Diameter Maintenance and Extensions (DIME) S. Donovan
Internet-Draft Oracle
Intended status: Standards Track May 26, 2015
Expires: November 27, 2015

Diameter Routing Message Priority
draft-donovan-dime-drmp-01.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 defines 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.

Table of Contents

1. Introduction

The DOIC solution [I-D.ietf-dime-ovli] 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 clean 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 [I-D.ietf-dime-ovli]. 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 [I-D.ietf-dime-ovli]. 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 higher priority value implies a higher relative importance of the message.

Throttling

As defined in [I-D.ietf-dime-ovli]. 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 discussed 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 job.

The United States Wireless Priority Services (WPS) and Government Emergency Telecommunications Service (GETS) are examples of systems designed to address 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 restoration procedure might be given a higher priority than a ULR 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 handing 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. The agent also records the transaction priority in the transaction state. 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.

5. Agent handling the answer - Agents handling answer messages use the priority information stored with the transaction state to determine the priority of relaying the answer message. This implies that answers for higher priority transactions are given preferential treatment to lower priority transactions.

6. Answer handler - The handler of the answer message uses the priority of the transaction when allocating resources for processing that occurs after the receipt of the answer message.

7. Normative Behavior

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

When routing priority information is availble, Diameter nodes SHOULD include Diameter routing message priority 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 SHOULD NOT require the Diameter Agents to understand application specific AVPs.

When routing priority information is availble, Diameter nodes SHOULD use DRMP information when making Diameter overload related throttling decisions.

Diameter agents MAY use DRMP information when relaying messages. This includes the selection of routes and the ordering of messages relayed.

> The priority information included applies to both the request message and answer message associated with the transaction. As such it is used in the processing of both types of messages.

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

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

When there is a mix of transactions specifying priority in request messages and transactions that do not have the priority specified, transactions that do not have a specified priority SHOULD be treated as having the PRIORITY_0 priority.

When setting and using priorities, PRIORITY_0 MUST be treated as the lowest priority.

When setting and using priorities, PRIORITY_1 MUST be treated as a higher priority than PRIORITY_0 and a lower priority than PRIORITY_2.

When setting and using priorities, PRIORITY_2 MUST be treated as a higher priority than PRIORITY_1 and a lower priority than PRIORITY_3.

When setting and using priorities, PRIORITY_3 MUST be treated as a higher priority than PRIORITY_2 and a lower priority than PRIORITY_4.

When setting and using priorities, PRIORITY_4 MUST be the highest priority.

> Editor’s note: It is likely that there are other considerations for setting and using priorities. For instance, it might be good to use priority 1 to indicate elevated priority for strictly protocol reasons (e.g.; the S6a use case). Priorities 3, 4 and 5 would then be used for non protocol reasons.

8. Attribute Value Pairs

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

8.1. 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_0 0 Priority 0 is the lowest priority.

PRIORITY_1 1 Priority 1 is the second lowest priority.

PRIORITY_2 2 Priority 2 is the middle priority.

PRIORITY_3 3 Priority 3 is the second higest priority.

PRIORITY_4 4 Priority 4 is the higest priority.

8.2. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
DRMP	TBD1	8.1	Grouped		V

9. IANA Considerations

9.1. AVP codes

New AVPs defined by this specification are listed in Section 8. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

9.2. New registries

There are no new IANA registries introduced by this document.

> Editor's Note: The current assumption is that there is no need to extend the number of priorities beyond the five defined in this specification. This assumption needs to be verified. If there is the need for extensability then a new IANA registry would be required. This new registry would be established as part of the standardization effort associated with the definition of new priority values.

10. Security Considerations

The DRMP could be used to get better access to services. This could result in one segment of a Diameter network limiting service to another segment of a Diameter network.

11. Contributors

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

- Janet P. Gunn

12. References

12.1. Normative References

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

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

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

12.2. Informative References

[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.

[RFC4412] Schulzrinne, H. and J. Polk, "Communications Resource Priority for the Session Initiation Protocol (SIP)", RFC 4412, February 2006.

[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.

Appendix A. Design Considerations and Questions

This section contains a list of questions that will influence the design of the DRMP mechanism. It is expected that this section will be removed once the DRMP mechanism is defined.

A.1. Relationship with SIP Resource Priority

Question 1: Is there value with aligning the Diameter Routing Message Priority design with the SIP Resource Priority [RFC4412]work?

Current thoughts: SIP Resource Priority is considered to be addressing a superset of the requirements that DRMP addresses. The concensus seems to be that there is no need for multiple name spaces with DRMP.

Question 2: If so, is there value in reusing the existing SIP Resource Priority name spaces and request handling strategies?

Current thoughts: Given that the direction for DRMP is to have a single set of priority values, DRMP will not reuse name spaces.

A.2. Priority Encoding Method

Question 3: Is there a preference for handling DRMP by introducing AVPs or by using existing bits in the Diameter Command Flags field?

Current thoughts: The advantage of using bits in the Command Flags field is that it would reduce parsing overhead for elements that need access to the routing priority information. The question is whether this optimization in parsing overhead is worth the expense of using the reserved bits.

There are four bits remaining in the Command Flags header. If this approach is taken then the expectation would be that three of the bits would be used, allowing for eight priority levels.

This approach has questionable utility if multiple namespaces are to be used as the namespace identity would still require an AVP. Once the requirement for parsing the namespace AVP is introduced the incremental savings from utilizing the Command Flags would be minimal.

The current direction is to use AVPs to communicate priority. This gives the ability to extend the DRMP mechanism if additional functionality, such as name spaces, is determined to be required.

A.3. Base Protocol versus Application Extension

Question 4: Should DRMP be base protocol behavior or should Diameter applications be required to explicitly incorporate DRMP behavior?

The direction is to make the behavior generic across all applications.

A.4. Scope of Priority Setting

Question 5: Which of the following does the DRMP priority apply to:

Messages - meaning that a separate priority can be set for request messages and answer messages?

Transactions - meaning that the priority set in the request message also applies to the answer messages?

Request messages - meaning that answer message priority always has an implied higher priority than all request messages?

Current thoughts: The consensus is to have the DRMP priority apply to transactions.

Author's Address

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

Email: srdonovan@usdonovans.com

Internet Engineering Task Force C. Zhou
Internet-Draft Huawei Technologies
Intended status: Standards Track T. Taylor
Expires: December 25, 2015 PT Taylor Consulting
Q. Sun
China Telecom
M. Boucadair
France Telecom
June 23, 2015

Attribute-Value Pairs For Provisioning Customer Equipment Supporting IPv4-Over-IPv6 Transitional Solutions

draft-ietf-dime-4over6-provisioning-03

Abstract

During the transition from IPv4 to IPv6, customer equipment may have to support one of the various transition methods that have been defined for carrying IPv4 packets over IPv6. This document enumerates the information that needs to be provisioned on a customer edge router to support a list of transition techniques based on tunneling IPv4 in IPv6, with a view to defining reusable components for a reasonable transition path between these techniques. To the extent that the provisioning is done dynamically, AAA support is needed to provide the information to the network server responsible for passing the information to the customer equipment. This document specifies Diameter (RFC 6733) attribute-value pairs to be used for that purpose.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A number of transition techniques have been defined to allow IPv4 packets to pass between hosts and IPv4 networks over an intervening IPv6 network while minimizing the number of public IPv4 addresses that need to be consumed by the hosts. Different operators will deploy different technologies, and sometimes one operator will use more than one technology, depending on what is supported by the available equipment and upon other factors both technical and economic.

Each technique requires the provisioning of some subscriber-specific information on the customer edge device. The provisioning may be by DHCPv6 [RFC3315] or by some other method. This document is indifferent to the specific provisioning technique used, but assumes a deployment in which that information is managed by AAA (Authentication, Authorization, and Accounting) servers. It further assumes that this information is delivered to intermediate network nodes for onward provisioning using the Diameter protocol [RFC6733].

As described below, in the particular case where the Lightweight IPv4 Over IPv6 (Lw4o6) [I-D.ietf-softwire-lw4over6] transition method has been deployed, per-subscriber-site information almost identical to that passed to the subscriber site [I-D.ietf-softwire-map-dhcp] also needs to be delivered to the border router serving that site. The Diameter protocol may be used for this purpose too.

This document analyzes the information required to configure the customer edge equipment for the following set of transition methods:

- o Dual-Stack Lite (DS-Lite) [RFC6333],
- o Lightweight IPv4 Over IPv6 (LW4over6) [I-D.ietf-softwire-lw4over6], and
- o Mapping of Address and Port with Encapsulation (MAP-E) [I-D.ietf-softwire-map].

[I-D.ietf-softwire-dslite-multicast] specifies a generic solution for delivery of IPv4 multicast services to IPv4 clients over an IPv6 multicast network. The solution was developed with DS-Lite in mind but it is however not limited to DS-Lite. As such, it applies also for LW4over6 and MAP-E. This document analyzes the information required to configure the customer edge equipment for the support of multicast in the context of DS-Lite, MAP, and LW4over6 in particular.

On the basis of those analyses it specifies a number of attribute-value pairs (AVPs) to allow the necessary subscriber-site-specific configuration information to be carried in Diameter.

This document doesn’t specify any new commands or Application-Ids and that the AVPs could be used for any Diameter application suitable for provisioning.

1.1. Requirements Language

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

The abbreviation "CE" denotes the equipment at the customer edge that terminates the customer end of an IPv6 transitional tunnel. This will usually be a router, but could be a host directly connected to the network.

