NICC ND1653 V2.1.1 (2020-08)

NICC Document

Overload Control for SIP in UK Networks

NICC Standards Limited

The Old Rectory,

Church Street,

Weybridge,

Surrey KT13 8DE

Tel.: +44(0) 20 7036 3636

Registered in England and Wales under number 6613589

NICC Standards Limited

1 Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to NICC.

Pursuant to the NICC IPR Policy, no investigation, including IPR searches, has been carried out by NICC. No guarantee can be given as to the existence of other IPRs which are, or may be, or may become, essential to the present document.

Foreword

This NICC Document (ND) has been produced by NICC SOC TG.

Introduction

Without a means to limit demand, network systems processing SIP signalling traffic may be subject to load in excess of the capacity to process it sufficiently quickly, leading to protocol timeouts or user abandons, and consequent retries, leading to a collapse of good throughput or ‘goodput’.

There are many possible causes of excessive demand, including unplanned events in the media, natural disasters, re-routeing due to network system failures or processing speed degradation.

In order to maintain throughput, the type of SIP requests admitted or rejected and the way in which this is done is critical. In particular, it is insufficient for a target node in overload to rely on autonomously rejecting SIP requests either by returning a failure response, thereby incurring a cost (e.g. processing) for each rejection, or more cheaply by discarding (and ignoring) SIP requests. Both of these actions cause a degradation in goodput.

It is therefore necessary for the rejection to be done externally to the target node which is in overload.

This specification defines such a method for use over SIP NNI in the UK.

It is designed to be extensible to any other SIP interface by configuration.

1 Scope

The present document provides overload control requirements for SIP Network to Network Interconnection implementations in accordance with ND1647 [1]. Its use in relation to other SIP interfaces is not precluded.

SIP overload control refers to the management of load on network elements by limiting the rate of SIP requests in order to maintain or maximise successful throughput and to limit signalling delays, in particular for session set-up.

Media congestion is not managed through SIP overload control, although where the media resource is a component of the overall SIP server its throughput may also be protected through these procedures.

The requirements in this document are based on the nodal overload control mechanisms defined in IETF RFC7415 [3], which in turn utilises the mechanisms defined in RFC7339 [4]. The use of a centralised network to network SIP overload control mechanism is outside the scope of the present document.

These requirements complement the SIP NNI protocol specification ND1035 [2].

As required by TSG for publication of this ND, an Internet Draft entitled *Session Initiation Protocol (SIP) Non-eXempt Rate Control* (draft-williams-soc-nxrate-control) was submitted to the IETF on the 3rd October 2019. Although it expired on 3rd April 2020 before prompting sufficient discussion that could have led to the development of an RFC, it is a useful companion document that is consistent with this one (and could be used again in future) which has therefore been incorporated into a ND [i7]. For the avoidance of doubt, the present document does not depend in any way upon the Internet Draft or the derived document ND1038 [i7], which is cited here only as an informative reference.

2 References

2.1 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] ND1647 SIP-NNI Basic Voice Architecture.

[2] ND1035 SIP Network to Network Interface Signalling (SIP-NNI)

[3] RFC7415 Session Initiation Protocol (SIP) Rate Control

[4] RFC7339 Session Initiation Protocol (SIP) Overload Control

2.2 Informative references

[i1] RFC7200 A Session Initiation Protocol (SIP) Load-Control Event Package

[i2] RFC3265 Session Initiation Protocol (SIP) - Specific Event Notification

[i3] ITU-T, Specification Description Language, Z.100-Z.106

[i4] ND1007 ISDN User Part (ISUP)

[i5] ND1433 UK-ISUP Automatic Congestion Control Guidelines: Effective Schemes

[i6] ND1027 UK BICC for use between PLMNs

[i7] ND1038 Internet Draft: Session Initiation Protocol (SIP) Non-eXempt Rate Control

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Client – Source of SIP traffic. The client throttles traffic towards the server.

Compliant source node – A source node that advertises support for overload control in the SIP requests that it sends as described in ND1653.

Conforming source node - A source node that is compliant and modulates the SIP requests that it sends to a target node according to the control information that it has received.

Discard – For function B to *discard* a request/query from function A means that B ignores the request, i.e. does not take any further action (in particular does not notify function A or store any request data) apart from possibly counting the discard action.

Goal Rate – The maximum arrival rate of SIP requests to a target node, over a given time interval, that will give acceptable performance. Note that in ND1653 this refers to the non-exempt request rate (§5) whereas in RFC7415 [3] and RFC7339 [4] this refers to the entire request rate.

Goodput – For a specific SIP signalling network node or subsystem we take this to mean the request rate through that system that is successfully onward routed. It does not include requests that 'turn bad' due to upstream time-outs or originating user abandons.

Non-compliant source node – A source node that does not advertise support for overload control in the SIP requests that it sends as described in ND1653.

Non-conforming source node - A source node that is either non-compliant, or compliant but fails to modulate the rate it sends SIP requests accordingly.

Reject – For function B to *reject* a request/query from function A, an explicit notification of rejection is sent from B to A (rather than ignoring the request). The form of this notification depends upon (and should be clear from) the context, e.g. if B is a control restrictor then the notification is internal, but at a SIP signalling level it would imply that a SIP response would be sent from SIP node B to SIP node A.

Restriction – The function of limiting the rate at which requests are sent

Restrictor – An element performing restriction

Server – Target for SIP traffic. The server is protected by the overload control algorithm.

Source Node – The functional entity which is transmitting SIP request messages across the SIP signalling interface to the target node.

Target Node - The functional entity subject to overload which is returning SIP response messages to the source node.

TAU – Synonymous with τ defined below in §3.2.

Throttle - Synonymous with restrictor

Throttling – Synonymous with restriction.

3.2 Symbols

For the purposes of the present document, the following symbols apply:

τ – Reject or '*tolerance*' threshold in the leaky bucket fill for the default restriction method used by RFC7415 [3] where it is called *TAU*.

τ^* - Value of the leaky bucket *'discard'* threshold that is supplemented to the default restriction algorithm of RFC7415 [3]. It must be greater than any other threshold, and is the maximum possible fill of the bucket.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ACC	Auto Congestion Control, for ISUP [i4,i5] or BICC [i6]
NNI	Network to Network Interconnect
OC-SIP	Overload Control for SIP – the general concept/mechanism of overload control as applied to the SIP protocol, for which this specification defines the preferred implementation for UK networks
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SOC	SIP Overload Control – referring specifically to the title and mechanism of RFC7339
SRC	SIP Rate Control – the title of RFC7415

4 Document style

This document contains both requirements and explanatory text. To distinguish explanatory text from the more succinct and specific requirements, text for the latter is presented in the following way:

Requirement: To highlight a requirement and differentiate it from explanatory text, it is prefixed (bold/italics) and the whole paragraph enclosed in a coloured box, as this example.

5 Use of SIP Rate Control (RFC7415)

Requirement: SIP Rate Control (SRC) as defined by RFC7415 [3] shall be used for SIP overload control, with the *oc-algo* token "*rate*" substituted by "*nxrate*". This token is interpreted by clients and servers to mean the maximum rate of SIP requests that are not exempt from restriction, i.e. dialogue-initiating requests or out-of-dialogue requests.

Therefore when signalling support for UK SIP overload control:

- A compliant (source) client shall include the token "*nxrate*" in the *oc-algo* list
- A compliant (target) server shall respond to the receipt of compliant overload control parameters (i.e. the *oc-algo* list includes "*nxrate*") by setting the *oc-algo* value to the token "*nxrate*"

RFC7415 is a hop-by-hop method whereby sending clients indicate support for overload control in requests, and feedback of control information is piggy-backed on responses by means of special Via Headers. The restriction algorithm is of maximum rate type (termed *rate-based* in the RFCs), which is indicated in the RFCs by the client offering the *oc-algo* token "*rate*" and is confirmed by the server in the responses. Whereas this value refers to all SIP requests, ND1653 uses the new token "*nxrate*" which refers to the subset of requests defined above. The reasons why this new value has been created are explained in §B.6 of Annex B (informative). Henceforth in this specification *rate* of SIP requests will refer to the rate of *non-exempt* SIP requests unless specified otherwise.

The generic abbreviation OC-SIP will be used to refer to the SIP overload control functions used by ND1653 in order to distinguish this from SOC, the title of RFC7339.

The signalling mechanism used is as defined by SIP Overload Control RFC7339 [4], which specifies general signalling procedures for SIP overload control, as well as a default scheme of proportional rejection type (termed *loss-based* in the RFCs). RFC7415 extends and qualifies these procedures for use with the rate-based scheme. Refer to §15.1 for interworking with these methods.

Note: The SIP Load Control Event Package RFC7200 [i1] has different objectives from ND1653, being concerned with destination-based load control, as well as its signalling infrastructure, using Subscribe-Notify methods (RFC3265 [i2]). It is therefore not adopted by NICC for the present application.

6 Architecture

6.1 Introduction

6.1.1 SIP NNI architecture

ND1647 [1] describes the (UK) SIP-NNI as comprising a signalling interface (Qs) and associated media interface(s) (Qm), operating between the signalling/media border functional entities in two different networks.

Figure 1, below, shows the scope of the SIP overload control mechanism covered by this document in relation to the functional model that is described by ND1647 [1].

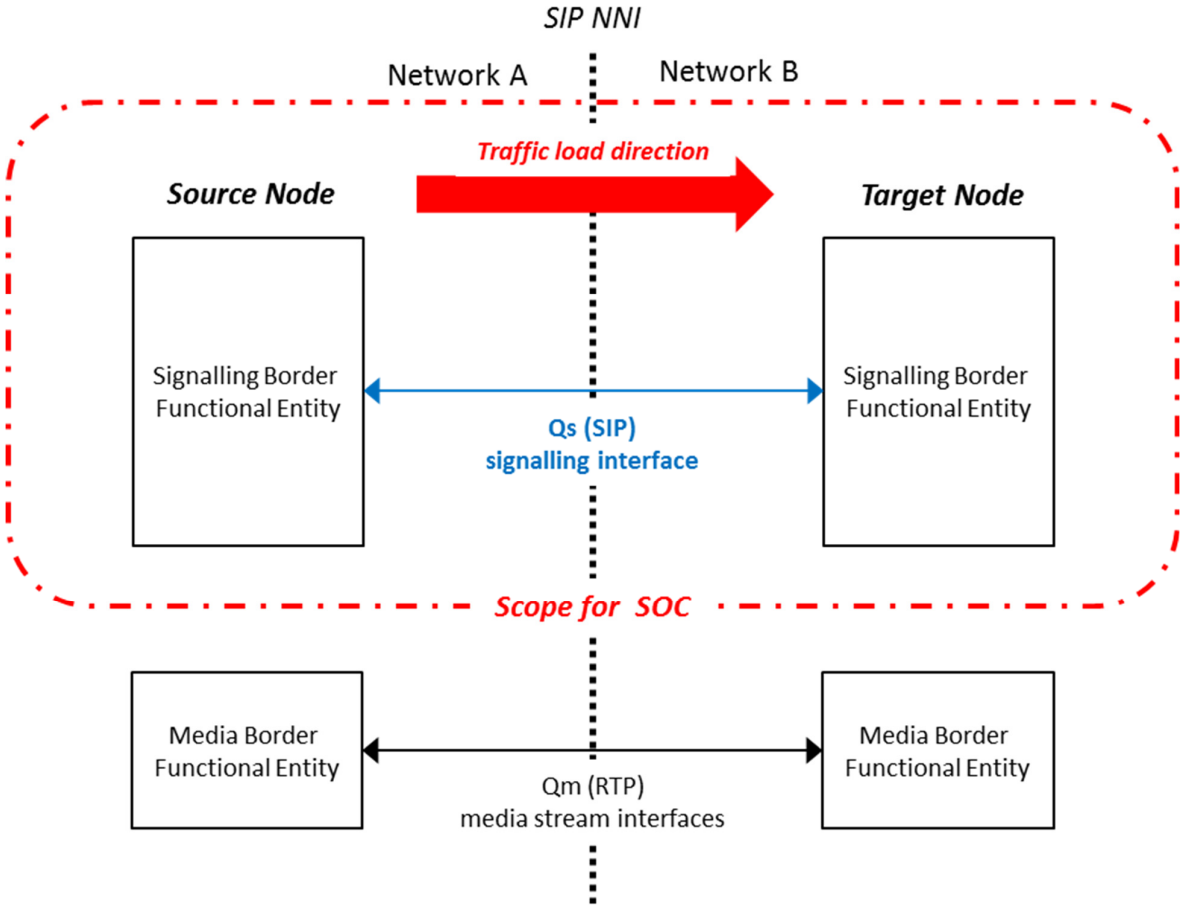

Figure 1: SIP-NNI Functional Model, with OC-SIP scope identified

Note: The functional model is symmetrical about the point of interconnect, so any given functional entity is likely to act as both a Source Node and a Target Node at the same time (with respect to the corresponding traffic load directions). However, the SIP/overload control functions themselves are independent in each direction (although there may be some common dependence in terms of resources and loading e.g. SIP client and server functions may be implemented by means of the same process and share the same resources, such as processors and memory).

It is only the interaction between the signalling border functional entities operating across the Qs (SIP) signalling interface that is in scope for SIP overload control.

In relation to the SIP overload control mechanism that is described in this document, the functional entity which transmits SIP request messages across the SIP signalling interface is referred to as the Source Node, whilst the functional entity which returns SIP response messages (to the Source Node) is referred to as the Target Node.

6.1.2 OC-SIP Mechanism

The SIP overload control architecture described by this document utilises the SIP Rate Control mechanism (SRC) that is described in IETF RFC7415 [3].

A specific parameter value (*oc* parameter values) carried in the Via header field of SIP responses is used to indicate what maximum SIP request rate the Target Node expects from an individual Source Node. The Source Node is required to apply a restriction to any (non-exempt) traffic that is destined towards that Target Node, to conform to the indicated maximum SIP request rate.

6.1.3 Handling of OC-SIP parameters

6.1.3.1 Source Node

Where a source node supports OC-SIP, the source node may advertise its OC-SIP capability (by including an *oc* parameter in the Via header of its request messages) to all target nodes that it is connected to or it may only advertise its OC-SIP capability to specific target nodes.

> Note: For example, a CP may specifically not want to use (or advertise support for) OC-SIP on a given NNI connection, even though its equipment has that capability.

Where a source node advertises support for OC-SIP to a target node, the source node shall process and react to any *oc*-related parameters that it may receive from that target node, as required by RFC7415.

Where a source node has not advertised support for OC-SIP to a target node, the source node shall not process, or react to, any *oc*-related parameters it receives from the target node. Ignoring unrecognised header fields is normal SIP behaviour. The SIP node may optionally report to the operator, that OC header fields are being received and not being acted on.

Where a source node advertises support for OC-SIP to a target node but does not receive any *oc* headers in responses from that node, it may optionally report this to the operator.

6.1.3.2 Target Node

Where a target node supports OC-SIP, the target node may confirm its support for OC-SIP (by including *oc*-related parameters in responses it sends back) to all OC-SIP supporting source nodes it is connected to or it may choose to only confirm its support for OC-SIP to specific such nodes.

Note: For example, a CP may specifically not want to use (or confirm its support for) OC-SIP on a given NNI connection, even though its equipment has that capability.

Where a target node supports OC-SIP but a source node has not advertised support for OC-SIP, the target node shall not include *oc*-related parameters in the Via header of response messages it sends back to that source node.

