IETF SOC Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 28, 2014

C. Shen
H. Schulzrinne
Columbia U.
A. Koike
NTT

November 24, 2013

A Session Initiation Protocol (SIP) Load Control Event Package
draft-ietf-soc-load-control-event-package-11.txt

Abstract

We define a load control event package for the Session Initiation Protocol (SIP). It allows SIP entities to distribute load filtering policies to other SIP entities in the network. The load filtering policies contain rules to throttle calls based on their source or destination domain, telephone number prefix or for a specific user. The mechanism helps to prevent signaling overload and complements feedback-based SIP overload control efforts.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 28, 2014.

Copyright Notice

Table of Contents

1. Introduction

Proper functioning of Session Initiation Protocol (SIP) [RFC3261] signaling servers is critical in SIP-based communications networks. The performance of SIP servers can be severely degraded when the server is overloaded with excessive number of signaling requests. Both legitimate and malicious traffic can overload SIP servers, despite appropriate capacity planning. Some of the SIP server overload situations are predictable, while others are not.

There are four common examples of predictable short-term increases in call volumes. Viewer-voting TV shows or ticket giveaways, special holidays such as New Year's Day and Mother's Day, end-of-season peaks in phone orders, or after forecastable natural disasters such as hurricanes. On the other hand, examples of unpredicted or unpredictable mass calling may happen after earthquakes, terrorist attacks, or at call centers that provide troubleshooting services when a mass service fails. When possible, only calls traversing overloaded servers should be throttled under overload conditions.

SIP load control mechanisms are needed to prevent congestion collapse in these cases [RFC5390]. There are two types of load control approaches. In the first approach, feedback control, SIP servers provide load limits to upstream servers, to reduce the incoming rate of all SIP requests [I-D.ietf-soc-overload-control]. These upstream servers then drop or delay incoming SIP requests. Feedback control is reactive and affects signaling messages that have already been issued by user agent clients. They work well when SIP proxy servers in the core networks (core proxy servers) or destination-specific SIP proxy servers in the edge networks (edge proxy servers) are overloaded. By their nature, they need to distribute rate, drop or window information to all upstream SIP proxy servers and normally affect all calls equally, regardless of destination. For example, in the ticket giveaway case, almost all calls to the hotline will fail at the core proxy servers; if the edge proxy servers leading to the core proxy servers are also overloaded, calls to other destinations will also be rejected or dropped.

Here, we propose an additional, complementary load control mechanism, called load filtering. Network operators create load filtering policies that indicate calls to specific destinations or from specific sources should be rate-limited or randomly dropped. These load filtering policies are then distributed to SIP servers and possibly SIP user agents that are likely to generate calls to the affected destinations or from the affected sources. Load filtering works best if it prevents calls as close to the originating user agent clients as possible.

The load filtering approach is most applicable for situations where a traffic surge and its source/destination distribution can be predicted in advance. It is less likely to be effective for the initial phase of unpredicted or unpredictable mass calling events. In the latter cases, the local traffic load may peak by more than an order of magnitude in minutes, if not seconds. This does not allow time to either effectively identify the load filtering policies needed, nor distribute them to the appropriate servers soon enough to prevent server congestion. Once other, more immediate, techniques (such as the loss-based or rate-based load feedback control methods) have prevented the initial congestion collapse, the load filtering approach can be used to effectively control the continuing overload.

Performing SIP load filtering involves the following components of load filtering policies: format definition, computation, distribution and enforcement. This specification defines the load filtering policy, distribution and enforcement in the SIP load control event package built upon existing SIP event notification framework. However, load filtering policy computation is out of scope of this specification, because it depends heavily on the actual network topology and other service provider policies.

It should be noted that although the SIP load filtering mechanism is motivated by the SIP overload control problem, which is why this specification refers extensively to parallel SIP overload control related efforts, the applicability of SIP load filtering extends beyond the overload control purpose. For example, it can also be used to implement quality of service or other service level agreement commitments. Therefore, we use the term "load control event package", instead of a narrower term "overload control event package".

The rest of this specification is structured as follows: we begin by listing the design requirements for this work in Section 4. We then give an overview of load filtering operation in Section 5. The load control event package for load filtering policy distribution is detailed in Section 6. The load filtering policy format is defined in the two sections that follow, with Section 7 introducing the XML

document for load filtering policies and Section 8 listing the associated schema. Section 9 relates this work to corresponding mechanisms in PSTN and other IETF efforts addressing SIP overload control. Section 10 evaluates whether this specification meets the SIP overload control requirements set forth by [RFC5390]. Finally, Section 11 presents security considerations and Section 12 provides IANA considerations.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Definitions

This specification reuses the definitions for "Event Package", "Notification", "Notifier", "Subscriber", "Subscription" as in [RFC6665]. The following additional definitions are also used.

Load Filtering: A load control mechanism which applies specific actions to selected loads (e.g., SIP requests) matching specific conditions.

Load Filtering Policy: A set of zero or more load filtering rules, also known as load filtering rule set.

Load Filtering Rule: Conditions and actions to be applied for load filtering.

Load Filtering Condition: Elements that describe how to select loads to apply load filtering actions. This specification defines the "call identity", "method", "target SIP identity", and "validity" condition elements (Section 7.3).

Load Filtering Action: An operation to be taken by a load filtering server on loads that match the load filtering conditions. This specification allows actions such as accept, reject and redirect of loads (Section 7.4).

Load Filtering Server: A server which performs load filtering. In the context of this specification, the load filtering server is the subscriber, which receives load filtering policies from the notifier and enforces those policies during load filtering.

Load Control Document: An XML document that describes the load filtering policies (Section 7). It inherits and enhances the common policy document defined in [RFC4745].

4. Design Requirements

The SIP load filtering mechanism needs to satisfy the following requirements:

- To simplify the solution, we focus on a method for controlling SIP load, rather than a generic application-layer mechanism.
- The load filtering policy needs to be distributed efficiently to possibly a large subset of all SIP elements.
- The solution should re-use existing SIP protocol mechanisms to reduce implementation and deployment complexity.
- For predictable overload situations, such as holidays and mass calling events, the load filtering policy should specify during what time it is to be applied, so that the information can be distributed ahead of time.
- For destination-specific overload situations, the load filtering policy should be able to describe the destination domain or the callee.
- To address accidental and intentional high-volume call generators, the load filtering policy should be able to specify the caller.
- Caller and callee need to be specified as both SIP URIs and 'tel' URIs [RFC3966] in load filtering policies.
- It should be possible to specify particular information in the SIP headers (e.g., prefixes in telephone numbers) which allow load filtering over limited regionally-focused overloads.
- The solution should draw upon experiences from related PSTN mechanisms [Q.1248.2][E.412][E.300SerSup3] where applicable.
- The solution should be extensible to meet future needs.

5. SIP Load Filtering Overview

5.1. Load Filtering Policy Format

Load filtering policies are specified by sets of rules. Each rule contains both load filtering conditions and actions. The load filtering conditions define identities of the targets to be filtered(Section 7.3.1). For example, there are two typical resource limits in a possible overload situation, i.e., human destination limits (N number of call takers) and node capacity limits. The load

filtering targets in these two cases can be the specific callee numbers or the destination domain corresponding to the overload. Load filtering conditions also indicate the specific message type to be matched (Section 7.3.2), with which target SIP entity the filtering policy is associated (Section 7.3.3) and the period of time when the filtering policy should be activated and deactivated (Section 7.3.4). Load filtering actions describe the desired control functions such as limiting the request rate below a certain level (Section 7.4).

5.2. Load Filtering Policy Computation

Computing the load filtering policies needs to take into consideration information such as overload time, scope and network topology, as well as service policies. It is also important to make sure that there is no resource allocation loop, and that server capacity is allocated in a way which both prevents overload and maximizes effective throughput (aka goodput). In some cases, in order to better utilize system resources, it may be preferable to employ an algorithm which dynamically computes the load filtering policies based on currently observed server load status, rather than using a purely static filtering policy assignment. The computation algorithm for load filtering policies is out of scope of this specification.

