Publication VIII

Jouni Mäenpää. Reducing P2PSIP Session Setup Delays. In *2013 IEEE International Conference on Computing, Networking and Communications (ICNC 2013)*, San Diego, USA, pp. 1-6, January 2013.

Reducing P2PSIP Session Setup Delays

Jouni Mäenpää
Ericsson Research
jouni.maenpaa@ericsson.com

***Abstract*—It has been shown in previous work that session setup delays in a Peer-to-Peer Session Initiation Protocol (P2PSIP) based Voice over IP (VoIP) system can be unacceptably high. This is due to the need to route several messages across the P2P overlay and also due to the fact that P2PSIP needs to establish separate connections for SIP signaling and Real-Time Protocol (RTP) media between the communicating parties across Network Address Translators (NATs). In this paper, we will develop optimizations to reduce P2PSIP session setup delays. We will also analyze the performance of these optimizations. Using the proposed optimizations, it is possible to achieve delays that are up to three times lower than those of unoptimized P2PSIP.**

***Index Terms*—P2PSIP, ICE, RELOAD, session setup delay**

I. Introduction

Peer-to-Peer Session Initiation Protocol (P2PSIP) is a new decentralized communication system that is being standardized in the Internet Engineering Task Force (IETF). The main idea in P2PSIP is to replace the centralized proxy and registrar servers of client/server SIP [1] with a Distributed Hash Table (DHT) based P2P overlay network. P2PSIP uses a protocol called REsource LOcation And Discovery (RELOAD) [2] for rendezvous, data storage, and exchange of overlay maintenance messages. P2PSIP uses Chord [3] as the mandatory-to-implement DHT algorithm. For Network Address Translator (NAT) traversal, P2PSIP utilizes the Interactive Connectivity Establishment (ICE) [4] NAT traversal mechanism that has been standardized by the IETF.

We have shown in previous work [5] that P2PSIP session setup delays can be unacceptably high, especially when the communicating parties are located behind the most restrictive types of NATs. Our results in [5] indicate that in the worst case, session setup can take as long as over 30 seconds. To put this figure in perspective, the ITU-T recommendation for average call setup delay of international calls is 8.0s [6]. Therefore, it is clear that optimizations are needed to bring P2PSIP session setup delays down.

In this paper, we will design and evaluate different approaches for reducing P2PSIP session setup delays. The remainder of the paper is structured as follows. Section II gives an introduction to ICE and P2PSIP session setup. Section III presents related work. Section IV introduces the optimizations that we propose and describes the setup of the experiments that we used to evaluate the proposed optimizations. Section V presents the results of the experiments. Finally, Section VI concludes the paper.

II. Technologies

This section will give an introduction to key topics in this paper: ICE and the P2PSIP session setup procedure.

A. NAT Traversal Using ICE

ICE [4] is a NAT traversal technique that has been designed by the IETF. In ICE, hosts that wish to establish a direct connection first gather a set of candidate addresses that can potentially be used for communication. There are three main types of candidate addresses. A host candidate is the transport address (i.e., IP address and port) of a local network interface. A server reflexive candidate is an address that has been assigned to a host by a NAT. A relayed candidate is a transport address that has been allocated for a host at a relay server for the purpose of relaying traffic. Having gathered the candidates, the hosts run connectivity checks to test connectivity between pairs of candidate addresses. The checks are done in priority order in such a way that host candidates have the highest and relayed candidates the lowest priority.

ICE makes use of Session Traversal Utilities for NAT (STUN) [7] and Traversal Using Relays around NAT (TURN) [8] protocols. STUN is used to gather candidates, for the connectivity checks, and for keep-alive signaling. TURN is used for communicating with a relay server.

There are several ICE parameters with which we experiment in this paper. The *Ta* parameter defines the interval at which ICE paces STUN and TURN transactions. *N* is the maximum number of connectivity checks that a host can perform. *Rc* determines the maximum number of times that STUN requests are transmitted. *Retransmission TimeOut (RTO)* is the STUN retransmit timer. The *stopping criterion* determines the maximum time that connectivity checks are run before selecting one of the valid candidate pairs.

B. P2PSIP Session Setup Procedure

Figure 1 shows the P2PSIP session setup signaling flow. In the figure, Alice establishes a VoIP session with Bob. Alice and Bob are acting as RELOAD clients that are connected to a P2PSIP overlay. Alice is using Peer 1 as her TURN server, whereas Bob is using Peer 2. In steps 1-2, Alice performs a RELOAD lookup (i.e., Fetch) operation to fetch Bob's contact information from the overlay. Next, Alice starts the ICE connection establishment procedure for SIP. In steps 3-4, Alice performs ICE candidate gathering. Alice sends her candidates to Bob via the overlay in a RELOAD AppAttach request in steps 5-6. Having received the candidates, Bob