6.2 Basic functional model

6.2.1 Overview

The functional model used in this specification to describe the behaviour of the Source and Target nodes is shown in Figure 2 (without OC-SIP-specific functions)**.**

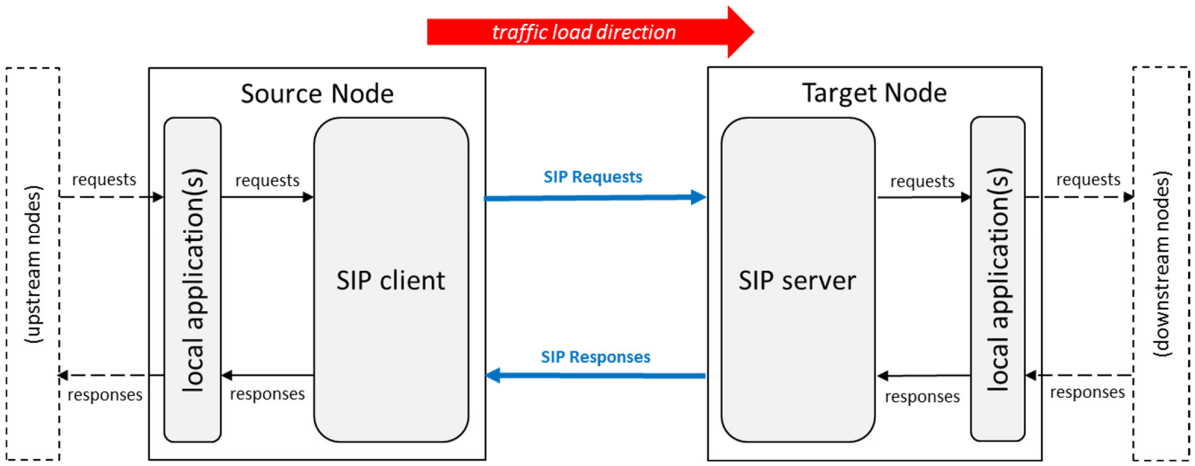

Figure 2: SIP-NNI Functional Elements decomposition, without OC-SIP

The Source Node contains a SIP client function, which sends SIP Request messages to a SIP server function located in the Target node. The SIP server function located in the Target Node generates SIP response messages back to the SIP client function in the Source Node.

Unless otherwise stated, the request rate refers to the rate of requests that are *non-exempt* from restriction (see §8.1).

6.2.2 Source node SIP client function

The Source Node contains a SIP client function which:

- Generates SIP Request messages as a result of requests received from local applications within the Source Node. The local applications may either generate such requests themselves or do so as a result of requests received from an upstream node. Requests from an upstream node may be either SIP signalling or some other signalling protocol e.g. ISUP signalling (in which case, the corresponding local application would typically be an ISUP to SIP interworking function)
- Receives SIP response messages (from the Target Node) and consequently sends responses to local applications within the Source Node, which may either consume such responses themselves or use them to generate responses destined for an upstream node, which may be either SIP signalling or some other signalling protocol e.g. ISUP signalling

6.2.3 Target node SIP server function

The Target Node contains a SIP server function which:

- Receives SIP request messages (from the Source Node) and consequently sends requests to local applications within the Target Node. The local applications may either consume such requests themselves or use them to generate requests destined for a downstream node. Requests destined for a downstream node may be either SIP signalling or some other signalling protocol e.g. ISUP signalling (in which case, the corresponding local application would typically be a SIP to ISUP interworking function)
- Sends SIP response messages (back towards the Source Node) as a result of responses received from local applications within the Target Node, which in turn may either generate such responses themselves or do so as a result of responses received from a downstream node. Responses from a downstream node may be either SIP signalling or some other signalling protocol e.g. ISUP signalling

6.3 OC-SIP functional model

6.3.1 Overview

Figure 3 shows the addition of OC-SIP functions to the basic functional model.

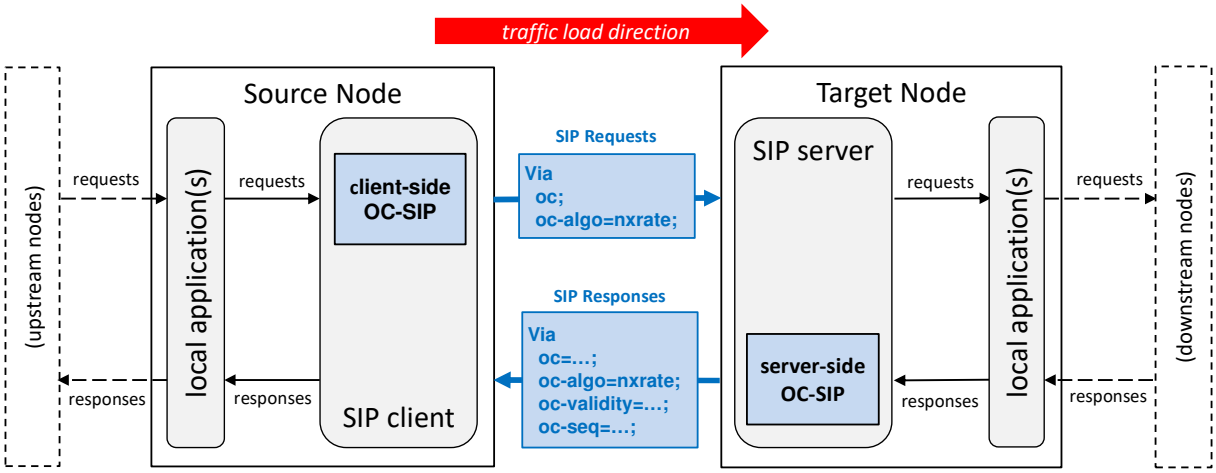

Figure 3: SIP-NNI Functional Elements decomposition, with OC-SIP

The server-side OC-SIP function (located in the Target Node SIP server function) is responsible for notifying the Source Node(s) of the maximum request rate admissible from them. The maximum request rate information is conveyed in parameters carried in the Via header of SIP response messages.

The client-side OC-SIP function (located in the Source Node SIP client function) is responsible for controlling the rate of SIP traffic (i.e. SIP Request messages) sent to the Target Node in order to conform to the maximum request rate that has been notified to it by the Target Node. The Source Node indicates its support for SIP overload control through parameters that are carried in the Via header of SIP Request messages.

6.3.2 Server-side OC-SIP function

6.3.2.1 Overview

The functional architecture of the server-side OC-SIP function is shown in Figure 4.

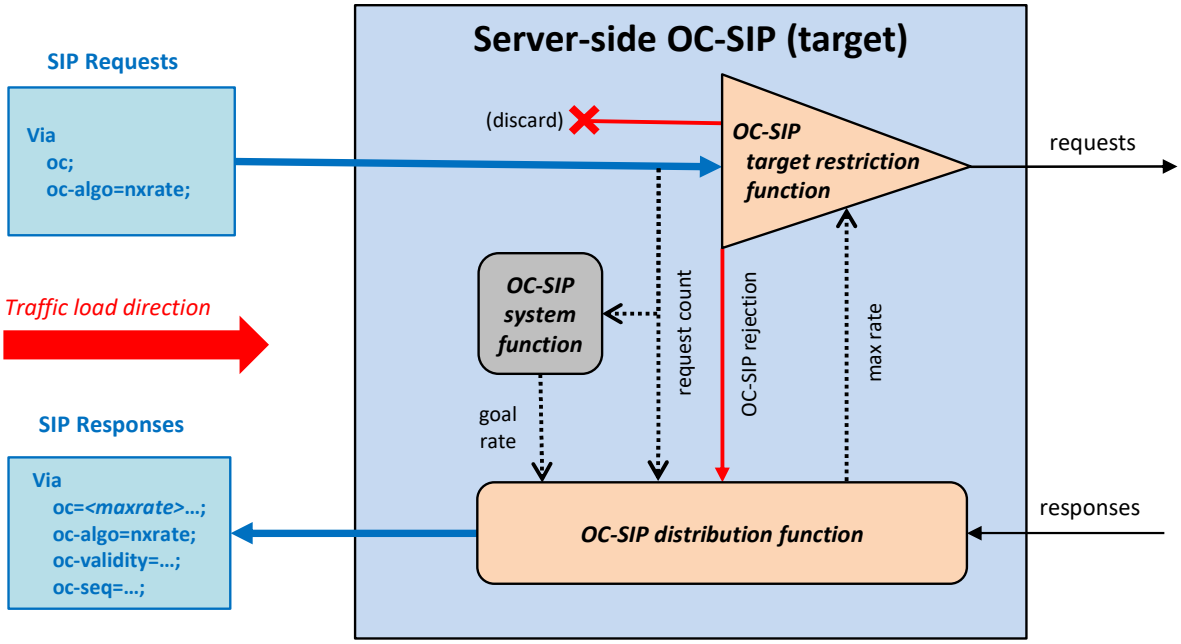

Figure 4: Server-side OC-SIP functional architecture

The main purpose of the server-side OC-SIP is to determine a goal rate for SIP requests arriving at that node and to notify each Source Node of the maximum request rate admissible.

6.3.2.2 OC-SIP-system function

The OC-SIP-system function determines the desired overall goal rate for SIP request arrival at the Target Node. How the OC-SIP-system function determines the Target Node's goal rate is beyond the normative scope of this document but guidance on possible approaches is given in §B.5 of Annex B (informative). It may include taking into account various system metrics associated with the Target Node, such as resource utilisation and request arrival rate.

6.3.2.3 OC-SIP-distribution function

OC-SIP-distribution function apportions the overall goal rate (as determined by the OC-SIP-system function) between sources, to derive a maximum admissible request rate for each Source Node (see §8.4). This rate is notified to the respective Source Node by including the OC-SIP parameters in the Via header of all response messages that are sent to that Source Node.

Where a logical SIP server node consists of more than one SIP server process sharing a single SIP signalling address, it is important that the OC-SIP system function in the SIP server is aware of the SIP request rate across all of the component SIP server processes, in order that it may derive goal request rates for the whole logical SIP server.

6.3.2.4 OC-SIP-target restriction function

At the server side the OC-SIP-target restriction function restricts the admission rate (where necessary) from Source Nodes which cannot or do not sufficiently restrict the rate requested by the Target at the client side. The OC-SIP-target restriction function, which is therefore a proxy source

restriction function at the target, is applied to all traffic from each and every Source Node, and therefore may refuse to admit some traffic from a source node that is non-compliant or non-conforming (see §3.1 - Definitions).

The OC-SIP-target restriction function is intended to ensure that non-compliant or non-conforming nodes do not receive an undue share of available resources when conforming Source Nodes are being subject to rate restriction (see further detail in §13). It does so either by rejecting or discarding (as determined by local policy according to target restriction control parameters – see §13.1 and §B.4 of Annex B) excess traffic arriving from each non-compliant/non-conforming Source Node using its associated maximum admissible request rate as determined by the OC-SIP-distribution function. That is

- Where the OC-SIP-target restriction function determines that a SIP Request message is to be rejected, an appropriate SIP response is generated and sent back to the Source Node
- If a SIP Request message is to be discarded, the SIP Request message is simply dropped and no SIP response is generated

6.3.3 Client-side OC-SIP function

6.3.3.1 Overview

The functional architecture of the client-side OC-SIP function is shown in Figure 5.

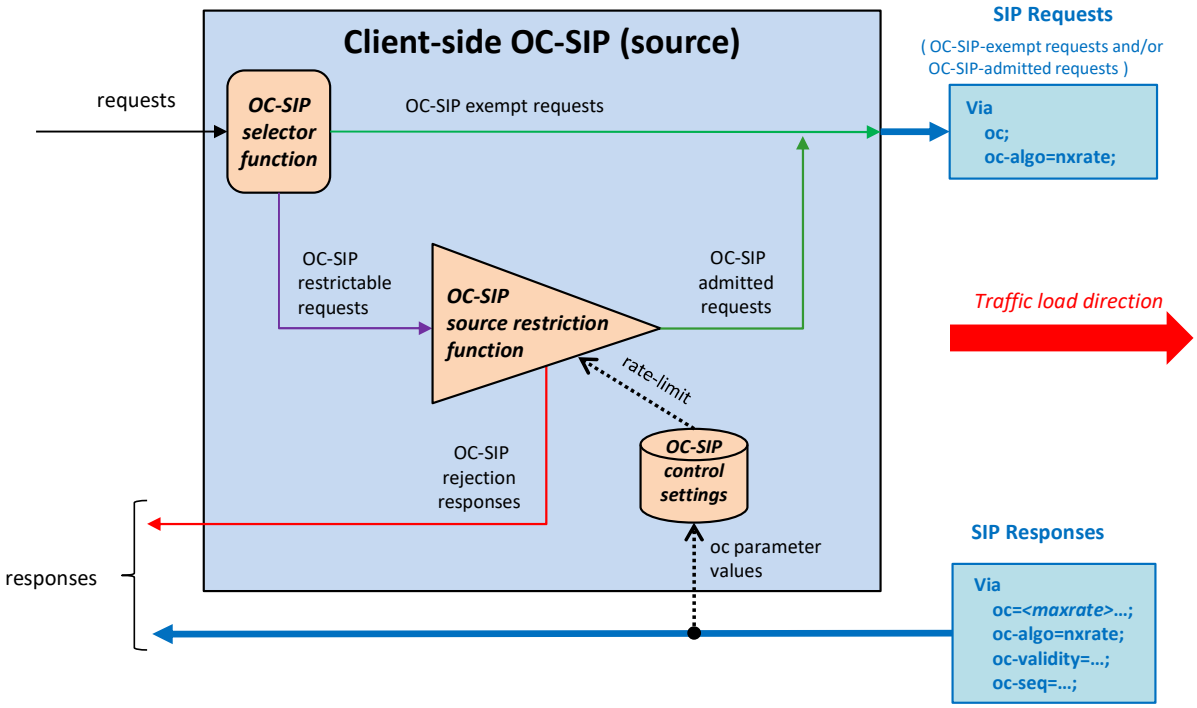

Figure 5: Client-side OC-SIP functional architecture

The main purpose of the client-side OC-SIP is to apply the restriction necessary to ensure that traffic sent towards the Target Node conforms to the maximum SIP request rate which the Target Node has indicated.

6.3.3.2 OC-SIP-selector function

The OC-SIP-selector function classifies any request arriving at the client-side OC-SIP function as either OC-SIP-exempt (see §8.1) or OC-SIP-restrictable:

- An OC-SIP-exempt request is sent to the Target Node, bypassing the OC-SIP-source restriction function (§6.3.3.3). Any request sent to the Target Node must include the required OC-SIP parameters in the Via header
- An OC-SIP-restrictable request is subject to processing by the OC-SIP-source restriction function (§6.3.3.3)

6.3.3.3 OC-SIP-source restriction function

The OC-SIP-source restriction function applies a restriction algorithm to restrict the rate of (OC-SIP-restrictable) SIP Requests sent towards a Target Node to conform to the current admissible rate for that Target Node, subject to the priorities assigned to restrictable requests (§8.2, §8.3). Consequently

- A request that is admitted is sent to the Target Node. Any request sent to the Target Node must include the required OC-SIP parameters in the Via header
- A request that is not admitted by the client-side OC-SIP function is rejected, notifying the local SIP client function. It is then the responsibility of this function and/or the local application within the Source Node to determine what subsequent action is to be taken (see also §11)
- Where the logical Source Node SIP client function is implemented across more than one SIP client process sharing a single IP address, the OC-SIP Source Restriction Function shall conform to the current admissible rate for the whole logical SIP client function. That is, the cumulative rate of SIP requests from such a logical SIP Source Node client (with single IP address) to a single target SIP server does not exceed the maximum SIP request rate which the Target Node has indicated

6.4 Implementing OC-SIP in virtualised networks

The functional behaviour required for OC-SIP compliance is independent of particular network architecture and implementation options. In particular, where functions are implemented on a virtualised server infrastructure their logical behaviour shall be as specified in this document. In order to achieve this, it may be necessary to consider issues such as resource contention between different virtual applications and appropriate system metrics to determine goal rates. These considerations are beyond the scope of this specification.

7 Restriction algorithm

Requirement:

Clients shall implement a maximum admission rate type of algorithm, as required by RFC7415 [3], which shall also be able to differentiate between the number of priority levels used as defined in §8.2 (Priority of restrictable requests), 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 rate.

The default leaky bucket algorithm in §3.5.1 of RFC7415 [3] with the additional features in §3.5.2 (for priorities) should be used, where the number of 'tolerance' reject thresholds τ required will be equal to the number of priorities used according to §8.2. The features of §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 (see §B.1 of Annex B).

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.

