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], result of work of the “Autonomic Networking Integrated Model and Approach” (ANIMA) WG of the IETF. These RFCs complete the first charter round of that working group, which was started in late 2014 (see [11] for a summary of the inception back then). This work now allows to build IETF standardized network solutions for an “Autonomous Networking Infrastructure” (ANI) into every network device.

What is this all about? One way to sum it up is “plug and play” for the network. This can mean “plug and play for the ISP” or “… for the enterprise.” or “… for the (industrial) networks”. This is a significant step forward from the well known idea of plug and play for home networks which the IETF addresses in the HOMENET WG.

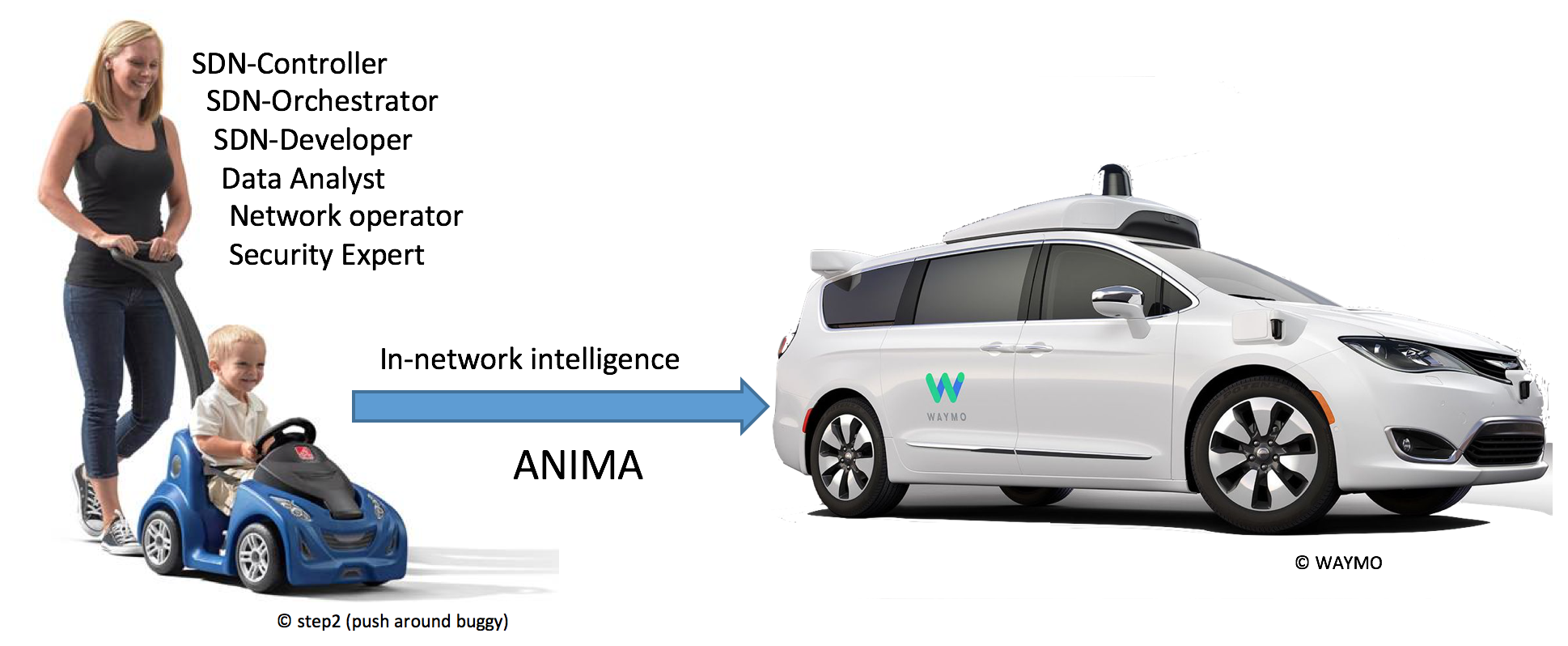
The term “autonomic computing” was coined as early as 20 years ago by IBM. It led naturally to the idea of autonomic networking, which became a topic of discussion and work in the IRTF Network Management Research Group that resulted in RFCs [12], [13] describing the outline of an envisioned ANI and ultimately in the creation of the ANIMA WG. Since then, various aspects of the problem space where addressed in research, and vendor 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 is the ongoing topic of the ANIMA working.

The goal is self-management of networks, including self-configuration, self-optimization, self-healing and self-protection (also called self-X). 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. Distributed routing protocols as introduced with the ARPANET in the 1970’th and later the Internet (OSPF, ISIS for example) are at their core autonomous: self-configuring, self-optimizing, self-healing. But over the decades since that original idea and reality, even those protocols evolved to become provisioning monsters of nerd-knob parameters for operators. A whole industry and research faculty for Network Operations Administration and Management – OAM) evolved to define architectures consisting of an ever more complex multitude of layers between the actual Intent for the Service Level Objectives of the network (and by implication its protocols) and all those magic parameters that need to be provisioned consistently and dynamically into each network device whenever there is any change.

