ANIMA WG
Internet-Draft
Updates: 4291,4193 (if approved)
Intended status: Standards Track
Expires: February 3, 2018

M. Behringer, Ed.

T. Eckert, Ed.
Huawei

S. Bjarnason
Arbor Networks

August 2, 2017

An Autonomic Control Plane (ACP)
draft-ietf-anima-autonomic-control-plane-09

Abstract

Autonomic functions need a control plane to communicate, which depends on some addressing and routing. This Autonomic Control Plane should ideally be self-managing, and as independent as possible of configuration. This document defines an "Autonomic Control Plane", with the primary use as a control plane for autonomic functions. It also serves as a "virtual out of band channel" for OAM communications over a network that is not configured, or mis-configured.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 3, 2018.

Copyright Notice

Table of Contents

1. Introduction

Autonomic Networking is a concept of self-management: Autonomic functions self-configure, and negotiate parameters and settings across the network.[RFC7575] defines the fundamental ideas and design goals of Autonomic Networking. A gap analysis of Autonomic Networking is given in [RFC7576]. The reference architecture for Autonomic Networking in the IETF is currently being defined in the document [I-D.ietf-anima-reference-model]

Autonomic functions need a stable and robust infrastructure to communicate on. This infrastructure should be as robust as possible, and it should be re-usable by all autonomic functions.[RFC7575] calls it the "Autonomic Control Plane". This document defines the Autonomic Control Plane.

Today, the management and control plane of networks typically runs in the global routing table, which is dependent on correct configuration and routing. Misconfigurations or routing problems can therefore disrupt management and control channels. Traditionally, an out of band network has been used to recover from such problems, or personnel is sent on site to access devices through console ports (craft ports). However, both options are operationally expensive.

In increasingly automated networks either controllers or distributed autonomic service agents in the network require a control plane which is independent of the network they manage, to avoid impacting their own operations.

This document describes options for a self-forming, self-managing and self-protecting "Autonomic Control Plane" (ACP) which is inband on the network, yet as independent as possible of configuration, addressing and routing problems (for details how this achieved, see Section 6). It therefore remains operational even in the presence of configuration errors, addressing or routing issues, or where policy could inadvertently affect control plane connectivity. The Autonomic Control Plane serves several purposes at the same time:

- Autonomic functions communicate over the ACP. The ACP therefore supports directly Autonomic Networking functions, as described in [I-D.ietf-anima-reference-model]. For example, GRASP

[I-D.ietf-anima-grasp] runs securely inside the ACP and depends on the ACP as its "security substrate".

- An operator can use it to log into remote devices, even if the data plane is misconfigured or unconfigured.

- A controller or network management system can use it to securely bootstrap network devices in remote locations, even if the network in between is not yet configured; no data-plane dependent bootstrap configuration is required. An example of such a secure bootstrap process is described in [I-D.ietf-anima-bootstrapping-keyinfra]

This document describes some use cases for the ACP in Section 3, it defines the requirements in Section 4, Section 5 gives an overview how an Autonomic Control Plane is constructed, and in Section 6 the detailed process is explained.Section 8 explains how non-autonomic nodes and networks can be integrated, and Section 6.7 the first channel types for the ACP.

The document "Autonomic Network Stable Connectivity" [I-D.ietf-anima-stable-connectivity] describes how the ACP can be used to provide stable connectivity for OAM applications. It also explains on how existing management solutions can leverage the ACP in parallel with traditional management models, when to use the ACP versus the data plane, how to integrate IPv4 based management, etc.

2. Terminology

This document uses the following terms (sorted alphabetically):

ACP "Autonomic Control Plane". The Autonomic Function defined in this document. It provides secure zero-touch network wide IPv6 connectivity between devices supporting it. The ACP is primarily meant to be used as a component of the ANI to enable Autonomic Networks but it can equally be used in simple ANI networks (with no other Autonomic Functions) or completely by itself.

ACP address An IPv6 ULA address assigned to the ACP device. It is stored in the ACP information field of an ACP devices LDevID.

ACP connect A physical interface on an ACP device providing access to the ACP for non ACP capable devices. See Section 8.1.1.

ACP device A device supporting the ACP according to this document.

ACP information (field) An rfc822Name information element (eg: field) in the Domain Certificate in which the ACP relevant information is encoded: the AN Domain Name and the ACP address.

ACP (loopback) interface The loopback interface in the ACP VRF that hosts the ACP address.

ACP (ULA) prefix(es) The prefixes routed across the ACP. In the normal/simple case, the ACP has one ULA prefix, see Section 6.10. The ACP routing table may include multiple ULA prefixes if the "rsub" option is used to create addresses from more than one ULA prefix. See Section 6.1.1. The ACP may also include non-ULA prefixes if those are configured on ACP connect interfaces. See Section 8.1.1.

ACP secure channel A virtual subinterface/tunnel established hop-by-hop between adjacent ACP devices to carry traffic of the ACP VRF separated from Data Plane traffic inband over the same links as the Data Plane.

ACP secure channel protocol The protocol used to build an ACP secure channel, eg: IKEv2/IPsec or dTLS.

ACP virtual interface An interface in the ACP VRF mapped to one or more ACP secure channels. See Section 6.12.4.

ACP VRF The ACP is modelled in this document as a "Virtual Routing and Forwarding" (VRF) component in a network device.

AN "Autonomic Network". A network according to [I-D.ietf-anima-reference-model]. Its main components are Intent, Autonomic Functions and ANI.

AN Domain Name An FQDN (Fully Qualified Domain Name) identifying an Autonomic Network. It is stored in the ACP information field of an ANI devices LDevID. See Section 6.1.1.

ANI (device/network) "Autonomic Network Infrastructure". A device with ANI supports ACP, BRSKI and GRASP. The ANI is the infrastructure to enable autonomic functions. An ANI network or device is a most basic Autonomic Network or device: it does not need to have ASAs other than the ones implementing ACP, BRSKI and GRASP nor Intent support. A simple ANI network (without further autonomic functions) can for example support secure zero touch bootstrap and stble connectivity for SDN networks - see [I-D.ietf-anima-stable-connectivity].

ANIMA "Autonomic Networking Integrated Model and Approach". ACP, BRSKI and GRASP are work products of ANIMA.

ASA "Autonomic Service Agent". Autonomic software modules running on an ANI device. The components making up the ANI (BRSKI, ACP, GRASP) are also described as ASAs.

Autonomic Function A function/service in an Autonomic Network (AN) composed of one or more ASA across one or more ANI Devices.

BRSKI "Bootstrapping Remote Secure Key Infrastructures" ([I-D.ietf-anima-bootstrapping-keyinfra]. A protocol extending EST to enable secure zero touch bootstrap in conjunction with ACP. ANI devices use ACP and BRSKI.

Data Plane The counterpoint to the ACP in an ACP device: all VRFs other than the ACP. In a simple ACP or ANI device, the Data Plane is typically provisioned non-autonomic, for example manually (including across the ACP) or via SDN controllers. In a full Autonomic Network Device, the Data Plane is managed autonomically via Autonomic Functions and Intent.

Domain Certificate An LDevID with an information element defined in this document used by the ACP to derive and cryptographically assert its membership in an autonomic domain.

enrollment The process where a device presents identification (for example through keying material such as the private key of an IDevID) to a network and acquires a network specific identity and trust anchor such as an LDevID.

EST "Enrollment over Secure Transport" ([RFC7030]). IETF standard protocol for enrollment of a device with an LDevID. BRSKI is based on EST.

GRASP "Generic Autonomic Signaling Protocol". An extensible signaling protocol required by the ACP for ACP neighbor discovery. The ACP also provides the "security and transport substrate" for the "ACP instance of GRASP" which is run inside the ACP to support BRSKI and other future Autonomic Functions. See [I-D.ietf-anima-grasp].

IDevID An "Initial Device IDentity" X.509 certificate installed by the vendor on new equipment. Contains information that establishes the identity of the device in the context of its vendor/manufacturer such as device model/type and serial number.

LDevID A "Local Device IDentity" X.509 certificate installed during an "enrollment". The ACP depends on a Domain Certificate which is an LDevID.

MIC "Manufacturer Installed Certificate". Another word not used in this document to describe an IDevID.

RPL "IPv6 Routing Protocol for Low-Power and Lossy Networks". The routing protocol used in the ACP.

MASA (service) "Manufacturer Authorized Signing Authority". A vendor/manufacturer or delegated cloud service on the Internet used as part of the BRSKI protocol.

sUDI "secured Unique Device Identifier". Another term not used in this document to refer to an IDevID.

UDI "Unique Device Identifier". In the context of this document unsecured identity information of a device typically consisting of at least device model/type and serial number, often in a vendor specific format. See sUDI and LDevID.

ULA "Unique Local Address". The IPv6 equivalent to RFC1918 IPv4 addresses. ACP addresses are ULA.

3. Use Cases for an Autonomic Control Plane

3.1. An Infrastructure for Autonomic Functions

Autonomic Functions need a stable infrastructure to run on, and all autonomic functions should use the same infrastructure to minimise the complexity of the network. This way, there is only need for a single discovery mechanism, a single security mechanism, and other processes that distributed functions require.

3.2. Secure Bootstrap over an Unconfigured Network

Today, bootstrapping a new device typically requires all devices between a controlling node (such as an SDN controller) and the new device to be completely and correctly addressed, configured and secured. Therefore, bootstrapping a network happens in layers around the controller. Without console access (for example through an out of band network) it is not possible today to make devices securely reachable before having configured the entire network between.

With the ACP, secure bootstrap of new devices can happen without requiring any configuration on the network. A new device can automatically be bootstrapped in a secure fashion and be deployed

with a domain certificate. This does not require any configuration on intermediate nodes, because they can communicate through the ACP.

3.3. Data Plane Independent Permanent Reachability

Today, most critical control plane protocols and network management protocols are running in the data plane (global routing table) of the network. This leads to undesirable dependencies between control and management plane on one side and the data plane on the other: Only if the data plane is operational, will the other planes work as expected.

Data plane connectivity can be affected by errors and faults, for example certain AAA misconfigurations can lock an administrator out of a device; routing or addressing issues can make a device unreachable; shutting down interfaces over which a current management session is running can lock an admin irreversibly out of the device. Traditionally only console access can help recover from such issues.

Data plane dependencies also affect NOC/SDN controller applications: Certain network changes are today hard to operate, because the change itself may affect reachability of the devices. Examples are address or mask changes, routing changes, or security policies. Today such changes require precise hop-by-hop planning.

The ACP provides reachability that is largely independent of the data plane, which allows control plane and management plane to operate more robustly:

- For management plane protocols, the ACP provides the functionality of a "Virtual-out-of-band (VooB) channel", by providing connectivity to all devices regardless of their configuration or global routing table.

- For control plane protocols, the ACP allows their operation even when the data plane is temporarily faulty, or during transitional events, such as routing changes, which may affect the control plane at least temporarily. This is specifically important for autonomic service agents, which could affect data plane connectivity.

The document "Autonomic Network Stable Connectivity" [I-D.ietf-anima-stable-connectivity] explains the use cases for the ACP in significantly more detail and explains how the ACP can be used in practical network operations.

4. Requirements

The Autonomic Control Plane has the following requirements:

ACP1: The ACP SHOULD provide robust connectivity: As far as possible, it should be independent of configured addressing, configuration and routing. Requirements 2 and 3 build on this requirement, but also have value on their own.

ACP2: The ACP MUST have a separate address space from the data plane. Reason: traceability, debug-ability, separation from data plane, security (can block easily at edge).

ACP3: The ACP MUST use autonomically managed address space. Reason: easy bootstrap and setup ("autonomic"); robustness (admin can't mess things up so easily). This document suggests to use ULA addressing for this purpose.

ACP4: The ACP MUST be generic. Usable by all the functions and protocols of the AN infrastructure. It MUST NOT be tied to a particular protocol.

ACP5: The ACP MUST provide security: Messages coming through the ACP MUST be authenticated to be from a trusted node, and SHOULD (very strong SHOULD) be encrypted.

The default mode of operation of the ACP is hop-by-hop, because this interaction can be built on IPv6 link local addressing, which is autonomic, and has no dependency on configuration (requirement 1). It may be necessary to have ACP connectivity over non-autonomic nodes, for example to link autonomic nodes over the general Internet. This is possible, but then has a dependency on routing over the non-autonomic hops (see Section 8.2).

5. Overview

The Autonomic Control Plane is constructed in the following way (for details, see Section 6):

1. An autonomic node creates a virtual routing and forwarding (VRF) instance, or a similar virtual context.

2. It determines, following a policy, a candidate peer list. This is the list of nodes to which it should establish an Autonomic Control Plane. Default policy is: To all adjacent nodes in the same domain.

3. For each node in the candidate peer list, it authenticates that node and negotiates a mutually acceptable channel type.

4. It then establishes a secure tunnel of the negotiated channel type. These tunnels are placed into the previously set up VRF. This creates an overlay network with hop-by-hop tunnels.

5. Inside the ACP VRF, each node sets up a virtual (loopback) interface with its ULA IPv6 address.

6. Each node runs a lightweight routing protocol, to announce reachability of the virtual addresses inside the ACP.

Note:

- Non-autonomic NMS systems or controllers have to be manually connected into the ACP.

- Connecting over non-autonomic Layer-3 clouds initially requires a tunnel between autonomic nodes.

- None of the above operations (except manual ones) is reflected in the configuration of the device.

The following figure illustrates the ACP.

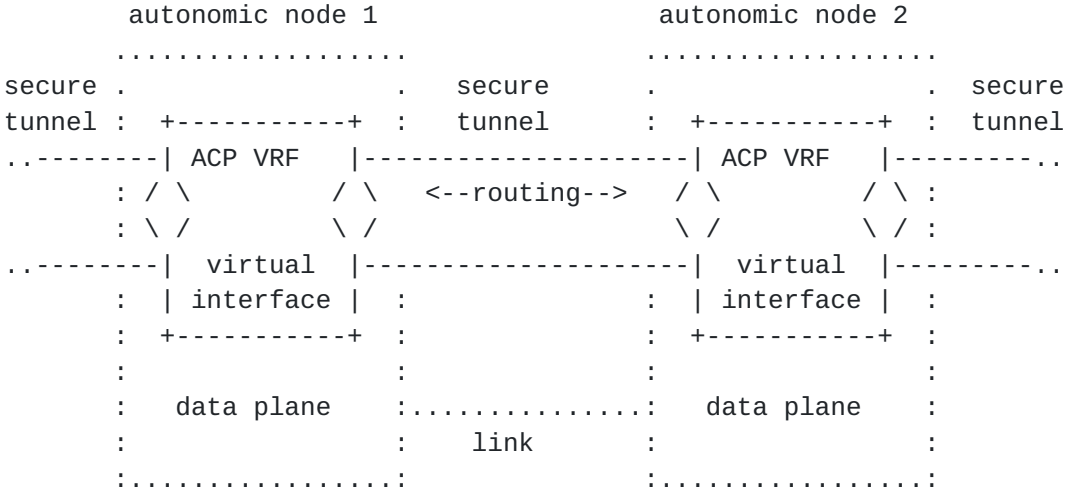

Figure 1

The resulting overlay network is normally based exclusively on hop-by-hop tunnels. This is because addressing used on links is IPv6 link local addressing, which does not require any prior set-up. This way the ACP can be built even if there is no configuration on the

devices, or if the data plane has issues such as addressing or routing problems.

6. Self-Creation of an Autonomic Control Plane (ACP) (Normative)

This section describes the steps to set up an Autonomic Control Plane (ACP), and highlights the key properties which make it "indestructible" against many inadvert changes to the data plane, for example caused by misconfigurations.

An ACP device can be a router, switch, controller, NMS host, or any other IP device. Initially, it must have a globally unique domain certificate (LDevID), as well as an adjacency table. It then can start to discover ACP neighbors and build the ACP. This is described step by step in the following sections:

6.1. Domain Certificate

To establish an ACP securely, an ACP device MUST have a globally unique domain certificate (LDevID), with which it can cryptographically assert its membership in the domain as well as the CA certificate chain used to sign the LDevID. This CA certificate chain is used to verify the LDevID of candidate ACP peers (see Section 6.6). The ACP does not mandate specific mechanism by which this certificate information is provisioned onto the ACP device, it only requires the following ACP specific information field in its LDevID as well as the LDevIDs of candidate ACP peers. See Section 10.1 for more information about enrollment or provisioning options.

