Scalable Floor Control in Conferencing Environments: The RBone Approach

Dirk Trossen
Nokia Research
5 Wayside Road, Burlington, MA 01803

Abstract- **Most features of conferencing applications are mostly independent from specific scenarios. Thus, it is useful to provide a generic service to accelerate and simplify the development of such applications. Scalability in terms of group size and distribution of conference members is a big issue in the design of such services especially when applying conferencing in large scaled Internet scenarios. Beside functionality for conference management and multipoint communication, floor control is a crucial issue for the provision of application state synchronization and controlled access to application resources. We present an approach to provide a generic floor control service even suited for large scaled environments. The proposal uses efficient multicast on local level combined with a tree-based routing on global level by introducing the Resource Backbone (RBone) approach, which promises to improve the responsiveness of the provided floor control. The service as well as the protocol mechanisms required for implementation are presented in this paper.**

Index terms - **floor control, resource backbone**

A. INTRODUCTION

Interactive collaborative scenarios like remote meetings, virtual classrooms, or sharing applications via the Internet have become more and more popular in the past ten years. From an application's point of view, the need for *tight control*, such as for synchronization of actions or controlled access to application resources, arises for these applications in contrast to loosely coupled scenarios like plain video streaming. Hence, coordination and synchronization means are required due to concurrent activities in these scenarios. Thus, it is crucial for these scenarios to provide means for the implementation of *floor control* [13], e.g., to map the real-life's *social protocol* [13] onto the distributed environment.

A crucial issue in the development of a floor control service is its *scalability* with respect to the responsiveness of the provided functionality depending on the number of participating users and their geographical distribution. Recently proposed conferencing toolkits and standards suffer especially from this key issue. Either unicast-based topologies, often simple stars, are used routing the requests to a centralized floor control manager or a multicast-based exchange of floor holder information is applied often leading to large response time of each request due to the overhead to ensure reliability of the exchanged floor information.

In this paper, a scalable floor control approach is presented combining both mechanisms. It starts with a presentation of the provided services. The main part of the paper is dealing with outlining the proposed protocol mechanisms in detail.

The basic idea of the proposal is to combine the usage of multicast and unicast by applying efficient multicast on local level and apply a tree-based unicast routing on global level. For that, similar to the MBone approach, multicast-capable local islands are interconnected using unicast *tunnels*. Within this tunnel topology, named as the *resource backbone* (RBone) throughout the paper, a floor control specific routing is used for optimization. Hence, an optimized interconnection of the distributed entities is achieved.

Thus, this approach is expected to improve the responsiveness of the provided service even in conferences of larger scale. This is especially true when considering scenarios in which the group of conference members is comprised of a few subgroups of participants each located in a fast local area network as for instance in internal corporate meetings among geographically distributed developer groups. However, a performance evaluation or measurements of real-life implementations are is not shown in this paper.

The remainder of the paper is organized as follows. Section B gives an overview of related work in the area of group communication toolkits and protocols. Section C outlines the provided floor control services, while in Section D the RBone approach is introduced for realizing the services. Finally, Section E concludes the paper, and Section F gives an outlook for future work.

B. RELATED WORK

Since the importance of group communication has been increased significantly during the past ten years, there are several conferencing environments being proposed for the implementation of collaborative applications.

While environments proposed in [1][2] focus on specific aspects of conferencing functionality, more generic platforms like the Scalable Conferencing Control Service (SCCS [17]) or standard-based solutions of the ITU (*International Telecommunications Union*) or the IETF (*Internet Engineering Task Force*) aim to provide a wide spectrum of services for the creation of conferencing applications.

For that, the ITU specifies a set of standards (T.120 [11]) providing multipoint transfer, conference management, and floor control functionality for data applications. A tree-based approach of interconnected service providers is used to which applications are attached. Due to the centralized approach used for the floor control functionality, this approach leads to a bad scalability in terms of responsiveness as shown in [10].

