An Analysis of Naming Conventions for Distributed Computer Systems

Douglas B. Terry

Computer Systems Research Group
Computer Science Division
University of California
Berkeley, CA 94720

ABSTRACT

Name servers that collectively manage a global name space facilitate sharing of resources in a large internetwork by providing means of locating named objects. The efficiency with which the name space can be managed is strongly influenced by the adopted naming convention. Structured name spaces are shown to simplify name space management from both an administrative and system viewpoint. Formulae have been derived which allow one to quantitatively measure the effect of the distributed name server configuration on a given client's level of performance. In general, the cost of a name server query can be reduced by distributing replicated copies of name server database entries in a way that exploits the locality of clients' reference patterns.

1. Introduction

Names have been widely used in computer systems as a convenient way of referring to shared resources [3, 4, 6, 8]. A *name* can be simply defined as a character string identifying some object. Names are preferred to lower level identifiers such as memory pointers, disk block numbers, or network addresses because they typically indicate something about the contents or function of their referents and are easily transmitted between users.

To facilitate sharing, standard ways of naming objects, called *naming conventions*, must be established. In many existing computer systems, separate naming conventions are used for different types of objects. For example, file names typically differ from user names. Due to the emergence of widely distributed systems in which a large number of computers are interconnected, the

This research was sponsored by the Defense Advance Research Projects Agency (DOD) Arpa Order No. 4031 and monitored by the Naval Electronic System Command under Contract No. N00039-C-0235.

need for uniform naming conventions to identify the many available resources is becoming critical.

The set of names complying with a given naming convention is called the *name space*. Name spaces are managed by *name servers*. For the purposes of this paper, a name server is defined to be any processor which stores information about named objects and provides facilities which enable users to access that information. Name servers act as distributed binding agents that bind an object's name to some of its properties, including the object's location. The name servers that store the information about a particular object are called the naming authorities or *authoritative* name servers for that object.

For environments consisting of a substantial number of interconnected networks with a possibly large number of hosts, the cost of communication between clients and name servers is the major bottleneck in locating remote resources. In such an environment, the performance of name server queries is dominated by the number of name servers that must be accessed and the cost of accessing those name servers. The group of name servers that collectively manage the name space should be configured so as to minimize this cost for the average client.

Once a naming convention has been adopted, the many factors affecting the efficiency with which the name space can be managed and the cost of retrieving name server information include:

- the clients' patterns of reference to name server information,
- the choice of authoritative name servers for parts of the name space,
- the placement of name servers throughout the internet,
- the amount of replication of name server information,
- the number of name servers that are currently operational,
- the performance of each individual name server.

The last factor has been neglected in this paper since standard performance evaluation and improvement techniques can be applied to enhance an individual name server's level of performance. Also, additional name servers can be employed if existing ones become overloaded.

The next section presents a model which incorporates the other factors so that the cost of a name server lookup can be quantified for various name server configurations. Several alternative schemes for naming objects or resources are identified in sections 3 through 6. For each scheme, the associated name space management strategies are analyzed. Section 7 provides an example of how the derived cost formulas may be applied to an existing network configuration.

2. The Model

The name server database is distributed among N servers, $NS_1 ... NS_N$. At any point in time, some fraction of these servers will be accessible; the others may have crashed or become detached from the network. F represents the current number of name servers whose data is inaccessible because of some failure. Name server failures are assumed to be uniformly distributed. That is, all name servers are presumed to have the same mean time to failure, and hence, crash with equal frequency and probability.

The various name server clients are enumerated $1 ... U$. The term "client" may refer to a specific program, host, network, or some combination thereof. In general, clients are distinguished by their location in the internet relative to the name servers and by the particular objects they reference.

A name server client need only know the location of a single name server, presumably the closest one, to make use of the name service. Name server queries are assumed to be iterative: if the primary name server, NS_{main}, is unable to answer the query then it returns the location of a more knowledgeable colleague. Several iterations may be necessary for some naming conventions and management strategies.

The name server database is strictly partitioned into K fragments, which often reflect partitions in the name space. In the degenerate case, each database entry is a separate fragment. The database partitions, $db_1 \ldots db_K$, correspond to indivisible units of storage. That is, either the complete database fragment is stored at a given name server or none of it is.

