Optimizing Quality for Collaborative Video Viewing

Dan Phung
Computer Science
Columbia University
New York City, New York
phung@cs.columbia.edu

Giuseppe Valetto
Computer Science
Columbia University
New York City, New York
and Telecom Italia Lab
Turin, Italy
valetto@cs.columbia.edu

Gail Kaiser
Computer Science
Columbia University
New York City, New York
kaiser@cs.columbia.edu

ABSTRACT

The increasing popularity of distance learning and online courses has highlighted the lack of collaborative tools for student groups. In addition, the introduction of lecture videos into the online curriculum has drawn attention to the disparity in the network resources used by the students. We present an architecture and adaptation model called AI^{2}TV (Adaptive Internet Interactive Team Video), a system that allows geographically dispersed participants, possibly some or all disadvantaged in network resources, to collaboratively view a video in synchrony. AI^{2}TV upholds the invariant that each participant will view semantically equivalent content at all times. Video player actions, like play, pause and stop, can be initiated by any of the participants and the results of those actions are seen by all the members. These features allow group members to review a lecture video in tandem to facilitate the learning process. We employ an autonomic (feedback loop) controller that monitors clients' video status and adjusts the quality of the video according to the resources of each client. We show in experimental trials that our system can successfully synchronize video for distributed clients while, at the same time, optimizing the video quality given actual (fluctuating) bandwidth by adaptively adjusting the quality level for each participant.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Client/server, Distributed applications; D.2.8 [**Software Engineering**]: Metrics – performance measures; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems; H.5.3 [**Group and Organization Interfaces**]: Computer-supported cooperative work, Synchronous interaction; K.3.1 [**Computer Uses In Education**]: Collaborative learning, Distance learning

ACM-MM 2004 New York, NY USA

General Terms

ALGORITHMS, MEASUREMENT, PERFORMANCE, EXPERIMENTATION

Keywords

Synchronized Collaborative Video, Autonomic Controller

1. INTRODUCTION

Distance learning programs such as the Columbia Video Network and the Stanford Center for Professional Development have evolved from fedexing lecture video tapes to their off-campus students to instead streaming the videos over the Internet. The lectures might be delivered "live", but are frequently post-processed and packaged for students to watch (and re-watch) at their convenience. This introduces the possibility of forming "study groups" among off-campus students who view the lecture videos together, and pause the video for discussion when desired, thus approximating the pedagogically valuable discussions of on-campus students. Although the instructor is probably not available for these discussions, this may be an advantage, since on-campus students are rarely afforded the opportunity to pause, rewind and fast-forward their instructors' lectures.

However, *collaborative video viewing* by multiple geographically dispersed users is not yet supported by conventional Internet-video technology. It is particularly challenging to support WISIWYS (what I see is what you see) when some of the users are relatively disadvantaged with respect to bandwidth (e.g., dial-up modems) and local computer resources (e.g., archaic graphics cards, small disks). We have adopted technology (developed by others, Liu and Kender [33]) for "semantically compressing" standard MPEG videos into sequences of still JPEG images. This technology automatically selects the most semantically meaningful frames to show for each time epoch, and can generate different sequences of JPEG images for a range of different compression (bandwidth) levels. This approach works very well for typical lecture videos, where it is important, for instance, to see what the instructor has written on the blackboard after he/she stands aside, but probably not so important to see the instructor actually doing the writing, when his/her hand and body may partially cover the blackboard.

The remaining technical challenge is *synchronizing* the downloading and display of the image sequences among each of the distributed user clients, including support for shared video player actions such as pause. Further, if student groups

do indeed sometimes pause the videos, or rewind to a point already available in local buffers (caches), it is desirable to take advantage of the then-idle network connection to prefetch future images at a higher quality level.

We have developed an approach to achieving this, using three mechanisms working in tandem. First, the software clocks of the video clients are synchronized using NTP [36]. This time is used for reference within the image sequences, where each image is associated with its start and end times relative to the beginning of the sequence. Second, the video clients communicate with each other over a distributed publish-subscribe event bus, which propagates video actions taken by one user in the group to all the other users in the group. Thus any user can select a video action, not just a "leader".

Finally, the main innovation of this research concerns optimizing video quality in this context: A decentralized feedback control loop dynamically adjusts each video client's choice of both the next image to display and also the next image to retrieve from the semantic compression levels available. The controller relies on sensors embedded in each client to periodically check what image is currently displaying, whether this image is "correct" for the current NTP time compared to what other clients are viewing, which images have already been buffered (cached) at that client, and what is the actual bandwidth recently perceived at that client. Actuators are also inserted into the video clients, to modify local configuration parameters on controller command. The controller utilizes detailed information about the image sequences available at the video server, including image start and stop times (both the individual images and their start and stop times tend to be different at different compression levels), but unlike local client data, video server data is unlikely to change while the video is showing. A single controller is used for all clients in the same user group, so it can detect "skew" across multiple clients, and may reside on the video server or on another host on the Internet.

