Diameter Maintenance and Extensions (DIME)
Internet-Draft
Intended status: Standards Track
Expires: June 20, 2015

S. Donovan
Oracle
E. Noel
AT&T Labs

December 17, 2014

# Diameter Overload Rate Control
# draft-ietf-dime-doic-rate-control-00.txt

## Abstract

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

## Requirements

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

## Status of This Memo

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 June 20, 2015.

## Copyright Notice

Copyright (c) 2014 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

This document defines a new Diameter overload control algorithm.

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

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

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

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

One of the benefits of a rate based algorithm is that it better handles spikes in traffic. Instead of sending a request to throttle a percentage of the traffic, the rate approach allows the reporting node to specify the maximum number of Diameter requests per second that can be sent to the reporting node. For instance, in this example, the reporting node could send a rate based request

specifying the maximum transactions per second to be 90. The reacting noce will send the 90 regardless of whether it is receiving 100 or 1000 service requests per second.

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

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

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

## 2. Terminology and Abbreviations

Diameter Node

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

Diameter Endpoint

An RFC6733 Diameter Client and RFC6733 Diameter Server.

DOIC Node

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

Reporting Node

A DOIC Node that sends a DOIC overload report.

Reacting Node

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

## 3. Interaction with DOIC report types

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

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

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

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

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

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

## 4. Capability Announcement

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

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

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

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

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

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

For host or realm reports the selected algorithm MUST be reflected in the OC-Feature-Vector AVP sent as part of the OC-Selected-Features AVP included in answer messages for transaction where the request

contained an OC-Supported-Features AVP. This is per the precedures defined in [I-D.ietf-dime-ovli].

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

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

## 5. Overload Report Handling

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

### 5.1. Reporting Node Overload Control State

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

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

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

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

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

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

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

## 5.2. Reacting Node Overload Control State

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

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

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

## 5.3. Reporting Node Maintenance of Overload Control State

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

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

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

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

## 5.4. Reacting Node Maintenance of Overload Control State

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

## 5.5. Reporting Node Behavior for Rate Abatement Algorithm

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

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

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

## 5.6. Reacting Node Behavior for Rate Abatement Algorithm

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

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

## 6. Rate Abatement Algorithm AVPs

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

## 6.1. OC-Supported-Features AVP

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

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

### 6.1.1. OC-Feature-Vector AVP

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

OLR_RATE_ALGORITHM (0x0000000000000004)

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

## 6.2. OC-OLR AVP

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

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

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

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

### 6.2.1. OC-Rate AVP

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

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

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

### 6.3. Attribute Value Pair flag rules

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

## 7. Rate Based Abatement Algorithm

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

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

### 7.1. Overview

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

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

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

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

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

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

## 7.2. Reporting Node Behavior

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

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

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

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

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

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

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

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

### 7.3. Reacting Node Behavior

#### 7.3.1. Default algorithm

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A reference algorithm is shown below.

No priority case:

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

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

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

### 7.3.2. Priority treatment

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

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

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

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

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

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

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

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

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

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

A reference algorithm is shown below.

Priority case:

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

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

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

### 7.3.3. **Optional enhancement: avoidance of resonance**

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

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

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

if X' > 0, then u=0

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

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

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

This method has the following characteristics:

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

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

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

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

## 8.  IANA Consideration

TBD

## 9.  Security Considerations

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

## 10.  Acknowledgements

## 11.  References

## 11.1. Normative References

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

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

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

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

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

## 11.2. Informative References

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

Authors' Addresses

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

Email: srdonovan@usdonovans.com

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

Email: ecnoel@research.att.com