Optimization of Signaling Traffic in Centralized Conferences using SIP

IGOR MILADINOVIC[1,2], JOHANNES STADLER[1,2]
[1)]Institute of Communication Networks
Vienna University of Technology
Favoritenstr. 9/388, A-1040, Vienna
AUSTRIA
[2)]Forschungszentrum Telekommunikation Wien
Donau-City-Str. 1, A-1220, Vienna
AUSTRIA

Abstract: - Multiparty conferencing is becoming an important topic in the next generation networks. An essential part of multiparty conferencing is the signaling protocol, which has to establish, modify, update and terminate conferencing sessions. This paper focuses on signaling for closed centralized conferencing using Session Initiation Protocol (SIP) and proposes an approach that achieves an optimization of signaling traffic on the conference server. The presented approach also improves fault tolerance of centralized conferences by preventing the conference server from being a single point of failure. The results obtained with this optimization are also presented.

Key-Words: - multipoint conferencing, signaling, SIP, conference server, traffic optimization, fault tolerance

1 Introduction

Multiparty conferencing is becoming an important topic in the next generation networks. An indispensable part of the multiparty conferencing is the signaling protocol, which has to establish, modify, update and terminate conferencing sessions. Session Initiation Protocol (SIP) [1][2] is a relatively new signaling protocol that is already widely used in IP-networks. SIP will also be used by 3GPP for the signaling in Universal Mobile Telecommunications System (UMTS) networks (up release 5) [3]. Therefore, this paper deals with multiparty conferences that use SIP as signaling protocol.

A conference can be either open or closed. Closed conferences require that the identity of a participant is known by other participants. These conferences produce more signaling traffic than open conferences, where participants are anonymous.

In general, there are two types of conference signaling, distributed or centralized. Distributed signaling requires end-terminals to maintain the state of the conference. This results in more complexity in end-terminals. SIP provides two approaches for distributed signaling, full-mesh and end system mixing. Full-mesh signaling requires a signaling connection between each pair of participants and hence produces very much signaling traffic [4] that is rapidly increasing with increasing number of conference participants. End system mixing conferences, described in [5], requires that some end-terminals in a conference are capable of mixing signaling and media data of two or more participants. When such an end-terminal leaves the conference, the connection between participants that were connected over this end-terminal also terminates. Another drawback is the possibility of loops [5]. Because of these reasons, distributed conferences in SIP do not scale and they are only suitable for very small conferences.

Centralized signaling uses an additional network element, called conference server, which maintains the state of the conference. Each conference participant obtains a point-to-point signaling connection with the conference server. The drawback of centralized conferences is that the conference server represents a single point of failure. Besides that, it could cause scalability problems in closed conferences, because the conference server has to notify each participant when a conference modification occurs (for example when a new user joins in or leaves the conference). Centralized signaling can be used for small and middle size conferences.

In this paper we propose a solution for both drawbacks of centralized signaling. The basic idea is to define a dedicated conference participant, called conference chair that possesses the state of the conference. In the case that the conference server crashed, the conference chair would choose another conference server and reinitiate the conference as soon as possible. The conference would be shortly interrupted and then continued. By this way, the

conference server is not a single point of failure any more. Furthermore, the conference chair is able to announce conference modifications to other participants . This reduces the signaling traffic on the conference server, because the conference server notifies only the conference chair after a conference modification. We also treat the case when the conference chair leaves the conference or crashes. The main focus of the paper is on closed conferences, because the signaling traffic is more critical for these conferences (an open conference can be seen as a special case of a closed conference, where none of the participants wants to know identities of the others).

In the next section we will give an introduction to the related work that is necessary for understanding of this paper. Section 3 explains the functionality of our approach and also treats special issues like changing of conference chair. A comparison of this approach with the conferencing without introduced optimization is given in section 4. We have used a prototype implementation to measure the signaling traffic with and without optimization. Section 5 completes the paper with a conclusion.

2 Related Work

This section gives a brief overview to SIP describing the basics of SIP messages and call-setup. Special accent stands on the centralized signaling for closed conferences.

2.1 SIP

Session Initiation Protocol (SIP) [1][2] is an application layer protocol originally developed by the Multiparty Multimedia Session Control (MMUSIC) working group of the IETF in 1999. Meanwhile, a SIP working group has been formed that continues the development of this protocol.

The purpose of SIP is establishment, modification and termination of all types of sessions. It is a text-based protocol, very similar to other text-based Internet protocols, like Hypertext Transfer Protocol (HTTP) [6] or Simple Mail Transfer Protocol (SMTP) [7]. A SIP session is usually described by the Session Description Protocol (SDP) [8], which is carried in the body of a SIP message. For transport of real-time media data (voice and video) in a session the Real-Time Transport Protocol (RTP) [9] is used.

