UNIVERSITÄT AUGSBURG
Universitätsbibliothek

OPEN ACCESS
OPUS AUGSBURG

Delivering user experience over networks: towards a quality of experience centered design cycle for improved design of networked applications

Anika Seufert, Svenja Schröder, Michael Seufert

Angaben zur Veröffentlichung / Publication details:

Seufert, Anika, Svenja Schröder, and Michael Seufert. 2021. "Delivering user experience over networks: towards a quality of experience centered design cycle for improved design of networked applications." *SN Computer Science* 2 (6): 463. https://doi.org/10.1007/s42979-021-00851-x.

Nutzungsbedingungen / Terms of use:

CC BY 4.0

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
CC-BY 4.0: Creative Commons: Namensnennung
Weitere Informationen finden Sie unter: / For more information see:
https://creativecommons.org/licenses/by/4.0/deed.de

SN Computer Science (2021) 2:463
https://doi.org/10.1007/s42979-021-00851-x

ORIGINAL RESEARCH

Delivering User Experience over Networks: Towards a Quality of Experience Centered Design Cycle for Improved Design of Networked Applications

Anika Seufert[1] · **Svenja Schröder**[2] · **Michael Seufert**[1]

Received: 7 June 2021 / Accepted: 6 September 2021 / Published online: 16 September 2021
© The Author(s) 2021

Abstract
To deliver the best user experience (UX), the human-centered design cycle (HCDC) serves as a well-established guideline to application developers. However, it does not yet cover network-specific requirements, which become increasingly crucial, as most applications deliver experience over the Internet. The missing network-centric view is provided by Quality of Experience (QoE), which could team up with UX towards an improved overall experience. By considering QoE aspects during the development process, it can be achieved that applications become network-aware by design. In this paper, the Quality of Experience Centered Design Cycle (QoE-CDC) is proposed, which provides guidelines on how to design applications with respect to network-specific requirements and QoE. Its practical value is showcased for popular application types and validated by outlining the design of a new smartphone application. We show that combining HCDC and QoE-CDC will result in an application design, which reaches a high UX and avoids QoE degradation.

Keywords User experience · Quality of experience · Application design · Design cycle · Human-centered design

Introduction

To deliver the best user experience (UX), the human-centered design cycle (HCDC) [1] serves as a well-established guideline to application developers. It provides a set of incremental steps to ensure the incorporation of user centricity during the design and development cycle of any software system. While the HCDC considers usability and UX factors of software, it does not yet cover network-specific requirements in a sufficient way. However, these network aspects become increasingly crucial, as nowadays, most applications rely on the Internet to provide or enhance the delivered experience for end users. For example, in 2014, already 83% of Android applications in the Google Play Store required the "Full Network Access" permission, and 69% of the applications required the "View Network Connections" permission [2]. Any app requiring access to the Internet to function properly would need to have one or both of these permissions. This shows that the requirement of Internet access is already nearly ubiquitous, and will probably become even more popular with the upcoming possibilities of 5G mobile networks. Note that this requirement is not limited to smartphone applications, but applies to most of today's software applications for end users. However, applications developers usually do not pay full attention to this aspect.

In research on communication networks, where network performance was historically always measured in terms of Quality of Service (QoS), i.e., with the help of technical metrics such as throughput or packet loss, the experience with networked applications and services was formalized as the concept of Quality of Experience (QoE). QoE was introduced to describe *"the degree of delight or annoyance of the user of an application or service. [...] In the context of communication services, QoE is influenced by service, content, network, device, application, and context of use"*

✉ Anika Seufert
anika.seufert@uni-wuerzburg.de

Svenja Schröder
svenja.schroeder@univie.ac.at

Michael Seufert
michael.seufert@uni-wuerzburg.de

[1] Chair of Communication Networks, University of Würzburg, Würzburg, Germany

[2] Research Group Cooperative Systems, University of Vienna, Vienna, Austria

[3]. In contrast to UX, where the assessment of experiential qualities calls for the assessment of a range of qualities like, for example, emotion, enjoyment and aesthetics [4], QoE focuses on the experienced (media) quality of a multimedia system [5]. In the following we will only use the term application, but here we include both applications and services.

For network operators, QoE more and more came to the center of industry thinking, as it was shown that a reduced QoE results in customer churn and a reduction of application revenues [6]. Over the last years, subjective studies were conducted to develop models for various types of Internet applications, which can describe relevant factors that influence the subjectively perceived quality and satisfaction with these applications, e.g., video streaming [7]. These models prove useful for network operators in the QoE-aware traffic management cycle [8], in which the QoE of a networked application is monitored using dedicated models, e.g., [9]. When QoE degradation is detected or imminent, traffic management actions are applied in the network to mitigate the QoE degradation, e.g., [10, 11].

In addition, the perceived QoE might influence the user behavior and lead to interactions with the application. These, in turn, might impact the network requirements and network traffic of that application, which again might affect the QoE. Thus, an additional QoE cycle [8] has to be considered, which describes the interplay between applications, networks, and QoE.

Although pure network management or application-aware networks might be in place, a joint network and application management could further enhance the experience for end users [12, 13]. However, this would additionally require network-aware applications, which also consider the network-related aspects of experience. Thus, we clearly see a chance here that QoE, which focuses on a network-centered view, teams up with UX and their human-centered view towards an improved overall experience. This can be achieved by considering QoE aspects during the software development process, such that applications become network-aware and QoE-aware by design.

Having this idea in mind, this paper proposes the Quality of Experience Centered Design Cycle (QoE-CDC). This cycle resembles and complements the HCDC, and provides guidelines on how to design applications with respect to network-specific requirements and QoE. Thus, the primary goal of the QoE-CDC is to ensure that the user experience, which was created by the HCDC, is not deteriorated by network-related issues. To show the practical value of the QoE-CDC, we will discuss past and potential improvements of several popular types of applications in this paper, which could have been resulted from employing the QoE-CDC. Moreover, we will point to open research questions and missing studies with respect to the subjectively perceived experience with these applications in this context. Finally,

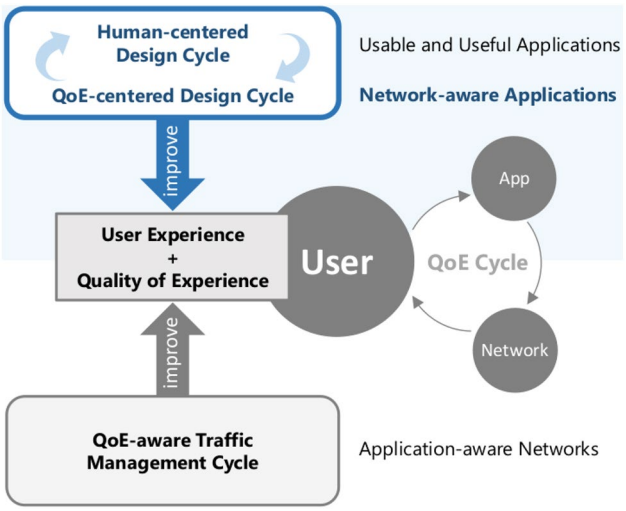

Fig. 1 Connection between the different cycles

we will validate the QoE-CDC with an app for crowdsourced video streaming QoE studies, which has been designed from scratch applying both the HCDC and the QoE-CDC.

The goal of the proposed cycle is that UX designers can follow it to learn from QoE research what network-originated degradations can occur and how they affect the user experience. With this knowledge, developers can find specific solutions to cover or mitigate possible weaknesses by skillfully adapting the design. For example, this could be possible by designs, which distract users from QoE degradation, which incorporate experience-friendly notifications in case of network problems, and which implement design concepts that weaken network requirements. Here, the UX designers and backend developers can work hand in hand to improve the application experience.

Figure 1 describes the overall picture and the connection between the above presented cycles. At the network layer at the bottom, the QoE-aware traffic management cycle [8] formalizes an application- and QoE-aware network, which aims at maximizing the QoE. At the application layer at the top, usable and useful network- and QoE-aware applications reside, which can be developed following both the HCDC and the QoE-CDC. Between these two cycles, there are strong interactions, such that both the HCDC and the QoE-CDC should be iterated when designing a networked application. In the end, a combination of both cycles will result in an application design, which reaches a high UX and avoids QoE degradation to also reach a high QoE. Finally, there will always be an interplay between application usage, network usage, and the resulting QoE for the user, which is described by the QoE cycle [8] at the right. However, together with application-aware networks, usable and useful network-aware applications could unleash the maximum experience and satisfaction for end users, which constitutes

a win–win situation for the user, the network provider, and the application provider.

The remainder of this work is structured as follows. The next section provides background information and related works on the HCDC, which our proposal complements. The four steps of the QoE-CDC are presented in the subsequent section followed by a demonstration of the practical value of the QoE-CDC. For this, several popular applications, namely, smartphone applications, video and audio streaming, live video streaming, and mobile instant messaging are discussed, and modifications to deliver improved experience are elaborated. Afterwards, the QoE-CDC is validated by outlining the design of an app for crowdsourced video QoE studies. The final section concludes this work.

Human-Centered Design Cycle

Many factors influence a user's experience with smartphone applications and thus, have to be considered when designing a new application [14]. A well-known paradigm for integrating the user's needs into the design process is human-centered design. According to the ISO 9241-210:2010 standard [1], *"Human-centred design is an approach to interactive systems development that aims to make systems usable and useful by focusing on the users, their needs and requirements, and by applying human factors/ergonomics, and usability knowledge and techniques."*. Thus, the aim is to maximize the user experience (UX) by paying close attention to the human perspective in each step of the design process.

In human-centered design, the user is clearly placed in the foreground, which is also reflected in the well-established human-centered design cycle (HCDC) [1]. Here, the design process has an initial planning stage, and is then followed by four steps:

1. **Understanding and specifying the context of use:** Identify and characterize users and stakeholders, their goals and task as well as the conditions under they will use it.
2. **Specifying the user requirements:** Identify the users' and stakeholders' needs, derive their requirements, and solving trade-offs between different user requirements.
3. **Producing design solutions to meet user requirements:** Design user tasks, interactions with the application, and the user interface.
4. **Evaluate the designs against requirements:** Conduct user-centred evaluations using user-based testing, inspection-based evaluation, and long-term monitoring.

