IMPROVING THE PERFORMANCE OF BIGBLUEBUTTON FOR TEACHING ONLINE COURSES

Christian Uhl and Bernd Freisleben
Department of Mathematics and Computer Science, University of Marburg, Germany

ABSTRACT
BigBlueButton (BBB) is a web conferencing system designed for online learning. It consists of a set of pre-configured open-source software tools to realize video conferencing functionality primarily for teaching purposes. Due to the COVID-19 pandemic, our university decided to roll out BBB for the university’s educational activities in the first nationwide lock-down in early 2020. Based on our experiences in deploying, operating, and using BBB at our university in the last 12 months, we present suggestions on how the services provided by BBB can be improved to meet the technical demands identified during online lecturing at our university. Our suggestions include the introduction of simulcast, improvements of encoding and muxing video feeds, and the 'Last-N' algorithm for video feed pagination. To demonstrate the benefits of the presented improvements, we experimentally evaluated most of them based on our own prototypical implementations.

KEYWORDS
BigBlueButton, Web Conferencing, Online Teaching

1. INTRODUCTION
Due to the currently prevailing COVID-19 pandemic and the disease control measures that have been put in place, many people are forced to abandon their accustomed behaviors and avoid human contact. For this reason, telepresence is increasingly used in workplaces, but also in teaching at schools and universities. This has led to an increased demand for web conference platforms to support distance teaching (de Campos). Due to a nationwide lockdown, our university had to provide a tailor-made web conference platform for teaching to all lecturers and students within less than a month, so that teaching activities could continue online as soon as possible.

The requirements within a university are quite diverse, depending on the particular teaching concept. For example, traditional top-down teaching based on lectures focuses on a single lecturer, with students taking a primarily passive stance. Such forms of teaching are also designed for large numbers of participants. In contrast, the number of participants (per group) is significantly lower in empowerment and participatory teaching. The focus here is on direct communication between participants, who often wish to see each other at all times. Thus, the following requirements have to be considered for distance teaching at a university:

- There are online lectures with more than 100 participants, in which at least one person uses a camera and a slide presentation.
- There are video conferences with over 30 active participants, in which each person shows a video of himself or herself.
- Some participants might use aged hardware or mobile devices with limited resources.
- There should be as little data traffic as possible.

In most cases, the hardware used by the participants is privately owned, which means that dedicated support for participants with weaker client hardware must be provided. Furthermore, the typical Internet connections of many home networks often have low bandwidths and/or low data volumes. This is especially relevant when a participant is connected via a mobile carrier. Currently, web conferencing providers such as Microsoft, Zoom, and Cisco are trying to expand their reach in teaching using proprietary tools. In the field of open-source software, popular projects are Jitsi (Jitsi Developer Community) and BigBlueButton (BBB) (BigBlueButton Developer Community). Both are based on the WebRTC (Web Real-Time Communication)
protocol (Jansen, Goodwin und Gupta), (Zhang, Zhang and Qi), (Xue and Zhang). Unfortunately, none of these systems meet all of our requirements without restrictions. Nevertheless, our university decided to rollout BBB for the university’s educational activities in the first nationwide lock-down in early 2020. To ensure fail-safety and thus undisputed teaching, our university decided to make WebEx from Cisco available to the lecturers in addition to BBB, but the university management recommends BBB as the primary system for distance learning.

Based on our experiences in deploying, operating, and using BBB at our university in the last 12 months, we present suggestions on how the services provided by BBB can be improved to meet the technical demands identified during online lecturing at our university. Our suggestions include the introduction of simulcast, improvements of encoding and muxing video feeds, and the ’Last-N’ algorithm for video feed pagination. In addition, we discuss some proposals from the community that are promising to improve the performance of BBB with relatively little effort. To demonstrate the benefits of the presented improvements, we experimentally evaluate most of them based on our own prototypical implementations.

The paper is organized as follows. Section 2 reviews related work. Section 3 briefly describes BBB. In Section 4, we present usage statistics. Section 5 discusses our proposed client- and server-side improvements of BBB and evaluates their performance properties. Section 6 concludes the paper and outlines areas for future work.

