Peer-to-Peer Assisted Livestreaming

Allison Turner, Guanzhou Hu, and Vishrant Tripathi

I. Introduction

Livestreaming applications like Twitch, Periscope and Facebook-Live are a crucial driver of internet traffic growth. For example, Periscope more than tripled the number of daily streams in a 3 month period in 2015 [1].

A typical livestreaming application generally has thousands of simultaneous broadcasters who generate video content and have low upload bandwidths; and millions of audience members who want that content as soon as possible in order to interact with it via comments and likes. Contrary to regular video traffic, livestreaming viewers demand an almost real-time, lifelike experience while broadcasters demand the ability to stream at an unprecedented scale.

The twin goals of **low latency and high scalability** are the key requirements of a livestreaming architecture. In the following sections, we describe how some of the most popular livestreaming services are able to achieve these goals, and explore whether there is scope for improvement via better design.

II. Current Architecture

We find that the general architecture described in [1] and [5] is indicative of how most current livestreaming systems are designed.

In Periscope, a broadcaster initiates a stream, transmitting their newly generated and encoded video to one CDN through the persistent connection-based Real-Time Messaging Protocol (RTMP). The first hundred or two hundred viewers connect to that same content distribution network and retrieve the video through RTMP. The video is propagated through this first CDN as well as a second one. Subsequent viewers connect to the second CDN and retrieve the video through the chunking-based HTTP Live Streaming protocol (HLS). All viewers connect to a third content distribution network in order to generate comments and other interactions, however, only that first group of a few hundred viewers can send comments.

The design is simple and it splits the job of latency and scalability between the two CDNs. The first CDN provides low latency to a few viewers while the second CDN provides reliable streaming to a large group of viewers.

III. P2P Assisted Livestreaming

CDNs are great at providing scalability and overcoming bandwidth limitations of the broadcasters. However, they are not the most efficient in terms of latency. This is especially true when viewers are closer to the the broadcaster as compared to the CDN. Frames are traveling from the broadcaster, past the viewers to the CDN, then back down to the viewers again.

We propose a system that uses a combination of *peer-to-peer connections, multicasting, and CDNs* to more efficiently stream live video. Audience members that are part of the broadcaster's local area network connect directly via a P2P connection. Audience members that are further away receive content from a CDN, similar to the usual setup.

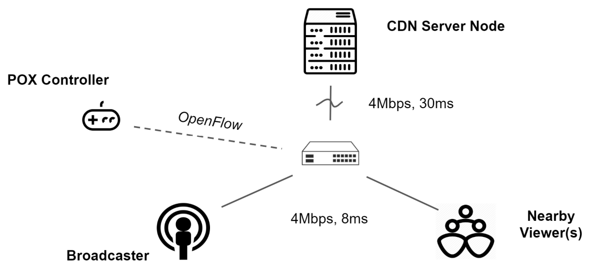

Fig. 1. Prototype Topology

Peer-to-peer systems for video traffic have been a popular area of study and there is a large body of work focusing on different aspects of the problem [2]–[4], [6]. However, most of these works consider a purely distributed P2P architecture. Our work involves augmenting a CDN based livestreaming system with switch-enabled P2P to enhance its delay performance while maintaining scalability.

IV. Proof of Concept System

We built a proof of concept system to show the viability of our proposed improvements[1]. We emulated a simple broadcaster-viewer-CDN topology in Mininet. We created logic to enable peer-to-peer connections with OpenFlow on a simulated POX controller. For testing, we created a simple video application using FFmpeg and MPlayer. We then examined our altered network traffic with Wireshark.

Our prototype topology consists of three hosts, a switch, and a controller (see Figure 1). In the regular setup, the broadcaster creates a connection to the CDN to upload content, and the audience member creates another connection to the CDN to request the content. When our improvements are enabled on the switch controller, the router can detect that the broadcaster and the viewer are in its local network by looking at RTMP metadata packets. This enables us to establish a direct peer-to-peer relationship between the broadcaster and viewer.

V. Experiments

We ran ping trains between hosts to test latency. We tested in both high bandwidth and bandwidth limited situations. We assumed the minimum round trip time for the broadcaster and viewer links to their switch should be lower than the RTT

[1]Our code can be found at this repo. A summary of our switch controller design can be found in these slides.

