

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 22, 2015

B. Carpenter
Univ. of Auckland
B. Liu
Huawei Technologies Co., Ltd
April 20, 2015

**A Generic Discovery and Negotiation Protocol for Autonomic Networking
draft-carpenter-anima-gdn-protocol-03**

Abstract

This document establishes requirements for a protocol that enables intelligent devices to dynamically discover peer devices, 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

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on October 22, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Requirement Analysis of Discovery, Synchronization and Negotiation	4
2.1.	Requirements for Discovery	4
2.2.	Requirements for Synchronization and Negotiation Capability	5
2.3.	Specific Technical Requirements	7
3.	GDN Protocol Overview	8
3.1.	Terminology	8
3.2.	High-Level Design Choices	9
3.3.	GDN Protocol Basic Properties and Mechanisms	13
3.3.1.	Required External Security Mechanism	13
3.3.2.	Transport Layer Usage	13
3.3.3.	Discovery Mechanism and Procedures	13
3.3.4.	Negotiation Procedures	15
3.3.5.	Synchronization Procedure	16
3.4.	GDN Constants	17
3.5.	Session Identifier (Session ID)	17
3.6.	GDN Messages	18
3.6.1.	GDN Message Format	18
3.6.2.	Discovery Message	19
3.6.3.	Response Message	19
3.6.4.	Request Message	20
3.6.5.	Negotiation Message	20
3.6.6.	Negotiation-ending Message	21
3.6.7.	Confirm-waiting Message	21
3.7.	GDN General Options	21
3.7.1.	Format of GDN Options	21
3.7.2.	Divert Option	22
3.7.3.	Accept Option	22
3.7.4.	Decline Option	23
3.7.5.	Waiting Time Option	23
3.7.6.	Device Identity Option	24
3.7.7.	Locator Options	24
3.8.	Objective Options	26
3.8.1.	Format of Objective Options	26
3.8.2.	General Considerations for Objective Options	27
3.8.3.	Organizing of Objective Options	27
3.8.4.	Vendor Specific Objective Options	28
3.8.5.	Experimental Objective Options	29
4.	Items for Future Work	29

5.	Security Considerations	30
6.	IANA Considerations	31
7.	Acknowledgements	33
8.	Change log [RFC Editor: Please remove]	33
9.	References	34
9.1.	Normative References	34
9.2.	Informative References	35
Appendix A.	Capability Analysis of Current Protocols	37
	Authors' Addresses	40

[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 [\[I-D.irtf-nmrg-autonomic-network-definitions\]](#) and [\[I-D.irtf-nmrg-an-gap-analysis\]](#). In order to fulfil autonomy, devices that embody autonomic service agents need to be able to discover each other, to synchronize state with each other, and to negotiate parameters and resources directly with each other. There is no restriction on the type of parameters and resources concerned, which include very basic information needed for addressing and routing, as well as anything else that might be configured in a conventional network.

Following this Introduction, [Section 2](#) describes the requirements for network device discovery, synchronization and negotiation. Negotiation is an iterative process, requiring multiple message exchanges forming a closed loop between the negotiating devices. State synchronization, when needed, can be regarded as a special case of negotiation, without iteration. [Section 3.2](#) describes a behavior model for a protocol intended to support discovery, synchronization and negotiation. The design of Generic Discovery and Negotiation Protocol (GDNP) in [Section 3](#) of this document is mainly based on this behavior model. The relevant capabilities of various existing protocols are reviewed in [Appendix A](#).

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. Although many negotiations will occur between horizontally distributed peers, many target scenarios are hierarchical networks, which is the predominant structure of current large-scale networks. However, when a device starts up with no pre-configuration, it has no knowledge of a hierarchical superior. 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 GDNP 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 GDNP. It is expected that some autonomic service agents will 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.

2.1. Requirements for Discovery

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. In some cases, when a new application session starts up within a device, the device may again lack information about relevant peer devices. 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. Therefore a basic requirement is that there must be a mechanism by which a device can separately discover peer devices for each of the technical objectives that it needs to manage. 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 peer devices 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 support both on-link discovery and off-link discovery of peers that support specific technical objectives.

The relevant peer devices 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. In many scenarios, the discovery process may be followed by a synchronization or negotiation process. Therefore, a discovery objective may be

associated with one or more synchronization or negotiation objectives.

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

In some networks, as mentioned above, there will be some hierarchical structure, at least for certain synchronization or negotiation objectives. A special case of discovery is that each device must be able to discover its hierarchical superior for each such objective that it is capable of handling. This is part of the more general requirement to discover off-link devices.

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 [\[I-D.pritikin-anima-bootstrapping-keyinfra\]](#). In addition, depending on the type of network involved, discovery of other central functions might be needed, such as the Network Operations Center (NOC) [\[I-D.eckert-anima-stable-connectivity\]](#). GDNP must be capable of supporting such discovery during initialisation, as well as discovery during ongoing operation.

