Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 7, 2015
B. Campbell
S. Donovan, Ed.
Oracle
JJ. Trottin
Alcatel-Lucent
March 6, 2015

Architectural Considerations for Diameter Load Information
draft-campbell-dime-load-considerations-01

Abstract

RFC 7068 describes requirements for Overload Control in Diameter. This includes a requirement to allow Diameter nodes to send "load" information, even when the node is not overloaded. The Diameter Overload Information Conveyance (DOIC) solution describes a mechanism meeting most of the requirements, but does not currently include the ability to send load information. This document explores some architectural considerations for a mechanism to send Diameter load information.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC7068] describes requirements for Overload Control in Diameter [RFC6733]. At the time of this writing, the DIME working group is working on the Diameter Overload Information Conveyance (DOIC) mechanism [I-D.ietf-dime-ovli] . As currently specified, DOIC fulfills some, but not all, of the requirements.

In particular, DOIC does not fulfill Req 24, which requires a mechanism where Diameter nodes can indicate their current load, even if they are not currently overloaded. DOIC also does not fulfill Req 23, which requires that nodes that divert traffic away from overloaded nodes be provided with sufficient information to select targets that are most likely to have sufficient capacity.

There are several other requirements in RFC 7068 that mention both overload and load information that are only partially fulfilled by DOIC.

The DIME working group explicitly chose not to fulfill these requirements in DOIC due to several reasons. A principal reason was that the working group did not agree on a general approach for conveying load information. It chose to progress the rest of DOIC, and defer load information conveyance to a DOIC extension or a separate mechanism.

This document describes some high level architectural decisions that the working group will need to consider in order to solve the load-related requirements from RFC 7068.

At the time of this writing, there have been several attempts to create mechanisms for conveyance of both load and overload control information that were not adopted by the DIME working group. While these drafts are not expected to progress, they may be instructive when considering these decisions.

- [I-D.tschofenig-dime-dlba] proposed a dedicated Diameter application for exchanging load balancing information.

- [I-D.roach-dime-overload-ctrl] described a strictly peer-to-peer exchange of both load and overload information in new AVPs piggy-backed on existing Diameter messages.

- [I-D.korhonen-dime-ovl] described a dedicated Diameter application for exchanging both load and overload information.

2. Differences between Load and Overload information

Previous discussions of how to solve the load-related requirements in [RFC7068] have shown that people do not have an agreed-upon concept of how "load" information differs from "overload" information. The two concepts are highly interrelated, and so far the working group has not defined a bright line between what constitutes load information and what constitutes overload information.

In the opinion of the authors, there are two primary differences. First, a Diameter node always has a load. At any given time that load maybe effectively zero, effectively fully loaded, or somewhere in between. In contrast, overload is an exceptional condition. A node only has overload information when it in an overloaded state.

Furthermore, the relationship between a node’s load level and overload state at any given time may be vague. For example, a node may normally operate at a "fully loaded" level, but still not be considered overloaded. Another node may declare itself to be "overloaded" even though it might not be fully "loaded".

Second, Overload information, in the form of a DOIC Overload Report (OLR) [I-D.ietf-dime-ovli] indicates an explicit request for action on the part of the reacting node. That is, the OLR requests that the reacting node reduce the offered load -- the actual traffic sent to the reporting node after overload abatement and routing decisions are made -- by an indicated amount or to an indicated level. Effectively, DOIC provides a contract between the reporting node and the reacting node.

In contrast, load is informational. That is, load information can be considered a hint to the recipient node. That node may use the load information for load balancing purposes, as an input to certain overload abatement techniques, to make inferences about the likelihood that the sending node becomes overloaded in the immediate future, or for other purposes.

None of this prevents a Diameter node from deciding to reduce the offered load based on load information. The fundamental difference is that an overload report requires that reduction. It is also reasonable for a Diameter node to decide to increase the offered load based on load information.

3. How is Load Information Used?

[RFC7068] contemplates two primary uses for load information. Req 23 discusses how load information might be used when performing diversion as an overload abatement technique, as described in [I-D.ietf-dime-ovli]. When a reacting node diverts traffic away from an overloaded node, it needs load information for the other candidates for that traffic in order to effectively load balance the diverted load between potential candidates. Otherwise, diversion has a greater potential to drive other nodes into overload.

Req 24 discusses how Diameter load information might be used when no overload condition currently exists. Diameter nodes can use the load information to make decisions to try to avoid overload conditions in the first place. Normal load-balancing falls into this category. A node might also take other proactive steps to reduce offered load based on load information, so that the loaded node never goes into overload in the first place.

If the loaded nodes are Diameter servers (or clients in the case of server-to-client transactions), both of these uses are most effectively accomplished by a Diameter node that performs server selection. Typically, server selection is performed by a node (a client or an agent) that is an immediate peer of the server. However, there are scenarios (see Section 10) where a client or proxy that is not the immediate peer to the selected servers performs server selection. In this case, the client or proxy enforces the server selection by inserting a Destination-Host AVP.

> For example, a Diameter node (e.g. client) can use a redirect agent to get candidate destination host addresses. The redirect agent might return several destination host addresses, from which the Diameter node selects one. The Diameter node can use load information received from these hosts to make the selection.

Just as load information can be used as part of server selection, it can also be used as input to the selection of the next-hop peer to which a request is to be routed.

One area that requires thought is how load information is used, if at all, in the presence of an overload report from the same Diameter node. It might be that the load information from that Diameter node is ignored for the duration of the time that the overload report is in effect. It might also be possible that the load information can aid in the routing of non-abated requests targeted for the overloaded Diameter node.

4. Piggy-Backing vs a Dedicated Application.

[I-D.roach-dime-overload-ctrl] imbeds load and overload information onto messages of existing applications. This is known as a "piggy-back" approach. Such an approach has the advantage of not requiring new messages to carry load information. It has an additional advantage of scaling with load; that is, the more the transaction load, the more opportunities to send load information.

DOIC [I-D.ietf-dime-ovli] also uses a piggy-backed approach to send OLRs. Given the potentially tight connection between load and overload information, there may be advantages to maintaining consistency with DOIC.

[I-D.tschofenig-dime-dlba] used a dedicated application to carry load information. This application has quasi-subscription semantics, where a client requests updates according to a cadence. The server can send unsolicited updates if the load level changes between updates in the cadence.

[I-D.korhonen-dime-ovl] also used a dedicated application, but allowed nodes to send unsolicited reports containing load and overload information. The mechanism has an issue that the sender of load information may not know which other nodes need the information. It may be possible to infer that information from other application messages handled by the sender.

Another potential approach is that of a dedicated Diameter application with a slightly different subscription semantic than that of [I-D.tschofenig-dime-dlba]. In such an application, a node that consumes load information sends a Diameter request to the source of the load information. This request indicates that the consumer wishes to receive load information for some period of time. The load source would send periodic Diameter requests indicating the current load level, until such time that the subscription period expired, or the subscribe explicitly unsubscribed. After the initial notification, the sender would only send updates when the load level changed.

5. Which Nodes Exchange Load Information?

Section 10 illustrates a number of Diameter network topologies where load information may be useful. However, there are potentially limitless configurations where load information might be used to make peer and server selection choices. Nodes may be unaware of the topology beyond their immediate peers, which may limit the utility of load information for nodes beyond that peer.

There may in fact be scenarios where a peer-selection decision is impacted by the load of non-adjacent nodes, or where a node needs to force selection of a particular non-adjacent server. While explicit knowledge of the load of such non-adjacent nodes may be useful in such decisions, the working group should consider whether this utility is worth the added complexity.

> For instance, one approach would be to support two types of load reports, endpoint load reports and peer load reports. In this scenario, load reports would likely require an AVP indicating the Diameter node to which the report applies. This would be needed to differentiate between endpoint load reports and next hop load reports. This would imply that a single message will likely have two load reports, one for the endpoint and one for the next hop. This would also add complexity in agents, sometimes needing to strip next hop load reports and sometimes not.

Previous load related efforts have made different assumptions about which Diameter nodes exchange load information.

[I-D.roach-dime-overload-ctrl] operated in a strictly peer-to-peer mode. Each node would only learn the load (and overload) information from its immediate peers.

[I-D.korhonen-dime-ovl] and [I-D.tschofenig-dime-dlba] are each effectively any-to-any. That is, they each allowed any node to send load information to any other node that supported the dedicated overload or load application, respectively.

In the latter case, load is effectively sent between clients and servers of the dedicated application, but those roles may not match the client and server roles for the "main" Diameter applications in use. For example, a pair of adjacent diameter agents might be "client" and "server" for the dedicated "load" application, effectively creating a peer-to-peer relationship similar to that of [I-D.roach-dime-overload-ctrl].

Each approach has advantages. Peer-to-peer transmission covers the case when server selection is done by the servers immediate peers. Additionally, selection of non-terminal nodes is generally done on a peer-to-peer basis. If the loaded node is an agent, for example, the load information is only useful to immediate peers. Peer-to-peer transmission is the easiest to negotiate. (See Section 9)

Any-to-Any transmission offers more flexibility, and could potentially cover the case where server selection is done by nodes that are not peers to the candidate servers.

6. Scope of Load Information

Load information could refer to several different scopes:

- Load of a Node -- The load information refers to the load for an entire Diameter host, that is a Client, Agent, or Server described by a Diameter Identity.

- Load of an Application -- The load for a specific Diameter node that supports multiple Diameter applications might differ between applications.

- Load of a set of nodes -- The load would likely be the aggregated load of the nodes in the set. This would likely require a separate Diameter identity be assigned to the set of nodes and the load information would be associated with that Diameter identity.

- Aggregate Load -- Different paths via different agents may exist between a node making a peer selection decision and the final destination of the request. The least loaded destination may only be reachable via certain peers.

- Load of an agent plus load of a Diameter endpoint -- Different paths via different Diameter agents may exist between the node doing the server selection and the targeted Diameter endpoint. The load information on the Diameter endpoint might be used for server selection and the load information on the agent might be used for selecting the next hop in the route to the Diameter endpoint.

The "scope" of load information defines what the load indication applies to. For example, load could apply to a whole Diameter node, or a node could report different load for different application. It might be possible to have a load value for a whole realm, or a group of nodes.

[I-D.roach-dime-overload-ctrl] has a very expressive concept of scope, which applies both to load and overload information. It defines the scopes of "Destination-Realm", "Application-ID", "Destination-Host", "Host", "Connection", "Session", and "Session-Group". Scopes can be combined.

[I-D.tschofenig-dime-dlba] does not have an explicit concept of scope. Load information describes the load of a server for all Diameter purposes.

[I-D.korhonen-dime-ovl] defines several scopes for overload information. However, load information applies to the a whole node.

One view is that the load level of a Diameter node will usually apply to the whole node. In this case, the working group should consider a single "whole node" scope for load information. Alternatively, a "per-connection" scope could simulate "whole node" scope without requiring the recipient to pay attention to whether multiple transport connections terminate at the same peer.

Other scopes might also be considered based on the analysis of the use cases identified for the use of load information.

7. Frequency of Sending Load Information

While it is true that a node always has a discrete load, a determination needs to be made as to the frequency with which load information is sent.

This interacts with the method for transporting load information -- piggy-backed versus a dedicated application -- discussed in Section 5.

With a piggy-backed approach the following alternatives exist:

1. Send load information in every message.

2. Send load information when it changes by some amount. For instance, only send a new load report when the load value has changed by some percentage.

3. Send load information every interval of time. With this approach, load information would be sent every some number of seconds.

With alternatives 2 and 3 there would need to be a mechanism for the sender of the load information to ensure that all consumers of the load information receive the periodic load information. This is more straightforward if the load information is sent only to peers. It becomes more difficult if the load information is sent to non adjacent nodes. This might require option one if the load mechanism supports sending of load information to non adjacent nodes.

If a dedicated application is used for transporting of load information then part of the application definition would need to define the frequency of sending load information. Options 2 and 3 in the above list would be the likely alternatives.

8. Load Information Semantics

Both [I-D.tschofenig-dime-dlba] and [I-D.korhonen-dime-ovl] define load level to be a range between zero and some maximum value, where zero means no load at all and the max value means fully loaded. The former uses a range of 0-10, while the later uses 0-100.

[I-D.roach-dime-overload-ctrl] treats load information as a strictly relative weighting factor. The weight is only meaningful when load-balancing across multiple destinations. That is, a maximum load value does not necessarily imply that the node is cannot handle more traffic. The load level scale is zero to 65535. That scale was chosen to match the resolution of the weight field from a DNS SRV record, [RFC2782]

9. Is Negotiation of Support Needed?

The working group should discuss whether a load conveyance mechanism requires negotiation or declaration of support. Several considerations apply to this discussion.

If load information is treated as a hint, it can be safely ignored by nodes that don't understand it. However, security considerations may apply if load information is accidentally leaked across a non-supporting node to a node that is not authorized to receive it.

If load information is conveyed using a dedicated Diameter application, the normal mechanisms for negotiation support for Diameter applications apply. However, the Diameter Capabilities Exchange [RFC6733] mechanism is inherently peer-to-peer. If there is a need to convey load information across a node that does not understand the mechanism, the standard Diameter mechanism would involve probing for support by sending load requests and watching for error answers with a result code of DIAMETER_APPLICATION_UNSUPPORTED. If the probe request also includes load information, there is again a potential for leaking load information to unauthorized parties.

If load information was treated in a strictly peer-to-peer fashion, there would be no need to probe to see if non-adjacent nodes support the mechanism. However, there would still be a need to control whether a non-supporting node would leak load information. Such a leak could be prevented if adjacent peers declared support, and never sent load information to a peer that did not declare support.

A peer-to-peer mechanism would also need a way to make sure that, if load information leaked across a non-supporting node, the receiving node would not mistakenly think the information came from the non-supporting node. This could be mitigated with a mechanism to declare support as in the previous paragraph, or with a mechanism to identify the origin of the load information. In the latter case, the receiving node would treat any load information as invalid if the origin of that information did not match the identity of the peer node.

10. Topology Scenarios

This section presents a number of Diameter topology scenarios, and discusses how load information might be used in each scenario. Nothing in this section should be construed to mean that a given scenario is in scope for this effort, or even a good idea. Some scenarios might be considered as not relevant in practice and subsequently discarded.

10.1. No Agent

Figure 1 shows a simple client-server scenario, where a client picks from a set of candidate servers available for a particular realm and application. The client selects the server for a given transaction using the load information received from each server.

```
  ------S1
 /
C
 \
  ------S2
```

Figure 1: Basic Client Server Scenario

Open Issue: Will a Diameter node include potential peers that it is not currently connected to as part of the candidate set? It is unlikely the client would have load information from peers that it is not currently connected to.

Note: The use of dynamic connections needs to be considered.

10.2. Single Agent

Figure 2 shows a client that sends requests to an agent. The agent selects the request destination from a set of candidate servers, using load information received from each server. The client does not need to receive load information, since it does not select between multiple agents.

```
       ------S1
      /
C----A
      \
       ------S2
```

Figure 2: Simple Agent Scenario

10.3. Multiple Agents

Figure 3 shows a client selecting between multiple agents, and each agent selecting from multiple servers. The client selects an agent based on the load information received from each agent. Each agent selects a server based on the load information received from its servers.

This scenario adds a complication that one set of servers may be more loaded than the other set. If, for example, S4 was the least loaded server, C would need to know to select agent A2 to reach S4. This might require C to receive load information from the servers as well as the agents. Alternatively, each agent might use the load of its servers as an input into calculating its own load, in effect aggregating upstream load.

Similarly, if C sends a host-routed request [I-D.ietf-dime-ovli], it needs to know which agent can deliver requests to the selected server. Without some special, potentially proprietary, knowledge of the topology upstream of A1 and A2, C would select the agent based on the normal peer selection procedures for the realm and application, and perhaps consider the load information from A1 and A2. If C sends a request to A1 that contains a Destination-Host AVP with a value of S4, A1 will not be able to deliver the request.

```
        -----S3
       /
  ---A1------S1
 /
C
 \
  ---A2------S2
       \
        ---- S4
```

Figure 3: Multiple Agents and Servers

10.4. Linked Agents

Figure 4 shows a scenario similar to that of Figure 3, except that the agents are linked, so that A1 can forward a request to A2, and vice-versa. Each agent could receive load information from the linked agent, as well as its connected servers.

This somewhat simplifies the complication from Figure 3, due to the fact that C does not necessarily need to choose a particular agent to reach a particular server. But it creates a similar question of how, for example, A1 might know that S4 was less loaded than S1 or S3. Additionally, it creates the opportunity for sub-optimal request paths. For example [C,A1,A2,S4] vs. [C,A2,S4].

A likely application for linked agents is when each agent prefers to route only to directly connected servers and only forwards requests to another agent under exceptional circumstances. For example, A1 might not forward requests to A2 unless both S1 and S3 are overloaded. In this case, A1 might use the load information from S1 and S3 to select between those, and only consider the load information from A2 (and other connected agents) if it needs to divert requests to different agents.

```
         -----S3
        /
   ---A1------S1
  /     |
C       |
  \     |
   ---A2------S2
        \
         ---- S4
```

Figure 4: Linked Agents

Figure 5 is a variant of Figure 4. In this case, C1 sends all traffic through A1 and C2 sends all traffic through A2. By default, A1 will load balance traffic between S1 and S3 and A2 will load balance traffic between S2 and S4.

Now, if S1 S3 are significantly more loaded than S2 S4, A1 may route some C1 traffic to A2. This is non optimal path but allows a better load balancing between the servers. To achieve this, A1 needs to receive some load info from A2 about S2/S4 load.

```
         -----S3
        /
C1----A1------S1
      |
      |
      |
C2----A2------S2
        \
         ---- S4
```

Figure 5: Linked Agents

10.5. Shared Server Pools

Figure 6 is similar to Figure 4, except that instead of a link between agents, each agent is linked to all servers. (The links to each set of servers should be interpreted as a link to each server. The links are not shown separately due to the limitations of ASCII art.)

In this scenario, each agent can select among all of the servers, based on the load information from the servers. The client need only be concerned with the load information of the agents.

```
  ---A1---S[1], S[2]...S[p]
 /      \ /
C        x
 \      / \
  ---A2---S[p+1], S[p+2] ...S[n]
```

Figure 6: Shared Server Pools

10.6. Agent Chains

The scenario in Figure 7 is similar to that of Figure 3, except that, instead of the client possibly needing to select an agent that can route requests to the least loaded server, in this case A1 and A2 need to make similar decisions when selecting between A3 or A4. As the former scenario, this could be mitigated if A3 and A4 aggregate upstream loads into the load information they report downstream.

```
  ---A1---A3----S[1], S[2]...S[p]
 /    | \ /
C     |  x
 \    | / \
  ---A2---A4----S[p+1], S[p+2] ...S[n]
```

Figure 7: Agent Chains

10.7. Fully Meshed Layers

Figure 8 extends the scenario in Figure 6 by adding an extra layer of agents. But since each layer of nodes can reach any node in the next layer, each node only needs to consider the load of its next-hop peer.

```
  ---A1---A3---S[1], S[2]...S[p]
 /    | \ / |\ /
C     |  x  | x
 \    | / \ |/ \
  ---A2---A4---S[p+1], S[p+2] ...S[n]
```

Figure 8: Full Mesh

10.8. Partitions

A Diameter network with multiple is said to be "partitioned" when only a subset of available servers can server a particular realm-routed request. For example, one group of servers may handle users whose names start with "A" through "M", and another group may handle "N" through "Z".

In such a partitioned network, nodes cannot load-balance requests across partitions, since not all servers can handle the request. A client, or an intermediate agent, may still be able to load-balance between servers inside a partition.

10.9. Active-Standby Nodes

The previous scenarios assume that traffic can be load balanced among all peers that are eligible to handle a request. That is, the peers operate in an "active-active" configuration. In an "active-standby" configuration, traffic would be load-balanced among active peers. Requests would only be sent to peers in a "standby" state if the active peers became unavailable. For example, requests might be diverted to a stand-by peer if one or more active peers becomes overloaded.

10.10. Addition and removal of Nodes

When a Diameter node is added, the new node will start by advertising its load. Downstream nodes will need to factor the new load information into load balancing decisions. The downstream nodes should attempt to ensure a smooth increase of the traffic to the new node, avoiding an immediate spike of traffic to the new node. It should be determined if this use case is in the scope of the load control mechanism.

When removing a node in a controlled way (e.g. for maintenance purpose, so outside a failure case), it might be appropriate to progressively reduce the traffic to this node by routing traffic to other nodes. Simple load information (load percentage) would be not sufficient. It should be determined if this use case is in the scope of the load control mechanism.

11. Security Considerations

Load information may be sensitive information in some cases. Depending on the mechanism. an unauthorized recipient might be able to infer the topology of a Diameter network from load information. Load information might be useful in identifying targets for Denial of Service (DoS) attacks, where a node known to be already heavily loaded might be a tempting target. Load information might also be useful as feedback about the success of an ongoing DoS attack.

Any load information conveyance mechanism will need to allow operators to avoid sending load information to nodes that are not authorized to receive it. Since Diameter currently only offers authentication of nodes at the transport level, any solution that sends load information to non-peer nodes might require a transitive-trust model.

12. IANA Considerations

This document makes no requests of IANA.

13. References

13.1. Normative References

[I-D.ietf-dime-ovli]
Korhonen, J., Donovan, S., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", draft-ietf-dime-ovli-03 (work in progress), July 2014.

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

[RFC7068] McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", RFC 7068, November 2013.

13.2. Informative References

[I-D.korhonen-dime-ovl]
Korhonen, J. and H. Tschofenig, "The Diameter Overload Control Application (DOCA)", draft-korhonen-dime-ovl-01 (work in progress), February 2013.

[I-D.roach-dime-overload-ctrl]
Roach, A. and E. McMurry, "A Mechanism for Diameter Overload Control", draft-roach-dime-overload-ctrl-03 (work in progress), May 2013.

[I-D.tschofenig-dime-dlba]
Tschofenig, H., "The Diameter Load Balancing Application (DLBA)", draft-tschofenig-dime-dlba-00 (work in progress), July 2013.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

Authors' Addresses

Ben Campbell
Oracle
7460 Warren Parkway # 300
Frisco, Texas 75034
USA

Email: ben@nostrum.com

Steve Donovan (editor)
Oracle
7460 Warren Parkway # 300
Frisco, Texas 75034
United States

Email: srdonovan@usdonovans.com

Jean-Jacques Trottin
Alcatel-Lucent
Route de Villejust
91620 Nozay
France

Email: jean-jacques.trottin@alcatel-lucent.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: May 11, 2015
S. Donovan
Oracle
November 7, 2014

Diameter Agent Overload
draft-donovan-dime-agent-overload-03.txt

Abstract

This specification documents an extension to the Diameter Overload Control (DOC) base solution. The extension addresses the handling of occurrances of overload of a Diameter agent.

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

Copyright Notice

Table of Contents

1. Introduction

This document defines the behavior of Diameter nodes when Diameter agents enter an overload condition and send an overload report requesting a reduction of traffic.

The base Diameter overload specification [I-D.ietf-dime-ovli] addresses the handling of overload when a Diameter endpoint (a

Diameter Client or Diameter Server as defined in [RFC6733]) becomes overloaded.

In the base specification, the goal is to handle abatement of the overload occurrance as close to the source of the Diameter traffic as is feasible. When possible this is done at the originator of the traffic, generally referred to as a Diameter Client. A Diameter Agent might also handle the overload mitigation. For instance, a Diameter Agent might handle Diameter overload mitigation when it knows that a Diameter Client does not support the DOIC extension.

This document extends the base Diameter endpoint overload specification to address the case when Diameter Agents become overloaded. Just as is the case with other Diameter nodes -- Diameter Clients and Diameter Servers -- surges in Diameter traffic can cause a Diameter Agent to be asked to handle more Diameter traffic than it was configured to handle. For a more detailed discussion of what can cause the overload of Diameter nodes, refer to the Diameter Overload Requirements [RFC7068].

This document defines a new overload report type to communicate occurrances of agent overlaod. This report type works for the "Loss" overload mitigation algorithm defined in [I-D.ietf-dime-ovli] and is expected to work for other overload abatement algorithms defined in extensions to the DOIC solution.

The handling of endpoint overload and agent overload is very similar. The primary differences are the following:

o Endpoint overload is handled as close to the originator of the traffic as possible.

o Agent overload is handled by the previous hop Diameter Node.

o Endpoint overload mitigation deals with traffic targeted for a single Diameter application. As such, it is assumed that an overload report impacts just the application implied by the message carrying the overload report.

o Agent overload deals with all traffic targeted for an agent, independent of the application. As such, a single agent overload report can impact multiple applications.

Editor’s Note: Open Issue - Does a peer report apply to the implicitly communicated application-id in the same way as host and realm reports do or does it apply to all applications handled by the peer? Do we need the ability for to support both cases?

Open Issue - To support the ability of an agent to select a different abatement algorithm than endpoints, we probably need to extend the OC-Supported-Features AVP to include an OC-Abatement-Algorithm AVP. This is currently shown to be in the OC-OLR AVP but needs to be moved as this information is needed prior to receiving the OC-OLR. It probably needs to be changed to OC-Peer-Abatement-Algorithm.

2. Terminology and Abbreviations

Editors note - These definitions need to be made consistent with the base Diameter overload specification defined in [I-D.ietf-dime-ovli].

Diameter Node

A RFC6733 Diameter Client, an RFC6733 Diameter Server, and RFC6733 agent.

Diameter Endpoint

An RFC6733 Diameter Client and RFC6733 Server.

Reporting Node

A DOIC Node that sends and overload report in Diameter answer message.

Reacting Node

A DOIC Node that receives and acts on a Diameter overload report.

DIOC Node

A Diameter Node that supports the DOIC solution defined in [I-D.ietf-dime-ovli].

3. Peer Report Use Cases

This section outlines representative use cases for the peer report.

There are two primary classes of use cases, those involving the overload of agents and those involving overload of Diameter endpoints (Diameter Clients and Diameter Servers).

3.1. Diameter Agent Overload Use Cases

The agent overload extension must support following use cases.

