Network-based Online Video freeze prediction in HTTP Adaptive Streaming

Tingyao Wu, *Member, IEEE,* Stefano Petrangeli, *Member, IEEE,* Rafael Huysegems, *Member, IEEE,* and Tom Bostoen, *Life Fellow, IEEE*

***Abstract*—HTTP adaptive streaming (HAS) has become a prevailing technology for media delivery technology over mobile and fixed networks. The client's Quality of Experience (QoE) for HAS video sessions is particularly of interests in network providers and Content Delivery Network (CDN) providers. But typically, network providers are not able to assess to clients to obtain QoE relevant parameters, such as freeze, initial loading time, quality switches, etc.**

In our previous work, we designed a HAS QoE monitoring system based on the sequence of HTTP GET requests collected at the CDN nodes. The system relies on a technique called session reconstruction to retrieve the major QoE parameters without modification of the clients. However, session reconstruction is computationally intensive and requires manual configuration of reconstruction rules. To overcome the limitations of session reconstruction, this paper proposes a scalable machine learning (ML) based scheme that detects video freezes using a few high-level features extracted from the network-based monitoring data. We determine the discriminative features for session representation and assess five potential classifiers. We select the C4.5 decision tree as classifier because of its simplicity, scalability, accuracy, and explainability. To evaluate our solution, we use traces of Apple HTTP Live Streaming video sessions obtained from a number of operational CDN nodes and traces of Microsoft Smooth Streaming video sessions acquired in a controlled lab environment. Experimental results show that an accuracy of about 98%, 98%, and 90% can be obtained for the detection of a video freeze, a long video freeze, and multiple video freezes, respectively. Excluding log parsing, the computational cost of the proposed video-freeze detection is 33 times smaller than needed for session reconstruction.

***Index Terms*—HTTP Adaptive Streaming, Decision Tree C4.5, Freeze Prediction, Quality of Experience**

I. Introduction

VIDEO streaming has occupied more than half of traffic over the Internet. In recent years, video delivery over traditional best-effort Internet becomes very popular. Among this, HTTP Adaptives Steaming (HAS) is increasingly adopted and has become a key technology. HTTP adaptive streaming systems [1] include MPEG-DASH, Apple's HTTP Live Streaming, Microsofts Smooth Streaming, Adobes Dynamic Streaming, etc. Typically, in a HAS architecture, video content, hosted on an HTTP Web server, is encoded in different quality levels (bit-rates), and chunked into independent segments. A HAS client requests the segments with different qualities in a linear way using HTTP GET requests and downloads them using plain HTTP progressive download. The retrieved segments can be played back as a seamless video, possibly switching among different quality levels. The key feature of HAS is that it is the client that is responsible for determining which quality level to download according to its available resources. Server/network providers do not have the control over the quality requests. One of the main advantages of introducing HAS as a delivery/playout method is that, the client could adapt its quality requests based on the perceived bandwidth: it enables the client to smoothly play the video with a low quality level even when the perceived bandwidth is very limited, while when a high bandwidth is available, it supports the client to demand high quality levels.

With the growing pervasion of HAS deployments, network and CDN providers raise interests in knowing the end-user quality of experience (QoE) of HAS sessions over their network, because the user-centric QoE reveals the satisfaction of their users. Although QoE is the subjective perception of end-users, the objective QoE relevant parameters, including the statistics of bit-rates, freezes frequency and freeze durations, etc, can be used to model the end-users' Mean Opinion Score (MOS) [2]. This inspires network/CDN providers to obtain the QoE relevant parameters, as an indirect way to evaluate the satisfaction of end-users. While a HAS client may report these parameters to the Web server (for instance, enabling Advanced Logging in Microsoft Internet Information Services (IIS) could require Microsoft Silverlight clients to send the status of medie content consumption to the IIS server at a regular time basis), typically it does not report these parameters to intermediate network nodes; they are only able to intercept the sequence of HTTP GET requests sent from the client to the server/CDN. So retrieving QoE relevant parameters from the sequence of HTTP GET requests within a single video session is a plausible method to evaluate end-user's QoE. Indeed, in our previous study [3], we demonstrated that from the sequence of HTTP GET requests collected at intermediate network elements, the HAS session can be reconstructed to derive all QoE related parameters. These parameters include the average playout quality, changes in the play-out quality, rebuffering due to buffer starvation, rebuffering caused by interactivity, etc. The session reconstruction technique relies on some manually rules discovered by the trial-and-error method, without modifying the original HAS client, and thus is a feasible network based method to evaluate the video delivery quality of HAS.

