Internet-Draft
SOC Working Group (concluded)
Intended status: Standards Track
Expires: April 3, 2020

P M Williams, Ed.
NICC Standards & BT

October 3, 2019

Session Initiation Protocol (SIP) Non-eXempt Rate Control
draft-williams-soc-nxrate-control-00.txt

Status of this Memo

Copyright Notice

carefully, as they describe your rights and restrictions with respect to this document.

Abstract

A SIP overload control signalling protocol framework has previously been published, with the flexibility to allow different SIP request rate limiting algorithms to be selected. This document proposes a similar algorithm of maximum rate type with two advantages over existing proposals. Firstly, it exempts certain classes of SIP requests that are fundamental to correct operation of the SIP protocol which, if rejected by control, would worsen rather than improve SIP performance. Secondly, it allows target servers to control SIP traffic from sources not compliant with this document so that it can be deployed in heterogeneous network environments.

Table of Contents

1. Introduction

A SIP signalling protocol for control of SIP overload is already specified [RFC7339]. Although this framework supports the use of alternative client restriction algorithms, so far only proportional "loss" - the mandatory default method in [RFC7339], and maximum "rate" - the default of [RFC7415], have been defined. In both cases the value of the control parameter, whether the % of requests rejected or the maximum rate of requests respectively, applies to the entire stream of requests that a (source) client is trying to send to the (target) server subjected to overload. The priority assigned to requests, that determines the probability of being admitted or rejected by the clients, is the responsibility of clients alone.

This has several serious drawbacks. Certain within-dialogue requests methods are critical to the performance of SIP and therefore should never be rejected because doing so would lead to timeouts, retransmissions, inefficient use of resources and degraded user experience. Leaving the decision up to clients alone about which requests these are and how they are treated is unreliable. Some could make poor decisions about which request methods are exempt from control.

Because certain requests are exempt from rate limiting, for the maximum rate-based method the rate enforced at the client is lower than the entire sent rate. This rate reduction requires every client to execute a function - in real-time because the exempt traffic mix changes over time - to map from the received oc parameter rate to a lower rate that applies only to requests that are not exempt from restriction. As a consequence a server that is the target of overload

could be receiving requests from a range of sources where each client is dynamically making different choices about which requests are exempt and which are not. This compromises the ability of the target server to protect itself and maintain good throughput.

This new recommendation proposes a simpler and more reliable approach, which is to specify the exempt request methods so that they are known in advance by server and clients, and for the target server (rate-based method) to produce a maximum rate that applies only to request methods that are not exempt from rejection by control. The framework of [RFC7339] conveniently enables this if a new oc-algo parameter value "nxrate" (non-exempt rate) is introduced. The algorithm negotiation mechanism of [RFC7339] still applies so that clients and server can differentiate between "nxrate", "rate" and "loss" and therefore still operate in a mixed control scheme network environment whilst enabling convergence to support any one of these.

Also included is an enhanced rate control algorithm at the target server that is a simple extension of the [RFC7415] default, to control traffic from (source) clients that do not comply with the use of the oc Via header parameters, but in a way that favours compliant sources. This is essential in the likely situation that a network has connection between many communication or equipment providers, some of which do not support the use of these oc parameters. The need for this is recognised in [RFC7339] (section 5.10.2) but a solution is not provided.

The appendix of [RFC7339] provides an assessment against the requirements for SIP overload control as described in [RFC5390]. See the Appendix A RFC5390 requirements herein for an updated assessment against this specification.

2. Terminology

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 [RFC2119].

In this document a "source" is used to refer to any SIP (client) entity that sends traffic to a downstream (next hop) SIP (server) entity, referred to as a "target".

Limiting the rate of sending SIP requests will be referred to as "restriction", and an instance of a control that does this is referred to as a "restrictor". Elsewhere in published literature

these terms are sometimes referred to as "throttling" and a "throttle" respectively.

If a SIP request is not admitted by a restrictor, then "rejection" means that an explicit SIP failure response is returned, whereas "discard" means that the request is ignored with minimal action taken (such as being counted) and in particular no SIP response is returned.

3. Overview

The functional architecture is identical to that described in [RFC7415], i.e. the target (server) is subject to load and protected by the overload control scheme and the sources (clients) restrict traffic towards the target.

The Via header parameters oc, oc-algo, oc-validity, oc-seq have the same usage as defined in section 4 of [RFC7339], with the same method of algorithm selection as section 5.1.

Here we introduce the new oc-algo non-exempt rate token "nxrate".

4. Request priorities, including exemption from restriction

According to [RFC7339], prioritizing which requests are subject to rejection by control is ’largely’ a matter of local (client) policy (section 5.10.1) and, in order that client and server have a common understanding of which messages overload control applies to, the oc parameter value applies to all requests from client to server (section 5.3).

Because of this approach the server will not know which request types may be rejected by a client and which are exempt from control, which could also vary from one client to another. Furthermore if certain request types are to be exempt from control then each client must compute an adjustment to the oc parameter value received from the server. For the rate-based method of [RFC7415] this amounts to a ’rate reduction factor’ that multiplies the rate to obtain a lower maximum admit rate. This must be dynamic since it depends upon the request method mix, and the computation method and its parameters (including update frequency) may vary from one client to another. It is possible that some clients may make poor decisions about which request methods are exempt from control or have poor implementations of the rate reduction factor function. This could have consequences

for the target server to properly protect itself and maintain adequate and effective throughput.

