Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 4, 2015
July 3, 2014

S. Donovan
B. Campbell
Oracle

Analysis of Agent Use Cases for Diameter Overload Information Conveyance (DOIC)

draft-donovan-doic-agent-cases-00

Abstract

The Diameter Overload Information Conveyance (DOIC) solution describes a mechanism for exchanging information about Diameter Overload among Diameter nodes. A DOIC node is a Diameter node that acts as either a reporting node are a reacting node. A reporting node originates overload reports, requesting reacting nodes to reduce the amount of traffic sent. DOIC allows Diameter agents to act as reporting nodes, reacting nodes, or both, but does not describe agent behavior. This document explores several use cases for agents to participate in overload control, and makes recommendations for certain agent behaviors to be added to DOIC.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Diameter Overload Information Conveyance (DOIC) [I-D.ietf-dime-ovli] solution describes a mechanism for exchanging diameter overload information among Diameter nodes. DOIC defines the concept of a DOIC endpoint (referred to as a "DOIC node" in this document.) A DOIC node is a Diameter node that acts as either a

reporting node or a reacting node. A reporting node originates overload reports, requesting reacting nodes to reduce the offered load. An overload report has a "type". The type of overload report determines the scope of the request for traffic reduction, and possibly other semantics.

DOIC nodes do not necessarily correspond to Diameter clients and servers. Any Diameter node that supports DOIC is a DOIC node. This includes Diameter agents, as well as Diameter clients and servers. However, DOIC does not currently describe how agents should behave as part of an overload control solution. This document explores several use cases for agents to participate in overload control, and makes recommendations for certain agent behaviors to be added to DOIC.

1.1. Terminology and Abbreviations

Diameter Node: A Diameter client, agent or server.

DOIC node: A Diameter node that supports DOIC. A DOIC node can simultaneously be both a reacting node and a reporting node.

Reacting node: A DOIC node that can receive overload reports from a reporting node and ensures that the overload reports are honored.

Reporting node: A DOIC node that can send overload reports, requesting overload abatement.

Abating node: A reacting node that directly performs overload abatement.

Transaction Client (TC) A diameter node that originates a Diameter request. This is distinct from "Diameter Client" as used in RFC 6733. Note that a Diameter Server acts as a TC if and when it originates a request towards a Diameter Client.

TransactionServer (TS) A diameter node that consumes a Diameter request, and responds with a Diameter answer. This is distinct from "Diameter Server" as used in RFC 6733. Note that a Diameter Client acts as a TS if and when it answers a request sent by a Diameter Server.

Client Application The application that uses Diameter for various Authentication, Authorization, and/or Accounting (AAA) functions. For example, a Network Access Server (NAS) performs certain network attachment services, detachment services, packet forwarding, etc. These are collectively the NAS client application, which depends on Diameter for AAA services.

Overload Abatement: Actions taken by a reacting node to reduce the load offered to an overloaded Diameter node. The specific actions required to perform overload abatement are described by the DOIC algorithm. Overload abatement actions may involve local traffic reduction, or delegation of actions towards the client. Local traffic reduction can be achieved by either throttling a request or routing a request to a different destination.

Throttling Overload abatement through the rejection of some number of requests. Throttling at an agent requires the agent to reject requests with appropriate error codes. Throttling at a transaction client requires the client to indicate appropriate errors to the client application

Diversion Overload abatement through the routing of some number of requests away from an overloaded node towards one or more appropriate nodes that are less-overloaded.

2. Deployment Architectures

This section outlines the deployment architectures used to determine agent related Diameter DOIC requirements.

These deployment architectures include the use of Diameter agents to route Diameter requests between Diameter clients and Diameter servers.

In all cases, a small number of client and server nodes are shown for simplicity. Adding additional clients and/or servers does not change the fundamental characteristics of the deployments.

Figure 1 shows an architecture with a single agent sitting between Diameter clients and Diameter servers.

```
+--+                +--+
|C1|-----      -----|S1|
+--+     \+--+/     +--+
          |A1|
+--+     /+--+\     +--+
|C2|-----      -----|S2|
+--+                +--+
```

Figure 1

Figure 2 shows an architecture with multiple agents sitting between Diameter clients and Diameter servers.

```
+--+                        +--+
|C1|-----               -----|S1|
+--+     \+--+      +--+/     +--+
          |A1|------|A2|
+--+     /+--+      +--+\     +--+
|C2|-----               -----|S2|
+--+                        +--+
```

Figure 2

This document focuses on use cases that involve agents, and does not therefore include scenarios with no agent(s) between the transaction client and transaction server. It is, however, important that any changes to the DOIC solution introduced to support networks that contain agents also work when there is no agent sitting between Diameter clients and servers.

3. Diameter Routing

Diameter supports two primary methods [RFC6733] for nodes to select the next hop for a request . Normally, Diameter nodes base peer-selection on the Destination-Realm and Application-Id AVP values. That is, they select a next hop from their Diameter peer table entry that matches the realm and application of the request. For the sake of this analysis, we refer to requests routed by this method as "realm-routed requests"

A Diameter TC may also control the final destination of a request by inserting a Destination-Host AVP. When a node forwards a request that includes Destination-Host, it checks to see if it has a matching Diameter identity in its peer table. If so, it forwards the request to that peer. Otherwise, it follows the normal peer-selection for the realm and application. We refer to requests routed this way as "host-routed requests".

In general, throttling (Section 4) is the the only abatement technique that works for host-routed requests. Diversion (Section 4) is typically not possible, since only a single TS can handle any given request.

There may be some exceptions to this rule. For example, a node might have multiple peer table entries that share the same

Diameter Identity. A node might map Diameter identities in a way that results in multiple next-hop destinations for a given Destination-Host value.

On the other hand, diversion is often useful for abating realm-routed traffic. Since realm-routed requests are not bound to a particular TS, it is often be possible to divert a number of them to other servers that are less overloaded.

4. Overload Abatement Methods

When a Diameter node becomes overloaded, there often must be a reduction of the number of both realm-routed and host-routed requests, in order to have the desired reduction of the overall offered-load. We refer to this reductions as "Overload Abatement".

