Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 23, 2017

B. Campbell
S. Donovan, Ed.
Oracle
JJ. Trottin
Nokia

March 22, 2017

Diameter Load Information Conveyance
draft-ietf-dime-load-09

Abstract

RFC7068 describes requirements for Overload Control in Diameter. This includes a requirement to allow Diameter nodes to send "load" information, even when the node is not overloaded. RFC7683 (Diameter Overload Information Conveyance (DOIC)) solution describes a mechanism meeting most of the requirements, but does not currently include the ability to send load information. This document defines a mechanism for conveying of Diameter load information.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC7068] describes requirements for Overload Control in Diameter [RFC6733]. The DIME working group has finished the Diameter Overload Information Conveyance (DOIC) mechanism [RFC7683]. As currently specified, DOIC fulfills some, but not all, of the requirements.

In particular, DOIC does not fulfill Req 23 and Req 24:

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

There are several other requirements in [RFC7068] that mention both overload and load information that are only partially fulfilled by DOIC.

The DIME working group explicitly chose not to fulfill these requirements when publishing DOIC [RFC7683] due to several reasons. A principal reason was that the working group did not agree on a general approach for conveying load information. It chose to progress the rest of DOIC, and deferred load information conveyance to a DOIC extension or a separate mechanism.

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

2. Terminology and Abbreviations

AVP

> Attribute Value Pair

DOIC

> Diameter Overload Information Conveyance ([RFC7683])

Load

> The relative usage of the Diameter message processing capacity of a Diameter node. A low load level indicates that the Diameter

node is under utilized. A high load level indicates that the node is closer to being fully utilized.

Offered Load

The actual traffic sent to the reporting node after overload abatement and routing decisions are made.

Reporting Node

Reporting Node: A Diameter node that generates a load report.

Reacting Node

Reacting Node: A Diameter node that acts upon a load report.

Routing Information

Routing Information referred to in this document can include the Routing and Peer tables defined in RFC 6733. It can also include other implementation specific tables used to store load information. This document does not define the structure of such tables.

3. Conventions Used in This Document

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

RFC 2119 [RFC2119] interpretation does not apply for the above listed words when they are not used in all-caps format.

4. Background

4.1. Differences between Load and Overload information

Previous discussions of how to solve the load-related requirements in [RFC7068] have shown that people did not have an agreed-upon concept of how "load" information differs from "overload" information. While the two concepts are highly interrelated, there are two primary differences. First, a Diameter node always has a load. At any given time that load may be effectively zero, effectively fully loaded, or somewhere in between. In contrast, overload is an exceptional condition. A node only has overload information when it is in an overloaded state. Furthermore, the relationship between a node's load level and overload state at any given time may be vague. For example, a node may normally operate at a "fully loaded" level, but

still not be considered overloaded. Another node may declare itself to be "overloaded" even though it might not be fully "loaded".

Second, Overload information, in the form of a DOIC Overload Report (OLR) [RFC7683] indicates an explicit request for action on the part of the reacting node. That is, the OLR requests that the reacting node reduces the offered load -- the actual traffic sent to the reporting node after overload abatement and routing decisions are made -- by an indicated amount (by default), or as prescribed by the selected abatement algorithm. Effectively, DOIC provides a contract between the reporting node and the reacting node.

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

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

4.2. How is Load Information Used?

[RFC7068] contemplates two primary uses for load information. Req 23 discusses how load information might be used when performing diversion as an overload abatement technique, as described in [RFC7683]. When a reacting node diverts traffic away from an overloaded node, it needs load information for the other candidates for that traffic in order to effectively load balance the diverted load between potential candidates. Otherwise, diversion has a greater potential to drive other nodes into overload.

Req 24 discusses how Diameter load information might be used when no overload condition currently exists. Diameter nodes can use the load information to make decisions to try to avoid overload conditions in the first place. Normal load-balancing falls into this category, but the diameter node can take other proactive steps as well.

