Combination of Video Streaming Technologies as CDN/P2P Hybrid & ABR/P2P Hybrid

Andrew L. Peterson
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
pete9443@morris.umn.edu

ABSTRACT

Peer-to-peer networks, content delivery networks, and adaptive bitrate streaming are all approaches to distributing content over the internet. These technologies can be used to stream video, but improvements in efficiency or scaling these separate technologies can be difficult. By combining them, these technologies can be improved. Peer-to-peer networks can be combined with content delivery networks to take some of the load of distributing data off of the content delivery network to increase its capacity. Peer-to-peer networks can be combined with adaptive bitrate streaming to provide a flexible video streaming service.

Keywords

Video Streaming, Peer-to-Peer, Content Delivery Networks, Adaptive Bitrate Streaming

1. INTRODUCTION

Streaming video is a popular form of data that is distributed all over the internet. There are multiple approaches to streaming videos. Content delivery networks and peer-to-peer networks are infrastructure setups that will allow clients to receive video at a large scale. Adaptive bitrate streaming is a method to improve video streaming so clients may have the best viewing experience regardless of network congestion. These technologies work well on their own and are currently implemented in real world video streaming services. However, these technologies have room for improvement. Peer-to-peer networks can only share data at a fixed bit rate and are not adaptive. Content delivery networks can be expensive to scale for large user bases. Adaptive bitrate streaming requires servers to encode videos which can be expensive if it is to serve large user bases. By combining these technologies it is possible to create new and better methods for streaming video. By combining content delivery networks and peer-to-peer networks as a hybrid it is possible to improve user satisfaction and lower network costs than if the technologies were separate. Alternatively by combining peer-to-peer networks and adaptive bitrate streaming it is possible to serve multiple bitrates over a peer-to-peer network when that was not possible before. These combined technologies were stress-tested in simulations and compared to older methods of video streaming. When compared, it is easy to see these new combinations can be more efficient.

Section 2 will describe necessary knowledge for understanding video streaming and technologies that will be combined and discussed later on in the paper. Section 3 will discuss a CDN/P2P hybrid, its implementation, what new responsibilities it adopts, and analysis of simulations. Section 4 will cover Joint-Family, an ABR/P2P hybrid. This section will also talk about its implementation and analyze a set of simulations this approach went through. Section 5 provides conclusion.

UMM CSci Senior Seminar Conference, December 2016 Morris, MN.

2. VIDEO STREAMING TECHNOLOGIES

Video streaming is the process of playing back video as it is being downloaded. This could be for either prerecorded or live video. Video streaming is a popular use of the internet. Many people use popular video streaming services such as YouTube and Netflix every day. These combinations of technologies could improve today's streaming capabilities. Here are some technologies that can be combined with one another.

2.1 Centralized Server

A very common way to host data on the internet is through a single server. In this method a server will process requests sent by clients. A request may be to calculate a math equation, and the server will respond with a solution, or a request might be to send the client a picture, and the server will respond accordingly. Streaming a video from a server acts a similar way but through multiple requests from the client. A client will request a streamed video and as it is playing it will send more and more requests to let the server know to continue streaming the video to them. The server will respond with chunks of data that contain part of the video. When stitched together on the client side, it creates a whole seamless video. This is a simple approach to streaming a video but it does not scale well. If a specific video on a single server were to become popular, the amount of requests would increase and could potentially overload the server. This means that the server cannot process all incoming requests in a reasonable amount of time, and users will begin to stop receiving service. This makes a centralized server a poor approach to streaming video. However, using a server is still an option, only in a different way, such as part of a

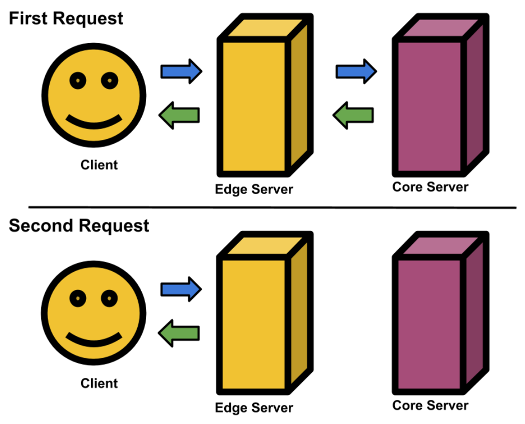

Figure 1: After content has been retrieved from the core server, later requests require a much shorter trip to collect the data since the data has been cached on the edge server.

content delivery network.

