A Survey of Challenges in Streaming Multimedia Systems on the Internet

E. O. Oyekanmi[1], S. A. Oluwadare[2], O. Olabode[3] and
O. K. Boyinbode[4]
[1]Department of Mathematical Sciences,
Achievers University, Owo, Ondo State,
Nigeria.
[2,3,4]Department of Computer Science,
Federal University of Technology, Akure, Ondo State,
Nigeria.
Email: [1]e.oyekanmi@achievers.edu.ng, [2]aoluwadare@futa.edu.ng, [3]o.olabode@futa.edu.ng,
[4]okboyinbode@futa.edu.ng

ABSTRACT

Internet streaming video poses many challenges especially the issue of overloading on multimedia server. To address these challenges, extensive research has been conducted. This paper is aimed at disseminating the contributions of some researchers in the field of streaming video over the internet and highlighting some common issues found in a distributed multimedia system. It also provides an integral view of the necessary way-forward to overcome these challenges. Specifically, this paper covers components of multimedia system, problems posed by them and mechanisms to tackle these problems. Discussion was also done on the trade-offs on the solutions which can be implemented in any of the multimedia components.

Keywords: *Video Streaming, Multimedia System, Admission Control, Servers' Overloading.*

African Journal of Computing & ICT Reference Format:

E. O. Oyekanmi, S. A. Oluwadare, O. Olabode and O. K. Boyinbode (2019), A Survey of Challenges in Streaming Multimedia Systems on the Internet,
Afr. J. Comp. & ICT, Vol.12, No. 4, pp. 115 - 121.

I. INTRODUCTION

Streaming media is video or audio data transmitted in compressed form via the internet which must be played out continuously, rather than being saved to the hard drive. In other words, a user does not necessarily have to download a file before it is played. Once the media is sent in a continuous stream of data it plays as it arrives. Users can pause, rewind or fast-forward, just as it could be done with a downloaded file. In actual fact, however, the real-time transport of live video or stored video is the predominant part of real-time multimedia [1]. This paper is concerned with the challenges of the video streaming which is the real-time transmission of stored video from a multicore system.

The two modes for transmission of stored video over the internet include: the download mode and the streaming mode (i.e., video streaming). In the download mode, a user downloads the entire video file and then plays it back.

The file is saved to one's computer or network media player's hard drive where it will be played at a later time. Services like iTunes and Vudu, permit watching of a movie downloads after a sufficient amount of time [2], however, in the streaming mode, the video content need not be downloaded in full, but is played out while parts of the content are being received and decoded [1]. This is also referred to as Video-on-demand (VoD) streaming. There are various online video services (platforms) that enable one to watch movies online, examples include: NetFlix, YouTube, iROKOtv and iBAKATV.

During streaming mode transmission, a network media player or media streamer (including Smart TVs and most Blu-ray players) can access a requested file and play it without the need to move or copy the file to the device that is playing it. Videos played on YouTube or a TV show such as ABC, NBC, CBS, or Hulu, are streamed media from the website to one's computer, network media player, or media streamer. In a nutshell, streaming is like waterfall, it happens in real-time [2]. The stream media server software realizes functions of applying data stream from data source and network distribution through multithreading. The data source can be files that are in a directory of personal computer (PC); real-time facilities such as DVR/DVS/HC board card or any other streaming device. With this, sizing streaming media server hardware could therefore be a difficult process [3].

When people (media server owners) are asked how much server capacity will achieve their aim? The answers usually include "as little as possible," and "enough to get the job done". This uncertainty is understandable because they don't want to buy too much, and then end up in shortage, either. However, these answers were given by the media server owners because of the unpredictable, resource-intensive nature of streaming-video delivery. There could be many factors that affect smooth streaming-video delivery, the most important consideration is balancing two constraints which include: network bandwidth and processing capacity.

Considering the fact that there are large numbers of sources and perhaps large numbers of stream, these constraints can quickly become bottlenecks in streaming video delivery. More so, when the bandwidth limitation is reached or the system runs out of memory, the servers could as well hit maximum capacity when streaming. However, there are some best practices that can be followed to avoid reaching server overload. These are discussed in section IV of this paper.