Each name server has authority over some subset of the database partitions. Typically, no single name server stores the complete database. The set S_k contains those name servers that store partition db_k. S_k, for $k = 1 \ldots K$, is a subset of $\{NS_1 \ldots NS_N\}$.

For each name server, d_i denotes the cost of executing a query at NS_i. This cost, which could depend on such things as the overall size of the database maintained by NS_i and the kind of database facilities employed, is assumed to be fixed over time.

The round trip transmission cost between client u and name server i is given by c_{ui}. Observe that c_{ui} strongly depends on the site at which the client is executing. It varies according to the number of gateways traversed and the speeds of the intermediate transmission lines. The number of bytes transferred is assumed to have a negligible influence on the communication cost since name server queries and responses are generally quite small. Variations due to network congestion are also ignored.

For a given name server, NS_i, C_{ui} specifies the complete cost of accessing that name server remotely from client u. This cost includes both the communication and processing costs. Hence, C_{ui} is the sum of d_i and c_{ui}.

Each client has a set of objects (or resources) that it regularly references. Different clients generally use different sets of objects. Client u's reference mix is represented by $r_{u1} \ldots r_{uK}$. That is, r_{uk} is the percentage of name server accesses performed by client u to the database partition db_k.

In summary, the distributed name server configuration is characterized as follows:

$NS_1 \ldots NS_N \equiv$ set of name servers
$1 \ldots U \equiv$ name server clients
$db_1 \ldots db_K \equiv$ name server database partitions
$S_k \equiv$ set of authoritative name servers for db_k
$r_{uk} \equiv$ fraction of client u's accesses to db_k
$c_{ui} \equiv$ cost of communicating with NS_i from client u
$d_i \equiv$ cost of executing a query at NS_i
$C_{ui} \equiv c_{ui} + d_i$
$F \equiv$ number of failed name servers

The cost of retrieving the name server information about a set of named resources varies per client according to the client's location relative to the various name servers and the client's reference mix. The expected value of this cost for client u is denoted by $E(L_u)$. In general,

$$E(L_u) = \sum_{k=1}^{K} r_{uk} L_{uk}$$

where L_{uk} represents the total cost of querying the information in database partition db_k, including the cost of locating the desired data. This cost may involve accessing configuration data from one or more name servers.

Although a client's reference mix, which the system designer has no control over, contributes significantly to the client's expected name server lookup cost, it plays no part in the cost of retrieving an individual object's name server entry. Thus, the following sections ignore the clients' access patterns in $E(L_u)$ and concentrate on formulating L_{uk}. The client subscript u is left out of the formulas to increase their clarity; this can be safely done since the performance observed by a particular client is independent of the locations of other clients.

3. Flat Name Space

The simplest naming convention that one can imagine is a flat name space where names are character strings exhibiting no structure. For managing such a name space, two obvious choices exist: store the complete name server database at all sites or store each name server entry at some arbitrary site.

In the first case, $S_k = \{NS_1 \ldots NS_N\}$. The primary name server, generally the physically closest one, can always be queried since it contains the complete set of information about all named objects in the environment. Thus, the retrieval cost is simply

$$L_k = C_{main} \ .$$

This approach has been used by the Arpanet, for example, in which the complete host table mapping host names to host numbers is copied to every host. For widely distributed environments containing a large number of named objects, the effort needed to distribute the entire database and maintain consistency between copies may be prohibitive. Even the cost of storing this huge database may be excessive for many installations. The Arpanet maintainers have been experiencing precisely these problems as the Arpanet has vastly expanded in recent years.

If the naming information is stored at a single name server, $S_k = \{NS_i\}$, then the storage and consistency problems with the previous approach are avoided. However, since the name space offers no clues as to a database entry's whereabouts, locating the desired data may necessitate querying each name server in succession until the authoritative one is discovered. If the authoritative name server for an object is chosen at random then half of the name servers must be accessed on the average to retrieve the object's information. The retrieval cost becomes

$$L_k = \sum_{j=1}^{i} C_j \ ,$$

assuming that the name servers are queried in numerical order.

Whereas the first approach was costly in terms of storage, this second approach is costly in terms of name server lookups. Neither of them is very practical for large environments. The principle problem with a flat name space is the difficulty in locating the storage site for the named information.