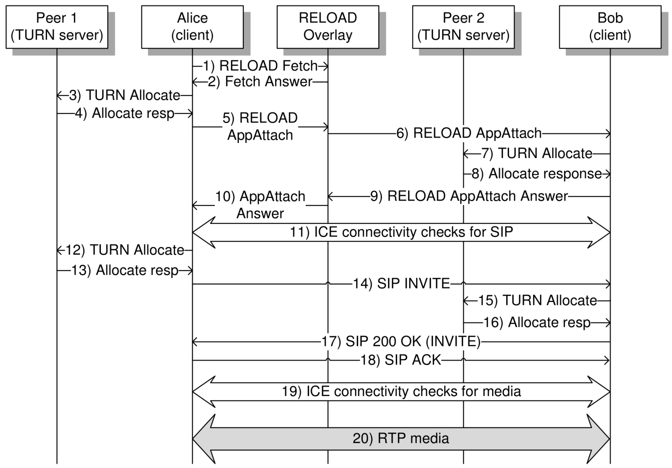

Fig. 1. P2PSIP call setup

gathers his own candidates in steps 7-8. In steps 9-10, Bob sends his candidates to Alice via the overlay. Next, Alice and Bob perform ICE connectivity checks for SIP in step 11. The result of the checks is a direct UDP connection for SIP between Alice and Bob. In steps 12-13, Alice performs ICE candidate gathering for RTP. In step 14, Alice sends a SIP INVITE request carrying the candidates to Bob. Having received the INVITE, Bob gathers his ICE candidates for RTP in steps 15-16. In step 17, Bob sends a SIP 200 OK response carrying his candidates to Alice. The 200 OK is acknowledged in step 18. ICE connectivity checks for RTP are done in step 19. Finally, in step 20, RTP media starts to flow between Alice and Bob.

In this paper, session setup delay refers to the delay between the caller starting the lookup operation in step 1 of Figure 1 and the ICE connectivity checks being finished in step 19. The callee is assumed to always accept the session immediately.

III. Related Work

In [5], we study P2PSIP session setup delays with different NAT types and show that the delays can be unacceptably high. In the worst case, session setup can take over 30 seconds. The difference between [5] and this paper is that in this paper, we will develop mechanisms to reduce the session setup delay.

In [9], we show how P2PSIP session setup delays can be reduced by running P2PSIP on top of the Host Identity Protocol Based Overlay Networking Environment (HIP BONE) [10] framework. The difference to the present work is that the mechanisms in this paper do not need to rely on HIP BONE. Further, the optimizations in this paper result in significantly lower session setup delays than the HIP BONE approach.

In [11], an alternative to ICE called Address List Extension (ALEX) is proposed. ALEX attempts to reduce SIP session setup delays when NATs are present. The difference to our work is that we focus on P2PSIP rather than SIP and that we use standard ICE rather than an alternative to ICE.

IV. Experiments

A. P2PSIP Simulator

The results presented in this paper were obtained from our P2PSIP simulator, which is an event-driven, message-level simulator. It has been implemented in the Java programming language. The simulator uses the same code base as our a real-world P2PSIP implementation that we have used in previous work [12], [13], [5] to run experiments in PlanetLab. The simulator uses a topology generator that assigns peers randomly to 206 different locations around the world, which correspond to PlanetLab sites. The pairwise delays between simulated peers are generated from a distribution of real pairwise delays that we measured between these 206 PlanetLab sites.

The simulator has been used extensively in previous work [14], [15], [16]. The simulator has been validated in [16] by comparing its results to results obtained by running the same code base in the real Internet.

B. Traffic Model and Parameters

The measurements were carried out in a simulated 5000-peer P2PSIP overlay network. The network was created during the first 9250s of the simulations. During this period, the size of the overlay grew from zero to 5000 peers. The measurements started when the size of the overlay reached 5000 peers and were continued for one hour of simulated time. During this one hour period, the overlay was experiencing churn. The churn, that is, the arrival and departure of peers, was modeled as a Poisson process. The join and leave intervals of peers were exponentially distributed with a mean of 5.76s, which corresponds to a mean online time of 8 hours. As an example, a global company-internal P2PSIP VoIP system might have an 8-hour (full working day) mean online time. By online time, we refer to the time between a peer joining the system and leaving it. Since the mean join and leave intervals were identical, the size of the overlay stayed constant at 5000 peers during the one-hour measurement period.

During the one-hour period, peers were initiating calls. This period was modeled as a busy hour. The number of busy hour call attempts per user was 2.21. This rate of call attempts was chosen based on the results in [17], [18]; in [17], it is suggested that VoIP users initiate 13 calls per day. Further, [18] states that 17% of calls per day can be used to represent busy hour traffic, which results in a number of busy hour call attempts equal to 2.21. This number was used as a mean rate for the arrival of calls, which was modeled as a Poisson process.

All the users initiating calls were P2PSIP clients. The clients were using 3G High Speed Downlink Packet Access (HSDPA) access. In contrast to the clients, all peers participating in the P2PSIP overlay were using fixed broadband access. For the delays between the clients and between clients and peers, the delay generator used a large set of delays that we collected in [15] through measurements in a 3G HSDPA network with a high number of packet sizes ranging from 10 to 1400 bytes.