If the loaded nodes are Diameter servers (or clients in the case of server-to-client transactions), both of these uses of load information should be accomplished by a Diameter node that performs server selection (selection of the Diameter endpont to which the request is to be routed for processing). Typically, server selection is performed by a node (a client or an agent) that is an immediate

peer of the server. However, there are scenarios (see Appendix A) where a client or proxy that is not the immediate peer to the selected servers performs server selection. In this case, the client or proxy enforces the server selection by inserting a Destination-Host AVP.

> For example, a Diameter node (e.g. client) can use a redirect agent to get candidate destination host addresses. The redirect agent might return several destination host addresses, from which the Diameter node selects one. The Diameter node can use load information received from these hosts to make the selection.

Just as load information can be used as part of server selection, it can also be used as input to the selection of the next-hop peer to which a request is to be routed.

It should be noted that a Diameter node will need to process both Load reports and Overload reports from the same Diameter node. The reacting node for the Overload report always has the responsibility to reduce the amount of Diameter traffic sent to the overloaded node. If, or how, the reacting node uses load information to achieve this is left as an implementation decision.

5. Solution Overview

The mechanism defined here for the conveyance of load information is similar in some ways to the mechanism defined for DOIC and is different in other ways.

As with DOIC, load information is conveyed by piggy-backing the Load AVPs on existing Diameter applications.

There are two primary differences. First, there is no capability negotiation process for load. The sender of the load information is sending it with the expectation that any supporting nodes will use it when making routing decisions. If there are no nodes that support the Load mechanism then the load information is ignored.

The second big difference between DOIC and Load is visibility of the DOIC or load information within a Diameter network. DOIC information is sent end-to-end resulting in the ability of all nodes in the path of the answer message that carries the OC-OLR AVP to act on the information, although only one node actually comsumes and reacts to the report. The DOIC overload reports remain in the message all the way from the reporting node to the node that is the target for the answer message.

For the Load mechanism there are two types of Load reports and only the first one is transmitted end-to-end.

The first type of Load report is a HOST report which contains the load of the endpoint sending the answer message. This Load report is carried end-to-end to enable any nodes that make server selection decisions to use the load status of the sending endpoint as part of the server selection decision. Unlike with DOIC, more than one node may make use of the load information received.

The second type of Load report is a PEER report. This report is used by Diameter nodes as part of the logic to select the next-hop Diameter node and, as such, does not have significance beyond the peer node. Load reports of type PEER are removed by the first supporting Diameter node to receive the report.

Because Load reports can traverse Diameter nodes that do not support the Load mechanism, it is necessary to include the identity of the node to which the Load report applies as part of the Load report. This allows for a Diameter node to verify that a Load report applies to its peer or if it should be ignored.

The Load report includes a value indicating relative load of the sending node, specified in a manner consistent with that defined for DNS SRV [RFC2782].

The goal is to make it possible to use both the load values received as a part of the Diameter Load mechanism and weight values received as a result of a DNS SRV query. As a result, the Diameter load value has a range of 0-65535. This value and DNS SRV weight values are then used in a distribution algorithm similar to that specified in [RFC2782].

The DNS SRV distribution algorithm results in more messages being sent to a node with a higher weight value. As a result, a higher Diameter load value indicates a LOWER load on the sending node. A node that is heavily loaded sends a lower Diameter load value. Stated another way, a node that has zero load would have a load value of 65535. A node that is 100% loaded would have a load value of 0.

The distribution algorithm used by Diameter nodes supporting the Diameter Load mechanism is an implementation decision but it needs to result in similar behavior to the algorithm described for the use of weight values specified in [RFC2782].

The method for calculating the load value included in the Load report is also left as an implementation decision.

The frequency for sending of Load reports is also left as an implementation decision. The sending node might choose to send Load reports in all messages or it might choose to only send Load reports when the load value has changed by some implementation specific amount. The important consideration is that all nodes needing the load information have a sufficiently accurate view of the node's load.

5.1. Theory of Operation

This section outlines how the Diameter Load mechanism is expected to work.

For this discussion, assume the following Diameter network configuration:

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

Figure 1: Example Diameter Network

Note that in this diagram, S[1], S[2] through S[p] are peers to A3. S[p+1], S[p+2] through S[n] are peers to A4.