The HCDC approach is non-linear and iterative, meaning that after each run through the four steps, the developer can jump back to any other step, according to the results of the evaluation (step 4).

Human-centered design processes are utilized in various fields for designing applications, which require high usability, e.g., in geography [15], aerospace [16], or medicine [17].

Since the HCDC is widely used and is considered the basis for usable and useful applications, extensions have been proposed in different areas. To find a good trade-off between security and usability, for example, an integration of usable security and user authentication into the HCDC was introduced in [18]. Furthermore, the application of HCDC on software development was evaluated in [17, 19], where the authors discussed the impact of the HCDC and how to combine the HCDC with a software development process. Similarly, a combination of the usability and software engineering life cycles was presented in [20]. While the importance of considering software engineering requirements in the HCDC has already been discussed in literature [21], communication networks has not been considered in human-centered design processes so far.

A major drawback of the HCDC is that it does not explicitly consider network-specific requirements although most applications connect to the Internet nowadays. This is especially evident in mobile networks, where video streaming, as well as social and messaging applications dominate the traffic volume according to [22]. With the advent of 5G, applications will increasingly rely on Internet access and require high network throughput and low latency. However, perfect network conditions cannot always be guaranteed, which can cause serious performance and experience issues. Thus, application developers of networked applications need to be aware which experience is actually delivered over the network.

Network operators and communications researchers have already acknowledged that the subjectively perceived experience with networked applications is a major business factor. The introduction of the concept of Quality of Experience (QoE) [3, 6, 23] moved the focus from the system to the user, putting the user and his perceived experience to the center of the evaluation process [24]. This paradigm shift led to the advent of so called QoE studies, i.e., studies about the impact of technical parameters of systems and networks on the experience of end users, which has produced an enormous amount of findings. The results of these studies are considered by network operators to avoid QoE degradation and improve the experience with a networked application by traffic management, e.g., [10, 11]. By also considering these findings of the QoE community during the development of new or improved applications, QoE degradation due to network issues or fluctuating network conditions can be mitigated by design, which will positively affect the user experience.

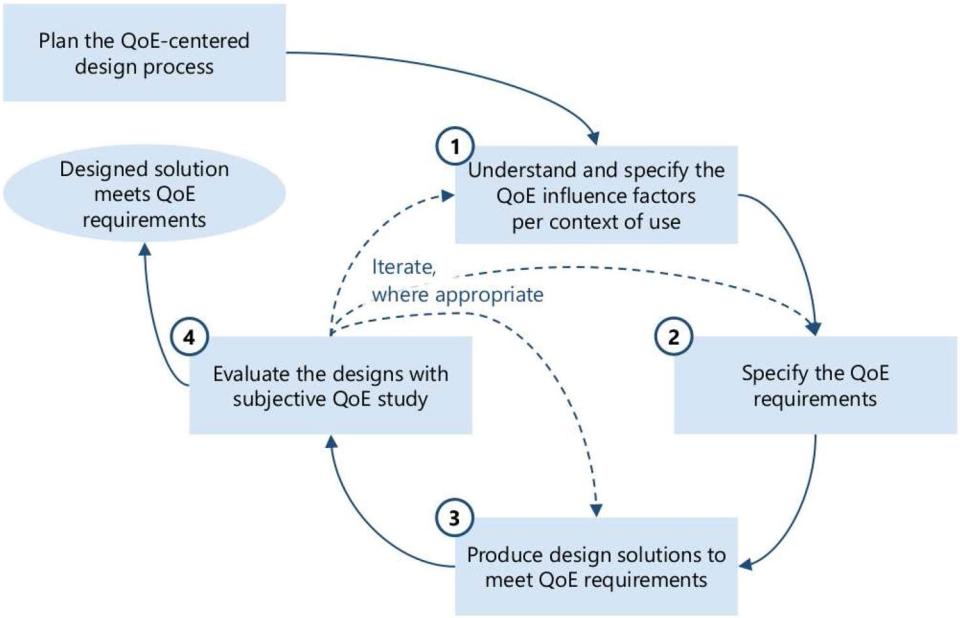

Fig. 2 Quality of Experience Centered Design Cycle

To provide general guidelines on how to consider network-specific requirements and QoE in the design process, we propose the Quality of Experience Centered Design Cycle (QoE-CDC). Note that the purpose of the QoE-CDC is not to replace the HCDC, but to complement it, as this allows to integrate both UX aspects and QoE aspects into the developed application. However, we explicitly do not call for an extension of the HCDC or a complete integration of both cycles. Instead, the advantage of standalone cycles is higher flexibility for application developers, such that both cycles can be iterated independently, e.g., in parallel with separate teams, one after the other with a single team, or one cycle might be iterated multiple times before the other cycle is triggered. For example, HCI and UX specialists could iterate the HCDC for designing the human-centered frontend of an application, while networking and QoE specialists could apply the QoE-CDC to design the network-aware backend of an application.

Quality of Experience Centered Design Cycle

The proposed Quality of Experience Centered Design Cycle (QoE-CDC) is depicted in Fig. 2. It borrows the general appearance from the HCDC defined in the ISO 9241-210:2010 standard [1]. Moreover, it has to be noted that there is a strong interaction between the two cycles, which will be discussed below. However, the focus of the proposed cycle is to consider Quality of Experience, which is underrepresented in the previous cycle, but has become a key factor for the success of networked applications [23] due to the ubiquity of the Internet.

Similar to the HCDC, the QoE-CDC has to be well planned before all design activities. This includes, among others, to define responsibilities, milestones, and a schedule for the development of a QoE-centered design. After the planning stage, the iterative steps of the QoE-CDC start. First, it is paramount to **identify, understand, and specify the QoE influence factors per context of use**. This means that it has to be investigated which application behavior is influenced by the network, and how network problems will be perceived by users. These questions need to be answered per context of use, e.g., in mobile or fixed network access, in business or entertainment use, or in a single user or an interacting/collaborative situation with multiple users. Moreover, the different QoE factors have to be ranked per context of use according to the frequency and severity of potential QoE degradation.

In the second step, the **QoE requirements have to be specified**. This requires identifying the most important QoE factors, which can and shall be controlled by the application. This includes to defined thresholds for the QoE factors per context of use, which can be translated into network requirements. These network requirements have to be specified in an appropriate format depending on the considered application and QoE factor, such that thresholds can range from qualitative, high-level definitions (e.g., connectivity to cloud server) to technical definitions via objective metrics (e.g., downlink bandwidth larger than 3 Mbps).

Next, **design solutions have to be produced** to meet the QoE requirements. For this step, the perspective has to be changed, such that the network and its performance have to be considered as independent/exogenous variables. Based on that perspective, it has to be investigated how the

previously specified network requirements can be minimized and adapted to the current network performance for all considered contexts of use. Further, it has to be defined how the application shall react in case the requirements are violated. This might include to trade-off the requirements if only some can be (partially) fulfilled at the same time. Moreover, it might require reworking the user interface or application feedback to mask or hide network problems.

Finally, the **produced designs have to be evaluated** against the QoE requirements with a subjective QoE study. The study has to be conducted in the previously identified contexts of use and under typical (emulated) network conditions. Traditionally, in the telecommunication community, standard rating scales are used, such as the Absolute Category Rating scale [25] or Technology Acceptance rating scales [26]. The benefits of those scales are their simplicity and their speed, while at the same time being well researched [27]. In the HCI community, standalone rating scales are not very popular, since crucial contextual and information inherent to the user might be missed. More often, a mixture of quantitative and qualitative research methods is used, like, for example, questionnaires, interviews, and observations [28]. We advise to use a combined approach, to both assess network-related aspects of quality on well-known standard rating scales, and to gain deeper insights from additional qualitative or qualitative assessments.

If the QoE results fulfill the QoE requirements in every context of use, the QoE-CDC can be terminated and an iteration of the HCDC can be triggered to also fulfill remaining user requirements, if needed. If the QoE results do not meet the QoE requirements, further iterations of the QoE-CDC are required.

As mentioned above, there is a strong interaction between the HCDC and the QoE-CDC, such that the latter could be initiated as a part of the HCDC, when designing networked-based backend components of an application. Moreover, there are also entry points in the other direction, such that the HCDC could be triggered during the execution of the QoE-CDC. For example, as part of the QoE-CDC, it might be required to rework the frontend user interface or application feedback to mask or hide network problems, which constitutes an entry point to the HCDC. Also, when design solutions are produced, it might be possible to oscillate between both cycles to produce solutions, which meet not only QoE requirements, but also general user requirements specified in the HCDC. Finally, in the evaluation step, it is again possible to bridge between both cycles, such that the produced design is evaluated against both QoE and user requirements.

In the end, the termination of both the human-centered and QoE-CDC will result in an application design, which reaches a high UX and avoids QoE degradation to also reach a high QoE. This constitutes a win–win situation for the user, the network provider, and the application provider.

To sum up, the four steps of the QoE-CDC are as follows:

1. **Identify, understand and specify the QoE influence factors per context of use.**
2. **Specify the QoE requirements.**
3. **Produce design solutions to meet QoE requirements.**
4. **Evaluate the designs with subjective QoE studies.**

In the following, we will discuss the QoE-CDC for four popular application types, namely smartphone applications, as well as on-demand music/video streaming, live video streaming/video conferencing, and mobile instant messaging applications.

Use Cases

In this section, we will demonstrate potential applications of the Quality of Experience Centered Design Cycle (QoE-CDC) by means of applying it to typical network-centric use cases. The interplay of the QoE-CDC and the HCDC will be highlighted along the way. For each use case, the four steps of the QoE-CDC will be evaluated. Note that the planning phase of the QoE-CDC will be omitted as it will be specific to each individual project. In the end, we will summarize the potential results from applying the QoE-CDC for the discussed use cases, i.e., the identified connections between QoE requirements and UX design implications, which can be leveraged to improve the user experience with network-based applications.

