Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2014

J. Korhonen, Ed.
Broadcom Communications
S. Donovan
B. Campbell
Oracle

October 21, 2013

Diameter Overload Indication Conveyance
draft-docdt-dime-ovli-00.txt

Abstract

This specification documents a Diameter Overload Information Conveyance (DOIC) 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].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on April 24, 2014.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a base solution for the Diameter Overload Information Conveyance (DOIC). The requirements for the solution are described and discussed in the corresponding design requirements document [I-D.ietf-dime-overload-reqs]. Note that the overload control solution defined in this specification does not address all the requirements listed in [I-D.ietf-dime-overload-reqs]. A number of overload control related features are left for the future specifications. See Appendix A for more detailed discussion on those.

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.

[OpenIssue: Is a server farm assumed to support a single realm? That is, does it support a set of applications in a single realm?]

Server Front End

A Server Front End (SFE) is a role that can be performed by a Diameter agent -- either a relay or a proxy -- that sits between Diameter clients and a Server Farm. An SFE can perform various functions for the server farm it sits in front of. This includes some or all of the following functions:

* Diameter Routing
* Diameter layer load balancing
* Load Management
* Overload Management
* Topology Hiding
* Server Farm Identity Management

[OpenIssue: We used the concept of a server farm and SFE for internal discussions. Do we still need those concepts to explain the mechanism? It doesn't seem like we use them much.]

Diameter Routing:

Diameter Routing determines the destination of Diameter messages addressed to either a Diameter Realm and Application in general, or to a specific server using Destination-Host. This function is defined in [RFC6733]. Application level routing specifications that expand on [RFC6733] also exist.

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.

Load Management:

This functionality ensures that the consolidated load state for the server farm is collected, and processed. The exact algorithm for computing the load at the SFE is implementation specific but enough semantic of the conveyed load information needs to be specified so that deterministic behavior can be ensured.

Overload Management:

The SFE is the entity that understands the consolidated overload state for the server farm. Just as it is outside the scope of

this document to specify how a Diameter server calculates its overload state, it is also outside the scope of this document to specify how an SFE calculates the overload state for the set of servers. This document describes how the SFE communicates Overload information to Diameter Clients.

[OpenIssue: Does this mean the way servers communicate overload info to an SFE is also out of scope? It would be nice if the mechanism is useful for that purpose.]

Topology Hiding:

Topology Hiding is loosely defined as ensuring that no Diameter topology information about the server farm 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 server farm. It can also include hiding the number of various Diameter entities in the server farm. 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.

Server Farm Identity Management:

Server Farm Identity Management (SFIM) is a mechanism that can be used by the SFE to present a single Diameter identity that can be used by clients to send Diameter requests to the server farm. This requires that the SFE modifies Origin-Host information in answers coming from servers in the server farm. An agent that performs SFIM appears as a server from the client's perspective.

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".

3. Solution Overview

3.1. Architectural Assumptions

This section describes the high-level architectural and semantic assumptions that underly 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 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. The primary differentiator 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.

The 3GPP-defined S6a application is an example of a session-less application. The following, copied from section 7.1.4 of 29.272, explicitly states that sessions are implicitly terminated and that the server does not maintain session state:

> "Between the MME and the HSS and between the SGSN and the HSS and between the MME and the EIR, Diameter sessions shall be implicitly terminated. An implicitly terminated session is one for which the server does not maintain state information. The client shall not send any re-authorization or session termination requests to the server.
>
> The Diameter base protocol includes the Auth-Session-State AVP as the mechanism for the implementation of implicitly terminated sessions.
>
> The client (server) shall include in its requests (responses) the Auth-Session-State AVP set to the value NO_STATE_MAINTAINED (1),

> as described in [RFC6733]. As a consequence, the server shall not maintain any state information about this session and the client shall not send any session termination request. Neither the Authorization-Lifetime AVP nor the Session-Timeout AVP shall be present in requests or responses."

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.

