Streaming Characteristics of Spotify Sessions

Anika Schwind, Florian Wamser, Thomas Gensler, Phuoc Tran-Gia
Insitute of Computer Science, University of Würzburg
Würzburg, Germany
{anika.schwind | florian.wamser | trangia}@informatik.uni-wuerzburg.de

Michael Seufert, Pedro Casas
AIT – Austrian Institute of Technology
Vienna, Austria
{michael.seufert.fl | pedro.casas}@ait.ac.at

***Abstract*—Internet Service Providers need a thorough understanding of a service to maximize the Quality of Experience (QoE) of their customers by network management. Instead of quantifying the user satisfaction with long and cost-intensive subjective user studies, the QoE can often be estimated with the help of dedicated measurements of application and network parameters. We designed a QoE measurement tool for the popular audio streaming service Spotify that runs inside a Docker software container. The container is able to run headlessly as active measurement probe and emulates a user who is streaming audio files via Spotify. While streaming, network and application parameters are collected that have a high correlation to the user's QoE. The results of the measurements are used to characterize audio streaming in Spotify on application and network layer, and to evaluate important QoE factors.**

***Index Terms*—Audio Streaming; Spotify; QoE; Distributed Active Measurements**

I. Introduction

To understand and properly manage services delivered through their networks, Internet Service Providers (ISPs) need metrics to quantify the experience and satisfaction of their customers. Therefore, the concept of Quality of Experience (QoE) was developed, which focuses on the subjective user experience using a service or application. The QoE indicates the degree of delight or annoyance of a user of an application as perceived subjectively [1], and depends on various factors in the network or in the application. Such QoE-related performance indicators, which can be mapped to QoE using a QoE model, can be monitored on application layer, which requires a modification of the client application or the end user device. On network layer, Deep Packet Inspection (DPI) or statistical analyses of network and flow statistics (e.g., machine learning) can be applied to estimate the perceived QoE factors.

While video streaming services like YouTube and their QoE have been thoroughly investigated in previous research (e.g., [2], [3]), music streaming services like Spotify and their corresponding QoE have not been considered yet. Similar to video streaming, music streaming is increasingly popular and can be run in different versions and on various devices. However, the service usage is inherently different, as music streaming is often a long-lasting background service, which runs in parallel to other foreground services. Thus, it is less interactive and its traffic often has to compete with the foreground services of the user. Still, the service is continuously consumed and QoE degradations can be well perceived. Thus, in order to maximize the satisfaction of their customers with music streaming, operators need to understand this service in detail, especially regarding the traffic and the resulting QoE.

Since QoE is highly dependent on the actual application and user perception, usually extensive, multi-disciplinary subjective user studies are required to understand all influence factors. In order to avoid those long and cost-intensive user studies, the QoE can often be estimated with the help of dedicated measurements of application and network parameters. A complementary approach that has come into focus for ISPs is active measurements using dedicated measurement tools and mobile broadband testbeds. An active measurement software allows to emulate the user and to monitor both, QoE factors on application layer and network statistics.

In this work, audio streaming via Spotify is characterized on application layer as well as on network layer, and QoE factors are analyzed and discussed. Therefore, an active measurement concept is presented, which allows to headlessly stream audio files using the Spotify web application in a virtualized Docker environment, while monitoring performance parameters on application and network side that have a high correlation to QoE. The measurements show that the buffering is triggered only depending on the currently buffered playback time without prebuffering subsequent tracks in a queue. In addition, the network traffic of audio streaming via Spotify has a segment interarrival time of 10 s, and thus, is very consistent and can be easily modeled. Furthermore, we compare the streaming and buffering behavior under different limitations and map them to the corresponding QoE values.

In Section II, related work is summarized and discussed. Section III introduces and describes the QoE monitoring approach. Afterwards, in Section IV, audio streaming via Spotify is characterized on application and network layer and QoE factors are discussed under different bandwidth limitations. The paper is concluded in Section V.

II. Related Work

In this section we summarize existing and related work on audio streaming and its QoE. We start from the substantial body of work on streaming, which mainly focuses on video streaming, describe its basic idea, and transition to audio streaming. Afterwards, QoE approaches are outlined, which provide an introduction to QoE monitoring and QoE modeling in this context.

