Journal of Network and Information Security
11 (1) 2023, 08-12
http://www.publishingindia.com/jnis

High Bandwidth Applications: An Analysis of Strategies Employed by Video Hosting and Distribution Platforms to Handle High-Volume Traffic

David C. Culbreth[1] and Ayad Barsoum[2*]
[1]Student, St. Mary's University, San Antonio, TX, USA. Email: dculbreth@mail.stmarytx.edu
[2]Associate Professor, Department of Computer Science, St. Mary's University, San Antonio, TX, USA.
Email: abarsoum@stmarytx.edu
*Corresponding Author

Abstract: In the modern era, people all around the world utilize video conferencing and video streaming applications for their jobs, to see family, for school, for entertainment, and much more. This now ubiquitous technology accounts for more than 80 percent of all internet protocol traffic in the world, making broader understanding and further refinement even more crucial to further supporting the most burdensome application of modern telecommunication technology. In pursuit of enabling more video streams of higher quality, computer scientists and software engineers have taken the opportunity to optimize the video transmission process at every level. The predominant CODECs – video encoders and decoders – like AVC and HEVC have taken the first step by lowering the quantity of raw data required to deliver a video or video segment. Traditional video hosting platforms like YouTube, Netflix, and Amazon Prime Video utilize these CODECs across their services to both minimize the storage space required as well as network capacity needed to deliver the content. Video and voice conferencing platforms like Zoom and Discord have taken these compression techniques with a different approach, prioritizing the responsiveness of the video stream over the raw image and audio quality. Live streams and internet shows, distributed through platforms like Twitch and Facebook Live, take yet another approach by broadcasting a single live video stream to hundreds or thousands of viewers, which brings yet another set of technical challenges to the table surrounding managing the traffic originating from one specific source at a moment in time. This paper seeks to explore the challenges of each application and some of the distribution strategies taken by each to improve their performance and minimize their costs.

Keywords: Content delivery network, High bandwidth applications, Video streaming.

I. Introduction

While video broadcasting in and of itself is no new technology, the distribution of video content both on demand and at scale, to clients in arbitrary parts of the world is a functionality that has only been made possible by recent advances in network and client capabilities, enabling internet network infrastructure to deliver truly astonishing volumes of video data. More than 80 percent of all internet traffic in the world is now attributed to streamed video [1, 4]. Delivering this volume of data requires serious consideration of the greater costs, impacts and needs of providing this service at scale.

The convenience of looking up a tutorial video, calling up your family or streaming your favorite show live is a luxury that is afforded to modern citizens of the internet. This luxury, however, comes at a toll on the network, by occupying bandwidth that could be used for other kinds of internet traffic. With increased demand for on-demand video content, internet service providers and internet backbone services have not only increased the bandwidth capacity of their services, but also begun coordinating with major players in the video distribution space to physically locate video resources in strategic positions on the network to improve latency and minimize the overall burden on the network [2, 6].

II. Compression and CODECs

The simplest and first strategy to high bandwidth networking needs is to reduce the overall data that must be sent over the network [8, 10]. That is, establish a compression standard that allows fewer overall data to be sent across the network. In video-reliant applications, this compression and decompression is accomplished with cooperative encoder and decoder programs, known together as a CODEC [9, 11].

When establishing a CODEC standard, there are three factors (shown in Fig. 1) that must be balanced: 1) computation time and

complexity, 2) the compression ratio, and 3) the video quality. When the intent of the resulting video is to be archived or sent across the network, the CODEC will prioritize the compression ratio and quality of the video, utilizing higher computation time and complexity to achieve very high compression while still retaining near-full quality. Modern compression CODECs like AVC (a.k.a. H.264 [7]) and HEVC (a.k.a. H.265 [6]) have optimal compression ratios of 60:1 and 1000:1 respectively [11], meaning that a 1 MB file would be compressed to 16 KB or 1 KB. While these CODECs allow for amazing levels of compression and therefore higher quality video across the same network connection, this compression comes at the cost of longer computation time. In the case between AVC and HEVC, there is a 40% increase in compression at a cost of 10x the computation load [10].