There are two ways to perform overload abatement. The first is to reject some number of Diameter requests, also known as "throttling". When a TC throttles traffic, it rejects or defers certain client application requests, as appropriate for the client application. When an agent or TS throttles traffic, it rejects Diameter requests with an appropriate error code, so that the TC can behave correctly at the client application layer.

The second way to abate overload is to move some number of requests from an overloaded node to one or more eligible nodes that are less overloaded. For the purposes of this draft, we refer to this abatement method as "Diversion".

Either method, separately or in combination, continue over the duration of the overload condition.

There are a few architectural principles that should be considered when building Diameter networks to be resilient to overload, or when deploying DOIC into existing Diameter network.

All things being equal, diversion is better than throttling. Diversion potentially allows more requests to succeed, which will almost always have less negative impact on the client application. However, there are situations where diversion is not possible. For example, diversion is usually not possible for host-routed requests (see Section 3). Diversion may not be helpful if all potential destinations are overloaded. If proprietary load balancing mechanisms are in use, diversion for DOIC purposes may be redundant with those mechanisms.

If diversion is performed, the diversion should occur as close as possible to the TS, but not at the TS itself (since this would defeat

the point of overload abatement.) Diversion should optimally occur at an immediate peer to the TS; that is, a node that shares a direct transport connection with the TS. A directly connected peer will have the most knowledge of alternative destinations and their current loads. If nodes further from the TS attempt to diversion, topology knowledge and overload state knowledge must be pushed further down the chain. Even then, a node usually cannot control how a realm-routed request might be routed upstream of the immediate peer.

Throttling should occur at, or as close as possible, to the TC. The TC has the best knowledge of the client application and its current state. The TC can choose to reject requests that have the lease impact on the client application, or provide the most effect for traffic reduction over time. Furthermore, throttling at the TC completely avoids Diameter overhead for rejected requests. Each additional hop traversed by requests that will eventually be rejected increases the impact of those requests.

5. DOIC Use Cases

This section outlines example use cases involving agents. Each of these use cases is evaluated with the goal of identifying any required changes to [I-D.ietf-dime-ovli] needed to support the use case.

The following is the list of use cases considered. This is not an exhaustive list of DOIC use cases but is rather a list of use cases identified by the authors as being impacted by the presence of agents in the deployment.

- Simple Agent - Overload capability announcement and overload report handling in a deployment with a single agent as illustrated in Figure 1. In this case all Diameter nodes are assumed to support DOIC. This use case is discussed in Section 5.1.

  This use case includes four sub-cases:

  1. OC Capability Announcement where the TC and Agent support the same OC capabilities.

  2. Host overload report handling for host-routed requests. This case illustrates throttling of host-routed requests at the transaction client and throttling of realm-routed reqeusts at the agent.

  3. Host overload report handling realm-routed requests when there is a second TS to which requests can be diverted when one of the servers is in an overload state.

4. Multiple host overload reports resulting in a realm overload report.

- Mixed Capabilities - Overload capability announcement and overload report handling in deployments where agents support capabilities that are not included in the set of capabilities advertised by reacting nodes. This use case is discussed in Section 5.2.

  This use case illustrates one scenario where an agent consumes an overload report and replaces it with a new overload report of a different type.

- Non-supporting DOIC nodes - Agent behavior in the face of Diameter nodes that do not support the DOIC solution. These use cases are addressed in Section 5.3. There are four sub-use cases that are addressed:

  * Non-supporting TC. In this case a DOIC supporting agent handles overload abatement on behalf of the non-supporting TC. An agent or a reporting node can detect if there is a reacting node in the path of a request by the presence of the OC-Supported-Features AVP in the request message. This use case is discussed in Section 5.3.1.

  * Non-supporting TS. In this case a DOIC supporting agent may act as the reporting node on behalf of the TS. In this case a DOIC supporting agent can detect if there is a reporting node in the path of the transaction by the presence of the OC-Supported-Features AVP in the answer message for the transaction. This use case is discussed in Section 5.3.2.

  * Non-supporting agent between the TC and a supporting agent.

  * Non-supporting agent between a supporting agent and the TS. In this case, the agent that supports DOIC cannot reliably divert requests as a result of a host report. This use case is discussed in Section 5.3.3.

    This use case illustrates when this deployment scenario is not recommended.

- Inter domain or untrusted node authorization.

  This use case illustrates one case where a node needs to know if an OC-S-F AVP came from a supported peer, or was forwarded by a non-supporting peer. This use case is discussed in Section 5.4.

5.1. Simple Agent

This section addresses overload capability announcement and overload report handling in a deployment with a single agent as illustrated in Figure 1.

This use case assumes that all nodes support DOIC and that all nodes support the same set of overload features.

This use case includes four sub-cases:

1. OC Capability Announcement where the TC and Agent support the same OC capabilities.

2. Host overload report handling for host-routed requests. This case illustrates throttling of host-routed requests at the transaction client and throttling of realm-routed reqeusts at the agent.

3. Host overload report handling realm-routed requests when there is a second TS to which requests can be diverted when one of the servers is in an overload state.

4. Multiple host overload reports resulting in a realm overload report.

5.1.1. Capability Announcement

This section explores capability announcement for the simple agent use case.

This use case assumes that the capabilities supported by the TC and those supported by the agent are the same. (A scenario with differing capabilities is supported in discussed in Section 5.2.)

Figure 3 shows the message flow for this use case.

The nomenclature OC-S-F:x is short for OC-Supported-Feature with the ":x" indicating the Diameter node that inserted the AVP into the message.

```
         +--+                       +-+                       +--+
         |TC|                       |A|                       |TS|
         +--+                       +-+                       +--+
          |                          |                          |
     1>   |-- xxR OC-S-F:C---------->|                          |
          |                          |                          |
     2>   |                          |-- xxR OC-S-F:C---------->|
          |                          |                          |
     3>   |                          |<---------- xxA OC-S-F:S--|
          |                          |                          |
     4>   |<-------- xxA OC-S-F:S----|                          |
          |                          |                          |
```

Figure 3

1. The transaction client (TC) originates a request. The TC supports DOIC and, as such, includes the OC-Supported-Features AVP in all requests. The OC-S-F AVP contains the clients capabilities.

