Network Working Group
Internet-Draft
Updates: 3261 (if approved)
Intended status: Standards Track
Expires: March 15, 2010

R. Sparks
Tekelec
T. Zourzouvillys
VoIP.co.uk

Sept 11, 2009

Correct transaction handling for 200 responses to Session Initiation Protocol INVITE requests
draft-sparks-sipcore-invfix-01

Status of this Memo

Copyright Notice

Abstract

This document normatively updates RFC 3261, the Session Initiation

Protocol (SIP), to address an error in the specified handling of success (200 class) responses to INVITE requests. Elements following RFC 3261 exactly will misidentify retransmissions of the request as a new, unassociated, request. The correction involves modifying the INVITE transaction state machines. The correction also changes the way responses that cannot be matched to an existing transaction are handled to address a security risk.

Table of Contents

1. Conventions and Definitions

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].

2. Introduction

This document describes an essential correction to the Session Initiation Protocol (SIP), defined in [RFC3261], using the process defined in [I-D.drage-sip-essential-correction]. The change addresses an error in the handling of 200 class responses to INVITE requests that leads to retransmissions of the INVITE being treated as new requests and forbids forwarding stray INVITE responses.

3. Reason for Change

One use of the INVITE method in SIP is to establish new sessions. These "initial" INVITEs may fork at intermediaries, and more than one receiving endpoint may choose to accept the request. SIP is designed such that the requester receives all of these success responses.

Two sets of requirements in [RFC3261] work together to allow multiple 200s to be processed correctly by the requester. First, all elements are required to immediately destroy any INVITE client transaction state upon forwarding a matching 200 OK response. This requirement applies to both proxies and user agents (proxies forward the response upstream, the transaction layer at user agents forward the response to its "UA core"). Second, all proxies are required to statelessly forward any 200 OK responses that do not match an existing transaction, also called stray responses, upstream. The transaction layer at user agents is required to forward these responses to its UA core. Logic in the UA core deals with acknowledging each of these responses.

This technique for specifying the behavior was chosen over adjusting INVITE client transaction state machines as a simpler way to specify the correct behavior.

Over time, implementation experience demonstrated the existing text is in error. Once any element with a server transaction (say, a proxy in the path of the INVITE) deletes that transaction state, any retransmission of the INVITE will be treated as a new request, potentially forwarded to different locations than the original. Many implementations in the field have made proprietary adjustments to their transaction logic to avoid this error.

The requirement to statelessly forward stray responses has also been identified as a security risk. Through it, elements compliant to [RFC3261] are compelled to do work (forward packets) that is not protected by the admission policies applied to requests. This can be leveraged to, for instance, use a SIP proxy as an anonymizing forwarder of packets in a distributed DOS attack. General internet endpoints can also collude to tunnel non-SIP content through such proxies by wrapping them in an SIP response envelope.

Additionally, [RFC3261] requires that if an unrecoverable transport error is encountered while sending a response in a client transaction, that the transaction moves immediately into the Terminated state. This will result in any re-transmitted INVITE requests received after such an error was encountered be processed as a new request instead of being absorbed as a re-transmission.

4. Summary of Change

This correction document updates [RFC3261], adding a state and changing the transitions in the INVITE client state machine such that the INVITE client transaction remains in place to receive multiple 200 OK responses. It adds a state to the INVITE server state machine to absorb retransmissions of the INVITE after a 200 OK response has been sent. It modifies state transitions in the INVITE server state machine to absorb retransmissions of the INVITE request after encountering a unrecoverable transport error when sending a response. It also forbids forwarding stray responses to INVITE requests (not just 200 OK responses), which RFC3261 requires.

5. Consequences if Not Approved

Implementations strictly conformant to [RFC3261] will process retransmitted initial INVITE requests as new requests. Proxies may forward them to different locations than the original. Proxies may also be used as anonymizing forwarders of bulk traffic. Implementations will process any retransmitted INVITE request as new request after an attempt to send a response resulted in a unrecoverable error.

6. The Change

An element sending or receiving a 200 OK to an INVITE transaction MUST NOT destroy any matching INVITE transaction state. This state is necessary to ensure correct processing of retransmissions of the request and the retransmission of the 200 OK and ACK that follow.

An element encountering an unrecoverable tranport error when trying to send a response to an INVITE request MUST NOT immediately destroy the associated INVITE server transaction state. This state is necessary to ensure correct processing of retransmissions of the request.

When receiving any SIP response, a transaction-stateful proxy MUST compare the transaction identifier in that response against its existing transaction state machines. The proxy MUST NOT forward the response if there is no matching transaction state machine.

