SIP Providers' Awareness of Media Connectivity

Stefan Gasterstädt, Markus Gusowski, Bettina Schnor
Institute of Computer Science
University of Potsdam
Potsdam, Germany
{gasterstaedt,gusowski,schnor}@cs.uni-potsdam.de

***Abstract*—Voice-over-IP (VoIP) has become an important service in the Internet. In contrast to the Public Switched Telephone Network where the delivery of all messages and streams is the responsibility of the calling parties' providers, VoIP media data is sent directly between the user agents without provider interaction in most cases. Hence, a VoIP provider is not aware of media connectivity, i. e., whether a call was successful or not. This may lead to incorrect behavior when a VoIP provider offers services beyond signaling (for example, SPIT prevention, payment). In this paper, we discuss several approaches for the detection of media connectivity and present a solution that conforms with the existing standards. The modified behavior of the user agents, the use of SCTP and provider's awareness of media connectivity are described in detail. Finally, measurements show that our solution results in neglectable overhead.**

***Keywords*-Voice-over-IP (VoIP), media connectivity, SCTP**

I. Introduction

The Session Initiation Protocol (SIP) [22] has become a majorly used protocol in Voice-over-IP (VoIP) communication. It utilizes the Uniform Resource Identifier (URI) schema to address users, single devices or end points and resolves these URIs to Internet Protocol (IP) addresses by using SIP proxy servers and Domain Name Service (DNS) lookups. Users can call others without knowing their current IP address, because session invitations are routed to the SIP proxy that is responsible for the callee's URI domain; and as a next step, this proxy uses its location service to locate the callee[1] and forwards the *INVITE* request to the addressed user (cf. Fig. 1). Depending on its configuration, a SIP proxy may or may not request to stay in the route of any further SIP signaling. Normally, the media transmission is done directly between the user agents (UAs) via RTP.

It is a known problem that the basic SIP infrastructure does not conform to the Network Address Translator (NAT)-friendly application design guidelines described in RFC 3235 [23], and thus, NATs and firewalls cause serious problems for SIP message delivery and media connectivity in conjunction with the separation of signaling and media delivery, dynamic port allocation, or RTP's "$x + 1$" port schema. In contrast to the UA-to-UA media connection, there are solutions for SIP messages; for example, by simply traversing NAT using symmetric response routing [21]. Examples of NAT and firewall traversal for SIP are given in [19].

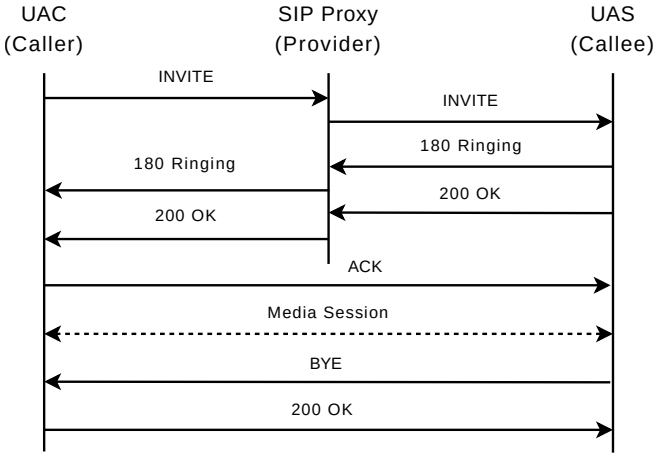

Figure 1: SIP Dialog of a Call

The explicit separation between the session signaling and media delivery comes along with a significant implication: VoIP providers offering SIP services are unaware of whether or not the media stream is actually received by the endpoint(s), i. e., whether there is *connectivity* or not. SIP does not check for connectivity, and the condition is not signaled in any way. Therefore, a SIP provider cannot know if two users will actually be able to communicate, even if a SIP session was successfully established. There are several reasons why media streams negotiated between the UAs may be blocked in one or both directions, mainly because of NATs and/or firewalls [10], [24], but other network problems like the lack of a network route, node crash, configuration problems, or codec mismatch could be responsible as well [1]. This is in contrast to the traditional Public Switched Telephone Network (PSTN), where there is always connectivity once signaling completes successfully.[2]

There are, however, important scenarios where it is desir-

[1]The location bindings can be updated by each respective user sending a *REGISTER* request to its SIP provider's registrar.