Pseudo-session applications: While this class of application does not use the Diameter Session-ID AVP to correlate requests, there is an implied ordering of transactions defined by the application. Transactions in a pseudo-session typically need to be handled by the same server. The 3GPP defined Cx application [reference] is an example of a pseudo-session application.

The accounting application defined in [RFC6733] and the Credit-Control application defined in [RFC4006] are examples of Diameter session-based applications.

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. Section 3.1.5 discusses considerations for handling overload conditions based on the network deployment scenario.

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-stateful applications: Pseudo stateful applications are also stateless applications in that there is no session Diameter state maintained between transactions. There is, however, an implied ordering of requests. As a result, decisions about which transactions to reject as a result of an overloaded entity 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.

Stateful applications: Overload handling for stateful applications must take into consideration the work associated with setting up an maintaining a session. As such, the Diameter entity handling overload for a stateful application might tend to reject new session requests before rejecting intra-session requests. In addition, session ending requests might be given a lower chance of being rejected, since rejecting session ending requests could result in session status being out of sync between the Diameter clients and servers, while successful execution might actually free up resources. Nodes that reject mid-session requests will need to consider whether the rejection invalidates the session, and any session clean-up that may be required.

3.1.3. Request Transaction Classification

Independent Request: An independent request is not a part of a Diameter session 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, most notably in the 3GPP PCC architecture, where multiple Diameter sessions are correlated and must be handled by the same Diameter server. This is a special case of a Session-Initiating Request. Gx CCR-I requests and Rx AAR messages are examples of correlated session-initiating requests.

[OpenIssue: The previous paragraph needs references.]

Intra-Session Request: An intra-session request is a request that uses a session-id for an already established session. 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. CCR-U and CCR-T requests defined in [RFC4006] are further examples of intra-session requests.

Pseudo-Session Requests: Pseudo session requests are independent requests and, as such, the request transactions are not tied together using the Diameter session-id. 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 application are examples of pseudo-session requests.

[OpenIssue: This section offers discusses priorities around throttling of requests. Should we also discuss considerations for diverting requests non-overloaded destinations?]

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.

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

Session-creating requests: Session-creating requests represent more work than independent or intra-session requests. As such, throttling decisions might favor intra-session requests over session-creating requests. Individual session-creating requests can be given equal treatment when making throttling decisions.

Correlated session-creating requests: A Request that results in a new binding, where the binding is used for routing of subsequent session-creating requests, represents more work 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 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 is a request that ends a session. The second is a request that is used to convey session related state between the Diameter client and server. Session ending request should be throttled less aggressively in order to keep session state consistent between the client and server, and possibly reduce the sessions 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 Deployment Scenarios

This section discusses various Diameter network deployment scenarios and the implications of those deployment models on handling of overload reports.

The scenarios vary based on the following:

- The presence or absence of Diameter agents
- Which Diameter entities support the DOIC extension
- The amount of the network topology understood by Diameter clients
- The complexity of the Diameter server deployment for a Diameter application
- Number of Diameter applications supported by Diameter clients and Diameter servers

Without consideration for which elements support the DOIC extension,

the following is a representative list of deployment scenarios:

- Client --- Server
- Client --- Multiple equivalent servers
- Client --- Agent --- Multiple equivalent servers
- Client --- Agent [--- Agent] --- Partitioned server

- Client --- Edge Agent [--- Edge Agent] --- { Multiple Equivalent Servers | Partitioned Servers }
- Client --- Session Correlating Agent --- Multiple Equivalent Servers

[OpenIssue: Do the "multiple equivalent servers" cases change for session-stateful applications? Do we need to distinguish equivalence for session-initiation requests vs intra-session requests?]

The following is a list of representative DOIC deployment scenarios:

- Direct connection between a DOIC client and a DOIC server
- DOIC client --- one or more non-DOIC agent(s) --- DOIC server
- DOIC client --- DOIC agent --- DOIC server
- Non-DOIC client --- DOIC agent --- DOIC server
- Non-DOIC client --- DOIC agent --- Mix of DOIC and non-DOIC servers
- DOIC client --- DOIC agent --- Partitioned/Segmented DOIC server
- DOIC client --- DOIC agent --- DOIC agent --- Partitioned/ Segmented DOIC server
- DOIC client --- DOIC edge agent --- DOIC edge agent --- DOIC server

3.1.6. 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 defined. This is important because agents may actively participate in the handling of 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, and acting as a server front end for a server farm of real Diameter servers.

