A Decentralized and Remote Controlled Webinar Approach, Utilizing Client-side Capabilities: To Increase Participant Limits and Reduce Operating Costs

Roy Meissner, Kurt Junghanns and Michael Martin
Institute of Applied Informatics, Leipzig University, Hainstrasse 11, Leipzig, Germany

Keywords: WebRTC, Remote Control, Peer-to-Peer, Online Lecture, Webinar, Accessibility, SlideWiki, Decentralization.

Abstract: We present a concept and implementation on increasing the efficiency of webinar software by a remote control approach using the technology WebRTC. This technology enables strong security and privacy, is cross-device usable, uses open-source technology and enables a new level of interactiveness to webinars. We used SlideWiki, WebRTC, and browser speech to text engines to provide innovative accessibility features like multilingual presentations and live subtitles. Our solution was rated for real world usage aspects, tested within the SlideWiki project and we determined technological limits. Such measurements are currently not available and show that our approach outperforms open-source market competitors by efficiency and costs.

1 INTRODUCTION

A modern vision for education, that many teachers, students, professors as well as the European Union (EU)[1] shares is that education should be:

- Open
- Free for everyone
- Accessible
- Inclusive
- Modern
- Based on Open Educational Resources (OER)
- Multilingual
- High-Quality
- Engaging

The recent two decades showed that education uses more and more digital approaches. Thus more and more specialized tools for e.g. webinars appeared. Most of these tools have not been created by this vision, but fulfil some of these aspects and are thus usable for educational purposes. But they lack to project the vision holistically. In contrast, SlideWiki is an EU project about creation and management of OER and realizes all aspects of the painted vision. We chose to extend SlideWiki by a tool for educational lectures, usable for webinars and hybrid lectures[2].

We show in this paper how a remote-control approach, that focuses on utilizing client-side capabilities decreases server load and thus overall costs for providing a webinar service. This approach allows us to support much more participants than known competitors free of cost and thus enables more people world wide to join or give lectures. Our focus on a browser only solution is considered inclusive to e.g. rising countries, as modern browsers are available on all devices, even low-end ones. Furthermore the integration with SlideWiki and use of WebRTC enables us to support multilingual audiences in an inclusive way, which no competitor yet achieved. We want to also present how the technology WebRTC is utilized to remote control a browser application as of a webinar scenario. Furthermore we fill the gap for scientific measurements of resource usage and limits in this technological context. Thus our main contributions are:

- Increased participant limit of a webinar by approx. 1:10 by using a remote-control approach
- Utilization of client-side capabilities to reduce server-side workload and thus operating costs
- Measurements for three different WebRTC broadcast scenarios and of technology specific limits

At first, we introduce into the use case and observed real world aspects that influenced the design of our solution. Starting with section 2.2, we present

[1] See http://ec.europa.eu/education/policy/strategic-framework_en and linked documents

[2] A hybrid lecture is a face-to-face lecture, for which individuals are able to participate from a remote location.

Meissner, R., Junghanns, K. and Martin, M.
A Decentralized and Remote Controlled Webinar Approach, Utilizing Client-side Capabilities: To Increase Participant Limits and Reduce Operating Costs.
DOI: 10.5220/0006923901530160
In *Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018)*, pages 153-160
ISBN: 978-989-758-324-7

technical aspects of the developed tool and showcase possibilities and limits of the used technologies. In section 3 we evaluate the tool by measuring its performance characteristics as close to our targeted use case as possible. Finally, in section 4 related projects are presented, as well as a separation of our work from different existing tools and approaches.