3.1.1. Single Agent

This use case is illustrated in Figure 1. In this case, the client sends all traffic through the single agent. If there is a failure in the agent then the client is unable to send Diameter traffic toward the server.

```
                          +-+    +-+    +-+
                          |c|----|a|----|s|
                          +-+    +-+    +-+
```

Figure 1

A more likely case for the use of agents is illustrated in Figure 2. In this case, there are multiple servers behind the single agent. The client sends all traffic through the agent and the agent determines how to distribute the traffic to the servers based on local routing and load distribution policy.

```
                                        +-+
                                      --|s|
                          +-+    +-+ /  +-+
                          |c|----|a|-   ...
                          +-+    +-+ \  +-+
                                      --|s|
                                        +-+
```

Figure 2

In both of these cases, the occurrence of overload in the single agent must by handled by the client in a similar fashion as if the client were handling the overload of a directly connected server. When the agent becomes overloaded it will insert an overload report in answer messages flowing to the client. This overload report will contain a requested reduction in the amount of traffic sent to the agent. The client will apply overload abatement behavior as defined in the base Diameter overload specification [I-D.ietf-dime-ovli] or the extension draft that defines the indicated overoad abatement algorithm. This will result in the abated traffic that would have been sent to the agent being dropped, as there is no alternative route, with the appropriate indication given to the service request that resulted in the need for the Diameter transaction.

Editor’s note: Need to address case where the agent requests a different abatement algorithm than requested by a host or realm reporting node.

3.1.2. Redundant Agents

Figure 3 and Figure 4 illustrate a second, and more likely,type of deployment scenario involving agents. In both of these cases, the client has Diameter connections to two agents.

Figure 3 illustrates a client that has a primary connection to one of the agents (agent a1) and a secondary connection to the other agent (agent a2). In this scenario, under normal circumstances, the client will use the primary connection for all traffic. The secondary connection is used when there is a failure scenario of some sort.

```
       +--+   +-+
     --|a1|---|s|
+-+ /  +--+\ /+-+
|c|-        x
+-+ .  +--+/ \+-+
     ..|a2|---|s|
       +--+   +-+
```

Figure 3

The second case, in Figure 4, illustrates the case where the connections to the agents are both actively used. In this case, the client will have local distribution policy to determine the percentage of the traffic sent through each client.

```
       +--+   +-+
     --|a1|---|s|
+-+ /  +--+\ /+-+
|c|-        x
+-+ \  +--+/ \+-+
     --|a2|---|s|
       +--+   +-+
```

Figure 4

In the case where one of the agents in the above scenarios become overloaded, the client should reduce the amount of traffic sent to the overloaded agent by the amount requested. This traffic should instead be routed through the non-overloaded agent. For example, assume that the overloaded agent requests a reduction of 10 percent. The client should send 10 percent of the traffic that would have been routed to the overloaded agent through the non-overloaded agent.

When the client has an active and a standby connection to the two agents then an alternative strategy for responding to an overload report from an agent is to change to standby connection to active and route all traffic through the new active connection.

In the case where both agents are reporting overload, the client may need to start decreasing the total traffic sent to the agents. This would be done in a similar fashion as discussed in section 3.1. The amount of traffic depends on the combined reduction requested by the two agents.

3.1.3. Agent Chains

There are also deployment scenarios where there can be multiple Diameter Agents between Diameter Clients and Diameter Servers. Examples of this type of deployment include when there are edge agents between Diameter networks. Another example of this type of deployment is when there are multiple sets of servers, each supporting a subset of the Diameter traffic.

Figure 5 illustrates one such network deployment case. Note that while this figure shows a maximum of two agents being involved in a Diameter transaction, it is possible that more than two agents could be in the path of a transaction.

```
               +---+     +---+   +-+
             --|a11|-----|a21|---|s|
        +-+ /  +---+ \ / +---+\ /+-+
        |c|-          x        x
        +-+ \  +---+ / \ +---+/ \+-+
             --|a12|-----|a22|---|s|
               +---+     +---+   +-+
```

Figure 5

Handling of overload of one or both of agents a11 or a12 in this case is equivalent to that discussed in section 2.2.

Overload of agents a21 and a22 must be handled by the previous hop agents. As such, agents a11 and a12 must handle the overload mitigation logic when receiving an agent overload report from agents a21 and a22.

Editor’s note: Probably need to elaborate the reasoning behind the need for the agent overload report being handled by the previous hop agent.

The handling of the overload reports is similar to that discussed in section 2.2. If the overload can be addressed using diversion then this approach should be taken.

If both of the agents have requested a reduction in traffic then the previous hop agent must start throttling the appropriate percentage of transactions. When throttling requests, the agent must use the same error responses as defined in the base DOIC specification [I-D.ietf-dime-ovli].

3.2. Diameter Endpoint Use Cases

This section outlines use cases for the peer report feature involving Diameter Clients and Diameter Servers.

3.2.1. Hop-by-hop Abatement Algorithms

It is envisioned that abatement algorithms will be defined that will support the option for Diameter Endpoints to send peer reports. For instance, it is envisioned that one usage scenario for the rate algorithm, which is being worked on by the DIME working group as this is written, will involve abatement being done on a hop-by-hop basis.

This rate deployment scenario would involve Diameter Endpoints generating peer reports and selecting the rate algorithm for abatement of overload conditions.

4. Interaction Between Host/Realm and Peer Overload Reports

It is possible that both an agent and a server in the path of a transaction are overloaded at the same time. When this occurs, Diameter entities will need to handle both overload reports. When this occurs the reacting node should first handle the throttling of the overloaded host or realm. Any messages that survive throttling due to host or realm reports should then go through abatement for the peer overload report.

> Editor’s note: Do we need to prevent double throttling of requests or is that a local implementation consideration?

5. Peer Report Behavior

This section defines the normative behavior associated with the Peer Report extension to the DOIC solution.

5.1. Capability Announcement

Editor's Note: Issue - how does an agent indicate the selected abatement algorithm? It cannot use the OC-Feature-Vector in the OC-Supported-Features AVP as that applies to host and realm report types. Need a new AVP in the OC-Supported-Features AVP.

When sending a Diameter request a DOIC node that supports the Peer Report feature MUST include an OC-Supported-Features AVP with an OC-Feature-Vector AVP with the OLR_PEER_REPORT bit set.

> The sender of a request can be a Diameter Client or Diameter Server that originates the Diamter request or a Diameter Agent that relays the request.
>
> Support for the peer report feature does not impact the logic for setting of other feature bits in the OC-Feature-Vector AVP.

When sending a request a DOIC node that supports the Peer Report feature MUST include an OC-SourceID AVP in the OC-Supported-Features AVP with its own DiameterID.

> This allows the next DOIC node in the path of the request to determine if the indication of support came from a Diameter peer or if the request traversed a node that does not support the peer feature.

When receiving a request a DOIC node that supports the Peer Report feature MUST update transaction state with an indication of whether or not the peer from which the request was received supports the Peer Report feature.

> The transaction state is used when the DOIC node is acting as a peer report reporting node and needs to insert OC-OLR reports of type peer into answer messages. The OLR should only be included in answer messages being sent to peers that support the peer report feature.

The following are indications that the peer does not support the Peer Reports feature:

> The request does not contain an OC-Supported-Features AVP.
>
> The received request contains an OC-Supported-Features AVP with no a OC-Feature-Vector.
>
> The received request contains an OC-Supported-Features AVP with a OC-Feature-Vector with the OLR_PEER_REPORT feature bit cleared.

The received request contains an OC-Supported-Features AVP with a OC-Feature-Vector with the OLR_PEER_REPORT feature bit set but with an OC-SourceID AVP with a DiameterID that does not match the DiameterID of the peer from which the request was received.

The peer supports the Peer Reports feature if the received request contains an OC-Supported-Features AVP with the OC-Feature-Vector with the OLR_PEER_REPORT feature bit set and with an OC-SourceID AVP with a Diameter ID that matches the DiameterID of the peer from which the request was received.

When receiving a request a DOIC node that supports the Peer Report feature MUST remove any received OC-SourceID AVP from the OC-Supported-Features AVP. This is done to prevent the OC-SourceID AVP from being included in a relayed message through a node that supports the Peer Report feature.

Editor’s Note: Need to add behavior for handling of answer messages to define how the OC-Supported-Features AVP that will be included in a relayed answer message is constructed. This includes logic on whether or not the peer report feature bit is set and whether or not the OC-Peer-Algo AVP is included in the OC-Supported-Features AVP.

5.2. Peer Report Overload Report Handling

This section defines the behavior for the handling of overload reports of type peer.

5.2.1. Overload Control State

This section describes the Overload Control State (OCS) that might be maintained by both the peer report reporting node and the peer report reacting node.

5.2.1.1. Reporting Node Peer Report OCS

A DOIC Node that supports the Peer Report feature SHOULD maintain Reporting Node Peer Report OCS. This is used to record overload events and build overload reports at the reporting node.

If different abatement specific contents are sent to each peer then the reporting node MUST maintain a separate peer node peer report OCS entry per peer to which a peer overload report is sent.

The rate overload abatement algorithm allows for different rates to be sent to each peer.

The Reporting Node Peer Report OCS entry MAY include the following information (the actual information stored is an implementation decision):

- o Sequence number
- o Validity Duration
- o Expiration Time
- o Abatement Algorithm
- o Algorithm specific input data (for example, the Reduction Percentage for the Loss Abatement Algorithm)

5.2.1.2. Reacting Node Peer Report OCS

A DOIC node that supports the Peer Report feature SHOULD maintain Reacting Node Peer Report OCS for each peer with which it communicates. This is used to record overload reports received from peer nodes.

A Reacting Node Peer Report OCS entry is identified by the DiameterID of the peer as communicated during the RFC6733 defined Capability Exchange procedure

The Reacting Node Peer Report OCS entry MAY include the following information (the actual information stored is an implementation decision):

- o Sequence number
- o Expiration Time
- o Abatement Algorithm
- o Algorithm specific input data (for example, the Reduction Percentage for the Loss Abatement Algorithm)

5.2.2. Reporting Node Maintenace of Peer Report OCS

A reporting node SHOULD create a new Reporting Node Peer Report OCS entry Section 5.2.1.1 in an overload condition and sending a peer overload report to a peer for the first time.

> If the reporting node knows that there are no reacting nodes supporting the Peer Report feature then the reporting node can choose to not create OCS entries.

All rules for managing the reporting node OCS enteries defined in [DOIC] apply to the peer report.

5.2.3. Reacting Node Maintenace of Peer Report OCS

When a reacting node receives an OC-OLR AVP with an a report type of peer it MUST determine if the report was generated by the Diameter peer from which the report was received.

If the DiameterID in the SourceID contained in the OLR matches the DiameterID of the peer from which the request was received then the report was received from a Diameter peer.

If a reacting node receives an OC-OLR AVP of type peer and the OC-SourceID does not match the ID of the Diameter peer from which the request was received then the reacting node MUST strip the OC-OLR AVP from the message and not use it to update reacting node peer report OCS entries.

If the Peer Report OLR was received from a Diameter peer then the reacting node MUST determine if it is for an existing or new overload condition.

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

For a peer report-type this means the DiameterID received in the SourceID AVP matches the DiameterID of an existing peer report OLR.

If the OLR is for an existing overload condition then it 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 the 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 the reacting node MUST silently ignore the received OLR. The matching OCS MUST NOT be updated in this case.

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

> Editor’s note: The above four paragraphs are copied form the DOIC specification. Is it possible to include this behavior by

refrence or do we need to include all of these statements in this specification as well.

For a peer report this means it creates an OCS entry with an DiameterID from the SourceID AVP in the received OC-OLR AVP.

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

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").

The reacting node sets the abatement algorithm based on the OC-Peer-Algo AVP in the received OC-Supported-Features AVP.

5.2.4. Peer Report Reporting Node Behavior

When there is an existing peer report reporting node OCS entry, the reporting node MUST include an OC-OLR AVP with a report type of peer using the contents of the peer report reporting node OCS entry in all answer messages sent by the reporting node to peers that support the peer report feature.

The reporting node determines if a peer supports the peer report feature based on the indication recorded in the reporting nodes transaction state.

The reporting node MUST include its DiameterID in the OC-SourceID AVP in the OC-OLR AVP. This is used by DOIC nodes that support the peer report feature to determine if the report was received from a Diameter peer.

The reporting agent must follow all other overload reporting node behaviors outlined in the DOIC specification. This includes sending a report with a reduction percentage of zero when the need for a reduction has ended. It also includes sending a new overload report, with a new sequence number, to refresh the abatement duration.

5.2.5. Peer Report Reacting Node Behavior

A reacting node supporting this extension MUST support the receipt of multiple overload reports in a single message. The message might inlude a host overload report, a realm overload report and a peer overload report.

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

If the request matches and active OCS then the reacting node MUST apply abatement treatment on the request. The abatement treatment applied depends on the abatement algorithm stored in the OCS.

For peer overload reports, the preferred abatement treatment is diversion. As such, the reacting node SHOULD attempt to divert requests identified as needing abatement to other peers.

If a host-routed request, as defined in the DOIC specification, is selected for abatement and the request must be routed to the DOIC node that generated the peer overload report -- meaning that the request is a host-routed request as defined in the DOIC specification -- then the reacting node MUST throttle the request.

> This would result from an overloaded Diameter endpoint (Diameter Server or Diameter Client) sending a peer overload report and the request contains a Destination-Host AVP with a DiameterID that matches the DiameterID in the SourceID AVP received in the peer overload report.

If there is not sufficient capacity to divert abated traffic then the reacting node MUST throttle the necessary requests to fit within the available capacity of the peers able to handle the requests.

If the 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 the DOIC specification.

In the case that the OCS entry validity duration expires or has a validity duration of zero ("0"), meaning that it the reporting node has explicitly signaled the end of the overload condition then abatement associated with the overload abatement MUST be ended in a controlled fashion.

6. Peer Report AVPs

6.1. OC-Supported-Features AVP

This extension adds a new feature to the OC-Feature-Vector AVP. This feature indication shows support for handling of peer overload reports. Peer overload reports are used by agents to indicate the need for overload abatement handling by the agents peer.

A supporting node must also include the OC-SourceID AVP in the OC-Supported-Features capability AVP.

This AVP contains the Diameter Identity of the node that supports the OLR_PEER_REPORT feature. This AVP is used to determine if support for the peer overload report is in an adjectent node. The value of this AVP should be the same Diameter identity used as part of the CER/CEA base Diameter capabilities exchange.

```
 OC-Supported-Features ::= < AVP Header: TBD1 >
                           [ OC-Feature-Vector ]
                           [ OC-SourceID ]
                           [ OC-Peer-Algo]
                         * [ AVP ]
```

6.1.1. OC-Feature-Vector

The peer report feature defines a new feature bit is added for the OC-Feature-Vector AVP.

OLR_PEER_REPORT (0x0000000000000010)

When this flag is set by a DOIC node it indicates that the DOIC node supports the peer overload report type.

6.1.2. OC-Peer-Algo

The OC-Peer-Algo AVP (AVP code TBD6) 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.

Feature bits defined for the OC-Feature-Vector AVP and associated with overload abatement algorithms are reused in for this AVP. This include the following value defined in the DOIC specification.

Editor’s node: This is to avoid the need for an additional IANA registry.

6.2. OC-OLR AVP

This extension makes no changes to the SequenceNumber or ValidityDuration AVPs in the OC-OLR AVP. These AVPs are also be used in peer overload reports.

The peer report feature extends the base Diameter overload specification by defining a new overload report type of "peer". See section [4.5] in [I-D.ietf-dime-ovli] for a description of the overload report type AVP.

The overload report must also include the Diameter identity of the agent that generated the report. This is necessary to handle the case where there is a non supporting agent between the reporting node and the reacting node. Without the indication of the agent that generated the overload request, the reacting node could erroneously assume that the report applied to the non supporting node. This could, in turn, result in unnecessary traffic being either redistributed or throttled.

The OC-SourceID AVP is used in the OC-OLR AVP to carry this DiameterID.

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

6.2.1. OC-Report-Type AVP

The following new report type is defined for the OC-Report-Type AVP.

2 Peer. The overload treatment should apply to all requests bound for the peer identified in the overload report. If the peer identified in the overload report is not a peer to the reacting endpoint then the overload report should be stripped and not acted upon.

This extension uses the OC-SourceID AVP for this purpose.

6.3. OC-SourceID

The SourceID AVP (AVP code TBD) is of type DiameterIdentity and is inserted by the DOIC node that either indicates support for this feature (in the OC-Supported-Features AVP) or that generates an OC-OLR AVP with a report type of peer.

It contains the Diameter Identity of the inserting node. This is used by other DOIC nodes to determine if the a peer indicated indicated support this feature or inserted the peer report

6.4. Attribute Value Pair flag rules

```
                                                         +---------+
                                                         |AVP flag |
                                                         |rules    |
                                                         +----+----+
                        AVP   Section                    |    |MUST|
    Attribute Name      Code  Defined Value Type         |MUST| NOT|
   +-----------------------------------------------------+----+----+
   |OC-SourceID         TBD1    x.x   Unsigned64         |    | V  |
   |OC-Peer-Algo        TBD1    x.x   Unsigned64         |    | V  |
   +-----------------------------------------------------+----+----+
```

7. IANA Considerations

Editors note: This section will be completed once the base overload document has finished the definition of extension IANA requirements.

8. Security Considerations

Agent overload is an extension to the based Diameter overload mechanism. As such, all of the security considerations outlined in [I-D.ietf-dime-ovli] apply to the agent overload scenarios.

It is possible that the malicious insertion of an agent overload report could have a bigger impact on a Diameter network as agents can be concentration points in a Diameter network. Where an end-point report would impact the traffic sent to a single Diameter server, for example, a peer report could throttle all traffic to the Diameter network.

This impact is amplified in an agent that sits at the edge of a Diameter network that serves as the entry point from all other Diameter networks.

9. Acknowledgements

Adam Roach and Eric McMurry for the work done in defining a comprehensive Diameter overload solution in draft-roach-dime-overload-ctrl-03.txt.

Ben Campbell for his insights and review of early versions of this document.

10. Normative References

[I-D.ietf-dime-ovli]
Korhonen, J., "Diameter Overload Indication Conveyance", October 2013.

[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.

[RFC7068] McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", RFC 7068, November 2013.

Author’s Address

Steve Donovan
Oracle
7460 Warren Parkway, Suite 300
Frisco, Texas 75034
United States

Email: srdonovan@usdonovans.com

Diameter Maintenance and Extensions (DIME) S. Donovan
Internet-Draft Oracle
Intended status: Standards Track E. Noel
Expires: May 11, 2015 AT&T Labs
November 7, 2014

Diameter Overload Rate Control
draft-donovan-dime-doc-rate-control-02.txt

Abstract

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) base solution. This extension adds a new overload control abatement algorithm. This abatement algorithm allows for a DOIC reporting node to specify a maximum rate at which a DOIC reacting node sends Diameter requests sent to the DOIC reporting node.

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

Copyright Notice

Table of Contents

1. Introduction

This document defines a new Diameter overload control algorithm.

The base Diameter overload specification [I-D.ietf-dime-ovli] defines the loss algorithm as the default Diameter overload abatement algorithm. The loss algorithm allows a reporting node to instruct a reacting node to reduce the amount of traffic sent to the reporting node by throttling a percentage of requests sent to the server. While this can effectively decrease the load handled by the server, it does not directly address cases where the rate of arrival of service requests increase quickly. If the service requests that result in Diameter transactions increases quickly then the loss algorithm can be slow to protect the stability of reporting nodes.

Consider the case where a reacting node is handling 100 service requests per second, where each of these service requests results in one Diameter transaction being sent to a reacting node. If the reacting node is approaching an overload state, or is already in an overload state, it will send a Diameter overload report requesting a percentage reduction in traffic sent. Assume for this discussion that the reporting node requests a 10% reduction. The reacting node will then throttle ten Diameter transactions a second, sending the remaining 90 transactions per second to the reacting node.

Now assume that the reacting node’s service requests spikes to 1000 requests per second. The reacting node will continue to honor the reporting nodes request to throttle 10% of the traffic. This results, in this example, in the reacting node sending 900 Diameter transactions per second, throttling the remaining 100 transactions per second. This spike in traffic is significantly higher than the reporting node is expecting to handle and can result in negative impacts to the stability of the reporting node.

The reporting node can, and likely would, send another overload report requesting that the reacting node throttle 91% of requests to get back to the desired 90 transactions per second. However, once the spike has abated and the reacting node handled service requests returns to 100 per second, this will result in just 9 transactions per second being sent to the reporting node, requiring a new overload report setting the reduction percentage back to 10%.

One of the benefits of a rate based algorithm is that it better handles spikes in traffic. Instead of sending a request to throttle a percentage of the traffic, the rate approach allows the reporting node to specify the maximum number of Diameter requests per second that can be sent to the reporting node. For instance, in this example, the reporting node could send a rate based request specifying the maximum transactions per second to be 90. The reacting noce will send the 90 regardless of whether it is receiving 100 or 1000 service requests per second.

This document extends the base DOIC solution [I-D.ietf-dime-ovli] to add support for the rate based overload abatement algorithm.

This document draws heavily on work in the RIA SIP Overload Control working group. The definitions of the rate abatement algorithmm is copied almost verbatim from the SOC document [I-D.SOC-overload-rate-control], with changes focused on making the wording consistent with the DOIC solution and the Diameter protocol.

> Editor’s Note: Need to verify that the latest text from the SOC document is currently being used.

2. Terminology and Abbreviations

Diameter Node

> A RFC6733 Diameter Client, an RFC6733 Diameter Server, and RFC6733 Diameter Agent.

Diameter Endpoint

> An RFC6733 Diameter Client and RFC6733 Diameter Server.

DOIC Node

> A Diameter Node that supports the DOIC solution defined in [I-D.ietf-dime-ovli].

Reporting Node

> A DOIC Node that sends a DOIC overload report.

Reacting Node

> A DOIC Node that receives and acts on a DOIC overload report.

3. Interaction with DOIC report types

As of the publication of this specification there are two DOIC report types defined with the specification of a third in progress:

1. Host - Overload of a specific Diameter Application at a specific Diameter Node as defined in [I-D.ietf-dime-ovli].

2. Realm - Overlaod of a specific Diameter Application at a specific Diameter Realm as defined in [I-D.ietf-dime-ovli].

3. Peer - Overload of a specific Diameter peer as defined in [I-D.donovan-agent-overload].

The rate algorithm MAY be selected by reporting nodes for any of these report types.

It is expected that all report types defined in the future will indicate whether or not the rate algorithm can be used with that report type.

4. Capability Announcement

Editors Note: This section depends upon the completion of the base Diameter Overload specification. As such, it cannot be complete until the data model and extension mechanism are finalized in the base DOC specification. Details for any new AVPs or modifications to existing AVPs will be finalized in a future version of the draft after the base DOC specification has stabilized.

This extension defines the rate abatement algorithm (referred to as rate in this document) feature. Support for the rate feature will be reflected by use of a new value, as defined in Section 6.1.1, in the OC-Feature-Vector AVP per the rules defined in [I-D.ietf-dime-ovli].

Note that Diameter nodes that support the rate feature will, by definition, support both the loss and rate based abatement algorithms. DOIC reacting nodes SHOULD indicate support for both the loss and rate algorithms in the OC-Feature-Vector AVP.

> There may be local policy reasons that cause a DOIC node that supports the rate to not include it in the OC-Feature-Vector. All reacting nodes, however, must continue to include loss in the OC-Feature-Vector in order to remain compliant with [I-D.ietf-dime-ovli].

A reporting nodes MUST select either the rate or the loss algorithm when receiving a request that contains an OC-Supported-Features AVP.

A reporting node MAY select one abatement algorithm to apply to host and realm reports and a different algorithm to apply to peer reports.

For host or realm reports the selected algorithm MUST be reflected in the OC-Feature-Vector AVP sent as part of the OC-Selected-Features AVP included in answer messages for transaction where the request contained an OC-Supported-Features AVP. This is per the precedures defined in [I-D.ietf-dime-ovli].

For peer reports the selected algorithm MUST be reflected in the OC-Peer-Abatement-Algorithm AVP sent as part of the OC-Supported-Features AVP included answer messages for transaction where the request contained an OC-Supported-Features AVP. This is per the procedures defined in [I-D.donovan-agent-overload].

> Editor’s Node: The peer report specification is still under development and, as such, the above paragraph is subject to change.

5. Overload Report Handling

This section describes any changes to the behavior defined in [I-D.ietf-dime-ovli] for handling of overload reports when the rate overload abatement algorithm is used.

5.1. Reporting Node Overload Control State

A reporting node that uses the rate abatement algorithm SHOULD maintain reporting node OCS for each reacting node to which it sends a rate OLR.

> This is different from the behavior defines in [DOIC] where there is a single loss percentage sent to all reacting nodes.

A reporting node SHOULD maintain OCS entries when using the rate abatement algorithm per supported Diameter application, per targeted reacting node and per report-type.

A rate OCS entry is identified by the tuple of Application-Id, report-type and peer-id of the target of the rate OLR.

A reporting node that supports the rate abatement algorithm MUST be able to include rate as the selected abatement algorithm in the reporting node OCS.

A reporting node that supports the rate abatement algorithm MUST be able to include the specified rate in the abatement algoritm specific portion of the reporting node OCS.

All other elements for the OCS defined in [I-D.ietf-dime-ovli] and [I-D.donovan-agent-overload] also apply to the reporting nodes OCS when using the rate abatement algorithm.

5.2. Reacting Node Overload Control State

A reacting node that supports the rate abatement algorithm MUST be able to include rate as the selected abatement algorithm in the reacting node OCS.

A reacting node that supports the rate abatement algorithm MUST be able to include the rate specified in the OC-Rate AVP included in the OC-OLR AVP as an element of the abatement algoritm specific portion of reacting node OCS entries.

All other elements for the OCS defined in [I-D.ietf-dime-ovli] and [I-D.donovan-agent-overload] also apply to the reporting nodes OCS when using the rate abatement algorithm.

5.3. Reporting Node Maintenance of Overload Control State

A reporting node that has selected the rate overload abatement algorithm and enters an overload condition MUST indicate rate as the abatement algorithm in the resulting reporting node OCS entries.

A reporting node that has selected the rate abatement algorithm and enters an overload condition MUST indicate the selected rate in the resulting reporting node OCS entries.