2. RELATED WORK

There are several publications that evaluate functional aspects and performance issues of web conferencing systems, in particular BBB. For example, (Vasconcelos, de Araújo Freitas and Marques) use KVM and OpenVZ as virtualization platforms to deploy conference systems using BBB. The authors explore BBB’s virtual performance under a real-world workload and a set of benchmarks that stress different aspects such as computing power, latency and memory, I/O, and network bandwidth. (Cižmešija and Bubas) present an approach to evaluate the use of BBB in e-learning. The authors combine established information system success criteria with models of usability and user experience in software use. (Lu, Zhao and Kuipers) evaluate four representative video conferencing applications with respect to various aspects, including traffic load control and load balancing algorithms, traffic shaping policies, and adaptively re-encoding streams in real time to limit the overall traffic. (Byrne, Furuyabu und Moore) analyze performance issues of Google Hangouts and Jitsi Meet when simultaneously using several video conferencing tools. (Petrangeli, Pauwels and van der Hooft) show that forwarding selective streams and dynamically adjusting bit rates on the fly leads to performance improvements of up to 15%.

Furthermore, several reports on developing and using web conferencing tools in online learning and teaching have been published in the literature. For example, (Sandars, Correia and Dankbaar) present a compendium of key principles and practical recommendations that allow educators to rapidly migrate to online learning during the COVID-19 pandemic. (Voegeli, Clark and Pullen) use web technologies to build the MIST/C open-source multimedia Internet client to support teaching in the classroom and online simultaneously. (J. M. Pullen) describes his experiences in synchronous online teaching and learning based on his own open-source software system called NetworkEducationWare. (Pullen and Clark) combine synchronous and asynchronous approaches to distance education, using Moodle and MIST/C, and report their experiences in supporting M.Sc. programs in computer science. None of these publications present approaches to improve the performance of BBB with respect to client and server improvements based on experiences of using BBB for teaching online courses.

3. BIG BLUE BUTTON

BBB (BigBlueButton Developer Community) is a collection of open-source software tools that implement services to provide a web conferencing platform as a local and/or on-premise web service. In particular, BBB relies on FreeSwitch (SignalWire) to provide basic phone integration, Kurento (Kurento) as a media server, LibreOffice (LibreOffice Foundation) for presentations, and Etherpad (Etherpad Foundation) for collaborative text editing. BBB itself is transmitted to the user as a web application via a web browser. The
web browser connects to the physical audio and video devices via common communication interfaces and transmits the information to a media sharing server, which then passes it on to all other participants without conversion via an integrated media server. The underlying standard for service delivery has been defined by the W3C under the name WebRTC (Jansen, Goodwin und Gupta), (Xue and Zhang), (Zhang, Zhang and Qi).

A main strength of BBB is that a conference participant is not forced to install any software on his or her local machine. This strength is also BBB’s biggest weakness, because web browsers can only communicate with the available hardware via supplied interfaces and usually require more local resources than a native application. Native applications can be optimized for the underlying hardware to improve the overall experience and implement features that are impossible to recreate inside a browser window. Furthermore, not every browser offers the same subset of functions and displayable formats. The general limiting factors of BBB (and generally all other video conferencing systems) belong to three major categories:

- **Traffic / maximum data throughput.** In conferences where each participant shares his or her own camera or media device, the amount of data to be transmitted increases quadratically with the number of total participants. Due to the fact that the data streams from the underlying media server (here a multi-point conferencing unit, MCU) are only distributed but not transformed, a large amount of traffic can be generated.
- **Client load.** The underlying technology of the web browser requires that each video is rendered in a separate subtask. This increases the required load on the end user devices proportionally to the number of participants and their quality settings. Even modern computers with considerable computing power quickly reach their performance limits.
- **Server load.** The administration, preparation, and delivery of video streams require sufficient computing power on the server. BBB is optimized by default to keep the server load as low as possible.

### 4. USAGE STATISTICS

To quantify the adoption of web conferencing systems for online teaching in our university, we collected usage data for BBB and WebEx. We set up a BBB installation consisting of up to 15 worker nodes and two load balancers. We started with BBB Version 2.2.3 and currently use BBB Version 2.3.3. The BBB instances were monitored using Prometheus and analyzed using Grafana. WebEx stores all necessary usage data by itself, eliminating the need for using external software for collecting data. The corresponding data has been exported via the WebEx admin panel and analyzed using Microsoft Excel. Since we apply a strict data minimization policy, only completely anonymized data is recorded. This prevents information about the organizer or participant from being collected. Therefore, we cannot distinguish the purpose for which a web conference has been opened. Thus, we primarily look at the number of concurrent web conferences, the number of web conference sessions in total, and the utilization of physical hardware.

