                  Indication of Support for Keep-Alive

Abstract

   This specification defines a new Session Initiation Protocol (SIP)
   Via header field parameter, "keep", which allows adjacent SIP
   entities to explicitly negotiate usage of the Network Address
   Translation (NAT) keep-alive mechanisms defined in SIP Outbound, in
   cases where SIP Outbound is not supported, cannot be applied, or
   where usage of keep-alives is not implicitly negotiated as part of
   the SIP Outbound negotiation.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6223.

Table of Contents

1. Introduction

   Section 3.5 of SIP Outbound [RFC5626] defines two keep-alive
   mechanisms. Even though the keep-alive mechanisms are separated from
   the rest of the SIP Outbound mechanism, SIP Outbound does not define
   a mechanism to explicitly negotiate usage of the keep-alive
   mechanisms. In some cases, usage of keep-alives can be implicitly
   negotiated as part of the SIP Outbound negotiation.

   However, there are SIP Outbound use-cases where usage of keep-alives
   is not implicitly negotiated as part of the SIP Outbound negotiation.
   In addition, there are cases where SIP Outbound is not supported, or
   where it cannot be applied, but where there is still a need to be

able to negotiate usage of keep-alives. Last, SIP Outbound only
allows keep-alives to be negotiated between a User Agent (UA) and an
edge proxy, and not between other SIP entities.

This specification defines a new Session Initiation Protocol (SIP)
[RFC3261] Via header field parameter, "keep", which allows adjacent
SIP entities to explicitly negotiate usage of the NAT keep-alive
mechanisms defined in SIP Outbound. The "keep" parameter allows SIP
entities to indicate willingness to send keep-alives, to indicate
willingness to receive keep-alives, and -- for SIP entities willing
to receive keep-alives -- to provide a recommended keep-alive
frequency.

The following sections describe use-cases where a mechanism to
explicitly negotiate usage of keep-alives is needed.

1.1. Use-Case: Dialog from Non-Registered UAs

In some cases, a User Agent Client (UAC) does not register itself
before it establishes a dialog, but in order to maintain NAT bindings
open during the lifetime of the dialog, it still needs to be able to
negotiate the sending of keep-alives towards its adjacent downstream
SIP entity. A typical example is an emergency call, where a
registration is not always required in order to make the call.

1.2. Use-Case: SIP Outbound Not Supported

In some cases, some SIP entities that need to be able to negotiate
the use of keep-alives might not support SIP Outbound. However, they
might still support the keep-alive mechanisms defined in SIP Outbound
and need to be able to negotiate usage of them.

1.3. Use-Case: SIP Dialog Initiated Outbound Flows

SIP Outbound allows the establishment of flows using the initial
request for a dialog. As specified in RFC 5626 [RFC5626], usage of
keep-alives is not implicitly negotiated for such flows.

2. Conventions

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 BCP 14, RFC 2119
[RFC2119].

3. Definitions

   Edge proxy: As defined in RFC 5626, a SIP proxy that is located
   topologically between the registering User Agent (UA) and the
   Authoritative Proxy.

      NOTE: In some deployments, the edge proxy might be physically
      located in the same SIP entity as the Authoritative Proxy.

   Keep-alives: The keep-alive messages defined in RFC 5626.

   "keep" parameter: A SIP Via header field parameter that a SIP entity
   can insert in the topmost Via header field that it adds to the
   request, to explicitly indicate willingness to send keep-alives
   towards its adjacent downstream SIP entity. A SIP entity can add a
   parameter value to the "keep" parameter in a response to explicitly
   indicate willingness to receive keep-alives from its adjacent
   upstream SIP entity.

   SIP entity: SIP User Agent (UA), or proxy, as defined in RFC 3261.

   Adjacent downstream SIP entity: The adjacent SIP entity in the
   direction towards which a SIP request is sent.

   Adjacent upstream SIP entity: The adjacent SIP entity in the
   direction from which a SIP request is received.

4. User Agent and Proxy Behavior

4.1. General

   This section describes how SIP UAs and proxies negotiate usage of
   keep-alives associated with a registration or a dialog, which types
   of SIP requests can be used in order to negotiate the usage, and the
   lifetime of the negotiated keep-alives.

   SIP entities indicate willingness to send keep-alives towards the
   adjacent downstream SIP entity using SIP requests. The associated
   responses are used by SIP entities to indicate willingness to receive
   keep-alives. SIP entities that indicate willingness to receive keep-
   alives can provide a recommended keep-alive frequency.

   The procedures to negotiate usage of keep-alives are identical for
   SIP UAs and proxies.