Since the solution defined in this specification must not break the Diameter base protocol assumptions 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:

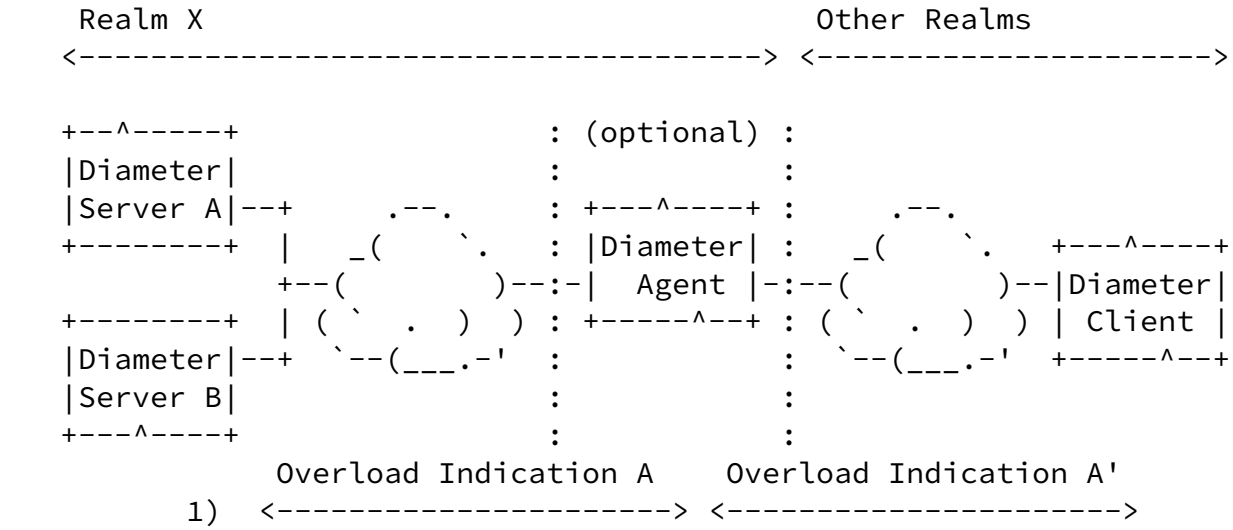
- If a Diameter agent presents itself as the "end node", perhaps acting as a topology hiding SFE, the DOIC mechanism MUST NOT leak information of the Diameter nodes behind it. From the Diameter client point of view the final destination to its requests and the original source for the answers MUST be the Diameter agent. This requirement means that such a Diameter agent acts as a back-to-back-agent for DOIC purposes. How the agent in this case appears to the Diameter nodes it is representing (i.e. the real Diameter servers), is an implementation and a deployment specific within the realm the Diameter agent is deployed.

- This requirement also implies that 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 a given server identified by the Origin-Host identity.

[OpenIssue: We've discussed multiple situations where an agent might insert an OLR. I don't think we mean to force them to always perform topology hiding or SFIM in order to do so. We cannot assume that an

OLR is always "from" or "about" the Origin-Host. Also, the section seems to assume that topology hiding agents act as b2b overload agents, but non-topology hiding agents never do. It don't think that's the right abstraction. It's possible that topology-hiding agents must do this, but I don't think we can preclude non-topology hiding agents from also doing it, at least some of the time.]

3.1.7. Simplified Example Architecture



Simplified architecture choices for overload indication delivery

[OpenIssue: Need to clarify the meaning of option 2 with the agent in place. Does this mean the agent is not an Overload Endpoint?]

3.2. Conveyance of the Overload Indication

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

features.