[2]Admittedly, there are some rare cases where people cannot talk to each other allthough there has been a successful ringing and call acceptance before. However, the PSTN phone provider will be aware of this failure.

 ISBN:978-1-61208-113-7

able for the provider to know the media connectivity status between the endpoints.

Payment: In some cases, the callee or the caller request some fee in order to accept or initiate a call. Examples include duration-based fees (similar to the PSTN); (fixed) fees relating to the (voice based) service a callee is offering, such as a support hotline; fees for calls a callee subscribed for, such as severe thunderstorm warning; or, in the case of Spam over Internet Telephony (SPIT) prevention, where a caller may be confronted with a small fee if its sincerity is in doubt [11], [12].

For whatever reason a session involves payment by at least one party, it is desirable to delay finalizing the payment transaction until connectivity is assured.

Reputation: Some approaches to detect and prevent SPIT use a reputation score in order to help determine the caller's nature [3], [12], [17]. Each user's reputation is related to its behavior and is calculated from several metrics that are collected by the providers. For examples, a short call duration may indicate an unsolicited call that prompted that callee to hang up immediately. Unfortunately, it may also indicate that at least one participant could not hear the other due to a lack of (bidirectional) media connectivity. In this case, the caller's reputation would falsely be reduced.

Forensics: In the area of law enforcement, reliable evidence is crucial. Regarding the question of whether or not a call took place, SIP can only provide information about signaling – if the phone rang, if the phone was picked up, and if the phone was hung up. This may not be sufficient: The information may be required as to whether or not the two parties in a call were actually able to communicate.

Call Detail Record Analysis: Call Detail Records (CDRs) are collected and analyzed for several reasons. These records contain information about each call, for example, the caller's and callee's IDs, the invitation time, the duration, and how the call terminated. This data can be used to conduct statistical analysis, to profile users' behavior, to reduce traffic congestion or, in general, to detect any kind of anomaly. It is not sufficient if the CDRs are based on the SIP messages only, without knowing whether or not there was media connectivity. This might result in contra-productive network configuration, misinterpretation of someone's reputation or, even worse, will black-list a participant.

In this paper we present a solution for the *VoIP Media Connectivity Awareness Problem*, which fulfills the following requirements:

1) Focus: It is the *SIP Provider* who needs to obtain knowledge about the connectivity status.

2) Multiple (bi-directional) streams: It is important to consider *all media streams* negotiated between the calling parties. Any single uni-directional stream that is not established successfully might be the reason for one of the participant to end the call (immediately). Thus, the provider needs to determine at least the connectivity status for the stream aggregate, with respect to each stream in any direction.

3) Genuineness: In order to prevent false conclusions (and subsequent actions), the connectivity status gathered by the provider should be *genuine*.

4) Compatibility: The number of changes put into the SIP message sequences should be as small as possible. Ideally, neither extra SIP messages nor additional SIP headers should be required.

This paper is structured as follows: In Section II, we discuss several approaches that have some relation to the awareness of media connectivity. In Section III, our approach is presented. This includes detailed scenarios and preliminary investigation of the Stream Control Transmission Protocol (SCTP). Finally, Section IV contains the measurements of the overhead of our solution.

II. Related Work

There are some approaches that relate to the awareness of media connectivity but which are motivated by different goals.

A. Dealing with the NAT

One possibility to solve the connectivity problem is the use of an Application Layer Gateway (ALG) in addition to the NAT. In reality, however, ALGs are deployed in the fewest scenarios, even though most users manage their own private home networks. Furthermore, an ALG might increase the chance to achieve media connectivity, but the SIP provider still does not know about it.

Traversing the NAT for the media streams can be done using Interactive Connectivity Establishment (ICE) [18]. ICE describes NAT traversal for multimedia signaling protocols like SIP, and it extends the Session Description Protocol (SDP) [9] to convey additional data. In order to operate, ICE utilizes the protocols Session Traversal Utilities for NAT (STUN) [20] and Traversal Using Relays around NAT (TURN) [14].

The goal of ICE is to *establish* connectivity, but not to require it or to inform a third party of the connectivity status.