   In general, it can be useful for SIP entities to indicate willingness
   to send keep-alives, even if they are not aware of any necessity for
   them to send keep-alives, since the adjacent downstream SIP entity

might have knowledge about the necessity. Similarly, if the adjacent
upstream SIP entity has indicated willingness to send keep-alives, it
can be useful for SIP entities to indicate willingness to receive
keep-alives, even if they are not aware of any necessity for the
adjacent upstream SIP entity to send them.

   NOTE: Usage of keep-alives is negotiated per direction. If a SIP
   entity has indicated willingness to receive keep-alives from an
   adjacent SIP entity, the sending of keep-alives towards that
   adjacent SIP entity needs to be separately negotiated.

   NOTE: Since there are SIP entities that already use a combination
   of Carriage Return and Line Feed (CRLF) as keep-alive messages,
   and SIP entities are expected to be able to receive those, this
   specification does not forbid the sending of double-CRLF keep-
   alive messages towards an adjacent SIP entity even if usage of
   keep-alives with that SIP entity has not been negotiated.
   However, the "keep" parameter is still important in order for a
   SIP entity to indicate that it supports the sending of double-CRLF
   keep-alive messages, so that the adjacent downstream SIP entity
   does not use other mechanisms (e.g., short registration refresh
   intervals) in order to keep NAT bindings open.

4.2. Lifetime of Keep-Alives

4.2.1. General

   The lifetime of negotiated keep-alives depends on whether the keep-
   alives are associated with a registration or a dialog. This section
   describes the lifetime of negotiated keep-alives.

4.2.2. Keep-Alives Associated with Registration

   SIP entities use a registration request in order to negotiate usage
   of keep-alives associated with a registration. Usage of keep-alives
   can be negotiated when the registration is established, or later
   during the registration. Once negotiated, keep-alives are sent until
   the registration is terminated, or until a subsequent registration
   refresh request is sent or forwarded. When a subsequent registration
   refresh request is sent or forwarded, if a SIP entity is willing to
   continue sending keep-alives associated with the registration, usage
   of keep-alives MUST be re-negotiated. If usage is not successfully
   re-negotiated, the SIP entity MUST cease the sending of keep-alives
   associated with the registration.

   NOTE: The sending of keep-alives associated with a registration
   can only be negotiated in the direction from the registering SIP
   entity towards the registrar.

4.2.3. Keep-Alives Associated with Dialog

   SIP entities use an initial request for a dialog, or a mid-dialog
   target refresh request [RFC3261], in order to negotiate the sending
   and receiving of keep-alives associated with a dialog. Usage of
   keep-alives can be negotiated when the dialog is established, or
   later during the lifetime of the dialog. Once negotiated, keep-
   alives MUST be sent for the lifetime of the dialog, until the dialog
   is terminated. Once the usage of keep-alives associated with a
   dialog has been negotiated, it is not possible to re-negotiate the
   usage associated with the dialog.

4.3. Behavior of a SIP Entity Willing to Send Keep-Alives

   As defined in RFC 5626, a SIP entity that supports the sending of
   keep-alives must act as a Session Traversal Utilities for NAT (STUN)
   client [RFC5389]. The SIP entity must support those aspects of STUN
   that are required in order to apply the STUN keep-alive mechanism
   defined in RFC 5626, and it must support the CRLF keep-alive
   mechanism defined in RFC 5626. RFC 5626 defines when to use STUN and
   when to use double-CRLF for keep-alives.

   When a SIP entity sends or forwards a request, if it wants to
   negotiate the sending of keep-alives associated with a registration
   or a dialog, it MUST insert a "keep" parameter in the topmost Via
   header field that it adds to the request, to indicate willingness to
   send keep-alives.

   When the SIP entity receives the associated response, if the "keep"
   parameter in the topmost Via header field of the response contains a
   "keep" parameter value, it MUST start sending keep-alives towards the
   same destination where it would send a subsequent request (e.g.,
   REGISTER requests and initial requests for dialog) associated with
   the registration (if the keep-alive negotiation is for a
   registration), or where it would send subsequent mid-dialog requests
   (if the keep-alive negotiation is for a dialog). Subsequent
   mid-dialog requests are addressed based on the dialog route set.

   Once a SIP entity has negotiated the sending of keep-alives
   associated with a dialog towards an adjacent SIP entity, it MUST NOT
   insert a "keep" parameter in any subsequent SIP requests associated
   with that dialog towards that adjacent SIP entity. Such "keep"
   parameters MUST be ignored, if received.