In [17], the scalable conferencing control service (SCCS) is proposed with an ITU-compliant service model using more sophisticated protocol mechanisms to improve the scalability of the environment even in large scaled scenarios. For that, a resource management scheme is introduced leading to a higher responsiveness of the system when locally handling requests in the tree of providers. However, the proposed mechanism does not use underlying multicast facilities which can be seen as the major drawback similar to the centralized version of the ITU.

e-mail: dirk.trossen@nokia.com phone: +1 (781) 993 3605

The proposed *Internet Multimedia Conferencing Architecture* of the IETF [7] (see Figure 1) outlines the components and protocols to be used for realization of conferencing scenarios in the Internet.

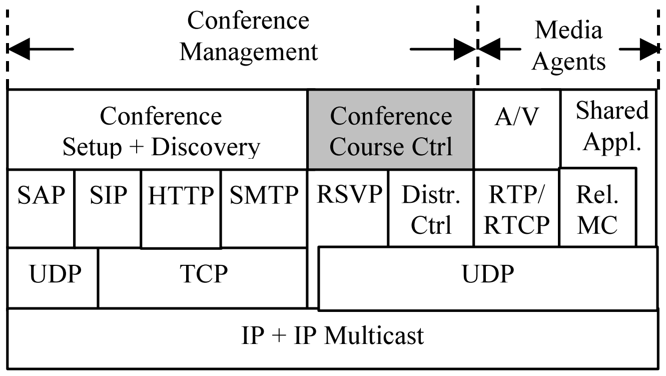

Figure 1: Internet Multimedia Conferencing Architecture

For the realization of the *conference course control* component for the control of tightly coupled conferences services like the ITU T.120 protocol stack or SCCS might be used. For the support of loosely coupled conferences (or *light-weight sessions* [6]) following the ALF (*application level framing* [6]) approach, conference control is realized by loosely interconnected participants using multicast-based information exchange with a lack of centralized control of membership and floor holder information. However, these approaches have to deal with large response times in large scaled conferences due to the distributed handling of the request, e.g., using quorum-based approaches [15].

An approach to provide floor control in the Internet as an extension to existing MBone tools is proposed by Malpani and Rowe in [14] using a centralized architecture. Since the approach only maintains a speakers list in a conference, the large response time of the system is not a critical issue. However, this lack of efficiency is not acceptable for generic floor control services.

Dommel [3] proposes group coordination in a larger scope including floor control for application state synchronization. The proposal uses a shared tree for multicast delivery, sub-group support, and floor control message routing. This approach extends the usual IP multicast routing by proposing a sub-group addressing. However, a single floor controller approach is used for the floor control protocol which applies a centralized approach for which the shared tree routing is used. Furthermore, the multicast tree routing has to be extended using the proposed sub-group addressing.

Hence, it can be summarized that the related work either implement tree- or simple star-based approaches by interconnecting conference members, or use multicast-based scheme by allowing temporary inconsistencies, or do not implement floor control at all. In the following two sections, a floor control protocol is presented using a combination of local multicast and global unicast.

C. Provided Services

Compared to the approaches presented in the related work section, the remainder of the paper will focus on the provision of floor control services. Hence, other conference course control functionality (see Figure 1), such as membership control, is not within the scope of the paper.

As proposed in [17], a floor control service should provide facilities to support application state synchronization and controlled access to application resources. Hence, the service shall enable to map social protocols, i.e., the rules to access application objects like audiovisual streams, onto distributed systems. The list of possible scenarios includes conducted meetings or even more complicated mediated conferences, but also access control on resources as for shared applications. However, the mapping of floors onto application semantics is not within the scope of the proposed service.

