Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2009

A. Niemi
K. Kiss
Nokia
S. Loreto
Ericsson

Feb 22, 2009

Session Initiation Protocol (SIP) Event Notification Extension for Notification Throttling
draft-niemi-sipping-event-throttle-08

Status of this Memo

Copyright Notice

Abstract

This memo specifies the throttle, forge and average mechanisms for adjusting the rate of Session Initiation Protocol (SIP) event notifications. These mechanisms can be applied in subscriptions to all SIP event packages, but in particular the throttle mechanism is especially designed to be used in combination with a subscription to a Resource List Server (RLS).

Table of Contents

1. Introduction

The SIP events framework [RFC3265] defines a generic framework for subscriptions to and notifications of events related to SIP systems. This framework defines the methods SUBSCRIBE and NOTIFY, and introduces the concept of an event package, which is a concrete application of the SIP events framework to a particular class of events.

One of the things the SIP events framework mandates is that each event package specification defines an absolute maximum on the rate at which notifications are allowed to be generated by a single notifier. Such a limit is provided in order to reduce network congestion.

All of the existing event package specifications include a maximum notification rate recommendation, ranging from once in every five seconds [RFC3856], [RFC3680], [RFC3857] to once per second [RFC3842].

Per the SIP events framework, each event package specification is also allowed to define additional throttle mechanisms which allow the subscriber to further limit the rate of event notification. So far none of the event package specifications have defined such a mechanism.

The resource list extension [RFC4662] to the SIP events framework also deals with rate limiting of event notifications. The extension allows a subscriber to subscribe to a heterogenous list of resources with a single SUBSCRIBE request, rather than having to install a subscription for each resource separately. The event list subscription also allows rate limiting, or throttling of notifications, by means of the Resource List Server (RLS) buffering notifications of resource state changes, and sending them in batches. However, the event list mechanism provides no means for the subscriber to set the interval for the throttling; it is strictly an implementation decision whether batching of notifications is supported, and by what means.

This document defines an extension to the SIP events framework defining the following three "Event" header field parameters that allow a subscriber to set a Minimum, a Maximum and an Average rate of event notifications generated by the notifier:

Throttle: specifies a minimum notification time period between two notifications, in seconds.

Force: specifies a maximum notification time period between two notifications, in seconds. Whenever the time since the most recent notification exceeds the value in the "force" parameter, then the current state would be sent in its entirety (just like after a subscription refresh).

Average: specifies an average cadence at which notifications are desired, in seconds. It works similar to the "force" parameter, except that it will reduce the frequency at which notifications are sent if several have already been sent recently.

The requirements and model are further discussed in Section 3. All those mechanisms are simply timer values that indicates the minimum, maximum and average time period allowed between two notifications. As a result of those mechanism, a compliant notifier will adjust the rate at which it generates notifications.

These mechanisms are applicable to any event subscription, both single event subscription and event list subscription.

2. Definitions and Document 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 RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

Indented passages such as this one are used in this document to provide additional information and clarifying text. They do not contain normative protocol behavior.

3. Overview

3.1. Throttle Use Case

A presence client in a mobile device contains a list of 100 buddies or presentities. In order to decrease the processing and network load of watching 100 presentities, the presence client has employed a Resource List Server (RLS) with the list of buddies, and therefore only needs a single subscription to the RLS in order to receive

notification of the presence state of the resource list.

In order to control the buffer policy of the RLS, the presence client sets a throttle interval via the event throttle extension. Alternatively, the presence client could set a default throttle for the resource list, via a list manipulation interface, e.g., using the XML Configuration Access Protocol (XCAP) [RFC4825].

The RLS will buffer notifications that do not comply with the throttle interval, and batch all of the buffered state changes together in a single notification when allowed by the throttle. The throttle applies to the overall resource list, which means that there is a hard cap imposed by the throttle to the amount of traffic the presence client can expect to receive.

For example, with a throttle of 20 seconds, the presence application can expect to receive a notification every 20 seconds at a maximum.

The presence client can also modify the throttle during the lifetime of the subscription. For example, if the User Interface (UI) of the application shows inactivity for a period of time, it can simply pause the notifications by setting the throttle interval to the subscription expiration time, while still keeping the subscription alive. When the user becomes active again, the presence client can resume the stream of notifications by re-setting the throttle to the earlier used value.

