and speed up resolution. Because the DNS forms a singly-rooted hierarchy, the root is the most likely candidate to become a hotspot and so is replicated most widely.

A second resolution question is whether there is exactly one resolution service for each namespace or part of a namespace. The IETF URN Working Group [DM97, SM94, Sol98] took the position that in order to have a namespace authorized it was recommended that at least one resolution service be specified, but others might provide resolution as well, either initially or over time. In contrast, Vadhat et al. in their *Active Names* project [VDA+99] assume that each namespace has exactly one namespace program associated with it, which will translate its names. In addition, another resolution tradeoff can be seen in network routing. If there is a single initial routing computation, traffic may not flow until a valid route is known. If discovery of whether or not a valid route exists is only done on a hop-by-hop basis by the traffic itself, the traffic may be carried far into the net before it is learned whether or not a valid route exists. This can lead to a significant waste in

[8] Each of the example naming systems in this section will be considered in more detail in Section 5.

network bandwidth if a route does not exist, but flexibility if it does.[9]

A third issue in making choices has to do with the number of entities being named. If the number of requests for names is likely to be high, then there is good reason to choose a distributed mechanism for assigning them, so that the mechanism scales well with the number of entities. If resolution requests occur particularly frequently, then that scaling might lead to a more widely distributed resolution mechanism. The tradeoffs in these cases may mean that information may be either more widely replicated or distributed and coordinated. Thus, space or numbers of network accesses may be traded off against congestion at more central locations. One must consider where the scaling may occur and what the cost will be of addressing that scaling.

Fourth, if names are not always resolvable, one should not waste many resources learning when that fact is true. In both the DNS and URN work, a name may be assigned or reserved, but not resolvable at any specific time. In the DNS, this may happen if a host is off-line or a name leased but not yet in use. The URN group assumed that names might be selected as placeholders for entities not in existence at any particular time.

The fifth issue relates to the locations of the users of names and the entities being named either topologically or in a more abstract sense (e.g. administratively). For example, although it may cost more in replicated and distributed information management, it may be valuable to keep either name assignment or name resolution within a LAN or other reasonably local network. There may also be policy reasons for such a choice. Again, the cost may be in managing widely distributed services for name assignment and resolution. As the numbers increase, these mechanisms may also increase in complexity.[10]

Orthogonally to the issues raised above, one could consider allowing for different solutions in different situations. One reason for this is that a single solution may not best in all situations, or it may mean that even within what one might consider the same situation, different solutions may be more appropriate for different parts of the net.

We return again to the URN resolution framework. In the URN universe, there was an assumption that different URN namespaces might be used for widely differing purposes. That leads to different choices being made in different parts of the URN space [DVI+99]. For example, US Library of Congress numbers might be one namespace, ISBNs another, and so on.[11] Each will have it's own resolution mechanism, and, in fact, each library that uses Library of Congress numbers might choose to have a distinct resolution mechanism. Furthermore, it was assumed that the resolution service specification might change with time and more than one might be available at any given time.
Tradeoffs can be made in a number of dimensions, time, space, network bandwidth, computing or human resources, and so on. Generally, these sorts of decisions are not simple, and need careful examination of the ramifications of a scheme. Furthermore, different solutions may be appropriate for the same problem in different places or times .

[9] We realize that there is a middle ground here in which it is known globally whether a route exists, although the exact route will only be discovered on a hop-by-hop basis.

[10] A number of years ago, Demers et al. [DGH+87] reported on an interesting situation in the Grapevine distributed naming and email service developed at Xerox PARC, which had reconciliation problems each time it scaled up in size, until the authors took a long and close look at non-linearities in the mechanisms.

[11] It is interesting to note that these two namespaces both may refer to books, but they have different concepts of when two books are the same or different. When a book is a new version of a book, and when it is not, is different for each naming authority, and hence when and whether a new name will be assigned and how such names will be resolved will be different.

Relationships that are larger than *O[N]* or even in some cases, as bad as *O[N]* can make solutions untenable in practice. Naming and name resolution are generally not an end in themselves, but rather one step in achieving some other objective and hence must be appropriate for that other situation.

5. Some examples

In this section we will consider a variety of naming systems, in order to illustrate the choices made to meet differing requirements. This list is not intended to be exhaustive, but rather is selected to provide examples of a variety of choices.