In today’s networks, these layers are almost exclusively implemented through a highly complex and most often centralized set of “Software Defined Networking” (SDN) Controller and Orchestrator software and human operators. These solutions are difficult and expensive to build, maintain, validate, predict, secure and foremost to make reliable and resilient. These problems are rarely seen from the outside, but only when network services are under oversight of regulatory entities that publish reports of those problems, such as [14]. SDN architectures are also highly proprietary, very often single-vendor centric and typically require for every multi-vendor network deployment significant customization through programming and therefore require network owners to not only staff network operators but have them become SDN developers.

Nevertheless, these SDN methods are in the face of existing networks the best option. They are marketed with terms that evolved in the last few years, such as “Zero Touch Networks”, “Intent Based Networking”, or “Self Driving Networks”. In the metaphor of a networking being a car, the following picture shows how much actually self-driving todays networks are, and how much ANIMA would like them to be.



Nevertheless, the vision for ANIMA is not the same as its achieved and ongoing standardization goals. Much like the near term focus for most cars are better and better driver-assist systems, the ANI as defined in the just released ANIMA RFCs is intended to provide the most foundational assist building blocks. These building blocks are meant to fit seamlessly with existing network and SDN/OAM designs and to improve their metrics such as simplicity, reliability and security. Likewise, the ANI allows to more easily embed more automation into network devices whenever there is a need.

~~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.~~

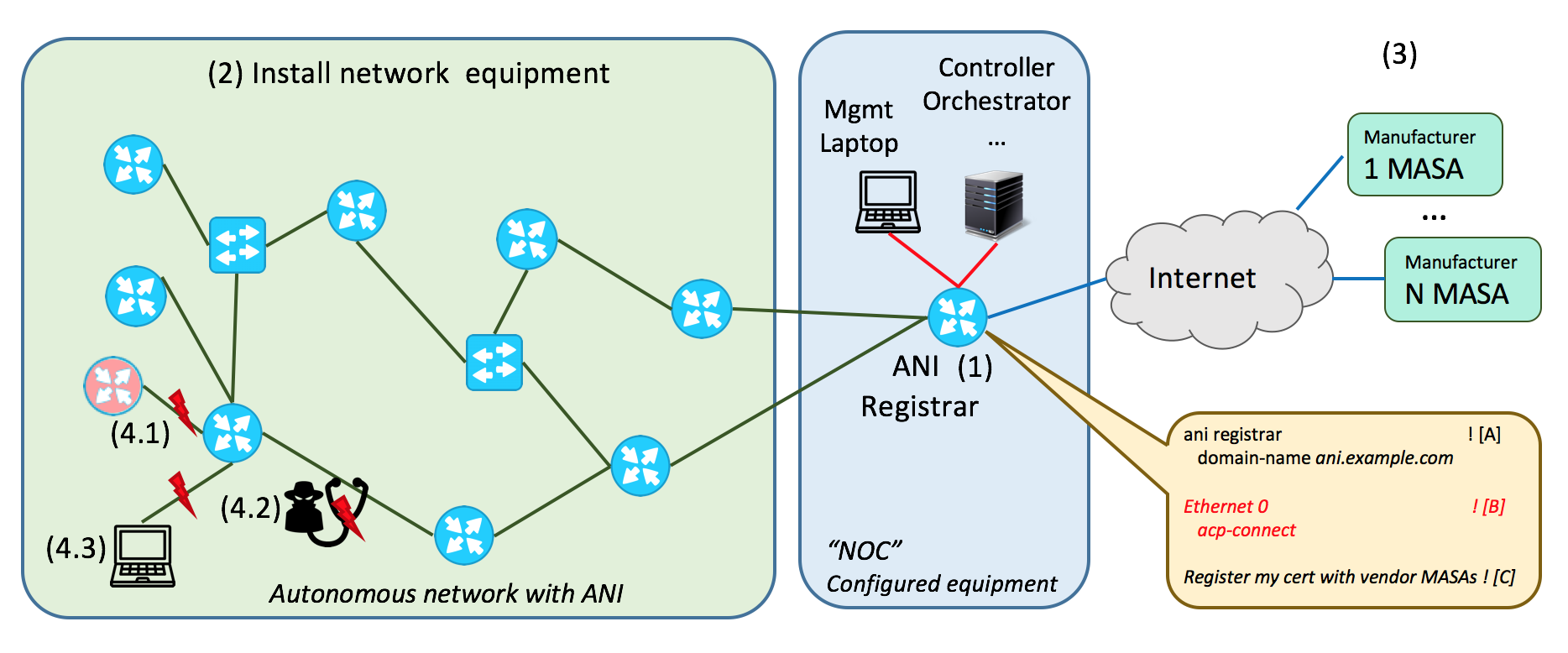
~~Of course it’s fundamental that AN techniques must co-exist with and interoperate with standard network management tools and methods; they must be “NOC-friendly,” i.e., fit seamlessly with existing Network Operations Centers. Also, various aspects of security are vital for AN. Exactly as with self-driving 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.

