Adaptive Synchronization of Semantically Compressed Instructional Videos for Collaborative Distance Learning

Dan Phung[1], Giuseppe Valetto[3], Gail E. Kaiser[1], Tiecheng Liu[2] and John R. Kender[1]

[1]Columbia University, New York, NY, USA
[2]University of South Carolina, Columbia, SC, USA
[3]IBM T.J. Watson Research Center, Hawthorne, NY, USA

Abstract

The increasing popularity of online courses has highlighted the need for collaborative learning 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 available to students. We present an e-Learning architecture and adaptation model called AI^{2}TV (Adaptive Interactive Internet Team Video), which allows groups of students to collaboratively view a video in synchrony. AI^{2}TV upholds the invariant that each student will view semantically equivalent content at all times. A semantic compression model is developed to provide instructional videos at different level-of-details to accommodate dynamic network conditions and users' requirements; video player actions, like play, pause and stop, can be initiated by any group member. These features allow students to review a lecture video in tandem, facilitating the learning process. Experimental trials show that AI^{2}TV successfully synchronizes instructional videos for distributed students while, at the same time, optimizing the video quality, even under conditions of fluctuating bandwidth, by adaptively adjusting the quality level for each student while still maintaining the invariant.

1 Introduction

Distance learning programs such as the Columbia Video Network (www.cvn.columbia.edu) have evolved from fedexing lecture video tapes to their off-campus students to streaming videos over the Web. The lectures might be delivered "live", but are more frequently post-processed and packaged for students to watch (and re-watch) at their convenience. This introduces the possibility of forming "study groups" among students who can view the lecture videos together and

pause, rewind or fast-forward the video to discussion points, thus approximating the pedagogically valuable discussions that occur during on-campus lectures. However, conventional Internet-video technology does not yet support *collaborative video viewing* by multiple geographically dispersed users. It is particularly challenging to support WISIWYS (What I See Is What You See) when some users are relatively disadvantaged with respect to bandwidth (e.g., dial-up modems) and local resources (e.g., old graphics cards, small disks).

AI^{2}TV (Adaptive Interactive Internet Team Video) is an e-Learning architecture supporting virtual student groups. To that end, we have developed the technology for "semantically adapting" standard MPEG videos into sequences of still 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. It was designed with typical lecture videos in mind: for instance, it recognizes that it is more important to see the blackboard content after the instructor has finished writing, than showing the instructor's back as she writes it on the board.

The other technical challenges are *synchronizing* and *adapting* the downloading and display of the image sequences among the distributed students, including support for shared video player actions. We have developed an approach that achieves this using three mechanisms working in tandem: First, the software clocks of the video clients for each student are synchronized using NTP, hence they use the same time reference with respect to 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 any user to all of the group, as well as other events occurring on the video clients. Finally, since we are particularly concerned about disenfranchised user communities that have relatively low bandwidth, the final contribution of AI^{2}TV concerns enabling the optimization of the video quality according to the bandwidth constraints of each user, while enforcing group synchronization, through a distributed feedback control loop that dynamically adapts each video client.

This paper presents the architecture and dynamic adaptation model of AI^{2}TV, describes how it tackles the challenges of quality optimization and synchronization in collaborative video viewing, and provides an evaluation of the effectiveness of our approach, with empirical results obtained using real lecture videos from Columbia's Video Network.

2 Motivation and Background

Correspondence courses have been available for over a century, e.g., the University of Wyoming began offering extension courses in 1892 [1], Correspondence courses have traditionally been designed for individual students with a self-motivated learning style, studying primarily from text materials.

An NSF Report [2] discusses how technology, from radio to television, to audio and video cassettes, to audio and video conferencing, has affected distance education. 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".

Even though some Internet–based tools, like instant messaging, desktop sharing and co-browsing can be used to facilitate the communicative aspects of synchronous collaboration, dedicated support for synchronous collaboration in long–distance education over the Web remains a major concern in courses where group work is encouraged [3], since there are few educational tools that offer that kind of support to a group of online students [4]. 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 class discussion.