SIP is independent of the transport protocol, so that SIP messages can be sent over User Datagram Protocol (UDP) [10], Transmission Control Protocol (TCP) [11] or Stream Control Transmission Protocol (SCTP) [12]. SIP provides a reliability mechanism when an unreliable protocol (UDP) is used.

There are two types of messages in SIP, requests and responses. The SIP standard defines six different requests, which differentiate in their methods: INVITE, ACK, BYE, CANCEL, OPTIONS and REGISTER. Each of these requests (except ACK) must be replied with a response that can be generally divided in provisional and final responses. A provisional response has only informational character and must be always followed by a final response.

There are two types of entities in SIP, User Agents (UA) and network servers. A SIP UA could be seen as end device and acts either as user terminal or as automated connection endpoint, for instance a SIP/PSTN gateway. Network servers are used for registration, call routing and they can also be enabled to perform different kinds of applications. They are divided into proxy servers, redirect servers and registers.

Addressing in SIP is very similar to e-mail addressing. Each user obtains a SIP address that is global and unique. This address is also called address-of-record. With this address a user can be contacted independent of the device that this user currently uses under the precondition that the user is registered.

A call-setup consists of three steps. In the first step, the caller sends an INVITE request to the callee that replies this request either with an OK response or with an error response (e.g. DECLINE response if the callee declines the call). In the last step an ACK request is sent from caller to callee to confirm the call-setup. These three steps are called three-way handshake. To tear down the call, either caller or callee sends a BYE request, which is replied with an OK response.

2.2 Centralized Conferencing

SIP uses an additional network element for centralized conferencing, which is called Conference Server (CS). There are two types of CS, dial-in or dial-out. In both cases, each participant in a conference establishes a point-to-point signaling connection with the CS. In the case of a dial-in CS this connection is initiated by participant's UA, otherwise by the CS itself.

A conference is identified by the request URI that is distributed either by SIP REFER message [13] or by another protocol (e.g. HTTP, SMTP) to each user that is expected to participate in the conference.

Unfortunately, as described in [5], centralized conferences use RTP and RTCP for distribution of participant identities and therefore do not fulfill the

requirements of a closed conference, where the identity of each participant is know by others before starting media transmission. A SIP extension [4][14] solves this problem by introduction of a new SIP method called CONF. Each time when a change in the conference state occurs (e.g. a participant leaves the conference), the CS notifies each participant by sending a CONF request. This request contains the actual state of the conference participants. Fig.1 shows signaling traffic that is generated when a new user (D) participates in an existing conference with three participants (A, B, C). Note that this example shows a dial-in CS (represented by the server icon in fig.1), but the amount of the signaling traffic remains the same also for a conference with a dial-out CS. The only difference is that the signaling traffic between D and CS flows in reverse direction.

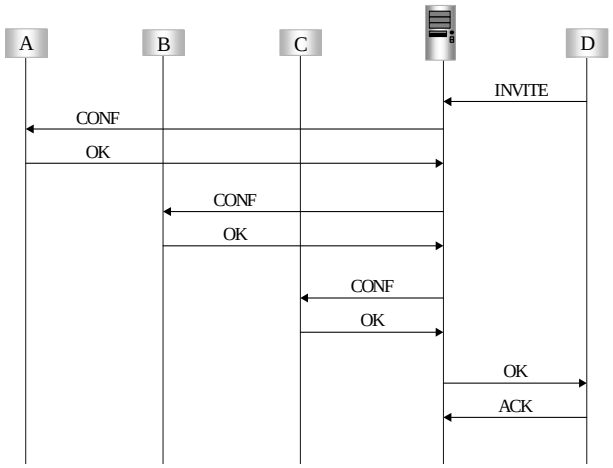

Fig.1: Joining a new participant in the conference

Fig.1 shows clearly that a modification of the conference (joining a new participant) causes relatively high signaling traffic on the CS. This traffic is much higher for a conference with more participants.

3 Signaling Traffic Optimization

The SIP extension proposed in [14] can be improved for closed centralized conference in two points:

- Signaling traffic on the CS should be reduced. Usually a CS manages a large number of conferences at once so that a reduction of the signaling traffic of each single conference would significantly improve scaling.
- The CS remains a single point of failure. Although each participant possesses the addresses of others, it is not clear which participant should reinitialize the conference using another CS. As a result more than one participant could try to reinitialize the conference which would cause confusion.