## What can the ANI do for you ?

Instead of jumping directly into explanation how the ANI works, let’s first give a simple example of what the operator experience of a typical, simple ANI network could be.

In the following picture, an operator wants to have a new network of devices (routers and switches) deployed. The actual reception of the new, factory fresh equipment, unpacking and physical attachment is performed in the different locations by other personnel. The operator herself only needs to set up an ANI seed router, called the ANI registrar (1), for example in a NOC. This setup consists of only three simple steps: [A] Making the router be the registrar and assigning a name to the ANI, [B] configuring some local port(s) to provide access to the ANI to connect management equipment such as a Notebook for manual access or an SDN controller, and finally registering the certificate of the registrar with the so-called MASA services of the vendors whose routers and switches are being used in the new network (we will see in a moment what that does).



As long as his seed setup is not in place, new ANI routers/switches may be physically interconnected, but they won’t do anything. Once they have connectivity to a configured registrar, they will automatically form an ANI with high level the following step:

The new ANI device (called pledge) will automatically create a connection with the ANI registrar and attempt to get enrolled with an ANI certificate by that ANI so it can participate in the ABI. But the registrar needs to prove that it ‘owns’ the ANI device. To do that, the registrar communicates (for example over the Internet) with the MASA of the vendor of that device. That MASA has the information that this pledge is actually owned by this registrars network and provides in return a voucher back to the pledge which will now make the pledge trust the registrar. It will therefore accept an ANI certificate from the registrar. This process runs completely automated without any further handholding or configuration. It is the Bootstrap of Remote Key Infrastructures (BRSKI) part of ANI.

Once a new devices is enrolled with an ANI certificate it begins to establish an Autonomic Control Plane (ACP) connection with all its neighbors, authenticated and authorized mutually by the devices ANI certificates. This too happens without any further handholding or configuration.

Assume for example all devices where connected to each other as shown in the picture and the ANI registrar is connected last (after it was configured). Within minutes, all the devices will have run through BSKI, and set up the ACP. In result, the network operator now has from her management laptop and SDN controller IPv6 connectivity to all ANI devices and can configure them manually or through SDN automation using this ACP IPv6 connectivity. Each ANI device has a permanent, so-called ULA IPv6 address within the ANI that does not change, even when the device is physically moved in the network to a different point of attachment.

But wait! How is this at all different from 30 year old ethernet technology ? One can simply buy a set of inexpensive ethernet switches, interconnect them, attach a configuration system at one point and have achieved the same thing. No ?

Indeed, the simplicity of operating ethernet networks was inspiration and goal for the ANI, but beyond that, the ANI is fundamentally different.

The ANI is foremost secure, whereas the avode default behavior of ethernet switches is not. An ANI device can only become part of the ANI when it is actually owned by the operator as known by the device manufacturers MASA, for example via sales records or a variety of other options. This means that a stolen device cannot be activated for the ANI in another network. It also means that a device not belonging to this network operator (4.1) cannot be physically attached to an ANI network in the hope to become part of the ANI to then attack it.

All ACP traffic is hop-by-encrypted, therefore also all management traffic that uses the ACP including any legacy, not end-to-end encrypted management protocol cannot be snooped or spoofed by an attacker (4.2).

Last but not least, ANI devices even after having formed the ACP are still unconfigured, and ideally this means that they should behave like current unconfigured routers: There is nothing running that would provide likely undesirable network connectivity to any hosts that attach, like some user or attacker notebook (4.3). Such an attached device would get no connectivity whatsoever. In result, there is never a window of opportunity for attackers to attack unprotected equipment. Instead, network operations has all the time it wants to remotely provision all the desired configuration on the devices.

Compared to many other zero-touch solutions, the ANI does not only focus on so-called day-0/day-1 behavior up until the network is operational. Instead its services are through the whole lifecycle. The ANI provides automated certificate renewal for all ANI devices to maintain and refresh its security model. The ACP protects any ongoing network OAM traffic that uses it. By its use of hop-by-hop encryption it also continuously protects the whole network and attached OAM equipment from traffic injection/spoofing attacks.

But how about actual autonomous networking ?