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

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

Figure 2: Request Message Path

When sending the answer message, an endpoint node that supports the Diameter Load mechanism includes its own load information in the answer message. Because it is a Diameter endpoint it includes a HOST Load report.

```
      C     A1     A4     S[n]
      |      |      |      |
      |      |      |<-----|
      |      |       xxA(Load type:HOST, source:S[n])
      |      |      |      |
```

Figure 3: Answer Message from S[n]

If Agent A4 supports the Load mechanism then A4's actions depend on whether A4 is responsible for doing server selection. If A4 is not doing server selection then A4 ignores the HOST Load report. If A4 is responsible for doing server selection then it stores the load information for S[n] in its routing information for the handling of subsequent request messages. In both cases A4 leaves the HOST report in the message.

> Note: If A4 does not support the Load mechanism then it will relay the answer message without doing any processing on the load information. In this case the load information AVPs will be relayed without change.

A4 then calculates its own load information and inserts load information AVPs of type PEER in the message before sending the message to A1.

```
      C     A1     A4     S[n]
      |      |      |      |
      |      |<-----|      |
      |       xxA(Load type:PEER, source:A4)
      |       xxA(Load type:HOST, source:S[n])
      |      |      |      |
```

Figure 4: Answer Message from A4

If A1 supports the Load mechanism then it processes each of the Load reports it receives separately.

For the PEER Load report, A1 first determines if the source of the report indicated in the Load report matches the DiameterIdentity of the Diameter node from which the request was received. If the identities do not match then the PEER Load report is discarded. If the identities match then A1 saves the load information in its routing information for routing of subsequent request messages. In both cases A1 strips the PEER Load report from the message.

For the HOST Load report, A1's actions depend on whether A1 is responsible for doing server selection. If A1 is not doing server selection then A1 ignores the HOST Load report. If A1 is responsible for doing server selection then it stores the load information for S[n] in its routing information for the handling of subsequent request messages. In both cases A1 leaves the HOST report in the message.

A1 then calculates its own load information and inserts load information AVPs of type PEER in the message before sending the message to C:

```
      C     A1     A4     S[n]
      |      |      |      |
      |<-----|      |      |
       xxA(Load type:PEER, source:A1)
       xxA(Load type:HOST, source:S[n])
```

Figure 5: Answer Message from A1

As with A1, C processes each Load report separately.

For the PEER Load report, C follows the same procedure as A1 for determining if the Load report was received from the peer from which the report was sent. When finding it does, C stores the load information for use when making future routing decisions.

For the HOST Load report, C saves the load information only if it is responsible for doing server selection.

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

6. Load Mechanism Procedures

This section defines the normative behaviors for the Load mechanism.

6.1. Reporting Node Behavior

This section defines the procedures of Diameter reporting nodes that generate Load reports.

6.1.1. Endpoint Reporting Node Behavior

A Diameter endpoint that supports the Diameter Load mechanism MUST include a Load report of type HOST in sufficient answer messages to

ensure that all consumers of the load information receive timely updates.

The Diameter endpoint MUST include its own DiameterIdentity in the SourceID AVP included in the Load AVP.

The Diameter endpoint MUST include a Load-Type AVP of type HOST in the Load AVP.

The Diameter endpoint MUST include its load value in the Load-Value AVP in the Load AVP.

The LOAD value should be calculated in a way that reflects the available load independently of the weight of each server, in order to accurately compare LOAD values from different nodes. Any specific LOAD value needs to identify the same amount of available capacity, regardless the Diameter node that calculates the value.

The mechanism used to calculate the LOAD value that fulfills this requirement is an implementation decision.

The frequency of sending Load reports is an implementation decision.

> For instance, if the only consumer of the Load reports is the endpoint's peer then the endpoint can choose to only include a Load report when the load of the endpoint has changed by a meaningful percentage. If there are consumers of the endpoint Load report other then the endpoint's peer (this will be the case if other nodes are responsible for server selection) then the endpoint might choose to include Load reports in all answer messages as a way of ensuring that all nodes doing server selection get accurate load information.