In contrast this specification assumes that certain request methods, defined below, are critical to the performance of SIP and must not be rejected by control, and these exempt methods will therefore be known in advance by server and clients. By means of defining a new oc-algo parameter token value (section 5) the client will understand that the oc parameter only applies to request methods that are not exempt, so there is a common understanding of which messages control applies to. This approach also simplifies implementation and configuration since it obviates the need for each client to calculate a rate reduction factor value in real-time.

4.1. Exempt request methods

Because a SIP response to an ACK is disallowed, not admitting an ACK would lead to timeout and retransmission. Similarly rejecting a PRACK would lead to timeout and retransmission. CANCEL and BYE both result in terminating a dialogue and therefore freeing up resources. For these reasons the following request methods, and only these methods, MUST be exempt from restriction: ACK, PRACK, CANCEL, BYE.

These four request methods are implicitly prevented from overloading the SIP network by limiting the rate of sending dialogue-initiating INVITEs, as follows.

- An ACK is sent to confirm the receipt of a final response to sending an INVITE (whether dialogue-initiating or within dialogue), and hence there is a one-to-one correspondence between INVITE's sent and ACKs sent. Consequently in the longer term (although not under very short-term transients) limiting the rate of sending INVITEs will also limit the rate of sending ACKs;

- A PRACK is sent to confirm receipt of a provisional response to an INVITE, so that there is a one-to-one correspondence with each provisional response, and although there is no strict bound on the number of provisional responses per INVITE sent, in practice the number is likely to be very small (and often one);

- Similarly there is an upper bound on the ratio of CANCELs to INVITEs since there can be at most one CANCEL per INVITE (and usually zero);

- o BYE is the only way to terminate a confirmed dialogue - and therefore release associated resources - and hence there is a similar correspondence with dialogue-initiating INVITEs, although the timescale of dependence is greater.

Note: As per section 14 of [RFC3261], automatic generation of BYEs following detection of media failure should not be synchronised, i.e. sending should be spread out over time (e.g. randomised), in order to avoid overload. If needed a server may control the non-exempt (e.g. INVITE) acceptance rate to reduce the queuing and processing time required for BYEs.

4.2. Non-exempt request priorities

In line with section 5.10.1 of [RFC7339], the non-exempt priority levels SHOULD be derived with respect to the general principles that requests within a dialogue shall be given higher priority than those that are not within a dialogue, and that the highest priority levels of non-exempt requests includes all those associated with emergency calls as identified by the SIP user or SOS URN, or contents of the Resource-Priority header, etc. Note that although these non-exempt requests are given the highest priority, it is possible that (for example) emergency traffic alone could be sufficient to cause overload, therefore they cannot be made exempt from restriction.

4.2.1. Factors determining priorities of within dialogue request methods

It is undesirable to reject within-dialogue (early or confirmed) requests because this could prevent certain sessions from being established and would cause inefficient use of SIP resources or poor quality of service.

For example, an INVITE may be sent within a dialogue in order to modify the dialogue state or session parameters. Rejecting such a re-INVITE may result in termination of the session, causing poor quality of user experience, inefficient use of resources - which had previously been devoted to the session, and a possible user retry. The UPDATE method is similar, but in addition it may be sent for an early dialogue and hence required for dialogue initiation. In this case rejecting it may prevent certain types of sessions from being setup.

Although it is undesirable to reject such within-dialogue request methods, more extreme/unusual traffic conditions could arise which dictate that such methods need to be rejected. For example, if there

were a very large number of concurrent dialogues at a SIP node because the holding time is large and the rate of within-dialogue requests such as UPDATEs for every dialogue is also high (perhaps because of some future application or because of rogue behaviour), if UPDATEs were exempt from restriction then overload could arise and the control would be unable to do anything about it.

4.2.2. Illustrative numerical values of priority

It is convenient to define default priority values by means of counting numbers, ordered inversely to importance. The requests with the highest importance, exempt requests, have a priority of 0 which does not change, and the remaining non-exempt request priorities are in order of decreasing importance, from 1 upwards. This makes it straightforward to map one-one between priorities and multiple leaky bucket thresholds of the default restriction method included in section 3.5.2 of [RFC7415]. It also means that that priority values can be straightforwardly assigned by configuration of SIP or the SIP user.

It is recognised that the most important requests, which includes emergency calls, may include more than one category of request, so we assume that there are n distinct additional such levels.

The principles above result in default priority values having the requests associated as shown in Table 1.

When the request method is complemented by the other criteria, then with a single level of highest priority non-exempt requests (n=1), we obtain the complete list of assigned priority values shown in Table 2.

priority values	requests methods
0	exempt methods ACK, PRACK, CANCEL, BYE
1...n	highest priority non-exempt request levels, including emergency calls
1+n	within-dialogue (confirmed or unconfirmed) methods, not associated with highest non-exempt priorities
2+n	out-of-dialogue methods other than INVITE or REGISTER, not associated with highest non-exempt priorities
3+n	out-of-dialogue INVITE or REGISTER methods (e.g. new calls), not associated with highest non-exempt priorities