B. Connectivity Preconditions

UAs may use Connectivity Preconditions as defined in RFC 5898 [2] to *verify* whether there is connectivity or not. Based on the concept of a SDP precondition in SIP as specified by RFC 3312 [5] (generalized by RFC 4032 [4]), the connectivity precondition defined by RFC 5898 tries to ensure that session progress is delayed[3] until media stream connectivity has been verified.

Similar to a part of the solution described in this paper (cf. Sec. III), it enables the UAs to delay the SIP session establishment until connectivity is ensured. In contrast to our approach, the provider cannot enforce the UAs to make use

[3]including suppression of alerting the called party

 ISBN:978-1-61208-113-7

of this extension. In addition, it does not inform a third party (such as the provider) of the connectivity status – neither implicitly nor explicitly.

Furthermore, RFC 5898 does not assure that session establishment comes along with media connectivity. In RFC 3312 (which RFC 5898 relates to), alerting the user until all the mandatory preconditions are met has a "SHOULD NOT" semantics.

C. Disconnection Tolerance

Ott and Xiaojun [16] present mechanisms for detection and recovery from temporary service failures for mobile SIP users. For detection of connectivity loss, they suggest a media-based approach: Missing Real-time Transport Protocol (RTP) packets, RTP Control Protocol (RTCP) packets, or STUN packets along with some additional criteria are used as indicators that connectivity has been lost. If the connectivity loss persists longer ("call interruptions"), the UAs will automatically try to re-establish the session after locally terminating the session. For this purpose, the authors introduce the new SIP *Recovery* header field, which is set to `true` in the *INVITE* message used to re-establish the session. The focus of this paper is on obtaining the connectivity status during an ongoing session *after* the session has been established. Implicitly, it assumes that connectivity was given at the beginning of the session.

D. Conclusion

In all solutions presented the focus is always on the endpoints. Whether the main goal is to establish connectivity, ensure connectivity, detect/monitor connectivity status, or recover from connectivity loss, the assumption is always that the *endpoints* are the entities which are interested in the goal.

Hence, the provider is not aware of the media connectivity; and even when the connectivity information can be obtained, its validity and/or genuineness may be questionable.

III. Implicit Connectivity Detection and Notification

One major difference between the approaches presented above is *when* information pertaining to connectivity status is obtained. Three distinct cases can be identified: before session establishment (ICE, Connectivity Preconditions), after session establishment (Disconnection Tolerance [detection only]), and at the end of the conversation (Disconnection Tolerance [signaled through *Recovery* header field]). In the second case, the information can also be obtained continually during the ongoing session.

Another difference is found in the direction of a media stream for which connectivity status is determined and whether media streams are considered separately or jointly on a "session level." Most mechanisms distinguish between individual streams and, as streams are usually considered uni-directional, also between receiving and sending direction. Connectivity Preconditions distinguish both direction and individual streams, but the consequence (suspension of session establishment) is affected by the aggregate of the streams for which the precondition was requested. The Disconnection Tolerance solution disregards direction as symmetric connectivity is assumed; it also disregards individual streams because the existence of only one audio stream is assumed (point-to-point audio conversation).

In our approach, connectivity detection and notification is done before session establishment. Further, our solution regards both, different streams and direction.

A. Implicit Connectivity Notification

SIP itself already offers several possibilities to modify the message routing. For example, a SIP proxy can request to stay in the route of any further SIP messages. Any UA sending a new SIP request needs to insert corresponding routing information. Thus, in contrast to the normal SIP call (see Fig. 1) a proxy can become a mandatory node of the last SIP 3-way-handshake's message (i. e., the *ACK* request). Furthermore, the user agent server (UAS) does not necessarily need to send a *180 Ringing* response and notify the called person. Instead, it can respond with a *183 Session Progress* message to indicate further action prior to call acceptance.

This response message plus the modified message routing can be combined with a modified UA behavior. By using the *183* response's payload, the callee can answer the caller's SDP offer. Thus, both parties know the parameters of all media sessions that normally will be established *after* the SIP session has been accepted. In our solution, the media sessions are established *beforehand*, and both parties **must hold back** the *180 Ringing*, *200 OK*, and the *ACK* messages until this has happened. Furthermore, each UA **must ignore** any incoming media packets as long as the calling partner did not acknowledge the connectivity.

