ANIMA WG M. Behringer, Ed.
Internet-Draft S. Bjarnason
Intended status: Standards Track Balaji. BL
Expires: April 8, 2016 T. Eckert
Cisco Systems
October 6, 2015

An Autonomic Control Plane
draft-ietf-anima-autonomic-control-plane-01

Abstract

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

Status of This Memo

This Internet-Draft will expire on April 8, 2016.

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

This document describes options for a self-forming, self-managing and self-protecting "Autonomic Control Plane" (ACP) which is inband on the network, yet as independent as possible of configuration, addressing and routing problems (for details how this achieved, see Section 5). It therefore remains operational even in the presence of configuration errors, addressing or routing issues, or where policy could inadvertently affect control plane connectivity. The Autonomic Control Plane serves several purposes at the same time:

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

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

- A controller or network management system can use it to securely bootstrap network devices in remote locations, even if the network

in between is not yet configured; no data-plane dependent bootstrap configuration is required. An example of such a secure bootstrap process is described in [I-D.ietf-anima-bootstrapping-keyinfra]

This document describes some use cases for the ACP in Section 2, it defines the requirements in Section 3, Section 4 gives an overview how an Autonomic Control Plane is constructed, and in Section 5 the detailed process is explained. Section 6 explains how non-autonomic nodes and networks can be integrated, Section 7 defines the negotiation protocol, and Section 8 the first channel types for the ACP.

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

2. Use Cases for an Autonomic Control Plane

2.1. An Infrastructure for Autonomic Functions

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

2.2. Secure Bootstrap over an Unconfigured Network

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

With the ACP, secure bootstrap of new devices can happen without requiring any configuration on the network. A new device can automatically be bootstrapped in a secure fashion and be deployed with a domain certificate. This does not require any configuration on intermediate nodes, because they can communicate through the ACP.

2.3. Data Plane Independent Permanent Reachability

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

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

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

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

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

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

The document "Autonomic Network Stable Connectivity" [I-D.eckert-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.

3. Requirements

The Autonomic Control Plane has the following requirements:

1. 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.

2. 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).

3. 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.

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

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

The default mode of operation of the ACP is hop-by-hop, because this interaction can be built on IPv6 link local addressing, which is autonomic, and has no dependency on configuration (requirement 1). It may be necessary to have end-to-end connectivity in some cases, for example to provide an end-to-end security association for some protocols. This is possible, but then has a dependency on routable address space.

4. Overview

The Autonomic Control Plane is constructed in the following way (for details, see Section 5):

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

- It determines, following a policy, a candidate peer list. This is the list of nodes to which it should establish an autonomic control plane. Default policy is: To all adjacent nodes in the same domain. Intent can override this default policy.

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

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

- Inside the ACP VRF, each node sets up a virtual interface with its ULA IPv6 address.

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

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

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

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

The following figure illustrates the ACP.

```
           autonomic node 1                  autonomic node 2
          ...................               ...................
   secure .                 .   secure      .                 .  secure
   tunnel :  +-----------+  :   tunnel      :  +-----------+  :  tunnel
   ..--------| ACP VRF   |---------------------| ACP VRF   |---------..
          : / \         / \   <--routing-->   / \         / \ :
          : \ /         \ /                   \ /         \ / :
   ..--------|  virtual  |---------------------|  virtual  |---------..
          :  | interface |  :               :  | interface |  :
          :  +-----------+  :               :  +-----------+  :
          :                 :               :                 :
          :   data plane    :...............:   data plane    :
          :                 :    link       :                 :
          :.................:               :.................:
```

Figure 1

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

5. Self-Creation of an Autonomic Control Plane

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

5.1. Preconditions

An autonomic node can be a router, switch, controller, NMS host, or any other IP device. We assume an autonomic node has:

- A globally unique domain certificate, with which it can cryptographically assert its membership of the domain. The document [I-D.ietf-anima-bootstrapping-keyinfra] describes how a domain certificate can be automatically and securely derived from a vendor specific Unique Device Identifier (UDI) or IDevID certificate. (Note the UDI used in this document is NOT the UUID specified in [RFC4122].)

- An adjacency table, which contains information about adjacent autonomic nodes, at a minimum: node-ID, IP address, domain, certificate. An autonomic device maintains this adjacency table up to date. Where the next autonomic device is not directly adjacent, the information in the adjacency table can be supplemented by configuration. For example, the node-ID and IP address could be configured.

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

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

5.2. Candidate ACP Neighbor Selection

An autonomic node must determine to which other autonomic nodes in the adjacency table it should build an ACP connection.

