P2P-Enhanced Content Serving for Hyperscale Short-form Video Applications

Julian Bell, Carson Garland

COMS 6998: Hyperscale Infrastructure, Columbia University

Abstract

The rise of short-form video platforms such as TikTok and the growing utilization of mobile networks for content consumption have underscored the need for efficient and scalable content delivery mechanisms. This paper investigates the feasibility of integrating peer-to-peer (P2P) content sharing with client-server networks in mobile networks to optimize short-form video streaming performance. We present a library created to simulate this hybrid service and measure its efficacy against the client-server model in different conditions. Overall, we find that P2P is most effective in environments where smaller amounts of content are distributed (e.g. tens of videos) and network bandwidth is low. These findings point to the drawbacks of broadly implementing P2P for short-form video distribution, but indicate that it could have utility in poorly serviced areas or a subset of viral videos. We also analyze the inherent challenges to integrating P2P into a service like this, including resource constraints and dynamic network topologies characteristic of mobile devices.

1 Introduction

The recent surge in short-form video content, fueled by platforms such as TikTok, has reshaped online content consumption patterns and ushered in a new hyperscaled service. As of 2021, video traffic accounted for 69% of the total mobile network data traffic, and is predicted to rise to 79% by 2027[7]. Competing services—including Instagram, YouTube, Facebook and others—have embraced this format, creating a burgeoning ecosystem of ultra-short videos designed to engage users within seconds. These videos are often personalized to individual preferences through proprietary "black box" algorithms and exhibit a high propensity for virality, particularly within niche audiences.

The integration of P2P networks is particularly advantageous in scenarios where multiple users are accessing the same content within overlapping time windows. P2P technology has been used at a large scale to distribute music and video, share files, transfer payments, and help perform other services. In these cases, the first client to retrieve the content can serve as a peer for subsequent users, thus alleviating the load on centralized servers and reducing overall delivery time. Additionally, this model is especially effective for distributing smaller content files, as the limited size requires fewer peers to achieve efficient dissemination. The P2P component is typically supported by a caching mechanism wherein client devices temporarily store small portions of content. This cached content facilitates redistribution to other peers, thus enhancing network resilience and efficiency. Given the rapid consumption and redistribution patterns inherent in short-form video content, exploring the integration of peer-to-peer (P2P) networks within hyperscale services' client-server content delivery system presents a compelling research opportunity.

This study proposes to investigate the impact of hybrid P2P and client-server architectures on the distribution and caching strategies employed by hyperscale platforms for short-form video content. This dual approach leverages the strengths of each paradigm to address specific challenges associated with content distribution. Specifically, it aims to

evaluate how such a model affects delivery quality of experience (QoE), considering the unique consumption patterns and rapid dissemination associated with short-form videos. QoE for short-form videos is defined by video quality, re-buffering, and start-up delay [4]. Our research will touch on the last two, as these are the metrics impacted by time to fetch content.

Key considerations include the suitability of P2P mechanisms for handling the viral nature of such content, the optimization of caching strategies to balance storage constraints and retrieval efficiency, and the overall implications for network performance and cost-effectiveness. In particular, many of these videos are consumed on mobile devices, which have smaller cache sizes and more unstable network connections than desktop clients. By examining these dynamics, this research seeks to contribute to the development of innovative and scalable content delivery solutions for the next generation of digital media platforms.

Our hypothesis is that the addition of a P2P network with optimal strategy can yield better latency and lower server load than just a client-server connection to serve short-form video content. In this study, we present a P2P client-server hybrid service simulation to test its efficacy. In Section 2, we provide a brief overview of existing content delivery systems, and outline our simulation. In Section 3, we summarize our experiments and their results. In Section 4, related work is explored. Section 5 discusses possible follow-up projects, and the benefits and shortcomings of hybrid services.

