Diameter Overload Control Mechanism

draft-roach-dime-overload-ctrl-00

Adam Roach

adam@nostrum.com

MECHANISM OVERVIEW AND HIGH-LEVEL DESCRIPTION

Mechanism Overview

- Operates on a hop-by-hop basis
  - For the most part, at least. Agents have the ability to report load and overload for contexts that correspond to upstream constructs, such as hosts, realms, and applications.
- Extensible Load Shedding Algorithms
  - Initially defined with “drop” algorithm (similar to SIP overload mechanism)
- Information piggybacks on any existing Diameter request/response pair
  - Including DWR/DWA (watchdog) messages, to allow exchange of load state during on otherwise quiescent connections.
- Overload information conveyed in compound AVP
  - Contains overload capabilities negotiation, load and overload metrics (including period of validity), load shedding algorithm to be used, scope for which the information is valid.

Why Hop-by-Hop?

"The main problem of end-to-end overload control is its inherent complexity, since [clients or] servers need to monitor all potential paths to a destination in order to determine which requests should be throttled and which requests may be sent. Even if this information is available, it is not clear which path a specific request will take." (RFC 6357, section 5.2)

Why Message Piggybacking?

- Based on REQ 14: “The mechanism SHOULD provide for increased feedback when traffic levels increase. The mechanism MUST NOT do this in such a way that it increases the number of messages while at high loads.”
- Adding a new application (for example) would require an increase in message load to communicate overload.
- If we decide to avoid piggybacking, the only viable other option appears to be definition of a new application.

New AVPs

- Overload-Info-Scope – 1 or more per Load-Info
  - Indicates scope for which information is valid. (See next slide for discussion of "scope" concept)
- Supported-Scopes – Only in CER/CEA messages
  - Indicates scopes the sender can support
- Overload-Algorithm – Only in CER/CEA messages
  - The algorithm to be used by the client to shed load. By default, includes base "message loss" algorithm.
- Overload-Metric – exactly 1 per Load-Info
  - Algorithm-specific indicator of the degree of overload; used as input to the algorithm behavior.
  - Should be the first AVP in "Load-Info"
- Load – Mandatory, can appear once per Load-Info
  - Indicates an abstract "load" metric
- Period-of-Validity – Required if Overload-Metric is non-zero
  - Indicates how long the recipient should act on the overload control metric
- Session-Group – optional; may only appear in first request and/or first response in session
  - Assigns the session to a Session Group (explained later) which can be used in scope tag

Overload Scopes

In normal operation, a Diameter node may be congested for some but not all possible requests.

For example, an agent that supports more than one realm may route traffic to one set of servers for realm A, and another for realm B. If the realm A servers are congested but realm B servers are not, then the agent is effectively congested for realm A but not for realm B.

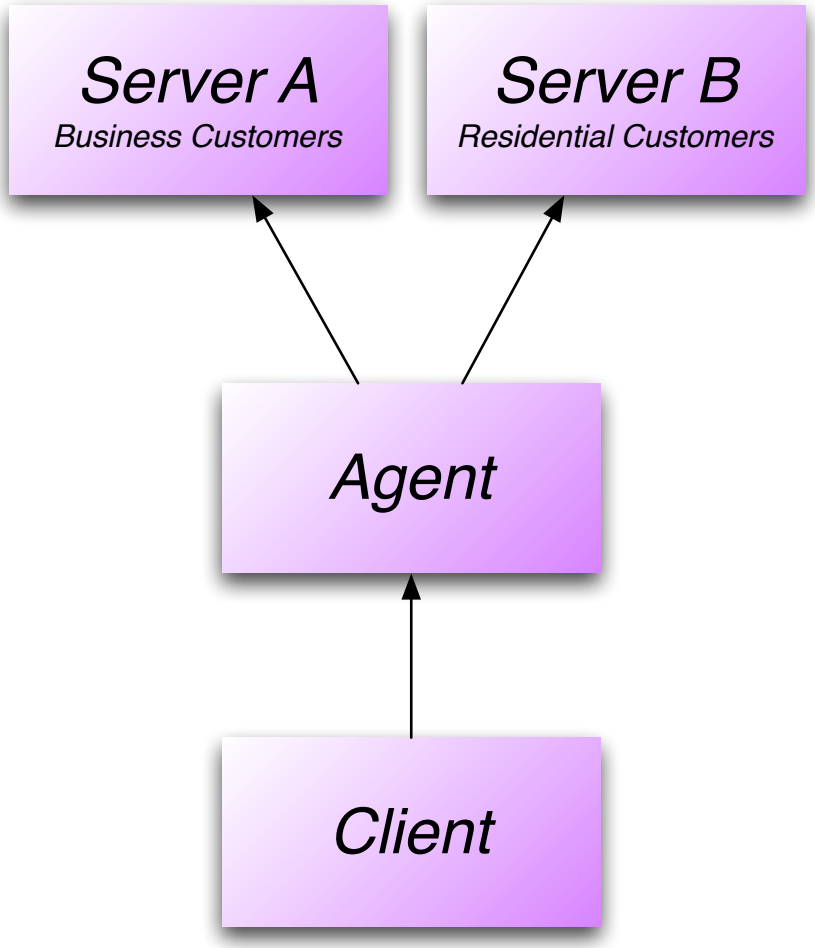
Similar situations may arise in servers that need to make use of external resources for certain applications but not for others.