978-1-5386-2605-4/18/$31.00 ©2018 IEEE

A. Streaming Techniques

As with Spotify, streaming generally refers to the simultaneous transmission and playback of video or audio data. The challenge is to deliver the data in a timely manner. In [2], [4], current video streaming approaches and their challenges are exhaustively listed. In particular, fluctuating and unreliable network conditions can be overcome using buffer-based adaptive streaming in conjunction with client- and chunk-based HTTP requests. In [5], the streaming approach of YouTube is described and characterized. First works dealing with audio streaming are [6]–[9]. They often discuss VoIP calls in this context, but this case is different from ours because VoIP requires real-time constraints.

Technically, at least in the past, Spotify used a combination of client-server access and peer-to-peer protocol for streaming. A detailed description can be found in [9]. In [10], the P2P-based streaming network of Spotify is measured. The performance of the Spotify backend and its distributed key-value storage system is investigated in [11].

B. Quality of Experience for Audio Streaming

Quality of Experience (QoE) is the subjective user assessment of a service or application on the Internet. The related works in this area are focused either on the streaming quality or the user aspect. These works target the user behavior in Spotify [12], user requirements [13], or algorithms for generating music playlists, which provides the user with a coherent listening experience [14]. While QoE for video streaming mainly depends on initial delay, playout interruptions, and visual streaming quality [2], [5], there is only one work that explicitly refers to Spotify and audio QoE factors [15]. In this work, the impact of temporal impairments on the QoE of music streaming is investigated in subjective lab experiments. In addition to subjective studies, testbeds can be used, in which performance indicators with high correlation to the actual QoE are measured directly on the application and network layer on a large scale under realistic conditions [16]–[18]. Our work follows this approach and presents an active QoE measurement tool for Spotify.

III. Measurement Concept

To characterize a Spotify audio streaming session and to quantify QoE related factors of Spotify, a measurement tool was designed, which collects data on application as well as on network layer. Here, a user is emulated using a Docker container, who streams audio in mobile networks. The container can be run without user interaction at simple network measurement nodes, i.e., downsized, headless Linux PCs as widely used in mobile broadband testbeds [17]. Docker was used to enable a simplified distribution on the measurement nodes and to create a uniform, reproducible measurement environment so that reliable measurements can be made. As it is not possible to collect application layer parameters directly from the Spotify player, the Spotify API was used to query information about the audio stream. In the following section, the measurement setup and technical details about the Docker container and the polling server are described. Afterwards, the monitored parameters on application and network layer are presented.

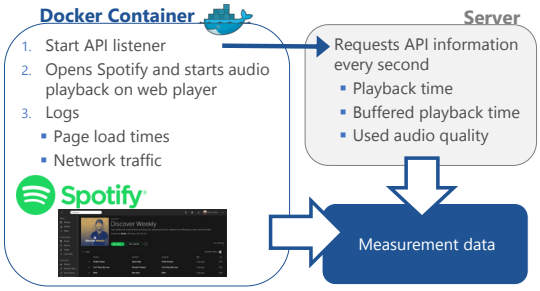


Fig. 1: Setup of the measurement tool

A. Technical Implementation

The Docker container is based on the *debian:stretch*[1] distribution, whose current version is Debian 9.3. The web browser automation tool *Geb*[2] is used within the Docker container to simulate a user requesting an audio file in Spotify in a Firefox browser. To play out audio files even when no sound card is available, *PulseAudio*[3] is used. Network traffic information is measured during the whole experiment by using Firefox's *HTTP logging* functionality[4]. If enabled, Firefox automatically logs all HTTP-related information from the browser, for example, all request and response headers, and stores the information in a file. On application layer, the audio streaming in the docker container is monitored with the help of a polling server. The server is running Apache Tomcat and queries the *Spotify API*[5] to retrieve and log the selected QoE relevant performance indicators of the application.

