Autonomic Networking Gets Serious

By the ANIMA author team

Introduction

In May 2021, six RFCs about autonomic networking were published[5,6,7,8,9,10]. What's this all about? One way to sum it up is "plug and play" for the network. Sometimes, that means "plug and play for the ISP" or "plug and play for the enterprise." This is a step forward from the well known idea of plug and play for home networks.

The term "autonomic computing" was coined as much as 20 years ago by IBM. It led naturally to the idea of autonomic networking, which became a topic of discussion in the IRTF Network Management Research Group some years ago, and has led to both research projects and proprietary implementations. [Short description of some proprietary solutions here?] But as always, the need is for interoperability, so proprietary methods have to give way to industry standards. This has been the topic of the IETF's Autonomic Networking Integrated Model and Approach (ANIMA) working group since late 2014.

The goal is self-management of networks, including self-configuration, self-optimization, self-healing and self-protection. Autonomic Networking (AN) puts operational intelligence into algorithms at the node level, to minimize dependency on human administrators and central management. Nodes capable of AN will discover information about the surrounding network and negotiate parameter settings with their neighbors and other nodes. Later, nodes may also have learning and cognitive capability, i.e. the ability to self-adapt their decision-making process based on information and knowledge sensed from their environment.

Science fiction? Not really. We've had routing protocols that meet the definition of "autonomic" for many years. The new idea is that anything that today requires top-down configuration could be configured autonomically by a discovery and negotiation process, governed by some general rules referred to as policy. Why now? Partly because large operators are suffering more and more from the problems and difficulties caused by central configuration of hundreds or thousands of network elements. Partly because after some years of discussion, ideas about how to achieve autonomic networking are becoming concrete. And partly because it is now economic to provide enough computing power in network elements to support AN.

Of course it's fundamental that AN techniques must co-exist with and interoperate with standard network management tools and methods; they must be "NOC-friendly," i.e., fit seamlessly with existing Network Operations Centers. Also, various aspects of security are vital for AN. Exactly as with self-driving cars, a self-managing network needs to be significantly more secure than a manually configured network, or no enterprise will accept it. Indeed, getting security right has been a major part of the ANIMA working group's job.

At the same time, continued rapid growth of size and complexity of home networks is expected. It goes without saying that they need to be completely self-managing, but the role of AN techniques in

home networks remains to be explored and is not discussed in this article. More work is also needed on how AN applies to the Internet of Things.

Terminology

According to various dictionaries, there are differences between the terms *automatic*, *autonomous* and *autonomic*.

Automatic: as if done by a machine.

Autonomous: responding and reacting on its own, with no external control.

Autonomic: behaving spontaneously due to internal stimuli.

The last two are certainly similar, but following industry practice we prefer *autonomic*. The *autonomic nervous system* acts largely unconsciously and regulates bodily functions such as heart rate. *Autonomic computing* was defined by IBM in 2001 as referring to "self-managing distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity from operators and users." We define an *autonomic network* as self-managing (self-configuring, selfprotecting, self-healing, self-optimizing) but allowing high-level guidance by a central entity.

Autonomic Function: A specific self-managing feature or function.

Autonomic Service Agent (ASA): An agent that implements an autonomic function, in part (for a distributed function) or whole.

Autonomic Node: A node that embodies autonomic functions

Autonomic Control Plane (ACP): A self-configuring, fully secure, virtual network used for all autonomic messaging.

More details about these terms can be found in RFC7575[1] and RFC8993[8].

Outline of the ANIMA model

As always in network management, there are literally thousands or millions of details that cannot be standardized or even described centrally. What we can do is define a model, a platform, and a toolkit, just as SNMP (Simple Network Management Protocol) and NETCONF (Network Configuration Protocol) have done in the past. The main items in the model are:

- Bootstrapping and trust infrastructure. This covers how nodes are authenticated and securely admitted to an autonomic network, and how they establish mutual trust.
- Secure Autonomic Control Plane (ACP). This is an automatically constructed encrypted virtual network, containing only authenticated nodes that rightfully belong to a particular autonomic domain.
- Discovery for autonomic nodes. This is a mechanism by which nodes attached to the ACP can discover each other. In practice, discovery occurs at a finer grain than nodes, since it really operates at the level of a node's capabilities and objectives.