2.2. Requirements for Synchronization and Negotiation Capability

We start by considering routing protocols, the closest approximation to autonomic networking in widespread use. Routing protocols use a largely autonomic model based on distributed devices that communicate repeatedly with each other. However, routing is mainly based on one-way information synchronization (in either direction), rather than on bi-directional negotiation. The focus is reachability, so current routing protocols only 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. Other information, such as latency, congestion, capacity, and particularly unused capacity, would be helpful to get better path selection and utilization rate, but are not normally used in distributed routing algorithms. Also, autonomic networks need to be able to manage many more dimensions, such as security settings, power saving, load balancing, etc. In general, these items do not apply to all participating nodes, but only to a subset. A basic requirement for the protocol is therefore the ability to represent, discover, synchronize and negotiate almost any kind of network parameter among arbitrary subsets of participating nodes.

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 follows that a requirement for the protocol is to be capable of running in any device that would otherwise need human intervention.

Human intervention in large networks is often replaced by use of a top-down network management system (NMS). It therefore follows that a requirement for the protocol is to be capable of running in any device that would otherwise be managed by an NMS, and that it can co-exist with an NMS.

Since the goal is to minimize human intervention, it is necessary that the network can in effect "think ahead" before changing its parameters. In other words there must be a possibility of forecasting the effect of a change by a "dry run" mechanism before actually installing the change. This will be an application of the protocol rather than a feature of the protocol itself.

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.

Recovery from faults and identification of faulty devices should be as automatic as possible. However, the protocol's role is limited to the ability to handle discovery, synchronization and negotiation at any time, in case an autonomic service agent detects an anomaly such as a negotiation counterpart failing.

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

The protocol needs to be able to deal with a wide variety of technical objectives, covering any type of network parameter. Therefore the protocol will need either an explicit information model describing its messages, or at least a flexible and extensible message format. One design consideration is whether to adopt an existing information model or to design a new one. Another consideration is whether it should be able to carry some or all of the message formats used by existing configuration protocols.

2.3. Specific Technical Requirements

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 or broadcasting on a link, might need to be IP version dependent. In case of doubt, IPv6 should be preferred.

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

The negotiation process 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 this must be defined specifically for each use case, the protocol should have some general mechanisms in support of loop and deadlock prevention.

Dependencies and conflicts: In order to decide a configuration on 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 autonomic service agents involved. To allow this, there need to be clear boundaries and convergence mechanisms for negotiations. Also some mechanisms are needed to avoid loop dependencies.

Policy constraints: There must be provision for general policy intent rules to be applied by all devices in the network (e.g., security rules, prefix length, resource sharing rules). However, policy intent distribution might not use the negotiation protocol itself.

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 may not use the negotiation protocol itself.

The protocol needs to be fully secure against forged messages and man-in-the middle attacks, and as secure as reasonably possible against denial of service attacks. It needs to be capable of encryption in order to resist unwanted monitoring, although this capability may not be required in all deployments. However, it is

not required that the protocol itself provides these security features; it may depend on an existing secure environment.

3. GDN 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 [\[I-D.irtf-nmrg-autonomic-network-definitions\]](#).

The following additional terms are used throughout this document:

- o Discovery: a process by which a device discovers peer devices according to a specific discovery objective. The discovery results may be different according to the different discovery objectives. The discovered peer devices may later be used as negotiation counterparts or as sources of synchronization data.
- o Negotiation: a process by which two (or more) devices interact iteratively to agree on parameter settings that best satisfy the objectives of one or more devices.
- o State Synchronization: a process by which two (or more) devices interact to agree on the current state of parameter values stored in each device. This is a special case of negotiation in which information is sent but the devices do not request their peers to change parameter settings. All other definitions apply to both negotiation and synchronization.
- o Objective: An objective in GDNP is a configurable state of some kind, which occurs in three contexts: Discovery, Negotiation and Synchronization. In the protocol, an objective is represented by an identifier (actually a GDNP option number) and if relevant a value. Normally, a given objective will occur during discovery and negotiation, or during discovery and synchronization, but not in all three contexts.
 - * One device 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 be an autonomic service agent which may itself manage other nodes.
 - * Discovery Objective: if a node needs to synchronize or negotiate a specific objective but does not know a peer that supports this objective, it starts a discovery process. The objective is called a Discovery Objective during this process.
 - * Synchronization Objective: an objective whose specific technical content needs to be synchronized among two or more devices.
 - * Negotiation Objective: an objective whose specific technical content needs to be decided in coordination with another network device.
- o Discovery Initiator: a device that spontaneously starts discovery by sending a discovery message referring to a specific discovery objective.
 - o Discovery Responder: a peer device which responds to the discovery objective initiated by the discovery initiator.
 - o Synchronization Initiator: a device that spontaneously starts synchronization by sending a request message referring to a specific synchronization objective.
 - o Synchronization Responder: a peer device which responds with the value of a synchronization objective.
 - o Negotiation Initiator: a device that spontaneously starts negotiation by sending a request message referring to a specific negotiation objective.
 - o Negotiation Counterpart: a peer device with which the Negotiation Initiator negotiates a specific negotiation objective.

3.2. High-Level Design Choices

This section describes a behavior model and some considerations for designing a generic discovery, synchronization and negotiation protocol, which can act as a platform for different technical objectives.

NOTE: This protocol is described here in a stand-alone fashion as a proof of concept. An early version was prototyped by Huawei and the Beijing University of Posts and Telecommunications. However, this is not yet a definitive proposal for IETF adoption. In particular, adaptation and extension of one of the protocols discussed in [Appendix A](#) might be an option. This whole specification is subject to change as a result.

- 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 synchronization or negotiation objectives and the different pairs of counterparts.