Our AI^{2}TV project contributes to synchronous collaboration support for life-long and distance education, and specifically to the problem of collaborative video viewing, to foster virtual classrooms and borderless education. Our design is intended for small classes or study groups within a larger class, and reaches out to disenfranchised users with relatively low bandwidths, who constitute a significant portion of the Internet user community [5], to allow collaboration with other users who enjoy higher bandwidth. Since it is likely that future bandwidth improvements will also be unevenly available, our architecture will remain effective even when the disparity is between bandwidths that all have a higher absolute level of performance than today's.

Collaborative video viewing poses a twofold technical challenge: on the one hand, all users **must** be kept synchronized with respect to the content they are supposed to see at any moment during play time; on the other hand, each individual user **should** be provided with a level of quality that is optimized with respect to her available resources, which may vary during the course of the video.

One way to address the problem of balancing the group synchronization requirement with the optimization of individual viewing experiences is to use videos with cumulative layering [6], also known as scalable coding [7]. In this approach, the client video player selects a quality level appropriate for 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. Our semantic compression algorithm [8] reduces a video to a set of semantically significant key frames. That tool operates on conventional MPEG videos and outputs sequences of JPEG frames. The semantic compression algorithm profiles video frames within a sliding time window and selects in that window key frames that have the most significant instructional information.

A conceptual diagram of a layered video produced from this semantic compression is shown in Figure 1. While a detailed discussion of that video compression algorithm and system is available in Section 4, it is interesting to note at this point that the semantic compression algorithm produces key frames based on semantic content of instructional videos: when there are pockets of relatively high frequency semantic change, i.e., more key frames are produced. Therefore, the resulting video plays back at a *variable* frame rate, which adds substantial complexity to the bandwidth demands of the client.

The bottom-left in-set in Figure 1 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, start and end times of frame sets rarely match up that precisely, and the determination of the optimal frame to semantically represent a given frame set remains a research issue.

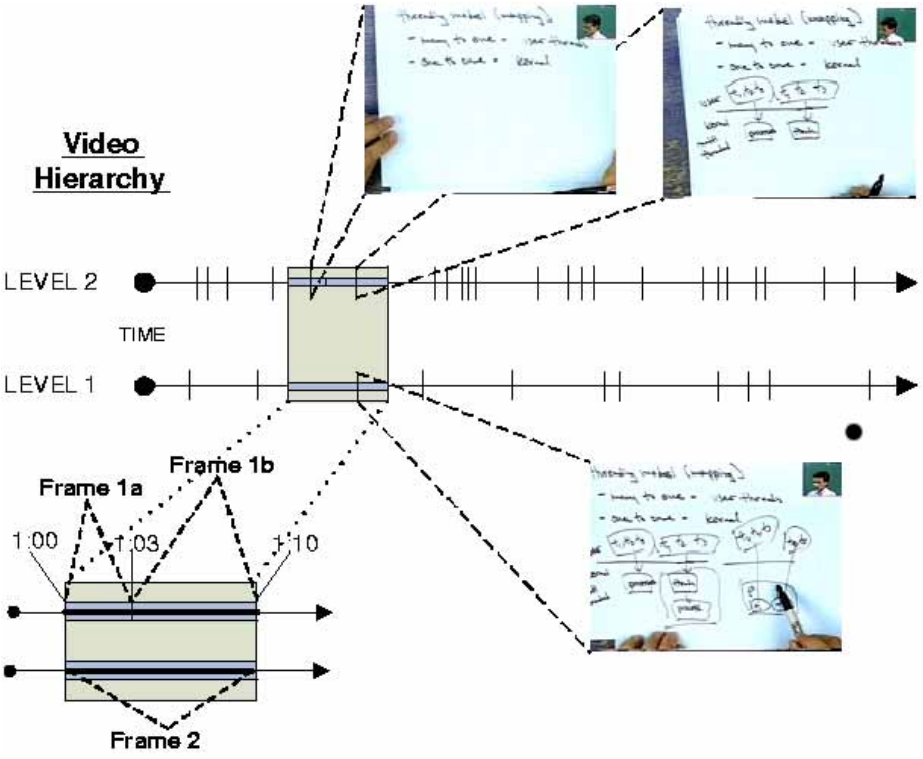

Figure 1: Semantic Video Scenario

To take advantage of the semantic instructional video compression algorithm in order to provide semantically equivalent content to a group of students with diverse resources, but still provide the best quality video possible at any given moment, we dynamically adjust the compression level assigned to each client while watching the video. Thus, for our purposes, synchronization of collaborative video boils down to showing semantically equivalent frames at all times. To adjust the video 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 quality seen by each student.