2 PRESENTATION ROOMS

The educational material at SlideWiki is intended for lecture series at educational institutions, online courses and self education. Especially for the former two and in university courses, a typical scenario is that some educator presents a deck (a collection of slides) in a lecture style to the audience, regardless whether this is online or not. The audience is intended to follow the presentation and to ask questions about what they have seen and heard afterwards. Depending on the lecturers style sometimes also during the presentation. Lecturers often use dynamic practices like to conduct polls or to ask for suggestions that aim at including the audience and to keep it motivated. Even though many studies, like (Mason et al., 2013; Strayer, 2012; Raymond et al., 2016) promise advantages of peer learning or the inverted classroom concept, a large part of university, higher schools and online courses are held in a lecturer centred style. We thus chose to use this style as a basis for our concept. According to our own experiences, dynamic practices like mentioned above are not very well supported in today's online lecture systems. Thus these are often issued as tasks that shall be completed later on and thus no longer fulfil the presented aims. Besides these participation challenges, disabilities and language barriers make it difficult or prevent to access educational content. Especially language barriers are typically ignored by today's online lecture systems. In contrast, our approach focuses on these issues and barriers and is based on the ideas and concepts of university face-to-face lectures, extended by the advanced possibilities of online lecture systems and new technologies. We have come up with a pure web solution that is usable for face-to-face lectures, online lectures as well as for hybrid lectures.

In our approach a presenter opens a new virtual room in which a chosen deck shall be presented. Interested people may join this room via an invitation or by discovering it on SlideWiki as part of the deck overview. We decided that each room is public and not access restricted by any means. This aims at supporting libre and free teaching on a libre and free OER platform. Nevertheless, access restrictions may be easily added for other use cases. A room enables to share the presenters slide progress and voice to the participants. This may be imagined like screen and audio sharing. We have added various dynamic features for the presenter, that are intended as proof of concepts (POCs) and may be extended or altered in the future. One of these POCs is to ask the audience to complete a task and to receive live feedback about the audiences progress. Another POC is a one way chat for participants to the presenter in order to ask questions or to send in requested input. We chose this to be text based as we want the presenter to be in control of the presentation and thus to decide the point in time and if at all to react to input, like a lecturer would in a face-to-face lecture. Sent messages are only readable by the presenter, not by any other participant. Furthermore we added two features that aim at improving the accessibility of the tool, besides having an accessible user interface. One is a live transcription of the presenters voice to text, that is displayed as a subtitle to participants (also a POC). This enables aurally handicapped persons to attend presentations. The second one is about multilingual presentations and described in the paragraph below. All of the above described features are named a presenter console, that is stylized in figure 1.

As mentioned above, a room is publicly listed on SlideWiki and interested people do not need a user account on SlideWiki to join it. A participants view of a room is very similar to the presenter console (see figure 1), showing the currently presented and remote controlled deck, a subtitle, some controls as well as the one way chat. The available controls allow participants to pause and resume the remote control of the shown deck. This means that participants can switch slides individually and e.g. continue to read a slide that the presenter already left. A SlideWiki related feature is that everyone joining a room is able to choose the language of the presented deck from the available translations on SlideWiki without loosing the remote control by the presenter. So one participant may view the deck in Indonesian, another one in French, even though the presenter chose to present the deck in English. This feature is currently limited to the deck itself and we do not offer to live translate the presenters voice or the subtitle, even though this is imaginable. Participants may switch between available languages at any time while the presentation is in progress. SlideWiki focuses on a crowdsourced approach to improve and maintain decks and slides. Thus we have added links for participants to navigate to the currently shown slide on SlideWiki to e.g. improve it and to issue a change request of the improvement to the deck owner (SlideWiki feature).

If the presenter chose to leave the room, participants are still able to use it, e.g. for navigating slides, to reread the subtitle and also their issued questions, until they leave the room themselves.

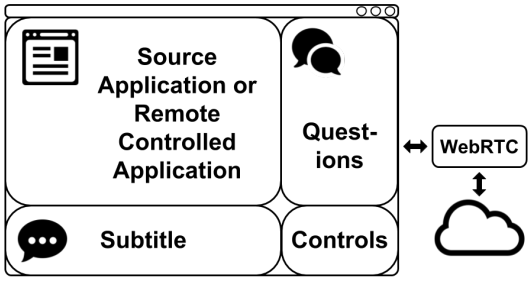

Figure 1: Components of the presenter/learner console.