The basic idea of our proposal is the introduction of a dedicated participant called conference chair. This requires a new status value of the participant header field described in [14]. Currently, following status values are defined: active, invited or joining. We propose the status value “chair” to be added to the participant header. It must be ensured that exactly one participant obtains the chair status at once, what can be easily done by a CS.

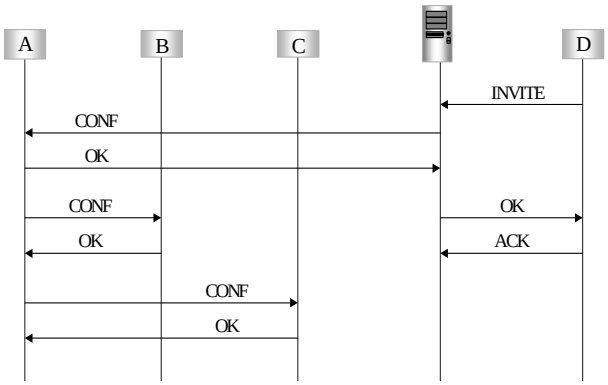

Fig.2: Joining a new participant with optimization

When the conference state changes, the CS notifies only the conference chair and not each single participant separately. It is the responsibility of the chair to distribute this information to other participants. Fig.2 shows the same example as fig.1, but with optimization of signaling traffic through the CS. The order of sending CONF requests by the conference chair (A) is not important.

Besides the optimization of the signaling traffic at the CS, there is another advantage of defining the conference chair. The CS does not represent a single point of failure any more, because the conference chair would reinitialize the conference when the CS crashes. This is possible because the conference chair knows the SIP address of each participant. In that case other participants would be contacted either by the conference chair (with the REFER method) in the case of a dial-in CS, or by a new CS (with the INVITE method) in the case of a dial-out CS. On the other side, if the UA of the conference chair crashed, the CS would recognize that by missing responses from the conference chair. In that case, the CS would choose another chair.

3.1 Changing of the conference chair

At the begin of a conference, the conference chair is automatically the initiator of the conference. In some cases it is useful to change the conference chair, e.g. when the chair wants to leave the conference without

terminating the whole conference. In that case, the chair is able to choose a new chair by sending a CONF request to the CS. This request contains the list of participants where the chair status is given to another participant. The CS sends a CONF request to the new chair that replies with an OK response. This message flow is depictured in the upper part of fig.3 (marked with I). If the previous chair does not specify the new chair, the CS will choose a new chair randomly between participants that were not the chair of this conference so far.

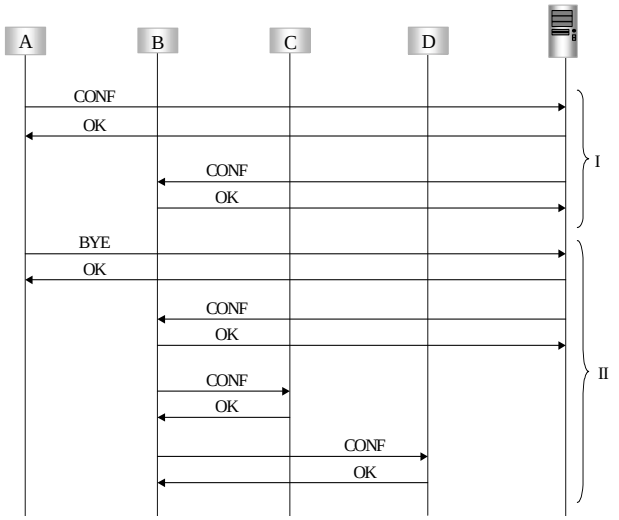

Fig.3: Chair changing and leaving the conference

A participant that was the chair of a conference cannot be chosen to be the chair of the same conference by the CS. However, this participant can be chosen to be the chair of the conference again by the current chair. The reason for that is that a verbal agreement between participants is assumed.

3.2 Terminating and leaving the conference

The conference chair is able to terminate the whole conference at any time by sending a BYE request to the CS. After arriving of this request, the CS sends a BYE request to each conference participant and the conference is terminated.

A participant that is not the chair can leave the conference by sending a BYE request to the CS. In this case the conference will not be terminated, but only this participant will be removed from the participant list. Of course, the CS will notify the chair about this change and the chair will notify other participants as shown in the bottom part of fig. 3 (marked with II).

If the conference chair only wants to leave the conference and not to terminate it, a BYE request must not be sent. The chair must firstly require the change of the chair (as described in 3.1) and then leave the conference as usual. The message flow of this scenario is shown in fig.3 (both parts).