- Negotiation and synchronization for autonomic nodes. Once nodes have discovered each other, they can synchronize data between themselves, or actively negotiate parameters and resources.
- Autonomic functions operate by negotiating and synchronizing data with their peers in other nodes, and by directly configuring manageable devices in their own scope.
- Discovery, synchronization and negotiation proceed by use of the GeneRic Autonomic Signaling Protocol (GRASP).
- Autonomic service agents (ASAs) are composed of one or more autonomic functions.
- Centrally defined policy or configuration rules may be obtained by an ASA via GRASP synchronization, or if appropriate by conventional methods such as an interface to NETCONF or DNS-SD (DNS Service Discovery).

Figure 1 shows an outline of the model as a whole.

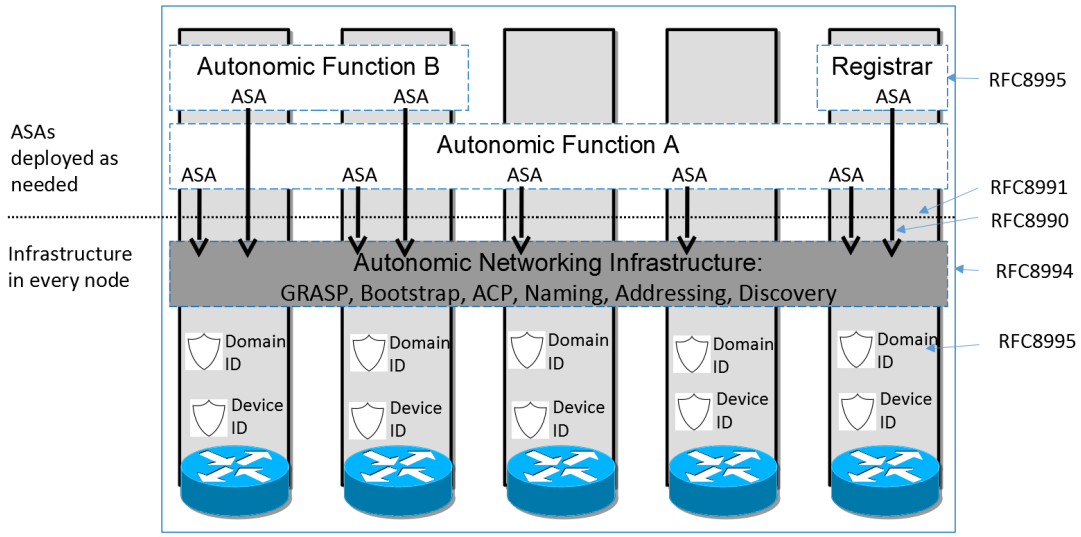

Fig. 1 Network with autonomic functions (RFC8993)

Security

ANIMA does not attempt a monolithic bootstrap of a network from a predefined configuration. Instead, it proceeds step by step, and security comes first. The first stage of creating a secure autonomic control plane is bootstrapping a suitable key infrastructure that covers all the nodes that will constitute the ACP. This is done by a method known as BRSKI (pronounced “Brewski”, Bootstrapping Remote Secure Key Infrastructure[10]). This process uses manufacturer-installed X.509 certificates, in combination with a manufacturer's authorizing service. The network administrator decides which devices are authorized to join the network (e.g., by serial number), but relies on the manufacturer to validate each device’s certificate whenever the device attempts to join the network via a local “join proxy”. These proxies all use a single “domain registrar” node that mediates the authorizing service.

The join proxies themselves join the network by the same process; a GRASP mechanism is used for joining nodes (known as “pledges”) to find proxies, and for proxies to find each other and the registrar. Only the registrar needs to be configured in advance.