Each floor is identified using a conference-unique name. The naming pattern is not within the scope of the service. However, it is recommended to use a decimal naming scheme to simplify naming conflict resolving. The following floor control services are provided:

- *grab floor*: allocates a floor for exclusive use by the requesting participant
- *inhibit floor*: allocates a floor for non-exclusive use by several participants
- *release floor*: releases an allocated floor; changes the state of the floor accordingly
- *test floor*: asks for the current state (F_FREE, F_GRABBED, F_INHIBITED) of the floor
- *ask floor*: asks the current floor holder to grant an exclusive floor to the requesting entity
- *give floor*: grants an exclusive floor to another participant
- *holders of floor*: asks for a list of current floor holders

It can be seen that the provided floor control service is very similar to the T.122 [12] of the H.323 standard. However, requesting the current floor holders is not supported by the T.122 standard. Additionally, appropriate repairing mechanisms are not provided to recover from node or network failures.

D. Service Realization: The RBone Approach

In the following sections, the protocol environment of the service is presented together with a description of maintenance functions for the topology. Furthermore, it is outlined in detail how to handle the floor control service requests in this environment.

1st. Protocol Environment

Figure 2 shows the protocol environment in terms of the used topology to realize the proposed services. It can be seen that the topology is comprised of the conference participants which must join the *conference management group* (CMG). This group can also be used for other conference management service, e.g., to exchange membership information.

Furthermore, selected participants, the *RB providers*, are interconnected within a tree topology, the *Resource Backbone* (RBone or RB), with a dedicated top node. Each RB provider is responsible for handling floor control requests within its own local *floor control island* (FCI). This FCI is a local multicast group containing all joined conference participants within local scope.

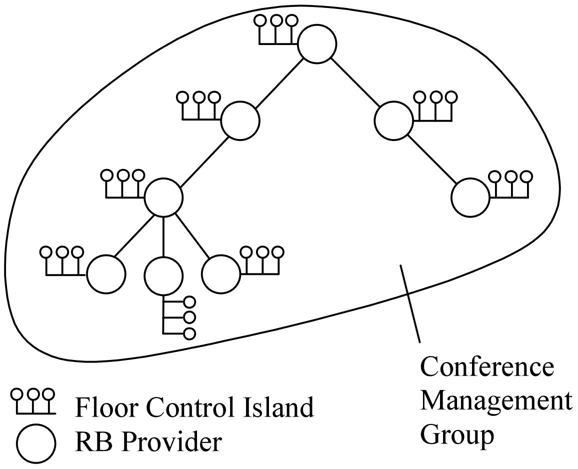

Figure 2: Protocol Environment

Hence, the RBone tree topology connects multicast-capable islands using unicast *tunnels* similar to the MBone approach [5]. However, a floor control specific routing is used within the unicast topology.

It is not within the scope of the approach how to define the different multicast group addresses. However, means like the *session description protocol* (SDP [9]) together with the *session announcement protocol* (SAP [8]) might be used to distribute the appropriate information.

It will be outlined in the following sections how the proposed scheme applies multicast-based floor control on local level and tree-based routing on global level. But first, the used transfer mechanisms of underlying transport layers are depicted.

2nd. Encoding and Transfer of Messages

In the following protocol description of the floor control service, a simple message transfer is applied. For that, it is assumed to use an underlying multicast transport service. In addition, dedicated participants, namely the RB providers, are interconnected using an underlying unicast transport service. Both services are assumed to provide reliable, consistent delivery of data units called *messages*. The encoding of these messages is not within the scope of the paper. However, the presented protocol description easily enables to extract a message format for the exchanged messages.

Reliability is bounded by the fact that member end systems may find that they no longer can reliably interact with the other members, e.g., due to network partitioning. For the unicast case, a *connection failure indication* is mandatory to be delivered to the participant. Messages are *globally ordered*. Thus, each message is assigned a message number by the appropriate transport service, and messages are delivered to participants in monotonic message number order. In the rest of this document, the term *distribute* will be used to indicate that a member end system sends a message using the appropriate transport service.

3rd. Floor Context

Each FCI member maintains state information of floors valid for the local FCI. This state information includes the name and state of locally allocated floors and the name of local holders. Additionally, each FCI member maintains a list of current FCI members. If a floor is indicated as free using a FLOOR_STATUS message, the appropriate floor entry is deleted from the floor context.

The top RB provider maintains an additional global floor context containing all local floor context information. In addition, each floor entry contains the information in which branch of the tree this floor entry is valid.

