Network Working Group   C. Bormann
Internet-Draft   Universitaet Bremen TZI
Intended status: Standards Track   B. Carpenter, Ed.
Expires: June 18, 2017   Univ. of Auckland
                                                           B. Liu, Ed.
                                             Huawei Technologies Co., Ltd
                                                       December 15, 2016

A Generic Autonomic Signaling Protocol (GRASP)
draft-ietf-anima-grasp-09

Abstract

   This document establishes requirements for a signaling protocol that
   enables autonomic devices and autonomic service agents to dynamically
   discover peers, to synchronize state with them, and to negotiate
   parameter settings mutually with them. The document then defines a
   general protocol for discovery, synchronization and negotiation,
   while the technical objectives for specific scenarios are to be
   described in separate documents. An Appendix briefly discusses
   existing protocols with comparable features.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   The success of the Internet has made IP-based networks bigger and
   more complicated. Large-scale ISP and enterprise networks have
   become more and more problematic for human based management. Also,
   operational costs are growing quickly. Consequently, there are
   increased requirements for autonomic behavior in the networks.
   General aspects of autonomic networks are discussed in [RFC7575] and
   [RFC7576].

   One approach is to largely decentralize the logic of network
   management by migrating it into network elements. A reference model
   for autonomic networking on this basis is given in
   [I-D.ietf-anima-reference-model]. The reader should consult this
   document to understand how various autonomic components fit together.
   In order to fulfil autonomy, devices that embody Autonomic Service
   Agents (ASAs, [RFC7575]) have specific signaling requirements. In
   particular they need to discover each other, to synchronize state

with each other, and to negotiate parameters and resources directly
with each other. There is no limitation on the types of parameters
and resources concerned, which can include very basic information
needed for addressing and routing, as well as anything else that
might be configured in a conventional non-autonomic network. The
atomic unit of discovery, synchronization or negotiation is referred
to as a technical objective, i.e, a configurable parameter or set of
parameters (defined more precisely in Section 3.1).

Following this Introduction, Section 2 describes the requirements for
discovery, synchronization and negotiation. Negotiation is an
iterative process, requiring multiple message exchanges forming a
closed loop between the negotiating entities. In fact, these
entities are ASAs, normally but not necessarily in different network
devices. State synchronization, when needed, can be regarded as a
special case of negotiation, without iteration. Section 3.3
describes a behavior model for a protocol intended to support
discovery, synchronization and negotiation. The design of GeneRic
Autonomic Signaling Protocol (GRASP) in Section 3 of this document is
mainly based on this behavior model. The relevant capabilities of
various existing protocols are reviewed in Appendix E.

The proposed discovery mechanism is oriented towards synchronization
and negotiation objectives. It is based on a neighbor discovery
process, but also supports diversion to off-link peers. There is no
assumption of any particular form of network topology. When a device
starts up with no pre-configuration, it has no knowledge of the
topology. The protocol itself is capable of being used in a small
and/or flat network structure such as a small office or home network
as well as a professionally managed network. Therefore, the
discovery mechanism needs to be able to allow a device to bootstrap
itself without making any prior assumptions about network structure.

Because GRASP can be used to perform a decision process among
distributed devices or between networks, it must run in a secure and
strongly authenticated environment.

It is understood that in realistic deployments, not all devices will
support GRASP. It is expected that some autonomic service agents
will directly manage a group of non-autonomic nodes, and that other
non-autonomic nodes will be managed traditionally. Such mixed
scenarios are not discussed in this specification.

2. Requirement Analysis of Discovery, Synchronization and Negotiation

This section discusses the requirements for discovery, negotiation
and synchronization capabilities. The primary user of the protocol
is an autonomic service agent (ASA), so the requirements are mainly

expressed as the features needed by an ASA. A single physical device
might contain several ASAs, and a single ASA might manage several
technical objectives. If a technical objective is managed by several
ASAs, any necessary coordination is outside the scope of the
signaling protocol itself.

Note that requirements for ASAs themselves, such as the processing of
Intent [RFC7575] or interfaces for coordination between ASAs are out
of scope for the present document.

2.1. Requirements for Discovery

D1. ASAs may be designed to manage anything, as required in
Section 2.2. A basic requirement is therefore that the protocol can
represent and discover any kind of technical objective among
arbitrary subsets of participating nodes.

In an autonomic network we must assume that when a device starts up
it has no information about any peer devices, the network structure,
or what specific role it must play. The ASA(s) inside the device are
in the same situation. In some cases, when a new application session
starts up within a device, the device or ASA may again lack
information about relevant peers. For example, it might be necessary
to set up resources on multiple other devices, coordinated and
matched to each other so that there is no wasted resource. Security
settings might also need updating to allow for the new device or
user. The relevant peers may be different for different technical
objectives. Therefore discovery needs to be repeated as often as
necessary to find peers capable of acting as counterparts for each
objective that a discovery initiator needs to handle. From this
background we derive the next three requirements:

D2. When an ASA first starts up, it has no knowledge of the specific
network to which it is attached. Therefore the discovery process
must be able to support any network scenario, assuming only that the
device concerned is bootstrapped from factory condition.

D3. When an ASA starts up, it must require no configured location
information about any peers in order to discover them.

D4. If an ASA supports multiple technical objectives, relevant peers
may be different for different discovery objectives, so discovery
needs to be performed separately to find counterparts for each
objective. Thus, there must be a mechanism by which an ASA can
separately discover peer ASAs for each of the technical objectives
that it needs to manage, whenever necessary.

D5. Following discovery, an ASA will normally perform negotiation or
synchronization for the corresponding objectives. The design should
allow for this by conveniently linking discovery to negotiation and
synchronization. It may provide an optional mechanism to combine
discovery and negotiation/synchronization in a single call.

D6. Some objectives may only be significant on the local link, but
others may be significant across the routed network and require off-
link operations. Thus, the relevant peers might be immediate
neighbors on the same layer 2 link, or they might be more distant and
only accessible via layer 3. The mechanism must therefore provide
both on-link and off-link discovery of ASAs supporting specific
technical objectives.

D7. The discovery process should be flexible enough to allow for
special cases, such as the following:

o During initialisation, a device must be able to establish mutual
   trust with the rest of the network and join an authentication
   mechanism. Although this will inevitably start with a discovery
   action, it is a special case precisely because trust is not yet
   established. This topic is the subject of
   [I-D.ietf-anima-bootstrapping-keyinfra]. We require that once
   trust has been established for a device, all ASAs within the
   device inherit the device's credentials and are also trusted.

o Depending on the type of network involved, discovery of other
   central functions might be needed, such as the Network Operations
   Center (NOC) [I-D.ietf-anima-stable-connectivity]. The protocol
   must be capable of supporting such discovery during
   initialisation, as well as discovery during ongoing operation.

D8. The discovery process must not generate excessive traffic and
must take account of sleeping nodes.

D9. There must be a mechanism for handling stale discovery results.

2.2. Requirements for Synchronization and Negotiation Capability

As background, consider the example of routing protocols, the closest
approximation to autonomic networking already in widespread use.
Routing protocols use a largely autonomic model based on distributed
devices that communicate repeatedly with each other. The focus is
reachability, so current routing protocols mainly consider simple
link status, i.e., up or down, and an underlying assumption is that
all nodes need a consistent view of the network topology in order for
the routing algorithm to converge. Thus, routing is mainly based on
information synchronization between peers, rather than on bi-

directional negotiation. Other information, such as latency,
congestion, capacity, and particularly unused capacity, would be
helpful to get better path selection and utilization rate, but is not
normally used in distributed routing algorithms. Additionally,
autonomic networks need to be able to manage many more dimensions,
such as security settings, power saving, load balancing, etc. Status
information and traffic metrics need to be shared between nodes for
dynamic adjustment of resources and for monitoring purposes. While
this might be achieved by existing protocols when they are available,
the new protocol needs to be able to support parameter exchange,
including mutual synchronization, even when no negotiation as such is
required. In general, these parameters do not apply to all
participating nodes, but only to a subset.

SN1. A basic requirement for the protocol is therefore the ability
to represent, discover, synchronize and negotiate almost any kind of
network parameter among selected subsets of participating nodes.

SN2. Negotiation is a request/response process that must be
guaranteed to terminate (with success or failure) and if necessary it
must contain tie-breaking rules for each technical objective that
requires them. While these must be defined specifically for each use
case, the protocol should have some general mechanisms in support of
loop and deadlock prevention, such as hop count limits or timeouts.

SN3. Synchronization might concern small groups of nodes or very
large groups. Different solutions might be needed at different
scales.

SN4. To avoid "reinventing the wheel", the protocol should be able
to encapsulate the data formats used by existing configuration
protocols (such as NETCONF/YANG) in cases where that is convenient.

SN5. Human intervention in complex situations is costly and error-
prone. Therefore, synchronization or negotiation of parameters
without human intervention is desirable whenever the coordination of
multiple devices can improve overall network performance. It
therefore follows that the protocol, as part of the Autonomic
Networking Infrastructure, should be capable of running in any device
that would otherwise need human intervention. The issue of running
in constrained nodes is discussed in
[I-D.ietf-anima-reference-model].

SN6. Human intervention in large networks is often replaced by use
of a top-down network management system (NMS). It therefore follows
that the protocol, as part of the Autonomic Networking
Infrastructure, should be capable of running in any device that would

otherwise be managed by an NMS, and that it can co-exist with an NMS, and with protocols such as SNMP and NETCONF.

SN7. Some features are expected to be implemented by individual ASAs, but the protocol must be general enough to allow them:

o Dependencies and conflicts: In order to decide upon a configuration for a given device, the device may need information from neighbors. This can be established through the negotiation procedure, or through synchronization if that is sufficient. However, a given item in a neighbor may depend on other information from its own neighbors, which may need another negotiation or synchronization procedure to obtain or decide. Therefore, there are potential dependencies and conflicts among negotiation or synchronization procedures. Resolving dependencies and conflicts is a matter for the individual ASAs involved. To allow this, there need to be clear boundaries and convergence mechanisms for negotiations. Also some mechanisms are needed to avoid loop dependencies. In such a case, the protocol's role is limited to bilateral signaling between ASAs.

o Recovery from faults and identification of faulty devices should be as automatic as possible. The protocol's role is limited to discovery, synchronization and negotiation. These processes can occur at any time, and an ASA may need to repeat any of these steps when the ASA detects an anomaly such as a negotiation counterpart failing.

o Since the goal is to minimize human intervention, it is necessary that the network can in effect "think ahead" before changing its parameters. One aspect of this is an ASA that relies on a knowledge base to predict network behavior. This is out of scope for the signaling protocol. However, another aspect is forecasting the effect of a change by a "dry run" negotiation before actually installing the change. This will be an application of the protocol rather than a feature of the protocol itself.

o Management logging, monitoring, alerts and tools for intervention are required. However, these can only be features of individual ASAs. Another document [I-D.ietf-anima-stable-connectivity] discusses how such agents may be linked into conventional OAM systems via an Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane].

SN8. The protocol will be able to deal with a wide variety of technical objectives, covering any type of network parameter. Therefore the protocol will need a flexible and easily extensible

format for describing objectives. At a later stage it may be
desirable to adopt an explicit information model. One consideration
is whether to adopt an existing information model or to design a new
one.

2.3. Specific Technical Requirements

T1. It should be convenient for ASA designers to define new
technical objectives and for programmers to express them, without
excessive impact on run-time efficiency and footprint. In
particular, it should be possible for ASAs to be implemented
independently of each other as user space programs rather than as
kernel code. The classes of device in which the protocol might run
is discussed in [I-D.ietf-anima-reference-model].

T2. The protocol should be easily extensible in case the initially
defined discovery, synchronization and negotiation mechanisms prove
to be insufficient.

T3. To be a generic platform, the protocol payload format should be
independent of the transport protocol or IP version. In particular,
it should be able to run over IPv6 or IPv4. However, some functions,
such as multicasting on a link, might need to be IP version
dependent. In case of doubt, IPv6 should be preferred.

T4. The protocol must be able to access off-link counterparts via
routable addresses, i.e., must not be restricted to link-local
operation.

T5. It must also be possible for an external discovery mechanism to
be used, if appropriate for a given technical objective. In other
words, GRASP discovery must not be a prerequisite for GRASP
negotiation or synchronization.

T6. The protocol must be capable of supporting multiple simultaneous
operations, especially when wait states occur.

T7. Intent: There must be provision for general Intent rules to be
applied by all devices in the network (e.g., security rules, prefix
length, resource sharing rules). However, Intent distribution might
not use the signaling protocol itself, but its design should not
exclude such use.

T8. Management monitoring, alerts and intervention: Devices should
be able to report to a monitoring system. Some events must be able
to generate operator alerts and some provision for emergency
intervention must be possible (e.g. to freeze synchronization or
negotiation in a mis-behaving device). These features might not use

the signaling protocol itself, but its design should not exclude such
use.

T9. The protocol needs to be fully secured against forged messages
and man-in-the middle attacks, and secured as much as reasonably
possible against denial of service attacks. It needs to be capable
of encryption in order to resist unwanted monitoring. However, it is
not required that the protocol itself provides these security
features; it may depend on an existing secure environment.

3. GRASP Protocol Overview

3.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
[RFC2119] when they appear in ALL CAPS. When these words are not in
ALL CAPS (such as "should" or "Should"), they have their usual
English meanings, and are not to be interpreted as [RFC2119] key
words.

This document uses terminology defined in [RFC7575].

The following additional terms are used throughout this document:

o Autonomic Device: identical to Autonomic Node.

o Discovery: a process by which an ASA discovers peers according to
   a specific discovery objective. The discovery results may be
   different according to the different discovery objectives. The
   discovered peers may later be used as negotiation counterparts or
   as sources of synchronization data.

o Negotiation: a process by which two ASAs interact iteratively to
   agree on parameter settings that best satisfy the objectives of
   both ASAs.

o State Synchronization: a process by which ASAs interact to receive
   the current state of parameter values stored in other ASAs. This
   is a special case of negotiation in which information is sent but
   the ASAs do not request their peers to change parameter settings.
   All other definitions apply to both negotiation and
   synchronization.

o Technical Objective (usually abbreviated as Objective): A
   technical objective is a configurable parameter or set of
   parameters of some kind, which occurs in three contexts:

Discovery, Negotiation and Synchronization. In the protocol, an objective is represented by an identifier and, if relevant, a value. Normally, a given objective will not occur in negotiation and synchronization contexts simultaneously.

* One ASA may support multiple independent objectives.

* The parameter described by a given objective is naturally based on a specific service or function or action. It may in principle be anything that can be set to a specific logical, numerical or string value, or a more complex data structure, by a network node. That node is generally expected to contain an ASA which may itself manage subsidiary non-autonomic nodes.

* Discovery Objective: an objective in the process of discovery. Its value may be undefined.

* Synchronization Objective: an objective whose specific technical content needs to be synchronized among two or more ASAs.

* Negotiation Objective: an objective whose specific technical content needs to be decided in coordination with another ASA.