In the next section, we further motivate the collaborative video viewing problem, provide background on the semantically compressed video repository, and explain the technical difficulties of optimizing quality while synchronizing such semantically compressed videos. The following section presents our architecture and dynamic adaptation model, and its implementation in AI^{2}TV (Adaptive Interactive Internet Team Video). In the evaluation section, we describe the criteria used to evaluate the effectiveness of our approach, and show empirical results obtained when applied to real lecture videos distributed for a recent Columbia Video Network course. We compare to related work, and then summarize our contributions.

2. MOTIVATION AND BACKGROUND

Correspondence courses have been available to working adult and/or geographically remote learners for over a century, e.g., the American School in Illinois has offered high school courses since 1897 [6], and the University of Wyoming began offering extension courses in 1892, launching a full-fledged college distance learning program in 1906 [35]. Correspondence courses have traditionally been designed for individual students with a self-motivated learning style, studying primarily from text materials.

An NSF Report [7] discusses how technology, from radio to television, to audio and video cassettes, to audio and video conferencing, has affected distance education. These technologies have enabled educational institutions to offer certification and degree tracks using live or pre-taped audio and/or video of regular on-campus classroom lectures. The report states that the recent use of Internet technologies, especially the Web, has "allowed both synchronous and asynchronous communication among students and between faculty and students" and has "stimulated renewed interest in distance education". It also mentions that "stimulating interaction among students" can help reduce dropout rates, which it says may be higher in distance education than in traditional courses. Finally, it cites some studies that "suggest the Web is superior to earlier distance education technologies because it allows teachers to build collaborative and team-oriented communities rather than either the passive classes of the conventional academy or the individual study of traditional correspondence courses".

Today's equivalent of correspondence courses are often offered online through a Web portal interface, with some for-profit schools like University of Phoenix [3] and Capella University [1] primarily online. Tools such as instant messaging, application and desktop sharing [4, 5], and co-browsing [11, 30, 41] facilitate the communicative aspects of synchronous collaboration but are not designed specifically for educational purposes. Support for synchronous collaboration remains a major concern in courses where group work is encouraged [46], yet there are few educational tools that allow synchronous collaboration across a group of online students [9]. However, it seems that Web-based video streaming should enable synchronous collaboration "situated" by collaborative lecture video viewing, approximating the experience of on-campus students physically attending the lecture and class discussion.

Our AI^{2}TV project aims to contribute to the area of synchronous collaboration support for distance education, specifically in the context of collaborative video viewing over the Internet. Our approach is directed at users with dial-up level bandwidths, who still constitute a significant portion of the Internet user community [37], to allow them to collaborate with other users that enjoy broadband or higher bandwidth resources.

Viewing video on the Internet usually requires relatively high bandwidth resources, and low-bandwidth or lossy network connections can lead to lost video content. This is particularly a problem for group review of lecture videos, if different members of the group miss different portions of the video or fall behind to different degrees due to extensive buffering. Furthermore, disadvantages in network and computing resources may make it difficult if not impossible – with current Internet video technology – for some students to participate in collaborative lecture video viewing at all.

Technically, collaborative video sharing poses a twofold problem: on the one hand, it is mandatory to keep all users synchronized with respect to the content they are supposed to see at any moment during play time; on the other hand, it is important to provide each individual user with a level of quality that is optimized with respect to the user's available resources, which may vary during the course of the video.

One solution to the problem of balancing the group synchronization requirement with the optimization of individual viewing experiences is to use videos with cumulative layering [34], also known as scalable coding [28]. In this approach, the client video player selects a quality level appropriate for

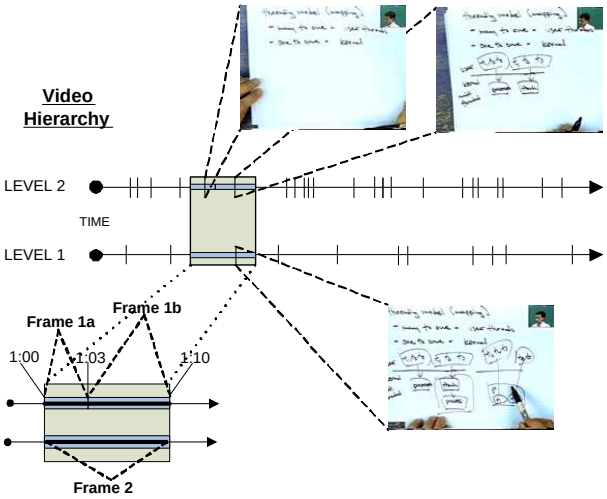

Figure 1: Semantic Video Scenario

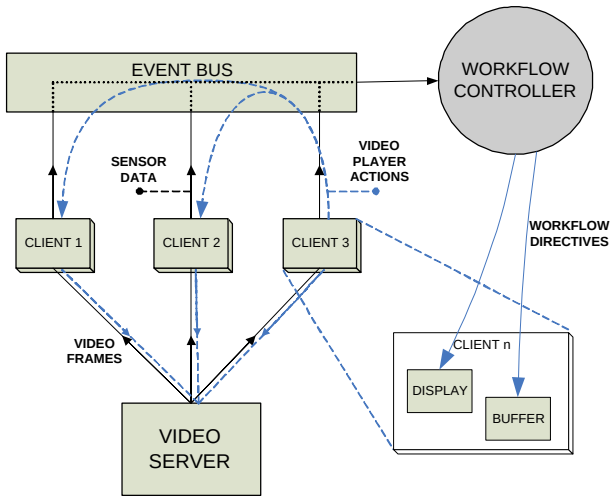

Figure 2: AI^{2}TV Architecture

that client's resources from a hierarchy of several different encodings for that video. Thus a client could receive an appropriate quality of video content while staying in sync with the other members of the group.

