Adaptive Internet Interactive Team Video

Dan Phung[1], Giuseppe Valetto[2], and Gail Kaiser[1]

[1] Columbia University, New York, USA
[2] Telecom Italia Lab, Turin, Italy

Abstract. The increasing popularity of 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 students. We present an e-Learning architecture and adaptation model called AI^{2}TV (Adaptive Internet Interactive Team Video), which allows virtual students, possibly some or all disadvantaged in network resources, to collaboratively view a video in synchrony. AI^{2}TV upholds the invariant that each student will view semantically equivalent content at all times. Video player actions, like play, pause and stop, can be initiated by any student and their results are seen by all the other students. These features allow group members to review a lecture video in tandem, facilitating the learning process. Experimental trials show that AI^{2}TV can successfully synchronize video for distributed students while, at the same time, optimizing the video quality, given fluctuating bandwidth, by adaptively adjusting the quality level for each student.

1 Introduction

Life-long and distance learning programs such as the Columbia Video Network 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 rewatch) 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. To that end, we provide an e-Learning architecture supporting virtual student groups.

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 of the users are relatively disadvantaged with respect to bandwidth (e.g., dial-up modems) and local computer resources (e.g., old graphics cards, small disks). We have adopted technology 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

R.W.H. Lau et al. (Eds.): ICWL 2005, LNCS 3583, pp. 66–77, 2005.

can generate different sequences of JPEG images for a range of different compression 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 remaining 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 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 third and main contribution of AI^{2}TV concerns optimizing the video quality according to the bandwidth constraints of each user, while enforcing group synchronization.

A distributed feedback control loop dynamically adapts each video client regarding group synchronization and video quality. The controller relies on sensors embedded in each client to periodically check information about the synchronization state, the buffering level and the bandwidth perceived at each client, and on actuators that in response tune local configuration parameters, such as the choice of both the next image to display and the next image to retrieve from the semantic compression levels available. A single controller is used for all clients in the same group, so it can detect and react to "skew" across the clients; it may reside on the video server or on another host on the Internet.

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 collaborating video viewing, and provides an evaluation of the effectiveness of our approach, with empirical results obtained with real lecture videos.

2 Motivation and Background

Correspondence courses have been available for over a century, e.g., the University of Wyoming[1] began offering extension courses in 1892, 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 if some Internet–based tools, like instant messaging, application or 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 lecture and class discussion.

Our AI^{2}TV project contributes to synchronous collaboration support for lifelong and distance education, and specifically to the area of collaborative video viewing, to foster virtual classrooms and borderless education. Our design is intended for small classes or small study groups within a larger class, and reaches out to disenfranchised users with dial-up level bandwidths, who still constitute a significant portion of the Internet user community [5], to allow them to collaborate with other users that enjoy higher bandwidth resources.

Collaborative video viewing poses a twofold problem: 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. a We use *semantic compression* to produce a video with cumulative layering. The semantic compression algorithm, developed by Liu and Kender [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 semantic information. 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 Fig. 1. Note that the semantic compression algorithm produces an effectively random distribution of key frames: when there are pockets of relatively high frequency semantic change, more key frames are produced.

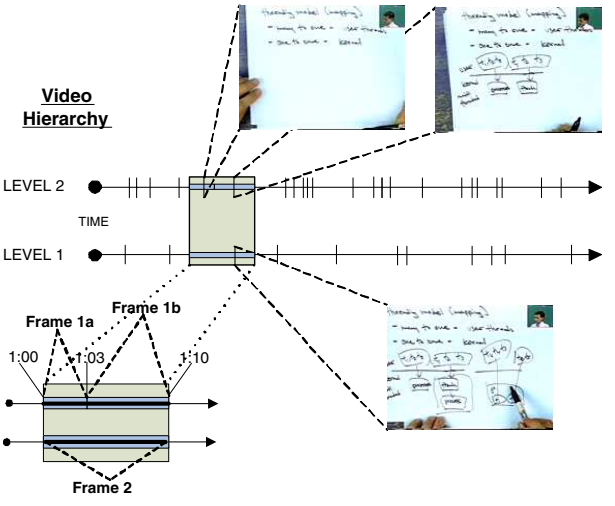

Fig. 1. Semantic Video Scenario

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 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.

Through the use of the Liu/Kender video compression algorithm, we can potentially provide semantically equivalent content to a group of students with diverse resources, by adjusting 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. The 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 at al. [10]. 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 [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 Architecture and Adaptation Model

3.1 System Architecture

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

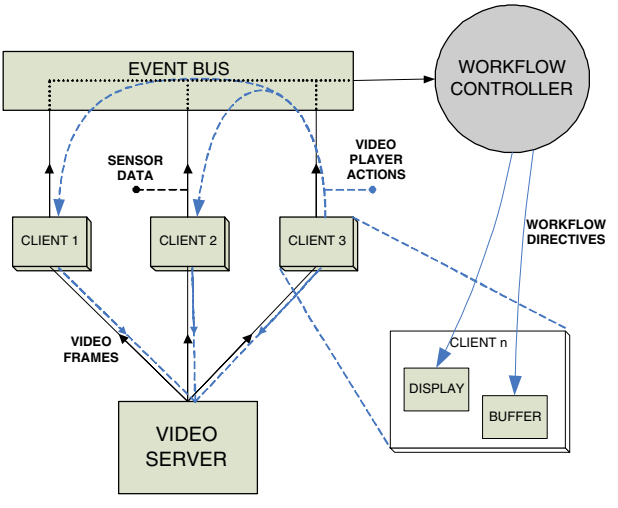

Fig. 2. AI^{2}TV Architecture

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 multiple times 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, 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 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 a control hook 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 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 architecture provides an end-to-end closed control loop, in which sensors attached to a generic 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 [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 hooks (i.e., the 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.

