Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: April 16, 2016

S. Donovan
Oracle

October 14, 2015

Diameter Agent Overload and the Peer Overload Report
draft-ietf-dime-agent-overload-03.txt

Abstract

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) base solution. The extension defines the Peer overload report type. The initial use case for the Peer report is the handling of occurrences 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

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

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

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

This Internet-Draft will expire on April 16, 2016.

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. It also defines new overload

report type, the Peer overload report type, that is used for handling of agent overload conditions. The Peer overload report type is defined in a generic fashion so that it can also be used for other Diameter overload scenaios.

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 occurrence 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 occurrences of agent overload. 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.

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 Diameter Agent.

Diameter Endpoint

An RFC6733 Diameter Client and RFC6733 Diameter Server.

Reporting Node

A DOIC Node that sends and overload report in a 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 used to communicate agent overload.

There are two primary classes of use cases currently identified, those involving the overload of agents and those involving overload of Diameter endpoints (Diameter Clients and Diameter Servers) that wish to use an overload algorithm suited controlling traffic sent from a peer.

3.1. Diameter Agent Overload Use Cases

The peer report needs to support the 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.

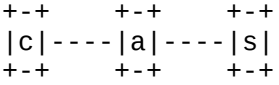

Figure 1

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.

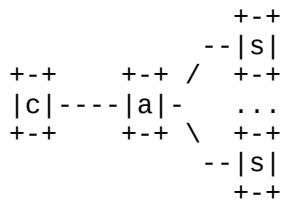

Figure 2

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 overload abatement algorithm. This will result in the throtting of the abated traffic that would have been sent to the agent, as there is no alternative route, with the appropriate indication given to the service request that resulted in the need for the Diameter transaction.

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.

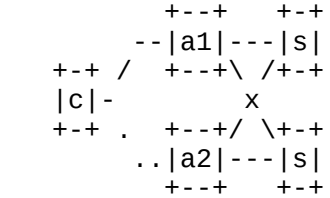

Figure 3

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.

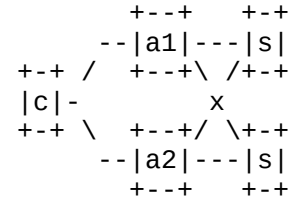

Figure 4

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.

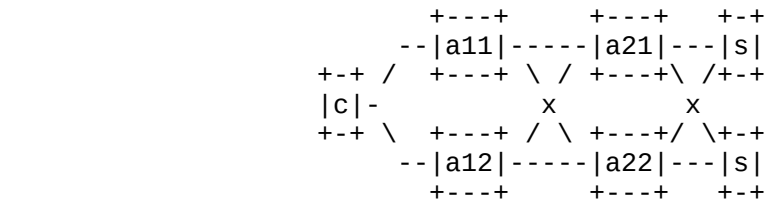

Figure 5

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.

The handling of peer 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 number of transactions. When throttling requests, an agent uses 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 overload report 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, [I-D.ietf-dime-doic-rate-control], 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 an end-point in the path of a transaction are overloaded at the same time. When this occurs, Diameter entities need to handle both overload reports. In this scenario 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.

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. A new AVP in the OC-Supported-Features AVP has been added.

5.1.1. Reacting Node Behavior

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.

> Note: 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.

> Note: This allows the DOIC nodes 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 relaying a request that includes an OC-SourceID AVP in the OC-Supported-Features AVP, a DOIC node that supuports the Peer Report feature must remove the received OC-SourceID AVP and replace it with an OC-SourceID AVP containing its own Diameter identity.

5.1.2. Reporting Node Behavior

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.

> Note: The transaction state is used when the DOIC node is acting as a peer-report reporting node and needs send OC-OLR reports of type PEER_REPORT in answer messages. The peer overload reports are only included in answer messages being sent to peers that support the OLR_PEER_REPORT feature.

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

> The request does not contain an OC-Supported-Features AVP.
>
> The received request contains an OC-Supported-Features AVP with no 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 OLR_PEER_REPORT 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 relaying an answer message, a reporting node that supports the OLR_PEER_REPORT feature MUST strip any SourceID AVP from the OC-Supported-Features AVP.

When sending an answer message, a reporting node that supports the OLR_PEER_REPORT feature MUST determine if the peer to which the answer is to be sent supports the OLR_PEER_REPORT feature.

If the peer supports the OLR_PEER_REPORT feature then the reporting node MUST indicate support for the feature in the Supported-Features AVP.

If the peer supports the OLR_PEER_REPORT feature then the reporting node MUST insert the OC-SourceID AVP in the OC-Supported-Features AVP in the answer message.

If the peer supports the OLR_PEER_REPORT feature then the reporting node MUST insert the OC-Peer-Algo AVP in the OC-Supported-Features AVP. The OC-Peer-Algo AVP MUST indicate the overload abatement algorithm that the reporting node wants the reacting nodes to use should the reporting node send a peer overload report as a result of becoming overloaded.

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.

> Note: 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):

- Sequence number
- Validity Duration
- Expiration Time
- Abatement Algorithm

- 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):

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

5.2.2. Reporting Node Maintenance 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 entries defined in [I-D.ietf-dime-ovli] apply to the peer report.

5.2.3. Reacting Node Maintenance of Peer Report OCS

When a reacting node receives an OC-OLR AVP with 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 ignore the overload report.

In all cases, if the reacting node is a relay then it MUST strip the OC-OLR AVP from the message.

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.

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 reporting node peer report OCS entry, the reporting node MUST include an OC-OLR AVP with a report type of peer using the contents of the reporting node peer report 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.

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 include a host overload report, a realm overload report and/or 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 indicated 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 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 [I-D.ietf-dime-ovli].

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 adjacent node. The value of this AVP should be the same Diameter identity used as part of the CER/CEA base Diameter capabilities exchange.

This extension also adds the OC-Peer-Algo AVP to the OC-Supported-Features AVP. This AVP is used by a reporting node to indicate the abatement algorithm it will use for peer overload reports.

```
 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 TBD1) 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.

> 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 [7.6] in [I-D.ietf-dime-ovli] for a description of the OC-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.

PEER_REPORT 2 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.

6.3. OC-SourceID

The SourceID AVP (AVP code TBD2) 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 support this feature or inserted the peer report.

6.4. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
OC-SourceID	TBD1	x.x	Unsigned64		V
OC-Peer-Algo	TBD2	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 base 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-doic-rate-control]
Donovan, S. and E. Noel, "Diameter Overload Rate Control", draft-ietf-dime-doic-rate-control-01 (work in progress), March 2015.

[I-D.ietf-dime-ovli]
Korhonen, J., Donovan, S., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", draft-ietf-dime-ovli-08 (work in progress), February 2015.

[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>.

[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>.

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)
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

S. Donovan
Oracle

March 18, 2016

Diameter Agent Overload and the Peer Overload Report
draft-ietf-dime-agent-overload-04.txt

Abstract

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) [RFC7683] base solution. The extension defines the Peer overload report type. The initial use case for the Peer report is the handling of occurrences 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

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

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

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

This Internet-Draft will expire on September 19, 2016.

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. It also defines new overload report type, the Peer overload report type, that is used for handling of agent overload conditions. The Peer overload report type is defined in a generic fashion so that it can also be used for other Diameter overload scenaios.

The base Diameter overload specification [RFC7683] 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 occurrence 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 occurrences of agent overload. This report type works for the "Loss" overload mitigation algorithm defined in [RFC7683] and is expected to work for other overload abatement algorithms defined in extensions to the DOIC solution.

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.

Reporting Node

A DOIC Node that sends and overload report in a 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 [RFC7683].

3. Peer Report Use Cases

This section outlines representative use cases for the peer report used to communicate agent overload.

There are two primary classes of use cases currently identified, those involving the overload of agents and those involving overload of Diameter endpoints (Diameter Clients and Diameter Servers) that wish to use an overload algorithm suited controlling traffic sent from a peer.

3.1. Diameter Agent Overload Use Cases

The peer report needs to support the 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.

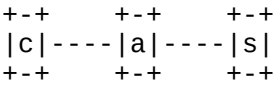

Figure 1

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.

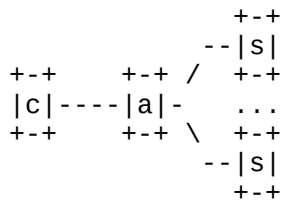

Figure 2

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 [RFC7683] or the extension draft that defines the indicated overload abatement algorithm. This will result in the throtting of the abated traffic that would have been sent to the agent, as there is no alternative route, with the appropriate indication given to the service request that resulted in the need for the Diameter transaction.

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.

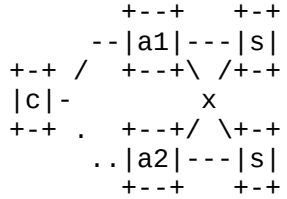

Figure 3

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.

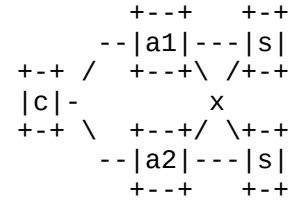

Figure 4

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.

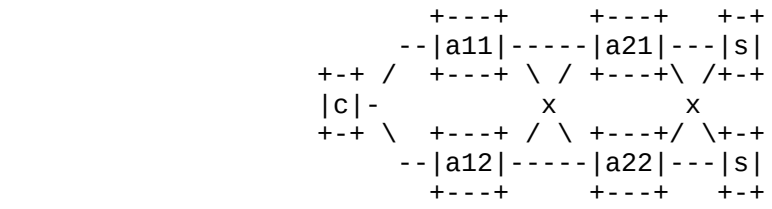

Figure 5

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.

The handling of peer 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 number of transactions. When throttling requests, an agent uses the same error responses as defined in the base DOIC specification [RFC7683].

3.2. Diameter Endpoint Use Cases

This section outlines use cases for the peer overload report 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, [I-D.ietf-dime-doic-rate-control], which is being worked on by the DIME working group as this document is being 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 an end-point in the path of a transaction are overloaded at the same time. When this occurs, Diameter entities need to handle both overload reports. In this scenario 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.

5. Peer Report Behavior

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

5.1. Capability Announcement

5.1.1. Reacting Node Behavior

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

> Note: 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 OC_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 OC_PEER_REPORT feature MUST include an OC-SourceID AVP in the OC-Supported-Features AVP with its own DiameterIdentity.

> Note: This allows the DOIC nodes 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 OC_PEER_REPORT feature.

When relaying a request that includes an OC-SourceID AVP in the OC-Supported-Features AVP, a DOIC node that supuports the OC_PEER_REPORT feature must remove the received OC-SourceID AVP and replace it with an OC-SourceID AVP containing its own Diameter identity.

5.1.2. Reporting Node Behavior

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

> Note: The transaction state is used when the DOIC node is acting as a peer-report reporting node and needs send OC-OLR reports of type PEER_REPORT in answer messages. The peer overload reports are only included in answer messages being sent to peers that support the OC_PEER_REPORT feature.

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

> The request does not contain an OC-Supported-Features AVP.
>
> The received request contains an OC-Supported-Features AVP with no OC-Feature-Vector.
>
> The received request contains an OC-Supported-Features AVP with a OC-Feature-Vector with the OC_PEER_REPORT feature bit cleared.
>
> The received request contains an OC-Supported-Features AVP with a OC-Feature-Vector with the OC_PEER_REPORT feature bit set but with an OC-SourceID AVP with a DiameterIdentity that does not match the DiameterIdentity of the peer from which the request was received.

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

When relaying an answer message, a reporting node that supports the OC_PEER_REPORT feature MUST strip any SourceID AVP from the OC-Supported-Features AVP.

When sending an answer message, a reporting node that supports the OC_PEER_REPORT feature MUST determine if the peer to which the answer is to be sent supports the OC_PEER_REPORT feature.

If the peer supports the OC_PEER_REPORT feature then the reporting node MUST indicate support for the feature in the Supported-Features AVP.

If the peer supports the OC_PEER_REPORT feature then the reporting node MUST insert the OC-SourceID AVP in the OC-Supported-Features AVP in the answer message.

If the peer supports the OC_PEER_REPORT feature then the reporting node MUST insert the OC-Peer-Algo AVP in the OC-Supported-Features AVP. The OC-Peer-Algo AVP MUST indicate the overload abatement algorithm that the reporting node wants the reacting nodes to use should the reporting node send a peer overload report as a result of becoming overloaded.

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 OC_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.

> Note: 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):

- Sequence number
- Validity Duration
- Expiration Time
- Abatement Algorithm

- 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 OC_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 DiameterIdentity 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):

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

5.2.2. Reporting Node Maintenance 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 OC_PEER_REPORT feature then the reporting node can choose to not create OCS entries.

All rules for managing the reporting node OCS entries defined in [RFC7683] apply to the peer report.

5.2.3. Reacting Node Maintenance of Peer Report OCS

When a reacting node receives an OC-OLR AVP with 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 DiameterIdentity 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 ignore the overload report.

In all cases, if the reacting node is a relay then it MUST strip the OC-OLR AVP from the message.

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 DiameterIdentity received in the SourceID AVP matches the DiameterIdentity 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.

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

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

The reporting node MUST include its DiameterIdentity in the OC-SourceID AVP in the OC-OLR AVP. This is used by DOIC nodes that support the OC_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.

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 include a host overload report, a realm overload report and/or 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 indicated 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 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 [RFC7683].

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

This extension also adds the OC-Peer-Algo AVP to the OC-Supported-Features AVP. This AVP is used by a reporting node to indicate the abatement algorithm it will use for peer overload reports.

```
 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.

OC_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 TBD1) 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.

