                   Diameter Overload Indication Conveyance

Abstract

   This specification defines a base solution for Diameter overload
   control, referred to as Diameter Overload Indication Conveyance
   (DOIC).

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7683.

Table of Contents

1. Introduction

   This specification defines a base solution for Diameter overload
   control, referred to as Diameter Overload Indication Conveyance
   (DOIC), based on the requirements identified in [RFC7068].

   This specification addresses Diameter overload control between
   Diameter nodes that support the DOIC solution. The solution, which
   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.

   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. It is the responsibility of the Diameter application
   designers to define how overload control mechanisms work on that
   application.

   Note that the overload control solution defined in this specification
   does not address all the requirements listed in [RFC7068]. A number
   of features related to overload control are left for future
   specifications. See Appendix A for a list of extensions that are
   currently being considered.

2. Terminology and Abbreviations

   Abatement

      Reaction to receipt of an overload report resulting in a reduction
      in traffic sent to the reporting node. Abatement actions include
      diversion and throttling.

   Abatement Algorithm

      An extensible method requested by reporting nodes and used by
      reacting nodes to reduce the amount of traffic sent during an
      occurrence of overload control.

   Diversion

      An overload abatement treatment where the reacting node selects
      alternate destinations or paths for requests.

   Host-Routed Requests

      Requests that a reacting node knows will be served by a particular
      host, either due to the presence of a Destination-Host Attribute
      Value Pair (AVP) or by some other local knowledge on the part of
      the reacting node.

   Overload Control State (OCS)

      Internal state maintained by a reporting or reacting node
      describing occurrences of overload control.

   Overload Report (OLR)

      Overload control information for a particular overload occurrence
      sent by a reporting node.

   Reacting Node

      A Diameter node that acts upon an overload report.

   Realm-Routed Requests

      Requests sent by a reacting node where the reacting node does not
      know to which host the request will be routed.

   Reporting Node

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

   Throttling

      An abatement treatment that limits the number of requests sent by
      the reacting node. Throttling can include a Diameter Client
      choosing to not send requests, or a Diameter Agent or Server
      rejecting requests with appropriate error responses. In both
      cases, the result of the throttling is a permanent rejection of
      the transaction.

3. Conventions Used in This Document

   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].

   The interpretation from RFC 2119 [RFC2119] does not apply for the
   above listed words when they are not used in all caps.

4. Solution Overview

   The Diameter Overload Information Conveyance (DOIC) solution allows
   Diameter nodes to request that other Diameter nodes perform overload
   abatement actions, that is, actions to reduce the load offered to the
   overloaded node or realm.

   A Diameter node that supports DOIC is known as a "DOIC node". Any
   Diameter node can act as a DOIC node, including Diameter Clients,
   Diameter Servers, and Diameter Agents. DOIC nodes are further
   divided into "Reporting Nodes" and "Reacting Nodes." A reporting
   node requests overload abatement by sending Overload Reports (OLRs).

   A reacting node acts upon OLRs and performs whatever actions are
   needed to fulfill the abatement requests included in the OLRs. A
   reporting node may report overload on its own behalf or on behalf of
   other nodes. Likewise, a reacting node may perform overload
   abatement on its own behalf or on behalf of other nodes.

   A Diameter node's role as a DOIC node is independent of its Diameter
   role. For example, Diameter Agents may act as DOIC nodes, even
   though they are not endpoints in the Diameter sense. Since Diameter
   enables bidirectional applications, where Diameter Servers can send
   requests towards Diameter Clients, a given Diameter node can
   simultaneously act as both a reporting node and a reacting node.

   Likewise, a Diameter Agent may act as a reacting node from the
   perspective of upstream nodes, and a reporting node from the
   perspective of downstream nodes.

DOIC nodes do not generate new messages to carry DOIC-related information. Rather, they "piggyback" DOIC information over existing Diameter messages by inserting new AVPs into existing Diameter requests and responses. Nodes indicate support for DOIC, and any needed DOIC parameters, by inserting an OC-Supported-Features AVP (Section 7.1) into existing requests and responses. Reporting nodes send OLRs by inserting OC-OLR AVPs (Section 7.3).

A given OLR applies to the Diameter realm and application of the Diameter message that carries it. If a reporting node supports more than one realm and/or application, it reports independently for each combination of realm and application. Similarly, the OC-Supported-Features AVP applies to the realm and application of the enclosing message. This implies that a node may support DOIC for one application and/or realm, but not another, and may indicate different DOIC parameters for each application and realm for which it supports DOIC.

Reacting nodes perform overload abatement according to an agreed-upon abatement algorithm. An abatement algorithm defines the meaning of some of the parameters of an OLR and the procedures required for overload abatement. An overload abatement algorithm separates Diameter requests into two sets. The first set contains the requests that are to undergo overload abatement treatment of either throttling or diversion. The second set contains the requests that are to be given normal routing treatment. This document specifies a single "must-support" algorithm, namely, the "loss" algorithm (Section 6). Future specifications may introduce new algorithms.

Overload conditions may vary in scope. For example, a single Diameter node may be overloaded, in which case, reacting nodes may attempt to send requests to other destinations. On the other hand, an entire Diameter realm may be overloaded, in which case, such attempts would do harm. DOIC OLRs have a concept of "report type" (Section 7.6), where the type defines such behaviors. Report types are extensible. This document defines report types for overload of a specific host and for overload of an entire realm.

DOIC works through non-supporting Diameter Agents that properly pass unknown AVPs unchanged.

4.1. Piggybacking

There is no new Diameter application defined to carry overload-related AVPs. The overload control AVPs defined in this specification have been designed to be piggybacked on top of existing

application messages. This is made possible by adding the optional
overload control AVPs OC-OLR and OC-Supported-Features into existing
commands.

Reacting nodes indicate support for DOIC by including the
OC-Supported-Features AVP in all request messages originated or
relayed by the reacting node.

Reporting nodes indicate support for DOIC by including the
OC-Supported-Features AVP in all answer messages that are originated
or relayed by the reporting node and that are in response to a
request that contained the OC-Supported-Features AVP. Reporting
nodes may include overload reports using the OC-OLR AVP in answer
messages.

Note that the overload control solution does not have fixed server
and client roles. The DOIC node 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., sent by a "reporting node").
Therefore, in a typical client-server deployment, the Diameter Client
may report its overload condition to the Diameter Server for any
Diameter-Server-initiated message exchange. An example of such is
the Diameter Server requesting a re-authentication from a Diameter
Client.

4.2. DOIC Capability Announcement

The DOIC solution supports the ability for Diameter nodes to
determine if other nodes in the path of a request support the
solution. This capability is referred to as DOIC Capability
Announcement (DCA) and is separate from the Diameter Capability
Exchange.

