SIPCORE
Internet-Draft
Updates: RFC 3263 (if approved)
Intended status: Standards Track
Expires: January 19, 2017

D. Worley
Ariadne Internet Services

July 18, 2016

Contacting Session Initiation Protocol (SIP) Servers in a Dual-Stack IP Network
draft-worley-sipcore-dual-stack-01

Abstract

In a dual-stack (IPv4 and IPv6) environment, the procedures of RFC 3263 by which a Session Initiation Protocol (SIP) client contacts a server may not suffice to provide a good user experience. This document describes "Happy Eyeballs" modifications -- modifications of the procedures of RFC 3263, as well as additional client procedures -- which improve the SIP user experience in many circumstances.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on January 19, 2017.

Copyright Notice

Table of Contents

1. Introduction

The sections of this document cover a number of topics which arise in dual-stack environments. As this document matures, some of these topics may be split into seperate documents. The current text is a very rough draft, including proposed requirements, proposed solutions, observations about SIP systems in practice, and design discussions.

2. Target Selection

2.1. Requirements

- Solve the "happy eyeballs" problem for INVITE, i.e., if a destination URI is advertised with both multiple targets (IPv4 and IPv6 or otherwise), but connectivity to only one target exists, usually message transmission will not be delayed unacceptably, and in particular, not by a full timeout as prescribed in 3261.

- Note that the solution may be decidedly different based on the transport protocol(s) for the targets.

- There is uncertainty about whether adding an RTT to signaling times is acceptable. For example, in practice RTT on the overall Internet is less than 1/2 second, and that is an acceptable delay in call setup times under ordinary circumstances.

- The solution must "approximate" 3263, in that if several targets are specified by *different* SRV records and are reachable over a long period of time, the relative traffic shares sent to them must be compatible with the priorities and weights of the SRV records. (If they are alternatives that are *not* derived from different SRV records, the solution is unconstrained regarding relative traffic shares.)

2.2. Terminology

a "client" is the entity that wishes to send a SIP message

a "target" or "transport target" is an address/port/protocol triple that is an address for the transport layer of the stack. A target is derived from a SIP URI (for a request) or a host-port (for a response).

a "flow" is a sequence of related messages between the client and a target. If the protocol is connection-oriented, the flow encompasses the connection. If the protocol requires cryptographic setup, the flow encompasses the cryptographic session.

a "probe" is an operation executed on a flow by a client to determine whether it can successfully communicate with the target, without changing the SIP dialog state with the target. Probes can take many forms:

- Setting up a connection of a connection-oriented protocol.

- Performing a cryptographic handshake.

- Performing a keep-alive as defined by the protocol.

- Sending an OPTIONS request with Max-Forwards 0.
- Sending a CR-LF-CR-LF keep-alive on a connection as described in [RFC5626].
- Sending a STUN keep-alive message using a datagram protocol as described in [RFC5626].

Note that the sending an OPTIONS request can be used with any(!) protocol. If the OPTIONS reaches the target, the target is required to respond with either a 200 or 483 response (without forwarding it to another entity). Conveniently, a server can respond to such a request statelessly, so such requests are low-overhead. (Although the [RFC5626] keep-alive methods are even lower overhead.)

2.3. Solution

The current state of the solution (as I know of it) is:

Note that some SIP messages are time-sensitive for the usesr experience (e.g., initial INVITEs), while others are not (e.g., a refreshing REGISTER). A client MAY choose not to apply the following rules for non-time-sensitive messages.

Devices MAY change the target order prescribed by RFC 3263/2782. The device SHOULD follow the Happy Eyeballs rules, viz.:

- Prefer targets with existing flows
- Prefer targets with a different address family than that of a non-responsive address
- Deprecate targets known to be non-responsive
- May simultaneously initiate flows with multiple targets as long as UA does not have more than one simultaneous outstanding copy of a request
- Prefer simultaneously initiating flows with targets in different address families

Devices MAY contact targets in any order, including those obtained via different SRV records, notwithstanding the priority/weight specified in the SRV records. But in doing this, they MUST approximate the behavior specified by RFC 3263, in this sense:

> If a set of targets is all of the targets derived from two or more SRV records, and at least one target for each SRV record is

reachable by the client over a long period of time, the relative shares of traffic sent to each subset of targets derived from one SRV record must converge to the traffic shares that would result if the client contacted these targets in the order specified by RFC 3263 (i.e., according to the priorities and weights of the SRV records).