4. Physically Partitioned

By adding structure to an object's name that reflects the management authority for the name, some of the difficulty in ascertaining the object's authoritative name server can be alleviated. Specifically, the naming convention *proper_name@name_server* could be adopted. The *name_server* part of an object's name identifies the name server that is responsible for managing information about the object. The *proper_name* unambiguously identifies the object in the context of the naming authority. The complete name *proper_name@name_server* is thus globally unambiguous as long as the name servers are unambiguously named.

A two part naming convention of this sort partitions the name space in a way that simplifies its management. Not only does the appendage of the naming authority to an object's name facilitate locating the data about the object, but it also simplifies name assignment since the *proper name* need only be locally unambiguous. That is, each name server has sufficient information to guarantee the global unambiguity of a newly assigned object name. To effectively manage a name space following such a naming convention, each name server must know the location of every other name server, as well as storing the naming information for all objects under its authority.

A name space of this sort is said to be *physically partitioned* since a name reveals the physical storage site of information about its referent. Also, a name server is generally the naming authority for objects in its proximity, and thus the name space reflects the physical distribution of objects in the internetwork. A prime example is the naming convention often used by electronic mail applications: *user@host*. In this case, every host is the naming authority for all users who receive mail on that host.

With a physically partitioned name space, a one-to-one mapping exists between database partitions and name servers. That is, $K=N$ and $S_k = \{NS_k\}$. Two accesses are required to obtain the information about a given object: one to locate the naming authority and one to access the data. A special case arises if the desired naming information is stored at the primary name server; in this case, only a single access is required since the main name server can recognize that it is the authority and return the data directly. The cost of a lookup is thus

$$L_k = \begin{cases} C_{main} + C_k & \text{if } k \neq main \\ C_{main} & \text{if } k = main \end{cases} .$$

However, if the total number of name servers is small, clients can easily cache the network addresses of the various name servers, thereby reducing the cost to

$$L_k = C_k .$$

The access to the local name server has been eliminated since the individual hosts are knowledgeable enough to query the correct storage site directly. The resulting lookup algorithm is optimal given the assumptions that naming data is stored exactly once.

5. Organizationally Partitioned

5.1. Overview

Even though a physically partitioned name space has been shown to perform very well in terms of lookup time, it is not the favored naming convention of many systems for managerial reasons. Resource names are primarily assigned by people, and people are generally under various administrative authorities working on various projects. These different administrations may share a name server but wish to maintain complete control over their resources and the names of their resources. Hence, they often prefer a name space that is *organizationally partitioned.* Names take the form *proper_name@organization* in which the authority for assigning names is explicitly recognized.

With such a naming scheme, the database partitions correspond to organizations rather than name servers; each name server can be the authority for some subset of the organizations. Since the assignment of object names is independent of the assignment of responsibility for maintaining information about the objects, the name service can be easily reconfigured. That is, new name servers can be added to the environment and acquire authority over part of the existing name space; application programs which rely on the name service are unaffected since the object names do not reflect the name server configuration.

Suppose, for a moment, that each organization's data is managed by a single name server as with a physically partitioned name space, $S_k = \{NS_i\}$. In order to efficiently locate an object's name server data, each name server should know which server has responsibility for each organization. This way, name server queries can be processed in two steps as before. First, the primary name server maps the organization name to the authoritative name server for that organization and returns its network address. Then the remote name server is contacted to retrieve the appropriate naming information. The lookup cost is basically the same as for physically partitioned data,

$$L_k = C_{main} + C_i ,$$

except that two database retrievals are always required since a name server can not determine whether or not it is the authority for the desired data without consulting the local database. One round trip transmission cost can be saved, however, if the primary name server retrieves and returns the name server entry directly upon discovering that it is the storage site for the desired data. Thus,

$$L_k = C_{main} + d_{main} \quad \text{if } S_k = \{NS_{main}\} .$$

5.2. Replicated Data

Perhaps the greatest advantage of an organizationally partitioned name space is the ease with which replicated naming information can be accommodated. In a distributed system, replication can enhance the availability of the data by providing several independent sources for retrieval and can improve performance by allowing the data to be stored closer to where it is most frequently accessed. These benefits can be obtained by making the name server database partially redundant, that is, by storing the database partitions as replicated copies at many sites. The cardinality of set S_k is the degree of replication.

