# A Survey of Naming Systems: Classification and Analysis of the Current Schemes Using a New Naming Reference Model

Subharthi Paul, Jianli Pan, Raj Jain
Department of Computer Science and Engineering
Washington University in Saint Louis
Saint Louis, MO 63130 USA
{pauls,jp10,jain}@cse.wustl.edu

***Abstract* — The Internet has evolved immensely since its inception to being a network of academic and government institutions to its present magnanimous commercial avatar. It is currently deemed to be the single largest resource for information and services. From the times it was managed by central authorities, first by DARPA and then by NSF, the present internet has matured to become a commercially driven, distributed structure with minimal centralized control. The availability of large-scale distributed heterogeneous networks and multiple services makes it important to identify all the computing and non-computing entities and the different services that are a part of it. So when it comes to considering the design for the next generation internet, the naming problem becomes an important issue. In this paper, we try to present a comprehensive, systematic and objective discussion on the naming problem. We present a three-dimensional model of naming to outline some of the major research contributions in this area. The model includes three planes: management plane, transport plane and control plane. We classify all the current naming schemes into these three planes according to the difference in focus of the schemes. A detailed analysis and comparison of these schemes is given according to the three-plane classification model. We try to provide a wholesome discussion on every aspect and dimension of the naming problem and try to make our discussion objective through evaluations, comparisons and evolutionary walkthroughs, wherever relevant. The objective of this study is to gain better understanding of past naming systems to allow designing future naming and addressing schemes for the Next Generation Internet.**

***Index Terms*—Identity, Name, Naming, Addressing, Next Generation Internet, Next Generation Network, Identity location split**

The authors are with the Department of Computer Science and Engineering, Washington University in Saint Louis, MO 63130, USA. This work was sponsored by a grant from Intel Corporation.

## Table of Contents

# Table of Figures

# Table of Tables

# 1. Introduction

WITH the impressive development of the modern Internet over the last few decades, the connectivity, reachability and accessibility of the internet has seen an exponential growth. It is the single largest resource for information and communication in the present world. Along with the evolution of the internet, the number of network-able devices has seen manifold increase, both in number and diversity. Individuals can now connect to the internet through a diverse plethora of devices, e.g., desktop personal computers with Ethernet adapters, laptops enabled with wireless function, mobile PDA or WAP embedded mobile phones, etc. Numerous new and exciting services are being offered over the Internet and are testing its capabilities to the limit. All these computing and non-computing entities, resources and services, and their users need some mechanism for identification and address. Informally, this process of assigning names or identities to objects is called *naming*, and the mapping of names to a specific address and route, and arriving at the object with specific resources is called *name mapping* or *name resolution*.

A good naming scheme should be able to provide human-friendly names for entities, easy and extensible name syntax, structural and systematic name management , scalable and secure system architecture, name consistency, support for mobility, and easiness of name lookup and service discovery. These essential aspects form a series of very basic and crucial evaluation criteria, which are very important factors in designing a naming scheme, and they will be addressed in this survey in greater details.

Since 1980s and 1990s, before the deployment of the DNS (Domain Name System) [46] as the de facto naming standard on the Internet, a lot of research projects had already been undertaken on the naming problems for different kinds of network architectures running on different protocols, such as Grapevine [10], Clearinghouse[51], and V-System [13] [14]. These primitive schemes tried to solve specific problems relevant at that time, and inevitably could not meet all the requirements of a good naming scheme, especially in today’s context. URI [8][9][19][45] and DNS seemed to be the only naming scheme that became popular and were widely adopted and used. However, with new services and new requirements being introduced, the current URI and DNS dominant Internet is faced with some serious challenges as to how the static naming scheme could name dynamic objects and keep the names consistent, how to address the mobility and multi-homing issues, how to solve the security problem, how to support service discovery, and how to provide anycast and multicast functionalities.

In this survey, we try to put forth an objective discussion of most of the naming schemes of the past decade, analyze them and put them into a three-dimensional naming reference model (shown in figure 1) according to the naming issue they address. The model will be explained late in section 3. We also compare different schemes according to a series of evaluation criteria. By doing this, we hope to better understand the essence of the naming problem, and gain a more precise view on what the future naming schemes should be like.

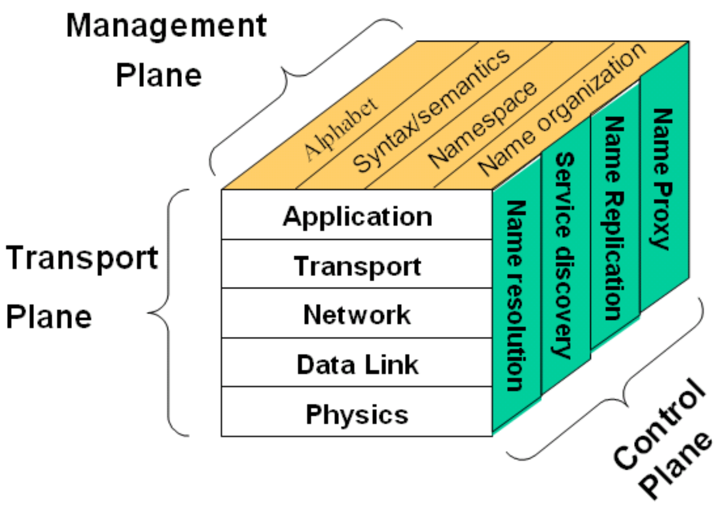


Figure 1: Naming reference model

The remainder of this survey is organized as follows. Section 2 is a brief explanation of some of the terminologies related to naming and addressing that we use in this paper. In section 3, we explain the three-dimensional naming reference model, the respective function of the three planes, and why we use this reference model to study various naming systems. We then analyze the schemes belonging to these planes accordingly in part 4, 5, 6, and 7. The comparison and the evaluation criteria also are given in these four sections. After we discuss some non-typical naming schemes in section 8, we then move on to discuss the future directions of naming schemes in section 9 and finally, we conclude the survey in section 10.

## 2. TERMINOLOGY

**Name:** *A syntactic entity that denotes an object* [11].
**Namespace:** *The collection of all valid names.*
There is also another alternative definition in RFC 2611 [18] which give the meaning as the collection of unique identifier that have already assigned. Comparatively, our definition is more practical and general.
**Address:** *An intermediate identifier between a name and a route* [25] *to the destination on which the resource or the service resides.*
**Address space:** *The collection of all the valid addresses.*
**Naming reference model:** *The three-dimensional reference model we propose to illustrate the key research issues on the field of naming, which includes the "Management Plane", "Transport Plane" and "Control Plane".*
Not all the three plane function are addressed and provided by every naming scheme. Some only provide part of a specific plane function.
**Management plane of naming:** *The plane that is in charge of the rules of the alphabet, syntax/semantics, namespace, name assignment, and name organization.*
It decides the syntactic and semantic rules of the naming of different objects. This plane focuses on the management function of the pre-defined rules of naming.
**Transport plane of naming:** *The plane that is in charge of the identity-location decoupling among different layers, name resolution, name binding, routing, mobility and multi-homing, and security policies.* Transport plane functions focus on name mapping and transporting data to the destination designated by name.
**Control plane of naming:** *The plane that is in charge of the registration of clients, name lookup, service*

*discovery, name server replication, and name proxy.* Control plane functions focus on structural control keeping the naming services available and effective.

**Name resolution:** *The mapping from a name to an address, or more generally, the mapping from the namespace to address space.*

**Name organization:** *The structure of the name servers and the namespace partition policies.* It could be centralized, hierarchical, or fully distributed.

**Name extensibility:** *The ability of naming systems to support extension of names, i.e., whether the name is expandable.* For example, most of the descriptive names are expandable to accommodate new attribution of services.

**Name scalability:** *The ability of naming systems to support scalable structure of name servers.* Good naming scheme should be scalable and suitable to be deployed in large-scale heterogeneous networks.

**Name consistency:** *The ability of naming systems to keep the names consistent across the whole system* while allowing replication, caching,, and support for dynamic names.

## 3. Reference Model of Naming Systems

Generally speaking, a naming scheme may consist of different aspects such as scalability, mobility of end-hosts and hence changes in name mappings, mobility of services, security, etc. Some of these aspects may be related while others are independent of each other. Moreover, different naming schemes always try to solve specific problems concerned with specific aspects under specific conditions. For example, Grapevine [10] was designed in1980s to build a primitive electronic mailing system for a relatively small network. Currently there is no single naming scheme that addresses each of these aspects fully and comprehensively. Furthermore, the development of technology and new research progress in naming has introduced new functions and structures.

We propose a naming reference model to provide a comprehensive and wholesome discussion on every aspect of naming and addressing systems that we deem relevant. Our study captures some of the most popular naming and addressing schemes and classifies them into logically independent planes of the reference model. By doing so, we make an effort to delineate the problem into orthogonal planes, such that they may be more logically meaningful and individually addressable.

Ahmed [2] gives a survey on the service naming problem on large-scale and multi-domain networks. Some schemes related with service naming are introduced and compared according to a series of criteria defined by the author. However, the author did not introduce any clear model for the basis of the discussion, and simply used a set of flat criteria for the schemes that may belong to different functional planes. This makes the survey incomplete in terms of coverage as well as logical classification.

By introducing the naming reference model, we try to cover many more aspects of the problem as relevant in today’s Internet scenario.

As shown in figure 1, our naming reference model is a three-dimensional model. It includes three planes: management plane, transport plane and control plane. The three dimensional model does not however imply that all the attributes in each plane is covered by the current naming schemes, but rather implies a directive as to what a good naming scheme should essentially implement. In the following parts, we will discuss the function of these three planes in greater detail.

## 3.1 *Management Plane*

This plane are relates to the format, grammar, and management of name assignment and name organization. For any naming scheme, these aspects need to be defined and designed before the scheme can be implemented.

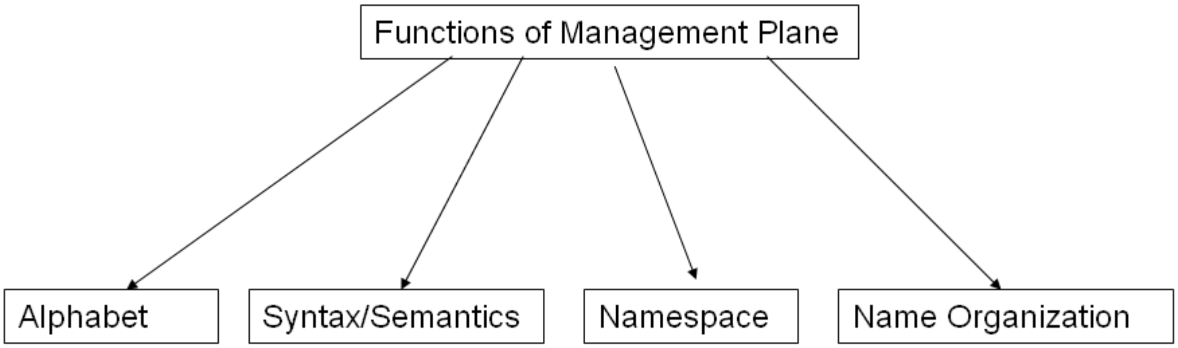


Figure 2: Functions of management plane

As illustrated in figure 2, the management plane is responsible for the following functions:

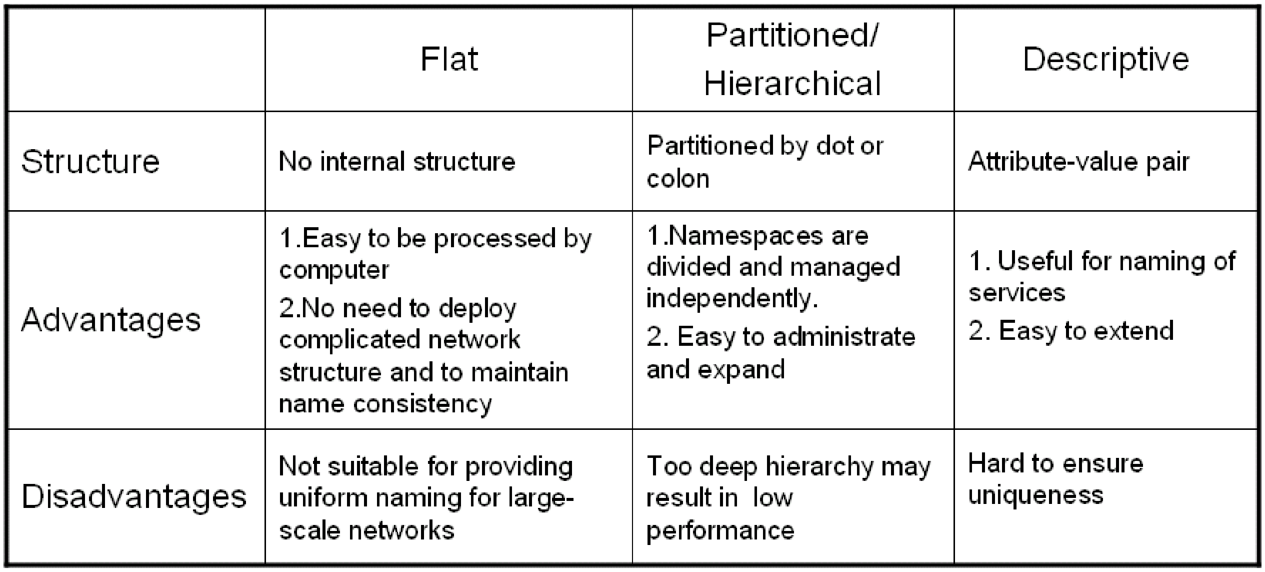
- Alphabet: The alphabet used to constitute a name can include digits, letters, ASCII codes or Unicode. It can also be a combination of parts or all of the above depending on the design of the protocols.
- Syntax/semantics: The syntax/semantics of a naming scheme concerns the organization of name components to form a name and the meaning of these components. The semantic aspects relate to whether the scheme is human-friendly and easy to understand. According to syntax/semantics, the structure of names can be characterized into three ways: flat, partitioned, and descriptive.

  *A Flat name* has no internal structure and generally consists of a non-human readable string to constitute a unique identity. For example, UUID [40] uses a string of 128 bits to denote a name.

  *Partitioned/Hierarchical names* use a "." or ":" to divide different parts of a name. The sequence of the parts between the divider may represent the domain and sub-domain order (as Clearinghouse [51] and URL [9]). These partitioned and hierarchical names have the advantages such that the size of individual databases in a sub-domain name server is reduced and partitioned namespace are and easy to administrate. However, such partitioning can also result in lower performance than a flat namespace if the depth of the hierarchy is large thereby increasing the cost of searching and mapping a name. That explains why Clearinghouse and DNS restrict the depth of the hierarchy.

  *Descriptive names use* a series of "attribute-value pair" to name an object. These attribute-value pairs describe the properties of the object to be named. Descriptive names are especially useful to name a service. For example, when a user wants to get a specific service from the Internet, but doesn't know where this service is located, he/she could send out a query with the set of properties that he/she deems relevant to the service. The name servers could resolve the user query by looking descriptive names tables and returning all the services with the requested properties. Several naming schemes, such as Profile Naming Service [55], INS [1], and Solar [12], use descriptive names. Intuitively, the partitioned naming scheme (such as URL [9]) could also be seen as a form of descriptive name with strict structural constraints. Table 1 gives a brief comparison of the structure of the names. The advantage and disadvantage of each kind of structure are also illustrated and compared.

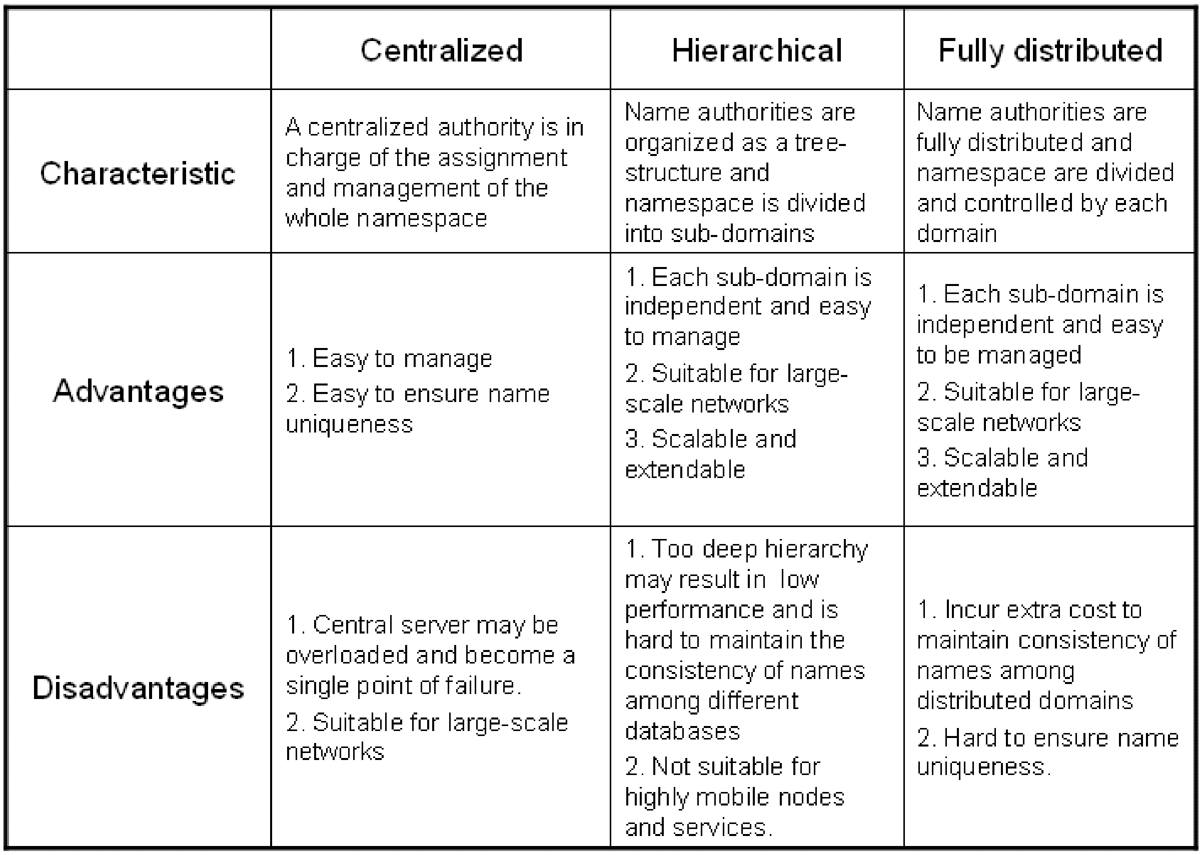
Table 1: Comparison of the organization of name components



| | Flat | Partitioned/ Hierarchical | Descriptive |
|---|---|---|---|
| Structure | No internal structure | Partitioned by dot or colon | Attribute-value pair |
| Advantages | 1.Easy to be processed by computer<br>2.No need to deploy complicated network structure and to maintain name consistency | 1.Namespaces are divided and managed independently.<br>2. Easy to administrate and expand | 1. Useful for naming of services<br>2. Easy to extend |
| Disadvantages | Not suitable for providing uniform naming for large-scale networks | Too deep hierarchy may result in low performance | Hard to ensure uniqueness |

- Namespace: The design and composition of the namespace are closely related to the syntax of the naming scheme and the organization of the naming systems. For example, URL uses a partitioned hierarchical name and the namespace is split into different domains and sub-domains which can be managed separately. The namespace could be finite or infinite depending on the design. For example, UUID scheme uses 128 bit strings as name, therefore, results in a finite namespace of size $2^{128}$. Most of the other naming schemes have infinite namespace.
- Name Assignment and Organization: A name could be assigned and organized in a centralized, hierarchical or fully distributed manner, and the namespace could be controlled by a *centralized* organization (for example, the social security numbers of people in the United States are managed by the Social Security Administration) or split between *hierarchical* domains (for example, URL [9], INS [1], Solar [12]), or even in a *fully distributed* way (for example, UUID [40]). Table 2 gives a brief comparison of the three different mechanisms of assigning and organizing names. The advantages and disadvantages of each mechanism are also given. The comparison clearly shows that when considering scalability, performance and fitness for large scale network architectures, hierarchical and fully distributed mechanisms are best.