In result, the provider can conclude the media's connectivity status by just analyzing the messages it is routing. Therefore, we call the approach *implicit*. The provider will **conclude that there is connectivity if and only if** the UAS has sent a *200 OK* and then the user agent client (UAC) has sent an *ACK*.

In case the media connection could be established successfully, there will be a notification (*180 Ringing*), acceptance (*200 OK*) and acknowledgement (*ACK*) (see Fig. 2). In result, the provider concludes that there is media connectivity.

If the UAS notices that establishing the media connection failed, it will reject the call by sending a *418* error response (see Fig. 3). If the failure is detected by the UAC (similar to Fig. 4, not shown separately), it will cancel the call using the *CANCEL* request causing the UAS to respond to the

 ISBN:978-1-61208-113-7

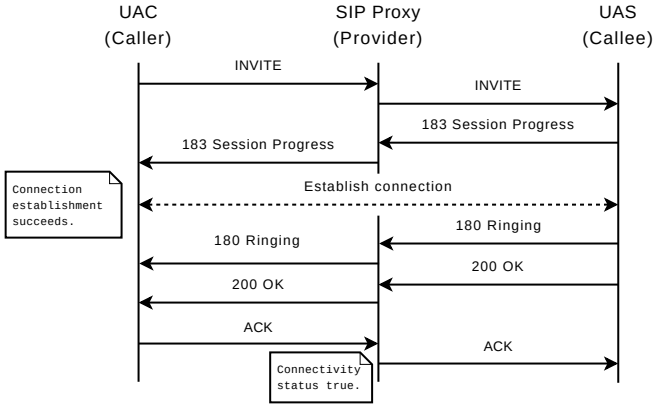

Figure 2: Accepted Call with Prechecked Media Connectivity

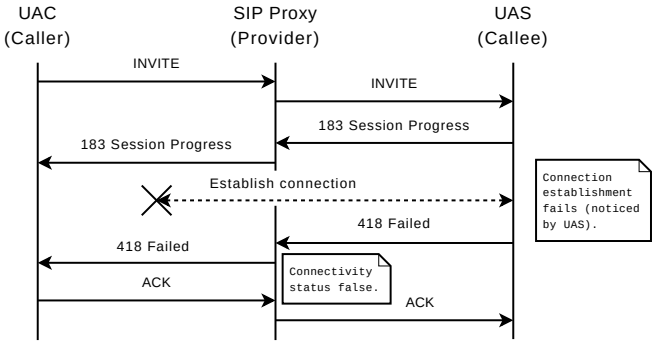

Figure 3: Call Abortion in the Case of no Media Connectivity

invitation with *487 Request Terminated*. In both cases, the provider concludes that there is no media connectivity.

In Figure 4, the media connection has been established successfully but the callee is unavailable. Thus, the caller will cancel the call when a timeout appeared. Again, the provider concludes lack of media connectivity.

B. Detection Connectivity

Due to the fact that the provider is simply analyzing the messages it is routing, it is up to the clients to verify the connectivity status. In detail, they need to check every single media stream for connectivity (cf. Requirement 2). This can be complex and time consuming.

In order to limit this overhead, we propose the use of the Stream Control Transmission Protocol (SCTP) [25] as the media's underlying transport layer. First of all, SCTP is connection oriented; thus, the SCTP's 4-way-handshake at the beginning already ensures transport layer connectivity. In result, neither a media packet nor a notice of receipt need to be sent in order to check for connectivity. Secondly, SCTP itself offers multiplexing; so there is no need for more than one connection, as every single RTP/RTCP stream can be sent using the same unique connection. In result, the time

