IETF SOC Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 13, 2012

C. Shen
AT&T
H. Schulzrinne
Columbia U.
A. Koike
NTT

January 10, 2012

A Session Initiation Protocol (SIP) Load Control Event Package
draft-ietf-soc-load-control-event-package-02.txt

Abstract

We define a load control event package for the Session Initiation Protocol (SIP). It allows SIP servers to distribute load filters to other SIP servers in the network. The load filters 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

Copyright Notice

Table of Contents

1. Introduction

Proper functioning of Session Initiation Protocol (SIP) [RFC3265] 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.

There are three common examples of legitimate short-term increases in call volumes. Viewer-voting TV shows or ticket giveaways may generate millions of calls within a few minutes. Call volume may also spike during special holidays such as New Year's Day and Mother's Day. Finally, callers may want to reach friends and family in natural disaster areas such as those affected by earthquakes. When possible, only calls traversing overloaded servers should be throttled under those 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. However, feedback control is usually ineffective for overload of more general purpose SIP edge proxy servers. 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 mechanism, called load filtering. Network operators create load filters that indicate that calls to specific destinations or from specific sources should be rate-limited or randomly dropped. These load filters are then distributed to SIP servers and possibly user agents 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 user agent clients as possible.

Performing SIP load filtering requires three components: load filter

format, load filter computation method, and load filter distribution mechanism. This specification addresses two of these three components. The load filter format is defined in a SIP load control event package, while the load filter distribution mechanism is built upon the existing SIP event framework. The remaining component, load filter computation method, depends heavily on the actual network topology and service provider policies. Therefore it is out of scope of this specification.

It is helpful to clarify two aspects regarding some terminology used in this specification. Firstly, although the SIP load filtering mechanism is motivated by the overload control problem, which is why this specification refers extensively to other parallel SIP overload control related efforts, the applicability of 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 SIP "load control event package", instead of a narrower term "overload control event package". Secondly, since we are describing a specific control mechanism based on filtering, the term "load control" in this specification is used inter-changeably with the term "load filtering" unless when associated with other explicit context. This specification, however, does not preclude the load control document defined here (Section 6) to be extended in the future for other forms of control as appropriate.

The rest of this specification is structured as follows: we begin by listing the design requirements for this work in Section 3. We then give an overview of load filtering operation in Section 4. The load control event package for filter distribution is detailed in Section 5. The load filter format is defined in the two sections that follow, with Section 6 introducing the XML document for load control and Section 7 listing the associated schema. Section 8 relates this work to corresponding mechanisms in PSTN and other IETF efforts addressing SIP load control. Section 9 evaluates whether this specification meets the SIP overload control requirements set forth by RFC5390 [RFC5390]. Finally, Section 10 presents security considerations and Section 11 provides IANA considerations.

2. Requirements Notation

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. 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 filter 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 call-in events, the load filter should specify during what time period it is to be applied, so that the information can be distributed ahead of time.
- For destination-specific overload situations, the load filter needs to be able to describe the callee.
- To address accidental and intentional high-volume call generators, the load filter should allow to specify the caller.
- Caller and callee need to be specified as both SIP URIs and 'Tel' URIs[RFC3966].
- For telephone numbers, it should be possible to specify prefixes which allow control over limited regionally-focused overloads.
- The solution should draw upon experiences from related PSTN mechanisms where applicable.
- The solution should be extensible to meet future needs.

4. SIP Load Filtering Overview

4.1. Filter Format

A load filter contains both conditions and actions. Filter conditions include the identities of the targets to be controlled. For example, there are two typical resource limits in a possible overload situation, i.e., human destination limits (N number of call takers) and proxy capacity limits. The control targets in these two cases can be the specific callee numbers or the destination domains corresponding to the overload. Filter conditions also indicate the period of time during which the control should be activated, and the specific message type to be controlled, e.g., the INVITE message of a SIP session. Filter actions describe the desired control functions such as limiting the request rate below a certain level. Detailed formats of filter conditions and actions are defined in Section 6.