Table 2: Comparison of name assignment and organization mechanisms



| | Centralized | Hierarchical | Fully distributed |
|---|---|---|---|
| Characteristic | A centralized authority is in charge of the assignment and management of the whole namespace | Name authorities are organized as a tree-structure and namespace is divided into sub-domains | Name authorities are fully distributed and namespace are divided and controlled by each domain |
| Advantages | 1. Easy to manage<br>2. Easy to ensure name uniqueness | 1. Each sub-domain is independent and easy to manage<br>2. Suitable for large-scale networks<br>3. Scalable and extendable | 1. Each sub-domain is independent and easy to be managed<br>2. Suitable for large-scale networks<br>3. Scalable and extendable |
| Disadvantages | 1. Central server may be overloaded and become a single point of failure.<br>2. Suitable for large-scale networks | 1. Too deep hierarchy may result in low performance and is hard to maintain the consistency of names among different databases<br>2. Not suitable for highly mobile nodes and services. | 1. Incur extra cost to maintain consistency of names among distributed domains<br>2. Hard to ensure name uniqueness. |

## 3.2 *Transport Plane*

The transport plane deals with the architectural issues in name resolution framework. Every naming system requires a framework wherein the names are resolved to concrete objects. The key design issues in such an architectural framework are scalability, security, support for mobility and heterogeneity, and administrative and infrastructural overheads.

The current Internet depends on a static domain name (DNS) to identify an object and map it into IP address. IP addresses are semantically overloaded with the dual role of being a locator as well as an identifier [63]. Caching and replication mechanisms make it difficult for DNS to support emerging mobility and multi-homing requirement. Mobile IP [31][53][54] tries to decouple the name and the locator with a HoA (Home Address) and CoA (Care-of Address), in which, HoA acts as an unique identity and CoA acts as a locator thatchanges as the mobile host changes its point of network attachment. HIP [3][36][48][64] uses Host Identity and Host Identity Tag as the identity. Here, Host Identity is a public key and Host Identity Tag is a hash of the public key. HIP detaches the naming function from the IP address and supports mobility and security. HIP is one of the most promising recent advancements in this area and the HIP working group at IETF is presently working on various HIP extensions in order to cover all the relevant issues.

In discussing the transport plane, we make an effort to study various naming scheme architectures which try to address the problems of IP semantic overloading, naming and addressing across heterogeneous contexts, and scalable addressing of flat namespaces. In section 4, we analyze and compare several schemes on this plane, such as HIP, FARA [16], DoA [71][70], Layered Naming Architecture [4], I3 [68], Hi3 [24][49], 4+4 [76], and Plutarch [17] among others.

## 3.3 *Control Plane*

Control plane functions are concerned with the efficiency, reliability and security issues of maintaining the architectural structure the naming system. Most of the naming schemes have some architectural structure with the exception of UUID, which uses a 128 bits flat name and does not specify any name resolution architecture. Since for most of the large-scale, heterogeneous networks, it is necessary to construct and maintain a unified and structural name organization. Some naming schemes adopt a hierarchical structure while others are fully-distributed. Both of these two structured name organizations have many advantages. For example, namespaces are divided into independent small domains and easy to maintain and extend; different parts of the system are deployed independently and donot influence each other, and will not cause any single point of failure problem.

However, compared with flat names without any structure such as UUID, hierarchical or fully-distributed naming schemes could bring extra costs, e.g., the overhead of maintaining the structure of the domains and the management cost of internal and external communications. So in the control plane function of naming system reference model, our discussion focus on how to maintain the efficient, reliable and secure architecture of naming systems. Related to these problems, there are series of issues such as name resolution, service discovery, name caching, name replication, and name proxy. For example, naming system such as URI, in which caching and name lookup function are provided, and naming system such as INS and Solar, in which service discovery mechanism are provided, in each domain, there are one or more name servers performing the specific functions, so these servers must handle all the control functions and routine communications between the clients and the servers. At the same time, the servers should keep track of the names changes of the objects and update these information in their database and probably multicast it to the neighbors to keep the information correct and consistent among the whole networks as soon as possible [41]. Moreover, to keep the naming system reliable and available, there may be some replications between these servers. And some name proxies servers may be provided to address the NAT problems in naming systems [70][71].

So from an architectural view, for a naming scheme designed for large-scale and heterogeneous networks, the control plane function is necessary to keep the naming service available, reliable, efficient and secure. In the following section, we will address these function and issues in detail.

To summarize, in this section, we discussed the naming reference model and the three planes function separately. We also address the reason why we propose this model and its significance of serving as a guide for our future research. Through this reference model, we also could dissect and classify the current considerable researches related on naming. In the next part, we will discuss the current available schemes focused on the functions of three planes respectively. For every plane, to introduce the schemes, we firstly give brief introduction and then try to give a set of evaluation criteria, by which we could compare them and get a clearer understanding of different schemes.

# 4. Brief Introduction And Analysis of Management Plane Naming Schemes

Since 1980s, dozens of naming schemes that focused on the management plane functions were proposed based on the technical requirements and constraints of that time. Different naming schemes were designed for meeting different application needs. Some focus on specific applications for small-scale networks (for example, Grapevine [10] and Clearinghouse [51]), and some address the IPC (for example, XRL [73]), the file system (NFS [67] and CIFS [66]) while others were proposed for use in large-scale Web Services (HFN [6]) and generalized resource naming in the internet (for example, URI). Some new proposals were made for service naming, mobility, multi-cast and any-cast problems in large scale networks (for example, INS and Solar). In this section, we will briefly discuss some of the naming systems which are relevant to the research on this area. Basically, according to our three-dimensional naming reference model, most of these schemes focus on the function on the *management plane*, that is to say, the syntax/semantics, name assignment and management, and name system organization. Based on this, in section V, we propose some evaluation criteria upon which we shall compare the various schemes in order to gain relative comparative understanding of the schemes together.

### *4.1 Grapevine*

Grapevine [10] was one of the first large-scale distributed naming systems, proposed in 1982 by Xerox. It was designed to be a computer mail delivery service for the Internet. It was a prototype of the present email system. It also offered authentication and access control services for the clients.

Grapevine maintains a "*registration data base*" on name servers for mapping names (users, machines, or services) to the information about the name. Every entry in the database is called an *RName*. The namespace organization of *RName* is a two-level hierarchy. The form is like "*F.R*", where R is the *registry* name and F is the object name within the registry. Registries represent organizational, geographical, or other potential administrative partitions. Registries are the unit of replication and distribution in Grapevine.

There are two types of Grapevine servers: *registration server* and the *message serve*r. They are two different logical entities but may reside on the same Grapevine server.

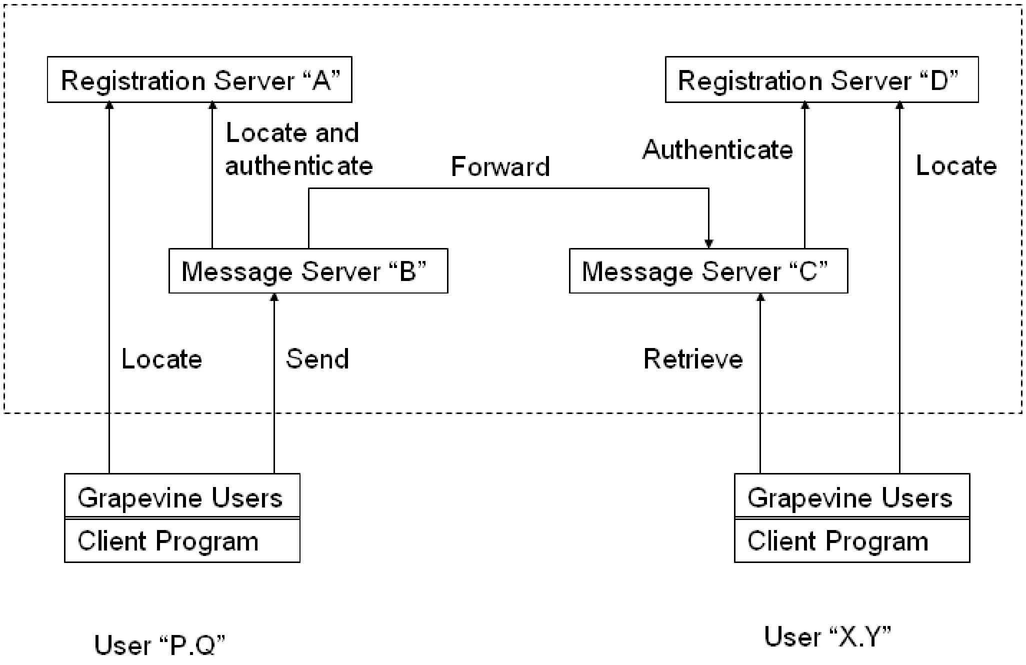


Figure 3: A Grapevine example

As an example, figure 3 gives the mail delivery procedure of Grapevine. Suppose the user named "P.Q" want to send an electronic mail to user named "X.Y". When the user has prepared the message using a certain client program, the client program will call the mail delivery function GrapevineUser to contact some registration server (in this example, it is registration server "A"). GrapevineUser also uses Grapevine location function to locate any message server (in this example, it is message server "B"). Server "B" will also use resource location function to determine the recipient's best inbox site (it is message server "C" in this example), server "C" will buffer the mail message locally waiting for X.Y to retrieve. The registration servers, in which the registry names reside, play an important role during the process of name resolution and locating the message service.

Grapevine also provides replication functionality. It provides a *pseudo registry, GV* (Grapevine) that contains information on all other registries. GV registry is replicated in every registration server. The GV registry controls the distribution and replication of the registration database, and allows clients to locate appropriate registration servers for particular RNames.

### *4.2 Clearinghouse*

Clearinghouse (CH) [51] was proposed in 1983, also by Xerox. It evolved from the early version of Grapevine [10]. It is used mainly to name mailboxes, users and servers. It addresses the problem of naming and locating objects in a distributed environment by providing a decentralized server named "Clearinghouse" for supporting the naming of these distributed objects. The namespace of the Clearinghouse is managed by a collection of Clearinghouse servers. Each server manages a portion of the global namespace, but the namespace is not strictly partitioned between servers. Clearinghouse also provides replication between servers. Clearinghouse provides a three-level hierarchical namespace, and Clearinghouse names are of the form "*L:D:O*", where L is the local name, D is the domain, and O is the organization. Clearinghouse maps each name into a set of descriptive properties of the form like <*PropertyName, PropertyType, PropertyValue*>, the *PropertyType* could be item or group. As shown in the example in figure 3, each *PropertyName* must be globally registered through a naming authority. Since the Clearinghouse bind the name with a set of properties, Clearinghouse can store arbitrary information about any named object.

### *4.3 V-System*

V-System [13][14] was an experimental project of Stanford University in 1989. It is a distributed operation system designed for a cluster of computer workstations that are connected by a high-speed local-area network. V-System is not the name of the naming scheme, but within it, it uses some naming mechanism to name the files and programs, and these new mechanisms are of importance for the development of naming systems. So in this paper, we refer to "V-System" as the name of the naming scheme.

The naming scheme used in V-System is a three-level naming architecture that consists of *global*, *administrational*, and *managerial* levels. It affords a decentralized approach to the lower levels, in which naming is handled directly by the managers of the named objects that implements the object, enabling the name mapping to be performed as part of the operation that refers to the object by name. So there is no need to handle a separate lookup operation as in DNS. This mechanism of integrating naming with object management increases the efficiency of the name mapping and the reliability of the system. Moreover, the naming system provides client-based *caching* and *multicast* mechanism to optimize the performance and fault tolerance.

Figure 4 shows a concrete example of the integration of naming with object management, and three-level structure of V-System. In this example, “%edu” is the directory of educational institutions, whose entries include “%edu/wustl”, for the organization, Washington University, and “%edu/uiuc” for University of Illinois. “%edu/wustl” is the highest –level administrational directory for Washington University, and “%edu/wustl/cts” is the root for a subordinate administration, called Computer Technology Service group. Since the managerial directory is implemented by the object manager that implements the objects named in the directory. The sub-trees rooted at the directory “%edu/wustl/cts/bin” and “%edu/wustl/cts/lib” are both implemented by CTS file server1, which thus covers all the names with prefixes “%edu/wustl/cts/bin” and prefix “%edu/wustl/cts/lib”, and accordingly, file server 1 stores all the files and directories under these two sub-trees. So is the situation in server 2.

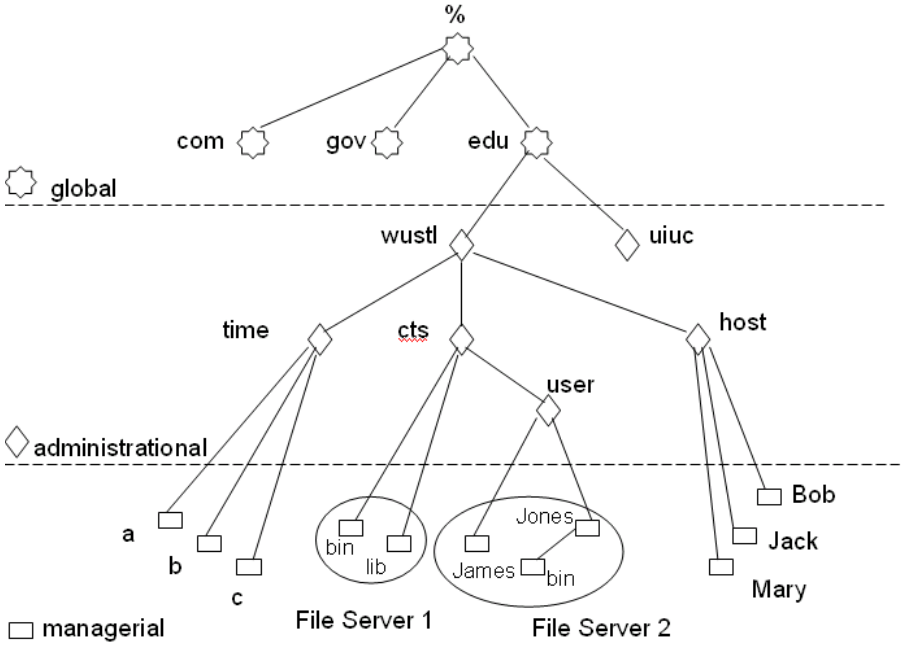


Figure 4: An example of V-System

## 4.4 URI

URI (Uniform Resource Identifier) [8] is standardized by IETF and is a naming scheme that is running on the current Internet as a de facto standard. It could be seen as the most successful naming scheme that is well-know and widely accepted by the industry. URI consists of two parts: URL (Uniform Resource Locators) [9] and URN (Uniform Resource Names) [19][45].

URL defines a compact string representation for a resource accessible via the Internet. URLs are used to “locate” resources by providing an abstract identification of the resource location. Having located a resource, a system may perform a variety of operations on the resource, as might be characterized by such words as “access”, “update”, “replace” and “find attributes”. In general, only the “access” method needs to be specified for any URL scheme. The URL general syntax is as follows:
<scheme>: <scheme-specific-part>
For the “scheme” part, URL defines a series of schemes such as “ftp”, “http”, “gopher”, “mailto”, “news”, “nntp”, “telnet”, “wais”, “file” and “prospero”. As a more common and concrete example of a URL is as follows:

http://cse.seas.wustl.edu/research
In general, when a URL is given, the location of a specific object is decided, and the URL is always mapped by DNS (Domain Name System) into an IP address of the resource location.
URNs try to provide location-independent and persistent resource identifiers for entities on the Internet. Different from the URL trying to "locate" the resource, URNs are intended to be designed to make it easy to map other namespaces into URN-space.
URNs are distinguished from URLs by the initial "urn:" followed by a Namespace identifier (NID), a colon, and a namespace-specific string. The general syntax is as follows:
"urn:" <NID> ":" <NSS>
NID is the namespace identifier, and <NSS> is the Namespace Specific String. By now, there are twenty formal registered NIDs such as: "ietf", "pin", "issn", "isbn", etc. A more concrete example of URN is as follows:
urn: isbn: 0-296-37351-2
By defining syntax like this, it is easy to map other namespace such as ietf, isbn, issn, even UUID namespace into URN space.

### *4.5 XRL*

XRL (XORP Resource Locators) [43] is used to mediate the IPC (Inter-Process Communication) within XORP (eXtensible Open Routing Platform). XORP is an open source route platform for testing new routing software and new routing protocols. To provide robustness and extensibility for routers which run under different platforms using different operation system, an inter-process communication mechanism is needed. XRL realize this mechanism and is able to provide a consistent and transparent interface irrespective of the underlying transport mechanism used, for instance, SNMP or HTTP. XRLs are responsible for describing an inter-process calls and their arguments. It comprises the protocol family to be used for transport, the arguments for the protocol family, the interface of the target being called and its version, the method, and an argument list.

XRLs' name structure is very similar with URLs. Instead of representing a hierarchical resource location, XRLs' syntax describes a human-readable form comprised of inter-process procedure call protocol and its parameters. Two formats of the same XRL are depicted in figure 5: Unresolved form and Resolved form. In the unresolved form the protocol family is set to "finder" (as shown in figure 5(a)), and the protocol parameters set to the target name that the XRL call is intended for. Initially, a process passes the unresolved XRL to the "finder", which returns the resolved form (as shown in figure 5(b)) with appropriate protocol family and protocol family arguments. After the resolution, the resolved forms of XRLs are typically maintained in a client side cache.

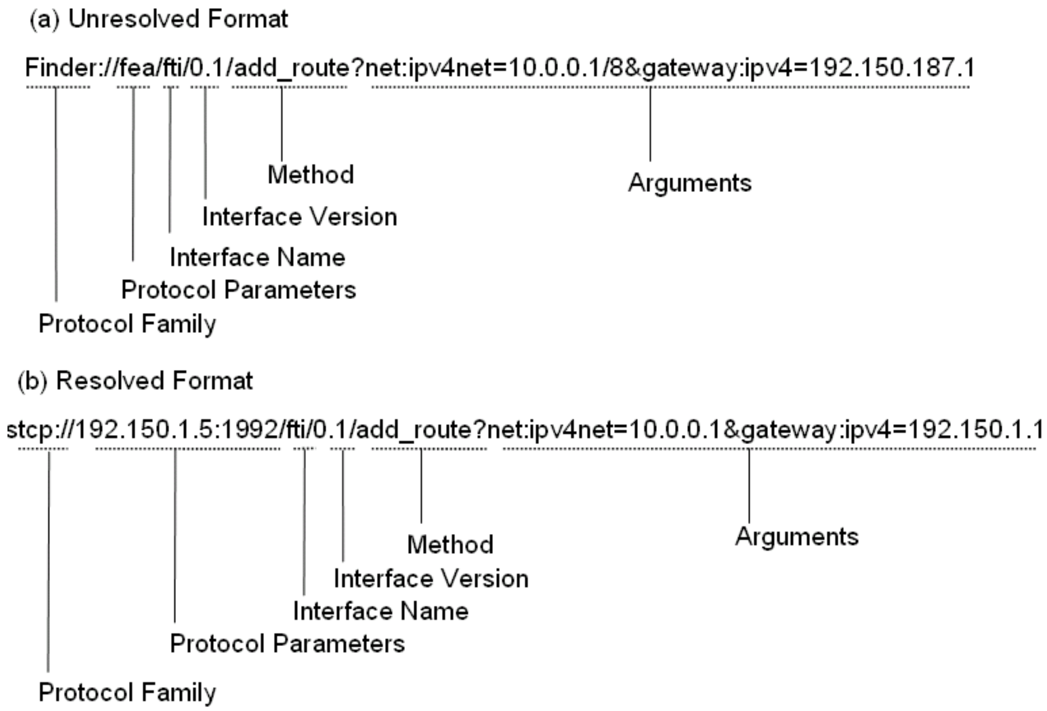


Figure 5: Example of two name formats in XRL

The “finder” is actually a central process which performs the name resolution in XRL and keeps track of all the registered target names and the communication protocols they support. The “finder” replaces the target name with a network address and “finder” with appropriate protocol, and sends it back the client. The client uses this resolved XRL to access the desired resource.

