Video Splicing Techniques for P2P Video Streaming

MD Iftakharul Islam and Javed I Khan
Kent State University
Email: mislam4@kent.edu, javed@cs.kent.edu

***Abstract*—In HTTP live streaming (HLS), a video is spliced into multiple segments of equal duration. The size of the segments has a major impact on the stalls. In adaptive video streaming, clients adapt the bit-rate of the video to minimize the stalls. But it often degrades the video quality. In order to maintain a high video quality, we can adapt the duration of the segments instead of the bit-rate. Thus the splicing technique can have a major impact on a video streaming.**

In this paper, we studied different video splicing techniques for peer-to-peer video streaming. We spliced MPEG-4 video based on different durations and Group of Picture (GOP). We found that the duration based splicing performs better than the GOP based splicing, though the duration based splicing requires much more data to be transferred than the GOP based splicing. In P2P video streaming, peers can leave the swarm anytime. To maximize the availability of a segment, peers often download multiple segments simultaneously. Again, if a peer downloads too many segments at a time, it will increase stalls. In this paper, we proposed a formula to calculate the number of video segments a peer should download simultaneously in order to reduce the stalls. We conducted the experiment on GENI.

***Index Terms*—P2P Video Streaming; Video Splicing, GENI**

I. Introduction

Video streaming is a major part of Internet traffic. Netflix and Youtube account for more than half of the network traffic in the United States [1]. Cisco also predicted that Internet video will be 79% of all consumer Internet traffic by 2018. Though most video streaming techniques use a Content Delivery Network (CDN), peer-to-peer architecture is still being used for scalability, fault-tolerance, and anonymity. P2P Internet TVs are very popular in some countries in Asia. PPStream [2], SopCast [3], PPLive [4], etc are some of the most popular P2P video streaming services. But it is difficult to study those large scale P2P systems for their proprietary nature. This paper studies different video splicing techniques for P2P video streaming. We conducted the experiment on GENI. We will demonstrate the video streaming on GENI.

HTTP and TCP are widely being used as a transport mechanism for P2P video streaming services. In HLS [5], [6], a video is spliced into multiple segments of equal duration. For example, Netflix and Hulu splice the videos into 4-second and 8-second segments respectively [7]. Their clients determine a bit-rate based on the available bandwidth. As they keep the duration of the segment constant and vary the bit-rates, it will degrade the video quality when the bandwidth becomes low or the network becomes congested (due to competing flows). Instead of varying the bit-rate, we can vary the segment duration. In this way, we can adapt the segment size to avoid stalls without degrading the video quality. This paper studies the effect of splicing techniques on the stalls. We conducted the experiment on a MPEG-4 [8] video. MPEG-4 is well suited for video streaming and is supported by almost all the major browsers and devices. MPEG-4 is a sequence of GOP (Group of Pictures). A GOP is a sequence of frames. GOP starts with an I-frame. The rest of the GOP consists of P and B frames. An I-frame is an independent frame, which can be decoded independently. On the other hand, P and B frames are dependent on the I-frame. In this way, a GOP is an independent segment of video that can be played independently. We spliced the video based on GOP. The main benefit of GOP based splicing is there is no overhead. On the other hand, in the duration based splicing, an extra I-frame may need to be inserted for each segment. As the size of an I-frame is significantly larger than P and B frames, this approach requires much more data to be transferred. Though the duration based splicing has major overhead, we found that it performs better than GOP based splicing for P2P video streaming.

In P2P video streaming, a peer can download multiple segments simultaneously. As peers can leave a swarm anytime, downloading a segment beforehand increases the chance of the availability of a segment. It also increases the bandwidth utilization. Again downloading too many segments may increase stalls as well. The optimum number of segments to download depends on the segment size, available bandwidth, and the duration of the video that the peer already has downloaded. In this paper, we propose an equation to calculate the optimum number of segments such that stalls are not increased.