The ACP forms itself among pledges as soon as they have completed their BRSKI enrolment. It is best described as a Virtual Routing and Forwarding (VRF) instance. It is based on a virtual router at each node, consisting of a separate IPv6 forwarding table to which the ACP’s virtual interfaces are attached, and an associated IPv6 routing table separate from the data plane. Actual packet transmission occurs only as IPv6 link-local packets. This choice was made to ensure that there is no dependency on any pre-existing data plane (either IPv4 or IPv6), because autonomic functions must be able to operate *even if the normal data plane and normal routing are broken.* All that is required is for each node to create its own IPv6 link-local address on each physical interface, as any modern network device does by default. The VRF consists of point-to-point IPv6 links and is secured using IPsec (IP Security) or DTLS (Datagram Transport Layer Security), both via IKEv2 (Internet Key Exchange Protocol Version 2). From the viewpoint of autonomic service agents, the ACP uses an automatically generated IPv6 Unique Local Address prefix, and it uses RPL (Routing Protocol for Low-Power and Lossy Networks) internally. Like BRSKI, the ACP bootstraps itself, starting with a GRASP-based discovery process.

After the secure control plane has configured itself in this way, the next stage is to bootstrap connectivity for network management. When this has been achieved, conventional mechanisms (such as an SDN controller) can already reliably and securely reach remote nodes and configure them safely without risk of cutting themselves off. In addition, fully autonomic management mechanisms (i.e., ASAs) can start up. To understand how this works, we need to give more details about the GRASP protocol.

GRASP

GRASP, the GeneRic Autonomic Signaling Protocol[5], is used for signaling between ASAs. These include special-purpose mini-ASAs that support BRSKI (discovery of join proxies and the domain registrar) and ACP creation (discovery of ACP neighbors). Readers will notice that these operations must take place *before* ACP security is in place, so they use a highly restricted subset of GRASP that is limited to specific link-local operations.

After that, GRASP runs over the ACP to guarantee security, so there are no restrictions on allowed operations and any two ASAs in the local domain may trust and communicate with each other. GRASP provides discovery, flooding, synchronization and negotiation mechanisms for the objectives supported by ASAs.

Rather than being a traditional type-length-value protocol, GRASP is based on CBOR (Concise Binary Object Representation) messages. This has the advantage of allowing very flexible encoding, and GRASP can therefore accommodate a very wide range of data types, with the possibility of mapping protocol elements directly into various high-level language representations.

The word “objective” has a special meaning in GRASP. It is a data structure whose main contents are a *name* and a *value*. An objective occurs in three contexts: discovery, negotiation, and synchronization. A single ASA may support multiple independent objectives.

The *name* of an objective is simply a unique string describing its purpose.

The *value* consists of a single configurable parameter or a set of parameters of some kind. The parameter(s) apply to a specific service or function or action. They may in principle be anything that can be set to a specific logical, numerical, or string value, or a more complex data structure. Basically, an objective is defined in the way that best suits its application; that is the great advantage of CBOR encoding. If desired, for example, an objective's *value* could be expressed in JSON (JavaScript Object Notation). When an objective is shared between ASAs by flooding, synchronization or negotiation, each ASA will maintain its own copy of the objective and its latest value.

GRASP messages allow for *discovery* of an ASA that handles a given objective name; *flooding* a given objective to all ACP nodes (the simplest form of synchronization); *synchronization* of the value of a given objective between two peer ASAs; and *negotiation* of the value of a given objective with a peer ASA.

An Application Programming Interface (API) for GRASP has been defined[6] and implemented as part of a Python 3 prototype. This makes it very easy to implement demonstration ASAs in Python. A partial GRASP implementation has also been made as part of an ACP implementation in the RUST language.

Talking to the NOC

As noted above, a key requirement for the success of ANIMA is smooth integration with existing network management tools and in particular with Network Operations Centers. To this end, an integration mechanism has been documented[4]. The simplest approach is for trusted edge devices in the ACP to "leak" the (otherwise encrypted) ACP natively to certain network management hosts, presumed to be well secured. These edge devices would act as default routers to those management hosts and provide them with IPv6 connectivity into the ACP. A more complex approach would allow the management hosts simultaneous connectivity into the ACP and the traditional data plane.

A related issue is that if the NOC uses DNS Service Discovery (DNS-SD) to announce management services to managed nodes, these announcements will not be automatically available in the ACP, which for security reasons will not have routed access to the data plane where the DNS is available. This again can be solved by a trusted edge device that obtains service information from DNS-SD and redistributes it within the ACP, possibly by the GRASP flooding mechanism. For example, the information for a service named *syslog* could be flooded in a GRASP objective named *SRV.syslog*. Here, the flexibility of CBOR encoding is be of great value since a JSON-like representation of service data is common.