When receiving an ACK that matches an existing INVITE server transaction, and the ACK does not contain a branch parameter containing the magic cookie defined in RFC 3261, the matching transaction MUST be checked to see if it is in the "Accepted" state. If it is, then the ACK must be passed directly to the transaction user instead of absorbing it in the transaction. This is necessary as requests from RFC 2543 clients will not include a unique branch parameter, and the mechanisms for calculating the transaction id from such a request will be the same for both INVITE and ACKs.

7. Change Details

These changes impact requirements in several sections of RFC3261. The exact effect on that text is detailed in Section 8. This section describes the details of the change, particularly the impact on the INVITE state machines, more succinctly to facilitate review and simplify implementation.

7.1. Server Transaction Impacts

To allow a SIP element to recognize retransmissions of an INVITE as retransmissions instead of new requests, a new state, "Accepted", is added to the INVITE server transaction state machine. A new timer, Timer L, is also added to ultimately allow the state machine to terminate. A server transaction in the "Proceeding" state will transition to the "Accepted" state when it issues a 2xx response, and will remain in that state just long enough to absorb any retransmissions of the INVITE.

If the SIP elements's TU issues a 2xx response for this transaction while the state machine is in the "Proceeding" state, it MUST transition to the "Accepted" state and set Timer L to 64*T1.

While in the "Accepted" state, any retransmissions of the INVITE received will match this transaction state machine and will be absorbed by the machine without changing its state. These

retransmissions are not passed onto the TU. RFC3261 requires the TU to periodically retransmit the 2xx response until it receives an ACK. The server transaction MUST NOT generate 2xx retransmissions on its own. Any retransmission of the 2xx response passed from the TU to the transaction while in the "Accepted" state MUST be passed to the transport layer for transmission. Any ACKs received from the network while in the "Accepted" state MUST be passed directly to the TU and not absorbed.

When Timer L fires and the state machine is in the "Accepted" state, the machine MUST transition to the "Terminated" state. Once the transaction is in the "Terminated" state, it MUST be destroyed immediately. Timer L reflects the amount of time the server transaction could receive 2xx responses for retransmission from the TU while it is waiting to receive an ACK.

A server transaction MUST NOT discard transaction state based only on encountering a non-recoverable transport error when sending a response. Instead the assocated INVITE server transaction state machine MUST remain in its current state. (Timers will eventually cause it to transition to the Terminated state). This allows retransmissions of the INVITE to be absorbed instead of being processed as a new request.

Figure 1 and Figure 2 graphically show the parts of the INVITE server state machine that has changed. The entire new INVITE server state machine is shown in Figure 5.

