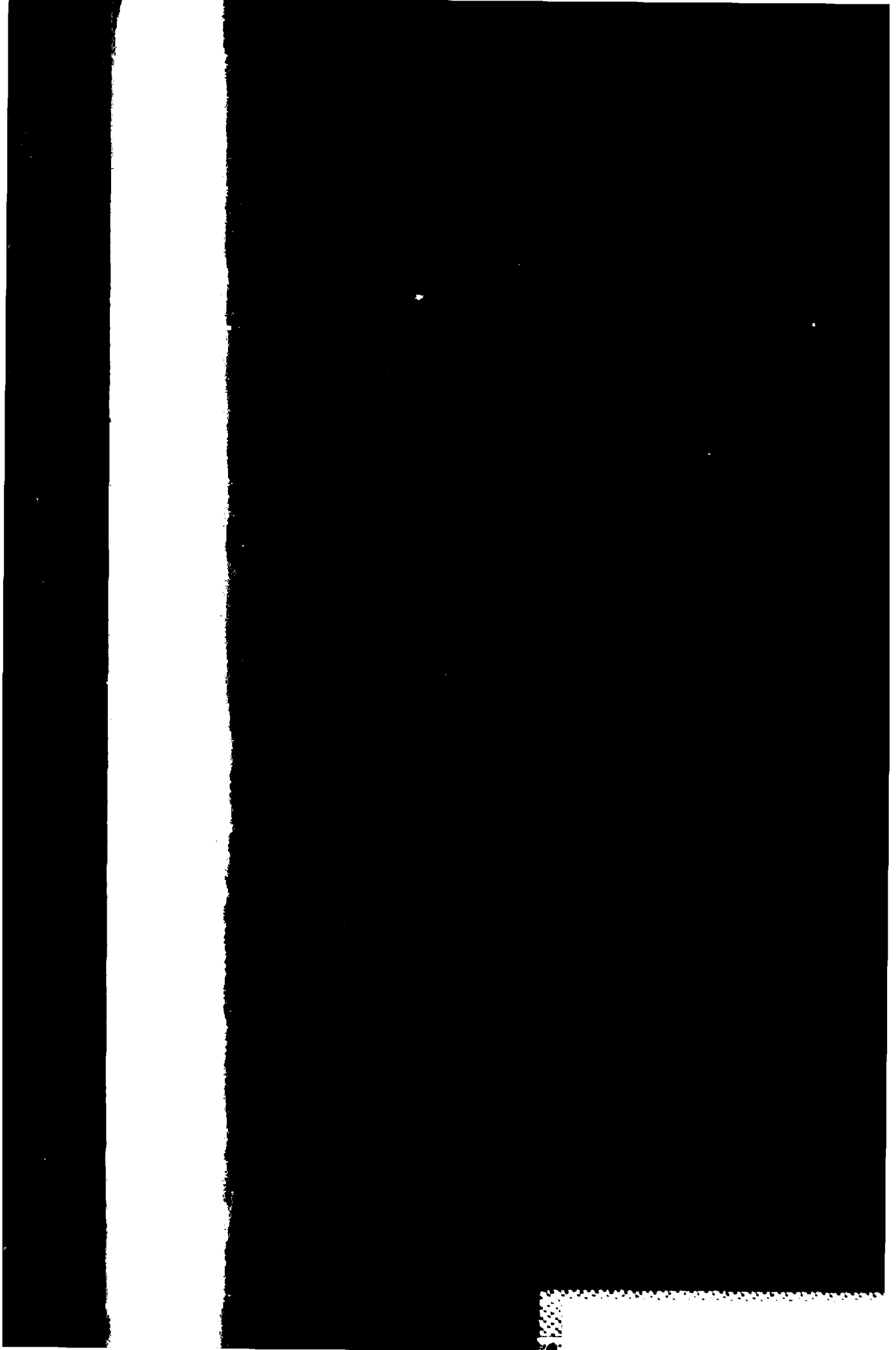
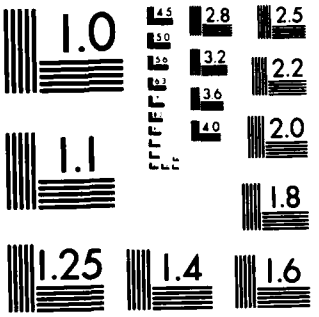
AD-A143 618 THE DOMAIN NAME SYSTEM(U) UNIVERSITY OF SOUTHERN 1/1
CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST
P MOCKAPETRIS JUN 84 ISI/RS-84-133 MDA903-81-C-0335
UNCLASSIFIED F/G 9/2 NL





ISI Reprint Seri
ISI/RS-84-13
June 19

(10)

University
of Southern
California

Paul Mockapetris

The Domain Name Syste

Reprinted from the *Proceedings of the IF*
6.5 Working Conference held in Nottingha
England, 1-4 May 1984.

INFORMATION
SCIENCES
INSTITUTE

213/822-1
4676 Admiralty Way/Marina del Rey/California 90292-6

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ISI/RS-84-133	2. GOVT ACCESSION NO. AD-A143618	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Domain Name System		5. TYPE OF REPORT & PERIOD COVERED Research Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Paul Mockapetris		8. CONTRACT OR GRANT NUMBER(s) MDA903 81 C 0335
9. PERFORMING ORGANIZATION NAME AND ADDRESS USC/Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE June 1984 13. NUMBER OF PAGES 18
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document is approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

.....

