Taking Care of Your Domain

Artur Romão
<artur@fct.unl.pt>

FCCN

May, 1994

Document-ID: ripe-114

1. INTRODUCTION

Today more than 2,200,000 computers are inter-connected in a global Internet, comprising several millions of end-users, able to reach any of those hosts just by naming it. This facility is possible thanks to the world widest distributed database, the Domain Name System, used to provide distributed applications various services, the most notable one being translating names into IP addresses and vice-versa. This happens when you do an ftp or telnet, when your gopher client follows a link to some remote server, when you click on a hypertext item and have to reach a server as defined by the URL, when you talk to someuser@some.host, when your mail has to be routed through a set of gateways before it reaches the final recipient, when you post an article to Usenet and want it propagated all over the world. While these may be the most visible uses of DNS, a lot more applications rely on this system to operate, e.g. network security, monitoring and accounting tools, just to mention a few.

DNS owes much of its success to its distributed administration. Each component (called a zone, the same as a domain in most cases), is seen as an independent entity, being responsible for what happens inside its domain of authority, how and what information changes and for letting the tree grow downwards, creating new components. On the other side, many inconsistencies arise from this distributed nature: many administrators make mistakes in the way they configure their domains and when they delegate authority to sub-domains; many of them don't even know how to do these properly, letting problems last and propagate. Also, many problems occur due to bad implementations of both DNS clients and servers, especially very old ones, either by not following the standards or by being error prone, creating or allowing many of the above problems to happen.

All these anomalies make DNS less efficient than it could be, causing trouble to network operations, thus affecting the overall Internet. This document tries to show how important it is to have

DNS properly managed, including what is already in place to help administrators taking better care of their domains.

2. DNS DEBUGGING

To help finding problems in DNS configurations and/or implementations there is a set of tools developed specifically for this purpose. There is probably a lot of people in charge of domain administration having no idea of these tools (and, worse, not aware of the anomalies that may exist in their configurations). What follows is a description of some of these programs, their scope, motivations and availability, and is hoped to serve as an introduction to the subject of DNS debugging, as well as a guide to those who are looking for something to help them finding out how healthy their domains and servers are.

Some prior knowledge from the reader is assumed, both on DNS basics and some other tools (e.g. dig and nslookup), which are not analyzed in detail here; hopefully they are well-known enough from daily usage.

2.1. HOST

Host is a program used to retrieve DNS information from nameservers. This information may be used simply to get simple things like address-to-name mapping, or some more advanced purposes, e.g. performing sanity checks on the data. It was created at Rutgers University, but then Eric Wassenaar from Nikhef did a major rewrite and still seems to be actively working on it. The program is available from ftp.nikhef.nl:/pub/network/host_YYMMDD.tar.Z (YYMMDD is the date of the latest release).

By default, host just maps host names to Internet addresses, querying the default servers (as defined in /etc/resolv.conf) or some specific one. It is possible, though, to get any kind of data (resource records) by specifying different query types and classes and asking for verbose or debugging output, from any nameserver. You can also control several parameters like recursion, retry times, timeouts, use of virtual circuits vs. datagrams, etc., when talking to nameservers. This way you can simulate a resolvers behavior, in order to find any problems associated with resolver operations (which is to say, any application using the resolver library). As a query program it may not be as powerful as others like nslookup or dig, but you can live with it perfectly well.

As a debugger, host analyzes some set of the DNS space (e.g. an entire zone) and produces reports with the results of its operation. To do this, host first performs a zone transfer, which may be recursive, getting information from a zone and all its sub-zones. This data is then analyzed as requested by the arguments given on the command line. Note that zone transfers are done by contacting authoritative nameservers for that zone, so it must be possible to make this kind of request from such servers: some of them refuse zone transfers (except from secondaries) to avoid congestion.