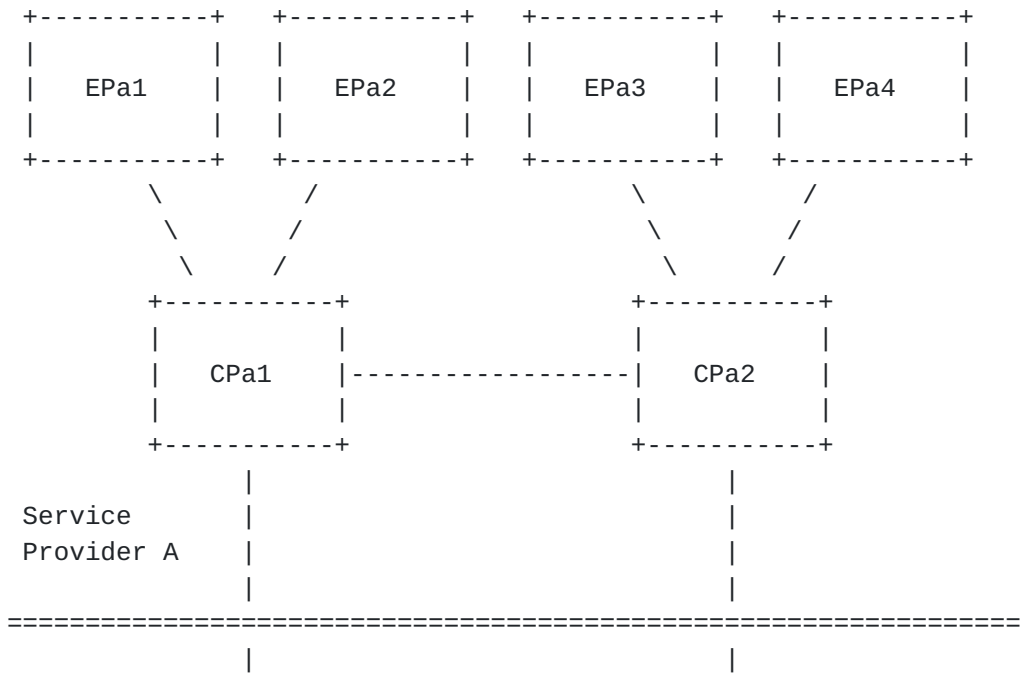
5.3. Load Filtering Policy Distribution

For load filtering policy distribution, this specification defines the SIP event package for load control, which is an "instantiation" of the generic SIP event notification framework [RFC6665]. The SIP event notification framework provides an existing method for SIP entities to subscribe to and receive notifications when certain events occur. Such a framework forms a scalable event distribution architecture that suits our needs. This specification also defines XML schema of a load control document (Section 7), which is used to encode load filtering policies.

In order for load filtering policies to be properly distributed, each capable SIP entity in the network SHOULD subscribe to the SIP load control event package from all its outgoing signaling neighbors, known as notifiers (Section 6.6). Subscription is initiated and maintained during normal server operation. Signaling neighbors are discovered by sending signaling messages. For instance, if A sends signaling requests to B, B is an outgoing signaling neighbor of A. A needs to subscribe to the load control event package of B in case B wants to curb requests from A. On the other hand, if B also sends signaling requests to A, then B also needs to subscribe to A. The subscription of neighboring SIP entities needs to be persistent so

that it is in place independently of any specific events requiring load filtering. Key to this is the fact that following initial subscription, the notifier sends a notification without a body if no load filtering policy is defined (Section 6.7), and that the subscription needs to be refreshed periodically to make it persistent, as described in Section 4.1 and Section 4.2 of [RFC6665]. The notifier will send a notification to its subscribers each time a new subscription or a subscription refresh is accepted (Section 6.7). The notification request includes in its body the current load filtering policies (Section 7.1) from the notifier. If no such load filtering policy exists, the notification request is sent without a body. The subscribers MAY terminate the subscription if it no longer considers the notifiers as its signaling neighbor, e.g., after an extended period of absence of signaling message exchange. However, if after un-subscribing, the subscriber determines that signaling with the notifier becomes active again, it MUST immediately subscribe to that notifier again.

We use the example architecture shown in Figure 1 to illustrate load filtering policy distribution based on the SIP load control event package mechanism. This scenario consists of two networks belonging to Service Provider A and Service Provider B, respectively. Each provider's network is made up of two SIP core proxy servers and four SIP edge proxy servers. The core proxy servers and edge proxy servers of Service Provider A are denoted as CPa1 to CPa2 and EPa1 to EPa4; the core proxy servers and edge proxy servers of Service Provider B are denoted as CPb1 to CPb2 and EPb1 to EPb4.



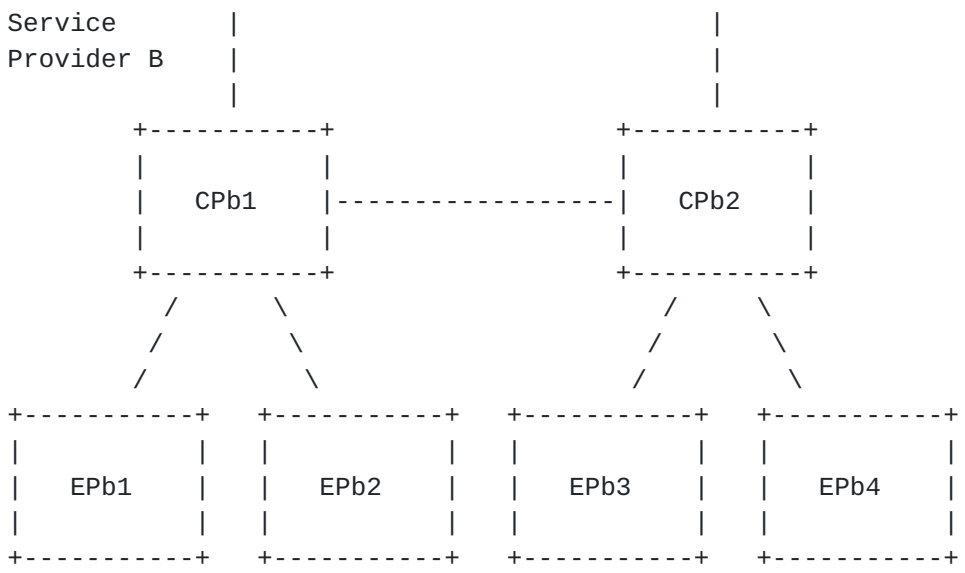


Figure 1: Example Network Scenario Using SIP Load Control Event Package Mechanism

At initialization stage, the proxy servers first identify all their outgoing signaling neighbors and subscribe to them. The neighbor identification process can be performed by service providers through direct provisioning, or by the proxy servers themselves via progressive learning from the signaling messages sent and received. Assuming all signaling relationships in Figure 1 are bi-directional, after this initialization stage, each proxy server will be subscribed to all its neighbors. That is, EPa1 subscribes to CPa1; CPa1 subscribes to EPa1, EPa2, CPa2 and CPb1, so on and so forth. The following cases then show two examples of how load filtering policy distribution in this network works.

Case I: EPa1 serves a TV program hotline and decides to limit the total number of incoming calls to the hotline to prevent an overload. To do so, EPa1 sends a notification to CPa1 with the specific hotline number, time of activation and total acceptable call rate. Depending on the load filtering policy computation algorithm, CPa1 may allocate the received total acceptable call rate among its neighbors, namely, EPa2, CPa2, and CPb1, and notify them about the resulting allocation along with the hotline number and the activation time. CPa2 and CPb1 may perform further allocation among their own neighbors and notify the corresponding proxy servers. This process continues until all edge proxy servers in the network have been informed about the event and have proper load filtering policy configured.

In the above case, the network entity where load filtering policy is

first introduced is the SIP server providing access to the resource that creates the overload situation. In other cases, the network entry point of introducing load filtering policy could also be an entity that hosts this resource. For example, an operator may host an application server that performs 800 number translation services. The application server may itself be a SIP proxy server or a SIP Back-to-Back User Agent (B2BUA). If one of the 800 numbers hosted at the application server creates the overload condition, the load filtering policies can be introduced from the application server and then propagated to other SIP proxy servers in the network.

Case II: a hurricane affects the region covered by CPb2, EPb3 and EPb4. All these three SIP proxy servers are overloaded. The rescue team determines that outbound calls are more valuable than inbound calls in this specific situation. Therefore, EPb3 and EPb4 are configured with load filtering policies to accept more outbound calls than inbound calls. CPb2 may be configured the same way or receive dynamically computed load filtering policies from EPb3 and EPb4. Depending on the load filtering policy computation algorithm, CPb2 may also send out notifications to its outside neighbors, namely CPb1 and CPa2, specifying a limit on the acceptable rate of inbound calls to CPb2's responsible domain. CPb1 and CPa2 may subsequently notify their neighbors about limiting the calls to CPb2's area. The same process could continue until all edge proxy servers are notified and have load filtering policies configured.