4.2. Filter Computation

Load filter computation needs to take into consideration information such as the 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 loads are allocated in a way which both prevents overload and minimizes the likelihood of network resource under-utilization. In some cases, in order to better utilize system resources, it may be preferable to employ a dynamic load computation algorithm which adapts to current network status, rather than using a purely static mechanism. The load filter computation algorithm is out of scope of this specification.

4.3. Filter Distribution

For load filter distribution, this specification defines the SIP event package for load control, which is an "instantiation" of the generic SIP events framework [RFC3265]. The SIP events framework provides an existing method for SIP entities to subscribe to and receive notifications when certain events have occurred. Such a framework forms a scalable event distribution architecture that suits our needs. This specification also defines the XML schema of a load control document (Section 6), which is used to encode load filtering rules.

In order for load filters to be properly distributed, each SIP proxy server in the network is required to subscribe to the load control event package from all its outgoing signaling neighbors, known as notifiers (Section 5.6). Subscription is initiated and maintained during normal server operation. Signaling neighbors are defined 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 subscribes to A. Subscription of neighboring SIP entities needs to be persistent so that they are in place independently of any specific load filtering events. Key to this is the fact that notification following initial subscription includes an empty message body if no events are configured (Section 5.7), and that the subscription needs to be refreshed periodically to make it persistent, as described in Section 3.1.6 and Section 3.1.4.2 of [RFC3265]. The notifier will send a notification with its current control policy to its subscribers each time a new subscription or a subscription refreshing is accepted (Section 5.7). The same subscription dialog can also be used to convey policies for multiple different load filtering events in a set of rules (Section 6.1).

```
      +-----------+   +-----------+   +-----------+   +-----------+
      |           |   |           |   |           |   |           |
      |   EPa1    |   |   EPa2    |   |   EPa3    |   |   EPa4    |
      |           |   |           |   |           |   |           |
```

```
      +-----------+   +-----------+   +-----------+   +-----------+
              \         /                    \          /
               \       /                      \        /
                \     /                        \      /
              +-----------+                  +-----------+
              |           |                  |           |
              |   CPa1    |------------------|   CPa2    |
              |           |                  |           |
              +-----------+                  +-----------+
                    |                              |
      Service       |                              |
      Provider A    |                              |
                    |                              |
     =================================================================
                    |                              |
      Service       |                              |
      Provider B    |                              |
                    |                              |
              +-----------+                  +-----------+
              |           |                  |           |
              |   CPb1    |------------------|   CPb2    |
              |           |                  |           |
              +-----------+                  +-----------+
                /      \                        /     \
               /        \                      /       \
              /          \                    /         \
      +-----------+   +-----------+   +-----------+   +-----------+
      |           |   |           |   |           |   |           |
      |   EPb1    |   |   EPb2    |   |   EPb3    |   |   EPb4    |
      |           |   |           |   |           |   |           |
      +-----------+   +-----------+   +-----------+   +-----------+
```

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

We use the example architecture shown in Figure 1 to illustrate load filter distribution based on the SIP load control event package. 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. At the 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 progressively learning from the singling messages sent and received. Assuming all signaling relationship in Figure 1 is 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 filter 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 filter computation algorithm, CPa1 may allocate the received total acceptable 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 filter configured.

Case II: an earthquake affects the region covered by CPb2, EPb3 and EPb4. All the three 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 filters to accept more outbound calls than inbound calls. CPb2 may be configured the same way or receive dynamically computed filters from EPb3 and EPb4. Depending on the filter 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 filters configured.

In the above two cases, the network entity where load filtering policy is first introduced is the SIP server to be protected. In other cases, the network entry point of load filtering policy could also be an entity that the protected SIP server is connected to. For example, an operator may host an application server that performs 800 number translation services. The application server may itself be a SIP proxy 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 propogated to other SIP proxy servers in the network.

