An Adaptive Cloud Downloading Service

Yipeng Zhou, *Member, IEEE,* Tom Z. J. Fu, Dah Ming Chiu, *Fellow, IEEE,* and Yan Huang.

Abstract

Video content downloading using the P2P approach is scalable, but does not always give good performance. Recently, subscription-based premium services have emerged, referred to as *cloud downloading*. In this service, the cloud storage and server caches user interested content, and updates the cache based on user downloading requests. If a requested video is not in the cache, the request is held in a waiting state until the cache is updated. We call this design *server mode*. An alternative design is to let the cloud server serve all downloading requests as soon as they arrive, behaving as a helper peer. We call this design *helper mode*. Our model and analysis show that both these designs are useful for certain operating regimes. The helper mode is good at handling high request rate, while the server mode is good at scaling with video population size. We design an adaptive algorithm (AMS) to select the service mode automatically. Intuitively, AMS switches service mode from server mode to helper mode when too many peers request for blocked movies, and vice versa. The ability of AMS to achieve good performance in different operating regimes is validated by simulation.

Keywords

cloud server, peer-to-peer, helper, file downloading, video;

I. Introduction

Video content distribution is a challenging research problem because of its high bandwidth requirement and the fast growing video population. In recent years, it is reported that Internet traffic is already dominated by video [1].

Yipeng Zhou, Tom Z.J. Fu and Dah Ming Chiu are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (email: {zyp009,zjfu6,dmchiu}@ie.cuhk.edu.hk).

Yan Huang is with Tencent Research (e-mail: {galehuang@qq.com}).

Manuscript received March 2, 2012; revised July 27, 2012; finalized Nov. 19, 2012.

The mechanisms to distribute video content include CDN (such as Akamai [2]) and Peer-to-Peer (P2P). CDN is a traditional solution based on deploying servers at the edge of the network, near video access points. Scalability is a limitation of CDN because the server capacity becomes a bottleneck when there is a large number of concurrent peer requests.

Later P2P becomes a popular solution, and is adopted by many content distribution systems such as [3]–[5]. In a P2P content distribution system, peers who create demand for videos also share their content with other peers. The service capacity thus increases automatically with increasing peer population, making scalability an advantage of the P2P solution. Usually, however, peers are not stable since they can leave and join the system at any time. Furthermore, the online peers cannot always provide stable and powerful service capacity. Therefore, most commercial P2P VoD streaming systems (such as [3], [4]) deploy additional servers as ultimate backup for service capacity. Peers will get support from a server if they cannot obtain enough content for streaming from other peers.

In file sharing scenarios, however, dedicated server is not commonly deployed for service capacity. Peers requesting unpopular videos often suffer low downloading rate. To remedy this, a *cloud downloading* service is deployed in P2P video downloading systems to enhance the performance of downloading. A cloud storage system is used to cache a large fraction of video content, and high bandwidth is provided to access this cache. Peers can get a big performance boost by connecting to cloud downloading. The architecture and implementation of such a cloud downloading system is introduced in [1]. This work is based on a popular P2P file sharing system [6] in China with more then ten million users.

On the other hand, smart phone becomes quite popular in recent years. File downloading demand by mobile devices is growing fast. Some literature proposed mobile P2P architecture and protocols for file sharing among mobile devices such as [7]–[9]. However, due to limited uplink bandwidth, CPU and memory resource, unstable availability and limited battery power, it is very challenging to rely on a P2P approach to realize high speed file downloading. Therefore, a cloud downloading system with high service capacity is necessary. How to utilize the resource of cloud server efficiently is the key to determine system performance.

There are two generic service modes for cloud servers. In the first mode, the cloud server is primarily focused on serving the content already cached at the cloud storage system. Requests for content not in the cache are blocked until such content becomes cached. The cloud storage system updates its cache periodically to replace content without requests by content with requests awaiting. We call this the *server mode*. This is what is implemented in [1]. An alternative mode is the *helper mode*, in which the cloud server does not block any requests. For videos that are not cached, the cloud server simply relay chunks

from some peers to other peers, acting as a helper peer. One contribution of our study is to compare these two modes analytically. The results are interesting, in the sense that both modes can be advantageous for some operating regimes - the server mode when video population is large compared to cache size, and the helper mode when peer request rate is high compared to server bandwidth. We integrate these two modes into a single adaptive cloud downloading service.

The rest of the paper is organized as follow: in section II, we introduce notations and the assumptions in our analysis. In section III, we discuss a static model for simple analysis. The analysis of both helper mode and server mode in a dynamic model is included in section IV. An automatic service mode selection algorithm is proposed in section V. In section VI, we use simulation to validate our model. Before conclusion, we include some discussion of related works in section VII.

II. Assumption

To keep the problem tractable, we make some simplifying assumptions for the models we study in this paper.

- The set of videos remains unchanged. The study of video population and popularity churn is for future work.
- Peer population is much larger than video population [10].
- All videos have the same size. The cloud storage can only store a small subset of the videos. In simulation, we do use heterogeneous videos length obtained from practical system [6].
- Peers have the same upload capacity. In most networks, peer's download capacity is much more than upload capacity. We do not consider the constraint of download capacity in this work.
- Each peer issues one downloading request at any time. A multiple concurrent request model will be considered in future study.
- Peers are fully connected, forming a full mesh topology [10]–[12].
- The cloud server is able to replace any cached video with a new one instantly. In other words, we assume the time for video replacement can be ignored.
- The video size is large enough to be divided into an infinite number of small chunks. Based on this assumption, we build a fluid model [11], [12].

We assume homogeneous upload capacity to simplify the scheduling strategy. In this paper, each peer share its upload capacity to all other peers downloading the same video equally. In heterogeneous network, peers with different upload capacity can be treated differently. Downloading performance can be affected significantly by using different scheduling strategies. Intuitively speaking, other peers be benefit from

delaying the leave of super peers, though this is not fair for super peers with large upload capacity. The tradeoff between fairness and download performance in P2P file downloading system is discussed in paper [13]. Since, we focus on analyzing download performance we only analyze homogeneous scenario.