6.1.1. ACP information

The domain certificate (LDevID) of an autonomic node MUST contain ACP specific information, specifically the domain name, the address of the device in the ACP with the Zone-ID set to zero ("ACP address"). This information MUST be encoded in the LDevID in the subjectAltName / rfc822Name field in the following way:

anima.acp+<acp-address>{+<rsub>{+<extensions>}}@<domain>

Example:

anima.acp+fda379A6f6ee00000200000064000001+area51.research@acp.example.com

The acp-address MUST be specified as a string of 32 hex characters with only lower letters a-f and numbers 0-9 so that the local part of

the address can matches the simple dot-atom format of [RFC5322] (":" are not allowed in that format).

<domain> is used to indicate the Autonomic Domain across which all ACP nodes trust each other and are willing to build ACP channel with each other. See Section 6.6. It SHOULD be the FQDN of a domain owned by the operator assigning the certificate. This is a simple method to ensure that the domain name is globally unique and collision of ACP addresses would therefore only happen due to ULA hash collisions. If the operator does not own any FQDN, it should choose am FQDN format string that intends to be equally unique.

{<rsub>.}<domain> is the autonomic "routing subdomain" that is used used in addressing to calculate the hash used in the creation of the ACP address of the device. As the name implies, every routing subdomain is also a separate routing subdomain. <rsub> is optional and should only used when its impacts are understood. The domain without any leading rsub field is also just another routing subdomain.

The optional <extensions> field is used for future extensions to this specification. It MUST be ignored if present and not understood.

The subjectAlName / rfc822Name encoding of the ACP domain name and ACP address is used for the following reasons:

- There are a wide range of pre-existing protocols/services where authentication with LDevID is desirable. Enrolling and maintaining separate LDevIDs for each of these protocols/services is often undesirable overhead. Therefore, the information element required for the ACP in the domain certificate should be encoded in a way that minimizes the possibility of creating incompatibilites with such other uses beside the authentication for the ACP.

- The elements in the LDevID required for the ACP should not cause incompatibilities with any pre-existing ASN.1 software potentially in use in those other pre-existing SW systems. This eliminates the use of novel information elements because those require extensions to those pre-existing ASN.1 parsers.

- subjectAltname / rfc822Name is a pre-existing element that must be supported by all existing ASN.1 parsers for LDevID.

- The elements in the LDevID required for the ACP should also not be misinterpreted by any pre-existing protocol/service that might use the LDevID. If the elements used for the ACP are interpreted by other protocols/services, then the impact should be benign.

- Using an IP address format encoding could result in non-benign misinterpretation of the ACP information, for example other protocol/services unaware of the ACP could try to do something with the ACP address that would fail to work correctly. For example, the address could be interpreted to be an address of the device in a VRF other than the ACP VRF.

- At minimum, both the AN domain name and the non-domain name derived part of the ACP address need to be encoded in one or more appropriate fields of the certificate, so there are not many alternatives with pre-existing fields where the only possible conflicts would likely be beneficial.

- rfc822Name encoding is quite flexible. We choose to encode the full ACP address AND the domain name with sub part into a single rfc822Name information element it, so that it is easier to examine/use the encoded "ACP information (field)".

- The format of the rfc822Name is choosen so that an operator can set up a mailbox called anima.acp@<domain> that would receive emails sent towards the rfc822Name of any AN device inside a domain. This is possible because components behind a plus symbol are considered part of a single mailbox. In other words, it is not necessary to set up a separate mailbox for every autonomic devices ACP information field, but only one for the whole domain.

- In result, if any unexpected use of the ACP addressing information in a certificate happens, it is benign and detectable: it would be mail to that mailbox.

See section 4.2.1.6 of [RFC5280] for details on the subjectAltName field.

6.1.2. Maintenance

The ACP network MUST have one or more nodes that support EST server ([RFC7030] functionality (eg: as an ASA) through which ACP nodes can renew their domain certificate. The ACP address of at least one such EST server SHOULD have been enrolled/provisioned into the ACP device during initial installation of the domain certificate.

EST servers MUST announce their service via GRASP in the ACP through M_FLOOD messages:

Example:

```
[M_FLOOD, 12340815, h'fda379a6f6ee0000200000064000001', 180000,
    ["AN_join_registrar", SYNCH-FLAG, 255, "EST-TLS"],
    [O_IPv6_LOCATOR,
         h'fda379a6f6ee0000200000064000001', TCP, 80]
]
```

The formal CDDL definition is:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]

objective = ["AN_join_registrar", objective-flags, loop-count,
                                       objective-value]

objective-flags = SYNCH-FLAG ; as in GRASP spec
loop-count      = 255        ; mandatory maximum
objective-value = text       ; name of the (list of) of supported
                             ; protocols: "EST-TLS" for RFC7030.
```

The M_FLOOD message MUST be sent periodically. The period is subject to network administrator policy (EST server configuration). It must be so low that the aggregate amount of periodic M_FLOODs from all EST servers causes negligible traffic across the ACP.

In the above (recommended) example the period could be 60 seconds and the indicated ttl of 180000 msec means that the objective would continuously be cached by ACP devices even when two out of three messages are dropped in transit (which is unlikely because GRASP hop-by-hop forwarding is realiable).

The locator-option indicates the ACP transport address for the EST server. The loop-count MUST be sete to 255. When an ACP node receives the M_FLOOD, it will have been reduced by the number of hops from the EST server.

When it is time for domain certificate reneal, the ACP device MUST attempt to connect to the EST server(s) learned via GRASP starting with the one that has the highest remaining loop-count (closest one). If certificate renewal does not succeed, the device MUST attempt to use the EST server(s) learned during initial provisioning/enrollment of the certificate. After successful renewal of the domain certificate, the ACP address from the certificate of the EST server (as learned during the TLS handshake) is added to the top of the list or provisioned/configured EST-server(s).

This logic of selecting an EST server for renewal is choosen to allow for distributed EST servers to be used effectively but to also allow fallback to the most reliably learned EST server - those that performed already successful enrollment in before. A compromised (non EST-server) ACP device for example can filter or fake GRASP announcements, but it can not successfully renew a certificate and can only prohibit traffic to a valid EST server when it is on the path between the ACP device and the EST server.

The ACP device MUST support Certificate Revocation Lists via HTTPs from one or more Certificate Distribution Points. These CDPs MUST be indicated in the Domain Certificate when used. If the CDP URL uses an IPv6 ULA, the ACP device will try to reach it via the ACP. In that case the ACP address in the domain certificate of the CDP as learned by the ACP device during the HTTPs TLS handshake MUST match that ULA address in the HTTPs URL.

Renewal of certificates SHOULD start after less than 50% of the domain certificate lifetime so that network operations has ample time to investigate and resolve any problems that cause a device to not renew its domain certificate in time - and to allow prolonged periods of running parts of a network disconnected from any CA.

Certificate lifetime should be set to be as short as feasible. Given how certificate renewal is fully automated via ACP and EST, the primarily imiting factor for shorter certificate lifetimes (than the typical one year) is load on the EST server(s) and CA. It is therefore recommended that ACP domain certificates are managed via a CA chain where the assigning CA has enough peformance to manage short lived certificates.

See Section 10.1 for further optimizationss of certificate optimizations when BRSKI can be used.

6.2. AN Adjacency Table

To know to which nodes to establish an ACP channel, every autonomic node maintains an adjacency table. The adjacency table contains information about adjacent autonomic nodes, at a minimum: node-ID, IP address, domain, certificate. An autonomic device MUST maintain this adjacency table up to date. This table is used to determine to which neighbor an ACP connection is established.

Where the next autonomic device is not directly adjacent, the information in the adjacency table can be supplemented by configuration. For example, the node-ID and IP address could be configured.

The adjacency table MAY contain information about the validity and trust of the adjacent autonomic node's certificate. However, subsequent steps MUST always start with authenticating the peer.

The adjacency table contains information about adjacent autonomic nodes in general, independently of their domain and trust status. The next step determines to which of those autonomic nodes an ACP connection should be established.

6.3. Neighbor Discovery with DULL GRASP

Because of the the considerations in Section 10.2, the ACP uses DULL (Discovery Unsolicited Link-Local) insecure instances of GRASP for discovery of ACP neighbors. See section 3.5.2.2 of [I-D.ietf-anima-grasp] for its formal definition.

The ACP uses one instance of DULL GRASP for every physical L2 subnet of the ACP device to discovery physically adjacent candidate ACP neighbors. Ideally, all physical interfaces SHOULD be brought up enough so that ACP discovery can be performed and any physically connected interfaces with ACP neighbors can then be brought into the ACP even if the interface is otherwise not configured. Reception of packets on such otherwise unconfigure interfaces MUST be limited so that at first only IPv6 link-local address assignment (SLAAC) and DULL GRASP works and then only the following ACP secure channel setup packets - but not any other unnecessary traffic (eg: no other link-local IPv6 transport stack responders for example).

ACP discovery MUST NOT be enabled by default on any non-physical interfaces. In particular, ACP discovery MUST NOT run inside the ACP. See Section 8.2.2 how to enable and use ACP with auto-discovery across configured tunnels.

See Section 7 for how ACP should be extended on L3/L2 devices.

Note: If an ACP device also implements BRSKI (see Section 10.1) then the above considerations also apply to discovery for BRSKI. Each DULL instance of GRASP set up for ACP is then also used for the discovery of a bootstrap proxy via BRSKI when the device does not have a domain certificate. Discovery of ACP neighbors happens only when the device does have the certificate. The device therefore never needs to discover both a bootstrap proxy and ACP neighbor at the same time.

An autonomic node announces itself to potential ACP peers by use of the "AN_ACP" objective. This is a synchronization objective intended to be flooded on a single link using the GRASP Flood Synchronization (M_FLOOD) message. In accordance with the design of the Flood

message, a locator consisting of a specific link-local IP address, IP protocol number and port number will be distributed with the flooded objective. An example of the message is informally:

Example:

```
[M_FLOOD, 12340815, h'fe80000000000000c0011001FEEF0000, 1,
    ["AN_ACP", SYNCH-FLAG, 1, "IKEv2"],
    [O_IPv6_LOCATOR,
         h'fe80000000000000c0011001FEEF0000, UDP, 15000]
]
```

The formal CDDL definition is:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]

objective = ["AN_ACP", objective-flags, loop-count,
                                        objective-value]

objective-flags = ; as in the GRASP specification
loop-count = 1    ; limit to link-local operation
objective-value = text ; name of the (list of) secure
                       ; channel negotiation protocol(s)
```

The objective-flags field is set to indicate synchronization.

The ttl and loop-count are fixed at 1 since this is a link-local operation.

The session-id is a random number used for loop prevention (distinguishing a message from a prior instance of the same message). In DULL this field is irrelevant but must still be set according to the GRASP specification.

The originator MUST be the IPv6 link local address of the originating autonomic node on the sending interface.

The 'objective-value' parameter is (normally) a string indicating the secure channel protocol available at the specified or implied locator.

The locator is optional and only required when the secure channel protocol is not offered at a well-defined port number, or if there is no well defined port number. For example, "IKEv2" has a well defined port number 500, but in the above example, the candidate ACP neighbor is offering ACP secure channel negotiation via IKEv2 on port 15000 (for the sake of creating a non-standard example).

If a locator is included, it MUST be an O_IPv6_LOCATOR, and the IPv6 address MUST be the same as the initiator address (these are DULL requirements to minimize third party DoS attacks).

The secure channel methods defined in this document use the objective values of "IKEv2" and "dTLS". There is no distinction between IKEv2 native and GRE-IKEv2 because this is purely negotiated via IKEv2.

A node that supports more than one secure channel protocol needs to flood multiple versions of the "AN_ACP" objective, each accompanied by its own locator. This can be in a single GRASP M_FLOOD message.

If multiple secure channel protocols are supported that all are run on well-defined ports, then they can be announced via a single AN_ACP objective using a list of string names as the objective value without a following locator-option.

Note that a node serving both as an ACP node and BRSKI Join Proxy may choose to distribute the "AN_ACP" objective and "AN_join_proxy" objective in the same flood message, since GRASP allows multiple objectives in one Flood message. This may be impractical though if ACP and BRSKI operations are implemented via separate software modules / ASAs though.

The result of the discovery is the IPv6 link-local address of the neighbor as well as its supported secure channel protocols (and non-standard port they are running on). It is stored in the AN Adjacency Table, see Section 6.2 which then drives the further building of the ACP to that neighbor.

6.4. Candidate ACP Neighbor Selection

An autonomic node must determine to which other autonomic nodes in the adjacency table it should build an ACP connection. This is based on the information in the AN Adjacency table.

The ACP is by default established exclusively between nodes in the same domain. This includes all routing subdomains.Section 10.5 explains how ACP connections across routing subdomains are special.

Future extensions to this document including Intent can change this default behaviour. Examples include:

- Build the ACP across all domains that have a common parent domain. For example ACP nodes with domain "example.com", nodes of "example.com", "access.example.com", "core.example.com" and "city.core.example.com" could all establish one single ACP.

- ACP connections across domains with different CA (certificate authorities) could establish a common ACP by installing the alternate domains' CA into the trusted anchor store. This is an executive management action that could easily be accomplished through the control channel created by the ACP.

Since Intent is transported over the ACP, the first ACP connection a node establishes is always following the default behaviour. See Section 10.5 for more details.

The result of the candidate ACP neighbor selection process is a list of adjacent or configured autonomic neighbors to which an ACP channel should be established. The next step begins that channel establishment.

6.5. Channel Selection

To avoid attacks, initial discovery of candidate ACP peers can not include any non-protected negotiation. To avoid re-inventing and validating security association mechanisms, the next step after discoving the address of a candidate neighbor can only be to try first to establish a security association with that neighbor using a well-known security association method.

At this time in the lifecycle of autonomic devices, it is unclear whether it is feasible to even decide on a single MTI (mandatory to implement) security association protocol across all autonomic devices:

From the use-cases it seems clear that not all type of autonomic devices can or need to connect directly to each other or are able to support or prefer all possible mechanisms. For example, code space limited IoT devices may only support dTLS (because that code exists already on them for end-to-end security use-cases), but low-end in-ceiling L2 switches may only want to support MacSec because that is also supported in HW, and only a more flexible gateway device may need to support both of these mechanisms and potentially more.

To support extensible secure channel protocol selection without a single common MTI protocol, autonomic devices must try all the ACP secure channel protocols it supports and that are feasible because the candidate ACP neighbor also announced them via its AN_ACP GRASP parameters (these are called the "feasible" ACP secure channel protocols).

To ensure that the selection of the secure channel protocols always succeeds in a predictable fashion without blocking, the following rules apply:

An autonomic device may choose to attempt initiate the different feasible ACP secure channel protocol it supports according to its local policies sequentially or in parallel, but it MUST support acting as a responder to all of them in parallel.

Once the first secure channel protocol succeeds, the two peers know each others certificates (because that must be used by all secure channel protocols for mutual authentication. The device with the lower Device-ID in the ACP address becomes Bob, the one with the higher Device-ID in the certificate Alice.

Bob becomes passive, he does not attempt to further initiate ACP secure channel protocols with Alice and does not consider it to be an error when Alice closes secure channels. Alice becomes the active party, continues to attempt setting up secure channel protocols with Bob until she arrives at the best one (from her view) that also works with Bob.

For example, originally Bob could have been the initiator of one ACP secure channel protocol that Bob preferred and the security association succeeded. The roles of Bob abd Alice are then assigned. At this stage, the protocol may not even have completed negotiating a common security profile. The protocol could for example have been IPsec. It is not up to Alice to devide how to proceed. Even if the IPsec connecting determined a working profile with Bob, Alice might prefer some other secure protocol (eg: dTLS) and try to set that up with Bob. If that succeeds, she would close the IPsec connection. If no better protocol attempt succeeds, she would keep the IPsec connection.

All this negotiation is in the context of an "L2 interface". Alice and Bob will build ACP connections to each other on every "L2 interface" that they both connect to. An autonomic device must not assume that neighbors with the same L2 or link-local IPv6 addresses on different L2 interfaces are the ame devices. This can only be determined after examining the certificate after a successful security association attempt.

6.6. Candidate ACP Neighbor certificate verification

Independent of the security association protocol choosen, candidate ACP neighbors need to be authenticated based on their autonomic domain certificate. This implies that any security association protocol MUST support certificate based authentication that can support the following verification steps:

- The certificate is valid as proven by the security associations protocol exchanges.