   Since an ACK request does not have an associated response, it cannot
   be used to negotiate usage of keep-alives. Therefore, a SIP entity
   MUST NOT insert a "keep" parameter in the topmost Via header field of
   an ACK request. Such "keep" parameters MUST be ignored, if received.

   A SIP entity MUST NOT indicate willingness to send keep-alives
   associated with a dialog, unless it has also inserted itself in the
   dialog route set [RFC3261].

      NOTE: When a SIP entity sends an initial request for a dialog, if
      the adjacent downstream SIP entity does not insert itself in the
      dialog route set using a Record-Route header field [RFC3261], the
      adjacent downstream SIP entity will change once the dialog route
      set has been established. If a SIP entity inserts a "keep"
      parameter in the topmost Via header field of an initial request
      for a dialog, and the "keep" parameter in the associated response
      does not contain a parameter value, the SIP entity might choose to
      insert a "keep" parameter in the topmost Via header field of a
      subsequent SIP request associated with the dialog, in case the new
      adjacent downstream SIP entity (based on the dialog route set) is
      willing to receive keep-alives (in which case it will add a
      parameter value to the "keep" parameter).

   If an INVITE request is used to indicate willingness to send keep-
   alives, as long as at least one response (provisional or final) to
   the INVITE request contains a "keep" parameter with a parameter
   value, it is seen as an indication that the adjacent downstream SIP
   entity is willing to receive keep-alives associated with the dialog
   on which the response is received.

4.4. Behavior of a SIP Entity Willing to Receive Keep-Alives

   As defined in RFC 5626, a SIP entity that supports the receiving of
   keep-alives must act as a STUN server [RFC5389]. The SIP entity must
   support those aspects of STUN that are required in order to apply the
   STUN keep-alive mechanism defined in RFC 5626, and it must support
   the CRLF keep-alive mechanism defined in RFC 5626.

   When a SIP entity sends or forwards a response, and the adjacent
   upstream SIP entity has indicated willingness to send keep-alives, if
   the SIP entity is willing to receive keep-alives associated with the
   registration or with the dialog from that adjacent upstream SIP
   entity, then it MUST add a parameter value to the "keep" parameter
   before sending or forwarding the response. The parameter value, if
   present and with a value other than zero, represents a recommended
   keep-alive frequency, given in seconds.

   There might be multiple responses to an INVITE request. When a SIP
   entity indicates willingness to receive keep-alives in a response to
   an INVITE request, it MUST add a parameter value to the "keep"
   parameter in at least one reliable response to the request. The SIP
   entity MAY add identical parameter values to the "keep" parameters in
   other responses to the same request. The SIP entity MUST NOT add

different parameter values to the "keep" parameters in responses to
the same request. The SIP entity SHOULD indicate the willingness to
receive keep-alives as soon as possible.

A SIP entity MUST NOT indicate willingness to receive keep-alives
associated with a dialog, unless it has also inserted itself in the
dialog route set [RFC3261].

5. Keep-Alive Frequency

If a SIP entity receives a SIP response, where the topmost Via header
field contains a "keep" parameter with a non-zero value that
indicates a recommended keep-alive frequency, given in seconds, it
MUST use the procedures defined for the Flow-Timer header field
[RFC5626]. According to the procedures, the SIP entity must send
keep-alives at least as often as the indicated recommended keep-alive
frequency, and if the SIP entity uses the recommended keep-alive
frequency, then it should send its keep-alives so that the interval
between each keep-alive is randomly distributed between 80% and 100%
of the recommended keep-alive frequency.

If the received "keep" parameter value is zero, the SIP entity can
send keep-alives at its discretion. RFC 5626 provides additional
guidance on selecting the keep-alive frequency in case a recommended
keep-alive frequency is not provided.

This specification does not specify actions to take if negotiated
keep-alives are not received. As defined in RFC 5626, the receiving
SIP entity may consider a connection to be dead in such situations.

If a SIP entity that adds a parameter value to the "keep" parameter
in order to indicate willingness to receive keep-alives also inserts
a Flow-Timer header field (that can happen if the SIP entity is using
both the Outbound mechanism and the keep-alive mechanism) in the same
SIP message, the header field value and the "keep" parameter value
MUST be identical.

SIP Outbound uses the Flow-Timer header field to indicate the server-
recommended keep-alive frequency; however, it will only be sent
between a UA and an edge proxy. On the other hand, by using the
"keep" parameter, the sending and receiving of keep-alives can be
negotiated between multiple entities on the signalling path. In
addition, since the server-recommended keep-alive frequency might
vary between different SIP entities, a single Flow-Timer header field
cannot be used to indicate all the different frequency values.

