IADIS International Journal on Computer Science and Information Systems
Vol. 17, No. 1, pp. 1-18
ISSN: 1646-3692

PERFORMANCE IMPROVEMENTS OF BIGBLUEBUTTON FOR DISTANCE TEACHING

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 for about 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, 1999). 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, 2021) and BigBlueButton (BBB) (BigBlueButton Developer Community, 2021). Both are based on the WebRTC (Web Real-Time Communication) protocol (Jansen et al., 2017), (Zhang et al., 2019), (Xue and Zhang, 2016). 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 of 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 for about 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 et al. (2016) 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 (2020) 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 et al. (2010) 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 et al. (2020) analyze performance issues of Google Hangouts and Jitsi Meet when simultaneously using several video conferencing tools. Petrangeli et al. (2018) show that forwarding selective streams and dynamically adjusting bit rates on the fly lead 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 et al. (2020) present a compendium of key principles and practical recommendations that allow educators to rapidly migrate to online learning during the COVID-19 pandemic. Voegeli et al. (2016) use web technologies to build the MIST/C open-source multimedia Internet client to support teaching in the classroom and online simultaneously. Pullen (2006) describes his experiences in synchronous online teaching and learning based on his own open-source software system called NetworkEducationWare. Pullen and Clark (2011) 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 enhance the services of BBB with respect to client and server improvements based on experiences of using BBB for teaching online courses.

3. BIGBLUEBUTTON

BBB (BigBlueButton Developer Community, 2021) 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, 2021) to provide basic phone integration, Kurento (Kurento, 2021) as a media server, LibreOffice (Libre Office Foundation, 2021) for presentations, and Etherpad (Etherpad Foundation, 2021) for collaborative text editing. BBB itself is transmitted to the user via a web browser. It 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 et al., 2017), (Xue and Zhang, 2016), (Zhang et al., 2019).

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 performance 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 1. Participant data

	regular	lecture-free	weekend
Sum of participants	558,874	13,709	39,002
% of staff per day	17.44%	2.24%	6.38%

Table 2: Maximum values in BBB (per day)

Description	Number	Description	Number
meetings	1,233	active camera	1,352
participants	18,748	active audio	5,233
concurrent meetings	202	listener	11,741
concurrent participants	2,697		

The observed time period includes both the summer semester and the winter semester of 2020/21. Due to the pandemic situation in the summer semester of 2020, the service had to be introduced quickly and with little preparation of the teaching staff, i.e., the acceptance of video conferencing as an alternative to attendance-based teaching was not as high as first assumed. In the following winter semester 2020/21, we observed an increase in user numbers for BBB from an average of 1,121 concurrent users to an average of 6,687 concurrent users within the lecture hours (Fig. 3). We also observed an increase in user numbers for our WebEx offering, from an average of 208 concurrent users to an average of 2,044 concurrent users.

The uniformity of participants over the semester suggests that a switch of the teaching method to web conferencing has occurred only rarely within the same semester. This behavior was also confirmed by checking the user numbers of our WebEx offering.

It is noticeable that the number of participants declined slightly over the course of the semester and only jumped shortly at the end of the semester, while the overall number of conferences remained relatively constant.

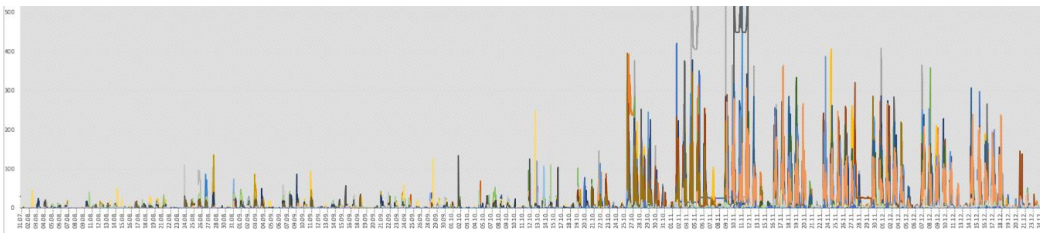
Figure 3. Number of conference participants from July 2020 to December 2020

4.3 Servers

4.3.1 BBB in Linux Containers

Despite the fact that we have a virtualization cluster based on VMware in our university, our plan was to roll out BBB in the form of Linux containers on dedicated, smaller servers. One of the main reasons for this approach was that the latency of network-heavy services based on many smaller packages often increases significantly within virtualized environments (Ferrer, 2021).

