ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: June 20, 2018

T. Eckert, Ed.
Huawei
M. Behringer, Ed.

S. Bjarnason
Arbor Networks
December 17, 2017

An Autonomic Control Plane (ACP)
draft-ietf-anima-autonomic-control-plane-13

Abstract

Autonomic functions need a control plane to communicate, which depends on some addressing and routing. This Autonomic Management and Control Plane should ideally be self-managing, and as independent as possible of configuration. This document defines such a plane and calls it the "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 (Operations Administration and Management) communications over a network that is secure and reliable even when the network is not configured, or not misconfigured.

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 https://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 June 20, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 4
2. Acronyms and Terminology 6
3. Use Cases for an Autonomic Control Plane 11
3.1. An Infrastructure for Autonomic Functions 11
3.2. Secure Bootstrap over a not configured Network 11
3.3. Data-Plane Independent Permanent Reachability 11
4. Requirements 12
5. Overview 13
6. Self-Creation of an Autonomic Control Plane (ACP) (Normative) 14
6.1. ACP Domain, Certificate and Network 15
6.1.1. Certificate Domain Information Field 16
6.1.2. ACP domain membership check 18
6.1.3. Certificate Maintenance 19
6.2. ACP Adjacency Table 21
6.3. Neighbor Discovery with DULL GRASP 21
6.4. Candidate ACP Neighbor Selection 24
6.5. Channel Selection 25
6.6. Candidate ACP Neighbor verification 26
6.7. Security Association protocols 27
6.7.1. ACP via IKEv2 27
6.7.2. ACP via dTLS 28
6.7.3. ACP Secure Channel Requirements 28
6.8. GRASP in the ACP 29
6.8.1. GRASP as a core service of the ACP 29
6.8.2. ACP as the Security and Transport substrate for GRASP 30
6.9. Context Separation 33
6.10. Addressing inside the ACP 34
6.10.1. Fundamental Concepts of Autonomic Addressing 34
6.10.2. The ACP Addressing Base Scheme 35
6.10.3. ACP Zone Addressing Sub-Scheme 36
6.10.4. ACP Manual Addressing Sub-Scheme 39
6.10.5. ACP Vlong Addressing Sub-Scheme 40
6.10.6. Other ACP Addressing Sub-Schemes 41
6.11. Routing in the ACP 41
6.11.1. RPL Profile 42
6.12. General ACP Considerations 45
6.12.1. Performance 45
6.12.2. Addressing of Secure Channels in the data-plane . . 46

6.12.3. MTU 46
6.12.4. Multiple links between nodes 47
6.12.5. ACP interfaces 47
7. ACP support on L2 switches/ports (Normative) 50
7.1. Why 50
7.2. How (per L2 port DULL GRASP) 51
8. Support for Non-ACP Components (Normative) 53
8.1. ACP Connect 53
8.1.1. Non-ACP Controller / NMS system 53
8.1.2. Software Components 55
8.1.3. Auto Configuration 56
8.1.4. Combined ACP/Data-Plane Interface (VRF Select) . . . 57
8.1.5. Use of GRASP 58
8.2. ACP through Non-ACP L3 Clouds (Remote ACP neighbors) . . 59
8.2.1. Configured Remote ACP neighbor 59
8.2.2. Tunneled Remote ACP Neighbor 61
8.2.3. Summary 61
9. Benefits (Informative) 61
9.1. Self-Healing Properties 61
9.2. Self-Protection Properties 63
9.2.1. From the outside 63
9.2.2. From the inside 63
9.3. The Administrator View 64
10. Further Considerations (Informative) 65
10.1. BRSKI Bootstrap (ANI) 65
10.2. ACP (and BRSKI) Diagnostics 66
10.3. Enabling and disabling ACP/ANI 71
10.3.1. Filtering for non-ACP/ANI packets 71
10.3.2. Admin Down State 72
10.3.3. Interface level ACP/ANI enable 74
10.3.4. Which interfaces to auto-enable ? 75
10.3.5. Node Level ACP/ANI enable 76
10.3.6. Undoing ANI/ACP enable 78
10.3.7. Summary 78
10.4. ACP Neighbor discovery protocol selection 78
10.4.1. LLDP 79
10.4.2. mDNS and L2 support 79
10.4.3. Why DULL GRASP 79
10.5. Choice of routing protocol (RPL) 80
10.6. Extending ACP channel negotiation (via GRASP) 81
10.7. CAs, domains and routing subdomains 83
10.8. Adopting ACP concepts for other environments 84
11. Security Considerations 86
12. IANA Considerations 87
13. Acknowledgements 88
14. Change log [RFC Editor: Please remove] 88
14.1. Initial version 88
14.2. draft-behringer-anima-autonomic-control-plane-00 88

14.3. draft-behringer-anima-autonomic-control-plane-01 88
14.4. draft-behringer-anima-autonomic-control-plane-02 89
14.5. draft-behringer-anima-autonomic-control-plane-03 89
14.6. draft-ietf-anima-autonomic-control-plane-00 89
14.7. draft-ietf-anima-autonomic-control-plane-01 89
14.8. draft-ietf-anima-autonomic-control-plane-02 90
14.9. draft-ietf-anima-autonomic-control-plane-03 90
14.10. draft-ietf-anima-autonomic-control-plane-04 91
14.11. draft-ietf-anima-autonomic-control-plane-05 91
14.12. draft-ietf-anima-autonomic-control-plane-06 92
14.13. draft-ietf-anima-autonomic-control-plane-07 92
14.14. draft-ietf-anima-autonomic-control-plane-08 94
14.15. draft-ietf-anima-autonomic-control-plane-09 95
14.16. draft-ietf-anima-autonomic-control-plane-10 97
14.17. draft-ietf-anima-autonomic-control-plane-11 99
14.18. draft-ietf-anima-autonomic-control-plane-12 99
14.19. draft-ietf-anima-autonomic-control-plane-13 101
15. References 103
15.1. Normative References 103
15.2. Informative References 105
Authors' Addresses 109

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 an autonomically built communications infrastructure or network plane (there is no well-established name for this). This infrastructure needs to be secure, resilient and re-usable by all autonomic functions. Section 5 of [RFC7575] introduces that infrastructure and calls it the "Autonomic Control Plane" (ACP). More descriptively it would be the "Autonomic communications infrastructure for Management and Control". For naming consistency with that prior document, this document continues to use the name ACP though.

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 expensive.

In increasingly automated networks either centralized management systems or distributed autonomic service agents in the network require a control plane which is independent of the configuration of the network they manage, to avoid impacting their own operations through the configuration actions they take.

This document describes a modular design for a self-forming, self-managing and self-protecting "Autonomic Control Plane" (ACP) which is a virtual in-band network designed to be as independent as possible of configuration, addressing and routing problems. The details how this achieved are defined in Section 6. The ACP is designed to remains operational even in the presence of configuration errors, addressing or routing issues, or where policy could inadvertently affect connectivity of both data packets or control packets.

This document uses the term "data-plane" to refer to anything in the network nodes that is not the ACP, and therefore considered to be dependent on (mis-)configuration. This data-plen includes both the traditional forwarding-plane, as well as any pre-existing control-plane, such as routing protocols that establish routing tables for the forwarding plane.

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 and transport substrate".

- An operator can use it to log into remote devices, even if the network is misconfigured or not configured.

- 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 these use cases for the ACP in Section 3, it defines the requirements in Section 4. Section 5 gives an overview how the ACP is constructed, and in Section 6 the process is defined in detail. Section 7 defines how to support ACP on L2 switches. Section 8 explains how non-ACP nodes and networks can be integrated.

The following sections are non-normative: Section 9 reviews benefits of the ACP (after all the details have been defined), Section 10 provides additional explanations and describes additional details or future work possibilities that where considered not to be appropriate for standardization in this document but nevertheless assumed to be helpful for candidate adopters of the ACP.

The ACP as defined in this document can be implemented and operated without dependency against other components of autonomic networks except for the GRASP protocol on which it depends. The document "Autonomic Network Stable Connectivity" [I-D.ietf-anima-stable-connectivity] describes how the ACP alone can be used to provide stable connectivity for autonomic and non-autonomic OAM applications ("Operations Administration and Management"). It also explains on how existing management solutions can leverage the ACP in parallel with traditional management models, when to use the ACP and, how to integrate IPv4 based management, etc.

Combining ACP with BRSKI ("Bootstrapping Remote Secure Key Infrastructures", see [I-D.ietf-anima-bootstrapping-keyinfra]) results in the "Autonomic Network Infrastructure" as defined in [I-D.ietf-anima-reference-model]. which provides autonomic connectivity (from ACP) with full secure zero touch bootstrap (from BRSKI). The ANI itself does not constitute an Autonomic Network, but it enables building more or less autonomic networks on top of it - using either centralized, SDN ("Software Defined Networking", see [RFC7426]) style automation or distributed automation via ASA ("Autonomic Service Agents") / "Autonomic Functions" - or a mixture of both. See [I-D.ietf-anima-reference-model] for more information.

2. Acronyms and Terminology

In the rest of the document we will refer to systems using the ACP as "nodes". Typically such a node is a physical (network equipment) device, but it can equally be some virtualized system. Therefore, we do not refer to them as devices unless the context specifically calls for a physical system.

This document introduces or uses the following terms (sorted alphabetically). Terms introduced are explained on first use, so this list is for reference only.

ACP: "Autonomic Control Plane". The Autonomic Function as defined in this document. It provides secure zero-touch transitive (network wide) IPv6 connectivity for all nodes in the same ACP domain as well as a GRASP instance running across this ACP IPv6 connectivity. 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 address assigned to the ACP node. It is stored in the domain information field of the ACP domain certificate (Paragraph 21).

ACP address range/set: The ACP address may imply a range or set of addresses that the node can assign for different purposes. This address range/set is derived by the node from the format of the ACP address called the "addressing sub-scheme".

ACP connect: An interface on an ACP node providing access to the ACP for non ACP capable nodes without using an ACP secure channel. See Section 8.1.1.

ACP domain: The ACP domain is the set of nodes with domain certificates that allow them to authenticate each other as members of the ACP domain. See Section 6.1.2.

domain information (field): An rfc822Name information element (e.g.: field) in the domain certificate in which the ACP relevant information is encoded: the domain name and the ACP address.

ACP loopback interface: The loopback interface in the ACP VRF that has the ACP address assigned to it.

ACP network: The ACP network constitutes all the nodes that have access to the ACP. It is the set of active and transitively connected nodes of an ACP domain plus all nodes that get access to the ACP of that domain via ACP edge nodes.

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 security association established hop-by-hop between adjacent ACP nodes to carry traffic of the ACP VRF separated from data-plane traffic in-band over the same links as the data-plane.

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

ACP virtual interface: An interface in the ACP VRF mapped to one or more ACP secure channels. See Section 6.12.5.

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

(AN) Domain Name: An FQDN (Fully Qualified Domain Name) in the domain information field of the Domain Certificate. See Section 6.1.1.

ANI (nodes/network): "Autonomic Network Infrastructure". The ANI is the infrastructure to enable Autonomic Networks. It includes ACP, BRSKI and GRASP. Every Autonomic Network includes the ANI, but not every ANI network needs to include autonomic functions beyond the ANI (nor intent). An ANI network without further autonomic functions can for example support secure zero touch bootstrap and stable connectivity for SDN networks - see [I-D.ietf-anima-stable-connectivity].

ANIMA: "Autonomic Networking Integrated Model and Approach". ACP, BRSKI and GRASP are products of the IETF ANIMA working group.

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 nodes.

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 nodes use ACP, BRSKI and GRASP.

data-plane: The counterpoint to the ACP VRF in an ACP node: all VRFs other than the ACP VRF. In a simple ACP or ANI node, the data-plane is typically provisioned non-autonomic, for example manually (including across the ACP) or via SDN controllers. In a full Autonomic Network node, the data-plane is managed autonomically via Autonomic Functions and Intent. Note that other (non-ANIMA) RFC use the data-plane to refer to what is better called the forwarding plane. This is not the way the term is used in this document!

ACP (ANI/AN) Domain Certificate: A provisioned certificate (LDevID) carrying the domain information field which is used by the ACP to

learn its address in the ACP and to derive and cryptographically assert its membership in the ACP domain.

device: A physical system, or physical node.

Enrollment: The process where a node 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 node 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 node in the context of its vendor/manufacturer such as device model/type and serial number. See [AR8021].

Intent: Northbound operator and automation facing interface of an Autonomic Network according to [I-D.ietf-anima-reference-model].

loopback interface: The conventional name for an internal IP interface to which addresses may be assigned, but which transmits no external traffic.

LDevID: A "Local Device IDentity" is an X.509 certificate installed during "enrollment". The Domain Certificate used by the ACP is an LDevID. See [AR8021].

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

native interface: Interfaces existing on a node without configuration of the already running node. On physical nodes these are usually physical interfaces. On virtual nodes their equivalent.

node: A system, e.g.: supporting the ACP according to this document. Can be virtual or physical. Physical nodes are called devices.

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 node typically consisting of at least device model/type and serial number, often in a vendor specific format. See sUDI and LDevID.

ULA: A "Unique Local Address" (ULA) is an IPv6 address in the block fc00::/7, defined in [RFC4193]. It is the approximate IPv6 counterpart of the IPv4 private address ([RFC1918]).

(ACP) VRF: The ACP is modelled in this document as a "Virtual Routing and Forwarding" instance (VRF). This means that it is based on a "virtual router" consisting of a separate IPv6 forwarding table to which the ACP virtual interfaces are attached and an associated separate IPv6 routing table. Unlike the VRFs on MPLS/VPN-PE ([RFC4364]) or LISP XTR ([RFC6830]), the ACP VRF does not have any special "core facing" functionality or routing/mapping protocols shared across multiple VRFs. In vendor products a VRF such as the ACP-VRF may also be referred to as a so called VRF-lite.