18. SUPPLEMENTARY NOTES

Reprinted from *Proceedings of the IFIP 6.5 Working Conference* held in Nottingham, England, 1-4 May 1984.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

binding, distributed database, names, naming, name server, protocols

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The domain name system is a protocol and a set of servers which provides a uniform method for associating the names of resources (e.g., mailboxes, host names) to information about the resources (e.g. mail server addresses, network addresses). The name database is distributed among multiple name servers scattered through one or more internets. The protocol provides tools for controlling both the distribution of the database and the responsiblity for maintenance of the distributed pieces of the database.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ISI Reprint Series
ISI/RS-84-133
June 1984

University of Southern California

Paul Mockapetris

The Domain Name System

Reprinted from the *Proceedings of the IFIP 6.5 Working Conference* **held in Nottingham, England, 1-4 May 1984.**

INFORMATION SCIENCES INSTITUTE

213/822-151
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

ISI Reprint Series

This report is one in a series of reprints of articles and papers written by ISI research staff and published in professional journals and conference proceedings. For a complete list of ISI reports, write to

Document Distribution
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
USA

Table of Contents

OVERVIEW

The problem

Most contemporary problems with naming in computer networks result from two trends: the first is the rapid expansion in the number of users, hosts, networks, and other resources to be named, and the second is the connection of more and more systems with different data formats and characteristics. The resulting problems include:

- A large number of names as well as a large rate of growth in the number of names. Continual gradual growth occurs as users and hosts are added; large jumps can occur when networks or internets are connected. The size of this problem varies with the resource being named; for example, there is usually a difference of orders of magnitude between the number of hosts and the number of mailboxes on those hosts.

- The need to distribute responsibility for naming. Organizations usually want to have authority for assigning names on their own hosts, networks, etc.

- The need to accommodate varying classes of information associated with a name. For example, two different networks might have different formats for host addresses, yet have identical formats for mailboxes. Thus the information associated with a name may depend on what the named object represents and may also vary according to the network or internet in which the object resides. In general, we can hope for consistency in some resources (for example, mailboxes), but we expect inconsistency in other types of resources (for example, host addresses). The domain system enforces consistency for the formats of names, but allows variety in the uses of a name and the information associated with a name.

- Variety in the capabilities of the hosts and networks that use names. Solutions that are appropriate for highly connected networks (e.g., the ARPANET) may be inappropriate for networks using once-a-day phone calls; similarly, solutions appropriate for large timeshared hosts may be inappropriate for personal computers.

The ARPA Internet illustrates several of these problems; it is a large system and is likely to grow much larger. Currently hosts in the ARPA Internet are registered with the Network Information Center (NIC) and listed in a global table [1]. The size of this table, and especially the frequency of updates to the table, are near the limit of manageability.

Mailbox names in the ARPA Internet are not distributed according to any network-wide scheme, although many organizations distribute mailbox lists for the organization to all hosts serving the organization. Inconsistency in the semantics of mailbox descriptions is a continuing source of problems; most of these problems arise from various ad hoc encodings of routes in the mailbox name.

Characteristics of the proposed solution

Several of our basic design assumptions for the domain name system evolved from the concerns outlined above:

- The database must be distributed. The size and update rate of the database prohibit centralization. By distributing the database, we may also improve the performance and availability of the system.

- From the user's point of view, the database should appear to be centralized. That is, measures to allow distribution, redundant copies, etc., should be transparent to the user.

- The costs of implementing such a facility dictate that it be generally useful and not restricted to a single application. We should be able to add new resource types indefinitely.

- The database must be hierarchical. Such an organization offers the opportunity for delegating responsibility for "subtrees" to separate organizations.

- The hierarchy must be extensible. The spread of local networks will place the same pressures on organizations to allow partitioning of responsibility as are now apparent in long-haul networks. We would like the capability to partition the database whenever such a partition makes database management more convenient.

We also imposed restrictions on the initial domain system:

- The domain system provides facilities for distributing the database and using redundant copies, but relies on local system administrators to configure the database so that it works properly. This responsibility includes change control, authentication, etc.

- Rather than including a mechanism for performing atomic updates, the domain system periodically distributes updated data. Thus redundant copies of parts of the database may be incorrect for short periods. The system administrator who creates a particular piece of data also assigns a refresh interval for that data. The update interval can be made arbitrarily short, or the update problem can be avoided by prohibiting copies of particular data.

These restrictions simplify the initial implementation task, but may be changed in the future.