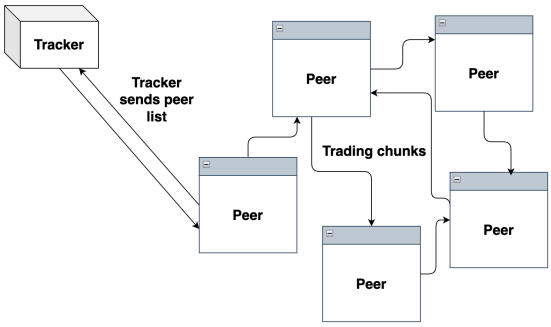

Figure 1: P2P network

In a P2P network, workload is divided among peers. Peers that share resources are grouped in a unit, often called an *overlay*, usually by some structuring principle like geographic locality, or randomly in an unstructured overlay. There is little to no intervention from a centralized server in a P2P network, although there may be a tracker server that coordinates peers, e.g. by sending nodes their nearby peers, as depicted in Fig.1. P2P networks can offer fault tolerance and scalability through distributed resource allocation, but they can have the drawbacks of higher maintenance overhead and high lookup times if not properly calibrated [13].

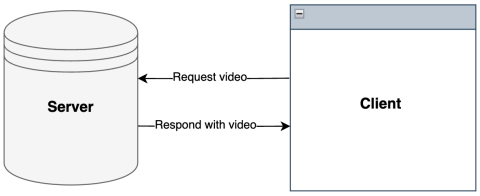

Figure 2: Client-server

2 Experiment

2.1 Overview

A range of different systems and networking topologies have been used to tackle the problem of content distribution. In this section we will discuss the three most prominent, along with their respective strengths and weaknesses.

This is contrasted by the client-server model, seen in Fig.2, where a central server coordinates all content distribution, often also working with an edge servers placed in eyeball networks. This is the traditionally design of hyperscaler's content delivery networks; giants such as Netflix and Google implement this route.

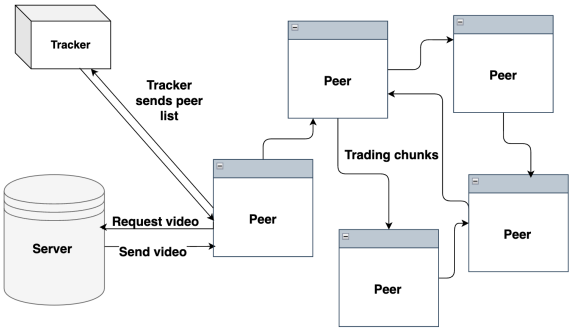

Figure 3: Hybrid network

This paper focuses on a hybrid system, combining elements of traditional client/server models with P2P elements to optimize video streaming, as shown in Fig.3. When combined with client-server networks, P2P systems can assist in serving content to users, and have the potential to lower overall latency, especially when serving cached content [11]. In this hybrid set-up, nodes can request video data from a central server or from peers in the network, enabling a balanced trade-off between centralized control and decentralized resource sharing. This architecture reduces server load while leveraging the collective caching power of peer nodes. We investigated using an existing, publicly available P2P library for this experiment, but could not find one that was clearly documented and written in a high-level programming language that would allow for easy customization.

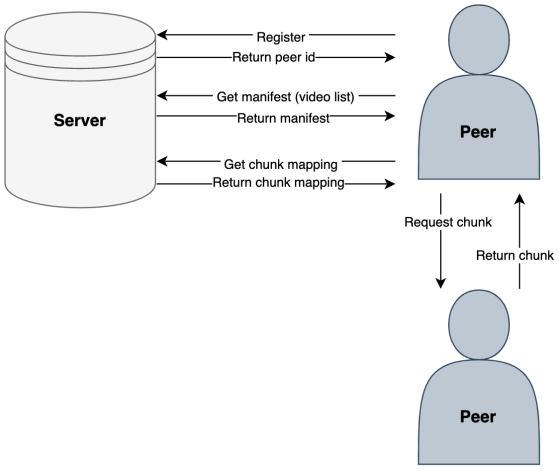

Figure 4: System architecture

The system diagram is depicted by Fig.4 and shows the complete data flow. A collection of cache eviction polices are explored including: Least Recently Added (LRA) and Least Recently Used (LRU). Cache diversity is achieved as nodes independently cache the chunks they fetch or serve, and nodes only cache specific chunks of videos (determined by a node's unique id) to prevent significant cache redundancy.

