

ANIMA DNS-SD compatible services auto configuration

draft-eckert-anima-grasp-dnssd-02

draft-eckert-anima-services-dns-autoconfig-00

IETF111 July 2021

Toerless Eckert (Futurewei USA), tte@cs.fau.de

Mohamed Boucadair, mohamed.boucadair@orange.com

Christian Jacquenet, christian.jacquenet@orange.com

Michael H. Behringer, michael.h.behringer@gmail.com

Background

- Work already published around IETF100/IETF101, presentend to ANIMA, DNS-SD.
- Author let it expire due to dependencies, priorities
 - Needed to finish ANIMA charter round 1 work (8 RFC, 420 pages, published May 2021).
- Now reviving the work, adding new co-authors, only minor textual improvements (more work to come)
- Editorial change:
 - draft-eckert-anima-noc-autoconfig changed to draft-eckert-anima-services-dns-autoconfig after discussion with Ignas – term NOC not commonly the correct term in some networks

draft-eckert-anima-services-dns-autoconfig

The core stuff we are missing for RFC8368 managing existing network via ACP

- Most designs would look into just defining what is necessary for minimal bootstrap and enable remote operator
- Proposal here is to do that stuff in a way that it continues to provide value not only for day 1 but through lifecycle, and support existing ANI ASA
 - BRSKI EST.
- Aka: automate the necessary and obvious, expecting no good automation in NOC
 - Aka: many if not most networks still do a lot of infrastructure provisioning manually.

What do we need ?

- How does NOC learn about newly enrolled devices ?
 - Do not assume that there is great integrated automation between BRSKI enrollment, and whoever needs to act next.
 - One simple answer is syslog of core device status. As soon as it can connect to ACP. And syslog is valuable through lifecycle as well.
- How do we make sure certificates work and get renewed correctly
 - We need working clocks. NTP is the standard service. Aka: autoconfigure NTP
- How does the operator remotely access the network devices
 - SSH for CLI, NetConf via SSH or TLS for automated systems. NetConf ZeroTouch (call home) also an option.
 - Can easily be auto started when node starts having running ACP
- How is operator authorized to access device ?
 - Certificates do not convey privileges.
 - Auto-discover auto/use user authentication for SSH, NetConf: Radius, Diameter, Tacacs+
 - Device may only need to support one such protocol and will use the one it prefer, choice of preference can be product specific ?!

How to make it work ?

- How to discover NTP, syslog, radius, diameter, tacacs+ servers
When multiple (redundancy, failover, maybe site-local syslog server) are available, how to select ?
- DNS-SD is a great framework that can be used, it is designed for this
- But how to autoconfigure DNS-SD across an ACP network ?
 - Do not want third-party dependencies (DNS-SD servers)
- Define GRASP objective(s) for DNS-SD compatible service discovery
- Servers announce DNS-SD services with parameters, all routers listen and autoconfigure the services client (NTP, syslog, authentication) side they need/want
- Selection of best server via DNS standard prio/weight parameters, extending them with distance and maybe more

What else ?

- Scheme can of course be extended to any other auto configured widely used service
- Firmware update ? “Yuck TFTP” / http (need to discuss with SUIT WG).
- Do operators like IPv6 addresses ? Autoconfigure DNS for OAM so CLI can do name resolution (but make nothing in ANI depend on those names)
- Services may require too much manual config to fully autoconfigure with easy recipes, but “server” could still be configured to be auto-discovered via ACP DNS-SD
 - Example: monitoring via IPfix/Netflow, DHCP parameters for user devices (DHCP server), SNMP traps, Yang/NetConf-push (similar to syslog)

draft-eckert-anima-grasp-dnssd

Autonomic Control Plane (ACP, RFC8994)

ACP is an autonomically built, hop-by-hop IPsec encrypted IPv6 “VRF-lite” across all router/switches (nodes) in a network

Runs even when nothing is configured or anything else is broken (not mis-configurable by operator/SDN controllers).

Primarily intended for OAM, automated network management, M2M communications.