6. Connection Reuse

   Keep-alives are often sent in order to keep NAT bindings open, so
   that SIP requests sent in the reverse direction will pass by the NAT
   and reuse the same connection. In the case of non-connection-
   oriented transport protocols, keep-alives would permit the same path
   to be reused. This specification does not define such a connection
   reuse mechanism. The keep-alive mechanism defined in this
   specification is only used to negotiate the sending and receiving of
   keep-alives. Entities that want to reuse connections need to use
   another mechanism to ensure that security aspects associated with
   connection reuse are taken into consideration.

   RFC 5923 [RFC5923] specifies a mechanism for using connection-
   oriented transports to send requests in the reverse direction, and an
   entity that wants to use connection reuse as well as indicate support
   of keep-alives on that connection will insert both the "alias"
   parameter defined in RFC 5923 and the "keep" parameter defined in
   this specification.

   SIP Outbound specifies how registration flows are used to send
   requests in the reverse direction.

7. Examples

7.1. General

   This section shows example flows where usage of keep-alives,
   associated with a registration and a dialog, is negotiated between
   different SIP entities.

      NOTE: The examples do not show the actual syntactical encoding of
      the request lines, response lines, and the Via header fields, but
      rather a pseudocode in order to identify the message type and also
      identify to which SIP entity a Via header field is associated.

7.2. Keep-Alive Negotiation Associated with Registration: UA-Proxy

   Figure 1 shows an example where Alice sends a REGISTER request. She
   indicates willingness to send keep-alives by inserting a "keep"
   parameter in the Via header field of her request. The edge proxy
   (P1) forwards the request towards the registrar.

   P1 is willing to receive keep-alives from Alice for the duration of
   the registration, so when P1 receives the associated response it adds
   a "keep" parameter value, which indicates a recommended keep-alive
   frequency of 30 seconds, to Alice's Via header field, before it
   forwards the response towards Alice.

When Alice receives the response, she determines from her Via header
field that P1 is willing to receive keep-alives associated with the
registration. Until either the registration expires or Alice sends a
registration refresh request, Alice then sends periodic keep-alives
(in this example using the STUN keep-alive technique) towards P1,
using the recommended keep-alive frequency indicated by the "keep"
parameter value.

```
   Alice                           P1                       REGISTRAR
     |                             |                             |
     |--- REGISTER------------->|                             |
     |     Via: Alice;keep         |                             |
     |                             |--- REGISTER-------------->|
     |                             |     Via: P1                 |
     |                             |     Via: Alice;keep         |
     |                             |                             |
     |                             |<-- 200 OK ----------------|
     |                             |     Via: P1                 |
     |                             |     Via: Alice;keep         |
     |<-- 200 OK --------------|                             |
     |     Via: Alice;keep=30      |                             |
     |                             |                             |
     |                             |                             |
     |                *** Timeout ***                            |
     |                             |                             |
     |=== STUN request ======>|                             |
     |<== STUN response ======|                             |
     |                             |                             |
     |                *** Timeout ***                            |
     |                             |                             |
     |=== STUN request ======>|                             |
     |<== STUN response ======|                             |
     |                             |                             |
```
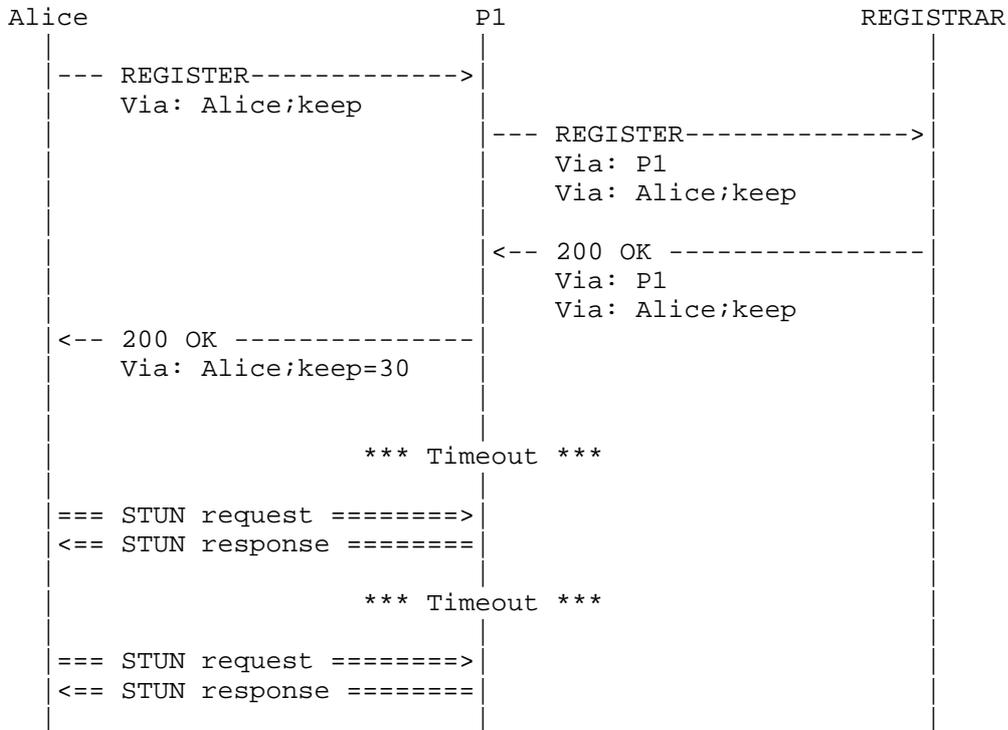
Figure 1: Example Call Flow

