AARHUS UNIVERSITY

Coversheet

This is the accepted manuscript (post-print version) of the article.

Contentwise, the post-print version is identical to the final published version, but there may be differences in typography and layout.

How to cite this publication

Please cite the final published version:

Hiranandani, B., Sarkar, M., Mihovska, A., & Das, A. (2018, 17-20 April 2018). *User throughput-based quality of experience evaluation for in flight video streaming*. Paper presented at the 2018 Wireless Telecommunications Symposium (WTS).

Publication metadata

Title:	<i>User throughput-based quality of experience evaluation for in flight video streaming</i>
Author(s):	Hiranandani, B., Sarkar, M., Mihovska, A., & Das, A.
Journal:	2018 Wireless Telecommunications Symposium (WTS)
DOI/Link:	10.1109/WTS.2018.8363950
Document version:	Accepted manuscript (post-print)

General Rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

User Throughput-Based Quality of Experience Evaluation for In Flight Video Streaming

Bhaves Hiranandani, Mahasweta Sarkar

Department of Electrical and Computer Engineering,

San Diego State University, San Diego, California, USA

Email: bhiranandani-w@sdsu.edu, msarkar2@mail.sdsu.edu

Albena Mihovska

BTECH, Aarhus University

Herning, Denmark

Email: amihovska@btech.au.dk

Aniruddha Das

Viasat Inc.

California, USA

Email: nandan.das@viasat.com

Abstract—Satisfaction of end users is one of the most important factors for every service provider. Given that a lot of factors along with the network conditions affect the user's experience, the network quality given by the Quality of Service (QoS) parameters alone cannot provide an accurate measure of the users Quality of Experience (QoE). Today, most of the airlines have started providing in-flight wi-fi services, which allow the passengers to use various Internet services while on-board the flight, such as send and receive e-mails, stream video from diverse online service providers, and so forth. Statistics show that more than 50% of the passengers use the provided wi-fi service to stream video, therefore, their perception of the video service will be determining for the service provider's performance. One easy way to evaluate the perceived video streaming (i.e. QoE) is by estimating the frequency of stalls. In our paper, we propose an evaluation method based on the user throughput values to determine the number of stalls the user may have experienced during an adaptive video streaming service.

Our algorithm is not dependent on any user device property and requires only one input, namely, the throughput values delivered to the user device, which can be collected at either the router or if available, at the traffic shaper.

Index Terms—QoS, QoE, stall, throughput, adaptive video streaming

I. INTRODUCTION

It is very important for any Internet Service Provider (ISP) to provide a satisfactory Quality of Experience (QoE) to their users [1]. ISP can measure the QoE of their users either by subjective metrics such as Mean Opinion Score (MOS) or by objective metrics such as network quality given by network delay, bandwidth. In the case of video streaming, video stalls (i.e. the number of interruptions during the streaming of the video) impact the QoE of the users significantly.

Almost all of the major Media Service Providers (MSP) have now started using HTTP adaptive video streaming. The main idea behind the adaptive video streaming is to adapt the bandwidth required by the video stream to the throughput available over the network path from source to client [2].

In this technique, the MSP store every video in multiple small files called slice and each slice is encoded in multiple quality levels and thus vary in bit-rate. Based on the estimation of the available throughput, a client might request the subsequent slice of different video quality to avoid any video interruption and to cope with varying network conditions. The main challenge for a client is to decide the optimal quality for each of the slice required to optimize the viewing experience.

The client faces the following challenges: 1) It must closely and continuously estimate the available throughput, 2) it must control the local buffer level to avoid any underflows, which will result in a playback interruption or any overflow which will result in loss of data, 3) it must minimize any unnecessary quality fluctuations during the playback and 4) it must minimize the delay between any user's request and start of playback [2]. This allows for the client to have full control on the next slice download and on the desired quality level.