This controller remains conceptually separate from the controlled video system, and employs our decentralized workflow engine, named Workflakes [9]. 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 *et al.* [10]). Workflakes has also been used in a variety of more conventional "autonomic computing" domains, where it orchestrates the work of software actuators to achieve the automated dynamic adaptation of distributed applications [11, 12]. In 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 System Architecture

AI^{2}TV includes a video server, several video clients, an autonomic controller, and a common communications infrastructure (the event bus), as shown in Figure 2.

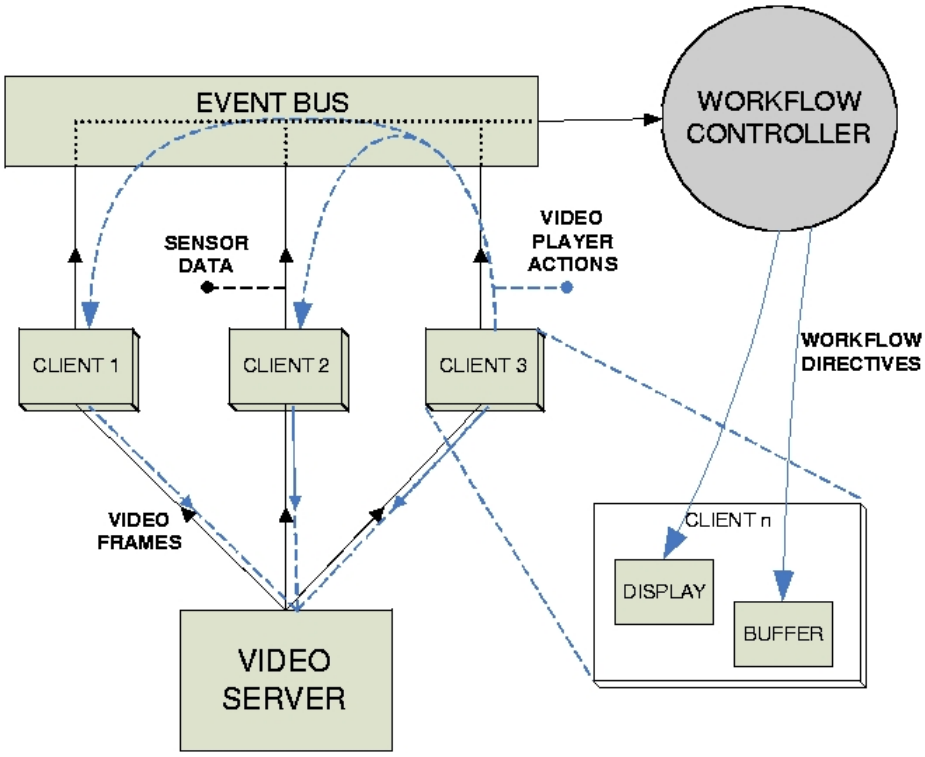

Figure 2: AI^{2}TV Architecture.

The video server provides the educational video content to the clients. Each lecture video is stored in the form of a hierarchy of versions, produced by running the semantic compression tool with different settings. 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, a video buffer that feeds the display, and a manager for fetching frames.

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. 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 via the event bus, 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 uses the current video time and display quality level to index into the frame index for the frame to be displayed. Before trying to render the needed frame, it asks the video buffer manager if it is available. The video display also includes an actuator that enables 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 an actuator that enables external entities, such as the controller, to adjust the current downloading quality level.

The main purpose of the autonomic controller is to ensure that, given the synchronization constraint, each client plays at its highest attainable quality level. The architecture provides an end-to-end closed control loop, in which sensors attached to the 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 the controller. The controller coordinates 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 the 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 [13], 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 coordination engine. When a trigger is raised, it enacts a workflow plan, which is executed on the end hosts by taking advantage of actuators embedded in the clients.

Communication among the video clients, as well as between the sensors and actuators at the clients and the autonomic controller, is provided by an asynchronous event bus that channels video player actions, sensor reports, and adaptation directives.