We use *semantic compression* to produce a video with cumulative layering. A semantic compression algorithm developed by Liu and Kender [33] reduces a video to a set of semantically significant key frames. That tool operates on conventional MPEG format videos and outputs sequences of JPEG frames, some of which are displayed in figure 1. The semantic compression algorithm profiles video frames within a sliding time window and selects key frames that have the most semantic information with respect to that window. By increasing the size of the window, a key frame will represent a larger time slice, which means that a larger window size will produce less key frames as compared to a smaller window size setting.

A conceptual diagram of a layered video produced from this semantic compression is shown in figure 1. Note that the semantic compression algorithm produces an effectively random distribution of key frames, hence the video produced by the package plays back at a *variable* frame rate. The variability in the frame rate is most significant when there are pockets of relatively high frequency semantic change, which result in sections in the video that demand a higher frame rate. The variable frame rate video adds substantial complexity to the bandwidth demands of the client.

In figure 1, the bottom-left in-set shows the juxtaposition of individual frames from two different quality levels. Each frame has a representative time interval `[start:end]`. For the higher level, Frame 1a represents the interval from 1:00 to 1:03, and Frame 1b represents the interval from 1:04 to 1:10. For the lower level, Frame 2 represents the entire interval from 1:00 to 1:10. In this diagram, Frame 2 is semantically equivalent to Frame 1a and 1b together. However, in real JPEG frame sequences produced from the same MPEG video for different quality levels, the start and end times of frame sets rarely match up as ideally as in our example.

Through the use of the Liu/Kender video compression algorithm, we can potentially provide semantically equivalent content to a group of user clients with diverse resources by adjusting the compression level assigned to each client while the users are watching the video. Thus for our purposes, synchronization of collaborative video boils down to showing semantically equivalent frames.

To adjust the clients in response to the changing environment, we use an "autonomic" controller to maintain the synchronization of the group of video clients while simultaneously fine tuning the video quality for each client. The term *autonomic* is borrowed from IBM to mean a self-managing system that uses a (software) feedback control loop [23]. Their terminology was invented for the self-management of data center operations, whereas our application applies to the novel domain of multi-user video synchronization.

Our autonomic controller remains conceptually separate from the controlled video system, employing our decentralized workflow engine, named Workflakes [42], to achieve the control capabilities. Said workflow coordinates the behavior of software entities, as opposed to conventional human-oriented workflow systems; the use of workflow technology for the specification and enactment of the processes coordinating software entities was previously suggested by Wise at al. [27]. Workflakes has previously been used in a variety of more conventional "autonomic computing" domains, where it orchestrates the work of software actuators to achieve the fully automated dynamic adaptation of distributed applications [43, 25, 26]. In the context of AI^{2}TV, Workflakes monitors the video clients and consequently coordinates the dynamic adjustment of the compression (quality) level currently assigned to each client.

3. ARCHITECTURE AND ADAPTATION MODEL

3.1 System Architecture

AI^{2}TV involves several major components: a video server, video clients, an autonomic controller, and a common communications infrastructure, as shown in figure 2.

The video server provides the educational video content to the clients for viewing. Each lecture video is stored in the form of a hierarchy of versions, produced by running the semantic compression tool multiple times with settings for

different compression levels. Each run produces a sequence of JPEG frames with a corresponding frame index file. The task of the video server is simply to provide remote download access to the collection of index files and frames over HTTP.

The task of each video client is to acquire video frames, display them at the correct times, and provide a set of basic video functions. Taking a functional design perspective, the client is composed of four major modules: a time controller, video display, video buffer that feeds the display, and a manager for fetching frames into the buffer.

The time controller's task is to ensure that a common video clock is maintained across clients. It relies on NTP to synchronize the system's software clocks, therefore ensuring a common time base from which each client can reference the video indices. Using this foundation, the task of each client is simplified to displaying the client's needed frame at the correct time. Since all the clients refer to the same time base, then all the clients are showing semantically equivalent frames from the same or different quality levels.

The video display renders the JPEG frames at the correct time into a window and provides a user interface for play, pause, goto and stop. When any participant initiates such an action, all other group members receive the same command, thus all the video actions are synchronized. Video actions are time stamped so that clients can respond to those commands in reference to the common time base. The video display knows which frame to display by using the current video time and display quality level to index into the frame index for the representative frame. Before trying to render the frame, it asks the video buffer manager if the needed frame is available. The video display also includes a control hook that enables external entities, like the autonomic controller, to adjust the current display quality level.

The video manager constitutes a downloading daemon that continuously downloads frames at a certain level into the video buffer. It keeps a hash of the available frames and a count of the current reserve frames (frames buffered) for each quality level. The buffer manager also includes a control hook that enables external entities to adjust the current downloading quality level.

