October 2013
(updated 10 November)
Geoff Huston

Dotless

It was never obvious at the outset of this grand Internet experiment that the one aspect of the network's infrastructure that would truly prove to be the most fascinating, intriguing, painful, lucrative and just plain confusing, would be the Internet's Domain Name System.

After all, it all seemed so simple to start with: network applications rendezvous with their counterparts using protocol-level addresses, but we users prefer to use "natural" identifiers that act as aliases for these addresses. So rather than trying to remember that my local server is 192.0.2.34, or, perhaps more challenging, 2001:db8:a25d:8664:dff1:73b6:fe40:34, I can give it a memorable name, such as "server". So I make an entry in a local translation table that makes the name "server" equivalent to those two IP addresses, and now I can use the term "server" to direct applications to rendezvous with this system.

This works well in a small scale local context, but how do we scale this up to a larger network that spans many local name contexts? The answer that was devised back in the early 1980's was to use a hierarchically structured common name scheme. So we had some generic so-called "top-level" names, such as *.edu*, *.gov*, *.mil*, *.org*, and *.com*, and folk then reserved a name in one of these name spaces, and then populated the lower levels of the name hierarchy with their particular needs for identification. So I now might have my *server* name now called *server.example.com*, assuming that I had been delegated the name space of *example* within *.com*.

Over the years the set of these top-level domain names have expanded. A major expansion was to integrate the 2 letter country codes from the ISO-3166 country name table into the set of top level DNS labels. Since then there has been a continual discussion about expansion of the so-called "generic top-level domain name" (gTLDs) set (as distinct from the set of ISO-3166 two letter domain names that map to country codes, whose management is considered to be a national issue for each country to work through). Under the umbrella of ICANN's stewardship of the top level domain space of the DNS some 14 new gTLDs were added in the past 13 years, bringing the total to some 22 currently available names (most notable of which was the *.xxx* gTLD, that excited considerable interest at the time). However, after more than a decade of careful rationing and slow progress in terms of opening up the top level name space of the DNS, at the start of 2012 ICANN changed its stance and opened up the gTLD space for applicants. Of course nothing in the DNS comes for free, and the cost of making such an application is reported to be some $185,000. (Illustrating that even in domain names vanity comes at a considerable price!) Despite this application cost, some 1,930 application were received for new gTLDs, and the process of evaluation of these applications is underway.

"Dotless" TLDs

Within the evaluation process there have evidently been a number of proposals that propose relatively novel use of these top level DNS labels, and the one I'd like to look at here is that of the so-called "dotless" domains proposed for the DNS root zone. It's as if I had decided to revive the name *server* and propose it to be not only a top-level domain name that is defined across the entire Internet, but to have this name resolve to an IP address.

Can such dotless domain names, such as *server*, really function as top level domains?

For example, if I have my way, and *server* becomes a new gTLD, will that imply that everyone that entered *server* into their web browser, or everyone who sends mail to *me@server*, or uses this *server* name in any other network application will end up communicating with my server?

The DNS is not so simple, and in this case we have developed so called "search lists" that interfere with these so-called "dotless" domain names in various ways. The concept behind the search list was to allow local names to be used without explicitly adding a global DNS context. I can use a local service name for a named service and have the DNS automatically append the common DNS suffix to the local name to form a fully qualified DNS query name. So, if I have a local search list with *example.com*, then when I use the name *server*, the local DNS resolution system will append the search list, *example.com*, and then attempt to resolve the fully qualified name *server.example.com*. So how do search lists actually work? Or perhaps maybe we should first ask: how were search lists intended to work?

Consulting the RFCs

The base specification of the DNS, RFC1034, assumed that any name requiring resolution was either a complete (or "absolute") domain name, or an incomplete (or "relative") domain name. Absolute domain names were names that would be passed as a query to the DNS without further modification, while relative domain names required the appending of a locally defined name to form an absolute name. RFC1034 proposed that name forms that ended with a "." were to be interpreted as absolute names, while other names were relative names.

This approach has the downside of potentially directing many queries to the servers who are authoritative for the root of the DNS, and this specification in RFC1034 was subsequently modified in RFC1123 such that the local name resolution system should require that a name has three or more individual labels ("two or more interior dots in a generated domain name") before submitting the query to the DNS for resolution.

