teacher.js: A low-bandwidth digital Tool for Outdoor Online Teaching

Frederic Brodbeck
Digital Education
Berlin University of the Arts
Berlin, Germany
f.brodbeck@udk-berlin.de

Daniel Devatman Hromada
Einstein Center Digital Future
Berlin University of the Arts
Berlin, Germany
dh@udk-berlin.de
0000-0002-0125-0373

***Abstract*—teacher.js is a web-based digital communication and teaching tool. Developed as part of the teacher.solar project one of the main requirements was to keep both the bandwidth and power consumption low. For that reason teacher.js does not make use of screen sharing by means of video streaming (which is very taxing in terms of data volume and computation) but instead implements an event-driven code-casting system, where events in the teacher's client (e.g. user interactions causing changes in the application state) are broadcast to all other connected participants so that they stay in sync. Besides audio / voice call functionality, the application can be extended by integrating additional modules for all kinds of different purposes. Currently there are a text chat module, a presentation module, a Wikipedia module, and a collaborative text editing module. The whole software suite can run on a Raspberry Pi 4 microcomputer, allowing a teacher to be the full master of the on-line teaching platform they are using to give their courses. In comparison with BigBlueButton, teacher.js consumes significantly less bandwidth for both incoming and outgoing connections.**

Keywords—code-casting, outdoor teaching, on-line teaching, websockets, Matrix protocol, Raspberry Pi

I SCREENCASTING PARADIGM

As the COVID-19 pandemic took planet Earth by surprise, educators swiftly adopted existing tools to transfer at least a certain part of the teaching processes on-line. In countries were network infrastructure allowed it, video-conferencing tools like Jitsi, Webex, Teams, Zoom, Hangouts, Skype or BigBlueButton (c.f. [1] for comparison of these systems in educational settings) were suddenly promoted from what they used to be before the pandemic – i.e. mostly plain video-call and video-conferencing and chatting tools – to instruments of teaching and learning alike.

An important feature which lead to the swift adoption of above-mentioned tools in educational environments has been the so-called "screen-sharing" function, allowing the presentator users to broadcast content of their screen to all other participants of the on-line session. In the weeks that followed the first lockdowns, this so-called "screen-casting paradigm" (SC-paradigm) had become a de facto standard in real-time content sharing, with only few people questioning whether the SC-paradigm is the most optimal way of doing on-line teaching and if not, what alternatives exist. It turned out that the SC-paradigm is sub-optimal at least in two major regards:

- COSTLY: Encoding, streaming and decoding of Megabytes of video signal to N participants incurs significant network and processing costs and is not feasible in low-bandwith areas or in use cases were radical reduction of energy consumption is an issue.
- ASEMANTIC: The rich internal structure of many screen-casted contents (presentation slides, web documents etc.) is lost during its encoding into video signals. Thus, both transmitter as well as receiver devices are unaware of semantics and internal structure of what is being broadcasted. This prohibits viewers to execute any other interaction with the content, apart from passive consumption (e.g. students cannot click on the links displayed on a screen-casted website, cannot search for a term in a displayed text document, increase the font size, etc.).

II CODECASTING PARADIGM

Codecasting (CC, shortened form of "code-broadcasting") is the distribution and execution of program code snippets and associated data forwarded from the source (e.g. teacher) viewport to one or multiple target viewports (e.g. student browsers).

Thus, instead of SC-based systems, which transfer data encoding the visual signal, CC-based systems transfer following the CC-formula:

high-level representation (e.g. HTML, SVG, MIDI etc.) of document D

+

code to render or transform D in a manner satisfying constrains C1, C2 … CN

Note that the idea of streaming code juxtaposed to data is far from being new. From ANSI and ECMA-48 escape codes [2] shells for mobile clients [3] to graphic primitives of Virtual Network Computing (VNC) [4], the history of information and communication technologies (ICT) abounds in more or less high-level examples of the CC-paradigm.

Still, aside from some lesser-known features such as, for example, the "share external video" function in BigBlueButton which allows the presenter to point viewer browsers to an external video URL, giving freedom to move through their video on their own pace while receiving play/pause/rewind commands from the presenter; and aside from some experimental teaching concepts in collaborative coding or editing by means of platforms like Etherpad [5], CoLab or Overleaf, the CC-paradigm has not yet achieved its full potential in the domain of on-line learning.

