

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 3, 2014

E. McMurry
B. Campbell
Tekelec
September 30, 2013

Diameter Overload Control Requirements
draft-ietf-dime-overload-reqs-13

Abstract

When a Diameter server or agent becomes overloaded, it needs to be able to gracefully reduce its load, typically by informing clients to reduce sending traffic for some period of time. Otherwise, it must continue to expend resources parsing and responding to Diameter messages, possibly resulting in a progressively more severe overload condition. The existing Diameter mechanisms are not sufficient for this purpose. This document describes the limitations of the existing mechanisms. Requirements for new overload management mechanisms are also provided.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 3, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Documentation Conventions	3
1.2. Causes of Overload	4
1.3. Effects of Overload	5
1.4. Overload vs. Network Congestion	5
1.5. Diameter Applications in a Broader Network	6
2. Overload Control Scenarios	6
2.1. Peer to Peer Scenarios	7
2.2. Agent Scenarios	9
2.3. Interconnect Scenario	12
3. Diameter Overload Case Studies	13
3.1. Overload in Mobile Data Networks	13
3.2. 3GPP Study on Core Network Overload	15
4. Existing Mechanisms	15
5. Issues with the Current Mechanisms	16
5.1. Problems with Implicit Mechanism	17
5.2. Problems with Explicit Mechanisms	17
6. Extensibility and Application Independence	18
7. Solution Requirements	19
7.1. General	19
7.2. Performance	20
7.3. Heterogeneous Support for Solution	21
7.4. Granular Control	21
7.5. Priority and Policy	22
7.6. Security	22
7.7. Flexibility and Extensibility	23
8. IANA Considerations	24
9. Security Considerations	24
9.1. Access Control	24
9.2. Denial-of-Service Attacks	25
9.3. Replay Attacks	25
9.4. Man-in-the-Middle Attacks	25
9.5. Compromised Hosts	26
10. References	26
10.1. Normative References	26
10.2. Informative References	26
Appendix A. Contributors	27
Appendix B. Acknowledgements	27
Authors' Addresses	28

1. Introduction

A Diameter [RFC6733] node is said to be overloaded when it has insufficient resources to successfully process all of the Diameter requests that it receives. When a node becomes overloaded, it needs to be able to gracefully reduce its load, typically by informing clients to reduce sending traffic for some period of time. Otherwise, it must continue to expend resources parsing and responding to Diameter messages, possibly resulting in a progressively more severe overload condition. The existing mechanisms provided by Diameter are not sufficient for this purpose. This document describes the limitations of the existing mechanisms, and provides requirements for new overload management mechanisms.

This document draws on the work done on SIP overload control ([RFC5390], [RFC6357]) as well as on experience gained via overload handling in Signaling System No. 7 (SS7) networks and studies done by the Third Generation Partnership Project (3GPP) (Section 3).

Diameter is not typically an end-user protocol; rather it is generally used as one component in support of some end-user activity.

For example, a SIP server might use Diameter to authenticate and authorize user access. Overload in the Diameter backend infrastructure will likely impact the experience observed by the end user in the SIP application.

The impact of Diameter overload on the client application (a client application may use the Diameter protocol and other protocols to do its job) is beyond the scope of this document.

This document presents non-normative descriptions of causes of overload along with related scenarios and studies. Finally, it offers a set of normative requirements for an improved overload indication mechanism.

1.1. Documentation Conventions

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 defined in [RFC2119], with the exception that they are not intended for interoperability of implementations. Rather, they are used to describe requirements towards future specifications where the interoperability requirements will be defined.

The terms "client", "server", "agent", "node", "peer", "upstream", and "downstream" are used as defined in [RFC6733].

1.2. Causes of Overload

Overload occurs when an element, such as a Diameter server or agent, has insufficient resources to successfully process all of the traffic it is receiving. Resources include all of the capabilities of the element used to process a request, including CPU processing, memory, I/O, and disk resources. It can also include external resources such as a database or DNS server, in which case the CPU, processing, memory, I/O, and disk resources of those elements are effectively part of the logical element processing the request.

External resources can include upstream Diameter nodes; for example, a Diameter agent can become effectively overloaded if one or more upstream nodes are overloaded.

A Diameter node can become overloaded due to request levels that exceed its capacity, a reduction of available resources (for example, a local or upstream hardware failure) or a combination of the two.

Overload can occur for many reasons, including:

Inadequate capacity: When designing Diameter networks, that is, application layer multi-node Diameter deployments, it can be very difficult to predict all scenarios that may cause elevated traffic. It may also be more costly to implement support for some scenarios than a network operator may deem worthwhile. This results in the likelihood that a Diameter network will not have adequate capacity to handle all situations.

Dependency failures: A Diameter node can become overloaded because a resource on which it depends has failed or become overloaded, greatly reducing the logical capacity of the node. In these cases, even minimal traffic might cause the node to go into overload. Examples of such dependency overloads include DNS servers, databases, disks, and network interfaces.

Component failures: A Diameter node can become overloaded when it is a member of a cluster of servers that each share the load of traffic, and one or more of the other members in the cluster fail. In this case, the remaining nodes take over the work of the failed nodes. Normally, capacity planning takes such failures into account, and servers are typically run with enough spare capacity to handle failure of another node. However, unusual failure conditions can cause many nodes to fail at once. This is often the case with software failures, where a bad packet or bad database entry hits the same bug in a set of nodes in a cluster.

Network Initiated Traffic Flood: Issues with the radio access network in a mobile network such as radio overlays with frequent handovers, and operational changes are examples of network events that can precipitate a flood of Diameter signaling traffic, such as an avalanche restart. Failure of a Diameter proxy may also result in a large amount of signaling as connections and sessions are reestablished.

Subscriber Initiated Traffic Flood: Large gatherings of subscribers or events that result in many subscribers interacting with the network in close time proximity can result in Diameter signaling traffic floods. For example, the finale of a large fireworks show could be immediately followed by many subscribers posting messages, pictures, and videos concentrated on one portion of a network. Subscriber devices, such as smartphones, may use aggressive registration strategies that generate unusually high Diameter traffic loads.

DoS attacks: An attacker, wishing to disrupt service in the network, can cause a large amount of traffic to be launched at a target element. This can be done from a central source of traffic or through a distributed DoS attack. In all cases, the volume of traffic well exceeds the capacity of the element, sending the system into overload.

1.3. Effects of Overload

Modern Diameter networks, composed of application layer multi-node deployments of Diameter elements, may operate at very large transaction volumes. If a Diameter node becomes overloaded, or even worse, fails completely, a large number of messages may be lost very quickly. Even with redundant servers, many messages can be lost in the time it takes for failover to complete. While a Diameter client or agent should be able to retry such requests, an overloaded peer may cause a sudden large increase in the number of transaction transactions needing to be retried, rapidly filling local queues or otherwise contributing to local overload. Therefore Diameter devices need to be able to shed load before critical failures can occur.

1.4. Overload vs. Network Congestion

This document uses the term "overload" to refer to application-layer overload at Diameter nodes. This is distinct from "network congestion", that is, congestion that occurs at the lower networking layers that may impact the delivery of Diameter messages between nodes. This document recognizes that element overload and network congestion are interrelated, and that overload can contribute to network congestion and vice versa.

Network congestion issues are better handled by the transport protocols. Diameter uses TCP and SCTP, both of which include congestion management features. Analysis of whether those features are sufficient for transport level congestion between Diameter nodes, and any work to further mitigate network congestion is out of scope both for this document, and for the work proposed by this document.

1.5. Diameter Applications in a Broader Network

Most elements using Diameter applications do not use Diameter exclusively. It is important to realize that overload of an element can be caused by a number of factors that may be unrelated to the processing of Diameter or Diameter applications.

An element that doesn't use Diameter exclusively needs to be able to signal to Diameter peers that it is experiencing overload regardless of the cause of the overload, since the overload will affect that element's ability to process Diameter transactions. If the element communicates with protocols other than Diameter, it may also need to signal the overload situation on these protocols depending on its function and the architecture of the network and application it is providing services for. Whether that is necessary can only be decided within the context of that architecture and use cases. A mechanism for signaling overload with Diameter, which this specification details the requirements for, provides Diameter nodes the ability to signal their Diameter peers of overload, mitigating that part of the issue. Diameter nodes may need to use this, as well as other mechanisms, to solve their broader overload issues. Indicating overload on protocols other than Diameter is out of scope for this document, and for the work proposed by this document.

2. Overload Control Scenarios

Several Diameter deployment scenarios exist that may impact overload management. The following scenarios help motivate the requirements for an overload management mechanism.

These scenarios are by no means exhaustive, and are in general simplified for the sake of clarity. In particular, this document assumes for the sake of clarity that the client sends Diameter requests to the server, and the server sends responses to client, even though Diameter supports bidirectional applications. Each direction in such an application can be modeled separately.

In a large scale deployment, many of the nodes represented in these scenarios would be deployed as clusters of servers. This document assumes that such a cluster is responsible for managing its own

internal load balancing and overload management so that it appears as a single Diameter node. That is, other Diameter nodes can treat it as single, monolithic node for the purposes of overload management.

These scenarios do not illustrate the client application. As mentioned in Section 1, Diameter is not typically an end-user protocol; rather it is generally used in support of some other client application. These scenarios do not consider the impact of Diameter overload on the client application.

2.1. Peer to Peer Scenarios

This section describes Diameter peer-to-peer scenarios. That is, scenarios where a Diameter client talks directly with a Diameter server, without the use of a Diameter agent.

Figure 1 illustrates the simplest possible Diameter relationship. The client and server share a one-to-one peer-to-peer relationship. If the server becomes overloaded, either because the client exceeds the server's capacity, or because the server's capacity is reduced due to some resource dependency, the client needs to reduce the amount of Diameter traffic it sends to the server. Since the client cannot forward requests to another server, it must either queue requests until the server recovers, or itself become overloaded in the context of the client application and other protocols it may also use.

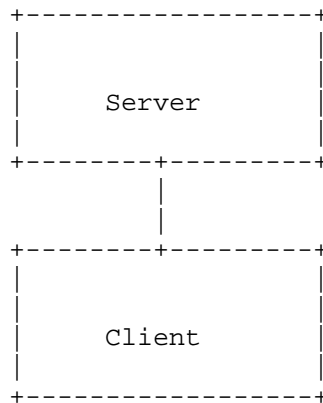


Figure 1: Basic Peer to Peer Scenario

Figure 2 shows a similar scenario, except in this case the client has multiple servers that can handle work for a specific realm and

application. If server 1 becomes overloaded, the client can forward traffic to server 2. Assuming server 2 has sufficient reserve capacity to handle the forwarded traffic, the client should be able to continue serving client application protocol users. If server 1 is approaching overload, but can still handle some number of new request, it needs to be able to instruct the client to forward a subset of its traffic to server 2.

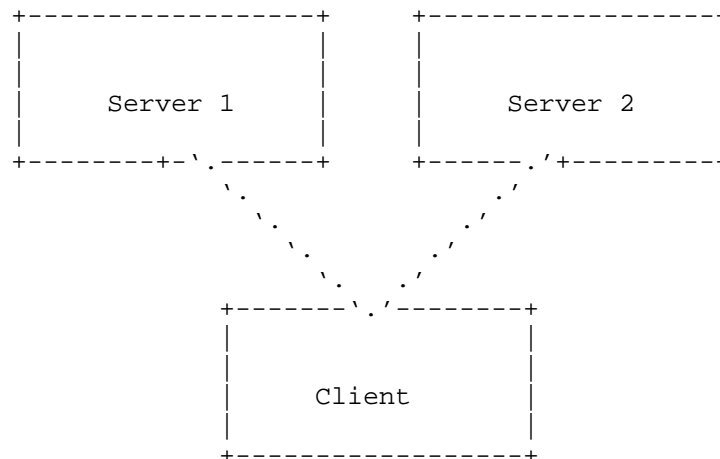


Figure 2: Multiple Server Peer to Peer Scenario

Figure 3 illustrates a peer-to-peer scenario with multiple Diameter realm and application combinations. In this example, server 2 can handle work for both applications. Each application might have different resource dependencies. For example, a server might need to access one database for application A, and another for application B. This creates a possibility that Server 2 could become overloaded for application A but not for application B, in which case the client would need to divert some part of its application A requests to server 1, but should not divert any application B requests. This requires server 2 to be able to distinguish between applications when it indicates an overload condition to the client.

On the other hand, it's possible that the servers host many applications. If server 2 becomes overloaded for all applications, it would be undesirable for it to have to notify the client separately for each application. Therefore it also needs a way to indicate that it is overloaded for all possible applications.

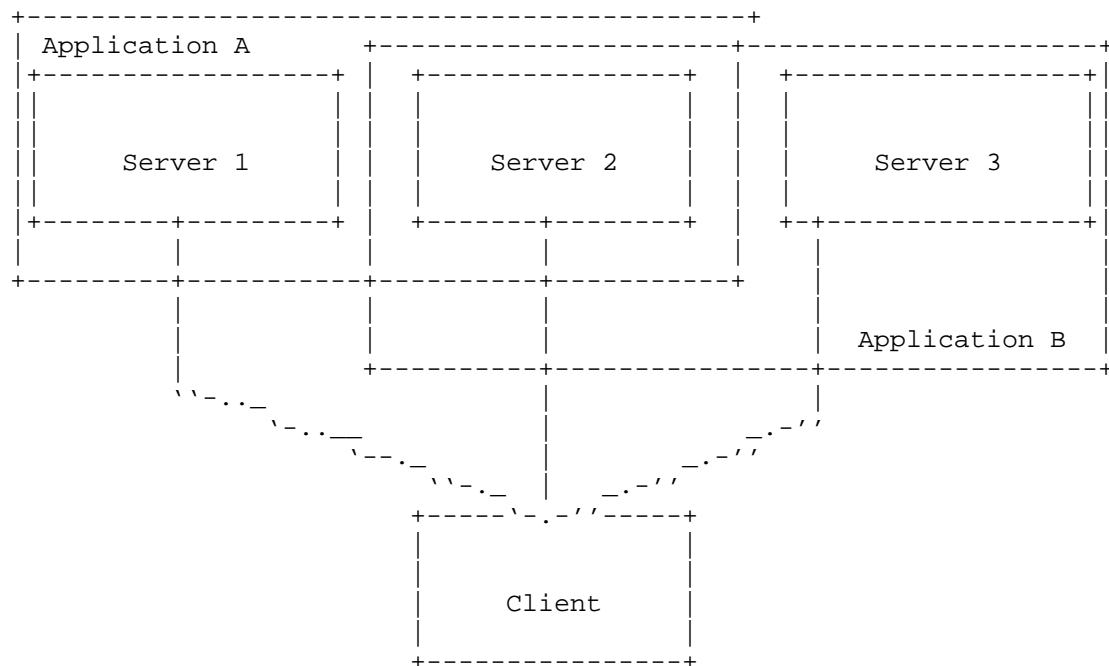


Figure 3: Multiple Application Peer to Peer Scenario

2.2. Agent Scenarios

This section describes scenarios that include a Diameter agent, either in the form of a Diameter relay or Diameter proxy. These scenarios do not consider Diameter redirect agents, since they are more readily modeled as end-servers. The examples have been kept simple deliberately, to illustrate basic concepts. Significantly more complicated topologies are possible with Diameter, including multiple intermediate agents in a path connected in a variety of ways.

Figure 4 illustrates a simple Diameter agent scenario with a single client, agent, and server. In this case, overload can occur at the server, at the agent, or both. But in most cases, client behavior is the same whether overload occurs at the server or at the agent. From the client's perspective, server overload and agent overload is the same thing.

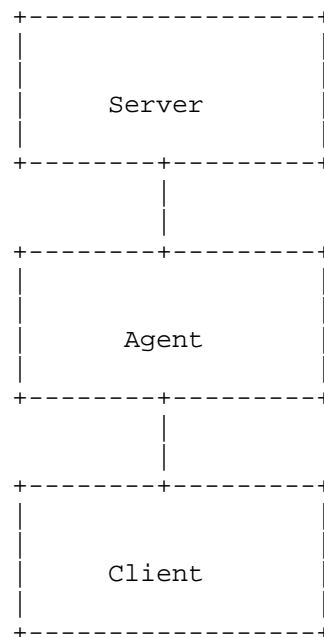


Figure 4: Basic Agent Scenario

Figure 5 shows an agent scenario with multiple servers. If server 1 becomes overloaded, but server 2 has sufficient reserve capacity, the agent may be able to transparently divert some or all Diameter requests originally bound for server 1 to server 2.

In most cases, the client does not have detailed knowledge of the Diameter topology upstream of the agent. If the agent uses dynamic discovery to find eligible servers, the set of eligible servers may not be enumerable from the perspective of the client. Therefore, in most cases the agent needs to deal with any upstream overload issues in a way that is transparent to the client. If one server notifies the agent that it has become overloaded, the notification should not be passed back to the client in a way that the client could mistakenly perceive the agent itself as being overloaded. If the set of all possible destinations upstream of the agent no longer has sufficient capacity for incoming load, the agent itself becomes effectively overloaded.

On the other hand, there are cases where the client needs to be able to select a particular server from behind an agent. For example, if a Diameter request is part of a multiple-round-trip authentication, or is otherwise part of a Diameter "session", it may have a

DestinationHost AVP that requires the request to be served by server 1. Therefore the agent may need to inform a client that a particular upstream server is overloaded or otherwise unavailable. Note that there can be many ways a server can be specified, which may have different implications (e.g. by IP address, by host name, etc).

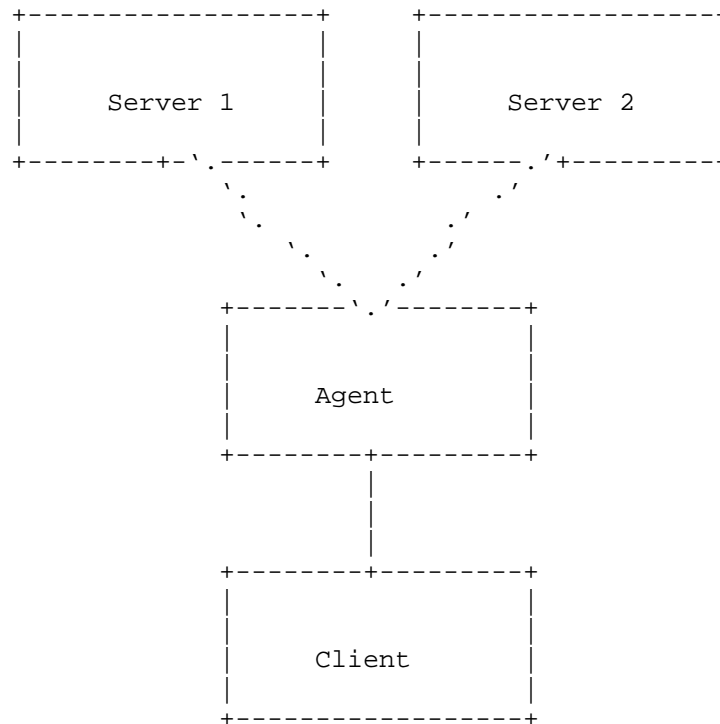


Figure 5: Multiple Server Agent Scenario

Figure 6 shows a scenario where an agent routes requests to a set of servers for more than one Diameter realm and application. In this scenario, if server 1 becomes overloaded or unavailable while server 2 still has available capacity, the agent may effectively operate at reduced capacity for application A, but at full capacity for application B. Therefore, the agent needs to be able to report that it is overloaded for one application, but not for another.

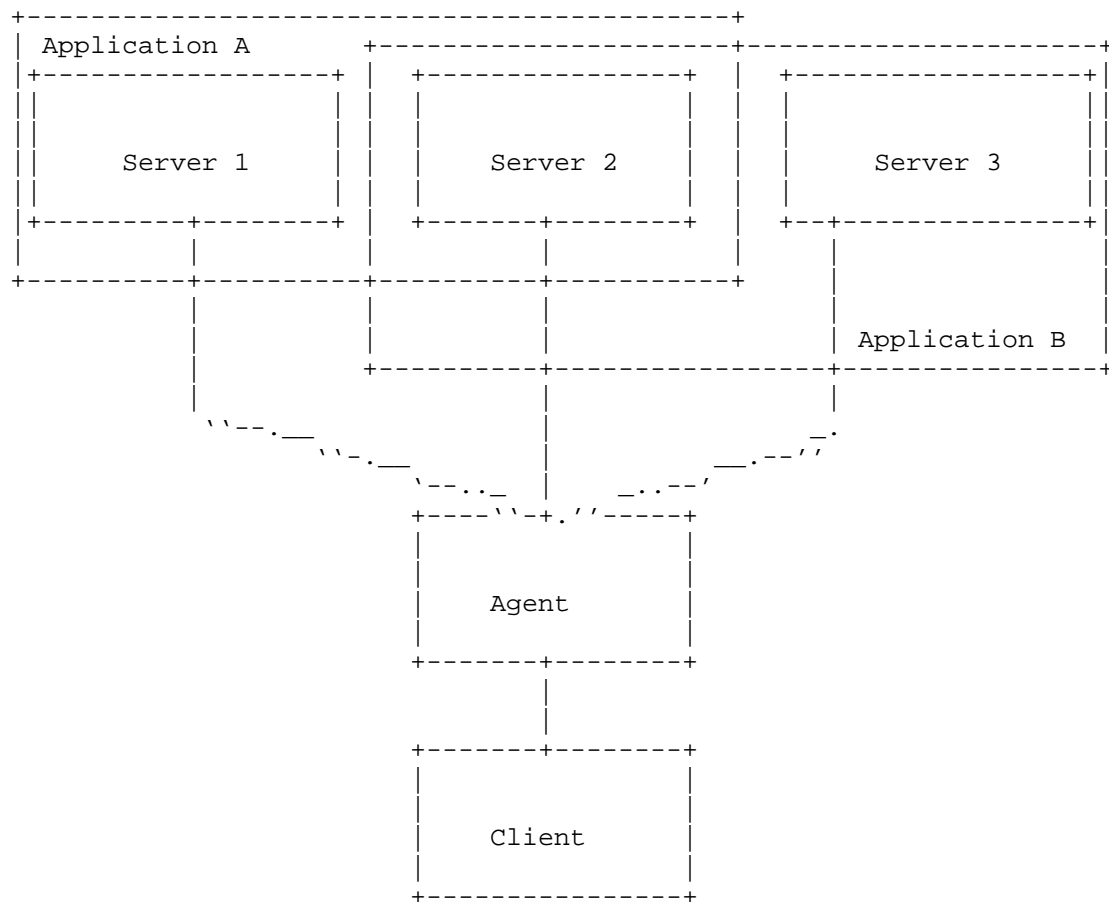


Figure 6: Multiple Application Agent Scenario

2.3. Interconnect Scenario

Another scenario to consider when looking at Diameter overload is that of multiple network operators using Diameter components connected through an interconnect service, e.g. using IPX. IPX (IP eXchange) [IR.34] is an Inter-Operator IP Backbone that provides roaming interconnection network between mobile operators and service providers. The IPX is also used to transport Diameter signaling between operators [IR.88]. Figure 7 shows two network operators with an interconnect network in-between. There could be any number of these networks between any two network operator's networks.