4th. Joining the Floor Control Island

Each participant wishing to use the floor control services must join the multicast group representing the FCI and sends an FCI_JOIN message to the FCI. For that, it is assumed that the underlying multicast protocol supports the establishment of locally scoped groups.

After sending that message, the *FCI general* timer is set to two seconds assuming a fairly small response time due to the local character of the FCI. If the newly joined participant receives an FCI_THERE before the FCI general timer expires, the join procedure is finished and the newly joined member becomes a normal FCI member. The FCI_THERE message contains the local floor context. All FCI members store the received member information of the new participant.

If the timer expires before receiving the FCI_THERE, it is assumed that either the old RB provider failed or the newly joined member is the first FCI member. In both cases, the newly joined member becomes the new RB provider and sends an FCI_INQUIRY to the FCI. Furthermore, the *heartbeat mechanism* (see Section D.6) is started. Each member of the FCI has to respond with an FCI_REGISTER containing its presence information and the local floor context within the time interval defined by FCI general. If the old RB provider is still working, it has to release all RB connections and has to respond to the FCI_INQUIRY, if needed.

After getting the local floor context and FCI member list, the RB provider continues with the RB backbone establishment procedure (see Section D.5).

5th. Establishing and Extending the Resource Backbone

An RB provider connects to the resource backbone by sending an RB_JOIN to the CMG containing its presence information. Note that this message is not sent by the initiator of the conference since this participant does not need to connect to the resource backbone as the first member of the conference. If the RB provider is connecting to the RB for the first time, an appropriate flag of the RB_JOIN message is set for indication.

Any other RB provider responds by sending an RB_THERE message to the CMG. This response is delayed when there is another RB extension operation pending. Hence, any response is delayed until an RB_FINISH message is received at the CMG.

The RB_THERE message contains the responding RB provider's presence information. This send operation is delayed randomly to avoid message explosion. When other RB providers receive this message, they stop sending their own response because a responding RB provider was found.

All responding RB providers try to establish a unicast transport connection to the requesting RB provider. From the set of responding RB providers, only one connection request is positively confirmed while all other transport connection requests are refused. After connection establishment, the requesting RB provider sends an RB_JOIN message via the unicast connection, i.e., via the RBone topology. This message is routed upward in the tree until a) it reaches the top RB provider or b) it reaches the requesting RB provider again. Note that this message is only routed upward on connections which are valid. This means that the establishment procedure is finished for this connection. Furthermore, it can be seen that the established RB connection is an upward connection from the requesting provider's point of view.

In case a), the RB is established successfully and the completion of the procedure is signaled (see below). In case b), a loop was built. As a result, the requesting RB provider releases its connection again and becomes the new top RB provider. To collect valid floor state information, the new RB top provider sends an RB_GET_CONTEXT message downwards. This message is sent until it reaches a leaf node, which sends an RB_CONTEXT message back in upward direction. These messages are cumulated in each branching node until it reaches the new top RB provider with updated global floor context information.

The completion of the RB extension procedure is signaled by the current top RB provider sending an RB_FINISH to the CMG.

6th. Heartbeat

As an indication that a local RB provider is still alive, a heartbeat mechanism is used. For that, the current RB provider regularly sends an FCI_HEARTBEAT message to the FCI. The interval for sending the heartbeat is defined by the FCI general value. Since the FCI general interval determines the responsiveness of the system against RB provider failures, this interval is kept small to recover fast enough from an RB provider failure.

A failure of the RB provider is detected by missing FCI_HEARTBEAT messages. For this, a detection timer is used which is set to twice the value of the FCI general interval by default.

7th. Repairing the Resource Backbone

Repairing an existing resource backbone is necessary in three cases, namely when an RB connection fails, or a local RB provider quits, or a local RB provider fails. In the following, the mechanisms for all these cases are presented.

1) RB connection fails

In this case, it is assumed that both connection endpoints are still intact. Hence, both endpoints start the RB extension procedure (see Section D.5) again to find new RB endpoints.