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 OC_PEER_REPORT feature extends the base Diameter overload specification by defining a new overload report type of "peer". See section [7.6] in [RFC7683] for a description of the OC-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 DiameterIdentity.

```
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.

PEER_REPORT 2 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.

6.3. OC-SourceID

The SourceID AVP (AVP code TBD2) 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 support this feature or inserted the peer report.

6.4. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
OC-SourceID	TBD1	x.x	DiameterIdentity		V
OC-Peer-Algo	TBD2	x.x	Unsigned64		V

7. IANA Considerations

7.1. AVP codes

New AVPs defined by this specification are listed in Section 6. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

7.2. New registries

There are no new IANA registries introduced by this document.

The values used for the OC-Peer-Algo AVP are the subset of the "OC-Feature-Vector AVP Values (code 622)" registry. Only the values in that registry that apply to overload abatement algorithms apply to the OC-Peer-Algo AVP.

8. Security Considerations

Agent overload is an extension to the base Diameter overload mechanism. As such, all of the security considerations outlined in [RFC7683] 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-doic-rate-control]
Donovan, S. and E. Noel, "Diameter Overload Rate Control", draft-ietf-dime-doic-rate-control-01 (work in progress), March 2015.

[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>.

[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>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

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)
Internet-Draft
Updates: RFC7683 (if approved)
Intended status: Standards Track
Expires: September 23, 2017

S. Donovan
Oracle

March 22, 2017

Diameter Agent Overload and the Peer Overload Report
draft-ietf-dime-agent-overload-11.txt

Abstract

This specification documents an extension to RFC 7683 (Diameter Overload Indication Conveyance (DOIC)) base solution. The extension defines the Peer overload report type. The initial use case for the Peer report is the handling of occurrences 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

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

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

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

This Internet-Draft will expire on September 23, 2017.

Copyright Notice

Table of Contents

1. Introduction

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) [RFC7683] base solution. The extension defines the Peer overload report type. The initial use case for the Peer report is the handling of occurrences of overload of a Diameter agent.

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. It also defines new overload report type, the Peer overload report type, that is used for handling of agent overload conditions. The Peer overload report type is defined in a generic fashion so that it can also be used for other Diameter overload scenarios.

The base Diameter overload specification [RFC7683] 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 occurrence as close to the source of the Diameter traffic as 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 occurrences of agent overload. This report type works for the "Loss" overload mitigation algorithm defined in [RFC7683] and is expected to work for other overload abatement algorithms defined in extensions to the DOIC solution.

2. Terminology and Abbreviations

AVP

Attribute Value Pair

Diameter Node

A [RFC7683] Diameter Client, an [RFC7683] Diameter Server, and [RFC7683] Diameter Agent.

Diameter Endpoint

An [RFC7683] Diameter Client and [RFC7683] Diameter Server.

Diameter Agent

An [RFC7683] Diameter Agent.

Reporting Node

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

Reacting Node

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

DOIC Node

A Diameter Node that supports the DOIC solution defined in [RFC7683].

3. Peer Report Use Cases

This section outlines representative use cases for the peer report used to communicate agent overload.

There are two primary classes of use cases currently identified, those involving the overload of agents and those involving overload of Diameter endpoints. In both cases the goal is to use an overload algorithm that controls traffic sent towards peers.

3.1. Diameter Agent Overload Use Cases

The peer report needs to support the following use cases.

In the figures in this section, elements labeled "c" are Diameter Clients, elements labeled "a" are Diameter Agents and elements labeled "s" are Diameter Servers.

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.

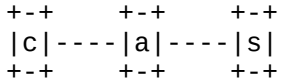

Figure 1

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.

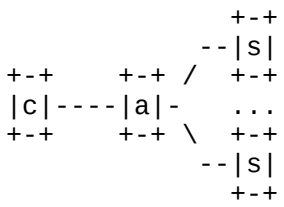

Figure 2

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 [RFC7683] or the extension draft that defines the indicated overload abatement algorithm. This will result in the throttling of the abated traffic that would have been sent to the agent, as there is no alternative route. The client sends an appropriate error response to the originator of the request.

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.

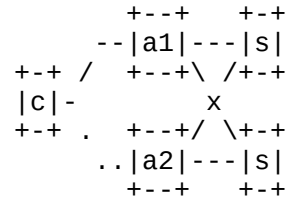

Figure 3

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 traffic sent through each client.

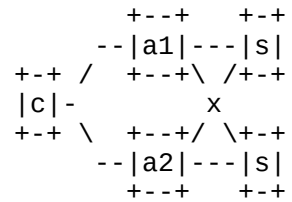

Figure 4

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 the standby connection to active. This will result in all traffic being routed 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 3.1.1 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. An example of this type of deployment includes when there are Diameter agents between administrative domains.

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.

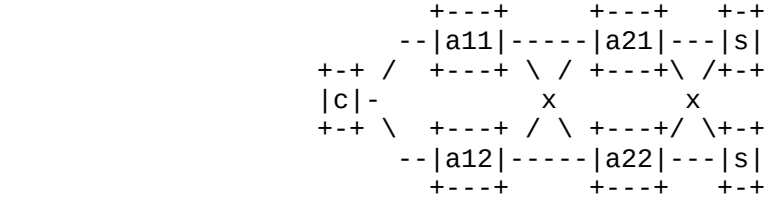

Figure 5

Handling of overload of one or both of agents a11 or a12 in this case is equivalent to that discussed in Section 3.1.2.

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.

The handling of peer overload reports is similar to that discussed in Section 3.1.2. 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 number of transactions. When throttling requests, an agent uses the same error responses as defined in the base DOIC specification [RFC7683].

3.2. Diameter Endpoint Use Cases

This section outlines use cases for the peer overload report 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, [I-D.ietf-dime-doic-rate-control], which is being worked on by the DIME working group as this document is being 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 an end-point in the path of a transaction are overloaded at the same time. When this occurs, Diameter entities need to handle both overload reports. In this scenario 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. In this scenario, when doing abatement on the PEER report, the reacting node SHOULD take into consideration the number of messages already throttled by the handling of the HOST/REALM report abatement.

> Note: The goal is to avoid traffic oscillations that might result from throttling of messages for both the HOST/REALM overload reports and the PEER overload reports. This is especially a concern if both reports indicate the LOSS abatement algorithm.

5. Peer Report Behavior

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

5.1. Capability Announcement

5.1.1. Reacting Node Behavior

When sending a Diameter request a DOIC Node that supports the OC_PEER_REPORT (as defined in Section 6.1.1) feature MUST include in the OC-Supported-Features AVP an OC-Feature-Vector AVP with the OC_PEER_REPORT bit set.

When sending a request a DOIC Node that supports the OC_PEER_REPORT feature MUST include a SourceID AVP in the OC-Supported-Features AVP with its own DiameterIdentity.

When a Diameter Agent relays a request that includes a SourceID AVP in the OC-Supported-Features AVP, if the Diameter Agent supports the OC_PEER_REPORT feature then it MUST remove the received SourceID AVP and replace it with a SourceID AVP containing its own DiameterIdentity.

5.1.2. Reporting Node Behavior

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

> Note: The transaction state is used when the DOIC Node is acting as a peer-report reporting node and needs send OC-OLR reports of type peer in answer messages. The peer overload reports are only included in answer messages being sent to peers that support the OC_PEER_REPORT feature.

The peer supports the OC_PEER_REPORT feature if the received request contains an OC-Supported-Features AVP with the OC-Feature-Vector with the OC_PEER_REPORT feature bit set and with a SourceID AVP with a value that matches the DiameterIdentity of the peer from which the request was received.

When an agent relays an answer message, a reporting node that supports the OC_PEER_REPORT feature MUST strip any SourceID AVP from the OC-Supported-Features AVP.

When sending an answer message, a reporting node that supports the OC_PEER_REPORT feature MUST determine if the peer to which the answer is to be sent supports the OC_PEER_REPORT feature.

If the peer supports the OC_PEER_REPORT feature then the reporting node MUST indicate support for the feature in the OC-Supported-Features AVP.

If the peer supports the OC_PEER_REPORT feature then the reporting node MUST insert the SourceID AVP in the OC-Supported-Features AVP in the answer message.

If the peer supports the OC_PEER_REPORT feature then the reporting node MUST insert the OC-Peer-Algo AVP in the OC-Supported-Features AVP. The OC-Peer-Algo AVP MUST indicate the overload abatement algorithm that the reporting node wants the reacting nodes to use should the reporting node send a peer overload report as a result of becoming overloaded.

5.2. Peer 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.

This is an extension of the OCS handling defined in [RFC7683].

5.2.1.1. Reporting Node Peer Report OCS

A DOIC Node that supports the OC_PEER_REPORT feature SHOULD maintain Reporting Node OCS, as defined in [RFC7683] and extended here.

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

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

5.2.1.2. Reacting Node Peer Report OCS

In addition to OCS maintained as defined in [RFC7683], a reacting node that supports the OC_PEER_REPORT feature maintains the following OCS per supported Diameter application:

A peer-type OCS entry for each peer to which it sends requests.

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

The peer-type OCS entry include the following information (the actual information stored is an implementation decision):

Sequence number (as received in the OC-OLR AVP).

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).

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

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

5.2.2. Reporting Node Maintenance of Peer Report OCS

All rules for managing the reporting node OCS entries defined in [RFC7683] apply to the peer report.

5.2.3. Reacting Node Maintenance of Peer Report OCS

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

If a reacting node receives an OC-OLR AVP of type peer and the SourceID matches the DiameterIdentity of the Diameter peer from which the response message was received then the report was generated by a Diameter peer.

If a reacting node receives an OC-OLR AVP of type peer and the SourceID does not match the DiameterIdentity of the Diameter peer from which the response message was received then the reacting node MUST ignore the overload report.

Note: Under normal circumstances, a Diameter node will not add a peer report when sending to a peer that does not support this extension. This requirement is to handle the case where peer reports are erroneously or maliciously inserted into response messages.

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

The peer report is for an existing overload condition if the reacting node has an OCS that matches the received peer report. For a peer report, this means it matches the Application-ID and the peer's DiameterIdentity in an existing OCS entry.

If the peer report is for an existing overload condition then it MUST determine if the peer report is a retransmission or an update to the existing OLR.

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

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

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

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

The reporting node determines if a peer supports the OC_PEER_REPORT feature based on the indication recorded in the reporting node's transaction state.

The reporting node MUST include its DiameterIdentity in the SourceID AVP in the OC-OLR AVP. This is used by DOIC Nodes that support the OC_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.

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 include a host overload report, a realm overload report and/or a peer overload report.

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

In all cases, if the reacting node is an agent then it MUST strip the Peer Report OC-OLR AVP from the message.

If the request matches an active OCS then the reacting node MUST apply abatement treatment to the request. The abatement treatment applied depends on the abatement algorithm indicated 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 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 response as defined in [RFC7683].

In the case that the OCS entry validity duration expires or has a validity duration of zero ("0"), meaning that if the reporting node has explicitly signaled the end of the overload condition then abatement associated with the OCS entry 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 agent's peer.

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

This AVP contains the DiameterIdentity of the node that supports the OC_PEER_REPORT feature. This AVP is used to determine if support for the peer overload report is in an adjacent node. The value of this AVP should be the same Diameter identity used as part of the Diameter Capabilities Exchange procedure defined in [RFC7683].

This extension also adds the OC-Peer-Algo AVP to the OC-Supported-Features AVP. This AVP is used by a reporting node to indicate the abatement algorithm it will use for peer overload reports.

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

6.1.1. OC-Feature-Vector AVP

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

OC_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 AVP

The OC-Peer-Algo AVP (AVP code TBD1) 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 for this AVP.

6.2. OC-OLR AVP

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

The OC_PEER_REPORT feature extends the base Diameter overload specification by defining a new overload report type of "peer". See section [7.6] in [RFC7683] for a description of the OC-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 report, 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 diverted or throttled.

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

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

6.2.1. OC-Report-Type AVP

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

PEER_REPORT 2 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.

6.3. SourceID AVP

The SourceID AVP (AVP code TBD2) is of type DiameterIdentity and is inserted by a Diameter node to indicate the source of the AVP in which it is a part.

In the case of peer reports, the SourceID AVP indicates the node that supports this feature (in the OC-Supported-Features AVP) or the node that generates an overload with a report type of peer (in the OC-OLR AVP).

It contains the DiameterIdentity of the inserting node. This is used by other Diameter nodes to determine the node that inserted the enclosing AVP that contains the SourceID AVP.

6.4. Attribute Value Pair Flag Rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
OC-Peer-Algo	TBD1	6.1.2	Unsigned64		V
SourceID	TBD2	6.3	DiameterIdentity		V

7. IANA Considerations

7.1. AVP Codes

New AVPs defined by this specification are listed in Section 6.4. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

One new OC-Report-Type AVP value is defined in Section 6.2.1

7.2. New Registries

There are no new IANA registries introduced by this document.

The values used for the OC-Peer-Algo AVP are the subset of the "OC-Feature-Vector AVP Values (code 622)" registry. Only the values in that registry that apply to overload abatement algorithms apply to the OC-Peer-Algo AVP.

8. Security Considerations

Agent overload is an extension to the base Diameter overload mechanism. As such, all of the security considerations outlined in [RFC7683] 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.

The impacts of this attack, as well as the mitigation strategies, are the same as outlined in [RFC7683].

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. References

10.1. Informative 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>.

[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>.

10.2. Normative References

[I-D.ietf-dime-doic-rate-control] Donovan, S. and E. Noel, "Diameter Overload Rate Control", draft-ietf-dime-doic-rate-control-03 (work in progress), March 2016.

[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>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

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)
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

S. Donovan, Ed.
Oracle
E. Noel
AT&T Labs

March 18, 2016

Diameter Overload Rate Control
draft-ietf-dime-doic-rate-control-03.txt

Abstract

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) [RFC7683] 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 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

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

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

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

This Internet-Draft will expire on September 19, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document defines a new Diameter overload control abatement algorithm.

The base Diameter overload specification [RFC7683] 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 abating (diverting or 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 cannot guarantee the load presented to the server remains below a specific rate level. The loss algorithm can be slow to protect the stability of reporting nodes when subjected with rapidly changing loads.

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 abate (diverting or throttling) 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 for a 10% reduction in traffic. This results, in this example, in the reacting node sending 900 Diameter transactions per second, abating 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 abate 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%. This control feedback loop has the potential to make the situation worse.

One of the benefits of a rate based algorithm is that it better handles spikes in traffic. Instead of sending a request to reduce traffic by a percentage, 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 node will send the 90 regardless of whether it is receiving 100 or 1000 service requests per second.

This document extends the base DOIC solution [RFC7683] 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 definition of the rate abatement algorithm is copied almost verbatim from the SOC document [RFC7415], with changes focused on making the wording consistent with the DOIC solution and the Diameter protocol.

2. Terminology and Abbreviations

Diameter Node

A RFC6733 Diameter Client, RFC6733 Diameter Server, or RFC6733 Diameter Agent.

Diameter Endpoint

An RFC6733 Diameter Client or RFC6733 Diameter Server.

DOIC Node

A Diameter Node that supports the DOIC solution defined in [RFC7683].

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 [RFC7683].

2. Realm - Overload of a specific Diameter Application at a specific Diameter Realm as defined in [RFC7683].

3. Peer - Overload of a specific Diameter peer as defined in [I-D.ietf-dime-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

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 [RFC7683].

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 abatement algorithm 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 [RFC7683].

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 is 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 procedures defined in [RFC7683].
>
> For peer reports the selected algorithm is reflected in the OC-Peer-Algo AVP sent as part of the OC-Supported-Features AVP included answer messages for transactions where the request contained an OC-Supported-Features AVP. This is per the procedures defined in [I-D.ietf-dime-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 [RFC7683] 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 [RFC7683] where 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 DiameterID of the target of the rate OLR.

A reporting node that supports the rate abatement algorithm MUST include the specified rate in the abatement algorithm specific portion of the reporting node rate OCS when sending a rate OLR.

All other elements for the OCS defined in [RFC7683] and [I-D.ietf-dime-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 indicate rate as the selected abatement algorithm in the reacting node OCS when receiving a rate OLR.

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

All other elements for the OCS defined in [RFC7683] and [I-D.ietf-dime-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 selecting the rate algorithm in the response 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:

- For Host reports the target is the DiameterIdentity contained in the Origin-Host AVP received in the request.
- For Realm reports the target is the DiameterIdentity contained in the Origin-Realm AVP received in the request.
- For Peer reports the target is the DiameterIdentity of the Diameter Peer from which the request was received.

5.4. Reacting Node Maintenance of Overload Control State

When receiving an answer message indicating that the reacting node has selected the rate algorithm, a reaction node MUST indicate the rate abatement algorithm in the reacting node OCS entry for the reporting node.

A reacting node receiving an overload report for the rate abatement algorithm MUST save the rate received in the OC-Maximum-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 abatement algorithm, 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.

When sending an overload report for the Rate algorithm, the OC-Maximum-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 MAY use the algorithm detailed in Section 6.

> Note: Other algorithms for controlling the rate can be implemented by the reacting node as long as they result in the correct rate of traffic being sent to the reporting node.

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 [RFC7683] and [I-D.ietf-dime-agent-overload] apply.

6. Rate Abatement Algorithm AVPs

6.1. OC-Supported-Features AVP

The rate algorithm does not add any new 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-Maximum-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-Maximum-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 [RFC7683] apply to the rate control algorithm. The peer report type defined in [I-D.ietf-dime-agent-overload] also applies to the rate control algorithm.

6.2.1. OC-Maximum-Rate AVP

The OC-Maximum-Rate AVP (AVP code TBD1) 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.

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-Maximum-Rate	TBD1	x.x	Unsigned64		V

7. Rate Based Abatement Algorithm

This section is pulled from [RFC7415], with minor changes needed to make it apply to the Diameter protocol.

7.1. Overview

The reporting node is the one protected by the overload control algorithm defined here. The reacting node is the one that abates traffic towards the server.

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

Then periodically, the reporting node relies on internal measurements (e.g. CPU utilization or queuing delay) to evaluate its overload state and estimate a target maximum 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 maximum Diameter request rate, each reacting node applies abatement treatment for 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 maximum 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 reporting node must allocate a portion of the target Diameter request rate to each of its reacting nodes. The reporting node may set the same rate for every reacting node, or may set different rates for different reacting node.

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

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 maximum 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 maximum Diameter request rate and to notify them that the rate overload abatement is in effect.

The reporting node MUST use the OC-Maximum-Rate AVP defined in this specification to communicate a target maximum 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 can use any algorithm that limits the message rate to the OC-Maximum-Rate AVP value in units of messages per second. For ease of discussion, we define T = 1/[OC-Maximum-Rate] as the target inter-Diameter request interval. 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 reacting node, or they may be negotiated independently between reacting node and reporting node.

In either case, the coordination is out of scope for this document. The default algorithms presented here (one with and one without provisions for prioritizing traffic) are only examples.

To apply abatement treatment to 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 that forward messages in conformance with the upper bound of 1/T messages per second.

The default Leaky Bucket algorithm presented here is based on [ITU-T Rec. I.371] Appendix A.2. The algorithm makes it possible for reacting nodes 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 apply abatement treatment to 100% of Diameter requests destined to the overloaded reporting node. However, when the OC-Validity-Duration value is 0, the reacting node should stop applying abatement treatment.

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 abatement treatment is applied to the Diameter request.

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).

In situations where reacting nodes are configured with some knowledge about the reporting node (e.g., operator pre-provisioning), it can be beneficial to choose a value of TAU based on how many reacting nodes will be sending requests to the reporting node.

Reporting nodes with a very large number of reacting nodes, 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 abatement treatment is applied to the new Diameter request 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 abatement 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:

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

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

- 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 queuing.

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 Diameter requests
// TAU2: tolerance parameter of priority Diameter requests
// ta: arrival time of the most recent arrival
// LCT: arrival time of last Diameter 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 Diameter request
  // Update X and LCT
  X = max (0, Xp) + T;
  LCT = ta;
} else {
  // Apply abatement treatment to Diameter 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 resonance can occur, then a simple way to avoid resonance is to randomize the bucket occupancy at two appropriate points -- at the activation of control and whenever the bucket empties -- as described below.

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 let u be set to a random value 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 activate control at the same time due to an extremely high request rate, then the time until the first request admitted by each reacting node 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

The rate overload abatement mechanism is an extension to the based Diameter overload mechanism. As such, all of the security considerations outlined in [RFC7683] apply to the rate overload abatement mechanism.

10. Acknowledgements

11. References

11.1. Normative References

[I-D.ietf-dime-agent-overload]
Donovan, S., "Diameter Agent Overload", draft-ietf-dime-agent-overload-00 (work in progress), December 2014.

[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>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

11.2. Informative References

[RFC7415] Noel, E. and P. Williams, "Session Initiation Protocol (SIP) Rate Control", RFC 7415, DOI 10.17487/RFC7415, February 2015, <http://www.rfc-editor.org/info/rfc7415>.

Authors' Addresses

Steve Donovan (editor)
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

Email: ecnoel@research.att.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: August 15, 2019

S. Donovan, Ed.
Oracle
E. Noel
AT&T Labs

February 11, 2019

Diameter Overload Rate Control
draft-ietf-dime-doic-rate-control-11

Abstract

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) [RFC7683] 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 to the DOIC reporting node.

Requirements

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.

Status of This Memo

Table of Contents

1. Introduction

This document defines a new Diameter overload control abatement algorithm, the "rate" algorithm.

The base Diameter overload specification [RFC7683] defines the "loss" algorithm as the default Diameter overload abatement algorithm. The loss algorithm allows a reporting node (see Section 2) to instruct a reacting node (see Section 2) to reduce the amount of traffic sent to the reporting node by abating (diverting or 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 changes quickly. For instance, if the service requests that result in Diameter transactions increase quickly then the loss algorithm cannot guarantee the load presented to the server remains below a specific rate level. The loss algorithm can be slow to ensure the stability of reporting nodes when subjected to rapidly-changing loads. The "loss" algorithm errs both in throttling too much when there is a dip in offered load, and throttling not enough when there is a spike in offered load.

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 reporting node. If the reporting 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 when the loss algorithm is used as Diameter overload abatement algorithm. Assume for this discussion that the reporting node requests a 10% reduction. The reacting node will then abate (diverting or throttling) ten Diameter transactions a second, sending the remaining 90 transactions per second to the reporting 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 node's request for a 10% reduction in traffic. This results, in this example, in the reacting node sending 900 Diameter transactions per second, abating 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 abate 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%. This control feedback loop has the potential to make the situation worse by causing wide fluctuations in traffic on multiple nodes in the Diameter network.

One of the benefits of a rate-based algorithm over the loss algorithm is that it better handles spikes in traffic. Instead of sending a request to reduce traffic by a percentage, 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 node will send the 90 regardless of whether it is receiving 100 or 1000 service requests per second.

It should be noted that one of the implications of the rate-based algorithm is that the reporting node needs to determine how it wants to distribute its load over the set of reacting nodes from which it is receiving traffic. For instance, if the reporting node is receiving Diameter traffic from 10 reacting nodes and has a capacity of 100 transactions per second then the reporting node could choose to set the rate for each of the reacting nodes to 10 transactions per second. This, of course, is assuming that each of the reacting nodes has equal performance characteristics. The reporting node could also choose to have a high capacity reacting node send 55 transactions per second and the remaining 9 low capacity reacting nodes send 5 transactions per second. The ability of the reporting node to specify the amount of traffic on a per-reacting-node basis implies that the reporting node must maintain state for each of the reacting nodes. This state includes the current allocation of Diameter traffic to that reacting node. If the number of reacting nodes changes, either because new nodes are added, nodes are removed from service or nodes fail, then the reporting node will need to redistribute the maximum Diameter transactions over the new set of reacting nodes.

This document extends the base Diameter Overload Indication Conveyance (DOIC) solution [RFC7683] to add support for the rate-based overload abatement algorithm.

This document draws heavily on work in the SIP Overload Control working group. The definition of the rate abatement algorithm is

copied almost verbatim from the SIP Overload Control (SOC) document [RFC7415], with changes focused on making the wording consistent with the DOIC solution and the Diameter protocol.

2. Terminology

Diameter Node

A Diameter Client, Diameter Server, or Diameter Agent. [RFC6733]

Diameter Endpoint

A Diameter Client or Diameter Server. [RFC6733]

DOIC Node

A Diameter Node that supports the DOIC solution defined in [RFC7683].

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:

HOST_REPORT 0 Overload of a specific Diameter Application at a specific Diameter Node as defined in [RFC7683]

REALM_REPORT 1 Overload of a specific Diameter Application at a specific Diameter Realm as defined in [RFC7683]

PEER_REPORT 2 Overload of a specific Diameter peer as defined in [I-D.ietf-dime-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

This document defines the rate abatement algorithm (referred to as rate in this document) feature. Support for the rate feature by a DOIC node will be indicated by a new value of the OC-Feature-Vector AVP, as described in Section 6.1.1, per the rules defined in [RFC7683].

Since all nodes that support DOIC are required to support the loss algorithm, DOIC nodes supporting the rate feature will support both the loss and rate-based abatement algorithms.

DOIC reacting nodes supporting the rate feature MUST indicate support for both the loss and rate algorithms in the OC-Feature-Vector AVP and MAY indicate support for other algorithms.

As defined in [RFC7683], a DOIC reporting node supporting the rate feature selects a single abatement algorithm in the OC-Feature-Vector AVP and OC-Peer-Algo AVP in the answer message sent to the DOIC reacting nodes.

A reporting node can 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 is reflected in the OC-Feature-Vector AVP sent as part of the OC-Supported-Features AVP included in answer messages for transaction where the request contained an OC-Supported-Features AVP. This is per the procedures defined in [RFC7683].

For peer reports the selected algorithm is reflected in the OC-Peer-Algo AVP sent as part of the OC-Supported-Features AVP included answer messages for transactions where the request contained an OC-Supported-Features AVP. This is per the procedures defined in [I-D.ietf-dime-agent-overload].

5. Overload Report Handling

This section describes any changes to the behavior defined in [RFC7683] 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 Overload Control State (OCS) for each reacting node to which it sends a rate Overload Report (OLR).

This is different from the behavior defined in [RFC7683] where a reporting node sends a single loss percentage 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 DiameterIdentity of the target of the rate OLR.

The rate OCS entry SHOULD include the rate allocated to the reacting note.

A reporting node that has selected the rate overload abatement algorithm MUST indicate the rate requested to be applied by DOIC reacting nodes in the OC-Maximum-Rate AVP included in the OC-OLR AVP.

All other elements for the OCS defined in [RFC7683] and [I-D.ietf-dime-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 indicate rate as the selected abatement algorithm in the reacting node OCS based on the OC-Feature-Vector AVP or the OC-Peer-Algo AVP in the received OC-Supported-Features AVP.

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

All other elements for the OCS defined in [RFC7683] and [I-D.ietf-dime-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 and MUST indicate the selected rate in the resulting reporting node OCS entries.

When selecting the rate algorithm in the response 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:

- For Host reports, the target is the DiameterIdentity contained in the Origin-Host AVP received in the request.
- For Realm reports, the target is the DiameterIdentity contained in the Origin-Realm AVP received in the request.
- For Peer reports, the target is the DiameterIdentity of the Diameter Peer from which the request was received.

A reporting node that receives a capability announcement from a new reacting node, meaning a reacting node for which it does not have an OCS entry, and the reporting node chooses the rate algorithm for that reacting node may need to recalculate the rate to be allocated to all reacting nodes. Any changed rate values will be communicated in the next OLR sent to each reacting node.

5.4. Reacting Node Maintenance of Overload Control State

When receiving an answer message indicating that the reporting node has selected the rate algorithm, a reacting node MUST indicate the rate abatement algorithm in the reacting node OCS entry for the reporting node.

A reacting node receiving an overload report for the rate abatement algorithm MUST save the rate received in the OC-Maximum-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 abatement algorithm, 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.

> Note: It is also possible for the reporting node to send overload reports with the rate algorithm indicated even when the reporting node is not in an overloaded state. This could be a strategy to proactively avoid entering into an overloaded state. Whether to do so is up to local policy.

When sending an overload report for the rate algorithm, the OC-Maximum-Rate AVP MUST be included in the OC-OLR AVP and the OC-Reduction-Percentage AVP MUST NOT be 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 can choose to use the algorithm detailed in Section 7.

Other algorithms for controlling the rate MAY be implemented by the reacting node. Any algorithm implemented MUST correctly limit the maximum rate of traffic being sent to the reporting node.

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 [RFC7683] and [I-D.ietf-dime-agent-overload] apply.

6. Rate Abatement Algorithm AVPs

6.1. OC-Supported-Features AVP

The rate algorithm does not add any new 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 capability to the OC-Feature-Vector AVP.

OLR_RATE_ALGORITHM (bit 2)

> Bit 2 is assigned to the rate overload abatement algorithm. When this flag is set by the overload control endpoint it indicates that the DOIC Node supports the rate overload abatement algorithm.

6.2. OC-OLR AVP

This extension defines the OC-Maximum-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 ]
           [ SourceID ]
           [ OC-Maximum-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 [RFC7683] apply to the rate control algorithm. The peer report type defined in [I-D.ietf-dime-agent-overload] also applies to the rate control algorithm.

6.2.1. OC-Maximum-Rate AVP

The OC-Maximum-Rate AVP (AVP code TBD1) is of type Unsigned32 and describes the maximum rate that the sender is requested to send traffic. This is specified in terms of requests per second.

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	MUST NOT
OC-Maximum-Rate	TBD1	6.2	Unsigned32		V

7. Rate-Based Abatement Algorithm

This section is pulled from [RFC7415], with minor changes needed to make it apply to the Diameter protocol.

7.1. Overview

The reporting node is the one protected by the overload control algorithm defined here. The reacting node is the one that abates traffic towards the server.

Following the procedures defined in [RFC7683], the reacting node and reporting node signal their support for rate-based overload control.

Then periodically, the reporting node relies on internal measurements (e.g. CPU utilization or queuing delay) to evaluate its overload state and estimate a target maximum 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 maximum Diameter request rate, each reacting node applies overload abatement for 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 maximum 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 reporting node must allocate a portion of the target Diameter request rate to each of its reacting nodes. The reporting node may set the same rate for every reacting node, or may set different rates for different reacting node.

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

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 value of OC-Maximum-Rate AVP (in request messages per second) for the rate algorithm provides a loose upper bound on the traffic sent by the reacting node to the reporting node.

In other words, when multiple reacting nodes are being controlled by an overloaded reporting node, at any given time, some reporting nodes may receive requests at a rate below its target maximum 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 or a lower rate.

Upon detection of overload, and the determination to invoke overload controls, the reporting node follows the specifications in [RFC7683] to notify its clients of the allocated target maximum Diameter request rate and to notify them that the rate overload abatement is in effect.

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

7.3. Reacting Node Behavior

7.3.1. Default Algorithm for Rate-based Control

A reference algorithm is shown below.

Note that use of // below inidcates a comment.

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

In determining whether or not to transmit a specific message, the reacting node can use any algorithm that limits the message rate to the OC-Maximum-Rate AVP value in units of messages per second. For ease of discussion, we define T = 1/[OC-Maximum-Rate] as the target inter-Diameter request interval. 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 reacting node, or they may be negotiated independently between reacting node and reporting node.

In either case, the coordination is out of scope for this document. The default algorithms presented here (one with and one without provisions for prioritizing traffic) are only examples.

To apply abatement treatment to 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 that forward messages in conformance with the upper bound of 1/T messages per second.

The default Leaky Bucket algorithm presented here is based on [ITU-T Rec. I.371] Appendix A.2. The algorithm makes it possible for reacting nodes 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 apply abatement treatment to 100% of Diameter requests destined to the overloaded reporting node. However, when the OC-Validity-Duration value is 0, the reacting node should stop applying abatement treatment.

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 abatement treatment is applied to the Diameter request.

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).

In situations where reacting nodes are configured with some knowledge about the reporting node and other traffic sources (e.g., operator pre-provisioning), it can be beneficial to choose a value of TAU based on how many reacting nodes will be sending requests to the reporting node.

Reporting nodes with a very large number of reacting nodes, 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 abatement treatment is applied to the new Diameter request 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 abatement 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.

7.3.2. Priority Treatment

A reference algorithm is shown below.

Priority case:

```
// T: inter-transmission interval, set to 1 / OC-Maximum-Rate
// TAU1: tolerance parameter of no priority Diameter requests
// TAU2: tolerance parameter of priority Diameter requests
// ta: arrival time of the most recent arrival
// LCT: arrival time of last Diameter 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 Diameter request
   // Update X and LCT
   X = max (0, Xp) + T;
   LCT = ta;
 } else {
   // Apply abatement treatment to Diameter request
   // Do not update X and LCT
 }
```

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:

- All new requests would be admitted when the leaky bucket counter is at or below TAU1,
- Only higher priority requests would be admitted when the leaky bucket counter is between TAU1 and TAU2,
- 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.

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 queuing.

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.

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 resonance can occur, then a simple way to avoid resonance is to randomize the bucket occupancy at two appropriate points -- at the activation of control and whenever the bucket empties -- as described below.

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 let u be set to a random value uniformly distributed between -1/2 and +1/2

Then (only) if the arrival is admitted, increase the bucket content 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 activate control at the same time due to an extremely high request rate, then the time until the first request admitted by each reacting node 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

8.1. AVP Codes

New AVPs defined by this specification are listed in Section 6. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

8.2. OC-Supported-Features

As indicated in Section 6.1.1, a new allocation is required in the OC-Feature-Vector AVP.

8.3. New DOIC report types

All DOIC report types defined in the future MUST indicate whether or not the rate algorithm can be used with that report type.

9. Security Considerations

The rate overload abatement mechanism is an extension to the base Diameter overload mechanism. As such, all of the security considerations outlined in [RFC7683] apply to the rate overload abatement mechanism.

In addition, the rate algorithm could be used to handle DoS attacks more effectively than the loss algorithm.

10. Acknowledgements

Lionel Morand for his contributions to the document.

11. References

11.1. Normative References

[I-D.ietf-dime-agent-overload]
Donovan, S., "Diameter Agent Overload", draft-ietf-dime-agent-overload-00 (work in progress), December 2014.

[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>.

[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>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <https://www.rfc-editor.org/info/rfc7683>.

[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>.

11.2. Informative References

[Erramilli]
Erramilli, A. and L. Forys, "Traffic Synchronization Effects In Teletraffic Systems", 1991.

[RFC7415] Noel, E. and P. Williams, "Session Initiation Protocol (SIP) Rate Control", RFC 7415, DOI 10.17487/RFC7415, February 2015, <https://www.rfc-editor.org/info/rfc7415>.

Authors' Addresses

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

Email: srdonovan@usdonovans.com

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

Email: ecnoel@research.att.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2016

S. Donovan
Oracle

March 10, 2016

Diameter Routing Message Priority
draft-ietf-dime-drmp-04.txt

Abstract

When making routing and resource allocation decisions, Diameter nodes currently have no generic mechanism to determine the relative priority of Diameter messages. This document addresses this by defining a mechanism to allow Diameter endpoints to indicate the relative priority of Diameter transactions. With this information Diameter nodes can factor that priority into routing, resource allocation and overload abatement decisions.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2016.

Copyright Notice

Table of Contents

1. Introduction

The DOIC solution [RFC7683] for Diameter overload control introduces scenarios where Diameter routing decisions made by Diameter nodes can be influenced by the overload state of other Diameter nodes. This includes the scenarios where Diameter endpoints and Diameter agents can throttle requests as a result of the target for the request being overloaded.

With currently available mechanisms these Diameter nodes do not have a mechanism to differentiate request message priorities when making these throttling decisions. As such, all requests are treated the same, meaning that all requests have the same probability of being throttled.

There are scenarios where treating all requests the same can cause issues. For instance it might be considered important to reduce the probability of transactions involving first responders during a period of heavy signaling resulting from a natural disaster being throttled during overload scenarios.

This document defines a mechanism that allows Diameter nodes to indicate the relative priority of Diameter transactions. With this information other Diameter nodes can factor the relative priority of requests into routing and throttling decisions.

2. Terminology and Abbreviations

Diversion

As defined in [RFC7683]. An overload abatement treatment where the reacting node selects alternate destinations or paths for requests.

DOIC

Diameter Overload Indication Conveyance.

DRMP

Diameter Routing Message Priority.

Overload Abatement

As defined in [RFC7683]. Reaction to receipt of an overload report resulting in a reduction in traffic sent to the reporting node. Abatement actions include diversion and throttling.

Priority

The relative importance of a Diameter message. A lower priority value implies a higher relative importance of the message.

Throttling

As defined in [RFC7683]. An abatement treatment that limits the number of requests sent by the DIOC 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. Problem Statement

With the introduction of overload control mechanisms, Diameter nodes will be required to make decisions regarding which Diameter request messages should be throttled as a result of overloaded Diameter nodes.

There is currently no generic mechanism to indicate which request messages should be given preferential treatment when these throttling decisions are made.

As a result, all messages are treated equally and, as such, have an equal probability of being throttled.

There are a number of scenarios where it is appropriate for an application to mark a request as being of a higher priority than other application requests. These are discussed in the next section.

This document defines a mechanism for applications to indicate priority for individual transactions, reducing the probability of those transactions being throttled if there are other lower priority transactions that are eligible for throttling treatment.

While the primary usage of DRMP defined priorities is for input to Diameter overload control related throttling decisions, it is also expected that the priority information could also be used for other routing related functionality. This might include giving higher priority transactions preferential treatment when selecting routes.

It is also envisioned that DRMP priority information could be used by Diameter endpoints to make resource allocation decisions. For instance, a Diameter Server might choose to use the priority information to treat higher priority requests ahead of lower priority requests.

> Note: There are a number of application specific definitions indicating various views of application level priority for different requests. Using these application specific priority AVPs as input to throttling and other Diameter routing decisions

would require Diameter agents to understand all applications and do application specific parsing of all messages in order to determine the priority of individual messages. This is considered an unacceptable level of complexity to put on elements whose primary responsibility is to route Diameter messages.

5. Use Cases

This section discusses various scenarios where Diameter transactions can benefit from the use of priority information.

5.1. First Responder Related Signaling

Natural disasters can result in a considerable increase in usage of network resources. This can be made worse if the disaster results in a loss of network capacity.

The combination of added load and reduced capacity can lead to Diameter nodes becoming overloaded and, as a result, the use of DOIC mechanisms to request a reduction in traffic. This in turn results in requests being throttled in an attempt to control the overload scenario and prevent the overloaded node from failing.

There is the need for first responders and other individuals responsible for handling the after effects of the disaster to be assured that they can gain access to the network resources in order to communicate both between themselves and with other network resources.

Signaling associated with first responders needs to be given a higher priority to help ensure they can most effectively do their jobs.

The United States Wireless Priority Services (WPS) and Government Emergency Telecommunications Service (GETS) are examples of systems designed to address the command and control aspects of these first responder needs.

5.2. Emergency Call Related Signaling

Similar to the first responder scenario, there is also signaling associated with emergency calls. Given the critical nature of these emergency calls, this signaling should also be given preferential treatment when possible.

5.3. Differentiated Services

Operators may desire to differentiate network-based services by providing a service level agreement that includes preferential Diameter routing behavior. This might, for example, be modeled as Platinum, Gold and Silver levels of service.

In this scenario an operator might offer a Platinum SLA the includes ensuring that all signaling for a customer who purchases the Platinum service being marked as having a higher priority than signaling associated with Gold and Silver customers.

5.4. Application Specific Priorities

There are scenarios within Diameter applications where it might be appropriate to give a subset of the transactions for the application a higher priority than other transactions for that application.

For instance, when there is a series of transactions required for a user to gain access to network services, it might be appropriate to mark transactions that occur later in the series at a higher priority than those that occur early in the series. This would recognize that there was potentially significant work done by the network already that would be lost if those later transactions were throttled.

There are also scenarios where an agent cannot easily differentiate a request that starts a session from requests that update or end sessions. In these scenarios it might be appropriate to mark the requests that establish new sessions with a lower priority than updates and session ending requests. This also recognizes that more work has already taken place for established sessions and, as a result, it might be more harmful if the session update and session ending requests were to be throttled.

There are also scenarios where the priority of requests for individual command codes within an application depends on the context that exists when the request is sent. There isn’t always information in the message from which this context can be determined by Diameter nodes other than the node that originates the request.

This is similar to the scenario where a series of requests are needed to access a network service. It is different in that the series of requests involve different application command codes. In this scenario it is requests with the same command code that have different implied priorities.

One example of this is in the 3GPP application [S6a] where a ULR request resulting from an MME (Mobility Management Entity)

restoration procedure might be given a higher priority than a ULR (Update Location Request) resulting from an initial attach.

6. Theory of Operation

This section outlines the envisioned usage of DRMP.

The expected behavior depends on the role (request sender, agent or request handler) of the Diameter node handling the request.

The following behavior is expected during the flow of a Diameter transaction.

1. Request sender - The sender of a request, be it a Diameter Client or a Diameter Server, determines the relative priority of the request and includes that priority information in the request. The method for determining the relative priority is application specific and is outside the scope of this specification. The request sender also saves the priority information with the transaction state. This will be used when handling the answer messages.

2. Agents handling the request - Agents use the priority information when making routing decisions. This can include determining which requests to route first, which requests to throttle and where the request is routed. For instance, requests with higher priority might have a lower probability of being throttled. The mechanism for how the agent determines which requests are throttled is implementation dependent and is outside the scope of this document. Before forwarding request messages, agents generally do not modify the priority information present in the received request message nor include the priority information when absent in the received request message. However, in some scenarios, agents can modify the priority information, for example, edge agents modifying the priority values set by an adjacent operator. There might be other scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert the priority information in the request messages for that non supporting endpoint. When forwarding the request messages, the agent also saves the transaction priority in the transaction state, either as locally managed state or using the Proxy-Info mechanism defined in [RFC6733]. This will be used when handling the associated answer message for the transaction.

3. Request handler - The handler of the request, be it a Diameter Server or a Diameter Client, can use the priority information to determine how to handle the request. This could include

determining the order in which requests are handled and resources that are applied to handling of the request.

4. Answer sender - The handler of the request is also the sender of the answer. The answer sender uses the priority information received in the request message when sending the answer. This implies that answers for higher priority transactions are given preferential treatment to lower priority transactions. The answer sender also has the option of including priority information in the answer message. This is done when the answer message needs to have a different priority than the priority carried in the request message. The priority included by the answer sender is application specific.

5. Agent handling the answer - By default, agents handling answer messages use the priority information stored with the transaction state to determine the priority of relaying the answer message. However, priority information included in the answer message, when present, is used in place of the stored priority information. The use of priority information implies that answers for higher priority transactions are given preferential treatment to lower priority transactions. When forwarding the answer messages, agents generally do not modify the priority information present in the received answer messages nor include the priority information when absent in the received answer messages. However, in some scenarios, agents can modify the priority information, for example, edge agents modifying the priority values set by an adjacent operator. There might be other scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert the priority information for that non supporting endpoint.

6. Answer handler - The answer handler uses the same method as the agent to determine the priority of the answer message. By default the handler of the answer message uses the priority saved in the transaction’s state. Priority information in the answer message is used when present. The priority is used when allocating resources for processing that occurs after the receipt of the answer message.

7. Extensibility

This document does not define extensibility mechanisms that are specific to the DRMP mechanism. As a result, any extension that requires new AVPs will be required to use existing Diameter extensibility mechanisms defined in [RFC6733].

8. Normative Behavior

This section contains the normative behavior associated with Diameter Resource Message Priority (DRMP).

When routing priority information is available, Diameter nodes SHOULD include Diameter routing message priority in the DRMP AVP in all Diameter request messages.

> Note: The method of determining the priority value included in the request is application specific and is not in the scope of this specification.

The priority marking scheme does not require the Diameter Agents to understand application specific AVPs.

When available, Diameter nodes SHOULD use routing priority information included in the DRMP AVP when making Diameter overload throttling decisions.

Diameter agents MAY use routing priority information included in the DRMP AVP when relaying request and answer messages. This includes the selection of routes and the ordering of messages relayed.

> The priority information included in the DRMP AVP in request messages applies to both the request message and, by default, answer message associated with the transaction.

While done only in exceptional circumstances, Diameter agents MAY modify priority information when relaying request and answer messages.

> There might be scenarios where a Diameter agent does modify priority information. For instance, an edge agent might need to modify the priority values set by an adjacent operator.

While done only in exceptional circumstances, Diameter agents MAY add priority information when relaying request and answer messages.

> There might be scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert priority information for that non supporting endpoint.

Diameter endpoints MAY use routing priority information included in the DRMP AVP when making resource allocation decisions for the transaction associated with the request message that contains the DRMP information.

Diameter endpoints MAY use routing priority information included in the DRMP AVP when making resource allocation decisions for the transaction associated with the answer messages using the DRMP information associated with the transaction.

Diameter endpoints MAY include the DRMP AVP in answer messages. This is done when the priority for the answer message needs to have a different priority than the priority carried in the request message.

When determining the priority to apply to answer messages, Diameter nodes MUST use the priority indicated in the DRMP AVP carried in the answer message, if it exists. Otherwise, the Diameter node MUST use the priority indicated in the DRMP AVP of the associated request message.

> Note: One method to determine what priority to apply to an answer when there is no DRMP AVP in the answer message is to save the priority included in the request message in state associated with the Diameter transaction. Another is to use the Proxy-Info mechanism defined in [RFC6733].

Diameter nodes MUST have a default priority to apply to transactions that do not have an explicit priority set in the DRMP AVP.

Diameter nodes SHOULD use the PRIORITY_10 priority as this default value.

Diameter nodes MUST support the ability for the default priority to be modified through local configuration interfaces.

> Note: There are scenarios where operators might want to specify a different default value for transactions that do not have an explicit priority. In this case, the operator defined local policy would override the use of PRIORITY_10 as the default priority.

When using DRMP priority information, Diameter nodes MUST use the default priority for transactions that do not have priority specified in a DRMP AVP.

> Note: This guidance on the handling of messages without a priority does not result in a Diameter agent inserting a DRMP AVP into the message. Rather, it gives guidance on how that specific transaction should be treated when its priority is compared with other requests. When a Diameter agent relays the request it will not insert a DRMP AVP with a priority value of 10.

When setting and using priorities, for all integers x,y in [0,15] treat PRIORITY_<x> as lower priority than PRIOIRTY_<y> when y<x.

Note: As a result PRIORITY_0 is the highest priority.

9. Attribute Value Pairs

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

9.1. DRMP AVP

The DRMP (AVP code TBD1) is of type Enumerated. The value of the AVP indicates the routing message priority for the transaction. The following values are initially defined:

PRIORITY_15 15 PRIORITY_15 is the lowest priority.

PRIORITY_14 14 PRIORITY_14 is a higher priority than PRIORITY_15 and a lower priority than PRIORITY_13.

PRIORITY_13 13 PRIORITY_13 is a higher priority than PRIORITY_14 and a lower priority than PRIORITY_12.

PRIORITY_12 12 PRIORITY_12 is a higher priority than PRIORITY_13 and a lower priority than PRIORITY_11.

PRIORITY_11 11 PRIORITY_11 is a higher priority than PRIORITY_12 and a lower priority than PRIORITY_10.

PRIORITY_10 10 PRIORITY_10 is a higher priority than PRIORITY_11 and a lower priority than PRIORITY_9.

PRIORITY_9 9 PRIORITY_9 is a higher priority than PRIORITY_10 and a lower priority than PRIORITY_8.

PRIORITY_8 8 PRIORITY_8 is a higher priority than PRIORITY_9 and a lower priority than PRIORITY_7.

PRIORITY_7 7 PRIORITY_7 is a higher priority than PRIORITY_8 and a lower priority than PRIORITY_6.

PRIORITY_6 6 PRIORITY_6 is a higher priority than PRIORITY_7 and a lower priority than PRIORITY_5.

PRIORITY_5 5 PRIORITY_5 is a higher priority than PRIORITY_6 and a lower priority than PRIORITY_4.

PRIORITY_4 4 PRIORITY_4 is a higher priority than PRIORITY_5 and a lower priority than PRIORITY_3.

PRIORITY_3 3 PRIORITY_3 is a higher priority than PRIORITY_4 and a lower priority than PRIORITY_2.

PRIORITY_2 2 PRIORITY_2 is a higher priority than PRIORITY_3 and a lower priority than PRIORITY_1.

PRIORITY_1 1 PRIORITY_1 is a higher priority than PRIORITY_2 and a lower priority than PRIORITY_0.

PRIORITY_0 0 Priority 0 is the highest priority.

9.2. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
DRMP	TBD1	x.x	Enumerated		V

10. IANA Considerations

10.1. AVP codes

New AVPs defined by this specification are listed in Section 9. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

10.2. New registries

There are no new IANA registries introduced by this document.

11. Security Considerations

DRMP gives Diameter nodes the ability to influence which requests are are throttled during overload scenarios. Improper use of the DRMP mechanism could result in the malicious Diameter node gaining preferential treatment, by reducing the probability of its requests being throttled, over other Diameter nodes. This would be achieved by the malicious node inserting artificially high priority values.

Diameter does not include features to provide end-to-end authentication, integrity protection, or confidentiality. This opens the possibility that malicious or compromised agents in the path of a request could modify the DRMP AVP to reflect a priority different than that asserted by the sender of the request.

11.1. 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 the DRMP AVP.

There are several ways an attacker might attempt to exploit the DRMP mechanism. A malicious or compromised Diameter node might insert invalid priority values resulting in either preferential treatment, resulting from higher values, or degraded treatment resulting from lower values, for that node.

A similar attack involves a malicious or compromised Diameter agent changing the priority value resulting in the sending Diameter node getting either preferential or degraded service.

The DRMP mechanism can be used to aid in overload throttling decisions. When this is the case then the above attacks are limited in scope to when one or more Diameter nodes are in an overloaded state.

The DRMP mechanism can also be used to influence the order in which Diameter messages are handled by Diameter nodes. The above attacks have a potentially greater impact in this scenario as the priority indication impacts the handling of all requests at all times, independent of the overload status of Diameter nodes in the Diameter network.

11.2. Denial of Service Attacks

The DRMP mechanism does not open direct denial of service attack vectors. Rather, it introduces a mechanism where a node can gain unwarranted preferential treatment. It also introduces a mechanism where a node can get degrated service in the scenario where a rogue agent changes the priority value included in messages.

11.3. End-to End-Security Issues

The lack of end-to-end integrity features makes it difficult to establish trust in DRMP AVPs received from non-adjacent nodes. Any agents in the message path may insert or modify DRMP AVPs. 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 DRMP MUST give operators the ability to select which peers are trusted to deliver DRMP AVPs, and whether they are trusted to forward the DRMP AVPs from non-adjacent nodes. Diameter nodes MUST strip DRMP AVPs from messages received from peers that are not trusted for DRMP purposes.

It is expected that work on end-to-end Diameter security might make it easier to establish trust in non-adjacent nodes for DRMP purposes. Readers should be reminded, however, that the DRMP mechanism allows Diameter agents to modify AVPs in 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 DRMP will require careful consideration, and are beyond the scope of this document.

12. Contributors

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

- Janet P. Gunn

13. References

13.1. 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>.

[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>.

13.2. Informative References

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

[S6a] 3GPP, "Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol", 3GPP TS 29.272 10.8.0, June 2013.

Author’s Address

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

Email: srdonovan@usdonovans.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: October 7, 2016

S. Donovan
Oracle

April 5, 2016

Diameter Routing Message Priority
draft-ietf-dime-drmp-05.txt

Abstract

When making routing and resource allocation decisions, Diameter nodes currently have no generic mechanism to determine the relative priority of Diameter messages. This document addresses this by defining a mechanism to allow Diameter endpoints to indicate the relative priority of Diameter transactions. With this information Diameter nodes can factor that priority into routing, resource allocation and overload abatement decisions.

Status of This Memo

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

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

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

This Internet-Draft will expire on October 7, 2016.

Copyright Notice

Table of Contents

1. Introduction

The Diameter Overload Indication Conveyance (DOIC) solution [RFC7683] for Diameter overload control introduces scenarios where Diameter routing decisions made by Diameter nodes can be influenced by the overload state of other Diameter nodes. This includes the scenarios where Diameter endpoints and Diameter agents can throttle requests as a result of the target for the request being overloaded.

With currently available mechanisms these Diameter nodes do not have a mechanism to differentiate request message priorities when making these throttling decisions. As such, all requests are treated the same, meaning that all requests have the same probability of being throttled.

There are scenarios where treating all requests the same can cause issues. For instance, it might be considered important to reduce the probability of transactions involving first responders being throttled during overload scenarios caused, for example, by a period of heavy signaling resulting from a natural disaster.

This document defines a mechanism that allows Diameter nodes to indicate the relative priority of Diameter transactions. With this information other Diameter nodes can factor the relative priority of requests into routing and throttling decisions.

2. Terminology and Abbreviations

Diversion

As defined in [RFC7683]. An overload abatement treatment where the reacting node selects alternate destinations or paths for requests.

DOIC

Diameter Overload Indication Conveyance.

DRMP

Diameter Routing Message Priority.

Overload Abatement

As defined in [RFC7683]. Reaction to receipt of an overload report resulting in a reduction in traffic sent to the reporting node. Abatement actions include diversion and throttling.

Priority

The relative importance of a Diameter message. A lower priority value implies a higher relative importance of the message.

Throttling

As defined in [RFC7683]. An abatement treatment that limits the number of requests sent by the DOIC 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. Problem Statement

With the introduction of overload control mechanisms, Diameter nodes will be required to make decisions regarding which Diameter request messages should be throttled as a result of overloaded Diameter nodes.

There is currently no generic mechanism to indicate which request messages should be given preferential treatment when these throttling decisions are made.

As a result, all messages are treated equally and, as such, have an equal probability of being throttled.

There are a number of scenarios where it is appropriate for an application to mark a request as being of a higher priority than other application requests. These are discussed in the next section.

This document defines a mechanism for applications to indicate priority for individual transactions, reducing the probability of those transactions being throttled if there are other lower priority transactions that are eligible for throttling treatment.

While the primary usage of DRMP defined priorities is for input to Diameter overload control related throttling decisions, it is also expected that the priority information could also be used for other routing related functionality. This might include giving higher priority transactions preferential treatment when selecting routes.

It is also envisioned that DRMP priority information could be used by Diameter endpoints to make resource allocation decisions. For instance, a Diameter Server might choose to use the priority information to treat higher priority requests ahead of lower priority requests.

> Note: There are a number of application specific definitions indicating various views of application level priority for different requests. Using these application specific priority Attribute Value Pairs (AVPs) as input to throttling and other

Diameter routing decisions would require Diameter agents to understand all applications and do application specific parsing of all messages in order to determine the priority of individual messages. This is considered an unacceptable level of complexity to put on elements whose primary responsibility is to route Diameter messages.

5. Use Cases

This section discusses various scenarios where Diameter transactions can benefit from the use of priority information.

5.1. First Responder Related Signaling

Natural disasters can result in a considerable increase in usage of network resources. This can be made worse if the disaster results in a loss of network capacity.

The combination of added load and reduced capacity can lead to Diameter nodes becoming overloaded and, as a result, the use of DOIC mechanisms to request a reduction in traffic. This in turn results in requests being throttled in an attempt to control the overload scenario and prevent the overloaded node from failing.

There is the need for first responders and other individuals responsible for handling the after effects of the disaster to be assured that they can gain access to the network resources in order to communicate both between themselves and with other network resources.

Signaling associated with first responders needs to be given a higher priority to help ensure they can most effectively do their jobs.

The United States Wireless Priority Services (WPS) and Government Emergency Telecommunications Service (GETS) are examples of systems designed to address the command and control aspects of these first responder needs.

5.2. Emergency Call Related Signaling

Similar to the first responder scenario, there is also signaling associated with emergency calls. Given the critical nature of these emergency calls, this signaling should also be given preferential treatment when possible.

5.3. Differentiated Services

Operators may desire to differentiate network-based services by providing a service level agreement that includes preferential Diameter routing behavior. This might, for example, be modeled as Platinum, Gold and Silver levels of service.

In this scenario an operator might offer a Platinum SLA that includes ensuring that all signaling for a customer who purchases the Platinum service being marked as having a higher priority than signaling associated with Gold and Silver customers.

5.4. Application Specific Priorities

There are scenarios within Diameter applications where it might be appropriate to give a subset of the transactions for the application a higher priority than other transactions for that application.

For instance, when there is a series of transactions required for a user to gain access to network services, it might be appropriate to mark transactions that occur later in the series at a higher priority than those that occur early in the series. This would recognize that there was potentially significant work done by the network already that would be lost if those later transactions were throttled.

There are also scenarios where an agent cannot easily differentiate a request that starts a session from requests that update or end sessions. In these scenarios it might be appropriate to mark the requests that establish new sessions with a lower priority than updates and session ending requests. This also recognizes that more work has already taken place for established sessions and, as a result, it might be more harmful if the session update and session ending requests were to be throttled.

There are also scenarios where the priority of requests for individual command codes within an application depends on the context that exists when the request is sent. There isn’t always information in the message from which this context can be determined by Diameter nodes other than the node that originates the request.

This is similar to the scenario where a series of requests are needed to access a network service. It is different in that the series of requests involve different application command codes. In this scenario requests with the same command code have different implied priorities.

One example of this is in the 3GPP application [S6a] where an Update Location Request (ULR) request resulting from an MME

(Mobility Management Entity) restoration procedure might be given a higher priority than a ULR (Update Location Request) resulting from an initial attach.

6. Theory of Operation

This section outlines the envisioned usage of DRMP.

The expected behavior depends on the role (request sender, agent or request handler) of the Diameter node handling the request.

The following behavior is expected during the flow of a Diameter transaction.

1. Request sender - The sender of a request, be it a Diameter Client or a Diameter Server, determines the relative priority of the request and includes that priority information in the request. The method for determining the relative priority is application specific and is outside the scope of this specification. The request sender also saves the priority information with the transaction state. This will be used when handling the answer messages.

2. Agents handling the request - Agents use the priority information when making routing decisions. This can include determining which requests to route first, which requests to throttle and where the request is routed. For instance, requests with higher priority might have a lower probability of being throttled. The mechanism for how the agent determines which requests are throttled is implementation dependent and is outside the scope of this document. Before forwarding request messages, agents generally do not modify the priority information present in the received request message nor include the priority information when absent in the received request message. However, in some scenarios, agents can modify the priority information, for example, edge agents modifying the priority values set by an adjacent operator. There might be other scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert the priority information in the request messages for that non supporting endpoint. When forwarding the request messages, the agent also saves the transaction priority in the transaction state, either as locally managed state or using the Proxy-Info mechanism defined in [RFC6733]. This will be used when handling the associated answer message for the transaction.

3. Request handler - The handler of the request, be it a Diameter Server or a Diameter Client, can use the priority information to determine how to handle the request. This could include

determining the order in which requests are handled and resources that are applied to handling of the request.

4. Answer sender - The handler of the request is also the sender of the answer. The answer sender uses the priority information received in the request message when sending the answer. This implies that answers for higher priority transactions are given preferential treatment to lower priority transactions. The answer sender also has the option of including priority information in the answer message. This is done when the answer message needs to have a different priority than the priority carried in the request message. The priority included by the answer sender is application specific.

5. Agent handling the answer - By default, agents handling answer messages use the priority information stored with the transaction state to determine the priority of relaying the answer message. However, priority information included in the answer message, when present, is used in place of the stored priority information. The use of priority information implies that answers for higher priority transactions are given preferential treatment to lower priority transactions. When forwarding the answer messages, agents generally do not modify the priority information present in the received answer messages nor include the priority information when absent in the received answer messages. However, in some scenarios, agents can modify the priority information, for example, edge agents modifying the priority values set by an adjacent operator. There might be other scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert the priority information for that non-supporting endpoint.

6. Answer handler - The answer handler uses the same method as the agent to determine the priority of the answer message. By default the handler of the answer message uses the priority saved in the transaction’s state. Priority information in the answer message is used when present. The priority is used when allocating resources for processing that occurs after the receipt of the answer message.

7. Extensibility

This document does not define extensibility mechanisms that are specific to the DRMP mechanism. As a result, any extension that requires new AVPs will be required to use existing Diameter extensibility mechanisms defined in [RFC6733].

8. Normative Behavior

This section contains the normative behavior associated with Diameter Resource Message Priority (DRMP).

When routing priority information is available, Diameter nodes SHOULD include Diameter routing message priority in the DRMP AVP in all Diameter request messages.

> Note: The method of determining the priority value included in the request is application specific and is not in the scope of this specification.

The priority marking scheme does not require the Diameter Agents to understand application specific AVPs.

When available, Diameter nodes SHOULD use routing priority information included in the DRMP AVP when making Diameter overload throttling decisions.

Diameter agents MAY use routing priority information included in the DRMP AVP when relaying request and answer messages. This includes the selection of routes and the ordering of messages relayed.

> Note: The priority information included in the DRMP AVP in request messages applies to both the request message and, by default, answer message associated with the transaction.

While done only in exceptional circumstances, Diameter agents MAY modify priority information when relaying request and answer messages.

> Note: There might be scenarios where a Diameter agent does modify priority information. For instance, an edge agent might need to modify the priority values set by an adjacent operator.

While done only in exceptional circumstances, Diameter agents MAY add priority information when relaying request and answer messages.

> Note: There might be scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert priority information for that non-supporting endpoint.

Diameter endpoints MAY use routing priority information included in the DRMP AVP when making resource allocation decisions for the transaction associated with the request message that contains the DRMP information.

Diameter endpoints MAY use routing priority information included in the DRMP AVP when making resource allocation decisions for the transaction associated with the answer messages using the DRMP information associated with the transaction.

Diameter endpoints MAY include the DRMP AVP in answer messages. This is done when the priority for the answer message needs to have a different priority than the priority carried in the request message.

When determining the priority to apply to answer messages, Diameter nodes MUST use the priority indicated in the DRMP AVP carried in the answer message, if it exists. Otherwise, the Diameter node MUST use the priority indicated in the DRMP AVP of the associated request message.

> Note: One method to determine what priority to apply to an answer when there is no DRMP AVP in the answer message is to save the priority included in the request message in state associated with the Diameter transaction. Another is to use the Proxy-Info mechanism defined in [RFC6733].

Diameter nodes MUST have a default priority to apply to transactions that do not have an explicit priority set in the DRMP AVP.

Diameter nodes SHOULD use the PRIORITY_10 priority as this default value.

Diameter nodes MUST support the ability for the default priority to be modified through local configuration interfaces.

> Note: There are scenarios where operators might want to specify a different default value for transactions that do not have an explicit priority. In this case, the operator defined local policy would override the use of PRIORITY_10 as the default priority.

When using DRMP priority information, Diameter nodes MUST use the default priority for transactions that do not have priority specified in a DRMP AVP.

> Note: This guidance on the handling of messages without a priority does not result in a Diameter agent inserting a DRMP AVP into the message. Rather, it gives guidance on how that specific transaction should be treated when its priority is compared with other requests. When a Diameter agent relays the request it will not insert a DRMP AVP with a priority value of 10.

When setting and using priorities, for all integers x,y in [0,15] treat PRIORITY_<x> as lower priority than PRIOIRTY_<y> when y<x.

Note: As a result PRIORITY_0 is the highest priority.

9. Attribute Value Pairs

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

9.1. DRMP AVP

The DRMP (AVP code TBD1) is of type Enumerated. The value of the AVP indicates the routing message priority for the transaction. The following values are defined:

PRIORITY_15 15 PRIORITY_15 is the lowest priority.

PRIORITY_14 14 PRIORITY_14 is a higher priority than PRIORITY_15 and a lower priority than PRIORITY_13.

PRIORITY_13 13 PRIORITY_13 is a higher priority than PRIORITY_14 and a lower priority than PRIORITY_12.

PRIORITY_12 12 PRIORITY_12 is a higher priority than PRIORITY_13 and a lower priority than PRIORITY_11.

PRIORITY_11 11 PRIORITY_11 is a higher priority than PRIORITY_12 and a lower priority than PRIORITY_10.

PRIORITY_10 10 PRIORITY_10 is a higher priority than PRIORITY_11 and a lower priority than PRIORITY_9.

PRIORITY_9 9 PRIORITY_9 is a higher priority than PRIORITY_10 and a lower priority than PRIORITY_8.

PRIORITY_8 8 PRIORITY_8 is a higher priority than PRIORITY_9 and a lower priority than PRIORITY_7.

PRIORITY_7 7 PRIORITY_7 is a higher priority than PRIORITY_8 and a lower priority than PRIORITY_6.

PRIORITY_6 6 PRIORITY_6 is a higher priority than PRIORITY_7 and a lower priority than PRIORITY_5.

PRIORITY_5 5 PRIORITY_5 is a higher priority than PRIORITY_6 and a lower priority than PRIORITY_4.

PRIORITY_4 4 PRIORITY_4 is a higher priority than PRIORITY_5 and a lower priority than PRIORITY_3.

PRIORITY_3 3 PRIORITY_3 is a higher priority than PRIORITY_4 and a lower priority than PRIORITY_2.

PRIORITY_2 2 PRIORITY_2 is a higher priority than PRIORITY_3 and a lower priority than PRIORITY_1.

PRIORITY_1 1 PRIORITY_1 is a higher priority than PRIORITY_2 and a lower priority than PRIORITY_0.

PRIORITY_0 0 Priority 0 is the highest priority.

9.2. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT	AVP flag rules: V
DRMP	TBD1	x.x	Enumerated			V

10. IANA Considerations

10.1. AVP codes

New AVPs defined by this specification are listed in Section 9. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

10.2. New registries

There are no new IANA registries introduced by this document.

11. Security Considerations

DRMP gives Diameter nodes the ability to influence which requests are throttled during overload scenarios. Improper use of the DRMP mechanism could result in the malicious Diameter node gaining preferential treatment, by reducing the probability of its requests being throttled, over other Diameter nodes. This would be achieved by the malicious node inserting artificially high priority values.

Diameter does not include features to provide end-to-end authentication, integrity protection, or confidentiality. This opens the possibility that malicious or compromised agents in the path of a request could modify the DRMP AVP to reflect a priority different than that asserted by the sender of the request.

11.1. 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., Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP)) connection, or the messages may traverse one or more intermediaries, known as Diameter Agents. Diameter nodes use Transport Layer Security (TLS), Datagram Transport Layer Security (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 the DRMP AVP.

There are several ways an attacker might attempt to exploit the DRMP mechanism. A malicious or compromised Diameter node might insert invalid priority values resulting in either preferential treatment, resulting from higher values, or degraded treatment resulting from lower values, for that node.

A similar attack involves a malicious or compromised Diameter agent changing the priority value resulting in the sending Diameter node getting either preferential or degraded service.

The DRMP mechanism can be used to aid in overload throttling decisions. When this is the case then the above attacks are limited in scope to when one or more Diameter nodes are in an overloaded state.

The DRMP mechanism can also be used to influence the order in which Diameter messages are handled by Diameter nodes. The above attacks have a potentially greater impact in this scenario as the priority indication impacts the handling of all requests at all times,

independent of the overload status of Diameter nodes in the Diameter network.

11.2. Denial of Service Attacks

The DRMP mechanism does not open direct denial of service attack vectors. Rather, it introduces a mechanism where a node can gain unwarranted preferential treatment. It also introduces a mechanism where a node can get degraded service in the scenario where a rogue agent changes the priority value included in messages.

11.3. End-to End-Security Issues

The lack of end-to-end integrity features in Diameter [RFC6733] makes it difficult to establish trust in DRMP AVPs received from non-adjacent nodes. Any agents in the message path may insert or modify DRMP AVPs. 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 DRMP MUST give operators the ability to select which peers are trusted to deliver DRMP AVPs, and whether they are trusted to forward the DRMP AVPs from non-adjacent nodes. Diameter nodes MUST strip DRMP AVPs from messages received from peers that are not trusted for DRMP purposes.

It is expected that work on end-to-end Diameter security might make it easier to establish trust in non-adjacent nodes for DRMP purposes. Readers should be reminded, however, that the DRMP mechanism allows Diameter agents to modify AVPs in 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 DRMP will require careful consideration, and are beyond the scope of this document.

12. Contributors

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

- Janet P. Gunn

13. References

13.1. 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>.

[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>.

13.2. Informative References

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

[S6a] 3GPP, "Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol", 3GPP TS 29.272 10.8.0, June 2013.

Author’s Address

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

Email: srdonovan@usdonovans.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2016

S. Donovan
Oracle

June 3, 2016

Diameter Routing Message Priority
draft-ietf-dime-drmp-07.txt

Abstract

When making routing and resource allocation decisions, Diameter nodes currently have no generic mechanism to determine the relative priority of Diameter messages. This document addresses this by defining a mechanism to allow Diameter endpoints to indicate the relative priority of Diameter transactions. With this information Diameter nodes can factor that priority into routing, resource allocation and overload abatement decisions.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 5, 2016.

Copyright Notice

Table of Contents

1. Introduction

The Diameter Overload Indication Conveyance (DOIC) solution [RFC7683] for Diameter overload control introduces scenarios where Diameter routing decisions made by Diameter nodes can be influenced by the overload state of other Diameter nodes. This includes the scenarios where Diameter endpoints and Diameter agents can throttle requests as a result of the target for the request being overloaded.

With currently available mechanisms these Diameter nodes do not have a mechanism to differentiate request message priorities when making these throttling decisions. As such, all requests are treated the

same, meaning that all requests have the same probability of being throttled.

There are scenarios where treating all requests the same can cause issues. For instance, it might be considered important to reduce the probability of transactions involving first responders being throttled during overload scenarios caused, for example, by a period of heavy signaling resulting from a natural disaster.

This document defines a mechanism that allows Diameter nodes to indicate the relative priority of Diameter transactions. With this information other Diameter nodes can factor the relative priority of requests into routing and throttling decisions.

1.1. Applicability

There are two primary considerations that must be addressed for the mechanism described in this document to work effectively. The first takes into consideration that the Diameter base protocol defined in [RFC6733] is designed to transport multiple Diameter applications and that Diameter nodes can be implemented that support multiple applications. In order for the DRMP mechanism to work, the priorities defined for all messages across all applications used in a Diameter administrative domain must be defined in a consistent and coordinated fashion, taking the default priority into account. See Section 10 for a discussion of some of the considerations that need to be factored into the setting of DRMP priorities used by Diameter applications.

> Note that this consideration does not apply to Diameter networks where all Diameter nodes only support a single application.

Without this cross application priority design taken into consideration it is possible for messages for one application to gain unwarranted preferential treatment over messages for other applications.

This mechanism also depends on all of the messages that carry the DRMP AVP are inserted into Diameter messages by trusted nodes within the Diameter administrative domain. As discussed in Section 12, misbehaving nodes have the ability to use the DRMP mechanism to gain unwarranted preferential treatment.

When messages cross Diameter administrative boundaries, care should be taken to either strip or modify the DRMP priority values in these messages. If the priority definitions vary between the two Diameter administrative domains then it is possible for messages from a foreign domain to gain unwarranted preferential treatment.

2. Terminology and Abbreviations

Diversion

As defined in [RFC7683]. An overload abatement treatment where the reacting node selects alternate destinations or paths for requests.

DOIC

Diameter Overload Indication Conveyance.

DRMP

Diameter Routing Message Priority.

Overload Abatement

As defined in [RFC7683]. Reaction to receipt of an overload report resulting in a reduction in traffic sent to the reporting node. Abatement actions include diversion and throttling.

Priority

The relative importance of a Diameter message. A lower priority value implies a higher relative importance of the message.

Throttling

As defined in [RFC7683]. An abatement treatment that limits the number of requests sent by the DOIC 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. Problem Statement

With the introduction of overload control mechanisms, Diameter nodes will be required to make decisions regarding which Diameter request messages should be throttled as a result of overloaded Diameter nodes.

There is currently no generic mechanism to indicate which request messages should be given preferential treatment when these throttling decisions are made.

As a result, all messages are treated equally and, as such, have an equal probability of being throttled.

There are a number of scenarios where it is appropriate for an application to mark a request as being of a higher priority than other application requests. These are discussed in the next section.

This document defines a mechanism for applications to indicate priority for individual transactions, reducing the probability of those transactions being throttled if there are other lower priority transactions that are eligible for throttling treatment.

While the primary usage of DRMP defined priorities is for input to Diameter overload control related throttling decisions, it is also expected that the priority information could also be used for other routing related functionality. This might include giving higher priority transactions preferential treatment when selecting routes.

It is also envisioned that DRMP priority information could be used by Diameter endpoints to make resource allocation decisions. For instance, a Diameter Server might choose to use the priority information to treat higher priority requests ahead of lower priority requests. It might also use the priority information to as a reason to fail a request as a result of insufficient resources.

> Note: There are a number of application specific definitions indicating various views of application level priority for different requests. Using these application specific priority Attribute Value Pairs (AVPs) as input to throttling and other Diameter routing decisions would require Diameter agents to understand all applications and do application specific parsing of all messages in order to determine the priority of individual messages. This is considered an unacceptable level of complexity to put on elements whose primary responsibility is to route Diameter messages.

5. Use Cases

This section discusses various scenarios where Diameter transactions can benefit from the use of priority information.

It is important to note that for priority information to be reliably usable, the Diameter nodes sending and consuming DRMP AVPs must have pre-established trust relationships of the sort described in Section 12.

5.1. First Responder Related Signaling

Natural disasters can result in a considerable increase in usage of network resources. This can be made worse if the disaster results in a loss of network capacity.

The combination of added load and reduced capacity can lead to Diameter nodes becoming overloaded and, as a result, the use of DOIC mechanisms to request a reduction in traffic. This in turn results in requests being throttled in an attempt to control the overload scenario and prevent the overloaded node from failing.

There is the need for first responders and other individuals responsible for handling the after effects of the disaster to be assured that they can gain access to the network resources in order to communicate both between themselves and with other network resources.

Signaling associated with first responders needs to be given a higher priority to help ensure they can most effectively do their jobs.

The United States Wireless Priority Services (WPS) and Government Emergency Telecommunications Service (GETS) are examples of systems designed to address the command and control aspects of these first responder needs.

5.2. Emergency Call Related Signaling

Similar to the first responder scenario, there is also signaling associated with emergency calls. Given the critical nature of these emergency calls, this signaling should also be given preferential treatment when possible.

5.3. Differentiated Services

Operators may desire to differentiate network-based services by providing a service level agreement that includes preferential

Diameter routing behavior. This might, for example, be modeled as Platinum, Gold and Silver levels of service.

In this scenario an operator might offer a Platinum SLA that includes ensuring that all signaling for a customer who purchases the Platinum service being marked as having a higher priority than signaling associated with Gold and Silver customers.

5.4. Application Specific Priorities

There are scenarios within Diameter applications where it might be appropriate to give a subset of the transactions for the application a higher priority than other transactions for that application.