The sizes of the Chord finger table and successor list were set to 12 following the recommendation in [3] to use on the order of $O(\log N)$ fingers and successors. The size of the predecessor list was set to 3 based on the minimum recommended in [2]. The Chord stabilization interval was set to 95s based on the formula presented in [19] that guides how to set the interval for a given churn rate and network size.

ICE uses different *Ta* values for RTP and non-RTP sessions. For RTP sessions, we use the standard formula in the ICE RFC for setting the value of the *Ta* timer. This results in a minimum *Ta* value of 20ms. For SIP sessions, we use initially the default *Ta* value of 500ms specified in the ICE RFC and later experiment with other values.

We use initially the default value of 7 specified in the STUN RFC for *Rc* and later experiment with values 6 and 5. The *RTO* was set using the standard formula in the ICE RFC. This formula results in a minimum *RTO* value of 100ms.

For *N*, we use initially the default value of 100 specified in the ICE RFC and later experiment with an alternative value.

We use an ICE stopping criterion consisting of two steps: a soft deadline and a hard deadline. When the soft deadline is reached, the hosts will try to nominate any valid candidate pair that is not using a relay. If no such pair has been discovered, the hosts will wait until the hard deadline is reached to see if any candidate pair not using a relay succeeds. If not, the hosts will nominate the highest priority pair utilizing a relay.

The soft deadline was set to 2s. This value was chosen because the International Telecommunication Union (ITU) recommendation for average post-selection delay on local connections is 3s [6]. A 2s delay ensures that there is still 1s left for signaling before and after ICE that can be used for instance to perform candidate exchange over SIP.

The hard deadline was initially set to 10s. This value was chosen due to the reasons explained below. First, when using the default values for *Ta*, *N*, and *RTO*, the final connectivity check will time out roughly after 9.9s from the start of the checks, assuming that checks are paced at 20ms and that the maximum number of checks is reached. Second, a 10s delay ensures that there is a good chance of finding a path if one exists even with a rather high RTT and some packet loss. Third, in person-to-person communication, a user is not likely to tolerate a delay longer than 10s. Fourth, a 10s value is also used by other widely used ICE implementations [20]. Our traffic model and parameters are summarized in Table I.

TABLE I
TRAFFIC MODEL AND PARAMETERS

Parameter	**Value**
Peer interarrival & departure time	5.76s
Network size (N)	5000 peers
Busy hour call attemps	2.21 calls/user
Fingers/successors/predecessors	12/12/3
Chord stabilization interval	95s
ICE Ta timer, RTP (min)	20ms
ICE Ta timer, SIP (min)	500ms or 20ms
ICE soft deadline	2s
ICE hard deadline	10s, 7s, 5s, or 4s
STUN Rc	5, 6, or 7
ICE RTO	100ms
Maximum number of checks	50 or 100

C. Optimizations

The optimizations that we propose and analyze in this paper to reduce P2PSIP session setup delays are described in the subsections below.

1) Unoptimized P2PSIP: This scenario uses the standard P2PSIP session establishment flow described in Figure 1. In this scenario, no optimizations have been applied.

2) Timers & joint candidate gathering: This optimization introduces two changes to the unoptimized P2PSIP session setup procedure. The first change is to modify the *Ta* value that ICE uses for non-RTP sessions; instead of using a *Ta* with a minimum of 500ms as specified by the ICE RFC for non-RTP sessions, we use the same minimum *Ta* value that ICE uses for RTP (i.e., 20ms). The reason ICE uses a more conservative *Ta* value for non-RTP sessions is that it might not be known whether an aggressive rate will work for non-RTP sessions in the network in which ICE is deployed. However, since our experiments with ICE in a P2PSIP overlay network running in the real Internet in [5] showed that *Ta*=20ms appears to work without problems also for non-RTP sessions, we chose to use that value also in this optimization. The second change that we make is that we gather candidate addresses simultaneously for SIP and RTP at the calling party. This change halves the time that candidate gathering requires. The call setup flow of the *Timers & joint candidate gathering* optimization is identical to Figure 1 except for the parallel gathering of SIP and RTP candidates.

3) Reuse ICE result: This optimization makes use of the result of ICE connectivity checks for SIP when prioritizing the candidate pairs for RTP. As an example, if the ICE checks for SIP resulted in the selection of a candidate pair that uses a relay, the corresponding pair will be assigned the highest priority in the ICE checks for RTP. The benefit of this optimization is that it removes the need to re-test the paths that are already known to fail with a high probability as a result of the first ICE negotiation. Thus, the optimization can significantly reduce the session setup delay especially in cases where a relay needs to be used. The call setup flow of the *Reuse ICE result* optimization is identical to Figure 1.