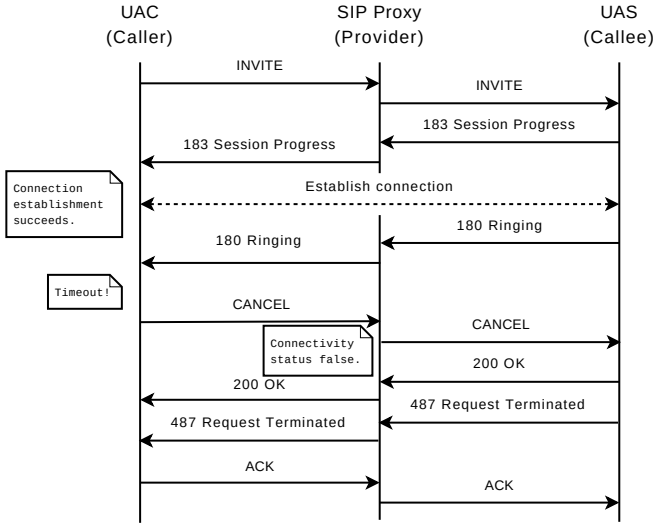

Figure 4: Timed-out Call with Prechecked Media Connectivity

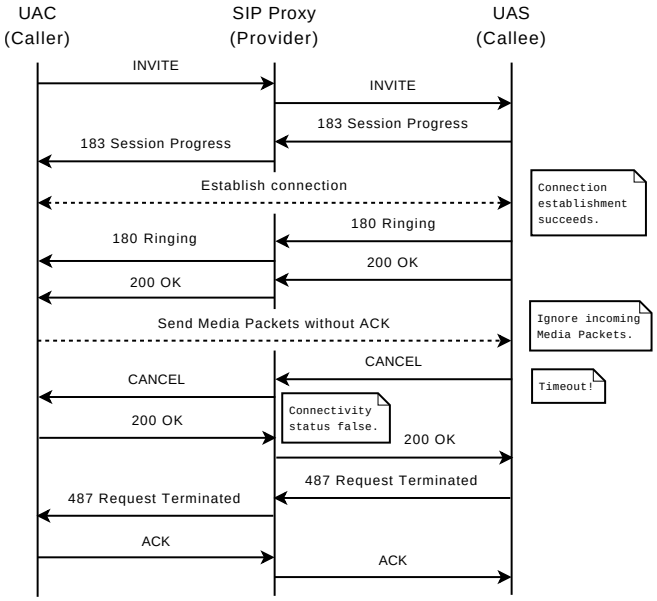

Figure 5: Missbehaving UAC

required to check each media stream (and media control stream) is reduced to a single check only. Last but not least, in contrast to TCP, SCTP offers unordered transport, meaning a lost packet does not delay delivery of succeeding packets. In addition, the partial reliable mode can be used to improve the media quality in case a lost packet can be resent immediately.

To confirm our proposal, we measured the SCTP performance in comparison to UDP. The environment consists of two machines with identical hardware and software running Debian GNU/Linux 5.0.3 (lenny) with kernel version 2.6.26

 ISBN:978-1-61208-113-7

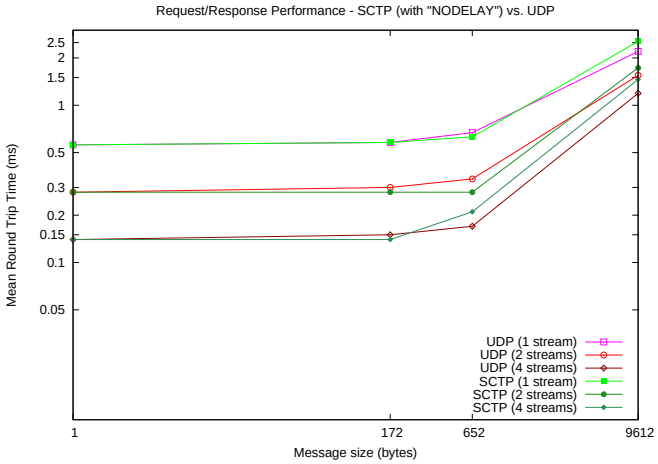

Figure 6: SCTP vs. UDP

(i686). Both machines are equipped with an Intel Core 2 Duo E7500 dual core CPU running at 2.93 GHz and an Intel 82567LM-3 network adapter and connected via a FastEthernet switch (100 Mbps Full Duplex). The environment also determines the UDP and SCTP implementations used – those of the Linux kernel. The benchmark itself is a simple ping-pong application that can send multiple messages at once, approximating multiple concurrent media streams. Figure 6 shows the mean round-trip time (RTT) in relation to the size of the messages. The sizes of 172 Bytes and 652 Bytes correlate to the RTP packet sizes produced by the G.711 codec using packet transmission cycles of 20 ms and 80 ms, respectively. One can see that the values of SCTP are quite similar to UDP, and hence, we expect no performance loss due to the use of SCTP.