The term "tunnel source address" is used to denote the IPv6 source address used in the outer header of packets sent from the CE through an LW4over6 transitional tunnel to the border router.

2. Description of the Parameters Required By Each Transition Method

This section reviews the parameters that need to be provisioned for each of the transition methods listed above. This enumeration provides the justification for the AVPs defined in the next section.

A means is required to indicate which transition method(s) a given subscriber is allowed to use. The approach taken in this document is to specify Grouped AVPs specific to LW4over6 and MAP-E. The operator can control which of these two transition methods a given subscriber uses by ensuring that AAA passes only the Grouped AVP relevant to that method. A Grouped AVP is unnecessary for Dual-Stack Lite, since AAA has only to provide the Fully Qualified Domain Name (FQDN) of the DS-Lite Address Family Transition Router (AFTR) (see Section 2.1). Hence when no Grouped AVP is provided either for LW4over6 or MAP-E and only the AFTR’s FQDN is present, this indicates that the subscriber equipment will use the Dual-Stack Lite transition method. Provisioning of multicast is an orthogonal activity, since it is independent of the transition method.

2.1. Parameters For Dual-Stack Lite (DS-Lite)

DS-Lite is documented in [RFC6333]. The Basic Bridging BroadBand (B4) element at the customer premises needs to discover the IPv6 address of the AFTR (border router). For the reasons discussed in Section 3.2, the AAA server provision the B4 element with the AFTR's Fully Qualified Domain Name (FQDN) that is passed to a B4's IP resolution library. The AFTR's FQDN is contained in the Border-Router-Name AVP (see Section 3.2).

The B4 element could also be configured with the IPv4 address of the B4 interface facing the tunnel, with valid values from 192.0.0.2 to 192.0.0.7 and the default value of 192.0.0.2 in the absence of provisioning. Provisioning such information through AAA is problematic because it is most likely used in a case where multiple B4 instances occupy the same device. This document therefore assumes that the B4 interface address is determined by other means than AAA (implementation-dependent or static assignment).

2.2. Lightweight IPv4 Over IPv6 (LW4over6)

Light Weight IPv4 Over IPv6 (LW4over6) is documented in [I-D.ietf-softwire-lw4over6]. LW4over6 requires four items to be provisioned to the customer equipment:

- IPv6 address of the border router.

- IPv6 prefix used by the CE to construct the tunnel source address. In the terminology of [I-D.ietf-softwire-lw4over6], this is the IPv6 Binding Prefix.

- an IPv4 address to be used on the external side of the CE; and

- if the IPv4 address is shared, a specification of the port set the subscriber site is allowed to use. Please see the description in Section 2.3. For LW4over6, all three of the parameters 'a', 'k', and PSID described in that section are required. The default value of the offset parameter 'a' is 0.

As discussed in Section 4 of [I-D.ietf-softwire-lw4over6], it is necessary to synchronize this configuration with corresponding per-subscriber configuration at the border router. The border router information consists of the same public IPv4 address and port set parameters that are passed to the CE, bound together with the full /128 IPv6 address (not just the Binding Prefix) configured as the tunnel source address at the CE.

2.3. Port Set Specification

When an external IPv4 address is shared, LW4over6 and MAP-E restrict the CE to use of a subset of all available ports on the external side. Both transition methods use the algorithm defined in Appendix B of [I-D.ietf-softwire-map] to derive the values of the port numbers in the port set. This algorithm features three parameters, describing the positioning and value of the Port Set Identifier (PSID) within each port number of the generated set:

- an offset 'a' from the beginning of the port number to the first bit of the PSID;

- the length 'k' of the PSID within the port number, in bits; and

- the value of the PSID itself.

2.4. Mapping of Address and Port with Encapsulation (MAP-E)

Mapping of Address and Port with Encapsulation (MAP-E) is described in [I-D.ietf-softwire-map]. MAP-E requires the provisioning of the following per-subscriber information at the customer edge device:

- the IPv6 address of one or more border routers, or in MAP-E terminology, MAP border relays.

- the unique End-user IPv6 prefix for the customer edge device. This may be provided by AAA or acquired by other means.

- the Basic Mapping Rule for the customer edge device. This includes the following parameters:

  * the rule IPv6 prefix and length;

  * the rule IPv4 prefix and length. A prefix length of 0 indicates that the entire IPv4 address or prefix is coded in the Extended Address (EA) bits of the End-user IPv6 prefix rather than in the mapping rule.

  * the number of EA bits included in the End-user IPv6 prefix;

  * port set parameters giving the set of ports the CE is allowed to use when the IPv4 address is shared. Please see the description of these parameters in Section 2.3. At a minimum, the offset parameter 'a' is required. For MAP-E this has the default value 6. The parameters 'k' and PSID are needed if they cannot be derived from the mapping rule information and

the EA bits (final case of Section 5.2 of [I-D.ietf-softwire-map]).

- o whether the device is to operate in mesh or hub-and-spoke mode;

- o in mesh mode only, zero or more Forwarding Mapping Rules, described by the same set of parameters as the Basic Mapping Rule;

As indicated in Section 5, bullet 1 of the MAP-E document, a MAP CE can be provisioned with multiple End-user IPv6 prefixes, each associated with its own Basic Mapping Rule. This does not change the basic requirement for representation of the corresponding information in the form of Diameter AVPs, but adds a potential requirement for multiple instances of this information to be present in the Diameter message, differing in the value of the End-user IPv6 prefix (in contrast to the Forward Mapping Rule instances).

The border router needs to be configured with the superset of the Mapping Rules passed to the customer sites it serves. Since this requirement does not require direct coordination with CE configuration in the way LW4over6 does, it is out of scope of the present document. However, the AVPs defined here may be useful if a separate Diameter application is used to configure the border router.

2.5. Parameters For Multicast

[I-D.ietf-softwire-dslite-multicast] specifies a generic solution for delivery of IPv4 multicast services to IPv4 clients over an IPv6 multicast network. The solution can be in particular deployed in a DS-Lite context, but is also adaptable to LW4over6 and MAP-E. For example, [I-D.ietf-softwire-multicast-prefix-option] specifies how DHCPv6 [RFC3315] can be used to provision multicast-related information. The following lists the multicast-related information that needs to be provisioned:

- o ASM-mPrefix64: the IPv6 multicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast groups in the Any-Source Multicast (ASM) mode. This is achieved by concatenating the ASM-mPrefix64 and a IPv4 multicast address; the Pv4 multicast address is inserted in the last 32 bits of the IPv4-embedded IPv6 multicast address.

- o SSM-mPrefix64: the IPv6 multicast prefix to be used to synthesize the IPv4-embedded IPv6 addresses of the multicast groups in the Source-Specific Multicast (SSM, [RFC4607]) mode. This is achieved by concatenating the SSM-mPrefix64 and a IPv4 multicast address; the Pv4 multicast address is inserted in the last 32 bits of the IPv4-embedded IPv6 multicast address.

- o uPrefix64: the IPv6 unicast prefix to be used in SSM mode for constructing the IPv4-embedded IPv6 addresses representing the IPv4 multicast sources in the IPv6 domain. uPrefix64 may also be used to extract the IPv4 address from the received multicast data flows. The address mapping follows the guidelines documented in [RFC6052].

2.6. Summary and Discussion

It appears that two items are common to the different transition methods and the corresponding AVPs to carry them can be reused:

- o a representation of the IPv6 address of a border router;

- o A set of prefixes for delivery of multicast services to IPv4 clients over an IPv6 multicast network.

[RFC6519] sets a precedent for representation of the IPv6 address of a border router as an FQDN. This can be dereferenced to one or more IP addresses by the provisioning system before being passed to the customer equipment, or left as an FQDN as it as in [RFC6334].

The remaining requirements are transition-method-specific:

- o for LW4over6, a representation of a binding between (1) either the IPv6 Binding Prefix or a full /128 IPv6 address, (2) a public IPv4 address, and (3) (if the IPv4 address is shared) a port set identifier;

- o for MAP-E, a representation of the unique End-user IPv6 prefix for the CE, if not provided by other means;

- o for MAP-E, a representation of a Mapping Rule;

- o for MAP-E, an indication of whether mesh mode or hub-and-spoke mode is to be used.

3. Attribute-Value Pair Definitions

This section provides the specifications for the AVPs needed to meet the requirements summarized in Section 2.6. Within the context of their usage, all of these AVPs MUST have the M bit set and the V bit cleared.

3.1. IP-Prefix-Length AVP

The IP-Prefix-Length AVP (AVP code TBD00) is of type Unsigned32. It provides the length of an IPv4 or IPv6 prefix. Valid values are from 0 to 32 for IPv4, and from 0 to 128 for IPv6. Tighter limits are given below for particular contexts of use of this AVP.

NOTE: The IP-Prefix-Length AVP is only relevant when associated with an IP-Address AVP in a Grouped AVP.

3.2. Border-Router-Name AVP

Following on the precedent set by [RFC6334] and [RFC6519], this document identifies a border router using an FQDN rather than an address. The Border-Router-Name AVP (AVP Code TBD01) is of type OctetString. The FQDN encoding MUST follow the Name Syntax defined in [RFC1035][RFC1123][RFC2181] and are represented in ASCII form.

3.3. 64-Multicast-Attributes AVP

The 64-Multicast-Attributes AVP (AVP Code TBD02) is of type Grouped. It contains the multicast-related IPv6 prefixes needed for providing IPv4 multicast over IPv6 using DS-Lite, MAP-E, or LW4over6, as mentioned in Section 2.5.