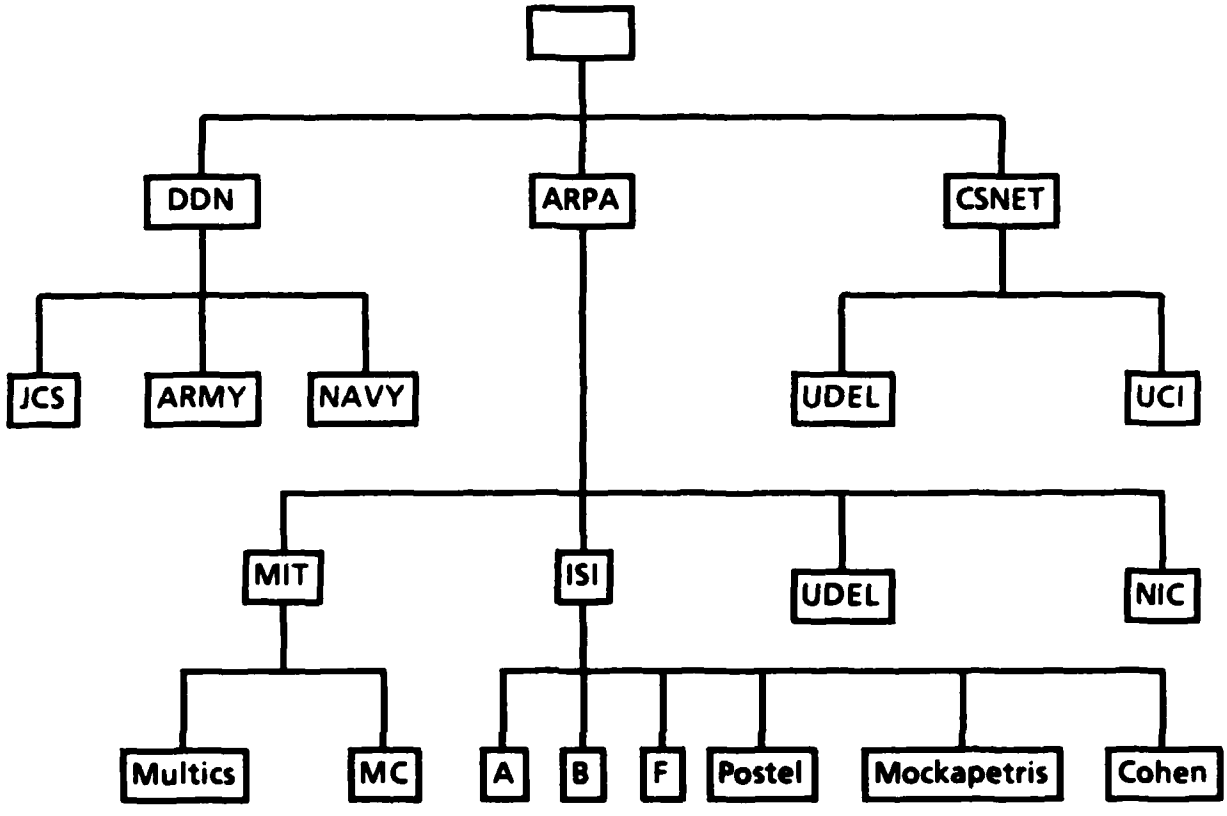
THE ABSTRACT DATABASE

Since the distribution of data is hidden from the typical user by the programs used to access the domain system's distributed database, the user is mainly concerned with the structure and contents of the abstract database (i.e., the database which would result if all of the distributed data were collected in a single database).

The domain name space

The domain system uses names that are hierarchical; conceptually, the name space is a tree with labels on the nodes of the tree. A node's domain name is the list of labels associated with the nodes on the path from the node to the root of the tree. By convention, we list the labels from left to right corresponding to the most specific node to the least specific (the root). With the exception of the root node, which has a null label, labels are not required to be unique. The root's unique label is a convenience which allows programs to easily recognize the end of a name.

Although the domain system does not require the name space's structure to correspond to any other structure, by convention we structure the majority of the domain name space to roughly correspond to the "nesting" of organizations that use the domain system, followed by other levels that correspond to the nesting of mailboxes, hosts, etc. within organizations. An example domain name space follows.



The tree is anchored at the root and is divided into three domains, DDN, ARPA, and CSNET, corresponding to the DDN network, the ARPA Internet, and the Computer Science Network. (The term domain is used to refer to any subtree of the abstract tree.) In the ARPA domain, the four subdomains MIT, ISI, UDEL, and NIC, are organizations that we have chosen to place in the ARPA domain. Each of these is its own domain for creating mailbox names, host names, etc.; the example diagram does not show this substructure, with the exception of a few example entries under ISI.ARPA.

The ISI domain shows a substructure for three hosts (A, B, and F) and three mailboxes (Postel, Mockapetris, and Cohen). Note that the name space itself does not imply this binding in any way; the correspondence is created using resource records.

Conventions

Internally, the domain system maintains names using a binary structure. However, many applications need methods for representing domain names as printable text. The default method is to list the labels from most specific to least specific, using dots to separate the labels. Since all domain names end in the root label, the root label and its preceding dot are omitted. A different method is used for mailboxes. A mailbox specification of the form local-part@global-part is mapped to a domain name whose most specific label is "local-part" and whose remaining labels are taken from "global-part", with dots representing label divisions. For example, the mailbox Mockapetris@ISI.ARPA maps to domain name Mockapetris.ISI.ARPA.

The domain system uses case-insensitive matching rules for comparing domain names, but retains the case of all data in the system. For example, if the system administrator defines a mailbox for Mockapetris@ISI.ARPA, that data will match queries for mockapetris@isi.arpa as well as queries for MOCKAPETRIS@ISI.ARPA; however, the returned answer will always be Mockapetris@ISI.ARPA.

Resources

In order to create a correspondence between a resource and its name, we associate resource records (RRs) with nodes. For example, to make the domain name Mockapetris.ISI.ARPA correspond to a mailbox, we store a mailbox resource record at that node; to make A.ISI.ARPA a host name, we store a host name resource record at the A.ISI.ARPA node.