(ACP) Zone: An ACP zone is a connected region of the ACP where nodes derive from their non-aggregatable ACP address (identifier address) an aggregateable ACP zone address (locator address). See the definition of the ACP Zone Addressing Sub-Scheme (Section 6.10.3). The complete definition of zones is subject to future work because this document does not describe the routing protocols details for aggregation of ACP zone addresses, but only their addressing scheme.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

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 minimize 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 a not configured Network

Today, bootstrapping a new node typically requires all nodes between a controlling node such as an SDN controller ("Software Defined Networking", see [RFC7426]) and the new node to be completely and correctly addressed, configured and secured. Bootstrapping and configuration of a network happens in rings around the controller - configuring each ring of devices before the next one can be bootstrapped. 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 leading up to them.

With the ACP, secure bootstrap of new devices can happen without requiring any configuration such as the transit connectivity to bootstrap further devices. 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 zero-touch and securely 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 misconfigurations that make AAA (Authentication, Authorization and Accounting) servers unreachable 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 applications in a NOC ("Network Operations Center") such as 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 independent of the data-plane (except for the dependency discussed in Section 6.12.2 which can be removed through future work), 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 nodes 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 ("Unique Local Address", see [RFC4193]).

ACP4: The ACP MUST be generic. Usable by all the functions and protocols of the AN infrastructure. Clients of the ACP MUST NOT be tied to a particular application or transport 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 ACP operates 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 across non-ACP nodes, for example to link ACP nodes over the general Internet. This is possible, but introduces a dependency against stable/resilient routing over the non-ACP hops (see Section 8.2).

5. Overview

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

1. An ACP node creates a VRF ("Virtual Routing and Forwarding") 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 link-layer adjacent nodes supporting ACP.

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

4. For each node in the candidate peer list, it then establishes a secure tunnel of the negotiated type. The resulting tunnels are then 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 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 (see Section 6.12.5).

Note:

- Non-autonomic NMS ("Network Management Systems") or SDN controllers have to be explicitly configured for connection into the ACP.

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

- None of the above operations (except explicit configured ones) are reflected in the configuration of the node.

The following figure illustrates the ACP.

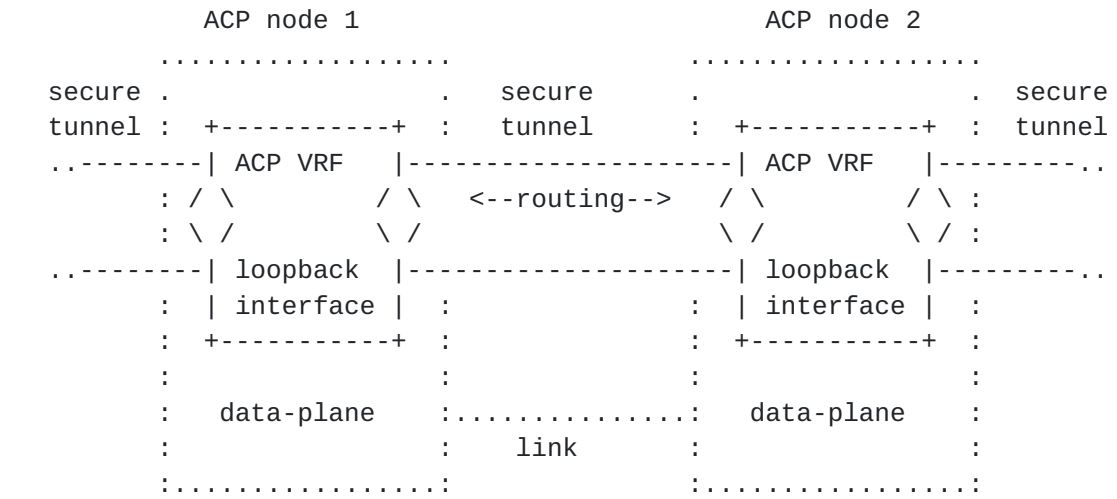


Figure 1

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

An ACP node can be a router, switch, controller, NMS host, or any other IP capable node. Initially, it must have a certificate, as well as an (empty) ACP Adjacency Table (described in Section 6.2). It then can start to discover ACP neighbors and build the ACP. This is described step by step in the following sections:

6.1. ACP Domain, Certificate and Network

ACP relies on group security. An ACP domain is a group of nodes that trust each other to participate in ACP operations. To establish trust, the ACP requires certificates: An ACP node MUST have keying material consisting of a certificate (LDevID), with which it can cryptographically assert its membership in the ACP domain and trust anchor(s) associated with that certificate with which it can verify the membership of other nodes (see Section 6.1.2). The certificate is called the ACP domain certificate, the trust anchor(s) are the CA ("Certificate Authority") of the ACP domain.

The ACP does not mandate specific mechanisms by which this keying material is provisioned into the ACP node, it only requires the Domain information field as specified in Section 6.1.1 in its domain certificate as well as those of candidate ACP peers. See Section 10.1 for more information about enrollment or provisioning options.

Note: LDevID ("Local Device IDentification") is the term used to indicate a certificate that was provisioned by the owner of a node as opposed to IDevID ("Initial Device IDentifier") that may has been loaded on the node during manufacturing time. Those IDevID do not include owner and deployment specific information to allows autonomic establishment of trust for the operations of an ACP domain (e.g.: between two ACP nodes without relying on any third party).

This document uses the term ACP in many places where its reference document use the word autonomic. This is done because those reference document consider fully autonomic network and nodes, but support of ACP does not require support for other components of autonomic networks. Therefore the word autonomic would be irritating to operators interested in only the ACP:

[RFC7575] defines the term "Autonomic Domain" as a collection of autonomic nodes. ACP nodes do not need to be fully autonomic, but when they are, then the ACP domain is an autonomic domain. Likewise, [I-D.ietf-anima-reference-model] defines the term "Domain Certificate" as the certificate used in an autonomic domain. The ACP domain certificate is that domain certificate when ACP nodes are (fully) autonomic nodes. Finally, this document uses the term ACP network to refer to the network created by active ACP nodes in an ACP domain. The ACP network itself can extend beyond ACP nodes through the mechanisms described in Section 8.1).

The ACP domain certificate can and should be used for any authentication between ACP nodes where the required security is domain membership. Section 6.1.2 defines this "ACP domain membership

check". The uses of this check that are standardized in this document are for the establishment of ACP secure channels (Section 6.6) and for ACP GRASP (Section 6.8.2). Other uses are subject to future work, but it is recommended that it is the default security check for any end-to-end connections between ASA. It is equally useable by other functions such as legacy OAM functions.

6.1.1. Certificate Domain Information Field

Information about the domain MUST be encoded in the domain certificate in a subjectAltName / rfc822Name field according to the following ABNF definition ([RFC5234]):

[RFC Editor: Please substitute SELF in all occurences of rfcSELF with the RFC number assigned to this document and remove this comment line]

```
domain-information = local-part "@" domain
local-part = key "." local-info
key = "rfcSELF"
local-info = [ acp-address ] [ "+" rsub extensions ]
acp-address = 32hex-dig
hex-dig = DIGIT / "a" / "b" / "c" / "d" / "e" / "f"
rsub = [ domain-name ] ; empty if not used
domain = domain-name
routing-subdomain = [ rsub " ." ] domain
domain-name = ; <domain> ; as of RFC1034, section 3.5
extensions = *( "+" extension )
extension = ; future definition.
            ; Must fit RFC5322 simple dot-atom format.

Example:
domain-information = rfcSELF+fd89b714f3db000002000000064000000
                     +area51.research@acp.example.com
routing-subdomain = area51.research.acp.example.com
```

"acp-address" MUST be the ACP address of the node. It is optional to support variations of the ACP mechanisms, for example other means for nodes to assign ACP addresses to themselves. Such methods are subject to future work though.

Note: "acp-address" cannot use standard IPv6 address formats because it must match the simple dot-atom format of [RFC5322]. ":" are not allowed in that format.

"domain" is used to indicate the ACP Domain across which all ACP nodes trust each other and are willing to build ACP channel to each other. See Section 6.1.2. Domain SHOULD be the FQDN of a domain

owned by the operator assigning the certificate. This is a simple method to ensure that the domain 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 a string in FQDN format that intends to be equally unique.

"routing-subdomain" is the autonomic subdomain that is used to calculate the hash for the ULA prefix of the ACP address of the node. "rsub" is optional; its syntax is defined in this document, but its semantics are for further study. Understanding the benefits of using rsub may depend on the results of future work on enhancing routing for the ACP. When "rsub" is not used, "routing-subdomain" is the same as "domain".

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

Note that the maximum size of "domain-information" is 254 characters and the maximum size of node-info is 64 characters according to [RFC5280] that is referring to [RFC2821] (superseded by [RFC5321]).

The subjectAltName / 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 incompatibilities 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 domain information field; 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 node 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 "domain information field".

- The format of the rfc822Name is chosen so that an operator can set up a mailbox called rfcSELF@<domain> that would receive emails sent towards the rfc822Name of any node inside a domain. This is possible because in many modern mail systems, components behind a "+" character are considered part of a single mailbox. In other words, it is not necessary to set up a separate mailbox for every ACP node, 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. ACP domain membership check

The following points constitute the ACP domain membership check:

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

- The peers certificate is signed by one of the trust anchors associated with the ACP domain certificate.

- If the node certificates indicate a CDP (or OCSP) then the peer's certificate must be valid according to those criteria. e.g.: OCSP check across the ACP or not listed in the CRL retrieved from the CDP.

- The peers certificate has a syntactically valid domain information field (subjectAltName / rfc822Name) and the domain name in that peers domain information field is the same as in this ACP node certificate. Note that future Intent rules may modify this. See Section 10.7.

6.1.3. Certificate Maintenance

ACP nodes MUST support certificate renewal via EST ("Enrollment over Secure Transport", see [RFC7030]) and MAY support other mechanisms. An ACP network MUST have at least one ACP node supporting EST server functionality across the ACP so that EST renewal is useable. The mechanism by which the domain certificate was initially provisioned SHOULD provide a mechanism to store the URL of one EST server with its ACP address into the node for later renewal. This server does not have to be the same as the one performing the initial certificate enrolment.

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

Example:

```
[M_FLOOD, 12340815, h'fd89b714f3db0000200000064000001', 210000,
    ["SRV.est", 4, 255 ],
    [O_IPv6_LOCATOR,
         h'fd89b714f3db0000200000064000001', TCP, 80]
]
```

The formal CDDL definition is:

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

objective = ["SRV.est", objective-flags, loop-count,
                                     objective-value]

objective-flags = sync-only  ; as in GRASP spec
sync-only =  4               ; M_FLOOD only requires synchronization
loop-count      = 255        ; recommended
objective-value =            ; Not used (yet)
```

The objective value "SRV.est" indicates that the objective is an [RFC7030] compliant EST server because "est" is an [RFC6335] registered service name for [RFC7030]. Future backward compatible extensions/alternatives to [RFC7030] may be indicated through

objective-value. Future non-backward compatible certificate renewal options must use a different objective-name.

The M_FLOOD message MUST be sent periodically. The default SHOULD be 60 seconds, the value SHOULD be operator configurable. It must be so high that the aggregate amount of periodic M_FLOODs from all flooded objectives causes only negligible traffic across the ACP. The ttl parameter SHOULD be 3.5 times the period so that up to three consecutive messages can be dropped before considering an announcement expired. In the example above, the ttl is 210000 msec, 3.5 times 60 seconds.

Domain certificates SHOULD by default be renewed 50% into their lifetime. When performing renewal, the node SHOULD attempt to connect to the remembered EST server. If that fails, it SHOULD attempt to connect to EST server(s) learned via GRASP. The server with which certificate renewal succeeds SHOULD be remembered for the next renewal.

Remembering the last renewal server and preferring it provides stickiness which can help diagnostics. It also provides some protection against off-path compromised ACP members announcing bogus information into GRASP.

The ACP node MUST support CRLs ("Certificate Revocation Lists") via HTTPs from one or more CDPs ("CRL Distribution Points"). These CDPs MUST be indicated in the Domain Certificate when used. If the CDP URL uses an IPv6 ULA, the ACP node 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 node during the HTTPs TLS handshake SHOULD 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 node 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 limiting 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 performance to manage short lived certificates.

See Section 10.1 for further optimizations of certificate maintenance when BRSKI can be used ("Bootstrapping Remote Secure Key Infrastructures", see [I-D.ietf-anima-bootstrapping-keyinfra]).

6.2. ACP Adjacency Table

To know to which nodes to establish an ACP channel, every ACP node maintains an adjacency table. The adjacency table contains information about adjacent ACP nodes, at a minimum: Node-ID, interface on which neighbor was discovered (by GRASP as explained below), link-local IPv6 address of neighbor on that interface, certificate (including domain information field). An ACP node 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 ACP node 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 ACP node's certificate. However, subsequent steps MUST always start with authenticating the peer.

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

Interaction between ACP and other autonomic elements like GRASP (see below) or ASAs should be via an API that allows (appropriately access controlled) read/write access to the ACP Adjacency Table. Specification of such an API is subject to future work.

6.3. Neighbor Discovery with DULL GRASP

The ACP uses one instance of DULL GRASP (See section 3.5.2.2 of [I-D.ietf-anima-grasp] for its formal definition) for every physical L2 subnet of the ACP node to discover physically adjacent candidate ACP neighbors. Native interfaces (e.g.: physical interfaces on physical nodes) SHOULD be brought up automatically enough so that ACP discovery can be performed and any native 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 not configured 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 (e.g.: no other link-local IPv6 transport stack responders for example).

Note that the use of the IPv6 link-local multicast address (ALL_GRASP_NEIGHBORS) implies the need to use MLD ([RFC3810]) to announce the desire to receive packets for that address. Otherwise DULL GRASP could fail to operate correctly in the presence of MLD snooping, non-ACP enabled L2 switches - because those would stop forwarding DULL GRASP packets. Switches not supporting MLD snooping simply need to operate as pure L2 bridges for IPv6 multicast packets for DULL GRASP to work.

ACP discovery SHOULD NOT be enabled by default on non-native interfaces. In particular, ACP discovery MUST NOT run inside the ACP across ACP virtual interfaces. See Section 10.3 for further, non-normative suggestions how to enable/disable ACP at node and interface level. See Section 8.2.2 for more details about tunnels (typical non-native interfaces). See Section 7 for how ACP should be extended on devices operating (also) as L2 bridges.

Note: If an ACP node 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 node does not have a domain certificate. Discovery of ACP neighbors happens only when the node does have the certificate. The node therefore never needs to discover both a bootstrap proxy and ACP neighbor at the same time.

An ACP 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, 180000,
    ["AN_ACP", 4, 1, "IKEv2" ],
    [O_IPv6_LOCATOR,
         h'fe80000000000000c0011001FEEF0000, UDP, 15000]
    ["AN_ACP", 4, 1, "dTLS" ],
    [O_IPv6_LOCATOR,
         h'fe80000000000000c0011001FEEF0000, UDP, 17000]
]
```