2. The agent inspects the OC-S-F AVP and determines that the agent supports the same set of OC features. The agent relays the request unchanged to the server.

   Note: It is an open question whether the agent needs to include an indication that it also supports DOIC or if attribution of the OC-S-F is needed. One could also question whether the agent forwarded OC-S-F:C unchanged, rather than consuming the OC-S-F, and inserted another identical one. This is likely a purely philosophical difference, but might impact the inter-domain authorization use case (Section 5.4).

3. The transaction server (TS), acting as the reporting node, inspects the OC-S-F AVP in the request and generates an OC-S-F to be included in the answer message. This is done according to the behavior defined in the DOIC specification [I-D.ietf-dime-ovli].

4. The agent relays the answer message unchanged.

   The presence of the OC-S-F header in the answer message indicates to the TC that it needs to be prepared for overload reports in subsequent requests of the same type.

   With the loss algorithm defined in [I-D.ietf-dime-ovli] there is no explicit action required of the TC. Stateful abatement algorithms will likely require action to be taken by the TC to be able to handle subsequent overload reports.

5.1.2. Overload Report Handling

This section addresses overload report handling in a deployment with a single agent as illustrated in Figure 1.

The following three scenarios are illustrated:

- o Figure 4 shows a message flow illustrating handling of host reports for host-routed requests and realm-routed requests when there is a single TS.

- o Figure 5 shows the handling of realm-routed requests when there is a second TS to which requests can be diverted when one of the servers is in an overload state.

- o Figure 6 illustrates the agents behavior when it has received a host report from all servers. In this case the agent generates a Realm report. This is only possible when the agent knows the overload state of all TSs for the given realm and application.

In these message flows, "xxR" indicates a Diameter request, and "xxA" indicates an answer. "HR" under a request indicates that the request is host-routed. "RR" indicates the request is realm-routed. If neither is present for a request, then it can be either host or realm routed. "OLR:Host" indicates an overload report of type Host. "OLR:Realm" indicates an overload report of type Realm.

```
     +-+                    +-+                    +-+
     |C|                    |A|                    |S|
     +-+                    +-+                    +-+
      |                      |                      |
   1> |-- xxR OC-S-F:C----->|                      |
      |                      |                      |
   2> |                      |-- xxR OC-S-F:C----->|
      |                      |                      |
   3> |                      |<----- xxA OC-S-F:S--|
      |                      |           OLR:Host  |
   4> |<--- xxA OC-S-F:S----|                      |
      |         OLR:Host     |                      |
      |                      |                      |
   5> |-- xxR OC-S-F:C---X   |                      |
      |       HR             |                      |
      |                      |                      |
   6> |-- xxR OC-S-F:C----->|                      |
      |       HR             |                      |
      |                      |                      |
   7> |                      |-- xxR OC-S-F:C----->|
```

```
        |                     |                     |
        |                     |                     |
     8> |                     |<----- xxA OC-S-F:S--|
        |                     |           OLR:Host  |
        |                     |                     |
     9> |<--- xxA OC-S-F:S----|                     |
        |         OLR:Host    |                     |
        |                     |                     |
     10>|-- xxR OC-S-F:C----->|                     |
        |       RR            |                     |
     11>|                     |-- xxR OC-S-F:C---X  |
        |                     |       RR            |
        |                     |                     |
     12>|<--- xxA Throttled---|                     |
        |                     |                     |
        |                     |                     |
     13>|-- xxR OC-S-F:C----->|                     |
        |       RR            |                     |
        |                     |                     |
     14>|                     |-- xxR OC-S-F:C----->|
        |                     |       RR            |
        |                     |                     |
     15>|                     |<----- xxA OC-S-F:S--|
        |                     |           OLR:Host  |
        |                     |                     |
     16>|<--- xxA OC-S-F:S----|                     |
        |         OLR:Host    |                     |
        |                     |                     |
```

Figure 4

1. Same as in Figure 3.

2. Same as in Figure 3.

3. S, acting as a reporting node, has determined that it needs to request a reduction in traffic. S includes the OC-S-F AVP per [I-D.ietf-dime-ovli], and selects the loss algorithm for the included report. S also includes the OC-OLR AVP to indicate the requested reduction in traffic.

4. A saves the overload state of S based on the OC-OLR AVP. A will use this overload state for handling of future realm routed requests.

   This behavior is not yet specified in the DOIC specification. It is based on the principle that only nodes with a direct

transport connection to an overloaded host should throttle those requests as other nodes earlier in the requests path do not have the topology knowledge to know if diversion of the request would have been successful.

A relays the answer message without change to OC-S-F or OC-OLR. Upon receipt of the answer, C saves overload state based on the overload report.

5. C invokes the "loss" algorithm on host-routed requests. This step illustrates a host-routed request that is rejected locally by C due to throttling. C gives application appropriate feedback to the client application.

6. This step represents a Host-routed requests that survived abatement. Such requests are handled the same as if there where no overload report for the host to which the request is routed.

7. A relays the request based on the included Destination-Host AVP.

8. A generates an answer which includes the OC-S-F AVP and OC-OLR AVP.

9. If the OC-OLR is new, then A updates the overload state associated with the report. A relays the answer without change to OC-S-F or OC-OLR.

   C determines if the OC-OLR is new. If so, C updates its locally stored overload state for S.

10. C originates a realm-routed request. C does not apply abatement to this request since it does not match any locally stored overload state (in this scenario, a realm overload report has not yet been sent.)

11. A determines that there is overload state associated with this request (the host report received from S). A uses this overload state as input to routing decisions for the request. In this case it is assumed that there is no alternative route to divert request toward and, as such, A applies throttling, and rejects the request.

12. A generates an error response indicating that the request was throttled and should not be retried.

13. C originates another realm-routed request.

14. A determines that there is overload state associated with this request (the host report received from S). A uses this overload state as input to routing decisions for the request. This request survives abatement and is routed to S.