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

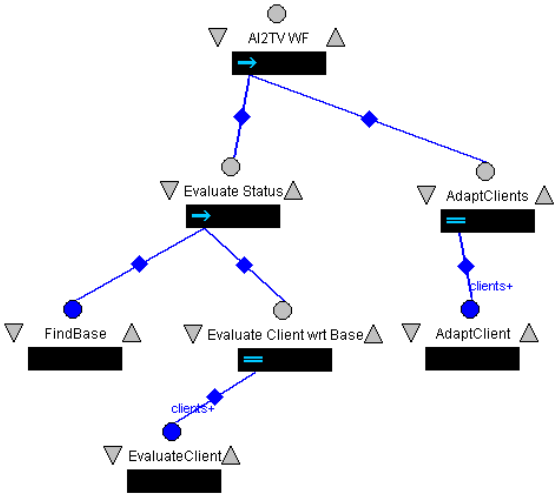

Fig. 3. AI^{2}TV Workflow diagram

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 Fig. 3. 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 low bandwidth, it 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 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 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 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 drops back down one quality level.

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 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 different 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 shaperd. 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 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, like receiving Internet audio or 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 amount of missed frames, these results provide evidence of the benefits of the adaptive scheme implemented by the autonomic controller.

The data show 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.

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., [14, 15, 16]) 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% (± 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.

5 Related Work

Yin *et al.* [17] 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. 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 [18]. 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 implementation of AI^{2}TV is described in [19]. In that version, a collaborative virtual environment (CVE) supported a variety of team interactions [20], 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 [21], the work of Gonzalez and Adbel-Wahab [22], or that of Liu and El Zarki[23], 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.

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 [24]. 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.

6 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. 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 video together with optimized video quality to distributed student groups.

Acknowledgments

We would like to thank John Kender, Tiecheng Liu and other members of the High-Level Vision Lab for their help with their semantic compression software. We would also like to thank other members 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 is funded in part by National Science Foundation grants CNS-0426623, CCR-0203876, EIA-0202063, EIA-0071954, CCR-9970790 and in part by Microsoft Research.

References

1. Miller, J., Ditzler, C., Lamb, J.: 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. (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** (2000)
4. Burgess, L.A., Strong, S.D.: Trends in online education: Case study at southwest missouri state university. Journal of Industrial Teacher Education **19** (2003)
5. Richtel, M.: In a Fast-Moving Web World, Some Prefer the Dial-Up Lane. The New York Times (2004)
6. McCanne, S., Jacobson, V., Vetterli, M.: Receiver-driven layered multicast. In: ACM SIGCOMM. Volume 26,4., New York, ACM Press (1996) 117–130
7. Li, W.: Overview of the fine granularity scalability in mpeg-4 video standard. IEEE Transactions on Circuits and Systems for Video Technology **11** (2001) 301–317

8. Liu, T., Kender, J.R.: Time-constrained dynamic semantic compression for video indexing and interactive searching. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume 2. (2001) 531–538
9. Valetto, G.: Orchestrating the Dynamic Adaptation of Distributed Software with Process Technology. PhD thesis, Columbia University (2004)
10. Lemer, B.S., McCall, E.K., Wise, A., Cass, A.G., Osterweil, L.J., Jr., S.M.S.: Using little-jil to coordinate agents in software engineering. In: Automated Software Engineering Conference. (2000)
11. Valetto, G., Kaiser, G.: Using Process Technology to Control and Coordinate Software Adaptation. In: International Conference on Software Engineering. (2003)
12. Parekh, J., Kaiser, G., Gross, P., Valetto, G.: Retrofitting Autonomic Capabilities onto Legacy Systems. In: 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: International Conference on Multimedia Computing and Systems. (1998) 233–236
14. 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** (1996) 1388–1403
15. 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** (1998) 102–112
16. Wang, Y., Ostermann, J., Zhang, Y.Q.: Video Processing and Communications. Prentice Hall (2002)
17. Yin, H., Lin, C., Zhuang, J.J., Ni, Q.: An adaptive distance learning system based on media streaming. In: International Conference on Web-Based Learning. (2004) 184–192
18. 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: 26th Applied Imagery Pattern Recognition Workshop, SPIE (1997)
19. Gupta, S., Kaiser, G.: A Virtual Environment for Collaborative Distance Learning With Video Synchronization. In: 7th IASTED International Conference on Computers and Advanced Technology in Education. (2004)
20. Dossick, S.E., Kaiser, G.E.: 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. (1999) 464–475
21. Rothermel, K., Helbig, T.: An Adaptive Protocol for Synchronizing Media Streams. Multimedia Systems **5** (1997) 324–336
22. Gonzalez, A.J., Adbel-Wahab, H.: Lightweight stream synchronization framework for multimedia collaborative applications. In: 5th IEEE Symposium on Computers and Communications. (2000)
23. Liu, H., Zarki, M.E.: A synchronization control scheme for real-time streaming multimedia applications. In: Packet Video 2003. (2003)
24. Liu, J., Li, B., Zhang, Y.Q.: Adaptive video multicast over the internet. IEEE Multimedia **10** (2003) 22–33