Peer-Assisted Distribution of User Generated Content

Zhengye Liu[*], Yuan Ding[†], Yong Liu[‡], Keith Ross[†]

[†]Computer Science & Engineering, Polytechnic Institute of New York University
[‡]Electrical & Computer Engineering, Polytechnic Institute of New York University
[*]AT&T Labs, San Ramon

Abstract—User Generated Content (UGC) video applications, such as YouTube, are enormously popular. UGC systems can potentially reduce their distribution costs by allowing peers to store and redistribute the videos that they have seen in the past. We study peer-assisted UGC from three perspectives. First, we undertake a measurement study of the peer-assisted distribution system of Tudou (a popular UGC network in China), revealing several fundamental characteristics that models need to take into account. Second, we develop analytical models for peer-assisted distribution of UGC. Our models capture essential aspects of peer-assisted UGC systems, including system size, peer bandwidth heterogeneity, limited peer storage, and video characteristics. We apply these models to numerically study YouTube-like UGC services. And third, we develop analytical models to understand the rate at which users would install P2P client applications to make peer-assisted UGC a success. Our results provide a comprehensive study of peer-assisted UGC distribution, exposing its fundamental characteristics and limitations.

I. INTRODUCTION

Over the past several years, we witnessed the emergence of User Generated Content (UGC) video systems such as Youtube, which is hugely popular throughout much of the world, and Tudou [1] and Youku [2], which are enormously popular in China. A recent study estimates nearly a half billion videos have been uploaded to YouTube, and that these videos have attracted more than 1.5 trillion views [12]. Tudou states it is one of the world's largest bandwidth users, moving more than 1 Petabyte per day to 7 million users. According to Tudou, YouTube serves a larger number of videos per day, but since the average Tudou video is longer in duration, the total amount of minutes of video being streamed daily from Tudou is significantly larger - about 15 billion minutes per month vs. 3 billion for YouTube [1].

Youtube currently distributes its videos from Google data centers and caches, located in more than 45 cities in 25 countries [7]. By some estimates, YouTube spends more than 1 million dollars a day on bandwidth [9]. The question that we explore in this paper is to what extent can UGC video sites, such as YouTube, exploit peer-assisted video distribution to reduce bandwidth and server costs without degrading the user experience. The answer to this question is not obvious given that much of user-generated content is long-tail content, and that users will in general only allocate a limited amount of their disk and upload bandwidth to P2P distribution.

With peer-assisted distribution of UGC, rather than providing VoD services purely with P2P, the peers assist the server/CDN in delivering video. Peer-assisted UGC deployments can use a P2P plug-in, which users optionally install. By installing the plug-in, a peer can redistribute to other peers the video content it has seen in the past. It is not mandatory for users to install the P2P plug-in, since users can always stream videos directly from the servers. However, installing and using the P2P plug-in potentially reduces the server cost for the service provider and improves the received video quality for the users. Examples of peer-assisted systems using a P2P plug-in include the Tudou accelerator [1] and fast Ku6 [3]. The Tudou accelerator and fast Ku6 are specifically designed and developed for Tudou.com and Ku6.com, the top two UGC video sharing websites for user generated videos in China. Tudou claims more than 50 million Internet users have installed the Tudou accelerator [1].

There is a lack of tractable performance models for peer-assisted UGC systems. Tractable performance models are needed to help understand overall system design. In particular, for a given peer-assisted UGC system, many design parameters may affect its overall system performance. The parameters include peer characteristics (e.g., number of peers, peer storage, peer upload bandwidth, and willingness of peers to contribute) and video characteristics (e.g., number of videos, video length, video rate, and video popularity). Incentives and video popularity shaping need also to be considered. Furthermore, it is important to investigate whether and how a P2P plug-in can become popular, and what are the major factors that affect its rate of adoption. It is critical to understand which design issues are of the greatest importance.

The contributions of this paper are as follows:

- We first undertake a measurement and analysis study of Tudou's peer-assisted distribution system. We determine the basic principles underlying Tudou's system. We also compare Tudou's performance with and without peer assistance, from the perspectives of both user experience and bandwidth usage.
- We then develop analytical models for peer-assisted UGC distribution. Our models capture essential aspects of peer-assisted UGC systems, including system size, peer bandwidth heterogeneity, limited peer storage, and video characteristics. Using linear programming and Monte Carlo averaging, we develop a methodology for evaluating the maximum possible peer contribution. We develop a bounding methodology, enabling us to circumvent solving the linear program at each Monte Carlo iteration. For an important special case, we show how the bound can be calculated analytically without using Monte Carlo averaging. We develop a scheduling heuristic which combines rarest video selection and water-filling bandwidth allocation, and show that this heuristic nearly achieves the upper bound.

- We apply these models to numerically study YouTube-like UGC services, and find that without tailored incentive and caching designs, peer-assisted UGC is less effective in terms of server bandwidth cost. Under a representative system setup, peer assistance reduces server bandwidth by at most 60%.
- We investigate the impact of the parameters and design choices. We observe that: (i) the number of shared videos plays a critical role in system performance; (ii) video scheduling between peers is less critical and does not need to be highly optimized; (iii) incentives to encourage idle peers to contribute are essential; (iv) system performance is very sensitive to video popularity. Reshaping video popularity (e.g., by using a video recommendation system) can significantly improve performance.
- We model how the popularity of a P2P plug-in evolves. We highlight two critical factors, namely, P2P plug-in performance and server bandwidth allocation policy, that highly affect the popularity evolution.

A. Related Work

Over the past few years, there has been a number of proposals for P2P VoD [21], [11], [14], [15], [20], [4], [9], [5]. Most of these proposals focus on P2P VoD system design and optimization. Recently, a measurement study on peer-assisted music-on-demand streaming system is presented in [13].

There are only a few theoretical performance studies of P2P streaming. Most of the studies focus on P2P live streaming [23], [19], [17], [8], [16]. Some recent work [18], [14], [20] study P2P VoD systems. Parvez et al. [18] developed models to study BitTorrent-like on-demand video systems. Their study focuses on P2P-dominated systems for which a peer can only serve other peers with the video it is currently watching. They analyzed the performance of different video chunk scheduling algorithms for a single video torrent. Our study addresses peer-assisted VoD with multiple videos. Suh. et al. [20] proposed and analyzed the Push-to-Peer system, mainly for set-top box networks. In Push-to-Peer, the server first pushes the videos to peers; after the push phase, the peers serve each other in a peer-dominated fashion. The authors proposed optimal active caching and scheduling schemes, and applied queueing models to study the system performance. We consider peer-assisted systems with passive caching, which has been adopted by most P2P VoD deployments. Huang et al. [14] analyze a peer-assisted system for which a peer can only serve other peers with the video it is currently watching. *We model peer-assisted system with the multiple-video design, that is, where a peer can serve other peers with its previously watched videos. To our knowledge, this is the first paper to model a peer-assisted VoD system with a multiple-video design.*

B. Organization

The rest of the paper is organized as follows. Section II measures and analyzes an existing peer-assisted UGC system, namely, Tudou, and provides a brief overview of the peer-assisted UGC design. Section III derives the performance models for the peer-assisted UGC systems. In Section VI, we apply the models to investigate a peer-assisted YouTube-like UGC system. Section V derives the models for the popularity evolution of P2P plug-ins. Section VII concludes the paper.

II. Tudou: A Case Study for Peer-Assisted UGC Video System

In this section, we study Tudou's peer-assisted UGC system. Users optionally download and install the Tudou P2P plug-in. After installing the P2P plug-in, when a user is watching a video through a web browser, the browser can download data from other peers who are also running the plug-in. Tudou's P2P plug-in must run with a Web-based video application, and alone it cannot stream, decode, and play video. Although the Tudou plug-in is closed source and proprietary, we have done a protocol analysis of the Tudou traffic to gain insight into its basic operation.