8 Restriction Policy

This section concerns the following functions:

- The priority given to a request when deciding whether it should be admitted, including the exemption of certain requests from restriction as enforced by the OC-SIP-selector function (§6.3.3.2)
- How the *oc* value (control rate) [3] should be determined by the target node

8.1 Restriction exempt requests

Since a SIP response to an ACK is disallowed, if an ACK were not admitted by an OC-SIP restriction then it 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 the above reasons these request methods are critical to the performant behaviour of SIP. Furthermore the rate of these request methods is controlled indirectly by the rate of sending dialogue-initiating INVITEs (§B.2.1).

Requirement: The following request methods are exempt from restriction i.e. they shall always be admitted by overload control:

ACK

BYE

CANCEL

PRACK

except when the OC-SIP *target* restriction function's restrictor is in the extreme discard state (see §13).

All other requests may be rejected by overload control and are termed *restrictable*.

8.2 Priority of restrictable requests

Although emergency calls are required to be given the highest priority, it is possible that emergency traffic alone could be sufficient to cause overload, therefore they cannot be made exempt from restriction.

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 (§B.2.2).

Requirement: The following principles shall be applied to determine the relative priority levels of restrictable requests:

- Requests within a dialogue shall be given higher priority than those that are not within a dialogue
- Requests associated with emergency calls (§12) shall be given a higher priority than any other restrictable request

8.3 Default restriction priority levels

Table 1, below, shows the detailed assignment of priority values to requests which results from the application of the principles specified above (§8.1 & §8.2), using a convention whereby numerical priority values are in inverse order to the priority, so that 0 is restriction exempt, 1 is for emergency calls, etc. This represents a general scheme which is extensible to include additional methods and may be used in applications not limited to SIP NNI. Table 1 specifies relative priorities for the current set of SIP methods, not all of which are necessarily applicable to the UK SIP NNI.

Requirement:

For all requests that are specified explicitly in ND1035 [2] (including its optional Annexes) the priority assignments specified in Table 1 *shall* be used at the UK SIP NNI.

For all requests that are NOT specified in ND1035 [2], by default, the priority assignments specified in Table 1 *should* be used at the UK SIP NNI. Any divergence from the specified priorities should be by joint agreement between the network operators involved and based on careful analysis of the traffic characteristics and impact of the particular application under consideration. The principles specified in §8.2 above shall apply.

Table 1, broken down by Request Method, shows that:

- ACK, BYE, CANCEL and PRACK are all exempt (priority=0) from OC-SIP-restrictions (§8.1)
- The highest priority (=1) [i.e. those that are the last candidates for any restriction to be applied to] are methods (be they either in-dialogue or out-of-dialogue) which are associated with an ‘emergency call’ (§8.2)
- The next highest priority (=2) are in-dialogue methods (e.g. related to ongoing calls) that are not associated with emergency calls (§8.2)
- The next highest priority (=3) are out-of-dialogue methods other than INVITE or REGISTER that are not associated with emergency calls (§8.2)
- The lowest priority (=4) [i.e. those that would be the first candidates for any restriction to be applied to] are out-of-dialogue INVITE or REGISTER methods (e.g. new calls) that are not associated with emergency calls (§8.2)

Request Method	Exempt?	Within Dialogue?	Emergency call	Priority Level
ACK	yes	yes	no	0
			yes	0
BYE	yes	yes	no	0
			yes	0
CANCEL	yes	yes	no	0
			yes	0
PRACK	yes	yes	no	0
			yes	0
INFO	no	yes	no	2
			yes	1
INVITE	no	no	no	4
			yes	1
		yes	no	2
			yes	1
MESSAGE	no	no	no	3
			yes	1
		yes	no	2
			yes	1
NOTIFY	no	yes	no	2
			yes	1
OPTIONS	no	no	no	3
			yes[Note 1]	1
		yes	no	2
			yes	1
PUBLISH	no	no	no	3
			yes	1
REFER	no	no	no	3
			yes	1
REGISTER	no	no	no	4
			yes	1
SUBSCRIBE	no	no	no	3
			yes	1
		yes	no	2
			yes	1
UPDATE	no	yes	no	2
			yes	1

Table 1: Default OC-SIP restriction priority levels and their criteria

Note 1 An out-of-dialogue OPTIONS message cannot strictly be deemed an ‘emergency call’. However, although it is not a ‘call’, the OPTIONS message could still potentially be treated as an ‘emergency’ message if the request can be identified as being of an ‘emergency’ type e.g. if it has the Resource-Priority header associated with it.

8.4 Control rate (*oc* value) allocation over sources

Whilst RFC7415 [3] provides the mechanism to distribute levels of control (maximum admission rates), by means of the *oc* values from the overloaded target to compliant sources, it does not specify how those maximum rate values should be derived. This function is critical since it must be

done in a way that meets key objectives (see §8.4.2 below) which ensure that overload control is effective.

8.4.1 Deriving the goal rate for the target of overload

At any given moment in time the maximum rate at which the target can sustainably process SIP requests whilst meeting performance requirements will be termed the *goal* rate. This will usually be a dynamic value which is changed over time, but might be static for simple systems. In any case, it will depend upon the target system architecture, including factors such as resources, process structure and scheduling priorities, and non-SIP workload such as management functions.

In view of this the mechanism for deriving the goal rate will not be specified by NICC, although guiding principles for doing this are given (§B.5).

8.4.2 Key objectives

Requirement: The overload control must satisfy the following two objectives.

Objective 1 Total rate received from all source nodes

The primary objective of control is to prevent the target from overload and to maximise the rate received by the target, subject to the goal rate bound, i.e. when the total source rate

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

This 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.

Objective 2 Allocation of rates over source nodes

The secondary objective is to 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 previously agreed SLA between CPs.

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:

Example 1 Depending upon the total magnitude of the demand from all sources, it may be possible to satisfy Objective 1 in a way so that some source nodes are denied any service (given an *oc* rate of 0) at all. Furthermore, such an extreme distribution could vary over time, i.e. the sources denied any service may switch from some source nodes to others, perhaps randomly.

Example 2 The overload of a target may be dominated by traffic from just a few sources (e.g. from a specific CP), 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.

8.4.3 Design for control rate allocation and distribution

Requirement: To satisfy the objectives in §8.4.2 the solution design shall structure the *oc* rate allocated to each source as a sum of components that depend on:

- A control variable that is adapted to meet Objective 1

- SLA parameters that can be configured to give the differentiated levels of service as required by Objective 2

The structure of the source rate is defined in §A.1.1.3 of Annex A (normative)

The method of adapting the control variable is specified in §A.1.2 of Annex A.

The SLA parameters, defined in §A.1.1 of Annex A, have the following characteristics:

- Minimum rates for each source, which are 'guaranteed' in normal circumstances, independent of the traffic originating from other sources
- Enable unused 'spare' rates from sources originating traffic below their minimum to be used by other sources, in a declared and predictable way

9 Load-sharing and re-routeing

When routeing calls at a SIP node, load-sharing over several target servers is generally used for the next hop. Since load-sharing is normally static, i.e. initial requests are delivered in appropriate but static proportions without using the load state of each node, each of these target/servers may be subject to overload independently of each other since their actual SIP resource load state is not necessarily the same.

Requirement: Load-sharing shall be done prior to source/client restriction in order that the control parameter values are specific only to each target from which they have been received.

In addition, appropriate limited alternative routeing may be used on rejection by client restriction due to overload control, re-trying another of the same load-sharing group, or another route altogether. These re-routeing attempts must also be subject to any restriction due to overload against the chosen next node. In summary:

Requirement: All routeing decisions, whether by load-sharing or alternative choices, shall subsequently go through any overload control before being sent to a target server.

10 Resilience, failover, and control parameter values

In order to provide resilience to failure, usually each active target SIP server also has a standby. The following different ways in which such resilience may be realised have implications for overload control:

- Each active and standby pair of target SIP servers may share an IP address/ports, or they may have distinct IP addresses/ports
- The active SIP server may or may not share overload control state with the standby SIP server

During the period following failover the load on the newly active target server should stabilise. The time this 'expected duration of failover stabilisation' takes will depend on several factors, including (see further §B.3 of Annex B):

- the method of target server failure detection used by the sources
- whether or not control state is shared between active and standby
- whether overload control is active before or after the failover (or both)

From an overload control perspective, control being inactive before but activated afterward failover is equivalent to control arising for a single target (the activated standby) and so this scenario does not require any special consideration. But overload control being active just before failover has implications for certain control parameter values.

10.1 Minimum *oc-validity* value

Requirement: A positive value of the *oc-validity* parameter [4,§4.3 & 3,§3.2] returned by the target shall be distributed uniformly (which may be random) over a range between the following minimum:

$$2 \times (\text{time between } oc \text{ value updates}) + (\text{expected duration of failover stabilisation})$$

and a maximum that should be by default

$$3 \times (\text{time between } oc \text{ value updates}) + (\text{expected duration of failover stabilisation})$$

The default *oc-validity* applied by a source/client should be of a similar magnitude and may be obtained by consultation with connecting CPs operating target servers.

This is necessary because overload control may be active before a target server fails and this ensures that control does not terminate at the sources during the failover process, as illustrated in Figure 13 of §B.3.1 in Annex B.

Note:

- the time between *oc* value updates may not be constant
- a value of *oc-validity* = 0 to terminate control
- RFC7339 [4,§4.3] specifies that a default value of 500ms should be applied by the source/client when an *oc-validity* value has not been received. This will in general be too small.

10.2 Relationship of source restrictor to target and its standby

Requirement: A target shall ensure that source rate control persists during target failover in one (or more) of the following ways:

- share a common IP address with its standby, setting the *oc-seq* values [4,§4.4 & 3,§3.2] as per §10.3
- share overload control state with its standby
- using another mechanism agreed between relevant CPs

For purposes of overload control, a distinct target SIP server is identified (to a source SIP server) by its IP address and port number. Thus, at the source, a restrictor relating to a particular target is associated with a single address/port identifying that target.

If a target server and its standby share a common address/port then a single restrictor at the source will necessarily be associated with the target server **and** its standby. Note that there is no mechanism defined by which a source can discover whether this is the case.

If a target server and its standby do **not** share a common address/port but do 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 mechanism defined by which a source can discover whether this is the case.

If, by means of administrative arrangements which are outside the scope of this specification, a source knows that a target server and its standby do **not** have a shared (common) IP address/port **and** do **not** share overload state then it may take additional measures to mitigate the effect of target failover during overload e.g. by associating a single restrictor with both IP addresses.

The above scenarios are explained in more detail in §B.3.3 of Annex B.

10.3 *oc-seq* value

During normal operation the *oc-seq* value, which is included in all responses, is increased by a server according to RFC7339 [4,§4.4]. This is qualified by the following.

Requirements: The oc-seq value:

- shall 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
- should not be increased when there has not been a control update

The first of these is necessary to cause the oc-validity timer to be restarted (as per §5.4 of [4]). The second prevents 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. It requires setting the *oc-validity* duration sufficiently long, according to §10.1.

To account for target server failover the following special behaviour is required in order to ensure that the responses do not terminate any already active control at the sources (see §B.3.2 of Annex B).

Requirement: If a target server does not share overload control state with its standby, then during the expected duration of stabilisation when the newly activated target is not in an overload state (*oc-validity* =0 sent), the value of the *oc-seq* parameter sent in responses must be lower than the value sent before failover.

If the *oc-seq* is normally derived from the current time, then the above can be achieved by the activated standby deriving the *oc-seq* as it becomes activated (on restart or failover) from the following expression for adjusted time

(activation time) – (maximum configured server non-zero *oc-validity* value)

The server must revert to deriving *oc-seq* from the current time at the start of overload control following activation of the standby, when *oc-validity* > 0 for the first time.

11 Response codes for failure on rejection by control

11.1 Target node

Requirement: A target node which rejects a request as a result of overload shall do so using a SIP 503 (Service Unavailable) response.

Note: A 503 response generated as a result of an overload condition is not distinguishable from a 503 response generated by any other failure condition and so cannot be subject to any OC-SIP-specific handling at an upstream server which receives it.

11.2 Source node

A source node may have one or more possible choices of egress route on which to forward a request (selected according to local policy).

Requirement: A source node which rejects a request as a result of applying a OC-SIP restriction on the final choice of egress route shall do so using a SIP 503 (Service Unavailable) response (in the case of incoming SIP request); or an equivalent cause indication, where interworking to another protocol applies.

Note: When a request cannot be forwarded on a particular route because it is not admitted by a OC-SIP restriction applied to that route, the node may choose to attempt to forward the request via alternative routes (if available). If such a request subsequently fails, the response code returned will depend upon the final reason for failure and will not reflect the previous rejection by the OC-SIP mechanism.

12 Emergency traffic

Identification of SIP requests that are associated with emergency traffic has been defined by NICC in ND1035 [3, §A.1]. Note that this may differ from the identification given by §5.10.1 of RFC7339 [4], also implied by RFC7415 [3].

Requirement: SIP requests that are associated with emergencies shall be given high restriction priority. The specific priority depends upon certain criteria:

- Certain in-dialogue emergency requests are restriction exempt, as they are for non-emergency traffic (§8.1)
- All other emergency requests are restrictable but are given the highest priority (§8.2)

13 Non-compliant or non-conforming sources

A *non-compliant source* node and a *non-conforming source* node have already been defined in §3.1.

Requirement: When a target node activates overload control, for each connected source node it shall allocate an instance of the OC-SIP target restriction function (see Figure 4) as described in §13.1 below. Every request from the source shall 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 (only if non-exempt), or discarded (whether exempt requests may be discarded is dependent upon precise implementation of the algorithm, see §13.1).

Note that such a OC-SIP target restrictor is allocated for *all* source nodes to which the target is connected, including those thought to be compliant. In this way if such a node turns out to be non-conforming it will still be controlled to the required maximum rate.

An alternative whereby a source can apply static rate restriction towards the target is not acceptable since it would give preferential treatment to non-compliant sources and would not give adequate resilience for the target since the rate could not be adapted.

13.1 Rate control algorithm for the OC-SIP target restriction function

Requirement: The OC-SIP target restriction function should use the default algorithm defined in RFC7415 (with multiple priorities) as per §7, in which case it shall be with the following additional features:

- An additional bucket 'discard' threshold τ^* is introduced that shall be greater than any other threshold, and is the maximum possible fill of the bucket. Whenever a request is received and the fill is greater than τ^*, then the request shall be *discarded*, i.e. ignored without sending a response. All request types may be discarded, possibly including exempt requests (*Note 1* below)
- The bucket fill is now not just increased when admitting a request but also when rejecting a request (but not when discarding). The amount the fill is increased on rejection will be system

dependent, so it is not specified here. Some possibilities are outlined in §B.4.3. The fill is not increased when discarding a request

As per §7, 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.

The original default algorithm of RFC7415 [3], which is used for the OC-SIP source restriction function, is in fact a special case of the enhanced algorithm, obtained simply by setting a 0 cost of rejection. This is because the *discard* threshold τ^*, which must be larger than any *reject* threshold, will have no effect since the fill cannot increase beyond the highest reject threshold.

Note 1: 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 be* admitted (even if it then isn't), in which case the fill cannot be above the threshold for that priority following admission (although it can on rejection). For the latter variant the maximum fill of the bucket is the discard threshold, whereas for the former it will be the discard threshold plus the maximum increase on rejection. Since exempt requests cannot be rejected they incur no increase in fill and so for the former variant where the fill always remains below the discard they would therefore never be discarded.

14 Security

Ways in which control could be modified maliciously in order to deny service are described in RFC7339 Session Initiated Protocol Overload [4], also referenced by RFC7415 Session Initiated Protocol Rate Control [3].

Securing SIP in order to prevent such infiltration is described in ND1647 SIP-NNI Basic Voice Architecture [1].

15 Interworking with other signalled overload control schemes

15.1 Interworking and co-existence with non-ND1653 OC-SIP implementations

Interworking with implementations of SIP overload control according to RFC7415 or 7339 that do not comply with ND1653 is outside the scope of this specification. However, it is recognised that, in practice, a SIP node may support more than one variant of a SIP overload control mechanism and that ND1653 may co-exist in a mixed network environment with other variants. The following remarks are intended to provide guidance to minimise potential difficulties in such circumstances.