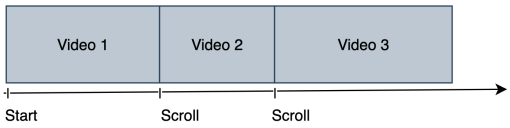

Figure 5: Session model

2.2 Experiment architecture

2.2.1 Nodes and Servers

Our system relies on peers and servers to distribute video information. A node registers itself with the server and fetches the manifest of available videos and a list of peers. When a node requests a video, it first fetches the chunk-to-peer mapping. Then the node prioritizes fetching chunks from peers if they are available, falling back to the server only when peers cannot provide the requested chunks. Simultaneously, as videos are played, nodes continuously add received chunks to their cache and evict older chunks as needed.

The node and server were developed with features to mimic their behavior in a production environment. Nodes simulate video playback by continuously buffering video chunks while handling re-buffering and playback latency. Buffering logic ensures uninterrupted playback unless chunks are delayed. In a short-form video platform, content changes are very quick, with some estimates as low as every 12 seconds [16]. A typical user session, where scroll actions signal the next video to play, is shown in Fig.5. Since users often switch to new videos, low start-up delay is especially important for QoE. Network conditions, including latency and bandwidth, are simulated with random variations to model real-world conditions. Delays are calculated based on the size of the data being

transmitted and the node's bandwidth, approximated from existing benchmarks found in other research.

2.2.2 Default parameters

To determine setting default parameters for variables, we investigated how these values are set in experiments and production, when possible. The average size of a video in our experiment is 22 MB, calculated by considering that an average TikTok length is 35 seconds, 1080p resolution, and 5 MBps. The cache size allocated to video chunk caching is 125 MB, based on an analysis of cache sizes for mobile apps, which can be over 1 GB for data intensive apps, and a reasonable amount of cache space that could be allocated for a P2P distribution scheme. The latency per node in P2P distribution is on average 0.1 seconds, with actual latencies normally distributed around this median.

3 Results

During the evaluation stage, we prioritized two key metrics: playback latency (synonymous with start-up delay) and server load, with the former measured in seconds and the latter in bytes per second (Bps). These metrics were selected to align with the practical concerns and performance indicators of real systems, ensuring that our findings would be both relevant and actionable. Playback latency was chosen to measure the end-user experience, as reduced latency is critical for maintain seamless content delivery in video streaming. Server load served as a proxy for resource efficiency, reflecting the potential cost savings and scalability improvements achievable through our approach, especially since P2P might be attractive for services that are quickly growing but do not have the corresponding infrastructure.

The primary goal of our initial experiments was to explore the conditions under which integrating the P2P service would provide tangible benefits, if any. Specifically, we sought to determine whether the hybrid P2P and client-server model could successfully alleviate server load while maintaining or improving playback latency for end users. Through systematic testing, we aimed to identify scenarios where a P2P integration could add value.

3.1 Server Load

Initially, we tested how the load placed on the server (measured number bytes handled by the server) would change with and without P2P aid while varying the number of videos in the set — 10, 50, and 100. The system included a singular server with an average latency of 100 ms and bandwidth of 100 MBps, and ten peers with average latency 100 ms and bandwidth 10 MBps. Fig.6 shows the results of this experiment with the legend listing the six configurations of simulated.

A quick note: when reading the figures presented in this results section, the color represents the size of the video set and the line style represents the inclusion or exclusion of P2P (dotted and solid, respectively).

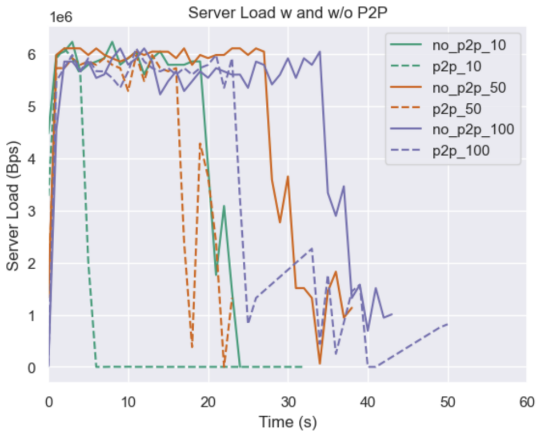

Figure 6: Server Load with variable bandwidth

There are couple observations of note that arise from the experiment depicted in Fig.6. We find that there is a substantial savings when using P2P across all sizes of video sets - evidenced by shift leftward of the of the dotted lines compared to their solid counterpart. P2P appears to provide a greater impact to server load for small video sets. To demonstrate this, consider when the video set size is 10 (the set of green lines), the server supported by P2P experiences a drop in server load at around 5 seconds into the simulation while the server without P2P help sees this drop at 20 seconds - a massive 75% decrease in total server load. However, for a video set of size 100, there is only a approximately 25% decrease.