Smartphone Applications

Smartphone applications (apps) exist for many purposes and in huge variety, e.g., for news, social media, or web shops. While native apps are developed for specific platforms, i.e., operating systems, they run locally on the device, and thus, can run offline, in principle. Native apps can fully leverage the features of the device but need to be implemented specifically for each platform. Web apps, in contrast, run completely on a web server and are accessed over the Internet through the browser of the smartphone. Thus, they can be implemented independent of the platform, but they are very limited with respect to the features of the device and need a constant Internet connection. In between, there exist hybrid apps, which consist of a simple, native app that wraps around an internal browser, e.g., Android's WebView, to access a web app. These hybrid apps mix both worlds, such that they can leverage local features of the device, but most of the app can be implemented independent of the platform. Since web or hybrid apps require an Internet connection to access and interact with the app, and, in general, most of the smartphone apps access the Internet to up- or download

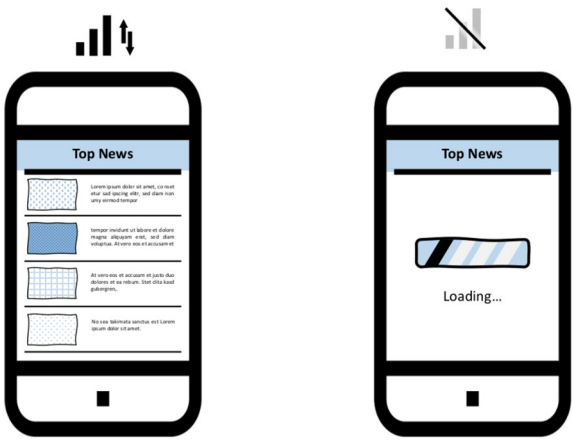

Fig. 3 Loading times in a smartphone application

content, the QoE needs to be considered when developing a smartphone app.

In **Step 1** of the Quality of Experience Centered Design Cycle, the QoE influence factors of smartphone apps have to be identified. As mentioned above, Internet connectivity is a functional requirement of most smartphone apps, thus, it is also a QoE influence factor. If the app cannot be used when there is no Internet, the users are not satisfied and their QoE will be bad. Apart from connectivity, loading times have a well-known impact on QoE of web browsing related tasks [29]. With smartphone apps, those loading times are omnipresent, especially if large amounts of data have to be downloaded from the Internet and displayed, e.g., in AR/VR apps, social network apps, or mobile gaming, see Fig. 3. However, shorter loading times might also occur in case computations are offloaded to data centers, or when mobile webpages are browsed in simple web or hybrid apps [30]. A 2015 report [31] found that mobile app users are impatient, such that 61% expected apps to start in 4 seconds or less, and 49% expected apps to respond in 2 seconds or less. Moreover, 80% of the users indicated that they will only retry an app up to three times, if they experience problems.

Thus, the QoE requirements of smartphone apps (**Step 2**) are permanent Internet connectivity and high bandwidth to minimize loading times. Note that the bandwidth requirement heavily depends on the purpose and functionality that a particular smartphone app offers to its users. Thus, we will not specify these requirements here in full detail. Moreover, due to cost aspects, network-centric use cases often face limited Internet connectivity, e.g., in terms of data caps, bandwidth throttling, or network coverage in mobile networks. These limitations suggest the QoE requirement that apps should optimize their network usage and avoid excessive data transmissions.

Next, a solution has to be designed, which meets these QoE requirements (**Step 3**). For this, we will mainly focus on the connectivity requirement, which is common to all smartphone apps. In this process, we have to change the perspective and consider that Internet connectivity is not always available, which can lead to delays when loading content. As the above-described studies reported, such loading times significantly reduce the QoE. To avoid users staring at a blank screen, it is advisable to use loading screens [32]. Using these, users often face a load screen, e.g., a blank screen with a spinning icon or progress bar, which indicates that users have to wait for a specific amount of time. As an alternative to loading screens, skeleton screens become increasingly popular. While loading the content, here, the outline (skeleton) of the content to come is displayed using a simplified presentation, for example, gray boxes and lines. Another possible solution to this problem would be to mask or hide waiting times from the user. If the app notices that no Internet connection is available, it could communicate this to the user and minimize itself to the background. In the background, the app would try to access the Internet and send a push notification to the user as soon as connectivity is available, and the app can be used. This way, users would not be blocked waiting in a load screen, but they could put their attention to something else in the meantime. As soon as they are notified that Internet connectivity is available, they could return and bring the app to the foreground again to continue using it. Note that the same solution could also mask or hide slow Internet connections, where users would have to wait for some content to be downloaded. As a more advanced solution, apps could monitor the mobility of the user, and notify the user, when network coverage or Internet connectivity was lost due to mobility. Then, the user could decide to go back to a place with network connectivity for some time to manually or automatically download some content for the offline phase. Note that some apps already offer the option to manually download content as a preparation for offline phases, e.g., episodes of series or even entire movies in streaming apps. A last potential design solution is caching. It allows to keep popular content in the storage of the smartphone of the user. If the user wants to access this content again, the app does not need an Internet connection because the content is already available on the local device. This technology is already used by so-called progressive web applications [33]. One step further, the app could even pre-fetch content, which is potentially interesting to the user in the future. Pre-fetching means that content is speculatively loaded during times, in which the app has access to the Internet, such that the content would be locally available if the Internet connection breaks. Such pre-fetching could be based on content popularity or typical user interaction patterns. Note that pre-fetching irrelevant content will reduce the available bandwidth and the available storage and increase the risk for exceeding data caps, which has to be considered when implementing this solution.

After design solutions have been proposed, they have to be evaluated with subjective QoE studies in **Step 4**. To increase the user experience when using progress bars, the authors of [34] evaluated the impact of various progress bar behaviors on user perception of process duration. They found that it is possible to modify the progress bar in a way that they appear faster by, for example, using non-linear but accelerating progress. Further improvements in user satisfaction can also be achieved by changing the design of the loading bar. Progress bars with storytelling animations as well as interactive games can increase user perception by reducing users' time perception [35]. In [36], the authors compared the usefulness of skeleton screens as an alternative to progress bars. They found that skeleton screens are as effective in reducing the perceived loading time for the user as progress bars. To the best of our knowledge, no subjective QoE studies were conducted in the direction of masking or hiding waiting times from the user. Thus, we leave this open to future work for now. We are also not aware of any work in the area of caching within applications. A similar approach, which could be applied and tested in smartphone applications, was investigated in [37]. Here, a system was developed which could pre-fetch and cache individually relevant content for each user based on social information, i.e., information from his online social network profile. To evaluate network performance and the resulting QoE of mobile apps, QoE Doctor [38] was implemented. Using this tool, active measurements on network as well as application layer can be conducted to evaluate applications.

Finally, after studying the QoE-aware design cycle for smartphone apps, we will present some examples of popular apps, which currently use one or more of the above mentioned solutions [1]. For example, the popular social news application Reddit uses loading screens to avoid users facing a blank screen while loading content. Instead of showing a simple loading bar, the app uses an animated icon of an alien to entertain users while waiting. The file hosting service Google Drive takes a different approach to shorten the perceived waiting time for the users using skeleton screens with a moving shadow animation. A combination of both approaches can be seen for the social networking app Instagram, which not only shows a loading icon at the top of the screen while users wait for their content to be shown, but also presents a skeleton screen which outlines the content

[1] The listed applications and their QoE-aware design approaches were accessed and described by the authors on July 15, 2021. They may be subject to change in the future. Note that the selection of specific applications must not be considered an advertisement for these apps. Our only motivation was to provide positive examples of the implementation of the mentioned design solutions. We did not receive any monetary or other incentives for selecting specific applications.

Fig. 4 Stalling during video streaming, i.e., an interruption of the video playback due to empty buffer

to be expected. As a last example, another approach of hiding waiting times can be seen when using Google's search engine with the Chrome browser app. When the Internet connection of a user breaks down, the app saves the request and asks whether the user wants to be notified as soon as the search results are available.

On-Demand Music/Video Streaming

Next, we look at on-demand streaming applications, especially music and video streaming. Streaming applications are very popular nowadays and account for 62.1% mobile traffic share worldwide [22]. Since streaming applications require a network connection to receive the media data, which shall be played out, QoE has to be considered here.

If streaming applications shall be improved in the QoE-centered design process, the QoE influence factors have to be investigated (**Step 1**). For video QoE, most works on video streaming agree that initial delay, stalling, and quality adaptation are the most dominant QoE factors [7]. Stalling, i.e., playback interruptions due to buffer depletion, is considered the worst QoE degradation [39, 40], and should be avoided, see Fig. 4. Furthermore, video streams should be played out with high visual quality [41]. In contrast, initial delay has only a small impact on the QoE [29]. For music streaming, similar trends are visible. Here again, stalling is considered as the biggest influence factor of QoE while initial delay plays only a minor role [42, 43]. Having a look beyond the streaming itself, the user satisfaction can also be degraded for increased navigation time (time between starting the app and the actual start of the audio playback) [44].

Considering the different use cases of streaming applications, these technical QoE influence factors stay the same, regardless if the users are on a PC and use a wired Internet connection, or if they use a mobile app on a mobile network, although expectations might differ. This means that, for example, users driving on a highway with a mobile Internet

access might be more tolerant to degradation than users at home with a fixed broadband Internet access. Nevertheless, there are other non-technical effects on the QoE, which have to be considered in some use cases, for example context factors like used device, content, or usage [7]. For example, mobile users might perceive a bad QoE with a streaming app if the permanent network usage of the app exceeds their mobile data plan. However, in the following, we will focus on the technical QoE factors only.

After understanding the technical influence factors, the QoE requirements have to be derived from the above findings (**Step 2**). The most important aspect to avoid stalling is that media data has to be downloaded faster than it is played out. This means that the download bandwidth has to be higher than the music/video bitrate. To reach a high visual or audio quality, strong compression of the media should be avoided. It has to be noted that there is a trade-off between the visual/audio quality and the resulting bitrate, such that less compression leads to better visual/audio quality but also to higher media bitrate. Thus, there is a QoE requirement for high bandwidths to support the streaming of high bitrates. Finally, as initial delay also has some impact on the QoE, there is a requirement that the start of the playback should not be delayed too much.