Our notations are given below:

- N is the peer population. Since one peer only issues one request, the peer population is equal to the request population. The peer population for a video j is denoted by N_j.
- $\mathcal{M}$ is the set of all videos. The total number of videos is M.
- $\mathcal{K}$ is the subset of videos cached by cloud storage. The number of cached videos is K. $\mathcal{M} - \mathcal{K}$ is the set of videos not cached by cloud storage.
- Each video has a relative popularity η_j, i.e. the probability that the video is the target of any request. By definition, $\sum_{j=1}^{M} \eta_j = 1$.
- Peer upload capacity is U. The upload capacity of the server (that sources a video content) is U_0.
- The (total) upload capacity of the cloud server(s) is H.
- File size for each video is denoted by F.

The cloud server is able to serve multiple requests simultaneously. We assume a simple fair sharing scheduling strategy. In other words, the cloud server allocates its upload bandwidth evenly among all video requests being served.

III. Static Model

We begin analysis with static model. Let N denote the number of peers, hence downloading requests; by *static*, we mean the number requests remains unchanged at N.

Given the full mesh, and uplink limiting assumptions, the peers, server and cloud server can be thought forming a logical star network as shown in Fig. 1. This abstraction is consistent with studies in [11], [12]. Here, we add the analysis with the addition of helpers. Helpers are not interested in any movie and just help amplify the system capacity to improve user experience. In this work, the helper is a cloud server with storage to cache some movies to enhance service capacity.

Given cloud servers as helpers, there are two intuitive strategies to serve each downloading request:

- Helper Mode: The cloud server begins to help the downloading request without caching the video in its cloud storage. The cloud server downloads video chunks from the P2P system and redistributes it to other peers who are without these chunks. Then, these chunks are discarded.
- Server Mode: The downloading requests are not served until the requested video is cached by the cloud storage. The requests for videos not in the cache are blocked. The cloud storage is updated

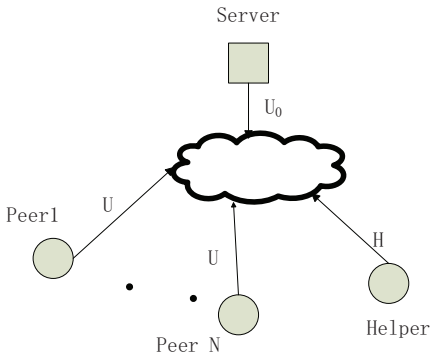

Fig. 1. File downloading with helper.

periodically in order to serve blocked requests.

Now, we analyze the static case by considering a single movie.

The performance objective of minimizing downloading time is equivalent to maximizing throughput rate. The maximum throughput is also referred to as the capacity of the P2P system, denoted C, and has been derived by previous studies. For example, the derivation of the capacity without helpers can be found in [11], [12], as:

$$C = min\{U_0, U + \frac{U_0}{N}\}. \tag{1}$$

Intuitively speaking, there are two possible bottlenecks. One is the capacity of source, i.e. server. The other one is the total capacity. By constructing a set of 1-hop and 2-hop spanning trees, [12] and [11] derive the maximum throughput and find a centralized scheduling strategy to achieve the maximum throughput.

If the cloud server adopts helper mode, the cloud server needs to download video content from P2P or the source, which is a waste of bandwidth compared with server mode. To maximize the throughput, the cloud server should distribute any particular chunk to all peers. Thus, the efficiency to utilize cloud server's upload capacity is $\frac{N-1}{N}$.

Theorem *1:* The maximum throughput in helper mode is:

$$C = min\{U_0, U + \frac{U_0 + \frac{N-1}{N}H}{N}\}. \tag{2}$$

The proof is in appendix.

The server mode case is a straightforward extension of the basic single-file downloading result in Eq. 1:

Theorem *2:* If the cloud server already cached this video, the maximum throughput is

$$C = min\{U_0 + H, U + \frac{U_0 + H}{N}\}. \tag{3}$$

In the static model, the server mode is always better than the helper mode with the cost of additional storage to cache the particular video. since we only consider a single video in the static model, the storage cost can be ignored. However, in practical systems, the cloud server is unable to cache all videos. In that case the helper mode may surpass the server model. We will further analyze this problem in the dynamic model introduced in the next section.

The above results are upper bounds that is achieved (theoretically) assuming *centralized* scheduling. In our dynamic model, we will model the effectiveness of distributed scheduling using a parameter, in a fashion similar to the *sharing efficiency* parameter used in [10].

IV. Dynamic Model

In this section, we extend our discussion to a multi-video system. The peer downloading requests arrive as a Poisson process. Once downloading is complete, a peer leaves the system immediately. To simplify the expressions, we ignore U_0 here, since it is a constant and is relatively insignificant compared to the cloud service capacity. A distributed chunk scheduling strategy is adopted for exchanging video content between peers. For benchmarking, we also analyze the unlimited-cache case, in addition to the cases of helper mode and server mode. In the unlimited-cache case, we assume all videos are stored in the cloud storage.

A. The Unlimited-cache Case

In this case we assume the cloud storage is able to store all videos, i.e. $\mathcal{K} = \mathcal{M}$.

For video j, the arrival rate of requests is $\lambda \times \eta_j$. We can write the following equation for each video:

$$\begin{aligned} &\frac{dN_j}{dt} = \lambda \times \eta_j - \frac{\beta_j N_j U + H \times \frac{N_j}{N}}{F}, j = 1, \ldots, M, \\ &s.t. \quad \sum_{j=1}^{M} N_j = N, \quad \sum_{j=1}^{M} \eta_j = 1. \end{aligned} \tag{4}$$

β_j is the effectiveness of file sharing, which is the efficiency for peers to help each other download content. It is shown in [10], $\beta_j \approx 1$ when the video can be divided into many small chunks. For simplicity, we let $\beta_j = 1$ in our analysis. For EQ. 4, on the right hand side, the first term is the arrival rate of requests and the second term is the departure rate. The departure rate is determined by the system capacity.

In steady state, we have $\frac{dN_j}{dt} = 0$, by summing M equations we get:

$$\sum_{j=1}^{M} \lambda \eta_j = \frac{1}{F}(\sum_{j=1}^{M} N_j U + H) \tag{5}$$

From Little's law and EQ. 5, the average downloading time is:

$$T_{opt} = \frac{N}{\lambda} = \frac{\lambda F - H}{\lambda U} = \frac{F}{U} - \frac{H}{\lambda U}. \tag{6}$$

It is straightforward to deduce that the average downloading time becomes $\frac{F}{U}$ as λ goes to infinity. Note, EQ. 4 is not valid when $N_j = 1$. Since our focus is on the difference between helper mode and server mode, this special case is neglected. The implication is that $\lambda F > H$ must be true when $T_{opt} > 0$. This rules out the scenarios when H is large enough to serve all downloading requests (without needing P2P).