RFC7339 and RFC7415 have been designed so that the restriction methods supported and that preferred, by a source (client) and target (server) node, can be mutually discovered by signalling dialogue alone, using the transfer of the *oc-algo* list. By adding the new token "nxrate" ND1653 capability can be differentiated from the default methods defined in the RFCs. It also avoids the need to provision nodes with the capability of peers, and prevents inconsistencies that could arise from configuration error.

15.1.1 ND1653 client

Table 2 shows possible *oc-algo* list combinations which an ND1653 client is / is not permitted to send under various conditions of interworking/co-existence, with guidance on why particular combinations are / are not permitted.

***oc-algo* value (token list) sent**	**ND1653 compliant?**	**Comments**
"nxrate" *Note 2*	Yes	Client only supports "nxrate" restriction scheme. *Note 1*
<other values including "nxrate"> *Note 2, Note 3*	Yes	Client supports multiple restriction schemes including "nxrate". *Note 1*
<other values excluding "nxrate"> *Note 3*	No: "nxrate" must appear.	Outside the scope of ND1653.

Table 2: ND1653 client interworking and co-existence with RFC7415 and RFC7339

Note 1: A non-ND1653 server should be able to accept and ignore the unrecognised value "nxrate" without problems. A well-designed server overload control should then be able to apply an appropriate local restriction scheme like that specified for ND1653 servers with non-compliant sources.

Note 2: This does not strictly comply with RFC7339 nor RFC7415 because the *oc-algo* token "nxrate" is not defined in the RFCs. However the signalling framework is compliant since RFC7339 refers to other methods in the *oc-algo* list (§4.2 of RFC7339).

Note 3: To be compliant with RFC7339 the list of <other values> must include "loss" (defined in the RFC) and may include other values. To be compliant with RFC7415 the list must include "rate" and may or may not also include "loss" (unclear from RFC7415).

15.1.2 ND1653 server

Table 3 shows the permitted handling of possible *oc-algo* list combinations which an ND1653 server may receive under various conditions of interworking/co-existence (the behaviour of a non-ND1653 server is out of scope for this specification).

***oc-algo* value (token list) received**	***oc-algo* value response**	**Comments**
"nxrate"	"nxrate"	ND1653 server will always select "nxrate" if present.
<other values including "nxrate">	"nxrate"	
<other values excluding "nxrate">	none	Treat as a non-compliant source as per ND1653.

Table 3: ND1653 server interworking and co-existence with RFC7415 and RFC7339

15.2 Interworking with ISUP/BICC ACC

15.2.1 Signalling interworking

The OC-SIP mechanism specified by this document is a hop-by-hop method, hence the scope of the OC-SIP signalling is limited to the interface between the target node (SIP server) and source node (SIP client). There is no requirement for any direct interworking between the OC-SIP protocol and any other overload control protocol (such as ISUP ACC [i4,i5] or BICC ACC [i6]).

Where a node has ISUP/BICC interfaces supporting ACC (in addition to SIP interfaces supporting OC-SIP), the overload control protocols operate independently on each interface (possibly subject to coordination by higher-level node functions which are beyond the scope of this specification).

15.2.2 Target node with SIP and ISUP/BICC

A target node which has ISUP/BICC traffic sources (in addition to SIP sources) may need to take account of the traffic load associated with these sources in determining the goal rate and the target rates to be allocated to the various sources.

15.2.3 Source node with SIP and ISUP/BICC

15.2.3.1 Source node with ISUP/BICC ingress

Requirement: A source node which has ISUP/BICC interfaces (in addition to SIP interfaces) shall apply any required OC-SIP restriction for a given SIP egress route in the normal way regardless of the ingress signalling type (i.e. ISUP/BICC or SIP) from which requests have originated.

Refer to §11.2 for behaviour on rejection by OC-SIP restriction and §9 for alternative routeing behaviour.

15.2.3.2 Source node with ISUP/BICC egress

A source node which is unable to forward a SIP request on a particular route because it is not admitted by a OC-SIP restriction applied to that route, may choose (according to local policy) to attempt to forward the request via alternative routes (if available) which may include ISUP/BICC egress routes.

Annex A (normative)

A.1 Control rate allocation and adaptation

The target system under SIP load is assumed to derive the following from measurements, at intervals of time:

- A *goal* rate, which is the maximum expected request rate such that response time requirements will just be satisfied
- The arrival rate of requests over the last measurement interval.

How this is done is not prescribed, although guidance is given (§B.5)

The rate to be sent in responses to each source is derived from these values. These source rates are structured in terms of configurable 'SLA parameters' which have values specific to each source, and a common single control variable *X* that is changed when overload control is active (the parameter values should not be changed whilst overload control is in effect because they affect control behaviour). This enables the arrival rate at the target to be adapted and to converge to the goal rate whilst allocating maximum rates to each source in a manner that satisfy source-specific SLAs.

A.1.1 Control rate allocation and distribution over sources

When control adaptation described in §A.1.2 performs an update, it provides a new value of the control variable *X* to be used in updating the values of the *oc* max rate value for each known source. These *oc* rate values also depend upon two 'SLA parameters' configured for each source *i* which determine how much of the available capacity each is allocated. These two parameters (Table 4 below) are a guaranteed rate s_i, and a weight w_i that governs the share of the remaining capacity.

The definition and use of these parameters in their full generality is described below in §A.1.1.1 to §A.1.1.3, but as shown in §A.1.1.8 it is perfectly acceptable (and simpler) to use particular fixed values if such generality isn't required, and still be compliant with this method.

Long name	Short notation	Description
SourceRate	s_i	Guaranteed minimum rate associated with source *i*
SourceWeight	w_i	Weight associated with source *i*

Table 4: Configurable parameters associated with control distribution

Long name	Short notation	Description
SumRates	S	Sum of guaranteed rates s_i over each source i $S = \sum s_i$
SumWeights	W	Sum of weights over each source i $W = \sum w_i$
SourceProportion	p_i	Proportion derived from weight w_i associated with source i: $p_i = \frac{w_i}{W}$
RatioMin	r	Minimum ratio of rate to proportion over each source i $r = \min\left\{\frac{s_i}{p_i}: \quad p_i > 0\right\}$

Table 5: Variables associated with control rate distribution

A.1.1.1 Capacity allocation 'rate guarantees'

Each source id i (which will be associated with an IP address) is given a nominal guaranteed minimum rate allocation s_i . These values should be configured such that the sum S of such rates over all sources (Table 5 above) is less than the normal expected minimum value of the goal rate Γ, i.e. less than the expected capacity available. In order that this is true in all but exceptional circumstances, the configured values should allow for some degree of partial loss of capacity (e.g. single component resource failure).

However, in exceptional circumstances, the available capacity could be lower than expected, and in case the capacity of the system has dropped close to or below the sum of guaranteed rates S, the configured rates would be reduced by multiplying them by a coefficient θ less than 1 (see Table 8, §A.1.2.4), with a value dynamically determined according to how low the goal rate has dropped relative to S. Under normal circumstances the value of θ is 1 and this re-evaluation is triggered according to a configurable parameter which determines how close Γ is to S. This threshold (configurable) parameter $e > 0$ (see Table 7, §A.1.2.4) measures the excess of the ratio Γ/S compared to 1, i.e. $1 + e$. In choosing the coefficient θ we want to enforce

$$\Gamma/\theta S \geq (1+e) > 1$$

Therefore the coefficient is chosen to be

$$\theta = \min\left\{1, \left(\frac{1}{1+e}\right)\frac{\Gamma}{S}\right\}$$

This calculation is done after an adaptation of the control variable X and before it is used by control distribution.

A.1.1.2 Capacity allocation 'weights'

The previous section defines the guaranteed part of the control rate for each source. In order to derive the rates for distribution, distribution 'weights' w_i are also required (Table 4 above) for each source i in addition to the previously configured rate guarantees s_i which define the adaptable part of the rate distributed to each source. Since the proportion p_i derived from the set of weights is

required (which sum to 1) it is convenient to re-compute this whenever a weight value changes or a source is added or deleted, by setting

$$p_i = \frac{w_i}{W}$$

where W is the sum of all the weights over all sources (Table 5 above).

A.1.1.3 *oc* rate for each source

The rate specific to source id i is

$$R_i = \theta s_i + p_i \left(X - \theta S \right)$$

in terms of the control variable, the configurable parameters described in §A.1.1.1 and §A.1.1.2 and listed in Table 4, and the derived variables listed in Table 5.

A.1.1.4 Changing SLA parameter values

The rate sent to each source (§A.1.1.3) depends upon the min rate (§A.1.1.1) and weight (§A.1.1.2) values configured for all sources.

In addition the origin of adaptation depends upon the sum of rates S and the minimum r of the ratios of s_i / p_i over all sources (Table 5).

Therefore, whenever the values of the configured rates or weights change, or sources are added or removed, these values have to be re-evaluated and provided to the adaptation function.

A.1.1.5 Compliant sources

For each source i compliant to SRC the *oc* value is assigned the computed rate R_i and returned to i in the Via header of all responses to i.

To also guard against non-conformance of such a source, the same rate should be applied to a local proxy rate control instance allocated against id i at the target, as per §13.

A.1.1.6 Non-compliant sources

A non-compliant source i should be allocated a local proxy rate control instance allocated against id i at the target, and the computed rate R_i against i applied. As per §13 this rate control is enhanced to incur a cost on rejecting requests and a discard threshold.

A.1.1.7 Special treatment for sources with a distribution weight of 0

If a source has a configured weight of $w_i = 0$ then control must be activated semi-permanently, by which is meant that once configured it is always sent a value of θs_i even when the target node is not in an overload state (usually this will be s_i since under normal circumstances $\theta = 1$). This is because only binary adaptation, i.e. turning on or off, is possible, not ‘continuous’ adaptation. This has the potential to destabilise adaptation because the coefficient in the control variable X is always 0,

thereby the variable becomes impotent and adaptation isn't possible for such a source. The effect of such configuration over *all* sources would mean that control would always over-restrict or under-restrict. Over-restricting would lead to control turning off, and then turning on again in a repeating cycle and such instability must be avoided. Under-restricting would mean that the coefficient θ would have to be reduced below 1 under normal conditions, whereas it is only meant to be used in exceptional circumstances. Assigning non-zero weights to some sources does not always avoid this since such sources may not be generating any traffic during overload, and so the control parameter has no effect. Therefore, if such static rate sources are really required, they should be activated semi-permanently, i.e. not affected by adaptive control (although allowed to be changed manually).

A.1.1.8 Special simple configurations

If the fully general configuration of SLA parameters is not required then it is still acceptable to use a specialised simpler design. Possibly the most useful would be the following:

SLA-0 As above but with 0 minimum – all sources 'best effort' without guaranteed minimum

SLA-1 Allocate minimum rate and use of unused allowance equally over all sources

The system must 'know' the number of sources. SLA-0 is a specialisation of the SLA-1 and doesn't require any configuration of minimum rates. In fact, there is little point in using SLA-1 because it would still require the dynamic coefficient θ to be used in case capacity dropped below that expected, in which case they both exhibit the same behaviour.

The cases where the source weights (proportions) are 0 have to be given special consideration (see §A.1.1.7). These have been included in Table 6, below, that lists a broad range of configuration possibilities.

s_i	p_i	*oc* rate R_i	Effect	Control allocation
$=0$	$=0$	0	No capacity for source *i*. Unstable if allocated dynamically. Could be useful for target to disable all traffic from *i*.	Semi-permanent
>0	$=0$	s_i	Static max rate for source *i*. Unstable if allocated dynamically.	Semi-permanent
$=0$	>0	$p_i(X-S)$	'Best effort' for source *i* – no lower bound.	Dynamic
$=0$	$=\frac{1}{N}$	$=\frac{1}{N}X$	Special case of previous example in which an equal weight is given to every source (*N* is the number of sources).	Dynamic
>0	>0	$s_i+p_i(X-S)$	General case for source *i*: Minimum guarantee and use of 'unused allowance'.	Dynamic
$=\frac{S}{N}$	$=\frac{1}{N}$	$=\frac{S}{N}+\frac{1}{N}(X-S)=\frac{1}{N}X$	Special case of previous example with an equal min rate and equal weight given to every source.	Dynamic

Table 6: Configuration possibilities from the simplest to the most general. The coefficient θ in the sum of rates *S* has been suppressed for clarity (normal value 1), but should still be used

A.1.2 Control behaviour

There are 3 main phases of control behaviour:

- Activation of control (§A.1.2.1)
- Linear adaptation of the control variable (§A.1.2.2)
- Termination of control when the arrival rate is consistently below the goal rate (§A.1.2.3)

A.1.2.1 Activation

Control must activate when the total unrestricted rate from all sources exceeds the goal rate Γ derived by the system under load. The choice of the initial control variable value X is not prescribed, since the options available depend upon what is being measured.

If the individual source rate from every source were being measured before control activation then, in principle, the required initial value of the control variable X can be computed.

Otherwise it can be shown that a value of X equal to the goal rate Γ would produce the lowest admission rate and is therefore the safest (most conservative) initial estimate. Higher values could be used, for example by multiplying this by a coefficient greater than 1. Such a coefficient could be a fixed configurable value or could be derived from the known number of sources actively sending traffic.

A.1.2.2 Linear adaptation

The method of adaptation is linear, using the latest value of the measured arrival rate A, the goal arrival rate Γ derived by the system, and the current value of the control variable X. It works by taking a straight line through the point (X, A) from the origin of adaptation at ($\theta(S\text{-}r)$, 0), and finding the intersection with Γ, illustrated in Figure 6, where the total admitted rate function is shown to be an increasing function of the control variable X.

It can be shown that this function is convex (gradient does not increase as X increases), and the gradient of the straight lines used for adaption are not less than the gradient of the admission rate at $X = \theta S$, which ensures convergence to a unique solution, as long as the admission rate eventually exceeds the goal rate.

The origin of adaptation depends upon the sum of rates S, their coefficient θ (Table 8) and the minimum ratio r (Table 5).

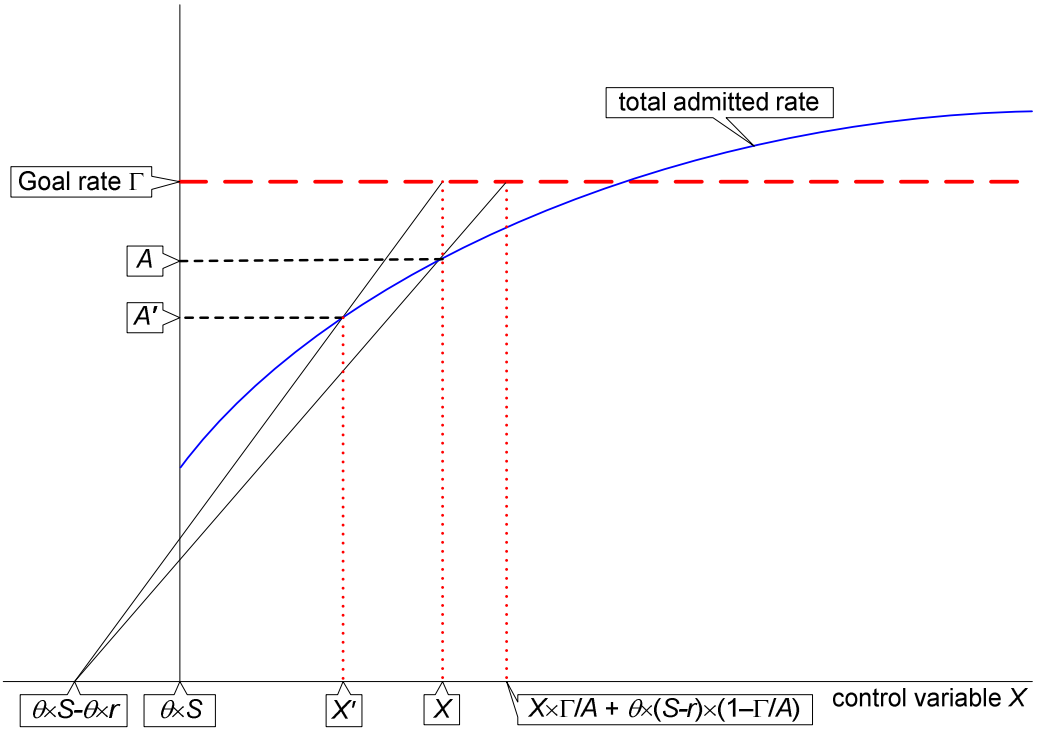

Figure 6: Linear adaptation of the control variable *X*

A.1.2.3 Termination

Control is no longer required once the total unrestricted rate from all sources drops below the goal arrival rate. It is necessary to detect this whilst control is still being applied. It can be seen from Figure 7 that in this state the linearly adapted values of the control parameter X would get larger and larger, whilst increases in the arrival rate would get smaller and smaller, which is undesirable since if there were a sudden surge in load, linear adaptation to reduce the control variable again could take many iterations.

