Network Working Group
Request for Comments: 4321
Category: Informational

R. Sparks
Estacado Systems

January 2006

Problems Identified Associated with the Session Initiation Protocol's (SIP) Non-INVITE Transaction

Status of This Memo

Copyright Notice

Abstract

This document describes several problems that have been identified with the Session Initiation Protocol's (SIP) non-INVITE transaction.

Table of Contents

1. Problems under the Current Specifications

There are a number of unpleasant edge conditions created by the SIP non-INVITE transaction (NIT) model's fixed duration. The negative aspects of some of these are exacerbated by the effect that provisional responses have on the non-INVITE transaction state machines as currently defined.

1.1. NITs must complete immediately or risk losing a race

The non-INVITE transaction defined in RFC 3261 [1] is designed to have a fixed and finite duration (dependent on T1). A consequence of this design is that participants must strive to complete the transaction as quickly as possible. Consider the race condition shown in Figure 1.

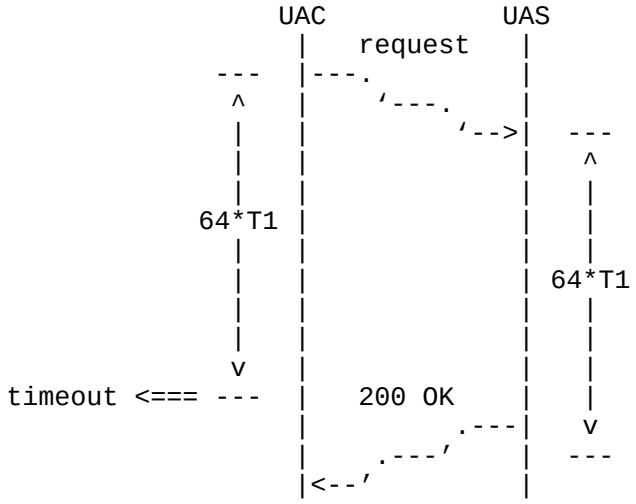

Figure 1: Non-Invite Race Condition

The User Agent Server (UAS) in this figure believes it has responded to the request in time, and that the request succeeded. The User Agent Client (UAC), on the other hand, believes the request has timed-out, hence failed. No longer having a matching client transaction, the UAC core will ignore what it believes to be a spurious response. As far as the UAC is concerned, it received no response at all to its request. The ultimate result is that the UAS and UAC have conflicting views of the outcome of the transaction.

Therefore, a UAS cannot wait until the last possible moment to send a final response within a NIT. It must, instead, send its response so that it will arrive at the UAC before that UAC times out. Unfortunately, the UAS has no way to accurately measure the propagation time of the request or predict the propagation time of the response. The uncertainty it faces is compounded by each proxy that participates in the transaction. Thus, the UAS's only choice is to send its final response as soon as it possibly can and hope for the best.

This result constrains the set of problems that can be solved with a single NIT. Any delay introduced during processing of a request increases the probability of losing the race. If the timing characteristics of that processing are not predictable and controllable, a single NIT is an inappropriate model for handling the request. One viable alternative is to accept the request with a 202 and send the ultimate results in a new request in the reciprocal direction.

In specialized networks, a UAS might have some reliable knowledge of inter-hop latency and could use that knowledge to determine if it has time to delay its final response in order to perform some processing such as a database lookup while mitigating its risk of losing the race in Figure 1. Establishing this knowledge across arbitrary networks (perhaps using resource reservation techniques and deterministic transports) is not currently feasible.

1.2. Provisional responses can delay recovery from lost final responses

The non-INVITE client transaction state machine provides reliability for NITs over unreliable transports (UDP) through retransmission of the request message. Timer E is set to T1 when a request is initially transmitted. As long as the machine remains in the Trying state, each time Timer E fires, it will be reset to twice its previous value (capping at T2) and the request is retransmitted.

If the non-INVITE client transaction state machine sees a provisional response, it transitions to the Proceeding state, where retransmission continues, but the algorithm for resetting Timer E is simply to use T2 instead of doubling at each firing. (Note that Timer E is not altered during the transition to Proceeding.)