The whole measurement setup is illustrated in Figure 1. First, Geb is used to open a Firefox browser. By starting the browser, a special web page on the polling server is called to activate the API requests from the server. From now on, the current and buffered playtime are queried and logged by the polling server every second. At the same time, the Docker container continues its workflow by opening the login page of the Spotify web player in Firefox and logging into a Spotify account. Afterwards, a dummy track is started in the Spotify web player. This action clears the cache of the account, so that the actual track, which will be analyzed, starts at its beginning and all segments have to be downloaded. After playing out the dummy track for 5 s, the actual song is started and fully played. Then, the user is logged off from Spotify and the container is terminated. During the whole experiment, all page load times are recorded. In addition, the network traffic during the streaming is logged. All resulting measurement logs can

[1] https://hub.docker.com/_/debian/
[2] http://www.gebish.org/
[3] https://www.freedesktop.org/wiki/Software/PulseAudio/
[4] https://developer.mozilla.org/docs/Mozilla/Debugging/HTTP_logging
[5] https://beta.developer.spotify.com/documentation/web-api/

TABLE I: Monitored performance indicators

Application layer	Current playtime per second Buffered playback time per second Initial delay Page load times
Network layer	Audio bitrate Segment sizes Segment interarrival times Segment download times

be used to characterize the streaming sessions of Spotify and quantify important QoE factors, as shown in Section IV.

B. Monitored Data - QoE Key Performance Indicators

While an audio file is streamed within the Docker container via Spotify, several performance indicators are monitored, which are presented in Table I.

On application layer, the current playtime as well as the buffered playtime per second is logged during the streaming of the audio file. In addition, the initial delay – the time between the request of the audio playback (click on the play button) and the start of the playback – is monitored. Besides the streaming parameters, also all page load times are logged from the login page of Spotify until the end of the audio playback.

To monitor the network traffic during the experiment, Firefox's HTTP logging is used. These data includes the headers of all sent requests and responses. They can be used to calculate the bitrate of the audio file or the size of the audio segments as well as their interarrival times or download durations.

IV. Spotify Streaming Characteristics

This section presents the characteristics of a Spotify audio streaming session. Therefore, we used different versions of Spotify and monitored the streaming with the polling server. This allows to describe the session on application layer by evaluating the playback and buffer behavior of an audio stream and showing details of the used audio file. Afterwards, network layer parameters of a streaming session in the web player are shown. Here, general network characteristics of Spotify audio streaming are described and a detailed time schedule with three phases is presented. Table II summarizes the findings.

The encoding, and thus, the quality of the streamed audio file, depends on the used player and account (standard or premium user)[6]. As a standard user, Spotify is free to use but includes advertisements. To avoid advertisements, it is possible to pay a specific fee to upgrade to a premium account. For the web player and the desktop application, there are two qualities available, while the Android and iOS application offer a third, very low quality level in addition. Adaptivity of the used quality depending on the given network conditions is only available for the Android and iOS app. Here, it is more likely that the users are connected to the mobile network and moving, and thus, face fluctuating bandwidth during the streaming. In this case, the adaptivity can help to align the streaming quality, i.e., bitrate, with the current network conditions, i.e., bandwidth. To stream an audio file via Spotify, regardless of the used player, a persistent HTTP connection over TCP is used. The audio file is a DASH fragmented MP4 [19]. Using this standard, the audio file is split into a header file and several fragments, which can be downloaded one after the other.

[6]https://support.spotify.com/using_spotify/system_settings/high-quality-streaming/

A. Application Layer Parameters

On application layer, the focus of the analysis is set on the playback and buffer behavior of Spotify audio streaming sessions using the web player in Firefox within our measurement container. Thereby, the streaming behavior was consistent for different audio files (cf. Table III). In the following, the numeric results of the streaming of one audio file are shown, which is the song *Alles neu* by Peter Fox (track No. 1). Each audio file has a Spotify-wide track ID, which is "5hqxBvQJ3XJDSbxT9vyyqA" for this track. The total duration of this audio file is 260.09 s which leads to a size of 4.46 MB for a standard user streaming via the Spotify web player (encoding: AAC, bitrate: 128 kbit/s).