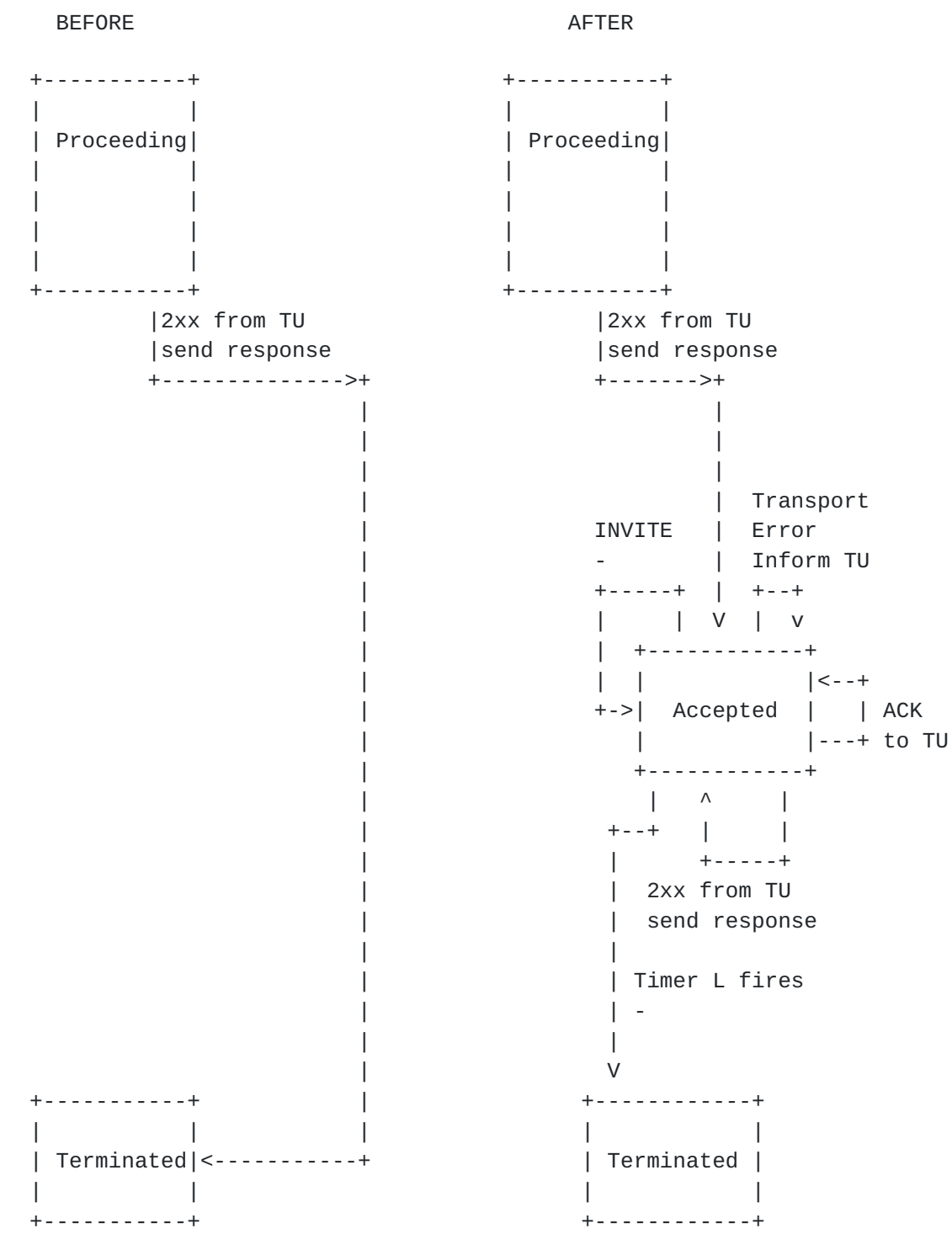

Figure 1: Changes to the INVITE server transaction state machine when sending 2xx