The DCA mechanism uses the OC-Supported-Features AVPs to indicate the
Diameter overload features supported.

The first node in the path of a Diameter request that supports the
DOIC solution inserts the OC-Supported-Features AVP in the request
message.

The individual features supported by the DOIC nodes are indicated in
the OC-Feature-Vector AVP. Any semantics associated with the
features will be defined in extension specifications that introduce
the features.

   Note: As discussed elsewhere in the document, agents in the path
   of the request can modify the OC-Supported-Features AVP.

Note: The DOIC solution must support deployments where Diameter
Clients and/or Diameter Servers do not support the DOIC solution.
In this scenario, Diameter Agents that support the DOIC solution
may handle overload abatement for the non-supporting Diameter
nodes. In this case, the DOIC agent will insert the OC-Supported-
Features AVP in requests that do not already contain one, telling
the reporting node that there is a DOIC node that will handle
overload abatement. For transactions where there was an
OC-Supporting-Features AVP in the request, the agent will insert
the OC-Supported-Features AVP in answers, telling the reacting
node that there is a reporting node.

The OC-Feature-Vector AVP will always contain an indication of
support for the loss overload abatement algorithm defined in this
specification (see Section 6). This ensures that a reporting node
always supports at least one of the advertised abatement algorithms
received in a request messages.

The reporting node inserts the OC-Supported-Features AVP in all
answer messages to requests that contained the OC-Supported-Features
AVP. The contents of the reporting node's OC-Supported-Features AVP
indicate the set of Diameter overload features supported by the
reporting node. This specification defines one exception -- the
reporting node only includes an indication of support for one
overload abatement algorithm, independent of the number of overload
abatement algorithms actually supported by the reacting node. The
overload abatement algorithm indicated is the algorithm that the
reporting node intends to use should it enter an overload condition.
Reacting nodes can use the indicated overload abatement algorithm to
prepare for possible overload reports and must use the indicated
overload abatement algorithm if traffic reduction is actually
requested.

Note that the loss algorithm defined in this document is a
stateless abatement algorithm. As a result, it does not require
any actions by reacting nodes prior to the receipt of an overload
report. Stateful abatement algorithms that base the abatement
logic on a history of request messages sent might require reacting
nodes to maintain state in advance of receiving an overload report
to ensure that the overload reports can be properly handled.

While it should only be done in exceptional circumstances and not
during an active occurrence of overload, a reacting node that wishes
to transition to a different abatement algorithm can stop advertising
support for the algorithm indicated by the reporting node, as long as
support for the loss algorithm is always advertised.

The DCA mechanism must also allow the scenario where the set of
features supported by the sender of a request and by agents in the
path of a request differ. In this case, the agent can update the
OC-Supported-Features AVP to reflect the mixture of the two sets of
supported features.

   Note: The logic to determine if the content of the OC-Supported-
   Features AVP should be changed is out of scope for this document,
   as is the logic to determine the content of a modified
   OC-Supported-Features AVP. These are left to implementation
   decisions. Care must be taken not to introduce interoperability
   issues for downstream or upstream DOIC nodes. As such, the agent
   must act as a fully compliant reporting node to the downstream
   reacting node and as a fully compliant reacting node to the
   upstream reporting node.

4.3. DOIC Overload Condition Reporting

   As with DOIC capability announcement, overload condition reporting
   uses new AVPs (Section 7.3) to indicate an overload condition.

   The OC-OLR AVP is referred to as an overload report. The OC-OLR AVP
   includes the type of report, a sequence number, the length of time
   that the report is valid, and AVPs specific to the abatement
   algorithm.

   Two types of overload reports are defined in this document: host
   reports and realm reports.

   A report of type "HOST_REPORT" is sent to indicate the overload of a
   specific host, identified by the Origin-Host AVP of the message
   containing the OLR, for the Application-ID indicated in the
   transaction. When receiving an OLR of type "HOST_REPORT", a reacting
   node applies overload abatement treatment to the host-routed requests
   identified by the overload abatement algorithm (as defined in
   Section 2) sent for this application to the overloaded host.

   A report of type "REALM_REPORT" is sent to indicate the overload of a
   realm for the Application-ID indicated in the transaction. The
   overloaded realm is identified by the Destination-Realm AVP of the
   message containing the OLR. When receiving an OLR of type
   "REALM_REPORT", a reacting node applies overload abatement treatment
   to realm-routed requests identified by the overload abatement
   algorithm (as defined in Section 2) sent for this application to the
   overloaded realm.

This document assumes that there is a single source for realm reports
for a given realm, or that if multiple nodes can send realm reports,
that each such node has full knowledge of the overload state of the
entire realm. A reacting node cannot distinguish between receiving
realm reports from a single node or from multiple nodes.

> Note: Known issues exist if there are multiple sources for
> overload reports that apply to the same Diameter entity. Reacting
> nodes have no way of determining the source and, as such, will
> treat them as coming from a single source. Variance in sequence
> numbers between the two sources can then cause incorrect overload
> abatement treatment to be applied for indeterminate periods of
> time.

Reporting nodes are responsible for determining the need for a
reduction of traffic. The method for making this determination is
implementation specific and depends on the type of overload report
being generated. A host report might be generated by tracking use of
resources required by the host to handle transactions for the
Diameter application. A realm report generally impacts the traffic
sent to multiple hosts and, as such, requires tracking the capacity
of all servers able to handle realm-routed requests for the
application and realm.

Once a reporting node determines the need for a reduction in traffic,
it uses the DOIC-defined AVPs to report on the condition. These AVPs
are included in answer messages sent or relayed by the reporting
node. The reporting node indicates the overload abatement algorithm
that is to be used to handle the traffic reduction in the
OC-Supported-Features AVP. The OC-OLR AVP is used to communicate
information about the requested reduction.

Reacting nodes, upon receipt of an overload report, apply the
overload abatement algorithm to traffic impacted by the overload
report. The method used to determine the requests that are to
receive overload abatement treatment is dependent on the abatement
algorithm. The loss abatement algorithm is defined in this document
(Section 6). Other abatement algorithms can be defined in extensions
to the DOIC solution.

Two types of overload abatement treatment are defined, diversion and
throttling. Reacting nodes are responsible for determining which
treatment is appropriate for individual requests.

As the conditions that lead to the generation of the overload report
change, the reporting node can send new overload reports requesting
greater reduction if the condition gets worse or less reduction if
the condition improves. The reporting node sends an overload report

with a duration of zero to indicate that the overload condition has
ended and abatement is no longer needed.

The reacting node also determines when the overload report expires
based on the OC-Validity-Duration AVP in the overload report and
stops applying the abatement algorithm when the report expires.