Table 1: Descriptions of requests associated with default priority values

Request Method	exempt ?	within dialogue?	highest priority non-exempt?	priority Value
ACK	yes	yes	n/a	0
BYE	yes	yes	n/a	0
CANCEL	yes	yes	n/a	0
PRACK	yes	yes	n/a	0
INFO	no	yes	no	2
INFO	no	yes	yes	1
INVITE	no	no	no	4
INVITE	no	no	yes	1
INVITE	no	yes	no	2
INVITE	no	yes	yes	1
MESSAGE	no	no	no	3
MESSAGE	no	no	yes	1
MESSAGE	no	yes	no	2
MESSAGE	no	yes	yes	1
NOTIFY	no	yes	no	2
NOTIFY	no	yes	yes	1
OPTIONS	no	no	no	3
OPTIONS	no	no	yes	1
OPTIONS	no	yes	no	2
OPTIONS	no	yes	yes	1
PUBLISH	no	no	no	3
PUBLISH	no	no	yes	1
REFER	no	no	no	3
REFER	no	no	yes	1
REGISTER	no	no	no	4
REGISTER	no	no	yes	1
SUBSCRIBE	no	no	no	3
SUBSCRIBE	no	no	yes	1
SUBSCRIBE	no	yes	no	2
SUBSCRIBE	no	yes	yes	1
UPDATE	no	yes	no	2
UPDATE	no	yes	yes	1

Table 2: Priority value assignment to request methods with additional criteria, for a single highest request category that includes emergency calls (n=1 highest category)

5. "nxrate" token for oc-algo parameter value

According to section 5.3 of [RFC7339] the server produces an oc parameter value that "...it expects the receiving SIP clients to

apply to all downstream SIP requests (dialogue forming as well as in-dialogue)...". Similarly section 3.4 of [RFC7415] requires that " The maximum rate determined by the server for a client applies to the entire stream of SIP requests, even though throttling may only affect a particular subset of the requests...".

The scheme specified here interprets the oc parameter differently because it only applies to the non-exempt requests that have been defined in section 4.1. To ensure that a client communicating with overload control capable servers can differentiate the max rate-based method of [RFC7415], which uses the oc-algo token "rate", from this max rate-based method, we introduce the new token "nxrate" indicating that the oc value provides the non-exempt rate.

5.1. Control algorithm selection

For a source (client) to be compliant with non-exempt rate-based control it MUST include the token "nxrate" in the oc-algo list.

For a target (server) to be compliant with non-exempt rate-based control it MUST select and return "nxrate" as the only oc-algo list value if and only if "nxrate" is present in the oc-algo list received from a source. In addition, if "nxrate" is not present it MUST treat the source as non-compliant and allocate a local restrictor as defined in section 6.1.1.

6. Rate control algorithm

Clients MUST implement a type of algorithm that enforces maximum admission rate (as required by [RFC7415]), which shall also be able to differentiate between the range of non-exempt priority levels defined in section 4.2, in such a way that higher priority requests are admitted in preference to lower priority requests, whilst the total admission rate for all priorities of non-exempt requests is subject to the maximum non-exempt rate.

The default leaky bucket algorithm in section 3.5.1 of [RFC7415] with the additional features in section 3.5.2 (for priorities) SHOULD be used, where the number of 'tolerance' reject thresholds required will be equal to the number of priorities. The features of section 3.5.3 SHOULD also be included to avoid performance degradation due to 'resonance' which can occur when there are a large number of sources connected to a target.

Isomorphic algorithms that give identical behaviour MAY be used as alternatives to a leaky bucket, e.g. a token bank rather than a leaky bucket, whereby the bank is filled with tokens at a rate equal to the leak rate.

Servers employ an enhanced version of the rate control for non-compliant or non-conforming sources, as described in the following subsections.

6.1. Target server rate control for non-compliant or non-conforming sources (clients)

For a realistic overload control deployment, especially for connection between networks managed by different communication providers, or where the control capability is being deployed in stages, a SIP server is very likely to be connected to a mix of some client nodes that conform with an overload control protocol ([RFC7339] or [RFC7415]) and others that do not. A source (client) is conformant if it is both compliant with the overload control protocol but also modulates the SIP requests that it sends to a target according to the control information that it has received; being compliant means that it correctly advertises support for an overload control in the SIP requests that it sends. Therefore a non-conforming source may be non-compliant or may be compliant but not apply control properly.

A target must be able to protect itself effectively against traffic from non-compliant sources. But rejecting requests incurs an overhead that must be limited by the target, else it cannot properly protect its capacity and prevent overload. In a less trustworthy/tested network environment it must be able to protect against non-conforming sources as well. Furthermore it should be advantageous for sources to become compliant and conformant as an incentive for this overload control scheme to be deployed and a deterrent to malicious non-conformance.

Some of the above factors are recognised in section 5.10.2 of [RFC7339] where requirements on a server connected to non-compliant clients are outlined, but a solution is not provided.

To address this a rate control algorithm is defined (section 6.1.1) that is an enhancement of the default algorithm in [RFC7415]. When a such rate controller should be allocated at a target, and how it treats arriving requests is described in section 6.1.1, and the

steady-state throughput of this algorithm in terms of different request outcomes is analysed in section 6.1.4.

6.1.1. Enhanced rate control algorithm for target server

The target restriction function MUST use the type of algorithm defined as the default in [RFC7415], including multiple priorities as per section 3.5.2, and with the following additional features:

- An additional bucket 'discard' threshold TAU* that is greater than any other threshold. Whenever a request (including those that are exempt) is received and the fill is greater than TAU*, then the request is discarded, i.e. ignored without sending a response.