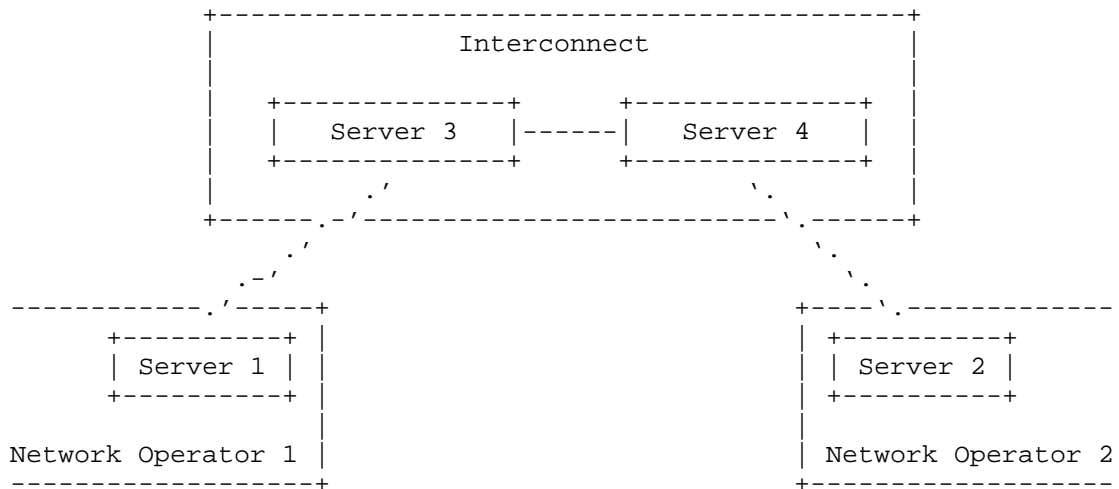


Figure 7: Two Network Interconnect Scenario

The characteristics of the information that an operator would want to share over such a connection are different from the information shared between components within a network operator's network. Network operators may not want to convey topology or operational information, which limits how much overload and loading information can be sent. For the interconnect scenario shown, Server 2 may want to signal overload to Server 1, to affect traffic coming from Network Operator 1.

This case is distinct from those internal to a network operator's network, where there may be many more elements in a more complicated topology. Also, the elements in the interconnect network may not support Diameter overload control, and the network operators may not want the interconnect network to use overload or loading information. They may only want the information to pass through the interconnect network without further processing or action by the interconnect network even if the elements in the interconnect network do support Diameter overload control.

3. Diameter Overload Case Studies

3.1. Overload in Mobile Data Networks

As the number of Third Generation (3G) and Long Term Evolution (LTE) enabled smartphone devices continue to expand in mobile networks, there have been situations where high signaling traffic load led to overload events at the Diameter-based Home Location Registries (HLR)

and/or Home Subscriber Servers (HSS) [TR23.843]. The root causes of the HLR overload events were manifold but included hardware failure and procedural errors. The result was high signaling traffic load on the HLR and HSS.

The 3GPP architecture [TS23.002] makes extensive use of Diameter. It is used for mobility management [TS29.272] (and others), (IP Multimedia Subsystem) IMS [TS29.228] (and others), policy and charging control [TS29.212] (and others) as well as other functions. The details of the architecture are out of scope for this document, but it is worth noting that there are quite a few Diameter applications, some with quite large amounts of Diameter signaling in deployed networks.

The 3GPP specifications do not currently address overload for Diameter applications or provide an equivalent load control mechanism to those provided in the more traditional SS7 elements in (Global System for Mobile Communications) GSM [TS29.002]. The capabilities specified in the 3GPP standards do not adequately address the abnormal condition where excessively high signaling traffic load situations are experienced.

Smartphones, an increasingly large percentage of mobile devices, contribute much more heavily, relative to non-smartphones, to the continuation of a registration surge due to their very aggressive registration algorithms. Smartphone behavior contributes to network loading and can contribute to overload conditions. The aggressive smartphone logic is designed to:

- a. always have voice and data registration, and
- b. constantly try to be on 3G or LTE data (and thus on 3G voice or VoLTE [IR.92]) for their added benefits.

Non-smartphones typically have logic to wait for a time period after registering successfully on voice and data.

The smartphone aggressive registration is problematic in two ways:

- o first by generating excessive signaling load towards the HSS that is ten times that from a non-smartphone,
- o and second by causing continual registration attempts when a network failure affects registrations through the 3G data network.

3.2. 3GPP Study on Core Network Overload

A study in 3GPP SA2 on core network overload has produced the technical report [TR23.843]. This enumerates several causes of overload in mobile core networks including portions that are signaled using Diameter. TR23.843 is a work in progress and is not complete. However, it is useful for pointing out scenarios and the general need for an overload control mechanism for Diameter.

It is common for mobile networks to employ more than one radio technology and to do so in an overlay fashion with multiple technologies present in the same location (such as 2nd or 3rd generation mobile technologies along with LTE). This presents opportunities for traffic storms when issues occur on one overlay and not another as all devices that had been on the overlay with issues switch. This causes a large amount of Diameter traffic as locations and policies are updated.

Another scenario called out by this study is a flood of registration and mobility management events caused by some element in the core network failing. This flood of traffic from end nodes falls under the network initiated traffic flood category. There is likely to also be traffic resulting directly from the component failure in this case. A similar flood can occur when elements or components recover as well.

Subscriber initiated traffic floods are also indicated in this study as an overload mechanism where a large number of mobile devices attempting to access services at the same time, such as in response to an entertainment event or a catastrophic event.

While this 3GPP study is concerned with the broader effects of these scenarios on wireless networks and their elements, they have implications specifically for Diameter signaling. One of the goals of this document is to provide guidance for a core mechanism that can be used to mitigate the scenarios called out by this study.

4. Existing Mechanisms

Diameter offers both implicit and explicit mechanisms for a Diameter node to learn that a peer is overloaded or unreachable. The implicit mechanism is simply the lack of responses to requests. If a client fails to receive a response in a certain time period, it assumes the upstream peer is unavailable, or overloaded to the point of effective unavailability. The watchdog mechanism [RFC3539] ensures that a certain rate of transaction responses occur even when there is otherwise little or no other Diameter traffic.

The explicit mechanism can involve specific protocol error responses, where an agent or server tells a downstream peer that it is either too busy to handle a request (`DIAMETER_TOO_BUSY`) or unable to route a request to an upstream destination (`DIAMETER_UNABLE_TO_DELIVER`), perhaps because that destination itself is overloaded to the point of unavailability.

Another explicit mechanism, a DPR (Disconnect-Peer-Request) message, can be sent with a Disconnect-Cause of `BUSY`. This signals the sender's intent to close the transport connection, and requests the client not to reconnect.

Once a Diameter node learns that an upstream peer has become overloaded via one of these mechanisms, it can then attempt to take action to reduce the load. This usually means forwarding traffic to an alternate destination, if available. If no alternate destination is available, the node must either reduce the number of messages it originates (in the case of a client) or inform the client to reduce traffic (in the case of an agent.)

Diameter requires the use of a congestion-managed transport layer, currently TCP or SCTP, to mitigate network congestion. It is expected that these transports manage network congestion and that issues with transport (e.g. congestion propagation and window management) are managed at that level. But even with a congestion-managed transport, a Diameter node can become overloaded at the Diameter protocol or application layers due to the causes described in Section 1.2 and congestion managed transports do not provide facilities (and are at the wrong level) to handle server overload. Transport level congestion management is also not sufficient to address overload in cases of multi-hop and multi-destination signaling.

5. Issues with the Current Mechanisms

The currently available Diameter mechanisms for indicating an overload condition are not adequate to avoid service outages due to overload. This inadequacy may, in turn, contribute to broader impacts resulting from overload due to unresponsive Diameter nodes causing application or transport layer retransmissions. In particular, they do not allow a Diameter agent or server to shed load as it approaches overload. At best, a node can only indicate that it needs to entirely stop receiving requests, i.e. that it has effectively failed. Even that is problematic due to the inability to indicate durational validity on the transient errors available in the base Diameter protocol. Diameter offers no mechanism to allow a node to indicate different overload states for different categories of

messages, for example, if it is overloaded for one Diameter application but not another.

5.1. Problems with Implicit Mechanism

The implicit mechanism doesn't allow an agent or server to inform the client of a problem until it is effectively too late to do anything about it. The client does not know to take action until the upstream node has effectively failed. A Diameter node has no opportunity to shed load early to avoid collapse in the first place.

Additionally, the implicit mechanism cannot distinguish between overload of a Diameter node and network congestion. Diameter treats the failure to receive an answer as a transport failure.

5.2. Problems with Explicit Mechanisms

The Diameter specification is ambiguous on how a client should handle receipt of a `DIAMETER_TOO_BUSY` response. The base specification [RFC6733] indicates that the sending client should attempt to send the request to a different peer. It makes no suggestion that the receipt of a `DIAMETER_TOO_BUSY` response should affect future Diameter messages in any way.

The Authentication, Authorization, and Accounting (AAA) Transport Profile [RFC3539] recommends that a AAA node that receives a "Busy" response failover all remaining requests to a different agent or server. But while the Diameter base specification explicitly depends on RFC3539 to define transport behavior, it does not refer to RFC3539 in the description of behavior on receipt of `DIAMETER_TOO_BUSY`. There's a strong likelihood that at least some implementations will continue to send Diameter requests to an upstream peer even after receiving a `DIAMETER_TOO_BUSY` error.

BCP 41 [RFC2914] describes, among other things, how end-to-end application behavior can help avoid congestion collapse. In particular, an application should avoid sending messages that will never be delivered or processed. The `DIAMETER_TOO_BUSY` behavior as described in the Diameter base specification fails at this, since if an upstream node becomes overloaded, a client attempts each request, and does not discover the need to failover the request until the initial attempt fails.

The situation is improved if implementations follow the [RFC3539] recommendation and keep state about upstream peer overload. But even then, the Diameter specification offers no guidance on how long a client should wait before retrying the overloaded destination. If an agent or server supports multiple realms and/or applications,

DIAMETER_TOO_BUSY offers no way to indicate that it is overloaded for one application but not another. A DIAMETER_TOO_BUSY error can only indicate overload at a "whole server" scope.

Agent processing of a DIAMETER_TOO_BUSY response is also problematic as described in the base specification. DIAMETER_TOO_BUSY is defined as a protocol error. If an agent receives a protocol error, it may either handle it locally or it may forward the response back towards the downstream peer. If a downstream peer receives the DIAMETER_TOO_BUSY response, it may stop sending all requests to the agent for some period of time, even though the agent may still be able to deliver requests to other upstream peers.

DIAMETER_UNABLE_TO_DELIVER, or using DPR with cause code BUSY also have no mechanisms for specifying the scope or cause of the failure, or the durational validity.

The issues with error responses in [RFC6733] extend beyond the particular issues for overload control and have been addressed in an ad hoc fashion by various implementations. Addressing these in a standard way would be a useful exercise, but it is beyond the scope of this document.

6. Extensibility and Application Independence

Given the variety of scenarios Diameter elements can be deployed in, and the variety of roles they can fulfill with Diameter and other technologies, a single algorithm for handling overload may not be sufficient. For purposes of this discussion, algorithm is inclusive of behavior for control of overload, but does not encompass the general mechanism or transport of control information. This effort cannot anticipate all possible future scenarios and roles. Extensibility, particularly of algorithms used to deal with overload, will be important to cover these cases.

Similarly, the scopes that overload information may apply to may include cases that have not yet been considered. Extensibility in this area will also be important.

The basic mechanism is intended to be application-independent, that is, a Diameter node can use it across any existing and future Diameter applications and expect reasonable results. Certain Diameter applications might, however, benefit from application-specific behavior over and above the mechanism's defaults. For example, an application specification might specify relative priorities of messages or selection of a specific overload control algorithm.

7. Solution Requirements

This section proposes requirements for an improved mechanism to control Diameter overload, with the goals of addressing the issues described in Section 5 and supporting the scenarios described in Section 2. These requirements are stated primarily in terms of individual node behavior to inform the design of the improved mechanism; solution designers should keep in mind that the overall goal is improved overall system behavior across all the nodes involved, not just improved behavior from specific individual nodes.

7.1. General

- REQ 1: The solution **MUST** provide a communication method for Diameter nodes to exchange load and overload information.
- 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.
- 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.
- 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.
- 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.
- 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 this knowledge to be configured at each node.

7.2. 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.
- REQ 8: Supporting nodes MUST be able to distinguish current overload information from stale information.
- 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.
- REQ 10: Consumers of overload information MUST be able to determine when the overload condition improves or ends.
- REQ 11: The solution MUST be able to operate in networks of different sizes.
- 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 this on other nodes in the network. This helps to prevent a small-scale failure from becoming a widespread outage.
- REQ 13: The solution MUST NOT introduce substantial additional work for 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.
- REQ 14: Some scenarios that result in overload involve a rapid increase of traffic with little time between normal levels and overload inducing levels. The solution SHOULD provide for rapid feedback when traffic levels increase.
- 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.

7.3. 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.
- REQ 17: In a mixed environment with nodes that support the solution and 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.
- REQ 18: In a mixed environment of nodes that support the solution and that do not, the solution MUST NOT preclude elements that support overload control from treating elements that do not support overload control in a 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 implementors for dealing with elements not supporting overload control.
- REQ 19: It MUST be possible to use the solution between nodes in different realms and in different administrative domains.
- REQ 20: Any explicit overload indication MUST be clearly distinguishable from other errors reported via Diameter.
- 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 was not in place.

7.4. 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.

- REQ 23: The solution MUST provide sufficient information to enable a load balancing node to divert messages that are rejected or otherwise throttled by an overloaded upstream node to other upstream nodes that are the most likely to have sufficient capacity to process them.
- REQ 24: The solution MUST provide a mechanism for indicating load levels even when not in an overloaded condition, to assist nodes making decisions to prevent overload conditions from occurring.

7.5. 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.
- 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.

7.6. 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 (Section 9)). 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.

- 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.
- 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.
- 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.

7.7. 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.
- REQ 32: The solution MUST provide a method for extending the information communicated and the algorithms used for overload control.

REQ 33: The solution MUST provide a default algorithm that 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.

8. IANA Considerations

This document makes no requests of IANA.

9. Security Considerations

A Diameter overload control mechanism is primarily concerned with the load and overload related behavior of nodes in a Diameter network, and the information used to affect that behavior. Load and overload information is shared between nodes and directly affects the behavior and thus is potentially vulnerable to a number of methods of attack.

Load and overload information may also be sensitive from both business and network protection viewpoints. Operators of Diameter equipment want to control visibility to load and overload information to keep it from being used for competitive intelligence or for targeting attacks. It is also important that the Diameter overload control mechanism not introduce any way in which any other information carried by Diameter is sent inappropriately.

Note that the Diameter base specification [RFC6733] lacks end to end security, making verifying the authenticity and ownership of load and overload information difficult for non-adjacent nodes. Authentication of load and overload information helps to alleviate several of the security issues listed in this section.

This document includes requirements intended to mitigate the effects of attacks and to protect the information used by the mechanism. This section discusses potential security considerations for overload control solutions. This discussion provides the motivation for several normative requirements described in Section 7. The discussion includes specific references to the normative requirements that apply for each issue.

9.1. Access Control

To control the visibility of load and overload information, sending should be subject to some form of authentication and authorization of

the receiver. It is also important to the receivers that they are confident the load and overload information they receive is from a legitimate source. REQ 28 requires the solution to work without assuming that all Diameter nodes in a network are trusted for the purposes of exchanging overload and load information. REQ 29 requires the solution to let nodes restrict unauthorized parties from seeing overload information. Note that this implies a certain amount of configurability on the nodes supporting the Diameter overload control mechanism.

9.2. Denial-of-Service Attacks

An overload control mechanism provides a very attractive target for denial-of-service attacks. A small number of messages may affect a large service disruption by falsely reporting overload conditions. Alternately, attacking servers nearing, or in, overload may also be facilitated by disrupting their overload indications, potentially preventing them from mitigating their overload condition.

A design goal for the Diameter overload control mechanism is to minimize or eliminate the possibility of using the mechanism for this type of attack. More strongly, REQ 27 forbids the solution from introducing new vulnerabilities to malicious attack. Additionally, REQ 30 stipulates that the solution not interfere with other mechanisms used for protection against denial of service attacks.

As the intent of some denial-of-service attacks is to induce overload conditions, an effective overload control mechanism should help to mitigate the effects of an such an attack.

9.3. Replay Attacks

An attacker that has managed to obtain some messages from the overload control mechanism may attempt to affect the behavior of nodes supporting the mechanism by sending those messages at potentially inopportune times. In addition to time shifting, replay attacks may send messages to other nodes as well (target shifting).

A design goal for the Diameter overload control solution is to minimize or eliminate the possibility of causing disruption by using a replay attack on the Diameter overload control mechanism. (Allowing a replay attack using the overload control solution would violate REQ 27.)

9.4. Man-in-the-Middle Attacks

By inserting themselves in between two nodes supporting the Diameter overload control mechanism, an attacker may potentially both access

and alter the information sent between those nodes. This can be used for information gathering for business intelligence and attack targeting, as well as direct attacks.

REQs 27, 28, and 29 imply a need to prevent man-in-the-middle attacks on the overload control solution. A transport using TLS and/or IPSEC may be desirable for this purpose.

9.5. Compromised Hosts

A compromised host that supports the Diameter overload control mechanism could be used for information gathering as well as for sending malicious information to any Diameter node that would normally accept information from it. While it is beyond the scope of the Diameter overload control mechanism to mitigate any operational interruption to the compromised host, REQs 28 and 29 imply a need to minimize the impact that a compromised host can have on other nodes through the use of the Diameter overload control mechanism. Of course, a compromised host could be used to cause damage in a number of other ways. This is out of scope for a Diameter overload control mechanism.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.
- [RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, June 2003.

10.2. Informative References

- [RFC5390] Rosenberg, J., "Requirements for Management of Overload in the Session Initiation Protocol", RFC 5390, December 2008.
- [RFC6357] Hilt, V., Noel, E., Shen, C., and A. Abdelal, "Design Considerations for Session Initiation Protocol (SIP) Overload Control", RFC 6357, August 2011.

- [TR23.843] 3GPP, "Study on Core Network Overload Solutions (Work in Progress)", TR 23.843 0.6.0, October 2012.
- [IR.34] GSMA, "Inter-Service Provider IP Backbone Guidelines", IR 34 7.0, January 2012.
- [IR.88] GSMA, "LTE Roaming Guidelines", IR 88 7.0, January 2012.
- [IR.92] GSMA, "IMS Profile for Voice and SMS", IR 92 7.0, March 2013.
- [TS23.002] 3GPP, "Network Architecture", TS 23.002 12.0.0, September 2012.
- [TS29.272] 3GPP, "Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol", TS 29.272 11.4.0, September 2012.
- [TS29.212] 3GPP, "Policy and Charging Control (PCC) over Gx/Sd reference point", TS 29.212 11.6.0, September 2012.
- [TS29.228] 3GPP, "IP Multimedia (IM) Subsystem Cx and Dx interfaces; Signalling flows and message contents", TS 29.228 11.5.0, September 2012.
- [TS29.002] 3GPP, "Mobile Application Part (MAP) specification", TS 29.002 11.4.0, September 2012.

Appendix A. Contributors

Significant contributions to this document were made by Adam Roach and Eric Noel.

Appendix B. Acknowledgements

Review of, and contributions to, this specification by Martin Dolly, Carolyn Johnson, Jianrong Wang, Imtiaz Shaikh, Jouni Korhonen, Robert Sparks, Dieter Jacobsohn, Janet Gunn, Jean-Jacques Trottin, Laurent Thiebaut, Andrew Booth, and Lionel Morand were most appreciated. We

would like to thank them for their time and expertise.

