A Survey on YouTube Streaming Service

Majed Haddad[1], Eitan Altman[1], Rachid El-Azouzi[2], Tania Jiménez[2],
Salah Eddine Elayoubi[3], Sana Ben Jamaa[3],
Arnaud Legout[1] and Ashwin Rao[1]
[1]INRIA, Sophia-Antipolis, France. email: majed.haddad, eitan.altman@inria.fr
[2]LIA, Université d'Avignon, France
[3]France Telecom, Issy les Moulineaux, France

ABSTRACT

Established in 2005, YouTube is one of the fastest-growing websites, and has become one of the most accessed sites in the Internet. It has a significant impact on the Internet traffic distribution, but itself is suffering from severe scalability constraints and quality of service. Understanding the features of YouTube is thus crucial to network traffic engineering and to sustainable development of this new generation of services. In this survey, we first present an overview of previous works and analysis on YouTube with a particular attention on Quality of Experience. We then describe how the increased availability of meta-data in Web 2.0 (e.g., popularity distribution of video clips) could be effectively exploited to improve the performance and scalability of YouTube. In particular, we study the benefit gained by local caching along with prefetching in terms of reducing the client access time and start up delay in watching video.

1. INTRODUCTION

According to estimates, with 100 million video views per day YouTube accounts for approximately 60% of the videos watched on the Internet; YouTube is also growing at a rapid pace, with 65,000 video uploads per day [1]. Studying the attributes of YouTube is valuable for network traffic management. Distinct characteristics of YouTube site direct researchers to many new challenges. For example, huge popularity of YouTube has imposed a significant impact on the Internet traffic which resulted in scalability limitations for YouTube and large bandwidth demand. All these aspects make it very important to have a look at the end users' perception of YouTube videos.

Quality of Service (QoS) and Quality of Experience (QoE) of YouTube's video sharing services are undoubtedly key concepts within today's innovation development process of video sharing services. In practice however, both concepts are still often confused. Furthermore, it was shown that the measurement of QoS and QoE still happens too fragmented and that the current measurement approaches tend to be focused on one particular aspect or dimension. As a result, there is still a prevailing gap between QoE-measurement and QoS-development. The degree of personalization and especially the users' expectations, satisfaction and perceived experience QoE, have become crucial determinants for the success of a service like YouTube, even more important than its performance (QoS). Hence, the ultimate measure for mobile media networks and services is how the end-user perceives and experiences the quality of the new media and services. As a result, one of the major challenges consists in making the offer of media technologies and services user-centric enough, and in narrowing this above mentioned gap between QoE and QoS. The point is that measurement for QoE would make operators, device providers and service providers more convenient to judge and improve the service they provide for YouTube consumers.

This paper has the objective to cover and relate different works on YouTube. We begin by presenting an overview of YouTube traffic characterization and analysis. We review different contributions on YouTube workload characterization. Next, we present recent approaches for buffer management that tends to improve the quality of streaming media delivery. We then describe the current subjective and objective QoE procedures approaches for measuring subjective QoS along with metrics to manage QoE for multimedia streaming. In order to gain insights into current streaming services and thus provide guidance on designing resource efficient and high quality streaming media systems, we then present caching and prefetching solutions of popular YouTube videos. We survey the approaches being applied to system design described in the literature to address the YouTube performance issues such as scalability and large bandwidth demand. Last, we conclude with a discussion of outstanding issues and promising areas for future work on deriving more informative indications for quality of experience measurement and quality of service development.

2. YOUTUBE DESCRIPTION

According to estimates, with 100 million video views per day YouTube accounts for approximately 60% of the videos watched on the Internet; YouTube is also growing at a rapid pace, with 65,000 video uploads per day [1] with an average size of 10 MB for each video. This constant growth of YouTube makes capturing its behavior by examining a single point in time almost impossible. In the literature, one can find many analysis that reflect trends of YouTube traffic from both a local campus network and a global (i.e., Internet wide) perspective [2], [3], [4], [5], [6] and [7].

