Distributed Teams Watching Video Lectures

Amer Chamseddine
CHILI, I&C, EPFL

***Abstract*—Online education has the potential to revolutionize the way we currently teach. We mention active learning, instant feedback, and self pacing, to list a few of the many benefits. However, students end up watching video lectures individually, which negatively affects their performance, according to a Stanford study. We present a system that allows students to collaboratively watch video lectures, in small groups. It combines all the benefits of online education, without losing on the interaction side. We show that the system is technically feasible, and, by carefully designing the UI, can be effective and easy to use.**

***Index Terms*—online, education, distributed, group, video, watching, MOOC, multimedia, stream, synchronization, notification, awareness, collaboration**

I. Introduction

MASSIVE Open Online Courses (MOOCs) have recently attracted the attention of many researchers. MOOCs are a recent development in the area of distance learning. The idea is to allow virtually any number of students to take a course online. Many people believe that this concept can revolutionize the way education is delivered. People like Anant Agarwal (director of edX, an open-source MOOC platform) and Salman Khan (creator of Khan Academy, one of the first platforms for online education) argue that it is the next big technological advance in the history of education, after the invention of the press.

Proposal submitted to committee: March 21st, 2014; Candidacy exam date: March 28th, 2014; Candidacy exam committee: Babak Falsafi, Pierre Dillenbourg, George Candea.

This research plan has been approved:

Date: _____

Doctoral candidate: _____
(name and signature)

Thesis director: _____
(name and signature)

Thesis co-director: _____
(if applicable) (name and signature)

Doct. prog. director: _____
(B. Falsafi) (signature)

The advantages of online education are numerous:

- Internet-hosted video lectures provide students with time- and space- *flexibility*. They can speed up, slow down, pause, and restart the video, to match their *own pace*.
- By splitting long lectures into smaller modules, and interleaving them with exercises, we achieve *active learning*. Students engage by interacting with the material. Forcing them to correctly solve an exercise before they can proceed achieves *mastery learning*.
- Computers can automatically check exercise solutions, which enables students to benefit from *instant feedback*.
- By applying *gamification* techniques, we keep students stimulated by the course material.
- In forums, students can answer each other's questions, and therefore benefit from *peer learning* (learning by teaching).
- Finally, since the entire system is computer-based, instructors can *instantly track* the progress of students, spot the ones who need help, and address their needs, in real-time.

Despite all the advantages stated in the previous paragraph, in an online education setting, students end up watching video lectures individually. They lose all the peer interaction, which can be extremely beneficial, according to a study performed by Gibbons et al. at Stanford [1]. Gibbons et al. compared students attending lectures, in four different conditions:

- Students attend lectures in the classroom.
- Students remotely watch lectures being broadcast live through TV or similar channels.
- Students individually watch pre-recorded lectures.
- Students watch pre-recorded lectures in co-located, small groups (3–6 people), with a tutor (TVI).

The study showed that students in the fourth condition outperform the three other conditions. It used grade point averages as the comparison metric. Further studies showed that this result can be extended to the distributed group setting (DTVI), and that a tutor is not needed. In the distributed group video watching setting, students benefit from all the advantages of online education, without losing peer interactions.

In Section II, we discuss a study performed by Cadiz et al. at Microsoft Research [2], which compares groups of people following video lectures, under different conditions. Section III compares different multimedia stream synchronization techniques, which can potentially be used in the distributed group video watching setting. We then discuss notification systems and awareness in collaborative activities, in Section IV. There is a recommended set of features a distributed group video watching system should implement, in order to satisfy members' need to maintain awareness of

each other's activity, within a group. Finally, in Section V, we present a preliminary work we have done in the area of collaborative watching of video lectures, and discuss further research directions.

II. Distributed Collaborative Video Viewing

People's current lifelong learning needs pushed scientists to find simplified and easily accessible ways for knowledge acquisition. Historically, people have tried various channels to diffuse educational material, ranging from broadcasting lectures via TV channels and video on-demand sites, to Collaborative Video Viewing (CVV) and Distributed Collaborative Video Viewing (DCVV).

DCVV does not constrain students in terms of location (online) and time (small teams), is scalable (teams are independent), and achieves better learning outcomes. Technically, DCVV can be implemented by extending existing collaboration tools, and run on currently available commercial systems.