#### 4.1 Data

Currently, our university consists of 4,576 paid employees and 24,394 active students. These are combined as "staff". We recorded data during the period from 08/01/2020 to 01/25/2021. To interpret the data, it is important to know that the regular operation at our university took place from 10/12/2020 to 02/12/2021. Furthermore, there is an activity free period from 12/18/2020 to 01/08/2021.

#### 4.2 Adoption

Over the course of the monitored period, a total of 75,071 web conferences with 611,585 participants were held using BBB and WebEx. It should be noted that these are not unique participants, since, as an example, reconnecting or participating in two web conferences at the same time cannot be attributed to a single person due to the need for data sparsity. The number of web conferences conducted with BBB during the entire time period was significantly higher than the number of WebEx conferences. There are 3.27 BBB attendees for every WebEx attendee, and for every web conference held in WebEx, there are 4.36 BBB conferences. The
number of participants in the regular, lecture-free, and weekend periods together with the percentage of staff per day is shown in Table 1. The average conference duration was 59.01 minutes. This includes test connections to the web conferencing system. Assuming that all web conferences under 5 minutes or with less than two participants are tests and removing them from the data set, the average conference duration increases to 73.98 minutes. The maximum values of our BBB cluster per day are shown in Table 2. Here, a distinction is made between the number of meetings and participants, the active webcams, voice participants, and listeners.

<table>
<thead>
<tr>
<th>Description</th>
<th>Number</th>
<th>Description</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>meetings</td>
<td>1,233</td>
<td>active camera</td>
<td>1,352</td>
</tr>
<tr>
<td>participants</td>
<td>18,748</td>
<td>active audio</td>
<td>5,233</td>
</tr>
<tr>
<td>concurrent meetings</td>
<td>202</td>
<td>listener</td>
<td>11,741</td>
</tr>
<tr>
<td>concurrent participant</td>
<td>2,697</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Maximum values in BBB (per day)

4.3 Server Load

To evaluate the server load, only the values of BBB are considered. We cannot assess WebEx’s server utilization because the underlying hardware infrastructure is not under our control and the usage information is not open to the public. We used 10 servers with Ryzen 7 3700X (8-core) and 64 GB RAM as our BBB test cluster. The web meetings were load-balanced between these servers using the software Scalelite. The CPU load was recorded with Netdata and normalized based on the available cores. The data visualized in Figure 1 shows a roughly linear slope of the required CPU load with increasing numbers of concurrent meetings / participants. This makes it very easy to estimate the resources required, even for large clusters.

![Figure 1. Performance of BBB in terms of CPU usage](image)

5. IMPROVING THE PERFORMANCE OF BBB

We now present suggestions on how the services provided by BBB can be improved. To demonstrate the benefits of the presented improvements, we experimentally evaluated most of them using our own prototypical implementations. In our experimental evaluation, we used the following client and server devices:

**Server**  Intel Xeon E2246G 64 GB RAM, Debian Buster

**Client**
- Macbook Pro 13" [2020] Core i5-1038NG7, 16 GB RAM, MacOS 10.15
- Ryzen9 3900X, 64 GB RAM AMD RX5700XT, Ubuntu 20.04
- Ryzen7 4750G, 32 GB RAM NVIDIA GTX1070, Windows 10
- iPhone 11 Pro, iOS 14
- Sony Xperia Z3C, Android 10
5.1 Client Performance Improvements

5.1.1 Pagination of Video Feeds

To improve the performance of a client, we can specify that never more than \( N \) videos should be displayed at once. If more video streams exist, pagination occurs and the user needs to scroll through them. This can be adjusted at the client and reduces both the load on the client and the number of data streams on both the client and the server.