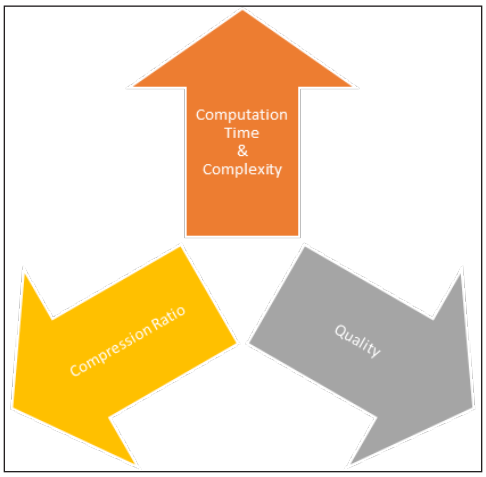

Fig. 1: Three Factors to Consider when Designing a CODEC

While increasing the compression ratio in a CODEC is the most effective method for minimizing overall bandwidth, just bandwidth consumption does not fully inform whether a given CODEC is well-suited for the specific application. If the CODEC does not sacrifice some video quality, the compression can take a substantial amount of time – enough to impact the perceivable video quality and make the video appear laggy, even when there is ample bandwidth. For this reason, many platforms that deliver interactive video, like conference calls, will develop their own CODEC in order to prioritize lower-latency and provide for smoother-appearing application interaction. In 2014, Ohm *et al.* [2] developed a CODEC standard that dropped their encoding latency from 33 ms, or 2 frames, to 2 ms, which is a fraction of a frame. This meant that before the next frame was even captured, the data from the previous frame was ready to be delivered. Considering that CODECs like AVC and HEVC require that frames be sent in clusters of 5 or 10 at a time, this can drastically improve the perceived call quality.

III. Content Delivery Networks (CDNs)

Having addressed the obvious approach of reducing the overall data required to deliver a video across the network, when bandwidth remains an issue, further steps must be taken to provide access to the application. In networked applications delivering content, the tool to use is a content delivery network. Content Delivery Networks are built with the consideration that clients will connect from many different geographic locations and deploy a cluster of intermediate servers to both provide a fast, stable, local connection, as well as reduce the load on the original service host [13]. The precise architecture of the CDN (Content Delivery Network) will depend on the nature of the application being served. Right now, there are three distinct styles of applications: 1) traditional hosted video platforms like YouTube and Netflix, 2) live-streamed video platforms like Twitch and Facebook Live, and 3) conference-style applications like Zoom and Teams.

The first of these styles utilizes a more traditional CDN, with a sole source, or a central master record management server that delivers records – in our example, videos – to a collection of geographically positioned edge nodes. With the correct DNS configuration, clients in a particular geographical region will be routed to their nearest node when requesting the resource. Fig. 2 shows an example architecture diagram of a typical CDN.

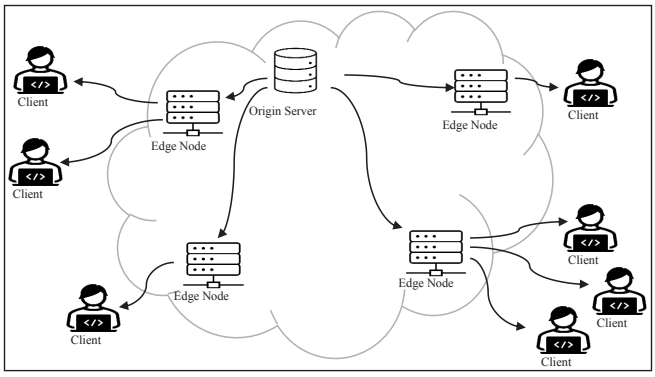

Fig. 2: Example Architecture Diagram of a Typical CDN [5]

When a resource requested by a client is not available on the edge node, the edge node will request said resource from the central management server, cache it for further requests, and then deliver it to the client. This method of caching is the "pull" method, where the clients' requests themselves dictate the cached content of each node. This method has elegance as a general-purpose solution because there is no complicated formula for determining what content should be available on each edge node.