For instance, when there is a series of transactions required for a user to gain access to network services, it might be appropriate to mark transactions that occur later in the series at a higher priority than those that occur early in the series. This would recognize that there was potentially significant work done by the network already that would be lost if those later transactions were throttled.

There are also scenarios where an agent cannot easily differentiate a request that starts a session from requests that update or end sessions. In these scenarios it might be appropriate to mark the requests that establish new sessions with a lower priority than updates and session ending requests. This also recognizes that more work has already taken place for established sessions and, as a result, it might be more harmful from a signaling point of view if the session update and session ending requests were to be throttled.

There are also scenarios where the priority of requests for individual command codes within an application depends on the context that exists when the request is sent. There isn’t always information in the message from which this context can be determined by Diameter nodes other than the node that originates the request.

This is similar to the scenario where a series of requests are needed to access a network service. It is different in that the series of requests involve different application command codes. In this scenario requests with the same command code have different implied priorities.

> One example of this is in the 3GPP application [S6a] where an Update Location Request (ULR) request resulting from an MME (Mobility Management Entity) restoration procedure might be given a higher priority than a ULR (Update Location Request) resulting from an initial attach.

6. Theory of Operation

This section outlines the envisioned usage of DRMP.

The expected behavior depends on the role (request sender, agent or request handler) of the Diameter node handling the request.

The following behavior is expected during the flow of a Diameter transaction.

1. Request sender - The sender of a request, be it a Diameter Client or a Diameter Server, determines the relative priority of the request and includes that priority information in the request. The method for determining the relative priority is application specific and is outside the scope of this specification. The request sender also saves the priority information with the transaction state. This will be used when handling the answer messages.

2. Agents handling the request - Agents use the priority information when making routing decisions. This can include determining which requests to route first, which requests to throttle and where the request is routed. For instance, requests with higher priority might have a lower probability of being throttled. The mechanism for how the agent determines which requests are candidates to be throttled is implementation dependent and is outside the scope of this document. Before forwarding request messages, agents generally do not modify the priority information present in the received request message nor include the priority information when absent in the received request message. However, in some scenarios, agents can modify the priority information, for example, edge agents modifying the priority values set by an adjacent operator. There might be other scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert the priority information in the request messages for that non supporting endpoint. When forwarding the request messages, the agent also saves the transaction priority in the transaction state, either as locally managed state or using the Proxy-Info mechanism defined in [RFC6733]. This will be used when handling the associated answer message for the transaction.

3. Request handler - The handler of the request, be it a Diameter Server or a Diameter Client, can use the priority information to determine how to handle the request. This could include determining the order in which requests are handled and resources that are applied to handling of the request.

4. Answer sender - The handler of the request is also the sender of the answer. The answer sender uses the priority information received in the request message when sending the answer. This implies that answers for higher priority transactions are given preferential treatment to lower priority transactions. The answer sender also has the option of including priority information in the answer message. This is done when the answer message needs to have a different priority than the priority carried in the request message. The priority included by the answer sender is application specific.

5. Agent handling the answer - By default, agents handling answer messages use the priority information stored with the transaction state to determine the priority of relaying the answer message. However, priority information included in the answer message, when present, is used in place of the stored priority information. The use of priority information implies that answers for higher priority transactions are given preferential treatment to lower priority transactions. When forwarding the answer messages, agents generally do not modify the priority information present in the received answer messages nor include the priority information when absent in the received answer messages. However, in some scenarios, agents can modify the priority information, for example, edge agents modifying the priority values set by an adjacent operator. There might be other scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert the priority information for that non-supporting endpoint.

6. Answer handler - The answer handler uses the same method as the agent to determine the priority of the answer message. By default the handler of the answer message uses the priority saved in the transaction’s state. Priority information in the answer message is used when present. The priority is used when allocating resources for processing that occurs after the receipt of the answer message.

7. Extensibility

This document does not define extensibility mechanisms that are specific to the DRMP mechanism. As a result, any extension that requires new AVPs will be required to use existing Diameter extensibility mechanisms defined in [RFC6733].

8. Normative Behavior

This section contains the normative behavior associated with Diameter Resource Message Priority (DRMP).

When routing priority information is available, Diameter nodes SHOULD include Diameter routing message priority in the DRMP AVP in all Diameter request messages.

> Note: The method of determining the priority value included in the request is application specific and is not in the scope of this specification.

The priority marking scheme does not require the Diameter Agents to understand application specific AVPs.

When available, Diameter nodes SHOULD use routing priority information included in the DRMP AVP when making Diameter overload throttling decisions.

Diameter agents MAY use routing priority information included in the DRMP AVP when relaying request and answer messages. This includes the selection of routes and the ordering of messages relayed.

> Note: The priority information included in the DRMP AVP in request messages applies to both the request message and, by default, answer message associated with the transaction.

While done only in exceptional circumstances, Diameter agents MAY modify priority information when relaying request and answer messages.

> Note: There might be scenarios where a Diameter agent does modify priority information. For instance, an edge agent might need to modify the priority values set by an adjacent operator.

While done only in exceptional circumstances, Diameter agents MAY add priority information when relaying request and answer messages.

> Note: There might be scenarios where a Diameter endpoint does not support the DRMP mechanism and agents insert priority information for that non-supporting endpoint.

Diameter endpoints MAY use routing priority information included in the DRMP AVP when making resource allocation decisions for the transaction associated with the request message that contains the DRMP information.

Diameter endpoints MAY use routing priority information included in the DRMP AVP when making resource allocation decisions for the transaction associated with the answer messages using the DRMP information associated with the transaction.

Diameter endpoints MAY include the DRMP AVP in answer messages. This is done when the priority for the answer message needs to have a different priority than the priority carried in the request message.

When determining the priority to apply to answer messages, Diameter nodes SHOULD use the priority indicated in the DRMP AVP carried in the answer message, if it exists. If there is not DRMP AVP in the answer message then the Diameter node SHOULD use the priority indicated in the DRMP AVP of the associated request message.

> Note: One method to determine what priority to apply to an answer when there is no DRMP AVP in the answer message is to save the priority included in the request message in state associated with the Diameter transaction. Another is to use the Proxy-Info mechanism defined in [RFC6733].

Diameter nodes MUST have a default priority to apply to transactions that do not have an explicit priority set in the DRMP AVP.

In order to guaranty consistent handling of messages from nonupgraded Diameter clients, Diameter nodes SHOULD use the PRIORITY_10 priority as this default priority value.

> PRIORITY_10 is a mid range priority that corresponds to "normal" traffic and thus would be a suitable default for most deployments, while still allowing different Diameter applications to designate other priorities for lower and higher priority traffic.
>
> Note: This does not imply that a DRMP AVP is added to the message. Rather, the message is treated the same as a message that has a DRMP AVP with priority value of PRIORITY_10.

Diameter nodes MUST support the ability for the default priority to be modified through local configuration interfaces.

> Note: There are scenarios where operators might want to specify a different default value for transactions that do not have an explicit priority. In this case, the operator defined local policy would override the use of PRIORITY_10 as the default priority.

When using DRMP priority information, Diameter nodes MUST use the default priority for transactions that do not have priority specified in a DRMP AVP.

Note: This guidance on the handling of messages without a priority does not result in a Diameter agent inserting a DRMP AVP into the message. Rather, it gives guidance on how that specific transaction should be treated when its priority is compared with other requests. When a Diameter agent relays the request it will not insert a DRMP AVP with a priority value of 10.

When setting and using priorities, for all integers x,y in [0,15] treat PRIORITY_<x> as lower priority than PRIOIRTY_<y> when y<x.

Note: As a result PRIORITY_0 is the highest priority.

9. Attribute Value Pairs

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

9.1. DRMP AVP

The DRMP (AVP code TBD1) is of type Enumerated. The value of the AVP indicates the routing message priority for the transaction. The following values are defined:

PRIORITY_15 15 PRIORITY_15 is the lowest priority.

PRIORITY_14 14 PRIORITY_14 is a higher priority than PRIORITY_15 and a lower priority than PRIORITY_13.

PRIORITY_13 13 PRIORITY_13 is a higher priority than PRIORITY_14 and a lower priority than PRIORITY_12.

PRIORITY_12 12 PRIORITY_12 is a higher priority than PRIORITY_13 and a lower priority than PRIORITY_11.

PRIORITY_11 11 PRIORITY_11 is a higher priority than PRIORITY_12 and a lower priority than PRIORITY_10.

PRIORITY_10 10 PRIORITY_10 is a higher priority than PRIORITY_11 and a lower priority than PRIORITY_9.

PRIORITY_9 9 PRIORITY_9 is a higher priority than PRIORITY_10 and a lower priority than PRIORITY_8.

PRIORITY_8 8 PRIORITY_8 is a higher priority than PRIORITY_9 and a lower priority than PRIORITY_7.

PRIORITY_7 7 PRIORITY_7 is a higher priority than PRIORITY_8 and a lower priority than PRIORITY_6.

PRIORITY_6 6 PRIORITY_6 is a higher priority than PRIORITY_7 and a lower priority than PRIORITY_5.

PRIORITY_5 5 PRIORITY_5 is a higher priority than PRIORITY_6 and a lower priority than PRIORITY_4.

PRIORITY_4 4 PRIORITY_4 is a higher priority than PRIORITY_5 and a lower priority than PRIORITY_3.

PRIORITY_3 3 PRIORITY_3 is a higher priority than PRIORITY_4 and a lower priority than PRIORITY_2.

PRIORITY_2 2 PRIORITY_2 is a higher priority than PRIORITY_3 and a lower priority than PRIORITY_1.

PRIORITY_1 1 PRIORITY_1 is a higher priority than PRIORITY_2 and a lower priority than PRIORITY_0.

PRIORITY_0 0 Priority 0 is the highest priority.

9.2. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules		
					MUST	MUST NOT
DRMP	TBD1	x.x	Enumerated			V

10. Considerations When Defining Application Priorities

As discussed in Section 1.1, it is important that the definition of priority values used by all applications within a single Diameter administrative domain be done in a consistent and coordinated manner.

The following are some things to be considered when defining the DRMP priorities to be used in Diameter networks which support Diameter nodes handling multiple applications.

1. As with any prioritization scheme, it is possible for higher priority messages to block lower priority messages from ever being handled. In a Diameter network this will often result in those Diameter transactions being retried. This can result in more traffic than the network would have handled without use of the DRMP mechanism.

   One potential guideline to prevent unwanted starving of lower priority messages is to have higher priority messages represent a relatively small portion of messages handled by the Diameter network under normal scenarios.

      Note that there are scenarios, such as first responder messages, where the blocking of lower priority messages is a requirement.

2. When setting priorities for any of the use cases outlined in Section 5 it is important to use the same priority values across applications. For instance, when defining priority for the First Responder use case discussed in Section 5.1 and the Emergency call use case discussed in Section 5.2 use cases, one high priority value might be used for all first responder messages, say PRIORITY_2, and a slightly lower priority value, say PRIORITY_3, might be used for emergency call related messages. These values should be specified for these use cases across all applications used within the Diameter administrative domain.

      Note that the values mentioned here are strictly for illustrative purposes. The actual values used for these use cases are likely to be different.

3. Messages without the DRMP AVP will be given default priority value threatment. This will include messages from Diameter Clients that have not been updated to support the DRMP mechanism. It might also include messages from foreign administrative domains if the DRMP AVPs are stripped from messages crossing the Diameter administrative domains.

4. The process used to introduce the DRMP mechanism into a Diameter network should also be taken into consideration. Messages of the same type within the same application might get different treatment depending on whether those messages are sent from nodes that are upgraded to support the DRMP mechanism versus nodes that have not yet been upgraded to support the DRMP mechanism.

11. IANA Considerations

11.1. AVP codes

The new AVP defined by this specification is listed in Section 9. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

11.2. New registries

There are no new IANA registries introduced by this document.

12. Security Considerations

DRMP gives Diameter nodes the ability to influence which requests are throttled during overload scenarios. In addition, DRMP can be used in determining the routing decisions for request messages. Improper use of the DRMP mechanism could result in the malicious Diameter node gaining preferential treatment, by reducing the probability of its requests being throttled, over other Diameter nodes. This would be achieved by the malicious node inserting artificially high priority values.

Diameter does not include features to provide end-to-end authentication, integrity protection, or confidentiality. This opens the possibility that malicious or compromised agents in the path of a request could modify the DRMP AVP to reflect a priority different than that asserted by the sender of the request.

12.1. 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., Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP)) connection, or the messages may traverse one or more intermediaries, known as Diameter Agents. Diameter nodes use Transport Layer Security (TLS), Datagram Transport Layer Security (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 the DRMP AVP.

There are several ways an attacker might attempt to exploit the DRMP mechanism. A malicious or compromised Diameter node might insert invalid priority values resulting in either preferential treatment, resulting from higher values, or degraded treatment resulting from lower values, for that node.

A similar attack involves a malicious or compromised Diameter agent changing the priority value resulting in the sending Diameter node getting either preferential or degraded service.

The DRMP mechanism can be used to aid in overload throttling decisions. When this is the case then the above attacks are limited in scope to when one or more Diameter nodes are in an overloaded state.

The DRMP mechanism can also be used to influence the order in which Diameter messages are handled by Diameter nodes. The above attacks have a potentially greater impact in this scenario as the priority indication impacts the handling of all requests at all times, independent of the overload status of Diameter nodes in the Diameter network.

12.2. Denial of Service Attacks

The DRMP mechanism does not open direct denial of service attack vectors. Rather, it introduces a mechanism where a node can gain unwarranted preferential treatment. It also introduces a mechanism where a node can get degraded service in the scenario where a rogue agent changes the priority value included in messages.

12.3. End-to End-Security Issues

The lack of end-to-end integrity features in Diameter [RFC6733] makes it difficult to establish trust in DRMP AVPs received from non-adjacent nodes. Any agents in the message path may insert or modify DRMP AVPs. 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 DRMP MUST give operators the ability to select which peers are trusted to deliver DRMP AVPs, and whether they are trusted to forward the DRMP AVPs from non-adjacent nodes. Diameter nodes MUST strip DRMP AVPs from messages received from peers that are not trusted for DRMP purposes.

It is expected that work on end-to-end Diameter security might make it easier to establish trust in non-adjacent nodes for DRMP purposes. Readers should be reminded, however, that the DRMP mechanism allows Diameter agents to modify AVPs in 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 DRMP will require careful consideration, and are beyond the scope of this document.

13. Contributors

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

- Janet P. Gunn

14. References

14.1. 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>.

[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>.

14.2. Informative References

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

[S6a] 3GPP, "Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol", 3GPP TS 29.272 10.8.0, June 2013.

Author’s Address

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

Email: srdonovan@usdonovans.com

DIME
Internet-Draft
Intended status: Informational
Expires: July 16, 2016

H. Tschofenig
ARM Limited
J. Korhonen, Ed.
Broadcom Corporation
G. Zorn
Network Zen
K. Pillay
Oracle Communications

January 13, 2016

Diameter AVP Level Security End-to-End Security: Scenarios and Requirements
draft-ietf-dime-e2e-sec-req-04.txt

Abstract

This specification discusses requirements for providing Diameter security at the level of individual Attribute-Value Pairs.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 16, 2016.

Copyright Notice

Table of Contents

1. Introduction

The Diameter base protocol specification [2] offers security protection between neighboring Diameter peers and mandates that peer connections must be protected by TLS (for TCP), DTLS (for SCTP) or alternative security mechanisms independent of Diameter (e.g., IPsec) is used. These security protocols offer a wide range of security properties, including entity authentication, data-origin authentication, integrity, confidentiality protection and replay protection. They also support a large number of cryptographic algorithms, algorithm negotiation, and different types of credentials. It should be understood that TLS/DTLS/IPsec in Diameter context does not provide end-to-end security unless the Diameter nodes are direct peers i.e., neighboring Diameter nodes. The current Diameter security is realized hop-by-hop.

The need to also offer additional security protection of AVPs between non-neighboring Diameter nodes was recognized very early in the work on Diameter. This led to work on Diameter security using the Cryptographic Message Syntax (CMS) [3]. Due to lack of deployment interest at that time (and the complexity of the developed solution) the specification was, however, never completed.

In the meanwhile Diameter had received a lot of deployment interest from the cellular operator community and because of the sophistication of those deployments the need for protecting Diameter AVPs between non-neighboring nodes re-surfaced. Since early 2000 (when the work on [3] was discontinued) the Internet community had seen advances in cryptographic algorithms (for example, authenticated

encryption algorithms) and new security building blocks were developed.

This document collects requirements for developing a solution to protect Diameter AVPs.

2. Terminology

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in this documents are to be interpreted as described in RFC 2119 [1].

This document re-uses terminology from the Diameter base specification [2].

In the figures below we use the symbols 'AVP' and '{AVP}k'. AVP refers to an unprotected AVP and {AVP}k refers to an AVP that experiences security protection (using key "k") without further distinguishing between integrity and confidentiality protection.

3. Security Threats

The following description aims to illustrate various security threats that raise the need for protecting Diameter Attribute-Value Pairs (AVPs). Figure 1 illustrates an example of Diameter based roaming architecture in which Diameter clients within the visited networks need to interact with Diameter servers in the home domain. AAA domains are interconnected using a Diameter-based AAA interconnection network labeled as AAA Broker.

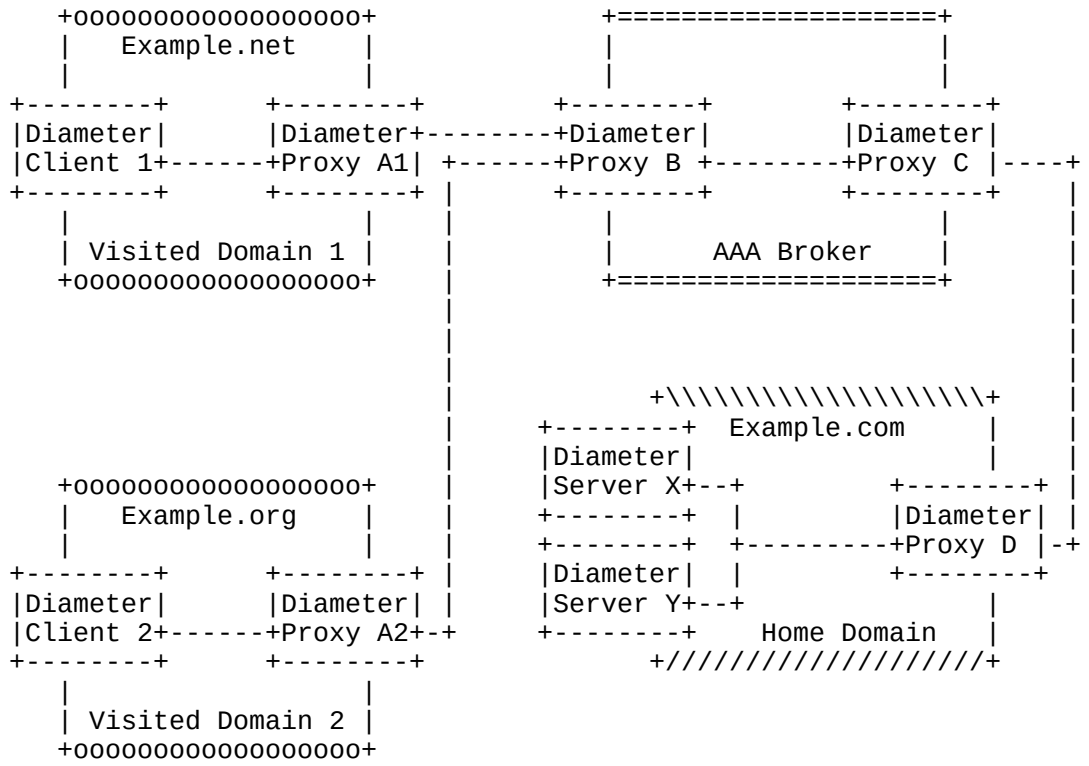

Figure 1: Example Diameter Deployment.

Eavesdropping: Some Diameter applications carry information that is only intended for consumption by end points, either by the Diameter client or by the Diameter server but not by intermediaries. As an example, consider the Diameter EAP application [4] that allows the transport of keying material between the Diameter server to the Diameter client (using the EAP-Master-Session-Key AVP) for the protection of air interface between the end device and the network access server. The content of the EAP-Master-Session-Key AVP should benefit from protection against eavesdropping by intermediaries. Other AVPs, for example those listed in Section 13.3 of [2], might also carry sensitive personal data that, when collected by intermediaries, allow for traffic analysis.

In context of the deployment shown in Figure 1 the adversary could, for example, be in the AAA broker network.

Injection and Manipulation: The Diameter base protocol specification mandates security protection between neighboring nodes but Diameter agents may be compromised or misconfigured and inject/ manipulate AVPs. To detect such actions additional security protection needs to be applied at the Diameter layer.

Nodes that could launch such an attack are any Diameter agents along the end-to-end communication path.

Impersonation: Imagine a case where a Diameter message from Example.net contains information claiming to be from Example.org. This would either require strict verification at the edge of the AAA broker network or cryptographic assurance at the Diameter layer to prevent a successful impersonation attack.

Any Diameter realm could launch such an attack aiming for financial benefits or to disrupt service availability.

4. Scenarios for Diameter AVP-Level Protection

This scenario outlines a number of cases for deploying security protection of individual Diameter AVPs.

In the first scenario, shown in Figure 2, end-to-end security protection is provided between the Diameter client and the Diameter server with any number of intermediate Diameter agents. Diameter AVPs exchanged between these two Diameter nodes may be protected end-to-end (notation '{AVP}k') or unprotected (notation 'AVP').

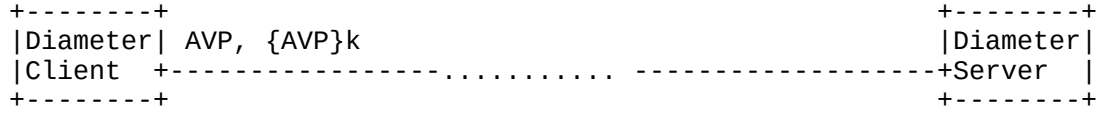

Figure 2: End-to-End Diameter AVP Security Protection.

In the second scenario, shown in Figure 3, a Diameter proxy acts on behalf of the Diameter client with regard to security protection. It applies security protection to outgoing Diameter AVPs and verifies incoming AVPs. Typically, the proxy enforcing the security protection belongs to the same domain as the Diameter client/server without end-to-end security features.

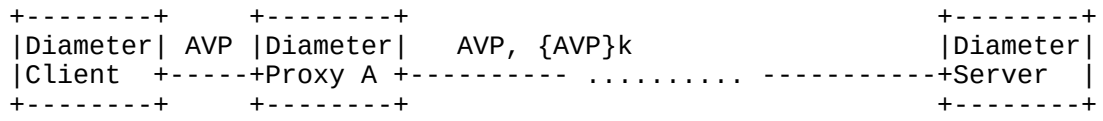

Figure 3: Middle-to-End Diameter AVP Security Protection.

In the third scenario shown in Figure 4 a Diameter proxy acts on behalf of the Diameter server.

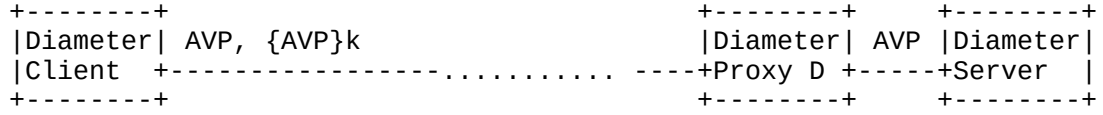

Figure 4: End-to-Middle Diameter AVP Security Protection.

The fourth and the final scenario (see Figure 5) is a combination of the end-to-middle and the middle-to-end scenario shown in Figure 4 and in Figure 3. From a deployment point of view this scenario is easier to accomplish for two reasons: First, Diameter clients and Diameter servers remain unmodified. This ensures that no modifications are needed to the installed Diameter infrastructure. Second, key management is also simplified since fewer number of keys need to be negotiated and provisioned.

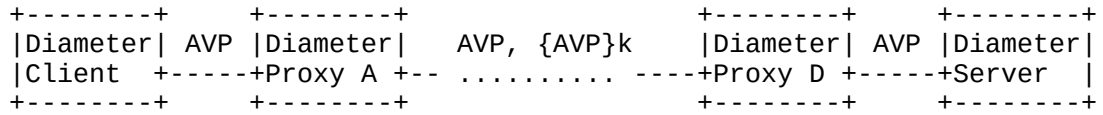

Figure 5: Middle-to-Middle Diameter AVP Security Protection.

Various security threats are mitigated by selectively applying security protection for individual Diameter AVPs. Without protection there is the possibility for password sniffing, confidentiality violation, AVP insertion, deletion or modification. Additionally, applying digital signature offers non-repudiation capabilities; a feature not yet available in today's Diameter deployment. Modification of certain Diameter AVPs may not necessarily be the act of malicious behavior but could also be the result of misconfiguration. An over-aggressively configured firewalling Diameter proxy may also remove certain AVPs. In most cases data

origin authentication and integrity protection of AVPs will provide the most benefits for existing deployments with minimal overhead and (potentially) operating in a full-backwards compatible manner.

5. Requirements

Requirement #1: The solution MUST support an extensible set of cryptographic algorithms.

> Motivation: Solutions MUST be able to evolve to adapt to evolving cryptographic algorithms and security requirements. This may include the provision of a modular mechanism to allow cryptographic algorithms to be updated without substantial disruption to deployed implementations.

Requirement #2: The solution MUST support confidentiality, integrity, and data-origin authentication. Solutions for integrity protection MUST work in a backwards-compatible way with existing Diameter applications.

Requirement #3: The solution MUST support replay protection. All Diameter nodes have access to network time and thus can synchronize their clocks.

Requirement #4: The solution MUST support the ability to delegate security functionality to another entity

> Motivation: As described in Section 4 the ability to let a Diameter proxy to perform security services on behalf of all clients within the same administrative domain is important for incremental deployability. The same applies to the other communication side where a load balancer terminates security services for the servers it interfaces.

Requirement #5: The solution MUST be able to selectively apply their cryptographic protection to certain Diameter AVPs.

> Motivation: Some Diameter applications assume that certain AVPs are added, removed, or modified by intermediaries. As such, it MUST be possible to apply security protection selectively. Furthermore, there are AVPs that MUST NOT be confidentiality protected but MAY still be integrity protected such as those required for Diameter message routing.

Requirement #6: The solution MUST define a mandatory-to-implement cryptographic algorithm.

Motivation: For interoperability purposes it is beneficial to have a mandatory-to-implement cryptographic algorithm specified (unless profiles for specific usage environments specify otherwise).

Requirement #7: The solution MUST support symmetric keys and asymmetric keys.

Motivation: Symmetric and asymmetric cryptographic algorithms provide different security services. Asymmetric algorithms, for example, allow non-repudiation services to be offered.

Requirement #8: A solution for dynamic key management MUST be included in the overall solution framework.

However, it is assumed that no "new" key management protocol needs to be developed; instead existing ones are re-used, if at all possible. Rekeying could be triggered by (a) management actions and (b) expiring keying material.

6. Security Considerations

This entire document focused on the discussion of new functionality for securing Diameter AVPs selectively between non-neighboring nodes.

7. IANA Considerations

This document does not require actions by IANA.

8. Acknowledgments

We would like to thank Guenther Horn, Martin Dolly, Steve Donovan, Lionel Morand and Tom Taylor (rest in peace Tom) for their review comments.

9. References

9.1. Normative References

[1] 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>.

[2] 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>.

9.2. Informative References

[3] Calhoun, P., Farrell, S., and W. Bulley, "Diameter CMS Security Application", draft-ietf-aaa-diameter-cms-sec-04 (work in progress), March 2002.

[4] Eronen, P., Ed., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, DOI 10.17487/RFC4072, August 2005, <http://www.rfc-editor.org/info/rfc4072>.

Authors' Addresses

Hannes Tschofenig
ARM Limited
Austria

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

Jouni Korhonen (editor)
Broadcom Corporation
3151 Zanker Rd.
San Jose, CA 95134
USA

Email: jouni.nospam@gmail.com

Glen Zorn
Network Zen
227/358 Thanon Sanphawut
Bang Na Bangkok 10260
Thailand

Email: glenzorn@gmail.com

Kervin Pillay
Oracle Communications
100 Crosby Drive
Bedford, Massachusettes 01730
USA

Email: kervin.pillay@oracle.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

M. Jones

M. Liebsch

L. Morand

March 21, 2016

Diameter Group Signaling
draft-ietf-dime-group-signaling-06.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", "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 server 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 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. 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 node 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 node 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. Prerequisite for deletion of a session group is that the Diameter node created the session beforehand, hence the node became the group owner.

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 (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 a different Diameter node created 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

*) Editors’ note: The protocol specification in this document does not consider overruling a node’s assignment of a session to a session group. Here, overruling is to be understood as a node changing the group(s) assignment as per the node’s request. Group signaling enabled applications may take such protocol support and associated protocol semantics into account in their specification. One exception is adopted in this specification, which allows a Diameter server to reject a group assignment as per the client’s request.

4. Protocol Description

4.1. Session Grouping Capability Discovery

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

4.1.1. 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, as well as out-of-band mechanisms for capability exchange 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. Implicit Capability Discovery

If no explicit mechanism for capability discovery is deployed to enable Diameter nodes to learn about nodes' capability to support session grouping and group commands, a Diameter node SHOULD append the Session-Group-Capability-Vector AVP to any Diameter messages exchanged with its nodes to announce its capability to support session grouping and session group operations. Implementations following the specification as per this document 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 Diameter node maintains a log to remember the node's capability to support group commands.

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 application to determine the policy of session grouping. In case 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. 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 should be 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 [RFC4005], containing one or more session group identifiers. Each of these groups needs 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 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

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 includes in the request the Session-Group-Info AVP in the request including the Session-Group-Control-Vector AVP with 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 comprises at least one Session-Group-Info AVP having the SESSION_GROUP_ALLOCATION_ACTION flag set 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 but be treated as single session.

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. In case one or multiple identified session groups are not already stored by the server, the server must store the new 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 [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 a case, the server adds 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 [RFC4005], 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.

If the Diameter server receives a command request from a Diameter client and the command comprises one or multiple Session-Group-Info AVPs and none of them including 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 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 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.

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 and continue to treat 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.

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 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) 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. 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.

4.2.3. 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 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.

4.3. 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 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 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 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 sends a request including the Group-Response-Action AVP set to ALL_GROUPS (1) or PER_GROUP (2), it MUST expect some delays 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 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.

The Diameter node 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 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 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 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. 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

In case of a present stateful Proxy Agent between a Diameter client and a Diameter server, this specification assumes that the Proxy Agent is aware of session groups and session group handling. 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 case a 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.

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 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> 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 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

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. The DIME working group is currently working on solutions for providing end-to-end security features. When available, these features should 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 assume 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 for the thorough review and comments on the adopted 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4005] Calhoun, P., Zorn, G., Spence, D., and D. Mitton, "Diameter Network Access Server Application", RFC 4005, DOI 10.17487/RFC4005, August 2005, <http://www.rfc-editor.org/info/rfc4005>.

[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>.

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	Send GASA with Result-Code = SUCCESS, Send STR	Discon

	client will comply with request to end the session	per session	
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	Discon. user/device	Idle

received.

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)
Internet-Draft
Intended status: Standards Track
Expires: July 1, 2019

M. Jones

M. Liebsch

L. Morand

December 28, 2018

Diameter Group Signaling
draft-ietf-dime-group-signaling-12.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.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 1, 2019.

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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 mid-session. For example, if a user's subscription profile changes mid-session, a Diameter server 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 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. 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 node 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 node 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. Prerequisite for deletion of a session group is that the Diameter node created the session beforehand, hence the node became the group owner.

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 (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 a different Diameter node created 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

*) Editors' note: The protocol specification in this document does not consider overruling a node's assignment of a session to a session group. Here, overruling is to be understood as a node changing the group(s) assignment as per the node's request. Group signaling enabled applications may take such protocol support and associated protocol semantics into account in their specification. One exception is adopted in this specification, which allows a Diameter server to reject a group assignment as per the client's request.

4. Protocol Description

4.1. Session Grouping Capability Discovery

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

4.1.1. 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, as well as out-of-band mechanisms for capability exchange 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. Implicit Capability Discovery

If no explicit mechanism for capability discovery is deployed to enable Diameter nodes to learn about nodes' capability to support session grouping and group commands, a Diameter node SHOULD append the Session-Group-Capability-Vector AVP to any Diameter messages exchanged with its nodes to announce its capability to support session grouping and session group operations. Implementations following the specification as per this document 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 Diameter node maintains a log to remember the node's capability to support group commands.

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 application to determine the policy of session grouping. In case 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. 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 should be 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 comprises 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. In case one or multiple identified session groups are not already stored by the server, the server MUST store the new 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 Diameter server accepts the client's request for a group assignment, but 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 as information to the client only 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 comprises one or multiple Session-Group-Info AVPs and none of them includes 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 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. 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. When the Diameter client is confident that the Diameter server supports session grouping and

group signaling, the Diameter client SHOULD 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 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) 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. 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.

4.2.3. 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 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 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 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 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. When a server sends, as example, a Re-Authorization Request (RAR) or a service-specific server-initiated request to the client, it can indicate to the client whether to process the request, after

having sent the RAA to the server, with either sending a single RAR message for all identified groups (server sets the Group-Response-Action AVP to ALL_GROUPS (1)), or sending a single RAR message for each identified group individually (server sets the Group-Response-Action AVP to PER_GROUP (1)). The server may also request the client to follow-up with a single RAR message per impacted session (server sets the Group-Response-Action AVP to PER_SESSION). In such case, the client sends only one RAR message for an impacted session in case the session is included in more than one of the identified session groups.

If the sender sends a request including the Group-Response-Action AVP set to ALL_GROUPS (1) or PER_GROUP (2), it MUST 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 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 case of a present stateful Proxy Agent between a Diameter client and a Diameter server, this specification assumes that the Proxy Agent is aware of session groups and session group handling. 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.

In case 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 SHOULD be 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)

			AVP Flag rules			
Attribute Name	AVP Code	Value Type	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 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> 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. The DIME working group is currently working on solutions for providing end-to-end security features. When available, these features should 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 assume 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>.

[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	Send GASA with Result-Code = SUCCESS,	Discon

	received group(s) and client will comply with request to end the session	Send GSTR.	
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	Send GRAA, Send service specific re-auth req	Pending

	client will perform subsequent re-auth	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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

B. Campbell
S. Donovan, Ed.
Oracle
JJ. Trottin
Nokia

March 18, 2016

Diameter Load Information Conveyance
draft-ietf-dime-load-02

Abstract

This document defines a mechanism for sharing of Diameter load information. [RFC7068] 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) [RFC7683] solution describes a mechanism meeting most of the requirements, but does not currently include the ability to send load information.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 19, 2016.

Copyright Notice

Table of Contents

1. Introduction

[RFC7068] describes requirements for Overload Control in Diameter [RFC6733]. The DIME working group has finished the Diameter Overload Information Conveyance (DOIC) mechanism [RFC7683]. 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 [RFC7068] 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 deferred load information conveyance to a DOIC extension or a separate mechanism.

This document defines a mechanism that addresses the load-related requirements from RFC 7068.

2. Terminology and Abbreviations

DOIC

Diameter Overload Information Conveyance ([RFC7683])

Load

The relative capacity of a Diameter node. A low load level indicates that the Diameter node is under utilized. A high load level indicates that the node is closer to being fully utilized.

Offered Load

The actual traffic sent to the reporting node after overload abatement and routing decisions are made.

Reporting, Reacting Node

Reporting node and reacting node terminology is defined in [RFC7683].

Routing Information

Routing Information - Routing information referred to in this document can include the Routing and Peer tables defined in RFC 6733. It can also include other implementation specific tables used to store load information. This document does not define the structure of such tables.

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. Background

4.1. Differences between Load and Overload information

Previous discussions of how to solve the load-related requirements in [RFC7068] have shown that people did not had an agreed-upon concept of how "load" information differs from "overload" information. While the two concepts are highly interrelated, 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 is 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) [RFC7683] 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.

4.2. 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 [RFC7683]. 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 Appendix A) 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.

It should be noted that a Diameter node will need to process both Load reports and Overload reports from the same Diameter node. The reacting node for the Overload report always has the responsibility to reduce the amount of Diameter traffic sent to the overloaded node. If, or how, the reacting node uses Load information to achieve this is left as an implementation decision.

5. Solution Overview

The mechanism defined here for the conveyance of load information is similar in some ways to the mechanism defined for DOIC and is different in other ways.

As with DOIC, load information is conveyed by piggy-backing the load AVPs on existing Diameter applications.

There are two primary differences. First, there is no capability negotiation process for load. The sender of the load information is sending it with the expectation that any supporting nodes will use it when making routing decisions. If there are no nodes that support the Load mechanism then the load information is ignored.

The second big difference between DOIC and Load is visibility of the DOIC or Load information within a Diameter network. DOIC information is sent end-to-end resulting in the ability of all nodes in the path of the answer message that carries the OC-OLR AVP to act on the information. The DOIC overload reports much remain in the message all the way from the reporting node to the node that is the target for the answer message.

For the Load mechanism there are two types of load reports.

The first is the load of the endpoint sending the answer message. This load report is carried end-to-end to enable any nodes that make server selection decisions to use the load status of the sending endpoint as part of the server selection decision.

The second type of load report is a peer report. This report is used by Diameter nodes as part of the logic to select the next hop Diameter node and, as such, do not have significance beyond the peer node. These load reports are removed by the first supporting Diameter node to receive the report.

Because load reports can traverse Diameter nodes that do not support the Load mechanism, it is necessary to include the identity of the node to which the load report applies as part of the load report. This allows for a Diameter node to verify that a load report applies to its peer or if it should be ignored.

The load report includes the relative load of the sending node. This relative load is specified in a manner consistent with that defined for DNS SRV [RFC2782].

The goal is make it possible to use both the load values received as a part of the Diameter Load mechanism and weight values received as a result of a DNS SRV query. As a result, the Diameter load value has a range of 0-65535. This value and DNS SRV weight values are then used in a distribution algorithm similar to that specified in [RFC2782].

The DNS SRV distribution algorithm results in more messages being sent to a node with a higher weight value. As a result, a higher Diameter load value indicates a LOWER load on the sending node. A node that is heavily loaded sends a lower Diameter load value. Stated another way, a node that has zero load would have a load value of 65535. A node that is 100% loaded would have a load value of 0.

The distribution algorithm used by Diameter nodes supporting the Diameter Load mechanism is an implementation decision but it needs to result in similar behavior as the algorithm specified in [RFC2782].

The method for calculating the load value included in the load report is also left as an implementation decision.

The frequency for sending of load reports is also left as an implementation decision. The sending node might choose to send load reports in all messages or it might choose to only send load reports when the load value has changed by some implementation specific amount. The important consideration is that all nodes needing the load information have a sufficiently accurate view of the nodes load.

5.1. Theory of Operation

This section outlines how the Diameter Load mechanism is expected to work.

For this discussion, assume the following Diameter network configuration:

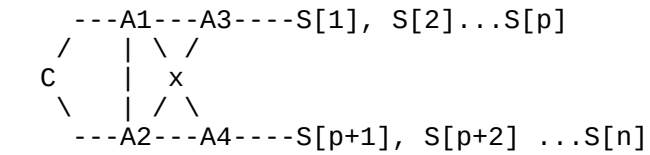

Figure 1: Example Diameter Network

Note that in this diagram, S[1], S[2] through S[p] are peers to A3. S[p+1], S[p+2] through S[n] are peers to A4.

Also assume that the request for a Diameter transaction takes the following path:

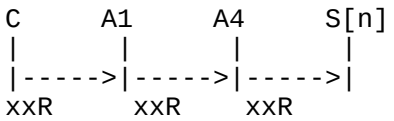

Figure 2: Request Message Path

When sending the answer message, an endpoint node that supports the Diameter Load mechanism includes it’s own load information in the answer message. Because it is a Diameter endpoint it includes a HOST load report.

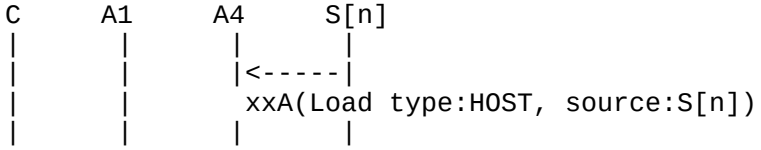

Figure 3: Answer Message from S[n]

If Agent A4 supports the Load mechanism then it will verify that the load information received is valid. For a HOST load report this is achieved by matching the identity included in the load information with the identity of the host node from which the answer message was received.

Note: If A4 does not support the Load mechanism then it will relay the answer message without doing any processing on the load information. In this case the load information AVPs will be relayed without change.

If the identity included in the load information AVPs matches the identity of the host from which the load information is received then Agent A4 stores the load information for S[n] in its routing information.

Because the load report is an HOST load report, A4 leaves the load report in the message it relays.

A4 then calculates its own load information and inserts load information AVPs of type PEER in the message before sending the message to A1:

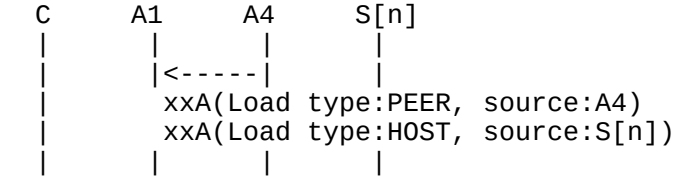

Figure 4: Answer Message from A4

If A1 supports the Load mechanism then it processes each of the Load reports it receives separately.

For the PEER load report, A1 first determines if the source of the report indicated in the load report matches the DiameterIdentity of the Diameter node from which the request was received. If the identities do not match then the PEER load report is discarded. If the identities match then A1 saves the load information in its routing information for routing of subsequent request messages. In both cases A1 strips the PEER load report from the message.

For the HOST load report, A1's actions depend on whether A1 is responsible for doing server selection. If A1 is not doing server selection then A1 ignores the HOST load report. If A1 is responsible for doing server selection then it stores the load information for S[n] in its routing information for the handling of subsequent request messages. In both cases A1 leaves the HOST report in the message.

A1 then calculates its own load information and inserts load information AVPs of type PEER in the message before sending the message to A1:

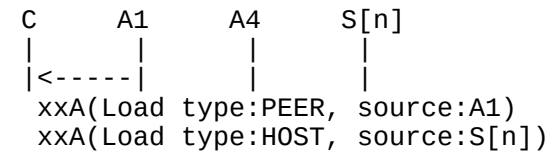

Figure 5: Answer Message from A1

As with A1, C processes each load report separately.

For the PEER load report, C follows the same procedure as A1 for determining if the Load report was received from the peer from which the report was sent and, when finding it does, stores the load information for use when making future routing decisions.

For the HOST load report, C saves the load information only if it is responsible for doing server selection.

The Load information received by all nodes is then used for routing of subsequent request messages.

6. Load Mechanism Procedures

This section defines the normative behaviors for the Load mechanism.

6.1. Reporting Node Behavior

This section defines the procedures of Diameter reporting nodes that generate load reports.

6.1.1. Endpoint Reporting Node Behavior

A Diameter endpoint that supports the Diameter Load mechanism MUST include a load report of type HOST in sufficient answer messages to ensure that all consumers of the load information receive timely updates.

The Diameter endpoint MUST include it’s own DiameterIdentity in the Source-ID AVP included in the Load AVP.

The Diameter endpoint MUST include a Load-Type AVP of type HOST in the Load AVP.

The Diameter endpoint MUST include its load value in the Value AVP in the load AVP.

The method for determining the load value included in the load report is an implementation decision.

The frequency of sending load reports is an implementation decision.

For instance, if the only consumer of the load reports is the endpoints peer then the endpoint can choose to only include a load report when the load of the endpoint has changed by a meaningful percentage. If there are consumers of the endpoint load report other then the endpoints peer (this will be the case if other nodes are responsible for server selection) then the endpoint might choose to include load reports in all answer messages as a way of ensuring that all nodes doing server selection get accurate load information.

6.1.2. Agent Reporting Node Behavior

A Diameter agent that supports the Diameter Load mechanism MUST include a PEER load report in sufficient answer messages to ensure that all users of the load information receive timely updates.

The Diameter agent MUST include it’s own DiameterIdentity in the Source-ID AVP included in the Load AVP.

The Diameter agent MUST include a Load-Type AVP of type PEER in the Load AVP.

The Diameter agent MUST include its load value in the Value AVP in the load AVP.

The method for determining the load value included in the load report is an implementation decision.

The frequency of sending load reports is an implementation decision.

Note: In the case of peer load reports it is only necessary to include load reports when the load value has changed by some meaningful value, as long as the agent insures that all peers receive the report. It is also acceptable to include the load report in every answer message handled by the Diameter agent.

6.2. Receiving Node Behavior

This section defines the behavior of Diameter nodes processing load reports.

A Diameter node MUST be prepared to process load reports of type HOST and of type PEER, as indicated in the Load-Type AVP included in the

Load AVP received in the same answer message or from multiple answer messages.

Note that the node needs to be able to handle messages with no load reports, messages with just a PEER load report, messages with just an HOST load report and messages with both types of load reports.

If the Diameter node is not responsible for doing server selection then it SHOULD ignore load reports of type HOST.

If the Diameter node is responsible for doing server selection then it SHOULD save the load value included in the Value AVP included in the Load AVP of type HOST in its routing information.

If the Diameter node receives a Load report of type PEER then the Diameter node MUST determine if the Load report was inserted into the answer message by the peer from which the message was received. This is achieved by comparing the DiameterIdentity associated with the connection from which the message was received with the DiameterIdentity included in the Source-ID AVP in the Load report.

If the Diameter node determines that the Load report of type PEER was not received from the peer that sent or relayed the answer message then the node MUST ignore the Load report.

If the Diameter node determines that the Load report of type PEER was received from the peer that sent or relayed the answer message then the node SHOULD save the load information in its routing information.

How a Diameter node uses load information for making routing decisions is an implementation decision. However, the distribution algorithm MUST result in similar behavior as the algorithm described for the use of weigth values in [RFC2782].

6.3. Extensibility

The Load mechanism can be extended to include additional information in the load reports.

Any extension may define new AVPs for use in Load reports. These new AVPs SHOULD be defined to be extensions to the Load 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.

As with any Diameter specification, [RFC6733] requires all new AVPs to be registered with IANA. See Section 9 for the required procedures.

7. Attribute Value Pairs

The section defines the AVPs required for the Load mechanism.

7.1. Load AVP

The Load AVP (AVP code TBD1) is of type Grouped and is used to convey load information between Diameter nodes.