### *4.6 UUID*

UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier) [40] are 128-bit identifiers and require no centralized authority for the administration and registration processes. So the generation of ID on demand can be automated and used for many purposes.

UUIDs are of a fixed size and reasonably small compared to other alternatives, and hence easy and less costly to be allocated, sorted, hashed, stored, and programmed.

UUIDs are generated locally and uniquely by a combination of components. UUID contains a reference to the network address of the local host, a timestamp, and a randomly generated component. So the uniqueness of these identifiers is guaranteed. Because of this uniqueness and persistence, UUIDs make excellent Uniform Resource Names with low cost. An example of string representation of a UUID is as follows:
Uuid: f81d4fae-7dec-11d0-a76-00a0c91e6bf6

A UUID is unique across both space and time. Since it has a fixed size and contains a time field, it is possible for the values to rollover (estimated to be around A.D. 3400). A UUID can be used for multiple purposes, from tagging objects with extremely short lifetime, to reliably identifying very persistent objects across a network.

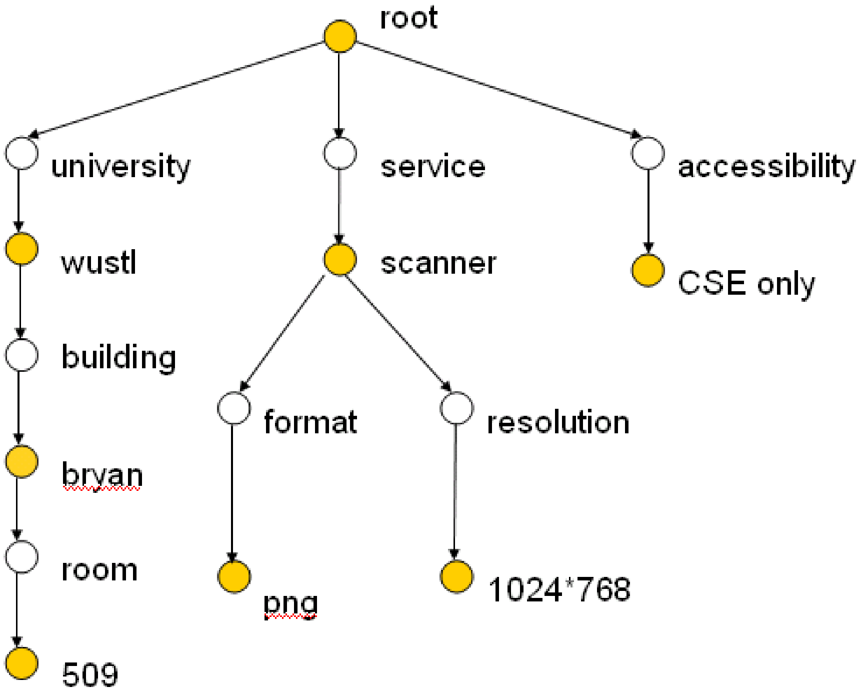
UUID does not specify name resolution architecture. Applications that use UUID for naming must provide their own name resolution mechanism. e.g., in “Layered naming architecture”, ULD->SID->EID mapping structure, the SID use UUID.

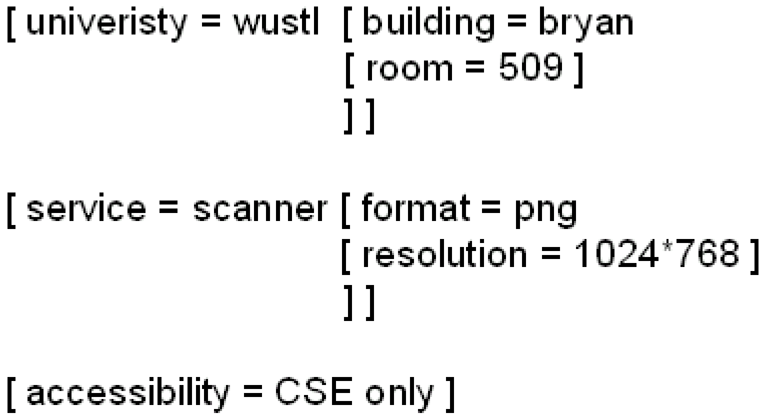
### *4.7 INS*

INS (Intentional Naming System) [1] was introduced in 1999 as a research project at the MIT Laboratory for computer science. Basically, it is a resource discovery and service location system for dynamic and mobile networks, which use a simple language based on attributes and values for its names. And the users of the application could use the descriptive language consisting of “attributes and value pair” to describe what kind of service they want, i.e., their intent, but not precisely where to find the resource, i.e. the precise IP address of the destination host. Different from conventional routing strategy based on IP address, the routing function in INS is decoupled from IP address by implementing a late binding mechanism that integrates the resolution from names to address and the message routing. The late binding mechanism makes it possible to support mobility when endpoint keeps moving and the name-to-address mappings keep changing. The name resolution function is fulfilled by an overlay application level of INR (Intentional Name Resolver), which is a self-organized network formed to perform the name mapping, service discovering, as well as the message routing.

As for the syntax of INS name, INS integrates resource /service name and description into tree-like hierarchical descriptive names of attributes and values, called Intentional Names or name-specifiers. As a name example, in figure 6, it defines a name specifier for a public scanner, and all the properties of the scanner are organized into a tree (figure 6 (a)). It could also be represented into a wire format (figure 6 (b)).

INS names are human-friendly and have fewer structural restriction than other naming scheme such as URLs and XRLs. INS names also have no restriction on the number of levels or the numbers of the children of a node, and hence the name is extensible.





```
[ univeristy = wustl  [ building = bryan
                        [ room = 509 ]
                        ] ]

[ service = scanner [ format = png
                      [ resolution = 1024*768 ]
                      ] ]

[ accessibility = CSE only ]
```

Figure 6: Example of the two representation of an INS name

### *4.8 Solar*

Solar [12] was an experimental project of Dartmouth College in 2003. It allows resources to advertise their names according to their context and allows applications to query these context-based names. Similar to INS, Solar also uses descriptive names whose attributes and value could be changed as needed. One of the typical usages of these attributes is for the location of mobile node. A major difference between INS and Solar is that Solar supports dynamic names (names changing all the time) while INS does not. An example of such dynamicity could be: Jack has a mobile laptop, when he work with it in office in Bryan Hall room 405, it can be named:
[Device="Laptop", Name= "Jack-PC", Owner="Jack", Room= "405", Building = "Bryan Hall"].

When Jack moves to room 217 of building Cupple II, the name will be changed to:
[Device="Laptop", Name= "Jack-PC", Owner="Jack", Room= "405", Building = "Bryan Hall"].

From the architectural point of view, Solar uses an extended version of INS as resolution mechanism. It provides dynamic naming by introducing a "solar layer" over INS architecture. Mobile services connect to a solar "Planet" (actually a proxy server between Solar client or server, and INS networks), who resolves a dynamic name into a static name, and the static name is resolved and disseminated by INS networks. The architecture could be illustrated in figure 7. As we could see from the figure, the proxy acts as an intermediate. When name changes (supposing Jack moving to another room in another building), the dynamic name will be processed by proxy server and resolved into a static name before it goes to the core INS network for further operation. After that, new name information of the mobile node will be notified to the client by some proxy near the client.

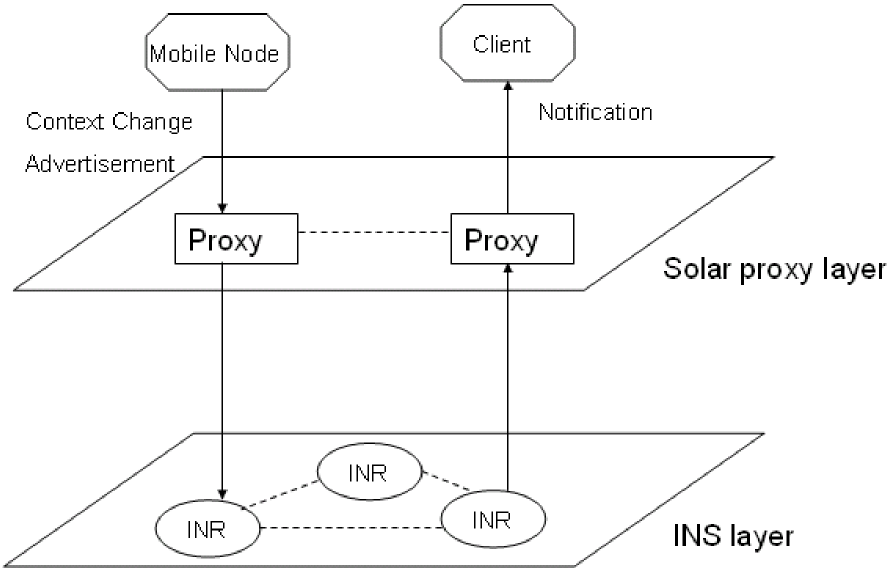


Figure 7: Architecture of Solar

To summarize, in this section, we have given a brief introduction for eight different naming schemes focusing on the functions of the *management plane*. The basic characteristics of the systems are given before we could compare and evaluate them. In the following section, a series of key evaluation criteria are proposed and we will analyze every scheme according to these criterias respectively.

## 5. Evaluation and Comparison of Management Plane Naming Schemes

To provide an in-depth analysis and horizontal evaluation of the various naming approaches, so that we could use them to guide us for the design of future naming schemes of the next generation Internet [30], we define a set of evaluation criteria based on the management plane functions that could characterize the essence of each of these different naming schemes.. These criterias have also been addressed by the previous work separately and unsystematically, however we will collect all the properties and discuss them together. Ahmed [2] has tried to give a set of criteria for some service naming in multi-domain networks. However, it concerns only naming of services, but not general naming problems that include naming of people, services, and other common objects. Secondly, because it does not use a systematic naming reference model, all the criteria and related factors are discussed indistinctly. To avoid this ambiguity, we use a three-dimensional naming reference model so that the different function and properties are differentiated, and they could be discussed more clearly. For this section, we will give a series of evaluation criteria to appraise the naming schemes related with management plane function in naming reference model. A brief comparison of these schemes according to these criterias will be given as well.

### *5.1 Evaluation criteria*

### *5.1.1 Alphabet*

The alphabet means the constituent of a name. Depending on the design, it could be digits, letters, ASCII codes, Unicode, or the combination of all above. Since Grapevine and Clearinghouse are generally used as electronic mail delivery service, the alphabet used is non-null character strings in partitioned forms separated by a dot. The name defined by V-System is like “%edu/wustl/cse/fileserver/bin/listdir”, where the character “%” designates the root and the “/” character separates the name components. As in section IV where we give examples of URL and URN names, we could see that the alphabet of URL and URN include both ASCII and non-ASCII codes, in which the octets could be encoded by a triplet of character consisting of the character “%” followed by two hexadecimal digits [9]. The XRL’s alphabet is similar with URL, and it is composed of ASCII and non-ASCII codes. UUID is 128-bit identifier, so the alphabet includes digits of “0” and “1”, which could be denoted by hexadecimal digits with “-” for representation. The name identifiers of attribute-value pair in INS and Solar are free-form strings [1], which are consisted of ASCII codes.

### *5.1.2 Namespace*

According to the definition, namespace is *the collection of all valid names.* It is decided by the alphabet and the combination rules of the alphabet. The alphabet could be digits, letters, ASCII codes or Unicode depending on the design of the protocols. The combinatorial rules decide whether the namespace would be finite or infinite. For example, the address space of IPv4 [20] is limited and face the danger or depletion. Theoretically, the IPv6 [21] address space is also limited but it is much larger than the current requirement. For telephone service, the digits used for a telephone number are also increasing.

Similarly, for the namespace design, it is also necessary to consider the namespace problem in advance before it could meet the depletion issue as in IPv4. It is desirable to adopt a naming scheme that would have a namespace large enough to support a future unexpected explosion of requirement for the namespace,. Also, the uniqueness of the names for the current and future entities must be ensured. An example of limited namespace will be UUID in which it uses 128-bits as its unique name and forms a limited namespace of size of $2^{128}$. So the UUID is not suitable to perform as a global and uniform naming infrastructure for the future Internet, but it may complement the global naming schemes.

### *5.1.3 Name Syntax and Semantics*

The syntax and semantics of a naming scheme concerns the organization of name components to form a name, and the meaning of every component. The structure of name components can be characterized in three ways: flat, partitioned, and descriptive.

A *flat name* has no internal structure, it is not easily readable and comprehensible by human-beings, but it is easier to be processed by machines such as computers, and has no extra complicated structures. The *partitioned/hierarchical names* denote the split of the namespace to independent sub-domains, which could be administrated independently and efficiently. However, if the hierarchy is too shallow, the scale of the naming system could be limited, on the contrary, if the hierarchy is too deep, it could lead to lower performance. The *descriptive names* use “attribute-value pair” to denote an object and it is much more flexible and suitable to name services than name entities such as machines, people etc.

**Grapevine—** Grapevine uses two-level hierarchical names called *RName*, which is in form like *“F.R”*. R is the *registry* name and F is the object name within the registry. Registries represent organizational, geographic, or other potential administrative partitions.
**Clearinghouse—** Clearinghouse has a three-level hierarchical namespace in form of “*L:D:O*”, in which L is the local name, D is the domain, and O is the organization. The organization and domain are constituted by

Clearinghouse servers, which resolve the name required by the Clearinghouse clients.

**V-System—** The namespace of V-System is also a three-level hierarchical structure. The architecture consists of *global*, *administrational*, and *managerial* levels.

**URI—** As the description in the former section, URI includes URL and URN. URL provides a mechanism to locate a resource by a given location, while URN specifies a resource by a globally unique name and is persistent even when the resource stops to exist or become unavailable and inaccessible. URL uses DNS to denote the destination entity in which the resource resides, and the DNS name is organized as a hierarchical structure which does not specify a fixed depth of hierarchy. The hierarchy also split the namespace among the domains which are easier to manage separately. On the contrary, URN proposes to label a resource with a persistent identifier, and the identifier is formed from a set of pre-defined namespaces, each of which may have their own name assignment procedure, syntax, and name structure.

**XRL—** The syntax of XRL is similar with URL, but XRL does not specify hierarchical resource location as URL, and its "protocol parameters" are resolved by the central process "finder".

**UUID—** UUID is a 128-bit *flat* identifier. It has no internal structure, and thus there is no centralized authority to manage the namespace. The identifiers are generated distributively and there is a specific algorithm to ensure that the name generated is unique.

**INS—** The name INS uses is called an Intentional Name or name-specifiers. Intentional Names are constituted of *descriptive* names of attributes and values organized into a tree-like format.

**SOLAR—** Solar works as an extended version of INS. It uses the name resolution function of INS, and adds a proxy layer as an intermediate between client and INS servers, which could support dynamic names. Since Solar use INS, the names of Solar are also *descriptive* names.

### *5.1.4 Human-understandability*

This evaluation criteria is to evaluate if the name syntax is organized well with human habit, and the semantic of the name is easily understandable by common people, and easy to be remembered and transcribed. Human-understandability is not only decided by the alphabet used in the naming scheme, but also the syntax/semantics of the names. To determine whether a specific scheme is human-understandable or not is a subjective measure and it can not be quantified easily. But the designer of some naming schemes takes this problem into consideration and tries to make name formats or structures, easy to understand. For these schemes, we deem it to be human-understandable.

**Grapevine—** The two-level hierarchical RName *"F.R"* of Grapevine somehow reflects the organizational, geographic or administrative partition of the naming system. The name is easy to understand.

**Clearinghouse—** The three-level hierarchical name of Clearinghouse reflects the structure of the namespace and the organization of the system. The hierarchical name also could be mapped into a set of descriptive properties in a form that is easy to understand.

**V-System—** The name in V-System architecture consists of *global*, *administrational*, and *managerial* levels. Name is coupled with the managers of the named objects. The tree-like name of V-System is easily understandable by common people.

**URI—** Generally speaking, URI is not required to be readable by human although it is required to be transcribeable by human, i.e., it should allow users to copy the URI on paper or by computer. For URL, the host part denoted by a DNS name is generally a partitioned and hierarchical name which is always easy to read and understand although there are some false assumptions on DNS [61]. The other parts of URL are normally made up of human-understandable ASCII codes. But according to the URL syntax specification [9], non –ASCII codes are also permitted to be included in URLs. For URN, the NID (Namespace Identifier) is predefined using ASCII codes, which is human-friendly. However, the NSS (Namespace Specific String)

part is not necessarily denoted by ASCII codes.
**XRL—** For the name of XRL, as shown in figure 5(a), before the name is resolved by the "finder", it is like URL and readable. In the resolved form of XRL, as figure 5(b) shown, the "protocol parameters" are resolved into a not very human-friendly name of IP address and TCP port number. However, the resolved form is to be processed by a computer application and it will be transparent to users.
**UUID—** The 128-bit flat identifier of UUID is not readable by common people, even if it is represented in hexadecimal strings
**INS—** The descriptive name with "attribute-value pair" used in INS fits quite well with the human habit of describing certain things with certain properties. So generally speaking, INS name is human-friendly.
**SOLAR—**Solar is also human-friendly. It also uses descriptive "attribute-value pair" to denote a name, just like INS.

### *5.1.5 Name Extensibility*

Name extensibility is to evaluate if the structure of components of a name is fixed or extensible - that could support future upgrade. Generally speaking, flat names are mostly fixed The extensibility of partitioned names depends on the design. Descriptive names, however, are most extensible..

**Grapevine—** The two-hierarchy names of Grapevine are fixed. So Grapevine is not extensible.
**Clearinghouse—** The Clearinghouse names have three-hierarchies, so it is also not extensible.
**V-System—** Names used in V-System are also a three-level naming architecture which is formed by *global*, *administrational*, and *managerial* level. Because of the usage of "%" to designate the root and the "/" character to separate the name components, the names could be extended if new low level servers join and new objects to be named are added.
**URI—** For URL, the hierarchical DNS structure is extensible with the expansion of requirement of certain sub-domain. The namespace of the URL is also extensible by adding new schemes representing new type of resource. For example, the most popular schemes used in URL are *http, ftp, telnet etc*. New schemes could be added into this list. For URN, one of the design goals of URN is to make it easy to map other namespaces into URN-space, so it is ready to be extended.
**XRL—** XRL is designed for Inter-Process Communication for XORP, and all the components in the XRL name are predefined and fixed. Moreover, the structure of the components in the name is also fixed and difficult to be extended.
**UUID—** UUID name is 128-bits long and does not support longer bits to cater future expansion needs for larger namespace.
If the length is to be enlarged, the UUID generating algorithm is also required to adjust somehow, especially, the old 128-bits UUID should be compatible with the current enlarged one.
**INS—** The descriptive "attribute-value pair" does not put any restrictions on the number of types of attributes, the level of the name-specifier tree. Intuitively, the name-specifier tree could be expanded as the user wants. For INS which is originally intended to name services, the expandable descriptive name is very suitable.
**SOLAR—** Similar with INS, name in Solar consists of a series of "attribute-value pair", which is extensible according to the user's requirement.

### *5.1.6 Name Assigning and Organization*

This evaluation criteria of naming scheme refers to the way of assigning, managing and organizing the names. The name could be assigned and organized by a *centralized*, *hierarchical*, and a *distributed* way depending on the design and implementation of the naming scheme. *Centralized* name organization is easy to be administrated and collision of names could be avoided effectively but the central server could be easily overloaded and become a single point of failure pointing the system with the expansion of the system in large-scale Internet. *Hierarchical* domains split namespace into independent tree-like sub-domains which avoid namespace collision efficiently. Fully *distributed* name organization also benefit from the namespace split. Moreover, these separated namespaces are completely independent and managed in a fully distributed fashion, unlike the system of hierarchical name organization.