The syntax is shown in Figure 1.

```
        64-Multicast-Attributes  ::= < AVP Header: TBD02 >
                           [ ASM-mPrefix64 ]
                           [ SSM-mPrefix64 ]
                           [ Delegated-IPv6-Prefix ]
                          *[ AVP ]
```

Figure 1: 64-Multicast-Attributes AVP

64-Multicast-Attributes AVP MUST include at least the ASM-mPrefix64 AVP or the SSM-mPrefix64 AVP.

Both the ASM-mPrefix64 AVP and the SSM-mPrefix64 AVP MAY be present.

The Delegated-IPv6-Prefix AVP MUST be present when the SSM-mPrefix64 AVP is present. The Delegated-IPv6-Prefix AVP MAY be present when the ASM-mPrefix64 AVP is present.

3.3.1. ASM-mPrefix64 AVP

The ASM-mPrefix64 AVP (AVP Code TBD03) conveys the value of ASM_mPrefix64 as mentioned in Section 2.5. The ASM-mPrefix64 AVP is of type Grouped, as shown in Figure 2.

```
          ASM-mPrefix64  ::= < AVP Header: TBD03 >
                             { IP-Address }
                             { IP-Prefix-Length }
                            *[ AVP ]
```

Figure 2: ASM-mPrefix64 AVP

IP-Address (AVP code 518) is defined in [RFC5777] and is of type Address. Within the ASM-mPrefix64 AVP, it provides the value of an IPv6 prefix. The AddressType field in IP-Address MUST have value 2 (IPv6). The conveyed multicast IPv6 prefix MUST belong to the ASM range. Unused bits in IP-Address beyond the actual prefix MUST be set to zeroes by the sender and ignored by the receiver.

The IP-Prefix-Length AVP (AVP code TBD00) provides the actual length of the prefix contained in the IP-Address AVP. Within the ASM-mPrefix64 AVP, valid values of the IP-Prefix-Length AVP are from 24 to 96.

3.3.2. SSM-mPrefix64 AVP

The SSM-mPrefix64 AVP (AVP Code TBD04) conveys the value of SSM_mPrefix64 as mentioned in Section 2.5. The SSM-mPrefix64 AVP is of type Grouped, as shown in Figure 3.

```
          SSM-mPrefix64  ::= < AVP Header: TBD04 >
                             { IP-Address }
                             { IP-Prefix-Length }
                            *[ AVP ]
```

Figure 3: SSM-mPrefix64 AVP

IP-Address (AVP code 518) provides the value of an IPv6 prefix. The AddressType field in IP-Address MUST have value 2 (IPv6). The conveyed multicast IPv6 prefix MUST belong to the SSM range. Unused bits in IP-Address beyond the actual prefix MUST be set to zeroes by the sender and ignored by the receiver.

The IP-Prefix-Length AVP (AVP code TBD00) provides the actual length of the prefix contained in the IP-Address AVP. With regard to prefix length, note that Section 6 of [RFC3306] requires that bits 33-95 of an SSM address in the FF3x range be set to zero, meaning that the prefix length for an SSM prefix is effectively 96. However, Section 1 of [RFC4607] suggests that the lower limit of 32 bits be preserved to allow potential future use of bits 33-95. Hence applications SHOULD accept prefix lengths between 32 and 96 inclusive.

3.3.3. Delegated-IPv6-Prefix AVP As uPrefix64

Within the 64-Multicast-Attributes AVP, the Delegated-IPv6-Prefix AVP (AVP Code 123) conveys the value of uPrefix64, a unicast IPv6 prefix, as mentioned in Section 2.5. The Delegated-IPv6-Prefix AVP is defined in [RFC4818]. As specified by [RFC6052], the value in the Prefix-Length field MUST be one of 32, 48, 56, 64 or 96.

3.4. Tunnel-Source-Pref-Or-Addr AVP

The Tunnel-Source-Pref-Or-Addr AVP (AVP Code TBD05) conveys either the IPv6 Binding Prefix or the tunnel source address on the CE, as described in Section 2.2. The Tunnel-Source-Pref-Or-Addr AVP is of type Grouped, with syntax as shown in Figure 4. The Tunnel-Source-Pref-Or-Addr AVP MUST contain either the Delegated-IPv6-Prefix AVP or the Tunnel-Source-IPv6-Address AVP, not both.

```
        Tunnel-Source-Pref-Or-Addr  ::= < AVP Header: TBD05 >
                       [ Delegated-IPv6-Prefix ]
                       [ Tunnel-Source-IPv6-Address ]
                      *[ AVP ]
```

Figure 4: Tunnel-Source-Pref-Or-Addr AVP

This AVP is defined separately from the LW4over6-Binding AVP (which includes it) to provide flexibility in the transport of the tunnel source address from the provisioning system to AAA while also supporting the provision of a complete binding to the LW4over6 border router.

3.4.1. Delegated-IPv6-Prefix As the IPv6 Binding Prefix

The Delegated-IPv6-Prefix AVP (AVP code 123) is of type Octetstring, and is defined in [RFC4818]. Within the Tunnel-Source-Pref-Or-Addr AVP, it conveys the IPv6 Binding Prefix assigned to the CE. Valid values in the Prefix-Length field are from 0 to 128 (full address), although a more restricted range is obviously more reasonable.

3.4.2. Tunnel-Source-IPv6-Address AVP

The Tunnel-Source-IPv6-Address AVP (AVP code TBD06) is of type Address. It provides the address that the CE has assigned to its end of an LW4over6 tunnel. The AddressType field in this AVP MUST be set to 2 (IPv6).

3.5. Port-Set-Identifier

The Port-Set-Identifier AVP (AVP Code TBD07) is a structured OctetString with four octets of data, hence a total AVP length of 12. The description of the structure which follows refers to refers to the parameters described in Section 2.3.

- o The first (high-order) octet is the Offset field. It is interpreted as an 8-bit unsigned integer giving the offset 'a' from the beginning of a port number to the beginning of the port set identifier (PSID) to which that port belongs. Valid values are from 0 to 15.

- o The next octet, the PSIDLength, is also interpreted as an 8-bit unsigned integer and gives the length 'k' in bits of the port set identifier (PSID). Valid values are from 0 to (16 - a). A value of 0 indicates that the PSID is not present (probable case for MAP-E, see Section 2.4), and the PSIDValue field MUST be ignored.

- o The final two octets contain the PSIDValue field. They give the value of the PSID itself, right-justified within the field. That is, the value of the PSID occupies the 'k' lowest-order bits of the PSIDValue field.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Offset     |    Length     |        PSID  Value            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.6. LW4over6-Binding AVP

The LW4over6-Binding AVP (AVP Code TBD08) is of type Grouped. It contains the elements of configuration that constitute the binding between an LW4over6 tunnel and IPv4 packets sent through that tunnel, as described in Section 2.2.

```
                    LW4over6-Binding  ::= < AVP Header: TBD08 >
                             { Tunnel-Source-Pref-Or-Addr }
                             { LW4over6-External-IPv4-Addr }
                             [ Port-Set-Identifier ]
                            *[ AVP ]
```

Figure 5

The Tunnel-Source-Pref-Or-Addr AVP is defined in Section 3.4 and provides either the Binding Prefix or the full IPv6 tunnel source address.

The LW4over6-External-IPv4-Addr AVP is defined in Section 3.6.1.

The Port-Set-Identifier AVP is defined in Section 3.5. It identifies the specific set of ports assigned to the LW4over6 tunnel, when the IPv4 address is being shared.

3.6.1. LW4over6-External-IPv4-Addr AVP

The LW4over6-External-IPv4-Addr AVP (AVP Code TBD09) uses the Address derived data format defined in Section 4.3.1 of [RFC6733]. It provides the CE's external IPv4 address within the LW4over6 tunnel associated with the given binding. The AddressType field MUST be set to 1 (IPv4), and the total length of the AVP MUST be 14 octets.

3.7. MAP-E-Attributes

The MAP-E-Attributes AVP (AVP Code TBD10) is of type Grouped. It contains the configuration data identified in Section 2.4 for all of the mapping rules (Basic and Forwarding) in a single MAP domain. Multiple instances of this AVP will be present if the CE belongs to multiple MAP domains.

```
                    MAP-E-Attributes  ::= < AVP Header: TBD06 >
                           1*{ Border-Router-Name }
                           1*{ MAP-Mapping-Rule }
                             [ MAP-Mesh-Mode ]
                             [ Delegated-IPv6-Prefix ]
                            *[ AVP ]
```

Figure 6

The Border-Router-Name AVP is defined in Section 3.2. It provides the FQDN of a MAP border relay at the edge of the MAP domain to which the containing MAP-E-Attributes AVP relates. At least one instance of this AVP MUST be present.

The MAP-Mapping-Rule AVP is defined in Section 3.9. At least one instance of this AVP MUST be present. If the MAP-E domain supports mesh mode (indicated by the presence of the MAP-Mesh-Mode AVP), additional MAP-Mapping-Rule instances MAY be present. If the MAP-E domain is operating in hub-and-spoke mode, additional MAP-Mapping-Rule instances MUST NOT be present.

The MAP-Mesh-Mode AVP is defined in Section 3.8. The absence of the mesh mode indicator attribute indicates that the CE is required to operate in hub-and-spoke mode.

