By Verena Kahmann,
Jens Brandt, *and*
Lars Wolf

COLLABORATIVE STREAMING AND DYNAMIC SCENARIOS

This media streaming architecture solves the problems of collaborative session management, from session sharing to control of the common session state to session transfer among networked clients.

As her family watches a movie in the living room, the teenage daughter prefers to stay in her bedroom but also wants to know which movie the others are watching. She turns on her personal Internet device and joins the film session already in progress. Although she does not particularly like the movie, she wants to know whether it has a happy ending, so she jumps to a later scene without bothering anyone. Several minutes later, a French-speaking friend of the family drops by. He is invited to listen to the French audio stream of the movie through the earphones of his streaming audio-video-enabled PDA. Later, the mother asks to pause the movie using a remote control while she goes for food in the kitchen. The movie on the main TV screen, along with the French audio on the PDA, are now both paused. In this scenario, several media data transmissions between media source and user devices are closely related and provide for personalized collaboration among users.

Another example of collaborative media streaming involves distance learning in which a group of learners collaboratively participates in a course presentation. These people may be located in the same learning center or in several different centers. Remote attendance also helps reduce costs by sparing them having to travel. Trainers may be located at remote sites even as they moderate the presentation; when appropriate, they may also form subgroups of learners based on personal learning effort and level. Additionally, learners who want to advance more quickly may skip portions of the coursework on their own and later resynchronize with a common group timeline.

In these collaborative contexts, would well-known standard streaming protocols and mechanisms be suitable for the related media streaming? To answer, we must first weigh the differences between collaborative and individual streaming. In collaborative streaming,

IN HETEROGENEOUS

a group of users participates in presentations streamed from a remote media server to a number of client devices. Just as in standard media streaming architectures, a so-called streaming session must be established between each of the clients and the media server. The state of the session contains the current viewing position in time, as well as the identifiers of all tracks being presented.

Exactly how this state is controlled is the main difference between collaborative and individual media streaming. Clients easily control the session state in individual media streaming (such as by pausing the transmission of the stream), as the session is not associated with other users. In contrast, in a collaborative scenario the streaming sessions of all group members are associated with one another, but session changes do not always affect other group members. Consider again the family watching a movie when the teenage daughter jumps forward without the others. Hence, a new group is automatically created for her on the fly. Later, the mother controls the session state of all participants by pausing their common sessions together.

Another difference between collaborative streaming and standard streaming scenarios is the need to transfer streaming sessions to other clients (such as from the main display to the daughter's device). Here, the joining member is added to the group and informed about the group's policy and streaming session state. A third difference is that we cannot assume all group members are using similar devices. Devices are heterogeneous in nature, with different capabilities regarding display size, processing power, and network connections.

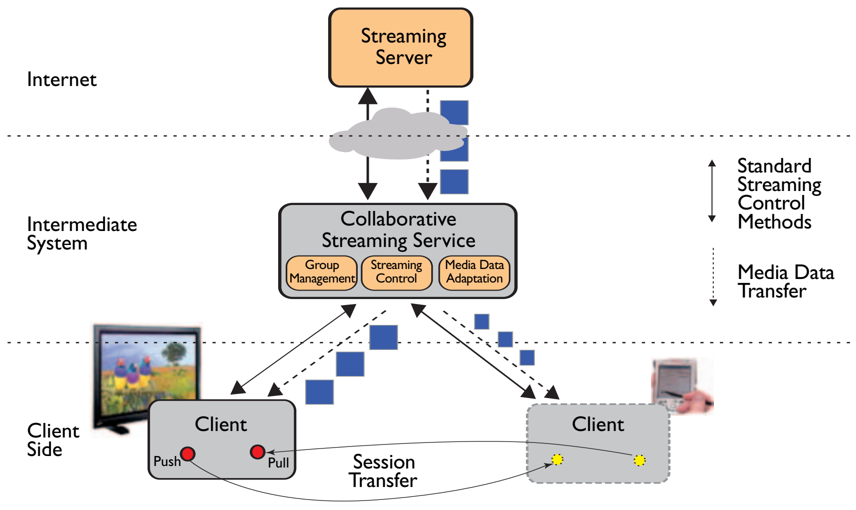


Figure 1. Collaborative streaming.