As the quality of video streaming highly depends on the available bandwidth, we varied bandwidth during the experiment. We conducted our experiment on GENI. GENI [9] is a United State wide networking test-bed. One can create a virtual network and deploy his applications on the virtual nodes of GENI. This is why GENI is a very suitable platform for experimenting with P2P applications. Our experimental result on GENI shows that the duration based segmentation performs better than the GOP based segmentation.

The main contributions of this paper are as following:

- We described the pros and cons of different video splicing techniques for peer-to-peer video streaming (Section 2).
- We proposed a formula to calculate the number of segments a peer should download simultaneously in order to reduce the stalls. The formula also gives us an idea about the segment size (Section 3 and 4).
- We conducted the video streaming on GENI. We will demonstrate the video streaming in our presentation. The experimental result gives us an insight into different splicing techniques.

II. VIDEO SEGMENTATION

In this section, we will discuss different video splicing techniques. We will describe the advantages and disadvantages of each of them.

A. GOP Based Segmentation

Group of Picture (GOP) is a group of I-frame, P-frame, and B-frame. Each GOP can be played independently. The advantage of GOP based splicing is that there is no redundancy among the segments. This makes GOP based splicing an ideal candidate. But we found that the GOPs can be very big, so it can cause stalls. GOPs can be divided into two categories: Open GOP and Closed GOP. In Open GOP, one GOP can be dependent on another. Here we are only dealing with closed GOP where each GOP is completely independent.

B. Duration Based Segmentation

Here video is sliced based on a duration. This splicing is frame-accurate. Thus every segment can be played independently. The main disadvantage of this approach is an I-frame is inserted at the beginning of each segment. As the size of an I-frame is significantly larger than P and B frames, if a video is spliced into many very small segments, the total size of the video increases significantly.

III. DOWNLOADING POLICY

In P2P systems, peers usually download multiple segments simultaneously based on the expected bandwidth among the peers. Though video streaming is a sequential process, downloading the segments ahead of time reduces stalls. Besides peers can leave the swarm any time. That is why, it is a good idea to download a segment ahead of time when the segment is available. Again, if a peer downloads too many segments at the same time, it may cause stalling. So we need to find out the optimal number of segments that a peer should download simultaneously, in order to maximize the bandwidth utilization without causing stalling.

For example, a peer has buffered through segment N and is downloading the next k segments, $N+1, N+2, ..., N+k$ simultaneously. However, if segment $N+1$ has not been fully downloaded by the time currently buffered segments are played (T seconds), a stall will occur. In order to avoid the stall, all the k segments have to be downloaded by T seconds. This is because the peer may download different segments from different peers and all the downloads are sharing some resources, so we cannot determine the order of the downloads of each segment ahead of time. Let's assume that the size of each segment is W bytes. The available bandwidth among the peers is B B/s. The peer will calculate the number of segments using the following equation.

$$\text{Number of segments to download } \leq \max\left(\frac{B*T}{W}, 1\right) \quad (1)$$

At the beginning of streaming or if the peer is already stalled or has just finished playing all the buffered video, $T = 0$ second. In those situations, a peer will always download only one segment. Again, if T is very small, $\frac{B*T}{W}$ will be less than one. The peer will download only one segment in that case as well. In the rest of the cases, peer will download at most $\frac{B*T}{W}$ segments.

To avoid a stall, the peer has to download all the segments in the downloading pool within T seconds. Otherwise the peer may have played the rest of the video in the buffer, but the next segment has not been downloaded. The peer can download $B*T$ bytes in T seconds. As the size of the segment is W bytes, the peer can download $\frac{B*T}{W}$ segments in T seconds. This is the maximum number of segments a peer will download simultaneously. There are many ways to calculate the available bandwidth in the real-world scenario. [10] showed a method to calculate the expected bandwidth among the peers in a P2P video streaming. In that paper, they determined the expected bandwidth based on the packet inter-arrival time, round-trip delay, packet-loss, and so on. But in this paper, we simulated the bandwidth on GENI.

