UNIVERSITY OF CALIFORNIA

SANTA CRUZ

Census : Collecting Host Information on a Wide Area Network

A thesis submitted in partial satisfaction
of the requirements for the degree of

BACHELOR OF ARTS

in

COMPUTER AND INFORMATION SCIENCE

by

Nitin K. Ganatra

June 1992

The thesis of Nitin K. Ganatra is
approved:

Prof. Darrell D. E. Long

Census : Collecting Host Information on a Wide Area Network

Nitin K. Ganatra

ABSTRACT

The exponential growth of the Internet presents new problems in recording global network information. The Domain Name System provided a way to cope with the enormous growth rates by distributing network information among all domains. This distribution accomplished its goal of eliminating the need for a central database, but at the same time eliminated the source of centralized network information. Such information is useful in applications dealing with resource discovery on the Internet and in studies of the network topography. Census is a program to traverse the Internet domain tree and collect information by recursively querying name servers for host information and other sub-domains. During the development of the Census program (about 6 months), the host population of the Internet has grown in size by 30 percent, and has showed no sign of abating.

Chapter 1

Introduction

Since the mid-1980's, an explosion of growth on the Internet has taken place that strained earlier methods of recording network information. Originally, all hosts on the Internet were registered with the Network Information Center (NIC). Data on each host were kept in a central database for use by other hosts in need of the information. Local copies of this database were kept on Internet hosts (on UNIX systems in /etc/hosts) as a reference for host names, addresses, and other information. Changes to the structure of the Internet were recorded in the central host table through a request to the NIC. If changes were made and a host was still using an old database, there was a chance that a host name-to-address mapping would fail. Consequently, it was up to hosts to make sure they had the most recent table by periodically deleting their host table and copying the latest file from the NIC. The size of the Internet dictated how frequently changes to this central database occurred. As more hosts and domains started following Internet protocols, changes to the central host table became more frequent.

As the size of the Internet started to grow almost exponentially [4], the burden of holding and accessing all of the Internet's centralized information became much less feasible. Scanning the list for host information took linear time, which was acceptable when the Internet's host population was smaller. The time to retrieve network information became substantial. As the Internet's size increased, the database look-up time increased as well, thereby slowing all network services. Distribution of the file became much more cumbersome as network, disk, and CPU resources were being used simply to access and update this host file. Furthermore, failure of the NIC was a growing concern. Copies of the central database were kept at other network information centers, but what if a crash occurred during the updating process? Any other NICs, and the entire Internet, would have to rely on older databases for network information until changes were once again recorded and distributed.

The Domain Name System (DNS) was devised as a better way to deal with large amounts of network information [5]. The DNS incorporated a hierarchical name structure and provided guidelines for host databases at individual domains. Thus, the DNS lifted the burden from the NIC, and made it possible to deal with enormous growth rates. It was no longer necessary to rely on one centralized database. Information about any part of the network could be obtained from a traversal of the domain name tree.

Ironically, the biggest advantage of the DNS is also one of its disadvantages—a lack of centralized data about the Internet. The Census program fills the need for centralized information by producing a reasonably accurate snapshot of the Internet. This information is invaluable for studies pertaining to the characteristics and topography of the Internet, and even studies of the DNS protocol's use.

Census works by walking over the entire Internet domain name tree, collecting hosts and domains that still need to be traversed. Census yields a *very* large[1] database on the Internet and its population. Census is written in portable C, making it possible to run the program on different computer architectures.

[1]The latest run in June of 1992 produced a final database over 60 megabytes in size.

This report concentrates on the workings of Census, some of the problems associated with writing such a piece of software, and the results from the program. The remainder of the paper is organized as follows. Chapter 2 gives background information about the Internet and the Domain Name System. Chapter 3 discusses related work that has been done in the field of resource discovery on the Internet. Finally, chapter 4 gives an overview of the Census program, including some problems encountered during the development of the program and some results from data gathered by the Census program.

Chapter 2

The Internet

The Internet is a network consisting of local area and wide area networks. These networks all follow certain regulations and standards (TCP/IP) that allow them to interoperate in a transparent and consistent way. This allows for a large base of shared information.