**Grapevine—** The Grapevine name is assigned in a distributed fashion. More specifically, the name system is organized into a two-level hierarchy, in which all clients named “F.R” are served by the registration server R.
**Clearinghouse—** The name organization of Clearinghouse is also a three-level hierarchy with local clients, domain servers, and organization servers.
**V-System—**Names of V-System consist of a three-level hierarchy. But, the name of V-System is tightly coupled with the object manager which is in charge of the object.
**URI—** Currently, the domain name part of the URL is organized as a multi-hierarchy scheme. The name is assigned and managed *hierarchically* by the sub-domains. URN incorporates different namespaces by the NID (Namespace Identifier). The URN name could also be assigned hierarchically, and each domain with specific NID could choose its own namespace and name structure.
**XRL—** The names in XRL are predefined and resolved by the process “finder”. Since XRL is designed to mediate the IPC within XORP, the names in XRL are not assigned and managed with a structure like other distributed systems. Generally, the name assigning and organization could be considered *centralized*, since all the module names, interface names, interface version and method names that form a XRL name are handled by a centralized process named “finder”
**UUID—** UUID is assigned a *fully distributed* structure wherein a distributed name assignment algorithm is in charge of generating unique names without introducing any infrastructure.
**INS—** Because INS uses descriptive names, it allows the users or services to select their own name according to their attributes. Users or servers could give any values to these attributes. So the descriptive name must be generated in a *fully-distributed fashion*. However, because of this, it is difficult to ensure the uniqueness of the names generated by fully distributed objects.
**SOLAR—** Just like INS, Solar names could be generated *distributed* by users and services according to their attributes and values.

### *5.1.7 System Scalability*

Different from *name extensibility*, system scalability is the criterion to evaluate the scalability of the naming system. While name extensibility is concerned with the syntactic structure of the components of the name, *system scalability is* concerned with the structure of the servers of the naming system, whether it could be expanded easily to support larger scale networks or larger scale applications.
**Grapevine—** Grapevine is designed specifically for electronic mail delivery service. Its two level hierarchical name structure also reflect its two level network structure: the clients and the Grapevine servers. The server level includes registration server and message server. Grapevine does not support other applications, and cannot scale to fit for large-scale networks because of the structure of the registries and the replication strategy of the registries.
**Clearinghouse—** The structures of the Clearinghouse servers have 2 levels – domain server level and organizational server level. The structure is larger than Grapevine and could support larger scale applications. However, its fixed three-level organization makes it not scalable, just like Grapevine.

**V-System—** The tree-like structure of V-System makes it scalable when new global, administrational or managerial namespace added.

**URI—** The DNS has a multiple hierarchical structure. When a new namespace is added and new management domain entities are introduced, the DNS structure could be scalable to support the new features and new domain entities.

**XRL—** XRL is used for Inter-Process Communication, which includes no architecture for name resolution, and all the resolution functions are performed by the central process "finder". So XRL is not scalable.

**UUID—** UUID uses flat names and has no resolution architecture, so it does not support architectural scalability.

**INS—** In INS, the name-tree is replicated in each INR so that the name resolution could be performed locally and efficiently. However, to keep the name-trees among these INR consistent will cost a lot of traffic and band width, especially for large-scale dynamic networks. Because of this, INS does not scale well in large-scale networks and large-scale applications. An improvement measure named INS/Twine has been proposed as an extension to INS in 2002 by MIT Laboratory. Name-specifier in INS has been split into strands, generating a numeric key for the strands, and use protocol Chord to distribute the keys among INRs.

**SOLAR—** Solar uses a proxy between a client and an INR to resolve a dynamic name into a static name and then transmit the static name to INS network to perform further resolution. When mobile entities move between proxies and the dynamic name changes in the proxy, all the proxies that the name was previously registered with will be notified via intentional multicast. The multicast cost considerable network bandwidth. Moreover, because Solar use INS as its core resolution network, the INR is also replicated. The proxy servers also need a lot of replication if services move between proxy servers too often. These factors limit the scalability of SOLAR.

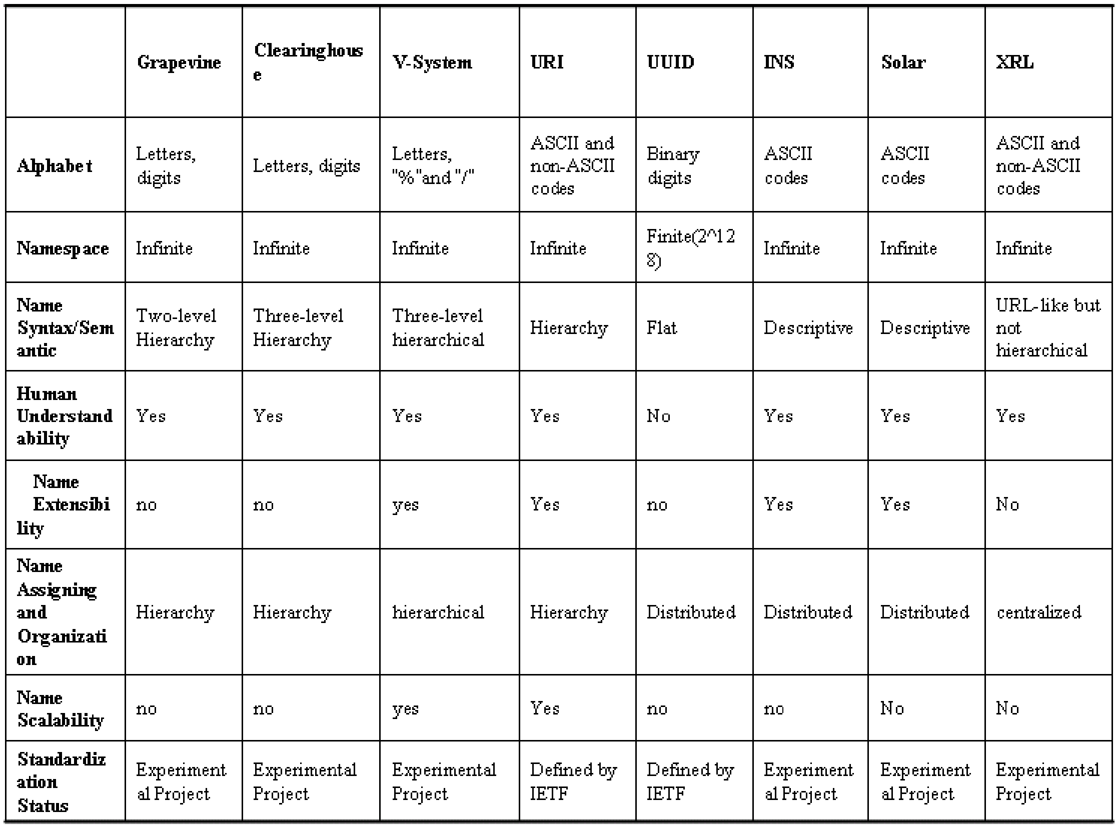
### *5.1.8 Standardization*

For all the eight schemes we discussed above. The UUID is standardized by IETF in RFC 4122 [40], and the URI is standardized by IETF in a series of RFCs [8][9][19][45]. All the other schemes are experimental project by corporations, universities and research institutes.

## *5.2 Comparison*

After the detailed analysis and comparison of these schemes for management plane function, we give a summary of the evaluation for every scheme in table 3. The horizontal row of the table includes all the schemes we discussed above, the vertical column are the evaluation criteria we set for them.

Table 3: Comparison of the schemes for management plane



| | Grapevine | Clearinghouse | V-System | URI | UUID | INS | Solar | XRL |
|---|---|---|---|---|---|---|---|---|
| Alphabet | Letters, digits | Letters, digits | Letters, "%"and "/" | ASCII and non-ASCII codes | Binary digits | ASCII codes | ASCII codes | ASCII and non-ASCII codes |
| Namespace | Infinite | Infinite | Infinite | Infinite | Finite(2^128) | Infinite | Infinite | Infinite |
| Name Syntax/Semantic | Two-level Hierarchy | Three-level Hierarchy | Three-level hierarchical | Hierarchy | Flat | Descriptive | Descriptive | URL-like but not hierarchical |
| Human Understandability | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes |
| Name Extensibility | no | no | yes | Yes | no | Yes | Yes | No |
| Name Assigning and Organization | Hierarchy | Hierarchy | hierarchical | Hierarchy | Distributed | Distributed | Distributed | centralized |
| Name Scalability | no | no | yes | Yes | no | no | No | No |
| Standardization Status | Experimental Project | Experimental Project | Experimental Project | Defined by IETF | Defined by IETF | Experimental Project | Experimental Project | Experimental Project |

As a summary of this section, we give eight important evaluation criteria to appraise the eight *management plane* naming schemes in great detail. By comparing them carefully, we get a better understanding of the function related with *management plane*, which could also be useful and meaningful as a guide for our design of future naming system for Next Generation Internet, specifically, Internet 3.0 [30]. In the following section, we will further discuss the transport plane naming schemes and also address a set of different evaluation criteria for them.

## 6. Introduction And Analysis of Transport Plane Schemes:

Over the last decade, there have been quite a few significant changes in the networking scenario relative to the era when the current internet was designed. The current internet was designed as an internetworking platform between friendly academic and government institutions with limited number of users and fixed workstations. It was centrally controlled, first by DARPA and then by NSF. The needs from the network were quite humble and involved mostly academic exchanges. Eventually, the internet was opened for public use. Central control was relieved. The internet was going through a phase of commercialization and self sustenance. The popularity of this new found media of communication was fast increasing. The internet was growing and growing faster than its founding fathers could ever imagine. Soon the 32 bit IP address space, which was deemed more than enough, seemed too small. The environment was no longer friendly academicians talking research. It was an environment which provided access to anybody and everybody who could connect to it. Academic exchanges were supplemented with commercial transactions. Another significant change that came about at the same time was the advent of the era of wireless networks. The last mile of access networks started supporting wireless connectivity. Fixed work stations evolved into mobile end-hosts.

The Internet design had to evolve to incorporate the changes in the networking environment, but it was way too huge already to bring about any significant change cheaply. Hence, researchers looked for ad-hoc solutions to immediate problems. But the disparate solutions were neither holistic nor complete. They verged upon disobeying the initial design principles of the internet. One such scenario was the introduction of the Network Address Translation (NAT) scheme to subvert the rate of depleting IP addresses. NAT successfully curbed the rate of depleting IP addresses but broke the end-to-end semantic design principle of the internet. It introduced heterogeneous private address contexts within the internet causing problems in IP addressing and routing, since IP was designed for a homogeneous network scenario.

Some of the design invariants of the internet which provided consistency in the past, seemed to be constraints to free evolution in the present. Significant among them was the semantic overloading of IP addresses as locaters and identifiers [63]. The semantic overloading of IP addresses was a major hindrance to solutions for mobility. IP addresses were identifiers to which transport protocols were bound and also locaters, acting as indexes in routing tables. Hence Location – Identity split was an immediate requirement of the present internet.

A paradigm of peer to peer exchanges evolved out of the traditional client server model. Peer to peer system design required scalable methods for content addressing, discovery and retrieval in the absence of well known global servers. This saw the rise of Distributed Hash Tables ( DHT) for scalably managing addressing and routing of flat address spaces.

In our study of the Transport Plane of naming, we will address the various architectural frameworks which were proposed to deal with the problems of :

1. Naming and addressing across heterogeneous address contexts
2. Identity/ Location Separation
3. Flat Address Space Architectures

*6.1 Naming and addressing across heterogeneous address contexts*

One of the design principles of the internet was to ensure end to end semantics with every host being equal in terms of naming and reachability. The initial IP design ensured the said principles by assigning globally unique address to each host on the internet. The TCP/IP based internet became popular, and supposedly a bit more popular than it was ready to handle. The unique address space in IP was 32 bits long and embedded with hierarchy to handle the scalability issues of the routers, routing these addresses. The embedded hierarchy helped the cause of scalability but led to the inefficient use and thus wastage of a large part of the address space. On the other hand, the number of hosts connected to the internet was rapidly growing. Soon, it was realized that the depletion of the 32 bit IP space was inevitable [44]. Now the problem was that the internet was already too huge to be able to accommodate any major changes to the way it functioned. The proposal of using IPv6 , a 128 bit address space, was researched and standardized with much effort but unfortunately IPv6 ran into interoperability issues with the present IPv4 addressing and naming and hence was never deployed on a wide scale. An alternative solution was presented at about the same time as IPv6 in the form of NAT- Network Address Translation .NAT was based on the idea of having private address networks connected to the public internet through a relatively few global addresses, the mapping between private and public addresses handled by an intermediary NAT box. The NAT box generally used port numbers to maintain a one-many relation between public – private addresses. NAT was an instant hit and became profusely popular. The reason for the popularity of NAT can be attributed to the relative ease of its deployment and to the obvious commercial and technical benefits to the providers. But then NAT was not without its own share of perils. It successfully curbed the rate of IPv4 address depletion but at the same time it destroyed the end-to-end semantics of the original internet design. It affected any architecture or scheme which was based on an end-to-end assumption. It introduced regions of heterogeneity to the homogeneous internet cloud. Private network hosts could easily reach out to the public network but the vice versa was not true. Fortunately, most web services are based on the client – server architecture and most servers are placed in the public networks. Classical web applications thus remain unaffected.

Naming, addressing and routing across heterogeneous contexts has always been an interesting issue that has been much researched. But the widespread popularity of TCP/IP networks and thus a kind of homogenization of the Internet has to some extent curbed such research endeavors. The NAT problem poses an almost similar situation with the public and private address regions being the separate contexts. We deemed it important to address the situation as it is immediately relevant to the present form of the internet. In this section we shall be discussing IPNL [52], 4+4 [76], Plutarch [17] and Triad [15] in an attempt to discuss the various solutions proposed to the NAT problem. Instead of going into the exact details of each scheme, we shall try to have an objective look at each scheme pertaining to a near common situation. Our aim is to project the different flavors of the various solutions.

### *6.1.1 Approach 1: 4+4*

The 4+4 approach is supposedly the simplest approach towards re-installing the end-to-end semantics across private network contexts.4+4 proposes the concatenation of a private IP of the end-host and the public IP of the NAT router to refer each host uniquely in the private network context.

4+4 is basically an encapsulation mechanism where each router is presented with the address that it can interpret. Routers in the global IP domain are presented with the public IP of the NAT box, while the NAT box routers are presented with the private IP of the host that it administers.

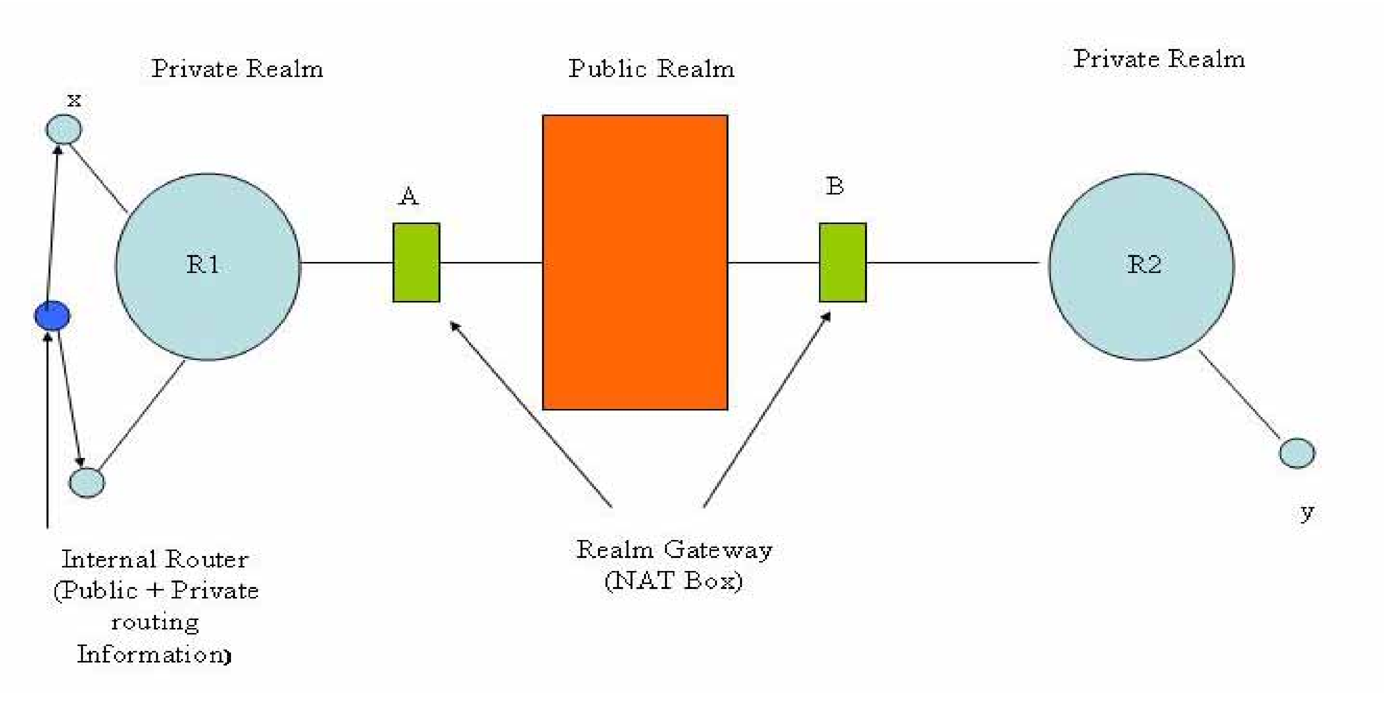


Figure 8: Approach 1: 4+4

As in Figure 8 , The simplest network topology representing today's internet can be viewed as a huge number of private networks (called private realms in 4+4) connected to a public realm through NAT boxes (Realm gateways in 4+4).Each realm may have internal realm routers which are for routing intra-realm packets. Packets to other private realms or the public realm are routed through the realm gateways.

The main issues in designing such a system are:

- Initial Address Discovery
- Routing : Presenting each routing entity with addresses which lie within their paradigm
- Deployment in the present internet

4+4 suggests an extension of the DNS naming system to provide unique names to each host in the internet. The DNS stores 2 names per end host – a level 1 name, mapping to the IP of the hosts realm gateway and a level2 name, mapping to the end host IP. Thus initial address discovery involves DNS lookup based on the FQDN of the host, twice.

The main design rationale behind routing in 4+4 is to present each intermediate router with the address that it understands. As shown in Figure 8, the endeavor is to send packet from X to Y, assuming both X and Y are 4+4 aware. First, X needs to query the DNS for Y. The DNS reply is then checked to see whether the level1 address part of Y matches any of its own level 1 address .If a match is found, both these are in the same address realm and hence X just sends an IPv4 packet toward Y through the internal routers. If the level 1 addresses do not match, then the destination address is selected as B.Y and the source address is A.X. The source node node creates a 4+4 header with X and B in the outer source/destination fields and A and Y in the inner source/destination fields thus the routing is from X → B. The packet is sent. The realm router of A intercepts this packet and swaps X with A as source address. The Public network thus routes the packet from A→B. At B, the realm router of R2, The destination addresses are swapped to Y in the outer field and Br in the inner field , thus routing is directed by A→ Y. Thus a simple trick of address swapping between the inner and outer fields helps to maintain the compatibility in routing with the existing networks.

The next issue deals with deployment in the current internet. 4+4 retains backward compatibility with existing networks by presenting itself as an IP-in-IP mechanism to 4+4 incompatible hosts while introducing a new 4+4 header for 4+4 compliant hosts.

So to summarize, 4+4 provides a IP tunneling mechanism to get past the problem of addressing and routing to private address contexts, but since its routing mechanism involves swapping of source and destination addresses enroute, it is bound to add an overhead for router processing. Also end to end security mechanisms may suffer as a result of address swapping. Also 4+4 does not explicitly discuss the overheads of providing a FQDN to each host in the internet and also having to support 2 entries per hostname.

### *6.1.2 Approach 2: IPNL*

Internet Protocol Next Layer proposes a new layer above the IP layer to address naming and routing across private and public contexts. IPNL and 4+4 are essentially similar in the way they make private hosts addressable but IPNL is a bit more complicated architecture when compared with the simplicity of 4+4. However, we believe, IPNL helps to capture the essence of the problem in greater detail.