- The bucket fill is increased when rejecting a request as well as when admitting one. The increase fill amount represents the cost of rejection and will be system dependent. Some possibilities are outlined in section 6.1.4. The fill is NOT increased when discarding a request.

Isomorphic algorithms that give identical behaviour may be used as alternatives to a leaky bucket, e.g. a token bank rather than a leaky bucket, whereby the bank is filled with tokens at a rate equal to the leak rate, and that take account of the cost of rejection and allow discard of requests in a similar way.

Note: The algorithm described in [RFC7415] compares the bucket fill with a reject or 'tolerance' threshold before admitting a request, so that if admitted the fill may increase above the threshold. Similar rate control behaviour results if the fill comparison is made assuming that the request would have been admitted (even if it then isn't), in which case the fill cannot increase above the threshold for that request on admission, although it can on rejection. In the latter case the maximum fill of the bucket is the discard threshold, whereas in the former case the maximum fill will be the discard threshold plus the maximum increase on rejection.

The original algorithm, without the above enhancements, which is used for the source restriction function (since discard is forbidden at this point), is in fact a special case of the enhanced algorithm, obtained simply by setting a 0 cost of rejection. As long as the discard threshold TAU* is sufficiently larger than the highest reject threshold, it will have no effect since the fill cannot attain TAU*.

6.1.2. Rate controller allocation and treatment of requests for a target server

When a target server activates overload control it MUST derive a maximum non-exempt rate - the control rate - for every source from which it receives SIP traffic. It SHOULD allocate a rate controller of the type defined in section 6.1.1 to control traffic from every source that is non-compliant and it MUST allocate one for any non-compliant source where the received non-exempt rate exceeds the control rate.

To guard against any non-conformant sources the target MAY allocate a rate controller against compliant sources as well. This would be desirable for less trusted or less tested connections, but may be deemed unnecessary within a trusted or wholly owned network domain.

Every request from the source MUST query the restrictor on arrival at the target, including requests that are exempt from restriction at the source. As a result the request may be

- admitted,
- rejected with response, but only if the request is non-exempt,
- discarded, possibly even if the request is exempt (under an extreme condition defined in section 6.1.1, depending upon the leaky bucket implementation variant).

To discard means that the request is ignored, i.e. the receiving target does not take any further action (such as returning a SIP response), apart from possibly counting the discard action e.g. for management statistics.

6.1.3. Summary of performance characteristics of the solution

As shown in section 6.1.4 the solution has the following desirable behaviour:

- For a request arrival rate up to the control (oc) rate derived by the target, requests will be processed normally;
- As the request arrival rate rises above the control rate, the rate of admitted requests decreases and the remainder are rejected;

- After the admission rate reaches 0 (because all requests are being rejected), as the arrival rate increases further, the arrivals that make up the excess rate are discarded, using minimal processing. Once in this region the source receives no service from the target;

- Non-compliant sources are penalised if they exceed their assigned control rates because the success rate decreases and eventually they get no service at all.

In this way the target workload due to the arriving traffic represented by the rate controller always remains bounded.

6.1.4. Steady-state outcome rates of the enhanced target rate control

For any specific traffic source the maximum admitted request rate for that source as derived by the target will be related in some way to the average resource workload required to process requests according to the current mix of traffic. Whereas the default rate controller algorithm as specified in section 3.5 of [RFC7415] does not take account of the overhead of rejecting a request, the enhanced algorithm does so by increasing the bucket fill on rejection by an amount that depends in general on the system design.

For example this amount MAY be of the simple form T0+pT in terms of the current value of the bucket increment T (defined in [RFC7415] as the reciprocal of the oc rate value), with parameters T0 (constant) and p, the fraction of the cost of admission, where one or the other (but not both) of these could be 0.

The effect of the algorithm will be that as the request arrival rate increases above the max rate R, the admitted rate will decrease. This will be illustrated by an example.

The bucket leaks at a constant rate of 1, so that once the bucket is 'full' the bucket load is 1 and the arrival rate A and admission rate a are just equal (no rejects):

$$AT = aT = 1$$

Since T=1/R these rates equal to the oc rate R.

As the arrival rate increases the admission rate will (automatically) decrease just enough so that the bucket utilisation remains at 1, i.e.

```
    A + (A-a)(pT+T0)=1
```

which has the solution

```
         /                          A    A < R
    a = ( (R - A(p+RT0))/(1-p-RT0)     R <= A <= R/(p+RT0)
         \                          0    R/(p+RT0) <= A
```

so that as the arrival rate increases above R the admit rate decreases linearly with gradient -(p+RT0))/(1-p-RT0).

Once the bucket is rejecting every request, i.e. a=0, if the arrival rate continues to increase then A > R/(p+RT0) which implies that

```
    A + (A-a)(pT+T0) > R(pT+T0)/(p+RT0) = 1
```

i.e. the offered bucket load is greater than 1, and since it leaks at a constant rate of 1 this means that it is unstable and the occupancy (fill) would rise constantly. This is prevented by having the additional discard threshold TAU*. In this range of extreme arrival rate, we have:

```
    a = 0
    r = R/(p+RT0)
    d = A - r
```

where r is the reject rate and d the discard rate, i.e. the reject rate remains constant and the discard rate continues to rise as the difference between the arrival rate and the reject rate.