The TCP/IP suite consists of over 100 protocols used to connect and organize computers on a network [7]. Examples of these protocols include the File Transfer Protocol (FTP), telnet, and the Simple Mail Transfer Protocol (SMTP). Originally, the TCP/IP suite was designed for use by the Defense Advanced Research Projects Agency (DARPA) for its own network research. To be a part of the Internet, a network must use the TCP/IP suite, though not all networks using TCP/IP are connected to the Internet. As an example, Sun.Com has a gateway between its network and the rest of the Internet that makes queries about hosts in the network impossible. Such an arrangement allows greater security and autonomy for hosts at Sun Microsystems, but brings up the question, are these hosts part of the Internet?

2.1 Domain Name System

In the mid-80's, the DNS introduced a hierarchical naming structure that allowed all the Internet's information to be distributed around the network. The DNS protocol lifted the load from the NIC, allowing growth of the Internet to continue unimpeded by a dependency on the NIC and its resources [5]. All Internet services depend on the DNS to communicate with other hosts, either within a domain or with hosts in other domains.

The main theory behind the DNS is that every host doesn't *need* to know about every other host in the network—it just has to know how to find needed information. For example, in order for a domain in Australia to find information about another domain in Australia, the host look-up should only be concerned with other domains on the continent. An information look-up should not have to sort through hosts from a foreign domain unless that is where the desired information resides. Domain and host changes are therefore only recorded locally, and any information needed on a particular domain can be obtained from a host local to the domain. With this in mind, the DNS has been implemented in an attempt to operate transparently using a minimum of network resources. Design elements [5] of the Domain Name System include *domain name space*, *resource records*, *name servers*, and *resolvers*.

2.1.1 Domain Name Space

The Domain Name System imposes a tree structure on the network. This structure treats leaf nodes as hosts and all other nodes in the tree as a name in a host's domain. At the root of the tree is the domain dot (.) which includes all top-level domains as its children. Each of these top-level domains can then have many different children under it. Each hierarchy level in the host or domain name is separated by a dot. For example, willow.UCSC.EDU means the host willow is under the UCSC domain, which is under the EDU domain. Names within the domain name space are case-insensitive, although services should treat queries in a case-sensitive manner. Treating queries as such will keep future changes to a minimum if the DNS is modified to be case-sensitive [5].

2.1.2 Name Servers

Name servers are programs running on machines that hold information about some domain's structure. They are responsible for answering queries about these domains. Duplicate name servers are kept in the interest of fault tolerance, because of their importance in providing host name-to-address mapping necessary for Internet services. Often domains have duplicate name servers in geographically remote locations[1] in case of some large-scale network failure or physical disaster. Depending on the implementation of the name server, it may choose to cache frequently requested information to speed up information look-ups [6].

2.1.3 Resource Records

The information that each name server holds is stored in a *resource record.* The resource record holds more than simply a name-to-address mapping. Resource records also hold : the name of the authoritative server for its domain, start of zone authority (where a domain begins and another ends with respect to the domain tree), mail exchange information, and common aliases for hosts. In order to obtain

[1]For example, the California Institute of Technology's domain (caltech.edu) has a duplicate name server in Ohio.

all host information from each domain's name server, a zone transfer (AXFR) query is sent to the resolver. AXFR queries send the entire contents of a resource record over the network.

2.1.4 Resolvers

Resolvers sit between the name server and the client [5, 6]. They take in queries from clients, querying the name server for the appropriate information, and then send that information back to the client. If a particular query cannot be answered by a name server, the resolver will query the name server of the domain that is one level higher. This process continues until a matching domain is found, or until the root domain (.) is queried. If the root domain name server has no information about the domain in question, the client's domain query cannot be answered.

