SCALING PEER-TO-PEER VIDEO-ON-DEMAND SYSTEMS USING HELPERS

Hao Zhang[†], *Jiajun Wang*[‡], *Minghua Chen*[§] *and Kannan Ramchandran*[†]

[†]Dept. of EECS
Univ. of California, Berkeley
Berkeley, CA 94720, USA

[‡] NVIDIA Corp.
Santa Clara, CA 95050, USA

[§]Dept. of Information Engineering
The Chinese Univ. of Hong Kong
Shatin, NT, Hong Kong

ABSTRACT

The throughput of Peer-to-Peer (P2P) Video-on-Demand (VoD) systems is typically capped by the users' aggregate upload bandwidth [1]. The drastic increase in the popularity of VoD and the demand of higher quality content has thus placed substantial burden on the content servers. We investigate a novel P2P VoD architecture that leverages idle internet resources, which we call helpers, to provide a scalable solution to P2P VoD systems. Helpers are volatile in nature, and can be individually unreliable. However, we investigate the statistical aggregation of a large number of helpers to guarantee quality of service. Since helpers do not come with "free" preloaded content, trade-offs between how much a helper should download and how much it can aid the system need to be explored. In this paper, the optimal steady-state design parameters are derived to maximize the helpers' upload bandwidth utilization. Packet level simulations have verified the efficiency of the system. In a typical scenario of 240 users and a required theoretical minimum of 120 helpers with an average upload bandwidth of 256 kbps, a streaming rate of 384 kbps can be sustained with $< 2\%$ relative server load. Results also show that the system is robust to helper churn.

***Index Terms*—** Video communications, video-on-demand, peer-to-peer, helpers

1. INTRODUCTION

Internet VoD has seen an explosive growth over the past several years [1]. The overall global VoD market is expected to expand from below 2 billion dollars in 2002 to about 13 billion dollars in 2010, by which time there will be nearly 150 million active VoD users worldwide. At the same time, the upsurging need for higher quality content has driven the upgrade in content from conventional low quality videos to SD or even HD quality videos. In fact, these high quality videos are already available at YouTube, Hulu etc. Due to the significant increase in demand, providing these VoD services with server-client infrastructures has become very costly. It has been reported that the major Internet VoD publishers including YouTube are paying a growing cost of bandwidth usage currently estimated at millions of dollars per month to internet service providers and content distribution networks [1].

P2P video streaming, on the other hand, has been shown to successfully reduce the burden of content providers [1, 2, 3, 4]. In P2P VoD, the server (or sever farms) will need to make up the difference in streaming rate to guarantee smooth video playback only when the users cannot by themselves redistribute the content. However, since a large number of internet subscribers have asymmetric upload/download bandwidth connections, the throughput of the P2P VoD system is typically capped by the aggregate upload bandwidth of the participating users [1]. As a result, the drastic increase in the required streaming bitrate of high quality content will surpass what is sustainable by the users' upload bandwidth, placing substantial burden on service providers and compromising the system scalability.

This motivates us to explore new paradigms in collaborative content distribution that leverages the use of internet "helpers" in large scale VoD systems. Helpers are idle internet users who are not interested in the content but have spare resources to share. An important feature that distinguishes helpers from infrastructure nodes is their individual high dynamics, i.e., helpers may frequently join and leave the network without being dedicated to guaranteed service. However, one can leverage the strength of numbers to sustain a reliable system. In addition, helpers oftentimes do not come with preloaded content, and they need to consume the system resource before being able to help. Therefore, it remains to study *what* and *how much* the helpers should download, and what *system architecture* should the helpers maintain. The goal of this work is to efficiently utilize these helpers to improve VoD quality while maintaining minimal server load even when helpers are highly dynamic, providing a salable solution to P2P VoD systems. Although the question of incentives for the helpers remains as future work, existing mechanisms such as that designed by Pouwelse et al. [5] could potentially be adopted for the proposed system.