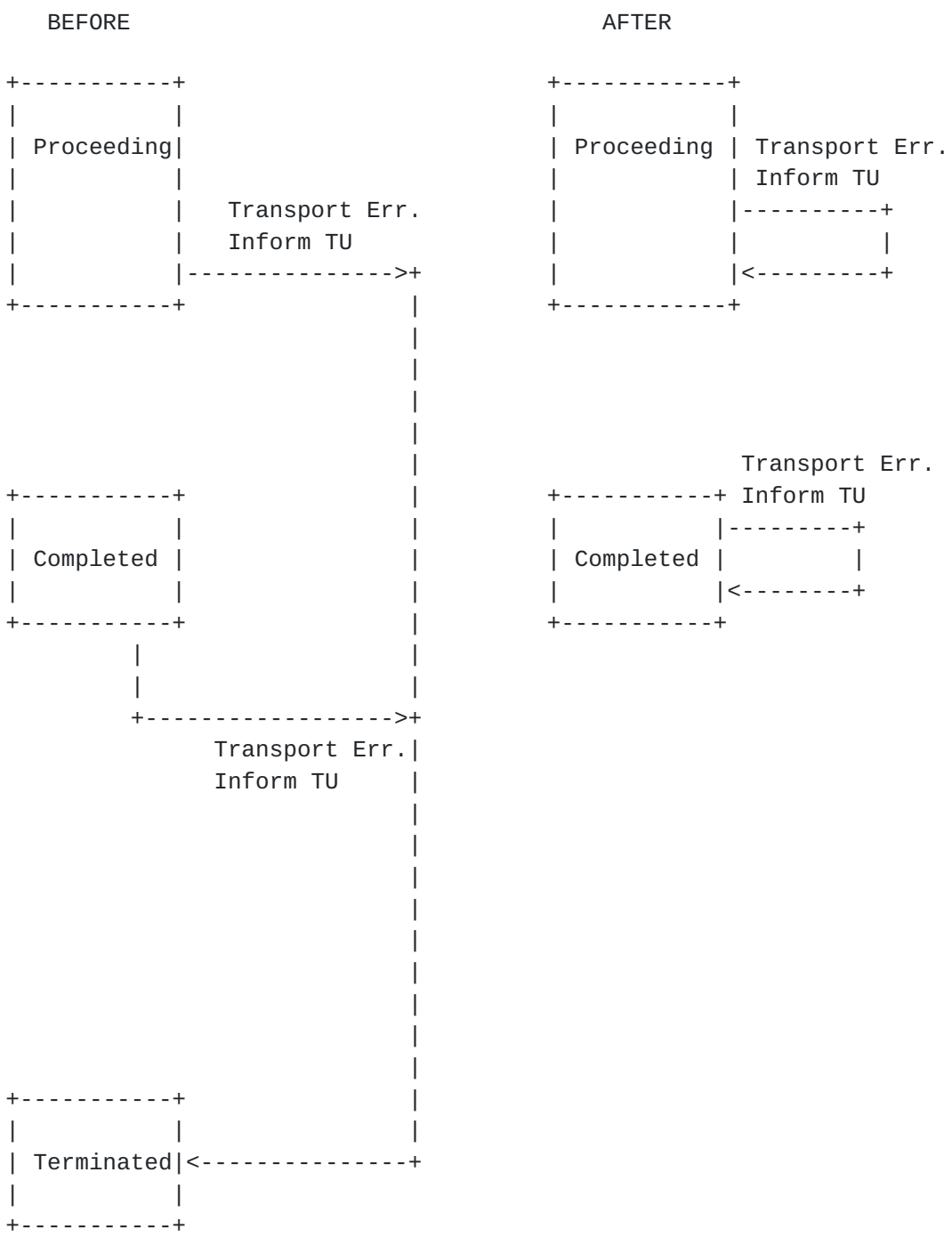

Figure 2: Changes to the INVITE server transaction state machine on encountering transport error

7.2. Client Transaction Impacts

In order to correctly distinguish retransmissions of 2xx responses from stray 2xx responses, the INVITE client state machine is modified to not transition immediately to "Terminated" on receipt of a 2xx response. Instead, the machine will transition to a new "Accepted" state, and remain there just long enough, determined by a new timer M, to receive and pass to the TU any retransmissions of the 2xx response or any additional 2xx responses from other branches of a downstream fork of the matching request. If a 2xx response is received while the client INVITE state machine is in the "Calling" or "Proceeding" states, it MUST transition to the "Accepted" state, pass the 2xx response to the TU, and set Timer M to 64*T1. A 2xx response received while in the "Accepted" state MUST be passed to the TU and the machine remains in the "Accepted" state. The client transaction MUST NOT generate an ACK to any 2xx response on its own. The TU responsible for the transaction will generate the ACK.

When Timer M fires and the state machine is in the "Accepted" state, the machine MUST transition to the "Terminated" state. Once the transaction is in the "Terminated" state, it MUST be destroyed immediately.

Any response received which does not match an existing client transaction state machine is simply dropped. (Implementations are, of course, free to log or do other implementation specific things with such responses, but the implementer should be sure to consider the impact of large numbers of malicious stray responses).

Note that it is not necessary to preserve client transaction state upon the detection of unrecoverable transport errors. Existing requirements ensure the TU has been notified, and the new requirements in this document ensure that any received retransmitted response will be dropped since there will no longer be any matching transaction state.

Figure 3 graphically shows the part of the INVITE client state machine that has changed. The entire new INVITE client state machine is shown in Figure 4.

