www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume – 4 Issue - 12 December, 2015 Page No. 15274-15280

Http Streaming Based Ajax Player

Hina Rani[1], Khushboo Bansal[2]
[1]Dept. of Computer Science engineering, Desh Bhagat University , Mandi Gobindgarh ,India
hinu30.mittal@gmail.com
[2]Dept. of computer science *engineering*, Desh Bhagat University, Mandi Gobindgarh, India
bansal_khushboo@gmail.com

***Abstract*:** In the previous couple of years streaming of video on the web has encountered quick development and will keep on expanding in significance as broadband innovations and authoring tools keep on making strides. As the internet turns into an inexorably famous alternative to traditional communications media, internet streaming will turn into a critical segment of numerous content providers' communications strategy. In this paper we proposed a solution to HTTP live streaming, which assesses the weights of media segments to choose the transmitting needs taking into account the present playing time and alter the proper transmission path.

Keywords: Streaming, HTTP, Video, Adaption, Ajax, Dynamic

I. Introduction

The internet broadband revolution is likely to significantly change the way that we interact with computers and the internet as a whole. Internet streaming is expected to play an increasingly important role in an on-line world with high bandwidth connections. However even when end-users have high-bandwidth connections to the Internet, the problem of distributing the content to them will be a limiting factor for any content provider that wants to reach that audience.

Video traffic is becoming the dominant share of Internet traffic today [5]. This growth in video is accompanied, and in large part driven, by a key technology trend: the shift from customized connection-oriented video transport protocols (e.g., RTMP [9]) to HTTP-based adaptive streaming protocols (e.g., [10-13]).

With an HTTP-based adaptive streaming protocol, a video player can dynamically (at the granularity of seconds) adjust the video bit rate based on the available network bandwidth. As video traffic is expected to dominate Internet traffic [5], the design of robust adaptive HTTP streaming algorithms is important not only for the performance of video applications, but also the performance of the Internet as a whole. Drawing an analogy to the early days of the Internet, a robust TCP was critical to prevent "congestion collapse" [15]; we are potentially at a similar juncture today with respect to HTTP streaming protocols. Building on this high-level analogy, it is evident that the design of a robust adaptive video algorithm must look beyond a single player view to account for the interactions across multiple adaptive streaming players [16] that compete at bottleneck links. In this respect, there are three (potentially conflicting) goals that a robust adaptive video algorithm must strive to achieve:

• Fairness: Multiple competing players sharing a bottleneck link should be able to converge to an equitable allocation of the network resources.

• Efficiency: A group of players must choose the highest feasible set of bitrates to maximize the user experience.

• Stability: A player should avoid needless bitrate switches as this can adversely affect the user experience. Recent measurements show that two widely used commercial solutions fail to achieve one or more of these properties when two players compete at a bottleneck link [17]. We extend these experiments (Section 2) and confirm that the problems manifest across many state-of-art HTTP adaptive streaming protocols: Smooth Streaming [12], Netflix [18], Adobe OSMF [7], and Akamai HD [8]. Furthermore, these problems worsen as the number of competing players increases.

II. Design Principles

HTTP-based progressive download does have significant market adoption. Therefore, HTTP-based streaming should be as closely aligned to HTTP-based progressive download as possible, but take into account the above-mentioned deficiencies.

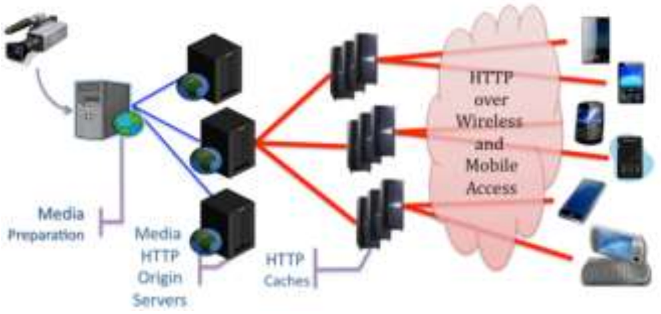

HTTP over Wireless and Mobile Access
Media Preparation
Media HTTP Origin Servers
HTTP Caches