In this work, a novel helper-assisted P2P VoD architecture is presented. Using steady-state analysis, the optimal system parameters are derived as to *what* and *how much* the helpers should download and *how many helpers* are needed to maintain a self-sustainable system. The design is shown to yield the minimum number of helpers needed and maximum utilization of helpers' resources. Comprehensive simulation results have further corroborated the efficiency of the system. Although we focus on steady-state analysis as a first step towards understanding this complicated problem, we believe this work would provide some useful insights to the design of scalable P2P VoD systems. Extending work to higher order analysis and optimization based rate allocation strategies remain part of our ongoing and future work.

2. RELATED WORK

There are various works that introduce the notion of helpers in content delivery systems [2, 6, 7]. Of particular interest is the helper-assisted P2P live video streaming system proposed by Wang et al. [2]. In live streaming, as in a video stream of a live sport event, the video content is available only at one particular time and its users have synchronous (or loose synchronous) playback times. Wang et al. proposed that each helper downloads only one coded packet of the segment that is currently being streamed. Simulation results showed that the proposed system can achieve significantly improved streaming bitrate without incurring additional server load.

On the other hand, VoD streams are often preloaded on the server, and are available to be transmitted at users' requests. Since users can start watching the video at arbitrary time stamps, they often have asynchronous playback times. This imposes new design challenges with regard to *what* and *how much* the helpers should download to satisfy different users' needs while maximizing their own utility. Our work differs from that of Wang et al. [2] on several counts. First, we include helpers' arrival/departure processes in the model. Second, we use steady-state analysis to derive the minimal helpers' download to maximize their contribution given other system parameters. Third, we show that our design can achieve the minimum number of required helpers.

3. SYSTEM DESCRIPTION

3.1. Design Overview

Table 1 lists some useful notations. For convenience of discussion, we assume in this paper that users start watching the beginning of the video, and leave the system immediately after the video ends. We also divide the video into segments of equal length u each consisting of k equal packets of size s, where a packet is the smallest processing unit. In the following analysis, we assume that the users can fully utilize their upload bandwidth by assisting each other. This can be verified to be almost achievable as will be shown in the simulation results. To increase the number of users that each helper can potentially assist, we divide the helper's download d_h into m segments with one and only one packet per segment such that $ms = d_h$. In this way, a helper can maximize the number of users that may need its assistance, which facilitates full utilization of its upload bandwidth.

Notation	Definition
r	video streaming rate (kbps)
l	video length (seconds)
$\lambda_u(\lambda_h)$	average user (helper) arrival rate
τ_h	average helper sojourn time
$b_u(b_h)$	average user (helper) upload bandwidth (kbps)
u	video segment length (seconds)
k	number of packets per video segment
s	packet size (kb)
d_h	helper download amount (kb)
m	number of helper downloaded packets, $m = \frac{d_h}{s}$

Table 1: Notations

It is worth noting that if the helpers carry uncoded data packets, the probability that the users will be missing these particular packets at each time instant can be very small. On the other hand, there may be other particular packets that many users will need. Therefore, it would also be desirable that the helpers download coded parity packets [8] which are equally useful to all the users in need of the corresponding segments.

Based on the above assumptions and guidelines, we give an overview of the proposed solution demonstrated in Figure 1. The server breaks the video into chunks $C_i, i = 1, 2, \ldots, N$, each consisting of m segments of equal duration u, where $Nmu = l$. Each segment contains k packets of size s such that $r = \frac{ks}{u}$. The server also generates $n - k$ parity packets per segment using an (n, k) Maximum-Distance-Separable (MDS) systematic erasure code [8]. The helpers form N separate swarms, i.e., $S_i, i = 1, 2, \ldots, N$, each serving one corresponding chunk C_i. Within each swarm S_i, helpers further break themselves up into $n - k$ clusters with γ helpers each. Each cluster downloads and circulates one and only one unique parity packet in every segment for a total of m segments. The parity packets in different clusters are mutually exclusive.