This trend of diminishing reductions in server load with a growing set of videos is due in part to users being unlikely to request the same content

as their peers. However, when we take into consideration how these short-form video applications choose and recommend content to users, it is possible for them produce an artificially small set of the most popular videos, thus maximizing the savings in server load that P2P is able to provide (in the case of our experiment, a up to 75% reduction in server load over its no-P2P counterpart).

3.2 Start-up Delay

In subsequent simulations, we shifted focus to investigate our other key performance metric, start-up delay. When looking at start-up delay in the same set-up used in the discussion of server load (a variable sized set of videos), we observed that P2P reduces start-up delay in all sizes of video set but becomes less effective as the sets grow. These findings can be seen in Fig.7, with the dotted lines representing the results of the hybrid-P2P system consistently shifted left of their traditional client-server comparison. With larger numbers of videos, the gap between them narrows.

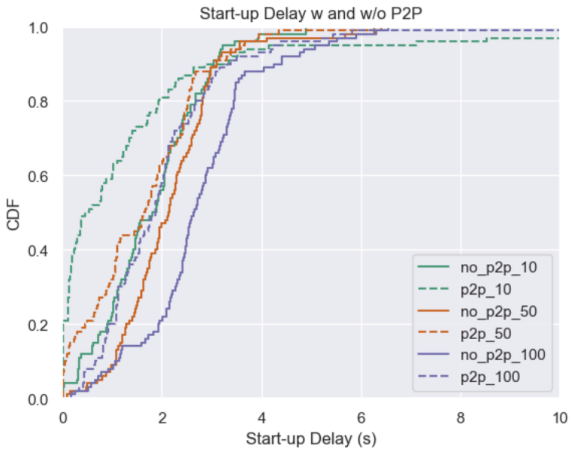

Figure 7: Start-up Delay with variable video sets

This is a result of it being less likely that a peer will have the video a user wants when the number of videos increase. As such, the users are increasingly forced to fall back to requesting chunks from the server, thus approaching the delay of a purely client-server system. Once again, due to the unique characteristics of short-form video applications that control most of what the user sees, the timeline of each user could be tailored to cultivate a minimally-sized video set to reap the most gains in start-up delay.

We additionally investigated the effects of a varying the quality of the server connection (the bandwidth and latency of the bottleneck link). This was simulated via injecting artificial delay into the system in accordance with a semi-random throughput and latency value selected on a per chunk basis. The links between peers were set to be significantly less than that of between peer and server (around 10MBps) as this most likely reflects the real-world conditions.

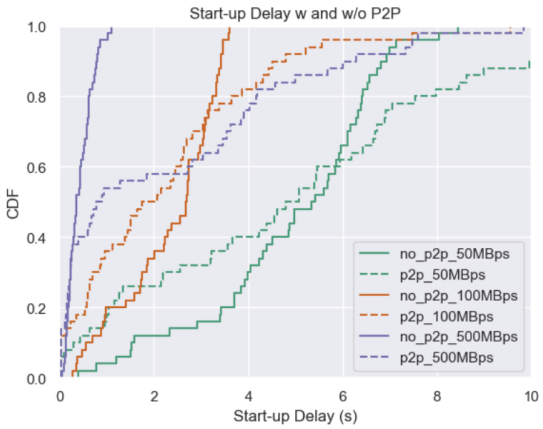

Figure 8: Start-up Delay with variable bandwidth

Fig.8 depicts the resulting start-delay CDFs of three different server bandwidth simulation runs. When looking at the pair of green lines corresponding to a server link average throughput of 50MBps, we find that the two different systems have a similar mean start-up delay around 5.5 seconds. The P2P implementation does outperform for the first quartile with around 40% of videos starting to play under 4 seconds in comparison to the no-P2P's 30%. However, as the bandwidth of the server link increases to 100MBps and finally to 500MBps, the traditional client-server model begins to slightly and then drastically outperform the hybrid system. At the top end, the no-P2P implementation has a maximum delay of just over 1 second while P2P has 40% of videos with a start-up delay of over 2 seconds.