4) SIP and RTP Mux: This optimization multiplexes SIP and RTP on the same port. The benefit of such multiplexing is that it eliminates the need to perform an ICE negotiation for RTP. This is because the connection that ICE established for SIP can be reused for RTP. The downside is that multiplexing requires changes to SIP and RTP stacks; there is a need to implement a multiplexing layer between the socket layer and the rest of the protocol stack. The task of the multiplexing layer is to determine whether the received datagram carries a SIP message or RTP packet and deliver the datagram to either the SIP stack or the RTP stack. The signaling flow that the *SIP and RTP Mux* optimization uses is illustrated in Figure 2. The differences compared to the standard P2PSIP session setup flow shown in Figure 1 include that no ICE candidates need to be gathered and that no ICE connectivity checks need to be executed for RTP. Steps 1-11 of Figure 2 are identical to

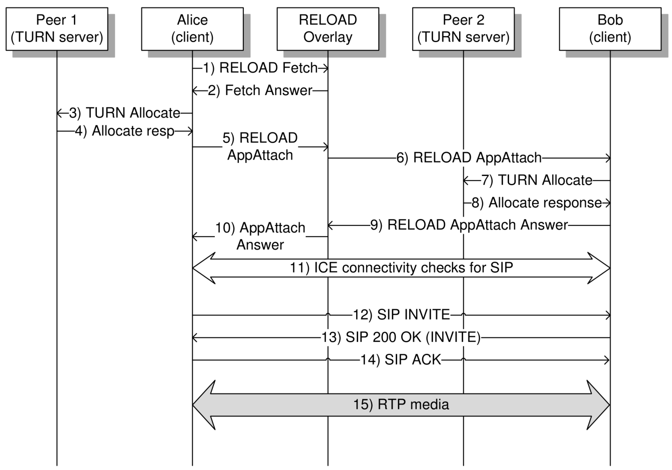

Fig. 2. P2PSIP call setup, SIP and RTP Mux optimization

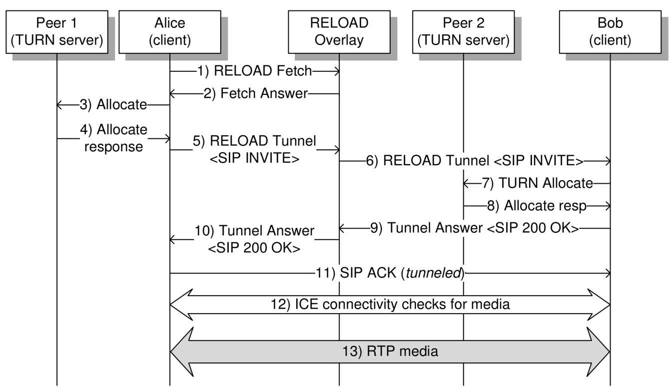

Fig. 3. P2PSIP call setup, SIP via Overlay optimization

Figure 1. In step 12, a SIP INVITE request is sent to the called user. The called users accepts the call by returning a SIP 200 OK response in step 13. Neither the INVITE nor the 200 OK carry ICE candidates. This is because RTP packets will be sent on the already existing connection being used for SIP signaling. Thus, RTP packets can start flowing between the users in step 15 directly after the SIP signaling has finished, without the need to run ICE connectivity checks.

5) SIP via Overlay (SvO): In the *SIP via Overlay (SvO)* optimization, we use RELOAD to transport SIP messages, that is, we send SIP messages encapsulated in RELOAD messages across the overlay. For this purpose, we use a new RELOAD Tunnel message. The Tunnel message is similar to any other RELOAD message with the key difference that a Tunnel request and its response can carry application data (i.e., SIP messages) in their payloads. The benefit of the SvO approach is that it eliminates the need to perform an ICE negotiation for SIP. Thus, it results in significant reduction in the session setup delay. The signalling flow used by the SvO optimization is illustrated in Figure 3. The differences compared to Figure 1 include, in addition to the encapsulation of SIP messages in RELOAD Tunnel messages, that the AppAttach transaction is not needed and that there is no need to gather ICE candidates or run connectivity checks for SIP. In steps 1-2 of the figure, the called user's Node-ID is fetched from the overlay. In steps 3-4, ICE candidates are gathered for RTP. In steps 5-6, a RELOAD Tunnel request carrying the SIP INVITE request is sent to the called user. In steps 7-8, the called user gathers ICE candidates for RTP. In steps 9-10, the ICE candidates are returned in a SIP 200 OK response that is included in the payload of a RELOAD Tunnel answer. In step 11, a SIP ACK is tunneled across the overlay. The empty Tunnel answer has been omitted from the figure for brevity. In step 12, ICE checks are run for RTP. Finally, in step 13, the ICE checks have finished and RTP packets are flowing between the users.

6) SvO Rc=6: This optimization, in addition to using the same mechanisms as the *SvO* optimization above, also modifies the STUN *Rc* parameter by setting its value to 6 instead of the default value of 7. Reducing *Rc* allows us to drop the hard deadline of the ICE stopping criterion to 7s. This is possible since if we assume that ICE checks are paced exactly at 20ms (i.e., *Ta*=20ms), *Rc*=6, *RTO*=100ms, and *N*=100, the last check completes at the latest at 6.7s.