- 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 assumed to run within an existing secure environment with strong authentication.

On the other hand, a limited negotiation model might be deployed based on a limited trust relationship. For example, between two administrative domains, devices might also exchange limited information and negotiate some particular configurations based on a limited conventional or contractual trust relationship.

- o Discovery, synchronization and negotiation 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. These processes can also be performed independently when appropriate.

- * GDN discovery is appropriate for efficient discovery of GDN peers and allows a rapid mode of operation described in [Section 3.3.3](#). For some parameters, especially those concerned with application layer services, a text-based discovery mechanism such as DNS Service Discovery [[I-D.ietf-dnssd-requirements](#)] or Service Location Protocol [[RFC2608](#)] might be more appropriate. The choice is left to the designers of individual Autonomic Service Agents.

- o A uniform pattern for technical contents

The synchronization and negotiation contents are defined according to a uniform pattern. They could be carried either in simple TLV (Type, Length and Value) format or in payloads described by a flexible language. The initial protocol design uses the TLV approach. The format is extensible for unknown future requirements.

- o A conservative model for synchronization

GDNP supports bilateral synchronization, which could be used to perform synchronization among a small number of nodes.

- * For some parameters, synchronization across large groups of nodes, possibly including all autonomic nodes, might be needed. For such cases, a flooding mechanism such as ADNCP [[I-D.stenberg-anima-adncp](#)] is considered more appropriate. GDNP is designed to coexist with ADNCP. The choice is left to the designers of individual Autonomic Service Agents.

- o A simple initiator/responder model for negotiation

Multi-party negotiations are too complicated to be modeled and there might be too many dependencies among the parties to converge efficiently. A simple initiator/responder model is more feasible and can complete multiple-party negotiations by indirect steps.

- o Organizing of synchronization or negotiation content

Naturally, the technical content will be organized according to the relevant function or service. The content from different functions or services is kept independent from each other. They are not combined into a single option or single session because these contents may be negotiated or synchronized with different counterparts or may be different in response time.

- o Self aware network device

Every network device will be pre-loaded with various functions and be aware of its own capabilities, typically decided by the hardware, firmware or pre-installed software. Its exact role may depend on the surrounding network behaviors, which may include forwarding behaviors, aggregation properties, topology location, bandwidth, tunnel or translation properties, etc. The surrounding

topology will depend on the network planning. Following an initial discovery phase, the device properties and those of its neighbors are the foundation of the synchronization or negotiation behavior of a specific device. A device has no pre-configuration for the particular network in which it is installed.

- o Requests and responses in negotiation procedures

The initiator can negotiate with its relevant negotiation counterpart devices, which may be different according to the specific negotiation objective. It can request relevant information from the negotiation counterpart so that it can decide its local configuration to give the most coordinated performance. It can request the negotiation counterpart to make a matching configuration in order to set up a successful communication with it. It can request certain simulation or forecast results by sending some dry run conditions.

Beyond the traditional yes/no answer, the responder can reply with a suggested alternative if its answer is 'no'. This would start a bi-directional negotiation ending in a compromise between the two devices.

- o Convergence of negotiation procedures

To enable convergence, when a responder makes a suggestion of a changed condition in a negative reply, it should be as close as possible to the original request or previous suggestion. The suggested value of the third or later negotiation steps should be chosen between the suggested values from the last two negotiation steps. In any case there must be a mechanism to guarantee convergence (or failure) in a small number of steps, such as a timeout or maximum number of iterations.

- * End of negotiation

A limited number of rounds, for example three, or a timeout, is needed on each device for each negotiation objective. It may be an implementation choice, a pre-configurable parameter, or a network-wide policy intent. These choices might vary between different types of autonomic service agent. Therefore, the definition of each negotiation objective MUST clearly specify this, so that the negotiation can always be terminated properly.

- * Failed negotiation

There must be a well-defined procedure for concluding that a negotiation cannot succeed, and if so deciding what happens next (deadlock resolution, tie-breaking, or revert to best-effort service). Again, this MUST be specified for individual negotiation objectives, as an implementation choice, a pre-configurable parameter, or a network-wide policy intent.

3.3. GDN Protocol Basic Properties and Mechanisms

3.3.1. Required External Security Mechanism

The protocol SHOULD run within a secure Autonomic Control Plane (ACP) [[I-D.behringer-anima-autonomic-control-plane](#)]. If this is impossible, it MUST use TLS [[RFC5246](#)] or DTLS [[RFC6347](#)] for all messages, based on a local Public Key Infrastructure (PKI) [[RFC5280](#)] managed within the autonomic network itself.

Link-local multicast is used for discovery messages. These cannot be secured, but responses to discovery messages MUST be secured. However, during initialisation, before a node has joined the applicable trust infrastructure, e.g., [[I-D.pritikin-anima-bootstrapping-keyinfra](#)], it will be impossible to secure certain messages. Such messages MUST be limited to the strictly necessary minimum.

3.3.2. Transport Layer Usage

The protocol is capable of running over UDP or TCP, except for multicast discovery messages which can only run over UDP. When running within an ACP, UDP SHOULD be used for messages not exceeding the minimum IPv6 path MTU, and TCP SHOULD be used for longer messages. In other words, IPv6 fragmentation should be avoided. When running without an ACP, TLS MUST be used by default, except for multicast discovery messages. DTLS MAY be supported as an alternative.