This is likely because the bottleneck link shifts from being from a certain user to the server to being from the user to the other peers - alluding to the existence of a tipping point in server throughput potential in which P2P becomes hurtful rather than helpful. On the opposite side, if an application was targeting undeserved areas, without numerous data centers that provide high-

bandwidth connections, the hybrid-P2P architecture could provide users with a better experience via lower start-up delays.

The final experimental factor that we varied was the cache eviction policy. We implemented both the common Least Recently Added aka FIFO (LRA) and Least Recently Used (LRU) methods before comparing their performance using our default P2P system. The results are shown in Fig.9.

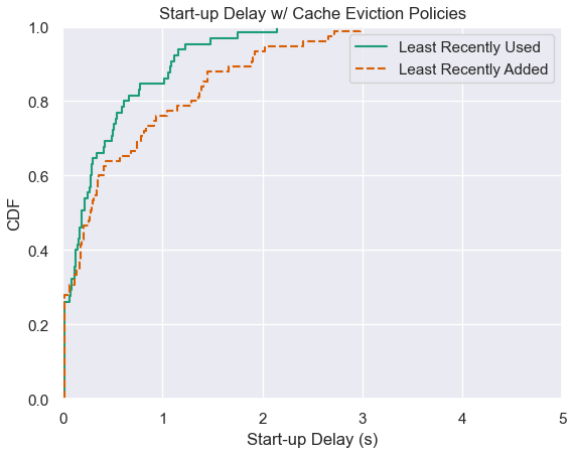

Figure 9: Start-up Delay with variable cache eviction policies

We found that LRU outperforms LRA by an entire second at the 99th percentile. This comes unsurprisingly as LRU is the widely accepted solution for cache eviction but the result is important nonetheless as it confirms its effectiveness to enable better performance out of the relatively small buffers of peers in a hybrid-P2P system.

A result that cannot be well quantified, however, is the cost of overhead that comes with the implementation of a P2P system. Even within the work that this paper is built on, there was a disproportionate amount of time put into developing the very simple P2P network with its client-server counterpart was trival. As such, this is as important a result as any other. It may reflect why hybrid-P2P content delivery networks have not seen large adoption in practice.

In sum, we find that, at the cost of operational overhead, hybrid-P2P systems are able to reduce both server load and start-up delay under the conditions that the video set is relatively small and the server bandwidth is moderately low - both of which could potentially hold for short-form video applications in the wild.

4 Related work

Our work greatly benefited from a general literature review of short-form video applications, P2P networks, and their respective simulators. This section will explore some of what was found.

Similar simulation services There are numerous published P2P simulation experiments. We evaluated surveys and existing studies of similar libraries designed for experimental purposes. This work illuminated many of the parameters we would consider for our own system, such as cache size, video size, and server upload bandwidth [14;13]. P2P systems exhibit a diverse range of architectural designs, with no single standardized model prevailing across all implementations. In the context of music distribution, for instance, the relatively small size of audio files makes it feasible for peers to cache and distribute entire songs, allowing peers to get a whole song from a single peer [10]. In contrast, video files, due to their significantly larger size, present unique challenges. It is often impractical for a single peer to store an entire video within its cache, necessitating more sophisticated strategies for chunking and distribution [11].

Simulations that incorporate a social graph further diversify P2P architecture by enabling peer connections based on pre-existing social relationships. This approach not only facilitates efficient data sharing but also capitalizes on the likelihood that individuals with social ties are interested in similar content [14]. Key technical parameters, such as cache eviction policies (for ex., FIFO vs. LRU) and chunk request algorithms (for ex., rarest-first vs. greedy strategies) are tailored to align with the service's specific goals and user behaviors. Rather than converging on a universally optimized standard, services deliberately customize their architecture and operational parameters to address their unique requirements and challenges.