3.2.1. Negotiation and Versioning

Since the Diameter overload control mechanism is also designed to work over existing application (i.e., the piggybacking principle), a proper negotiation is hard to accomplish. The "capability negotiation" is based on the existense of specific non-mandatory APVs, such as the OC-Feature-Vector AVP (see Section 4.1. Although the OC-Feature-Vector 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.

3.2.2. Transmission of the Attribute Value Pairs

The Diameter overload control APVs SHOULD always be sent as an optional AVPs. This requirement stems from the fact that piggybacking overload control information on top of existing application cannot really use AVPs with the M-bit set. However, there are certain exceptions as explained in Section 5.4.

From the Diameter overload control functionality point of view, the "Reacting node" is always the requester of the overload report information and the "Reporting node" is the provider of the overload report. The overload report or the capability information in the request message is always interpreted as an announcement of a "capability". The capability information and the overload report in the answer is always interpreted respectively as a report of supported common functionality and as a status report of an overload condition.

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 encoded 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 Overload Indication Attribute Value Pairs (AVPs).

4.1. OC-Feature-Vector AVP

The OC-Feature-Vector AVP (AVP code TBD1) is type of Unsigned64 and contains a 64 bit flags field of supported capabilities of an overload control endpoint. Receiving the OC-Feature-Vector AVP with the value 0 indicates that two endpoints do not share a single common capability (or a capability they could agree based on the local policy and/or configuration). A request message initiating endpoint (a reacting node) MUST NOT send the OC-Feature-Vector AVP with the value 0.

[OpenIssue: We need further discussion on whether the "no shared capability" case is allowed, or if we guarantee certain basic levels of compatibility by using mandatory-to-support defaults.]

An overload control endpoint (a reacting node) MAY include this AVP to indicate its capabilities to the other overload control endpoint (the reporting node). For example, the endpoint (reacting node) may indicate which traffic abatement algorithms it supports in addition to the default.

[OpenIssue: There is an ongoing discussion as to whether the OC-Feature-Vector AVP should be an optional (MAY vs MUST) way to declare support, where new Diameter applications could define other ways, or whether this should be the "one true" way. The latter approach

prevents agents that are not application aware from supporting DOIC, but the latter may reduce the flexibility for defining new applications.]

During the message exchange the overload control endpoints express their common set of supported capabilities. The endpoint sending a request (the reacting node) includes the OC-Feature-Vector AVP with those flags set that correspond what it supports. The endpoint that sends the answer (the reporting node) also includes the OC-Feature-Vector AVP with flags set to describe the capabilities it both supports and agrees with the request sender (e.g., based on the local policy and/or configuration). The answer sending endpoint (the reporting node) does not need to advertise those capabilities it is not going to use with the request sending endpoint (the reacting node).

Note that when the OC-Feature-Vector AVP is used together with the OC-OLR AVPs, the contents of the announced features and the contents of the OC-OLR AVPs MUST NOT contradict each other. In a case they do, the receiver of contradicting information SHOULD discard the AVPs as if they were not present to start with and log the event.

In some cases a single flag bit in the OC-Feature-Vector AVP is not verbose enough to describe all of the advertised capability. This concerns the situation where the OC-Feature-Vector AVP is sent in a request message. In this particular case, the OC-OLR AVP MUST contain the rest of the required parameters. For example, if the advertised capability concerns an abatement algorithm that needs more algorithm specific parameters to agree on, then the OC-OLR abatement algorithm specific AVPs MUST contain the rest of the parameter information.

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.

4.2. OC-OLR AVP

The OC-OLR AVP (AVP code TBD2) is type of Grouped and contains the necessary information to convey an overload report. OC-OLR may also be used to convey additional information about an extension that is declared in the OC-Feature-Vector AVP.

Since the OC-OLR AVP contains information that may be critical for handling overload conditions, reporting nodes SHOULD place the AVP as early in the Diameter message as possible.

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 found from the encapsulating Diameter message.

