Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
CWcollab: Presenting multimedia with web-based context-aware collaboration

Chunxu Tang, Beinan Wang, C.Y. Roger Chen, Huijun Wu

1. Introduction

Real-Time Collaboration (RTC) is gaining significant popularity during the COVID-19 pandemic period. It is usually essential for geographically dispersed teams to have a collaboration tool. Thanks to the distributed and rapidly increasing volume of data and expeditious development of modern web browsers, an organized web-based group-aware platform supporting collaboration among a large number of users is necessary. In such RTC platforms, not only are the artifacts of multimedia shared but also the controls on the artifacts are broadcast and synchronized.

Some of the pioneering work, such as GroupSketch [1], Video-Whiteboard [2], and Liveboard [3], implemented collaborative multimedia systems by capturing users' drawings and projecting them to remote screens. Based on the same technology, various screen sharing products, such as Zoom, Cisco WebEx, and Google Meet, were developed and are now widely used. Recently, with the rapid development of modern web technologies, capturing and projecting users' drawings were applied to lots of web-based collaboration tools including Collabode [4], RichReview++ [5], and Tele-Board [6].

To our knowledge, even though various web-based groupware tools have been developed for versatile purposes such as whiteboard drawing and document editing, no systems can integrate these functionalities to suit general-purpose multimedia collaboration. Kim et al. [7] made a valuable contribution in this direction that they proposed an Model-View-Controller (MVC) architecture for ubiquitous collaboration. Their tool still only handled static media like whiteboard drawings and images, without effectively working on other types of media with dynamic contents such as videos and web pages.

Furthermore, for current popular screen sharing products, both the contents of the media and manipulations on the media in a specific session are transmitted by capturing the display continuously, leading to large consumption of network bandwidth. This poses challenges for large geographically dispersed teams with an unstable network quality.

By contrast, we propose to split the contents of a presentation into static media resources and dynamic actions. The static media resources, for example, a video or a PDF document, can be transmitted to attendees beforehand; and the dynamic events occurring in a session such as muting a video are broadcast and synchronized on the fly. As a result, a collaboration session is organized as the combination of static materials and dynamic events encapsulated in an event-driven stream of messages. With these messages, we also implement the precise recording and replay of collaboration events, differentiating our work from traditional collaboration platforms.
To cover the research gaps in prior work mentioned above, we propose a context-aware web-based collaborative multimedia system—CWcollab. Specifically, our contributions are:

1. A general-purpose collaborative multimedia system.
   - Support for general multimedia. Not only static media (PDF documents, images, etc.) but also dynamic media (videos, web pages, etc.), is supported in CWcollab. This notable feature differentiates our work from prior work, as illustrated in Table 1.
   - Support for general events. All actions in a session are captured as events, sent and handled on the fly. This also indicates the system is open to other possible extensions in a plugin pattern. A developer can add supports for various events of collaboration on different kinds of media following our uniform interface, demonstrated in Section 3.1.
   - Support for general environment. Our system is totally web-based. Users can access the system from various platforms, including desktops, tablets, and mobile devices, as long as web browsers are supported. This significantly eliminates the complexity of setup on different platforms.

2. An object-prioritized context-aware approach to capture and replay media actions for rich functionalities with low network bandwidth consumption.
   - Object-prioritized media controls. To our knowledge, most web-based collaboration tools are position-based or proportion-based, which implies that media controls are synchronized through absolute or relative positions. However, many websites have applied the responsive web design, where the user interface is automatically adjusted on various devices with various screen sizes. This poses challenges for capturing and replaying events with the traditional position-based approach. By contrast, in CWcollab, media controls are related to media objects. We propose an object-prioritized hybrid synchronization approach, representing each action in simple messages, discussed in Section 3.3.
   - Rich functionalities. As each media control is represented by a simple message, CWcollab also supports rich interactive functionalities for collaboration and presentation such as material preparation, real-time synchronization, and precise replay of a session.
   - Low network traffic consumption. The usage of an object-prioritized approach also implies that CWcollab has a very low bandwidth usage, compared with current video conferencing tools. Some tools fall in the former category. For example, Han et al. [16] implemented a comparison study on three groupware tools: desktop-based Java tool XCHIPS [17], browser-based Adobe Flash application ThinkTank, and browser-based AJAX system PowerMeeting. They claimed that the last one has advantages in features, user interface, usefulness, etc. over the others. Moreover, Pimentel et al. [18] studied WebSocket, polling and long polling. They claimed that WebSocket has an obvious lower latency than the others.

Web-based collaboration systems could be categorized into two types, depending on users share one screen or each has his/her own display. Some tools fall in the former category. For example, Han et al. [19] proposed a unified XML framework for multi-device collaborative web-browsing. Schmid et al. [20] employed this XML format to develop a web-based interactive collaborative environment. To our knowledge, most of the current web-based collaboration tools are of the latter category, and this type of collaboration has been applied to various domains. For example, Google Docs and Etherpad are both rather mature products which enable real-time document editing collaboration.

2. Related Work

Real-Time Collaboration is usually considered as a subdomain of Computer Supported Cooperative Work (CSCW). Previously, people achieved basic remote collaboration by means of video/audio calls. While, it is obvious that this form of communication can only allow very preliminary cooperation. Afterwards, researchers developed more complicated collaborative applications. Some of the pioneer work includes GroupSketch [1], VideoWhiteboard [2], and Liveboard [3]. In these preliminary systems, users and their drawings are captured by cameras, transmitted and projected to remote screens. Another example is the Jazz project, developed by Cheng et al. [8]. It brings collaboration into programming bounded with Eclipse, via screen sharing. An issue of these systems is that real contents are not transmitted, so they lack the flexibility to provide interaction among people geographically dispersed.

Subsequently, researchers developed more elaborate native desktop collaborative applications. For instance, Booth et al. [9] proposed a “mighty mouse” multi-screen collaboration tool, which provides smooth mouse movement cross platform, via Virtual Network Computing (VNC) protocol. Wu et al. [10] created Software Design Board (SDB), which is a prototype collaborative software design tool. Callaghan et al. [11] leveraged the client-server architecture to create a collaborative learning environment, implemented in .NET remote services. Gallardo et al. [12] presented a collaborative modeling tool, integrated with Eclipse. A drawback of this kind of systems is that they usually require complicated setups on different platforms, due to distinct platform-specific implementations.