II. LITERATURE REVIEW

Research has shown that the social trends and technological advances have led to the advent of various popular web-based live streaming platforms, such as YouTube Live, TwitchTV, Instagram Livestream and Facebook Live. In reality, these platforms are designed to maximize scalability of streaming delivery on live mode, however, the delay (several seconds or more) is still relatively high thereby given room for the usage of larger buffer, heavier compression and more effective transcoding techniques than they otherwise could [4]. Multimedia, traditionally, was known with limited support [5, 6]; hence, applications that depend on it had to find workarounds. Mostly, many chose to rely on non-standard plugins such as Java Applets5 or Adobe Flash while the rest accepted a significant decrease in their quality or performance, or could not be migrated at all during streaming process [4].

Although, with the advent of HTML5 and other related Web standards such as WebGL, this issue of delay is starting to change. However, one of the features for which applications had to rely, as an external plug-in, was video streaming which also causes some delay [4]. Human Computer Interaction (HCI) analysis conducted by Nielsen in 1994 shows that beyond a 0.1 seconds delay a user can notice a system is not reacting instantaneously, and beyond 1 second the user's flow of thought is interrupted [7]. In other words, there still exist a very significant delay

between the moment a frame is captured and the moment it is displayed in the target device [8, 9]. The delay is not only as a result of network or hardware latency but also from the design to achieve a higher scalability [10]. This paper, aside delay, discussed more issues related to video streaming in server, network and clients in a distributed multimedia system or VoD system. An example of VoD system is YouTube.

III. COMMUNICATION BETWEEN CLIENT AND YOU-TUBE

YouTube, as shown in Fig1, works on 3-tier physical cache hierarchy using 38 primary locations, 8 secondary locations and 5 tertiary locations [11]. When a client requested a video using HTTP GET message in the form of Uniform Resource Locator (URL) such as http://www.youtube.com/watch?v=t4H_Zoh7G5A, it returns to a HTML page with embedded URLS and points to the respective dash video server that is responsible to serve the video. On the click of play button of the selected video, another HTTP GET message will be sent from client to server to put on buffer a video with the unique video identifier. On the receipt of the GET message by the server, a HTTP 303, which contains location response that redirects the client to the video servers, is generated to stream the requested video.

YouTube is a popular and ever-increasing multimedia system, therefore some strategies have been implemented in the past to ensure efficient network traffic engineering so as to get sustainable development [11]. However, with a lot of strategies put in place, client still experience various problems such as freezing and re-buffering as shown in Fig2 during video delivery.

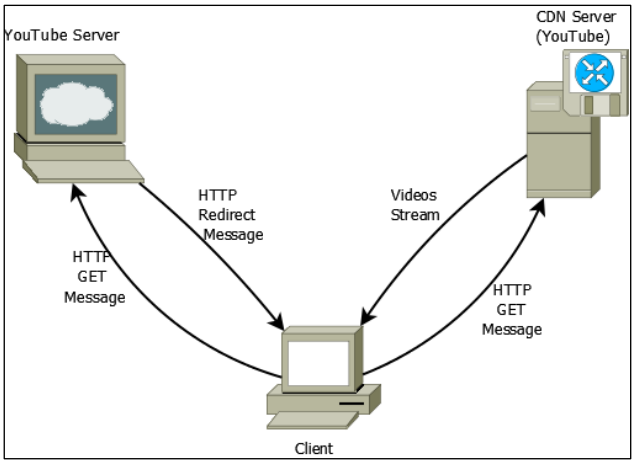

Fig 1: Architecture of the Communication between client and YouTube [11]

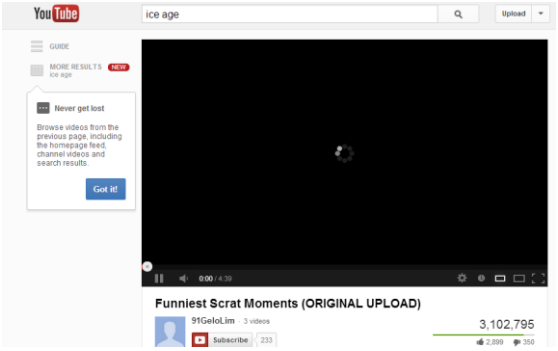

Fig 2: Snapshot of video buffering [11]

IV. COMPONENTS OF MULTIMEDIA AND CHALLENGES THEREIN

Three major components constitute a distributed multimedia streaming system: the server, the network, and the clients.