But even this approach generates excessive query traffic at the root of the DNS. RFC1536 noted the example of a local name of *usc.edu* within a local domain search space of *foo.bar.com*. Because the original name is missing that essential trailing dot it is being interpreted as a relative domain name, and the search list of *foo.bar.com* is appended, resulting in a query for *usc.edu.foo.bar.com*, then *usc.edu.bar.com*, then finally *usc.edu*. RFC1536 (an informational RFC) proposed a modification to the original absolute/relative name distinction, that a name that already contains multiple labels ("contains any dots", to quote RFC1536) should first be tried as a fully qualified domain name, and if that fails then the local search list can be applied to the name. In other words the reverse of the original search list behaviour. If the name is a single label, then the search list can be applied immediately. RFC3397 attempted to clarify this application of search lists to local names, taking the description in RFC1536 and re-casting them in the context of a Proposed Standard, without changing the substance of the proposed modifications.

So how have implementations interpreted these specifications?

Testing Search Lists and Dotless Names

How do various name resolver libraries implement the application of search lists in their local name resolution libraries? Will "dotless" domain names in the DNS be masked out by local domain name search lists? I have performed some basic tests on a number of popular operating systems, using their default configuration settings for name resolution.

The tests were set up using a simple application that called the local system's name resolution subroutine library. The local system's DNS query traffic was captured using tcpdump to allow the DNS queries to be recorded. There are four types of behaviour that have been observed in this experiment:

- **never** - the search list is not applied, and the original name is queried in the DNS
- **always** - the search list is always applied and the synthesised names are queried in the DNS, but the original name is never queried in the DNS

- **pre** - the search list is applied to the original name in DNS queries, and if all permutations of the application of the search list generate a NXDOMAIN response then the original name is queried in the DNS

- **post** - the original name is queried in the DNS, and if this generates an NXDOMAIN response then the search list is applied to the original name in DNS queries.

The results of this survey of the name resolution behaviour of the more popular systems are shown in the following table:

System	Absolute *server.*	Relative Single Label *server*	Relative Multi-Label *www.server*
MAC OSX 10.9	never	always	never
Windows XP	never	always	post
Windows Vista	never	always	never
Windows 7	never	always	never
Windows 8	never	always	never
FreeBSD 9.1	never	pre	post
Ubuntu 13.04	never	pre	post

Table 1 – Application of Search Lists in Name Resolution for various Operating Systems

The mix of behaviours seen here shows some evolution in the thinking about the appropriate use of search names across the various operating system vendors.

In the case of "absolute" domain names (ending with a dot) all systems interpret the name as a fully qualified name and do not apply any local search list.

In the case of so-called "relative names", the behaviours are variable. For single label domains we see many systems use the RFC1034 specification, and assume that the original name is a relative name and requires the use of the local search list in all cases. The FreeBSD and Ubuntu systems use a variation to his approach, and if the application of the search list yields no such domain (NXDOMAIN) responses in all cases, then a query for the dotless name is then passed into the DNS.

For multi-label domains some systems implement the procedures described in RFC1536, and treat the multi-label name firstly as an absolute name, and if that produces a NXDOMAIN response, then the local search list is applied. However later versions of Windows (Vista, 7 and 8) and the current MAC OSX systems treat the multi-label name as an absolute name and will not apply the local search list, even if the queries based on the application of the search list all generate NXDOMAIN responses.

The FreeBSD and Ubuntu behaviours also appear anomalous to me, in so far as the ordering of the search list and the base name queries is reversed, depending on whether the base name was dotless or not, but perhaps that a criticism of the state of the standard specification where the combination of RFC1034 and RFC1536 can lead an implementor to this outcome as being consistent with the specifications contained in these two documents.

But that's what happens when an application allows the underlying operating system to perform the name resolution task. The browser writers appear to have largely taken responsibility for their own name resolution needs, and browsers appears to perform the task differently. Perhaps more curiously we also see that some browsers perform differently on some platforms than others. Table 2 shows the

results of testing the same three variants of queried name forms on a number of browser platforms on two different operating system platforms.

