ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2016

M. Behringer, Ed.
Cisco Systems
S. Bjarnason
Balaji. BL
T. Eckert
Cisco

June 30, 2015

An Autonomic Control Plane
draft-behringer-anima-autonomic-control-plane-03

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. One application is a "virtual out of band channel" for communications over a network that is not configured or mis-configured. This document describes requirements and implementation options for an "Autonomic Control Plane".

Status of This Memo

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

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

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

This Internet-Draft will expire on January 1, 2016.

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 requirements and implementation options of an 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.
- 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.pritikin-anima-bootstrapping-keyinfra]
- Devices can use the ACP for direct decentralised communications, such as negotiations or discovery. The ACP therefore supports directly Autonomic Networking functions, as described in

[I-D.behringer-anima-reference-model]. For example, GDNP [I-D.carpenter-anima-gdn-protocol] can run inside the ACP.

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. 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 process 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. (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 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). ULA seems like a good choice for 1 and 2.

4. The ACP MUST be generic. Usable by all the functions and protocols of the AN infrastructure. 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):

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

- When an autonomic node discovers another autonomic node from the same domain, it authenticates that node and negotiates a secure tunnel to it. 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 loopback interface with a ULA IPv6 address.

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

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

- None of the above operations is reflected in the configuration of the device.

The following figure illustrates the ACP.

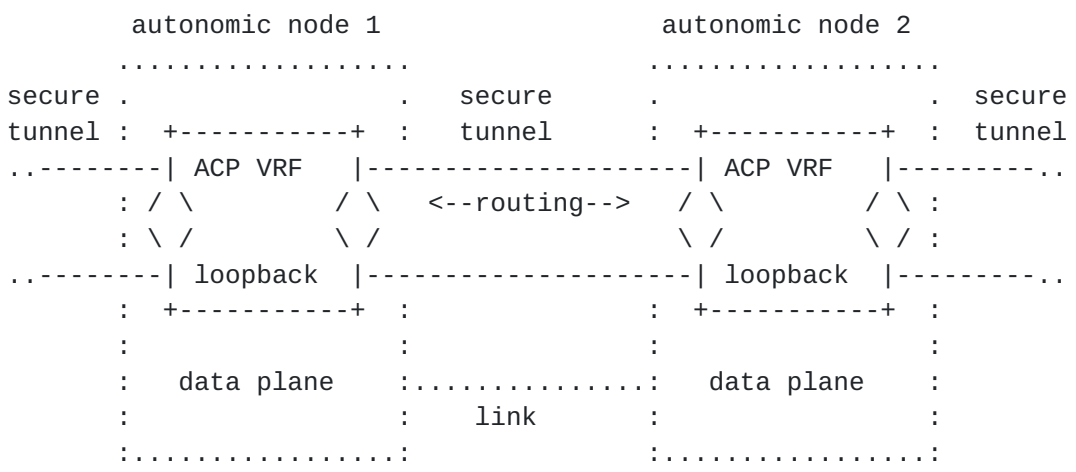

Figure 1

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

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.

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

Each autonomic device has a globally unique domain certificate, with which it can cryptographically assert its membership of the domain. The document [I-D.pritikin-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].)

5.2. Adjacency Discovery

Adjacency discovery exchanges identity information about neighbors, either the UDI or, if present, the domain certificate (see Section 5.1. This document assumes the existence of a domain certificate.

Adjacency discovery provides a table of information of adjacent neighbors. Each neighbor is identified by a globally unique device identifier (UDI).

The adjacency table contains the following information about the adjacent neighbors.

- Globally valid Unique device identifier (UDI).
- Link Local IPv6 address with its scope.
- Trust information: The certificate chain, if available.
- Validity of the trust (once validated, see next section).

Adjacency discovery can populate this table by several means. One such mechanism is to discover using link local multicast probes, which has no dependency on configured addressing and is preferable in an autonomic network.

The "Generic Discovery and Negotiation Protocol" GDNP described in [I-D.carpenter-anima-gdn-protocol] is a possible candidate protocol to meet the requirements for Adjacency Discovery described here.

5.3. Authenticating Neighbors

Each neighbor in the adjacency table is authenticated. The result of the authentication of the neighbor information is stored in the adjacency table. We distinguish the following cases:

- Inside the domain: If the domain certificate presented is validated (including proof of possession of the corresponding private key) to be in the same domain as that of the autonomic entity then the neighbor is deemed to be inside the autonomic domain. Only entities inside the autonomic domain will by default be able to establish the autonomic control plane. Alternatively, policy can define whether to simply trust devices with the same trust anchor. An ACP channel will be established.
- Outside the domain: If there is no domain certificate presented by the neighbor, or if the domain certificate presented is invalid or

expired, then the neighbor is deemed to be outside the autonomic domain. No ACP channel will be established.