Using Wireshark and packet analysis, we have determined that Tudou has the following properties:
- **Multiple-video design:** A Tudou peer can serve other peers any Tudou video it has previously viewed and cached. A peer not only contributes its upload bandwidth, but also contributes its storage for caching the videos. The Tudou plug-in exploits a large amount of peer storage (1 GB as default), which can typically cache a large number of UGC videos. For a peer to upload a video, only the plugin needs to be running; it is not necessary that a browser process be active at the time of uploading.
- **Passive caching:** A peer passively caches the videos it has previously viewed. Videos are not pushed to the peers.
- **Caching in FIFO:** A Tudou peer replaces its previously cached videos in a first-in-first-out (FIFO) fashion. Specifically, when the peer reaches its storage limitation, it replaces the earliest cached video by the most recently watched video. It has been reported in [22] that a simple passive caching scheme can achieve a performance comparable with the optimal passive caching scheme.
- **Peer-assistance:** When a peer requests a particular video from the Tudou web site, a Tudou server begins to upload the video to the peer over HTTP. At the same time, the peer contacts the Tudou tracker, which returns a list of peers that also have a copy of the video. The peer can then request video chunks from other peers on this list. Our experiments have shown that Tudou only caches and tracks Tudou videos; it does not cache and track videos from other sites (such as YouTube).
- **Partial video caching:** Videos are broken up into chunks of size 2 MBytes. If a peer only watches a portion of a video, it caches the corresponding chunks. After a downloading peer learns from the tracker which other peers have the video, the downloading peer immediately contacts those peers (over TCP connections), who respond by indicating which chunks they have. The downloading peer then requests specific chunks from the various peers.

We now turn our attention to the performance of peer-assisted Tudou. Table 1 shows the results of an experiment of a user in USA downloading five different videos from Tudou. For each video, the user first downloaded the file without the plugin and then with the plugin. For each video, Table 1 provides the download time without the plugin, the download time with the plugin, the number of peers that participated with the plugin, and the percentage of data obtained from the peers with the plugin. We see that the plugin significantly shortened the download time, the speed-up factor is as large as 10.89. This was accomplished with a relatively small

TABLE I

TUDOU PEER-ASSISTANCE PERFORMANCE: USA USER

video	download time no plugin	download time with plugin	speed-up factor	number of peers	percentage of data from peers	number of views
1	94 sec	25 sec	3.76	3	36%	5332
2	943 sec	159 sec	5.93	5	77%	39280
3	285 sec	52 sec	5.48	9	92%	208294
4	501 sec	46 sec	10.89	4	59%	13724
5	2292 sec	262 sec	8.75	5	83%	115149

TABLE II

TUDOU PEER-ASSISTANCE PERFORMANCE: CHINA USER

video	download time no plugin	download time with plugin	speed-up factor	number of peers	percentage of data from peers	number of views
1	27 sec	24 sec	1.13	6	82%	5080
2	366 sec	45 sec	8.13	17	89%	10551
3	533 sec	227 sec	2.34	45	94%	77534
4	693 sec	179 sec	3.87	2	91%	15531
5	1288 sec	215 sec	5.99	20	96%	343490

number of assist peers (between 3 and 9) for each of the videos. Most of the peers providing the peer assistance were also located in the USA. Moreover, the plugin significantly reduced the bandwidth usage at the server, with the peers contributing between 36% and 92% of the bandwidth. We also observed that there is a strong correlation between server bandwidth saving and video popularity: videos with more views tend to benefit more from P2P sharing. This suggests that video popularity has significant impact on the efficiency of P2P sharing. We will formally investigate this issue through modeling and numerical study in the following two sections.

Table 2 shows the results of the same experiment, but with the downloading peer now located in China. (Also the set of five videos is not the same.) For this experiment, we observed more variations in the number of assisting peers, ranging from 6 to . Also, the peers contribute even a larger fraction of the bandwidth, ranging from 76% to 96%. Since the bandwidth savings are generally high, there is no obvious correlation between video popularity and P2P bandwidth saving. We also observed that most of the peers providing peer assistance were located in China. Thus, it appears that Tudou attempts to localize the P2P traffic.

In conclusion, we see from our experiments on Tudou that peer-assistance has the potential to not only improve user-perceived performance but also reduce bandwidth expenditures for UGC video systems. In the next section, using a novel analytic model, we investigate the extent to which bandwidth can be reduced in a UGC with heavy-tail content.

III. PERFORMANCE MODELING OF PEER-ASSISTED UGC

In a peer-assisted UGC system, each peer may act as a receiver, as a supplier, or as both. As a receiver, it requests video data from the other peers and/or the server; as a supplier, it uses its stored videos and upload bandwidth to serve other peers. In peer-assisted UGC, if a receiver cannot find enough suppliers for the video it is watching (due to the lack of upload bandwidth or missing video content from the peers), it can be served by the server.

For the clarity of presentation, we use full video as the basic unit for P2P video caching and sharing. If caching and sharing operate at the video chunk level, one can simply treat each video chunk as a short video. An online peer, whether it is idle or watching a video, can distribute a video that it has in its cache. The server has sufficient capacity and stores all videos, so that when a peer cannot receive the full video rate from other peers, it can always receive the remaining rate from the video server.

In general, the performance of peer-assisted UGC system is determined by four factors: (i) video content demand, (ii) peer upload bandwidth, (iii) video placement, and (iv) peer upload scheduling. Figure 1 gives an example on how these factors affect the system performance. In this example, there are three suppliers and two receivers. The upload bandwidth of suppliers A, B, and C are 300 kbps, 500 kbps, and 200 kbps, respectively. Assume the video rate is 400 kbps. We also indicate the stored videos in each supplier. Receivers D and E request videos 3 and 2, respectively. Based on current scheduling, E receives 200 kbps from B and 200 kbps from C for video 2. Since only B stores video 3, D needs to receive 100 kbps from the server in addition to receiving 300 kbps from B. A better scheduling policy would let E receive 200 kbps from A. In this case, D can receive the full video rate from B, and does not need to consume any server bandwidth.

To evaluate the performance of a peer-assisted UGC system, we will use the *server cost ratio*, i.e., the ratio of the average server rate (denoted by $\mathcal{S}$) over the average content consumption rate (denoted by $\mathcal{R}$). $\mathcal{R}$ is equivalent to the average server rate without peer-assistance.

Discrete-event simulation can be used to analyze the server cost ratio in peer-assisted UGC. But simulation is slow and does not allow us to easily investigate how performance varies on the numerous design parameters involved. Instead, we develop a Monte Carlo sampling and averaging procedure to
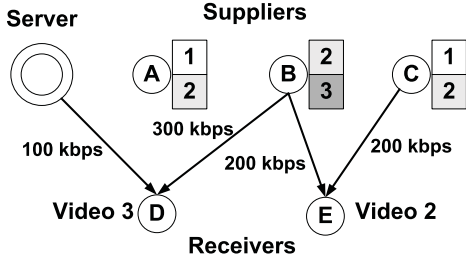
Fig. 1. Illustration on scheduling in peer-assisted UGC.

study performance. At the heart of our procedure is a bounding methodology, which we describe next.

A. Optimal Scheduling and Bounds

Evaluating the maximal peer contribution across all scheduling policies is a fundamental but difficult problem. Our approach in this paper is to find a tight upper bound on the peer contribution, and then construct a scheduling heuristic that provides performance close to the upper bound. To this end, we first formulate the optimal scheduling problem for peer-assisted UGC.