B. Helper Mode

It is reasonable to consider the regime that only a fraction of all the videos are stored in the cloud. In helper mode, the cloud server serves all downloading requests whether the video is cached or not. If the video is not cached, the cloud server will relay and amplify the video content downloaded from other peers. The cloud server has no interest in replicating any video. As implied by EQ. 2, about $\frac{1}{N_j}$ of the cloud server's bandwidth (allocated to video j) will be wasted.

It is not easy to utilize the bandwidth of cloud server as efficiently as the analytical upper bound in theorem 1, assuming helper mode with distributed P2P scheduling. Although some paper [14] claim that full helper bandwidth utilization can be achieved using distributed P2P scheduling, they still require centralized coordination. Below, we first analyze the average downloading time of the best case for helper mode. Then, we analyze the average downloading time based on distributed scheduling.

When $K < M$, the differential equation for each video is:

$$\begin{aligned} &\frac{dN_j}{dt} = \lambda \times \eta_j - \frac{N_j U + H \times \frac{N_j}{N} \times \frac{N_j + x_j - 1}{N_j}}{F}, \\ &s.t. \quad \sum_{j=1}^{M} N_j = N, \quad \sum_{j=1}^{M} \eta_j = 1, \\ &\qquad \sum_{j=1}^{M} x_j = K, x_j = \{0, 1\}. \end{aligned} \tag{7}$$

The variable x_j denotes whether movie j is cached by cloud server. $x_j = 1$ if the movie is cached, otherwise $x_j = 0$. In steady state, we have $\frac{dN_j}{dt} = 0$. By summing the above differential equations, we

have:

$$\begin{aligned}\lambda &= \frac{1}{F}(\sum_{j=1}^{M} N_j U + H\frac{N+K-M}{N}) \\ N &= \frac{\lambda F - H + \sqrt{(\lambda F-H)^2 + 4U(M-K)}}{2U}. \\ T_h &= \frac{N}{\lambda}. \end{aligned} \tag{8}$$

The gap between the downloading time in helper mode and the optimal case is:

$$\begin{aligned}\Delta T &= T_h - T_{opt} \\ &= \frac{\sqrt{(\lambda F-H)^2 + 4U(M-K)}}{2\lambda U} - \frac{\lambda F - H}{2\lambda U}. \\ &= \frac{2(M-K)}{\lambda(\sqrt{(\lambda F-H)^2 + 4U(M-K)} + \sqrt{(\lambda F-H)^2})} \\ &< \frac{M-K}{\lambda(\lambda F - H)} \end{aligned} \tag{9}$$

From EQ. 9, the average downloading time gap between helper mode and the optimal case increases rapidly with increasing $M-K$ but decreases with λ. The insight is, when peer population increases, the helper mode works more efficiently because the fraction of wasted bandwidth is roughly $\frac{1}{N_j}$ of the bandwidth allocated to each video. As discussed earlier, we assume $N_j \geq 1$, hence $\frac{N_j-1}{N_j} > 0$. Thus the average downloading time will never go to infinity. The worst case is when the helper's bandwidth is totally wasted and the resulting average downloading time is $\frac{F}{U}$.

However, as far as we know, there is no pure distributed algorithm to achieve the maximal throughput discussed in theorem 1. To generalize our model, we let α_j denote the probability that the cloud server can help any peer watching movie j by caching only one particular chunk. Thus, the number of requests satisfied by cloud server is about $\alpha_j \times N_j$ if movie j is in helper mode. To have a uniform expression, we let $\alpha_j = 1$ if $j \in \mathcal{K}$. $j \in \mathcal{K}$ also implies $x_j = 1$ (in EQ. 10). The actual number of requests served by cloud server is $N' = \sum_{j=1}^{M} \alpha_j \times N_j$. Server bandwidth is divided equally among N' peers in this case. Thus, the differential equation becomes:

$$\begin{aligned}\frac{dN_j}{dt} &= \lambda \times \eta_j - \frac{N_j U + H \times \frac{\alpha_j N_j}{N'} \times \frac{\alpha_j N_j + x_j - 1}{\alpha_j N_j}}{F}, \\ & \qquad j = 1, \ldots, M. \\ N' &= \sum_{j=1}^{M} \alpha_j N_j, \end{aligned} \tag{10}$$

Intuitively, N' is the expected population of peers served by the cloud. By summing the equations above together, we get a similar result to 8:

$$\lambda = \frac{1}{F}(NU + H\frac{N' + K - M}{N'})$$

Without the knowledge of movies cached by the cloud server, it is difficult to get the value of N' and average downloading time T_h in closed form. But, similar to the best case of helper mode, the performance will degrade significantly with increasing movie population, i.e. $M - K$. The wasted bandwidth by any particular movie is $\frac{H}{N'}$, if that movie is not cached by the cloud storage. Since $\alpha_j < 1$ and $N' < N$ if $j \in \mathcal{M} - \mathcal{K}$, the performance is worse than the best case of helper mode, as expected.

Since the performance result depends on N' and α_j, we need to consider a reasonable value for them to use in our simulations. One way to establish a reasonable value for α_j is based on a stochastic model of how many chunks a peer has [15], [16]. If each movie is split into B chunks, peers can be in states $0, \ldots, B-1$. Peers with i chunks is in state i. In steady state, the probability that any peer in state i is π_i.

$$\alpha_j = \sum_{i=0}^{B-1} \pi_i \frac{B-i}{B} = 1 - \sum_{i=0}^{B-1} \frac{i\pi_i}{B}$$

From EQ. 11, the distribution of π_i can be used to determine the value of α_j. According to the result of [15], [16], we have $\pi_0 < \pi_1 < \cdots < \pi_{B-1}$.

$$\alpha_j \leq 1 - \sum_{i=0}^{B-1} \frac{i}{B}\frac{1}{B} = \frac{1}{2}(1 + \frac{1}{B}) \approx \frac{1}{2}$$

Equality is achieved when $\pi_0 = \pi_1 = \cdots = \pi_{B-1} = \frac{1}{B}$. Thus, the value of α_j could be very low in a distributed P2P system. In our simulation study, we let $\alpha_j = \frac{1}{2}$ in most of the scenarios we simulated.