C. Missbehaving user agents

In some cases, either the UAS or the UAC might try to falsify the information it tells about the connectivity status. Our solution would require to send a *200 OK* (UAS) or *ACK* (UAC) to convey "connectivity."

For example, a caller might announce "no connectivity" to the provider in order to send SPIT calls without consequences. In our solution, the UAC would have to suppress the *ACK* message. Fortunately, this would cause the callee to ignore any incoming media packet (cf. Fig. 5). In the payment example, the callee might say "connectivity" in order to receive his fee anyway. In this case, the caller receives a *200 OK* even though there is no media connectivity. In result, he can abort the call by sending a *CANCEL* request. In general, for whatever reason a UA might missbehave – our solution enables the other party to react appropriately, enabling the provider to know the actual connectivity status.

The case that both, caller and callee, are lying cooperatively, this is only a problem in the forensics scenario. It is doubtful, however, that the calling partners would use a provider at all in such a scenario.

D. Protocol Extensions

There has been some work in the past for SCTP and SIP, but unfortunately, it is incomplete and has been abandoned [6], is limited in its scope [13], and does not deal with the use of RTP over SCTP.

In line with the last requirement, our solution only needs to slightly extend the abilities of SDP in order to specify the SCTP parameters. The use of the SCTP connection and the modified UA behavior can be indicated by naming our extension (i. e., *sctp-tunnel*) within a *Require* header. If an incoming *INVITE* does not indicate usage of this extension the provider must reject this request by sending *421 Extension Required*. As described above, the extension just specifies the way the UAs must behave and the provider can draw conclusions; it does not specify any new SIP messages or headers – all of them have existed before. The syntactical details of both modifications can be found in [8].

IV. Measurements

Although we minimized the changes to the existing VoIP infrastructure, the provider still has to be aware of the *sctp-tunnel* extension indicated within the SIP messages. Whether the extension is stated or not, the provider has to use different message handling and routing. It is thus important to know how much overhead our extension creates.

Note that the following measurements do not cover the impact of SCTP. SCTP is used as the underlying protocol of the UA-to-UA media session only, whereas SIP messages still use UDP. In result, a SIP proxy does not need to be adapted to use another transport protocol. On the other hand, media gateways (not considered by the following measurements) need to be altered to conform to our approach.

A. Testbed, Scenarios

We used three nodes (each with 2 x AMD Opteron 244 CPU (1.8 GHz), 4 GB RAM, Gigabit Ethernet Interconnection) to setup one SIP proxy (Kamailio [15], v3.0.3) and two UAs (SIPp [7], v3.1) that generated and processed a various number of SIP calls. Kamailio has been configured to use 1024 MB of memory, to create four processes, and its log level was set to zero.

In general, we measured three scenarios: a) default behavior of the proxy, b) modified behavior of the proxy where the UAs already indicated the use of the *sctp-tunnel* extension, and c) the modified behavior of the proxy without initial indication by the UAs. The third scenario is the most expensive one since the provider needs to reject incoming invitations first, and then has to deal with the reformulated ones. In addition, we measured d) the SIPp-SIPp-interconnectivity to determine the overhead of Kamailio in general.

In scenarios a) and b), the UAC and the UAS send and receive SIP messages according to Figure 7. In contrast to