We first introduce some notation. Let M denote the total number of videos in the system and let r_m denote the rate of video m. Let N denote the total number of users that subscribe to the peer-assisted UGC service, e.g., by installing the P2P software. Because the number of subscribers is relatively stable compared with the peer dynamics, we assume N is fixed. Our model permits heterogeneous peer upload bandwidth. Let u_i be the upload bandwidth of the ith peer, $i = 1, \ldots, N$. For simplicity, we assume all videos are of the same size and use the number of videos to measure peer storage size. We assume each peer can store v videos.

A given peer i can be in one of $M + 2$ states: offline, online but not watching a video, or watching video m, $m = 1, \ldots, M$. Let n_m be the number of peers watching video m, n_h be the number of online peers that are not watching any video, n_o be the number of offline peers. Then $n = n_h + n_1 + \cdots + n_M$ is the total number of online peers. Let $\mathbb{N}$ be the set of online peers. The demand for video m is given by $r_m n_m$. Denote $\mathbf{n} = (n_o, n_h, n_1, \ldots, n_M)$. We refer to $\mathbb{N}$ and $\mathbf{n}$ as the *availability state* and *demand state*, respectively.

Having specified the availability and demand states, let us now consider the current video supply. Let $\mathbb{M}_i$ be the set of videos that peer i currently stores. Also let $\mathbb{M} = (\mathbb{M}_1, \ldots, \mathbb{M}_N)$ denote the storage sets across all peers. We refer to $\mathbb{M}$ as the *storage state*. We refer to the *state of the system* as $(\mathbb{N}, \mathbf{n}, \mathbb{M})$.

Given the current storage state $\mathbb{M}$, current availability state $\mathbb{N}$, and the current demand state $\mathbf{n}$, we now consider the scheduling problem. Each online peer allocates its upload bandwidth among the videos that it currently stores. Let x_{im}, $m \in \mathbb{M}_i$, be the amount of upload bandwidth peer i allocates to video m. We refer to the allocations $\{x_{im}, i \in \mathbb{N}, m \in \mathbb{M}_i\}$ taken together as the **scheduling policy**, which is denoted by $\mathscr{F}$. A scheduling policy is said to be *feasible* if the supply does not exceed the video content demand, that is,

$$\sum_{i \in \mathbb{N}} x_{im} \leq r_m n_m, \quad \forall m = 1, \ldots, M$$

and the supply does not exceed the upload bandwidth limits, that is,

$$\sum_{m \in \mathbb{M}_i} x_{im} \leq u_i, \quad \forall i \in \mathbb{N}.$$

Given a feasible scheduling policy $\mathscr{F}$, the bandwidth contributed by the peers is

$$c(\mathscr{F}|\mathbb{N}, \mathbf{n}, \mathbb{M}) = \sum_{i \in \mathbb{N}} \sum_{m \in \mathbb{M}_i} x_{im} \tag{1}$$

An optimal scheduling policy for the current state of the system $(\mathbb{N}, \mathbf{n}, \mathbb{M})$ is given by the solution of the following linear program:

$$
\begin{aligned}
\max \quad & \sum_{i \in \mathbb{N}} \sum_{m \in \mathbb{M}_i} x_{im} \\
\text{st.} \quad & \sum_{i \in \mathbb{N}} x_{im} \leq r_m n_m, \quad m = 1, \ldots, M \\
& \sum_{m \in \mathbb{M}_i} x_{im} \leq u_i, \quad i \in \mathbb{N}
\end{aligned}
\tag{2}
$$

It is not hard to see that this linear program is a max-flow problem, and can be solved in $O(n^3)$ time for a given state $(\mathbb{N}, \mathbf{n}, \mathbb{M})$. Let $\mathscr{F}^*$ denote an optimal scheduling policy.

B. Probabilistic Models

Let β_m denote the popularity of video m, that is, β_m is the probability that a user requests to view video m. Each peer is either offline, idle, or watching one of the M videos. Thus, a peer can be in one of the $M + 2$ states. After a peer watches a video m, it may go on to watch another video, become idle, or go offline. The probabilities of switching to offline, idle, and watching video $1, \ldots, M$ are denoted as $\lambda_o, \lambda_h, \lambda_1, \ldots, \lambda_M$, respectively. Clearly, λ_m is related to video m's popularity β_m, and can be expressed as $\lambda_m = \beta_m(1 - \lambda_o - \lambda_h)$.

Denote the average offline time, average idle time, and average viewing times for the videos $m = 1, \ldots, M$ as $1/\mu_o$, $1/\mu_h$, $1/\mu_1, \ldots, 1/\mu_M$, respectively. Let $\rho_o = \lambda_o/\mu_o$, $\rho_h = \lambda_h/\mu_h$, and $\rho_m = \lambda_m/\mu_m$ for $m = 1, \ldots, M$. Normalize all the ρ_o, ρ_h, and ρ_m's, so that $\rho_o + \rho_h + \sum_{m=1}^{M} \rho_m = 1$. It is easy to see that the probabilities that a peer is offline, online but not viewing, and viewing video m are given by ρ_o, ρ_h, and ρ_m, respectively.

Having provided probabilistic models for peer availability and demand, we now model peer storage. First we make the natural assumption that users choose videos to view independently of each other. We also assume that each of the v videos in a user's cache is selected independently, with video m selected with probability β_m. We notice that our model incorrectly allows a peer to cache duplicated copies of a single video, which is not the case in real world systems. But when the video catalog size is much larger than the per-peer storage size, $M \gg c$, the probability of caching duplicated copies at a single peer should be very low.

C. Monte Carlo Averaging Procedure

Given this natural probabilistic model, we can now evaluate the performance of a scheduling policy using Monte Carlo methods. Specifically, for each peer, we randomly generate its state, and for each online peer we randomly generate the v videos that it caches. We then use (1) to determine the peer contribution for the policy and the given state. We repeat this procedure many times, generating a new random state at each iteration, and finally take the average. In this way, we can obtain the average peer contribution (denoted by $C(\mathscr{F})$)

for a given $\mathscr{F}$. We can also calculate confidence intervals, allowing us to determine the number of iterations for any desired accuracy. Similarly, we can determine the maximal contribution from the peers by solving the max-flow problem (2) at each Monte Carlo iteration.

D. Bounding Performance

Although the solution of the optimal scheduling problem (2) can in principle be used to construct a tight upper bound on the performance of peer-assisted UGC, due to the complexity of the max-flow problem, we instead seek a simpler performance bound. We will show in Section VI via simulations that this upper bound is nearly tight.

We claim that under any feasible scheduling policy, the peer contribution $c(\mathscr{F}|\mathbb{N}, \mathbf{n}, \mathbb{M})$ is bounded by:

$$c(\mathscr{F}|\mathbb{N}, \mathbf{n}, \mathbb{M}) \leq \min\left(\sum_{i\in\mathbb{N}} u_i, \sum_{m=1}^{M} \min(n_m r_m, \sum_{i\in\mathbb{N}(m)} u_i)\right),$$
(3)

where $\mathbb{N}(m)$ is the set of online peers that store video m. This bound is obtained as follows. The peer contribution rate is limited by two constraints: (i) the peers cannot upload more than their aggregate upload rate, namely, $\sum_{i\in\mathbb{N}} u_i$; and (ii) for each video m, the peers cannot upload more than the video demand for video m, namely, $r_m n_m$, and also cannot upload more than $\sum_{i\in\mathbb{N}(m)} u_i$. When no peer caches video m, i.e., $\mathbb{N}(m) = \emptyset$, the peer contribution for video m is zero, and all demands have to be served by the server. With limited peer storage, the caching probability for long-tail videos is small. Even though the requesting probability for individual long-tail video is small, the aggregate bandwidth requirement is still significant. As will be shown in our numerical study, with limited peer caching size, peer contribution levels off quickly as their upload bandwidth increases.

By using (3) in the Monte Carlo procedure, we no longer have to solve a max-flow problem at each iteration. Instead at each iteration, we simply evaluate the bound in (3).