3.3.3. Discovery Mechanism and Procedures

- o Separated discovery and negotiation mechanisms

Although discovery and negotiation or synchronization are defined together in the GDN, they are separated mechanisms. The discovery process could run independently from the negotiation or synchronization process. Upon receiving a discovery ([Section 3.6.2](#)) or request ([Section 3.6.4](#)) message, the recipient device should return a message in which it either

indicates itself as a discovery responder or diverts the initiator towards another more suitable device.

The discovery action will normally be followed by a negotiation or synchronization action. The discovery results could be utilized by the negotiation protocol to decide which device the initiator will negotiate with.

o Discovery Procedures

Discovery starts as an on-link operation. The Divert option can tell the discovery initiator to contact an off-link discovery objective device. Every DISCOVERY message is sent by a discovery initiator via UDP to the ALL_GDNP_NEIGHBOR multicast address ([Section 3.4](#)). Every network device that supports the GDNP always listens to a well-known UDP port to capture the discovery messages.

If the neighbor device supports the requested discovery objective, it MAY respond with a Response message ([Section 3.6.3](#)) with locator option(s). Otherwise, if the neighbor device has cached information about a device that supports the requested discovery objective (usually because it discovered the same objective before), it SHOULD respond with a Response message with a Divert option pointing to the appropriate Discovery Responder.

If no discovery response is received within a reasonable timeout (default GDNP_DEF_TIMEOUT milliseconds, [Section 3.4](#)), the DISCOVERY message MAY be repeated, with a newly generated Session ID ([Section 3.5](#)). An exponential backoff SHOULD be used for subsequent repetitions, in order to mitigate possible denial of service attacks.

After a GDNP device successfully discovers a Discovery Responder supporting a specific objective, it MUST cache this information. This cache record MAY be used for future negotiation or synchronization, and SHOULD be passed on when appropriate as a Divert option to another Discovery Initiator. The cache lifetime is an implementation choice.

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.

A GDNP device with multiple link-layer interfaces (typically a router) MUST support discovery on all interfaces. If it

receives a DISCOVERY message on a given interface for a specific objective that it does not support and for which it has not previously discovered a Discovery Responder, it MUST relay the query by re-issuing the same DISCOVERY message on its other interfaces. However, 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 of each relayed discovery message and, to prevent loops, MUST NOT relay a DISCOVERY message which carries such a cached Session ID.

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

- o A complete discovery process will start with multicast on the local link; a neighbor might divert it to an off-link destination, which could be a default higher-level gateway in a hierarchical network. Then discovery would continue with a unicast to that gateway; if that gateway is still not the right counterpart, it should divert to another device, which is in principle closer to the right counterpart. Finally the right counterpart responds to start the negotiation or synchronization process.
- o Rapid Mode (Discovery/Negotiation binding)

A Discovery message MAY include one or more Negotiation Objective option(s). This allows a rapid mode of negotiation described in [Section 3.3.4](#). A similar mechanism is defined for synchronization in [Section 3.3.5](#).

[3.3.4](#). Negotiation Procedures

A negotiation initiator sends a negotiation request to a counterpart device, 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 the counterpart can immediately apply the requested configuration, it will give an immediate positive (accept) answer. This will end the negotiation phase immediately. Otherwise, it will 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 to reach a compromise between the two network devices.

The negotiation procedure is ended when one of the negotiation peers sends a Negotiation Ending message, which contains an accept or 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 devices, or in simultaneous negotiations about different objectives. Thus, GDNP 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.

Rapid Mode (Discovery/Negotiation linkage)

A Discovery message MAY include a Negotiation Objective option. In this case the Discovery message also acts as a Request message to indicate to the Discovery Responder that it could directly reply to the Discovery Initiator with a Negotiation message for rapid processing, if it could act as the corresponding negotiation counterpart. However, the indication is only advisory not prescriptive.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. This rapid negotiation function SHOULD be configured off by default and MAY be configured on or off by policy intent.

3.3.5. Synchronization Procedure

A synchronization initiator sends a synchronization request to a counterpart device, including a specific synchronization objective. The counterpart responds with a Response message containing the current value of the requested synchronization objective. No further messages are needed. If no Response message is received, the synchronization request MAY be repeated after a suitable timeout.

In the case just described, the message exchange is unicast and concerns only one synchronization objective. For large groups of nodes requiring mutual synchronization, ADNCP [[I-D.stenberg-anima-adncp](#)] is considered more appropriate. In the following case, several synchronization objectives may be combined.

Rapid Mode (Discovery/Synchronization linkage)

A Discovery message MAY include one or more Synchronization Objective option(s). In this case the Discovery message also acts as a Request message to indicate to the Discovery Responder that it could directly reply to the Discovery Initiator with a Response message with synchronization data for rapid processing, if the discovery target supports the corresponding synchronization objective(s). However, the indication is only advisory not prescriptive.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. This rapid synchronization function SHOULD be configured off by default and MAY be configured on or off by policy intent.

3.4. GDNP Constants

o ALL_GDNP_NEIGHBOR

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

* IPv6 multicast address: TBD1

* IPv4 multicast address: TBD2

o GDNP Listen Port (TBD3)

A UDP and TCP port that every GDNP-enabled network device always listens to.

o GDNP_DEF_TIMEOUT (60000 milliseconds)

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

o GDNP_DEF_LOOPCT (6)

The default loop count used to determine that a negotiation has failed to complete.