We can associate as many RRs as we desire to a particular name. The most frequent use of this facility is to associate multiple host addresses to a single host or multiple mailbox names to a mailing list. However, we could also use a particular name to refer to both a host address and a mailbox name. Such multiple use is not forbidden by the domain system, but rather is avoided to spare users unnecessary confusion.

Each RR contains several standard fields, including a type and a class, as well as a variable-length resource data (RDATA) field, which contains type- and class-specific information describing the resource.

Type values are drawn from a set of well-known codes, and they refer to abstract resource types and include such types as "mailbox", "host address", and "mailing list".

Class values are drawn from a set of well-known codes and specify the system used to represent the data in the resource record. Class values usually identify an internet. For example, the ARPA Internet is one class, and RRs with type=A (host address) and class=IN (ARPA Internet) use 32-bit addresses, while another internet might use 10-digit phone numbers. Note that the class field does not represent protocol families per se; separate classes could be used for two private copies of a particular protocol, and there are cases such as the CSNET class, which uses ARPA Internet addresses as well as phone numbers and X.25 addresses in the RDATA field of its host address RRs.

Class definitions are orthogonal to domain structure. Thus, although all resources of a particular class may be organized into a specific domain, this type of organization is not required. For example, the CSNET domain and the CSNET class may happen to be related, but they are not constrained to be so.

In addition to defining data format, class information is used to guide the search process associated with queries. For example, a requestor looking for mailbox information might constrain the search to resource data of a class compatible with the requestor's machine; alternatively, the requestor might collect all of the resource information for the mailbox, regardless of class, and then decide which information to use.

The following is a partial listing of the RRs in the ISI section of the sample domain space:

Owner	Type	Class	RDATA
A.ISI.ARPA	A	IN	10.1.0.32
B.ISI.ARPA	A	IN	10.3.0.52
F.ISI.ARPA	A	IN	10.2.0.52
Postel.ISI.ARPA	MB	IN	F.ISI.ARPA
Mockapetris.ISI.ARPA	MB	IN	F.ISI.ARPA
Cohen.ISI.ARPA	MB	IN	B.ISI.ARPA

The RRs are attached to particular nodes; the node that "owns" a particular RR is shown in the owner column. The next two columns show the type and class of the RR; the two types shown here are host address (A) and mailbox (MB), the only class shown is ARPA Internet (IN). Host address RRs include the 4-octet ARPA Internet host address; this 32-bit quantity is shown here using the ARPA standard method of octets separated by dots (this is NOT a domain name). The RDATA section for mailbox records contains domain names, and points to a mail server for the particular mailbox. Thus the mailbox Postel@ISI.ARPA is bound to the mail server on F.ISI.ARPA, while Cohen@ISI.ARPA is bound to host B.ISI.ARPA.

QUERIES

From the user's point of view, the domain database is accessed though three kinds of query: simple, completion, and inverse. The user interface to the query mechanism is typically through operating system calls, and hence depends on local conventions for details of the call, but the general form of the query follows one of these three kinds.

Simple queries

In a simple query the user presents a query which contains the domain name, the type, and the class of a resource. The system returns either the corresponding RR or an indication that the RR does not exist, or possibly a transient error indicating that the appropriate database cannot be accessed.

For example, the user could ask for the resource record with domain name=F.ISI.ARPA, type=A (host address), and class=IN (ARPA Internet). This would bind the host name F.ISI.ARPA to its address. A similar query for the domain name Mockapetris.ISI.ARPA would return a non-existent RR error, indicating that the appropriate type of RR was not found, but a type=MB (mailbox) query would return the Mockapetris mailbox RR.

Completion queries

Completion queries allow a user to identify a resource using a partial name specification. This feature can be used to create shorthand notation for local resources, or a completion facility similar to that of the TOPS-20 operating system. The arguments for a completion query include the partial domain name, a type and class, and a target domain name. The type and class specify the eligible resources; in addition, the answer must be contained in the domain specified by the target domain name.

For example, a user at ISI who wishes to send mail to Mockapetris.ISI.ARPA might use a mail program which used completion queries. When the user asked to send mail to "Moc", the mail program might create a completion query with a partial string of "Moc", type=MB, class=IN, and a target domain name of ISI.ARPA. This query would be interpreted as a request for a mailbox which begins with the text "Moc" and

resides in the ISI.ARPA domain. Using our example, the program would receive the mailbox RR for Mockapetris.ISI.ARPA.

Such a query may well be ambiguous, especially if the target domain does not greatly restrict the search. To deal with this, and to allow user programs a chance to resolve the ambiguity if they choose, completion queries also specify whether the requestor wants all possible matches, or if the domain system is to resolve any ambiguities using simple rules based on the number of labels in the possible answers.

Implementation of this feature is optional, and hence it may be available for certain domains and not for others.