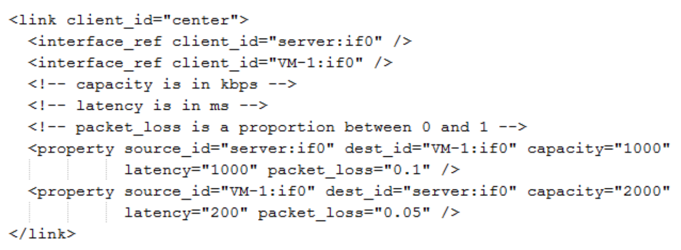
IV. SEGMENT SIZE

In our experiment, we found that downloading one large segment is faster than downloading multiple smaller segments. If we increase the size of the segment, the number of segments to download decreases according to (1). So, keeping the segment large and the number of segments smaller increase the total throughput. But larger segments also increase the load on each peer as they have to upload the segments. So, the size of the segment has to be small enough such that it does not overload the uploading peers. At the same time, the size of the segment should be large enough such that network throughput is not decreased.

In traditional video streaming, a Content Delivery Networks (CDN) is used for serving content. Many of the P2P video streaming services adopted hybrid architecture where contents are served by peers as well as a CDN. When a video is served by a CDN, peers can download one segment at a time instead of downloading mutiple segments in order to increase the bandwidth utilization. In that case, the maximum size of the segment will be $B*T$ bytes according to (1). In this way, we can adapt the segment size to increase the network throughput as well as keep the load minimum on each peer without causing a stall.

V. EXPERIMENTAL SETUP

We developed the P2P video streaming application in Java. The application itself works as a seeder and a leecher. We conducted the experiment on GENI [9]. GENI enables us to create a virtual network connecting virtual hosts on the data centers located in different campus networks across the United States. We used RSpec [11] to create the virtual network. RSpec also enabled us to deploy the necessary software packages on every virtual host. These capabilities made GENI



```
<link client_id="center">
  <interface_ref client_id="server:if0" />
  <interface_ref client_id="VM-1:if0" />
  <!-- capacity is in kbps -->
  <!-- latency is in ms -->
  <!-- packet_loss is a proportion between 0 and 1 -->
  <property source_id="server:if0" dest_id="VM-1:if0" capacity="1000"
            latency="1000" packet_loss="0.1" />
  <property source_id="VM-1:if0" dest_id="server:if0" capacity="2000"
            latency="200" packet_loss="0.05" />
</link>
```

Fig. 1. RSpec snippet of a link

a very convenient platform for experimenting with peer-to-peer applications. We conducted the experiment on twenty nodes. A simple RSpec snippet is shown in Figure 1. Here we showed how we can simulate different network properties among the virtual nodes. The nodes are connected in a star topology using another virtual node. We varied the bandwidth among nodes during the experiment. Here all the nodes are XEN virtual machines running Ubuntu 64-bit operating system. One of the nodes is a seeder here. The seeder slices the video into multiple segments. It slices the video based on GOP or duration according to the configuration.

We developed the application in Java. Developing the application in Java enabled us to be productive as well as to experiment across multiple platforms. We used Xuggler [12] for manipulating the MPEG-4 videos. Xuggler is a library based on FFmpeg [13]. We also used vlcj [14] for video playback. vlcj is a library based on LibVLC [15]. The application uses Java socket as the transport mechanism. We implemented our own BitTorrent like messaging protocol. We experimented with a 1 Mbps (128kB/s) MPEG-4 video of 2 minutes. We deployed our application on a InstaGENI rack. The nodes of GENI do not come with an X-window system. In order to play the video on the VMs of GENI, we had to install Unity desktop and VNC server on each node. We automated most part of the experiment using RSpec. Some of the applications require users to accept the terms and conditions during installation. We could not install those application using Rspec. We installed those applications manually on each node. We measured the total number of stalls, total stall duration, and startup time by varying the bandwidth among the peers.

VI. EXPERIMENTAL RESULTS

In this section, we present the experimental results of the video slicing techniques. We measured the total number of stalls, total stall duration, and startup time for different types of splicing. We also present the effect of the downloading policy on the stall count.