A. DCVV System Challenges and Implementation

The video needs to be synchronized across all players. Cadiz et al. show that naive techniques (synchronize over the phone, or share desktop) do not work. One has to implement a partial sharing mechanism, whereby VCR controls are shared, but the video is independently streamed to participants. Cadiz et al. implemented this using Microsoft NetMeeting, and Windows Media Player ActiveX components.

The system is distributed, and event-based, which makes room for race conditions. Cadiz et al. employed techniques to work around these conditions, mainly by using timestamps to order events, based on the video time.

Being a collaboration tool, DCVV should implement communications mechanisms, to allow users to interact. Cadiz et al. featured three communication channels, in their system: chat, audio, and video.

B. Study

Cadiz et al. performed a study where they compared groups of people watching video lectures, under different conditions. The videos used were recordings from Harvard Business School, accelerated at 113% of their original speed. Recruited participants were intermediate computer users, from Seattle. In the distributed setting, headsets were used because of echo. In the co-located setting, one person controlled the video. Participants were not told that they have to solve a quiz at the end. To evaluate, they used three families of metrics: learning and comprehension, group interaction and discussion, and participants satisfaction. Measurements were done using seven-point Likert scale.

The pre-study was within-subject, and had four conditions, with a tutor: co-located group, distributed group with access to chat channel, distributed group with access to audio channel, and distributed group with access to audio and video channels. Participants were split in two groups of four people, and one group of three people.

The main study was between-subject, and had three conditions, without tutor: the distributed group with access to chat condition was dropped. Participants were split in groups of 3–4 participants: six co-located groups, four distributed groups with audio channel, six distributed groups with audio and video channels.

C. CVV versus DCVV

CVV and DCVV groups had similar scores. The number of times they paused the video was not significantly different, and they both were satisfied with the number of times the video was paused. However, CVV groups felt more comfortable pausing the video, and asking questions or making comments. CVV groups spent on average more time discussing the video, but both groups thought that the amount of time spent discussing was adequate. From a satisfaction angle, DCVV groups were significantly less bored than CVV.

D. DCVV-audio versus DCVV-video

All groups had identical scores, and reported similar levels of comprehension, but audio only groups thought their team members' comments were significantly more helpful. The number of times groups paused the video was not significantly different, and they all thought it was adequate. All groups spent an equivalent amount of time discussing the videos, and felt the same comfort asking questions or making comments. Audio only groups thought there was too much discussion, but they were significantly more satisfied with the discussion.

E. Discussion

Cadiz et al. did not use real-world educational videos, and participants were not real students: a field study can be conducted to validate the results. DCVV users were reluctant to pause the video: they claim they do not want to interrupt their peers, but the real reason could be that they are not familiar with the system. Improving the UI could help. Alternatively, videos could have in-built pauses, by design. In DCVV, Cadiz et al. did not show whether the presence of a tutor improves learning outcome.

The relevant contribution of Cadiz's work consists in showing that: (1) the chat channel alone is useless—people will not pause the video because they do not need to talk, but at the same time, they cannot focus on two things (chatting and following the video) simultaneously, and (2) the video channel has no major added value over the audio channel—video is mainly used to see if others are bored.

Although the study shows that the video is not needed, in the context of awareness, showing the presence of the other parties, visually, is essential, especially when the video is playing, because the audio channel is muted. Simple presence awareness can be achieved without fully fledged video channel, though.

III. Multimedia Stream Synchronization

We look at techniques to synchronize multimedia streams. In the distributed group video watching setting (discussed in the previous section), the video playback, the audio channel,

and the video channel must be synchronized, in order to have a usable system.

Streams are of two kinds: continuous and static. Continuous streams, such as audio, video, etc., have temporal dependency among their Media Data Units (MDUs), unlike static streams, such as text, slides, etc. Continuous streams can be further subdivided into: live and syntactic streams. In live streams (like VoIP calls), recording and playback are done simultaneously, unlike syntactic streams (like YouTube videos).