Linux containers also have the advantage that applications can directly access the hardware, i.e., CPU extensions like onboard GPUs can be used quite easily. This is particularly relevant for encoding video streams, since the processor load can be offloaded from the CPU to the GPU (Safin et al., 2020).

Although a rollout of BBB in containers is not unusual, we were not aware of any German universities that used this method in April 2020. We were advised from many sides not to roll out BBB in containers. The reason often given was that the community itself had only little experience with side effects when running in containers. To determine the actual effects, we set up two demo clusters, one within VMware and one based on Linux containers. The tests revealed that despite comparable performance (CPU power, CPU number, RAM), the Linux containers were able to serve about 30-40% more users simultaneously. Therefore, we set up newly installed computing cluster with LXD (i.e., the Linux system container and virtual machine manager).

We made our experiences about how to run BBB inside Linux containers available to the BBB community. In the middle of 2020, BBB has been officially declared to be compatible with LXD (BigBlueButton Documentation, 2021).

4.3.2 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.

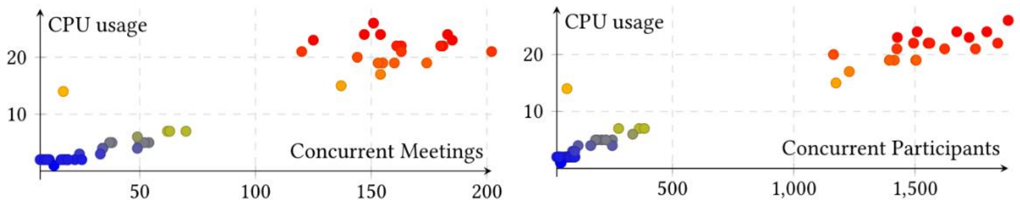

Figure 1. Performance of BBB in terms of CPU usage

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
Client	Ryzen9 3900X, 64 GB RAM AMD RX5700XT, Ubuntu 20.04
Client	Ryzen7 4750G, 32 GB RAM NVIDIA GTX1070, Windows 10
Client	iPhone 11 Pro, iOS 14
Client	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, 2021). 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 et al., 2009) 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 et al., 2007). These layers depend on each other and are used to improve the quality of the video stream (W3, 2021).

Implementation. We developed a standalone implementation of simulcast using Mediasoup (Mediasoup Developers, 2021) 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, 2021) served through a virtual webcam driver from OBS (OBS, 2021)) 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, 2021), 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.

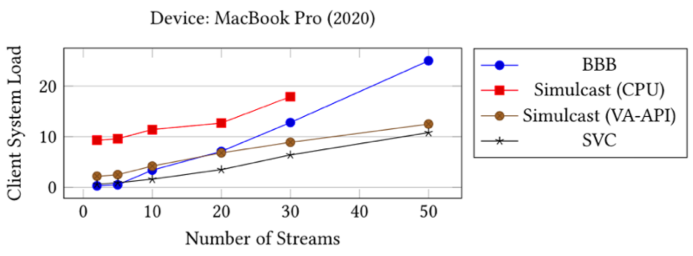

Figure 2. Performance of BBB (a) without simulcast and SVC, (b) with simulcast, (c) with SVC

5.1.3 MCU / Multiplexing

Using appropriate multiplexing (muxing) (Mekuria et al., 2016) 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, 2021) that uses similar technologies as BBB. In this implementation, we forwarded the source material using RTP to FFmpeg (FFmpeg, 2021), 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, 2021).

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 et al., 2015). 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, 2021). 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 et al., 2016) is used for WebRTC in BBB to encode video signals. Since January 2016, its successor VP9 (Mukherjee et al., 2013) is supported in WebRTC by Google Chrome (version 48+) and since 2017 in Firefox (version 46+) (Alphabet, 2021), (Mozilla, 2021). 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.

5.1.8 Implementing a Native Application on Mobile Hardware

BBB cannot access all components of the underlying hardware using a web browser. This is especially restrictive when graphics accelerators are available. Although most web browsers offer APIs to use graphics acceleration hardware from within websites, native applications can directly communicate with a graphics accelerator and therefore can utilize the graphics accelerator in a more efficient way (Hoetzlein, 2012).