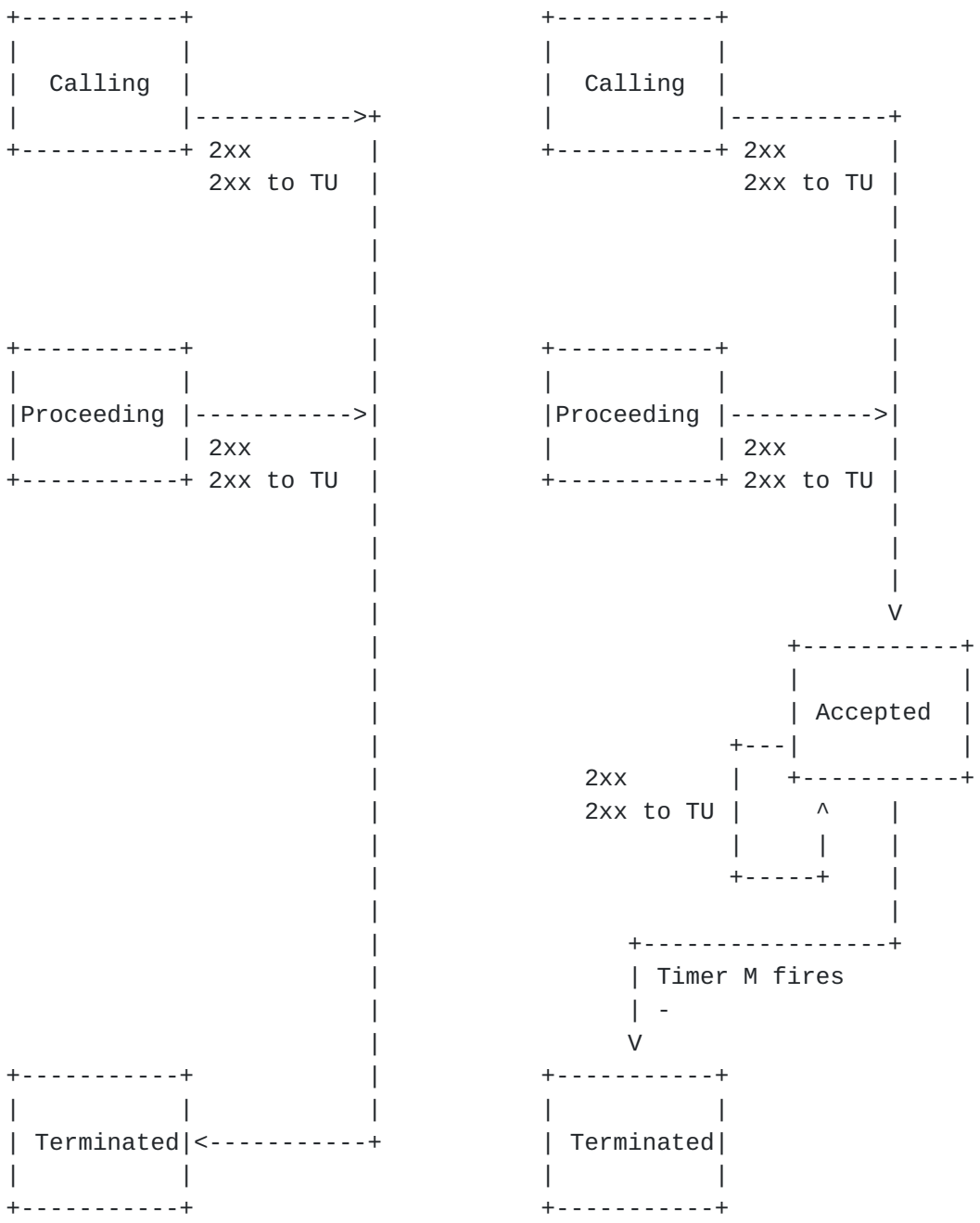

Figure 3: Changes to the INVITE client transaction state machine

7.3. Proxy Considerations

This document changes the behaviour of transaction-stateful proxies to not forward stray INVITE responses. When receiving any SIP response, a transaction-stateful proxy MUST compare the transaction identifier in that response against its existing transaction state machines. The proxy MUST NOT forward the response if there is no

matching transaction state machine.

8. Exact changes to RFC3261

This section describes exactly the same changes as above, but shows exactly which text in RFC3261 is affected.

8.1. Page 85

Section 13.3.1.4 paragraph 4 is replaced entirely by

> Once the response has been constructed, it is passed to the INVITE server transaction. In order to ensure reliable end-to-end transport of the response, it is necessary to periodically pass the response directly to the transport until the ACK arrives. The 2xx response is passed to the transport with an interval that starts at T1 seconds and doubles for each retransmission until it reaches T2 seconds (T1 and T2 are defined in Section 17). Response retransmissions cease when an ACK request for the response is received. This is independent of whatever transport protocols are used to send the response.

8.2. Page 107

Section 16.7 paragraphs 1 and 2 are replaced entirely by

> When a response is received by an element, it first tries to locate a client transaction (Section 17.1.3) matching the response. If a transaction is found, the response is handed to the client transaction. If none is found, the element MUST NOT forward the response.

8.3. Page 114

Section 16.7, part 9, first paragraph. Replace this sentence

> If the server transaction is no longer available to handle the transmission, the element MUST forward the response statelessly by sending it to the server transport.

with

> If the server transaction is no longer available to handle the transmission, the response is simply discarded.

8.4. Pages 126 through 128

Section 17.1.1.2. Replace paragraph 7 (starting "When in either") through the end of the section with

> When in either the "Calling" or "Proceeding" states, reception of a response with status code from 300-699 MUST cause the client transaction to transition to "Completed". The client transaction MUST pass the received response up to the TU, and the client transaction MUST generate an ACK request, even if the transport is reliable (guidelines for constructing the ACK from the response are given in Section 17.1.1.3) and then pass the ACK to the transport layer for transmission. The ACK MUST be sent to the same address, port, and transport to which the original request was sent.
>
> The client transaction MUST start timer D when it enters the "Completed" state for any reason, with a value of at least 32 seconds for unreliable transports, and a value of zero seconds for reliable transports. Timer D reflects the amount of time that the server transaction can remain in the "Completed" state when unreliable transports are used. This is equal to Timer H in the INVITE server transaction, whose default is 64*T1, and is also equal to the time a UAS core will wait for an ACK once it sends a 2xx response. However, the client transaction does not know the value of T1 in use by the server transaction or any downstream UAS cores, so an absolute minimum of 32s is used instead of basing Timer D on T1.
>
> Any retransmissions of a response with status code 300-699 that are received while in the "Completed" state MUST cause the ACK to be re-passed to the transport layer for retransmission, but the newly received response MUST NOT be passed up to the TU.
>
> A retransmission of the response is defined as any response which would match the same client transaction based on the rules of Section 17.1.3.
>
> If timer D fires while the client transaction is in the "Completed" state, the client transaction MUST move to the "Terminated" state.
>
> When a 2xx response is received while in either the "Calling" or "Proceeding" states, the client transaction MUST transition to the "Accepted" state, and Timer M MUST be started with a value of 64*T1. The 2xx response MUST be passed up to the TU. The client transaction MUST NOT generate an ACK to the 2xx response - its handling is delegated to the TU. A UAC core will send an ACK to

the 2xx response using a new transaction. A proxy core will always forward the 2xx response upstream.