Authors' Addresses

Eric McMurry
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

Email: emcmurphy@computer.org

Ben Campbell
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

Email: ben@nostrum.com

Internet Engineering Task Force
Internet-Draft
Updates: 6733 (if approved)
Intended status: Standards Track
Expires: April 04, 2014

T. Tsou
Huawei Technologies (USA)
R. Hao
Comcast Cable
T. Taylor, Ed.
Huawei Technologies
October 01, 2013

Realm-Based Redirection In Diameter
draft-ietf-dime-realm-based-redirect-13

Abstract

The Diameter protocol includes a capability for message redirection, controlled by an application-independent "redirect agent". In some circumstances, an operator may wish to redirect messages to an alternate domain without specifying individual hosts. This document specifies an application-specific mechanism by which a Diameter server or proxy (node) can perform such a redirection when S-NAPTR is not used for dynamic peer discovery. A node performing this new function is referred to as a "Realm-based Redirect Server".

This memo updates Sections 6.13 and 6.14 of RFC6733 with respect to the usage of the Redirect-Host-Usage and Redirect-Max-Cache-Time AVPs.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 04, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Support of Realm-Based Redirection	3
Applications	3
3. Realm-Based Redirection	4
3.1. Configuration of the Realm-based Redirect Server	5
3.2. Behaviour of Diameter Nodes	5
3.2.1. Behaviour at the Realm-based Redirect Server	5
3.2.2. Proxy Behaviour	6
3.2.3. Client Behaviour	6
3.3. The Redirect-Realm AVP	7
3.4. DIAMETER_REALM_REDIRECT_INDICATION Protocol Error Code	7
4. Security Considerations	7
5. IANA Considerations	8
6. Acknowledgements	8
7. References	9
7.1. Normative References	9
7.2. Informative References	9
Authors' Addresses	9

1. Introduction

The Diameter base protocol [RFC6733] specifies a basic redirection service provided by redirect agent. The redirect indication returned by the redirect agent is described in Section 6.1.8 and Sections 6.12-6.14 of [RFC6733], and provides to the message sender one or more individual hosts as destination of the redirected message.

However, consider the case where an operator has offered a specific service but no longer wishes to do so. The operator has arranged for an alternative domain to provide the service. To aid in the transition to the new arrangement, the original operator maintains a redirect server to indicate to the message sender the alternative domain to which redirect the request. However, the original operator should be relieved from configuring in the redirect server a list of hosts to contact in the alternative operator's domain, and should

simply be able to provide redirect indications to the domain as a whole.

1.1. 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 [RFC2119].

Within this specification, the term "realm-based redirection" is used to refer to a mode of operation where a realm rather than an individual host is returned as redirect indication.

The term "Realm-based Redirect Server" denotes the Diameter node (Diameter server or proxy) that returns the realm-based redirection. The behaviour of the Realm-based Redirect Server itself is a slight modification of the behaviour of a basic redirect agent as described in Section 6.1.8 of [RFC6733].

This document uses a number of terms consistently with their usage in [RFC6733]: "Diameter client", "Diameter node", "Diameter peer", "Diameter server", "proxy", "realm" or "domain", "redirect agent", and "session" as defined in Section 1.2, and "application" as defined implicitly by Sections 1.3.4, 2.3, and 2.4.

2. Support of Realm-Based Redirection Within Applications

The DNS-based dynamic peer discovery mechanism defined in the Diameter base protocol [RFC6733] provides a simple mechanism for realm-based redirection, using the S-NAPTR DDDS application [RFC3958]. When S-NAPTR is used for peer discovery, redirection of Diameter requests from the original realm to a new realm may be performed by updating the existing NAPTR resource records for the original realm as follows: the NAPTR RR for the desired application(s) and supported application protocol(s) provided by the new realm will have an empty FLAG field and the REPLACEMENT field will contain the new realm to use for the next DNS lookup. The new realm can be arbitrary; the restriction in [RFC6733] that the NAPTR replacement field match the domain of the original query does not apply for realm-based redirect purposes.

However, the use of DNS-based dynamic peer discovery is optional for Diameter implementations. For deployments which do not make use of S-NAPTR peer discovery, support of realm-based redirection needs to be specified as part of functionality supported by a Diameter application. In this way, support of the considered Diameter application (discovered during capabilities exchange procedures as defined in Diameter base protocol [RFC6733]) indicates implicit

support of the realm-based redirection mechanism. A new application specification can incorporate the mechanism specified here by making it mandatory to implement for the application, and referencing this specification normatively.

The result of making realm-based redirection an application-specific behaviour is that it cannot be performed by a redirect agent as defined in [RFC6733], but **MUST** be performed instead by an application-aware Diameter node (Diameter server or proxy) (hereafter called a "Realm-based Redirect Server").

An application can specify that realm-based redirection operates only on complete sessions beginning with the initial message, or on every message within the application, even if earlier messages of the same session were not redirected. This distinction matters only when realm-based redirection is first initiated. In the former case, existing sessions will not be disrupted by the deployment of realm-based redirection. In the latter case, existing sessions will be disrupted if they are stateful.

3. Realm-Based Redirection

This section specifies an extension of the Diameter base protocol [RFC6733] to achieve realm-based redirection. The elements of this solution are:

- o a new result code, `DIAMETER_REALM_REDIRECT_INDICATION` (3xxx TBD1);
- o a new attribute-value pair (AVP), `Redirect-Realm` (code TBD2); and
- o associated behaviour at Diameter nodes implementing this specification.

This behaviour includes the optional use of the `Redirect-Host-Usage` and `Redirect-Max-Cache-Time` AVPs. In this document, these AVPs apply to the peer discovered by a node acting on the redirect server's response, an extension to their normal usage as described in Sections 6.13 and 6.14 of [RFC6733].

Section 3.2.2 and Section 3.2.3 describe how a proxy or client may update its routing table for the application and initial realm, as a result of selecting a peer in the new realm after realm-based redirection. Note that as a result, the proxy or client will automatically route subsequent requests for that application to the new realm (with the possible exception of requests within sessions already established with the initial realm) until the cached routing entry expires. This should be borne in mind if the rerouting is intended to be temporary.

3.1. Configuration of the Realm-based Redirect Server

A Diameter node (Diameter server or proxy) acting as Realm-based Redirect Server MUST be configured as follows to execute realm-based redirection:

- o configured with an application that incorporates realm-based redirection;
- o the Local Action field of the routing table described in Section 2.7 of [RFC6733] is set to LOCAL;
- o an application-specific field is set to indicate that the required local action is to perform realm-based redirection;
- o an associated application-specific field is configured with the identities of one or more realms to which the request should be redirected.

3.2. Behaviour of Diameter Nodes

3.2.1. Behaviour at the Realm-based Redirect Server

As mentioned in Section 2, an application can specify that realm-based redirection operates only on complete sessions beginning with the initial message (i.e., to prevent disruption of established sessions), or on every message within the application, even if earlier messages of the same session were not redirected.

If a Realm-based Redirect Server configured as described in Section 3.1 receives a request to which realm-based redirection applies, the Realm-based Redirect Server MUST reply with an answer message with the 'E' bit set, while maintaining the Hop-by-Hop Identifier in the header. The Realm-based Redirect Server MUST include the Result-Code AVP set to `DIAMETER_REALM_REDIRECT_INDICATION`. The Realm-based Redirect Server MUST also include the alternate realm identifier(s) with which it has been configured, each in a separate Redirect-Realm AVP instance.

The Realm-based Redirect Server MAY include a copy of the Redirect-Host-Usage AVP, which SHOULD be set to `REALM_AND_APPLICATION`. If this AVP is added, the Redirect-Max-Cache-Time AVP MUST also be included. Note that these AVPs apply to the peer discovered by a node acting on the Realm-based Redirect Server's response, as described in the next section. This is an extension of their normal usage as described by Sections 6.13 and 6.14 of [RFC6733].

Realm-based redirection MAY be applied even if a Destination-Host AVP is present in the request, depending on the operator-based policy.

3.2.2. Proxy Behaviour

A proxy conforming to this specification that receives an answer message with the Result-Code AVP set to `DIAMETER_REALM_REDIRECT_INDICATION` MUST attempt to reroute the original request to a server in a realm identified by a Redirect-Realm AVP instance in the answer message, and if it fails MUST forward the indication toward the client. To reroute the request, it MUST take the following actions:

1. Select a specific realm from amongst those identified in instances of the Redirect-Realm AVP in the answer message.
2. If successful, locate and establish a route to a peer in the realm given by the Redirect-Realm AVP, using normal discovery procedures as described in Section 5.2 of [RFC6733].
3. If again successful:
 - a. update its cache of routing entries for the realm and application to which the original request was directed, taking into account the Redirect-Host-Usage and Redirect-Max-Cache-Time AVPs, if present in the answer.
 - b. Remove the Destination-Host (if present) and Destination-Realm AVPs from the original request and add a new Destination-Realm AVP containing the realm selected in the initial step.
 - c. Forward the modified request.
4. If either of the preceding steps 2-3 fail and additional realms have been identified in the original answer, select another instance of the Redirect-Realm AVP in that answer and repeat steps 2-3 for the realm that it identifies.

3.2.3. Client Behaviour

A client conforming to this specification MUST be prepared to receive either an answer message containing a Result-Code AVP set to `DIAMETER_REALM_REDIRECT_INDICATION`, or, as the result of proxy action, some other result from a realm differing from the one to which it sent the original request. In the case where it receives `DIAMETER_REALM_REDIRECT_INDICATION`, the client SHOULD follow the same

steps prescribed in the previous section for a proxy, in order both to update its routing table and to obtain service for the original request.

3.3. The Redirect-Realm AVP

The Redirect-Realm AVP (code TBD2) is of type DiameterIdentity. It specifies a realm to which a node receiving a redirect indication containing the result code value `DIAMETER_REALM_REDIRECT_INDICATION` and the Redirect-Realm AVP SHOULD route the original request.

3.4. `DIAMETER_REALM_REDIRECT_INDICATION` Protocol Error Code

The `DIAMETER_REALM_REDIRECT_INDICATION` (3xxx TBD1) Protocol error code indicates that a server has determined that the request within an application supporting realm-based redirection could not be satisfied locally and the initiator of the request SHOULD direct the request directly to a peer within a realm that has been identified in the response. When set, the Redirect-Realm AVP MUST be present.

4. Security Considerations

The general recommendations given in the section 13 of the Diameter base protocol [RFC6733] apply. Specific security recommendations related to the realm-based redirection defined in this document are described below.

Realm-based redirection implies a change of the business relationships between organizations. Before redirecting a request towards a realm different from the initial realm, the client or proxy MUST ensure that the authorization checks have been performed at each connection along the path toward the realm identified in the realm-based redirect indication. Details on Diameter authorization path set-up are given in section 2.9 of [RFC6733]. Section 13 of [RFC6733] provides recommendations on how to authenticate and secure each peer-to-peer connection (using on TLS, DTLS or IPsec) along the way, thus permitting the necessary hop-by-hop authorization checks.

Although it is assumed that the administrative domains are secure, a compromised Diameter node acting as Realm-Based Redirect Server would be able to redirect a large number of Diameter requests towards a victim domain which would then be flooded with undesired Diameter requests. Such an attack is nevertheless discouraged by the use of secure Diameter peer-to-peer connections and authorization checks, since these would enable a potential victim domain to discover from where an attack is coming. That in itself, however, does not prevent such a DoS attack.

Because realm-based redirection defined in this document implies that the Destination-Realm AVP in a client-initiated request can be changed by a Diameter proxy in the path between the client and the server, any cryptographic algorithm that would use the Destination-Realm AVP as input to the calculation performed by the client and the server would be broken by this form of redirection. Application specifications that would rely on such cryptographic algorithm SHOULD NOT incorporate this realm-based redirection.

5. IANA Considerations

This specification adds a new AVP code [TBD2] Redirect-Realm in the AVP Code registry under Authentication, Authorization, and Accounting (AAA) Parameters.

This specification allocates a new Result-Code value `DIAMETER_REALM_REDIRECT_INDICATION` (3xxx TBD1) in the Result-Code AVP Values (code 268) - Protocol Errors registry under Authentication, Authorization, and Accounting (AAA) Parameters.

6. Acknowledgements

Glen Zorn, Sebastien Decugis, Wolfgang Steigerwald, Mark Jones, Victor Fajardo, Jouni Korhonen, Avi Lior, and Lionel Morand contributed comments that helped to shape this document. As shepherd, Lionel contributed a second set of comments that added polish to the document before it was submitted to the IESG. Benoit Claise picked up additional points which were quickly resolved with Lionel's help. During IETF Last Call Review, Enrico Marocco picked up some important editorial corrections. Stefan Winter contributed text on the use of S-NAPTR as an alternative method of realm-based redirection already specified in [RFC6733]. Derek Atkins performed a review on behalf of the Security Directorate. Lionel noted one more correction.

Finally, this document benefited from comments and discussion raised by IESG members Stewart Bryant, Stephen Farrell, Barry Leiba, Pete Resnick, Jaari Arkko, and Sean Turner during IESG review.

The authors are very grateful to Lionel Morand for his active role as document shepherd. At each stage, he worked to summarize and resolve comments, making the editor's role easy. Thank you.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

7.2. Informative References

- [RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.

Authors' Addresses

Tina Tsou
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1 408 330 4424
Email: Tina.Tsou.Zouting@huawei.com
URI: <http://tinatsou.weebly.com/contact.html>

Ruibing Hao
Comcast Cable
One Comcast Center
Philadelphia, PA 19103
USA

Phone: +1 215 286 3991(O)
Email: Ruibing_Hao@cable.comcast.com

Tom Taylor (editor)
Huawei Technologies
Ottawa
Canada

Email: tom.taylor.stds@gmail.com

DIME
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2016

J. Korhonen
Broadcom Ltd.
H. Tschofenig
ARM Ltd.
February 29, 2016

Diameter AVP Level Security: Keyed Message Digests, Digital Signatures,
and Encryption
draft-korhonen-dime-e2e-security-03.txt

Abstract

This document defines an extension for end to end authentication, integrity and confidentiality protection of Diameter Attribute Value Pairs. The solutions focuses on protecting Diameter Attribute Value Pairs and leaves the key distribution solution to a separate specification. The integrity protection can be introduced in a backward compatible manner to existing application. The confidentiality protection requires an explicit support from an application, thus is applicable only for newly defined applications.

Terminology

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 1, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Solution description	4
2.1. Integrity protection of AVPs	4
2.2. Confidentiality protection of AVPs	7
2.3. Definition of the 'End Point'	8
3. AVP Encoding	8
3.1. Signed-Data AVP	8
3.2. JWS-Header AVP	8
3.3. Header-Parameters AVP	8
3.4. JWS-AVP-Payload AVP	9
3.5. JWS-Signature AVP	9
3.6. Encrypted-Data AVP	9
3.7. JWE-Header AVP	9
3.8. JWE-Enc-Key AVP	10
3.9. JWE-Init-Vec AVP	10
3.10. JWE-AVP-Ciphertext AVP	10
4. Result-Code AVP Values	10
4.1. Transient Failures	10
4.2. Permanent Failures	11
5. IANA Considerations	11
6. Security Considerations	11
7. Acknowledgements	12
8. References	12
8.1. Normative References	12
8.2. Informational References	12
Authors' Addresses	13

1. Introduction

The Diameter base protocol [RFC6733] leverages IPsec and TLS for mutual authentication between neighboring Diameter nodes and for channel security offering data origin authentication, integrity and confidentiality protection. The Diameter base protocol, however, also defines Diameter agents, namely Relay Agents, Proxy Agents, Redirect Agents, and Translation Agents.

Relay Agents are Diameter agents that accept requests and route messages to other Diameter nodes based on information found in the messages. Since Relays do not perform any application level processing, they provide relaying services for all Diameter applications.

Similarly to Relays, Proxy Agents route Diameter messages using the Diameter routing table. However, they differ since they modify messages to implement policy enforcement.

Redirect Agents do not relay messages, and only return an answer with the information necessary for Diameter agents to communicate directly, they do not modify messages. Redirect Agents do not have negative impacts on end-to-end security and are therefore not considered in this document.

A Translation Agent is a device that provides translation between two protocols. To offer end-to-end security across different protocol requires the ability to convey and process the AVPs defined in this document by both end points. Since such support is very likely not available this document does not cover this functionality.

The Diameter extension defined in this document specifies how AVP authentication, integrity and confidentiality protection can be offered using either symmetric or asymmetric cryptography. As a solution mechanism is derived from Javascript Object Signing and Encryption (JOSE). JOSE offers a simple encoding with small set of features ideal for the purpose of Diameter. This document further defines a binary efficient coding of JOSE objects.

This document focuses on protecting Diameter AVP and leaves the key distribution solution to a separate specification, which most likely is going to be a specific key exchange application. To offer the functionality two grouped AVPs are defined: Signed-Data and Encrypted-Data. The respective JOSE objects are transported within these two AVPs.

2. Solution description

2.1. Integrity protection of AVPs

JWS represents digitally signed or HMACed content using JSON data structures. The representation in [I-D.ietf-jose-json-web-signature] consists of three parts: the JWS Header, the JWS Payload, and the JWS Signature. The three parts are represented as the concatenation of the encoded strings in that order, with the three strings being separated by period ('.') characters. For the JWS Payload one would define a new JSON object that contains an array of AVP code number and a hash of AVP pairs. The JWS Signature then covers the all APVs to be signed or HMACed. Both JWS Payload and signature MUST use the same hash algorithm of the cryptographic algorithm indicated in the JWS Header.

Although the solution relies on the JSON, the encoding into Diameter AVPs differ from the text based encoding of the JSON objects. Specifically, none of the JWS Header, JWS Payload or JWS Signature are not BASE64 encoded but are processed in their plaintext or binary representation formats. For example, the JWS Header is encoded in its plaintext format into the Header-Parameters AVP:

```
{  "typ": "JWT",
  "alg": "HS256",
  "kid": "abc123"
}
```

The JWS Payload and the JWS Signature hashes and AVP Code values are encoded in their binary format as octets, not in textual or BASE64 encoded formats. Sections 3.4 and 3.5 describe the encodings of the needed AVPs.

To package a set of AVPs for signing, each AVP octet representation to be protected are first individually hashed and encoded into the "JSON object" with its four octets AVP code number. The entire AVP MUST be input to the hash calculation, from the first byte of the AVP code to the last byte of the AVP data, including all other fields, length, reserved/flags, and optional vendor IDs, and padding. The AVP MUST be input to the hash calculation in network byte order.

The JWS Signature is calculated over the entire JWS Payloads and then the all three JWS parts are placed in the Signed-Data AVP. There can be multiple Signed-Data AVPs in a Diameter message. The AVP code in the JWS Payload is to indicate which AVP this hash possibly refers to. If there are multiple instances of the same AVP in the Diameter message, there is no other way than make the verification against all of those. It is possible that the message sender only hashed one AVP

of the same type and, therefore, the receiver MUST verify the hash against all occurrences of the AVP of the same code number. Such flexibility is added there to allow reordering of the AVPs and addition or deletion of new AVPs by intermediating agents.

If a receiver detects errors with the processing of the Signed-Data AVP it MAY return one of the errors defined in Section 4. If a receiver does not find any AVP the Signed-Data AVP has a signature for, it MAY also return one of the errors defined in Section 4.

When AVPs are to be both encrypted and signed, the Encrypted-Data AVP MUST be created first. This means that signing is "outside" encryption.

Here is an example: Imagine the following AVPs from the QoS-Resources AVP in the QoS-Install Request (defined in RFC 5866 [RFC5866] message shall be signed. The resulting example message has the following structure:

```
<QoS-Install-Request> ::= < Diameter Header: 327, REQ, PXY >
                           < Session-Id >
                           { Auth-Application-Id }
                           { Origin-Host }
                           { Origin-Realm }
                           { Destination-Realm }
                           { Auth-Request-Type }
                           [ Signed-Data ]
                           * [ QoS-Resources ]
                           ...
```

Example Diameter Message with Signed-Data AVP

The Signed-Data AVP in this example may contain a JWS Header that indicates the use of the HMAC SHA-256 algorithm with the key id 'abc123'. The protected AVPs are Session-Id, Origin-Host and Origin-Realm. The calculated HMAC SHA-256 values are for example purposes only (i.e., are not real):

JWS Header encoded as such in JWS-Header AVP:

```
{ "typ": "JWT",
  "alg": "HS256",
  "kid": "abc123"
}
```

```
0x00000xxx // JWS-Header code 'xxx'
0x00000034 // Flags=0, Length=52
'{"typ": "JWT", "alg": "HS256", "kid": "abc123"}' // 41
0x00,0x00,0x00 // 3 octets padding
```

JWS Payload encoded into three JWS-AVP-Payload AVPs:

```
0x00000zzz // JWS-AVP-Payload code 'zzz' <--+
0x0000001c // Flags=0, Length=28 |
0x00000107 // 263, Session-Id, 4 octets s
0x9d0e0495 // hash of Session-Id, 128 bits i
0xba8c0312 g
0xb6274c52 n
0x7d51a048 a
t
0x00000zzz // JWS-AVP-Payload code 'zzz' u
0x0000001c // Flags=0, Length=28 r
0x00000108 // 264, Origin-Host, 4 octets e
0x39ca88ff // hash of Origin-Host, 128 bits |
0xaa5a6ff9 c
0x029ed95b o
0xa534e028 v
e
0x00000zzz // JWS-AVP-Payload code 'zzz' r
0x0000001c // Flags=0, Length=28 a
0x00000128 // 296, Origin-Realm, 4 octets g
0x202730ac // hash of Origin-Realm, 128 bits e
0xa6e3a180 |
0x2f44a633 |
0xf250f6fe <--+
```

JWS Signature encoded into the JWS-Signature AVP:

```
0x00000yyy // JWS-Signature code 'yyy'
0x00000018 // Flags=0, Length=24
0xaabbccdd,0xdddeeff0,0x11223344,0x55667788
```

Example JWS Header, Payload and Signature

2.2. Confidentiality protection of AVPs

The Encrypted-Data AVP (AVP Code TBD) is of type OctetString and contains the JSON Web Encryption (JWE) [I-D.ietf-jose-json-web-encryption] data structure and consists of four parts: the JWE Header, the JWE Encrypted Key, the JWE Initialization Vector and the JWE Ciphertext. The four parts are represented as the concatenation of the encoded strings in that order, with the three strings being separated by period ('.') characters. JWE does not add a content integrity check if not provided by the underlying encryption algorithm.