T. Wu, R. Huysegems and T. Bostoen are with Bell Labs, Alcatel-Lucent, Copernicuslaan 50, B-2018 Antwerpen, Belgium. e-mail: (tingyao.wu@alcatel-lucent.com).

S. Petrangeli is with Department of Information Technology (INTEC), Ghent University- iMinds, Gaston Crommenlaan 8 (Bus 201), 9050 Ghent, Belgium.

Manuscript received April 19, 2005; revised September 17, 2014.

Among all objective QoE relevant metrics, freeze/rebuffering is undoubtedly the most deleterious factor to destroy end-users' satisfaction and engagement. The 2015 annual report of Conviva [4] shows that 28.8% of video sessions experience at least one rebuffering event, but only 1% increase of rebuffering could reduce the video engagement with 14 minutes. Identifying freeze/rebuffering events is thus the primary task for retrieving the QoE relevant metrics. Although the session reconstruction in our previous work has shown its effectiveness, the manual effort of the trial-and-error rule exploration reduces its generalization capability; one has to manually re-configure the construction rules for a new HAS deployment. Meanwhile, for about 70% buffering-free video sessions, such *no-freeze* information is not known until a session has been fully reconstructed. Since session reconstruction needs a time-consuming segment-by-segment alignment, it is more desirable to have a quick method to identify whether a video session contains freeze(s) before taking session reconstruction. This inspires us to develop a scalable freeze detection/prediction technique at the intermediate network node for a video session, ideally without the requirement of any deployment-dependent reconstruction rules. In fact, in [5], we have demonstrated the concept of using a decision tree-based freeze detection technique to retrieve the freeze information for a full video session with a high accuracy, whereof this paper represents an extension.

The main contributions of this paper are two-fold. First, we investigate a few feasible machine learning techniques and select C4.5 decision tree as our classifier. The selection compromise the computational cost, simplicity, stability and understandability. Second, as the extension of our work in [5], which assumes that there is no user interactivity, we examine the performance of the proposed method in the existence of user interactivity, including pause and re-positioning. Third, the approach is transplanted into an OpenFlow-based framework to help clients avoiding video freezes under limited bandwidth conditions. In this scenario, the approach is used to online predict whether the normal delivery of a video segment could lead to a freeze at the client side in a near future; if a potential freeze is foreseen, a priority to the current segment is given to avoid the true occurrence of the freeze.

The remainder of this paper is organized as follows. Section II gives the overview of the research of HAS QoE and the introductin of OpenFlow. Section III presents the proposed freeze detection/prediction framework, while in section V we demonstrate the designed experments and the corresponding results. The conclusions and future work are presented in section VI.

II. Related work

A. QoE in HTTP Adaptive Streaming

Quality of Experience (QoE) is a user-perceived metric that describes the the degree of satisfaction of the user of an application or service. In video service, the subjective user test is the golden rule metric measurement for the QoE. The subjective perception test can be easily influenced by the human psychological factor and by the bias of user interests. Moreover, it is typically a time-comsuming and expensive procedure. Thus some intrinsic quality metrics, including Peak Signal to Noise Ratio (PSNR), Structural Similarity (SSIM), and Video Quality Metric (VQM), are taken into account in traditional video QoE analysis, attempting to map these metrics to the degree of human satisfaction.

Lately, the HAS based QoE research has already received a lot attention. As HAS could adapt to different network enviroments by increasing/decreasing quality levels and also by filling/depleting the buffer filling level, it could mitigate the freeze occurrences. Thus it is considered to be an effective technique to enhance the end-user's QoE. On the contrary, the reaction of classial HTTP video streaming (e.g. progressive downloading) to fluctuating network conditions only relies on the changing of buffer level. [6] studies the QoE of DASH in comparison with fixed-bitrate streaming and gives the evidence that adaptive streaming improves end-users' subjective perception greatly compared with fixed-rate streaming in terms of QoE. This advantage is also verified in [7], which compares adaptive and fixed-rate streaming under vehicular mobility circunstance and reveals that HAS can effectively reduce freeze by 80% when bandwidth decreases, could better utilize the available bandwidth when bandwidth increases.