Standard streaming mechanisms alone are insufficient for collaborative streaming. The "co-stream" architecture we've been developing since 2002 at the Institute of Communication Systems and Computer Networks at the Technische Universität Braunschweig supports cooperation between the higher-layer IETF protocols Real Time Streaming Protocol (RTSP) [7] for streaming control and Session Initiation Protocol (SIP) [6] for session transfer. (Figure 1 outlines a networked collaborative scenario.) In this architecture, we use an intermediate system as the collaborative streaming service to provide group-specific streaming control, a group management component to account for membership, and a media adaptation component. The adaptation component uses transcoding mechanisms to mitigate user preferences and heterogeneous device requirements. In the figure, mitigation is indicated by the different packet sizes at the links to the respective client devices. The figure also outlines a session transfer between the left and the right clients.

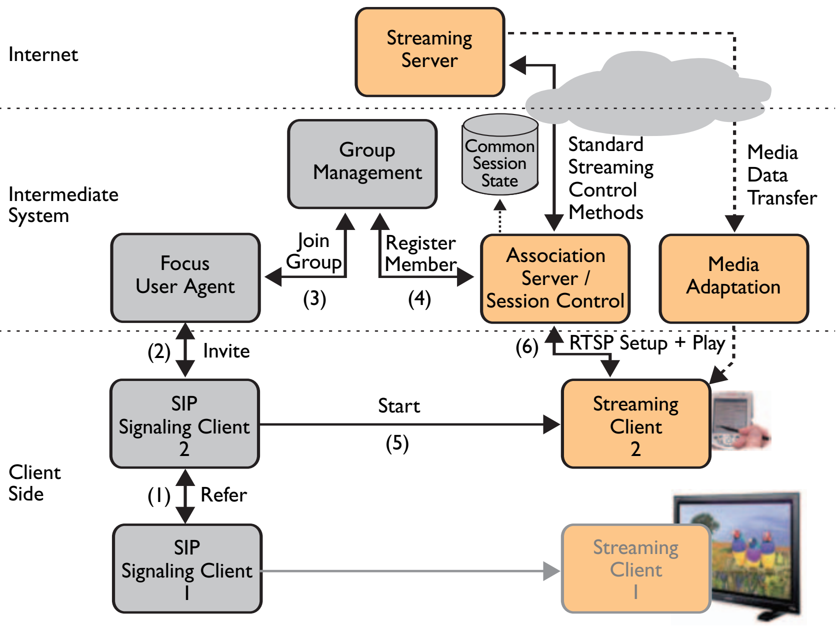


Figure 2. Message flow in copy-session transfer.

Main Services

A collaborative streaming architecture must offer two main services:

Session transfer. Session transfer denotes the signaling messages needed to transfer an existing streaming session from one device to another executed in two directions: Either the owner of a session wants to push the session to another device or someone who is interested in the session wants to pull it from the owner's device. In either case it must be determined whether the owner's session should be continued (the session is copied) or moved to the new device, requiring the media session be stopped on the owner's device; and

Session control. Session control means controlling the session state, or the play-time position, and the chosen tracks. For a movie streamed on a single device, the set of tracks is controlled in an aggregate fashion (such as by pausing the audio together with the video). Aggregate control for a collaborative group means that control actions are executed for all members, though this is not always reasonable. Thus, as a reaction to each control action, either a group is partitioned or the state of the whole group is changed to the newly requested play-time position and track.

Collaboration control involves solving several problems:

Defining a common session state. State in this context corresponds mostly to play-time position, though the tracks of a presentation are also useful (such as for viewing a presentation through a common camera angle if several cameras are available). Individual quality is not in the common session state, as heterogeneous device capabilities may exist, and each client receives the stream at an individual quality level;

Signaling session transfer and late joining of a streaming session. A newcomer must be equipped with the common streaming state automatically;

Controlling collaborative sessions. Conflicting control requests from the various group members must be resolved. Aggregate and nonaggregate control must be offered and provided on demand;

Supporting heterogeneous user preferences and device requirements. Media streams must be adapted to device capabilities, and the architecture must support standard clients as far as possible; and

Providing user, service, and session identification and location. Search engines and dynamic service registries can be used for service location and adapted for user location. However, the more short-lived streaming sessions are only rarely registered at any search engine or location service.

Additionally, standard real-time streaming mechanisms that allow for session control are required for collaborative streaming. Thus, it is reasonable to use standard streaming servers and clients, extending their functionality as needed.