Figure 2 shows a possible media distribution architecture for HTTP-based streaming. The media preparation process typically generates segments that contain different encoded versions of one or several of the media components of the media content. The segments are then hosted on one or several media origin servers typically, along with the media presentation description (MPD). The media origin server is preferably an HTTP server such that any communication with the server is HTTP-based (indicated by a bold line in the picture). Based on this MPD metadata information that describes the relation of the segments and how they form a media presentation, clients request the segments using HTTP GET or partial GET methods. The client fully controls the streaming session, i.e., it manages the on-time request and smooth play out of the sequence of segments, potentially adjusting bitrates or other attributes, for example to react to changes of the device state or the user preferences. Massively scalable media distribution requires the availability of server farms to handle the connections to all individual clients. HTTP-based Content Distribution Networks (CDNs) have successfully been used to serve Web pages, offloading origin servers and reducing download latency. Such systems generally consist of a distributed set of caching Web proxies and a set of request redirectors. Given the scale, coverage, and reliability of HTTP based CDN systems, it is appealing to use them as base to launch streaming services that build on this existing infrastructure. This can reduce capital and operational expenses, and reduces or eliminates decisions about resource provisioning on the nodes. This principle is indicated in Figure 2 by the intermediate HTTP servers/caches/proxies. Scalability, reliability, and proximity to the user's location and high-availability are provided by general purpose servers. The reasons that lead to the choice of HTTP as the delivery protocol for streaming services are summarized below:

1. HTTP streaming is spreading widely as a form of delivery of Internet video.

2. There is a clear trend towards using HTTP as the main protocol for multimedia delivery over the Open Internet.

3. HTTP-based delivery enables easy and effortless streaming services by avoiding NAT and firewall traversal issues.

4. HTTP-based delivery provides reliability and deployment simplicity due as HTTP and the underlying TCP/IP protocol are widely implemented and deployed.

5. HTTP-based delivery provides the ability to use standard HTTP servers and standard HTTP caches (or cheap servers in general) to deliver the content, so that it can be delivered from a CDN or any other standard server farm.

6. HTTP-based delivery provides the ability to move control of "streaming session" entirely to the client. The client basically only opens one or several or many TCP connections to one or several standard HTTP servers or caches.

7. HTTP-based delivery provides the ability to the client to automatically choose initial content rate to match initial available bandwidth without requiring the negotiation with the streaming server.

8. HTTP-based delivery provides a simple means to seamlessly change content rate on-the-fly in reaction to changes in available bandwidth, within a given content or service, without requiring negotiation with the streaming server.

9. HTTP-based streaming has the potential to accelerate fixed mobile convergence of video streaming services as HTTP based CDN can be used as a common delivery platform

III. QUALITY OF EXPERIENCE IN HTTP VIDEO STREAMING

HTTP video streaming (video on demand streaming) is a combination of download and concurrent

DOI: 10.18535/Ijecs/v4i12.32

playback. It transmits the video data to the client via HTTP where it is stored in an application buffer. After a sufficient amount of data has been downloaded (i.e., the video file download needs not to be complete yet), the client can start to play out

the video from the buffer. As the video is transmitted over TCP, the client receives an undisturbed copy of the video file. However, there are a number of real world scenarios in which the properties (most importantly instantaneous throughput and latency) of a communication link serving a certain multimedia service are fluctuating. Such changes can typically appear when communicating through a best effort network (e.g., Internet) where the networking infrastructure is not under control of an operator from end to end, and thus its performance cannot be guaranteed. Another example is reception of multimedia content through a mobile channel, where the channel conditions are changing over time, due to fading, interferences, and noise. These network issues (e.g., packet loss, insufficient bandwidth, delay, and jitter) will decrease the throughput and introduce delays at the application layer. As a consequence, the playout buffer fills more slowly or even depletes. If the buffer is empty, the playback of the video has to be interrupted until enough data for playback continuation has been received. These interruptions are referred to as stalling or rebuffering. In telecommunication networks, the Quality of Service (QoS) is expressed objectively by network parameters like packet loss, delay, or jitter. However, a good QoS does not guarantee that all customers experience the service to be good. Thus, Quality of Experience (QoE) – a concept of subjectively perceived quality – was introduced [6]. It takes into account how customers perceive the overall value of a service, and thus, relies on subjective criteria. For HTTP video streaming, [4, 7] showed in their results that initial delay and stalling are the key influence factors of QoE. However, changing the transmitted video quality as employed by HTTP adaptive streaming introduces a new perceptual dimension.