6.1.2. Agent Reporting Node Behavior

A Diameter Agent that supports the Diameter Load mechanism MUST include a PEER Load report in sufficient answer messages to ensure that all users of the load information receive timely updates.

The Diameter Agent MUST include its own DiameterIdentity in the SourceID AVP included in the Load AVP.

The Diameter Agent MUST include a Load-Type AVP of type PEER in the Load AVP.

The Diameter Agent MUST include its load value in the Load-Value AVP in the Load AVP.

The LOAD value should be calculated in a way that reflects the available load independently of the weight of each agent, in order to accurately compare LOAD values from different nodes. Any specific LOAD value needs to identify the same amount of available capacity, regardless the Diameter node that calculates the value.

The mechanism used to calculate the LOAD value that fulfills this requirement is an implementation decision.

The frequency of sending Load reports is an implementation decision.

> Note: In the case of peer Load reports it is only necessary to include Load reports when the load value has changed by some meaningful value, as long as the agent ensures that all peers receive the report. It is also acceptable to include the Load report in every answer message handled by the Diameter Agent.

6.2. Reacting Node Behavior

This section defines the behavior of Diameter nodes processing Load reports.

A Diameter node that supports the Diameter Load mechanism MUST be prepared to process Load reports of type HOST and of type PEER, as indicated in the Load-Type AVP included in the Load AVP received in the same answer message or from multiple answer messages.

> Note that the node needs to be able to handle messages with no load reports, messages with just a PEER Load report, messages with just an HOST Load report and messages with both types of Load reports.

If the Diameter node is not responsible for doing server selection then it SHOULD ignore Load reports of type HOST.

If the Diameter node is responsible for doing server selection then it SHOULD save the load value included in the Load-Value AVP included in the Load AVP of type HOST in its routing information.

If the Diameter node receives a Load report of type PEER then the Diameter node MUST determine if the Load report was inserted into the answer message by the peer from which the message was received. This is achieved by comparing the DiameterIdentity associated with the connection from which the message was received with the DiameterIdentity included in the SourceID AVP in the Load report.

If the Diameter node determines that the Load report of type PEER was not received from the peer that sent or relayed the answer message then the node MUST ignore the Load report.

If the Diameter node determines that the Load report of type PEER was received from the peer that sent or relayed the answer message then the node SHOULD save the load information in its routing information.

In all cases, a Diameter Agent MUST strip all Load reports of type PEER received in answer messages.

> Note: This ensures that there will be precisely one Load report of type PEER, that of the Diameter node sending the message, in any answer messages sent by the Diameter Agent.

How a Diameter node uses load information for making routing decisions is an implementation decision. However, the distribution algorithm MUST result in similar behavior as the algorithm described for the use of weight values in [RFC2782].

6.3. Extensibility

The Load mechanism can be extended to include additional information in the Load reports.

Any extension may define new AVPs for use in Load reports. These new AVPs SHOULD be defined to be extensions to the Load AVPs defined in this document.

[RFC6733] defined Grouped AVP extension mechanisms apply. This allows, for example, defining a new feature that is mandatory to be understood even when piggybacked on an existing application.

As with any Diameter specification, [RFC6733] requires all new AVPs to be registered with IANA. See Section 9 for the required procedures.

6.4. Addition and Removal of Nodes

When a Diameter node is added, the new node will start by advertising its load. Downstream nodes will need to factor the new load information into load balancing decisions. The downstream nodes can attempt to ensure a smooth increase of the traffic to the new node, avoiding an immediate spike of traffic to the new node. The method for handling of such a smooth increase is implementation specific but it can rely on the evolution of load information received from the new node and from the other nodes.

When removing a node in a controlled way (e.g. for maintenance purpose, so outside a failure case), it might be appropriate to progressively reduce the traffic to this node by routing traffic to other nodes. Simple load information (load percentage) would not be sufficient. The method for handling of the node removal is implementation specific but it can rely on the evolution of the load information received from the node to be removed.

7. Attribute Value Pairs

The section defines the AVPs required for the Load mechanism.