There are three families of (temporal) synchronization of continuous streams:

- *Intra-stream synchronization* (a.k.a. intra-media synchronization) consists in synchronizing the source and receiver to respectively produce and consume MDUs at the same rate. The primary goal of such a synchronization algorithm is to make sure the receiver buffer does not run in underflow or overflow situations.
- *Inter-stream synchronization* (a.k.a. inter-media synchronization) consists in synchronizing different media that have temporal relationship, e.g., lip-synchronization. Typically the algorithm needs to take actions during playback, in order to correct deviations.
- *Group synchronization* (a.k.a. inter-destination synchronization) consists in synchronizing different receivers playing the same stream (in multicast). Tele-teaching is one example.

Another subcategory of temporal synchronization is *interactive synchronization*. In this setting, receivers can influence the presentation of a stream, and this obviously must be synchronized. One example is when users pause, rewind, or fast-forward a video, in a group synchronization setting. This adds an extra layer of complexity in the synchronization algorithm.

Event-based synchronization is similar to temporal inter-stream synchronization except that time is replaced by events. It is used in networked games.

Multiple factors influence maintaining temporal dependency between (or within) streams, including: *network delays*, *network jitter*, *end-system jitters*, *clock skew*, *clock drift*, *rate drift*, *network skew*, and *presentation skew*.

Playout synchronization algorithms are classified according to three orthogonal dimensions (*time*, *location*, and *method*). A system is built differently depending on:

- Whether the system uses *explicit common understanding of time* or not.
- Whether correcting actions are applied at the *source* or at the *receiver*.
- Whether the system uses *stuffing techniques* (duplicating/skipping MDUs) to recover synchrony, or *adjusts the rate* of transmission/playback (speed up/slow down generation/presentation of MDUs).

A. Synchronization Techniques

Different synchronization techniques can be applied in group synchronization:

- In a *master/slave receiver scheme*, one receiver is elected as master. The other receivers do their best to synchronize with it. This is achieved via a multicast channel from the master receiver to all other receivers.
- In the *synchronization maestro scheme*, one node is elected as the maestro. The maestro can be a receiver node or a source node. All receivers send unicast messages to the maestro, which in turn sends multicast control packets to orchestrate the playback.
- The *distributed control scheme* consists in sending messages between any two receivers in order to synchronize the playback.

We can classify the above techniques in two categories: centralized versus distributed. On the one hand, with a centralized technique, it is easier to maintain causality. On the other hand, from a scalability angle, a distributed technique is superior. The synchronization maestro scheme is in between: it is easy for the maestro to keep causality, as all receivers report to it, while, at the same time, it is possible to dynamically substitute the maestro, to avoid a single point of failure.

Similar techniques can be applied in inter-stream synchronization. One stream can be selected as the master stream, and all other stream synchronize with it.

In a master/slave scheme, it can be beneficial to dynamically change the master. In some systems, the stream with highest global importance is declared as the master stream, e.g., audio is more important than video in conference calls. In other systems, the master receiver can be the slowest receiver, e.g., distributed group watching a video.

Depending on the algorithm, when asynchrony is detected, a corrective action is taken. Actions can typically be to increase/lower transmission/playback rate, slow down/speed up clock, skip/duplicate MDUs, contract/expand virtual time, etc. Some application do not tolerate abrupt skipping/pausing; they implement playout rate adjustment, or enforce blocking (or restricted blocking) policies.

Depending on the application, the algorithm can be conservative or optimistic. Conservative algorithms wait to receive everybody's acknowledgement for a particular event, before they proceed, whereas optimistic algorithms continue executing, and can detect and repair inconsistent states, by, for instance, rolling back. One example is Trailing State Synchronization.

B. Techniques Classification

Boronat et al. [3] have classified synchronization techniques in four different categories (summarized in Table I):

- *Basic* techniques are employed by most synchronization algorithms to maintain temporal synchronicity.
- *Preventive* techniques are used to prevent a state of asynchrony from happening.
- *Reactive* techniques serve to correct asynchrony, after it happens.
- *Common* techniques can be used for both avoiding and/or recovering from asynchrony. They are typically application dependent.