When responding to a request that contained an OC-Supporting-Features AVP with an OC-Feature-Vector AVP indicating support for the rate feature, a reporting node MUST ensure that a reporting node OCS entry exists for the target of the overload report. The target is defined as follows:

- o For Host reports the target is the DiameterID contained in the Origin-Host AVP received in the request.

- o For Realm reports the target is the DiameterID contained in the Origin-Realm AVP received in the request.

- o For Peer reports the target is the Diameter ID of the Diameter Peer from which the request was received.

5.4. Reacting Node Maintenance of Overload Control State

A reacting node receiving an overload report for the rate abatement algorithm MUST save the rate received in the OC-Rate AVP contained in the OC-OLR AVP in the reacting node OCS entry.

5.5. Reporting Node Behavior for Rate Abatement Algorithm

When in an overload condition with rate selected as the overload abatement algorithm and when handling a request that contained an OC-Supported-Features AVP that indicated support for the rate featre, a reporting node SHOULD include an OC-OLR AVP for the rate algorithm using the parameters stored in the reporting node OCS for the target of the overload report.

Editor’s Note: The above is a pretty complicated way of saying that the reporting node should include an OC-OLR in the appropriate answer messages. The basic requirement isn’t rate feature specific but rather that in all cases the reporting node generates an OC-OLR according to the parameters of the appropriate OCS entry. This wording probably can be improved based on the generic behavior definition.

When sending an overload report for the Rate algorithm, the OC-Rate AVP is included and the OC-Reduction-Percentage AVP is not included.

5.6. Reacting Node Behavior for Rate Abatement Algorithm

When determining if abatement treatment should be applied to a request being sent to a reporting node that has selected the rate overload abatement algorithm, the reacting node MUST use the the algorithm detailed in Section 6 to make the determination.

Once a determination is made by the reacting node that an individual Diameter request is to be subjected to abatement treatment then the procedures for throttling and diversion defined in [I-D.ietf-dime-ovli] and [I-D.donovan-agent-overload] apply.

6. Rate Abatement Algorithm AVPs

Editors Note: This section depends upon the completion of the base DOIC specification. As such, it cannot be complete until the data model and extension mechanism are finalized. Details for any new AVPs or modifications to existing AVPs will be finalized in a future version of the draft after the base DOC specification has stabilized.

6.1. OC-Supported-Features AVP

The rate algorithm does not add any AVPs to the OC-Supported-Features AVP.

The rate algorithm does add a new feature bit to be carried in the OC-Feature-Vector AVP.

6.1.1. OC-Feature-Vector AVP

This extension adds the following capabilities to the OC-Feature-Vector AVP.

OLR_RATE_ALGORITHM (0x0000000000000004)

When this flag is set by the overload control endpoint it indicates that the DOIC Node supports the rate overload control algorithm.

6.2. OC-OLR AVP

This extension defines the OC-Rate AVP to be an optional part of the OC-OLR AVP.

```
OC-OLR ::= < AVP Header: TBD2 >
           < OC-Sequence-Number >
           < OC-Report-Type >
           [ OC-Reduction-Percentage ]
           [ OC-Validity-Duration ]
           [ OC-Source-ID ]
           [ OC-Abatement-Algorithm ]
           [ OC-Rate ]
         * [ AVP ]
```

This extension makes no changes to the other AVPs that are part of the OC-OLR AVP.

This extension does not define new overload report types. The existing report types of host and realm defined in [I-D.ietf-dime-ovli] apply to the rate control algorithm. The peer report time defined in [I-D.donovan-agent-overload] also applies to the rate control algorithm.

6.2.1. OC-Rate AVP

The OC-Rate AVP (AVP code TBD8) is type of Unsigned32 and describes the maximum rate that that the sender is requested to send traffic. This is specified in terms of requests per second.

Editor’s note: Do we need to specify a maximum value?

A value of zero indicates that no traffic is to be sent.

6.3. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules MUST	AVP flag rules MUST NOT
OC-Rate	TBD1	x.x	Unsigned64		V

7. Rate Based Abatement Algorithm

Editor’s Note: Need to scrub this section to use the reporting node and reacting node terminology and remove the server and client terms use for the SOC description.

This section is pulled from [I-D.SOC-overload-rate-control], with minor changes needed to make it apply to the Diameter protocol.

7.1. Overview

The server is the one protected by the overload control algorithm defined here. This is also referred to as the reporting node. The client is the one that throttles traffic towards the server. This is also referred to as the reacting node.

Following the procedures defined in [draft-ietf-dime-doic], the server and clients signal one another support for rate-based overload control.

Editor’s Note: Need to scrub this section to use the reporting node and reacting node terminology and remove the server and client terms.

Then periodically, the server relies on internal measurements (e.g. CPU utilization, queueing delay...) to evaluate its overload state and estimate a target Diameter request rate in number of requests per second (as opposed to target percent reduction in the case of loss-based abatement).

When in an overloaded state, the reporting node uses the OC-OLR AVP to inform reacting nodes of its overload state and of the target Diameter request rate.

Upon receiving the overload report with a target Diameter request rate, each reacting node throttles new Diameter requests towards the reporting node.

7.2. Reporting Node Behavior

The actual algorithm used by the reporting node to determine its overload state and estimate a target Diameter request rate is beyond the scope of this document.

However, the reporting node MUST periodically evaluate its overload state and estimate a target Diameter request rate beyond which it would become overloaded. The server must allocate a portion of the target Diameter request rate to each of its reacting nodes. The server may set the same rate for every reacting node, or may set different rates for different reacting node.

The max rate determined by the reporting node for a reacting node applies to the entire stream of Diameter requests, even though throttling may only affect a particular subset of the requests, since the reacting node might can apply priority as part of its decision of which requests to throttle.

When setting the maximum rate for a particular reacting node, the reporting node may need take into account the workload (e.g. cpu load per request) of the distribution of message types from that reacting node. Furthermore, because the reacting node may prioritize the specific types of messages it sends while under overload restriction, this distribution of message types may be different from the message distribution for that reacting node under non-overload conditions (e.g., either higher or lower cpu load).

Note that the AVP for the rate algorithm is an upper bound (in request messages per second) on the traffic sent by the reacting node to the reporting node. The reacting node may send traffic at a rate significantly lower than the upper bound, for a variety of reasons.

In other words, when multiple reacting nodes are being controlled by an overloaded reporting node, at any given time some reacting nodes may receive requests at a rate below its target Diameter request rate while others above that target rate. But the resulting request rate presented to the overloaded reporting node will converge towards the target Diameter request rate.

Upon detection of overload, and the determination to invoke overload controls, the reporting node MUST follow the specifications in [draft-ietf-dime-ovli] to notify its clients of the allocated target Diameter request rate.

The reporting node MUST use the OC-Maximum-Rate AVP defined in this specification to communicate a target Diameter request rate to each of its clients.

7.3. Reacting Node Behavior

7.3.1. Default algorithm

In determining whether or not to transmit a specific message, the reacting node may use any algorithm that limits the message rate to 1/T messages per second. It may be strictly deterministic, or it may be probabilistic. It may, or may not, have a tolerance factor, to allow for short bursts, as long as the long term rate remains below 1/T. The algorithm may have provisions for prioritizing traffic.

If the algorithm requires other parameters (in addition to "T", which is 1/OC-Maximum-Rate), they may be set autonomously by the client, or they may be negotiated independently between client and server.

In either case, the coordination is out of scope for this document. The default algorithms presented here (one without provisions for prioritizing traffic, one with) are only examples. Other algorithms that forward messages in conformance with the upper bound of 1/T messages per second may be used.

To throttle new Diameter requests at the rate specified in the OC-Maximum-Rate AVP value sent by the reporting node to its reacting nodes, the reacting node MAY use the proposed default algorithm for rate-based control or any other equivalent algorithm.

The default Leaky Bucket algorithm presented here is based on [ITU-T Rec. I.371] Appendix A.2. The algorithm makes it possible for clients to deliver Diameter requests at a rate specified in the OC-Maximum-Rate value with tolerance parameter TAU (preferably configurable).

Conceptually, the Leaky Bucket algorithm can be viewed as a finite capacity bucket whose real-valued content drains out at a continuous rate of 1 unit of content per time unit and whose content increases by the increment T for each forwarded Diameter request. T is computed as the inverse of the rate specified in the OC-Maximum-Rate AVP value, namely T = 1 / OC-Maximum-Rate.

Note that when the OC-Maximum-Rate value is 0 with a non-zero OC-Validity-Duration, then the reacting node should reject 100% of Diameter requests destined to the overloaded reporting node. However, when the OC-Validity-Duration value is 0, the client should stop throttling.

If, at a new Diameter request arrival, the content of the bucket is less than or equal to the limit value TAU, then the Diameter request is forwarded to the server; otherwise, the Diameter request is rejected.

Note that the capacity of the bucket (the upper bound of the counter) is (T + TAU).

The tolerance parameter TAU determines how close the long-term admitted rate is to an ideal control that would admit all Diameter requests for arrival rates less than 1/T and then admit Diameter requests precisely at the rate of 1/T for arrival rates above 1/T. In particular at mean arrival rates close to 1/T, it determines the tolerance to deviation of the inter-arrival time from T (the larger TAU the more tolerance to deviations from the inter-departure interval T).

This deviation from the inter-departure interval influences the admitted rate burstyness, or the number of consecutive Diameter requests forwarded to the reporting node (burst size proportional to TAU over the difference between 1/T and the arrival rate).

Reporting nodes with a very large number of clients, each with a relatively small arrival rate, will generally benefit from a smaller value for TAU in order to limit queuing (and hence response times) at the reporting node when subjected to a sudden surge of traffic from all reacting nodes. Conversely, a reporting node with a relatively small number of reacting nodes, each with proportionally larger arrival rate, will benefit from a larger value of TAU.

Once the control has been activated, at the arrival time of the k-th new Diameter request, ta(k), the content of the bucket is provisionally updated to the value

X' = X - (ta(k) - LCT)

where X is the value of the leaky bucket counter after arrival of the last forwarded Diameter request, and LCT is the time at which the last Diameter request was forwarded.

If X' is less than or equal to the limit value TAU, then the new Diameter request is forwarded and the leaky bucket counter X is set to X' (or to 0 if X' is negative) plus the increment T, and LCT is set to the current time ta(k). If X' is greater than the limit value TAU, then the new Diameter request is rejected and the values of X and LCT are unchanged.

When the first response from the reporting node has been received indicating control activation (OC-Validity-Duration>0), LCT is set to the time of activation, and the leaky bucket counter is initialized to the parameter TAU0 (preferably configurable) which is 0 or larger but less than or equal to TAU.

TAU can assume any positive real number value and is not necessarily bounded by T.

TAU=4*T is a reasonable compromise between burst size and throttled rate adaptation at low offered rate.

Note that specification of a value for TAU, and any communication or coordination between servers, is beyond the scope of this document.

A reference algorithm is shown below.

No priority case:

```
// T: inter-transmission interval, set to 1 / OC-Maximum-Rate
// TAU: tolerance parameter
// ta: arrival time of the most recent arrival
// LCT: arrival time of last SIP request that was sent to the server
//      (initialized to the first arrival time)
// X: current value of the leaky bucket counter (initialized to
//    TAU0)

// After most recent arrival, calculate auxiliary variable Xp
Xp = X - (ta - LCT);

if (Xp <= TAU) {
  // Transmit SIP request
  // Update X and LCT
  X = max (0, Xp) + T;
  LCT = ta;
} else {
  // Reject SIP request
  // Do not update X and LCT
}
```

7.3.2. Priority treatment

The reacting node is responsible for applying message priority and for maintaining two categories of requests: Request candidates for reduction, requests not subject to reduction (except under extenuating circumstances when there aren’t any messages in the first category that can be reduced).

Accordingly, the proposed Leaky bucket implementation is modified to support priority using two thresholds for Diameter requests in the set of request candidates for reduction. With two priorities, the proposed Leaky bucket requires two thresholds TAU1 < TAU2:

o All new requests would be admitted when the leaky bucket counter is at or below TAU1,

o Only higher priority requests would be admitted when the leaky bucket counter is between TAU1 and TAU2,

o All requests would be rejected when the bucket counter is above TAU2.

This can be generalized to n priorities using n thresholds for n>2 in the obvious way.

With a priority scheme that relies on two tolerance parameters (TAU2 influences the priority traffic, TAU1 influences the non-priority traffic), always set TAU1 <= TAU2 (TAU is replaced by TAU1 and TAU2). Setting both tolerance parameters to the same value is equivalent to having no priority. TAU1 influences the admitted rate the same way as TAU does when no priority is set. And the larger the difference between TAU1 and TAU2, the closer the control is to strict priority queueing.

TAU1 and TAU2 can assume any positive real number value and is not necessarily bounded by T.

Reasonable values for TAU0, TAU1 & TAU2 are: TAU0 = 0, TAU1 = 1/2 * TAU2 and TAU2 = 10 * T.

Note that specification of a value for TAU1 and TAU2, and any communication or coordination between servers, is beyond the scope of this document.

A reference algorithm is shown below.

Priority case:

```
// T: inter-transmission interval, set to 1 / OC-Maximum-Rate
// TAU1: tolerance parameter of no priority SIP requests
// TAU2: tolerance parameter of priority SIP requests
// ta: arrival time of the most recent arrival
// LCT: arrival time of last SIP request that was sent to the server
//      (initialized to the first arrival time)
// X: current value of the leaky bucket counter (initialized to
//    TAU0)

// After most recent arrival, calculate auxiliary variable Xp
Xp = X - (ta - LCT);

if (AnyRequestReceived && Xp <= TAU1) || (PriorityRequestReceived &&
Xp <= TAU2 && Xp > TAU1) {
  // Transmit SIP request
  // Update X and LCT
  X = max (0, Xp) + T;
  LCT = ta;
} else {
  // Reject SIP request
  // Do not update X and LCT
}
```

7.3.3. Optional enhancement: avoidance of resonance

As the number of reacting node sources of traffic increases and the throughput of the reporting node decreases, the maximum rate admitted by each reacting node needs to decrease, and therefore the value of T becomes larger. Under some circumstances, e.g. if the traffic arises very quickly simultaneously at many sources, the occupancies of each bucket can become synchronized, resulting in the admissions from each source being close in time and batched or very 'peaky' arrivals at the reporting node, which not only gives rise to control instability, but also very poor delays and even lost messages. An appropriate term for this is 'resonance' [Erramilli].

If the network topology is such that this can occur, then a simple way to avoid this is to randomize the bucket occupancy at two appropriate points: At the activation of control, and whenever the bucket empties, as follows.

After updating the value of the leaky bucket to X', generate a value u as follows:

if X' > 0, then u=0

else if X' <= 0 then uniformly distributed between -1/2 and +1/2

Then (only) if the arrival is admitted, increase the bucket by an amount T + uT, which will therefore be just T if the bucket hadn't emptied, or lie between T/2 and 3T/2 if it had.

This randomization should also be done when control is activated, i.e. instead of simply initializing the leaky bucket counter to TAU0, initialize it to TAU0 + uT, where u is uniformly distributed as above. Since activation would have been a result of response to a request sent by the reacting node, the second term in this expression can be interpreted as being the bucket increment following that admission.

This method has the following characteristics:

- If TAU0 is chosen to be equal to TAU and all sources were to activate control at the same time due to an extremely high request rate, then the time until the first request admitted by each client would be uniformly distributed over [0,T];

- The maximum occupancy is TAU + (3/2)T, rather than TAU + T without randomization;

- For the special case of 'classic gapping' where TAU=0, then the minimum time between admissions is uniformly distributed over [T/2, 3T/2], and the mean time between admissions is the same, i.e. T+1/R where R is the request arrival rate;

- At high load randomization rarely occurs, so there is no loss of precision of the admitted rate, even though the randomized 'phasing' of the buckets remains.

8. IANA Consideration

TBD

9. Security Considerations

Agent overload is an extension to the based Diameter overload mechanism. As such, all of the security considerations outlined in [I-D.ietf-dime-ovli] apply to the agent overload scenarios.

10. Acknowledgements

11. References

11.1. Normative References

[I-D.SOC-overload-rate-control]
Noel, E., "SIP Overload Rate Control", February 2014.

[I-D.ietf-dime-ovli]
Korhonen, J., "Diameter Overload Indication Conveyance", October 2013.

[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.donovan-agent-overload]
Donovan, S., "Diameter Agent Overload", February 2014.

Authors' Addresses

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

Email: srdonovan@usdonovans.com

Eric Noel
AT&T Labs
200s Laurel Avenue
Middletown, NJ 07747
United States

Diameter Maintenance and Extensions (DIME)   L. Morand, Ed.
Internet-Draft   Orange Labs
Intended status: Best Current Practice   V. Fajardo
Expires: March 27, 2015   Fluke Networks
                                                               H. Tschofenig

September 23, 2014

Diameter Applications Design Guidelines
draft-ietf-dime-app-design-guide-28

Abstract

The Diameter base protocol provides facilities for protocol extensibility enabling to define new Diameter applications or modify existing applications. This document is a companion document to the Diameter Base protocol that further explains and clarifies the rules to extend Diameter. Furthermore, this document provides guidelines to Diameter application designers reusing/defining Diameter applications or creating generic Diameter extensions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Diameter base protocol [RFC6733] is intended to provide an Authentication, Authorization, and Accounting (AAA) framework for applications such as network access or IP mobility in both local and roaming situations. This protocol provides the ability for Diameter peers to exchange messages carrying data in the form of Attribute-Value Pairs (AVPs).

The Diameter base protocol provides facilities to extend Diameter (see Section 1.3 of [RFC6733]) to support new functionality. In the context of this document, extending Diameter means one of the following:

1. Addition of new functionality to an existing Diameter application without defining a new application.

2. Addition of new functionality to an existing Diameter application that requires the definition of a new application.

3. The definition of an entirely new Diameter application to offer functionality not supported by existing applications.

4. The definition of a new generic functionality that can be reused across different applications.

All of these choices are design decisions that can be done by any combination of reusing existing or defining new commands, AVPs or AVP values. However, application designers do not have complete freedom when making their design. A number of rules have been defined in [RFC6733] that place constraints on when an extension requires the allocation of a new Diameter application identifier or a new command code value. The objective of this document is the following:

- Clarify the Diameter extensibility rules as defined in the Diameter base protocol.

o Discuss design choices and provide guidelines when defining new applications.

o Present trade-off choices.

2. Terminology

This document reuses the terminology defined in [RFC6733]. Additionally, the following terms and acronyms are used in this application:

Application Extension of the Diameter base protocol [RFC6733] via the addition of new commands or AVPs. Each application is uniquely identified by an IANA-allocated application identifier value.

Command Diameter request or answer carrying AVPs between Diameter endpoints. Each command is uniquely identified by a IANA-allocated command code value and is described by a Command Code Format (CCF) for an application.

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 [RFC2119].

3. Overview

As designed, the Diameter base protocol [RFC6733] can be seen as a two-layer protocol. The lower layer is mainly responsible for managing connections between neighboring peers and for message routing. The upper layer is where the Diameter applications reside. This model is in line with a Diameter node having an application layer and a peer-to-peer delivery layer. The Diameter base protocol document defines the architecture and behavior of the message delivery layer and then provides the framework for designing Diameter applications on the application layer. This framework includes definitions of application sessions and accounting support (see Section 8 and Section 9 of [RFC6733]). Accordingly, a Diameter node is seen in this document as a single instance of a Diameter message delivery layer and one or more Diameter applications using it.

The Diameter base protocol is designed to be extensible and the principles are described in the Section 1.3 of [RFC6733]. As a summary, Diameter can be extended by:

1. Defining new AVP values

2. Creating new AVPs

3. Creating new commands

4. Creating new applications

As a main guiding principle, application designers SHOULD follow the following recommendation: "try to re-use as much as possible!". It will reduce the time to finalize specification writing, and it will lead to a smaller implementation effort as well as reduce the need for testing. In general, it is clever to avoid duplicate effort when possible.

However, re-use is not appropriate when the existing functionality does not fit the new requirement and/or the re-use leads to ambiguity.

The impact on extending existing applications can be categorized into two groups:

Minor Extension: Enhancing the functional scope of an existing application by the addition of optional features to support. Such enhancement has no backward compatibility issue with the existing application.

A typical example would be the definition of a new optional AVP for use in an existing command. Diameter implementations supporting the existing application but not the new AVP will simply ignore it, without consequences for the Diameter message handling, as described in [RFC6733]. The standardization effort will be fairly small.

Major Extension: Enhancing an application that requires the definition of a new Diameter application. Such enhancement causes backward compatibility issue with existing implementations supporting the application.

Typical examples would be the creation of a new command for providing functionality not supported by existing applications or the definition of a new AVP to be carried in an existing command with the M-bit set in the AVP flags (see Section 4.1 of [RFC6733] for definition of the "M-bit"). For such extension, a significant specification effort is required and a careful approach is recommended.

4. Reusing Existing Diameter Applications

An existing application may need to be enhanced to fulfill new requirements and these modifications can be at the command level and/or at the AVP level. The following sections describe the possible modifications that can be performed on existing applications and their related impact.

4.1. Adding a New Command

Adding a new command to an existing application is considered as a major extension and requires a new Diameter application to be defined, as stated in the Section 1.3.4 of [RFC6733]. The need for a new application is because a Diameter node that is not upgraded to support the new command(s) within the (existing) application would reject any unknown command with the protocol error DIAMETER_COMMAND_UNSUPPORTED and cause the failure of the transaction. The new application ensures that Diameter nodes only receive commands within the context of applications they support.

Adding a new command means either defining a completely new command or importing the command's Command Code Format (CCF) syntax from another application whereby the new application inherits some or all of the functionality of the application where the command came from. In the former case, the decision to create a new application is straightforward since this is typically a result of adding a new functionality that does not exist yet. For the latter, the decision to create a new application will depend on whether importing the command in a new application is more suitable than simply using the existing application as it is in conjunction with any other application.

An example considers the Diameter EAP application [RFC4072] and the Diameter Network Access Server application [RFC7155]. When network access authentication using EAP is required, the Diameter EAP commands (Diameter-EAP-Request/Diameter-EAP-Answer) are used; otherwise the Diameter Network Access Server application will be used. When the Diameter EAP application is used, the accounting exchanges defined in the Diameter Network Access Server may be used.

However, in general, it is difficult to come to a hard guideline, and so a case-by-case study of each application requirement should be applied. Before adding or importing a command, application designers should consider the following:

- o Can the new functionality be fulfilled by creating a new command independent from any existing command? In this case, the resulting new application and the existing application can work independent of, but cooperating with each other.

- Can the existing command be reused without major extensions and therefore without the need for the definition of a new application, e.g. new functionality introduced by the creation of new optional AVPs.

It is important to note that importing commands too liberally could result in a monolithic and hard to manage application supporting too many different features.

4.2. Deleting an Existing Command

Although this process is not typical, removing a command from an application requires a new Diameter application to be defined and then it is considered as a major extension. This is due to the fact that the reception of the deleted command would systematically result in a protocol error (i.e., DIAMETER_COMMAND_UNSUPPORTED).

It is unusual to delete an existing command from an application for the sake of deleting it or the functionality it represents. An exception might be if the intent of the deletion is to create a newer variance of the same application that is somehow simpler than the application initially specified.

4.3. Reusing Existing Commands

This section discusses rules in adding and/or deleting AVPs from an existing command of an existing application. The cases described in this section may not necessarily result in the creation of new applications.

From a historical point of view, it is worth to note that there was a strong recommendation to re-use existing commands in the [RFC3588] to prevent rapid depletion of code values available for vendor-specific commands. However, [RFC6733] has relaxed the allocation policy and enlarged the range of available code values for vendor-specific applications. Although reuse of existing commands is still RECOMMENDED, protocol designers can consider defining a new command when it provides a solution more suitable than the twisting of an existing command’s use and applications.

4.3.1. Adding AVPs to a Command

Based on the rules in [RFC6733], AVPs that are added to an existing command can be categorized into:

- o Mandatory (to understand) AVPs. As defined in [RFC6733], these are AVPs with the M-bit flag set in this command, which means that a Diameter node receiving them is required to understand not only their values but also their semantics. Failure to do so will cause an message handling error: either a error message with the result-code set to DIAMETER_AVP_UNSUPPORTED if the AVP not understood in a request or a application specific error handling if the given AVP is in an answer.

- o Optional (to understand) AVPs. As defined in [RFC6733], these are AVPs with the M-bit flag cleared in this command. A Diameter node receiving these AVPs can simply ignore them if it does not support them.

It is important to note that the definition given above are independent of whether these AVPs are required or optional in the command as specified by the command’s Command Code Format (CCF) syntax [RFC6733].

NOTE: As stated in [RFC6733], the M-bit setting for a given AVP is relevant to an application and each command within that application that includes the AVP.

The rules are strict in the case where the AVPs to be added in an exiting command are mandatory to understand, i.e., they have the M-bit set. A mandatory AVP MUST NOT be added to an existing command without defining a new Diameter application, as stated in [RFC6733]. This falls into the "Major Extensions" category. Despite the clarity of the rule, ambiguity still arises when evaluating whether a new AVP being added should be mandatory to begin with. Application designers should consider the following questions when deciding about the M-bit for a new AVP:

- o Would it be required for the receiving side to be able to process and understand the AVP and its content?

- o Would the new AVPs change the state machine of the application?

- o Would the presence of the new AVP lead to a different number of round-trips, effectively changing the state machine of the application?

- o Would the new AVP be used to differentiate between old and new variances of the same application whereby the two variances are not backward compatible?

- o Would the new AVP have duality in meaning, i.e., be used to carry application-related information as well as to indicate that the message is for a new application?

If the answer to at least one of the questions is "yes" then the M-bit MUST be set for the new AVP and a new Diameter application MUST be defined. This list of questions is non-exhaustive and other criteria MAY be taken into account in the decision process.

If application designers are instead contemplating the use of optional AVPs, i.e., with the M-bit cleared, there are still pitfalls that will cause interoperability problems and therefore must be avoided. Some examples of these pitfalls are :

- Use of optional AVPs with intersecting meaning. One AVP has partially the same usage and meaning as another AVP. The presence of both can lead to confusion.

- An optional AVPs with dual purpose, i.e., to carry application data as well as to indicate support for one or more features. This has a tendency to introduce interpretation issues.