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 = sync-only ; as in the GRASP specification
sync-only =  4    ; M_FLOOD only requires synchronization
loop-count = 1    ; limit to link-local operation
objective-value = method
method = "IKEv2" / "dTLS"  ; or future methods
```

The objective-flags field is set to indicate synchronization.

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

In the above (recommended) example the period of sending of the objective could be 60 seconds the indicated ttl of 180000 msec means that the objective would be cached by ACP nodes even when two out of three messages are dropped in transit.

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 ACP node on the sending interface.

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

The locator-option 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.

"IKEv2" is the abbreviation for "Internet Key Exchange protocol version 2", as defined in [RFC7296]. It is the main protocol used by the Internet IP security architecture ("IPsec", see [RFC4301]). We therefore use the term "IKEv2" and not "IPsec" in the GRASP definitions and example above. "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 method needs to flood multiple versions of the "AN_ACP" objective so that each method can be accompanied by its own locator-option. This can use a single GRASP M_FLOOD message as shown in above example.

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

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

The ACP is by default established exclusively between nodes in the same domain. This includes all routing subdomains. Section 10.7 explains how ACP connections across multiple routing subdomains are special.

Future extensions to this document including Intent can change this default behavior. 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 behavior. See Section 10.7 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 cannot include any non-protected negotiation. To avoid re-inventing and validating security association mechanisms, the next step after discovering 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 ACP nodes, it is unclear whether it is feasible to even decide on a single MTI (mandatory to implement) security association protocol across all ACP nodes:

From the use-cases it seems clear that not all type of ACP nodes 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 ("datagram Transport Layer Security version 1.2", see [RFC6347]) because that code exists already on them for end-to-end security, but low-end in-ceiling L2 switches may only want to support MacSec because that is also supported in their chips. Only a 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, ACP nodes 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 ACP node may choose to attempt initiate the different feasible ACP secure channel protocols 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 other's certificates because it must be used by all secure channel protocols for mutual authentication. The node with the lower Node-ID in the ACP address becomes Bob, the one with the higher Node-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 prefers and the security association succeeded. The roles of Bob and Alice are then assigned. At this stage, the protocol may not even have completed negotiating a common security profile. The protocol could for example could have been IPsec via IKEv2 ("IP security", see [RFC4301] and "Internet Key Exchange protocol version 2", see [RFC7296]. It is now up to Alice to decide how to proceed. Even if the IPsec connecting determined a working profile with Bob, Alice might prefer some other secure protocol (e.g.: 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 node must not assume that neighbors with the same L2 or link-local IPv6 addresses on different L2 interfaces are the same node. This can only be determined after examining the certificate after a successful security association attempt.

6.6. Candidate ACP Neighbor verification

Independent of the security association protocol chosen, candidate ACP neighbors need to be authenticated based on their domain certificate. This implies that any secure channel protocol MUST support certificate based authentication that can support the ACP domain membership check as defined in Section 6.1.2. If it fails, 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 ACP node announces its ability to support IKEv2 as the ACP secure channel protocol in GRASP as "IKEv2".

6.7.1.1. Native IPsec

To run ACP via IPsec natively, no further IANA assignments/ definitions are required. An ACP node supporting native IPsec MUST use IPsec security setup via IKEv2, tunnel mode, local and peer link-local IPv6 addresses used for encapsulation. It MUST support ESP with AES256 for encryption and SHA256 hash and MUST NOT permit weaker crypto options.

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

IPsec tunnel mode is required because the ACP will route/forward packets received from any other ACP node across the ACP secure channels, and not only its own generated ACP packets. With IPsec transport mode, it would only be possible to send packets originated by the ACP node itself.

ESP is used because ACP mandates the use of encryption for ACP secure channels.

6.7.1.2. IPsec with GRE encapsulation

In network devices it is often more common to implement high performance virtual interfaces on top of GRE encapsulation than on top of a "native" IPsec association (without any other encapsulation than those defined by IPsec). On those devices it may be beneficial to run the ACP secure channel on top of GRE protected by the IPsec association.

To run ACP via GRE/IPsec, no further IANA assignments/definitions are required. The ACP node MUST support IPsec security setup via IKEv2, IPsec transport mode, local and peer link-local IPv6 addresses used for encapsuation, ESP with AES256 encryption and SHA256 hash.

When GRE is used, transport mode is sufficient because the routed ACP packets are not "tunneled" by IPsec but rather by GRE: IPsec only has to deal with the GRE/IP packet which always uses the local and peer

link-local IPv6 addresses and is therefore applicable to transport mode.

ESP is used because ACP mandates the use of encryption for ACP secure channels.

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 the offer for native IPsec as described above (because that option is mandatory to support).

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 ACP nodes supporting dTLS as a secure channel protocol MUST support AES256 encryption and MUST 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 ACP node MUST support IPsec natively and MAY support IPsec via GRE. A constrained ACP node MUST support dTLS. ACP nodes connecting constrained areas with baseline areas MUST therefore support IPsec and dTLS.

ACP nodes need to specify in documentation the set of secure ACP mechanisms they support.

An ACP secure channel MUST immediately be terminated when the lifetime of any certificate in the chain used to authenticate the neighbor expires or becomes revoked. Note that this 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 was added as an afterthought and relied upon inconsistently.

ACP provides IP unicast routing via the RPL routing protocol (described below).

The ACP does not use IP multicast routing nor does it provide generic IP multicast services (the handling of GRASP link-local multicast messages is explained in the following section). Instead, the ACP provides service discovery via the objective discovery/announcement and negotiation mechanisms of the ACP GRASP instance (services are a form of objectives). These mechanisms use hop-by-hop reliable flooding of GRASP messages for both service discovery (GRASP M_DISCOVERY messages) and service announcement (GRASP M_FLOOD messages).

IP multicast is not used by the ACP because the ANI (Autonomic Networking Infrastructure) itself does not require IP multicast but only service announcement/discovery. Using IP multicast for that would have made it necessary to develop a zero-touch autoconfiguring solution for ASM (Any Source Multicast - original form of IP multicast defined in [RFC1112]), which would be quite complex and difficult to justify. One aspect of complexity that has never been attempted to be solved in IETF documents is the automatic-selection of routers that should be PIM-SM rendezvous points (RPs) (see [RFC7761]). The other aspects of complexity are the implementation of MLD ([RFC4604]), PIM-SM and Anycast-RP (see [RFC4610]). If those implementations already exist in a product, then they would be very likely tied to accelerated forwarding which consumes hardware resources, and that in return is difficult to justify as a cost of performing only service discovery.

Future ASA may need high performance in-network data replication. That is the case when the use of IP multicast is justified. These ASA can then use service discovery from ACP GRASP, and then they do not need ASM but only SSM (Source Specific Multicast, see [RFC4607]) for the IP multicast replication. SSM itself can simply be enabled

in the data-plane (or even in an update to the ACP) without any other configuration than just enabling it on all nodes and only requires a simpler version of MLD (see [RFC5790]).

LSP (Link State Protocol) based IGP routing protocols typically have a mechanism to flood information, and such a mechanism could be used to flood GRASP objectives by defining them to be information of that IGP. This would be a possible optimization in future variations of the ACP that do use an LSP routing protocol. Note though that such a mechanism would not work easily for GRASP M_DISCOVERY messages which are constrained flooded up to a node where a responder is found. We do expect that many future services in ASA will have only few consuming ASA, and for those cases, M_DISCOVERY is the more efficient method than flooding across the whole domain.

Because the ACP uses RPL, one desirable future extension is to use RPLs existing notion of loop-free distribution trees (DODAG) to make GRASPs flooding more efficient both for M_FLOOD and M_DISCOVERY) See Section 6.12.5 how this will be specifically beneficial when using NBMA interfaces. This is not currently specified in this document because it is not quite clear yet what exactly the implications are to make GRASP flooding depend on RPL DODAG convergence and how difficult it would be to let GRASP flooding access the DODAG information.

6.8.2. ACP as the Security and Transport substrate for GRASP

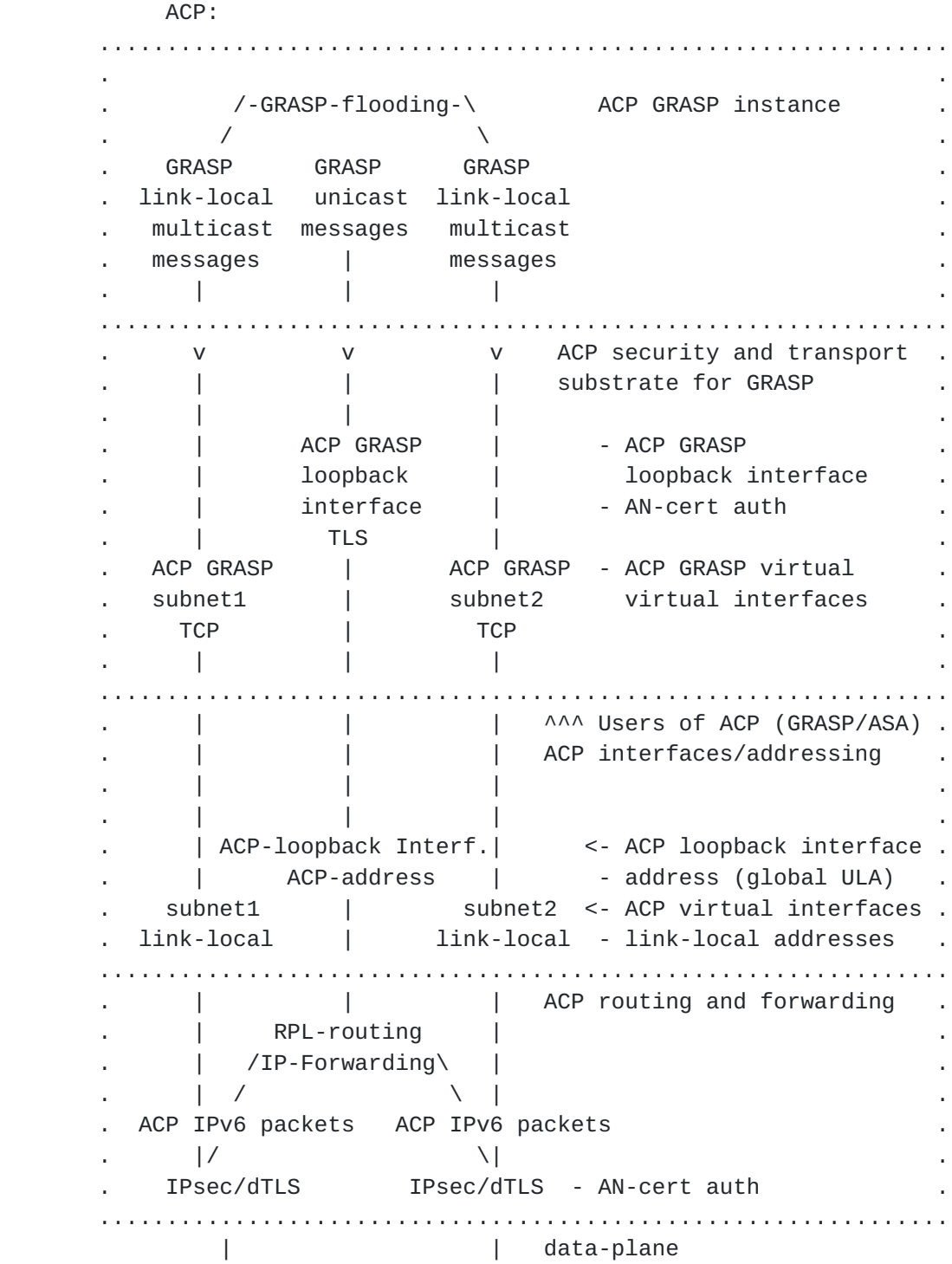
In the terminology of GRASP ([I-D.ietf-anima-grasp]), the ACP is the security and transport substrate for the GRASP instance run inside the ACP ("ACP GRASP").

This means that the ACP is responsible to ensure that this instance of GRASP is only sending messages across the ACP GRASP 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 node has a domain certificate, and continues to exist even if all of its neighbors cease operation.

In this case ASAs using GRASP running on the same node would still need to be able to discover each other's 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 operate.

The way ACP acts as the security and transport substrate for GRASP is visualized in the following picture:

[RFC Editor: please try to put the following picture on a single page and remove this note. We cannot figure out how to do this with XML. The picture does fit on a single page.]



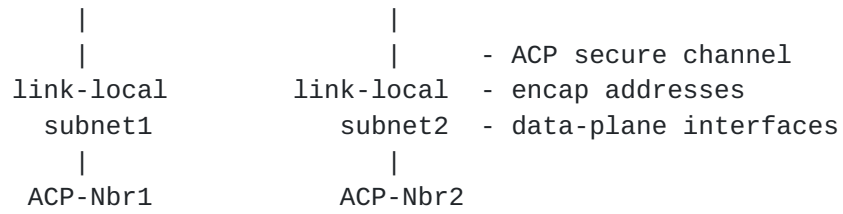


Figure 2

GRASP unicast messages inside the ACP always use the ACP address. Link-local ACP addresses must not be used inside objectives. GRASP unicast messages inside the ACP are transported via TLS 1.2 ([RFC5246]) connections with AES256 encryption and SHA256. Mutual authentication uses the ACP domain membership check defined in (Section 6.1.2).

GRASP link-local multicast messages are targeted for a specific ACP virtual interface (as defined Section 6.12.5) but are sent by the ACP into an ACP GRASP virtual interface that is constructed from the TCP connection(s) to the IPv6 link-local neighbor address(es) on the underlying ACP virtual interface. If the ACP GRASP virtual interface has two or more neighbors, the GRASP link-local multicast messages are replicated to all neighbor TCP connections.

TLS and TLS connections for GRASP in the ACP use the IANA assigned TCP port for GRASP (7107). Effectively the transport stack is expected to be TLS for connections from/to the ACP address (e.g.: global scope address(es)) and TCP for connections from/to link-local addresses on the ACP virtual interfaces. The latter ones are only used for flooding of GRASP messages.

6.8.2.1. Discussion

TCP encapsulation for GRASP M_DISCOVERY and M_FLOOD link local messages is used because these messages are flooded across potentially many hops to all ACP nodes and a single link with even temporary packet loss issues (e.g.: WiFi/Powerline link) can reduce the probability for loss free transmission so much that applications would want to increase the frequency with which they send these messages. This would result in more traffic flooding than hop-by-hop reliable retransmission as provided for by TCP.

TLS is mandated for GRASP non-link-local unicast 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 (e.g.: via domain certificate revocation / expiry).