[OpenIssue: There is ongoing discussion on whether it's best to infer information like application, realm, reporting node identity, etc, from the enclosing Diameter message vs making the overload reports self-contained.]

```
OC-OLR ::= < AVP Header: TBD2 >
           < TimeStamp >
           [ ValidityDuration ]
           [ ReportType ]
         * [ OC-Algorithm ]
         * [ AVP ]
```

The TimeStamp AVP indicates when the original OC-OLR AVP with the current content was created. It is possible to replay the same OC-OLR AVP multiple times between the overload endpoints, however, when the OC-OLR AVP content changes or the other information sending endpoint wants the receiving endpoint to update its overload control information, then the TimeStamp AVP MUST contain a new value.

[OpenIssue: Is this necessarily a timestamp, or is it just a sequence number that can be implemented as a timestamp? We should also consider whether either a timestamp or sequence number is needed for protection against replay attacks.]

4.3. TimeStamp AVP

The TimeStamp AVP (AVP code TBD3) is type of Time. Its usage in the context of the overload control is described in Section 4.2. From the functionality point of view, the TimeStamp AVP is merely used as a non-volatile increasing counter between two overload control endpoints.

4.4. ValidityDuration AVP

The ValidityDuration AVP (AVP code TBD4) is type of Unsigned32 and describes the number of seconds the OC-OLR AVP and its content is valid since the creation of the OC-OLR AVP (as indicated by the TimeStamp AVP).

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 4.8 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. Rather, an overload endpoint should explicitly signal either the continuance of the overload condition by sending an new overload report. This new report would indicate a continuance of the overload condition by including a non-zero ValidityDuration value, or indicate the end of the condition by including a zero value. This approach leaves no need for the reacting node to reason or guess the current condition of the reporting node.

4.5. ReportType AVP

The ReportType 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 Reserved.

1 Destination-Host report. The overload treatment should apply to requests that the sender knows will reach the overloaded server. For example, requests with a Destination-Host AVP indicating the server.

2 Realm (aggregated) report. The overload treatment should apply to all requests bound for the overloaded realm.

The ReportType 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 requests that are targeted to a specific

server by the presence of a Destination-Host AVP than for requests that can be handled by any server in a realm. The example in Appendix B.3 illustrates this usage.

[OpenIssue: There is an ongoing discussion about whether the ReportType AVP is the right way to solve that issue, and whether it's needed at all.]

[OpenIssue: ReportType should probably be extensible, and have its own IANA table.]

4.6. OC-Algorithm AVP

The OC-Algorithm AVP (AVP code TBD6) is type of Grouped. The AVP contains the necessary sub-AVPs and information for the use for the traffic abatement algorithm. The OC-Algorithm AVP serves as a generic template for all future traffic abatement algorithms.

This specification defines an identifier for the default (loss) algorithm (see Section 4.1 for the OC-Feature-Vector flag corresponding to the algorithm), as well as the format and meaning of that algorithm's input parameter.

```
OC-Algorithm ::= < AVP Header: TBD6 >
                 < Algorithm-ID >
                 [ Reduction-Percentage ]
               * [ AVP ]
```

As already discussed in Section 4.1 in certain cases, the Algorithm AVP MAY be used in a request message together with the OC-Feature-Vector AVP to describe the detailed parameterization of the abatement

algorithm to the other endpoint (i.e. to the reporting node). This implies, the possible future algorithms and their sub-AVPs must be designed accordingly.

4.7. Algorithm-ID AVP

The Algorithm-ID AVP (AVP code TBD7) is type of Enumerated and identifies the traffic abatement algorithm the OC-Algorithm AVP "describes" and implements. This specification defines the following algorithms:

0 Reserved.

1 Default (loss) algorithm.

4.8. Reduction-Percentage AVP

The 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 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 sender of the information is under a severe load and ceases to process any new messages. The value of 0 means that the sender of the information is in a stable state and has no requests to the other endpoint to apply any traffic abatement.

[OpenIssue: We should consider an algorithm independent way to end an overload condition. Perhaps setting the validitytime to zero? Counter comment; since the abatement is based on a specific algorithm, it is natural to indicate that from the abatement algorithm point of view status quo has been reached.]