The other caching method, known as the "push" method, involves predicting which videos will be likely to be requested, taking numerous factors into account, like the popularity of the author, the length of the video, or the scheduled release date. The downside in the "push" method is that determining the content to deliver is purely a guess, with a potential for unnecessary traffic being generated if the guess is wrong. However, not all traffic needs to be a guess. The "push" and "pull" methods can be used concurrently, pushing the highly probable requests, and allowing the rest to be pulled. With a combination of "push"

and "pull", the relevant content for a particular geographical area can be kept up to date on its edge node, and the popular content can be delivered in advance to prevent a rush on the central server [12]. Fig. 3 shows a comparison of the "push" and "pull" methodologies.

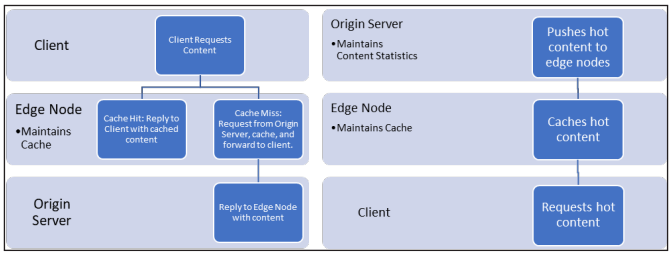

Fig. 3: A Comparison of the "Push" and "Pull" CDN Distribution Methodologies

Determining what content is delivered to each edge node can be simply accomplished using a combination of the two methods above, but that still leaves the problem that the Edge nodes also do not have infinite storage. There must be some decision process about which content to delete from a full cache in order to make space for pushed or pulled records. In both instances, the edge node can maintain an index of the content for when it was last used, and purge records with the least recent usage. In this schema, frequently requested content will constantly be brought to the top of the list, the less popular content will sink to the bottom, and eventually be deleted as other content is delivered to the edge node.

Live-streamed applications have additional challenges that must be addressed, revolving around the fact that the data being delivered exists only moments before it is distributed. Additionally, the video, once delivered to the platform, must be processed into several different resolutions to deliver to clients having a broad range of bandwidth capabilities. The remainder of the delivery process can be delivered using the previously described CDN architecture [3]. Fig. 4 shows an example of a network architecture diagram of a CDN oriented for live-streaming video content.

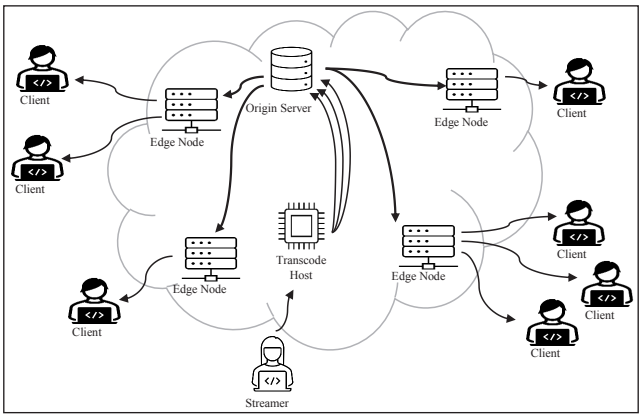

Fig. 4: Example Network Architecture Diagram of a CDN Oriented for Live-Streaming Video Content [3]