> Currently, a subscription refresh is needed in order to update the throttle interval. However, this is highly inefficient, since each refresh automatically generates a (full-state) notification carrying the latest resource state. There is work [I-D.ietf-sip-subnot-etags] ongoing to solve these inefficiencies.

3.2. Force Use Case

A location application is monitoring the movement of a target.

In order to decrease the processing and network load, the location application has made a subscription with a set of location filters [I-D.ietf-geopriv-loc-filters] that specify, for example, to send an update only when the target has moved at least 10 meters. However the application is interested in receiving an update periodically even if the target has not moved more than 10 meters in a second.

The application is interested in discovering if the state is changed, even when it has not changed enough to satisfy any of the 'trigger' criteria

In order to control the actual state, the location application sets a force interval via the event force extension. The force triggers a notification that is exactly and precisely like a notification after a subscription refresh.

The location application can also modify the force during the lifetime of the subscription.

3.3. Average Use Case

The throttle and force mechanisms introduce a static and instantaneous rate control. However there are some applications that would work better with an adaptive rate control (i.e. an average rate). This section illustrates the tracking scenario.

A tracking application is monitoring a target.

In order to decrease the processing and network load, the tracking application wants to make a subscription that dynamically reduces the frequency at which notifications are sent if the target has started to move sending out already several notifications recently.

In order to set an average rate control, the application defines a average interval via the event average extension. The average value is used by the notifier to dynamically calculate the maximum time allowed between two subscriptions. In order to dinamically calculate the maximum, the Notifer takes into consideration the frequency at which notifications have been sent recently.

The average rate control allows the notifier to dynamically increase or decrease the Notification frequency.

The tracking application can also modify the average interval during the lifetime of the subscription by setting the event average extension to a different value.

3.4. Requirements

REQ1: The subscriber must be able to set the minimum time (throttle) period between two notifications in a specific subscription.

REQ2: The subscriber must be able to set the maximum time period (force) between two notifications in a specific subscription.

REQ3: The subscriber must be able to set an average cadence (average) at which notifications are desired in a specific subscription.

REQ4: It must be possible to apply all together, or in any combination, the throttle, force and average mechanisms in a specific subscription.

REQ5: It must be possible to use of the throttle, force and average mechanisms in subscriptions to any events.

REQ6: It must be possible to use the throttle, force and average mechanisms together with any other event filtering mechanisms.

REQ7: The notifier must be allowed to use a throttling policy in which the minimum time period between two notifications is adjusted from the value given by the subscriber.

For example, due to congestion reasons, local policy at the notifier could temporarily dictate a throttling policy that in effect increases the subscriber-configured minimum time period between two notifications.

REQ8: The throttle mechanism must discuss corner cases for setting the minimum period between two notifications. At a minimum, the throttling mechanism must include discussion of the situation resulting from a minimum time period which exceeds the subscription duration, and should provide mechanisms for avoiding this situation.

REQ9: A throttle, force and average must be possible to be installed, modified, or removed in the course of an active subscription.

REQ10: A throttle, force and average mechanism must allow for the application of authentication and integrity protection mechanisms to subscriptions invoking that mechanism.

Note that Section 9 contains further discussion on the security implications of the throttle mechanism.

3.5. Event Throttle Model for Resource List Server

When applied to a list subscription, the event throttle mechanism has some additional considerations. Specifically, the throttle applies to the aggregate notification stream resulting from the list subscription, rather than explicitly controlling the notification of

each of the implied constituent events. Moreover, the list event notifier can use the throttle mechanism on its own to control the rate of the individual subscriptions to avoid overflowing its buffer.

The notifier is responsible for sending out event notifications upon state changes of the subscribed resource.We can model the notifier as consisting of three components: the event state resource(s), the Resource List Server (RLS) (or any other notifier), a notification buffer, and finally the subscriber, or watcher of the event state, as shown in Figure 1.