2.1 Security, Privacy and Requirements

Slides and decks may be modified on SlideWiki while a presentation of the same deck is in progress. Due to the SlideWiki revision feature it is impossible to manipulate an ongoing presentation. Each update of a slide or deck on SlideWiki creates a new revision of these and does not override an existing one. If a room is opened up it is associated with a specific deck revision.

In the current state of implementation, all participants of a room create room specific data that are only temporarily saved at the participants local devices, but not on any remote servers. In case a presentation was finished and the room was left by all participants, all room specific data is discarded - even that the room existed.

Our implementation depends on a stable and working network connection of the presenter. It is currently not possible to recover a room on network issues. If the network connection is unstable and thus not usable for online or hybrid lectures, we recommend to use the SlideWiki presentation mode instead. Furthermore the used browser needs to support WebRTC.

2.2 Technical Background

We are using WebRTC as a base technology. While using this technology a presenter acts as a broadcasting[3] peer that has opened up a room on a socket.io server. Other peers may to join this room via our application. The socket.io server acts as a signaling server for WebRTC and maintains a live list of all opened rooms. This list is used to display available rooms at decks on SlideWiki. WebRTC itself negotiates via its signaling process an end-to-end encrypted peer-to-peer connection between the broadcasting peer and all other peers. We chose to establish a one way connection for audio (microphone of the presenter) and a two way connection for data, using audio streams and data channels. Thus, audience members can not speak up. The mentioned data channel is used to broadcast the current application state to connected peers. This enables them to mirror the application state of the presenter at their local machine. We use data channels furthermore to transmit the speech transcript, as well as commands for features introduced in section 2.

We have modularized our approach and are using an *Iframe* to display both, the remote controlled application at participants, as well as the source application at the presenter. In order to mirror the presenter application state and control the remote application, we are gathering emitted events of the source application, execute a postprocessing and send relevant events to peers. Received events are preprocessed and issued to the remote controlled application in order to trigger the same action that was executed at the source application. This flow is depicted in figure 2. As WebRTC data channels support to keep sent messages in order, we do not need to care about event reordering.

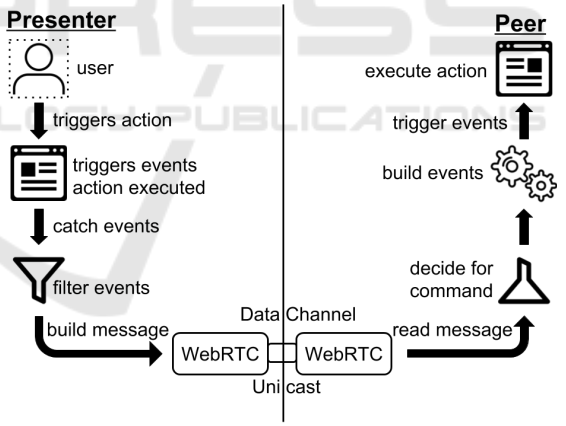

Figure 2: Remote Control Pipeline.

Iframes are due to browser security policies only controllable from their parent page if both, the parent page, as well as the Iframe originate at the same domain. Thus this is precondition to relay any events that originate in an Iframe. Due to a bug of the Blink engine[4] (used by Chromium, Google Chrome, Opera and more browsers), events must be issued as generic events. The concrete problem and solution is described as of https://stackoverflow.com/questions/45457271/forwar

[3]"broadcast" is used for simplicity reasons, even though these are several unicasts

[4]Blink engine bug about event creation: https://bugs.chromium.org/p/chromium/issues/detail?id=327853

d-a-keydown-event-from-the-parent-window-to-an-iframe-that-contains-reveal.