IV. Literature Review

In [2], author presented a principled understanding of bit rate adaptation and analyze several commercial players through the lens of an abstract player model. Through this framework, they identify the root causes of several undesirable interactions that arise as a consequence of overlaying the video bit rate adaptation over HTTP. Building on these insights, they develop a suite of techniques that can systematically guide the tradeoffs between stability, fairness and efficiency and thus lead to a general framework for robust video adaptation. We pick one concrete instance from this design space and show that it significantly outperforms today's commercial

players on all three key metrics across a range of experimental scenarios.

Merwe et al [19] and Cherkasova and Gupta[20] also present characterizations of streaming video traffic and show that various parts of a clip have different probabilities of being viewed. Thus they conclude that content segmentation and caching of selective segments is more cost effective and offers better performance than caching of whole media files. While our system deals with Video On Demand content in ways similar to those described by these studies, the focus of this paper is live streams that obviously do not lend themselves to caching.

Junchen Jiang et.al [21], the growth of Internet video and the role that video quality plays in user engagement (and thus revenues) has sparked a renewed interest in redesigning various aspects of the content delivery ecosystem ranging from video players, CDNs, multi-CDN optimizations, and global control planes. In this paper an initial attempt to bridge this gap. We find, perhaps surprisingly, that a small number of potential problem causes can account for a large number of problem sessions. Furthermore, these problem causes are amenable to simple solutions, either via using offline traces to identify the sources of these problems or by reacting only to long-lasting outages. We believe that these observations bodes well for the Internet video ecosystem going forward as many of the aforementioned efforts to improve video quality could be simplified to achieve the same benefits.

Leonidas Kontothanassis et.al [22], in this paper we have discussed the design decisions we have made while building a content delivery network for live streaming. We have described how to achieve high degrees of scalability, quality, and reliability by focusing on modular design and eliminating single points of failure. We have evaluated multiple techniques for delivering data to edge servers before deciding on a combination of retransmits and multiple paths as our approach of choice.

Furthermore we have shown that delivery of stream data at rates higher than the encoded rate for the first few seconds of a session, can significantly improve an end user's quality. Finally we have introduced a mechanism for eliminating single points of failure at the entry points of our system. The described system currently serves millions of streams per day to end users across the world, and has scaled to 80,000+ concurrent users and 16 gigabits per second of traffic. Despite the success of the developed system a number of issues remain as interesting technical questions. We would like to determine whether the reflector hierarchy itself can be bypassed altogether for unpopular streams and how the system would have to be modified to handle the transition of a stream from the unpopular to the popular category and viceversa. It would also be interesting to have edge regions choose their parent set reflectors in a completely dynamic fashion and not have to rely on bucketing techniques for load balancing. The multipath transmission system can potentially benefit from modifications that would allow it to pick the best path amongst its choices, rather than the number of paths necessary to provide good quality. Finally the fault tolerant system can be further tuned to ensure fault recovery transitions go completely unnoticeable by end users. Those questions notwithstanding, the existing system offers tremendous benefits over both centralized and naive distributed CDN implementations, and we believe it is a good compromise between engineering and operations cost, and customer benefit.