15. S generates an answer message.

16. A relays the answer.

The following shows the case of host overload report handling of realm-routed requests when there is a second TS to which requests can be diverted when one of the servers is in an overload state.

```
    +-+                   +-+                  +--+                  +--+
    |C|                   |A|                  |S1|                  |S2|
    +-+                   +-+                  +--+                  +--+
     |                     |                     |                     |
  1> |-- xxR OC-S-F:C----->|                     |                     |
     |                     |                     |                     |
  2> |                     |-- xxR OC-S-F:C----->|                     |
     |                     |                     |                     |
  3> |                     |<----- xxA OC-S-F:S--|                     |
     |                     |           OLR:Host  |                     |
  4> |<--- xxA OC-S-F:S----|                     |                     |
     |         OLR:Host    |                     |                     |
     |                     |                     |                     |
  5> |-- xxR OC-S-F:C----->|                     |                     |
     |       RR            |                     |                     |
  6> |                     |-- xxR OC-S-F:C--------------------------->|
     |                     |       RR            |                     |
     |                     |                     |                     |
  7> |                     |<--------------------------- xxA OC-S-F:S--|
     |                     |                     |                     |
  8> |<--- xxA OC-S-F:S----|                     |                     |
     |                     |                     |                     |
```

Figure 5

1. Same as in Figure 3.

2. Same as in Figure 3.

3. Same as in Figure 4.

4. Same as in Figure 4.

5. C originates a realm-routed request. C does not apply overload abatement to this request as it does not match any locally stored overload state (the assumption for this scenario is that a realm overload report has not yet been sent.)

6. A determines that there is overload state associated with this request (the host report received from S1). A uses this overload state as input to routing decisions for the request. In this case, it is assumed that the request would have been normally routed to S1 but is instead routed to S2 as a result of the overload report.

7. S2 generates an answer message.

8. A relays the answer.

The following illustrates the agents behavior when it has received a host report from all servers. In this case the agent generates a Realm report. This is only possible when the agent knows the overload state of all TSs for the given realm and application.

```
    +-+                   +-+                  +--+                  +--+
    |C|                   |A|                  |S1|                  |S2|
    +-+                   +-+                  +--+                  +--+
     |                     |                     |                     |
  1> |-- xxR OC-S-F:C----->|                     |                     |
     |                     |                     |                     |
  2> |                     |-- xxR OC-S-F:C----->|                     |
     |                     |                     |                     |
     |                     |                     |                     |
  3> |                     |<----- xxA OC-S-F:S--|                     |
     |                     |           OLR:Host  |                     |
  4> |<--- xxA OC-S-F:S----|                     |                     |
     |         OLR:Host    |                     |                     |
     |                     |                     |                     |
  5> |-- xxR OC-S-F:C----->|                     |                     |
     |       RR            |                     |                     |
     |                     |                     |                     |
  6> |                     |-- xxR OC-S-F:C---------------------------->|
     |                     |                     |                     |
  7> |                     |<--------------------------- xxA OC-S-F:S--|
     |                     |                     |           OLR:Host  |
  8> |<--- xxA OC-S-F:S----|                     |                     |
     |         OLR:Host    |                     |                     |
     |         OLR:Realm   |                     |                     |
     |                     |                     |                     |
```

Figure 6

1. Same as in Figure 4.

2. Same as in Figure 4.

3. Same as in Figure 4.

4. Same as in Figure 4.

5. Same as in Figure 4.

6. Same as in Figure 4.

7. Same as in Figure 4, with the addition that S2 also includes an overload report in the answer message.

8. A determines that the available capacity of all servers in the realm has been reduced to the degree that it must generate a realm report. A adds this report to the answer message, in

addition to the existing host report from S2 . C saves overload state associated with the new Realm overload report. For the duration of the realm report the client performs the requested abatement on realm-routed requests.

5.1.3. DOIC Specification Impacts

The following is a list of behavior that needs to be reflected in the DOIC specification.

- o There can be multiple abating nodes for a single overload report. In this use case, A TC handles abatement of host-routed requests. An agent with a direct transport connection to an overloaded node handles abatement of realm-routed requests that would normally be routed to that node.

- o

    Note: It is also possible that an agent will handle abatement of host-routed requests, as illustrated in Section 5.3.1.

- o Syntax for the OC-OLR AVP must support multiple OC-OLR AVPs in answer messages.

- o The working group must define a Diameter-Throttled error response that indicates that the request was rejected due to overload and that the request should not be retried.

5.2. Mixed Capabilities

This use case explores the impact of having a different set of DOIC capabilities supported by the TC and one or more agents in the path of the request.

5.2.1. Capability Announcement

Figure 7 illustrates the case. In this figure, "OC-S-F:C" indicates it carries the set of capabilities supported by C. "OC-S-FC:AC" indicates the set of capabilities that A declares to S. "OC-S-FC:S" indicates S's response to the AC set of capabilities. OC-S-FC:AS indicates A's modification to the capabilities selected by S. This is needed in the case where S's capabilities are not compatible with C's.

"OC-S-FC:AC" could indicate the merged capabilities of C and A, based on local policies at A. For example, A could indicate a union or intersection of the its local capabilities with those of C. Alternately, it A could declare an entirely different set of

capabilities towards S. If the capabilites selected between A and S differ from those selected between C and A, A becomes responsible for mapping any overload information it receives from S to fit the capabilities it negotiated with C.

```
        +-+                        +-+                        +-+
        |C|                        |A|                        |S|
        +-+                        +-+                        +-+
         |                          |                          |
   1>    |-- xxR OC-S-F:C---------->|                          |
         |                          |                          |
   2>    |                          |-- xxR OC-S-F:AC--------->|
         |                          |                          |
   3>    |                          |<---------- xxA OC-S-F:S--|
         |                          |                          |
   4>    |<-------- xxA OC-S-F:AS---|                          |
         |                          |                          |
```

Figure 7

1. C originates a request including OC-S-F:C, indicating the DOIC features supported by C.

2. A inspects OC-S-F:C and determines that A supports features not included. A relays the request, replacing OC-S-F:C APV with an OC-S-F:AC.

3. S responds to the set of advertised features with the OC-S-F:S AVP. There is no change in S's behavior beyond what is specified in [I-D.ietf-dime-ovli] and any other extensions documenting the features in the received OC-S-F AVP.

