The secret life of a DNS query

Igor Sviridov <sia@nest.org>
20120522

Preface

Nowadays, when we type URL (or is it a search string? ;-) into a browser (or mobile device) many things happen. While most of them are simple, over the time with iterations and generations of Internet technologies overlapping and flourishing the complexity multiplied, so that often full scope and detail is missed even from mostly qualified observer.

Let's try to uncover the secret life of DNS query.

No DNSSEC or IPv6 were harmed :)

No DNSSEC or IPv6 fairies were sacrificed while writing this presentation. Things are already convoluted without it, especially for time allotted. This is reserved for the next iteration of this talk, if any.

Query Origination

Usually we type URL in a browser, or attempt to access a non-web network service (send email, access IMAP, etc) and client DNS resolver faithfully attempts to translate a name into a transport destination (this is obviously a simplification, since multiple name lookup mechanisms may be involved).

Resolver library on client

- often part of more complex hierarchical name resolution scheme (Bonjour/mDNS, local, NIS etc)
- configured statically or dynamically (DHCP etc) with configuration parameters:
  - list of nameservers
  - explicit or implicit search suffix
  - other resolver options (ndots, timeouts, retries etc)
- resolver issues series of name resolution lookups until successful
- implements server timeouts and retries

Recursive nameserver (DNS resolver): types

- ISP provided recursive nameserver
- 3-rd party recursive nameservers (Google DNS, OpenDNS etc); filtering and augmentation/pollution
- Hijacked resolver settings - nefarious DNS servers
- Local recursive nameserver - on client host
- DNS walled gardens (esp mobile)
- Last-mile DNS spoofing (WiFi captive portals, etc)
- Legal(?!) DNS modifications on backbone level

Our query (likely with various attached suffixes) is repeatedly sent to one of above destinations until we get a positive answer - or end up with nonexistent name once all suffixes are exhausted.

OpenDNS pollution examples:
- interception/proxying of Google search (likely thing of the past now that Google switched to SSL
- replacing NXDOMAIN responses with advertisement

Google Public DNS stated policy is to never interfere or modify DNS response data
WiFi captive portals usually provide same very low TTL response to any query, sending traffic to a portal.
Walled gardens are a bit similar, but provide not a temporary but permanent DNS redirection service, for example forwarding all (or some) names to a proxy server, removing or adding parts of the namespace.
Finally, backbone operators in some countries have being requested by their governments to remove or redirect records to specific domains deemed illegal.

Recursive nameserver

- respond from cache (if it has it) - ordering and responses potentially affected by sortlist, DNS views
- perform DNS traversal talking to authoritative servers starting with root (and again using cached data); affected by:
  - forwarders configuration
  - local root cache - ICANN root zone cache and alternative roots
- anycast can be utilized to reach recursive server or authoritative servers

Anycast use for nameservers

- Anycast DNS is now in common use for both authoritative and for recursive servers; it complicates diagnosis when things go wrong
- how to determine anycast DNS server location:
  - dig @server id.server TXT chaos [RFC4892]
  - dig @server hostname.bind TXT chaos
- authoritative - how to determine anycast DNS server location:
  - dig @tld1.ultradns.net A whoareyou.ultradns.net
  - dig @opendns.server TXT which.opendns.com
  - dig any self.myresolver.info +short [@server]
  - dig +nsid +norec soa domain. @server [RFC5001]

If specific instance in Anycast is not responding, or, even worse, is returning wrong or stale data, it's often hard to pinpoint which server is providing wrong data, especially if you need to trace it via nameservers you do not control. Mechanisms

Authoritative nameservers

- responses can be static or synthesized
- can be affected by DNS views, rrset-order
- may be by DNS geographic load balancing, used CDNs/ADNs
- Interplay of CDN's and anycasted DNS; quality of mapping/discovery of client resolvers; edns-client-subnet proposal; Interplay with mobile and VPN
- Local DNS load balancing (failover)

- some CDNs/ADNs are using DNS not only to provide geographical load balancing, but also to balance between sets of local servers (or load balancers), for example to provide failover between two load balancers; they also can return ordered data
- CDN vs Anycast recursive (resolver) nameserver issue - anycast-ed nameserver has multiple locations, so CDN's cannot track it unless Anycast server sends it's queries from "local" (non-anycasted) ranges. Even worse, clients which reach wrong anycast instance (say due to VPN connection) end up populating cache with wrong (suboptimal) data, breaking it for everybody; this is quite common for Google Public DNS and, for example, Akamai. Google and Neustar have proposed EDNS extension to embed part of client IP into query; this would allow to geo-target responses based on Client (and not his resolver) IP, and even more important, to limit scope of cached DNS responses to a specific prefix.

Conclusion

DNS departed quite far from days of RFC 1035, and continues to become more complex.

When you need to debug problem with IDN name, over anycasted IPv6, protected by DNSSEC, generated by GEO-balanced CDN, things become a bit convoluted, promising us future intellectual challenges and gainful employment for years to come.

Cheers :)