Date: May 4, 1979 IEN 103

An Experimental Network Information Center Name Server (NICNAME)

J. R. Pickens, E. J. Feinler, and J. E. Mathis - SRI

Introduction

This IEN reports a preliminary design and implementation of an experimental NIC-based name server. A name server is essentially a transaction based inquiry-response process which returns information on entities which can be named or addressed--hosts in this particular case. The Arpanet Network Information Center (NIC) maintains and distributes the Official Host Table [1] for the Arpanet, as well as a variety of other related information.

The motivation for this development comes both from current needs in the Arpanet community for such a service, and from the similar but wider needs of the burgeoning Internet community. Existing Arpanet needs are exemplified by the NIC charter to provide formatted Host Table information [1]. Existing Internet needs are exemplified by the need for terminal interface units (TIUs) on ANY network to have dynamic access to addresses of internet service hosts.

A name server service, as described herein, will permit more efficient access to, and distribution of, host information within the Arpanet. It can also support a need for host information, especially pertaining to the Arpanet, from the Internet environment.

The name server function is evolving. Before much of what is proposed can be provided, or even agreed upon, additional administrative and technical design decisions are required. The purpose of this note, therefore, is to catalyze an expanded discussion of the functions and facilities for the name server service.

The discussion is structured as follows: Section 1 contains a description of the current service and how it is derived from the NIC database service. Section 2 describes possible extensions of the name server concept by allowing a richer syntax, and by allowing queries for services to be built on top of the queries for host addresses. Section 3 discusses architectural issues, and presents some preliminary thinking on how we can get from the current centralized, hierarchic name server service to a distributed service with one (or more) name servers per network.

1. The SRI Name Server Experiment

An experimental name server has been installed on SRI-KA. It is accessed via a variant of the Internet Name Server protocol [2]. The

ITEM	TYPE	
Official Name	Text	
Alias Names	Text	
Host Address	Integer(32)	InterNet Services
Port	Integer(32)	
Protocol	Integer(8)	
Host Address	Integer(24)	Arpanet NCP Services
Socket	Integer(32)	

Syntactically, service queries can be derived from host queries by the addition of a service name field, as below:

"!NET!REST!SERVICE"

A network independent service query, for example, can be represented as:

"!*!*!SERVICE"

2.3. Name Server Options

The need for options has already been suggested in the discussion of the "similar names" function. Another group of options may be used to specify the format of the reply. At one extreme is the compact, binary, style, such as is currently specified. At the other extreme is an expanded, textual, style, such as would be represented by a host table record, and with official/alias names included. Options can be envisaged which specify:

- binary vs text format
- inclusion of each field in the reply
- inclusion of official name, per field, in the reply
- inclusion of alias names, per field, in the reply
- inclusion of other miscellaneous information, such as operating system, machine type, access restrictions, etc.

Other options can be envisioned which specify the scope of the search, such as "ignore TIPS and USER hosts". Likewise, an alternate form for specifying formats may be to settle on several standard ones, and allow an option to select between them.

Certainly, not all name servers will be able to support all such options, and not all options are equally useful. Thus, the proposed

initial implementation uses the internet header of TCP 2.5 [3] and will be converted to Internet Version 4 [4] once it becomes available.

The information which drives the name server service originates from the Arpanet Official Host Table. (A new host table format suitable for representing host information for multiple networks has been designed and will be described in a forthcoming RFC [5]) A massaged version of this database is presented to the name server, upon program initiation, as an input file. Work is in progress to investigate the feasibility of abstracting host related information from the NIC database management system via direct system calls.

User access to the service takes two forms. In the first form, a simple user process is provided to format user input into name server requests. In response to a query of the form "HOST !ARPA!FOO-TENEX" is returned an address such as "10 2 0 9" (NET=10, HOST=2, LHOST=0, IMP=9); the details of the user interface will, of course, vary from system to system.

This first primitive form of name server access has been implemented on several Arpanet and PRNET sites as PDP-10 TENEX and LSI-11 TIU programs. While initially the TENEX program is of little practical value since all sites have a complete name table, the LSI-11 program is intended to augment the TIU's host table. The scenario here is that when the packet radio TIU comes alive, it contains only a minimal host table, including the addresses of perhaps a few candidate name servers. The user can query the name server with a simple manual query process and then use the address obtained to open a TELNET connection to the desired host.