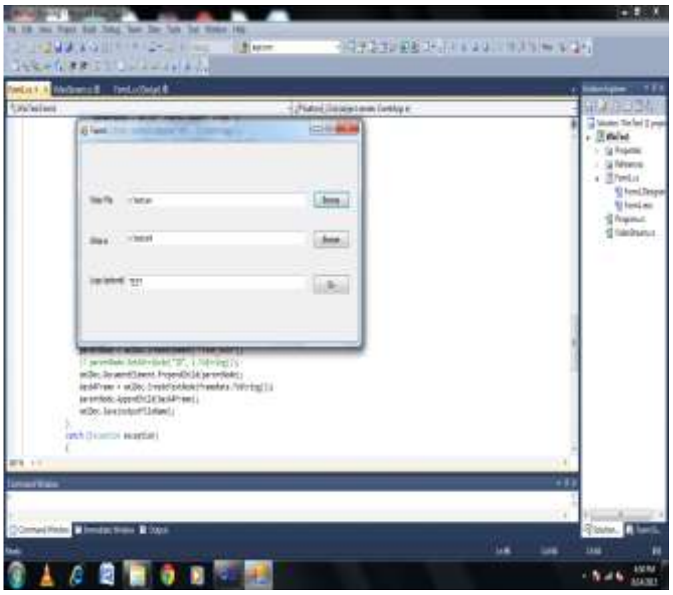
Thomas Stockhammer et.al [10], in this paper, we provide some insight and background into the Dynamic Adaptive Streaming over HTTP (DASH) specifications as available from 3GPP and in draft version also from MPEG. Specifically, the 3GPP version provides a normative description of a Media Presentation, the formats of a Segment, and the delivery protocol. In addition, it adds an informative description on how a DASH Client may use the provided information to establish a streaming service for the user. The solution supports different service types (e.g., On-Demand, Live, Time-Shift Viewing), different features (e.g., adaptive bitrate switching, multiple language support, ad insertion, trick modes, DRM) and different deployment options. Design principles and some forward-looking considerations are provided.

V. Proposed Methodology

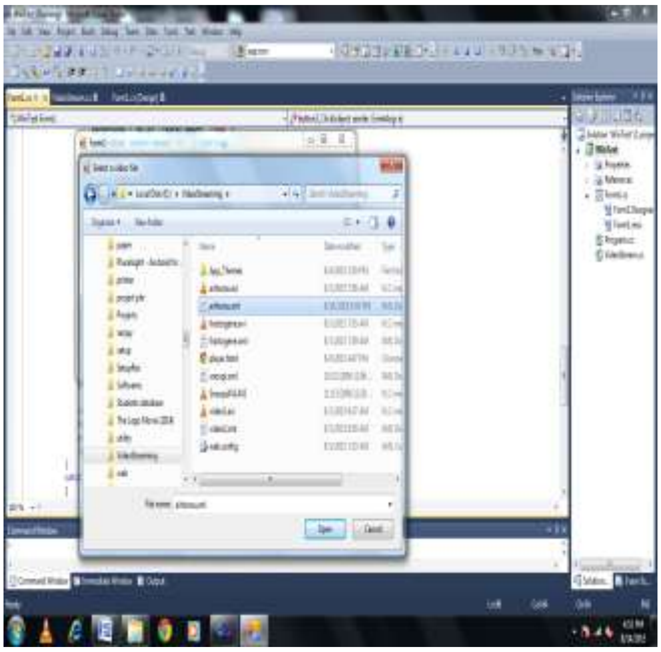
Step1: Read video file.
Step2: define frame rate, clip size, stream length.
Step 3: extract all property of video file.
Step 4: Convert video file into Frames.
Step 5: Convert Frame into Bitmap.
Step 6: Convert bitmap into XML.
Step 7: Write XML file into hard disk.
Step 8: Make Ajax Video player.
Step 9: XML file play in Ajax video player.
Step10: Process done.

VI. Experiment Result

1) Run Wintest

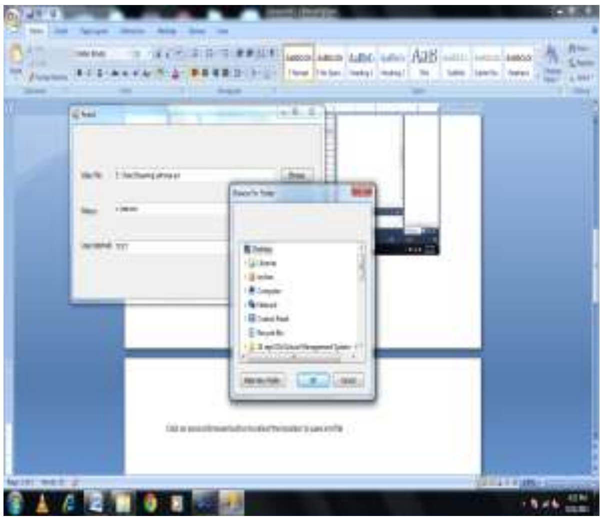


2) Click on first browse button to select avi file PAPER.