The Delegated-IPv6-Prefix AVP (AVP Code 123) provides the End-user IPv6 prefix assigned to the CE for the MAP domain to which the containing MAP-E-Attributes AVP relates. The AVP is defined in [RFC4818]. Valid values of the Prefix-Length field range from 0 to 128.

The Delegated-IPv6-Prefix AVP is optional because, depending on deployment, the End-user IPv6 prefix may be provided by AAA or by other means. If multiple instances of the MAP-E-Attributes AVP containing the Delegated-IPv6-Prefix AVP are present, each instance of the latter MUST have a different value.

3.8. MAP-Mesh-Mode

The MAP-Mesh-Mode AVP (AVP Code TBD11) is of type Enumerated and indicates whether the CE has to operate in mesh mode or hub-and-spoke when using MAP-E. The following values are supported:

0 MESH

1 HUB_AND_SPOKE

The absence of the mesh mode indicator attribute indicates that the CE is required to operate in hub-and-spoke mode.

3.9. MAP-Mapping-Rule

The MAP-Mapping-Rule AVP (AVP Code TBD12) is of type Grouped, and is used only in conjunction with MAP-based transition methods. Mapping rules are required both by the MAP border relay and by the CE. The components of the MAP-Mapping-Rule AVP provide the contents of a mapping rule as described in Section 2.4.

The syntax of the MAP-Mapping-Rule AVP is as follows:

```
MAP-Mapping-Rule  ::= < AVP Header: TBD12 >
                 { Rule-IPv4-Addr-Or-Prefix }
                 { Rule-IPv6-Prefix    }
                 { EA-Field-Length     }
                 { Port-Set-Identifier }
                *[ AVP ]
```

Figure 7

The Rule-IPv4-Addr-Or-Prefix, Rule-IPv6-Prefix, EA-Field-Length, and Port-Set-Identifier AVPs MUST all be present.

The Port-Set-Identifier AVP provides information to identify the specific set of ports assigned to the CE. For more information see Section 2.4 and Section 2.3. The Port-Set-Identifier AVP is defined in Section 3.5.

3.9.1. Rule-IPv4-Addr-Or-Prefix AVP

The Rule-IPv4-Addr-Or-Prefix AVP (AVP Code TBD13) conveys the rule IPv4 prefix and length as described in Section 2.4. The Rule-IPv4-Addr-Or-Prefix AVP is of type Grouped, as shown in Figure 8.

```
Rule-IPv4-Addr-Or-Prefix  ::= < AVP Header: TBD13 >
               { IP-Address }
               { IP-Prefix-Length }
              *[ AVP ]
```

Figure 8: Rule-IPv4-Addr-Or-Prefix AVP

IP-Address (AVP code 518) is defined in [RFC5777] and is of type Address. Within the Rule-IPv4-Addr-Or-Prefix AVP, it provides the value of a unicast IPv4 address or prefix. The AddressType field in IP-Address MUST have value 1 (IPv4). Unused bits in IP-Address beyond the actual prefix MUST be set to zeroes by the sender and ignored by the receiver.

The IP-Prefix-Length AVP (AVP code TBD00) provides the actual length of the prefix contained in the IP-Address AVP. Within the Rule-IPv4-Addr-Or-Prefix AVP, valid values of the IP-Prefix-Length AVP are from 0 to 32 (full address), based on the different cases identified in Section 5.2 of [I-D.ietf-softwire-map].

3.9.2. Rule-IPv6-Prefix AVP

The Rule-IPv6-Prefix AVP (AVP Code TBD14) conveys the rule IPv6 prefix and length as described in Section 2.4. The Rule-IPv6-Prefix AVP is of type Grouped, as shown in Figure 9.

```
Rule-IPv6-Prefix  ::= < AVP Header: TBD14 >
             { IP-Address }
             { IP-Prefix-Length }
            *[ AVP ]
```

Figure 9: Rule-IPv6-Prefix AVP

IP-Address (AVP code 518) is defined in [RFC5777] and is of type Address. Within the Rule-IPv6-Prefix AVP, it provides the value of a unicast IPv6 prefix. The AddressType field in IP- Address MUST have value 2 (IPv6). Unused bits in IP-Address beyond the actual prefix MUST be set to zeroes by the sender and ignored by the receiver.

The IP-Prefix-Length AVP (AVP code TBD00) provides the actual length of the prefix contained in the IP-Address AVP. Within the Rule-IPv6-Prefix AVP, the minimum valid prefix length is 0. The maximum value is bounded by the length of the End-user IPv6 prefix associated with the mapping rule, if present in the form of the Delegated-IPv6-Prefix AVP in the enclosing MAP-E-Attributes AVP. Otherwise the maximum value is 128.

3.9.3. EA-Field-Length AVP

The EA-Field-Length AVP (AVP Code TBD15) is of type Unsigned32. Valid values range from 0 to 48. See Section 5.2 of [I-D.ietf-softwire-map] for a description of the use of this parameter in deriving IPv4 address and port number configuration.

4. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
IP-Prefix-Length	TBD00	3.1	Unsigned32		V
Border-Router-Name	TBD01	3.2	OctetString		V
64-Multicast-Attributes	TBD02	3.3	Grouped		V
ASM-mPrefix64	TBD03	3.3.1	Grouped		V
SSM-mPrefix64	TBD04	3.3.2	Grouped		V
Tunnel-Source-Pref-Or-Addr	TBD05	3.4	Grouped		V
Tunnel-Source-IPv6-Address	TBD06	3.4.2	Address		V
Port-Set-Identifier	TBD07	3.5	OctetString		V
LW4over6-Binding	TBD08	3.6	Grouped		V
LW4over6-External-IPv4-Addr	TBD09	3.6.1	Address		V
MAP-E-Attributes	TBD10	3.7	Grouped		V
MAP-Mesh-Mode	TBD11	3.8	Enumerated		V
MAP-Mapping-Rule	TBD12	3.9	Grouped		V
Rule-IPv4-Addr-Or-Prefix	TBD13	3.9.1	Grouped		V
Rule-IPv6-Prefix	TBD14	3.9.2	Grouped		V
EA-Field-Length	TBD15	3.9.3	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.

5. Acknowledgements

Huawei Technologies funded Tom Taylor's work on earlier versions of this document.

Special thanks to Lionel Morand for the detailed review.

6. IANA Considerations

This memo requests to IANA to register the following Diameter AVP codes:

Code	Attribute Name	Reference
TBD00	IP-Prefix-Length	This document
TBD01	Border-Router-Name	This document
TBD02	64-Multicast-Attributes	This document
TBD03	ASM-mPrefix64	This document
TBD04	SSM-mPrefix64	This document
TBD05	Tunnel-Source-Pref-Or-Addr	This document
TBD06	Tunnel-Source-IPv6-Address	This document
TBD07	Port-Set-Identifier	This document
TBD08	LW4over6-Binding	This document
TBD09	LW4over6-External-IPv4-Addr	This document
TBD10	MAP-E-Attributes	This document
TBD11	MAP-Mesh-Mode	This document
TBD12	MAP-Mapping-Rule	This document
TBD13	Rule-IPv4-Addr-Or-Prefix	This document
TBD14	Rule-IPv6-Prefix	This document
TBD15	EA-Field-Length	This document

Table 1

7. Security Considerations

The AVPs defined in this document face two threats, both dependent on man-in-the-middle attacks on the Diameter delivery path. The more serious threat is denial of service through modification of the AVP contents leading to misconfiguration. The lesser threat is disclosure of subscriber addresses allowing the attacker to track subscriber activity.

Diameter security is currently provided on a hop-by-hop basis (see Section 2.2 of [RFC6733]). The Diameter end-to-end security problem has not been solved, so man-in-the-middle attacks on Diameter peers along the path are possible. The present document does not propose to solve that general problem, but simply warn that it exists.

8. References

8.1. Normative References

[I-D.ietf-softwire-lw4over6]
Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture (work in progress)", March 2014.

[I-D.ietf-softwire-map]
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP) (work in progress)", January 2014.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

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

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, August 2002.

[RFC4818] Salowey, J. and R. Droms, "RADIUS Delegated-IPv6-Prefix Attribute", RFC 4818, April 2007.

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

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

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

8.2. Informative References

[I-D.ietf-softwire-dslite-multicast]
Qin, J., Boucadair, M., Jacquenet, C., Lee, Y., and Q. Wang, "Delivery of IPv4 Multicast Services to IPv4 Clients over an IPv6 Multicast Network", draft-ietf-softwire-dslite-multicast-09 (work in progress), March 2015.

[I-D.ietf-softwire-map-dhcp]
Mrugalski, T., Troan, O., Farrer, I., Perrault, S., Dec, W., Bao, C., Yeh, L., and X. Deng, "DHCPv6 Options for configuration of Softwire Address and Port Mapped Clients (Work in progress)", March 2014.

[I-D.ietf-softwire-multicast-prefix-option]
Boucadair, M., Qin, J., Tsou, T., and X. Deng, "DHCPv6 Option for IPv4-Embedded Multicast and Unicast IPv6 Prefixes", draft-ietf-softwire-multicast-prefix-option-08 (work in progress), March 2015.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

[RFC6334] Hankins, D. and T. Mrugalski, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite", RFC 6334, August 2011.

[RFC6519] Maglione, R. and A. Durand, "RADIUS Extensions for Dual-Stack Lite", RFC 6519, February 2012.