7.3. Keep-Alive Negotiation Associated with Dialog: UA-Proxy

   Figure 2 shows an example where Alice sends an initial INVITE request
   for a dialog. She indicates willingness to send keep-alives by
   inserting a "keep" parameter in the Via header field of her request.
   The edge proxy (P1) adds itself to the dialog route set by adding
   itself to a Record-Route header field, before it forwards the request
   towards Bob.

   P1 is willing to receive keep-alives from Alice for the duration of
   the dialog, so when P1 receives the associated response it adds a
   "keep" parameter value, which indicates a recommended keep-alive
   frequency of 30 seconds, to Alice's Via header field, before it
   forwards the response towards Alice.

   When Alice receives the response, she determines from her Via header
   field that P1 is willing to receive keep-alives associated with the
   dialog. For the lifetime of the dialog, Alice then sends periodic
   keep-alives (in this example using the STUN keep-alive technique)
   towards P1, using the recommended keep-alive frequency indicated by
   the "keep" parameter value.

```
       Alice                       P1                        Bob
        |                          |                          |
        |--- INVITE -------------->|                          |
        |     Via: Alice;keep      |                          |
        |                          |--- INVITE -------------->|
        |                          |     Via: P1              |
        |                          |     Via: Alice;keep      |
        |                          |     Record-Route: P1     |
        |                          |                          |
        |                          |<-- 200 OK ---------------|
        |                          |     Via: P1              |
        |                          |     Via: Alice;keep      |
        |                          |     Record-Route: P1     |
        |<-- 200 OK ---------------|                          |
        |     Via: Alice;keep=30   |                          |
        |     Record-Route: P1     |                          |
        |                          |                          |
        |--- ACK ----------------->|                          |
        |                          |                          |
        |                          |--- ACK ----------------->|
        |                          |                          |
        |             *** Timeout ***                         |
        |                          |                          |
        |=== STUN request ========>|                          |
        |<== STUN response ========|                          |
        |                          |                          |
        |             *** Timeout ***                         |
        |                          |                          |
        |=== STUN request ========>|                          |
        |<== STUN response ========|                          |
        |                          |                          |
        |                          |                          |
        |--- BYE ----------------->|                          |
        |                          |                          |
        |                          |--- BYE ------------------>|
        |                          |                          |
        |                          |<-- 200 OK ---------------|
        |                          |                          |
```
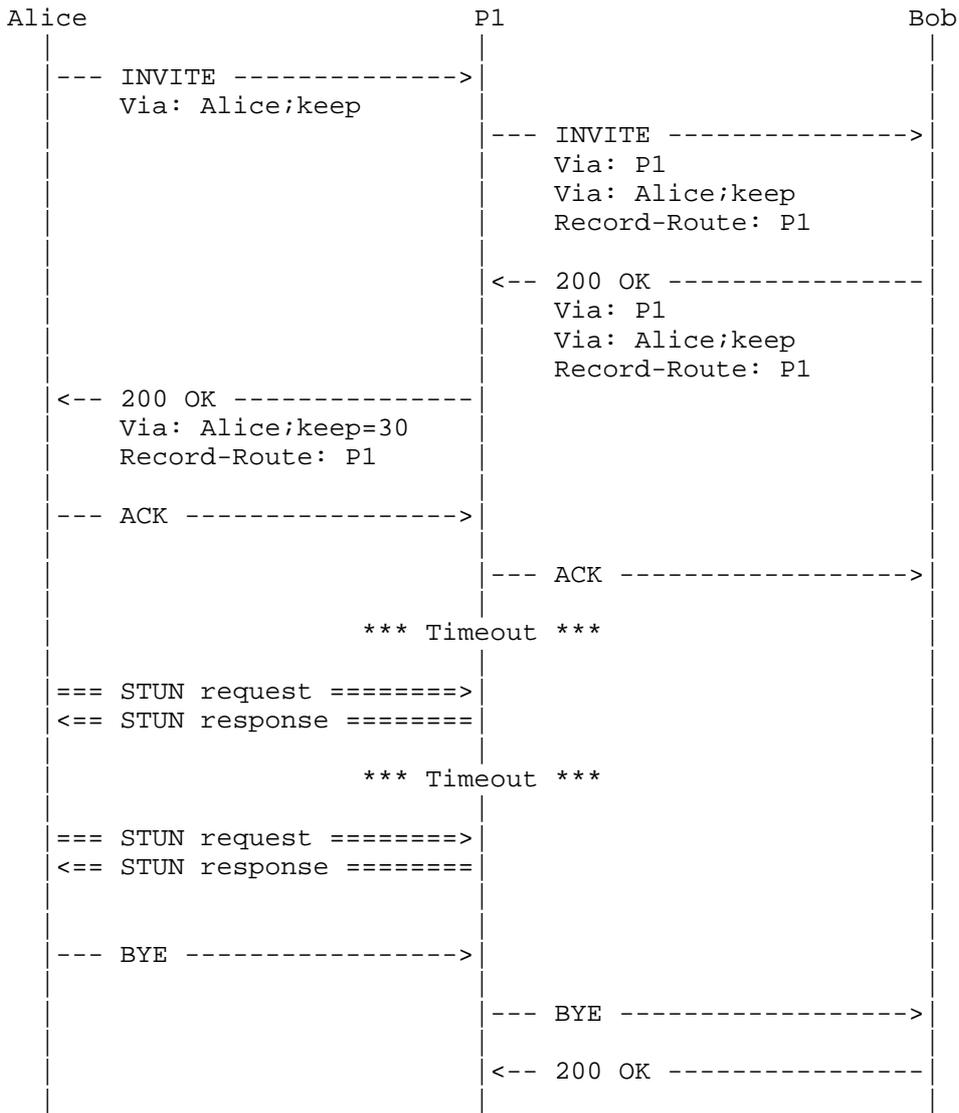
Figure 2: Example Call Flow