The ACP is by default established exclusively between nodes in the same domain.

Intent can change this default behaviour. The precise format for this Intent needs to be defined outside this document. Example Intent policies are:

- The ACP should be built between all sub-domains for a given parent domain. For example: For domain "example.com", nodes of "example.com", "access.example.com", "core.example.com" and "city.core.example.com" should all establish one single ACP.

- Two domains should build one single ACP between themselves, for example "example1.com" should establish the ACP also with nodes from "example2.com". For this case, the two domains must be able to validate their trust, typically by cross-signing their certificate infrastructure.

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.

5.3. Capability Negotiation

Autonomic devices may have different capabilities based on the type of device, OS version, etc. To establish a trusted secure ACP channel, devices must first negotiate their mutual capabilities in the data plane. This allows for the support of different channel types in the future.

For each node on the candidate ACP neighbor list, capabilities need to be exchanged. The capability negotiation is based on GRASP [I-D.ietf-anima-grasp]. The relevant protocol details are defined in Section 7. This negotiation MUST be secure: The identity of the other node MUST be validated during capability negotiation, and the exchange MUST be authenticated.

The first parameter to be negotiated is the ACP Channel type. The channel types are defined in Section 8. Other parameters may be added later.

Intent may also influence the capability negotiation. For example, Intent may require a minimum ACP tunnel security. This is outside scope for this document.

5.4. Channel Establishment

After authentication and capability negotiation autonomic nodes establish a secure channel towards the AN neighbors with the above negotiated parameters.

The channel establishment MUST be authenticated. Whether or not, and how, a channel is encrypted is part of the capability negotiation, potentially controlled by Intent.

In order to be independent of configured link addresses, channels SHOULD use IPv6 link local addresses between adjacent neighbors wherever possible. This way, the ACP tunnels are independent of correct network wide routing.

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 5.7.

If two nodes are connected via several links, the ACP SHOULD be established on every link, but it is possible to establish the ACP only on a sub-set of links. Having an ACP channel on every link has a number of advantages, for example it allows for a faster failover in case of link failure, and it reflects the physical topology more closely. Using a subset of links (for example, a single link), reduces resource consumption on the devices, because state needs to be kept per ACP channel.

5.5. Context Separation

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

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

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

[EDNOTE: Previous versions of this document also discussed an option where the ACP runs in the data plane without logical separation. Consensus is to focus only on the separated ACP now, and to remove the ACP in the data plane from this document. See Appendix B for the reasons for this decision.]

5.6. Addressing inside the ACP

The channels explained above typically only establish communication between two adjacent nodes. In order for communication to happen across multiple hops, the autonomic control plane requires internal

network wide valid addresses and routing. Each autonomic node must create a virtual interface with a network wide unique address inside the ACP context mentioned in Section 5.5.

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.

Once an autonomic node is enrolled in a domain, it automatically creates a network wide Unique Local Addresses (ULA) in accordance with [RFC4193] with the following algorithm:

- Prefix FD00::/8, defining locally assigned unique local addresses. See Section 3.1 of [RFC4193].

- Global ID: an MD5 hash of the domain ID, using the 40 least significant bits. This results in a pseudo-random global ID, in accordance with Section 3.2 of [RFC4193].

- Subnet ID and interface ID: [I-D.behringer-anima-autonomic-addressing] defines how these fields can be constructed and used.

With this algorithm, all autonomic devices in the same domain have the same /48 prefix. Conversely, global IDs from different domains are unlikely to clash, such that two networks can be merged, as long as the policy allows that merge. See also Section 9 for a discussion on merging domains.

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

5.7. Routing in the ACP

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

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

All routing updates are automatically secured in transit as the channels of the autonomic control plane are by default secured.

The routing protocol inside the ACP should be light weight and highly scalable to ensure that the ACP does not become a limiting factor in network scalability. We suggest the use of RPL [RFC6550] as one such protocol which is light weight and scales well for the control plane traffic. See Appendix A for more details on the choice of RPL.

6. Workarounds for Non-Autonomic Nodes

6.1. Connecting a Non-Autonomic Controller / NMS system

The Autonomic Control Plane can be used by management systems, such as controllers or network management system (NMS) hosts (henceforth called simply "NMS hosts"), to connect to devices through it. For this, an NMS host must have access to the ACP. By default, the ACP is a self-protecting overlay network, which only allows access to trusted systems. Therefore, a traditional, non-autonomic NMS system does not have access to the ACP by default, just like any other external device.