Authors' Addresses

Cathy Zhou
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: cathy.zhou@huawei.com

Tom Taylor
PT Taylor Consulting
Ottawa
Canada

Email: tom.taylor.stds@gmail.com

Qiong Sun
China Telecom
P.R.China

Phone: 86 10 58552936
Email: sunqiong@ctbri.com.cn

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Diameter Agent Overload
draft-ietf-dime-agent-overload-01.txt

Abstract

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) base solution. The extension addresses the handling of occurrences of overload of a Diameter agent, or more generally, a Diameter peer.

Requirements

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

The base Diameter overload specification [I-D.ietf-dime-ovli] addresses the handling of overload when a Diameter endpoint (a Diameter Client or Diameter Server as defined in [RFC6733]) becomes overloaded.

In the base specification, the goal is to handle abatement of the overload 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.

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

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

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

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

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

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

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

2. Terminology and Abbreviations

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

Diameter Node

A RFC6733 Diameter Client, an RFC6733 Diameter Server, and RFC6733 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, those involving the overload of agents and those involving overload of Diameter endpoints (Diameter Clients and Diameter Servers).

3.1. Diameter Agent Overload Use Cases

The agent overload extension must support 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.

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

Figure 1

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

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

Figure 2

In both of these cases, the occurrence of overload in the single agent must by handled by the client in a similar fashion as if the client were handling the overload of a directly connected server. When the agent becomes overloaded it will insert an overload report in answer messages flowing to the client. This overload report will contain a requested reduction in the amount of traffic sent to the agent. The client will apply overload abatement behavior as defined in the base Diameter overload specification [I-D.ietf-dime-ovli] or the extension draft that defines the indicated 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.

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

Figure 3

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

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

Figure 4

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

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

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

3.1.3. Agent Chains

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

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

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

Figure 5

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

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

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

The handling of 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 report feature involving Diameter Clients and Diameter Servers.

3.2.1. Hop-by-hop Abatement Algorithms

It is envisioned that abatement algorithms will be defined that will support the option for Diameter Endpoints to send peer reports. For instance, it is envisioned that one usage scenario for the rate algorithm, [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 a server in the path of a transaction are overloaded at the same time. When this occurs, Diameter entities will need to handle both overload reports. 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 next DOIC node in the path of the request to determine if the indication of support came from a Diameter peer or if the request traversed a node that does not support the peer feature.

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 to insert OC-Supported-Feature AVP indicating support for the OLR_PEER_REPORT feature and OC-OLR reports of type PEER_REPORT into answer messages. These AVP OLR 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 receiving a request a DOIC node that supports the Peer Report feature MUST remove any received OC-SourceID AVP from the OC-Supported-Features AVP. This is done to prevent the OC-SourceID AVP from being included in a relayed message through a node that supports the Peer Report feature.

Note: If the DOIC node relays the message then it will insert an OC-SourceID AVP with its own DiameterID in the OC-Supported-Features AVP in the relayed message.

When sending 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 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.

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

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

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

5.2.1.2. Reacting Node Peer Report OCS

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

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

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

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

5.2.2. Reporting Node 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 [DOIC] 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 an a report type of peer it MUST determine if the report was generated by the Diameter peer from which the report was received.

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

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

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

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

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

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

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

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

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

> Editor’s note: The above four paragraphs are copied form the DOIC specification. Is it possible to include this behavior by reference or do we need to include all of these statements in this specification as well.

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

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

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

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

5.2.4. Peer Report Reporting Node Behavior

When there is an existing 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 a peer overload report.

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

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

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

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

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

matches the DiameterID in the SourceID AVP received in the peer overload report.

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

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

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

6.1.1. OC-Feature-Vector

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

OLR_PEER_REPORT (0x0000000000000010)

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

6.1.2. OC-Peer-Algo

The OC-Peer-Algo AVP (AVP code TBD6) is of type Unsigned64 and contains a 64 bit flags field of announced capabilities of a DOIC node. The value of zero (0) is reserved.

Feature bits defined for the OC-Feature-Vector AVP and associated with overload abatement algorithms are reused in for this AVP.

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

It contains the Diameter Identity of the inserting node. This is used by other DOIC nodes to determine if the a peer indicated 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	TBD1	x.x	Unsigned64		V

7. IANA Considerations

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

8. Security Considerations

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

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

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

9. Acknowledgements

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

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

10. Normative References

[I-D.ietf-dime-doic-rate-control]
Donovan, S. and E. Noel, "Diameter Overload Rate Control", draft-ietf-dime-doic-rate-control-00 (work in progress), December 2014.

[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, March 1997.

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

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

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

Author's Address

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

Email: srdonovan@usdonovans.com

Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended Status: Proposed Standard
Expires: April 30, 2014

L. Bertz
S. Manning
Sprint
B. Hirschman
October 2014

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

Abstract

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

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

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

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

Status of This Memo

Table of Contents

1. Introduction

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

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

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

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

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

2. Terminology and Abbreviations

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

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

3.1. ECN-IP-Codepoint AVP

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

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

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

3.2. Congestion-Treatment AVP

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

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

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

3.3. Flow-Count AVP

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

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

3.4. Packet-Count AVP

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

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

4. IANA Considerations

4.1. AVP Codes

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

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

5. Examples

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

5.1. Classifier Example

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

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

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

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

5.2. Diameter Credit Control (CC) with Congestion Information

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

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

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

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

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

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

6. Security Considerations

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

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

7. Acknowledgements

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

8. References

8.1. Normative References

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

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

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

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

Authors' Addresses

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

EMail: lyleb551144@gmail.com

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

EMail: sergem913@gmail.com

Brent Hirschman

EMail: Brent.Hirschman@gmail.com

Diameter Maintenance and Extensions (DIME) S. Donovan, Ed.
Internet-Draft Oracle
Intended status: Standards Track E. Noel
Expires: September 7, 2015 AT&T Labs
March 6, 2015

Diameter Overload Rate Control
draft-ietf-dime-doic-rate-control-01.txt

Abstract

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

Requirements

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a new Diameter overload control abatement algorithm.

The base Diameter overload specification [I-D.ietf-dime-ovli] defines the loss algorithm as the default Diameter overload abatement algorithm. The loss algorithm allows a reporting node to instruct a reacting node to reduce the amount of traffic sent to the reporting node by 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 subject 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 [I-D.ietf-dime-ovli] to add support for the rate based overload abatement algorithm.

This document draws heavily on work in the RIA SIP Overload Control working group. The 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.

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

2. Terminology and Abbreviations

Diameter Node

> A RFC6733 Diameter Client, 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 [I-D.ietf-dime-ovli].

Reporting Node

> A DOIC Node that sends a DOIC overload report.

Reacting Node

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

3. Interaction with DOIC report types

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

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

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

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

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

> Editor’s note: It needs to be validated that use of the rate algorithm applies to the host and realm 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 [I-D.ietf-dime-ovli].

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

> There may be local policy reasons that cause a DOIC node that supports the rate 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 [I-D.ietf-dime-ovli].

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 [I-D.ietf-dime-ovli].

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 transaction 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 [I-D.ietf-dime-ovli] for handling of overload reports when the rate overload abatement algorithm is used.

5.1. Reporting Node Overload Control State

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

> This is different from the behavior defines in [I-D.ietf-dime-ovli] 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 be able to include the specified rate in the abatement algorithm specific portion of the reporting node rate OCS.

All other elements for the OCS defined in [I-D.ietf-dime-ovli] 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 be able to include rate as the selected abatement algorithm in the reacting node OCS.

A reacting node that supports the rate abatement algorithm MUST be able to include the rate specified in the OC-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 [I-D.ietf-dime-ovli] 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:

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

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

- o For Peer reports the target is the DiameterID 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.

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

When sending an overload report for the Rate algorithm, the OC-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 [I-D.ietf-dime-ovli] and [I-D.ietf-dime-agent-overload] apply.

6. Rate Abatement Algorithm AVPs

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

6.1. OC-Supported-Features AVP

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

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

6.1.1. OC-Feature-Vector AVP

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

OLR_RATE_ALGORITHM (0x0000000000000004)

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

6.2. OC-OLR AVP

This extension defines the OC-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 [I-D.ietf-dime-ovli] 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.

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

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

6.3. Attribute Value Pair flag rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
OC-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:

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

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

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

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

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

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

10. Acknowledgements

11. References

11.1. Normative References

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

[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, March 1997.

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

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

11.2. Informative References

[RFC7415] Noel, E. and P. Williams, "Session Initiation Protocol (SIP) Rate Control", RFC 7415, February 2015.

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

DIME H. Tschofenig
Internet-Draft ARM Limited
Intended status: Informational J. Korhonen, Ed.
Expires: December 19, 2015 Broadcom Corporation
G. Zorn
Network Zen
K. Pillay
Oracle Communications
June 17, 2015

Diameter AVP Level Security End-to-End Security: Scenarios and Requirements
draft-ietf-dime-e2e-sec-req-03.txt

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Diameter Base specification [2] offers security protection between neighboring Diameter peers and mandates that either TLS (for TCP), DTLS (for SCTP), or 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.