Recently, with the significant enhancement of modern web browsers, especially with the advent of technologies brought from HTML5 [13] standard, such as WebSocket [14], researchers began to consider web browsers as the backbone for real-time collaboration applications. Gutwin et al. [15] performed elaborate experiments on collaboration with three approaches: XMLHttpRequest (XHR), WebSocket and Java Applet. They claimed that WebSocket has a relatively high performance on various platforms including mobile devices. Similarly, Megan et al. [16] implemented a comparison study on three groupware tools: desktop-based Java tool XCHIPS [17], browser-based Adobe Flash application ThinkTank, and browser-based AJAX system PowerMeeting. They claimed that the last one has advantages in features, user interface, usefulness, etc. over the others. Moreover, Pimentel et al. [18] studied WebSocket, polling and long polling. They claimed that WebSocket has an obvious lower latency than the others.

Web-based collaboration systems could be categorized into two types, depending on users share one screen or each has his/her own display. Some tools fall in the former category. For example, Han et al. [19] proposed a unified XML framework for multi-device collaborative web-browsing. Schmid et al. [20] employed this XML format to develop a web-based interactive collaborative environment. To our knowledge, most of the current web-based collaboration tools are of the latter category, and this type of collaboration has been applied to various domains. For example, Google Docs and Etherpad are both rather mature products which enable real-time document editing collaboration.

Table 1
Comparison of multimedia collaboration system features.

| Document editing | Web browsing | Whiteboard | Image annotation | Video watching | Video annotation |
|------------------|--------------|------------|-----------------|----------------|-----------------|
| Google Docs      | ✓            |            | X               |                | X               |
| Google Meet      | X            | ✓          | ✓               |                | X               |
| SpreadVector [21]| X            | ✓          | ✓               |                | ✓               |
| Collaboard [26]  | X            |            | ✓               |                | X               |
| Kim et al. [7]   | X            |            | ✓               |                | ✓               |
| CWcollab         | X*           | ✓          | ✓               |                | ✓               |

*: Although CWcollab does not support fully-functional collaborative document editing, it does support collaborative annotation and manipulations on PDF documents.
for multiple users. Fetter et al. [21] created a collaborative web browsing tool distilling the concept of lightweight interference, transitions, and adoptions. Goldman et al. [4] created a Web IDE, Collabode, for real-time collaborative coding. It could also synchronize errors during programming. Chen et al. [22] proposed a framework for multiplayer online games with the help of WebGL and WebSocket. Ocaya [23] proposed a framework for collaborative remote experimentation, with a web browser interface. Yoon et al. [5] designed RichReview++ to support collaborative annotation system, especially for teaching purposes. Binda et al. [24] developed a photo-sharing system to stay aware of family members’ health. Boronat et al. [25] created Wersync, a web-based platform to enable collaborative social viewing.

Meanwhile, by utilizing modern HTML5 technologies, some researchers are devoted to integrate collaboration into traditional video conferencing products. For instance, Knuz et al. [26] created a device named CollaBoard which supports remote collaborative whiteboard based on video conferencing. To enhance the convenience of setup and usage, recently, Wenzel et al. [6] developed Tele-Board to provide web-based real-time collaboration combined with WebRTC-based video conferencing. They embedded the video in an iframe element on web browsers and adhered a drawing layer on top of the video to hold shared artifacts in the workspace. Chang et al. [27] proposed AlphaRead to support collaborative annotation in video-based objects.

With the boosted development of artificial intelligence in the recent decade, we also witnessed research works [28–32] integrating machine learning approaches into multimedia like image retrieval and face recognition. For example, Yan et al. [28] introduced a multi-view deep neural network into the hash learning domain to significantly improve the performance in image retrieval. In [30], the authors applied attention mechanisms to image captioning, helping learn non-visual clues for non-visual words.

In our study of four prior collaboration systems, they only support very limited types of media, as demonstrated in Table 1. Here, Google Docs supports document editing, Google Meet supports video watching, SpreadVector [21] is for collaborative web browsing, and Collabode [26] has a whiteboard functionality. The system described in [7] works for more types of media, even though it still only supports whiteboard and image editing. By contrast, CWCollab supports all types of multimedia listed in the table, especially for presentation purposes. This notable feature differentiates our work from prior work.

3. Architectural Design & Implementation

3.1. High Level Design

A crucial feature distinguishing our design from other collaboration systems is the general-purpose distributed collaboration with a uniform interface. As events are transmitted in a standardized message format, this design significantly relieves the difficulty of supporting a new type of media. Fig. 1 shows the structure of collaboration based on a uniform interface.

Sender flow. For the client who initiates a media event for collaboration, there is a media event capturer module to monitor and capture the event. A media state recorder will be invoked when necessary to record the current media state for possible resynchronization. For example, the state of a video may include the play/pause state, volume, progress, playback rate, related annotations, etc. At the same time, the event is sent to a message serializer, which encapsulates the event into a standardized message and sends it to the backend service.

Recipient flow. After the message is routed to one target client via the backend service, a message deserializer takes charge of unwrapping the message and sending information to the media event replayer to update the current media state. Simultaneously, the change of media state may also be recorded in the media state recorder module.

With the uniform interface, if a developer would like to add collaboration mechanisms for a new type of media, he/she just needs to follow the structure by adding functions in the media event capturer, media state recorder, and media event replayer. Message serializer and message deserializer should be left intact if the format of transmitted messages is not changed. The feature of extendability significantly relieves the burden of implementing new functionalities for collaboration.

