DIME
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2014

H. Tschofenig
Nokia Siemens Networks

July 16, 2013

Diameter Overload Architecture and Information Model
draft-tschofenig-dime-overload-arch-00.txt

Abstract

This document describes the architecture for Diameter overload control architecture in form of principles and presents an information model.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 17, 2014.

Table of Contents

1. Introduction

Diameter was developed to provide an Authentication, Authorization, and Accounting (AAA) framework for a number of applications, including network access, mobility, and real-time multimedia application layer services. Over the last 10 years it has enjoyed widespread industry acceptance and can be found in many large (mobile) operator networks.

When a Diameter server becomes overloaded, it may need to ask Diameter clients and agents to gracefully reduce the amount of signaling traffic destined for it. For Diameter clients and agents that are asked to reduce traffic there are two basic approaches for doing so:

- by sending a portion of new requests to other Diameter servers which still have capacity available (load balancing), and
- by rejecting new requests.

[3] aims to provide background information and requirements. This document defines the next step, namely the architecture and the information model.

2. Terminology

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

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

The term "load balancer" in this document refers to a Diameter proxy that uses load information received from Diameter servers and configuration settings to adjust standard Diameter routing.

3. Architectural Principles

This section outlines several principles guiding the solution design.

Avoid premature optimizations. Diameter is an extensible protocol and new extensions can be added at any time (when necessary). Instead of developing the most "complete" solution there are benefits from starting with a minimal core that helps to gain initial implementation and deployment experience. This minimal core can then be used as a basis for subsequent refinements. Complexity often comes with optimizations and constraints imposed by incremental deployment strategies.

Focus on real-world problems. The number of theoretical use cases is large, almost unbounded. What use cases and optimizations are worthwhile to explore and to standardize solutions for? Is the "sounds good" test enough or is evidence of a documented and reproducible real-world problem needed?

Overload conditions are rare events. The Diameter backend infrastructure is hopefully dimensioned in a way that overload situations are the exception rather than the norm. Consequently, it is less useful to develop many optimization for those rare events (e.g., optimizations in the form of message reduction). Instead, it is more beneficial to optimize for the case where there is no overload.

Consider advances in information technology. At the time of writing cloud computing and virtualization has gained widespread industry interest, even in the telecommunication sector. These new computing paradigms provide built-in support for handling load variation (e.g., by spawning new virtual machines or migrating code). The orchestration and management of these virtual machine instances is often provided in a centralized fashion using a hypervisor and these hypervisors come with management interfaces that obtain load and other status information from their virtual machine instances. It is not unlikely that these developments will also impact deployments of Diameter servers within a single administrative domain.

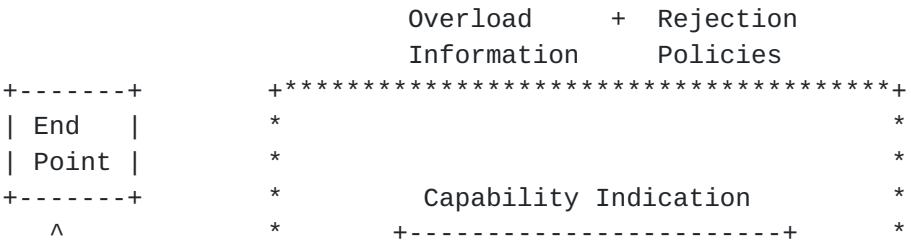
Load Balancing and Rejecting Requests are different. Load balancing is a technique that is best applied within a local administrative domain to re-distribute requests. For maximum benefit knowledge by the load balancer about the system architecture and the nature of applications is required. Rejecting requests, on the other hand, requires information signaling to the Diameter clients since Diameter is not an end-to-end protocol and information about the failure situation needs to be passed along to the source of the end system, such as VoIP phones initiating phone calls, end devices seeking network access.

Delegating rejection policies creates a lot of complexity. When a Diameter server reaches an overload state and wants to reduce the number of requests it gets it has a number of choices. In the simplest form it could return reject responses without engaging in heavy application specific processing. In such a model the Diameter server acts as a Policy Decision Point (PDP) and as a Policy Enforcement Point (PEP). By co-locating the PEP and the PDP the decision making is implementation internal, which is also the beautify of this model. If the PEP functionality, however, gets moved to the Diameter client (or to any other node) the need for standardizing a "Diameter request rejection policy language" arises. Delegating functionality from the PDP to the PEP requires the PEP to have a reasonable amount of information about the problem the Diameter server and the support infrastructure is facing. Additionally, the PEP would benefit from knowing about the tradeoff decisions the network operators wants to make regarding the different types of requests.