2) Local RB Provider quits

In this case, the local RB provider quits orderly. This is done by sending an FCI_GIVE to the FCI containing the list of remaining FCI members and the presence information of any FCI member indicating the new RB provider. This member must take over the role of the new RB provider by sending an FCI_GIVEN message to the FCI. If the chosen member does not respond within FCI general seconds, a failure is assumed, the chosen member is removed from the list, and the selection process is restarted. If there is no member left on the list, the old RB provider deletes the FCI.

If the selection of a new RB provider was successful, the old RB provider must release all RB connections, and the new RB provider starts the RB extension process (see Section D.5) to find an RBone endpoint.

3) Local RB Provider fails

As indicated in Section D.6, an RB provider failure is detected by missing FCI_HEARTBEAT messages. In that case, the oldest remaining FCI member must sent an FCI_THERE message containing the current list of FCI members. If this message is not received within FCI general seconds, the next member on the list must send the message, and so forth. All members not sending the FCI_THERE message are deleted from the list. If they are still working, they are supposed to join the FCI again. If the old RB provider is still working, it has to release all RB connections, and it has to join the FCI again for usage of floor control services, if needed.

The newly selected local RB provider connect to the RB using the procedure of Section D.5.

8th. Service Request Handling

The basic rule for handling floor control service requests is that each FCI tries to respond to a request locally. If this is not possible due to missing information, the request is sent upward in the RB for further processing.

This general rule is explained in more detail for each floor control service request in this section. The mechanisms are depicted using an indented bullet notation for better illustrating the protocol functionality.

4) Grab Floor

The requesting participant sends a FLOOR_GRAB message to the FCI. The RB provider checks its floor context whether the floor is already grabbed locally.

- If yes, a FLOOR_ERROR message is sent to the FCI with error code E_GRABBED. Note that this case can be avoided by the requesting participant by checking its own floor context before sending the message.
- If not, the FLOOR_GRAB message is routed upward in the RB.

- If an RB provider is passed whose local FCI contains the current floor holder, a FLOOR_ERROR message is sent back to the originating RB provider immediately with error code E_GRABBED or E_INHIBITED, respectively.
- If the message reaches the top RB provider, the global floor context is checked.
  - If the floor status is F_FREE, the global floor context is changed, and a FLOOR_STATUS message is sent back via the RB indicating the new status F_GRABBED. The FLOOR_STATUS message is relayed to the local FCI by the originating RB provider. Each FCI member updates its floor context appropriately.
  - If the floor is allocated, a FLOOR_ERROR message is sent back to the originating RB provider with error code E_GRABBED or E_INHIBITED. This message is relayed to the FCI.

5) Inhibit Floor

The requesting participant sends a FLOOR_INHIBIT message to the FCI. The RB provider checks its floor context whether the floor is already grabbed, i.e., allocated exclusively.

- If yes, a FLOOR_ERROR message with error code E_GRABBED is sent to the FCI indicating the erroneous message.
- If not, the RB provider checks whether the floor is inhibited locally.
  - If yes, the RB provider updates its floor context and sends a FLOOR_STATUS message to the FCI indicating the new floor context entry to the other members.
  - If not, the FLOOR_INHIBIT message is routed upward in the RB.
    - If the message passes an RB provider whose local floor context indicates the floor as grabbed, a FLOOR_ERROR message with code E_GRABBED is sent back to the originator.
    - If the message passes an RB provider whose local floor context indicates the floor as F_INHIBITED, an appropriate FLOOR_STATUS message is sent back to the originating RB provider.
    - If the message reaches the top RB provider, the global floor context is checked.
      - If the floor status is F_FREE or F_INHIBITED, the global floor context is changed accordingly, and a FLOOR_STATUS message is sent back via the RB indicating the current status. The FLOOR_STATUS message is relayed to the local FCI by the RB provider. Each FCI member updates its floor context appropriately.
      - If the floor status is F_GRABBED, a FLOOR_ERROR message with error code E_GRABBED is sent back to the originating RB provider being relayed on the FCI for indication.