The aim of the teacher.js project is to fill this gap by using web browser environment as the rendering engine and JavaScript as code-casted language.

III TEACHER.SOLAR & TEACHER.JS

III.A Teacher.solar

teacher.solar [6] is an experimental initiative to take teaching outside. That is in part motivated by trying to reduce the negative side-effects of spending long periods of time indoors (vitamin D deficiency, CO_2 having negative impact on the ability to concentrate and think, bad body posture from sitting in front of a computer all day, increased risk of

1 Magic Wand 0 artefact [6] and associated e-ink screens were solar-powered devices to run and/or communicate with the teacher.js software.

burn-out). At the same time it is an opportunity to reduce the carbon footprint of one's digital activities by harnessing solar energy; taking a step closer towards CO_2 neutrality. E-ink displays perform well in daylight / in the sun and in addition to that are much more energy efficient than other display technologies. Similarly employing lower-end portable devices reduce energy consumption even further. This is why both of them play a central role in the project.

III.B Teacher.js

teacher.js is the accompanying web app used in the teacher.solar project. It is taylored to the project to a certain degree in that it aims for keeping bandwidth and power consumption low. At the same time it is deliberately kept as simple as possible with the possibility to be extended in the future in mind. Instead of re-inventing the wheel as a monolithic custom application, teacher.js loosely integrates existing (open source) solutions by incorporating these existing tools and libraries as modules in the user interface of the application.

2 teacher.js interface running the Wikipedia module.

IV Concept

In line with the idea of minimizing the resources used, teacher.js refrains from using video streaming. Instead of broadcasting the instructor's screen as a video capture, the application uses an event-based system that only sends notifications about state changes (very small pieces of *JSON* data) to rest of the connected clients. Such events include moving to the next slide (in the context of a presentation), activating a different content module, navigating to a Wikipedia page or jumping to a specific section within an article, for instance.

V Hardware

Figure 1. displays photo of the Magic Wand 0 [7] artefact which the authors of this article used in the summer 2020 to give their experimental outdoor online teaching course titled "Design & Deployment of solar-powered artefact".

In its current state the hardware setup used is based on the Raspberry Pi 4 platform in combination with an optional 4G extension module for mobile data connectivity. Mobile data is associated to a fixed IP address, allowing the teacher to roam freely with their "portable server". Inversely the device can also act as a WLAN access point and DNS server for students which makes the system useful for creation of LAN zones in hybrid digital outdoor teaching scenarios. The system is powered by a solar power-bank which can be additionally charged by a solar backpack. All software needed for running teacher.js is installed and running on the Raspberry Pi itself.

VI Architecture

VI.A Server

The server component is a custom *Node.js* application fulfilling multiple roles:

A.1 Websocket server

The websocket server holds the global application state and relays state update messages between clients.

A.2 Proxy server

Frontend modules are embedded via <*iframe*> elements. The proxy server is used to get around any CORS (Cross-Origin Resource Sharing) restrictions, but also to inject custom code snippets into the page. Those are needed to enable cross-site communication between the iframe and the host application via the *postMessage()* method[1].

VI.B Audio conference

Next to the server script the device also hosts an instance of the *Janus* WebRTC server [8]. The audio conference is implemented using the official *Audiobridge plugin*[2] and JavaScript library[3].

VI.C Frontend

The frontend / GUI is a single-page web application (SPA) written in *TypeScript*. We chose the *Svelte*[4] framework over *React* as it is both smaller in size and also less computationally expensive. While React (with its *virtual DOM diffing* approach) does most of its work at runtime, Svelte's approach does the heavy lifting during the build stage, which means the computational load on the individual clients is reduced.

1 https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage

2 https://janus.conf.meetecho.com/docs/audiobridge.html

3 https://janus.conf.meetecho.com/docs/JS.html

4 https://svelte.dev/

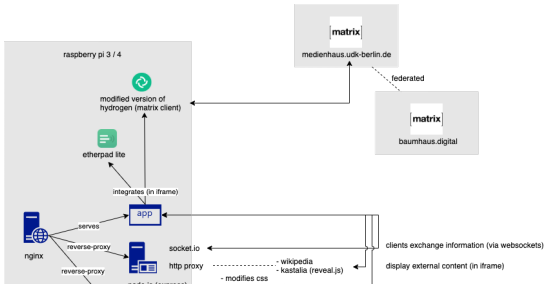

Fig. 3. teacher.js architecture

VII Modules

VII.A Chat

Text chat is implemented using the *Matrix protocol*. Of the currently available web-based matrix clients we chose *Hydrogen*[5] over *Element*[6] because it is more lightweight. The Hydrogen source code was slightly modified for the purpose of being embedded in teacher.js, most notably by adding the functionality for extracting the user's Matrix user name to be used in the teacher.js interface.