For speech recognition (SR) we have used the current experimental browser API, that is currently only available on Google Chrome. Chrome's SR is usable for short phrases, but is problematic for continuous input as it spontaneously stops transcoding without proper error reports. Furthermore it can not recognize punctuation characters, unless explicitly spoken.

All source-code has been implemented as part of the SlideWiki FLOSS project and is hosted at Github https://github.com/slidewiki/slidewiki-platform/tree/master/components/webrtc for WebRTC specific functionality and for the signaling server at https://github.com/slidewiki/WebRTC-Signaling-Service.

2.3 Technical Security Considerations

Besides the initial signaling process to establish a WebRTC connection, no server is involved to transmit any data between peers. The only server-side available information is the room id and presented slideshow. These information are only available for the time a room exists. The encrypted peer-to-peer approach (see (W3C, 2018)) results in a partly resistance against server-side failures and unauthorized data collection. It has no impact on already opened rooms and connected peers if the signaling service or SlideWiki itself is not reachable. The only impact is that no new rooms can be opened up and no peers are able to join existing rooms.

In case of client-side issues, an erroneous audience peer has no impact on the room or presenter peer. Such peers may just join the room again as of a new browser tab. In contrast, if the presenter-peer fails, the room will be marked as closed and the presenter needs to open up and share a new room. This has been only experienced rarely due to browser tab crashes that we expect to get less frequent in the future as the implementation of WebRTC stabilizes.

As of possible attacks, we can only imagine man-in-the-middle attacks by breaking into the server-side signaling service or SlideWiki itself. There are two possible scenarios, besides common scenarios like system intrusion:

1. Mallory manipulates SlideWiki and exchanges the link for joining a room, that is displayed at decks. Thus, every new peer connects to Mallory instead of the actual presenter. Mallory is using a custom client that is connected to the actual presenter and relays everything to Mallory's peers.
2. Mallory manipulates the signaling service and acts like in scenario 1, without exchanging the link for joining a room on SlideWiki.

Both scenarios require that Mallory breaks into a server system and manipulates or exchanges running code. In both scenarios Mallory tries to attack the connection establishment process between a presenter and peers. There exists an approach to improve the security of the signaling process, that we outline in section 5.1.

2.4 Workload Estimation

WebRTC requires to open up a new connection for every peer, for which streams are encoded specifically to the capabilities of this peer. Singh et al. showed in (Singh et al., 2013) that WebRTC chooses automatically the most efficient codec possible, which are self adapting to bandwidth limits. Thus the presenter machines upload and processing capabilities are the major limitations to the number of connected peers and thus the amount of audience members in a broadcast scenario. Furthermore the workload is influenced by the availability of hardware encoders for various audio and video formats[5], like VP8, VP9 and OPUS. As WebRTC requires to open up a new connection for every peer we expect the workload to increase linear to the number of connected peers. We determined the limits for an exemplary machine in an unrestricted network, as well as measured possible side effects. Results are presented as of section 3.

In contrast, a participant machine does not need to provide extensive capabilities, as it receives only one audio and data stream, that it needs to decode. This enables to use low-end devices, like Smartphones as peers. Their workload is also influenced by the availability of hardware decoders for the used formats.

3 EVALUATION

The presented approach is integrated with SlideWiki, has several novel features and uses experimental technologies. Thus, it might be evaluated for many different aspects, like impact on learning performance of students, workflow changes for teachers, usability for pure online lectures, and many more. All of the mentioned evaluations need to be settled on a foundation that classifies and rates the implemented by technology specific limits. This is why we focus on technological performance and limit measurements in the following subsections. Performance measurements are important in order to

[5]See the following URLs for supported codecs: https://webrtc.org/faq/#what-codecs-are-supported-in-webrtc and https://developers.google.com/web/updates/2016/01/vp9-webrtc

- show the efficiency of the approach in terms of hardware as well as software and service costs
- know in which area the software is applicable, e.g. to not schedule courses with too many students, which the software can not handle
- fill the gap for scientific measurements about the broadcast WebRTC usage scenario.