Now we explore developing yet a more efficient procedure for bounding the peer contribution. Let l_m be the number of available copies of video m in the system and define $\mathbf{l} = (l_1, \ldots, l_M)$. The idea here is to generate at each iteration only n, $\mathbf{n}$, and $\mathbf{l}$ and not $\mathbb{N}$ and $\mathbb{M}$. To this end, we replace (3) with an approximate bound given by

$$c(\mathscr{F}|n, \mathbf{n}, \mathbf{l}) \leq \min\left(nw, \sum_{m=1}^{M} \min(n_m r_m, l_m u_{\max})\right),$$
(4)

where $u_{\max}$ is the maximum upload rate among all peers and w is the average upload rate across all peers.

We now derive the distribution of $\mathbf{n}$ and $\mathbf{l}$, which will be used to generate the Monte Carlo samples of $\mathbf{n}$ and $\mathbf{l}$. With the probabilistic models shown in Section III-B, because the states of the peers are independent, it is easy to see that $\mathbf{n}$ follows a multinomial distribution:

$$P(\mathbf{n}) = N! \frac{\rho_o^{n_o}}{n_o!} \frac{\rho_h^{n_h}}{n_h!} \prod_{m=1}^{M} \frac{\rho_m^{n_m}}{n_m!}.$$
(5)

Also, since each peer caches a video from M videos at each storage slot independently, for a given n, $\mathbf{l}$ also follows a multinomial distribution:

$$P(\mathbf{l} \mid n) = (nc)! \prod_{m=1}^{M} \frac{\beta_m^{l_m}}{l_m!}.$$
(6)

E. Closed-form expression

In the previous section, we have developed an efficient Monte Carlo methodology for bounding the maximal peer contribution in the system. We now develop an analytical procedure for an important special case.

We first note that the video rate for YouTube is approximately 300 kbps, which is lower than the average upload rate of the peers in the Internet [14]. Furthermore, after the idle peers behave as helpers and serve other peers, it is possible that the aggregate supply exceeds the aggregate demand even when high rate videos are streamed. Making this observation, we now consider the special case of the aggregate bandwidth supply always exceeding the aggregate bandwidth demand, specifically,

$$\sum_{m=1}^{M} r_m n_m \leq \sum_{i\in\mathbb{N}} u_i.$$
(7)

A special case when this occurs is that the maximum video rate among all M videos is smaller than the minimum upload bandwidth among all N peers.

With the assumption (7), the upper bound (3) becomes:

$$c(\mathscr{F}|n, \mathbf{n}, \mathbf{l}) \leq \sum_{m=1}^{M} \min(r_m n_m, l_m u_{\max}).$$
(8)

Therefore, the average peer contribution $C(\mathscr{F})$ satisfies:

$$
\begin{aligned}
&C(\mathscr{F}) \\
\leq\ & \sum_{m=1}^{M} \sum_{\mathbf{n}} \sum_{\mathbf{l}} [\min(n_m r_m, l_m u_{\max}) P(\mathbf{l}|\mathbf{n}) P(\mathbf{n})] \\
=\ & \sum_{m=1}^{M} \sum_{n=0}^{N} \sum_{k_m=0}^{n} \sum_{l_m=0}^{nc} [\min(k_m r_m, l_m u_{\max}) \\
& P(l_m|n) P(k_m|n) P(n)],
\end{aligned}
$$
(9)

where $P(l_m|n)$ is the probability that given n active peers, l_m peers cache video m; $P(k_m|n)$ is the probability that given n active peers, k_m peers watch video m; and $P(n)$ is the probability that n peers are active. They all follow binomial distributions:

$$
\begin{aligned}
P(l_m|n) &= \tfrac{(nc)!}{l_m!(nc-l_m)!} \beta_m^{l_m}(1-\beta_m)^{nc-l_m}, \\
P(k_m|n) &= \tfrac{n!}{k_m!(n-k_m)!} \rho_m^{l_m}(1-\rho_m)^{n-k_m}, \\
P(n) &= \tfrac{N!}{n!(N-n)!}(1-\rho_o)^n \rho_o^{N-n}.
\end{aligned}
$$

IV. NUMERICAL RESULTS OF PERFORMANCE MODEL

Using our models of the previous sections, we now numerically investigate the behavior of a Youtube-like peer-assisted UGC service. We focus on how tight is the derived peer contribution upper bound and how the different parameters impact system performance.

Given a set of model parameters, we first calculate the content rate $\mathcal{R}$. We then calculate the lower bound of the server rate based on the upper bound of the peer rate in Equation (4). Additionally, we calculate the server rate $\mathcal{S}$ for a specific scheduling algorithm $\mathscr{F}$ with Monte Carlo averaging, which provides an achievable upper bound on server rate.
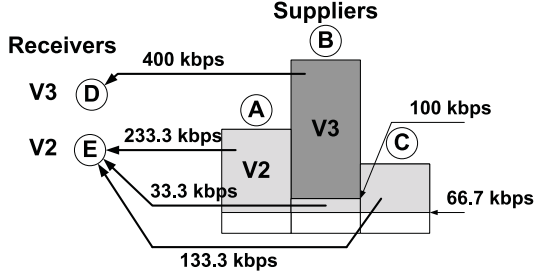
Fig. 2. Illustration of rarest-first with water filling. The height of a rectangle indicates the amount of bandwidth. Dark gray, gray, and white indicate the bandwidth allocated to receiver D for video 3, the bandwidth allocated to receiver E for video 2, and the leftover bandwidth, respectively.

1) Video Scheduling Heuristic: First, we need to specify a scheduling policy. As discussed in Section 3, for a given system state $(\mathbb{N}, \mathbf{n}, \mathbb{M})$, the optimal scheduling algorithm can be formulated as a classical max-flow problem. Instead of computing the max-flow for each instance of the system, we consider a heuristic policy, which we call *rarest-first video scheduling with water-filling*, to approximate the optimal solution. This policy is in the same spirit as the rarest-first scheduling algorithms that have been widely used in P2P file sharing and P2P live video streaming. The basic idea is that the system will first serve the demand for the video with the rarest supply (i.e., the video with the smallest number of cache copies), and then the video with the next rarest supply, and so on, until either all video demands have been satisfied or all user bandwidth has been used up. For a given video, if multiple peers have a copy, then bandwidth is allocated among the peers so as to equalize the residual bandwidth in the peers holding the copies.

We re-use the example shown in Figure 1 to illustrate the rarest-first algorithm. Since between video 2 and video 3, video 3 has fewer copies available, the system will serve video 3 first. As shown in Figure 2, B serves D with video 3 at a full video rate 400 kbps. The available bandwidth at B reduces to 100 kbps. Next, video 2 can be served by A, B, and C. To maintain the workload balance, the system then serves video 2 by allocating bandwidth on A, B and C in a water-filling fashion – after serving video 2, A, B and C have the same leftover upload bandwidth.

2) Model parameters: We now present the default model parameters. In our default model, there are totally $N = 100,000$ users subscribed in a UGC service with all of them installing a P2P plug-in. On average there are 10,000 active online peers, so that $\sum_m \rho_m = 0.1$. There are two types of peers: $p_1 = 40\%$ of high bandwidth peers with upload bandwidth $u_1 = 1$ Mbps, and $p_2 = 60\%$ of low bandwidth peers with upload bandwidth $u_2 = 400$ kbps. By default, we assume all idle peers are selfish and are not willing to serve other peers.

In [6], it was shown that the number of videos in YouTube is much larger than the number of active online users. It is reasonable to assume that the number of videos is two orders of magnitude larger than the number of active peers. Thus, we use totally $M = 1,000,000$ videos. We set the video rate to 400 kbps and the video length is 6 minutes. Each peer caches $v = 50$ videos in this case, so that the storage used at

each peer is about 1 GB. We assume that the video popularity β_m follows a Zipf distribution with the parameter $\alpha = 0.54$ [9]. In the following sections, we will investigate the impact of system parameters by varying a particular parameter and fixing other default parameters.