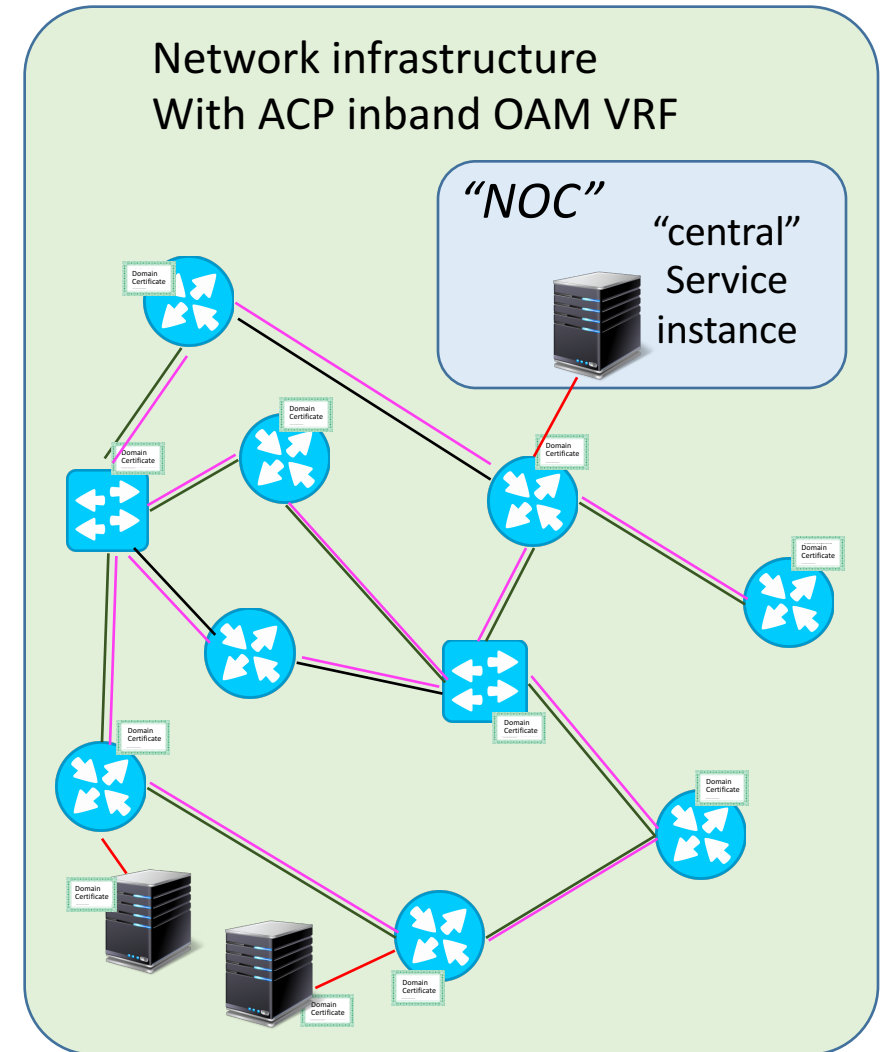
ACP IPv6 address is name of node: Automatically generated during onboarding of node, attached to node ACP certificate.

Therefore no need for other node names (e.g.: DNS name)

BUT: ACP needs service discovery.

Primary services of interest: infrastructure service instances used on many/all nodes to autoconfigure relevant services for nodes themselves and/or network users:

Servers for protocols such as NetConf, ZeroTouch, TFTP, Diameter, Radius, NTP, PTP, syslog, SNMP trap, DHCP, DNS, Ipfix collector,...