2.2 Content Delivery Networks

A Content Delivery Network (CDN) is an expansion on centralized servers. This approach has many geographically distributed servers connected to each other. Requests are delegated to different servers depending on the geographic location of the server and the client making the request.

A CDN will consist of core servers and edge servers. Core servers, sometimes called origin servers, maintain the CDN by giving content to the edge servers when necessary. An edge server will respond to a client request.

Fig. 1 shows an example of how the client, core server, and edge server interact with each other. If a client requests some data from an edge server and it does not have that content, the edge server will get it from a core server. The edge server will cache that data for later use. If the client returns and requests the data again, or another client requests the same data, then the edge server can respond immediately from the cache.

This is better than a centralized server because it does not have to take the full trip every time, and just needs to make a shorter trip to the edge server. It also distributes the load among multiple edge servers. The distribution of load can be seen as an improvement on a centralized server. More edge and core servers can be added to scale its load capacity even further. This approach is good for hosting popular content. However, it can be expensive because of the number of servers needed for it to work.

2.3 Adaptive Bitrate Streaming

Adaptive bitrate streaming (ABR) is a method of streaming content at a changing rate for overall best user experience. In terms of video streaming, ABR will serve video to a client at different rates depending on the rate at which a client can download a video. That download rate may vary from time to time, and may even change while the client is streaming. This can be due to network congestion. Network congestion happens when many people are using a local network at the same time. Because there is a limit to the amount of information that can be transmitted over a network in a finite amount of time, the local network will bottleneck and experience slower internet connection. When this happens, it might take longer to download a video than when the local network is not as congested [5].

A bitrate is the amount of bits that can be transferred in a specified amount of time. When a server is transferring video, a higher quality video will take more bits to send because it contains more data than a lower quality video would. So in the context of video a bitrate relates to quality. It can be thought of as how much detail is transferred to the client in a specified amount of time.

It is important that the client downloads the video faster than the video is being played back, or else the video will need to pause and wait for more chunks to arrive. When the client needs to wait due to insufficient download speed, it is called lag.

It is up to ABR to serve at different rates to accommodate the client's fluctuating download speed. By sending lower quality video to the client it is thus sending a lower bitrate. Conversely by sending higher quality video the bitrate is raised. ABR will change bitrates depending on the clients download speed capabilities. Then the client will have less or no lag and an improved viewing experience.

When ABR is implemented it will exist on a server for clients to interact with. When a client wants to receive video it will send a request to the server. The server will not respond with the video but instead send a manifest of all possible bitrates the server has of that video.

Fig. 2 shows an example of how ABR may work. The client will receive the manifest and decide on a bitrate. A bitrate is decided and the client sends another request stating which bitrate they would like to receive. The server will begin to serve the video at the specified bitrate and the client will playback the video at the same time they are downloading it. This is the streaming part of ABR. This protocol may vary slightly. YouTube for example may begin to serve with the same bitrate that was served during a previous video viewing. Or a server can automatically find the most optimal bitrate by serving the video at an arbitrary bitrate and switch bitrates when necessary. Now ABR just needs to know how to switch bitrates when necessary. This is done through thresholds.

The client keeps track of a buffer and two thresholds. A buffer is the amount of video that is downloaded but not been played back yet. One threshold is how much buffer remains before the video should be downgraded. If the video is played back faster than it is downloaded the buffer will shrink. As it shrinks it could pass this threshold. The client will automatically send the server a request to change the bitrate to a lower quality if possible. Conversely if the video is played back slower than it is downloaded then the buffer may exceed the second threshold. When this happens it means the bitrate is too low and the client can have a better viewing experience and is able to upgrade to a higher quality. The client automatically sends a request to the server when this threshold is passed to increase the quality if possible. This is the adaptive part of ABR. These thresholds are how ABR knows when to upgrade or downgrade a stream, and thus prevent the client from lagging while still providing the highest video quality.