Using P2P in practice By examining real-world applications of P2P, we aimed to shed light on both the potential and the limitations of P2P technology in contemporary content distribution. Our investigation extended to understanding how large-scale companies leverage P2P technology to enhance or supplement their content delivery

frameworks. One original case study was Spotify, which used P2P to supplement their music delivery to users and delivered less 9% of songs in 2010 [10]. However, a few years later, the company expanded its server infrastructure and found that its P2P feature added overhead, due to the time it took for clients to find peers and send or receive songs from them [6]. There are additional issues with using P2P in mobile networks, which are explored in more detail in the next section. Currently, there are existing frameworks that enable P2P communication on mobile devices in practice. For instance, Android has a service called Wi-Fi Direct, which allows compatible devices to tether directly over a Wi-Fi network without any other access point and transfer data over greater distances than Bluetooth connectivity allows [1]. Further, Meshtastic is an open source mesh network built for low-power devices to communicate via P2P [12].

Short-form video content Given the recency and discretion of short-form video platforms, there is not a wealth of research about these platforms in practice. However, we were able to use the results of one study on real data from an anonymous, top-10 short-form video platform based in China [16]. Additionally, other studies without platform cooperation studied these services indirectly, either through quantitative analysis or observation [4;9].

A 2021 analysis found that commercial short-form video platforms use a streaming approach, sequentially downloading videos to the user's device, without optimization for network resources [4]. Further, since the video data is often immediately discarded with only 45% of the downloaded data being viewed, there is an excess of waste [4]. There are many proposed methods or enhancements to more effectively pre-fetch short form videos. Since the next video can be expected to play at any time when a user swipes away, pre-fetch strategy is integral to user experience. However, since unwatched video chunks constitute almost half of all fetched data, adaptive pre-fetching strategies, where videos are pre-fetched based on a reinforcement learning algorithm, or algorithms that dynamically pre-fetch segments of the video based on network conditions to reduce wastage, have been documented in the literature [4]. Another study proposed a time-aware pre-fetch approach, where the video with the shortest expected playing time is pre-fetched first among some candidate videos, to reduce the risk of start-up delay [8]. Most similar to our approach is a proposal to cache popular videos on edge servers close to users, reducing traffic in the larger network [2].

Two of the most relevant areas of study are the related issues of the "black box" recommendation algorithms, and the nature of virality on these services. The study on the anonymous Chinese service found that, in contrast to longer-form video services like YouTube, the viewing patterns for short-form videos do not obey Zipf's law, an exponential power-law distribution where access frequency is CR^{β} (a normalizing constant times the video rank, raised to Zipf's coefficient) [16]. Due to the nature of short-form video viewing, videos can become viral within minutes, decay in hours, and are rarely re-watched by the same viewers over a longer timespan, unlike longer-form videos. Additionally, videos can go "viral" within a smaller niche of users, due to algorithmic feeds. Understanding the nature of user segmentation and virality on these platforms is important to most effectively develop a caching strategy, as it would help determine which videos to cache, which chunks of those videos to prioritize (if any), and even potentially how to divide peers into an overlay.

5 Future development

5.1 Potential library enhancement

Based on our findings and emerging trends, incorporating certain features into future experiments could enhance the performance of the client-server/P2P hybrid. Our library is a basic simulation and could be significantly enhanced to better reflect actual production conditions.

Further optimizations include adding more sophisticated features that model more closely the known behavior of users engaging with short-form video platforms, and adapting our cache strategy to mirror that. For instance, one study found that users only watch the first 12 seconds of a video on average [16]. By abruptly stopping playback of videos at arbitrary points before they finish, this behavior could be represented in the simulation. Following this, prioritizing caching initial chunks of videos could reflect this behavior, and reduce start-up delay.

Additionally, integrating a notion of popularity or

virality into the cache policy could optimize resource utilization by focusing on storing and sharing content with higher demand, reducing redundant requests. As mentioned in the previously, viral videos on short-form platforms do not have as severe viewership spikes as longer-form videos, but they still represent significant outliers. Currently, our cache used two different greedy strategies (LRA and LRU), where the last item would be evicted to make room for a new chunk, regardless of its content. Ranking chunks by popularity (i.e. their likelihood that other users would request them) could potentially improve hit rate. Studies have found that it is possible to predict a video's virality potential before it has even reached peak visibility, which could inform rank as well [9].