4. Information Exchanges

4.1. End-to-End Overload Feedback

The information elements described in this document follow the pattern shown in Figure 1. Following the exchange of Diameter application messages between a Diameter client and a Diameter server (through zero or more Diameter intermediaries) a Diameter server may indicate that it is currently suffering an overload situation. To ensure that the load information is understood by the Diameter client a prior capability exchange is provided.



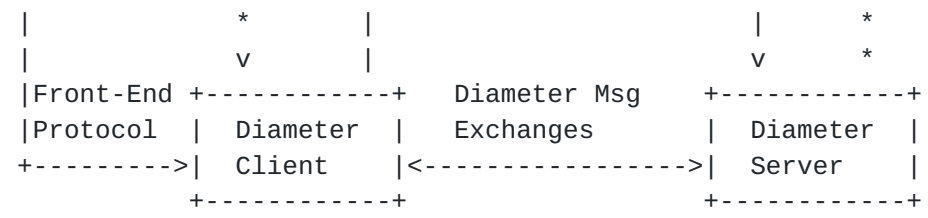


Figure 1: Information Exchange for End-to-End Overload Feedback.

The basic functionality starts with the support of DIAMETER_TOO_BUSY, which is defined in the Diameter Base specification. While it does indeed not provide information to the Diameter client about how it react on future Diameter messages. While this can be seen as a design weakness it also has the benefit of a lower standardization need and minimal implementation complexity. This document defines the information elements that can be used by an existing Diameter application to convey overload information.

The necessary AVPs are defined in the Section 5.1 and in Section 5.2.

4.2. Load Balancing

Figure 2 shows the information exchange between different Diameter servers and a load balancer within the same administrative domain. Following the capability exchange between the load balancer and the Diameter servers load information is exchanged. Incoming Diameter requests are distributed based on the collected load information and based on the configuration of the load balancer.

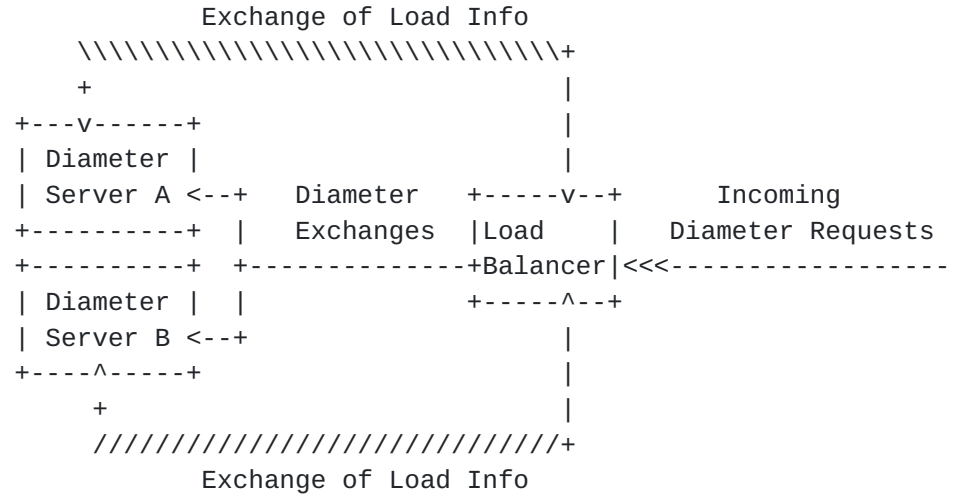


Figure 2: Information Exchange for Load Balancing.

The information elements relevant to load balancing are described in Section 5.1 and in Section 5.3.

5. Information Elements

5.1. Capability Indication

5.1.1. Supported-Features AVP

The Supported-Features AVP (AVP code TBD) is of type Uint64, and contains a bitmap that allows capability indication. The bitmap allows for up to 64 total values to be defined.

Bitmask	Feature
0000001	[TBD: This document.]

5.1.2. Sending-Rate AVP

The Sending-Rate AVP (AVP Code TBD10) is of type Unsigned32 and indicates the time in milliseconds between subsequent load updates. Higher values lead to fewer load updates and therefore reduce the signaling overhead but lead to less up-to-date information.