If GRASP peer connections would just use TCP, compromised ACP members could simply eavesdrop passively on GRASP peer connections for whom they are on-path ("Man In The Middle" - MITM). Or intercept and modify them. With TLS, it is not possible to completely eliminate problems with compromised ACP members, but attacks are a lot more complex:

Eavesdropping/spoofing by a compromised ACP node is still possible because in the model of the ACP and GRASP, the provider and consumer of an objective have initially no unique information (such as an identity) about the other side which would allow them to distinguish a benevolent from a compromised peer. The compromised ACP node would simply announce the objective as well, potentially filter the original objective in GRASP when it is a 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, but in an AN environment this is quite likely public knowledge or evens standardized.

The GRASP TLS connections are run like any other ACP traffic through the ACP secure channels. This leads to double authentication/ encryption. Future work optimizations could avoid this but it is unclear how beneficial/feasible this is:

- The security considerations for GRASP change against attacks from non-ACP (e.g.: "outside") nodes: TLS is subject to reset attacks while secure channel protocols may be not (e.g.: IPsec is not).

- The secure channel method may leverage hardware acceleration and there may be little or no gain in eliminating it.

- The GRASP TLS connections need to implement any additional security options that are required for secure channels. For example the closing of connections when the peers certificate has expired.

6.9. Context Separation

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

In classical network systems, a dedicated so called "Virtual routing and forwarding instance" (VRF) is one logical implementation option for the ACP. If possible by the systems software 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 node. 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 ACP network wide valid addresses and routing. Each ACP node must create a loopback interface with an ACP network wide unique address inside the ACP context (as explained in in Section 6.9). This address may be used also in other virtual contexts.

With the algorithm introduced here, all ACP nodes in the same routing subdomain have the same /48 ULA global ID prefix. Conversely, ULA 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 nodes ACP 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 ACP loopback interfaces (and potentially those configured for "ACP connect", see Section 8.1) carry routable address(es); all other interfaces (called ACP virtual interfaces) only use IPv6 link local addresses. The usage of IPv6 link local addressing is discussed in [RFC7404].

- Use-ULA: For loopback interfaces of ACP nodes, we use Unique Local Addresses (ULA), specifically ULA-Random, as defined in [[RFC4193] with L=1].

- 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].

- Addresses in the ACP are not considered sensitive on privacy grounds because ACP nodes are not expected to be end-user devices. Therefore, ACP addresses do not need to be pseudo-random as discussed in [RFC7721]. Because they are not propagated to untrusted (non ACP) nodes and stay within a domain (of trust), we also consider them not to be subject to scanning attacks.

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 ACP nodes has the following format:

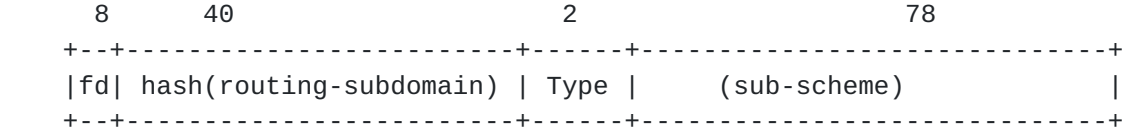


Figure 3: ACP Addressing Base Scheme

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 40 bits ULA "global ID" (term from [RFC4193]) for ACP addresses carried in the domain information field of domain certificates are the first 40 bits of the SHA256 hash of the routing subdomain from the same domain information field. In the example of Section 6.1.1, the routing subdomain is "area51.research.acp.example.com" and the 40 bits ULA "global ID" 89b714f3db.
- To allow for extensibility, the fact that the ULA "global ID" is a hash of the routing subdomain SHOULD NOT be assumed by any ACP node 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 domain information field in response to the CSR Attribute Request by the pledge.
- Type: This field allows different address sub-schemes. This addresses the "upgradability" requirement. Assignment of types for this field will be maintained by IANA.

The sub-scheme may imply a range or set of addresses assigned to the node, this is called the ACP address range/set and explained in each sub-scheme.

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.

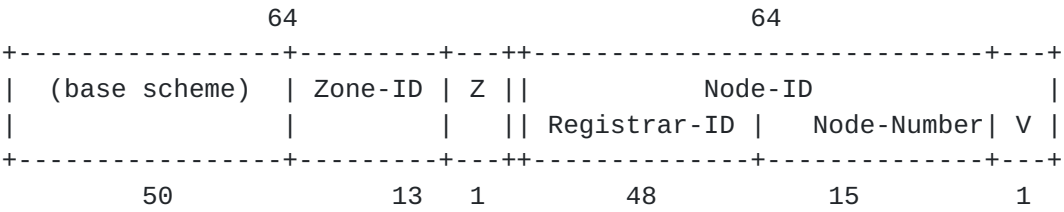


Figure 4: ACP Zone Addressing Sub-Scheme

The fields are defined as follows:

- Zone-ID: If set to all zero bits: The Node-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 6.10.3.1 on how this field is used in detail.

- Z: MUST be 0.

- Node-ID: A unique value for each node.

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

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

- Node-Number: A number which is unique for a given registrar, to identify the node. This can be a sequentially assigned number.

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

In the ACP Zone Addressing Sub-Scheme, the ACP address in the certificate has Zone-ID and V fields as all zero bits. The ACP address set includes addresses with any Zone-ID value and any V value.

The "Node-ID" itself is unique in a domain (i.e., the Zone-ID is not required for uniqueness). Therefore, a node can be addressed either as part of a flat hierarchy (Zone-ID = 0), or with an aggregation scheme (any other Zone-ID). An address with Zone-ID = 0 is an identifier, with a Zone-ID !=0 it is a locator. See Section 6.10.3.1 for more details.

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 (autonomic 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 p2p virtual 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 ACP node. For example, in a network with 20,000 ACP nodes, this avoid 20,000 additional routes in the routing table.

6.10.3.1. Usage of the Zone-ID Field

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

Zone-ID = 0 is the default addressing scheme in an ACP domain. Every ACP node with a Zone Addressing Sub-Scheme address MUST respond to its ACP address with Zone-ID = 0. Used on its own this leads to a non-hierarchical address scheme, which is suitable for networks up to a certain size. Zone-ID = 0 addresses act as identifiers for the nodes, and aggregation of these address in the ACP routing table is not possible.

If aggregation is required, the 13 bit Zone-ID value allows for up to 8191 zones. The allocation of Zone-ID's may either happen automatically through a to-be-defined algorithm; or it could be configured and maintained explicitly.

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

A node knowing it is in a zone MUST also use that Zone-ID != 0 address in GRASP locator fields. This eliminates the use of the identifier address (Zone-ID = 0) in forwarding and the need for network wide reachability of those non-aggregateable identifier addresses. Zone-ID != 0 addresses are assumed to be aggregateable in routing/forwarding based on how they are allocated in the ACP topology (subject to future work).

Note: 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 ACP domain. This gives future work two options to consider.

Note: Zones and Zone-ID as defined here are not related to [RFC4007] zones or zone_id. ACP zone addresses are not scoped (reachable only from within an RFC4007 zone) but reachable across the whole ACP. An RFC4007 zone_id is a zone index that has only local significance on a

node, whereas an ACP Zone-ID is an identifier for an ACP zone that is unique across that ACP.

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.

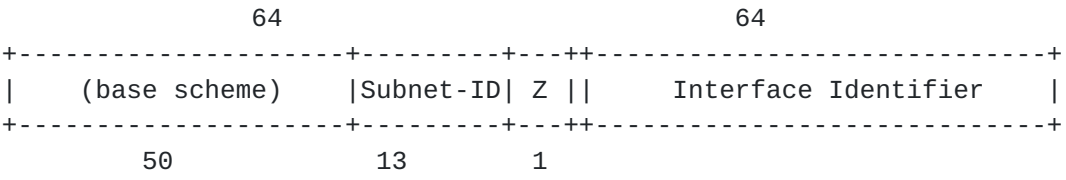


Figure 5: ACP Manual Addressing Sub-Scheme

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 cannot 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 nodes 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 aggregatable routing approach across them. Z=0 (Zone-ID sub-scheme) would then be used for network wide unique, registrar assigned (and certificate protected) Node-IDs primarily for ACP nodes while Z=1 would be used for node-level assigned Interface Identifiers primarily for non-ACP-nodes (on logical subnets where the ACP node is a router).

Manual addressing sub-scheme addresses SHOULD only be used in domain certificates assigned to nodes that cannot fully participate in the automatic establishment of ACP secure channels or ACP routing. The intended use are nodes connecting to the ACP via an ACP edge node and ACP connect (see Section 8.1) - such as legacy NOC equipment. They would not use their domain certificate for ACP secure channel creation and therefore do not need to participate in ACP routing either. They would use the certificate for authentication of any transport services. The value of the Interface Identifier is left for future definitions.

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.

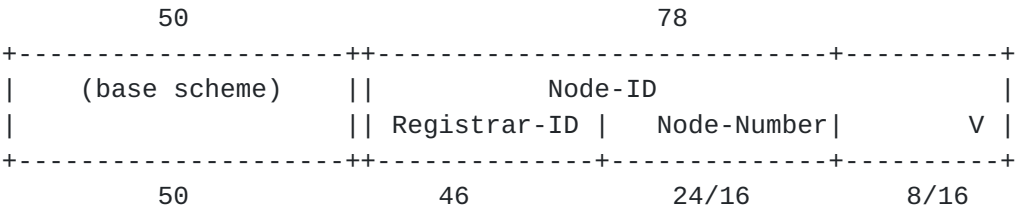


Figure 6: ACP Vlong Addressing Sub-Scheme

This addressing scheme foregoes the Zone-ID field to allow for larger, flatter routed networks (e.g.: as in IoT) with 8421376 Node-Numbers (2^23+2^15). It also allows for up to 2^16 - 65536 different virtualized addresses within a node, which could be used to address individual software components in an ACP node.

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

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

- Registrar-ID: To maximize Node-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 cannot be used to distinguish MAC addresses).

- If the first bit of the "Node-Number" is "1", then the Node-Number is 16 bit long and the V field is 16 bit long. Otherwise the Node-Number is 24 bit long and the V field is 8 bit long. "0" bit Node-Numbers are intended to be used for "general purpose" ACP

nodes that would potentially have a limited number (< 256) of clients (ASA/Autonomic Functions or legacy services) of the ACP that require separate V(irtual) addresses. "1" bit Node-Numbers are intended for ACP nodes that are ACP edge nodes (see Section 8.1.1) or that have a large number of clients requiring separate V(irtual) addresses. For example large SDN controllers with container modular software architecture (see Section 8.1.2).

In the Vlong addressing sub-scheme, the ACP address in the certificate has all V field bits as zero. The ACP address set for the node includes any V value.

6.10.6. Other ACP Addressing Sub-Schemes

Before further addressing sub-schemes are defined, experience with the schemes defined here should be collected. The schemes defined in this document have been devised to allow hopefully sufficiently flexible setup of ACPs for a variety of situation. These reasons also lead to the fairly liberal use of address space: The Zone Addressing Sub-Scheme is intended to enable optimized routing in large networks by reserving bits for Zone-ID's. The Vlong addressing sub-scheme enables the allocation of 8/16 bit of addresses inside individual ACP nodes. Both address spaces allow distributed, uncoordinated allocation of node addresses by reserving bits for the Registrar-ID field in the address.

IANA is asked need to assign a new "type" for each new addressing sub-scheme. With the current allocations, only 2 more schemes are possible, so the last addressing scheme should consider to be extensible in itself (e.g.: by reserving bits from it for further extensions.

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 explicit 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 10.5 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 10.7 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 data-plane 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") according to the data-plane 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 data-plane 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 accommodate only a single class of routing table and cannot 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 data-plane 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 (e.g.: 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 selected 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 node (RPL node) announces an IPv6 prefix covering the address(es) used in the ACP node. The prefix length depends on the chosen addressing sub-scheme of the ACP address provisioned into the certificate of the ACP node, e.g.: /127 for Zone Addressing Sub-Scheme or /112 or /120 for Vlong addressing sub-scheme. See Section 6.10 for more details.

Every ACP node MUST install a black hole (aka null) 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 Data-Plane 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 node. 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. Performance

There are no performance requirements against ACP implementations defined in this document because the performance requirements depend on the intended use case. It is expected that full autonomic node with a wide range of ASA can require high forwarding plane