A user of adaptive video streaming might face fluctuations in the video quality due to poor network conditions. These fluctuations are a consequence of the unavailability of video data in the user device buffer. Hence, one way to estimate the stalls at the user is to analyze their local buffer status. But in most cases, the ISP's do not have access to their user's buffer and hence it becomes difficult to estimate the QoE of the users. These fluctuations at the user device can also be seen as an effect of a long-time degradation in the throughput values delivered to the user device. Because the video that is being played at the user device directly depends on the throughput values delivered to the device, the throughput values can be used to estimate any fluctuations or stall that may have occurred at the user device.

This paper proposes a novel algorithm that replicate the behavior of a user device while streaming a video and allows for measuring the number of stalls the user would have experienced. The algorithm takes as an input only the throughput values delivered to the client and tries to estimate the buffer status of the user device and any video quality degradation or stall that would have occurred at the user device. This paper is further organized as follows. Section II analyzes the state of the art in the area of evaluating the QoE during video streaming. Section III describes the proposed algorithm. Section IV shows how the algorithm has been implemented stepwise. Section V evaluates the performance and discusses the results. Section VI concludes the paper and gives incentives for future work.

II. RELATED WORK

To estimate the video stalls, many authors have analyzed the playback buffer for different queue models. In [3], authors derived explicit expressions for the probability distribution of the number of lost packets within a block of consecutive

packet arrivals into a finite buffer using various versions of the powerful Ballot Theorem for a single source M/M/1/K queue. The authors in [4] have derived the probability of the video stall for various sizes of playback buffer where they modeled the playback buffer as M/D/1 queue. In their work, they assumed that the arrival of the packets follows a Poisson distribution, while the departure rate of packets is always constant. Authors in [5] formulated the playback buffer as G/D/1 queue and analyzed it using the diffusion approximation method. They also calculated the probability of video stall for different video qualities and an average rate of packet arrival. Authors in [6] used the recursive approach to calculate the packet loss probability for M/M/1 queues. Further, they used the fluid model to estimate the probability of video stalls at the file level. In [7] authors have presented an expression for the analysis of the interruption probability that incorporates the effect of finite video lengths in the G/G/1 buffer model.

Other authors have tried to first reconstruct the video playback buffer and then analyze it to estimate any stall. In [8], the authors adopted an approach, in which they reconstructed the video playback by monitoring the network traffic flow extracted from the HTTP meta information from the video packets. The authors in [9] followed a similar approach and tried to reconstruct the video playback parameters by analyzing the TCP and IP headers of video packets instead of the HTTP headers.

Although the above approaches provide a good measure of stalls, they are very complex methods since we first need to extract the information from either the HTTP meta data or from the TCP and IP headers and then analyze them to detect the stalls. In our proposed work we do not need to process any meta data or headers. We just use the throughput values delivered to the user device to reconstruct the video playback buffer and analyze it to detect the stalls. While the authors in [8] and [9] have considered just one video quality while estimating the stalls, in our proposed work we are considering three different video qualities while estimating the stalls.

The relationship between the user device buffer and video stall is shown in Fig [1].

The x-axis of the graph represents the video playback time and Y-axis represents the video data available in the buffer. Fig. 1 shows that a video stalls whenever the user device's buffer is empty, and the video playback resumes when the data in the buffer exceeds a certain threshold θ . Different MSPs will have different value for this threshold and it needs to be selected in a way that would allow for maintaining balance between the frequency and the length of the stalls. The red dotted line in the graph denotes that video is stalling and the green line denotes that the video is playing and the time Δt_1 and Δt_2 denotes the length of each stall.

The majority of online video streaming websites define their video quality as the low-standard definition (LSD), standard-definition (SD), high definition (HD), and ultra-high definition (UHD). In order to reconstruct the video playback of the user device, it is very important to estimate the different video qualities that would be playing at the user device and also their

Fig. 1: Illustration of user device buffer during playback

start and stop time. The parameter, which differentiates these qualities is their bit rate, defined as the amount of data required to play a specific duration of the video. The amount of data required per second to play HD video on a user device will be much higher than the amount of data required per second to play the SD video, which means that the throughput values at a user device streaming an HD video will be much higher than the throughput values delivered to a user device streaming an SD or LSD video. Hence, by observing and analyzing the throughput values of the video data being delivered to a user device, we can estimate the different video qualities that would be playing at the user device and their start and stop time. Once we have the above information it can be used to reconstruct the video playback buffer and further to estimate the stalls.