It is up to the resolver to determine if the information has been cached and is still residing in memory. If so, a name server look-up (which involves a more time-consuming disk access) is not necessary. The resolver can pull necessary information from the cache, thus saving time and CPU cycles on the name server. The advantages of cached information become apparent when considering how often similar information is requested from name servers of high-level domains. For instance, watson.IBM.COM is an authoritative name server for IBM. Most queries of this name server involve finding the name server of a commonly accessed sub-domain of IBM.COM. Once this name server is found, it is then queried about either some host or yet another sub-domain. If such a route is common in IBM.COM's daily network traffic, and if information caching is not performed, the disk accesses needed for each look-up would be very taxing on the name server. Usually retrieved information is cached for a period of 24 hours, at which time it is removed. This policy relies on the (generally correct) assumption that if information is needed once, it will probably be needed sometime in the near future.

2.2 Problems with the DNS

One problem with the DNS is that its specifications make no reference to error correction. Because the DNS can be implemented on many different machines, bugs that are unique to each implementation are possible.

The specifications for the workings of the DNS [5, 6] detail all aspects about every aspect of the DNS's proper implementation, with the exception of error correction. Consequently, the basic error correction algorithms are left to the programmer. This allows it to be implemented in different, sometimes erroneous, ways. Often, these error correction methods introduce loops into the querying process, leading to an unnecessarily high number of packets being sent over networks.[2] Due to the nature of distributed, replicated systems these bugs often go unnoticed [1]. This causes the name server and network load to be artificially high, and sometimes reduces the response time of a query. Until standards are set to deal with errors in a consistent manner, and until all DNSs are implemented accordingly, these problems will most likely continue.

[2]The number of packets sent through the backbone of the Internet is at least twenty times higher than absolutely necessary, according to one study by P. Danzig [1].

Chapter 3

Related Work

In the field of resource discovery on the Internet, many different studies have been done or are currently being developed. The primary work related to Census is the Zealot of Name Edification (ZONE) program written by Mark Lottor at SRI International [4], but other works also make use of the DNS to perform information retrieval, and could benefit from the use of Census.

3.1 Zealot of Name Edification

As the Internet community was making the transition to the DNS, a program called Zealot of Name Edification (ZONE) was written. ZONE produced a host table that non-DNS hosts could use while in the process of making the transition to DNS [4]. In the end, ZONE's purpose was altered, and it came to be used to study the growth characteristics of the Internet. The program is now run every three months at SRI International for the purpose of studying changes in the Internet's population. Unfortunately, the ZONE program is in need of a replacement. In Mark Lottor's own words [4]:

> The amount of data is quickly reaching the limits of the DEC-20 section address space, and the hardware's ability to survive gets slimmer each

day.

During the development of Census, many of Lottor's future design concepts were considered and indeed incorporated in its design. ZONE's basic algorithm to collect host information is recursive, and is almost identical to that of Census. The main difference between Census and ZONE is in the temporary storage of information—ZONE holds all of its host information in memory until the end of the data retrieval, where Census saves information to disk as it is found. The advantage of holding all data in memory is that multiple hosts can be eliminated as they are received. With Census, these duplicate host entries must be eliminated using other tools once the data collection is completed.

3.2 Archie

Archie is another example of a wide-area information gathering service that relies on the DNS. Archie was developed to make Internet services more accessible as they grow rapidly. It provides a large database of software and information that is available from anonymous FTP sites [2]. There are nine Archie information servers which provide access for the entire Internet community. In its current implementation, Archie maintains two databases; one holds the names of files that are available, and the other holds brief descriptions of a fraction of the files. From these two databases, Archie users can find out if certain software is available, and exactly where it is on the Internet. When compared to telnetting to each site and searching through directories by hand, Archie provides an invaluable service. In order to provide recent information, the Archie servers update information about a group of sites every 24 hours, and cycles through the list of sites about once a month [2]. Archie could benefit from the use of Census to update its list of hosts with publically accessible information, as could the following service.

3.3 Service-Level Reachability in the Global Internet

Mike Schwartz at the University of Colorado currently leads a study to test the degree of connectivity on the Internet. The study attempts to make domain connections bimonthly over a period of one year [7]. Connections are attempted with almost 13,000 domains to perform 13 different TCP services, including finger, telnet, rsh, and SMTP. If a connection to the specified service is made, it is recorded as a success, and the connection is closed [7]. From fluctuations in the successes and failures, different characteristics of the Internet can be studied. The resulting information would be useful in future plans to design distributed information exchanges, such as libraries, encyclopedias, and file systems.