- Adding one or more optional AVPs and indicating (usually within descriptive text for the command) that at least one of them has to be understood by the receiver of the command. This would be equivalent to adding a mandatory AVP, i.e., an AVP with the M-bit set, to the command.

4.3.2. Deleting AVPs from a Command

Application designers may want to reuse an existing command but some of the AVP present in the command's CCF syntax specification may be irrelevant for the functionality foreseen to be supported by this command. It may be then tempting to delete those AVPs from the command.

The impacts of deleting an AVP from a command depends on its command code format specification and M-bit setting:

- Case 1: Deleting an AVP that is indicated as a required AVP (noted as {AVP}) in the command's CCF syntax specification (regardless of the M-bit setting).

  In this case, a new command code and subsequently a new Diameter application MUST be specified.

- Case 2: Deleting an AVP, which has the M-bit set, and is indicated as optional AVP (noted as [AVP]) in the command CCF) in the command's CCF syntax specification.

  In this case, no new command code has to be specified but the definition of a new Diameter application is REQUIRED.

- Case 3: Deleting an AVP, which has the M-bit cleared, and is indicated as [AVP] in the command's CCF syntax specification.

  In this case, the AVP can be deleted without consequences.

Application designers SHOULD attempt the reuse the command's CCF syntax specification without modification and simply ignore (but not delete) any optional AVP that will not be used. This is to maintain compatibility with existing applications that will not know about the new functionality as well as maintain the integrity of existing dictionaries.

4.3.3. Changing the Flags Setting of AVP in existing Commands

Although unusual, implementors may want to change the setting of the AVP flags a given AVP used in a command.

Into an existing command, a AVP that was initially defined as mandatory AVP to understand, i.e., an AVP with the M-bit flag set in the command, MAY be safely turned to an optional AVP, i.e., with the M-bit cleared. Any node supporting the existing application will still understand the AVP, whatever the setting of the M-bit. On the contrary, an AVP initially defined as an optional AVP to understand, i.e., an AVP with the M-bit flag cleared in the command, MUST NOT be changed into a mandatory AVP with the M-bit flag set without defining a new Diameter application. Setting the M-bit for an AVP that was defined as an optional AVP is equivalent to adding a new mandatory AVP to an existing command and the rules given in the section 4.3.1 apply.

All other AVP flags (V-bit, P-bit, reserved bits) MUST remain unchanged.

4.4. Reusing Existing AVPs

This section discusses rules in reusing existing AVP when reusing an existing command or defining a new command in a new application.

4.4.1. Setting of the AVP Flags

When reusing existing AVPs in a new application, application designers MUST specify the setting of the M-bit flag for a new Diameter application and, if necessary, for every command of the application that can carry these AVPs. In general, for AVPs defined outside of the Diameter base protocol, the characteristics of an AVP are tied to its role within a given application and the commands used in this application.

All other AVP flags (V-bit, P-bit, reserved bits) MUST remain unchanged.

4.4.2. Reuse of AVP of Type Enumerated

When reusing an AVP of type Enumerated in a command for a new application, it is RECOMMENDED to avoid modifying the set of valid values defined for this AVP. Modifying the set of Enumerated values includes adding a value or deprecating the use of a value defined initially for the AVP. Modifying the set of values will impact the application defining this AVP and all the applications using this AVP, causing potential interoperability issues: a value used by a peer that will not be recognized by all the nodes between the client and the server will cause an error response with the Result-Code AVP set to DIAMETER_INVALID_AVP_VALUE. When the full range of values defined for this Enumerated AVP is not suitable for the new application, it is RECOMMENDED to define a new AVP to avoid backwards compatibility issues with existing implementations.

5. Defining New Diameter Applications

5.1. Introduction

This section discusses the case where new applications have requirements that cannot be fulfilled by existing applications and would require definition of completely new commands, AVPs and/or AVP values. Typically, there is little ambiguity about the decision to create these types of applications. Some examples are the interfaces defined for the IP Multimedia Subsystem of 3GPP, e.g., Cx/Dx ([TS29.228] and [TS29.229]), Sh ([TS29.328] and [TS29.329]) etc.

Application designers SHOULD try to import existing AVPs and AVP values for any newly defined commands. In certain cases where accounting will be used, the models described in Section 5.10 SHOULD also be considered.

Additional considerations are described in the following sections.

5.2. Defining New Commands

As a general recommendation, commands SHOULD NOT be defined from scratch. It is instead RECOMMENDED to re-use an existing command offering similar functionality and use it as a starting point. Code re-use lead to a smaller implementation effort as well as reduce the need for testing.

Moreover, the new command's CCF syntax specification SHOULD be carefully defined when considering applicability and extensibility of the application. If most of the AVPs contained in the command are indicated as fixed or required, it might be difficult to reuse the same command and therefore the same application in a slightly changed environment. Defining a command with most of the AVPs indicated as optional is considered as a good design choice in many cases, despite the flexibility it introduces in the protocol. Protocol designers MUST clearly state the reasons why these optional AVPs might or might not be present and properly define the corresponding behavior of the Diameter nodes when these AVPs are absent from the command.

NOTE: As a hint for protocol designers, it is not sufficient to just look at the command's CCF syntax specification. It is also necessary to carefully read through the accompanying text in the specification.

In the same way, the CCF syntax specification SHOULD be defined such that it will be possible to add any arbitrary optional AVPs with the M-bit cleared (including vendor-specific AVPs) without modifying the application. For this purpose, "* [AVP]" SHOULD be added in the command's CCF, which allows the addition of any arbitrary number of optional AVPs as described in [RFC6733].

5.3. Use of Application-Id in a Message

When designing new applications, application designers SHOULD specify that the Application Id carried in all session-level messages is the Application Id of the application using those messages. This includes the session-level messages defined in Diameter base protocol, i.e., RAR/RAA, STR/STA, ASR/ASA and possibly ACR/ACA in the coupled accounting model, see Section 5.10. Some existing specifications do not adhere to this rule for historical reasons. However, this guidance SHOULD be followed by new applications to avoid routing problems.

When a new application has been allocated with a new Application Id and it also reuses existing commands with or without modifications, the commands SHOULD use the newly allocated Application Id in the header and in all relevant Application Id AVPs (Auth-Application-Id or Acct-Application-Id) present in the commands message body.

Additionally, application designers using Vendor-Specific-Application-Id AVP SHOULD NOT use the Vendor-Id AVP to further dissect or differentiate the vendor-specification Application Id. Diameter routing is not based on the Vendor-Id. As such, the Vendor-Id SHOULD NOT be used as an additional input for routing or delivery of messages. The Vendor-Id AVP is an informational AVP only and kept for backward compatibility reasons.

5.4. Application-Specific Session State Machines

Section 8 of [RFC6733] provides session state machines for authentication, authorization and accounting (AAA) services and these session state machines are not intended to cover behavior outside of AAA. If a new application cannot clearly be categorized into any of these AAA services, it is RECOMMENDED that the application defines its own session state machine. Support for server-initiated request is a clear example where an application-specific session state machine would be needed, for example, the Rw interface for ITU-T push model (cf.[Q.3303.3]).

5.5. Session-Id AVP and Session Management

Diameter applications are usually designed with the aim of managing user sessions (e.g., Diameter network access session (NASREQ) application [RFC4005]) or specific service access session (e.g., Diameter SIP application [RFC4740]). In the Diameter base protocol, session state is referenced using the Session-Id AVP. All Diameter messages that use the same Session-Id will be bound to the same session. Diameter-based session management also implies that both Diameter client and server (and potentially proxy agents along the path) maintain session state information.

However, some applications may not need to rely on the Session-Id to identify and manage sessions because other information can be used instead to correlate Diameter messages. Indeed, the User-Name AVP or any other specific AVP can be present in every Diameter message and used therefore for message correlation. Some applications might not require the notion of Diameter session concept at all. For such applications, the Auth-Session-State AVP is usually set to NO_STATE_MAINTAINED in all Diameter messages and these applications are therefore designed as a set of stand-alone transactions. Even if an explicit access session termination is required, application-specific commands are defined and used instead of the Session-Termination-Request/Answer (STR/STA) or Abort-Session-Request/Answer (ASR/ASA) defined in the Diameter base protocol [RFC6733]. In such a case, the Session-Id is not significant.

Based on these considerations, protocol designers should carefully appraise whether the Diameter application being defined relies on the session management specified in the Diameter base protocol:

- o If it is, the Diameter command defined for the new application MUST include the Session-Id AVP defined in the Diameter base protocol [RFC6733] and the Session-Id AVP MUST be used for correlation of messages related to the same session. Guidance on the use of the Auth-Session-State AVP is given in the Diameter base protocol [RFC6733].

- Otherwise, because session management is not required or the application relies on its own session management mechanism, Diameter commands for the application need not include the Session-Id AVP. If any specific session management concept is supported by the application, the application documentation MUST clearly specify how the session is handled between client and server (and possibly Diameter agents in the path). Moreover, because the application is not maintaining session state at the Diameter base protocol level, the Auth-Session-State AVP MUST be included in all Diameter commands for the application and MUST be set to NO_STATE_MAINTAINED.

5.6. Use of Enumerated Type AVPs

The type Enumerated was initially defined to provide a list of valid values for an AVP with their respective interpretation described in the specification. For instance, AVPs of type Enumerated can be used to provide further information on the reason for the termination of a session or a specific action to perform upon the reception of the request.

As described in the section 4.4.2 above, defining an AVP of type Enumerated presents some limitations in term of extensibility and reusability. Indeed, the finite set of valid values defined at the definition of the AVP of type Enumerated cannot be modified in practice without causing backward compatibility issues with existing implementations. As a consequence, AVPs of Type Enumerated MUST NOT be extended by adding new values to support new capabilities. Diameter protocol designers SHOULD carefully consider before defining an Enumerated AVP whether the set of values will remain unchanged or new values may be required in a near future. If such extension is foreseen or cannot be avoided, it is RECOMMENED to rather define AVPs of type Unsigned32 or Unsigned64 in which the data field would contain an address space representing "values" that would have the same use of Enumerated values. Whereas only the initial values defined at the definition of the AVP of type Enumerated are valid as described in section 4.4.2, any value from the address space from 0 to 2^32 - 1 for AVPs of type Unsigned32 or from 0 to 2^64 - 1 for AVPs of type Unsigned64 is valid at the Diameter base protocol level and will not interoperability issues for intermediary nodes between clients and servers. Only clients and servers will be able to process the values at the application layer.

For illustration, an AVP describing possible access networks would be defined as follow:

Access-Network-Type AVP (XXX) is of type Unsigned32 and contains a 32-bit address space representing types of access networks. This application defines the following classes of access networks, all identified by the thousands digit in the decimal notation:

- o 1xxx (Mobile Access Networks)
- o 2xxx (Fixed Access Network)
- o 3xxx (Wireless Access Networks)

Values that fall within the Mobile Access Networks category are used to inform a peer that a request has been sent for a user attached to a mobile access network. The following values are defined in this application:

1001: 3GPP-GERAN

The user is attached to a GSM EDGE Radio Access Network.

1002: 3GPP-UTRAN-FDD

The user is attached to a UMTS access network that uses frequency-division duplexing for duplexing.

Unlike Enumerated AVP, any new value can be added in the address space defined by this Unsigned32 AVP without modifying the definition of the AVP. There is therefore no risk of backward compatibility issue, especially when intermediate nodes may be present between Diameter endpoints.

In the same line, AVPs of type Enumerated are too often used as a simple Boolean flag, indicating for instance a specific permission or capability, and therefore only two values are defined, e.g., TRUE/FALSE, AUTORIZED/UNAUTHORIZED or SUPPORTED/UNSUPPORTED. This is a sub-optimal design since it limits the extensibility of the application: any new capability/permission would have to be supported by a new AVP or new Enumerated value of the already defined AVP, with the backward compatibility issues described above. Instead of using an Enumerated AVP for a Boolean flag, protocol designers SHOULD use AVPs of type Unsigned32 or Unsigned64 AVP in which the data field would be defined as bit mask whose bit settings are described in the relevant Diameter application specification. Such AVPs can be reused and extended without major impact on the Diameter application. The bit mask SHOULD leave room for future additions. Examples of AVPs that use bit masks are the Session-Binding AVP defined in [RFC6733] and the MIP6-Feature-Vector AVP defined in [RFC5447].

5.7. Application-Specific Message Routing

As described in [RFC6733], a Diameter request that needs to be sent to a home server serving a specific realm, but not to a specific server (such as the first request of a series of round trips), will contain a Destination-Realm AVP and no Destination-Host AVP.

For such a request, the message routing usually relies only on the Destination-Realm AVP and the Application Id present in the request message header. However, some applications may need to rely on the User-Name AVP or any other application-specific AVP present in the request to determine the final destination of a request, e.g., to find the target AAA server hosting the authorization information for a given user when multiple AAA servers are addressable in the realm.

In such a context, basic routing mechanisms described in [RFC6733] are not fully suitable, and additional application-level routing mechanisms MUST be described in the application documentation to provide such specific AVP-based routing. Such functionality will be basically hosted by an application-specific proxy agent that will be responsible for routing decisions based on the received specific AVPs.

Examples of such application-specific routing functions can be found in the Cx/Dx applications ([TS29.228] and [TS29.229]) of the 3GPP IP Multimedia Subsystem, in which the proxy agent (Subscriber Location Function aka SLF) uses specific application-level identities found in the request to determine the final destination of the message.

Whatever the criteria used to establish the routing path of the request, the routing of the answer MUST follow the reverse path of the request, as described in [RFC6733], with the answer being sent to the source of the received request, using transaction states and hop-by-hop identifier matching. This ensures that the Diameter Relay or Proxy agents in the request routing path will be able to release the transaction state upon receipt of the corresponding answer, avoiding unnecessary failover. Moreover, especially in roaming cases, proxy agents in the path must be able to apply local policies when receiving the answer from the server during authentication/ authorization and/or accounting procedures, and maintain up-to-date session state information by keeping track of all authorized active sessions. Therefore, application designers MUST NOT modify the answer-routing principles described in [RFC6733] when defining a new application.

5.8. Translation Agents

As defined in [RFC6733], a translation agent is a device that provides interworking between Diameter and another AAA protocol, such as RADIUS .

In the case of RADIUS, it was initially thought that defining the translation function would be straightforward by adopting few basic principles, e.g., by the use of a shared range of code values for RADIUS attributes and Diameter AVPs. Guidelines for implementing a RADIUS-Diameter translation agent were put into the Diameter NASREQ Application ([RFC4005]).

However, it was acknowledged that such translation mechanism was not so obvious and deeper protocol analysis was required to ensure efficient interworking between RADIUS and Diameter. Moreover, the interworking requirements depend on the functionalities provided by the Diameter application under specification, and a case-by-case analysis is required. As a consequence, all the material related to RADIUS-to-Diameter translation is removed from the new version of the Diameter NASREQ application specification [RFC7155], which deprecates the RFC4005 ([RFC4005]).

Therefore, protocol designers SHOULD NOT assume the availability of a "standard" Diameter-to-RADIUS gateways agent when planning to interoperate with the RADIUS infrastructure. They SHOULD specify the required translation mechanism along with the Diameter application, if needed. This recommendation applies for any kind of translation.

5.9. End-to-End Application Capabilities Exchange

Diameter applications can rely on optional AVPs to exchange application-specific capabilities and features. These AVPs can be exchanged on an end-to-end basis at the application layer. Examples of this can be found with the MIP6-Feature-Vector AVP in [RFC5447] and the QoS-Capability AVP in [RFC5777].

End-to-end capabilities AVPs can be added as optional AVPs with the M-bit cleared to existing applications to announce support of new functionality. Receivers that do not understand these AVPs or the AVP values can simply ignore them, as stated in [RFC6733]. When supported, receivers of these AVPs can discover the additional functionality supported by the Diameter end-point originating the request and behave accordingly when processing the request. Senders of these AVPs can safely assume the receiving end-point does not support any functionality carried by the AVP if it is not present in corresponding response. This is useful in cases where deployment choices are offered, and the generic design can be made available for a number of applications.

When used in a new application, these end-to-end capabilities AVPs SHOULD be added as optional AVP into the CCF of the commands used by the new application. Protocol designers SHOULD clearly specify this end-to-end capabilities exchange and the corresponding behaviour of the Diameter nodes supporting the application.

It is also important to note that this end-to-end capabilities exchange relying on the use of optional AVPs is not meant as a generic mechanism to support extensibility of Diameter applications with arbitrary functionality. When the added features drastically change the Diameter application or when Diameter agents must be upgraded to support the new features, a new application SHOULD be defined, as recommended in [RFC6733].

5.10. Diameter Accounting Support

Accounting can be treated as an auxiliary application that is used in support of other applications. In most cases, accounting support is required when defining new applications. This document provides two possible models for using accounting:

Split Accounting Model:

In this model, the accounting messages will use the Diameter base accounting Application Id (value of 3). The design implication for this is that the accounting is treated as an independent application, especially for Diameter routing. This means that accounting commands emanating from an application may be routed separately from the rest of the other application messages. This may also imply that the messages end up in a central accounting server. A split accounting model is a good design choice when:

* The application itself does not define its own accounting commands.

* The overall system architecture permits the use of centralized accounting for one or more Diameter applications.

Centralizing accounting may have advantages but there are also drawbacks. The model assumes that the accounting server can differentiate received accounting messages. Since the received accounting messages can be for any application and/or service, the accounting server MUST have a method to match accounting messages with applications and/or services being accounted for. This may mean defining new AVPs, checking the presence, absence or contents of existing AVPs, or checking the contents of the accounting record itself. One of these means could be to insert into the request sent to the accounting server an Auth-Application-Id AVP containing the identifier of the application for which the accounting request is sent. But in general, there is no clean and generic scheme for sorting these messages. Therefore, this model SHOULD NOT be used when all received accounting messages cannot be clearly identified and sorted. For most cases, the use of Coupled Accounting Model is RECOMMENDED.

Coupled Accounting Model:

In this model, the accounting messages will use the Application Id of the application using the accounting service. The design implication for this is that the accounting messages are tightly coupled with the application itself; meaning that accounting messages will be routed like the other application messages. It would then be the responsibility of the application server (application entity receiving the ACR message) to send the accounting records carried by the accounting messages to the proper accounting server. The application server is also responsible for formulating a proper response (ACA). A coupled accounting model is a good design choice when:

* The system architecture or deployment does not provide an accounting server that supports Diameter. Consequently, the application server MUST be provisioned to use a different protocol to access the accounting server, e.g., via LDAP, SOAP etc. This case includes the support of older accounting systems that are not Diameter aware.

* The system architecture or deployment requires that the accounting service for the specific application should be handled by the application itself.

In all cases above, there will generally be no direct Diameter access to the accounting server.

These models provide a basis for using accounting messages. Application designers may obviously deviate from these models provided that the factors being addressed here have also been taken into account. As a general recommendation, application designers SHOULD NOT define a new set of commands to carry application-specific accounting records.

5.11. Diameter Security Mechanisms

As specified in [RFC6733], the Diameter message exchange SHOULD be secured between neighboring Diameter peers using TLS/TCP or DTLS/SCTP. However, IPsec MAY also be deployed to secure communication between Diameter peers. When IPsec is used instead of TLS or DTLS, the following recommendations apply.

IPsec ESP [RFC4301] in transport mode with non-null encryption and authentication algorithms MUST be used to provide per-packet authentication, integrity protection and confidentiality, and support the replay protection mechanisms of IPsec. IKEv2 [RFC5996] SHOULD be used for performing mutual authentication and for establishing and maintaining security associations (SAs).

IKEv1 [RFC2409] was used with RFC 3588 [RFC3588] and for easier migration from IKEv1 based implementations both RSA digital signatures and pre-shared keys SHOULD be supported in IKEv2. However, if IKEv1 is used, implementers SHOULD follow the guidelines given in Section 13.1 of RFC 3588 [RFC3588].

6. Defining Generic Diameter Extensions

Generic Diameter extensions are AVPs, commands or applications that are designed to support other Diameter applications. They are auxiliary applications meant to improve or enhance the Diameter protocol itself or Diameter applications/functionality. Some examples include the extensions to support realm-based redirection of Diameter requests (see [RFC7075]), convey a specific set of priority parameters influencing the distribution of resources (see [RFC6735]), and the support for QoS AVPs (see [RFC5777]).

Since generic extensions may cover many aspects of Diameter and Diameter applications, it is not possible to enumerate all scenarios. However, some of the most common considerations are as follows:

Backward Compatibility:

When defining generic extensions designed to be supported by existing Diameter applications, protocol designers MUST consider the potential impacts of the introduction of the new extension on the behavior of node that would not be yet upgraded to support/understand this new extension. Designers MUST also ensure that new extensions do not break expected message delivery layer behavior.

Forward Compatibility:

Protocol designers MUST ensure that their design will not introduce undue restrictions for future applications.

Trade-off in Signaling:

Designers may have to choose between the use of optional AVPs piggybacked onto existing commands versus defining new commands and applications. Optional AVPs are simpler to implement and may not need changes to existing applications. However, this ties the sending of extension data to the application's transmission of a message. This has consequences if the application and the extensions have different timing requirements. The use of commands and applications solves this issue, but the trade-off is the additional complexity of defining and deploying a new application. It is left up to the designer to find a good balance among these trade-offs based on the requirements of the extension.

In practice, generic extensions often use optional AVPs because they are simple and non-intrusive to the application that would carry them. Peers that do not support the generic extensions need not understand nor recognize these optional AVPs. However, it is RECOMMENDED that the authors of the extension specify the context or usage of the optional AVPs. As an example, in the case that the AVP can be used only by a specific set of applications then the specification MUST enumerate these applications and the scenarios when the optional AVPs will be used. In the case where the optional AVPs can be carried by any application, it should be sufficient to specify such a use case and perhaps provide specific examples of applications using them.

In most cases, these optional AVPs piggybacked by applications would be defined as a Grouped AVP and it would encapsulate all the functionality of the generic extension. In practice, it is not uncommon that the Grouped AVP will encapsulate an existing AVP that has previously been defined as mandatory ('M'-bit set) e.g., 3GPP IMS Cx/Dx interfaces ([TS29.228] and [TS29.229]).

7. Guidelines for Registrations of Diameter Values

As summarized in the Section 3 of this document and further described in the Section 1.3 of [RFC6733], there are four main ways to extend Diameter. The process for defining new functionality slightly varies based on the different extensions. This section provides protocol designers with some guidance regarding the definition of values for possible Diameter extensions and the necessary interaction with IANA to register the new functionality.

a. Defining new AVP values

The specifications defining AVPs and AVP values MUST provide guidance for defining new values and the corresponding policy for adding these values. For example, the RFC 5777 [RFC5777] defines the Treatment-Action AVP which contains a list of valid values corresponding to pre-defined actions (drop, shape, mark, permit). This set of values can be extended following the Specification Required policy defined in [RFC5226]. As a second example, the Diameter base specification [RFC6733] defines the Result-Code AVP that contains a 32-bit address space used to identity possible errors. According to the Section 11.3.2 of [RFC6733], new values can be assigned by IANA via an IETF Review process [RFC5226].

b. Creating new AVPs

Two different types of AVP Codes namespaces can be used to create a new AVPs:

* IETF AVP Codes namespace;

* Vendor-specific AVP Codes namespace.

In the latter case, a vendor needs to be first assigned by IANA with a private enterprise number, which can be used within the Vendor-Id field of the vendor-specific AVP. This enterprise number delimits a private namespace in which the vendor is responsible for vendor-specific AVP code value assignment. The absence of a Vendor-Id or a Vendor-Id value of zero (0) in the AVP header identifies standard AVPs from the IETF AVP Codes namespace managed by IANA. The allocation of code values from the IANA-managed namespace is conditioned by an Expert Review of the specification defining the AVPs or an IETF review if a block of AVPs needs to be assigned. Moreover, the remaining bits of the AVP Flags field of the AVP header are also assigned via Standard Action if the creation of new AVP Flags is desired.

c. Creating new commands

Unlike the AVP Code namespace, the Command Code namespace is flat but the range of values is subdivided into three chunks with distinct IANA registration policies:

* A range of standard Command Code values that are allocated via IETF review;

* A range of vendor-specific Command Code values that are allocated on a First-Come/First-Served basis;

* A range of values reserved only for experimental and testing purposes.

As for AVP Flags, the remaining bits of the Command Flags field of the Diameter header are also assigned via a Standards Action to create new Command Flags if required.

d. Creating new applications

Similarly to the Command Code namespace, the Application-Id namespace is flat but divided into two distinct ranges:

* A range of values reserved for standard Application-Ids allocated after Expert Review of the specification defining the standard application;

* A range for values for vendor specific applications, allocated by IANA on a First-Come/First-Serve basis.

The IANA AAA parameters page can be found at http://www.iana.org/assignments/aaa-parameters and the enterprise number IANA page is available at http://www.iana.org/assignments/enterprise-numbers. More details on the policies followed by IANA for namespace management (e.g. First-Come/First-Served, Expert Review, IETF Review, etc.) can be found in [RFC5226].

NOTE:
When the same functionality/extension is used by more than one vendor, it is RECOMMENDED to define a standard extension. Moreover, a vendor-specific extension SHOULD be registered to avoid interoperability issues in the same network. With this aim, the registration policy of vendor-specific extension has been simplified with the publication of [RFC6733] and the namespace reserved for vendor-specific extensions is large enough to avoid exhaustion.

8. IANA Considerations

This document does not require actions by IANA.

9. Security Considerations

This document provides guidelines and considerations for extending Diameter and Diameter applications. Although such an extension may be related to a security functionality, the document does not explicitly give additional guidance on enhancing Diameter with respect to security. However, as a general guideline, it is recommended that any Diameter extension SHOULD NOT break the security concept given in the [RFC6733]. In particular, it is reminded here that any command defined or reused in a new Diameter application SHOULD be secured by using TLS [RFC5246] or DTLS/SCTP [RFC6083] and MUST NOT be used without one of TLS, DTLS, or IPsec [RFC4301]. When defining a new Diameter extension, any possible impact of the existing security principles described in the [RFC6733] MUST be carefully appraised and documented in the Diameter application specification.