With host you may look for anomalies like those concerning authority (e.g. lame delegations, described below) or some more exotic cases like extrazone hosts (a host of the form host.some.dom.ain, where some.dom.ain is not a delegated zone of dom.ain). These errors are produced upon explicit request on the command line, but you may get a variety of other error messages as a result of host's operations, something like secondary effects. These may be mere warnings (which may be suppressed) or serious errors (in fact, warning messages are not that simple, most of them are due to misconfigured zones, so it might not be a good idea to just ignore them).

Error messages have to do with serious anomalies, either with the packets exchanged with the queried servers (size errors, invalid ancounts, nscounts and the like), or others related to the DNS information itself (also called "status messages" in the program's man page): inconsistencies between SOA records as shown by different servers for a domain, unexpected address-to-name mappings, nameservers not responding, not reachable, not running or not existing at all, and so on.

Host performs all its querying on-line, i.e., it only works with data received from nameservers, which means you have to query a nameserver more than once if you want to get different kinds of reports on some particular piece of data. You can always arrange arguments in such a way that you get all information you want by running it once, but if you forget something or for any reason have to run it again, this means extra zone transfers, extra load on nameservers, extra DNS traffic.

Host is an excellent tool, if used carefully. Like most other querying programs it may generate lots of traffic, just by issuing a simple command. Apart from that, its resolver simulation and debug capabilities make it useful to find many common and some not so common DNS configuration errors, as well as generate useful reports and statistics about the DNS tree. As an example, RIPE (Reseaux IP Europeens) NCC uses it to generate a monthly european hostcount, giving an overview of the Internet usage evolution in Europe. Along with these counts, error reports are generated, one per country, and the whole information is made available in the RIPE archive.

2.2. DNSWALK

Dnswalk is a DNS debugger written in Perl by David Barr, from Pennsylvania State University. You'll find the latest version at ftp.pop.psu.edu, in directory /pub/src/dnswalk. With the software comes a small document where the author points some useful advices so it may be worth reading it.

The program checks domain configurations stored locally. Data is arranged hierarchically in directories, resembling the DNS tree organization of domains. To set up this information dnswalk may first perform zone transfers from authoritative nameservers. You can have a recursive transfer of a domain and its sub-domains, though you should take care when doing this, as it may generate a great amount of traffic. If the data is already present, dnswalk may skip these transfers, provided that it is up to date.

Dnswalk looks for inconsistencies in RRs, such as MX and aliases pointing to aliases or to unknwon hosts, incoherent PTR, A and CNAME records, invalid characters in names, missing trailing dots, unnecessary glue information, and so on. It also does some checking on authority information, namely lame delegations and domains with only one nameserver. It is easy to use, you only have to specify the domain to analyze and some optional parameters and the program does the rest. Only one domain (and its sub-domains, if that's the case) can be checked at a time, though.

While in the process of checking data, dnswalk uses dig and resolver routines (gethostbyXXXX from the Perl library) a lot, to get such data as authority information from the servers of the analyzed domains, names from IP addresses so as to verify the existence of PTR records, aliases and so on. So, besides the zone transfers you may count on some more extra traffic (maybe not negligible if you are debugging a relatively large amount of data and care about query retries and timeouts), just by running the program.

As of this writing, the program's operation is somewhat constrained by dig's behavior and the author is working hard on "un-dig'ing" it, using just Perl routines. It may be worthwhile upgrading your version if the new, dig-free dnswalk is available.

2.3. LAMERS

A lame delegation is a serious error in DNS configurations, yet a (too) common one. It happens when a nameserver is listed in the NS records for some domain, and in fact it is not a server for that domain. Queries are thus sent to the wrong servers, who don't know nothing (at least not as expected) about the queried domain. Furthermore, sometimes the hosts in question (if they exist!) don't even run nameservers. As a result, queries are timed out and resent, only to fail, thus creating (more) unnecessary traffic on the Internet.