7. Server operation

7.1. Control rate (oc value) allocation over sources

The actual algorithm used by the server to determine its overload state and estimate a target maximum SIP request rate, which we will refer to here as the goal rate, is not within the scope of [RFC7415]. However the server is required to evaluate this goal rate and it must determine how it will distribute the rates over its client sources.

Although neither the objectives nor the method of this rate distribution function are prescribed by [RFC7415], they are critical since they affect the quality of service, both absolute, and relative between, different client sources. Although the method is not specified here, the objectives are made more precise.

7.2. Overload control objectives

The target server overload control MUST satisfy the following two objectives:

1. Objective (total rate received from all sources) - Prevent the target from overload and maximise the rate received by the target, subject to the goal rate bound, i.e. when the total source rate

o exceeds the goal rate, then the arrival rate is equal to, or very close to, the goal rate

o less than the goal rate, no requests are rejected at the sources, i.e. the rate received by the target is the entire source rate;

2. Objective (allocation of rates over sources) - Shape the allocation of rates received from each source in a predictable way that can be regarded as 'fair' in some precise sense or conforming with an SLA previously agreed with client sources.

The first objective will make the most effective use of the target capacity - in particular it is essential to avoid 'over-restriction' whereby the received rate is significantly lower than the goal if the total arrival rate at the sources is less than the goal rate, or 'under-restriction', whereby insufficient demand is rejected, which could lead to the problems of overload.

To demonstrate the importance of Objective 2 it is useful to give examples of control behaviour that would be unacceptable or at least highly undesirable, even though they both meet Objective 1:

1. Example - It may be possible to satisfy Objective 1 in a way so that some sources are denied any service at all (given an oc rate of 0). Furthermore, such an extreme distribution could vary over time, i.e. those sources that are denied any service could change, perhaps randomly.

2. Example - The overload of a target may be dominated by traffic from just a few sources, thereby significantly affecting the rate of loss due to rejection experienced by traffic from other sources, even though their levels of traffic are relatively low at 'normal' expected levels.

8. Target server failover configuration

In order to provide resilience to failure it is usual for each active target server to have a standby. It is possible for a target server failover to occur whilst overload control is ongoing. This requires special consideration.

Active and standby servers may or may not share IP addresses and they may or may not share overload control state. These factors affect overload control behaviour during failover and choice of oc-validity and oc-seq parameter values.

The control parameters that are returned from an overloaded target are associated with its IP address and port number. If that server fails the restrictor allocated at the source against that address/port will persist through the failover with the oc max rate parameter value from the failed target, as long as the duration timer (of length oc-validity) remains running or until an update with a new oc value and higher oc-seq value is received for the same address/port. If the activated standby has a different IP address and it signals overload then a new restrictor would be allocated against that address, unless an identification between the active and standby addresses is made by the source. If control state is not shared then the activated standby would not start in an overload state and therefore return an oc-validity value of 0, which would have the effect of terminating any control already active against its address/port.

The above factors have implications for the minimum size of the oc-validity parameter and the oc-seq values that are sent in responses from the standby target as it activates.

8.1. oc-validity values

The required minimum size of oc-validity can be determined with reference to the timing of events shown in Error! Reference source not found..

Under normal conditions when the target remains active, to prevent a control restrictor at a source expiring before another update is received, thereby causing a sudden surge of traffic being sent, the oc-validity value should be larger than the time between updates (which may be variable) and the time it takes to distribute control to all sources that are sending at a ’sufficiently high’ rate. Such sources are those sending at a rate above the max control rate

determined by the oc value (if a source with a lower rate terminates control it would have no effect on the admitted traffic, so such sources do not need to receive an update - which would take a long time to be received in any case). Since the distribution time will be variable it is safer to use an oc-validity value equal to twice the (maximum) time between updates, since updates should not normally be made before distribution has become effective.

Considering now failover from an active target, in terms of ensuring that control persists for long enough according to the oc-validity size, the worst time for this to occur would be just before all updates have been received (see Error! Reference source not found.). So the restrictors should persist for an additional time which is the time it takes for the newly activated server to start responding and stabilise. This duration will depend upon several factors, including

o the method of target failure detection used by the sources

o whether or not state is shared between active and standby

o whether overload control is active before or after the failover (or both)

Therefore the minimum value of oc-validity returned by a server MUST be

2x(time between control updates)
+ (expected duration of failover stabilisation)

In addition if client/sources receive updates at nearly the same time, then if they were all to apply the same oc-validity value the duration timers would expire near simultaneously, resulting in a surge of traffic. Therefore it is better to spread out the expiry times.

Values of oc-validity returned by an overloaded target MUST be distributed over a range with the minimum value above. The maximum of this range SHOULD by default be

3x(time between control updates)
+ (expected duration of failover stabilisation)

so that the range duration is the length of time between control updates.