(Note that the relative traffic shares between targets that are *not* derived from different SRV records (e.g., alternative A records for a DNS name) are not constrained by this requirement.)

In general, this means that cached reachability information about targets should time out, causing the behavior of the client to revert to RFC 3263 over time.

(Beware that we have to define "reachable" above to include responsiveness -- a high-priority target that has a 5 sec RTT shouldn’t be able to commandeer all of the traffic.)

If a client does not have recent reachability information for the flow to a given target, the client SHOULD probe the flow before sending a request to the target.

This is because in the worst case, sending a request commits the client to waiting for a timeout before it can send a duplicate request to another target. Note that probes do not change the SIP dialog state of any entity, so probes can be sent in parallel to multiple targets.

Reduce client transaction timeouts: Timer B and Timer F are currently 64*T1, which defaults to 32 seconds

It seems that reducing the default T1 from 500 msec to 100 msec suffices for this. It seems that RTT to arbitrary places on the Internet can take as long as 500 msec, but RTT to web servers generally takes 100 msec or less. That argues for reducing T1 to 100 msec, which makes timers B/F 6.4 sec. In practice, SIP servers are likely to have connectivity like web servers. But we want global public SIP to work (e.g., in peer-to-peer SIP), so SIP to arbitrary addresses should only rarely time out.

The retransmission schedule specified by RFC 3261 is:

1st send is at time 0
2nd send is at time T1
3rd send is at time 3*T1
4th send is at time 7*T1
5th send is at time 15*T1
6th send is at time 31*T1
7th send is at time 63*T1
timer B fires at time 64*T1, terminating the transmission

This is just long enough to allow 7 transmissions of the message. If we reduce T1 to 100 msec (from 500 msec), the total length of the schedule is reduced to 6.4 sec, which seems tolerable as a fail-over delay. It still alllows 7 retransmissions.

2.3.1. Problems with reducing T1

Brett Tate notes that there are problems with reducing T1:

- Brett: The issue with using a smaller T1 value is that it doesn't just impact the desired timer. For instance, it can cause more retries. Because of this, the draft might want to specify allowing the specific smaller timers in a way which doesn't rely upon a smaller T1.

  If it matters, RFC 3261 section 17.1.1.2 has normative statements concerning the topic of lowing T1.

- Dale: It seems to me that the core concept of that section is "T1 is an estimate of the round-trip time (RTT)". And while the average RTT over the global Internet is nearly 500ms, the RTT to places that expect a lot of traffic seems to be 100ms or less. (I expect RTT on carrier-managed networks to be even less.) So I don't see cutting the default T1 as contravening that section.

  True, that does increase the number of retries, but that seems to be the correct change if it's true that if you don't see a response in 100ms, it is more likely that the request was lost than that it is delayed in transit.

  Do we have any data on what RTT and packet loss is like in real systems?

-

Brett: Although this draft can update it if needed, RFC 3261 section 17.1.1.2 paragraph four was the text of potential concern since it looks like an attempt to limit where T1 can be lowered.

"Elements MAY (though it is NOT RECOMMENDED) use smaller values of T1 within closed, private networks that do not permit general Internet connection. T1 MAY be chosen larger, and this is RECOMMENDED if it is known in advance (such as on high latency access links) that the RTT is larger."

It seems like the extra retries would be an unnecessary burden upon the next hop. However, I guess that it depends upon if more concerned with 1) dropped packets or 2) the retries increasing the frequency of devices becoming overloaded.

Unless this draft updates the RFC 4320 behavior, there can be many retries before the next hop can return 100 for non-INVITE requests.