To support replication, the lookup algorithm described previously for an organizationally partitioned name space could be easily extended to recognize several authoritative name servers for each organization rather than just a single responsible one. Each name server could know the complete set of authoritative name servers for each organization, S_k. The Grapevine system [1] developed at Xerox Palo Alto Research Center, for example, is based on this approach. The Grapevine name space is partitioned into ***registries*** which can be arbitrarily dispersed among the various ***registration servers***. Techniques are employed to maintain consistency among the various copies.

Any available copy of an organization's name server data can be used to answer queries. For performance reasons, accessing the closest authoritative name server for the named object is generally desirable. In the Grapevine system, each registration server maintains a complete list of the other servers ordered by distance. This may not be feasible for a large environment with a substantial number of name servers [5].

Assuming that the closest authoritative name server, $NS_{\min_k} \in S_k$, can be determined with negligible cost, the name server lookup cost becomes

$$L_k = \begin{cases} C_{main} + C_{\min_k} & \text{if } \min_k \neq main \\ C_{main} + d_{main} & \text{if } \min_k = main \end{cases} .$$

Although this formula looks similar to the previous formulas, the cost should be less with replicated data since the name server accessed by various clients, $NS_{\min_k}$, could differ from client to client, whereas before each client was forced to access the same server. Nevertheless, without some knowledge of how the authoritative name servers are selected and how many exist for a given database partition, comparing the costs of the different name space management techniques is very difficult.

One simple approach would be to distribute R copies ($R \leq N$) of the name server data uniformly. In other words, R authoritative name servers are chosen at random for each database partition. Without loss of generality, assume for the moment that the name servers are ordered such that $C_i \leq C_j$ for $i < j$ and $NS_1 = NS_{main}$. Under this assumption, the expected lookup cost can be computed as follows,

$$L_{uniform} = C_{main} + Prob\,(main = \min_k)\; d_{main} + \sum_{i=2}^{N} Prob\,(i = \min_k)\; C_i$$

$$= C_{main} + \frac{R}{N} d_{main} + \sum_{i=2}^{N} \frac{\binom{N-i}{R-1}}{\binom{N}{R}} C_i \; .$$

This formula allows one to quantitatively determine the benefit of replication on performance by increasing the value of R. Of course, the benefits that can be achieved depend greatly on the physical configuration of the internet and the placement of the name servers.

5.3. Name Server Failures

With partially redundant name server data, the failure of a name server should potentially degrade performance but should not render any information unavailable provided the number of failures is less than the degree of replication. The effect of name server failures on performance can be gauged by incorporating such failures into the previous lookup cost formula. For F failed name servers selected at random, $F < R$, this formula remains

$$L_{uniform} = C_{main} + Prob\,(main = \min_k)\; d_{main} + \sum_{i=2}^{N} Prob\,(i = \min_k)\; C_i \,.$$

But the probability of retrieving the desired resource information from name server i becomes substantially more complex,

$$\begin{aligned}
&Prob\,(i = \min_k) \\
&= \sum_{q=0}^{R-1} Prob\,(\textit{name server } i \textit{ stores the data} \\
&\qquad \text{and } \textit{name server } i \textit{ is not dead} \\
&\qquad \text{and } \textit{q closer name servers store the data} \\
&\qquad \text{and } \textit{all q closer name servers are dead})
\end{aligned}$$

$$\begin{aligned}
&= \sum_{q=0}^{R-1} \frac{\binom{i-1}{q}\binom{N-i}{R-q-1}}{\binom{N}{R}} Prob\,(\textit{name server } i \textit{ is not dead}) \\
&\qquad \times Prob\,(\textit{all q closer name servers are dead} \mid \textit{name server } i \textit{ is not dead})
\end{aligned}$$

$$\begin{aligned}
&= \sum_{q=0}^{R-1} \frac{\binom{i-1}{q}\binom{N-i}{R-q-1}}{\binom{N}{R}} [1 - \frac{F}{N}] \\
&\qquad \times \prod_{j=0}^{q-1} Prob\,(\textit{j+1st is dead} \mid \textit{j are dead } \text{and } \textit{name server i is not dead})
\end{aligned}$$

$$= \sum_{q=0}^{R-1} \frac{\binom{i-1}{q}\binom{N-i}{R-q-1}}{\binom{N}{R}} [1 - \frac{F}{N}] \prod_{j=0}^{q-1} \frac{F-j}{N-1-j} \,.$$

If the number of name server failures exceeds the degree of replication then it is no longer possible to assume that all data is still available. Hence, determining the performance degradation is not feasible. Nevertheless, it is instructive to look at the probability that a given piece of data is inaccessible, that is, all responsible name servers for the information have crashed. This probability,

$$P(\mathit{data\ inaccessible}) = \prod_{l=0}^{R-1} \frac{F-l}{N-l},$$

is valid for any values of F and R, although it is always zero for $F < R$.