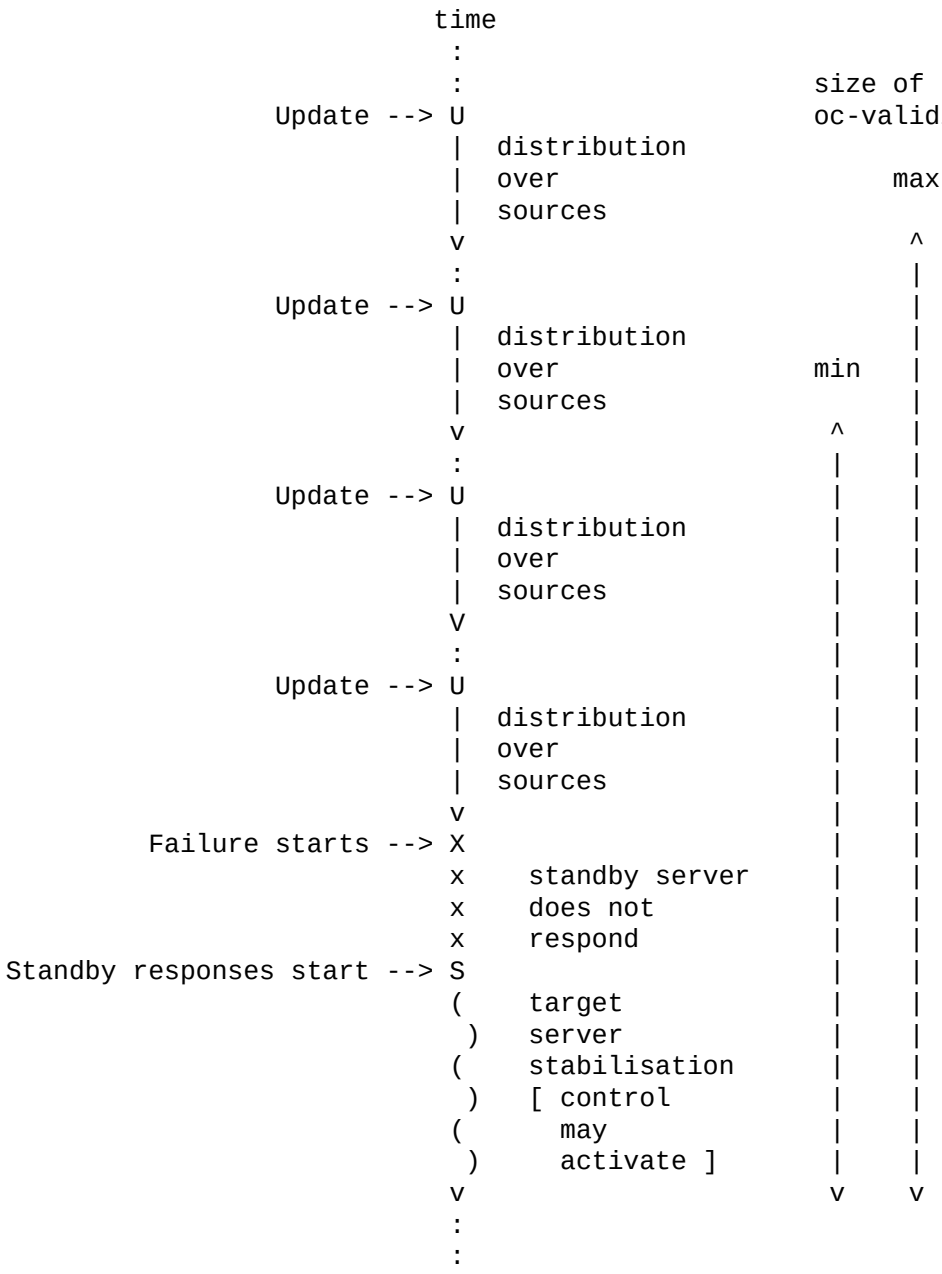

Figure 1 Timeline of events around target server failover effecting the minimum duration of the oc-validity parameter

It is not possible to be precise about the default oc-validity applied by a client because it depends upon the server behaviour and the network connectivity. However, unlike the proportional loss-based method of [RFC7339] which needs to be updated more frequently because, for the same rejection probability, the admitted rate is proportional to the arrival rate (which is in general changing), there is less risk with leaving control on for longer and more risk with terminating it prematurely.

Therefore it is recommended that the client SHOULD use a default value of 10 seconds rather than the 500ms of [RFC7339] and [RFC7415].

8.2. oc-seq values

During normal operation the oc-seq value is increased by a server according to section 4.4 of [RFC7339]. We clarify this process, and add further recommendations under failover conditions.

The oc-seq value MUST be increased when a control update (i.e. a re-evaluation of the oc value) has been performed by the server, even if this results in the same oc value. This is necessary to cause the oc-validity timer to be restarted, as per section 5.4 of [RFC7339]. The oc-seq SHOULD NOT be increased when there has not been a control update because this would cause the unnecessary overhead of restarting/extending the oc-validity timer and re-applying the same received oc value of rate, in effect for every request received by the target.

The oc-seq value after failover depends upon whether control state is shared between active and standby servers.

8.2.1. Control state is shared

Control updates of the standby are straightforwardly managed by increasing oc-seq from the value before failover.

8.2.2. Control state is not shared

The value of the oc-seq parameter sent in responses MUST be set lower than the value sent before failover, during the period of stabilisation when the newly activated target is not in an overload state (oc-validity=0 sent). This is required to ensure that the responses do not terminate any already active control at the sources.