performance in the ACP, for example for telemetry, but that determination is for future work. Implementations of ACP to solely support traditional/SDN style use cases can benefit from ACP at lower performance, especially if the ACP is used only for critical operations, e.g.: when the data-plane is not available. See [I-D.ietf-anima-stable-connectivity] for more details.

6.12.2. 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 specifies extensions in which secure channels are tunnels. For those, 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 (e.g.: IPsec / dTLS) across IPv6 link-local addresses, something that may be an uncommon profile. Functionally, 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 (e.g.: tunneling across L3 or use of L2 only encapsulations).

6.12.3. 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.

6.12.4. 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 node, 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 (e.g.: on a different link or with a different address announced in GRASP).

6.12.5. 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" was introduced initially to refer to an internal interface on a node that would allow IP traffic between transport endpoints on the node in the absence or failure of any or all external interfaces, see [RFC4291] section 2.5.3.

Even though loopback interfaces where originally designed to hold only loopback addresses not reachable from outside the node, these interfaces are also commonly used today to hold addresses reachable from the outside. They are meant to be reachable independent of any external interface being operational, and therefore to be more resilient. These addresses on loopback interfaces can be thought of as "node addresses" instead of "interface addresses", and that is what ACP address(es) are. This construct makes it therefore possible to address ACP nodes with a well-defined set of addresses independent of the number of external interfaces.

For these reason, the ACP (ULA) address(es) are assigned to loopback interface(s).

ACP secure channels, e.g.: IPsec, dTLS or other future security associations with neighboring ACP nodes can be mapped to ACP virtual interfaces in different ways:

ACP point-to-point virtual interface:

Each 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.

ACP multi-access virtual interface:

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 underlying (physical) subnet. IPv6 link-local multicast packets sent into an ACP multi-access virtual interface are replicated to every ACP secure channel mapped into the ACP multicast-access virtual interface. IPv6 unicast packets sent into an ACP multi-access virtual interface are sent to the ACP secure channel that belongs to the ACP neighbor that is the next-hop in the ACP forwarding table entry used to reach the packets destination address.

There is no requirement for all ACP nodes on the same multi-access subnet to use the same type of ACP virtual interface. This is purely a node local decision.

ACP nodes MUST perform standard IPv6 operations across ACP virtual interfaces including SLAAC (Stateless Address Auto-Configuration - [RFC4862]) to assign their IPv6 link local address on the ACP virtual interface and ND (Neighbor Discovery - [RFC4861]) to discover which IPv6 link-local neighbor address belongs to which ACP secure channel mapped to the ACP virtual interface. This is independent of whether the ACP virtual interface is point-to-point or multi-access.

ACP nodes MAY reduce the amount of link-local IPv6 multicast packets from ND by learning the IPv6 link-local neighbor address to ACP secure channel mapping from other messages such as the source address of IPv6 link-local multicast RPL messages - and therefore forego the need to send Neighbor Solicitation messages.

ACP nodes MUST NOT derive their ACP virtual interface IPv6 link local address from their IPv6 link-local address used on the underlying interface (e.g.: the address that is used as the encapsulation address in the ACP secure channel protocols defined in this document). This ensures that the ACP virtual interface operations will not depend on the specifics of the encapsulation used by the ACP secure channel and that attacks against SLAAC on the physical interface will not introduce new attack vectors against the operations of the ACP virtual interface.

The link-layer address of an ACP virtual interface is the address used for the underlying interface across which the secure tunnels are

built, typically Ethernet addresses. Because unicast IPv6 packets sent to an ACP virtual interface are not sent to a link-layer destination address but rather an ACP secure channel, the link-layer address fields SHOULD be ignored on reception and instead the ACP secure channel from which the message was received should be remembered.

Multi-access ACP virtual interfaces are preferable implementations when the underlying interface is a (broadcast) multi-access subnet because they do reflect the presence of the underlying multi-access subnet 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 have been built running natively on the multi-access interfaces.

Consider also the impact of point-to-point vs. multi-access virtual interface on the efficiency of flooding via link local multicasted messages:

Assume a LAN with three ACP neighbors, Alice, Bob and Carol. Alice's ACP GRASP wants to send a link-local GRASP multicast message to Bob and Carol. If Alice's ACP emulates the LAN as one point-to-point virtual interface to Bob and one to Carol, The sending applications itself will send two copies, if Alice's 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 ACP point-to-point virtual interfaces, their GRASP instance would forward the packet from Alice to each other as part of the GRASP flooding procedure. These packets are unnecessary and would be discarded by GRASP on receipt as duplicates (by use of the GRASP Session ID). If Bob and Charlies ACP would emulate a multi-access virtual interface, then this would not happen, because GRASPs flooding procedure does not replicate back packets to the interface that they were received from.

Note that link-local GRASP multicast messages are not sent directly as IPv6 link-local multicast UDP messages into ACP virtual interfaces, but instead into ACP GRASP virtual interfaces, that are layered on top of ACP virtual interfaces to add TCP reliability to link-local multicast GRASP messages. Nevertheless, these ACP GRASP virtual interfaces perform the same replication of message and, therefore, result in the same impact on flooding. See Section 6.8.2 for more details.

RPL does support operations and correct routing table construction across non-broadcast multi-access (NBMA) subnets. This is common when using many radio technologies. When such NBMA subnets are used, they MUST NOT be represented as ACP multi-access virtual interfaces

because the replication of IPv6 link-local multicast messages will not reach all NBMA subnet neighbors. In result, GRASP message flooding would fail. Instead, each ACP secure channel across such an interface MUST be represented as a ACP point-to-point virtual interface. These requirements can be avoided by coupling the ACP flooding mechanism for GRASP messages directly to RPL (flood GRASP across DODAG), but such an enhancement is subject for future work.

Care must also be taken when creating multi-access ACP virtual interfaces across ACP secure channels between ACP nodes 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 ACP point-to-point virtual interfaces for these secure channels.

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

7.1. Why

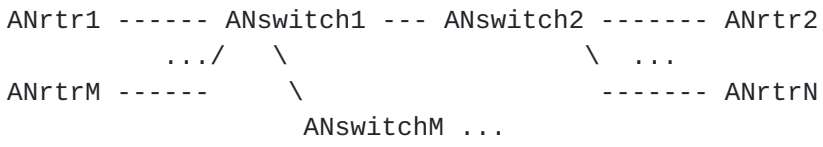


Figure 7

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 ACP router will see all other ACP 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 neighbors. An ACP router with just 4 physical 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, ACP capable L2 switches can ensure that discovery messages terminate on them so that neighboring ACP routers and switches will only find the physically connected ACP 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 ACP capable, and ANswitch2 is not ACP capable. The desired ACP topology is that ANrtr1 and ANrtrM only have an ACP connection 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 switched ports of an L3 device, it is necessary 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 link-local IPv6 multicast address (ALL_GRASP_NEIGHBORS), it is sufficient that all 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.

If the device with L2 ports is supporting per L2 port ACP DULL GRASP as well as MLD snooping ([RFC4541]), then MLD snooping must be changed to never forward packets for ALL_GRASP_NEIGHBORS because that would cause the problem that per L2 port ACP DULL GRASP is meant to overcome (forwarding DULL GRASP packets across L2 ports).

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 6.12.5 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 therefore 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 perform 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", e.g.: 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 re-convergence 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 the ACP would equally be interrupted and re-converge with STP changes.

8. Support for Non-ACP Components (Normative)

8.1. ACP Connect

8.1.1. Non-ACP 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 (or other type of nodes) 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-ACP NMS system does not have access to the ACP by default, just like any other external node.

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-ACP nodes, an ACP node must support "ACP connect" (sometimes also connect "autonomic connect"):

"ACP connect" is a function on an autonomic node that is called an "ACP edge node". With "ACP connect", interfaces on the node 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).

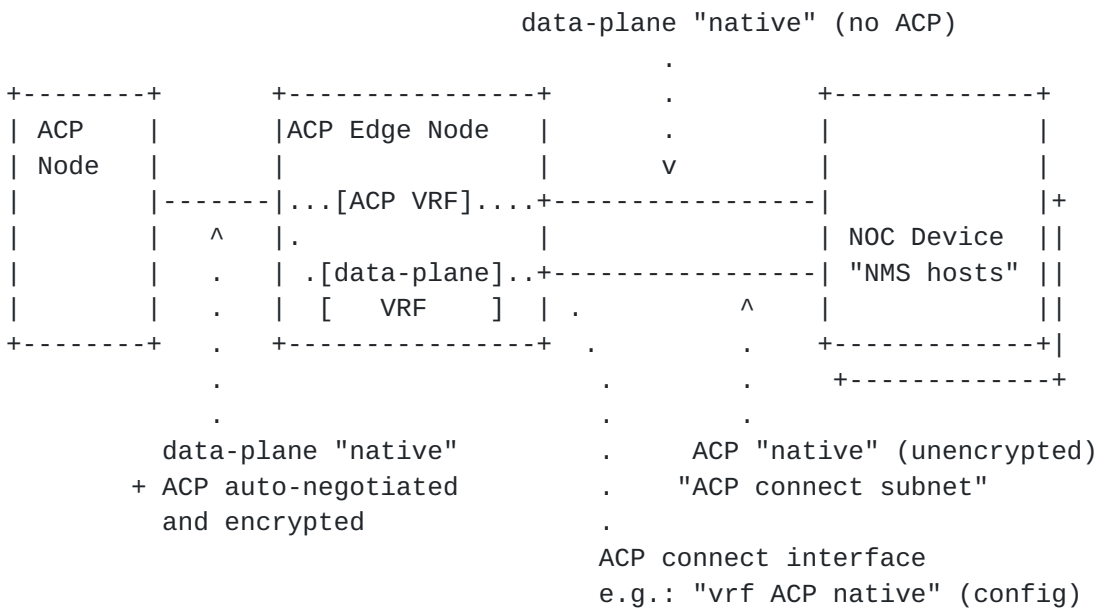


Figure 8: ACP connect

ACP connect has security consequences: All systems and processes connected via ACP connect have access to all ACP 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 node 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 uses only 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 Nodes 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, nodes supporting it SHOULD implement a security mechanism to allow configuration/use of ACP connect interfaces only on nodes explicitly targeted to be deployed with it (such as those physically secure locations like a NOC). For example, the certificate of such node could include an extension required to permit configuration of ACP connect interfaces. This prohibits that a random ACP node 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 node 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 node 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 ACP enabled 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 nodes, NMS hosts and software components that as described 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 node 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 node 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 node 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 node.

Aggregated prefix means that the ACP edge node needs to only announce the /48 ULA prefixes used in the ACP but none of the actual /64 (Manual Addressing Sub-Scheme), /127 (ACP Zone Addressing Sub-Scheme), /112 or /120 (Vlong Addressing Sub-Scheme) routes of actual ACP nodes. If ACP interfaces are configured with non ULA prefixes,

then those prefixes cannot be aggregated without further configured policy on the ACP edge node. 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 signaled via RFC4191 to NMS hosts and software components.

The ACP edge nodes 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.

8.1.4. Combined ACP/Data-Plane Interface (VRF Select)

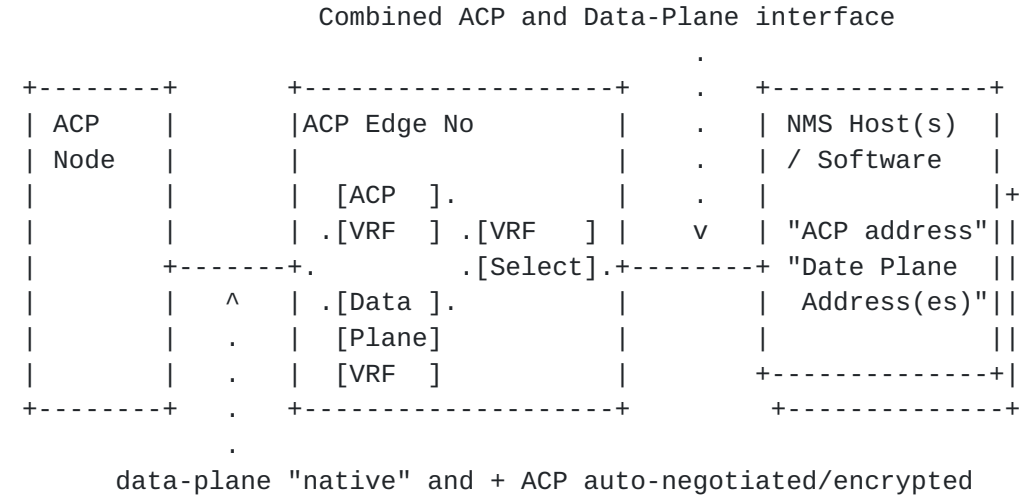


Figure 9: VRF select

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 node needs to de-multiplex 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 node 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 node 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 Node 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 preferable method of attachment. Hosts implementing [RFC8028] should (instead of may) implement [RFC6724] Rule 5.5, so it is preferred for hosts to support [RFC8028].

ACP edge nodes 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 (e.g.: ASA or legacy NMS

applications) to the ACP. Given how the ACP is the security and transport substrate for GRASP, the trustworthiness of nodes/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 nodes. ACP edge nodes SHOULD have options to filter GRASP messages in and out of ACP connect interfaces (permit/deny) and MAY have more fine-grained filtering (e.g.: 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 nodes. 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 assumed that all GRASP packets on a Combined ACP and data-plane interface belong to the GRASP ACP Domain. They must 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-ACP L3 Clouds (Remote ACP neighbors)

Not all nodes 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 neighbors 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 ACP node, remote ACP neighbors are configured as follows. Future work could transform this into a YANG ([RFC7950]) data model.