Although the solution relies on the JSON, the encoding into Diameter AVPs differ from the text based encoding of the JSON objects. Specifically, none of the the JWE Header, the JWE Encrypted Key, the JWE Initialization Vector and the JWE Ciphertext are not BASE64 encoded but are processed in their plaintext or binary representation formats. The concept follows what was already described in Section 2.1.

A single AVP or an entire list of AVPs MUST be input to the encryption process, from the first byte of the AVP code to the last byte of the AVP data, including all other fields, length, reserved/flags, and optional vendor IDs, and padding. The AVP MUST be input to the encryption process in network byte order, and the encryptor is free to order AVPs whatever way it chooses. When AVPs are to be both encrypted and authenticated, the Encrypted-Data AVP MUST be created first.

Note that the usage of the Encrypted-Data AVP requires explicit support by the Diameter application since a receiving Diameter node must first decrypt the content of the Encrypted-Data AVP in order to evaluate the AVPs carried in the message. In case that a Diameter node is unable to understand the Encrypted-Data AVP and ignores the AVP then two possible outcomes are possible: First, if the encrypted AVPs are optional then their content is not considered by the receiving Diameter server without any indication to the sender that they have not been processed. Worse, in the second case when the encrypted AVPs are mandatory to be processed then the receiving Diameter node will return an error that may not inform the sender about the failure to decrypt the Encrypted-Data AVP. Consequently, the usage of the Encrypted-Data AVP may require changes to the ABNF definition of a Diameter application.

If a receiver detects that the contents of the Encrypted-Data AVP is invalid, it SHOULD return the new Result-Code AVP value defined in Section 4.

2.3. Definition of the 'End Point'

Although this specification claims to introduce the end-to-end security into Diameter, the definition who actually is the 'end point' is not obvious. The 'end point' does not need to be the original Diameter request or answer originator but the Diameter node that inserts the Signed-Data or the Encrypted-Data AVPs into the Diameter message. The node can be the request or answer originator or a proxy agent. Use of proxy agents doing the 'end-to-end' security on behalf of other nodes mimics the deployments where site-to-site VPNs are used.

3. AVP Encoding

3.1. Signed-Data AVP

The Signed-Data AVP (AVP Code TBD1) is of type Grouped and utilizes the JSON Web signature (JWS) mechanism defined in [I-D.ietf-jose-json-web-signature]. The JWS payload is then encoded into the Signed-Data AVP:

```
Signed-Data ::= < AVP Header: TBD1 >
               { JWS-Header }
               * { JWS-AVP-Payload }
               { JWS-Signature }
               * [ AVP ]
```

3.2. JWS-Header AVP

The JWS-Header AVP (AVP Code TBD2) is of type UTF8String and contains the JSON Web Signature Header. The contents of the AVP follow the rules for the header found in [I-D.ietf-jose-json-web-signature], which implies the required IANA registries are also defined by JSON documents.

```
JWS-Header ::= < AVP Header: TBD2 >
               { Header-Parameters }
               * [ AVP ]
```

The "alg" is the only REQUIRED Header Parameter for the signature purposes. The "typ" and "kid" Header Parameters are also RECOMMENDED.

3.3. Header-Parameters AVP

The Header-Parameters AVP (AVP Code TBD3) is of type UTF8String and contains the JSON Header Parameter Name and its value as described in [I-D.ietf-jose-json-web-signature]. The encoding (textual) also

follows [I-D.ietf-jose-json-web-signature]. Differing from the JSON specifications the parameter names and values are not BASE64 encoded but in their original UTF-8 representation format.

3.4. JWS-AVP-Payload AVP

The JWS-AVP-Payload AVP (AVP Code TBD4) is of type OctetString and contains both an AVP Code and a hash of the entire AVP identified by the AVP Code. The first four octets contain the AVP Code in a network byte order followed by the hash octets. The length of the hash octets depends on the used hash algorithm.

3.5. JWS-Signature AVP

The JWS-Signature AVP (AVP Code TBD5) is of type OctetString and contains the signature calculated over the array of complete JWS-AVP-Payload AVPs (including AVP header fields etc) in the order they appear in the Signed-Data AVP. The length of the signature octets depends on the used signature algorithm.

3.6. Encrypted-Data AVP

The Encrypted-Data AVP (AVP Code TBD6) is of type Grouped and utilizes the JSON Web Encryption (JWE) mechanism defined in [I-D.ietf-jose-json-web-encryption]. The JWE payload is then encoded into the Encrypted-Data AVP:

```
Encrypted-Data ::= < AVP Header: TBD1 >
                { JWE-Header }
                { JWE-Enc-Key }
                [ JWE-Init-Vec ]
                { JWE-AVP-Ciphertext }
                * [ AVP ]
```

3.7. JWE-Header AVP

The JWE-Header AVP (AVP Code TBD7) is of type UTF8String and contains the JSON Web Encryption Header. The contents of the AVP follow the rules for the header found in [I-D.ietf-jose-json-web-encryption], which implies the required IANA registries are also defined by JSON documents.

```
JWE-Header ::= < AVP Header: TBD7 >
              { Header-Parameters }
              * [ AVP ]
```

The "alg" and "enc" are the REQUIRED Header Parameter for the encryption purposes. The "typ" and "kid" Header Parameters are also RECOMMENDED.

3.8. JWE-Enc-Key AVP

The JWE-Enc-Key AVP (AVP Code TBD8) is of type OctetString and contains the JWE Encrypted Key in its binary format.

3.9. JWE-Init-Vec AVP

The JWE-Init-Vec AVP (AVP Code TBD9) is of type OctetString and contains the JWE Initialization Vector in its binary format.

3.10. JWE-AVP-Ciphertext AVP

The JWE-AVP-Ciphertext AVP (AVP Code TBD10) is of type OctetString and contains the encrypted AVPs. The encrypted AVPs are first concatenated into one large plaintext octet blob and then encrypted as a whole. The length of the ciphertext depends on the used algorithm and encrypted AVPs. The plaintext to be encrypted is never BASE64 encoded but MAY be compressed if a "zip" parameter was included in the JWE Header.

4. Result-Code AVP Values

This section defines new Diameter result code values for usage with Diameter applications.

4.1. Transient Failures

Errors that fall within the transient failures category are used to inform a peer that the request could not be satisfied at the time it was received, but MAY be able to satisfy the request in the future.

DIAMETER_KEY_UNKNOWN (TBD11)

This error code is returned when a Signed-Data or an Encrypted-Data AVP is received that was generated using a key that cannot be found in the key store. This error may, for example, be caused if one of the endpoints of an end-to-end security association lost a previously agreed upon key, perhaps as a result of a reboot. To recover a new end-to-end key establishment procedure may need to be invoked.

DIAMETER_HEADER_NAME_ERROR (TBD12)

This error code is returned when a Header Parameter Name is not understood in the JWS-Header AVP or in the JWE-Header AVP.

4.2. Permanent Failures

Errors that fall within the permanent failures category are used to inform the peer that the request failed, and should not be attempted again.

DIAMETER_DECRYPTION_ERROR (TBD13)

This error code is returned when an Encrypted-Data AVP is received and the decryption fails for an unknown reason.

DIAMETER_SIGNATURE_ERROR (TBD14)

This error code is returned when a Signed-Data AVP is received and the verification fails for an unknown reason.

5. IANA Considerations

IANA is requested to allocate AVP codes for the following AVPs:

AVP Name	AVP Code	Section Defined	Data Type
Signed-Data	TBD1	3.1	Grouped
JWS-Header	TBD2	3.x	Grouped
JWS-AVP-Payload	TBD3	3.x	OctetString
JWS-Signature	TBD4	3.x	OctetString
Header-Parameters	TBD5	3.x	UTF8String
Encrypted-Data	TBD6	3.x	Grouped
JWE-Header	TBD7	3.x	Grouped
JWE-Enc-Key	TBD8	3.x	OctetString
JWE-Init-Vec	TBD9	3.x	OctetString
JWE-AVP-Ciphertext	TBD10	3.x	OctetString

This specification additionally defines a few Result-Code AVP values, see Section 4.

6. Security Considerations

The purpose of this document is to offer end-to-end security mechanisms for calculating keyed message digest, for signing, and for encryption of Diameter AVPs.

An intermediate Diameter agent that for a reason or other reorders the AVPs within the Signed-Data AVP may cause the signature verification fail even if no AVP was actually tampered.

7. Acknowledgements

We would like to thank the authors of [I-D.ietf-aaa-diameter-e2e-sec] for their work on CMS end-to-end security for Diameter. Their document inspired us.

8. References

8.1. Normative References

- [I-D.ietf-jose-json-web-encryption]
Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption-40 (work in progress), January 2015.
- [I-D.ietf-jose-json-web-signature]
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", draft-ietf-jose-json-web-signature-41 (work in progress), January 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <<http://www.rfc-editor.org/info/rfc6733>>.

8.2. Informational References

- [I-D.ietf-aaa-diameter-e2e-sec]
Calhoun, P., "Diameter End-2-End Security Extension", 2001.
- [RFC5866] Sun, D., Ed., McCann, P., Tschofenig, H., Tsou, T., Doria, A., and G. Zorn, Ed., "Diameter Quality-of-Service Application", RFC 5866, DOI 10.17487/RFC5866, May 2010, <<http://www.rfc-editor.org/info/rfc5866>>.

Authors' Addresses

Jouni Korhonen
Broadcom Ltd.
3151 Zanker Road
CA 95134
US

Email: jouni.nospam@gmail.com

Hannes Tschofenig
ARM Ltd.

Email: Hannes.Tschofenig@gmx.de
URI: <http://www.tschofenig.priv.at>

Diameter Maintenance and Extensions
(DIME)
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

J. Korhonen
Renesas Mobile
H. Tschofenig, Ed.
Nokia Siemens Networks
February 25, 2013

The Diameter Overload Control Application (DOCA)
draft-korhonen-dime-ovl-01.txt

Abstract

This specification documents a Diameter Overload Control Application (DOCA), which uses the normal Diameter application approach for the capability negotiation, propagation and management of Diameter overload control information between Diameter nodes.

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 August 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements	3
3. DOCA Overview	4
4. DOCA Commands	5
5. Attribute Value Pairs	6
5.1. OC-Information AVP	6
5.2. OC-Scope AVP	7
5.3. OC-Applications AVP	8
5.4. OC-Action AVP	9
5.5. OC-Algorithm AVP	9
5.6. OC-Level AVP	10
5.7. OC-Utilization AVP	11
5.8. OC-Tocl AVP	11
5.9. OC-Sending-Rate AVP	11
5.10. OC-Best-Before AVP	12
5.11. OC-Origin AVP	12
5.12. OC-Priority AVP	12
5.13. Attribute Value Pair flag rules	13
6. Transport Considerations	13
7. Examples	14
8. IANA Considerations	15
8.1. Application Identifiers	15
8.2. SCTP Payload Protocol Identifier	15
8.3. Command Codes	15
8.4. AVP Codes	15
8.5. Result-Code Values	15
8.6. New Registries	16
9. Security Considerations	16
10. Acknowledgements	16
11. References	17
11.1. Normative References	17
11.2. Informative References	17
Appendix A. Design Justification	17
Authors' Addresses	18

1. Introduction

The existing toolbox offered by the Diameter Base Protocol [RFC6733] to prevent and recover from signaling overload situations is rather limited. Apart from out-of-band altering of the transport connection congestion control behavior or other non-standard application level throttling, the protocol error `DIAMETER_TOO_BUSY`, the permanent error `DIAMETER_UNABLE_TO_COMPLY` (for some unspecified reason) and the Disconnect-Cause Attribute Value Pair (AVP) code `BUSY` or `DO_NOT_WANT_TO_TALK_TO_YOU` are more or less all there is. Unfortunately, the mentioned three indications are coarse, concern one peer connection at a time or lack detailed information for problem diagnosis and mitigation. They also treat all applications in a single Diameter node (identified by a single `DiameterIdentity`) as a lump. There is no way to communicate any kind of grouping of applications or what is the scope/partitioning of the delivered information. Furthermore, there is no way to signal when the overload situation is over. The request initiator and forwarders are therefore forced to keep re-submitting their messages to determine whether the situation has changed.

The situation is further complicated by the hop-by-hop nature of Diameter deployments. This makes the propagation of possible overload situation information non-trivial, even for existing protocol errors since every intermediate Diameter node is allowed to react to the error situation. Either the information is never propagated to the originator of the request or it takes an unacceptable long time.

A problem statement of overload control for Diameter and requirements are documented in [I-D.ietf-dime-overload-reqs]. This specification describes a solution to the Diameter overload Diameter Overload Control Application (DOCA), which fulfills the requirements of [I-D.ietf-dime-overload-reqs] and defines a Diameter application to convey overload information between Diameter nodes.

Note: The recent publication of [I-D.campbell-dime-overload-data-analysis] illustrates the overload information data model and the design space. As the working group makes progress in deciding about specific features this document will be updated accordingly.

2. 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].

3. DOCA Overview

Any the DOCA capable Diameter node MAY initiate a DOCA-Report-Request at any given time. The receiver of the DOCA-Report-Request acknowledges with a DOCA-Report-Answer and includes the Result-Code AVP indicating whether it could honor the action/report in the request. The DOCA-Report-Answer SHOULD also piggyback overload control information.

A DOCA client MUST set the Auth-Session-State AVP to the value NO_STATE_MAINTAINED and SHOULD include the OC-Information AVP with overload information into the DOCA-Report-Request, if available. The DOCA-Report-Response message MUST contain the Auth-Session-State AVP set to value NO_STATE_MAINTAINED.

Note that information exchanges regarding various DOCA related timers serve only as a hint since they cannot be enforced. Consequently, care should be taken not to send DOCA-Report-Requests too frequently.

When a Diameter node receives overload control information and is also requested to act on it, the DOCA functionality is applied to all specified applications within a given scope. How the Diameter node accomplishes the node wide DOCA action enforcement is implementation specific.

When a Diameter node receives (interim) overload information but the overload condition has not exceeded a certain threshold, then the receiver is not required to act based on the received information. However, it is RECOMMENDED that the receiver makes proactive actions to avoid entering the overload condition based on the newly received overload information.

There may be zero or more intermediate Diameter agents on the path between the DOCA client and the DOCA server. Understanding the DOCA functionality is not expected from relays and redirect agents. A Diameter proxy, which obviously understands the DOCA application, MAY inspect the DOCA related AVPs in the DOCA-Report-Request/Answer message pair and depending on the value of the OC-Scope AVP (see Section 5.2) inject its own information. A proxy is always RECOMMENDED to react according to the overload information when it comes to, for example, peer selection and traffic throttling.

When a Diameter agent receives overload control information and is also requested to act on it, the DOCA functionality is applied to all specified applications within a given scope. How the Diameter agent accomplishes the node wide DOCA action enforcement is implementation specific.

4. DOCA Commands

The DOCA-Report-Request (DRR) message is used to report overload condition information. The message can be originated as a result of emerging overload condition or as a periodic unsolicited report.

```
<DOCA-Report-Request> ::= < Diameter Header: TBD2, REQ, PXY >
    < Session-Id >
    { Auth-Application-Id }
    { Origin-Host }
    { Origin-Realm }
    { Destination-Realm }
    { Auth-Request-Type }
    { Destination-Host }
    [ Auth-Session-State ]
    * [ Class ]
    [ Origin-State-Id ]
    * [ Proxy-Info ]
    * [ Route-Record ]

    { OC-Scope }
    [ OC-Algorithm ]
    [ OC-Action ]
    [ OC-Tocl ]
    [ OC-Applications ]
    * [ OC-Information ]

    * [ AVP ]
```

The DOCA-Report-Answer (DRA) message is used as a response to the DOCA-Report-Request. The message MAY piggyback overload condition information in order to avoid unnecessary DOCA-Report-Request messages to the reverse direction.


```
<DOCA-Report-Answer> ::= < Diameter Header: TBD2, PXY >
    < Session-Id >
    { Result-Code }
    { Origin-Host }
    { Origin-Realm }
    [ Auth-Session-State ]
    * [ Class ]
    [ Error-Message ]
    [ Error-Reporting-Host ]
    [ Failed-AVP ]
    [ Origin-State-Id ]
    * [ Redirect-Host ]
    [ Redirect-Host-Usage ]
    [ Redirect-Max-Cache-Time ]
    * [ Proxy-Info ]

    { OC-Scope }
    [ OC-Action ]
    * [ OC-Information ]

    * [ AVP ]
```

5. Attribute Value Pairs

5.1. OC-Information AVP

The OC-Information AVP (AVP Code TBD3) is of type Grouped and contains a set AVPs that identify the source of the overload control information (the OC-Origin AVP), the overload information itself and which applications the information concerns.

```
OC-Information ::= < AVP Header: TBD3 >
    { OC-Origin }
    { OC-Best-Before }
    [ OC-Level ]
    [ OC-Algorithm ]
    [ OC-Sending-Rate ]
    [ Vendor-Id ]
    [ OC-Applications ]
    [ Product-Name ]
    [ OC-Utilization ]
    [ OC-Priority ]
    * [ AVP ]
```

Diameter proxies on path MAY add one or more OC-Information AVPs into the DOCA-Report-Request/answer messages.

5.2. OC-Scope AVP

The OC-Scope (AVP Code TBD4) is of type Unsigned32 and contains the scope where and concerning what the overload control information can be injected. The OC-Scope is formatted as a vector of scope flag bits. The following scopes are supported:

Host scope (0x00000001)

The OC-Information AVP concerns only a single host within a realm (which internally MAY represent of pool).

Realm scope (0x00000002)

The OC-Information AVP concerns a realm. No specific hosts are identified.

Only origin realm (0x00000004)

The OC-Information AVP can only be included by a Diameter node on the path that has the same Origin-Realm as the DOCA client.

Application information (0x00010000)

The OC-Information AVP MAY contain application related information (the OC-Applications AVP).

Node utilization information (0x00020000)

The OC-Information AVP MAY contain node wide load related information (the OC-Utilization AVP).

Application priorities (0x00040000)

The OC-Information AVP SHOULD priority information (the OC-Priority AVP) so when the overload condition is on, Diameter nodes are able to prioritize between different applications, for example, when dropping or throttling messages.

Any other value is reserved.

A scope is active when a corresponding flag is set in the OC-Scope AVP. During the initialization state a DOCA client includes those scopes it supports and is interested in. A DOCA server then returns the scope that it has in common with the DOCA client (and intends to use). The common scopes are then used during the established state. Note that some scope combinations make little sense while still being valid. The general guide when multiple scopes collide is that the

least restrictive wins.

A sender of the overload information MUST adhere to the scope it announces regarding the information it itself sends.

If a DOCA server does not have a common scope with a DOCA client or the DOCA server cannot agree on one based on a local policy, then the DOCA server MUST send the DOCA-Report-Answer indicating an error and set the Result-Code to the DIAMETER_NO_COMMON_SCOPE value.

5.3. OC-Applications AVP

The OC-Applications (AVP Code TBD5) is of type Grouped and contains a list of Application-IDs of interest when found in the DOCA-Report-Request/Answer command main level and meant to be used during the initialization state to agree on the common set of supported applications of monitoring interest. When used within the OC-Information AVP, the OC-Applications AVP identify those applications the overload information concerns. The OC-Applications AVPs on the command main level and inside the OC-Information AVP MUST NOT have conflicting views of the applications of interest. However, the OC-Applications AVP can be seen as a superset of applications i.e., not all applications of interest need to be included every time into the OC-Information AVP.

```
OC-Applications ::= < AVP Header: TBD3 >
                  * [ Auth-Application-Id ]
                  * [ Acct-Application-Id ]
                  * [ Vendor-Specific-Application-Id ]
                  * [ AVP ]
```

The absence of the OC-Applications AVP indicates the Diameter node has no specific preference or interest in specific applications. The overload information is then signaled as concerning the whole Diameter node. This default behavior is useful when the DOCA does not maintain session state. If there are no common applications, then the DOCA-Report-Answer MUST contain the Result-Code with the DIAMETER_NO_COMMON_APPLICATION value.

When the DOCA maintains state, there is no need to include the OC-Applications AVP into the DOCA-Report-Request/Answer command main level after the initial message exchange. The agreed common set of application is expected to be known by both DOCA client and server throughout the session lifetime.

5.4. OC-Action AVP

The OC-Action (AVP Code TBD6) is of type OctetString and size of one octet. The octet has the following three possible values:

Start (1)

Signals the start of the overload condition. This implies the receiver is requested to act according to the information found in the OC-Information.

Stop (2)

Signals the end of the overload condition.

Interim (3)

Updates the overload information. The interim can be sent during the overload condition or during the normal condition. This is the default value.

Any other value is reserved.

5.5. OC-Algorithm AVP

The OC-Algorithm (AVP Code TBD7) is of type Unsigned32. The contains supported 'algorithms' to mitigate the overload condition. The OC-Algorithm AVP is formatted as a vector of algorithm flag bits. The following 'algorithms' are supported:

Drop (0x00000001)

Messages are plain dropped. It is RECOMMENDED to drop messages selectively based, for example, on application priorities. This is the default algorithm.