3.5. Session Identifier (Session ID)

A 24-bit opaque value used to distinguish multiple sessions between the same two devices. A new Session ID MUST be generated for every new Discovery or Request message, and for every unsolicited Response message. All follow-up messages in the same discovery,

synchronization or negotiation procedure, which is initiated by the request message, MUST carry the same Session ID.

The Session ID SHOULD have a very low collision rate locally. It is RECOMMENDED to be generated by a pseudo-random algorithm using a seed which is unlikely to be used by any other device in the same network [[RFC4086](#)].

3.6. GDNP Messages

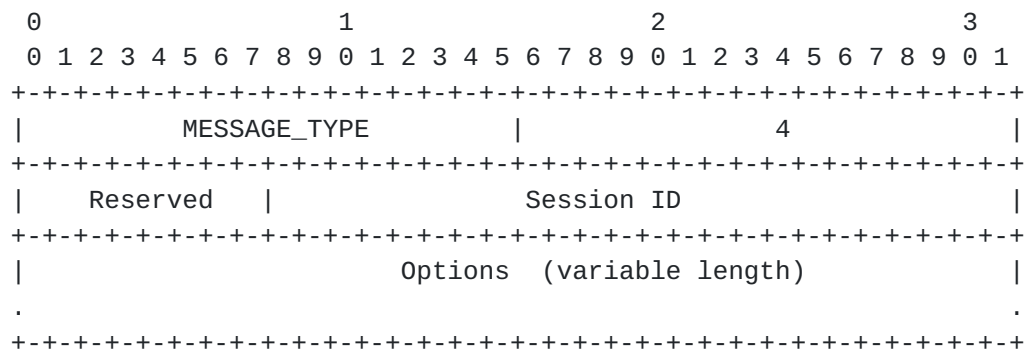
This document defines the following GDNP message format and types. Message types not listed here are reserved for future use. The numeric encoding for each message type is shown in parentheses.

3.6.1. GDNP Message Format

GDNP messages share an identical fixed format header and a variable format area for options. GDNP message headers and options are in the type-length-value (TLV) format defined in DNCP (see Section "Type-Length-Value Objects" in [[I-D.ietf-homenet-dncp](#)]).

Every GDNP message carries a Session ID. Options are presented serially in the options field, with padding to 4-byte alignment.

The following diagram illustrates the format of GDNP messages:



MESSAGE_TYPE: Identifies the GDNP message type. 16-bit.

Reserved: Set to zero, ignored on receipt. 8-bit.

Session ID: Identifies this GDNP session, as defined in [Section 3.5](#). 24-bit.

Options: GDNP Options carried in this message. Options are defined starting at [Section 3.7](#).

3.6.2. Discovery Message

DISCOVERY (MESSAGE_TYPE = G1):

A discovery initiator sends a DISCOVERY message to initiate a discovery process.

The discovery initiator sends the DISCOVERY messages to the link-local ALL_GDNP_NEIGHBOR multicast address for discovery, and stores the discovery results (including responding discovery objectives and corresponding unicast addresses or FQDNs).

A DISCOVERY message MUST include exactly one of the following:

- o a discovery objective option ([Section 3.8.1](#)).
- o a negotiation objective option ([Section 3.8.1](#)) to indicate to the discovery target that it MAY directly reply to the discovery initiator 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 message ([Section 3.6.4](#)).
- o one or more synchronization objective options ([Section 3.8.1](#)) to indicate to the discovery target that it MAY directly reply to the discovery initiator with a RESPONSE message for rapid processing, if it could act as the corresponding synchronization counterpart.

3.6.3. Response Message

RESPONSE (MESSAGE_TYPE = G2):

A node which receives a DISCOVERY message sends a Response message to respond to a discovery. It MUST contain the same Session ID as the DISCOVERY message. It MAY include a copy of the discovery objective from the DISCOVERY message.

If the responding node supports the discovery objective of the discovery, it MUST include at least one kind of locator option ([Section 3.7.7](#)) to indicate its own location. A combination of multiple kinds of locator options (e.g. IP address option + 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 3.7.2](#)) embedding a locator option or a combination of

multiple kinds of locator options which indicate the locator(s) of the discovery objective.

A node which receives a synchronization request sends a Response message with the synchronization data, in the form of GDNP Option(s) for the specific synchronization objective(s).

3.6.4. Request Message

REQUEST (MESSAGE_TYPE = G3):

A negotiation or synchronization requesting node sends the REQUEST message to the unicast address (directly stored or resolved from the FQDN) of the negotiation or synchronization counterpart (selected from the discovery results).

A request message MUST include the relevant objective option, with the requested value in the case of negotiation.

When an initiator sends a REQUEST message, it MUST initialize a negotiation timer for the new negotiation thread with the value GDNP_DEF_TIMEOUT milliseconds. Unless this timeout is modified by a CONFIRM-WAITING message ([Section 3.6.7](#)), the initiator will consider that the negotiation has failed when 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 GDNP_DEF_LOOPCT.

3.6.5. Negotiation Message

NEGOTIATION (MESSAGE_TYPE = G4):

A negotiation counterpart sends a NEGOTIATION message in response to a REQUEST 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 both parties will consider that the negotiation has failed.