*Implementation*. A simplified version of this feature was introduced in BBB in version 2.2.23 on August 26, 2020. It has fixed values and cannot be customized from within the client (BigBlueButton Developer Community). Based on this implementation, we rolled out a customized variant with \( N=8+1 \) video feeds per page in our live system.

*Results*. Due to the fact that only in rare cases more than 8 video feeds are used for teaching online courses at our university, no difference could be observed in the overall performance of the university wide cluster. In an individual test, the maximum data rate was capped at 1.081 kb/s and the CPU load reached a maximum at 63%, due to the upper limit of simultaneously displayed video feeds. A disadvantage of pagination with a static upper limit is a massive reduction of the comfort of use. The user must manually instruct the client which videos he or she wants to actively view by selecting the corresponding page. Therefore, the feature is not practicable as a single performance improvement. In combination with further improvements, such as a Last-N algorithm for calculating the most active or recent video feeds (as discussed later), the loss of comfort can be significantly reduced.

*Summary*. Our implementation demonstrates that pagination of video feeds can significantly reduce the load on the client and at the same time can reduce traffic. The BBB developer team should improve their simplified implementation of pagination of video feeds introduced in BBB version 2.2.23.

5.1.2 Simulcast and Scalable Video Coding

Simulcast (Bouras, Kioumourtzis and Gkamas) relies on the fact that each client makes its video available to the media server in various formats and/or resolutions. The server decides individually which client gets which stream. In this way, the data transfer rate and the computational effort on the receiving clients can be greatly reduced. As an alternative to simulcast, a single video data stream can be divided into several layers via Scalable Video Coding (SVC) (Schwarz, Marpe und Wiegand). These layers depend on each other and are used to improve the quality of the video stream (W3).

*Implementation*. We developed a standalone implementation of simulcast using mediasoup (Mediasoup Developers) that uses similar technologies as BBB. This prototype was fed with three prerendered video feeds from the same video source (4K video pre-down-scaled to 1080p at 60 FPS (Peach Open Video Project) served through a virtual webcam driver from OBS (OBS)) to 720p30 FPS, 480p25 FPS, and 160p10 FPS. The clients were set up to submit a fixed performance score to pick the most appropriate feed. In our evaluation of SVC, we used the most widely supported implementation based on the H.264 codec. We modified the spatial (resolution) and temporal (frame rate) parameters of the codec on the fly. The quality for decoding was set to auto.

*Results*. Figure 2 shows the performance of using BBB on a MacBook Pro (2020) (a) without simulcast, (b) with simulcast on CPU and VA-API (Intel), and (c) using SVC. The x-axis shows the number of streams; the y-axis shows the MacOS system load. Since most conferences in our university have less than 10 active camera feeds, the effort for simulcast did not pay off in our measurements. Using the CPU to reencode the video feeds individually from the same source resulted in very high system loads. We were unable to finish the tests due to a reoccurring computer freeze whenever we exceeded 32 streams. Simulcast only became practical when hardware acceleration with the VA-API interface of the integrated graphics card was used and more than 20 video feeds were served. This is the break-even point of this setup. The outsourcing of the computations to the server caused a strong reduction of the maximum number of simultaneous users of our test system from about 200 to approximately 20 users. However, reencoding the videos can shift the load from the client to the server. This is beneficial if the clients are mobile devices with limited resources. In our tests, using SVC had a positive effect on performance. The disadvantages of computing a multi-stream were nearly eliminated, while the advantages are still present.
Summary. Our implementation demonstrates that the introduction of simulcast and SVC shifts some of the load from the receiver to the sender or the server. With many simultaneous participants, a significant reduction in traffic and client load can be achieved. To introduce simulcast and SVC in BBB, parts of the BBB server code have to be rewritten.

![Figure 2. Performance of BBB (a) without simulcast and SVC, (b) with simulcast, (c) with SVC](image)

5.1.3 MCU / Multiplexing

Using appropriate multiplexing (muxing) (Mekuria, Fennema and Griffioen) servers, it is possible to merge several video streams into one. For this purpose, the video streams are arranged into a grid model and then a new (high-quality) video stream is calculated from it. Instead of distributing the original video data, the new stream is distributed to the clients. This leads to a significant reduction of the load on the end devices. A further advantage is that such a data stream can be distributed via various media servers or content delivery networks. If placed within a local area network, it can be used to distribute the content within this network and reduce the overall network load on the wide area network interface. The Kurento media server used in BBB can be switched from the current SFU (Selective Forwarding Unit) mode to MCU (Multipoint Conferencing Unit) mode required for muxing.