Note that erroneous overload reports can be used for DoS attacks.
This includes the ability to indicate that a significant reduction in
traffic, up to and including a request for no traffic, should be sent
to a reporting node. As such, care should be taken to verify the
sender of overload reports.

4.4. DOIC Extensibility

The DOIC solution is designed to be extensible. This extensibility
is based on existing Diameter-based extensibility mechanisms, along
with the DOIC capability announcement mechanism.

There are multiple categories of extensions that are expected. This
includes the definition of new overload abatement algorithms, the
definition of new report types, and the definition of new scopes of
messages impacted by an overload report.

A DOIC node communicates supported features by including them in the
OC-Feature-Vector AVP, as a sub-AVP of OC-Supported-Features. Any
non-backwards-compatible DOIC extensions define new values for the
OC-Feature-Vector AVP. DOIC extensions also have the ability to add
new AVPs to the OC-Supported-Features AVP, if additional information
about the new feature is required.

Overload reports can also be extended by adding new sub-AVPs to the
OC-OLR AVP, allowing reporting nodes to communicate additional
information about handling an overload condition.

If necessary, new extensions can also define new AVPs that are not
part of the OC-Supported-Features and OC-OLR group AVPs. It is,
however, recommended that DOIC extensions use the OC-Supported-
Features AVP and OC-OLR AVP to carry all DOIC-related AVPs.

4.5. Simplified Example Architecture

   Figure 1 illustrates the simplified architecture for Diameter
   overload information conveyance.

```
    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)   <--------------------> <---------------------->
               Diameter Application Y   Diameter Application Y
```
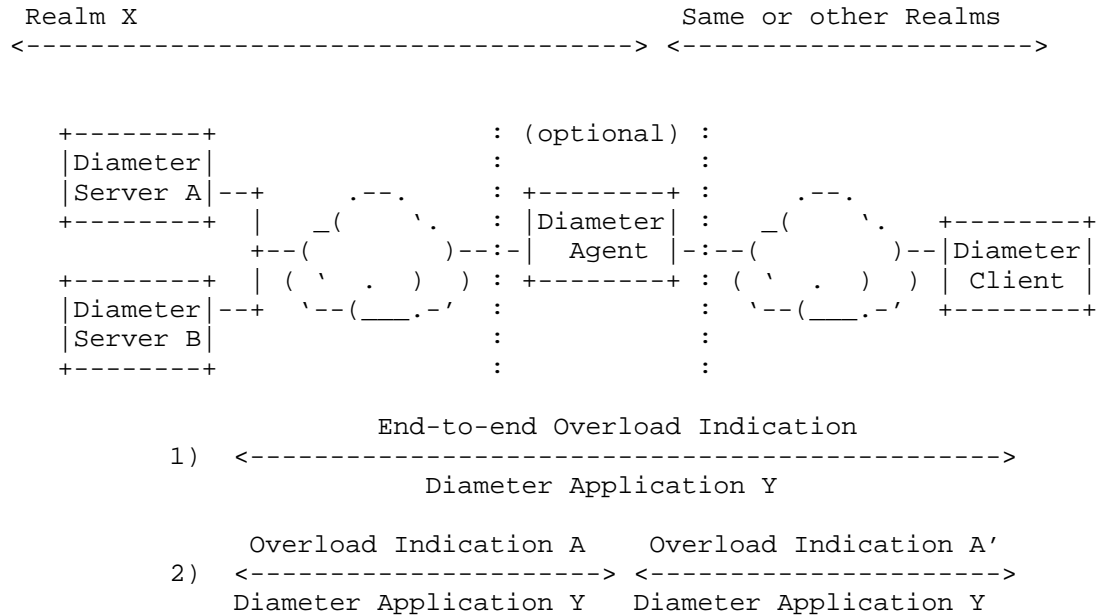
          Figure 1: Simplified Architecture Choices for Overload Indication
                                 Delivery

   In Figure 1, the Diameter overload indication can be conveyed (1)
   end-to-end between servers and clients or (2) between servers and the
   Diameter Agent inside the realm and then between the Diameter Agent
   and the clients.

5. Solution Procedures

   This section outlines the normative behavior for the DOIC solution.

5.1. Capability Announcement

   This section defines DOIC Capability Announcement (DCA) behavior.

      Note: This specification assumes that changes in DOIC node
      capabilities are relatively rare events that occur as a result of
      administrative action. Reacting nodes ought to minimize changes
      that force the reporting node to change the features being used,
      especially during active overload conditions. But even if

      reacting nodes avoid such changes, reporting nodes still have to
      be prepared for them to occur. For example, differing
      capabilities between multiple reacting nodes may still force a
      reporting node to select different features on a per-transaction
      basis.

5.1.1. Reacting Node Behavior

   A reacting node MUST include the OC-Supported-Features AVP in all
   requests. It MAY include the OC-Feature-Vector AVP, as a sub-AVP of
   OC-Supported-Features. If it does so, it MUST indicate support for
   the "loss" algorithm. If the reacting node is configured to support
   features (including other algorithms) in addition to the loss
   algorithm, it MUST indicate such support in an OC-Feature-Vector AVP.

   An OC-Supported-Features AVP in answer messages indicates there is a
   reporting node for the transaction. The reacting node MAY take
   action, for example, creating state for some stateful abatement
   algorithm, based on the features indicated in the OC-Feature-Vector
   AVP.

      Note: The loss abatement algorithm does not require stateful
      behavior when there is no active overload report.

   Reacting nodes need to be prepared for the reporting node to change
   selected algorithms. This can happen at any time, including when the
   reporting node has sent an active overload report. The reacting node
   can minimize the potential for changes by modifying the advertised
   abatement algorithms sent to an overloaded reporting node to the
   currently selected algorithm and loss (or just loss if it is the
   currently selected algorithm). This has the effect of limiting the
   potential change in abatement algorithm from the currently selected
   algorithm to loss, avoiding changes to more complex abatement
   algorithms that require state to operate properly.

5.1.2. Reporting Node Behavior

   Upon receipt of a request message, a reporting node determines if
   there is a reacting node for the transaction based on the presence of
   the OC-Supported-Features AVP in the request message.

   If the request message contains an OC-Supported-Features AVP, then a
   reporting node MUST include the OC-Supported-Features AVP in the
   answer message for that transaction.

      Note: Capability announcement is done on a per-transaction basis.
      The reporting node cannot assume that the capabilities announced
      by a reacting node will be the same between transactions.

A reporting node MUST NOT include the OC-Supported-Features AVP,
OC-OLR AVP, or any other overload control AVPs defined in extension
documents in response messages for transactions where the request
message does not include the OC-Supported-Features AVP. Lack of the
OC-Supported-Features AVP in the request message indicates that there
is no reacting node for the transaction.

