DOI:10.1145/1610252.1610269

Article development led by acmqueue
queue.acm.org

DNS is many things to many people—perhaps too many things to too many people.

BY PAUL VIXIE

What DNS Is Not

A DOMAIN NAME SYSTEM (DNS) is a hierarchical, distributed, autonomous, reliable database. The first and only of its kind, it offers real-time performance levels to a global audience with global contributors. Every TCP/IP traffic flow including every Web page view begins with at least one DNS transaction. DNS is, in a word, glorious.

To underline our understanding of what DNS *is*, we must differentiate it from what it *is not*. The Internet economy rewards unlimited creativity in the monetization of human action, and fairly often this takes the form of some kind of intermediation. For DNS, monetized intermediation means lying. The innovators who bring us such monetized intermediation do not call what they sell lies, but in this case it walks like a duck and quacks like one, too.

Not all misuses of DNS take the form of lying. Another frequently seen abuse is to treat DNS as a directory system, which it is not. In a directory system one can ask approximate questions and get approximate answers. Think of a printed telephone white pages directory here: users often find what they want in the printed directory not by knowing exactly what the listing is but by starting with a guess or a general idea. DNS has nothing like that: all questions and all answers are exact. But DNS has at least two mechanisms that can be misused to support approximate matching at some considerable cost to everybody else, and a lot of that goes on.

Stupid DNS Tricks

The first widespread form of a DNS

lie was to treat DNS lookups as mapping requests. Content distribution networks (CDNs), such as Akamai, and Web optimizer products, such as Cisco Distributed Director, treat incoming DNS lookups as opportunities to direct the activities of Web browsers. Using the IP source address of a DNS request, these products and services try to guess the proximity of the requester to each of many replicated content servers. Based on the measured load of each content server's system and network, and on an estimate of each content server's proximity to that requester, a DNS response is crafted to direct that requester to the closest or best content server for that URI domain.

Problems abound from this approach, but none affects the CDN operator's revenue. First and foremost it is necessary to defeat or severely limit caching and reuse of this policy-based data ("DNS lies"). Caching and reuse, which once were considered essential to the performance and scalability of DNS, would allow a policy-based response intended for requester A also to be seen by requester B, which might not otherwise receive the same answer—for example, when server loads have changed and there's a new balance. The effects of this noncaching are a higher DNS request rate (perhaps leading to higher revenue for CDNs that charge by the transaction) and more network load for access-side networks and a slightly higher floor for average transaction time.

Furthermore, it has never been wise to assume that a DNS request's IP source address gives any hint of an end-system Web browser's network location. This is because DNS requests heard by a CDN come from recursive DNS servers as a result of cache misses; they do not come from end systems themselves. Some ISPs regionalize their recursive name servers, allowing CDNs to encode rules improving the quality of their estimates. Many recursive name servers are per-country or per-continent or even per-hemisphere, however, so it's always necessary for a CDN to deploy well-connected supernodes, and these always end up hearing a lot of out-of-region requests.

The primary benefit of a CDN is the same as gimmick-free outsourcing: it gives a content owner somebody to sue if things don't go well. That DNS system performance and stability must pay the price for such liability shielding is at best unfortunate. Given a CDN still requires supernodes that will hear many out-of-region requests, a gimmick-free approach here would be to answer DNS truthfully and let existing pseudorandom distribution mechanisms do their work. Noting that there is no patent on the existing pseudorandomization technologies and that nobody ever got fired for buying a CDN, we can expect to see more content distributed this way in the decades to come.