4. A inspects OC-S-F:S. If necessary A replaces OC-S-F:S with OC-S-F:AS'.

   Section 5.2.2 illustrates one example were A needs to OC-S-F:S with OC-S-F:AS'.

5.2.2. Mixed Abatement Algorithms

Figure 8 illustrates one specific type of mixed capabilities. In this case, C only supports the loss abatement algorithm, A supports both loss and rate and S selects rate.

```
              +-+                        +-+                        +-+
              |C|                        |A|                        |S|
              +-+                        +-+                        +-+
               |                          |                          |
         1>    |-- xxR OC-S-F:C---------->|                          |
               |       loss               |                          |
               |                          |                          |
         2>    |                          |-- xxR OC-S-F:AC--------->|
               |                          |       loss, rate         |
               |                          |                          |
         3>    |                          |<---------- xxA OC-S-F:S--|
               |                          |                rate      |
               |                          |             OC-OLR:S     |
               |                          |                rate      |
               |                          |                          |
         4>    |<-------- xxA OC-S-F:AS---|                          |
               |              loss        |                          |
               |           OC-OLR:A       |                          |
               |              loss        |                          |
```

Figure 8

1. C originates a request with OC-S-F:C, indicating support for only the "loss" algorithm.

2. A inspects OC-S-F:C and determines it needs to advertise support for additional capabilities. A removes OC-S-F:C and inserts OC-S-F:AC, indicating support for both the "loss" and "rate" algorithms. A stores the state from OC-S-F:C to be referenced when it receives the associated answer.

3. S responds with OC-S-F:S, indicating that the rate algorithm will be used for overload reports, and OC-OLR:rate, indicating an overload condition with a specific requested rate-limit.

4. A recalls that C did not indicate support the rate algorithm and replaces OC-S-F:S with OC-S-F:AS, which indicates that the loss abatement algorithm will be used for overload reports sent to C. A must enforce the rate limit locally, so it removes OC-OLR:rate. If C's offered load exceeds what A can handle without violating the requested rate-limit, it inserts OC-OLR:A, requesting a traffic reduction using the "loss" algorithm.

This flow assumes that A is able to handle rate-based overload reports, even though the TC cannot. How this is done in practice is implementation specific. In this example, A applies local rate-

limiting, but send loss-based OLRs to C if A cannot handle C's offered load without violating the rate-limit.

If A chose to apply local throttling to enforce the rate limit, instead of sending load-based OLRs back to the TC, the scenario would be closer to that for a TC that did not support DOIC (Section 5.3.1).

5.2.3. DOIC Specification Impacts

An agent must have the ability to replace the OC-S-F AVP in request messages.

An agent must have the ability to remove or replace the OC-S-F AVP in answer messages.

An agent must have the ability to remove or replace the OC-OLR AVP in answer messages.

5.3. Non-Supporting Nodes

This section outlines the impact of agent based scenarios where there is a node that does not support DOIC in the path of a request. There are five variations of this use case:

1. Non-supporting TC.

2. Non-supporting TS.

3. Non-supporting agent between the TC and a DOIC agent.

4. Non-supporting agent between a DOIC agent and the TS.

5. Non-supporting agent between DOIC agents.

5.3.1. Non-Supporting Transaction Client

This section outlines the handling of non-supporting transaction client.

This use case is illustrated in Figure 9. In this case assume that C1 supports DOIC and C2 does not.

```
             +--+                +--+
             |C1|-----      -----|S1|
             +--+     \+--+/     +--+
                       |A1|
             ****     /+--+\     +--+
             *C2*-----      -----|S2|
             ****                +--+
```

Figure 9

Figure 10 illustrates capability announcement for both the supporting and non-supporting client. This scenario assumes that the capabilities supported by C1 and A are the same.

There is no change from the simple agent use case for transactions originated by C1.

For transactions originated by the non-supporting reacting node C2, A1 determines that C2 does not support DOIC by the absence of an OC-S-F AVP and inserts an OC-S-F AVP indicating the OC features supported by A1.

```
    +--+                  ****                  +-+                   +-+
    |C1|                  *C2*                  |A|                   |S|
    +--+                  ****                  +-+                   +-+
     |                     |                     |                     |
  1> |-- xxR OC-S-F:C--------------------------->|-- xxR OC-S-F:C----->|
     |                     |                     |                     |
  2> |<---------------------------xxA OC-S-F:S---|<-----xxA OC-S-F:S---|
     |                     |                     |                     |
  3> |                     |-- xxR ------------->|                     |
     |                     |                     |                     |
  4> |                     |                     |-- xxR OC-S-F:A----->|
     |                     |                     |                     |
  5> |                     |                     |<-----xxA OC-S-F:S---|
     |                     |                     |                     |
  6> |                     |<------------- xxA---|                     |
     |                     |                     |                     |
```

Figure 10

1. C1 supports DOIC and, as such, includes the OC-S-F AVP in all request messages sent. A relays the request to S based on normal request handling.

2. S supports DOIC and, as such, includes the OC-S-F AVP in all response messages sent. A relays the answer to C1 based on normal answer handling.

3. C2 does not support DOIC and, as such, does not insert the OC-S-F AVP into request messages.

4. A recognizes that C2 does not support DOIC, since the request does not contain the OC-S-F AVP. A inserts an OC-S-F AVP that reflects the OC capabilities of A.

5. S does normal DOIC capability announcement handling, inserting the OC-S-F AVP in the answer.

6. A removes the OC-S-F AVP from the answer given that C2 does not support DOIC.

Figure 11 illustrates overload report handling for this scenario.

There is no change in overload handling for requests originated by C1. C1 is responsible for abatement of host routed requests and A is responsible for abatement of realm-routed requests.

For requests originated by C2, it becomes the responsibility of A to handle overload abatement requested by S. In this case A is responsible for abatement of both host-routed and realm-routed requests, as A has a direct transport connection to S. The way an agent handles this in practice is implementation specific. Depending on the algorithm, an agent might be able to treat requests from all non-supporting clients as a pool. More complex implementations might maintain (and certain algorithms might require) the agent to maintain an overload control state machine for each known non-supporting client.