Key Concepts

Collaborative streaming uses streaming control messages that can be mapped directly to RTSP methods that manage the streaming session state, or play-time position. SIP, in turn, provides methods for estab-

In a collaborative scenario the streaming sessions of all group members are associated with one another, but SESSION CHANGES DO NOT ALWAYS AFFECT OTHER GROUP MEMBERS.

lishing multimedia sessions (INVITE) and transfer (REFER) [6]. For tightly coupled conferences, the SIP conferencing model [5] uses a centralized focus to establish signaling relationships among all clients. Our architecture uses this model to manage collaborative streaming groups. Our solution—implementing a collaborative session transfer—involves the cooperation of SIP and RTSP through a proxy architecture. We divided the proxy into logical components that map standard protocol methods to collaborative functions (such as synchronization and state change). Thus, existing server components do not have to be changed. Alternatively, RTSP and SIP can be merged, as proposed in [10]. A different architectural concept called peer-to-peer streaming service, developed at the University of Illinois, distributes media among clients and can be extended to collaborative session control.

In order to manage streaming session state collaboratively, while also accounting for the interests of subgroups, we defined the concept of association: Two clients are in the same association if their common session state is the same, that is, they view the same play-time position at any given moment. The so-called Association Service, an extension of an RTSP streaming proxy, interprets usual RTSP control methods (such as SETUP and PLAY trick modes) and maintains associations according to the policy of the group. As a result, a client is synchronized to group state, the group state is changed, or an own association within a group (a logical subgroup with its own session state) is opened.

The collaboration policy of the group is based on the role-based access control model [1]. Our architecture defines several roles in which members and their permissions are defined. Member lists can consist of single-device identifiers or groups of clients (such as from a particular domain). Permissions allow users to execute control actions (such as joining, pausing, and seeking). Additionally, our architecture defines for each session control action of a particular role whether the association service should change the streaming state of a group or open a new association. Other policy-specification techniques are discussed in [3].

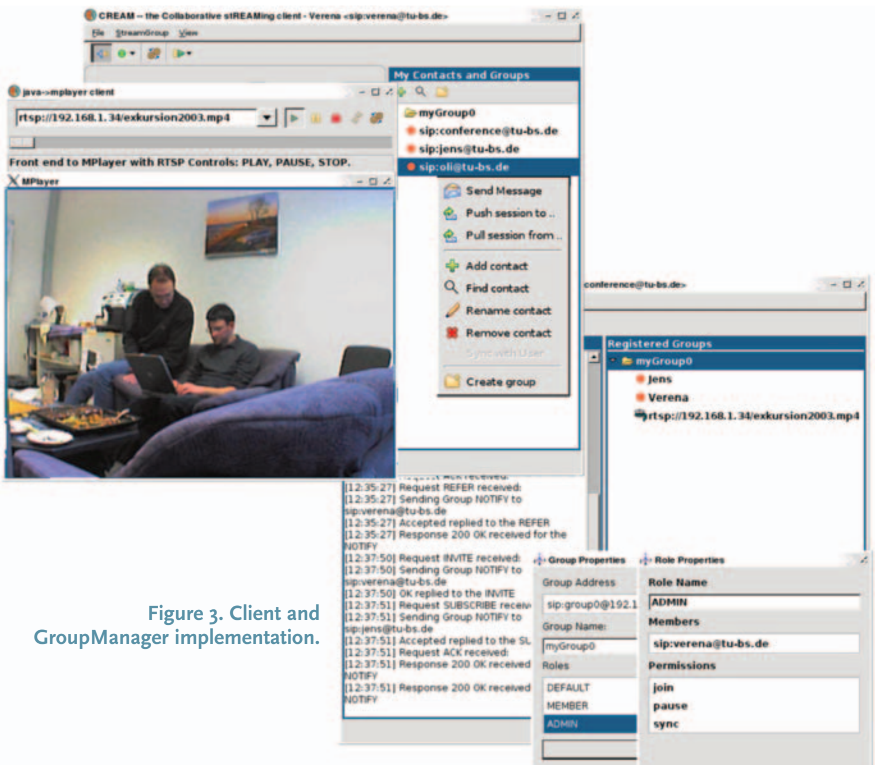
The collaboration policy of a group is established at the time of session startup by the group creator and saved in the second logical component of the collaborative streaming proxy (the group management component). Externally, this component is represented by a SIP conferencing user agent (called Focus), implementing common conferencing functionality accessed through SIP methods.

Providing for full collaborative session transfer, the group management component registers information at the association service and retrieves notifications about subgroups. Instead of centralized group management, information can also be kept at the clients themselves.

Internet signaling protocols offer interesting and extensible methods to help build user-friendly interfaces. The separation of signaling and data transport ensures continuous presentation of media streams. To assist users in finding session identifiers, SIP offers subscriptions to the session state. SIP also provides for challenge-based authentication. Meanwhile, preferences (such as the French language in the movie-watching-family example) are initiated by RTSP methods. Another aspect of user friendliness is the automatic adaptation of media resources to the various capabilities of media devices, so users are spared having to choose from a possibly confusing set of media formats. Hence, the collaborative streaming service in our architecture offers a media adaptation service component to tailor video and audio streams to device capabilities and user requirements on the fly.