6. Hierarchical Names

The techniques previously described and analyzed for two part structured names can be extended to ***hierarchical*** names consisting of more than two parts. Hierarchical names, which have been used in file systems for many years, have recently been adopted for naming network objects. System designers may choose to either fix the number of levels or allow an arbitrary hierarchy. The new DARPA Internet Domain Naming Convention [7] , for instance, uses an unbounded tree structured name space for identifying network hosts and mail recipients, while the Xerox clearinghouse [2] enforces a three-part naming structure. In general, each layer represents either a physical or organizational partitioning of the name space.

With hierarchical name space management, the complete name server configuration data (S_k for all k) need not be stored at every name server. It is sufficient that each name server store only enough information to locate the authoritative name servers for the top level of the hierarchy. The top level name servers, in turn, should know the authoritative name servers for the name space subtrees directly under their administrative control. The amount of configuration data that must be maintained by name servers at the various levels of the hierarchy is proportional to the degree of branching of the name space tree. For this reason, hierarchical naming conventions with several levels are often better suited for naming large numbers of objects.

The analysis of hierarchical naming schemes can be performed by recursively applying the techniques for two part names. For example, to analyze the clearinghouse system, where names are of the form *local_name:domain:organization*, the lookup cost formulas for an organizationally partitioned name space, such as Grapevine from which the clearinghouse evolved, can be applied in two steps. The two part name *domain:organization* can be considered the convention for naming Grapevine registries. The algorithm remains the same: lookup the authorities for the registry then contact the closest authority to retrieve the desired information. However, the first step now involves looking up the authorities for the organization and then contacting the closest one to retrieve the authorities for the domain. The cost formula for accessing the clearinghouse is thus

$$L_k = (C_{main} + C_{\min_{org}}) + C_{\min_k} .$$

The analysis for hierarchies of more than three levels is a straightforward extension.

7. A Sample Environment

In practice, the cost formulas derived for name server queries can be applied to existing environments to analyze and subsequently improve the performance of the system, or they can aid in making design decisions when configuring a new system. For instance, a network administrator may wish to assess the benefits of increased replication or the addition of a new name server.