The terminating condition is therefore detected by all of the following being satisfied:

$$A' < \Gamma' \qquad A < \Gamma$$
$$A - A' < \delta \qquad |X - X'| > \Delta$$

where A', Γ' and X' are prior values of the variable, and δ and Δ are configurable (non-zero) parameters (Table 7). The last condition - that changes in X are sufficiently large - is necessary to distinguish the terminating condition from linear convergence to the goal rate from below, where increases in both A and X get smaller and smaller.

In order to ensure that control is not terminated prematurely, when the above conditions are satisfied the system enters a terminating state and sets a termination timer, and the conditions are retested at each system update until the timer expires. In this state the control variable reverts to its previous value at each update in order to keep X bounded. Thus, when staying in the terminating state, the control variable alternates between two values until a timer expires causing control to be turned off. This is the reason that the condition uses the absolute value of $X - X'$ which will alternate in sign under such behaviour.

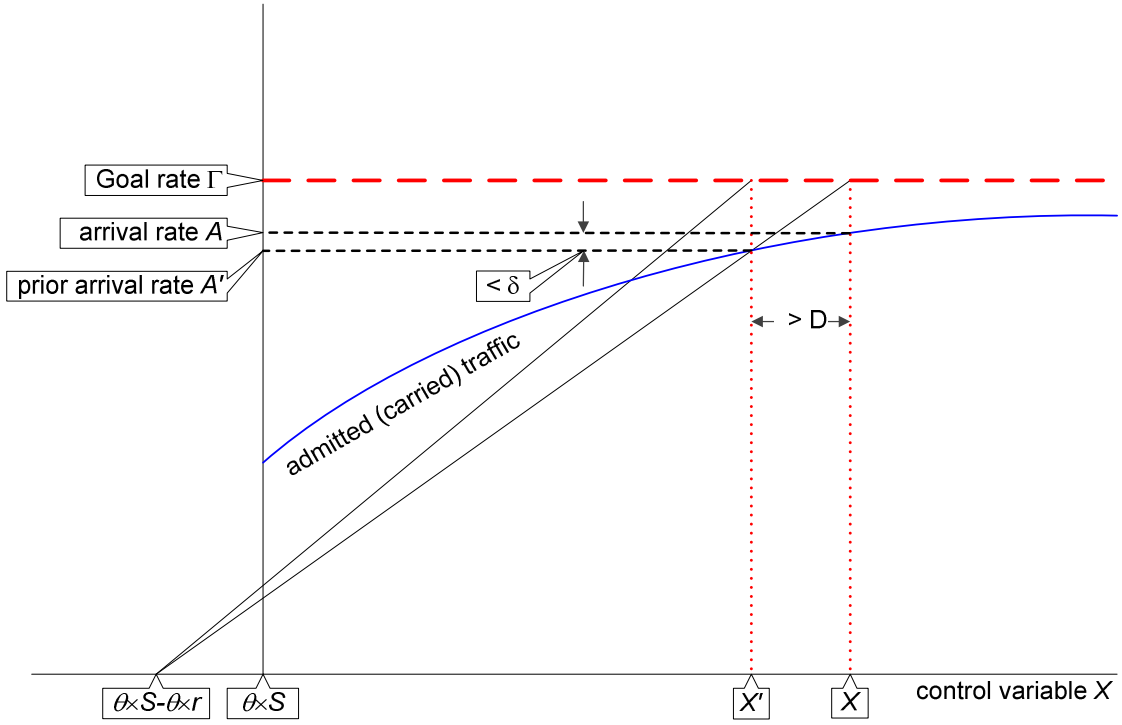

Figure 7: Conditions for adaptation being in a Terminating state: Arrival rate and changes to the arrival rate and control variable

A.1.2.4 Variables and parameters

The configurable parameters and variables used by the target adaptation function are listed below in Table 7 and Table 8 respectively. It is important to have configurable bounds for most of the variables, e.g. the control variable *X*. Similarly the range of parameter values should be restricted. These are not prescribed here.

Long name	Short notation	Description
dArrRate	δ	Upper bound of a sufficiently small increase to the arrival rate that determines transition to the terminating state.
DX	Δ	Lower bound of a sufficiently large change to the control parameter that determines transition to the terminating state.
DurationTP	D_{TP}	Duration for the termination pending timer which is set when entering the terminating state.
	e	Excess in the ratio of the goal rate to the sum of rates S which gives the threshold of when to re-evaluate the coefficient θ in the sum of rates.

Table 7: Configurable parameters held by target adaptation function

Long Name	Short notation	Description	Updated by
ArrRate	A	Total arrival rate of requests over the most recent complete time interval of measurement.	System under load, from dynamic measurement.
priorArrRate	A'	Total arrival rate of requests over the time interval prior to the most recent one.	System under load, from dynamic measurement.
X	X	The control variable.	Adaptation function.
priorX	X'	Prior value of the control variable.	Adaptation function.
tempX	X''	Value of the control variable used to swap X and X'.	Adaptation function.
GoalRate	Γ	Goal arrival rate of requests derived from the most recent time interval of measurement.	System under load, from dynamic measurement.
priorGoalRate	Γ'	Prior value of the goal arrival rate of requests.	System under load, from dynamic measurement.
SumRates	S	See Table 5.	From configured data.
SumRatesCoeff	θ	The coefficient multiplying the sum of rates S which is used to reduce the guaranteed capacity, only if the capacity as measured by the goal rate Γ drops sufficiently close to S, i.e. below $(1+e)S$. See Table 5.	Adaptation function, as a function of S, Γ, and e.
RatioMin	r	See Table 5.	From configured data.

Table 8: Variables held by target adaptation function

A.1.2.5 Diagrams defining the method of adapting the control variable *X*

Specification Description Language (SDL) [i3] is used informally in Figure 9,

Figure 10 and Figure 11 to define in detail the behaviour described in §A.1.2.1, §A.1.2.2, §A.1.2.3.

Figure 8 below shows the control state space defined in the SDL diagrams and the possible transitions between them.