7.1. Load AVP

The Load AVP (AVP code TBD1) is of type Grouped and is used to convey load information between Diameter nodes.

```
 Load ::= < AVP Header: TBD1 >
          [ Load-Type ]
          [ Load-Value ]
          [ SourceID ]
        * [ AVP ]
```

7.2. Load-Type AVP

The Load-Type AVP (AVP code TBD2) is of type Enumerated. It is used to convey the type of Diameter node that sent the load information. The following values are defined:

HOST 0 The Load report is for a host.

PEER 1 The Load report is for a peer.

7.3. Load-Value AVP

The Load-Value AVP (AVP code TBD3) is of type Unsigned64. It is used to convey relative load information about the sender of the Load report.

The Load-Value AVP is specified in a manner similar to the weight value in DNS SRV ([RFC2782]).

The Load-Value has a range of 0-65535.

A higher value indicates a lower load on the sending node. A lower value indicates that the sending node is heavily loaded.

Stated another way, a node that has zero load would have a load value of 65535. A node that is 100% loaded would have a load value of 0.

7.4. SourceID AVP

The SourceID AVP is defined in [I-D.ietf-dime-agent-overload]. It is used to identify the Diameter node that sent the Load report.

7.5. Attribute Value Pair flag rules

				AVP flag rules	
Attribute Name	AVP Code	Section Defined	Value Type	MUST	MUST NOT
Load	TBD1	x.1	Grouped		V
Load-Type	TBD2	x.2	Enumerated		V
Load-Value	TBD3	x.3	Unsigned64		V
SourceID	TBD4	x.4	DiameterIdentity		V

As described in the Diameter base protocol [RFC6733], the M-bit usage for a given AVP in a given command may be defined by the application.

8. Security Considerations

Load information may be sensitive information in some cases. Depending on the mechanism, an unauthorized recipient might be able to infer the topology of a Diameter network from load information. Load information might be useful in identifying targets for Denial of Service (DoS) attacks, where a node known to be already heavily loaded might be a tempting target. Load information might also be useful as feedback about the success of an ongoing DoS attack.

Given that routing decisions are impacted by load information, there is potential for negative impacts on a Diameter network caused by erroneous or malicious Load reports. This includes the malicious changing of load values by Diameter Agents.

Any load information conveyance mechanism will need to allow operators to avoid sending load information to nodes that are not

authorized to receive it. Since Diameter currently only offers authentication of nodes at the transport level and does not support end-to-end security mechanisms, any solution that sends load information to non-peer nodes requires a transitive-trust model.

9. IANA Considerations

9.1. AVP Codes

New AVPs defined by this specification are listed in Section Section 7. All AVP codes are allocated from the 'Authentication, Authorization, and Accounting (AAA) Parameters' AVP Codes registry.

9.2. New Registries

This document makes no new registry requests of IANA.

10. References

10.1. Normative References

[I-D.ietf-dime-agent-overload]
Donovan, S., "Diameter Agent Overload", draft-ietf-dime-agent-overload-02 (work in progress), August 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <http://www.rfc-editor.org/info/rfc7683>.

10.2. Informative References

[RFC7068] McMurry, E. and B. Campbell, "Diameter Overload Control Requirements", RFC 7068, DOI 10.17487/RFC7068, November 2013, <http://www.rfc-editor.org/info/rfc7068>.

Appendix A. Topology Scenarios

This section presents a number of Diameter topology scenarios, and discusses how load information might be used in each scenario.

A.1. No Agent

Figure 6 shows a simple client-server scenario, where a client picks from a set of candidate servers available for a particular realm and application. The client selects the server for a given transaction using the load information received from each server.

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

Figure 6: Basic Client Server Scenario

If a node supports dynamic discovery, it will not obtain load information from the nodes with which it has no Diameter connection established. Nevertheless it might take into account the load information from the other nodes to decide to add connections to new nodes with the dynamic discovery mechanism.

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

A.2. Single Agent

Figure 7 shows a client that sends requests to an agent. The agent selects the request destination from a set of candidate servers, using load information received from each server. The client does not need to receive load information, since it does not select between multiple agents.