Making the transition to the Proceeding state before Timer E is reset to T2 can cause recovery from a lost final response to take extra time. Figure 2 shows recovery from a lost final response with and without a provisional message during this window. Recovery occurs within 2*T1 in the case without the provisional. With the provisional, recovery is delayed until T2, which by default is 8*T1.

In practical terms, a provisional response to a NIT in currently deployed networks can delay transaction completion by up to 3.5 seconds.

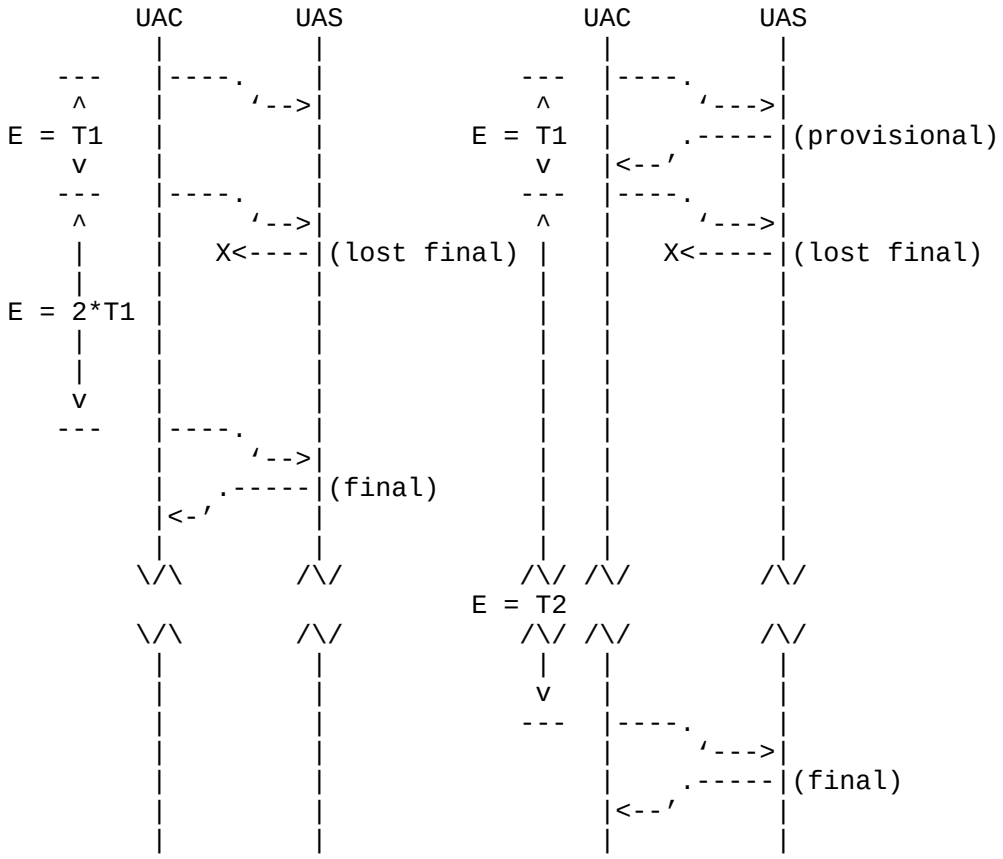

Figure 2: Provisionals Can Harm Recovery

No additional delay is introduced if the first provisional response is received after Timer E has reached its maximum reset interval of T2.

1.3. Delayed responses will temporarily blacklist an element

A SIP element's use of DNS Service Record Resource Records [3] is specified in RFC 3263 [2]. That specification discusses how SIP ensures high availability by having upstream elements detect failure of downstream elements. It proceeds to define several types of failure detection and instructions for failover. Two of the behaviors it describes are important to this document:

- Within a transaction, transport failure is detected either through an explicit report from the transport layer or through timeout. Note specifically that timeout will indicates transport failure regardless of the transport in use. When transport failure is detected, the request is retried at the next element from the sorted results of the SRV query.

- Between transactions, locations reporting temporary failure (through 503/Retry-After, for example) are not used until their requested black-out period expires.

The specification notes the benefit of caching locations that are successfully contacted, but does not discuss how such a cache is maintained. It is unclear whether an element should stop using (temporarily blacklist) a location returned in the SRV query that results in a transport error. If it does, when should such a location be removed from the blacklist?

