Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 3, 2007

T. Asveren
Ulticom Inc.

August 30, 2006

Diameter Duplicate Detection Cons.
draft-asveren-dime-dupcons-00.txt

Status of this Memo

Copyright Notice

Abstract

Diameter transport mechanism relies on storing data about received requests to detect duplicate requests. This document discusses implementation and deployment considerations regarding this functionality.

Table of Contents

1. Introduction

Diameter Base Protocol[1] defines the transport mechanism to be used for sending/receiving requests/answers. The capability to detect duplicate requests is also included in this mechanism to prevent multiple processing of the same request. This capability relies on storing data about received requests on the server. Origin-Host AVP and End-to-End Identifiers of received messages need to be stored for duplicate detection. If the application is unable to regenerate the exact answer which was sent for the initial request, the answer message itself needs to be stored as well.

2. Reasons for Duplicate Requests

Duplicate requests may be received due to client or intermediate node restarts.

2.1. Restart of Client

When a client fails, it may retransmit requests, which were sent before the failure but for which no corresponding answer has been received yet. This may cause a server to receive the same request twice.

```
+-------+               +-------+
|       |---(1)Req----->|       |
|Client |(2)X<--Ans-----|Server |
|       |---(3)Req----->|       |
|       |<--(4)Ans------|       |
+-------+               +-------+
```

Figure 1: Retransmission of Request After Client Restart

(1) Client sends a request, request is received by server.
(2) Client goes down but before this is detected by the server, server sends back the corresponding answer.
(3) Client restarts, and resends the request with T-bit set.

2.2. Restart of Intermediate Node

When an intermediary node in the path from client to server fails, the node before it needs to retransmit requests, for which no corresponding answer has been received yet.

```
+------+          +-------+          +-------+
|Client|-(1)Req-->|Relay  |-(2)Req-->| Server|
|      |          |Agent 1|(3)X<-Ans-|       |
+------+          +-------+          +-------+
  ^   |                               ^  |
  |   |           +-------+           |  |
  |   +--(4)Req-->|Relay  |-(5)Req----+  |
  +-(7)Ans--------|Agent 2|<-----(6)Ans--+
                  +-------+
```

Figure 2: Retransmission of Request After Relay Agent Failure

(1) Client sends the request to Relay Agent 1.
(2) Relay Agent 1 forwards the request to server.
(3) Relay Agent 1 goes down and before the server can detect it, the server sends the answer.
(4) Client detects that Relay Agent 1 went down and retransmits the request to Relay Agent 2.
(5) Relay Agent 2 forwards the request to the server.
(6) Server sends the answer message to Relay Agent 2.
(7) Relay Agent 2 forwards the answer to the client.

3. Arrival of Retransmission Before Original Request

A retransmitted request may arrive to the server before the corresponding original request. This may happen due to requests taking different paths in the diameter or IP networks. Because of the latter, even a client and server, which are directly connected from diameter point of view may observe retransmitted requests arriving before the original ones, if the client restarts.

4. Duplicate Detection Implementation Guidelines

4.1. Buffering of Requests with T-bit not Set

Origin-Host AVP and End-to-End Identifier for all requests received by a server MUST be saved, until it is guaranteed that no corresponding retransmission will be received. If the server is unable to regenerate the exact answer which was sent as response to the original request, this answer message MUST be saved as well.

Diameter base protocol does not provide a mechanism, by which a server can detect that an answer message has been received by the client, which sent the corresponding request message. This would have indicated the server that buffered information for that request could be deleted because from that moment on no retransmissions for

that request are possible.

Implementations MAY configure a value for the maximum time, after which no retransmission of a request will arrive, e.g. maximum expected downtime for any client + maximum network delay. Although such a value could be only a guess and needs to be configured generously to prevent non-detection of retransmissions, it still MAY be used to decide when buffered information can be deleted.

