Diameter Maintenance and Extensions   J. Korhonen, Ed.
(DIME)   Broadcom
Internet-Draft   S. Donovan
Intended status: Standards Track   B. Campbell
Expires: June 20, 2014   Oracle
                                                            L. Morand
                                                          Orange Labs
                                                    December 17, 2013

               Diameter Overload Indication Conveyance
                     draft-ietf-dime-ovli-01.txt

   Abstract

      This specification documents a Diameter Overload Control (DOC) base
      solution and the dissemination of the overload report information.

   Requirements

      The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
      "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
      document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction

   This specification defines a base solution for Diameter Overload
   Control (DOC). The requirements for the solution are described and
   discussed in the corresponding design requirements document
   [RFC7068]. Note that the overload control solution defined in this
   specification does not address all the requirements listed in
   [RFC7068]. A number of overload control related features are left
   for the future specifications.

   The solution defined in this specification addresses the Diameter
   overload control between two endpoints (see Section 5.1).
   Furthermore, the solution is designed to apply to existing and future
   Diameter applications, requires no changes to the Diameter base
   protocol [RFC6733] and is deployable in environments where some
   Diameter nodes do not implement the Diameter overload control
   solution defined in this specification.

2. Terminology and Abbreviations

   Server Farm

      A set of Diameter servers that can handle any request for a given
      set of Diameter applications. While these servers support the
      same set of applications, they do not necessarily all have the
      same capacity. An individual server farm might also support a
      subset of the users for a Diameter Realm. A server farm may host
      a single or multiple realms.

   Diameter Routing:

      Diameter Routing between non-adjacent nodes relies on the
      Destination-Realm AVP to determine the Diameter realm in which the
      request needs to be processed. A Destination-Host AVP may also be
      present in the request to address a specific server inside the
      Diameter realm. This function is defined in [RFC6733]. However,
      it is possible to enhance the routing decisions with application
      level knowledge as it done in 3GPP PCC [3GPP.23.203] and NAI-based
      source routing [RFC5729].

   Diameter layer Load Balancing:

      Diameter layer load balancing allows Diameter requests to be
      distributed across the set of servers. Definition of this
      function is outside the scope of this document.

   Topology Hiding:

      Topology Hiding is loosely defined as ensuring that no Diameter
      topology information about a Diameter network can be discovered
      from Diameter messages sent outside a predefined boundary
      (typically an administrative domain). This includes obfuscating
      identifiers and address information of Diameter entities in the
      Diameter network. It can also include hiding the number of
      various Diameter entities in the Diameter network. Identifying
      information can occur in many Diameter Attribute-Value Pairs
      (AVPs), including Origin-Host, Destination-Host, Route-Record,
      Proxy-Info, Session-ID and other AVPs.

   Throttling:

      Throttling is the reduction of the number of requests sent to an
      entity. Throttling can include a client dropping requests, or an
      agent rejecting requests with appropriate error responses.
      Clients and agents can also choose to redirect throttled requests
      to some other entity or entities capable of handling them.

   Reporting Node

      A Diameter node that generates an overload report. (This may or
      may not be the actually overloaded node.)

   Reacting Node

      A Diameter node that consumes and acts upon a report. Note that
      "act upon" does not necessarily mean the reacting node applies an
      abatement algorithm; it might decide to delegate that downstream,
      in which case it also becomes a "reporting node".

   OLR Overload Report.

3. Solution Overview

3.1. Architectural Assumptions

   This section describes the high-level architectural and semantic
   assumptions that underlie the Diameter Overload Control Mechanism.

3.1.1. Application Classification

   The following is a classification of Diameter applications and
   requests. This discussion is meant to document factors that play
   into decisions made by the Diameter identity responsible for handling

overload reports.

Section 8.1 of [RFC6733] defines two state machines that imply two
types of applications, session-less and session-based applications.
The primary difference between these types of applications is the
lifetime of Session-Ids.

For session-based applications, the Session-Id is used to tie
multiple requests into a single session.

In session-less applications, the lifetime of the Session-Id is a
single Diameter transaction, i.e. the session is implicitly
terminated after a single Diameter transaction and a new Session-Id
is generated for each Diameter request.

For the purposes of this discussion, session-less applications are
further divided into two types of applications:

Stateless applications:

   Requests within a stateless application have no relationship to
   each other. The 3GPP defined S13 application is an example of a
   stateless application [3GPP.29.272], where only a Diameter command
   is defined between a client and a server and no state is
   maintained between two consecutive transactions.

Pseudo-session applications:

   Applications that do not rely on the Session-Id AVP for
   correlation of application messages related to the same session
   but use other session-related information in the Diameter requests
   for this purpose. The 3GPP defined Cx application [3GPP.29.229]
   is an example of a pseudo-session application.

The Credit-Control application defined in [RFC4006] is an example of
a Diameter session-based application.

The handling of overload reports must take the type of application
into consideration, as discussed in Section 3.1.2.

3.1.2. Application Type Overload Implications

This section discusses considerations for mitigating overload
reported by a Diameter entity. This discussion focuses on the type
of application. Section 3.1.3 discusses considerations for handling
various request types when the target server is known to be in an
overloaded state.

These discussions assume that the strategy for mitigating the
reported overload is to reduce the overall workload sent to the
overloaded entity. The concept of applying overload treatment to
requests targeted for an overloaded Diameter entity is inherent to
this discussion. The method used to reduce offered load is not
specified here but could include routing requests to another Diameter
entity known to be able to handle them, or it could mean rejecting
certain requests. For a Diameter agent, rejecting requests will
usually mean generating appropriate Diameter error responses. For a
Diameter client, rejecting requests will depend upon the application.
For example, it could mean giving an indication to the entity
requesting the Diameter service that the network is busy and to try
again later.

Stateless applications:

   By definition there is no relationship between individual requests
   in a stateless application. As a result, when a request is sent
   or relayed to an overloaded Diameter entity - either a Diameter
   Server or a Diameter Agent - the sending or relaying entity can
   choose to apply the overload treatment to any request targeted for
   the overloaded entity.

Pseudo-session applications:

   For pseudo-session applications, there is an implied ordering of
   requests. As a result, decisions about which requests towards an
   overloaded entity to reject could take the command code of the
   request into consideration. This generally means that
   transactions later in the sequence of transactions should be given
   more favorable treatment than messages earlier in the sequence.
   This is because more work has already been done by the Diameter
   network for those transactions that occur later in the sequence.
   Rejecting them could result in increasing the load on the network
   as the transactions earlier in the sequence might also need to be
   repeated.

Session-based applications:

   Overload handling for session-based applications must take into
   consideration the work load associated with setting up and
   maintaining a session. As such, the entity sending requests
   towards an overloaded Diameter entity for a session-based
   application might tend to reject new session requests prior to
   rejecting intra-session requests. In addition, session ending
   requests might be given a lower probability of being rejected as
   rejecting session ending requests could result in session status
   being out of sync between the Diameter clients and servers.

Application designers that would decide to reject mid-session
requests will need to consider whether the rejection invalidates
the session and any resulting session clean-up procedures.

3.1.3. Request Transaction Classification

Independent Request:

An independent request is not correlated to any other requests
and, as such, the lifetime of the session-id is constrained to an
individual transaction.

Session-Initiating Request:

A session-initiating request is the initial message that
establishes a Diameter session. The ACR message defined in
[RFC6733] is an example of a session-initiating request.

Correlated Session-Initiating Request:

There are cases when multiple session-initiated requests must be
correlated and managed by the same Diameter server. It is notably
the case in the 3GPP PCC architecture [3GPP.23.203], where
multiple apparently independent Diameter application sessions are
actually correlated and must be handled by the same Diameter
server.

Intra-Session Request:

An intra session request is a request that uses the same
Session-Id than the one used in a previous request. An intra
session request generally needs to be delivered to the server that
handled the session creating request for the session. The STR
message defined in [RFC6733] is an example of an intra-session
requests.

Pseudo-Session Requests:

Pseudo-session requests are independent requests and do not use
the same Session-Id but are correlated by other session-related
information contained in the request. There exists Diameter
applications that define an expected ordering of transactions.
This sequencing of independent transactions results in a pseudo
session. The AIR, MAR and SAR requests in the 3GPP defined Cx
application are examples of pseudo-session requests.

3.1.4. Request Type Overload Implications

   The request classes identified in Section 3.1.3 have implications on
   decisions about which requests should be throttled first. The
   following list of request treatment regarding throttling is provided
   as guidelines for application designers when implementing the
   Diameter overload control mechanism described in this document.
   Exact behavior regarding throttling must be defined per application.