In **Step 3**, a solution has to be found to meet the QoE requirements. Therefore, it is necessary to change the perspective and consider that perfect network conditions are not always given. Thus, it cannot be taken for granted that there is always a high bandwidth, such that high visual/audio quality with a high bitrate can be streamed to the users. Instead, the bandwidth fluctuates over time or there might even be an outage, which is out of control of the app.

To overcome that short network outages or short-term bandwidth fluctuations cause stalling, a playout buffer can be used to store a few seconds of playtime ahead of the current position. For this, the playback start can be delayed until the buffer has filled up, which results in a trade-off between initial delay and stalling. However, a slight increase of the initial delay only has a small impact on the QoE and is preferred compared to the huge impact of a possible stalling by most streaming services. This shows that it might not always be possible to fulfill all QoE requirements at the same time. Instead, there might be the need to trade-off some requirements.

One solution to overcome long-term bandwidth reductions is to dynamically adjust the music/video bitrate using several representations of the media data with different bitrates. In case the bandwidth drops, a representation with lower bitrate can be streamed, such that stalling is avoided, which is the worst QoE degradation. However, there is again a trade-off as the visual/audio quality of the streamed media will be reduced if media with lower bitrate and higher compression is downloaded. This idea of adaptive streaming is already widely used by many streaming services and the corresponding HTTP Adaptive Streaming (HAS) or Adaptive Bitrate Streaming (ABR) technology has also been standardized by MPEG Dynamic Streaming over HTTP (MPEG-DASH) [45]. It utilizes an adaptation logic, i.e., an algorithm on the client side, which controls the trade-offs between measured bandwidth, buffer fill, and downloaded bitrate.

The buffer approach can be further extended, such that the app further increases the buffer in situations where high bandwidth is available to download more high bitrate media. In case of streaming playlists where the next song/video is known in advance, which is especially common for listening to music albums or binge watching of series, it even extends to future media. This means, after the current song/video has been completely downloaded and while the remaining buffered playtime is played out, (parts of) the next track/episode of the playlist can already be downloaded to leverage the available bandwidth and provide for a future bandwidth reduction or outage. However, to reduce the server load and avoid unnecessary transmission of media data in case the user aborts the playback, this approach is rarely used by current video streaming services, which instead prefer to limit the playout buffer. In contrast, it is common for music streaming platforms to already load the next songs of a playlist.

In the end, the designed app has to be evaluated by a subjective user study (**Step 4**). Since (adaptive) streaming is well investigated, it is already known that streaming benefits from employing a playout buffer and adaptive selection of an appropriate representation [7, 42]. Nevertheless, the implementation of an adaptation logic for different network conditions and use cases is still subject to ongoing research. To further improve streaming, in the future, additional (non-technical) QoE requirements could be added, such that the QoE-CDC needs to be repeated until the designed solution meets the QoE requirements.

Note that the presented improvements of streaming applications were developed over many decades and are already implemented in most applications. However, they could have also resulted from a thoroughly executed QoE-CDC in shorter time, if awareness had been given earlier to QoE-centered design. Thus, it is especially important for new and upcoming use cases to consider QoE from the start to faster obtain designs for both high UX and high QoE.

To sum up, for on-demand music and video streaming, different solutions are available to overcome network outages or bandwidth fluctuations. The following examples show how popular on-demand streaming applications implement these approaches [2]. For example, the music streaming

[2] The listed applications and their QoE-aware design approaches were accessed and described by the authors on July 15, 2021. They may be subject to change in the future. Note that the selection of

application Spotify adapts the audio quality to the available bandwidth and pre-buffers subsequent songs of a playlist to avoid playback interruptions from network outages or bandwidth fluctuations. The same approaches can also be seen with the popular video streaming platform YouTube, which also relies on adaptive streaming to adjust the video and audio bitrate to the network conditions, and additionally implements buffers to store a limited amount of playtime ahead of the current position.

Live Video Streaming/Video Conferencing

Third, we will investigate live video streaming. Here, the video content is streamed in or near real time, either only unidirectional to the client, e.g., in case of live transmission of sports events, or bidirectional to and from the client in a so-called video conference, e.g., for a telepresence business meeting or for a doctor-to-patient communication in telemedicine. Since unidirectional live streaming is a subproblem of the bidirectional case, in the following, special emphasis is put to the latter case of video conferencing. As the client, which simultaneously transmit and receive audio and video data, needs an Internet connection, QoE needs to be considered here.

Next in the QoE-centered design process, the QoE influence factors of live video streaming have to be identified, understood, and specified (**Step 1**). Since live video streaming is a form of video streaming, the same QoE factors, which were highlighted above for on-demand video streaming, are relevant. This is, initial delay, stalling, and quality adaptation [46]. To reach a high QoE with live video streaming [47], the bitrate should be maximized to reach a high video and audio quality, and quality adaptation should be minimized. Moreover, all video content should be played out with a high frame rate, low stalling, and a low delay towards the live event. Furthermore, for video conferencing, the synchronization between audio and video plays an important role [48], see Fig. 5. The major difference to classical on-demand video streaming is that live video streaming cannot utilize a large playout buffer as this would lead to a large live delay. Often, UDP-based streaming is used to further reduce the live delay, but it can lead to packet loss, and thus, to artifacts in the transmitted video, which reduce the QoE [49, 50]. Some special use cases might even have more strict requirements, e.g., in the telemedicine use case,

Fig. 5 Poor visual quality and audio/video synchronization problems during video conferencing

it might be important to transmit a low compression, high resolution video from the patient to the doctor to be able to make medical diagnoses [51, 52]. However, in this case, the oppositely directed video from the doctor to the patient might not require such high visual quality, which shows that QoE requirements might also not be equal for all clients.

In **Step 2**, the requirements for high QoE need to be specified. As discussed above, to satisfy the purely video streaming related QoE aspects, it is sufficient that the available bandwidth must be higher than the video bitrate. Moreover, latency and packet loss have be low, such that live delay is minimized, and, in case of unreliable UDP-based transmission, video artifacts can be avoided. Finally, the played-out video and audio should be perfectly synchronized to allow for an as natural conversation over the Internet as possible.

As the next step (**Step 3**), design solutions need to be presented, which meet the QoE requirements. Here again, quality adaption is a good method to align the bitrate of the video to the network conditions, and thus, avoid stalling. However, in contrast to on-demand streaming above, times with bad or no Internet connection cannot be compensated with a large buffer in case of live streaming or video conferencing. This would increase the live delay, which has to be minimized. Thus, solutions should limit themselves to a small buffer and try to compensate bad network conditions and optimize the users' QoE by other means.

The first option is to adapt the playout speed of the video to avoiding stalling and/or skipping of video content while keeping live delay low. In case of a stalling event, a typical strategy in live streaming is to skip frames or segments [47, 53] to catch up with the live event, which causes users to miss some video content. In contrast, after a stalling event when the buffer has filled, the video could be resumed and played out with a higher speed to catch up with the live event, which would avoid skipping and missing video content. The same technique can also be applied during live streaming to keep live delay low while avoiding stalling.

Footnote 2 (continued)

specific applications must not be considered an advertisement for these apps. Our only motivation was to provide positive examples of the implementation of the mentioned design solutions. We did not receive any monetary or other incentives for selecting specific applications.

In case the playout buffer empties, the playout speed is reduced to avoid stalling, which, however, increases the live delay. In contrast, if the playout buffer fills, the playout speed is increased to reduce the live delay. This strategy of modifying the playout speed is called Adaptive Media Playout (AMP) [54, 55], and is widely used in live streaming services.

Next, users could be informed whenever a bad Internet connection is detected. In case the network still allows a smooth streaming but with very low visual quality or frequent quality fluctuations, which could annoy the users, the users could be asked whether they want to continue the live streaming or video conference anyway, e.g., in case it is an important virtual meeting that cannot be moved, or if the streaming or call should be suspended until the connection improves. In the latter case, the app could periodically trigger measurements in the background to find when the network has improved and notify the users accordingly that the streaming or call can be resumed. If the network connection is even worse, such that playout glitches or interruptions of the streaming are impending, users could be given an early warning that the streaming might be terminated at any time soon. By this means, participants of a video conference could prepare and be able to quickly find and agree on an alternative communication procedure.

Another option is that the app fails gracefully, i.e., if a video conference is not possible under the current bandwidth conditions, the app could fall back to audio conferencing only, which has reduced bitrate requirements. If also audio conferencing cannot be sustained under the given network conditions, the app could eventually offer a text-based messaging communication as a last fallback to keep the conversation ongoing.

Finally, in the use case of doctor-to-patient communication in telemedicine, where it is mandatory to have a high-quality video from the patient to the doctor, other options might need to be considered. For example, in case the network condition does not support a high bitrate video stream, the client of the player could offer to locally record important video content in high visual quality. The high-quality video file can be transmitted reliably while the video conference is ongoing or suspended. Then, the doctor could be notified by his client when the transmission was completed, and the video is ready to watch. This way, important video content does not need to be compressed to low visual quality, but there is a trade-off when the important video becomes available for the doctor. Note that a background transmission during an ongoing call might further strain the bandwidth and negatively affect the quality of the ongoing call. So, it might actually be better to suspend the call, such that the doctor can turn to other patients until the transmission is completed, and resume the video conference with this patient afterwards.

After solutions have been designed, they have to be evaluated in designated QoE studies as **Step 4** of the QoE-CDC. The QoE impact of AMP has been investigated in [56]. It was found that the QoE remained high for playout rate changes in the range from 80 to 180% of the regular playout speed. However, increasing or decreasing the playout speed further caused a huge drop in the QoE. As we are not aware of any solution, which implements the other proposed QoE-aware designs, we leave this open to future work for now.

Well-known live video streaming and video conferencing applications typically implement such specific QoE-aware design solutions, often in addition to other solutions presented above, as can be seen for the following examples [3]. To avoid stalling events, for example, the live streaming service Twitch uses adaptive media playout, and thus, adjusts the playback speed according to the available playback time in the buffer to reduce the live delay. When it comes to video calls, WhatsApp and Facetime warn their users during a call when the network connection deteriorates, and automatically stop the video transmission to maintain an uninterrupted voice transmission.