The purpose of the "Accepted" state is to allow the client transaction to continue to exist to receive, and pass to the TU, any retransmissions of the 2xx response and any additional 2xx responses from other branches of the INVITE if it forked downstream. Timer M reflects the amount of time that transaction user will wait for such messages.

Any 2xx responses matching this client transaction that are received while in the "Accepted" state MUST be passed up to the TU. The client transaction MUST NOT generate an ACK to the 2xx response. The client transaction takes no further action.

If timer M fires while the client transaction is in the "Accepted" state, the client transaction MUST move to the "Terminated" state.

The client transaction MUST be destroyed the instant it enters the "Terminated" state.

Replace Figure 5 with

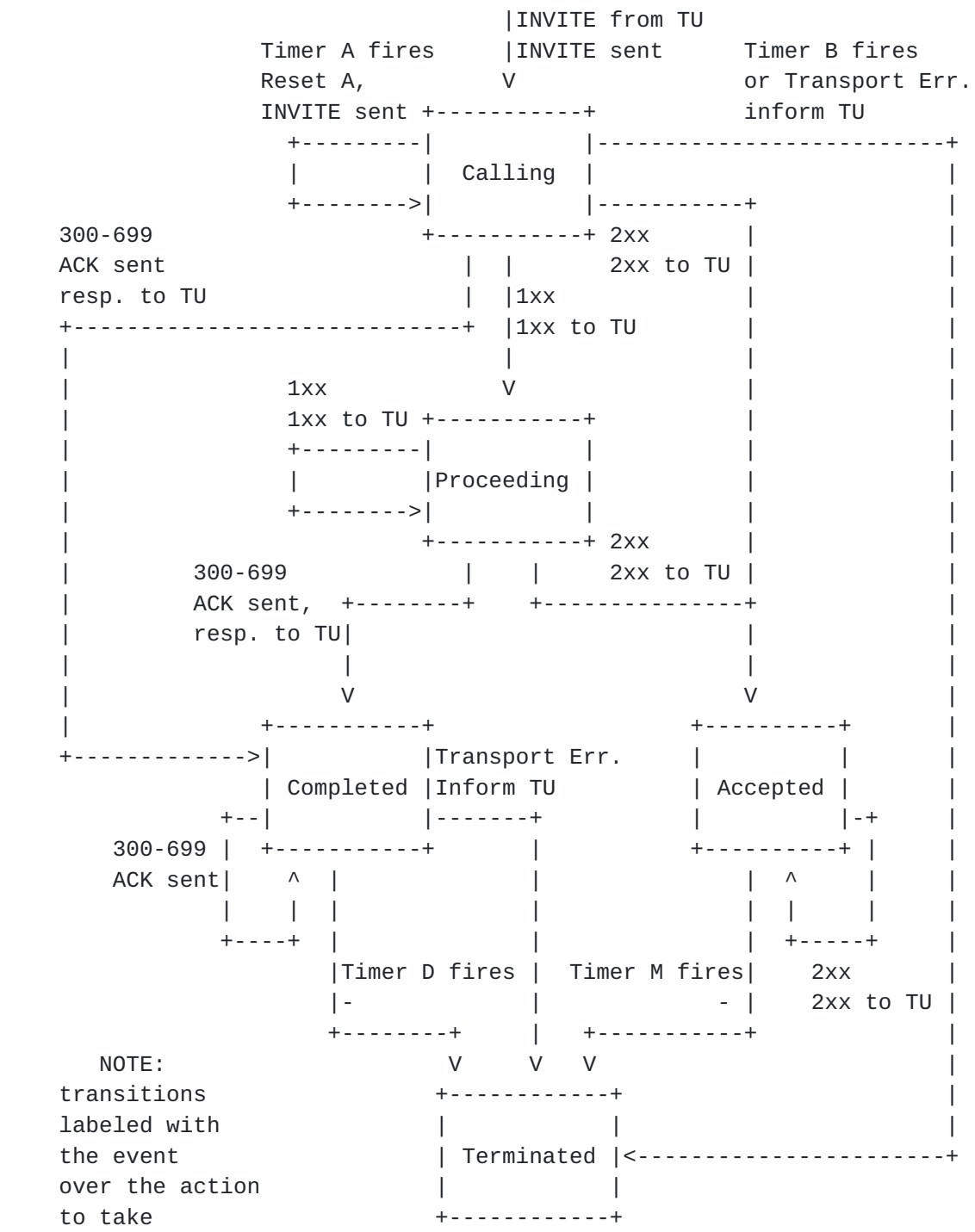

Figure 4: INVITE client transaction

8.5. Pages 134 to 135

Section 17.2.1 paragraph 4 is replaced with

> If, while in the "Proceeding" state, the TU passes a 2xx response to the server transaction, the server transaction MUST pass this response to the transport layer for transmission. It is not retransmitted by the server transaction; retransmissions of 2xx responses are handled by the TU. The server transaction MUST then transition to the "Accepted" state.