Implementation. We developed a standalone MCU implementation using mediasoup (Mediasoup Developers) that uses similar technologies as BBB. In this implementation, we forwarded the source material using RTP to FFmpeg (FFmpeg), which merges and reencodes the received feeds to a newly mixed, grid-oriented video. This video is then reinjected into mediasoup as the primary video feed.

Results. The recomputation of a video is a resource hungry process for the media server. This approach suffered from similar performance issues as the reencoding required in simulcast described above and is only practicable when hardware acceleration is available. This approach is not practical for conference systems with multiple concurrent conferences, since even hardware accelerated systems can only run a limited number of concurrent de-/encoding tasks. Since several data streams are combined at the same time, the load on the server is lower than in the multi-stream approach described in the previous subsection, because only one reencoding and multiple decoding processes are required per conference, resulting in roughly 12 concurrent conferences. In our tests, the latency increases noticeably. Even under optimized conditions, an additional average offset of 522 ms was obtained. Since only a single video had to be displayed regardless of the number of participants, the load on the client decreased considerably. We only performed a limited number of measurements, since this approach is not feasible for our cluster. The additional latency as well as the elevated requirements on the server hardware were not acceptable.

Summary. Our implementation demonstrates that MCU increases the load on the server, but noticeably reduces traffic and client load. To introduce MCU in BBB, parts of the BBB server code have to be rewritten.

5.1.4 Dynamic Profiles

When entering a video conference in BBB, users are asked in which quality the video signal should be made available to the other participants. The profiles include general conditions such as resolution, frame rate, and data rate. One possibility to ensure stable operation of the service even at higher workloads is to dynamically adapt these parameters. For example, the quality of low-quality video signals can be reduced even further when the server is under a higher load. As a trigger for an adjustment, the load values of the server can be included. Examples include the statistical evaluation of lost data packets, client load, processor load, and network load.
Implementation. A simplified implementation, which is exclusively based on the number of concurrent video streams on the server, was introduced in BBB in version 2.2.22 on August 11, 2020 after a suggestion by us in the official BBB bug tracker (BigBlueButton Developer Community).

Results. The current implementation is designed to keep the server up and running and reduces quality if the server load exceeds the maximum limit. We measured a noticeable reduction in peak loads on the university’s live system. Previously, a node was fully utilized about once a month. Using our implementation, the load of a node on the live cluster leveled off at 90% maximum.

Summary. Dynamically adjusting the broadcast profiles has a significant effect on reducing traffic but causes a noticeably inferior video quality. The current implementation available in BBB has too few triggers. To implement more triggers as suggested by us in the official BBB bug tracker, larger parts of the BBB source code must be modified.

5.1.5 Last-N Algorithm

The available computing capacity on both the client and server can be improved by displaying only the last $N$ users (Grozev, Marinov and Singh). BBB already contains an indicator that highlights the last speaking persons. This function can therefore be chosen as a starting point for further improvements. Some examples are:

- **Throttling.** If the user has not communicated in the conference for a long time, the user’s client can independently throttle its own bit rate by switching to a lower transmission profile. In this case, it is recommended to reduce the number of frames per second of the user’s own video stream continuously down to a fixed minimum. The bit rate itself can also be reduced steadily.

- **Quality of service.** Based on this information, the priority of the data streams can be increased on the server side for active participants and reduced for inactive participants. This ensures that available resources do not have to be kept available for inactive participants and that a lack of free resources will not lead to noticeable failures.

- **Pagination.** In combination with pagination, less relevant video streams can be removed from the other participants’ active field of view. This could also reduce the computational load of the clients.

Implementation. We manipulated the official BBB pagination using Javascript and the browser extension GreaseMonkey (Greasemonkey). Based on the activity indicator of the voice activation in BBB, we filled an array that subsequently replaced the displayed video feeds with the eight most active speakers of the last 5 minutes. The Last-N algorithm was officially introduced in BBB version 2.3.

