# Stalling Assessment for Wireless Online Video Streams via ISP Traffic Monitoring

Hui Tang[1], Liang Chen[1], Yue Wang [2], Na Wang[1], Xia Li[1,3]
[1]College of Information Engineering, Shenzhen University
[2]Department of Information Management, Central University of Finance and Economics
[3]The Chinese University of Hong Kong, Shenzhen
Email: tanghui1@email.szu.edu.cn[1], lchen@szu.edu.cn[1], yuelwang@163.com[2], wangna@szu.edu.cn[1], lixia@cuhk.edu.cn[3]

***Abstract*—[1] Nowadays, ISPs witness an increasing network traffic generated by wireless online video services. Facing the intense mutual competition, it is important for an ISP to improve QoE of wireless video services. Since it is difficult for ISPs to collect user experience information at routers and switches, we propose an approach based on ISP traffic monitoring to access the stalling of a video stream, which is the key factor to video QoE. Concretely, we passively sniff out the following parameters to reconstruct video playing and stalling: playback rate of a slice, start-up delay and playback resuming threshold. Our difference from previous works is that we need not intrude into packets to extract HTTP meta information for these parameters. Moreover, we develop a video stalling assessment system based on Apache Storm, which can process the network traffic data in real time. Experiments on popular online video services (Tencent Video, iQiyi and Youku) in China as well as Youtube in US show not bad absolute difference of stalling times and total stalling duration between our traffic monitoring approach and real measurement.**

***Index Terms*—video stalling; wireless online video service; traffic monitoring; DASH.**

## I. Introduction

Mobile Internet enables users to access online services via WiFi and 4G networks. Currently, users are shifting from PCs to mobile devices, and there are over 70% data traffic contributed by wireless online video services [1]. Both Internet Service Providers (ISPs) and Media Service Providers (MSPs) are facing a crucial challenge of maintaining satisfactory Quality of Experience (QoE) for users. There are many metrics to assess the QoE for MSPs including subjective metrics such as Mean Opinion Score (MOS) the objective metrics such as video quality or network quality (e.g. network delay, jitter and loss). Video stalling is usually the most direct way to evaluate QoE performance and it impacts users' QoE significantly.

Due to volatile wireless conditions and many users competing limited bandwidth, users may suffer bad network quality and get poor QoE when watching videos. As a result, unsatisfied users often blame bad service of ISPs and ISPs will lose users eventually. If ISPs take early actions (e.g., adding bandwidth, replacing hardware, improving wireless environment) to provide better Quality of Service (QoS) before obvious video stalling occurs, it can win over users. However, it is inconvenient and costly for ISPs to get feedback of QoE from end-clients or MSPs for each network it maintains. Furthermore, ISPs are forbidden to intrude into packets to extract HTTP meta information for privacy issues.

Nowadays, almost all MSPs have turned to popular *HTTP-based streaming*. Under this video transportation technique, a video is cut into multiple small files called *slices*. A slice is a video fragment that plays a few seconds to several minutes. Before a client downloads a video, it gets from the server a list that records the IP addresses of all slices of that video (different slices can be located in different servers), then the client download the slices one by one by sending HTTP requests and the underlying transport protocol is TCP instead of RTSP (Real Time Streaming Protocol), a video transport protocol proposed before but got little implementation. This is the so-called HTTP-based streaming. The main reason for its being popular is its simplicity: First, it is the client rather than the server to control when to download the next slice; Second, a slice can have multiple versions for different definitions and the client can request the next downloading slice of a certain definition based on the current network condition - this technique is called *DASH* (*Dynamic Adaptive Streaming over HTTP*) [2], which is applied by Youtube in US; Third, the use of TCP and of HTTP over TCP in particular, greatly simplifies the traversal of firewalls and NATs (Network Address Translation) and can utilize CDNs (Content Distribution Networks) without modification.