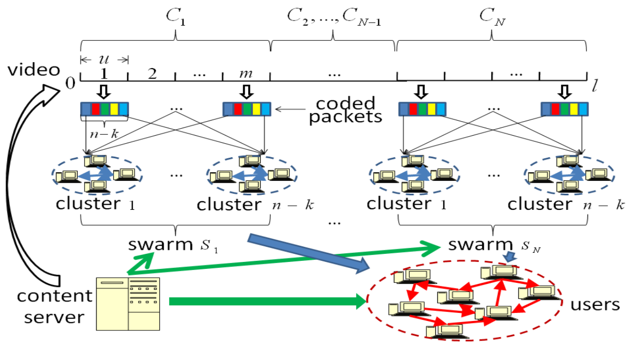


Fig. 1: Proposed helper-assisted P2P VoD system. Helpers break into N swarms each serving one chunk of video. Each swarm consists of $n - k$ clusters. Helpers within each cluster download one coded packet for every segment for a total of m segments.

3.2. System Parameters

Based on the proposed scheme, we now answer the question of, given the video information r, u, k, s, l and the user/helper statistics $\lambda_u, b_u, \lambda_h, \tau_h, b_h$, how should one choose the number of clusters $n - k$ and the number of helpers per cluster γ such that the system can be self-sustainable. We also derive the optimal d_h that maximizes helpers' contribution during their sojourn.

3.2.1. *Number of Clusters*

The number of helper clusters should be able to minimize the server load while sustaining the required streaming rate. Since an average of $\frac{(r-b_u)u}{s}$ unique parity packets are needed in each segment, a reasonable choice of $n - k$ is $n - k = \frac{(r-b_u)u}{s}$.

3.2.2. *Helper's Download Amount*

Denote by U_{ave} the average upload rate (in kbps) of a helper during its sojourn. In steady state, a helper's contribution is $C_{ave} = U_{ave}\tau_h - d_h$. To maximize C_{ave}, the optimal d_h, denoted by d_h^*, should satisfy the following equation:

$$d_h^* = \frac{b_h \tau_h}{1 + \lambda_u \tau_h} \quad (1)$$

We first derive U_{ave} as follows. Using the proposed system, each helper downloads one parity packet per u seconds for a total of mu seconds. In steady state, there are $\lambda_u mu$

users within that time span. It follows that a helper's unique parity packets each of size s of every u seconds can supply $\lambda_u mu$ users per u seconds, yielding the rate at which the helper uploads to the users to be $\lambda_u mu\frac{s}{u}$. Since by definition $m = \frac{d_h}{s}$, we arrive at $\lambda_u mu\frac{s}{u} = \lambda_u d_h$. In addition, the rate needed from each helper to make up for helpers' churning in the same cluster is $\frac{d_h}{\tau_h}$. Since the helper's upload rate is also upper bounded by its upload capacity b_h, we have $U_{ave} = \min((\lambda_u + \frac{1}{\tau_h})d_h, b_h)$. It follows that:

$$d_h^* = \arg\max_{d_h \geq 0}(\tau_h \min((\lambda_u + \frac{1}{\tau_h})d_h, b_h) - d_h) \qquad (2)$$

which yields (1). With this optimal download amount, the "effective upload bandwidth" of a helper, denoted by $\widetilde{b_h}$, is simply $\widetilde{b_h} = \frac{C_{ave}}{\tau_h} = \frac{\lambda_u \tau_h}{1+\lambda_u \tau_h} b_h$. This expression has an intuitive explanation: in a time interval of τ_h, each helper can supply $\lambda_u \tau_h$ users while having to sustain one helper's churning, which yields $\widetilde{b_h}$ a $\frac{\lambda_u \tau_h}{1+\lambda_u \tau_h}$ fraction of b_h.