7.4. Keep-Alive Negotiation Associated with Dialog: UA-UA

   Figure 3 shows an example where Alice sends an initial INVITE request
   for a dialog. She indicates willingness to send keep-alives by
   inserting a "keep" parameter in the Via header field of her request.
   In this scenario, the edge proxy (P1) does not add itself to a
   Record-Route header field (and so will not be added to the dialog
   route set) before forwarding the request towards Bob.

   When Alice receives the response, she determines from the Via header
   field that P1 is not willing to receive keep-alives associated with
   the dialog from her. When the dialog route set has been established,
   Alice sends a mid-dialog UPDATE request towards Bob (since P1 did not
   insert itself in the dialog route set), and she once again indicates
   willingness to send keep-alives by inserting a "keep" parameter in
   the Via header field of her request. Bob supports the keep-alive
   mechanism, and is willing to receive keep-alives associated with the
   dialog from Alice, so he creates a response and adds a "keep"
   parameter value, which indicates a recommended keep-alive frequency
   of 30 seconds, to Alice's Via header field, before he forwards the
   response towards Alice.

   When Alice receives the response, she determines from her Via header
   field that Bob is willing to receive keep-alives associated with the
   dialog. For the lifetime of the dialog, Alice then sends periodic
   keep-alives (in this example using the STUN keep-alive technique)
   towards Bob, using the recommended keep-alive frequency indicated by
   the "keep" parameter value.

```
         Alice                          P1                          Bob
           |                            |                            |
           |--- INVITE -------------->|                            |
           |     Via: Alice;keep        |                            |
           |                            |--- INVITE --------------->|
           |                            |     Via: P1                |
           |                            |     Via: Alice;keep        |
           |                            |                            |
           |                            |<-- 200 OK ----------------|
           |                            |     Via: P1                |