A. Server

The first step to create a streaming system is the selection of the server. Apple opens the multimedia field in 1991 and since the appearance of RealNetworks in 1995, the multimedia transmission over internet was available [12]. Referring to streaming as a technology, it arises in 1995 for the transmission of audio, and a little bit later, the transmission of video. This technology opened possibilities over the internet to allow access to the content without the need to download it.

Some of the most important characteristics that allowed this progress are the new codecs and the types of compression. The main function of these features is to optimize the quality of the service in a real-time.

A streaming server usually performs three tasks namely: process client requests, retrieve the requested media data and then transmit it into the network [13]. While processing the client request, the server needs to parse it by utilizing the Central Processing Unit (CPU). Once the request is parsed, the server populates it into memory buffers with the data requested by the client using disk bandwidth. Finally,

network interface bandwidth is required to transmit data into the network. The resources utilized by the server along the access path in order to satisfy a single media stream are collectively termed as "server channel".

Problem identified in server

Problem 1: Congested server channel
The allocated server channel may remain occupied over a varied length of time depending on the size of the media file. This allocation is typically long in terms of time because multimedia files are inherently large and their delivery takes time to complete. With increase in the demand, a server may need to support hundreds of thousands of requests at a given time. Indeed, if separate channels are dedicated for individual client requests, the channels at the disposal of the server will saturate very quickly.

Mohammad in [13] discovered that one approach to manage the predicament of channel saturation caused by increased popularity of multimedia streaming applications is to simply increase server capacity by building a server cluster using thousands of low cost desktop machines. This approach alone, however, cannot yield a desirable solution to the saturation problem of requests mentioned above, as it fails to address the potential bottlenecks that may incur at the server disk access bandwidth. So, it turns out that this is a temporary solution to a large resource problem.

Problem 2: Optimization of multicore
The backbone of any server is the number of CPUs that will power it, as well as the actual model and the type of the CPU [14]. From this point, there is need for the addition of Random Access Memory (RAM), storage and other options that the server's use case requires.

A Core is a physical part of a CPU. Cores act like processors within a single CPU chip. The more cores a CPU has, the more tasks it can perform simultaneously. One core can perform one task at a time while other cores handle other tasks the system assigns thereby improving the overall performance substantially when compared to old single-core CPUs. There are also logical cores that function as separate threads within a core. While they boost performance, logical cores (for hyper-threading) are not a match for physical cores [14]. Hyper-threading was reported in [15] that first appeared in February 2002 on Xeon server processors and in November 2002 on Pentium 4 desktop CPUs. Later, Intel included this technology in Itanium, Atom, and Core 'i' Series CPUs, among others. Hyper-Threading has been in Intel's processors for years. It's the main difference between the Core i5 first-gen and i7. However, many games do not use it and this explains why i5s power some serious gaming rigs. Also, there are many tasks and commands which are going to the processor through cores like dual-core, quad-core or octa-core and more but sometimes the commands are not processed through the cores properly which end up with the slow processing speed of the CPU [15]. The usage of the logical core in a server, especially multimedia server, is very expedient as this will help in the optimization and smooth service rendering by such a multimedia server. Hence, this is also a problem.

B. Network
Aside problems discovered at the server-side, issues are also in the network channel. When the server sends the requested media file into the network, it is routed to the client site that initiated the request for the file in the first place. The media file is transported to the requesting client over the network from the server in form of packets. These packets of the media data flow through the network and during this flow, resources such as, processing power and buffer space at each router, and, bandwidth on each link is used. The resources utilized by the data packets during their flow through the network are collectively termed as the "network channel".