C. Server Mode

In server mode, any request for a video not cached is blocked until the cloud storage update. Therefore, it is necessary to differentiate peers as *downloading peers* and *waiting peers*. The former are peers downloading a video cached by the cloud storage, whereas the latter are peers waiting for the cloud storage to be refreshed. The hit rate, denoted by γ, is the probability that the requested video is cached by the cloud storage. The average time interval between updates of cloud storage is denoted by $\frac{1}{\mu}$. A new video is brought into the cloud storage by removing a video without requests. The population of waiting peers and downloading peers are denoted by N_W and N_D respectively. They satisfy the following differential equations:

$$\Delta N_D = \lambda\gamma\frac{1}{\mu} + N_W \times \frac{K - R(N_D, \mathcal{K})}{R(N_W, \mathcal{M} - \mathcal{K})} - \frac{N_D U + H}{F}\frac{1}{\mu},$$
$$\Delta N_W = \lambda(1-\gamma)\frac{1}{\mu} - N_W \times \frac{K - R(N_D, \mathcal{K})}{R(N_W, \mathcal{M} - \mathcal{K})}. \tag{11}$$

Let $R(N, \mathcal{K})$ be the expected number of videos requested by N peers from a set $\mathcal{K}$. In steady state, $\Delta N_D = 0$ and $\Delta N_W = 0$, thus:

$$N_D = \frac{\lambda F - H}{U},$$
$$T_D = T_{opt} = \frac{F}{U} - \frac{H}{\lambda U},$$
$$N_W = \lambda(1-\gamma)\frac{1}{\mu} \times \frac{R(N_W, \mathcal{M} - \mathcal{K})}{K - R(N_D, \mathcal{K})}. \tag{12}$$

If peers do not care about waiting time, peers enjoy optimal downloading time in server mode. Usually, however, the waiting time does matter and must be considered. In the general case, it is complicated to analyze the waiting time. We study the worst case in this work. The worst case is achieved by

$$min \qquad \{\gamma, -R(N_D, \mathcal{K}), \frac{1}{R(N_W, \mathcal{M} - \mathcal{K})}\}. \tag{13}$$

Intuitively speaking, the worst is achieved by maximizing waiting peers. As shown in EQ. 12, we minimize the three terms $\gamma, -R(N_D, \mathcal{K})$ and $\frac{1}{R(N_W, \mathcal{M}-\mathcal{K}}$ simultaneously to maximize waiting peers. These values are related to the video popularity distribution. For hit rate, if the cloud server adopts the Least-Frequently-Requested (LFR) cache management strategy [1], the top K popular videos will be cached by cloud storage. It is generally accepted that the video popularity distribution is Zipf [17]. Without loss of generality, the videos can be ranked by decreasing order of popularity. The popularity of the j^{th} video can be expressed as:

$$\eta_j \quad = \quad \frac{j^{-\theta}}{\sum_{i=1}^{M} i^{-\theta}}, \tag{14}$$

where $\sum_{j=1}^{M} \eta_j = 1$. η_j means the probability that any request for video j is η_j. With the LFU strategy, the hit rate is:

$$\gamma = \frac{\sum_{j=1}^{K} j^{-\theta}}{\sum_{j=1}^{M} j^{-\theta}}, \tag{15}$$

Lemma *1:* γ is an increasing function with K and θ.

The proof is in the appendix. When $\theta = 0$, the hit rate achieves the minimum value of $\frac{K}{M}$.

Since N_D is independent on the hit rate γ and N_W, the analysis of $R(N_D, \mathcal{K})$ is easier. Let $I(N_D, j)$ denote the random variable that the video j is requested by some peer from a total of N_D peers:

$$I(N_D, j) = \begin{cases} 1, & \text{j is requested by some peer} \\ 0, & \text{No peer selects video j} \end{cases} \tag{16}$$

$$Pr\{I(N_D, j) = 1\} = 1 - (1 - \eta'_j)^{N_D}, j \in \mathcal{K},$$

where η'_j is the relative popularity of videos in set $\mathcal{K}$. Thus,

$$\begin{aligned} R(N_D, \mathcal{K}) &= \sum_{j \in \mathcal{K}} E[I(N_D, j)] = K - \sum_{j \in \mathcal{K}} (1 - \eta'_j)^{N_D}, \\ s.t. \quad & \sum_{j \in \mathcal{K}} \eta'_j = 1. \end{aligned}$$

This is a convex optimization problem. $R(N_D, \mathcal{K})$ achieves the minimum value when all videos have the same popularity, i.e. $\eta'_j = \frac{1}{K}$. The maximized value is $R(N_D, \mathcal{K}) = K - K(1 - \frac{1}{K})^{N_D} \approx K - Ke^{-\frac{N_D}{K}}$

Similarly, we can derive $R(N_W, \mathcal{M} - \mathcal{K})$ as:

$$\begin{aligned} R(N_W, \mathcal{M} - \mathcal{K}) &= \sum_{j \in \mathcal{M} - \mathcal{K}} E[I(N_W, j)] \\ &\leq (M - K)(1 - e^{-\frac{N_W}{M-K}}). \end{aligned}$$

For server mode, we study the case when $\frac{N_W}{M-K} \gg 1$. In other words, we focus on the worst case with heavy load. Thus,

$$\begin{aligned} N_W &= \lambda(1 - \gamma)\frac{1}{\mu}\frac{R(N_W, \mathcal{M} - \mathcal{K})}{K - R(N_D, \mathcal{K})}, \\ &\approx \lambda(1 - \gamma)\frac{1}{\mu}\frac{M - K}{Ke^{-\frac{N_D}{K}}}, \\ T_W &= \frac{N_W}{\lambda(1 - \gamma)}, \\ &= \frac{1}{\mu}(\frac{M}{K} - 1)e^{\frac{N_D}{K}} = \frac{1}{\mu}(\frac{M}{K} - 1)e^{\frac{\lambda F - H}{UK}}. \end{aligned}$$

Clear, the waiting time increases rapidly with λ.