   Independent requests:

      Independent requests can be given equal treatment when making
      throttling decisions.

   Session-initiating requests:

      Session-initiating requests represent more work than independent
      or intra-session requests. Moreover, session-initiating requests
      are typically followed by other related session-related requests.
      As such, as the main objective of the overload control is to
      reduce the total number of requests sent to the overloaded entity,
      throttling decisions might favor allowing intra-session requests
      over session-initiating requests. Individual session-initiating
      requests can be given equal treatment when making throttling
      decisions.

   Correlated session-initiating requests:

      A Request that results in a new binding, where the binding is used
      for routing of subsequent session-initiating requests to the same
      server, represents more work load than other requests. As such,
      these requests might be throttled more frequently than other
      request types.

   Pseudo-session requests:

      Throttling decisions for pseudo-session requests can take into
      consideration where individual requests fit into the overall
      sequence of requests within the pseudo session. Requests that are
      earlier in the sequence might be throttled more aggressively than
      requests that occur later in the sequence.

   Intra-session requests

      There are two classes of intra-sessions requests. The first class
      consists of requests that terminate a session. The second one
      contains the set of requests that are used by the Diameter client
      and server to maintain the ongoing session state. Session

terminating requests should be throttled less aggressively in
order to gracefully terminate sessions, allow clean-up of the
related resources (e.g. session state) and get rid of the need for
other intra-session requests, reducing the session management
impact on the overloaded entity. The default handling of other
intra-session requests might be to treat them equally when making
throttling decisions. There might also be application level
considerations whether some request types are favored over others.

3.1.5. Diameter Agent Behaviour

In the context of the Diameter Overload Indication Conveyance (DOIC)
and reacting to the overload information, the functional behaviour of
Diameter agents in front of servers, especially Diameter proxies,
needs to be common. This is important because agents may actively
participate in the handling of an overload conditions. For example,
they may make intelligent next hop selection decisions based on
overload conditions, or aggregate overload information to be
disseminated downstream. Diameter agents may have other deployment
related tasks that are not defined in the Diameter base protocol
[RFC6733]. These include, among other tasks, topology hiding, or
agent acting as a Server Front End (SFE) for a farm of Diameter
servers.

Since the solution defined in this specification must not break the
Diameter base protocol [RFC6733] at any time, great care has to be
taken not to assume functionality from the Diameter agents that would
break base protocol behavior, or to assume agent functionality beyond
the Diameter base protocol. Effectively this means the following
from a Diameter agent:

o If a Diameter agent presents itself as the "end node", as an agent
   acting as an topology hiding SFE, the agent is the final
   destination of requests initiated by Diameter clients, the
   original source for the corresponding answers and server-initiated
   requests. As a consequence, the DOIC mechanism MUST NOT leak
   information of the Diameter nodes behind it. This requirement
   means that such a Diameter agent acts as a back-to-back-agent for
   DOIC purposes. How the Diameter agent in this case appears to the
   Diameter servers in the farm, is specific to the implementation
   and deployment within the realm the Diameter agent is deployed.

o If the Diameter agent does not impersonate the servers behind it,
   the Diameter dialogue is established between clients and servers
   and any overload information received by a client would be from
   the server identified by the Origin-Host identity contained in the
   Diameter message.

3.1.6. Simplified Example Architecture

   Figure 1 illustrates the simplified architecture for Diameter
   overload information conveyance. See Section 5.1 for more discussion
   and details how different Diameter nodes fit into the architecture
   from the DOIC point of view.

```
    Realm X                                  Same or other Realms
   <-------------------------------->  <--------------------->

   +--^-----+               : (optional) :
   |Diameter|               :            :
   |Server A|--+      .--.   : +---^----+ :      .--.
   +--------+  |    _(    `. : |Diameter| :    _(    `.    +---^----+
            +--(        )--:-| Agent |-:--(        )--|Diameter|
   +--------+  | ( `  .  )  ) : +-----^--+ : ( `  .  )  )  | Client |
   |Diameter|--+  `--(___.-'  :            :  `--(___.-'  +-----^--+
   |Server B|               :            :
   +---^----+               :            :

                   End-to-end Overload Indication
         1)   <-------------------------------------------->
                     Diameter Application Y


            Overload Indication A   Overload Indication A'
         2)   <--------------------> <--------------------->
              standard base protocol  standard base protocol
```
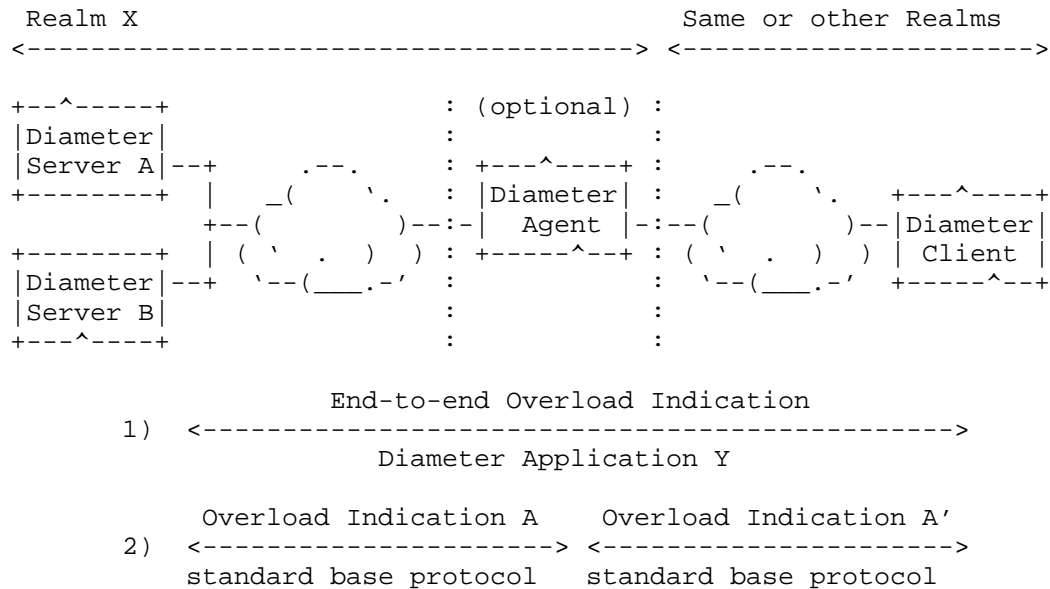
        Figure 1: Simplified architecture choices for overload indication
                              delivery

   In Figure 1, the Diameter overload indication can be conveyed (1)
   end-to-end between servers and clients or (2) between servers and
   Diameter agent inside the realm and then between the Diameter agent
   and the clients when the Diameter agent acting as back-to-back-agent
   for DOIC purposes.

3.2. Conveyance of the Overload Indication

   The following sections describe new Diameter AVPs used for sending
   overload reports, and for declaring support for certain DOIC
   features.

3.2.1. DOIC Capability Discovery

Support of DOIC may be specified as part of the functionality
supported by a new Diameter application. In this way, support of the
considered Diameter application (discovered during capabilities
exchange phase as defined in Diameter base protocol [RFC6733])
indicates implicit support of the DOIC mechanism.

When the DOIC mechanism is introduced in existing Diameter
applications, a specific capability discovery mechanism is required.
The "DOIC capability discovery mechanism" is based on the presence of
specific optional AVPs in the Diameter messages, such as the OC-
Supported-Features AVP (see Section 4.1). Although the OC-Supported-
Features AVP can be used to advertise a certain set of new or
existing Diameter overload control capabilities, it is not a
versioning solution per se, however, it can be used to achieve the
same result.

From the Diameter overload control functionality point of view, the
"Reacting node" is the requester of the overload report information
and the "Reporting node" is the provider of the overload report. The
OC-Supported-Features AVP in the request message is always
interpreted as an announcement of "DOIC supported capabilities". The
OC-Supported-Features AVP in the answer is also interpreted as a
report of "DOIC supported capabilities" and at least one of supported
capabilities MUST be common with the "Reacting node" (see
Section 4.1).

3.3. Overload Condition Indication

Diameter nodes can request a reduction in offered load by indicating
an overload condition in the form of an overload report. The
overload report contains information about how much load should be
reduced, and may contain other information about the overload
condition. This information is conveyed in Diameter Attribute Value
Pairs (AVPs).

Certain new AVPs may also be used to declare certain DOIC
capabilities and extensions.

4. Attribute Value Pairs

This section describes the encoding and semantics of the Diameter
Overload Indication Attribute Value Pairs (AVPs) defined in this
document.