As mentioned, the smaller T1 value would not just impact the desired timers. It would impact other letter timers within RFC 3261 and RFC 6026 such as Timer L, Timer M, Timer H, Timer J, and as mentioned Timer A, Timer E, and Timer G. Should these timers really be reduced? For instance, is it acceptable for UAS to use T1 of 100ms when the UAC and intermediaries are using T1 of 500ms?

o

Roman: This all depends what you mean by the "real systems". For USA hosted PBX systems working with office based networks, RTT is 70ms (typical delay from east coast USA to LA datacenter) or less. If service is geo optimized between two or more data centers, RTT is 40ms or less. Packet loss is nearly zero. These conditions account for majority of hosted PBX traffic.

If you are dealing with international, RTT is around 500ms or less for most locations. Brazil to Singapore is around 420ms RTT, for instance. Western Europe to LA is around 150ms RTT. There are, of cause, locations where RTT is much higher, especially when satellite links are involved.

Packet loss heavily depends on the last mile link utilization. There are still links which are 128K which are used for VoIP. If someone starts uploading things on this link, you start seeing packet loss or 2-3 sec packet delays. Most of the time,

form locations with broadband connections, packet loss is close to zero.

If you start dealing with mobile, things are a lot less predictable. Packet loss can be in 10-20% range. RTT delays can be up to a second. Some other locations, such as hotels, especially internationally, are often even worse then mobile due to random traffic blocking or deliberate attempts to block VoIP.

3. Client-side NAT

3.1. Requirements

- Clients behind client-side NATs must be fully supported.
- The solution should be as compatible with SIP Outbound as practicable.
- The difficult part of this is "the solution for simple UDP (i.e., not using SIP Outbound)". There is a strong perception that in systems where an edge proxy services 10^5 or 10^6 UAs that only UDP-without-Outbound has a low enough per-flow overhead to be workable. The important requirements for simple UDP seem to be:
  - The per-UA state in the edge proxy must be no larger than about what is kept for a registration.
  - The update rate of the per-UA state in the edge proxy must be no larger than about the update rate for a registration.
- It is perceived that TCP, TLS, and even UDP-with-Outbound do not meet these requirements. A substantial fraction of the UAs tested in the latest SIPit do not support TCP, TLS, or Outbound, showing that there is substantial market presence of simple-UDP-only UAs.
- It is possible that a "Simple Outbound" can be defined that provides the needed part of the functionality of Outbound at a sufficiently low overhead and complexity that it meets these requirements. If so, it is desirable that it be conceptually and operationally upward-compatible with Outbound.

3.2. Discussion

It’s clear to me that the problem is *solvable*, because existing SIP systems do handle the client-side NAT problem. E.g., the open-source sipX system has full client-side NAT support. That scheme doesn’t require SIP Outbound support in the client at all. NAT support is

triggered by the client's requests arriving from an address that is different than what is specified in the request. Support is implemented by manipulating the client's behavior, rewriting requests/responses to substitute IP addresses, and providing (essentially) a TURN server to relay media.

As far as I can remember, sipX's NAT support is recorded and implemented in the standard registration/redirect database. However, NAT support does depend on forcing the client to re-register frequently enough to be assured that the NAT mapping is not released. Since processing re-registrations is by far the bulk of the signaling traffic even without NAT support, this is not a trivial change.

My expectation is that almost all commercial SIP systems have NAT support of this sort.

One difference between this sort of NAT support and Outbound is that NAT support is done only at the registrar/proxy; if there is a separate edge proxy, it only passes UDP messages and can easily be stateless. This might be a significant factor in very large deployments.

Perhaps a significant problem with Outbound is that it has to be implemented in both the phone and the switch, leading to a network effect problem.

At this point, it seems to me that we need to get a better understanding of what people are doing in the market to deal with NATs and find out why they don't use Outbound. (Since Outbound is the standard method, I would think it has a strategic advantage in the technological competition.)

Roman notes:

There is a significant number of end points, such as Polycom phones, that do support SIP outbound.

The most efficient way for the client to keep the NAT hole open is to send STUN based keep alive messages. They are widely supported. For instance OpenSIPS supports via Stun Module.