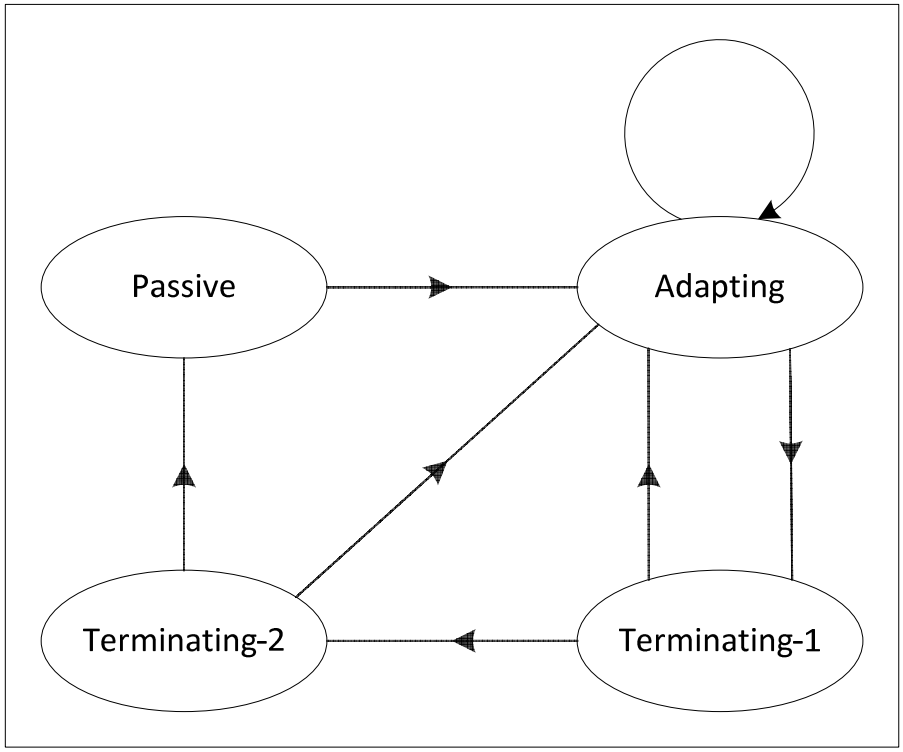

Figure 8: Overload control states and the possible transitions between them

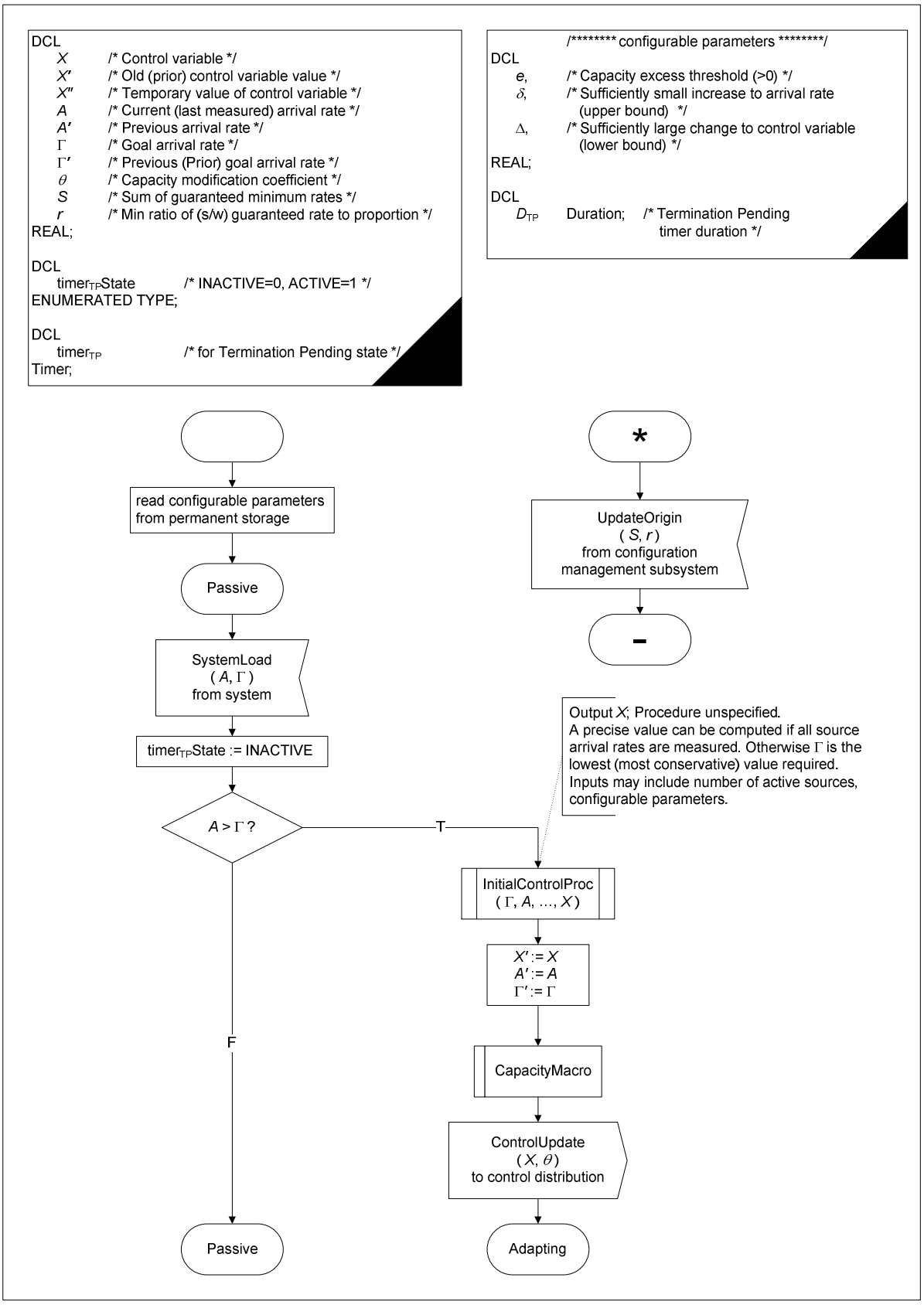

Figure 9: Informal SDL for adaptation function: Declarations and activation

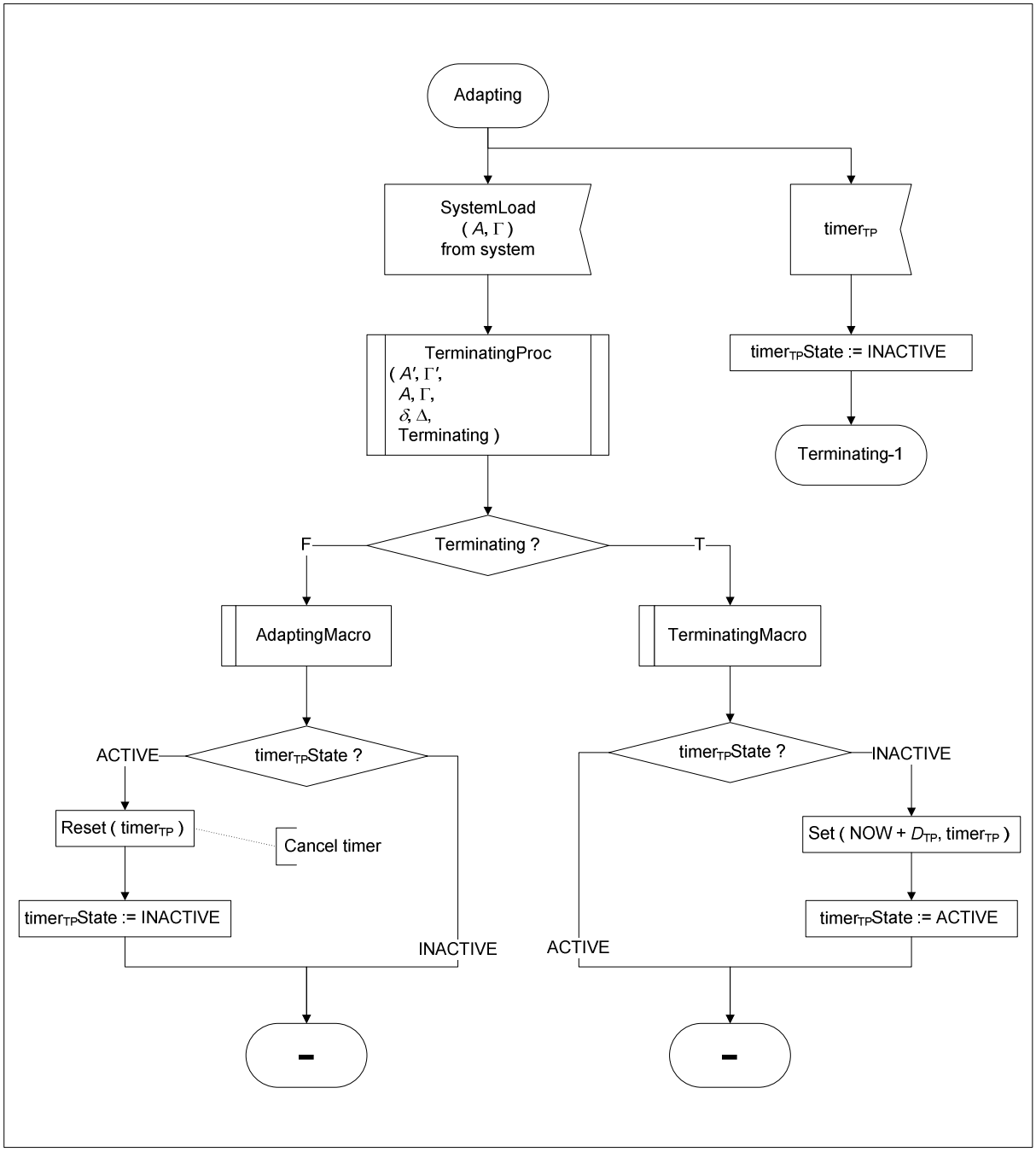

Figure 10: Informal SDL for adaptation function: Adapting state

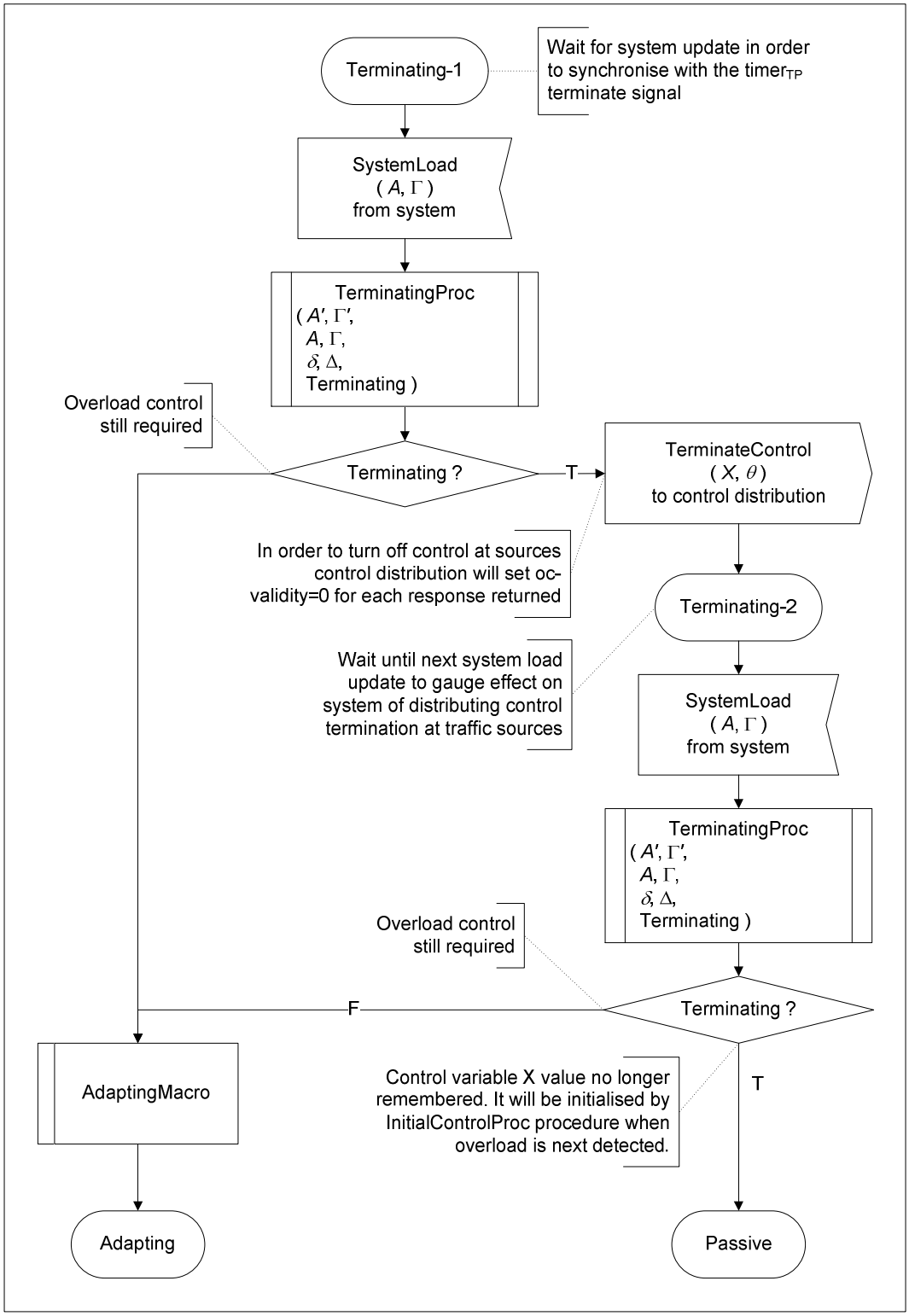

Figure 11: Informal SDL for adaptation function: Terminating states

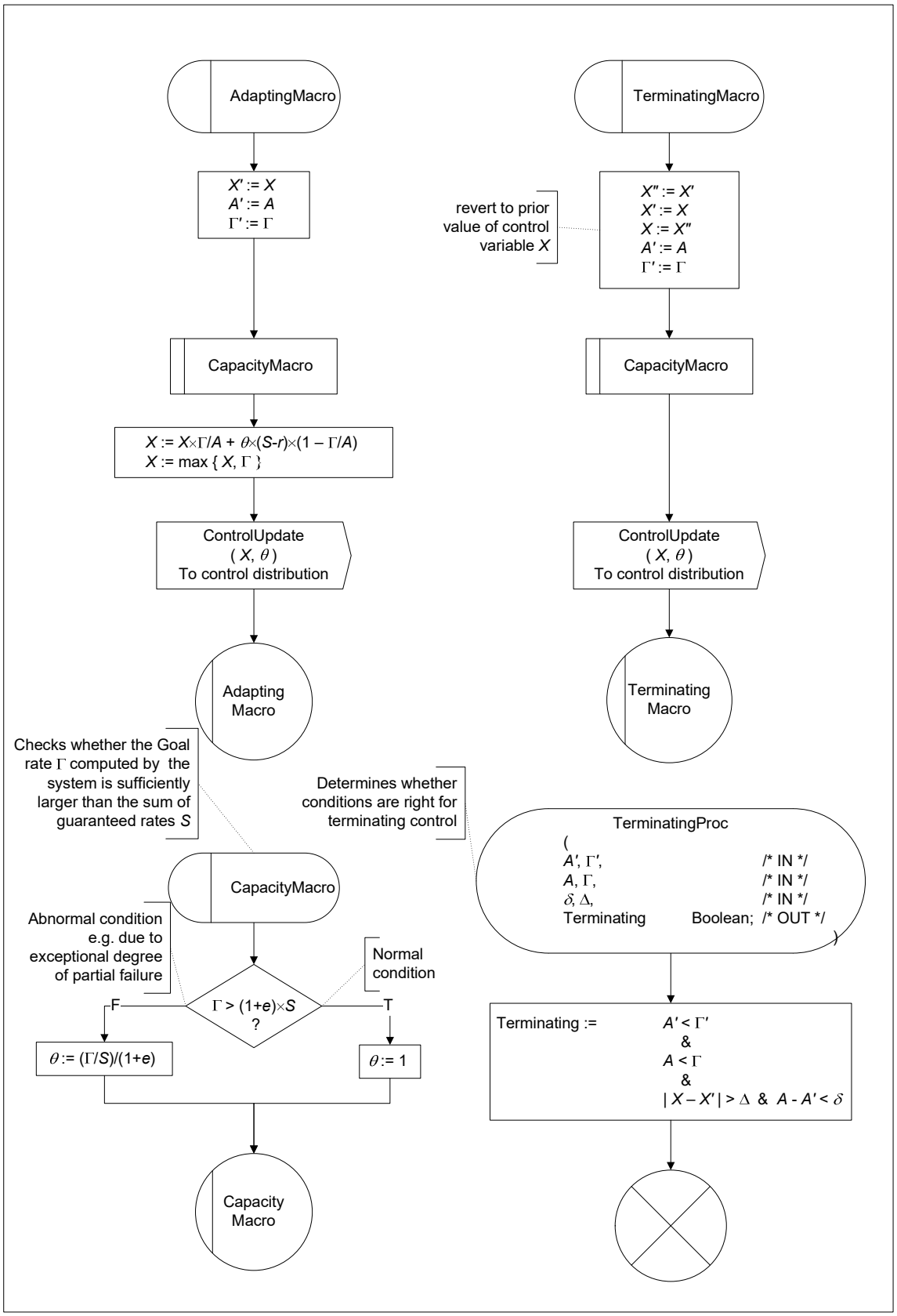

Figure 12: Informal SDL for adaptation function: Macros & Procedure

Annex B (informative)

B.1 Beneficial properties of the default rate control algorithm described in RFC7415 [3]

When using the default leaky bucket algorithm described in §3.5.1 of RFC7415, the *oc* parameter is inverted to obtain the control parameter: the minimum (average) time interval *T* between SIP requests. This algorithm differs from common leaky-bucket variants where the bucket leak rate is varied and the fill amount is constant, by having (equivalently) a constant leak rate and a variable fill amount. With a fixed control parameter, the admission rate behaviour in the long-term is the same, but may be different over shorter time intervals. In particular the amount that can be added to an empty bucket due to a traffic surge is not constant, i.e. it is dependent on the control rate *oc*.

To see this, consider the scenario whereby the bucket occupancy (fill) has dropped to 0 and then there is a sudden burst of calls. The number admitted will be Int[τ/T] + 1, where τ is the value (TAU used in [3]) of the reject or 'tolerance' threshold and Int[x] is the greatest integer less than or equal to x, the first term of which is proportional to the control rate. This bucket fill takes a time of $\tau + T$ to drain to 0. Roughly speaking one can regard τ as representing this source/client's component of the queuing time at the server, which is therefore bounded by the bucket (and similarly T is this client's component of the service time at the server). Therefore this meets one of the principle objectives which is to bound response times, but with a degree of adaptation to the number of clients generating traffic, because the more source/clients the lower the rate $oc = 1/T$ will be.

We have shown that the fill of a bucket up to the reject ('tolerance') threshold τ represents the waiting time at the server, so that the number admitted in a traffic surge is proportional to the current control rate specified by the *oc* parameter, and therefore it adapts well to varying numbers of source/clients.

If a target server of overload is connected to a large number of sources then the state (fill) of the buckets at sources can become synchronised, e.g. when a large simultaneous surge of traffic fills all the buckets so that they all become full at about the same time. Each bucket would then drain down to τ at about the same time, so that the next requests from the sources would be admitted nearly simultaneously. This would result in very bursty arrivals at the target, and hence long response times. To avoid such 'resonance' each source should include the enhancement described in §3.5.3 of RFC7415 by randomising the bucket fill when it becomes empty.

B.2 Restriction priority levels

The default restriction priority levels specified in §8.3 have been derived from key principles (§8.1 and §8.2), the rationale for which is explained in the following sections.

B.2.1 Correspondence between exempt request methods and INVITEs

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. Similarly 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), so that limiting the rate of INVITEs will also limit the rate of PRACKs. 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).

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).

In summary, because an exempt method (ACK, PRACK, CANCEL, BYE) can only follow on from a preceding INVITE, the above correspondences imply that (in the longer term) controlling the rate of INVITEs generated will indirectly control the rate of those exempt request methods generated.

B.2.2 Within dialogue request methods

An INVITE may be sent within a dialogue in order to modify the dialogue state or session parameters. It is undesirable to reject such a re-INVITE because it 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 to a re-INVITE because it is within-dialogue. In addition it may also 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 should not be exempt from restriction. Consider a situation where there are a very large number of concurrent dialogues at a SIP node (e.g. because the holding time is large) and where the rate of within-dialogue requests such as UPDATEs for each and every dialogue is very high, perhaps because of some future application or because of some rogue behaviour. If UPDATEs were exempt from restriction this could cause overload and the control would be unable to do anything about it.

The above opposing factors have been used to derive the default restriction priorities (§8.3), where (non-emergency) within-dialogue request methods including re-INVITE and UPDATE are given a higher priority of 2, which is the highest possible non-emergency priority, compared to 4 for a dialogue-initiating INVITE.

Where the traffic mix is not extreme (in contrast to the example above), such that the ratio of priority 2 within-dialogue requests to dialogue-initiating INVITEs is of a similar magnitude to that between ACKs and such INVITEs, then the probability of rejecting such priority 2 requests will be very low because of their relatively higher priority and because their rate will be indirectly limited by restriction of the dialogue-initiating INVITE rate.

For a given scenario the actual probabilities of rejection of each SIP request cannot be quantified precisely without a detailed knowledge of the traffic mix, including the dependence upon time, and would require application of theory or discrete event simulation.

B.3 Failover and control parameters *oc-validity* and *oc-seq*

A special case that requires detailed consideration is when overload control is active before target server failover.

The control parameters which are returned from an overloaded target (in the Via Headers) before it fails are associated with the IP address of that target. The restrictor allocated at the source/client against that address will persist through the failover process with the *oc* max rate parameter value from the previously active target, as long as the duration timer using the *oc-validity* parameter value remains running or until an update with a new *oc* value and higher *oc-seq* value is received for the same IP address. 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.

These factors have implications for the size of the *oc-validity* parameter and the *oc-seq* values which are both sent in responses from the standby target server as it activates, that are discussed in §B.3.1 and §B.3.2 below.

B.3.1 *oc-validity*

The required minimum size of *oc-validity* can be determined with reference to the timing of events as shown in Figure 13.

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) plus the time it takes to distribute control to all sources which 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 failover to occur would be just before all updates have been received (see Figure 13). So the restrictors should persist for an additional time which is the time it takes for the newly activated target server to start responding and stabilise. This duration will depend upon several factors, including

- the method of target failure detection used by the sources
- whether or not state is shared between active and standby
- whether overload control is active before or after the failover (or both)

Therefore the minimum value of *oc-validity* must be

$$2 \times (\text{time between control updates}) + (\text{expected duration of failover stabilisation})$$

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.

Consequently 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

$$3 \times (\text{time between control updates}) + (\text{expected duration of failover stabilisation})$$

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

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 [4] 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 [4] and [3].The behaviour will also be affected by the *oc-seq* parameter (§4.4 of [4]). This is an unsigned integer value that increases over time, used by the source to apply the latest level of control and ignore older responses, and must always be inserted by the server. The values returned by the target before failover and the standby when becoming active will affect the behaviour of control at the source.

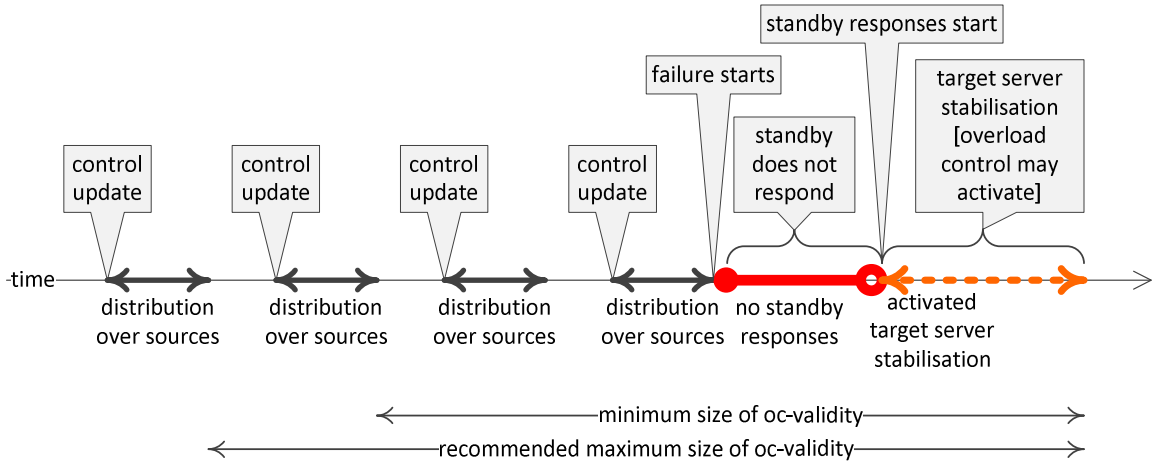

Figure 13: Timeline of events around target server failover effecting the duration range of the *oc-validity* parameter

B.3.2 *oc-seq*

During normal operation the oc-seq value is increased by a server according to §10.3.