… I will stop for today. What is missing from above description is any mentioning of GRASP and how it is not only used by ANI itself for BRSKI and ACP, but also how it is the core element of ANI for steps towards autonomous networks. I think I would be best to have similar text as above, but now with the example of building simple ASA that leverage GRASP to autoconfigure services. Such as Michaels beloeved and well proven concept of ASA used to autoconfigure any of the 10 different existing protocol securities (ASA to auto-secure ISIS / OSPF, BGP, PIM ….)

## 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].

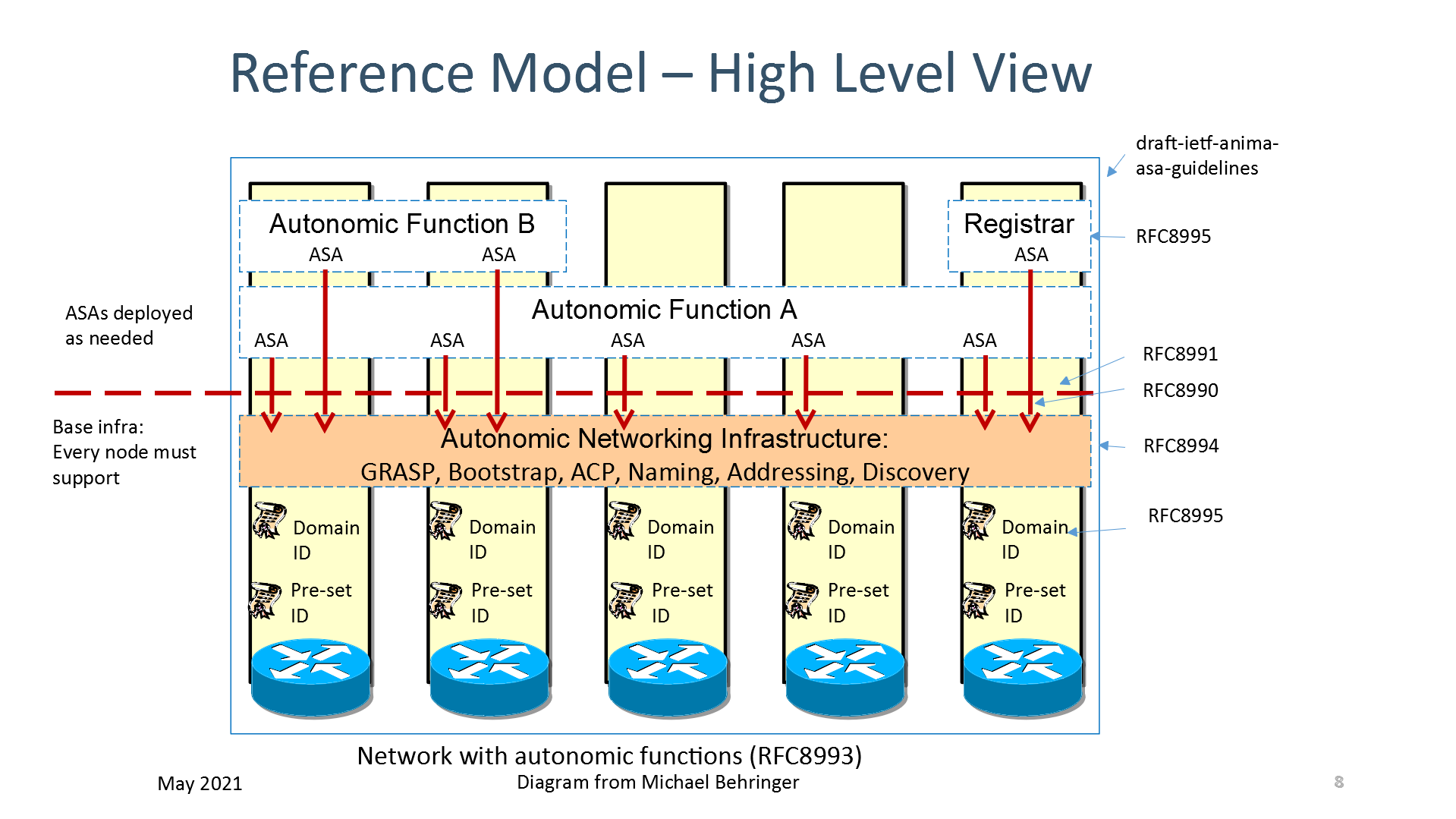
## Outline of the ANIMA model

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.

[Old version of figure follows - needs simplifying for IPJ guidelines]



## 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

Prefix management

Another?

## 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)

[11] <https://www.ietfjournal.org/autonomic-networking/>

[12] RFC7475

[13] RFC7476

[14] FCC, "June 15, 2020 T-Mobile Network Outage Report", A Report of the Public Safety and Homeland Security Bureau Federal Communications Commission, PS Docket No. 20-183, October 2020, <<https://docs.fcc.gov/public/attachments/DOC-367699A1.docx>>.

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.