The purpose of the autonomic controller is to ensure that, given the synchronization constraint, each client plays at its highest attainable quality level. The controller is itself a distributed system, whose design derives from a conceptual reference architecture for autonomic computing platforms proposed by Kaiser *et al.* [24], which is shown in figure 3.1. The architecture provides an end-to-end closed control loop, in which sensors attached to a generic (possibly legacy) target system continuously collect and send streams of data to gauges. The gauges analyze the incoming data streams and recognize adverse conditions that need adaptation, relaying that information to controllers. The controllers coordinate the expression and orchestration of the workflow needed to carry out the adaptation. At the end of the loop, actuators attached to the target system effect the needed adjustments under the supervision of the controller.

In the AI^{2}TV case, sensors at each client monitor for currently displayed frame, its quality level, the quality level currently being fetched by the manager, the time range covered by buffer reserve frames, and the current bandwidth. Gauges are embedded together with the controller for expediency in design and to minimize communication latency. They receive the sensor reports from individual clients, collect them in buckets, similar to the approach in [20], and pass the bucket data structure to the controller's coordination engine. A set of helper functions tailored specifically for this application operate on this data structure and produce triggers for the coordinator. When a trigger is raised, the coordination engine enacts an adaptation scheme, basically a workflow plan, which is executed on the end hosts by taking advantage of the hooks provided to the actuators by the clients.

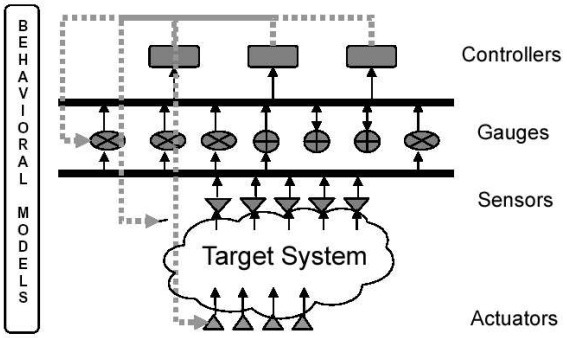

Figure 3: Conceptual Reference Architecture

Communication among the video clients, as well as between the sensors and actuators at the clients and the autonomic controller, is provided by a publish-subscribe event bus. There are three kinds of events: video player actions, sensor reports, and adaptation directives (see figure 2).

3.2 Adaptation Model

The adaptation scheme consists of two levels: a higher level data flow, and a lower level adjustment heuristic. The former directs the flow of data through a logical sequence to provide a formal decision process, while the latter provides the criteria as to when to make certain adjustments.

The higher level logic is shown in figure 4. The diagram shows the task decomposition hierarchy according to which the adaptation workflow unfolds. Note that the evaluation of clients' state with respect to the group (`EvaluateClient`) and the issuing of adaptation directives (`AdaptClient`) is carried out as a set of parallel steps. Also note that the multiplicity of those parallel steps is dynamically determined via the number of entries in the `clients` variable, which maps to a collection of AI^{2}TV clients.

The adaptation scheme at the lower level falls into two categories: directives that adjust the client in response to relatively low bandwidth situations, and those that take advantage of relatively high bandwidth situations.

In the situation where a client has relatively low bandwidth, below its baseline bandwidth level, the client may not be able download the next frame at the current quality level by the time it needs to begin displaying that frame. Then both the client and buffer quality levels are adjusted downwards one level. If the client is already at the lowest level (among those available from the video server), the controller will calculate the next possible frame that most likely can be successfully retrieved before its own start time while remaining synchronized with the rest of the group. The client will then be directed to jump ahead to that frame.

To take advantage of relatively high bandwidth situations, the buffer manager will start to accumulate a reserve buffer.

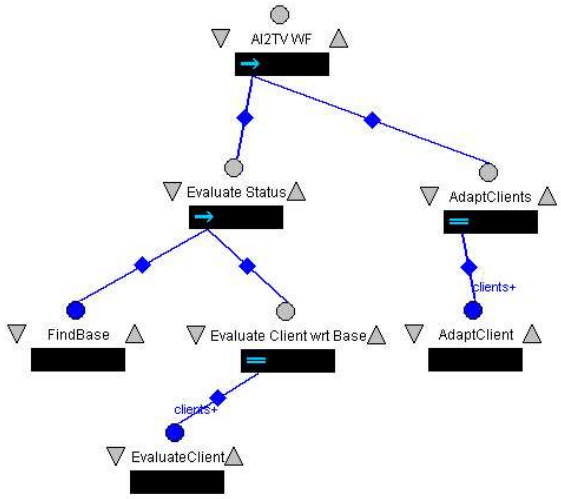

Figure 4: AI^{2}TV Workflow diagram

Once the buffer reaches a threshold value (e.g., 10 buffered frames), the controller will direct the manager to start fetching frames at a higher quality level. Once sufficient reserve is accumulated also at that higher level, the client is then ordered to display frames at that quality level. If the bandwidth drops before the buffer manager can accumulate enough frames in the higher-level reserve, the buffer manager is dropped back down one quality level.