3.2. Collaboration Subjects

A collaboration subject is an attendee in a collaboration session. A collaboration subject’s functionalities are usually reflected by its agents, for example, web browsers. We categorize a collaboration subject to be active or passive. An active collaboration subject, considered as a sender, actively controls the collaboration artifacts and broadcasts updates to other subjects. In a presentation scenario, an active collaboration subject is a presenter. By contrast, a passive collaboration subject, regarded as a recipient, passively receives messages and updates media states. Take Fig. 1 as an example: the client on the left side is an active collaboration subject, and the client on the right side is a passive

![Fig. 1. The implementation of collaboration based on a uniform interface.](image-url)
The role of a collaboration subject is mutable. At a particular moment, an active collaboration subject can be updated to become a passive one, and a passive collaboration subject can be changed to become an active one, taking charge of broadcasting media events. We discuss the details of access control in Section 4.4.

3.3. Media Event Capturer

The main functionality of the media event capturer component is to capture an abstract media event via Document Object Model (DOM) events. This makes CWcollab aware of the multimedia context, greatly differentiating our work from prior work. In our design, by using event propagation, especially event bubbling, which is now supported in all of the modern browsers, we create another layer on top of original media objects to capture these events. We leverage an object-prioritized hybrid approach to capture media events. From our implementation, we posit that most web-based multimedia events can be covered by this approach. Specifically, it consists of:

- **Object-based.** Events are captured as DOM events on the media object or a User Interface (UI) component. This provides more flexibility than the traditional position-based approach, as no matter where a media event fires, we only trace the object effected, isolated from the influence of positions. This type of capture scenario is much more often seen than others. Fig. 2 illustrates how to capture object-based video events. In the figure, every action is triggered through a UI component. For example, when a user plays/pauses a video, he/she clicks the video control button, which is eventually captured by the media event capturer as a click button DOM event. Similarly, when a user seeks to a timestamp in the video player, he/she clicks the progress bar, and the media event capturer captures this event as an update slider DOM event. (see Fig. 3)

- **Proportion-based.** Events are captured as DOM events related to the positions proportionally to the display. Although object-based capture handles most scenarios, some media events may not be object-based, especially those related to mouse events. For example, after capturing an image moving event through mouse events or button-related events, we can use the relative position of an image in the screen to synchronize the event.

- **Value-based.** Events are captured as DOM events related to value changes. For example, when scrolling a mouse to zoom in/out an image, the event can be captured via a wheel, mousewheel, or DOM-MouseScroll event. This type of event is fired with a value to represent how much the mouse has scrolled.

To summarize the handling of media events captured in our system, we demonstrate the captured events related to concrete media events in Table 2. Note that we also list annotations as a type of media, whose events like free drawing can also be captured via DOM events.

### Table 2

| Semantic event                  | Captured event                  | Synchronization |
|---------------------------------|---------------------------------|-----------------|
| Video play/pause video          | button click                    | Object-based    |
| stop video                      | button click                    | Object-based    |
| mute/unmute video               | button click                    | Object-based    |
| jump to time                    | progress bar click              | Object-based    |
| change speed                    | dropdown list click             | Object-based    |
| prev/next page                  | button click                    | Object-based    |
| jump to page                    | form submit                     | Object-based    |
| scroll page                     | mouse scrolling                 | Value-based     |
| zoom in/out                     | mouse scrolling                 | Value-based     |
| move                            | mouse down/move/up              | Propotion-based |
| crop                            | mouse down/move/up              | Propotion-based |
| webpage visit a provided URL    | form submit                     | Object-based    |
| load a URL on page              | link click                      | Object-based    |
| prev/next page                  | button click                    | Object-based    |
| Annotation free drawing         | mouse down/move/up              | Propotion-based |
| highlight text                  | selection change                | Object-based    |
| insert shapes                   | dropdown list click             | Object-based    |

3.4. Media Event Recorder and Replayer

The media event recorder is used to track the current state of the media. The state of a media block depends on the type of media. A state is a set of key-value pairs. For example, a state of a video contains the video source, current timestamp, muted or not, volume, playback rate, annotations, etc. The key-value pairs can be compressed into messages for further transmission.

The main functionality of a media event replayer is to replay events attached with context-aware media event information. To achieve that efficiently, we design a hierarchical tree structure-Handler Tree-to handle messages, as shown in Fig. 4.

An event is propagated in the handler tree via parent–child chains until arriving at a leaf node. During the propagation, the received message is parsed gradually, and finally, in a leaf handler, the message is totally consumed to update the UI if necessary. An example of handling a video play/pause event in CWcollab is demonstrated in Fig. 5.

When an event is received by an audience, the event is first sent to the root node, also acting as a dispatcher in the handler tree. The next level in the handler tree contains various handlers for corresponding types of media. The message is sent to the appropriate media handler at
this level. Then, a specific action handler is invoked to fire the target DOM event. For the example in Fig. 5, the message is first handled by the root handler and dispatched to the video handler according to its media type. Because the event type is button click, the message is next sent to the button click handler for further process. Then the handler reads the target id and triggers the click event on the element with this identifier. The handler also loads the value of current time in the data field to update the current progress of the video to that timestamp. Considering that our system is real-time, the tree is flat to reduce the overhead.

With the help of the media event replayer, the events occurring in a collaboration session can be replayed sequentially according to their timestamps. The events occurring in a session are pushed into a queue structure. These events are scheduled to be popped from the queue and sent to the handler tree for replay.

3.5. Event Messages

In CWcollab, collaboration events are captured and represented by a standardized format of strings, and the strings are transferred among collaboration subjects. There are two different types of messages: media events and control events. A media event message represents an action related to a specific media block, such as scrolling a web page. By contrast, a control event message represents an action involving controls in a collaboration session, such as resynchronizing the media state.

3.5.1. Media Event Messages

The major skeleton of a media event message is shown in Table 3. The message is represented by key-value pairs. Generally, each message adheres new fields to the message for other synchronization and collaboration purposes. We give some more examples below.

- **Media type identifier (media-type).** This key identifies the type of the media, such as video, audio, or image, whose event is fired.
- **Event type identifier (event-type).** This field is crucial for a message, as it describes the type of event captured to represent the target media event. This is also used in other collaboration subjects to replay the media action to achieve synchronization. The specific event types included in this field highly depend on the replaying mechanisms specified by users. We give some more examples below. Note that for each event type, there must exist a corresponding type of each media event replayer. Otherwise, the action cannot be repeated on other collaboration subjects’ displays.
  - button-click, the event of clicking a button.
  - move, the event of moving a media material.
  - mouse-scroll, the event of scrolling the mouse.
  - form-submit, the event to submit a specific form.