It's easy to create a lame delegation: the most common case happens when an administrator changes the NS list for his domain, dropping one or more servers from that list, without informing his parent domain administration, who delegate him authority over his domain. From now on the parent nameserver announces one or more servers for the domain, which will receive queries for something they don't know about. On the other hand, servers may be added to the list without the parent's servers knowing, thus hiding valuable information from them (this is not a lame delegation, but is also an error). Other examples are the inclusion of a name in an NS list without telling the administrator of that host that it's being announced as a nameserver for a domain it doesn't know about, or when a server suddenly stops providing name service for the domain.

To detect and warn DNS administrators all over the world about this problem, Bryan Beecher from University of Michigan wrote lamers, a program to analyze named's logging information [BEE92]. To produce useful logs, named was applied a patch to detect (and log) lame delegations (this patch was originally written by Don Lewis from Silicon Systems and is now part of the latest, experimental, release of BIND thanks to Bryan Beecher, so it is expected to be widely available in the near future). Lamers is a small shell script that simply scans these logs and reports on the lame delegations found. This reporting is done by sending mail to the hostmasters of affected domains, as stated in the SOA record for each of them. If this is not possible, the message is sent to the affected nameservers postmasters instead. Manual processing is needed in case of bounces, caused by careless setup of those records or invalid postmaster addresses. A report of the errors found by the UM servers is also posted twice a month on the USENET newsgroup comp.protocols.tcp-ip.domains.

If you ever receive such a report, you should study it carefully in order to find and correct problems in your domain, or see if your hosts are being affected by the spreading of erroneous information. Better yet, lamers could be run on your servers to detect more lame delegations (UM can't see them all!). Also if you receive a mail reporting a lame delegation affecting your domain or some of your hosts, don't just ignore it or flame the senders. They're really trying to help!

You can get lamers from terminator.cc.umich.edu:/dns/lamers.sh.

2.4. DOC

Authority information is one of the most significant part of the DNS data, as the whole mechanism depends on it to correctly traverse the domain tree. Incorrect authority information leads to problems such as lame delegations or even, in extreme cases, the inaccessibility of a domain. Take the case where the information given about all its nameservers is incorrect. Being unable to contact the real servers you will end up being unable to reach anything inside that domain. This may be exagerated, but if you're on the DNS business long enough you've probably have seen some lightened examples of this scenario.

To look for this kind of problems Paul Mockapetris and Steve Hotz, from the Information Sciences Institute, University of Southern California, wrote a C-shell script named DOC (Domain Obscenity Control), an automated domain testing tool that uses dig to query the appropriate nameservers and analyzes the responses.

DOC limits its analisys to authority data since the authors antecipated that people would complain about such things as invasions of privacy. Also, at the time it was written most domains were so messy that they thought there wouldn't be much point in checking anything deeper until the basic problems weren't fixed.

Only one domain is analyzed each time: the program checks if all the servers for the parent domain agree about the delegation information for the domain in question. DOC then picks a list of nameservers for the domain (obtained from one of the parent's servers) and starts the checking on its information, querying each of them: looks for the SOA record, checks if the response is authoritative, compares the various records retrieved, gets each one's list of NS, compares the lists (both among these servers and the parent's servers), and for those servers inside the domain the program looks for PTR records for them.

Due to several factors, DOC seems to have freezed since its first public release, back in 1990. Within the distribution there is an RFC draft about automated domain testing, which was never published. Nevertheless, it may provide useful reading. The software can be fetched from ftp.uu.net:/networking/ip/dns/doc.2.0.tar.Z.

2.5. DDT

DDT (Domain Debug Tools) is a package of programs to scan DNS information for error detection, developed originally by Jorge Frazao from PUUG - Portuguese UNIX Users Group and later rewritten by the author, at the time at the Faculty of Sciences of University of Lisbon. Each program is specialized in a given set of anomalies: you have a checker for authority information, another for glue data, mail exchangers, reverse-mappings and miscellaneous errors found in all kinds of resource records. As a whole, they do a rather extensive checking on DNS configurations.