Mobile Instant Messaging

For mobile instant messaging (MIM) applications like WhatsApp, Facebook Messenger, or WeChat, different requirements apply than for streaming. The main difference is that they are primarily used in mobile networks and the workload is very irregularly distributed, depending on the frequency of sending and receiving messages. Furthermore, the type of a message has a high influence on the network requirements, as media messages, like videos, images, or voice messages, are significantly larger than simple text messages.

Focusing the QoE of MIM applications (**Step 1**), up- and download time of messages and files are considered the most relevant feature [57], see Fig. 6. To reduce waiting times before audio or video playback, MIM applications often use streaming. This means that playback of the file can start even before it has been completely downloaded. Here, the same QoE requirements, which were mentioned above for on-demand video streaming, are relevant.

This means for the QoE requirements (**Step 2**) that, to enable real-time communication, a permanent Internet connection is needed and the transmission rate, i.e., both

[3] The listed applications and their QoE-aware design approaches were accessed and described by the authors on July 15, 2021. They may be subject to change in the future. Note that the selection of specific applications must not be considered an advertisement for these apps. Our only motivation was to provide positive examples of the implementation of the mentioned design solutions. We did not receive any monetary or other incentives for selecting specific applications.

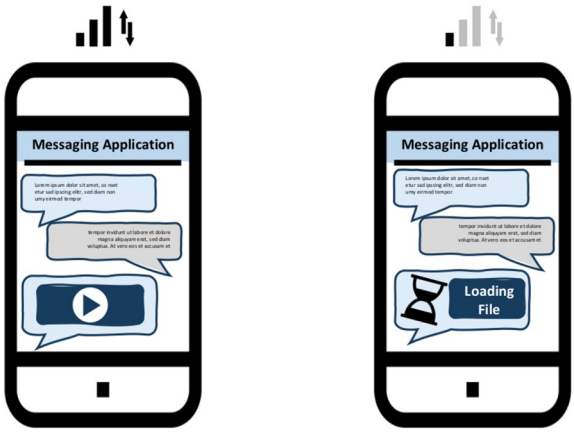

Fig. 6 Up- and download times of messages and files in mobile instant messaging applications

sending to the application server and receiving from it, must be as fast as possible. Therefore, the applications need low delay and large upload and download bandwidth to quickly transmit messages.

In **Step 3**, when considering the network as an exogenous variable, several solutions can be used in case of bad or no Internet connection. For example, in case of no Internet connection, automatic retransmissions could be implemented, such that a user does not have to manually send the message again. When sending media files, which have a potentially large transmission time, small thumbnails could be created and sent beforehand, such that users can already see a preview as long as the actual media file is being downloaded. To reduce the file size, and thus, the transmission time, media compression can additionally be used. Furthermore, instead of uploading and downloading entire voice or video files, messaging applications could leverage streaming. This way, the media can already be played out as soon as sufficient data has been downloaded, cf. initial delay of streaming, such that users experience a shorter transmission delay. However, the QoE requirements of streaming additionally apply as described above, which have to be taken into account by the designed solution. For images, MIM applications could use interlacing, which is an encoding that transmits the most important information of the image first. Upon reception of a few parts of the image, a blurry preview image can be interpolated and displayed, and the transmission of additional data allows to add more and more details until the image is visually complete. This way, users do not have to wait the full time until the image is downloaded completely before being able to watch it, but they quickly see a preview of the image content, which gradually improves over time, and thus, reduces the perceived waiting time. Finally, to avoid that the client periodically has to poll the app server whether there are new messages for him, which is very cost intense with respect to bandwidth and battery, messaging applications could keep a TCP sockets waiting in accept mode. This does not use much power or data, but allows the app to download incoming messages in the background and notify the user quickly when a new message has arrived. Such solutions are readily available to app developers, e.g., Google's Firebase Cloud Messaging (FCM).

Again, the last step of the QoE-CDC (**Step 4**) is to evaluate the designed solutions in QoE studies. To the best of our knowledge, currently no QoE studies exist, which investigate mechanisms for QoE improvement of messaging applications. Thus, we leave such research for future work.

When looking at MIM applications, examples for QoE-aware design solutions can be found in several popular applications [4]. For example, when the user's Internet connection is not good enough to download an image or a video, Facebook Messenger informs its users that the file is being downloaded, and eventually, starts showing a progress bar for the download. Similar behavior can bee seen for the MIM app Signal, which shows a spinner and a blurred version of the image as a preview, indicating that an encoding format is used, which supports interlacing. Afterwards, the spinner is turned into a loading circle, showing the progress of the download. In addition, both apps use image compression to reduce the file size, and thus, the transmission time.

In this section we demonstrated potential applications of the QoE-CDC and highlighted the interplay of the QoE-CDC and the HCDC as well as possible UX design solutions. A summary of the presented UX design implications of the four use cases can be found in Table 1, which can be used to improve the user experience with network-based applications.

Validation of the QoE-CDC: App for Crowdsourced Video Streaming QoE Studies

Many of the above-presented developments and improvements in app design have evolved over time. This means, although the same improvements could have been achieved with the QoE-CDC, these examples do not suffice to validate the QoE-CDC. Thus, we finally showcase a completely different type of app, namely an app for conducting video

[4] The listed applications and their QoE-aware design approaches were accessed and described by the authors on July 15, 2021. They may be subject to change in the future. Note that the selection of specific applications must not be considered an advertisement for these apps. Our only motivation was to provide positive examples of the implementation of the mentioned design solutions. We did not receive any monetary or other incentives for selecting specific applications.

Table 1 Summary of exemplary use cases and their UX design implications

Use case	QoE requirements	Exemplary UX design implications
Smartphone applications	Permanent Internet connection	Loading and skeleton screens
	High bandwidth	Background download and notification
		Caching/pre-fetching
On-demand music and video streaming	Short initial delay	Bitrate adaptation
	No stalling events	Playout buffer
	High audio/visual quality	Caching/pre-fetching song/video
Live video streaming and video conferencing	Similar to video streaming	Adaptive media playout (AMP)
	Minimize live delay	Informing users
	Audio/video synchronization	Graceful failing
		Side channel file download
Mobile instant messaging	Permanent Internet connection	Automatic retransmission
	Low delay	Content preview (thumbnail)
	High bandwidth	Media compression
		Streaming/interlacing
		Background download and notification

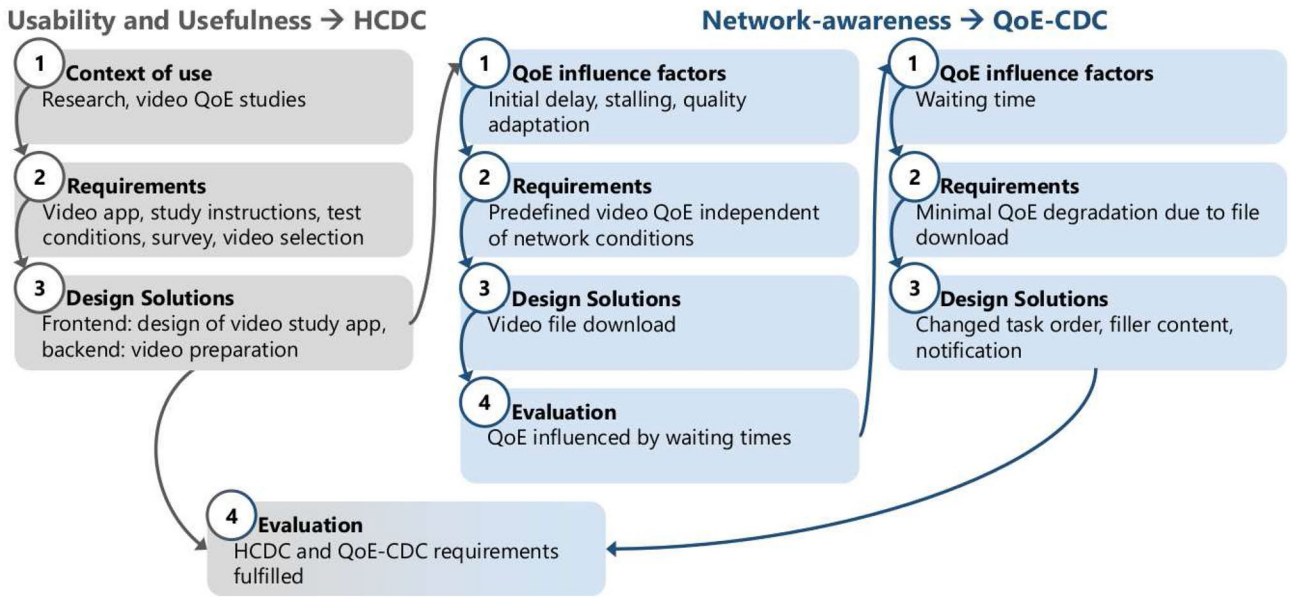

Fig. 7 Overview of HCDC and QoE-CDC iterations for design and development of the CroQoE application

streaming QoE studies via crowdsourcing on the smartphones of study participants [58, 59], which we designed from scratch using the HCDC in combination with the QoE-CDC. This type of app is especially critical with respect to QoE, as QoE has to be controlled for to not bias the QoE study.

Figure 7 gives an overview of the design and development of the app for crowdsourced QoE studies of HTTP Adaptive Streaming (HAS), named CroQoE [58, 59]. After the first steps of the HCDC, we improved the initial app design with respect to the resulting QoE by two iterations of the QoE-CDC. In the following, we will outline the combined application of both the HCDC and the QoE-CDC, and their iterations and design improvements in full detail.

Application of the HCDC

First, we kicked off the CroQoE design by considering the HCDC. This means, we identified and specified the context of use (HCDC, Step 1). In our case, the stakeholders were researchers, who want to conduct reliable and valid videos QoE studies. The users of the app were the study

participants, who should be using the app on their smartphones in an uncontrolled environment, which is typical for crowdsourcing studies.