Since a Reduction-Percentage of 100% prevents the reporting node from explicitly ending the overload condition, such a condition can only end due to a report timeout. When an overload control endpoint comes out of the 100 percent traffic reduction as a result, the following concerns are RECOMMENDED to be applied. The endpoint sending the traffic should be conservative and, for example, first send few

"probe" messages to learn the overload condition of the other endpoint before converging to any traffic level decided by the sender. Similar concerns actually apply in all cases when the overload report times out unless the previous overload report stated 0 percent reduction.

4.9. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
OC-Feature-Vector	TBD1	x.x	Unsigned64		V
OC-OLR	TBD2	x.x	Grouped		V
TimeStamp	TBD3	x.x	Time		V
ValidityPeriod	TBD4	x.x	Unsigned32		V

ReportType	TBD5	x.x	Enumerated		V
OC-Algorithm	TBD6	x.x	Grouped		V
Algorithm-ID	TBD7	x.x	Enumerated		V
Reduction -Percentage	TBD8	x.x	Unsigned32		V

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. An overload control "association"

exists between two Diameter nodes that exchanging overload control information. These nodes are called "overload control endpoints". These endpoints, namely the "reacting node" and the "reporting node" do not need to be adjacent Diameter peer nodes, nor do 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 endpoint 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.

[Editor's note: a picture illustrating the endpoint concept would be useful.]

5.2. Piggybacking Principle

The overload control solution defined AVPs are essentially 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-Feature-Vector AVP into existing commands (this has an assumption that the application CCF allows adding new AVPs into the Diameter messages.

In a case of newly defined Diameter applications, it is RECOMMENDED to add and define how overload control mechanisms works on that application. Using OC-Feature-Vector and OC-OLR AVPs is optional, and is intended only existing applications.

[OpenIssue: The guidance in the previous paragraph depends on the outcome of the open issue mentioned at the definition of OC-Feature-Vector.]

Note that the overload control solution does not have fixed server and client roles. The overload control endpoint role is determined based on the sent message type: whether the message is a request for overload information (i.e. sent by a "reacting node") or an overload report (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 Negotiation

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 negotiation or rather the capability announcement relies on the exchange of the OC-Feature-Vector and OC-OLR AVPs between the endpoints. The negotiation solution also works when carried out on existing applications. For the newly defines application the negotiation can be more exact based on the application specification. The negotiated set of capabilities MUST NOT change during the life time of the Diameter session (or transaction in a case of non-session maintaining applications).

[OpenIssue: Some of the guidance in the previous paragraph depends on the outcome of the open issue mentioned at the definition of OC-Feature-Vector.]

[OpenIssue: We need to think more about the general flow for capabilities negotiation. Call flows would be helpful here. A counter comment: the text in Section 4.1 should be rather clear now regarding the capability negotiation.]

5.3.1. Request Message Initiator 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 a Diameter request message the OC-Feature-Vector AVP with those capability flag bits set that it supports and is willing to use for this Diameter session (or transaction in a case of a non-session state maintaining

applications). In a case of session maintaining applications the request message initiating endpoint does not need to do the capability announcement more than once for the lifetime of the Diameter session. In a case of non-session maintaining applications, 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.

[OpenIssue: We need to think about the lifetime of a capabilities declaration. It's probably not the same as for a session. We have had proposals that the feature vector needs to go into every request sent by an OC node. For peer to peer cases, this can be associated with connection lifetime, but it's more complex for non-adjacent OC support.]

If the OC-Feature-Vector AVP does not have enough information about the supported feature or the traffic abatement algorithm, then the request message initiating endpoint MUST also include the OC-OLR AVP with an appropriate content in it (such as a rate based abatement algorithm would include the desired rate information AVPs inside the OC-OLR AVP). See the discussion in Section 4.1 and in Section 4.6.

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-Feature-Vector and/or OC-OLR AVPs in the Diameter answer for existing application. For the newly defined applications the support for the overload control is already part of the application specification. Based on capability knowledge the request message initiating endpoint can select the preferred common traffic abatement algorithm and act accordingly for the subsequent message exchanges.