Without such a blacklist (or equivalent mechanism), the intended availability mechanism fails miserably. Consider traffic between two domains. Proxy pA in domain A needs to forward a sequence of non-INVITE requests to domain B. Through DNS SRV, pA discovers pB1 and pB2, and the ordering rules of [2] and [3] indicate it should use pB1 first. The first request to pB1 times out. Since pA is a proxy and a NIT has a fixed duration, pA has no opportunity to retry the request at pB2. If pA does not remember pB1's failure, the second request (and all subsequent non-INVITE requests until pB1 recovers) are doomed to the same failure. Caching would allow the subsequent requests to be tried at pB2.

Since miserable failure is not acceptable in deployed networks, we should anticipate that elements will, in fact, cache timeout failures between transactions. Then the race in Figure 1 becomes important. If an element fails to respond "soon enough", it has effectively not responded at all and will be blacklisted at its peer for some period of time.

(Note that even with caching, the first request timeout results in a timeout failure all the way back to the original submitter. The failover mechanisms in [2] work well to increase the resiliency of a given INVITE transaction, but do nothing for a given non-INVITE transaction.)

1.4. 408 for non-INVITE is not useful

Consider the race condition in Figure 1 when the final response is 408 instead of 200. Under the current specification, the race is guaranteed to be lost. Most existing endpoints will emit a 408 for a non-INVITE request 64*T1 after receiving the request if they have not emitted an earlier final response. Such a 408 is guaranteed to arrive at the next upstream element too late to be useful. In fact, in the presence of proxies, these messages are even harmful. When the 408 arrives, each proxy will have already terminated its associated client transaction due to timeout. Therefore, each proxy must forward the 408 upstream statelessly. This, in turn, is guaranteed to arrive too late. As Figure 3 shows, this can ultimately result in bombarding the original requester with spurious 408s. (Note that the proxy's client transaction state machine never enters the Completed state, so Timer K does not enter into play.)

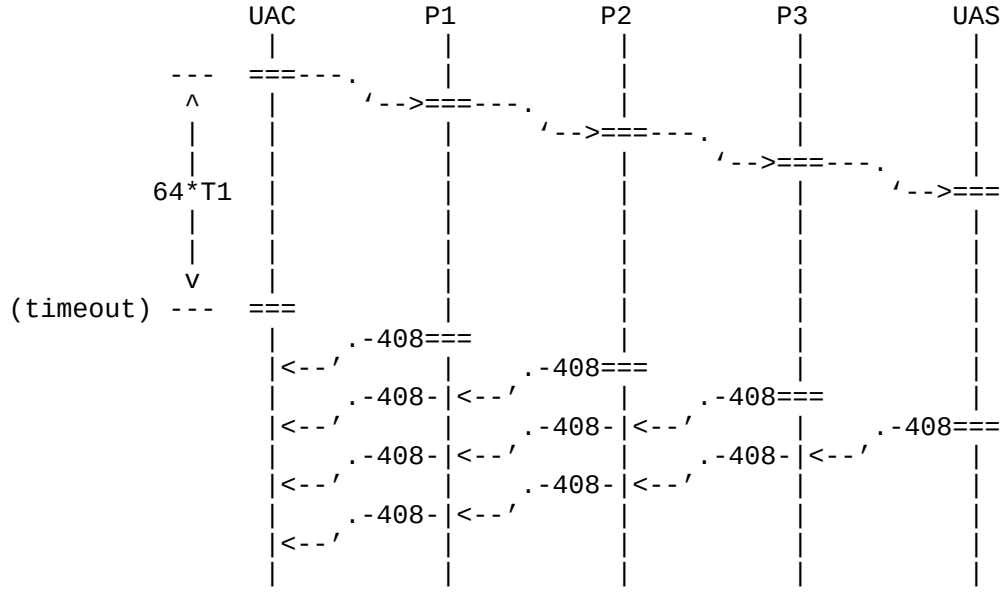

Figure 3: Late 408s to Non-INVITEs

This response bombardment is not limited to the 408 response, though it only exists when participating client transaction state machines are timing out. Figure 4 generalizes Figure 1 to include multiple hops. Note that even though the UAS responds "in time" to P3, the response is too late for P2, P1, and the UAC.