> If there were an upstream DOIC agent between A and S then A would no longer have a direct transport connection and would not be able to do abatement of realm-routed requests. It would become the responsibility of the upstream DOIC agent with the transport connection to handle abatement of realm-routed requests.

```
    +--+                  ****                  +-+                   +-+
    |C1|                  *C2*                  |A|                   |S|
    +--+                  ****                  +-+                   +-+
     |                     |                     |                     |
  1> |-- xxR OC-S-F:C--------------------------->|-- xxR OC-S-F:C----->|
     |                     |                     |                     |
  2> |<---------------------------xxA OC-S-F:S---|<-----xxA OC-S-F:S---|
     |                     |          OLR        |          OLR        |
     |                     |                     |                     |
     |HR abatement handled by C1                 |RR abatement handled by A
     |                     |                     |                     |
  3> |                     |-- xxR ------------->|-- xxR OC-S-F:A----->|
     |                     |                     |                     |
  4> |                     |<------------- xxA---|<-----xxA OC-S-F:S---|
     |                     |                     |          OLR        |
     |                     |                     |                     |
     |                     |                     | HR and RR abatement |
     |                     |                     | handled by A        |
     |                     |                     |                     |
  5> |                     |-- xxR ------------->|-- xxR OC-S-F:A----->|
     |                     |                     |                     |
  6> |                     |<------------- xxA---|<-----xxA OC-S-F:S---|
     |                     |                     |          OLR        |
     |                     |                     |                     |
  7> |                     |-- xxR ------------->|                     |
     |                     |                     |                     |
  8> |                     |<------------- xxA---|                     |
     |                     |               Diameter-Throttled          |
     |                     |                     |                     |
```

Figure 11

1. Request from C1, a supporting TC.

2. Response indicating S is requesting a reduction in traffic sent due to an overload condition. C1 becomes responsible for abatement of host-routed requests and A becomes responsible for abatement of realm-routed requests.

3. Request from C2, a non-supporting TC. A inserts an OC-S-F AVP.

4. Response indicating that S is overloaded. A stores overload state based on the content of the overload report. A also removed the OC-S-F and OC-OLR AVPs from the answer message. A becomes responsible for abatement handling of all requests originated by C2.

5. Request from non-supporting node originated after the overload report is received. This request survives abatement by A.

6. Response for message that survived abatement by A.

7. Request from non-supporting node originated after the overload report is received. This request does not survive abatement and is rejected by A.

8. Response for request that did not survive abatement by A, with an appropriate error code to indicate the request was throttled and should not be retried.

5.3.2. Non-supporting Transaction Server

This section shows the case where there is a mix of transaction servers that support DOIC and those that do not support DOIC.

In this case, it becomes the responsibility of a DOIC agent to become the reporting node for the non-supporting transaction server. The method the agent uses to determine if abatement of traffic is required for the non-supporting node is implementation specific. (For example, an agent may infer that a TS is overloaded by observing Diameter or transport errors, or it may have some proprietary, out-of-band mechanism for learning about TS overload.)

```
+--+                +--+
|C1|-----      -----|S1|
+--+     \+--+/     +--+
          |A1|
+--+     /+--+\     ****
|C2|-----      -----*S2*
+--+                ****
```

Figure 12

```
       +-+                   ---                  +--+                  ****
       |C|                   |A|                  |S1|                  *S2*
       +-+                   ---                  +--+                  ****
        |                     |                     |                     |
     1> |-- xxR OC-S-F:C----->|-- xxR OC-S-F:C----->|                     |
        |                     |                     |                     |
     2> |<-----xxA OC-S-F:S---|<-----xxA OC-S-F:S---|                     |
        |                     |          OLR        |                     |
        |                     |                     |                     |
     3> |-- xxR OC-S-F:C----->|                     |                     |
        |                     |                     |                     |
     4> |                     |-- xxR OC-S-F:C--------------------------->|
        |                     |                     |                     |
     5> |                     |<-----------------------------------xxA ---|
        |                     |                     |                     |
     6> |<-----xxA OC-S-F:A---|                     |                     |
        |          OLR        |                     |                     |
        |                     |                     |                     |
```

Figure 13

1. Normal DOIC processing resulting in the request being routed to S1.

2. Normal DOIC processing.

3. Normal DOIC processing

4. Normal DOIC processing resulting in the request being routed to S2. The agent doesn't know yet that S2 doesn't support DOIC.

5. S2 does not support DOIC and, as a result, does not insert the OC-S-F AVP in the answer message.

6. A takes on responsibility for becoming the reporting node for S2, and inserts an OC-S-F AVP. In this case A has determined that S2 is in an overload condition and inserts an OC-OLR AVP in the answer message.

   C handles the OC-OLR overload report in the same way it handles all OC-OLR reports.

5.3.3. Non-Supporting Agent

There are two sub-cases for non-supporting agents.

Figure 14 illustrates the first non-supporting agent case, where the first agent in a chain of agents does not support DOIC.

In this case, A2 picks up the responsibility of handling overload abatement in the case that either C1 or C2 do not support DOIC.

A2 is also responsible for abating realm-routed requests for host reports received from S1 or S2.

```
          +--+                          +--+
          |C1|-----                -----|S1|
          +--+     \****      +--+/     +--+
                    *A1*------|A2|
          +--+     /****      +--+\     +--+
          |C2|-----                -----|S2|
          +--+                          +--+
```

Figure 14

Figure 15 illustrates the second non-supporting agent case, where the last agent in the chain does not support DOIC.

In this scenario, there is no DOIC node that has a direct transport connection with S1 and S2. As a result, there is no DOIC node that can correctly handle abatement of realm-routed requests resulting. In the example, A1 cannot perform diversion, because it cannot control whether any given request goes to S1 or S2. And it cannot correctly determine how much to throttle unless it has advance knowledge of the topology behind A2, which is currently out of scope for DOIC.

As a result, this deployment scenario should be avoided.

```
+--+                         +--+
|C1|-----               -----|S1|
+--+     \+--+      ****/     +--+
          |A1|------*A2*
+--+     /+--+      ****\     +--+
|C2|-----               -----|S2|
+--+                         +--+
```

Figure 15

5.3.4. DOIC Specification Impacts

- o Agents must be allowed to insert OC-S-F AVPs into request and answer messages.