           |                            |     Via: Alice;keep        |
           |<-- 200 OK --------------|                            |
           |     Via: Alice;keep        |                            |
           |                            |                            |
           |                            |                            |
           |--- ACK --------------------------------------------->|
           |                            |                            |
           |--- UPDATE ------------------------------------------->|
           |     Via: Alice;keep        |                            |
           |                            |                            |
           |<-- 200 OK --------------------------------------------|
           |     Via: Alice;keep=30     |                            |
           |                            |                            |
           |                            |                            |
           |                   *** Timeout ***                       |
           |                            |                            |
           |=== STUN request ====================================>|
           |<== STUN response ===================================|
           |                            |                            |
           |                   *** Timeout ***                       |
           |                            |                            |
           |=== STUN request ====================================>|
           |<== STUN response ===================================|
           |                            |                            |
           |                            |                            |
           |--- BYE --------------------------------------------->|
           |                            |                            |
           |<-- 200 OK --------------------------------------------|
           |                            |                            |
```
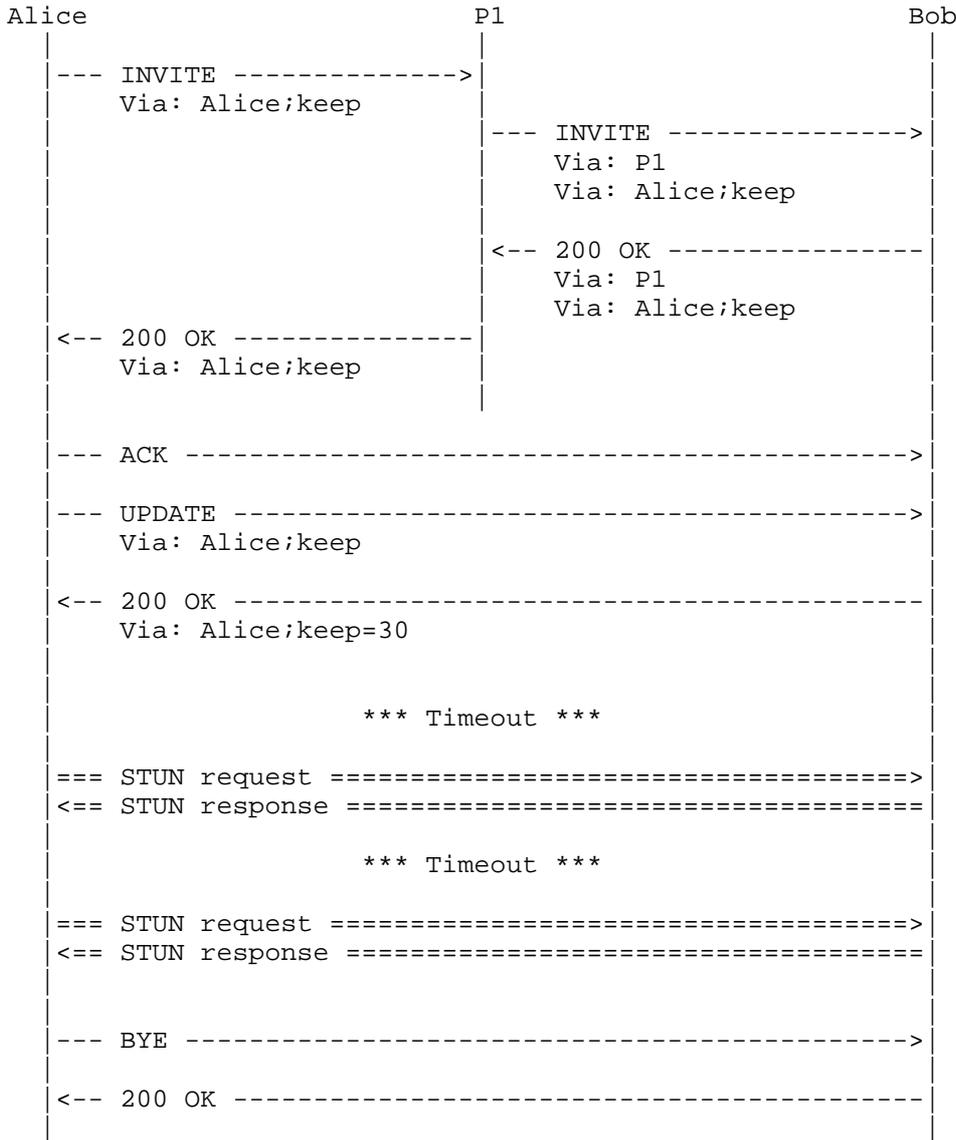
Figure 3: Example Call Flow

8. Grammar

8.1. General

   This section extends the ABNF definition of via-params from [RFC3261]
   by adding a new Via header field parameter, "keep". The ABNF defined
   in this specification is conformant to RFC 5234 [RFC5234]. "EQUAL"
   is defined in RFC 3261. "DIGIT" is defined in RFC 5234.