Note that this specification does not define the provisioning interface between the party who determines the load filtering policy and the network entry point where the policy is introduced. One of the options for the provisioning interface is the Extensible Markup Language (XML) Configuration Access Protocol (XCAP) [RFC4825].

5.4. Applicable Network Domains

This specification MUST be applied inside a 'Trust Domain'. The concept of a Trust Domain is similar to that defined in [RFC3324]. A Trust Domain for the purpose of SIP load filtering is a set of SIP entities such as SIP proxy servers that are trusted to exchange load filtering policies defined in this specification. In the simplest case, a Trust Domain is a network of SIP entities belonging to a single service provider who deploys it and accurately knows the behaviour of those SIP entities. Such simple Trust Domains may be joined to form larger Trust Domains by bi-lateral agreements between the service providers of the SIP entities.

The key requirement of a Trust Domain for the purpose of SIP load filtering is that the behavior of all SIP entities within a given Trust Domain is known to comply to the following set of

specifications.

- The mechanisms used to secure the communication among SIP entities within the Trust Domain.
- The manner used to determine which SIP entities are part of the Trust Domain.
- That SIP entities in the Trust Domain are compliant to SIP [RFC3261]
- That SIP entities in the Trust Domain are compliant to SIP [RFC6665]
- That SIP entities in the Trust Domain are compliant to this document.
- The agreement on what types of calls can be affected by this SIP load filtering mechanism. For example, call identity condition elements (Section 7.3.1) <one> and <many> might be limited to describe specific domains; <many-tel> and <except-tel> might be limited to describe within certain prefixes.
- The agreement on the destinations to which calls may be redirected when the "redirect" action (Section 7.4) is used. For example, the URI might have to match a given set of domains.

It is important to note that effectiveness of SIP load filtering requires that all neighbors that are possible signaling sources participate and enforce the designated load filtering policies. Otherwise, a single non-conforming neighbor could make the whole filtering efforts useless by pumping in excessive traffic to overload the server. Therefore, the SIP server that distributes load filtering policies needs to take counter-measures towards any non-conforming neighbors. A simple method is to reject excessive requests with 503 (Service Unavailable) response messages as if they were obeying the rate. Considering the rejection costs, a more complicated but fairer method would be to allocate at the overloaded server the same amount of processing to the combination of both normal processing and rejection as the overloaded server would devote to processing requests for a conforming upstream SIP server. These approaches work as long as the total rejection cost does not overwhelm the entire server resources. In addition, SIP servers need to handle message prioritization properly while performing load filtering, which is described in Section 6.8.

6. Load Control Event Package

The SIP load filtering mechanism defines a load control event package for SIP based on [RFC6665].

6.1. Event Package Name

The name of this event package is "load-control". This name is carried in the Event and Allow-Events header, as specified in [RFC6665].

6.2. Event Package Parameters

No package specific event header field parameters are defined for this event package.

6.3. SUBSCRIBE Bodies

This document does not define the content of SUBSCRIBE bodies. Future specifications could define bodies for SUBSCRIBE messages, for example to request specific types of load control event notifications.

A SUBSCRIBE request sent without a body implies the default subscription behavior as specified in Section 6.7.

6.4. SUBSCRIBE Duration

The default expiration time for a subscription to load filtering policy is one hour. Since the desired expiration time may vary significantly for subscriptions among SIP entities with different signaling relationships, the subscribers and notifiers are RECOMMENDED to explicitly negotiate appropriate subscription duration when knowledge about the mutual signaling relationship is available.

6.5. NOTIFY Bodies

The body of a NOTIFY request in this event package contains load filtering policies. The format of the NOTIFY request body MUST be in one of the formats defined in the Accept header field of the SUBSCRIBE request or be the default format, as specified in [RFC6665]. The default data format for the NOTIFY request body of this event package is "application/load-control+xml" (defined in Section 7). This means that when NOTIFY request body exists but no Accept header field is specified in a SUBSCRIBE request, the NOTIFY request body MUST contain "application/load-control+xml" format.

6.6. Notifier Processing of SUBSCRIBE Requests

The notifier accepts a new subscription or updates an existing

subscription upon receiving a valid SUBSCRIBE request.

If the identity of the subscriber sending the SUBSCRIBE request is not allowed to receive load filtering policy, the notifier MUST return a 403 "Forbidden" response.

If none of MIME types specified in the Accept header of the SUBSCRIBE request is supported, the notifier SHOULD return 406 "Not Acceptable" response.

6.7. Notifier Generation of NOTIFY Requests

A notifier MUST send a NOTIFY request with its current load filtering policy to the subscriber upon successfully accepting or refreshing a subscription. If no load filtering policy needs to be distributed when the subscription is received, the notifier SHOULD sent a NOTIFY request without body to the subscriber. The content-type header field of this NOTIFY request MUST indicate the correct body format as if the body were present (e.g., "application/load-control+xml"). Sending this NOTIFY request without body is often the case when a subscription is initiated for the first time, e.g., when a SIP entity is just introduced, because there may be no planned events that require load filtering at that time. A notifier SHOULD generate NOTIFY requests each time the load filtering policy changes, with the maximum notification rate not exceeding values defined in Section 6.10.

6.8. Subscriber Processing of NOTIFY Requests

The subscriber is the load filtering server which enforces load filtering policies received from the notifier. The way subscribers process NOTIFY requests depends on the load filtering policies conveyed in the notifications. Typically, load filtering policies consist of rules specifying actions to be applied to requests matching certain conditions. A subscriber receiving a notification first installs these rules and then enforce corresponding actions on requests matching those conditions, for example, limiting the sending rate of call requests destined for a specific callee.

In the case when load filtering policies specify a future validity, it is possible that when the validity time comes, the subscription to the specific notifier that conveyed the rules has expired. In this case, it is RECOMMENDED that the subscriber re-activate its subscription with the corresponding notifier. Regardless of whether this re-activation of subscription is successful or not, when the validity time is reached, the subscriber SHOULD enforce the corresponding rules.

Upon receipt of a NOTIFY request with a Subscription-State header field containing the value "terminated", the subscription status with the particular notifier will be terminated. Meanwhile, subscribers MUST also terminate previously received load filtering policies from that notifier.

The subscriber SHOULD discard unknown bodies. If the NOTIFY request contains several bodies, none of them being supported, it SHOULD unsubscribe. A NOTIFY request without a body indicates that no load filtering policies need to be updated.

When the subscriber enforces load filtering policies, it 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 the subscriber 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 subscriber will aim to preserve local policy to the fullest extent possible.

- The subscriber SHOULD honor the local policy for prioritizing SIP requests such as policies based on message type, e.g., INVITEs versus requests associated with existing sessions.

- The subscriber SHOULD honor the local policy for prioritizing SIP requests based on the content of the Resource-Priority header (RPH, [RFC4412]). Specific (namespace.value) RPH contents may indicate high priority requests that should be preserved as much as possible during overload. The RPH contents can also indicate a low-priority request that is eligible to be dropped during times of overload.

- The subscriber SHOULD honor the local policy for prioritizing SIP requests relating to emergency calls as identified by the SOS URN [RFC5031] indicating an emergency request.

A local policy can be expected to combine both the SIP request type and the prioritization markings, and SHOULD be honored when overload conditions prevail.

6.9. Handling of Forked Requests

Forking is not applicable when this load control event package mechanism is used within a single-hop distance between neighboring SIP entities. If communication scope of the load control event package mechanism is among multiple hops, forking is not expected to happen either because the subscription request is addressed to a

clearly defined SIP entity. However, in the unlikely case when forking does happen, the load control event package only allows the first potential dialog-establishing message to create a dialog, as specified in Section 5.9 of [RFC6665].

6.10. Rate of Notifications

Rate of notifications is likely not a concern for this local control event package mechanism when it is used in a non-real-time mode for relatively static load filtering policies. Nevertheless, if situation does arise that a rather frequent load filtering policy update is needed, it is RECOMMENDED that the notifier do not generate notifications at a rate higher than once per-second in all cases, in order to avoid the NOTIFY request itself overloading the system.

6.11. State Delta

It is likely that updates to specific load filtering policies are made by changing only part of the policy parameters only (e.g. acceptable request rate or percentage, but not matching identities). This will typically be because the utilization of a resource subject to overload depends upon dynamic unknowns such as holding time and the relative distribution of offered loads over subscribing SIP entities. The updates could originate manually or be determined automatically by an algorithm that dynamically computes the load filtering policies (Section 5.2). Another factor that is usually not known precisely or needs to be computed automatically is the duration of the event requiring load filtering. Therefore it would also be common for the validity to change frequently.