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 Diameter topology where a Diameter clients want to interact with the example.com home domain. To interconnect the two visited networks a AAA interconnection provider, labeled as AAA Broker, is used.

```
      +oooooooooooooooooo+              +====================+
      |   Example.net    |              |                    |
      |                  |              |                    |
   +--------+      +--------+        +--------+        +--------+
   |Diameter|      |Diameter+--------+Diameter|        |Diameter|
   |Client 1+------+Proxy A1| +------+Proxy B +--------+Proxy C |----+
   +--------+      +--------+ |      +--------+        +--------+    |
      |                  |    |         |                    |       |
      | Visited Domain 1 |    |         |      AAA Broker    |       |
      +oooooooooooooooooo+    |         +====================+       |
                              |                                      |
                              |                                      |
                              |                                      |
                              |            +\\\\\\\\\\\\\\\\\\\\+    |
                              |     +--------+  Example.com     |    |
                              |     |Diameter|                  |    |
      +oooooooooooooooooo+    |     |Server X+--+         +--------+ |
      |   Example.org    |    |     +--------+  |         |Diameter| |
      |                  |    |     +--------+  +---------+Proxy D |-+
   +--------+      +--------+ |     |Diameter|  |         +--------+
   |Diameter|      |Diameter| |     |Server Y+--+               |
   |Client 2+------+Proxy A2+-+     +--------+    Home Domain   |
   +--------+      +--------+              +////////////////////+
      |                  |
      | Visited Domain 2 |
      +oooooooooooooooooo+
```

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 keying material for the protection of air interface between the end device and the network access server to be carried from the Diameter server to the Diameter client (using the EAP-Master-Session-Key AVP). The content of the EAP-Master-Session-Key AVP would benefit from protection against eavesdropping by intermediaries. Other AVPs 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 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. Diameter AVPs exchanged between these two Diameter nodes are protected.

```
+--------+                                                +--------+
|Diameter| AVP, {AVP}k                                    |Diameter|
|Client  +------------------.......... ------------------+Server  |
+--------+                                                +--------+
```

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.

```
+--------+     +--------+                                 +--------+
|Diameter| AVP |Diameter|   AVP, {AVP}k                   |Diameter|
|Client  +-----+Proxy A +---------- .......... -----------+Server  |
+--------+     +--------+                                 +--------+
```

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.

```
+--------+                                 +--------+     +--------+
|Diameter| AVP, {AVP}k                     |Diameter| AVP |Diameter|
|Client  +------------------.......... ----+Proxy D +-----+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 key pairs need to be negotiated and provisioned.

```
+--------+     +--------+                  +--------+     +--------+
|Diameter| AVP |Diameter|   AVP, {AVP}k    |Diameter| AVP |Diameter|
|Client  +-----+Proxy A +-- .......... ----+Proxy D +-----+Server  |
+--------+     +--------+                  +--------+     +--------+
```

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: Solutions 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: Solutions 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: Solutions MUST support replay protection. All Diameter nodes have access to network time and thus can synchronize their clocks.

Requirement #4: Solutions 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: Solutions 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: Solutions MUST recommend 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: Solutions 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 and Tom Taylor 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, March 1997.

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

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

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: January 7, 2016

M. Jones
M. Liebsch
L. Morand

July 6, 2015

Diameter Group Signaling
draft-ietf-dime-group-signaling-05.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 specifcation, 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-intiated 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 (lient 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, March 1997.

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

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

Appendix A. Session Management -- Exemplary Session State Machine

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

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

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

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

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

CLIENT, STATEFUL

State	Event	Action	New State
Idle	Client or Device Requests access	Send service specific auth req optionally including groups	Pending
Open	GASR received with Group-Response-Action = ALL_GROUPS, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send GSTR.	Discon
Open	GASR received with Group-Response-Action = PER_GROUPS, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send GSTR per group	Discon
Open	GASR received with Group-Response-Action = PER_SESSION, session is assigned to received group(s) and client will comply with request to end the session	Send GASA with Result-Code = SUCCESS, Send STR per session	Discon
Open	GASR received, client will not comply with request to end all session	Send GASA with Result-Code	Open

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

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

SERVER, STATEFUL

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

Authors' Addresses

Mark Jones

Email: mark@azu.ca

Marco Liebsch

Email: marco.liebsch@neclab.eu

Lionel Morand

Email: lionel.morand@orange.com

Internet Engineering Task Force B. Campbell
Internet-Draft S. Donovan, Ed.
Intended status: Informational Oracle
Expires: January 7, 2016 JJ. Trottin
Alcatel-Lucent
July 6, 2015

Diameter Load Information Conveyance
draft-ietf-dime-load-00

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

2. Terminology and Abbreviations

DOIC

Diameter Overload Information Conveyance

Load

The relative capacity of a Diameter node. A low value indicates that the Diameter node is under utilized. A high value indicated 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.

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

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

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

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 [I-D.ietf-dime-ovli]. When a reacting node diverts traffic away from an overloaded node, it needs load information for the other candidates for that traffic in order to effectively load balance the diverted load between potential candidates. Otherwise, diversion has a greater potential to drive other nodes into overload.

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

If the loaded nodes are Diameter servers (or clients in the case of server-to-client transactions), both of these uses are most effectively accomplished by a Diameter node that performs server selection. Typically, server selection is performed by a node (a client or an agent) that is an immediate peer of the server. However, there are scenarios (see 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.

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

5. Solution Overview

The mechanism defined here for the conveyance of load information is similar in some says 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 extension then the load information is ignored.

The second big difference between DOIC and Load is that DOIC information is sent end-to-end. 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, the load information is targeted for the peer of the sender of the Load AVPs.

> Editor’s note: It is still being discussed whether a second type of Load report can be included that is the load of the end-point and is carried end-to-end.

The Load report applies to an individual node. The Diameter identity of the node to which it applies is included in the Load report.

> Note, this is necessary so that supporting nodes can determine if the load report came from a peer node. If the identity is for a non peer node than the peer load report can be ignored.

In addition to the identity of the node to which the report applies, 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 method for calculating the load value included in the load report is 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 value. 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:

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

Figure 1: Example Diameter Network

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

```
C      A1      A4      S[n]
|      |       |       |
|----->|------>|------>|
xxR     xxR     xxR
```

Figure 2: Request Message Path

When sending the answer message, a sending node that supports the Diameter Load mechanism includes it’s own load information in the answer message:

```
C      A1      A4      S[n]
|      |       |       |
|      |       |<------|
                xxA(Load S[n])
```

Figure 3: Answer Message from S[n]

If Agent A4 supports the Load mechanism then it will verify that the load information received was from its peer. This is achieved by matching the identity included in the load information with the identity of the peer node from which the answer message was received.

If the load information does not match then the agent will strip the load AVPs from the message.

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

In all cases, A4 strips the load AVPs from the message.

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

```
C      A1      A4      S[n]
|       |       |       |
|       |<------|       |
         xxA(Load A4)
```

Figure 4: Answer Message from A4

A1 follows the same procedures as A4, resulting in the following message sent to C:

```
C      A1      A4      S[n]
|       |       |       |
|<------|       |       |
 xxA(Load A1)
```

Figure 5: Answer Message from A1

C follows the same procedure 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.

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

Editor’s note: The above needs to be updated if there is agreement to support end-point Load reports.

6. Solution Procedures

6.1. Reporting Node Behavior

6.1.1. Endpoint Reporting Node Behavior

6.1.2. Agent Reporting Node Behavior

6.2. Receiving Node Behavior

6.2.1. Endpoint Receiving Node Behavior

6.2.2. Agent Receiving Node Behavior

6.3. Extensibility

7. Attribute Value Pairs

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

This document makes no requests of IANA.

10. References

10.1. Normative References

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

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

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

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

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, February 2000.

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.

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

Figure 6: Basic Client Server Scenario

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

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

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.

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

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

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

Figure 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.

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

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.

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

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.

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

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.

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

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.

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

Figure 13: Full Mesh

A.8. Partitions

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

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

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. It should be determined if this use case is in the scope of the load control mechanism.

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

Authors' Addresses

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

Email: ben@nostrum.com

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

Email: srdonovan@usdonovans.com

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

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

Diameter Maintenance and Extensions (DIME) J. Korhonen, Ed.
Internet-Draft Broadcom
Intended status: Standards Track S. Donovan, Ed.
Expires: August 8, 2015 B. Campbell
Oracle
L. Morand
Orange Labs
February 4, 2015

Diameter Overload Indication Conveyance
draft-ietf-dime-ovli-08.txt

Abstract

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

Table of Contents

1. Introduction

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

This specification addresses Diameter overload control between Diameter nodes that support the DOIC solution. The solution, which is designed to apply to existing and future Diameter applications, requires no changes to the Diameter base protocol [RFC6733] and is deployable in environments where some Diameter nodes do not implement the Diameter overload control solution defined in this specification.

A new application specification can incorporate the overload control mechanism specified in this document by making it mandatory to implement for the application and referencing this specification normatively. It is the responsibility of the Diameter application designers to define how overload control mechanisms works on that application.

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

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 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 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. 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.

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.

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.

4.4. DOIC Extensibility

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

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

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

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

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

4.5. Simplified Example Architecture

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

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


      +--------+                : (optional) :
      |Diameter|                :            :
      |Server A|--+     .--.    : +--------+ :     .--.
      +--------+  |   _(    ‘.  : |Diameter| :   _(    ‘.   +--------+
                  +--(        )--:-|  Agent |-:--(        )--|Diameter|
      +--------+  | ( ‘  .  )  ) : +--------+ : ( ‘  .  )  ) | Client |
      |Diameter|--+  ‘--(___.-’  :            :  ‘--(___.-’  +--------+
      |Server B|                :            :
      +--------+                :            :

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

                  Overload Indication A    Overload Indication A’
             2)  <----------------------> <---------------------->
                 Diameter Application Y   Diameter Application Y
```

Figure 1: Simplified architecture choices for overload indication delivery

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

5. Solution Procedures

This section outlines the normative behavior for the DOIC solution.