However, implementing a native BBB application on mobile hardware is quite time consuming and requires frequent adaptation. The way in which data is made available differs significantly between different BBB versions, i.e., a mobile BBB app must always be updated together with the corresponding BBB version. Unlike an update of a web interface, an update of a native app on every client would interrupt the usability of the web conferencing service until the update is completed. Furthermore, possible evaluations of the app by the app store operators might significantly delay a short-term rollout. In addition, a transition period must be adhered to, in which the end users have time to update to a new version. Thus, implementing a native BBB application on mobile hardware requires a significant development and maintenance effort. On the other hand, most commercial web conferencing tools provide a mobile app, and thus a mobile BBB app might be highly desirable to enhance competitiveness of BBB.

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 et al., 2006). 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 foreach 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 IP 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 et al., 2017) 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, 2021). 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, 2021) 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 et al., 2018). 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 et al., 2014). 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 (BigBlueButton Developer Community, 2021).

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.

5.2.4 Switching to TCP / Enforcing TURN

WebRTC uses RTP (Real-Time Transport Protocol) (Schulzrinne et al., 2003) that is designed for UDP to transmit the required data packets. Since there is no direct transmission check, UDP provides lower latency than TCP. Missing packets can only be determined during the reconstruction of the video stream based on the meta-information of the packets themselves. In various communication channels (forums and bug trackers), it is frequently recommended to operate WebRTC via TCP only. This is supposed to considerably reduce the processor load during the reconstruction of the data streams.

Implementation. We configured BBB to use a technique called Traversal using Relays around NAT (short: TURN) (Rosenberg et al., 2010), whenever the communication using UDP fails. In this case, a server receives the UDP packets and forwards them using TCP in a continuous stream. This is generally considered as a fallback solution and has been enforced to conduct the test.

Results. The evaluation of our implementation showed that the performance differences were within the expected measurement deviation. However, the latency increased noticeably by about 5-10%. A general change to TCP is not recommended from an optimization point of view, since no impact on the processor load was measurable.

Summary. The enforcement of using a TURN server has a negative impact on the overall latency as well as traffic and is not recommended.

5.2.5 Migrating SFU from Kurento to Mediasoup

In our prototypical implementations, we generally used the SFU Mediasoup instead of Kurento due to the ease of implementation. Compared to Kurento, Mediasoup additionally has clear performance advantages (André et al., 2018).

Implementation. We have realized a large part of the previous prototypical implementations with Mediasoup. For evaluation purposes, some of the tests were repeated again with an adapted standalone version of Kurento and compared in terms of performance. We compared the maximum concurrent sessions on our server hardware.

Results. In our tests, we found that especially for larger conferences, Mediasoup delivers significantly better performance values than Kurento (Fig. 4). While smaller conferences of up to three participants allow almost five times the number of simultaneous participants, this value has increased to almost eight times for larger conferences. Mediasoup is also much faster at establishing connections and supports more modern codecs (VP9). With Kurento, our tests showed that our test server was overloaded by about 40 video participants in a single web conference. In comparison, Mediasoup was able to establish conferences with over 100 simultaneous video participants in a single web conference. The average connection time for establishing a video connection was reduced from 1.56 seconds to less than 0.3 seconds. This is especially beneficial when pagination is used, since the video feeds of the participants will be visible in a much shorter time.

Summary. A switch to Mediasoup requires major changes in BBB's structure, since the application interfaces differ significantly. The disadvantage of Mediasoup is its functionality. For example, functions from Kurento, such as real-time manipulation of video streams, are not readily available in Mediasoup. Since most of the functions are not required for the regular operation of BBB, these disadvantages are only of limited importance. In the current pre-release version of BBB (version 2.4 RC5), Mediasoup is officially implemented as an alternative SFU. However, the implementation is not complete, since many functions are still based on old APIs - one example is the recording function. We noticed significant improvements in this version, but the performance gain was not quite as high as in our tests. However, this may be due to the fact that Kurento is still used in addition to Mediasoup, since the implementation is not complete. We expect that a further performance gain can be measured when the switch is complete.