The derived requirements (HCDC, Step 2) specified that CroQoE should look and feel like a typical video streaming app. In addition, the app should allow to display study instructions and it should allow to control the tested QoE conditions, which are presented to the app users, e.g., a certain number of stalling events or a certain video bitrate/codec/resolution adaptation pattern. Moreover, the app should include surveys, such that participants could submit demographic information as well as their QoE feedback about the experienced video streaming. As crowdsourced QoE studies are conducted in an unsupervised fashion, the app should monitor the test execution, and allow to ask consistency questions to the participants [60]. Finally, the app should allow users to select the video content, which they would like to watch during the study. Note that this a novel feature, which makes CroQoE unique. Previously, all content in crowdsourced video QoE study had always been pre-selected by the researchers. However, a realistic video streaming experience includes that users can select the video content themselves. Thus, the following user path resulted: first, the participants should be welcomed, and the study instructions should be displayed. Then, participants should select the video content, watch the video including the tested QoE conditions, and finally, rate the QoE.

In Step 3 of the HCDC, design solutions had to be produced that meet the user requirements. For this, the frontend of CroQoE was implemented as Android application, and we adapted the design of a popular video streaming app to make CroQoE look realistic. To make the app usable for QoE studies, we included new screens for study instructions, for surveys about the demographic information and about the experience of the participants, and for specifying the desired video content. Additionally, we implemented a backend server, which is responsible for preparing the QoE test conditions and for data storage.

At app start, the app connects via REST API to the backend server to determine the current app state and to register the study participant. The QoE study is presented in the application with which users can interact as specified. This means, participants read the study instructions, enter their demographic information, and can then submit keywords for video content of their interests. As soon as the user hits the button to start the study, the backend starts to process the sent request, i.e., the preparation of the test videos.

For this, the backend crawls a major video streaming provider for matching videos. Based on the submitted user's keywords, the API of a big video database is used to find matching videos. A video matches when it fits specific guidelines, i.e., the top five short HD videos, which are sorted by view count. By selecting random video IDs from the crawled video IDs, constant repetition of the same video is avoided.

As soon as the videos are available, the QoE test conditions are dynamically inserted into the videos. For this purpose, FFmpeg is used. First, the video is cut to the desired length. Then, initial delay or stalling are added to the video, or the visual quality is modified to replicate a desired adaptation pattern. For example, for stalling, the video is cut into multiple parts. Between these consecutive parts, stalling is emulated by creating video sequences of desired stalling length which show a still image and an overlaid buffering GIF. In the end, the available parts are again concatenated to the final video.

With respect to transmitting the final video from the backend to the CroQoE app, it immediately became evident that live streaming of the video content was not an option as the uncontrolled environment, especially, the uncontrolled network conditions, in which users would participate in the QoE study, could introduce arbitrary QoE degradation. This would result in the presentation of uncontrolled QoE conditions to participants, which is not acceptable in a QoE study. Thus, we applied the QoE-CDC at this stage of the design of CroQoE to consider and better control the delivered QoE.

First Iteration of the QoE-CDC

In **Step 1** of the QoE-CDC the QoE influence factors have to be identified. Our app CroQoE features adaptive video streaming, for which, as discussed above, initial delay, stalling, and quality adaptation are the most dominant QoE factors [7].

Step 2 demands to specify the QoE requirements. Considering an app for crowdsourced video QoE studies, the QoE requirements follow directly from the general requirements, which were already specified in the HDCD. This means, CroQoE should deliver exactly the predefined video QoE as specified in the QoE test conditions. This should be achieved irrespective of the network conditions of the user, as they could not be controlled in a crowdsourcing setup.

To meet the QoE requirements (**Step 3**) and control the delivered QoE, we could not stream the videos from the backend server to the app. Thus, we implemented a file download to transmit the final video files to the frontend app. Only after the videos are completely downloaded to the app, CroQoE allows users to proceed to watching the locally played out videos. This way, additional QoE degradation introduced by fluctuating network conditions of the users can be avoided. Note that this method of pre-download and local playout is typical for crowdsourced video studies, e.g., [61]. Full-screen mode and landscape orientation are used for the video playout. Also, users are not able to control the media during playback. When a video ends, CroQoE displays the experience survey, in which users have to submit ratings

on visual quality, streaming quality, quality acceptance, and content liking.

Finally, the proposed design had to be evaluated. For this, we combined both **Step 4** of the QoE-CDC as well as the last step of the HCDC. It was obvious that the application interface closely resembled a video streaming app, it included all screens needed to conduct a video QoE study, and the presented solution could exactly replicate the technical QoE factors within the videos by design. However, we realized that additional waiting times were introduced by the preprocessing of the video content on the backend server and by the download of the final video files from the backend server to the CroQoE app. They again constituted an uncontrolled factor, which could negatively influence the QoE. Thus, we applied a second iteration of the QoE-CDC to control for the delivered QoE, this time especially considering the newly introduced file downloads.

Second Iteration of the QoE-CDC

The most important QoE factor of file downloads (**Step 1**) are waiting times [29, 62], which have a logarithmic relationship to the resulting QoE degradation. This is especially detrimental for QoE studies. As shown in [63], long waiting times during a study can result in an annoyance of the participants which can directly influence the participant's QoE.

The resulting QoE requirements again follow from the general requirements of the HCDC due to the specific purpose of the CroQoE app (**Step 2**). This means that the file downloads, which were introduced in the first iteration of the QoE-CDC, should have a minimal impact on the resulting QoE. Thereby, the overall QoE of the participants should only be affected by the tested QoE conditions within the streamed videos.

In **Step 3**, we came up with the idea that participants should be entertained with filler content while waiting for the file downloads to finish to avoid negative bias from perceived waiting times. Therefore, we changed the app's user path, such that the participants should select the video content first before entering demographic information. Thus, the backend can already start to process the video requests, while the CroQoE frontend starts to guide the participants through a survey. As soon as the videos are prepared by the backend server, the files are downloaded to the app in the background, which is not visible to the user.

As the waiting time until the backend is finished strongly depends on the complexity of the video preparation task and the available resources, we added more filler content to increase the time budget for backend processing and file downloads. This means, we added another survey on video consumption behavior, four tests for color blindness, as well as a test for macular degeneration into CroQoE. Under typical network conditions, the time to answer all survey questions and vision tests is sufficient to prepare and download the videos. Thus, after the last vision test, participants can directly start to watch the videos without perceiving any waiting time.

Technically, we implemented Google's Firebase Cloud Messaging (FCM) to notify the user's device that the backend server has finished its task. This push notification is processed by the CroQoE frontend in the background so that the participant, who is busy with the survey and the vision tests, is not aware of it. Devices are identified with a Firebase token that registers the client app instance. This token is also responsible for all authentication between app and backend. Hence, content can only be downloaded from the server when user's app has the correct token.

Furthermore, the whole backend has been designed in a way to make CroQoE usable by many participants simultaneously and still provide a reasonable server processing time. Higher processing times result in longer tasks, which should be avoided at any cost as stated above. To overcome this problem, the backend server has been containerized with Docker. On the server multiple backend instances are started which are connected to the Internet. By adding the mapped Android secure device ID [64] to the URL of the HTTP requests in the app, the reverse proxy is able to redirect the requests to the correct backend instances. Thus, each smartphone communicates with a dedicated backend server and multiple jobs can be executed in parallel. Finally, each backend instance connects to a central database where all collected data is stored.

If the backend processing and file downloads take longer than users need for the survey, we allow users to leave the app. Thereby, users are not kept waiting in CroQoE, but can spend their time otherwise, e.g., using other applications. As soon as the videos are ready watch, the push notification will inform the users. In this case, users can open a new session with CroQoE, and immediately start watching the videos without perceiving any waiting time. At this stage, we branched back into the HCDC to develop designs for the newly introduced app elements, i.e., the additional survey and vision tests, as well as the app leave and session restart screens.

Combined Evaluation of the HCDC and the QoE-CDC

Finally, we needed to evaluate the resulting app design with a subjective QoE study (**Step 4** of the QoE-CDC), in which we compared video QoE results obtained with CroQoE to a previous QoE study on desktop PCs [61]. For the new CroQoE study [59], the app was running on four Android smartphones (a Google Pixel XL, a Google Pixel 2 XL, and two Google Pixel 3A) in parallel. These devices were handed to the participants at the beginning of a study. The backend server was hosted close to the location of the

study to provide high bandwidth and high CPU for video download and video preparation. We divided the participants into two groups. The first group could not select the video content, but it was given pre-selected test content similar to the previous study for a better comparability of the two studies. Only the second group of participants could select the video content dynamically, according to the design of CroQoE. Moreover, to fully evaluate the app design with respect to the requirements (Step 4 of HCDC), another survey was added at the end of the study, which queried the satisfaction of the participants with the app design and the time investment, as well as their overall experience with the study.

The videos were modified with one of four stalling patterns. In alignment to the previous study [61], each stalling event had a length of four seconds and the videos showed either zero, one, two, or four stalling events. The stalling events occurred in a periodic pattern, i.e., the i-th stalling event was played out after $i \cdot \frac{L}{n+1}$ seconds, where L is the video length (here 30 seconds) and n is the total number of stalling events. Further, the stalling patterns were drawn randomly without replacement within each session so that a participant did not experience the same stalling pattern twice. When neglecting stallings, all videos have an exact playout length of 30 seconds. The playback of the video starts at 20% of the actual playback to avoid any introducing scenes, e.g., the studio names in a movie trailer. All videos, i.e., the dynamically selected videos during the study and the pre-selected videos beforehand, are also downloaded in the best available quality to avoid any visual bias. For the pre-selected content, similar test content as in [61] is used, i.e., a music video, a sports video, and a movie trailer. For the dynamically selected video content, the matching videos are ordered by view count and one of the five most often watched videos is returned, as described above.