This event package allows the use of state delta as in [RFC6665] to accommodate frequent updates of partial policy parameters. For each NOTIFY transaction in a subscription, a version number that increases by exactly one MUST be included in the NOTIFY request body when the body is present. When the subscriber receives a state delta, it associates the partial updates to the particular policy by matching the appropriate rule id (Section 7.5). If the subscriber receives a NOTIFY request with a version number that is increased by more than one, it knows that it has missed a state delta and needs to ask for a full state snapshot. Therefore, the subscriber ignores that NOTIFY request containing the state delta, and re-sends a SUBSCRIBE request to force a NOTIFY request containing a complete state snapshot.

7. Load Control Document

7.1. Format

A load control document is an XML document that describes the load

filtering policies. It inherits and enhances the common policy document defined in [RFC4745]. A common policy document contains a set of rules. Each rule consists of three parts: conditions, actions and transformations. The conditions part is a set of expressions containing attributes such as identity, domain, and validity time information. Each expression evaluates to TRUE or FALSE. Conditions are matched on "equality" or "greater than" style comparison. There is no regular expression matching. Conditions are evaluated on receipt of an initial SIP request for a dialog or standalone transaction. If a request matches all conditions in a rule set, the action part and the transformation part are consulted to determine the "permission" on how to handle the request. Each action or transformation specifies a positive grant to the policy server to perform the resulting actions. Well-defined mechanism are available for combining actions and transformations obtained from more than one sources.

7.2. Namespace

The namespace URI for elements defined by this specification is a Uniform Resource Namespace (URN) ([RFC2141]), using the namespace identifier 'ietf' defined by [RFC2648] and extended by [RFC3688]. The URN is as follows:

urn:ietf:params:xml:ns:load-control

7.3. Conditions

[RFC4745] defines three condition elements: <identity>, <sphere> and <validity>. In this specification, we define new condition elements and reuse the <validity> element. The <sphere> element is not used.

7.3.1. Call Identity

Since the problem space of this specification is different from that of [RFC4745], the [RFC4745] <identity> element is not sufficient for use with load filtering. First, load filtering may be applied to different identities contained in a request, including identities of both the receiving entity and the sending entity. Second, the importance of authentication varies when different identities of a request are concerned. This specification defines new identity conditions that can accommodate the granularity of specific SIP identity header fields. The requirement for authentication depends on which field is to be matched.

The identity condition for load filtering is specified by the <call-identity> element and its sub-element <sip>. The <sip> element itself contains sub-elements representing SIP sending and receiving

identity header fields: <from>, <to>, <request-uri> and <p-asserted-identity>. All those sub-elements are of an extended form of the [RFC4745] <identity> element. In addition to the sub-elements including <one>, <except>, and <many> in the [RFC4745] <identity> element, the extended form adds two new sub-elements, namely, <many-tel> and <except-tel>, which will be explained later in this section.

The [RFC4745] <one> and <except> elements may contain an "id" attribute, which is the URI of a single entity to be included or excluded in the condition. When used in the <from>, <to>, <request-uri> and <p-asserted-identity> elements, this "id" value is the URI contained in the corresponding SIP header field, i.e., From, To, Request-URI, and P-Asserted-Identity.

When the <call-identity> element contains multiple <sip> sub-elements, the result is combined using logical OR. When the <from>, <to>, <request-uri> and <p-asserted-identity> elements contain multiple <one> or <many> or <many-tel> sub-elements, the result is also combined using logical OR. When the <many> sub-element further contains one or more <except> sub-elements, or when the <many-tel> sub-element further contains one or more <except-tel> sub-elements, the result of each <except> or <except-tel> sub-element is combined using a logical OR, similar to that of the [RFC4745] <identity> element. However, when the <sip> element contains multiple of the <from>, <to>, <request-uri> and <p-asserted-identity> sub-elements, the result is combined using logical AND. This allows the call identity to be specified by multiple fields of a SIP request simultaneously, e.g., both the From and the To header fields.

The following shows an example of the <call-identity> element, which matches call requests whose To header field contains the SIP URI "sip:alice@hotline.example.com", or the 'tel' URI "tel:+1-212-555-1234".

```
<call-identity>
    <sip>
        <to>
            <one id="sip:alice@hotline.example.com"/>
            <one id="tel:+1-212-555-1234"/>
        </to>
    </sip>
</call-identity>
```

Before evaluating call-identity conditions, the subscriber shall convert URIs received in SIP header fields in canonical form as per [RFC3261], except that the phone-context parameter shall not be

removed, if present.

The [RFC4745] <many> and <except> elements may take a "domain" attribute. The "domain" attribute specifies a domain name to be matched by the domain part of the candidate identity. Thus, it allows matching a large and possibly unknown number of entities within a domain. The "domain" attribute works well for SIP URIs.

A URI identifying a SIP user, however, can also be a 'tel' URI. We therefore need a similar way to match a group of 'tel' URIs. According to [RFC3966], there are two forms of 'tel' URIs for global numbers and local numbers, respectively. All phone numbers must be expressed in global form when possible. The global number 'tel' URIs start with a "+". The rest of the numbers are expressed as local numbers, which must be qualified by a "phone-context" parameter. The "phone-context" parameter may be labelled as a global number or any number of its leading digits, or a domain name. Both forms of the 'tel' URI make the resulting URI globally unique.

'Tel' URIs of global numbers can be grouped by prefixes consisting of any number of common leading digits. For example, a prefix formed by a country code or both the country and area code identifies telephone numbers within a country or an area. Since the length of the country and area code for different regions are different, the length of the number prefix also varies. This allows further flexibility such as grouping the numbers into sub-areas within the same area code. 'Tel' URIs of local numbers can be grouped by the value of the "phone-context" parameter.

The <many> and <except> sub-elements in the [RFC4745] <identity> element do not allow additional attributes to be added directly. Redefining behavior of their existing <domain> attribute creates backward-compatibility issues. Therefore, this specification defines the <many-tel> and <except-tel> sub-elements that extend the [RFC4745] <identity> element. Both of them have a "prefix" attribute for grouping 'tel' URIs, similar to the "domain" attribute for grouping SIP URIs in existing <many> and <except> sub-elements. For global numbers, the "prefix" attribute value holds any number of common leading digits, for example, "+1-212" for U.S. phone numbers within area code "212" or "+1-212-854" for the organization with U.S. area code "212" and local prefix "854". For local numbers, the "prefix" attribute value contains the "phone-context" parameter value. It should be noted that visual separators (such as the "-" sign) in 'tel' URIs are not used for URI comparison as per [RFC3966].

The following example shows the use of the "prefix" attribute along with the "domain" attribute. It matches those requests calling to the number "+1-202-999-1234" but are not calling from a "+1-212"

prefix or a SIP From URI domain of "manhattan.example.com".

```
<call-identity>
    <sip>
        <from>
            <many>
                <except domain="manhattan.example.com"/>
            </many>
            <many-tel>
                <except-tel prefix="+1-212"/>
            </many-tel>
        </from>
        <to>
            <one id="tel:+1-202-999-1234"/>
        </to>
    </sip>
</call-identity>
```

7.3.2. Method

The load created on a SIP server depends on the type of initial SIP requests for dialogs or standalone transactions. The <method> element specifies the SIP method to which the load filtering action applies. When this element is not included, the load filtering actions are applicable to all applicable initial requests. These requests include INVITE, MESSAGE, REGISTER, SUBSCRIBE, OPTIONS, and PUBLISH. Non-initial requests, such as ACK, BYE and CANCEL MUST NOT be subjected to load filtering. In addition, SUBSCRIBE requests are not filtered if the event-type header field indicates the event package defined in this specification.

The following example shows the use of the <method> element in the case the filtering actions should be applied to INVITE requests.

```
<method>INVITE</method>
```

7.3.3. Target SIP Entity

A SIP server that performs load filtering may have multiple paths to route call requests matching the same set of call identity elements. In those situations, the SIP load filtering server may desire to take advantage of alternative paths and only apply load filtering actions to matching requests for the next hop SIP entity that originated the corresponding load filtering policy. To achieve that, the SIP load filtering server needs to associate every load filtering policy with its originating SIP entity. The <target-sip-entity> element is defined for that purpose and it contains the URI of the entity that initiated the load filtering policy, which is generally the

corresponding notifier. A notifier MAY include this element as part of the condition of its filtering policy being sent to the subscriber, as below.

<target-sip-entity>sip:biloxi.example.com</target-sip-entity>

When a SIP load filtering server receives a policy with a <target-sip-entity> element, it SHOULD record it and take it into consideration when making load filtering decisions. If the load filtering server receives a load filtering policy that does not contain a <target-sip-entity> element, it MAY still record the URI of the load filtering policy's originator as the <target-sip-entity> information and consider it when making load filtering decisions.