6) Release Floor

The requesting participant sends a FLOOR_RELEASE message to the FCI. The RB provider checks its floor context whether the floor is either inhibited or grabbed locally.

- If not, the message is ignored since a floor is released which has not been allocated before.
- If yes, the RB provider must consider the following cases:
  - *grabbed floor*: The FLOOR_RELEASE message is sent upward via the resource backbone to indicate the status change to the top RB provider which updates the global floor context and sends a FLOOR_STATUS message back downward the RB indicating the new status. The local RB provider updates its local floor context and relays the FLOOR_STATUS message to the FCI indicating the status update to the FCI members.
  - *inhibited floor*: the requesting FCI member is deleted from the local floor holder list in the floor context (if not a floor holder, the message is ignored), the status is changed accordingly, and the new context is indicated by sending a FLOOR_STATUS to the FCI.
    - If the requesting FCI member was the last local holder of the floor, the FLOOR_RELEASE message is sent upward via the RB.
      - If the FLOOR_RELEASE message passes an RB provider whose local FCI still has at least one holder of that floor, the message is not routed upwards anymore because there is no status change necessary to be indicated.
      - If the message reaches the top RB provider and there is no other branch in the tree containing floor holders, the top RB provider changes the status in its global floor context accordingly.

7) Test Floor

The requesting participant first checks its own floor context information for getting the local floor information.

- If there is no floor context entry, it sends a FLOOR_TEST message to the FCI. The RB provider forwards this message upward in the RB.
  - If the message passes an RB provider with sufficient information, i.e., a valid floor context entry, it generates a FLOOR_STATUS message to be routed downwards via the RB. This RB provider is finally the top RB provider holding the global floor context information.

8) Ask Floor

The requesting participant sends a FLOOR_ASK message to the FCI. The RB provider checks its floor context information and must handle three cases.

- If the floor is inhibited locally, the RB provider sends a FLOOR_ERROR message with error code E_INHIBITED to the FCI.
- If the floor is grabbed locally, there is nothing to do since the floor holder received the FLOOR_ASK message, too.
- If there is no floor entry in the local context, the RB provider forwards the FLOOR_ASK message upwards in the RB.
  - If the message passes an RB provider with sufficient information, this RB provider relays the message to its local FCI when the floor is grabbed.
  - If the floor is indicated as inhibited in a passed RB provider's floor context, a FLOOR_ERROR message with error code E_INHIBITED is sent back via the RB downwards.
  - If the message reaches the top RB provider, the global floor context is checked.
    - If the floor status is F_GRABBED, a FLOOR_ASK message is forwarded downward the RB on the appropriate branch of the floor holders.
    - If the floor is either inhibited or free, a FLOOR_ERROR message with error code E_INHIBITED or E_FREE is sent back to the originating RB provider which relays this message to the FCI.

It can be seen that there is no response for a FLOOR_ASK message. However, this message is usually supplemented by an appropriate floor passing operation of the application.

9) Give Floor

The giving participant sends a FLOOR_GIVE message to the FCI. If the giving participant is not the floor holder or the floor is indicated as being inhibited, the message is ignored by both the RB provider and the FCI members. If the giving participant is the current floor holder, two cases must be considered.

- If the given participant is a local FCI member, all FCI members, including the RB provider, change their local floor context information indicating the given participant as the new floor holder.
- If the given participant is not a local FCI member, the RB provider forwards the FLOOR_GIVE message upwards via the RB, sets the local floor status entry to F_GIVING, and indicates the temporary floor context entry to the FCI by sending a FLOOR_STATUS message.
  - If the FLOOR_GIVE message reaches the top RB provider, the message is forwarded downwards via the appropriate branch of the RB.
  - If the message reaches the top RB provider on the same branch on which the floor holder should reside and the floor holder is not in the FCI of the top RB provider, a FLOOR_STATUS message is sent back to the originating RB provider to be relayed on the FCI indicating the old floor holder as the new one. Hence, the old status is re-established.
  - If during forwarding the message either upwards or downwards the RB provider is reached whose FCI contains the given participant, an appropriate FLOOR_STATUS message is relayed to the FCI and forwarding is stopped. Furthermore, a FLOOR_GIVEN message is sent back by the receiving RB provider in the reverse direction.
    - If the FLOOR_GIVEN message passes the top RB provider, the RBone branch information is changed in the global floor context (storing the old and new one).
    - If the FLOOR_GIVEN message is received by the originating RB provider, a FLOOR_STATUS message is sent to the local FCI to indicate the free status of the floor.

