SOC Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 12 2012

Eric Noel
AT&T Labs
Philip M Williams
BT Innovate & Design

December 12, 2011

Session Initiation Protocol (SIP) Rate Control
draft-noel-soc-overload-rate-control-02.txt

Abstract

The prevalent use of Session Initiation Protocol (SIP) [RFC3261] in Next Generation Networks necessitates that SIP networks provide adequate control mechanisms to maintain transaction throughput by preventing congestion collapse during traffic overloads. Already [draft-ietf-soc-overload-control-05] proposes a loss-based solution to remedy known vulnerabilities of the [RFC3261] SIP 503 (service unavailable) overload control mechanism. This document proposes a rate-based control solution to complement the loss-based control defined in [draft-ietf-soc-overload-control-05].

Table of Contents

1. Introduction

The use of SIP in large scale Next Generation Networks requires that SIP based networks provide adequate control mechanisms for handling traffic growth. In particular, SIP networks must be able to handle traffic overloads gracefully, maintaining transaction throughput by preventing congestion collapse.

A promising SIP based overload control solution has been proposed in [draft-ietf-soc-overload-control-05]. That solution provides a communication scheme for overload control algorithms. It also includes a default loss-based overload control algorithm that makes

it possible for a set of clients to limit offered load towards an overloaded server.

However, such loss control algorithm is sensitive to variations in load so that any increase in load would be directly reflected by the clients in the offered load presented to the overloaded servers. More importantly, a loss-based control cannot guarantee clients to produce a bounded offered load towards an overloaded server and requires frequent updates which may have implications for stability.

This document proposes extensions to [draft-ietf-soc-overload-control-05] so as to support a rate-based control that guarantees clients produce a limited upper bound to the Invite rate towards an overloaded server, which is constant between server updates. The penalty for such a benefit is in terms of algorithmic complexity, since the overloaded server must estimate a target offered load and allocate a portion to each conversing client.

The proposed rate-based overload control algorithm mitigates congestion in SIP networks while adhering to the overload signaling scheme in [draft-ietf-soc-overload-control-05] and proposing a rate control in addition to the default loss-based control in [draft-ietf-soc-overload-control-05].

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 RFC 2119 [RFC2119].

The normative statements in this specification as they apply to SIP clients and SIP servers assume that both the SIP clients and SIP servers support this specification. If, for instance, only a SIP client supports this specification and not the SIP server, then follows that the normative statements in this specification pertinent to the behavior of a SIP server do not apply to the server that does not support this specification.

3. Rate-based algorithm scheme

3.1. Overview

The server is what the overload control algorithm defined here protects and the client is what throttles traffic towards the server.

Following the procedures defined in [draft-ietf-soc-overload-control-05], the server and clients signal one another support for rate-based overload control.

Then periodically, the server relies on internal measurements (e.g. CPU utilization, queueing delay...) to evaluate its overload state and estimate a target SIP request rate (as opposed to target percent loss in the case of loss-based control).

When in overload, the server uses [draft-ietf-soc-overload-control-05] via header oc parameters of SIP responses to inform the clients of its overload state and of the target SIP request rate.

Upon receiving the oc parameters with a target SIP request rate, each client throttles new SIP requests towards the overloaded server.

3.2. Summary of via headers parameters for overload control

To do: Repeat draft-ietf-soc-overload-control-05 definitions of the oc parameters here.

The use of the via header oc parameter(s) inform of the desired rate, but they don't explicitly "inform clients of the overload state".

3.3. Client and server rate-control algorithm selection

Per [draft-ietf-soc-overload-control-05], new clients indicate supported overload control algorithms to servers by inserting oc and oc-algo in Via header of SIP requests destined to servers. While servers notify clients of selected overload control algorithm

through the oc-algo parameter in the Via header of SIP responses to clients.

Support of rate-based control MUST be indicated by clients and servers by setting oc-algo to "rate".

3.4. Server operation

The actual algorithm used by the server to determine its overload state and estimate a target SIP request rate is beyond the scope of this document.