In this work, we focus on assessment of video stalling from ISP traffic monitoring. Because only arrival traffic can be monitored by ISPs, the main challenge is how to reconstruct the video playback process *by ISPs* and *via traffic monitoring only* in order to assess video stalling. Specifically, we need to answer the following questions: (1) How to determine the boundary of a video slice and its *definition* (equivalently, *playback bit rate*)? (2) How to determine a video's start-up delay due to advertisement? And How to determine the resuming buffer threshold after a stalling occurs? (3) After the playback process is reconstructed from (1) and (2), how

[1]This work is supported by the Natural Science Foundation of China (No. 61502315, No. 61309030), the Natural Science Foundation of Guangdong province (No. 2015A030310366), the Technological Innovation of Shenzhen city (No. JCYJ20160422112909302) and the Fundamental Research Funds for the Shenzhen University (201558). It was partially supported by Tencent "Rhinoceros Birds" - Scientific Research Foundation for Young Teachers of Shenzhen University, and also granted by Discipline Construction Foundation of Central University of Finance and Economics as well as Graduate Educational and Teaching Reform Research Project of Central University of Finance and Economics. Liang Chen is the corresponding author (lchen@szu.edu.cn).

to derive the stalling situation in real time?

In this work, we will show that the *sound* playback process can be reconstructed through measuring and learning the playback characteristics of different MSPs *prior*. We validate our method on popular online video service (Tencent Video, iQiyi and Youku) in China as well as Youtube in US. In contrast to previous work, we can estimate without intruding the HTTP content (and it is hard to decode in reality) and we find a way to deal with DASH. As a result, we develop the stalling assessment system based on Apache Storm [3], which can process the video traffic data in real time.

This work presents a new way for video stalling assessment by ISP traffic monitoring and we plan to further elaborate on it in the future. The rest of the paper is organized as follow. Section II reviews the related work. Section III introduces the background. Section IV presents our solutions. Section V shows the experiment results. Section VI concludes this paper and discuss the future work.

## II. Related Work

Theoretically, the analysis of the video stalling is based on the busy period in transient queues. Some existing work analyzed the playback buffer for different queue models. In [4], Baccelli et al. proposed modifying Bessel functions to solve the transient distribution of queue length for a M/M/1 queue. In [5], Gurewitz et al. used the powerful Ballot theorem to compute the probability of the number of packets arrived for M/M/1 queue. Authors in [6] [7] calculated the packet loss probability for M/M/1 queue in terms of the recursive approach. Furthermore, Xu et al. [7] proposed using the fluid model to analyze the probability of video stalling on the file level. Luan et al. in [8] analyzed the probability of video stalling considering video size and playback buffer size, modeling the system as a G/G/1 queue. Based on [8], Anttonen et al. [9] provided new inference on how the video length and delay parameters affect the probability of video stalling. However, we assume the playback rate is constant in real scenarios, as opposed to their assumption.

In [10], Parandehgheibi et al. deduced the probability of video stalling for different sizes of playback buffer. They assumed that the packets arrival follows Poisson distribution and packets departure rate is constant, and the playback buffer is modeled as a M/D/1 queue. Yu et al. formulated the playback buffer as a G/D/1 queue and analyzed it by the diffusion approximation method in [11]. Moreover, they calculated the probability of video stalling for different video definitions and average packet arrival rates. However, they do not provide the insights of the exact number and total time of stalling for a video.

Schatz et al. [12] took a different approach of reconstructing video playback via monitoring network traffic and extracting the HTTP meta information from packets and URLs. We follow this method since network traffic is volatile and dynamic over time and can not be modeled accurately by a queuing model. Considering the protection of users privacy, we try to estimate the video playback parameters from TCP and IP headers of video packets, instead of intruding HTTP header.

## III. Background

In this work, we study HTTP-based online video streaming, including constant and dynamic playback rate adaptation.

### A. Video definition, start-up delay and slicing

The *definitions* of wireless online video mainly are: low-standard (LSD), standard (SD), medium(MD), high (HD) and very-high definitions (VHD). In the following context, we often denote a video definition by its *playback rate*. MSPs usually adopt Variable Byte Rate (VBR) to encode a video, resulting in dynamic size of per GOP (Group of Pictures).