As an illustration, consider the network topology of Figure 1.† The circles

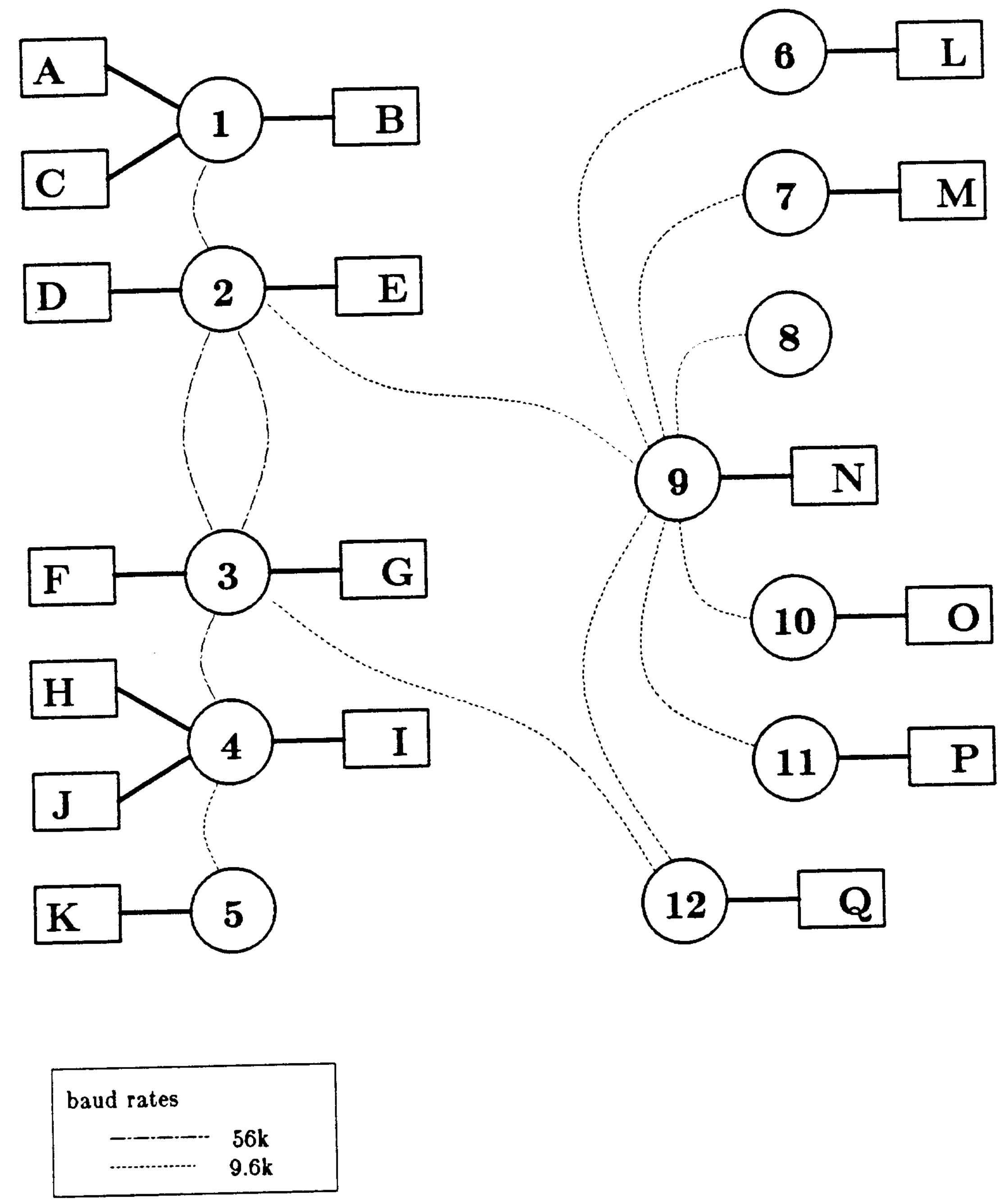

Figure 1. A Sample Internet

represent 3 megabits/second Ethernets, while the lines are long distance links with data rates of either 56 kilobits/second or 9.6 kilobits/second. The local

† This is the configuration of Grapevine servers in everyday use at the Xerox Palo Alto Research Center as of summer 1983 [5]. It is intended to serve as a sample widely distributed environment.

networks are numbered from 1 to 12. The rectangles depict the various name servers, labeled from A to Q.

from	to server								
network	A	B	C	D	E	F	G	H	I
1	1	1	1	56	56	111	111	166	166
2	56	56	56	1	1	56	56	111	111
3	111	111	111	56	56	1	1	56	56
4	166	166	166	111	111	56	56	1	1
5	479	479	479	424	424	369	369	314	314
6	682	682	682	627	627	682	682	737	737
7	682	682	682	627	627	682	682	737	737
8	682	682	682	627	627	682	682	737	737
9	369	369	369	314	314	369	369	424	424
10	682	682	682	627	627	682	682	737	737
11	682	682	682	627	627	682	682	737	737
12	424	424	424	369	369	314	314	369	369

from	to server							
network	J	K	L	M	N	O	P	Q
1	166	479	682	682	369	682	682	424
2	111	424	627	627	314	627	627	369
3	56	369	682	682	369	682	682	314
4	1	314	737	737	424	737	737	369
5	314	1	1050	1050	737	1050	1050	682
6	737	1050	1	627	314	627	627	627
7	737	1050	627	1	314	627	627	627
8	737	1050	627	627	314	627	627	627
9	424	737	314	314	1	314	314	314
10	737	1050	627	627	314	1	627	627
11	737	1050	627	627	314	627	1	627
12	369	682	627	627	314	627	627	1

Table 1. Communication costs

Generally, the values for c_i and d_i would be obtained from measurement studies. For sake of this example, fictitious, but reasonable, numbers have been assigned for these quantities. Communication costs are normalized such that communicating over a local Ethernet costs one time unit T. Assuming that the communication cost is proportional to the data transmission rate of the communication medium then transmission over a 56K bps line costs approximately $54T$, and similarly, communication over a 9.6K bps line costs around $312T$. The host to host communication costs, then, are derived by adding the costs of the various communication links traversed. Table 1 enumerates the costs (in units of T) of communicating between a process on each network with each name server.