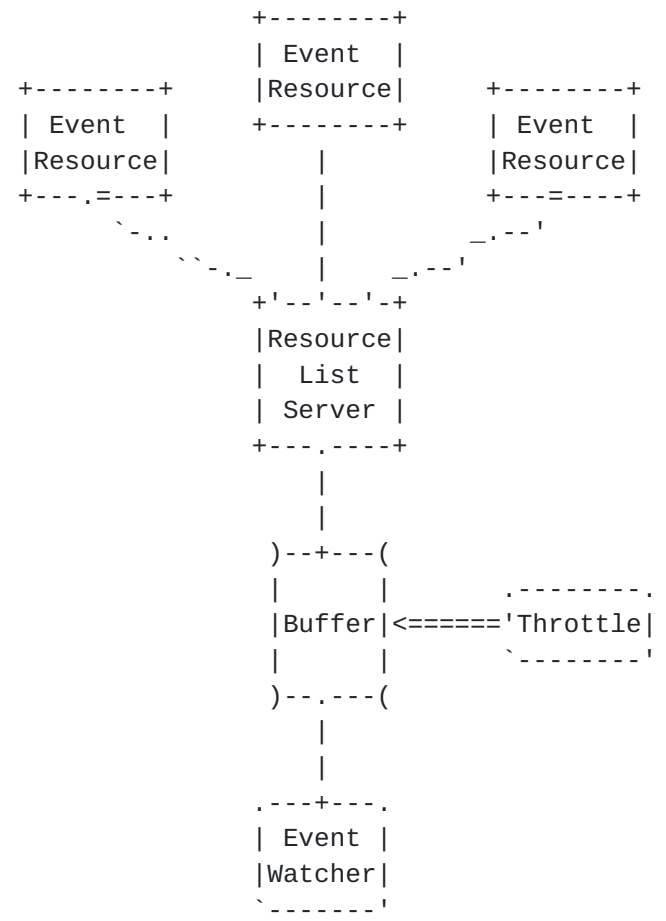

Figure 1: Model for the Resource List Server (RLS) Supporting Throttling

In short, the RLS reads event state changes from the event state resource, either by creating a backend subscription, or by other means; it packages them into event notifications, and submits them into the output buffer. The rate at which this output buffer drains is controlled by the subscriber via the event throttle mechanism. When a set of notifications are batched together, the way in which

overlapping resource state is handled depends on the type of the resource state:

In theory, there are many buffer policies that the notifier could implement. However, we only concentrate on two practical buffer policies in this specification, leaving additional ones for further study and out of the scope of this work. These two buffer policies depend on the mode in which the notifier is operating.

Full-state: Last (most recent) full state notification of each resource is sent out, and all others in the buffer are discarded. This policy applies to those event packages that carry full-state notifications.

Partial-state: The state deltas of each buffered partial notification per resource are merged, and the resulting notification is sent out. This policy applies to those event packages that carry partial-state notifications.

3.6. Basic Operation

A subscriber that wants to limit the rate of event notification in a specific event subscription does so by including a throttle as part of the SUBSCRIBE request. The throttle indicating the minimum time allowed between transmission of two consecutive notifications in a subscription is given as an Event header parameter in the SUBSCRIBE request.

Note that the witnessed time between two consecutive received notifications may not conform to the set throttle for a number of reasons. For example, network jitter and retransmissions may result in the subscriber receiving the notifications in lesser intervals than what the throttle recommends.

A subscriber that wants to have a maximum notification time period in a specific event subscription does so by including a force as part of the SUBSCRIBE request. The force indicating the maximum time allowed between transmission of two consecutive notifications in a subscription is given as an Event header parameter in the SUBSCRIBE request.

A subscriber that wants to have an average cadence at which notifications are desired in a specific event subscription does so by including an average as part of the SUBSCRIBE request. The average is given as an Event header parameter in the SUBSCRIBE request.

A notifier that supports the throttle, force and average mechanisms will comply with value given in the throttle, force and average and

adjust its rate of notification accordingly. However, if the notifier needs to lower the subscription expiration value or a local policy at the notifier can not meet the requested throttle value, then the notifier can adjust opportunely the received throttle value.

Throttled, forced and averaged notifications will have exactly the same properties as the ones the un-throttled, un-forced and un-averaged, with the exception that they will be generated with the frequency that has been requested.

3.7. Usage of Throttle, Force and Average in a subscription

Applications can subscribe to an event package using all the throttle, force and average mechanisms singly, or in combination; in fact there is no technical incompatibility among them. However there are some combinations that make little sense to be used together. This section lists all the possible combinations that is possible to insert in a subscription; the utility to use each combination in a subscription is also analyzed.