There is a *start-up delay* before a video is played. During the delay, an advertisement can be downloaded and played, which is a major income for MSPs. The start-up delay is closely related to video stalling since it determines the initial length of playback buffer.

During transmission, a video is cut into several *slices* when its length exceeds a certain threshold. A slice is a video fragment of several seconds to several-minutes, whose length depends on MSPs and which is transmitted by a TCP connection. In *DASH* (*Dynamic Adaptive Streaming over HTTP*), a slice has multiple versions of definitions and the client decides to request which definition depending on current network conditions, that is, the downloaded slices one by one can have different definitions. Besides, our measurement reveals that two adjacent video slices can be transmitted by different servers.

### B. Playback and stalling

We treat the playback buffer as a queue, shown in Fig. 1, where packets arrival to the buffer follows TCP and network conditions and the service rate is equal to the playback rate of the currently used video definition. The video stalling occurs when the playback buffer is empty, and video playback resumes until bytes accumulated in the buffer exceed a certain threshold, denoted by $\theta$. MSPs set an appropriate $\theta$ to balance between frequent video stallings and long waiting time during video stalling.

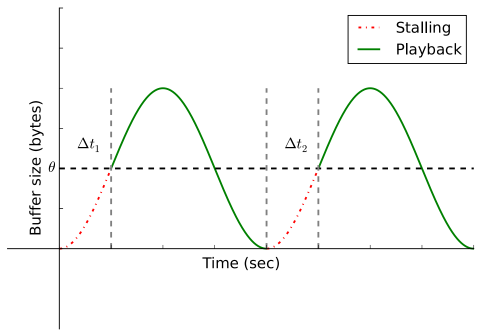

Fig. 1. Illustration of playback buffer changing during playback.

## IV. Solution

### A. Measured Characteristics of Online Video Services

MSPs usually adopt HTML to control the actions of video players in mobile devices. Selenium [13] records the corresponding timestamps of video playback by monitoring the changes of labels in clients' HTML pages. We use Selenium to watch online videos and record various timestamps of video playback (i.e. start timestamp, end timestamp, stalling timestamp, playback resuming timestamp) automatically, combined with the timestamps of packets arrived at the client (measured by Wireshark [14]) to get the characteristics of video services. The measured characteristic parameters of MSPs are later used in stalling assessment.

*1) Video playback rate:* Table I shows our measured video playback rates of three main definitions[2]. We find that the video playback rate of the same definition depends on MSPs and it is not a constant.

TABLE I
Video playback rates of three main definitions of each MSP

| MSP | SD | MD | HD |
|---|---|---|---|
| Tencent Video | 34 ∼ 39KB/s | 110 ∼ 120KB/s | 195 ∼ 200KB/s |
| iQiyi | 44 ∼ 52KB/s | 76 ∼ 84KB/s | 188 ∼ 200KB/s |
| Youku | 50 ∼ 53KB/s | 112 ∼ 120KB/s | 195 ∼ 200KB/s |
| Youtube | 42 ∼ 50KB/s | 100 ∼ 120KB/s | 188 ∼ 200KB/s |

*2) Video slicing:* The video *slicing thresholds* and *slice durations* (i.e., *the playback duration of a slice*) are shown in Table II. Our measurement shows that a video is partitioned into slices when its playback duration is larger than a slicing threshold. For the same MSP, the duration of a slice is almost in the same range independent of video definitions.

TABLE II
Slicing thresholds and durations

| MSP | Slicing threshold | Slice duration |
|---|---|---|
| Tencent Video | 10min | 4 ∼ 5min |
| iQiyi | 9min | 6 ∼ 7min |
| Youku | 7min | 4 ∼ 7min |
| Youtube | 10sec | 10 ∼ 15sec |

*3) Playback resuming buffer threshold:* The playback resuming buffer thresholds tested by us are shown in Table III, which are independent of video definitions.

