GeneRic Autonomic Signaling Protocol (GRASP)

An Introduction

Brian Carpenter

October 2021

RFC8990
RFC8991

Note

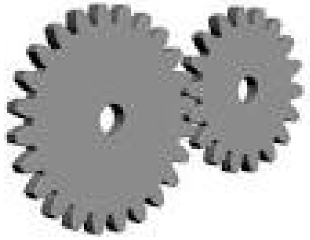
- For general background on the IETF approach to autonomic networking, see the following article:

  Autonomic Networking Gets Serious, Internet Protocol Journal 24(3), pp2-18, October 2021.

- For a detailed tutorial on GRASP, see https://github.com/becarpenter/graspdoc/raw/main/GRASP-tutorial.pdf

English lesson

- Automatic
  - done as if by machine; self-acting or self-regulating mechanism



- Autonomous
  - without outside control; responding, reacting, or developing independently of the whole

- Autonomic
  - occurring involuntarily or spontaneously; occurring as a result of internal stimuli

Terminology (1)

- Autonomic Nervous System: a control system that acts largely unconsciously and regulates bodily functions such as heart rate.
- Autonomic Computing: self-managing distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity from operators and users (IBM, 2001).
- Autonomic Network: Self-managing (self-configuring, self-protecting, self-healing, self-optimizing) but allowing high-level guidance by a central entity ("Intent")
  - "Plug and play for the ISP" or "plug and play for the enterprise"

Terminology (2)

- 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 employs autonomic functions
- Autonomic Control Plane (ACP): Self-configuring fully secure virtual network used for all autonomic messaging.

For more details see RFC7575 and RFC8993

Autonomic Networking Integrated Model and Approach (ANIMA) WG

- Initial work items (RFCs imminent)
  - Bootstrapping & trust infrastructure
  - Secure Autonomic Control Plane (ACP)
  - Discovery for autonomic nodes
  - Negotiation & synchronization for autonomic nodes

- Next steps
  - Intent (high level policy)
  - Defining the domain boundary
  - ASA life cycle, authorization and coordination
  - Reporting