A reporting node knows what overload control functionality is
supported by the reacting node based on the content or absence of the
OC-Feature-Vector AVP within the OC-Supported-Features AVP in the
request message.

A reporting node MUST select a single abatement algorithm in the
OC-Feature-Vector AVP. The abatement algorithm selected MUST
indicate the abatement algorithm the reporting node wants the
reacting node to use when the reporting node enters an overload
condition.

The abatement algorithm selected MUST be from the set of abatement
algorithms contained in the request message's OC-Feature-Vector AVP.

A reporting node that selects the loss algorithm may do so by
including the OC-Feature-Vector AVP with an explicit indication of
the loss algorithm, or it MAY omit the OC-Feature-Vector AVP. If it
selects a different algorithm, it MUST include the OC-Feature-Vector
AVP with an explicit indication of the selected algorithm.

The reporting node SHOULD indicate support for other DOIC features
defined in extension documents that it supports and that apply to the
transaction. It does so using the OC-Feature-Vector AVP.

   Note: Not all DOIC features will apply to all Diameter
   applications or deployment scenarios. The features included in
   the OC-Feature-Vector AVP are based on local policy of the
   reporting node.

5.1.3. Agent Behavior

   Diameter Agents that support DOIC can ensure that all messages
   relayed by the agent contain the OC-Supported-Features AVP.

   A Diameter Agent MAY take on reacting node behavior for Diameter
   endpoints that do not support the DOIC solution. A Diameter Agent
   detects that a Diameter endpoint does not support DOIC reacting node
   behavior when there is no OC-Supported-Features AVP in a request
   message.

For a Diameter Agent to be a reacting node for a non-supporting
Diameter endpoint, the Diameter Agent MUST include the OC-Supported-
Features AVP in request messages it relays that do not contain the
OC-Supported-Features AVP.

A Diameter Agent MAY take on reporting node behavior for Diameter
endpoints that do not support the DOIC solution. The Diameter Agent
MUST have visibility to all traffic destined for the non-supporting
host in order to become the reporting node for the Diameter endpoint.
A Diameter Agent detects that a Diameter endpoint does not support
DOIC reporting node behavior when there is no OC-Supported-Features
AVP in an answer message for a transaction that contained the
OC-Supported-Features AVP in the request message.

If a request already has the OC-Supported-Features AVP, a Diameter
Agent MAY modify it to reflect the features appropriate for the
transaction. Otherwise, the agent relays the OC-Supported-Features
AVP without change.

   Example: If the agent supports a superset of the features reported
   by the reacting node, then the agent might choose, based on local
   policy, to advertise that superset of features to the reporting
   node.

If the Diameter Agent changes the OC-Supported-Features AVP in a
request message, then it is likely it will also need to modify the
OC-Supported-Features AVP in the answer message for the transaction.
A Diameter Agent MAY modify the OC-Supported-Features AVP carried in
answer messages.

When making changes to the OC-Supported-Features or OC-OLR AVPs, the
Diameter Agent needs to ensure consistency in its behavior with both
upstream and downstream DOIC nodes.

5.2. Overload Report Processing

5.2.1. Overload Control State

Both reacting and reporting nodes maintain Overload Control State
(OCS) for active overload conditions. The following sections define
behavior associated with that OCS.

The contents of the OCS in the reporting node and in the reacting
node represent logical constructs. The actual internal physical
structure of the state included in the OCS is an implementation
decision.

5.2.1.1. Overload Control State for Reacting Nodes

   A reacting node maintains the following OCS per supported Diameter
   application:

   o a host-type OCS entry for each Destination-Host to which it sends
      host-type requests and

   o a realm-type OCS entry for each Destination-Realm to which it
      sends realm-type requests.

   A host-type OCS entry is identified by the pair of Application-ID and
   the node's DiameterIdentity.

   A realm-type OCS entry is identified by the pair of Application-ID
   and realm.

   The host-type and realm-type OCS entries include the following
   information (the actual information stored is an implementation
   decision):

   o Sequence number (as received in OC-OLR; see Section 7.3)

   o Time of expiry (derived from OC-Validity-Duration AVP received in
      the OC-OLR AVP and time of reception of the message carrying
      OC-OLR AVP)

   o Selected abatement algorithm (as received in the OC-Supported-
      Features AVP)

   o Input data that is abatement algorithm specific (as received in
      the OC-OLR AVP -- for example, OC-Reduction-Percentage for the
      loss abatement algorithm)

5.2.1.2. Overload Control State for Reporting Nodes

   A reporting node maintains OCS entries per supported Diameter
   application, per supported (and eventually selected) abatement
   algorithm, and per report type.

   An OCS entry is identified by the tuple of Application-ID, report
   type, and abatement algorithm, and it includes the following
   information (the actual information stored is an implementation
   decision):

   o Sequence number

   o Validity duration

   o Expiration time

   o Input data that is algorithm specific (for example, the reduction
      percentage for the loss abatement algorithm)

5.2.1.3. Reacting Node's Maintenance of Overload Control State

   When a reacting node receives an OC-OLR AVP, it MUST determine if it
   is for an existing or new overload condition.

      Note: For the remainder of this section, the term "OLR" refers to
      the combination of the contents of the received OC-OLR AVP and the
      abatement algorithm indicated in the received OC-Supported-
      Features AVP.

   When receiving an answer message with multiple OLRs of different
   supported report types, a reacting node MUST process each received
   OLR.

   The OLR is for an existing overload condition if a reacting node has
   an OCS that matches the received OLR.

   For a host report, this means it matches the Application-ID and the
   host's DiameterIdentity in an existing host OCS entry.

   For a realm report, this means it matches the Application-ID and the
   realm in an existing realm OCS entry.

   If the OLR is for an existing overload condition, then a reacting
   node MUST determine if the OLR is a retransmission or an update to
   the existing OLR.

   If the sequence number for the received OLR is greater than the
   sequence number stored in the matching OCS entry, then a reacting
   node MUST update the matching OCS entry.

   If the sequence number for the received OLR is less than or equal to
   the sequence number in the matching OCS entry, then a reacting node
   MUST silently ignore the received OLR. The matching OCS MUST NOT be
   updated in this case.

   If the reacting node determines that the sequence number has rolled
   over, then the reacting node MUST update the matching OCS entry.
   This can be determined by recognizing that the number has changed
   from a value within 1% of the maximum value in the OC-Sequence-Number
   AVP to a value within 1% of the minimum value in the OC-Sequence-
   Number AVP.

If the received OLR is for a new overload condition, then a reacting node MUST generate a new OCS entry for the overload condition.

For a host report, this means a reacting node creates an OCS entry with the Application-ID in the received message and DiameterIdentity of the Origin-Host in the received message.

   Note: This solution assumes that the Origin-Host AVP in the answer message included by the reporting node is not changed along the path to the reacting node.