```
connection = [ method , local-addr, remote-addr, ?pmtu ]
method   = [ "IKEv2" , ?port ]
method //= [ "dTLS",    port ]
local-addr  = [ address , ?vrf  ]
remote-addr = [ address ]
address = ("any" | ipv4-address | ipv6-address )
vrf = tstr ; Name of a VRF on this node with local-address
```

Explicit configuration of a remote-peer provides all the information to build a secure channel without requiring a tunnel to that peer and running DULL GRASP inside of it.

The configuration includes the parameters otherwise signaled via DULL GRASP: local address, remote (peer) locator and method. The differences over DULL GRASP local neighbor discovery and secure channel creation are as follows:

- The local and remote address can be IPv4 or IPv6 and are typically global scope addresses.

- The vrf across which the connection is built (and in which local-addr exists) can to be specified. If vrf is not specified, it is the default vrf on the node. In DULL GRASP the VRF is implied by the interface across which DULL GRASP operates.

- If local address is "any", the local address used when initiating a secure channel connection is decided by source address selection ([RFC6724] for IPv6). As a responder, the connection listens on all addresses of the node in the selected vrf.

- Configuration of port is only required for methods where no defaults exist (e.g.: "dTLS").

- If remote address is "any", the connection is only a responder. It is a "hub" that can be used by multiple remote peers to connect simultaneously - without having to know or configure their addresses. Example: Hub site for remote "spoke" sites reachable over the Internet.

- Pmtu should be configurable to overcome issues/limitations of PMTUD (Path MTU Discovery).

- IKEv2/IPsec to remote peers should support the optional NAT-T procedures (NAT Traversal).

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 native 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 nodes 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 nodes.

- If the domain certificate of an existing ACP node 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 node 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. Nodes 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 nodes 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 nodes can be enrolled everywhere. Since all nodes 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 which the ACP is running. Especially if bringing down the ACP is known to disconnect the operator from the node. 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 nodes 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 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 nodes 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 cannot 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 nodes, 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 node 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 (e.g.: bootstrap), that explain decisions made in this document (e.g.: choice of GRASP) or that explain desirable extensions or implementation details for the ACP that are not considered to be appropriate to standardize in this document.

10.1. BRSKI Bootstrap (ANI)

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) describes how nodes with an IDevID certificate can securely and zero-touch enroll with a domain certificate (LDevID) to support the ACP. BRSKI also leverages the ACP to enable zero touch bootstrap of new nodes across networks without any configuration requirements across the transit nodes (e.g.: no DHCP/DS forwarding/server setup). This includes otherwise not configured 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. Nodes supporting such an infrastructure (BRSKI and ACP) are called ANI nodes (Autonomic Networking Infrastructure), see [I-D.ietf-anima-reference-model]. Nodes that do not support an IDevID but only an (insecure) vendor specific Unique Device Identifier (UDI) or nodes 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 node (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 the 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 node fails to connect to the ACP within its certificate lifetime, it cannot connect to the ACP to renew its certificate across it (using just EST), but it can still renew its certificate as an "enrolled/expired pledge" via the BRSKI bootstrap proxy. This requires only that the BRSKI registrar 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 could also render CRLs unnecessary because the BRSKI registrar in conjunction with the CA would not renew revoked certificates - only a "no-not-renew" list would be necessary on registrars/CA.

In the absence of BRSKI or less secure variants thereof, provisioning of certificates may involve one or more touches or non-standardized automation. Node vendors usually support provisioning of certificates into nodes via PKCS#7 (see [RFC2315]) and may support this provisioning through vendor specific models via Netconf ([RFC6241]). If such nodes also support Netconf Zero-Touch ([I-D.ietf-netconf-zerotouch]) then this can be combined to zero-touch provisioning of domain certificates into nodes. 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 a not configured network though.

10.2. ACP (and BRSKI) Diagnostics

Even though ACP and ANI in general are taking out many manual configuration mistakes through their automation, it is important to provide good diagnostics for them.

The basic diagnostics is support of (yang) data models representing the complete (auto-)configuration and operational state of all components: BRSKI, GRASP, ACP and the infrastructure used by them: TLS/dTLS, IPsec, certificates, trust anchors, time, VRF and so on. While necessary, this is not sufficient:

Simply representing the state of components does not allow operators to quickly take action - unless they do understand how to interpret the data, and that can mean a requirement for deep understanding of all components and how they interact in the ACP/ANI.

Diagnostic supports should help to quickly answer the questions operators are expected to ask, such as "is the ACP working correctly ?", or "why is there no ACP connection to a known neighboring node ?"

In current network management approaches, the logic to answer these questions is most often built as centralized diagnostics software that leverages the above mentioned data models. While this approach is feasible for components utilizing the ANI, it is not sufficient to diagnose the ANI itself:

- Developing the logic to identify common issues requires operational experience with the components of the ANI. Letting each management system define its own analysis is inefficient. As

much as possible, future work should attempt to standardize data models that support common error diagnostic.

- When the ANI is not operating correctly, it may not be possible to run diagnostics from remote because of missing connectivity. The ANI should therefore have diagnostic capabilities available locally on the nodes themselves.

- Certain operations are difficult or impossible to monitor in real-time, such as initial bootstrap issues in a network location where no capabilities exist to attach local diagnostics. Therefore it is important to also define means of capturing (logging) diagnostics locally for later retrieval. Ideally, these captures are also non-volatile so that they can survive extended power-off conditions - for example when a device that fails to be brought up zero-touch is being sent back for diagnostics at a more appropriate location.

The most simple form of diagnostics answering questions like the above is to represent the relevant information sequentially in dependency order, so that the first non-expected/non-operational item is the most likely root cause. Or just log/highlight that item. For example:

Q: Is ACP operational to accept neighbor connections:

- Check if any potentially necessary configuration to make ACP/ANI operational are correct (see Section 10.3 for a discussion of such commands).

- Does the system time look reasonable, or could it be the default system time after clock chip battery failure (certificate checks depend on reasonable notion of time).

- Does the node have keying material - domain certificate, trust anchors.

- If no keying material and ANI is supported/enabled, check the state of BRSKI (not detailed in this example).

- Check the validity of the domain certificate:

  * Does the certificate authenticate against the trust anchor ?

  * Has it been revoked ?

  * Was the last scheduled attempt to retrieve a CRL successful (e.g.: do we know that our CRL information is up to date).

    * Is the certificate valid: validity start time in the past, expiration time in the future ?

    * Does the certificate have a correctly formatted ACP information field ?

- Was the ACP VRF successfully created ?

- Is ACP enabled on one or more interfaces that are up and running ?

If all this looks good, the ACP should be running locally "fine" - but we did not check any ACP neighborships.

Question: why does the node not create a working ACP connection to a neighbor on an interface ?

- Is the interface physically up ? Does it have an IPv6 link-local address ?

- Is it enabled for ACP ?

- Do we successfully send DULL GRASP messages to the interface (link layer errors) ?

- Do we receive DULL GRASP messages on the interface ? If not, some intervening L2 equipment performing bad MLD snooping could have caused problems. Provide e.g.: diagnostics of the MLD querier IPv6 and MAC address.

- Do we see the ACP objective in any DULL GRASP message from that interface ? Diagnose the supported secure channel methods.

- Do we know the MAC address of the neighbor with the ACP objective ? If not, diagnose SLAAC/ND state.

- When did we last attempt to build an ACP secure channel to the neighbor ?

- If it failed, why:

    * Did the neighbor close the connection on us or did we close the connection on it because the domain certificate membership failed ?

    * If the neighbor closed the connection on us, provide any error diagnostics from the secure channel protocol.

    * If we failed the attempt, display our local reason:

+ There was no common secure channel protocol supported by the two neighbors (this could not happen on nodes supporting this specification because it mandates common support for IPsec).

+ The ACP domain certificate membership check (Section 6.1.2) fails:

  - The neighbors certificate does not have the required trust anchor. Provide diagnostics which trust anchor it has (can identify whom the device belongs to).

  - The neighbors certificate does not have the same domain (or no domain at all). Diagnose domain-name and potentially other other cert info.

  - The neighbors certificate has been revoked or could not be authenticated by OCSP.

  - The neighbors certificate has expired - or is not yet valid.

* Any other connection issues in e.g.: IKEv2 / IPsec, dTLS ?".

Question: Is the ACP operating correctly across its secure channels ?:

o Are there one or more active ACP neighbors with secure channels ?

o Is the RPL routing protocol for the ACP running ?

o Is there a default route to the root in the ACP routing table ?

o Is there for each direct ACP neighbor not reachable over the ACP virtual interface to the root a route in the ACP routing table ?

o Is ACP GRASP running ?

o Is at least one SRV.est objective cached (to support certificate renewal) ?

o Is there at least one BRSKI registrar objective cached (in case BRSKI is supported)

o Is BRSKI proxy operating normally on all interfaces where ACP is operating ?

o ...

These lists are not necessarily complete, but illustrate the principle and show that there are variety of issues ranging from normal operational causes (a neighbor in another ACP domain) over problems in the credentials management (certificate lifetimes), explicit security actions (revocation) or unexpected connectivity issues (intervening L2 equipment).

The items so far are illustrating how the ANI operations can be diagnosed with passive observation of the operational state of its components including historic/cached/counted events. This is not necessary sufficient to provide good enough diagnostics overall:

The components of ACP and BRSKI are designed with security in mind but they do not attempt to provide diagnostics for building the network itself. Consider two examples:

1. BRSKI does not allow for a neighboring device to identify the pledges certificate (IDevID). Only the selected BRSKI-registrar can do this, but it may be difficult to disseminate information about undesired pledges from those registrars to locations/nodes where information about those pledges is desired.

2. LLDP disseminates information about nodes to their immediate neighbors, such as node model/type/software and interface name/number of the connection. This information is often helpful or even necessary in network diagnostics. It can equally considered to be too insecure to make this information available unprotected to all possible neighbors.

An "interested adjacent party" can always determine the IDevID of a BRSKI pledge by behaving like a BRSKI proxy/registrar. Therefore the IDevID of a BRSKI pledge is not meant to be protected - it just has to be queried and is not signaled unsolicited (as it would be in LLDP) so that other observers on the same subnet can determine who is an "interested adjacent party".

Desirable options for additional diagnostics subject to future work include:

1. Determine if LLDP should be a recommended functionality for ANI devices to improve diagnostics, and if so, which information elements it should signal (insecure).

2. In alternative to LLDP, A DULL GRASP diagnostics objective could be defined to carry these information elements.

3. The IDevID of BRSKI pledges should be included in the selected insecure diagnostics option.

4. A richer set of diagnostics information should be made available via the secured ACP channels, using either single-hop GRASP or network wide "topology discovery" mechanisms.

10.3. Enabling and disabling ACP/ANI

Both ACP and BRSKI require interfaces to be operational enough to support sending/receiving their packets. In node types where interfaces are by default (e.g.: without operator configuration) enabled, such as most L2 switches, this would be less of a change in behavior than in most L3 devices (e.g.: routers), where interfaces are by default disabled. In almost all network devices it is common though for configuration to change interfaces to a physically disabled state and that would break the ACP.

In this section, we discuss a suggested operational model to enable/disable interfaces and nodes for ACP/ANI in a way that minimizes the risk of operator action to break the ACP in this way, and that also minimizes operator surprise when ACP/ANI becomes supported in node software.

10.3.1. Filtering for non-ACP/ANI packets

Whenever this document refers to enabling an interface for ACP (or BRSKI), it only requires to permit the interface to send/receive packets necessary to operate ACP (or BRSKI) - but not any other data-plane packets. Unless the data-plane is explicitly configured/enabled, all packets not required for ACP/BRSKI should be filtered on input and output:

Both BRSKI and ACP require link-local only IPv6 operations on interfaces and DULL GRASP. IPv6 link-local operations means the minimum signaling to auto-assign an IPv6 link-local address and talk to neighbors via their link-local address: SLAAC (Stateless Address Auto-Configuration - [RFC4862]) and ND (Neighbor Discovery - [RFC4861]). When the device is a BRSKI pledge, it may also require TCP/TLS connections to BRSKI proxies on the interface. When the device has keying material, and the ACP is running, it requires DULL GRASP packets and packets necessary for the secure-channel mechanism it supports, e.g.: IKEv2 and IPsec ESP packets or dTLS packets to the IPv6 link-local address of an ACP neighbor on the interface. It also requires TCP/TLS packets for its BRSKI proxy functionality, if it does support BRSKI.

10.3.2. Admin Down State

Interfaces on most network equipment have at least two states: "up" and "down". These may have product specific names. "down" for example could be called "shutdown" and "up" could be called "no shutdown". The "down" state disables all interface operations down to the physical level. The "up" state enables the interface enough for all possible L2/L3 services to operate on top of it and it may also auto-enable some subset of them. More commonly, the operations of various L2/L3 services is controlled via additional node-wide or interface level options, but they all become only active when the interface is not "down". Therefore an easy way to ensure that all L2/L3 operations on an interface are inactive is to put the interface into "down" state. The fact that this also physically shuts down the interface is in many cases just a side effect, but it may be important in other cases (see below).

To provide ACP/ANI resilience against operators configuring interfaces to "down" state, this document recommends to separate the "down" state of interfaces into an "admin down" state where the physical layer is kept running and ACP/ANI can use the interface and a "physical down" state. Any existing "down" configurations would map to "admin down". In "admin down", any existing L2/L3 services of the data-plane should see no difference to "physical down" state. To ensure that no data-plane packets could be sent/received, packet filtering could be established automatically as described above in Section 10.3.1.

As necessary (see discussion below) new configuration options could be introduced to issue "physical down". The options should be provided with additional checks to minimize the risk of issuing them in a way that breaks the ACP without automatic restoration. For example they could be denied to be issued from a control connection (netconf/ssh) that goes across the interface itself ("do not disconnect yourself"). Or they could be performed only temporary and only be made permanent with additional later reconfirmation.

In the following sub-sections important aspects to the introduction of "admin down" state are discussed.

10.3.2.1. Security

Interfaces are physically brought down (or left in default down state) as a form of security. "Admin down" state as described above provides also a high level of security because it only permits ACP/ANI operations which are both well secured. Ultimately, it is subject to security review for the deployment whether "admin down" is a feasible replacement for "physical down".