- The peers certificate is signed by the same CA as the devices domain certificate.

- If the device certificates indicate a CDP or OCSP then the peer's certificate must be valid according to those criteria. eg: OCSP check across the ACP or not listed in the CRL.

- The peers certificate has a valid ACP information field (subjectAltName / rfc822Name) and the domain name in that peers ACP information field is the same as in the devices certificate. Note that future Intent rules may modify this for example in support of subdomains. See Section 10.5.

If the peers certificate fails any of these checks, the connection attempt is aborted and an error logged (with throttling).

6.7. Security Association protocols

The following sections define the security association protocols that we consider to be important and feasible to specify in this document:

6.7.1. ACP via IKEv2

An autonomic device announces its ability to support IKEv2 as the ACP secure channel protcol in GRASP as "IKEv2".

6.7.1.1. Native IPsec

To run ACP via IPsec transport mode, no further IANA assignments/ definitions are required. All autonomic devices supporting IPsec MUST support IPsec security setup via IKEv2, transport mode encapsulation via the device and peer link-local IPv6 addresses, AES256 encryption and SHA256 hash.

In terms of IKEv2, this means the initiator will offer to support IPsec transport mode with next protocol equal 41 (IPv6).

6.7.1.2. IPsec with GRE encapsulation

In network devices it is often easier to provide virtual interfaces on top of GRE encapsulation than natively on top of a simple IPsec association. On those devices it may be necessary to run the ACP secure channel on top of a GRE connection protected by the IPsec association. The requirements for the IPsec association are the same as in the native IPsec case, but instead of directly carrying the ACP IPv6 packets, the payload is an ACP IPv6 packet inside GRE/IPv6. The mandatory security profile is the same as for native IPsec: peer link-local IPv6 addresses, AES256 encryption, SHA256 hash.

In terms of IKEv2 negotiation, this means the initiator must offer to support IPsec transport mode with next protocol equal to GRE (47), followed by 41 (IPv6) (because native IPsec is required to be supported, see below).

If IKEv2 initiator and responder support GRE, it will be selected. The version of GRE to be used must the according to [RFC7676].

6.7.2. ACP via dTLS

We define the use of ACP via dTLS in the assumption that it is likely the first transport encryption code basis supported in some classes of constrained devices.

To run ACP via UDP and dTLS v1.2 [RFC6347] a locally assigned UDP port is used that is announced as a parameter in the GRASP AN_ACP objective to candidate neighbors. All autonomic devices supporting ACP via dTLS MUST support AES256 encryption and not permit weaker crypto options.

There is no additional session setup or other security association besides this simple dTLS setup. As soon as the dTLS session is functional, the ACP peers will exchange ACP IPv6 packets as the payload of the dTLS transport connection. Any dTLS defined security association mechanisms such as re-keying are used as they would be for any transport application relying solely on dTLS.

6.7.3. ACP Secure Channel Requirements

A baseline autonomic device MUST support IPsec natively and MAY support IPsec via GRE. A constrained autonomic device MUST support dTLS. Autonomic edge device connecting constrained areas with baseline areas MUST therefore support IPsec and dTLS.

Autonomic devices need to specify in documentation the set of secure ACP mechanisms they suppport.

ACP secure channel MUST imediately be terminated when the lifetime of any certificate in the chain used to authenticate the neighbor expires or becomes revoked. Note that is is not standard behavior in secure channel protocols such as IPsec because the certificate authentication only influences the setup of the secure channel in these protocols.

6.8. GRASP in the ACP

6.8.1. GRASP as a core service of the ACP

The ACP MUST run an instance of GRASP inside of it. It is a key part of the ACP services. They function in GRASP that makes it fundamental as a service is the ability for ACP wide service discovery (called objectives in GRASP). In most other solution designs such distributed discovery does not exist at all or it was added as an afterthought and relied upon inconsistently resulting in diminished self configuration capabilities. of prior solutions.

The ACP does not provide generic IP multicast services, but only IP unicast which is realized via the RPL routing protocol (described below) and objective discovery and negotiation realized via the ACP instance of GRASP. We consider this to be a more lightweight, modular and easier to extend approach than trying to put service announcement and discovery onto some autoconfigured network wide IP multicast layer (for which so far there is no good definition) or embed it into some IGP flooding mechanism (which makes it less modular and agile to improve upon).

6.8.2. ACP as the Security and Transport substrate for GRASP

In the terminology of GRASP ([I-D.ietf-anima-grasp]), the ACP is the security and transport substrate for the GRASP instance run inside the ACP.

This means that the ACP is responsible to ensure that this instance of GRASP is only using the ACP virtual interfaces. Whenever the ACP adds or deletes such an interface (because of new ACP secure channels or loss thereof), the ACP needs to indicate this to the ACP instance of GRASP. The ACP exists also in the absence of any active ACP neighbors. It is created when the device has a domain certificate. In this case ASAs using GRASP running on the same device would still need to be able to discover each others objectives. When the ACP does not exist ASAs leveraging the ACP instance of GRASP via APIs MUST still be able to operate, and MUST be able to understand that there is no ACP and that therefore the ACP instance of GRASP can not provide services.

GRASP inside the ACP uses link-local UDP IPv6 multicast across the ACP virtual interfaces for GRASP neighbor discovery and IPv6 over TLS across the ACP virtual interfaces for any of its unicast messages. TLS is TLS 1.2 ([RFC5246]) with AES256 encryption and SHA256. Authentication is via the the domain certificates on both sides.

TLS is mandated for GRASP because the ACP secure channel mandatory authentication and encryption protects only against attacks from the outside but not against attacks from the inside - compromised ACP members that have (not yet) been detected and removed (eg: via domain certificate revocation / expiry).

Eavesdropping/spoofing by a compromised ACP device is possible because the provider and consumer of an objective have no unique information about the other side that would allow them to distinguish a benevolent from a compromised peer. The compromised ACP device would simply announce the objective as well, potentially filter the original objective in GRASP when it is a Man In The Middle (MITM) and act as an application level proxy. This of course requires that the compromised ACP node understand the semantic of the GRASP negotiation to an extend that allows it to proxy it without being detected.

6.9. Context Separation

The ACP is in a separate context from the normal data plane of the device. This context includes the ACP channels IPv6 forwarding and routing as well as any required higher layer ACP functions.

In classical network device platforms, a dedicated so called "Virtual routing and forwarding instance" (VRF) is one logical implementation option for the ACP. If possible by the platform SW architecture, separation options that minimize shared components are preferred, such as a logical container or virtual machine instance. The context for the ACP needs to be established automatically during bootstrap of a device. As much as possible it should be protected from being modified unintentionally by data plane configuration.

Context separation improves security, because the ACP is not reachable from the global routing table. Also, configuration errors from the data plane setup do not affect the ACP.

6.10. Addressing inside the ACP

The channels explained above typically only establish communication between two adjacent nodes. In order for communication to happen across multiple hops, the autonomic control plane requires internal network wide valid addresses and routing. Each autonomic node must create a virtual interface with a network wide unique address inside the ACP context mentioned in Section 6.9. This address may be used also in other virtual contexts.

With the algorithm introduced here, all ACP devices in the same subdomain have the same /48 prefix. Conversely, global IDs from different domains are unlikely to clash, such that two networks can

be merged, as long as the policy allows that merge. See also Section 9.1 for a discussion on merging domains.

Links inside the ACP only use link-local IPv6 addressing, such that each node only requires one routable virtual address.

6.10.1. Fundamental Concepts of Autonomic Addressing

- Usage: Autonomic addresses are exclusively used for self-management functions inside a trusted domain. They are not used for user traffic. Communications with entities outside the trusted domain use another address space, for example normally managed routable address space (called "Data Plane" in this document).

- Separation: Autonomic address space is used separately from user address space and other address realms. This supports the robustness requirement.

- Loopback-only: Only loopback interfaces of autonomic nodes carry routable address(es); all other interfaces exclusively use IPv6 link local for autonomic functions. The usage of IPv6 link local addressing is discussed in [RFC7404].

- Use-ULA: For loopback interfaces of autonomic nodes, we use Unique Local Addresses (ULA), as specified in [RFC4193]. An alternative scheme was discussed, using assigned ULA addressing. The consensus was to use ULA-random [[RFC4193] with L=1], because it was deemed to be sufficient.

- No external connectivity: They do not provide access to the Internet. If a node requires further reaching connectivity, it should use another, traditionally managed address scheme in parallel.

- Addresses in the ACP are permanent, and do not support temporary addresses as defined in [RFC4941].

The ACP is based exclusively on IPv6 addressing, for a variety of reasons:

- Simplicity, reliability and scale: If other network layer protocols were supported, each would have to have its own set of security associations, routing table and process, etc.

- Autonomic functions do not require IPv4: Autonomic functions and autonomic service agents are new concepts. They can be

exclusively built on IPv6 from day one. There is no need for backward compatibility.

- OAM protocols no not require IPv4: The ACP may carry OAM protocols. All relevant protocols (SNMP, TFTP, SSH, SCP, Radius, Diameter, ...) are available in IPv6.

6.10.2. The ACP Addressing Base Scheme

The Base ULA addressing scheme for autonomic nodes has the following format:

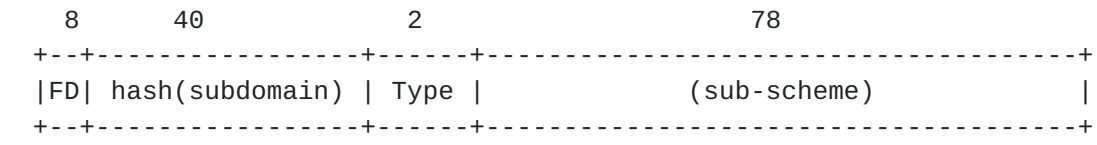

Figure 2: ACP Addressing Base Scheme

The first 48 bits follow the ULA scheme, as defined in [RFC4193], to which a type field is added:

- "FD" identifies a locally defined ULA address.
- The ULA "global ID" is set here to be a hash of the subdomain name, which results in a pseudo-random 40 bit value. It is calculated as the first 40 bits of the SHA256 hash of the subdomain name, in the example of Section 6.1.1 "area51.research.acp.example.com".
- To allow for extensibility, the fact that the ULA "global ID" is a hash of the subdomain name SHOULD NOT be assumed by any autonomic device during normal operations. The hash function is only executed during the creation of the certificate. If BRSKI is used then the registrar will create the ACP information field in response to the CSR Attribute Request by the pledge.
- Type: This field allows different address sub-schemes in the future. The goal is to start with a single sub-schemes, but to allow for extensions later if and when required. This addresses the "upgradability" requirement. Assignment of types for this field should be maintained by IANA.

6.10.3. ACP Zone Addressing Sub-Scheme

The sub-scheme defined here is defined by the Type value 00b (zero) in the base scheme.

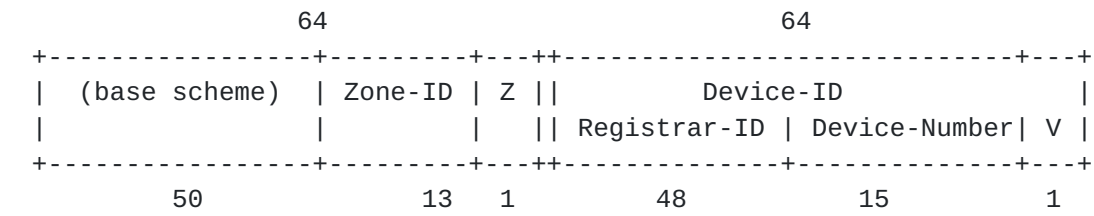

Figure 3: ACP Zone Addressing Sub-Scheme

The fields are defined as follows:

- Zone-ID: If set to all zero bits: The Device-ID bits are used as an identifier (as opposed to a locator). This results in a non-hierarchical, flat addressing scheme. Any other value indicates a zone. See section Section 6.10.3.1 on how this field is used in detail.

- Z: MUST be 0.

- Device-ID: A unique value for each device.

The 64 bit Device-ID is derived and composed as follows:

- Registrar-ID (48 bit): A number unique inside the domain that identifies the registrar which assigned the Device-ID to the device. A MAC address of the registrar can be used for this purpose.

- Device-Number (Device-Number): A number which is unique for a given registrar, to identify the device. This can be a sequentially assigned number.

- V (1 bit): Virtualization bit: 0: Indicates the ACP itself ("autonomic node base system); 1: Indicates the optional "host" context on the ACP device (see below).

When referring to the Device-ID of an ACP device overall, the V bit(s) is/are always "0". This is also the address used in the ACP information in the domain certificate. The V bit(s) identify the bits the device can assign itself. The Device knows how many bits those are from the addressing sub-scheme (see below for a Sub-Scheme with more than one V-bit).

The "Device-ID" itself is unique in a domain (i.e., the Zone-ID is not required for uniqueness). Therefore, a device can be addressed either as part of a flat hierarchy (zone ID = 0), or with an aggregation scheme (any other zone ID). A address with zone-ID = 0

is an identifier, with another zone-ID as a locator. See Section 6.10.3.1 for a description of the zone bits.

The Virtual bit in this sub-scheme allows to easily add the ACP as a component to existing systems without causing problems in the port number space between the services in the ACP and the existing system. V:0 is the ACP router (autonomous node base system), V:1 is the host with pre-existing transport endpoints on it that could collide with the transport endpoints used by the ACP router. The ACP host could for example have a virtual p2p interface with the V:0 address as its router into the ACP. Depending on the SW design of ASA (outside the scope of this specification), they may use the V:0 or V:1 address.

The location of the V bit(s) at the end of the address allows to announce a single prefix for each autonomic node. For example, in a network with 20,000 ACP devices, this avoid 20,000 additional routes in the routing table.

6.10.3.1. Usage of the Zone Field

The "Zone-ID" allows for the introduction of structure in the addressing scheme.

Zone = zero is the default addressing scheme in an autonomic domain. Every autonomic node MUST respond to its ACP address with zone=0. Used on its own this leads to a non-hierarchical address scheme, which is suitable for networks up to a certain size. In this case, the addresses primarily act as identifiers for the nodes, and aggregation is not possible.

If aggregation is required, the 13 bit value allows for up to 8192 zones. The allocation of zone numbers may either happen automatically through a to-be-defined algorithm; or it could be configured and maintained manually.

If a device learns through an autonomic method or through configuration that it is part of a zone, it MUST also respond to its ACP address with that zone number. In this case the ACP loopback is configured with two ACP addresses: One for zone 0 and one for the assigned zone. This method allows for a smooth transition between a flat addressing scheme and an hierarchical one.

(Theoretically, the 13 bits for the Zone-ID would allow also for two levels of zones, introducing a sub-hierarchy. We do not think this is required at this point, but a new type could be used in the future to support such a scheme.)

Note: The Zone-ID is one method to introduce structure or hierarchy into the ACP. Another way is the use of the routing subdomain field in the ACP that leads to different /40 ULA prefixes within an autonomic domain. This gives followup work two options to consider.

6.10.4. ACP Manual Addressing Sub-Scheme

The sub-scheme defined here is defined by the Type value 00b (zero) in the base scheme.

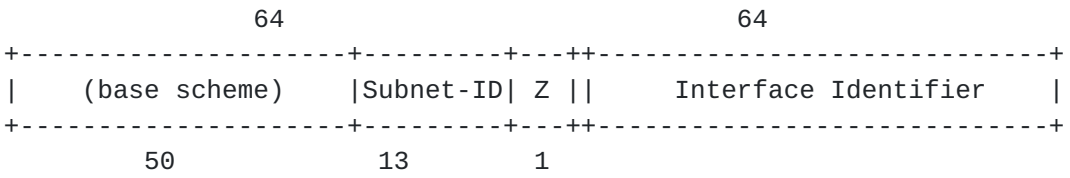

Figure 4: ACP Manual Addressing Sub-Scheme

The fields are defined as follows:

- Subnet-ID: Configured subnet identifier.
- Z: MUST be 1.
- Interface Identifier.

This sub-scheme is meant for "manual" allocation to subnets where the other addressing schemes can not be used. The primary use case is for assignment to ACP connect subnets (see Section 8.1.1).

"Manual" means that allocations of the Subnet-ID need to be done today with pre-existing, non-autonomic mechanisms. Every subnet that uses this addressing sub-scheme needs to use a unique Subnet-ID (unless some anycast setup is done). Future work may define mechanisms for auto-coordination between ACP devices and auto-allocation of Subnet-IDs between them.