The following are two examples of using the <target-sip-entity> element.

Use case I: the network has user A connected to SIP Proxy 1 (SP1), user B connected to SIP Proxy 3 (SP3), SP1 and SP3 connected via SIP Proxy 2 (SP2), and SP2 connected to an Application Server (AS). Under normal load conditions, a call from A to B is routed along the following path: A-SP1-SP2-AS-SP3-B. The AS provides a non-essential service and can be bypassed in case of overload. Now let's assume that AS is overloaded and sends to SP2 a load filtering policy requesting that 50% of all INVITE requests be dropped. SP2 can maintain AS as the <target-sip-entity> for that policy so that it knows the 50% drop action is only applicable to call requests that must go through AS, without affecting those calls directly routed through SP3 to B.

Use case II: An 800 translation service is installed on two Application Servers, AS1 and AS2. User A is connected to SP1 and calls 800-1234-4529, which is translated by AS1 and AS2 into a regular E.164 number depending on, e.g., the caller's location. SP1 forwards INVITE requests with Request-URI = "800 number" to AS1 or AS2 based on a load balancing strategy. As calls to 800-1234-4529 creates a pre-overload condition in AS1, AS1 sends to SP1 a load filtering policy requesting that 50% of calls towards 800-1234-4529 be rejected. In this case, SP1 can maintain AS1 as the <target-sip-entity> for the rule, and only apply the load filtering policy on incoming requests that are intended to be sent to AS1. Those requests that are sent to AS2, although matching the <call-identity> of the filter, will not be affected.

7.3.4. Validity

A filtering policy is usually associated with a validity period condition. This specification reuses the <validity> element of

[RFC4745], which specifies a period of validity time by pairs of <from> and <until> sub-elements. When multiple time periods are defined, the validity condition is evaluated to TRUE if the current time falls into any of the specified time periods. i.e., it represents a logical OR operation across all validity time periods.

The following example shows a <validity> element specifying a valid period from 12:00 to 15:00 US Eastern Standard Time on 2008-05-31.

```
<validity>
    <from>2008-05-31T12:00:00-05:00</from>
    <until>2008-05-31T15:00:00-05:00</until>
</validity>
```

7.4. Actions

The actions a load filtering server takes on loads matching the load filtering conditions are defined by the <accept> element in the load filtering policy, which includes any one of the three sub-elements <rate>, <percent>, and <win>. The <rate> element denotes an absolute value of the maximum acceptable request rate in requests per second; the <percent> element specifies the relative percentage of incoming requests that should be accepted; the <win> element describes the acceptable window size supplied by the receiver, which is applicable in window-based load filtering. In static load filtering policy configuration scenarios, using the <rate> sub-element is RECOMMENDED because it is hard to enforce the percentage rate or window-based load filtering when incoming load from upstream or reactions from downstream are uncertain. (See [I-D.ietf-soc-overload-control] [RFC6357] for more details on rate-based, loss-based and window-based load control.)

In addition, the <accept> element takes an optional "alt-action" attribute which can be used to explicitly specify the desired action in case a request cannot be processed. The "alt-action" can take one of the following three values: "reject", "redirect" and "drop".

- The "reject" action is the default value for "alt-action". It means that the load filtering server will reject the request with a 503 (Service Unavailable) response message.

- The "redirect" action means redirecting the request to another target. When it is used, an "alt-target" attribute MUST be defined. The "alt-target" specifies one URI or a list of URIs where the request should be redirected. The server sends out the redirect URIs in a 300-class response message.

- The "drop" action means simply ignoring the request without doing

anything, which can in certain cases help save processing capability during overload. For example, when SIP is running over a reliable transport such as TCP, the "drop" action does not send out the rejection response, neither does it close the transport connection. The "drop" action is generally also good at dealing with telephony DOS attacks. However, when running SIP over an unreliable transport such as UDP, using the "drop" action will create message retransmissions that further worsen the possible overload situation. Therefore, any "drop" action applied to an unreliable transport MUST be treated as if it were "reject".

In the following <actions> element example, the server accepts maximum of 100 call requests per second. The remaining calls are redirected to an answering machine.

```
<actions>
    <accept alt-action="redirect" alt-target=
            "sip:answer-machine@example.com">
        <rate>100</rate>
    </accept>
</actions>
```

7.5. Complete Examples

7.5.1. Load Control Document Examples

This section presents two complete examples of load control documents valid with respect to the XML schema defined in Section 8.

The first example assumes that a set of hotlines are set up at "sip:alice@hotline.example.com" and "tel:+1-212-555-1234". The hotlines are activated from 12:00 to 15:00 US Eastern Standard Time on 2008-05-31. The goal is to limit the incoming calls to the hotlines to 100 requests per second. Calls that exceed the rate limit are explicitly rejected.

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy"
            xmlns:lc="urn:ietf:params:xml:ns:load-control"
            version="0" state="full">

    <rule id="f3g44k1">
        <conditions>
            <lc:call-identity>
                <lc:sip>
                    <lc:to>
                        <one id="sip:alice@hotline.example.com"/>
                        <one id="tel:+1-212-555-1234"/>
```

```
                    </lc:to>
                </lc:sip>
            </lc:call-identity>
            <method>INVITE</method>
            <validity>
                <from>2008-05-31T12:00:00-05:00</from>
                <until>2008-05-31T15:00:00-05:00</until>
            </validity>
        </conditions>
        <actions>
            <lc:accept alt-action="reject">
                <lc:rate>100</lc:rate>
            </lc:accept>
        </actions>

    </rule>
</ruleset>
```

The second example considers optimizing server resource usage of a three-day period during the aftermath of a hurricane. Incoming calls to the domain "sandy.example.com" or to call destinations with prefix "+1-212" will be limited to a rate of 100 requests per second, except for those calls originating from a particular rescue team domain "rescue.example.com". Outgoing calls from the hurricane domain or calls within the local domain are never limited. All calls that are throttled due to the rate limit will be forwarded to an answering machine with updated hurricane rescue information.

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy"
    xmlns:lc="urn:ietf:params:xml:ns:load-control"
    version="1" state="full">

    <rule id="f3g44k2">
        <conditions>
            <lc:call-identity>
                <lc:sip>
                    <lc:to>
                        <many domain="sandy.example.com"/>
                        <many-tel prefix="+1-212"/>
                    </lc:to>
                    <lc:from>
                        <many>
                            <except domain="sandy.example.com"/>
                            <except domain="rescue.example.com"/>
                        </many>
                    </lc:from>
```

```
                </lc:sip>
            </lc:call-identity>
            <method>INVITE</method>
            <validity>
                <from>2012-10-25T09:00:00+01:00</from>
                <until>2012-10-28T09:00:00+01:00</until>
            </validity>
        </conditions>
        <actions>
            <lc:accept alt-action="redirect" alt-target=
                    "sip:sandy@update.example.com">
                <lc:rate>100</lc:rate>
            </lc:accept>
        </actions>

    </rule>
</ruleset>
```

Sometimes it may occur that multiple rules in a ruleset define actions that match the same methods, call identity and validity. In those cases, the "first-match-wins" principle is used. For example, in the following ruleset, the first rule requires all calls from the "example.com" domain to be rejected. Even though the rule following that one specifies that calls from "sip:alice@example.com" be redirected to a specific target "sip:eve@example.com", the calls from "sip:alice@example.com" will still be rejected because they have already been matched by the earlier rule.

```
<?xml version="1.0" encoding="UTF-8"?>
<ruleset xmlns="urn:ietf:params:xml:ns:common-policy"
    xmlns:lc="urn:ietf:params:xml:ns:load-control"
    version="1" state="full">

    <rule id="f3g44k3">
        <conditions>
            <lc:call-identity>
                <lc:sip>
                    <lc:from>
                        <many domain="example.com"/>
                    </lc:from>
                </lc:sip>
            </lc:call-identity>
            <method>INVITE</method>
            <validity>
                <from>2013-7-2T09:00:00+01:00</from>
                <until>2013-7-3T09:00:00+01:00</until>
            </validity>
```