The need to trust into the security of ACP/ANI operations need to be weighed against the operational benefits of permitting this: Consider the typical example of a CPE (customer premises equipment) with no on-site network expert. User ports are in physical down state unless explicitly configured not to be. In a misconfiguration situation, the uplink connection is incorrectly plugged into such a user port. The device is disconnected from the network and therefore no diagnostics from the network side is possible anymore. Alternatively, all ports default to "admin down". The ACP (but not the data-plane) would still automatically form. Diagnostics from the network side is possible and operator reaction could include to either make this port the operational uplink port or to instruct re-cabling. Security wise, only ACP/ANI could be attacked, all other functions are filtered on interfaces in "admin down" state.

10.3.2.2. Fast state propagation and Diagnostics

"Physical down" state propagates on many interface types (e.g.: Ethernet) to the other side. This can trigger fast L2/L3 protocol reaction on the other side and "admin down" would not have the same (fast) result.

Bringing interfaces to "physical down" state is to the best of our knowledge always a result of operator action, but today, never the result of (autonomic) L2/L3 services running on the nodes. Therefore one option is to change the operator action to not rely on link-state propagation anymore. This may not be possible when both sides are under different operator control, but in that case it is unlikely that the ACP is running across the link and actually putting the interface into "physical down" state may still be a good option.

Ideally, fast physical state propagation is replaced by fast software driven state propagation. For example a DULL GRASP "admin-state" objective could be used to autoconfigure a BFD session between the two sides of the link that would be used to propagate the "up" vs. admin down state.

Triggering physical down state may also be used as a mean of diagnosing cabling in the absence of easier methods. It is more complex than automated neighbor diagnostics because it requires coordinated remote access to both (likely) sides of a link to determine whether up/down toggling will cause the same reaction on the remote side.

See Section 10.2 for a discussion about how LLDP and/or diagnostics via GRASP could be used to provide neighbor diagnostics, and therefore hopefully eliminating the need for "physical down" for neighbor diagnostics - as long as both neighbors support ACP/ANI.

10.3.2.3. Low Level Link Diagnostics

"Physical down" is performed to diagnose low-level interface behavior when higher layer services (e.g.: IPv6) are not working. Especially Ethernet links are subject to a wide variety of possible wrong configuration/cablings if they do not support automatic selection of variable parameters such as speed (10/100/1000 Mbps), crossover (Auto-MDIX) and connector (fiber, copper - when interfaces have multiple but can only enable one at a time). The need for low level link diagnostic can therefore be minimized by using fully autoconfiguring links.

In addition to "Physical down", low level diagnostics of Ethernet or other interfaces also involve the creation of other states on interfaces, such as physical loopback (internal and/or external) or bringing down all packet transmissions for reflection/cable-length measurements. Any of these options would disrupt ACP as well.

In cases where such low-level diagnostics of an operational link is desired but where the link could be a single point of failure for the ACP, ASA on both nodes of the link could perform a negotiated diagnostics that automatically terminates in a predetermined manner without dependence on external input ensuring the link will become operational again.

10.3.2.4. Power Consumption

Power consumption of "physical down" interfaces may be significantly lower than those in "admin down" state, for example on long range fiber interfaces. Assuming reasonable clocks on devices, mechanisms for infrequent periodic probing could allow to automatically establish ACP connectivity across such links. Bring up interfaces for 5 seconds to probe if there is an ACP neighbor on the remote end every 500 seconds = 1% power consumption.

10.3.3. Interface level ACP/ANI enable

The interface level configuration option "ACP enable" enables ACP operations on an interface, starting with ACP neighbor discovery via DULL GRAP. The interface level configuration option "ANI enable" on nodes supporting BRSKI and ACP starts with BRSKI pledge operations when there is no domain certificate on the node. On ACP/BRSKI nodes, "ACP enable" may not need to be supported, but only "ANI enable". Unless overridden by global configuration options (see later), "ACP/ANI enable" will result in "down" state on an interface to behave as "admin down".

10.3.4. Which interfaces to auto-enable ?

(Section 6.3) requires that "ACP enable" is automatically set on native interfaces, but not on non-native interfaces (reminder: a native interface is one that exists without operator configuration action such as physical interfaces in physical devices).

Ideally, ACP enable is set automatically on all interfaces that provide access to additional connectivity that allows to reach more nodes of the ACP domain. The best set of interfaces necessary to achieve this is not possible to determine automatically. Native interfaces are the best automatic approximation.

Consider an ACP domain of ACP nodes transitively connected via native interfaces. A data-plane tunnel between two of these nodes that are non-adjacent is created and "ACP enable" is set for that tunnel. ACP RPL sees this tunnel as just as a single hop. Routes in the ACP would use this hop as an attractive path element to connect regions adjacent to the tunnel nodes. In result, the actual hop-by-hop paths used by traffic in the ACP can become worse. In addition, correct forwarding in the ACP now depends on correct data-plane forwarding config including QoS, filtering and other security on the data-plane path across which this tunnel runs. This is the main issue why "ACP/ANI enable" should not be set automatically on non-native interfaces.

If the tunnel would connect two previously disjoint ACP regions, then it likely would be useful for the ACP. A data-plane tunnel could also run across nodes without ACP and provide additional connectivity for an already connected ACP network. The benefit of this additional ACP redundancy has to be weighed against the problems of relying on the data-plane. If a tunnel connects two separate ACP regions: how many tunnels should be created to connect these ACP regions reliably enough ? Between which nodes ? These are all standard tunneled network design questions not specific to the ACP, and there are no generic fully automated answers.

Instead of automatically setting "ACP enable" on these type of interfaces, the decision needs to be based on the use purpose of the non-native interface and "ACP enable" needs to be set in conjunction with the mechanism through which the non-native interface is created/configured.

In addition to explicit setting of "ACP/ANI enable", non-native interfaces also need to support configuration of the ACP RPL cost of the link - to avoid the problems of attracting too much traffic to the link as described above.

Even native interfaces may not be able to automatically perform BRSKI or ACP because they may require additional operator input to become operational. Example include DSL interfaces requiring PPPoE credentials or mobile interfaces requiring credentials from a SIM card. Whatever mechanism is used to provide the necessary config to the device to enable the interface can also be expanded to decide on whether or not to set "ACP/ANI enable".

The goal of automatically setting "ACP/ANI enable" on interfaces (native or not) is to eliminate unnecessary "touches" to the node to make its operation as much as possible "zero-touch" with respect to ACP/ANI. If there are "unavoidable touches" such a creating/ configuring a non-native interface or provisioning credentials for a native interface, then "ACP/ANI enable" should be added as an option to that "touch". If a wrong "touch" is easily fixed (not creating another high-cost touch), then the default should be not to enable ANI/ACP, and if it is potentially expensive or slow to fix (e.g.: parameters on SIM card shipped to remote location), then the default should be to enable ACP/ANI.

10.3.5. Node Level ACP/ANI enable

A node level command "ACP/ANI enable [up-if-only]" enables ACP or ANI on the node (ANI = ACP + BRSKI). Without this command set, any interface level "ACP/ANI enable" is ignored. Once set, ACP/ANI will operate interface where "ACP/ANI enable" is set. Setting of interface level "ACP/ANI enable" is either automatic (default) or explicit through operator action as described in the previous section.

If the option "up-if-only" is selected, the behavior of "down" interfaces is unchanged, and ACP/ANI will only operate on interfaces where "ACP/ANI enable" is set and that are "up". When it is not set, then "down" state of interfaces with "ACP/ANI enable" is modified to behave as "admin down".

10.3.5.1. Brownfield nodes

A "brownfield" node is one that already has a configured data-plane.

Executing global "ACP/ANI enable [up-if-only]" on each node is the only command necessary to create an ACP across a network of brownfield nodes once all the nodes have a domain certificate. When BRSKI is used ("ANI enable"), provisioning of the certificates only requires set-up of a single BRSKI-registrar node which could also implement a CA for the network. This is the most simple way to introduce ACP/ANI into existing (== brownfield) networks.

The need to explicitly enable ACP/ANI is especially important in brownfield nodes because otherwise software updates may introduce support for ACP/ANI: Automatic enablement of ACP/ANI in networks where the operator does not only not want ACP/ANI but where he likely never even heard of it could be quite irritating to him. Especially when "down" behavior is changed to "admin down".

Automatically setting "ANI enable" on brownfield nodes where the operator is unaware of it could also be a critical security issue depending on the vouchers used by BRKSI on these nodes. An attacker could claim to be the owner of these devices and create an ACP that the attacker has access/control over. In network where the operator explicitly wants to enable the ANI this could not happen, because he would create a BRSKI registrar that would discover attack attempts. Nodes requiring "ownership vouchers" would not be subject to that attack. See [I-D.ietf-anima-bootstrapping-keyinfra] for more details. Note that a global "ACP enable" alone is not subject to these type of attacks, because it always depends on some other mechanism first to provision domain certificates into the device.

10.3.5.2. Greenfield nodes

A "greenfield" node is one that did not have any prior configuration.

For greenfield nodes, only "ANI enable" is relevant. If another mechanism than BRSKI is used to (zero-touch) bootstrap a node, then it is up to that mechanism to provision domain certificates and to set global "ACP enable" as desired.

Nodes supporting full ANI functionality set "ANI enable" automatically when they decide that they are greenfield, e.g.: that they are powering on from factory condition. They will then put all native interfaces into "admin down" state and start to perform BRSKI pledge functionality - and once a domain certificate is enrolled they automatically enable ACP.

Attempts for BRSKI pledge operations in greenfield state should terminate automatically when another method of configuring the node is used. Methods that indicate some form of physical possession of the device such as configuration via the serial console could lead to immediate termination of BRSKI, while other parallel autoconfiguration methods subject to remote attacks might lead to BRSKI termination only after they were successful. Details of this may vary widely over different type of nodes. When BRSKI pledge operation terminates, this will automatically unset "ANI enable" and should terminate any temporarily needed state on the device to perform BRSKI - DULL GRASP, BRSKI pledge and any IPv6 configuration on interfaces.

10.3.6. Undoing ANI/ACP enable

Disabling ANI/ACP by undoing "ACP/ANI enable" is a risk for the reliable operations of the ACP if it can be executed by mistake or unauthorized. This behavior could be influenced through some additional property in the certificate (e.g.: in the domain information extension field) subject to future work: In an ANI deployment intended for convenience, disabling it could be allowed without further constraints. In an ANI deployment considered to be critical more checks would be required. One very controlled option would be to not permit these commands unless the domain certificate has been revoked or is denied renewal. Configuring this option would be a parameter on the BRSKI registrar(s). As long as the node did not receive a domain certificate, undoing "ANI/ACP enable" should not have any additional constraints.

10.3.7. Summary

Node-wide "ACP/ANI enable [up-if-only]" commands enable the operation of ACP/ANI. This is only auto-enabled on ANI greenfield devices, otherwise it must be configured explicitly.

If the option "up-if-only" is not selected, interfaces enabled for ACP/ANI interpret "down" state as "admin down" and not "physical down". In "admin-down" all non-ACP/ANI packets are filtered, but the physical layer is kept running to permit ACP/ANI to operate.

(New) commands that result in physical interruption ("physical down", "loopback) of ACP/ANI enabled interfaces should be built to protect continuance or reestablishment of ACP as much as possible.

Interface level "ACP/ANI enable" control per-interface operations. It is enabled by default on native interfaces and has to be configured explicitly on other interfaces.

Disabling "ACP/ANI enable" global and per-interface should have additional checks to minimize undesired breakage of ACP. The degree of control could be a domain wide parameter in the domain certificates.

10.4. ACP Neighbor discovery protocol selection

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

10.4.1. 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.

10.4.2. 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 nodes. With low performance of software forwarding in many L2 switches, this could also make the ACP risky to support on such L2 switches.

10.4.3. 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 node, 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.

10.5. 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 synchronization.

- 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 behavior.

- 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).

10.6. 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 multitude 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 nodes 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.

10.7. 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 domain 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 nodes using certificates from the same CA using the same domain field. GRASP inside the ACP is run across all transitively connected ACP nodes in a domain.

The rsub element in the domain 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 (e.g.: via GRASP). This is subject for further work.

RPL scales very well. It is not necessary to use multiple routing subdomains to scale ACP 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 domain 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 domain information element of certificates.

It is not necessary to have a separate 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 nodes, 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 built 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 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 (e.g.: GRASP negotiation).

10.8. Adopting ACP concepts for other environments

The ACP as specified in this document is very explicit about the choice of options to allow interoperable implementations. The choices made may not be the best for all environments, but the concepts used by the ACP can be used to build derived solutions:

The ACP specifies the use of ULA and deriving its prefix from the domain name so that no address allocation is required to deploy the ACP. The ACP will equally work not using ULA but any other /50 IPv6 prefix. This prefix could simply be a configuration of the registrars when using BRSKI to enroll the domain certificates - instead of the registrar deriving the /50 ULA prefix from the AN domain name.

Some solutions may already have an auto-addressing scheme, for example derived from existing unique device identifiers (e.g.: MAC addresses). In those cases it may not be desirable to assign addresses to devices via the ACP address information field in the way described in this document. The certificate may simply serve to identify the ACP domain, and the address field could be empty/unused. The only fix required in the remaining way the ACP operate is to define another element in the domain certificate for the two peers to decide who is Alice and who is Bob during secure channel building. Note though that future work may leverage the acp address to authenticate "ownership" of the address by the device. If the address used by a device is derived from some pre-existing permanent local ID (such as MAC address), then it would be useful to store that address in the certificate using the format of the access address information field or in a similar way.

The ACP is defined as a separate VRF because it intends to support well managed networks with a wide variety of configurations. Therefore, reliable, configuration-indestructible connectivity cannot be achieved from the data-plane itself. In solutions where all transit connectivity impacting functions are fully automated (including security), indestructible and resilient, it would be possible to eliminate the need for the ACP to be a separate VRF. Consider the most simple example system in which there is no separate data-plane, but the ACP is the data-plane. Add BRSKI, and it becomes a fully autonomic network - except that it does not support automatic addressing for user equipment. This gap can then be closed for example by adding a solution derived from [I-D.ietf-anima-prefix-management].

The routing protocol chosen by the ACP design (RPL) does explicitly not optimize for shortest paths and fastest convergence. Variations of the ACP may want to use a different routing protocol.

Variations such as what routing protocol to use, or whether to instantiate an ACP in a VRF or (as suggested above) as the actual data-plane, can be automatically chosen in implementations built to support multiple options by deriving them from future parameters in the certificate. Parameters in certificates should be limited to those that would not need to be changed more often than certificates

would need to be updated anyhow; Or by ensuring that these parameters can be provisioned before the variation of an ACP is activated in a node. Using BRSKI, this could be done for example as additional follow-up signaling directly after the certificate enrolment, still leveraging the BRSKI TLS connection and therefore not introducing any additional connectivity requirements.