 ISBN:978-1-61208-113-7

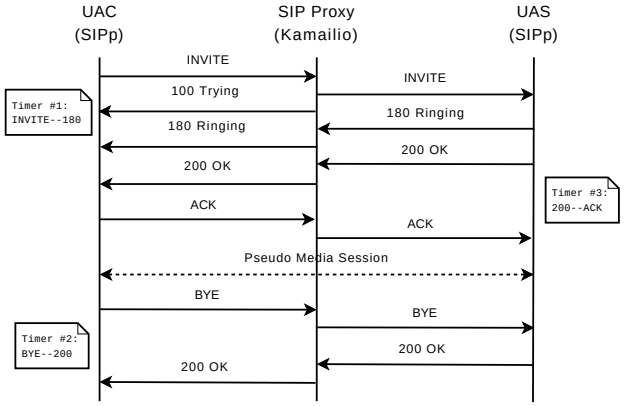

Figure 7: Measurement Scenario

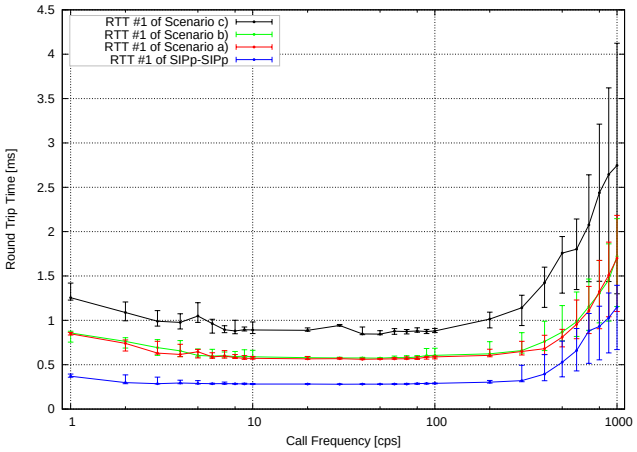

Figure 8: Comparison of RTT #1

the example given in Figure 1, the proxy stays in the route for the whole call. Not shown separately, scenario c) requires three more messages at the beginning: the first *INVITE* will be rejected with a *421* response that has to be *ACK*ed.

Each call generates three round-trip time values: RTT #1 represents the delay of a UAS's response including Kamailio action (such as lookup and extension verification); RTT #2 represents the delay of a UAS's response in case the request can be forwarded immediately; RTT #3 represents the delay of a UAC's feedback. Each series lasted five minutes, using a constant call frequency (between 1 and 1000 calls per second). The proxy and the UAs were restarted for each frequency.

B. Results

For all scenarios and each frequency we calculated the corresponding median and quartile values for each RTT. As expected, in scenarios a)–c), the values of RTT #2 and RTT #3 are nearly the same. The SIPp-SIPp interconnection's second and third RTT are ∼0.25–0.55 ms lower only. RTT #2 and #3 are not shown separately since our proposal does not alter the way *200 OK* and *ACK* messages or session tear down is handled.

The comparison of RTT #1 is shown in Figure 8. Again, one can see the additional time required (∼0.5–0.6 ms) when Kamailio is put between the SIPp instances. Furthermore, we expected the overhead of the header verification to be very small since we only slightly modified the routing logic of Kamailio. This small RTT increase can be seen when comparing the values of scenarios a) and b).

In scenario c), where the proxy had to enforce the use of the SIP extension, RTT #1 increases a little more. This happens because Kamailio is involved one more time and three more messages are sent until the first callee's response is received by the inviting caller.

V. Conclusion

In this paper, we have given several scenarios motivating the need of SIP providers' awareness of media connectivity, such as payment, reputation, forensics, and call detail record analysis.

In our solution, the provider is *implicitly* informed about the media connectivity: The SIP provider can draw genuine conclusions by simply analyzing the messages it is routing. The UA, however, needs to alter its behavior. This behavior is specified by way of a new SIP extension and its usage can be enforced by the provider.

To reduce the overhead of media connectivity detection, we propose to use SCTP for media transport. This requires a slight extension of SDP.

The measurements have shown that the overhead induced by our solution is neglectable, as long as the UAs indicate the use of our extension from the beginning. In addition, our approach can easily be integrated into existing VoIP infrastructures as it fully conforms to existing protocols. If a UA is not aware of our extension it is at the discretion of the provider to proceed with the call (without the ability to conclude media connectivity) or to reject it.

Future work will deal with Quality of Service (QoS) aspects. Besides a lack of connectivity, low quality can also cause a call to be aborted prematurely by one of the participants. We therefore need to conduct further investigation in order to deal with this problem.