Throttle (0x00000002)

The message sending rate is according to the OC-Sending-Rate AVP.

Prioritize (0x00000004)

Apply priorities among applications and the other used means for holding traffic.

Any other value is reserved.

The 'algorithms' are only applied at a Diameter node when the

overload condition has been signaled.

During the initialization state a DOCA client includes those algorithms it supports and is interested in. A DOCA server then returns the algorithm that it has in common with the DOCA client (and intends to use). One or more common algorithms are then used during the established state.

If a DOCA server does not have a common algorithm with a DOCA client or the DOCA server cannot agree on one based on a local policy, then the DOCA server **MUST** send the DOCA-Report-Answer indicating an error and set the Result-Code to the DIAMETER_NO_COMMON_ALGORITHM value.

5.6. OC-Level AVP

The OC-Level (AVP Code TBD8) is of type OctetString and size of one octet. The octet has the following five possible values:

Normal (1)

Everything is in control. Meaningful only when the OC-Action is set to 'Interim' since when the overload condition level is considered normal, the overload condition **SHOULD** be stopped. This is the default value.

Raising (2)

There is a sign of increasing load.

Alarming (3)

The overload condition is reaching the level where quick measures **SHOULD** be done to mitigate the overload condition.

Panic (4)

The overload condition is severe. Apply any measure to mitigate the overload condition but still allowed to send messages.

Hold (5)

Do not send any messages, please. When this level is signaled, the OC-Best-Before time **SHOULD NOT** be respected but an explicit overload condition stop has to be received (with an exception the Diameter node realizes its other end has rebooted or otherwise lost its state).

Switch servers (6)

Do not talk to me again. When this level is signaled, the DOCA client MUST switch to an alternative server.

Any other value is reserved.

If the receiver cannot agree on or does not understand the OC-Level AVP value, then an error MUST be returned with the Result-Code AVP set to the value `DIAMETER_INVALID_AVP_VALUE` and the Failed-AVP AVP containing the OC-Level AVP.

5.7. OC-Utilization AVP

The OC-Utilization (AVP Code TBD9) is of type `Float32` and tells the overall utilization level percentage of the Diameter node. Values between 0.0 to 100.0 are valid.

5.8. OC-Tocl AVP

The OC-Tocl (AVP Code TBD10) is of type `Unsigned32` and tells the Tolc timer value in milliseconds. This timer defines the interval for sending periodic DOCA-Report-Request messages with the OC-Action AVP set to 'Interim'. The value of zero (0) means no periodic DOCA-Report-Request messages are sent or desired. The default value is 120000.

The OC-Tocl AVP can be considered as a hint for a desired sending rate of subsequent messages.

If a DOCA server finds the Tocl value proposed by a DOCA client either too small (i.e. too frequent periodic messages) or too big (i.e. too seldom periodic messages), then the DOCA server MUST send the DOCA-Report-Answer indicating an error and set the Result-Code either to the `DIAMETER_TOCL_TOO_SMALL` or `DIAMETER_TOCL_TOO_BIG` value.

5.9. OC-Sending-Rate AVP

The OC-Sending-Rate (AVP Code TBD11) is of type `Float32` and tells the the maximum Diameter message sending rate per second the sender of this information wishes to receive Diameter messages. Only positive values are valid. A value of zero (0.0) or the absence of this AVP means the information sender has no specific rate preference.

If a DOCA server finds the sending rate value proposed by a DOCA client too big (i.e., too frequent periodic messages), then the DOCA server MUST send the DOCA-Report-Answer indicating an error and set

the Result-Code to the DIAMETER_RATE_TOO_BIG value.

5.10. OC-Best-Before AVP

The OC-Best-Before (AVP Code TBD12) is of type Time and tells the expiration time/date for the information received in the OC-Information. For example, when the overload condition is on, the expiration of the 'best before' timer causes the same as receiving a DOCA-Report-Request/Answer with the OC-Action set to 'Stop'.

[Editor's note: to be decided whether a duration timer is a better measure. Using Time has the assumptions nodes have actually clocks that a running approximately same time.]

5.11. OC-Origin AVP

The OC-Origin (AVP Code TBD13) is of type DiameterIdentity and tells the identity of the Diameter node that originated included the overload control information. Both host and realm information MUST be included in the OC-Origin AVP. Note, if the OC-Scope AVP indicates only a realm wide scope for the overload information, then the realm part of the OC-Origin AVP is meaningful and the host information only serves as an additional information of the representative for the realm wide information.

5.12. OC-Priority AVP

The OC-Priority (AVP Code TBD14) is of type Unsigned32 and defines the priority level. The value of 0x00000000 is the highest priority and the value of 0xffffffff is the lowest priority. The absence of the OC-Priority AVP means there is not specific priority level defined and the priority SHOULD be considered as the lowest possible.

When used within the OC-Information grouped AVP, the OC-Priority AVP defines the priority for the listed applications within the OC-Applications AVP.

5.13. Attribute Value Pair flag rules

				+-----+ AVP flag rules +-----+	
Attribute Name	AVP Code	Section Defined	Value Type	MUST	NOT
OC-Information	TBD3	x.x	Grouped	M	V
OC-Scope	TBD4	x.x	Unsigned32	M	V
OC-Application	TBD5	x.x	Grouped	M	V
OC-Action	TBD6	x.x	OctetString	M	V
OC-Algorithm	TBD7	x.x	Unsigned32	M	V
OC-Level	TBD8	x.x	OctetString	M	V
OC-Utilization	TBD9	x.x	Float32	M	V
OC-Tocl	TBD10	x.x	Unsigned32	M	V
OC-Sending-Rate	TBD11	x.x	Float32	M	V
OC-Best-Before	TBD12	x.x	Time	M	V
OC-Origin	TBD13	x.x	DiameterIdentity	M	V
OC-Priority	TBD14	x.x	Unsigned32	M	V

6. Transport Considerations

In case of Stream Control Transmission Protocol (SCTP) transport, the DOCA application is RECOMMENDED to mark its Diameter packets using the DOCA defined SCTP Payload Protocol Identifier (PPID) TBD1. The PPID MAY be used by intermediating network nodes or agents to peek into SCTP message and find out that this is about overload control. Such information can be used for prioritizing SCTP packet handling as an example.

7. Examples

Consider the following simplified scenario shown in Figure 1 where two servers are connected to a proxy. All three nodes understand the DOCA application. These three nodes belong to the same administrative domain and the operator decided that he wants to hide the Diameter topology of his own network. Consequently, aggregate information is provided by the proxy for any Diameter overload message exchange. The Diameter client also supports the DOCA application. Between the client and the Diameter proxy we assume an arbitrary Diameter network that passes Diameter messages back and forth.

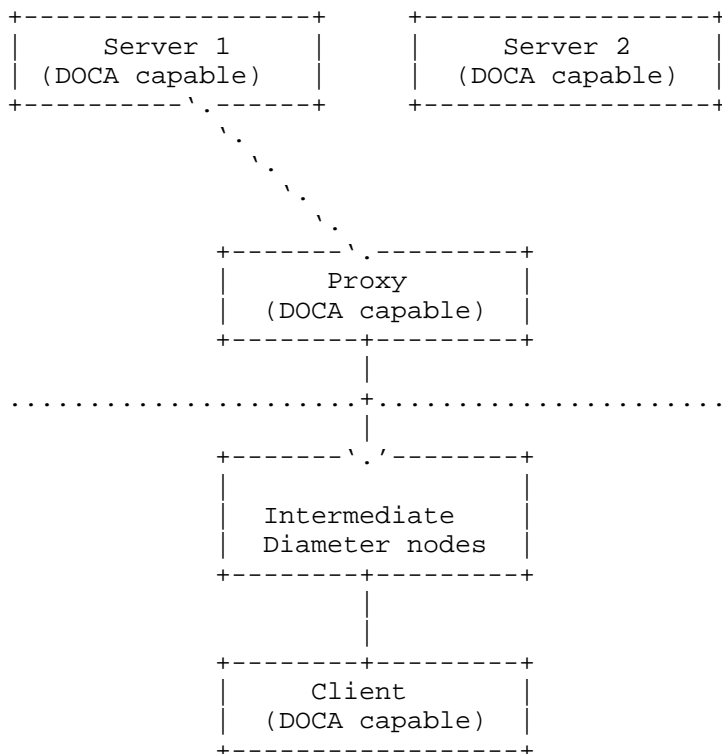


Figure 1

Let us assume that the DOCA exchange is initiated by server 1 who determines that the load situation increases. It sends a DOCA-Report-Request message (with piggybacked overload information) towards the client. The message also instructs the client to reduce

it's sending rate. The proxy, who receives the DOCA-Report-Request decides to change the included OC-Origin information and forwards the request to the client.

When the client receives the DOCA-Report-Request message is processes the content, evaluates the overload information content and reacts accordingly, and returns a DOCA-Report-Answer message back to acknowledge the receipt.

Alternatively, let us assume that the proxy does not forward the message but instead terminates the DOCA-Report-Request received from Server 1. It instead decides to route traffic to the backup server, Server 2. In this case the entire process was transparent for the client.

8. IANA Considerations

8.1. Application Identifiers

This specification reserves a new Diameter Application-ID TBD14 for the Diameter Overload Control Application (DOCA) from the 'Authentication, Authorization, and Accounting (AAA) Parameters' Application IDs registry.

8.2. SCTP Payload Protocol Identifier

Section 6 reserves a new SCTP Payload Protocol Identifier for the DOCA application usage. The value is reserved from the existing SCTP Payload Protocol Identifiers registry.

8.3. Command Codes

Two command codes are defined in Section 4. The DOCA-Report-Request Command Code is TBD and the DOCA-Report-Answer Command Code is TBD. Both are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' Command Codes registry.

8.4. AVP Codes

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

8.5. Result-Code Values

This specification adds several Diameter Overload Control Application specific Permanent Failure codes from the 'Authentication,

Authorization, and Accounting (AAA) Parameters' Result-Code AVP Values (code 268) - Permanent Failure registry:

AVP Values	Attribute Name	Reference
5xxx	DIAMETER_NO_COMMON_SCOPE	RFCxxxx
5xxx	DIAMETER_NO_COMMON_ALGORITHM	RFCxxxx
5xxx	DIAMETER_TOCL_TOO_SMALL	RFCxxxx
5xxx	DIAMETER_TOCL_TOO_BIG	RFCxxxx
5xxx	DIAMETER_RATE_TOO_BIG	RFCxxxx

8.6. New Registries

Four new registries are needed under the 'Authentication, Authorization, and Accounting (AAA) Parameters' registry:

- o OC-Scope AVP Values: the policy for this registry is Specification Required.
- o OC-Action AVP Values: the policy for this registry is Standards Action.
- o OC-Level AVP Values: the policy for this registry is Standards Action.
- o OC-Algorithm AVP Values: the policy for this registry is Specification Required.

9. Security Considerations

The security properties of the Diameter Overload Control Application (DOCA) follows the security model of Diameter [RFC6733]. This implies there is no proper means to verify the message and AVP content correctness if multiple intermediate Diameter agents are present on the path between the DOCA client and server. As a result a malicious intermediate could feed incorrect overload control information to DOCA clients and peers, and thus affect negatively to the overload condition recovery. A possible way to overcome the obvious security vulnerability is to mandate the use of end-to-end security at the Diameter AVP level.

As such, like any other Diameter application this document would benefit from a Diameter end-to-end security mechanism. While work is in progress it has not yet been finalized and therefore this specification does not rely on it.

10. Acknowledgements

The author thanks Annett Seefeldt for her constructive comments on

the technical aspects on this document.

11. References

11.1. Normative References

- [I-D.ietf-dime-overload-reqs]
McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", draft-ietf-dime-overload-reqs-04 (work in progress), February 2013.
- [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.

11.2. Informative References

- [I-D.campbell-dime-overload-data-analysis]
Campbell, B., Tschofenig, H., Korhonen, J., and A. Roach, "Diameter Overload Data Analysis", draft-campbell-dime-overload-data-analysis-00 (work in progress), February 2013.
- [RFC6408] Jones, M., Korhonen, J., and L. Morand, "Diameter Straightforward-Naming Authority Pointer (S-NAPTR) Usage", RFC 6408, November 2011.

Appendix A. Design Justification

Section 1 discussed the motivation and the background for the Diameter enhancements for an explicit Diameter overload control solution. This specification solves the overload control at the application level instead of extending the Diameter base protocol or piggybacking overload control information on top of existing applications. The reasoning is the following:

1. The support for Diameter overload control capability between Diameter peers is explicit (i.e., a new application-id is advertised) and thus not build on an exchange of optional Attribute Value Pairs (AVPs).
2. The support for Diameter overload control capability between Diameter client and server is explicit.

3. The peer selection follows the existing standards including DNS-based discovery [RFC6408] and does not assume additional peer selection criteria learnt from an exchange of optional AVPs.
4. The application based solution is able to traverse and also propagate overload control information through realms that deploy relay agents without Diameter overload control support.
5. The propagation does not depend on a modified behavior of already specified Diameter command codes.
6. Pretending not to establish a state when there actually is an overload capability and information state still maintained. The state might not be at the application level but is there.
7. Trying to avoid information flooding, especially across administrative domains.
8. The use of the application concept allows established mechanisms for filtering and Diameter traffic engineering, since it behaves like any other Diameter application.
9. The use of the dedicated application allows to isolate (even physically) the overload signaling into a dedicated transport that is not affected by other Diameter messages and network traffic.

Authors' Addresses

Jouni Korhonen
Renesas Mobile
Porkkalankatu 24
Helsinki 00180
Finland

Email: jouni.nospam@gmail.com

Hannes Tschofenig (editor)
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

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

DIME
Internet-Draft
Intended status: Standards Track
Expires: November 18, 2013

A. B. Roach
Mozilla
E. McMurry
Tekelec
May 17, 2013

A Mechanism for Diameter Overload Control
draft-roach-dime-overload-ctrl-03

Abstract

When a Diameter server or agent becomes overloaded, it needs to be able to gracefully reduce its load, typically by informing clients to reduce or stop sending traffic for some period of time. Otherwise, it must continue to expend resources parsing and responding to Diameter messages.

This document proposes a concrete, application-independent mechanism to address the challenge of communicating load and overload state among Diameter peers, and specifies an algorithm for load abatement to address such overload conditions as they occur. The load abatement algorithm is extensible, allowing for future documents to define additional load abatement approaches.

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 November 18, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Mechanism Properties	4
1.2. Overview of Operation	6
1.3. Documentation Conventions	6
2. Overload Scopes	6
2.1. Scope Descriptions	7
2.2. Combining Scopes	8
3. Diameter Node Behavior	9
3.1. Connection Establishment Procedures	9
3.2. Diameter Client and Diameter Server Behavior	11
3.2.1. Sending a Request to a Compliant Peer	12
3.2.2. Receiving a Request	13
3.2.3. Sending an Answer to a Compliant Peer	14
3.2.4. Receiving an Answer from a Compliant Peer	15
3.3. Diameter Agent Behavior	16
3.3.1. Proxying a Request	16
3.3.2. Proxying an Answer	16
3.4. Proactive Load and Overload Communication	17
3.5. Load Processing	17
3.5.1. Sending Load Information	17
3.5.2. Receiving Load Information	18
3.6. Session Establishment for Session Groups	20
3.6.1. Session Group Concepts	20
3.6.2. Session Group Procedures	22
4. Loss-Based Overload Control Algorithm	22
4.1. Overload-Metric values for the 'Loss' Algorithm	23
4.2. Example Implementation	24
5. Diameter AVPs for Overload	28
5.1. Load-Info AVP	28
5.2. Supported-Scopes AVP	28
5.3. Overload-Algorithm AVP	29
5.4. Overload-Info-Scope AVP	30
5.4.1. Realm Scope	31
5.4.2. Application-ID Scope	31
5.4.3. Host Scope	31
5.4.4. Session Scope	31

5.4.5.	Connection Scope	31
5.4.6.	Session Group Scope	32
5.5.	Overload-Metric AVP	32
5.6.	Period-Of-Validity AVP	32
5.7.	Session-Group AVP	32
5.8.	Load AVP	32
6.	Security Considerations	32
7.	IANA Considerations	33
7.1.	New Diameter AVPs	33
7.2.	New Diameter Disconnect-Cause	33
7.3.	New Diameter Response Code	34
7.4.	New Command Flag	34
7.5.	Overload Algorithm Registry	34
7.6.	Overload Scope Registry	34
8.	References	35
8.1.	Normative References	35
8.2.	Informative References	35
Appendix A.	Acknowledgements	35
Appendix B.	Requirements Analysis	36
Appendix C.	Extending the Overload Mechanism	45
C.1.	New Algorithms	45
C.2.	New Scopes	46
Appendix D.	Design Rationale	46
D.1.	Piggybacking	47
D.2.	Load AVP in All Packets	48
D.3.	Graceful Failure	48
Authors' Addresses	49

1. Introduction

When a Diameter [RFC6733] server or agent becomes overloaded, it needs to be able to gracefully reduce its load, typically by informing clients to reduce or stop sending traffic for some period of time. Otherwise, it must continue to expend resources parsing and responding to Diameter messages.

This document defines a mechanism for communicating the load and overload information among Diameter nodes. It also defines a base algorithm for shedding traffic under overload circumstances. The design of the mechanism described in this document allows for the definition of alternate load abatement algorithms as well.

The mechanism proposed in this document is heavily influenced by the work performed in the IETF Session Initiation Protocol (SIP) Overload Control Working Group, and draws on the conclusions reached by that working group after extensive network modeling.

The solution described in this document is intended to satisfy the requirements described in [I-D.ietf-dime-overload-reqs], with the exception of REQ 34. As discussed in that document, the intention of a Diameter overload mechanism is to handle overload of the actual message processing portions of Diameter servers. This is in contrast to congestion, which is the inability of the underlying switching and routing fabric of the network to carry the volume of traffic at the volume that IP hosts wish to send it. Handling of congestion is relegated to the underlying transport protocol (TCP or SCTP), and will not be discussed.

Philosophically, the approach in designing this mechanism is based on the prospect that building a base-level, fully compliant implementation should be a very simple and straightforward exercise. However, the protocol includes many additional features that may be implemented to allow Diameter nodes to apply increasingly sophisticated behaviors. This approach gives implementors the freedom to implement as sophisticated a scheme as they desire, while freeing them from the burden of unnecessary complexity. By doing so, the mechanism allows for the rapid development and deployment of the mechanism followed by a period of steady and gradual improvements as implementations become more capable.

1.1. Mechanism Properties

The core Diameter overload mechanism described in this document is fundamentally hop-by-hop. The rationale for using a hop-by-hop approach is the same as is described in section 5.1 of [RFC6357]. However, due to the fact that Diameter networks frequently have

traffic that is easily grouped into a few well-defined categories, we have added some concepts that allow Diameter agents to push back on subsets of traffic that correspond to certain well-defined and client-visible constructs (such as Destination-Host, Destination-Realm, and Application-ID). These constructs are termed "Scopes" in this document. A more complete discussion of Scopes is found in Section 2.

The key information transmitted between Diameter peers is the current server load (to allow for better balancing of traffic, so as to preempt overload in the first place) as well as an indication of overload state and severity (overload information). The actual load and overload information is conveyed as a new compound AVP, added to any Diameter messages that allow for extensibility. As discussed in section 3.2 of [RFC6733], all CCFs are encouraged to include AVP-level extensibility by inclusion of a "* [AVP]" construct in their syntax definition. The document author has conducted an extensive (although admittedly not exhaustive) audit of existing applications, and found none lacking this property. The inclusion of load and overload information in existing messages has the property that the frequency with which information can be exchanged increases as load on the system goes up.

For the purpose of grouping the several different parts of load information together, this mechanism makes use of a Grouped AVP, called "Load-Info". The Load-Info AVP may appear one or more times in any extensible command, with the restriction that each instance of the Load-Info AVP must contain different Scopes.

Load and overload information can be conveyed during times of inter-node quiescence through the use of DWR/DWA exchanges. These exchanges can also be used to proactively change the overload or load level of a server when no other transaction is ready to be sent. Finally, in the unlikely event that an application is defined that precludes the inclusion of new AVPs in its commands, DWR/DWA exchanges can be sent at any rate acceptable to the server in order to convey load and overload information.

In [RFC3588], the DWR and DWA message syntax did not allow for the addition of new AVPs in the DWR and DWA messages. This oversight was fixed in [RFC6733]. To allow for transmission of load information on quiescent links, implementations of the mechanism described in this document are expected to correctly handle extension AVPs in DWR and DWA messages, even if such implementations have not otherwise been upgraded to support [RFC6733].

1.2. Overview of Operation

During the capabilities exchange phase of connection establishment, peers determine whether the connection will make use of the overload control mechanism; and, if so, which optional behaviors are to be employed.

The information sent between adjacent nodes includes two key metrics: Load (which, roughly speaking, provides a linear metric of how busy the node is), and Overload-Metric (which is input to the negotiated load abatement algorithm).

Message originators (whether originating a request or an answer) include one or more Load-Info AVPs in messages when they form them. These Load-Info AVPs reflect the originators' own load and overload state.

Because information is being used on a hop-by-hop basis, it is exchanged only between adjacent nodes. This means that any Diameter agent that forwards a message (request or answer) is required to remove any information received from the previous hop, and act upon it as necessary. Agents also add their own load and overload information (which may, at implementors' preference, take previous-hop information into account) into a new Load-Info AVP before sending the request or answer along.

Because the mechanism requires affirmative indication of support in the capabilities exchange phase of connection establishment, load and overload information will never be sent to intermediaries that do not support the overload mechanism. Therefore, no special provisions need to be made for removal of information at such intermediaries -- it will simply not be sent to them.