2.4 Peer-to-Peer Networks

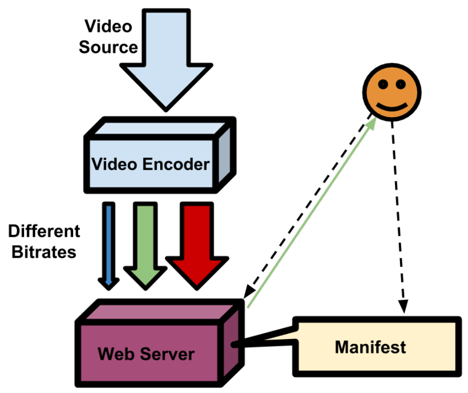

Figure 2: A video can be encoded at multiple bitrates, stored on a server. A client can choose a bitrate from the manifest that exists on the server and have the video served to them.

Clients themselves have the ability to share data with each other without the help of servers. When clients connect to each other directly to share data it is called a peer-to-peer network (P2P). A client in a P2P network is called a peer. A sub-group within the P2P network is called a swarm. A swarm of peers shares data with each other and do not interact with the larger network. There are no servers in this approach to sharing data and only the computers used to connect to other peers. This means there is no centralized server cost when distributing data.

In a P2P network a peer can host some data from their computer. When a peer does this, they are called a seeder because they are seeding data. Other peers are allowed to download this data. These peers are called leechers because it is like they are leeching the data. Now that other peers have the data they are allowed to seed it as well. During their download they can seed the data, and once they are done downloading and continue seeding they become seeders. Popular content can be downloaded quickly. This is because there are many peers that have the popular content that leechers can download from, and thus there are more points of service for these leechers [6]. All peers and leechers have an upload rate and download rate. This is how fast a peer can upload and download date to and from other peers respectively.

Streaming video over a P2P network is not common but it is possible. A reason this is not done is because it is unpredictable and can only be served at a single bitrate. P2P networks are not made for encoding and sharing data, and are built for a one time download situation. Not only that, but it is hard to predict the popularity of video. This means if one peer wants to watch a video that is not popular, it will be a slow download and then streaming the video will be undoable due to lag. P2P networks are a great approach to cost effective distribution, but it is a challenge to make it possible to use with streaming video[7].

3. CDN/P2P HYBRID

CDN/P2P hybrid is an intersection between the two technologies. The CDN network alters some of its responsibilities but for the most part operates the same as it would usually. The P2P network makes no changes in its protocol. The benefit from intersecting these two technologies is that the CDN can delegate server load to the P2P network.

The P2P network needs no major changes. Most peers in the network will be blind to the CDN portion of this system. The P2P network will be seeded initial data from the CDN. From there the P2P network will function the same as it usually would by sharing data with other peers. Server capacity will be freed up by having the responsibility of data sharing on the P2P network, and this extra capacity can be used to respond to client requests which scales the CDN.

The CDN still contains all its usual infrastructure such as core servers and edge servers. In this hybrid an addition is made to this structure. A proxy server is attached to each edge server. A proxy server can exist as a physical server or software existing on the edge server. In either case it interacts only and directly with the assigned edge server.

When clients request a stream they usually are connected to the edge server which will complete their request. When a proxy server is attached the request will be directed to the proxy server where it will be responsible for three things [4].

- Computing the best bitrate adaptation strategy for clients if the capacity is full or is being filled
- Running a directory service. This service helps the CDN map videos that are currently being streamed to which clients that are requesting it.
- Based on the computed best bitrate adaptation strategy the proxy server will need to decide how much bitrate will be allocated to the client. It will also decide if the client is to be served the video directly or through the P2P network it is interacting with.

In the context of video, a CDN will serve data to a client who requests it from the edge server, whether it was cached there or collected from a core server first. The video can be encoded at different bitrates on a core server and be requested at different rates. CDN it is not adaptive, and the bitrate is constant. The way a change in streaming bitrate would change on a CDN would be if a client left and no longer requested chunks for a video and another arrived requesting chunks for the same video at a different bitrate [4].