VII.B Presentation

The JavaScript snippet injected by the proxy server adds a *hook* that reacts to changes in the presentation state, which in turn notifies the host application about them. Presentations which are currently supported are web-based presentations generated by means of JavaScript frontend libraries like deck.js, impress.js or reveal.js. The backend for these presentations is our own knowledge management system (KMS) Kastalia which is to be introduced in our next article.

VII.C Wikipedia

For the purpose of making optimal use of the *screen real estate*, the layout of embedded Wikipedia articles is slightly adjusted to remove the navigation side bar and extra spacing. Furthermore the page is enhanced with functionality to detect navigation events (when links are clicked), and to determine which section of the article is currently visible in the viewport, so that the instructor can direct the students to specific parts of the article. Visited wikipedia articles and associated media files are cached locally by means of a web-server reverse proxy[7]. This allows the teacher to reuse the cached material even in an oline outdoor teaching scenario in absence of any Internet connection.

VII.D Collaborative text editing

teacher.js embeds an *Etherpad-Lite*[8] instance of the Etherpad collaborative editing tool [5], which is running on the same device. No further modifications were needed.

VII.E Replaying sessions

Optionally, there is the possibility to record a session (to be replayed at a later time). For this the audio conference is saved to a file, and relevant events are time-stamped and logged to a file on the server. In addition to this any action of the instructor relating to a public URL (such as opening a presentation, opening a chat room, navigating to a Wikipedia article, etc.) will cause a link to said URL to automatically be posted to a predefined Matrix room.

VIII Comparison: BigBlueButton vs. teacher.js

In an experimental test run we recorded the rates of incoming and outgoing data to / from the BigBlueButton server of the Magic Wand artefact running teacher.js. With 4 clients connected we simulated a short teaching session, which included common activities, such as N-to-N voice chat, going through presentation slides, as well as opening and scrolling though a few Wikipedia articles.

	title	avg. in (KB/s)	avg. out (KB/s)
0	bbb (4 clients)	52.57	281.36
1	teacher.js (4 clients)	23.30	42.76

Fig. 5. Averaged results of teacher.js / BigBlueButton comparison.

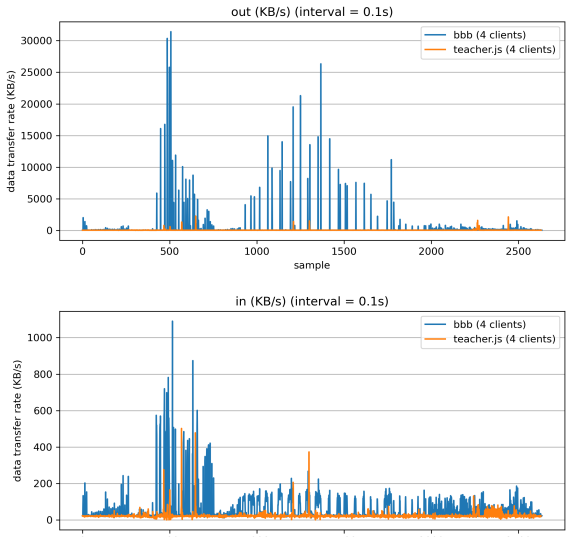

Fig. 4. Comparison of the incoming / outgoing date transfer rates

As expected, these preliminary results show a lower average data transfer rate (and therefore lower total amount of data transmitted) for teacher.js – in both the incoming and outgoing direction:

It is important to realize that should the teacher.js device store its own copy of Wikipedia or other external assets which would be subsequently relayed to student browsers during teaching sessions, the outcoming bandwidth could be even more reduced.

IX Conclusion

In this paper we introduce teacher.js, a modular online teaching tool that aims for low bandwidth consumption for use cases with restricted connectivity and power resources. We present the results of measuring the data transfer rates in a comparison with BigBlueButton, one of the established video conferencing / screen sharing tools, showing significantly less data transferred in the case of teacher.js.