The setting of the oc-seq value after failover described in §10.3 depends upon whether control state is shared between active and standby servers.

B.3.2.1 Control state is shared

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

B.3.2.2 Control state is not shared

If the target's control state is not shared with its standby, and the latter's overload control state is initially inactive as it starts to return responses, it would send *oc-validity*=0 indicating this inactive state, resulting in a surge of traffic from the sources. To prevent this the newly activated standby must return an initial value of *oc-seq* that is lower than was received before failover in order that the sources would ignore the previous values of *oc* and *oc-validity*, (but it must not be 'substantially' lower than the previous values from the failed target otherwise it would be assumed that a numerical overflow had occurred and control would carry on from the new lower *oc-seq* value, as per §4.4 of [4]). By doing this the standby is less likely to enter an overloaded state for a period following failover, but it is likely to go into overload as the oc-validity duration timers (see §B.3.1) expire at the sources. To ensure that oc values are then effective at sources the oc-seq value must be

increased to a value greater than before failover, i.e. when oc-validity > 0 for the first time following failover.

If a sequence value proportional to time is used, a solution to the above is for the standby to set the initial *oc-seq* to a value derived from the time of activation minus the value of *oc-validity*. This does not violate the requirement that *oc-seq* should be an increasing function of time (§4.4 of [4]) because the implication is that it applies to a particular target under control, and in this case control is transferring from the failed target to the standby, which are different servers. The newly activated target should revert to 'normal' derivation of *oc-seq* from 'proper time' at or before the next control activation.

B.3.3 IP Address/port and Control State

Table 9 below summarises the factors discussed above in §B.3.1 and §B.3.2 for the range of cases covering control state being shared (between active and standby target) or not being shared, and IP address/port being shared or not.

<table>
<tr><th></th><th>Common address/port</th><th>Distinct IP address/ports</th></tr>
<tr><td rowspan="3">Control state is not shared (nor call control state).
The standby target server is not aware that control is active at the sources, and in particular any control values.</td><td colspan="2">Since the standby is only 'warm' when activated the starting overload control state is likely to be 'not overloaded' (determined by oc=0).</td></tr>
<tr><td>There will be a single source restrictor since active and standby target have the same address/port.</td><td>Requests would be uncontrolled as soon as requests are sent to the new address/port. This would result in a surge of traffic from the source.
A solution is for the source restrictor to be associated with the address/port of both active and standby.</td></tr>
<tr><td colspan="2">Whether or not a common address/port is used, for control to be effective the value of oc-seq needs to be set according to §B.3.2.</td></tr>
<tr><td>Control state is shared.
The standby target server will know that control is active and certain control parameter values; it may know the values of additional adaptation parameters.</td><td>Adaptation would continue from the failed target state to the newly active standby target state. Whatever the load state of the latter, smoother transition would be expected between states.</td><td>From the view of the target this is the same as with a common address/port. In contrast, at the source a new restrictor would be allocated against the new address/port (unless the two addresses are identified), leading to the old restrictor being unused and terminated since overload control parameters received would not refer to it. There would be a short transient difference in behaviour because the new restrictor would start from a different state, e.g. initial bucket fill, but this effect is likely to be minor.</td></tr>
</table>

Table 9: Possible behaviours on failover from a target node to a standby when overload control is active at the target before failure

B.4 Background and analysis for solution to non-compliant and non-conforming sources

B.4.1 Requirements and restriction location

RFC7415 [3] is designed so that requests are rejected at the source (client) sending to the overload target (server), using recommended methods (§7), because in this way no load is incurred on the target per rejected request.

However, as per §5.10.2 of RFC7339 [4], allowance should be made for non-compliant sources in case the source restriction function has not yet been implemented. In this situation, restriction may have to be performed at the target as a proxy source rate control, rather than located at the remote source as usual. Whilst it is preferable that rejections should be handled as normal with an explicit response (see requirement in §11.1), the resource devoted to rejecting in this way must be limited. This can be done by discarding requests, i.e. throwing them way without response, once the rate of rejection becomes intolerably high.

The solution to this required by ND1653 is described in §13.

As a result there are now 3 possible ways in which requests destined for an overloaded target can be restricted:

- Explicit rejection of non-exempt (§6.3.3) requests at a source compliant with RFC7415
- Explicit rejection of non-exempt (§6.3.2) requests at the target
- Discard of all requests (§6.3.2) at the target

where the last two mechanisms are defined in §13. A detailed analysis of these follows in §B.4.2, §B.4.3 and §B.4.4.

B.4.2 Objectives

The SRC function of a given overload target node will derive request rates to be sent to each source. This can include sources that are known to be non-compliant.

A source could be managed in a functionally equivalent way by the target applying an identical restriction function locally for all requests arriving from that source. This would mean that it would have to do all the usual request differentiation in order to apply priorities, and requests which are not admitted should be rejected by returning an appropriate explicit response for the usual reasons. But there would be a significant overhead of rejecting such requests. It is this that must be bounded by the target, otherwise the target cannot properly protect its capacity and prevent overload.

So the ideal objective of managing a non-compliant source is as follows:

- Treat requests in the same way as they would be treated at the source, in terms of request differentiation, response, etc.
- Bound the resource/workload at the Target to be the same as that would be the case if all requests were admitted at the rate derived by the Target

Since the arrival rate for such a source may be unbounded, ultimately both of these objectives cannot be met, and discard will be necessary. The same is true for a non-conforming source where the objectives are less clear-cut. Fortunately an enhanced leaky bucket rate limiter provides a proxy source rate control function that can manage both scenarios with a single feature.

B.4.3 Simplified analysis of an enhanced leaky bucket rate limiter

The source rates derived by the target will be related to the average resource workload required to process requests, according to the current mix of traffic. The default rate controller algorithm as specified in §3.5 of RFC7415 does not measure the overhead of rejecting a request, because when a request is rejected the bucket fill is not increased. However it is straightforward to extend the capability to do this simply by increasing the bucket fill by an amount that is less than that which is added when a request is admitted. This amount will depend in general on the system design and so is not been prescribed here. For example, it might be proportional to the cost of admitting a call. Given that with the default algorithm the fill amount T is dependent upon the (variable) *oc* rate $T = 1/R$ (see §B.1), a way to do this would be to have a parameter which is the cost of rejection as a fraction ϕ of the cost of admission. Alternatively it may be that the cost of rejection is near constant, say T_0, since rejection is done before any significant call processing. So a good general representation might by a simple linear one $T_0 + \varphi T$ with the two parameters T_0 and ϕ.

The effect of this will be that as the request arrival rate increases above the max rate R, the admission rate will decrease.

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 λ and admission rate γ are just equal (no rejects):

$$\lambda T = \gamma T = 1$$

and since $T = 1/R$ we have all rates equal to the *oc* rate:

$$\lambda = \gamma = R$$

As the arrival rate increases the admission rate decreases just enough so that the bucket utilisation remains at 1, i.e.

$$\gamma T + (\lambda - \gamma)(\varphi T + T_0) = 1$$

which has the solution

$$\gamma = \begin{cases} \lambda & \lambda < R \\ \dfrac{R - \lambda(\varphi + RT_0)}{1 - \varphi - RT_0} & R \le \lambda \le \dfrac{R}{\varphi + RT_0} \\ 0 & \dfrac{R}{\varphi + RT_0} \le \lambda \end{cases}$$

This is illustrated in Figure 14.

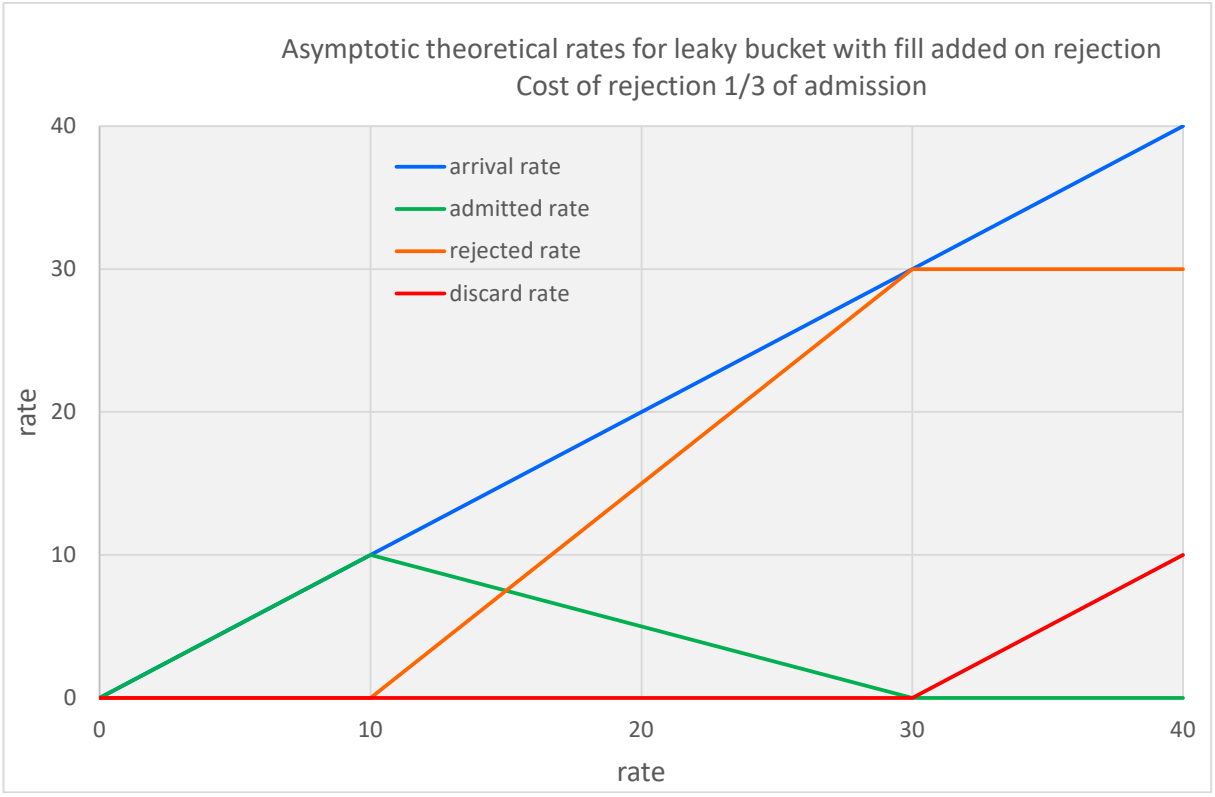

Figure 14: Rates for enhanced leaky bucket with fill added to represent cost of rejection, with R=10, ϕ=1/3, T_0=0.

Once the bucket is rejecting every request, i.e. γ=0, if the arrival rate continues to increase then we have $\lambda > \frac{R}{\varphi + RT_0}$, which implies that

$$\gamma T + (\lambda - \gamma)(\varphi T + T_0) > \frac{R(\varphi T + T_0)}{(\varphi + RT_0)}$$
$$= \frac{RT(\varphi T + T_0)}{(\varphi T + RTT_0)}$$
$$= 1$$

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 can be detected by having an additional *discard* 'tolerance' threshold $\tau*$ which is higher than any of the *reject* 'tolerance' thresholds. At this point the target should ignore requests with as little processing as possible in order to minimise the resources spent on processing messages. The fill would not be increased as each request is discarded (in principle it could be increased by an amount that reflects the cost of discard, but this should be very small). As the fill decreases below this threshold, requests could again be rejected. Therefore in this range of extreme arrival rate, denoting the reject rate by ω and the discard rate by δ, we have:

$$\gamma = 0$$
$$\omega = \frac{R}{\varphi + RT_0}$$
$$\delta = \lambda - \omega$$

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, as shown in Figure 14, where $\omega = \frac{10}{\left(\frac{1}{3}\right)} = 30$.

B.4.4 Summary of characteristics of the solution

The solution described in §B.4.3 has the following benefits:

- 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, so that the workload remains bounded
- The next critical arrival rate is when 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. Thus the workload still remains bounded. Once in this region the source receives no service from the target
- CPs are penalised by exceeding their assigned control rates because the success rate decreases and eventually they get no service at all. This gives an incentive to comply with SRC

B.5 Deriving the goal rate of SIP requests

Below we argue that periodic measurement of system (e.g. process) utilisation and received SIP request rate allow an estimate of the system capacity in terms of a (variable) goal request rate to be derived according to simple computation, even when the nature of the system workload changes quickly.

B.5.1 Goal rate and resource utilisation

RFC7415 (and RFC7339) work by restricting the rate of SIP requests, and to achieve this ND1653 requires that the target node is required to derive a goal rate of requests (§8.4.1), which is the maximum request rate at which acceptable performance – in particular response time - will be met. However the workload at a target node is typically measured by utilisation of resources such as process/processor, which will depend upon the complexity of request processing and will vary according to the type of requests, complexity of routeing, dialogs or sessions, etc, often referred to as the (call/session/message) 'mix'. Consequently the goal rate will also vary over time. Fortunately there is usually a fairly precise maximum value of utilisation that gives acceptable performance which can be determined through system performance testing, and therefore this 'goal utilisation' value becomes a proxy for the (variable) goal rate. Note that for a constant 'mix' the relationship between utilisation and (accepted) request rate is approximately linear, whereas the response time or message queue sizes are markedly non-linear, e.g. the former tends to infinity as the system fills up. Thus although response time or queue size measurements can indicate when you are in or very close to overload they would be very difficult to use for predicting (at lower loads) the goal rate.

The next section considers the relationship between resource utilisation and request rate and how this varies over time.

B.5.2 Estimating the workload per request

In order to activate overload control quickly it is important for a target to estimate the goal rate before any overload arises. For a constant traffic mix, process utilisation usually depends linearly

upon the request rate, and this enables the goal rate to be estimated at intervals of time, by computing the average utilisation per request. Traditionally (e.g. ISUP networks) the utilisation per call initiation (IAM/IFAM) would be measured, which becomes *per initial request* (non-dialogue or dialogue-initiating) in a SIP context, and this would enable a goal *initial* request rate to be determined by linear extrapolation. In this case within-dialogue requests do not even need to be counted, even though they still contribute to the average utilisation per initial request. In contrast RFC7415 and RFC7339 both stipulate that the control parameter refers to *all* requests, in which case the utilisation *per request* is required at the target, and the mapping to the rate of *restrictable* requests (i.e. that are not exempt from restriction) is done at the point where restriction is applied, by means of a rate reduction factor calculation.

Whilst solutions for the traditional and the SIP RFC approaches can be similar, differences arise because of the measurement and variability of the utilisation per non-exempt (initial) request compared to the utilisation per (any) request.

Usually such an average utilisation will vary slowly, but it is necessary to take account of the variation in the workload mix, which may sometimes change abruptly, for example due to

- Session/dialogue setup traffic surges followed by (a holding time later) corresponding disconnection surges. The effect of the latter will be worst when there is little variation in holding time, e.g. as occurs for short announcements
- Session/dialogue setup surges where the complexity of calls suddenly changes, e.g. from simple to an IN call
- Mass disconnection surges due to media resource failure
- Mass reconnection (e.g. re-INVITE surges) due to SIP server failover

A sudden surge of initial (non-exempt) requests causes an abrupt change in the mix, being dominated by initial requests without any follow-on request workload - the latent committed workload. For dialogue-initiating requests this implies that the start of the surge is dominated by INVITEs. The latent workload – consisting of ACKs (or PRACKs) and BYEs/200OK, arriving at different timescales, must not be rejected by the control.

For the traditional method of measuring the average utilisation per initial (non-exempt) request, before the load due to follow-on workload has arrived, the utilisation per initial (non-exempt) request will be lower than afterwards. For the SIP RFC methods of measuring the utilisation per (any) request, if the resource cost of initial requests were higher than per follow-on in-dialogue requests then the resource cost per request at the start of a surge would be relatively higher than when the follow-on load arrives (although it could instead be that the relative change in resource cost is from low to high).