It would also be accurate to add a social graph component, as these services allow users to follow creators and be followed in turn. This social graph would inform the videos that users are served. It could also help determine peer groups, as users who follow each other likely have similar interests, as this strategy was effective for a social P2P service, as mentioned in the previous section [14].

One of the most significant differences of our service, which selects the next video randomly from the manifest, as opposed to a real TikTok-like platform is the lack of a personalized algorithm that determines the next video viewed. A significant barrier to creating a fully lifelike simulation is due to the fact that the exact strategy of current services is not known. There are many likely methods that are employed, but the limiting factor for implementing many of these strategies is user data, which is used to form a portrait of a user's preferences and create interest categories for them. Due to the amount of first-party data these companies have access to, it would be difficult to implement these features in a restricted experimental setting. This detracts from the verisimilitude of any simulation, as one of the defining features of these platforms is their hyper-personalized algorithmic selection, so it is central to capturing the user experience [9].

Finally, experimenting with varying cache sizes based on device capabilities could reflect the trend that smartphones increasingly have more memory and processing power available.

5.2 Limitations of P2P on mobile networks

As mentioned in the previous section, there are multiple factors that make P2P difficult to implement at a practical scale in mobile networks. Mobile environments introduce additional complexities such as highly inconsistent participants, variable network reliability, limited bandwidth, and hardware constraints, which must be accounted for in the design and implementation of P2P systems. In particular, the quickly shifting movement of mobile devices across mobile networks, called ad hoc mobile networks, makes the topology of P2P peer groups highly unstable. However, attempts to mitigate this have been made through constructing novel ways to assemble peer overlays in mobile ad hoc networks, such as measuring intra-neighbor distance of peers, calculating a logical identifier, and using this to assemble groups [15], although several have been proposed and studied [13].

Additionally, privacy concerns in P2P networks, particularly on mobile platforms, are significant. Because P2P networks inherently distribute data between many devices, the risk of exposing sensitive information if intercepted by bad actors increases. Mobile networks exacerbate these risks because devices frequently connect to multiple, often insecure, networks like public Wi-Fi or cellular data with varying levels of encryption or security. Additionally, mobile devices often share location data, contact lists, and other information that could be unintentionally exposed during P2P interactions [3]. As mobile P2P usage grows, addressing these privacy challenges will be crucial to fostering user trust and ensuring secure communications.

The mobile market continues to experience growth, particularly in developing regions where mobile devices are often the primary means of accessing the internet [5]. This rapid growth underscores the critical need for efficient networking solutions. In these regions, where infrastructure is often resource-constrained, techniques like peer-to-peer sharing and optimized caching strategies may find a foothold. Developing scalable and efficient networking protocols will be paramount to supporting the expanding mobile user base and meeting the growing demand for high-quality, seamless digital experiences.

It's possible that advancements in mobile hard-

ware, consolidation of the mobile market, and smarter resource management techniques, will create potential to overcome, or at least minimize, the current challenges. The services enabling P2P communication on mobile phones mentioned in the last section point to the appetite for this kind of network communication.

6 Conclusions

In this paper, we have explored the challenges and opportunities associated with implementing a hybrid client-server/P2P network in the context of modern short-form video platforms. While the inherent resource constraints and dynamic nature of mobile ad hoc networks pose significant barriers to practical scalability, advancements in hardware, dynamic caching policies, and the potential for leveraging network virality present avenues for future exploration.

Our results found that P2P gains are more efficient with a small amount of videos. This meshes with the general "long tail" viewing pattern of social content with short form videos in particular have high-frequency, short-lived virality cycles. Thus, this hybrid system is a valuable mechanism that could be ideal to distribute the small amount of hyper-viral videos that are likely to be served to huge swaths of the user base.