VALUETOOLS 2011, May 16-20, Paris, France
Copyright © 2011 ICST 978-1-936968-09-1
DOI 10.4108/icst.valuetools.2011.245880

2.1 How does YouTube work?

One of the keys to YouTube's success is its use of Adobe's Flash Video (FLV) format for video delivery [8]. While users may upload content in a variety of media formats (e.g., WMV, MPEG and AVI), YouTube converts them to Flash Video before posting them. This enables users to watch the videos without downloading any additional browser plugins provided they have the Flash Player installed. To enable playback of the Flash video before the content is completely downloaded, YouTube relies on Adobe's progressive download technology. Traditional download-and-play requires the full FLV file to be downloaded before playback can begin. Adobe's progressive download feature allows the playback to begin without downloading the entire file. This is accomplished using ActionScript commands that supply the FLV file to the player as it is being downloaded, enabling playback of the partially downloaded file. Progressive download works with Web servers and video content is delivered using HTTP/TCP. This delivery technique is sometimes referred to as *pseudo-streaming* to distinguish it from traditional media streaming. Traditional on-demand streaming of stored media files typically requires the use of dedicated streaming servers that facilitate client-server interaction during the course of the video playback. This interaction may be used for adaptation of video quality or user interactions such as fast forward or rewind operations. While video content is usually the focus of a visit to the YouTube Web site, there are many file transfers that happen behind the scenes to embed the video file and display the surrounding Web site content. For example, when a user clicks on a video of interest, a GET request for the title HTML page for the requested video is made. This HTML page typically includes references to a number of Javascript files. These scripts are responsible for embedding the Shockwave Flash (SWF) player file, and other peripheral tasks such as processing video ratings and comments. The SWF file is relatively small (26 KB), so the page loads quickly. Once the player is embedded, a request for the FLV video file is issued. The FLV video file is downloaded to the user's computer using an HTTP GET request, which is serviced by either a YouTube server or a server from a content distribution network (CDN).

Fig.1 illustrates the communication among the client, YouTube server, and a server of the CDN. When a client has chosen a specific video, an HTTP GET message is sent from the client to the YouTube web server. This message indicates that the client is requesting a specific video which is identified by a unique video identifier. After receiving the GET message, the web server replies with a redirection message. This message contains a location response-header field, which redirects the client to the video server from which the video will be streamed. The redirection mechanism introduces load balancing to the system since the web server can redirect the client to a video server in a dynamic way. Therefore, the web server must know which videos can be served from which video servers and the actual load of the video servers. To allow for load balancing, YouTube makes use of a CDN service provided by both the YouTube and Google Video server farms and the Limelight CDN [8]. Such a service automatically distributes videos to servers in the CDN.

2.2 YouTube Traffic Characterization

In [3], Gill *et al.* analyze and characterize YouTube by considering the network resources consumed by YouTube traffic as well as the viewing habits of campus YouTube users. Globally, authors consider characteristics of the most popular videos on YouTube and examine the relationship between globally popular videos and videos that are popular on campus. In particular, it was found that approximately 24% of video transactions fall into the "*interrupted*" category. It is argued that there are two primary reasons why a video download may fall into this category:

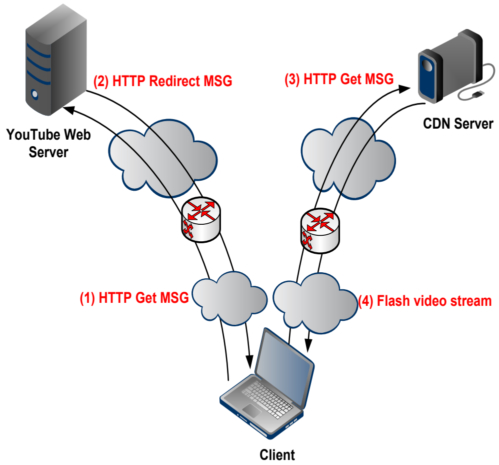

Figure 1: Video retrieval in YouTube [7].

- *poor performance*: i.e., slow download rate,
- *poor content quality*: e.g., the viewer does not find the content interesting.

For example, approximately 10% of the interrupted transactions had a slower download speed than encoded bit rate. For these transfers, the users likely became impatient with the video playback and aborted the transfer. Another 80% of interrupted transfers had ratios similar to the bulk of the completed transfers. For these authors hypothesized that the users simply found the content uninteresting, and aborted the transfer some time before the end of the video.