10) Holders of Floor

The requesting participant sends a FLOOR_HOLDER message to the FCI. Three cases have to be considered.

- If the floor is grabbed locally, the current floor owner responds by sending a FLOOR_HOLDER_LIST message with its own presence information. This information exchange can be avoided by checking the local floor context information in the requesting participant.
- If the floor is grabbed in another FCI, the RB provider forwards the FLOOR_HOLDER message upwards via the RB.
  - If the message reaches the top RB provider, it is forwarded downwards on the appropriate branch.
  - If the message passes the RB provider whose local FCI contains the current floor holder, this provider generates a FLOOR_HOLDER_LIST message containing the presence information of the current floor holder and sends it back in the reverse direction.
    - If the FLOOR_HOLDER_LIST message reaches the top RB provider, the message is forwarded downwards on the appropriate branch.
    - If the message reaches the RB provider of the requesting participant, the message is relayed on the FCI.
- If the floor is inhibited, the FLOOR_HOLDER message is forwarded upwards by the RB provider via the RB.
  - If the message reaches the top RB provider, a FLOOR_HOLDER_ASK message with an empty floor holder list is sent downwards on all branches in which the floor is marked as allocated.
    - These messages are forwarded in all RB providers on all connected branches until they reach a leaf node. This leaf node inserts its local floor holder information, if available, and sends the message back in upward direction.

- Each branching node cumulates the FLOOR_HOLDER_ASK messages on all branches into one message to be forwarded until a single message reaches the top RB provider.
- An appropriate FLOOR_HOLDER_LIST message is sent back an the appropriate branch of the RB in downward direction.
- If the message reaches the RB provider of the requesting participant, the message is relayed on the FCI.

11) Handling during RB Repair operations

If an RB repair procedure (see Section D.7) is started during forwarding any floor control message, forwarding is halted until the repair procedure is finished, signaled by RB_FINISH.

After finishing the repair procedure, forwarding messages is restarted at the originating RB provider. All other forwarding messages in the RB providers are discarded.

If the originating RB provider failed or quit, the successor of this RB provider has to restart the forwarding procedure. Thus, each local FCI member has to be aware of pending operations. This is feasible due to the multicast-based state information change.

For the floor give case, the originating RB provider re-sends the FLOOR_GIVE message with the temporary floor entry (state F_GIVING). The following cases must be considered:

- If the FLOOR_GIVE message reaches the RB provider whose FCI contains the new floor holder, the procedure is continued as in the original case depending on the current floor context entry.
- If the FLOOR_GIVE message reaches the top RB provider, the message is forwarded on the old branch using the appropriate entry in the global floor context. Further forwarding operations are handled the same way.

As it was stated in the introduction, this section extensively outlined the protocol mechanisms to be used to realize the proposed floor control services following the RBone approach. Although the explicit message notations are not presented, their extraction from the presented description should be easily feasible.

E. CONCLUSIONS

This paper presented a floor control approach which offers sophisticated services for the implementation of distributed application state synchronization and application resource access.

The main focus in the development of the underlying mechanisms was on providing a scalable solution in terms of participating users in the conference to ensure high responsiveness of the services. For that, the proposal applied the idea of establishing a *resource backbone* (RBone) topology interconnecting multicast-capable floor control islands similar to the MBone approach in the Internet. The RBone is comprised of a tree of selected entities in the conference being responsible for routing floor control requests outside the local island.