However, the server MUST be able to evaluate periodically its overload state and estimate a target SIP request rate beyond which it would become overloaded. The server must allocate a portion of the target SIP request rate to each of its client. Note that the target SIP request rate is a max rate that may not be attained by the arrival rate at the client, and the server cannot assume that it will.

Upon detection of overload, the server MUST follow the specifications in [draft-ietf-soc-overload-control-05] to notify its clients of its overload state and of the allocated target SIP request rate.

The server MUST use [draft-ietf-soc-overload-control-05] oc parameter to send a target SIP request rate to each of its client.

3.5. Client operation

3.5.1. Default algorithm

To throttle new SIP requests at the rate specified in the oc value sent by the server to its clients, the client MAY use the proposed default algorithm for rate-based control or any other equivalent algorithm.

The default Leaky Bucket algorithm presented here is based on [ITU-T Rec. I.371] Appendix A.2. The algorithm makes it possible for

clients to deliver SIP requests at a rate specified in the oc value with tolerance parameter TAU (preferably configurable).

Conceptually, the Leaky Bucket algorithm can be viewed as a finite capacity bucket whose real-valued content drains out at a continuous rate of 1 unit of content per time unit and whose content increases by the increment T for each forwarded SIP request. T is computed as the inverse of the rate specified in the oc value, namely T = 1 / oc-value.

Note that when the oc-value is 0 with a non zero oc-validity, then the client should reject 100% of SIP requests destined to the overload server. However, when both oc-value and oc-validity are 0, the client should immediately stop throttling.

If at a new SIP request arrival the content of the bucket is less than or equal to the limit value TAU, then the SIP request is forwarded to the server; otherwise, the SIP request is rejected.

Note that the capacity of the bucket (the upper bound of the counter) is (T + TAU).

At the arrival time of the k-th new SIP request ta(k) after control has been activated, the content of the bucket is provisionally updated to the value

X' = X - ([ta(k) - LCT])

where X is the content of the bucket after arrival of the last forwarded SIP request, and LCT is the time at which the last SIP request was forwarded.

If X' is less than or equal to the limit value TAU, then the new SIP request is forwarded and the bucket content X is set to X' (or to 0 if X' is negative) plus the increment T, and LCT is set to the current time ta(k). If X' is greater than the limit value tau, then the new SIP request is rejected and the values of X and LCT are unchanged.

When the first response from the server has been received indicating control activation (oc-validity>0), LCT is set to the time of activation, and the occupancy of the bucket is initialized to the parameter TAU0 (preferably configurable) which is 0 or larger but less than or equal to TAU.

Note that specification of a value for TAU is beyond the scope of this document.

3.5.2. Optional enhancement: avoidance of resonance

As the number of client sources of traffic increases and the throughput of the server decreases, the maximum rate admitted by each client needs to decrease, and therefore the value of T becomes larger. Under some circumstances, e.g. if the traffic arises very quickly simultaneously at many sources, the occupancies of each bucket can become synchronized, resulting in the admissions from each source being close in time and batched or very 'peaky' arrivals at the server, which not only gives rise to control instability, but also very poor delays and even lost messages. An appropriate term for this is 'resonance' [Erramilli].

If the network topology is such that this can occur, then a simple way to avoid this is to randomize the bucket occupancy at two appropriate points: At the activation of control, and whenever the bucket empties, as follows.

After updating the bucket occupancy to X', generate a value u as follows:

```
if X' > 0, then u=0

else if X' <= 0 then uniformly distributed between -1/2 and +1/2
```

Then (only) if the arrival is admitted, increase the bucket by an amount T + uT, which will therefore be just T if the bucket hadn't emptied, or lie between T/2 and 3T/2 if it had.

This randomization should also be done when control is activated, i.e. instead of simply initializing the bucket fill to TAU0, initialize it to TAU0 + uT, where u is uniformly distributed as above. Since activation would have been a result of response to a request sent by the client, the second term in this expression can be interpreted as being the bucket increment following that admission.