Inverse queries

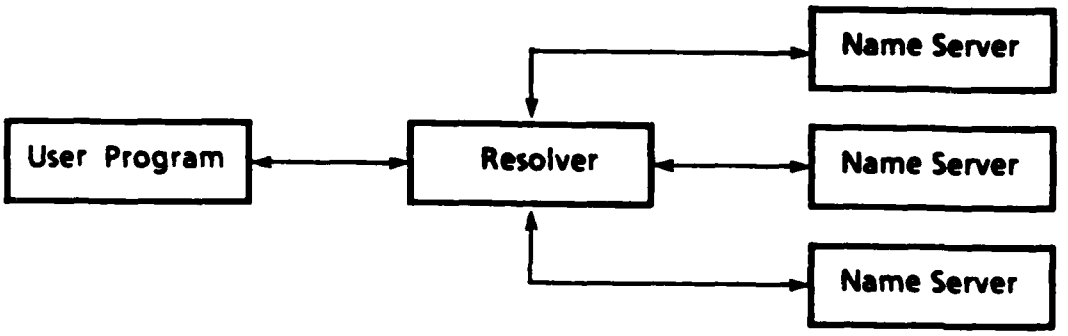
Inverse queries allow a user to perform the inverse of simple query mappings, i.e., given a resource record, an inverse query returns the domain name or names that possess such a resource record.

Uniqueness is rarely a problem for most applications of this type of query. For example, the most frequent use is mapping a host address to a host name. In situations where ambiguity exists, the response to the query contains all domain names found.

Implementation of this feature is optional, and hence it may be available for certain domains and not for others.

DISTRIBUTION OF THE DATABASE

While the distributed nature of the domain system is hidden from the user program, a user query may cause activity in several processes, both on the local host and in foreign hosts. The possible interprocess communication is shown below:



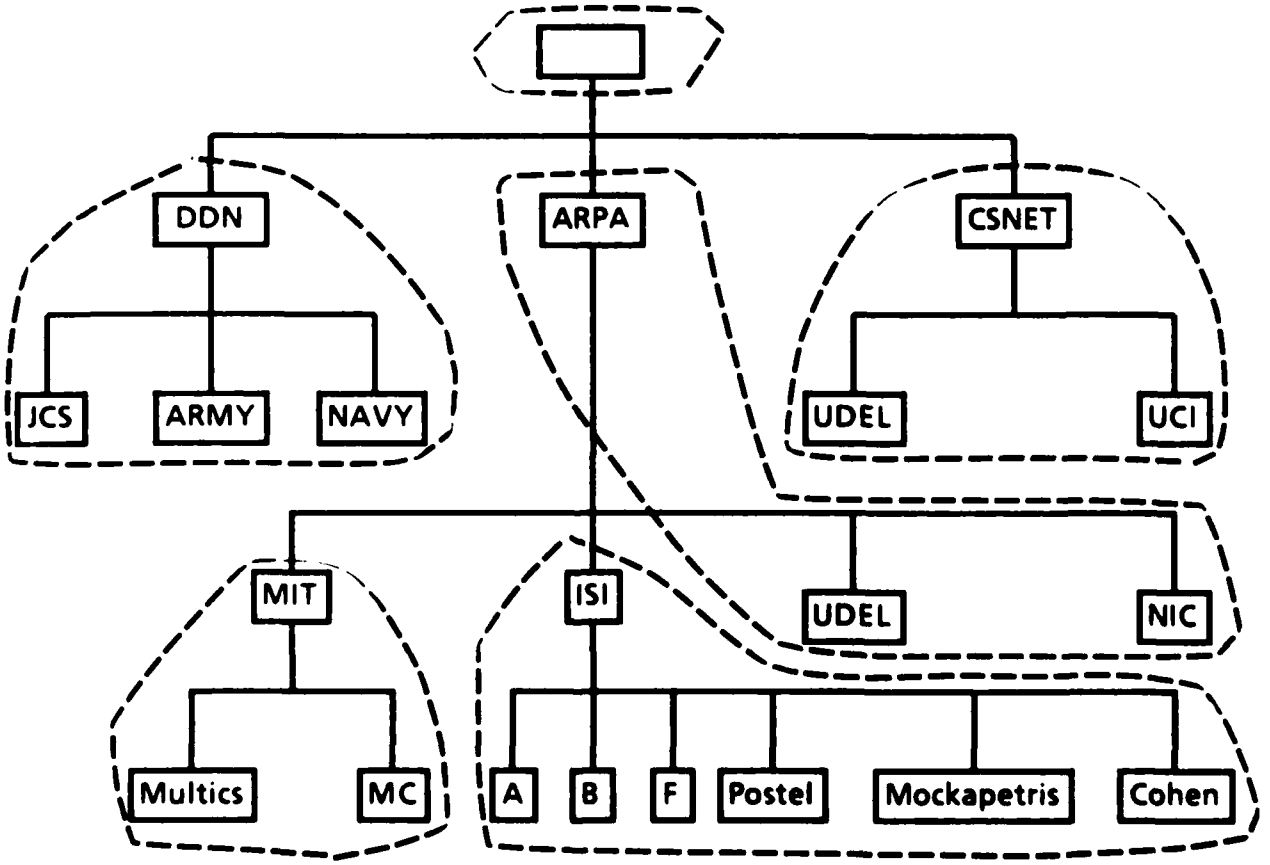
The user program issues a query via some sort of call to a local program called a resolver. This query is expressed according to local conventions, and the resolver is usually part of the host operating system. The resolver answers the query using information it acquires from one or more name servers. The transaction between the resolver and the name server is expressed using the domain protocol.