Server capacity is limited in a CDN as it would be on a centralized server. If a client requests video and they are not vying for any server capacity they can be served at the highest bitrate they can request because it can only process a number of requests at a given time. This is a first come first serve approach, and it leaves those who request video later to be served video quality at a lower bitrate than requested. A different approach is to serve every client lower quality bitrate than requested to leave room for later coming clients. This is an attempt at maximizing edge server capacity. In the hybrid approach both of these methods will be used depending on the situation.

This is where the best bitrate adaptation comes in. The best bitrate adaptation strategy, which can be called the BAS, is how the CDN decides the best bitrate for each client. This depends on user arrival rates, video viewing duration, and distribution of requested video bitrates. When a server has no requests, the capacity is completely free and thus has no competition. As clients begin to request chunks the capacity is filled. Once there is no capacity left is when BAS

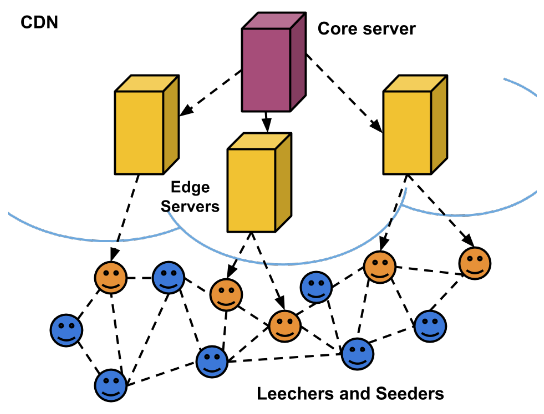

Figure 3: Video content originates from the core server and is distributed to the edge servers. Clients can request the video and later seed the video to other leechers and take on the responsibility of distributing the video.

is implemented. As more requests arrive with not enough capacity to process and serve them, the server can lower other bitrates to make room for the new request [4].

When a CDN uses the BAS it can be called an Adaptive CDN. Since this combination uses P2P in the system it is called Adaptive CDN/P2P. For simplicity's sake it will be referred to as CDN/P2P unless compared against other systems.

The BAS is a linear optimization problem: Can the capacity of the server be divided into different bitrates that can be served to all clients in a way that would minimize dissatisfaction and accommodate as many users as possible? Dissatisfaction is measured by the difference in bitrate downgrading when a client requested a higher bitrate. This can be represented in a formula. To understand this formula, know that there exists a list of bitrates $1, 2, ..., R$ that are represented as r_i.

$$min \sum_{i=1}^{R} \sum_{j=i}^{R} x_{ij} l_i \left(r_i - r_j \right) \tag{1}$$

Here $(r_i - r_j)$, is the amount the client is going to be dissatisfied where r_i is the bitrate requested and r_j is the bitrate that was served. The difference between these two bitrates is the amount the bitrate was downgraded. If the difference is small then the dissatisfaction is small. This is multiplied by l_i which is the number of leechers for the ith bitrate. This number is multiplied by x_{ij}. x_{ij} represents a fraction of the total number of clients who requested bitrate i but instead were downgraded to bitrate j. The amount of dissatisfaction, mutiliplied by the number of clients who were downgraded summed together for all possible pairs of bitrates available, represented as the double summation in formula (1), gives us the overall dissatisfaction in the system. The min part of this formula means this dissatisfaction must be as small as possible, or optimized. Minimizing dissatisfaction is similar to maximizing inter-client fairness. Inter-client fairness is the weighted average of fairness over all clients. Fairness of a client is the ratio of the bitrate that was delivered to the bitrate that was requested. Higher inter-client fairness means there is a wide spread fairness throughout the clients and thus less overall dissatisfaction. It is possible to find the minimal amount of dissatisfaction given two conditions.

$$l_i u_s \geq \left(l_i x_{ii} + \sum_{k=1}^{i-1} n_{lk} x_{ki} - s_i \right) (r_i - u_l) \tag{2}$$

$$\sum_{i=1}^{R} s_i r_i \leq C_{proxy} \tag{3}$$

In these formulas know that variables with subscripts i or l will mean seeders or leechers respectively. u is a variable that represents upload rate. Therefore u_s and u_l are the upload rates of seeders and leechers respectively.