The traditional CDN architecture serves the previous two application styles very well, due to there being a single source of the content. When considering conference video applications, there may be as many as 50 sources and destinations, or more. In this situation, there are two naïve approaches to distribute all the content: centralized distribution and peer-to-peer distribution. Centralized distribution would have all the clients connect to one central server, which handles redistributing each of the video streams. The issues with this approach are revealed when clients from geographically distant locations or communicating over potentially unreliable connections. The other naïve approach seeks to solve these by directly connecting the clients, each to every other participant on the call. This approach brings its own issues when considering that the number of connections increases with the square of the number of clients. With these two naïve approaches determined to be non-starters when scaling the application, the solution lies somewhere in the middle. A pseudo-peer-to-peer CDN retains the collection of edge nodes, to service clients in distributed geographic regions, but delivers the bundled video streams across what is presumably a reliable connection to each participating region's edge node [2]. This hybrid infrastructure allows the service to manage several aspects of the call, including the quality of the streams sent to each client.

Fig. 5 shows an example of a network architecture diagram of a pseudo-peer-to-peer network. In this example, there are four different geographical regions, each serving its own distinct clients. This design is intended to minimize the required connections while still maintaining connection stability in multiple geographic regions. Note that in this image, the arrows point in both directions, indicating how each client both consumes and provides content that must be delivered.

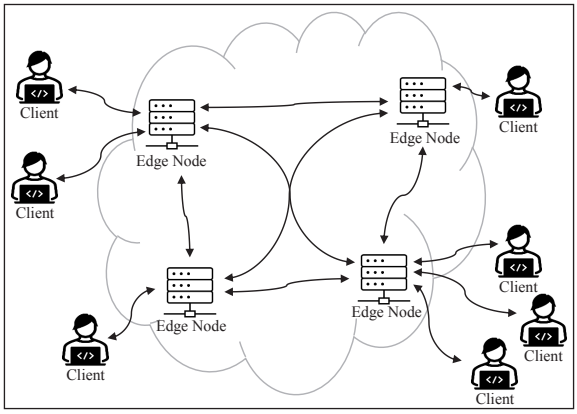

Fig. 5: Example Network Architecture Diagram of a Pseudo-Peer-To-Peer Network [2]

The management of data volume delivered to a given client during a conference video call is critical for maintaining smooth call quality. The service provider must consider that a client's capacity to send and receive information may vary during the duration of the call. This is important because if a client were inundated with call network traffic, the conference call

itself would not be the only service denied. At best, network packets delivering call data would be dropped, resulting in a choppy stuttering call experience, and at worst, the client could be forced to reset its network adapter due to the high volume, dropping the connection altogether. By placing the service as a middleman in the call, the service can deliver the packets.

Another advantage to this pseudo-peer-to-peer architecture is the granular route control afforded to the service provider. With the larger distances being taken between edge nodes rather than clients, the routes for the data can be managed more efficiently, which allows for more reliable delivery of the video stream in a timely manner, or stream splitting to deliver packets from the same stream across several routes simultaneously.

IV. Development and New Improvements

Amongst all the established standards and procedures, there is still much active development in the area of video distribution. Hardware-accelerated video encoding greatly improves to the slower portion of the CODEC process by developing dedicated circuits to process the video information. Recent advances in machine learning algorithms have brought to light the potential for utilizing AI to upscale frames and generate entire frames with more realistic predicted inter-frames. In the distribution space, peer-to-peer processes are being developed to accelerate the distribution of content while further minimizing the impact on the source server.

In the encoding acceleration space, recent microarchitecture improvements have greatly reduced the time required to compress a video. With the assistance of dedicated hardware, video encoding processes can be encoded at 10 times the speed overall, and 950 times the speed of encoding inter-frames. This dramatic performance improvement comes at only 2 to 3 times the power consumption at the time of the study. More recent microprocessor node improvements will further improve both speed and power efficiency [13].

In addition to accelerating the encoding itself, machine learning algorithms can be used to efficiently upscale smaller images. This makes the possibility of delivering a lower-quality video more viable when it can be reasonably upscaled in a way that is relatively imperceptible. This is especially useful in conference videos where precise quality is not always a priority. In addition to simply upscaling images, entire frames can be generated or predicted using machine learning techniques [5]. The current issues with this approach largely revolve around efficiently caching the machine learning dataset. In order to properly predict or upscale frames, a rather large metadata map must be delivered and stored on the clients for use in upscaling the video stream.