A detailed discussion of objectives, including their format, is found in Section 3.10.

o Discovery Initiator: an ASA that spontaneously starts discovery by sending a discovery message referring to a specific discovery objective.

o Discovery Responder: a peer that either contains an ASA supporting the discovery objective indicated by the discovery initiator, or caches the locator(s) of the ASA(s) supporting the objective. It sends a Discovery Response, as described later.

o Synchronization Initiator: an ASA that spontaneously starts synchronization by sending a request message referring to a specific synchronization objective.

o Synchronization Responder: a peer ASA which responds with the value of a synchronization objective.

o Negotiation Initiator: an ASA that spontaneously starts negotiation by sending a request message referring to a specific negotiation objective.

   o Negotiation Counterpart: a peer with which the Negotiation
      Initiator negotiates a specific negotiation objective.

3.2. High Level Deployment Model

   It is expected that a GRASP implementation will reside in an
   autonomic node that also contains both the appropriate security
   environment, preferably the Autonomic Control Plane (ACP)
   [I-D.ietf-anima-autonomic-control-plane], and one or more Autonomic
   Service Agents (ASAs). In the minimal case of a single-purpose
   device, these three components might be fully integrated. A more
   common model is expected to be a multi-purpose device capable of
   containing several ASAs. In this case it is expected that the ACP,
   GRASP and the ASAs will be implemented as separate processes, which
   are probably multi-threaded to support asynchronous and simultaneous
   operations. It is expected that GRASP will access the ACP by using a
   typical socket interface. A well defined Application Programming
   Interface (API) will be needed between GRASP and the ASAs. In some
   implementations, ASAs would run in user space with a GRASP library
   providing the API, and this library would in turn communicate via
   system calls with core GRASP functions. For further details of
   possible deployment models, see [I-D.ietf-anima-reference-model].

   A GRASP instance must be aware of its network interfaces, and of its
   own global-scope and link-local addresses. In the presence of the
   ACP, such information will be available from the adjacency table
   discussed in [I-D.ietf-anima-reference-model]. In other cases, GRASP
   must determine such information for itself. Details depend on the
   operating system.

   Because GRASP needs to work whatever happens, especially during
   bootstrapping and during fault conditions, it is essential that every
   implementation is as robust as possible. For example, discovery
   failures, or any kind of socket error at any time, must not cause
   irrecoverable failures in GRASP itself, and must return suitable
   error codes through the API so that ASAs can also recover.

   GRASP must always start up correctly after a system restart. All run
   time error conditions, and events such as address renumbering,
   network interface failures, and CPU sleep/wake cycles, must be
   handled in such a way that GRASP will still operate correctly and
   securely (Section 3.5.1) afterwards.

   An autonomic node will normally run a single instance of GRASP, used
   by multiple ASAs. However, scenarios where multiple instances of
   GRASP run in a single node, perhaps with different security
   properties, are not excluded. In this case, each instance MUST

listen independently for GRASP link-local multicasts in order for
discovery and flooding to work correctly.

[3.3](). **High Level Design Choices**

This section describes a behavior model and design choices for GRASP,
supporting discovery, synchronization and negotiation, to act as a
platform for different technical objectives.

o A generic platform

   The protocol is designed as a generic platform, which is
   independent from the synchronization or negotiation contents. It
   takes care of the general intercommunication between counterparts.
   The technical contents will vary according to the various
   technical objectives and the different pairs of counterparts.

o The protocol is expected to form part of an Autonomic Networking
   Infrastructure [[I-D.ietf-anima-reference-model]()]. It will provide
   services to ASAs via a suitable application programming interface
   (API), which will reflect the protocol elements but will not
   necessarily be in one-to-one correspondence to them. This API is
   out of scope for the present document.

o It is normally expected that a single main instance of GRASP will
   exist in an autonomic node, and that the protocol engine and each
   ASA will run as independent asynchronous processes. However,
   separate GRASP instances may exist for security-related reasons
   ([Section 3.5.2]()).

o Security infrastructure and trust relationship

   Because this negotiation protocol may directly cause changes to
   device configurations and bring significant impacts to a running
   network, this protocol is required to run within an existing
   secure environment with strong authentication. As a design
   choice, the protocol itself is not provided with built-in security
   functionality.

   On the other hand, a limited negotiation model might be deployed
   based on a limited trust relationship such as that between two
   administrative domains. ASAs might then exchange limited
   information and negotiate some particular configurations.

o Discovery, synchronization and negotiation are designed together.

   The discovery method and the synchronization and negotiation
   methods are designed in the same way and can be combined when this
   is useful, allowing a rapid mode of operation described in
   Section 3.5.4. These processes can also be performed
   independently when appropriate.

   * Thus, for some objectives, especially those concerned with
      application layer services, another discovery mechanism such as
      the future DNS Service Discovery [RFC7558] MAY be used. The
      choice is left to the designers of individual ASAs.

o A uniform pattern for technical objectives

   The synchronization and negotiation objectives are defined
   according to a uniform pattern. The values that they contain
   could be carried either in a simple binary format or in a complex
   object format. The basic protocol design uses the Concise Binary
   Object Representation (CBOR) [RFC7049], which is readily
   extensible for unknown future requirements.

o A flexible model for synchronization

   GRASP supports bilateral synchronization, which could be used to
   perform synchronization among a small number of nodes. It also
   supports an unsolicited flooding mode when large groups of nodes,
   possibly including all autonomic nodes, need data for the same
   technical objective.

   * There may be some network parameters for which a more
      traditional flooding mechanism such as DNCP [RFC7787] is
      considered more appropriate. GRASP can coexist with DNCP.

o A simple initiator/responder model for negotiation

   Multi-party negotiations are very complicated to model and cannot
   readily be guaranteed to converge. GRASP uses a simple bilateral
   model and can support multi-party negotiations by indirect steps.

o Organizing of synchronization or negotiation content

   The technical content transmitted by GRASP will be organized
   according to the relevant function or service. The objectives for

    different functions or services are kept separate, because they
    may be negotiated or synchronized with different counterparts or
    have different response times. Thus a normal arrangement would be
    a single ASA managing a small set of closely related objectives,
    with a version of that ASA in each relevant autonomic node.
    Further discussion of this aspect is out of scope for the current
    document.

    o Requests and responses in negotiation procedures

       The initiator can negotiate a specific negotiation objective with
       relevant counterpart ASAs. It can request relevant information
       from a counterpart so that it can coordinate its local
       configuration. It can request the counterpart to make a matching
       configuration. It can request simulation or forecast results by
       sending some dry run conditions.

       Beyond the traditional yes/no answer, the responder can reply with
       a suggested alternative value for the objective concerned. This
       would start a bi-directional negotiation ending in a compromise
       between the two ASAs.

    o Convergence of negotiation procedures

       To enable convergence, when a responder suggests a new value or
       condition in a negotiation step reply, it should be as close as
       possible to the original request or previous suggestion. The
       suggested value of later negotiation steps should be chosen
       between the suggested values from the previous two steps. GRASP
       provides mechanisms to guarantee convergence (or failure) in a
       small number of steps, i.e. a timeout and a maximum number of
       iterations.

    o Extensibility

       GRASP does not have a version number. In most cases new semantics
       will be added by defining new synchronization or negotiation
       objectives. However, the protocol could be extended by adding new
       message types and options in future.

3.4. Quick Operating Overview

GRASP is expected to run as an operating system core module,
providing an API (such as [I-D.liu-anima-grasp-api]) to interface to
less privileged ASAs. Thus ASAs may operate without special
privilege, unless they need it for other reasons (such as configuring
IP addresses or manipulating routing tables).

The GRASP mechanisms used by the ASA are built around GRASP
objectives defined as data structures containing administrative
information such as the objective's unique name, and its current
value. The format and size of the value is not restricted by the
protocol, except that it must be possible to serialise it for
transmission in CBOR, which is no restriction at all in practice.

The GRASP provides the following mechanisms:

o A discovery mechanism (M_DISCOVERY, M_RESPONSE), by which an ASA
   can discover other ASAs supporting a given objective.

o A negotiation request mechanism (M_REQ_NEG), by which an ASA can
   start negotiation of an objective with a counterpart ASA. Once a
   negotiation has started, the process is symmetrical, and there is
   a negotiation step message (M_NEGOTIATE) for each ASA to use in
   turn. Two other functions support negotiating steps (M_WAIT,
   M_END).

o A synchronization mechanism (M_REQ_SYN), by which an ASA can
   request the current value of an objective from a counterpart ASA.
   With this, there is a corresponding response function (M_SYNCH)
   for an ASA that wishes to respond to synchronization requests.

o A flood mechanism (M_FLOOD), by which an ASA can cause the current
   value of an objective to be flooded throughout the AN so that any
   ASA can receive it. One application of this is to act as an
   announcement, avoiding the need for discovery of a widely
   applicable objective.

Some example messages and simple message flows are provided in
Appendix D.

3.5. GRASP Protocol Basic Properties and Mechanisms

3.5.1. Required External Security Mechanism

The protocol SHOULD always run within a secure Autonomic Control
Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. The ACP is
assumed to carry all messages securely, including link-local

multicast if possible. A GRASP implementation MUST verify whether
the ACP is operational.

If there is no ACP, the protocol MUST use another form of strong
authentication and SHOULD use a form of strong encryption. See
Section 3.5.2.1 for further discussion.

The ACP, or in its absence another security mechanism, sets the
boundary within which nodes are trusted as GRASP peers. A GRASP
implementation MUST refuse to execute GRASP synchronization and
negotiation functions if there is neither an operational ACP nor
another secure environment.

Link-local multicast is used for discovery messages. Responses to
discovery messages MUST be secured, with one exception mentioned in
the next section.

3.5.2. Constrained Instances

This section describes some examples of cases where additional
instances of GRASP subject to certain constraints are appropriate.

3.5.2.1. No ACP

As mentioned in Section 3.3, some GRASP operations might be performed
across an administrative domain boundary by mutual agreement, without
the benefit of an ACP. Such operations MUST be confined to a
separate instance of GRASP with its own copy of all GRASP data
structures. Messages MUST be authenticated and SHOULD be encrypted.
TLS [RFC5246] and DTLS [RFC6347] based on a Public Key Infrastructure
(PKI) [RFC5280] are RECOMMENDED for this purpose. Further details
are out of scope for this document.

3.5.2.2. Discovery Unsolicited Link-Local

Some services may need to use insecure GRASP discovery, response and
flood messages without being able to use pre-existing security
associations. Such operations being intrinsically insecure, they
need to be confined to link-local use to minimise the risk of
malicious actions. Possible examples include discovery of candidate
ACP neighbors [I-D.ietf-anima-autonomic-control-plane], discovery of
bootstrap proxies [I-D.ietf-anima-bootstrapping-keyinfra] or perhaps
initialisation services in networks using GRASP without being fully
autonomic (e.g., no ACP). Such usage MUST be limited to link-local
operations and MUST be confined to a separate insecure instance of
GRASP with its own copy of all GRASP data structures. This instance
is nicknamed DULL - Discovery Unsolicited Link-Local.

The detailed rules for the DULL instance of GRASP are as follows:

o An initiator MUST only send Discovery or Flood Synchronization
   link-local multicast messages with a loop count of 1. A responder
   SHOULD NOT send a Discovery Response message unless it cannot be
   avoided. Other GRASP message types MUST NOT be sent.

o A responder MUST silently discard any message whose loop count is
   not 1.

o A responder MUST silently discard any message referring to a GRASP
   Objective that is not directly part of a service that requires
   this insecure mode.

o A responder MUST NOT relay any multicast messages.

o A Discovery Response MUST indicate a link-local address.

o A Discovery Response MUST NOT include a Divert option.

o A node MUST silently discard any message whose source address is
   not link-local.

GRASP traffic SHOULD be minimized by using only Flood Synchronization
to announce objectives and their associated locators, rather than by
using Discovery and Response. Further details are out of scope for
this document

3.5.2.3. Secure Only Neighbor Negotiation

Some services might use insecure on-link operations as in DULL, but
also use unicast synchronization or negotiation operations protected
by TLS. A separate instance of GRASP is used, with its own copy of
all GRASP data structures. This instance is nicknamed SONN - Secure
Only Neighbor Negotiation.

The detailed rules for the SONN instance of GRASP are as follows:

o Any type of GRASP message MAY be sent.

o An initiator MUST send any Discovery or Flood Synchronization
   link-local multicast messages with a loop count of 1.

o A responder MUST silently discard any Discovery or Flood
   Synchronization message whose loop count is not 1.

o A responder MUST silently discard any message referring to a GRASP
   Objective that is not directly part of the service concerned.

o A responder MUST NOT relay any multicast messages.

o A Discovery Response MUST indicate a link-local address.

o A Discovery Response MUST NOT include a Divert option.

o A node MUST silently discard any message whose source address is
   not link-local.

Further details, including TLS and PKI usage, are out of scope for
this document.

3.5.3. Transport Layer Usage

GRASP discovery and flooding messages are designed for use over link-
local multicast UDP. They MUST NOT be fragmented, and therefore MUST
NOT exceed the link MTU size. Nothing in principle prevents them
from working over some other method of sending packets to all on-link
neighbors, but this is out of scope for the present specification.

All other GRASP messages are unicast and could in principle run over
any transport protocol. An implementation MUST support use of TCP.
It MAY support use of another transport protocol. However, GRASP
itself does not provide for error detection or retransmission. Use
of an unreliable transport protocol is therefore NOT RECOMMENDED.

Nevertheless, when running within a secure ACP on reliable
infrastructure, UDP MAY be used for unicast messages not exceeding
the minimum IPv6 path MTU; however, TCP MUST be used for longer
messages. In other words, IPv6 fragmentation is avoided. If a node
receives a UDP message but the reply is too long, it MUST open a TCP
connection to the peer for the reply. Note that when the network is
under heavy load or in a fault condition, UDP might become
unreliable. Since this is when autonomic functions are most
necessary, automatic fallback to TCP MUST be implemented. The
simplest implementation is therefore to use only TCP.

For considerations when running without an ACP, see Section 3.5.2.1.

For link-local multicast, the GRASP protocol listens to the well-
known GRASP Listen Port (Section 3.6). For unicast transport
sessions used for discovery responses, synchronization and
negotiation, the ASA concerned normally listens on its own
dynamically assigned ports, which are communicated to its peers
during discovery. However, a minimal implementation MAY use the
GRASP Listen Port for this purpose.

3.5.4. Discovery Mechanism and Procedures

3.5.4.1. Separated discovery and negotiation mechanisms

   Although discovery and negotiation or synchronization are defined
   together in GRASP, they are separate mechanisms. The discovery
   process could run independently from the negotiation or
   synchronization process. Upon receiving a Discovery (Section 3.8.4)
   message, the recipient node should return a response message in which
   it either indicates itself as a discovery responder or diverts the
   initiator towards another more suitable ASA.

   The discovery action (M_DISCOVERY) will normally be followed by a
   negotiation (M_REQ_NEG) or synchronization (M_REQ_SYN) action. The
   discovery results could be utilized by the negotiation protocol to
   decide which ASA the initiator will negotiate with.