- **Sequence identifier (seq-id).** Since the events occurring in a collaboration session are in sequential order, the events are in a happens-before relationship, namely causal dependency [33]. To maintain the order of the events, similar to the mechanism in the traditional database logging system, a unique auto-incremental sequence number is assigned to every event.
- **Timestamp (timestamp).** This field contains the timestamp when the media event occurs and is captured. This can be useful, for example, if we want to replay all media manipulations based on the timeline.
- **Semantic (description).** This is the semantic description related to the media event. It makes the event more human-readable.
- **Optional data (data).** This field carries further information about the event, thus it is highly implementation-specific. The information included here may be consumed by essential media event replayers on another collaboration subject side. For instance, for a form-submit event, this field contains the value to be filled and submitted, such that the media event can be reproduced precisely.

Based on the basic structure we have explained above, we demonstrate two message examples in Fig. 6. In Fig. 6a, there is a message representing a play event of a video. It contains all the fields we have described. In the data field, a piece of current time information is included to indicate the time to play or pause. In Fig. 6b, there is a message for zooming out an image. Here, its data field holds a value of delta, standing for how much a user has zoomed out the image.

Our message structure is open to extension, and a developer can adhere new fields to the message for other synchronization and collaboration purposes, as long as the following rules are obeyed.

**Self-descriptive.** Each message should carry complete information to describe itself. This indicates that a media event replayer can obtain sufficient information to repeat the action, without any knowledge of prior events.

**No duplicate fields.** For example, when there has been a media-type field, it is not necessary to have another field like media-category that serves the same purpose.

**The fields except data should exist in all messages.** If a field can only exist in some of the messages, it should be moved to the data field. Take the id field in Fig. 6a as an example: in a button click event, an id refers to the identifier of the clicked button element. Considering that not all messages need a unique identifier to point to a web element, this value should be placed in the data field.

**Immutable.** After a message is created, it shall not be modified on the client or server side. To modify the message, we have to create a new message and discard the useless one.

**Deterministic.** All the values contained in the message for execution should be deterministic. That is, no randomness could exist. The execution of the message on every client should produce the same result.

### Table 3

| Element               | Field        | Example               |
|-----------------------|--------------|-----------------------|
| Media type identifier | media-type   | image                 |
| Media identifier      | media-id     | image-block           |
| Event type identifier | event-type   | mouse-scroll          |
| Sequence identifier   | seq-id       | 5000                  |
| Timestamp             | timestamp    | zoom out an image     |
| Semantic              | description  | delta: −1.5           |
| Optional data         | data         |                       |
3.5.2. Control Event Message

Besides media event messages, control event messages are also necessary to represent controlling information, such as change of authorities, resynchronization, and reorganization of panels, in a collaboration session. The major skeleton of a control event message, illustrated in Table 4, is quite similar to that of a media event message, except:

- **Control type identifier** (control-type). This value identifies the type of control event, such as resync. This field, together with media type identifier, is used to judge whether a message is a media event or a control event.

- **No indispensable media-related fields**. Unlike media events, the media type identifier and media identifier are not essential any more in control events. However, some media-related information may still be necessary to be included in the data field. For example, in Table 4, we put media-id in the data field to point to the specific media block to resynchronize.

Developers can also add new fields to the structure of a control event message, as long as they follow the same rules as those of media event messages.

3.6. Message Serializer/Deserializer

The responsibilities of a message serializer and message deserializer are to wrap a media message into a standardized format of string and to read the message from such a string, respectively. To maintain consistency, the two components share the same message format.

Currently, there are two widely used API formats: XML and JSON. Extensible Markup Language (XML) is a markup language that defines a very flexible text format, and it plays a crucial role in the communication of web services [34]. XML uses elements, tags, and optional attributes to describe data exchanged. It is especially conducive to represent data in hierarchical structures. During parsing, it is usually converted into a tree structure, XML Document Object Model. Some previous work, including [35–37], applied XML-based messages, especially Extensible Messaging and Presence Protocol (XMPP) to collaboration systems. By contrast, JSON, short for JavaScript Object Notation, is another human-readable data exchange format consisting of name-value pairs and an ordered list of values. Some prior work such as [38–40], claimed that JSON is much faster than XML in web applications. In Fig. 7, we demonstrate the comparison of messages of a play event of a video in JSON and XML formats. Because of the requirement of brackets in XML messages, they may need more bytes to represent messages compared with JSON.

Furthermore, binary message formats such as Protocol Buffers [41] and Apache Thrift [42] are also used in some collaboration platforms such as [43,44]. We do not restrict our application to a specific format, and it is easy to extend the message structure to other formats.

4. Design of Presentations

4.1. Presentation Preparation

In a presentation, if some static materials such as videos, images, and documents are presented, these resources could be prepared beforehand (preload mode), instead of downloading them on the fly (on-the-fly load mode). To achieve that, the static materials can be uploaded to the server. Other users are allowed to download them before the start of the presentation. Meanwhile, we also allow a presenter to dynamically add multimedia materials during a presentation by external URIs. This will influence the bandwidth usage of our system, as discussed in Section 5.

Besides the preparation of static materials, the concrete events in the presentation can also be prepared. Unlike some traditional presentation programs such as Microsoft PowerPoint and Google Docs, which organize a presentation in pages, we concentrate on the media events occurring in a session. Consequently, we allow users to prepare the events such as playing a video, zooming in an image, scrolling down a webpage, etc. beforehand.

4.2. Presentation Replaying

After attending a specific presentation, it is common that an attendee wants to replay the presentation. For example, after an online class, a student may watch the class contents again for a review. In previous collaborative systems, to our knowledge, only the screen-sharing technique could achieve this kind of mechanism, usually by recording all the actions occurring in a session as a video. By contrast, in our work,
because all media events are wrapped into simple messages, we could
replay the whole presentation by re-executing the messages one by one.
The structure of replaying a presentation is illustrated in Fig. 8.

