Simple Homenet Naming Architecture

Ted Lemon <ted.lemon@nominum.com>
Daniel Migault <daniel.migault@ericsson.com>

Goals of the naming architecture

Describe how to:

- Look up names on the internet
- Publish services across the entire homenet
- Discover services anywhere on the homenet

Non-goals:

- Publish a DNS zone for the homenet in the public DNS
- Make service discovery available off the homenet
- Allow off-homenet devices to publish services on the homenet
- Secure homenet naming/service discovery using DNSSEC

Name Resolution

- Provisioning devices with resolver IP addresses
  - DHCPv4, RA+RDNSS, stateless DHCPv6
- Providing resolution
  - One or more DNS proxies that forward to the correct upstream resolver
- Unsolved problem: how to do multihoming in the presence of CDNs

Multihoming and CDNs

- Name lookups for resources stored on CDNs give different answers depending on the network connection
- Host on homenet may look up name using resolver from provider A, then connect to CDN using provider B
- This will generate support requests
- What to do?

CDN solution (proposed)

- Two kinds of hosts: multihome-fähig and not
- Network picks one ISP and advertises its prefixes to all hosts
- Hosts that are multihome-fähig get the other prefix as well
- DNS proxy for ISP A gets address in prefix from ISP A
- DNS proxy for ISP B gets address in prefix from ISP B
- Exact mechanism uncertain, probably needs RA extension
- For IPv4, we could just pick.

Provisioning Devices

- Devices signal that they are fähig using RA discovery
- Network signals extra ISPs using host-specific RAs
- DHCPv6 stateless doesn't work for multihoming
- Single-homed hosts get resolver(s) for primary ISP
- Multihomed hosts get resolver(s) for each ISP
- Resolvers are DNS proxies on the homenet, not the ISP's resolvers
- Comments from implementors would be really helpful

DNS Proxy

- Determine whether a name is local ('home.arpa') or not, and send either to local resolver or to ISP resolver
- Use source address to decide which ISP's resolvers to use for resolving names on the internet
- Could be a caching resolver, but minimal functionality requires is a DNS proxy with the ability to direct traffic

Service Discovery

- Main model is dnssd Discovery Proxy (formerly "hybrid proxy")
- Service Discovery uses split model described in draft-sctl-mdns-relay.
- Each link has its own subdomain of 'home.arpa'
- Advertising proxy updates a separate zone, 'home.arpa'
- Hosts browse all of these zones

Required infrastructure

- Every homenet router can if needed provide:
  - Discovery Broker
  - Discovery Proxy
  - Advertising Proxy
  - Discovery Relay
- Every homenet has at least one:
  - Discovery Broker
  - Discovery Proxy
  - Advertising Proxy
- Every link has at least one:
  - Discovery Relay

A typical homenet (perhaps)

home.arpa: the advertising proxy zone

link1.home.arpa: advertisements for link #1

link2.home.arpa: advertisements for link #2

link3.home.arpa: advertisements for link #3

home.arpa is listed as the only browsing zone

A service discovery lookup

host -> DNS proxy:

    _ipp._tcp.home.arpa IN PTR?

DNS proxy -> discovery broker:

    _ipp._tcp.home.arpa IN PTR?

broker -> advertising proxy:

    _ipp._tcp.home.arpa IN PTR?

broker -> discovery proxy:

    _ipp._tcp.link1.home.arpa IN PTR?
    _ipp._tcp.link2.home.arpa IN PTR?
    _ipp._tcp.link3.home.arpa IN PTR?

Answers come back

advertising proxy->broker:

```
_ipp._tcp.home.arpa IN PTR reg-printer.home.arpa
```

discovery proxy->broker:

```
_ipp._tcp.link1.home.arpa IN PTR mdns-printer.link1.home.arpa
```

broker->DNS proxy

```
_ipp._tcp.home.arpa IN PTR reg-printer.home.arpa
_ipp._tcp.link1.home.arpa IN PTR mdns-printer.link1.home.arpa
```

DNS proxy->host:

```
_ipp._tcp.home.arpa IN PTR reg-printer.home.arpa
_ipp._tcp.link1.home.arpa IN PTR mdns-printer.link1.home.arpa
```

Discovery Relay

- Very lightweight, essentially allows a discovery proxy to do mDNS on networks to which it is not directly connected
- Gives us a way to centralize mDNS to the extent we want to.
- Allows us to have multiple Discovery Proxies serving the whole homenet
- draft-sctl-mdns-relay
- https://github.com/abhayakara/dnssd-lite

Discovery Proxy

- Translates between one or more mDNS ".local" zones and their corresponding per-link zones
- Can in principle either be centralized, or one per HNR
- Centralizing gives better cache performance, which means less multicast, at least in theory
- Stateless (aside from cache)
- draft-ietf-dnssd-hybrid-proxy

Discovery Broker

- Aggregator for discovery proxies
- Allows the client to ask one question about services, and get all the answers
- Otherwise client has to browse multiple zones
- Not clear this is needed, but it's nice-to-have
- draft-sctl-discovery-broker

Advertising Proxy

- Essentially a DNS authoritative server for a zone
- But additionally provides sleep proxy functionality
- Allows DNS updates using the protocol described in draft-sctl-service-registration
- Essentially stateless--state can be recovered over time if lost
- Could in principle be replicated using zone transfers
- draft-sctl-service-registration

Status

- Everything but the multihome solution has a spec written
- Implementation of mdns-relay is complete but not in sync with current specification, will update after IETF
- Implementation of registration protocol not quite there, but Toke independently did something similar
- If WG decided this was the way to go, we could be done in fairly short order and get on to the advanced homenet naming architecture

The nsregd name registration daemon

- Implemented as a DNS Update proxy server (and client)
- Will authenticate clients (on a FCFS/TOFU basis) and the zones they update (and optionally the addresses)
- Speaks TSIG-signed update or unbound's config protocol to upstream server(s)
- Client will lookup server, get IPs from OS, and register hostname (no config required)
- ~2500 lines of Go, GPLv3. https://github.com/tohojo/nsregd

Differences from draft-sctl-service-registration-00

- Overloads TTL instead of using "Update Leases" (from draft-sekar-dns-ul-01)
- The client does another lookup step (for SRV _nsreg._tcp.<zone>)
- Server only speaks TCP (to avoid spoofing)
- No sleep proxy (Section 4)
- Server only allows A and AAAA records (and will optionally generate reverse PTRs)