Resolver AAAA Opt-in/out

Erik Kline <ek@google.com>

RIPE62

AAAAs, the DNS, and IPv6 transition

- DNS resolution of AAAAs is effectively the one and only control switch for enabling/disabling IPv6 traffic.

- RFC 3596: "*The IP protocol version used for querying resource records is independent of the protocol version of the resource records; e.g., IPv4 transport can be used to query IPv6 records and vice versa.*"
  - basically required...but it does break fate-sharing

- How to restore some semblance of fate-sharing?
  - BIND's disable-aaaa-on-v4-transport
  - draft-vandergaast-edns-client-ip
  - temporary use of "whitelisting" (access control lists)

Why use resolver ACLs?

To express the quality of working IPv6

- Fate-sharing for DNS only indicates that a ~512 byte packet wasn't dropped

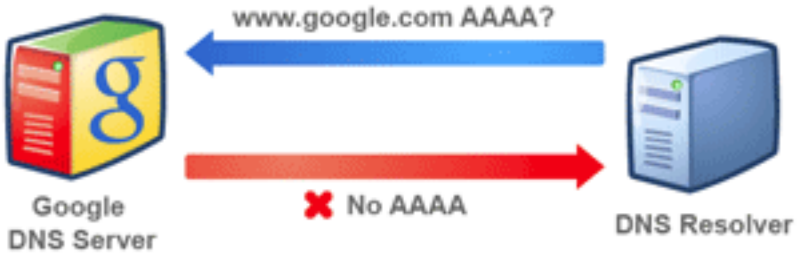
- Want users to have the best possible experience
  - what is the impact of >0.05+% of users experiencing high latency or even not reaching the site at all?

- Not all IPv6 connectivity is equal
  - an AS may have worse IPv6 redundancy than IPv4

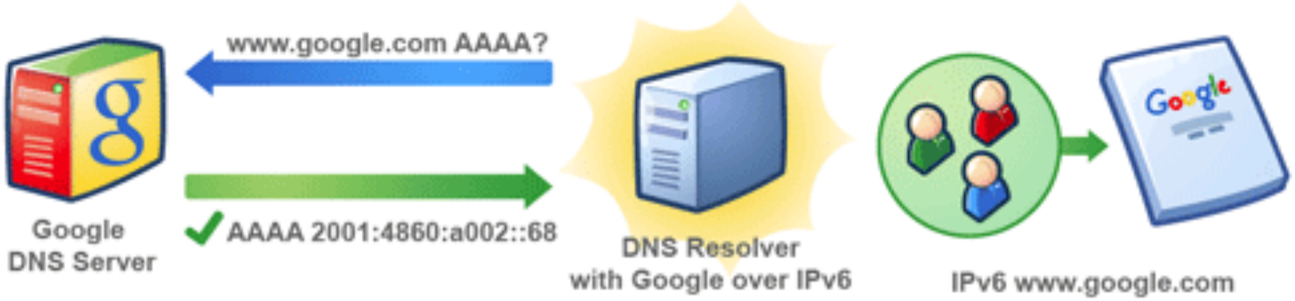
- Not all IPv6 networks are equally well supported
  - some operators may not want the IPv6 traffic (yet)

Exempli gratia

Normally, if a DNS resolver requests an IPv6 address for a Google web site, it will not receive one…



…but a DNS resolver in the Google over IPv6 "whitelist" will receive an IPv6 address, and its users will be able to connect to Google web sites using IPv6.



http://google.com/ipv6/

For each Google over IPv6 request:

1. Receive a list of resolvers or prefixes
2. Attempt to verify the requester owns/operates said prefixes
3. Convert to ASN(s), complete list of IPv4 and IPv6 prefixes
4. Verify mutual IPv6 connectivity is not worse than IPv4:
    - routing table comparison
    - look at brokenness statistics
5. Record commitment to production-quality operations
6. Possibly coordinate go-live time:
    - try to find a light traffic time
    - deal with timezone issues
    - coordinate handling of brokenness reports with NOCs
7. Possibly deal with emergency revert requests

Can we automate some of these steps?

Currently have a method that:

- can explicitly signal desire/readiness to [not] receive AAAAs
    - can also express per-AS opt-in/out
- uses "reverse DNS" delegations for loose verification of operational ownership
- optionally uses TTLs to express desired lifetimes
    - ...but operational reality may trump this
- is fairly simple, in the common case, for network operators
    - don't have to contact each AAAA provider individually

Example

For each resolver: signal readiness/desire to receive AAAAs

```
;; 192.2.0.1
_aaaa.1.2.0.192.in-addr.arpa. 1W IN TXT "ok"

;; 192.2.0.2
_aaaa.2.2.0.192.in-addr.arpa. 1W IN TXT "!ok"

;; 192.2.0.3
_aaaa.3.2.0.192.in-addr.arpa. 1W IN TXT "ok !ok=15169,32934"
```

AAAA provider-side processes

1. Log resolver IP addresses

2. Background lookups of "_aaaa.reverse DNS" names for TXT records with a specified format

3. Process and merge results into ACLs, optionally with TTLs
    - remove (or deny) formerly permitted resolvers now opting out or no longer listing TXT records (expired)
    - impact analysis of proposed new whitelist entries
        - add or discard as determined by analysis
    - update running nameservers with new config

4. GOTO 1

Limitations

- Implementation (software and processes) may be a non-trivial effort
- Compliance is not required
- Update timeliness not guaranteed
- Does not address suitability analysis phase
    - i.e. still have to review connectivity and brokenness
- Results of impact analysis still opaque to requester
    - ...and privacy requirements hamper cooperation

Thanks

ipv6whitelist.org