Message recipients are responsible for reading and acting upon load and overload information that they receive in such messages.

1.3. Documentation Conventions

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].

2. Overload Scopes

In normal operation, a Diameter node may be overloaded for some but not all possible requests. For example, an agent that supports two realms (realm A and realm B in this example) may route traffic to one

set of servers for realm A, and another set of servers for realm B. If the realm A servers are overloaded but realm B servers are not, then the agent is effectively overloaded for realm A but not for realm B.

Despite the fact that Diameter agents can report on scopes that semantically map to constructs elsewhere in the network, it is important to keep in mind that overload state is still reported on a hop-by-hop basis. In other words, the overload state reported for realm A in the example above represents the aggregate of the agent's overload state along with the overload state being reported by applicable upstream servers (those serving realm A).

Even without the use of Diameter agents, similar situations may arise in servers that need to make use of external resources for certain applications but not for others. For example, if a single server is handling two applications, one of which uses an external database while the other does not, it may become overloaded for the application that uses the external database when the database response latency increases.

The indication of scopes for overload information (using the Overload-Info-Scope AVP; see Section 5.4) allows a node to indicate a subset of requests to which overload information is to be applied. This document defines seven scopes; only "Connection" scope is mandatory to implement. The use of the optional scopes, along with the use of any additional scopes defined in other documents, is negotiated at connection establishment time; see Section 3.1.

2.1. Scope Descriptions

Destination-Realm: This scope, which nodes **MUST** implement, pertains to all transactions that have a Destination-Realm AVP matching the indicated value.

Application-ID: This scope, which nodes **MUST** implement, pertains to all transactions that contain an Application-ID field matching the indicated value.

Destination-Host: This scope, which nodes **SHOULD** implement, pertains to all transactions that have a Destination-Host AVP matching the indicated value.

Host: This scope, which nodes **SHOULD** implement, pertains to all transactions sent directly to the host matching the indicated value.

Connection: This scope, which nodes MUST implement, pertains to all transactions sent on the same TCP connection or SCTP association. This scope has no details indicating which connection or association it applies to; instead, the recipient of an indication of "Connection" scope is to use the connection or association on which the message was received as the indicated connection or association. In other words, any use of Connection scope applies to "this connection."

Session-Group: This scope, which nodes MAY implement, pertains to all transactions in a session that has been assigned to the indicated group. For more information on assigning sessions to groups, see Section 3.6.

Session: This scope, which nodes MAY implement, pertains to all transactions in the indicated session.

Some applications do not have long-running sessions containing multiple transactions. For such applications, the use of "Session-Group" and "Session" scopes do not make sense. Such applications will instead make use of the most applicable of the remaining scopes (plus any negotiated extension scopes) to achieve overload control.

OPEN ISSUE: Is there value to including a stream-level scope for SCTP? We haven't been able to come up with a use case for doing so yet, but it wouldn't necessarily be unreasonable.

2.2. Combining Scopes

To allow for the expression of more complicated scopes than the primitives defined above, multiple Overload-Info-Scope AVPs may be included in a single Load-Info AVP. Semantically, these scopes are included in the following way:

- o Attributes of the different kinds are logically and-ed together (e.g., if both "Destination-Realm" and "Application-ID" are present, the information applies to requests sent that match both the realm and the application).
- o Attributes of the same kind are logically or-ed together (e.g., if two "Destination-Realm"s are present, the information applies to requests sent to either realm).
- o If a transaction falls within more than one scope, the "most overloaded" scope is used for traffic shaping.

To prevent the complexity of implementing arbitrary scope combination rules, only the following combinations of scopes are allowed (OPEN ISSUE -- we need to figure out what makes most sense for expressing these combinations. Formal grammar? Prose? A table of some kind? For now, they're expressed as a pseudo-ABNF):

- o 1*(Destination-Realm) 0*1(Application-ID)
- o 1*(Application-ID) 0*1(Destination-Realm)
- o 1*(Application-ID) 0*1(Destination-Host)
- o 1*(Application-ID) 0*1(Host)
- o 1*(Application-ID) 0*1(Connection)
- o 1*(Destination-Host)
- o 1*(Host)
- o 1*(Connection)
- o 1*(Session-Group) 0*1(Host | Connection)
- o 1*(Session) 0*1(Host | Connection)

OPEN ISSUE: Is this the right set of scope combinations? Is there a need for more? Are any of these unnecessary? Ideally, this should be the smallest set of combinations that lets nodes report what they realistically need to report.

Any document that creates additional scopes MUST define how they may be combined with all scopes registered with IANA at the time of their publication.

3. Diameter Node Behavior

The following sections outline the behavior expected of Diameter clients, servers, and agents that implement the overload control mechanism.

OPEN ISSUE: SIP Overload Control includes a sequence parameter to ensure that out-of-order messages do not cause the receiver to act on state that is no longer accurate. Is message reordering a concern in Diameter? That is, do we need to include sequence numbers in the messages to ensure that the receiver does not act on stale state information? Because Diameter uses only reliable, in-order transports, it seems that this isn't likely to be an issue. Is there room for a race when multiple connections are in use?

3.1. Connection Establishment Procedures

Negotiation for support of this mechanism is performed during Diameter capabilities exchange. Optional protocol features and extensions to this mechanism are also negotiated at this time. No

provision is provided for renegotiation of mechanism use or extensions during the course of a connection. If peers wish to make changes to the mechanism, they must create a new connection to do so.

The connection initiator includes a Load-Info AVP in the CER (Capabilities-Exchange-Request) message that it sends after establishing the connection. This Load-Info AVP MUST contain a Supported-Scopes AVP and an Overload-Algorithm AVP. The Supported-Scopes AVP includes a comprehensive list of scopes supported that the connection initiator can receive and understand. See Section 5.2 for information on the format of the Supported-Scopes AVP.

The Load-Info AVP in a CER message also MAY contain one or more Overload-Algorithm AVPs. If present, these AVPs indicate every Overload-Algorithm the connection initiator is willing to support for the connection that is being established. If the connection initiator supports only the "Loss" algorithm, it MAY indicate this fact by omitting the Overload-Algorithm altogether.

The Load-Info AVP in a CER message MAY also contain additional AVPs, as defined in other documents, for the purpose of negotiation extensions to the Overload mechanism.

The Diameter node that receives a CER message first examines it for the presence of a Load-Info AVP. If no such AVP is present, the node concludes that the overload control mechanism is not supported for this connection, and no further overload-related negotiation is performed. If the received CER contains a Load-Info AVP, the recipient of that message stores that information locally in the context of the connection being established. It then examines the Overload-Algorithm AVPs, if present, and selects a single algorithm from that list. If no Overload-Algorithm is indicated, then the base "Loss" algorithm is used for the connection. In either case, the recipient of the CER stores this algorithm in the context of the connection.

When a node conformant to this specification sends a Capabilities-Exchange-Answer (CEA) message in answer to a CER that contained a Load-Info AVP, the CEA MUST contain a Load-Info AVP. This Load-Info AVP MUST contain a Supported-Scopes AVP that includes a comprehensive list of scopes supported that the connection initiator can receive and understand. The CEA also contains zero or one Overload-Algorithm AVPs. If present, this Overload-Algorithm MUST match one of the Overload-Algorithm AVPs sent in the CER, and it indicates the overload control algorithm that will be used for the connection. If the CEA contains no Overload-Algorithm, the connection will use the "Loss" algorithm.

When a node receives a CEA message, it examines it for the presence of a Load-Info AVP. If no such AVP is present, the node concludes that the overload mechanism is not supported for this connection. If the received CEA contains a Load-Info AVP, then the recipient extracts the Supported-Scopes information, and stores them locally in the context of the connection being established. It then checks for the presence of an Overload-Algorithm AVP. If present, this AVP indicates the overload control algorithm that will be used for the connection. If absent, then the connection will use the "Loss" algorithm.

If a node receives a CEA message that indicates support for a scope that it did not indicate in its CER or which selects an overload control algorithm that it did not advertise in its CER, then it MUST terminate the connection by sending a DPR with a Disconnect-Cause of `NEGOTIATION_FAILURE`, (128 [actual value TBD]) indicating that the CEA sender has failed to properly follow the negotiation process described above.

Note that the Supported-Scopes announcement during capabilities exchange is a set of mutual advertisements of which scopes the two nodes are willing to receive information about. It is not a negotiation. It is perfectly acceptable for a node to send information for scopes it did not include in the Supported-Scopes AVP it sent, as long as the recipient indicated support for receiving such a scope. For example, a Diameter agent, during connection establishment with a client, may indicate support for receiving only "Connection" and "Host" scope; however, if the client indicated support for "Application" scope, then the agent is free to send Load-Info AVPs that make use of "Application" scope to the client.

3.2. Diameter Client and Diameter Server Behavior

The following sections describe the behavior that Diameter clients and Diameter servers implement for the overload control mechanism. Behavior at Diameter Agents is described in Section 3.3.

To implement overload control, Diameter nodes need to keep track of three important metrics for each of the scopes for which information has been received: the overload metric for the scope, the period of validity for that overload metric, and the load within that scope. Conceptually, these are data records indexed by the scope to which they apply. In the following sections, we refer to these data records with the term "scope entry." Further, when it is necessary to distinguish between those scope entries referring to the load information received from other nodes and those referring to the load information sent to other nodes, we use the term "remote scope entry" to refer to the information received from other nodes, and "local

scope entry" to refer to that information that is being maintained to send to other nodes.

In order to allow recipients of overload information to perform certain performance optimizations, we also define a new command flag, called 'O'verload. This bit, when set, indicates that the message contains at least one Load-Info AVP with a non-zero Overload-Metric -- in other words, the sending node is overloaded for at least one context. See Section 7.4 for the definition of the 'O'verload bit.

OPEN ISSUE: Is there anything we can do to make this 'O'verload bit even more useful? Perhaps setting it only when the overload value has changed, or changed by a certain amount?

3.2.1. Sending a Request to a Compliant Peer

This section applies only to those requests sent to peers who negotiated use of the overload control mechanism during capabilities exchange. Requests sent over other connections are handled the same as they would in the absence of the overload control mechanism.

Before sending a request, a Diameter node must first determine which scope applies. It does this as follows: first, a next hop host and connection are determined, according to normal Diameter procedures (potentially modified as described in Section 3.5.2). The sending node then searches through its list of remote scope entries (ignoring any whose Period-of-Validity has expired) to determine which ones match the combination of the fields in the current request, the next-hop host, and the selected connection. If none of the matching scope entries are in overload, then the message is handled normally, and no additional processing is required.

As an optimization, a sending node MAY choose to track whether any of its peers are in overload, and to skip the preceding step if it knows that no scopes are in overload.

If one or more matching scope entries are in overload, then the sending node determines which scope is most overloaded. The sending node then sends, drops, or otherwise modifies handling of the request according to the negotiated overload control algorithm, using the Overload-Metric from the selected scope entry as input to the algorithm.

When determining which requests are impacted by the overload control algorithm, request senders MAY take into account the type of message being sent and its contents. For example, messages within an existing session may be prioritized over those that create a new session. The exact rules for such prioritization will likely vary

from application to application. The authors expect that specifications that define or specify the use of specific Diameter Applications may choose to formally define a set of rules for such prioritization on a per-Application basis.

The foregoing notwithstanding, senders **MUST NOT** use the content or type of request to exempt that request from overload handling. For example, if a peer requests a 50% decrease in sent traffic using the "Loss" algorithm (see Section 4), but the traffic that the sending node wishes to send consists 65% of traffic that the sender considers critical, then the sender is nonetheless obliged to drop some portion of that critical traffic (e.g., it may elect to drop all non-critical traffic and 23% of the critical traffic, resulting in an overall 50% reduction).

The sending node then inserts one or more Load-Info AVPs (see Section 5.1) into the request. If the sender inserts more than one Load-Info AVP, then each Load-Info AVP **MUST** contain a unique scope, as specified by the Overload-Scope AVP(s) inside the Load-Info AVP.

Each Load-Info AVP in the request **MUST** contain an Overload-Metric (see Section 5.5), indicating whether (and to what degree) the sender is overloaded for the indicated scope. If this metric is not zero, then the Load-Info AVP **MUST** also contain a Period-Of-Validity AVP (see Section 5.6), indicating the maximum period the recipient should consider the Overload-Metric to be valid. Any message containing a non-zero Overload-Metric also **MUST** set the 'O'verload bit in the Command Flags field to indicate to the recipient that the message contains an overload indication. See Section 7.4 for the definition of the 'O'verload bit.

Each Load-Info AVP **MUST** also contain a Load AVP, indicating the server's load level within the context of the indicated scope. See Section 3.5.1 for details on generating this load metric. Note that a server's load may frequently be identical for all the scopes for which it sends information.

3.2.2. Receiving a Request

3.2.2.1. Receiving a Request from a Compliant Peer

A node that receives a request from a peer that has negotiated support for the overload control mechanism will extract the Load-Info AVPs from the request and use each of them to update its remote scope entries. First, the node attempts to locate an existing scope entry that corresponds to the Overload-Scope indicated in the Load-Info AVP. If one does not exist, it is created. The scope entry is then populated with the overload metric, period of validity, and load

information. The message is then processed as normal.

In some circumstances, request recipients can become sufficiently overloaded that even those messages received from complaint clients can overwhelm its processing capabilities. Under such circumstances, nodes MAY begin treating a subset of such requests as if they were received from noncompliant peers (as explained in the following section).

3.2.2.2. Receiving a Request from a Noncompliant Peer

An important aspect of the overload control mechanism is that Diameter nodes that do not implement the mechanism cannot have an advantage over those that do. In other words, it is necessary to prevent the situation that a network in overload will cease servicing those transactions from overload-compliant nodes in favor of those sent by those nodes that do not implement the overload control mechanism. To achieve this goal, message recipients need to track the overload control metric on behalf of those sending nodes that do not implement overload, and to reject messages from those nodes that would have been dropped if the sender had implemented the overload mechanism.

A node that receives a request from a peer that has not negotiated support for the overload control mechanism searches through its list of local scope entries to determine which ones match the combination of the fields in the received request. (These are the entries that indicate the Overload-Metric that the node would have sent to the peer if the peer had supported the overload mechanism). If none of the matching scope entries are in overload, then the message is sent normally, and no additional processing is required.

If one or more matching local scope entries are in overload, then the node determines which scope is most overloaded. The node then executes the "Loss" overload control algorithm (see Section 4) using the overload metric in that most overloaded scope. If the result of running that algorithm determines that a sender who had implemented the overload control mechanism would have dropped the message, then the recipient MUST reply to the request with a `DIAMETER_PEER_IN_OVERLOAD` response (see Section 7.3).

3.2.3. Sending an Answer to a Compliant Peer

This section applies only to those answers sent to peers who negotiated use of the overload control mechanism during capabilities exchange.

When sending an answer, a Diameter node inserts one or more Load-Info

AVPs (see Section 5.1) into the answer. If the sender inserts more than one Load-Info AVP, then each Load-Info AVP MUST contain a unique scope, as specified by the Overload-Scope AVP(s) inside the Load-Info AVP.

Each Load-Info AVP in the answer MUST contain an Overload-Metric (see Section 5.5), indicating whether (and to what degree) the server is overloaded for the indicated scope. If this metric is not zero, then the Load-Info AVP MUST also contain a Period-Of-Validity AVP (see Section 5.6), indicating the maximum period the recipient should consider the Overload-Metric to be valid. Any message containing a non-zero Overload-Metric also MUST set the 'O'verload bit in the Command Flags field to indicate to the recipient that the message contains an overload indication. See Section 7.4 for the definition of the 'O'verload bit.

It is important to note that using this mechanism creates a closed feedback loop, with some amount of lag introduced by overload processing and the network. As such, implementers must be aware of the potential for such a system to produce oscillations in the overload level. Without proper control, it is also possible for these oscillations to diverge, resulting in undesirable behavior. There are several ways to address this issue, and it is left to implementors to determine the best way for their particular situation. However, at a minimum senders of overload control information SHOULD apply hysteresis to the Overload-Metric, and signal easing of overload more slowly than signaling increases.

Each Load-Info AVP MUST also contain a Load AVP, indicating the server's load level within the context of the indicated scope. See Section 3.5.1 for details on generating this load metric. Note that a server's load may frequently be identical for all the scopes for which it sends information.

3.2.4. Receiving an Answer from a Compliant Peer

A node that receives an answer from a peer that has negotiated support for the overload control mechanism will extract the Load-Info AVPs from the answer and use each of them to update its remote scope entries. First, the node attempts to locate an existing scope entry that corresponds to the Overload-Scope indicated in the Load-Info AVP. If one does not exist, it is created. The scope entry is then populated with the overload metric, period of validity, and load information. The message is then processed as normal.

3.3. Diameter Agent Behavior

This section discusses the behavior of a Diameter Agent acting as a Proxy or Relay. Diameter Agents that provide redirect or translation services behave the same as Diameter Servers for the purpose of overload control, and follow the procedures defined in Section 3.2.

Whenever sending a request or an answer, Agents MUST include a Load-Info AVP reflecting the Agent's overload and load information. In formulating this information, the Agent may choose to use only that information relating to its own local resources. However, better network behavior can be achieved if agents incorporate information received from their peers when generating overload information. The exact means for incorporating such information is left to local policy at the agent.

For example: consider an agent that distributes sessions and transactions among three Diameter servers, each hosting a different Diameter application. While it would be compliant for the Agent to only report its own overload state (i.e., at "Host" scope), overall network behavior would be improved if it chose to also report overload state for up to three additional scopes (i.e. at "Application-ID" scope), incorporating the Overload information received from each server in these scopes.

3.3.1. Proxying a Request

Upon receiving a request, a Diameter Proxy or Relay performs the steps detailed in Section 3.2.2.

The agent then MUST remove all Load-Info AVPs from the request: Load-Info is never passed through a Proxy or Relay transparently.

When the Diameter Agent proxies or relays a request, it follows the process outlined in Section 3.2.1.

3.3.2. Proxying an Answer

Upon receiving an answer, a Diameter Agent follows the process described in Section 3.2.4 to update its remote scope entries.

The Agent then MUST remove all Load-Info AVPs from the answer: Load-Info is never passed through a Proxy or Relay transparently.

When the Diameter Agent proxies or relays a response, it follows the process outlined in Section 3.2.3.

3.4. Proactive Load and Overload Communication

Because not all Diameter links will have constant traffic, it may be occasionally necessary to send overload and/or load information over links that would otherwise be quiescent. To proactively send such information to peers, the Diameter node with information to convey may choose to send a Diameter Watchdog Request (DWR) message to its peers. The procedure described in Section 3.2.1 applies to these requests, which provides the means to send load and overload information.

In order to prevent unnecessarily diminished throughput between peers, a Diameter node SHOULD proactively send a DWR to all its peers whenever it leaves an overload state. Similarly, in order to provide peers the proper data for load distribution, nodes SHOULD send DWR messages to a peer if the load information most recently sent to that peer has changed by more than 20% and is more than 5 seconds old.

3.5. Load Processing

While the remainder of the mechanism described in this document is aimed at handling overload situations once they occur, it is far better for a system if overload can be avoided altogether. In order to facilitate overload avoidance, the overload mechanism includes the ability to convey node load information.

Semantically, the Load information sent by a Diameter node indicates the current utilization of its most constrained resource. It is a linear scale from 0 (least loaded) to 65535 (most loaded).

It is critical to distinguish between the value conveyed in the Load AVP and the value conveyed in the Overload-Metric AVP. The Load AVP is computed and used independent of the Overload-Algorithm selected for a connection, while the Overload-Metric is meaningful only in the context of the selected algorithm. Most importantly, the Load information never has any impact on the behavior specified in the overload algorithm. If a node reports a Load of 65535, but the Overload-Metric does not indicate any need to apply the selected overload control algorithm, then the sender MUST NOT apply the selected overload control algorithm. Conversely, if a node is reporting an Overload-Metric that requires the recipient to take action to reduce traffic, those actions MUST be taken, even if the node is simultaneously reporting a Load value of 0.

3.5.1. Sending Load Information

Diameter nodes implementing the overload mechanism described in this document MUST include a Load AVP (inside a Load-Info AVP) in every

Diameter message (request and answer) they send over a connection that has been negotiated to use the overload control mechanism. Note that this requirement does not necessitate calculation of the Load metric each time a message is sent; the Load value may be calculated periodically (e.g., every 100 ms), and used for every message sent until it is recalculated.

The algorithm for generation of the load metric is a matter of local policy at the Diameter node, and may vary widely based on the internal software architecture of that node.

For advanced calculations of Load, anticipated inputs to the computation include CPU utilization, network utilization, processor interrupts, I/O throughput, and internal message queue depths.

To free implementors from the potential complexity of determining an optimal calculation for load, we define a very simple, baseline load calculation that MAY be used for the purpose of populating the Load AVP. Implementations using this simplified calculation will use a configured, hard-coded, or Service Level Agreement (SLA)-defined maximum number of transactions per second (TPS) which a node is known to be able to support without issue. These implementations simply report their load as a linear representation of how much of this known capacity is currently in use:

$$\text{Load} = \text{MIN}(\text{Current_TPS} * 65535 / \text{Maximum_TPS}, 65535)$$

To prevent rapid fluctuations in the load metric, nodes SHOULD report a rolling average of the calculated load rather than the actual instantaneous load at any given moment.

Load information is scoped to the level indicated by the Overload-Info-Scope AVP present in the Load-Info AVP in which the Load AVP appears.

3.5.2. Receiving Load Information

While sending load information is mandatory, the actual processing of load information at a recipient is completely optional. Ideally, recipients will use the load information as input to a decision regarding which of multiple equivalent servers to use when initiating a new connection. Recipients may choose to update load information on receipt of every message; alternately, they may periodically "sample" messages from a host to determine the load it is currently reporting.