5.2. Information for Rejecting Diameter Requests

5.2.1. Overload-Info AVP

The Overload-Info AVP (AVP Code TBD) is of type Grouped, and is used as a top-level container for information about the overload status.

The grouped data field of the Overload-Info AVP has the following grammar:

```
< Overload-Info > ::= < AVP Header: TBD >
                      { Overload }
                      [ Period-Of-Validity ]
                    * [ AVP ]
```

5.2.2. Period-Of-Validity AVP

The Period-Of-Validity AVP (AVP code TBD) is of type Unsigned32, and is used to indicate the length of time, in seconds, the Overload-Metric is to be considered valid. The maximum value in this AVP is 5 minutes, which is also the default value. If this AVP is absent in a subsequent message then the indicated value is valid till the next overload report is received.

5.2.3. Overload AVP

The Overload AVP (AVP code TBD) is of type Enumerated. Four values are defined:

NO_OVERLOAD 0 The Diameter server uses this value to indicate that it is currently not overload and that the Diameter client should not reject any request.

INCREASING_OVERLOAD 1 The Diameter server uses this value to inform the client that overload has increased and that the Diameter client shall decrease the sending rate into half (calculated over the last 10 minutes).

DECREASING_OVERLOAD 2 The Diameter server uses this value to inform the client that the overload situation has improved and that the Diameter client is allowed to increase the rate of Diameter requests by 10% of the requests transmitted since the last overload message was received from the server. Consequencely, the Diameter server can quickly increase the sending rate by the client by including Overload AVPs with a value set to 'DECREASING_OVERLOAD' in subsequent exchanges.

OVERLOADED 3 The Diameter server uses this value to inform the client that it is completely overloaded and that the Diameter client shall not send further requests.

The increase and decrease of the sending rate refers to new requests to the same realm and the same application ID as the message carrying the overload information.

5.3. Information Elements for Load Balancing

5.3.1. Load-Info AVP

The Load-Info AVP (AVP Code TBD) is of type Grouped, and is used as a top-level container for information about the load status.

The grouped data field of the Load-Info AVP has the following grammar:

```
< Load-Info > ::= < AVP Header: TBD >
                    { Load }
                  * [ AVP ]
```

5.3.2. Load AVP

The Load AVP (AVP code TBD) is of type Unsigned32. The indicated value MUST be between 0 and 10. The semantics of the values are as follows: a Diameter server indicating a value of zero (0) notifies a load balancer that there is no overload condition. A Diameter server indicating a value of ten (10) indicates that it is experiencing extreme overload and cannot process any further requests. The remaining other values express situations between these two extremes. Note that there is no requirement that the Diameter server reports values in incremental steps; a Diameter server may, for whatever reason, notice that it needs to report a value of 10 even though it had previously reported a value of 0. A load balancer receiving values other than 0 MUST reduce the sending rate of Diameter requests to the Diameter server correspondingly by redistributing a portion of the requests (the higher the value the bigger the portion) to Diameter servers.

Note that the load value does not refer to any specific resource, such as CPU utilization, database interconnection, etc. The value is computed locally by the Diameter server based on an algorithm that is not further specified. Implementers should, however, ensure that the resources that matter for the specific Diameter deployment are taken into account in defining the algorithm. The lack of a standardized algorithm does, however, not impact interoperability. A load S.balancer provided by vendor A and Diameter servers provided by vendor B will interoperate because they make decisions based on these artificial values. For example, a network operator may decide to configure the load balancer in such a way that the second server is only used once the load value of the first server reaches a certain threshold.

The lifetime of the load information is bound to the value communicated in the Sending-Rate AVP during the capability exchange.

6. Security Considerations

This document describes architectural principles and an information model. Security considerations are described in the documents that utilize these AVPs.

7. IANA Considerations

7.1. Overload Capability Registry

IANA is requested to create a new registry under the 'Authentication, Authorization, and Accounting (AAA) Parameters' registry for the overload capability bitmap (with a total of 64 values). The policy for this registry is 'Specification Required'. One value is defined by this document, see Section 5.1.1.

7.2. AVP Codes

New AVPs defined by this specification are listed in Section 5. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

8. Acknowledgments

The author would like to thank Ulrich Wiehe for his feedback.

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] McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", draft-ietf-dime-overload-reqs-07 (work in progress), June 2013.

Author's Address

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at