4 Semantic Compression of Instructional Videos

Effective video compression model for collaborative distance learning environments is required to provide multiple different "versions" of videos, for students that may access instructional content using devices that differ significantly in resolution, computing capability, storage, and available network resources. Several additional requirements exist in this kind of video compression process: First, the compression should be content-based, by analyzing instructional content; Second, the semantic video compression should produce different levels of content summarization, with each level showing different details of video content; Third, the compression should be dynamic so that it can also handle live instructional videos recorded in classrooms, enabling real-time multicasting of instructional videos; Fourth, the compression model should be computationally efficient.

With the consideration of the above requirements, we have developed a semantic video compression model for real-time adaptation of instructional videos. The compression ratio is tunable, thus different levels of compression can be achieved. Each layer contains semantic information of the instructional video at a certain level of detail. Since this model is specially designed for instructional videos with emphasis on the characteristics of this video domain, it is different from previous work on signal-level video compression. The scalable coding scheme adopted in MPEG-4 compression standard [14] is still based on

signal-level video compression. Previous video key-frame-based compression and summarization methods [15, 16, 17, 18, 19] are tuned for professionally-edited videos such as movies, sports and news videos. By comparison, instructional videos are (mostly) unedited videos without salient video structures like shots and scenes, and the visual content is mostly embedded in handwritten or printed characters/figures.

The model of semantic compression of instructional videos is illustrated in Figure 4. Basically, it is a dynamic video buffer in which video frames are evaluated and comparatively insignificant video frames are dropped. For a video buffer of n slots (slots $S_1, S_2, \ldots, S_n$), the video stream comes into S_1, moves through the buffer slots $S_2, S_3, \ldots, S_n$ in order, partially leaks from the buffer at any slot, and finally goes out of the buffer at slot S_n. The surviving outgoing frames form a semantically compressed version of the instructional video.

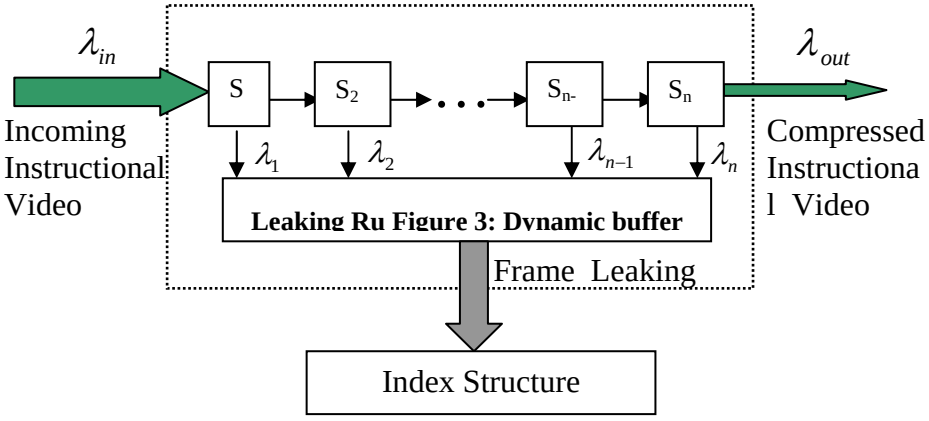

Figure 4: Dynamic buffer model for semantic compression of instructional videos.

For each video frame coming into the video buffer, a content analysis module first extracts its instructional content. Here the instructional video content refers to the textual content on board, paper and slides. To extract the text area from video frames efficiently, we use a block-based processing approach. We divide each frame into blocks of size of 16 by 16 and classify these blocks into three categories: paper/board blocks, irrelevant blocks and uncertain blocks, based on the portion of paper background color pixels in a block. Paper blocks have only paper and/or text, and irrelevant blocks have images of the instructor and other non-content areas. Uncertain blocks are those that fail this initial coarse filter, and they are further processed by more sophisticated and expensive techniques, i.e., using region merging and growing techniques

to remove holes in a region. As shown in Figure 5 , the written content is extracted, and the irrelevant regions are removed.