These studies may benefit from the use of Census to gather hosts of the Internet. The study may be carried on with different domains discovered by Census in order to vary the parts of the Internet being tested. This would give a better idea of what domain services are offered at given times, making the study of the Internet more comprehensive. Census could also be run to obtain host information such as what kinds of computers are being tested, and correlations between the failure rates of domains and their implementations of the DNS.

Chapter 4

Census

Census is a tool to collect host names, addresses, and other information of a substantial portion of the Internet and store them in one central location. By following the DNS protocol, a list of the top level domain names can be transformed into a list of virtually all the domains on the Internet. From these domains a list of all the hosts and addresses can be composed.

The only thing Census requires is a list of domains in the domains.in file.[1] When Census starts running, a few files are created for its use:

- tryagain.out holds all domains that are to be tried again once domains.in is exhausted.
- temporary.out holds host names and addresses of the domain currently being queried.
- hosts.out contains the host names and addresses of the domains that have been successfully been queried.
- successful.out domain names that have successfully been queried. Used in crash recovery to find the last successfully queried domain.

[1]The domains.in file is not needed when Census is run with either the -all option, or the -domain option, since one is created automatically.

- lastdomain.out the last domain that was queried.

The main data structure in Census is a queue of domains upon which data collection is to be performed. Domains are taken from the queue (which is in the domains.in file) to be queried for host information and any sub-domains. All host information found is stored in the temporary.out file, and sub-domains are tested for uniqueness. This test consists of comparing the newly found domain against all other domains known by Census. If the domain is unique, it is added to the domain queue. If not, it is discarded. In theory, the DNS should be implemented in such a way that all domains would only be seen once. However, relying on this to be implemented correctly could lead to infinite loops in the discovery of new domains. Finally, in the event of a system crash, if domain uniqueness not checked, Census would record the same domains twice, leading to redundant host information.

Failed domain queries occur for a number of reasons, the most common being the improper implementation of the DNS or the AXFR query.[2] However, queries often fail for more mechanical reasons, such as network or gateway failure or name server crashes. For this reason, domain queries must be retried until no new information is received. Currently, Census retries the failed domain list three times. This has proven to be sufficient by the fact that the domain queue has the same domains as the tryagain.out file upon termination of the program, meaning that all failed domains have been tried numerous times and have failed. Census collects hosts and domains in a breadth-first manner. It starts with domains that are specified before the program is run and works its way down the queue to the most recently discovered domains.

[2]Some domains have chosen to intentionally disable certain services for security reasons, or as a measure to reduce the load on its name servers.

4.1 Crash Recovery

Census was designed to run for a period of several days. This makes the possibility of a reboot or operating system failure far greater than in a program that runs for a few minutes. The size of the Internet was also motivation to keep all data that had been collected before a crash, instead of starting the data collection fresh. If Census runs for two days and the system crashes, it's preferable to start running on the domain where the data collection was stopped. Obviously, a dependence on volatile memory had to be kept to a minimum. Whenever Census reads all pertinent information in a domain, it appends the completed data to hosts.out and stores the name of the domain in successful.out. Since this information is stored in the file system, it is protected from system failure more than volatile RAM. This disk-intensive design is slower, but is justified when compared to the option of having to restart the data collection at every crash.

4.2 Hanging Reads

Occasionally, when a *read* is set up from a socket that is connected to a remote name server, it will hang indefinitely. During these hangs, the program does nothing while waiting for data that will never be sent. Reasons for these hangs vary from a failure of the network somewhere between the local host and the remote server, to a failure of the remote server (sometimes a crash or a power failure). By constraining the period of time a socket can be connected to a remote server, Census avoids such problems. The select system call is used to set a timer on the socket. If data is received the timer is reset, and the socket continues to wait for more data. If, however, no data is read through the socket within the specified amount of time (two minutes), the following action is taken:

- Close the waiting socket.
- Throw out any information in temporary.out.
- Save the domain name in tryagain.out to try the domain later.