Before the replaying of a presentation, the user first fetches all the
events of the presentation from the server. Afterward, all the events,
sorted by timestamp or seq-num, are pushed into a media events queue.
Because the timestamp of an event has been recorded during the pre-
sentation, we use timers to schedule these events for replaying and
formulate an event loop to handle the fired events sequentially. To
replay an event, we still utilize the handler tree shown in Fig. 5 for
parsing, execution, and replaying.

4.3. Presentation Resync

It is common that during a presentation, especially one with a large
number of attendees, some users’ contents may be out of sync. For
instance, there might be a temporary network error on a user’s machine.
After the user fixes that issue and comes back later, his/her machine
does not maintain the events that took place during the offline period.
Another scenario is that during a presentation, some users may come
late and join the session after it has started, without knowledge of the
media events fired before.

Because we model each presentation state as the execution of a
stream of media events, an approach for recovery is to re-execute all the
previous events, similar to the mechanism in presentation replaying.
However, it might be very time-consuming, and more events may take
place during the tedious catch-up. To overcome the out-of-sync issue, we
implement a passive message-based resynchronization mechanism in
CWcollab, by enforcing that every media block has a state property to
store the current state of the media. For example, for an image block, the
state property stores the scaling factor of the image (how much it has
been zoomed in/out) and the position in the percentage of the image,
relative to the outer canvas container. A resynchronization request is
first initialized by a listener. After the request is routed to the presenter
in the session via the cloud service, the current image state is extracted
and sent to the target user(s) in a message of event-type resync. A state
restoration mechanism is also realized. Still taking an image block as an
example, on the user’s side, the image is zoomed to the required scaling
factor, as well as moved to the appropriate position.

Moreover, it is possible that during a presentation, not an audience
but the presenter loses the network connection. When he/she reconnec-
ts, all the previous information may have already lost. To recover the
state, the presenter needs to fetch the stream of events from the cloud
service, and replay the events sequentially. During a presentation,
considering the impact on performance, we do not store any state
information in the server. The state information is only created for resync
purposes.

4.4. Access Control

Access control is crucial in protecting the security of a collaborative
web service. For example, in CWcollab, only the presenters are allowed
to broadcast the media events fired to other attendees. To our knowl-
edge, role-based access control (RBAC) [45] model, which has been
utilized by some previous work, including [46–48], seems to be the most
appropriate model for our system. In an RBAC system, the access per-
missions are related to roles in the system. A user is granted one or more
roles to get access to resources. For example, in a database management
system, the administrator has both read and write access to the data in
the database, while a user is only granted to read the data.

We created an RBAC model, shown in a matrix representation in
Table 5, to achieve the access control. There are three roles in our model:
administrator, presenter, and listener. By default, the owner of a présen-
tation is assigned as the administrator. The major duty of this user is to
grant a presenter, who could be the administrator himself/herself. The
presenter (active collaboration subject) is the user who controls the
events that happened in the session and broadcasts them to other lis-
teners (passive collaboration subject) through the cloud service. More-
ever, the presenter can also resync the current state, if a resync request is
sent from other users. In a session, at one time, there is only one pre-
senter, whereas, in the whole presentation, there could be multiple

| Role       | Assign a presenter | Sync media events | Resync Ask to present | Ask for a resync |
|------------|--------------------|-------------------|-----------------------|-----------------|
| Admin      | ✓                  |                  |                       |                 |
| Presenter  | ✓                  | ✓                 | ✓                     | ✓               |
| Listener   | ✓                  |                  | ✓                     |                 |

Table 5
Access control matrix representation of our model.
presenters. Additionally, a listener can apply to be the presenter, and the administrator decides whether the role needs to be transferred or not.

We also add an additional constraint in access control to ensure the permissions are strictly followed according to the access control model we propose, by creating authorization mechanisms on both the client and server. Before a client sends a media event, the client-side JavaScript code scrutinizes the role of the user to check whether the user has enough authority to synchronize the event. To prevent possible security issues caused by client-side hacking, the server also keeps another authorization layer to ensure that the synchronization action is allowed and that the user has the role he/she claims. The event will not be broadcast to other target clients until all the checks are passed.

5. Evaluations

5.1. Web Application

We implemented a web application\(^1\) to provide a collaborative presentation service, including account management, materials preparation, presentation synchronization, and replay. The graphical user interfaces are implemented in web browsers. The backend service is implemented in Node.js, which is an event-driven, non-blocking, and cross-platform JavaScript run-time environment. We use MongoDB, a document-oriented NoSQL database to store presentation data, especially considering that MongoDB has a built-in JSON schema for documents. For the message broker, we choose Redis, which is an in-memory key-value database, famous for high performance. It provides a Publish/Subscribe message paradigm.

In Fig. 9, we show presentation windows of two users of different roles. The left presentation window belongs to the presenter (also the administrator), whose name is Alice. On the right side, there is a listener, Bob’s presentation window. As illustrated in the figure, there are four major components in a web presentation window: Toolbar, Materials, Add Material, and Presentation Panel.

In the Toolbar block, there are useful functionalities for a presentation. For example, a user can show/hide the chatbox component for chatting, as well as the annotation bar to add annotations on a media block. A listener is allowed to send resync requests from the toolbar. The presenter can receive the request and resync the current media state. As a result, users from different roles have various tools here. For example, in Bob’s Toolbar block, there are no recording or saving presentation mechanisms. On the other side, Alice cannot send a resync request, because she has already been the presenter. The Materials panel contains essential multimedia resources for the session. In the figure, there is a webpage, a PDF document, a video, and an image. We also allow a presenter to add a material dynamically via the Add Material component. The inserted material will also be pushed into the list in the Materials panel. Finally, the Presentation Panel displays the current presenting media. In the example, if Alice would like to present another media material, she just needs to drag the material from the Materials panel and drop it onto the Presentation Panel. The specific material insertion event will be automatically broadcast to all the listeners in the session, and the new material will be shown on their panels.