Note that this specification does not define the provisioning interface between the party who determines the load control 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].

4.4. Applicability in Different Network Environments

SIP load filtering is more effective when the filters can be pushed to the proximity of signaling sources. But even if only part of the signaling path towards the signaling source could be covered, use of this mechanism can still be beneficial. In fact, due to possibly sophisticated call routing and security concerns, trying to apply automated load filter distribution in the entire inter-domain network path could get extremely complicated and be unrealistic.

The scenarios where this mechanism could be most useful are environments consisting of servers with secure and trust relationship and with relatively straightforward routing configuration known to the filter computation algorithm. These scenarios may include intra-domain environments such as those inside a service provider or enterprise domain; inter-domain environments such as where enterprise connecting to a few service providers or between service providers with manageable routing configurations.

Another important aspect that affects the applicability of SIP load filtering is that all possible signaling source neighbors need to participate and enforce the designated filter. Otherwise, a single non-conforming neighbor could make the whole control efforts useless by pumping in excessive traffic to overload the server. Therefore, the SIP server that initiates the filter needs to take counter-measures towards any non-conforming neighbors. A simple policy is to reject excessive requests with 500 responses as if they were obeying the rate. Considering the rejection costs, a more complicated but fairer policy 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, whatever the actual policy is, SIP servers SHOULD honor the Resource-Priority Header (RPH) [RFC4412] when processing messages. The RPH contents may indicate high priority requests that should be preserved as much as possible, or low priority requests that could be dropped during overload. SIP request rejection and message prioritization at an overloaded server are also discussed in Section 5.1 of [I-D.ietf-soc-overload-control] and Section 12 of [RFC6357].

5. Load Control Event Package

The SIP load filtering mechanism uses the SIP event package for load control. This section defines details of the SIP event package for load control according to [RFC3265].

5.1. 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 [RFC3265].

5.2. Event Package Parameters

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

5.3. SUBSCRIBE Bodies

The effectiveness of SIP load filtering relies on the scope of distribution and installation of the control policies in the network. Since wide distribution of control policies is desirable, subscribers SHOULD try to subscribe to all those notifiers with which they have regular signaling exchanges, although not all such notifiers may permit such a subscription.

A SUBSCRIBE request for the SIP load control event package MAY contain a body to filter the requested load control event notification. For example, a subscriber may be interested in some specific types of load control policy only. The details of the subscription filter specification are not yet defined.

A SUBSCRIBE request sent without a body implies the default subscription behavior as specified in Section 5.7.

5.4. SUBSCRIBE Duration

The default expiration time for a subscription to load control 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 durations when knowledge about the mutual signaling relationship is available.

5.5. NOTIFY Bodies

The body of a NOTIFY request in this event package contains load

control policy. As specified in [RFC3265], the format of the NOTIFY body MUST be in one of the formats defined in the Accept header field of the SUBSCRIBE request or be the default format. The default data format for the NOTIFY body of this event package is "application/load-control+xml" (defined in Section 6). This means that if no Accept header field is specified to a SUBSCRIBE request, the NOTIFY request will contain a body in the "application/load-control+xml" format. If the Accept header field is present, it MUST include "application/load-control+xml" and MAY include any other types.

5.6. Notifier Processing of SUBSCRIBE Requests

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 control policy, the notifier MUST return a 403 "Forbidden" response.

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

5.7. Notifier Generation of NOTIFY Requests

Following [RFC3265] specification, a notifier MUST send a NOTIFY with its current load control policy to the subscriber upon successfully accepting or refreshing a subscription. If no applicable restriction is active when the subscription request is received, an empty message body is attached to the NOTIFY request. This 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 configured at that time. A notifier SHOULD generate NOTIFY requests each time the load control policy changes, with the maximum notification rate not exceeding values defined in Section 5.10.