A. *Comparison between Splicing Techniques*

We measured the total number of stalls for the video for GOP and duration based splicing. Here we varied the bandwidth among the peers and kept the latency 50 milliseconds and the packet loss 5%. The result is shown in Figure 2. Here we assumed that users are viewing the video sequentially. Research has shown that 95% of users of a P2P TV watch video sequentially [16]. The video is two minutes long. We ran the application three times for each bandwidth and took the rounded average. Figure 2 shows that the GOP based splicing causes more stalls than the duration based splicing. The duration of the GOPs can vary based on the content of the video. For example, if a video contains constantly changing scenery, the duration of the GOP will be very short. If a video contains a stationary scene or a scene where an object is moving slowly, the duration of the GOP can be very long. In GOP based splicing, long and short GOPs result very large and small segments respectively. So, the size of a long GOP can be significantly larger than the size of a short GOP. Very large segments cause stalls in a video streaming because the peer may not have something to play while downloading a large segment. Again very small segments reduce network throughput. In this way, GOP based splicing causes stalls and reduces network throughput. On the other hand, duration based splicing splices the video into equal-duration segments. Here we spliced the video into two, four, and eight seconds segments. Duration based splicing creates segments which are neither too small nor too big. Thus it causes less stalls than GOP based splicing. We found that the 2-second segments cause more stalls than 4-second segments when the available bandwidth among the peers is small. This is because it creates many small TCP connections that create congestion in the network. As we increase the bandwidth, 2-second segments do slightly better and perform similar to 4-second segments. Again 8-second segments results larger segments, thus results more stalls than the smaller segments.

Figure 3 compares the total duration of the stalls for different types of splicing. It is clear from the figure that the GOP based splicing results in a longer stall. Again a smaller segment results in a shorter stall. Note that the total number of stalls for the smaller segments can outnumber the total number of stalls for the larger segments, but the total stall duration time is shorter for the shorter segments.

Figure 4 compares the startup time among different duration based splicings. Startup times of GOP based splicing are different for different videos. Here we compared among 2, 4, and 8-second segments. At the beginning, each peer contacts the seeder and gets different information about the video and the swarm. Here we considered that latency between seeder and peer is 500 milliseconds. Latency among the peers is 50 milliseconds. The figure shows that the large segments can result in a very high startup time in a low bandwidth network.

B. *Effect of Downloading Policy*

In this section, we measured the effect of the downloading policy that we described in (1). We compared the our policy to the fixed size pooling where the application downloads a fixed number of segments at a time. Here the application maintains a downloading pool. We conducted the experiment by varying the size of the downloading pool. Figure 5 shows