3.1 Experiment Description

We identified 5 aspects that are listed below and need to be measured. To be able to measure any aspect, a WebRTC data channel must be established (see next paragraph for the measuring method description). Measuring data transmission and audio streaming has been chosen as this matches our implemented approach. Furthermore we chose to measure data transmission, audio and video streaming as this is what all listed tools in section 4 do. All of the following aspects aim at identifying possible limits and to gain insights into their respective resource usage.

- Maximum number of simultaneously connected peers
- Signal delay for increasing number of peers
- CPU and RAM usage for data channels only for increasing number of peers
- CPU and RAM usage for data channels and audio streams for increasing number of peers
- CPU and RAM usage for data channels, audio and video streams for increasing number of peers

We have set up three different scenarios to measure these aspects - (1) determine the maximum peer number for a browser tab, the maximum peer numbers for audio only (2) and audio and video streaming (3), as well arising delays in each of the scenarios. Signal delays are measured by regularly sending the current timestamp to all connected peers via a WebRTC data channel. The first peer that connects to the presenter measures a default delay. For rising peer numbers, the measuring peer continues to compare sent timestamps to its local timestamps minus the default delay to calculate the effective signal delay (100 measurements per data point). The default delay has been measured for each scenario independently. System statistics, like CPU and RAM usage, are monitored independently by an OS specific program, called sysstat. To exclude network interference effects, we have measured all scenarios in a local network that routes packages through a maximum of 2 switches. We have executed all scenarios on a Core i7-6500U with 8GB DDR3 RAM, SATA-SSD, and Gigabit Ethernet. As browsers we use Google Chrome (version 63.0.3239.132) and Firefox (version 57.0.1) on Fedora 27 (Linux Kernel 4.14.11).

We described all needed steps to repeat the experiment inside the Readme file of the Github repository https://github.com/rmeissn/WebRTC-Broadcast-Performance-Test. This repository also contains all measured data in one of its branches. The different scenarios need minor code modifications, that are implemented and described as branches of the repository.

3.2 Results

A often cited value for the maximum number of connected peers per browser tab is 256 peers[6]. We showed in scenario 1 that the maximum number of connected peers per browser tab can be higher. We expect that there is only a soft limit determined by hardware and software capabilities, like the maximum number of simultaneous network connections. Due to hardware limitations, we were not able to measure more than 280 peers. Our scenario (1) was to disable audio and video streams, so only a data channel is opened up. This reduced CPU, RAM and network usage of the test setup to a minimum. We furthermore observed that there is no significant delay increase for rising peer numbers, no significant impact on RAM usage and only a minor impact on CPU usage, namely $\approx$21% per 200 connected peers. The exact results are depicted in figure 3. As is visible in this figure, the delay increases about linear to the number of peers.

In a second scenario (2) we tested for the maximum number of peers for audio streaming and data transmission. As the audio stream will be encoded the same number of times as peers are connected (see (W3C, 2018)), the limiting factor for the number of peer connections is the processing capability of the presenters computer. We were able to connect up to 70 peers before WebRTC started to adjust the audio quality (see network and CPU usage in figure 4). The overall delay is a little higher than in figure 3, but still low enough to speak of real-time streaming and increases on a linear scale, as in scenario 1. Due to hardware limitations at the load generating machine we were only able to connect up to 110 peers, before the machine reached its capacity. According to the results the presenter machine is able to connect more peers and we anticipate about 140 possible connections before the presenter machine runs into hardware

[6]Maximum WebRTC connections per tab: https://stackoverflow.com/questions/16015304/webrtc-peer-connections-limit or https://stackoverflow.com/questions/41194545/maximum-number-of-rtcpeerconnection/41205991#41205991 and see section 4