If the oc-seq value is derived from the current time, then this MAY be achieved by deriving the oc-seq of the activated standby from the following expression which gives a time before the standby becomes activated (on restart or failover):

(activation time) - (maximum configured server oc-validity value)

A consequence of this approach, together with the setting of sufficiently long oc-validity timer (as per section 8.1), is that, if overload control of the server before failover was active, then the standby is less likely to enter an overloaded state for a period after failover. However it is likely to go into overload when the oc-validity duration timers expire at the sources, and to ensure that oc values are effective at sources the oc-seq value must then be derived in the normal way. I.e. the server MUST revert to its normal oc-seq derivation according to the current time at the start of overload control following activation of the standby, when oc-validity > 0 for the first time.

In summary, if oc-seq is derived from time, then the only changes to its value will be when control updates are made or when (or before) the server becomes active.

8.3. Relationship of source restrictor to a target and its standby

A source applying control identifies an overloaded target by its IP address and port number. Thus if a target server and its standby

- share a common address/port then a single restrictor at the source will necessarily be associated with both the target and its standby.

- do not share a common address/port then there will be an independent restrictor associated with each. However if they share overload state then the new restrictor allocated on failover will be quickly assigned the control state of the failed target.

Note that there is no SIP mechanism defined by which a source can discover which of the above address configurations is the case.

Consequently a target MUST ensure that source rate control persists during failover of a target to its standby in one (or more) of the following ways:

- Share a common IP address and port number with its standby and setting the oc-seq values (section 8.2.2);

- Share overload control state with its standby (section 8.2.1);

- Use another mechanism that has been agreed between sources and the target server. If server and standby have distinct IP addresses this MAY be done by the source associating a single restrictor with both addresses.

9. Example - including failover

The example generalises the example in Section 4 of [RFC7415] with multiple client sources s1,s2,...,s8 sending requests to an active downstream target server T1, which has a standby T2. Overload control activates for T1 which later fails whilst overload is ongoing so that T2 takes over. This example assumes that T1 and T2 have the same IP address/port but they do not share control state, which is one of the cases in section 8.3.

```
INVITE sips:user@example.com SIP/2.0
Via: SIP/2.0/TLS s7.example.net;
branch=z9hG4bKs714400.3;
oc;oc-algo="nxrate,rate,loss"
...
SIP/2.0 100 Trying
Via: SIP/2.0/TLS s7.example.net
branch=z9hG4bKs714400.3;received=192.0.2.117;
oc=0;oc-algo="nxrate";oc-validity=0;
oc-seq=1546214400.5
```

The first message above is sent by s7 to T1. This message is a SIP request; because s7 supports overload control, it inserts the "oc" parameter in the topmost Via header field that it created. s7 supports three overload control algorithms: "nxrate", "loss", "rate".

The second message, a SIP response, shows the topmost Via header field amended by T2 according to this specification and sent to s7. Because T2 also supports the non-exempt rate overload control method, it chooses this nxrate-based scheme and sends that back to s7 in the "oc-algo" parameter. It uses oc-validity=0 to indicate no overload control. In this example, "oc=0", but "oc" could be any value as "oc" is ignored when "oc-validity=0". Assume for this example that oc-seq is a time in seconds with precision 0.1 sec.

At some later time, T1 starts to experience overload. Just before the next response is about to be returned, to s3, the maximum rate admissible from s3 is 15 non-exempt requests per second. The time between target control updates is 3 seconds and suppose that prior design and validation has shown that stabilisation of failover from T1 to T2 takes at most 4 seconds. Therefore T1 returns the following SIP message indicating S3 should send SIP requests at a rate less than or equal to 15 non-exempt SIP requests per second and (as per section 8.1) for a duration chosen between 2x3+4=10 and 3x3+4=13 seconds (note oc-validity is in ms):

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TLS s3.example.net;
branch=z9hG4bKs314460.1;received=192.0.2.113;
oc=15;oc-algo="nxrate";oc-validity=12765;
oc-seq=1546214460.4
...
```

Just after this response has been sent the server T1 fails and no longer responds to requests, which causes T2 (which has the same IP address/port) to takeover. Suppose T2 starts responding 0.5 sec later and after receiving an INVITE from s8 it returns the following indicating that overload control should be inactive because it had no prior load and it does not shared control state with T1 and therefore is not aware that T1 was in overload:

```
SIP/2.0 100 Trying
Via: SIP/2.0/TLS s8.example.net;
branch=z9hG4bKs814460.2;received=192.0.2.118;
oc=0;oc-algo="nxrate";oc-validity=0;
oc-seq=1546214447.9
```

Because T2 is activating it has taken the actual time 1546214460.9 and reduced it by 13 secs to derive the oc-seq value according to section 8.2.2. Consequently the rate control at source s3 (and similarly for other sources s1,...,s8) will not be terminated even though it has received oc-validity=0 because the oc-seq value is lower than the last value received. T2 is now receiving requests at approximately the same rate as T1 was. 7.1 seconds after activating, T2 determines that it needs to activate control and so it sets oc-seq according to the actual time and chooses another non-zero value of oc-validity in its configured range:

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS s1.example.net;
branch=z9hG4bKs114467.2;received=192.0.2.111;
oc=0;oc-algo="nxrate";oc-validity=10763;
oc-seq=1546214468.0
```

It is returning this to s1 knowing that it cannot have already received a higher oc-seq value (in the recent past).

10. Syntax

This specification extends the existing definition of the Via header field parameters of [RFC7339] as follows:

```
algo-list =/ "nxrate"
```

11. Backwards compatibility

As highlighted in section 10.2 of [RFC7339], a new overload control protocol needs to be able to be gradually introduced into a network and function properly if only a subset of the nodes support it.