The Overload-Info-Scope AVP allows a node to indicate a subset of requests to which load and overload information should be applied.

Overload-Info-Scope AVP

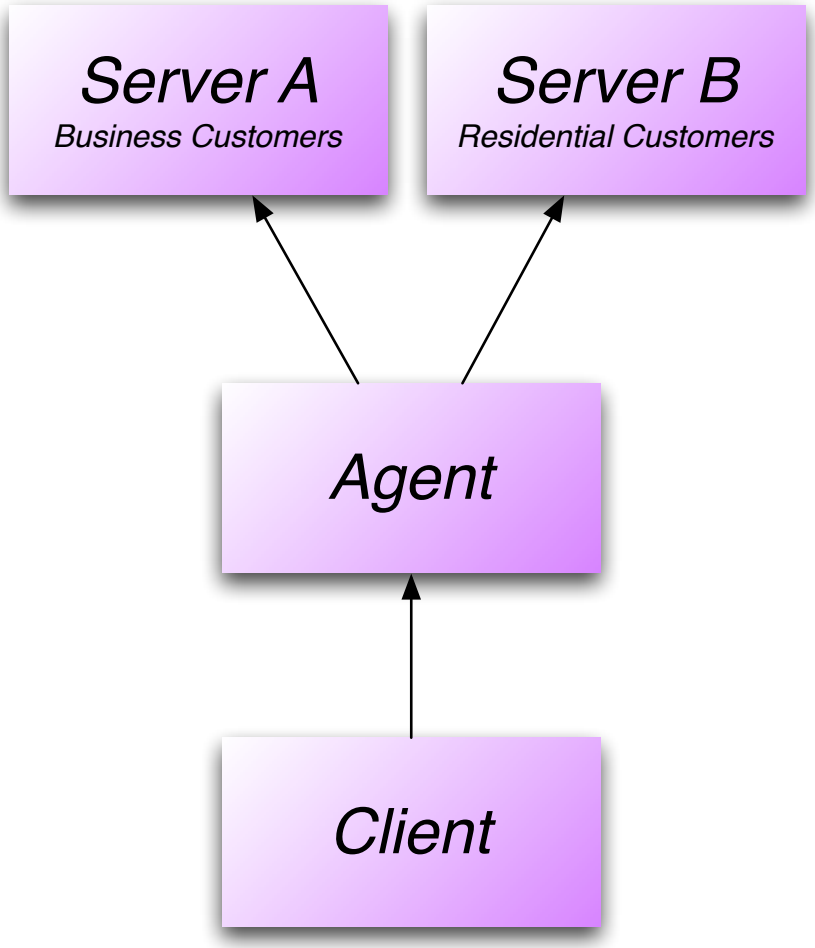
- Can specify seven scopes for overload information:
  1. Destination-Realm – Applies to the indicated realm
  2. Application – Applies to the indicated application
  3. Destination-Host – Applies to the indicated Destination-Host
  4. Host – Applies to messages sent directly to the indicated host
  5. Connection – Applies to the connection on which the indication is sent
  6. Session Group (see later slides)
  7. Session – Feedback applies to the session(s) indicated. Only relevant when topology hiding is being used.
- If a transaction falls within more than one scope, the “most congested” scope is used for traffic shaping (proposal to reword: “scope that would result in the greatest traffic reduction”)