As an administrator, one is allowed to record the events in a presentation session into simple messages and store them in a database instance on the server side. Afterward, attendees can replay the whole session for a review. The replay is based on the re-execution of events, without the help of any screen recording techniques.

Moreover, based on the system we created, we study the performance of collaboration, we set up an Amazon Elastic Compute Cloud (EC2) t2.micro instance, deployed in Oregon. Two web browsers at the east coast of the US, Chrome and Firefox, are used on one MacBook Pro to act as the active and passive collaboration subject, respectively. We tested the time elapsed between the active collaboration subject triggering the play event of a video (other events are sharing similar structures) and the passive collaboration subject replaying the event. The end-to-end delay is only 48 ms of which most of network delay. CWcollab responds to user’s actions in an instant.

5.2. Comparison with Screen Sharing Tools

Currently, screen sharing products such as Zoom and Google Meet are widely used to provide basic collaboration support in video conferencing. In this section, we compare Google Meet with CWcollab from multiple perspectives. Although we didn’t evaluate other video conferencing products in details, we reckon that they share similar mechanisms for screen sharing.

We performed a series of experiments on multimedia, including videos, PDF documents, images, and web pages, to measure the network bandwidth usages in one minute of Google Meet and CWcollab, on the presenter side. By contrast to transferring both multimedia materials and events in a session (on-the-fly mode), the static materials can be loaded beforehand (preload mode), so only minimal events information is transferred in a session. We found that CWcollab requires a much lower bandwidth usage, as illustrated in Figs. 10-12. Note that the y-axis units are different in these figures.

Fig. 10 shows the network bandwidth consumption of both Google Meet and CWcollab for presenting a video. Fig. 10a shows the Google Meet network bandwidth consumption is around 60 KB/s - 100 KB/s, which is mainly for transferring video frames. However, CWcollab presents the static media-the video-in this experiment, and the network bandwidth consumption is only around 0B/s - 1 KB/s in Fig. 10b. The bandwidth consumption comparison is summarized in Fig. 10c. Similarly, Fig. 11 shows Google Meet utilizes around 60 KB/s bandwidth while CWcollab only needs a maximum 3 KB/s for presenting a PDF document: Fig. 12 shows Google Meet utilizes around 70 KB/s and CWcollab only needs a maximum 2 KB/s for presenting an image. To summarize Fig. 10-12, we can see that in the measurement of one minute, most events only require hundreds of bytes per second. The most expensive action is the free drawing event whose bandwidth usages can be as high as around 3 KB/s. However, the usages are still much lower than those of Google Meet.

What if the media materials are loaded on the fly? Figs. 13–16 summarize the network usages of Google Meet and CWcollab when materials are loaded during the presentation. Here, we noticed that even though the dynamic loading of materials can cause some peaks in the network usages (for example, there are three peaks in Fig. 13 to load video chunks), CWcollab require almost 0B/s most of the time. The peaks are due to the loading materials, while the 0B/s periods show the system is loading nothing.

We also summed up the bytes transmitted in this one minute and showed the results in Table 6. CWcollab transmits very few bytes in a presentation, especially noticing that in a collaboration session in the preload mode, in one minute, the total bytes transmitted in CWcollab are less than 10 KB compared to Google Meet consumption of several MBs, because CWcollab leverages an object-prioritized approach to capture media events and represent them in simple messages. As the web pages have to be loaded dynamically, even though loading pages costs a large amount of network bandwidth, CWcollab still requires much lower bandwidth than Google Meet.

Media file sizes impact the total bandwidth usage of both Google Meet and CWcollab. However, they do not weaken the network usage benefits CWcollab provides over Google Meet. Take the PDF media in Table 6 as an example. Case 1) comparing the on-the-fly mode and the preload mode: both Google Meet and CWcollab need to download the PDF file dynamically no matter if the sharing mechanism underneath is screen-sharing or object-prioritized events. As a result, the network usage difference between the on-the-fly load mode and the preload mode is the media file size. In the PDF example, the file size is around 2

\(^{1}\) Demonstration video: https://youtu.be/X-IWFSBv8Sc.
MB, which is the same for both Google Meet and CWcollab. Case 2) comparing Google Meet and CWcollab: the network usage difference between Google Meet and CWcollab in the same mode is the benefit an object-prioritized approach can offer. In the PDF example, the saved network usage of CWcollab compared with Google Meet is around 4 MB, which is the difference between 4.01 MB and 7.76 KB or the difference between 6.13 MB and 2.30 MB.

When presenting a media file collaboratively, whether the media file has already been loaded beforehand (the difference between the preload mode and the on-the-fly mode) has an influence on the latency for both Google Meet and CWcollab. To be specific, in the preload mode, as the media file has been loaded into the web browser beforehand, there is minimal latency to capture and synchronize the media events. By contrast, in the on-the-fly mode, as the browser needs to download the media file during a collaboration session from the file server, the latency is location-dependent, dominated by the network connection between

---

**Fig. 9.** Two users’ displays of presenting an image in one collaboration session. The presenter (also the administrator)’s windows is on the left, and a listener’s is on the right. There are four major panels in a presentation web window: **Toolbar**, **Materials**, **Add Material**, and **Presentation Panel**.

**Fig. 10.** Comparison of network usages when presenting a video with operations such as playing, pausing, and free drawing using Google Meet and CWcollab (preload mode).

**Fig. 11.** Comparison of network usages when presenting a PDF document with operations such as paging up/down, scrolling, and commenting using Google Meet and CWcollab (preload mode).
Fig. 12. Comparison of network usages when presenting an image with operations such as zooming in/out, moving, and free drawing using Google Meet and CWcollab (preload mode).

Fig. 13. Comparison of network usages when presenting a video with operations such as playing, pausing, and free drawing using Google Meet and CWcollab (on-the-fly load mode).

Fig. 14. Comparison of network usages when presenting a PDF document with operations such as paging up/down, scrolling and commenting using Google Meet and CWcollab (on-the-fly load mode).

Fig. 15. Comparison of network usages when presenting an image with operations such as zooming in/out, moving, and free drawing using Google Meet and CWcollab (on-the-fly load mode).
and uses simple messages to represent media controls. We demonstrated compared CWcollab with screen sharing tools such as Google Meet. Our discussed the implementation of architectural components. We the design methodologies of a distributed collaboration framework and multimedia system-CWcollab. It supports collaboration on multimedia.