The Z field is following the Subnet-ID field so that future work could allocate/coordinate both Zone-ID and Subnet-ID consistently and use an integrated aggregateable routing approach across them. Z=0 (Zone sub-scheme) would then be used for network wide unique, registrar assigned (and certificate protected) Device-IDs primarily for ACP devices while Z=1 would be used for device-level assigned Interface Identifiers primarily for non-ACP-devices.

6.10.5. ACP Vlong Addressing Sub-Scheme

The sub-scheme defined here is defined by the Type value 01b (one) in the base scheme.

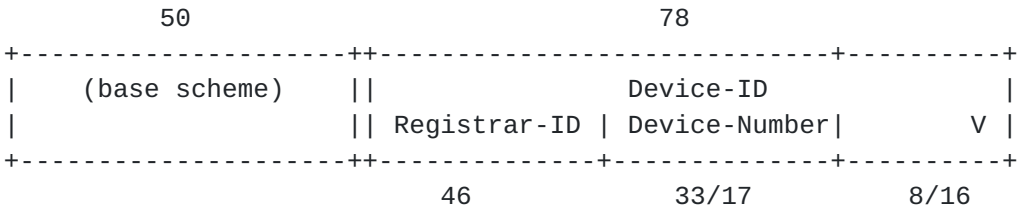

Figure 5: ACP Vlong Addressing Sub-Scheme

This addressing scheme foregoes the Zone field to allow for larger, flatter routed networks (eg: as in IoT) with more than 2^32 Device-Numbers. It also allows for up to 2^16 - 65536 different virtualized adddresses, which could be used to address individual software components in an ACP device.

The fields are the same as in the Zone sub-scheme with the following refinements:

- V: Virtualization bit: Values 0 and 1 as in Zone sub-scheme, further values use via definition in followup work.

- Registrar-ID: To maximize Device-Number and V, the Registrar-ID is reduced to 46 bits. This still allows to use the MAC address of a registrar by removing the V and U bits from the 48 bits of a MAC address (those two bits are never unique, so they can not be used to distinguish MAC addresses).

- If the first bit of the "Device-Number" is "1", then the Device number is 17 bit long and the V field is 16 bit long. Otherwise the Device-Number is 33 bit long and the V field is 8 bit long. "0" bit Device-Numbers are intended to be used for "general purpose" ACP devices that would potentially have a limited number (< 256) of internal users of the ACP that require separate V(irtual) addresses. "1" bit Device-Numbers are intended for ACP devices that are ACP edge devices (see Section 8.1.1) or that have a large number of internal ACP users requiring separate V(irtual) addresses. For example large SDN controllers with container modular software architecture (see Section 8.1.2).

6.10.6. Other ACP Addressing Sub-Schemes

Other ACP addressing sub-schemes can be defined if and when required. IANA would need to assign a new "type" for each new addressing sub-scheme. With the current allocations, 5 more schemes are possible without further reducing the number of bits in a future scheme.

6.11. Routing in the ACP

Once ULA address are set up all autonomic entities should run a routing protocol within the autonomic control plane context. This routing protocol distributes the ULA created in the previous section for reachability. The use of the autonomic control plane specific context eliminates the probable clash with the global routing table and also secures the ACP from interference from the configuration mismatch or incorrect routing updates.

The establishment of the routing plane and its parameters are automatic and strictly within the confines of the autonomic control plane. Therefore, no manual configuration is required.

All routing updates are automatically secured in transit as the channels of the autonomic control plane are by default secured, and this routing runs only inside the ACP.

The routing protocol inside the ACP is RPL ([RFC6550]). See Section 10.3 for more details on the choice of RPL.

RPL adjacencies are set up across all ACP channels in the same domain including all its routing subdomains. See Section 10.5 for more details.

6.11.1. RPL Profile

The following is a description of the RPL profile that ACP nodes need to support by default. The format of this section is derived from draft-ietf-roll-applicability-template.

6.11.1.1. Summary

In summary, the profile chosen for RPL is one that expects a fairly reliable network reasonable fast links so that RPL convergence will be triggered immediately upon recognition of link failure/recovery.

The key limitation of the chosen profile is that it is designed to not require any dataplane artifacts (such as [RFC6553]). While the senders/receivers of ACP packets can be legacy NOC devices connected via "ACP connect" (see Section 8.1.1 to the ACP, their connectivity

can be handled as non-RPL-aware leafs (or "Internet") accoding to the dataplane architecture explained in [I-D.ietf-roll-useofrplinfo]. This non-artifact profile is largely driven by the desire to avoid introducing the required Hop-by-Hop headers into the ACP VRF control plane. Many devices will have their VRF forwarding code designed into silicon.

In this profile choice, RPL has no dataplane artifacts. A simple destination prefix based upon the routing table is used. A consequence of supporting only a single instanceID (containing one DODAG), the ACP will only accomodate only a single class of routing table and can not create optimized routing paths to accomplish latency or energy goals.

Consider a network that has multiple NOCs in different locations. Only one NOC will become the DODAG root. Other NOCs will have to send traffic through the DODAG (tree) rooted in the primary NOC. Depending on topology, this can be an annoyance from a latency point of view, but it does not represent a single point of failure, as the DODAG can reconfigure itself when it detects data plane forwarding failures.

The lack of a RPI (the header defined by [RFC6553]), means that the dataplane will have no rank value that can be used to detect loops. As a result, traffic may loop until the TTL of the packet reaches zero. This the same behavior as that of other IGPs that do not have the data plane options as RPPL. There are a variety of heuristics that can be used to signal from the data plane to the RPL control plane that a new route is needed.

Additionally, failed ACP tunnels will be detected by IKEv2 Dead Peer Detection (which can function as a replacement for an LLN's ETX). A failure of an ACP tunnel should signal the RPL control plane to pick a different parent.

Future Extensions to this RPL profile can provide optimality for multiple NOCs. This requires utilizing data plane artifact including IPinIP encap/decap on ACP routers and processing of IPv6 RPI headers. Alternatively (Src,Dst) routing table entries could be used. A decision for the preferred technology would have to be done when such extension is defined.

6.11.1.2. RPL Instances

Single RPL instance. Default RPLInstanceID = 0.

6.11.1.3. Storing vs. Non-Storing Mode

RPL Mode of Operations (MOP): mode 3 "Storing Mode of Operations with multicast support". Implementations should support also other modes. Note: Root indicates mode in DIO flow.

6.11.1.4. DAO Policy

Proactive, aggressive DAO state maintenance:

- Use K-flag in unsolicited DAO indicating change from previous information (to require DAO-ACK).

- Retry such DAO DAO-RETRIES(3) times with DAO- ACK_TIME_OUT(256ms) in between.

6.11.1.5. Path Metric

Hopcount.

6.11.1.6. Objective Function

Objective Function (OF): Use OF0 [RFC6552]. No use of metric containers.

rank_factor: Derived from link speed: <= 100Mbps: LOW_SPEED_FACTOR(5), else HIGH_SPEED_FACTOR(1)

6.11.1.7. DODAG Repair

Global Repair: we assume stable links and ranks (metrics), so no need to periodically rebuild DODAG. DODAG version only incremented under catastrophic events (eg: administrative action).

Local Repair: As soon as link breakage is detected, send No-Path DAO for all the targets that where reachable only via this link. As soon as link repair is detected, validate if this link provides you a better parent. If so, compute your new rank, and send new DIO that advertises your new rank. Then send a DAO with a new path sequence about yourself.

stretch_rank: none provided ("not stretched").

Data Path Validation: Not used.

Trickle: Not used.

6.11.1.8. Multicast

Not used yet but possible because of the seleced mode of operations.

6.11.1.9. Security

[RFC6550] security not used, substituted by ACP security.

6.11.1.10. P2P communications

Not used.

6.11.1.11. IPv6 address configuration

Every ACP device (RPL node) announces an IPv6 prefix covering the address(es) used internally in the ACP device. The prefix length depends on the choosen addressing sub-scheme of the ACP address provisioned into the certificate of the ACP device, eg: /127 for Zone addressing sub-scheme or /112 or /120 for Vlong addressing sub-scheme. See Section 6.10 for more details.

Every ACP device MUST install a blackhole (aka null route) route for whatever ACP address space that it advertises (i.e: the /96 or /127). This is avoid routing loops for addresses that an ACP node has not (yet) used.

6.11.1.12. Administrative parameters

Administrative Preference ([RFC6552], 3.2.6 - to become root): Indicated in DODAGPreference field of DIO message.

- Explicit configured "root": 0b100
- Registrar (Default): 0b011
- AN-connect (non registrar): 0b010
- Default: 0b001.

6.11.1.13. RPL Dataplane artifacts

RPI (RPL Packet Information [RFC6553]): Not used as there is only a single instance, and data path validation is not being used.

SRH (RPL Source Routing - RFC6552): Not used. Storing mode is being used.

6.11.1.14. Unknown Destinations

Because RPL minimizes the size of the routing and forwarding table, prefixes reachable through the same interface as the RPL root are not known on every ACP device. Therefore traffic to unknown destination addresses can only be discovered at the RPL root. The RPL root SHOULD have attach safe mechanisms to operationally discover and log such packets.

6.12. General ACP Considerations

Since channels are by default established between adjacent neighbors, the resulting overlay network does hop by hop encryption. Each node decrypts incoming traffic from the ACP, and encrypts outgoing traffic to its neighbors in the ACP. Routing is discussed in Section 6.11.

6.12.1. Addressing of Secure Channels in the data plane

In order to be independent of the Data Plane configuration of global IPv6 subnet addresses (that may not exist when the ACP is brought up), Link-local secure channels MUST use IPv6 link local addresses between adjacent neighbors. The fully autonomic mechanisms in this document only specify these link-local secure channels.Section 8.2 specify extensions in which secure channels are tunnels, then this requirement does not apply.

The Link-local secure channels specified in this document therefore depend on basic IPv6 link-local functionality to be auto-enabled by the ACP and prohibiting the Data Plane from disabling it. The ACP also depends on being able to operate the secure channel protocol (eg: IPsec / dTLS) across IPv6 link-local addresses, something that may be an uncommon profile. Functionaly, these are the only interactions with the Data Plane that the ACP needs to have.

To mitigate these interactions with the Data Plane, extensions to this document may specify additional layer 2 or layer encapsulations for ACP secure channels as well as other protocols to auto-discover peer endpoints for such encapsulations (eg: tunneling across L3 or use of L2 only encapsulations).

6.12.2. MTU

The MTU for ACP secure channels must be derived locally from the underlying link MTU minus the secure channel encapsulation overhead.

ACP secure Channel protocols do not need to perform MTU discovery because they are built across L2 adjacencies - the MTU on both sides connecting to the L2 connection are assumed to be consistent.

Extensions to ACP where the ACP is for example tunneled need to consider how to guarantee MTU consistency. This is a standard issue with tunneling, not specific to running the ACP across it. Transport stacks running across ACP can perform normal PMTUD (Path MTU Discovery). Because the ACP is meant to be prioritize reliability over performance, they MAY opt to only expect IPv6 minimum MTU (1280) to avoid running into PMTUD implementation bugs or underlying link MTU mismatch problems.

6.12.3. Multiple links between nodes

If two nodes are connected via several links, the ACP SHOULD be established across every link, but it is possible to establish the ACP only on a sub-set of links. Having an ACP channel on every link has a number of advantages, for example it allows for a faster failover in case of link failure, and it reflects the physical topology more closely. Using a subset of links (for example, a single link), reduces resource consumption on the devices, because state needs to be kept per ACP channel. The negotiation scheme explained in Section 6.5 allows Alice (the node with the higher ACP address) to drop all but the desired ACP channels to Bob - and Bob will not re-try to build these secure channels from his side unless Alice shows up with a previously unknown GRASP announcement (eg: on a different link or with a different address announced in GRASP).

6.12.4. ACP interfaces

The ACP VRF has conceptually two type of interfaces: The ACP loopback interface(s) to which the ACP ULA address(es) are assigned and the "ACP virtual interfaces" that are mapped to the ACP secure channels.

The term loopback interface is commonly referred to internal pseudo interfaces through which the device can only reach itself. In network devices these interfaces are used to hold addresses used by the transport stack of the system when an address should be reliable reachable in the presence of arbitrary link failures. As long as addresses on a loopback interface are routeable in the routing protocol used, they will be reachable as long as there is at least one working network path. This is opposed to routeable address assigned to an externally connected interface. That address will become unreachable when that interface goes down. For this reason, the ACP (ULA) address(es) are assigned to loopback type interface.

ACP secure channels, eg: IPsec, dTLS or other future security associations with neighboring ACP devices can be mapped to ACP virtual interfaces in different ways: In one case, every ACP secure channel is mapped into a separate point-to-point ACP virtual interface. If a physical subnet has more than two ACP capable nodes

(in the same domain), this implementation approach will lead to a full mesh of ACP virtual interfaces between them. In a more advanced implementation approach, the ACP will construct a single multi-access ACP virtual interface for all ACP secure channels to ACP capable nodes reachable across the same physical subnet. The multi-access ACP virtual interace needs to replicate link-local IPv6 multicast packets sent into the interface towards all ACP secure channels mapped into it. There is no need for all ACP capable nodes on the same physical multi-access subnet to agree on a common implementation approach. This is purely a node local decision.

Multi-access ACP virtual interfaces are preferrable because they do reflect the presence of a multi-access physical networks into the virtual interfaces of the ACP. This makes it for example simpler to build services with topology awareness inside the ACP VRF in the same way as they could habe been built running natively on the multi-access interfaces. One example is the efficiency of flooding of GRASP messages. Assume such a LAN with three ACP neighbors, Alice Bob and Carol. Alice sends a GRASP link-local multicast message to Bob and Carol. If Alices ACP emulates the LAN as one point-to-point interface to Bob and one to Carol, GRASP itself will send two copies, if Alices ACP emulates a LAN, GRASP will send one packet and the ACP will replicate it. The result is the same. The difference happens when Bob and Carol receive their packet. If they use point-to-point ACP virtual interfaces, their GRASP instance would forward the packet from Alice to each other as part of the GRASP flooding procedure. These packets are of course redundant (unnecessary) and would be discarded by GRASP on receipt as duplicates. If Bob and Charlies ACP would emulate a LAN, then this would not happen, because GRASPs flooding procedure does not send bac packets to the interface they where received from.

For this reason, the ACP SHOULD map ACP secure channels on multi-access LANs into multi-access ACP virtual interfaces.

Care must be taken when creating multi-access ACP virtual interfaces across ACP secure channels between ACP devices in different domains or routing subdomains. The policies to be negotiated may be described as peer-to-peer policies in which case it is easier to create point-to-point ACP virtual interfaces for these secure channels.

7. ACP support on L2 switches/ports (Normative)

7.1. Why

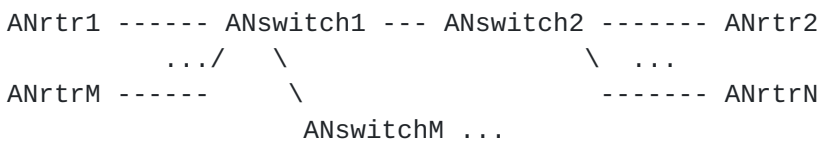

Figure 6

Consider a large L2 LAN with ANrtr1...ANrtrN connected via some topology of L2 switches. Examples include large enterprise campus networks with an L2 core, IoT networks or broadband aggregation networks which often have even a multi-level L2 switched topology.

If the discovery protocol used for the ACP is operating at the subnet level, every AN router will see all other AN routers on the LAN as neighbors and a full mesh of ACP channels will be built. If some or all of the AN switches are autonomic with the same discovery protocol, then the full mesh would include those switches as well.

A full mesh of ACP connections like this can creates fundamental scale challenges. The number of security associations of the secure channel protocols will likely not scale arbitrarily, especially when they leverage platform accelerated encryption/decryption. Likewise, any other ACP operations (such as routing) needs to scale to the number of direct ACP neigbors. An AN router with just 4 interfaces might be deployed into a LAN with hundreds of neighbors connected via switches. Introducing such a new unpredictable scaling factor requirement makes it harder to support the ACP on arbitrary platforms and in arbitrary deployments.