3.5.2.1. Example Load Handling

This section describes a non-normative example of how recipients can use Load information received from other Diameter nodes. At a high level, the concept is that received load metrics are used to scale the distribution algorithm that the node uses for selection of a server from a group of equivalent servers.

Consider a client that uses DNS to resolve a host name into IP addresses. In this example, the client is attempting to reach the server for the realm example.com. It performs a NAPTR query for the "AAA+D2T" record for that domain, and receives a result pointing to the SRV record "_diameter._tcp.example.com". Querying for this SRV record, in turn, results in three entries, with the same priorities:

SRV Weight	Server Name
20	server-a.example.com
20	server-b.example.com
60	server-c.example.com

The client then examines the currently reported loads for each of the three servers. In this example, we are asserting that the reported load metrics are as follows:

Load	Server Name
13107 (20%)	server-a.example.com
26214 (60%)	server-b.example.com
52428 (80%)	server-c.example.com

Based on this load information, the client scales the SRV weights proportional to each server's reported load; the general formula is:

$$\text{new_weight} = \text{original_weight} * (65535 - \text{load}) / 65535$$

The node then calculates a new set of weights for the destination hosts:

- o server-a: new_weight = 20 * (65535 - 13107) / 65535 = 16
- o server-b: new_weight = 20 * (65535 - 26214) / 65535 = 12
- o server-c: new_weight = 60 * (65535 - 52428) / 65535 = 12

These three new weights (16, 12, and 12) are then used as input to

the random selection process traditionally used when selecting among several SRV records.

Note that this example is provided in the context of DNS SRV processing; however, it works equally well in the case that server processing weights are provisioned or made available through an alternate resolution process.

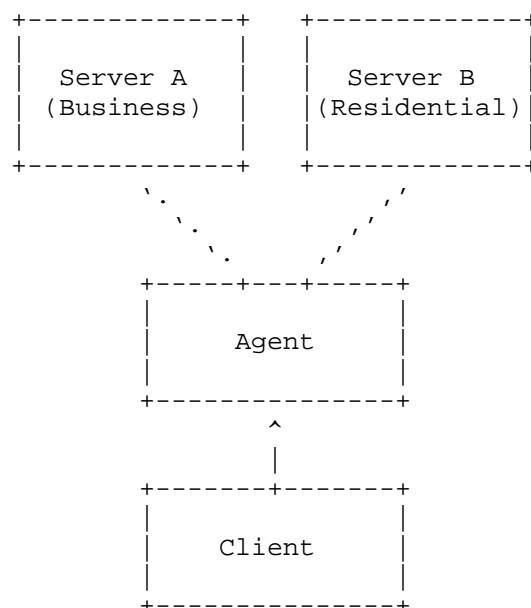
3.6. Session Establishment for Session Groups

The procedure in this section applies to any Diameter operation that may result in the creation of a new Diameter session. Note that these operations are performed in addition to any normal message processing, and in addition to the operations described in the following sections.

3.6.1. Session Group Concepts

At the time a session is established, the server and/or the client may choose to assign the newly created session to a Session Group that they can use to refer to the session (and other sessions in the same group) in later overload-related messages. This grouping is intended to be used by servers that have visibility into resources that may be independently overloaded, but which do not correspond to an existing Diameter construct (such as Application, Realm, or Destination Server).

One example of a server having visibility into resources that don't have a corresponding Diameter construct is a Diameter Agent servicing a mixed community of users -- say, one authenticated by a "Business" server, and another authenticated by a "Residential" server. The client in this network does not know which group any given session belongs in; the routing of sessions is based on information available only to the agent.



In this case, the Agent may wish to assign sessions to two client-visible Session Groups when the session is established. By doing so, the Agent gains the ability to report Load and Overload metrics to the Client independently for the two classes of users. This can be extremely helpful, for example, in allowing the Agent to ask the Client to throttle traffic for the Residential server when it becomes overloaded, without impacting sessions pertaining to the Business server.

Similar situations can arise even without the presence of Diameter Agents in the network: a server may have a class of sessions that require access to an off-board database (which can, itself, become overloaded), while also servicing a class of sessions that is handled entirely by a local authentication table. The server can use Session Groups to assign these two classes of sessions to different groups, and report overload on the class using the (overloaded) off-board database without impacting the other sessions.

In some applications, it is possible to have the session established by one peer (e.g., in the upstream direction), while some subsequent in-session transactions are initiated by the other peer (e.g., in the downstream direction). Because of this possibility, the overload mechanism allows both peers to establish a Session Group at the time the session is set up. The session identifiers are scoped to the node that sends them. In other words, if a server assigns a session to a group called "Residential", this group is not related to a

client group (if any) by the same name. For clarity, this document will refer to the session group assigned by the server performing the processing as a "local session group," and the session group assigned by the remote node as a "remote session group."

Nodes that send a session-creating request follow normal Diameter procedures, along with the additional behavior described in Section 3.2.1 and Section 3.3.1, as appropriate. Such nodes may also assign the session to a Session Group, as long as the peer to which they are communicating indicated support for the "Session-Group" scope during capabilities exchange. Whether to do so and what group to assign a session to is done according to local policy. To perform such assignment, the node will include a Session-Group AVP (see Section 5.7 in the Load-Info AVP for the session creating request. These nodes also store the assigned name as the session's local session group.

3.6.2. Session Group Procedures

The procedures in this section only apply on connections for which support for the "Session-Group" scope has been negotiated during capabilities exchange. See Section 3.1.

When a node receives a session creating request, it MUST check that request for the presence for a Session-Group AVP in its Load-Info AVP. If one is present, it stores that session group name as the remote session group name for that server. This allows clients to assign the session to a group, allowing it to indicate overload for server-initiated transactions in the resulting session.

When a node replies to a session creating request, it can choose to assign the newly-established session to a session group. Whether it chooses to do so is independent of whether the remote node assigned the session to a session group. To perform such an assignment, the node includes a Session-Group AVP in the Load-Info AVP sent in answer to the session-creating request. These nodes also store the assigned name as the session's local session group.

Finally, when a node that has sent a session-creating request receives a corresponding answer message, it MUST check that answer for the presence of a Session-Group AVP in its Load-Info AVP. If one is present, it stores that session group name as the remote session group name for that server.

4. Loss-Based Overload Control Algorithm

This section describes a baseline, mandatory-to-implement overload

control algorithm, identified by the indicator "Loss". This algorithm allows a Diameter peer to ask its peers to reduce the number of requests they would ordinarily send by a specified percentage. For example, if a peer requests of another peer that it reduce the traffic it is sending by 10%, then that peer will redirect, reject, or treat as failed, 10% of the traffic that would have otherwise been sent to this Diameter node.

4.1. Overload-Metric values for the 'Loss' Algorithm

A Diameter node entering the overload state for any of the scopes that it uses with its peers will calculate a value for its Overload Metric, in the range of 0 to 100 (inclusive). This value indicates the percentage traffic reduction the Diameter node wishes its peers to implement. The computation of the exact value for this parameter is left as an implementation choice at the sending node. It is acceptable for implementations to request different levels of traffic reduction to different peers according to local policy at the Diameter node. These Overload Metrics are then communicated to peers using the Overload-Metric AVP in requests and answers sent by this node.

Recipients of Overload-Metric AVPs on connections for which the "Loss" algorithm has been specified MUST reduce the number of requests sent in the corresponding scope by that percentage, either by redirecting them to an alternate destination, or by failing the request. For a Diameter Agent, these failures are indicated to the peer who originated the request by sending a `DIAMETER_PEER_IN_OVERLOAD` response (see Section 7.3). For diameter clients, these failures cause the client to behave as if they received a transient error in response to the request.

It is acceptable, when implementing the "Loss" algorithm, for the reduction in transactions to make use of a statistical loss function (e.g., random assignment of transactions into "success" and "failure" categories based on the indicated percentage). In such a case, the actual traffic reduction might vary slightly from the percentage indicated, albeit in an insignificant amount.

The selection of which messages to withhold from sending does not need to be arbitrary. For example, implementations are allowed to distinguish between higher-priority and lower-priority messages, and drop the lower-priority messages in favor of dropping the higher priority messages, as long as the total reduction in traffic conforms to the Overload-Metric in effect at the time. The selection of which messages to prioritize over others will likely vary from application to application (and may even be subject to standardization as part of the application definition). One example of such a prioritization

scheme would be to treat those messages that result in the creation of a new session as lower priority than those messages sent in the context of an established session.

4.2. Example Implementation

The exact means a client uses to implement the requirement that it reduce traffic by a requested percentage is left to the discretion of the implementor. However, to aid in understanding the nature of such an implementation, we present an example of a valid implementation in pseudo-code.

In this example, we consider that the sending node maintains two classes of request. The first category are considered of lower priority than the second category. If a reduction in traffic is required, then these lower priority requests will be dropped before any of the higher priority requests are dropped.

The sending Diameter node determines the mix of requests falling into the first category, and those falling into the second category. For example, 40% of the requests may be in the lower-priority category, while 60% are in the higher-priority category.

When a node receives an overload indication from one of its peers, it converts the Overload-Metric value to a value that applies to the first category of requests. For example, if the Overload-Metric for the applicable context is "10", and 40% of the requests are in the lower-priority category, then:

$$10 / 40 * 100 = 25$$

Or 25% of the requests in the first category can be dropped, with an overall reduction in sent traffic of 10%. The sender then drops 25% of all category 1 requests. This can be done stochastically, by selecting a random number for each sent packet between 1 to 100 (inclusive), and dropping any packet for which the resulting percentage is equal to or less than 25. In this set of circumstances, messages in the second category do not require any reduction to meet the requirement of 25% traffic reduction.

A reference algorithm is shown below, using pseudo-code.

```
cat1 := 80.0           // Category 1 --- subject to reduction
cat2 := 100.0 - cat1 // Category 2 --- Under normal operations
// only subject to reduction after category 1 is exhausted.
// Note that the above ratio is simply a reasonable default.
// The actual values will change through periodic sampling
// as the traffic mix changes over time.
```

```
while (true) {
    // We're modeling message processing as a single work queue
    // that contains both incoming and outgoing messages.
    msg := get_next_message_from_work_queue()

    update_mix(cat1, cat2) // See Note below

    switch (msg.type) {

    case outbound request:
        destination := get_next_hop(msg)
        oc_context := get_oc_scope(destination,msg)

        if (we are in overload) {
            add_overload_avps(msg)
        }

        if (oc_context == null) {
            send_to_network(msg) // Process it normally by sending the
            // request to the next hop since this particular
            // destination is not subject to overload
        }
        else {
            // Determine if server wants to enter in overload or is in
            // overload
            in_oc := extract_in_oc(oc_context)

            oc_value := extract_oc(oc_context)
            oc_validity := extract_oc_validity(oc_context)

            if (in_oc == false or oc_validity is not in effect) {
                send_to_network(msg) // Process it normally by sending
                // the request to the next hop since this particular
                // destination is not subject to overload. Optionally,
                // clear the oc context for this server (not shown).
            }
            else { // Begin perform overload control
                r := random()
                drop_msg := false

                if (cat1 >= cat2) {
                    category := assign_msg_to_category(msg)
                    pct_to_reduce_cat2 := 0
                    pct_to_reduce_cat1 := oc_value / cat1 * 100
                    if (pct_to_reduce_cat1 > 100) {
                        // Get remaining messages from category 2
                        pct_to_reduce_cat2 := 100 - pct_to_reduce_cat1
                        pct_to_reduce_cat1 := 100
                    }
                }
            }
        }
    }
```

```
    }

    if (category == cat1) {
        if (r <= pct_to_reduce_cat1) {
            drop_msg := true
        }
    }
    else { // Message from category 2
        if (r <= pct_to_reduce_cat2) {
            drop_msg := true
        }
    }
}
else { // More category 2 messages than category 1;
      // indicative of an emergency situation. Since
      // there are more category 2 messages, don't
      // bother distinguishing between category 1 or
      // 2 --- treat them equal (for simplicity).
    if (r <= oc_value)
        drop_msg := true
}

if (drop_msg == false) {
    send_to_network(msg) // Process it normally by
    // sending the request to the next hop
}
else {
    // Do not send request downstream, handle locally by
    // generating response (if a proxy) or treating as
    // an error (if a user agent).
}
} // End perform overload control
}

end case // outbound request

case outbound answer:
    if (we are in overload) {
        add_overload_avps(msg)
    }
    send_to_network(msg)

end case // outbound answer

case inbound answer:
    create_or_update_oc_scope() // For the specific server
    // that sent the answer, create or update the oc scope;
    // i.e., extract the values of the overload AVPs
```



```

        // and store them in the proper scopes for later use.
        process_msg(msg)

    end case // inbound answer
    case inbound request:
        create_or_update_oc_scope()

        if (we are not in overload) {
            process_msg(msg)
        }
        else { // We are in overload
            if ( connection supports overload)
                process_msg(msg)
            }
            else { // Sender does not support oc
                if (local_policy(msg) says process message) {
                    process_msg(msg)
                }
                else {
                    send_answer(msg, DIAMETER_PEER_IN_OVERLOAD)
                }
            }
        }
    end case // inbound request
}

```

A simple way to sample the traffic mix for category 1 and category 2 is to associate a counter with each category of message. Periodically (every 5-10s), get the value of the counters and calculate the ratio of category 1 messages to category 2 messages since the last calculation.

Example: In the last 5 seconds, a total of 500 requests were scheduled to be sent. Assume that 450 out of 500 were messages subject to reduction and 50 out of 500 were classified as requests not subject to reduction. Based on this ratio, cat1 := 90 and cat2 := 10, or a 90/10 mix will be used in overload calculations.

Of course, this scheme can be generalized to include an arbitrary number of priorities, depending on how many different classes of messages make sense for the given application.

5. Diameter AVPs for Overload

NOTE: THE AVP NUMBERS IN THIS SECTION ARE USED FOR EXAMPLE PURPOSES ONLY. THE FINAL AVP CODES TO BE USED WILL BE ASSIGNED BY IANA DURING THE PUBLICATION PROCESS, WHEN AND IF THIS DOCUMENT IS PUBLISHED AS AN RFC.

Attribute Name	AVP Code	Sec. Def.	Data Type	MUST	MUST NOT
Load-Info	1600	5.1	Grouped		M,V
Supported-Scopes	1601	5.2	Unsigned64		M,V
Overload-Algorithm	1602	5.3	Enumerated		M,V
Overload-Info-Scope	1603	5.4	OctetString		M,V
Overload-Metric	1604	5.5	Unsigned32		M,V
Period-Of-Validity	1605	5.6	Unsigned32		M,V
Session-Group	1606	5.7	UTF8String		M,V
Load	1607	5.8	Unsigned32		M,V

5.1. Load-Info AVP

The Load-Info AVP (AVP code 1600) is of type Grouped, and is used as a top-level container to group together all information pertaining to load and overload information. Every Load-Info AVP MUST contain one Overload-Information-Scope AVP, and one Overload-Metric AVP.

The Grouped Data field of the Load-Info AVP has the following CCF grammar:

```

< Load-Info > ::= < AVP Header: 1600 >
                  < Overload-Metric >
                  * { Overload-Info-Scope }
                  [ Supported-Scopes ]
                  * [ Overload-Algorithm ]
                  [ Period-Of-Validity ]
                  [ Session-Group ]
                  [ Load ]
                  * [ AVP ]

```

5.2. Supported-Scopes AVP

The Supported-Scopes AVP (AVP code 1601) is of type Uint64, and is used during capabilities exchange to indicate the scopes that a given node can receive on the connection. Nodes that support the mechanism defined in this document MUST include a Supported-Scopes AVP in all CER messages. It also MUST appear in any CEA messages sent in answer

to a CER message containing a Load-Info AVP. The Supported-Scopes AVP MUST NOT appear in any other message types. See Section 5.4 for an initial list of scopes.

The Supported-Scopes AVP contains a bitmap that indicates the scopes supported by the sender. Within the bitmap, the least significant bit indicates support for scope 1 (Destination-Realm), while the next least significant bit indicates support for scope 2 (Application-ID), and so on. In general, if we consider the bits to be numbered from 0 (LSB) to 63 (MSB), then any bit n corresponds to the scope type numbered $n+1$. This scheme allows for up to 64 total scopes to be supported. More formally, the bitmask used to indicate support for any specific context is calculated as follows (where the symbol " \ll " indicates a bit shift left):

$$\text{bitmask} = 1 \ll (n - 1)$$

For additional clarity, the bitmasks for the scopes defined in this document are as follows:

Scope	Bitmask	Scope
1	0x0000000000000001	Destination-Realm
2	0x0000000000000002	Application-ID
3	0x0000000000000004	Destination-Host
4	0x0000000000000008	Host
5	0x0000000000000010	Connection
6	0x0000000000000020	Session-Group
7	0x0000000000000040	Session

The advertisement process that makes use of the Supported-Scopes AVP is described in Section 3.1.

5.3. Overload-Algorithm AVP

The Overload-Algorithm AVP (AVP code 1602) is of type Enumerated, and is used to negotiate the algorithm that will be used for load abatement. The Overload-Algorithm AVP MAY appear in CER and CEA messages, and MUST NOT appear in any other message types. If absent, an Overload Algorithm of type 1 (Loss) is indicated. Additional values can be registered by other documents; see Appendix C.1. Initial values for the enumeration are as follows:

AVP Values	Attribute Name	Reference
0	Reserved	-
1	Loss	[RFC xxxx]

5.4. Overload-Info-Scope AVP

The Overload-Info-Scope AVP (AVP code 1603) is of type OctetString, and is used to indicate to which scope the Overload-Metric applies.

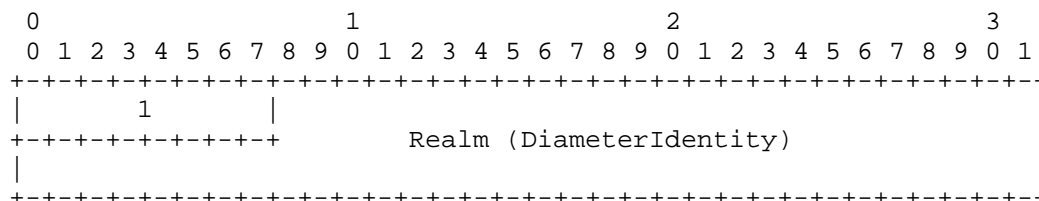
See Section 2 for a definition of the different scope types and a formal description of how they are applied. Other documents may define additional scopes; see Appendix C.2 for details.

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								
Scope										Details																													

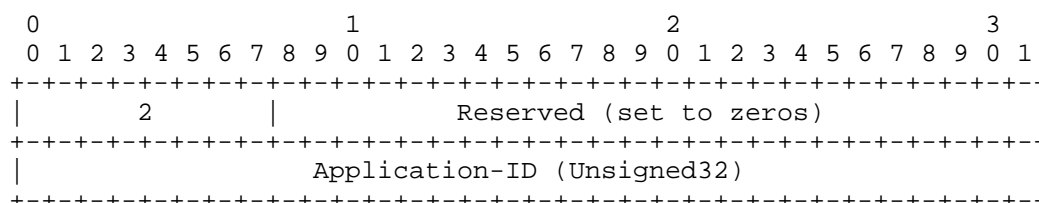
Scope	Attribute Name	Reference
0	Reserved	[RFC xxxx]
1	Destination-Realm	[RFC xxxx]
2	Application-ID	[RFC xxxx]
3	Destination-Host	[RFC xxxx]
4	Host	[RFC xxxx]
5	Connection	[RFC xxxx]
6	Session-Group	[RFC xxxx]
7	Session	[RFC xxxx]

Each Overload-Info-Scope has a different encoding, according to the identifier used to designate the corresponding scope. The formats for the seven scopes defined in this document are given in the following section.

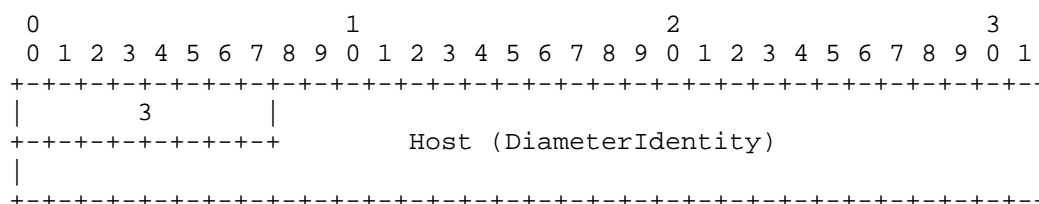
5.4.1. Realm Scope



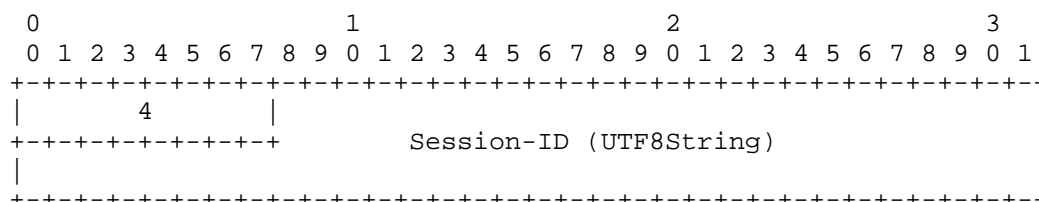
5.4.2. Application-ID Scope



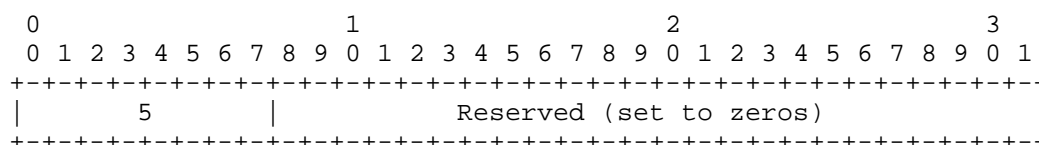
5.4.3. Host Scope



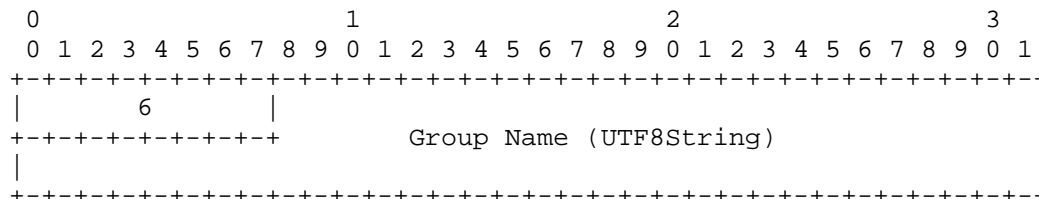
5.4.4. Session Scope



5.4.5. Connection Scope



5.4.6. Session Group Scope



5.5. Overload-Metric AVP

The Overload-Metric AVP (AVP code 1604) is of type Unsigned32, and is used as input to the load mitigation algorithm. Its definition and interpretation is left up to each individual algorithm, with the exception that an Overload-Metric of "0" always indicates that the node is not in overload (that is, no load abatement procedures are in effect) for the indicated scope.