This scheme can co-exist in a network where nodes support either [RFC7339], or [RFC7415], or neither. The use of the new oc-algo token "nxrate" together with the control algorithm type negotiation of section 5.1 of [RFC7339] enables any client or server to determine (dynamically) whether or not each neighbouring node supports this scheme. If a node does so as a server, then the method of section 6.1. enables allocation of a proxy source restrictor at the target for each non-compliant source. In terms of request priority differentiation, rejection response, etc, the arriving traffic is managed in the same way at the target as if at a source. The cost of rejection is also accounted for, so as to bound the total workload (resource utilisation) in such a way that the rate of requests admitted from a non-compliant source decreases as the arrival rate increases. Since the arrival rate from such a source may be unlimited, ultimately discard of some requests, i.e. throwing them away without returning an explicit response, will be necessary. In this way compliant sources are favoured over non-compliant or non-conforming sources.

12. Security considerations

This mechanism does not introduce any security concerns beyond the general overload control security issues discussed in section 11 of [RFC7339].

13. IANA considerations

This specification uses the Via header parameters defined in [RFC7339] and registered with IANA in the "Header Field Parameter and Parameter Values" sub-registry, appearing as follows, with the new value for oc-algo defined in section 10:

Header Field	Parameter Name	Predefined Values	Reference
Via	oc	Yes	[RFC7339]
Via	oc-validity	Yes	[RFC7339]
Via	oc-seq	Yes	[RFC7339]
Via	oc-algo	Yes	[RFC7339][RFC7415]

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC3261, June 2002.

[RFC4412] Schulzrinne, H. and J. Polk, "Communications Resource Priority for the Session Initiation Protocol (SIP)", RFC4412, June 2002.

[RFC7339] Gurbani, V., Ed., Hilt, V., and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control", RFC7339, September 2014.

[RFC7415] Noel, E., and P. Williams, "Session Initiation Protocol (SIP) Rate Control", RFC7415, February 2015.

14.2. Informative References

[RFC5390] Rosenberg, J., "Requirements for Management of Overload in the Session Initiation Protocol", RFC5390, December 2008.

15. Acknowledgments

Thanks are due to the following people for their contribution to and review of material included in this specification: Mark Ashworth, Chris Bovis, Ken Hatt, Adrian Moss, Nick Ireland, Tim Pierrepont, Martin Simmonds, Paul Theobald, Nigel Weinronk, Jonathan Welton, and other members of the NICC SIP Overload Control Task Group.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. RFC5390 requirements

Appendix B of [RFC7339] lists each of the requirements from [RFC5390] and assesses whether each is fulfilled. The subset of those requirements that this specification meets more effectively or more completely is listed below.

REQ 4: The mechanism must be capable of dealing with elements that do not support it so that a network can consist of a mix of elements that do and don't support it. In other words, the mechanism should not work only in environments where all elements support it. It is reasonable to assume that it works better in such environments, of course. Ideally, there should be incremental improvements in overall network throughput as increasing numbers of elements in the network support the mechanism.

> Met as follows: The issue of how a target server should treat traffic that it receives from sources that do not support overload control is discussed in section 5.10.2 of [RFC7339], but no solution is provided. This specification provides a simple but effective scheme in section 6.1. See also REQ 20.

REQ 5: The mechanism should not assume that it will only be deployed in environments with completely trusted elements. It should seek to operate as effectively as possible in environments where other elements are malicious; this includes preventing malicious elements from obtaining more than a fair share of service.

> Met as follows: The scheme described here in section 6.1 not only controls traffic from sources that are non-compliant, but also those that declare themselves to be compliant by means of the protocol parameters but (perhaps maliciously) do not restrict the request rate accordingly.

REQ 13: The mechanism must not dictate a specific algorithm for prioritizing the processing of work within a proxy during times of overload. It must permit a proxy to prioritize requests based on any local policy so that certain ones (such as a call for emergency services or a call with a specific value of the Resource-Priority header field ([RFC4412]) are given preferential treatment, such as not being dropped, being given additional retransmission, or being processed ahead of others.

Met as follows: A modified view of this requirement is taken here, i.e. that the requests should be divided into two subsets - those that are exempt from control, and those that are not. The exempt requests are defined (section 4.1) and therefore common for clients and servers. Recommendations are given for how to assign priority levels of non-exempt requests (section 4.2), together with default values (section 4.2.2), but this does not preclude the use of different choices.

REQ 17: The overload mechanism must not provide an avenue for malicious attack, including DoS and DDoS attacks.

Met as follows: As per REQ 5, a malicious attack from non-conforming sources with high traffic rates is controlled according to section 6.1. The issues associated with using insecure communications are the same as presented in [RFC7339].

REQ 20: In a mixed environment of elements that do and do not implement the overload mechanism, no disproportionate benefit shall accrue to the users or operators of the elements that do not implement the mechanism.

Met as follows: Objective 2 (section 7.2) requires that rates are allocated precisely over source elements, including those that are non-compliant, and the enhanced rate control algorithm (section 6.1) ensures that, in terms of admitted traffic, for a non-compliant source element there is a disadvantage that becomes greater the more traffic it sends.

Author’s Address

Philip Williams
NICC Standards
BT Technology
Adastral Park
Ipswich IP5 7RE
England

Email: phil.m.williams@bt.com