DISPATCH
Internet-Draft
Intended status: Standards Track
Expires: October 15, 2018

T. Asveren
Ribbon

April 13, 2018

HTTP Overload Control Mechanism
draft-asveren-dispatch-http-overload-control-00

Abstract

This document specifies a generic overload control mechanism for HTTP/HTTPS applications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 15, 2018.

Table of Contents

1. Overview

HTTP is used between clients and servers for request/response based interaction in the context of many different applications. Some of these applications have tight timing requirements for receiving a response. For example, a signing request done toward a Signing Server in the context of STIR RFC 8224 [RFC8224] usually is associated with a real time session setup procedure which needs to complete in a given time frame. For such applications it is imperative to consider the actual load on a server to meet the timing requirements. Current HTTP overload control relies on using 503 with a Retry-After header indicating for how long no request should be sent to a server. This, although sufficient for certain applications, does not always provide satisfactory results as it allows only binary control of the load following a step function pattern. TCP congestion window does not address this issue either as it does not allow an application direct control and is impacted by network conditions like latency, jitter, packet loss. Therefor there is a need for a mechanism which can address the overload control needs of HTTP applications in general for which existing mechanisms are not sufficient. Similar phenomena is observed for other protocols as well. For example, for SIP the issue is addressed by defining a specific mechanism in RFC 7339 [RFC7339]. Therefore This document specifies a generic overload control mechanism for HTTP/ HTTPS applications.

The notion of applying different drop probabilities for separate categories is supported as well. A category is an abstract construct as far as this specification is concerned. It pertains to different

policy characteristics and is meaningful in the context of an application. It, for example, may refer to different priority levels for requests, e.g. regular, emergency, government agency.

2. Requirements Language

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.

3. Overload Control Mechanism

3.1. Algorithm

The overload control algorithm described in this document relies on the concept of dropping a request with a certain probability on the client which otherwise would bee sent to a particular server.

This approach is favored considering that neither the number of clients sending requests to a server nor the volume of requests from a client is known a priori at a given point in time and can change dynamically. This dynamic nature makes any alternative approach which would utilize a quota based control on a per client basis unviable. An efficient solution should act in a as fair as possible way on the aggregate of requests received by a server from all clients.

3.2. Capability Negotiation

A client supporting this specification SHOULD add a Pragma header to a request for which overload control is applicable with the value "overload-control". A client MAY choose to send this Pragma header only for the first or some requests to the server.

Example:
Pragma: overload-control

A server supporting this specification MUST interpret presence of the Pragma header with the value "overload-control" as an indication that client supports this specification and SHOULD follow server related semantics defined in this specification. Receipt of a single such request SHOULD be interpreted as the client supporting the overload control mechanism.

A server not supporting this specification ignores a Pragma header with the value "overload-control".

A server MAY assume that a client supports this specification even if no Pragma header with the value "overload-control" is received.

3.3. Overload-Control Header

A new entity header named "Overload-Control" is defined and used in HTTP responses.

It is used to carry information about drop probability for messages in a specified category.

TBD: A parameter indicating validity time will be added.

Example:
Overload-Control: oc=1, odp=30; oc=2, odp=45; oc, odp=60

- Drop category1 requests with a probability of 30%
- Drop category2 requests with a probability of 45%
- Drop category3 requests with a probability of 60%

Drop probability indicated by an Overload-Control header without a category parameter applies to all categories.

3.4. Procedures at the Client

A client supporting this specification SHOULD drop a request which would be sent to a server based on the probability value received in Overload-Control header received in the latest response. This SHOULD be done as follows:

Create a local table for all categories. All categories initial drop probability is "0".

When an Overload-Control header is received:

- Update the drop percentage for the category in the local table.
- Drop a request based on the probability value for its category in the local table.

A client MUST interpret and honor a Retry-After header according to existing HTTP standard even if it supports and uses overload conytol mechanism specified in this document.

3.5. Procedures at the Server

A server supporting this specification SHOULD determine its own overload state. How this is done is implementation dependent and not subject to standardization in the context of this document.

A server SHOULD add an Overload-Control header indicating the category and the drop probability of requests to a response if its overload state changes. The mapping between actual overload state and drop probability is implementation dependent.

3.6. Procedures at the Proxy

There is no specific functionality required from a Proxy in the context of this specification. Proxies will transparently pass Pragma headers in requests and Overload-Control headers in responses.

4. Formal Syntax

The syntax of the Overload-Control header is described as follows:

```
   Overload-Control = "Overload-Control" HCOLON 1*(*(overload-
category)1(overload-drop-probability))
        overload-category = "oc COMMA" EQUAL overload-category-value
        overload-category-value = 1*ALPHA
        overload-drop-probability = "odp" EQUAL overload-drop-probability-
value
        overload-drop-probability-value = 1DIGIT/2DIGIT/"100"
```

5. IANA Considerations

Overload-Control is defined as an "entity" header and does not need to be registered in IANA "Permanent Header Field Names" Registry List.