   The initiator of a discovery action for a given objective need not be
   capable of responding to that objective as a Negotiation Counterpart,
   as a Synchronization Responder or as source for flooding. For
   example, an ASA might perform discovery even if it only wishes to act
   a Synchronization Initiator or Negotiation Initiator. Such an ASA
   does not itself need to respond to discovery messages.

   It is also entirely possible to use GRASP discovery without any
   subsequent negotiation or synchronization action. In this case, the
   discovered objective is simply used as a name during the discovery
   process and any subsequent operations between the peers are outside
   the scope of GRASP.

3.5.4.2. Discovery Overview

   A complete discovery process will start with a multicast (of
   M_DISCOVERY) on the local link. On-link neighbors supporting the
   discovery objective will respond directly (with M_RESPONSE). A
   neighbor with multiple interfaces will respond with a cached
   discovery response if any. However, it SHOULD NOT respond with a
   cached response on an interface if it learnt that information from
   the same interface. If it has no cached response, it will relay the
   discovery on its other interfaces, for example reaching a higher-
   level gateway in a hierarchical network. If a node receiving the
   relayed discovery supports the discovery objective, it will respond
   to the relayed discovery. If it has a cached response, it will
   respond with that. If not, it will repeat the discovery process,
   which thereby becomes recursive. The loop count and timeout will
   ensure that the process ends.

Exceptionally, a Discovery message MAY be sent unicast (via UDP or
TCP) to a peer node, which will then proceed exactly as if the
message had been multicast, except that when TCP is used, the
response will be on the same socket as the query. However, this mode
does not guarantee successful discovery in the general case.

3.5.4.3. Discovery Procedures

Discovery starts as an on-link operation. The Divert option can tell
the discovery initiator to contact an off-link ASA for that discovery
objective. Every Discovery message is sent by a discovery initiator
via UDP to the ALL_GRASP_NEIGHBOR link-local multicast address
(Section 3.6). Every network device that supports GRASP always
listens to a well-known UDP port to capture the discovery messages.
Because this port is unique in a device, this is a function of the
GRASP core and not of an individual ASA. As a result, each ASA will
need to register the objectives that it supports with the GRASP core.

If an ASA in a neighbor device supports the requested discovery
objective, the device SHOULD respond to the link-local multicast with
a unicast Discovery Response message (Section 3.8.5) with locator
option(s), unless it is temporarily unavailable. Otherwise, if the
neighbor has cached information about an ASA that supports the
requested discovery objective (usually because it discovered the same
objective before), it SHOULD respond with a Discovery Response
message with a Divert option pointing to the appropriate Discovery
Responder.

If a device has no information about the requested discovery
objective, and is not acting as a discovery relay (see below) it MUST
silently discard the Discovery message.

If no discovery response is received within a reasonable timeout
(default GRASP_DEF_TIMEOUT milliseconds, Section 3.6), the Discovery
message MAY be repeated, with a newly generated Session ID
(Section 3.7). An exponential backoff SHOULD be used for subsequent
repetitions, to limit the load during busy periods. Frequent
repetition might be symptomatic of a denial of service attack.

After a GRASP device successfully discovers a locator for a Discovery
Responder supporting a specific objective, it MUST cache this
information, including the interface identifier via which it was
discovered. This cache record MAY be used for future negotiation or
synchronization, and the locator SHOULD be passed on when appropriate
as a Divert option to another Discovery Initiator.

The cache mechanism MUST include a lifetime for each entry. The
lifetime is derived from a time-to-live (ttl) parameter in each

Discovery Response message. Cached entries MUST be ignored or deleted after their lifetime expires. In some environments, unplanned address renumbering might occur. In such cases, the lifetime SHOULD be short compared to the typical address lifetime and a mechanism to flush the discovery cache SHOULD be implemented. The discovery mechanism needs to track the node's current address to ensure that Discovery Responses always indicate the correct address.

If multiple Discovery Responders are found for the same objective, they SHOULD all be cached, unless this creates a resource shortage. The method of choosing between multiple responders is an implementation choice. This choice MUST be available to each ASA but the GRASP implementation SHOULD provide a default choice.

Because Discovery Responders will be cached in a finite cache, they might be deleted at any time. In this case, discovery will need to be repeated. If an ASA exits for any reason, its locator might still be cached for some time, and attempts to connect to it will fail. ASAs need to be robust in these circumstances.

3.5.4.4. Discovery Relaying

A GRASP instance with multiple link-layer interfaces (typically running in a router) MUST support discovery on all interfaces. We refer to this as a 'relaying instance'.

However, different interfaces can be at different security levels: each group of interfaces with the same security level SHOULD be serviced by the same GRASP process, except for Limited Security Instances Section 3.5.2 which are always single-interface instances and MUST NOT perform discovery relaying.

If a relaying instance receives a Discovery message on a given interface for a specific objective that it does not support and for which it has not previously cached a Discovery Responder, it MUST relay the query by re-issuing a Discovery message as a link-local multicast on its other interfaces.

The relayed discovery message MUST have the same Session ID as the incoming discovery message and MUST be tagged with the IP address of its original initiator (see Section 3.8.4). Note that this initiator address is only used to allow for disambiguation of the Session ID and is never used to address Response packets.

Since the relay device is unaware of the timeout set by the original initiator it SHOULD set a timeout at least equal to GRASP_DEF_TIMEOUT milliseconds.

The relaying instance MUST decrement the loop count within the
objective, and MUST NOT relay the Discovery message if the result is
zero. Also, it MUST limit the total rate at which it relays
discovery messages to a reasonable value, in order to mitigate
possible denial of service attacks. It MUST cache the Session ID
value and initiator address of each relayed Discovery message until
any Discovery Responses have arrived or the discovery process has
timed out. To prevent loops, it MUST NOT relay a Discovery message
which carries a given cached Session ID and initiator address more
than once. These precautions avoid discovery loops and mitigate
potential overload.

The discovery results received by the relaying instance MUST in turn
be sent as a Discovery Response message to the Discovery message that
caused the relay action.

This relayed discovery mechanism, with caching of the results, should
be sufficient to support most network bootstrapping scenarios.

3.5.4.5. Rapid Mode (Discovery/Negotiation binding)

A Discovery message MAY include a Negotiation Objective option. This
allows a rapid mode of negotiation described in Section 3.5.5. A
similar mechanism is defined for synchronization in Section 3.5.6.

Note that rapid mode is currently limited to a single objective for
simplicity of design and implementation. A possible future extension
is to allow multiple objectives in rapid mode for greater efficiency.

3.5.5. Negotiation Procedures

A negotiation initiator sends a negotiation request to a counterpart
ASA, including a specific negotiation objective. It may request the
negotiation counterpart to make a specific configuration.
Alternatively, it may request a certain simulation or forecast result
by sending a dry run configuration. The details, including the
distinction between dry run and an actual configuration change, will
be defined separately for each type of negotiation objective.

If no reply message of any kind is received within a reasonable
timeout (default GRASP_DEF_TIMEOUT milliseconds, Section 3.6), the
negotiation request MAY be repeated, with a newly generated Session
ID (Section 3.7). An exponential backoff SHOULD be used for
subsequent repetitions.

If the counterpart can immediately apply the requested configuration,
it will give an immediate positive (O_ACCEPT) answer (using M_END).
This will end the negotiation phase immediately. Otherwise, it will

negotiate (using M_NEGOTIATE). It will reply with a proposed
alternative configuration that it can apply (typically, a
configuration that uses fewer resources than requested by the
negotiation initiator). This will start a bi-directional negotiation
(using M_NEGOTIATE) to reach a compromise between the two ASAs.

The negotiation procedure is ended when one of the negotiation peers
sends a Negotiation Ending (M_END) message, which contains an accept
(O_ACCEPT) or decline (O_DECLINE) option and does not need a response
from the negotiation peer. Negotiation may also end in failure
(equivalent to a decline) if a timeout is exceeded or a loop count is
exceeded.

A negotiation procedure concerns one objective and one counterpart.
Both the initiator and the counterpart may take part in simultaneous
negotiations with various other ASAs, or in simultaneous negotiations
about different objectives. Thus, GRASP is expected to be used in a
multi-threaded mode. Certain negotiation objectives may have
restrictions on multi-threading, for example to avoid over-allocating
resources.

Some configuration actions, for example wavelength switching in
optical networks, might take considerable time to execute. The ASA
concerned needs to allow for this by design, but GRASP does allow for
a peer to insert latency in a negotiation process if necessary
([Section 3.8.9](#), M_WAIT).

[3.5.5.1](#). Rapid Mode (Discovery/Negotiation Linkage)

A Discovery message MAY include a Negotiation Objective option. In
this case it is as if the initiator sent the sequence M_DISCOVERY,
immediately followed by M_REQ_NEG. This has implications for the
construction of the GRASP core, as it must carefully pass the
contents of the Negotiation Objective option to the ASA so that it
may evaluate the objective directly. When a Negotiation Objective
option is present the ASA replies with an M_NEGOTIATE message (or
M_END with O_ACCEPT if it is immediately satisfied with the
proposal), rather than with an M_RESPONSE. However, if the recipient
node does not support rapid mode, discovery will continue normally.

It is possible that a Discovery Response will arrive from a responder
that does not support rapid mode, before such a Negotiation message
arrives. In this case, rapid mode will not occur.

This rapid mode could reduce the interactions between nodes so that a
higher efficiency could be achieved. However, a network in which
some nodes support rapid mode and others do not will have complex
timing-dependent behaviors. Therefore, the rapid negotiation

function SHOULD be configured off by default and MAY be configured on
or off by Intent.

[3.5.6](#). Synchronization and Flooding Procedure

A synchronization initiator sends a synchronization request to a
counterpart, including a specific synchronization objective. The
counterpart responds with a Synchronization message ([Section 3.8.10](#))
containing the current value of the requested synchronization
objective. No further messages are needed.

If no reply message of any kind is received within a reasonable
timeout (default GRASP_DEF_TIMEOUT milliseconds, [Section 3.6](#)), the
synchronization request MAY be repeated, with a newly generated
Session ID ([Section 3.7](#)). An exponential backoff SHOULD be used for
subsequent repetitions.

[3.5.6.1](#). Flooding

In the case just described, the message exchange is unicast and
concerns only one synchronization objective. For large groups of
nodes requiring the same data, synchronization flooding is available.
For this, a flooding initiator MAY send an unsolicited Flood
Synchronization message containing one or more Synchronization
Objective option(s), if and only if the specification of those
objectives permits it. This is sent as a multicast message to the
ALL_GRASP_NEIGHBOR multicast address ([Section 3.6](#)).

Every network device that supports GRASP always listens to a well-
known UDP port to capture flooding messages. Because this port is
unique in a device, this is a function of the GRASP core.

To ensure that flooding does not result in a loop, the originator of
the Flood Synchronization message MUST set the loop count in the
objectives to a suitable value (the default is GRASP_DEF_LOOPCT).
Also, a suitable mechanism is needed to avoid excessive multicast
traffic. This mechanism MUST be defined as part of the specification
of the synchronization objective(s) concerned. It might be a simple
rate limit or a more complex mechanism such as the Trickle algorithm
[[RFC6206](#)].

A GRASP device with multiple link-layer interfaces (typically a
router) MUST support synchronization flooding on all interfaces. If
it receives a multicast Flood Synchronization message on a given
interface, it MUST relay it by re-issuing a Flood Synchronization
message on its other interfaces. The relayed message MUST have the
same Session ID as the incoming message and MUST be tagged with the
IP address of its original initiator.

Link-layer Flooding is supported by GRASP by setting the loop count
to 1, and sending with a link-local source address. Floods with
link-local source addresses and a loop count other than 1 are
invalid, and such messages MUST be discarded.

The relaying device MUST decrement the loop count within the first
objective, and MUST NOT relay the Flood Synchronization message if
the result is zero. Also, it MUST limit the total rate at which it
relays Flood Synchronization messages to a reasonable value, in order
to mitigate possible denial of service attacks. It MUST cache the
Session ID value and initiator address of each relayed Flood
Synchronization message for a finite time not less than twice
GRASP_DEF_TIMEOUT milliseconds. To prevent loops, it MUST NOT relay
a Flood Synchronization message which carries a given cached Session
ID and initiator address more than once. These precautions avoid
synchronization loops and mitigate potential overload.

Note that this mechanism is unreliable in the case of sleeping nodes,
or new nodes that join the network, or nodes that rejoin the network
after a fault. An ASA that initiates a flood SHOULD repeat the flood
at a suitable frequency and SHOULD also act as a synchronization
responder for the objective(s) concerned. Thus nodes that require an
objective subject to flooding can either wait for the next flood or
request unicast synchronization for that objective.

The multicast messages for synchronization flooding are subject to
the security rules in Section 3.5.1. In practice this means that
they MUST NOT be transmitted and MUST be ignored on receipt unless
there is an operational ACP or equivalent strong security in place.
However, because of the security weakness of link-local multicast
(Section 5), synchronization objectives that are flooded SHOULD NOT
contain unencrypted private information and SHOULD be validated by
the recipient ASA.

3.5.6.2. Rapid Mode (Discovery/Synchronization Linkage)

A Discovery message MAY include a Synchronization Objective option.
In this case the Discovery message also acts as a Request
Synchronization message to indicate to the Discovery Responder that
it could directly reply to the Discovery Initiator with a
Synchronization message Section 3.8.10 with synchronization data for
rapid processing, if the discovery target supports the corresponding
synchronization objective. The design implications are similar to
those discussed in Section 3.5.5.1.

It is possible that a Discovery Response will arrive from a responder
that does not support rapid mode, before such a Synchronization
message arrives. In this case, rapid mode will not occur.

This rapid mode could reduce the interactions between nodes so that a
higher efficiency could be achieved. However, a network in which
some nodes support rapid mode and others do not will have complex
timing-dependent behaviors. Therefore, the rapid synchronization
function SHOULD be configured off by default and MAY be configured on
or off by Intent.

3.6. GRASP Constants

o ALL_GRASP_NEIGHBOR

   A link-local scope multicast address used by a GRASP-enabled
   device to discover GRASP-enabled neighbor (i.e., on-link) devices
   . All devices that support GRASP are members of this multicast
   group.

   * IPv6 multicast address: TBD1

   * IPv4 multicast address: TBD2

o GRASP_LISTEN_PORT (TBD3)

   A well-known UDP user port that every GRASP-enabled network device
   MUST always listen to for link-local multicasts. Additionally,
   this user port MAY be used to listen for TCP or UDP unicast
   messages in a simple implementation of GRASP (Section 3.5.3).

o GRASP_DEF_TIMEOUT (60000 milliseconds)

   The default timeout used to determine that a discovery etc. has
   failed to complete.

o GRASP_DEF_LOOPCT (6)

   The default loop count used to determine that a negotiation has
   failed to complete, and to avoid looping messages.

o GRASP_DEF_MAX_SIZE (2048)

   The default maximum message size in bytes.

3.7. Session Identifier (Session ID)

This is an up to 32-bit opaque value used to distinguish multiple
sessions between the same two devices. A new Session ID MUST be
generated by the initiator for every new Discovery, Flood
Synchronization or Request message. All responses and follow-up

messages in the same discovery, synchronization or negotiation
procedure MUST carry the same Session ID.