C. Discussion

The main contribution of Boronat's work consists in collecting a comprehensive list of 53 multimedia synchroniza-

Class	Location	Purpose
Basic	Source	Attach timestamp/sequence number
	Receiver	Buffering
Preventive	Source	Deadline-based transmission scheduling
		Initial playout instant calculation
		Interleaving MDUs
	Receiver	Preventive skips/pauses
		Enlarge/shorten silence time
		Change buffering waiting time
Reactive	Source	Adjust transmission rate
		Decrease number of media streams
		Drop low-priority MDUs
	Receiver	Adjust playback rate
		Reactive skips/pauses
		Virtual time contraction/expansion
Common	Source	Advance transmission timing
		Skip/pause
		Adjust input rate
		Media scaling
	Receiver	Adjust playout rate
		Data interpolation

TABLE I
SYNCHRONIZATION TECHNIQUES

tion solutions (including group synchronization, and many-to-many synchronization techniques), checking their features, and tabulating them in a way to make it easy to select the most appropriate one for a particular application. Boronat et al. did not quantitatively evaluate the presented solutions.

IV. NOTIFICATION SYSTEMS AND AWARENESS

To collaborate effectively, people need to know a large amount of information about their collaborators. This knowledge forms the basis of any discussion between two people, and is known as common ground. In face-to-face meetings it is easy to build common ground, using indirect cues. Humans are particularly good, perhaps due to evolution, at collecting meta information, and analyzing it subconsciously, to build a common ground.

In online meetings, many meta data channels are disrupted. Often, the field of view is reduced or not present, limiting the amount of gestures, or facial expressions one can use. Deixis and spatial references are not possible, and the varying network delays may lead to misconceptions. In this environment, maintaining common ground becomes challenging, and requires significant effort that people are not always willing to spend.

For an online collaboration to be successful, participants should continuously maintain awareness of others' presence, actions, and intents. One way to maintain awareness is to employ a notification system. A notification system is a "lightweight event-triggered display of information, peripheral to a person's current task-oriented concern." Notifications enrich the display area outside one's primary task window, and help keep awareness of events beyond one's current task. They work well for signaling spontaneous events, such as the reception of an email, but are less adapted to complex ongoing collaborative activities.

Activity Theorists define activity as a long-term endeavor directed at major goals. It includes: top-down goal decomposition, nonlinear development of partially ordered plan fragments, interleaving of planning acting and evaluation, and opportunistic plan revision. Additionally, an activity entails performing tasks like coordinating, assigning roles, making decisions, negotiating, and prioritizing.

A. Different Types of Awarenesses

Social awareness answers the question "who is around?" Opening a video link between two remote offices is an example of social awareness. According to Buxton et al., it is necessary but not sufficient for effective common ground [4].

Action awareness answers the question "what is happening?" Examples include seeing partner's pointer or gaze, etc. With action awareness, collaborators can see each other's progress on individual tasks, e.g., a version control system keeps the history of modification of individual files. There is a tradeoff between the amount of information displayed, and the ability of a user to interpret it. Since it is up to the user to interpret this info, often, from a user perspective, the awareness maintenance is a separate meta task that must be done besides the real task they have to accomplish.

Activity awareness answers the question "how are things going?" It is the awareness of project work that supports group performance in complex tasks. Part of activity awareness is situation awareness. Situation awareness is "knowing what's going on around you." It is the "perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in near future" [5]. Activity awareness puts emphasis on the aspects of a situation that have consequences for how a group works toward a shared goal over time. It implies awareness of other people's plans and understandings, and knowledge of how a task components are identified, coordinated, and carried out.

B. The Virtual School Tool

Carroll et al. developed a remote collaboration tool, called Virtual School, to support students who want to work in teams. The tool features a notification system, which works well to promote common ground and awareness, but fails to support activity awareness, according to Carroll et al. [6]. Based on their failure, the authors give recommendations on how to build a system that supports activity awareness.

Carroll et al. worked closely with school teachers to build their tool. The system is a collaborative editor that features a planning tool, a bibliography tool, a shared whiteboard, and a text/graphics page. It allows users to communicate using chat, email, and video conferencing. The awareness-oriented features include: auditory cues to notify users when people leave or join the session, a visual indicator showing when the shared notebook was last edited, a permanent notification log of significant user actions, a planner page displaying charts for project state, tasks, and deadlines, and indicators showing who is locking which document for editing.

Carroll et al. evaluated their Virtual School on an Aerodynamics project, in which middle and high school students had to collaborate to build kites. Students met once a week, but had a loose division of work into subtasks. The plan failed. Each subgroup felt that the others had just not bothered.