8.6. Page 136

Replace Figure 7 with

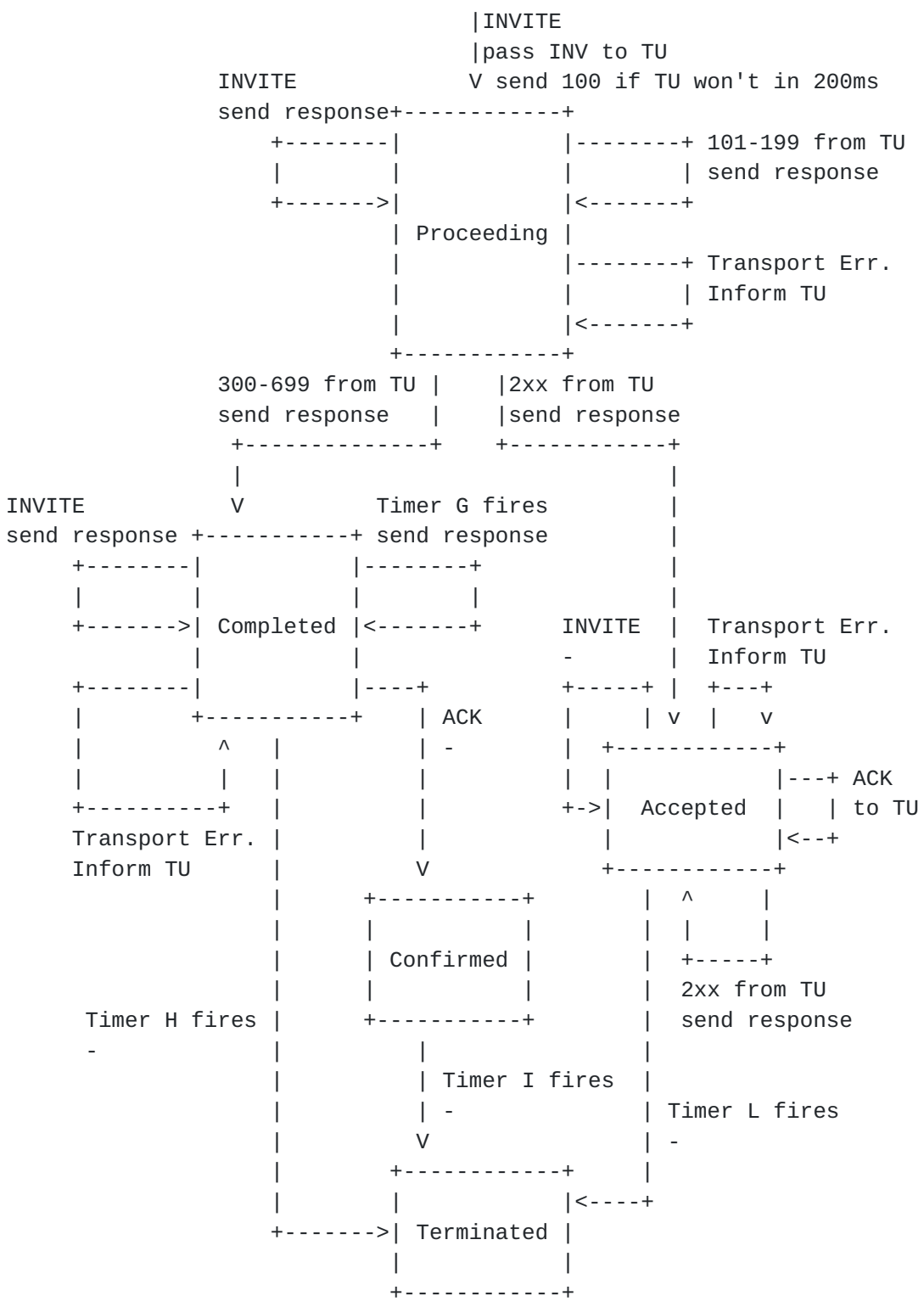

Figure 5: INVITE server transaction

8.7. Page 137

Section 17.2.1 - Replace the last paragraph (starting "Once the transaction") with