There is a new behaviour that some browsers have implemented:

- **none** - the search list is not applied, and the original name is not queried in the DNS

This behaviour is evident with Safari on Mac OS platforms, as Safari appears to have an internal copy of the root zone, which is in effect a list of all delegated tld strings. If the name being queried is not part of a known tld it will not query the DNS at all, and pass the string over to the configured search tool instead.

MacOS OSX 10.9

Browser\Query	**Absolute** *server.*	**Relative Single Label** *server*	**Relative Multi-Label** *name.server*
Chrome (31.0.1650.39 beta)	never	always	pre
Opera (12.16)	never	always	never
Firefox (25.0)	post*	always	post*
Safari (7.0 9537.71)	none**	none**	none**

* Firefox looked up the base name, then added prefix of "www.", then tried prefixing the "www." and also appending the search list. If the base name starts with "www." it will look up the base name, then lookup the base name with the search list appended

** Safari seems to be aware of TLDs and does not perform DNS lookups when the name is not a known delegated TLD, irrespective of the local search list. Safari is also aware of SLD-restricted TLDs and will not perform queries when the SLD is restricted, unless the SLD matches the restricted list. In known TLDs Safari will query <name>.tld and then www.<name.tld>, unless the query name already starts with "www."

Windows 8.1

Browser\Query	**Absolute** *server.*	**Relative Single Label** *server*	**Relative Multi-Label** *name.server*
Chrome (30.0.1599.101 m)	never	always	never
Opera (17.0)	none	none	none*
Firefox (25.0)	never**	always	never**
Safari (5.1.7 7534.57.2)	never***	always****	never***
Explorer (11.0.900.16384)	never	none	never

* Opera is aware of delegated tlds, and only queries the DNS when the last label is a known TLD. In that case it does not use the local search list

** Firefox performs a second query by adding a prefix of "www" if the base name has only one or two labels and the base name query resulted in NXDOMAIN

*** Safari performed a second query by adding a prefix of "www" to the base name

**** Safari performed an additional query after querying for the basename plus the search list, by querying the FQDN "www.<base name>.com

Table 2 – Application of Search Lists in Name Resolution for various Browser and Operating System configurations

The variance of behaviours in this table is unexpected. Also the efforts by some browsers to load the list of delegated TLD strings, and SLD strings in restricted TLDs is entirely surprising to me.

There are some subtleties here in the above tables that probably need to be stated up front. In the Old Days, browsers were equipped with a "go" bar. You were supposed to enter URLs of the form "`http://<domain_name>/<locator_suffix>`", but we quickly got tired of all that silly cruft to type in. So the first change was to eliminate the need for the "`http://`" prefix. If you didn't enter it, the browser simply assumed that's what you meant anyway. Then search engines came along, and for many years browsers

had two data entry panels: one for URLs and one for search terms. However, all along it seems that we really wanted something simpler, and browsers eventually adjusted their interface to use a single data entry panel. If want you entered looked like a search term it was sent to the search engine, otherwise it was treated like a URL, and the domain name part of the URL was queried in the DNS. The variance between browsers and operating systems platforms that we observe in Table 2 in not in fact variance of the underlying DNS resolution library, but variance in the ruleset used by browsers to distinguish between a search term and a candidate URL.

Can we remove the factor of the varying search term/URL rulesets and expose the underlying DNS library name resolution behaviours? One approach is to enter the "http://" prefix into the browser along with the name being tested, but in some browsers the input processing appears to strip off this prefix and pass the resultant string back into a common input processing module that attempts to distinguish between search terms and URLs. Another approach is to code up the tests in Javascript, and have the script generate these unique URLs and then instruct the browser to perform the fetch. The script is at http://www.potaroo.net/tools/dnstest.html, and it simply instructs the browser to load 3 objects, each of which use a name form that includes a unique digit string. The three domain names are composed of a single label, two labels and three labels, and are intended to be nonsensical (that is, a DNS query for these domains will generate NXDOMAIN responses). If you run up a packet capture tool such as *tcpdump* it is possible to expose the DNS queries your browser makes as it attempts to resolve these names.

The results of this scripted test of browsers' name resolution behaviour is shown in Table 3.