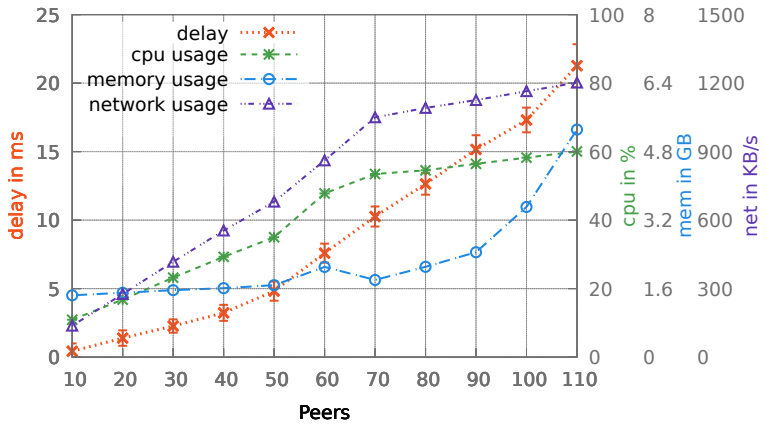

Figure 3: Simultaneously connected peers as of the *data only scenario* (1).

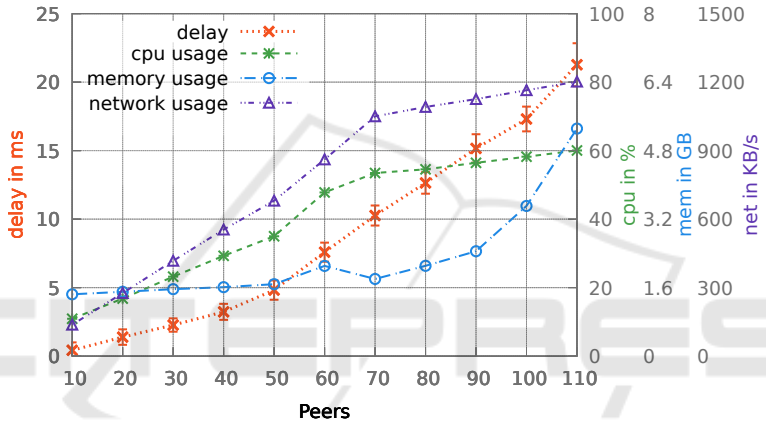

Figure 4: Measurements as of the *data transmission and audio streaming scenario* (2).

caused performance issues, e.g. because of too much memory consumption.

As of the third scenario (3) we tested for the maximum number of peers for video streaming, audio streaming and data transmission. As the video and audio streams are encoded the same number of times as peers are connected (see (W3C, 2018)), the limiting factor for the number of peer connections is the same as in scenario 2, but we expected a significantly lower number of possible connections than for scenario 2. A significant delay started to occur at 12 connected peers, as well as visible codec adjustments (see network and CPU usage in figure 5). The presenter machine reached its processing limits at 14 connected peers, resulting in visible jitter effects of the video stream at the peers.

These results show that without a TURN server[7], the first two scenarios (1, 2) are the only two scenarios that are suitable for a larger number of peers. The results also show that our remote controlling approach is superior (in terms of resource usage) to screencasting (video streaming) with the same technology, as about seven times more peers are able to connect before any codec adjustments are implied. About ten times before the presenter machine reaches its processing limits. Thus these results reveal the benefits of our chosen remote-control approach, outlined in section 2.2. Our findings also align with the results of Muaz Kahns tests, that are presented in section 4. All of the presented competing solutions from section 4 are using a video streaming approach. Some of them also use WebRTC and might be thus used in a decentralized scenario too (without a turn server). Nevertheless we expect them to match the performance characteristics of scenario 3, as they rely on video streaming. We have further discussed benefits and possible downsides of our solution in sections 2 and 4.