7) SvO Rc=5: This optimization uses the same mechanisms as the *SvO RC=6* optimization with the exception that we set the *Rc* parameter to 5. This allows us to drop the hard deadline further down to 5s since when reducing *Rc* to 5, the last check completes at the latest at 5s if the checks are paced at 20ms.

It should be noted that the risk with reducing the number of STUN retransmissions (i.e., using *Rc* values of 5 and 6) is that in some rare cases ICE checks might really require 7 STUN transmissions to succeed. However, in our simulations for this paper and in our experiments in [5] where we used ICE to establish connections between two mobile phones in a P2PSIP overlay running in PlanetLab, we did not observe any degradation in performance with *Rc*=5; if an ICE check did not succeed with 5 transmissions, it was not likely to succeed with further transmissions either given the RTTs in our experiments.

8) SvO Rc=5 N=50: In this optimization, we use *Rc*=5 and also limit the total number of connectivity checks (*N*) that a host can perform to 50 instead of using the ICE default limit of 100 that we used in the optimizations above. This allows us to drop the hard deadline to 4s since with *Rc*=5 and *N*=50, all the checks have completed roughly at 4s if paced at 20ms. The risk with reducing *N* to 50 is that in some rare cases ICE might be able to find a more optimal candidate pair if more checks were allowed. However, in our experiments and when running ICE in a P2PSIP overlay in PlanetLab in [5], we did not experience any performance degradation with *N*=50.

9) SvO SigComp: In this optimization, we use the same mechanisms as in the *SvO Rc=5 N=50* optimization and additionally, also use Signaling Compression (SigComp) [21] to compress SIP messages. SigComp is an IETF-specified mechanism for compressing the messages of application protocols. The benefit of using SigComp is that it reduces the size of SIP messages considerably, which results in a lower propagation delay for them on the 3G radio link. Compressing SIP messages is worth the effort since due to the textual

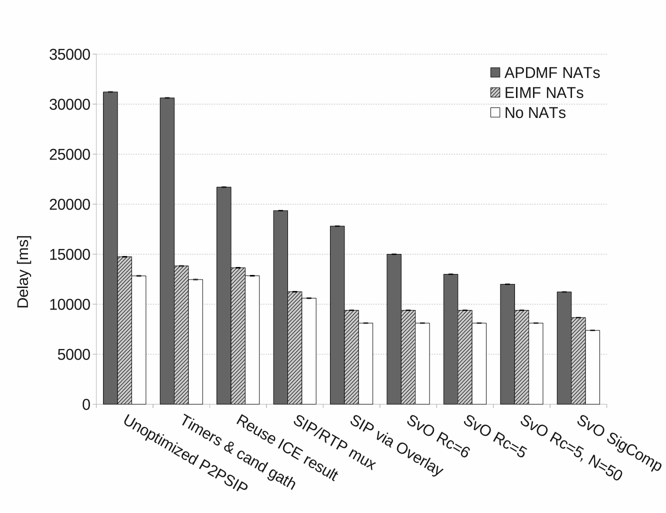

Fig. 4. Total session establishment delay

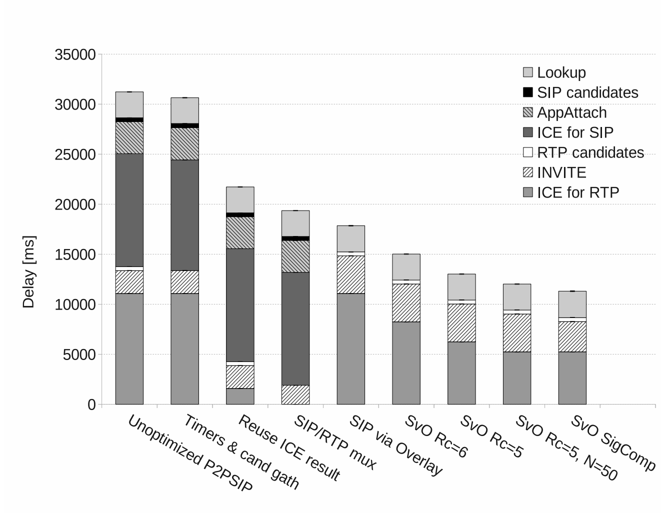

Fig. 5. Session establishment delay, APDMF NATs

encoding that SIP uses, SIP messages are multiple times larger than binary-encoded RELOAD and STUN messages.

In our experiments, we will study the performance of each of the optimizations above in three different scenarios. In the *No NATs* scenario, both communicating parties are publically reachable, that is, not located behind NATs. In the *EIMF NATs* scenario, the parties are located behind NATs using Endpoint Independent Mapping and Filtering (EIMF) behavior. In the *APDMF NATs* scenario, the parties are located behind the most restrictive types of NATs, that is, NATs using Address and Port Dependent Mapping and Filtering (APDMF) behavior. The main difference between the *EIMF NATs* and *APDMF NATs* scenarios is that in the latter, the parties need to use a TURN relay server to relay all traffic between them.