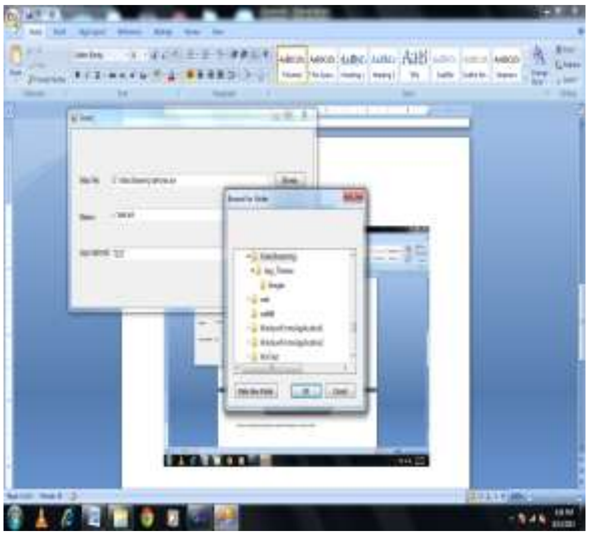


DOI: 10.18535/Ijecs/v4i12.32

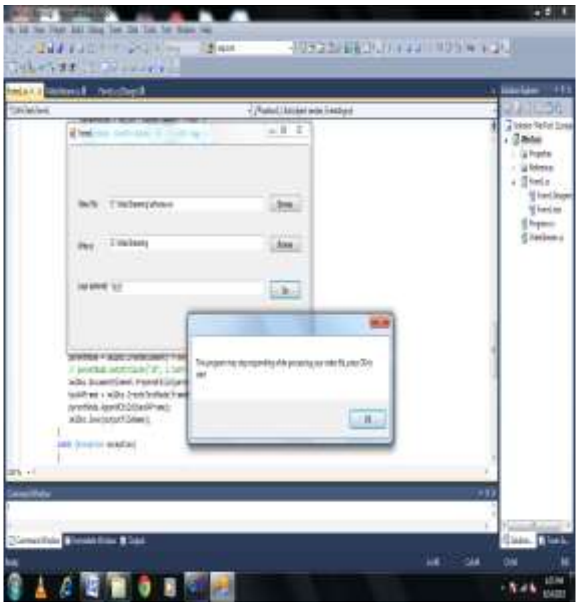
3) Click on second browse button to select the location to save xml file