- o Agents must be allowed to remove OC-S-F AVPs from request and answer messages.

- o Agents must be able to insert OC-OLR AVPs of type "Host Report" into answer messages. (The ability to insert OC-ORL AVPs of type "Realm Report" is already assumed.)

- o Agents must be allowed to remove OC-OLR AVPs from answer messages.

- o Agents must be allowed to abate host-routed requests.

5.4. Inter Domain Authentication

Figure 16 shows three administrative domains, Dom 1, Dom 2, and Dom 3. Dom 3 does not wish information about the condition of it's network to be shared with Dom 1, but is willing to share overload information with Dom 2 in order to optimize inter-domain traffic.

```
       +--------------------+  +-----------+  +--------------------+
       |  Dom 1             |  | Dom 2     |  | Dom 3              |
       |     +--+           |  |           |  |            +--+    |
       |     |C1|-----      |  |   +---+   |  |       -----|S1|    |
       |     +--+     \+--+--------|A2A|---------+--+/     +--+    |
       |               |A1| |  |   +---+   |  |  |A3|              |
       |     +--+     /|  | |  |   *****   |  |  |  |\     +--+    |
       |     |C2|----- +--+--------*A2B*---------+--+ -----|S2|    |
       |     +--+           |  |   *****   |  |            +--+    |
       |                    |  |           |  |                    |
       +--------------------+  +-----------+  +--------------------+
```

Figure 16

Dom 2 in in the process of incremental deployment of DOIC. Agent A2A supports DOIC, but A2B does not. A1 and A3 are configured so that Diameter requests between them may traverse either A2A or A2B.

Figure 17 shows a Diameter message exchange for each potential route. The originating TC and selected TS are not relevant for the example, so they are omitted from the diagram.

```
        +--+                 +---+                 *****                  +--+
        |A1|                 |A2A|                 *A2B*                  |A3|
        +--+                 +---+                 *****                  +--+
         |                     |                     |                     |
      1> |-- xxR OC-S-F:A1---->|------xxR OC-S-F:A1----------------------->|
         |                     |                     |                     |
      2> |<---------------xxA--|<---------------------------xxA OC-S-F:A3--|
         |                     |                     |                     |
      3> |---xxR OC-S-F:A1-------------------------->|-xxR OC-S-F:A1------>|
         |                     |                     |                     |
      4> |<---------------------------xxA OC-S-F:A3--|<-----xxA OC-S-F:A3--|
         |                     |                     |                     |
```

Figure 17

1. A1 sends a request that contains OC-S-F:A1 to A2A. A2A supports the same DOIC capabilities, so it forwards the request with OC-S-F:A1 unchanged.

2. A3 receives the answer from the TC. It forwards the response to A2A, including it's capabilities in OC-S-F:A3. A2A is configured to enforce Dom 3's wishes that it's overload information not be

sent to Dom 1, so it strips the AVP before forwarding the response back to A1.

3. A1 again sends a request including OC-S-F:A1, but this one traverses A2B. Since A2B does not support DOIC at all, it treats the AVP as an unknown AVP and forwards it unchanged to A3. Note that the OC-S-F AVP observed by A3 is identical to that from step 1.

4. A3 receives the answer from the TC. It mistakenly believes A2B supports DOIC, and therefore forwards the response with it's DOIC capabilities in OC-S-F:A3. Since A2B does not recognize the AVP, it forwards it back to A1 without change.

This is a somewhat contrived example, but it shows a case where Dom 3 leaked information to an untrusted domain, because it could not tell the difference between an OC-S-F AVP received from a trusted peer that supports DOIC, or one forwarded from downstream by a non-supporting peer.

5.4.1. DOIC Specification Impacts

A DOIC supporting node must be able to distinguish between an OC-S-F AVP sent by a peer that supports DOIC, and one sent by a non-adjacent node and forwarded by a non-supporting peer. That is not possible in DOIC at the time of this writing.

> The ability to limit overload information to nodes that are authorized to receive it may require the ability to fully attribute a given OC-S-F AVP to the node that included the AVP. Whether this is required, and how an AVP that is forwarded by a DOIC supporting relay is for further study.

6. Recommendations

This section summarizes the recommendations made in previous sections. These recommendations are presented without normative language, but the authors expect that some of the recommendations will require new normative language in [I-D.ietf-dime-ovli]. Others may result in non-normative guidance.

As noted earlier, nothing in this draft should imply that relays are required to deploy DOIC. The majority of these recommendations should be interpreted to allow certain relay behaviors, but not to require them, and to offer architectural guidance on how an operator can best utilize relays in a DOIC deployment if they choose to do so.

6.1. General Recommendations

This section describes recommendations that apply to the DOIC mechanism in general:

The working group should define a "Diameter-Throttled" error code, that indicates a request has failed due to overload, and should not be retried.[Section 5.1.3] TCs need to recognize the Diameter-Throttled error code, and interpret it as a final failure for the transaction.

The OC-OLR AVP syntax must allow multiple occurrences in the same Diameter answer message. [Section 5.1.3]

6.2. Agent Behavior Recommendations

The discussion in Section 4, Section 3, and Section 5 suggest certain recommendations for DOIC supporting Diameter relay behavior. The authors recommend that language be added to [I-D.ietf-dime-ovli] to the general effect of the following sections:

6.2.1. Capabilites Exchange Behaviors

This section describes recommended Agent behaviors with respect to the OC-Supported-Features AVP.

A DOIC supporting agent may act as a reporting-node, a reacting-node, or both.

An agent may act as a reporting node on behalf of a non-supporting TS, an abating node on behalf of a non-supporting TC.[Section 5.3]

An agent that acts as a reacting node must include an OC-Supported-Features in each Diameter request that it forwards in that role. If the inbound request included an OC-Supported-Features AVP, the relay may copy its content to the one in the outbound request, or may replace the contents if it wishes to indicate different DOIC capabilities to upstream nodes. If an inbound request does not contain an OC-Supported-Features AVP, the agent must insert one into the outbound request, indicating the DOIC capabilities of the agent itself.