4.3 Future Improvements

With the use of the select system call, the performance of Census slowed slightly—a small price to pay considering the benefits of its use. It would be favorable, however, to gain back some speed. One way to do this would be to have Census collect information from several hosts in parallel. Currently, Census collects data sequentially, which takes a few days to complete. A parallel Census could read domains from one file, fork off processes to gather host information from several domains at once, and store their data in another file. The main concern (and it is quite a concern) with implementing a parallel Census lies in its use of a large fraction of network resources.

4.4 Results

In the development stage of Census, it's usefulness in determining the growth of the Internet has already materialized. Early runs of Census in the middle of March indicated that the Internet's host population hovered around 840,000, up from 727,000 reported in January [4].

In its latest runs, Census has been collecting approximately 940,000 unique hosts that are reachable on the network. This is approximately a 30 percent increase of the Internet's size in a period of four months. The program currently takes a

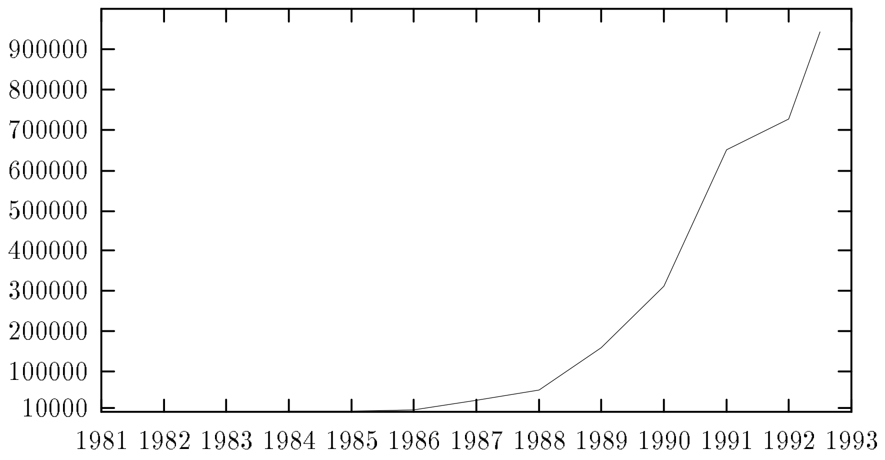

Figure 4.1: Growth of the Internet since 1982.

little under four days to complete the entire host collection, which is shorter than ZONE's collection period of over a week.

Figure 1 shows a graph of the growth of the Internet from 1982 to present, with the results of Census appended to results of the ZONE program. Figure 2 shows the breakdown of Internet hosts by CPU. Unfortunately, this estimation only involves hosts that have recorded the CPU type in the host information section of their resource records. Over 700,000 hosts have either chosen to not record this information, or have recorded it erroneously. Figure 3 gives a breakdown of the population of the Internet by the number of hosts in each top-level domain.

Machine Types on the Internet	
IBM PC	67,590
IBM RS/6000	1,145
IBM Other	14,007
Macintosh	67,891
Sun	53,024
DEC	19,213
HP	11,535
NeXT	3,732
SGI	3,444
ISI	1,359
Mips	313
Cray	97

Figure 4.2: Breakdown of hosts by CPU brand as of June, 1992. Obtained with -hinfo option.

Internet Composition							
EDU	296,053	NL	19,580	DK	2,521	SG	870
COM	242,485	CH	15,353	KR	2,152	MX	722
GOV	54,072	FI	14,534	NZ	1,741	IE	583
AU	42,336	JP	14,102	TW	1,477	GR	565
DE	41,888	NO	13,707	ZA	1,401	IS	361
CA	35,556	NET	7,445	HK	1,374	PL	345
MIL	39,523	AT	5,171	FR	1,311	CS	190
UK	28,411	IT	4,783	PT	1,241	US	172
ORG	23,671	ES	2,860	BE	1,135	CL	106
SE	19,942	IL	2,727	BR	1,017	INT	41

Figure 4.3: Distribution of hosts in the top 40 largest domains of the Internet as of June, 1992.