Comparing to traditional video delivery technique, the HAS extends single quality level to mutliple quality levels with respective to resolution. This extended dimension introduce a new factor, changing video quality levels (i.e., adaptation) and thus has new impacts on QoE. In fact, traditional quality metrics (PSNR, SSIM and VQM, etc) fail to predict user-perceived video quality due to quality fluctuations inherent to HAS systems [8]. Instead, the playback start time, number of freezes, duration of freezes, ratio of freezes, quality levels (together with their first order and second order statistics), bitrate switching frequency, etc are the objective representives [9] that are able to capture the factors that influence the user's QoE in HAS system. In [10], the authors overview the recent studies targeting the optimization of HAS adaptation strategy, concerning subjective evaluation of HAS QoE, and also highlight the challenges and open research questions related to QoE assessment in HAS. Seufert et al. [11] provides a comprehensive survey for the relationship of HAS and subjectively perceived quality. In the survey, the main influence factors of QoE of HTTP video streaming are discussed, and the influence of temporal resolution, spatial resolution and video quality on QoE are also presented.

B. Freeze reduction in HAS client

Almost all research agree that video freeze sharply degrades the QoE. Freeze, also called stalling/interruption/buffer-underruning, etc in the literature, is the stopping of video playback because of the depletion of playout buffer. This typically happens when the received video segment in time is continuously lower than the speed of playback, leading to the decreasing of buffer filling level at the client. Eventually, insufficient data is available in the buffer and the playback of the video has to stop. The playback continues when a certain amount of video data is received again. Freeze is a sudden

unexpected interruption during watching. This unexpectation is probably processed differently by the human sensory system, and degrades the users' satisfaction [12].

By nature, HAS is designed to reduce freeze in volatile network situation. However, as the increasing demand of high quality videos, it is unavoidable to experience the occurrence of freeze during HAS video playout, especially when a sudden bandwidth drop occurs [13], [14]. There have been quite a few studies concerning further reducing the occurrence of freezes in HAS by elaborately designing the client rate adaptation heuristic. For instance, [15] and [16] use the reinforcement learning technique as a decision agent at the HAS client. The agent learns to request the most suitable quality level given the client's instant status by progressively optimizing the QoE-related reward. In the design, the punishment to a freeze is about 2 order of magnitude than low qualities and quality switching. In [17], a rate adaptation algorithm for HAS was proposed to detect bandwidth changes using a smoothed HTTP throughput measure based on the segment fetch time. It deploys a step-wise increase/aggressive decrease method to switch up/down between the different quality levels to prevent from the freeze that might be incurred by sudden bandwidth drops. In [18], [19], the authors argue that only observing and controlling the playback buffer, without having to take network capacity into account, is already able to avoid unnecessary rebuffering and control the delivered video quality.

C. Network based freeze reduction/discovery

In-network HAS QoE monitoring provides a solution for network/CDN providers to reveal how end-user is satisfied with the video service. [3] demonstrates a session reconstruction technique lying at an intermediate node to fully recover how the media content was consumed at the client. The technique could reconstruct the place and duration of the occurence of freeze, and the pause and re-positioning initiated by the end-users, using the sequence of intercepted HTTP GET messages sent by the client to the server. By the session reconstruction, all QoE metrics can be retrieved. However, this technique relies on manual effort to define some heurisitc dependent rules thus it has scalability issues when dealing with newly emerged HAS clients.

To reduce the impact of freeze and optimize the QoE for multiple clients, some researchers also seek for in-network solutions for HAS. In [20], [21] presents merits of deploying intelligent network elements that manage the QoE in HAS delivery network. It was argued that a combination of HAS enabled clients with the operator-centric of user-centric policy enforced at the intelligent proxies leads to the best solution. [22] presents an interesting OpenFlow-based framework, in which an in-network controller is in charge of introducing prioritized delivery of HAS segments based on status collected from both the network nodes and the HAS clients. Once a HAS client is approaching buffer-underrun, the prioritization mechanism at the controller is triggered for the client, then the requested segment (and possibly the next few segments) will be delivered to the client with a high priority, in order to avoid the potential freeze really happening. However, as we stated in the section I, in reality, the network node usually is not able to acquire the report from the client. Moreover, the system could be attacked by a malicious client who transmit incorrect information to take advantage of the prioritization. Thus a pure in-network solution is more desirable.

III. In-network freeze detection/prediction

In this section, we detail our approach of detecting/predicting freeze for HAS video sessions from the perspective of an intermediate node. As freeze is the most influential factor to impact the end user's QoE, the network/CDN providers could use this knowledge to actively manage, monitor or enhance the QoE in their network in general. Macroscopically, aggregated HAS QoE metrics for different sessions provides a good view on the occupation of the network. For example, a low average viewing quality and frequent freezing occurences could be a sign of the neccesity of enlarging the bandwidth with certain network nodes for the operator. Microscopically, the network provider could faciliate the output of the online in-network freeze prediction: without the feedback from the client, the provider can decide if a prioritization is necceary assign higher priority traffic class when the play-out buffer at the client is close to an underrun, thereby actively preventing a likely rebuffering event for that HAS client.