Helper mode is scalable in request rate because high arrival rate λ will improve efficiency of bandwidth utilization. In contrast, server mode is scalable in video population but cannot achieve good scalability with peer population. When λ goes to infinity, the video downloading time is mainly determined by the P2P bandwidth resource, i.e. U. However, the cloud server becomes a bottleneck in server mode, with more and more peer requests blocked. On the other hand, when video population M goes to infinity, the

average peer population requesting for each video becomes less. The helper mode tends to waste more bandwidth than server mode.

V. Adaptive Algorithm

From the above analysis, there are both strengths and drawbacks for both the helper mode and server mode. The helper mode wastes P2P resource because the cloud server needs to keep downloading new content to help peers; while the server mode wastes the bandwidth resource of blocked peers. In this section, we design an adaptive algorithm to determine the service mode for each movie. The cloud server adjusts its strategy periodically, by running the following Automatic Mode Selection (AMS) algorithm to determine the mode for each movie. We assume the value of N' is known. The movies in helper mode have higher priority to be included into cloud storage. Then, we consider the other movies in the order of decreasing peer population.

Algorithm 1 Automatic Mode Selection (AMS) Algorithm

for Each movie j not in $\mathcal{K}$. **do**
  if The active movie is less than K **then**
    Update cloud storage to add movie j by replacing any movie without request.
    $N' = N' + N_j$
  else
    if $\frac{H}{N'+\alpha_j N_j} < N_j U$ **then**
      Use helper mode for movie j.
      $N' = N' + \alpha_j N_j$
    else
      Keep blocking peers requesting for movie j
    end if
  end if
end for

Based on our analysis, the weakness of helper mode is the additional bandwidth cost to download the requested video by cloud server. The benefit is that more peers can contribute their upload capacity by switching their state from waiting to downloading. Thus Alg. 1 compare the cost and the benefit and start helper mode once the benefit is larger than cost.

AMS algorithm is friendly for implementation. In practice, N' can be calculated by counting the average number of requests served by the cloud server. The value of α_j can be estimated by the ratio $\frac{N'_j}{N_j}$, where N'_j is the average number of peers viewing video j can be helped by cloud server and N_j is the peer population to view video j. In the next section, we will use simulation to validate the effectiveness of AMS algorithm.

VI. SIMULATION

In this section, we use simulation to validate our analytical model proposed in section IV and test the performance of our AMS algorithm proposed in section V.

A. Simulation Setting

We use a time-slotted discrete simulation model. The requests arrive with a Poisson process. At the end of each time slot, those peers who have finished downloading leave the system. The peers generating new video requests join the system. The skewness of the video popularity distribution is set as $\theta = 0.3$, which is introduced in EQ. 14. The unit of peers' upload capacity is 0.5 unless otherwise stated. The file size for all movies is 100. The upload capacity of cloud server is 5000. Each experiment is run more than 1500 time slots. The time interval to update cloud storage is 5 time slots for server mode. For each simulation. the lower bound of average downloading time is achieved by letting $\mathcal{K} = \mathcal{M}$. **modified by yipeng** Average downloading time is the average time slots for a peer to spend to download a video.

B. Simulation Result

a) Varying λ: The first simulation is used to compare the performance of each algorithm by varying arrival rate of new peers. As shown in Fig. 2, the arrival rate λ varies from 30 to 130. The total number of videos is 4000. The capacity of cloud storage is $K = 1000$. From the simulation result, we verified that helper mode works better than server mode in situations with large peer population. As we expected, the AMS algorithm achieves better performance than helper mode and server mode.

b) Varying M: In the second simulation, we compare different algorithms by varying total movie population M. As shown in Fig. 3, the video population varies from 2000 to 6000. The arrival rate is fixed at $\lambda = 70$. The capacity of cloud storage is $K = \frac{M}{4}$. From the simulation result we can find that the server mode is better at dealing with cases with large video population than helper mode. And AMS achieves the best performance over different values of M.

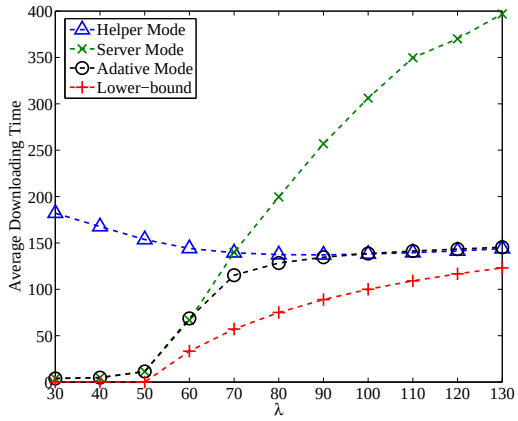

Fig. 2. Simulation by varying λ from 30 to 130.

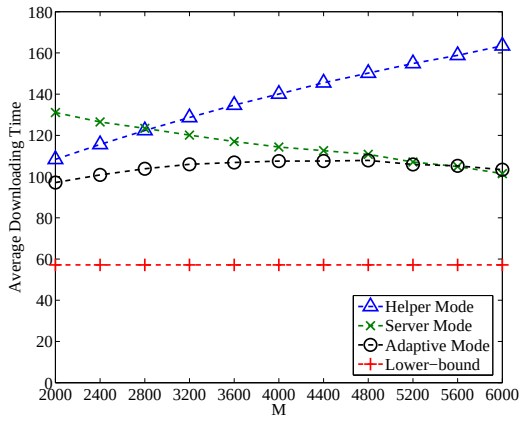

Fig. 3. Simulation by varying M from 2000 to 6000.

c) Varying α: In the analysis of helper mode, we introduce a parameter α_j for each movie j to indicate the efficiency to utilize the cloud server's bandwidth, if movie j is not cached. The value of α_j is determined by the scheduling algorithm used by peers to exchange content, which is out of our control. To consider different degree of scheduling efficiency, we simulate different values of α from 0.2 to 1. The arrival rate is 70, Movie population is $M = 4000$ and the capacity of cloud storage is $K = 1000$. As shown in Fig. 4, server mode will not be affected by α. AMS still achieves the best performance for different α.

d) Varying U: In different systems, the peers must have quite different upload capacity. For example, in mobile systems, peers' upload capacity is very small compared with traditional peers in wired networks.