3.3 Implementation

Our system is implemented in Java. The video client uses `javax.swing` to render JPEG images. The controller, Workflakes, is built on top of the open-source Cougaar multi-agent system [2], which it extends to allow the orchestration of distributed software agents for autonomic purposes (explained further in [43]). We used the Little-JIL graphical workflow specification language [13] for defining adaptation plans. We chose a freely available, content-based, publish-subscribe event system, Siena [12], as our communication bus.

4. EVALUATION

Our assessment considers the ability of AI^{2}TV to synchronize the clients and to optimally adjust their video quality. Our results were computed from client configurations consisting of 1, 2, 3, and 5 clients together running a semantically summarized video for 5 minutes, with sensors probing clients state every 5 seconds. The compression hierarchy we employed has 5 different quality levels.

We define a baseline client against which the performance of our approach can be compared. The baseline client's quality level is set at the beginning of the video and not changed thereafter, using a value we identify as the average bandwidth per level. This value is computed by summing the total size in bytes of all frames produced at a certain compression level and dividing by the total video time. This value provides the bandwidth needed, on average, for the buffer manager to download the next frame on time. We provide the baseline client with the corresponding bandwidth for its chosen level by using a bandwidth throttling tool ([39]) to adjust the bandwidth between that client and the video server. Note that using the average as the baseline does not account for the inherent variability in video frame rate and likely fluctuations in real-world network bandwidth, where adaptive control can make a difference.

Each controller-assisted client is assigned an initial level in the compression hierarchy and the same bandwidth as the baseline client for that hierarchy level. For each experimental trial, we record any differences resulting from the controller's adaptation of the clients' behavior *versus* the behavior of the baseline client, with respect to synchrony and frame rate.

4.1 Evaluating Synchronization

The primary goal of our system is to provide synchronous viewing of lecture videos to small groups of geographically dispersed students, some possibly with relatively meager resources. Our initial experiments evaluate the level of synchronization for several small groups of clients, where each group is involved in a video session. Each client is preset at a designated level of compression and given the average baseline bandwidth required for that compression level. To measure the effectiveness of the synchronization, we probe the video clients at periodic time intervals and log the frame currently being displayed. This procedure effectively takes a series of system snapshots, which we can evaluate for synchronization correctness. We check whether the frame being displayed at a certain time corresponds to one of the valid frames for that time, on *any* quality level. We allow an arbitrary level here because the semantic compression algorithm ensures that all frames designated for a given time will contain the semantically equivalent information. We obtain a score by summing the number of clients not showing an acceptable frame and normalizing over the total number of clients. A score of 0 indicates a fully synchronized system.

These experiments showed a total score of 0 for all trials, meaning that all of the clients were viewing appropriate frames when probed. Notwithstanding the variations in the frame rate and/or occasional fluctuations in the actual bandwidth of the clients, no frames were missed. This result demonstrates that the chosen baseline combinations of compression levels and throttled bandwidths do not push the clients beyond their bandwidth resource capacity.

Then we ran another set of experiments, in which the clients were assigned more casually selected levels of starting bandwidths. Said casual selection is representative of real-world situations, like listening to Internet radio, where users must choose a desired frame rate to receive. The user may have been informed that she is allocated a certain bandwidth level from her Internet service provider, but may actually be receiving a significantly lower rate. The clients were assigned bandwidths one level lower than the preset quality level. We ran this set of experiments first without the aid of the autonomic controller and then with it. In the former case, clients with insufficient bandwidth were stuck at the compression level originally selected, and thus missed an average of 63% of the needed frames. In the latter case, the same clients only missed 35% of the needed frames. Although both situations show a significant amount of missed frames, these results provide evidence of the benefits of the adaptive scheme implemented by the autonomic controller.

4.2 Evaluating Quality of Service

The most interesting technical innovation of the AI^{2}TV system is our autonomic controller approach to optimizing video quality. Here we analogously use a scoring system relative to the baseline client's quality level. We give a weighted score for each level above or below the baseline quality level. The weighted score is calculated as the ratio of the frame rate of the two levels. For example, if a client is able to play at one level higher then the baseline, and the baseline plays at an average `n` frames per second (fps) while the level higher plays at `2*n` fps, the score for playing at the higher level is 2. The weighted score is calculated between the computed average frame rates of the chosen quality levels. Theoretically, the baseline client should receive a score of 1. Note that we formulated this scoring system because other scoring systems (e.g., [8, 16, 45]) measure unrelated factors such as the synchronization between different streams (audio and video), image resolution, or human perceived quality, and are not constrained by the group synchronization requirement. This restriction mandates a scoring system sensitive to the relative differences between quality hierarchies.

Our experiments show that baseline clients scored a group score of 1 (as expected) while the controller-assisted clients scored a group score of 1.25. The one-tailed t-score of this difference is 3.01, which is significant for an α value of .005 (N=17). This result demonstrates that using the autonomic controller enabled our system to achieve a significant positive difference in received frame rate, or quality of services (QoS). Note that the t-score does not measure the degree of the positive difference: To demonstrate the degree of benefit, we measure the proportion of additional frames that each client is able to enjoy. We found that, overall, those clients received 20.4% ($\pm$ 9.7, N=17) more frames than clients operating at a baseline rate.