The DNS [Moc87a, Moc87b] is an extremely large and long-lived naming system. The namespace is organized, assigned and represented as a global, singly-rooted hierarchy. Each domain or subnamespace below the root is delegated to an authority; thus for example each ISO-assigned two-character country code is assigned to that country as a direct child of the root. The syntax is that the hierarchy is represented in Roman letters and Arabic numerals, from right to left with periods (".") separating the components. Name assignments are long-lived but not intended to be persistent for all time. Resolution is also hierarchical and is delegated along the same hierarchy as naming authority delegation. In practice resolution may be distributed because the DNS supports local caching of resolution information. The original design was that a name would be universal and resolve to a single IP address, but as mentioned above, in order to improve resolution or performance that is no longer always true. Finally, in the DNS resolution may occur in stages, although generally it is all done prior to transmitting any payload.

File systems present us with other, typically hierarchical naming systems. The most common of these are local file systems, limited to one person's or a community's set of files. If one tries to federate such file systems into larger systems, or perhaps a more global file system, one finds that the same names may have been used in many places. One must then resort to a scheme such as that proposed by Lampson [Lam86][12], in which one prepends[13] the name of each namespace in which independent and perhaps overlapping naming may have occurred. This approach builds more of a hierarchy upwards on to represent the federation, meaning that file names now have been extended, which in turn may lead to a variety of technical problems.

URI syntax[BFM98] as used in URL's provides a combined approach. In general, URL's combine three components. First, the left-most name identities the transport, because resolution and transport are more or less coupled. Second from the left is generally a DNS name, in the normal DNS syntax, going hierarchically from right to left. Finally, the rightmost component of the name is often some local specifier, such as a filename, defined hierarchically by the local file system. Each subspace (DNS, filenames) has its own character set, size, and punctuation constraints. In addition, URLs have their own punctuation and syntax. Thus, the URL namespace is global, hierarchical, and intended to be universal in the same sense as the DNS. A URL has no more persistence than the shortest lifetime of any of its components. If the local filesystem is reorganized, the URL may become invalid. If the company merges and changes its name, the URL may become invalid, and so on. Resolution of URLs is also a composite, based on resolving DNS names and the local component separately. Again, resolution may also be modified in the presence of caches, proxies, and other such services.

[12] Lampson is certainly not alone in this approach, we just use this as an example.

[13] In this kind of approach, these larger scope names need not be prepended literally, as long as the syntax is understood, in order to understand which names are scoped within which others.

In the IETF, the work on Uniform Resource Names falls into two key components, the namespaces [SM94, DVI+99] and the resolution work [DM97, Sol98].[14] The namespace uses a syntax that falls within the bounds of the URI definition [BFM98] in terms of syntax (character sets and punctuation, combination) and is hierarchical in assignment. Thus, name assignment delegation is rooted in IANA [15] and delegated from there. The intention of that group was that resolution might also be hierarchical and delegated in order to distribute the work, but that the resolution hierarchy could and probably would be different in the long run. The issues were persistence and potential service differentiation. The persistence issue was that the names and resolution might have different lifetimes and hence should not be tightly coupled. The differentiation issue was that it ought to be possible to allow for service competition in the business of resolution. Furthermore, the resolution model was that a resolver might completely transform its input into some alternative for further resolution. In the DNS, the transformation that occurs is that hierarchical components of the name are stripped off, so each domain will be resolving a more and more local name. In the URN framework, it is possible that the name may be transformed completely. This leads to a potential for significant complexity and increased possibility for failure or corruption, with the tradeoff of significantly increased flexibility and independence. It is worth noting that the first public implementation of URN's used the DNS as its infrastructure [DM97] and hence had all the limitations of using only the DNS, although the design had grander intentions.

Now, let us consider in turn two naming systems that try to do more; both provide naming, resolution, and transport of payload, with the intention of improving efficiency. The first also supports mobility of end points. The two systems are the Intentional Naming System (INS) [ASB+99] and Active Names[VDA+99].

The INS defines a hierarchical namespace, different from the others we have seen. Each component of a name consists of an attribute name and value pair. The actual syntax is not discussed, but the character set is ASCII characters and numbers. The path from the root to any leaf node passes through nodes that alternate strictly between attribute names and values for those attributes. A particular name is specified as a series of pairs or *name specifiers.* Services advertise their names periodically to resolvers, which consider their resolution information as soft state. When an entity moves it advertises its new location, which is propagated among the resolvers. The resolvers build routes among themselves for access to the end point entities. A lookup at a resolver results in both an IP address and a next hop INS router. Hence, if there is extremely new location information because a destination has moved, as the payload moves through the INS route, that new information will be discovered. Hence resolution in this system is not hierarchical as in most of the preceding examples, although it is staged in order to support mobility.