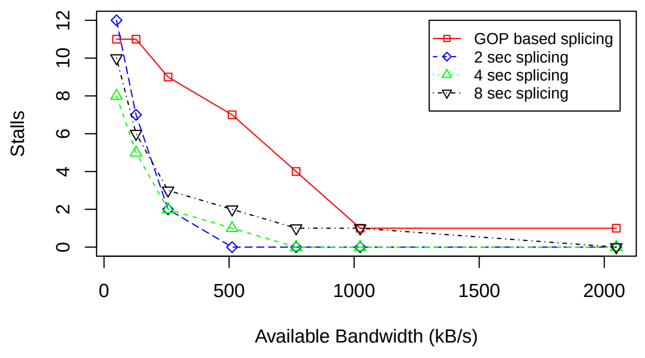


Fig. 2. Total number of stalls for different bandwidths

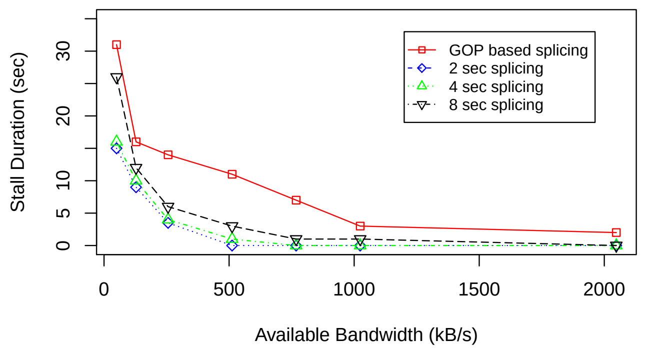


Fig. 3. Total stall duration for different bandwidths

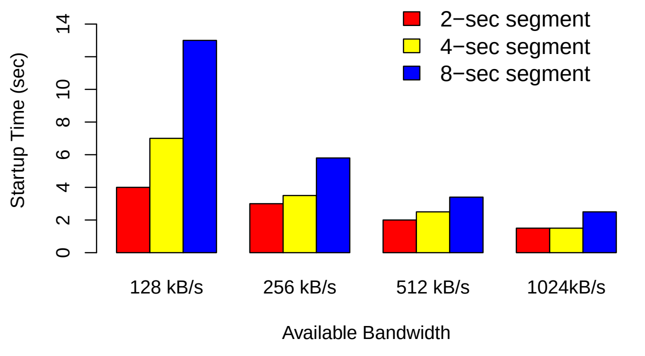


Fig. 4. Startup time for different bandwidths

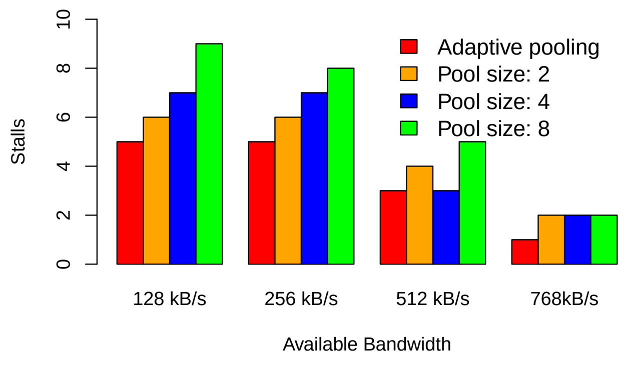


Fig. 5. Total number of stalls for different pool sizes

that our downloading technique (adaptive pooling) results in less number of stalls than downloading a predefined number of segments at a time. When the bandwidth is small, a large pool size increases the network overload in the peer's network which increases the stalls. Our technique used the information of available bandwidth and the buffered video to limit the number of segments to be downloaded. Again, if users have sufficient bandwidth, the pool size should be large to maximize the bandwidth utilization.

VII. Related Works

BitTorrent [17] is the most popular protocol for transferring files in a P2P system, but it was not designed for sharing interactive media. A video streaming solution has to have a small startup time and minimal stalls. The Internet Engineering Task Force (IETF) defined several P2P video streaming protocols, such as Libswift [18], PPSPP [19], etc. These protocols are based on UDP. In [7], streaming techniques of Netflix, Hulu, and Vudu have been studied. The researchers studied how the clients adapt the bit-rate based on available bandwidth. Several researchers have described the splicing technique of P2P video streaming. A layered encoding technique has been proposed for P2P video streaming where the video is composed of a base layer and enhancement layers [20]. Here the splicing is implemented on the layers. PPLive [4] , a popular P2P IPTV, splices the video into a fixed sized block of size 20MB [21], [22]. Several researches have been done on video streaming on GENI. [23] presented a video streaming solution that exploits multiple paths on GENI. [24] presented a recursive architecture to program the network traffic for video streaming on GENI. In this paper, we experimented GOP and duration based segmentation for TCP based P2P video streaming. We also proposed a prefetching strategy to minimize the stalls. We conducted the experiment on GENI.

VIII. Conclusion and Future Work

This paper measured the effect of splicing on the stalls. It compares the stalls and startup-time for duration and GOP based splicing. We found that duration based splicing performs better than GOP based splicing. We also proposed a prefetching strategy to reduce the stalls. We proposed a formula to calculate the number of segments a peer should download simultaneously. We also determined the segment size in a hybrid P2P system where contents are also served by a CDN. We conducted the experiment on GENI. This study gives us an insight into the splicing techniques for P2P video streaming. This paper showed that splicing techniques have a major impact on stalls and have the potential to be useful to improve the video streaming.