A. Architectural description

Figure 1 illustrates the utility and position of the proposed network-based video freeze detection/prediction in the CDN loop. The network topology is composed of original HAS server, CDN and HAS client, together with a *priority controller* module [22] adhered to the CDN. This module behaves the interface for the task of freeze prediction. The CDN node maintains a log file tracking all segment-level HTTP requests from multiple clients' HAS video sessions. The log file is parsed through the *session log parsing* block, and is split into multiple session logs, each containing all HTTP requests for one single video session. The session separation is done by examining the unique MD5 of a session embedded in each of HTTP request. The feature extraction module is used to extract a few attributes either for a video session in the task of freeze detection, or for a segment request in the task of freeze prediction. In the training phase, the status report of the HAS client, called groundtruth, is required for both tasks, in order to train the detection/prediction classifers in a supervised mode. The status report includes, if applicable, the knowledge of the occurred user interactivies and the experienced freezes.

1) Session-level freeze detection: The session-level freeze detection aims at quickly providing moderately in-depth knowledge of freeze in a video session. The knowledge of freeze includes: (a) Does the concnerned video session contain any freeze? (b) Is the longest freeze, if exists, longer than a threshold, e.g. 10 seconds? (c) Are there multiple freezes in the session? The yes/no binary answers to the above three questions in fact are three binary classification tasks. Note that the answers give a general idea of the rebuffering, but do not reveal the details, e.g. the position and the exact

duration of freezes. On the other hand, the scalable expenditure of computational cost is another concerned issue. According to [4], around 70% of video sessions do not contain any freeze. Usually there are ten of thousands of concurrent video sessions delivered by a CDN node. Thus it is desired to quickly focus on problemic sessions. Therefore, in the freeze detection, we focus on session-level classification.

In the training step, all individual training sessions are tracked and separated from the CDN log. A feature vector is then extracted to represent the high level information of a single video session. In the meanwhile, the groundtruth, containing is collected from the corresponding client. The feature vectors and the groundtruth are coupled in the *training* module. Then three classifiers in terms of the above three questions are built. In the test, a testing feature vector for a testing video session is classified by the three binary classifiers. The combined recognized results represent the freeze status of the testing session.

2) Segment-level freeze prediction: The in-network segment-level freeze prediction is an online process. It attempts to forecast whether a freeze is approaching at the client when a request is received and the requested segment is about to deliver to the client from the CDN. If the prediction result gives an indication that the client is experiencing a freeze or will have a freeze shortly, the *priority controller*, as described in [22], will assign a priority-delivery note to this segment. The segment will be delivered with a higher priority, to rescue the client from freeze, or to avoid the freeze really happening.

In the training, the feature vector is extracted for each single segment request. The true label of a feature vector would be the event that happens at the client in between the time that the current and the next segment are about to be delivered. All feature vectors and their labels are trained in a supervised way to generate a prediction classifier. In the test phase, when the *priority controller* is going to deliver a segment to the client, it will inquiry the prediction classifier to check whether there is a high risk that the client is close to freeze. If the prediction result is no, then the segment is delivered as the usual way. Otherwise, the priority will be given to the segment.

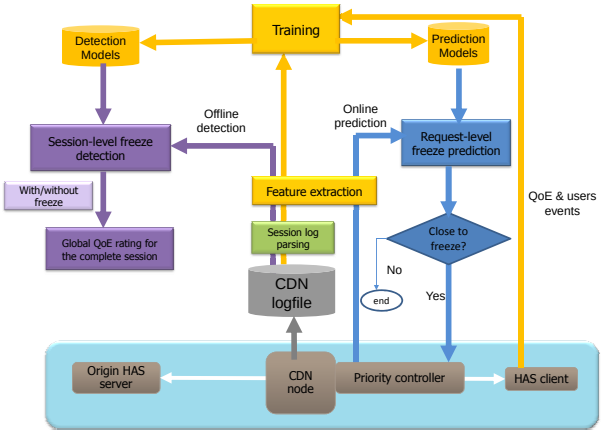

Fig. 1. Position of freeze detection/prediction in the CDN loop

IV. Methodology

A. Problem statement

From the machine learning point of view, the freeze detection/prediction is to look for a mapping function between the sequence of HTTP GET messages for a single video session from either an origin server or a CDN node with the occurrence of freezes at the client. A few disturbances may reduce the mapping accuracy.