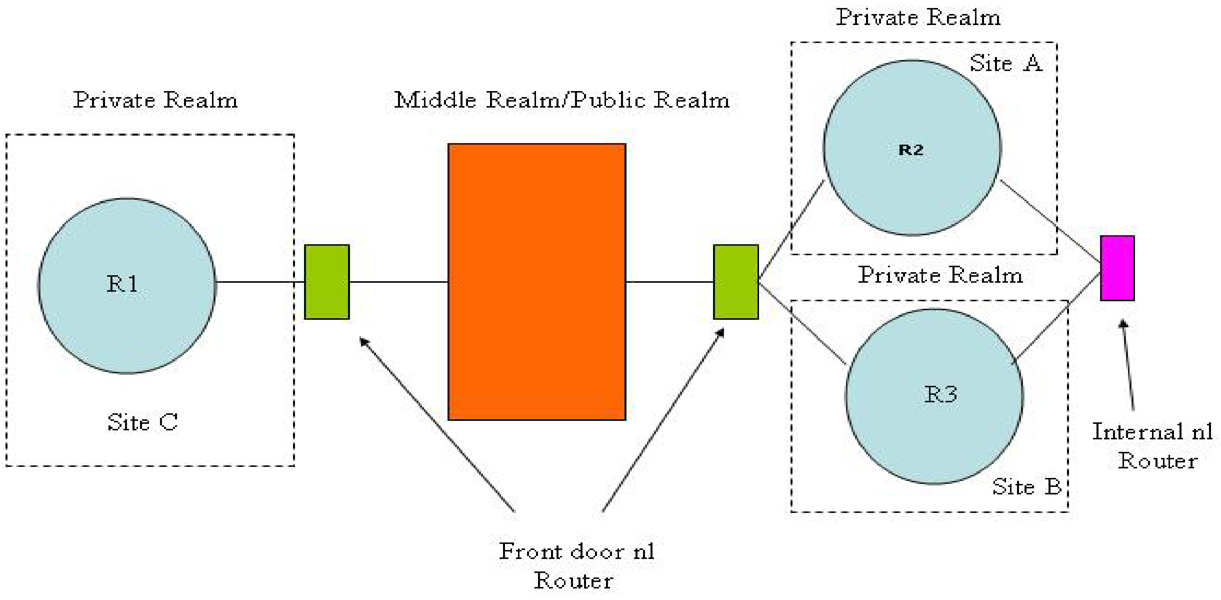


Figure 9: Approach 2: IPNL

As in Figure 9, IPNL considers the Internet topology to be largely comprised of Public or Middle Realms and Private Realms. Private realms correspond to the private address context while the public or middle realm corresponds to the public internet. The private realms are free to choose their own mechanisms for naming and routing for all intra-realm packets. An internal router called the internal nl-router connects two private realms and a frontdoor nl-router connects a private realm to the public realm. A frontdoor nl-router may have more than one private realm behind it, supported on separate interfaces. The various sites in the figure merely emphasize the administrative domains. This topology is based on numbering with IP addresses. Another view of the same topology is in terms of DNS names, wherein each realm is a DNS zone. Every realm may have more than one zone, but each zone is supposed to service just one realm.

The key idea is to introduce two separate address spaces i) Fully Qualified Domain name (FQDN) for each internet host which is kind of static identity for the hosts and ii) IPNL addresses which is more dynamic in nature formed by the concatenation of [MRIP →Middle Realm Global IP ] + [ RN→ 2 Byte Realm number ] + [ EHIP → End Host IP ]. The FQDN name of the host is supposed to be of variable length but static and unique for each internet host. Hence it is used for initial lookup and path discovery. The path discovery mechanism involves learning of the various unknown pieces of the IPNL address such as the source and destination MRIP, source and destination RN and the destination EHIP. After path discovery, subsequent packets are supposed to carry just the IPNL addresses which are of fixed size and hence a better choice for protocol header parameter.

Now, we consider the various concerns in this IPNL proposal:

- Each host is to acquire a FQDN, requiring scale up of the DNS model.
- An FQDN based routing mechanism has to be implemented, hence DNS has added functionality of routing.
- Separate Policies needs to be fixed for inter-site and intra-site packet forwarding.

- Site isolation
    1. Isolation with regards to numbering independent of the frontdoor nl-router.
    2. Isolation with regards to strict non-usability of EHIP outside a realm context

Each of these issues have been dealt with in the IPNL proposal. The scale up of the DNS model is suggested to be implemented through a distributed mechanism wherein every nl-router is supposed to know about every host in each realm it is connected to. This assumes that realms are not very big entities containing only a few end hosts.

Routing using FQDN requires that forwarding entries based on FQDN be maintained by routers. This is achieved by having the middle realm DNS maintain ‘A’ records for mapping DNS names to front door nl-routers and using the public IP routing mechanism to route to the front door nl – router of the host. Local DNS information is used for routing packets from the front door nl-router to the proper realm and thence to the proper host.

Separate local and global headers ensure that there is a fixed policy for inter-site and intra- site packet forwarding. Locally addressed packets need not have any MRIP since they need not be globally unique. The knowledge that a packet is for a local host is learnt during the initial stage through FQDN based routing.

Site isolation is an important consideration since it has a lot of administrative advantages. Site isolation eases the administrative task of renumbering. The idea is that, the more loosely coupled addresses are to their logical hierarchies, the less disruption is caused during site renumbering. As an example, if all hosts within a realm take addresses from a block assigned to the front door nl-router, each time the nl-router needs to change its association with its higher levels, each host behind the router needs to be renumbered. IPNL avoids such a situation by assigning independent addressing of hosts relative to their nl-routers. Also, since each realm is uniquely identified by their DNS zones, it is possible to have a separate interpretation of Realm numbers between the hosts and the front door nl-routers. This helps in decentralizing the activity of providing realm numbers when there are more than one realm behind the same front door.

To summarize, IPNL introduces a new layer between IP and TCP, taking the responsibility of end-to end routing on itself while reducing IP to the task of a point to point routing mechanism similar to the link layer.

### *6.1.3 Approach 3: TRIAD*

TRIAD seems to be a combination of the ideas of 4+4 and IPNL. The topology as shown in figure 10, is similar to the earlier schemes. TRIAD too advocates use of DNS names as global identifiers to uniquely identify each end host from any context. In TRIAD, intra realm routing is furnished conventional IP routers using present routing methodologies while inter-realm routing is name based as in the initial stage of IPNL and done between realm routers.

The relay agent of a realm is responsible for maintaining a directory service for name lookups within its own realm and also determining the next hop for names referring to other realms. For initial route discovery TRIAD uses its Directory service to forward the initial packet towards the destination , with each relay agent in the path performing a lookup within its directory to select the egress port for the packet and appending the same in the forward token header field of the Wide Area Relay-Addressing Protocol (WRAP) header. WRAP is an IP Encapsulation Protocol for the relaying of packets across various RA’s. WRAP headers carry 2 Internet Relay Tokens(IRT), the forward token and the reverse token, which in the simplest scenario is a list of IP addresses of the intermediate path RA’s egress ports. Such a listing of IP addresses in the IRT is called transparent relaying. In opaque relaying however, the intermediate RA’s write a random number in the IRT’s and maintain state for mapping the random number into egress port addresses. The end-to-end semantics is maintained by having a pseudo transport layer (TRIAD-TCP) which is a minor extension of TCP and is setup as a negotiated option between source and destination be bound to names rather than addresses at both ends.

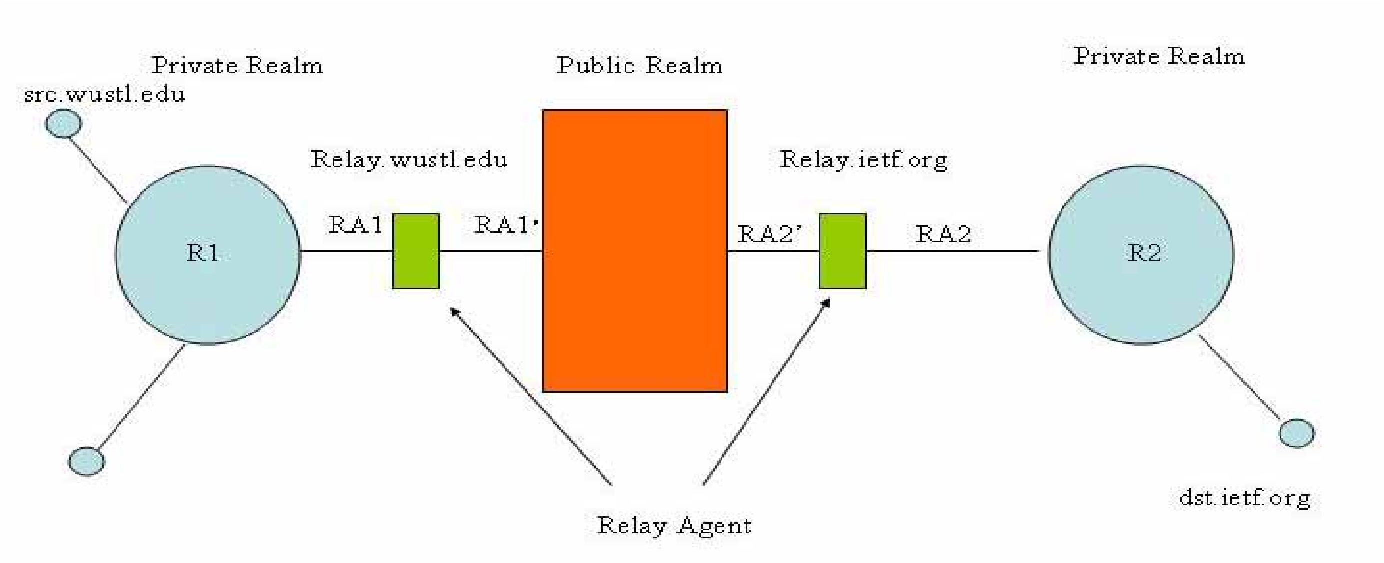


Figure 10: Approach 3: TRIAD

TRIAD defines a network topology similar to the current internet topology with NAT based private networks interconnected using BGP based routers. TRIAD calls these private networks as realms and the BGP routers are called Relay Agents. A source ‘s’ uniquely known by its DNS name, suppose s.wustl.edu wanting to communicate with the destination , suppose d.ietf.org, sends out a name lookup request to its RA. The TRIAD Directory service, integrated into each RA, provides lookups for the next hop determination toward the destination. The RA’s use this information to route the initial packet from ‘relay.wustl.edu’ to ‘relay.ietf.org’ and appending to the forward IRT field in the WRAP header. On receiving the query packet , ‘relay.ietf.org’ sets up a local mapping state between its public IP interface and the destination host interface and returns the packet to the source appending an identifier “f’” representing the local state to the forward IRT. “relay.wustl.edu” then returns an identifier “f “ to the source and creates a mapping state for f→f’. After this initial communication, IP based tunneling is used with inner WRAP header and outer IP Header for delivering packets to the destination. The Outer IP header is used for Hop-by-Hop routing of packets between RA’s while end-to-end routing is maintained by the WRAP header. At each Hop, the RA, removes an entry from the forward token, appends an entry to the reverse token and places its egress port IP address to the source field and the destination RA’ IP in the destination field in the outer IP header. The reverse token is initialized with an identifier “r’ “ , representing the mapping state between the public IP of “ relay.wustl.edu” and the source. At reaching “ relay.ietf.org” the forward token only has “ f’ “ and the reverse token has the full reverse path in it”. “relay.ietf.org” replaces “r’ “ in the reverse token with a local state identifier “ r “ and forwards the packet to f’. When the destination need to communicate with the source, it sends the packet with forward IP = r and destination IP = “IP of relay.ietf.org”. the relay router “relay.ietf.org” is now responsible for filling in the forward token of the packet with the “reverse token” of the received packet from the source. To summarize, TRIAD has a similar architecture as 4+4 and IPNL, however TRIAD provides details of its directory service integrated to its relay agents. The directory service does a DNS like name → address mapping for names within the realm and for lookups for inter realm names it also maintains routing information.

### *6.1.4 Approach 4: Plutarch*

Our final approach, Plutarch, differs from the rest of the schemes in that it attempts to solve the generalized problem of naming and addressing across varied contexts, of which the NAT problem is just an instance. Plutarch, basically proposes some basic principles over which naming and addressing heterogeneity can be made explicit and made to co-exist with other heterogeneous contexts. Two congruent contexts i.e., two GPRS networks should be able to communicate across a public IP network even though the public IP network is completely opaque to the semantics of the GPRS network. This is the network Plurality that Plutarch advocates.

As in figure 11, Context A (GPRS Network) and Context B (Sensor Network) might want to communicate through Context C(Public Internet) which is completely heterogeneous to A and B. However at the border between two contexts, there are interstitial functions which are responsible for resolving the differences between the two contexts and allow them to communicate.

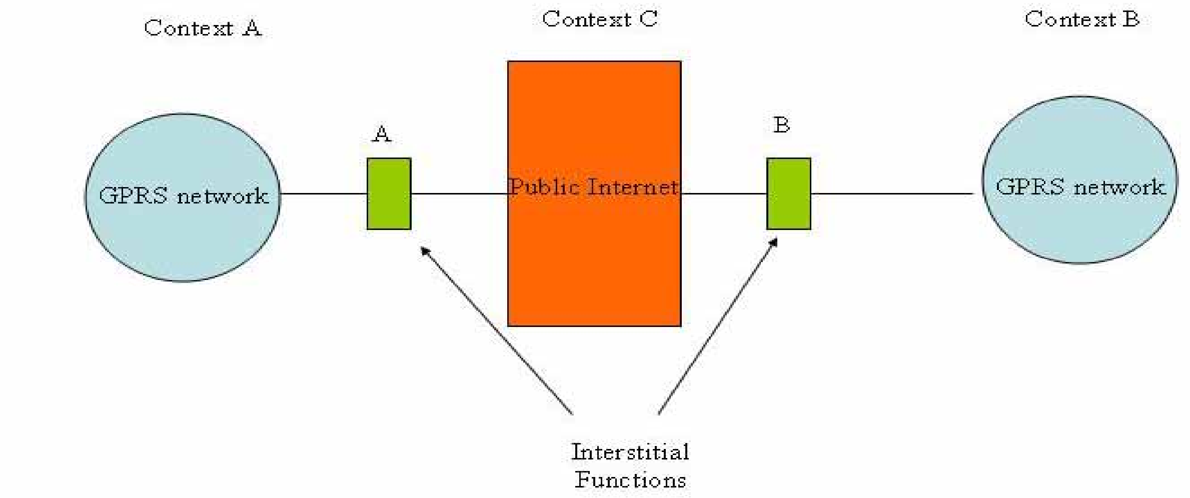


Figure 11: Approach 4: Plutarch

Within each context, the space is completely homogeneous with respect to addressing, packet format, transport protocol and naming services. In Plutarch there is no concept of global names. Instead names are unique within a context. Each end host is globally known through its context and its address within that context. This idea is quite similar to the notion of public and private addresses in the NAT based architecture discussed earlier. The route discovery mechanism, employs broadcast gossip type advertisements and returns a chain of plausible contexts between the source and destination. This scheme is very similar to a chain of IP address between two relay agents in the forwarding token of TRIAD.

Plutarch, is thus a generalized framework with almost similar ideas as the other three schemes which pertain more to the specific problem of NAT based contexts.

Summary

IPNL, 4+4, TRIAD and Plutarch all address the same common concern of naming and routing across heterogeneous contexts in the internet. All 4 schemes provide 4 separate approaches for the same. None of the approaches seem to be markedly different from any other. In fact there is more common denomination among them than differences. All advocate within context homogeneity in addressing and routing. IPNL,4+4 and Triad propose DNS names to be the unique identifier of all hosts and propose some variant of name based routing. All the schemes rely on the information and methodologies used at the border routers to map the differences between the varied contexts. However, we feel that deployment of each of these schemes require substantial amount of changes either at the end-host protocol layers or/and the intermediate router. It is really difficult to put forward any authoritative comment as to the feasibility of such changes. The present internet can easily claim to be the best test bed for any scalability tests. Such is its vastness that even the most minor changes require mammoth effort, especially if it involves changes at the end hosts. In all the schemes that we have discussed here, none of these schemes tend to work without infusing some small changes at each layer of the internet topology. But then, for the internet to evolve, a solution needs to be found to homogenize the network and re install its end-to-end semantics. Not only for NAT introduced heterogeneity, but also for heterogeneity being introduced by newer and newer kinds/types of networks, we feel that this area of research is extremely relevant for the future evolution of the internet.

### *6.2 The Identity/Location Split*

In this section we will be presenting an objective study of some of the most quoted naming schemes proposed over the past few years which we think take a positive step towards solving the present weakness(semantic overloading of IP addresses) of the internet. We shall be evaluating these schemes by putting these schemes to test against a standard questionnaire, which shall help us better understand the relevance of each work towards solving the present problem. The questions that we shall be asking are as follows:

- Is a new Global Namespace being introduced ?
- What are the security considerations if any ?
- How is mobility handled?
- Does it depend on IP at some level? Which?
- How is Scalability resolved ?
- What are the various Name Bindings?

The name systems that shall feature in our study are FARA [16], Delegation Oriented Architecture [70][71], Layered Naming Architecture [4], I3 [68], Intentional Naming System [1], HIP [47][50][32][36] [48][64] and LISP [23]

In the next section we shall be presenting a brief overview of each of these naming systems and then in the subsequent sections we shall try to present an objective evaluation of each scheme under some common parameters.

We shall be presenting a general summarized discussion about each naming scheme which shall better prepare us to undertake an objective evaluation in the later sections.

### *6.2.1 FARA*

FARA, Forwarding directive, Association and Rendezvous Architecture presents an abstract meta architecture of a naming scheme based upon the decoupling of end system identity from IP addresses thus resolving the problem of IP overloading. It is a top down approach with the aim to look objectively and broadly at the best design possible and then culminate at a more concrete and practical implementation of the same. The basic components of FARA are i) entities – generalization of the end point of a communication and the smallest unit that can be mobile, ii) Association – persistent communication state within communicating entities iii) Communication Substrate – a generalized connectionless packet delivery network communication subsystem responsible for addressing and routing packets for FARA associations using a header having a Forwarding directive [FD] to reach the destination

#### *6.2.1.1 FARA Architecture*

Two entities A and B in FARA wishing to communicate shall have to setup an association state between them, referenced by an Association Id(Aid), by means of some kind of handshaking. Aid's are unique within the communicating entities. Initially however A needs to have the Forward Directive to reach B which can be obtained from the FARA directory service (fDS) . fDS is a high level discovery mechanism which may resemble DNS like lookup mechanism. Since Entity names in FARA need not be globally unique, the directory lookup may map to more than one (FD,RI string pair) and a rendezvous process might have to do the final disambiguation. The RI string is a Rendezvous Information string and used during the initial part of the communication for setting up a unique AId between the two parties. The sender may generate the complete destination FD from the RI string or it may forward the packet to a proxy which may rewrite the FD, RI string pair and forwards to the destination. After the initial handshaking is over and an AId has been setup, the client and server communicate normally with the Forwarding Directive routing packets and the AId delivering the packets to the correct endpoint entity.

#### 6.2.1.1 *Objective Summary of FARA*

**Namespaces**
The various names/Identities involved in the FARA architecture are i) Aid, ii) Entity name, iii) End System address. Among these the Aid's are supposed to be unique within the entity and has no global namespace. An Entity too does not require to have a global namespace. To communicate with a specific entity it must be possible to determine its location by some method such as user friendly character string lookup etc. FARA does not require end system addresses to be picked up from a single global address space. In an instantiation of FARA called MFARA, an idea of an "addressing realm" is introduced. Addresses within a certain realm are unique.

**Security**
FARA decouples FD's from Associations, thus removing the limited assurance provided by overloaded IP addresses which gives some assurance to the source of the packet. FARA leaves it to the implementer to decide upon the level of security and technique to be used for end to end source verification.

**Mobility**
FARA mandates the implementation of an FD management system in each instantiation, which is responsible for managing and updating FD's as entities change their logical point of attachment. During mobility, though

the FD may change the AId remains the same. In MFARA instantiation a new M-agent is introduced which is the static rendezvous point associated with each mobile entity  to keep track of the changed FD of the mobile Entity based on information from the Mobile Entity.