Chapter 5

Conclusion

The distribution of information among all Internet domains provides a method of dealing with large amounts of data. For the purposes of studying the dynamics of growth, however, it is desirable to have the data all in one database. This is what Census is designed to accomplish. Though Census was designed to collect only host information, the data collected also provides useful information about the implementation of DNS throughout the Internet.

The Internet's explosion of growth is showing no signs of slowing down, as more organizations follow the standards of the Internet Protocol. In many ways, it is unfortunate that studying the growth dynamics of the Internet is outside the scope of this report. However, there is now a tool to do just that, and I have no doubt that there will be interest in this topic, not only within computer science but in other fields as well.

Acknowledgments

I could devote an entire report to the information I learned from my many interactions with Professor Darrell Long, not only about the workings of networks, UNIX, and world religion, but also on the value of finding answers for yourself.

Thanks to Dean Long and Tim Kolar for their innumerable contributions and bug-fixes. A special thanks to Leslie Bunnage for her extensive writing assistance and general inspiration. I'd also like to thank Michelle Abram, Richard Golding, William Osser, David Schreiber, and Bob Ellefson for their countless contributions. Finally, thanks to Professors Charlie M^{c}Dowell and Kim Taylor for their willingness to review my work of the past six months.

References

[1] Danzig, Peter "Probablistic Error Checkers : Fixing DNS", 13 pages, Spring 1992.

[2] Emtage, Alan "Archie – An Electronic Directory Service for the Internet," USENIX WINTER 92, 21 pages, Winter 92.

[3] Krol, E. "The Hitchhikers Guide to the Internet," RFC1118, 24 pages, September 1989.

[4] Lottor, Mark "Internet Growth (1981-1991)," RFC1296, 9 pages, January 1992.

[5] Mockapetris, P. "Domain Names—Concepts and Facilities," RFC1034, 55 pages, November 1987.

[6] Mockapetris, P. "Domain Names—Implementation and Specification," RFC1035, 55 pages, November 1987.

[7] Schwartz, M. "A Measurement Study of Changes in the Service-Level Reachability in the Global TCP/IP Internet", RFC1273, 8 pages, November 1991.

Appendix A – Manual Page

NAME

census – collect host information from specified domains.

SYNOPSIS

census [-stubborn] [-domain <name>] [-all] [-restart] [-hinfo] [-verbose]

DESCRIPTION

Census is a tool used to collect a list of host names, addresses, and possibly host CPU type and operating system. Census works by reading domains from domains.in, and querying the domain's name servers to obtain the desired information. The default is a query to find all the domain's host names and addresses. Any new domains found are appended to domains.in to be queried once all previously specified domains are queried.

The input required to run census:

domains.in – a file that contains the names of domains that are to be queried. New domains that are discovered during nameserver queries are appended to the end of this file.

The output of census:

hosts.out – contains all host information from domains that have been successfully queried.

tryagain.out – the names of domains that have been un-successfully queried. These domains are tried again once all the domains in domains.in have been queried once.

successful.out – a list of all domains that have been successfully queried.

lastdomain.out – the last domain queried. Only used if census is run with the '-restart' option.

temporary.out – used to collect the hosts of a domain that is currently being queried. If zone data is sent successfully, the contents of temporary.out are appended to hosts.out, and the file is deleted.

OPTIONS

-a[ll] – removes the current domains.in file, and creates one containing all top-level domains.

-h[info] – perform a hinfo query on each host, to find the CPU type and operating system. Will return 'unknown' if one isn't found.

-r[estart] – run census after the domain specified in lastdomain.out. Use in case data collection is halted.

-v[erbose] – Verbose output to stdout. Tells how many name servers were discovered, which name server is currently being queried for hosts, and any name server errors.

-d[omain] <domain name> – Creates a new domains.in file with only the domain specified, and starts the domain traversal at that part of the domain 'tree'.

-s[tubborn] – Remove the current domains.in and move all domains in tryagain.out to domains.in to be retried.