5.8. Subscriber Processing of NOTIFY Requests

The way subscribers process NOTIFY requests depends on the contents of the notifications. Typically, a load control notification consists of rules that should be applied to requests matching certain identities. A subscriber receiving the notification first installs these rules and then filter incoming requests to enforce actions on appropriate requests, for example, limiting the sending rate of call requests destined for a specific SIP entity.

In the case when load control rules specify a future validity time, 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. However, subscribers SHOULD NOT change previously received load control policies from that notifier because of this change in subscription status, unless it has other specific reasons to do so. Modifications of existing load control policies at the subscriber is performed after directly receiving notifications containing updated load control policies.

The subscriber SHALL discard unknown bodies. If the NOTIFY request contains several bodies, none of them being supported, it SHOULD unsubscribe. A NOTIFY request that does not contain a body MUST be ignored.

5.9. Handling of Forked Requests

Forking is not applicable when the load control event package is used within a single-hop distance between neighboring SIP entities. If the communication scope of the load control event package 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 4.4.9 of [RFC3265].

5.10. Rate of Notifications

Rate of notifications is likely not a concern for this event package when it is used in a non-real-time mode for relatively static load control policies. Nevertheless, if situation does arise that a rather frequent load control policy update is needed, it is RECOMMENDED that the notifier does 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.

5.11. State Delta

It is likely that updates to specific load control events are made by changing the control restriction parameter information only (e.g. rate, percent), but not other rule elements, such as call-identity.

This will typically be because the utilisation 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 a dynamic filter computation algorithm (Section 4.2). Another factor usually not known precisely or is computed automatically is the validity duration of the load control event. Therefore it would also be common for the validity to change frequently.

This event package allows the use of state delta to accommodate frequent updates of partial rule parameters. As in [RFC3265], a version number that increases by exactly one is included in the NOTIFY body for each NOTIFY transaction in a subscription. When the subscriber receives a state delta, it associates the partial updates to the particular rules by matching the appropriate rule id (Section 6.5). If the subscriber receives a NOTIFY that has a version number that is increased by more than one, it knows that it has missed a state delta. The subscriber then keeps the version number, ignores the NOTIFY request containing the state delta, and re-sends a SUBSCRIBE to force a NOTIFY containing a complete state snapshot.

5.12. State Agents

The load control policy can be directly generated by concerned SIP entities distributed in the network. Alternatively, qualified state agents external to the SIP entities MAY be defined to take charge of determining load control policies.

6. Load Control Document

6.1. Format

A load control document is an XML document that 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.

6.2. 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

6.3. Conditions

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

6.3.1. 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 control. First, load control 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 control 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>, each is of the same type as the <identity> element in [RFC4745]. Therefore, they also inherit the sub-elements of the <identity> element, including <one>, <except>, and <many>.

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>, <except>, or <many> sub-elements, the result is also combined using logical OR, similar to that of the <identity> element in [RFC4745]. 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.

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

This example 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".

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 is also variable. 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.

To include the two forms of 'tel' URI grouping in the <many> and <except> elements, one approach is to add a new attribute similar to the "domain" attribute. In this specification, we decided on a simpler approach. There are basically two types of grouping attribute values for both SIP URIs and 'tel' URIs: domain name and number prefix starting with "+". Both of them can be expressed as strings. Therefore, we re-interpret the existing "domain" attribute of the <many> and <except> elements to allow it to contain both types of grouping attribute values. In particular, when the "domain" attribute value starts with "+", it denotes a number prefix, otherwise, the value denotes a domain name. Note that the tradeoff of this simpler approach is the overlap in matching different types of URIs. Specifically, a domain name in the "domain" attribute could be matched by both a SIP URI with that domain name and a local number 'tel' URI containing the same domain name in the "phone-context". On the other hand, a number prefix in the "domain" attribute could be matched by both global number 'tel' URIs starting with those leading digits, and local number 'tel' URIs having the same prefix in the "phone-context" parameter. These overlap situations would not be a big problem because of two reasons. First, when the "phone-context" coincides with the SIP domain name or the global number prefix, it is usually the case that the related phone numbers indeed belong to the same domain or the same area, which means the overlap is not inappropriate. Second, use of the local number 'tel' URI in practice is expected to be rare. As a result, the chance of such overlap happening is very small.