**IP dependence**

FARA proposes a new high level meta architecture. The IP may be used as a part in the FD of a certain FARA instantiation.

**Scalability**

The Scalability issues within the FARA meta Architecture depends completely on particular instantiations of the architecture.

**Name Bindings**

The fDS map the Entity names with a forwarding directive. The binding need not be an injective mapping and it depends on a rendezvous service to provide an unambiguous binding. During mobility the mapping between the entity and FD may change and may require a rebinding using some FD Management service. The binding between the Aid and the communication session never changes within the session.

### 6.2.2 *Delegation Oriented Architecture (DOA)*

DOA was proposed with the purpose of establishing middleboxes as first class citizens of the internet. Internet middleboxes such as NAT, firewalls etc are deemed harmful since they break two basic principles of the internet design, namely, i) Each communicating entity shall have an unique network layer identifier (IP) and ii) Entities shall not look at packets not addressed to them. Middleboxes such as NAT's do hide large private networks behind few gloabal IP addresses and firewalls often screen content of every packet going through them into their network. DOA allows hosts to explicitly authorize or delegate part of their jobs to middleboxes. It introduces a Location/Identity split which is extremely relevant to our undertaking.

#### *6.2.2.1 DOA Architecture*

DOA introduces a new name identifier called the EID (Entity Identifier) which provides an identity to communicating host, separate from its IP address which is reduced to being just a locater. An entity wishing to delegate some of its tasks may post an ERecord in the EID resolution infrastructure mapping its EID to the delegated hosts IP address. Thus any client A trying to send a packet to Server S with EID Si, first looks for the servers EID in the Infrastructure. The Infrastructure returns the client with an ERecord.corresponding to the EID Si. The client then looks up the IP address from the ERecord and sends the packet. The packet goes to the delegated hosts IP and after the delegated host has done the required processing of the packet sends it over to the server for which the packet was originally meant for. The ERecord may map an EID to a list of EID's, reflecting the servers wish to have the packet go through number of delegated hosts before it reaches the server.

#### *6.2.2.2 Objective Summary of DOA*

**Namespaces**

DOA introduces the concept of a separate global identity for each host called EID which separates the upper layers of the TCP/IP protocol from IP. EID's are non human readable cryptographic names in a flat global address space.

**Security**
EID's are the hash of a public key and every ERecord is signed with the corresponding private key. Hence ERecords are verifiable to the source and also only the owner can modify a ERecord. However source EID spoofing is still possible, but then upper layers can verify a source EID by getting the actual ERecord corresponding to the spoofed EID from the infrastructure and thus the server replies to the original IP address.

**Mobility**
DOA does not separately discuss mobility because it is not the purpose of DOA. However, with a location Identity split and with the EID resolution infrastructure in place, DOA could easily support mobility with added functionality of rapid mapping updates and a signaling mechanism.

**IP Dependence**
IP is used still used, but only as a locater and not as an end point identifier.

**Scalability**
The EID resolution infrastructure deals with flat names and hence providing scalability is a challenge. The scalability issue is resolved through the use of distributed hash tables.

**Name Bindings**
In DOA the name bindings are from DNS→EID →EID* →IP. Initially a DNS lookup is necessary to search for the EID of the server.Once the EID is known , The second level of binding takes place at the EID resolution Infrastructure and an EID can be bound to either one or more EID's or an IP address.

### *6.2.3 Internet Indirection Infrastructure (I3)*

The basic motivation of I3 was to provide anycast, multicast and mobility over the existing IP infrastructure by adding an overlay based indirection infrastructure providing rendezvous based communication abstraction. I3 associates with each host and Identifier (id) and injects an id→addr (IP+port) mapping into the overlay infrastructure called the "trigger".It is one of the first attempts towads Identity/Location split.

#### *6.2.3.1 I3 Architecture*

The basis of I3 is the id →addr mapping in the overlay infrastructure. Any host wishing to communicate with another host sends (data,id) to the infrastructure which routes the packet based on the id→addr mapping that it has.This simple architecture has the potential of supporting wide variety of services such as mobility, multicast , anycast etc.By changing the trigger mapping any mobile host can update the infrastructure of it present location. By allowing multiple hosts to register to a single id, I3 supports multicast. A group of hosts can form an anycast froup by sharing a common prefix.Any host trying to get to a particular member of the anycast group is routed using the longest prefix match.

#### *6.2.3.2 Objective Summary of I3*

**Namespace**
I3 introduces a new globally unique flat namespace for the trigger identifiers.

**Security**
I3 proposes the use of public and private triggers as a means to ensure security for every ongoing communication. The public triggers are those that are well known and published in public. These are used for an initial communication with a host. Both the hosts may setup a private trigger for a particular session after the initial communication through the public trigger, thus protecting against eavesdropping in case a malicious host inserts a trigger with the hosts id and its own address. Another form of attack can be DoS attack on a host by inserting a hierarchy of triggers and all triggers at the last point pointing towards the host, or on the infrastructure by setting up trigger loops by connecting trigger leaves of a trigger hierarchy to the root trigger of thet hirarchy. These attacks are avoided by the infrastructure checking for loops and also verifying the host setting up a trigger by sending a challenge to the address setup against the trigger, thus verifying whether the trigger was setup by its rightful owner.

**Mobility**
I3 supports mobility by allowing hosts who change their point of attachment to change their id mapping in the infrastructure.

**IP dependence**
IP acts only as the locater. The infrastructure uses IP to route the packets once it has mapped the id to the corresponding {IP, port} pair.

**Scalability**
The trigger database is stored as a Distributed Hash Table thus providing scalability to a flat namespace.

**Name Bindings**
In I3, the Infrastructure binds trigger identifiers to IP addresses. Thus, transport endpoints know only the Id's which are routed using the corresponding IP by the indirection infrastructure.

### *6.2.4 Layered Naming Architecture (LNA)*

DNS and IP addresses all name hosts in the internet. All Services and data are named relative to the host in which they reside. LNA is an attempt to give data and services an independent existence in the internet by decoupling them from their host names. Accordingly LNA introduces two new name spaces namely SID (Service Identifiers) – providing Service Name, and EID (End point Identifiers) –Transport layer endpoint. It also introduces the concept of ULD (User Level Descriptors) to aid in searching for Services according to User needs.

#### *6.2.4.1 LNA Architecture*

In LNA, name resolution has to take place at various layers of the TCP/IP stack. Initially, Applications map ULD's to SID using a lookup or search service. The applications session layer protocol is then responsible for mapping the SID to EID using a SID resolution service. Finally Transport layer protocol binds the EID to an IP address for routing the packets to the specified host. An application "A" running on host "H", gets to know the SID through some search mechanism using ULD. Now to access Service "S", A hands S to the service resolution layer. A  Distributed Hash Table based Service Resolution Layer, resolves the SID to one or more (EID, transport, port) triplet. A then seeks the services at the specified EID, using the specified transport protocol and port number. The transport protocols are now assumed to be bound to EID's rather than IP's. The transport protocol passes down the packet to the EID resolution layer which resolves the EID to one or more IP's. The EID's are carried in every packet to aid for any delegations on the path.

#### *6.2.4.2 Objective Summary of LNA*

**Namespace**

LNA introduces two new global namespaces, Service Identifier (SID) and Endpoint Identifier (EID)Both SID and EID are flat namespaces and globally unique.

**Security**

The issue of Security is not discussed explicitly but LNA does acknowledge borrowing ideas from HIP which leads us to think that the identifiers could probably be Cryptographic and provide some sort of authentication security.

**Mobility**

The whole spirit of LNA is to support mobility of data and services. Whenever a host is mobile, it may just change the EID→ IP mapping in the EID mapping infrastructure. Also, during communication, If the IP of a host changes then the transport layer of the peer layer can re-initiate a lookup and rebind.

**IP dependence**

IP is used as the locater and binds only to the IP layer. The transport protocol binds to EID and delivers packets to destined for destination EID and hands over the packet to the EID resolution layer, which in turn is responsible for resolving EID to IP and thus routing the packet to the correct destination.

**Scalability**

Both EID and SID are flat names which creates quite a challenge to the scalability issue.

**Name Bindings**

The various name bindings are as follows i) ULD→Service Identifiers at Application level, ii) SID→EID at SID resolution layer between the Session layer and the Transport layer, iii) EID → IP at the EID resolution layer between the Transport Layer and the IP layer.

### *6.2.5 Intentional Naming System*

INS is a resource discovery and service location system designed for networks of mobile and dynamic devices and computers. Applications can describe what they are looking for (Intent) in a standard language consisting of attribute-value pairs. It forms an overlay network of name resolvers to bind User/Application intent to services and also to enable a late binding mechanism between the service names and locations to allow for communication session to continue even if this binding changes midway. An unique feature of INS is that the resolvers maintain a soft state weak consistency (every mapping is associated with a lifetime, to be renewed by constant periodic advertisements by the publisher) in the name mappings which makes it flexible enough to incorporate dynamic changes. INS also supports Intentional anycast based on application layer service level parameters such as server load etc and Intentional Multicast delivering the data to all services which satisfy a particular service name.

### *6.2.5.1 INS Architecture*

INS forms an application level overlay network of resolvers (INR) to resolve application intent to service names and also integrates application level message routing using the mechanism of “Late Binding”. Each service attaches itself to an INR and advertises its service in an attribute value format. Clients requesting services connect to an INR and send their requests using a simple query expression. The INR network provide the service of “Early Binding” and “Late Binding”. In Early binding the the INR resolves the user query to a set of matching services and returns the user with the set of IP addresses where the services are available. In late binding, the INR is responsible for a best effort routing of the message to the appropriate location. The INR’s maintain the mapping information by way of periodic dissemination of published services information among its neighbours through periodic updates or triggered updates (when an INR receives an update with new information). The published service names are maintained in a soft-state (with a time to live) and hence deletes entries which has not been advertised for a particular period of time. INS thus do not need services to de-register specifically. The INR’s form a spanning tree application level overlay network among themselves. A Domain Space Resolver (DSR) maintains a list of active and candidate INR’s .The DSR is the source of information about the overlay network for any new INR that wishes to join.

### *6.2.5.2 Objective summary of INS*

**Namespace**
INS does not introduce any particular new global namespace. It however contributes to the mapping of service names to user queries (Intention) based on a system of attribute-value pair description of services.

**Security**
INS certainly does not discuss any aspects related to security in its system.

**Mobility**
INS views mobility from two aspects, i) Node mobility and ii) Service mobility. Node mobility is a situation when the point of attachment of the communicating node changes due to mobility of the node. Service mobility is a situation where the service remains stationary but the best provider of a given service at a given instant of time changes based on application level parameter considerations such as system load etc. INS solves both the forms of mobility by introducing the idea of “Late Binding” wherein the mapping between the service name – location is avoided till actual message delivery time. Specifically, service mobility is supported through updated information about load conditions and other application level parameters of the various nodes maintained by the overlay network. Node mobility is supported through maintaining a soft mapping state at the overlay nodes and also having periodic updates or triggers to report about changes in the mapping. INS does not depend upon Mobile IP for this.

**IP Dependence**
INS is very much dependant on IP for actual network layer routing. The Overlay network is an application layer network and performs application level routing. The actual locater services are provided by the IP. If an application chooses late binding, INS does a best effort application level routing to reach the host based upon the IP infrastructure.

**Scalability**
INS considers the scalability issue in two areas: i) name lookup and ii) name update processing. To arrange for lookup scalability the DSR maintains a list of currently inactive candidate INR’s which a currently heavily loader INR may use to spawn an instance of itself. An INR is also allowed to drop off if it is sparsely

loaded after informing its peers and the DSR so that the spanning tree may be adjusted. The design of INS wherein every INR should know about all the names in the system creates a problem in terms of network availability and processing capacity of a given INR. INR proposes the technique of “virtual spaces”, partitioning the namespace into virtual spaces and giving each INR responsibility for a subset of all active virtual spaces, to resolve this bottleneck. The idea of “virtual spaces” may be considered as a precursor to the later revolutionary ideas of Distributed hash tables.

**Name Bindings**
INS has two schemes for name bindings. i) Early binding – Binding Service request queries → Service names → IP addresses at Service request time and ii) Late binding – binding service names→ IP address of service provider at message delivery time. This innovative scheme gives INS the ability to support service and node mobility.

### *6.2.6 Host Identity Protocol*

The Host Identity Protocol (HIP) is one of the most recent proposals among the identity/Location Split Architectures. HIP proposes the inclusion of a HIP layer between the transport and IP layer. The HIP layer serves the dual purpose of providing an independent identity to the transport layer and also providing a secure authentication mechanism to the hosts. The IP is used for routing purposes only.

#### *6.2.6.1 HIP Architecture*

In the Architecture of HIP, the transport endpoints are known by the cryptographic hash of a public key(Host Identity) known as the Host Identity Tag(HIT). The HIT is used during initial base exchange by the communicating host to set up a shared secret using diffie hellman key exchange. HIP architecture therafter depends on IPSec using the shared key for security of the session. HIT’s provides decoupling of the Identity/Locator roles of the IP in the sense that transport layer protocol can now just address packets using the corresponding HIT and hand over transport packets to the HIP layer. The HIP layer is responsible for the HIT – IP mapping for routing the packet to its desired location. There have been many proposals as to how HIP might achieve the mapping. One of the suggestions has been to use the DNS to store such mappings. But then such a scheme, though scalable would inherit the demerits of slow update propagation of DNS. Others have proposed various overlay based mechanisms. Overlay mechanisms, though much more responsive to updates in mapping information has the challenge of providing scalability, especially when HIT’s are 128 bits flat names. A significant architecture based on HIP is HI3 [49]. HI3 tries to use HIP over I3, thus using the Overlay mechanism of I3 and using HIT’s as I3 triggers for security.

#### *6.2.6.2 Objective Summary of HIP*

**Namespace**
HIP introduces a new namespace of Host Identities which are the public keys of a public/private cryptographic key pairs. Different encryption and authentication algorithms tend to use different sized keys. Hence, the host identity is mapped to fixed length (128 bits) key called Host identity tag (HIT) by applying a cryptographic hash over the Host Identity , for use in protocol headers. The HIT’s can be generated locally an has the property of being self certifying and statistically unique.

**Security**
One of the main design principles of HIP was to provide authentication of the communicating hosts and setup a shared secret between the authenticated hosts. HIP does not provide any mechanism for data security, a

responsibility it bestows on other security protocols such as IPSec, using the shared secret. The HIP specification leaves the responsibility of developing an effective public key distribution system for secure distribution of Host Identities on the implementer, suggesting the use of either Public key Infrastructure or DNS, and hence inheriting the security loopholes present therin. It also proposes the use of DNSSEC for obtaining a secure channel between the DNS and the host, but such a step does not remove the vulnerability of fraudlent publishers in DNS spoofing attack. The HIP base exchange for setting up a HIP association between two parties, is especially cautious about any form of Denial of Service attacks by introducing a novel methodology of puzzle solving by the initiator before the responder commits any state for the association.

**Mobility**
HIP provides locater/identity split and thus provides an independent existence of the Upper protocol layers. Though the HIP charter does not provide a clear cut mobility architecture, it nonetheless proposes two feasible schemes to support mobility. The first scheme involves direct signaling between the corresponding hosts through UPDATE packets in case of any change in the underlying IP mapping. However this scheme does not work in conditions of simultaneous movement at both ends and also in situations of locating a mobile host. In such cases, HIP proposes the deployment of a rendezvous system which receives updates from hosts regarding their Address changes and is responsible for dynamic HIT to IP mapping to support mobility.

**IP Dependence**
As in most locater/Identity split architecture, IP is reduced to being just locaters of hosts. The HIT's are just host identifiers and are not routable. The mapping between HIT and IP can be done at the Domain name servers in static scenarios and through rendezvous server systems in mobile scenarios.

**Scalability**
**F**or mapping HIT to IP addresses using the DNS, The scalability of the DNS is inherited. In case of rendezvous mechanisms and in fact any mechanisms in which HIT's are used to index large tables scalability would be an important consideration and it is left open for the implementer to decide on the best solution applicable. Also, since HIT's are statistically global flat identifiers, no central authority is responsible for maintaining their uniqueness.

**Name Bindings**
Upper layer protocol elements such as transport layer sockets and IPSec Security parameter Index all bind to the HIT. The HIT in turn bind to IP in the HIP sublayer to route packets to their destination.

### *6.2.7 Locator ID Split Protocol (LISP)*

LISP is amongst the latest proposals in the area of Location Identity split architecture family and as of this writing, an IRTF routing research group active proposal. We include this in our study to provide a flavor of the most recent developments that is part of active research in this area. The fundamental motivation of LISP derives from the present internet scenario of differing requirements of the end host sites and the providers. The end host sites expect to be multihomed for better fault tolerance and connectivity and achieve mobility , provider switch ( for multihomed sites) and renumbering with session survivability. For providers on the contrary , multihoming cause loss of Aggregation and hence cause scalability issues, efficient resource management require them to do traffic engineering and they need to provide scalable solutions and infrastructure for supporting end host mobility with session survivability. The rational behind LISP design is having provider independent hierarchical addressing (non topological and hence non agggregatable) for end hosts and topologically aggregatable addresses for RLOCs. The idea being to use hierarchical end host addresses for local routing and scalable indexing of End host → RLOC mapping database while using topologically aggregatable address of RLOC to do routing across the internet scalably.

### *6.2.7.1 LISP Architecture*

In LISP Architecture, each end site has one or more configured tunnel routers or RLOCs and are known as Egress tunnel router (ETR) or Ingress tunnel router(ITR) based on the role it is playing. The end host 'S' wishing to communicate with another end host 'D' in a different end site , gets to know D's EID by DNS lookup. 'S' then sends an IP packet destined towards 'D' to its default router which is configured as an ITR. The ITR prepends a LISP header to the packet with source as its own IP and the destination as 'D's IP. The packet is routed normally as today through the internet. When the egress router of 'D' gets the packet it performs a lookup in the EID-to-RLOC database. The ETR sends over this mapping to the ITR and subsequent packets for 'D' are tunneled between ITR and ETR and then forwarded by ETR to the destination. When a site is multihomed each tunnel router has an address from the address block from each of the provider address block.

### *6.2.7.2 Objective Summary of LISP*

**Namespace**

LISP does not introduce any new namespace but defines a way of namespace allocation based on the current IP addresses. The IP address of the End host sites are called EID's and they are supposed to be allocated in a Provider Independent manner and preferably have some sort of hierarchy for scalability in EID-to-RLOC database lookups. The RLOC's are provided aggregatable address from the provider address block to provide for scalable internet routing.

**Security**

LISP does not discuss much in aspects of security, the primary concern being solving present routing and addressing problems in the internet. The only explicit security concern discussed is that of database service security when obtaining mapping between EID-to-RLOC, to avoid ETR spoofing.

**Mobility**

Mobility with session survivability is realized as one of the most urgent needs of the infrastructure by LISP. Basically for fast end host mobility with session survivability, LISP proposes the existing solution of Mobile IP. However when a mobile node moves to foreign location, the home RLOC needs to learn the new EID-RLOC mapping of the mobile end host in the new network. The issue of fast network mobility has also been considered, but it is one of the future endevours of LISP.

**IP Dependence**

LISP is completely designed over the present IPv4 system and is supposed to be incrementally deployable over the present IP netwotk.The EID and RLOC are all IP addresses, differing only on the mode of allocation. The basic idea of LISP is based on IP tunneling between the RLOC's IP addresses to deliver packets between EID's associated with these RLOC's.

**Scalability**

Routing table scalability is achieved by having the RLOC's pick up addresses from a topological aggregatable address space. In EID-RLOC mapping databases, scalability is achieved by injecting some sort of hierarchy into the EID names. These are basically the two important areas where scalability is a serious concern.

**Name Binding**

The only explicit name binding that needs to take place is the EID-RLOC binding at the RLOC's.