5.6. Period-Of-Validity AVP

The Period-Of-Validity AVP (AVP code 1605) is of type Unsigned32, and is used to indicate the length of time, in seconds, the Overload-Metric is to be considered valid (unless overridden by a subsequent Overload-Metric in the same scope). It MUST NOT be present if the Overload-Metric is '0', and MUST be present otherwise.

5.7. Session-Group AVP

The Session-Group AVP (AVP code 1606) is of type UTF8String, and is used to assign a new session to the session group that it names. The Session-Group AVP MAY appear once in the answer to a session-creating request, and MUST NOT appear in any other message types.

5.8. Load AVP

The Load AVP (AVP code 1607) is of type Unsigned32, and is used to indicate the load level of the scope in which it appears. See Section 3.5 for additional information.

6. Security Considerations

A key concern for recipients of overload metrics and load information is whether the peer from which the information has been received is authorized to speak for the indicated scope. For scopes such as "Host" and "Connection", such authorization is obvious. For other scopes, such as "Application-ID" and "Realm", the potential for a

peer to maliciously or accidentally reduce traffic to a third party is evident. Implementations may choose to ignore indications from hosts which do not clearly have authority over the indicated scope; alternately, they may wish to further restrict the scope to apply only to the host from which the information has been received.

On the other hand, multiple nodes that are under the same administrative control (or a tightly controlled confederation of control) may be implicitly trusted to speak for all scopes within that domain of control. Implementations are encouraged to allow configuration of inherently trusted servers to which the foregoing restrictions are not applied.

Open Issue: There are almost certainly other security issues to take into consideration here. For example, we might need to include guidance around who gets to see our own load information, and potentially changing the granularity of information presented based on trust relationships.

7. IANA Considerations

This document defines new entries in several existing IANA tables. It also creates two new tables.

7.1. New Diameter AVPs

The following entries are added to the "AVP Codes" table under the "aaa-parameters" registry.

AVP Code	Attribute Name	Reference
1600	Load-Info	RFC xxxx
1601	Supported-Scopes	RFC xxxx
1602	Overload-Algorithm	RFC xxxx
1603	Overload-Info-Scope	RFC xxxx
1604	Overload-Metric	RFC xxxx
1605	Period-Of-Validity	RFC xxxx
1606	Session-Group	RFC xxxx
1607	Load	RFC xxxx

7.2. New Diameter Disconnect-Cause

The following entry is added to the "Disconnect-Cause AVP Values (code 273)" table in the "aaa-parameters" registry:

AVP Values	Attribute Name	Reference
128 [actual value TBD]	NEGOTIATION_FAILURE	RFC xxxxx

7.3. New Diameter Response Code

The following entry is added to the "Result-Code AVP Values (code 268) - Transient Failures" table in the "aaa-parameters" registry:

AVP Values	Attribute Name	Reference
4128 [actual value TBD]	DIAMETER_PEER_IN_OVERLOAD	RFC xxxxx

7.4. New Command Flag

The following entry is added to the "Command Flags" table in the "aaa-parameters" registry:

bit	Name	Reference
4	'O'verload	RFC xxxxx

7.5. Overload Algorithm Registry

This document defines a new table, to be titled "Overload-Algorithm Values (code 1602)", in the "aaa-parameters" registry. Its initial values are to be taken from the table in Section 5.3.

New entries in this table follow the IANA policy of "Specification Required." (Open Issue: The WG should discuss registration policy to ensure that we think this is the right balance).

7.6. Overload Scope Registry

This document defines a new table, to be titled "Overload-Info-Scope Values (code 1603)", in the "aaa-parameters" registry. Its initial values are to be taken from the table in Section 5.4.

New entries in this table follow the IANA policy of "Specification Required." (Open Issue: The WG should discuss registration policy to ensure that we think this is the right balance).

8. References

8.1. Normative References

- [I-D.ietf-dime-overload-reqs]
McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", draft-ietf-dime-overload-reqs-06 (work in progress), April 2013.
- [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.

8.2. Informative References

- [I-D.ietf-soc-overload-control]
Gurbani, V., Hilt, V., and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control", draft-ietf-soc-overload-control-12 (work in progress), February 2013.
- [I-D.ietf-soc-overload-rate-control]
Noel, E. and P. Williams, "Session Initiation Protocol (SIP) Rate Control", draft-ietf-soc-overload-rate-control-04 (work in progress), April 2013.
- [RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.
- [RFC6357] Hilt, V., Noel, E., Shen, C., and A. Abdelal, "Design Considerations for Session Initiation Protocol (SIP) Overload Control", RFC 6357, August 2011.

Appendix A. Acknowledgements

This work was inspired by and borrows heavily from the SIP overload control mechanism described in [I-D.ietf-soc-overload-control]. The authors of this document are deeply grateful to the editor and authors of that work, as well as its many contributors.

Thanks to Ben Campbell for significant input to the initial mechanism design. The author also thanks Martin Dolly, Bob Wallace, John Gilmore, Matt McCann, Jonathan Palmer, Kedar Karmarkar, Imtiaz Shaikh, Jouni Korhonen, Uri Baniel, Jianrong Wang, Brian Freeman, and

Eric Noel for early feedback on the mechanism.

Appendix B. Requirements Analysis

This section analyzes the mechanism described in this document against the set of requirements detailed in [I-D.ietf-dime-overload-reqs].

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

Compliant. The mechanism uses new AVPs piggybacked on existing Diameter messages to exchange load and overload information.

REQ 2: The mechanism MUST allow Diameter nodes to support overload control regardless of which Diameter applications they support. Diameter clients 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.

Compliant. Piggybacked AVPs conveying overload control information is sent on every Diameter message to compliant peers, without regard to its Application-ID. The use of the Application-ID scope allows information relevant to one application to be piggybacked on messages for other applications.

Information sent to peers includes load and overload information for use by overload control algorithms, intended for graceful overload mitigation. The mechanism is hop-by-hop and has provisions for agents to forward or aggregate load and overload information towards clients and servers so that each element can have appropriate information for graceful overload control.

REQ 3: The overload control mechanism 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 an overload control mechanism.

Compliant. The mechanism provides information nodes use to affect the impacts of overload according to agreed upon algorithms. By controlling or reducing traffic sent towards overloaded elements, using overload control information as described in the mechanism, the effects of overload can be limited. Use of scopes provides a means to minimize the impact of overload mitigation, increasing overall useful throughput during overload conditions.

- 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 mechanism **MUST** allow each side of a connection to independently inform the other of its overload status.

Compliant. Overload control information can be piggybacked on any Diameter message. This applies for requests and answers sent from either side of a connection.

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

Compliant. The mechanism makes no assumptions as to how peers are determined. Discovery of supporting peers is accomplished as part of the normal capabilities exchange and does not affect how or where these exchanges occur.

- REQ 6: The mechanism 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 mechanism, rather than to require this knowledge to be configured at each node.

Compliant. The mechanism adds information to the existing Diameter capabilities exchange mechanism for determining peer support and overload control characteristics. Since this is accomplished dynamically at the start of connections, no provisioning is required to establish which peers support the mechanism and in what fashion. Implementations are free to add configuration for local policy and other control of the mechanism, but this is not required.

- REQ 7: The overload control mechanism and any associated default algorithm(s) MUST ensure that the system remains stable. At some point after an overload condition has ended, the mechanism 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 mechanism MUST allow nodes to shed load without introducing non converging oscillations during or after an overload condition.

Compliant. It is possible for an implementation using this to meet this requirement, and the hop-by-hop nature limits the impact of overload control actions. Additional guidance is provided for implementors on sending of the Overload-Metric and its implications for the closed loop control system created by this mechanism.

- REQ 8: Supporting nodes MUST be able to distinguish current overload information from stale information, and SHOULD make decisions using the most currently available information.

Compliant. The mechanism provides for rapid updates of overload control information as well as having timeouts on the validity of overload information that must be provided by senders.

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

Compliant. The mechanism uses piggybacked information transfer, which will generally result in the ability to transfer information on a similar rate to loading. It also provides for triggering the use of DWR with piggybacked information for quiescent connections.

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

Compliant. The mechanism provides for rapid updates of overload control information, including abatement information, as well as mandatory timeouts on the validity of overload information that must be provided by senders (it is soft state). Additionally, the mechanism provides for sending a DWR with piggybacked information to inform of overload abatement more quickly.

- REQ 11: The overload control mechanism MUST be able to operate in networks of different sizes.

Compliant. The hop-by-hop nature of the mechanism restricts the impacts that large networks might have on the ability of nodes to deal with overload control information, as well as restricting the signaling needed to convey overload information. The use of piggybacked information transfer limits the additional messaging imposed by the mechanism for large and small networks and has the characteristic of scaling with the amount of Diameter traffic on a network. Additionally, the dynamic nature of the capabilities exchange reduces the provisioning burden that can be incurred at large scales.

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

Compliant. The mechanism provides for information about such issues to be conveyed in order for nodes to take appropriate action to mitigate the situation and prevent cascades.

- REQ 13: The mechanism MUST NOT introduce substantial additional work for 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. Existing messaging is likely to have the characteristic of increasing as an overload condition approaches, allowing for the possibility of increased feedback for information piggybacked on it.

Compliant. The mechanism requires sending load and overload information on all messages exchanged with compliant peers. It does not, however, require that the information be recalculated or updated with each message. The update frequency is up to the implementation, and each implementation can make decisions on balancing the update of overload information along with its other priorities. It is expected that using a periodically updated grouped AVP added to all messages sent to compliant peers will not add substantial additional work. Piggyback base transport also does not require composition, sending, or

parsing of new Diameter messages for the purpose of conveying overload control information.

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

Compliant. The use of piggybacked information transport by the mechanism allows for overload control information to be sent at the same rate as the normal traffic. It is presumed that the rate of normal traffic will go up as nodes approach, or enter, overload. Additionally, DWR messages may be proactively triggered with piggybacked overload control information to provide overload control information transfer in an ad hoc fashion.

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

Compliant. The mechanism does not require interaction with any underlying congestion control. It relies solely on piggybacked transport and does not request or recommend changes in how the underlying connections are performed.

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

Compliant. The mechanism specifies behavior for dealing with non-supporting elements.

- REQ 17: In a mixed environment with nodes that support the overload control mechanism and that do not, the mechanism MUST result in at least as much useful throughput as would have resulted if the mechanism were not present. It SHOULD result in less severe congestion in this environment.

Compliant. When dealing with supporting, and non-supporting nodes, the mechanism specifies behavior that attempts to apply relevant information to decisions on sending to non-compliant hosts. This behavior should result in reductions in traffic that increase the

likelihood of successful overload mitigation in mixed networks.

- REQ 18: In a mixed environment of nodes that support the overload control mechanism and that do not, the mechanism 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 mechanism MUST NOT unfairly benefit from the mechanism. The mechanism specification SHOULD provide guidance to implementors for dealing with elements not supporting overload control.

Compliant. When dealing with supporting, and non-supporting nodes, the mechanism specifies behavior that attempts to apply relevant information to decisions on sending to non-compliant hosts. This allows nodes to treat non-supporting elements in a similar, and fair, fashion relative to non-supporting elements.

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

Compliant. Scoping of overload information to realms is explicitly specified by the mechanism. There are no requirements imposed by the mechanism that would prevent overload control information from crossing between adjacent nodes that were in separate administrative domains.

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

Compliant. A new grouped AVP conveys all overload control information, and this is transported on existing messages that are not related to overload control. No existing Diameter error codes are used by the mechanism. One new transient error code is defined by the mechanism.

- 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 congestion. The mechanism MUST result in at least as much useful throughput as would have resulted if the overload control mechanism was not in place.

Compliant. Procedures are defined cases where supporting nodes become too overloaded to send overload information. No retries or sending of additional messages are required during overload that would reduce useful throughput in these situations.

- REQ 22: The mechanism 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 mechanism provides a 32 bit overload severity indication. Interpretation of the value is specific to the algorithm being employed. In the case of the mandatory to implement loss algorithm, the values 0-100 are used to progressively control the amount of traffic dropped.

- REQ 23: The mechanism 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.

Compliant. The mechanism provides information so that a load balancing node can determine that an upstream node is in overload. Additionally, it provides load information that can be used as input for balancing decisions.

- REQ 24: The mechanism MUST provide a mechanism for indicating load levels even when not in an overloaded condition, to assist nodes making decisions to prevent overload conditions from occurring.

Compliant. The mechanism provides load information in each message as well as guidelines for implementing the determination of load to be sent.

- REQ 25: The base specification for the overload control mechanism 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. Some guidance is provided for priority selection and how to deal with different priority messages is described in an example algorithm implementation.

- REQ 26: The mechanism 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. The mechanism does not place restrictions on how decisions are made to prioritize messages.

- REQ 27: The overload control mechanism 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.

Compliant. The hop-by-hop nature of the mechanism allows existing Diameter security mechanisms to be used for securing the connections between peers. ***Detailed analysis by persons with security expertise would be beneficial.***

- REQ 28: The mechanism 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 mechanism to detect, or take countermeasures against, malicious nodes.

Compliant. Using a hop-by-hop mechanism limits the scope of potentially malicious information. Guidance is provided for trust, in particular relative to scopes. Additional specification around trust relationships could be useful to clarify authorization of overload control information. ***Detailed analysis by persons with security expertise would be beneficial.***

- 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 overload control mechanism itself will help prevent overload from that peer node.

Compliant. The mechanism provides guidance for authorization decisions and takes no action to restrict local policy when dealing with authorization.
Detailed analysis by persons with security expertise would be beneficial.

- REQ 30: The mechanism 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 mechanism allows for local policy overrides for the bulk of its behavior.

- 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 mechanism 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 mechanism 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.

Compliant. The mechanism allows for flexible specification on the scope that overload control information applies to. It also allows for additional scopes to be specified as extensions.

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

Compliant. The mechanism allows for new algorithms to be specified as extensions. It provides an AVP for communicating overload information that can be interpreted differently by different algorithms. It also provides for extension of information transmitted.

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

Compliant. The mechanism specifies the drop algorithm as mandatory to implement.

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

Not Compliant. Additional analysis is needed.

Appendix C. Extending the Overload Mechanism

This specification includes two key extension points to allow for new behaviors to be smoothly added to the mechanism in the future. The following sections discuss the means by which future documents are expected to extend the mechanism.

C.1. New Algorithms

In order to provide the ability for different means of traffic abatement in the future, this specification allows for descriptions of new traffic reduction algorithms. In general, documents that define new algorithms need to describe externally-observable node behavior in sufficient detail as to allow interoperation.

At a minimum, such description needs to include:

1. The name and IANA-registered number for negotiating the algorithm (see Section 5.3).
2. A clear description of how the Overload-Metric AVP is to be interpreted, keeping in mind that "0" is reserved to indicate that no overload condition exists.
3. An example, proof-of-concept description (preferably in pseudo-code) of how nodes can implement the algorithm.

New algorithms must be capable of working with all applications, not just a subset of applications.

It is generally expected that new algorithms will make use of the available overload control information as specified in this document. However, if additional information is needed, the Load-Info AVP allows for additional optional AVPs to be included. It is recommended that designers of any new AVPs defined for this purpose consider reusing existing AVPs first, and also design their AVPS so that they may be reused by others when possible.

C.2. New Scopes

Because it is impossible to foresee all the potential constructs that it might be useful to scope operations to for the purposes of overload, we allow for the registration of new scopes.

At a minimum, such description needs to include:

1. The name and IANA-registered number for negotiating and indicating the scope (see Section 5.4).
2. A syntax for the "Details" field of the Overload-Info-Scope AVP, preferably derived from one of the base Diameter data types.
3. An explicit and unambiguous description of how both parties to the overload control mechanism can determine which transactions correspond to the indicated scope.
4. A clear and exhaustive list that extends the one in Section 2.2, indicating exactly which combinations of scopes are allowed with the new scope. This list must take into account all of the IANA-registered scopes at the time of its publication.

It is acceptable for new scopes to be specific to constructs within one or several applications. In other words, it may be desirable to define scopes that can be applied to one kind of application while not making sense for another. Extension documents should be very clear that such is the case, however, if they choose to do so.

Appendix D. Design Rationale

The current design proposed in this document takes into account several trade-offs and requirements that may not be immediately obvious. The remainder of this appendix highlights some of the potentially more controversial and/or non-obvious of these, and attempts to explain why such decisions were made they way they were.

That said, none of the following text is intended to represent a line in the sand. All of the decisions can be revisited if necessary, especially if additional facts are brought into the analysis that change the balance of the decisions.

D.1. Piggybacking

The decision to piggyback load information on existing messages derives primarily from REQ 14 in [I-D.ietf-dime-overload-reqs]: "The mechanism SHOULD provide for increased feedback when traffic levels increase. The mechanism MUST NOT do this in such a way that it increases the number of messages while at high loads."

If we were to introduce new messaging -- say, by defining a new overload control Application -- then a node in overload would be required to generate more messages at high load in order to keep overload information in its peers up-to-date.

If further analysis determines that other factors are ultimately more important than the provisions of REQ 14, several factors would need to be considered.

First and foremost would be the prohibition, in the base Diameter specification ([RFC6733]), against adding new commands to an existing application. Specifically, section 1.3.4 stipulates: "a new Diameter application MUST be created when one or more of the following criteria are met:... A new command is used within the existing application either because an additional command is added, an existing command has been modified so that a new Command Code had to be registered, or a command has been deleted." Because of this stipulation, the addition of new command codes to existing applications would require registration of entirely new application IDs for those applications to support overload control. We consider this to be too disruptive a change to consider.

By the author's reading, there is no provision that exempts the "Diameter Common Messages" Application (Application ID 0) from the above clauses. This effectively prohibits the addition of new messages to this Application. While it may be theoretically possible to specify behavior that hijacks the DWR/DWA watchdog messages for the purpose of overload control messaging, doing so requires a complete redefinition of their behavior and, fundamentally, their semantics. This approach seems, at first blush, to be an unacceptable change to the base Application.

The remaining approach -- defining a new application for overload control -- has some promise, if we decide not to fulfill REQ 14. It remains to be seen whether the users of the Diameter protocol, including other SDOs who define applications for Diameter, are willing to specify the use of multiple Diameter Applications for use on a single reference point.

D.2. Load AVP in All Packets

Some have questioned the currently specified behavior of message senders including a Load AVP in every message sent. This is being proposed as a potential performance enhancement, with the idea being that message recipients can save processing time by examining arbitrarily selected messages for load information, rather than looking for a Load AVP in every message that arrives. Of course, to enable this kind of sampling, the Load AVP must be guaranteed to be present; otherwise, attempts to find it will occasionally fail.

The reciprocal approach, of sending a Load AVP only when the Load has changed (or changed by more than a certain amount), requires the recipient to search through the Load-Info grouped AVP in every message received in order to determine whether a Load AVP is present.

On a cursory analysis, we determined that appending a Load AVP to each message is fundamentally a cheaper operation than traversing the contents of each Load-Info AVP to determine whether a Load AVP is present.

If a later decision is made to require examination of each message to determine whether it include a Load AVP, we may be able to obtain some efficiencies by requiring Load to be the first AVP in the Load-Info AVP.

D.3. Graceful Failure

Some commenters have raised the question of whether a node can reject an incoming connection upon recognizing that the remote node does not support the Diameter overload control mechanism. One suggestion has been to add a response code to indicate exactly such a situation.

So far, we have opted against doing so. Instead, we anticipate an incremental deployment of the overload control mechanism, which will likely consist of a mixture of nodes that support and node that do not support the mechanism. Were we to allow the rejection of connections that do not support the mechanism, we would create a situation that necessitates a "flag day," on which every Diameter node in a network is required to simultaneously, and in perfect synchronization, switch from not supporting the overload mechanism, to supporting it.

Given the operational difficulty of the foregoing, we have decided that defining a response code, even if optional, that was to be used to reject connections merely for the lack of overload control support, would form an attractive nuisance for implementors. The result could easily be a potential operational nightmare for network

operators.

Authors' Addresses

Adam Roach
Mozilla
Dallas, TX
US

Email: adam@nostrum.com

Eric McMurry
Tekelec
Dallas, TX
US

Email: emcmurphy@computer.org