Problem identified on network
The most prevalent method of streaming by current multimedia streaming applications through network is through the use of unicast, i.e. each stream holds a distinct network channel. In a typical scenario, where many users access the same 'popular' file at the same time, multiple replicas of the same data packets flow over the same links through distinct channels thereby wasting network resources. This may even cause bottlenecks in the network and saturate the network channel. The data packets would not seize the flow in a short period of time, as multimedia files are typically very large and their uninterrupted streaming may last from a couple of minutes to even a couple of hours. Hence, unicast streaming also causes a waste of server resources.

C. Client
At the client-side component of multimedia system, there are provision of special software which are used by client requesting a service from a multimedia streaming system [16]. Examples of these software include: Microsoft Media Player and RealPlayer. There are also some specialized hardware such as a set-top box (STB) which are used to playback the requested resource (multimedia file) on the client system. However, with this arrangement, there is still a problem.

Problem Identified at client side

The media file is typically composed of audio and video data and by nature has high storage and bandwidth requirements. The audio and video components of the multimedia files are usually compressed when stored or when they are delivered across the network. The compressed multimedia file needs to be decompressed at the client side before being rendered onto the screen. Video and audio data are bound by temporal limitations and are, by nature, delay-sensitive. This means that a particular piece of data belonging to a multimedia file must be received and displayed at a particular time in a fixed order. After passing of the time or the order, the received data becomes unusable. The current internet only supports best-effort service. This means that it cannot guarantee any end-to-end delay bound, or any stability in terms of delays. The unstable and unpredictable delay caused by network is called delay-jitter. This indirectly affects the client-side of the system.

V. PROPOSED SOLUTIONS

The major way by which these problems can be tackled is how to avoid server overloading. The proposed solutions to these challenges as stated in [3] are in four ways.

A. Determine the number of incoming streams and how it will be packaged

The central processing unit (CPU) and random access memory (RAM) are the two major hardware that affect server's processing capacity. To avoid overloads caused by limitations on processing capacity, there is need to know how many streams a server will have to process concurrently, and how much memory this will require.

B. Estimate Peak Concurrent Streams from Server

It is a difficult task to estimate the number of viewers for a streamed file, or the renditions those viewers will need. To avoid bandwidth-related overloads, an estimate of the peak outbound bandwidth of the server is needed to handle this.

C. Choose the Right Deployment Model

This involves whether to deploy the streaming of the files on a cloud infrastructure, or to deploy it on-premises using a "bare-metal" hardware server. Cloud deployment has efficient resource utilization and reduced overhead costs. However, it employs virtualization and orchestration management of software stacks—which adds additional workload to physical resources, including CPU, networking and RAM. When combined with common overprovisioning practices, this overhead reduces the load serviceable on a given virtual machine.

A bare-metal infrastructure, on the other hand, typically allows greater processing capacity. On a bare-metal server, utilization is up to 80 percent of total network bandwidth, and even higher percentages for CPU. In comparable virtualized and cloud deployments, useable CPU typically tops out around 65 percent, and accessible networking at around 50 percent [3]. A bare-metal configurations has the advantage of greater resource availability, the virtualized cloud environments are flexible, easy to use and offer self-service capabilities. For many people, these positives outweigh the negatives.

D. Use Graphics-Processing Unit (GPU) Offload and Content Delivery Networks (CDN)

Server's processing capacity can be improved through the use of infrastructures like GPU offload and CDNs. The use of graphics-processor offload of transcoder workloads can maximize processing power. When GPU scaling is used, users of both cloud and bare-metal configurations have successfully offloaded up to 75 percent of their CPU transcoding workload. GPU scaling reduces CPU consumption from 68 percent to 43 percent. In actual sense, some server cases, more than 90-percent reductions in primary CPU workload are possible.