For a realm report, this means a reacting node creates an OCS entry with the Application-ID in the received message and realm of the Origin-Realm in the received message.

If the received OLR contains a validity duration of zero ("0"), then a reacting node MUST update the OCS entry as being expired.

   Note: It is not necessarily appropriate to delete the OCS entry, as the recommended behavior is that the reacting node slowly returns to full traffic when ending an overload abatement period.

The reacting node does not delete an OCS when receiving an answer message that does not contain an OC-OLR AVP (i.e., absence of OLR means "no change").

5.2.1.4. Reporting Node's Maintenance of Overload Control State

A reporting node SHOULD create a new OCS entry when entering an overload condition.

   Note: If a reporting node knows through absence of the OC-Supported-Features AVP in received messages that there are no reacting nodes supporting DOIC, then the reporting node can choose to not create OCS entries.

When generating a new OCS entry, the sequence number SHOULD be set to zero ("0").

When generating sequence numbers for new overload conditions, the new sequence number MUST be greater than any sequence number in an active (unexpired) overload report for the same application and report type previously sent by the reporting node. This property MUST hold over a reboot of the reporting node.

   Note: One way of addressing this over a reboot of a reporting node
   is to use a timestamp for the first overload condition that occurs
   after the report and to start using sequences beginning with zero
   for subsequent overload conditions.

A reporting node MUST update an OCS entry when it needs to adjust the
validity duration of the overload condition at reacting nodes.

   Example: If a reporting node wishes to instruct reacting nodes to
   continue overload abatement for a longer period of time than
   originally communicated. This also applies if the reporting node
   wishes to shorten the period of time that overload abatement is to
   continue.

A reporting node MUST update an OCS entry when it wishes to adjust
any parameters specific to the abatement algorithm, including, for
example, the reduction percentage used for the loss abatement
algorithm.

   Example: If a reporting node wishes to change the reduction
   percentage either higher (if the overload condition has worsened)
   or lower (if the overload condition has improved), then the
   reporting node would update the appropriate OCS entry.

A reporting node MUST increment the sequence number associated with
the OCS entry anytime the contents of the OCS entry are changed.
This will result in a new sequence number being sent to reacting
nodes, instructing them to process the OC-OLR AVP.

A reporting node SHOULD update an OCS entry with a validity duration
of zero ("0") when the overload condition ends.

   Note: If a reporting node knows that the OCS entries in the
   reacting nodes are near expiration, then the reporting node might
   decide not to send an OLR with a validity duration of zero.

A reporting node MUST keep an OCS entry with a validity duration of
zero ("0") for a period of time long enough to ensure that any
unexpired reacting node's OCS entry created as a result of the
overload condition in the reporting node is deleted.

5.2.2. Reacting Node Behavior

   When a reacting node sends a request, it MUST determine if that
   request matches an active OCS.

If the request matches an active OCS, then the reacting node MUST use the overload abatement algorithm indicated in the OCS to determine if the request is to receive overload abatement treatment.

For the loss abatement algorithm defined in this specification, see Section 6 for the overload abatement algorithm logic applied.

If the overload abatement algorithm selects the request for overload abatement treatment, then the reacting node MUST apply overload abatement treatment on the request. The abatement treatment applied depends on the context of the request.

If diversion abatement treatment is possible (i.e., a different path for the request can be selected where the overloaded node is not part of the different path), then the reacting node SHOULD apply diversion abatement treatment to the request. The reacting node MUST apply throttling abatement treatment to requests identified for abatement treatment when diversion treatment is not possible or was not applied.

> Note: This only addresses the case where there are two defined abatement treatments, diversion and throttling. Any extension that defines a new abatement treatment must also define its interaction with existing treatments.

If the overload abatement treatment results in throttling of the request and if the reacting node is an agent, then the agent MUST send an appropriate error as defined in Section 8.

Diameter endpoints that throttle requests need to do so according to the rules of the client application. Those rules will vary by application and are beyond the scope of this document.

In the case that the OCS entry indicated no traffic was to be sent to the overloaded entity and the validity duration expires, then overload abatement associated with the overload report MUST be ended in a controlled fashion.

5.2.3. Reporting Node Behavior

If there is an active OCS entry, then a reporting node SHOULD include the OC-OLR AVP in all answers to requests that contain the OC-Supported-Features AVP and that match the active OCS entry.

> Note: A request matches 1) if the Application-ID in the request matches the Application-ID in any active OCS entry and 2) if the report type in the OCS entry matches a report type supported by the reporting node as indicated in the OC-Supported-Features AVP.

The contents of the OC-OLR AVP depend on the selected algorithm.

A reporting node MAY choose to not resend an overload report to a
reacting node if it can guarantee that this overload report is
already active in the reacting node.

> Note: In some cases (e.g., when there are one or more agents in
> the path between reporting and reacting nodes, or when overload
> reports are discarded by reacting nodes), a reporting node may not
> be able to guarantee that the reacting node has received the
> report.

A reporting node MUST NOT send overload reports of a type that has
not been advertised as supported by the reacting node.

> Note: A reacting node implicitly advertises support for the host
> and realm report types by including the OC-Supported-Features AVP
> in the request. Support for other report types will be explicitly
> indicated by new feature bits in the OC-Feature-Vector AVP.

A reporting node SHOULD explicitly indicate the end of an overload
occurrence by sending a new OLR with OC-Validity-Duration set to a
value of zero ("0"). The reporting node SHOULD ensure that all
reacting nodes receive the updated overload report.

A reporting node MAY rely on the OC-Validity-Duration AVP values for
the implicit cleanup of overload control state on the reacting node.

> Note: All OLRs sent have an expiration time calculated by adding
> the validity duration contained in the OLR to the time the message
> was sent. Transit time for the OLR can be safely ignored. The
> reporting node can ensure that all reacting nodes have received
> the OLR by continuing to send it in answer messages until the
> expiration time for all OLRs sent for that overload condition have
> expired.

When a reporting node sends an OLR, it effectively delegates any
necessary throttling to downstream nodes. If the reporting node also
locally throttles the same set of messages, the overall number of
throttled requests may be higher than intended. Therefore, before
applying local message throttling, a reporting node needs to check if
these messages match existing OCS entries, indicating that these
messages have survived throttling applied by downstream nodes that
have received the related OLR.

However, even if the set of messages match existing OCS entries, the
reporting node can still apply other abatement methods such as
diversion. The reporting node might also need to throttle requests

for reasons other than overload. For example, an agent or server
might have a configured rate limit for each client and might throttle
requests that exceed that limit, even if such requests had already
been candidates for throttling by downstream nodes. The reporting
node also has the option to send new OLRs requesting greater
reductions in traffic, reducing the need for local throttling.