The authors of the Active Names system have made a different set of design choices. The primary focus in this work is on resolution and transport. The work is not specific about the namespaces themselves, other than that each namespace has a program associated with it and that names are composites, which can be decomposed piecemeal by a series of namespace programs. Resolution is hierarchical and rooted in the sender. Each active name is composed of a pair consisting of the *name* to be resolved and a *namespace program name.* The sender sends these to a local resolver. A namespace program name is also an active name, consisting of the program name and a protocol for retrieving the program. The intention is that name resolution is staged and

[14] The relationship intended between URN's and URL's is that a URN was intended to provide identification of an entity and URL identifies the entity's location.

[15] Internet Assigned Numbers Authority

each namespace program will operate on the next component in a name. The code is all mobile and capable of being executed anywhere, although for policy or other design reasons (access to necessary data, etc.) a namespace program may operate only at certain locations. A namespace program may be as simple as a translator to a next hop namespace program, to which is handed the name with a component removed. It may also be more sophisticated and transform the payload, for example by encoding or decoding, or performance of other activities such as authentication or authorization.

There are two additional features of this "programming model". The first is that it operates in a "continuation" style, rather than procedural, with a return to the original requester. So, each namespace program simply invokes the next and the payload is transported along that path. Because each namespace program identifies and discovers its successor, resolution is hierarchical; resolution in one namespace program is dependent on its parent or predecessor. Invoking a program initially means moving the code to the local processor; if the program in fact will not execute locally, what is transferred is a stub that knows where the real program will run and transfers control and input to that location.

The second feature is the *after methods list*. Each namespace program can add an element to this list. The resulting list can then be used to expedite any return traffic, without having to go through the overhead of discovering new namespace programs. We have yet to see the value of "pre-computing" such a return list proven, but it is worth exploring. It isn't clear that one can discover the best or most efficient or even a workable return path through the series of namespace programs in the outbound path. Resolution on the outbound path is delegated piecemeal, from one namespace program to the next. On the return path, it is all delegated to the namespace programs of the outbound path. Names themselves need to be global, so that any sender can learn about the namespace program to be invoked initially for a name; as mentioned above, namespace program names also need to be global. In this system, we see a very different set of design choices from our previous examples.

As an interesting alternative to these, Clark [Cl02], in the context of the NewArch Project, has proposed the following naming scheme for transport. The intention is that flows of information between a sender and a receiver be identified to those parties, as in a TCP connection, but with the realization that there is no need for such identifiers to be globally defined. Only the sender and receiver need to distinguish a flow from any other flow. Furthermore, as in the INS, mobility must be supported. Thus, one needs to discover the current address, of each party but not be bound to only that address. Anonymous2 proposes that user or application friendly names may be resolved in any of a large number of user or application friendly ways, none of which need to support a global, universal, persistent, etc. namespace and resolution. Instead, the intent is that one find an address and some form of introduction by one of many possible means; this allows sender and receiver to agree to communicate and create an association identifier, which in turn only they need to understand or recognize. This is a much more decentralized and localized approach that avoids much of the cost of deploying another global namespace system and resolution mechanism. The tradeoffs here have to do with reducing naming infrastructure overhead, at the cost perhaps of needing to find some alternative for figuring out whether or not one is in fact communicating with whom one thinks one is, and making bootstrapping perhaps much more challenging.

In our final example HIP [Mos01], Moskowitz is proposing to provide global, authenticatable identification to support a secure association between two parties that is resistant to denial of service attacks and provide anonymity, because identity cannot be found from the various identifiers in HIP.

The ultimate host identity is a public key pair.[16] In a packet one includes a fixed length hash of the host identity, which also may then be looked up as needed in a global directory service, such as the DNS. In a "four-way" (four packet) protocol the security association is built. Thus, identification is not delegated in any way, although resolution using the DNS will be delegated. The work does not explain how this will occur. Clearly, there is a fixed syntax for these identifiers, designed to meet their functional needs, of providing a public key pair and being hashed into a smaller fixed size appropriate for a packet header. It is important to note, that in this case, neither host identifiers nor their hashed versions are translated into addresses. IP addressing is orthogonal to HIP and assumed to occur at a lower layer.

As a last comment, outside the scope of networking, let us return to Linnaeus and his taxonomy. He defined namespaces with specific functions in mind. His hierarchical naming scheme is universal and describes characteristics of hierarchically nested groups of species. This part of the taxonomy has been expanded to include other top level groupings as well as revising some of what he did. This allows for descriptive naming of every species, assuming one knows the ordering of the hierarchically organized characteristics. Orthogonal to the hierarchy, each species is given a two-part unique name. The first part is Latin-like and unique. The second is a common name, by which non-scientist might name the species. The two parts are separated by a period (".") This allows for global distinction within a flat namespace. We see in this, the recognized need for different namespace decisions to meet differing needs.