Predictable scaling requirements for ACP neighbors can most easily be achieved if in topologies like these, AN capable L2 switches can ensure that discovery messages terminate on them so that neighboring AN routers and switches will only find the physically connected AN L2 switches as their candidate ACP neighbors. With such a discovery mechanism in place, the ACP and its security associations will only need to scale to the number of physical interfaces instead of a potentially much larger number of "LAN-connected" neighbors. And the ACP topology will follow directly the physical topology, something which can then also be leveraged in management operations or by ASAs.

In the example above, consider ANswitch1 and ANswitchM are AN capable, and ANswitch2 is not AN capable. The desired ACP topology is therefore that ANrtr1 and ANrtrM only have an ACP connetion to ANswitch1, and that ANswitch1, ANrtr2, ANrtrN have a full mesh of ACP connection amongst each other. ANswitch1 also has an ACP connection

with ANswitchM and ANswitchM has ACP connections to anything else behind it.

7.2. How (per L2 port DULL GRASP)

To support ACP on L2 switches or L2 switches ports of an L3 device, it is necessary to to make those L2 ports look like L3 interfaces for the ACP implementation. This primarily involves the creation of a separate DULL GRASP instance/domain on every such L2 port. Because GRASP has a dedicated IPv6 link-local multicast address, it is sufficient that all ethernet packets for this address are being extracted at the port level and passed to that DULL GRASP instance. Likewise the IPv6 link-local multicast packets sent by that DULL GRASP instance need to be sent only towards the L2 port for this DULL GRASP instance.

The rest of ACP operations can operate in the same way as in L3 devices: Assume for example that the device is an L3/L2 hybrid device where L3 interfaces are assigned to VLANs and each VLAN has potentially multiple ports. DULL GRASP is run as described individually on each L2 port. When it discovers a candidate ACP neighbor, it passes its IPv6 link-local address and supported secure channel protocols to the ACP secure channel negotiation that can be bound to the L3 (VLAN) interface. It will simply use link-local IPv6 multicast packets to the candidate ACP neighbor. Once a secure channel is established to such a neighbor, the virtual interface to which this secure channel is mapped should then actually be the L2 port and not the L3 interface to best map the actual physical topology into the ACP virtual interfaces. See Section 6.12.4 for more details about how to map secure channels into ACP virtual interfaces. Note that a single L2 port can still have multiple ACP neighbors if it connect for example to multiple ACP neighbors via a non-ACP enabled switch. The per L2 port ACP virtual interface can threfore still be a multi-access virtual LAN.

For example, in the above picture, ANswitch1 would run separate DULL GRASP instances on its ports to ANrtr1, ANswitch2 and ANswitchI, even though all those three ports may be in the data plane in the same (V)LAN and perfom L2 switching between these ports, ANswitch1 would perform ACP L3 routing between them.

The description in the previous paragraph was specifically meant to illustrate that on hybrid L3/L2 devices that are common in enterprise, IoT and broadband aggregation, there is only the GRASP packet extraction (by ethernet address) and GRASP link-local multicast per L2-port packet injection that has to consider L2 ports at the hardware forwarding level. The remaining operations are purely ACP control plane and setup of secure channels across the L3

interface. This hopefully makes support for per-L2 port ACP on those hybrid devices easy.

This L2/L3 optimized approach is subject to "address stealing", eg: where a device on one port uses addresses of a device on another port. This is a generic issue in L2 LANs and switches often already have some form of "port security" to prohibit this. They rely on NDP or DHCP learning of which port/MAC-address and IPv6 address belong together and block duplicates. This type of function needs to be enabled to prohibit DoS attacks. Likewise the GRASP DULL instance needs to ensure that the IPv6 address in the locator-option matches the source IPv6 address of the DULL GRASP packet.

In devices without such a mix of L2 port/interfaces and L3 interfaces (to terminate any transport layer connections), implementation details will differ. Logically most simply every L2 port is considered and used as a separate L3 subnet for all ACP operations. The fact that the ACP only requires IPv6 link-local unicast and multicast should make support for it on any type of L2 devices as simple as possible, but the need to support secure channel protocols may be a limiting factor to supporting ACP on such devices. Future options such as 802.1ae could improve that situation.

A generic issue with ACP in L2 switched networks is the interaction with the Spanning Tree Protocol. Ideally, the ACP should be built also across ports that are blocked in STP so that the ACP does not depend on STP and can continue to run unaffected across STP topology changes (where reconvergence can be quite slow). The above described simple implementation options are not sufficient for this. Instead they would simply have the ACP run across the active STP topology and therefore the ACP would equally be interrupted and reconverge with STP changes.

L3/L2 devices SHOULD support per-L2 port ACP.

8. Support for Non-Autonomic Components (Normative)

8.1. ACP Connect

8.1.1. Non-Autonomic Controller / NMS system

The Autonomic Control Plane can be used by management systems, such as controllers or network management system (NMS) hosts (henceforth called simply "NMS hosts"), to connect to devices through it. For this, an NMS host must have access to the ACP. The ACP is a self-protecting overlay network, which allows by default access only to trusted, autonomic systems. Therefore, a traditional, non-autonomic

NMS system does not have access to the ACP by default, just like any other external device.

If the NMS host is not autonomic, i.e., it does not support autonomic negotiation of the ACP, then it can be brought into the ACP by explicit configuration. To support connections to adjacent non-autonomic nodes, an autonomic node with ACP must support "ACP connect" (sometimes also connect "autonomic connect"):

"ACP connect" is a function on an autonomic device that is called an "ACP edge device". With "ACP connect", interfaces on the device can be configured to be put into the ACP VRF. The ACP is then accessible to other (NOC) systems on such an interface without those systems having to support any ACP discovery or ACP channel setup. This is also called "native" access to the ACP because to those (NOC) systems the interface looks like a normal network interface (without any encryption/novel-signaling).

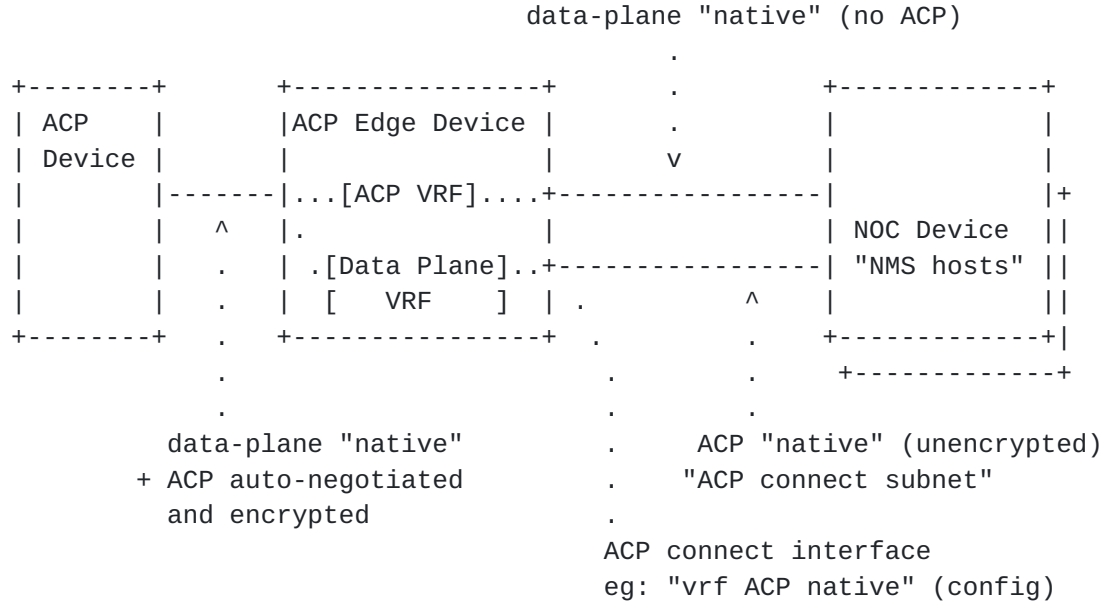

Figure 7: ACP connect

ACP connect has security consequences: All systems and processes connected via ACP connect have access to all autonomic nodes on the entire ACP, without further authentication. Thus, the ACP connect interface and (NOC) systems connected to it must be physically controlled/secured. For this reason the mechanisms described here do explicitly not include options to allow for a non-ACP router to be connected across an ACP connect interface and addresses behind such a router routed inside the ACP.

An ACP connect interface provides exclusively access to only the ACP. This is likely insufficient for many NMS hosts. Instead, they would require a second "data-plane" interface outside the ACP for connections between the NMS host and administrators, or Internet based services, or for direct access to the data plane. The document "Autonomic Network Stable Connectivity" [I-D.ietf-anima-stable-connectivity] explains in more detail how the ACP can be integrated in a mixed NOC environment.

The ACP connect interface must be (auto-)configured with an IPv6 address prefix. Is prefix SHOULD be covered by one of the (ULA) prefix(es) used in the ACP. If using non-autonomic configuration, it SHOULD use the ACP Manual Addressing Sub-Scheme (Section 6.10.4). It SHOULD NOT use a prefix that is also routed outside the ACP so that the addresses clearly indicate whether it is used inside the ACP or not.

The prefix of ACP connect subnets MUST be distributed by the ACP edge device into the ACP routing protocol (RPL). The NMS hosts MUST connect to prefixes in the ACP routing table via its ACP connect interface. In the simple case where the ACP usesonly one ULA prefix and all ACP connect subnets have prefixes covered by that ULA prefix, NMS hosts can rely on [RFC6724] - The NMS host will select the ACP connect interface because any ACP destination address is best matched by the address on the ACP connect interface. If the NMS hosts ACP connect interface uses another prefix or if the ACP uses multiple ULA prefixes, then the NMS hosts require (static) routes towards the ACP interface.

ACP Edge Device MUST only forward IPv6 packets received from an ACP connect interface into the ACP that has an IPv6 address from the ACP prefix assigned to this interface (sometimes called "RPF filtering"). This MAY be changed through administrative measures.

To limit the security impact of ACP connect, devices supporting it SHOULD implement a security mechanism to allow configuration/use of ACP connect interfaces only on devices explicitly targeted to be deployed with it (such as those physically secure locations like a NOC). For example, the certificate of such devices could include an extension required to permit configuration of ACP connect interfaces. This prohibits that a random ACP device with easy physical access that is not meant to run ACP connect could start leaking the ACP when it becomes compromised and the intruder configures ACP connect on it. The full workflow including the mechanism by which a registrar would select which device to give such a certificate to is subject to future work.

8.1.2. Software Components

The ACP connect mechanism be only be used to connect physically external systems (NMS hosts) to the ACP but also other applications, containers or virtual machines. In fact, one possible way to eliminate the security issue of the external ACP connect interface is to collocate an ACP edge device and an NMS host by making one a virtual machine or container inside the other - and therefore converting the unprotected external ACP subnet into an internal virtual subnet in a single device. This would ultimately result in a fully autonomic NMS host with minimum impact to the NMS hosts software architecture. This approach is not limited to NMS hosts but could equally be applied to devices consisting of one or more VNF (virtual network functions): An internal virtual subnet connecting out-of-band-management interfaces of the VNFs to an ACP edge router VNF.

The core requirement is that the software components need to have a network stack that permits access to the ACP and optionally also the data plane. Like in the physical setup for NMS hosts this can be realized via two internal virtual subnets. One that is connecting to the ACP (which could be a container or virtual machine by itself), and one (or more) connecting into the data plane.

This "internal" use of ACP connect approach should not considered to be a "workaround" because in this case it is possible to build a correct security model: It is not necessary to rely on unprovable external physical security mechanisms as in the case of external NMS hosts. Instead, the orchestration of the ACP, the virtual subnets and the software components can be done by trusted software that could be considered to be part of the ANI (or even an extended ACP). This software component is responsible to ensure that only trusted software components will get access to that virtual subnet and that only even more trusted software components will get access to both the ACP virtual subnet and the data plane (because those ACP users could leak traffic between ACP and data plane). This trust could be established for example through cryptographic means such signed software packages. The specification of these mechanisms is subject to future work.

Note that ASA (Autonomic Software Agents) could also be software components as described in this section, but further details of ASAs are subject to future work.

8.1.3. Auto Configuration

ACP edge devices, NMS hosts and software components that as describd in the previous section are meant to be composed via virtual interfaces SHOULD support on the ACP connect subnet Stateless Address Autoconfiguration (SLAAC - [RFC4862]) and route autoconfiguration according to [RFC4191].

The ACP edge device acts as the router on the ACP connect subnet, providing the (auto-)configured prefix for the ACP connect subnet to NMS hosts and/or software components. The ACP edge device uses route prefix option of RFC4191 to announce the default route (::/) with a lifetime of 0 and aggregated prefixes for routes in the ACP routing table with normal lifetimes. This will ensure that the ACP edge device does not become a default router, but that the NMS hosts and software components will route the prefixes used in the ACP to the ACP edge device.

Aggregated prefix means that the ACP edge device needs to only announce the /48 ULA prefixes used in the ACP but none of the actual /64 (Manual Addressing Sub-Scheme), /127 (Zone Addressing Sub-Scheme), /112 or /120 (Vlong Addressing Sub-Scheme) routes of actual ACP devices. If ACP interfaces are configured with non ULA prefixes, then those prefixes can not be aggreged without further configured policy on the ACP edge device. This explains the above recommendation to use ACP ULA prefix covered prefixes for ACP connect interfaces: They allow for a shorter list of prefixes to be signalled via RFC4191 to NMS hosts and software components.

The ACP edge devices that have a Vlong ACP address MAY allocate a subset of their /112 or /120 address prefix to ACP connect interface(s) to eliminate the need to non-autonomically configure/ provision the address prefixes for such ACP connect interfaces. See Section 11 for considerations how this updates the IPv6 address architecture and ULA specification.

8.1.4. Combined ACP and Data Data Plane Interface (VRF Select)

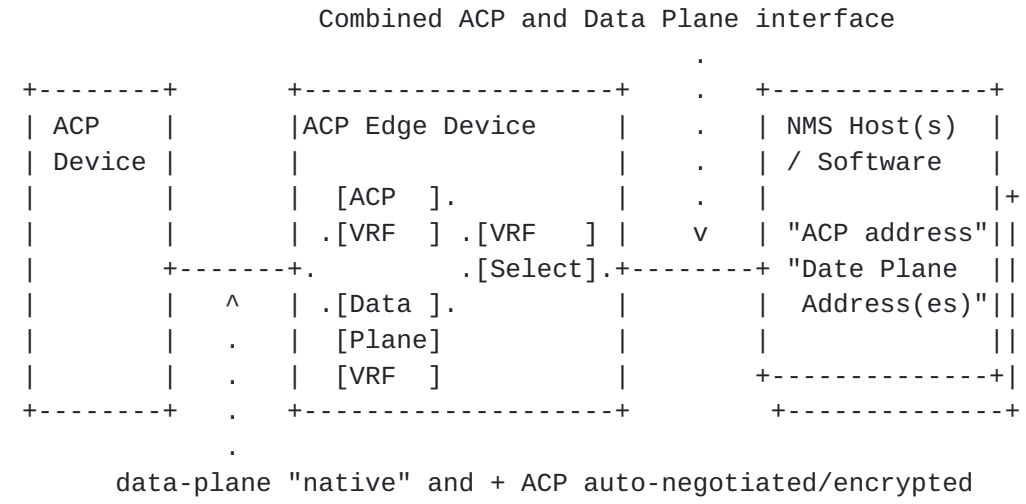

Figure 8: VRF select

Using two physical and/or virtual subnets (and therefore interfaces) into NMS Hosts (as per Section 8.1.1) or Software (as per Section 8.1.2) may be seen as additional complexity, for example with legacy NMS Hosts that support only one IP interface.

To provide a single subnet into both ACP and Data Plane, the ACP Edge device needs to demultiplex packets from NMS hosts into ACP VRF and Data Plane VRF. This is sometimes called "VRF select". If the ACP VRF has no overlapping IPv6 addresses with the Data Plane (as it should), then this function can use the IPv6 Destination address. The problem is Source Address Selection on the NMS Host(s) according to RFC6724.

Consider the simple case: The ACP uses only one ULA prefix, the ACP IPv6 prefix for the Combined ACP and Data Plane interface is covered by that ULA prefix. The ACP edge device announces both the ACP IPv6 prefix and one (or more) prefixes for the data plane. Without further policy configurations on the NMS Host(s), it may select its ACP address as a source address for Data Plane ULA destinations because of Rule 8 of RFC6724. The ACP edge device can pass on the packet to the Data Plane, but the ACP source address should not be used for Data Plane traffic, and return traffic may fail.