Certificate management questions such as enrolment, revocation, renewal, etc, are not discussed in this draft. Please refer to [I-D.pritikin-anima-bootstrapping-keyinfra] for more details.

5.4. Capability Negotiation

Autonomic devices have different capabilities based on the type of device and where it is deployed. To establish a trusted secure communication channel, devices must be able to negotiate with each neighbor a set of parameters for establishing the communication channel, most notably channel type and security type. the communication channel, most notably channel type and security type. The channel type could be any tunnel mechanism that is feasible between two adjacent neighbors, for example a GRE tunnel. The security type could be any of the channel protection mechanism that is available between two adjacent neighbors on a given channel type, for example TLS, DTLS or IPsec. The establishment of the autonomic control plane can happen after the channel type and security type is negotiated.

The "Generic Discovery and Negotiation Protocol GDNP described in [I-D.carpenter-anima-gdn-protocol] is a possible candidate protocol to meet the requirements for capability negotiation described here.

5.5. Channel Establishment

After authentication and capability negotiation autonomic nodes establish a secure channel towards their direct AN neighbors with the above negotiated parameters. In order to be independent of configured link addresses, these channels can be implemented in several ways:

- As a secure IP tunnel (e.g., IPsec, DTLS, TLS, etc.), using IPv6 link local addresses between two adjacent neighbors. This way, the ACP tunnels are independent of correct network wide routing. They also do not require larger than link local scope addresses, which would normally need to be configured or maintained. Each AN node MUST support this function.

- L2 separation, for example via a separate 802.1q tag for ACP traffic. This even further reduces dependency against the data plane (not even IPv6 link-local there required), but may be harder to implement.

Since channels are 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 5.8.

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.

5.6. 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 and - as necessitated by the implementation option be protected from being modified unintential from data plane configuration.

In addition this provides for 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.

5.7. Addressing inside the ACP

The channels explained above only establish communication between two adjacent neighbors. In order for the communication to happen across multiple hops, the autonomic control plane requires internal network wide valid addresses and routing. Each autonomic node must create a loopback interface with a network wide unique address inside the ACP context mentioned in Section 5.6.

We suggest to create network wide Unique Local Addresses (ULA) in accordance with [RFC4193] with the following algorithm:

- Prefix FC01::/8

- Global ID: a hash of the domain ID; this way all devices in the same domain have the same /48 prefix. Conversely, global ID 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 6 for a discussion on merging domains.

- Subnet ID and interface ID: These can be either derived deterministically from the name of the device, or assigned at registration time of the device.

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

5.8. 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 as one such protocol which is light weight and scales well for the control plane traffic.

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

The preferred way for an NMS host to connect to the ACP of a network is to enrol that NMS host as a domain device, such that it shares a domain certificate with the same trust anchor as the network devices.

Then, the NMS host can automatically discover an adjacent network element, and join the ACP automatically, just like a network device would connect to a neighboring device. Alternatively, if there is no directly connected autonomic network element, a secure connection to a single remote network element can be established by configuration, authenticated using the domain certificates. There, the NMS host "enters" the ACP, from which point it can use the ACP to reach further nodes.

If the NMS host does not support autonomic negotiation of the ACP, then it can be brought into the ACP by 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.

In both options, the NMS host must be routed in the ACP. This involves two parts: 1) the NMS host must point default to the AN device for all IPv6, or 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. 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 5.7).

7. 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.

8. The Administrator View

An ACP is self-forming, self-managing and self-protecting, therefore has minimal dependencies on the administrator of the network. Specifically, it cannot be configured, there is therefore 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 5.9 for more details on how to connect an NMS host into the ACP.

9. 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.

10. IANA Considerations

This document requests no action by IANA.

11. 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, Toerless Eckert, Yves Hertoghs, Bruno Klauser, Max Pritikin, Ravi Kumar Vadapalli.

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

12. Change log [RFC Editor: Please remove]

12.1. Initial version

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

12.2. version 00

Initial version of the anima document; only minor edits.

12.3. version 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.

12.4. version 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

12.5. version 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.

13. References

[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.carpenter-anima-gdn-protocol]
Carpenter, B. and B. Liu, "A Generic Discovery and Negotiation Protocol for Autonomic Networking", draft-carpenter-anima-gdn-protocol-04 (work in progress), June 2015.

[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.pritikin-anima-bootstrapping-keyinfra]
Pritikin, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-pritikin-anima-bootstrapping-keyinfra-01 (work in progress), February 2015.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, June 2015.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, June 2015.

Authors' Addresses

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

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com

Balaji BL
Cisco

Email: blbalaji@cisco.com

Toerless Eckert
Cisco

Email: eckert@cisco.com