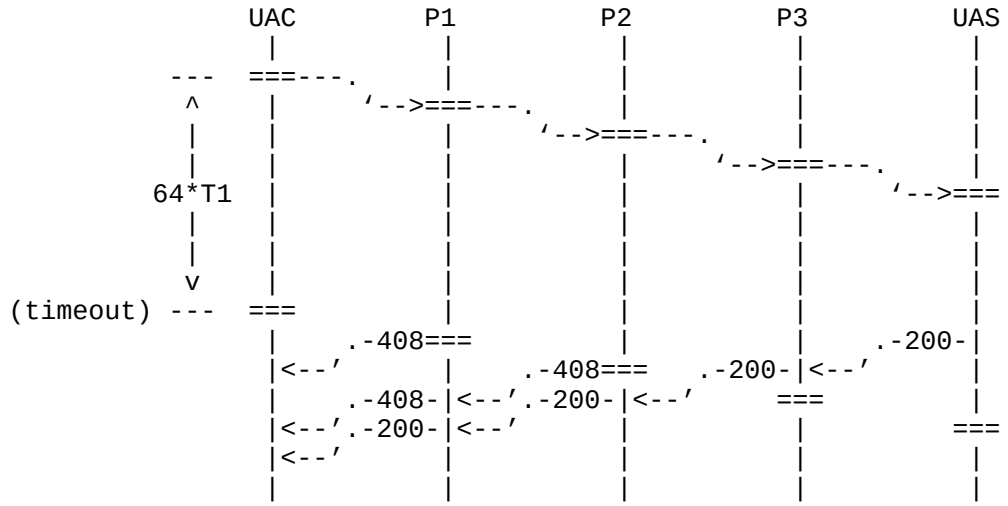

Figure 4: Additional Timeout-Related Error

1.5. Non-INVITE timeouts doom forking proxies

A single branch with a delayed or missing final response will dominate the processing at proxy that receives no 2xx responses to a forked non-INVITE request. This proxy is required to allow all of its client transactions to terminate before choosing a "best response". This forces the proxy's server transaction to lose the race in Figure 1. Any response it ultimately forwards (a 401, for example) will arrive at the upstream elements too late to be used. Thus, if no element among the branches would return a 2xx response, failure of a single element (or its transport) dooms the proxy to failure.

1.6. Mismatched timer values make winning the race harder

There are many failure scenarios due to misconfiguration or misbehavior that the SIP specification does not discuss. One is placing two elements with different configured values for T1 and T2 on the same network. Review of Figure 1 illustrates that the race failure is only made more likely in this misconfigured state (it may appear that shortening T1 at the element behaving as a UAS improves this particular situation, but remember that these elements may trade roles on the next request). Since the protocol provides no mechanism for discovering/negotiating a peer's timer values, exceptional care must be taken when deploying systems with non-defaults to ensure that they will never directly communicate with elements with default values.

2. Security Considerations

This document describes some problems in the core SIP specification [1] related to the SIP non-INVITE requests, the messages other than INVITE that begin transactions. A few of the problems lead to flooding or forgery risk, and could possibly be exploited by an adversary in a denial of service attack. Solutions are defined in the companion document [4].

One solution there prohibits proxies and User Agents from sending 408 responses to non-INVITE transactions. Without this change, proxies automatically generate a storm of useless responses. An attacker could capitalize on this by enticing User Agents to send non-INVITE requests to a black hole (through social engineering or DNS poisoning) or by selectively dropping responses.

Another solution prohibits proxies from forwarding late responses. Without this change, an attacker could easily forge messages which appear to be late responses. All proxies compliant with RFC 3261 are required to forward these responses, wasting bandwidth and CPU and potentially overwhelming target User Agents (especially those with low speed connections).

3. Acknowledgements

This document captures many conversations about non-INVITE issues. Significant contributers include Ben Campbell, Gonzalo Camarillo, Steve Donovan, Rohan Mahy, Dan Petrie, Adam Roach, Jonathan Rosenberg, and Dean Willis.

4. Informative References

[1] 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.

[2] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[3] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[4] Sparks, R., "Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction", RFC 4320, January 2006.

Author's Address

Robert J. Sparks
Estacado Systems
17210 Campbell Road
Suite 250
Dallas, TX 75252-4203

EMail: rjsparks@estacado.net

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).