Client failures could be hardware related, where replacement of equipment may be necessary. Such cases could result downtimes of a few hours. This would cause buffering of large amounts of data on servers. For example consider a server which handles 1000 messages per second, which can't regenerate answers:
length of End-to-End Identifier: 4 bytes
average answer length: 280 bytes
average Origin-Host AVP length: 16 bytes
maximum buffering time: 2 hours
amount of data to be buffered: 1000*7200(4+16+280) ~ 2 GBytes

Especially with larger answer messages, amount of data to be buffered can get much bigger. Usually, that type of memory requirement is considered undesirable. Implementation MAY choose to store this information in non-volatile memory but frequent writes to non-volatile memory can cause a significant performace penalty.

Applications MAY use new requests arriving from a peer as indirect acknowledgements to decrease the amount of data buffered for duplicate detection, if a value can be configured as the maximum end-to-end delay in the Diameter network. Each new request MAY be interpreted as that answers sent 2*maximum end-to-end delay ago are received by the originator of the request, and buffered data associated with the corresponding request can be deleted. This technique could decrease memory requirements for duplicate detection significantly but it should be noted that it MAY cause failure to detect duplicates, if maximum end-to-end delay is not choosen carefully.

Applications MAY try to guess end-to-end delay between two peers dynamically. This can be achieved by sending an invalid message to other peers and measuring the time difference between sending the message and receiving corresponding error answer. By considering multiple measurements and providing a generous buffer, the calculated value can be utilized while using requests as implicit acknowledgements.

4.2. Buffering of Requests with T-bit Set

Information related with requests with T-bit set MUST be buffered as well, if the original request is not received yet, because it is possible for a retransmission to arrive before the corresponding original request. In such a case, the original request MUST be treated as a duplicate.

Information buffered for requests with T-bit SHOULD be buffered as long as the expected maximum network delay. Usually this value could be around a few seconds and considering that requests with T-bit set are rare, it is not expected that memory requirements will be high.

4.3. End-to-End Id Selection

End-to-End Identifier is important from duplicate detection point of view because it uniquely identifies requests sent by a specific peer.

Diameter base protocol mandates that End-to-End Id must be unique at least for a period of 4 minutes. This MAY cause false duplicate detections, if a client goes down for more than 4 minutes, because a retransmission of a request from the previous boot-cycle and a new request MAY have the same End-to-End Id.

Considering that End-to-End Id is 32-bits, the duration of its uniqueness can be generated as a function of average number of messages per second and minimum restart time. Enough bits need to be allocated to distinguish between each message in a boot cycle and between boot cycles.

The uniqueness period t_u MUST satisfy the following inequality:
32 >= ceiling[log_2(msg_rate*t_u)] + ceiling[log_2(t_u/min_restart)]

For example:
message rate: 500 msg/sec min_restart: 1 sec
A uniqueness period of 1035 seconds could be guaranteed:
ceiling[log_2(500*1035)] + ceiling[log_2(1035)] =
19 + 11 = 30 < 32

Even if uniqueness of End-to-End Id is guaranteed for more than 4 minutes, as long as uniquenees period is less than the maximum expected downtime, false duplicate detections MAY occure but longer uniqueness periods statistically will decrease the probability of that to happen.

Implementations MAY consider Session-Id as well to decrease the possibility of false duplicate detections, in addition to End-to-End Id and Origin-Host AVP.

4.4. Retransmission of First Request in a Session

First requests in a session are different than the subsequent ones, because the first requests MAY NOT contain Destination-Host AVP. In such a case, the request is routed based on Destination-Realm AVP and Application-Id.

Considering that information about request are buffered at the server where they have been sent, retransmission of a request SHOULD be sent to the same server so that duplicate detection can be performed. To guarantee this type of behavior, all Diameter nodes SHOULD guarantee that all requests with the same End-to-End Id are sent to the same next hop.

5. IANA Considerations

This document does not require any action from IANA.

6. Security Considerations

This document does not introduce new security considerations and the considerations given in RFC3588 [1] do apply.

7. Acknowledgments

The author would like to thank David Lehmann for his invaluable comments.

8. Normative References

[1] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

Author's Address

Tolga Asveren
Ulticom Inc.
1020 Briggs Road
Mount Laurel, NJ, 08054
USA

Email: asveren@ulticom.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).