Collaborative Streaming Architecture

Figure 2 clarifies the construction of the architecture by showing how the family in the movie-watching example pushes and copies its streaming session from its flat screen (Client 1) to the French visitor's PDA (Client 2). Thus, Client 1 sends (1) a REFER to this client, including identifying the group to which Client 2 should be connected (the group URI). In case the SIP address of Client 2 is unknown to Client 1, it must use a search request to look it up before sending the REFER (not shown in the figure). Client 2 accepts the REFER and sends (2) an INVITE to the group URI, which is routed to the Focus, a central SIP conference manager. The Focus implements basic group signaling features and executes (3) a JoinGroup function call at the group management component. Once the policy checks whether the member is allowed to join, the new member is registered (4) at the association service. Client 2 may now start (5) its streaming client and send (6) RTSP requests to set up a streaming session to the association service. There, the joining client is synchronized to the streaming session.



rtsp://192.168.1.34/exkursion2003.mp4

Front end to MPlayer with RTSP Controls: PLAY, PAUSE, STOP.

```
[12:35:27] Request REFER received:
[12:35:27] Sending Group NOTIFY to
sip:verena@tu-bs.de
[12:35:27] Accepted replied to the REFER
[12:35:27] Response 200 OK received for the
NOTIFY
[12:37:50] Request INVITE received:
[12:37:50] Sending Group NOTIFY to
sip:verena@tu-bs.de
[12:37:50] OK replied to the INVITE
[12:37:51] Request SUBSCRIBE receiv
[12:37:51] Sending Group NOTIFY to
sip:jens@tu-bs.de
[12:37:51] Accepted replied to the SL
[12:37:51] Request ACK received:
[12:37:51] Response 200 OK received
NOTIFY
[12:37:51] Response 200 OK received
NOTIFY
```

Group Address: sip:group0@192.1

Group Name: myGroup0

Roles: DEFAULT, MEMBER, ADMIN

Role Name: ADMIN

Members: sip:verena@tu-bs.de

Permissions: join, pause, sync

Figure 3. Client and GroupManager implementation.

The association service and group management component can both be implemented on the same intermediate system. It is reasonable to integrate the conferencing focus and group management component into a single application, as each group-signaling method is able to manipulate the group state. Physically, the system should be located close to the clients, at least for the association service, in order to achieve good synchronization between clients and small answering delays. The streaming server itself can be located elsewhere on the global Internet.

As mentioned, an adaptation of the media stream is needed when streaming to different devices with different capabilities. Due to its compressed nature, digital video cannot be tailored directly but needs to be decompressed, adapted, and compressed again. This procedure is quite costly in terms of processing time. Another possibility is the use of transcoding techniques, through which the video stream is tailored within the compressed state. Thus, processing time for decompressing and compressing can be saved at the expense of less flexibility and more complexity compared to tailoring uncompressed video. Still, a good deal of processing power is needed to adapt the stream.

Therefore, tailoring video is not possible on the presenting device and needs assistance from the network. Our architecture, as shown in Figures 1 and 2, uses the proxy as the point of adaptation. The device capabilities, as well as the user's preferences, are negotiated between the mobile device and the proxy when the RTSP session is set up. The adaptation component uses them to tailor the stream to the requirements of the client. It is also possible to change these requirements during the RTSP session, perhaps following a change in network conditions.

The group-signaling and conferencing functions we have developed and incorporated into the architecture are based on the National Institute of Standards and Technology (NIST) SIP API implementation; we have also used the NIST presence proxy for registration and routing functionality [4]. In addition, we have implemented our own RTSP/RTP proxy in C++ for synchronization among clients. For the client-side RTSP, we use a standard media player with our own RTSP integration to support streaming control. This player is controlled by a front-end providing for VCR functionality and session control keys integrated in our client implementation; Figure 3 includes a screenshot of the client implementation (top-left corner) and the group manager application (bottom-right corner). The graphical user interface of the group management application allows administrators to control who is a group member at any given time and to define and edit member roles accordingly.