### B. Estimation of Start-up Delay of Video

An advertisement may be downloaded and played before a video playback, and the video begins downloading during the playback of the advertisement. Usually, the server transmitting an advertisement is different from the one transmitting the requested video. The video's start-up delay is the duration of the video's playback start timestamp minus the video's downloading start timestamp. The start-up delay determines the initial buffer size of the video and it is closely related to stalling.

[2]Youtube has another SD level of 60∼78 KB/s and a LSD of 28∼35 KB/s.

TABLE III
Playback resuming buffer thresholds

| MSP | Resuming buffer threshold |
|---|---|
| Tencent Video | 5 ∼ 6sec |
| iQiyi | 4 ∼ 5sec |
| Youku | 2 ∼ 3sec |
| Youtube | 4 ∼ 6sec |

We now estimate the start-up delay. Note that the video's playback start timestamp is equal to the advertisement's downloading start timestamp plus the advertisement's playback duration. First, the advertisement's downloading start timestamp can be obtained from the first packet of the advertisement. Second, the advertisement's playback duration is the advertisement's total bytes divided by its playback rate. The advertisement playback rate is a fixed value determined by the MSP, which is 35-40KB/s. Third, the video's downloading start timestamp can be obtained from the first packet of the video. With the above parameters estimated, we can calculate the start-up delay of the video.

### C. Estimation of Video Playback Rate: Fixed Case

A fixed playback rate is that the playback rate of a video is fixed throughout the entire playback duration (unless users switch to another playback rate), which is specified by users or by default. In our testing MSPs, Tencent Video, iQiyi and Youku offer fixed playback rate. A video is cut into multiple slices only when it exceeds a given threshold, so we need to consider two cases:

*1) A video of multiple slices:* Each slice is initiated by an HTTP request and setting up a TCP connection. By receiving HTTP request packets or TCP initialization packets, we can know the boundary of a slice and get its byte size via monitoring packet arrivals when downloading the slice. We can compute the duration of the first slice by dividing its byte size by the playback rate of each definition (see Table I), respectively. After we get a candidate slice duration for each definition, we compare them with the sound range given the video's MSP is known[3] (see Table II). And we determine the final slice duration that is nearest to the range. Finally, we get the playback rate by dividing the slice byte size by its slice duration.

*2) A video of one slice:* There is only one slice when the playback duration of a video is less than the slice threshold. In here, only one HTTP request and TCP initialization are monitored by us. We propose another method to determine the video playback rate because the duration of the slice is

[3]By collecting IP addresses under the domain name of a MSP, we can establish an IP database to relate the IP address of a media server to the name of its MSP.

below the slice threshold. The average downloading rate is dividing the total bytes downloaded by the downloading time. Referring to Table I, we choose the largest playback rate below the average downloading rate as the estimated playback rate. The reasons are as follows. First, severe stalling will occur if the playback rate is larger than the average downloading rate. And authors in [15] showed that more than 70% users can not endure stalling duration of more than 30 seconds. Second, MSPs can initialize the appropriate definition of a video according to current network condition. Thus, we choose the largest playback rate that can be supported by the average downloading rate.

## *D. Estimation of Video Playback Rate: DASH Case*

The basic principle of DASH is that playback rate of a slice requested by a client is adjusted according to network downloading rate. A example of MSPs using DASH is Youtube in US. In computer networks, rate is often smoothed by *exponential moving average (EMA)* to eliminate short-term variations, which is very necessary in here because the momentary rate of TCP itself is highly dynamic, needless to say that available bandwidth of a video flow varies all the time [16].

We use $r(t)$ to denote the *instantaneous downloading rate* at the $t$th *sampling window* and $\bar{r}(t)$ the *recent average downloading rate*, calculated by the EMA of $r(t)$:

$$\bar{r}(t) = \alpha \cdot r(t) + (1 - \alpha) \cdot \bar{r}(t-1), \tag{1}$$

where $\alpha$ is the weight of the instantaneous rate, too large causing high variability of playback rate and too small causing slow reaction.