6. Conclusion

In this paper, we have proposed a framework or taxonomy of naming systems chosen to clarify the variety design choices available to the naming system designer, in order to take best advantage of engineering tradeoffs. The taxonomy reflects three aspects of naming systems and particular choices within them: (1) namespaces, (2) name assignment, and (3) name resolution. In many cases, the tradeoffs become particularly apparent in the face of distribution, scaling, and delegation, as well as authority and policy control related issues. As a demonstration of the framework, we examined a number of namespace designs with widely varying characteristics, reflecting widely differing choices. In the final analysis it is especially important to make such choices wisely, because naming is only one component of actually completing a task, and therefore should be as efficient and effective as possible without getting in the way.

References

[ASB+99] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, *The design and implementation of an Intentional Naming System,* **Proc. 17th ACM SOSP**, Kiawah Island, SC, December 1999.

[BFM98] T. Berners-Lee, R. Fielding, L. Massinter, **Uniform Resource Identifiers (URI): Generic Syntax**, RFC 2396, August, 1998.

[Cl02] D. Clark, **Forwarding Directives, Associations, Rendezvous, & Directory Service (FARADS)**, unpublished note in the NewArch Project, March 15, 2002.

[DM97] R. Daniel, M. Mealling, **Resolution of Uniform Resource Identifiers using the Domain Name System**, RFC 2168, June, 1997.

[DGH+87] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, *Epidemic Algorithms for Replicated*

[16] The document says that the identity is a public key pair, but it is possible that only one of the keys in the intended identity.

Database Maintenance, **Proc. Sixth Symposium on Principles of Distributed Computing**, Vancouver, B.C., Canada, August 1987, pages 1-12. Reprinted in **ACM Operating Systems Review 22(**1): 8-32, January 1988.

[DVI+99] L. Daigle, D. van Gulik, R. Iannela, P. Faltstrom, URN **Namespace Definition Mechanisms**, RFC 2611, June, 1999.

[FKS+95] P. Francis, T. Kambayashi, S. Sato, S. Shimizu, *Ingrid: A Self-Configuring Information Navigation Infrastructure*, **Proc. 5th World Wide Web Conference**, December, 1995, pp. 519-537, Boston, MA

[Hal85] R. Halstead, *Multilisp : A Language for Concurrent Symbolic Computation*, **Trans. On Programming Languages and Systems**, **7**(4), pp. 501-538, October, 1985.

[Lam86] B. Lampson, *Designing a Global Name Service*, **Proc. Fourth Symposium of Principles of Distributed Computing,**, pp. 1-10, Minaki, Ontario, 1986, ACM.

[Moc87a] P. V. Mockapetris, **Domain names - concepts and facilities**, RFC 1034, Nov. 1, 1987.

[Moc87b] P. V. Mockapetris, **Domain names - implementation and specification**, RFC 1035, Nov. 1, 1987.

[Mos01] R. Moskowitz, unpublished documents on the Host Identity Payload Architecture, 2001. Currently the work only exists as Internet Drafts.

[Riv95] R. Rivest, **The MD5 Message-Digest Algorithm**, RFC 1321, April, 1992.

[Sal78] J. Saltzer, *Naming and Binding of Objects*, in Rudolph Bayer, et al., **Operating Systems--An Advanced Course**, Springer-Verlag, 1978, pages 99-208, **Lecture Notes on Computer Science 60**.

[Sal82] J. Saltzer, *On Naming and Binding of Network Destinations*, published in **Local Computer Networks**, ed. P. Ravasio et al., North-Holland Publishing Company, Amsterdam, 1982, pp. 311-317. Also available as RFC 1498, August, 1993.

[Sho78] J. Shoch, *Inter-Network Naming, Addressing, and Routing*, **Proc. COMPCON 78 Fall,** IEEE, 1978, pp. 280-287.

[SM94] K. Sollins and L. Massinter, **Functional Requirements for Uniform Resource Names**, RFC 1737, December, 1994.

[Sol98] K. Sollins, **Architectural Principles of Uniform Resource Name Resolution**, RFC 2276, January, 1998.

[VDA+99] A. Vahdat, M. Dahlin, T. Anderson, A. Aggarwal, *Active Names: Flexible Location and Transport of Wide-Are Resources*, **Proc. 2nd Usenix Symposium on Internet Technologies and Systems**, October, 1999.