If the ACP carries multiple ULA prefixes or non-ULA ACP connect prefixes, then the correct source address selection becomes even more problematic.

With separate ACP connect and Data Plane subnets and RFC4191 prefix announcements that are to be routed across the ACP connect interface, RFC6724 source address selection Rule 5 (use address of outgoing

interface) will be used, so that above problems do not occur, even in more complex cases of multiple ULA and non-ULA prefixes in the ACP routing table.

To achieve the same behavior with a Combined ACP and Data Plane interface, the ACP Edge Device needs to behave as two separate routers on the interface: One link-local IPv6 address/router for its ACP reachability, and one link-local IPv6 address/router for its Data Plane reachability. The Router Advertisements for both are as described above (Section 8.1.3): For the ACP, the ACP prefix is announced together with RFC4191 option for the prefixes routed across the ACP and lifetime=0 to disqualify this next-hop as a default router. For the Data Plane, the Data Plane prefix(es) are announced together with whatever dafault router parameters are used for the Data Plane.

In result, RFC6724 source address selection Rule 5.5 may result in the same correct source address selection behavior of NMS hosts without further configuration on it as the separate ACP connect and Data Plane interfaces. As described in the text for Rule 5.5, this is only a may, because IPv6 hosts are not required to track next-hop information. If an NMS Host does not do this, then separate ACP connect and Data Plane interfaces are the preferrable method of attachment.

ACP edge devices MAY support the Combined ACP and Data Plane interface.

8.1.5. Use of GRASP

GRASP can and should be possible to use across ACP connect interfaces, especially in the architectural correct solution when it is used as a mechanism to connect Software (eg: ASA or legacy NMS applications) to the ACP. Given how the ACP is the security and transport substrate for GRASP, the trustworthyness of devices/ software allowed to participate in the ACP GRASP domain is one of the main reasons why the ACP section describes no solution with non-ACP routers participating in the ACP routing table.

ACP connect interfaces can be dealt with in the GRASP ACP domain like any other ACP interface assuming that any physical ACP connect interface is physically protected from attacks and that the connected Software or NMS Hosts are equally trusted as that on other ACP devices. ACP edge devices SHOULD have options to filter GRASP messages in and out of ACP connect interfaces (permit/deny) and MAY have more fine-grained filtering (eg: based on IPv6 address of originator or objective).

When using "Combined ACP and Data Plane Interfaces", care must be taken that only GRASP messages intended for the ACP GRASP domain received from Software or NMS Hosts are forwarded by ACP edge devices. Currently there is no definition for a GRASP security and transport substrate beside the ACP, so there is no definition how such Software/NMS Host could participate in two separate GRASP Domains across the same subnet (ACP and Data Plane domains). At current it is therefore assumed that all GRASP packets on a Combined ACP and Data Plane interface belong to the GRASP ACP Domain. They must therefore all use the ACP IPv6 addresses of the Software/NMS Hosts. The link-local IPv6 addresses of Software/NMS Hosts (used for GRASP M_DISCOVERY and M_FLOOD messages) are also assumed to belong to the ACP address space.

8.2. ACP through Non-Autonomic L3 Clouds (Remote ACP neighbors)

Not all devices in a network may support the ACP. If non-ACP Layer-2 devices are between ACP nodes, the ACP will work across it since it is IP based. However, the autonomic discovery of ACP neigbhors via DULL GRASP is only intended to work across L2 connections, so it is not sufficient to autonomically create ACP connections across non-ACP Layer-3 devices.

8.2.1. Configured Remote ACP neighbor

On the autonomic device remote ACP neighbors are configured as follows:

```
remote-peer = [ local-address, method, remote-address ]
local-address  = ip-address
remote-address = transport-address
transport-address =
   [ (ip-address | pattern) ?( , protocol ?(, port)) (, pmtu) ]
ip-address = (ipv4-address | ipv6-address )
method = "IKEv2" / "dTLS" / ..
pattern = some IP address set
```

For each candidate configured remote ACP neighbor, the secure channel protocol "method" is configured with its expected local IP address and remote transport endpoint (transport protocol and port number for the remote transport endpoint are usually not necessary to configure if defaults for the secure channel protocol method exist.

This is the same information that would be communicated via DULL for L2 adjacent candidate ACP neighbors. DULL is not used because the remote IP address would need to be configured anyhow and if the remote transport address would not be configured but learned via DULL then this would create a third party attack vector.

The secure channel method leverages the configuration to filter incoming connection requests by the remote IP address. This is suplemental security. The primary security is via the mutual domain certificate based authentication of the secure channel protocol.

On a hub device, the remote IP address may be set to some pattern instead of explicit IP addresses. In this case, the device does not attempt to initiate secure channel connections but only acts as their responder. This allows for simple hub&spoke setups for the ACP where some method (subject to further specification) provisions the transport-address of hubs into spokes and hubs accept connections from any spokes. The typical use case for this are spokes connecting via the Internet to hubs. For example, this would be simple extension to BRSKI to allow zero-touch security across the Internet.

Unlike adjacent ACP neighbor connections, configured remote ACP neighbor connections can also be across IPv4. Not all (future) secure channel methods may support running IPv6 (as used in the ACP across the secure channel connection) over IPv4 encapsulation.

Unless the secure channel method supports PMTUD, it needs to be set up with minimum MTU or the path mtu (pmtu) should be configured.

8.2.2. Tunneled Remote ACP Neighbor

An IPinIP, GRE or other form of pre-existing tunnel is configured between two remote ACP peers and the virtual interfaces representing the tunnel are configured to "ACP enable". This will enable IPv6 link local addresses and DULL on this tunnel. In result, the tunnel is used for normal "L2 adjacent" candidate ACP neighbor discovery with DULL and secure channel setup procedures described in this document.

Tunneled Remote ACP Neighbor requires two encapsulations: the configured tunnel and the secure channel inside of that tunnel. This makes it in general less desirable than Configured Remote ACP Neighbor. Benefits of tunnels are that it may be easier to implement because there is no change to the ACP functionality - just running it over a virtual (tunnel) interface instead of only physical interfaces. The tunnel itself may also provide PMTUD while the secure channel method may not. Or the tunnel mechanism is permitted/ possible through some firewall while the secure channel method may not.

8.2.3. Summary

Configured/Tunneled Remote ACP neighbors are less "indestructible" than L2 adjacent ACP neighbors based on link local addressing, since they depend on more correct data plane operations, such as routing and global addressing.

Nevertheless, these options may be crucial to incrementally deploy the ACP, especially if it is meant to connect islands across the Internet. Implementations SHOULD support at least Tunneled Remote ACP Neighbors via GRE tunnels - which is likely the most common router-to-router tunneling protocol in use today.

Future work could envisage an option where the edge devices of the L3 cloud is configured to automatically forward ACP discovery messages to the right exit point. This optimisation is not considered in this document.

9. Benefits (Informative)

9.1. Self-Healing Properties

The ACP is self-healing:

- New neighbors will automatically join the ACP after successful validation and will become reachable using their unique ULA address across the ACP.

- When any changes happen in the topology, the routing protocol used in the ACP will automatically adapt to the changes and will continue to provide reachability to all devices.

- If an existing device gets revoked, it will automatically be denied access to the ACP as its domain certificate will be validated against a Certificate Revocation List during authentication. Since the revocation check is only done at the establishment of a new security association, existing ones are not automatically torn down. If an immediate disconnect is required, existing sessions to a freshly revoked device can be re-set.

The ACP can also sustain network partitions and mergers. Practically all ACP operations are link local, where a network partition has no impact. Devices authenticate each other using the domain certificates to establish the ACP locally. Addressing inside the ACP remains unchanged, and the routing protocol inside both parts of the ACP will lead to two working (although partitioned) ACPs.

There are few central dependencies: A certificate revocation list (CRL) may not be available during a network partition; a suitable policy to not immediately disconnect neighbors when no CRL is available can address this issue. Also, a registrar or Certificate Authority might not be available during a partition. This may delay renewal of certificates that are to expire in the future, and it may prevent the enrolment of new devices during the partition.

After a network partition, a re-merge will just establish the previous status, certificates can be renewed, the CRL is available, and new devices can be enrolled everywhere. Since all devices use the same trust anchor, a re-merge will be smooth.

Merging two networks with different trust anchors requires the trust anchors to mutually trust each other (for example, by cross-signing). As long as the domain names are different, the addressing will not overlap (see Section 6.10).

It is also highly desirable for implementation of the ACP to be able to run it over interfaces that are administratively down. If this is not feasible, then it might instead be possible to request explicit operator override upon administrative actions that would administratively bring down an interface across whicht the ACP is running. Especially if bringing down the ACP is known to disconnect the operator from the device. For example any such down administrative action could perform a dependency check to see if the transport connection across which this action is performed is affected by the down action (with default RPL routing used, packet forwarding will be symmetric, so this is actually possible to check).

9.2. Self-Protection Properties

9.2.1. From the outside

As explained in Section 6, the ACP is based on secure channels built between devices that have mutually authenticated each other with their domain certificates. The channels themselves are protected using standard encryption technologies like DTLS or IPsec which provide additional authentication during channel establishment, data integrity and data confidentiality protection of data inside the ACP and in addition, provide replay protection.

An attacker will therefore not be able to join the ACP unless having a valid domain certificate, also packet injection and sniffing traffic will not be possible due to the security provided by the encryption protocol.

The ACP also serves as protection (through authentication and encryption) for protocols relevant to OAM that may not have secured protocol stack options or where implementation or deployment of those options fails on some vendor/product/customer limitations. This includes protocols such as SNMP, NTP/PTP, DNS, DHCP, syslog, Radius/Diameter/Tacacs, IPFIX/Netflow - just to name a few. Protection via the ACP secure hop-by-hop channels for these protocols is meant to be only a stopgap though: The ultimate goal is for these and other protocols to use end-to-end encryption utilizing the domain certificate and rely on the ACP secure channels primarily for zero-touch reliable connectivity, but not primarily for security.

The remaining attack vector would be to attack the underlying AN protocols themselves, either via directed attacks or by denial-of-service attacks. However, as the ACP is built using link-local IPv6 address, remote attacks are impossible. The ULA addresses are only reachable inside the ACP context, therefore unreachable from the data plane. Also, the ACP protocols should be implemented to be attack resistant and not consume unnecessary resources even while under attack.

9.2.2. From the inside

The security model of the ACP is based on trusting all members of the group of devices that do receive an ACP domain certificate for the same domain. Attacks from the inside by a compromised group member are therefore the biggest challenge.

Group members must overall the secured so that there are no easy way to compromise them, such as data plane accessible privilege level with simple passwords. This is a lot easier to do in devices whose software is designed from the ground up with security in mind than with legacy software based system where ACP is added on as another feature.

As explained above, traffic across the ACP SHOULD still be end-to-end encrypted whenever possible. This includes traffic such as GRASP, EST and BRSKI inside the ACP. This minimizes man in the middle attacks by compromised ACP group members. Such attackers can not eavesdrop or modify communications, they can just filter them (which is unavoidable by any means).

Further security can be achieved by constraining communication patterns inside the ACP, for example through roles that could be encoded into the domain certificates. This is subject for future work.

9.3. The Administrator View

An ACP is self-forming, self-managing and self-protecting, therefore has minimal dependencies on the administrator of the network. Specifically, since it is independent of configuration, there is no scope for configuration errors on the ACP itself. The administrator may have the option to enable or disable the entire approach, but detailed configuration is not possible. This means that the ACP must not be reflected in the running configuration of devices, except a possible on/off switch.

While configuration is not possible, an administrator must have full visibility of the ACP and all its parameters, to be able to do trouble-shooting. Therefore, an ACP must support all show and debug options, as for any other network function. Specifically, a network management system or controller must be able to discover the ACP, and monitor its health. This visibility of ACP operations must clearly be separated from visibility of data plane so automated systems will never have to deal with ACP aspect unless they explicitly desire to do so.

Since an ACP is self-protecting, a device not supporting the ACP, or without a valid domain certificate cannot connect to it. This means that by default a traditional controller or network management system cannot connect to an ACP. See Section 8.1.1 for more details on how to connect an NMS host into the ACP.

10. Further Considerations (Informative)

The following sections cover topics that are beyond the primary cope of this document (eg: bootstrap), that explain decisions made in this document (eg: choice of GRASP) or that explain desirable extensions to the behavior of the ACP that are not far enough worked out to be already standardized in this document.

10.1. Domain Certificate provisioning / enrollment

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) describes how devices with an IDevID certificate can securely and zero-touch enroll with a domain certificate to support the ACP. BRSKI also leverages the ACP to enable zero touch bootstrap of new devices across networks without any configuration requirements across the transit devices (eg: no DHCP/DS forwarding/server setup). This includes otherwise unconfigured networks as described in Section 3.2. Therefore BRSKI in conjunction with ACP provides for a secure and zero-touch management solution for complete networks. Devices supporting such an infrastructure (BRSKI and ACP) are called ANI devices (Autonomic Networking Infrstructure), see [I-D.ietf-anima-reference-model].

Devices that do not support an IDevID but only an (insecure) vendor specific Unique Device Identifier (UDI) or devices whose manufacturer does not support a MASA could use some future security reduced version of BRSKI.

When BRSKI is used to provision a domain certificate (which is called enrollment), the registrar (acting as an EST server) MUST include the subjectAltName / rfc822Name encoded ACP address and domain name to the enrolling device (called pledge) via its response to the pledges EST CSR Attribute request that is mandatory in BRSKI.

The Certificate Authority in an ACP network MUST not change this, and create the respective subjectAltName / rfc822Name in the certificate. The ACP nodes can therefore find their ACP address and domain using this field in the domain certificate, both for themselves, as well as for other nodes.

The use of BRSKI in conjunction with the ACP can also help to further simplify maintenance and renewal of domain certificates. Instead of relying on CRL, the lifetime of certificates can be made extremely small, for example in the order of hours. When a device fails to connect to the ACP within its certificate lifetime, it can not connect to the ACP to renew its certificate across it, but it can still renew its certificate as an "enrolled/expired pledge" via the BRSKI bootstrap proxy. This requires only that the enhanced EST server that is part of BRSKI honors expired domain certificates and that the pledge first attempts to perform TLS authentication for BRSKI bootstrap with its expired domain certificate - and only reverts to its IDevID when this fails. This mechanism also replaces CRLs because the EST server (in conunction with the CA) would not renew revoked certficates - but in this scheme only the EST-server need to know which certificate was revoked.

In the absence of BRSKI or less secure variants thereof, provisioning of certificates may involve one or more touches or non-standardized automation. Device vendors usually support provisioning of certificates into devices via PKCS#7 (see [RFC2315]) and may support this provisioning through vendor specific models via Netconf ([RFC6241]). If such devices also support Netconf Zerotouch ([I-D.ietf-netconf-zerotouch]) then this can be combined to zero-touch provisioning of domain certificates into devices. Unless there are equivalent integration of Netconf connections across the ACP as there is in BRSKI, this combination would not support zero-touch bootstrap across an unconfigured network though.

10.2. ACP Neighbor discovery protocol selection

This section discusses why GRASP DULL was choosen as the discovery protocol for L2 adjacent candidate ACP neighbors. The contenders considered where GRASP, mDNS or LLDP.

10.2.1. LLDP

LLDP (and Cisco's similar CDP) are example of L2 discovery protocols that terminate their messages on L2 ports. If those protocols would be chosen for ACP neighbor discovery, ACP neighbor discovery would therefore also terminate on L2 ports. This would prevent ACP construction over non-ACP capable but LLDP or CDP enabled L2 switches. LLDP has extensions using different MAC addresses and this could have been an option for ACP discovery as well, but the additional required IEEE standardization and definition of a profile for such a modified instance of LLDP seemed to be more work than the benefit of "reusing the existing protocol" LLDP for this very simple purpose.

10.2.2. mDNS and L2 support

mDNS [RFC6762] with DNS-SD RRs (Resource Records) as defined in [RFC6763] is a key contender as an ACP discovery protocol. because it relies on link-local IP multicast, it does operates at the subnet level, and is also found in L2 switches. The authors of this document are not aware of mDNS implementation that terminate their mDNS messages on L2 ports instead of the subnet level. If mDNS was used as the ACP discovery mechanism on an ACP capable (L3)/L2 switch as outlined in Section 7, then this would be necessary to implement. It is likely that termination of mDNS messages could only be applied to all mDNS messages from such a port, which would then make it necessary to software forward any non-ACP related mDNS messages to maintain prior non-ACP mDNS functionality. Adding support for ACP into such L2 switches with mDNS could therefore create regression problems for prior mDNS functionality on those devices. With low performance of software forwarding in many L2 switches, this could also make the ACP risky to support on such L2 switches.