III. PROPOSED ALGORITHM IMPLEMENTATION

To test and implement our algorithm, we streamed Amazon prime videos of different genre on a Dell PC. We captured incoming video packets with the help of wireshark [10] and controlled the network bandwidth with the help of Net-limiter [11]. We implemented our algorithm in Python.

The most important parameter of our algorithm is the bit rate of different video qualities. MSP's nowadays use variable bit rate to encode their videos and hence it is almost impossible to know the exact bit rate of each slice of video over the entire video duration. But by taking a long-time average, we can estimate an average bit rate for different video qualities. We streamed various videos of three different qualities, LSD, SD, and HD, and captured the total received data packets at our desktop for each quality and averaged it over the total video play time to estimate the data required to play one second of the video at each quality. This provides us with the bit rate for LSD, SD and HD video qualities. Further, the bit rate also specifies the minimum throughput values that must be delivered to the user device to play a particular video quality. We define the SD threshold as the minimum throughput value to be delivered to the user device to stream a video at SD quality and that will be equal to SD bit rate and HD threshold

as the minimum throughput value to stream a video at HD quality and that will be equal to HD bit rate.

We implemented our algorithm over the different genre of videos from Amazon videos and controlled the network bandwidth to either make the video stall or to degrade the video quality at the PC screen. Manual observations were made for the number of stalls that occurred on the desktop screen while streaming the video and were later used to verify the accuracy of the algorithm.

The algorithm's pseudo code is provided in Algorithm 1. The algorithm takes two input parameters (1) received video data at the user device and (2) time instants of the received data. The most important step in the algorithm is to identify the start and stop time of the different video qualities at the user device. This is important in order to identify the bit rate to be used to convert the video data to video duration. For our testing purpose, we captured the video data packets using Wireshark, but in real time any ISP can capture these parameters at the router or at the traffic shaper. The output of the algorithm is the number of stalls the user may have experienced.

It is evident that when a video is playing in SD, it can only degrade to LSD, while when a video is playing in HD it can degrade to either SD or LSD depending on the throughput values being delivered to the user device. Hence, to simplify the analysis the algorithm will evaluate the SD and HD video period separately. For a scenario where the video starts at SD quality and later switches to HD quality, the algorithm first identifies the start time of HD quality. This is achieved by analyzing the throughput values. We observed that video data generally comes in a burst manner, hence the algorithm considers the throughput values averaged over 10 seconds. Whenever the average throughput value exceeds the HD threshold for consecutive 30 seconds, we conclude that the HD video data starts coming at the user device.

Next, the algorithm defines a function that will estimate the buffer status of the user device. This function is an implementation of the relation described in Fig. 1, where the input of the function is the received video duration and the output is the predicted number of stalls. The function defines the video at the user device to be either in playing or stalling state by considering the video duration available in the user device buffer. Initially, the video will stall until the buffer has a minimum data duration of θ seconds and this stall duration is called, startup latency, when the video starts playing, the function will add the video duration received at each second in the buffer and subtracts one second of video play duration from the buffer. A stall occurs when the buffer becomes empty. While the video stalls, the function will only add the received video duration and will not subtract any data from the buffer since the video is not playing at the user device. The playback resumes when the user device buffer once again has θ seconds of video. Depending on the number of times the buffer goes empty, this function will return the predicted number of stalls.

The next part of the algorithm evaluates the SD video period to detect any stall. The algorithm first converts the received video data bytes over SD period to video duration using the

estimated SD bit rate and identify any video stall by calling the buffer status function. If the function returns any stall, it means that the user device would have switched to LSD right after the stall. Hence, the algorithm will modify the video duration by using the LSD bit rate to convert the video data to video duration right after the stall until the throughput values goes above SD threshold. After updating the video duration, the algorithm will re-run the buffer status function to detect any further stall. If the function returns any stall, the algorithm will repeat the above defined steps, and if the buffer status function returns no more stall or no stall at all, the algorithm will exit the loop and move to HD calculation.