5.1. Capability Announcement

This section defines DOIC Capability Announcement (DCA) behavior.

Note: This specification assumes that changes in DOIC node capabilities are relatively rare events that occur as a result of administrative action. Reacting nodes ought to minimize changes that force the reporting node to change the features being used, especially during active overload conditions. But even if reacting nodes avoid such changes, reporting nodes still have to be prepared for them to occur. For example, differing capabilities between multiple reacting nodes may still force a reporting node to select different features on a per-transaction basis.

5.1.1. Reacting Node Behavior

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

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

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

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

5.1.2. Reporting Node Behavior

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

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

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

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

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

A reporting node MUST indicate support for one and only one abatement algorithm in the OC-Feature-Vector AVP. The abatement algorithm selected MUST indicate the abatement algorithm the reporting node wants the reacting node to use when the reporting node enters an overload condition.

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

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

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

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

5.1.3. Agent Behavior

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

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

For a Diameter Agent to be a reacting node for a non supporting Diameter endpoint, the Diameter Agent MUST include the OC-Supported-

Features AVP in request messages it relays that do not contain the OC-Supported-Features AVP.

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

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

> For instance, if the agent supports a superset of the features reported by the reacting node then the agent might choose, based on local policy, to advertise that superset of features to the reporting node.

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

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

5.2. Overload Report Processing

5.2.1. Overload Control State

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

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

5.2.1.1. Overload Control State for Reacting Nodes

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

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

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

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

A realm-type OCS entry is identified by the pair of application-id and realm.

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

- o Sequence number (as received in OC-OLR)

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

- o Selected Abatement Algorithm (as received in the OC-Supported-Features AVP)

- o Abatement Algorithm specific input data (as received in the OC-OLR AVP, for example, OC-Reduction-Percentage for the Loss abatement algorithm)

5.2.1.2. Overload Control State for Reporting Nodes

A reporting node maintains OCS entries per supported Diameter application, per supported (and eventually selected) Abatement Algorithm and per report-type.

An OCS entry is identified by the tuple of Application-Id, Report-Type and Abatement Algorithm and includes the following information (the actual information stored is an implementation decision):

- o Sequence number

- o Validity Duration

- o Expiration Time

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

5.2.1.3. Reacting Node Maintenance of Overload Control State

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

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

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

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

For a host-report this means it matches the application-id and the host’s DiameterIdentity in an existing host OCS entry.

For a realm-report this means it matches the application-id and the realm in an existing realm OCS entry.

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

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

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

If the 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 defined 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 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 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.

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).

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

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

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.

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 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. 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, and 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.

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

Diameter end-to-end security mechanism with overload control will require careful consideration, and are beyond the scope of this document.

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, March 1997.

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

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

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, August 2005.

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

[S13] 3GPP, , "ETSI TS 129 272 V11.9.0", December 2012.

Appendix A. Issues left for future specifications

The base solution for the overload control does not cover all possible use cases. A number of solution aspects were intentionally left for future specification and protocol work. 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. The DIME working group is considering defining additional error codes or AVPs 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.

Appendix C. Requirements Conformance Analysis

This section contains the result of an analysis of the DOIC solutions conformance to the requirements defined in [RFC7068].

C.1. Deferred Requirements

The 3GPP has adopted an early version of this document as normative references in various Diameter related specifications to support the overload control mechanism in their release 12 framework. The DIME working group has therefore decided to defer certain requirements in order to complete the design of an extensible, generic solution before the deadline scheduled by the 3GPP for the completion of the release 12 protocol work by the end of 2014. The deferred work includes the following:

- Agent Overload - The ability for an agent to report an overload condition of the agent itself.

- Load Information - The ability for a node to report its load level when not overloaded.

At the time of this writing, DIME has begun separate work efforts for these requirements.

C.2. Detection of non-supporting Intermediaries

The DOIC mechanism as currently defined does not allow supporting nodes to automatically determine whether OC-Supported-Features or OC-OLR AVPs are originated by a peer node, or by a non-peer node and sent across a non-supporting peer. This makes it impossible to detect the presence of non-supporting nodes between supporting nodes, except by configuration. The working group determined that such a configuration requirement is acceptable.

This limits full compliance with certain requirements related to the limitation of new configuration, deployment in environments with mixed support, operating across non-supporting agents, and authorization.

C.3. Implicit Application Indication

The working group elected to determine the application for an overload report from that of the enclosing message. This prevents sending an OLR for an application when there are no transactions for that application.

As a consequence, DOIC does not comply with the requirement to be able to report overload information across quiescent connections. DOIC does not fully comply with requirements to operate on up-to-date information, since if an OLR causes all transactions to stop for an application, the only way traffic will resume is for the OLR to expire.

C.4. Stateless Operation

RFC7068 explicitly discourages the sending of OLRs in every answer message, as part of the requirement to avoid additional work for overloaded nodes. DOIC recommends exactly that behavior during active overload conditions. The working group determined that doing otherwise would reduce reliability and increase statefulness. (Note that DOIC does allow nodes to avoid sending OLRs in every answer if they have some other method of ensuring that OLRs get to all relevant reacting nodes.)

C.5. No New Vulnerabilities

The working group believes that DOIC is compliant with the requirement to avoid introducing new vulnerabilities. However, this requirement may warrant an early security expert review.

C.6. Detailed Requirements

[RFC Editor: Please remove this section and subsections prior to publication as an RFC.]

C.6.1. General

REQ 1: The solution MUST provide a communication method for Diameter nodes to exchange load and overload information.

Partially Compliant. The mechanism uses new AVPs piggybacked on existing Diameter messages to exchange overload information. It does not currently support "load" information or the ability to report overload of an agent. These have been left for future extensions.

REQ 2: The solution MUST allow Diameter nodes to support overload control regardless of which Diameter applications they support. Diameter clients and agents must be able to use the received load and overload information to support graceful behavior during an overload condition. Graceful behavior under overload conditions is best described by REQ 3.

Partially Compliant. The DOIC AVPs can be used in any application that allows the extension of AVPs. However, "load" information is not currently supported.

REQ 3: The solution MUST limit the impact of overload on the overall useful throughput of a Diameter server, even when the incoming load on the network is far in excess of its capacity. The overall useful throughput under load is the ultimate measure of the value of a solution.

Compliant. DOIC provides information that nodes can use to reduce the impact of overload.

REQ 4: Diameter allows requests to be sent from either side of a connection, and either side of a connection may have need to provide its overload status. The solution MUST allow each side of a connection to independently inform the other of its overload status.

Compliant. DOIC AVPs can be included regardless of transaction "direction"

REQ 5: Diameter allows nodes to determine their peers via dynamic discovery or manual configuration. The solution MUST work consistently without regard to how peers are determined.

Compliant. DOIC contains no assumptions about how peers are discovered.

REQ 6: The solution designers SHOULD seek to minimize the amount of new configuration required in order to work. For example, it is better to allow peers to advertise or negotiate support for the solution, rather than to require that this knowledge to be configured at each node.

Partially Compliant. Most DOIC parameters are advertised using the DOIC capability announcement mechanism. However, there are some situations where configuration is required. For example, a DOIC node detect the fact that a peer may not support DOIC when nodes on the other side of the non-supporting node do support DOIC without configuration.

C.6.2. Performance

REQ 7: The solution and any associated default algorithm(s) MUST ensure that the system remains stable. At some point after an overload condition has ended, the solution MUST enable capacity to stabilize and become equal to what it would be in the absence of an overload condition. Note that this also requires that the solution MUST allow nodes to shed load without introducing non-converging oscillations during or after an overload condition.

Compliant. The specification offers guidance that implementations should apply hysteresis when recovering from overload, and avoid sudden ramp ups in offered load when recovering.

REQ 8: Supporting nodes MUST be able to distinguish current overload information from stale information.

Partially Compliant. DOIC overload reports are "soft state", that is they expire after an indicated period. DOIC nodes may also send reports that end existing overload conditions. DOIC requires reporting nodes to ensure that all relevant reacting nodes receive overload reports.

However, since DOIC does not allow reporting nodes to send OLRs in watchdog messages, if an overload condition results in zero offered load, the reporting node cannot update the condition until the expiration of the original OLR.

REQ 9: The solution MUST function across fully loaded as well as quiescent transport connections. This is partially derived from the requirement for stability in REQ 7.

Not Compliant. DOIC does not allow OLRs to be sent over quiescent transport connections. This is due to the fact that OLRs cannot be sent outside of the application to which they apply.

REQ 10: Consumers of overload information MUST be able to determine when the overload condition improves or ends.

Partially Compliant. (See response to previous two requirements.)

REQ 11: The solution MUST be able to operate in networks of different sizes.

Compliant. DOIC makes no assumptions about the size of the network. DOIC can operate purely between clients and servers, or across agents.

REQ 12: When a single network node fails, goes into overload, or suffers from reduced processing capacity, the solution MUST make it possible to limit the impact of the affected node on other nodes in the network. This helps to prevent a small-scale failure from becoming a widespread outage.

Partially Compliant. DOIC allows overload reports for an entire realm, where abated traffic will not be redirected towards another server. But in situations where nodes choose to divert traffic to other nodes, DOIC offers no way of knowing whether the new recipients can handle the traffic if they have not already indicated overload. This may be mitigated with the use of a future "load" extension, or with the use of proprietary dynamic load-balancing mechanisms.