A. Results and Analysis

We now investigate the impact of five parameters on the performance of a peer-assisted systems.

Peer Upload Bandwidth

Peer upload bandwidth is one of the most important parameters in P2P applications. We vary the upload bandwidth of each peer by scaling the default upload bandwidth. The scaling factor ranges from 0.2 to 2.0. Figure 3 relates the peer upload bandwidth to the server cost ratio. We use "bound" to indicate the lower bound of server cost ratio, and "heuristic" to indicate the server cost ratio obtained by using the rarest-first water-filling policy. We plot the server cost ratios as a function of the average upload bandwidth in the figure. First, we observe that the lower bound of server cost ratio is quite close to that obtained by using the rarest-first water-leveling policy. This indicates that the lower bound is relatively tight and the rarest-first algorithm can achieve near optimal performance. It also indicates that video scheduling is less critical in peer-assisted UGC, and elaborate scheduling designs are not necessary. Second, increasing peer upload bandwidth beyond a certain point does not significantly improve the system performance. The server cost ratio seems converge to a relatively high value (always higher than 40%) as peer upload bandwidth increases. Intuitively, this can be explained as the "storage bottleneck" problem. In the example of YouTube, with tens of millions of user-generated videos, the chance of peers to watch the same video is much lower than in other VoD systems, e.g., Hulu, that share a much smaller number of professionally generated videos. Given the limited peer storage size, the possibility for a peer to download a long-tail video from other peers is small.

To gain quantitative understanding, let's re-examine equation (8). When $u_{\max}$ is really large, we can assume that $l_m u_{\max} > r_m n_m, \forall l_m \geq 1$. In other words, as long as one peer caches video m, it can serve all peers watching video m. Then the upper bound becomes

$$c(\mathscr{F}|n, \mathbf{n}, \mathbf{l}) \leq \sum_{m=1}^{M} r_m n_m \mathbf{1}(l_m), \qquad (10)$$

where $\mathbf{1}(\cdot)$ is the indicator function that takes value 1 if $l_m \geq 1$, and value 0 if $l_m = 0$. Given n active peers, the average peer contribution $C(\mathscr{F}|n)$ satisfies:

$$
\begin{aligned}
&C(\mathscr{F}|n) \\
\leq\ & \sum_{m=1}^{M} \sum_{\mathbf{n}} n_m r_m P(l_m \geq 1|\mathbf{n})P(\mathbf{n}|n)] \\
=\ & \sum_{m=1}^{M} r_m E[n_m|n]P(l_m \geq 1|n) \\
=\ & n \sum_{m=1}^{M} r_m \rho_m \left(1 - (1 - \beta_m)^{nc}\right).
\end{aligned}
\qquad (11)
$$

From the server point of view, it must serve peers watching videos that are not cached by any peer. Given there are n active peers, the server rate satisfies:

$$\mathcal{S}_n \geq \sum_{m=1}^{M} r_m E[n_m|n]P(l_m = 0|n) = n \sum_{m=1}^{M} r_m \rho_m (1 - \beta_m)^{nc}. \qquad (12)$$
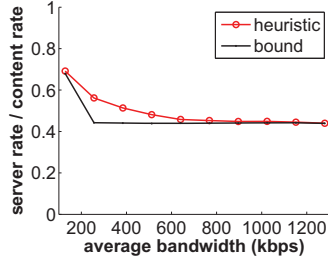
266

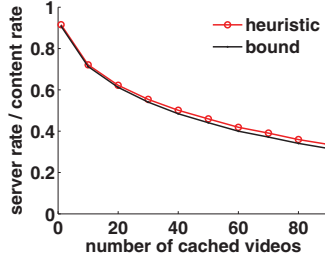Fig. 3. Peer upload bandwidth.
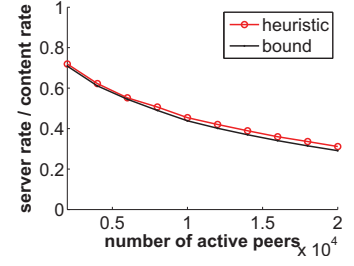

Fig. 4. Peer storage.

Fig. 5. System size on server cost ratio.

It is suggested in (12) that, when peer upload bandwidth is abundant, the server cost ratio will converge to a constant determined by peer storage c, system size n, and video popularity distribution $\{\rho_m, 1 \leq m \leq M\}$ and $\{\beta_m, 1 \leq m \leq M\}$. In the following, we will numerically study the impact of each of those parameters.

Peer Storage

Figure 4 relates the server cost ratio to peer storage. It shows that the server cost ratio decreases as peer storage size increases. This can also be explained from the server cost for missed video in (12), where S_n is a decreasing function of c. The decreasing speed is determined by both the system size n and video popularity distributions. In Figure 4, initially, the server cost ratio drops dramatically as peer storage increases. The decrease becomes slower as peer storage increases further. This indicates that the system needs to use a large amount of peer storage to reach a low server cost ratio. Instead of a "bandwidth bottleneck," we see that YouTube-like UGC systems encounter a severe "storage bottleneck" problem, due to the huge number of videos and limited peer storage. Although storage is cheap these days, it is still challenging to provide appropriate incentives to encourage peers to contribute their storage to cache watched UGC videos.

System Size

We investigate how the system size, i.e., the number of active peers, affects system performance in peer-assisted UGC. Figure 5 relates the server cost ratio to the number of active peers. The server cost ratio decreases dramatically from a small system to a large system. This is because the system size directly determines the aggregate system storage and the content availability. Based on (12), the server cost for missed video normalized by the system size can be calculated as

$$\frac{S_n}{n} = \sum_{m=1}^{M} r_m \rho_m (1 - \beta_m)^{nc},$$

which is a decreasing function of n. Since n and c show up in the equation as a product, they have similar impact on the server rate and content rate ratio. This explains why Figure 4 and 5 have very similar shapes.

As the number of receivers increases, the content rate also increases. Even though the server cost ratio decreases, it does not necessarily mean that the server is running with a lower rate. We investigate the server rate under different peer populations when the system resource index, defined as the ratio between average peer upload bandwidth and average peer video download rate, is larger than 1. We plot in Figure 6

the server rate when the system size varies from 2,000 active peers to 70,000 active peers. We observe that after reaching a maximum server rate, as the number of active peers increases, the server rate decreases. When more peers become active, P2P sharing becomes more efficient, not only covering the additional content rate introduced by the new active peers, but also reducing the server bandwidth for the original active peers. Importantly, it indicates that *peer-assisted UGC system is scalable* – after the system reaches a certain size, P2P dominates the uploading so that the server rate is bounded.

Video Popularity

We now investigate the impact of video popularity on server bandwidth cost. Let's assume video popularity follows a Zipf distribution. After videos are ranked based on popularity, the popularity for video m is $\beta_m = \frac{k}{m^{1+\alpha}}$, where k is the normalization constant, and α is the Zipf parameter, the smaller the α, the more long-tail the distribution. We vary the Zipf parameter α from 0 to 1.2. In Figure 7, we can observe that the video popularity has a big impact on the server cost ratio. The system is more sensitive to video popularity than other system parameters that have been investigated.