The next part of the algorithm evaluates the HD video period to detect any stall. Here, we first convert the received video data bytes over the HD period to video duration using the estimated HD bit rate and measure the video stalls by calling the buffer status function. If the function returns any stall, it means the user device would have switched to either SD or LSD right after the stall. The algorithm analyzes the throughput values right after the stall to check which quality the user device degraded to. If the values exceeds the SD threshold, it means the user device switched to SD data, else, the algorithm concludes that the user device switched to LSD data. If the video degrades to LSD, the algorithm will again analyze the throughput values to identify when it goes above SD threshold, which gives us the start and stop time of LSD video. If the video degrades to SD, the algorithm will analyze the throughput values to identify when it goes above HD threshold, which gives us the start and stop time of SD video. The algorithm will then update the video duration by using the corresponding bit rate for the different data received duration. After updating the video duration, the algorithm will re-run the buffer status function to detect any further stall. If the function returns any stall, the algorithm will repeat the defined steps or else will exit the loop.

At this point we are able to identify the different video qualities played at the user device and their start and stop time. The algorithm will now convert the entire received video data to video duration using the appropriate bit rates and will call the buffer status function over the entire video duration to estimate the total number of stalls. Summarizing, the algorithm first estimates the different video qualities and their start and stop time over the entire video duration and then uses the corresponding bit rate to map video data to video duration and finally predicts the total number of stalls the user may have experienced while streaming the video.

IV. ALGORITHM IMPLEMENTATION

The algorithm presented in the following section evaluates the number of stalls for a user streaming an adaptive video of three different video qualities, low standard definition, standard definition, and high definition. We assume that the user starts streaming at SD video quality initially and then upgrade to HD or downgrade LSD due to bandwidth variation. The variation in the bandwidth values of the network causes the throughput values delivered to the user device to vary and

will force the user device to either downgrade or upgrade the requested video quality to improve the user QoE.

Algorithm 1 Algorithm for Stall Detection

Input: Incoming video packets

Input: Time instants of incoming video packets

Output: Number of stalls

```

Initialization :
1: TPS[]=Throughput per second
2: TPTS[]=Throughput per 10 seconds
3: Let y seconds= video stop time
Detecting HD start time:
4: for  $i = 0$  to  $(y/10)$  do
5:   if  $(TPTS[i] \ \& \ TPTS[i + 1] \ \& \ TPTS[i + 2] > HD \ threshold)$  then
6:     SD plays from 0 to  $[i]*10$  seconds
7:     HD plays from  $[i]*10$  to y seconds
8:   end if
9: end for
Buffer status function: Buff_status(x)
10: Buffer += x
11: if video was stalling then
12:   if  $Buffer > \theta$  then
13:     Video starts playing
14:   else
15:     Video continue stalling
16:   end if
17: else if video was playing then
18:   Subtract play time from buffer
19:   if  $Buffer > 0$  then
20:     Video continue playing
21:   else
22:     Video stalls
23:     Stall_count+=1
24:   end if
25: end if
26: Let A := 0 //SD start time
27: for  $Time = A$  to  $i$  do
28:   Video_Duration=Append([(Received bytes)/SD bit rate])
29:   Buff_status(Video_Duration)
30:   if (Video stalls at D seconds) then
31:     for  $Time = D$  to  $i$  do
32:       clear Video_Duration
33:       Let S = Time when throughput rises above SD threshold
34:       for  $Time = D$  to  $S$  do
35:         Video_Duration=Append([(Received bytes)/LSD bit rate])
36:       end for
37:       A := S
38:     end for
39:   else if video never stalls then
40:     No stalls over SD period