4.1. OC-Supported-Features AVP

   The OC-Supported-Features AVP (AVP code TBD1) is type of Grouped and
   serves for two purposes. First, it announces node's support for the
   DOIC in general. Second, it contains the description of the
   supported DOIC features of the sending node. The OC-Supported-
   Features AVP SHOULD be included into every Diameter message a DOIC
   supporting node sends (and intends to use for DOIC purposes).

   OC-Supported-Features ::= < AVP Header: TBD1 >
                             < OC-Sequence-Number >
                             [OC-Feature-Vector]
                           * [AVP]

   The OC-Sequence-Number AVP is used to indicate whether the contents
   of the OC-Supported-Features AVP has changed since last time the node
   included the OC-Supported-Features AVP (see Section 4.4). Although
   sending the OC-Sequence-Number AVP is mandatory in the OC-Supported-
   Features AVP, the receiving node MAY always choose to ignore the
   sequence number if it can determine the feature support changes
   otherwise.

   The OC-Feature-Vector sub-AVP is used to announced the DOIC features
   supported by the endpoint, in the form of a flag bits field in which
   each bit announces one feature or capability supported by the node
   (see Section 4.2). The absence of the OC-Feature-Vector AVP
   indicates that only the default traffic abatement algorithm described
   in this specification is supported.

   A reacting node includes this AVP to indicate its capabilities to a
   reporting node. For example, the endpoint (reacting node) may
   indicate which (future defined) traffic abatement algorithms it
   supports in addition to the default.

   During the message exchange the overload control endpoints express
   their common set of supported capabilities. The reacting node
   includes the OC-Supported-Features AVP that announces what it
   supports. The reporting node that sends the answer also includes the
   OC-Supported-Features AVP that describes the capabilities it
   supports. The set of capabilities advertised by the reporting node
   depends on local policies. At least one of the announced
   capabilities MUST match mutually. If there is no single matching
   capability the reacting node MUST act as if it does not implement
   DOIC and cease inserting any DOIC related AVPs into any Diameter
   messages with this specific reacting node.

4.2. OC-Feature-Vector AVP

   The OC-Feature-Vector AVP (AVP code TBD6) is type of Unsigned64 and
   contains a 64 bit flags field of announced capabilities of an
   overload control endpoint. The value of zero (0) is reserved.

   The following capabilities are defined in this document:

   OLR_DEFAULT_ALGO (0x0000000000000001)

      When this flag is set by the overload control endpoint it means
      that the default traffic abatement (loss) algorithm is supported.

4.3. OC-OLR AVP

   The OC-OLR AVP (AVP code TBD2) is type of Grouped and contains the
   necessary information to convey an overload report. The OC-OLR AVP
   does not contain explicit information to which application it applies
   to and who inserted the AVP or whom the specific OC-OLR AVP concerns
   to. Both these information is implicitly learned from the
   encapsulating Diameter message/command. The application the OC-OLR
   AVP applies to is the same as the Application-Id found in the
   Diameter message header. The identity the OC-OLR AVP concerns is
   determined from the Origin-Host AVP (and Origin-Realm AVP as well)
   found from the encapsulating Diameter command. The OC-OLR AVP is
   intended to be sent only by a reporting node.

   OC-OLR ::= < AVP Header: TBD2 >
              < OC-Sequence-Number >
              [OC-Report-Type]
              [OC-Reduction-Percentage]
              [OC-Validity-Duration]
            * [AVP]

   The Sequence-Number AVP indicates the "freshness" of the OC-OLR AVP.
   It is possible to replay the same OC-OLR AVP multiple times between
   the overload control endpoints, however, when the OC-OLR AVP content
   changes or sending endpoint otherwise wants the receiving endpoint to
   update its overload control information, then the OC-Sequence-Number
   AVP MUST contain a new greater value than the previously received.
   The receiver SHOULD discard an OC-OLR AVP with a sequence number that
   is less than previously received one.

   Note that if a Diameter command were to contain multiple OC-OLR AVPs
   they all MUST have different OC-Report-Type AVP value. OC-OLR AVPs
   with unknown values SHOULD be silently discarded and the event SHOULD
   be logged.

The OC-OLR AVP can be expanded with optional sub-AVPs only if a
legacy implementation can safely ignore them without breaking
backward compatibility for the given OC-Report-Type AVP value implied
report handling semantics. If the new sub-AVPs imply new semantics
for the report handling, then a new OC-Report-Type AVP value MUST be
defined.

4.4. OC-Sequence-Number AVP

The OC-Sequence-Number AVP (AVP code TBD3) is type of Time. Its
usage in the context of the overload control is described in Sections
4.1 and 4.3.

From the functionality point of view, the OC-Sequence-Number AVP MUST
be used as a non-volatile increasing counter between two overload
control endpoints (neglecting the fact that the contents of the AVP
is a 64-bit NTP timestamp [RFC5905]). The sequence number is only
required to be unique between two overload control endpoints.
Sequence numbers are treated in uni-directional manner, i.e. two
sequence numbers on each direction between two endpoints are not
related or correlated.

When generating sequence numbers, the new sequence number MUST be
greater than any sequence number previously seen between two
endpoints within a time window that tolerates the wraparound of the
NTP timestamp (i.e. approximately 68 years).

4.5. OC-Validity-Duration AVP

The OC-Validity-Duration AVP (AVP code TBD4) is type of Unsigned32
and describes the number of seconds the "new and fresh" OC-OLR AVP
and its content is valid since the reception of the new OC-OLR AVP.
The default value for the OC-Validity-Duration AVP value is 5 (i.e.,
5 seconds). When the OC-Validity-Duration AVP is not present in the
OC-OLR AVP, the default value applies. Validity duration values 0
(i.e., 0 seconds) and above 86400 (i.e., 24 hours) MUST NOT be used.
Invalid validity duration values are treated as if the OC-Validity-
Duration AVP were not present.

A timeout of the overload report has specific concerns that need to
be taken into account by the endpoint acting on the earlier received
overload report(s). Section 4.7 discusses the impacts of timeout in
the scope of the traffic abatement algorithms.

As a general guidance for implementations it is RECOMMENDED never to
let any overload report to timeout. Following to this rule, an
overload endpoint should explicitly signal the end of overload
condition and not rely on the expiration of the validity time of the

overload report in the reacting node. This leaves no need for the
reacting node to reason or guess the overload condition of the
reporting node.

4.6. OC-Report-Type AVP

The OC-Report-Type AVP (AVP code TBD5) is type of Enumerated. The
value of the AVP describes what the overload report concerns. The
following values are initially defined:

0 A host report. The overload treatment should apply to requests
   the reacting node knows that will reach the overloaded node. For
   example, requests with a Destination-Host AVP indicating the
   endpoint. The reacting node learns the "host" implicitly from the
   Origin-Host AVP of the received message that contained the OC-OLR
   AVP.

1 A realm report. The overload treatment should apply to all
   requests bound for the overloaded realm. The reacting node learns
   the "realm" implicitly from the Origin-Realm AVP of the received
   message that contained the OC-OLR AVP.

The default value of the OC-Report-Type AVP is 0 (i.e. the host
report).

The OC-Report-Type AVP is envisioned to be useful for situations
where a reacting node needs to apply different overload treatments
for different "types" of overload. For example, the reacting node(s)
might need to throttle differently requests sent to a specific server
(identified by the Destination-Host AVP in the request) and requests
that can be handled by any server in a realm. The example in
Appendix B.1 illustrates this usage.

When defining new report type values, the corresponding specification
MUST define the semantics of the new report types and how they affect
the OC-OLR AVP handling. The specification MUST also reserve a
corresponding new feature, see the OC-Supported-Features and OC-
Feature-Vector AVPs.

4.7. OC-Reduction-Percentage AVP

The OC-Reduction-Percentage AVP (AVP code TBD8) is type of Unsigned32
and describes the percentage of the traffic that the sender is
requested to reduce, compared to what it otherwise would have sent.
The OC-Reduction-Percentage AVP applies to the default (loss like)
algorithm specified in this specification. However, the AVP can be
reused for future abatement algorithms, if its semantics fit into the
new algorithm.

The value of the Reduction-Percentage AVP is between zero (0) and one
hundred (100). Values greater than 100 are interpreted as 100. The
value of 100 means that no traffic is expected, i.e. the reporting
node is under a severe load and ceases to process any new messages.
The value of 0 means that the reporting node is in a stable state and
has no requests to the other endpoint to apply any traffic abatement.
The default value of the OC-Reduction-Percentage AVP is 0. When the
OC-Reduction-Percentage AVP is not present in the overload report,
the default value applies.