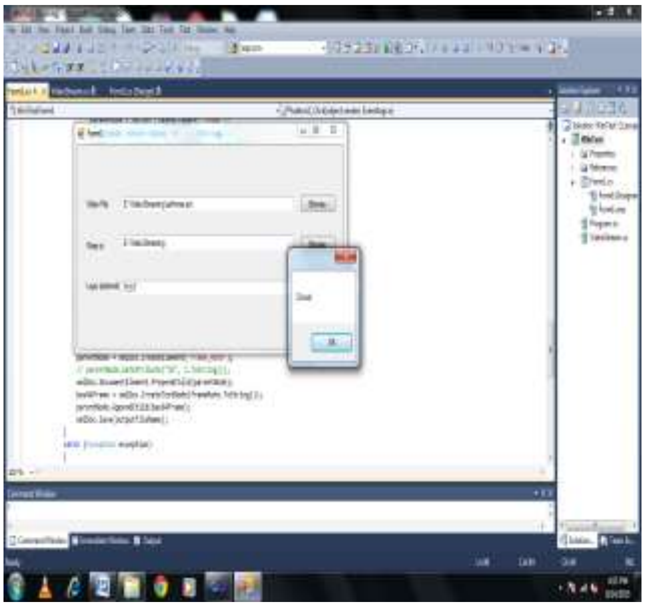
4) Shows the selected folder



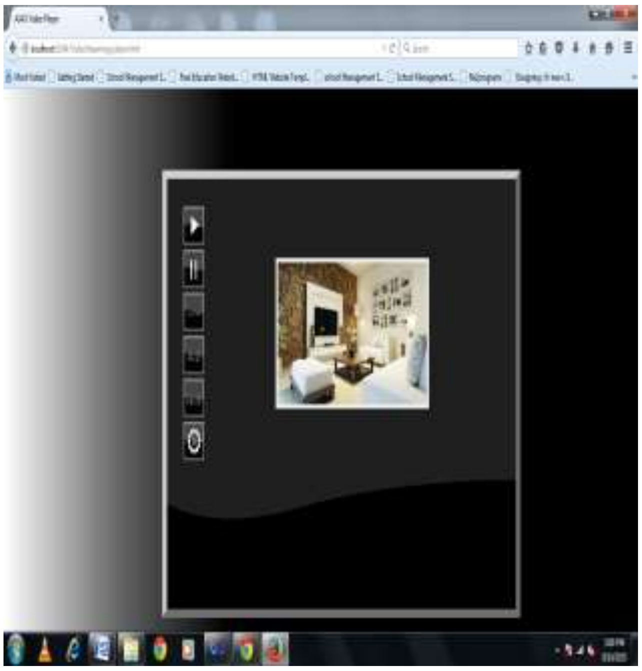
5) Then click on go button to get xml file



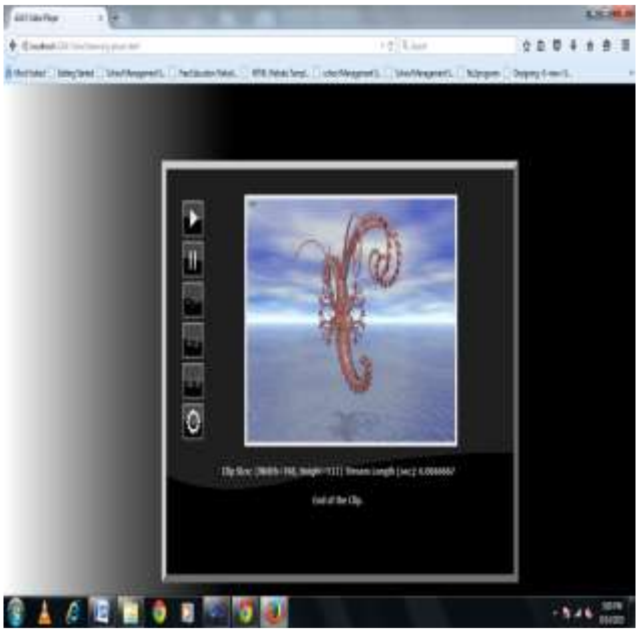
6) Process completed now



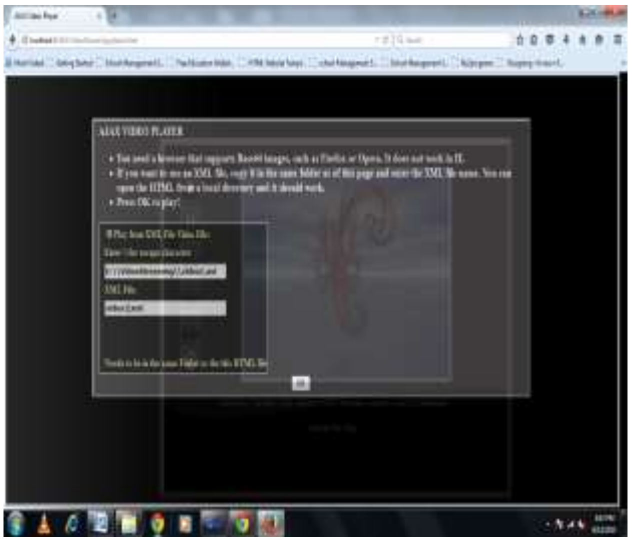
7) This process is same for all avi files
8) Run the website in Mozilla Firefox.



9) After playing Video



10) Click on setting button to change the avi name and xml name.