In Content Delivery Networks, there are three important goals achieved by deploying a cluster of edge nodes: 1) reducing load on the source host, 2) reducing latency, and 3) providing greater bandwidth to deliver content where it's needed. Internal to the CDN, by utilizing peer-to-peer distribution strategies, all three of these objectives can be further accomplished. If the edge nodes maintain a set of torrent magnets, the source server can simply validate the checksum of each record, dramatically decreasing both the bandwidth and capacity required by the source host and improving distribution speed to each edge node. When needed, an edge node can fetch multiple chunks of the same record simultaneously [12]. By implementing peer-to-peer strategies, many of the key performance improvements provided by the architecture can be further enhanced.

V. Conclusion

High-bandwidth applications operate most efficiently when every opportunity is taken to improve performance. Optimizing compression algorithms and designing dedicated silicon to accelerate compression and video prediction enable fewer overall bytes to be delivered over the wire, with impressive performance when utilizing modern silicon. Additionally, by utilizing a content delivery network with the appropriate architecture, impressive amounts of data can be delivered with relatively minimal load on a central source service. High bandwidth applications must carefully consider each technical choice, from video CODEC to the route the data takes to arrive at the client because at scale, each of these factors can help or hinder the process.

References

[1] I. Richardson, H.264 video Codec Inter-Prediction. [Online]. Available: http://www.staroceans.org/e-book/vcodex/H264_interpred_wp.pdf

[2] Zoom Video Communications, Inc., Zoom Global Infrastructure. [Online]. Available: https://explore.zoom.us/docs/doc/Zoom_Global_Infrastructure.pdf

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

[4] Cisco: Global 2021 Forecast Highlights. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf

[5] I. Salian, NVIDIA Corporation. What is AI Upscaling? [Online]. Available: https://blogs.nvidia.com/blog/2020/02/03/what-is-ai-upscaling/

[6] J. Ozer, "Subjective HEVC transcoding quality," INTNET Streaming Learning Center. [Online]. Available: https://netint.ca/wp-content/uploads/2019/12/NETINT-White-Paper_final2.pdf

[7] A. Kashyap, and B. Bing, "Efficient HD video streaming over the internet," *Proceedings of the IEEE*

SoutheastCon 2010 (SoutheastCon), 2010, pp. 272-275, doi: 10.1109/SECON.2010.5453873.

[8] P. Martinez-Julia, E. T. García, J. O. Murillo, and A. F. Skarmeta, "Evaluating video streaming in network architectures for the internet of things," *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2013, pp. 411-415, doi: 10.1109/IMIS.2013.76.

[9] H. Noh, and H. Song, "Cooperative and distributive caching system for video streaming services over the information centric networking," *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, 2019, pp. 210-213, doi: 10.1109/LCN44214.2019.8990762.

[10] J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards-including High Efficiency Video Coding (HEVC)," In *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669-1684, Dec. 2012, doi: 10.1109/TCSVT.2012.2221192.

[11] J. Ohm, and G. J. Sullivan, "Meeting report of the 13th meeting of the Joint Collaborative Team on Video Coding (JCT-VC)," Incheon, KR, Apr. 18-26, 2013.

[12] T. Sanguankotchakorn, and N. Krueakampliw, "A hybrid pull-push protocol in hybrid CDN-P2P mesh-based architecture for live video streaming," 2*017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2017, pp. 187-192, doi: 10.1109/APNOMS.2017.8094200.

[13] A. Vakali, and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68-74, Nov.-Dec. 2003, doi: 10.1109/MIC.2003.1250586.

[14] J. Kufa, and T. Kratochvil, "Software and hardware HEVC encoding," *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2017, pp. 1-5, doi: 10.1109/IWSSIP.2017.7965585.