Extending that point, since GRASP easily allows for JSON (or practically any other format), it is possible to integrate ASAs communicating via GRASP into almost any part of an existing network management system. For example, an ASA acting as a NETCONF client could retrieve YANG documents from a NOC database via GRASP and the ACP.

Examples

A use case that has been fully defined is a GRASP-based mechanism for managing and assigning IP address prefixes[7]. Firstly, we define two GRASP *objectives* for IPv4 or IPv6 prefix management at the edge of large-scale ISP networks. The first objective can be represented thus (in a simplified form):

```
["PrefixManager", [IP_version, prefix_length, prefix]]
```

and the second as

```
["PrefixManager.Params", parameter_info]
```
.

The first objective will be used in GRASP negotiations between two “prefix manager” ASAs in nodes that need to delegate address space to subsidiary routers (using standard IPv6 prefix delegation), when one node is short of spare prefixes and the other one has an adequate pool of unused prefixes. If negotiation succeeds, prefixes will be transferred from one ASA’s pool to the other’s. If negotiation fails, the ASA that is short of prefixes will use GRASP discovery to find another ASA that can help it. This will completely obviate any need for human management of an ISP’s distributed pool of prefixes, beyond initially configuring the maximum pool in one place.

The second objective may be flooded to all “prefix manager” ASAs to convey relevant policy. For example, if the flooded parameter information is

```
[
   [["role", "A"],["prefix_length", 34]],
   [["role", "B"],["prefix_length", 44]],
   [["role", "C"],["prefix_length", 56]]
]
```

it would mean that devices of type A are allowed to receive IPv6 prefixes of length 34 bits, and so on.

We can see from this example that GRASP’s use of CBOR and its easy representation of JSON-like formats gives it great expressiveness and flexibility.

Another example?

Conclusion

TBD

Rererences and Further Reading

[1] RFC7575, Autonomic Networking: Definitions and Design Goals. M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, L. Ciavaglia. June 2015. (DOI: 10.17487/RFC7575)

[2] RFC7576, General Gap Analysis for Autonomic Networking. S. Jiang, B. Carpenter, M. Behringer. June 2015. (DOI: 10.17487/RFC7576)

[3] RFC8366, A Voucher Artifact for Bootstrapping Protocols. K. Watsen, M. Richardson, M. Pritikin, T. Eckert. May 2018. (DOI: 10.17487/RFC8366)

[4] RFC8368, Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM). T. Eckert, Ed., M. Behringer. May 2018. (DOI: 10.17487/RFC8368)

[5] RFC8990, GeneRic Autonomic Signaling Protocol (GRASP). C. Bormann, B. Carpenter, Ed., B. Liu, Ed. May 2021. (DOI: 10.17487/RFC8990)

[6] RFC8991, GeneRic Autonomic Signaling Protocol Application Program Interface (GRASP API). B. Carpenter, B. Liu, Ed., W. Wang, X. Gong. May 2021. (DOI: 10.17487/RFC8991)

[7] RFC8992, Autonomic IPv6 Edge Prefix Management in Large-Scale Networks. S. Jiang, Ed., Z. Du, B. Carpenter, Q. Sun. May 2021. (DOI: 10.17487/RFC8992)

[8] RFC8993, A Reference Model for Autonomic Networking. M. Behringer, Ed., B. Carpenter, T. Eckert, L. Ciavaglia, J. Nobre. May 2021. (DOI: 10.17487/RFC8993)

[9] RFC8994, An Autonomic Control Plane (ACP). T. Eckert, Ed., M. Behringer, Ed., S. Bjarnason. May 2021. (DOI: 10.17487/RFC8994)

[10] RFC8995, Bootstrapping Remote Secure Key Infrastructure (BRSKI). M. Pritikin, M. Richardson, T. Eckert, M. Behringer, K. Watsen. May 2021. (DOI: 10.17487/RFC8995)

THE ANIMA AUTHOR TEAM is a group of participants in the IETF's ANIMA Working Group, including Michael Behringer, Brian E. Carpenter, Toerless Eckert, Sheng Jiang, Yizhou Li, Michael Richardson, YOUR NAME HERE. They may be contacted at anima@ietf.org.