10. Contributors

The content of this document was influenced by a design team created to revisit the Diameter extensibility rules. The team was formed in February 2008 and finished its work in June 2008. Except the authors, the design team members were:

- Avi Lior
- Glen Zorn
- Jari Arkko
- Jouni Korhonen
- Mark Jones
- Tolga Asveren
- Glenn McGregor
- Dave Frascone

We would like to thank Tolga Asveren, Glenn McGregor, and John Loughney for their contributions as co-authors to earlier versions of this document.

11. Acknowledgments

We greatly appreciate the insight provided by Diameter implementers who have highlighted the issues and concerns being addressed by this document. The authors would also like to thank Jean Mahoney, Ben Campbell, Sebastien Decugis and Benoit Claise for their invaluable detailed reviews and comments on this document.

12. References

12.1. Normative References

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

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

12.2. Informative References

[Q.3303.3] 3rd Generation Partnership Project, "ITU-T Recommendation Q.3303.3, "Resource control protocol no. 3 (rcp3): Protocol at the Rw interface between the Policy Decision Physical Entity (PD-PE) and the Policy Enforcement Physical Entity (PE-PE): Diameter"", 2008.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", RFC 4005, August 2005.

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4740] Garcia-Martin, M., Belinchon, M., Pallares-Lopez, M., Canales-Valenzuela, C., and K. Tammi, "Diameter Session Initiation Protocol (SIP) Application", RFC 4740, November 2006.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5447] Korhonen, J., Bournelle, J., Tschofenig, H., Perkins, C., and K. Chowdhury, "Diameter Mobile IPv6: Support for Network Access Server to Diameter Server Interaction", RFC 5447, February 2009.

[RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., and A. Lior, "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", RFC 5777, February 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011.

[RFC6735] Carlberg, K. and T. Taylor, "Diameter Priority Attribute-Value Pairs", RFC 6735, October 2012.

[RFC7075] Tsou, T., Hao, R., and T. Taylor, "Realm-Based Redirection In Diameter", RFC 7075, November 2013.

[RFC7155] Zorn, G., "Diameter Network Access Server Application", RFC 7155, April 2014.

[TS29.228]
3rd Generation Partnership Project, "3GPP TS 29.228; Technical Specification Group Core Network and Terminals; IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents", <http://www.3gpp.org/ftp/Specs/html-info/29272.htm>.

[TS29.229]
3rd Generation Partnership Project, "3GPP TS 29.229; Technical Specification Group Core Network and Terminals; Cx and Dx interfaces based on the Diameter protocol; Protocol details", <http://www.3gpp.org/ftp/Specs/html-info/29229.htm>.

[TS29.328]
3rd Generation Partnership Project, "3GPP TS 29.328; Technical Specification Group Core Network and Terminals; IP Multimedia (IM) Subsystem Sh interface; signalling flows and message content", <http://www.3gpp.org/ftp/Specs/html-info/29328.htm>.

[TS29.329]
3rd Generation Partnership Project, "3GPP TS 29.329; Technical Specification Group Core Network and Terminals; Sh Interface based on the Diameter protocol; Protocol details", <http://www.3gpp.org/ftp/Specs/html-info/29329.htm>.

Authors' Addresses

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

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

Victor Fajardo
Fluke Networks

Email: vf0213@gmail.com

Hannes Tschofenig
Hall in Tirol 6060
Austria

Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended Status: Proposed Standard
Expires: April 30, 2014
L. Bertz
S. Manning
Sprint
B. Hirschman
October 2014

Diameter Congestion and Filter Attributes
draft-ietf-dime-congestion-flow-attributes-01.txt

Abstract

This document defines optional ECN and filter related attributes that can be used for improved traffic identification, support of ECN and minimized filter administration within Diameter.

RFC 5777 defines a Filter-Rule AVP that accommodates extensions for classification, conditions and actions. It does not support traffic identification for packets using Explicit Congestion Notification as defined in RFC 3168 and does not provide specific actions when the flow(s) described by the Filter-Rule are congested.

A Filter-Rule can describe multiple flows but not the exact number of flows. Flow count and other associated data (e.g. packets) is not captured in Accounting applications, leaving administrators without useful information regarding the effectiveness or understanding of the filter definition.

These optional attributes are forward and backwards compatible with RFC 5777.

Copyright Notice

Table of Contents

1. Introduction

Two optional Explicit Congestion Notification (ECN) [RFC3168] related AVPs are specified in the document. The first AVP provides direct support for ECN [RFC3168] in the IP header and the second AVP provides the ability to define alternate traffic treatment when congestion is experienced.

This document also defines two optional AVPs, Flow-Count and Packet-Count, used for conveying flow information within the Diameter protocol [RFC6733]. These AVPs were found to be useful for a wide range of applications. The AVPs provide a way to convey information of the group of flows described by the Filter-Rule, IPFilterRule or other Diameter traffic filters.

The semantics and encoding of all AVPs can be found in Section 3.

Such AVPs are, for example, needed by some ECN applications to determine the number of flows congested or used by administrators to determine the impact of filter definitions.

Additional parameters may be defined in future documents as the need arises. All parameters are defined as Diameter-encoded Attribute Value Pairs (AVPs), which are described using a modified version of the Augmented Backus-Naur Form (ABNF), see [RFC6733]. The data types are also taken from [RFC6733].

2. Terminology and Abbreviations

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 RFC2119 [RFC2119].

3. ECN-IP-Codepoint, Congestion-Treatment and Filter Attributes

3.1. ECN-IP-Codepoint AVP

The ECN-IP-Codepoint AVP (AVP Code TBD) is of type Enumerated and specifies the Explicit Congestion Notification codepoint values to match in the IP header.

Value	Binary	Keyword	References
0	00	Non-ECT (Not ECN-Capable Transport)	[RFC3168]
1	01	ECT(1) (ECN-Capable Transport)	[RFC3168]
2	10	ECT(0) (ECN-Capable Transport)	[RFC3168]
3	11	CE (Congestion Experienced)	[RFC3168]

When this AVP is used for classification in the Filter-Rule it MUST be part of Classifier Grouped AVP as defined in RFC5777.

3.2. Congestion-Treatment AVP

The Congestion-Treatment AVP (AVP Code TBD) is of type Grouped and indicates how congested traffic, i.e., traffic that has Explicit Congestion Notification Congestion Experienced marking set or some other administratively defined criteria, is treated. In case the Congestion-Treatment AVP is absent the treatment of the congested traffic is left to the discretion of the node performing QoS treatment.

```
Congestion-Treatment ::= < AVP Header: TBD >
            { Treatment-Action }
            [ QoS-Profile-Template ]
            [ QoS-Parameters ]
          * [ AVP ]
```

Treatment-Action, QoS-Profile-Template and QoS-Parameters are defined in [RFC5777]. The Congestion-Treatment AVP is an action and MUST be an attribute of the Filter-Rule Grouped AVP as defined in RFC5777.

3.3. Flow-Count AVP

The Flow-Count AVP (AVP Code TBD) is of type Unsigned64.

It indicates the number of protocol specific flows. The protocol is determined by the filter (e.g. IPFilterRule, Filter-Id, etc.).

3.4. Packet-Count AVP

The Packet-Count AVP (AVP Code TBD) is of type Unsigned64.

It indicates the number of protocol specific packets. The protocol is determined by the filter (e.g. IPFilterRule, Filter-Id, etc.).

4. IANA Considerations

4.1. AVP Codes

IANA allocated AVP codes in the IANA-controlled namespace registry specified in Section 11.1.1 of [RFC6733] for the following AVPs that are defined in this document.

AVP	AVP Code	Section Defined	Data Type
ECN-IP-Codepoint	TBD	3.1	Enumerated
Congestion-Treatment	TBD	3.2	Grouped
Flow-Count	TBD	3.3	Unsigned64
Packet-Count	TBD	3.4	Unsinged64

5. Examples

The following examples illustrate the use of the AVPs defined in this draft.

5.1. Classifier Example

The Classifier AVP (AVP Code 511) specified in RFC 5777 is a grouped AVP that consists of a set of attributes that specify how to match a packet. The addition of the ECN-IP-Codepoint is shown here.

```
Classifier ::= < AVP Header: 511 >
               { Classifier-ID }
               [ Protocol ]
               [ Direction ]
               [ ECP-IP-Codepoint ]
             * [ From-Spec ]
             * [ To-Spec ]
             * [ Diffserv-Code-Point ]
               [ Fragmentation-Flag ]
             * [ IP-Option ]
             * [ TCP-Option ]
               [ TCP-Flags ]
             * [ ICMP-Type ]
             * [ ETH-Option ]
             * [ AVP ]
```

Setting the ECP-IP-Codepoint value to 'CE' would permit the capture of CE flags in the Flow.

Another Classifier with the ECP-IP-Codepoint value of 'ECT' could be specified and, when coupled with the Flow-Count AVP, reports the number of ECT capable flows.

5.2. Diameter Credit Control (CC) with Congestion Information

Diameter nodes using Charge Control can use the Congestion-Treatment AVP to trigger specific actions when congestion occurs. This is similar to the Excess-Treatment Action. The ability to detect when congestion occurs is specific to the AVPs in the Filter-Rule and Diameter Client and is no different than how 'Excess' can be determined for Excess-Treatment. If Excess-Treatment or Congestion-Treatment has occurred Diameter Clients may autonomously send CCRs during the Service Delivery session as interim events. This is shown in Figure 1.

```
                              Service Element
        End User            (CC Client)                        CC Server
           |                     |                                  |
           |(1) Service Request  |                                  |
           |-------------------->|                                  |
           |                     |(2) CCR (Initial,                 |
           |                     |    QoS-Resources(QoS-Desired))   |
           |                     |--------------------------------->|
           |                     |(3) CCA (Granted-Units,           |
           |                     |    QoS-Resources(QoS-Authorized))|
           |                     |<---------------------------------|
           |(4) Service Delivery |                                  |
           |<------------------->|                                  |
           |              (5) Congestion Detected                   |
           |              (6) Congestion Treatment Occurs           |
           |                     |(7) CCR (Termination, Used-Units, |
           |                     |    Flow-Count, Packet-Count,     |
           |                     |    QoS-Resources(QoS-Delivered)) |
           |                     |--------------------------------->|
           |                     |(8) CCA                           |
           |                     |<-------------------------------->|
           |                     |                                  |
           |                     |                                  |
           |(9) End of Service   |                                  |
           |-------------------->|                                  |
           |                     |(10)CCR (Termination, Used-Units, |
           |                     |    Flow-Count, Packet-Count,     |
           |                     |    QoS-Resources(QoS-Delivered)) |
           |                     |--------------------------------->|
           |                     |(11) CCA                          |
           |                     |<---------------------------------|
```

Figure 1: Example of a Diameter Credit Control with Congestion Information

The 'Used-Service-Units' described in RFC5777 examples is customarily a Service-Units, Time-Units or Byte-Count AVP. This is insufficient to represent network state and does not differentiate between throughput and good-put (good or quality throughput) even though the filters may imply good or poor throughput.

Flow-Count and Packet-Count AVPs defined in this document could be sent with a CCR when the triggering event is related to Congestion-Treatment. This provides the CC Server with a better view of the type of congested traffic for improved decision making and Charging. Sending such AVPs under any condition permits rudimentary traffic profiling regardless of network conditions. For instance, low byte per packet counts is indicative of web traffic and high byte counts per packet with a small number of flows may be indicative of video traffic. Enriched reporting described here provides relief from Deep Packet Inspection load and loss of information as traffic becomes increasingly encrypted.

Some services, e.g. Streaming Services, limit the number of flows, Flow-Count, as opposed to other Units, i.e. Byte-Count. In such a case the Flow-Count AVP may be used in place of Service-Units.

6. Security Considerations

The document does not raise any new security concerns. This document describes an extension of RFC5777 that introduces a new filter parameter applied to ECN as defined by [RFC3168]. It also defines a new Grouped AVP that expresses what action to take should congestion be detected. The Grouped AVP reuses attributes defined in RFC5777.

The security considerations of the Diameter protocol itself have been discussed in RFC 6733 [RFC6733]. Use of the AVPs defined in this document MUST take into consideration the security issues and requirements of the Diameter base protocol.

7. Acknowledgements

We would like to thank Avi Lior for his guidance and feedback during the development of this specification.

8. References

8.1. Normative References

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

[RFC3168] Black, D., Floyd, S., and K. Ramakrishnan, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

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

[RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Lior, A. and Jones, M. Ed., "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", RFC 5777, February 2010.

Authors' Addresses

Lyle Bertz
Sprint
6220 Sprint Parkway
Overland Park, KS 66251
United States

EMail: lyleb551144@gmail.com

Serge Manning
Sprint
6220 Sprint Parkway
Overland Park, KS 66251
United States

EMail: sergem913@gmail.com

Brent Hirschman

EMail: Brent.Hirschman@gmail.com

Diameter Maintenance and Extensions (DIME) M. Jones
Internet-Draft
Intended status: Standards Track M. Liebsch
Expires: January 5, 2015
L. Morand

July 4, 2014

Diameter Group Signaling
draft-ietf-dime-group-signaling-04.txt

Abstract

In large network deployments, a single Diameter peer can support over a million concurrent Diameter sessions. Recent use cases have revealed the need for Diameter peers to apply the same operation to a large group of Diameter sessions concurrently. The Diameter base protocol commands operate on a single session so these use cases could result in many thousands of command exchanges to enforce the same operation on each session in the group. In order to reduce signaling, it would be desirable to enable bulk operations on all (or part of) the sessions managed by a Diameter peer using a single or a few command exchanges. This document specifies the Diameter protocol extensions to achieve this signaling optimization.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In large network deployments, a single Diameter peer can support over a million concurrent Diameter sessions. Recent use cases have revealed the need for Diameter peers to apply the same operation to a large group of Diameter sessions concurrently. For example, a policy decision point may need to modify the authorized quality of service for all active users having the same type of subscription. The Diameter base protocol commands operate on a single session so these use cases could result in many thousands of command exchanges to enforce the same operation on each session in the group. In order to reduce signaling, it would be desirable to enable bulk operations on all (or part of) the sessions managed by a Diameter peer using a single or a few command exchanges.

This document describes mechanisms for grouping Diameter sessions and applying Diameter commands, such as performing re-authentication, re-authorization, termination and abortion of sessions to a group of sessions. This document does not define a new Diameter application. Instead it defines mechanisms, commands and AVPs that may be used by any Diameter application that requires management of groups of sessions.

These mechanisms take the following design goals and features into account:

o Minimal impact to existing applications

o Extension of existing commands' Command Code Format (CCF) with optional AVPs to enable grouping and group operations

o Fallback to single session operation

o Implicit discovery of capability to support grouping and group operations in case no external mechanism is available to discover a Diameter peer's capability to support session grouping and session group operations.

2. Terminology

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].

This document uses terminology defined [RFC6733].

3. Protocol Overview

3.1. Building and Modifying Session Groups

Client and Server can assign a new Diameter session to a group, e.g. in case the subscription profile of the associated user has similar characteristics as the profile of other users whose Diameter session has been assigned to one or multiple groups. A single command can be issued and applied to all sessions associated with such group(s), e.g. to adjust common profile or policy settings.

The assignment of a Diameter session to a group can be changed mid-session. For example, if a user's subscription profile changes mid-session, a Diameter peer may remove the session from its current group and assign the session to a different group that is more appropriate for the new subscription profile.

In case of mobile users, the user's session may get transferred to a new Diameter client during handover and assigned to a different group, which is maintained at the new Diameter client, mid-session.

A session group, which has sessions assigned, can be deleted, e.g. due to a change in multiple users' subscription profile so that the group's assigned sessions do not share certain characteristics anymore. Deletion of such group requires subsequent individual treatment of each of the assigned sessions. A peer may decide to assign some of these sessions to any other existing or new group.

3.2. Issuing Group Commands

Changes in the network condition may result in the Diameter server's decision to close all sessions in a given group. The server issues a single Session Termination Request (STR) command , identifying the group of sessions which are to be terminated. The Diameter client treats the STR as group command and initiates termination of all sessions associated with the identified group. Subsequently, the client confirms successful termination of these sessions to the server by sending a single Session Termination Answer (STA) command, which includes the identifier of the group.

3.3. Permission Considerations

Permission considerations in the context of this draft apply to the permission of Diameter nodes to build new session groups, to assign/ remove a session to/from a session group and to delete an existing session group.

This specification follows the most flexible model where both, a Diameter client and a Diameter server can create a new group and assign a new identifier to that session group. When a Diameter node decides to create a new session group, e.g. to group all sessions which share certain characteristics, the node builds a session group identifier according to the rules described in Section 7.3) and becomes the owner of the group. This specification does not constrain the permission to add or remove a session to/from a session group to the group owner, instead each peer can add a session to any known group or remove a session from a group. A session group is deleted and its identifier released after the last session has been removed from the session group. Also the modification of groups in terms of moving a session from one session group to a different session group is permitted to any Diameter node. A Diameter peer can delete a session group and its group identifier mid-session, resulting in individual treatment of the sessions which have been previously assigned to the deleted group.

The enforcement of more constrained permissions is left to the specification of a particular group signaling enabled Diameter application and compliant implementations of such application must enforce the associated permission model. Details about enforcing a more constraint permission model are out of scope of this specification. For example, a more constrained model could require that a client MUST NOT remove a session from a group which is owned by the server.

The following table depicts the permission considerations as per the present specification:

Operation	Server	Client
Create a new Session Group (peer becomes the group owner)	X	X
Assign a Session to an owned Session Group	X	X
Assign a Session to a non-owned Session Group	X	X
Remove a Session from an owned Session Group	X	X
Remove a Session from a non-owned Session Group	X	X
Remove a Session from a Session Group where the peer created the assignment	X	X
Remove a Session from a Session Group where the peer did not create the assignment		
Overrule a peer's group assignment *)		
Delete a Session Group owned by the peer	X	X
Delete a Session Group not owned by the peer		

Default Permission as per this Specification

*) Editors' note: The protocol specification in this document does not consider overruling a peer's assignment of a session to a session group. Group signaling enabled applications may take such protocol support and associated protocol semantics into account in their specification.

4. Protocol Description

4.1. Session Grouping

Either Diameter peer can initiate the assignment of a session to a single or multiple session groups. Modification of a group by removing or adding a single or multiple user sessions can be initiated and performed mid-session by either Diameter peer. Diameter AAA applications typically assign client and server roles to the Diameter peers, which are referred to as relevant Diameter peers to utilize session grouping and issue group commands. Section 5 describes particularities about session grouping and performing group commands when relay agents or proxies are deployed.

Diameter peers, which are group-aware, must store and maintain an entry about the group assignment together with a session’s state. A list of all known session groups should be locally maintained on each peer, each group pointing to individual sessions being assigned to the group. A peer must also keep a record about sessions, which have been assigned to a session group by that peer.

4.1.1. Group assignment at session initiation

To assign a session to a group at session initiation, a Diameter client sends a service specific request, e.g. NASREQ AAR [RFC4005], containing one or more group identifiers. Each of these groups need to be identified by a unique Session-Group-Id contained in a separate Session-Group-Info AVP as specified in Section 7.

The client may choose one or multiple sessions from a list of existing session groups. Alternatively, the client may decide to create a new group and identify itself in the DiameterIdentity element of the Group-Session-Id AVP as per Section 7.3

The client MUST set the SESSION_GROUP_ALLOCATION_ACTION of the Session-Group-Control-Vector AVP in each appended Session-Group-Info AVP to indicate that the identified session should be assigned to the identified session group.

If the Diameter server receives a command request from a Diameter client and the command comprises at least one Session-Group-Info AVP having the SESSION_GROUP_ALLOCATION_ACTION flag of the Session-Group-Control-Vector AVP set, the server must assign the new session to each of the one or multiple identified session groups. In case one or multiple identified session groups are not know to the server, the server must add the one or multiple new groups to its local list of known session groups. When sending the response to the client, e.g. a service-specific auth response as per NASREQ AAA [RFC4005], the server must include all Session-Group-Info AVPs as received in the client’s request.

In addition to the one or multiple session groups identified in the client’s request, the server may decide to assign the new session to one or multiple additional groups. In such case, the server adds to the response additional Session-Group-Info AVPs, each identifying a session group, to which the server has assigned the new session. Each of the Session-Group-Info AVP added by the server must have the SESSION_GROUP_ALLOCATION_ACTION flag of the Session-Group-Control-Vector AVP set.

If the Diameter client receives a response to its previously issued request from the server and the response comprises at least one Session-Group-Info AVP having the SESSION_GROUP_ALLOCATION_ACTION flag of the associated Session-Group-Control-Vector AVP set, the client must add the new session to all session groups as identified in the one or multiple Session-Group-Info AVPs.

A Diameter server receiving a command for session initiation which includes at least one Session-Group-Info AVP but the server does not understand the semantics of this optional AVP because it does not support group operations according to the specification in this document, MUST ignore the optional group operations specific AVPs and proceed with processing the command for a single session.

A Diameter client, which sent a request for session initiation to a Diameter server and appended a single or multiple Session-Group-Id AVPs but cannot find any Session-Group-Info AVP in the associated response from the Diameter server proceeds with processing the command for a single session. Furthermore, the client keeps a log to remember that the server is not able to perform group operations.

4.1.2. Removing a session from a session group

When a Diameter client decides to remove a session from a particular session group, the client sends a service-specific re-authorization request to the server and adds one Session-Group-Info AVP to the request for each session group, from which the client wants to remove the session. The session, which is to be removed from a group, is identified in the Session-Id AVP of the command request. The SESSION_GROUP_ALLOCATION_ACTION flag of the Session-Group-Control-Vector AVP in each Session-Group-Info AVP must be cleared to indicate removal of the session from the session group identified in the associated Session-Group-id AVP.

When a Diameter client decides to remove a session from all session groups, to which the session has been previously assigned, the client sends a service-specific re-authorization request to the server and adds a single Session-Group-Info AVP to the request which has the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP omitted. The session, which is to be removed from all groups, to which the session has been previously assigned, is identified in the Session-Id AVP of the command request.

If the Diameter server receives a request from the client which has at least one Session-Group-Info AVP appended with the SESSION_GROUP_ALLOCATION_ACTION flag cleared, the server must remove the session from the session group identified in the associated Session-Group-Id AVP. If the request comprises at least one Session-

Group-info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and no Session-Id AVP present, the server must remove the session from all session groups to which the session has been previously assigned. The server must include in its response to the requesting client all Session-Group-Id AVPs as received in the request.

When the Diameter server decides to remove a session from one or multiple particular session groups or from all session groups to which the session has been assigned beforehand, the server sends a Re-Authorization Request (RAR) to the client, indicating the session in the requests Session-Id AVP. The client sends a Re-Authorization Answer (RAA) to respond to the server's request. The client subsequently sends service-specific re-authorization request containing one or multiple Session-Group-Info AVPs, each indicating a session group, to which the session had been previously assigned. To indicate removal of the indicated session from one or multiple session groups, the server sends a service-specific auth response to the client, containing a list of Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP identifying the session group, from which the session should be removed. The server MAY include to the service-specific auth response a list of Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying session groups to which the session remains subscribed. In case the server decides to remove the identified session from all session groups, to which the session has been previously assigned, the server includes in the service-specific auth response at least one Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and Session-Group-Id AVP absent.

4.1.3. Mid-session group assignment modifications

Either Diameter peer can modify the group membership of an active Diameter session according to the specified permission considerations.

To update an assigned group mid-session, a Diameter client sends a service-specific re-authorization request to the server, containing one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP present, identifying the session group to which the session should be assigned. With the same message, the client may send one or multiple Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP identifying the session group from which the identified session is to be removed. To remove the session from all previously assigned session groups, the client includes at least one Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and no Session-Group-Id

AVP present. When the server received the service-specific re-authorization request, it must update its locally maintained view of the session groups for the identified session according to the appended Session-Group-Info AVPs. The server sends a service-specific auth response to the client containing one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying the new session group to which the identified session has been assigned.

When a Diameter server enforces an update to the assigned groups mid-session, it sends a Re-Authorization Request (RAR) message to the client identifying the session, for which the session group lists are to be updated. The client responds with a Re-Authorization Answer (RAA) message. The client subsequently sends service-specific re-authorization request containing one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying the session group to which the session had been previously assigned. The server responds with a service-specific auth response and includes one or multiple Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying the session group, to which the identified session is to be assigned. With the same response message, the server may send one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP identifying the session groups from which the identified session is to be removed. When server wants to remove the session from all previously assigned session groups, it send at least on Session-Group-Info AVP with the response having the SESSION_GROUP_ALLOCATION_ACTION flag cleared and no Session-Group-Id AVP present.

4.2. Session Grouping Capability Discovery

Diameter nodes should assign a session to a session group and perform session group operations with a peer only after having ensured that the peer announced associated support beforehand.

4.2.1. Explicit Capability Discovery

New Diameter applications may consider support for Diameter session grouping and for performing group commands during the standardization process. Such applications provide intrinsic discovery for the support of group commands and announce this capability through the assigned application ID.

System- and deployment-specific means for capability exchange can be used to announce peers' support for session grouping and session group operations. In such case, the optional Session-Group-

Capability-Vector AVP, as described in Section 4.2.2 can be omitted in Diameter messages being exchanged between peers.

4.2.2. Implicit Capability Discovery

If no explicit mechanism for capability discovery is deployed to enable Diameter nodes to learn about peers' capability to support session grouping and group commands, Diameter peers SHOULD append the Session-Group-Capability-Vector AVP to any Diameter messages exchanged with its peers to announce its capability to support session grouping and session group operations. Implementations following this specification set the BASE_SESSION_GROUP_CAPABILITY flag of the Session-Group-Capability-Vector AVP.

When a Diameter node receives at least one Session-Group-Capability-Vector AVP from a peer with the BASE_SESSION_GROUP_CAPABILITY flag set, the Diameter node maintains a log to remember the peer's capability to support group commands.

4.3. Deleting a Session Group

To delete a session group and release the associated Session-Group-Id value, the owner of a session group appends a single Session-Group-Info AVP having the SESSION_GROUP_STATUS_IND flag cleared and the Session-Group-Id AVP identifying the session group, which is to be deleted. The SESSION_GROUP_ALLOCATION_ACTION flag of the associated Session-Group-Control-Vector AVP MUST be cleared.