The Session ID SHOULD have a very low collision rate locally. It
MUST be generated by a pseudo-random algorithm using a locally
generated seed which is unlikely to be used by any other device in
the same network [RFC4086]. When allocating a new Session ID, GRASP
MUST check that the value is not already in use and SHOULD check that
it has not been used recently, by consulting a cache of current and
recent sessions. In the unlikely event of a clash, GRASP MUST
generate a new value.

However, there is a finite probability that two nodes might generate
the same Session ID value. For that reason, when a Session ID is
communicated via GRASP, the receiving node MUST tag it with the
initiator's IP address to allow disambiguation. In the highly
unlikely event of two peers opening sessions with the same Session ID
value, this tag will allow the two sessions to be distinguished.
Multicast GRASP messages and their responses, which may be relayed
between links, therefore include a field that carries the initiator's
global IP address.

There is a highly unlikely race condition in which two peers start
simultaneous negotiation sessions with each other using the same
Session ID value. Depending on various implementation choices, this
might lead to the two sessions being confused. See Section 3.8.6 for
details of how to avoid this.

3.8. GRASP Messages

3.8.1. Message Overview

This section defines the GRASP message format and message types.
Message types not listed here are reserved for future use.

The messages currently defined are:

   Discovery and Discovery Response.

   Request Negotiation, Negotiation, Confirm Waiting and Negotiation
   End.

   Request Synchronization, Synchronization, and Flood
   Synchronization.

   No Operation.

3.8.2. GRASP Message Format

GRASP messages share an identical header format and a variable format
area for options. GRASP message headers and options are transmitted
in Concise Binary Object Representation (CBOR) [RFC7049]. In this
specification, they are described using CBOR data definition language
(CDDL) [I-D.greevenbosch-appsawg-cbor-cddl]. Fragmentary CDDL is
used to describe each item in this section. A complete and normative
CDDL specification of GRASP is given in Section 6, including
constants such as message types.

Every GRASP message, except the No Operation message, carries a
Session ID (Section 3.7). Options are then presented serially in the
options field.

In fragmentary CDDL, every GRASP message follows the pattern:

```
grasp-message = (message .within message-structure) / noop-message

message-structure = [MESSAGE_TYPE, session-id, ?initiator,
                     *grasp-option]

MESSAGE_TYPE = 1..255
session-id = 0..4294967295 ;up to 32 bits
grasp-option = any
```

The MESSAGE_TYPE indicates the type of the message and thus defines
the expected options. Any options received that are not consistent
with the MESSAGE_TYPE SHOULD be silently discarded.

The No Operation (noop) message is described in Section 3.8.13.

The various MESSAGE_TYPE values are defined in Section 6.

All other message elements are described below and formally defined
in Section 6.

3.8.3. Message Size

GRASP nodes MUST be able to receive messages of at least
GRASP_DEF_MAX_SIZE bytes. GRASP nodes MUST NOT send messages longer
than GRASP_DEF_MAX_SIZE bytes unless a longer size is explicitly
allowed for the objective concerned. For example, GRASP negotiation
itself could be used to agree on a longer message size.

The message parser used by GRASP should be configured to know about
the GRASP_DEF_MAX_SIZE, or any larger negotiated message size, so
that it may defend against overly long messages.

3.8.4. **Discovery Message**

   In fragmentary CDDL, a Discovery message follows the pattern:

     discovery-message = [M_DISCOVERY, session-id, initiator, objective]

   A discovery initiator sends a Discovery message to initiate a
   discovery process for a particular objective option.

   The discovery initiator sends all Discovery messages via UDP to port
   GRASP_LISTEN_PORT at the link-local ALL_GRASP_NEIGHBOR multicast
   address on each link-layer interface in use by GRASP. It then
   listens for unicast TCP responses on a given port, and stores the
   discovery results (including responding discovery objectives and
   corresponding unicast locators).

   The listening port used for TCP MUST be the same port as used for
   sending the Discovery UDP multicast, on a given interface. In a low-
   end implementation this MAY be GRASP_LISTEN_PORT. In a more complex
   implementation, the GRASP discovery mechanism will find, for each
   interface, a dynamic port that it can bind to for both UDP and TCP
   before initiating any discovery.

   The 'initiator' field in the message is a globally unique IP address
   of the initiator, for the sole purpose of disambiguating the Session
   ID in other nodes. If for some reason the initiator does not have a
   globally unique IP address, it MUST use a link-local address for this
   purpose that is highly likely to be unique, for example using
   [RFC7217].

   A Discovery message MUST include exactly one of the following:

   o a discovery objective option (Section 3.10.1). Its loop count
      MUST be set to a suitable value to prevent discovery loops
      (default value is GRASP_DEF_LOOPCT). If the discovery initiator
      requires only on-link responses, the loop count MUST be set to 1.

   o a negotiation objective option (Section 3.10.1). This is used
      both for the purpose of discovery and to indicate to the discovery
      target that it MAY directly reply to the discovery initiatior with
      a Negotiation message for rapid processing, if it could act as the
      corresponding negotiation counterpart. The sender of such a
      Discovery message MUST initialize a negotiation timer and loop
      count in the same way as a Request Negotiation message
      (Section 3.8.6).

   o a synchronization objective option (Section 3.10.1). This is used
      both for the purpose of discovery and to indicate to the discovery

target that it MAY directly reply to the discovery initiator with
a Synchronization message for rapid processing, if it could act as
the corresponding synchronization counterpart. Its loop count
MUST be set to a suitable value to prevent discovery loops
(default value is GRASP_DEF_LOOPCT).

Exceptionally, a Discovery message MAY be sent unicast to a peer
node, which will then proceed exactly as if the message had been
multicast.

3.8.5. Discovery Response Message

In fragmentary CDDL, a Discovery Response message follows the
pattern:

```
response-message = [M_RESPONSE, session-id, initiator, ttl,
                   (+locator-option // divert-option), ?objective)]

ttl = 0..4294967295 ; in milliseconds
```

A node which receives a Discovery message SHOULD send a Discovery
Response message if and only if it can respond to the discovery.

It MUST contain the same Session ID and initiator as the Discovery
message.

It MUST contain a time-to-live (ttl) for the validity of the
response, given as a positive integer value in milliseconds. Zero
is treated as the default value GRASP_DEF_TIMEOUT (Section 3.6).

It MAY include a copy of the discovery objective from the
Discovery message.

It is sent to the sender of the Discovery message via TCP at the port
used to send the Discovery message (as explained in Section 3.8.4).

If the responding node supports the discovery objective of the
discovery, it MUST include at least one kind of locator option
(Section 3.9.5) to indicate its own location. A sequence of multiple
kinds of locator options (e.g. IP address option and FQDN option) is
also valid.

If the responding node itself does not support the discovery
objective, but it knows the locator of the discovery objective, then
it SHOULD respond to the discovery message with a divert option
(Section 3.9.2) embedding a locator option or a combination of
multiple kinds of locator options which indicate the locator(s) of
the discovery objective.

More details on the processing of Discovery Responses are given in
Section 3.5.4.

3.8.6. Request Messages

In fragmentary CDDL, Request Negotiation and Request Synchronization
messages follow the patterns:

request-negotiation-message = [M_REQ_NEG, session-id, objective]

request-synchronization-message = [M_REQ_SYN, session-id, objective]

A negotiation or synchronization requesting node sends the
appropriate Request message to the unicast address (directly stored
or resolved from an FQDN or URI) of the negotiation or
synchronization counterpart, using the appropriate protocol and port
numbers (selected from the discovery results).

A Request message MUST include the relevant objective option. In the
case of Request Negotiation, the objective option MUST include the
requested value.

When an initiator sends a Request Negotiation message, it MUST
initialize a negotiation timer for the new negotiation thread. The
default is GRASP_DEF_TIMEOUT milliseconds. Unless this timeout is
modified by a Confirm Waiting message (Section 3.8.9), the initiator
will consider that the negotiation has failed when the timer expires.

Similarly, when an initiator sends a Request Synchronization, it
SHOULD initialize a synchronization timer. The default is
GRASP_DEF_TIMEOUT milliseconds. The initiator will consider that
synchronization has failed if there is no response before the timer
expires.

When an initiator sends a Request message, it MUST initialize the
loop count of the objective option with a value defined in the
specification of the option or, if no such value is specified, with
GRASP_DEF_LOOPCT.

If a node receives a Request message for an objective for which no
ASA is currently listening, it MUST immediately close the relevant
socket to indicate this to the initiator. This is to avoid
unnecessary timeouts if, for example, an ASA exits prematurely but
the GRASP core is listening on its behalf.

To avoid the highly unlikely race condition in which two nodes
simultaneously request sessions with each other using the same
Session ID (Section 3.7), when a node receives a Request message, it
MUST verify that the received Session ID is not already locally
active. In case of a clash, it MUST discard the Request message, in
which case the initiator will detect a timeout.

3.8.7. Negotiation Message

In fragmentary CDDL, a Negotiation message follows the pattern:

```
negotiate-message = [M_NEGOTIATE, session-id, objective]
```

A negotiation counterpart sends a Negotiation message in response to
a Request Negotiation message, a Negotiation message, or a Discovery
message in Rapid Mode. A negotiation process MAY include multiple
steps.

The Negotiation message MUST include the relevant Negotiation
Objective option, with its value updated according to progress in the
negotiation. The sender MUST decrement the loop count by 1. If the
loop count becomes zero the message MUST NOT be sent. In this case
the negotiation session has failed and will time out.

3.8.8. Negotiation End Message

In fragmentary CDDL, a Negotiation End message follows the pattern:

```
end-message = [M_END, session-id, accept-option / decline-option]
```

A negotiation counterpart sends an Negotiation End message to close
the negotiation. It MUST contain either an accept or a decline
option, defined in Section 3.9.3 and Section 3.9.4. It could be sent
either by the requesting node or the responding node.

3.8.9. Confirm Waiting Message

In fragmentary CDDL, a Confirm Waiting message follows the pattern:

```
wait-message = [M_WAIT, session-id, waiting-time]
waiting-time =        0..4294967295 ; in milliseconds
```

A responding node sends a Confirm Waiting message to ask the
requesting node to wait for a further negotiation response. It might
be that the local process needs more time or that the negotiation
depends on another triggered negotiation. This message MUST NOT
include any other options. When received, the waiting time value

overwrites and restarts the current negotiation timer
(Section 3.8.6).

The responding node SHOULD send a Negotiation, Negotiation End or
another Confirm Waiting message before the negotiation timer expires.
If not, the initiator MUST abandon or restart the negotiation
procedure, to avoid an indefinite wait.

3.8.10. Synchronization Message

In fragmentary CDDL, a Synchronization message follows the pattern:

```
synch-message = [M_SYNCH, session-id, objective]
```

A node which receives a Request Synchronization, or a Discovery
message in Rapid Mode, sends back a unicast Synchronization message
with the synchronization data, in the form of a GRASP Option for the
specific synchronization objective present in the Request
Synchronization.

3.8.11. Flood Synchronization Message

In fragmentary CDDL, a Flood Synchronization message follows the
pattern:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]

ttl = 0..4294967295 ; in milliseconds
```

A node MAY initiate flooding by sending an unsolicited Flood
Synchronization Message with synchronization data. This MAY be sent
to the link-local ALL_GRASP_NEIGHBOR multicast address, in accordance
with the rules in Section 3.5.6.

The initiator address is provided, as described for Discovery
messages (Section 3.8.4), only to disambiguate the Session ID.

The message MUST contain a time-to-live (ttl) for the validity of
the contents, given as a positive integer value in milliseconds.
There is no default; zero indicates an indefinite lifetime.

The synchronization data are in the form of GRASP Option(s) for
specific synchronization objective(s). The loop count(s) MUST be
set to a suitable value to prevent flood loops (default value is
GRASP_DEF_LOOPCT).

Each objective option MAY be followed by a locator option
associated with the flooded objective. In its absence, an empty
option MUST be included to indicate a null locator.

A node that receives a Flood Synchronization message MUST cache the
received objectives for use by local ASAs. Each cached objective
MUST be tagged with the locator option sent with it, or with a null
tag if an empty locator option was sent. If a subsequent Flood
Synchronization message carrying the same objective arrives with the
same tag, the corresponding cached copy of the objective MUST be
overwritten. If a subsequent Flood Synchronization message carrying
the same objective arrives with a different tag, a new cached entry
MUST be created.

Note: the purpose of this mechanism is to allow the recipient of
flooded values to distinguish between different senders of the same
objective, and if necessary communicate with them using the locator,
protocol and port included in the locator option. Many objectives
will not need this mechanism, so they will be flooded with a null
locator.

Cached entries MUST be ignored or deleted after their lifetime
expires.

3.8.12. Invalid Message

In fragmentary CDDL, an Invalid message follows the pattern:

    invalid-message = [M_INVALID, session-id, ?any]

This message MAY be sent by an implementation in response to an
incoming message that it considers invalid. The session-id MUST be
copied from the incoming message. The content SHOULD be diagnostic
information such as a partial copy of the invalid message. An
M_INVALID message MAY be silently ignored by a recipient. However,
it could be used in support of extensibility, since it indicates that
the remote node does not support a new or obsolete message or option

An M_INVALID message MUST NOT be sent in response to an M_INVALID
message.

3.8.13. No Operation Message

In fragmentary CDDL, a No Operation message follows the pattern:

    noop-message = [M_NOOP]

This message MAY be sent by an implementation that for practical reasons needs to activate a socket. It MUST be silently ignored by a recipient.

3.9. GRASP Options

This section defines the GRASP options for the negotiation and synchronization protocol signaling. Additional options may be defined in the future.

3.9.1. Format of GRASP Options

GRASP options are CBOR objects that MUST start with an unsigned integer identifying the specific option type carried in this option. These option types are formally defined in Section 6. Apart from that the only format requirement is that each option MUST be a well-formed CBOR object. In general a CBOR array format is RECOMMENDED to limit overhead.

GRASP options are usually scoped by using encapsulation. However, this is not a requirement

3.9.2. Divert Option

The Divert option is used to redirect a GRASP request to another node, which may be more appropriate for the intended negotiation or synchronization. It may redirect to an entity that is known as a specific negotiation or synchronization counterpart (on-link or off-link) or a default gateway. The divert option MUST only be encapsulated in Discovery Response messages. If found elsewhere, it SHOULD be silently ignored.

A discovery initiator MAY ignore a Divert option if it only requires direct discovery responses.

In fragmentary CDDL, the Divert option follows the pattern:

```
divert-option = [O_DIVERT, +locator-option]
```

The embedded Locator Option(s) (Section 3.9.5) point to diverted destination target(s) in response to a Discovery message.

3.9.3. Accept Option

The accept option is used to indicate to the negotiation counterpart that the proposed negotiation content is accepted.

The accept option MUST only be encapsulated in Negotiation End
messages. If found elsewhere, it SHOULD be silently ignored.

In fragmentary CDDL, the Accept option follows the pattern:

```
accept-option = [O_ACCEPT]
```

3.9.4. Decline Option

The decline option is used to indicate to the negotiation counterpart
the proposed negotiation content is declined and end the negotiation
process.

The decline option MUST only be encapsulated in Negotiation End
messages. If found elsewhere, it SHOULD be silently ignored.

In fragmentary CDDL, the Decline option follows the pattern:

```
decline-option = [O_DECLINE, ?reason]
reason = text   ;optional error message
```

Note: there are scenarios where a negotiation counterpart wants to
decline the proposed negotiation content and continue the negotiation
process. For these scenarios, the negotiation counterpart SHOULD use
a Negotiate message, with either an objective option that contains a
data field set to indicate a meaningless initial value, or a specific
objective option that provides further conditions for convergence.

3.9.5. Locator Options

These locator options are used to present reachability information
for an ASA, a device or an interface. They are Locator IPv6 Address
Option, Locator IPv4 Address Option, Locator FQDN (Fully Qualified
Domain Name) Option and URI (Uniform Resource Identifier) Option.

Since ASAs will normally run as independent user programs, locator
options need to indicate the network layer locator plus the transport
protocol and port number for reaching the target. For this reason,
the Locator Options for IP addresses and FQDNs include this
information explicitly. In the case of the URI Option, this
information can be encoded in the URI itself.

Note: It is assumed that all locators are in scope throughout the
GRASP domain. GRASP is not intended to work across disjoint
addressing or naming realms.

3.9.5.1. Locator IPv6 address option

In fragmentary CDDL, the IPv6 address option follows the pattern:

```
   ipv6-locator-option = [O_IPv6_LOCATOR, ipv6-address,
                           transport-proto, port-number]
   ipv6-address = bytes .size 16

   transport-proto = IPPROTO_TCP / IPPROTO_UDP
   IPPROTO_TCP = 6
   IPPROTO_UDP = 17
   port-number = 0..65535
```

The content of this option is a binary IPv6 address followed by the
protocol number and port number to be used.

Note 1: The IPv6 address MUST normally have global scope.
Exceptionally, during initialisation, a link-local address MAY be
used for specific objectives only (Section 3.5.2). In this case the
corresponding Discovery Response message MUST be sent via the
interface to which the link-local address applies.

Note 2: A link-local IPv6 address MUST NOT be used when this option
is included in a Divert option.

3.9.5.2. Locator IPv4 address option

In fragmentary CDDL, the IPv4 address option follows the pattern:

```
   ipv4-locator-option = [O_IPv4_LOCATOR, ipv4-address,
                          transport-proto, port-number]
   ipv4-address = bytes .size 4
```

The content of this option is a binary IPv4 address followed by the
protocol number and port number to be used.

Note: If an operator has internal network address translation for
IPv4, this option MUST NOT be used within the Divert option.

3.9.5.3. Locator FQDN option

In fragmentary CDDL, the FQDN option follows the pattern:

```
   fqdn-locator-option = [O_FQDN_LOCATOR, text,
                          transport-proto, port-number]
```

The content of this option is the Fully Qualified Domain Name of the
target followed by the protocol number and port number to be used.

Note 1: Any FQDN which might not be valid throughout the network in
question, such as a Multicast DNS name [RFC6762], MUST NOT be used
when this option is used within the Divert option.

Note 2: Normal GRASP operations are not expected to use this option.
It is intended for special purposes such as discovering external
services.

3.9.5.4. Locator URI option

In fragmentary CDDL, the URI option follows the pattern:

```
uri-locator = [O_URI_LOCATOR, text]
```

The content of this option is the Uniform Resource Identifier of the
target [RFC3986].

Note 1: Any URI which might not be valid throughout the network in
question, such as one based on a Multicast DNS name [RFC6762], MUST
NOT be used when this option is used within the Divert option.

Note 2: Normal GRASP operations are not expected to use this option.
It is intended for special purposes such as discovering external
services.

3.10. Objective Options

3.10.1. Format of Objective Options

An objective option is used to identify objectives for the purposes
of discovery, negotiation or synchronization. All objectives MUST be
in the following format, described in fragmentary CDDL:

```
objective = [objective-name, objective-flags, loop-count, ?any]

objective-name = text
loop-count = 0..255
```

All objectives are identified by a unique name which is a case-
sensitive UTF-8 string.

The names of generic objectives MUST NOT include a colon (":") and
MUST be registered with IANA (Section 7).

The names of privately defined objectives MUST include at least one
colon (":"). The string preceding the last colon in the name MUST be
globally unique and in some way identify the entity or person

defining the objective. The following three methods MAY be used to
create such a globally unique string:

1. The unique string is a decimal number representing a registered
    32 bit Private Enterprise Number (PEN) [I-D.liang-iana-pen] that
    uniquely identifies the enterprise defining the objective.

2. The unique string is a fully qualified domain name that uniquely
    identifies the entity or person defining the objective.

3. The unique string is an email address that uniquely identifies
    the entity or person defining the objective.

The GRASP protocol treats the objective name as an opaque string.
For example, "EX1", "411:EX1", "example.com:EX1", "example.org:EX1
and "user@example.org:EX1" would be five different objectives.

The 'objective-flags' field is described below.

The 'loop-count' field is used for terminating negotiation as
described in Section 3.8.7. It is also used for terminating
discovery as described in Section 3.5.4, and for terminating flooding
as described in Section 3.5.6.1.

The 'any' field is to express the actual value of a negotiation or
synchronization objective. Its format is defined in the
specification of the objective and may be a single value or a data
structure of any kind. It is optional because it is optional in a
Discovery or Discovery Response message.

3.10.2. Objective flags

An objective may be relevant for discovery only, for discovery and
negotiation, or for discovery and synchronization. This is expressed
in the objective by logical flag bits:

```
  objective-flags = uint .bits objective-flag
  objective-flag = &(
  F_DISC: 0    ; valid for discovery
  F_NEG: 1     ; valid for negotiation
  F_SYNCH: 2   ; valid for synchronization
  F_NEG_DRY: 3 ; negotiation is dry-run
  )
```

These bits are independent and may be combined appropriately, e.g.
(F_DISC and F_SYNCH) or (F_DISC and F_NEG) or (F_DISC and F_NEG and
F_NEG_DRY).

Note that for a given negotiation session, an objective must be
either used for negotiation, or for dry-run negotiation. Mixing the
two modes in a single negotiation is not possible.

3.10.3. General Considerations for Objective Options

As mentioned above, Objective Options MUST be assigned a unique name.
As long as privately defined Objective Options obey the rules above,
this document does not restrict their choice of name, but the entity
or person concerned SHOULD publish the names in use.

All Objective Options MUST respect the CBOR patterns defined above as
"objective" and MUST replace the "any" field with a valid CBOR data
definition for the relevant use case and application.

An Objective Option that contains no additional fields beyond its
"loop-count" can only be a discovery objective and MUST only be used
in Discovery and Discovery Response messages.

The Negotiation Objective Options contain negotiation objectives,
which vary according to different functions/services. They MUST be
carried by Discovery, Request Negotiation or Negotiation messages
only. The negotiation initiator MUST set the initial "loop-count" to
a value specified in the specification of the objective or, if no
such value is specified, to GRASP_DEF_LOOPCT.

For most scenarios, there should be initial values in the negotiation
requests. Consequently, the Negotiation Objective options MUST
always be completely presented in a Request Negotiation message, or
in a Discovery message in rapid mode. If there is no initial value,
the bits in the value field SHOULD all be set to indicate a
meaningless value, unless this is inappropriate for the specific
negotiation objective.

Synchronization Objective Options are similar, but MUST be carried by
Discovery, Discovery Response, Request Synchronization, or Flood
Synchronization messages only. They include value fields only in
Synchronization or Flood Synchronization messages.

3.10.4. Organizing of Objective Options

Generic objective options MUST be specified in documents available to
the public and SHOULD be designed to use either the negotiation or
the synchronization mechanism described above.

As noted earlier, one negotiation objective is handled by each GRASP
negotiation thread. Therefore, a negotiation objective, which is
based on a specific function or action, SHOULD be organized as a

single GRASP option. It is NOT RECOMMENDED to organize multiple
negotiation objectives into a single option, nor to split a single
function or action into multiple negotiation objectives.

It is important to understand that GRASP negotiation does not support
transactional integrity. If transactional integrity is needed for a
specific objective, this must be ensured by the ASA. For example, an
ASA might need to ensure that it only participates in one negotiation
thread at the same time. Such an ASA would need to stop listening
for incoming negotiation requests before generating an outgoing
negotiation request.

A synchronization objective SHOULD be organized as a single GRASP
option.

Some objectives will support more than one operational mode. An
example is a negotiation objective with both a "dry run" mode (where
the negotiation is to find out whether the other end can in fact make
the requested change without problems) and a "live" mode. Such modes
will be defined in the specification of such an objective. These
objectives SHOULD include flags indicating the applicable mode(s).

An objective may have multiple parameters. Parameters can be
categorized into two classes: the obligatory ones presented as fixed
fields; and the optional ones presented in CBOR sub-options or some
other form of data structure embedded in CBOR. The format might be
inherited from an existing management or configuration protocol, the
objective option acting as a carrier for that format. The data
structure might be defined in a formal language, but that is a matter
for the specifications of individual objectives. There are many
candidates, according to the context, such as ABNF, RBNF, XML Schema,
possibly YANG, etc. The GRASP protocol itself is agnostic on these
questions.

It is NOT RECOMMENDED to split parameters in a single objective into
multiple options, unless they have different response periods. An
exception scenario may also be described by split objectives.

All objectives MUST support GRASP discovery. However, as mentioned
in Section 3.3, it is acceptable for an ASA to use an alternative
method of discovery.

Normally, a GRASP objective will refer to specific technical
parameters as explained in Section 3.1. However, it is acceptable to
define an abstract objective for the purpose of managing or
coordinating ASAs. It is also acceptable to define a special-purpose
objective for purposes such as trust bootstrapping or formation of
the ACP.

To guarantee convergence, a limited number of rounds or a timeout is
needed for each negotiation objective. Therefore, the definition of
each negotiation objective SHOULD clearly specify this, for example a
default loop count and timeout, so that the negotiation can always be
terminated properly. If not, the GRASP defaults will apply.

There must be a well-defined procedure for concluding that a
negotiation cannot succeed, and if so deciding what happens next
(e.g., deadlock resolution, tie-breaking, or revert to best-effort
service). This MUST be specified for individual negotiation
objectives.

3.10.5. Experimental and Example Objective Options

The names "EX0" through "EX9" have been reserved for experimental
options. Multiple names have been assigned because a single
experiment may use multiple options simultaneously. These
experimental options are highly likely to have different meanings
when used for different experiments. Therefore, they SHOULD NOT be
used without an explicit human decision and SHOULD NOT be used in
unmanaged networks such as home networks.

These names are also RECOMMENDED for use in documentation examples.

4. Implementation Status [RFC Editor: please remove]

Two prototype implementations of GRASP have been made.

4.1. BUPT C++ Implementation

o   Name: BaseNegotiator.cpp, msg.cpp, Client.cpp, Server.cpp

o   Description: C++ implementation of GRASP core and API

o   Maturity: Prototype code, interoperable between Ubuntu.

o   Coverage: Corresponds to draft-carpenter-anima-gdn-protocol-03.
    Since it was implemented based on the old version draft, the most
    significant limitations comparing to current protocol design
    include:

    *   Not support CBOR

    *   Not support Flooding

    *   Not support loop avoidance

    *   only coded for IPv6, any IPv4 is accidental

o Licensing: Huawei License.

o Experience: https://github.com/liubingpang/IETF-Anima-Signaling-Protocol/blob/master/README.md

o Contact: https://github.com/liubingpang/IETF-Anima-Signaling-Protocol

4.2. Python Implementation

o Name: graspy

o Description: Python 3 implementation of GRASP core and API.

o Maturity: Prototype code, interoperable between Windows 7 and Linux.

o Coverage: Corresponds to draft-ietf-anima-grasp-08. Limitations include:

   * insecure: uses a dummy ACP module and does not implement TLS

   * only coded for IPv6, any IPv4 is accidental

   * FQDN and URI locators incompletely supported

   * no code for rapid mode

   * relay code is lazy (no rate control)

   * all unicast transactions use TCP (no unicast UDP).
      Experimental code for unicast UDP proved to be complex and
      brittle.

   * optional Objective option in Response messages not implemented

   * workarounds for defects in Python socket module and Windows
      socket peculiarities

o Licensing: Simplified BSD

o Experience: https://www.cs.auckland.ac.nz/~brian/graspy/graspy.pdf

o Contact: https://www.cs.auckland.ac.nz/~brian/graspy/

5. Security Considerations

   It is obvious that a successful attack on negotiation-enabled nodes
   would be extremely harmful, as such nodes might end up with a
   completely undesirable configuration that would also adversely affect
   their peers. GRASP nodes and messages therefore require full
   protection.

   - Authentication

     A cryptographically authenticated identity for each device is
     needed in an autonomic network. It is not safe to assume that a
     large network is physically secured against interference or that
     all personnel are trustworthy. Each autonomic node MUST be
     capable of proving its identity and authenticating its messages.
     GRASP relies on a separate external certificate-based security
     mechanism to support authentication, data integrity protection,
     and anti-replay protection.

     Since GRASP is intended to be deployed in a single administrative
     domain operating its own trust anchor and CA, there is no need for
     a trusted public third party. In a network requiring "air gap"
     security, such a dependency would be unacceptable.

     If GRASP is used temporarily without an external security
     mechanism, for example during system bootstrap (Section 3.5.1),
     the Session ID (Section 3.7) will act as a nonce to provide
     limited protection against third parties injecting responses. A
     full analysis of the secure bootstrap process is out of scope for
     the present document.

   - Authorization and Roles

     The GRASP protocol is agnostic about the role of individual ASAs
     and about which objectives a particular ASA is authorized to
     support. An implementation might support precautions such as
     allowing only one ASA in a given node to modify a given objective,
     but this may not be appropriate in all cases. For example, it
     might be operationally useful to allow an old and a new version of
     the same ASA to run simultaneously during an overlap period.
     These questions are out of scope for the present specification.

   - Privacy and confidentiality

     Generally speaking, no personal information is expected to be
     involved in the signaling protocol, so there should be no direct
     impact on personal privacy. Nevertheless, traffic flow paths,
     VPNs, etc. could be negotiated, which could be of interest for

traffic analysis. Also, operators generally want to conceal
details of their network topology and traffic density from
outsiders. Therefore, since insider attacks cannot be excluded in
a large network, the security mechanism for the protocol MUST
provide message confidentiality. This is why Section 3.5.1
requires either an ACP or an alternative security mechanism.

- Link-local multicast security

GRASP has no reasonable alternative to using link-local multicast
for Discovery or Flood Synchronization messages and these messages
are sent in clear and with no authentication. They are therefore
available to on-link eavesdroppers, and could be forged by on-link
attackers. In the case of Discovery, the Discovery Responses are
unicast and will therefore be protected (Section 3.5.1), and an
untrusted forger will not be able to receive responses. In the
case of Flood Synchronization, an on-link eavesdropper will be
able to receive the flooded objectives but there is no response
message to consider. Some precautions for Flood Synchronization
messages are suggested in Section 3.5.6.1.

- DoS Attack Protection

GRASP discovery partly relies on insecure link-local multicast.
Since routers participating in GRASP sometimes relay discovery
messages from one link to another, this could be a vector for
denial of service attacks. Some mitigations are specified in
Section 3.5.4. However, malicious code installed inside the
Autonomic Control Plane could always launch DoS attacks consisting
of spurious discovery messages, or of spurious discovery
responses. Additionally, it is of great importance that firewalls
prevent any GRASP messages from entering the domain from an
untrusted source.

- Security during bootstrap and discovery

A node cannot authenticate GRASP traffic from other nodes until it
has identified the trust anchor and can validate certificates for
other nodes. Also, until it has succesfully enrolled
[I-D.ietf-anima-bootstrapping-keyinfra] it cannot assume that
other nodes are able to authenticate its own traffic. Therefore,
GRASP discovery during the bootstrap phase for a new device will
inevitably be insecure and GRASP synchronization and negotiation
will be impossible until enrollment is complete. Further details
are given in Section 3.5.2.

- Security of discovered locators

When GRASP discovery returns an IP address, it MUST be that of a
node within the secure environment ([Section 3.5.1](#)). If it returns
an FQDN or a URI, the ASA that receives it MUST NOT assume that
the target of the locator is within the secure environment.

[6](#). CDDL Specification of GRASP