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

Figure 7: Simple Agent Scenario

A.3. Multiple Agents

Figure 8 shows a client selecting between multiple agents, and each agent selecting from multiple servers. The client selects an agent based on the load information received from each agent. Each agent selects a server based on the load information received from its servers.

This scenario adds a complication that one set of servers may be more loaded than the other set. If, for example, S4 was the least loaded server, C would need to know to select agent A2 to reach S4. This might require C to receive load information from the servers as well as the agents. Alternatively, each agent might use the load of its servers as an input into calculating its own load, in effect aggregating upstream load.

Similarly, if C sends a host-routed request [RFC7683], it needs to know which agent can deliver requests to the selected server. Without some special, potentially proprietary, knowledge of the topology upstream of A1 and A2, C would select the agent based on the normal peer selection procedures for the realm and application, and perhaps consider the load information from A1 and A2. If C sends a request to A1 that contains a Destination-Host AVP with a value of S4, A1 will not be able to deliver the request.

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

Figure 8: Multiple Agents and Servers

A.4. Linked Agents

Figure 9 shows a scenario similar to that of Figure 8, except that the agents are linked, so that A1 can forward a request to A2, and vice-versa. Each agent could receive load information from the linked agent, as well as its connected servers.

This somewhat simplifies the complication from Figure 8, due to the fact that C does not necessarily need to choose a particular agent to reach a particular server. But it creates a similar question of how, for example, A1 might know that S4 was less loaded than S1 or S3. Additionally, it creates the opportunity for sub-optimal request paths. For example [C,A1,A2,S4] vs. [C,A2,S4].

A likely application for linked agents is when each agent prefers to route only to directly connected servers and only forwards requests to another agent under exceptional circumstances. For example, A1 might not forward requests to A2 unless both S1 and S3 are overloaded. In this case, A1 might use the load information from S1 and S3 to select between those, and only consider the load information from A2 (and other connected agents) if it needs to divert requests to different agents.

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

Figure 9: Linked Agents

Figure 10 is a variant of Figure 9. In this case, C1 sends all traffic through A1 and C2 sends all traffic through A2. By default, A1 will load balance traffic between S1 and S3 and A2 will load balance traffic between S2 and S4.

Now, if S1 S3 are significantly more loaded than S2 S4, A1 may route some C1 traffic to A2. This is non optimal path but allows a better load balancing between the servers. To achieve this, A1 needs to receive some load info from A2 about S2/S4 load.

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

Figure 10: Linked Agents

A.5. Shared Server Pools

Figure 11 is similar to Figure 9, except that instead of a link between agents, each agent is linked to all servers. (The links to each set of servers should be interpreted as a link to each server. The links are not shown separately due to the limitations of ASCII art.)

In this scenario, each agent can select among all of the servers, based on the load information from the servers. The client need only be concerned with the load information of the agents.

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

Figure 11: Shared Server Pools

A.6. Agent Chains

The scenario in Figure 12 is similar to that of Figure 8, except that, instead of the client possibly needing to select an agent that can route requests to the least loaded server, in this case A1 and A2 need to make similar decisions when selecting between A3 or A4. As the former scenario, this could be mitigated if A3 and A4 aggregate upstream loads into the load information they report downstream.

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

Figure 12: Agent Chains

A.7. Fully Meshed Layers

Figure 13 extends the scenario in Figure 11 by adding an extra layer of agents. But since each layer of nodes can reach any node in the next layer, each node only needs to consider the load of its next-hop peer.

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

Figure 13: Full Mesh

A.8. Partitions

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

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

A.9. Active-Standby Nodes

The previous scenarios assume that traffic can be load balanced among all peers that are eligible to handle a request. That is, the peers operate in an "active-active" configuration. In an "active-standby" configuration, traffic would be load-balanced among active peers. Requests would only be sent to peers in a "standby" state if the active peers became unavailable. For example, requests might be diverted to a stand-by peer if one or more active peers becomes overloaded.

Authors' Addresses

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

Email: ben@nostrum.com

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

Email: srdonovan@usdonovans.com

Jean-Jacques Trottin
Nokia
Route de Villejust
91620 Nozay
France

Email: jean-jacques.trottin@nokia.com