If an overload control endpoint comes out of the 100 percent traffic
reduction as a result of the overload report timing out, the
following concerns are RECOMMENDED to be applied. The reacting node
sending the traffic should be conservative and, for example, first
send "probe" messages to learn the overload condition of the
overloaded node before converging to any traffic amount/rate decided
by the sender. Similar concerns apply in all cases when the overload
report times out unless the previous overload report stated 0 percent
reduction.

4.8. Attribute Value Pair flag rules

| | AVP | Section | | AVP flag rules | |
Attribute Name	Code	Defined	Value Type	MUST	MUST NOT
OC-Supported-Features	TBD1	x.x	Grouped		V
OC-OLR	TBD2	x.x	Grouped		V
OC-Sequence-Number	TBD3	x.x	Time		V
OC-Validity-Duration	TBD4	x.x	Unsigned32		V
OC-Report-Type	TBD5	x.x	Enumerated		V
OC-Reduction -Percentage	TBD8	x.x	Unsigned32		V
OC-Feature-Vector	TBD6	x.x	Unsigned64		V

As described in the Diameter base protocol [RFC6733], the M-bit
setting for a given AVP is relevant to an application and each

command within that application that includes the AVP.

The Diameter overload control AVPs SHOULD always be sent with the
M-bit cleared when used within existing Diameter applications to
avoid backward compatibility issues. Otherwise, when reused in newly
defined Diameter applications, the DOC related AVPs SHOULD have the
M-bit set.

5. Overload Control Operation

5.1. Overload Control Endpoints

The overload control solution can be considered as an overlay on top
of an arbitrary Diameter network. The overload control information
is exchanged over on a "DOIC association" established between two
communication endpoints. The endpoints, namely the "reacting node"
and the "reporting node" do not need to be adjacent Diameter peer
nodes, nor they need to be the end-to-end Diameter nodes in a typical
"client-server" deployment with multiple intermediate Diameter agent
nodes in between. The overload control endpoints are the two
Diameter nodes that decide to exchange overload control information
between each other. How the endpoints are determined is specific to
a deployment, a Diameter node role in that deployment and local
configuration.

The following diagrams illustrate the concept of Diameter Overload
End-Points and how they differ from the standard [RFC6733] defined
client, server and agent Diameter nodes. The following is the key to
the elements in the diagrams:

C Diameter client as defined in [RFC6733].

S Diameter server as defined in [RFC6733].

A Diameter agent, in either a relay or proxy mode, as defined in
   [RFC6733].

DEP Diameter Overload End-Point as defined in this document. In the
   following figures a DEP may terminate two different DOIC
   associations being a reporter and reactor at the same time.

Diameter Session A Diameter session as defined in [RFC6733].

DOIC Association A DOIC association exists between two Diameter
   Overload End-Points. One of the end-points is the overload
   reporter and the other is the overload reactor.

Figure 2 illustrates the most basic configuration where a client is
connected directly to a server. In this case, the Diameter session
and the DOIC association are both between the client and server.

```
+-----+              +-----+
|  C  |              |  S  |
+-----+              +-----+
| DEP |              | DEP |
+--+--+              +--+--+
   |                    |
   |                    |
   |{Diameter Session}  |
   |                    |
   |{DOIC Association}  |
   |                    |
```
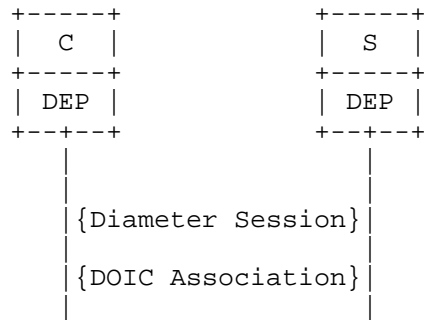
                 Figure 2: Basic DOIC deployment

In Figure 3 there is an agent that is not participating directly in
the exchange of overload reports. As a result, the Diameter session
and the DOIC association are still established between the client and
the server.

```
+-----+              +-----+              +-----+
|  C  |              |  A  |              |  S  |
+-----+              +--+--+              +-----+
| DEP |                 |                 | DEP |
+--+--+                 |                 +--+--+
   |                    |                    |
   |                    |                    |
   |----------{Diameter Session}---------|
   |                    |                    |
   |----------{DOIC Association}---------|
   |                    |                    |
```
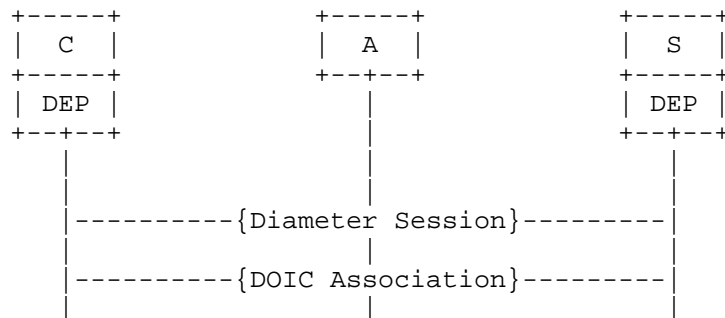
          Figure 3: DOIC deployment with non participating agent

Figure 4 illustrates the case where the client does not support
Diameter overload. In this case, the DOIC association is between the
agent and the server. The agent handles the role of the reactor for
overload reports generated by the server.

```
+-----+             +-----+             +-----+
| C   |             | A   |             | S   |
+--+--+             +-----+             +-----+
   |                | DEP |             | DEP |
   |                +--+--+             +--+--+
   |                   |                   |
   |                   |                   |
   |----------{Diameter Session}---------|
   |                   |                   |
   |                   |{DOIC Association}|
   |                   |                   |
```
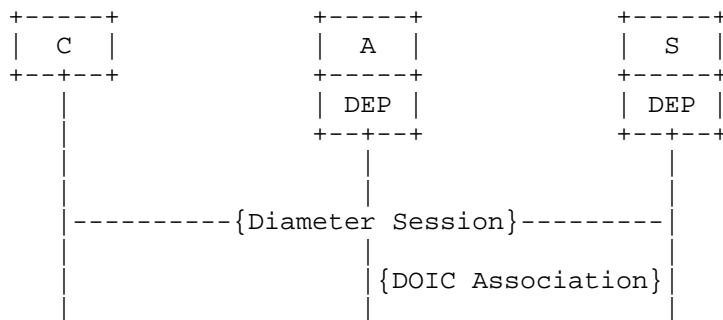
      Figure 4: DOIC deployment with non-DOIC client and DOIC enabled agent

   In Figure 5 there is a DOIC association between the client and the
   agent and a second DOIC association between the agent and the server.
   One use case requiring this configuration is when the agent is
   serving as a SFE for a set of servers.

```
+-----+             +-----+             +-----+
| C   |             | A   |             | S   |
+-----+             +-----+             +-----+
| DEP |             | DEP |             | DEP |
+--+--+             +--+--+             +--+--+
   |                   |                   |
   |                   |                   |
   |----------{Diameter Session}---------|
   |                   |                   |
   |{DOIC Association}|{DOIC Association}|
   |                   |            and/or
   |----------{DOIC Association}---------|
   |                   |                   |
```

            Figure 5: A deployment where all nodes support DOIC

   Figure 6 illustrates a deployment where some clients support Diameter
   overload control and some do not. In this case the agent must
   support Diameter overload control for the non supporting client. It
   might also need to have a DOIC association with the server, as shown
   here, to handle overload for a server farm and/or for managing Realm
   overload.

```
+-----+            +-----+            +-----+            +-----+
| C1  |            | C2  |            |  A  |            |  S  |
+-----+            +--+--+            +-----+            +-----+
| DEP |               |               | DEP |            | DEP |
+--+--+               |               +--+--+            +--+--+
   |                  |                  |                  |
   |                  |                  |                  |
   |------------------{Diameter Session}------------------|
   |                  |                  |                  |
   |                  |--------{Diameter Session}----------|
   |                  |                  |                  |
   |---------{DOIC Association}----------|{DOIC Association}|
   |                  |                  |        and/or
   |------------------{DOIC Association}------------------|
   |                  |                  |                  |
```