These tools work on cached DNS data, i.e., data stored locally after performing zone transfers (done by a slightly modified version of BIND's named-xfer, called ddt-xfer, which allows recursive transfers) from the appropriate servers, rather than querying nameservers on-line each time they run. This option was taken for several reasons [FRA92]: (1) efficiency, since it reads data from disk, avoiding network transit delays, (2) reduced network traffic, data has to be fetched only once and then run the programs over it as many times as you wish and (3) accessibility - in countries with limited Internet access, as was the case in Portugal by the time DDT was in its first stages, this may be the only practical way to use the tools.

Point (2) above deserves some special considerations: first, it is not entirely true that there aren't additional queries while processing the information, one of the tools, the authority checker, queries (via dig) each domain's purported nameservers in order to test the consistency of the authority information they provide about the domain. Second, it may be argued that when the actual tests are done the information used may be out of date. While this is true, you should note that this is the DNS nature, if you obtain some piece of information you can't be sure that one second later it is still valid. Furthermore, if your source was not the primary for the domain then you can't even be sure of the validity in the exact moment you got it in the first place. But experience shows that if you see an error, it is likely to be there in the next version of the domain information (and if it isn't, nothing was lost by having detected it in the past). On the other side, of course there's little point in checking one month old data...

When building a cache to analyze, you should be careful in getting all the relevant data, which includes both normal domain and reverse-mapping information, or else you get lots of "no PTR record found" messages. This means that using DDT requires some prior planning, so that all the necessary data is available, since the tools don't go out looking for more information, with the exception of authority one (see above). Once again, care should be taken when doing recursive transfers in order not to generate unnecessary traffic. If a host performs nameservice for some domains DDT can use the zone files already stored locally.

DDT tries to look for all kinds of errors, as known by the authors at the time of writing it. The list includes lame delegations, version number mismatches between servers (this may be a transient problem), non-existing servers, domains with only one server, unnecessary glue information, MX records pointing to hosts not in the analyzed domain (may not be an error, it's just to point possibly strange or expensive mail-routing policies), MX records pointing to aliases, A records without the respective PTR and vice-versa (may not be an error, see the paragraph above), missing trailing dots in configuration files, hostnames with no data (A or CNAME records) associated, aliases pointing to aliases, and some more. Given the specialized nature of each tool, it is possible to look for a well defined set of errors, instead of having the data analyzed in all possible ways. Within the package comes a set of small programs that perform several kinds of statistics on the cached data. Among these you have hosts per domain, nets per domain, aliases in a domain, most popular names, etc.

Except for ddt-xfer, all the programs are written in Perl. A new release may come into existence in a near future, after a thorough review of the methods used, the set of errors checked for and some bug fixing, of course. In the mean time, the latest version is available from ns.dns.pt:/pub/dns/ddt-2.0.tar.gz.

2.6. THE CHECKER PROJECT

The problem of the huge amount of DNS traffic over the Internet is getting researchers close attention for quite some time, mainly because most of it is unnecessary. Observations have shown that DNS consumes something like twenty times more bandwidth than it should [DAN92a]. Some causes for this undoubtedly catastrophic scenario lie on deficient resolver and nameserver implementations spread all over the world, from personal to super-computers, running all sorts of operating systems.

While the panacea is yet to be found (claims are made that the latest official version - 4.9.2 as of this writing - of BIND is a great step forward [KUM93]), work has been done in order to identify sources of anomalies, as a first approach in the search for a solution. The Checker Project is one such effort, developed at the University of Southern California [MIL94]. It consists of a set of C code patched into BIND's nameserver named, which monitors server activity, building a database with the history of that operation (queries and responses). It is then possible to generate reports from the database summarizing activity and identifying behavioral patterns from client requests, looking for anomalies. The named code alteration is small and simple unless you want do have PEC checking enabled (see below). You may find sources and documentation at caldera.usc.edu, in directory /pub/checker.