```

```

41:   end if
42: end for
43: Let B := i //HD start time
44: for  $Time = B$  to  $y$  do
45:   Video_Duration=Append([(Received bytes)/HD bit rate])
46:   Buff_status(Video_Duration)
47:   if (Video stalls at D seconds) then
48:     for  $Time = D$  to  $y$  do
49:       clear Video_Duration
50:       Let S = Time when throughput rises above SD threshold
51:       for  $Time = D$  to  $S$  do
52:         Video_Duration=Append([(Received bytes)/LSD bit rate])
53:       end for
54:       Let F = Time when throughput rises above HD threshold
55:       for  $Time = S$  to  $F$  do
56:         Video_Duration=Append([(Received bytes)/SD bit rate])
57:       end for
58:       B := F
59:     end for
60:   else if video never stalls then
61:     No stalls over HD period
62:   end if
63: end for
64: Buff_status(Video_Duration)

```

V. PERFORMANCE EVALUATION AND RESULTS

A. Training results

Table 1 provides the estimated bit rate for different video qualities. Since the bit rate specifies the amount of data required to play one second of video, we captured the total data received at our device while streaming different videos for all three video qualities and took an average to estimate the data required to play one second of that particular quality. Videos that have high action, moderate action and no action at all, were all used in training phase to estimate the bit rate for different video qualities. The estimated bit rates were then used in the algorithm as the thresholds (SD and HD) to determine the start and stop of different video qualities and also to map the received video data to video duration.

B. Wireshark graphs

In this section, we will showcase how the throughput values that are being received at the user device vary with the quality of the streamed video. Fig. 2, Fig. 3, and Fig. 4 show the video data received at our test device as captured by wireshark software and averaged over 10 seconds while streaming a video on Amazon prime for different video qualities. The x-axis of the graph has playback time in seconds and Y-axis has throughput values in bits/sec.

1) *Predicting the video quality:* Fig [2] showcases how the throughput values vary at user device while streaming SD and HD video qualities. The user device was streamed at SD video quality up to 130 seconds and at HD video quality after 130 seconds. It can be observed that the throughput values being delivered to the user device while playing an SD video are way too less than the throughput values while playing an HD video. Hence, we can say that the variation in the throughput values delivered at the user device while streaming an adaptive video can be analyzed to predict the different video qualities that might be playing at the user device.

Fig. 2: Variation of throughput values while streaming SD and HD video

2) *Degradation over SD duration:* Fig. 3 showcases the effect of throughput degradation when the video is playing at SD quality. The user device starts requesting SD video initially at 0 seconds and receives the throughput values of approximately 0.9 Mbps. At 90 seconds, due to poor bandwidth availability, throughput values degrades to almost 0.15 Mbps. At this point, even though the throughput values are way less than the SD threshold, the user device continue to demand SD data and utilizes the buffered data to keep playing the video at the user device screen. At 130 seconds, the user device buffer becomes empty and the video stalls. Just after the stall, the user device starts requesting LSD video data since it can only degrade to LSD video quality. The user device continues to demand LSD data up to 210 seconds when the throughput finally rises above SD threshold and once again starts requesting the SD video data.

3) *Degradation over HD duration:* Fig. 4 showcases the effect of throughput degradation when the video is playing at HD quality. The user device starts requesting HD video initially at 0 seconds and receives the initial throughput values of approximately 1.7 Mbps. Here, the throughput values

Fig. 3: Degradation over SD duration

LSD (Mbps)	SD (Mbps)	HD (Mbps)
0.25	0.585	1.53

TABLE I: Estimated bit rate

degrade at 140 seconds. At this point, the user device continues to demand HD data and utilizes the buffered data to keep playing the video. The video eventually stalls at 180 seconds and at this point the user device analyzes the recent throughput values to decide whether it should request the SD or LSD quality video data. If the recent throughput values lie above the SD threshold, user device should request SD data and if it lies below the SD threshold, it should request LSD data. In Fig. 4, when the video stall at time 180 seconds, since the recent throughput values are way below the SD threshold, the user device starts requesting LSD video data up to 210 seconds after which the throughput rises above the SD threshold and hence the user device starts requesting SD data. At 260 seconds, the throughput values again rise above HD threshold and now the user device once again starts requesting HD video data.