        Figure 6: A deployment with DOIC and non-DOIC supporting clients

   Figure 7 illustrates a deployment where some agents support Diameter
   overload control and others do not.

```
+-----+            +-----+            +-----+            +-----+
|  C  |            |  A  |            |  A  |            |  S  |
+-----+            +--+--+            +-----+            +-----+
| DEP |               |               | DEP |            | DEP |
+--+--+               |               +--+--+            +--+--+
   |                  |                  |                  |
   |                  |                  |                  |
   |------------------{Diameter Session}------------------|
   |                  |                  |                  |
   |                  |                  |                  |
   |---------{DOIC Association}----------|{DOIC Association}|
   |                  |                  |        and/or
   |------------------{DOIC Association}------------------|
   |                  |                  |                  |
```
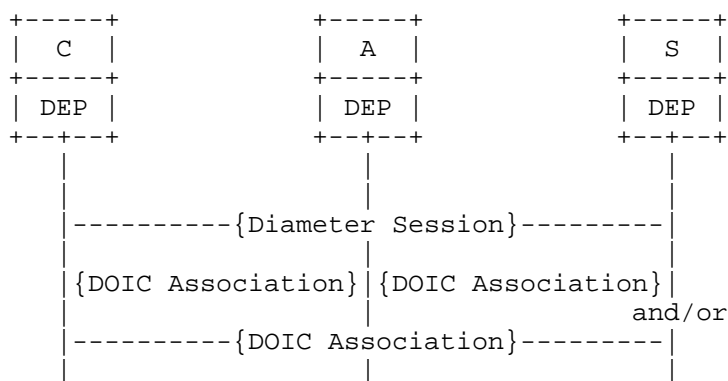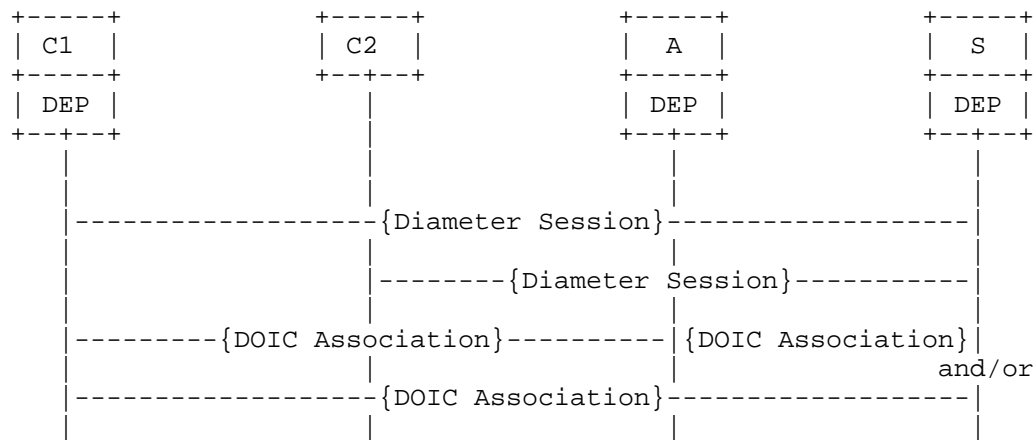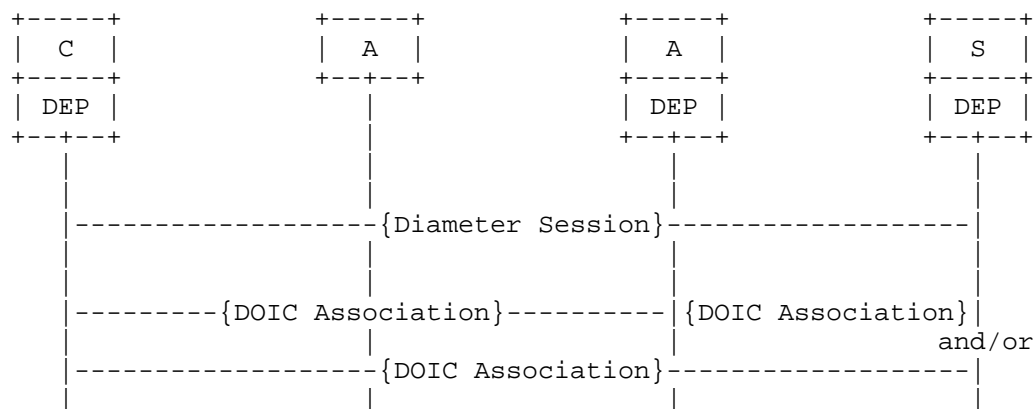
        Figure 7: A deployment with DOIC and non-DOIC supporting agents

5.2. Piggybacking Principle

   The overload control AVPs defined in this specification have been
   designed to be piggybacked on top of existing application message
   exchanges. This is made possible by adding overload control top
   level AVPs, the OC-OLR AVP and the OC-Supported-Features AVP as
   optional AVPs into existing commands when the corresponding Command
   Code Format (CCF) specification allows adding new optional AVPs (see

Section 1.3.4 of [RFC6733]).

When added to existing commands, both OC-Feature-Vector and OC-OLR
AVPs SHOULD have the M-bit flag cleared to avoid backward
compatibility issues.

A new application specification can incorporate the overload control
mechanism specified in this document by making it mandatory to
implement for the application and referencing this specification
normatively. In such a case, the OC-Feature-Vector and OC-OLR AVPs
reused in newly defined Diameter applications SHOULD have the M-bit
flag set. However, it is the responsibility of the Diameter
application designers to define how overload control mechanisms works
on that application.

Note that the overload control solution does not have fixed server
and client roles. The endpoint role is determined based on the
message type: whether the message is a request (i.e. sent by a
"reacting node") or an answer (i.e. send by a "reporting node").
Therefore, in a typical "client-server" deployment, the "client" MAY
report its overload condition to the "server" for any server
initiated message exchange. An example of such is the server
requesting a re-authentication from a client.

5.3. Capability Announcement

Since the overload control solution relies on the piggybacking
principle for the overload reporting and the overload control
endpoint are likely not adjacent peers, finding out whether the other
endpoint supports the overload control or what is the common traffic
abatement algorithm to apply for the traffic. The approach defined
in this specification for the end-to-end capability announcement
relies on the exchange of the OC-Supported-Features between the
endpoints. The feature announcement solution also works when carried
out on existing applications. For the newly defined application the
negotiation can be more exact based on the application specification.
The announced set of capabilities MUST NOT change during the life
time of the Diameter session (or transaction in case of non-session
maintaining applications).

5.3.1. Reacting Node Endpoint Considerations

The basic principle is that the request message initiating endpoint
(i.e. the "reacting node") announces its support for the overload
control mechanism by including in the request message the OC-
Supported-Features AVP with those capabilities it supports and is
willing to use for this Diameter session (or transaction in a case of
a non-session state maintaining applications, see Section 3.1.2 for

more details on Diameter sessions). It is RECOMMENDED that the
request message initiating endpoint includes the capability
announcement into every request regardless it has had prior message
exchanges with the give remote endpoint. In a case of a Diameter
session maintaining application, sending the OC-Supported-Features
AVP in every message is not really necessary after the initial
capability announcement or until there is a change in supported
features.

Once the endpoint that initiated the request message receives an
answer message from the remote endpoint, it can detect from the
received answer message whether the remote endpoint supports the
overload control solution and in a case it does, what features are
supported. The support for the overload control solution is based on
the presence of the OC-Supported-Features AVP in the Diameter answer
for existing application.

5.3.2. Reporting Node Endpoint Considerations

When a remote endpoint (i.e. a "reporting node") receives a request
message, it can detect whether the request message initiating
endpoint supports the overload control solution based on the presence
of the OC-Supported-Features AVP. For the newly defined applications
the overload control solution support can be part of the application
specification. Based on the content of the OC-Supported-Features AVP
the request message receiving endpoint knows what overload control
functionality the other endpoint supports and then act accordingly
for the subsequent answer messages it initiates. The answer message
initiating endpoint MAY announce as many supported capabilities as it
has (the announced set is a subject to local policy and
configuration). However, at least one of the announced capabilities
MUST be the same as received in the request message.

The answer message initiating endpoint MUST NOT include any overload
control solution defined AVPs into its answer messages if the request
message initiating endpoint has not indicated support at the
beginning of the created session (or transaction in a case of non-
session state maintaining applications). The same also applies if
none of the announced capabilities match between the two endpoints.

5.4. Protocol Extensibility

The overload control solution can be extended, e.g. with new traffic
abatement algorithms or new functionality. The new features and
algorithms MUST be registered with the IANA and for the possible use
with the OC-Supported-Features for announcing the support for the new
features (see Section 7 for the required procedures).