If the NMS host is not autonomic, i.e., it does not support autonomic negotiation of the ACP, then it can be brought into the ACP by explicit configuration. On an adjacent autonomic node with ACP, the interface with the NMS host can be configured to be part of the ACP. In this case, the NMS host is with this interface entirely and exclusively inside the ACP. It would likely require a second interface for connections between the NMS host and administrators, or Internet based services. This mode of connecting an NMS host has security consequences: All systems and processes connected to this implicitly trusted interface have access to all autonomic nodes on the entire ACP, without further authentication. Thus, this connection must be physically controlled.

The non-autonomic NMS host must be routed in the ACP. This involves two parts: 1) the NMS host must point default to the AN device for the ULA prefix used inside the ACP, and 2) the prefix used between AN node and NMS host must be announced into the ACP, and distributed there.

The document "Autonomic Network Stable Connectivity" [I-D.eckert-anima-stable-connectivity] explains in more detail how the ACP can be integrated in a mixed NOC environment.

6.2. ACP through Non-Autonomic L3 Clouds

Not all devices in a network may be autonomic. If non-autonomic Layer-2 devices are between autonomic nodes, the communications described in this document should work, since it is IP based. However, non-autonomic Layer-3 devices do not forward link local autonomic messages, and thus break the Autonomic Control Plane.

One workaround is to manually configure IP tunnels between autonomic nodes across a non-autonomic Layer-3 cloud. The tunnels are represented on each autonomic node as virtual interfaces, and all autonomic transactions work across such tunnels.

Such manually configured tunnels are less "indestructible" than an automatically created ACP based on link local addressing, since they depend on correct data plane operations, such as routing and addressing.

7. The Negotiation Protocol

This section describes the negotiation exchange in detail. It is based on GRASP [I-D.ietf-anima-grasp]. Since at the time of establishing the ACP channel there is obviously no ACP yet, this negotiation protocol must run in the data plane. This negotiation MUST be authenticated, to avoid downgrade attackes, where an attacker injects bogus negotiation messages demanding a less secure ACP channel type. The negotiation MAY be encrypted.

[The detailed negotiation flow and mapping into GRASP messages is to be completed.]

8. The Channel Type

Two adjacent nodes negotiate an ACP channel. This channel MUST be authenticated and SHOULD be encrypted.

The nodes negotiate a parameter called "ACP channel type". This document defines a single, MUST implement channel type: GRE with

IPsec transport mode. See IANA Considerations (Section 13) for the formal definition of this parameter.

9. Self-Healing Properties

The ACP is self-healing:

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

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

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

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

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

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

Merging two networks with different trust anchors requires the trust anchors to mutually trust each other (for example, by cross-signing). As long as the domain names are different, the addressing will not overlap (see Section 5.6).

10. Self-Protection Properties

As explained in Section 5, the ACP is based on channels being built between devices which have been previously authenticated based on their domain certificates. The channels themselves are protected using standard encryption technologies like DTLS or IPsec which provide additional authentication during channel establishment, data integrity and data confidentiality protection of data inside the ACP and in addition, provide replay protection.

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

The 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.

11. The Administrator View

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

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

Since an ACP is self-protecting, a device not supporting the ACP, or without a valid domain certificate cannot connect to it. This means that by default a traditional controller or network management system

cannot connect to an ACP. See Section 6.1 for more details on how to connect an NMS host into the ACP.

12. Security Considerations

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

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

- The usage of domain certificates depends on a valid supporting PKI infrastructure. If the chain of trust of this PKI infrastructure is compromised, the security of the ACP is also compromised. This is typically under the control of the network administrator.
- Security can be compromised by implementation errors (bugs), as in all products.

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

13. IANA Considerations

Section 8 describes an option for the channel negotiation, the channel type. We request IANA to create a registry for ACP channel types.

The ACP channel type is a 8-bit unsigned integer. This document only assigns the first value.

Number	Channel Type	RFC
0	GRE tunnel protected with IPsec transport mode	this document
1-255	reserved for future channel types	

14. Acknowledgements

This work originated from an Autonomic Networking project at Cisco Systems, which started in early 2010. Many people contributed to this project and the idea of the Autonomic Control Plane, amongst which (in alphabetical order): Ignas Bagdonas, Parag Bhide, Alex

Clemm, Yves Hertoghs, Bruno Klauser, Max Pritikin, Ravi Kumar Vadapalli.

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

15. Change log [RFC Editor: Please remove]

15.1. Initial version

First version of this document: [I-D.behringer-autonomic-control-plane]

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

Initial version of the anima document; only minor edits.