4.4. Performing Group Operations

4.4.1. Sending Group Commands

Either Diameter peer can request the recipient of a request to process an associated command for all sessions being assigned to one or multiple groups by identifying these groups in the request. The sender of the request appends for each group, to which the command applies, a Session-Group-Info AVP including the Session-Group-Id AVP to identify the associated session group. Both, the SESSION_GROUP_ALLOCATION_ACTION flag as well as the SESSION_GROUP_STATUS_IND flag must be set.

If the CCF of the request mandates a Session-Id AVP, the Session-Id AVP MUST identify a single session which is assigned to at least one of the groups being identified in the appended Session-Group-Id AVPs.

The sender of the request MUST indicate to the receiver how follow up message exchanges should be performed by appending a single instance of the Group-Response-Action AVP. Even if the request includes multiple instances of a Session-Group-Info AVP, the request MUST NOT comprise more than a single instance of a Group-Response-Action AVP. If the sender wants the receiver to perform follow up exchanges with a single command for all impacted groups, the sender sets the value of the Group-Response-Action AVP to ALL_GROUPS (1). If follow up message exchanges should be performed on a per-group basis in case multiple groups are identified in the group command, the value of the Group-Response-Action AVP is set to PER_GROUP (2). A value set to PER_SESSION (3) indicates to the receiver that all follow up exchanges should be performed using a single message for each impacted session.

If the sender wants the receiver of the request to process the associated command solely for a single session does not append any group identifier, but identifies the relevant session in the Session-Id AVP.

4.4.2. Receiving Group Commands

A Diameter peer receiving a request to process a command for a group of sessions identifies the relevant groups according to the appended Session-Group-Id AVP in the Session-Group-Info AVP and processes the group command according to the appended Group-Response-Action AVP . If the received request identifies multiple groups in multiple appended Session-Group-Id AVPs, the receiver should process the associated command for each of these groups. if a session has been assigned to more than one of the identified groups, the receiver must process the associated command only once per session.

The Diameter peer receiving a request which requests performing the command to at least on session group SHOULD perform follow up message exchanges according to the value identified in the Session-Group-Info AVP.

4.4.3. Error Handling for Group Commands

When a Diameter peer receives a request to process a command for one or more session groups and the result of processing the command is an error that applies to all sessions in the identified groups, an associated protocol error must be returned to the source of the request. In such case, the sender of the request MUST fall back to single-session processing and the session groups, which have been identified in the group command, MUST be deleted according to the procedure described in Section 4.3.

When a Diameter peer receives a request to process a command for one or more session groups and the result of processing the command succeeds for some sessions identified in one or multiple session groups, but fails for one or more sessions, the Result-Code AVP in the response message SHOULD indicate DIAMETER_LIMITED_SUCCESS as per Section 7.1.2 of [RFC6733]. In case of limited success, the sessions, for which the processing of the group command failed, MUST be identified using a Failed-AVP AVP as per Session 7.5 of [RFC6733].

4.4.4. Single-Session Fallback

Either Diameter peer, a Diameter client or a Diameter server, can fall back to single session operation by ignoring and omitting the optional group session-specific AVPs. Fallback to single-session operation is performed by processing the Diameter command solely for the session identified in the mandatory Session-Id AVP. The response to the group command must not identify any group but identify solely the single session for which the command has been processed.

5. Operation with Proxies Agents

This specification assumes in case of a present stateful Proxy Agent between a Diameter client and a Diameter server that the Proxy Agent is aware of session groups and session group handling. The Proxy MUST reflect the state of each session associated with a session group according to the result of a group command operated between a Diameter client and a server.

In case a Proxy Agent manipulates session groups, it MUST maintain consistency of session groups between a client and a server. This applies to deployment where the Proxy Agent utilizes session grouping and performing group commands with, for example, a Diameter server, whereas the Diameter client is not group-aware. The same applies to deployment where all nodes, the Diameter client and server, as well as the Proxy Agent are group-aware but the Proxy Agent manipulates groups, e.g. to adopt different administrative policies that apply to the client’s domain and the server’s domain.

6. Commands Formatting

This document does not specify new Diameter commands to enable group operations, but relies on command extensibility capability provided by the Diameter Base protocol. This section provides the guidelines to extend the CCF of existing Diameter commands with optional AVPs to enable the recipient of the command to perform the command to all sessions associated with the identified group(s).

6.1. Formatting Example: Group Re-Auth-Request

A request that one or more groups of users are re-authentication is issued by appending one or multiple Session-Group-Id AVP(s) to the Re-Auth-Request (RAR) and a single instance of a Group-Response-Action AVP. The one or multiple Session-Group-Id AVP(s) identify the associated group(s) for which the group re-authentication has been requested. The Group-Response-Action AVP identifies the expected means to perform and respond to the group command. The recipient of the group command initiates re-authentication for all users associated with the identified group(s). Furthermore, the sender of the group re-authentication request appends a Group-Response-Action AVP to provide more information to the receiver of the command about how to accomplish the group operation.

The value of the mandatory Session-Id AVP MUST identify a session associated with a single user, which is assigned to at least one of the groups being identified in the appended Session-Group-Id AVPs.

```
      <RAR>  ::= < Diameter Header: 258, REQ, PXY >
                 < Session-Id >
                 { Origin-Host }
                 { Origin-Realm }
                 { Destination-Realm }
                 { Destination-Host }
                 { Auth-Application-Id }
                 { Re-Auth-Request-Type }
                 [ User-Name ]
                 [ Origin-State-Id ]
               * [ Proxy-Info ]
               * [ Route-Record ]
                 [ Session-Group-Capability-Vector ]
               * [ Session-Group-Info ]
                 [ Group-Response-Action ]
               * [ AVP ]
```

7. Attribute-Value-Pairs (AVP)

Attribute Name	AVP Code	Value Type	AVP Flag rules: MUST	MAY	SHOULD NOT	MUST NOT
Session-Group-Info	TBD1	Grouped		P		V
Session-Group-Control-Vector	TBD2	Unsigned32		P		V
Session-Group-Id	TBD3	OctetString		P		V
Group-Response-Action	TBD4	Unsigned32		P		V
Session-Group-Capability-Vector	TBD5	Unsigned32		P		V

AVPs for the Diameter Group Signaling

7.1. Session-Group-Info AVP

The Session-Group-Info AVP (AVP Code TBD1) is of type Grouped. It contains the identifier of the session group as well as an indication of the node responsible for session group identifier assignment.

```
Session-Group-Info ::= < AVP Header: TBD1 >
                       < Session-Group-Control-Vector >
                       [ Session-Group-Id ]
                     * [ AVP ]
```

7.2. Session-Group-Control-Vector AVP

The Session-Group-Control-Vector AVP (AVP Code TBD2) is of type Unsigned32 and contains a 32-bit flags field to control the group assignment at session-group aware nodes.

The following capabilities are defined in this document:

SESSION_GROUP_ALLOCATION_ACTION (0x00000001)

This flag indicates the action to be performed for the identified session. When this flag is set, it indicates that the identified Diameter session is to be assigned to the session group as identified by the Session-Group-Id AVP or the session's assignment to the session group identified in the Session-Group-Id AVP is still valid. When the flag is cleared, the identified Diameter session is to be removed from at least one session group. When the flag is cleared and the Session-Group-Info AVP identifies a particular session group in the associated Session-Group-Id AVP, the session is to be removed solely from the identified session group. When the flag is cleared and the Session-Group-Info AVP does not identify a particular session group (Session-Group-Id AVP is absent), the identified Diameter session is to be removed from all session groups, to which it has been previously assigned.

SESSION_GROUP_STATUS_IND (0x00000010)

This flag indicates the status of the session group identified in the associated Session-Group-Id AVP. The flag is set when the identified session group has just been created or is still active. If the flag is cleared, the identified session group is deleted and the associated Session-Group-Id is released. If the Session-Group-Info AVP does not comprise a Session-Group-Id AVP, this flag is meaningless and MUST be ignored by the receiver.

7.3. Session-Group-Id AVP

The Session-Group-Id AVP (AVP Code TBD3) is of type UTF8String and identifies a group of Diameter sessions.

The Session-Group-Id MUST be globally and eternally unique, as it is meant to uniquely identify a group of Diameter sessions without reference to any other information.

The default format of the Session-Group-id MUST comply to the format recommended for a Session-Id, as defined in the section 8.8 of the [RFC6733]. The DiameterIdentity element of the Session-Group-Id MUST identify the Diameter node, which owns the session group.

7.4. Group-Response-Action AVP

The Group-Response-Action AVP (AVP Code TBD4) is of type Unsigned32 and contains a 32-bit address space representing values indicating how the peer SHOULD issue follow up exchanges in response to a command which impacts multiple sessions. The following values are defined by this application:

ALL_GROUPS (1)
Follow up exchanges should be performed with a single message exchange for all impacted groups.

PER_GROUP (2)
Follow up exchanges should be performed with a message exchange for each impacted group.

PER_SESSION (3)
Follow up exchanges should be performed with a message exchange for each impacted session.

7.5. Session-Group-Capability-Vector AVP

The Session-Group-Capability-Vector AVP (AVP Code TBD5) is of type Unsigned32 and contains a 32-bit flags field to indicate capabilities in the context of session-group assignment and group operations.

The following capabilities are defined in this document:

BASE_SESSION_GROUP_CAPABILITY (0x00000001)

This flag indicates the capability to support session grouping and session group operations according to this specification.

8. Result-Code AVP Values

This document does not define new Result-Code [RFC6733] values for existing applications, which are extended to support group commands. Specification documents of new applications, which will have intrinsic support for group commands, may specify new Result-Codes.

9. IANA Considerations

This section contains the namespaces that have either been created in this specification or had their values assigned to existing namespaces managed by IANA.

9.1. AVP Codes

This specification requires IANA to register the following new AVPs from the AVP Code namespace defined in [RFC6733].

- Session-Group-Info
- Session-Group-Control-Vector
- Session-Group-Id
- Group-Response-Action
- Session-Group-Capability-Vector

The AVPs are defined in Section 7.

10. Security Considerations

TODO

11. Acknowledgments

The authors of this document want to thank Ben Campbell and Eric McMurry for their valuable comments to early versions of this draft.

12. Normative References

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

[RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", RFC 4005, August 2005.

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

Appendix A. Session Management -- Exemplary Session State Machines

A.1. Authorization Session State Machine

Section 8.1 in [RFC6733] defines a set of finite state machines, representing the life cycle of Diameter sessions, and which MUST be observed by all Diameter implementations that make use of the authentication and/or authorization portion of a Diameter application. This section defines the additional state transitions related to the processing of the new commands which may impact multiple sessions.

The group membership is session state and therefore only those state machines from [RFC6733] in which the server is maintaining session state are relevant in this document. As in [RFC6733], the term Service-Specific below refers to a message defined in a Diameter application (e.g., Mobile IPv4, NASREQ).

The following state machine is observed by a client when state is maintained on the server. State transitions which are unmodified from [RFC6733] are not repeated here.

A Diameter group command in the following tables is differentiated from a single-session related command by a preceding 'G'. A Group Re-Auth Request, which applies to one or multiple session groups, has been exemplarily described in Section 6.1. Such Group RAR command is denoted as 'GRAR' in the following table. The same notation applies to other commands as per [RFC6733].

```
                          CLIENT, STATEFUL
   State     Event                            Action       New State
```

Idle	Client or Device Requests access	Send service specific auth req optionally including groups	Pending
Open	GASR received with Group-Response-Action = ALL_GROUPS, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send GSTR.	Discon
Open	GASR received with Group-Response-Action = PER_GROUPS, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send GSTR per group	Discon
Open	GASR received with Group-Response-Action = PER_SESSION, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send STR per session	Discon
Open	GASR received, client will not comply with request to end all session in received group(s)	Send GASA with Result-Code != SUCCESS	Open
Discon	GSTA Received	Discon. user/device	Idle
Open	GRAR received with Group-Response-Action = ALL_GROUPS, session is assigned to received group(s) and client will perform subsequent re-auth	Send GRAA, Send service specific group re-auth req	Pending

Open	GRAR received with Group-Response-Action = PER_GROUP, session is assigned to received group(s) and client will perform subsequent re-auth	Send GRAA, Send service specific group re-auth req per group	Pending
Open	GRAR received with Group-Response-Action = PER_SESSION, session is assigned to received group(s) and client will perform subsequent re-auth	Send GRAA, Send service specific re-auth req per session	Pending
Open	GRAR received and client will not perform subsequent re-auth	Send GRAA with Result-Code != SUCCESS, Discon. user/device	Idle
Pending	Successful service-specific group re-authorization answer received.	Provide service	Open
Pending	Failed service-specific group re-authorization answer received.	Discon. user/device	Idle

The following state machine is observed by a server when it is maintaining state for the session. State transitions which are unmodified from [RFC6733] are not repeated here.

SERVER, STATEFUL

State	Event	Action	New State
Idle	Service-specific authorization request received, and user is authorized	Send successful service specific answer optionally including groups	Open
Open	Server wants to terminate group(s)	Send GASR	Discon
Discon	GASA received	Cleanup	Idle
Any	GSTR received	Send GSTA, Cleanup	Idle
Open	Server wants to reauth group(s)	Send GRAR	Pending
Pending	GRAA received with Result-Code = SUCCESS	Update session(s)	Open
Pending	GRAA received with Result-Code != SUCCESS	Cleanup session(s)	Idle
Open	Service-specific group re-authoization request received and user is authorized	Send successful service specific group re-auth answer	Open
Open	Service-specific group re-authorization request received and user is not authorized	Send failed service specific group re-auth answer, cleanup	Idle

Authors' Addresses

Mark Jones

Email: mark@azu.ca

Marco Liebsch

Email: marco.liebsch@neclab.eu

Lionel Morand

Email: lionel.morand@orange.com

Diameter Maintenance and Extensions (DIME) M. Jones
Internet-Draft
Intended status: Standards Track M. Liebsch
Expires: September 10, 2020
L. Morand
March 9, 2020

Diameter Group Signaling
draft-ietf-dime-group-signaling-13.txt

Abstract

In large network deployments, a single Diameter node can support over a million concurrent Diameter sessions. Recent use cases have revealed the need for Diameter nodes to apply the same operation to a large group of Diameter sessions concurrently. The Diameter base protocol commands operate on a single session so these use cases could result in many thousands of command exchanges to enforce the same operation on each session in the group. In order to reduce signaling, it would be desirable to enable bulk operations on all (or part of) the sessions managed by a Diameter node using a single or a few command exchanges. This document specifies the Diameter protocol extensions to achieve this signaling optimization.

Table of Contents

1. Introduction

In large network deployments, a single Diameter node can support over a million concurrent Diameter sessions. Recent use cases have revealed the need for Diameter nodes to apply the same operation to a large group of Diameter sessions concurrently. For example, a policy decision point may need to modify the authorized quality of service for all active users having the same type of subscription. The Diameter base protocol commands operate on a single session so these use cases could result in many thousands of command exchanges to enforce the same operation on each session in the group. In order to reduce signaling, it would be desirable to enable bulk operations on all (or part of) the sessions managed by a Diameter node using a single or a few command exchanges.

This document describes mechanisms for grouping Diameter sessions and applying Diameter commands, such as performing re-authentication, re-authorization, termination and abortion of sessions to a group of sessions. This document does not define a new Diameter application. Instead it defines mechanisms, commands and AVPs that may be used by any Diameter application that requires management of groups of sessions.

These mechanisms take the following design goals and features into account:

o Minimal impact to existing applications

o Extension of existing commands' Command Code Format (CCF) with optional AVPs to enable grouping and group operations

o Fallback to single session operation

o Implicit discovery of capability to support grouping and group operations in case no external mechanism is available to discover a Diameter peer's capability to support session grouping and session group operations

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology defined in [RFC6733].

3. Protocol Overview

3.1. Building and Modifying Session Groups

Client and Server can assign a new Diameter session to a group, e.g., in case the subscription profile of the associated user has similar characteristics as the profile of other users whose Diameter session has been assigned to one or multiple groups. A single command can be issued and applied to all sessions associated with such group(s), e.g., to adjust common profile or policy settings.

The assignment of a Diameter session to a group can be changed during an ongoing session (mid-session). For example, if a user's subscription profile changes mid-session, a Diameter server may remove a session from an existing group and assign this session to a different group that is more appropriate for the new subscription profile.

In the case of mobile users, the user's session may get transferred mid-session to a new Diameter client during handover and assigned to a different group, which is maintained at the new Diameter client.

A session group, which has sessions assigned, can be deleted, e.g., due to a change in multiple users' subscription profile so that the group's assigned sessions do not share certain characteristics anymore. Deletion of such group requires subsequent individual treatment of each of the assigned sessions. A node may decide to assign some of these sessions to any other existing or new group.

3.2. Issuing Group Commands

Changes in the network condition may result in the Diameter server's decision to close all sessions in a given group. As example, the server issues a single Session Termination Request (STR) command, including the identifier of the group of sessions which are to be terminated. The Diameter client treats the STR as group command and initiates the termination of all sessions associated with the identified group. Subsequently, the client confirms the successful termination of these sessions to the server by sending a single Session Termination Answer (STA) command, which includes the identifier of the group.

3.3. Permission Considerations

Permission considerations in the context of this draft apply to the permission of Diameter nodes to build new session groups, to assign/ remove a session to/from a session group and to delete an existing session group.

This specification follows the most flexible model where both, a Diameter client and a Diameter server can create a new group and assign a new identifier to that session group. When a client or a server decides to create a new session group, e.g., to group all sessions which share certain characteristics, this node builds a session group identifier according to the rules described in Section 7.3 and becomes the owner of the group. This specification does not restrict the permission to add or remove a session to/from a session group to the group owner. Either the client and the server can assign a session to a group. However, a session can be removed from a session group and/or moved to another session group only by the node that has assigned this session to the session group. A session group is deleted and its identifier released after the last session has been removed from this session group. The owner of a session group can delete a session group and its group identifier mid-session, resulting in individual treatment of the sessions which have been previously assigned to the deleted group. A session group must only be deleted by the Diameter node that created it.

Diameter applications with implicit support for session groups MAY define a more constrained permission model. For example, a more constrained model could require that a client must not remove a session from a group which is owned by the server. Details about enforcing a more constraint permission model are out of scope of this specification.

The following table depicts the permission considerations as per the present specification:

Operation	Server	Client
Create a new Session Group (Diameter node becomes the group owner)	X	X
Assign a Session to an owned Session Group	X	X
Assign a Session to a non-owned Session Group	X	X
Remove a Session from an owned Session Group	X	X
Remove a Session from a non-owned Session Group	X	X
Remove a Session from a Session Group where the Diameter node created the assignment	X	X
Remove a Session from a Session Group where the Diameter node did not create the assignment		
Overrule a different Diameter node's group assignment *)		
Delete a Session Group which is owned by the Diameter node	X	X
Delete a Session Group which is not owned by the Diameter node		

Default Permission as per this Specification

4. Protocol Description

4.1. Session Grouping Capability Discovery

Diameter nodes SHOULD NOT perform group operations with peer nodes unless the node has advertised support for session grouping and group operations.

4.1.1. Implicit Capability Discovery

Newly defined Diameter applications may natively support Diameter session grouping and group operations. Such applications provide intrinsic discovery for the support of session grouping capability using the assigned Application Id advertised during the capability exchange phase two Diameter peers establish a transport connection (see Section 5.3 of [RFC6733]).

System- and deployment-specific means, as well as out-of-band mechanisms for capability discovery can be used to announce nodes' support for session grouping and session group operations. In such case, the optional Session-Group-Capability-Vector AVP, as described in Section 4.1.2 can be omitted in Diameter messages being exchanged between nodes.

4.1.2. Explicit Capability Discovery

If no other mechanism for capability discovery is deployed to enable Diameter nodes to learn about nodes' capability to support session grouping and group commands for a given application, a Diameter node SHOULD append the Session-Group-Capability-Vector AVP to any Diameter application messages exchanged with the other Diameter nodes to announce its capability to support session grouping and session group operations for the advertised application. Implementations following the specification as per this document MUST set the BASE_SESSION_GROUP_CAPABILITY flag of the Session-Group-Capability-Vector AVP.

When a Diameter node receives at least one Session-Group-Capability-Vector AVP from a node with the BASE_SESSION_GROUP_CAPABILITY flag set, the receiving Diameter node discovers the supported session grouping capability of the sending Diameter node for the advertised application and MUST cache this information for the lifetime of the routing table entry associated with the peer identity/Application Id pair (see Section 2.7 of [RFC6733]).

4.2. Session Grouping

This specification does not limit the number of session groups to which a single session is assigned. It is left to the implementation of an application to determine such limitations. If an application facilitates a session to belong to multiple session groups, the application MUST maintain consistency of associated application session states for these multiple session groups.

Either Diameter node (client or server) can initiate the assignment of a session to a single or multiple session groups. Modification of a group by removing or adding a single or multiple user sessions can be initiated and performed mid-session by either Diameter node responsible for the session assignment to this group. Diameter AAA applications typically assign client and server roles to the Diameter nodes, which are referred to as relevant Diameter nodes to utilize session grouping and issue group commands. Section 5 describes particularities about session grouping and performing group commands when relay agents or proxies are deployed.

Diameter nodes, which are group-aware, MUST store and maintain an entry about the group assignment together with a session's state. A list of all known session groups is locally maintained on each node, each group pointing to individual sessions being assigned to the group. A Diameter node MUST also keep a record about sessions, which have been assigned to a session group by itself.

4.2.1. Group assignment at session initiation

To assign a session to a group at session initiation, a Diameter client sends a service specific request, e.g., NASREQ AA-Request [RFC7155], containing one or more session group identifiers. Each of these groups MUST be identified by a unique Session-Group-Id contained in a separate Session-Group-Info AVP as specified in Section 7.

The client may choose one or multiple session groups from a list of existing session groups. Alternatively, the client may decide to create a new group to which the session is assigned and identify itself in the <DiameterIdentity> portion of the Session-Group-Id AVP as per Section 7.3. For all assignments of a session to an active session group made by the client or the server, the SESSION_GROUP_STATUS_IND flag in the Session-Group-Info AVP, which identifies the session group, MUST be set. A set SESSION_GROUP_STATUS_IND flag indicates that the identified session group has just been created or is still active.

The client MUST set the SESSION_GROUP_ALLOCATION_ACTION flag of the Session-Group-Control-Vector AVP in each appended Session-Group-Info AVP to indicate that the session contained in the request should be assigned to the identified session group.

The client may also indicate in the request that the server is responsible for the assignment of the session in one or multiple sessions owned by the server. In such a case, the client MUST include the Session-Group-Info AVP in the request including the Session-Group-Control-Vector AVP with the SESSION_GROUP_ALLOCATION_ACTION flag set but no Session-Group-Id AVP.

If the Diameter server receives a command request from a Diameter client and the command includes at least one Session-Group-Info AVP having the SESSION_GROUP_ALLOCATION_ACTION flag in the Session-Group-Control-Vector AVP set, the server can accept or reject the request for group assignment. Reasons for rejection may be e.g., lack of resources for managing additional groups. When rejected, the session MUST NOT be assigned to any session group.

If the Diameter server accepts the client's request for a group assignment, the server MUST assign the new session to each of the one or multiple identified session groups when present in the Session-Group-Info AVP. If one or multiple identified session groups are not already stored by the server, the server MUST store the newly identified group(s) to its local list of known session groups. When sending the response to the client, e.g., a service-specific auth response as per NASREQ AA-Answer [RFC7155], the server MUST include all Session-Group-Info AVPs as received in the client's request.

In addition to the one or multiple session groups identified in the client's request, the server may decide to assign the new session to one or multiple additional groups. In such a case, the server MUST add to the response the additional Session-Group-Info AVPs, each identifying a session group to which the new session is assigned by the server. Each of the Session-Group-Info AVP added by the server MUST have the SESSION_GROUP_ALLOCATION_ACTION flag set in the Session-Group-Control-Vector AVP set.

If the Diameter server rejects the client's request for a group assignment, the server sends the response to the client, e.g., a service-specific auth response as per NASREQ AA-Answer [RFC7155], and MUST include all Session-Group-Info AVPs as received in the client's request (if any) while clearing the SESSION_GROUP_ALLOCATION_ACTION flag of the Session-Group-Control-Vector AVP. The server MAY accept the client's request for the identified session but refuse the session's assignment to any session group. The server sends the response to the client indicating success in the result code. In such case the session is treated as single session without assignment to any session group by the Diameter nodes.

If the assignment of the session to one or some of the multiple identified session groups fails, the session group assignment is treated as failure. In such case the session is treated as single session without assignment to any session group by the Diameter nodes. The server sends the response to the client and MAY include those Session-Group-Info AVPs for which the group assignment failed. The SESSION_GROUP_ALLOCATION_ACTION flag of included Session-Group-Info AVPs MUST be cleared.

If the Diameter server receives a command request from a Diameter client and the command includes a Session-Group-Info AVP which does not include a Session-Group-Id AVP, the server MAY decide to assign the session to one or multiple session groups. For each session group, to which the server assigns the new session, the server includes a Session-Group-Info AVP with the Session-Group-Id AVP identifying a session group in the response sent to the client. Each of the Session-Group-Info AVPs included by the server MUST have the SESSION_GROUP_ALLOCATION_ACTION flag of the Session-Group-Control-Vector AVP set.

If the Diameter server receives a command request from a Diameter client and the command does not contain any Session-Group-Info AVP, the server MUST NOT assign the new session to any session group but treat the request as for a single session. The server MUST NOT return any Session-Group-Info AVP in the command response.

If the Diameter client receives a response to its previously issued request from the server and the response includes at least one Session-Group-Info AVP having the SESSION_GROUP_ALLOCATION_ACTION flag of the associated Session-Group-Control-Vector AVP set, the client MUST add the new session to all session groups as identified in the one or multiple Session-Group-Info AVPs. If the Diameter client fails to add the session to one or more session groups as identified in the one or multiple Session-Group-info AVPs, the client MUST terminate the session. The client MAY send a subsequent request for session initiation to the server without requesting the assignment of the session to a session group

If the Diameter client receives a response to its previously issued request from the server and the one or more Session-Group-Info AVPs have the SESSION_GROUP_ALLOCATION_ACTION flag of the associated Session-Group-Control-Vector AVP cleared, the client MUST terminate the assignment of the session to the one or multiple groups. If the response from the server indicates success in the result code but solely the assignment of the session to a session group has been rejected by the server, the client treats the session as single session without group assignment.