It should be noted that [RFC6733] defined Grouped AVP extension
mechanisms also apply. This allows, for example, defining a new
feature that is mandatory to understand even when piggybacked on an
existing applications. More specifically, the sub-AVPs inside the
OC-OLR AVP MAY have the M-bit set. However, when overload control
AVPs are piggybacked on top of an existing applications, setting
M-bit in sub-AVPs is NOT RECOMMENDED.

5.5. Overload Report Processing

5.5.1. Overload Control State

   Both reacting and reporting nodes maintain an overload condition
   state for each endpoint (a host or a realm) they communicate with and
   both endpoints have announced support for DOIC. See Sections 4.1 and
   5.3 for discussion about how the support for DOIC is determined. The
   overload condition state SHOULD be able to make a difference between
   a realm and a specific host in that realm.

   The overload condition state could include the following information
   (per host or realm):

   o The endpoint information (Diameter identity of the realm and/or
      host, application identifier, etc)

   o Reduction percentage

   o Validity period timer

   o Sequence number

   o Supported/selected traffic abatement algorithm

   The overload control state information SHOULD be maintained as long
   as the other endpoint is known to support DOIC (based on the presence
   of the DOIC AVPs or by a future application specification).

5.5.2. Reacting Node Considerations

   Once a reacting node receives an OC-OLR AVP from a reporting node, it
   applies the traffic abatement based on the commonly supported
   algorithm with the reporting node and the current overload condition.
   The reacting node learns the reporting node supported abatement
   algorithms directly from the received answer message containing the
   OC-Supported-Features AVP or indirectly remembering the previously
   used traffic abatement algorithm with the given reporting node.

   The received OC-Supported-Features AVP does not change the existing

overload condition and/or traffic abatement algorithm settings if the
OC-Sequence-Number AVP contains a value that is equal to the
previously received/recorded one. If the OC-Supported-Features AVP
is received for the first time for the reporting node or the OC-
Sequence-Number AVP value is less than the previously received/
recorded one (and is outside the valid overflow window), then either
the sequence number is stale (e.g. an intentional or unintentional
replay) and SHOULD be silently discarded.

The OC-OLR AVP contains the necessary information of the overload
condition on the reporting node. Similarly to the OC-Supported-
Features's sequence numbering, the OC-OLR AVP also has the OC-
Sequence-Number AVP and its handling is similar to the one in the OC-
Supported-Features AVP. The reacting node MUST update its overload
condition state whenever receiving the OC-OLR AVP for the first time
or the OC-Sequence-Number sub-AVP indicates a change in the OC-OLR
AVP.

As described in Section 4.3, the OC-OLR AVP contains the necessary
information of the overload condition on the reporting node.

From the OC-Report-Type AVP contained in the OC-OLR AVP, the reacting
node learns whether the overload condition report concerns a specific
host (as identified by the Origin-Host AVP of the answer message
containing the OC-OLR AVP) or the entire realm (as identified by the
Origin-Realm AVP of the answer message containing the OC-OLR AVP).
The reacting node learns the Diameter application to which the
overload report applies from the Application-ID of the answer message
containing the OC-OLR AVP. The reacting node MUST use this
information as an input for its traffic abatement algorithm. The
idea is that the reacting node applies different handling of the
traffic abatement, whether sent request messages are targeted to a
specific host (identified by the Diameter-Host AVP in the request) or
to any host in a realm (when only the Destination-Realm AVP is
present in the request). Note that future specifications MAY define
new OC-Report-Type AVP values that imply different handling of the
OC-OLR AVP. For example, in a form of new additional AVPs inside the
Grouped OC-OLR AVP that would define report target in a finer
granularity than just a host.

In the context of this specification and the default traffic
abatement algorithm, the OC-Reduction-Percentage AVP value MUST be
interpreted in the following way:

value == 0

   Indicates explicitly the end of overload condition and the
   reacting node SHOULD NOT apply the traffic abatement algorithm
   procedures anymore for the given reporting node (or realm).

value == 100

   Indicates that the reporting node (or realm) does not want to
   receive any traffic from the reacting node for the application the
   report concerns. The reacting node MUST do all measure not to
   send traffic to the reporting node (or realm) as long as the
   overload condition changes or expires.

0 < value < 100

   Indicates that the reporting node urges the reacting node to
   reduce its traffic by a given percentage. For example if the
   reacting node has been sending 100 packets per second to the
   reporting node, then a reception of OC-Reduction-Percentage value
   of 10 would mean that from now on the reacting node MUST only send
   90 packets per second. How the reacting node achieves the "true
   reduction" transactions leading to the sent request messages is up
   to the implementation. The reacting node MAY simply drop every
   10th packet from its output queue and let the generic application
   logic try to recover from it.

If the OC-OLR AVP is received for the first time, the reacting node
MUST create an overload condition state associated with the related
realm or a specific host in the realm identified in the message
carrying the OC-OLR AVP, as described in Section 5.5.1.

If the value of the OC-Sequence-Number AVP contained in the received
OC-OLR AVP is equal to or less than the value stored in an existing
overload condition state, the received OC-OLR AVP SHOULD be silently
discarded. If the value of the OC-Sequence-Number AVP contained in
the received OC-OLR AVP is greater than the value stored in an
existing overload condition state or there is no previously recorded
sequence number, the reacting node MUST update the overload condition
state associated with the realm or the specific node is the realm.

When an overload condition state is created or updated, the reacting
node MUST apply the traffic abatement requested in the OC-OLR AVP
using the algorithm announced in the OC-Supported-Features AVP
contained in the received answer message along with the OC-OLR AVP.

The validity duration of the overload information contained in the
OC-OLR AVP is either explicitly indicated in the OC-Validity-Duration

AVP or is implicitly equals to the default value (5 seconds) if the OC-Validity-Duration AVP is absent of the OC-OLR AVP. The reacting node MUST maintain the validity duration in the overload condition state. Once the validity duration times out, the reacting node MUST assume the overload condition reported in a previous OC-OLR AVP has ended.

5.5.3. Reporting Node Considerations

A reporting node is a Diameter node inserting an OC-OLR AVP in a Diameter message in order to inform a reacting node about an overload condition and request Diameter traffic abatement.

The operation on the reporting node is rather straight forward. The reporting node learns the capabilities of the reacting node when it receives the OC-Supported-Features AVP as part of any Diameter request message. If the reporting node shares at least one common feature with the reacting node, then the DOIC can be enabled between these two endpoints. See Section 5.3 for further discussion on the capability and feature announcement between two endpoints.

When a traffic reduction is required due to an overload condition and the overload control solution is supported by the sender of the Diameter request, the reporting node MUST include an OC-Supported-Features AVP and an OC-OLR AVP in the corresponding Diameter answer. The OC-OLR AVP contains the required traffic reduction and the OC-Supported-Features AVP indicates the traffic abatement algorithm to apply. This algorithm MUST be one of the algorithms advertised by the request sender.

A reporting node MAY rely on the OC-Validity-Duration AVP values for the implicit overload condition state cleanup on the reacting node. However, it is RECOMMENDED that the reporting node always explicitly indicates the end of a overload condition.

6. Transport Considerations

In order to reduce overload control introduced additional AVP and message processing it might be desirable/beneficial to signal whether the Diameter command carries overload control information that should be of interest of an overload aware Diameter node.

Should such indication be include is not part of this specification. It has not either been concluded at what layer such possible indication should be. Obvious candidates include transport layer protocols (e.g., SCTP PPID or TCP flags) or Diameter command header flags.

7. IANA Considerations

7.1. AVP codes

   New AVPs defined by this specification are listed in Section 4. All
   AVP codes allocated from the 'Authentication, Authorization, and
   Accounting (AAA) Parameters' AVP Codes registry.

7.2. New registries

   Three new registries are needed under the 'Authentication,
   Authorization, and Accounting (AAA) Parameters' registry.

   Section 4.2 defines a new "Overload Control Feature Vector" registry
   including the initial assignments. New values can be added into the
   registry using the Specification Required policy [RFC5226]. See
   Section 4.2 for the initial assignment in the registry.

   Section 4.6 defines a new "Overload Report Type" registry with its
   initial assignments. New types can be added using the Specification
   Required policy [RFC5226].

8. Security Considerations

   This mechanism gives Diameter nodes the ability to request that
   downstream nodes send fewer Diameter requests. Nodes do this by
   exchanging overload reports that directly affect this reduction.
   This exchange is potentially subject to multiple methods of attack,
   and has the potential to be used as a Denial-of-Service (DoS) attack
   vector.

   Overload reports may contain information about the topology and
   current status of a Diameter network. This information is
   potentially sensitive. Network operators may wish to control
   disclosure of overload reports to unauthorized parties to avoid its
   use for competitive intelligence or to target attacks.