Running the client close to or at a level higher than the average bandwidth needed puts the client at risk for missing more frames, because the autonomic controller is trying to push the client to a better but more resource-demanding level. To measure whether the controller-assisted client is adversely exposed to a higher risk of missing frames, we also count the number of missed frames during a video session. The scoring is a simple count of the missed frames. Note that this scoring is kept separate from the measure of the relative quality to discriminate between levels of concern, although they both indicate QoS characteristics.

There was only one instance in which a controller-assisted client missed two consecutive frames. Upon closer inspection, the time region during this event showed that the semantically compressed video demanded a higher frame rate at the same time that the network bandwidth available to that client was relatively low. The client was able to consistently maintain a high video quality level after this epoch.

Our AI^{2}TV system can achieve collaborative video viewing using relatively naive NTP-based synchronization, without the autonomic controller. But in typical real-world scenarios, network bandwidth varies over time, plus the variable frame rate of semantically compressed video does not permit the client to make an informed decision about the most appropriate quality level for the next frames without adaptive technology akin to our controller. Our experimental data shows that the autonomic controller makes a significant positive difference in achieving higher QoS.

5. RELATED WORK

Stream synchronization is a widely studied topic in multimedia research. Classifications of synchronization schemes consider whether the scheme is local or distributed (i.e., one or multiple sinks), whether they take action reactively or proactively, and whether a global clock is required. Our work does not address the problem of inter-media synchronization of multiple modalities (i.e., video and audio), where the concern is to ensure the correctly timed playback of related data originating from different streams. Our problem is instead related to intra-stream synchronization, which is concerned with ensuring the temporal ordering of data packets transmitted across a network from a single streaming source to one or more delivery sinks.

Most intra-stream synchronization schemes are based on data buffering at the sink(s) and on the introduction of a delay before the play-out of buffered data packets (i.e., frames). Those synchronization schemes can be rigid or adaptive [15]. In rigid schemes, such as [19], the play-out delay is chosen *a priori* in such a way that it accounts for the maximum network transfer delay that can likely occur across the sinks. Rigid schemes work under a worst-case scenario assumption and accept the introduction of delays that may be longer than necessary, in order to maximize the synchronization guarantees they can offer even in demanding situations.

Contrary to a rigid approach, adaptive schemes [38, 10, 18] recompute the delay parameter continuously while streaming: they try to "guess" the minimum delay that can be introduced, which still ensuring synchronization under actual operating conditions. In order to enhance quality of service in terms of minimized play-out delay, those schemes must accept some temporary synchronization inconsistencies and/or some data loss, in case the computed delay results are at times insufficient (due, e.g., to variations in network conditions) and may need to be corrected on the fly.

Our approach to synchronization can be classified as a distributed adaptive scheme that employs a global clock and operates in a proactive way. The most significant difference compared to other approaches, such as the Adaptive Synchronization Protocol [38], the work of Gonzalez and Adbel-Wahab [21], or that of Liu and El Zarki[31] (which can all be used equally for inter- and intra-stream applications), is that our approach is not based on the idea of play-out delay. Instead, we take advantage of layered semantic compression coupled with buffering to "buy more time" for clients that might not otherwise be able to remain in sync, by putting them on a less demanding level of the compression hierarchy.

To ensure stream synchronization across a group of clients, it is usually necessary to implement some form of tradeoff impacting the quality of service of some of the clients. Many schemes trade off synchronization for longer delays, while some other approaches, like the Concord local synchronization algorithm [40], allow a choice among other quality parameters besides delay, such as packet loss rate. Our approach sacrifices frame rates to achieve synchronization when resources are low.

Liu *et al.* provide a comprehensive summary of the mechanisms used in video multicast for quality and fairness adaptation as well as network and coding requirements [32]. To frame our work in that context, our current design and implementation models a single-rate server adaptation scheme to each of the clients because the video quality we provide is tailored specifically to that client's network resources. The

focus in our work is directed towards the client-side end-user perceived quality and synchrony, so we did not utilize the most efficient server model. The authors believe that it would be trivial to substitute in a simulcast server adaptation model [14, 29]. Our design also fits into the category of layered adaptation. Such an adaptation model defines a base quality level that users must achieve. Once users have acquired that level, the algorithm attempts to incrementally acquire more frames to present a higher quality video. In the work presented here, the definition of quality translates to a higher frame rate. Liu's discussion of bandwidth fairness, coding techniques and network transport perspectives lie out of the scope of this paper.

With respect to the software architecture, our approach most resembles the Lancaster Orchestration Service [10], since it is based on a central controller that coordinates the behavior of remote controlled units placed within the clients via appropriate directives (i.e., the AI^{2}TV video buffer and manager). The Lancaster approach employs the adaptive delay-based scheme described above, hence the playback of video focuses on adapting to the lowest bandwidth client. That approach would degrade the playback experience of the other participants to accommodate the lowest bandwidth client. Our approach seems preferable, since it enables each client to receive video quality commensurate with its bandwidth resources.