In (12), since ρ_m is proportional to β_m, the monotonicity of the server cost ratio with α is not immediately clear. Let's elaborate on this by plugging in the Zipf popularity function. The probability that video m is not cached by any peer can be calculated as:

$$P(l_m = 0) = \left(1 - \frac{k}{m^{1+\alpha}}\right)^{nc}.$$

$P(l_m = 0)$ is an increasing function of m, for any $0 < \eta < 1$, we can find M_η such that $P(l_m = 0) \geq \eta$ if $m \geq M_\eta$:

$$M_\eta = \left(\frac{k}{1 - \eta^{\frac{1}{nc}}}\right)^{\frac{1}{1+\alpha}}. \tag{13}$$

Meanwhile, $P(l_m = 0)$ is also an increasing function of α, and a decreasing function of n and c. The expected download workload for video m is

$$r_m n \rho_m = r_m n \beta_m (1 - \lambda_0 - \lambda_h)/\mu_m = \frac{n w_m}{m^{1+\alpha}},$$

where $w_m = r_m k (1 - \lambda_0 - \lambda_h)/\mu_m$. Following (12), the expected server workload for serving the missed videos is

$$S_n = \sum_{m=1}^{M} P(l_m = 0) \frac{n w_m}{m^{1+\alpha}}. \tag{14}$$

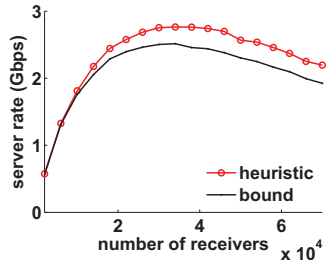According to (13), we can choose some $0 < \eta < 1$ so that

Fig. 6.   System size on server rate.

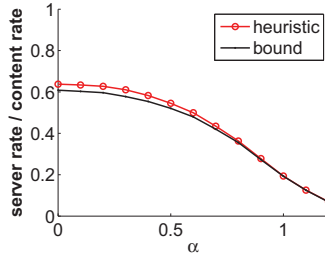
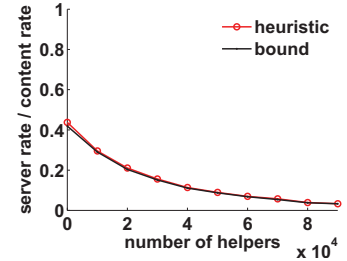Fig. 7.   Video popularity.

Fig. 8.   Helpers.

$M_\eta < M$, and $P(l_m = 0) \geq \eta$ if $m > M_\eta$. Then we have

$$\mathcal{S}_n \geq \sum_{m=\lceil M_\eta \rceil}^{M} P(l_m = 0) \frac{nw_m}{m^{1+\alpha}}$$

$$\geq \eta nw_0 \sum_{m=\lceil M_\eta \rceil}^{M} \frac{1}{m^{1+\alpha}},$$

where $w_0 \triangleq \min_{1 \leq i \leq M} w_m$. When M is large, we will have

$$\mathcal{S}_n \geq \frac{\eta nw_0}{M_\eta^\alpha} \propto n \left(1 - \eta^{\frac{1}{nc}}\right)^{\frac{\alpha}{1+\alpha}}. \tag{15}$$

Given n and c, $\mathcal{S}_n$ is a decreasing function of α. Notice that, (15) only takes into account the server bandwidth used to serve videos with popularity rank lower than M_η. Even though the probability for a peer to watch each of those less popular videos is small, also due to their low popularity, the probability that they miss in all peers' caches is high ($\geq \eta$ by definition). Due to the long-tailed popularity distribution function, there are a large number of them. In aggregate, they consume a significant portion of server bandwidth.

To reduce server cost ratio, a service provider can "shape" the video popularity. For example, YouTube has functions to suggest related videos to the users. In this way, YouTube can guide the users to watch more popular videos, thereby reduce the number of long-tail videos watched by users.

Number of Helpers

There are normally a large number of idle peers in a UGC system. With appropriate incentives, an idle peer can be motivated to contribute their resources. These helpers can potentially form a huge resource pool that can be used to reduce the server rate. We now investigate how idle peers can help. We assume that the helpers have the same characteristics as the active peers. We vary the number of helpers from 0 to 90,000. Figure 8 relates the server cost ratio to the number of helpers. We observe that the storage bottleneck problem is not easily resolved even with a large number of helpers. The server cost ratio reduces slowly as the number of helpers increases. However, we also note that when the number of helpers increases, the load on each active peer reduces. Therefore, with the help of idle peers, both server load and peer load can drop to a very low level. With proper incentives (and helpers), it is possible to distribute on-demand videos to a large number of users with very low server cost.

V. POPULARITY EVOLUTION MODELS

We now present the results on the popularity evolution of the P2P plug-in and its impact on the system performance. In Sections III, we assumed that all users install the P2P plug-in. We also assumed that the server is capable of serving all video sessions without any peer assistance. In this section, we discuss a different scenario without such assumptions.

We now suppose the server can at most serve a fraction γ of the total video demand. We further suppose the server can distribute its bandwidth to the video sessions so that a user viewing video m receives the fraction γ of the video rate r_m. A partial received video rate $\gamma < 1$ indicates a lower received video quality, e.g, a less clear image or a frequently frozen video session. This motivates users to install a P2P plug-in to improve their received video quality.

As discussed in Section III, the effectiveness of a P2P plug-in is highly dependent on the number of peers that use it. If there are only a small number of users using the P2P plug-in, it is ineffective and there is little motivation for users to use it. Critical questions include (i) whether a P2P plug-in can become popular, (ii) how a P2P plug-in becomes popular, (iii) how to increase the popularity of a P2P plug-in. To answer these questions, we model popularity evolution in this section.

Assume time advances in time slots. Assume at time t, there is a fraction $\Theta(t)$ of users use the P2P plug-in. We investigate the popularity evolution of the P2P plug-in, i.e., how $\Theta(t)$ evolves in time. To simplify our discussion, we assume that all videos have the same video rate r. We denote the video rates received by users without the P2P plug-in and with the P2P plug-in as $\omega_s(t)r$ and $\omega_p(t)r$, respectively. Since P2P always increases the received video rate, $0 \leq \omega_s(t) \leq \omega_p(t) \leq 1$ for any time slot t.

We model $\Theta(t)$ as follows:

$$\Theta(t) = \Theta(t-1) + (1 - \Theta(t-1))Q(\omega_p(t-1) - \omega_s(t-1)), \tag{16}$$

where $Q(x)$ is a non-decreasing function of x and less than or equal to 1. The rationale behind this is: the bigger the difference between using the P2P plug-in and not using the plug-in, the more likely it is for a user to adopt it. For simplicity, we use $Q(x) = \min(1, ax)$ in our discussion.

The above model shares some similarities with the popularity evolution model for Web page proposed in [10]. However, there are two key differences: (i) in the Web model, the quality of a Web page is constant, independent of its popularity; while in the P2P plug-in model, the quality (i.e., effectiveness of the plug-in) is highly dependent on its popularity; (ii) in the Web model, the main contributor to popularity evolution is the

users' awareness of the Website. Differently, for the P2P plug-in, the video site (like Tudou) keeps advertising the plug-in to its users; therefore the users are highly aware of it.

Different server bandwidth allocation strategies result in different $\omega_s(t)$ and $\omega_p(t)$. We consider two such strategies in our discussion. We say that the $(1 - \Theta(t))N$ users who do not use the P2P plug-in are in the *server* group, and the $\Theta(t)N$ users who use the P2P plug-in are in the *P2P* group. Assume that the demand is independent of the popularity of the plug-in, and that the average consumption rate R is not a function of time t. Further assume the users in both groups have the same viewing behavior. Therefore, the demands of the server and P2P groups are $(1 - \Theta(t))R$ and $\Theta(t)R$, respectively. Let $S_s(t)$ denote the server rate allocated to the server group. We have:

$$\omega_s(t) = \frac{S_s(t)}{(1 - \Theta(t))R}. \quad (17)$$

Let $S_p(t)$ denote the server rate allocated to the P2P group. Similarly, we have

$$\omega_p(t) = \frac{S_p(t) + C(t)}{\Theta(t)R}. \quad (18)$$

where $C(t)$ is the peer contribution, which has been discussed in Section III. Note that $C(t)$ and accordingly $\omega_p(t)$ are functions of $\Theta(t)$.