4 Results of the optimization

In order to prove the proposed optimization, we have implemented a prototype in C++ that supports centralized conferences with the conference chair. Using this prototype we measured the number of SIP messages on the CS. The CS is implemented as a dial-in server, so that the message flow is equivalent to the flow in examples above (fig.1-3).

In our implementation each SIP message is sent without retransmission (a message is retransmitted in SIP in order to ensure reliability if a message is sent over unreliable UDP). This assumption is always fulfilled when TCP is used or when UDP is used and each messaged is responded within the estimated round trip time. Even in the case that a message is retransmitted, the obtained results show a valid comparison because the number of messages is increased in each compared case approximate.

We will present results of two different scenarios. In the first scenario the number of conference participants is growing from three to fifteen without changing the conference chair. We call this scenario conference initialization, because the same traffic is produced when a conference with fifteen participants is being initialized.

Of course, each change of the conference chair causes additional signaling traffic that would not be produced without the proposed optimization. Therefore, for a fair comparison a scenario is necessary where at least one chair change occurs. Our second scenario shows the signaling traffic of a conference that originally consists of four participants. In the next step a new participant joins in the conference. Afterwards, the chair decides to leave the conference. Firstly, the chair of the conference is changed and secondly, the conference has only four participants left. Thereafter another change of the chair occurs and finally two participants join in the conference. At the end, the conference consists of six participants. This scenario contains six steps, two of which are changes of the conference chair. We call this scenario double chair change.

We have also simulated a crash of both, the CS and the conference chair. In the first case the conference chair missed the RTP and RTCP data from the CS (because the CS also acts as a RTP mixer) and reinitialized the conference using another CS. This results in an interruption of the conference for a few seconds. In the second case, the CS noticed the crash of the conference chair after a conference change had occurred, because of missing response to

the CONF request. After a timeout, a new chair is chosen by the CS.

Note that in the worst-case the CS can crash after a crash of the terminal of the conference chair, but before a conference change occurs and a new chair is chosen. In that case the conference is irrecoverably lost.

4.1 Conference initialization

In this scenario we compare the traffic on the CS with and without the proposed optimization. For the sake of completeness, we also show the signaling traffic that is generated in an open conference where CONF requests are not sent at all and therefore the identities of conferences participants are not known by others as described in section 2.2. This case represents a minimum of the signaling traffic.

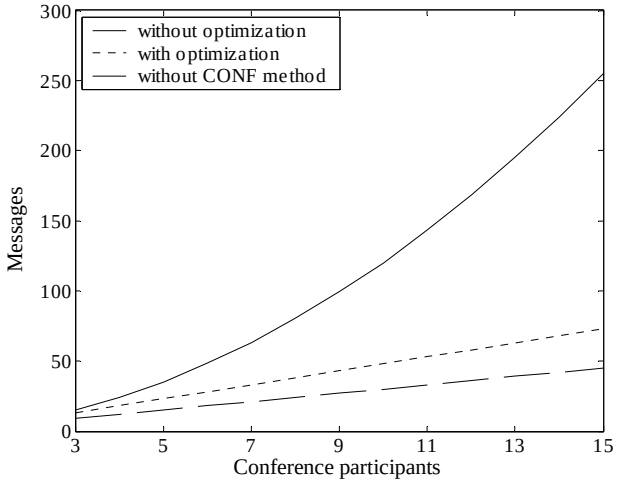

Fig.4: Signaling traffic of conference initialization scenario

Fig.4 shows that the signaling traffic on the CS is significantly reduced especially for conferences with many participants. The traffic is growing quadratically with the number of participants without optimization and nearly linearly with the optimization. Furthermore, the optimized traffic is only marginally higher than the traffic in an open conference, which is also linearly increasing with the number of participants.

An initialization of a conference with ten participants, for example, requires 120 messages to be managed by the CS without optimization. Using the optimization, there are only 48 messages left, what represents a reduction of 60%. The minimum of messages that is produces in an open conference amounts to 30 messages. The difference is even much bigger for conferences with more participants. Therefore this optimization improves scaling of centralized closed conferences essentially.

4.2 Double chair change

As we have seen, without changing the chair the proposed optimization obtains excellent results. Unfortunately, the critical point of this approach is a change of the chair, because additional signaling traffic is generated, which would not be necessary without the optimization. It is also theoretically possible that the "optimized" traffic is even higher than the traffic without optimization, if a chair change occurs very often. In this scenario we want to investigate signaling traffic when the chair of the conference changes relatively often.