However, as the benefits of P2P are nullified when server bandwidth increases past a certain threshold, there are limited use cases for P2P to be used by hyperscalers in practice. However, there are niches where this practice could be useful. Namely, these scenarios could occur when companies 1) are in a lean, high-growth phase, 2) are serving user groups who are in particularly low-bandwidth areas - e.g. poor network infrastructure, geographically distant from server, 3) are distributing low-weight highly-consumed content, e.g. short, viral videos. In sum, while integrating P2P may not be a one-size-fits-all solution for widespread video distribution, its strategic implementation in specific scenarios could offer significant benefits, particularly for emerging, lean platforms with unique challenges and opportunities.

7 Acknowledgments

Thank you Professor Nieh and Yoannis for a great semester!

References

[1] Create p2p connections with wi-fi direct. https://developer.android.com/develop/connectivity/wifi/wifi-direct. Accessed: 2010-09-30.

[2] F. Chen, P. Li, D. Zeng, and S. Guo. Edge-assisted short video sharing with guaranteed quality-of-experience. *IEEE Transactions on Cloud Computing*, 11(1):13–24, 2023.

[3] Yung-Ting Chuang and Jia-Zong Jhang. Trustworthy retrieval system in mobile p2p wireless network. *Ad Hoc Networks*, 154:103369, 2024.

[4] V. Long T. T. Huong D. Nguyen, P. Nguyen and P. N. Nam. Network-aware prefetching method for short-form video streaming. *IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, 2022.

[5] Digital around the world. https://datareportal.com/global-digital-overview. Accessed: 2024-12-19.

[6] Romain Dillet. Spotify removes peer-to-peer technology from its desktop client. https://techcrunch.com/2014/04/17/spotify-removes-peer-to-peer-technology-from-its-desktop-client/, April 2014.

[7] Ericsson mobility report. https://www.ericsson.com/4ad7e9/assets/local/reports-papers/mobility-report/documents/2021/ericsson-mobility-report-november-2021.pdf. Accessed: 2024-12-19.

[8] S. Fu, G. Yang, Z. Yao, S. Tan, Y. Shan, and W. Jiang. Pacs: Playing time-aware chunk selection in short video preloading. *Electronics*, 13(24):4864, 2024.

[9] D. Klug, Y. Qin, M. Evans, and G. Kaufman. Trick and please: A mixed-method study on user assumptions about the tiktok algorithm. pages 84–92, 2021.

[10] G. Kreitz and F. Niemela. Spotify – large scale, low latency, p2p music-on-demand streaming. pages 1–10, 2010.

[11] Z. Liu, Y. Ding, Y. Liu, and K. Ross. Peer-assisted distribution of user generated content. pages 261–272, 2012.

[12] Meshtastic: An open source, off-grid, decentralized, mesh network built to run on affordable, low-power devices. https://meshtastic.org/. Accessed: 2024-12-19.

[13] Nadir Shah, S.A. Abid, Depei Qian, and Waqar Mehmood. A survey of p2p content sharing in manets. *Computers & Electrical Engineering*, 57:55–68, 2017.

[14] H. Shen, Z. Li, Y. Lin, and J. Li. Socialtube: P2p-assisted video sharing in online social networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2428–2440, Sept 2014.

[15] Ali Tahir, Nadir Shah, Shahbaz Akhtar Abid, Wazir Zada Khan, Ali Kashif Bashir, and Yousaf Bin Zikria. A three-dimensional clustered peer-to-peer overlay protocol for mobile ad hoc networks. *Computers & Electrical Engineering*, 94:107364, 2021.

[16] Y. Zhang, Y. Liu, L. Guo, and J. Y. B. Lee. Measurement of a large-scale short-video service over mobile and wireless networks. *IEEE Transactions on Mobile Computing*, 2022.

8 Appendix

8.1 Code Reference

To see our simulation's code along with the logic to generate the plots found in this paper, visit this Github repository: https://github.com/jawblet/p2p. There is no doubt many opportunities for improvement, so feel free to contribute! If any questions arise, please feel free to reach out to either of the authors.

8.2 Experiment Reference

Figure Number	Variable	Metric(s)
6	Number of Videos	Server Load
7	Number of Videos	Start-up Delay
8	Bandwidth to Server	Start-up Delay
9	Cache Eviction Policy	Start-up Delay