Name servers are repositories for sections of the domain database. A given name server will typically only have information for a small part of the abstract database. Name servers internally break the abstract database into sections called zones. While name servers can be configured to treat each domain name as a separate zone, system administrators will usually configure name servers to group organizationally related domain names in a single zone. A particular zone may be replicated at several name servers to provide higher availability.

This section discusses the methods that are used to create the zones and the operations performed by name servers and resolvers to process user queries. Note that the division of responsibilities is often conceptual rather than actual; a host that possesses both a name server and a resolver will often mingle the functions to improve performance.

Zones

The abstract database is partitioned into zones by inserting zone boundaries on selected arcs of the abstract tree. A zone begins at the point where it is divided from its parent zone and ends at leaves in the abstract tree or at arcs where new children zones begin. Our example tree might be divided as follows:



Here the root has delegated authority to three zones: DDN, ARPA, and CSNET. Th
delegated authority to the MIT and ISI zones.

In addition to marking the delegation of naming authority, the zone boundaries also repres
of the domain database. Thus the term *zone* is also used to refer to the complete data for th
RRs for all nodes within a zone. An organization that has a zone is responsible for mai
data on name servers that make it available to hosts both within and outside of the zone.

The domain name of a name server and the domain names of the zones the name server po
related. An organization can either provide name servers on hosts in its own zone, or
service from other organizations. A name server can support multiple zones, as lo
remembers the boundaries. For example, the zones described in the previous example
name servers as follows:

Zone	Name servers
" " (root)	NIC.ARPA
DDN	JCS.DDN, B.ISI.ARPA
ARPA	NIC.ARPA, F.ISI.ARPA
MIT.ARPA	Multics.MIT.ARPA
ISI.ARPA	F.ISI.ARPA, B.ISI.ARPA
CSNET	UDEL.CSNET

Since zone boundaries provide the basis of redundant copy distribution, an organizati
partition its authority simply to separate the database and not to delegate authority. For e
zone might be partitioned into ARMY.DDN and NAVY.DDN subzones simply to spli
between two name servers, rather than to actually split administrative control of the data.

Similarly, a name server can arrange to receive a zone copy simply to improve performa
For example, the DDN zone might be supported on B.ISI.ARPA simply to provide direct
zone's data for users on B.ISI.ARPA.

The data that makes up a zone consists of three kinds of RRs:

1. A comprehensive set of RRs to describe all of the resources attached to nodes in the z

2. Special RRs, associated with the top node of a zone, which enumerate name serve
copies of the zone and describe the characteristics of the zone. The characteristics are
a single Start of Authority (SOA) RR; this record identifies a single master copy of th
which redundant copies receive updates. All name servers with copies of the zone a
with Name Server (NS) RRs.

3. Special RRs which mark delegation of authority to subzones. These RRs are also N
point to name servers which support the appropriate subzone.

For example, the data for the ARPA zone would include the following:

Owner	RR		
ARPA	SOA	IN	NIC.ARPA
ARPA	NS	IN	NIC.ARPA
ARPA	NS	IN	F.ISI.ARPA
ISI.ARPA	NS	IN	B.ISI.ARPA
ISI.ARPA	NS	IN	A.ISI.ARPA
MIT.ARPA	NS	IN	Multics.MIT.ARPA

The SOA and NS RRs attached to node ARPA name NIC.ARPA as the holder of the master copy of the zone, while the two NS records at ARPA point to name servers for the ARPA zone as being on hosts F.ISI.ARPA and NIC.ARPA. The NS RRs attached to ISI.ARPA point to hosts B.ISI.ARPA and F.ISI.ARPA as holders of zones for ISI.ARPA.

The special RRs that delimit a zone also provide connectivity for searches throughout the name space. If a resolver knows the address of any root server, it can find any node in the abstract tree by retrieving the appropriate NS records from the root name server and iterating "down" to the name server which has the desired information. To guarantee connectivity, we can also put pointers to root servers in all name servers; if this is done, the resolver needs to know how to reach only one name server to be able to eventually access all of the domain data.

Resolvers

Resolvers are programs that process user queries and chain through name servers to find the specified data. A user program issues a query, which is passed to the resolver via an OS call or some other mechanism. The resolver then queries local and foreign name servers to acquire the specified information. Since a particular query may involve several network transmissions to a series of foreign name servers, resolver actions take an indeterminate amount of time.

Name servers

The action taken by a name server to process a query depends on the kind of query and the composition of the zones possessed by the name server. For simple queries, the name server checks to see if the domain name in the query is contained within any of the name server's zones. If it is, the name server returns either the requested RRs or an indication that the requested resource does not exist. Since all copies of a zone are assumed to be equivalent, such an answer is marked as being authoritative, so that the resolver that sent the query will know there is no point in attempting further queries at other name servers. If the query does not refer to a node in one of the name server's zones, the name server searches its database for NS RRs corresponding to zones that are parents of the query domain name. Since all name servers include pointers to a name server for the root, this search always succeeds. The name server then returns the NS RRs for zones "closest" to the domain name in the query. This type of response is called a referral.

For completion queries, the name server examines the target name for the query and decides whether it refers to a node in one of the name server's zones. If so, the name server performs the completion search and returns the results. If not, it returns a referral.

For inverse queries, the name server has no indication of the domain name that is the answer to the query. Hence inverse queries are processed using whatever zones are available, and the answer is returned.