C. Test data results

We tested our algorithm for 20 different Amazon prime videos, each of length 1000 seconds where each video was streamed on low-standard, standard, and high definition for varying duration of time. Fig. 5 shows the comparison between observed and predicted number of stalls. For 65% of test data, our algorithm accurately predicted the number of stalls. For the next 25% of test data within ± 1 stall and rest 10% within ± 2 stalls.

For the test cases when the algorithm over-predicted the number of stalls, it was because the estimated bit rate was higher than the actual bit rate of the videos used in those test cases while when the algorithm under-predicted the number

Fig. 4: Degradation over HD duration

of stalls, it was because the estimated bit rate was lower than the actual bit rate.

Fig. 5: Comparison between observed number of stalls at user device and number of stalls predicted by the algorithm.

VI. CONCLUSIONS & FUTURE WORK

In this work, we proposed an algorithm that predicts the number of stalls a user may experience by using only the throughput values being delivered to the user device. This approach allows for estimating the stalls the user would have experienced in situations where the application service providers don't have any access to their users' buffer status, which further helps in estimating the user QoE metric. In the future, we plan to increase the prediction accuracy of our algorithm and we would also like to extend our algorithm to measure the length of each predicted stall.

VII. ACKNOWLEDGMENT

This work is supported by ViaSat Inc. Authors would like to especially thank Dr. Nandan Das, Shreyas Vidyarthi, Nobuo

Fukaya, Kevin Rennie, Andy Tirella, Kristi Jaska, and Josh Slater for sharing their insights on this topic.

REFERENCES

- [1] A. Perkis, S. Munkeby, and O. I. Hillestad, "A model for measuring quality of experience," in *Proceedings of the 7th Nordic Signal Processing Symposium - NORSIG 2006*, June 2006, pp. 198–201.
- [2] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over http," in *2012 19th International Packet Video Workshop (PV)*, May 2012, pp. 173–178.
- [3] O. Gurewitz, M. Sidi, and S. Cidon, "The ballot theorem strikes again: packet loss process distribution," *IEEE Transactions on Information Theory*, vol. 46, no. 7, pp. 2588–2595, Nov 2000.
- [4] A. ParandehGheibi, M. Medard, A. Ozdaglar, and S. Shakkottai, "Avoiding interruptions x2014; a qoe reliability function for streaming media applications," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 1064–1074, May 2011.
- [5] 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 *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Sept 2014, pp. 1996–2001.
- [6] Y. Xu, E. Altman, R. El-Azouzi, M. Haddad, S. Elayoubi, and T. Jimenez, "Probabilistic analysis of buffer starvation in markovian queues," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 1826–1834.
- [7] A. Anttonen and A. Mmmel, "Interruption probability of wireless video streaming with limited video lengths," *IEEE Transactions on Multimedia*, vol. 16, no. 4, pp. 1176–1180, June 2014.
- [8] R. Schatz, T. Hofeld, and P. Casas, "Passive youtube qoe monitoring for isps," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, July 2012, pp. 358–364.
- [9] H. Tang, L. Chen, Y. Wang, N. Wang, and X. Li, "Stalling assessment for wireless online video streams via isp traffic monitoring," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2017, pp. 1–6.
- [10] "https://www.wireshark.org/."
- [11] "https://www.netlimiter.com/."
- [12] T. Zhao, Q. Liu, and C. W. Chen, "Qoe in video transmission: A user experience-driven strategy," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 285–302, Firstquarter 2017.
- [13] Y. Chen, K. Wu, and Q. Zhang, "From qos to qoe: A tutorial on video quality assessment," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 1126–1165, Secondquarter 2015.
- [14] G.-M. Su, X. Su, Y. Bai, M. Wang, A. V. Vasilakos, and H. Wang, "Qoe in video streaming over wireless networks: perspectives and research challenges," *Wireless Networks*, vol. 22, no. 5, pp. 1571–1593, Jul 2016. [Online]. Available: <https://doi.org/10.1007/s11276-015-1028-7>
- [15] 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 *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Sept 2014, pp. 1996–2001.