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

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

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

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

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

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

- Took out requirement for IPv6 --> that's in the reference doc.

- Added requirement section.

- Changed focus: more focus on autonomic functions, not only virtual out of band. This goes a bit throughout the document, starting with a changed abstract and intro.

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

No changes; re-submitted as WG document.

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

- Added some paragraphs in addressing section on "why IPv6 only", to reflect the discussion on the list.

- Moved the data-plane ACP out of the main document, into an appendix. The focus is now the virtually separated ACP, since it has significant advantages, and isn't much harder to do.

- Changed the self-creation algorithm: Part of the initial steps go into the reference document. This document now assumes an adjacency table, and domain certificate. How those get onto the device is outside scope for this document.

- Created a new section 6 "workarounds for non-autonomic nodes", and put the previous controller section (5.9) into this new section. Now, section 5 is "autonomic only", and section 6 explains what to do with non-autonomic stuff. Much cleaner now.

- Added an appendix explaining the choice of RPL as a routing protocol.

- Formalised the creation process a bit more. Now, we create a "candidate peer list" from the adjacency table, and form the ACP with those candidates. Also it explains now better that policy (Intent) can influence the peer selection. (section 4 and 5)

- Introduce a section for the capability negotiation protocol (section 7). This needs to be worked out in more detail. This will likely be based on GRASP.

- Introduce a new parameter: ACP tunnel type. And defines it in the IANA considerations section. Suggest GRE protected with IPSec transport mode as the default tunnel type.

- Updated links, lots of small edits.

16. References

[I-D.behringer-anima-autonomic-addressing]
Behringer, M., "An Autonomic IPv6 Addressing Scheme", draft-behringer-anima-autonomic-addressing-01 (work in progress), June 2015.

[I-D.behringer-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Liu, B., Jeff, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-behringer-anima-reference-model-03 (work in progress), June 2015.

[I-D.behringer-autonomic-control-plane]
Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", draft-behringer-autonomic-control-plane-00 (work in progress), June 2014.

[I-D.eckert-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-eckert-anima-stable-connectivity-01 (work in progress), March 2015.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-ietf-anima-bootstrapping-keyinfra-00 (work in progress), August 2015.

[I-D.ietf-anima-grasp]
Carpenter, B. and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-00 (work in progress), August 2015.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

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

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

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

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

Appendix A. Background on the choice of routing protocol

In a pre-standard implementation, the "IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL, [RFC6550] was chosen. This Appendix explains the reasoning behind that decision.

Requirements for routing in the ACP are:

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

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

- Low resource consumption: The ACP supports traditional network infrastructure, thus runs in addition to traditional protocols. The ACP, and specifically the routing protocol must have low resource consumption both in terms of memory and CPU requirements. Specifically, at edge nodes, where memory and CPU are scarce, consumption should be minimal. RPL builds a destination-oriented

directed acyclic graph (DODAG), where the main resource consumption is at the root of the DODAG. The closer to the edge of the network, the less state needs to be maintained. This adapts nicely to the typical network design. Also, all changes below a common parent node are kept below that parent node.

- Support for unstructured address space: In the Autonomic Networking Infrastructure, node addresses are identifiers, and may not be assigned in a topological way. Also, nodes may move topologically, without changing their address. Therefore, the routing protocol must support completely unstructured address space. RPL is specifically made for mobile ad-hoc networks, with no assumptions on topologically aligned addressing.

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

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

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

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

Appendix B. Alternative: An ACP without Separation

Section 5 explains how the ACP is constructed as a virtually separated overlay network. An alternative ACP design can be achieved without the VRFs. In this case, the autonomic virtual addresses are part of the data plane, and subject to routing, filtering, QoS, etc on the data plane. The secure tunnels are in this case used by traffic to and from the autonomic address space. They are still

required to provide the authentication function for all autonomic packets.

At IETF 93 in Prague, the suggestion was made to not advance with the data plane ACP, and only continue with the virtually separate ACP. The reason for this decision is that the contextual separation of the ACP provides a range of benefits (more robustness, less potential interactions with user configurations), while it is not much harder to achieve.

This appendix serves to explain the decision; it will be removed in the next version of the draft.

Authors' Addresses

Michael H. Behringer (editor)
Cisco Systems
Building D, 45 Allee des Ormes
Mougins 06250
France

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco Systems

Email: sbjarnas@cisco.com

Balaji BL
Cisco Systems

Email: blbalaji@cisco.com

Toerless Eckert
Cisco Systems

Email: eckert@cisco.com