Last but not least, secure channel protocols including their encapsulation are easily added to ACP solutions. Secure channels may even be replaced by simple neighbor authentication to create simplified ACP variations for environments where no real security is required but just protection against non-malicious misconfiguration. Or for environments where all traffic is known or forced to be end-to-end protected and other means for infrastructure protection are used. Any future network OAM should always use end-to-end security anyhow and can leverage the domain certificates and is therefore not dependent on security to be provided for by ACP secure channels.

11. 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 node.

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.

Many details of ACP are designed with security in mind and discussed elsewhere in the document:

IPv6 addresses used by nodes in the ACP are covered as part of the nodes domain certificate as described in Section 6.1.1. This allows even verification of ownership of a peers IPv6 address when using a connection authenticated with the domain certificate.

The ACP acts as a security (and transport) substrate for GRASP inside the ACP such that GRASP is not only protected by attacks from the outside, but also by attacks from compromised inside attackers - by relying not only on hop-by-hop security of ACP secure channels, but adding end-to-end security for those GRASP messages. See Section 6.8.2.

ACP provides for secure, resilient zero-touch discovery of EST servers for certificate renewal. See Section 6.1.3.

ACP provides extensible, auto-configuring hop-by-hop protection of the ACP infrastructure via the negotiation of hop-by-hop secure channel protocols. See Section 6.5 and Section 10.6.

The ACP is designed to minimize attacks from the outside by minimizing its dependency against any non-ACP operations on a node. The only dependency in the specification in this document is the need to share link-local addresses for the ACP secure channel encapsulation with the data-plane. See Section 6.12.2.

In combination with BRSKI, ACP enables a resilient, fully zero-touch network solution for short-lived certificates that can be renewed or re-enrolled even after unintentional expiry (e.g.: because of interrupted connectivity). See Section 10.1.

12. IANA Considerations

This document defines the "Autonomic Control Plane".

The IANA is requested to register the value "AN_ACP" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.3.

The IANA is requested to register the value "SRV.est" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.1.3.

Note that the objective format "SRV.<service-name>" is intended to be used for any <service-name> that is an [RFC6335] registered service name. This is a proposed update to the GRASP registry subject to future work and only mentioned here for informational purposed to explain the unique format of the objective name.

The IANA is requested to create an ACP Parameter Registry with currently one registry table - the "ACP Address Type" table.

The IANA is requested to create an ACP Parameter Registry with currently one registry table - the "ACP Address Type" table.

"ACP Address Type" Table. The value in this table are numeric values 0...3 paired with a name (string). Future values MUST be assigned using the Standards Action policy defined by [RFC8126]. The following initial values are assigned by this document:

0: ACP Zone Addressing Sub-Scheme (ACP RFC Figure 4) / ACP Manual Addressing Sub-Scheme (ACP RFC Section 6.10.4)
1: ACP Vlong Addressing Sub-Scheme (ACP RFC Section 6.10.5)

13. 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 Brian Carpenter and Sheng Jiang for their thorough reviews and to Pascal Thubert and Michael Richardson to provide the details for the recommendations of the use of RPL in the ACP

Further input, review or suggestions were received from: Rene Struik, Brian Carpenter, Benoit Claise, William Atwood and Yongkang Zhang.

14. Change log [RFC Editor: Please remove]

14.1. Initial version

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

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

Initial version of the anima document; only minor edits.

14.3. 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.

14.4. 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

14.5. 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.

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

No changes; re-submitted as WG document.

14.7. 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.

14.8. 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.

14.9. 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.

14.10. 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).

14.11. 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).

14.12. 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.

14.13. 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 signaled via GRASP.

14.14. 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 Zero-Touch. 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, e.g.: 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.

14.15. 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 e.g.: 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 (e.g.: 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 (e.g.: 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.
- black hole 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.

14.16. draft-ietf-anima-autonomic-control-plane-10

Used the term routing subdomain more consistently where previously only subdomain was used. Clarified use of routing subdomain in creation of ULA "global ID" addressing prefix.

6.7.1.* Changed native IPsec encapsulation to tunnel mode (necessary), explaned why. Added notion that ESP is used, added explanations why tunnel/transport mode in native vs. GRE cases.

6.10.3/6.10.5 Added term "ACP address range/set" to be able to better explain how the address in the ACP certificate is actually the base address (lowest address) of a range/set that is available to the device.

6.10.4 Added note that manual address sub-scheme addresses must not be used within domain certificates (only for explicit configuration).

6.12.5 Refined explanation of how ACP virtual interfaces work (p2p and multipoint). Did seek for pre-existing RFCs that explain how to built a multi-access interface on top of a full mesh of p2p connections (6man WG, anima WG mailing lists), but could not find any prior work that had a succinct explanation. So wrote up an explanation here. Added hopefully all necessary and sufficient details how to map ACP unicast packets to ACP secure channel, how to deal with ND packet details. Added verbage for ACP not to assign the virtual interface link-local address from the underlying interface. Addd note that GRAP link-local messages are treated specially but logically the same. Added paragraph about NBMA interfaces.

remaining changes from Brian Carpenters review. See Github file draft-ietf-anima-autonomic-control-plane/08-carpenter-review-reply.tx for more detailst:

Added multiple new RFC references for terms/technologies used.

Fixed verbage in several places.

2. (terminology) Added 802.1AR as reference.

2. Fixed up definition of ULA.

6.1.1 Changed definition of ACP information in cert into ABNF format. Added warning about maximum size of ACP address field due to domain-name limitations.

6.2 Mentioned API requirement between ACP and clients leveraging adjacency table.

6.3 Fixed TTL in GRASP example: msec, not hop-count!.

6.8.2 MAYOR: expanded security/transport substrate text:

Introduced term ACP GRASP virtual interface to explain how GRASP link-local multicast messages are encapsulated and replicated to neighbors. Explain how ACP knows when to use TLS vs. TCP (TCP only for link-local address (sockets). Introduced "ladder" picture to visualize stack.

6.8.2.1 Expanded discussion/explanation of security model. TLS for GRASP unicsast connections across ACP is double encryption (plus underlying ACP secure channel), but highly necessary to avoid very simple man-in-the-middle attacks by compromised ACP members on-path. Ultimately, this is done to ensure that any apps using GRASP can get full end-to-end secrecy for information sent across GRASP. But for publically known ASA services, even this will not provide 100%

security (this is discussed). Also why double encryption is the better/easier solution than trying to optimize this.

6.10.1 Added discussion about pseudo-random addressing, scanning-attaacks (not an issue for ACP).

6.12.2 New performance requirements section added.

6.10.1 Added notion to first experiment with existing addressing schemes before defining new ones - we should be flexible enough.

6.3/7.2 clarified the interactions between MLD and DULL GRASP and specified what needs to be done (e.g.: in 2 switches doing ACP per L2 port).

12. Added explanations and cross-references to various security aspects of ACP discussed elsewhere in the document.

13. Added IANA requirements.

Added RFC2119 boilerplate.

14.17. draft-ietf-anima-autonomic-control-plane-11

Same text as -10 Unfortunately when uploading -10 .xml/.txt to datatracker, a wrong version of .txt got uploaded, only the .xml was correct. This impacts the -10 html version on datatra cker and the PDF versions as well. Because rfcdiff also compares the .txt version, this -11 version was crea ted so that one can compare changes from -09 and changes to the next version (-12).

14.18. draft-ietf-anima-autonomic-control-plane-12

Sheng Jiangs extensive review. Thanks! See Github file draft-ietf-anima-autonomic-control-plane/09-sheng-review-reply.txt for more details. Many of the larger changes listed below where inspired by the review.

Removed the claim that the document is updating RFC4291,RFC4193 and the section detailling it. Done on suggestion of Michael Richardson - just try to describe use of addressing in a way that would not suggest a need claim update to architecture.

Terminology cleanup:

- Replaced "device" with "node" in text. Kept "device" only when referring to "physical node". Added definitions for those words.

Includes changes of derived terms, especially in addressing: "Node-ID" and "Node-Number" in the addressing details.

- Replaced term "autonomic FOOBAR" with "acp FOOBAR" as whereever appropriate: "autonomic" would imply that the node would need to support more than the ACP, but that is not correct in most of the cases. Wanted to make sure that implementers know they only need to support/implement ACP - unless stated otherwise. Includes "AN->ACP node", "AN->ACP adjacency table" and so on.

1 Added explanation in the introduction about relationship between ACP, BRSKI, ANI and Autonomic Networks.

6.1.1 Improved terminology and features of the certificate information field. Now called domain information field instead of ACP information field. The acp-address field in the domain information field is now optional, enabling easier introduction of various future options.

6.1.2 Moved ACP domainer membership check from section 6.6 to (ACP secure channels setup) here because it is not only used for ACP secure channel setup.

6.1.3 Fix text about certificate renewal after discussion with Max Pritikin/Michael Richardson/Brian Carpenter:

- Version 10 erroneously assumed that the certificate itself could store a URL for renewal, but that is only possible for CRL URLs. Text now only refers to "remembered EST server" without implying that this is stored in the certificate.

- Objective for RFC7030/EST domain certificate renewal was changed to "SRV.est" See also IANA section for explanation.

- Removed detail of distance based service selection. This can be better done in future work because it would require a lot more detail for a good DNS-SD compatible approach.

- Removed detail about trying to create more security by using ACP address from certificate of peer. After rethinking, this does not seem to buy additional security.

6.10 Added reference to 6.12.5 in initial use of "loopback interface" in section 6.10 in result of email discussion michaelR/michaelB.

10.2 Introduced informational section (diagnostics) because of operational experience - ACP/ANI undeployable without at least diagnostics like this.

10.3 Introduced informational section (enabling/disabling) ACP. Important to discuss this for security reasons (e.g.: why to never never auto-enable ANI on brownfield devices), for implementers and to answer ongoing questions during WG meetings about how to deal with shutdown interface.

10.8 Added informational section discussing possible future variations of the ACP for potential adopters that cannot directly use the complete solution described in this document unmodified.

14.19. draft-ietf-anima-autonomic-control-plane-13

Swap author list (with permission).

6.1.1. Eliminate blank lines in definition by making it a picture (reformatting only).

6.10.3.1 New paragraph: Explained how nodes using Zone-ID != 0 need to use Zone-ID != 0 in GRASP so that we can avoid routing/forwarding of Zone-ID = 0 prefixes.

Rest of feedback from review of -12, see https://raw.githubusercontent.com/anima-wg/autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/12-feedback-reply.txt

Review from Brian Carpenter:

various: Autonomous -> autonomic(ally) in all remaining occurances.

various: changed "manual (configured)" to "explicitly (configured)" to not exclude the option of (SDN controller) automatic configuration (no humans involved).

1. Fixed reference to section 9.

2. Added definition of loopback interface == internal interface. After discus on WG mailing lists, including 6man.

6.1.3 Removed "EST-TLS", no objective value needed or beneficial, added explanation paragraph why.

6.2 Added to adjacency table the interface that a neighbor is discovered on.

6.3 Simplified CDDL syntax: Only one method per AN_ACP objective (because of locators). Example with two objectives in GRASP message.

6.8.1 Added note about link-local GRASP multicast message to avoid confusion.

8.1.4 Added RFC8028 as recommended on hosts to better support VRF-select with ACP.

8.2.1 Rewrote and Simplified CDDL for configured remote peer and explanations. Removed pattern option for remote peer. Not important enough to be mandated.

Review thread started by William Atwood:

2. Refined definition of VRF (vs. MPLS/VPN, LISP, VRF-LITE).

2. Refined definition of ACP (ACP includes ACP GRASP instance).

2. Added explanation for "zones" to terminology section and into Zone Addressing Sub Scheme section, relating it to RFC4007 zones (from Brian Carpenter).

4. Fixed text for ACP4 requirement (Clients of the ACP must not be tied to specific protocol.).

5. Fixed step 4. with proposed text.

6.1.1 Included suggested explanation for rsub semantics.

6.1.3 must->MUST for at least one EST server in ACP network to autonomically renew certs.

6.7.2 normative: AND MUST NOT (permit weaker crypto options.

6.7.1.1 also included text denying weaker IPsec profile options.

6.8.2 Fixed description how to build ACP GRASP virtual interfaces. Added text that ACP continues to exist in absence of ACP neighbors.

various: Make sure all "zone" words are used consistently.

6.10.2/various: fixed 40 bit RFC4193 ULA prefix in all examples to 89b714f3db (thanks MichaelR).

6.10.1 Removed comment about assigned ULA addressing. Decision not to use it now ancient history of WG decision making process, not worth nothing anymore in the RFC.

Review from Yongkang Zhang:

6.10.5 Fixed length of Node-Numbers in ACP Vlong Addressing Sub-Scheme.

15. References

15.1. 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.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://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, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7030>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

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

15.2. Informative References

[AR8021] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[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-09 (work in progress), October 2017.

[I-D.ietf-anima-prefix-management] Jiang, S., Du, Z., Carpenter, B., and Q. Sun, "Autonomic IPv6 Edge Prefix Management in Large-scale Networks", draft-ietf-anima-prefix-management-06 (work in progress), October 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-05 (work in progress), October 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-07 (work in progress), October 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-19 (work in progress), October 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-19 (work in progress), October 2017.

[MACSEC] IEEE SA-Standards Board, "IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security", June 2006, <https://standards.ieee.org/findstds/standard/802.1AE-2006.html>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

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

[RFC2821] Klensin, J., Ed., "Simple Mail Transfer Protocol", RFC 2821, DOI 10.17487/RFC2821, April 2001, <https://www.rfc-editor.org/info/rfc2821>.

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <https://www.rfc-editor.org/info/rfc4007>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <https://www.rfc-editor.org/info/rfc4541>.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <https://www.rfc-editor.org/info/rfc4604>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <https://www.rfc-editor.org/info/rfc4607>.

[RFC4610] Farinacci, D. and Y. Cai, "Anycast-RP Using Protocol Independent Multicast (PIM)", RFC 4610, DOI 10.17487/RFC4610, August 2006, <https://www.rfc-editor.org/info/rfc4610>.

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

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, DOI 10.17487/RFC5790, February 2010, <https://www.rfc-editor.org/info/rfc5790>.

[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, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc6724>.

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

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

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

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

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc7576>.

[RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <https://www.rfc-editor.org/info/rfc7721>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <https://www.rfc-editor.org/info/rfc7761>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <https://www.rfc-editor.org/info/rfc8028>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Authors' Addresses

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

Email: tte+ietf@cs.fau.de

Michael H. Behringer (editor)

Email: michael.h.behringer@gmail.com

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

Email: sbjarnason@arbor.net