Also, when streams are employed and pushed to a CDN, bottleneck with a single server is usually reduced. Using a CDN, it is possible to reduce the outbound streams to a single stream for each rendition.
Other solutions include:

E. Use of multicast for efficient delivery in network

The use of multicast has addressed the problem of unicast in that it efficiently deliver multimedia data packets from one server to multiple clients. Multicast can be employed inside the network, such as IP multicast, or deployed at the end hosts, as application layer multicast. It is known that network layer multicast uses network bandwidth very efficiently using a one-to-many communication model where during a streaming only a single server sends data to a large number of clients. Such applications motivate a new and simpler multicast service model called source-specific multicast [17]. As there is only a single sender in source-specific multicast, issues such as group management, pricing, routing, and security could also handled more easily and effectively in contrast to many-to-many type multicast.

As an approach to provide one-to-many along with many-to-many communication service, researchers have proposed application layer multicast technique [18, 19, 20, 21, 22], where the end hosts, instead of network layer routers, copy and forward data to their downstream hosts. This idea

implements a virtual network or an overlay network across the end hosts in the system. In other words, every virtual link from one host to another is a unicast path. A multicast distribution tree is created from server to all member nodes to provide a one-to-many multicast. After this, data is transmitted along the established distribution tree. During the transmission, the data is replicated and forwarded by the nodes at each branch point until it is received by every member node. This form of multicast (application layer multicast) does not require network support for one-to-many communication.

F. Applying start-up delay at client side

At client side, a start-up delay of possibly up to tens of seconds is usually introduced to accommodate very delay-jitter. This requires the need for a buffer to hold file data until it is played out.

G. Use of Admission Control Mechanism

Admission control mechanism for video-online-demand multimedia servers concern the acceptance decisions for new clients' requests so as to guarantee the continuous services to all clients within the multimedia system. This mechanism determines whether a new client's request can be accepted based on the consideration whether the underlying hardware can satisfy the quality of service (QoS) requirements of admitted requests. Different approaches have been proposed to achieve an admission control which can highly guarantee a quality of service in distributed multimedia systems. This approach can be implemented from any of the three major components of multimedia system to reduce the challenge of overloading.

VI. CONCLUSION

There are a number of basic problems that affect video streaming, the focus in this paper is on the case of video streaming over the Internet since it is an important, concrete example that helps to illustrate these problems. The peculiarity of video streaming from multimedia system involves that any data that is lost in transmission cannot be used at the receiver. Also, any data that arrives late is also useless. Precisely, any data that arrives after its decoding and display deadline is also too late to be displayed. Therefore, an important goal of video streaming is to perform the streaming in a manner so that this sequence of constraints is met. All these constraint could be as a result of the challenges posed by multimedia system's components. In this paper, the challenges have be highlighted and the observed solutions have been provided as well. Feature research should be looked into in the area of the implementation of admission control mechanism in the multimedia server to reduce the issue of overloading.

REFERENCES

[1]. W. Dapeng, T. H. Yiwei, Z. Wenwu, Z. Ya-Qin and M. P. Jon : "Streaming Video over the Internet: Approaches and Directions". *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 1, February 2001. http://www.ieee.org/organizations/pubs/transactions/tcsvt.htm

[2]. G. Barb. *The Difference Between Streaming and Downloading Media*. Lifewire, Dotdash publishing family, October 21, 2019. URL: https://www.lifewire.com/difference-between-streaming-and-downloading-media-1847372, Access 11/12/2019.

[3]. R. Lindsey."*4 Tips for Sizing Streaming Server Hardware*". Wowza Media Systems, LLC, June 8, 2017. Accessed on 16-05-2018. URL:https://www.wowza.com/blog/4-tips-for-sizing-streaming-server-hardware.

[4]. L. Rodriguez-Gil, P. Orduña and J. García-Zubia. *Multimed Tools Appl*, 77: 6471, 2018. https://doi.org/10.1007/s11042-017-4556-6

[5]. L. Rodriguez-Gil, P. Orduña, J. García-Zubia, I. Angulo and D. López-de-Ipiña. Graphic technologies for virtual, remote and hybrid laboratories: Weblab-fpga hybrid lab. In: *2014 11th international conference on remote engineering and virtual instrumentation* (REV). IEEE, pp 163–166, 2014.