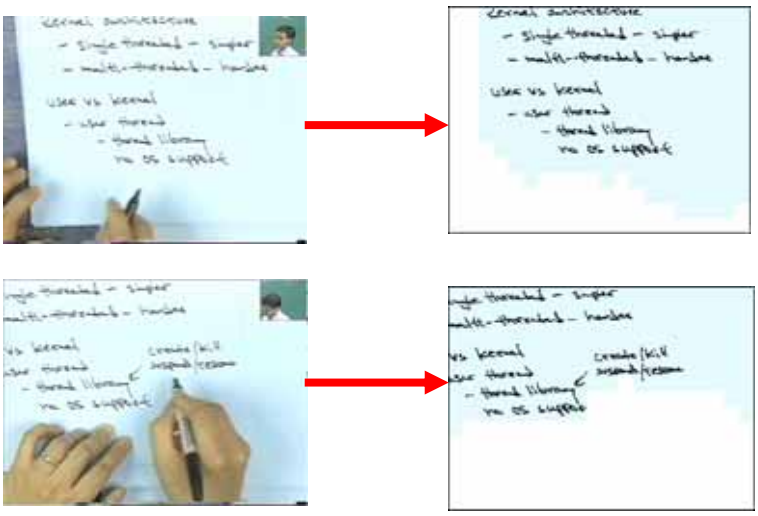

Figure 5: Content analysis of instructional video frames.

Based on the content analysis of instructional videos, we define the semantic distance between two adjacent video frames as the amount of different textual content pixels, and apply a leaking rule to the video to achieve semantic video compression. In compressing instructional videos, the frames in the video buffer are dropped at a leaking rate determined by the incoming frame rate (set by the video source or server) and the output frame rate (set by the available bandwidth to the client). The leaking process selects which frame in the video buffer to drop, and removes it from the queue. The queue, implemented as a doubly-linked list, then shifts all the frames in lowered-numbered slots to their successor buffer slots, freeing up the first slot again.

In a n-slot (slots $S_1, S_2, \ldots, S_n$) video buffer, let $d_{i,j}$ be the distance from frame f_i to frame f_j. We first find the minimum of all distances between adjacent frames in the video buffer. Suppose the minimum distance is $d_{k,k+1}$, i.e., $d_{k,k+1} = \min_i\{d_{i,i+1}\}$. The frame f_k and f_{k+1} have the minimum distance in the buffer, indicating that the content of these two frames are the most similar among all frame pairs in the buffer. Thus we may choose to delete one of f_k and f_{k+1} to reduce the content redundancy in the buffer. The decision of deleting which one comes from the test: Suppose frame f_k and f_{k+1} are adjacent to frame f_{k-1} and

f_{k+2} , respectively. If $\min\{d_{k-1,k+1}, d_{k+1,k+2}\} > \min\{d_{k-1,k}, d_{k,k+2}\}$, we delete frame f_k in the buffer, otherwise we delete frame f_{k+1}. This test measures the impact of the deletion, so we always remove the frame that makes the remaining frames less redundant in the video buffer. This leaking rule computes the effect that dropping either frame f_k or f_{k+1} has on their temporal context, and chooses to drop the frame that accentuates the semantic individuality of the neighboring frames of the evolving compressed video stream. This "Min-Min" leaking rule appears to maintain larger semantic differences in the buffer. This compression model is applicable to both pre-recorded videos and live video signals, and the leaking activities can be recorded in a data structure, which is further used for indexing and retrieval applications. By choosing a different output frame rate λ_{out} , we can compress an instructional video into different layers with different level-of-details.

5 Adaptation Model

The adaptation scheme we adopt 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 6. 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.

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. When a client has (relatively) low bandwidth, it may not be able to 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 calculates 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 is then directed to

jump ahead to that frame. When a client has instead (relatively) high bandwidth, the buffer manager starts to accumulate a reserve buffer. Once the buffer reaches a threshold value (e.g., 10 buffered frames), the controller directs the manager to start fetching frames at a higher quality level. Once sufficient reserve is accumulated 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 drops back down one quality level.