3.2.3. Optimizing the Number of Helpers

Denote by N_h^* the number of helpers needed to maintain a self-sustainable system in steady state. By Little's law, N_h^* should satisfy $N_h^* \widetilde{b_h} = \lambda_u l(r - b_u)$, where the right hand side is the total demand and the left hand side is the total supply. It follows that $N_h^* = \frac{\lambda_u l(r-b_u)}{b_h} + \frac{l(r-b_u)}{b_h \tau_h}$. This serves as the theoretical lower bound for the number of helpers required to maintain a self-sustainable system in steady state. We proceed to show that our proposed scheme can achieve the lower bound. Since a packet of size s is the minimum unit of upload/download, two separate cases are considered as follows.

Case 1: $d_h^* \geq s$, i.e., the optimal download is larger than or equal to the packet size. In this case, helpers can utilize each downloaded packet to supply all the users in the corresponding segment before exhausting their upload bandwidth. With $n - k = \frac{(r-b_u)u}{s}$ helper clusters per swarm, choosing the number of helpers per cluster $\gamma = 1$ is sufficient to sustain the required streaming rate for all the users in the corresponding chunk. The total number of swarms is $\frac{l}{mu}$, and the total number of helpers denoted by N_h is given by $N_h = (n-k)\frac{l}{mu} = \frac{(r-b_u)u}{s}\frac{l}{mu} = \frac{\lambda_u l(r-b_u)}{b_h} + \frac{l(r-b_u)}{b_h \tau_h}$, which achieves the lower bound.

Case 2: $d_h^* < s$, i.e., the optimal download is less than the packet size. We choose $d_h = s$ because the least a helper should download is one packet. In this case, a helper cannot supply all the users within the corresponding segment before exhausting its upload bandwidth. To sustain the required streaming rate, we need $\gamma > 1$. The rate balance equation within each cluster is $\lambda_u u \frac{s}{u} + \gamma \frac{s}{\tau_h} = \gamma b_h$. The right hand side equals to helpers' supply per unit time, and the left hand side is the the users' required streaming rate from a cluster of helpers per unit time plus the rated needed to sustain helpers' churn. This supply-demand equation yields $\gamma = \frac{\lambda_u s}{b_h - \frac{s}{\tau_h}}$. The total number of helpers needed is $N_h = (n-k)\gamma\frac{l}{u} = \frac{\lambda_u l(r-b_u)}{b_h - \frac{s}{\tau_h}}$, and the ratio $\frac{N_h}{N_h^*} = \frac{1}{(1+\frac{1}{\lambda_u \tau_h})(1-\frac{s}{b_h \tau_h})}$ which, according to the condition $d_h^* < s$ and that of (2), is always greater than 1. However, when users' arrival rate is much larger than helpers' churn rate (i.e., $\lambda_u \tau_h \gg 1$) and a helper's average upload amount is much larger than one packet during its stay (i.e., $b_h \tau_h \gg s$), this ratio will be close to 1. In fact, these two conditions should be naturally satisfied if one expects helpers to be "helpful" to start with.

In practice, when $\lambda_h \tau_h > N_h$, the number of helpers can eventually exceed what is actually needed. In this case, appropriate scheduling schemes can be designed such that extra helpers can be utilized to help other video streaming sessions. When $\lambda_h \tau_h < N_h$, the system is in rate deficit mode, and additional server load is inevitable.

3.3. Practical Implementation

With the above design and choice of parameters, we proceed to give a detailed description of the system implementation.

3.3.1. User Network

The video content provider (the server) maintains a tracker to keep track of all the participating peers in the streaming session and to assist building the overlay network [2]. When a user joins the system, it obtains from the tracker a neighbor list of existing users who are already watching the same video. Users who started in closer time stamps compared to the newly arrived user's own start time are favored in choosing the neighbor list to improve upload efficiency [3]. Each user will keep a certain number of neighbors proportional to its upload bandwidth.