Cen *et al.* provide a distributed real-time MPEG player that uses a software feedback loop between a single server and a single client to adjust frame rates [44]. Their architecture incorporates feedback logic within each video player and does not support synchronization across a group of players, while the work presented here explicitly supports the synchronization of semantically equivalent video frames across a small group of clients.

An earlier implementation of AI^{2}TV is described in [22]. In that version, a collaborative virtual environment (CVE) supported a variety of team interactions [17], with the optional lecture video display embedded in the wall of a CVE "room". The same semantic compression capability was used. Video synchronization data was piggybacked on top of the UDP peer-to-peer communication used primarily for CVE updates, such as tracking avatar movements in the style of multi-player 3D gaming. The video synchronization did not work very well, due to the heavy-weight CVE burden on local resources. Video quality optimization was not addressed. The new implementation of AI^{2}TV presented here can run alongside the CVE in a separate window.

6. CONCLUSION

We present an architecture and prototype system that allows geographically dispersed student groups to collaboratively view lecture videos in synchrony. AI^{2}TV employs an "autonomic" (feedback loop) controller to dynamically adapt the video quality according to each client's network bandwidth. We rely on a semantic compression algorithm to guarantee that the semantic composition of the simultaneously viewed video frames is equivalent for all clients, some which may have very limited resources. Our system distributes appropriate quality levels (different compression levels) of the video to clients, which are automatically adjusted according to their current and fluctuating bandwidth resources. We have demonstrated the advantages of this approach through experimental trials using bandwidth throttling to show that our system can provide synchronization of video together with optimized video quality.

7. ACKNOWLEDGMENTS

We would like to thank John Kender, Tiecheng Liu, and other members of the High-Level Vision Lab for their assistance in using their lecture-video semantic compression software. We would also like to thank the other members of the Programming Systems Lab, particularly Matias Pelenur who implemented PSL's Little-JIL interpreter on top of Workflakes/Cougaar. Little-JIL was developed by Lee Osterweil's LASER lab at the University of Massachusetts, Amherst. Cougaar was developed by a DARPA-funded consortium; our main Cougaar contact was Nathan Combs of BBN. Siena was developed by the University of Colorado, Boulder, in Alex Wolf's SERL lab. PSL is funded in part by National Science Foundation grants CCR-0203876, EIA-0202063 and EIA-0071954, and by Microsoft Research.

8. ADDITIONAL AUTHORS

Additional authors: Suhit Gupta `suhit@columbia.cs.edu`

9. REFERENCES

[1] Capella University: Education. Reborn. `http://www.capella.edu/`.

[2] Cognitive Agent Architecture (Cougaar) Open Source Project. `http://www.cougaar.org/`.

[3] University of Phoenix: University for Working Adults. `http://www.phoenix.edu/`.

[4] Vnc (virtual network computing). `http://www.realvnc.com/`.

[5] Webex: Web conferencing, video conferencing and online meeting services. `http://www.webex.com/`.

[6] Welcome To The American School. `http://www.americanschoolofcorr.com/`.

[7] The Application and Implications of Information Technologies in Postsecondary Distance Education: An Initial Bibliography. Technical Report NSF 03-305, National Science Foundation, Division of Science Resources Statistics, December 2002.

[8] S. Baqai, M. F. Khan, M. Woo, S. Shinkai, A. A. Khokhar, and A. Ghafoor. Quality-based evaluation of multimedia synchronization protocols for distributed multimedia information systems. *IEEE Journal of Selected Areas in Communications*, 14(7):1388–1403, 1996.

[9] L. A. Burgess and S. D. Strong. Trends in online education: Case study at southwest missouri state university. *Journal of Industrial Teacher Education*, 19(3), 2003.

[10] A. Campell, G. Coulson, F. Garcia, and D. Hutchison. A continuous media transport and orchestration service. In *SIGCOMM92: Communications Architectures and Protocols*, pages 99–110, 1992.

[11] M. Capps, B. Laddi, D. Stotts, and L. Nyland. Educational applications of multi-client synchronization through improved web graph semantics. In *5th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1996.

[12] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, Aug. 2001.

[13] A. G. Cass, Barbara Staudt Lerner, E. K. McCall, L. J. Osterweil, Stanley M. Sutton, Jr., and A. Wise. Little-JIL/Juliette: A Process Definition Language and Interpreter. In *22nd International Conference on Software Engineering*, pages 754–757, June 2000.

[14] S. Y. Cheung, M. H. Ammar, and X. Li. On the use of destination set grouping to improve fairness in multicast video distribution. In *Proceedings IEEE INFOCOM '96*, pages 553–560, 1996.

[15] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *ACM SIGCOMM: Communications architectures and protocols*, pages 14–26, 1992.

[16] A. L. Corte, A. Lombardo, S. Palazzo, and G. Schembra. Control of perceived quality of service in multimedia retrieval services: Prediction-based mechanism vs. compensation buffers. *Multimedia Systems*, 6(2):102–112, 1998.

[17] S. E. Dossick and G. E. Kaiser. CHIME: A Metadata-Based Distributed Software Development Environment. In *Joint 7th European Software Engineering Conference and 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, pages 464–475, 1999.