> The purpose of the "Accepted" state is to absorb retransmissions of an accepted INVITE request. Any such retransmissions are absorbed entirely within the server transaction. They are not passed up to the TU since any downstream UAS cores that accepted the request have taken responsibility for reliability and will already retransmit their 2xx responses if neccessary.
>
> While in the "Accepted" state, if the TU passes a 2xx response, the server transaction MUST pass the response to the transport layer for transmission.
>
> When the INVITE server transaction enters the "Accepted" state, Timer L MUST be set to fire in 64*T1 for all transports. This value matches both Timer B in the next upstream client state machine (the amount of time the previous hop will wait for a response when no provisionals have been sent) and the amount of time this (or any downstream) UAS core might be retransmitting the 2xx while waiting for an ACK. If an ACK is received while the INVITE server transaction is in the "Accepted" state, then the ACK must be passed up to the TU. If Timer L fires while the INVITE server transaction is in the "Accepted" state, the transaction MUST transition to the "Terminated" state.
>
> Once the transaction is in the "Terminated" state, it MUST be destroyed immediately.

8.8. Page 141

Section 17.2.4 - Replace the second paragraph with

> First, the procedures in [4] are followed, which attempt to deliver the response to a backup. If those should all fail, based on the definition of failure in [4], the server transaction SHOULD inform the TU that a failure has occurred, and MUST remain in the current state.

8.9. Page 144

Section 18.1.2 - Replace the second paragraph with

The client transport uses the matching procedures of Section 17.1.3 to attempt to match the response to an existing transaction. If there is a match, the response MUST be passed to that transaction. Otherwise, any element other than a stateless proxy MUST silently discard the response.

8.10. Page 146

Section 18.2.1 - Replace the last paragraph with

Next, the server transport attempts to match the request to a server transaction. It does so using the matching rules described in Section 17.2.3. If a matching server transaction is found, the request is passed to that transaction for processing. If no match is found, the request is passed to the core, which may decide to construct a new server transaction for that request.

8.11. Page 265

Add to Table 4:

Timer L	64*T1	Section 17.2.1	Wait time for accepted INVITE request retransmits
Timer M	64*T1	Section 17.1.1	Wait time for retransmission of 2xx to INVITE or additional 2xx from other branches of a forked INVITE

9. IANA Considerations

None.

10. Security Considerations

This document makes two changes to the Session Initiation Protocol to address the error discussed in Section 3. It changes the behavior of both the client and server INVITE transaction state machines, and it changes the way "stray" responses (those that don't match any existing transaction) are handled at transaction stateful elements.

The changes to the state machines cause elements to hold onto each

accepted INVITE transaction state longer (32 seconds) than what was specified in RFC 3261. This will have a direct impact on the amount of work an attacker leveraging state exhaustion will have to exert against the system. However, this additional state is necessary to achieve correct operation.

RFC 3261 required SIP proxies to forward any stray 200 class responses to an INVITE request upstream statelessly. As a result, conformant proxies can be forced to forward packets (that look sufficiently like SIP responses) to destinations of the sender's choosing. Section 3 discusses some of the malicious behavior this enables. This document reverses the stateless forwarding requirement, making it a violation of the specification to forward stray responses.

RFC 3261 defines a "stateless proxy" which forwards requests and responses without creating or maintaining any transaction state. The requirements introduced in this document do not change the behavior of these elements in any way. Stateless proxies are inherently vulnerable to the abuses discussed in Section 3. One way operators might mitigate this vulnerability is to carefully control which peer elements can present traffic to a given stateless proxy.

The changes introduced by this document are backward-compatible. Transaction behavior will be no less correct, and possible more correct, when only one peer in a transaction implements these changes. Except for the considerations mentioned earlier in this section, introducing elements implementing these changes into deployments with RFC 3261 implementations adds no additional security concerns.

11. Acknowledgments

Pekka Pessi reported the improper handling of INVITE retransmissions. Brett Tate performed a careful review uncovering the need for the Accepted state and Timer M in the client transaction state machine. Jan Kolomaznik noticed that a server transaction should let a TU know about transport errors when it attempts to send a 200-class response. Michael Procter corrected several nits.

12. References

12.1. 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.

12.2. Informative References

[I-D.drage-sip-essential-correction]
Drage, K., "A Process for Handling Essential Corrections to the Session Initiation Protocol (SIP)", draft-drage-sip-essential-correction-03 (work in progress), July 2008.

Authors' Addresses

Robert Sparks
Tekelec
17210 Campbell Road
Suite 250
Dallas, Texas 75252
USA

Email: RjS@nostrum.com

Theo Zourzouvillys
VoIP.co.uk
Commerce House
Telford Rd
Bicester, Oxfordshire OX26 6BU
UK

Email: theo@crazygreek.co.uk