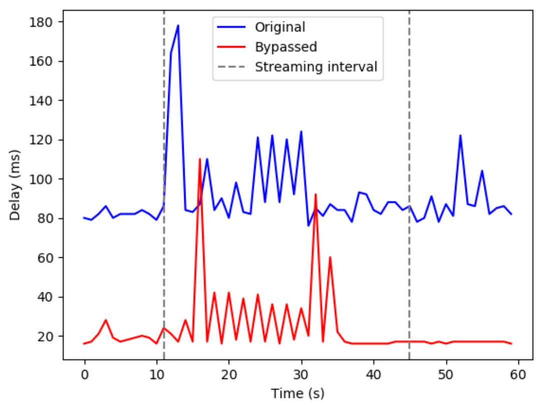

Fig. 2. Delay on a high bandwidth system. CDN link is 4mbps, 30ms and LAN links are 4mbps, 8ms.

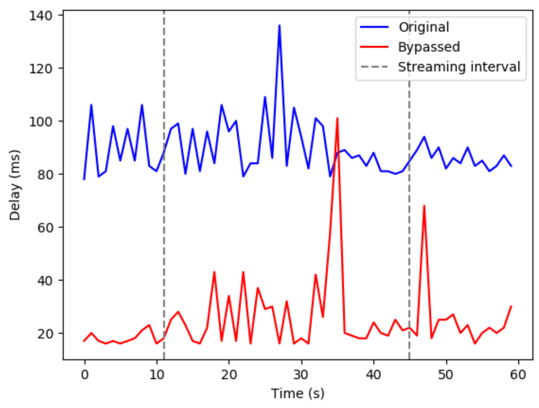

Fig. 3. Delay on a bandwidth limited system. CDN link is 100mbps, 30ms and LAN links are 10mbps, 8ms.

between the switch and the CDN, since ideally our peering system would leverage broadcasters and viewers sharing local area networks.

We found that latency on the stream was significantly reduced when the system switched into peer-to-peer enabled mode, as expected. There were some queueing delays, especially in the bandwidth limited case, but this is to be expected from a video application. See Figures 2 and 3 for the results of these experiments.

VI. Limitations

Our proof of concept system has several limitations. The protoype topology was very simple, only including a few hosts and one switch. We wanted to add another group of viewers far away from the CDN and enable multicasting, but troubleshooting the P2P enabled controller on groups of viewers that required different rules proved difficult.

Our livestreaming application was simplified, missing many of the control features of a fully developed application. We also did not include comments or reactions as the likes of Periscope and Twitch do. We chose not to implement these, and instead focus on the video aspects of the system, because we determined that such engagements were not a bottleneck for the streaming application.

VII. Future Work

On future iterations of this system, we would want to include IP multicasting and more P2P communication between viewing peers, since part of our goal is to eliminate extraneous client-server connections. We also want to explore edge and d2d caching on peers discussed in [6] to further reduce the amount of communication needed between the application CDNs and the users.

Exploring whether switch enabled P2P is a realistic design choice is also an interesting question. Future development could include reworking our method of implementing peering. Our router-assisted scheme might not be feasible for a real application, since getting local ISPs to change their switches for an application is unlikely. The peering mechanism would need to operate on an end-to-end, client-server basis. This is doable, but comes with some security concerns. One would want to be sure that any audience members or link interceptors would not be able to access any information about other audience members.

Acknowledgements

All of us were involved in discussions, surveying literature, and debugging problems to complete the project. A detailed distribution of work follows:

- Allison researched tools and libraries for our proof of concept and was responsible for much of the writing.
- Guanzhou proposed the novel design of the controller and switch-enabled p2p, and did the main part of coding.
- Vishrant got video streaming to work on the Mininet topology and was responsible for experiments and results.

References

[1] B. Wang, X. Zhang, G. Wang, H. Zheng, and B. Zhao, "Anatomy of a Personalized Livestreaming System," in Proc. IMC 2016, November 14-16, 2016, Santa Monica, CA, USA.

[2] D. Purandare and R. Guha, "An Alliance based Peering Scheme for Peer-to-Peer Live Media Streaming," in Proc. P2P-TV, August 31, 2007, Kyoto, Japan.

[3] E. Miller, "Akamai's Peer-To-Peer Love Story," https://blog.peer5.com/akamais-peer-to-peer-love-story/.

[4] K. Pires, G. Simon. "DASH in Twitch: Adaptive Bitrate Streaming in Live Game Streaming Platforms," VideoNext 2014 : 1st Workshop on Design, Quality and Deployment of Adaptive Video Streaming, Dec 2014, Sydney, Australia.

[5] "Twitch Engineering: An Introduction and Overview," Dec 2015, Online, https://blog.twitch.tv/en/2015/12/18/twitch-engineering-an-introduction-and-overview-a23917b71a25/.

[6] T. Zhang, X. Fang, Y. Liu, G.Y. Li, and W. Xu (2019). "D2D-Enabled Mobile User Edge Caching: A Multi-Winner Auction Approach." IEEE Transactions on Vehicular Technology.

[7] "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper," Online, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.