Checker only does this kind of collection and reporting, it does not try to enforce any rules on the administrators of the defective sites by any means whatsoever. Authors hope that the simple exhibition of the evidences is a reason strong enough for those administrators to have their problems corrected.

An interesting feature is PEC (proactive error checking): the server pretends to be unresponsive for some queries by randomly choosing some name and start refusing replies for queries on that name during a pre-determined period. Those queries are recorded, though, to try to reason about the retry and timeout schemes used by nameservers and resolvers. It is expected that properly implemented clients will choose another nameserver to query, while defective ones will keep on trying with the same server. This feature is still being tested as it is not completely clear yet how to interpret the results.

Presently Checker has been running on a secondary for the US domain for more than a year with little trouble. Authors feel confident it should run on any BSD platform (at least SunOS) without problems, and is planned to be included as part of the BIND nameserver.

Checker is part of a research project lead by Peter Danzig from USC, aimed to implement probabilistic error checking mechanisms like PEC on distributed systems [DAN92b]. DNS is one such system and it was chosen as the platform for testing the validity of these techniques over the NSFnet. It is hoped to achieve enough knowledge to provide means to improve performance and reliability of distributed systems. Anomalies like undetected server failures, query loops, bad retransmission backoff algorithms, misconfigurations and ressubmission of requests after negative replies are some of the targets for these checkers to detect.

2.7. OTHERS

All the tools described above are the result of systematic work on the issue of DNS debugging, some of them included in research projects. For the sake of completeness we mention here several other programs that, though just as serious, seem to have been developed in a somewhat adhoc fashion, without an implicit intention of being used outside the environments where they were born. This impression is, of course, arguable, nevertheless we didn't feel the necessity of dedicating an entire section to any of them. This doesn't mean they are not valuable contributions, in some cases they may be just what you are looking for, without having to install a complete package to do some testings on your domain.

We took as a reference the contrib directory in the latest BIND distribution. There you will find tools for creating your DNS configuration files and NIS maps from /etc/hosts and vice-versa or generate PTR from A records (these things may be important as a means of avoiding common typing errors and inconsistencies between those tables), syntax checkers for zone files, programs for querying and monitoring nameservers, all the short programs presented in [ALB93], and more. It is worth spending some time looking at them, maybe you'll find that program you were planning to write yourself. BIND can be found at gatekeeper.dec.com, in the directory /pub/misc/vixie (the file 4.9.2-940221.tar.gz has the latest public version).

You may also want to consider using a version control system (e.g. SCCS or RCS) to maintain your configuration files consistent through updates, or use tools like M4 macros to generate those files. As stated above, it's important to avoid human-generated errors, creating problems that are difficult to track down, since they're often hidden behind some mistyped name. Errors like this may end up in many queries for a non-existing name, just to mention the less serious kind. See [BEE93] for a description of the most common errors made while configuring domains.

3. WHY LOOK AFTER DNS?

We have presented several pieces of software to help people administer and debug their name services. They exhibit many differences in their way of doing things, scope and requirements and it may be difficult just to choose one of them to work with. For one thing, people's expectations from these tools vary according to their kind of envolvement with DNS. If you are responsible for a big domain, e.g. a top-level one or a big institutions with many hosts and sub-domains, you probably want to see how well is the tree below your node organized, since the consequences of errors tend to propagate upwards, thus affecting your own domain. For that you need some program that recursively descends the domain tree and analyzes each domain per se and the interdependencies between them all. You will have to consider how deep you want your analysis to be, the effects it will have on the network infrastructure, i.e. will it generate traffic only inside a campus network, no matter how big it is, or will it be spread over, say, a whole country (of course, your kind of connectivity plays an important role here).

You may simply want to perform some sanity checks on your own domain, without any further concerns. Or you may want to participate in some kind of global effort to monitor name server traffic, either for research purposes or just to point out the "trouble-queries" that flow around.