```
<CODE BEGINS>
grasp-message = (message .within message-structure) / noop-message

message-structure = [MESSAGE_TYPE, session-id, ?initiator,
                        *grasp-option]

MESSAGE_TYPE = 0..255
session-id = 0..4294967295 ;up to 32 bits
grasp-option = any

message /= discovery-message
discovery-message = [M_DISCOVERY, session-id, initiator, objective]

message /= response-message ;response to Discovery
response-message = [M_RESPONSE, session-id, initiator, ttl,
                     (+locator-option // divert-option), ?objective]

message /= synch-message ;response to Synchronization request
synch-message = [M_SYNCH, session-id, objective]

message /= flood-message
flood-message = [M_FLOOD, session-id, initiator, ttl,
                  +[objective, (locator-option / [])]]

message /= request-negotiation-message
request-negotiation-message = [M_REQ_NEG, session-id, objective]

message /= request-synchronization-message
request-synchronization-message = [M_REQ_SYN, session-id, objective]

message /= negotiation-message
negotiation-message = [M_NEGOTIATE, session-id, objective]

message /= end-message
end-message = [M_END, session-id, accept-option / decline-option ]

message /= wait-message
wait-message = [M_WAIT, session-id, waiting-time]

message /= invalid-message
invalid-message = [M_INVALID, session-id, ?any]
```

```
noop-message = [M_NOOP]

divert-option = [O_DIVERT, +locator-option]

accept-option = [O_ACCEPT]

decline-option = [O_DECLINE, ?reason]
reason = text  ;optional error message

waiting-time = 0..4294967295 ; in milliseconds
ttl = 0..4294967295 ; in milliseconds

locator-option /= [O_IPv4_LOCATOR, ipv4-address,
                  transport-proto, port-number]
ipv4-address = bytes .size 4

locator-option /= [O_IPv6_LOCATOR, ipv6-address,
                  transport-proto, port-number]
ipv6-address = bytes .size 16

locator-option /= [O_FQDN_LOCATOR, text, transport-proto, port-number]

transport-proto = IPPROTO_TCP / IPPROTO_UDP
IPPROTO_TCP = 6
IPPROTO_UDP = 17
port-number = 0..65535

locator-option /= [O_URI_LOCATOR, text]

initiator = ipv4-address / ipv6-address

objective-flags = uint .bits objective-flag

objective-flag = &(
  F_DISC: 0    ; valid for discovery
  F_NEG: 1     ; valid for negotiation
  F_SYNCH: 2   ; valid for synchronization
  F_NEG_DRY: 3 ; negotiation is dry-run
)

objective = [objective-name, objective-flags, loop-count, ?any]

objective-name = text ;see specification for uniqueness rules

loop-count = 0..255

; Constants for message types and option types
```

```
  M_NOOP = 0
  M_DISCOVERY = 1
  M_RESPONSE = 2
  M_REQ_NEG = 3
  M_REQ_SYN = 4
  M_NEGOTIATE = 5
  M_END = 6
  M_WAIT = 7
  M_SYNCH = 8
  M_FLOOD = 9
  M_INVALID = 99