We can not access Youtube directly due to administration issues in China. Using proxies to access it proposes another challenge that we can not find HTTP requests or TCP initialization packets in the traffic from proxy servers because they request slices instead of clients. Fortunately, we can utilize the rate adaptation mechanism, consequently leading to another way to estimate dynamic playback rates in DASH.

The challenge is that DASH algorithms developed by different companies are confidential to the public, so we have to rely on models to predict playback rates and then assess stalling. In the following, we assume a basic model of DASH: (a) Playback rate varies for every slice but it is constant within a slice; (b) The DASH algorithm estimates instantaneous downloading rate in every sampling window and calculate the recent average downloading rate by EMA; (c) The playback rate of the next slice requested is the maximum sustainable playback rate just smaller than the recent average downloading rate.

In the following, we show the three key parameters estimated from trace data:

*1) Slice duration:* We can infer slice duration when the observed (by Selenium) playback rate changes, which is often between 10 and 15 seconds. And for simplicity we treat it like a constant. By our parameter estimation method described below, the best-fit slice duration is 10 seconds.

*2) Sampling window size $\tau$:* By our parameter estimation method described below, the best-fit sampling window size is *1 second* and the performance is not sensitive to it.

*3) EMA weight $\alpha$:* By our parameter estimation method described below, the best-fit $\alpha$ is *0.1*.

*4) Best-fit parameter estimation:* We use the optimal approximation to estimate the above parameters. Simply speaking, we generate for a video a sequence of playback rate adjustment for each trial of parameter sets and calculate the *distance* to the realistic sequence observed by Selenium. Thus, we can find the optimal parameter set when the distance is minimized. Specifically, suppose the real playback rate is $p(t)$ and our simulated playback rate is $\hat{p}(t)$ in the $t$th sampling window ($n$ sampling times in total), then their distance is defined to be $\sum_{i=1}^{n} |d(t) - \hat{d}(t)|/n$. Finally, the optimal parameter set is $\tau = 1$s and $\alpha = 0.1$. Fig. 2 shows simulated versus real playback of a video streaming using the optimal parameter set.

Once the best-fit parameters are got, they are fixed and applied to stalling assessment. And our test showed that they are insensitive to video definitions.

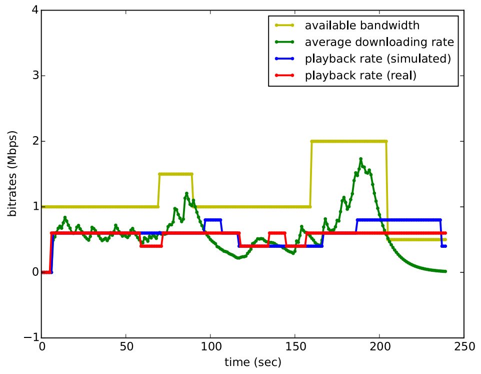

Fig. 2. Demonstration of a video using DASH.

*5) Estimating slice playback rate:* Note that the estimated playback rate of a slice should be aligned to the time when the slice plays, so that we can observe the corresponding real playback rate of a slice from Selenium, as illustrated in Fig. 3. The playback rate of a slice is determined by the recent average downloading rate via EMA prior to its downloading. Knowing playback rate and slice duration (assuming 10 seconds fixed), we can get its byte size. From the byte size and real measured traffic arrival, we can get the downloading time of each slice, shown in the first row in this figure. Then we can easily assign a slice downloading to its playback in the timeline.