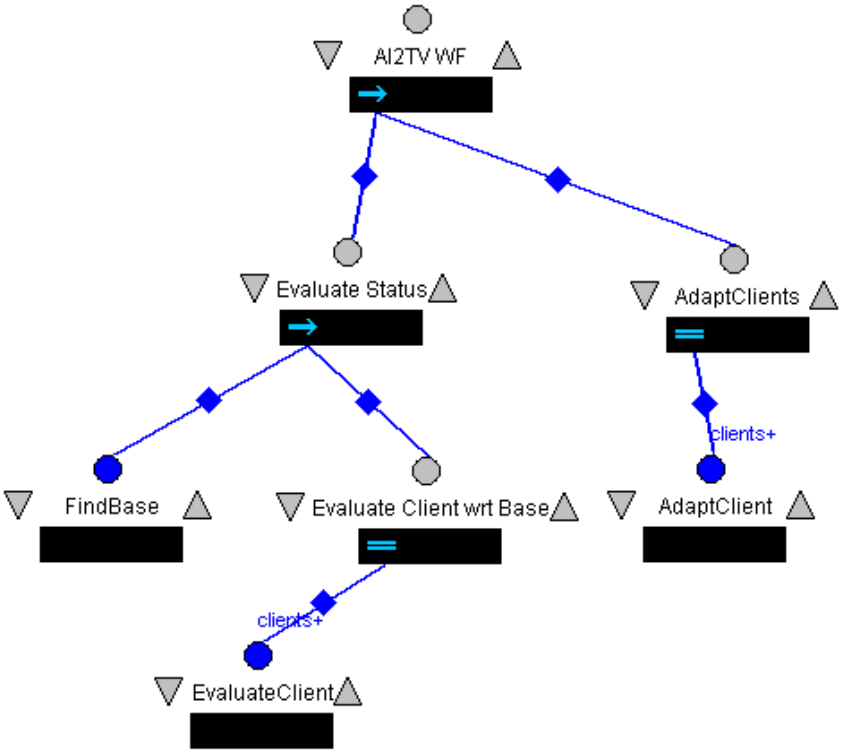

Figure 6: AI^{2}TV Workflow diagram.

6 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 simulating small study groups which consisted 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 quality levels.

We define a baseline client against which the performance of our approach is compared. The average bandwidth per level is computed, by summing the size in bytes of all frames produced at a certain compression level and dividing by the total video time. The baseline client's quality level is static for the duration of the video. We provide the baseline client with the corresponding bandwidth to the video server for its chosen level by using a bandwidth throttling tool. 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 *vs.* the behavior of the baseline client, with respect to synchrony and frame rate.

6.1 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 involved in a video session. Each client is preset at a designated level of compression and given the average baseline bandwidth required to sustain 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 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, such as receiving Internet audio/video streams, where users must choose a desired frame rate for the transmission of the content. 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 fraction of missed frames, these results provide evidence of the benefits of the adaptive scheme implemented by the autonomic controller.

This data shows how in typical real-world scenarios, in which network bandwidth fluctuations and the variable video frame rate do not permit an informed decision about the most appropriate quality level, the adaptive technology of our autonomic controller makes a significant positive difference.

6.2 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., [20, 21, 22]) 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 the quality of service (QoS) aspect that relates to received frame rate. 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 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 evaluate that risk, we also count the number of missed frames during a video session, which is intended as a separate measure of QoS characteristic with respect to the measure of relative quality described above. In all of our experiments, there was one single instance in which a controller-assisted client missed some frames: in particular it missed two consecutive frames in a time region of the semantically compressed video that demanded a higher frame rate, while at the same time the fluctuating bandwidth available to that client was relatively low.

7 Related Work

Yin *et al.* [23] provide an adaptive multimedia distribution system based on streaming, multicast and compression technology. They show that they can improve the level of QoS, but do not discuss user-level action synchronization, and use quality degradation rather than semantic compression to adapt to client resource constraints. Walpole *et al.* [24] 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. Their architecture incorporates feedback logic to each video player, which does not support group synchronization, while the work presented here explicitly supports the synchronization of (semantically equivalent) video frames across a small group of clients.

An earlier approach to AI^{2}TV is described in [25]. In that version, a collaborative virtual environment (CVE) supported a variety of team interactions [26], with the optional lecture video display embedded in the wall of a CVE "room". Video synchronization data was piggybacked on top of the UDP peer-to-peer communication used primarily for CVE updates, which did not work very well due to the heavy-weight CVE burden on local resources.

Our approach to synchronization can be classified as a distributed adaptive scheme that employs a global clock and operates proactively. The main difference compared to other approaches, such as the Adaptive Synchronization Protocol [27], the work of Gonzalez and Adbel-Wahab [28], or that of Liu and El Zarki [29], is that our approach is not based on 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. Liou *et al.* [30] develop a system for synchronizing videos, but that work provides no scalable videos and instructional video analysis.