  O_DIVERT = 100
  O_ACCEPT = 101
  O_DECLINE = 102
  O_IPv6_LOCATOR = 103
  O_IPv4_LOCATOR = 104
  O_FQDN_LOCATOR = 105
  O_URI_LOCATOR = 106
  <CODE ENDS>
```

7. IANA Considerations

   This document defines the Generic Autonomic Signaling Protocol
   (GRASP).

   Section 3.6 explains the following link-local multicast addresses,
   which IANA is requested to assign for use by GRASP:

   ALL_GRASP_NEIGHBOR multicast address (IPv6): (TBD1). Assigned in
      the IPv6 Link-Local Scope Multicast Addresses registry.

   ALL_GRASP_NEIGHBOR multicast address (IPv4): (TBD2). Assigned in
      the IPv4 Multicast Local Network Control Block.

   Section 3.6 explains the following User Port, which IANA is requested
   to assign for use by GRASP for both UDP and TCP:

   GRASP_LISTEN_PORT: (TBD3)
   Service Name: Generic Autonomic Signaling Protocol (GRASP)
   Transport Protocols: UDP, TCP
   Assignee: iesg@ietf.org
   Contact: chair@ietf.org
   Description: See Section 3.6
   Reference: RFC XXXX (this document)

The IANA is requested to create a GRASP Parameter Registry including
two registry tables. These are the GRASP Messages and Options
Table and the GRASP Objective Names Table.

GRASP Messages and Options Table. The values in this table are names
paired with decimal integers. Future values MUST be assigned using
the Standards Action policy defined by [RFC5226]. The following
initial values are assigned by this document:

M_NOOP = 0
M_DISCOVERY = 1
M_RESPONSE = 2
M_REQ_NEG = 3
M_REQ_SYN = 4
M_NEGOTIATE = 5
M_END = 6
M_WAIT = 7
M_SYNCH = 8
M_FLOOD = 9
M_INVALID = 99

O_DIVERT = 100
O_ACCEPT = 101
O_DECLINE = 102
O_IPv6_LOCATOR = 103
O_IPv4_LOCATOR = 104
O_FQDN_LOCATOR = 105
O_URI_LOCATOR = 106

GRASP Objective Names Table. The values in this table are UTF-8
strings. Future values MUST be assigned using the Specification
Required policy defined by [RFC5226]. The following initial values
are assigned by this document:

 EX0
 EX1
 EX2
 EX3
 EX4
 EX5
 EX6
 EX7
 EX8
 EX9

8. Acknowledgements

A major contribution to the original version of this document was
made by Sheng Jiang. Significant review inputs were received from
Joel Halpern, Toerless Eckert and Michael Richardson.

Valuable comments were received from Michael Behringer, Jeferson
Campos Nobre, Laurent Ciavaglia, Zongpeng Du, Yu Fu, Zhenbin Li,
Dimitri Papadimitriou, Pierre Peloso, Reshad Rahman, Markus Stenberg,
Rene Struik, Dacheng Zhang, and other participants in the NMRG
research group and the ANIMA working group.

9. References

9.1. Normative References

[I-D.greevenbosch-appsawg-cbor-cddl]
          Vigano, C. and H. Birkholz, "CBOR data definition language
          (CDDL): a notational convention to express CBOR data
          structures", draft-greevenbosch-appsawg-cbor-cddl-09 (work
          in progress), September 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119,
          DOI 10.17487/RFC2119, March 1997,
          <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
          Resource Identifier (URI): Generic Syntax", STD 66,
          RFC 3986, DOI 10.17487/RFC3986, January 2005,
          <http://www.rfc-editor.org/info/rfc3986>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker,
          "Randomness Requirements for Security", BCP 106, RFC 4086,
          DOI 10.17487/RFC4086, June 2005,
          <http://www.rfc-editor.org/info/rfc4086>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
          (TLS) Protocol Version 1.2", RFC 5246,
          DOI 10.17487/RFC5246, August 2008,
          <http://www.rfc-editor.org/info/rfc5246>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
          Housley, R., and W. Polk, "Internet X.509 Public Key
          Infrastructure Certificate and Certificate Revocation List
          (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
          <http://www.rfc-editor.org/info/rfc5280>.

   [RFC6347]   Rescorla, E. and N. Modadugu, "Datagram Transport Layer
               Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
               January 2012, <http://www.rfc-editor.org/info/rfc6347>.

   [RFC7049]   Bormann, C. and P. Hoffman, "Concise Binary Object
               Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,
               October 2013, <http://www.rfc-editor.org/info/rfc7049>.

   [RFC7217]   Gont, F., "A Method for Generating Semantically Opaque
               Interface Identifiers with IPv6 Stateless Address
               Autoconfiguration (SLAAC)", RFC 7217,
               DOI 10.17487/RFC7217, April 2014,
               <http://www.rfc-editor.org/info/rfc7217>.

9.2. Informative References

   [I-D.chaparadza-intarea-igcp]
               Behringer, M., Chaparadza, R., Petre, R., Li, X., and H.
               Mahkonen, "IP based Generic Control Protocol (IGCP)",
               draft-chaparadza-intarea-igcp-00 (work in progress), July
               2011.

   [I-D.ietf-anima-autonomic-control-plane]
               Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic
               Control Plane", draft-ietf-anima-autonomic-control-
               plane-04 (work in progress), October 2016.

   [I-D.ietf-anima-bootstrapping-keyinfra]
               Pritikin, M., Richardson, M., Behringer, M., Bjarnason,
               S., and K. Watsen, "Bootstrapping Remote Secure Key
               Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-
               keyinfra-04 (work in progress), October 2016.

   [I-D.ietf-anima-reference-model]
               Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L.,
               Pierre, P., Liu, B., Nobre, J., and J. Strassner, "A
               Reference Model for Autonomic Networking", draft-ietf-
               anima-reference-model-02 (work in progress), July 2016.

   [I-D.ietf-anima-stable-connectivity]
               Eckert, T. and M. Behringer, "Using Autonomic Control
               Plane for Stable Connectivity of Network OAM", draft-ietf-
               anima-stable-connectivity-01 (work in progress), July
               2016.

   [I-D.ietf-netconf-restconf]
              Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", draft-ietf-netconf-restconf-18 (work in
              progress), October 2016.

   [I-D.liang-iana-pen]
              Liang, P., Melnikov, A., and D. Conrad, "Private
              Enterprise Number (PEN) practices and Internet Assigned
              Numbers Authority (IANA) registration considerations",
              draft-liang-iana-pen-06 (work in progress), July 2015.

   [I-D.liu-anima-grasp-api]
              Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic
              Autonomic Signaling Protocol Application Program Interface
              (GRASP API)", draft-liu-anima-grasp-api-02 (work in
              progress), September 2016.

   [I-D.stenberg-anima-adncp]
              Stenberg, M., "Autonomic Distributed Node Consensus
              Protocol", draft-stenberg-anima-adncp-00 (work in
              progress), March 2015.

   [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S.
              Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1
              Functional Specification", RFC 2205, DOI 10.17487/RFC2205,
              September 1997, <http://www.rfc-editor.org/info/rfc2205>.

   [RFC2334] Luciani, J., Armitage, G., Halpern, J., and N. Doraswamy,
              "Server Cache Synchronization Protocol (SCSP)", RFC 2334,
              DOI 10.17487/RFC2334, April 1998,
              <http://www.rfc-editor.org/info/rfc2334>.

   [RFC2608] Guttman, E., Perkins, C., Veizades, J., and M. Day,
              "Service Location Protocol, Version 2", RFC 2608,
              DOI 10.17487/RFC2608, June 1999,
              <http://www.rfc-editor.org/info/rfc2608>.

   [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson,
              "Remote Authentication Dial In User Service (RADIUS)",
              RFC 2865, DOI 10.17487/RFC2865, June 2000,
              <http://www.rfc-editor.org/info/rfc2865>.

   [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
              and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP
              Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001,
              <http://www.rfc-editor.org/info/rfc3209>.

   [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
              C., and M. Carney, "Dynamic Host Configuration Protocol
              for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
              2003, <http://www.rfc-editor.org/info/rfc3315>.

   [RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations
              for the Simple Network Management Protocol (SNMP)",
              STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002,
              <http://www.rfc-editor.org/info/rfc3416>.

   [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman,
              "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
              DOI 10.17487/RFC4861, September 2007,
              <http://www.rfc-editor.org/info/rfc4861>.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 5226,
              DOI 10.17487/RFC5226, May 2008,
              <http://www.rfc-editor.org/info/rfc5226>.

   [RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet
              Signalling Transport", RFC 5971, DOI 10.17487/RFC5971,
              October 2010, <http://www.rfc-editor.org/info/rfc5971>.

   [RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko,
              "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206,
              March 2011, <http://www.rfc-editor.org/info/rfc6206>.

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
              and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <http://www.rfc-editor.org/info/rfc6241>.

   [RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn,
              Ed., "Diameter Base Protocol", RFC 6733,
              DOI 10.17487/RFC6733, October 2012,
              <http://www.rfc-editor.org/info/rfc6733>.

   [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762,
              DOI 10.17487/RFC6762, February 2013,
              <http://www.rfc-editor.org/info/rfc6762>.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <http://www.rfc-editor.org/info/rfc6763>.

   [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and
              P. Selkirk, "Port Control Protocol (PCP)", RFC 6887,
              DOI 10.17487/RFC6887, April 2013,
              <http://www.rfc-editor.org/info/rfc6887>.

   [RFC7558] Lynn, K., Cheshire, S., Blanchet, M., and D. Migault,
              "Requirements for Scalable DNS-Based Service Discovery
              (DNS-SD) / Multicast DNS (mDNS) Extensions", RFC 7558,
              DOI 10.17487/RFC7558, July 2015,
              <http://www.rfc-editor.org/info/rfc7558>.

   [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>.

   [RFC7787] Stenberg, M. and S. Barth, "Distributed Node Consensus
              Protocol", RFC 7787, DOI 10.17487/RFC7787, April 2016,
              <http://www.rfc-editor.org/info/rfc7787>.

   [RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking
              Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April
              2016, <http://www.rfc-editor.org/info/rfc7788>.

Appendix A. Open Issues [RFC Editor: Please remove if empty]

   o 63. Placeholder

Appendix B. Closed Issues [RFC Editor: Please remove]

   o 1. UDP vs TCP: For now, this specification suggests UDP and TCP
      as message transport mechanisms. This is not clarified yet. UDP
      is good for short conversations, is necessary for multicast
      discovery, and generally fits the discovery and divert scenarios
      well. However, it will cause problems with large messages. TCP
      is good for stable and long sessions, with a little bit of time
      consumption during the session establishment stage. If messages
      exceed a reasonable MTU, a TCP mode will be required in any case.
      This question may be affected by the security discussion.

      RESOLVED by specifying UDP for short message and TCP for longer
      one.

o 2. DTLS or TLS vs built-in security mechanism. For now, this
       specification has chosen a PKI based built-in security mechanism
       based on asymmetric cryptography. However, (D)TLS might be chosen
       as security solution to avoid duplication of effort. It also
       allows essentially similar security for short messages over UDP
       and longer ones over TCP. The implementation trade-offs are
       different. The current approach requires expensive asymmetric
       cryptographic calculations for every message. (D)TLS has startup
       overheads but cheaper crypto per message. DTLS is less mature
       than TLS.

       RESOLVED by specifying external security (ACP or (D)TLS).

o The following open issues applied only if the original security
   model was retained:

   * 2.1. For replay protection, GRASP currently requires every
      participant to have an NTP-synchronized clock. Is this OK for
      low-end devices, and how does it work during device
      bootstrapping? We could take the Timestamp out of signature
      option, to become an independent and OPTIONAL (or RECOMMENDED)
      option.

   * 2.2. The Signature Option states that this option could be any
      place in a message. Wouldn't it be better to specify a
      position (such as the end)? That would be much simpler to
      implement.

       RESOLVED by changing security model.

o 3. DoS Attack Protection needs work.

       RESOLVED by adding text.

o 4. Should we consider preferring a text-based approach to
       discovery (after the initial discovery needed for bootstrapping)?
       This could be a complementary mechanism for multicast based
       discovery, especially for a very large autonomic network.
       Centralized registration could be automatically deployed
       incrementally. At the very first stage, the repository could be
       empty; then it could be filled in by the objectives discovered by
       different devices (for example using Dynamic DNS Update). The
       more records are stored in the repository, the less the multicast-
       based discovery is needed. However, if we adopt such a mechanism,
       there would be challenges: stateful solution, and security.

       RESOLVED for now by adding optional use of DNS-SD by ASAs.
       Subsequently removed by editors as irrelevant to GRASP itself.

o 5. Need to expand description of the minimum requirements for the
       specification of an individual discovery, synchronization or
       negotiation objective.

       RESOLVED for now by extra wording.

o 6. Use case and protocol walkthrough. A description of how a
       node starts up, performs discovery, and conducts negotiation and
       synchronisation for a sample use case would help readers to
       understand the applicability of this specification. Maybe it
       should be an artificial use case or maybe a simple real one, based
       on a conceptual API. However, the authors have not yet decided
       whether to have a separate document or have it in the protocol
       document.

       RESOLVED: recommend a separate document.

o 7. Cross-check against other ANIMA WG documents for consistency
       and gaps.

       RESOLVED: Satisfied by WGLC.

o 8. Consideration of ADNCP proposal.

       RESOLVED by adding optional use of DNCP for flooding-type
       synchronization.

o 9. Clarify how a GDNP instance knows whether it is running inside
       the ACP. (Sheng)

       RESOLVED by improved text.

o 10. Clarify how a non-ACP GDNP instance initiates (D)TLS.
       (Sheng)

       RESOLVED by improved text and declaring DTLS out of scope for this
       draft.

o 11. Clarify how UDP/TCP choice is made. (Sheng) [Like DNS? -
       Brian]

       RESOLVED by improved text.

o 12. Justify that IP address within ACP or (D)TLS environment is
       sufficient to prove AN identity; or explain how Device Identity
       Option is used. (Sheng)

RESOLVED for now: we assume that all ASAs in a device are trusted as soon as the device is trusted, so they share credentials. In that case the Device Identity Option is useless. This needs to be reviewed later.

o 13. Emphasise that negotiation/synchronization are independent from discovery, although the rapid discovery mode includes the first step of a negotiation/synchronization. (Sheng)

   RESOLVED by improved text.

o 14. Do we need an unsolicited flooding mechanism for discovery (for discovery results that everyone needs), to reduce scaling impact of flooding discovery messages? (Toerless)

   RESOLVED: Yes, added to requirements and solution.

o 15. Do we need flag bits in Objective Options to distinguish distinguish Synchronization and Negotiation "Request" or rapid mode "Discovery" messages? (Bing)

   RESOLVED: yes, work on the API showed that these flags are essential.

o 16. (Related to issue 14). Should we revive the "unsolicited Response" for flooding synchronisation data? This has to be done carefully due to the well-known issues with flooding, but it could be useful, e.g. for Intent distribution, where DNCP doesn't seem applicable.

   RESOLVED: Yes, see #14.

o 17. Ensure that the discovery mechanism is completely proof against loops and protected against duplicate responses.

   RESOLVED: Added loop count mechanism.

o 18. Discuss the handling of multiple valid discovery responses.

   RESOLVED: Stated that the choice must be available to the ASA but GRASP implementation should pick a default.

o 19. Should we use a text-oriented format such as JSON/CBOR instead of native binary TLV format?

   RESOLVED: Yes, changed to CBOR.

o 20. Is the Divert option needed? If a discovery response
        provides a valid IP address or FQDN, the recipient doesn't gain
        any extra knowledge from the Divert. On the other hand, the
        presence of Divert informs the receiver that the target is off-
        link, which might be useful sometimes.

        RESOLVED: Decided to keep Divert option.

o 21. Rename the protocol as GRASP (GeneRic Autonomic Signaling
        Protocol)?

        RESOLVED: Yes, name changed.

o 22. Does discovery mechanism scale robustly as needed? Need hop
        limit on relaying?

        RESOLVED: Added hop limit.

o 23. Need more details on TTL for caching discovery responses.

        RESOLVED: Done.

o 24. Do we need "fast withdrawal" of discovery responses?

        RESOLVED: This doesn't seem necessary. If an ASA exits or stops
        supporting a given objective, peers will fail to start future
        sessions and will simply repeat discovery.

o 25. Does GDNP discovery meet the needs of multi-hop DNS-SD?

        RESOLVED: Decided not to consider this further as a GRASP protocol
        issue. GRASP objectives could embed DNS-SD formats if needed.

o 26. Add a URL type to the locator options (for security bootstrap
        etc.)

        RESOLVED: Done, later renamed as URI.

o 27. Security of Flood multicasts (Section 3.5.6.1).

        RESOLVED: added text.

o 28. Does ACP support secure link-local multicast?

        RESOLVED by new text in the Security Considerations.

o 29. PEN is used to distinguish vendor options. Would it be
        better to use a domain name? Anything unique will do.

      RESOLVED: Simplified this by removing PEN field and changing
      naming rules for objectives.

   o 30. Does response to discovery require randomized delays to
      mitigate amplification attacks?

      RESOLVED: WG feedback is that it's unnecessary.

   o 31. We have specified repeats for failed discovery etc. Is that
      sufficient to deal with sleeping nodes?

      RESOLVED: WG feedback is that it's unnecessary to say more.

   o 32. We have one-to-one synchronization and flooding
      synchronization. Do we also need selective flooding to a subset
      of nodes?

      RESOLVED: This will be discussed as a protocol extension in a
      separate draft (draft-liu-anima-grasp-distribution).

   o 33. Clarify if/when discovery needs to be repeated.

      RESOLVED: Done.

   o 34. Clarify what is mandatory for running in ACP, expand
      discussion of security boundary when running with no ACP - might
      rely on the local PKI infrastructure.

      RESOLVED: Done.

   o 35. State that role-based authorization of ASAs is out of scope
      for GRASP. GRASP doesn't recognize/handle any "roles".

      RESOLVED: Done.

   o 36. Reconsider CBOR definition for PEN syntax. (objective-name
      = text / [pen, text] ; pen = uint)

      RESOLVED: See issue 29.

   o 37. Are URI locators really needed?

      RESOLVED: Yes, e.g. for security bootstrap discovery, but added
      note that addresses are the normal case (same for FQDN locators).

   o 38. Is Session ID sufficient to identify relayed responses?
      Isn't the originator's address needed too?

      RESOLVED: Yes, this is needed for multicast messages and their
      responses.

   o 39. Clarify that a node will contain one GRASP instance
      supporting multiple ASAs.

      RESOLVED: Done.

   o 40. Add a "reason" code to the DECLINE option?

      RESOLVED: Done.

   o 41. What happens if an ASA cannot conveniently use one of the
      GRASP mechanisms? Do we (a) add a message type to GRASP, or (b)
      simply pass the discovery results to the ASA so that it can open
      its own socket?

      RESOLVED: Both would be possible, but (b) is preferred.

   o 42. Do we need a feature whereby an ASA can bypass the ACP and
      use the data plane for efficiency/throughput? This would require
      discovery to return non-ACP addresses and would evade ACP
      security.

      RESOLVED: This is considered out of scope for GRASP, but a comment
      has been added in security considerations.

   o 43. Rapid mode synchronization and negotiation is currently
      limited to a single objective for simplicity of design and
      implementation. A future consideration is to allow multiple
      objectives in rapid mode for greater efficiency.

      RESOLVED: This is considered out of scope for this version.

   o 44. In requirement T9, the words that encryption "may not be
      required in all deployments" were removed. Is that OK?.

      RESOLVED: No objections.

   o 45. Device Identity Option is unused. Can we remove it
      completely?.

      RESOLVED: No objections. Done.

   o 46. The 'initiator' field in DISCOVER, RESPONSE and FLOOD
      messages is intended to assist in loop prevention. However, we
      also have the loop count for that. Also, if we create a new
      Session ID each time a DISCOVER or FLOOD is relayed, that ID can

   be disambiguated by recipients. It would be simpler to remove the
   initiator from the messages, making parsing more uniform. Is that
   OK?

      RESOLVED: Yes. Done.

o 47. REQUEST is a dual purpose message (request negotiation or
   request synchronization). Would it be better to split this into
   two different messages (and adjust various message names
   accordingly)?

      RESOLVED: Yes. Done.

o 48. Should the Appendix "Capability Analysis of Current
   Protocols" be deleted before RFC publication?

      RESOLVED: No (per WG meeting at IETF 96).

o 49. Section 3.5.1 Should say more about signaling between two
   autonomic networks/domains.

      RESOLVED: Description of separate GRASP instance added.

o 50. Is Rapid mode limited to on-link only? What happens if first
   discovery responder does not support Rapid Mode? Section 3.5.5,
   Section 3.5.6)

      RESOLVED: Not limited to on-link. First responder wins.

o 51. Should flooded objectives have a time-to-live before they are
   deleted from the flood cache? And should they be tagged in the
   cache with their source locator?

      RESOLVED: TTL added to Flood (and Discovery Response) messages.
      Cached flooded objectives must be tagged with their originating
      ASA locator, and multiple copies must be kept if necessary.

o 52. Describe in detail what is allowed and disallowed in an
   insecure instance of GRASP.

      RESOLVED: Done.

o 53. Tune IANA Considerations to support early assignment request.

o 54. Is there a highly unlikely race condition if two peers
   simultaneously choose the same Session ID and send each other
   simultaneous M_REQ_NEG messages?

      RESOLVED: Yes. Enhanced text on Session ID generation, and added
      precaution when receiving a Request message.

   o 55. Could discovery be performed over TCP?

      RESOLVED: Unicast discovery added as an option.

   o 56. Change Session-ID to 32 bits?

      RESOLVED: Done.

   o 57. Add M_INVALID message?

      RESOLVED: Done.

   o 58. Maximum message size?

      RESOLVED by specifying default maximum message size (2048 bytes).

   o 59. Add F_NEG_DRY flag to specify a "dry run" objective?.

      RESOLVED: Done.

   o 60. Change M_FLOOD syntax to associate a locator with each
      objective?

      RESOLVED: Done.

   o 61. Is the SONN constrained instance really needed?

      RESOLVED: Retained but only as an option.

   o 62. Is it helpful to tag descriptive text with message names
      (M_DISCOVER etc.)?

      RESOLVED: Yes, done in various parts of the text.

Appendix C. Change log [RFC Editor: Please remove]

   draft-ietf-anima-grasp-09, 2016-12-15:

   Protocol change: Add F_NEG_DRY flag to specify a "dry run" objective.

   Protocol change: Change M_FLOOD syntax to associate a locator with
   each objective.

   Concentrated mentions of TLS in one section, with all details out of
   scope.

Clarified text around constrained instances of GRASP.

Strengthened text restricting LL addresses in locator options.

Clarified description of rapid mode processsing.

Specified that cached discovery results should not be returned on the same interface where they were learned.

Shortened text in "High Level Design Choices"

Dropped the word 'kernel' to avoid confusion with o/s kernel mode.

Editorial improvements and clarifications.

draft-ietf-anima-grasp-08, 2016-10-30:

Protocol change: Added M_INVALID message.

Protocol change: Increased Session ID space to 32 bits.

Enhanced rules to avoid Session ID clashes.

Corrected and completed description of timeouts for Request messages.

Improved wording about exponential backoff and DoS.

Clarified that discovery relaying is not done by limited security instances.

Corrected and expanded explanation of port used for Discovery Response.

Noted that Discovery message could be sent unicast in special cases.

Added paragraph on extensibility.

Specified default maximum message size.

Added Appendix for sample messages.

Added short protocol overview.

Editorial fixes, including minor re-ordering for readability.

draft-ietf-anima-grasp-07, 2016-09-13:

Protocol change: Added TTL field to Flood message (issue 51).

Protocol change: Added Locator option to Flood message (issue 51).

Protocol change: Added TTL field to Discovery Response message (corrollary to issue 51).

Clarified details of rapid mode (issues 43 and 50).

Description of inter-domain GRASP instance added (issue 49).

Description of limited security GRASP instances added (issue 52).

Strengthened advice to use TCP rather than UDP.

Updated IANA considerations and text about well-known port usage (issue 53).

Amended text about ASA authorization and roles to allow for overlapping ASAs.

Added text recommending that Flood should be repeated periodically.

Editorial fixes.

draft-ietf-anima-grasp-06, 2016-06-27:

Added text on discovery cache timeouts.

Noted that ASAs that are only initiators do not need to respond to discovery message.

Added text on unexpected address changes.

Added text on robust implementation.

Clarifications and editorial fixes for numerous review comments

Added open issues for some review comments.

draft-ietf-anima-grasp-05, 2016-05-13:

Noted in requirement T1 that it should be possible to implement ASAs independently as user space programs.

Protocol change: Added protocol number and port to discovery response. Updated protocol description, CDDL and IANA considerations accordingly.

Clarified that discovery and flood multicasts are handled by the
GRASP core, not directly by ASAs.

Clarified that a node may discover an objective without supporting it
for synchronization or negotiation.

Added Implementation Status section.

Added reference to SCSP.

Editorial fixes.

draft-ietf-anima-grasp-04, 2016-03-11:

Protocol change: Restored initiator field in certain messages and
adjusted relaying rules to provide complete loop detection.

Updated IANA Considerations.

draft-ietf-anima-grasp-03, 2016-02-24:

Protocol change: Removed initiator field from certain messages and
adjusted relaying requirement to simplify loop detection. Also
clarified narrative explanation of discovery relaying.

Protocol change: Split Request message into two (Request Negotiation
and Request Synchronization) and updated other message names for
clarity.

Protocol change: Dropped unused Device ID option.

Further clarified text on transport layer usage.

New text about multicast insecurity in Security Considerations.

Various other clarifications and editorial fixes, including moving
some material to Appendix.

draft-ietf-anima-grasp-02, 2016-01-13:

Resolved numerous issues according to WG discussions.

Renumbered requirements, added D9.

Protocol change: only allow one objective in rapid mode.

Protocol change: added optional error string to DECLINE option.

Protocol change: removed statement that seemed to say that a Request not preceded by a Discovery should cause a Discovery response. That made no sense, because there is no way the initiator would know where to send the Request.

Protocol change: Removed PEN option from vendor objectives, changed naming rule accordingly.

Protocol change: Added FLOOD message to simplify coding.

Protocol change: Added SYNCH message to simplify coding.

Protocol change: Added initiator id to DISCOVER, RESPONSE and FLOOD messages. But also allowed the relay process for DISCOVER and FLOOD to regenerate a Session ID.

Protocol change: Require that discovered addresses must be global (except during bootstrap).

Protocol change: Receiver of REQUEST message must close socket if no ASA is listening for the objective.

Protocol change: Simplified Waiting message.

Protocol change: Added No Operation message.

Renamed URL locator type as URI locator type.

Updated CDDL definition.

Various other clarifications and editorial fixes.

draft-ietf-anima-grasp-01, 2015-10-09:

Updated requirements after list discussion.

Changed from TLV to CBOR format - many detailed changes, added co-author.

Tightened up loop count and timeouts for various cases.

Noted that GRASP does not provide transactional integrity.

Various other clarifications and editorial fixes.

draft-ietf-anima-grasp-00, 2015-08-14:

File name and protocol name changed following WG adoption.

Added URL locator type.

draft-carpenter-anima-gdn-protocol-04, 2015-06-21:

Tuned wording around hierarchical structure.

Changed "device" to "ASA" in many places.

Reformulated requirements to be clear that the ASA is the main customer for signaling.

Added requirement for flooding unsolicited synch, and added it to protocol spec. Recognized DNCP as alternative for flooding synch data.

Requirements clarified, expanded and rearranged following design team discussion.

Clarified that GDNP discovery must not be a prerequisite for GDNP negotiation or synchronization (resolved issue 13).

Specified flag bits for objective options (resolved issue 15).

Clarified usage of ACP vs TLS/DTLS and TCP vs UDP (resolved issues 9,10,11).

Updated DNCP description from latest DNCP draft.

Editorial improvements.

draft-carpenter-anima-gdn-protocol-03, 2015-04-20:

Removed intrinsic security, required external security

Format changes to allow DNCP co-existence

Recognized DNS-SD as alternative discovery method.

Editorial improvements

draft-carpenter-anima-gdn-protocol-02, 2015-02-19:

Tuned requirements to clarify scope,

Clarified relationship between types of objective,

Clarified that objectives may be simple values or complex data structures,

Improved description of objective options,

Added loop-avoidance mechanisms (loop count and default timeout,
limitations on discovery relaying and on unsolicited responses),

Allow multiple discovery objectives in one response,

Provided for missing or multiple discovery responses,

Indicated how modes such as "dry run" should be supported,

Minor editorial and technical corrections and clarifications,

Reorganized future work list.

draft-carpenter-anima-gdn-protocol-01, restructured the logical flow
of the document, updated to describe synchronization completely, add
unsolicited responses, numerous corrections and clarifications,
expanded future work list, 2015-01-06.

draft-carpenter-anima-gdn-protocol-00, combination of draft-jiang-
config-negotiation-ps-03 and draft-jiang-config-negotiation-protocol-
02, 2014-10-08.

Appendix D. Example Message Formats

For readers unfamiliar with CBOR, this appendix shows a number of
example GRASP messages conforming to the CDDL syntax given in
Section 6. Each message is shown three times in the following
formats:

1. CBOR diagnostic notation.

2. Similar, but showing the names of the constants. (Details of the
    flag bit encoding are omitted.)

3. Hexadecimal version of the CBOR wire format.

Long lines are split for display purposes only.

D.1. Discovery Example

The initiator (2001:db8:f000:baaa:28cc:dc4c:9703:6781) multicasts a
discovery message looking for objective EX1:

```
   [1, 13948744, h'20010db8f000baaa28ccdc4c97036781', ["EX1", 5, 2, 0]]
   [M_DISCOVERY, 13948744, h'20010db8f000baaa28ccdc4c97036781',
                 ["EX1", F_SYNCH_bits, 2, 0]]
   h'84011a00d4d7485020010db8f000baaa28ccdc4c970367818463455831050200'
```

   A peer (2001:0db8:f000:baaa:f000:baaa:f000:baaa) responds with a
   locator:

```
   [2, 13948744, h'20010db8f000baaa28ccdc4c97036781', 60000,
                 [103, h'20010db8f000baaaf000baaaf000baaa', 6, 49443]]
   [M_RESPONSE, 13948744, h'20010db8f000baaa28ccdc4c97036781', 60000,
                 [O_IPv6_LOCATOR, h'20010db8f000baaaf000baaaf000baaa',
                  IPPROTO_TCP, 49443]]
   h'85021a00d4d7485020010db8f000baaa28ccdc4c9703678119ea6084186750
     20010db8f000baaaf000baaaf000baaa0619c123'
```

D.2. Flood Example

   The initiator multicasts a flood message. The single objective has a
   null locator. There is no response:

```
[9, 3504974, h'20010db8f000baaa28ccdc4c97036781', 10000,
             [["EX1", 5, 2, ["Example 1 value=", 100]],[] ] ]
[M_FLOOD, 3504974, h'20010db8f000baaa28ccdc4c97036781', 10000,
             [["EX1", F_SYNCH_bits, 2, ["Example 1 value=", 100]],[] ] ]
h'86091a00357b4e5020010db8f000baaa28ccdc4c97036781192710
  828463455831050282704578616d706c6520312076616c75653d186480'
```

D.3. Synchronization Example

   Following successful discovery of objective EX2, the initiator
   unicasts a request:

```
   [4, 4038926, ["EX2", 5, 5, 0]]
   [M_REQ_SYN, 4038926, ["EX2", F_SYNCH_bits, 5, 0]]
   h'83041a003da10e8463455832050500'
```

   The peer responds with a value:

```
 [8, 4038926, ["EX2", 5, 5, ["Example 2 value=", 200]]]
 [M_SYNCH, 4038926, ["EX2", F_SYNCH_bits, 5, ["Example 2 value=", 200]]]
 h'83081a003da10e84634558320505082704578616d706c6520322076616c75653d18c8'
```

D.4. Simple Negotiation Example

   Following successful discovery of objective EX3, the initiator
   unicasts a request:

```
[3, 802813, ["EX3", 3, 6, ["NZD", 47]]]
[M_REQ_NEG, 802813, ["EX3", F_NEG_bits, 6, ["NZD", 47]]]
h'83031a000c3ffd84634558330030682634e5a44182f'
```

The peer responds with immediate acceptance. Note that no objective
is needed, because the initiator's request was accepted without
change:

```
[6, 802813, [101]]
[M_END , 802813, [O_ACCEPT]]
h'83061a000c3ffd811865'
```

D.5. Complete Negotiation Example

Again the initiator unicasts a request:

```
[3, 13767778, ["EX3", 3, 6, ["NZD", 410]]]
[M_REQ_NEG, 13767778, ["EX3", F_NEG_bits, 6, ["NZD", 410]]]
h'83031a00d21462846345583303030682634e5a4419019a'
```

The responder starts to negotiate (making an offer):

```
[5, 13767778, ["EX3", 3, 6, ["NZD", 80]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 6, ["NZD", 80]]]
h'83051a00d21462846345583303030682634e5a441850'
```

The initiator continues to negotiate (reducing its request, and note
that the loop count is decremented):

```
[5, 13767778, ["EX3", 3, 5, ["NZD", 307]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 5, ["NZD", 307]]]
h'83051a00d21462846345583303030582634e5a44190133'
```

The responder asks for more time:

```
[7, 13767778, 34965]
[M_WAIT, 13767778, 34965]
h'83071a00d21462198895'
```

The responder continues to negotiate (increasing its offer):

```
[5, 13767778, ["EX3", 3, 4, ["NZD", 120]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 4, ["NZD", 120]]]
h'83051a00d21462846345583303030482634e5a441878'
```

The initiator continues to negotiate (reducing its request):

```
[5, 13767778, ["EX3", 3, 3, ["NZD", 246]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 3, ["NZD", 246]]]
h'83051a00d214628463455833030382634e5a4418f6'
```

The responder refuses to negotiate further:

```
[6, 13767778, [102, "Insufficient funds"]]
[M_END , 13767778, [O_DECLINE, "Insufficient funds"]]
h'83061a00d2146282186672496e73756666696369656e742066756e6473'
```

This negotiation has failed. If either side had sent [M_END,
13767778, [O_ACCEPT]] it would have succeeded, converging on the
objective value in the preceding M_NEGOTIATE. Note that apart from
the initial M_REQ_NEG, the process is symmetrical.

[Appendix E](). Capability Analysis of Current Protocols

This appendix discusses various existing protocols with properties
related to the requirements described in [Section 2](). The purpose is
to evaluate whether any existing protocol, or a simple combination of
existing protocols, can meet those requirements.

Numerous protocols include some form of discovery, but these all
appear to be very specific in their applicability. Service Location
Protocol (SLP) [[RFC2608]()] provides service discovery for managed
networks, but requires configuration of its own servers. DNS-SD
[[RFC6763]()] combined with mDNS [[RFC6762]()] provides service discovery for
small networks with a single link layer. [[RFC7558]()] aims to extend
this to larger autonomous networks but this is not yet standardized.
However, both SLP and DNS-SD appear to target primarily application
layer services, not the layer 2 and 3 objectives relevant to basic
network configuration. Both SLP and DNS-SD are text-based protocols.

Routing protocols are mainly one-way information announcements. The
receiver makes independent decisions based on the received
information and there is no direct feedback information to the
announcing peer. This remains true even though the protocol is used
in both directions between peer routers; there is state
synchronization, but no negotiation, and each peer runs its route
calculations independently.

Simple Network Management Protocol (SNMP) [[RFC3416]()] uses a command/
response model not well suited for peer negotiation. Network
Configuration Protocol (NETCONF) [[RFC6241]()] uses an RPC model that
does allow positive or negative responses from the target system, but
this is still not adequate for negotiation.

There are various existing protocols that have elementary negotiation
abilities, such as Dynamic Host Configuration Protocol for IPv6
(DHCPv6) [RFC3315], Neighbor Discovery (ND) [RFC4861], Port Control
Protocol (PCP) [RFC6887], Remote Authentication Dial In User Service
(RADIUS) [RFC2865], Diameter [RFC6733], etc. Most of them are
configuration or management protocols. However, they either provide
only a simple request/response model in a master/slave context or
very limited negotiation abilities.

There are some signaling protocols with an element of negotiation.
For example Resource ReSerVation Protocol (RSVP) [RFC2205] was
designed for negotiating quality of service parameters along the path
of a unicast or multicast flow. RSVP is a very specialised protocol
aimed at end-to-end flows. However, it has some flexibility, having
been extended for MPLS label distribution [RFC3209]. A more generic
design is General Internet Signalling Transport (GIST) [RFC5971], but
it is complex, tries to solve many problems, and is also aimed at
per-flow signaling across many hops rather than at device-to-device
signaling. However, we cannot completely exclude extended RSVP or
GIST as a synchronization and negotiation protocol. They do not
appear to be directly useable for peer discovery.

We now consider two protocols that are works in progress at the time
of this writing. Firstly, RESTCONF [I-D.ietf-netconf-restconf] is a
protocol intended to convey NETCONF information expressed in the YANG
language via HTTP, including the ability to transit HTML
intermediaries. While this is a powerful approach in the context of
centralised configuration of a complex network, it is not well
adapted to efficient interactive negotiation between peer devices,
especially simple ones that might not include YANG processing
already.

Secondly, we consider Distributed Node Consensus Protocol (DNCP)
[RFC7787]. This is defined as a generic form of state
synchronization protocol, with a proposed usage profile being the
Home Networking Control Protocol (HNCP) [RFC7788] for configuring
Homenet routers. A specific application of DNCP for autonomic
networking was proposed in [I-D.stenberg-anima-adncp].

DNCP "is designed to provide a way for each participating node to
publish a set of TLV (Type-Length-Value) tuples, and to provide a
shared and common view about the data published... DNCP is most
suitable for data that changes only infrequently... If constant rapid
state changes are needed, the preferable choice is to use an
additional point-to-point channel..."

Specific features of DNCP include:

o Every participating node has a unique node identifier.

o DNCP messages are encoded as a sequence of TLV objects, sent over
   unicast UDP or TCP, with or without (D)TLS security.

o Multicast is used only for discovery of DNCP neighbors when lower
   security is acceptable.

o Synchronization of state is maintained by a flooding process using
   the Trickle algorithm. There is no bilateral synchronization or
   negotiation capability.

o The HNCP profile of DNCP is designed to operate between directly
   connected neighbors on a shared link using UDP and link-local IPv6
   addresses.

DNCP does not meet the needs of a general negotiation protocol,
because it is designed specifically for flooding synchronization.
Also, in its HNCP profile it is limited to link-local messages and to
IPv6. However, at the minimum it is a very interesting test case for
this style of interaction between devices without needing a central
authority, and it is a proven method of network-wide state
synchronization by flooding.

The Server Cache Synchronization Protocol (SCSP) [RFC2334] also
describes a method for cache synchronization and cache replication
among a group of nodes.

A proposal was made some years ago for an IP based Generic Control
Protocol (IGCP) [I-D.chaparadza-intarea-igcp]. This was aimed at
information exchange and negotiation but not directly at peer
discovery. However, it has many points in common with the present
work.

None of the above solutions appears to completely meet the needs of
generic discovery, state synchronization and negotiation in a single
solution. Many of the protocols assume that they are working in a
traditional top-down or north-south scenario, rather than a fluid
peer-to-peer scenario. Most of them are specialized in one way or
another. As a result, we have not identified a combination of
existing protocols that meets the requirements in Section 2. Also,
we have not identified a path by which one of the existing protocols
could be extended to meet the requirements.

Authors' Addresses

   Carsten Bormann
   Universitaet Bremen TZI
   Postfach 330440
   D-28359 Bremen
   Germany

   Email: cabo@tzi.org

   Brian Carpenter (editor)
   Department of Computer Science
   University of Auckland
   PB 92019
   Auckland 1142
   New Zealand

   Email: brian.e.carpenter@gmail.com

   Bing Liu (editor)
   Huawei Technologies Co., Ltd
   Q14, Huawei Campus
   No.156 Beiqing Road
   Hai-Dian District, Beijing 100095
   P.R. China

   Email: leo.liubing@huawei.com