## *E. Estimation of Video Stalling*

A video enters into stalling when the playback buffer is empty, and it resumes when the playback buffer reaches a certain playback resuming threshold (see Table V). By queue theory and [9], we define the bytes in the playback buffer as $Q(t)$, the total bytes of accumulated arrival packets as $A(t)$ and the total bytes of accumulated departure packets as $D(t)$ at $t$ seconds. $a(i)$ and $d(i)$ at the end of the $i$th time slot denote

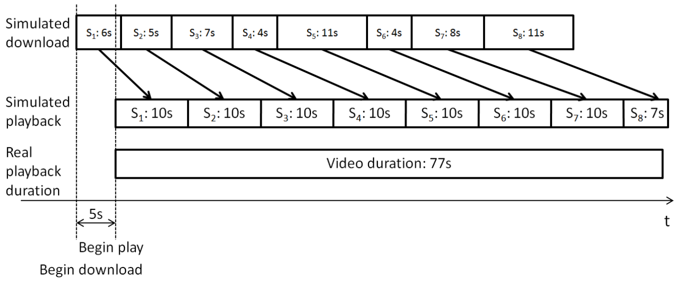

Fig. 3. Illustration of DASH playback estimation.

the bytes of arrival packets and departure packets, respectively. Besides, we define the video state at the $i$th time slot as $\phi(i)$, which is playing ($\phi(i) = 1$) or stalling ($\phi(i) = 0$). We have,

$$Q(t) = A(t) - D(t), \tag{2}$$

$$A(t) = \sum_{i=1}^{t} a(i), \tag{3}$$

$$D(t) = \sum_{i=k}^{t} d(i). \tag{4}$$

$k$ denotes that video begins playing at $k$th time slot.

We can obtain $a(i)$ by traffic monitoring and determine $d(i)$ following two cases.

$$d(i) = \begin{cases} 0, & if\ \phi(i) = 0 \\ v, & if\ \phi(i) = 1 \end{cases} \tag{5}$$

Where $v$ is the playback rate per time slot. The video state at the $i$th time slot $\phi(i)$ depends on $\phi(i-1)$ and $Q(i-1)$ according to the playback and stalling process (in Section III).

$$\phi(i) = 0, if \begin{cases} \phi(i-1) = 1 \wedge Q(i-1) = 0, or \\ \phi(i-1) = 0 \wedge Q(i-1) < \theta \end{cases} \tag{6}$$

$$\phi(i) = 1, if \begin{cases} \phi(i-1) = 1 \wedge Q(i-1) > 0, or \\ \phi(i-1) = 0 \wedge Q(i-1) \geq \theta \end{cases} \tag{7}$$

From $\phi(i)(i = 1, 2, 3, ..., t)$, we can calculate the total time slots of stalling as well as the number of stalling periods.

## V. Performance Evaluation

### A. Stalling Assessment System Implementation

We use Wireshark [14] to capture video traffic in a wireless network. The traffic is then sent to Apache Storm for processing and stalling detection. Since there may be hundreds to thousands of video streams flowing through the network and the traffic needs to be process in real time and can not be stored locally, we use multiple machines installing Apache Storm which is a distributed real-time computation system, each processing a portion of traffic.

The traffic monitoring process is described as follows. First, when a user makes a request for a video, a connection between a media server and the user is established. As all IP addresses of media servers of any MSP are collected and stored in a database beforehand, Storm can determine the MSP where the video comes from. By the IP address and port number, Storm can identify and traces a video stream established between the server and the client. Second, our system reconstructs the video's playback and stalling based on traffic arrival and estimated playback rates as well as start-up delays and playback resuming buffer thresholds according to the methods in Section IV. Note that the slice playback rate can be reconstructed in real time for either fixed or DASH cases (see Section IV-C-IV-D) as well as the stalling simulation (see Section IV-E). Finally, the system counts the times and total duration of stalling for each video stream it monitors and saves the results into a database.

### B. Experiment Results

Fig. 4 illustrates our reconstruction of a video playback. In the case of no stalling, the downloaded bytes number is far above from the played one. In the case of stalling, the played byte number stop increasing when the stalling occurs and it resumes when the video comes out of stalling.

