DNS-SD SERVICE REGISTRATION

Ted Lemon <mellon@fugue.com>
Stuart Cheshire <cheshire@apple.com>

STATUS

- Document was expired
- Update was posted prior to this IETF (-01)
- Discussion ensued on mailing list (thanks, Toke!)
- Second update, posted IETF Monday
- Call for adoption is underway
- Document is actually in pretty good shape
- Could use some review

WHAT IT DOES

- Provides a lightweight process services can use to register in the DNS
- Provides first-come, first-served protection for naming
- Provides garbage collection for
  - Claimed names (30 days? 14 days?)
  - Service registrations (2 hours?)
- Anycast single-transaction updates for constrained devices
- TCP updates for less-constrained devices

ISSUES

- This uses DNS update, but requires custom functionality
- I don't think there's a way around this that allows ad-hoc registration, which is an obvious requirement

USE OF .LOCAL

- Registrations update .local
- This is not where the registration will actually go—it will go to `dr._dns_sd.<domain>` or `x.y.z.q.in-addr.arpa` or `a.b.c.d.q.o.m.g.s.o.m.a.n.y.d.i.g.i.t.s.ip6.arpa.`
- Are we okay with this?

DOES NOT SUPPORT INTERNAL NATS

- A Registration for an IPv4 address will only be reachable if
  - the IPv4 address is global or
  - the user of the service is in the same RFC1918 routing domain
- I think this is okay

EVERY A/AAAA RECORD REQUIRES A SEPARATE REGISTRATION

- ∴ if a service wants to support dual-stack, it does two updates
- If a service has a ULA and a GUA, it has to pick, or do two updates
- Should we give advice about this? e.g.
  - If there is a ULA, use that by default
  - If configured for public access, use GUA if present
  - If only GUA present, use that?
  - What if there's more than one ULA or GUA?
- **Alternative**: let hosts update all addresses at once
  - Is that actually better?
  - What are the risks?

ONLY DNS-SD RECORDS SUPPORTED

- Very restrictive about what constitutes a Registration
- Service Name: only PTR, no delete
- Service Instance Name: only SRV and TXT
- Forward Mapping: only A or AAAA, plus required KEY
- Reverse Mapping: only PTR
- Service Name must point to Service Instance Name in update
- Service Instance Name SRV must point to Forward Mapping in update
- Reverse Mapping must point to Forward Mapping
- Benefit: we don't allow random updates
- Disadvantage: we don't allow random updates

TOKE'S CLOUD-BASED SOLUTION

- The idea is that the stateful part of the service is not on the local network
- This means that for RFC1918 addresses, IP source address validation isn't going to work end-to-end.
- To make this work, I think that you need a (mostly) stateless relay on the local network which validates the Registration and then uses TSIG or SIG(0) with its own key to do regular RFC2136-style updates to the cloud server
- Nothing technically hard about this, but do we need to specify it?

TOKE'S CLOUD SERVER, TAKE 2

- If we want public services,
  - combine this with PCP
  - cloud update points to PCP-assigned port on home router
  - which is mapped to the internal IP address of the service
  - now the service is publicly reachable
  - still requires a relay
- Do we care about this use case?
- Why not just use IPv6? :)

BACKWARDS COMPATIBILITY

- The document talks about backward compatibility
- Do we care about this?
- If so, probably needs more detail.

NEXT STEPS

- Despite being in CFA, I think document is actually nearly ready to publish
- If you don't think that, or are skeptical, please review and send comments
- I would like to move quickly with this
- What do you think?