A reporting node SHOULD decrease requested overload abatement
treatment in a controlled fashion to avoid oscillations in traffic.

   Example: A reporting node might wait some period of time after
   overload ends before terminating the OLR, or it might send a
   series of OLRs indicating progressively less overload severity.

5.3. Protocol Extensibility

The DOIC solution can be extended. Types of potential extensions
include new traffic abatement algorithms, new report types, or other
new functionality.

When defining a new extension that requires new normative behavior,
the specification must define a new feature for the OC-Feature-Vector
AVP. This feature bit is used to communicate support for the new
feature.

The extension may define new AVPs for use in the DOIC Capability
Announcement and for use in DOIC overload reporting. These new AVPs
SHOULD be defined to be extensions to the OC-Supported-Features or
OC-OLR AVPs defined in this document.

The Grouped AVP extension mechanisms defined in [RFC6733] apply.
This allows, for example, defining a new feature that is mandatory to
be understood even when piggybacked on an existing application.

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 OC-Supported-Feature and OC-OLR AVPs can be expanded with
optional sub-AVPs only if a legacy DOIC implementation can safely
ignore them without breaking backward compatibility for the given
OC-Report-Type AVP value. Any new sub-AVPs must not require that the
M-bit be set.

Documents that introduce new report types must describe any
limitations on their use across non-supporting agents.

As with any Diameter specification, RFC 6733 requires all new AVPs to
be registered with IANA. See Section 9 for the required procedures.
New features (feature bits in the OC-Feature-Vector AVP) and report
types (in the OC-Report-Type AVP) MUST be registered with IANA.

6. Loss Algorithm

   This section documents the Diameter overload loss abatement
   algorithm.

6.1. Overview

   The DOIC specification supports the ability for multiple overload
   abatement algorithms to be specified. The abatement algorithm used
   for any instance of overload is determined by the DOIC Capability
   Announcement process documented in Section 5.1.

   The loss algorithm described in this section is the default algorithm
   that must be supported by all Diameter nodes that support DOIC.

   The loss algorithm is designed to be a straightforward and stateless
   overload abatement algorithm. It is used by reporting nodes to
   request a percentage reduction in the amount of traffic sent. The
   traffic impacted by the requested reduction depends on the type of
   overload report.

   Reporting nodes request the stateless reduction of the number of
   requests by an indicated percentage. This percentage reduction is in
   comparison to the number of messages the node otherwise would send,
   regardless of how many requests the node might have sent in the past.

   From a conceptual level, the logic at the reacting node could be
   outlined as follows.

   1. An overload report is received, and the associated OCS is either
       saved or updated (if required) by the reacting node.

   2. A new Diameter request is generated by the application running on
       the reacting node.

   3. The reacting node determines that an active overload report
       applies to the request, as indicated by the corresponding OCS
       entry.

   4. The reacting node determines if overload abatement treatment
       should be applied to the request. One approach that could be
       taken for each request is to select a uniformly selected random
       number between 1 and 100. If the random number is less than or

equal to the indicated reduction percentage, then the request is
given abatement treatment; otherwise, the request is given normal
routing treatment.

6.2. Reporting Node Behavior

The method a reporting node uses to determine the amount of traffic
reduction required to address an overload condition is an
implementation decision.

When a reporting node that has selected the loss abatement algorithm
determines the need to request a reduction in traffic, it includes an
OC-OLR AVP in answer messages as described in Section 5.2.3.

When sending the OC-OLR AVP, the reporting node MUST indicate a
percentage reduction in the OC-Reduction-Percentage AVP.

The reporting node MAY change the reduction percentage in subsequent
overload reports. When doing so, the reporting node must conform to
overload report handling specified in Section 5.2.3.

6.3. Reacting Node Behavior

The method a reacting node uses to determine which request messages
are given abatement treatment is an implementation decision.

When receiving an OC-OLR in an answer message where the algorithm
indicated in the OC-Supported-Features AVP is the loss algorithm, the
reacting node MUST apply abatement treatment to the requested
percentage of request messages sent.

> Note: The loss algorithm is a stateless algorithm. As a result,
> the reacting node does not guarantee that there will be an
> absolute reduction in traffic sent. Rather, it guarantees that
> the requested percentage of new requests will be given abatement
> treatment.

If the reacting node comes out of the 100% traffic reduction
(meaning, it has received an OLR indicating that no traffic should be
sent, as a result of the overload report timing out), 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%
reduction.

      Note: The goal of this behavior is to reduce the probability of
      overload condition thrashing where an immediate transition from
      100% reduction to 0% reduction results in the reporting node
      moving quickly back into an overload condition.

7. Attribute Value Pairs

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

   Refer to Section 4 of [RFC6733] for more information on AVPs and AVP
   data types.

7.1. OC-Supported-Features AVP

   The OC-Supported-Features AVP (AVP Code 621) is of type Grouped and
   serves two purposes. First, it announces a node's support for the
   DOIC solution in general. Second, it contains the description of the
   supported DOIC features of the sending node. The OC-Supported-
   Features AVP MUST be included in every Diameter request message a
   DOIC supporting node sends.

      OC-Supported-Features ::= < AVP Header: 621 >
                                [OC-Feature-Vector]
                              * [AVP]

7.2. OC-Feature-Vector AVP

   The OC-Feature-Vector AVP (AVP Code 622) is of type Unsigned64 and
   contains a 64-bit flags field of announced capabilities of a DOIC
   node. The value of zero (0) is reserved.

   The OC-Feature-Vector sub-AVP is used to announce the DOIC features
   supported by the DOIC node, in the form of a flag-bits field in which
   each bit announces one feature or capability supported by the node.
   The absence of the OC-Feature-Vector AVP in request messages
   indicates that only the default traffic abatement algorithm described
   in this specification is supported. The absence of the OC-Feature-
   Vector AVP in answer messages indicates that the default traffic
   abatement algorithm described in this specification is selected
   (while other traffic abatement algorithms may be supported), and no
   features other than abatement algorithms are supported.

The following capability is defined in this document:

OLR_DEFAULT_ALGO (0x0000000000000001)

   When this flag is set by the a DOIC reacting node, it means that
   the default traffic abatement (loss) algorithm is supported. When
   this flag is set by a DOIC reporting node, it means that the loss
   algorithm will be used for requested overload abatement.

7.3. OC-OLR AVP

   The OC-OLR AVP (AVP Code 623) is of type Grouped and contains the
   information necessary to convey an overload report on an overload
   condition at the reporting node. The application the OC-OLR AVP
   applies to is identified by the Application-ID found in the Diameter
   message header. The host or realm the OC-OLR AVP concerns is
   determined from the Origin-Host AVP and/or Origin-Realm AVP found in
   the encapsulating Diameter command. The OC-OLR AVP is intended to be
   sent only by a reporting node.

```
        OC-OLR ::= < AVP Header: 623 >
                   < OC-Sequence-Number >
                   < OC-Report-Type >
                   [ OC-Reduction-Percentage ]
                   [ OC-Validity-Duration ]
                 * [ AVP ]
```

7.4. OC-Sequence-Number AVP

   The OC-Sequence-Number AVP (AVP Code 624) is of type Unsigned64. Its
   usage in the context of overload control is described in Section 5.2.