C. Analysis

To understand why Virtual School failed to provide the required level of activity awareness, Carroll et al. analyzed 26 hours of synchronous interaction, including comments and discussions, with the help of their critical incident tool. They classified the failure factors into four categories, and suggested workarounds:

Situational factors. During the course of the project, a previously unexpected event caused one of the teachers to move a deadline. The change was never communicated to the others, and caused a problem. This is an example of forgetting to notify. A graphical view of the project timeline, including deadlines and other important dates, helps avoid this situation.
Group factors. High school students did not have enough confidence that middle school students can do their job. This is an example of misperception of group members ability. It helps if the tool provides a mechanism to know more about one another, see contribution done by others, including interaction history, and emphasize project artifacts, by attributing the work to subgroups. Avatars also help reflect personality.
Task factors. Students did not understand and pursue their individual tasks *in the context of the group's overall purpose*. Most of the time, people missed their meetings, and worked asynchronously. A timeline showing project artifacts, with entire interaction history helps people to stay in the loop, even if they miss a meeting. A view of the timeline with coarse level of details also helps teachers keep track of the big picture.
Tool factors. Features provided by Virtual School did not match people's expectations. The log of actions was in a separate window, and the project planning was a page in the shared document. Teachers had hard time following students' progress. Integrating project planning and user work in the timeline, as well as making it easily accessible helps make the timeline a single place to track everything.

In the case of Virtual School, activity awareness can be achieved using a timeline that aggregates all information, including deadlines, communications, document versions, and artifacts.

The relevant contribution of Carroll's work consists in enumerating the different types of awarenesses, and defining them in details. Carroll et al. listed the main components of each kind of awareness, and enumerated the key factors to achieve it. For distributed groups watching video lectures, action awareness should be used. Progress bars display the current position of each participant, and auditory cues mark events such as participant leaving/joining a group.

V. Research Proposal

We have built a proof-of-concept system for distributed groups to watch video lectures together, and discuss them. Our system implements the basic set of features to allow collaborative watching. Features include: synchronized video, chat functionality, audio channel, and echo cancellation.

A. Design

We prototype the system as a PocketCampus plugin. PocketCampus is a mobile device application used at EPFL to give easy access for campus dwellers to various services. It provides a mobile platform that greatly simplifies building a client-server application, by automatically generating boilerplate code. To benefit from the advantages of this platform we implement the system as a PocketCampus plugin.

We integrate the system with edX and YouTube. edX is an open-source MOOC platform that can be customized to support features like the ones we want to implement. Because we want to be able to quickly test our system, we interface it with edX pages, without writing a custom edX module. This way, we directly benefit from real course contents. edX uses the YouTube platform to host its videos. We use YouTube's API to get access to these videos inside our application.

We implement our own VoIP protocol. Existing VoIP providers, e.g., Skype, do not have APIs to allow a third-party application to use them as an audio channel. Using SIP is not only complex in terms of server configuration, but also does not provide an easy way to perform multi-party calls. The quickest solution is to implement our own multi-party VoIP protocol.

B. Features

We use event-based synchronization to synchronize the video and chats. We track the users' interaction with the video using the YouTube API. Every interaction is an event. When a user presses a button, we broadcast the event to all other users in the group. We currently use an optimistic synchronization algorithm, and ignore inconsistent states. We should later detect inconsistencies, and recover from them. Chats are also event-based synchronized. They are broadcasted within the group.

The application allows participants to talk to each other. Assuming that the number of participants is n, the server receives n streams that need to be synchronized using an inter-stream synchronization technique. The server then needs to send back another n streams, which should be synchronized using a group synchronization technique. For the group synchronization, we use playback rate adjustment, at the receivers, in order to recover from asynchrony.

We use an echo cancellation module in order to allow study-room participants to talk in speaker-phone mode. Normally, in speaker-phone mode, a device's microphone picks up the audio being played back through the device's own speaker. When this happens, users get annoyed by hearing their own voice, echoed through another user's device. We do not want to force people to use headsets, like in Section II. We use echo cancellation libraries to filter out the played back signal, from the signal recorded by the device's microphone.