Results. Our tests showed that the combination of a Last-N algorithm with pagination has a significant impact on the performance and stability of the overall system. Despite constant changes of the displayed video streams, the additional load on the client has only increased slightly (<10%) compared to a static upper limit of simultaneous video streams due to pagination. We assume that the load increased due to the use of an external browser extension. No measurable additional load could be identified when using the feature already implemented in BBB version 2.3. In combination with SVC, an even more significant increase in performance can be expected.

Summary. The use of the Last-N algorithm has a significant impact on the usability of the pagination function and renders it more acceptable by users. From a technical point of view, there is no reason to not activate this feature.

5.1.6 Efficient Video Codecs

At present, the VP8 codec (Bienik, Uhrina and Kuba) is used for WebRTC in BBB to encode video signals. Since January 2016, its successor VP9 (Mukherjee, Han and Bankoski) is supported in WebRTC by Google Chrome (version 48+) and since 2017 in Firefox (version 46+) (Alphabet), (Mozilla). Compared to VP8, VP9 offers higher visual quality at the same data rate. VP9 requires more effort in encoding/decoding the video streams than VP8. VP9 is also not supported by some hardware accelerators. To make it even more difficult, VP9 is not natively supported on Apple platforms and needs to be rendered in software. This leads to a doubling of the CPU load on systems with MacOS. Therefore, a change to VP9 is not (yet) recommended.
5.1.7 Switching the Default View from Grid View to Speaker View

Grid view, also known as gallery view, is currently the predominant form of presentation for web conferences. In grid view, all participants are displayed in the same size and aligned on a rectangular grid. With this type of presentation, the participants have the expectation that all participants are available in full quality at all times. In combination with other suggested improvements, especially dynamic profiles, the view can be changed to a speaker centered view. In this view, the currently dominant speaker is given most of the screen space, while the other participants are only shown as thumbnails. In speaker view, the visual quality is dynamically adjusted based on the position on the screen, which reduces the load on the client.

Implementation. We used the same implementation as in our Last-N algorithm described above based on GreaseMonkey, but replaced the default presentation inside the BBB conference window with the video feed of the most frequent user to mimic a different layout. Since this test was conducted within BBB, we were unable to modify the quality settings of the video feeds on the fly.

Results. We conducted a series of tests with several test persons. These tests were completed in the presence of a psychologist and included the presentation of a handwritten note, an active discussion between four people, and a short teaching session on a chalkboard. Although no differences in video quality were present, the test persons assumed that the video feed of the primary speaker had better video quality than the other video feeds. In the test with the handwritten note, some of the test persons explicitly asked the person presenting the note to create some noise or talk to get him or her into the main view instead of zooming on the existing feed. It should also be mentioned that the speaker view was perceived as choppy by some of our test persons and was generally less desirable than the normal grid view.

Summary. Our implementation and our experiments show that speaker view encourages the choice of lower quality video codecs, and in combination with dynamic profiles can lead to a reduction of client load as well as network traffic. To introduce speaker view in BBB, only few adjustments of the BBB source code are necessary.

5.2 Server Performance Improvements

5.2.1 IP Multicast

To reduce data transfer from the server, it is possible to use IP multicast (Ratnasamy, Sylvia; Ermolinskiy, Andrey; Shenker, Scott). For this purpose, the video streams are not transmitted individually to each client, but are made available to all participants at the same time. The underlying network infrastructure takes care of the distribution of the data stream to the recipients. IP multicast has no influence on the load or the size of the data stream at the clients. Only the server benefits from the fact that data packets do not have to be addressed and transmitted individually for each client. IP multicast also requires a special configuration of the network, since most routers in the Internet do not forward multicast packets. The use of multicast also makes other optimization options (such as simulcast) more difficult and is therefore not recommended.

5.2.2 Low-latency Kernel on the Server

Operating system kernels optimized for low-latency (Belay, Prekas und Primorac) are often used for audio/video services. Ubuntu, the operating system used for BBB, offers its own precompiled kernel for low-latency tasks in the audio/video area (Canonical). The use of a low-latency kernel is therefore easy to implement by using a single command.