[7]A server that relays streams to peers

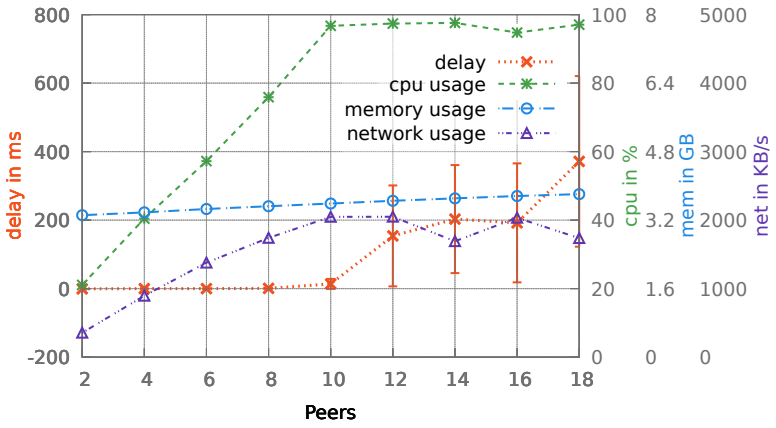

Figure 5: Measurements as of the *data transmission, audio and video streaming scenario* (3).

4 RELATED WORK

Based on the vision for modern education in section 1 we compare our solution to tools that meet three criteria: open-source, free of cost and browser-based. Video conferencing tools may be pooled into one category. The ones that match our criteria are: Jitsi, Jangouts, Spreed WebRTC, and Nextcloud Talk. There are no information available about participant limits for a and whether data is shared peer-to-peer or end-to-end encrypted for these tools, even though we assume so for the the last two as of the used technologies. Google Hangouts (that also uses WebRTC) for instance supports up to 10 participants per conference on their free plan[8]. Nextcloud Talk, Jangouts, and Spreed WebRTC need dedicated hosting of server components before these are usable, which forms a technological burden. As of accessibility and inclusiveness aspects, no special features are supported and we did not find basic accessibility support the source codes.

Another category is specialized webinar software. The only available tool that matches our criteria is BigBlueButton (BBB). BBB satisfies accessibility level A success criteria of WCAG 2.0[9]. Furthermore BBB supports all features of the former category, as well as a live whiteboard, shared notes, breakout rooms, manual closed captioning and the possibility to record a session. BBB is an exceptional webinar software and supports more features than our solution does. Nevertheless needs a dedicated hosting, which forms a technological burden. A downside of BBB is the usage of the technology Flash. Flash has been discontinued by Adobe, is regularly associated with security issues, and is not open-source. Furthermore BBB does not build upon peer-to-peer or end-to-end encrypted connections, but has a server based relay data flow.

Some research engages into the same technologies and similar approaches. Muaz Khan developed several WebRTC experiments that showcase a broadcast like solution for audio and video via WebRTC. He states at his experiments that there is a maximum peer limit of 256 peers[10] and stated via an email conversation that none of his experiments provides the possibility to broadcast a video stream to more than 10 peers, due to CPU and network limits.

Pinikas et al. include in (Pinikas et al., 2016) an approach to send commands via WebRTC data channels to peers in order to trigger functionalities. They have focused on a IoT scenario and have introduced a custom on top protocol specifically for IoT devices. They did not justify the need for a custom protocol and how this relates to their high level demo, in which they share virtual whiteboard sessions among all peers.

Zhao et al. present in (Zhao et al., 2016) a solution to stream video data via WebRTC data channels without the need to encode the stream for every peer separately, named MPEG DASH. This is lowering the CPU and memory usage of the streaming peer, but is not usable without third-party browser plugins as of July 2018.