```
 Load ::= < AVP Header: TBD1 >
          [ Load-Type ]
          [ Load-Value ]
          [ SourceID ]
        * [ AVP ]
```

7.2. Load-Type AVP

The Load-Type AVP (AVP code TBD2) is of type Enumerated. It is used to convey the type of Diameter node that sent the load information. The following values are defined:

HOST 0 The load report is for a host.

PEER 1 The load report is for a peer.

7.3. Load-Value AVP

The Load-Value AVP (AVP code TBD3) is of type Unsigned64. It is used to convey relative load information about the sender of the load report.

The Load-Value AVP is specified in a manner similar to the weight value in DNS SRV ([RFC2782]).

The Load-Value has a range of 0-65535.

A higher value indicates a lower load on the sending node. A lower value indicates that the sending node is heavily loaded.

> Stated another way, a node that has zero load would have a load value of 65535. A node that is 100% loaded would have a load value of 0.

7.4. SourceID AVP

The SourceID AVP is defined in [I-D.ietf-dime-agent-overload]. It is used to identify the Diameter node that sent the Load report.

7.5. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
Load	TBD1	x.1	Grouped		V
Load-Type	TBD2	x.2	Enumerated		V
Load-Value	TBD3	x.3	Unsigned64		V
SourceID	TBD4	x.4	DiameterIdentity		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. 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.

9. IANA Considerations

9.1. AVP Codes

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

9.2. New Registries

This document makes no new registry requests of IANA.

10. References

10.1. Normative References

[I-D.ietf-dime-agent-overload]
Donovan, S., "Diameter Agent Overload", draft-ietf-dime-agent-overload-02 (work in progress), August 2015.

[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>.

[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>.

[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>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

10.2. Informative References

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

Appendix A. 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.

A.1. No Agent

Figure 6 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.

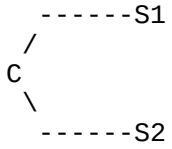

Figure 6: Basic Client Server Scenario

If a node supports dynamic discovery, it will not obtain load information from the nodes with which it has no Diameter connection established. Nevertheless it might take into account the load information from the other nodes to decide to add connections to new nodes with the dynamic discovery mechanism.

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

A.2. Single Agent

Figure 7 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.

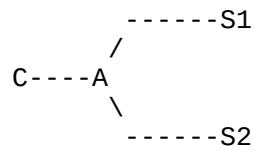

Figure 7: Simple Agent Scenario

A.3. Multiple Agents

Figure 8 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 [RFC7683], 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.

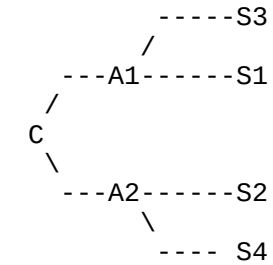

Figure 8: Multiple Agents and Servers

A.4. Linked Agents

Figure 9 shows a scenario similar to that of Figure 8, 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 8, 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.

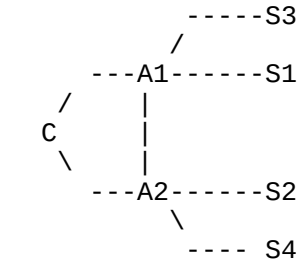

Figure 9: Linked Agents

Figure 10 is a variant of Figure 9. 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.

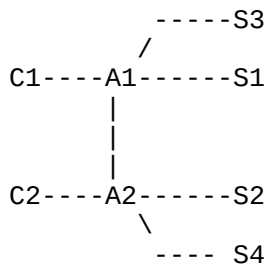

Figure 10: Linked Agents

A.5. Shared Server Pools

Figure 11 is similar to Figure 9, 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.

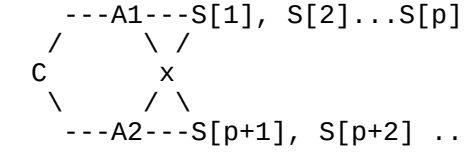

Figure 11: Shared Server Pools

A.6. Agent Chains

The scenario in Figure 12 is similar to that of Figure 8, 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.

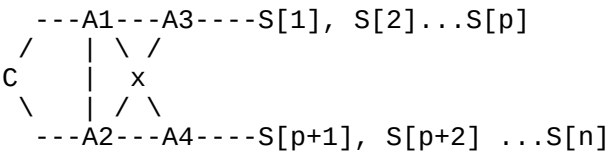

Figure 12: Agent Chains

A.7. Fully Meshed Layers

Figure 13 extends the scenario in Figure 11 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.

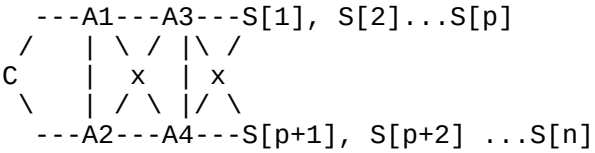

Figure 13: Full Mesh

A.8. Partitions

A Diameter network with multiple servers is said to be "partitioned" when only a subset of available servers can serve 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.

A.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.

A.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. The handling of such a smooth increase is implementation specific but it can rely on the evolution of load information received from the new node and from the other nodes.

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. The handling of the node removal is implementation specific but it can rely on the evolution of the load information received from the node to be removed

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
Nokia
Route de Villejust
91620 Nozay
France

Email: jean-jacques.trottin@nokia.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 23, 2017

B. Campbell
S. Donovan, Ed.
Oracle
JJ. Trottin
Nokia

March 22, 2017

Diameter Load Information Conveyance
draft-ietf-dime-load-09

Abstract

RFC7068 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. RFC7683 (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 defines a mechanism for conveying of Diameter load information.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 23, 2017.

Copyright Notice

Table of Contents

1. Introduction

[RFC7068] describes requirements for Overload Control in Diameter [RFC6733]. The DIME working group has finished the Diameter Overload Information Conveyance (DOIC) mechanism [RFC7683]. As currently specified, DOIC fulfills some, but not all, of the requirements.

In particular, DOIC does not fulfill Req 23 and Req 24:

> REQ 23: The solution MUST provide sufficient information to enable a load-balancing node to divert messages that are rejected or otherwise throttled by an overloaded upstream node to other upstream nodes that are the most likely to have sufficient capacity to process them.
>
> REQ 24: The solution MUST provide a mechanism for indicating load levels, even when not in an overload condition, to assist nodes in making decisions to prevent overload conditions from occurring.

There are several other requirements in [RFC7068] 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 when publishing DOIC [RFC7683] 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 deferred load information conveyance to a DOIC extension or a separate mechanism.

This document defines a mechanism that addresses the load-related requirements from RFC 7068.

2. Terminology and Abbreviations

AVP

> Attribute Value Pair

DOIC

> Diameter Overload Information Conveyance ([RFC7683])

Load

> The relative usage of the Diameter message processing capacity of a Diameter node. A low load level indicates that the Diameter

node is under utilized. A high load level indicates that the node is closer to being fully utilized.

Offered Load

The actual traffic sent to the reporting node after overload abatement and routing decisions are made.

Reporting Node

Reporting Node: A Diameter node that generates a load report.

Reacting Node

Reacting Node: A Diameter node that acts upon a load report.

Routing Information

Routing Information referred to in this document can include the Routing and Peer tables defined in RFC 6733. It can also include other implementation specific tables used to store load information. This document does not define the structure of such tables.

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. Background

4.1. Differences between Load and Overload information

Previous discussions of how to solve the load-related requirements in [RFC7068] have shown that people did not have an agreed-upon concept of how "load" information differs from "overload" information. While the two concepts are highly interrelated, there are two primary differences. First, a Diameter node always has a load. At any given time that load may be effectively zero, effectively fully loaded, or somewhere in between. In contrast, overload is an exceptional condition. A node only has overload information when it is 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) [RFC7683] indicates an explicit request for action on the part of the reacting node. That is, the OLR requests that the reacting node reduces the offered load -- the actual traffic sent to the reporting node after overload abatement and routing decisions are made -- by an indicated amount (by default), or as prescribed by the selected abatement algorithm. 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 the reduction of offered load. It is also reasonable for a Diameter node to decide to increase the offered load based on load information.

4.2. 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 [RFC7683]. 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, but the diameter node can take other proactive steps as well.

If the loaded nodes are Diameter servers (or clients in the case of server-to-client transactions), both of these uses of load information should be accomplished by a Diameter node that performs server selection (selection of the Diameter endpont to which the request is to be routed for processing). 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 Appendix A) 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.

It should be noted that a Diameter node will need to process both Load reports and Overload reports from the same Diameter node. The reacting node for the Overload report always has the responsibility to reduce the amount of Diameter traffic sent to the overloaded node. If, or how, the reacting node uses load information to achieve this is left as an implementation decision.

5. Solution Overview

The mechanism defined here for the conveyance of load information is similar in some ways to the mechanism defined for DOIC and is different in other ways.

As with DOIC, load information is conveyed by piggy-backing the Load AVPs on existing Diameter applications.

There are two primary differences. First, there is no capability negotiation process for load. The sender of the load information is sending it with the expectation that any supporting nodes will use it when making routing decisions. If there are no nodes that support the Load mechanism then the load information is ignored.

The second big difference between DOIC and Load is visibility of the DOIC or load information within a Diameter network. DOIC information is sent end-to-end resulting in the ability of all nodes in the path of the answer message that carries the OC-OLR AVP to act on the information, although only one node actually comsumes and reacts to the report. The DOIC overload reports remain in the message all the way from the reporting node to the node that is the target for the answer message.

For the Load mechanism there are two types of Load reports and only the first one is transmitted end-to-end.

The first type of Load report is a HOST report which contains the load of the endpoint sending the answer message. This Load report is carried end-to-end to enable any nodes that make server selection decisions to use the load status of the sending endpoint as part of the server selection decision. Unlike with DOIC, more than one node may make use of the load information received.

The second type of Load report is a PEER report. This report is used by Diameter nodes as part of the logic to select the next-hop Diameter node and, as such, does not have significance beyond the peer node. Load reports of type PEER are removed by the first supporting Diameter node to receive the report.

Because Load reports can traverse Diameter nodes that do not support the Load mechanism, it is necessary to include the identity of the node to which the Load report applies as part of the Load report. This allows for a Diameter node to verify that a Load report applies to its peer or if it should be ignored.

The Load report includes a value indicating relative load of the sending node, specified in a manner consistent with that defined for DNS SRV [RFC2782].

The goal is to make it possible to use both the load values received as a part of the Diameter Load mechanism and weight values received as a result of a DNS SRV query. As a result, the Diameter load value has a range of 0-65535. This value and DNS SRV weight values are then used in a distribution algorithm similar to that specified in [RFC2782].

The DNS SRV distribution algorithm results in more messages being sent to a node with a higher weight value. As a result, a higher Diameter load value indicates a LOWER load on the sending node. A node that is heavily loaded sends a lower Diameter load value. Stated another way, a node that has zero load would have a load value of 65535. A node that is 100% loaded would have a load value of 0.

The distribution algorithm used by Diameter nodes supporting the Diameter Load mechanism is an implementation decision but it needs to result in similar behavior to the algorithm described for the use of weight values specified in [RFC2782].

The method for calculating the load value included in the Load report is also left as an implementation decision.

The frequency for sending of Load reports is also left as an implementation decision. The sending node might choose to send Load reports in all messages or it might choose to only send Load reports when the load value has changed by some implementation specific amount. The important consideration is that all nodes needing the load information have a sufficiently accurate view of the node's load.

5.1. Theory of Operation

This section outlines how the Diameter Load mechanism is expected to work.

For this discussion, assume the following Diameter network configuration:

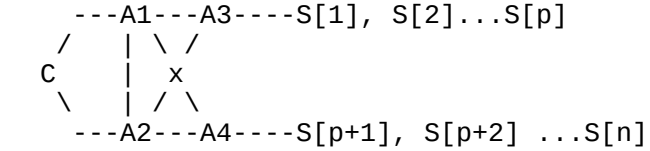

Figure 1: Example Diameter Network

Note that in this diagram, S[1], S[2] through S[p] are peers to A3. S[p+1], S[p+2] through S[n] are peers to A4.

Also assume that the request for a Diameter transaction takes the following path:

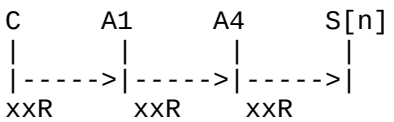

Figure 2: Request Message Path

When sending the answer message, an endpoint node that supports the Diameter Load mechanism includes its own load information in the answer message. Because it is a Diameter endpoint it includes a HOST Load report.

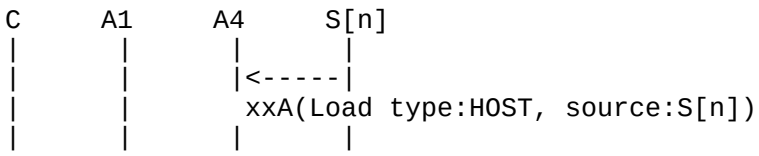

Figure 3: Answer Message from S[n]

If Agent A4 supports the Load mechanism then A4's actions depend on whether A4 is responsible for doing server selection. If A4 is not doing server selection then A4 ignores the HOST Load report. If A4 is responsible for doing server selection then it stores the load information for S[n] in its routing information for the handling of subsequent request messages. In both cases A4 leaves the HOST report in the message.

Note: If A4 does not support the Load mechanism then it will relay the answer message without doing any processing on the load information. In this case the load information AVPs will be relayed without change.

A4 then calculates its own load information and inserts load information AVPs of type PEER in the message before sending the message to A1.

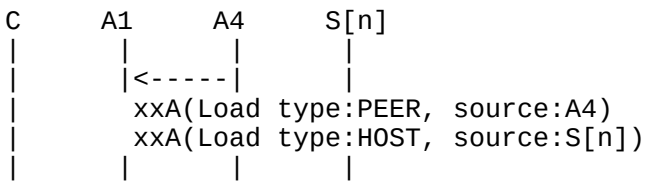

Figure 4: Answer Message from A4

If A1 supports the Load mechanism then it processes each of the Load reports it receives separately.

For the PEER Load report, A1 first determines if the source of the report indicated in the Load report matches the DiameterIdentity of the Diameter node from which the request was received. If the identities do not match then the PEER Load report is discarded. If the identities match then A1 saves the load information in its routing information for routing of subsequent request messages. In both cases A1 strips the PEER Load report from the message.

For the HOST Load report, A1's actions depend on whether A1 is responsible for doing server selection. If A1 is not doing server selection then A1 ignores the HOST Load report. If A1 is responsible for doing server selection then it stores the load information for S[n] in its routing information for the handling of subsequent request messages. In both cases A1 leaves the HOST report in the message.

A1 then calculates its own load information and inserts load information AVPs of type PEER in the message before sending the message to C:

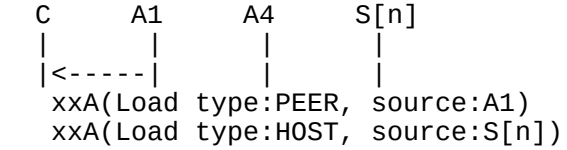

Figure 5: Answer Message from A1

As with A1, C processes each Load report separately.

For the PEER Load report, C follows the same procedure as A1 for determining if the Load report was received from the peer from which the report was sent. When finding it does, C stores the load information for use when making future routing decisions.

For the HOST Load report, C saves the load information only if it is responsible for doing server selection.

The load information received by all nodes is then used for routing of subsequent request messages.

6. Load Mechanism Procedures

This section defines the normative behaviors for the Load mechanism.

6.1. Reporting Node Behavior

This section defines the procedures of Diameter reporting nodes that generate Load reports.

6.1.1. Endpoint Reporting Node Behavior

A Diameter endpoint that supports the Diameter Load mechanism MUST include a Load report of type HOST in sufficient answer messages to

ensure that all consumers of the load information receive timely updates.

The Diameter endpoint MUST include its own DiameterIdentity in the SourceID AVP included in the Load AVP.

The Diameter endpoint MUST include a Load-Type AVP of type HOST in the Load AVP.

The Diameter endpoint MUST include its load value in the Load-Value AVP in the Load AVP.

The LOAD value should be calculated in a way that reflects the available load independently of the weight of each server, in order to accurately compare LOAD values from different nodes. Any specific LOAD value needs to identify the same amount of available capacity, regardless the Diameter node that calculates the value.

The mechanism used to calculate the LOAD value that fulfills this requirement is an implementation decision.

The frequency of sending Load reports is an implementation decision.

For instance, if the only consumer of the Load reports is the endpoint's peer then the endpoint can choose to only include a Load report when the load of the endpoint has changed by a meaningful percentage. If there are consumers of the endpoint Load report other then the endpoint's peer (this will be the case if other nodes are responsible for server selection) then the endpoint might choose to include Load reports in all answer messages as a way of ensuring that all nodes doing server selection get accurate load information.

6.1.2. Agent Reporting Node Behavior

A Diameter Agent that supports the Diameter Load mechanism MUST include a PEER Load report in sufficient answer messages to ensure that all users of the load information receive timely updates.

The Diameter Agent MUST include its own DiameterIdentity in the SourceID AVP included in the Load AVP.

The Diameter Agent MUST include a Load-Type AVP of type PEER in the Load AVP.

The Diameter Agent MUST include its load value in the Load-Value AVP in the Load AVP.

The LOAD value should be calculated in a way that reflects the available load independently of the weight of each agent, in order to accurately compare LOAD values from different nodes. Any specific LOAD value needs to identify the same amount of available capacity, regardless the Diameter node that calculates the value.

The mechanism used to calculate the LOAD value that fulfills this requirement is an implementation decision.

The frequency of sending Load reports is an implementation decision.

> Note: In the case of peer Load reports it is only necessary to include Load reports when the load value has changed by some meaningful value, as long as the agent ensures that all peers receive the report. It is also acceptable to include the Load report in every answer message handled by the Diameter Agent.

6.2. Reacting Node Behavior

This section defines the behavior of Diameter nodes processing Load reports.

A Diameter node that supports the Diameter Load mechanism MUST be prepared to process Load reports of type HOST and of type PEER, as indicated in the Load-Type AVP included in the Load AVP received in the same answer message or from multiple answer messages.

> Note that the node needs to be able to handle messages with no load reports, messages with just a PEER Load report, messages with just an HOST Load report and messages with both types of Load reports.

If the Diameter node is not responsible for doing server selection then it SHOULD ignore Load reports of type HOST.

If the Diameter node is responsible for doing server selection then it SHOULD save the load value included in the Load-Value AVP included in the Load AVP of type HOST in its routing information.

If the Diameter node receives a Load report of type PEER then the Diameter node MUST determine if the Load report was inserted into the answer message by the peer from which the message was received. This is achieved by comparing the DiameterIdentity associated with the connection from which the message was received with the DiameterIdentity included in the SourceID AVP in the Load report.

If the Diameter node determines that the Load report of type PEER was not received from the peer that sent or relayed the answer message then the node MUST ignore the Load report.

If the Diameter node determines that the Load report of type PEER was received from the peer that sent or relayed the answer message then the node SHOULD save the load information in its routing information.

In all cases, a Diameter Agent MUST strip all Load reports of type PEER received in answer messages.

> Note: This ensures that there will be precisely one Load report of type PEER, that of the Diameter node sending the message, in any answer messages sent by the Diameter Agent.

How a Diameter node uses load information for making routing decisions is an implementation decision. However, the distribution algorithm MUST result in similar behavior as the algorithm described for the use of weight values in [RFC2782].

6.3. Extensibility

The Load mechanism can be extended to include additional information in the Load reports.

Any extension may define new AVPs for use in Load reports. These new AVPs SHOULD be defined to be extensions to the Load 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.

As with any Diameter specification, [RFC6733] requires all new AVPs to be registered with IANA. See Section 9 for the required procedures.

6.4. 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 can attempt to ensure a smooth increase of the traffic to the new node, avoiding an immediate spike of traffic to the new node. The method for handling of such a smooth increase is implementation specific but it can rely on the evolution of load information received from the new node and from the other nodes.

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 not be sufficient. The method for handling of the node removal is implementation specific but it can rely on the evolution of the load information received from the node to be removed.

7. Attribute Value Pairs

The section defines the AVPs required for the Load mechanism.

7.1. Load AVP

The Load AVP (AVP code TBD1) is of type Grouped and is used to convey load information between Diameter nodes.