*Summary*

The findings of this section may be summarized as in Table 4.The horizontal rows present the various attributes which form the basis for our comparison while the vertical columns list each of the locator-identity split architectures discussed thus far in this section. All the architectures discussed in this section address the common concern of semantic overloading of IP addresses and the constraints imposed on mobility and security therewith. Achieving a Locator/identity split would decouple the transport and session protocols from the IP layer and thus make way for their independent existence. However as we can see , it is not a trivial undertaking and requires a some serious engineering thought. One of the key aspects of such a design is that it requires the introduction of some sort of added infrastructure. Hence scalability becomes a serious .The numbers one deals with in the scenario of deploying anything on the internet scale, often runs in billions, if not more. Any proposal which does not take this into consideration is bound to fail. Also, the present internet is not the best place where one may leave things to trust. It is supposedly one of the most insecure service environments. Hence, security is a huge concern for any application that needs to run over it. A naming architecture which in itself is not secure enough can be of no good to the services running over it. The internet is yet to see widesclale deployment of any locater/identity split architecture and hence it is too early to comment on the exact requirements of such an architecture. But it may well be said that the obvious benefits of such an architecture would compel its wide scale deployment in the near future. We hope that our study may help future designers of the internet to learn from earlier experiences, and design a robust, scalable and secure architecture attaining locater / identity split.

Table 4: A comparison of the locator/Identity Split Architectures

| | FARA | DOA | I3 | LNA | INS | HIP | LISP |
|---|---|---|---|---|---|---|---|
| **Namespace** | No Global Namespace necessarily required, implementation dependant | Global End host Identifier :EID | Global flat namespace for trigger Identifiers | 2 Global flat names:<br>i) Service Identifiers<br>ii) EndPoint Identifiers | No Global names,<br>User queries in name-value pair form mapped to service names | Statistically global, 128 bit Host Identifier Tags generated through cryptographic hash of public keys called Host Identities | No new Global namespace, End host Identifiers and Routing Locaters are IP addresses varying in the way they are allocated |
| **Security** | Implementation dependant | EID – hash of public keys, ERecords signed by private keys | Public and Private trigger Identifiers | No explicit method, but proposes Host Identity based methods | No explicit method | Initial public key based authentication and shared secret key exchange using Host Identities**,** data security using IPSec or other security protocols | No explicit method |
| **Mobility** | Rendezvous based overlay mechanism | No explicit proposal | Overlay based signaling | Overlay based signaling or dynamic peer initiated lookup and rebind | Overlay based node and service mobility | Overlay based signaling or direct signaling as apropriate | Mobile IP |
| **IP Dependence** | Meta Architecture in which IP may be used in an instantiation as a forwarding directive | Used as locator for routing | Used as locator for routing | Used as locator for routing | Used as locator for routing | Used as locator for routing | Based on IP, EID's are unaggregatable IP addresses and act as Identifiers, Routing Locaters are aggregatable IP addresses and used for intra-site routing |
| **Scalability** | Implementation dependant | Distributed Hash Tables managing Flat names | Distributed Hash Tables managing Flat names | No explicit method suggested to scale the management of EID and SID which are flat namespaces | Concept of "virtual spaces" for a distributed load balancing architecture | HIT-IP mapping : DNS scalability inherited.<br>Managing HIT space: Implementation dependent | Routing Tables : Aggregation of RLOC's.<br>EID's – Internal hierarchical structure within EID's |
| **Name Binding** | Entity name →FD at Directory Service, FD Management Service responsible for rebinding during mobility | DNS→EID at DNS, EID→EID* →IP at infrastructure | Trigger identifiers → IP at Infrastructure | ULD→SID at Application level, ii) SID→EID at SID resolution layer, iii) EID → IP at the EID resolution layer | Service request queries →service names→IP addresses,<br>a)Early Binding<br>b)Late Binding | TCP,IPSec →HIT.<br>HIT →IP at HIP sublayer | EID→RLOC binding at RLOC |

### *6.3 Flat Address Space Routing Systems*

Flat address spaces are characterized by the absence of any form of implicit or explicit structure in their names. The use of such names in any form of architecture is generally justified by the requirements of the architecture as there are some obvious pros and cons in using such names. The primary benefit of flat names is the absence of any organizational structure which sometimes limit the extendability of an architecture. But this advantage comes for the price of scalability. Routing of flat names is a serious scalability issue as it is difficult to delegate responsibilities. Thankfully, the concept of Distributed Hash Tables can be used very effectively to make such routing architectures scalable. Another advantage of Flat names is that they can be generated locally and still ensure global uniqueness with very high probability (the measure of the probability depends on the size of the name). Also, names with some cryptographic quality such as cryptographic hashes or public keys generally have to be flat since they represent some large number. However, flat names are not human readable. Hence their use is limited to situations where machines are involved rather than humans.

We shall be discussing about a few architectures based on DHT which use flat names, in an attempt to understand some of the issues and their solutions as proposed by various research activities in this field. Some of the Architecture we have seen above such as HIP, EID of LNS etc use flat names at some level. We shall not be discussing those architectures again but rather try to focus on the principles owing which use of flat namespaces in a scalable way became possible.

We shall be discussing two architectures: CAN [59], Tapestry [75] in the same order to maintain the chronological ordering of these proposals. These discussions provides an insight into the way DHT's can be engineered to real architectural solutions to reap all its benefits.

#### *6.3.1 Content Addressable Network (CAN)*

CAN (Content Addressable Network) was one of the first proposals of putting the concept of DHT to real life Engineering problem. The motivation came from trying to find a scalable way to index content in a large peer-to-peer environment. Existing solutions either depended on central huge indexes or decentralized indexes with request flooding to locate the mapping information. CAN proposed the use of DHT to store these indexes by having the file content hash to some unique value called the key and the key being stored in a node deemed responsible for storing the key based on a key space partitioning algorithm. This way any content file could be stored with an unique key which can be determined in constant time and using the index entry for the key can be easily determined by a key based routing algorithm. The scalability of the proposed method derives from the fact that each node is responsible for a fraction of the total number of keys, keys being distributed evenly among the nodes, and the average number of hops traversed to find the node responsible for maintaining the key is $(d/4)* n^{1/d}$, where 'd' is the number of dimensions in the key space and 'n' is the number of participating nodes.

##### *6.3.1.1 Routing with CAN*

Each node maintains coordinate routing table having the IP → virtual coordinate space mapping of each of its Immediate neighbors in the coordinate space. Neighbors are coordinate spans overlapping in (d -1) dimensions and abut in one dimension Every node employs "greedy forwarding" mechanism to forward each message closer to its destination. Having more than one neighboring node increases robustness in cases of node failure.

#### *6.3.1.2 CAN Construction*

A new node (joining the overlay) needs to find out the IP of a node already in the CAN, maybe through some sort of DNS lookup. The new node then randomly chooses a point “P” and sends a JOIN request destined for P to the CAN node. The existing CAN node forwards this request by CAN routing till it reaches the node associated with P The node associated with “P” , splits its zone and passes on half of its zone information to the new node. After getting its zone, the new node learns about its coordinate neighbors from the node from which it inherits + the previous occupant itself. The previous occupant updates its tables eliminating nodes no longer its neighbors. New and old neighbors are informed of the changes. New node joining effects O (number of dimensions) nodes in the whole system

#### *6.3.1.3 CAN Node departure or Failure*

Node Departure is a situation where a node departs from the overlay after delegating its part of the responsibility to some neighboring node, while, Node failure is a situation where the overlay node is not able to pass on its responsibility, making the part of the namespace it represents to be out of service till one of the neighbors recognize the failure and gather enough information to be able to take its place.

### *6.3.2 Tapestry*

CAN uses shortest overlay path available which may be quite a number of IP hops and also uses run time heuristics to optimize while Tapestry uses optimal local routes right from initialization and maintains them in order to reduce routing stretch. Tapestry participating nodes have Node ID’s ($N_{id}$), Objects O are given Globally unique Identifiers ($G_{uid}$) and Applications within a host given $A_{id}$ (similar to a port number).

#### *6.3.2.1 Routing and Object Location*

Tapestry maintains information about the neighboring links in actual physical sense and assigns levels to them according to the length of prefix match.

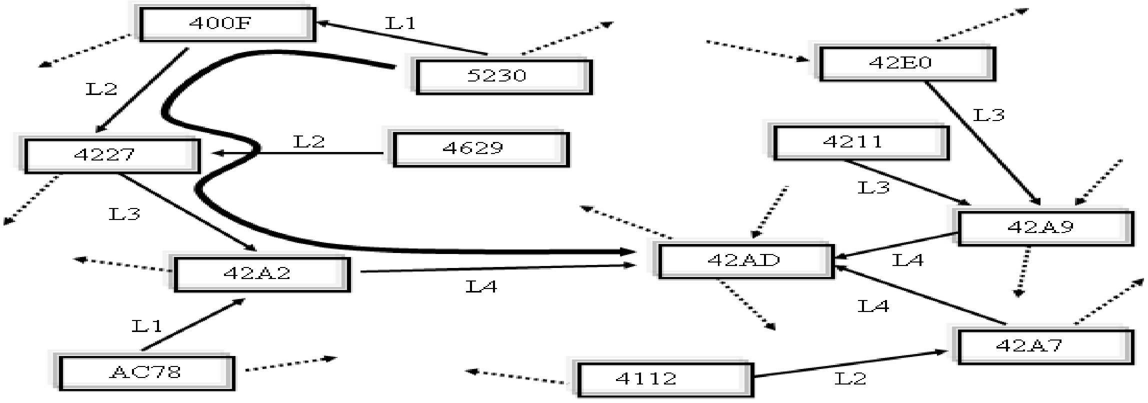


Figure 12: Routing in Tapestry

Whenever routing packets, they are forwarded to the highest level node according to prefix match. For instance, in routing from Node 5230 to Node 42AD (Figure 12), since there is no prefix match initially 5230 searches in its routing database for all level 1 location info it has and tries to find the highest prefix match for the destination. Each node tries to forward the message through a higher level match node and thus each hope takes the packet closer to its final destination.

### *6.3.2.2 Object publication and location*

A server S storing Object O with $G_{uid}$ $O_g$ and root $O_r$, publishes /advertises O by periodic public messages to root $O_{r.}$ The node id of the node storing O need not be equal to $O_{g.}$ If there exist a Node with $N_{id} = O_g$ then $O_r = O_g$ else $O_r$ = prefix "close to" $O_{g\ and}$ is deterministic. Each node along the publication path stores <S,$O_g$> for efficiency of mapping request for object O without having to go all the way to the root server storing the mapping for the object.

### *Summary*

There are other architectures of DHT which are equally popular such as CHORD, PASTRY [62] etc. However a discussion of CAN and TAPESTRY successfully captures the essence on which any DHT based architecture work. It is beyond the scope of our survey to make an objective study of the various DHT architectures. We have included this section merely for the sake of completeness of our work. An interested reader might follow the references for these architectures for having a better insight to these DHT based architectures

## 7 CONTROL PLANE FUNCTIONS AND ISSUES

Control Plane functions concern the organization and maintenance of the structure of the whole naming system - the communication between clients and servers to keep the name unique and consistent among the neighboring servers, to get locators bind with the given name, the procedure initiated by clients to discover if specific service is available, the caching of names in local host to expedite the mapping of the static name to locator, the replication between the name servers to keep the name service available and reliable; the delegation between name servers to make the system scalable and flexible.

In short, control plane functions of a naming system is dedicated to keep the structure of the system and thus make the name service available, reachable, reliable, and consistent.

## 7.1 Name resolution

A given name necessarily needs to be translated into a locator (or address) that allows the user to access the corresponding entities. This procedure is called name resolution or mapping. Generally, the resolution could be fulfilled by a *name lookup* between the client and the name server. For example, when a user tries to access the resource by a URL, the domain name component of the URL is resolved into an IP address by Domain Name System [46] by a lookup of operation between client application and domain name servers. The rest of the URL is resolved by application-specific components.

Some flat naming schemes such as UUID does not provide any resolution architecture, and an application using UUID is in charge of providing the name resolution mechanism. In XRL, the name resolution function is performed by an inter-process communication (IPC) call to the central process called *finder*, which tracks all the predefined and registered *target names* and the protocols supported. There are two formats of XRL: the unresolved and resolved XRL. For the unresolved XRL, as shown in figure 5 , the protocol family is set to "finder" and the protocol parameter is set to the target name to be invoked. The unresolved XRL is then resolved by finder and the "finder" is replaced by an appropriate protocol name and the target name is replaced with a network address. The resolved format is returned to the client and the client uses it to access the specified resource. For INS, a name-specifier can be resolved into one or more name records. And all the name-specifiers is stored in a name-tree. The name resolution process of INS uses the name-tree to lookup a name-specifier and returns the associated name-records.

## 7.2 Service Discovery

Here, we give two classical concepts related with naming derived from conventional telephone service: White Page (WP) data and Yellow Page (YP) data. In telephone services, White Page refer to getting a telephone number by the name of a person, while Yellow Page means getting the number by categories, address, and provider of the services. Generally speaking, White Page is suitable to be used to name people and Yellow Page is used to name services.

Service discovery means that for a given query attribute discover all the services available that match the query condition. Discovery differs from lookup in the nature of the query performed. When a name lookup is performed, the unique named object is usually known to whoever is performing the lookup. The aim of the lookup is to determine the value of some attributes of the object (such as address). When a discovery query is performed, however, the subject of the query is typically unknown. The aim of discovery is to determine the set of objects that have a certain set of attribute values. In other word, to discover a name or service, the operation could be performed by giving the description of the attributes of the services.

The discovery requirements for White Page and Yellow Page data are similar. Attribute-based architectures have proven superior for both types of data. Although the lookup and discovery queries for White Page and Yellow Page may be similar, the results may be handled differently. For example, when duplicate objects are returned from a White Page query, the user may need to further disambiguate the results to find the one true object. In the Yellow Page case, the user is often satisfied with any response and needs only to choose one.

### 7.3 Name Caching

Caching is a general mechanism wherein the client keeps certain resolved mapping between a name and a locator, or the local copies of information retrieved from previous queries.

DNS, which is of a hierarchical structure, has been proven to be efficient and scalable in the current Internet, but the upper-layer resolvers is potentially overloaded and may become a single points of failure. To solve this problem, DNS use a local caching mechanism at lower-layer DNS servers, which could significantly reduce the query hits for the upper-layer DNS servers and increase the efficiency of the resolution. However, although this local caching mechanism could increase performance and provide a measure for fault-tolerance, it could not meet the requirement of the system for supporting mobility, which is because it is difficult to propagate the name mapping changes through the system in a fast and efficient way. In the mobile scenario, the changing name could bring in, the problem of inconsistency for the system. That also explains why URL and DNS scheme fit for static names resolution but does not suit highly dynamic names.

### 7.4 Name Replication

Name replication could be achieved by replicating the name servers, which could prevent the single points of failure problem of the high-level name servers that handle a lot of naming resolution transactions. The replication mechanism increases the reliability and security of the naming system greatly, but also it could incur extra network overhead. For example, in the three-level naming tree of V-System, any failure in a global directory would cause a large sub-tree of the naming system to become inaccessible. V-System use replication for these servers due to the relatively slow rate of name changing at this level. But for large-scale Internet, replication like this would cost a large amount of extra overhead, and make it infeasible to deploy. Fortunately though, we could use some caching functionality to reduce the access rate for these high level servers. However, caching has the disadvantage of not being able to support dynamic names, i.e., to support mobility across the networks.

As an example, the high-level DNS servers are replicated, which make it fault-torrent. In INS, the name-tree is replicated in each INR (Intentional Name Resolver), so the name resolution of INS is performed locally in INR. To keep the replicating name-tree consistent among the INRs cost a lot of network traffic, especially for the highly dynamic networks. For Solar, as it uses the INR network, just like INS, the INR in Solar is also fully replicated. While Solar uses an overlay proxy server layer to redirect the dynamic name to the INR for further resolving, the proxy servers should also be replicated in the circumstance that services often move between proxies, which also incurs a considerable amount of network overhead.

### 7.5 Name Proxy

To increase the scalability and flexibility of a naming system, or to offer certain name resolution extension function, some naming systems employ a name proxy server. A name proxy server acts like a delegated agent for the name servers. For example, the proxy server layer in Solar [12] acts as an intermediate delegation server between dynamic clients and INS networks. By introducing these proxy servers, Solar could support mobile dynamical names.

In other cases such as NAT (Network Address Translation), the NAT boxes map one global address to multiple internal addresses has provided an ad-hoc solution to the depleting address space problem, and could address the security problem. However, for end-to-end applications, especially for future end-to-end name resolution and routing function, the NAT could meet some problems. Several schemes such as IPNL , 4+4 Plutarch and TRIAD all address the NAT problem for naming. Future naming schemes should provide transparency, efficiency, reliability and security for the end-to-end name resolution and traffic routing mechanisms even for the networks containing NAT.

To sum-up, in this section, several functions and issues related with the control plane have been addressed. These are important for the organization and maintenance of the structure of the whole naming system. In the next part, several non-typical naming schemes are given and discussed. They are not addressed in the former parts because they only address a very specific function or used in a very specific application related with names and naming. So we discuss them here to ensure the integrity of the paper.

## 8 Other Name Related Schemes

In this section, we will discuss some non-typical naming related schemes that address different problems. We did not discuss them during the former parts because these schemes aim at a small specific area or provide a specific solution concerning certain problem. They are not general schemes like URI, INS and Solar, which cover most of the functions of the three planes of the naming reference model.

### 8.1 HNS

“Heterogeneous Computer Systems” (HCS) was a research project of University of Washington. HNS (HCS Name Service) [65] was proposed to solve the naming problem among heterogeneous networks with different naming schemes and namespaces.

The general goal of HNS is to integrate different naming services into a global one without changing the current application or incurring any re-registration process.

HNS global namespace has two parts: A “*context*” identifies the local name service where resource can be found. An “*individual name*” determines the name of the object in that local service.

To resolve the name, HNS maps the name to a NSM (Name Semantic manager) to handle retrieval of data from the native server determined by “*context*”. Using NSM, Client can query the name and service without being concerned about the underlying name service. Figure 13 illustrates an example of integration between two different naming schemes by HNS. As the figure shows, a HCS client calls the HNS using heterogeneous RPC (HRPC), passing the HNS name. The HNS returns an HRPC Binding that allows the client to call the appropriate NSM. The client then calls the NSM directly using specific interface that includes original HNS name. The NSM translates the *individual name* portion to the corresponding local name, communicating with local service using this name, and return the results.

NSM is separated from both client and HNS, which allows reconfiguration without altering either, and adding new systems only requires building new NSM and registering it with HNS.

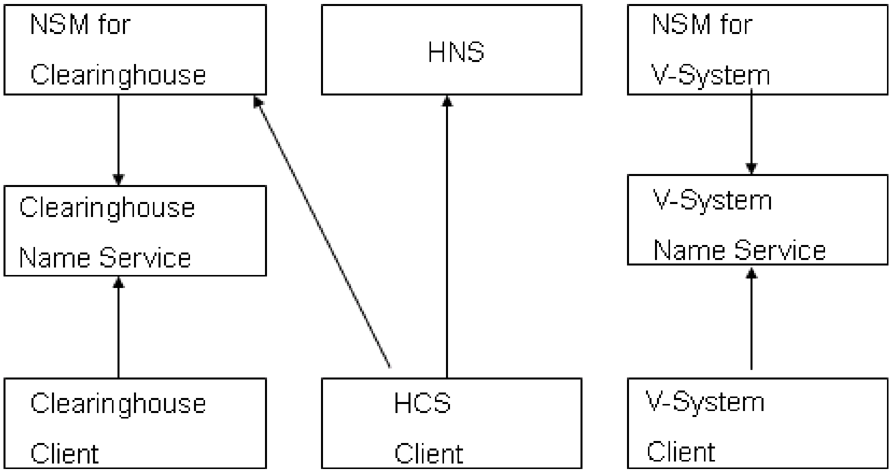


Figure 13: Example  scenario of HNS querying

## 8.2 UNS

Similar to HNS, Universal Name Semantics (UNS [28]) also tried to address the naming issues across heterogeneous networks and among the heterogeneous name spaces, but different from HNS, UNS tried to accomplish this by managing or merging different namespaces into one.

UNS attempted to partially automatically fulfill the naming translation (using a prototype language) between the different namespaces by exploiting “abstract similarity” i.e., merging different namespaces based on their similarity.