This method has the following characteristics:

. If TAU0 is chosen to be equal to TAU and all sources were to activate control at the same time due to an extremely high request rate, then the time until the first request admitted by each client would be uniformly distributed over [0,T];

. The maximum occupancy is TAU + (3/2)T, rather than TAU + T without randomization;

. For the special case of 'classic gapping' where TAU=0, then the minimum time between admissions is uniformly distributed over [T/2, 3T/2], and the mean time between admissions is the same, i.e. T+1/R where R is the request arrival rate;

. At high load randomization rarely occurs, so there is no loss of precision of the admitted rate, even though the randomized 'phasing' of the buckets remains.

3.5.3. Optional enhancement: priority

The proposed Leaky bucket implementation could be modified to support priority using multiple thresholds.

For instance, with two priorities it requires two thresholds TAU1 < TAU2:

. All new requests would be admitted when the bucket fill is at or below TAU1,

. Only higher priority requests would be admitted when the bucket fill is between TAU1 and TAU2,

. All requests would be rejected when the bucket fill is above TAU2.

This can be generalized to n priorities using n thresholds for n>2 in the obvious way.

4. Example

Adapting [draft-ietf-soc-overload-control-05] example in section 6.2 where SIP client P1 sends requests to a downstream server P2:

```
INVITE sips:user@example.com SIP/2.0

Via: SIP/2.0/TLS p1.example.net;

branch=z9hG4bK2d4790.1;received=192.0.2.111;

oc;oc-algo="loss,rate"

...


SIP/2.0 100 Trying

Via: SIP/2.0/TLS p1.example.net;

branch=z9hG4bK2d4790.1;received=192.0.2.111;

oc-algo="rate";oc-validity=0;

oc-seq=1282321615.781

 ...
```

In the messages above, the first line is sent by P1 to P2. This line is a SIP request; because P1 supports overload control, it inserts the "oc" parameter in the topmost Via header that it created. P1 supports two overload control algorithms: loss and rate.

The second line --- a SIP response --- shows the topmost Via header amended by P2 according to this specification and sent to P1. Because P2 also supports overload control, it chooses the "rate" based scheme and sends that back to P1 in the "oc-algo" parameter. It uses oc-validity=0 to indicate no overload.

At some later time, P2 starts to experience overload. It sends the following SIP message indicating P1 should send SIP requests at a rate no greater than or equal to 150 SIP requests per seconds.

```
SIP/2.0 180 Ringing

Via: SIP/2.0/TLS p1.example.net;

branch=z9hG4bK2d4790.1;received=192.0.2.111;
```

```
oc=150;oc-algo="rate";oc-validity=1000;

oc-seq=1282321615.782

 ...
```

5. Syntax

This specification extends the existing definition of the Via header field parameters of [RFC3261] as follows:

```
oc          = "oc" EQUAL oc-value

oc-value    = "NaN" / oc-num

oc-num      = 1*DIGIT
```

6. Security Considerations

To do: Use draft-ietf-soc-overload-control-05 section here.

7. IANA Considerations

None.

8. References

8.1. Normative References

[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.

8.2. Informative References

[draft-ietf-soc-overload-control-05]
Gurbani, V., Hilt, V., Schulzrinne, H., "Session Initiation Protocol (SIP) Overload Control", draft-ietf-soc-overload-control-05.

[ITU-T Rec. I.371]

"Traffic control and congestion control in B-ISDN", ITU-T Recommendation I.371.

[Erramilli]
A. Erramilli and L. J. Forys, "Traffic Synchronization Effects In Teletraffic Systems", ITC-13, 1991.

Appendix A. Acknowledgments

Many thanks for the contributions, comments and feedback on this document to: Janet Gunn.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Eric Noel
AT&T Labs
200s Laurel Avenue
Middletown, NJ, 07747
USA

Philip M Williams
BT Innovate & Design
UK