A central feature of this functionality is that name servers do not need to be concerned with the location of non-local data; they simply answer queries based on their local zones. This allows for quite simple name server implementations.

DATABASE MAINTENANCE

Two types of database maintenance activity occur in the processes that implement the domain system: refreshing of zone copies, which occurs on a planned and regular basis, and caching, or demand-driven copying of data for one resolver request on the assumption that the data may be useful for subsequent queries. Although the mechanisms are quite different, both rely on the notion that the creator of a particular RR should be able to set a time-to-live (TTL) for the RR, describing the maximum time that a copy of the RR can be assumed to be correct before the creator should be consulted. This TTL is effectively the length of time before a change to the database is guaranteed to be effective throughout the system. Of course, the time intervals must also take into account the cost of too short an interval.

Refreshing discipline

The master copy of the data that makes up a particular zone is assumed to reside on a single host. That host's name server creates a zone by reading a file. Other name servers use the domain protocol to get a copy of the zone.

All name servers which have copies of a zone are responsible for checking periodically to see if the zone has been updated. The name server with the master file does this by checking to see if the file has been updated; name servers which acquired a copy of the zone can check for updates by requesting the serial number of the zone from the name server that has the master copy.

The identity of the name server with the master copy, the time intervals for checking for updates, etc., are contained as part of the zone data. In general, the timeouts are set up so that the check for update (refresh interval) occurs frequently and the copy is allowed to persist (expiration interval) for a very long time when the master copy is unavailable.

For zones, a single TTL value covers all of the data in the zone. If this were not the case, false nonexistent RR errors could result.

Caching

Caching is performed as the result of resolver activities. The premise is that, if a resolver goes to the trouble of acquiring a particular RR, it should cache it for use in answering future queries. While this feature is optional in resolvers, and resolvers may use different cache sizes, etc., a resolver that caches RRs must also manage a TTL for each RR it caches.

The TTL for a particular RR is derived from a zone TTL when a name server supplies an RR to the resolver. Since the resolver does not have zone information, it times each RR out separately. The derived TTL will typically be much smaller than the TTL for the zone as a whole. The reason for this is that, while zone copies are tested periodically for updates, cache copies are not similarly protected.

Adequate performance probably dictates that resolvers should cache at least the referral RRs they acquire, to avoid repetitive chaining through name servers. Zone designers assist this process by assigning long TTLs to NS RRs whenever possible.

STATUS AND DIRECTION OF FUTURE WORK

Status

The domain system has been issued as a set of draft RFCs [11,12] and is currently being implemented according to a schedule described in [13]. After the experimental period, a final specification will be issued.

The experimental period is designed to allow operational experience with several features which are optional in the present design and to further refine the specification. This section discusses several of the optional features under study.

Connecting internets

The greatest challenge and least understood problem for the domain system is sharing information between different internets. The current system recognizes that such interconnection will require translation at some level, if only to forward queries over different transport protocols. The class notion is our mechanism for controlling routing to the appropriate translation process.

At the simplest level, each internet replicates its information for each possible class of requestor. For example, the ARPA internet information for mailboxes could be replicated in both the IN and CS classes, with separate zones on name servers for the respective classes.

In practice, we expect that each zone will be completely described in a single class corresponding to the system in use, but that in other classes the zone will consist of pointers to appropriate gateways. For example, while hosts in the IN class can acquire mailbox information for the ARPA domain, hosts in the CS class would receive a single forwarding RR regardless of the mailbox they name. This forwarding record directs the mail to a mail gateway. A slightly more powerful use of class directs the search to a name server which can perform translation between the RRs of one class and the RRs of another.

The most general system would create a universal class for data which could be represented in a class-insensitive way. In general, the only universally known data types would be domain names themselves, so this strategy implies a level of indirection in the binding. For example, a universal mapping might bind mailboxes to the domain names of mail agents, and then a class-sensitive binding could direct the mail sender to either the mail agent or a mail gateway capable of reaching the agent.

While these strategies all seem possible, the usual source of problems is interference with freedom at the lowest levels. For example, translation at the name server level implies variability in response time, and can make datagram name service difficult. Similarly, there are networks that will be connected to the domain system on an infrequent basis.

Our intuition is that the domain system will evolve to use a mixture of these strategies at the discretion of the clients of the domain system.

Update management

Although the refresh mechanism provides an adequate mechanism for distributing information that does not change with use, it is inadequate for applications that are intended to allocate names dynamically. Such applications might include dynamic creation of new names, and name binding that is a function of dynamic conditions (e.g., what is the name of an idle print server in the ISI domains). Such services are under study, but will not be added until the informational services described herein are stable.

ACKNOWLEDGMENTS

The author wishes to thank Jon Postel and Paul Kirton for their contributions to the domain name system, as well as Sheila Coyazo and Ruth Brungardt for their contributions in improving this paper.

REFERENCES

1. E. Feinler, K. Harrenstien, Z. Su, and V. White, "DOD Internet Host Table Specification", RFC 810, Network Information Center, SRI International, March 1982.