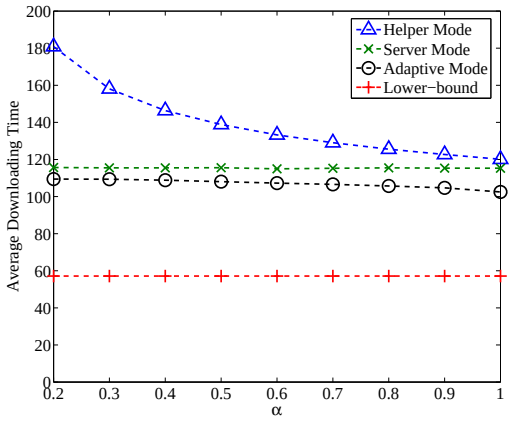

Fig. 4. Simulation by varying α from 0.2 to 1.

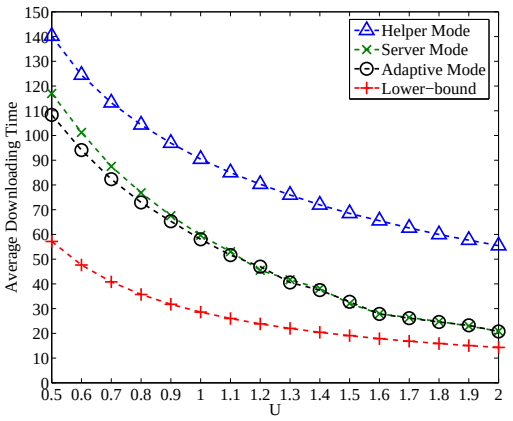

Fig. 5. Simulation by varying U from 0.5 to 2.

To study the influence of peers' upload capacity, we change the upload capacity U from 0.5 to 2. Except $\alpha = 0.5$, the other parameters are the same with the previous case. The result is shown in Fig. 5. The performance is better with increasing U for all cases. How the value of U will determine the mode selection for AMS algorithm is studied in the next experiment.

e) System parameter v.s. mode selection: From the above simulation results, we validated our model and the effectiveness of AMS algorithm by comparing average downloading time. However, how AMS adapts service mode for different movies in different cases is unknown. Thus, we summarize the average number of movies blocked by AMS in different simulations in Fig. 6. The result is explained below:

- λ: As the arrival rate λ increases, at some point, the number of blocked movies start to decrease.

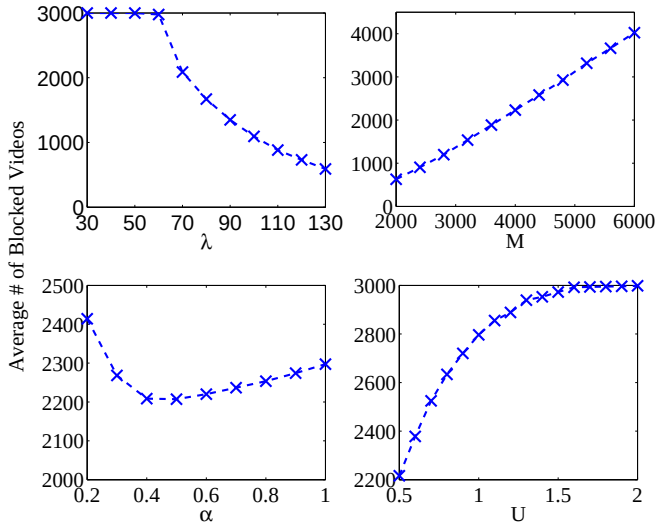

Fig. 6. The number of blocked movies in different cases.

This is because fewer blocked movies let more peers participate in file downloading.

- M: As movie population goes up, the average number of peers viewing each movie decreases, otherwise too much cloud server's bandwidth will be blocked.
- α: With the increasing of α, the helper mode works more efficiently and peers download movies with higher rate. The blocked movies will decreases. However, large α will waste less cloud server's bandwidth in helper mode and results that more movies select helper mode. This leads to a U shape for the number of blocked movies.
- U: With increasing value of U, the peer population becomes small. Though the peers' capacity is more valuable, the helper mode wastes more bandwidth with less peer population. The result is that more movies will be blocked due to the small peer population.

C. Trace Driven Simulation

Next, we use the trade data from practical system to drive our simulation. The real trace data is collected from one of QQ Xuanfeng system [6] during one day (June 6 2012, Wednesday). The data from practical system reflects the user arrival pattern and movie accessed pattern in real word. It is a good source to validate our proposed algorithms by driving our simulation. Fig. 7 plots the movie download requests arrivals grouped in 5-minute-window. During the period used to derive simulation, the total number of accessed videos varies from 7261 to 59281.

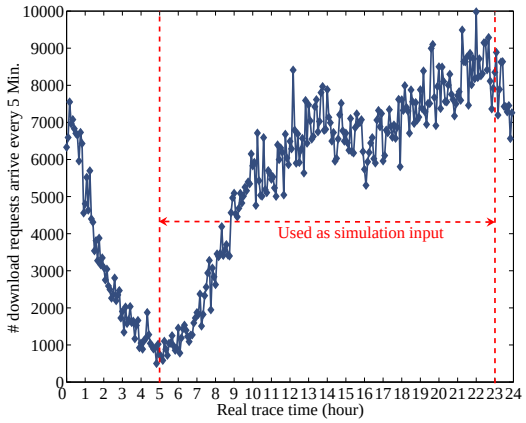

Fig. 7. Movie download requests arrival in 5-minute-window from QQ Xuanfeng system on June 6, 2012

From Fig. 7, the cloud server experiences light load from 4:00 - 7:00 and heavy load during 20:00 - 22:00. In our simulation, we choose the trace data from 5:00 to 23:00 such that we can test the algorithm under both light and heavy load scenarios. Since the arrival rate keeps increasing during this period, we use throughput, which is introduced in EQ. 1, as the metrics to measure performance of each mode.