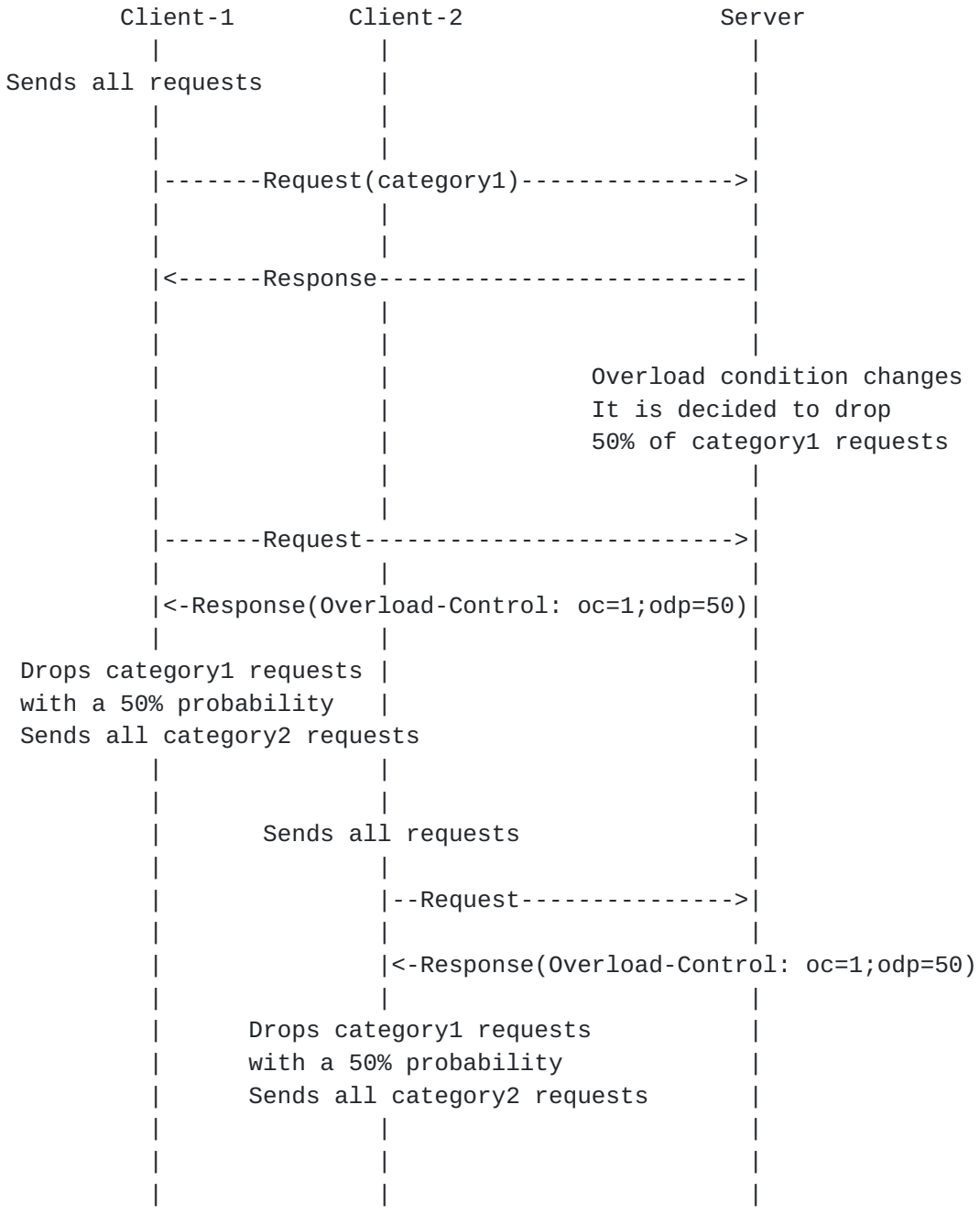
TBD: May change based on input from experts

6. Security Considerations

HTTPS SHOULD be used between client and server if overload control mechanism is used. This is needed to prevent attackers to change the drop probability used by clients to a value other than the one intended by the server.

7. Acknowledgements

8. Appendix A: Example Message Flow



9. Informative References

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

[RFC7339] Gurbani, V., Ed., Hilt, V., and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control", RFC 7339, DOI 10.17487/RFC7339, September 2014, <https://www.rfc-editor.org/info/rfc7339>.

Author's Address

Tolga Asveren
Ribbon Communications
Freehold, NJ 07728
USA

Email: tasveren@rbbn.com