   From the functionality point of view, the OC-Sequence-Number AVP is
   used as a nonvolatile increasing counter for a sequence of overload
   reports between two DOIC nodes for the same overload occurrence.
   Sequence numbers are treated in a unidirectional manner, i.e., two
   sequence numbers in each direction between two DOIC nodes are not
   related or correlated.

7.5. OC-Validity-Duration AVP

   The OC-Validity-Duration AVP (AVP Code 625) is of type Unsigned32 and
   indicates in seconds the validity time of the overload report. The
   number of seconds is measured after reception of the first OC-OLR AVP
   with a given value of OC-Sequence-Number AVP. The default value for
   the OC-Validity-Duration AVP is 30 seconds. When the OC-Validity-
   Duration AVP is not present in the OC-OLR AVP, the default value
   applies. The maximum value for the OC-Validity-Duration AVP is

86,400 seconds (24 hours). If the value received in the OC-Validity-
Duration is greater than the maximum value, then the default value
applies.

7.6. OC-Report-Type AVP

The OC-Report-Type AVP (AVP Code 626) is of type Enumerated. The
value of the AVP describes what the overload report concerns. The
following values are initially defined:

HOST_REPORT 0
   The overload report is for a host. Overload abatement treatment
   applies to host-routed requests.

REALM_REPORT 1
   The overload report is for a realm. Overload abatement treatment
   applies to realm-routed requests.

The values 2-4294967295 are unassigned.

7.7. OC-Reduction-Percentage AVP

The OC-Reduction-Percentage AVP (AVP Code 627) is of type Unsigned32
and describes the percentage of the traffic that the sender is
requested to reduce, compared to what it otherwise would send. The
OC-Reduction-Percentage AVP applies to the default (loss) 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 ignored. The value of
100 means that all traffic is to be throttled, 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 need for the reacting node to apply any traffic abatement.

7.8. AVP Flag Rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules	
				MUST	MUST NOT
OC-Supported-Features	621	7.1	Grouped		V
OC-Feature-Vector	622	7.2	Unsigned64		V
OC-OLR	623	7.3	Grouped		V
OC-Sequence-Number	624	7.4	Unsigned64		V
OC-Validity-Duration	625	7.5	Unsigned32		V
OC-Report-Type	626	7.6	Enumerated		V
OC-Reduction -Percentage	627	7.7	Unsigned32		V

   As described in the Diameter base protocol [RFC6733], the M-bit usage
   for a given AVP in a given command may be defined by the application.

8. Error Response Codes

   When a DOIC node rejects a Diameter request due to overload, the DOIC
   node MUST select an appropriate error response code. This
   determination is made based on the probability of the request
   succeeding if retried on a different path.

      Note: This only applies for DOIC nodes that are not the originator
      of the request.

   A reporting node rejecting a Diameter request due to an overload
   condition SHOULD send a DIAMETER_TOO_BUSY error response, if it can
   assume that the same request may succeed on a different path.

   If a reporting node knows or assumes that the same request will not
   succeed on a different path, the DIAMETER_UNABLE_TO_COMPLY error
   response SHOULD be used. Retrying would consume valuable resources
   during an occurrence of overload.

For instance, if the request arrived at the reporting node without
a Destination-Host AVP, then the reporting node might determine
that there is an alternative Diameter node that could successfully
process the request and that retrying the transaction would not
negatively impact the reporting node. DIAMETER_TOO_BUSY would be
sent in this case.

If the request arrived at the reporting node with a Destination-
Host AVP populated with its own Diameter identity, then the
reporting node can assume that retrying the request would result
in it coming to the same reporting node.
DIAMETER_UNABLE_TO_COMPLY would be sent in this case.

A second example is when an agent that supports the DOIC solution
is performing the role of a reacting node for a non-supporting
client. Requests that are rejected as a result of DOIC throttling
by the agent in this scenario would generally be rejected with a
DIAMETER_UNABLE_TO_COMPLY response code.

9. IANA Considerations

9.1. AVP Codes

   New AVPs defined by this specification are listed in Section 7. All
   AVP codes are allocated from the "AVP Codes" sub-registry under the
   "Authentication, Authorization, and Accounting (AAA) Parameters"
   registry.

9.2. New Registries

   Two new registries have been created in the "AVP Specific Values"
   sub-registry under the "Authentication, Authorization, and Accounting
   (AAA) Parameters" registry.

   A new "OC-Feature-Vector AVP Values (code 622)" registry has been
   created. This registry contains the following:

      Feature Vector Value Name

      Feature Vector Value

      Specification defining the new value

   See Section 7.2 for the initial Feature Vector Value in the registry.
   This specification defines the value. New values can be added to the
   registry using the Specification Required policy [RFC5226].

A new "OC-Report-Type AVP Values (code 626)" registry has been
created. This registry contains the following:

    Report Type Value Name

    Report Type Value

    Specification defining the new value

See Section 7.6 for the initial assignment in the registry. New
types can be added using the Specification Required policy [RFC5226].

10. Security Considerations

   DOIC gives Diameter nodes the ability to request that downstream
   nodes send fewer Diameter requests. Nodes do this by exchanging
   overload reports that directly effect 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. For
   instance, a series of injected realm OLRs with a requested reduction
   percentage of 100% could be used to completely eliminate any traffic
   from being sent to that realm.

   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 their
   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.

10.1. 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 transitive
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 occur 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 an answer. Such an answer would also need to
arrive at a Diameter node via a protected transport connection.
Therefore, implementations MUST validate that an answer containing an
overload report is a properly constructed response to a pending
request prior to acting on the overload report, and that the answer
was received via an appropriate transport connection.

A similar attack involves a compromised but otherwise authorized 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 a peer that applies to a realm
not handled by that peer is suspect. This may require out-of-band,
non-Diameter agreements and/or mechanisms.

   This attack is partially mitigated by the fact that the
   application, as well as host and realm, for a given OLR is
   determined implicitly by respective AVPs in the enclosing answer.
   If a reporting node modifies any of those AVPs, the enclosing
   transaction will also be affected.

10.2. Denial-of-Service Attacks

   Diameter overload reports, especially realm 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 attacks.
   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. In the worst case, where the Diameter application
is being used for access control into an IP network, a coordinated
DoS attack could result in the blockage of all traffic into that
network. Therefore, Diameter nodes MUST NOT honor or forward OLRs
received from peers that are not trusted to send them.