V. RESULTS

This section presents the results of our experiments. Our overall results for all the optimizations are shown in Figure 4 for all the three NAT scenarios. The error bars in the figure and all the other figures of this section represent 95% confidence intervals. Note that in many cases, the error bars are very small since thousands of calls were made during the measurement period. In all the figures, the differences between different optimizations and NAT scenarios are statistically significant at the 95% confidence level. From Figure 4, we can observe that unoptimized P2PSIP has the highest delays in all the three NAT scenarios. In the worst case, that is, in the APDMF NATs scenario, the session setup delay of unoptimized P2PSIP is as high as 31.2s.

For all the three NAT scenarios, the lowest delays are achieved in the *SvO SigComp* scenario. In case of the APDMF NATs scenario, the delay is reduced by 64%, that is, the delay is only 1/3 of the delay of unoptimized P2PSIP. Therefore, we can conclude that the optimizations seem to be very effective in reducing the session setup delay.

We will analyze the session setup delays of each of the three NAT scenarios in more detail in the subsections below.

A. APDMF NATs

The components of the session setup delay are shown for the APDMF NATs scenario in Figure 5. From the figure, we can see that the performance gain of the *Timers & joint candidate gathering* optimization is rather small, 594ms. In contrast, the *Reuse ICE result* optimization achieves a considerable improvement (a reduction of 30% to 21.7s) compared to unmodified P2PSIP. This is because the delay associated with running the ICE connectivity checks for RTP is reduced from 11.1s to 1.6s due to the reprioritization of RTP candidate pairs. The *SIP and RTP mux* optimization reduces the delay further to 19.3s by removing the need to run ICE checks for RTP.

The *SIP via Overlay (SvO)* optimization, which eliminates the ICE check delay for SIP, the candidate gathering delay for SIP, and the delay associated with exchanging the candidates via the overlay, results in an even greater gain than the *SIP and RTP mux* optimization, reducing the delay to 17.8s.

The remaining four optimizations attempt to bring the delay of the *SvO* optimization further down. The *Rc=6* optimization, which reduces *Rc* (STUN retransmission count) to 6 and lowers the hard deadline of the ICE stopping criterion to 7s brings the delay down to 15.0s. The *Rc=5* optimization, which lowers Rc by two and sets the hard deadline to 5s, reduces the delay to 13.0s. Reducing the maximum number of connectivity checks (*N*) by 50% and setting the hard deadline to 4s in the *SvO Rc=5 N=50* scenario squeezes the delay to 12.0s. Finally, the compression of SIP messages in the *SvO SigComp* scenario brings the total session setup delay down to 11.2s.

B. EIMF NATs

The detailed results for the *EIMF NATs* scenario are shown in Figure 6. From the figure, we can see that the best-performing scenario, *SvO SigComp*, has a 41% lower delay