3.6.6. Negotiation-ending Message

NEGOTIATION-ENDING (MESSAGE_TYPE = G5):

A negotiation counterpart sends an NEGOTIATION-ENDING message to close the negotiation. It MUST contain one, but only one of accept/decline option, defined in [Section 3.7.3](#) and [Section 3.7.4](#). It could be sent either by the requesting node or the responding node.

3.6.7. Confirm-waiting Message

CONFIRM-WAITING (MESSAGE_TYPE = G6):

A responding node sends a CONFIRM-WAITING message to indicate 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 than the Waiting Time Option ([Section 3.7.5](#)).

3.7. GDNP General Options

This section defines the GDNP general options for the negotiation and synchronization protocol signalling. Additional option types are reserved for GDNP general options defined in the future.

3.7.1. Format of GDNP Options

[illegible]

Option-code: An unsigned integer identifying the specific option type carried in this option.

Option-len: An unsigned integer giving the length of the option-data field in this option in octets.

Option-data: The data for the option; the format of this data depends on the definition of the option.

GDPN options are scoped by using encapsulation. If an option contains other options, the outer Option-len includes the total size

of the encapsulated options, and the latter apply only to the outer option.

3.7.2. Divert Option

The divert option is used to redirect a GDNP 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 Response messages. If found elsewhere, it **SHOULD** be silently ignored.

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          OPTION_DIVERT          |          option-len          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          Locator Option(s) of Diversion Device(s)          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Option-code: `OPTION_DIVERT` (G32).

Option-len: The total length of diverted destination sub-option(s) in octets.

Locator Option(s) of Diversion Device(s): Embedded Locator Option(s) ([Section 3.7.7](#)) that point to diverted destination device(s).

3.7.3. 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-ending messages. If found elsewhere, it **SHOULD** be silently ignored.

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          OPTION_ACCEPT          |          option-len          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Option-code: `OPTION_ACCEPT` (G33)

Option-len: 0

3.7.4. 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-ending messages. If found elsewhere, it **SHOULD** be silently ignored.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           OPTION_DECLINE           |           option-len           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Option-code: OPTION_DECLINE (G34)

Option-len: 0

Notes: 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 at least one data field with all bits set to 1 to indicate a meaningless initial value, or a specific objective option that provides further conditions for convergence.

3.7.5. Waiting Time Option

The waiting time option is used to indicate that the negotiation counterpart needs to wait for a further negotiation response, since the processing might need more time than usual or it might depend on another triggered negotiation.

The waiting time option **MUST** only be encapsulated in Confirm-waiting messages. If found elsewhere, it **SHOULD** be silently ignored. When received, its value overwrites the negotiation timer ([Section 3.6.4](#)).

