

A Scalable Overlay Video Mixing Service Model

Bin Yu

Department of Computer Science
University of Illinois at Urbana-Champaign,
1304 W. Springfield
Urbana, IL 61801, U.S.A.

binyu@uiuc.edu

Klara Nahrstedt

Department of Computer Science
University of Illinois at Urbana-Champaign,
1304 W. Springfield
Urbana, IL 61801, U.S.A.

klara@uiuc.edu

Categories and Subject Descriptors

H.5.3 [Group and Organization Interfaces]: Collaborative computing, Computer-supported cooperative work.

General Terms

Algorithms, Design, Human Factors.

Keywords

Multiparty video conferencing, video mixing, overlay, service.

1. INTRODUCTION

In recent years, personal computing devices have been upgrading rapidly in terms of Internet connection bandwidth, processing speed, storage capacity and multimedia capturing/rendering capabilities. At the same time, distributed multiparty video conferencing is becoming urgently needed as a distant visual communication tool for social and business activities, such as group discussion for distant learning, fact-to-face Internet chatting room and distributed collaboration. However, in most scenarios, it is desired that each host receives a *single* incoming video stream that is presented in a moderate-sized window for several reasons. First, some hosts may have limited resources in terms of inbound bandwidth, number of network connections (e.g., one connection for TV devices). Second, many personal computing devices have a limited screen size, and users may want to save display space for other tasks (e.g., Powerpoint presentation, shared white board, etc.). Third, viewers only have limited attention focus, and they cannot capture what is happening across too many streams when they are all presented. This mismatch between *multiple video sources* and *single stream presentation* is the key issue we have to solve to make such video conferencing systems an effective tool for distributed tasks.

The vision behind our solution is that the *information* contained in each video stream is changing through time, and no single stream is interesting to all user all the time. As shown in Figure 1, suppose we know how much interesting information is contained in the three streams 1, 2 and 3, then we could always present a user with the *currently most interesting* video stream at any time point. In addition, in case an event happened a short while ago is more interesting than what is happening currently, we can replay the cached segments if the user desires. This is also illustrated in Figure 1 where a segment of video 1 is shadowed by video 3 when it occurs, but this segment is cached and replayed later when it becomes more interesting than others.

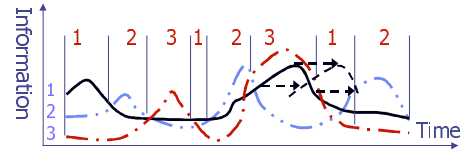


Figure 1. Information disparity over time and between streams

There are many challenges in implementing such a video conferencing system. First, we need to define how video streams are compared. Second, we need a scalable and flexible video editing framework that could support distributed video comparison and accommodate customized video mixing and delivery. To meet these challenges, we propose the notion of *information_score* to represent the information content of a video stream, and a distributed video mixing service model that combines automatic video analysis with user interaction to deliver the most informative video segments across time and space.

2. OUR SOLUTION

In this section, we first introduce the scoring system we use to compare video streams, and then present a video mixing service model that supports such distributed video editing/forwarding.

2.1 Information Content Scoring

We define an *Informative Activity* to be an activity that contains the information a user is interested in. There can be many such activities, including but not confined to: (1) human gestures and expressions, such as laughing, nodding/shaking heads, and leaving/resuming seat; (2) system meta data, such as timestamp and audio information (voice and silence); and (3) user interface events, such as circling an area or pressing a button.

The goal of our video conferencing system is focused on presenting the most informative video segments to a user at any time, and we need a comparison scheme to identify what a user wants most at a given time. Particularly, a trade-off has to be made between two extremes: (1) maximum user customization – allowing a user to choose the stream he wants all the time all by himself, and (2) maximum system automation – letting the system to make automatic decisions on which and how streams are compared. We achieve a middle-ground between them by assigning *Information_score* to each stream, where

$$\text{Information_score} = (\text{Base_score} + \text{Activity_score}) * \alpha^{\text{seconds_passed_by}} \quad (1)$$

Base_score is a score assigned to each video stream by a user via the GUI. All the available streams will be listed for the user as text information, static icons or low frame rate video, and the user

COPYRIGHT IS HELD BY THE AUTHOR/OWNER(S).

MM'03, NOVEMBER 2-8, 2003, BERKELEY, CALIFORNIA, USA.

ACM 1-58113-722-2/03/0011.

assigns a score to each of these streams. If user A is inclined to watch image of user B and C most of the time, he can assign a higher score for these two streams than others. *Activity_score* is associated with each informative activity, and viewers can assign different scores to different activities. For example, if a viewer always wants to watch who ever is speaking the loudest, he can assign a very high score to the activity of *high voice volume*. α is a value less than 1, and it is used to attenuate past activities' score. Basically we want to retain some activities happened before, yet we do not want to present too obsolete video segments. Overall, at the beginning of the conference session, streams are compared by their base scores, and whenever an informative activity happens or ends, the score for that stream will be changed. Also, as time goes by, the score of a past video segment will also decrease. Throughout the session, based on the viewer's preference, the N top ranking video segments are mixed and presented to a viewer.