The condition in formula (2) states that the number of seeders must be sufficient in relation to leechers. That is to say the seeders upload rate must be higher or equal to the download rate of the leechers. The term $l_i x_{ii}$ represents all leechers that got the bitrate they requested. The term $\sum_{k=1}^{i-i} n_{lk} x_{ki}$ represents all leechers who were downgraded. The number of seeders for a bitrate i is subtracted from these two terms. All of this is multiplied by the download rate i for all leechers of bitrate i. This completes the right side of the condition and represents the download rate of the leechers. The left side of the condition is the fraction of seeders for bitrate i multiplied by their upload rate, which represents the total upload rate for the leechers.

The condition in formula (3) requires the server capacity, C_{proxy}, to be more than the capacity the seeders take up. On the left of the condition there is the sum of all seeders in a certain bitrate multiplied by that bitrate. This can be seen as the server capacity. As long as this is less than or equal to the server capacity then there exists a solution for formula (1). In other words, as long as there is sufficient server capacity to serve clients, and there are enough clients to seed the leechers, then it is possible to achieve minimal dissatisfaction, or maximum inter-client fairness.

These two inequalities must be true for formula (1) to have a possible solution. If formula (2) or formula (3) have an outcome of false then it is not possible to find minimal dissatisfaction.

Fig. 4 shows the results of simulating this architecture. This is a simulation run to show the different ways of using CDN and how they compare with each other. The x axis plots ρ, which the average number of users in the system. The y axis plots the inter-client fairness percentage. Adaptive CDN/P2P, Adaptive CDN, and Single-rate CDN/P2P are compared. With lower numbers of average users in the system Adaptive P2P/CDN is comparable to Adaptive CDN, with single rate CDN/P2P at the disadvantage. As the number of clients in the system rises Adaptive CDN begins to fall in inter-client fairness and becomes more comparable to Single-rate CDN/P2P. Adaptive CDN/P2P remains the highest in inter-client fairness throughout. In this simulation there is a 40 percent increase in inter-client fairness in comparison to the other CDN implementations.

4. ABR/P2P HYBRID

Through changing protocols in a P2P network it is possible to implement ABR into it as ABR/P2P hybrid, which

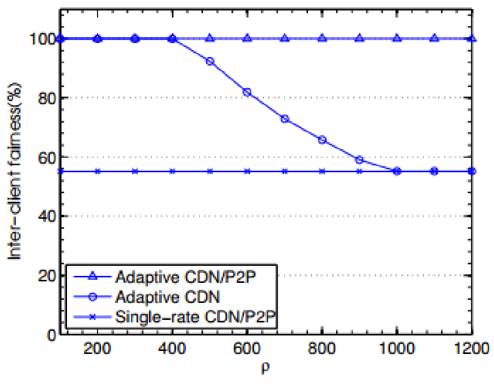

Figure 4: Adaptive CDN/P2P has the largest inter client fairness over Adaptive CDN and Single-rate CDN/P2P even at increased average number of clients.[4]

is usually called Joint-Family. This approach requires popularity of a video to work as usual. In highly popular videos there would be many peers to share chunks of data with each other. Otherwise, in less popular videos, there would be fewer peers to share chunks and thus the swarm would have less upload capacity. This lower upload capacity would not be able to support additional bitrates that ABR requires to work [2]. By improving seed staying time the system can strengthen upload capacity allowing for more than one bitrate. Seed staying time is the amount of time a seed is uploading data to other leechers before leaving the swarm. The longer the seed staying time the larger the swarm can get and thus improves upload capacity.

4.1 New Protocols

P2P requires some different protocols to be able to adopt ABR. In ABR there are multiple bitrates that clients can switch to when streaming a video. For this to happen in P2P the network must contain multiple bitrates. To do this the different bitrates for the same video will be contained in different swarms. For example if there were 2 bitrates for one video, there would be 2 swarms each containing a set of peers seeding and leeching different chunks of the video. The way a peer decides to increase or decrease the bitrate would be the same as the ABR technique with upper and lower thresholds for the buffer to react to. Once the peer decides to change the bitrate they will begin to request chunks from the appropriate swarm. Peers are allowed to participate in more than one swarm. For example if there are two swarms, swarm one and two, and a peer from swarm one may need chunks that a peer who is participating in swarm two has. The peer from swarm two would be able to share chunks in both swarms. This accommodates for peers who switch between swarms. If a peer changes swarm participation then the chunks previously retrieved from the first swarm are no longer relevant in the newly joined swarm. Multiple swarm participation remedies this [3].