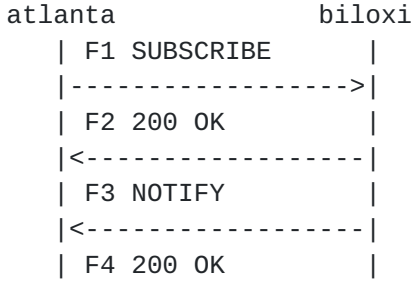
```
            </conditions>
            <actions>
                <lc:accept alt-action="reject">
                    <lc:rate>0</lc:rate>
                </lc:accept>
            </actions>
        </rule>

        <rule id="f3g44k4">
            <conditions>
                <lc:call-identity>
                    <lc:sip>
                        <lc:from>
                            <one id="sip:alice@example.com"/>
                        </lc:from>
                    </lc:sip>
                </lc:call-identity>
                <method>INVITE</method>
                <validity>
                    <from>2013-7-2T09:00:00+01:00</from>
                    <until>2013-7-3T09:00:00+01:00</until>
                </validity>
            </conditions>
            <actions>
                <lc:accept alt-action="redirect" alt-target=
                        "sip:eve@example.com">
                    <lc:rate>0</lc:rate>
                </lc:accept>
            </actions>
        </rule>

    </ruleset>
```

7.5.2. Message Flow Examples

This section presents an example message flow of using the load control event package mechanism defined in this specification.



```
         |------------------>|

      F1 SUBSCRIBE atlanta.example.com -> biloxi.example.com

         SUBSCRIBE sip:biloxi.example.com SIP/2.0
         Via: SIP/2.0/TCP atlanta.example.com;branch=z9hG4bKy7cjbu3
         From: sip:atlanta.example.com;tag=162ab5
         To: sip:biloxi.example.com
         Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
         CSeq: 2012 SUBSCRIBE
         Contact: sip:atlanta.example.com
         Event: load-control
         Max-Forwards: 70
         Accept: application/load-control+xml
         Expires: 3600
         Content-Length: 0

      F2 200 OK   biloxi.example.com -> atlanta.example.com

         SIP/2.0 200 OK
         Via: SIP/2.0/TCP biloxi.example.com;branch=z9hG4bKy7cjbu3
           ;received=192.0.2.1
         To: <sip:biloxi.example.com>;tag=331dc8
         From: <sip:atlanta.example.com>;tag=162ab5
         Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
         CSeq: 2012 SUBSCRIBE
         Expires: 3600
         Contact: sip:biloxi.example.com
         Content-Length: 0

      F3 NOTIFY  biloxi.example.com -> atlanta.example.com

         NOTIFY sip:atlanta.example.com SIP/2.0
         Via: SIP/2.0/TCP biloxi.example.com;branch=z9hG4bKy71g2ks
         From: <sip:biloxi.example.com>;tag=331dc8
         To: <sip:atlanta.example.com>;tag=162ab5
         Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
         Event: load-control
         Subscription-State: active;expires=3599
         Max-Forwards: 70
         CSeq: 1775 NOTIFY
         Contact: sip:biloxi.example.com
         Content-Type: application/load-control+xml
         Content-Length: ...

         [Load Control Document]

      F4 200 OK atlanta.example.com -> biloxi.example.com
```

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP atlanta.example.com;branch=z9hG4bKy71g2ks
  ;received=192.0.2.2
From: <sip:biloxi.example.com>;tag=331dc8
To: <sip:atlanta.example.com>;tag=162ab5
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1775 NOTIFY
Content-Length: 0
```

8. XML Schema Definition for Load Control

This section defines the XML schema for the load control document. It extends the Common Policy schema in [RFC4745] in two ways. Firstly, it defines two mandatory attributes for the <ruleset> element: version and state. The version attribute allows the recipient of the notification to properly order them. Versions start at 0, and increase by one for each new document sent to a subscriber within the same subscription. Versions MUST be representable using a non-negative 32 bit integer. The state attribute indicates whether the document contains a full load filtering policy update, or whether it contains only state delta as partial update. Secondly, it defines new members of the <conditions> and <actions> elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:load-control"
    xmlns:lc="urn:ietf:params:xml:ns:load-control"
    xmlns:cp="urn:ietf:params:xml:ns:common-policy"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributedFormDefault="unqualified">

<xs:import namespace="urn:ietf:params:xml:ns:common-policy"/>

<!-- RULESET -->

<xs:element name="ruleset">
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base="xs:anyType">
        <xs:sequence>
          <xs:element name="rule" type="cp:ruleType"
          minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
```

```
      <xs:attribute name="version" type="xs:integer" use="required"/>
      <xs:attribute name="state" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="full"/>
            <xs:enumeration value="partial"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <!-- CONDITIONS -->

  <!-- CALL IDENTITY -->
  <xs:element name="call-identity" type="lc:call-identity-type"/>

  <!-- CALL IDENTITY TYPE -->
  <xs:complexType name="call-identity-type">
    <xs:choice>
      <xs:element name="sip" type="lc:sip-id-type"/>
      <any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
    </xs:choice>
    <anyAtrribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- SIP ID TYPE -->
  <xs:complexType name="sip-id-type">
    <xs:sequence>
      <element name="from" type="lc:identityType" minOccurs="0"/>
      <element name="to" type="lc:identityType" minOccurs="0"/>
      <element name="request-uri" type="lc:identityType"
      minOccurs="0"/>
      <element name="p-asserted-identity" type="lc:identityType"
      minOccurs="0"/>
      <any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
    </xs:sequence>
    <anyAtrribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- IDENTITY TYPE -->
  <xs:complexType name="identityType">
    <xs:complexContent>
      <xs:restriction base="xs:anyType">
        <xs:choice minOccurs="1" maxOccurs="unbounded">
          <xs:element name="one" type="cp:oneType"/>
```

```
          <xs:element name="many" type="lc:manyType"/>
          <xs:element name="many-tel" type="lc:manyTelType"/>
          <xs:any namespace="##other" processContents="lax"/>
        </xs:choice>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <!-- MANY-TEL TYPE -->
  <xs:complexType name="manyTelType">
    <xs:complexContent>
      <xs:restriction base="xs:anyType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="except-tel" type="lc:exceptTelType"/>
          <xs:any namespace="##other"
          minOccurs="0" processContents="lax"/>
        </xs:choice>
        <xs:attribute name="prefix"
        use="optional" type="xs:string"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <!-- EXCEPT-TEL TYPE -->
  <xs:complexType name="exceptTelType">
    <xs:attribute name="prefix" type="xs:string" use="optional"/>
    <xs:attribute name="id" type="xs:anyURI" use="optional"/>
  </xs:complexType>

  <!-- METHOD -->
  <xs:element name="method" type="lc:method-type"/>

  <!-- METHOD TYPE -->
  <xs:simpleType name="method-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="INVITE"/>
      <xs:enumeration value="MESSAGE"/>
      <xs:enumeration value="REGISTER"/>
      <xs:enumeration value="SUBSCRIBE"/>
      <xs:enumeration value="OPTIONS"/>
      <xs:enumeration value="PUBLISH"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- TARGET SIP ENTITY -->
  <xs:element name="target-sip-entity" type="xs:anyURI" minOccurs="0"/>

  <!-- ACTIONS -->
```

```
   <xs:element name="accept">
     <xs:choice>
       <element name="rate" type="xs:decimal" minOccurs="0"/>
       <element name="win" type="xs:integer" minOccurs="0"/>
       <element name="percent" type="xs:decimal" minOccurs="0"/>
       <any namespace="##other" processContents="lax" minOccurs="0"
       maxOccurs="unbounded"/>
     </xs:choice>
     <xs:attribute name="alt-action" type="xs:string" default="reject"/>
     <xs:attribute name="alt-target" type="lc:alt-target-type"
     use="optional"/>
     <anyAtrribute namespace="##other" processContents="lax"/>
   </xs:element>

   <!-- ALT TARGET TYPE -->
   <xs:simpleType name="alt-target-type">
     <xs:list itemType="xs:anyURI"/>
   </xs:simpleType>

   </xs:schema>
```

9. Related Work

9.1. Relationship with Load Filtering in PSTN

It is known that existing PSTN network also uses a load filtering mechanism to prevent overload and the filtering policy configuration is done manually except in specific cases when the Intelligent Network architecture is used [Q.1248.2][E.412]. This specification defines a load filtering mechanism based on the SIP event notification framework that allows automated filtering policy distribution in suitable environments.

There are control messages associated with PSTN overload control which would specify an outgoing control list, call gap duration and control duration [Q.1248.2][E.412]. These items could be roughly correlated to the identity, action and time fields of the SIP load filtering policy defined in this specification. However, the load filtering policy defined in this specification is much more generic and flexible as opposed to its PSTN counterpart.

Firstly, PSTN load filtering only applies to telephone numbers. The identity element of SIP load filtering policy allows both SIP URI and telephone numbers (through 'tel' URI) to be specified. These identities can be arbitrarily grouped by SIP domains or any number of leading prefix of the telephone numbers.