Fixed server bandwidth allocation

With the fixed server bandwidth allocation, the maximum server bandwidth allocated to the server group and the P2P group are fixed to $\gamma(1 - \Theta(t))R$ and $\gamma\Theta(t))R$, respectively. Therefore, we can have

$$\omega_s(t) = \gamma, \ \omega_p(t) = \min(1, \gamma + \frac{C(t)}{\Theta(t)R}). \quad (19)$$

Flexible server bandwidth allocation

With the flexible server bandwidth allocation, the surplus server bandwidth originally allocated to the P2P group can be re-allocated to the server group, if and only if all peers in the P2P group receive the full video rate. If peers in the P2P group do not receive the full video rate, the server allocates its bandwidth to the groups proportional to the numbers of users. Compared with the fixed bandwidth allocation strategy, the strategy does not affect bandwidth allocation in the P2P group, thus resulting in the exactly same expression of $\omega_p(t)$ as shown in (19). However, since the server bandwidth saved by the P2P group is re-allocated to the server group, there is a correction term for $\omega_s(t)$:

$$\omega_s(t) = \min(1, \gamma + \frac{\Theta(t)}{1 - \Theta(t)}(\max(1, \gamma + \frac{C(t)}{\Theta(t)R}) - 1)). \quad (20)$$

With a given initial $\Theta(0)$, we can iteratively obtain $\Theta(t)$, and the corresponding $\omega_s(t)$, $\omega_p(t)$, $S_s(t)$, and $S_p(t)$ for all time slots.

VI. NUMERICAL RESULTS OF POPULARITY MODEL

A. *Popularity Evolution*

We now present the results on the popularity evolution of the P2P plug-in and its impact on the system performance. We use the default system configuration shown in Section IV-2. We

assume that the server cannot fully serve the users, and set $\gamma = 0.8$ by default. Parameter a in $Q(x) = \min(1, ax)$ is set to 1. We investigate popularity evolution with different parameters for both fixed server bandwidth allocation and flexible server bandwidth allocation strategies. Accordingly, we investigate the received video quality and server rate ratio.

1) Fixed server bandwidth allocation: First suppose the fixed server bandwidth allocation is in use. Figure 9(a) shows the popularity $\Theta(t)$ in a different time slot t with different system configurations, specifically storage v and total server rate ratio γ. We consider three systems and indicate the corresponding (v, γ) in the figure. For all three setups, it takes some time for the P2P plug-in to get popular. First consider systems with $\gamma = 0.8$ but with different v values. The figure shows that the system with $v = 50$ gets popular much faster than the system with $v = 5$. It also shows that popularity evolution is sensitive to the performance of the P2P plug-in. Similarly, tuning the video popularity and incentivizing more helpers should also affect the popularity evolution.

Now consider the systems with $v = 50$ but with different γ values. Both of these systems have the same P2P configuration. We see that the server policy plays an important role in the popularity evolution. When $\gamma = 0.95$, the users can almost receive the full video rate, so that they are less motived to install and use the P2P plug-in.

Figure 9(b) plots the received ratio, i.e., the ratio of the received video rate over the full video rate, for peers in both the server and peer groups. It also shows the average received ratio across both groups. The received ratio in the server group is constant at 0.8, since the server does not re-allocate the saved bandwidth. As the number of peers increases, the performance of the P2P plug-in increases dramatically, as does the received ratio of the peers. The average received ratio increases since more users adopt the P2P plug-in. Eventually, all users use the P2P plug-in, and the received ratio becomes 1. Figure 9(c) shows the server rate ratio. As the P2P plug-in gets more popular and more effective, more server bandwidth is saved.

2) Fixed server bandwidth allocation: Now suppose the flexible server bandwidth allocation is in use. Similar to Figure 9(a), Figure 10(a) shows the popularity evolution of a P2P plug-in with different parameters. Unlike the fixed allocation system, $\Theta(t)$ in the flexible allocation system may not ever approach 1. This is because the saved bandwidth of the P2P group is re-allocated to the server group, and it is possible for the users in the server group to receive the full video rate, discouraging them to install the plug-in. Again, both of the performance of the P2P system (controlled by v) and the server policy (controlled by γ) affect the popularity evolution.

Figure 10(b) plots the received ratios. On one hand, unlike the fixed allocation system, the users in the server group in the flexible allocation system can also quickly receive the full video rate. On the other hand, Figure 10(c) shows that the server ratio is much higher in the flexible allocation system than that in the fixed allocation system. This is a clear trade-off. Importantly, as shown in Figure 10(b) and Figure 9(b), both of the average receive ratios approaches to 1, indicating that all users in both systems eventually can receive the full video rate.
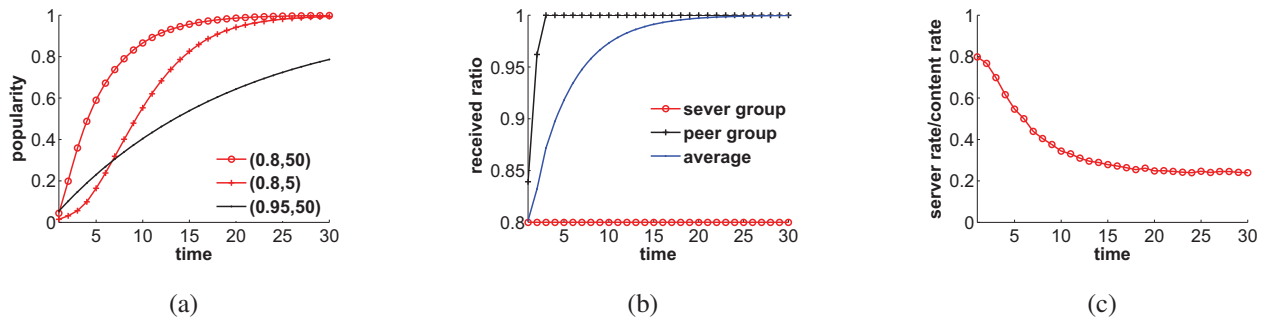
Fig. 9.   Evolution of P2P plug-in with the fixed server bandwidth allocation. (a) Popularity; (b) Received video ratio; (c) Server rate ratio.
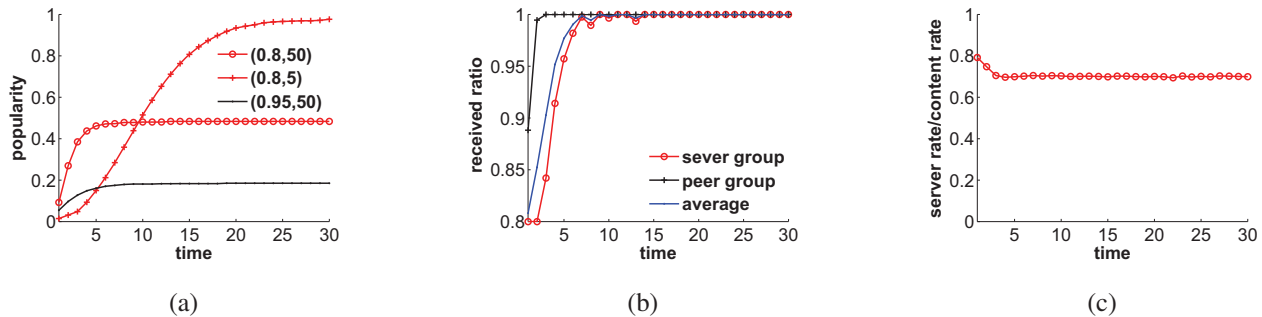

Fig. 10.   Evolution of P2P plug-in with the flexible server bandwidth allocation. (a) Popularity; (b) Received video ratio; (c) Server rate ratio.