In our simulation, we use the same movie request with the movie request record in the trace data. Time slot is set as 10 seconds of real time. Each simulation experiment lasts 1 hour. For other parameters, we set $H = 5000.0$, $U = 1.0$, $K = 1000$, update period $T_{\text{Update}} = 2$ and $\alpha = 0.5$. All rate unit is "MBytes" per time slot. The result is plotted in Fig. 8,

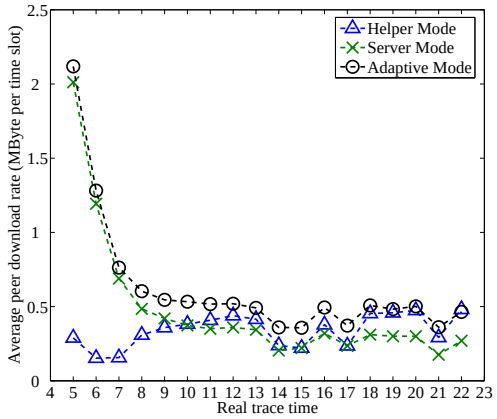

Fig. 8. Simulated throughput driven by trace data from QQ Xuanfeng.

As shown in Fig. 8, the X-axis is time (hours), which is the same as the region marked by red lines in Fig. 7. The Y-axis is the average download throughput (MBytes per time slot per peer). We can observe that server mode performs better during light load period (5:00 - 9:00), while helper mode becomes better during heavy load period (after 10:00). Since AMS algorithm automatically adapts with requests arrival, it performs best during the whole period.

VII. Related Work

Our work is inspired by the recent paper [1] that introduced the architecture and design of a cloud downloading system in operation. The improved average downloading time is measured from the real-world system and presented. In this system, however, the cloud sever operates only in server mode, with the cloud storage refreshed using a LFU cache management strategy. In this paper, we studied their strategy using an analytical model, and compare it to a different server strategy called the helper mode, and showed the advantage in each case.

There are many papers analyzing the P2P file sharing system. Work [11], [12] studied the static P2P file downloading system. They used the star network and spanning trees to derive the maximal throughput, based on fluid assumption (when the files are divided into infinite number of chunks). The paper [18] proved that the maximal throughput can be achieved by distributed algorithm if the file can be divided into infinite number of chunks.

The work [14] designed an algorithm to send and forward file chunks for P2P file distribution system. The algorithm is also based on the selecting spanning trees to maximize throughput, and included the consideration of helpers whose job is to help accelerate file distribution, rather than download file for themselves. However, this algorithm is not a pure distributed algorithm.

The work [10] first introduced a dynamic fluid stochastic model for P2P file downloading system in homogeneous network. The original model only analyzed the system with one downloading file, without considering helpers. The work [13] extended the model in [10] to heterogeneous network and studied the trade-off between fairness and average downloading time in P2P file distribution system. By delaying the super peers a little, the average downloading time can be reduced. By tuning the weight of discriminative uploading and uniform uploading, the authors showed the whole design space between fairness and performance.

However, in practical system, files can not be split into infinity number of chunks. The paper [15], [16] proposed discrete stochastic model for P2P file sharing system. The file is split into finite number of chunks. Peers are classified into different categories by the owned number of chunks. The average

downloading time can be derived by analyzing the steady state.

There are many papers on the protocol and architecture design of a mobile P2P system, for example [7]–[9]. But these protocols from several years ago have not made into practical implementations for various difficulties. We believe a cloud downloading system as described in this paper provides a more feasible solution.

For example, the literature [7] proposed architecture for a structured mobile P2P system. The paper [8] presents a new scalable and reliable mobile P2P network architecture which is composed of cellular Ad Hoc network, wireless access network and 3G core network. The new design emphasize both scalability and reliability issues. The paper [9] discussed feasible architecture for mobile P2P by enhancing the eDonkey protocol.

VIII. Conclusion

In this work, we build a theoretical model to analyze different strategies for a cloud downloading system. In particular, helper mode and server mode are used as abstraction of two different design philosophies - using the cloud as peer or as server. Our analysis reveals that each strategy can be advantages, for certain operating scenarios. Helper mode wastes some server bandwidth, but is best at leveraging P2P capacity when request load is high. On the other hand, server mode is most efficient for dealing with large video population relative to the cache size. We design an automatic mode selection (AMS) algorithm to choose the suitable service mode for different scenarios. Our analysis helps a cloud downloading system to optimize its design. We also discuss the potential benefit to apply our result for the mobile P2P case.

Acknowledgment

The authors wish to acknowledge the support from HK RGC grant CUHK411611.

References

[1] Y. Huang, Z. Li, G. Liu, and Y. Dai, "Cloud download: Using cloud utilities to achieve high-quality content distribution for unpopular videos," in *ACM Multimedia*, 2011.

[2] A. web site, "http://www.akamai.com."

[3] PPLive, "http://www.pptv.com/."

[4] PPStream, "http://www.ppstream.com/."

[5] Thunder, "http://dl.xunlei.com/xl7.html."

[6] QQ, "http://xf.qq.com/."

[7] H.-C. Hsiao and C.-T. King, "Bristle: a mobile structured peer-to-peer architecture," in *Proc. of Parallel and Distributed Processing Symposium*, 2003.

[8] J. Cheng, Y. Li, L. Jiao, and J. Ma, "A new mobile peer-to-peer architecture," in *Proc. of the 5th WSEAS International Conference on Applied Computer Science,*, 2006.

[9] F.-U. Andersen, H. d Meer, I. Dedinski, C. Kappler, A. Mader, J. Oberender, and K. Tutschku, "An architecture concept for mobile p2p file sharing services," in *In Workshop at Informatik 2004 - Algorithms and Protocols for Efficient Peer-to-Peer Applications*, 2004.

[10] D. Qiu and R. Srikant, "Modeling and performance analysis of bit torrent like peer to peer networks," in *Proc. of ACM Sigcomm*, 2004.

[11] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-topeer file dissemination," in *Preprint version in arXiv*, 2006.

[12] D. Chiu, R. Yeung, J. Huang, and B. Fan, "Can network coding help in p2p networks," in *Invited paper in Second Workshop of Network Coding*, 2006.

[13] B. Fan, J. C. Lui, and D. M. Chiu, "The delicate tradeoffs in bittorrent-like file sharing protocol design," in *Proc. of ICNP*, 2006.

[14] J. Li, P. A. Chou, and C. Zhang, "Mutualcast: an efficient mechanism for content distribution in a p2p network," in *Proc. Acm Sigcomm Asia Workshop*, 2005.