5.3.2. Answer Message Initiating Endpoint Considerations

When a remote endpoint (i.e. a "reporting node") receives a request message in can detect whether the request message initiating endpoint has support for the overload control solution based on the presence of the OC-Feature-Vector AVP and possibly the OC-OLR 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-Feature-Vector AVP and optionally the contents of the OC-OLR 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. It is RECOMMENDED that the answer message initiating endpoint selects one

common traffic abatement algorithm even if it would support multiple. 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 the created session (or transaction in a case of non-session state maintaining applications).

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-Feature-Vector 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. Sender Endpoint Considerations

5.5.2. Receiver Endpoint Considerations

[OpenIssue: did we now agree that e.g. a server can refrain sending OLR in answers based on some magical algorithm? (Note: We seem to have consensus that a server MAY repeat OLRs in subsequent messages, but is not required to do so, based on local policy.)]

[OpenIssue: We need to define some rules about throttling at an agent. In particular, that the agent needs to send errors back downstream if it drops requests, and propose a specific error code for this purpose.]

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 4.1 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].

Section 4.5 defines a new "Overload Report Type" registry with its initial assignments. New types can be added using the Specification Required policy [RFC5226].

Section 4.7 defines a new "Overload Control Algorithm" 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 currently 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 [I-D.ietf-dime-overload-reqs] 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.

[OpenIssue: This requires that a responding node be able to tell a peer-generated OLR from one generated by a non-adjacent node. One way of doing this would be to include the identity of the node that generated the report as part of the OLR]

[OpenIssue: Do we need further language about what rules an agent should apply before forwarding an OLR?]

The lack of end-to-end protection creates a tension between two requirements from the overload control requirements document. [I-D.ietf-dime-overload-reqs] Requirement 34 requires the ability to send overload reports across intermediaries (i.e. agents) that do not support overload control mechanism. Requirement 27 forbids the mechanism from adding new vulnerabilities or increasing the severity of existing ones. A non-supporting agent will most likely forward overload reports without inspecting them or applying any sort of validation or authorization. This makes the transitive trust issue considerably more of a problem. Without

the ability to authenticate and integrity protect overload reports across a non-supporting agent, the mechanism cannot comply with both requirements.

[OpenIssue: What do we want to do about this? Req27 is a normative MUST, while Req34 is "merely" a SHOULD. This would seem to imply that 27 has to take precedent. Can we say that overload reports MUST NOT be sent to and/or accepted from non-supporting agents until such time we can use end-to-end security?]

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:

- Eric McMurry

- Hannes Tschofenig
- Ulrich Wiehe
- Jean-Jacques Trottin
- Lionel Morand
- Maria Cruz Bartolome
- Martin Dolly
- Nirav Salot
- Susan Shishufeng

10. Acknowledgements

...

11. References

11.1. 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.

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

11.2. Informative References

[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.

[I-D.ietf-dime-overload-reqs]
McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", draft-ietf-dime-overload-reqs-13 (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.

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, 4.7 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 underspecified. 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 APVs should be discussed and possible be recommended to be used.

A.4. Load

This specification defines the mechanism for a Diameter end-point to request a reduction in traffic. The full solution envisioned by the Diameter overload requirements also included a mechanism to communicate load that a Diameter node is able to handle. This capability is expected to help to decrease the oscillation of overload events. This load capability has been left for follow on work.

Appendix B. Examples

B.1. 3GPP S6a interface overload indication

[TBD: Would cover S6a MME-HSS communication with several topology choices (such as with or without DRA, and with "generic" agents).]

B.2. 3GPP PCC interfaces overload indication

[TBD: Would cover Gx/Rx and maybe S9..]

B.3. Mix of Destination-Realm routed requests and Destination-Host reouted requests

[TBD: Add example showing the use of Destination-Host type OLRs and Realm type OLRs.]

Authors' Addresses

Jouni Korhonen (editor)
Broadcom Communications
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