Throttle and Force: this combination let possible to reduce the notification frequence rate, but at same time assures the reception of a notification every time the most recent notification exceeds a specified interval.

A subscriber SHOULD choose a "force" value higher than the "throttle" value, otherwise the notifier MUST adjust the subscriber provided "force" value to a value equivalent or higher than the "throttle" value.

Throttle and Average: it works in a similar way as the combination above, but with the difference that the interval at which notifications are assured changes dynamically.

A subscriber SHOULD choose an "average" value higher than the "throttle" value, otherwise the notifier MUST adjust the subscriber provided "average" value to a value equivalent or higher than the "throttle" value.

Force and Average: as both the parameters are designed to force an update, this combination makes sense only in some corner cases.

A subscriber SHOULD choose a "force" value higher than the "average" value, otherwise the notifier MUST not consider the "force" value.

Throttle, Force and Average: this combination makes little sense to be used.

4. Operation of Event Throttles

4.1. Negotiating the Use of Throttle

A subscriber that wishes to apply a throttle to notifications in a subscription constructs a SUBSCRIBE request that includes a throttle interval in a "throttle" Event header field parameter.

A compliant notifier will reflect back the possibly adjusted throttle interval in a "throttle" Subscription-State header field parameter of the subsequent NOTIFY requests. The indicated throttle value is adopted by the notifier, and the notification rate is adjusted accordingly.

A notifier that does not understand the event-throttle extension, will not reflect the "throttle" parameter in the NOTIFY requests; the absence of this parameter serves as a hint to the subscriber that no throttling is supported by the notifier.

A subscriber that wishes to remove a throttle from notifications constructs a SUBSCRIBE request that does not include a "throttle" Event header field parameter.

4.2. Setting the Throttle

4.2.1. Subscriber Behavior

In general, the way in which a subscriber generates SUBSCRIBE requests and processes NOTIFY requests is according to RFC 3265 [RFC3265].

A subscriber that wishes to throttle the notifications in a subscription includes a "throttle" Event header parameter in the SUBSCRIBE request, indicating in seconds the desired throttle value. The value of this parameter is an integral number of seconds in decimal.

There are two main consequences for the subscriber when applying the throttle mechanism: state transitions may be lost, and event notifications may be delayed. If either of these side effects constitute a problem to the application that is to utilize event throttles, developers are instructed not to use the mechanism.

4.2.2. Notifier Behavior

In general, the way in which a notifier processes SUBSCRIBE requests and generates NOTIFY requests is according to RFC 3265 [RFC3265].

A notifier that supports the event-throttle extension extracts the value of the "throttle" Event header parameter, and uses it as the suggested minimum time allowed between two notifications. This value can be adjusted by the notifier, as defined in Section 4.3.

The notifier MUST reflect back the possibly adjusted throttle interval in a "throttle" Subscription-State header field parameter of the subsequent NOTIFY requests.

A compliant notifier MUST NOT generate notifications more frequent than what the throttle allows for, except when generating the notification either upon receipt of a SUBSCRIBE request (the first notification), when the subscription state is changing from "pending" to "active" state or upon termination of the subscription (the last notification). Such notifications reset the throttle timer, even though they do not need to abide by it.

Retransmissions of NOTIFY requests are not affected by the throttle, i.e., the throttle only applies to the generation of new transactions. In other words, the throttle is reset only after the previous transaction has completed.

4.3. Selecting the Throttle Interval

Special care needs to be taken when selecting the throttle value. Using the throttle syntax it is possible to insist both very short and very long throttles to be applied to the subscription. For example, a throttle could potentially set a minimum time value between notifications that exceeds the subscription expiration value. Such a configuration would effectively quench the notifier, resulting in exactly two notifications to be generated.

In some cases it makes sense to pause the notification stream on an existing subscription dialog on a temporary basis without terminating the subscription, e.g. due to inactivity on the application UI. Whenever a subscriber discovers the need to perform the notification pause operation, it SHOULD set the throttle interval to the remaining subscription expiration value. This results in receiving no further notifications until the subscription expires, renewed or notifications are resumed by the subscriber.

The notifier is responsible for adjusting the proposed throttle value based on its local policy or other properties.