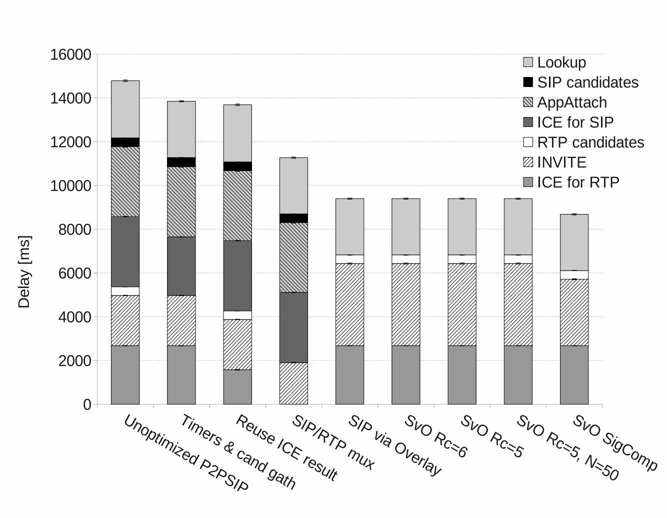

Fig. 6. Session establishment delay, EIMF NATs

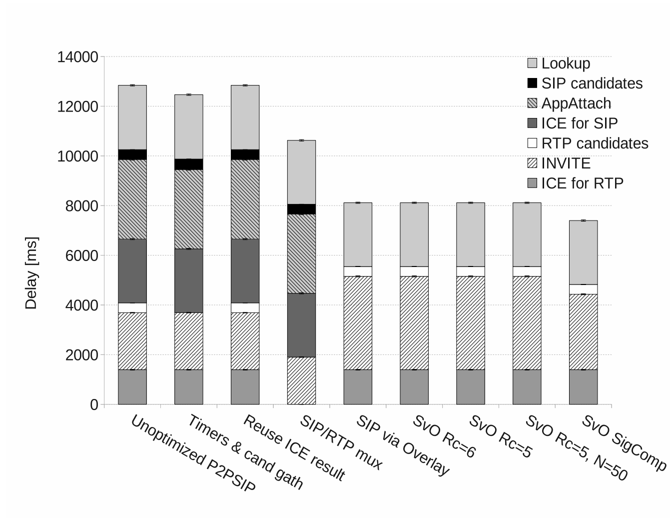

Fig. 7. Session establishment delay, no NATs

than unoptimized P2PSIP. Although this represents a considerable improvement, the gain is lower than in the *APDMF NATs* scenario. This is because in the *EIMF NATs* scenario, the ICE check delays are no longer the dominant component of the session setup delay. Thus, removal of one of the ICE negotiations in the *SIP and RTP mux* and *SvO* scenarios does not have as high an impact on the total session setup delay as in the *APDMF NATs* scenario.

We can observe from Figure 6 that modifying the *Rc*, *N*, and hard deadline values has no impact on the delays of the different *SvO* optimizations. This is due to the fact that since the NATs are not of the most restrictive type, ICE manages to always find a working, relay-free path before the hard deadline is reached. Thus, optimizations reducing the hard deadline do not have an impact.

Further, for the *Reuse ICE result* scenario, we can see that the performance gain is not as high as in the *APDMF NATs* scenario. This is due to the fact that there is less to optimize since the starting point is that the ICE delay for RTP is already much lower than in the *APDMF NATs* scenario. Thus, prioritizing the RTP candidates based on the result of the ICE negotiation for SIP can only reduce the RTP ICE negotiation delay by 1.1s.

C. No NATs

The results for the last scenario, *No NATs*, are shown in Figure 7. We can observe that the *SvO SigComp* optimization removes 42% of the delay compared to unmodified P2PSIP. Similar to the *EIMF NATs* scenario, the reduction comes from the elimination of the *ICE for SIP*, *SIP candidates*, and *AppAttach* components, and from the SigComp compression of SIP messages. The *Reuse ICE result* optimization has no impact in the *No NATs* scenario. This is because reusing the result of the SIP ICE negotiation in the RTP ICE negotiation (i.e., modifying the priorities) does not help; the highest priority candidate pair (host candidates) will be selected anyway regardless of the modification. For the *SvO* scenarios, modification of the *Rc*, *N*, and hard deadline values has no impact. The reason is the same as for the EIMF NATs scenario.

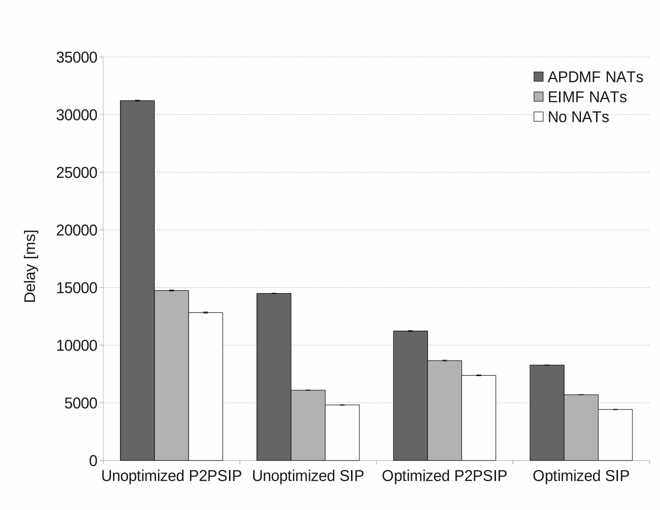

Fig. 8. Comparison of SIP and P2PSIP session establishment delays

D. Comparison to Client/Server SIP

We also ran simulations to measure session setup delays of client/server SIP to be able to compare them against the delays that our P2PSIP optimizations can achieve. Figure 8 shows the session setup delays of unoptimized P2PSIP, unoptimized client/server SIP, optimized P2PSIP, and optimized client/server SIP for our three NAT scenarios. Optimized P2PSIP refers to the *SvO SigComp* configuration. Some of the optimizations in the *SvO SigComp* configuration are also applicable to client/server SIP, including the use of SigComp and modification of the *Rc* and *N* ICE parameters. These optimizations have been applied to client/server SIP in the

Optimized SIP configuration.

From Figure 8, we can see that, perhaps surprisingly, optimized P2PSIP outperforms unoptimized client/server SIP in the *APDMF NATs* scenario. Further, our optimizations bring P2PSIP session setup delays closer to the client/server SIP delays also in the *EIMF NATs* and *No NATs* scenarios.

When comparing the session setup delays of optimized P2PSIP and optimized client/server SIP, we can see that our optimizations reduce considerably the difference between P2PSIP and client/server SIP; when comparing the unoptimized session setup flows, P2PSIP delays are always 8.0-16.7s higher (depending on the NAT scenario) than those of client/server SIP. However, when using the optimized session setup flows, the P2PSIP delays are only roughly 3.0s higher regardless of the NAT scenario.