The counterpart **SHOULD** send a Negotiation, Negotiation-Ending 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.


```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      OPTION_WAITING      |      option-len      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Time                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Option-code: OPTION_WAITING (G35)

Option-len: 4, in octets

Time: Time in milliseconds

3.7.6. Device Identity Option

The Device Identity option carries the identities of the sender and of the domain(s) that it belongs to. The format of the Device Identity option is as follows:

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      OPTION_DEVICE_ID      |      option-len      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               |
|                               Identities (variable length)          |
|                               |
|                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Option-code: OPTION_DEVICE_ID (G36)

Option-len: Length of identities in octets

Identities: A variable-length field containing the device identity and one or more domain identities. The format is not yet defined.

Note: Currently this option is a placeholder. It might be removed or modified.

3.7.7. Locator Options

These locator options are used to present a device's or interface's reachability information. They are Locator IPv4 Address Option, Locator IPv6 Address Option and Locator FQDN (Fully Qualified Domain Name) Option.

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

3.7.7.1. Locator IPv4 address option

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  OPTION_LOCATOR_IPV4ADDR  |          option-len          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               IPv4-Address                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Option-code: OPTION_LOCATOR_IPV4ADDR (G37)

Option-len: 4, in octets

IPv4-Address: The IPv4 address locator of the device/interface

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

3.7.7.2. Locator IPv6 address option

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  OPTION_LOCATOR_IPV6ADDR  |          option-len          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               IPv6-Address                  |
|                               |                               |
|                               |                               |
|                               |                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Option-code: OPTION_LOCATOR_IPV6ADDR (G38)

Option-len: 16, in octets

IPv6-Address: The IPv6 address locator of the device/interface

Note: A link-local IPv6 address MUST NOT be used when this option is used within the Divert option.

flags: Flag bits. This field is present if and only if defined in the specification of the objective.

value: This 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.

3.8.2. General Considerations for Objective Options

Objective Options MUST be assigned an option type greater than G63 in the GDNP option table.

An Objective Option that contains no additional fields, i.e., has a length of 4 octets, is a discovery objective and MUST only be used in Discovery and Response messages.

The Negotiation Objective Options contain negotiation objectives, which are various according to different functions/services. They MUST be carried by Discovery, Request 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 GDNP_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 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 1 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, Request or Response messages only. They include value fields only in Response messages.

3.8.3. Organizing of Objective Options

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

A synchronization objective SHOULD also be organized as a single GDNP 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 a "flags" octet, with bits 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 TLV sub-options or some other form of data structure. 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 GDNP protocol itself is agnostic on these questions.

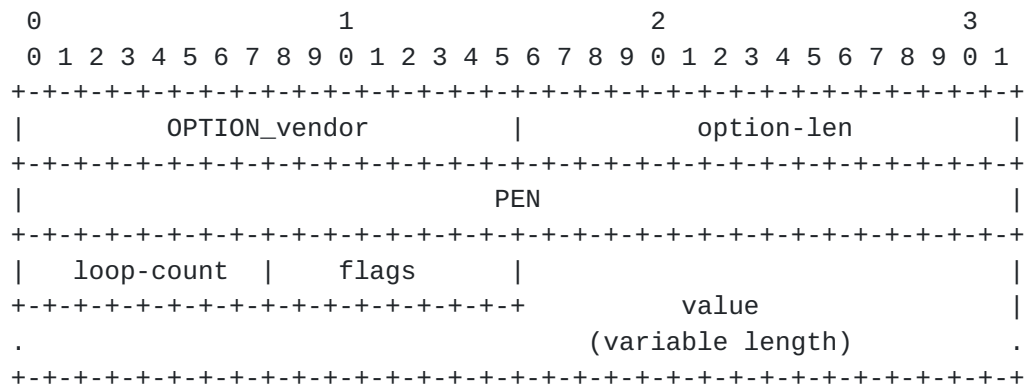
It is NOT RECOMMENDED to split parameters in a single objective into multiple options, unless they have different response periods. An exception scenario may also be described by split objectives.

3.8.4. Vendor Specific Objective Options

Option codes G128-159 have been reserved for vendor specific options. Multiple option codes have been assigned because a single vendor might use multiple options simultaneously. These vendor specific options are highly likely to have different meanings when used by different vendors. Therefore, they SHOULD NOT be used without an explicit human decision and SHOULD NOT be used in unmanaged networks such as home networks.

There is one general requirement that applies to all vendor specific options. They MUST start with a field that uniquely identifies the enterprise that defines the option, in the form of a registered 32 bit Private Enterprise Number (PEN) [[I-D.liang-iana-pen](#)]. There is no default value for this field. Note that it is not used during discovery. It MUST be verified during negotiation or synchronization.

In the case of a vendor-specific objective, the loop count and flags, if present, follow the PEN.



Option-code: OPTION_vendor (G128~159)

Option-len: The total length in octets.

PEN: Private Enterprise Number.

loop-count: The loop count. This field is present if and only if the objective is a negotiation objective.

flags: Flag bits. This field is present if and only if defined in the specification of the objective.

value: This field is to express the actual value of a negotiation or synchronization objective. Its format is defined in the vendor's specification of the objective.

3.8.5. Experimental Objective Options

Option codes G176~191 have been reserved for experimental options. Multiple option codes 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 option codes are also RECOMMENDED for use in documentation examples.

4. Items for Future Work

There are various design questions that are worthy of more work in the near future, as listed below (statically numbered for reference purposes):

- o 5. Need to expand description of the minimum requirements for the specification of an individual discovery, synchronization or negotiation objective.
- 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. However, the authors have not yet decided whether to have a separate document or have it in this document.
- o 7. Cross-check against other ANIMA WG documents for consistency and gaps.

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. GDNP 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 device MUST be capable of proving its identity and authenticating its messages. GDNP relies on a separate certificate-based security mechanism to support authentication, data integrity protection, and anti-replay protection.

Since GDNP 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.

- Privacy and confidentiality

Generally speaking, no personal information is expected to be involved in the negotiation 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.

- DoS Attack Protection

GDNP discovery partly relies on insecure link-local multicast. Since routers participating in GDNP sometimes relay discovery messages from one link to another, this could be a vector for denial of service attacks. Relevant mitigations are specified in [Section 3.3.3](#). Additionally, it is of great importance that firewalls prevent any GDNP messages from entering the domain from an untrusted source.

- Security during bootstrap and discovery

A node cannot authenticate GDNP traffic from other nodes until it has identified the trust anchor and can validate certificates for other nodes. Also, until it has successfully enrolled [[I-D.pritikin-anima-bootstrapping-keyinfra](#)] it cannot assume that other nodes are able to authenticate its own traffic. Therefore, GDNP discovery during the bootstrap phase for a new device will inevitably be insecure and GDNP synchronization and negotiation will be impossible until enrollment is complete.