3.3.2. Helper Network

Within each cluster, the helpers form an unstructured overlay network following the same mechanism as that of the users. The tracker will keep track of the total available upload bandwidth of every cluster in all the swarms. When a new helper joins the network, the tracker assigns it to the cluster with the least amount of aggregate upload bandwidth to make each cluster have approximately the same amount of upload bandwidth. Each helper downloads one parity packet in every segment for a total of m segments.

3.3.3. Packet Exchange Protocol

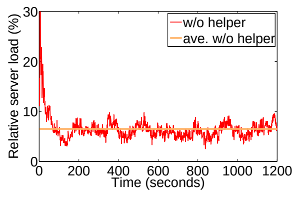
When a user joins the system, it also obtains from the tracker a list of helpers within swarm S_1. As the user continues to watch the video, its playback time will eventually fall beyond what swarm S_1 can supply (see Figure 1). When this happens, the user queries the tracker for a new list of helpers in the next swarm. The user continues to perform similar queries as it keeps streaming until to the end of the video. Each user maintains connections to at least one helper from each of the $n - k$ clusters in the corresponding swarm S_i. Each helper is allowed a maximum number of user neighbors proportional to its upload bandwidth.

The users will maintain a buffer which they will slide forward once every second. Only the packets within the buffer are downloaded. All the packets including the ones previously downloaded and those currently in the buffer are cached and made available to the neighbors. The users and helpers will update and announce to the neighbors their packet availability maps once every second. Each user will request packets and parity packets from its user and helper neighbors, in the order of importance of the packets. The importance of a packet is defined as follows. A segment is more important if it is closer to the user's playback time. Within each segment, the rarer the packet is in the user's one-hop neighborhood (as is determined by the availability maps), the more important it is. Each user will also mark all the packets in the most important segment, i.e., the one closest to its playback time, as urgent packets.

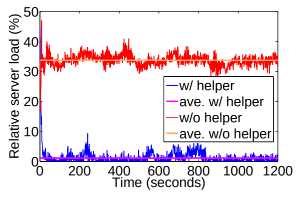
The users will perform a distributed optimization based rate allocation algorithm to determine which users' request to satisfy first and how much upload capacity should be allocated. The algorithm is similar in fashion to that developed by Wang et al. [3] but modified to adapt to the current system. Helpers will first satisfy other helpers' request to enable them to contribute their upload bandwidth. Among users' requests, helpers will prioritize urgent requests, and process the rest of the requests in ascending order of users' buffer levels. Finally, users will adaptively retrieve content from the server as needed to ensure smooth video playback.

4. SIMULATION RESULTS

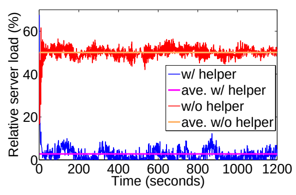
We evaluate the system performance using a discrete time packet level simulator. The following parameters are used: video length $l = 60$, segment length $u = 1$, packet size $s = 32$, user arrival rate $\lambda_u = 4$ and buffer length equal to 10 seconds. The average number of users in the system is 240. All the peers including users and helpers have unlimited download bandwidth and heterogeneous upload bandwidth including 25% with 128 kbps, 50% with 256 kbps and 25% with 384 kbps, averaged at $b_u = b_h = 256$ kbps. All simulations are 1200 seconds long. Although $l = 60$ is chosen, longer video lengths are also applicable when the simulation length is also adjusted to capture the steady state behaviors.



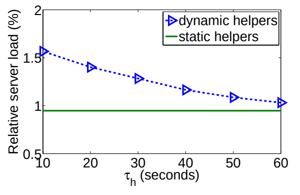
(a) $r = 256, \lambda_h = 0$



(b) $r = 384, \lambda_h = 2.01, \tau_h = 60$



(c) $r = 512, \lambda_h = 4.02, \tau_h = 60$



(d) $r = 384, \lambda_h\tau_h = N_h = 120.5$

Fig. 2: (a) (b) (c): Server load ($\%$) v.s. simulation time; (d) Dynamic helpers with different τ_h compared to static helpers.

