A Mechanism for Session Initiation Protocol (SIP) Avalanche Restart
Overload Control
draft-shen-sipping-avalanche-restart-overload-00

Abstract

When a large number of clients register with a SIP registrar server at approximately the same time, the server may become overloaded. Near-simultaneous floods of SIP SUBSCRIBE and PUBLISH requests may have similar effects. Such request avalanches can occur, for example, after a power failure and recovery in a metropolitan area. This document describes how to avoid such overload situations. Under this mechanism, a server estimates an avalanche restart backoff interval during its normal operation and conveys this interval to its clients through a new Register-Restart header in registration responses. Once an avalanche restart actually occurs, the clients perform backoff based on the previously received Register-Restart header value before sending out the first registration attempt. Thus, the mechanism spreads all the initial client registration requests and prevents them from overloading the registrar server.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at...
Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................. 5
3. Register-Restart Header for Registration Responses ............ 5
   3.1. Generating the Register-Restart Header .................. 5
   3.2. Determining the Register-Restart Header Value .......... 5
   3.3. Processing the Register-Restart Header ................. 6
   3.4. Using the Register-Restart Header ....................... 6
4. Syntax ......................................................... 7
5. Backward Compatibility ......................................... 7
6. Security Considerations ....................................... 8
7. IANA Considerations .......................................... 8
8. References ..................................................... 8
   8.1. Normative References ..................................... 8
   8.2. Informative References ................................. 8
Authors’ Addresses ............................................... 9
1. Introduction

A Session Initiation Protocol (SIP) [RFC3261] server can be overloaded for a number of different reasons. One of them is avalanche restart, which is described in [RFC5390] as follows:

Avalanche Restart: One of the most troubling sources of overload is avalanche restart. This happens when a large number of clients all simultaneously attempt to connect to the network with a SIP registration. Avalanche restart can be caused by several events. One is the "Manhattan Reboots" scenario, where there is a power failure in a large metropolitan area, such as Manhattan. When power is restored, all of the SIP phones, whether in PCs or standalone devices, simultaneously power on and begin booting. They will all then connect to the network and register, causing a flood of SIP registration messages. Another cause of avalanche restart is failure of a large network connection, for example, the access router for an enterprise. When it fails, SIP clients will detect the failure rapidly using the mechanisms in [RFC5626]. When connectivity is restored, this is detected, and clients re-registration, all within a short time period. Another source of avalanche restart is failure of a proxy server. If clients had all connected to the server with TCP, its failure will be detected, followed by re-connection and re-registration to another server. Note that [RFC5626] does provide some remedies to this case.

The SIP server avalanche restart overload problem is caused by the synchronized, simultaneous initial registration attempts after a failure recovery. If the first round of registration attempts from all clients cause server overload, most of those registrations will fail. Those clients will then by default all retry after the same amount of time, causing repeated server avalanche restart overload. [RFC5626] describes how to alleviate this situation: if the initial registration attempt after the boot fails, the clients wait for a randomized backoff time before retrying. This mechanism reduces the possibility of repeated avalanche restart. However, since all clients still send registration immediately after boot, it does not prevent the initial avalanche restart overload.

A key method to prevent avalanche restart server overload is to have clients backoff before their first registration attempt. The backoff intervals of each client must be carefully selected so that their registration attempts are spaced sufficiently far apart not to overload the server, and they are also not too conservative which may cause unnecessary client registration delays. An individual client, without knowing the state information of all other peer clients and the registrar server, is inherently incapable of choosing such an
appropriate backoff interval.

This document specifies a solution to the avalanche restart overload problem by allowing the registrar server to instruct the clients how long they should wait before the initial registration upon a restart event. Under this mechanism, the server estimates an avalanche restart backoff interval during its normal operation. This interval is the minimum period of time that the server needs to serve all the expected registration requests after an avalanche restart, assuming all the registration requests are properly spaced. In order for the server to convey this interval to its clients, this document defines a new SIP Register-Restart header. The registrar server places the avalanche restart backoff interval into the Register-Restart header and inserts it into regular responses to client registration requests. When an avalanche restart actually happens, each client is required to wait a randomly-chosen time between 0 and the avalanche restart backoff interval. Any PUBLISH or SUBSCRIBE requests after the restart must be sent after the registration has been completed.

This document also defines an algorithm to determine the avalanche restart backoff interval based on the server’s processing capability and the number of clients it is serving. The effectiveness of this algorithm depends on the assumption that both the server processing capability and the number of clients the server serves remain similar before and after the event that causes the avalanche restart. This assumption holds true in most avalanche restart cases when the registrar server before and after the avalanche restart is the same one, e.g., in the "Manhattan Reboots" scenario, and the failure and recovery of a large network connection scenario. The cases where the clients will all register with a different server after restart is more complicated, because those clients will most likely not have an avalanche restart backoff interval value from the new registrar server a priori. In those cases, if the initial avalanche restart overload still occurs, the mechanisms in [RFC5626] can be used to help prevent repeated avalanche restart overload.

Some devices, especially mobile terminals, may have lower layer (e.g., PHY or Data Link layer) backoff or blocking mechanisms during avalanche restart or congestion cases. This document addresses the SIP application layer. Operators can disable this application layer avalanche restart protection method if the lower layer has already provided similar mechanism. Such cross-layer optimizations, however, are out of scope of this document.

This document complements other SIP server overload control specifications which address different aspects of the SIP server overload space, such as [I-D.hilt-sipping-overload], [I-D.shen-sipping-load-control-event-package], and
2. Terminology

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

3. Register-Restart Header for Registration Responses

This document defines the SIP Register-Restart header for registration responses. The value of the Register-Restart header, in seconds, denotes the avalanche restart backoff interval, which is the minimum time the server needs to successfully service all likely client registration requests under an avalanche restart situation, assuming all requests are spaced evenly in time.