Secondly, the PSTN load filtering action is usually limited to call gapping. The action field in SIP load filtering policy allows more flexible possibilities such as rate throttle and others.

Thirdly, the duration field in PSTN load filtering specifies a value in seconds for the load filtering duration only, and the allowed values are mapped into a value set. The time field in SIP load filtering policy may specify not only a duration, but also a future activation time which could be especially useful for automating load filtering for predictable overloads.

PSTN load filtering can be performed in both edge switches and transit switches; SIP load filtering can also be applied in both edge proxy servers and core proxy servers, and even in capable user agents.

PSTN load filtering also has special accommodation for High Probability of Completion (HPC) calls, which would be similar to calls designated by the SIP Resource Priority Headers [RFC4412]. SIP load filtering mechanism also allows prioritizing the treatment of these calls by specifying favorable actions for them.

PSTN load filtering also provides administrative option for routing failed call attempts to either a reorder tone [E.300SerSup3] indicating overload conditions, or a special recorded announcement. Similar capability can be provided in SIP load filtering mechanism by specifying appropriate "alt-action" attribute in the SIP load filtering action field.

9.2. Relationship with Other IETF SIP Overload Control Efforts

The load filtering policies in this specification consist of identity, action and time. The identity can range from a single specific user to an arbitrary user aggregate, domains or areas. The user can be identified by either the source or the destination. When the user is identified by the source and a favorable action is specified, the result is to some extent similar to identifying a priority user based on authorized Resource Priority Headers [RFC4412] in the requests. Specifying a source user identity with an unfavorable action would cause an effect to some extent similar to an inverse SIP resource priority mechanism.

The load filtering policy defined in this specification is generic and expected to be applicable not only to the load filtering mechanism but also to the feedback overload control mechanism in [I-D.ietf-soc-overload-control]. In particular, both mechanisms could use specific or wildcard identities for load control and could share well-known load control actions. The time duration field in

the load filtering policy could also be used in both mechanisms. As mentioned in Section 1, the load filtering policy distribution mechanism and the feedback overload control mechanism address complementary areas in the overload control problem space. Load filtering is more proactive and focuses on distributing filtering policies towards the source of the traffic; the hop-by-hop feedback-based approach is reactive and targets more at traffic already accepted in the network. Therefore, they could also make different use of the generic load filtering policy components. For example, the load filtering mechanism may use the time field in the filtering policy to specify not only a control duration but also a future activation time to accommodate a predicable overload such as the one caused by Mother's Day greetings or a viewer-voting program; the feedback-based control might not need to use the time field or might use the time field to specify an immediate load control duration.

10. Discussion of this specification meeting the requirements of RFC5390

This section evaluates whether the load control event package mechanism defined in this specification satisfies various SIP overload control requirements set forth by RFC5390 [RFC5390]. Not all RFC5390 requirements are found applicable due to the scope of this specification. Therefore, we categorize the assessment results into Yes (meet the requirement), P/A (Partially Applicable), No (must be used in conjunction with another mechanism to meet the requirement), and N/A (Not Applicable).

> REQ 1: The overload mechanism shall strive to maintain the overall useful throughput (taking into consideration the quality-of-service needs of the using applications) of a SIP server at reasonable levels, even when the incoming load on the network is far in excess of its capacity. The overall throughput under load is the ultimate measure of the value of an overload control mechanism.

P/A. The goal of the load filtering is to prevent overload or maintain overall goodput during the time of overload, but it is dependent on the advance predictions of the load. If the predictions are incorrect, in either direction, the effectiveness of the mechanism will be affected.

> REQ 2: When a single network element fails, goes into overload, or suffers from reduced processing capacity, the mechanism should strive to limit the impact of this on other elements in the network. This helps to prevent a small-scale failure from becoming a widespread outage.

N/A if load filtering policies are installed in advance and do not change during the potential overload period. P/A if load filtering policies are dynamically adjusted. The algorithm to dynamically compute load filtering policies is outside the scope of this specification, while the distribution of the updated filtering policies uses the event package mechanism of this specification.

> REQ 3: The mechanism should seek to minimize the amount of configuration required in order to work. For example, it is better to avoid needing to configure a server with its SIP message throughput, as these kinds of quantities are hard to determine.

No. This mechanism is entirely dependent on advance configuration, based on advance knowledge. In order to satisfy Req 3, it should be used in conjunction with other mechanisms which are not based on advance configuration.

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

No. This mechanism is entirely dependent on the participation of all possible neighbors. In order to satisfy Req 4, it should be used in conjunction with other mechanisms, some of which are described in Section 5.4.

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

No. This mechanism is entirely dependent on the non-malicious participation of all possible neighbors. In order to satisfy Req 5, it should be used in conjunction with other mechanisms, some of which are described in Section 5.4.

> REQ 6: When overload is signaled by means of a specific message, the message must clearly indicate that it is being sent because of overload, as opposed to other, non overload-based failure conditions. This requirement is meant to avoid some of the

problems that have arisen from the reuse of the 503 response code for multiple purposes. Of course, overload is also signaled by lack of response to requests. This requirement applies only to explicit overload signals.

N/A. This mechanism signals anticipated overload, not actual overload. However the signals in this mechanism are not used for any other purpose.

REQ 7: The mechanism shall provide a way for an element to throttle the amount of traffic it receives from an upstream element. This throttling shall be graded so that it is not all-or-nothing as with the current 503 mechanism. This recognizes the fact that "overload" is not a binary state and that there are degrees of overload.

Yes. This event package allows rate/loss/window-based overload control options as discussed in Section 7.4.

REQ 8: The mechanism shall ensure that, when a request was not processed successfully due to overload (or failure) of a downstream element, the request will not be retried on another element that is also overloaded or whose status is unknown. This requirement derives from REQ 1.

N/A to the load control event package mechanism itself.

REQ 9: That a request has been rejected from an overloaded element shall not unduly restrict the ability of that request to be submitted to and processed by an element that is not overloaded. This requirement derives from REQ 1.

Yes. For example, load filtering policy [Section 5.1] allows the inclusion of alternative forwarding destinations for rejected requests.

REQ 10: The mechanism should support servers that receive requests from a large number of different upstream elements, where the set of upstream elements is not enumerable.

No. Because this mechanism requires advance configuration of specifically identified neighbors, it does not support environments where the number and identity of the upstream neighbors are not known in advance. In order to satisfy Req 10, it should be used in conjunction with other mechanisms.

REQ 11: The mechanism should support servers that receive requests from a finite set of upstream elements, where the set of upstream

elements is enumerable.

Yes. See also answer to REQ 10.

REQ 12: The mechanism should work between servers in different domains.

Yes. The load control event package mechanism is not limited by domain boundaries. However, it is likely more applicable in intra-domain scenarios than in inter-domain scenarios due to security and other concerns (See also Section 5.4).

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

P/A. This mechanism does not specifically address the prioritizing of work during times of overload. But it does not preclude any particular local policy.

REQ 14: The mechanism should provide unambiguous directions to clients on when they should retry a request and when they should not. This especially applies to TCP connection establishment and SIP registrations, in order to mitigate against avalanche restart.

N/A to the load control event package mechanism itself.

REQ 15: In cases where a network element fails, is so overloaded that it cannot process messages, or cannot communicate due to a network failure or network partition, it will not be able to provide explicit indications of the nature of the failure or its levels of congestion. The mechanism must properly function in these cases.

P/A. Because the load filtering policies are provisioned in advance, they are not affected by the overload or failure of other network elements. But, on the other hand, they may not, in those cases, be able to protect the overloaded network elements (see Req 1).

REQ 16: The mechanism should attempt to minimize the overhead of the overload control messaging.

Yes. The standardized SIP event package mechanism [RFC6665] is used.

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

P/A. This mechanism does provide a potential avenue for malicious attacks. Therefore the security mechanisms for SIP event packages in general [RFC6665] and of section 10 of this specification should be used.

REQ 18: The overload mechanism should be unambiguous about whether a load indication applies to a specific IP address, host, or URI, so that an upstream element can determine the load of the entity to which a request is to be sent.

Yes. The identity of load indication is covered in the load filtering policy format definition in Section 5.1.

REQ 19: The specification for the overload mechanism should give guidance on which message types might be desirable to process over others during times of overload, based on SIP-specific considerations. For example, it may be more beneficial to process a SUBSCRIBE refresh with Expires of zero than a SUBSCRIBE refresh with a non-zero expiration (since the former reduces the overall amount of load on the element), or to process re-INVITEs over new INVITEs.