For server, the most efficient way to keep the NAT hole open is to send OPTIONS or NOTIFY requests to the client. This is much more efficient then registrations with small timeouts.

Registration server which has an in-memory database for current registrations can be fairly efficient in reducing number of back-end DB updates due to registrations and subscriptions with small

timeouts. It is not going to be stateless, but its state does not need to be replicated and would automatically recover on the stand-by server on the next registration or subscription request from the client.

One of the big problems with encrypted SIP traffic is that it gets modified by ALG. For a lot of hosted PBX providers avoiding the need to troubleshoot customer routers is a bigger incentive to deploy TLS then user privacy.

4. Handoff between Interfaces

4.1. Requirements

- Deal with the handoff problem, i.e., when the call (signaling and media) are being sent over one interface, and that interface becomes unavailable, but another interface has become available, the signaling and media should be rerouted via the other interface.
- Similarly, if the external address of a NAT binding changes, the UA has, in effect, transitioned from using one interface to another.
- The primary requirement is that the signaling path is reestablished, i.e., the call is not dropped. It may take several seconds for signaling flow to be reestablished.
- The media flow should be interrupted for only a "short" period of time so the user does not assume that the call has been dropped. 2 seconds seems a reasonable target.

Generally, loss of connectivity can be detected by loss of incoming RTCP packets. It looks like the expected RTCP interval is 5 seconds or longer. Intermittent loss of RTP due to network congestion is likely, but we may have to consider detecting loss of RTP as an indicator of loss of connectivity. We have to consider both symmetric loss of connectivity, in which traffic in both directions is lost simultaneously, and asymmetric loss of connectivity, in which traffic in one direction is lost while traffic in the other continues.

Restoring RTP (media) connectivity is straightforward once SIP (signaling) connectivity is restored, by executing a re-INVITE to renegotiate RTP listening ports, etc.

4.2. Restoring Signaling Connectivity

I can see two ways to restore SIP connectivity: (1) sending re-INVITE to perform a target refresh, changing the UA's target URI, and (2) initiating a new dialog by sending an INVITE-with-Replaces to the remote target URI in order to replace the dialog with a new dialog.

In either case, the UA should not attempt to modify/replace the dialog before sending an OPTIONS request and receiving a response from the new interface to the URI that will be targeted by the new INVITE. (The round-trip OPTIONS ensures that there is two-way signaling connectivity to the targeted URI.) If the UA has more than one interface that is still working, it probably needs to probe the target URI using each interface (in parallel), because some URIs may not be reachable from some interfaces.

Sending a re-INVITE is a good method if the UA knows that the first URI in the route set can be reached from the UA's new address (interface). It seems to me that this will often not be the case, particularly when handing off between a carrier mobile network and a private WiFi network.

If the route set of the current dialog cannot be maintained, it is possible to create an entirely new dialog by directing an INVITE-with-Replaces to the remote target URI of the dialog. In a perfect world, the remote target URI is a GRUU, and the connectivity of a new INVITE to the GRUU is assured. Unfortunately there is no guarantee that will work, either.

The difficulty is that all the UA knows about the dialog is the route set, and there are no fixed conventions that allow the UA to extract from the route set a URI that can be targeted by an INVITE/Replaces. E.g., if the route set is:

A: UA's target
B: record route URI 1
C: record route URI 2
D: record route URI 3
E: remote target

It's possible that URI C is the only publicly routable URI, and the URI for the INVITE/Replaces should be E?Route=C&Route=D.

One possibility is probing each route URI with an OPTIONS request. That may not be a reliable test if the URI contains an IP address, especially if the address is in private-use space, as the UA may send the OPTIONS request to a different server that has the same address. Though probably if the URI contains a DNS name, then if the OPTIONS

succeeded, it probably reached the same server as the route URI indicates.

Absent any system for indicating which URIs are publicly routable (other than the "gr" parameter for GRUUs), we probably have to rely on the fact that most SIP telephones execute transfers using INVITE/Replaces requests that assume that the remote target URI that they see is publicly routable. As a consequence of this, SIP switches perform machinations to ensure that the remote target URIs seen by phones are publicly routable.