NXDOMAIN Remapping

Fairly often, as in millions of times per second worldwide, somebody looks up a domain name in DNS that isn't there. Maybe this is a user at a Web browser making a typographical error, or maybe there's a broken link on a Web site, or maybe a hardware or software error is causing nonexistent names to go into DNS requests. One way or another, the answer is generally supposed to be NXDOMAIN (sometimes written as RCODE=3). These negative answers are cacheable, as is any other kind of DNS information, since DNS is designed to express truth, not policy. A network application (perhaps a Web browser, or mail server, or indeed anything at all that uses TCP/IP flows to do its business) that gets back one of these negative responses is supposed to treat it as an error and reject its own underlying work item that led to this lookup. For a Web browser, rejection takes the form of an "error page." For a mail server, rejection takes the place of "bounced email." Every TCP/IP application, large or small, new or old, knows how to cope with NXDOMAIN.

The Web has changed the rules. Though the Web is young—and though the Internet was here before the Web and will be here after the Web and is much larger than the Web—the fact remains that the Web is what end users are looking at. Advertisers have a whole language to describe the value of end users, with words such as "impressions," "click-throughs," and "eyeballs." Why on earth, these advertisers ask, would you ever send back an NXDOMAIN if an impression was possible? So it is, increasingly, that in place of the NXDOMAIN your application knows how to handle, if you ask for a name that does not exist, you'll get a positive (deceptive; false; lying) answer that your application also knows how to handle.

Author Paul Vixie, noted by *Wired* magazine as "the godfather of DNS," has been solving DNS errors and mysteries for over 20 years.

For example, if I ask my own recursive name server for a name that does not exist, it will tell me NXDOMAIN. If I ask OpenDNS's recursive name server for a name that does not exist, it will send me a NOERROR response with an answer pointing at an advertising server. Note that I'm using OpenDNS as a convenient example; it did not invent this technique. Indeed, Nominum and other DNS vendors now sell an add-on to their recursive name service products to allow any ISP in the world to do this, and a growing number of ISPs are doing it. Why so many? The answer is

PHOTOGRAPH BY JOHN HEATLEY

simply whoever remaps these NXDOMAIN responses gets the impression revenue. There are unverified claims that some ISPs are blocking access to OpenDNS and/or all non-ISP name servers in order to force their customers to use the ISP's own name server. I say unverified, but I find the claims credible—ISPs have wafer-thin margins and if they see this kind of manna going out the door, they can't just let it happen.

To demonstrate the extreme desire to capture this revenue, a true story: A few years ago VeriSign, which operates the .COM domain under contract to ICANN (Internet Corporation for Assigned Names and Numbers), added a wild card to the top of the .COM zone (*.COM) so that its authoritative name servers would no longer generate NXDOMAIN responses. Instead they generated responses containing the address of SiteFinder's Web site—an advertising server. The outcry from the community (including your humble narrator) was loud and long, and before ICANN had a chance to file a lawsuit to stop this nonsense, many people had patched their recursive name servers to remap any response from a .COM name server that was not a delegation (for example, telling how to find the Google.com name servers) back into an NXDOMAIN. Some ISPs put logic into their policy-based routers to turn SiteFinder responses into pointers to the ISP's own advertising server instead.

Damage Control

NXDOMAIN wasn't designed to be a revenue hook—many applications depend on accurate error signals from DNS. For example, consider the "same origin trust model" used for Web cookies. If you're holding a cookie for Google.com and you can be fooled into following a link to KJHSDFKJHSKJHMJHER.GOOGLE.COM, and the resulting NXDOMAIN response is remapped into a positive answer to some advertising server, then you're going to send your Google.com cookie to that advertising server when you send your HTTP GET request there. Not such a bad thing for a Google.com cookie, but a real problem for a BANKOFAMERICA.COM cookie. (Thanks to Dan Kaminsky for telling me about the "same origin trust model" problem.)

Remapping could also cause email to be captured if a mail exchanger (MX) request is captured in this way. Many NXDOMAIN remappers try to avoid this by triggering only on A (address) requests, but to make this work they have to turn off caching, since NXDOMAINs are not type specific and since an SMTP initiator will fall back to type=A if it gets no answer from type=MX. Similar protections (designed to keep lawsuits away while still