If the subscriber requests a throttle greater than the subscription expiration,the notifier MUST lower the throttle value and set it to the expiration time left. According to RFC 3265 [RFC3265] the notifier may also shorten the subscription expiry anytime during an active subscription. For such cases, the notifier MUST also lower the throttle value and set it to the reduced expiration time.

The notifier MAY also choose a higher throttle value, e.g., because of static throttle value configuration given by local policy. The notifier MUST include the adjusted throttle value in the Subscription-State header field's "throttle" parameter in each of the NOTIFY requests. In addition, different event packages MAY define additional constraints to the allowed throttle intervals. Such constraints are out of the scope of this specification.

4.4. Buffer Policy Description

4.4.1. Partial State Notifications

With partial notifications, the notifier will always need to keep both a copy of the current full state of the resource F, as well as the last successfully communicated full state view F' of the resource in a specific subscription. The construction of a partial notification then involves creating a diff of the two states, and generating a notification that contains that diff.

When a throttle is applied to the subscription, it is important that F' is replaced with F only when the throttle is reset. Additionally, the notifier implementation SHOULD check to see that the size of an accumulated partial state notification is smaller than the full state, and if not, the notifier SHOULD send the full state notification instead.

4.4.2. Full State Notifications

With full state notifications, the notifier only needs to keep the full state of the resource, and when that changes, send the resulting notification over to the subscriber.

When a throttle is applied in the subscription, the notifier receives the state changes of the resource, and generates a notification. If there is a pending notification, the notifier simply replaces that notification with the new notification, discarding the older state.

4.5. Estimated Bandwidth Savings

It is difficult to estimate the total bandwidth savings accrued by using the throttle mechanism over a subscription, since such

estimates will vary depending on the usage scenarios. However, it is easy to see that given a subscription where several full state notification would have normally been sent in any given throttle interval, a throttled subscription would only send a single notification during the same interval, yielding bandwidth savings of several times the notification size.

With partial-state notifications, drawing estimates is further complicated by the fact that the states of consecutive updates may or may not overlap. However, even in the worst case scenario, where each partial update is to a different part of the full state, a throttled notification merging all of these n partial states together should at a maximum be the size of a full-state update. In this case, the bandwidth savings are approximately n times the size of the NOTIFY header.

It is also true that there are several compression schemes available that have been designed to save bandwidth in SIP, e.g., SigComp [RFC3320] and TLS compression [RFC3943]. However, such compression schemes are complementary rather than competing mechanisms to the throttle mechanism. After all, they can both be applied simultaneously, and in such a way that the compound savings are as good as the sum of applying each one alone.

5. Operation of Event Force

5.1. Negotiating the Use of Force

A subscriber that wishes to apply a maximum notification time period between two notifications in a subscription constructs a SUBSCRIBE request that includes a proposed maximum interval in a "force" Event header field parameter.

A compliant notifier will reflect back the possibly adjusted forced interval in a "force" Subscription-State header field parameter of the subsequent NOTIFY requests. The indicated force value is adopted by the notifier, and the notification rate is adjusted accordingly.

A notifier that does not understand the event-force extension, will not reflect the "force" parameter in the NOTIFY requests; the absence of this parameter serves as a hint to the subscriber that no forcing is supported by the notifier.

A subscriber that wishes to remove a force from notifications constructs a SUBSCRIBE request that does not include a "force" Event header field parameter.

5.2. Setting the Force

5.2.1. Subscriber Behavior

In general, the way in which a subscriber generates SUBSCRIBE requests and processes NOTIFY requests is according to RFC 3265 [RFC3265].

A subscriber that wishes to apply a maximum notification time period between the notifications in a subscription includes a "force" Event header parameter in the SUBSCRIBE request, indicating in seconds the desired force value. The value of this parameter is an integral number of seconds in decimal.

The main consequence for the subscriber when applying the force mechanism is that it can receive a notification even if nothing has changed in the current state of the notifier.

There is work [I-D.ietf-sip-subnot-etags] ongoing to only send a reference in a notification if nothing has changed.

5.2.2. Notifier Behavior

In general, the way in which a notifier processes SUBSCRIBE requests and generates NOTIFY requests is according to RFC 3265 [RFC3265].