2.2 Distributed Video Mixing Service Model

To utilize *information_score* in video selection and delivery, we need a service model that is scalable, flexible, self-growing and resource efficient. We have chosen the overlay service model where service components are deployed on participating hosts or application-level service proxies.

There are primarily three kinds of system components: *source*, *mixer* and *receiver*. One *source* node resides in each video stream source host, and it serves three functions: (1) sending video stream to mixers via Multicast or application level Multicast, (2) detecting informative activity and notifying registered mixers for score changing events, and (3) caching L_1 seconds of video segment from this source that is of highest score in the last L_2 seconds, and forwarding this segment when its score is higher than other live video segments (L_1 and L_2 are engineering parameters set by users). One *receiver* node resides on each viewer host, and it serves two functions: (1) receiving and presenting the single video stream to the viewer, and (2) collecting and publishing viewer's score assignment and video mixing layout choices (4-video-in-1, Picture-in-Picture, etc.) to mixers. *Mixer* nodes may be deployed on any host with enough resources (CPU, memory, bandwidth, etc.), and they have both *control* functions and *data* functions. For control functions, mixer nodes collaboratively maintain the session membership, and make decisions on stream score comparison and stream mixing scheme. For data functions, mixer nodes carry out the actual video editing and rate-adaptation operations such as down-scaling, tiling, forwarding and frame-dropping.

We are working on a video mixing tree construction algorithm, and Figure 2 shows an example topology. There are nine hosts (A to I) involved, and we focus on the receiver node *R1* on host *I*. Logically, video streams from six sources *S1* to *S6* are edited/mixed by four mixers *M1* to *M4* before the final stream is sent to *R1*. Physically, these sources/mixers/receivers may be deployed on the same host. At the current time, *M1* is down-scaling and forwarding *S1*'s video to *M4*, while *M3* is down-scaling and forwarding *S1*'s video to *M4*. *M4* mixes the two incoming streams and forwards the result to *R1*. Currently we only assume each viewer will have a set of dedicated mixing resources.

3. RELATED WORK

A huge amount of work has been done on video conferencing, and we could only list a few that are most closely related to our

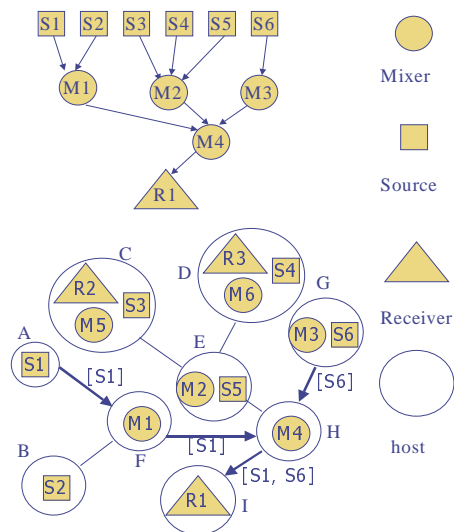


Figure 2. Logical and Physical Video Mixing Tree

project. In [1], Radenkovic et al. present a partial mixing architecture for audio streams similar to our work, but she focuses on TCP-friendly rate control while we work on video comparison. In [2], Kelly et al. propose a Multiple Perspective Interactive Video system that selects the “best view” from multiple cameras, and we extend their idea into a more general score-based framework that works with distributed events in addition to a single location event. In [3,4], Ooi et al. describe how to locate programmable media gateways across the Internet, but they assume the functional graph is given in advance while we combine the mixing tree construction with mixer deployment. In “Where Were We” [5], Minneman et al. advocate the notion of *near-synchronous* video presentation that gives us the idea of delayed presentation of past interesting video segments.

4. CONCLUSION AND FUTURE WORK

In this paper, we have described a distributed video mixing model for delivery of multiple video streams in a single video stream. For the next step, we will be working on (1) a video mixing tree construction algorithm, (2) grouping receivers of similar interests to share resources and reduce cost, and (3) implementation and evaluation of a video conferencing tool.

5. ACKNOWLEDGEMENT

The authors want to thank the insightful comments from the reviewers.

6. REFERENCES

- [1] M. Radenkovic and C. Greenhalgh, Multi-party Distributed Audio Service with TCP Fairness, ACM Multimedia 2002.
- [2] P. Kelly, A. Katkere, D. Kuramura, S. Moezzi, S. Chatterjee, and R. Jain, An architecture for multiple perspective interactive video, ACM Multimedia 1995.
- [3] W. T. Ooi, R. van Renesse and B. Smith, Design and Implementation of Programmable Media Gateways, NOSSDAV 2000.
- [4] W. T. Ooi, oi and R. van Renesse, Distributing Media Transformation Over Multiple Media Gateways, ACM Multimedia 2001.
- [5] S. L. Minneman and S. R. Harrison, Where Were We: Making and Using Near-Synchronous, Pre-Narrative Video, ACM Multimedia 1993.