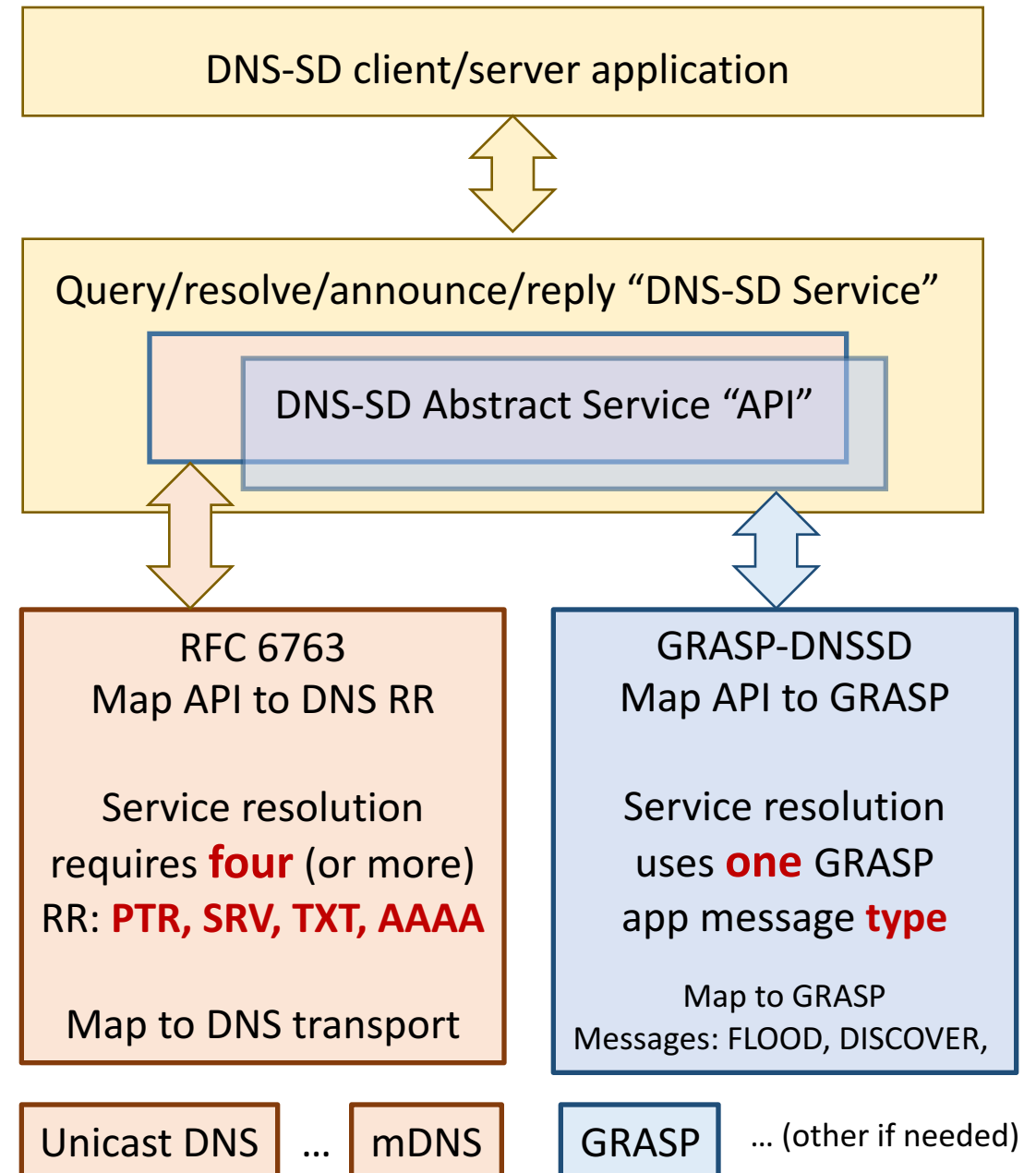
— ACP secure virtual connections
With GRASP signaling
ACP: Autonomic Control Plane

Service announcement/discovery for network infrastructure services

- DNS-SD unicast not feasible / ideal
 - Infeasible: No other system to discover DNS-SD servers. We are talking about the root service discovery system in networks.
 - Not ideal: Do not want a limited-redundancy, third party dependency between service announcer and consumer. Also difficult to autoconfigure a sufficient set of DNS servers on routers/switches (limited functionality nodes).
- Network wide flooding/multicast is great for our purpose
 - Limited number of services, no third party dependency, needed on most/every node anyhow
- mDNS multihop flooding never standardized AFAIK
 - Would also limit message transport to only DNS messages/transactions.
 - And we (ANIMA) wanted something generic/extensible
 - *Past commercial attempts showed challenges building reliable multihop flooding issues using DNS messages themselves to build loop-free flooding*

DNS-SD “service” API

- No explicit IETF specification ?
 - Maybe this is a good work item too.
 - But many concrete, OS APIs.
- Ideally would like API to be able to choose whether to use RFC6763 or GRASP-DNSSD (or both).
 - Some concrete API may be too explicit, asking user to even select transport (mDNS, Unicast).
- May not be 100% match
 - Some unicast name related functions not possible (TBD)
 - Some extensions very useful and only easily possible with GRASPs approach
 - Flooding allows to introduce “distance” as service selection parameter (DNS-SD only has priority/wight, independent of distance to service instance – based on unicast DNS).