8.2. ABNF

   via-params =/ keep

   keep   = "keep" [EQUAL 1*(DIGIT)]

9. IANA Considerations

9.1. "keep" Via Header Field Parameter

   This specification defines a new Via header field parameter called
   "keep" in the "Header Field Parameters and Parameter Values"
   sub-registry as per the registry created by [RFC3968]. The syntax is
   defined in Section 8 of this document. IANA has registered the
   following:

Header Field	Parameter Name	Predefined Values	Reference
Via	keep	No	[RFC6223]

10. Security Considerations

   SIP entities that send or receive keep-alives are often required to
   use a connection reuse mechanism, in order to ensure that requests
   sent in the reverse direction, towards the sender of the keep-alives,
   traverse NATs, etc. This specification does not define a connection
   reuse mechanism, and it does not address security issues related to
   connection reuse. SIP entities that wish to reuse connections need
   to use a dedicated connection reuse mechanism, in conjunction with
   the keep-alive negotiation mechanism.

   Unless SIP messages are integrity protected hop-by-hop, e.g., using
   Transport Layer Security (TLS) [RFC5246] or Datagram Transport Layer
   Security (DTLS) [RFC4347], a man-in-the-middle can modify Via header
   fields used by two entities to negotiate the sending of keep-alives,
   e.g., by removing the designations used to indicate willingness to
   send and receive keep-alives, or by decreasing the timer value to a
   very low value, which might trigger additional resource consumption
   due to the frequently sent keep-alives.

The behaviors defined in Sections 4.3 and 4.4 require a SIP entity
using the mechanism defined in this specification to place a value in
the "keep" parameter in the topmost Via header field value of a
response the SIP entity sends. They do not instruct the entity to
place a value in a "keep" parameter of any request it forwards. In
particular, a SIP proxy MUST NOT place a value into the "keep"
parameter of the topmost Via header field value of a request it
receives before forwarding it. A SIP proxy implementing this
specification SHOULD remove any "keep" parameter values in any Via
header field values below the topmost one in responses it receives
before forwarding them.

When requests are forwarded across multiple hops, it is possible for
a malicious downstream SIP entity to tamper with the accrued values
in the Via header field. The malicious SIP entity could place a
value, or change an existing value in a "keep" parameter in any of
the Via header field values -- not just the topmost value. A proxy
implementation that simply forwards responses by stripping the
topmost Via header field value and not inspecting the resulting new
topmost Via header field value risks being adversely affected by such
a malicious downstream SIP entity. In particular, such a proxy may
start receiving STUN requests if it blindly forwards a response with
a "keep" parameter with a value it did not create in the topmost Via
header field.

To lower the chances of the malicious SIP entity's actions having
adverse effects on such proxies, when a SIP entity sends STUN keep-
alives to an adjacent downstream SIP entity and does not receive a
response to those STUN messages (as described in Section 7.2.1 of
RFC 5389 [RFC5389], it MUST stop sending keep-alives for the
remaining duration of the dialog (if the sending of keep-alives were
negotiated for a dialog) or until the sending of keep-alives is
re-negotiated for the registration (if the sending keep-alives were
negotiated for a registration).

Apart from the issues described above, this specification does not
introduce security considerations in addition to those specified for
keep-alives in [RFC5626].

11. Acknowledgements

Thanks to Staffan Blau, Francois Audet, Hadriel Kaplan, Sean
Schneyer, and Milo Orsic for their comments on the initial draft
version of this document. Thanks to Juha Heinanen, Jiri Kuthan, Dean
Willis, John Elwell, Paul Kyzivat, Peter Musgrave, Dale Worley, Adam
Roach, and Robert Sparks for their comments on the sipcore mailing
list. Thanks to Vijay Gurbani for providing text about the
relationship with the connect reuse specification.

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.

   [RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for
              Syntax Specifications: ABNF", STD 68, RFC 5234,
              January 2008.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              October 2008.

   [RFC5626] Jennings, C., Ed., Mahy, R., Ed., and F. Audet, Ed.,
              "Managing Client-Initiated Connections in the Session
              Initiation Protocol (SIP)", RFC 5626, October 2009.

12.2. Informative References

   [RFC3968] Camarillo, G., "The Internet Assigned Number Authority
              (IANA) Header Field Parameter Registry for the Session
              Initiation Protocol (SIP)", BCP 98, RFC 3968,
              December 2004.

   [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security", RFC 4347, April 2006.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246, August 2008.

   [RFC5923] Gurbani, V., Ed., Mahy, R., and B. Tate, "Connection Reuse
              in the Session Initiation Protocol (SIP)", RFC 5923,
              June 2010.

Author's Address

     Christer Holmberg
     Ericsson
     Hirsalantie 11
     Jorvas   02420
     Finland

     EMail: christer.holmberg@ericsson.com