A Diameter client, which sent a request for session initiation to a Diameter server and appended a single or multiple Session-Group-Id AVPs but cannot find any Session-Group-Info AVP in the associated response from the Diameter server proceeds as if the request was processed for a single session. The Diameter client MUST NOT retry to request group assignment for this session, but MAY try to request group assignment for other new sessions.

4.2.2. Removing a session from a session group

When a Diameter client decides to remove a session from a particular session group, the client sends a service-specific re-authorization request to the server and adds one Session-Group-Info AVP to the request for each session group, from which the client wants to remove the session. The session, which is to be removed from a group, is identified in the Session-Id AVP of the command request. The SESSION_GROUP_ALLOCATION_ACTION flag of the Session-Group-Control-Vector AVP in each Session-Group-Info AVP MUST be cleared to indicate removal of the session from the session group identified in the associated Session-Group-id AVP.

When a Diameter client decides to remove a session from all session groups, to which the session has been previously assigned, the client sends a service-specific re-authorization request to the server and adds a single Session-Group-Info AVP to the request which has the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP omitted. The session, which is to be removed from all groups, to which the session has been previously assigned, is identified in the Session-Id AVP of the command request.

If the Diameter server receives a request from the client which has at least one Session-Group-Info AVP appended with the SESSION_GROUP_ALLOCATION_ACTION flag cleared, the server MUST remove the session from the session group identified in the associated Session-Group-Id AVP. If the request includes at least one Session-Group-info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and no Session-Id AVP present, the server MUST remove the session from all session groups to which the session has been previously assigned. The server MUST include in its response to the requesting client all Session-Group-Id AVPs as received in the request.

When the Diameter server decides to remove a session from one or multiple particular session groups or from all session groups to which the session has been assigned beforehand, the server sends a Re-Authorization Request (RAR) or a service-specific server-initiated request to the client, indicating the session in the Session-Id AVP of the request. The client sends a Re-Authorization Answer (RAA) or a service-specific answer to respond to the server's request. The client subsequently sends service-specific re-authorization request containing one or multiple Session-Group-Info AVPs, each indicating a session group, to which the session had been previously assigned. To indicate removal of the indicated session from one or multiple session groups, the server sends a service-specific auth response to the client, containing a list of Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP identifying the session group, from which the session should be removed. The server MAY include to the service-specific auth response a list of Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying session groups to which the session remains subscribed. If the server decides to remove the identified session from all session groups, to which the session has been previously assigned, the server includes in the service-specific auth response at least one Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and Session-Group-Id AVP absent.

4.2.3. Mid-session group assignment modifications

Either Diameter node (client or server) can modify the group membership of an active Diameter session according to the specified permission considerations.

To update an assigned group mid-session, a Diameter client sends a service-specific re-authorization request to the server, containing one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP present, identifying the session group to which the session should be assigned. With the same message, the client may send one or multiple Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP identifying the session group from which the identified session is to be removed. To remove the session from all previously assigned session groups, the client includes at least one Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and no Session-Group-Id AVP present. When the server received the service-specific re-authorization request, it MUST update its locally maintained view of the session groups for the identified session according to the appended Session-Group-Info AVPs. The server sends a service-specific auth response to the client containing one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying the new session group to which the identified session has been assigned.

When a Diameter server enforces an update to the assigned groups mid-session, it sends a Re-Authorization Request (RAR) message or a service-specific request to the client identifying the session, for which the session group lists are to be updated. The client responds with a Re-Authorization Answer (RAA) message or a service-specific answer. The client subsequently sends a service-specific re-authorization request containing one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying the session group to which the session had been previously assigned. The server responds with a service-specific auth response and includes one or multiple Session-Group-Info AVP with the SESSION_GROUP_ALLOCATION_ACTION flag set and the Session-Group-Id AVP identifying the session group, to which the identified session is to be assigned. With the same response message, the server may send one or multiple Session-Group-Info AVPs with the SESSION_GROUP_ALLOCATION_ACTION flag cleared and the Session-Group-Id AVP identifying the session groups from which the identified session is to be removed. When server wants to remove the session from all previously assigned session groups, it sends at least one Session-Group-Info AVP with the response having the SESSION_GROUP_ALLOCATION_ACTION flag cleared and no Session-Group-Id AVP present.

4.3. Deleting a Session Group

To delete a session group and release the associated Session-Group-Id value, the owner of a session group appends a single Session-Group-Info AVP having the SESSION_GROUP_STATUS_IND flag cleared and the Session-Group-Id AVP identifying the session group, which is to be deleted. The SESSION_GROUP_ALLOCATION_ACTION flag of the associated Session-Group-Control-Vector AVP MUST be cleared.

4.4. Performing Group Operations

4.4.1. Sending Group Commands

Either Diameter node (client or server) can request the recipient of a request to process an associated command for all sessions assigned to one or multiple groups by identifying these groups in the request. The sender of the request appends for each group, to which the command applies, a Session-Group-Info AVP including the Session-Group-Id AVP to identify the associated session group. Both, the SESSION_GROUP_ALLOCATION_ACTION flag as well as the SESSION_GROUP_STATUS_IND flag MUST be set.

If the Command Code Format (CCF) of the request mandates a Session-Id AVP, the Session-Id AVP MUST identify one of the single sessions which is assigned to at least one of the groups being identified in the appended Session-Group-Id AVPs.

The sender of the request MUST indicate to the receiver how multiple resulting transactions associated with a group command are to be treated by appending a single instance of a Group-Response-Action AVP. For example, when a server sends a Re-Authorization Request (RAR) or a service-specific server-initiated request to the client, it indicates to the client to follow the request according to one of three possible procedures. When the server sets the Group-Response-Action AVP to ALL_GROUPS (1), the client sends a single RAR message for all identified groups. When the server sets the Group-Response-Action AVP to PER_GROUP (2), the client sends a single RAR message for each identified group individually. When the server sets the Group-Response-Action AVP to PER_SESSION (3), the client follows-up with a single RAR message per impacted session. If a session is included in more than one of the identified session groups, the client sends only one RAR message for that session.

If the sender sends a request including the Group-Response-Action AVP set to ALL_GROUPS (1) or PER_GROUP (2), it has to expect some delay before receiving the corresponding answer(s) as the answer(s) will only be sent back when the request is processed for all the sessions or all the session of a session group. If the process of the request is delay-sensitive, the sender SHOULD NOT set the Group-Response-Action AVP to ALL_GROUPS (1) or PER_GROUP (2). If the answer can be sent before the complete process of the request for all the sessions or if the request timeout timer is high enough, the sender MAY set the Group-Response-Action AVP to ALL_GROUPS (1) or PER_GROUP (2).

If the sender wants the receiver of the request to process the associated command solely for a single session, the sender does not append any group identifier, but identifies the relevant session in the Session-Id AVP.

4.4.2. Receiving Group Commands

A Diameter node receiving a request to process a command for a group of sessions, identifies the relevant groups according to the appended Session-Group-Id AVP in the Session-Group-Info AVP and processes the group command according to the appended Group-Response-Action AVP . If the received request identifies multiple groups in multiple appended Session-Group-Id AVPs, the receiver SHOULD process the associated command for each of these groups. If a session has been assigned to more than one of the identified groups, the receiver MUST process the associated command only once per session.

4.4.3. Error Handling for Group Commands

When a Diameter node receives a request to process a command for one or more session groups and the result of processing the command is an error that applies to all sessions in the identified groups, an associated protocol error MUST be returned to the source of the request. In such case, the sender of the request MUST fall back to single-session processing and the session groups, which have been identified in the group command, MUST be deleted according to the procedure described in Section 4.3.

When a Diameter node receives a request to process a command for one or more session groups and the result of processing the command succeeds for some sessions identified in one or multiple session groups, but fails for one or more sessions, the Result-Code AVP in the response message SHOULD indicate DIAMETER_LIMITED_SUCCESS as per Section 7.1.2 of [RFC6733].

In the case of limited success, the sessions, for which the processing of the group command failed, MUST be identified using a

Failed-AVP AVP as per Section 7.5 of [RFC6733]. The sender of the request MUST fall back to single-session operation for each of the identified sessions, for which the group command failed. In addition, each of these sessions MUST be removed from all session groups to which the group command applied. To remove sessions from a session group, the Diameter client performs the procedure described in Section 4.2.2.

4.4.4. Single-Session Fallback

Either Diameter node can fall back to single session operation by ignoring and omitting the optional group session-specific AVPs. Fallback to single-session operation is performed by processing the Diameter command solely for the session identified in the mandatory Session-Id AVP. In such case, the response to the group command MUST NOT identify any group but identify solely the single session for which the command has been processed.

5. Operation with Proxy Agents

In the case of a present stateful Proxy Agent between a Diameter client and a Diameter server, the Proxy Agent MUST perform the same mechanisms per this specification to advertise session grouping and group operations capability towards the client and the server respectively. The Proxy MUST update and maintain consistency of its local session states as per the result of the group commands which are operated between a Diameter client and a server. In such case, the Proxy Agent MUST act as a Diameter server in front of the Diameter client and MUST act as a Diameter client in front of the Diameter server. Therefore, the client and server behavior described in Section 4 applies respectively to the stateful Proxy Agent.

If a stateful Proxy Agent manipulates session groups, it MUST maintain consistency of session groups between a client and a server. This applies to a deployment where the Proxy Agent utilizes session grouping and performs group operations with, for example, a Diameter server, whereas the Diameter client is not aware of session groups. In such case the Proxy Agent must reflect the states associated with the session groups as individual session operations towards the client and ensure the client has a consistent view of each session. The same applies to a deployment where all nodes, the Diameter client and server, as well as the Proxy Agent are group-aware but the Proxy Agent manipulates groups, e.g., to adopt different administrative policies that apply to the client's domain and the server's domain.

Stateless Proxy Agents do not maintain any session state (only transaction state are maintained). Consequently, the notion of session group is transparent for any stateless Proxy Agent present between a Diameter client and a Diameter server handling session groups. Session group related AVPs being defined as optional AVP are ignored by stateless Proxy Agents and should not be removed from the Diameter commands. If they are removed by the Proxy Agent for any reason, the Diameter client and Diameter server will discover the absence the related session group AVPs and will fall back to single-session processing, as described in Section 4.

6. Commands Formatting

This document does not specify new Diameter commands to enable group operations, but relies on command extensibility capability provided by the Diameter Base protocol. This section provides the guidelines to extend the CCF of existing Diameter commands with optional AVPs to enable the recipient of the command applying the command to all sessions associated with the identified group(s).

6.1. Formatting Example: Group Re-Auth-Request

A request for re-authentication of one or more groups of users is issued by appending one or multiple Session-Group-Id AVP(s), as well as a single instance of a Group-Response-Action AVP to the Re-Auth-Request (RAR). The one or multiple Session-Group-Id AVP(s) identify the associated group(s) for which the group re-authentication has been requested. The Group-Response-Action AVP identifies the expected means to perform and respond to the group command. The recipient of the group command initiates re-authentication for all users associated with the identified group(s). Furthermore, the sender of the group re-authentication request appends a Group-Response-Action AVP to provide more information to the receiver of the command about how to accomplish the group operation.

The value of the mandatory Session-Id AVP MUST identify a session associated with a single user, which is assigned to at least one of the groups being identified in the appended Session-Group-Id AVPs.

```
<RAR>  ::= < Diameter Header: 258, REQ, PXY >
           < Session-Id >
           { Origin-Host }
           { Origin-Realm }
           { Destination-Realm }
           { Destination-Host }
           { Auth-Application-Id }
           { Re-Auth-Request-Type }
           [ User-Name ]
           [ Origin-State-Id ]
         * [ Proxy-Info ]
         * [ Route-Record ]
           [ Session-Group-Capability-Vector ]
         * [ Session-Group-Info ]
           [ Group-Response-Action ]
         * [ AVP ]
```

7. Attribute-Value-Pairs (AVP)

Attribute Name	AVP Code	Value Type	AVP Flag rules MUST	MAY	SHOULD NOT	MUST NOT
Session-Group-Info	TBD1	Grouped		P		V
Session-Group-Control-Vector	TBD2	Unsigned32		P		V
Session-Group-Id	TBD3	OctetString		P		V
Group-Response-Action	TBD4	Unsigned32		P		V
Session-Group-Capability-Vector	TBD5	Unsigned32		P		V

AVPs for the Diameter Group Signaling

7.1. Session-Group-Info AVP

The Session-Group-Info AVP (AVP Code TBD1) is of type Grouped. It contains the identifier of the session group as well as an indication of the node responsible for session group identifier assignment.

```
Session-Group-Info ::= < AVP Header: TBD1 >
                       < Session-Group-Control-Vector >
                       [ Session-Group-Id ]
                     * [ AVP ]
```

7.2. Session-Group-Control-Vector AVP

The Session-Group-Control-Vector AVP (AVP Code TBD2) is of type Unsigned32 and contains a 32-bit flags field to control the group assignment at session-group aware nodes.

The following control flags are defined in this document:

SESSION_GROUP_ALLOCATION_ACTION (0x00000001)

This flag indicates the action to be performed for the identified session. When this flag is set, it indicates that the identified Diameter session is to be assigned to the session group as identified by the Session-Group-Id AVP or the session's assignment to the session group identified in the Session-Group-Id AVP is still valid. When the flag is cleared, the identified Diameter session is to be removed from at least one session group. When the flag is cleared and the Session-Group-Info AVP identifies a particular session group in the associated Session-Group-Id AVP, the session is to be removed solely from the identified session group. When the flag is cleared and the Session-Group-Info AVP does not identify a particular session group (Session-Group-Id AVP is absent), the identified Diameter session is to be removed from all session groups, to which it has been previously assigned.

SESSION_GROUP_STATUS_IND (0x00000010)

This flag indicates the status of the session group identified in the associated Session-Group-Id AVP. The flag is set when the identified session group has just been created or is still active. If the flag is cleared, the identified session group is deleted and the associated Session-Group-Id is released. If the Session-Group-Info AVP does not include a Session-Group-Id AVP, this flag is meaningless and MUST be ignored by the receiver.

7.3. Session-Group-Id AVP

The Session-Group-Id AVP (AVP Code TBD3) is of type UTF8String and identifies a group of Diameter sessions.

The Session-Group-Id MUST be globally unique. The default format of the Session-Group-id MUST comply to the format recommended for a Session-Id, as defined in the section 8.8 of the [RFC6733]. The <DiameterIdentity> portion of the Session-Group-Id MUST identify the Diameter node, which owns the session group.

7.4. Group-Response-Action AVP

The Group-Response-Action AVP (AVP Code TBD4) is of type Unsigned32 and contains a 32-bit address space representing values indicating how the node SHOULD issue follow up exchanges in response to a command which impacts multiple sessions. The following values are defined by this document:

ALL_GROUPS (1)
Follow up message exchanges associated with a group command should be performed with a single message exchange for all impacted groups.

PER_GROUP (2)
Follow up message exchanges associated with a group command should be performed with a separate message exchange for each impacted group.

PER_SESSION (3)
Follow up message exchanges associated with a group command should be performed with a separate message exchange for each impacted session.

7.5. Session-Group-Capability-Vector AVP

The Session-Group-Capability-Vector AVP (AVP Code TBD5) is of type Unsigned32 and contains a 32-bit flags field to indicate capabilities in the context of session-group assignment and group operations.

The following capabilities are defined in this document:

BASE_SESSION_GROUP_CAPABILITY (0x00000001)

This flag indicates the capability to support session grouping and session group operations according to this specification.

8. Result-Code AVP Values

This document does not define new Result-Code [RFC6733] values for existing applications, which are extended to support group commands. Specification documents of new applications, which will have intrinsic support for group commands, may specify new Result-Codes.

9. IANA Considerations

This section contains the namespaces that have either been created in this specification or had their values assigned to existing namespaces managed by IANA.

9.1. AVP Codes

This specification requires IANA to register the following new AVPs from the AVP Code namespace defined in [RFC6733].

- Session-Group-Info
- Session-Group-Control-Vector
- Session-Group-Id
- Group-Response-Action
- Session-Group-Capability-Vector

The AVPs are defined in Section 7.

9.2. New Registries

This specification requires IANA to create two registries:

- Session-Group-Control-Vector AVP registry for control bits with two initial assignments, which are described in Section 7.2. The future registration assignment policy is proposed to be Specification Required.
- Session-Group-Capability-Vector AVP with one initial assignment, which is described in Section 7.5. The future registration assignment policy is proposed to be Standards Action.

The AVP names can be used as registry names.

10. Security Considerations

The security considerations of the Diameter protocol itself are discussed in [RFC6733]. Use of the AVPs defined in this document MUST take into consideration the security issues and requirements of the Diameter base protocol. In particular, the Session-Group-Info AVP (including the Session-group-Control-Vector and the Session-Group-Id AVPs) should be considered as a security-sensitive AVPs in the same manner than the Session-Id AVP in the Diameter base protocol [RFC6733].

The management of session groups relies upon the existing trust relationship between the Diameter client and the Diameter server managing the groups of sessions. This document defines a mechanism that allows a client or a server to act on multiple sessions at the same time using only one command. if the Diameter client or server is compromised, an attacker could launch DoS attacks by terminating a large number of sessions with a limited set of commands using the session group management concept.

According to the Diameter base protocol [RFC6733], transport connections between Diameter peers are protected by TLS/TCP, DTLS/SCTP or alternative security mechanisms that are independent of Diameter, such as IPsec. However, the lack of end-to-end security features makes it difficult to establish trust in the session group related information received from non-adjacent nodes. Any Diameter agent in the message path can potentially modify the content of the message and therefore the information sent by the Diameter client or the server. There is ongoing work on the specification of end-to-end security features for Diameter. Such features would enable the establishment of trust relationship between non-adjacent nodes and the security required for session group management would normally rely on this end-to-end security. However, there is no assumption in this document that such end-to-end security mechanism will be available. It is only assumed that the solution defined on this document relies on the security framework provided by the Diameter based protocol.

In some cases, a Diameter Proxy agent can act on behalf of a client or server. In such a case, the security requirements that normally apply to a client (or a server) apply equally to the Proxy agent.

11. Acknowledgments

The authors of this document want to thank Ben Campbell and Eric McMurry for their valuable comments to early versions of this draft. Furthermore, authors thank Steve Donovan and Mark Bales for the thorough review and comments on advanced versions of the WG document, which helped a lot to improve this specification.

12. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

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

[RFC7155] Zorn, G., Ed., "Diameter Network Access Server Application", RFC 7155, DOI 10.17487/RFC7155, April 2014, <https://www.rfc-editor.org/info/rfc7155>.

Appendix A. Session Management -- Exemplary Session State Machine

A.1. Use of groups for the Authorization Session State Machine

Section 8.1 in [RFC6733] defines a set of finite state machines, representing the life cycle of Diameter sessions, and which must be observed by all Diameter implementations that make use of the authentication and/or authorization portion of a Diameter application. This section defines, as example, additional state transitions related to the processing of the group commands which may impact multiple sessions.

The group membership is session state and therefore only those state machines from [RFC6733] in which the server is maintaining session state are relevant in this document. As in [RFC6733], the term Service-Specific below refers to a message defined in a Diameter application (e.g., Mobile IPv4, NASREQ).

The following state machine is observed by a client when state is maintained on the server. State transitions which are unmodified from [RFC6733] are not repeated here.

The Diameter group command in the following tables is differentiated from a single-session related command by a preceding 'G' (Group). A Group Re-Auth Request, which applies to one or multiple session groups, has been exemplarily described in Section 6.1. Such Group RAR command is denoted as 'GRAR' in the following table. The same notation applies to other commands as per [RFC6733].

CLIENT, STATEFUL

State	Event	Action	New State
Idle	Client or Device Requests access	Send service specific auth req optionally including groups	Pending

Open	GASR received with Group-Response-Action = ALL_GROUPS, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send GSTR.	Discon
Open	GASR received with Group-Response-Action = PER_GROUPS, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send GSTR per group	Discon
Open	GASR received with Group-Response-Action = PER_SESSION, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send STR per session	Discon
Open	GASR received, client will not comply with request to end all session in received group(s)	Send GASA with Result-Code != SUCCESS	Open
Discon	GSTA Received	Discon. user/device	Idle
Open	GRAR received with Group-Response-Action = ALL_GROUPS, session is assigned to received group(s) and client will perform subsequent re-auth	Send GRAA, Send service specific group re-auth req	Pending
Open	GRAR received with Group-Response-Action = PER_GROUP, session is assigned to received group(s) and client will perform subsequent re-auth	Send GRAA, Send service specific group re-auth req per group	Pending
Open	GRAR received with	Send GRAA,	Pending

	Group-Response-Action = PER_SESSION, session is assigned to received group(s) and client will perform subsequent re-auth	Send service specific re-auth req per session	
Open	GRAR received and client will not perform subsequent re-auth	Send GRAA with Result-Code != SUCCESS, Discon. user/device	Idle
Pending	Successful service-specific group re-authorization answer received.	Provide service	Open
Pending	Failed service-specific group re-authorization answer received.	Discon. user/device	Idle

The following state machine is observed by a server when it is maintaining state for the session. State transitions which are unmodified from [RFC6733] are not repeated here.

SERVER, STATEFUL

State	Event	Action	New State
Idle	Service-specific authorization request received, and user is authorized	Send successful service specific answer optionally including groups	Open
Open	Server wants to terminate group(s)	Send GASR	Discon
Discon	GASA received	Cleanup	Idle
Any	GSTR received	Send GSTA, Cleanup	Idle
Open	Server wants to reauth group(s)	Send GRAR	Pending
Pending	GRAA received with Result-Code = SUCCESS	Update session(s)	Open
Pending	GRAA received with Result-Code != SUCCESS	Cleanup session(s)	Idle
Open	Service-specific group re-authoization request received and user is authorized	Send successful service specific group re-auth answer	Open
Open	Service-specific group re-authorization request received and user is not authorized	Send failed service specific group re-auth answer, cleanup	Idle

Authors' Addresses

Mark Jones

Email: mark@azu.ca

Marco Liebsch

Email: marco.liebsch@neclab.eu

Lionel Morand

Email: lionel.morand@orange.com

Diameter Maintenance and Extensions (DIME) J. Korhonen, Ed.
Internet-Draft Broadcom
Intended status: Standards Track S. Donovan, Ed.
Expires: April 30, 2015 B. Campbell
Oracle
L. Morand
Orange Labs
October 27, 2014

Diameter Overload Indication Conveyance
draft-ietf-dime-ovli-04.txt

Abstract

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

Requirements

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This specification defines a base solution for Diameter Overload Control (DOC), referred to as Diameter Overload Indication Conveyance (DOIC). The requirements for the solution are described and discussed in the corresponding design requirements document [RFC7068]. Note that the overload control solution defined in this specification does not address all the requirements listed in [RFC7068]. A number of overload control related features are left for the future specifications. See Appendix A for a list of extensions that are currently being considered. See Appendix C for an analysis of the conformance to the requirements specified in [RFC7068].

The solution defined in this specification addresses Diameter overload control between Diameter nodes that support the DOIC solution. Furthermore, 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.

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

Diversion

Abatement of traffic sent to a reporting node by a reacting node in response to receipt of an overload report. The abatement is achieved by diverting traffic from the reporting node to another Diameter node that is able to process the request.

Host-Routed Request

The set of requests that a reacting node knows will be served by a particular host, either due to the presence of a Destination-Host AVP, or by some other local knowledge on the part of the reacting node.

Overload Control State (OCS)

Reporting and reacting node internally maintained state describing occurrences of overload control.

Overload Report (OLR)

Information sent by a reporting node indicating the start, continuation or end of an occurrence of overload control.

Reacting Node

A Diameter node that acts upon an overload report.

Realm-Routed Request

The set of requests that a reacting node does not know the host that will service the request.

Reporting Node

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

Throttling

Throttling is the reduction of the number of requests sent to an entity. Throttling can include a Diameter Client or Diameter Server dropping requests, or a Diameter Agent rejecting requests with appropriate error responses. In extreme cases reporting nodes can also throttle requests when the requested reductions in traffic does not sufficiently address the overload scenario.

3. Solution Overview

The Diameter Overload Information Conveyance (DOIC) solution allows Diameter nodes to request other nodes to 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 clients, servers, and agents. DOIC nodes are further divided into "Reporting Nodes" and "Reacting Nodes." A reporting node requests overload abatement by sending an Overload Report (OLR) to one or more reacting nodes.

A reacting node acts upon OLRs, and performs whatever actions are needed to fulfil the abatement requests included in the OLRs. A Reporting node may report overload on its own behalf, or on behalf of other (typically upstream) nodes. Likewise, a reacting node may perform overload abatement on its own behalf, or on behalf of other (typically downstream) nodes.

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

Likewise, a relay or proxy 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 6.2) into existing requests and responses. Reporting nodes send OLRs by inserting OC-OLR AVPs (Section 6.3).

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 the parameters of an OLR and the procedures required for overload abatement. This document specifies a single must-support algorithm, namely the "loss" algorithm (Section 5). 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 reasonably attempt to send requests to other destinations or via other agents. 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 6.6), where the type defines such behaviors. Report types are extensible. This document defines report types for overload of a specific server, and for overload of an entire realm.

A report of type host is sent to indicate the overload of a specific server for the application-id indicated in the transaction. When receiving an OLR of type host, a reacting node applies overload abatement to what is referred to in this document as host-routed requests. This is the set of requests that the reacting node knows will be served by a particular host, either due to the presence of a Destination-Host AVP, or by some other local knowledge on the part of the reacting node. The reacting node applies overload abatement on those host-routed requests which the reacting node knows will be served by the server that matches the Origin-Host AVP of the received message that contained the received OLR of type host.

A report type of realm is sent to indicate the overload of all servers in a realm for the application-id. When receiving an OLR of type realm, a reacting node applies overload abatement to what is referred to in this document as realm-routed requests. This is the set of requests that are not host-routed as defined in the previous paragraph.

While a reporting node sends OLRs to "adjacent" reacting nodes, nodes that are "adjacent" for DOIC purposes may not be adjacent from a Diameter, or transport, perspective. For example, one or more Diameter agents that do not support DOIC may exist between a given pair of reporting and reacting nodes, as long as those agents pass unknown AVPs through unchanged. The report types described in this document can safely pass through non-supporting agents. This may not be true for report types defined in future specifications. Documents that introduce new report types MUST describe any limitations on their use across non-supporting agents.