In the second form of access, soon to be operational on packet radio TIUs, a process-level interface is provided that mediates between internal processes and the name server. The design issues for something other than a demonstration system are complex and involve tradeoffs. The most obvious tradeoff is in the area of network traffic versus "freshness" of information. The local host table handler can either send a message to the name server for every address request or it can perform some type of local caching to remember frequently requested names. SRI is currently implementing a process-level interface for the LSI-11 TIU's TELNET program in order to explore the problems of local host table management in small machines in a dynamic environment.

2. Name Server Issues

The name server, as currently specified, provides a simple address binding service [2]. In response to a datagram query [4, 6], the name server returns either an address, a list of similar names, or an error. Several useful additional functions can be envisioned for the name server such as service queries and broader access to host related information. First, however, a few refinements to the current name

server specification are proposed.

2.1. Refinements

The current specification needs clarification as to how to interpret the "similar names" error response. Should there be a fixed definition of what "similar names" means, or should it be left open to the whims of the implementor?

This function seems to be most useful in providing helpful information to a human interfacing process. It may be useful to model the behavior of the name server on the behavior of other known processes which present host-name information on demand. An example of this is a common implementation of User Telnet [7], in which three kinds of functions occur:

1. On termination of name input (e.g. <CR>), the user is only "beeped" if the name is not unique. If the name is unique, the name is filled out, and the requested operation is initiated.

2. In response to <ESC>, the name will be filled out if unique, or the user will get "beeped" if the name is not unique.

3. Only in response to "?" will a list of similar names be printed. "Similar names", in this case, means all names which begin with the same character string. The list is alphabetized.

In support of this style of user interface, it may be more appropriate to return the "similar names" response only when requested. Two ways to achieve this are 1) to set an option bit and 2) to use "?" to force the similar names response.

A second point upon which the specification may be enhanced is in the interpretation given to null network and host fields in the query string. Currently, if the network field is left out, as in "!REST" (normal query is "!NET!REST"), a local network query is assumed. "!!REST" and "!NET!" are not discussed in the current specification, and are presumably syntax errors.

Since host names tend to be unique anyway (at least at the present time) and since there is no way to make a network independent query under the current design, it may be useful to add to the notion of "null field", meaning "local", the notion of a special character like "*" which means "all".

The semantic range of queries afforded by adopting this convention is enumerated below (note: ~ is used to mean "null". Both network and host fields null ("!!") is, therefore, represented as "~ ~". N means "network" and R means "rest"):

~ ~	local net, local host (validity check?)
~ *	local net, all hosts
~ R	local net, named host
* ~	all nets, local host (inverse search)
* *	all nets, all hosts (probably prohibited)
* R	all nets, named host (today's situation)
N ~	named net, local host (inverse search)
N *	named net, all hosts
N R	named net, named host

By combining the on-demand-similar-names function, "all" and "local", and by allowing "*" to be prepended or postpended to the query string, one can have queries such as the following:

!!BBN*?	All hosts named BBN* on local net
!*!BBN*?	All hosts named BBN* on all nets
!*!*UNIX*?	All hosts named *UNIX* on all nets

2.2. Service Queries

It has been suggested that the name server can be generalized into that of a binding function [8]. In this context, a very useful extension is to allow service queries. One very real application of this service, which exists within the Packet Radio Project at SRI, is the need to find the addresses of Hosts which support the LoaderServer Service (the LoaderServer service allows packet radio TIUs to receive executable programs via down-line loading).

A characteristic of service querying, contrasted to host names querying, is the need for multiple responses. The requester would, upon receiving multiple service descriptors, attempt to establish access to each service, one-at-a-time, until successful.

Service descriptors are composed of at least the following (with more items probably required):

list will be expanded or contracted to fit the actual needs of processes using the name server service.

2.4. "More" Data

It is probably apparent to the discerning reader that several of the proposed name server extensions have the potential for generating more than a single datagram's worth of reply (576 octets max [9]). (Not of any consolation is the fact that the current practical PRNet Packet size is on the order of 256 octets.) Yet the size of such replies is not anticipated to require a full-blown streaming protocol. Several alternatives exist:

1. Disallow options which imply large replies,

2. Truncate the packet for large replies,

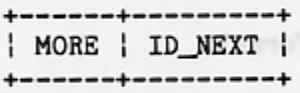
3. Ignore the recommended maximum datagram size,

4. Utilize an alternate base protocol for such requests,

5. Develop a "more data" pseudo-streaming protocol.

Alternative 1 may be chosen, but even within the current specification the potential for overflow exists (however remote). Alternative 2 implies unpredictable behaviors to the user of the name server service. Alternative 3 reduces the availability of the service. Alternative 4 is certainly possible, but may be over-kill.