An attacker might use the information in an OLR to assist in DoS
attacks. For example, an attacker could use information about
current overload conditions to time an attack for maximum effect, or
use subsequent overload reports as a feedback mechanism to learn the
results of a previous or ongoing attack. Operators need the ability
to ensure that OLRs are not leaked to untrusted parties.

10.3. Noncompliant Nodes

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 throttle 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.

When a Diameter node sends an overload report, it cannot assume that
all nodes will comply, even if they indicate support for DOIC. A
noncompliant node might continue to send requests with no reduction
in load. Such noncompliance could be done accidentally or
maliciously to gain an unfair advantage over compliant nodes.
Requirement 28 in [RFC7068] indicates that the overload control
solution cannot assume that all Diameter nodes in a network are
trusted. It also requires that malicious nodes not be allowed to
take advantage of the overload control mechanism to get more than
their fair share of service.

10.4. End-to-End Security Issues

The lack of end-to-end integrity features makes it difficult to
establish trust in overload reports received 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. DOIC
nodes MUST strip DOIC AVPs from messages received from peers that are
not trusted for DOIC purposes.

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.

   A DOIC node cannot always automatically detect that a peer also
   supports DOIC. For example, a node might have a peer that is a
   non-supporting agent. If nodes on the other side of that agent
   send OC-Supported-Features AVPs, the agent is likely to forward
   them as unknown AVPs. Messages received across the non-supporting
   agent may be indistinguishable from messages received across a
   DOIC supporting agent, giving the false impression that the non-
   supporting agent actually supports DOIC. This complicates the
   transitive-trust nature of DOIC. Operators need to be careful to
   avoid situations where a non-supporting agent is mistakenly
   trusted to enforce DOIC-related authorization policies.

It is expected that work on end-to-end Diameter security 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 allows 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.

11. References

11.1. Normative References

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

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

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

11.2. Informative References

   [Cx]   3GPP, "Cx and Dx interfaces based on the Diameter
              protocol; Protocol details", 3GPP TS 29.229 12.7.0,
              September 2015.

   [PCC]   3GPP, "Policy and charging control architecture", 3GPP
              TS 23.203 12.10.0, September 2015.

   [RFC4006] Hakala, H., Mattila, L., Koskinen, J-P., Stura, M., and J.
              Loughney, "Diameter Credit-Control Application", RFC 4006,
              DOI 10.17487/RFC4006, August 2005,
              <http://www.rfc-editor.org/info/rfc4006>.

   [RFC7068] McMurry, E. and B. Campbell, "Diameter Overload Control
              Requirements", RFC 7068, DOI 10.17487/RFC7068, November
              2013, <http://www.rfc-editor.org/info/rfc7068>.

   [S13]   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
              12.8.0, September 2015.

Appendix A. Issues Left for Future Specifications

   The base solution for overload control does not cover all possible
   use cases. A number of solution aspects were intentionally left for
   future specification and protocol work. The following subsections
   define some of the potential extensions to the DOIC solution.

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 7.2 and 9 for the required IANA
   steps.

A.2. Agent Overload

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

A.3. New Error Diagnostic AVP

   This specification indicates the use of existing error messages when
   nodes reject requests due to overload. There is an expectation that
   additional error codes or AVPs will be defined in a separate
   specification to indicate that overload was the reason for the
   rejection of the message.

Appendix B. Deployment Considerations

   Non-supporting Agents

      Due to the way that realm-routed requests are handled in Diameter
      networks with the server selection for the request done by an
      agent, network operators should enable DOIC at agents that perform
      server selection first.

   Topology-Hiding Interactions

      There exist proxies that implement what is referred to as Topology
      Hiding. This can include cases where the agent modifies the
      Origin-Host in answer messages. The behavior of the DOIC solution
      is not well understood when this happens. As such, the DOIC
      solution does not address this scenario.

   Inter-Realm/Administrative Domain Considerations

      There are likely to be special considerations for handling DOIC
      signaling across administrative boundaries. This includes
      considerations for whether or not information included in the DOIC
      signaling should be sent across those boundaries. In addition,
      consideration should be taken as to whether or not a reacting node
      in one realm can be trusted to implement the requested overload
      abatement handling for overload reports received from a separately
      administered realm.

Appendix C. Considerations for Applications Integrating the DOIC
             Solution

   This section outlines considerations to be taken into account when
   integrating the DOIC solution into Diameter applications.

C.1. Application Classification

   The following is a classification of Diameter applications and
   request types. This discussion is meant to document factors that
   play into decisions made by the Diameter entity 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.

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

   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 [S13], 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 [Cx] is an
   example of a pseudo-session application.

The handling of overload reports must take the type of application
into consideration, as discussed in Appendix C.2.

C.2. Implications of Application Type Overload

   This section discusses considerations for mitigating overload
   reported by a Diameter entity. This discussion focuses on the type
   of application. Appendix C.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 it 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 cleanup procedures.

C.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 Policy and Charging Control (PCC)
   architecture [PCC], 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 as 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 request.

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 exist 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
   [Cx] application are examples of pseudo-session requests.

C.4. Request Type Overload Implications

   The request classes identified in Appendix C.3 have implications on
   decisions about which requests should be throttled first. The
   following list of request treatments regarding throttling is provided
   as guidelines for application designers when implementing the
   Diameter overload control mechanism described in this document. The
   exact behavior regarding throttling is a matter of local policy,
   unless specifically defined for the application.

Independent Requests:

   Independent requests can generally be given equal treatment when
   making throttling decisions, unless otherwise indicated by
   application requirements or local policy.

Session-Initiating Requests:

   Session-initiating requests often represent more work than
   independent or intra-session requests. Moreover, session-
   initiating requests are typically followed by other session-
   related requests. Since the main objective of 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. In the absence of
   local policies or application-specific requirements to the
   contrary, 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, it 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 types of intra-sessions requests, requests that
   terminate a session and the remainder of intra-session requests.
   Implementers and operators may choose to throttle session-
   terminating requests less aggressively in order to gracefully
   terminate sessions, allow cleanup of the related resources (e.g.,
   session state), and avoid the need for additional intra-session
   requests. Favoring session termination requests may reduce 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.

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

Authors' Addresses

   Jouni Korhonen (editor)
   Broadcom Corporation
   3151 Zanker Road
   San Jose, CA 95134
   United States

   Email: jouni.nospam@gmail.com

   Steve Donovan (editor)
   Oracle
   7460 Warren Parkway
   Frisco, Texas 75034
   United States

   Email: srdonovan@usdonovans.com

   Ben Campbell
   Oracle
   7460 Warren Parkway
   Frisco, Texas 75034
   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