VII. CONCLUSION

In this paper we studied peer-assisted distribution of UGC from three perspectives. First, we undertake a measurement study of Tudou's peer-assisted distribution system, which revealed several fundamental characteristics that should be taken into account by modeling efforts. Second, we developed analytical models for peer-assisted UGC distribution, and applied these models to a numerical study of YouTube-like UGC services. And third, we developed analytical models to understand the rate at which users would install P2P client applications to make peer-assisted UGC a success. Our results provide a comprehensive study of peer-assisted UGC distribution, exposing its fundamental characteristics and limitations.

Some of the more important lessons learned include (i) peer-assistance is less effective for UGC systems than for movie systems, due to the large number of long-tail videos; (ii) the performance of peer-assisted UGC can be improved by providing incentives to helpers and/or re-shaping the video quality; (iii) video scheduling between peers is less critical and does not need to be highly optimized.

REFERENCES

[1] http://www.tudou.com/.
[2] http://www.youku.com/.
[3] http://www.ku6.com/.
[4] http://www.ppacc.com/en/index.html/.
[5] http://azureus.sourceforge.net/.
[6] "Comscore: YouTube surpasses 100 million U.S. viewers for the first time."
[7] V. K. Adhikari, S. Jain, and Z.-L. Zhang, "Reverse engineering the YouTube video delivery cloud," in *Proceedings of IEEE HotMD*, 2011.
[8] T. Bonald, L. Massoulie, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: optimal performance trade-offs," in *Proceedings of ACM SIGMETRICS*, 2008.
[9] X. Cheng and J. Liu, "NetTube: Exploring social networks for peer-to-peer short video sharing," in *Proceedings of IEEE INFOCOM*, 2009.
[10] J. Cho and S. Roy, "Impact of search engines on page popularity," in *Proceedings of WWW*, 2004.
[11] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE Journal on Selected Areas in Communication (JSAC)*, January 2004.
[12] Y. Ding, Y. Hu, Y. Du, Z. Liu, L. Wang, K. W. Ross, and A. Ghose, "Broadcast yourself: Understanding YouTube uploaders," in *Proceedings of IMC'11*, 2011.
[13] M. Goldmann and G. Kreitz, "Measurements on the spotify peer-assisted music-on-demand streaming system," in *Proceedings of IEEE P2P*, 2011.
[14] C. Huang, J. Li, and K. W. Ross, "Can Internet VoD be profitable?" in *ACM SIGCOMM*, August 2007.
[15] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P VoD system," in *ACM SIGCOMM*, August 2008.
[16] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in *Proceedings of ACM SIGMETRICS*, 2008.
[17] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomize decentralized broadcasting algorithms," in *IEEE INFOCOM*, 2007.
[18] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson, "Analysis of bittorrent-like protocols for on-demand stored media streaming," in *Proceedings of ACM SIGMETRICS*, 2008.
[19] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming," in *ACM Multimedia*, 2006.
[20] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-Peer video-on-demand system: Design and evaluation," *IEEE JSAC*, vol. 25, no. 9, December 2007.
[21] T.T.Do, K.A.Hua, and M.A.Tantaoui, "P2vod: providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Proceedings of IEEE ICC*, June 2004.
[22] J. Wu and B. Li, "Keep cache replacement simple in peer-assisted VoD systems," in *Proceedings of INFOCOM Mini-Conference*, 2009.
[23] Y. Zhou, D. Chiu, and J. C. Lui, "A simple model for analysis and design of p2p streaming algorithms," in *Proceedings of IEEE ICNP*, 2007.

Summary Review Documentation for

"Peer-Assisted Distribution of User Generated Content"

Authors: Zhengye Liu, Yuan Ding, Yong Liu, Keith W. Ross

REVIEWER #1

The authors show how a simple heuristic for allocating bandwidth to peers is near optimal in most cases, but factors such as system size and storage capacity have a larger impact. Real-world user dynamics are unlikely to be fully captured by the analytic models. Why does the scheduling heuristic deviate from the lower bound?

Strengths: Strengths include well written and easy to follow, interesting set of parameter combinations, and well motivated, and analysis implications are clearly explained.

Weaknesses: No empirical study to back up the theory and no application of analysis to measurement results.

REVIEWER #2

Each of the three contributions feels to some extent incomplete. Details about how measurements were taken and analyzed would be very helpful. Since it is difficult to serve niche "long-tail" content, why not use a cache eviction policy aiming to preserve rarest content? The model assumes an unrealistically "static" system (no churn, no change in set of videos or popularity over time, ...).

Strengths: The paper has some important contributions; the results are interesting and will generate discussion.

Weaknesses: Each contribution in the paper seems incomplete, the measurement methodology is not explained and a different caching strategy would work better than FIFO.

REVIEWER #3

The paper should be better structured, where the theoretical part is appropriately separated from the experimental part, and the theoretical part is separated into main results and (later) derivations.

Strengths: This is a novel problem; the paper includes a good theoretical contribution and extensive experiments.

Weaknesses: Too verbose; structure and readability need improvement.

REVIEWER #4

What are the recommendations based on the assumptions about video popularity? Why is there not always (for measurements) a correlation between the number of views, and the number of peers/speedup factor/percent of data from peers? The "actual" scheduler implemented is unrealistic (centralized, assumes videos infinitely "splittable", no bound on the number of neighbors a peer may have).

Strengths: The topic is relevant, the approach is original, and the results are interesting; the models were well explained and the analysis was clear.

Weaknesses: The model and numerical experiments were not tested against the measurement results; model assumptions may not be justified; focus is not entirely clear.

REVIEWER #5

YouTube content includes longer videos now; how about Tuduo? In the experiment with USA peers, were the servers all in China? The scheduling policy is unrealistic, and provides only an an upper bound on the contribution that peers can make. Assumptions about independence of users, and independence of video selection, are unrealistic.

Strengths: The paper is reasonably well written; the analysis seems sound; and the measurement data is interesting.

Weaknesses: The paper doesn't compare the analytic model results against either simulations or experimental measurements; the modeling is fairly standard looking; the analysis is not used to propose improvements to the designs of peer-assisted systems.

RESPONSE FROM THE AUTHORS

We would like to thank the reviewers for their insightful comments. We included the suggested reference but otherwise did not make any significant changes to the paper.

We are interested in understanding how peer-assisted design can help distribution networks with user generated video content, what the most important design parameters are, and how system performance can be enhanced. To achieve this, the best approach is to have simple models that are based on reasonable assumptions and also capture the most important system characteristics.

However, due to the large design parameter space of peer assisted UGC systems (e.g., single video vs. multiple video, passive caching vs. active caching, and so on) it is not trivial to have a set of reasonable assumptions. We took the approach of measuring a real-world peer assisted UGC system, namely, Tudou and its accelerator, to come up the assumptions. Certain assumptions, e.g., cache in FIFO, appear over simplified, however, they are actually adopted by real-world systems, probably due to system simplicity and relatively good system performance [22]. The objective of the measurement study was not to provide a fully comprehensive understanding of the Tudou peer-assisted UGC system. Instead, we used the measurement study to motivate the analytical study on peer-assisted UGC systems. Importantly, the measurement study helps us select the proper design parameters (and accordingly make proper assumptions) to build the models.

We tried to make the models simple, so that the impact of different design parameters can be easily observed. Because the assumptions are largely based on measurement study, the models are able to capture the most important system characteristics and provide reasonable estimation accuracy. We fully agree that validating the models with real-world measurement data would be highly desirable. However, since our models attempt to model the overall system performance (rather than individual accelerator installations), one may need

to extensively measure the Tudou system to have a global view of the system. This is extremely challenging, since the Tudou system is large, and the Tudou accelerator is closed and proprietary. We are considering validating the proposed models via detailed simulations with traces collected from real-world UGC systems [12].