There are several issues that still need to be addressed. We did not propose an algorithm to determine the optimal segment size. An adaptive splicing technique will be able to increase the performance of P2P video streaming. Here we assumed that the bandwidth among the peers are fixed. But in real-word scenario, available bandwidth changes over time. An experiment should be conducted to measure the effect of splicing on variable bandwidth environment. At the same time, we also should experiment how the splicing works in case of competing flows and high congestion environment. Here we considered that all the peers are generous, but peers are usually very selfish. Our downloading policy also should take that into account. Finally a research should be conducted to compare our splicing technique to the splicing techniques of the state-of-the art P2P video streaming services.

Acknowledgment

The authors would like to thank Sarah Edwards of BBN Technologies and Victor Orlikowski of Duke University for their advise and help on tunneling VNC over a SSH channel for playing video on the virtual machines of GENI. The authors also would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

[1] "Sandvine: Global Internet Phenomena Report," 2015, URL: http://www.usatoday.com/story/tech/personal/2014/11/20/netflix-most-popular-net-service/19324419/ [accessed: 2015-02-23].
[2] "PPStream," http://www.pps.tv/.
[3] "SopCast," http://www.sopcast.org/.
[4] "PPLive," http://www.pptv.com/.
[5] S. Akhshabi, A. C. Begen, C. Dovrolis, "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP," in ACM conference on Multimedia systems (MMSys), 2011.
[6] T. Stockhammer, "Dynamic adaptive streaming over HTTP –: standards and design principles," in ACM conference on Multimedia systems (MMSys), 2011.
[7] T.Y. Huang, N. Handigol, B. Heller, N. McKeown, R. Johari, "Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard," in Internet Measurement Conference (IMC), 2012.
[8] "ISO/IEC 14496 MPEG-4 Standards."
[9] "GENI," http://www.geni.net/.
[10] "Libswift," http://libswift.org/.
[11] "GENI RSpec," http://groups.geni.net/geni/wiki/GENIExperimenter/RSpecs.
[12] "Xuggler," http://www.xuggle.com/xuggler/.
[13] "FFmpeg," http://www.ffmpeg.org/.
[14] "VLCJ," https://github.com/caprica/vlcj/.
[15] "LibVLC," http://www.videolan.org/vlc/libvlc.html/.
[16] Y. Li, Y. Zhang, R. Yuan, "Measurement and Analysis of a Large Scale Commercial Mobile Internet TV System," in Internet Measurement Conference (IMC), 2011.
[17] "BitTorrent," http://www.bittorrent.com/.
[18] R. Petrocco, J. A. Pouwelse, D. H. J. Epema, "Performance analysis of the Libswift P2P streaming protocol," in P2P, 2012.
[19] A. Bakker, R. Petrocco, "Peer-to-Peer Streaming Peer Protocol (PPSPP)," IETF draft, 2012.
[20] Z. Liu, Y. Shen, K. W. Ross, "LayerP2P: Using Layered Video Chunks in P2P Live Streaming," IEEE Transactions on Multimedia, 2009.
[21] G. Deng, T. Wei, C. Chen, W. Zhu, B. Wang, D. R. Wu, "Moderate prefetching strategy based on video slicing mechanism for P2P VoD streaming system," in IET International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN), 2011.
[22] X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, "Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System," in IEEE Transactions on Multimedia, 2007.
[23] Q. Wang, K. Xu, R. Izard, B. Kribbs, J. Porter, K.C. Wang, A. Prakash, P. Ramanathan , "GENI Cinema: A SDN-Assisted Scalable Live Video Streaming Service ," in IEEE International Conference on Network Protocols (ICNP), 2014.
[24] Y. Wang, N. Akhtar, I. Matta, , "Programming Routing Policies for Video Trafc," in IEEE International Conference on Network Protocols (ICNP), 2014.