N/A to the load control event package mechanism itself.

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

No. This mechanism is entirely dependent on the participation of all possible neighbors. In order to satisfy Req 20, it should be used in conjunction with other mechanisms, some of which are described in Section 5.4.

REQ 21: The overload mechanism should ensure that the system remains stable. When the offered load drops from above the overall capacity of the network to below the overall capacity, the throughput should stabilize and become equal to the offered load.

N/A to the load control event package mechanism itself.

REQ 22: It must be possible to disable the reporting of load information towards upstream targets based on the identity of those targets. This allows a domain administrator who considers the load of their elements to be sensitive information, to

restrict access to that information. Of course, in such cases, there is no expectation that the overload mechanism itself will help prevent overload from that upstream target.

N/A to the load control event package mechanism itself.

REQ 23: It must be possible for the overload mechanism to work in cases where there is a load balancer in front of a farm of proxies.

Yes. The load control event package mechanism does not preclude its use in a scenario with server farms.

11. Security Considerations

Two aspects of security considerations arise from this specification. One is the SIP event notification framework-based load filtering policy distribution mechanism, the other is the load filtering policy enforcement mechanism.

Security considerations for SIP event package mechanisms are covered in Section 6 of [RFC6665]. A particularly relevant security concern for this event package is that if the notifiers can be spoofed, attackers can send fake notifications asking subscribers to throttle all traffic, leading to Denial-of-Service attacks. Therefore, this SIP load filtering mechanism MUST be used in a Trust Domain (Section 5.4). But if a legitimate notifier in the Trust Domain is itself compromised, additional mechanisms will be needed to detect the attack.

Security considerations for load filtering policy enforcement depends very much on the contents of the policy. This specification defines possible match of the following SIP header fields in a load filtering policy: <from>, <to>, <request-uri> and <p-asserted-identity>. The exact requirement to authenticate and authorize these fields is up to the service provider. In general, if the identity field represents the source of the request, it SHOULD be authenticated and authorized; if the identity field represents the destination of the request, the authentication and authorization is optional.

In addition, there could be one specific type of attack around the use the "redirect" action (Section 7.4). Assuming a number of SIP proxy servers in a Trust Domain are using UDP, and configured to get their policies from a central server. An attacker spoofs the central server's address to send a number of NOTIFY bodies telling the proxy servers to redirect all calls to victim@outside-of-trust-domain.com. The proxy servers then redirect all calls to victim, who is then DoSed off of the Internet. To address this type of threat, this

document requires that a Trust Domain agrees on what types of calls can be affected as well as on the destinations to which calls may be redirected, as in Section 5.4.

12. IANA Considerations

This specification registers a SIP event package, a new MIME type, a new XML namespace, and a new XML schema.

12.1. Load Control Event Package Registration

This section registers an event package based on the registration procedures defined in [RFC6665].

Package name: load-control

Type: package

Published specification: This specification

Person to contact: Charles Shen, charles@cs.columbia.edu

12.2. application/load-control+xml MIME Registration

This section registers a new MIME type based on the procedures defined in [RFC6838] and guidelines in [RFC3023].

MIME media type name: application

MIME subtype name: load-control+xml

Mandatory parameters: none

Optional parameters: Same as charset parameter application/xml in [RFC3023]

Encoding considerations: Same as encoding considerations of application/xml in [RFC3023]

Security considerations: See Section 10 of [RFC3023] and Section 11 of this specification

Interoperability considerations: None

Published Specification: This specification

Applications which use this media type: load control of SIP entities

Additional information:

Magic number: None

File extension: .xml

Macintosh file type code: 'TEXT'

Personal and email address for further information:

Charles Shen, charles@cs.columbia.edu

Intended usage: COMMON

Author/Change Controller: IETF SOC Working Group <sip-overload@ietf.org>, as designated by the IESG <iesg@ietf.org>

12.3. URN Sub-Namespace Registration

This section registers a new XML namespace, as per the guidelines in [RFC3688]

URI: The URI for this namespace is

urn:ietf:params:xml:ns:load-control

Registrant Contact: IETF SOC Working Group <sip-overload@ietf.org>, as designated by the IESG <iesg@ietf.org>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
             "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
    content="text/html;charset=iso-8859-1"/>
  <title>SIP Load Control Namespace</title>
</head>
<body>
  <h1>Namespace for SIP Load Control</h1>
  <h2>urn:ietf:params:xml:ns:load-control</h2>
  <p>See <a href="http://www.rfc-editor.org/rfc/rfc[?].txt">
```

```
      RFC[?]</a>.</p>
   </body>
   </html>
   END
```

12.4. Load Control Schema Registration

URI: urn:ietf:params:xml:schema:load-control

Registrant Contact: IETF SOC working group, Charles Shen (charles@cs.columbia.edu).

XML: the XML schema to be registered is contained in Section 8.

Its first line is

<?xml version="1.0" encoding="UTF-8"?>

and its last line is

</xs:schema>

13. Acknowledgements

The authors would like to thank Richard Barnes, Bruno Chatras, Martin Dolly, Keith Drage, Ashutosh Dutta, Janet Gunn, Vijay Gurbani, Volker Hilt, Geoff Hunt, Carolyn Johnson, Hadriel Kaplan, Paul Kyzivat, Pearl Liang, Salvatore Loreto, Timothy Moran, Eric Noel, Parthasarathi R, Adam Roach, Dan Romascanu, Shida Schubert, Robert Sparks, Phil Williams and other members of the SOC and SIPPING working group for many helpful comments. In particular, Bruno Chatras proposed the <method> and <target-sip-entity> condition elements along with many other text improvements. Janet Gunn provided detailed text suggestions including Section 10. Eric Noel suggested clarification on load filtering policy distribution initialization process. Shida Schubert made many suggestions such as terminology usage. Phil Williams suggested adding support for delta updates. Ashutosh Dutta gave pointers to PSTN references. Adam Roach suggested RFC6665-related and other helpful clarifications. Richard Barnes made many suggestions such as referencing the Trust Domain concept of RFC3324, the use of a separate element for 'tel' URI grouping and addressing the "redirect" action security threat.

14. References

14.1. Normative References

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

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004.

[RFC4745] Schulzrinne, H., Tschofenig, H., Morris, J., Cuellar, J., Polk, J., and J. Rosenberg, "Common Policy: A Document Format for Expressing Privacy Preferences", RFC 4745, February 2007.

[RFC6665] Roach, A., "SIP-Specific Event Notification", RFC 6665, July 2012.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

14.2. Informative References

[E.300SerSup3]
ITU-T, , "North American Precise Audible Tone Plan", E.300 Series Supplement 3 , November 1988.

[E.412] ITU-T, , "Network Management Controls", E.412-2003 , January 2003.

[I-D.ietf-soc-overload-control]
Gurbani, V., Hilt, V., and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control", draft-ietf-soc-overload-control-13 (work in progress), May 2013.

[Q.1248.2]
ITU-T, , "Interface Recommendation for Intelligent Network Capability Set4:SCF-SSF interface", Q.1248.2 , July 2001.

[RFC2648] Moats, R., "A URN Namespace for IETF Documents", RFC 2648, August 1999.

[RFC3324] Watson, M., "Short Term Requirements for Network Asserted Identity", RFC 3324, November 2002.

[RFC4412] Schulzrinne, H. and J. Polk, "Communications Resource Priority for the Session Initiation Protocol (SIP)", RFC 4412, February 2006.

[RFC4825] Rosenberg, J., "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", RFC 4825, May 2007.

[RFC5031] Schulzrinne, H., "A Uniform Resource Name (URN) for Emergency and Other Well-Known Services", RFC 5031, January 2008.

[RFC5390] Rosenberg, J., "Requirements for Management of Overload in the Session Initiation Protocol", RFC 5390, December 2008.

[RFC6357] Hilt, V., Noel, E., Shen, C., and A. Abdelal, "Design Considerations for Session Initiation Protocol (SIP) Overload Control", RFC 6357, August 2011.

Authors' Addresses

Charles Shen
Columbia University
Department of Computer Science
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA

Phone: +1 212 854 3109
Email: charles@cs.columbia.edu

Henning Schulzrinne
Columbia University
Department of Computer Science
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA

Phone: +1 212 939 7004
Email: schulzrinne@cs.columbia.edu

Arata Koike
NTT Service Integration Labs
3-9-11 Midori-cho Musashino-shi
Tokyo 184-0013
Japan

Phone: +81 422 59 6099
Email: koike.arata@lab.ntt.co.jp