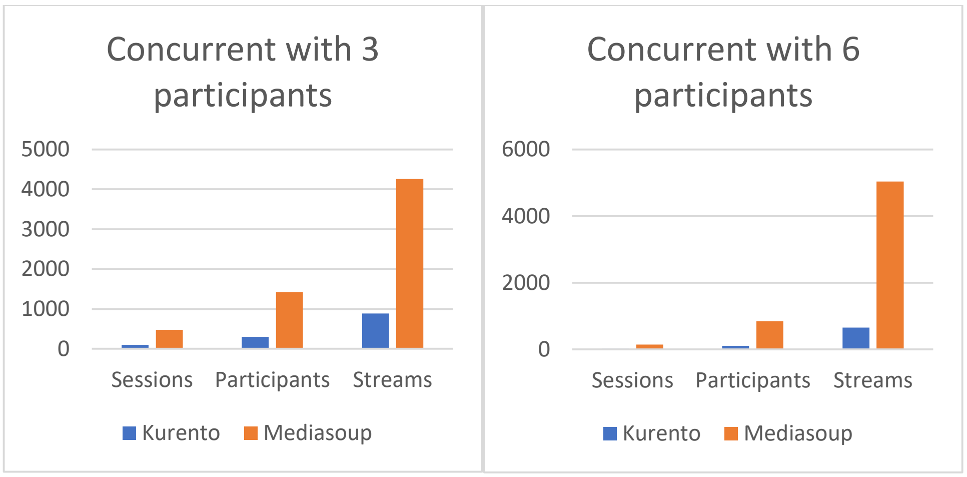

Figure 4. Performance of BBB with different SFUs

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 for about 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 Dr. Andreas Gabriel for the ongoing support in approving the implementation and evaluation of the presented suggestions for improving BBB.

REFERENCES

Alphabet. 2021 (accessed January 15, 2021). "Google Chrome Patchnotes VP9." *developers.google.com/web/updates/2016/01/vp9-webrtc.*

André, Emmanuel, Nicolas Le Breton, Augustin Lemesle, Ludovic Roux, and Alexandre Gouaillard. 2018. "Comparative Study of WebRTC Open Source SFUs for Video Conferencing." *2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, 2018, pp. 1-8, doi: 10.1109/IPTCOMM.2018.8567642.

Barik, R., M. Welzl, A. M. Elmokashfi, T. Dreibholz, and S. Gjessing. 2018. "Can WebRTC QoS Work? A DSCP Measurement Stud*y".* Vol. 01*, 30th International Teletraffic Congress (ITC 30)*, 167-175. doi:10.1109/ITC30.2018.00034.

Belay, Adam, George Prekas, Mia Primorac, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. 2016. "The IX Operating System: Combining Low Latency, High Throughput, and Efficiency in a Protected Dataplane." *ACM Trans. Comput. Syst.* 34. doi:10.1145/2997641.

Bienik, J., M. Uhrina, M. Kuba, and M. Vaculik. 2016. "Performance of H.264, H.265, VP8 and VP9 Compression Standards for High Resolutions." *19th International Conference on Network-Based Information Systems (NBiS)*, 246-252. doi:10.1109/NBiS.2016.70.

BigBlueButton Developer Community. 2021 (accessed January 15, 2021). "BigBlueButton." *groups.google.com/g/bigbluebutton-dev.*

—. 2021 (accessed January 15, 2021). "Github: BigBlueButton Tag 2.2.22." *github.com/bigbluebutton/bigbluebutton/releases/tag/v2.2.22.*

—. 2021 (accessed January 15, 2021). "Github: BigBlueButton Tag 2.2.23." *github.com/bigbluebutton/bigbluebutton/releases/tag/v2.2.23.*

BigBlueButton Documentation. 2021 (accessed August 21, 2021). "BigBlueButton Troubleshooting." *docs.bigbluebutton.org/support/troubleshooting.html.*

BigBlueButton. 2021 "BBB 2.3 Installation (accessed June 13, 2021)." *docs.bigbluebutton.org/2.3/install.html.* Accessed 6 13, 2021.

Bouras, C., G. Kioumourtzis, and A. Gkamas. 2009. *Simulcast Transmission for Video Applications: Performance Evaluation with an Integrated Simulation Environment.* Vol. 41, in *2009 International Symposium on Performance Evaluation of Computer Telecommunication Systems,* 339-346.