The following example shows the use of the re-interpreted "domain" attribute.

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

```
                   </sip>
               </call-identity>
```

This example 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".

6.3.2. Validity

A rule 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>
```

6.3.3. Method

The load created on a SIP server depends on the type of an initial SIP request for a dialog or standalone transaction. The <method> element specifies the SIP method to which a particular action applies. When this element is not included, the rule actions are applicable to all initial methods.

The following example shows the use of the <method> element.

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

6.4. Actions

As [RFC4745] specified, conditions form the 'if'-part of rules, while actions and transformations form the 'then'-part. Transformations are not used in the load control document. The actions for load control are defined by the <accept> element, which takes 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 control. In static load filter configuration scenarios, using the <rate> sub-element is RECOMMENDED because it is hard to enforce the percentage rate or window-based control when the incoming load from upstream or the 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 accepted. The possible "alt-action" values are "drop" for simple drop, "reject" for explicit rejection (e.g., sending a "500 Server Internal Error" response message to an INVITE request), and "forward". The default value is "reject" in order to avoid possible SIP retransmissions when an unreliable transport is used. If the "alt-action" value is "forward", an "alt-target" attribute MUST be defined. The "alt-target" specifies a URI where the request should be forwarded (e.g., an answering machine with explanation of why the request cannot be accepted).

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

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

6.5. Complete Examples

This section presents two complete examples of load control documents vliad with respect to the XML schema defined in Section 7.

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>
            <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 an earthquake. Incoming calls to the earthquake domain "pompeii.example.com" 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 earthquake 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 earthquake 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="pompeii.example.com"/>
                    </lc:to>
                    <lc:from>
```

```
                           <many>
                               <except domain="pompeii.example.com"/>
                               <except domain="rescue.example.com"/>
                           </many>
                       </lc:from>
                   </lc:sip>
               </lc:call-identity>
               <validity>
                   <from>79-08-24T09:00:00+01:00</from>
                   <until>79-08-27T09:00:00+01:00</until>
               </validity>
           </conditions>
           <actions>
               <lc:accept alt-action="forward" alt-target=
                       "sip:earthquake@update.example.com">
                   <lc:rate>100</lc:rate>
               </lc:accept>
           </actions>

       </rule>
   <ruleset>
```

7. 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 increment 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 the full control policy update, or whether it contains only state delta as partial update. Secondly, it defines new members of the <conditions> and <action> 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"/>
```