In [4], using traces crawled in a 3-month period, Cheng *et al.* reported that 87.6% of the videos contained FLV metadata specifying the video's bit-rate in the beginning of the file, indicating that they are Constant-Bit-Rate (CBR). For the rest of the videos that do not contain this meta-data (probably Variable-Bit-Rate, or VBR, videos), they calculate an average bit-rate from the file size and its length. It was found that most videos have a bit-rate around 330 kbps, with two other peaks at around 285 kbps and 200 kbps. This implies that YouTube videos have a moderate bit-rate that balances the quality and the bandwidth.

In [5], Abhari *et al.* developed a synthetic workload generator for YouTube in order to measure the performance of several scenarios with variable workload parameters. Zink *et al.* conduct in [7] a measurement study of YouTube traffic in a large university campus network to investigate the local popularity of YouTube video clips, the relationship between local and global popularity, and how time scale and user population impact the local popularity. Specifically, it is shown that the request rate is high during the day and the evening until around midnight and then decreases significantly in the early morning hours. Moreover, results emphasize the fact that a large number of videos are only requested once

(77% in all the traces) while a smaller fraction (23%) are requested at least twice.

3. BUFFER MANAGEMENT

While distributing the video content, a variety of functions can be called on the flash player to control the external playback of the video and to customize the player. One of the important controls of that kind is to set buffer size. There is a possibility to specify the number of seconds to buffer in memory before starting the playback. Buffer size can also be reset to a higher value during playback [9]. Service providers use different buffering strategies to provide a smooth playback experience to the client. Following are the three different buffering strategies which are widely in use:

- Standard video buffering,
- Dual-threshold buffering,
- H.264 encoded video buffering.

3.1 Standard video buffering

Streaming servers manage buffering in a rather simple manner. The server sends data to the client, which stores data in a buffer. The data is sent as quickly as possible. When the buffer is full (because it has reached a predefined threshold of stored data), the video playback starts and the server continues to send data to the client but it only sends the amount of data necessary to keep the buffer full. If the data rate becomes insufficient to keep the buffer full, the buffer becomes empty in the span of a few minutes. When this happens, the video playback is stopped and a rebuffering phase begins again. A small buffer results in instant playback, while a wider buffer provides higher resilience to bandwidth fluctuations. A video delivery application developer must choose between resilience and user experience: if you set a buffer too small, it fills up quickly but is vulnerable to adverse channel conditions. A sudden, temporary drop in data flow can easily empty a small buffer, causing the video playback to stutter. On the other hand, if you set a buffer too big, the playback is very reliable but requires that viewers endure a longer (and annoying) buffering time before the streaming video starts playing. In both cases, for some viewers, the experience could be considered unsatisfactory.

3.2 Dual-threshold buffering

The standard buffering process is vulnerable to bandwidth drops, as well as being unable to exploit a sudden increase of bandwidth. A dual-threshold buffering strategy addresses this issue, assuring a fast start and, at the same time, providing fairly high resilience to bandwidth fluctuations or other adverse conditions [10]. When the buffer is filled with a given amount of data (the first threshold), the playback starts as expected. But instead of trying to keep the buffer full to this level, the modified strategy involves trying to fill the buffer to a second, higher threshold. This extra amount of data could be extremely useful later if the Internet connection encounters temporarily bandwidth drops or fluctuations. However, in reality, we notice that prefetching does not stop after the beginning even for large videos. This could suggest that either YouTube does not apply this dual-threshold policy or it uses a high value of upper threshold.

3.3 H.264 encoded video buffering

The buffering of H.264 encoded videos is much more complex than the normal FLV video buffering because of the complexity in the encoding mechanism. H.264 uses various encoding methods and strategies. In H.264 encoded video, video frames can have multiple references in past and future. So, it might be required to load several frames before starting the playback. This means that the videos encoded with H.264 usually requires a deeper buffer. Because of this service providers are not encouraged to restrict the buffering of H.264 encoded videos. They might not be able to see the expected behavior of Flash Player, if there are any buffering limits. Flash Player does not restrict the buffering of H.264 encoded videos and it does not strictly follow the user specified initial buffer length [10].