Alternative 5 can be very simple. The concept is that the name server would return, as part of the reply, a code of the following form:



ID_NEXT is a name-server-chosen-quantity (1,2,4 octets?), syntax/semantics unspecified, which allows the name server to find the next block of reply, the next time it is queried. This quantity may be an internal pointer, a block number, or whatever the name server chooses. Follow-on queries may be implemented by recomputing the entire original query, discarding output until the ID_NEXT block is reached, or by efficiently storing the entire reply in a cache, fragmented into blocks (with appropriate decay algorithms).

2.5. Dynamic Updates

In all of the previous discussions, the host name database was assumed to be a static (or slowly changing entity) with an administrative and manual update authority. This model was

implemented for expediency and will well serve most of the needs of the Arpanet and Internet communities. However, a need can be envisioned for dynamic automated updating of the host table; (imagine the impact on the current system of any host who changed its address more than once a week!)

In a closed user group community (such as a local network of mutually trusting hosts), dynamic updating becomes simply a technical question concerning packet formats. In wider communities, a mechanism to authenticate the change request must be developed. Since the issues on authentication are outside the scope of this paper, we can only note that significant advances in practical deployment of dispersed processing and central services, such as automated host table management, can only be made when the problems of authentication become tractable.

3. Architecture

The name server concept is invaluable in allowing hosts with incomplete knowledge of the network address space to obtain full access to network services. Whether for reasons of insufficient Kernel space or of a dynamically changing environment, the need for the service is little questioned. The more significant issues, however, revolve around the methods for providing the service and for administering and updating the database.

In the current experiment, the service is centralized, and is supported by a database administered centrally by the NIC. In the long range, other architectures are possible which address ways to distribute host information within and between networks and administrative entities. These present opportunities for more dynamic, automated, approaches to the maintenance and sharing of data--particularly host name data.

From an evolutionary point of view, the name server service will likely exist initially as a centralized service, possibly with one large name server that has multiple network knowledge. From this beginning, an expansion in two orthogonal directions is possible.

- In the direction of internal distribution, the name server can be fragmented into multiple cooperating processes, on separate hosts. The data base can be replicated exactly or managed as a distributed database.

- In the direction of administrative distribution, multiple autonomous name servers may exist which exchange data in an appropriately administered fashion, on a per network or other administrative basis.

On the part of hosts with small host tables, a possibility for caching exists, where local, temporary copies are maintained of

subsets of the addressing database. Such copies may be obtained either by remembering previous queries made of name servers, or by receiving automatic distributions of data from name servers. For mobile hosts, in which even the home network is unknown, it is possible to maintain essentially an empty host table.

The potential exists, with service queries, for every host to contain a very primitive name server function. In response to a query of the form "!*!*!RealNameServer" is returned the address of a real name server service.

Finally, the possibility exists for multiple name servers to communicate dynamically, such as in attempting to resolve a query. If, for example, a name server on the Arpanet receives a query for a host on the Packet Radio Net, then the Arpanet name server can conceivably query the Packet radio net name server in order to resolve the reply.

4. Conclusion

In this note, a collection of design ideas on the name server service has been presented. An experimental service, based on the NIC host table database has been reported.

A continuing examination of the name server service is encouraged, scoping out the requirements and specifying its functional distribution. A level of service comparable to that outlined currently [2] will be provided initially, but a more expanded service merits consideration and discussion. Certainly many open questions have been raised in proposing an expansion of the service, but it is expected that such an expansion will result in more useful support of internet (and intranet) capability.

References

1. M. D. Kudlick and E. J. Feinler, Host Names On-line, RFC 608, SRI International, January 1974.

2. J. Postel, Internet Name Server, IEN 61, USC-Information Sciences Institute, October 1978.

3. V. Cerf, TCP Version 2 Specification, IEN 5, March 1977.

4. J. Postel, Internet Datagram Protocol, IEN 81, USC-Information Sciences Institute, February 1979.

5. E. J. Feinler, Proposed Official Host Table Format, SRI International (RFC in preparation).

6. D. Reed, J. Postel, User Datagram Protocol, IEN 71, USC-Information Sciences Institute, January 1979.

7. E. Leavitt et al, TENEX USER'S GUIDE, Bolt Beranek and Newman Inc.

8. Y. Dalal, Group discussion, January 24,25 1979 Internet Meeting.

9. J. Postel, Internet Meeting Notes - 25&26 January 1979, pp. 12, IEN 76, USC-Information Sciences Institute, February 1979.