MacOS OSX 10.9

Browser\Query	**Single Label** *label1*	**Multi-Label** *label1.label2*
Chrome (31.0.1650.39 beta)	always	post
Opera (12.16)	always*	never
Firefox (25.0)	always	never
Safari (7.0 9537.71)	always	never

* Opera also looks up www.*label1*.(com|org|net|edu|gov)

Windows 8.1

Browser\Query	**Single Label** *label1*	**Multi-Label** *label1.label2*
Chrome (30.0.1599.101 m)	always	never
Opera (17.0)	always	never
Firefox (25.0)	always	never
Safari (5.1.7 7534.57.2)	always	never
Explorer (11.0.900.16384)	always	never

Table 3 – Application of Search Lists in Scripted Name Resolution for various Browser and Operating System configurations

This result is perhaps a little more satisfying, in so far as, with the exceptions of Chrome and Opera on Mac OSX, all the surveyed browsers appear to behave consistently. With the noted exceptions, the remaining browser/OS combinations always apply the local domain name search list when attempting to resolve single label names, and never apply it to multi-label names.

Observations on Going Dotless

After all this, the question still remains: if you want to use a dotless label as a domain name, would it work? Would applications, including browsers, pass the single word query to the DNS for name resolution?

It's not looking good.

You could improve that answer a little if you somehow coerced all forms of use of the name to include the trailing dot to re-cast the name as an absolute name. However, the use of the trailing dot is largely fallen into disuse in terms of our use of domain names, and in its relative name form without that trailing dot, the dotless name looks a whole lot less viable.

It seems that most environments will interpret a single label as an incomplete identifier, and will attempt to use additional hints to create a multi-label name before querying the DNS. On the Mac OSX and recent Windows operating systems, the system libraries apply the local search list to the name, and then attempt to resolve the consequent multi-label name. In such contexts the dotless name is never queried. For FreeBSD and Ubuntu systems (and presumably Fedora systems and other Linux variants) the local system will first apply the local DNS search list to the name, and only if these names all generate NXDOMAIN responses would the local system query for the dotless name. Some browsers will always pass the dotless name to a search engine and never query the DNS, while others will always apply the local search list and if that does not resolver, then pass the base name to the search system. None of the browser/operating system combinations test here will query the DNS for the single label.

So, would "going dotless" really work?

I just can't see it working, so for me the answer is: "Probably not!"

Name Collision?

Into this somewhat convoluted picture of name resolution comes a related concept of "name collision". One way to phrase this question is: "Would the delegation of a new gTLD name somehow alter the name resolution behaviour of clients if the same name was already being used in some purely local context?"

There are certain similarities with "address squatting" relating to prior unauthorised users of unallocated IP address ranges. Would users who had, in effect, created "pseudo-TLDs" and defined local domain name scopes where the local domain name servers would answer queries into these pseudo-TLDs as if they were duly delegated TLDs, suffer from some form of altered and potentially compromised application behaviour in the event that this name was subsequently used as a duly delegated gTLD?

If the local use is accompanied by a locally scoped name resolver, then local clients will still resolve names in this name space within the context of the local resolver, and nothing has changed for them. If local clients were relocated into a name resolution environment outside the scope of the local name resolver, and these same users still attempted to resolve names using the local name space, then there are some potential issues. Whereas previously such displaced use of the name would result in NXDOMAIN responses from the global DNS, there is the potential for the name to be unexpectedly resolved in the context of this new gTLD. Now it must be stressed here that the problem is not the delegation of the gTLD per se that has caused this unexpected outcome, but the movement of a client system outside of the context of a locally scoped name environment that has been locally configured as a pseudo-TLD, while still retaining some vestige of the name's definition within the client's configuration settings. It is perhaps more that, as with many other cases of squatting, when the squatted use of resource conflicts with a duly delegated or authorised use of the same resource then there are inevitable conflicts.

The instance of single name queries in the queries presented to the DNS root servers is a testament to the observation that such locally scoped use of names, and the associated leakage of name scope, does occur, but the question is whether delegation of these name as a new gTLD would alter or otherwise compromise client behaviour in any fashion beyond what would be expected in an undelegated scenario. There is little to suggest form this particular survey of name resolution behaviours in popular

deployed systems and browsers that this scope leakage of private use of a name is a substantial issue, and there is nothing to suggest in the nature of the way local search lists are applied that the instantiation of a name as a new gTLD would significantly occlude the locally scoped use of the same name in ways that would compromise the operation of web services and other applications.