Byrne, Jason, Mariko Furuyabu, Jeff Moore, and Takehiko Ito. 2020. "The Unexpected Problem of Classroom Video Conferencing: An Analysis and Solution for Google Hangouts and Jitsi Meet." *Journal of Foreign Language Education and Technology* 5.

Canonical. 2021 (accessed January 15, 2021). "Ubuntu Low Latency Kernel." *wiki.ubuntuusers.de/Kernel.*

Chuangchunsong, N., Sinchai Kamolphiwong, T. Kamolphiwong, R. Elz, and P. Pongpaibool. 2014. "Performance Evaluation of IPv4/IPv6 Transition Mechanisms: IPv4-in-IPv6 Tunneling Techniques." In *The International Conference on Information Networking 2014 (ICOIN2014)*, 238-243. doi:10.1109/ICOIN.2014.6799698.

Cižmešija, Antonela, and Goran Bubas. 2020. "An Instrument for Evaluation of the Use of the Web Conferencing System BigBlueButton in E-learning." In *Central European Conference on Information and Intelligent Systems (CECIIS)*, 1-9.

de Campos, Geraldo Lino. 1999. "Minimum Requirements for Effective Distance Teaching Systems." In *Proceedings of the 4th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*, 182. New York, NY, USA: Association for Computing Machinery. doi:10.1145/305786.305926.

Etherpad Foundation. 2021 (accessed January 15, 2021). "Etherpad." *etherpad.org.*

Ferrer, Miquel. 2021. *Measuring Overhead Introduced by VMWare Workstation Hosted Virtual Machine Monitor Network Subsystem.* Paper, Barcelona, Spain: Computer Architecture Department, Technical University of Catalonia.

FFmpeg. 2021 (accessed January 15, 2021). "FFmpeg." *ffmpeg.org.*

Greasemonkey. 2021 (accessed January 15, 2021). "Greasemonkey Extension for Firefox Webbrowser." *www.greasespot.net.*

Grozev, Boris, Lyubomir Marinov, Varun Singh, and Emil Ivov. 2015. "Last N: Relevance-Based Selectivity for Forwarding Video in Multimedia Conferences." In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 19–24. New York, NY, USA: Association for Computing Machinery. doi:10.1145/2736084.2736094.

Hoetzlein, Rama C. 2012. "Graphics Performance in Rich Internet Applications." In *IEEE Computer Graphics and Applications*, 98 - 104. IEEE.

Intel. 2021 (accessed January 15, 2021). "VA-API." *01.org/linuxmedia.*

Jansen, Bart, Timothy Goodwin, Varun Gupta, Fernando Kuipers, and Gil Zussman. 2018. "Performance Evaluation of WebRTC-Based Video Conferencing." *SIGMETRICS Perform. Eval. Rev.* (Association for Computing Machinery) 45: 56–68. doi:10.1145/3199524.3199534.

Jitsi Developer Community. 2021 (accessed January 15, 2021). "Jitsi." *jitsi.org.*

Kurento. 2021 (accessed January 15, 2021). "Kurento." *www.kurento.org.*

LibreOffice Foundation. 2021 (accessed January 15, 2021). "LibreOffice." *de.libreoffice.org.*

Lu, Yue, Yong Zhao, Fernando Kuipers, and Piet Van Mieghem. 2010. "Measurement Study of Multi-party Video Conferencing." In *NETWORKING 2010*, edited by Mark Crovella, Laura Marie Feeney, Dan Rubenstein and S. V. Raghavan, 96–108. Berlin, Heidelberg: Springer.

Mediasoup Developers. 2021 (accessed January 15, 2021). "Mediasoup." *mediasoup.org.*

Mekuria, Rufael, Jelte Fennema, and Dirk Griffioen. 2016. "Multi-Protocol Video Delivery with Late Trans-Muxing." In *Proceedings of the 24th ACM International Conference on Multimedia*, 92–96. New York, NY, USA: Association for Computing Machinery. doi:10.1145/2964284.2967189.