REQ 13: The solution MUST NOT introduce substantial additional work for a node in an overloaded state. For example, a requirement for an overloaded node to send overload information every time it received a new request would introduce substantial work.

Not Compliant. DOIC does in fact encourage an overloaded node to send an OLR in every response. The working group that other mechanisms to ensure that every relevant node receives an OLR would create even more work. [Note: This needs discussion.]

REQ 14: Some scenarios that result in overload involve a rapid increase of traffic with little time between normal levels and levels that induce overload. The solution SHOULD provide for rapid feedback when traffic levels increase.

Compliant. The piggyback mechanism allows OLRs to be sent at the same rate as application traffic.

REQ 15: The solution MUST NOT interfere with the congestion control mechanisms of underlying transport protocols. For example, a solution that opened additional TCP connections when the network is congested would reduce the effectiveness of the underlying congestion control mechanisms.

Compliant. DOIC does not require or recommend changes in the handling of transport protocols or connections.

C.6.3. Heterogeneous Support for Solution

REQ 16: The solution is likely to be deployed incrementally. The solution MUST support a mixed environment where some, but not all, nodes implement it.

Partially Compliant. DOIC works with most mixed-deployment scenarios. However, it cannot work across a non-supporting proxy that modifies Origin-Host AVPs in answer messages. DOIC will have limited impact in networks where the nodes that perform server selections do not support the mechanism.

REQ 17: In a mixed environment with nodes that support the solution and nodes that do not, the solution MUST NOT result in materially less useful throughput during overload as would have resulted if the solution were not present. It SHOULD result in less severe overload in this environment.

Compliant. In most mixed-support deployment, DOIC will offer at least some value, and will not make things worse.

REQ 18: In a mixed environment of nodes that support the solution and nodes that do not, the solution MUST NOT preclude elements that support overload control from treating elements that do not support overload control in an equitable fashion relative to those that do. Users and operators of nodes that do not support the solution MUST NOT unfairly benefit from the solution. The solution specification SHOULD provide guidance to implementers for dealing with elements not supporting overload control.

Compliant. DOIC provides mechanisms to abate load from non-supporting sources. Furthermore, it recommends that reporting nodes will still need to be able to apply whatever protections they would ordinarily apply if DOIC were not in use.

REQ 19: It MUST be possible to use the solution between nodes in different realms and in different administrative domains.

Partially Compliant. DOIC allows sending OLRs across administrative domains, and potentially to nodes in other realms. However, an OLR cannot indicate overload for realms other than the one in the Origin-Realm AVP of the containing answer.

REQ 20: Any explicit overload indication MUST be clearly distinguishable from other errors reported via Diameter.

Compliant. DOIC sends explicit overload indication in overload reports. It does not depend on error result codes.

REQ 21: In cases where a network node fails, is so overloaded that it cannot process messages, or cannot communicate due to a network failure, it may not be able to provide explicit indications of the nature of the failure or its levels of overload. The solution MUST result in at least as much useful throughput as would have resulted if the solution were not in place.

Compliant. DOIC overload reports have the primary effect of suppressing message retries in overload conditions. DOIC recommends that messages never be silently dropped if at all possible.

C.6.4. Granular Control

REQ 22: The solution MUST provide a way for a node to throttle the amount of traffic it receives from a peer node. This throttling SHOULD be graded so that it can be applied gradually as offered load increases. Overload is not a binary state; there may be degrees of overload.

Compliant. The "loss" algorithm expresses a percentage reduction.

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.

Not Compliant. DOIC provides no built in mechanism to determine the best place to divert messages that would otherwise be throttled. This can be accomplished with a future "load" extension, or with proprietary load balancing mechanisms.

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.

Not Compliant. "Load" information has been left for a future extension.

C.6.5. Priority and Policy

REQ 25: The base specification for the solution SHOULD offer general guidance on which message types might be desirable to send or process over others during times of overload, based on application-specific considerations. For example, it may be more beneficial to process messages for existing sessions ahead of new sessions. Some networks may have a requirement to give priority to requests associated with emergency sessions. Any normative or otherwise detailed definition of the relative priorities of message types during an overload condition will be the responsibility of the application specification.

Compliant. The specification offers guidance on how requests might be prioritized for different types of applications.

REQ 26: The solution MUST NOT prevent a node from prioritizing requests based on any local policy, so that certain requests are given preferential treatment, given additional retransmission, not throttled, or processed ahead of others.

Compliant. Nothing in the specification prevents application-specific, implementation-specific, or local policies.

C.6.6. Security

REQ 27: The solution MUST NOT provide new vulnerabilities to malicious attack or increase the severity of any existing vulnerabilities. This includes vulnerabilities to DoS and DDoS attacks as well as replay and man-in-the-middle attacks. Note that the Diameter base specification [RFC6733] lacks end-to-end security and this must be considered (see the Security Considerations in [RFC7068]). Note that this requirement was expressed at a high level so as to not preclude any particular solution. It is expected that the solution will address this in more detail.

Compliant. The working group is not aware of any such vulnerabilities. [This may need further analysis.]

REQ 28: The solution MUST NOT depend on being deployed in environments where all Diameter nodes are completely trusted. It SHOULD operate as effectively as possible in environments where other nodes are malicious; this includes preventing malicious nodes from obtaining more than a fair share of service. Note that this does not imply any responsibility on the solution to detect, or take countermeasures against, malicious nodes.

Partially Compliant. Since all Diameter security is currently at the transport layer, nodes must trust immediate peers to enforce trust policies. However, there are situations where a DOIC node cannot determine if an immediate peer supports DOIC. The authors recommend an expert security review.

REQ 29: It MUST be possible for a supporting node to make authorization decisions about what information will be sent to peer nodes based on the identity of those nodes. This allows a domain administrator who considers the load of their nodes to be sensitive information to restrict access to that information. Of course, in such cases, there is no expectation that the solution itself will help prevent overload from that peer node.

Partially Compliant. (See response to previous requirement.)

REQ 30: The solution MUST NOT interfere with any Diameter-compliant method that a node may use to protect itself from overload from non-supporting nodes or from denial-of-service attacks.

Compliant. The specification recommends that any such protection mechanism needed without DOIC should continue to be employed with DOIC.

C.6.7. Flexibility and Extensibility

REQ 31: There are multiple situations where a Diameter node may be overloaded for some purposes but not others. For example, this can happen to an agent or server that supports multiple applications, or when a server depends on multiple external resources, some of which may become overloaded while others are fully available. The solution MUST allow Diameter nodes to indicate overload with sufficient granularity to allow clients to take action based on the overloaded resources without unreasonably forcing available capacity to go unused. The solution MUST support specification of overload information with granularities of at least "Diameter node", "realm", and "Diameter application" and MUST allow extensibility for others to be added in the future.

Partially Compliant. All DOIC overload reports are scoped to the specific application and realm. Inside that scope, overload can be reported at the specific server or whole realm scope. As currently specified, DOIC cannot indicate local overload for an agent. At the time of this writing, the DIME working group has plans to work on an agent-overload extension.

DOIC allows new "scopes" through the use of extended report types.

REQ 32: The solution MUST provide a method for extending the information communicated and the algorithms used for overload control.

Compliant. DOIC allows new report types and abatement algorithms to be created. These may be indicated using the OC-Supported-Features AVP.

REQ 33: The solution MUST provide a default algorithm that is mandatory to implement.

Compliant. The "loss" algorithm is mandatory to implement.

REQ 34: The solution SHOULD provide a method for exchanging overload and load information between elements that are connected by intermediaries that do not support the solution.

Partially Compliant. DOIC information can traverse non-supporting agents, as long as those agents do not modify certain AVPs. (e.g., Origin-Host). DOIC does not provide a way for supporting nodes to detect such modification.

Appendix D. Considerations for Applications Integrating the DOIC Solution

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

D.1. Application Classification

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

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

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

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

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

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

Stateless Applications:

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

Pseudo-Session Applications:

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

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

D.2. Application Type Overload Implications

This section discusses considerations for mitigating overload reported by a Diameter entity. This discussion focuses on the type of application. Appendix D.3 discusses considerations for handling various request types when the target server is known to be in an overloaded state.

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

Stateless Applications:

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

Pseudo-Session Applications:

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

Session-Based Applications:

Overload handling for session-based applications must take into consideration the work load associated with setting up and maintaining a session. As such, the entity sending requests towards an overloaded Diameter entity for a session-based application might tend to reject new session requests prior to rejecting intra-session requests. In addition, session ending requests might be given a lower probability of being rejected as rejecting session ending requests could result in session status being out of sync between the Diameter clients and servers. Application designers that would decide to reject mid-session requests will need to consider whether the rejection invalidates the session and any resulting session cleanup procedures.

D.3. Request Transaction Classification

Independent Request:

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

Session-Initiating Request:

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

Correlated Session-Initiating Request:

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

Intra-Session Request:

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

D.4. Request Type Overload Implications

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

Independent Requests:

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

Session-Initiating Requests:

Session-initiating requests often represent more work than independent or intra-session requests. Moreover, session-initiating requests are typically followed by other session-related requests. Since the main objective of the overload control is to reduce the total number of requests sent to the overloaded entity, throttling decisions might favor allowing intra-session requests over session-initiating requests. In the absence of local policies or application specific requirements to the contrary, Individual session-initiating requests can be given equal treatment when making throttling decisions.

Correlated Session-Initiating Requests:

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

Pseudo-Session Requests:

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

Intra-Session Requests:

There are two types of intra-sessions requests, requests that terminate a session and the remainder of intra-session requests. 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