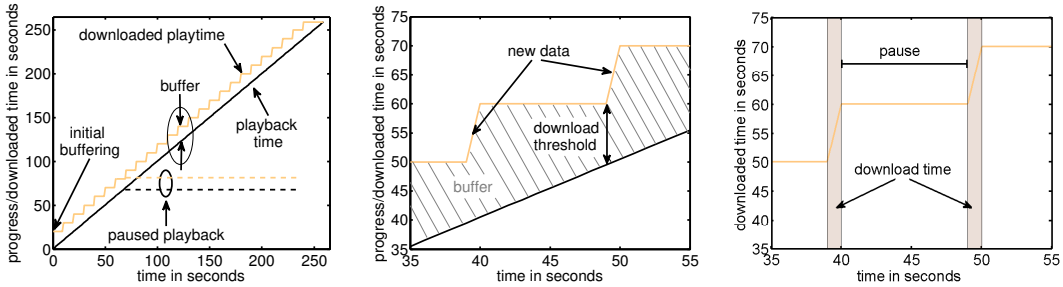
In all plots of Figure 2, the x-axis indicates the time in seconds after the start of the audio playback while the y-axis specifies the progressed or buffered time in seconds, respectively. In Figure 2a, the current playtime of the audio file is illustrated as black graph. As it is constantly increasing, no interruptions occurred during the playback. The stepwise orange graph shows the downloaded playtime during the audio playback. Thus, the space between the downloaded playtime and the playback time represents the buffer and is marked with an oval. The buffer can be seen in more detail in Figure 2b, marked in gray.

The playback of an audio file only starts after the initial buffering of the first segment containing 10 s playtime. For this track and setting, this first segment has a size of 139.53 KB. Whenever the buffer decreases to the threshold of 10 s, new data is downloaded to a maximum of 20 s playtime, which explains the stepwise increase of the downloaded playtime graph. Thus, during the whole playback, the buffer stays between 10 s and 20 s, and in the worst case, a network outage of 10 s can be tolerated. In case of insufficient bandwidth, stalling will only occur at segment borders if the download takes longer than 10 s, and the stalling lasts until the segment was downloaded completely. At the end of the track the buffers decreases to zero without prebuffering the subsequent track. Only when the playback of the track is finished the first segment of the subsequent track is requested. By applying these thresholds of 10 s, Spotify trades off the tolerance against network outages to limit the amount of unnecessarily downloaded data that would be wasted when the streaming is aborted by the user.

If the playback is manually paused, as indicated by the dotted lines in Figure 2a, the playback time does not increase anymore. After already requested segments were downloaded, the downloaded playtime stops increasing. As no additional segments are requested, the buffer stays constant on a level between 10 s and 20 s. This behavior and the respective thresholds could be observed for all investigated audio tracks.

TABLE II: Audio streaming characteristics on application layer

Audio settings	Encoding	*Web player*: standard user: AAC 128 kbit/s, premium user: AAC 256 kbit/s *Desktop app*: standard user: Ogg Vorbis 160 kbit/s, premium user: Ogg Vorbis 320 kbit/s *Android/iOS app*: normal quality: Ogg Vorbis 96 kbit/s, high quality: Ogg Vorbis 160 kbit/s, extreme quality: Ogg Vorbis 320 kbit/s (premium user) Adaptivity: only for Android and iOS app
	Container format	DASH fragmented MP4, 10 s segments
	Transmission protocol	HTTP persistent connection over TCP
Streaming characteristics (unlimited bandwidth)	Initial buffering	10 s playtime (std: 0, fixed),
	Buffering	[10; 20], avg: 15 s, linear between subsequent requests due to application layer download algorithm
	Buffer threshold	Minimum buffer: 10 s
	Pause behavior	Buffer stays at the same level (10 s-20 s). no further download requests



(a) Streaming characteristic (b) Download threshold (c) Segment requests

Fig. 2: Playtime, buffer, and request characteristic of a Spotify audio streaming session

B. Network Layer Parameter

Audio files streamed via Spotify are DASH fragmented MP4 with a 96 Bytes header. Each segment covers a playback duration of 10 s. To have a closer look at the used bitrates within the Spotify web player, Table III shows the bitrate characteristics of different audio tracks streamed using a standard account within our measurement container. On average, the download bitrate of all tracks is between 136 and 141 kbit/s. Considering these six tracks, an average segment has a size of 169.2 KB (1.353 Mbit). Having a look at the variance and the standard deviation, the values are very low except for track No. 1. In this special case, the first segment was much smaller than the others (139.53 KB compared to an average segment size for this song of 171.49 KB). This could be caused by the comparatively quiet intro at the beginning of the track.

As each segment covers a fixed playtime of 10 s, the number of segments per track depends on its duration. The number of segments per track is calculated as length of track in seconds divided by 10 and rounded up. For example, track No. 4 has a duration of 204.5 s, and thus, is split in 21 segments. There is one exception: If the last segment would be very small, as for track No. 1 with a duration of 260.09 s, the number of segments is rounded off, here to 26 segments.