An agent that acts as a reporting node must include an OC-Supported-Features AVP in each Diameter answer that it forwards in that role. If the agent modified the OC-Supported-Features AVP in the associated request, it must perform a reciprocal modification of the OC-Supported-Features AVP in the response.

An agent that does not support the DOIC mechanism is likely to forward an OC-Supported-Features AVP without modification. A DOIC node must be able to tell between an OC-Supported-Features AVP that was forwarded by such a non-supporting agent, and one inserted or copied by a DOIC-supporting node.[Section 5.4]

6.2.2. Overload Report Behaviors

When a DOIC-supporting relay inserts an OC-Supported-Features AVP (or passes through one received from downstream), it becomes responsible for ensuring that any OLRs it receives from upstream nodes are honored. It can honor an OLR by locally performing overload abatement, delegating abatement to downstream nodes, or a combination of both.

If a relay can honor the OLR by locally diverting traffic, it should do so before resorting to throttling. For example, if a relay receives a realm report from its upstream peer, and has other less-overloaded peers that are valid for the realm and application, it diverts traffic to the less overloaded peers as needed. The relay should apply any knowledge it has of the peers' relative load and capacity in determining how to divert traffic. Note that only a relay that has a direct peer relationship with the servers in question can effectively perform diversion, since a Diameter node cannot directly control how upstream relays will route requests.[Section 5.1]

When an overload condition requires throttling of traffic, an agent should delegate that throttling to downstream nodes if at all possible. Depending on local policy and the nature of the overload condition, this means the agent either originates a new OLR to send downstream, or forwards an OLR received from upstream. For example, if a relay receives a host report (which usually requires traffic throttling), the relay typically forwards that report downstream. The relay may modify the report based on local policy.

If an agent needs to perform local throttling, it must explicitly reject each throttled request with a "Diameter-Throttled" error code.[Section 5.1]

> There may be circumstances where an agent must perform local throttling. An obvious example is when downstream nodes do not support DOIC, that is, requests from downstream nodes do not contain OC-Supported-Features AVPs. Mismatched upstream and downstream capabilities may require local throttling. For example, if a relay uses a rate-limiting abatement algorithm upstream, but downstream devices do not support rate-limiting, it may have to locally throttle traffic to meet its upstream

abatement commitment. It might still invoke the "loss" algorithm downstream in order to reduce the amount of traffic that must be locally throttled.[Section 5.2, Section 5.3]

A relay should apply all the information at hand to determine upstream overload. For example, if a relay receives a host-report from a directly attached TS, that relay can reasonable infer that the overload condition applies to all traffic for the realm, and perform local abatement by diverting realm-routed traffic to other servers. If there is insufficient capacity to do so, then it can generate realm-reports downstream. A relay might also have knowledge of the overload or load state of other nodes through some non-DOIC mechanism.[Section 5.1]

Finally, a relay should not generate or forward OLRs in a way likely to cause redundant abatement. For example, if a relay locally throttles traffic due to a "loss" algorithm OLR, it should not forward the OLR downstream where other nodes will also apply abatement to the same traffic. [Section 5.3]

The idea of redundant abatement is at least somewhat specific to the algorithm. For example, a rate-limiting algorithm might allow both local and delegated abatement, since the algorithm creates a maximum rate limit. On the other hand, the "loss" algorithm requests a percentage reduction. If a relay receives an OLR for a 10 percentage reduction, applies local throttling, and also forwards the OLR downstream, the 10% reduction may be applied twice.

7. Security Considerations

Several use cases in this document involve Agents inserting, removing, or modifying DOIC related AVPs. [RFC6733] does not allow "relay" agents to modify any part of a Diameter message except for routing information. This has one of two implications; either relay agents cannot take an active role in DOIC, or the DOIC AVPs should be designated as "routing AVPs". The authors recommend the second.

But regardless of whether a relay is allowed to modify DOIC AVPs, proxy agents will almost certainly need to do so. Diameter currently only offers hop-by-hop integrity protection of message contents, but the DIME working group is considering requirements for end-to-end protection [I-D.ietf-dime-e2e-sec-req] at the time of this writing. Those requirements currently recognize that different AVPs may require different security treatments. The working group should carefully consider how DOIC will interact with end-to-end security when it is completed.

Until such end-to-end protection is deployed, Diameter follows a fundamentally transitive trust model. Adjacent nodes can authenticate each other's identity, and protect exchanged messages from tampering or eavesdropping. But Diameter nodes have no way of authenticating message content received from non-adjacent nodes, other than trusting the immediate peer to do the right thing.

DOIC related information may be sensitive, and could be destructive if forged or modified by unauthorized parties. DOIC nodes must trust DOIC supporting peers to ensure that no unauthorized parties insert overload reports, and to ensure that reports are not delivered to unauthorized parties. Peers that do not support DOIC cannot be expected to enforce such policies.

At the time of this writing, DOIC provides no way for a supporting node to distinguish between a DOIC AVP from a immediate peer that supports DOIC, and one forwarded by a non-supporting peer. This issue needs to be addressed before DOIC can meet requirements 27, 28, and 29 of [RFC7068]

8. References

8.1. Normative References

[DOIC-rate]
Donovan, S. and E. Noel, "Diameter Agent Overload", February 2014.

[I-D.ietf-dime-ovli]
Korhonen, J., Donovan, S., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", draft-ietf-dime-ovli-02 (work in progress), March 2014.

[RFC4006] Hakala, H., Mattila, L., Koskinen, J-P., Stura, M., and J. Loughney, "Diameter Credit-Control Application", RFC 4006, August 2005.

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

[agent-overload]
Donovan, S., "Diameter Agent Overload", March 2014.

8.2. Informative References

[I-D.ietf-dime-e2e-sec-req]
Tschofenig, H., Korhonen, J., Zorn, G., and K. Pillay, "Diameter AVP Level Security: Scenarios and Requirements", draft-ietf-dime-e2e-sec-req-01 (work in progress), October 2013.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

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

Authors' Addresses

Steve Donovan
Oracle
Frisco, Texas
USA

Phone: +1
Email: srdonovan@usdonovans.com

Ben Campbell
Oracle
Frisco, Texas
USA

Phone: +1
Email: ben@nostrum.com