[15] M. Lin, B. Fan, J. C. Lui, and D.-M. Chiu, "Stochastic analysis of file swarming systems," in *Performance Evaluation vol. 64(9-12):856-875*, 2007.

[16] L. Massoulie and M. Vojnovic, "Coupon replication systems," in *Proc. of ACM Sigmetrics*, 2005.

[17] N. Venkatasubramanian and S. Ramanathan, "Load management in distributed video servers," in *Proc. of IEEE ICDCS*, 1997.

[18] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *Proc. of IEEE Infocom*, 2007.

APPENDIX

Proof of EQ. 2 There are two possible bottlenecks. The first one is the server's upload capacity. It is straightforward that the throughput of the whole system can not exceed server's capacity. The second possible bottleneck is the upload capacity of peers and helper. We prove the maximum capacity by constructing trees to achieve the maximized throughput in different cases. Shown as follow, we discuss this problem by three cases.

Case I: $U_0 \leq \frac{\sum_{i=1}^{N} U_i}{N-1}$. We construct N spanning trees with two-hop. The capacity of the k^{th} spanning tree is $r_k = U_0 \frac{U_k}{\sum_{i=1}^{N} U_i}$. For the k^{th} spanning tree, the server sends content with rate r_k to the k^{th} peer and the k^{th} peers distribute the content to other $N-1$ peers with total rate $(N-1)r_k$. It is not difficult to verify that the throughput is U_0.

Case II: $\frac{\sum_{i=1}^{N} U_i}{N-1} < U_0 < \frac{\sum_{i=1}^{N} U_i}{N-1} + \frac{H}{N}$. We construct N two-hop spanning tree to saturate peers' upload capacity. The rate allocated to the k^{th} spanning tree is $r_k = \frac{U_i}{N-1}$. Then, we construct the $N+1^{th}$ tree with rate $U_0 - \frac{\sum_{i=1}^{N} U_i}{N-1}$ from server to helper. Since $U_0 < \frac{\sum_{i=1}^{N} U_i}{N-1} + \frac{H}{N}$, the helper has enough capacity to distribute the content to N peers. The throughput of the whole system is $U_0 - \frac{\sum_{i=1}^{N} U_i}{N-1} + \sum_{k=1}^{N} r_k = U_0$.

Case III: $U_0 \geq \frac{\sum_{i=1}^{N} U_i}{N-1} + \frac{H}{N}$. The peers' upload capacity together with helper's upload capacity is bottleneck. We construct N two-hop spanning tree to saturate peers' upload capacity. The structure of the two-hop spanning tree is the same with case I but the rate allocated to the k^{th} spanning tree is $r_k = \frac{U_i}{N-1}$. Then, we construct the $N+1^{th}$ tree with rate $\frac{H}{N}$, then, the helper distribute the content to N peers. At last, we construct a one-hop tree to utilize the left server's upload capacity. The rate is $\frac{U_0 - \frac{\sum_{i=1}^{N} U_i}{N-1}}{N} - \frac{H}{N}$. The throughput of the whole system is $\frac{U_0 - \frac{\sum_{i=1}^{N} U_i}{N-1} - \frac{H}{N}}{N} + \sum_{k=1}^{N} r_k + \frac{H}{N} = \frac{\sum_{i=0}^{N} U_i + \frac{N-1}{N} H}{N}$.

■

Proof of Lemma 1 The proof is trivial that the hit rate γ is an increasing function with the cache capacity K.

Differentiating γ over θ, with the condition $K < M$ we have:

$$\frac{d\gamma}{d\theta} = \frac{-\sum_{j=1}^{K} j^{-\theta} \sum_{i=1}^{M} i^{-\theta} lnj + \sum_{j=1}^{K} j^{-\theta} \sum_{i=1}^{M} i^{-\theta} lni}{(\sum_{j=1}^{M} j^{-\theta})^2}$$

Let $f(i,j)$ to denote the function $i^{-\theta} j^{-\theta} lni$, then the differentiation of γ over θ becomes:

$$\begin{aligned}
\frac{d\gamma}{d\theta} &= \frac{\sum_{j=1}^{K} \sum_{i=K+1}^{M} f(i,j) - \sum_{j=1}^{K} \sum_{i=K+1}^{M} f(j,i)}{(\sum_{j=1}^{M} j^{-\theta})^2} \\
&= \frac{\sum_{j=1}^{K} \sum_{i=K+1}^{M} (f(i,j) - f(j,i))}{(\sum_{j=1}^{M} j^{-\theta})^2} \\
&= \frac{\sum_{j=1}^{K} \sum_{i=K+1}^{M} i^{-\theta} j^{-\theta} (lni - lnj)}{(\sum_{j=1}^{M} j^{-\theta})^2} > 0
\end{aligned}$$

Thus, γ is an increasing function of θ. ■

Yipeng Zhou is a Postdoctoral Fellow in Institute of Network Coding of the Chinese University of Hong Kong (CUHK). received his B.S. from University of Science and Technology of China in 2006 , M.Phil. and Ph.D. degree from department of Information Engineering of CUHK in 2008 and 2012 respectively. His research interest includes modeling and analysis of P2P content distribution systems, performance evaluation, application of network coding, cloud computing and http streaming.

Tom Z.J. Fu received the B.Eng degree from Shanghai Jiao Tong University and the M.Phil degree from the Chinese University of Hong Kong in 2006 and 2008. He is now pursuing the Ph.D. from Information Engineering department of CUHK. His research interest includes P2P networking, traffic measurement and online social network.

Dah Ming Chiu received his first degree from Imperial College London, and his PhD degree from Harvard University. He is currently the department chairman of Information Engineering in the Chinese University of Hong Kong. Prior to joining CUHK, he has worked for Sun Labs, DEC, and AT&T Bell Labs. He is an IEEE Fellow, for his contribution to the resource allocation algorithms for the Internet. His current research interest includes P2P networks, network measurement, architecture and engineering, network economics, and social networks.

Yan Huang received his B.S. and Master Degree from University of Science and Technology of China. As the Director of Tencent Research, he is currently in charge of P2P and Cloud Computing technology. Before joining Tencent, he cofounded PPLive (now known as PPTV), and acted as the Chief Architect.