The study took place over three days in the beginning of January 2020, in which 150 people (78 male, 70 female, and 2 undisclosed gender) utilized the app on the campus of the University of Würzburg, Germany, resulting in 450 watched videos. During the study, 315 videos were watched on a Google Pixel 3a, 84 videos were watched on the Google Pixel 2 XL, and the remaining 51 videos were watched on the Google Pixel XL. The participants were mainly students and University employees with a mean age of 22.5 years. The dataset was strictly filtered for outlier sessions, similar to the previous study. This means, participants who did not pass the vision tests or submitted contradictory ratings were excluded from the dataset. After the filtering, 74 participants and 222 videos remained.

To evaluate the validity of the app in terms of the resulting QoE, the obtained ratings of this study are compared to the original results of the reference QoE study [61]. In

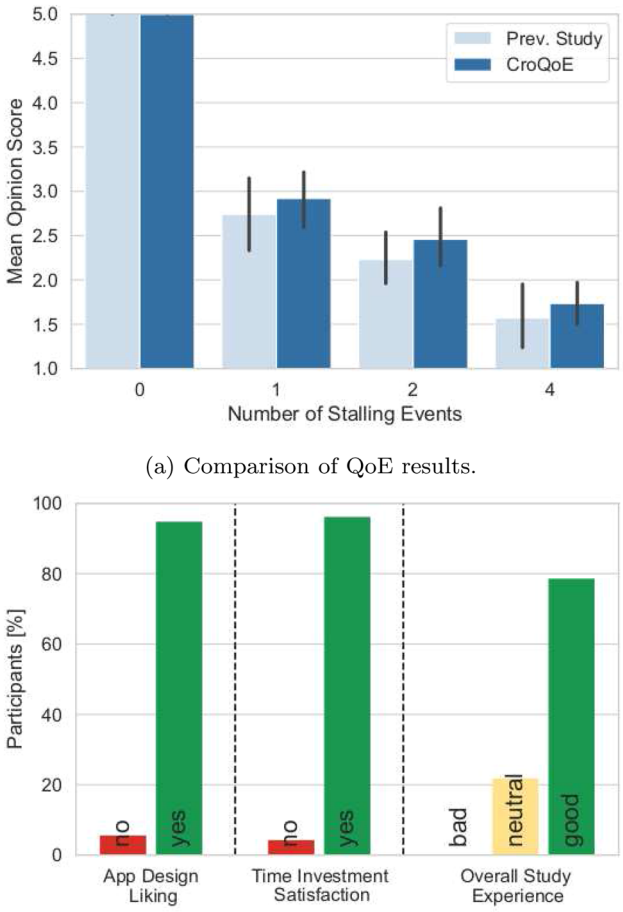

Fig. 8 Evaluation of CroQoE app in terms of QoE and design

both studies, the ratings were given on a Absolute Category Rating (ACR) [65] scale with five categories, i.e., bad (1), poor (2), fair (3), good (4), excellent (5). First, we analyze the Mean Opinion Score (MOS), which is the de facto standard QoE metric. It is computed as the average of all numerical rating scores. Figure 8a shows the Mean Opinion Scores (MOS) and 95% confidence intervals for each stalling condition. The x-axis depicts the number of stalling events, whereas the y-axis depicts the MOS. For each stalling condition, the light blue bars on the left correspond to the results of the previous QoE study, and the dark blue bars on the right correspond to the results of CroQoE users with pre-selected content. For zero stalling events, all participants rated excellent QoE. Note that this was an instruction in both studies, and all participants who failed on this instruction were strictly filtered out. For the remaining test conditions, CroQoE reaches slightly higher MOS values than the previous study. However, these differences are minor and the 95% confidence intervals, which are depicted as black whiskers on top of each bar, overlap.

As the rating data are ordinal, we perform a Mann-Whitney U test to check whether there are any differences between the datasets. The test returns a p-value of 0.18, thus, the hypothesis that the two datasets are obtained from the same distribution cannot be rejected. These results indicate that there is no difference in the resulting QoE ratings with CroQoE compared to the previous QoE study. This also confirms that the application of the QoE-CDC was successful, such that the developed design and the newly introduced waiting times do not negatively influence the QoE, and the QoE requirements of CroQoE are met.

Next, we evaluate the participants' ratings of the CroQoE app. For this, we added three questions at the end of the study, namely

1. Do you like the app design? (no/yes)
2. Are you satisfied with the time investment for the study? (no/yes)
3. What is your overall experience with the participation in this study? (bad/neutral/good)

Figure 8b shows the distributions of the participants' answers to the three questions. It can be seen that 94.59% of participants (70 participants) liked the app design and 95.95% of participants (71 participants) were satisfied with the time investment. The latter result again confirms that the waiting times introduced during the second iteration of the QoE-CDC do not negatively influence the experience of the participants. Moreover, considering the overall experience with the QoE study, 78.38% of participants (58 participants) had a positive experience, while 21.62% (16 participants) had a neutral experience. No participant had a bad experience when participating in the study. These ratings show that the application and iterations of both the HCDC and the QoE-CDC were successful, such that CroQoE meets all specified requirements, and is a usable and useful app for crowdsourced video streaming QoE studies.

To sum up, the implementation of CroQoE showcased the practical value and validated the QoE-CDC. Moreover, it could be seen that the proposed QoE-CDC can be easily combined with the established HCDC, which brings synergies to the design process. Resulting applications are able to reach both a high UX and a high QoE, which is a win–win situation for the user, the network provider, and the application provider.

Conclusion

This paper presented the Quality of Experience Centered Design Cycle (QoE-CDC), which gives guidelines to application developers with respect to network-specific requirements and QoE. The main steps of the cycle allow to identify, understand, and specify the QoE influence factors per context of use. Moreover, they allow to specify QoE requirements, and produce design solutions to meet these requirements. This way QoE aspects can be considered during the software development process, such that applications become network-aware and QoE-aware by design.

We showcased the practical value of the QoE-CDC by discussing past and potential improvements of smartphone applications, as well as on-demand music/video streaming, live video streaming/video conferencing, and mobile instant messaging applications. The presented improvements could have been resulted from employing the QoE-CDC and allow those applications to avoid QoE degradation, which would frustrate the end users. Moreover, our analysis highlighted open research questions and missing studies with respect to the subjectively perceived experience with these applications. Furthermore, we demonstrated the stepwise application of the combination of HCDC and QoE-CDC in the development of a new video study application to validate the usefulness of the QoE-CDC.

By further employing the QoE-CDC to existing and novel applications, and combining it with the human-centered design cycle (HCDC), new applications designs will evolve, which reach a high UX and avoid QoE degradation to also reach a high QoE. Together with QoE-aware traffic management, QoE-aware applications could unleash the maximum experience for end users, which constitutes a win–win situation for the user, the network provider, and the application provider.

Funding Open Access funding enabled and organized by Projekt DEAL.

Availability of data and material Not applicable.

Declarations

Funding Not applicable.

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Code availability Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will