   Diameter does not include features to provide end-to-end
   authentication, integrity protection, or confidentiality. This may
   cause complications when sending overload reports between non-
   adjacent nodes.

8.1. Potential Threat Modes

   The Diameter protocol involves transactions in the form of requests
   and answers exchanged between clients and servers. These clients and
   servers may be peers, that is,they may share a direct transport (e.g.

TCP or SCTP) connection, or the messages may traverse one or more
intermediaries, known as Diameter Agents. Diameter nodes use TLS,
DTLS, or IPSec to authenticate peers, and to provide confidentiality
and integrity protection of traffic between peers. Nodes can make
authorization decisions based on the peer identities authenticated at
the transport layer.

When agents are involved, this presents an effectively hop-by-hop
trust model. That is, a Diameter client or server can authorize an
agent for certain actions, but it must trust that agent to make
appropriate authorization decisions about its peers, and so on.

Since confidentiality and integrity protection occurs at the
transport layer. Agents can read, and perhaps modify, any part of a
Diameter message, including an overload report.

There are several ways an attacker might attempt to exploit the
overload control mechanism. An unauthorized third party might inject
an overload report into the network. If this third party is upstream
of an agent, and that agent fails to apply proper authorization
policies, downstream nodes may mistakenly trust the report. This
attack is at least partially mitigated by the assumption that nodes
include overload reports in Diameter answers but not in requests.
This requires an attacker to have knowledge of the original request
in order to construct a response. Therefore, implementations SHOULD
validate that an answer containing an overload report is a properly
constructed response to a pending request prior to acting on the
overload report.

A similar attack involves an otherwise authorized Diameter node that
sends an inappropriate overload report. For example, a server for
the realm "example.com" might send an overload report indicating that
a competitor's realm "example.net" is overloaded. If other nodes act
on the report, they may falsely believe that "example.net" is
overloaded, effectively reducing that realm's capacity. Therefore,
it's critical that nodes validate that an overload report received
from a peer actually falls within that peer's responsibility before
acting on the report or forwarding the report to other peers. For
example, an overload report from an peer that applies to a realm not
handled by that peer is suspect.

An attacker might use the information in an overload report to assist
in certain attacks. For example, an attacker could use information
about current overload conditions to time a DoS attack for maximum
effect, or use subsequent overload reports as a feedback mechanism to
learn the results of a previous or ongoing attack.

8.2. Denial of Service Attacks

   Diameter overload reports can cause a node to cease sending some or
   all Diameter requests for an extended period. This makes them a
   tempting vector for DoS tacks. Furthermore, since Diameter is almost
   always used in support of other protocols, a DoS attack on Diameter
   is likely to impact those protocols as well. Therefore, Diameter
   nodes MUST NOT honor or forward overload reports from unauthorized or
   otherwise untrusted sources.

8.3. Non-Compliant Nodes

   When a Diameter node sends an overload report, it cannot assume that
   all nodes will comply. A non-compliant node might continue to send
   requests with no reduction in load. Requirement 28 [RFC7068]
   indicates that the overload control solution cannot assume that all
   Diameter nodes in a network are necessarily trusted, and that
   malicious nodes not be allowed to take advantage of the overload
   control mechanism to get more than their fair share of service.

   In the absence of an overload control mechanism, Diameter nodes need
   to implement strategies to protect themselves from floods of
   requests, and to make sure that a disproportionate load from one
   source does not prevent other sources from receiving service. For
   example, a Diameter server might reject a certain percentage of
   requests from sources that exceed certain limits. Overload control
   can be thought of as an optimization for such strategies, where
   downstream nodes never send the excess requests in the first place.
   However, the presence of an overload control mechanism does not
   remove the need for these other protection strategies.

8.4. End-to End-Security Issues

   The lack of end-to-end security features makes it far more difficult
   to establish trust in overload reports that originate from non-
   adjacent nodes. Any agents in the message path may insert or modify
   overload reports. Nodes must trust that their adjacent peers perform
   proper checks on overload reports from their peers, and so on,
   creating a transitive-trust requirement extending for potentially
   long chains of nodes. Network operators must determine if this
   transitive trust requirement is acceptable for their deployments.
   Nodes supporting Diameter overload control MUST give operators the
   ability to select which peers are trusted to deliver overload
   reports, and whether they are trusted to forward overload reports
   from non-adjacent nodes.

   The lack of end-to-end confidentiality protection means that any
   Diameter agent in the path of an overload report can view the

contents of that report. In addition to the requirement to select
which peers are trusted to send overload reports, operators MUST be
able to select which peers are authorized to receive reports. A node
MUST not send an overload report to a peer not authorized to receive
it. Furthermore, an agent MUST remove any overload reports that
might have been inserted by other nodes before forwarding a Diameter
message to a peer that is not authorized to receive overload reports.

At the time of this writing, the DIME working group is studying
requirements for adding end-to-end security
[I-D.ietf-dime-e2e-sec-req] features to Diameter. These features,
when they become available, might make it easier to establish trust
in non-adjacent nodes for overload control purposes. Readers should
be reminded, however, that the overload control mechanism encourages
Diameter agents to modify AVPs in, or insert additional AVPs into,
existing messages that are originated by other nodes. If end-to-end
security is enabled, there is a risk that such modification could
violate integrity protection. The details of using any future
Diameter end-to-end security mechanism with overload control will
require careful consideration, and are beyond the scope of this
document.

9. Contributors

The following people contributed substantial ideas, feedback, and
discussion to this document:

o Eric McMurry

o Hannes Tschofenig

o Ulrich Wiehe

o Jean-Jacques Trottin

o Maria Cruz Bartolome

o Martin Dolly

o Nirav Salot

o Susan Shishufeng

10. References

10.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 5226,
              May 2008.

   [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network
              Time Protocol Version 4: Protocol and Algorithms
              Specification", RFC 5905, June 2010.

   [RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn,
              "Diameter Base Protocol", RFC 6733, October 2012.

10.2. Informative References

   [3GPP.23.203]
              3GPP, "Policy and charging control architecture", 3GPP
              TS 23.203 10.9.0, September 2013.

   [3GPP.29.229]
              3GPP, "Cx and Dx interfaces based on the Diameter
              protocol; Protocol details", 3GPP TS 29.229 10.5.0,
              March 2013.

   [3GPP.29.272]
              3GPP, "Evolved Packet System (EPS); Mobility Management
              Entity (MME) and Serving GPRS Support Node (SGSN) related
              interfaces based on Diameter protocol", 3GPP TS 29.272
              10.8.0, June 2013.

   [I-D.ietf-dime-e2e-sec-req]
              Tschofenig, H., Korhonen, J., Zorn, G., and K. Pillay,
              "Diameter AVP Level Security: Scenarios and Requirements",
              draft-ietf-dime-e2e-sec-req-00 (work in progress),
              September 2013.

   [RFC4006] Hakala, H., Mattila, L., Koskinen, J-P., Stura, M., and J.
              Loughney, "Diameter Credit-Control Application", RFC 4006,
              August 2005.

   [RFC5729] Korhonen, J., Jones, M., Morand, L., and T. Tsou,
              "Clarifications on the Routing of Diameter Requests Based
              on the Username and the Realm", RFC 5729, December 2009.

   [RFC7068] McMurry, E. and B. Campbell, "Diameter Overload Control

Requirements", RFC 7068, November 2013.

Appendix A. Issues left for future specifications

   The base solution for the overload control does not cover all
   possible use cases. A number of solution aspects were intentionally
   left for future specification and protocol work.

A.1. Additional traffic abatement algorithms

   This specification describes only means for a simple loss based
   algorithm. Future algorithms can be added using the designed
   solution extension mechanism. The new algorithms need to be
   registered with IANA. See Sections 4.1 and 7 for the required IANA
   steps.

A.2. Agent Overload

   This specification focuses on Diameter end-point (server or client)
   overload. A separate extension will be required to outline the
   handling the case of agent overload.

A.3. DIAMETER_TOO_BUSY clarifications

   The current [RFC6733] behaviour in a case of DIAMETER_TOO_BUSY is
   somewhat under specified. For example, there is no information how
   long the specific Diameter node is willing to be unavailable. A
   specification updating [RFC6733] should clarify the handling of
   DIAMETER_TOO_BUSY from the error answer initiating Diameter node
   point of view and from the original request initiating Diameter node
   point of view. Further, the inclusion of possible additional
   information providing AVPs should be discussed and possible be
   recommended to be used.