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 [31]. Our work can be framed in that context as 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.

Instructional video compression, adaptation and indexing are crucial for distance learning. Previous key frame selection methods [32, 33] are based on video segmentation, frame clustering, or some hybrid of these two [34]. In general, segmentation-based key frame selection methods choose one or more representative frames for each segmented video structural unit as key frames; clustering-based key frame selection methods classify frames of the original video sequence into clusters, then choose one key frame from each frame cluster.

However, video indexing and summarization methods based on video segmentation [15, 16, 17] appear tuned to highly structured and professionally-edited commercial products. For instructional videos, segmentation-based key frame selection is no longer appropriate because there are no salient structural units. Video indexing methods based on clustering [18, 19] avoid segmentation preprocessing; however, most video key frame clustering methods highly depend on thresholds that determine the size of cluster, the number of key frames, or the level of key frames in a key frame hierarchy. Since these thresholds vary

greatly among different video genres or even within the same video genre, they are difficult to choose. Furthermore, most clustering-based methods are expensive in time and storage.

8 Conclusion

We present an e-Learning architecture and prototype system that allows small, geographically dispersed student groups to collaboratively view lecture videos in synchrony. To accommodate disenfranchised users with relatively low-bandwidth, AI^{2}TV employs an "autonomic" (feedback loop) controller to dynamically adapt the video quality according to each client's network resources, aiming to provide the best quality attainable while remaining in synchrony. We rely on a semantic compression algorithm to guarantee that the semantic composition of the simultaneously viewed video frames is equivalent for all clients. Our system distributes appropriate quality levels of video to clients, and automatically adjusts them according to their current 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 real-world distance learning lecture videos together with optimized video quality to distributed student groups.

Acknowledgments

We would like to thank other members of the High-Level Vision Lab and of the Programming Systems Lab, particularly Matias Pelenur and Suhit Gupta. 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 was funded in part by National Science Foundation grants CNS-0426623, CCR-0203876, EIA-0202063, CCR-9970790 and in part by IBM and Microsoft Research. PSL and HLV were funded jointly by EIA-0071954.

References

1. Miller, J., Ditzler, C., Lamb, J., *Reviving a Print-based Correspondence Study Program In the Wake of Online Education*, American Association for Collegiate Independent Study: Distance Learning: Pioneering the Future, (2003)
2. *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 (2002)

3. Wells, J.G, *Effects of an On-line Computer-mediated Communication Course*, Journal of Industrial Technology, 37(3), 2000, http://scholar.lib.vt.edu/ejournals/JITE/v37n3/wells.html

4. Burgess, L.A., Strong, S.D., *Trends in online education: Case study at Southwest Missouri State University*, Journal of Industrial Teacher Education, **19(3)**, May 2003, http://www.nait.org/jit/Articles/burgess041403.pdf

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

6. McCanne, S., Jacobson, V., Vetterli, M., *Receiver-driven Layered Multicast*, in: ACM SIGCOMM. 26(4):117-130., New York, NY, USA, 1996.

7. Li, W., *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.

8. Liu, T., Kender, J.R., *Time-constrained dynamic semantic compression for video indexing and interactive searching*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2001).

9. Valetto, G., *Orchestrating the Dynamic Adaptation of Distributed Software with Process Technology*, Ph.D thesis, Columbia University, Technical Report CUCS-022-04, New York, NY, USA, May 2004, http://mice.cs.columbia.edu/getTechreport.php?techreportID=82&format=pdf&

10. Lemer, B.S., McCall, E.K., Wise, A., Cass, A.G., Osterweil, L.J., Sutton,.S.M Jr., *Using Little-JIL to Coordinate Agents in Software Engineering*, in Proceedings of the Automated Software Engineering Conference.(ASE 2000), Grenoble, France, September 11-15, 2000.

11. Valetto, G., Kaiser, G., *Using Process Technology to Control and Coordinate Software Adaptation*, in Proceedings of the 25th International Conference on Software Engineering. (ICSE 2003), Portland, Or., USA, May 2003.