2. K. Harrenstien and V. White, "NICNAME/WHOIS", RFC 812, Network Information Center, SRI International, March 1982.

3. M. Solomon, L. Landweber, and D. Neuhengen, "The CSNET Name Server", Computer Networks, vol. 6, no. 3, July 1982.

4. K. Harrenstien, "NAME/FINGER", RFC 742, Network Information Center, SRI International, December 1977.

5. J. Postel, "Internet Name Server", IEN 116, USC/Information Sciences Institute, August 1979.

6. K. Harrenstien, V. White, and E. Feinler, "Hostnames Server", RFC 811, Network Information Center, SRI International, March 1982.

7. J. Postel, "Transmission Control Protocol", RFC 793, USC/Information Sciences Institute, September 1981.

8. J. Postel, "User Datagram Protocol", RFC 768, USC/Information Sciences Institute, August 1980.

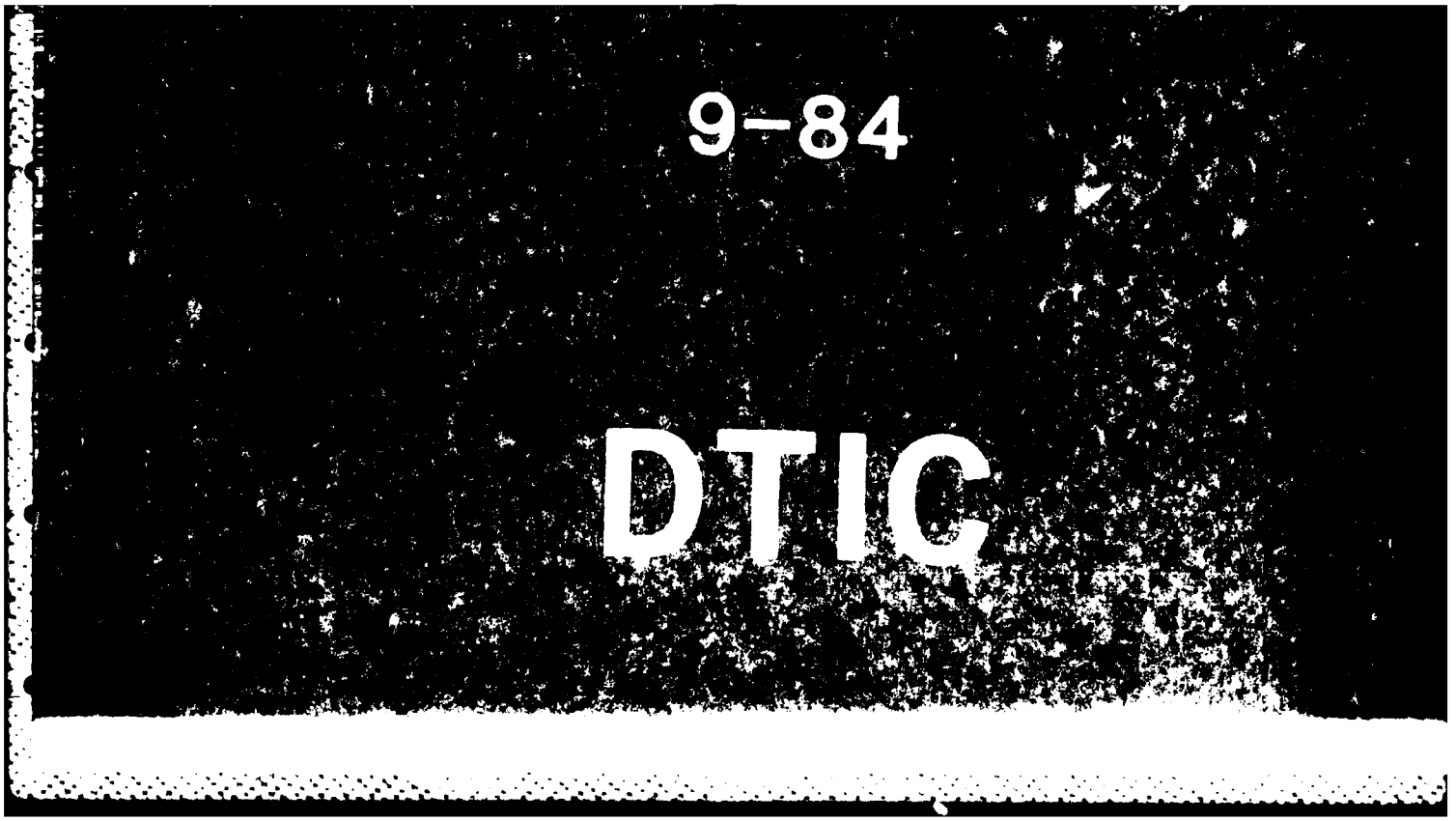
9. J. Postel, "Simple Mail Transfer Protocol", RFC 821, USC/Information Sciences Institute, August 1980.

10. J. Reynolds and J. Postel, "Assigned Numbers", RFC 870, USC/Information Sciences Institute, October 1983.

11. P. Mockapetris, "Domain Names - Concepts and Facilities", RFC 882, USC/Information Sciences Institute, November 1983.

12. P. Mockapetris, "Domain Names - Implementation and Specification", RFC 883, USC/Information Sciences Institute, November 1983.

13. J. Postel, "Domain Name System Implementation Schedule", RFC 897, USC/Information Sciences Institute, February 1984.


9-84
DTIC

[redacted]

ISI.ARPA N
ISI.ARPA N
MIT.ARPA N