Figure 2c shows the sequence of requesting and downloading new segments. A new segment is requested when the buffered playback time falls below 10 s. The download time (marked in brown) starts immediately and ends as soon as the whole segment is downloaded. For track No. 1, the average download time of one segment was 28.4 ms using a Internet connection with more than 700 Mbit/s. The next 10 seconds, the playback continues without the download of a new segment until the buffer threshold is reached again. In total, the download of the whole track takes 794 ms which is about 0.31% of the total duration of the track (260 093 ms corresponding to 4.33 min).

Three Phases: The streaming of an audio file using Spotify can be divided into three phases: beginning, steady phase, and depletion phase. By requesting the start of the stream, first, header information with an average size of 1 781 Bytes is downloaded. The average is calculated using the six tracks specified in Table III with a standard deviation of 31.9 Bytes. Immediately after this, the segment with the first 10 s playtime is requested and downloaded twice. The playback of the audio file only starts after the second download is finished. This is quite surprising and we could not figure out the reason for this behavior. As soon as the playback starts, the next segment of 10 s playtime is requested and consequently a buffer of almost 20 s is reached (20 s - download time of about 20 ms). In total, the beginning phase includes four requests with a total size of 506.7 KB on average.

After that, the steady phases starts. Here, the buffer is kept constant containing between 10 s and 20 s playtime. As soon as the buffer falls below the download threshold of 10 s, the next segment is requested. Thus, the interarrival time of the segments is 10 s if the playback runs without interruption.

TABLE III: Download bitrate characteristics of multiple tracks

No.	Track-ID	Avg kbit/s	Min kbit/s	Max kbit/s	Variance	Standard deviation
1	5hqxBvQJ3XJDSbxT9vyyqA	137.14	111.62	141.85	29.79	5.46
2	6tvQzeCPwOytDlBYCIOY0n	140.11	135.39	144.48	4.62	2.15
3	5ELRhUPf1f2Qc1x7hTFQ9l	138.60	118.97	143.06	5.60	2.37
4	0AQl7UJ5IKRZRzCxQFOwwo	136.53	133.15	138.28	0.81	0.90
5	7hhpH9L5L6jBTimrATV7fS	137.86	134.97	141.66	3.52	1.88
6	3lmebIvU89RA5336bjBIaQ	139.61	132.36	145.61	5.75	2.39

When all segments are downloaded, the depletion phase starts. Here, no further segments have to be requested and the buffer decreases until the audio playback is finished.

In case of a subsequent track in the playing queue, the transition to the next song does not behave differently, but the procedure mentioned above is just repeated. Hence, no prebuffering is made by the Spotify streaming algorithm, only the header information of the next track is downloaded before the end of the current track. The request of the first segment of the next audio file is sent when the previous track is fully played and the buffer is empty.

It can be seen that the network traffic of an audio streaming via Spotify is very consistent and shows a deterministic inter-arrival time of the segments of 10 s. Thus, the traffic caused by the Spotify streaming sessions should be identifiable although the network traffic is encrypted. The streaming behavior also can be easily simulated using the time schedule described in the three phases above.

C. Influence of Bandwidth Limitations

To analyze the influence of the given network capacity on the streaming behavior, measurements were run under different bandwidth limitations. Here, again track No. 1 was used, streaming from a premium user account using the measurement container. The average bitrate of this track is 270.27 kbit/s with a minimum of 211.64 kbit/s and a maximum of 279.21 kbit/s. The track was streamed using ten different bandwidth capacities in the range from 220 kbit/s to 460 kbit/s and with no bandwidth limitation (unlimited). For each setting, a minimal number of ten measurements were conducted and are considered in the following evaluations.

Previous user studies [2], [15] showed that initial delay and stalling, i.e., the interruption of the playback due to buffer depletion, are the most important influence factors, which affect the QoE of audio streaming. Thus, these two factors were analyzed in more detail by comparing them under different bandwidth capacities in Figure 3. The x-axes show the bandwidth limitation, while the y-axes present the initial delay and the total stalling time in seconds, respectively. Each bar represents the average value of the ten different measurements under the same conditions and the corresponding 95 % confidence interval is shown on top. The y-axes on the right of each figure indicate the mean opinion score (MOS). The red stars represent the MOS values corresponding to the occurred initial delays and total stalling times, and were calculated using the model presented in [15]. Note that this model considers the two factors (initial delay and stalling) separately. The model does not provide an overall MOS value, which considers the combination of both factors, although, in practice, both degradations would occur at the same time when the bandwidth is limited.