DNS-SD compatible GRASP objectives (CDDL):

objective-name // = SRV.<rfc6335-service-name>

objective-name // = NAME.<hostname> If we ever need them

service-element = {

?(&(private:0) => any), Non-standardized extensions
?(&(msg-type:1) => msg-type
?(&(service:2) => tstr), Service Name ("printer")
*(&(instance:3) => tstr), Instance Name („my-kitchen-printer“)
?(&(domain:4) => tstr), Empty = ANI/ACP (like .local), else VRF 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), For distance based service 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 – IPv4/IPv6addr/port

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

No need for duplicate “services” registry

- Define a DNS-SD GRASP application/objective group for DNS-SD compatible service announcement/discovery
- Want to reuse as much as we can from DNS-SD concepts:
- Services namespace, parameters, semantic
- This started because we started to define our own GRASP services (objectives) IANA registry (IMHO for many cases unnecessarily)
- Instead, draft would result in DNS service names to be reuseable across GRASP
- Example with existing service “est”
 - mDNS: `_est._udp.local`
 - GRASP: `SRV.est`
 - When this work becomes RFC, `SRV.xxx` names would not need to be registered explicitly for GRASP anymore instead, they will have the same semantic as in the DNS service name registry.

GRASP Objective Names

Registration Procedure(s)

Specification Required

Expert(s)

Michael Richardson

Reference

[\[RFC8990\]](#)

Available Formats



CSV

Objective Name	Reference
EX0	[RFC8990]
EX1	[RFC8990]
EX2	[RFC8990]
EX3	[RFC8990]
EX4	[RFC8990]
EX5	[RFC8990]
EX6	[RFC8990]
EX7	[RFC8990]
EX8	[RFC8990]
EX9	[RFC8990]
PrefixManager	[RFC8992]
PrefixManager.Params	[RFC8992]
AN_ACP	[RFC8994]
SRV.est	[RFC8994]
AN_Proxy	[RFC8995]
AN_join_registrar	[RFC8995]

<https://www.iana.org/assignments/grasp-parameters/grasp-parameters.xhtml#objective-names>

Expanding Applicability beyond ACP

- Easily defined to be reuseable for any existing network
 - Work not specified, might be a separate GRASP draft
- GRASP forwarding agent is very simple, limited code
 - Prototype code from Brian Carpenter (python)
 - <https://github.com/becarpenter/graspy/blob/master/AskDNSSD2.py>
 - <https://github.com/becarpenter/graspy/blob/master/GetDNSSD2.py>
 - <https://github.com/becarpenter/graspy/blob/master/graspy.pdf>
- Instead of using (PKI + IPsec secured) ACP, one can define more lightweight options:
 - No new config for operator.
 - E.g.: start GRASP automatically to every node known to be an IGP peer with GRASP over TCP connection
 - Or even TLS with same credentials as used for the IGP (if IGP uses credentials).
 - This is “secure enough”, because the IGP must be secured, as it is the biggest, easy attack vector against a network infrastructure.
 - “Clamshell” ACL protection to limit who can be an IGP neighbor (trusted internal interfaces) is common, and if it is acceptable for IGP, it is also acceptable for GRASP.

The End

- Questions
- Backup slides following

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

draft-eckert-anima-services-dns-autoconfig

- Explicitly describes the services to autoconfigure
- NTP, syslog,
- radius, diameter
 - Servers announce themselves, all ACP nodes autoconfigure themselves to use the “best” server they see via GRASP-DNSSD for authentication of any services across the ACP that require user authentication
- SSHD / NetConf
 - If/when radius and/or diameter servers are discovered, ACP routers start SSHD / NetConf via TLS) and allow user-authentication via radius/diameter
- Syslog, NTP
 - Servers announce themselves, client autoconfigure NTP, syslog when they see such service announcements
 - These services may benefit from actually interacting in parallel with not only one-best server, but multiple. How do we indicate this (extension of DNS-SD parameters ?)
- DNS
 - Even though ACP itself should not depend on normal DNS names, operators may want to rely on DNS when using CLI, and/or router show commands may want to resolve DNS names. So this is an option to announce DNS servers with the view of the DNS to be used for OAM on the ACP nodes.
 - Routers may or may not provide different DNS views for e.g.: users and/or OAM, so this needs to be refined in text.