Implementation. We installed different kernels (regular, low-latency and real-time optimized) from the official Ubuntu back-port repository inside a virtual machine and conducted some automated tests using Selenium (Selenium) on 20 identical client machines in a computer laboratory of our university. We opened a WebRTC audio/video connection with a dummy webcam to the server and measured the latency at idle, 50%, and 100% utilization of the CPU and the network interface of the server. We also started BBB and performed a load test by increasing the load by slowly adding more clients to a single conference.

Results. We measured slightly lower latencies in audio streams (about 2 ms). However, the slight reduction of the latencies leads to a slight increase in the general server load compared to an unmodified kernel. Since the adjustments of the low-latency kernel are quite extensive and the effects on the various services are unknown, more precise measurements were not carried out.
Summary. Our implementation demonstrates that even though only minimal effort is required to install a low-latency kernel on the server, the disadvantages outweigh the advantages in our measurements.

5.2.3 Switching to IPv6

Considering that a significant part of the Internet is natively connected via IPv6, we can try to optimize WebRTC using IPv6 (Barik, Welzl and Elmokashfi). However, in the existing Internet infrastructure, there are many connections that are only connected via Dual-Stack Lite and therefore only have a shared IPv4 address or have to tunnel the IPv4 traffic via IPv6 (Chuangchunsong, Kamolphiwong and Kamolphiwong). This often leads to problems with the maximum transmission unit (MTU), since the address metadata must be appended to the data packets. This results in a higher packet loss rate. Furthermore, ping suffers. Also, users have often reported asynchrony between video and audio signals. Since BBB version 2.3, the availability of IPv6 is set as a minimum server requirement (Inc.).

Implementation. Although most of the legacy academic infrastructure at our university only supports IPv4, we rolled out IPv6 in our university BBB production system.

Results. After our IPv6 rollout, we measured significantly improved latencies on the live BBB cluster of the university, and the overall subjective quality of video conferencing increased. Our evaluations showed that the average latency of all users on the cluster over a period of one month dropped by 6 ms compared to the previous month. Outliers with over 500 ms latency have become much rarer.

Summary. Since BBB version 2.3, IPv6 is enabled as the default protocol. While BBB can still be used with only IPv4, this is not recommended in a production environment. IPv6 has shown only advantages in our tests and should be enabled at any time.

5.3 Discussion

Several of the proposed improvements applied to BBB are already available in competing commercial products such as Zoom, Microsoft Teams, and Google Hangouts. Some of them are supported by the underlying open-source components (including the Kurento Media Server) of BBB. Furthermore, some implementations are already available in other open source projects, such as Jitsi Meet. We have submitted some of our suggestions for improvement to the developers of BBB in the official bug tracker over the last year. Some of our proposals have already been implemented by the BBB developers, although not to the full extent we suggest in this paper.

To summarize, BBB has a lot of room for improvement. Not all of our proposals are useful in every situation. For example, the presented possibilities for improvement are only designed for regular web conferences and lead to unforeseen problems with handicapped conference participants. Conferences conducted in sign language in particular are made more difficult or even get impossible by some of our improvements. Therefore, the practicability of each suggested improvement must be evaluated individually for each application scenario.

6. CONCLUSION

Based on our experiences in deploying, operating, and using BBB at our university in the last 12 months, we presented suggestions on how the services provided by BBB can be improved to meet the technical demands identified during online lecturing at our university. Our suggestions include the introduction of simulcast, improvements of encoding and muxing video feeds, and the 'Last-N' algorithm for video feed pagination. To demonstrate the benefits of the presented improvements, we experimentally evaluated most of them based on our own prototypical implementations. There are several areas for future work. For example, the impacts of our improvements should be determined in tests with a significantly larger number of test devices and/or test constellations. Furthermore, most of our tests are based on prototypical implementations, which may perform differently if implemented inside BBB. Finally, the feasibility of further improvements inspired by the functionality of other web conferencing systems should be investigated in the context of BBB.
ACKNOWLEDGEMENT

We would like to thank our university computer center and especially Andreas Gabriel for the ongoing support in approving the implementation and evaluation of the presented suggestions for improving BBB.

REFERENCES


Tag 2.2.22 <https://github.com/bigbluebutton/bigbluebutton/releases/tag/v2.2.22>

Tag 2.2.23 <https://github.com/bigbluebutton/bigbluebutton/releases/tag/v2.2.23>


135