[8]Participant limits of Google Hangout: https://productforums.google.com/forum/#!topic/hangouts/vuVoVNDFVeI

[9]BBB accessibility information: https://bigbluebutton.org/accessibility/

[10]Peer Limit: https://github.com/muaz-khan/WebRTC-Experiment/tree/master/webrtc-broadcasting

5 SUMMARY

We have presented in this paper a hybrid approach to teach online in a lecture hall style. Our tool is aligned with the vision of modern education, the EU project SlideWiki and offers educators, as well as students an alternative to expensive and proprietary competitor solutions. The tool is focused on interactive lectures, accessibility and inclusiveness, security, as well as privacy. We showed that the current implementation is suited for courses up to approximately 140 participants and we provided novel resource and limit measurement results for a WebRTC broadcast scenario. Furthermore we showed that by sending data that allows clients to replicate the same screen as the presenter sees, we extended the participant limit of a webinar about 10 times. We have tested our proof of concept to work as of various courses at participating institutions of the SlideWiki project.

5.1 Future Work

We have presented in this paper technological limits of a broadcast WebRTC usage and conceptually new ideas to webinar solutions. Thus we created a foundation to start other evaluations of the presented concept in the area of paedagogy, like outlined in section 3.

Muaz Khan (see section 4) showed that it is possible to use peers as relay peers in a broadcast scenario, effectively lowering the hardware requirements of the broadcaster as not all peers need to be directly connected to it. This eliminate the need to host a TURN server and leverage's computational power of the peer network even more. We expected this solution to increase the measured limits from section 3 tremendously, but it may need additional logic, like network optimization algorithms.

Another imaginable approach is to use a speech to text engine to transcribe the presenters voice, transmit the transcript to the peers and to use a text to speech (TTS) engine to regenerate audio output, instead of transmitting an audio stream. According to our test results, a data channel only setup is the most efficient one and thus allows the highest number of connected peers. E.g. Google presented in a recent (yet unpublished) research paper (Shen et al., 2017) a TTS engine that is mostly indistinguishable from a human voice and that might be usable in this scenario.

The mentioned signaling server is the only remaining part of the network or process that is considered central and needs to be hosted. Paik et al. showed in (Paik and Lee, 2015) a method to transfer the signaling process to a distributed hash table. This eliminates several use-case specific signaling servers and introduces a decentralized way to handle signaling. Based on their results, it seems like a promising solution to secure the signaling process.

ACKNOWLEDGEMENTS

This work was partly supported by the European Union's Horizon 2020 research and innovation program for the SlideWiki Project under grant agreement No 688095.

REFERENCES

Mason, G. S., Shuman, T. R., and Cook, K. E. (2013). Comparing the effectiveness of an inverted classroom to a traditional classroom in an upper-division engineering course. *IEEE Transactions on Education*, 56(4):430–435.

Paik, J. H. and Lee, D. H. (2015). Scalable signaling protocol for web real-time communication based on a distributed hash table. *Computer Communications*, 70:28–39.

Pinikas, N., Panagiotakis, S., Athanasaki, D., et al. (2016). Extension of the webrtc data channel towards remote collaboration and control. In *Proceedings of AmiEs '16*.

Raymond, A., Jacob, E., Jacob, D., et al. (2016). Peer learning a pedagogical approach to enhance online learning: A qualitative exploration. *Nurse education today*, 44:165–169.

Shen, J., Pang, R., Weiss, R. J., et al. (2017). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *arXiv preprint arXiv:1712.05884*.

Singh, V., Lozano, A. A., and Ott, J. (2013). Performance analysis of receive-side real-time congestion control for webrtc. In *20th International Packet Video Workshop*, pages 1–8. IEEE.

Strayer, J. F. (2012). How learning in an inverted classroom influences cooperation, innovation and task orientation. *Learning Environments Research*, 15(2):171–193.

W3C (2018). W3c editor's draft specification for webrtc. Last access time: 16 January 2018.

Zhao, S., Li, Z., and Medhi, D. (2016). Low delay mpeg dash streaming over the webrtc data channel. In *IEEE ICMEW*, pages 1–6. IEEE.