- Segment retrieval. There is no guarantee that the client can successfully receive the segment in time. Due to certain reasons, for example traffic congestion, the client may not receive the segment, which is unknown for the network provider.
- Unknow status and heuristic-dependent parameters The sequence of HTTP GET requests does not reveal the instant status of the heuristic (such as the remaining buffer filling level and the perceived bandwidth) and the intrinsic heuristic-dependent parameters (including the segment length, the maximum buffer size, the initial startup delay, etc).
- User Interactivity. Some interactive operations taken by end-users could be difficult to be detected. For instance, in the video-on-demand (VOD) scenario, when the end-user pauses the playout, there are two different behaviors with repective to the segment requests: if the client is in the loading state (the state in which the client wants to fill the buffer as quickly as possible), it will keep asking new segments to fulfill the buffer, until the maximum buffer size is reached; but if the client is already in the steady state when pause , it does not send any requests. In this case, the network node does not receive any requests as well, and may conjecture that the client has already been experiencing a long freeze.

A sequence of HTTP GET messages for a video session s ($s \in \mathbf{S}$, where $\mathbf{S}$ is the set of video sessions) typically contain the timestamp $T(s) = \{t_i^s, 1 \leq i \leq N_s\}$ that the message is received and the requested quality level $Q(s) = \{q_i^s, 1 \leq i \leq N_s\}$, where N_s is the number of segments in the session s. The inter-segment duration I_i^s between segment i and segment $i+1$ is defined as $I_i^s = t_{i+1}^s - ti^s$. Then for a session s, the useful information related to freeze directly obtained from HTTP GET messages can be represented as $M(s) = \{T(s), Q(s)\}$.

In the training step, from the client groundtruth of session s, we know exactly the freeze related segment-level event happened between t_i^s and t_{i+1}^s. The event e_i^s, $(0 \leq e_i^s < I_i^s)$ is the duration of freeze in this period; if there is no freeze, then $e_i^s = 0$. The $e_i^s = 0$ indeed is the true label in the task of freeze prediction. Moreover, from $E(s) = \{e_i^s, 1 \leq i \leq N_s - 1\}$, we can obtain the session level labels $YF(s)$, $YL(s)$ and $YM(s)$ for session s, in which $YF(s)$ is the binary label for the existence of freeze, $YL(s)$ is the label for the exitence of long

freeze, and $YM(s)$ represents the multi-freeze information:

$$YF(s) = \begin{cases} 1 & \exists j, \quad e_j^s > 0 \\ 0 & otherwise \end{cases} \tag{1}$$

$$YL(s) = \begin{cases} 1 & \exists j, \quad e_j^s > F \\ 0 & otherwise \end{cases} \tag{2}$$

$$YM(s) = \begin{cases} 1 & \exists i, j, \quad i \neq j, \quad e_j^s > 0 \cap e_i^s > 0 \\ 0 & otherwise \end{cases} \tag{3}$$

F is an arbitrary threshold for long freeze, for instance, 10 seconds.

The task of freeze detection/prediction is to learn a function $g : \mathfrak{X} \to \mathfrak{Y}$. The instance $\mathbf{x} \in \mathfrak{X}$ is a feature vector extracted from the original sequence of HTTP GET requests. Depending on various tasks, the procedure of feature extraction is slightly different (see section IV-B).

1) Assumption: A few prerequisites are assumed in this study. First, as we are investigating video picture freeze, the requests of audio segments are ignored, unless otherwise stated. Apparently, the discontinous audio could impact the end user's QoE. But because the byte size of audio segments is usually much smaller than that of video segments (even the lowest quality level), the delivery is less likely to be the reason of audio discontinuity. Second, user interactivity behaviors are restricted to pause/play and forward/backward jump. Typically, there are six types of user interactions with a video player: play, pause, forward, rewind, jump and stop. Forward/rewind can be seens as a special case of jump, and stop is the end of video playout. So forward/rewind and stop are not treated seperately in the study of freeze detection/prediction with user interactivity. Third, we do not consider caching between the delivery node and the video player; all segment requests are received by the CDN/network node. Actually, the impact of caching in HAS is not well studied [23]. Intuitively, multiple quality levels for the same video segment in HAS reduces the efficiency of a cache. In practise, from the observation of *field dataset* (see section V-B1), the caching operation only happens very few times, if any.