[18] J. Escobar, C. Partridge, and D. Deutsch. Flow synchronization protocol. *IEEE Transactions on Networking*, 1994.

[19] D. Ferrari. Design and application of a delay jitter control scheme for packet-switching internetworks. In *2nd International Conference on Network and Operating System Support for Digital Audio and Video*, pages 72–83, 1991.

[20] L. Gautier and C. Diot. Design and evaluation of mimaze, a multi-player game on the internet. In *International Conference on Multimedia Computing and Systems*, pages 233–236, 1998.

[21] A. J. Gonzalez and H. Adbel-Wahab. Lightweight stream synchronization framework for multimedia collaborative applications. In *5th IEEE Symposium on Computers and Communications*, July 2000.

[22] S. Gupta and G. Kaiser. A Virtual Environment for Collaborative Distance Learning With Video Synchronization. In *7th IASTED International Conference on Computers and Advanced Technology in Education*, August 2004.

[23] IBM. Autonomic Computing. `http://www.research.ibm.com/autonomic/`.

[24] G. Kaiser, P. Gross, G. Kc, J. Parekh, and G. Valetto. An Approach to Autonomizing Legacy Systems, in Workshop on Self-Healing, Adaptive and Self-Managed Systems. In *Workshop on Self-Healing, Adaptive and Self-Managed Systems*, June 2002.

[25] G. Kaiser, J. Parekh, P. Gross, and G. Valetto. Kinesthetics eXtreme: An External Infrastructure for Monitoring Distributed Legacy Systems. In *5th Annual International Active Middleware Workshop*, June 2003.

[26] G. Kaiser, J. Parekh, P. Gross, and G. Valetto. Retrofitting Autonomic Capabilities onto Legacy Systems. Technical Report CUCS-026-03, Columbia University Department of Computer Science, October 2003.

[27] B. S. Lemer, E. K. McCall, A. Wise, A. G. Cass, L. J. Osterweil, and S. M. S. Jr. Using little-jil to coordinate agents in software engineering. In *Automated Software Engineering Conference*, September 2000.

[28] W. Li. Overview of the fine granularity scalability in mpeg-4 video standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):301–317, March 2001.

[29] X. Li, M. H. Ammar, and S. Paul. Video multicast over the internet, April 1999.

[30] H. Lieberman, N. V. Dyke, and A. Vivacqua. Let's browse: A collaborative web browsing agent. In *International Conference on Intelligent User Interfaces*, 1999.

[31] H. Liu and M. E. Zarki. A synchronization control scheme for real-time streaming multimedia applications. In *Packet Video 2003*, April 2003.

[32] J. Liu, B. Li, and Y.-Q. Zhang. Adaptive video multicast over the internet. *IEEE Multimedia*, 10(1):22–33, January/March 2003.

[33] T. Liu and J. R. Kender. Time-constrained dynamic semantic compression for video indexing and interactive searching. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 531–538, 2001.

[34] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM SIGCOMM*, volume 26,4, pages 117–130, New York, AUG 1996. ACM Press.

[35] J. Miller, C. Ditzler, and J. Lamb. Reviving a Print-based Correspondence Study Program In the Wake of Online Education. In *American Association for Collegiate Independent Study: Distance Learning: Pioneering the Future*, October 2003.

[36] D. L. Mills. Network time protocol. RFC 958, 1985.

[37] M. Richtel. In a Fast-Moving Web World, Some Prefer the Dial-Up Lane. *The New York Times*, April 19 2004.

[38] K. Rothermel and T. Helbig. An Adaptive Protocol for Synchronizing Media Streams. *Multimedia Systems*, 5:324–336, 1997.

[39] L. Santi. A user-mode traffic shaper for tcp-ip networks. `http://freshmeat.net/projects/shaperd/`.

[40] N. Shivakumar, C. J. Sreenan, B. Narendran, and P. Agrawal. The concord algorithm for synchronization of networked multimedia streams. In *International Conference on Multimedia Computing and Systems*, pages 31–40, 1995.

[41] G. Sidler, A. Scott, and H. Wolf. Collaborative browsing in the world wide web. In *8th Joint European Networking Conference Proceedings*, 1997.

[42] G. Valetto. *Orchestrating the Dynamic Adaptation of Distributed Software with Process Technology*. PhD thesis, Columbia University, April 2004.

[43] G. Valetto and G. Kaiser. Using Process Technology to Control and Coordinate Software Adaptation. In *International Conference on Software Engineering*, May 2003.

[44] J. Walpole, R. Koster, S. Cen, C. Cowan, D. Maier, D. McNamee, C. Pu, D. Steere, and L. Yu. A Player for Adaptive MPEG Video Streaming Over The Internet. In *26th Applied Imagery Pattern Recognition Workshop*. SPIE, October 1997.

[45] Y. Wang, J. Ostermann, and Y.-Q. Zhang. *Video Processing and Communications*. Prentice Hall, 2002.

[46] J. G. Wells. Effects of an on-line computer-mediated communication course. *Journal of Industrial Technology*, 37(3), 2000.