Besides bandwidth usages, there are some other differences between screen sharing tools and CWcollab. Because the mechanism of screen sharing tools is to capture and broadcast the display, it cares nothing about what application the user is synchronizing. As a result, it is straightforward for these tools to support multimedia in web browsers and desktop environment. By contrast, our platform is web-based. To support another type of media, we need to instrument necessary control to achieve the collaboration. Additionally, CWcollab can be a supplement to current screen sharing products, considering that it provides an efficient platform to support message-based collaboration on multimedia.

6. Conclusion

In this paper, we proposed a context-aware web-based collaborative multimedia system-CWcollab. It supports collaboration on multimedia and uses simple messages to represent media controls. We demonstrated the design methodologies of a distributed collaboration framework and discussed the implementation of architectural components. We compared CWcollab with screen sharing tools such as Google Meet. Our evaluation results showed that our tool has a lower bandwidth usage than that of Google Meet. We posit that this work could serve as a uniform platform for efficient general-purpose multimedia collaboration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Greenberg, R. Bohnet, Group Sketch: A multi-user sketchpad for geographically-distributed small groups.
[2] J.C. Tang, S. Minneman, Videowhiteboard: Video shadows to support remote collaboration, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 1991, pp. 315–322.
[3] S. Elrod, R. Bruce, R. Gold, D. Goldberg, F. Halanz, W. Janssen, D. Lee, K. McCall, E. Pedersen, R. Pier, et al., Liveboard: A large interactive display supporting group meetings, presentations, and remote collaboration, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 1992, pp. 599–607.
[4] M. Goldman, G. Little, R.C. Miller, Real-time collaborative coding in a web IDE, in: Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM, 2011, pp. 155–164.
[5] D. Yue, N. Chen, B. Randels, A. Chentle, C.E. Lueckenhoff, S.J. Jackson, A. Sellen, F. Guimbretiere, RichReview +: Deployment of a collaborative multi-modal annotation system for instructor feedback and peer discussion, in: Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing, ACM, 2016, pp. 195–206.
[6] M. Wenzel, C. Meinel, Full-body WebRTC video conferencing in a web-based real-time collaboration system, in: 2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2016, pp. 334–339.
[7] K. Kim, W. Ha, O. Choi, H. Yeh, J.-H. Kim, M. Hong, T. Shon, An interactive pervasive whiteboard based on svc architecture for ubiquitous collaboration, Multimedia Tools Appl. 74 (5) (2015) 1557–1576.
[8] L.-T. Cheng, S. Hupfer, S. Ross, J. Patterson, Juxtaposing Eclipse with collaborative tools, in: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange, ACM, 2003, pp. 45–49.
[9] K.-S. Booth, B.D. Fisher, C.J.R. Lin, R. Argue, The mighty mouse multi-screen collaboration tool, in: Proceedings of the 15th annual ACM symposium on User interface software and technology, ACM, 2002, pp. 209–212.
[10] J. Wu, T.N. Graham, The software design board: A tool supporting workstyle transitions in collaborative software design, in: International Workshop on Design, Specification, and Verification of Interactive Systems, Springer, 2004, pp. 363–382.
[11] M.J. Callaghan, J. Harkin, E. McColgan, T.M. McGinnity, L.P. Maguire, Client-server architecture for collaborative remote experimentation, J. Network Comput. Appl. 30 (4) (2007) 1295–1308.
[12] J. Gallardo, C. Bravo, M.A. Redondo, A model-driven development method for collaborative modeling tools, J. Network Comput. Appl. 35 (3) (2012) 1086–1105.
[13] W3C, W3C html 5.2 recommendation. https://www.w3.org/TR/html5/.
[14] W3C, W3C WebSocket API. https://www.w3.org/TR/websockets/.
[15] C.A. Gutwin, M. Lippold, T. Graham, Real-time groupware in the browser: Testing the performance of web-based networking, in: Proceedings of the ACM 2011 conference on Computer supported cooperative work, ACM, 2011, pp. 167–176.
[16] S. Mogan, W. Wang, The impact of web 2.0 developments on real-time groupware, in: in: 2010 IEEE Second International Conference on Social Computing (SocialCom), IEEE, 2010, pp. 534–539.
[17] W. Wang, K. Finch, J. Rubart, J.M. Haake, A cooperative hypermedia approach to flexible process support for managing distributed projects, International Journal of Cooperative Information Systems 18 (03n04) (2009) 481–512.
[18] V. Pimentel, B.G. Nickerson, Communicating and displaying real-time data with WebSocket, IEEE Internet Comput. 16 (4) (2012) 45–53.
[19] R. Han, V. Perret, M. Naghshehine, WebSplitter: A unified XML framework for multi-device collaborative web browsing, in: Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM, 2000, pp. 221–230.
[20] O. Schmid, A.L. Masson, B. Hirsbrunner, Real-time collaboration through web applications: An introduction to the toolkit for web-based interactive collaborative environments (TWICE), Pers. Ubiquit. Comput. 18 (5) (2014) 1201–1211.
[21] M. Feiter, R. Strobel, T. Gross, Lightweight support for collaborative web browsing through spreadvector, in: Proceedings of the extended abstracts of the 22nd annual ACM conference on Human factors in computing systems, ACM, 2014, pp. 1339–1344.
[22] B. Chen, Z. Xiu, A framework for browser-based multiplayer online games using WebGL and WebSocket, in: Multimedia Technology (ICMT), 2011 International Conference on, IEEE, 2011, pp. 471–474.
[23] R. Ocaña, A framework for collaborative remote experimentation for a physical laboratory using a low cost embedded web server, Journal of Network and Computer Applications 34 (4) (2011) 1408–1415.
[24] J. Binda, E. Georgieva, Y. Yang, F. Gui, J. Beck, J.M. Carroll, Phamilyhealth: A photo sharing system for intergenerational family collaboration on health, in: Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing, ACM, 2018, pp. 337–340.