B. Feature extraction and classifier selection

V. PERFORMANCE EVALUATION

A. experimental setup

1) session-based freeze detection:

2) request-based online freeze prediction:

B. Dataset

1) Field data set:

2) lab data set:

3) openflow data set:

C. discussion

VI. CONCLUSIONS

Subsection text here.

ACKNOWLEDGMENT

The research was performed partially within the iMinds VFORCE (Video: 4K Composition and Efficient streaming) project under IWT grant agreement no. 130655.

REFERENCES

[1] T. Stockhammer, "Dynamic adaptive streaming over http–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[2] J. De Vriendt, D. De Vleeschauwer, and D. Robinson, "Model for estimating qoe of video delivered using http adaptive streaming," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 1288–1293.

[3] R. Huysegems, B. De Vleeschauwer, K. De Schepper, C. Hawinkel, T. Wu, K. Laevens, and W. Van Leekwijck, "Session reconstruction for http adaptive streaming: laying the foundation for network-based qoe monitoring," in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*. IEEE Press, 2012, p. 15.

[4] Conviva, "2015 viewer experience report," http://www.conviva.com/vxr-home/, 2015, [Online; accessed Feb, 2015].

[5] T. Wu, R. Huysegems, and T. Bostoen, "Scalable network-based video-freeze detection for http adaptive streaming," to appear in IEEE/ACM International Symposium on Quality of Service, June 2015.

[6] L. Yitong, S. Yun, M. Yinian, L. Jing, L. Qi, and Y. Dacheng, "A study on quality of experience for adaptive streaming service," in *Communications Workshops (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 682–686.

[7] J. Yao, S. S. Kanhere, I. Hossain, and M. Hassan, "Empirical evaluation of http adaptive streaming under vehicular mobility," in *NETWORKING 2011*. Springer, 2011, pp. 92–105.

[8] J. Lievens, A. Munteanu, D. De Vleeschauwer, and W. Van Leekwijck, "Perceptual video quality assessment in http adaptive streaming," in *Consumer Electronics (ICCE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 72–73.

[9] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *Communications Surveys and Tutorials, IEEE*, 2015.

[10] M. Garcia, F. De Simone, S. Tavakoli, N. Staelens, S. Egger, K. Brunnström, and A. Raake, "Quality of experience and http adaptive streaming: A review of subjective studies," *Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on*, pp. 141–146, 2014.

[11] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *Communications Surveys and Tutorials, IEEE*, vol. 17, pp. 469–592, Sep. 2014.

[12] A. Sackl and R. Schatz, "Evaluating the impact of expectations on end-user quality perception," in *Proceedings of International Workshop Perceptual Quality of Systems (PQS)*, 2013, pp. 122–128.

[13] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 271–287, 2012.

[14] H. Riiser, H. S. Bergsaker, P. Vigmostad, P. Halvorsen, and C. Griwodz, "A comparison of quality scheduling in commercial adaptive HTTP streaming solutions on a 3G network," in *Proceedings of the 4th Workshop on Mobile Video*. ACM, 2012, pp. 25–30.

[15] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design and optimisation of a (fa) q-learning-based http adaptive streaming client," *Connection Science*, vol. 26, no. 1, pp. 25–43, 2014.

[16] T. Wu and W. Van Leekwijck, "Factor selection for reinforcement learning in http adaptive streaming," in *MultiMedia Modeling*. Springer, 2014, pp. 553–567.

[17] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 169–174.

[18] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*. ACM, 2013, pp. 9–14.

[19] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "Using the buffer to avoid rebuffers: Evidence from a large video streaming service," *arXiv preprint arXiv:1401.2209*, 2014.

[20] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. Vleeschauwer, W. Leekwijck, and F. Turck, "Qoe optimization through in-network quality adaptation for HTTP adaptive streaming," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 2012, pp. 336–342.
[21] N. Bouten, R. d. O. Schmidt, J. Famaey, S. Latré, A. Pras, and F. De Turck, "Qoe-driven in-network optimization for adaptive video streaming based on packet sampling measurements," *Computer networks*, vol. 81, pp. 96–115, 2015.
[22] S. Petrangeli, T. Tim Wauters, R. Huysegems, T. Bostoen, and F. De Turck, "Network-based dynamic prioritization of http adaptive streams to avoid video freezes," to appear in QCMAN 2015 - Third IFIP/IEEE International Workshop on Quality of Experience Centric Management, May 2015.
[23] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in http adaptive streaming: Friend or foe?" in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. ACM, 2014, pp. 31–36.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.