[6]. W. VanLancker, D. VanDeursen, E. Mannens and R. VandeWalle. Implementation strategies for efficient media fragment retrieval. *Multimed Tools and Appl* 57(2):243–267, 2012.

[7]. J. Nielsen, *Usability engineering*. Elsevier, 1994.

[8]. S. Akhshabi, A. 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*, pp 157–168, 2011.

[9]. X. Hei, C. Liang, J. Liang, Y. Liu and K.W Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Trans Multimed* 9(8):1672–1687, 2007.

[10]. B. Wang, X. Zhang, G. Wang, H. Zheng and B. Zhao. Anatomy of a personalized livestreaming system. In: *Proceedings of the 2016 ACM on internet measurement conference*, pp 485–498, 2016.

[11]. R. Ravattu and P. Raj, "*Characterization of Youtube Video Streaming Traffic*", M.S. Thesis, Blekinge Institute of Technology, SOC, Electrical Eng, Sweden, June 2013. Retrieved on 16th November, 2019. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:830691/

[12]. L.C. Albert. "*Implementation of a streaming server*" (Master's thesis, University of Maribor, Slomškov trg 15, 2000 Maribor, Slovenia). 2011. Retrieved from https://upcommons.upc.edu/bitstream/handle/2099.1/13020/Final_Master_Thesis_Albert.pdf

[13]. A. S. Mohammad. "*Distributed Continuous Media Streaming- Using Redundant Hierarchy (RED-Hi) Servers*", (Unpublished doctoral thesis). Eastern Mediterranean University, Gazimağusa, North Cyprus, January 2014. Retrieved from http://i-rep.emu.edu.tr:8080/jspui/bitstream/11129/1316/1/ShahMohammad.pdf

[14]. Phoenixnap. "*Single Vs Dual Processor Servers, Which Is Right For You?*"PhoenixNAP | Global IT Services, posted February 20, 2019. Retrieved on 17th May, 2019.URL:https://phoenixnap.com/kb/single-vs-dual-processors-server.

[15]. Karan. "*Hyperthreading Vs No-Hyperthreading: Which CPU is better*"? Published: Tuesday, May 29, 2018, 16:15. Accessed on 17-05-2019.URL:https://www.gizbot.com/computer/features/hyperthreading-vs-no-hyperthreading-which-cpu-is-better/articlecontent-pf89242-050938.html

[16]. Microsoft. *Microsoft Windows Media* 9 series, July 2004. Retrieved on 29th May, 2017. URL: http://www.microsoft.com/windows/windowsmedia/default.aspx.

[17]. S. Bhattacharyya, C. Diot, L. Giuliano, R. Rockell, J. Meylor, D. Meyer and B. Haberman. "*An overview of source-specific multicast (SSM)*". RFC 3569, July 2003, Retrieved on 21st August, 2018.

[18]. Y. Chawathe, S. McCanne and E. Brewer. "*An Architecture for Internet Content Distribution as an Infrastructure Service". Unpublished work*, February 2000. http://yatin.chawathe.com/~yatin/papers/scattercast.ps. Retrieved on 21st August, 2018.

[19]. Y. H. Chu, S. G. Rao, S. Seshan and H. Zhang. "*A case for end system multicast*", 2002. Retrieved 20th April, 2018.

[20]. J. Jannotti, D. K. Gifford, K. L. Johnson and M. F. Kaashoek." Overcast: reliable Multicasting with on overlay network". In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4* (pp. 14-14), 2000. Usenix Association.

[21]. D. E. Pendarakis, S. Shi, D.C. Verma and M. Waldvogel. "An Application Level Multicast Infrastructure (ALMI)". In USITS (Vol. 1, pp. 5-5), 2001.

[22]. S. Q. Zhuang, B.Y Zhao, A. D. Joseph, R.H. Katz and J. D. Kubiatowicz." Bayeux: An Architecture for scalable and fault-tolerant wide-area data dissemination". In *Proceedings of the 11th international workshop on Network and operating systems Support for digital audio and video* (pp. 11-20). ACM, 2001.