3.4 Buffer Management Mechanism for TCP Streaming

A major problem in media streaming service is the quality of the TCP link caused by the network congestion. First, in order to start viewing TCP streaming, the users must get the first few percent of video data in the initial phase. After this phase, they can start viewing firstly. However, when network congestion occurs, and as a result, the bandwidth of transferring video file is reduced in this initial phase, there is significant delay to start viewing. Second, to keep viewing TCP streaming, the users must continue getting video data while viewing video. However, when network congestion occurs and the delivery bandwidth is excessively decreased on viewing TCP streaming, there occurs significant outage due to pausing the video in the middle of the view. These problems are caused by current best-effort network which does not control the bandwidth and the latency. In [11], it is proposed that TCP streaming should be guaranteed either the following two bit-rates according to the situation. First is the limited bandwidth per single flow when users download from the server (called "*limited rate*"). Second is video bit-rate in viewing video(called "*viewing rate*"). Under this situation, it enables to complete transferring video file in short time. Even if network fell into congestion during the transfer, the QoS of viewing the video is not significantly affected because almost all video data have been transferred before the playback. Allocating maximum rate to such flows in congestion is too excessive service and the limited capacity of network bandwidth is wasted. On the other hand, if there is no data to continue or start viewing video in user, reducing the bandwidth significantly affects QoS of viewing video.

4. SUBJECTIVE-OBJECTIVE QOE PROCEDURES

The evaluation of QoE can use either subjective or objective procedures. Subjective quality evaluation processes use subjective listening and watching tests, and require large amount of human resources. The most commonly used subjective video quality metric is the Mean Opinion Score (MOS). As an example, Perceptual Estimation of Video Quality (PEVQ) is a standardized measurement algorithm to estimate the user perceived quality of a video in terms of MOS. PEVQ is trained with subjective measurements of user experience and it has been proven that the output is correlating well with subjective results [12]. On the other hand, objective evaluation methods provide QoE evaluation results faster using physical instruments and/or algorithms. The most commonly used objective video quality metric is the Peak Signal-to-Noise Ratio (PSNR). Subjective QoE measurement is accurate and reliable, provided it is performed under stringiest conditions; but it is costly and time consuming. For that reason, more and more effort has been placed on developing objective measures that would evaluate service quality with high correlation [13]. Moreover, there exists in [14] a PSNR to MOS mapping: as an example a PSNR in dB of at least 25, 31 and 37, can be roughly mapped to a MOS of 3 (fair), 4 (good) and 5 (excellent), respectively. However, this mapping is only a rough estimate as PSNR does not capture all the aspects of human perception of a video [15].

Managing QoE for video streaming services has been extensively studied in both wired and wireless networks. Data from streamed video is usually buffered at the receiver before playback to prevent interruptions due to end-to-end packet delay variations. However, increasing the buffered data also increases the end-to-end delay. For Variable Bit Rate (VBR) videos, various smoothing techniques have been proposed to increase the network bandwidth utilization and to reduce receiver buffering requirements, while ensuring continuous playback [16]. Another approach for reducing receiver buffer is to adapt the playout rate [17].

QoE metrics for multimedia streaming

Internet connections are characterized by a number of statistically determined features, including bandwidth and latency. Bandwidth and latency values are not guaranteed. In fact, they can fluctuate a great deal depending on the local ISP network load and remote server load, as well as by local applications sharing network resources. Video delivered using a ”deterministic” channel (such as satellite, DVD, cable or, digital TV broadcasting) requires only a small buffer because data arrives to the media player with deterministic delay and rate, and very limited or infrequent data drops. Video delivered over IP networks is much more problematic. There is no guarantee that the data will flow to the users at the sufficient rate and determined delay. Instead, it arrives with both a rate and delay that can change consistently as the video streams. Therefore, an accurate buffering system for QoE of video streams is necessary for web delivery. Recently, there has been considerable interest in managing QoE for multimedia streaming [18], [19]. The QoE for a streaming service depends on conventional network metrics such as bit rate, packet loss rate, packet delay and jitter, and various video sequence specific factors, such as its encoding scheme and the video streaming application. Two additional important QoE metrics in a video streaming service are the receiver starvation probability and the received video quality [20]. The starvation probability is the long-run fraction of frames or packets that miss their playout deadline at the receiver. The received video quality can be measured using subjective or objective video quality metrics.