Mozilla. 2021 (accessed January 15, 2021). "Mozilla Firefox Release Notes V46." *wiki.mozilla.org/Media/WebRTC/ReleaseNotes/46.*

Mukherjee, D., J. Han, J. Bankoski, R. Bultje, A. Grange, J. Koleszar, P. Wilkins, and Y. Xu. 2013. "A Technical Overview of VP9 – The Latest Open-Source Video Codec." In *SMPTE 2013 Annual Technical Conference Exhibition*, 1-17. doi:10.5594/M001518.

OBS. 2021 (accessed January 15, 2021). "Open Broadcast Studio." *obsproject.com.*

Peach Open Video Project. 2021 (accessed January 15, 2021). "Big Buck Bunny." *Big Buck Bunny.*

Petrangeli, Stefano, Dries Pauwels, Jeroen van der Hooft, Tim Wauters, Filip De Turck, and Jürgen Slowack. 2018. "Improving Quality and Scalability of WebRTC Video Collaboration Applications." In *MMSys '18: Proceedings of the 9th ACM Multimedia Systems Conference*, 533-536. Amsterdam Netherlands: ACM.

Pullen, J. Mark. 2006. "Scaling up a Distance Education Program in Computer Science." In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 33–37. New York, NY, USA: Association for Computing Machinery. doi:10.1145/1140124.1140136.

Pullen, John Mark, and Nicholas K. Clark. 2011. "Moodle-Integrated Open Source Synchronous Teaching." In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, 353. New York, NY, USA: Association for Computing Machinery. doi:10.1145/1999747.1999867.

Ratnasamy, Sylvia; Ermolinskiy, Andrey; Shenker, Scott. 2006. "Revisiting IP Multicast." *SIGCOMM Comput. Commun. Rev.* (Association for Computing Machinery) 36: 15–26. doi:10.1145/1151659.1159917.

Rosenberg, R. Mahy and P. Matthews and J. 2010. "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)." 1-67. RFC.

Safin, Ramil, Emilia Garipova, Roman Lavrenov, Hongbing Li, Mikhail Svinin, and Evgeni Magid. 2020. "Hardware and Software Video Encoding Comparison." In *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. Chiang Mai, Thailand : IEEE.

Sandars, John, Raquel Correia, M. Dankbaar, Peter de Jong, Poh-Sun Goh, Inga Hege, Ken Masters, et al. 2020. "Twelve Tips for Rapidly Migrating to Online Learning During the COVID-19 Pandemic." *MedEdPublish* 9. doi:10.15694/mep.2020.000082.1.

Schulzrinne, H. and Casner, S. and Frederick, R. and Jacobson, V. 2003. "RFC3550: RTP: A Transport Protocol for Real-Time Applications." USA: RFC Editor.

Schwarz, H., D. Marpe, and T. Wiegand. 2007. "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard." *IEEE Transactions on Circuits and Systems for Video Technology* 17: 1103-1120. doi:10.1109/TCSVT.2007.905532.

Selenium. 2021 (accessed January 15, 2021). "Selenium." *www.selenium.dev.*

SignalWire. 2021 (accessed January 15, 2021). "Freeswitch." *freeswitch.com.*

Vasconcelos, P. R. Magalhães, G. A. de Araújo Freitas, and T. G. Marques. 2016. "Virtualization Technologies in Web Conferencing Systems: A Performance Overview." In *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, 376-383. doi:10.1109/ICITST.2016.7856734.

Voegeli, Dorian, Nicholas K. Clark, and John Mark Pullen. 2016. "Better Online Teaching Support Using Open-Source Web Applications." In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 365. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2899415.2925489.

W3. 2021 (accessed January 15, 2021). "W3 WebRTC and SVC." *W3 WebRTC and SVC.*

Xue, Huaying, and Yuan Zhang. 2016. "A WebRTC-based Video Conferencing System with Screen Sharing." In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 485-489. doi:10.1109/CompComm.2016.7924748.

Zhang, Jie, Yingpeng Zhang, Bingfang Qi, Hui Zhao, and Toyohide Watanabe. 2019. "A WebRTC e-Learning System Based on Kurento Media Server." In *E-Learning and Games*, edited by Abdennour El Rhalibi, Zhigeng Pan, Haiyan Jin, Dandan Ding, Andres A. Navarro-Newball and Yinghui Wang, 331–335. Cham: Springer International Publishing.