The database access cost, d_i, is taken to be $6T$. This is in accordance with experience indicating that for retrieval over a local network the cost of the data query generally dominates the communication costs by about a factor of 4 to 8.

client's	replication factor R =					
network	1	2	3	4	5	6
1	297.35	145.99	84.83	56.72	41.53	32.15
2	261.76	124.96	74.40	53.69	43.39	36.94
3	271.47	125.76	74.24	53.63	43.38	36.94
4	300.59	142.34	82.65	55.96	41.34	32.12
5	576.76	398.65	323.48	281.71	251.52	226.25
6	645.18	548.70	488.14	435.57	387.66	343.13
7	645.18	548.70	488.14	435.57	387.66	343.13
8	976.59	896.22	849.47	808.41	769.71	732.08
9	369.00	307.04	278.71	256.05	235.77	216.55
10	645.18	548.70	488.14	435.57	387.66	343.13
11	645.18	548.70	488.14	435.57	387.66	343.13
12	439.41	347.85	302.26	271.68	246.62	223.93
avg.	506.14	390.30	335.21	298.34	268.66	242.46
Δ%	---	-22.89	-14.11	-11.00	-9.95	-9.75

Table 2. Effects of replication on L_k

Suppose that a two part organizationally partitioned name space is being managed by the collection of name servers. Table 2 gives the effects of replication on the performance of name server retrievals. The number of copies of each partition has been varied from 1 to 6. The expected cost of a name server query is given for a client on each of the 12 networks.

On the average, having two copies of the data instead of one reduced the expected lookup cost by over 22%. For networks 1-4, which are connected by high speed lines, improvements of over 50% are achieved. Notice that clients on network 8, which has no local name server and is separated from the rest of the world by low speed lines, suffer the worst performance. Furthermore, replication doesn't help them as much as others. Networks 1 and 4, which have three local name servers apiece, benefit the most from replication. In all cases, adding an additional copy of the name server data has a substantial impact on performance regardless of the replication factor. These performance increases are due entirely to reducing the amount of communication between clients and very remote name servers.

client's	number of failures F =				
network	0	1	2	3	4
1	41.53	46.00	51.42	58.08	66.36
2	43.39	46.42	50.22	55.06	61.32
3	43.38	46.39	50.17	55.01	61.31
4	41.34	45.64	50.83	57.21	65.19
5	251.52	260.40	270.13	281.03	293.53
6	387.66	401.75	416.19	431.01	446.33
7	387.66	401.75	416.19	431.01	446.33
8	769.71	781.09	792.65	804.42	816.53
9	235.77	241.73	247.87	254.24	260.92
10	387.66	401.75	416.19	431.01	446.33
11	387.66	401.75	416.19	431.01	446.33
12	246.62	253.99	261.77	270.09	279.14
avg.	268.66	277.39	286.65	296.60	307.47
Δ%	---	3.25	3.34	3.47	3.66

Table 3. Effects of failures on L_k for $R = 5$

The effect of failures on the cost of retrieving name server information is presented in Table 3. Again, the results are given for clients on each network and averaged over all networks. These results indicate that name server failures actually degrade performance by very little for a replication factor of 5. Even if almost a fourth of the name servers are down, the expected lookup cost increases by only 15% on the average, and around 50% for the worst case. The availability of name server data, not performance, appears to be the primary concern when considering name server failures.

8. Conclusions

Distributed name servers provide a valuable tool for managing and locating objects in a large internetwork. Two important choices must be made in designing a distributed naming service. First, uniform conventions for naming objects or resources should be adopted. Second, techniques must be devised for managing the resulting name space. These two choices have been explored together in this paper since naming conventions have a substantial influence on the management of resources complying with those conventions.

In selecting a naming convention, the set of objects can be either named with a flat homogeneous name space or partitioned into disjoint classes. When partitioning is done syntactically, which is generally the case, the name structure reflects physical or organizational associations. Hierarchical naming conventions that successively partition the name space at various levels are becoming popular. Hierarchically structured names result in a tree structured name space.

In order to answer a query, a name server must be able to locate, given only the object's name, the site(s) that maintain data about the desired object. This is the role of name space management. To achieve adequate performance, the object's name itself must convey sufficient information to allow the determination of a small subset of name servers to query. Name server configuration data provides valuable assistance in locating the authoritative name server(s) for a given named object. However, the amount of configuration data that must be maintained can be substantial in the general case.