A notifier that supports the event-force extension extracts the value of the "force" Event header parameter, and uses it as the suggested maximum time allowed between two notifications. This value can be adjusted by the notifier based on its local policy or other properties.

The notifier MUST reflect back the possibly adjusted force value in a "force" Subscription-State header field parameter of the subsequent NOTIFY requests.

A compliant notifier MUST generate notifications whenever the time since the most recent notification exceeds the value in the "force" parameter. The NOTIFY request then MUST contain the current state in its entirety, just like after a subscription refresh.

Retransmissions of NOTIFY requests are not affected by the force, i.e., the force only applies to the generation of new transactions. In other words, the force is reset only after the previous transaction has completed.

6. Operation of Event Average

6.1. Negotiating the Use of Average

A subscriber that wishes to apply an average cadence at which notifications are desired in a subscription constructs a SUBSCRIBE request that includes a proposed average interval in an "average" Event header field parameter.

A compliant notifier will reflect back the possibly adjusted average interval in an "average" Subscription-State header field parameter of the subsequent NOTIFY requests. The indicated average value is adopted by the notifier, and the notification rate is adjusted accordingly.

A notifier that does not understand the event-average extension will not reflect the "average" parameter in the NOTIFY requests; the absence of this parameter serves as a hint to the subscriber that no averaging is supported by the notifier.

A subscriber that wishes to remove a average from notifications constructs a SUBSCRIBE request that does not include an "average" Event header field parameter.

6.2. Calculating the Average Interval

The formula used to vary the absolute pacing in a way that will meet the average requested over the period is given in equation (1):

```
timeout = (average ^ 2) * count / period                (1)
```

The output of the formula, "timeout", is the time to the next notification, expressed in seconds. The formula has three inputs:

average: The value of the "average" parameter conveyed in the "Event" header field, in seconds.

period: The rolling average period, in seconds. A suggested reasonable period is 60 seconds.

[OPEN ISSUE]Is the period value something we should be able to tune, or we can simply specify a reasonable period?

count: The number of notifications that have been sent during the last "period" of seconds.

In the case both the Throttle and the Average are used in the same

subscription the formula used to dynamically calculate the timeout is given in equation (2):

timeout = MAX[throttle, (average ^ 2) * count / period] (2)

throttle: The value of the "threshold" parameter conveyed in the "Event" header field, in seconds.

The formula in (2) makes sure that for all the possible value of throttle and average, with average > throttle, the timeout never results in a lower value than throttle.

6.3. Setting the Average

6.3.1. Subscriber Behavior

In general, the way in which a subscriber generates SUBSCRIBE requests and processes NOTIFY requests is according to RFC 3265 [RFC3265].

A subscriber that wishes to apply an average cadence at which notifications are desired in a subscription includes a "average" Event header parameter in the SUBSCRIBE request, indicating in seconds the desired average value. The value of this parameter is an integral number of seconds in decimal.

The main consequence for the subscriber when applying the average mechanism is that it can receive a notification even if nothing has changed in the current state of the notifier.

There is work [I-D.ietf-sip-subnot-etags] ongoing to only send a reference in a notification if nothing has changed.

6.3.2. Notifier Behavior

In general, the way in which a notifier processes SUBSCRIBE requests and generates NOTIFY requests is according to RFC 3265 [RFC3265].

A notifier that supports the event-average extension extracts the value of the "average" Event header parameter, and uses it to calculate the maximum time allowed between two transactions as defined in Section 6.2. This value can be adjusted by the notifier based on its local policy or other properties.

The notifier MUST reflect back the possibly adjusted average value in a "average" Subscription-State header field parameter of the subsequent NOTIFY requests.

A compliant notifier MUST generate notifications whenever the time since the most recent notification exceeds the value calculated using the formula defined in Section 6.2.

The Average mechanism is implemented as follows:

1) When a subscription is first created, the notifier creates a record that keeps track of the number of notifications that have been sent in the "period". This record is initialized to contain a history of having sent one message every "average" seconds for the "period".

2) The "timeout" value is calculated according to the equation given in section Section 6.2.

3) If the timeout period passes without a NOTIFY request being sent in the subscription, then the current resource state is sent (subject to any filtering associated with the subscription).