5. IMPLICATIONS FOR SERVICE PROVIDERS

In general, the analysis of the YouTube workload emphasis the fact that there are many similarities to media streaming workloads. For example, access patterns are strongly correlated with human behaviors, as traffic volumes vary significantly by time-of-day, day-of-week, as well as longer term activities (e.g., academic calendars). Similarly, video files are much larger than files of other types, and some videos are more popular than others. These and other characteristics suggest that caching should improve the performance and scalability of YouTube videos. However, there are differences as well. In particular, enabling anyone to publish content means growth in content will not only be larger than for traditional Web and media, but sustainable. This will place greater strain on centralized resources, and require decentralized approaches such as caching and CDNs. In addition, since much of the content is likely to be unpopular (the long tail effect), it will be important to minimize the cost for storing that content. This has several implications for the service provider, who must plan, purchase, install, operate and maintain the central infrastructure used by the site. Furthermore, the breadth and depth of available content reduces the concentration of references in the access stream, which can reduce the effectiveness of caching and prefetching strategies.

5.1 Implications on caching

One of the design criteria for video distribution systems is the popularity distribution of video clips requested by users in the access network. Obtaining and then analyzing video clip popularity allows us to identify critical design parameters for the appropriate infrastructure for a video distribution system like YouTube. Caching the most popular videos in the proxy along with prefetching the related videos in the client site can reduce the client access time and start up delay in watching video. In addition, the popularity distribution can be used to derive modeled data from the traces. As an example, observation of Zipf-like behavior in popular data set suggests that proxy caching of only YouTube popular videos instead of caching of all kinds of videos can significantly increase the scalability of the server and decrease the network traffic. In general, the benefit gained by local caching can be increased if popularity information about each video content is available. There is considerable published literature on caching of adaptive multimedia streams [21], [22] and [23].

In [7], authors used the trace collected from the measurement study to conduct trace-driven simulations to analyze and compare the potential bandwidth savings achieved by the alternative distribution infrastructures. The results of these simulations show that client-based local caching, P2P-based distribution, and proxy caching can reduce network traffic significantly and allow faster access to video clips.

5.2 Prefetching Related Videos

Exploiting the concept of related videos can improve the cache efficiency at the client site for regular videos. In the other word, for a group of videos related to each other, it is very possible a viewer watches next video from the related list after viewing the previous one. Therefore, once a video is played back by a user, the related videos of the selected video can be prefetched and then cached in the client site. Prefetching results in eliminating or reducing the startup delay in the client site and enhancing the playback quality.

Cheng *et al.* propose in [4] a novel peer-to-peer system based on social network, in which peers are re-distributing the videos that they have cached. A prefetching strategy is used to prefetch the beginning part of the next video in the related social network in each peer in order to decrease the start up delay of P2P overlay. Social networks existing among YouTube videos are made by groups of related videos, if one video has been watched, another video within the group will be more likely to be accessed. Their design is based on the following principle: peers are responsible for re-distributing the videos that they have cached. Cheng *et al.* [4] have also suggested that this fact can be used to design new peer-to-peer methods for short video sharing.

In [5], authors examined the idea of proxy caching of popular videos by conduction simulation and proposed the idea of prefetching of related videos at the client sites for regular videos. However, for each video the number of related videos is significant and user may watch none of the related videos. Considering this fact, the optimum number and bytes of the related videos for prefetching in the client site should be determined. Moreover, as [3] found that approximately 80% of the *interrupted* videos are due to viewers being uninterested by the content, it would be of particular interest to take this into account while prefetching related videos. This could be a good cache replacement strategy to save the cache space.