For this scenario we have chosen an unfavorable case with two chair changes in six steps. Statistically speaking, it means that every third change in a conference signifies a chair change. Moreover, this scenario treats a small conference (minimal four, maximal six participants), although the optimization obtains better results for more participants (fig.4).

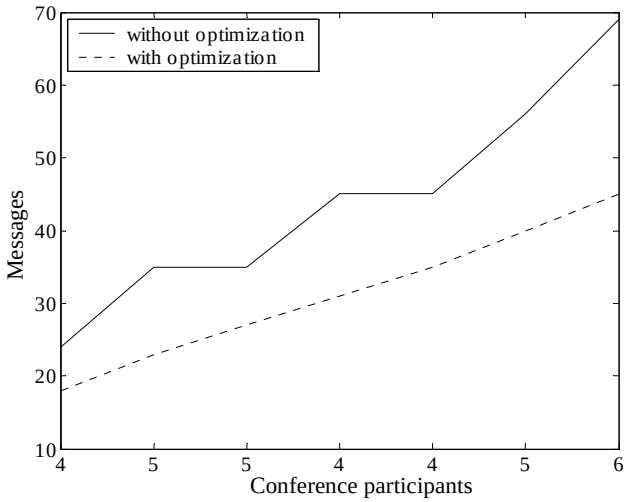

Fig.5: Signaling traffic of double chair change scenario

The results of this scenario are given in fig.5. Even in this case, the optimized traffic is always lower than the traffic without optimization. Without optimization there is no increasing of signaling traffic when the chair changes, what can be clearly seen in fig.5 when the number of participants remains the same between two steps. As expected, the difference between optimized and non-optimized traffic is not as big as in the first case, not only because of chair changes, but also because of the small number of participants (compare with the difference for six participants in fig.4). At the end of this scenario there are 69 messages that must be managed by the CS without optimization. The proposed approach requires the CS to manage 45 messages, what results in a reduction of about 35% of the signaling traffic on the CS.

5 Conclusion

This paper presents an approach for optimization of the signaling traffic on a conference server. The basic idea is the introduction of a special participant that is called conference chair. This participant communicates with the conference server and notifies other participants if a conference change occurs. Moreover, it is the responsibility of this participant to reinitialize the conference if the conference server crashes. By this way, the conference server does not represent a single point of failure any more.

The results measured with our implementation show that the proposed optimization improves scalability of closed centralized conferences essentially. Especially if the number of participants is relatively big (over ten participants), closed conferences without optimization scale very badly, because of quadratically increasing of the signaling traffic with the number of participants. The optimization proposed in this paper achieves a nearly linear dependency between signaling traffic and the number of participants.

References:

[1] Schulzrinne, H. and Rosenberg, J., *The Session Initiation Protocol: Internet-centric signaling*, IEEE Communications Magazine, Volume: 38 Issue: 10, 2000, pp. 134 –141.

[2] Handley, M., Schulzrinne, H., Schooler, E. and Rosenberg, J., *SIP: Session Initiation Protocol*, IETF RFC 2543, 1999.

[3] Richardson, K.W., *UMTS overview*, Electronics & Communication Engineering Journal, Volume: 12 Issue: 3, 2000, pp. 93 –100.

[4] Miladinovic, I. and Stalder, J., *Multiparty Conference Signalling using the Session Initiation Protocol (SIP)*, Conference Proceedings, INC 2002, pp. 191-198.

[5] Rosenberg, J. and Schulzrinne, H., *Models for Multi Party Conferencing in SIP*, IETF Internet Draft, 2001, work in progress.

[6] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T., *Hypertext Transfer Protocol - HTTP/1.1*, IETF RFC 2616, 1999.

[7] Klensin, J., *Simple Mail Transfer Protocol*, IETF RFC 2821, 2001.

[8] Handley, M. and Jacobson, V., *SDP: Session Description Protocol*, IETF RFC 2327, 1998.

[9] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V., *RTP: A Transport Protocol for Real-Time Applications*, IETF RFC 1889, 1996.

[10] Postel, J., *User Datagram Protocol*, IETF RFC 768, 1980.

[11] Postel, J., *Transmission Control Protocol*, IETF RFC 793, 1981.

[12] Stewart, R. and Metz, C., *SCTP: new transport protocol for TCP/IP*, IEEE Internet Computing, Volume: 5, Issue: 6, 2001, pp. 64 –69.

[13] Sparks, R., *The Refer Method*, IETF Internet Draft, 2001, work in progress.

[14] Miladinovic, I. and Stadler, J., *SIP Extension for Multiparty Conferencing*, IETF Internet Draft, 2002, work in progress.