UNS is somewhat creative on addressing this heterogeneous naming issue, but this merging mechanism seem to be infeasible when the different naming scheme are completely different. For example, it will be very difficult to merge a naming scheme with flat name structure into a naming scheme using a descriptive name mechanism, since their “similarity” somehow is very limited.

## 8.3 HFN

HFN (Human-Friendly Names) [6] was proposed for highly popular and replicated websites (such as search engine) which may have very large volume of simultaneous access requests at the same time. The current scheme for these websites is to use multiple URLs to identify each replica. HFN builds on the present URN/URL system and uses descriptive human-friendly names (HFNs) to identify a resource. The HFN is resolved to one or more URNs and each URN can be resolved to one or more URLs. As figure 14 (a) shows, it is the current scheme of using a URL for each replica, the system will not be user-friendly and the replication is not transparent to the users. So we may use a location-independent URN to name the resource and since URN does not care about the location of the resource, it has to be mapped to multiple URL. Figure 14 (b) provides this transparent replication for the web servers. Since URN refers to a resource rather than the location, the resource can be moved around without changing its URN, the mechanism in figure 14 (b) thus can support mobile resources. However, since URNs are designed to be used for computers, they are not required to be human-friendly. So by introducing HFN, scheme in figure 14 (c) allows descriptive, human-friendly, and highly usable names.

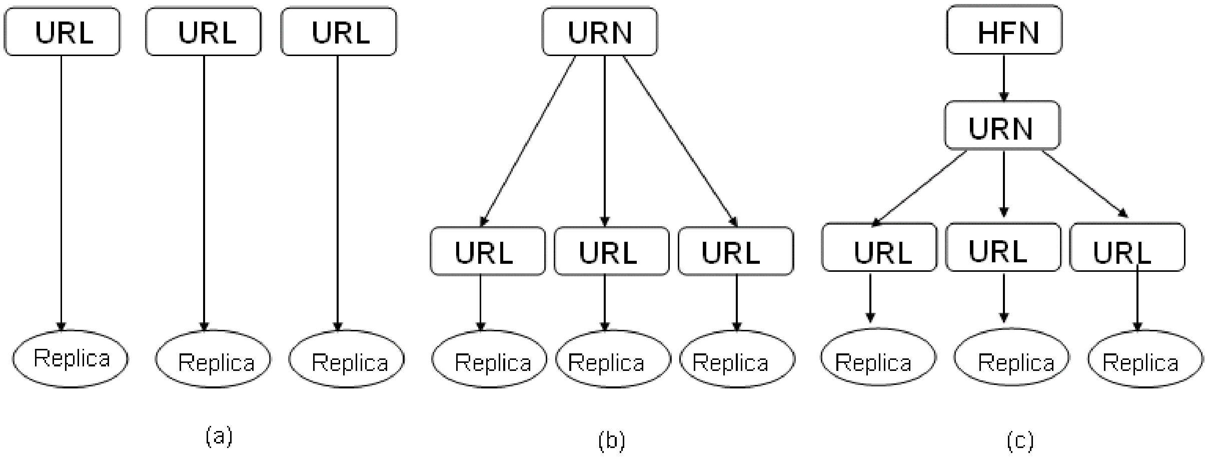


Figure 14: Comparison of three schemes for naming replicated resources

This two-step resolution process that binds the HFN to a URN and the URN to multiple URLs offer certain advantages. For example, replicating or moving a resource will not affect its name and users can freely change the HFN without affecting replica placement. HFN's HFN-to-URL resolution mechanism scheme support a large number of resources and resource distributed over a large geographical area.

However, HFN is specifically designed to meet the requirements of the highly popular and replicated web services. For common applications, its HFN-to-URN and URN-to-URL mapping mechanism could bring extra overhead. So this makes HFN inappropriate for a system supporting highly mobility and multi-homing functions.

### 8.4 CNRP

CNRP (Common Name Resolution Protocol) [43][56][57] is proposed by IETF CNRP working group. Common name is a word or a phrase, without imposed syntactic structure, that may be associated with a resource. It refers to things in the real world such as a trade name, company name, or a book title. These names are sometimes easier for people to remember and use than URLs, and don't have to worry about the problem with URLs that because of the limited syntax of URLs, companies and individuals are finding that the names that might be most reasonable  for their resources are being used elsewhere and are unavailable.

CNRP is a lightweight search protocol that lets users of client applications to communicate with common name resolution services in both the browser enhancement and search site paradigms. Name resolution services are not generic search services and thus do not need to provide complex Boolean query, relevance ranking or similar capabilities.

Resolving common names at different information providers permits acquiring different types of information. However the working group's scope does not include discovery, selection or implementation of a scalable common name resolution service, administration of resolution service, name registration, name ownership, and methods for creating, identifying or insuring unique common names.

Since Common Name Resolution Service is not based on web-content, but on highly structured directory service to identify a resource and the resolution service is external to the actual resource, so it is suitable to provide more specific service for the users, and it could act as a complement for the search engine. For example, it could provide business directories services such as real time stock information for NASDAQ; it could provide White Page services, E-commerce directories; publishing or entertainment directories; Yellow pages services, etc.

To sum up, in this section, we discussed several non-typical naming schemes. HNS and UNS tried to provide a tentative solution for the naming problems in heterogeneous systems using different naming schemes. HFN was proposed to provide better service for highly popular and replicated websites. CNRP gives a lightweight common name resolution protocol to provide specialized structural directory service, which could act as a complement to the current search engine.

## 9 NAMING SCHEME: FUTURE DIRECTIONS

The internet that we have today was designed and implemented around 40 years back. Since then it has undergone numerous modifications and enhancements, but the basic underlying principles of the design still remain the same. It can easily be awarded to have been one of the mot successful designs in the history of technology, having withstood the test of time for such an extended period. But technology in the world of communication sciences has taken massive strides since then and today we are forced to ask the question as to how would we have designed the internet if we were to design it today. One of the significant changes that we cannot fail to notice is the huge growth in the diversity of communication devices that we have today. And we cannot possibly overlook the fact that homogenization of the network into one single network type is clearly not the correct choice owing to diversities in technology and organizational / political disparities. The future of the internet is in having all these heterogeneous networks communicate with each other in their explicit contexts. In such a network, the problem of naming and addressing looms large. The future work of this survey points towards a successful design of a naming system which could address this problem of large scale convergence with explicit contexts. It is a mammoth undertaking and in our view, one of the most important considerations for the design of the next generation internet.

## 10 CONCLUSION

In this survey, we have presented a study of the various naming schemes that have been proposed for the present internet at various times and under various contexts. The uniqueness of our work, as opposed to other surveys in this are, lies in the fact that we have successfully delineate the problem of naming into three semantic planes which are somewhat orthogonal to each other. We believe, such a model of naming would allow future researchers in this area to be able to better organize their thoughts and hence come up with more practical and workable solutions. The motive of our work is to satisfy the survey needs of anyone envisioning to design a new naming scheme, hopefully for the next generation of the internet.

## ACKNOWLEDGMENT

We would like to thank Dr. Mic Bowman of Intel for encouraging us to work on this problem and helping define the problem.

## REFERENCES

[1] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley, "The design and implementation of an intentional naming system," ACM SIGOPS Operating Systems Review , Proceedings of the seventeenth ACM symposium on Operating systems principles (SOSP '99), Volume 33, Issue 5, December 1999

[2] R. Ahmed, R. Boutaba, F. Cuervo, et al, "Service naming in large-scale and multi-domain networks," IEEE Communications Surveys & Tutorials, Volume 7, Issue 3, Third Quarter 2005

[3] Tuomas Aura, Aarthi Nagarajan, and Andrei Gurtov, "Analysis of the HIP Base Exchange Protocol," Proceedings of ACISP 2005, published on Lecture Notes in Computer Science, Volume 3574, pp. 481-493, 2005, Springer-Verlag Berlin Heidelberg

[4] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish, "A Layered Naming Architecture for the Internet," Proceedings of ACM SIGCOMM'04, Volume 34, Issue 4, Portland, Oregon, USA, Aug. 30–Sept. 3, 2004

[5] Hari Balakrishnan, Scott Shenker, and Michael Walfish, "Peering Peer-to-Peer Providers," 4th International Workshop on Peer-to-Peer System(IPTPS '05), February 2005

[6] G.Ballintijn, M.van Steen, A.S.Tanenbaum, "Scalable human-friendly resource names," IEEE Internet Computing, Volume 5, Issue 5, Sept.-Oct. 2001

[7] Alberto Bartoli, Sape J. Mullender, Martijn van der Valk, "Wide-address spaces: exploring the design space," ACM SIGOPS Operating Systems Review, Volume 27, Issue 1, January 1993

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

[9] T.Berners-Lee, L.Masinter, and M.McCahill, "Uniform Resource Locators (URL)," RFC 1783, DEC 1994

[10]A. D. Birrell, R. Levin, M. D. Schroeder, R. M. Needham, "Grapevine: an exercise in distributed computing," Communications of the ACM, Volume 25, Issue 4, April 1982

[11]Mic Bowman, Saumya K. Debray, Larry L. Peterson, "Reasoning about naming systems," ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 15 Issue 5, November 1993

[12]Guanling Chen, D. Kotz, "Context-sensitive resource discovery," Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003), pp. 243- 252, March 2003

[13]D. R. Cheriton, "The V Distributed System," Communications of the ACM, Volume 31, Issue3, 1988

[14]D.R. Cheriton, "Decentralizing a Global Naming Service for Improved Performance and Fault Tolerance," ACM Transactions on Computer Systems (TOCS), Volume 7, Issue 2, May 1989

[15]D.R. Cheriton and M. Gritter, "TRIAD: A New Next-Generation Internet Architecture," Stanford University, Stanford, CA, Tech. Rep., 2001

[16]David Clark , Robert Bragden, Aaron Falk, and Venkata Pingali, "FARA:Reorganizing the Addressing Architecture," ACM SIGCOMM 2003 Workshops, Karlsruhe, Germany, August 25-27, 2003

[17]Jon Crowcroft, Steven Hand, Richard Mortier, Timothy Roscoe, and Andrew Warfield, "Plutarch: An Argument for Network Pluralism," ACM SIGCOMM 2003 Workshops, Karlsruhe, Germany, August 25-27, 2003

[18]L. Daigle, D. Van Gulik, et al, "URN Namespace Definition Mechanisms," RFC 2611, June 1999

[19]L. Daigle, D.W. van Gulik, et al, “Uniform Resource Names (URN) Namespace Definition Mechanisms,” RFC 3406, October 2002

[20]DARPA Internet Program, “Internet Protocol,” RFC 791, September 1981

[21]S. Deering and R. Hinden, “Internet Protocol Version 6 (IPv6) Specification,” RFC 2460, December 1998

[22]Jakob Eriksson, Michalis Faloutsos, and Srikanth Krishnamurthy, “PeerNet: Pushing Peer-to-Peer Down the Stack,” Proceedings of the 2003 International Workshop on Peer-To-Peer Systems (IPTPS ’03), Volume 2735, pp. 268-277, 2003

[23]D. Farinacci, V. Fuller, D.Oran, and D. Meyer, “Internet Draft: Locator/ID Separation Protocol(LISP),” draft-farinacci-LISP-03, August 13, 2007

[24]Andrei Gurtov, Dmitry Korzun, and Pekka Nikander, “Hi3: An Efficient and Secure Networking Architecture for Mobile Hosts,” HIIT Techinal Report, June 2005

[25]B. M. Hauzeur, “A model for naming, addressing and routing,” ACM Transactions on Information Systems (TOIS), Volume 4, Issue 4, December 1986

[26]T. Henderson and P. Nikander, “Internet Draft: Using the Host Identity Protocol with Legacy Applications,” draft-ietf-hip-applications-01, April 9, 2007

[27]T. Henderson, “Internet Draft: End-Host Mobility and Multihoming with the host Identity Protocol,” draft-ietf-hip-mm-05.txt, March 2, 2007

[28]Nigel Hinds and C. V. Ravishankar, “Name space models for locating services,” Proceedings of the 1991 conference of the Centre for Advanced Studies on Collaborative research (CASCON '91), October 1991

[29]Philip Homburg, Maarten van Steen, and Andrew S. Tanenbaum, “An architecture for a wide area distributed system,” Proceedings of the 7th workshop on ACM SIGOPS European workshop: Systems support for worldwide applications, September 1996

[30]Raj Jain, “Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation,” Milcom 2006, Washington DC, October 2006

[31]D. Johonson, C. Perkins, and J. Arkko, “Mobility Support in IPv6”, RFC 3775, June 2004

[32]P. Jokela, R. Moskowitz, and P. Nikander, “Internet Draft: Using ESP transport format with HIP,” draft-ietf-hip-esp-06.txt, June 11, 2007

[33]Anssi Kalliolahti, “Route Maintenance by Delegation,” Helsinki University of Technology, Seminar on Networking, 2005. Available at: http://www.tml.tkk.fi/Publications/C/18/kalliolahti.pdf

[34]M. Komu, “Internet Draft: Native Application ProgrammingInterfaces for SHIM APIs,” draft-ietf-hip-native-api-02.txt, July 7, 2007

[35]M. Komu, S. Schuetz, and M. Stiemerling, “Internet Draft: HIP Extensions for the Traversal of Network Address Translators,” draft-ietf-hip-nat-traversal-02.txt, July 6, 2007

[36]T.Koponen, A Gurtov, and P Nikander, “Application Mobility with HIP,” Proceedings of NDSS Wireless and Mobile Security Workshop, 2005

[37]J. Laganier, T. Koponen, L. Eggert, “Internet Draft: Host Identity Protocol (HIP) Registration Extension,” draft-ietf-hip-registration-02.txt, June 7, 2006

[38]J. Laganier,L. Eggert, “Internet Draft: Host Identity Protocol (HIP) Rendezvous Extension,” draft-ietf-hip-rvs-05.txt, June 7, 2006

[39]Keith A. Lantz, Judy L. Edighoffer, and Bruce L. Hitson, “Towards a universal directory service,” Proceedings of the fourth annual ACM symposium on Principles of distributed computing PODC 1985, August 1985

[40]P. Leach, M. Mealling, and R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace”, RFC 4122, July 2005

[41]Chaoying Ma, “On building very large naming systems, ” Proceedings of the 5th workshop on ACM SIGOPS European workshop: Models and paradigms for distributed systems structuring, September 1992

[42]M. S. Madan, “On naming considerations for networks,” ACM SIGCOMM Computer Communication Review, Volume 15, Issue 3 , July 1985

[43]M. Mealling, “The 'go' URI Scheme for the Common Name Resolution Protocol,” RFC 3368, August 2002.

[44]Minutes of Address Lifetime Expectations Working Group. Proceedings of 29th IETF Meeting, Seattle, April 1994

[45]R. Moats, “URN Syntax,” RFC 2141, May 1997

[46]P. Mockapetris, “Domain Names- Implementation and Specification,” RFC 1035, November, 1987

[47]R. Moskowitz, P. Nikander and P. Jokela, “Host Identity Protocol(HIP) Architecture,” RFC4423, May 2006

[48]P Nikander, J Ylitalo, and J Wall, “Integrating security, mobility, and multi-homing in a HIP way,” Network and Distributed Systems Security Symposium (NDSS’03) , 2003

[49]Pekka Nikander, Jari Arkko, and Borje Ohlman, “Host Identity Indirection Infrastructure (Hi3),” Proceedings of Second Swedish National Computer Networking Workshop, 2004

[50]P. Nikander, and J. Laganier, “Internet Draft: Host Identity Protocol (HIP) Domain Name System (DNS) Extensions,” draft-ietf-hip-dns-09.txt, April 13, 2007

[51]D.C. Oppen and Y.K. Dalal, “The Clearinghouse: A decentralized agent for locating named objects in a distributed environment,” ACM Transaction on Office Information Systems, July 1983

[52]Paul Francis and Ramakrishna Gummadi, “IPNL: A NAT-Extended Internet Architecture,” SIGCOMM’01, San Diego, California, USA , August 27-31, 2001

[53]C. Perkins, “IP Mobility Support”, RFC 2002, October 1996

[54]C. Perkins, “IP Mobility Support for IPv4”, RFC 3220, January 2002

[55]Larry L. Peterson, “The profile naming service,” ACM Transactions on Computer Systems (TOCS), Volume 6, Issue 4, November 1988

[56]N. Popp, M. Mealling, and L. Masinter, “Context and Goals for Common Name Resolution,” RFC 2972, October 2000

[57]N. Popp and M. Mealling, “Common Name Resolution Protocol (CNRP)”, RFC 3367, August 2002

[58]Jordi Pujol, Stefan Schmid, Lars Eggert, Marcus Brunner and Jurgen Quittek, “Scalability Analysis of theTurfNet Naming and Routing Architecture,” Proceedings of the 1$^{st}$ ACM workshop on Dynamic interconnection of network (DIN’05), Cologne, Germany, September 2, 2005

[59]Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, “A Scalable Content-Addressable Network,” ACM SIGCOMM’01, San Diego, California, USA, August 27-31, 2001

[60]Sylvia Ratnasamy, Scott Shenker, Steven McCanne, “Towards an Evolvable Internet Architecture,” Proceedings of ACM SIGCOMM'05, Volume 35, Issue 4, Philadelphia, Pennsylvania, USA, August 22-26, 2005

[61]J. Rosenberg, Ed., “What’s in a Name: False Assumption about DNS Names,” RFC 4367, February 2006

[62]Antony Rowstron and Peter Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems,” Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany, November 2001.

[63]J. Saltzer, “On the naming and binding of network destinations,” RFC 1498, August 1993.

[64]Mikko Sarela, Pekka Nikander, “Applying Host Identity Protocol to tactical networks,” Proceedings of IEEE MILCOM 2004, Monterey, California, USA, November 2004

[65]M. Schwartz, J. Zahorjan, D. Notkin, “A name service for evolving heterogeneous systems,” ACM SIGOPS Operating Systems Review, Proceedings of the eleventh ACM Symposium on Operating systems principles (SOSP ’87), Volume 21, Issue 5, November 1987

[66]R. Sharpe, “Just what is SMB?” October, 2002. Available at http://samba.org/cifs/docs/what-is-smb.html

[67]S. Shepler et al., “Network File System (NFS) Version 4,” RFC 3530, April 2003

[68]Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana, “Internet Indirection Infrastructure,” ACM SIGCOMM ’02, Pittsburgh, Pennsylvania, USA, 2002

[69]Hannes Tschofenig, Andrei Gurtov, Jukka Ylitalo, Aarthi Nagarajan, and Murugaraj Shanmugam, “Traversing Middleboxes with the Host Identity Protocol,” Proceedings of the Tenth Australasian Conference on Information Security and Privacy (ACISP ’05), Brisbane, Australia, July 4-6, 2005

[70]M Walfish, H Balakrishnan, “The location/identity split is useful for middleboxes, too,” Proceedings of Workshop on HIP and Related Architectures, November 2004

[71]Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker, “Middleboxes No Longer considered Harmful,” USENIX Association OSDI ’04: 6th Symposium on Operating Systems Design and Implementation, 2004

[72]Daniel J.Weitzner, “Whose Name Is It, Anyway? Decentralized Identity Systems on the Web,” IEEE Internet Computing, Volume 11, Issue 4, July-Aug 2007

[73]XORP Inter-Process Communication Library Overview Version 1.4, XORP Project, International Computer Science Institute, Berkeley, March 2007. Available at http://www.xorp.org

[74]Jukka Ylitalo, Patrik Salmela, and Hannes Tschofenig, “SPINAT: Integrating IPsec into Overlay Routing,” Security and Privacy for Emerging Areas in Communications Networks (SecureComm ’05), pp. 315- 326, September 05-09, 2005

[75]Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz, “Tapestry: A Resilient Global-Scale Overlay for Service Deployment,” IEEE Journal on selected area in communications, Volume 22, No. 1, pp. 41-53, January 2004

[76]Zoltan Turanyi, Andras Valko, and Andrew T. Campbell, “4+4: An Architecture for Evolving the Internet Address Space Back Toward Transparency,” ACM SIGCOMM Computer Communications Review ,Volume 33, No. 5, pp. 43-54, October 2003