6. CONCLUSION

Modern Internet streaming services like YouTube have utilized various techniques to improve the quality of streaming media delivery. As an example, caching the most popular data along with prefetching the related videos at proxies close to clients is an efficient approach to save bandwidth and prevent user latency. We surveyed several classes of caching policies. Despite the characterization of media access patterns and user behaviors in many measurement studies, we found that few studies have focused to derive modeled data from the traces. Indeed, even though caching and prefetching techniques has been successfully employed in many real-life systems, much work is required for achieving a deep understanding on how to design systems that can provides significant improvement in terms of experienced quality of service and resource efficiency. First, caching of video for progressive download typically involves caching of the entire video such that the entire video is replaced during cache replacement. Thus, when the replaced video is requested by the client again, the entire video has to be fetched from the server, resulting in inefficient utilization of server and network resources. Second, the increased availability of metadata in media services like YouTube (a direct result of social networking) can and should be exploited to make such models more effective especially in deriving modeled data from the traces and predicting the behavior of content that are more likely to create a buzz.

7. REFERENCES

[1] "USA Today, YouTube serves up 100 million videos a day online," July 2006.

[2] M. Cha, H. Kwak, P. Rodriguez, Y. yeol Ahn, and S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the world's largest user generated content video system," in *In Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMCŠ07)*, 2007.

[3] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube traffic characterization: A view from the edge," in *In: Proc. of IMC*, 2007.

[4] X. Cheng, "Understanding the characteristics of internet short video sharing: YouTube as a case study," in *Procs of the 7th ACM SIGCOMM Conference on Internet Measurement, San Diego (CA, USA)*, 2007, p. 28.

[5] A. Abhari and M. Soraya, "Workload generation for YouTube," *Multimedia Tools Appl.*, vol. 46, pp. 91–118, January 2010.

[6] G. Chatzopoulou, C. Sheng, and M. Faloutsos, in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, March 2010, pp. 1–6.

[7] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network - measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501 – 514, 2009.

[8] Api documentation (Youtube). [Online]. Available: http://youtube.com/dev

[9] A. Kapoor, "Actionscript guide to dynamic streaming," Jan 2009.

[10] F. Sonnati, "New buffering strategies in Flash Player 9 and Flash Media Server 3," March 2008.

[11] N. Yoshihara, H. Tode, and K. Murakami, "Buffer management mechanism suitable for TCP streaming in qos-aware ip router," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, 2010.

[12] "Pevq advanced perceptual evaluation of video quality." [Online]. Available: http://www.opticom.de/technology/pevq.html

[13] Y. Huang, "Towards efficient and accurate QoE reporting for emerging streaming services," in *Integrated Network Management-Workshops, 2009. IM '09. IFIP/IEEE International Symposium on*, 2009, pp. 225 –230.

[14] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid - a framework for video transmission and quality evaluation," in *In Proc. of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 2003, pp. 255–272.

[15] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of psnr in image/video quality assessment," *Electronics Letters*, vol. 44, no. 13, pp. 800–801, 2008.

[16] W.-C. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," in *In Proc. IEEE INFOCOM*, 1997, pp. 58–66.

[17] M. Kalman, E. Steinbach, M. Ieee, B. Girod, and F. Ieee, "Adaptive media playout for low delay video streaming over error-prone channels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 841–851, 2004.

[18] A. Vishwanath, P. Dutta, M. Chetlu, P. Gupta, S. Kalyanaraman, and A. Ghosh, "Perspectives on quality of experience for video streaming over wimax," *Mobile Computing and Communications Review*, vol. 13, no. 4, pp. 15–25, 2009.

[19] D. Migliorini, E. Mingozzi, and C. Vallati, "QoE-oriented performance evaluation of video streaming over wimax," in *Wired/Wireless Internet Communications*, vol. 6074. Springer Berlin / Heidelberg, 2010, pp. 240–251.

[20] P. Seeling, M. Reisslein, and B. Kulapala, “Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial,” *Communications Surveys Tutorials, IEEE*, vol. 6, no. 3, pp. 58 –78, 2004.

[21] R. Rejaie, H. Yu, M. H, and D. Estrin, “Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet,” 2000, pp. 980–989.

[22] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, “Distributing layered encoded video through caches,” in *IEEE Transactions on Computers*, 2001, pp. 622–636.

[23] S. Chattopadhyay, L. Ramaswamy, and S. M. Bhandarkar, “A framework for encoding and caching of video for quality adaptive progressive download,” in *Proceedings of the 15th international conference on Multimedia*, 2007, pp. 775–778.