Chunk selection will be different than usual in this ABR/ P2P approach. Using earliest-first (EF) chunk selection is good for video. EF looks for the next required chunk and downloads it. This is different than a usual chunk selection protocol like rarest-first. Rarest-first (RF) looks for the most uncommon chunk in the swarm and downloads it. With streaming video RF has the potential to collect chunks in reverse order and the peer would have to wait until all the chunks are downloaded to begin playback. This defeats the purpose of streaming video, so a more sequential chunk selection like EF is used for streaming video. EF will decrease interruptions and improve video startup time.

Peer selection will have to change as well. Earliest-deadline (ED) peer selection chooses the peer with the highest need for the next chunk available [1]. Seeders select leechers to give chunks to. Tit-for-tat is a more popular peer selection strategy where peers will give chunks to each other. ED works better because it responds more to download urgency which for streaming video can decrease interruptions.

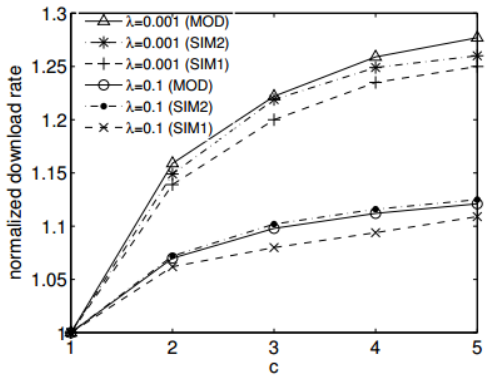

Figure 5: c is the number of videos cached, its increase leads to the increase in the normalized download rate.[1]

4.2 Caching Videos

The popularity of a video and the download rate for that video are correlated. Higher popularity of a video means there are many peers downloading and exchanging chunks of data with other peers. For a peer new to the swarm they have many peers to download from. For example, if there is one peer to serve many leechers it congests their upload rate and thus slows the download rate for the leechers. Adding another peer to seed relieves the upload congestion and improves the download rate of other leechers. By adding more seeders the download rate continues to rise. This is why high popularity videos have higher download rates. Now there is excess upload capacity among the seeders. Excess can be allocated to serve more videos.

Improving seed staying time will also improve download time. Seed staying time is the amount of time that a peer will seed a video in a certain swarm in comparison to the time spent downloading. If the peer spends time downloading but not seeding then the peer is not providing excess upload capacity for the swarm. If the peer were to stay in the swarm for longer they could help improve other peers download time. This adds incentive to keep a peer in a swarm for longer to strengthen the network.

To improve seed staying time the peer can cache the video to continue and serve it even after the peer has stopped watching the video. The number of cached videos can be set to a limit. For example if the limit of cached videos is

5, then when the peer moves to the 6th video it will drop the 1st video in cache and the newest video is cached in the now available slot.

Seeders have limited upload capacity. Therefore uploading multiple cached videos at a time will divide the upload capacity by how many cached videos there are.

Fig. 5 shows visually that caching videos can improve download rates. In this graph there are a couple of things that need to be described before they are discussed. Such as what is plotted on the x and y axis, λ, SIM1, SIM2, and MOD.

The x-axis plots c, which is the number of videos cached. The y-axis plots the normalized download rate, where 1 is no gain or loss on download rates. λ is the arrival rate of leechers. This is how fast new leechers are joining the swarm to download the content. λ can be thought of as popularity. When a video is more popular the λ will be higher because there will be more leechers looking to download the content. Less popular content will have a lower leecher arrival rate.

SIM1 and SIM2 in Fig. 5 are simulations. These simulations were tested using the protocols discussed earlier in this section. The simulations cache videos until it reaches c videos. When c videos are cached the newer video is cached and the oldest cached video is removed. The only difference between SIM1 and SIM2 is that SIM2 does not seed the cached videos while that peer is watching another video.