VI. Conclusions

In this paper, we proposed and analyzed several optimizations that reduce the P2PSIP session setup delay. Our results indicate that these optimizations are very effective; they reduce the delay by up to 64% when a relay server is needed between the communicating hosts, by up to 41% when ICE can find a direct path despite of NATs, and by up to 42% when the communicating hosts are publically reachable.

It is interesting to compare our results against ITU-T recommendations for call setup delays. ITU E.721 [6], recommends an average delay of 8.0s for international calls and sets the 95th percentile at 11.0s. The starting point for this paper was that unoptimized P2PSIP does not come even close to the recommendation regardless of the NAT scenario; the average delay is 60% (4.8s) larger than the recommendation even in the best case and 290% (23.2s) larger in the worst case. When using our optimizations, the average session setup delay is clearly lower than the recommendation when the communicating nodes are publically reachable. When ICE can find a direct path across NATs, the average delay is also in line with the recommendation. When a relay server is needed, our optimizations reduce the delay considerably, by 20s, to a level that is much closer to the recommendation. Thus, we can conclude that our optimizations result in a considerable performance improvement and bring P2PSIP session setup delays down to a level that is acceptable to users, even when compared to the strict ITU-T recommendations.

Our results indicate that ICE and STUN parameters can have a significant impact on session setup delays. Therefore, future work could study the impact of using a wider range of values for these parameters to see if the delays could be reduced even further. Finally, it should be noted that the optimizations presented in this paper are applicable not only to P2PSIP, but also to other usages of RELOAD. One example is the Constrained Application Protocol (CoAP) usage for RELOAD [15]. Thus, another topic for future research is analyzing the performance of our optimizations in the context of other RELOAD usages.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, IETF, 2002.

[2] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol," IETF, Internet Draft – work in progress, July 2012.

[3] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

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

[5] J. Mäenpää, V. Andersson, A. Keränen, and G. Camarillo, "Impact of Network Address Translator Traversal on Delays in Peer-to-Peer Session Initiation Traversal," in *Proc. of IEEE GLOBECOM*, Miami, USA, December 2010.

[6] ITU-T, "Network Grade of Service Parameters and Target Values for Circuit-Switched Services in the Evolving ISDN," International Telecommunication Union, Geneva, Recommendation E.721, May 1999.

[7] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389, IETF, 2009.

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

[9] G. Camarillo, J. Maenpaa, A. Keranen, and V. Andersson, "Reducing delays related to NAT traversal in P2PSIP session establishments," in *IEEE Consumer Communications and Networking Conference (CCNC)*, Jan. 2011, pp. 549–553.

[10] G. Camarillo, P. Nikander, J. Hautakorpi, A. Keranen, and A. Johnston, "HIP BONE: Host Identity Protocol (HIP) Based Overlay Networking Environment (BONE)," RFC 6079, IETF, 2011.

[11] M. Baldi, L. De Marco, F. Risso, and L. Torrero, "Providing end-to-end connectivity to sip user agents behind nats," in *Communications, 2008. ICC '08. IEEE International Conference on*, may 2008, pp. 5902 –5908.

[12] J. Mäenpää and G. Camarillo, "Study on Maintenance Operations in a Peer-to-Peer Session Initiation Protocol Overlay Network," in *Proc. of IEEE IPDPS*, 2009.

[13] J. Mäenpää and G. Camarillo, "Analysis of Delays in a Peer-to-Peer Session Initiation Protocol Overlay Network," in *Proc. of IEEE CCNC*, Las Vegas, USA, January 2010.

[14] ——, "Estimating Operating Conditions in a Session Initiation Protocol Overlay Network," in *Proc. of IEEE IPDPS*, Atlanta, USA, April 2010.

[15] J. Mäenpää, J. Jimenez, and S. Loreto, "Using RELOAD and CoAP for Wide Area Sensor and Actuator Networking," EURASIP Journal on Wireless Communications and Networking, March 2012.

[16] J. Mäenpää, "Performance Evaluation of Recursive Distributed Rendezvous based Service Discovery for Peer-to-Peer Session Initiation Protocol," Elsevier Journal on Computer Networks, June 2011.

[17] B. Athwal, F. C. Harmatzis, and V. P. Tanguturi, "Replacing centric voice services with hosted voip services: an application of real options approach," in *Proc. of the 16th international telecommunications society (ITS) European regional conference*, 2006.

[18] "Traffic analysis for voice over ip," 2001. [Online]. Available: http://www.cisco.com

[19] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Observations on the dynamic evolution of peer-to-peer networks," in *Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS '01)*. London, UK: Springer-Verlag, 2002, pp. 22–33.

[20] Microsoft, "Interactive Connectivity Establishment (ICE) Extensions," Microsoft Corporation, Technical specification v20100218, Feb 2010.

[21] R. Price, C. Bormann, J. Christoffersson, H. Hannu, Z. Liu, and J. Rosenberg, "Signaling Compression (SigComp)," RFC 3320, IETF, 2003.