4) Whenever a NOTIFY request is sent (regardless of whether due to a timeout or a state change), the notifier updates the notification history record, recalculates the value of "timeout," and returns to step 3.

Retransmissions of NOTIFY requests are not affected by the timeout, i.e., the timeout only applies to the generation of new transactions. In other words, the timeout is reset only after the previous transaction has completed.

7. Syntax

This section describes the syntax extensions required for the throttle, force and average mechanisms.

7.1. "throttle", "force" and "average" Header Field Parameters

The "throttle", "force" and "average" parameters are added to the rule definitions of the Event header field and the Subscription-State header field in the SIP Events [RFC3265] grammar. Usage of this parameter is described in section Section 4.2.

7.2. Augmented BNF Definitions

This section describes the Augmented BNF [RFC5234] definitions for the new syntax elements. Note that we derive here from the ruleset present in SIP Events [RFC3265], adding additional alternatives to the alternative sets of "event-param" and "subexp-params" defined

therein.

```
event-param    =/  throttle-param
subexp-params  =/  throttle-param
throttle-param =   "throttle" EQUAL delta-seconds

event-param    =/  force-param
subexp-params  =/  force-param
throttle-param =   "force" EQUAL delta-seconds

event-param    =/  average-param
subexp-params  =/  average-param
throttle-param =   "average" EQUAL delta-seconds
```

8. IANA Considerations

This specification registers three new SIP header field parameters, defined by the following information which is to be added to the Header Field Parameters and Parameter Values sub-registry under http://www.iana.org/assignments/sip-parameters.

Header Field	Parameter Name	Predefined Values	Reference
Event	throttle	No	[RFCxxxx]
Subscription-State	throttle	No	[RFCxxxx]
Event	force	No	[RFCxxxx]
Subscription-State	force	No	[RFCxxxx]
Event	average	No	[RFCxxxx]
Subscription-State	average	No	[RFCxxxx]

(Note to the RFC Editor: please replace "xxxx" with the RFC number of this specification, when assigned.)

9. Security Considerations

Naturally, the security considerations listed in SIP events [RFC3265], which the throttle mechanism extends, apply in entirety. In particular, authentication and message integrity SHOULD be applied to subscriptions with the event-throttle extension.

10. Acknowledgements

Thanks to Pekka Pessi, Dean Willis, Eric Burger, Alex Audu, Alexander Milinski, Jonathan Rosenberg, Cullen Jennings, Adam Roach, Hisham

Khartabil and Dale Worley for support and/or review of this work.

Thanks to Brian Rosen for the idea of the "force" and "average" mechanisms, and to Adam Roach for the work on the averaging algorithm.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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.

[RFC4662] Roach, A., Campbell, B., and J. Rosenberg, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists", RFC 4662, August 2006.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

11.2. Informative References

[I-D.ietf-geopriv-loc-filters]
Mahy, R. and B. Rosen, "A Document Format for Filtering and Reporting Location Notications in the Presence Information Document Format Location Object (PIDF-LO)", draft-ietf-geopriv-loc-filters-03 (work in progress), November 2008.

[I-D.ietf-sip-subnot-etags]
Niemi, A., "An Extension to Session Initiation Protocol (SIP) Events for Conditional Event Notification", draft-ietf-sip-subnot-etags-03 (work in progress), July 2008.

[RFC3320] Price, R., Bormann, C., Christoffersson, J., Hannu, H., Liu, Z., and J. Rosenberg, "Signaling Compression (SigComp)", RFC 3320, January 2003.

[RFC3680] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Registrations", RFC 3680, March 2004.

[RFC3842] Mahy, R., "A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)", RFC 3842, August 2004.

[RFC3856] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004.

[RFC3857] Rosenberg, J., "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)", RFC 3857, August 2004.

[RFC3943] Friend, R., "Transport Layer Security (TLS) Protocol Compression Using Lempel-Ziv-Stac (LZS)", RFC 3943, November 2004.

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

Authors' Addresses

Aki Niemi
Nokia
P.O. Box 407
NOKIA GROUP, FIN 00045
Finland

Phone: +358 50 389 1644
Email: aki.niemi@nokia.com

Krisztian Kiss
Nokia
313 Fairchild Dr
Mountain View, CA 94043
US

Phone: +1 650 391 5969
Email: krisztian.kiss@nokia.com

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com