References

[1] Alessandro Amirante, Simon Pietro Romano, Kyung Hwa Kim, and Henning Schulzrinne. Online Non-Intrusive Diagnosis of One-Way RTP Faults in VoIP Networks Using Cooperation. In Georg Carle, Helmut Reiser, Gonzallo Camarillo, and Vijay K. Gurbani, editors, *Proceedings of the 4th International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm 2010)*,

pages 153–160, Munich, Germany, August 2^{nd}–3^{rd}, 2010. Technical University Munich.

[2] F. Andreasen, G. Camarillo, D. Oran, and D. Wing. Connectivity Preconditions for Session Description Protocol (SDP) Media Streams. RFC 5898 (Proposed Standard), July 2010.

[3] Vijay A. Balasubramaniyan, Mustaque Ahamad, and Haesun Park. CallRank: Combating SPIT Using Call Duration, Social Networks and Global Reputation. In *Proceedings of the 4^{th} Conference on Email and AntiSpam, CEAS 2007*, August 2^{th}–3^{rd}, 2007.

[4] G. Camarillo and P. Kyzivat. Update to the Session Initiation Protocol (SIP) Preconditions Framework. RFC 4032 (Proposed Standard), March 2005.

[5] G. Camarillo, W. Marshall, and J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP). RFC 3312 (Proposed Standard), October 2002. Updated by RFCs 4032, 5027.

[6] R. Fairlie-Cuninghame. Guidelines for specifying SCTP-based media transport using SDP. Internet-Draft draft-fairlie-mmusic-sdp-sctp-00, Internet Engineering Task Force, May 2001. Work in progress.

[7] Richard Gayraud, Olivier Jacques, et al. SIPp: An Open Source Performance Testing Tool for SIP [v3.1]. http://sipp.sourceforge.net, March 17^{th}, 2009.

[8] Markus Gusowski. Media Connectivity in VoIP Infrastructures: Requirements, Detection, Enforcement. Master's thesis, University of Potsdam, July 2010.

[9] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566 (Proposed Standard), July 2006.

[10] M. Holdrege and P. Srisuresh. Protocol Complications with the IP Network Address Translator. RFC 3027 (Informational), January 2001.

[11] C. Jennings, J. Fischl, H. Tschofenig, and G. Jun. Payment for Services in Session Initiation Protocol (SIP). Internet-Draft draft-jennings-sipping-pay-06, Internet Engineering Task Force, July 2007. Work in progress.

[12] S. Liske, K. Rebensburg, and B. Schnor. SPIT-Erkennung, -Bekanntgabe und -Abwehr in SIP-Netzwerken. In U. Ultes-Nitsche, editor, *Proceedings of KiVS – NetSec 2007, Workshop „Secure Network Configuration"*, pages 33–38, February 2007.

[13] S. Loreto and G. Camarillo. Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP). Internet-Draft draft-loreto-mmusic-sctp-sdp-05, Internet Engineering Task Force, February 2010. Work in progress.

[14] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766 (Proposed Standard), April 2010.

[15] Ramona-Elena Modroiu, Bogdan Andrei Iancu, Daniel-Constantin Mierla, et al. Kamailio (OpenSER) [v3.0.3]. http://www.kamailio.org/, August 19^{th}, 2010.

[16] Jörg Ott and Lu Xiaojun. Disconnection tolerance for SIP-based real-time media sessions. In *MUM '07: Proceedings of the 6^{th} international conference on mobile and ubiquitous multimedia*, pages 14–23, New York, NY, USA, 2007. ACM.

[17] J. Rosenberg. The Session Initiation Protocol (SIP) and Spam. Internet-Draft draft-ietf-sipping-spam-03, Internet Engineering Task Force, October 2006. Work in progress.

[18] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 (Proposed Standard), April 2010.

[19] J. Rosenberg and G. Camarillo. Examples of Network Address Translation (NAT) and Firewall Traversal for the Session Initiation Protocol (SIP). Internet-Draft draft-rosenberg-sipping-nat-scenarios-03, Internet Engineering Task Force, July 2004. Work in progress.

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

[21] J. Rosenberg and H. Schulzrinne. An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing. RFC 3581 (Proposed Standard), August 2003.

[22] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922.

[23] D. Senie. Network Address Translator (NAT)-Friendly Application Design Guidelines. RFC 3235 (Informational), January 2002.

[24] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), January 2001.

[25] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), September 2007.

 ISBN:978-1-61208-113-7