Session Group Example (1/2)



- Consider a system in which the Diameter client is servicing a mixed community of users (e.g., business accounts and residential acounts)
- The client does not know which category users belong to.
- The Agent is able to steer requests to the appropriate Diameter server based on provisioned information.
- If Server B becomes overloaded, the Agent would ideally report congestion to the client in a way that does not impact traffic to server A.

Session Group Example (2/2)



- Whenever the Agent is involved in the establishment of a new session, it includes a "Session Group" AVP
  - For example, it can use a session group of "1" for server A, and "2" for Server B.
- Then, when Server B becomes overloaded, it can report overload to the client in a context of "Session Group 2"
  - This causes clients to change the amount of traffic they send in sessions that terminate on Server B without impacting sessions that terminate on Server A.

Supported-Scopes AVP

- Unsigned64
- Contains bitmap indicating which scopes a node can receive
  - Does not indicate what that same node might *send*
- New scopes are defined in additional extensions (IETF docs, other SDO publications)

Overload-Algorithm AVP

- Enumerated
- Sent during capabilities exchange to negotiate which (single) overload-control algorithm will be used for the duration of the connection
- Document specifies “drop,” modeled on draft-ietf-soc-overload-control.

Overload-Metric AVP

- Unsigned32
- Indicates level of overload
- Interpretation is based on actual algorithm negotiated at connection establishment

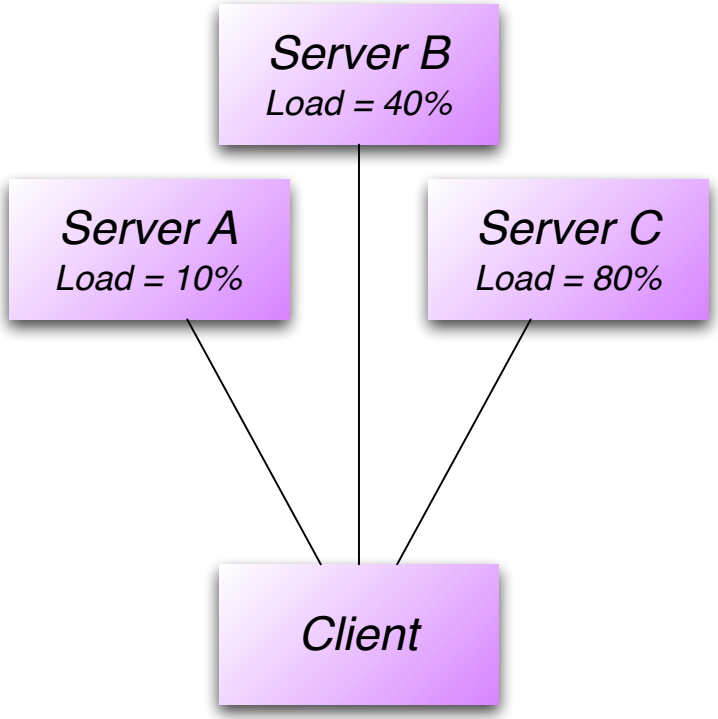
Load AVP

- Unsigned32, range 0-65535
- Based on the assumption that a system works better if it can avoid overload rather than simply react to it.
- Added to all messages
- Algorithm used to generate metric is left to implementors
  - Should generally reflect utilization of most constrained resource
  - Should be a **linear** representation of such utilization
  - Ideally, is a rolling-average load to prevent wild fluctuations

Why Load AVP in All Messages?

- Intended to reduce load on message *recipients*, by allowing periodic sampling of packets.
- The other option – sending Load AVP only when Load changes – requires recipients to examine the Load-Info AVP in all messages due to the possibility of Load being present.
- We presume that appending a non-changing AVP to a message is a cheaper operation than finding and parsing a sub-AVP, and optimized accordingly.