10.2.3. Why DULL GRASP

LLDP was not considered because of the above mentioned issues. mDNS was not selected because of the above L2 mDNS considerations and because of the following additional points:

If mDNS was not already existing in a device, it would be more work to implement than DULL GRASP, and if an existing implementation of mDNS was used, it would likely be more code space than a separate

implementation of DULL GRASP or a shared implementation of DULL GRASP and GRASP in the ACP.

10.3. Choice of routing protocol (RPL)

This Appendix explains why RPL - "IPv6 Routing Protocol for Low-Power and Lossy Networks ([RFC6550] was chosen as the default (and in this specification only) routing protocol for the ACP. The choice and above explained profile was derived from a pre-standard implementation of ACP that was successfully deployed in operational networks.

Requirements for routing in the ACP are:

- Self-management: The ACP must build automatically, without human intervention. Therefore routing protocol must also work completely automatically. RPL is a simple, self-managing protocol, which does not require zones or areas; it is also self-configuring, since configuration is carried as part of the protocol (see Section 6.7.6 of [RFC6550]).

- Scale: The ACP builds over an entire domain, which could be a large enterprise or service provider network. The routing protocol must therefore support domains of 100,000 nodes or more, ideally without the need for zoning or separation into areas. RPL has this scale property. This is based on extensive use of default routing. RPL also has other scalability improvements, such as selecting only a subset of peers instead of all possible ones, and trickle support for information synchronisation.

- Low resource consumption: The ACP supports traditional network infrastructure, thus runs in addition to traditional protocols. The ACP, and specifically the routing protocol must have low resource consumption both in terms of memory and CPU requirements. Specifically, at edge nodes, where memory and CPU are scarce, consumption should be minimal. RPL builds a destination-oriented directed acyclic graph (DODAG), where the main resource consumption is at the root of the DODAG. The closer to the edge of the network, the less state needs to be maintained. This adapts nicely to the typical network design. Also, all changes below a common parent node are kept below that parent node.

- Support for unstructured address space: In the Autonomic Networking Infrastructure, node addresses are identifiers, and may not be assigned in a topological way. Also, nodes may move topologically, without changing their address. Therefore, the routing protocol must support completely unstructured address

space. RPL is specifically made for mobile ad-hoc networks, with no assumptions on topologically aligned addressing.

- Modularity: To keep the initial implementation small, yet allow later for more complex methods, it is highly desirable that the routing protocol has a simple base functionality, but can import new functional modules if needed. RPL has this property with the concept of "objective function", which is a plugin to modify routing behaviour.

- Extensibility: Since the Autonomic Networking Infrastructure is a new concept, it is likely that changes in the way of operation will happen over time. RPL allows for new objective functions to be introduced later, which allow changes to the way the routing protocol creates the DAGs.

- Multi-topology support: It may become necessary in the future to support more than one DODAG for different purposes, using different objective functions. RPL allow for the creation of several parallel DODAGs, should this be required. This could be used to create different topologies to reach different roots.

- No need for path optimisation: RPL does not necessarily compute the optimal path between any two nodes. However, the ACP does not require this today, since it carries mainly non-delay-sensitive feedback loops. It is possible that different optimisation schemes become necessary in the future, but RPL can be expanded (see point "Extensibility" above).

10.4. Extending ACP channel negotiation (via GRASP)

The mechanism described in the normative part of this document to support multiple different ACP secure channel protocols without a single network wide MTI protocol is important to allow extending secure ACP channel protocols beyond what is specified in this document, but it will run into problem if it would be used for multiple protocols:

The need to potentially have multiple of these security associations even temporarily run in parallel to determine which of them works best does not support the most lightweight implementation options.

The simple policy of letting one side (Alice) decide what is best may not lead to the mutual best result.

The two limitations can easier be solved if the solution was more modular and as few as possible initial secure channel negotiation protocols would be used, and these protocols would then take on the

responsibility to support more flexible objectives to negotiate the mutually preferred ACP security channel protocol.

IKEv2 is the IETF standard protocol to negotiate network security associations. It is meant to be extensible, but it is unclear whether it would be feasible to extend IKEv2 to support possible future requirements for ACP secure channel negotiation:

Consider the simple case where the use of native IPsec vs. IPsec via GRE is to be negotiated and the objective is the maximum throughput. Both sides would indicate some agreed upon performance metric and the preferred encapsulation is the one with the higher performance of the slower side. IKEv2 does not support negotiation with this objective.

Consider dTLS and some form of 802.1AE (MacSEC) are to be added as negotiation options - and the performance objective should work across all IPsec, dDTLS and 802.1AE options. In the case of MacSEC, the negotiation would also need to determine a key for the peering. It is unclear if it would be even appropriate to consider extending the scope of negotiation in IKEv2 to those cases. Even if feasible to define, it is unclear if implementations of IKEv2 would be eager to adopt those type of extension given the long cycles of security testing that necessarily goes along with core security protocols such as IKEv2 implementations.

A more modular alternative to extending IKEv2 could be to layer a modular negotiation mechanism on top of the multitide of existing or possible future secure channel protocols. For this, GRASP over TLS could be considered as a first ACP secure channel negotiation protocol. The following are initial considerations for such an approach. A full specification is subject to a separate document:

To explicitly allow negotiation of the ACP channel protocol, GRASP over a TLS connection using the GRASP_LISTEN_PORT and the devices and peers link-local IPv6 address is used. When Alice and Bob support GRASP negotiation, they do prefer it over any other non-explicitly negotiated security association protocol and should wait trying any non-negotiated ACP channel protocol until after it is clear that GRASP/TLS will not work to the peer.

When Alice and Bob successfully establish the GRASP/TSL session, they will negotiate the channel mechanism to use using objectives such as performance and perceived quality of the security. After agreeing on a channel mechanism, Alice and Bob start the selected Channel protocol. Once the secure channel protocol is successfully running, the GRASP/TLS connection can be kept alive or timed out as long as the selected channel protocol has a secure association between Alice

and Bob. When it terminates, it needs to be re-negotiated via GRASP/TLS.

Notes:

- Negotiation of a channel type may require IANA assignments of code points.
- TLS is subject to reset attacks, which IKEv2 is not. Normally, ACP connections (as specified in this document) will be over link-local addresses so the attack surface for this one issue in TCP should be reduced (note that this may not be true when ACP is tunneled as described in Section 8.2.2.
- GRASP packets received inside a TLS connection established for GRASP/TLS ACP negotiation are assigned to a separate GRASP domain unique to that TLS connection.

10.5. CAs, domains and routing subdomains

There is a wide range of setting up different ACP solution by appropriately using CAs and the domain and rsub elements in the ACP information field of the domain certificate. We summarize these options here as they have been explained in different parts of the document in before and discuss possible and desirable extensions:

An ACP domain is the set of all ACP devices using certificates from the same CA using the same domain field. GRASP inside the ACP is run across all transitively connected ACP devices in a domain.

The rsub element in the ACP information field primarily allows to use addresses from different ULA prefixes. One use case is to create multiple networks that initially may be separated, but where it should be possible to connect them without further extensions to ACP when necessary.

Another use case for routing subdomains is as the starting point for structuring routing inside an ACP. For example, different routing subdomains could run different routing protocols or different instances of RPL and auto-aggregation / distribution of routes could be done across inter routing subdomain ACP channels based on negotiation (eg: via GRASP). This is subject for further work.

RPL scales very well. It is not necessary to use multiple routing subdomains to scale autonomic domains in a way it would be possible if other routing protocols where used. They exist only as options for the above mentioned reasons.

If different ACP domains are to be created that should not allow to connect to each other by default, these ACP domains simply need to have different domain elements in the ACP information field. These domain elements can be arbitrary, including subdomains of one another: Domains "example.com" and "research.example.com" are separate domains if both are domain elements in the ACP information element of certificates.

It is not necessary to have a sparate CA for different ACP domains: an operator can use a single CA to sign certificates for multiple ACP domains that are not allowed to connect to each other because the checks for ACP adjacencies includes comparison of the domain part.

If multiple independent networks choose the same domain name but had their own CA, these would not form a single ACP domain because of CA mismatch. Therefore there is no problem in choosing domain names that are potentially also used by others. Nevertheless it is highly recommended to use domain names that one can have high probability to be unique. It is recommended to use domain names that start with a DNS domain names owned by the assigning organization and unique within it. For example "acp.example.com" if you own "example.com".

Future extensions, primarily through intent can create more flexible options how to build ACP domains.

Intent could modify the ACP connection check to permit connections between different domains.

If different domains use the same CA one would change the ACP setup to permit for the ACP to be established between the two ACP devices, but no routing nor ACP GRASP to be built across this adjacency. The main difference over routing subdomains is to not permit for the ACP GRASP instance to be build across the adjacency. Instead, one would only build a point to point GRASP instance between those peers to negotiate what type of exchanges are desired across that connection. This would include routing negotiation, how much GRASP information to transit and what data-plane forwarding should be done. This approach could also allow for for Intent to only be injected into the network from one side and propagate via this GRASP connection.

If different domains have different CAs, they should start to trust each other by intent injected into both domains that would add the other domains CA as a trust point during the ACP connection setup - and then following up with the previous point of inter-domain connections across domains with the same CA (eg: GRASP negotiation).

11. RFC4291/RFC4193 Updates Considerations

This document may be considered to be updating the IPv6 addressing architecture ([RFC4291]) and/or the Unique Local IPv6 Unicast addresses ([RFC4193]) depending on how strict specific statements in those specs are to be interpreted. This section summarized and explains the necessity and benefits of these changes. The normative parts of this document cover the actual updates.

ACP addresses (Section 6.10) are used by network devices supporting the ACP. They are assigned during bootstrap of the device or initial provisioning of the ACP. They are encoded in the Domain Certificate of the device and are primarily used internally within the network device. In that role they can be thought of as loopback addresses.

Each ACP domain assigns ACP addresses from one or more ULA prefixes. Within an ACP network, addresses are assigned by nodes called registrars. A unique Registrar-ID(entifier) is used in ACP addresses to allow multiple registrars to assign addresses autonomously and uncoordinated from each other. Typically these Registrar-IDs are derived from the IEEE 802 48-bit MAC addresses of registrars.

In the ACP Zone Addressing Sub-Scheme (Section 6.10.3), the registrar assigns a unique 15 bit value to an ACP device. The ACP address has a 64-bit Device-ID(entifier) composed of the 48-bit Registrar-ID, the registrar assigned 15-bit Device-Number and 1 V(irtualization) bit that allows for an ACP device to have two addresses.

The 64-bit Device-Identifier in the ACP Zone Addressing Sub-Scheme matches the 64 bit Interface Identifier (IID) address part as specified in RFC4291 section 2.5.1. IIDs are unique across ACP devices, but all ACP devices with the same ULA prefix that use the ACP Zone Addressing Sub-Scheme will share the same subnet prefix (according to the definition of that term in RFC4291). Each ACP device injects a /127 route into the ACP routing table to cover its two assigned addresses (V(irtual) bit being 0 or 1). This approach is used because these ACP addresses are identifiers and not locators. The ACP device can connect anywhere in the autonomic domain without having to change its addresses. A lightweight, highly scaleable routing protocol (RPL) is used to allow for large scale ACP networks.

It is possible, that this scheme constitutes an update to RFC4191 because the same 64 bit subnet prefix is used across many ACP devices. The ACP Zone addressing Sub-Scheme is very similar to the common operational practices of assigning /128 loopback addresses to network devices from the same /48 or /64 subnet prefix.

In the ACP Vlong Addressing Sub-Scheme (Section 6.10.5), the address elements are the same as described for the Zone Addressing Sub-Scheme, but the V field is expanded from 1 bit to 8 or 16 bits. The ACP device with ACP Vlong addressing therefore injects /120 or /112 prefixes into the ACP routing table to cover its internal addresses.

The goal for the 8 or 16-bit addresses available to an ACP device in this scheme is to assign them as required to software components, which in autonomic networking are called ASA (Autonomic Service Agents). It could equally be used for existing software components such as VNFs (Virtual Networking Functions). The ACP interface would then be the out-of-band management interface of such a VNF software component. It should especially be possible to use these software components in a variety of contexts to allow standardized modular system composition: Native applications (in some VRF context if available), containers, virtual machines or other future ones. To modularily compose a system with containers and virtual machines and avoid problems such as port space collision or NAT, it is necessary not only to assign separate addresses to those contexts, but also to use the notion of virtual interfaces between these contexts to compose the system.

In practical terms, the ACP should be enabled to create from its /8 or /16 prefix one or more device internal virtual subnets and to start software components connected to those virtual subnets. Ideally, these software components should be able to autoconfigure their addresses on these virtual interfaces. Future work has to determine whether this address autoconfiguration for the virtual interface is best done with DHCPv6, if SLAAC should be recommended for these /8 or /16 virtual interfaces, or if some additional standardized method would be required.

In the ACP Vlong Addressing scheme, the Device-ID does not match the RFC4291/RFC4193 64 bith length for the Interface Identifier, so this addressing Sub-Scheme in the ACP is an update to both standards.

This document also specifies the workaround solution of exposing the ACP on physical interfaces in support of adoption by existing hardware and software solutions. A NOC based NMS host could for example be configured with a second physical interface connecting to an ACP device that exposes the ACP to that NMS host (called ACP edge device). The desired evolution of this workaround is that those two functions would merge into a single device, for example by making the ACP router a container/virtual machine inside the NMS host or vice versa. The addressing for those physical interfaces allows for manually configured address prefixes but it could be fully autonomous if it could leverage the Vlong addressing format. That would result

in a non /64 IID boundary on those external interfaces (but instead in /112 or /120 subnet prefixes).

Note that both in the internal as well as the workaround external use of ACP addresses, all ACP addresses are meant to be used exclusively by components that are part of network control and OAM, but not for network users such as normal hosts. This implies that for example no requirements for privacy addressing have been identified for ACP addresses.

12. Security Considerations

An ACP is self-protecting and there is no need to apply configuration to make it secure. Its security therefore does not depend on configuration.

However, the security of the ACP depends on a number of other factors:

- The usage of domain certificates depends on a valid supporting PKI infrastructure. If the chain of trust of this PKI infrastructure is compromised, the security of the ACP is also compromised. This is typically under the control of the network administrator.

- Security can be compromised by implementation errors (bugs), as in all products.

There is no prevention of source-address spoofing inside the ACP. This implies that if an attacker gains access to the ACP, it can spoof all addresses inside the ACP and fake messages from any other device.

Fundamentally, security depends on correct operation, implementation and architecture. Autonomic approaches such as the ACP largely eliminate the dependency on correct operation; implementation and architectural mistakes are still possible, as in all networking technologies.

13. IANA Considerations

14. Acknowledgements

This work originated from an Autonomic Networking project at Cisco Systems, which started in early 2010. Many people contributed to this project and the idea of the Autonomic Control Plane, amongst which (in alphabetical order): Ignas Bagdonas, Parag Bhide, Balaji BL, Alex Clemm, Yves Hertoghs, Bruno Klauser, Max Pritikin, Michael Richardson, Ravi Kumar Vadapalli.

Special thanks to Pascal Thubert and Michael Richardson to provide the details for the recommendations of the use of RPL in the ACP

Further input and suggestions were received from: Rene Struik, Brian Carpenter, Benoit Claise.

15. Change log [RFC Editor: Please remove]

15.1. Initial version

First version of this document: draft-behringer-autonomic-control-plane

15.2. draft-behringer-anima-autonomic-control-plane-00

Initial version of the anima document; only minor edits.

15.3. draft-behringer-anima-autonomic-control-plane-01

- Clarified that the ACP should be based on, and support only IPv6.
- Clarified in intro that ACP is for both, between devices, as well as for access from a central entity, such as an NMS.
- Added a section on how to connect an NMS system.
- Clarified the hop-by-hop crypto nature of the ACP.
- Added several references to GDNP as a candidate protocol.
- Added a discussion on network split and merge. Although, this should probably go into the certificate management story longer term.