Techniques for clustering named objects into partitions enable the amount of configuration data required in each name server to be reduced since it need only be maintained for each cluster and not each individual object. These techniques, in general, rely on particulars of the naming scheme. Syntactically partitioned name spaces provide natural object clusters, whereas flat name spaces are difficult to manage efficiently. Hence, structured names simplify management of the name space from both an administrative and system viewpoint.

Once a naming convention and associated name space management strategy have been selected, the observed performance of name server queries is dictated by the placement of the name servers and the distribution of the name server database. The lookup cost formulas derived throughout the paper allow one to quantitatively measure the impact of the distributed name server configuration on a given client's level of performance. The name servers should be configured so as to minimize the sum of the expected lookup costs for all clients. This basically involves exploiting the locality of client reference patterns.

In practice, a range of localities can be observed. In electronic mail applications, which have been the predominant clients of name servers, person-to-person mail is often quite localized. That is, users frequently send mail to other users in their same organization, group, or geographic area. These users are typically in a common name space partition, and hence, managed by common name servers. On the other hand, general interest distribution lists that include a substantial fraction of the entire user community may exhibit no locality. Such lists tend to ignore organizational and physical boundaries. Accesses to these lists are more unformly distributed.

Physical access patterns are dependent on both the frequency of accesses to the name server database entries (r_{uk} for $k=1...K$) and the mapping of data to storage sites (S_k for $k=1...K$). In other words, the amount of physical locality achievable in practice depends on the distribution of clients that are interested in a particular name server entry. The storage sites for the copies of a particular object's data must be carefully selected to coincide with the regions of interest. The selection of a fixed number of storage sites at random was analyzed as a particularly naive configuration technique. For such a scheme, the degree of replication of database entries was shown to considerably impact the cost of accessing a given database entry. In cases where some reference locality exists, the cost formulas based on randomly selected storage sites can serve as a lower bound on performance.

Finally, the concentration in this paper on the performance issues of distributed name servers is not meant to imply that performance is the only criteria to be considered in selecting a naming convention. Unfortunately, the performance aspects are the most overlooked and least understood. Other considerations have been briefly mentioned in places although they have not been adequately addressed. In general, the choice of a naming convention involves weighing the many tradeoffs between ease of name assignment, object mobility, availability of naming information, individual autonomy, storage space requirements, and the cost of maintaining data consistency.

Acknowledgements

I would like to thank my advisor, Professor Robert S. Fabry, for the support he has provided throughout the course of this research. I am also grateful to Professor Domenico Ferrari and my technical editor, Margaret Butler, for their thoughtful comments on earlier drafts of this paper. Special thanks go to Michael Schroeder of the Xerox Palo Alto Research Center for kindly allowing me to reproduce the Grapevine configuration in Figure 1.

References

[1] Birrell, A., Levin, R., Needham, R. M., and Schroeder, M. D..
Grapevine: An Exercise in Distributed Computing.
Communications of the ACM **25**, 4 (April 1982), 260-274.

[2] Oppen, D. C. and Dalal, Y. K..
The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment.
ACM Transactions on Office Information Systems 1, 3 (July 1983), 230-253.

[3] Saltzer, J. H..
Naming and Binding of Objects.
In *Operating Systems: An Advanced Course*, edited by R. Bayer, Springer-Verlag, 1978.

[4] Saltzer, J. H..
On the Naming and Binding of Network Destinations.
Proceedings IFIP/TC6 International Symposium on Local Computer Networks, Florence, Italy, April 19-21, 1982, pages 311-317.

[5] Schroeder, M. D., Birrell, A. D., and Needham, R. M..
Experience with Grapevine: The Growth of a Distributed System.
ACM Transactions on Computer Systems, to appear....

[6] Shoch, J. F..
Internetwork Naming, Addressing, and Routing.
Proceedings 17th IEEE Computer Society International Conference (COMPCON), September 1978, pages 72-79.

[7] Su, Z. and Postel, J..
The Domain Naming Convention for Internet User Applications.
Network Information Center, SRI International, RFC 819, August 1982.

[8] Watson, R. W..
Identifiers (Naming) in Distributed Systems.
In *Distributed Systems - Architecture and Implementation*, edited by B. W. Lampson, Springer-Verlag, 1981.