For either method the average utilisation could suddenly change in such a scenario. Consequently, given a known max tolerable utilisation, the capacity in terms of the max (initial or total) request rate would also suddenly change. Furthermore this cycle may repeat, such as when calls are being terminated at a large (media) capacity destination with a holding time that is very nearly constant. The resulting highly unstable throughput could lead to over-commitment of workload and lost messages or unacceptably long response times. What would be most serious and must be avoided would be to over-estimate capacity because of a short-term drop in utilisation per (initial or total) request, i.e. suddenly increase the estimate of the goal rate, higher than it truly is over longer timescales.

A solution to this is to smooth the estimate of the utilisation per request i.e. to limit the magnitude of the change in the estimate from one update to the next. Furthermore, the estimate of the average utilisation can be slowly smoothed when it is *decreasing* and conversely more quickly smoothed when it is *increasing* (explained in more detail in §B.5.5 following).

B.5.3 Measuring utilisation

When computing the average utilisation per SIP request from measurement, it is essential that the utilisation measurement is specific to the workload incurred by processing SIP requests only (including associated call/session workload) such as by measuring the CPU utilisation due to specific processes. Other unrelated workload, e.g. due to background or admin load including periodic processes such as garbage collection, provisioning, stats collection, etc, must not be included in the measurement, as these would introduce inaccuracy at best or instability at worst, because this workload is independent of the SIP request rate being controlled.

Furthermore the SIP request / call processing workload must be protected from this other workload anyway in order to protect SIP response times. Ways of doing this include:

- Using an effective CPU priority scheduling scheme is available, giving it the highest CPU scheduling priority
- Running it on dedicated CPU resource which is not used by other workload
- Limiting the number of processes/threads that the other workload uses, so that only a limited CPU utilisation is available to it

B.5.4 Measuring target workload per request

It is usually impractical to add up component processor time measurements used for *each* SIP request in order to obtain the average workload per request. Instead the total (time) average utilisation of the resource (e.g. processor utilisation) can be captured periodically for all requests.

For a measurement interval (t_i, t_i+T) of length T we can obtain the following measured sample of the resource (utilisation) time:

$$\tau_i = \frac{\text{(time) integrated utilisation over } (t_i, t_i + T)}{\text{number of request messages arriving in } (t_i, t_i + T)}$$

$$= \frac{\text{(time) average utilisation over } (t_i, t_i + T)}{\text{average rate of request messages arriving in } (t_i, t_i + T)}$$

For example, the average utilisation may be obtained by a measurement of process occupancy.

Considering the size of the measurement interval T, there is a trade-off:

- The longer it is the less variable will be the derived value of τ_i since it will capture more sessions/requests
- The shorter it is the quicker τ_i will respond to changes in the workload mix, but the more variable it will be

Some dialogs or sessions may be entirely in the interval, but many may not since the frequency of updating the measurement will mean that T will be smaller than the average session time. So to get a reliable estimate of the workload per request we rely on the multiplexing many dialogs/sessions together. The interval does not need to be of constant length, indeed there are reasons why, ideally, it should not be. For example one policy is to update measurements after a maximum number of requests have arrived, or a maximum time between updates has been reached (giving two configurable parameters), whichever occurs sooner. This allows control to be reactive to sudden surges of traffic but avoids too frequent updates when the traffic is lower.

B.5.5 Smoothing the target workload per request

A geometrically smoothed estimate $\overline{\tau}_i$ of the i^{th} resource utilisation time estimates τ_i can be derived as follows.

When the latest $(i+1)^{th}$ interval gives

- an increase in the resource time, i.e. $\tau_{i+1} > \overline{\tau}_i$, then use a geometric smoothing coefficient p_U (for '*Up*')

$$\overline{\tau}_{i+1} = p_U \tau_{i+1} + (1 - p_U)\overline{\tau}_i$$

- a decrease in the resource time, i.e. $\tau_{i+1} \le \overline{\tau}_i$ then use a different geometric smoothing coefficient p_D (for '*Down*')

$$\overline{\tau}_{i+1} = p_D \tau_{i+1} + (1 - p_D)\overline{\tau}_i$$

The coefficients must both be strictly greater than 0 to be adaptive, but if we make $0 < p_D < p_U \le 1$ then the smoothed estimate of the resource time per request will increase more quickly and decrease more slowly. In this way we can avoid sudden drops in the value, which would lead to sudden increases to the estimate of capacity, but still allow quicker adaptation to a reduction in capacity due to higher values of the resource effort.

B.5.6 The updated goal rate of SIP requests

Having obtained the smoothed estimate $\overline{\tau}_i$ of the utilisation time per request, then if the total utilisation available for SIP request processing (only) is U^* then the max possible SIP request rate is simply $U^*/\overline{\tau}_i$.

B.5.6.1 Example

Suppose that the target system has 32 CPUs and that it is known that in order to meet response time requirements the relative occupancy must be less than 81.25% (=26/32). Only 6 process (thread) instances can be used for background (non-SIP session) workload, therefore incurring a maximum 6/32=18.75% occupancy. Therefore the maximum allowed relative occupancy for SIP call/request processing is $U^* = 81.25\% - 18.75\% = 62.5\%$ If the latest smoothed estimate from measurement of CPU time per request is $\overline{\tau}_i = 2.5\text{ms}$ then the goal SIP request rate would be

$$\frac{U^*}{\overline{\tau}_i} = \frac{62.5\%}{2.5/1000} = 8000\ \text{reqs/sec}$$

B.6 Comparison of ND1653 variants with differing interpretations of rate

NICC considered variations on the scheme currently specified in ND1653, where the value of the *oc* control parameter is interpreted to mean the rate of *initial* SIP requests only, that are not exempt from restriction, i.e. those that are dialogue-initiating or are non-dialogue requests (for NNI currently this means the rate of Invites), rather than entire rate of SIP requests as required by RFC7339 and RFC7415.

B.6.1 Rationale for considering re-interpretation of rate

The main reason for taking this approach is that it results in significant simplifications to the control functions which are outlined below.

RFC7415 and RFC7339 specify that the control parameter value applies to all SIP requests (see RFC extracts: §3.4 of RFC7415 – SIP Rate Control and §5.3 of RFC7339 – SIP Overload Control). However, both RFCs imply that some SIP Requests should be exempt from rejection by the control (§5.10.1 of RFC7339 – SIP Overload Control) because rejecting them would degrade performance seriously. ND1653 identifies these as ACK, BYE, CANCEL, PRACK. It is therefore necessary to multiply the *oc* rate value by a Rate Reduction Factor (RRF) to obtain a maximum admission rate that is applied to a leaky bucket restrictor, at source (client) or target (server) of overload.

It is useful to examine the reason that RFC7339 gives for using the entire SIP request rate, i.e. ‘so that both the client and server have a common view of which messages the overload control applies to’ (see the Note in §5.3 of RFC7339 – SIP Overload Control). This makes sense in terms of differentiating between priorities of restrictable requests. However identifying restriction-exempt requests is fundamental because of the potential effects on performance, and we should expect that both source (client) and target (server) should agree in advance what the exempt subset should be, as has been done in ND1653. In which case the client is not at liberty to change this aspect of assigning priorities. Therefore why build this possibility into the protocol?

With the alternative interpretation being explored here, i.e. that the rate only applies to non-exempt (initial) requests, the RRF is therefore no longer required.

One way of realising this alternative would be for an ND1653 compliant node to re-interpret the *oc-algo* token "rate" in this way, and also mark ND1653 compliant nodes with which it communicates with as such, in particular to distinguish them from any RFC7415 node with which it also communicates.

Another way would be to define a new *oc-algo* token, say "nxrate" which stands for *non-exempt rate*. In this way a node can declare that it is ND1653 compliant through the protocol, and detect that it is communicating with an ND1653 compliant node. In fact, the structure of the SOC protocol has been designed to allow for additional restriction algorithms, which is clear from the RFC extracts – see §4.2 of RFC7339 – SIP Overload Control.

A comparison of the above two approaches with the current (RFC compliant) version of ND1653 is presented in Table 10.

B.6.2 Conclusions and NICC decision

NICC decided unanimously that ND1653 should interpret rate as meaning non-exempt request rate and differentiate this by means of a new *oc-algo* token “nxrate”, i.e. adopt the approach of the rightmost column of Table 10. It was also agreed that, after publication of ND1653, consideration should be given to registering the new value with IANA and drafting a new RFC if required.

B.6.3 Comparison of the differing interpretations of the control rate

ND1653 with control parameter:	**specifying the *entire* SIP request rate**	**specifying the *non-exempt* (restrictable) SIP request rate only**	
***oc-algo* token:**	**"rate"**	**"rate"**	**"nxrate"**
Compliant with RFC7339 and RFC7415	Yes. However ND1653 includes the detailed specification of functions and configuration that are not included in RFC7415 or RFC7339.	No. The RFCs require that the control parameter applies to the entire rate of SIP requests.	No. Adjacent column to the left applies. In addition, the RFCs only include the *oc-algo* tokens "loss" and "rate". However RFC7339 has been designed to accommodate additional restriction algorithms (in fact originally there was a third "win" for windows flow-control) and therefore in principle new *oc-algo* value tokens could be added without any fundamental changes to the signalling protocol.

Requires RRF	Yes. Each source (client) independently reads the *oc* value received from a target and using its own RRF instance maps to a value to apply to the restrictor. A target (server) does the same for each restrictor applied locally for traffic received from each (compliant or non-compliant) source. Therefore if there are *N* sources then there are effectively 2*N* independent RRF computations being done in total. Since there is likely to be some variation in the traffic mix from each source, e.g. in terms of clearing (BYE) direction, there is also likely to be some variation in the derived rate values, even for the same target. Note: A BYE is a SIP Request that is exempt from restriction and therefore would contribute to the reduction applied by the RRF, but the SIP Response 200 OK is not counted since control does not apply to responses. So the relative counts of these in each direction will affect the RRF.	No. Sources (clients) do not because they read the *oc* value and apply directly to the request restrictor, which is not queried for restriction-exempt requests (priority 0). A target (server) does the same for each restrictor applied locally for traffic received from each (compliant or non-compliant) source. The rate reduction is effectively included by the goal rate derivation function which can derive the non-exempt (initial) request rate directly. The simplest design for the goal rate function is to derive such a rate for all sources, although it would be possible to make this specific to each source. For operational reasons it would be useful to derive the entire request rate goal equivalent to the non-exempt request rate goal, and this can be obtained by the goal rate derivation function.
Rationale for rate interpretation	According to RFC7339 – SIP Overload Control (Note in §5.3), the *oc* rate is the *entire* request rate is '*so that both the client and server have a common view of which messages the overload control applies to*' and from RFC7415 – SIP Rate Control (§3.4) '*request prioritization is a client's responsibility*'.	Identifying restriction-exempt requests is fundamental because of the potential effects on performance, and we should expect that both source (client) and target (server) should agree in advance what the exempt subset should be, as has already been done in ND1653. In which case the client is not at liberty to change this aspect of assigning priorities and it is unnecessary for the control rate to refer to exempt requests.

Operational	Operations will require rate statistics in terms of calls (Invites), especially for NNI. Although this is not the same as the *oc* parameter rate it can easily be obtained after the RRF has been applied.	The *oc* rate is the non-exempt (initial) rate and therefore for NNI the same as the Invite (or calling) rate.	
Magnitude of the difference from the RFCs	ND1653 has many functions specified additional to the RFCs, but these are still compliant with RFC7415.	In addition to the many (compliant) functions specified that are additional to the RFCs, the interpretation of rate is non-compliant.	Relative to the functions additional to the RFCs specified for ND1653, the non-compliant addition of a new *oc-algo* token does not seem so significant. Furthermore the RFCs were developed in a way that would allow additional restriction algorithms to be used.
Distinguishing ND1653 compliance	Compliance with ND1653 cannot be declared or discovered by the protocol alone, and therefore compliance on a connection has to be configured at each end. I.e. *oc-algo*="rate" does not distinguish between RFC7415 and ND1653.	Adjacent column to the left applies.	A source (client) compliant with ND1653 must declare compliance by including "nxrate" in the *oc-algo* value list and similarly a compliant target (server) must also include "nxrate". Consequently compliance can be discovered through the protocol alone.

Table 10: Comparison of ND1653 with variant interpretations of rate

B.6.4 RFC extracts

B.6.4.1 RFC7339 – SIP Overload Control

4.2. The *oc-algo* Parameter

…

This parameter contains names of one or more classes of overload control algorithms … the SIP client MAY insert other tokens, separated by a comma, in the *oc-algo* parameter if it supports other overload control schemes such as a rate-based scheme [RATE-CONTROL]. Each element in the comma-separated list corresponds to the class of overload control algorithms supported by the SIP client. When more than one class of overload control algorithms is present in the *oc-algo* parameter, the client may indicate algorithm preference by ordering the list in a decreasing order of preference. However, the client cannot assume that the server will pick the most preferred algorithm.

When a downstream SIP server receives a request with multiple overload control algorithms specified in the *oc-algo* parameter (optionally sorted by decreasing order of preference), it chooses one algorithm from the list and MUST return the single selected algorithm to the client.

…

The *oc-algo* parameter does not define the exact algorithm to be used for traffic reduction; rather, the intent is to use any algorithm from a specific class of algorithms that affect traffic reduction similarly.

…

5.1. Determining Support for Overload Control

If a client determines that this is the first contact with a server, the client MUST insert the *oc* parameter without any value and MUST insert the *oc-algo* parameter with a list of algorithms it supports. This list MUST include "loss" and MAY include other algorithm names approved by IANA and described in corresponding documents…

…If the server determines that the message is from a new client or a client the server has not heard from in a long time, the server MUST choose one algorithm from the list of algorithms in the *oc-algo* parameter. It MUST put the chosen algorithm as the sole parameter value in the *oc-algo* parameter of the response it sends to the client…

5.3. Determining the *oc* Parameter Value

The value of the *oc* parameter is determined by the overloaded server using any pertinent information at its disposal. The only constraint imposed by this document is that the server control algorithm MUST produce a value for the *oc* parameter that it expects the receiving SIP clients to apply to all downstream SIP requests (dialogue forming as well as in-dialogue) to this SIP server. Beyond this stipulation, the process by which an overloaded server determines the value of the *oc* parameter is considered out of the scope of this document.

> Note: This stipulation is required so that both the client and server have a common view of which messages the overload control applies to. With this stipulation in place, the client can prioritize messages as discussed in Section 5.10.1.

…

5.5. Using the Overload Control Parameter Values

A SIP client MUST honour overload control values it receives from downstream neighbours. The SIP client MUST NOT forward more requests to a SIP server than allowed by the current *oc* and *oc-algo* parameter values from that particular downstream server…

The particular algorithm used to determine whether or not to forward a particular SIP request is a matter of local policy and may take into account a variety of prioritization factors…

5.10.1. Message Prioritization at the Hop before the Overloaded Server

During an overload condition, a SIP client needs to prioritize requests and select those requests that need to be rejected or redirected. This selection is largely a matter of local policy. It is expected that a SIP client will follow local policy as long as the result in reduction of traffic is consistent with the overload algorithm in effect at that node. Accordingly, the normative behavior in the next three paragraphs should be interpreted with the understanding that the SIP client will aim to preserve local policy to the fullest extent possible.

A SIP client SHOULD honour the local policy for prioritizing SIP requests such as policies based on message type, e.g., INVITEs versus requests associated with existing sessions.

…

B.6.4.2 RFC7415 – SIP Rate Control

3.3. Client and Server Rate-Based Control Algorithm Selection

…

Support of rate-based control MUST be indicated by clients including the token "rate" in the *oc-algo* list. Selection of rate-based control MUST be indicated by servers by setting *oc-algo* to the token "rate".

3.4. Server Operation

…

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, since as per [RFC7339] and REQ 13 of [RFC5390], request prioritization is a client's responsibility.

3.5.2. Priority Treatment

As with the loss-based algorithm in [RFC7339], a client implementing the rate-based algorithm also prioritizes messages into two or more categories of requests, for example, requests that are candidates for reduction and requests that are not subject to reduction (except under extenuating circumstances when there aren't any messages in the first category that can be reduced).

…

16 History

Document history		
Version	**Date**	**Milestone**
1.1.1	21st October 2019	Initial publication
2.1.1	8th August 2020	Second publication to accompany the publication of ND1038