3.1. Piggybacking Principle

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 overload control top-level AVPs, the OC-OLR AVP and the OC-Supported-Features AVP, as optional AVPs into existing commands when the corresponding Command Code Format (CCF) specification allows adding new optional AVPs (see Section 1.3.4 of [RFC6733]).

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 originated or relayed by the reporting node. Reporting nodes also include overload reports using the OC-OLR AVP in answer messages.

> Note: There is no new Diameter application defined to carry overload related AVPs. The DOIC AVPs are carried in existing Diameter application 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. send 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.

3.2. 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 Diameter Capability Exchange.

The DCA solution 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-Feature AVP in the request message. This includes an indication that it supports the loss overload abatement algorithm defined in this specification (see Section 5). This ensures that there is at least one commonly supported overload abatement algorithm between the reporting node and the reacting nodes in the path of the request.

> DOIC must support deployments where Diameter Clients and/or Diameter Servers do not support the DOIC solution. In this scenario, it is assumed that Diameter Agents that support the DOIC solution will handle overload abatement for the non supporting Diameter nodes. In this case the DOIC agent will insert the OC-Supporting-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.

The reporting node inserts the OC-Supported-Feature AVP in all answer messages to requests that contained the OC-Supported-Feature AVP. The contents of the reporting node's OC-Supported-Feature AVP indicate the set of Diameter overload features supported by the reporting node with one exception.

The reporting node only includes an indication of support for one overload abatement algorithm. This is the algorithm that the reporting node intends to use should it enter an overload condition or requests to use while it actually is in 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 to ensure that overload reports can be properly handled.

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.

The DCA mechanism must also support 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 updates the OC-Supported-Feature AVP to reflect the mixture of the two sets of supported features.

The logic to determine the content of the modified OC-Supported-Feature AVP is out-of-scope for this specification and is left to implementation decisions. Care must be taken not to introduce interoperability issues for downstream or upstream DOIC nodes.

3.3. DOIC Overload Condition Reporting

As with DOIC Capability Announcement, Overload Condition Reporting uses new AVPs (Section 6.3) 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 abatement algorithm specific AVPs.

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

A report of type host is sent to indicate the overload of a specific Diameter node for the application-id indicated in the transaction. When receiving an OLR of type host, a reacting node applies overload abatement to what is referred to in this document as host-routed requests. This is the set of requests that the reacting node knows will be served by a particular host, either due to the presence of a Destination-Host AVP, or by some other local knowledge on the part of the reacting node. The reacting node applies overload abatement on those host-routed requests which the reacting node knows will be served by the server that matches the Origin-Host AVP of the received message that contained the received OLR of type host.

Realm reports apply to realm-routed requests for a specific realm as indicated in the Destination-Realm AVP.

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

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, are responsible for applying the abatement algorithm to traffic impacted by the overload report. The method used for that abatement is dependent on the abatement algorithm. The loss abatement algorithm is defined in this document (Section 5). Other abatement algorithms can be defined in extensions to the DOIC solutions.

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 use of the abatement algorithm 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.

3.4. DOIC Extensibility

The DOIC solution is designed to be extensible. This extensibility is based on existing Diameter based extensibility mechanisms.

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 new definitions of the scope of messages impacted by an overload report.

The DOIC solution uses the OC-Supported-Features AVP for DOIC nodes to communicate supported features. The specific features supported by the DOIC node are indicated in the OC-Feature-Vector AVP. DOIC extensions must 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.

Reporting nodes use the OC-OLR AVP to communicate overload occurrences. This AVP can also be extended to add new AVPs allowing a reporting nodes to communicate additional information about handling an overload condition.

If necessary, new extensions can also define new top-level AVPs. It is, however, recommended that DOIC extensions use the OC-Supported-Features and OC-OLR to carry all DOIC related AVPs.

3.5. 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)  <-----------------------> <----------------------->
                 standard base protocol    standard base protocol
```

Figure 1: Simplified architecture choices for overload indication delivery

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

Diameter agent inside the realm and then between the Diameter agent and the clients.

4. Solution Procedures

This section outlines the normative behavior associated with the DOIC solution.

4.1. Capability Announcement

This section defines DOIC Capability Announcement (DCA) behavior.

4.1.1. Reacting Node Behavior

A reacting node MUST include the OC-Supported-Features AVP in all request messages.

A reacting node MAY include the OC-Feature-Vector AVP with an indication of the loss algorithm. A reacting node MUST include the OC-Feature-Vector AVP to indicate support for abatement algorithms in addition to the loss algorithm.

A reacting node SHOULD indicate support for all other DOIC features it supports.

> Not all DOIC features will necessarily apply to all transactions. For instance, there may be a future extension that only applies to session based applications. A reacting node that supports this extension can choose to not include it for non session based applications.

An OC-Supported-Features AVP in answer messages indicates there is a reporting node for the transaction. The reacting node MAY take action based on the features indicated in the OC-Feature-Vector AVP.

> Note that the loss abatement algorithm is the only feature described in this document and it does not require action to be taken when there is an active overload report. This behavior is described in Section 4.2 and Section 5.

4.1.2. 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.

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

The reporting node MUST NOT include the OC-Supported-Features AVP, OC-OLR AVP or any other overload control AVPs defined in extension drafts 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.

Based on the content of the OC-Supported-Features AVP in the request message, the reporting node knows what overload control functionality is supported by the reacting node. The reporting node then acts accordingly for the subsequent answer messages it initiates.

The reporting node MUST indicate support for one and only one abatement algorithm in the OC-Feature-Vector AVP. The abatement algorithm included MUST be from the set of abatement algorithms contained in the request message's OC-Supported-Features AVP. The abatement algorithm included MUST indicate the abatement algorithm the reporting node wants the reacting node to use when the reporting node enters an overload condition.

For an ongoing overload state, a reacting node MUST keep the algorithm that was selected by the reporting node in further requests towards the reporting node. The reporting node SHOULD NOT change the selected algorithm during a period of time that it is in an overload condition and, as a result, is sending OC-OLR AVPs in answer messages.

The reporting node SHOULD indicate support for other DOIC features defined in extension drafts that it supports and that apply to the transaction.

> Note that 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 reporting node policy.

4.1.3. Agent Behavior

Diameter agents that support DOIC MUST ensure that all messages have the OC-Supporting-Features AVP. If a message handled by the DOIC agent does not include the OC-Supported-Features AVP then the DOIC agent inserts the AVP. If the message already has the AVP then the agent either leaves it unchanged in the relayed message or modifies it to reflect a mixed set of DOIC features.

An agent MAY modify the OC-Supported-Features AVP carried in answer messages.

For instance, 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 agent modifies the OC-Supported-Features AVP sent to the reporting node then it might also need to modify the OC-Supported-Features AVP sent to a reacting node in the subsequent answer message, as it cannot send an indication of support for features that are not supported by the reacting node.

Editor’s note: There is an open issue on the wording around agent behavior in this case that needs to be resolved prior to finishing this document.

4.2. Overload Report Processing

4.2.1. Overload Control State

Both reacting and reporting nodes maintain Overload Control State (OCS) for active overload conditions.

4.2.1.1. Overload Control State for Reacting Nodes

A reacting node SHOULD maintain 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 Host-Id.

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

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

- o Sequence number (as received in OC-OLR)

- 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 OC-Supported-Features AVP)

- o Abatement Algorithm specific input data (as received within the OC-OLR AVP, for example, OC-Reduction-Percentage for the Loss abatement algorithm)

4.2.1.2. Overload Control State for Reporting Nodes

A reporting node SHOULD maintain OCS entries per supported Diameter application, per supported (and eventually selected) Abatement Algorithm and per report-type.

An OCS entry is identified by the pair of Application-Id and Abatement Algorithm.

The OCS entry for a given pair of Application and Abatement Algorithm MAY include the information (the actual information stored is an implementation decision):

- o Report type

- o Sequence number

- o Validity Duration

- o Expiration Time

- o Algorithm specific input data (for example, the Reduction Percentage for the Loss Abatement Algorithm)

4.2.1.3. Reacting Node 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.

> For the remainder of this section the term OLR referres to the combination of the contents of the received OC-OLR AVP and the abatement algorithm indicated in the received OC-Supported-Features AVP.

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

For a host report-type this means it matches the app-id and host-id in an existing host OCS entry.

For a realm report-type this means it matches the app-id and realm-id in an existing realm OCS entry.

If the OLR is for an existing overload condition then it 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 the 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 the reacting node MUST silently ignore the received OLR. The matching OCS MUST NOT be updated in this case.

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

For a host report-type this means it creates on OCS entry with the app-id of the application-id in the received message and host-id 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-type this means it creates on OCS entry with the app-id of the application-id in the received message and realm-id of the Origin-Realm in the received message.

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

> Note that it is not necessarily appropriate to delete the OCS entry, as there is recommended behavior 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").

4.2.1.4. Reporting Node Maintenance of Overload Control State

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

> If the 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 MAY be set to any value if there is no unexpired overload report for previous overload conditions sent to any reacting node for the same application and report-type.

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 previously sent by the reporting node. This property MUST hold over a reboot of the reporting node.

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

> For instance, if the reporting node wishes to instruct reacting nodes to continue overload abatement for a longer period of time that 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 NOT update the abatement algorithm in an active OCS entry.

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

> For instance, if the 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.

The reporting node MUST update 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 the reacting nodes 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.

If the reporting node knows that the OCS entries in the reacting nodes are near expiration then the reporting node can decide to delete the OCS entry.

The 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 non-expired reacting node’s OCS entry created as a result of the overload condition in the reporting node is deleted.

4.2.2. Reacting Node Behavior

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

If the request matches and active OCS then the reacting node MUST apply abatement treatment on the request. The abatement treatment applied depends on the abatement algorithm stored in the OCS.

For the Loss abatement algorithm defined in this specification, see Section 5 for the abatement logic applied.

If the 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 Section 7.

In the case that the OCS entry validity duration expires or has a validity duration of zero ("0"), meaning that it the reporting node has explicitly signaled the end of the overload condition then abatement associated with the overload abatement MUST be ended in a controlled fashion.

4.2.3. Reporting Node Behavior

The operation on the reporting node is straight forward.

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

A request matches if the application-id in the request matches the application-id in any active OCS entry and 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 MUST contain all information necessary for the abatement algorithm indicated in the OC-Supported-Features AVP that is also included in the answer message.

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) the 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 that a reacting node advertises support for the host and realm report types by including the OC-Supported-Features AVP in the request. Support for other report types must be explicitly indicated by new feature bits in the OC-Feature-Vector AVP.

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

The reporting node SHOULD 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.

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. Therefore, the reporting node SHOULD NOT apply throttling to the set of messages to which the OLR applies. That is, the same candidate set of messages SHOULD NOT be throttled multiple times.

However, when the reporting node sends and OLR downstream, it MAY still be responsible to apply other abatement methods such as diversion. The reporting node might also need to throttle requests for reasons other then overload. For example, an agent or server might have a configured rate limit for each client, and throttle requests that exceed that limit, even if such requests had already been candidates for throttling by downstream nodes.

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.

> Editor’s Note: There is not yet consensus on the above two paragraphs. Two alternatives are under consideration -- synchronization of sequence numbers and attribution of reports. If no consensus is reached then it will be left to be addressed as an extension.

4.3. Protocol Extensibility

The overload control solution can be extended, e.g. with new traffic abatement algorithms, new report types or other new functionality.

When defining a new extension a new feature bit MUST be defined for the OC-Feature-Vector. This feature bit is used to communicate support for the new feature.

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

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

The handling of feature bits in the OC-Feature-Vector AVP that are not associated with overload abatement algorithms MUST be specified by the extensions that define the features.

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

The OC-OLR AVP 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. If the new sub-AVPs imply new semantics for handling the indicated report type, then a new OC-Report-Type AVP value MUST be defined.

New features (feature bits in the OC-Feature-Vector AVP) and report types (in the OC-Report-Type AVP) MUST be registered with IANA. As with any Diameter specification, new AVPs MUST also be registered with IANA. See Section 8 for the required procedures.

5. Loss Algorithm

This section documents the Diameter overload loss abatement algorithm.

5.1. 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 Diameter Overload Capability Announcement process documented in Section 4.1.

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 use a strategy of applying abatement logic to the requested percentage of request messages sent (or handled in the case of agents) by the reacting node that are impacted by the overload report.

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

1. An overload report is received and the associated overload state 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 abatement should be applied to the request. One approach that could be taken for each request is to select a random number between 1 and 100. If the random number is less than the indicated reduction percentage then the

request is given abatement treatment, otherwise the request is given normal routing treatment.

5.2. Reporting Node Behavior

The method a reporting nodes 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 traffic reduction it includes an OC-OLR AVP in response messages as described in Section 4.2.3.

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 handing specified in Section 4.2.3.

When the reporting node determines it no longer needs a reduction in traffic the reporting node SHOULD send an overload report indicating the overload report is no longer valid, as specified in Section 4.2.3.

5.3. 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.

When applying overload abatement treatment for the load abatement algorithm, the reacting node MUST abate, either by throttling or diversion, the requested percentage of requests that would have otherwise been sent to the reporting host or realm.

If reacting node comes out of the 100 percent traffic reduction as a result of the overload report timing out, the following concerns are

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

If the reacting node does not receive an OLR in messages sent to the formerly overloaded node then the reacting node SHOULD slowly increase the rate of traffic sent to the overloaded node.

It is suggested that the reacting node decrease the amount of traffic given abatement treatment by 20% each second until the reduction is completely removed and no traffic is given abatement treatment.

> 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.

6. Attribute Value Pairs

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

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 works on that application.

6.1. OC-Supported-Features AVP

The OC-Supported-Features AVP (AVP code TBD1) is type of 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: TBD1 >
                          [ OC-Feature-Vector ]
                        * [ AVP ]
```

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 (see Section 6.2). The absence of the OC-Feature-Vector AVP indicates that only the default traffic abatement algorithm described in this specification is supported.

6.2. OC-Feature-Vector AVP

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

The following capabilities are defined in this document:

OLR_DEFAULT_ALGO (0x0000000000000001)

When this flag is set by the DOIC node it means that the default traffic abatement (loss) algorithm is supported.

6.3. OC-OLR AVP

The OC-OLR AVP (AVP code TBD2) is type of Grouped and contains the information necessary to convey an overload report on an overload condition at the reporting node. The OC-OLR AVP does not explicitly contain all information needed by the reacting node to decide whether a subsequent request must undergo a throttling process with the received reduction percentage. The value of the OC-Report-Type AVP within the OC-OLR AVP indicates which implicit information is relevant for this decision (see Section 6.6). The application the OC-OLR AVP applies to is the same as 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: TBD2 >
           < OC-Sequence-Number >
           < OC-Report-Type >
           [ OC-Reduction-Percentage ]
           [ OC-Validity-Duration ]
         * [ AVP ]
```

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

6.4. OC-Sequence-Number AVP

The OC-Sequence-Number AVP (AVP code TBD3) is type of Unsigned64. Its usage in the context of overload control is described in Section 4.2.

From the functionality point of view, the OC-Sequence-Number AVP MUST be used as a non-volatile increasing counter for a sequence of overload reports between two DOIC nodes for the same overload occurrence. The sequence number is only required to be unique between two DOIC nodes. Sequence numbers are treated in a uni-directional manner, i.e. two sequence numbers on each direction between two DOIC nodes are not related or correlated.

6.5. OC-Validity-Duration AVP

The OC-Validity-Duration AVP (AVP code TBD4) is type of Unsigned32 and indicates in milliseconds the validity time of the overload report. The number of milliseconds 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 5000 (i.e., 5 seconds). When the OC-Validity-Duration AVP is not present in the OC-OLR AVP, the default value applies. Validity duration with values above 86400 (i.e.; 24 hours) MUST NOT be used. Invalid duration values are treated as if the OC-Validity-Duration AVP were not present and result in the default value being used.

Editor’s note: There is an open discussion on whether to have an upper limit on the OC-Validity-Duration value, beyond that which can be indicated by an Unsigned32.

A timeout of the overload report has specific concerns that need to be taken into account by the DOIC node acting on the earlier received overload report(s). Section 6.7 discusses the impacts of timeout in the scope of the traffic abatement algorithms.

6.6. OC-Report-Type AVP

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

0 A host report. The overload treatment should apply to requests for which all of the following conditions are true:

Either the Destination-Host AVP is present in the request and its value matches the value of the Origin-Host AVP of the received message that contained the OC-OLR AVP; or the Destination-Host is not present in the request but the value of the peer identity associated with the connection used to send the request matches the value of the Origin-Host AVP of the received message that contained the OC-OLR AVP.

The value of the Destination-Realm AVP in the request matches the value of the Origin-Realm AVP of the received message that contained the OC-OLR AVP.

The value of the Application-ID in the Diameter Header of the request matches the value of the Application-ID of the Diameter Header of the received message that contained the OC-OLR AVP.

1 A realm report. The overload treatment should apply to requests for which all of the following conditions are true:

The Destination-Host AVP is absent in the requestand the value of the peer identity associated with the connection used to send the request does not match a server that could serve the request.

The value of the Destination-Realm AVP in the request matches the value of the Origin-Realm AVP of the received message that contained the OC-OLR AVP.

The value of the Application-ID in the Diameter Header of the request matches the value of the Application-ID of the Diameter Header of the received message that contained the OC-OLR AVP.

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

6.7. OC-Reduction-Percentage AVP

The OC-Reduction-Percentage AVP (AVP code TBD8) is type of Unsigned32 and describes the percentage of the traffic that the sender is requested to reduce, compared to what it otherwise would 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. The default value of the OC-Reduction-Percentage AVP is 0. When the OC-Reduction-Percentage AVP is not present in the overload report, the default value applies.

6.8. Attribute Value Pair flag rules

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

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

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

7. 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.

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, 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.

> For instance, 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.

8. IANA Considerations

8.1. AVP codes

New AVPs defined by this specification are listed in Section 6. All AVP codes allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

8.2. New registries

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

Section 6.2 defines a new "Overload Control Feature Vector" registry including the initial assignments. New values can be added into the registry using the Specification Required policy [RFC5226]. See Section 6.2 for the initial assignment in the registry.

Section 6.6 defines a new "Overload Report Type" registry with its initial assignments. New types can be added using the Specification Required policy [RFC5226].

9. Security Considerations

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

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

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

9.1. 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 a 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.

9.2. 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.

9.3. Non-Compliant Nodes

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

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

9.4. End-to End-Security Issues

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

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

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

10. Contributors

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

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

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.

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

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

11.2. Informative References

[Cx] 3GPP, , "ETSI TS 129 229 V11.4.0", August 2013.

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

[PCC] 3GPP, , "ETSI TS 123 203 V11.12.0", December 2013.

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

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

[RFC7068] McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", RFC 7068, November 2013.

[S13] 3GPP, , "ETSI TS 129 272 V11.9.0", December 2012.

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 6.1 and 8 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

The proposal was made to add a new Error Diagnostic AVP to supplement the error responces to be able 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, it is recommended that deployments enable all agents that do server selection to support the DOIC solution prior to enabling the DOIC solution in the Diameter network.

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.

Appendix C. Requirements Conformance Analysis

This section contains the result of an analysis of the DOIC solutions conformance to the requirements defined in [RFC7068].

To be completed.

Appendix D. Considerations for Applications Integrating the DOIC Solution

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

D.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 identity responsible for handling overload reports.

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

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

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 D.2.

D.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. Appendix D.3 discusses considerations for handling various request types when the target server is known to be in an overloaded state.

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

Stateless applications:

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

Pseudo-session applications:

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

Session-based applications:

Overload handling for session-based applications must take into consideration the work load associated with setting up and maintaining a session. As such, the entity sending requests towards an overloaded Diameter entity for a session-based application might tend to reject new session requests prior to rejecting intra-session requests. In addition, session ending requests might be given a lower probability of being rejected as rejecting session ending requests could result in session status being out of sync between the Diameter clients and servers. Application designers that would decide to reject mid-session requests will need to consider whether the rejection invalidates the session and any resulting session clean-up procedures.

D.3. Request Transaction Classification

Independent Request:

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

Session-Initiating Request:

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

Correlated Session-Initiating Request:

There are cases when multiple session-initiated requests must be correlated and managed by the same Diameter server. It is notably the case in the 3GPP PCC architecture [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 than the one used in a previous request. An intra session request generally needs to be delivered to the server that handled the session creating request for the session. The STR message defined in [RFC6733] is an example of an intra-session requests.

Pseudo-Session Requests:

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

D.4. Request Type Overload Implications

The request classes identified in Appendix D.3 have implications on decisions about which requests should be throttled first. The following list of request treatment regarding throttling is provided as guidelines for application designers when implementing the Diameter overload control mechanism described in this document. 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 the overload control is to reduce the total number of requests sent to the overloaded entity, throttling decisions might favor allowing intra-session requests over session-initiating requests. 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, 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. Implementors and operators may choose to throttle session-terminating requests less aggressively in order to gracefully terminate sessions, allow clean-up 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.

Authors' Addresses

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

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

Diameter Maintenance and Extensions (DIME) J. Korhonen, Ed.
Internet-Draft Broadcom
Intended status: Standards Track S. Donovan, Ed.
Expires: February 20, 2016 B. Campbell
Oracle
L. Morand
Orange Labs
August 19, 2015

Diameter Overload Indication Conveyance
draft-ietf-dime-ovli-10.txt

Abstract

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

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 works 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 overload control related features 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 that a reacting node does not know which host will service the request.

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].

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

4. Solution Overview

The Diameter Overload Information Conveyance (DOIC) solution allows Diameter nodes to request other Diameter nodes to 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 (OLR).

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 bi-directional 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 7.2) 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 originated or relayed by the reporting node 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 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 advertized 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 abatement algorithm specific AVPs.

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 (see definition 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 (see definition 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 multiple sources for overload reports which apply to the same Diameter entity exist. 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 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 drafts 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 OC-Feature-Vector. 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 drafts 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 reporting node policy.

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.

> For instance, 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 Abatement Algorithm specific input data (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 includes the following information (the actual information stored is an implementation decision):

- o Sequence number

- o Validity Duration

- o Expiration Time

- o Algorithm specific input data (for example, the Reduction Percentage for the Loss Abatement Algorithm)

5.2.1.3. Reacting Node 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 something close to the maximum value in the OC-Sequence-Number AVP to something close to 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 on 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 on 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 there is recommended behavior 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 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 time stamp 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.

For instance, 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 abatement algorithm specific parameters, including, for example, the reduction percentage used for the Loss abatement algorithm.

For instance, 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 reacting nodes 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 non-expired 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 the interaction of the new abatement treatment 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 if the application-id in the request matches the application-id in any active OCS entry and 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 overload control state cleanup 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 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.

For example, it 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. This feature bit is used to communicate support for the new feature.

The extension may define new AVPs for use in 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.

[RFC6733] defined Grouped AVP extension mechanisms 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, RFC6733 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 Diameter Overload 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 handing 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 reacting node comes out of the 100 percent 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 percent reduction.

> 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 TBD1) 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: TBD1 >
                          [ OC-Feature-Vector ]
                        * [ AVP ]
```

7.2. OC-Feature-Vector AVP

The OC-Feature-Vector AVP (AVP code TBD2) 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 capabilities are 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 TBD3) 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 the same as 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: TBD2 >
           < 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 TBD4) 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 non-volatile increasing counter for a sequence of overload reports between two DOIC nodes for the same overload occurrence. Sequence numbers are treated in a uni-directional manner, i.e., two sequence numbers on each direction between two DOIC nodes are not related or correlated.

7.5. OC-Validity-Duration AVP

The OC-Validity-Duration AVP (AVP code TBD5) 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 TBD6) 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.

7.7. OC-Reduction-Percentage AVP

The OC-Reduction-Percentage AVP (AVP code TBD7) 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. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules MUST	AVP flag rules MUST NOT
OC-Supported-Features	TBD1	7.1	Grouped		V
OC-Feature-Vector	TBD2	7.2	Unsigned64		V
OC-OLR	TBD3	7.3	Grouped		V
OC-Sequence-Number	TBD4	7.4	Unsigned64		V
OC-Validity-Duration	TBD5	7.5	Unsigned32		V
OC-Report-Type	TBD6	7.6	Enumerated		V
OC-Reduction -Percentage	TBD7	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, 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 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

9.2. New registries

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

A new "Overload Control Feature Vector" registry is required. The registry must contain the following:

Feature Vector Value Name

Feature Vector Value

Specification - the specification that defines the new value.

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

A new "Overload Report Type" registry is required. The registry must contain the following:

Report Type Value Name

Report Type Value

Specification - the specification that defines 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 its use for competitive intelligence or to target attacks.

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

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 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 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. Non-Compliant 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 non-compliant node might continue to send requests with no reduction in load. Such non-compliance could be done accidentally, or maliciously to gain an unfair advantage over compliant nodes. Requirement 28 [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. 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

12. References

12.1. 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>.

12.2. Informative References

[Cx] 3GPP, , "ETSI TS 129 229 V11.4.0", August 2013.

[I-D.ietf-dime-e2e-sec-req]
Tschofenig, H., Korhonen, J., Zorn, G., and K. Pillay, "Diameter AVP Level Security: Scenarios and Requirements", draft-ietf-dime-e2e-sec-req-01 (work in progress), October 2013.

[PCC] 3GPP, , "ETSI TS 123 203 V11.12.0", December 2013.

[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, , "ETSI TS 129 272 V11.9.0", December 2012.

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. The following sub-sections 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 7.1 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. Application Type Overload Implications

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 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 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 than the one used in a previous request. An intra-session request generally needs to be delivered to the server that handled the session creating request for the session. The STR message defined in [RFC6733] is an example of an intra-session 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 exists Diameter applications that define an expected ordering of transactions. This sequencing of independent transactions results in a pseudo session. The AIR, MAR and SAR requests in the 3GPP defined Cx [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 treatment 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 the overload control is to reduce the total number of requests sent to the overloaded entity, throttling decisions might favor allowing intra-session requests over session-initiating requests. 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, 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.

Authors' Addresses

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

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