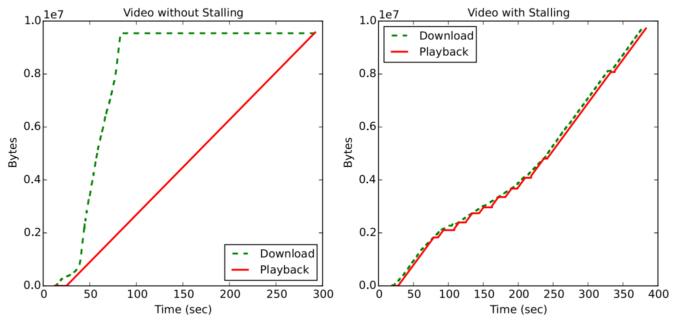

Fig. 4. Illustration of the reconstruction of video playback. The dashed line denotes the amount of downloaded bytes of a video and the solid line denotes the played bytes of the video.

We define the absolute difference $\Delta_{abs} = |v_{est} - v_{real}|$ and the relative difference $\Delta_{rel} = |v_{est} - v_{real}|/v_{real}$, where $v_{est}$ and $v_{real}$ are our estimated and real-measured values, respectively. The value can be stalling times or total stalling duration of videos.

We tested a number of online videos. Fig. 5 shows the absolute and relative differences between stalling times captured by Selenium (real measurement) and estimated by our traffic monitoring method, and Fig. 6 shows the case of total stalling duration. First, we evaluate stalling times. For fixed playback rate (Tencent Video, iQiyi and Youku), 90% of videos shows the absolute difference below one stalling, and 80% of videos shows the relative difference below 25%. For DASH playback rate (Youtube), 70% of videos shows the absolute difference below one stalling, and 70% of videos shows the relative difference below 35%. Next, we evaluate total stalling duration. For fixed playback rate, 85% of videos shows the absolute difference below 1 second, and 80% of videos shows the relative difference below 20%. For DASH playback rate, 70% of videos shows the absolute difference below 4 seconds, and 75% of videos shows the relative difference below 40%.

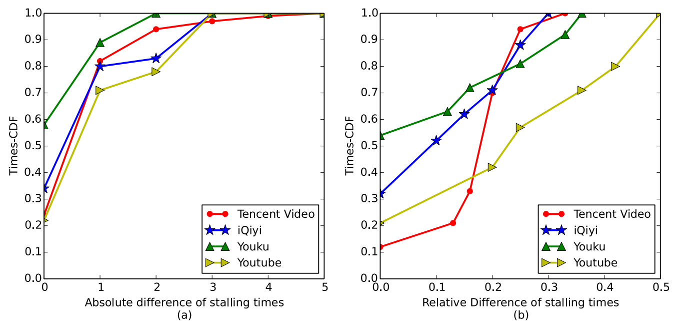

Fig. 5. Absolute and relative differences between stalling times captured by Selenium (real measurement) and estimated by our traffic monitoring method.

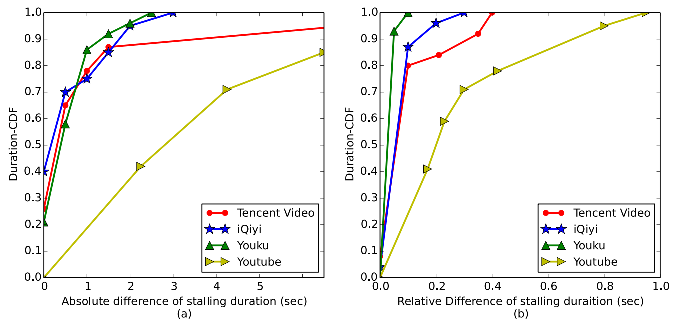

Fig. 6. Absolute and relative differences between total stalling duration captured by Selenium (real measurement) and estimated by our traffic monitoring method.

Fig. 7 shows the correlation between stalling times captured by Selenium (real measurement) and estimated by our traffic monitoring method as well as the case of total stalling duration.