Assuming we can assume that remote target URIs are publicly routable, then we can safely recommend that UAs always use INVITE/Replaces to restore signaling.

4.3. Maintaining the CLIENT'S GRUU

Since we expect a UA to use a GRUU as its target URI so that remote UAs can target the GRUU to reestablish signaling, a UA must ensure that its GRUU routes to all the addresses by which it is reachable. Generally, this means that the UA must update its registration promptly whenever an interface becomes usable.

However, it looks like there may be some ugly consequences of maintaining multiple mappings for a UA's GRUU -- how does a request get routed to the GRUU, serially or parallely? Can one use "Request-Disposition: parallel" to force an OPTIONS request to fork parallely to all of the contacts of a GRUU? The executing UA does not need to know which of the contacts of the remote UA were accessible via the GRUU, but it does need to know quite promptly that some contact of the remote UA is accessible via the GRUU.

OTOH, when the INVITE/Replaces is processed, we don't want it to be delayed due to serial forking to contacts that are no longer accessible, because the timeouts prescribed in SIP are long relative to the time we want handouts to occur in. But perhaps "Request-Disposition: parallel" can be used here, as the first fork of an INVITE/Replaces to reach a UA will be acted upon and generate a 200 response, and any later arrivals from other forks will receive 481 responses.

4.4. Glare

One risk of reestablishing the dialog is that both UAs might attempt to reestablish the dialog at the same time. If both UAs attempt to re-INVITE at the same time, and the invites cross in transit, the "glare" rules will require each UA to reject the other UA's re-INVITE, back off, and resend, as described in RFC 3261 section 14.

If one or both UA uses INVITE/Replaces, various conflicts can occur. It seems to me that the correct way to fix this is to treat the state of "an INVITE/Replaces to revive the dialog is outstanding" as a glare-creating condition that is handled the same way as "a re-INVITE is outstanding".

4.5. Charging Information

Ideally, if a handoff does not take the call outside the domain of a single carrier, the carrier should be given enough information to determine that the new dialog is a logical continuation of the old dialog, so that it can combine the charging records of the two dialogs. In may cases, the carrier can probably determine from the INVITE/Replaces that the new dialog is related to the old dialog. But should there be a rule that requires that the new dialog copy some charging-related information from the old dialog?

5. GRUUs

An essential characteristic of a GRUU is that it’s globally accessible. But if the device only implements one address family, or the intervening network carries only one protocol, then a URI isn’t accessible to a device that only implements the *other* protocol.

It seems that the theoretical answer is to require a GRUU to be accessible in practice from the global Internet via either address family, but it seems like that would de-GRUU-ize probably most of the GRUUs that are being used in the universe.

This is particularly troublesome if we use GRUUs to solve, e.g., the handoff problem, since a handoff may involve a change of protocol.

Olle notes that a GRUU (almost always) has the same hostpart as the AOR. So if a client can reach the AOR to establish a dialog, it can reach the GRUU to manipulate the dialog.

6. Security Considerations

There probably aren’t any security issues. Copy the security considerations section from draft-ietf-sipcore-dns-dual-stack.

7. IANA Considerations

This document does not require any actions by IANA.

8. Acknowledgments

So far: Brett, Roman, Olle

9. References

9.1. Normative References

[RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, DOI 10.17487/RFC3263, June 2002, <http://www.rfc-editor.org/info/rfc3263>.

9.2. Informative References

[RFC5626] Jennings, C., Ed., Mahy, R., Ed., and F. Audet, Ed., "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, DOI 10.17487/RFC5626, October 2009, <http://www.rfc-editor.org/info/rfc5626>.

Appendix A. Revision History

[Note to RFC Editor: Please remove this entire section upon publication as an RFC.]

A.1. Changes from draft-worley-sipcore-dual-stack-00 to draft-worley-sipcore-dual-stack-01

Author’s Address

Dale R. Worley
Ariadne Internet Services
738 Main St.
Waltham, MA 02451
US

Email: worley@ariadne.com