MOD in Fig. 5 is a mathematical model that is for constructing predictions for the simulation. That is to say if this model is accurate in plotting the simulation behavior it could be used to predict how ABR/P2P would act at any number of seeders, leechers, or even videos cached. The model inputs the size of a video in bits, the upload capacity of the peers in a swarm, popularity, average seed staying time, number of leechers and seeders, the playback rate of a video, and the number of videos the peers will be caching. The model for Fig. 5 is then able to output the normalized download rate for the peers in the swarm.

Fig. 5 shows the model is accurate enough in comparison to the simulations. All SIM1, SIM2, and MOD are very close when plotted and it can be said that MOD would be suitable for predicting behavior beyond what the simulations are capable of. For the analysis of Fig. 5 however it is only necessary to observe normalized download rate up to 5 cached videos since there is little improvement past that number. Adding more videos to the cache improves download rates, but the improvements are limited. A single video in the cache will see less of the peer's upload capacity dedicated to seeding it when there are other cached videos that need to get some portion of the upload capacity as well. If the number of videos is too high then it will only get a small fraction of the upload capacity and unable to benefit the video's swarm much at all to be of any use. This is why Fig. 5 only accounts for a maximum of 5 cached videos.

There are two groups to notice in Fig. 5, which are the higher λ and lower λ. This is to compare higher popularity videos and lower popularity videos and how each type of popularity would benefit from caching videos. The less popular the video the more it benefits from being cached. The high popularity videos still benefit from being cached but not as much as low popularity videos. This is not a concern because highly popular videos will have more seeders to provide extra upload capacity, but the less popular videos will need to get the extra upload capacity by being seeded from the cache.

Between SIM1 and SIM2 there is a small, but consistent difference. SIM1 has a slightly higher normalized download rate than SIM2. Because SIM2 is only seeding cached videos while not watching another video that leaves less time to seed. SIM1 can be seeding more since there is no restriction on seeding when viewing videos. This small difference adds a slight advantage in normalized download rates. Thus the more time a video is allowed to be seeded the higher the download rate is allowed to be.

5. CONCLUSION

There are multiple options to increase efficiency in streaming content over the internet. Combining CDN and P2P allows for higher scaling architecture. P2P adds to the scalability of CDN while also cutting costs by taking most of the responsibility of data sharing. By combining ABR and P2P it is possible to build a fully functioning adaptive streaming network on top of P2P. By caching videos the prerequisite of needing popularity to stream a video is no longer necessary. These technology combinations are an improvement on their individual parts.

6. ACKNOWLEDGMENTS

Thanks to Nic McPhee, Elena Machkasova, and Mike Maurer for their time, and helpful advice.

7. REFERENCES

[1] K.-W. Hwang, V. Gopalakrishnan, R. Jana, S. Lee, V. Misra, K. Ramakrishnan, and D. Rubenstein. Joint-family: Enabling adaptive bitrate streaming in peer-to-peer video-on-demand. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–10. IEEE, 2013.

[2] O. V. Joldzic, Z. I. Djuric, and D. R. Vukovic. Experiences and challenges in implementing adaptive bitrate multimedia streaming for live multimedia content. In *Telecommunications Forum Telfor (TELFOR), 2014 22nd*, pages 909–912. IEEE, 2014.

[3] D. Jurca, J. Chakareski, J.-P. Wagner, and P. Frossard. Enabling adaptive video streaming in P2P systems [peer-to-peer multimedia streaming]. *Communications Magazine, IEEE*, 45(6):108–114, 2007.

[4] A. Mansy and M. Ammar. Analysis of adaptive streaming for hybrid CDN/P2P live video systems. In *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, pages 276–285. IEEE, 2011.

[5] T. Stockhammer. Dynamic adaptive streaming over HTTP: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011.

[6] C. Wu, B. Li, and S. Zhao. Diagnosing network-wide P2P live streaming inefficiencies. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 8(1S):13, 2012.

[7] Y. Zhou, D. M. Chiu, and J. Lui. A simple model for analyzing P2P streaming protocols. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 226–235. IEEE, 2007.