```
<?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"/>
```

8. Related Work

8.1. Relationship with Load Filtering in PSTN

It is known that the existing PSTN network also uses a load filtering mechanism to prevent overload and the filter configuration is done manually. This specification defines a SIP events framework based distribution mechanism which allows automated filter 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 [AINGR]. These items could be roughly correlated to the identity, action and time fields of the SIP load filter defined in this specification. However, the filter 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, and the number of prefix to be matched for a group of telephone numbers is usually a fixed set. The SIP filter identity allows both SIP URI and telephone numbers (through Tel URI) to be specified. The identities can be arbitrarily grouped by SIP domains or any number of leading prefix of the telephone numbers.

Secondly, the PSTN filtering action is usually limited to call gapping with a fixed set of allowed gapping intervals. The action field in the SIP load filter allows more flexible rate throttle and other possibilities.

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

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

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

PSTN filtering also provides administrative option for routing failed call attempts to either Recorder Tone or a special announcement. Similar capability can be provided in the SIP filtering mechanism by specifying the appropriate "alt-action" attribute in the SIP filtering action field.

8.2. Relationship with Other IETF SIP Load Control Efforts

The load filtering rules in this specification consists 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 filter 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 filter identities for load control and could share well-known load control actions. The time duration field in the load filter could also be used in both mechanisms. As mentioned in Section 1, the load filter distribution mechanism and the feedback overload control mechanism address complementary areas in the load control problem space. Load filtering is more proactive and focuses on distributing the filter 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 filter components. For example, the load filtering mechanism may use the time field in the filter 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 control duration.

9. Discussion of this specification meeting the requirements of RFC5390

This section evaluates whether the load control event package defined in this specification satisfies the various SIP overload control requirements set forth by RFC5390 [RFC5390]. Not all RFC5390 requirements are found applicable due to the scope of this document. 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 filter values are installed in advance and do not change during the potential overload period. P/A if filter values are dynamically adjusted due to the specific filter computation algorithm. The dynamic filter computation algorithm is outside the scope of this specification, while the distribution of the updated filters 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 4.4.

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 4.4.

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/windows-based overload control options as discussed in Section 6.4.

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 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, the load filter [Section 4.1] 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 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 4.4).

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 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 filters are provisioned in advance, they are not affected by the overload or failure of other nodes. But, on the other hand, they may not, in those cases, be able to protect the overloaded node (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 RFC3265 [RFC3265] 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 [RFC3265] 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 filter format definition in Section 4.1.

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 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 4.4.

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 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 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 does not preclude its use in a scenario with server farms.

10. Security Considerations

Two aspects of security considerations arise from this specification. One is the SIP event framework based filter distribution mechanism, the other is the filter enforcement mechanism.

Security considerations for SIP event framework based mechanisms are covered in Section 5 of [RFC3265]. 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, all load control notification MUST be authenticated and authorized before being accepted. Standard authentication and authorization mechanisms recommended in [RFC3261] such as TLS [RFC5246] and IPSec [RFC4301] may serve this purpose. On the other hand, if a legitimate notifier is itself compromised, additional mechanisms will be needed to detect the attack.

Security considerations for filter enforcements vary depending on the filter itself. This specification defines possible filter match of the following SIP header fields: <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.

11. IANA Considerations

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

11.1. Load Control Event Package Registration

This section registers an event package based on the registration procedures defined in [RFC3265].

Package name: load-control

Type: package

Published specification: This specification

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

11.2. application/load-control+xml MIME Registration

This section registers a new MIME type based on the procedures defined in [RFC4288] 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 10 of this specification

Interpretability 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>

11.3. 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 7.

Its first line is

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

and its last line is

```
</xs:schema>
```

12. Acknowledgements

The authors would like to thank Bruno Chatras, Keith Drage, Ashutosh Dutta, Janet Gunn, Vijay Gurbani, Volker Hilt, Geoff Hunt, Hadriel Kaplan, Paul Kyzivat, Salvatore Loreto, Timothy Moran, Eric Noel, Parthasarathi R, Shida Schubert, Robert Sparks, Phil Williams and other members of the SOC and SIPPING working group for many helpful comments. In addition, Bruno Chatras proposed the <method> condition element. Janet Gunn provided detailed text suggestions for Section 9. Shida made many suggestions about terminology usage. Phil added support for delta updates. Ashutosh Dutta gave pointers to PSTN references.

13. References

13.1. 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.

[RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, 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.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[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.

13.2. **Informative References**

[AINGR] Bell Communications Research, "AINGR: Service Control Point (SCP) Network Traffic Management", GR-2938-CORE , December 1996.

[I-D.ietf-soc-overload-control]
Gurbani, V., Hilt, V., and H. Schulzrinne, "Session Initiation Protocol (SIP) Overload Control", draft-ietf-soc-overload-control-06 (work in progress), January 2012.

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

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[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.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 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
AT&T Security Research Center
33 Thomas Street
New York, NY 10007
USA

Phone: +1 212 513 2081
Email: shen@att.com

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