Appendix B. Examples

B.1. Mix of Destination-Realm routed requests and Destination-Host
      routed requests

   Diameter allows a client to optionally select the destination server
   of a request, even if there are agents between the client and the
   server. The client does this using the Destination-Host AVP. In
   cases where the client does not care if a specific server receives
   the request, it can omit Destination-Host and route the request using
   the Destination-Realm and Application Id, effectively letting an
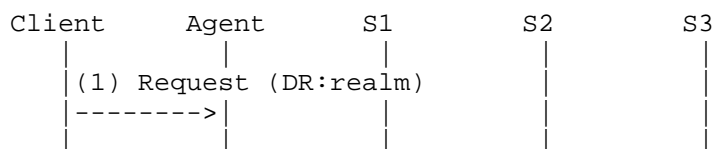   agent select the server.

Clients commonly send mixtures of Destination-Host and Destination-Realm routed requests. For example, in an application that uses user sessions, a client typically won't care which server handles a session-initiating requests. But once the session is initiated, the client will send all subsequent requests in that session to the same server. Therefore it would send the initial request with no Destination-Host AVP. If it receives a successful answer, the client would copy the Origin-Host value from the answer message into a Destination-Host AVP in each subsequent request in the session.

An agent has very limited options in applying overload abatement to requests that contain Destination-Host AVPs. It typically cannot route the request to a different server than the one identified in Destination-Host. It's only remaining options are to throttle such requests locally, or to send an overload report back towards the client so the client can throttle the requests. The second choice is usually more efficient, since it prevents any throttled requests from being sent in the first place, and removes the agent's need to send errors back to the client for each dropped request.

On the other hand, an agent has much more leeway to apply overload abatement for requests that do not contain Destination-Host AVPs. If the agent has multiple servers in its peer table for the given realm and application, it can route such requests to other, less overloaded servers.

If the overload severity increases, the agent may reach a point where there is not sufficient capacity across all servers to handle even realm-routed requests. In this case, the realm itself can be considered overloaded. The agent may need the client to throttle realm-routed requests in addition to Destination-Host routed requests. The overload severity may be different for each server, and the severity for the realm at is likely to be different than for any specific server. Therefore, an agent may need to forward, or originate, multiple overload reports with differing ReportType and Reduction-Percentage values.

Figure 8 illustrates such a mixed-routing scenario. In this example, the servers S1, S2, and S3 handle requests for the realm "realm". Any of the three can handle requests that are not part of a user session (i.e. routed by Destination-Realm). But once a session is established, all requests in that session must go to the same server.

```
   Client       Agent       S1          S2          S3
      |           |           |           |           |
      |(1) Request (DR:realm) |           |           |
      |---------->|           |           |           |
      |           |           |           |           |
```

```
|           |          |        |       |
|           |Agent selects S1    |       |
|           |          |        |       |
|           |          |        |       |
|           |          |        |       |
|           |(2) Request (DR:realm)      |
|           |-------->|         |       |
|           |          |        |       |
|           |          |        |       |
|           |          |S1 overloaded, returns OLR
|           |          |        |       |
|           |          |        |       |
|           |          |        |       |
|           |(3) Answer (OR:realm,OH:S1,OLR:RT=DH)
|           |<--------|         |       |
|           |          |        |       |
|           |          |        |       |
|           |sees OLR,routes DR traffic to S2&S3
|           |          |        |       |
|           |          |        |       |
|           |          |        |       |
|(4) Answer (OR:realm,OH:S1, OLR:RT=DH) |
|<--------|           |        |       |
|           |          |        |       |
|           |          |        |       |
|Client throttles requests with DH:S1   |
|           |          |        |       |
|           |          |        |       |
|           |          |        |       |
|(5) Request (DR:realm) |        |       |
|-------->|           |        |       |
|           |          |        |       |
|           |          |        |       |
|           |Agent selects S2    |       |
|           |          |        |       |
|           |          |        |       |
|           |(6) Request (DR:realm)      |
|           |----------------->|        |
|           |          |        |       |
|           |          |        |       |
|           |          |        |S2 is overloaded...
|           |          |        |       |
|           |          |        |       |
|           |          |        |       |
|           |(7) Answer (OH:S2, OLR:RT=DH)|
|           |<-----------------|        |
|           |          |        |       |
```

```
|               |             |             |             |
|               |Agent sees OLR, realm now overloaded
|               |             |             |             |
|               |             |             |             |
|               |             |             |             |
|(8) Answer (OR:realm,OH:S2, OLR:RT=DH, OLR: RT=R)
|<--------|               |             |             |
|               |             |             |             |
|               |             |             |             |
|Client throttles DH:S1, DH:S2, and DR:realm
|               |             |             |             |
|               |             |             |             |
|               |             |             |             |
|               |             |             |             |
|               |             |             |             |
```
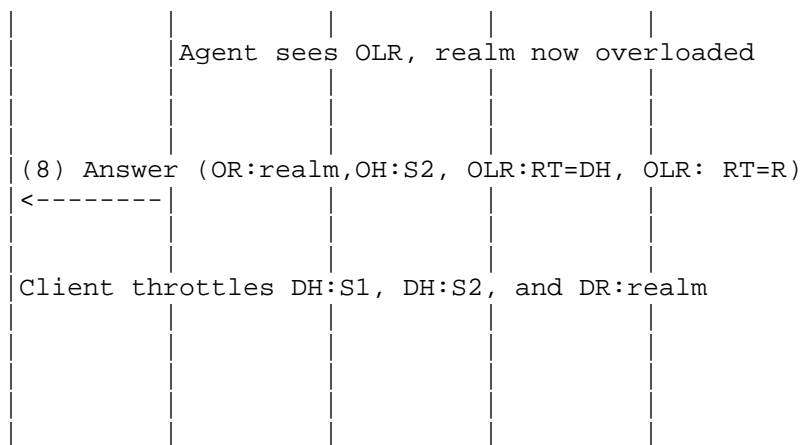
Figure 8: Mix of Destination-Host and Destination-Realm Routed
Requests

1. The client sends a request with no Destination-Host AVP (that is,
    a Destination-Realm routed request.)

2. The agent follows local policy to select a server from its peer
    table. In this case, the agent selects S2 and forwards the
    request.

3. S1 is overloaded. It sends a answer indicating success, but also
    includes an overload report. Since the overload report only
    applies to S1, the ReportType is "Destination-Host".

4. The agent sees the overload report, and records that S1 is
    overloaded by the value in the Reduction-Percentage AVP. It
    begins diverting the indicated percentage of realm-routed traffic
    from S1 to S2 and S3. Since it can't divert Destination-Host
    routed traffic, it forwards the overload report to the client.
    This effectively delegates the throttling of traffic with
    Destination-Host:S1 to the client.

5. The client sends another Destination-Realm routed request.

6. The agent selects S2, and forwards the request.

7. It turns out that S2 is also overloaded, perhaps due to all that
    traffic it took over for S1. S2 returns an successful answer
    containing an overload report. Since this report only applies to
    S2, the ReportType is "Destination-Host".

8. The agent sees that S2 is also overloaded by the value in
    Reduction-Percentage. This value is probably different than the
    value from S1's report. The agent diverts the remaining traffic
    to S3 as best as it can, but it calculates that the remaining
    capacity across all three servers is no longer sufficient to
    handle all of the realm-routed traffic. This means the realm
    itself is overloaded. The realm's overload percentage is most
    likely different than that for either S1 or S2. The agent
    forward's S2's report back to the client in the Diameter answer.
    Additionally, the agent generates a new report for the realm of
    "realm", and inserts that report into the answer. The client
    throttles requests with Destination-Host:S1 at one rate, requests
    with Destination-Host:S2 at another rate, and requests with no
    Destination-Host AVP at yet a third rate. (Since S3 has not
    indicated overload, the client does not throttle requests with
    Destination-Host:S3.)

Authors' Addresses

   Jouni Korhonen (editor)
   Broadcom
   Porkkalankatu 24
   Helsinki FIN-00180
   Finland

   Email: jouni.nospam@gmail.com

   Steve Donovan
   Oracle
   17210 Campbell Road
   Dallas, Texas 75254
   United States

   Email: srdonovan@usdonovans.com

   Ben Campbell
   Oracle
   17210 Campbell Road
   Dallas, Texas 75254
   United States

   Email: ben@nostrum.com

      Lionel Morand
      Orange Labs
      38/40 rue du General Leclerc
      Issy-Les-Moulineaux Cedex 9 92794
      France

      Phone: +33145296257
      Email: lionel.morand@orange.com