More importantly, by implementing codecasting which keeps interactivity, structure, and semantics of the shown content intact, teacher.js is able to provide functionalities and characteristics which no other on-line learning system known to us currently provides. Appendix I. enumerates most salient among such "unique" functionalities and characteristics which are already implemented, Appendix II. enumerates additional ideas which will be implemented in next iteration (v0.2) of the project.

Teacher.js is available under MIT licence at our GitHub repository[9] and everyone is welcome to join us.

5 https://github.com/vector-im/hydrogen-web

6 https://element.io/

7 https://github.com/pirate/wikipedia-mirror

8 https://etherpad.org/

9 https://github.com/freder/teacher.js

ACKNOWLEDGMENT

We thank the Stifterverband and Baden-Württemberg Stiftung for the attribution of the Senior Fellowship in High-School Education which lead to emergence of the teacher.solar and teacher.js projects.

APPENDIX I. UNIQUE CHARACTERISTICS ALREADY IMPLEMENTED IN TEACHER.JS V0.1

- Whenever the teacher enters the new document D (e.g. new slide, article, article section etc.) all students viewports load D as well.
- However, when the teacher is not broadcasting any event (e.g. "load new D") students are free to roam within D, look for keywords, scroll up and down, copy parts of the document and paste them into their other apps etc.
- Students can also apply visual and design modifications of D – e.g. increase in font size, colors, font types, zooming in & out etc. - the final layout of the document D is responsive to viewer's and not the presentator's viewport.
- Mono-tasking [6, p.5] is implemented to avoid unnecessary distraction of the learning process. Chatting parallel to presentation is disallowed, it is the teacher who can bring the student browsers from presentation mode to chat mode or vice versa.
- To avoid any further distractions due to gossip or other secretive behaviours which may potentially reduce the teacher's authority or disrupt the educational process in a different way, the Matrix classroom associated to the teaching session is public and transparent for inspection to all its present and potentially also future visitors.
- The principle of *1 teacher – 1 classroom – 1 server* is applied. Similar to a teacher being responsible for everything which happens in a physical classroom during their class, the person who unlocks the door of a teacher.js space is both a teacher, administrator of the local temporary zone and keymaster in the same time.
- The session replay functionality allows one to re-enact the original on-line teaching experience even aeons after teacher or students already passed away.

APPENDIX II. UNIQUE CHARACTERISTICS TO BE IMPLEMENTED IN TEACHER.JS V0.2

- Aside teacher.js, Kastalia KMS, and Janus, the Matrix homeserver is to run directly on teacher's device. Should the corresponding chatroom be marked as "federated", its content will automatically replicate and synchronize with nodes of other teachers.
- Modules for synchronized reading and singing and other collaboratively orchestrated activities.
- Lack of video-streams from students' or teacher's webcams is to be partially resolved by modules performing client-side detection of facial keypoints, their subsequent broadcasting and dynamic avatar face synthesis.

REFERENCES

[1] N. Cavus and D. Sekyere-Asiedu, "A comparison of online video conference platforms: Their contributions to education during COVID-19 pandemic," World J. Educ. Technol. Curr. Issues, vol. 13, no. 4, pp. 1162–1173, 2021.

[2] "ECMA-48," Ecma International. https://www.ecma-international.org/publications-and-standards/standards/ecma-48/ (accessed Jan. 31, 2022).

[3] K. Winstein and H. Balakrishnan, "Mosh: An interactive remote shell for mobile clients," in 2012 USENIX Annual Technical Conference, 2012, pp. 177–182.

[4] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," IEEE Internet Comput., vol. 2, no. 1, pp. 33–38, 1998.

[5] A. Erdal and S. S. Seferoglu, "Using EtherPad for online collaborative writing activities and learners with different language learning strategies," Eurasian J. Appl. Linguist., vol. 3, no. 2, pp. 205–233, 2017.

[6] D. D. Hromada, "Teacher. solar:: open source/hardware toolbox for CO2-neutral online outdoor teaching." https://teacher.solar/entwurf.pdf (accessed Jan. 31, 2022)

[7] D. D. Hromada, "Three principles (and 2 sub-principles) for harm minimization and prevention of technological addiction in human children," Educ. Innov. Emerg. Technol., vol. 1, no. 1, 2022.

[8] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "Janus: a general purpose WebRTC gateway," in Proceedings of the Conference on Principles, Systems and Applications of IP Telecommunications, 2014, pp. 1–8.