12. Parekh, J., Kaiser, G., Gross, P., Valetto, G., *Retrofitting Autonomic Capabilities onto Legacy Systems*, Journal of Cluster Computing, Kluwer (in press)

13. Gautier, L., Diot, C., *Design and Evaluation of MiMaze, a Multi-player Game on the Internet*, in Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Austin, Tx, USA, June 1998

14. Koenen, R. (ed), *Overview of the MPEG-4 Standard*, ISO/IEC JTC1/SC29/WG11 N4668, March 2002, http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm

15. Ardizzone, E., Hacid, M., *A Semantic Modeling Approach for Video Retrieval by Content*, in Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'99), Florence, Italy, June 7-11, 1999.

16. Chang, H.S., Sull, S., Lee, S.U., *Efficient Video Indexing Scheme for Content-based Retrieval*, IEEE Transactions on Circuits and Systems for Video Technology, 9(8): 1269-1279, December 1999

17. Smith, M., Kanade, T., *Video Skimming and Characterization through the Combination of Image and Language Understanding Techniques*, in Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'97), San Juan, Puerto Rico, June 17-19, 1997

18. Zhuang, Y., Rui, Y., Huang, T.S., Mehrotra, S., *Adaptive Key Frame Extraction Using Unsupervised Clustering*, in Proceedings of the IEEE International Conference on Image Processing, Chicago, Il., USA, October 4-7, 1998.

19. Girgensohn, A., Boreczky, J., *Time-Constrained Keyframe Selection Technique*, in Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'99), Florence, Italy, June 7-11, 1999

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

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

22. Wang, Y., Ostermann, J., Zhang, Y.Q., *Video Processing and Communications*, Prentice Hall, 2002, ISBN 0-13-017547-1.

23. Yin, H., Lin, C., Zhuang, J.J., Ni, Q., *An adaptive distance learning system based on media streaming*, in Proceedings of the 3rd International Conference on Web-Based Learning. (ICWL 2004), Beijing, China, August 8-11, 2004.

24. Walpole, J., Koster, R., Cen, S., Cowan, C., Maier, D., McNamee, D., Pu, C., Steere, D., Yu, L., *A Player for Adaptive MPEG Video Streaming Over The Internet*, in Proceedings of the 26th Applied Imagery Pattern Recognition Workshop, Washington DC, USA, October 15-17, 1997.

25. Gupta, S., Kaiser, G., *A Virtual Environment for Collaborative Distance Learning With Video Synchronization*, in Proceedings of the 7th IASTED International Conference on Computers and Advanced Technology in Education (CATE 2004), Kauai, Hawaii, USA, August 16-18, 2004

26. Dossick, S.E., Kaiser, G., *CHIME: A Metadata-Based Distributed Software Development Environment*, in Proceedings of the Joint 7th European Software Engineering Conference and 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (ESEC/FSE'99), Toulouse, France, September 1999

27. Rothermel, K., Helbig, T., *An Adaptive Protocol for Synchronizing Media Streams*, Multimedia Systems, 5(5):324–336, September 1997

28. Gonzalez, A.J., Adbel-Wahab, H., *Lightweight Stream Synchronization Framework for Multimedia Collaborative Applications*, in Proceedings of the 5th IEEE Symposium on Computers and Communications (ISCC 2000), Antibes, France, July 4-6, 2000

29. Liu, H., Zarki, M.E., *A Synchronization Control Scheme for Real-time Streaming Multimedia Applications*, In Proceedings of the Packet Video Workshop 2003, Nantes, France, April 28-29, 2003.

30. Liou, S., Toklu, C., Heckrodt, K., *VideoTalk: a Collaborative Environment for Video Content Discussion*, in Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Florence, Italy, June 7-11, 1999.

31. Liu, J., Li, B., Zhang, Y.Q., *Adaptive Video Multicast over the Internet*, IEEE Multimedia 10(1):22–33, January 2003

32. Mandal, M. K., Idris, F., Panchanathan, S., *A Critical Evaluation of Image and Video Indexing Techniques in Compressed Domain*, In: Image and Vision Computing, 17(1):513-529, January 1999

33. Idris, F., Panchanathan, S., *Review of Image and Video Indexing Techniques*, Journal of Visual Communication and Image Representation 8(2):146-166, June 1997.

34. Das, M., Liou, S., *A New Hybrid Approach to Video Organization for Content-Based Indexing*, in Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS'98), Austin, Tx., USA, June 28-July 1, 1998.