C. Implementation

We implemented our own algorithm to broadcast messages within study-rooms. We started with a simple message passing algorithm that sends a message to every user actively polling a particular room. With that algorithm, messages get duplicated as many times as there are participants in a room. As the room size grows bigger, the server has to store increasingly more copies of the same message, which are redundant data.

To resolve this problem, we came up with a new message passing algorithm where there is a single, permanently-stored queue of messages per room, and clients take care of keeping state of their read progress.

For audio, we simply cannot reuse the broadcast algorithm described in the previous paragraph. If there are n participants in a study-room, and if the server blindly forwards every audio frame it receives from a participant, to every other participant, then the total bandwidth consumption is n times more than what is required. The server should mix the incoming audio streams, and forward only one audio stream to each participant. For that, we made the server wait until it receives an audio frame from all required participants, then it mixes the received frames and forwards the resulting frame to the participant. In inter-stream synchronization, this technique is known as *blocking policy*.

The mixing algorithm presented in the previous paragraph needs to tolerate missing and delayed audio; students may leave rooms at any time, or their Internet connections may be slow or flaky. We handle these cases using a timeout mechanism that bounds the maximum time the server waits for participants. Messages that become delayed for more than the timeout value are sent out to clients without further delay. In inter-stream synchronization, this technique is known as *restricted blocking policy*.

D. Awareness

We have a social awareness (a.k.a. presence awareness) system. Participants are always aware of who is currently present in the study-group. When a participant joins/leaves, the system broadcasts a chat message, in the study-room, announcing that that participant has just joined/left. If a client connection is lost, the server waits for a certain time, then declares the participant as having left. These system messages can further be enhanced by an accompanying auditory cue.

From an action awareness perspective, we should display the current position of every participant in the group, on the video progress bar. Moreover, the system should announce, using a broadcasted message, when a person seeks, plays, or pauses.

E. Research Vision

Our current system is just a proof-of-concept. There is still room for improvement in terms of awareness. Also, different synchronization algorithms (such as the ones discussed in Section III) should be tried and compared. Later, we can perform a real-world study to test the system's actual effectiveness, and iterate on its features, after collecting users' feedback.

Quizzes and exercises functionality should be supported; it helps achieve Active Learning. Consensus quizzes is a functionality that requires participants to reach consensus, within their group, while responding to a quiz, before they can submit their agreed-on answer. We believe that this feature can boost discussion, therefore, enhance learning outcome.

We define the concept of elastic synchronicity by allowing one participant to detach from the group—perhaps, to go back and watch parts of the video—while keeping track of the progress of the others. A detached participant can always join back the group, at the expense of skipping some parts of the video, or by playing back the video at increased speed. This feature allows a participant to go back or pause, without disturbing the entire group, while, at the same time, allowing late participants to catch up.

Finally, the platform we have created is a fertile land for educational research to be conducted. One can come up with new hypotheses and quickly test them, validate/invalidate them, and reiterate. This way, we can answer questions like "what is the best group size for ideal learning outcome?", "is it more beneficial to have homogeneous or heterogeneous groups?", etc. We believe this is a valuable asset for research that we would like to further develop.

REFERENCES

[1] J. Gibbons, W. Kincheloe, and K. Down, "Tutored videotape instruction: a new use of electronics media in education," *Science*, vol. 195, no. 3, pp. 1139–1146, 1977.

[2] J. J. Cadiz, A. Balachandran, E. Sanocki, A. Gupta, J. Grudin, and G. Jancke, "Distance learning through distributed collaborative video viewing," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 2000, pp. 135–144.

[3] F. Boronat, J. Lloret, and M. García, "Multimedia group and inter-stream synchronization techniques: A comparative study," *Information Systems*, vol. 34, no. 1, pp. 108–131, 2009.

[4] W. A. Buxton, A. J. Sellen, and M. C. Sheasby, "Interfaces for multiparty videoconferences," *Video-mediated communication*, pp. 385–400, 1997.

[5] M. R. Endsley, "Design and evaluation for situation awareness enhancement," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 32, no. 2. SAGE Publications, 1988, pp. 97–101.

[6] J. M. Carroll, D. C. Neale, P. L. Isenhour, M. B. Rosson, and D. S. McCrickard, "Notification and awareness: synchronizing task-oriented collaborative activity," *International Journal of Human-Computer Studies*, vol. 58, no. 5, pp. 605–632, 2003.