It appears that global definition of a name would not, in general, occlude the visibility and use of the same name in locally scoped name resolution contexts.

Perhaps the only substantive issue here is the reverse form of name occlusion, where if a name that is commonly used in private context is delegated in the DNS root as a new gTLD, then none of the clients in these private contexts will have any direct visibility of the global name.

The piecemeal fragmentation of the name space through the use of these locally scoped pseudo-gTLDs was never the best of ideas in an identifier realm such as the DNS that has no implicit understanding of name scoping or name realms. Applications, users and servers have all implicitly assumed that DNS names are globally scoped identifiers, and when local practices break this assumption then in some sense the consequences in terms of name use conflict are self-inflicted. As far as I can tell, the basic intention of local name search lists was not to augment a local name realm with privately used pseudo-TLDs, but to allow the local users to use a common global name implicitly, so that local use of simple, single label name forms can be transformed into fully qualified domain names. Within my local scope I can refer to a resource or service with a single label, and the local search list should transform this single dotless label into a fully qualified domain name that is then resolved in the context of the DNS itself.

However, the operators of the root name servers of the DNS report a continuing load of queries for what looks to be "leaking" local use domains. As to why do these names "leak", I suppose that the most obvious explanation is that many users may have their browser's home page set to some form of corporate network service name, including their mobile devices. When they move their mobile device to a different environment, the query for this locally scoped domain name would still take place, but in this latter case the query will head to the root name servers and be counted as a "leaked" name query.

The tables of operating system library and browser behaviours above point to the conclusion that while many operating system name resolution libraries and many browser implementations will take names and pass them into the DNS as FQDN names for DNS resolution. Particularly so with names that have two or more labels. However, there is no evidence that any significant number of clients exhibit name resolution behaviour, in the form of the application of search lists once the original DNS query has failed with an NXDOMAIN, that would be affected by the implementation of a name as a new gTLD. The "post" form of search list application is only visible in the tests described here in the operating system libraries of Windows XP and some Unix platforms, while Firefox on a Mac OSX Platform exhibited this behaviour when the name was entered by a user, and Chrome on a MAC also showed this behaviour when the name was generated by a Javascript execution. But these are exceptions in a much larger environment where that behaviour is not evident.

In other words, while there is name leakage to the root from local domains to the global domain, there is little evidence to suggest that if any of these names were subsequently delegated as new gTLDs then there would be a significant risk of some widespread form of threats to the integrity of local services as a consequence. For me, the reports of the dire consequences of such name collisions appears to look a lot like a *furphy*! (http://en.wikipedia.org/wiki/Furphy).

But something else that is important is evident even in this small scale survey of browsers and operating systems. Everyone appears to do it differently. If only we could agree on the importance of a consistent interpretation of identifiers in the Internet. If only we could agree on a consistent interpretation of the use of names and have our operating systems and browsers all operate within a mutually consistent framework. If only.

Weren't standards meant to address exactly these kinds of issues?

Further Reading:

Internet Architecture Board: “Dotless Domains Considered Harmful,” July 2013
http://www.iab.org/documents/correspondence-reports-documents/2013-2/iab-statement-dotless-domains-considered-harmful/

New gTLD Program Committee of the ICANN Board, "Approved Resolution on Dotless Domains", September 2013.
http://www.icann.org/en/groups/board/documents/resolutions-new-gtld-13aug13-en.htm

ICANN Security and Stability Advisory Committee, "SSAC Report on Dotless Domains", February 2012.
http://www.icann.org/en/groups/ssac/documents/sac-053-en.pdf

J. Levine, P. Hoffman, “Top Level Domains that Are Already Dotless,” Internet Draft, October 2013.
http://datatracker.ietf.org/doc/draft-hoffine-already-dotless/?include_text=1

G. Huston, O. Kolkman, A. Sullivan, W. Kumari, G. Michaelson, “Using Test Delegations from the Root Prior to Full Allocation and Delegation,” Internet Draft, October 2013.
http://tools.ietf.org/html/draft-kolkman-root-test-delegation-01

Disclaimer

About the Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001.

www.potaroo.net