The paper depicted the mechanisms for maintaining the tree in terms of establishing and updating the topology in node and network failure cases. Furthermore, the service request handling was presented which follows the idea to handle requests locally, if possible, and to forward the requests globally, if needed.

Hence, the proposal applies efficient multicast transfer on local level and tree-based unicast routing on global level using a floor control specific routing scheme. Furthermore, recovery from network and node failures, both on local and global level, is supported to some extent as well using a heartbeat mechanism in the FCIs.

This approach is expected to improve the responsiveness of the provided floor control service even in conferences of larger scale, especially if the conference is comprised of a few subgroups each located in a fast local area network.

F. FUTURE WORK

There are several issues to be addressed in the future work. The first one is the proposal for a *naming scheme* for the floors. Currently, the naming of the floors is not within the scope of the protocol. However, to avoid conflicting floor names, a unique naming scheme might be desirable to be added to the service. This could easily be done by using a simple numeric identifier naming scheme similar to the ITU standards.

Secondly, the cases of failures in the protocol, specifically in the RBone, have to be studied more extensively to improve the robustness of the protocol.

Thirdly, the scalability and robustness of the protocol has to be studied in form of simulations or protocol prototyping.

G. REFERENCES

[1] M. Altenhofen, J. Dittrich, R. Hammerschmidt, T. Käppner, C. Kruschel, A. Kückes, T. Steinig, "The BERKOM multimedia collaboration service" in Proc. of ACM Multimedia, pp. 457-463, August 1993

[2] I. Beier, H. Koenig, "GCSVA - A Multiparty Videoconferencing System with Distributed Group and QoS Management" in Proc. of IEEE International Conference on Computer Communication and Networks (IC3N'98), pp. 594-598, October 1998

[3] C. Bormann, J. Ott, D. Kutscher, D. Trossen, "Simple Conference Control Protocol – Service Specification", Internet Draft, Work in Progress, February 2001

[4] H.-P. Dommel, J.J. Garcia-Luna-Aceves, "Group Coordination Support for Synchronous Internet Collaboration" in IEEE Internet Computing Magazine vol. 3 No. 2, April 1999

[5] H. Eriksson, "MBONE: The multicast backbone" in Communications ACM, vol. 37, no. 8, pp. 54-60, August 1994

[6] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing" in ACM Computer Communication Review, vol. 25, no. 4, pp. 342-356, October 1995

[7] M. Handley, J. Crowcroft, C. Bormann, J. Ott, "The Internet Multimedia Conferencing Architecture", Work in Progress, Nov. 2000

[8] M. Handley, C. Perkins, E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000

[9] M. Handley, V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998

[10] T. Helbig, D. Trossen, " The ITU T.120 Standard Family as Basis for Conferencing Applications" in Proc. of SPIE International Symposium Voice, Video, & Data Communications, pp. 190-201, November 1997

[11] ITU-T, "Data Protocols for Multimedia Conferencing", ITU-T Recommendation T.120, 1998

[12] ITU-T, "Multipoint Communication Service - Service Definition", ITU-T Recommendation T.122, 1998

[13] A. McKinlay, R. Procter, O. Masting, R. Woodburn, "Studies of turn-taking in computer-mediated communications" in Interacting with Computers, vol. 6 no. 2, pp. 151-171, June 1994

[14] R. Malpani, L. A. Rowe, "Floor control for large-scale MBone seminars" in Proc. of ACM Multimedia, November 1997

[15] L. Y. Ong, M. Schwartz, "Centralized and Distributed Control for Multimedia Conferencing" in Proc. of IEEE International Conference on Communications, pp. 197-201, June 1993

[16] D. Sisalem. H. Schulzrinne, "The Multimedia Internet Terminal" in Journal on Telecommunication Systems, Vol. 9 No. 3, pp. 423-444, 1998

[17] D. Trossen, "Scalable Conferencing Support for Tightly-Coupled Environments: Services, Mechanisms, and Implementation Design" in Proc. of IEEE International Conference on Communications, pp. 889-893, June 2000