```
 Load ::= < AVP Header: TBD1 >
          [ Load-Type ]
          [ Load-Value ]
          [ SourceID ]
        * [ AVP ]
```

7.2. Load-Type AVP

The Load-Type AVP (AVP code TBD2) is of type Enumerated. It is used to convey the type of Diameter node that sent the load information. The following values are defined:

HOST 0 The Load report is for a host.

PEER 1 The Load report is for a peer.

7.3. Load-Value AVP

The Load-Value AVP (AVP code TBD3) is of type Unsigned64. It is used to convey relative load information about the sender of the Load report.

The Load-Value AVP is specified in a manner similar to the weight value in DNS SRV ([RFC2782]).

The Load-Value has a range of 0-65535.

A higher value indicates a lower load on the sending node. A lower value indicates that the sending node is heavily loaded.

Stated another way, a node that has zero load would have a load value of 65535. A node that is 100% loaded would have a load value of 0.

7.4. SourceID AVP

The SourceID AVP is defined in [I-D.ietf-dime-agent-overload]. It is used to identify the Diameter node that sent the Load report.

7.5. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
Load	TBD1	x.1	Grouped		V
Load-Type	TBD2	x.2	Enumerated		V
Load-Value	TBD3	x.3	Unsigned64		V
SourceID	TBD4	x.4	DiameterIdentity		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. 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.

Given that routing decisions are impacted by load information, there is potential for negative impacts on a Diameter network caused by erroneous or malicious Load reports. This includes the malicious changing of load values by Diameter Agents.

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 and does not support end-to-end security mechanisms, any solution that sends load information to non-peer nodes requires a transitive-trust model.

9. IANA Considerations

9.1. AVP Codes

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

9.2. New Registries

This document makes no new registry requests of IANA.

10. References

10.1. Normative References

[I-D.ietf-dime-agent-overload]
Donovan, S., "Diameter Agent Overload", draft-ietf-dime-agent-overload-02 (work in progress), August 2015.

[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>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[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>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

10.2. Informative References

[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>.

Appendix A. Topology Scenarios

This section presents a number of Diameter topology scenarios, and discusses how load information might be used in each scenario.

A.1. No Agent

Figure 6 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.

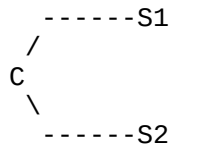

Figure 6: Basic Client Server Scenario

If a node supports dynamic discovery, it will not obtain load information from the nodes with which it has no Diameter connection established. Nevertheless it might take into account the load information from the other nodes to decide to add connections to new nodes with the dynamic discovery mechanism.

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

A.2. Single Agent

Figure 7 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.

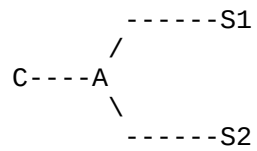

Figure 7: Simple Agent Scenario

A.3. Multiple Agents

Figure 8 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 [RFC7683], 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.

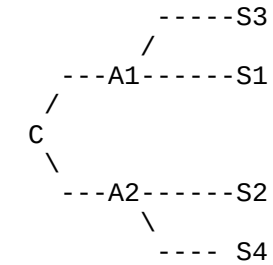

Figure 8: Multiple Agents and Servers

A.4. Linked Agents

Figure 9 shows a scenario similar to that of Figure 8, 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 8, 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.

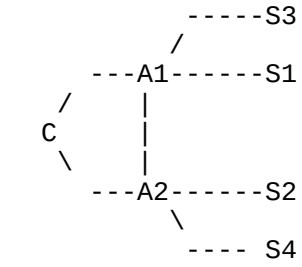

Figure 9: Linked Agents

Figure 10 is a variant of Figure 9. 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.

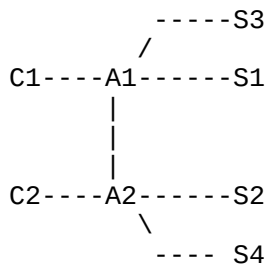

Figure 10: Linked Agents

A.5. Shared Server Pools

Figure 11 is similar to Figure 9, 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.

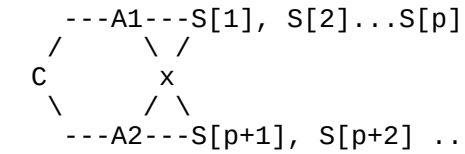

Figure 11: Shared Server Pools

A.6. Agent Chains

The scenario in Figure 12 is similar to that of Figure 8, 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.

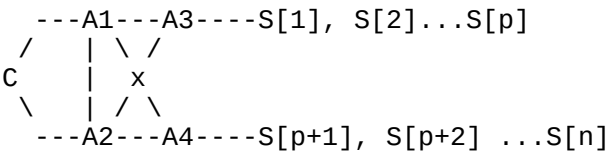

Figure 12: Agent Chains

A.7. Fully Meshed Layers

Figure 13 extends the scenario in Figure 11 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.

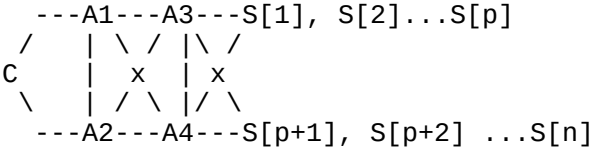

Figure 13: Full Mesh

A.8. Partitions

A Diameter network with multiple servers is said to be "partitioned" when only a subset of available servers can serve 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.

A.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.

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
Nokia
Route de Villejust
91620 Nozay
France

Email: jean-jacques.trottin@nokia.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: February 20, 2016

J. Korhonen, Ed.
Broadcom
S. Donovan, Ed.
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).

Status of This Memo

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

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

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

This Internet-Draft will expire on February 20, 2016.

Copyright Notice

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.

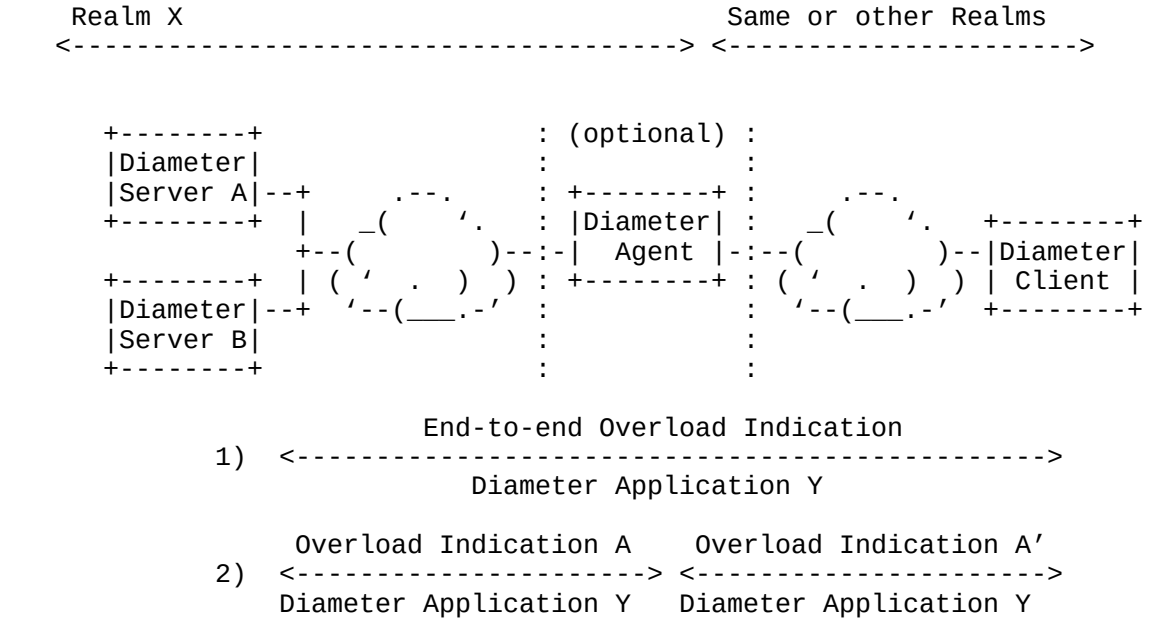

Figure 1: Simplified architecture choices for overload indication delivery

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:

- A host-type OCS entry for each Destination-Host to which it sends host-type requests and
- 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):

- Sequence number (as received in OC-OLR, see Section 7.3)
- 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)
- Selected Abatement Algorithm (as received in the OC-Supported-Features AVP)
- 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):

- Sequence number
- Validity Duration
- Expiration Time
- 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:

- Eric McMurry
- Hannes Tschofenig
- Ulrich Wiehe
- Jean-Jacques Trottin
- Maria Cruz Bartolome
- Martin Dolly
- Nirav Salot
- 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

DIME
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2016

J. Korhonen
Broadcom Ltd.
H. Tschofenig
ARM Ltd.

February 29, 2016

Diameter AVP Level Security: Keyed Message Digests, Digital Signatures, and Encryption
draft-korhonen-dime-e2e-security-03.txt

Abstract

This document defines an extension for end to end authentication, integrity and confidentiality protection of Diameter Attribute Value Pairs. The solutions focuses on protecting Diameter Attribute Value Pairs and leaves the key distribution solution to a separate specification. The integrity protection can be introduced in a backward compatible manner to existing application. The confidentiality protection requires an explicit support from an application, thus is applicable only for newly defined applications.

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].

Status of This Memo

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

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

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

This Internet-Draft will expire on September 1, 2016.

Table of Contents

1. Introduction

The Diameter base protocol [RFC6733] leverages IPsec and TLS for mutual authentication between neighboring Diameter nodes and for channel security offering data origin authentication, integrity and confidentiality protection. The Diameter base protocol, however, also defines Diameter agents, namely Relay Agents, Proxy Agents, Redirect Agents, and Translation Agents.

Relay Agents are Diameter agents that accept requests and route messages to other Diameter nodes based on information found in the messages. Since Relays do not perform any application level processing, they provide relaying services for all Diameter applications.

Similarly to Relays, Proxy Agents route Diameter messages using the Diameter routing table. However, they differ since they modify messages to implement policy enforcement.

Redirect Agents do not relay messages, and only return an answer with the information necessary for Diameter agents to communicate directly, they do not modify messages. Redirect Agents do not have negative impacts on end-to-end security and are therefore not considered in this document.

A Translation Agent is a device that provides translation between two protocols. To offer end-to-end security across different protocol requires the ability to convey and process the AVPs defined in this document by both end points. Since such support is very likely not available this document does not cover this functionality.

The Diameter extension defined in this document specifies how AVP authentication, integrity and confidentiality protection can be offered using either symmetric or asymmetric cryptography. As a solution mechanism is derived form Javascript Object Signing and Encryption (JOSE). JOSE offers a simple encoding with small set of features ideal for the purpose of Diameter. This document further defines a binary efficient coding of JOSE objects.

This document focuses on protecting Diameter AVP and leaves the key distribution solution to a separate specification, which most likely is going to be a specific key exchange application. To offer the functionality two grouped AVPs are defined: Signed-Data and Encrypted-Data. The respective JOSE objects are transported within these two AVPs.

2. Solution description

2.1. Integrity protection of AVPs

JWS represents digitally signed or HMACed content using JSON data structures. The representation in [I-D.ietf-jose-json-web-signature] consists of three parts: the JWS Header, the JWS Payload, and the JWS Signature. The three parts are represented as the concatenation of the encoded strings in that order, with the three strings being separated by period ('.') characters. For the JWS Payload one would define a new JSON object that contains an array of AVP code number and a hash of AVP pairs. The JWS Signature then covers the all APVs to be signed or HMACed. Both JWS Payload and signature MUST use the same hash algorithm of the cryptographic algorithm indicated in the JWS Header.

Although the solution relies on the JSON, the encoding into Diameter AVPs differ from the text based encoding of the JSON objects. Specifically, none of of the JWS Header, JWS Payload or JWS Signature are not BASE64 encoded but are processed in their plaintext or binary representation formats. For example, the JWS Header is encoded in its plaintext format into the Header-Parameters AVP:

```
{  "typ":"JWT",
   "alg":"HS256",
   "kid":"abc123"
}
```

The JWS Payload and the JWS Signature hashes and AVP Code values are encoded in their binary format as octets, not in textual or BASE64 encoded formats. Sections 3.4 and 3.5 describe the encodings of the needed AVPs.

To package a set of AVPs for signing, each AVP octet representation to be protected are first individually hashed and encoded into the "JSON object" with its four octets AVP code number. The entire AVP MUST be input to the hash calculation, from the first byte of the AVP code to the last byte of the AVP data, including all other fields, length, reserved/flags, and optional vendor IDs, and padding. The AVP MUST be input to the hash calculation in network byte order.

The JWS Signature is calculated over the entire JWS Payloads and then the all three JWS parts are placed in the Signed-Data AVP. There can be multiple Signed-Data AVPs in a Diameter message. The AVP code in the JWS Payload is to indicate which AVP this hash possibly refers to. If there are multiple instances of the same AVP in the Diameter message, there is no other way than make the verification against all of those. It is possible that the message sender only hashed one AVP

of the same type and, therefore, the receiver MUST verify the hash against all occurrences of the AVP of the same code number. Such flexibility is added there to allow reordering of the AVPs and addition or deletion of new AVPs by intermediating agents.

If a receiver detects errors with the processing of the Signed-Data AVP it MAY return one of the errors defined in Section 4. If a receiver does not find any AVP the Signed-Data AVP has a signature for, it MAY also return one of the errors defined in Section 4.

When AVPs are to be both encrypted and signed, the Encrypted-Data AVP MUST be created first. This means that signing is "outside" encryption.

Here is an example: Imagine the following AVPs from the QoS-Resources AVP in the QoS-Install Request (defined in RFC 5866 [RFC5866] message shall be signed. The resulting example message has the following structure:

