draft-eckert-anima-grasp-dnssd-00

IETF’100 Singapore, November 2017

Toerless Eckert, Huawei (Futurewei Technologies USA)
tte+ietf@cs.fau.de

Scope / Goals

- Proposed as additional (milestone) doc for existing ANI charter
  - Goal of ANI was to reuse/combine existing technologies
    - Q: Why do we not use DNS-SD for service discovery ? But instead GRASP ?
    - A (author): GRASP is intended to be new transport for DNS-SD compatible service discovery
    - ... But we did not finish writing up how to do it across that transport -> this work
  - Specific ANI use-cases
    - Announce/discovery of EST server for Cert renewal (ACP draft)
    - Announce/discovery of BRSKI server (registrar) for bootstrap (BRSKI draft)
    - Announce/discovery of NOC services (stable connectivity draft)

- What is missing then with existing GRASP definitions for DNS-SD like services ?
  - GRASP can not use existing IANA service-names.
    - Those exist for e.g.: EST, BRSKI (and many services to of interest for stable connectivity).
      - No need to reinvent new names for GRASP ?
  - No definitions how multiple GRASP objectives could share common attributes
    - E.g.: DNS-SD style priority/weight for service selection
    - How to indicate if locators are reachable via ACP or data plane
    - How to selected “closest” (distance based) server (“roughly possible” only with M_DISCOVERY)

Strategy

- Separate document:
  - Remove all complex service discovery details from ACP/BRSKI.
  - Document intended to “update” BRSKI / ACP RFCs and be backward compatible.
  - One “mandatory” element to avoid duplication of administrative work:

- Objective names for IANA service names: “SRV.<service-name>”
  - <service-name> registered according to RFC6335 (service name registration)
  - Addtl. Registration via GRASP registry desirable for new services
  - mDNS<->GRASP gateway for existing DNS-SD services.

- Encoding of services params via GRASP objective-value (“payload”)
  - No GRASP header extensions == avoid incompatibility, protocol update….
  - Encoding definitions extensible / re-useable bejond services
    - Support future common cross-GRASP-objective parameters
      - Example: original-hop-count (to measure distance from sender)

From DNS-SD to GRASP

```
<service>.<prot>.<domain>   PTR <instance>.<service>.<prot>.<domain> ! <service> = RFC6335 service-name
  printer._tcp              PTR myprinter1.printer._tcp              ! Service-instance-names allow
                            PTR yourprinter2.printer._tcp            ! human selection of desired
                            PTR ourprinter3.printer._tcp             ! Instance of a service

<instance>.<service>.<prot> SRV <prio> <weight> <port> <host-name>   ! <prio> <weight> - load balancing
                            TXT key1=value1; key2=value2; . . .      ! Service specific params

<host-name>                 A/AAAA  <IPv4-address>/<IPv6-address>
```

- DNS-SD uses DNS RRs types to encode desired information
  - No need to inherit unnecessary DNS complexities (RR type structure) into GRASP – just the service information!
  - But want to be able to support gateways converting GRASP<->DNS and common high-level service announce/discover API
- Service instance names
  - Browsing by service names when client is not human but ASA ? More likely based on distance/weight and service params
  - Make service instance names optional, but support browsing ("enumeration")
- Host names
  - GRASP domains may not have or need host names, e.g.: ACP !
  - Host names not required/used in GRASP service names
  - But also provide (optional) mechanism to look up host-names via GRASP.
- Missing
  - No common way to express addresses in different VRFs (eg: ACP vs. "data-plane" addresses)
  - No way to select instance based on network distance (closer is better) – distance not intrinsic to unicast or mDNS transport.

GRASP Service structure (CBOR/CDDL)

```
service-element  =  {
        ?( &(private:0)      => any), ............. Private parameters not useful for DNS-SD
        ?( &(msg-type:1      => msg-type),......... Message Purpose: describe/enumerate (-request)
        ?( &(service:2)      => tstr),............. Service Name (“printer“)
        *( &(instance:3)     => tstr),............. Instance Name („my-kitchen-printer“)
        ?( &(domain:4)       => tstr),............. Empty = .local (e.g.: ACP), else name
        ?( &(priority:5)     => 0..65535 ),........ As in DNS-SD
        ?( &(weight:6)       => 0..65535 ),........ As in DNS-SD
        *( &(kvpairs:7)      => { *(tstr: any) },.. Key Value pairs – as in DNS-SD
        ?( &(range:8)        => 0..255 ),.......... Controls distance or priority/weight selection
        *( &(clocator:9)     => clocator),......... GRASP locators with context indicator (“VRF”)
       }

   clocator = [ context, locator-option ] ......... Permit locators to be in data plane
   context = tstr ................................. Empty: ACP, „0“ = „VRF0“, else name of VRF
   locator-option = <unchanged> ................... from GRASP specification – addr/port

   msg-type = &( describe: 0, describe-request:1, enumerate:2, enumerate-request:3 ).
```

GRASP exchanges:

- GRASP M_FLOOD == unsolicited announcement of objective == service instance (GRASP flooded)
  - msg-type: “describe”

- GRASP M_DISCOVERY = find an objective == service instance (GRASP flooded)
  - msg-type: “describe-request”
  - Reply: GRASP M_REQ_SYN with msg-type: “describe” (unicast)
  - Flooding of request stops at first found objective providers (standard GRASP behavior)

- Describe/describe-request also useable in any unicast GRASP negotiations

- Msg-type “enumerate”, “enumerate-request”:
  - Do not provide clocators of instances (as ”describe” does), but only instance names (to support “browsing” as in DNS-SD)
  - This is then followed by a second round of “describe-request” – unicasted to originator of “enumerate”

- Backward compatibility with existing BRSKI/ACP definitions:
  - GRASP SRV.<service-name> objective without service-element in objective-value (including no objective-value at all)
  - Same as msg-type “describe”, clocator is the locator from the GRASP message header, weight/priority at default values

Common objective-value elements

```
objective-value      /= { 1*elements }
   elements         //= ( @rfcXXXX: { 1*relement } )
   relement           = ( relement-codepoint => relement-value )
   relement-codepoint = uint
   relement-value     = any

relement-codepoint //= ( &(sender-loop-count:1) => 1..255 )
relement-codepoint //= ( &(srv-element:2)            => service-element )
```

- If an objective wants to use reuseable elements:
  - Objective value must be a map. Reuseable elements use a well-known key in the map ("rfcXXXX")
- Reuseable elements have IANA assigned codepoint (and sepcification)
- Two reuseable elements defined:
  - Service element
  - Sender-loop-count (to enable distance from sender recognition in M_FLOOD / M_DISCOVERY)

Name resolution:

- Objective names: NAME.<hostname>

  - <hostname> as in DNS hostnames (without domain)
  - Uses same GRASP service structure, just most elements defined to be unused.
  - Allows to discover devices by their name
  - Objective names of this type are not to be IANA registered

  - Usefulness: TBD (opinions welcome)
    - Very much depending on size of GRASP domain and frequency of name loookups required
    - Quite useful for network administration diagnostics
      - Reminder: primary scope of GRASP users is network protocols / OAM , not end-user!
  - Example:
    - Typically infra equipment (router, switches,..) in a network have hostnames.
    - These should be in DNS… and they are.. in well organized networks (meaning: quite often not 100% consistent)
    - How do you find a device by name if they are not ?