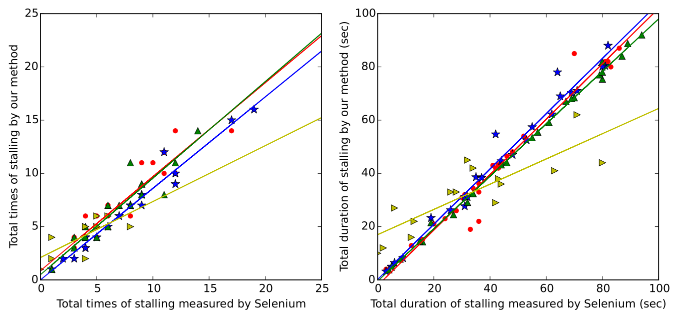

Fig. 7. Correlation between stalling times (left) and total stalling duration (right) for the videos captured by Selenium (real measurement) and estimated by our traffic monitoring method. A point denotes a video of a MSP (represented by a certain point type) and a line is the trend line of a MSP's videos.

## VI. Conclusion and Future Work

In this work, we assess stalling of wireless online video from the ISPs' perspective. The main challenge is to accurately reconstruct the situation of video playback by using network traffic data only. Due to privacy issues, we can not extract the HTTP meta information from packets and the playback parameters need to be measured from traffic arrivals. We provide the solutions of measuring the playback parameters which are video playback rate, start-up delay and video stalling. Based on them, we can detect the number and total duration of video stallings in real time. We develop a video stalling assessment system based on Apache Storm which can process the data in real time. The results provide a good reference of video stalling situation to warn ISPs to update wireless networks to accommodate increasing video demand. In the future, we plan to increase our prediction accuracy further, especially for DASH.

## References

[1] C. V. N. I. Cisco, "Global mobile data traffic forecast update, 2013–2018," *white paper*, 2014.

[2] 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.

[3] *Apache Storm*, http://storm.apache.org/.

[4] F. Baccelli and W. A. Massey, "A sample path analysis of the m/m/1 queue," *J.appl.probab*, vol. 31, no. 2, pp. 59–93, 1999.

[5] O. Gurewitz, M. Siki, and S. Cidon, "The ballot theorem strikes again: packet loss process distribution," *IEEE Transactions on Information Theory*, vol. 46, no. 7, pp. 2588–2595, 2000.

[6] I. Cidon, A. Khamisy, and M. Sidi, "Analysis of packet loss processes in high-speed networks," *IEEE Transactions on Information Theory*, vol. 39, no. 1, pp. 98–108, 1993.

[7] Y. Xu, E. Altman, R. El-Azouzi, M. Haddad, S. Elayoubi, and T. Jimenez, "Probabilistic analysis of buffer starvation in markovian queues," in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 1826–1834.

[8] T. H. Luan, L. X. Cai, and X. Shen, "Impact of network dynamics on user's video quality: Analytical framework and qos provision," *Multimedia IEEE Transactions on*, vol. 12, no. 1, pp. 64–78, 2010.

[9] A. Anttonen and A. Mammela, "Interruption probability of wireless video streaming with limited video lengths," *IEEE Transactions on Multimedia*, vol. 16, no. 4, pp. 1176–1180, 2014.

[10] A. Parandehgheibi, M. Medard, A. Ozdaglar, and S. Shakkottai, "Avoiding interruptions a qoe reliability function for streaming media applications," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1064–1074, 2011.

[11] F. Yu, H. Chen, L. Xie, and J. Li, "Impact of end-user playout buffer dynamics on http progressive video qoe in wireless networks," in *IEEE International Symposium on Personal, Indoor, and Mobile Radio Communication*, 2014, pp. A–410.

[12] R. Schatz, T. Hobfeld, and P. Casas, "Passive youtube qoe monitoring for isps," in *Best Paper Award at Workshop on Future Internet and Next Generation Networks (FINGNet-2012)*, 2012, pp. 358–364.

[13] *Selenium*, http://docs.seleniumhq.org/.

[14] *wireshark*, https://www.wireshark.org/.

[15] T. D. Pessemier, K. D. Moor, W. Joseph, and L. D. Marez, "Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching," *IEEE Transactions on Broadcasting*, vol. 59, no. 1, pp. 47–61, 2013.

[16] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 157–168.