```
<QoS-Install-Request> ::= < Diameter Header: 327, REQ, PXY >
                          < Session-Id >
                          { Auth-Application-Id }
                          { Origin-Host }
                          { Origin-Realm }
                          { Destination-Realm }
                          { Auth-Request-Type }
                          [ Signed-Data ]
                        * [ QoS-Resources ]
                          ...
```

Example Diameter Message with Signed-Data AVP

The Signed-Data AVP in this example may contain a JWS Header that indicates the use of the HMAC SHA-256 algorithm with the key id 'abc123'. The protected AVPs are Session-Id, Origin-Host and Origin-Realm. The calculated HMAC SHA-256 values are for example purposes only (i.e., are not real):

JWS Header encoded as such in JWS-Header AVP:

```
 {"typ":"JWT",
  "alg":"HS256",
  "kid":"abc123"
 }
```

```
0x00000xxx  // JWS-Header code 'xxx'
0x00000034  // Flags=0, Length=52
'{"typ":"JWT","alg":"HS256","kid":"abc123"}'  // 41
0x00,0x00,0x00  // 3 octets padding
```

JWS Payload encoded into three JWS-AVP-Payload AVPs:

```
0x00000zzz  // JWS-AVP-Payload code 'zzz'    <--+
0x0000001c  // Flags=0, Length=28               |
0x00000107  // 263, Session-Id, 4 octets        s
0x9d0e0495  // hash of Session-Id, 128 bits     i
0xba8c0312                                      g
0xb6274c52                                      n
0x7d51a048                                      a
                                                t
0x00000zzz  // JWS-AVP-Payload code 'zzz'        u
0x0000001c  // Flags=0, Length=28               r
0x00000108  // 264, Origin-Host, 4 octets       e
0x39ca88ff  // hash of Origin-Host, 128 bits    |
0xaa5a6ff9                                      c
0x029ed95b                                      o
0xa534e028                                      v
                                                e
0x00000zzz  // JWS-AVP-Payload code 'zzz'        r
0x0000001c  // Flags=0, Length=28               a
0x00000128  // 296, Origin-Realm, 4 octets      g
0x202730ac  // hash of Origin-Realm, 128 bits   e
0xa6e3a180                                      |
0x2f44a633                                      |
0xf250f6fe                                   <--+
```

JWS Signature encoded into the JWS-Signature AVP:

```
0x00000yyy  // JWS-Signature code 'yyy'
0x00000018  // Flags=0, Length=24
0xaabbccdd,0xddeeff00,0x11223344,0x55667788
```

Example JWS Header, Payload and Signature

2.2. Confidentiality protection of AVPs

The Encrypted-Data AVP (AVP Code TBD) is of type OctetString and contains the JSON Web Encryption (JWE) [I-D.ietf-jose-json-web-encryption] data structure and consists of four parts: the JWE Header, the JWE Encrypted Key, the JWE Initialization Vector and the JWE Ciphertext. The four parts are represented as the concatenation of the encoded strings in that order, with the three strings being separated by period ('.') characters. JWE does not add a content integrity check if not provided by the underlying encryption algorithm.

Although the solution relies on the JSON, the encoding into Diameter AVPs differ from the text based encoding of the JSON objects. Specifically, none of of the the JWE Header, the JWE Encrypted Key, the JWE Initialization Vector and the JWE Ciphertext are not BASE64 encoded but are processed in their plaintext or binary representation formats. The concept follows what was already described in Section 2.1.

A single AVP or an entire list of AVPs MUST be input to the encryption process, from the first byte of the AVP code to the last byte of the AVP data, including all other fields, length, reserved/ flags, and optional vendor IDs, and padding. The AVP MUST be input to the encryption process in network byte order, and the encryptor is free to order AVPs whatever way it chooses. When AVPs are to be both encrypted and authenticated, the Encrypted-Data AVP MUST be created first.

Note that the usage of the Encrypted-Data AVP requires explicit support by the Diameter application since a receiving Diameter node must first decrypt the content of the Encrypted-Data AVP in order to evaluate the AVPs carried in the message. In case that a Diameter node is unable to understand the Encrypted-Data AVP and ignores the AVP then two possible outcomes are possible: First, if the encrypted AVPs are optional then their content is not considered by the receiving Diameter server without any indication to the sender that they have not been processes. Worse, in the second case when the encrypted AVPs are mandatory to be processed then the receiving Diameter node will return an error that may not inform the sender about the failure to decrypt the Encrypted-Data AVP. Consequently, the usage of the Encrypted-Data AVP may require changes to the ABNF definition of a Diameter application.

If a receiver detects that the contents of the Encrypted-Data AVP is invalid, it SHOULD return the new Result-Code AVP value defined in Section 4.

2.3. Definition of the 'End Point'

Although this specification claims to introduce the end-to-end security into Diameter, the definition who actually is the 'end point' is not obvious. The 'end point' does not need to be the original Diameter request or answer originator but the Diameter node that inserts the Signed-Data or the Encrypted-Data AVPs into the Diameter message. The node can be the request or answer originator or a proxy agent. Use of proxy agents doing the 'end-to-end' security on behalf of other nodes mimics the deployments where site-to-site VPNs are used.

3. AVP Encoding

3.1. Signed-Data AVP

The Signed-Data AVP (AVP Code TBD1) is of type Grouped and utilizes the JSON Web signature (JWS) mechanism defined in [I-D.ietf-jose-json-web-signature]. The JWS payload is then encoded into the Signed-Data AVP:

```
Signed-Data  ::= < AVP Header: TBD1 >
                   { JWS-Header }
                 * { JWS-AVP-Payload }
                   { JWS-Signature }
                 * [ AVP ]
```

3.2. JWS-Header AVP

The JWS-Header AVP (AVP Code TBD2) is of type UTF8String and contains the JSON Web Signature Header. The contents of the AVP follow the rules for the header found in [I-D.ietf-jose-json-web-signature], which implies the required IANA registries are also defined by JSON documents.

```
JWS-Header  ::= < AVP Header: TBD2 >
                  { Header-Parameters }
                * [ AVP ]
```

The "alg" is the only REQUIRED Header Parameter for the signature purposes. The "typ" and "kid" Header Parameters are also RECOMMENDED.

3.3. Header-Parameters AVP

The Header-Parameters AVP (AVP Code TBD3) is of type UTF8String and contains the JSON Header Parameter Name and its value as described in [I-D.ietf-jose-json-web-signature]. The encoding (textual) also

follows [I-D.ietf-jose-json-web-signature]. Differing from the JSON specifications the parameter names and values are not BASE64 encoded but in their original UTF-8 representation format.

3.4. JWS-AVP-Payload AVP

The JWS-AVP-Payload AVP (AVP Code TBD4) is of type OctetString and contains both an AVP Code and a hash of the entire AVP identified by the AVP Code. The first four octets contain the AVP Code in a network byte order followed by the hash octets. The length of the hash octets depends on the used hash algorithm.

3.5. JWS-Signature AVP

The JWS-Signature AVP (AVP Code TBD5) is of type OctetString and contains the signature calculated over the array of complete JWS-AVP-Payload AVPs (including AVP header fields etc) in the order they appear in the Signed-Data AVP. The length of the signature octets depends on the used signature algorithm.

3.6. Encrypted-Data AVP

The Encrypted-Data AVP (AVP Code TBD6) is of type Grouped and utilizes the JSON Web Encryption (JWE) mechanism defined in [I-D.ietf-jose-json-web-encryption]. The JWE payload is then encoded into the Encrypted-Data AVP:

```
Encrypted-Data  ::= < AVP Header: TBD1 >
                    { JWE-Header }
                    { JWE-Enc-Key }
                    [ JWE-Init-Vec ]
                    { JWE-AVP-Ciphertext }
                  * [ AVP ]
```

3.7. JWE-Header AVP

The JWE-Header AVP (AVP Code TBD7) is of type UTF8String and contains the JSON Web Encryption Header. The contents of the AVP follow the rules for the header found in [I-D.ietf-jose-json-web-encryption], which implies the required IANA registries are also defined by JSON documents.

```
JWE-Header  ::= < AVP Header: TBD7 >
                { Header-Parameters }
              * [ AVP ]
```

The "alg" and "enc" are the REQUIRED Header Parameter for the encryption purposes. The "typ" and "kid" Header Parameters are also RECOMMENDED.

3.8. JWE-Enc-Key AVP

The JWE-Enc-Key AVP (AVP Code TBD8) is of type OctetString and contains the JWE Encrypted Key in its binary format.

3.9. JWE-Init-Vec AVP

The JWE-Init-Vec AVP (AVP Code TBD9) is of type OctetString and contains the JWE Initialization Vector in its binary format.

3.10. JWE-AVP-Ciphertext AVP

The JWE-AVP-Ciphertext AVP (AVP Code TBD10) is of type OctetString and contains the encrypted AVPs. The encrypted AVPs are first concatenated into one large plaintext octet blob and then encrypted as a whole. The length of the ciphertext depends on the used algorithm and encrypted AVPs. The plaintext to be encrypted is never BASE64 encoded but MAY be compressed if a "zip" parameter was included in the JWE Header.

4. Result-Code AVP Values

This section defines new Diameter result code values for usage with Diameter applications.

4.1. Transient Failures

Errors that fall within the transient failures category are used to inform a peer that the request could not be satisfied at the time it was received, but MAY be able to satisfy the request in the future.

DIAMETER_KEY_UNKNOWN (TBD11)

> This error code is returned when a Signed-Data or an Encrypted-Data AVP is received that was generated using a key that cannot be found in the key store. This error may, for example, be caused if one of the endpoints of an end-to-end security association lost a previously agreed upon key, perhaps as a result of a reboot. To recover a new end-to-end key establishment procedure may need to be invoked.

DIAMETER_HEADER_NAME_ERROR (TBD12)

This error code is returned when a Header Parameter Name is not understood in the JWS-Header AVP or in the JWE-Header AVP.

4.2. Permanent Failures

Errors that fall within the permanent failures category are used to inform the peer that the request failed, and should not be attempted again.

DIAMETER_DECRYPTION_ERROR (TBD13)

This error code is returned when an Encrypted-Data AVP is received and the decryption fails for an unknown reason.

DIAMETER_SIGNATURE_ERROR (TBD14)

This error code is returned when a Signed-Data AVP is received and the verification fails for an unknown reason.

5. IANA Considerations

IANA is requested to allocate AVP codes for the following AVPs:

AVP Name	AVP Code	Section Defined	Data Type
Signed-Data	TBD1	3.1	Grouped
JWS-Header	TBD2	3.x	Grouped
JWS-AVP-Paylaod	TBD3	3.x	OctetString
JSW-Signature	TBD4	3.x	OctetString
Header-Parameters	TBD5	3.x	UTF8String
Encrypted-Data	TBD6	3.x	Grouped
JWE-Header	TBD7	3.x	Grouped
JWE-Enc-Key	TBD8	3.x	OctetString
JWE-Init-Vec	TBD9	3.x	OctetString
JWE-AVP-Ciphertext	TBD10	3.x	OctetString

This specification additionally defines a few Result-Code AVP values, see Section 4.

6. Security Considerations

The purpose of this document is to offer end-to-end security mechanisms for calculating keyed message digest, for signing, and for encryption of Diameter AVPs.

An intermediate Diameter agent that for a reason or other reorders the AVPs within the Signed-Data AVP may cause the signature verification fail even if no AVP was actually tampered.

7. Acknowledgements

We would like to thank the authors of [I-D.ietf-aaa-diameter-e2e-sec] for their work on CMS end-to-end security for Diameter. Their document inspired us.

8. References

8.1. Normative References

[I-D.ietf-jose-json-web-encryption]
Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption-40 (work in progress), January 2015.

[I-D.ietf-jose-json-web-signature]
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", draft-ietf-jose-json-web-signature-41 (work in progress), January 2015.

[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>.

[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>.

8.2. Informational References

[I-D.ietf-aaa-diameter-e2e-sec]
Calhoun, P., "Diameter End-2-End Security Extension", 2001.

[RFC5866] Sun, D., Ed., McCann, P., Tschofenig, H., Tsou, T., Doria, A., and G. Zorn, Ed., "Diameter Quality-of-Service Application", RFC 5866, DOI 10.17487/RFC5866, May 2010, <http://www.rfc-editor.org/info/rfc5866>.

Authors' Addresses

Jouni Korhonen
Broadcom Ltd.
3151 Zanker Road
CA 95134
US

Email: jouni.nospam@gmail.com

Hannes Tschofenig
ARM Ltd.

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