Related Work

Aspects of collaborative streaming are being addressed through several alternative approaches. For example, session-transfer features have been implemented by SIP methods in mobile IP-based environments [8], as well as by multimedia middleware for home networks. For example, Network-Integrated Multimedia Middleware (NMM), Standard streaming mechanisms alone are insufficient for COLLABORATIVE STREAMING.

middleware supporting media-session sharing, uses a flow graph to which late-joining clients are connected [2]. NMM clients are synchronized through a common clock, and play-out delays are mitigated through a synchronization controller unit. A generic late-join service (combined with a media data transport for interactive applications called RTP/I) was proposed in [9]. The RTP/I media data packets themselves carry session state and events, making signaling scalable to larger groups, as well as to generic collaborative groupware. Group management aspects of collaborative streaming have been considered in architectures for collaborative groupware and for multicast conferencing [3].

Conclusion

We have implemented a comprehensive architecture for collaborative streaming to solve the problems of session transfer among clients. We use SIP conferencing features for session sharing, with management of the common session state through the association service and of session control through RTSP methods. A common group management application controls all actions while applying group policy.

Applying IETF application-layer signaling protocols enables general support for mobility. However, an open problem is the selection of a suitable intermediate system after a handover resulting from user mobility.

Another interesting task for us is the application of service composition. Several services must be integrated to form a collaborative streaming service. Providers may implement them differently; thus the automated search and matching of these services to form a comprehensive service is relevant for the enhanced, user-friendly autoconfiguration of online interactive media involving collaborative users in heterogeneous access networks. C

References

1. Ferraiolo, D. and Kuhn, R. Role-based access controls. In *Proceedings of the 15th National Institute of Standards and Technology National Computer Security Conference* (Baltimore, Oct. 13–16). U.S. Government Printing Office, Washington, D.C., 1992, 554–563.
2. Lohse, M., Repplinger, M., and Slusallek, P. Dynamic distributed multimedia: Seamless sharing and reconfiguration of multimedia flow graphs. In *Proceedings of the Second International Conference on Mobile and Ubiquitous Multimedia* (Norrköping, Sweden, Dec. 10–12). ACM Press, New York, 2003, 89–95.
3. Meissner, A., Musunoori, S., and Wolf, L. MGMS/GML: Towards a new policy specification framework for multicast group integrity. In *Proceedings of the Symposium on Applications for the Internet Publisher* (Tokyo, Jan. 26–30). IEEE Computer Society Press, Los Alamitos, CA, 2004, 233–239.
4. National Institute of Standards and Technology. *Project IP Telephony/VoIP, 2002–2006.* NIST, Gaithersburg, MD, 2002–2006; snad.ncsl.nist.gov/proj/iptel/.
5. Rosenberg, J. *A Framework for Conferencing with the Session Initiation Protocol.* RFC 4353, Internet Engineering Task Force (Feb. 2006); ietf.org/rfc/rfc4353.txt.
6. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, G., Handley, M., and Schooler, E. *SIP: Session Initiation Protocol.* RFC 3261, Internet Engineering Task Force (June 2002); ietf.org/rfc/rfc3261.txt.
7. Schulzrinne, H., Rao, A., and Lanphier, R. *Real-Time Streaming Protocol.* RFC 2326, Internet Engineering Task Force (Apr. 1998); ietf.org/rfc/rfc2326.txt.
8. Shacham, R., Schulzrinne, H., Thakolsri, S., and Kellerer, W. The virtual device: Expanding wireless communication services through service discovery and session mobility. In *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking, and Communications* (Montreal, Aug. 22–24). IEEE Communications Society, New York, 2005, 73–81.
9. Vogel, J., Mauve, M., Geyer, W., Hilt, V., and Kuhmunch, C. A generic late join service for distributed interactive media. In *Proceedings of the Eighth ACM Multimedia Conference* (Los Angeles, Oct. 30–Nov. 3). ACM Press, New York, 2000, 259–267.
10. Whitehead, S., Montpetit, M., and Marjou, X. *An Evaluation of Session Initiation Protocol for Use in Streaming Media Applications.* Internet Draft (June 2006); ietf.org/internet-drafts/draft-whitehead-mmusic-sip-for-streaming-media-01.txt.

Verena Kahmann (kahmann@ibr.cs.tu-bs.de) is a computer science Ph.D. candidate and research assistant in the Institute of Operating Systems and Computer Networks at Technische Universität, Braunschweig, Germany.

Jens Brandt (brandt@ibr.cs.tu-bs.de) is a computer science Ph.D. candidate and research assistant in the Institute of Operating Systems and Computer Networks at Technische Universität, Braunschweig, Germany.

Lars Wolf (wolf@ibr.cs.tu-bs.de) is a professor of computer science and head of the Institute of Operating Systems and Computer Networks at Technische Universität, Braunschweig, Germany.