3.1. Generating the Register-Restart Header

A SIP registrar server inserts a Register-Restart header containing its most up-to-date avalanche restart backoff interval value in the responses to registration requests.

Example:

SIP/2.0 200 OK
Register-Restart: 300

3.2. Determining the Register-Restart Header Value

A registrar server computes and updates the Register-Restart header values and conveys them to the clients during its normal operations. Once an avalanche restart actually happens, the most recent Register-Restart header value that the clients have received from the registrar server are used. A registrar server MAY use the following algorithm for determining the appropriate Register-Restart header value.

During the normal operation period, the SIP registrar server maintains the current count of all its registrants, e.g., assuming the number of registered clients is R. The SIP registrar server also estimates its processing capacity, e.g., assuming it is C requests per second. The Register-Restart value can be set to \((R/C)*(1+k)\), where k is a small coefficient that provides a capacity redundancy. A recommended value of k is 0.1.
It should be noted that change of either R or C adjusts the server computed Register-Restart value. The value C is usually stable with the same server and registration request pattern. The value R may change over time. The server SHOULD recompute the Register-Restart value whenever there is a change in either R or C, unless it is considered too expensive to do so, which is normally not the case. Since the updated Register-Restart value is only pushed to the clients when the client sends in a registration, there might be a short period where the server side updated Register-Restart value and some client side stored Register-Restart values are not synchronized. However, considering that the changing pace of R is slow, and the time scale between the possible happenings of avalanche restarts (e.g., months) is usually much larger than the interval between typical registration renewals (e.g., hours), these short periods of discrepancies are not a concern. Therefore, in general this approach provides a sufficiently accurate characterization of the system status. More importantly, the values of R and C are expected to remain constant for the same server before and after typical avalanche restart events, e.g., a power failure and recovery.

3.3. Processing the Register-Restart Header

Before receiving the very first registration response from a new registrar server, the client restart backoff value for that registrar server is zero, i.e., the restart backoff mechanism is disabled.

Upon receiving a response to the registration request containing the Register-Restart header, a SIP client that supports this specification MUST check if there is an existing Register-Restart header value for this registrar stored in the system. If not, it stores the newly received Register-Restart header value. Otherwise, it compares the new value with the existing one and updates it if they differ. The value of Register-Restart header MUST be stored together with the corresponding identity of the registrar server, e.g., the DNS name of the registrar server. There is no separate validity period parameter for Register-Restart. The validity duration of the Register-Restart header is the same as that of the corresponding registration operation.

3.4. Using the Register-Restart Header

At the client side, avalanche restart backoff is disabled by default, unless the client that supports this specification has received a positive Register-Restart header value from the corresponding registrar server.

A SIP client always keeps the most updated Register-Restart header value. When this value is positive and if the client detects that it
is about to perform the first registration with the same registrar server after a power-off reboot or a connection-loss recovery, the client SHOULD generate a uniformly distributed random interval between 0 and the current Register-Restart value, and wait until the end of that interval to send the registration request. However, the client side backoff MAY be manually disabled by a human operator when necessary, e.g., when the operator is expecting an urgent call, or when the power-off or connection-loss event is known as a local incident rather than a global event.

Furthermore, in order to prevent similar floods of SUBSCRIBE and PUBLISH requests after the restart, any PUBLISH and SUBSCRIBE MUST be sent after the registration has been completed.

It should be noted that the power-off reboot case requires that the state information about the Register-Restart value and the registrar server identity be stored in a memory space that could survive power restart.

4. Syntax

The new Register-Restart header adds the following lines to the existing SIP header definition.

```
message-header = Register-Restart

Register-Restart = "Register-Restart" HCOLON delta-seconds
```

5. Backward Compatibility

If a registrar server supports this specification, but not all of its clients are upgraded, then those non-compliant clients will ignore the Register-Restart header and not perform backoff. Although it appears that this might give the non-compliant clients an unfair advantage over those clients that do perform backoff, since the non-compliant clients will send synchronized registration attempts to the registrar server and can cause server overload, they will be penalized by registration failures. Depending on the number of the non-compliant clients vs. compliant clients, if the registrar server can still process requests when it does not receive registration storms from all the non-compliant clients, the requests from the compliant clients which spread apart, are more likely to succeed.
6. Security Considerations

The Register-Restart header can be used by an attacker to launch a possible denial-of-service attack on SIP clients if the attacker can insert an infinitely large Register-Restart value in the response sent to the clients. In those situations, the client may generate a very large backoff time before it attempts to send a registration request, and therefore the client is subject to denial-of-service attack. However, this kind of attack is only applicable after a power-cycle reboot or failure and recovery of a large network connection, which is rare. Furthermore, if the attacker can modify the registration request or response, that attacker can very easily prevent registration in any number of ways, so the Register-Restart header does not introduce new types of attacks. One method to prevent the registration request and response from being altered by attackers is to use TLS between the client and the registrar server.

7. IANA Considerations

[TBD]

8. References

8.1. Normative References


8.2. Informative References


Authors’ Addresses

Charles Shen
Columbia University
Department of Computer Science
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
Phone: +1 212 854 3109
Email: charles@cs.columbia.edu

Henning Schulzrinne
Columbia University
Department of Computer Science
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
Phone: +1 212 939 7004
Email: hgs@cs.columbia.edu

Arata Koike
NTT Service Integration Labs &
NTT Washington DC Representative Office
1100 13th St., NW, Suite 900
Washington DC, 20005
USA
Phone: +1 202 312 1451
Email: koike.arata@lab.ntt.co.jp