Table 6

|                     | Video | PDF  | Image | Webpage |
|---------------------|-------|------|-------|---------|
| Google Meet (preload)| 3.80 MB | 4.01 MB | 4.04 MB | -       |
| Google Meet (on-the-fly load) | 10.40 MB | 6.13 MB | 4.09 MB | 6.58 MB |
| CWcollab (preload)    | 6.73 KB  | 7.76 KB  | 8.80 KB  | -       |
| CWcollab (on-the-fly load) | 6.35 MB  | 2.30 MB  | 62.07 KB | 2.30 MB |

Fig. 16. Comparison of network usages when presenting a webpage with operations such as clicking links, scrolling, and highlighting using Google Meet and CWcollab (on-the-fly load mode).
[25] F. Boronat, M. Montagud, P. Salvador, J. Pastor, Wersync: A web platform for synchronized social viewing enabling interaction and collaboration, Journal of Network and Computer Applications 175 (2021) 102939.

[26] A. Kunz, T. Nescher, M. Kuchler, Collaboard: A novel interactive electronic whiteboard for remote collaboration with people on content, in: 2010 International Conference on Cyberworlds (CW), IEEE, 2010, pp. 430–437.

[27] Y.-C. Chang, H.-C. Wang, H.-K. Chu, S.-Y. Lin, S.-P. Wang, AlphaRead: Support unambiguous referencing in remote collaboration with readable object annotation, in: Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, ACM, 2017, pp. 2246–2259.

[28] C. Yan, B. Gong, Y. Wei, Y. Gao, Deep multi-view enhancement hashing for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 43 (4) (2020) 1445–1451.

[29] C. Yan, Z. Li, Y. Zhang, Y. Liu, X. Ji, Y. Zhang, Depth image denoising using nuclear norm and learning graph model, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 16 (4) (2020) 1–17.

[30] C. Yan, Y. Hao, L. Li, J. Yin, A. Liu, Z. Mao, Z. Chen, X. Gao, Task-adaptive attention for image captioning, IEEE Transactions on Circuits and Systems for Video technology 32 (1) (2021) 43–51.

[31] C. Yan, L. Meng, L. Li, J. Zhang, Z. Wang, J. Yin, J. Zhang, Y. Sun, B. Zheng, Age-invariant face recognition by multi-feature fusion and decomposition with self-attention, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 18 (1s) (2022) 1–18.

[32] W. Lloyd, M.J. Freedman, M. Kaminsky, D.G. Andersen, Don’t settle for eventual: Scalable causal consistency for wide-area storage with COPS, in: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, ACM, 2011, pp. 401–416.

[33] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau, Extensible markup language (XML), World Wide Web Journal 2 (4) (1997) 27–66.

[34] O. Ozturk, Introduction to XMPP protocol and developing online collaboration applications using open source software and libraries, in: Collaborative Technologies and Systems (CTS), 2010 International Symposium on, IEEE, 2010, pp. 21–25.

[35] T. Springer, D. Schuster, I. Braun, J. Janeiro, A.A. Loureiro, A flexible architecture for mobile collaboration services, in: Proceedings of the ACM/IFIP/USENIX Middleware’08 Conference Companion, ACM, 2008, pp. 118–120.

[36] P.V. Ghaete, H.K. Pati, Collaborative distributed communication in heterogeneous environments: A comprehensive survey, J. Network Comput. Appl. 61 (2016) 1–20.

[37] G. Wang, Improving data transmission in web applications via the translation between XML and JSON, in: Communications and Mobile Computing (CMC), 2011 Third International Conference on, IEEE, 2011, pp. 182–185.

[38] K. Maeda, Performance evaluation of object serialization libraries in XML, JSON and binary formats, in: Digital Information and Communication Technology and its Application (DICTAP), 2012 Second International Conference on, IEEE, 2012, pp. 177–182.

[39] B. Lin, Y. Chen, X. Chen, Y. Yu, Comparison between JSON and XML in applications based on AJAX, in: Computer science & service system (css), 2012 international conference on, IEEE, 2012, pp. 1174–1177.

[40] A. Kunz, T. Nescher, M. Kuchler, Wersync: A web platform for synchronized social viewing enabling interaction and collaboration using android, java me and windows mobile, in: Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual, IEEE, 2010, pp. 311–316.

[41] A. Bhardwaj, A. Deshpande, A.J. Elmore, D. Karger, S. Madden, A. Parameswaran, H. Subramanyam, E. Wu, R. Zhang, Collaborative data analytics with DataHub, Proceedings of the VLDB Endowment 8 (12) (2015) 1916–1919.

[42] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, Role-based access control models, Computer 29 (2) (1996) 38–47.

[43] L. Pearlman, V. Welch, M. Foster, C. Kesselman, S. Tuecke, A context-aware meeting room: Mobile interaction and collaboration using android, java me and windows mobile, in: Collaborative Data Analytics with DataHub, 2010 IEEE 34th Annual, IEEE, 2010, pp. 311–316.

[44] A. Prunicki, Apache Thrift, Tech. rep., Technical report, Object Computing, Inc (2009).

[45] T.-M. Gronli, J. Hansen, G. Ghinea, A context-aware meeting room: Mobile interaction and collaboration using android, java me and windows mobile, in: Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual, IEEE, 2010, pp. 311–316.

[46] A. Bhardwaj, A. Deshpande, A.J. Elmore, D. Karger, S. Madden, A. Parameswaran, H. Subramanyam, E. Wu, R. Zhang, Collaborative data analytics with DataHub, Proceedings of the VLDB Endowment 8 (12) (2015) 1916–1919.

[47] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, Role-based access control models, Computer 29 (2) (1996) 38–47.

[48] L. Pearlman, V. Welch, M. Foster, C. Kesselman, S. Tuecke, A context-aware meeting room: Mobile interaction and collaboration using android, java me and windows mobile, in: Collaborative Data Analytics with DataHub, 2010 IEEE 34th Annual, IEEE, 2010, pp. 311–316.

[49] A. Prunicki, Apache Thrift, Tech. rep., Technical report, Object Computing, Inc (2009).