Whatever your interest may be, you can almost surely find a tool to suit it. Eliminating problems like those described in this document is a major contribution for the efficiency of an important piece of the Internet mechanism. Just to have an idea of this importance, think of all the applications that depend on it, not just to get addresses out of names. Many systems rely on DNS to store, retrieve and spread the information they need: Internet electronic mail was already mentioned (see [PAR86] for details) and work is in progress to integrate, for example, X.400 operations with DNS; others include "remote printing" services [MAL93], distributed file systems [BIR93] and network routing purposes, among others. These features may be accomplished by some standard, well-known resource records [ROS93], or by new, experimental ones [EVE90], [MAN93]. Even if some of them won't succeed, one may well expect some more load on the DNS burden.

The ubiquitous DNS thus deserves a great deal of attention, perhaps much more than it generally has. One may say that it is a victim of its own success: if a user triggers an excessive amount of queries only to have one request satisfied, he won't worry about it (in fact, he won't notice it), won't complain to his system administrator, and things will just go on like this. Of course, DNS was designed to resist and provide its services despite all these anomalies. But, by doing so it is frequently forgotten, as long as people can telnet or ftp. As DNS will be given new responsibilities, as pointed in the above paragraph, the problems described in this text will grow more serious and new ones may appear (notably security ones [GAV93]), if nothing is done to purge them. We hope to have gain people's attention to this subject, so that the Net can benefit from a healthy, properly administered and used Domain Name System.

4. REFERENCES

[ALB92]

Albitz, P. and C. Liu. DNS and BIND. O'Reilly and Associates Inc., Oct. 1992

[BEE92]

Beecher, B. Dealing With Lame Delegations. Univ. Michigan, LISA VI, Oct. 1992

[BEE93]

Beertema, P. Common DNS Data File Configuration Errors. CWI, RFC 1537, Oct. 1993

[BIR93]
Birrel, A. D., A. Hisgen, C. Jerian, T. Mann and G. Swart. The Echo Distributed File System. SRC Research Report 111, Digital Equipment Corporation, Sep. 1993

[DAN92a]
Danzig, P. Probabilistic Error Checkers: Fixing DNS. Univ. Southern California, Technical Report, Feb. 1992

[DAN92b]
Danzig, P., K. Obraczka and A. Kumar. An Analisys of Wide-Area Name Server Traffic. Univ. Southern California, TR 92-504, 1992

[EVE90]
Everhart, C., L. Mamakos, R. Ullmann and P. Mockapetris (Ed.). New DNS RR Definitions. Transarc, Univ. Maryland, Prime Computer, Information Sciences Institute, RFC 1183, Oct. 1990

[FRA92]
Frazao, J. and J. L. Martins. Ddt - Domain Debug Tools, A Package to Debug the DNS Tree. Dept. Informatica da Fac. Ciencias Univ. Lisboa, DI-FCUL-1992-04, Jan. 1992

[GAV93]
Gavron, E. A Security Problem and Proposed Correction With Widely Deployed DNS Software. ACES Research Inc., RFC 1535, Oct 1993

[KUM93]
Kumar, A., J. Postel, C. Neuman, P. Danzig and S. Miller. Common DNS Implementation Errors and Suggested Fixes. Information Sciences Institute, Univ. Southern California, RFC 1536, Oct 1993

[MAL93]
Malamud, C., M. Rose. Principles of Operation for the TPC.INT Subdomain: General Principles and Policy. Internet Multicasting Service, Dover Beach Consulting Inc., RFC 1530, Oct. 1993

[MAN92]
Manning, B. DNS NSAP RRs. Rice University, RFC 1348, Jul. 1992

[MIL94]
Miller, S. and P. Danzig. The Checker Project, Installation and Operator's Manual. Univ. Southern California, TR CS94-560, 1994

[PAR86]
Partridge, C. Mail Routing and the Domain System. CSNET CIC BBN Laboratories Inc, RFC 974, Jan. 1986

[ROS93]
Rosenbaum, R. Using the Domain Name System to Store Arbitrary String Attributes. Digital Equipment Corporation, RFC 1464, May 1993