need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Ergonomics of Human-system Interaction—Part 210: Human-centered design for interactive systems. Standard, International Organization for Standardization 2019
2. Olmstead K, Atkinson M. Apps permissions in the google play store. 2015. http://www.pewinternet.org/2015/11/10/apps-permissions-in-the-google-play-store/
3. Le Callet P, Möller S, Perkis A, et al. Qualinet white paper on definitions of quality of experience. European network on quality of experience in multimedia systems and services (COST Action IC 1003) 2012;3
4. Bargas-Avila JA, Hornbæk K. Old wine in new bottles or novel challenges: a critical analysis of empirical studies of user experience. In: Proceedings of the SIGCHI conference on human factors in computing systems, 2011; pp. 2689–2698
5. Hammer F, Egger-Lampl S, Möller S. Quality-of-user-experience: a position paper. Qual User Exper. 2018;3(1):9.
6. Pavic B, Anstey C, Wagner J. Why speed matters. 2020. https://web.dev/why-speed-matters/
7. Seufert M, Egger S, Slanina M, Zinner T, Hoßfeld T, Tran-Gia P. A survey on quality of experience of HTTP adaptive streaming. IEEE Commun Surv Tutor. 2015a;17(1):469–92.
8. Seufert M, Wassermann S, Casas P. Considering user behavior in the quality of experience cycle: towards proactive QoE-aware traffic management. IEEE Commun Lett. 2019;23(7):1145–8.
9. Juluri P, Tamarapalli V, Medhi D. Measurement of quality of experience of video-on-demand services: a survey. IEEE Commun Surv Tutor. 2015;18(1):401–18.
10. Baraković S, Skorin-Kapov L. Survey and challenges of QoE management issues in wireless networks. J Comput Netw Commun. 2013;2013(165146):1–28.
11. Qadir QM, Kist AA, Zhang Z. Mechanisms for QoE optimisation of video traffic: a review paper. Australas J Inf Commun Technol Appl. 2015;1:1.
12. Zinner T, Jarschel M, Blenk A, Wamser F, Kellerer W. Dynamic application-aware resource management using software-defined networking: implementation prospects and challenges. In: 2014 IEEE network operations and management symposium (NOMS), IEEE 2014; pp. 1–6
13. Schatz R, Schwarzmann S, Zinner T, Dobrijevic O, Liotou E, Pocta P, Barakovic S, Husic JB, Skorin-Kapov L. QoE Management for future networks. In: Autonomous control for a reliable internet of services. Springer, Cham 2018; pp. 49–80
14. Hartson R, Pyla PS. The UX book: process and guidelines for ensuring a quality user experience. Amsterdam: Elsevier; 2012.
15. Gkonos C, Iosifescu Enescu I, Hurni L. Spinning the wheel of design: evaluating geoportal graphical user interface adaptations in terms of human-centred design. Int J Cartogr. 2019;5(1):23–43.
16. Hooey B, Foyle D, Andre A. Integration of cockpit displays for surface operations—the final stage of a human-centered design approach. In: 2000 World aviation conference 2000; p. 5521
17. Babion JN, Ocampo W, Haubrich S, Yang C, Zuk T, Kaufman J, Carpendale S, Ghali W, Altabbaa G. Human-centred design processes for clinical decision support: a pulmonary embolism case study. Int J Med Inform. 2020;147:104196.
18. Realpe-Munoz P, Collazos CA, Hurtado J, Granollers T, Velasco-Medina J. An integration of usable security and user authentication into the ISO 9241-210 and ISO/IEC 25010: 2011. In: International conference on human aspects of information security, privacy, and trust, Springer 2016; pp. 65–76
19. Farooqui T, Rana T, Jafari F. Impact of Human-centered Design Process (HCDP) on Software Development Process. In: 2019 2nd international conference on communication, computing and digital systems (C-CODE), IEEE 2019; pp. 110–114
20. Pyla PS, Pérez-Quiñones MA, Arthur JD, Hartson HR. Towards a model-based framework for integrating usability and software engineering life cycles. CLOSING THE GAPS: software engineering and human-computer interaction 2004; p. 1
21. Campos JC. The modelling gap between software engineering and human-computer interaction. In: ICSE 2004 workshop: bridging the gaps II, 2004; pp. 54–61
22. Sandvine: the mobile internet phenomena report. Tech Rep 2020
23. Hoßfeld T, Schatz R, Varela M, Timmerer C. Challenges of QoE management for cloud applications. IEEE Commun Mag. 2012;50(4):28–36.
24. Möller S, Raake A. Quality of experience: advanced concepts. Applications and methods. Berlin: Springer; 2014.
25. Rec I. P. 910. Subjective video quality assessment methods for multimedia applications 2008; p. 910
26. Davis FD. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS quarterly 1989; pp. 319–340
27. Brooks P, Hestnes B. User measures of quality of experience: why being objective and quantitative is important. IEEE Netw. 2010;24(2):8–13.
28. Kjeldskov J, Graham C. A Review of Mobile HCI Research Methods. In: International conference on mobile human-computer interaction, Springer 2003; pp. 317–335
29. Egger S, Hoßfeld T, Schatz R, Fiedler M. Waiting times in quality of experience for web based services. In: Proceedings of the 4th international workshop on quality of multimedia experience (QoMEX). Yarra Valley, Australia; 2012a
30. Casas P, Schatz R, Wamser F, Seufert M, Irmer R. Exploring QoE in cellular networks: how much bandwidth do you need for popular smartphone apps? In: Proceedings of the 5th workshop on all things cellular: operations, applications and challenges, 2015; pp. 13–18
31. Research D. Failing to meet mobile app user expectations: a mobile user survey. 2015. https://techbeacon.com/sites/default/files/gated_asset/mobile-app-user-survey-failing-meet-user-expectations.pdf
32. Myers BA. The importance of percent-done progress indicators for computer-human interfaces. ACM SIGCHI Bull. 1985;16(4):11–7.
33. Biørn-Hansen A, Majchrzak TA, Grønli TM. Progressive web apps: the possible web-native unifier for mobile development. In: International conference on web information systems and technologies, SCITEPRESS 2017;2:344–351
34. Harrison C, Amento B, Kuznetsov S, Bell R. Rethinking the progress bar. In: Proceedings of the 20th annual ACM symposium on user interface software and technology, 2007; pp. 115–118
35. Li W, Wang M, Li W, Cai B, Shi Y. An Improvement on the progress bar: make it a story, make it a game. In: International conference on applied human factors and ergonomics, Springer 2020;pp. 394–401
36. Mejtoft T, Långström A, Söderström U. The effect of skeleton screens: users' perception of speed and ease of navigation. In: Proceedings of the 36th European conference on cognitive ergonomics, 2018;pp. 1–4
37. Seufert M, Burger V, Hoßfeld T. HORST-home router sharing based on trust. In: Proceedings of the 9th international conference on network and service management (CNSM 2013), IEEE. 2013; pp. 402–405

38. Chen QA, Luo H, Rosen S, Mao ZM, Iyer K, Hui J, Sontineni K, Lau K. Qoe doctor: diagnosing mobile app QoE with automated UI control and cross-layer analysis. In: Proceedings of the 2014 conference on internet measurement conference, 2014; pp. 151–164
39. Ghadiyaram D, Pan J, Bovik AC: A time-varying subjective quality model for mobile streaming videos with stalling events. In: Proceedings of SPIE Applications of Digital Image Processing XXXVIII. San Diego, CA, USA; 2015
40. Zeng K, Yeganeh H, Wang Z. Quality-of-Experience of streaming video: interactions between presentation quality and playback stalling. In: Proceedings of the IEEE international conference on image processing (ICIP). Phoenix, AZ, USA 2016
41. Seufert M, Hoßfeld T, Sieber C. Impact of intermediate layer on quality of experience of HTTP adaptive streaming. In: Proceedings of the 11th international conference on network and service management (CNSM). 2015; Barcelona, Spain
42. Sackl A, Egger S, Schatz R. Where's the Music? Comparing the QoE impact of temporal impairments between music and video streaming. In: 2013 Fifth international workshop on quality of multimedia experience (QoMEX), IEEE 2013; pp. 64–69.
43. Schwind A, Moldovan C, Janiak T, Dworschak ND, Hoßfeld T. Don't Stop the Music: Crowdsourced QoE Assessment of Music Streaming with Stalling. In: 2020 Twelfth international conference on quality of multimedia experience (QoMEX), IEEE 2020; pp. 1–6.
44. Schwind A, Haberzettl L, Wamser F, Hoßfeld T. QoE analysis of spotify audio streaming and app browsing. In: Proceedings of the 4th internet-QoE workshop on QoE-based analysis and management of data communication networks, 2019; pp. 25–30
45. Information technology. Dynamic adaptive streaming over HTTP (DASH). Part 1: Media presentation description and segment formats. Standard, International Organization for Standardization 2012
46. Ahmed A, Shafiq Z, Bedi H, Khakpour A. Suffering from Buffering? Detecting QoE Impairments in Live Video Streams. In: 2017 IEEE 25th international conference on network protocols (ICNP), IEEE, 2017; pp. 1–10
47. Yi G, Yang D, Bentaleb A, Li W, Li Y, Zheng K, Liu J, Ooi WT, Cui Y. The acm multimedia 2019 live video streaming grand challenge. In: Proceedings of the 27th ACM international conference on multimedia, MM '19, 2019; p. 2622-2626
48. Berndtsson G, Folkesson M, Kulyk V. Subjective Quality Assessment of Video Conferences and Telemeetings. In: 2012 19th International Packet Video Workshop (PV), IEEE, 2012; pp. 25–30
49. Zinner T, Abboud O, Hohlfeld O, Hossfeld T, Tran-Gia P. Towards qoe management for scalable video streaming. In: 21th ITC specialist seminar on multimedia applications-traffic, performance and QoE, 2010; pp. 64–69. Citeseer
50. Hoßfeld T, Schatz R, Zinner T, Seufert M, Tran-Gia P. Transport protocol influences on youtube videostreaming qoe. Tech Rep University of Würzburg, Institute of computer science; 2011a.
51. Skorin-Kapov L, Matijasevic M. Analysis of QoS requirements for E-health services and mapping to evolved packet system QoS classes. Int J Telemed Appl 2010
52. De La Torre Díez I, Alonso SG, Hamrioui S, López-Coronado M, Cruz EM. Systematic review about QoS and QoE in telemedicine and ehealth services and applications. J Med Syst. 2018;42(10):182.
53. Miller K, Al-Tamimi AK, Wolisz A. Qoe-based low-delay live streaming using throughput predictions. ACM Trans Multimed Comput Commun Appl (TOMM). 2016;13(1):1–24.
54. Yuang MC, Liang ST, Chen YG, Shen CL. Dynamic video playout smoothing method for multimedia applications. In: Proceedings of ICC/SUPERCOMM'96-international conference on communications, IEEE 1996;3: 1365–1369
55. Kalman M, Steinbach E, Girod B. Adaptive media playout for low-delay video streaming over error-prone channels. IEEE Trans Circ Syst Video Technol. 2004;14(6):841–51. https://doi.org/10.1109/TCSVT.2004.828335.
56. Rainer B, Timmerer C. Quality of experience of web-based adaptive http streaming clients in real-world environments using crowdsourcing. In: Proceedings of the 2014 workshop on design, quality and deployment of adaptive video streaming, 2014; pp. 19–24
57. Fiadino P, Schiavone M, Casas P. Vivisecting WhatsApp in cellular networks: servers, flows, and quality of experience. In: International workshop on traffic monitoring and analysis, Springer 2015; pp. 49–63
58. Seufert M, Wehner N, Casas P. App for dynamic crowdsourced qoe studies of http adaptive streaming on mobile devices. In: 2018 Network traffic measurement and analysis conference (TMA), IEEE 2018 ;pp. 1–2
59. Wehner N, Mertinat N, Seufert M, Hoßfeld T. Studying the impact of the content selection method on the video qoe on mobile devices. In: 2020 Twelfth international conference on quality of multimedia experience (QoMEX), IEEE 2020; pp. 1–4
60. Hoßfeld T, Hirth M, Redi J, Mazza F, Korshunov P, Naderi B, Seufert M, Gardlo B, Egger S, Keimel C. Best practices and recommendations for crowdsourced qoe-lessons learned from the qualinet task force crowdsourcing. COST Action IC 1003 Qualinet Tech Rep; 2014
61. Hoßfeld T, Seufert M, Hirth M, Zinner T, Tran-Gia P, Schatz R. Quantification of youtube QoE via crowdsourcing. In: 2011 IEEE International symposium on multimedia, IEEE 2011b; pp. 494–499.
62. Egger S, Reichl P, Hoßfeld T, Schatz R. "time is bandwidth"? narrowing the gap between subjective time perception and quality of experience. In: 2012 IEEE international conference on communications (ICC), IEEE 2012b; pp. 1325–1330
63. Strohmeier D, Jumisko-Pyykkö S, Raake A. Toward task-dependent evaluation of web-qoe: Free exploration vs. who ate what?. In: 2012 IEEE globecom workshops, IEEE 2012; pp. 1309–1313
64. Sylvain Saurel: How to retrieve an unique ID to identify android devices? https://medium.com/@ssaurel/how-to-retrieve-an-unique-id-to-identify-android-devices-6f99fd5369eb
65. International Telecommunication Union: ITU-T Recommendation P.910: subjective video quality assessment methods for multimedia applications 2008

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.