- Left for much later
  - Tie in to machine learning and other AI techniques

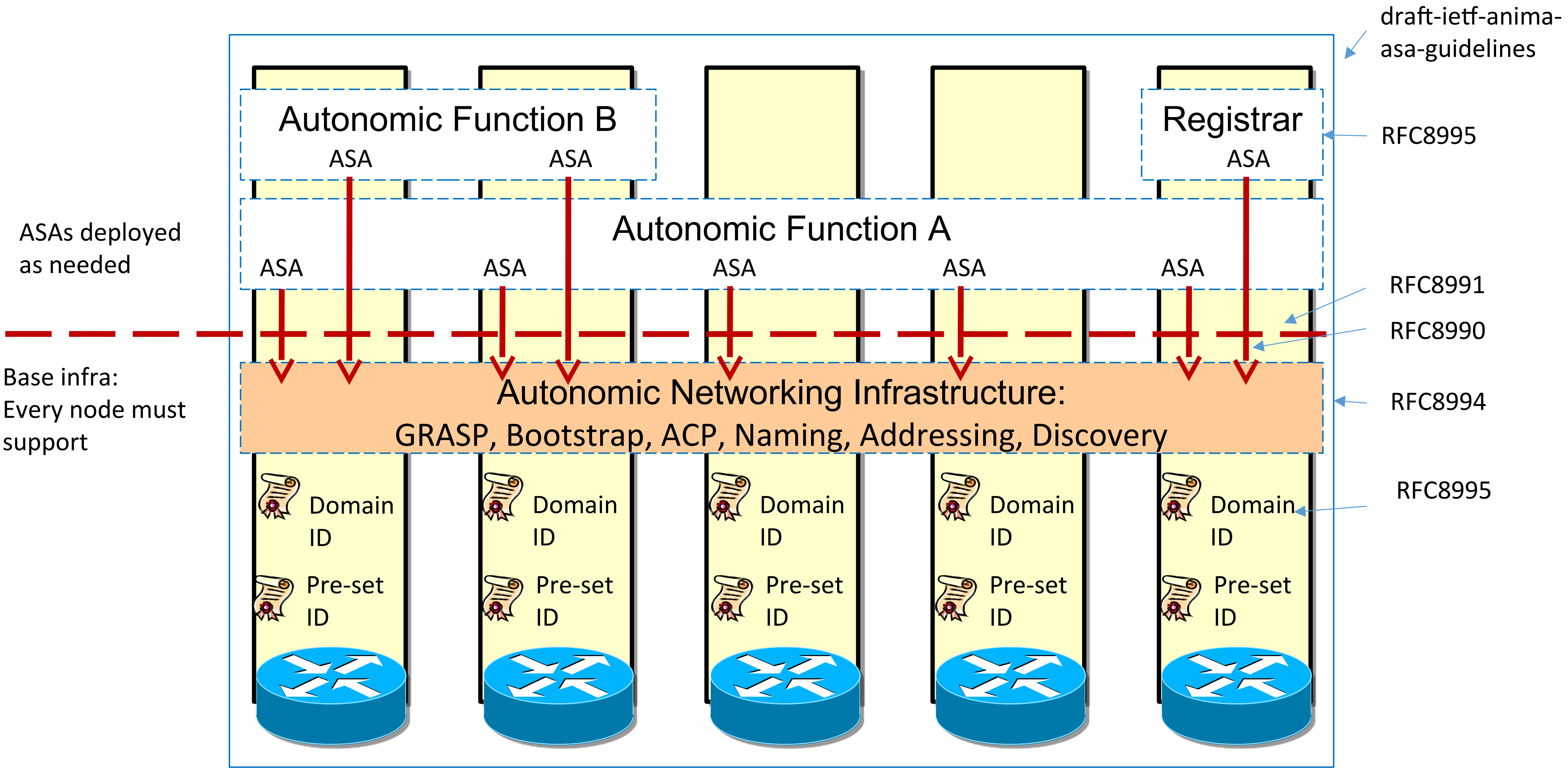
Bootstrap and ACP

- Secure bootstrap – all nodes must start (out of the box or after a factory reset) by using a registrar to authenticate themselves and obtain a domain certificate. This is coordinated with related work for IoT devices. No human intervention except to create the registrar.
- ACP – the ACP will bootstrap itself using only link-local IPv6 addresses and IPv6 Unique Local Address prefix. All links secured (IPsec). No human intervention except to define the domain boundary.

GeneRic Autonomic Signaling Protocol (GRASP)

- GRASP will be used for signaling between ASAs
  - That includes the special-purpose ASAs that support both secure bootstrap & ACP creation
  - After that, GRASP runs over the ACP to guarantee security
- GRASP provides discovery, flooding, synchronization and negotiation for the <u>technical objectives</u> supported by ASAs
  - Based on CBOR (Concise Binary Object Representation)
  - Objectives can be expressed in JSON or Python-like syntax & semantics

Reference Model – High Level View



Network with autonomic functions (RFC8993)
Diagram from Michael Behringer

More about a GRASP Objective

- A configurable parameter:
  - a logical, numerical or string value, or a more complex data structure.
  - used in Discovery, Negotiation, Flooding and Synchronization.
  - semantics depend on the autonomic function concerned, and are built into the code of each ASA.
- Example for IP prefix management:

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

GRASP Messages

• Discovery (multicast)	M_DISCOVERY
Discovery Response	M_RESPONSE
• Request Negotiation	M_REQ_NEG
Negotiation	M_NEGOTIATE
Confirm Waiting	M_WAIT
Negotiation End	M_END
• Request Synchronization	M_REQ_SYN
Synchronization	M_SYNCH
• Flood Synchronization (multicast)	M_FLOOD

GRASP API Functions

- *Registration*. An ASA can register itself and register the GRASP Objectives it manipulates.
- *Discovery*. An ASA can discover a peer willing to respond for a particular objective.
- *Negotiation*. An ASA can act as an initiator (requester) or responder (listener) for a negotiation session. Negotiation is a symmetric process, so most functions can be used by either party.
- *Synchronization*. An ASA can act as an initiator (requester) or responder (listener and data source) for data synchronization.
- *Flooding*. An ASA can send and receive a GRASP Objective that is flooded to all nodes of the ACP.

A negotiation session

```
Initiator                          Responder

                                   listen_negotiate() \ Await request

request_negotiate()
          M_REQ_NEG      ->         negotiate_step()   \ Open session,
                         <-        M_NEGOTIATE         / start negotiation
negotiate_step()
        M_NEGOTIATE      ->         negotiate_step()   \ Continue
                         <-        M_NEGOTIATE         / negotiation
                         ...
negotiate_step()
        M_NEGOTIATE      ->         negotiate_step()   \ Continue
                         <-        M_NEGOTIATE         / negotiation
negotiate_step()
        M_NEGOTIATE      ->         end_negotiate()    \ End
                         <-        M_END               / negotiation,
                                                       \ process results
```

GRASP Prototype

- A Python 3 implementation of GRASP as a module **`grasp.py`**
- About 2400 lines of code
- A test suite to exercise as many code paths as possible
- Various toy ASAs to test "real" operation across the network
  - bank/client negotiation
  - model of secure bootstrap process
  - model of IPv6 prefix management
  - bulk transfer using GRASP
- Some documentation

More…

- RFC 7575 & 7576 (IRTF background documents))
- RFC 8990 – 8995 (IETF specifications)
- https://datatracker.ietf.org/wg/anima/documents/
- https://github.com/becarpenter/graspy
    - doc at https://github.com/becarpenter/graspy/blob/master/graspy.pdf