Example Use of Load



Server	SRV Weight
A	20
B	20
C	60

- Consider a client with three choices of a server to connect to, with DNS SRV weights as shown in the table.
- Each server has communicated load information as shown.
- The client can apply loading information to the SRV weights as follows:
  - A: 90% x 20 = 18
  - B: 60% x 20 = 12
  - C: 20% x 60 = 12
- The client then distributes load according to this 18/12/12 metric (e.g., 43% of traffic goes to A; 28.5% goes to each of B and C)
- Note that this isn't specific to DNS, and could by applied to any capacity weighting information

Period-of-Validity AVP

- Unsigned32
- Indicates number of seconds the current overload metric should be applied.
  - Unless the metric is replaced by a subsequent message.
- May be refreshed by a subsequent message to prevent expiration.
- Upon expiration, any associated overload condition is considered to have ceased.
- Does not apply to “Load” AVP
  - Loads are valid until updated.

Session-Group AVP

- UTF8String (i.e., human-readable)
- Identifies a group of sessions independent of other protocol constructs (realms, hosts, etc.)
- Assigns the session to server-selected arbitrary context that can be reported on independently from applications, realms, etc.
- Appears only in initial exchange within a session
- Cannot be changed during the course of a session.

Why Can't we Fail Non-Controlled Connections?

- The mechanism does not include any way to require the use of overload. If negotiation of the mechanism fails, the connection proceeds normally, without any overload control.
- This is done to allow for incremental deployment of the mechanism, rather than requiring a "flag day" on which all nodes switch over.
- This is also required by REQ 16.

OPEN ISSUES

Open Issue 1: SCTP Stream Scope?

- Currently define seven scopes:
  - Destination-Realm
  - Application-ID
  - Destination-Host
  - Host
  - Connection (including SCTP associations)
  - Session-Group
  - Session
- Is there value to adding a scope for SCTP streams within an association? This isn't difficult to do, but we have not yet identified use cases for it.

Open Issue 1a: Scope Optionality

- Currently, only “Connection” scope is MUST
- Several are SHOULD:
  - Destination Realm
  - Application-ID
  - Destination-Host
  - Host
- Two are MAY:
  - Session
  - Session-Group

Open Issue 2: Valid Scope Combinations?

- 1*(Destination-Realm) 0*1(Application-ID)
- 1*(Application-ID) 0*1(Destination-Realm)
- 1*(Application-ID) 0*1(Destination-Host)
- 1*(Application-ID) 0*1(Host)
- 1*(Application-ID) 0*1(Connection)
- 1*(Destination-Host)
- 1*1(Host) o 1*1(Connection)
- 1*(Session-Group) 0*1(Host | Connection)
- 1*(Session) 0*1(Host | Connection)
- **Is this the right set?**

Open Issue 3: Sequencing

- SOC includes a “sequence” parameter to allow proper handling of re-ordered messages.
- Diameter is over connection-oriented transports (SCTP or TCP), which prevents reordering within a connection/stream
- However, it is possible to have multiple connections/streams between nodes. Do we need to add sequence numbers to handle this?

Open Issue 4: ‘O’verload Bit Handling

- Currently set if the message indicates a need to throttle traffic being sent (cleared otherwise).
- Suggestions have been made to set the bit only in those messages for which a material change (e.g., enter overload, leave overload, overload metric changes by more than a certain amount), so as to reduce the amount of processing required.

Open Issue 5: Security

- There are certainly other security items to consider than those presently called out in the draft. This slide is being included mostly to call this open issue out for further scrutiny; I doubt we can have much useful conversation face-to-face.

Open Issue 6: Algorithm Registration Policy

- Current document proposes “Specification Required” (as per RFC 5226, section 4.1).
- This allows new algorithms to be defined in any “permanent and readily available public specification.”
- Is that what we want? See RFC 5226 for other possibilities.

Open Issue 7: Overload Scope Registry

- Exactly the same as the previous slide, but for overload scopes rather than overload algorithms.