Having a look at the initial delays, they range from an average of 28.77 s for a bandwidth limit of 220 kbits/s down to 0.65 s for unlimited bandwidth. It can be seen that a bandwidth capacity of 460 kbits/s or higher corresponds to a good user satisfaction (MOS$\approx$4). However, even for a long initial delay of 28.77 s, the user satisfaction is still fair (MOS$\geq$3).

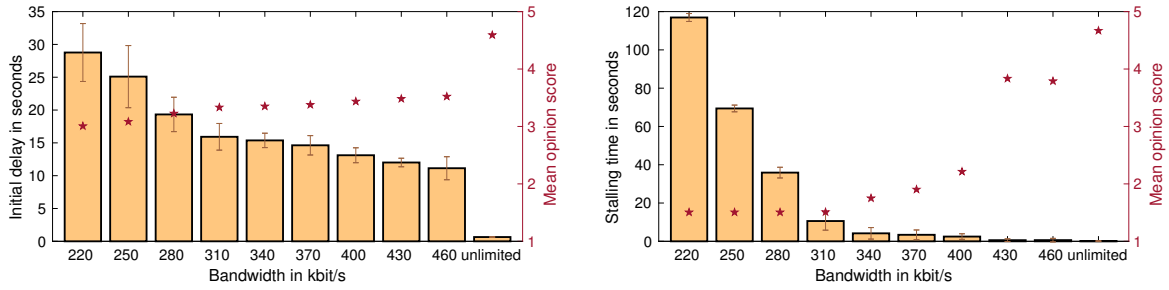
Although one would expect that no stallings occur when the set bandwidth capacity is higher than the maximum bitrate of the track of 279.21 kbit/s, even for higher capacities stalling occurred. According to [15], audio streams with more than 2 s total stalling time were always perceived as disturbing (MOS$\leq$2). Thus, Spotify streaming with a bandwidth capacity of 400 kbits/s or lower is not accepted by the users. With a capacity of 430 kbits/s or higher (3G), leading to a total stalling time of less than 0.7 s, the users rate the streaming as good (MOS$\approx$4). As the impairment from stalling is modeled with an exponential function, even short stalling has a high impact on the user experience. Thus, a sharp jump of the MOS values can be noticed from an average MOS of 2.21 for a bandwidth limitation of 400 kbit/s to an average MOS of 3.83 for 430 kbit/s.

It can be seen that stalling has a higher influence on the user's satisfaction than the initial delay. To have a high user satisfaction, a bandwidth capacity of nearly twice as high as the played bitrate is necessary. Nevertheless, as mentioned above, the model is not optimal to calculate the QoE as it does not consider the combined effect of initial delay and stalling. In real world scenarios, like in our measurements, initial delay and stalling will occur combined and thus, have to considered together in the MOS calculation.

V. Conclusion and Outlook

In this paper, we characterized Spotify audio streaming sessions on application as well as on network layer. Therefore, an active measurement tool was designed, which allows to headlessly stream audio files via Spotify from a browser in a virtualized Docker environment, while monitoring performance parameters that have a high correlation to QoE. Using this tool, the Spotify audio streaming sessions were characterized, for example, by analyzing the different audio encodings used for different Spotify players and accounts.

We found that the download of audio segments is only triggered according to a threshold of the buffered playback



(a) Initial delay

(b) Total stalling time

Fig. 3: Comparison of occurred stallings and initial delays for different bandwidth limitations

time without prebuffering for subsequent tracks in a queue. For every stream, the network traffic is consistent, when sufficient bandwidth is available, as the interarrival time of segments is constantly 10 s. In addition, we explained the three phases of audio streaming: beginning, steady phase, and depletion phase.

Finally, we investigated the streaming behavior of Spotify sessions under different bandwidth limitations. Here, we compared the initial delay as well as the total stalling time and mapped it to the corresponding QoE using the QoE model of [15]. We found out that Spotify streaming needs a bandwidth capacity nearly twice as high as the played bitrate to reach a high user satisfaction.