6. IANA Considerations

[Section 3.4](#) defines the following link-local multicast addresses, which have been assigned by IANA for use by GDNP:

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

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

(Note in draft: alternatively, we could use 224.0.0.1, currently defined as All Systems on this Subnet.)

[Section 3.4](#) defines the following UDP and TCP port, which has been assigned by IANA for use by GDNP:

GDNP Listen Port: (TBD3)

This document defines the General Discovery and Negotiation Protocol (GDNP). The IANA is requested to create a GDNP registry within the unused portion of the DNCP registry [[I-D.ietf-homenet-dncp](#)]. The IANA is also requested to add two new registry tables to the newly-

created GDNP registry. The two tables are the GDNP Messages table and GDNP Options table.

Initial values for these registries are given below. Future assignments are to be made through Standards Action or Specification Required [[RFC5226](#)]. Assignments for each registry consist of a type code value, a name and a document where the usage is defined.

Note to the RFC Editor: In the following tables and in the body of this document, the values G0, G1, etc., should be replaced by the assigned values.

GDNP Messages table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in [Section 3.6](#) in this document:

Type	Name	RFCs
-----+-----+-----		
G0	Reserved	this document
G1	Discovery Message	this document
G2	Response Message	this document
G3	Request Message	this document
G4	Negotiation Message	this document
G5	Negotiation-ending Message	this document
G6	Confirm-waiting Message	this document
G7-31	reserved for future messages	

GDNP Options table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in [Section 3.7](#) and [Section 3.8.1](#) in this document:

Type	Name	RFCs
G32	Divert Option	this document
G33	Accept Option	this document
G34	Decline Option	this document
G35	Waiting Time Option	this document
G36	Device Identity Option	this document
G37	Device IPv4 Address Option	this document
G38	Device IPv6 Address Option	this document
G39	Device FQDN Option	this document
G40~63	Reserved for future GDNP General Options	
G64~127	Reserved for future GDNP Objective Options	
G128~159	Vendor Specific Options	this document
G160~175	Reserved for future use	
G176~191	Experimental Options	this document
G192~???	Reserved for future use	

7. Acknowledgements

A major contribution to the original version of this document was made by Sheng Jiang.

Valuable comments were received from Michael Behringer, Zongpeng Du, Yu Fu, Zhenbin Li, Dimitri Papadimitriou, Michael Richardson, Markus Stenberg, Rene Struik, Dacheng Zhang, and other participants in the NMRG research group and the ANIMA working group.

This document was produced using the xml2rfc tool [[RFC2629](#)].

8. Change log [RFC Editor: Please remove]

[draft-carpenter-anima-discovery-negotiation-protocol-03](#), 2015-04-20:

Removed intrinsic security, required external security

Format changes to allow ADNCP co-existence

Recognized DNS-SD as alternative discovery method

Editorial improvements

[draft-carpenter-anima-discovery-negotiation-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-discovery-negotiation-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-discovery-negotiation-protocol-00](#), combination of [draft-jiang-config-negotiation-ps-03](#) and [draft-jiang-config-negotiation-protocol-02](#), 2014-10-08.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", [BCP 106](#), [RFC 4086](#), June 2005.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [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](#), May 2008.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), January 2012.

9.2. Informative References

- [I-D.behringer-anima-autonomic-control-plane]
Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", [draft-behringer-anima-autonomic-control-plane-02](#) (work in progress), March 2015.
- [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.eckert-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", [draft-eckert-anima-stable-connectivity-01](#) (work in progress), March 2015.
- [I-D.ietf-dnssd-requirements]
Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-SD/mDNS Extensions", [draft-ietf-dnssd-requirements-06](#) (work in progress), March 2015.
- [I-D.ietf-homenet-dncp]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", [draft-ietf-homenet-dncp-01](#) (work in progress), March 2015.
- [I-D.ietf-homenet-hncp]
Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", [draft-ietf-homenet-hncp-04](#) (work in progress), March 2015.
- [I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [draft-ietf-netconf-restconf-04](#) (work in progress), January 2015.
- [I-D.irtf-nmrg-an-gap-analysis]
Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", [draft-irtf-nmrg-an-gap-analysis-05](#) (work in progress), March 2015.

[I-D.irtf-nmrg-autonomic-network-definitions]

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", [draft-irtf-nmrg-autonomic-network-definitions-07](#) (work in progress), March 2015.

[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-05](#) (work in progress), March 2015.

[I-D.pritikin-anima-bootstrapping-keyinfra]

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

[I-D.stenberg-anima-adncp]

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

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSeRVation Protocol (RSVP) -- Version 1 Functional Specification", [RFC 2205](#), September 1997.

[RFC2608] Guttman, E., Perkins, C., Veizades, J., and M. Day, "Service Location Protocol, Version 2", [RFC 2608](#), June 1999.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", [RFC 2629](#), June 1999.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", [RFC 2865](#), June 2000.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", [RFC 3209](#), December 2001.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", [RFC 3315](#), July 2003.

- [RFC3416] Presuhn, R., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, [RFC 3416](#), December 2002.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", [RFC 4861](#), September 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.
- [RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", [RFC 5971](#), October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", [RFC 6241](#), June 2011.
- [RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", [RFC 6733](#), October 2012.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", [RFC 6762](#), February 2013.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", [RFC 6763](#), February 2013.
- [RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", [RFC 6887](#), April 2013.

[Appendix A](#). Capability Analysis of Current Protocols

This appendix discusses various existing protocols with properties related to the above negotiation and synchronisation requirements. 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.

[\[I-D.ietf-dnssd-requirements\]](#) aims to extend this to larger autonomous networks. 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 also signalling 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 signalling across many hops rather than at device-to-device signalling. 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 are unlikely to include YANG processing already.

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

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.

Clearly DNCP does not meet the needs of a general negotiation protocol, especially in its HNCP profile due to the limitation to link-local messages and its strict dependency on 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.

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. Neither is there an obvious combination of protocols that does so. Therefore, this document proposes the design of a protocol that does meet those needs. However, this proposal needs to be compared with alternatives such as extension and adaptation of GIST or DNCP, or combination with IGCP.

Authors' Addresses

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

Email: brian.e.carpenter@gmail.com

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

Email: leo.liubing@huawei.com