15.4. draft-behringer-anima-autonomic-control-plane-02

Addresses (numerous) comments from Brian Carpenter. See mailing list for details. The most important changes are:

- Introduced a new section "overview", to ease the understanding of the approach.
- Merged the previous "problem statement" and "use case" sections into a mostly re-written "use cases" section, since they were overlapping.
- Clarified the relationship with draft-ietf-anima-stable-connectivity

15.5. draft-behringer-anima-autonomic-control-plane-03

- Took out requirement for IPv6 --> that's in the reference doc.
- Added requirement section.
- Changed focus: more focus on autonomic functions, not only virtual out of band. This goes a bit throughout the document, starting with a changed abstract and intro.

15.6. draft-ietf-anima-autonomic-control-plane-00

No changes; re-submitted as WG document.

15.7. draft-ietf-anima-autonomic-control-plane-01

- Added some paragraphs in addressing section on "why IPv6 only", to reflect the discussion on the list.
- Moved the data-plane ACP out of the main document, into an appendix. The focus is now the virtually separated ACP, since it has significant advantages, and isn't much harder to do.
- Changed the self-creation algorithm: Part of the initial steps go into the reference document. This document now assumes an adjacency table, and domain certificate. How those get onto the device is outside scope for this document.
- Created a new section 6 "workarounds for non-autonomic nodes", and put the previous controller section (5.9) into this new section. Now, section 5 is "autonomic only", and section 6 explains what to do with non-autonomic stuff. Much cleaner now.
- Added an appendix explaining the choice of RPL as a routing protocol.
- Formalised the creation process a bit more. Now, we create a "candidate peer list" from the adjacency table, and form the ACP with those candidates. Also it explains now better that policy (Intent) can influence the peer selection. (section 4 and 5)
- Introduce a section for the capability negotiation protocol (section 7). This needs to be worked out in more detail. This will likely be based on GRASP.
- Introduce a new parameter: ACP tunnel type. And defines it in the IANA considerations section. Suggest GRE protected with IPSec transport mode as the default tunnel type.

- Updated links, lots of small edits.

15.8. draft-ietf-anima-autonomic-control-plane-02

- Added explicitly text for the ACP channel negotiation.

- Merged draft-behringer-anima-autonomic-addressing-02 into this document, as suggested by WG chairs.

15.9. draft-ietf-anima-autonomic-control-plane-03

- Changed Neighbor discovery protocol from GRASP to mDNS. Bootstrap protocol team decided to go with mDNS to discover bootstrap proxy, and ACP should be consistent with this. Reasons to go with mDNS in bootstrap were a) Bootstrap should be reuseable also outside of full anima solutions and introduce as few as possible new elements. mDNS was considered well-known and very-likely even pre-existing in low-end devices (IoT). b) Using GRASP both for the insecure neighbor discovery and secure ACP operatations raises the risk of introducing security issues through implementation issues/ non-isolation between those two instances of GRASP.

- Shortened the section on GRASP instances, because with mDNS being used for discovery, there is no insecure GRASP session any longer, simplifying the GRASP considerations.

- Added certificate requirements for ANIMA in section 5.1.1, specifically how the ANIMA information is encoded in subjectAltName.

- Deleted the appendix on "ACP without separation", as originally planned, and the paragraph in the main text referring to it.

- Deleted one sub-addressing scheme, focusing on a single scheme now.

- Included information on how ANIMA information must be encoded in the domain certificate in section "preconditions".

- Editorial changes, updated draft references, etc.

15.10. draft-ietf-anima-autonomic-control-plane-04

Changed discovery of ACP neighbor back from mDNS to GRASP after revisiting the L2 problem. Described problem in discovery section itself to justify. Added text to explain how ACP discovery relates to BRSKY (bootstrap) discovery and pointed to Michael Richardsons draft detailing it. Removed appendix section that contained the

original explanations why GRASP would be useful (current text is meant to be better).

15.11. draft-ietf-anima-autonomic-control-plane-05

- Section 5.3 (candidate ACP neighbor selection): Add that Intent can override only AFTER an initial default ACP establishment.
- Section 6.10.1 (addressing): State that addresses in the ACP are permanent, and do not support temporary addresses as defined in RFC4941.
- Modified Section 6.3 to point to the GRASP objective defined in draft-carpenter-anima-ani-objectives. (and added that reference)
- Section 6.10.2: changed from MD5 for calculating the first 40 bits to SHA256; reason is MD5 should not be used any more.
- Added address sub-scheme to the IANA section.
- Made the routing section more prescriptive.
- Clarified in Section 8.1.1 the ACP Connect port, and defined that term "ACP Connect".
- Section 8.2: Added some thoughts (from mcr) on how traversing a L3 cloud could be automated.
- Added a CRL check in Section 6.7.
- Added a note on the possibility of source-address spoofing into the security considerations section.
- Other editoral changes, including those proposed by Michael Richardson on 30 Nov 2016 (see ANIMA list).

15.12. draft-ietf-anima-autonomic-control-plane-06

- Added proposed RPL profile.
- detailed dTLS profile - dTLS with any additional negotiation/ signaling channel.
- Fixed up text for ACP/GRE encap. Removed text claiming its incompatible with non-GRE IPsec and detailled it.
- Added text to suggest admin down interfaces should still run ACP.

15.13. draft-ietf-anima-autonomic-control-plane-07

- Changed author association.

- Improved ACP connect setion (after confusion about term came up in the stable connectivity draft review). Added picture, defined complete terminology.

- Moved ACP channel negotiation from normative section to appendix because it can in the timeline of this document not be fully specified to be implementable. Aka: work for future document. That work would also need to include analysing IKEv2 and describin the difference of a proposed GRASP/TLS solution to it.

- Removed IANA request to allocate registry for GRASP/TLS. This would come with future draft (see above).

- Gave the name "ACP information field" to the field in the certificate carrying the ACP address and domain name.

- Changed the rules for mutual authentication of certificates to rely on the domain in the ACP information field of the certificate instead of the OU in the certificate. Also renewed the text pointing out that the ACP information field in the certificate is meant to be in a form that it does not disturb other uses of the certificate. As long as the ACP expected to rely on a common OU across all certificates in a domain, this was not really true: Other uses of the certificates might require different OUs for different areas/type of devices. With the rules in this draft version, the ACP authentication does not rely on any other fields in the certificate.

- Added an extension field to the ACP information field so that in the future additional fields like a subdomain could be inserted. An example using such a subdomain field was added to the pre-existing text suggesting sub-domains. This approach is necessary so that there can be a single (main) domain in the ACP information field, because that is used for mutual authentication of the certificate. Also clarified that only the register(s) SHOULD/MUST use that the ACP address was generated from the domain name - so that we can easier extend change this in extensions.

- Took the text for the GRASP discovery of ACP neighbors from Brians grasp-ani-objectives draft. Alas, that draft was behind the latest GRASP draft, so i had to overhaul. The mayor change is to describe in the ACP draft the whole format of the M_FLOOD message (and not only the actual objective). This should make it a lot easier to read (without having to go back and forth to the GRASP

RFC/draft). It was also necessary because the locator in the M_FLOOD messages has an important role and its not coded inside the objective. The specification of how to format the M_FLOOD message shuold now be complete, the text may be some duplicate with the DULL specificateion in GRASP, but no contradiction.

- One of the main outcomes of reworking the GRASP section was the notion that GRASP announces both the candidate peers IPv6 link local address but also the support ACP security protocol including the port it is running on. In the past we shied away from using this information because it is not secured, but i think the additional attack vectors possible by using this information are negligible: If an attacker on an L2 subnet can fake another devices GRASP message then it can already provide a similar amount of attack by purely faking the link-local address.

- Removed the section on discovery and BRSKI. This can be revived in the BRSKI document, but it seems mood given how we did remove mDNS from the latest BRSKI document (aka: this section discussed discrepancies between GRASP and mDNS discovery which should not exist anymore with latest BRSKI.

- Tried to resolve the EDNOTE about CRL vs. OCSP by pointing out we do not specify which one is to be used but that the ACP should be used to reach the URL included in the certificate to get to the CRL storage or OCSP server.

- Changed ACP via IPsec to ACP via IKEv2 and restructured the sections to make IPsec native and IPsec via GRE subsections.

- No need for any assigned dTLS port if ACP is run across dTLS because it is signalled via GRASP.

15.14. draft-ietf-anima-autonomic-control-plane-08

Modified mentioning of BRSKI to make it consistent with current (07/2017) target for BRSKI: MASA and IDevID are mandatory. Devices with only insecure UDI would need a security reduced variant of BRSKI. Also added mentioning of Netconf Zerotouch. Made BRSKI non-normative for ACP because wrt. ACP it is just one option how the domain certificate can be provisioned. Instead, BRSKI is mandatory when a device implements ANI which is ACP+BRSKI.

Enhanced text for ACP across tunnels to decribe two options: one across configured tunnels (GRE, IPinIP etc) a more efficient one via directed DULL.

Moved decription of BRSKI to appendex to emphasize that BRSKI is not a (normative) dependency of GRASP, enhanced text to indicate other options how Domain Certificates can be provisioned.

Added terminology section.

Separated references into normative and non-normative.

Enhanced section about ACP via "tunnels". Defined an option to run ACP secure channel without an outer tunnel, discussed PMTU, benefits of tunneling, potential of using this with BRSKI, made ACP via GREP a SHOULD requirement.

Moved appendix sections up before IANA section because there where concerns about appendices to be to far on the bottom to be read. Added (Informative) / (Normative) to section titles to clarify which sections are informative and which are normative

Moved explanation of ACP with L2 from precondition to separate section before workarounds, made it instructive enough to explain how to implement ACP on L2 ports for L3/L2 switches and made this part of normative requirement (L2/L3 switches SHOULD support this).

Rewrote section "GRASP in the ACP" to define GRASP in ACP as mandatory (and why), and define the ACP as security and transport substrate to GRASP in ACP. And how it works.

Enhanced "self-protection" properties section: protect legacy management protocols. Security in ACP is for protection from outside and those legacy protocols. Otherwise need end-to-end encryption also inside ACP, eg: with domain certificate.

Enhanced initial domain certificate section to include requirements for maintenance (renewal/revocation) of certificates. Added explanation to BRSKI informative section how to handle very short lived certificates (renewal via BRSKI with expired cert).

Modified the encoding of the ACP address to better fit RFC822 simple local-parts (":" as required by RFC5952 are not permitted in simple dot-atoms according to RFC5322. Removed reference to RFC5952 as its now not needed anymore.

Introduced a sub-domain field in the ACP information in the certificate to allow defining such subdomains with depending on future Intent definitions. It also makes it clear what the "main domain" is. Scheme is called "routing subdomain" to have a unique name.

Added V8 (now called Vlong) addressing sub-scheme according to suggestion from mcr in his mail from 30 Nov 2016 (https://mailarchive.ietf.org/arch/msg/anima/nZpEphrTqDCBdzsKMpaIn2gsIzI). Also modified the explanation of the single V bit in the first sub-scheme now renamed to Zone sub-scheme to distinguish it.

15.15. draft-ietf-anima-autonomic-control-plane-09

Added reference to RFC4191 and explained how it should be used on ACP edge routers to allow autoconfiguration of routing by NMS hosts. This came after review of stable connectivity draft where ACP connect is being referred to.

V8 addressing Sub-Scheme was modified to allow not only /8 device-local address space but also /16. This was in response to the possible need to have maybe as much as 2^12 local addresses for future encaps in BRSKI like IPinIP. It also would allow fully autonomic address assignment for ACP connect interfaces from this local address space (on an ACP edge device), subject to approval of the implied update to rfc4291/rfc4193 (IID length). Changed name to Vlong addressing sub-scheme.

Added text in response to Brian Carpenters review of draft-ietf-anima-stable-connectivity-04. The stable connectivity draft was vaguely describing ACP connect behavior that is better standardized in this ACP draft:

- Added new ACP "Manual" addressing sub-scheme with /64 subnets for use with ACP connect interfaces. Being covered by the ACP ULA prefix, these subnets do not require additional routing entries for NMS hosts. They also are fully 64-bit IID length compliant and therefore not subject to 4191bis considerations. And they avoid that operators manually assign prefixes from the ACP ULA prefixes that might later be assigned autonomiously.

- ACP connect auto-configuration: Defined that ACP edge devices, NMS hosts should use RFC4191 to automatically learn ACP prefixes. This is especially necessary when the ACP uses multiple ULA prefixes (via eg: the rsub domain certificate option), or if ACP connect subinterfaces use manually configured prefixes NOT covered by the ACP ULA prefixes.

- Explained how rfc6724 is (only) sufficient when the NMS host has a separate ACP connect and data plane interface. But not when there is a single interface.

- Added a separate subsection to talk about "software" instead of "NMS hosts" connecting to the ACP via the "ACP connect" method. The reason is to point out that the "ACP connect" method is not only a workaround (for NMS hosts), but an actual desirable long term architectural component to modularily build software (eg: ASA or OAM for VNF) into ACP devices.

- Added a section to define how to run ACP connect across the same interface as the Data Plane. This turns out to be quite challenging because we only want to rely on existing standards for the network stack in the NMS host/software and only define what features the ACP edge device needs.

- Added section about use of GRASP over ACP connect.

- Added text to indicate packet processing/filtering for security: filter incorrect packets arriving on ACP connect interfaces, diagnose on RPL root packets to incorrect destination address (not in ACP connect section, but because of it).

- Reaffirm security goal of ACP: Do not permit non-ACP routers into ACP routing domain.

Made this ACP document be an update to RFC4291 and RFC4193. At the core, some of the ACP addressing sub-schemes do effectively not use 64-bit IIDs as required by RFC4191 and debated in rfc4191bis. During 6man in prague, it was suggested that all documents that do not do this should be classified as such updates. Add a rather long section that summarizes the relevant parts of ACP addressing and usage and. Aka: This section is meant to be the primary review section for readers interested in these changes (eg: 6man WG.).

Added changes from Michael Richardsons review https://github.com/anima-wg/autonomic-control-plane/pull/3/commits, textual and:

- ACP discovery inside ACP is bad *doh*!.

- Better CA trust and revocation sentences.

- More details about RPL behavior in ACP.

- blackhole route to avoid loops in RPL.

Added requirement to terminate ACP channels upon cert expiry/revocation.

Added fixes from 08-mcr-review-reply.txt (on github):

- AN Domain Names are FQDNs.
- Fixed bit length of schemes, numerical writing of bits (00b/01b).
- Lets use US american english.

16. References

16.1. Normative References

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-15 (work in progress), July 2017.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <http://www.rfc-editor.org/info/rfc4191>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <http://www.rfc-editor.org/info/rfc4862>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <http://www.rfc-editor.org/info/rfc6550>.

[RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <http://www.rfc-editor.org/info/rfc6552>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <http://www.rfc-editor.org/info/rfc7030>.

[RFC7676] Pignataro, C., Bonica, R., and S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC 7676, DOI 10.17487/RFC7676, October 2015, <http://www.rfc-editor.org/info/rfc7676>.

16.2. Informative References

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-07 (work in progress), July 2017.

[I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Pierre, P., Liu, B., Nobre, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-04 (work in progress), July 2017.

[I-D.ietf-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-ietf-anima-stable-connectivity-04 (work in progress), July 2017.

[I-D.ietf-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-14 (work in progress), June 2017.

[I-D.ietf-roll-useofrplinfo]
Robles, I., Richardson, M., and P. Thubert, "When to use RFC 6553, 6554 and IPv6-in-IPv6", draft-ietf-roll-useofrplinfo-16 (work in progress), July 2017.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <http://www.rfc-editor.org/info/rfc2315>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <http://www.rfc-editor.org/info/rfc6553>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <http://www.rfc-editor.org/info/rfc6763>.

[RFC7404] Behringer, M. and E. Vyncke, "Using Only Link-Local Addressing inside an IPv6 Network", RFC 7404, DOI 10.17487/RFC7404, November 2014, <http://www.rfc-editor.org/info/rfc7404>.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <http://www.rfc-editor.org/info/rfc7575>.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, June 2015, <http://www.rfc-editor.org/info/rfc7576>.

Authors' Addresses

Michael H. Behringer (editor)

Email: michael.h.behringer@gmail.com

Toerless Eckert (editor)
Futurewei Technologies Inc.
2330 Central Expy
Santa Clara 95050
USA

Email: tte+ietf@cs.fau.de

Steinthor Bjarnason
Arbor Networks
2727 South State Street, Suite 200
Ann Arbor MI 48104
United States

Email: sbjarnason@arbor.net