In future work, large scale measurements in a mobile broadband network will be conducted to verify the results in the mobile network. In addition, the measurement tool will be extended to also cover other Spotify players like the Android or the desktop app. Furthermore, a detailed subjective user study on the QoE of audio streaming has to be done which also takes other QoE factors like page load times into account.

Acknowledgment

This work was partly funded in the framework of the EU ICT project MONROE (H2020-2014-ICT-644399, through open call project Mobi-QoE). The authors alone are responsible for the content.

References

[1] P. Le Callet, S. Möller, and A. Perkis, Eds., *Qualinet White Paper on Definitions of Quality of Experience (2012)*. Lausanne, Switzerland: European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), 2012.

[2] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[3] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of Quality of Experience of Video-on-Demand Services: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[4] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "Energy efficient multimedia streaming to mobile devices—a survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 579–597, 2014.

[5] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hoßfeld, "Modeling the YouTube Stack: from Packets to Quality of Experience," *Computer Networks*, vol. 109, no. 2, pp. 211–224, 2016.

[6] B. W. Wah, X. Su, and D. Lin, "A survey of error-concealment schemes for real-time audio and video transmissions over the internet," in *Proceedings of the International Symposium on Multimedia Software Engineering*. IEEE, 2000, pp. 17–24.

[7] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel, "Scalable on-demand media streaming with packet loss recovery," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 97–108, 2001.

[8] X. Chen, C. Wang, D. Xuan, Z. Li, Y. Min, and W. Zhao, "Survey on QoS management of VoIP," in *Proceedings of the International Conference on Computer Networks and Mobile Computing (ICCNMC)*. IEEE, 2003, pp. 69–77.

[9] G. Kreitz and F. Niemela, "Spotify–large scale, low latency, P2P music-on-demand streaming," in *Proceedings of the 10th IEEE International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2010, pp. 1–10.

[10] M. Goldmann and G. Kreitz, "Measurements on the spotify peer-assisted music-on-demand streaming system," in *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2011, pp. 206–211.

[11] R. Yanggratoke, G. Kreitz, M. Goldmann, R. Stadler, and V. Fodor, "On the performance of the spotify backend," *Journal of Network and Systems Management*, vol. 23, no. 1, pp. 210–237, 2015.

[12] B. Zhang, G. Kreitz, M. Isaksson, J. Ubillos, G. Urdaneta, J. A. Pouwelse, and D. Epema, "Understanding user behavior in spotify," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 220–224.

[13] L. Arhippainen and S. Hickey, "Classifying music user groups and identifying needs for mobile virtual music services," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. ACM, 2011, pp. 191–196.

[14] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, "Playlist prediction via metric embedding," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 714–722.

[15] A. Sackl, S. Egger, and R. Schatz, "Where's the Music? Comparing the QoE Impact of Temporal Impairments Between Music and Video Streaming," in *5th International Workshop on Quality of Multimedia Experience (QoMEX)*, Klagenfurt, Austria, 2013.

[16] A. Schwind, M. Seufert, Ö. Alay, P. Casas, P. Tran-Gia, and F. Wamser, "Concept and implementation of video qoe measurements in a mobile broadband testbed," in *IEEE/IFIP Workshop on Mobile Network Measurement (MNM'17)*, Dublin, Ireland, 6 2017.

[17] O. Alay, A. Lutu, R. García, M. Peón-Quiròs, V. Mancuso, T. Hirsch, T. Dely, J. Werme, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, A. S. Khatouni, M. Mellia, M. A. Marsan, R. Monno, and H. Lonsethagen, "Measuring and assessing mobile broadband networks with monroe," in *Proceedings of the 17th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2016, pp. 1–3.

[18] S. Galetto, P. Bottaro, C. Carrara, F. Secco, A. Guidolin, E. Targa, C. Narduzzi, and G. Giorgi, "Detection of video/audio streaming packet flows for non-intrusive QoS/QoE monitoring," in *IEEE International Workshop on Measurement and Networking (M&N)*. IEEE, 2017, pp. 1–6.

[19] T. Siglin, "Unifying Global Video Strategies: MP4 File Fragmentation for Broadcast, Mobile and Web Delivery," Transitions, Inc., Tech. Rep., 2011.