We first evaluate helpers' role in improving the video quality by varying the video streaming rate r to be $256, 384$ and 512 kbps. We choose correspondingly $\lambda_h = 0, 2.01, 4.02$ and fix $\tau_h = 60$ to satisfy the minimum number of helpers needed. Figure 2 (a)(b)(c) show the relative server load (percentage of the total streaming rate needed by all the users) versus the simulation time. When $r = 256 = b_u$, the user network can be efficiently sustained on its own with little required server load, i.e., only 6.5% of the total rate on average. This also shows that the rate allocation scheme [3] is very efficient in utilizing users' upload bandwidth. When $r = 384 > b_u$, the number of helpers needed is $N_h = 120.5 = \lambda_h\tau_h$. It is shown that the relative server load has an average of 33.7% without helpers and only 1.1% with helpers. When $r = 512$, the system with $N_h = 241 = \lambda_h\tau_h$ helpers in steady state still only needs 2.7% average relative server load while that without helpers substantially burdens the server with 50.0% relative rate. To see the impact of helper dynamics, we use the case in Figure 2 (b) but vary $\tau_h = 10, 20, 30, 40, 50, 60$ and choose λ_h correspondingly such that $\lambda_h\tau_h = N_h$. The results are shown in Figure 2 (d). The green line refers to that of 120 static helpers and the blue markers correspond to dynamic helpers. These results have demonstrated that helpers' upload bandwidth can be efficiently utilized and minimal server load can be maintained with various helper dynamics.

5. CONCLUSIONS AND FUTURE WORK

In this paper, a helper-assisted P2P VoD system is proposed to support video streaming rate beyond the users' upload bandwidth. The optimal steady-state system parameters are derived and are shown to yield maximal utilization of helpers' resources. The proposed solution also uses the minimum number of helpers needed. Simulation results have demonstrated that helpers' upload bandwidth can be efficiently utilized and the system is robust to various helper dynamics. We aim to work on the followings in future: (1) consider heterogenous peer arrival processes; (2) derive realtime optimization-based strategies to maximize system throughput; (3) study systems with multi-video sessions; and (4) embed incentive mechanisms for helpers.

6. REFERENCES

[1] C. Huang, J. Li, and K.W. Ross, "Can Internet Video-on-Demand be Profitable?," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press New York, NY, USA, 2007, pp. 133–144.

[2] J. Wang and K. Ramchandran, "Enhancing Peer-to-Peer Live Multicast Quality Using Helpers," in *IEEE International Conference on Image Processing.*, 2008, pp. 2300–2303.

[3] J. Wang, C. Huang, and J. Li, "On ISP-friendly Rate Allocation for Peer-Assisted VoD," in *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, 2008, pp. 279–288.

[4] Y. Lu, J.D. Mol, F. Kuipers, and P. Van Mieghem, "Analytical Model for Mesh-based P2PVoD," in *IEEE International Symposium on Multimedia*, 2008.

[5] J. Pouwelse, P. Garbacki, J. Wang, and et al., "TRIBLER: A Social-Based Peer-to-Peer System," *Concurrency and Computation*, vol. 20, no. 2, pp. 127, 2008.

[6] J. Wong, "Enhancing Collaborative Content Delivery with Helpers," *Masters thesis, Univeristy of British Columbia*, 2004.

[7] J. Wang, C. Yeo, V. Prabhakaran, and K. Ramchandran, "On the Role of Helpers in Peer-to-Peer File Download Systems: Design, Analysis and Simulation," in *Proc. of IPTPS*, 2007.

[8] A.G. Dimakis, P.B. Godfrey, M.J. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," in *Proc. of IEEE INFOCOM*, 2007.