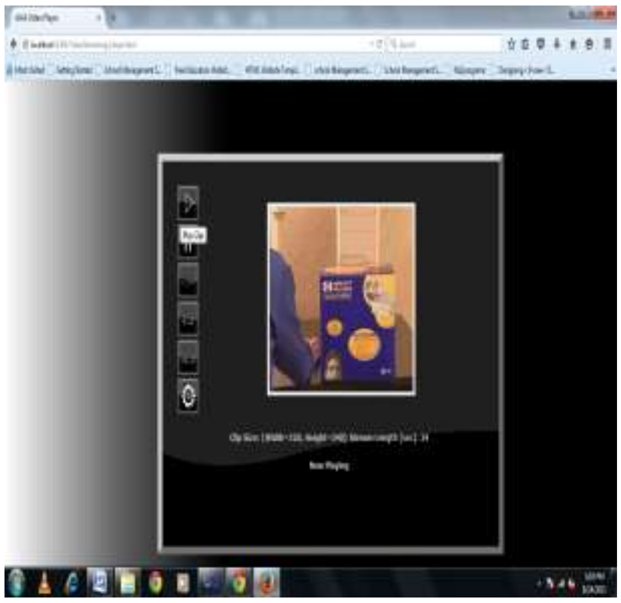
AJAX VIDEO PLAYER

- You need a browser that supports Base64 images, such as Firefox or Opera. It does not work in IE.
- If you want to see an XML file, copy it in the same folder as of this page and enter the XML file name. You can open the HTML from a local directory and it should work.
- Press OK to play!

Play from XML File Video file:

XML File:

Needs to be in the same folder as the this HTML file



Conclusion

Streaming of video on the internet has experienced rapid growth and will continue to increase in importance as broadband technologies and authoring tools continue to improve. In this paper, we proposed an approach for streaming media file for which convert avi file into Xml. We created Ajax video player for Xml file, because the speed of Xml file is far better from any player. Experiment results showed that proposed approach work well over previous method.

References

[1] Leonidas Kontothanassis, Ramesh Sitaraman, Joel Wein, Duke Hong, Robert Kleinberg, Brian Mancuso David Shaw, and Daniel Stodolsky "A Transport Layer for Live Streaming in a Content Delivery Network" Proceedings Of The IEEE.

[2] Junchen Jiang, Vyas Sekar and Hui Zhang "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE" CoNEXT'12, December 10-13, 2012, Nice, France. Copyright 2012

[3] Adobe http dynamic streaming. www.adobe.com/products/hds-dynamic-streaming.html.

[4] Cisco forecast http://goo.gl/hHzW4.

[5] Real-time messaging protocol. www.adobe.com/devnet/rtmp.html.

[6] Smooth streaming experience. http://www.iis.net/media/experiencesmoothstreaming

[7] Adobe osmf player. http://www.osmf.org.

[8] Akamai hd adaptive streaming. http://wwwns.akamai.com/hdnetwork/demo/index.html

[9] H. Parmar, M. Thornburgh (eds.) Adobe's Real Time Messaging Protocol, Adobe, December 21, 2012.

[10] T Stockhammer, "Dynamic adaptive streaming over HTTP--: standards and design principles", Proceedings of the second annual ACM conference on Multimedia systems, pp 133-144, 2011.

[11] J Jiang, V Sekar, H Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive", 8th international conference on Emerging networking experiments and technologies, pp. 97-108, 2012 .

[12] Smooth streaming experience, http://www.iis.net/media/experiencesmoothstreaming.

[13] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. IEEE Multimedia, 2011.

[14] R. Pantos. Http live streaming. 2011.

[15] V. Jacobson. Congestion avoidance and control. In ACM SIGCOMM Computer Communication Review, volume 18, pages 314–329. ACM, 1988.

[16] J. Esteban, S. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac. Interactions between HTTP Adaptive Streaming and TCP. In Proc. NOSSDAV, 2012.

[17] R. Houdaille and S. Gouache. Shaping http adaptive streams for a better user experience. In Proc. MMSys, 2012.

[18] Mail service costs Netflix 20 times more than streaming. http://goo.gl/msuYK.

[19] J. van der Merwe, S. Sen, and C. Kalmanek, “Streaming video traffic: Characterization and network impact,” in Workshop on Web Content Caching and Distribution (WCW), Boulder, CO, August 2002.

[20] L. Cherkasova and M. Gupta, “Characterizing locality, evolution, and life span of accesses in enterprise media server workload,” in NOSSDAV’02, Miami Beach, FL, May 2002.

[21] Junchen Jiang, Vyas Sekar, Ion Stoica, Hui Zhang, “Shedding light on the structure of internet video quality problems in the wild”, Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, pp. 357-368, 2013.

[22] Leonidas Kontothanassis, Chris Joerg, Michael J. Swain, Brian Eberman, Robert A . Iannucci, “Design, Implementation, and Analysis of a Multimedia Indexing and Delivery Server”, Cambridge Research Laboratory, 1999.