TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 03/12 pp29-39
Volume 17, Number 1, February 2012

Strategies of Collaboration in Multi-Swarm Peer-to-Peer Content Distribution*

Zhi Wang, Chuan Wu†, Lifeng Sun, Shiqiang Yang**

**Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China;
† Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong, China**

Abstract: In modern Peer-to-Peer (P2P) content distribution applications, multiple swarms typically exist, each corresponding to the dissemination of one content among interested peers. A common design in the existing P2P applications is to allow peers in one swarm to help each other, while different swarms are only coupled when sharing the upload bandwidth at the dedicated content servers/publishers. In recent years, a number of proposals have emerged which advocate inter-swarm collaboration and resource sharing, where peers in one swarm may contribute their storage and bandwidth resources to help peers in the swarm of another content. Such inter-swarm collaboration can improve content availability and optimize resource utilization in the entire system, at the cost of additional overhead for content preloading and inter-swarm coordination. This paper presents a survey of studies on effective inter-swarm collaboration mechanisms in the existing literature. This paper first discusses strategies of collaboration in P2P file sharing applications, and then presents multi-channel collaborative design for P2P live and Video-on-Demand (VoD) streaming. In particular, this paper elaborates our recent design of collaboration strategies among multiple streaming channels in a P2P VoD system, and shows that the server cost can be reduced by up to 25% while high streaming qualities are guaranteed in the entire system, even during extreme scenarios such as unexpected flash crowds. This paper also discusses representative approaches to implement inter-swarm collaborations in various P2P content distribution systems.

Key words: Peer-to-Peer (P2P) networks; content distribution; inter-swarm collaboration; media streaming

Introduction

Large-scale Peer-to-Peer (P2P) content distribution applications have proliferated in today's Internet, e.g., BitTorrent[1], PPLive[2], Zattoo[3], UUSee[4]. Hundreds and thousands of contents are distributed in these applications, each with a varying number of downloaders spanning from a single digit to tens of thousands. The peers downloading the same content constitute a swarm of this content. In classical design of a multi-swarm P2P content distribution application, peers in the same swarm contribute their bandwidth to upload available chunks of the content to each other; different swarms share upload bandwidth at the content servers/publishers, but typically do not share resources at individual peers, except at those which are simultaneously interested in contents in multiple swarms.

An apparent drawback of this classical design lies

Received: 2011-12-21; revised: 2011-12-30

* Supported by the National Basic Research and Development (973) Program of China (No. 2011CB302206), the National Natural Science Foundation of China (Nos. 60833009 and 60933013), and the Research Grants Council of Hong Kong (RGC GRF Ref: HKU 718710E)

** To whom correspondence should be addressed.
E-mail: yangshq@mail.tsinghua.edu.cn

at its sub-optimal utilization of resources in the entire P2P system: there are swarms with redundant upload capacities besides those used for distribution of their own contents, while there are other swarms with insufficient bandwidth to achieve a targeted download performance. In recent years, a number of studies have proposed *inter-swarm collaboration* and resource sharing, i.e., a peer in one swarm may contribute its storage and bandwidth resources to download, store, and serve contents in other swarms, even if the peer itself is not interested in viewing those contents. This collaborative design can bring two significant advantages, in terms of improved content availability and resource utilization in the entire system.

(1) Content availability. Highly skewed distribution of content popularity is common in a P2P file sharing or streaming system. Popular contents are widely replicated, and abundant upload bandwidth is typically available to serve those contents. On the other hand, unpopular contents are rarely replicated, leading to the difficulty for one peer to find enough neighbors from which to download the unpopular content[5]. If inter-swarm collaboration is enabled, a peer interested in a popular content may also download and cache some unpopular contents, and availability of unpopular contents in the system will be improved.

(2) Resource utilization. At one time, there may exist redundant upload bandwidth in some swarms (excluding that used for its own content distribution) — e.g., due to the existence of high-bandwidth peers — while some other swarms are experiencing significant deficiency of upload bandwidth. By allowing allocating redundant bandwidth from one swarm to another at different times, capacities of peers can be maximally utilized at all times, and the demand for server capacities is minimized.

On the other hand, designing and implementing an inter-swarm collaboration mechanism are challenging, in order to maximize the performance gain while minimizing the overhead of the more complicated protocol. In many inter-swarm collaboration schemes, peers need to preload a number of contents (or chunks in a content) to enable bandwidth contribution in multiple swarms. Such a preloading procedure, if not well designed, may negatively affect the download performance in the peer's original swarm. In addition, allocation of upload bandwidth at a peer participating in multiple swarms needs to be carefully carried out, in order to maximize the global resource utilization.

This paper presents a survey of studies on inter-swarm collaboration designs in the existing literature. We first introduce strategies of collaboration in P2P file sharing applications in Section 1, and then present multi-channel collaborative design for P2P live streaming and Video-on-Demand (VoD) streaming in Section 2 and Section 3, respectively. In particular, we elaborate our recent design of collaboration strategies among multiple streaming channels in a P2P VoD system, and show that the server cost can be significantly reduced while high streaming qualities are guaranteed in the entire system, even during extreme scenarios such as unexpected flash crowds. We also discuss representative approaches of implementing inter-swarm collaboration in different P2P systems in Section 4 and conclude the paper in Section 5.

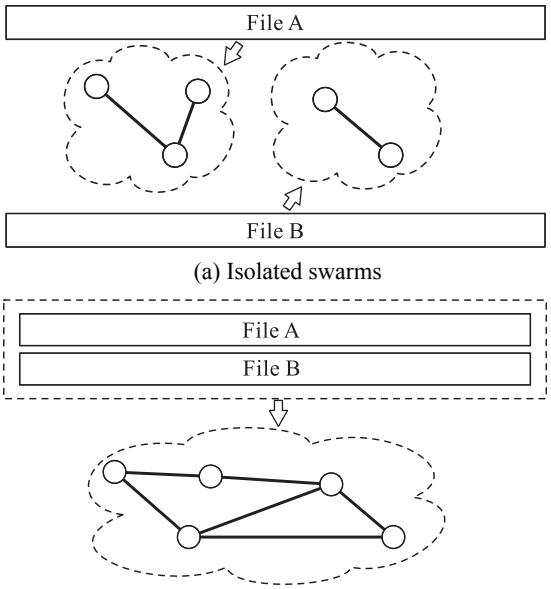
1 Inter-Swarm Collaboration in P2P File Sharing

In a P2P file sharing system, each swarm corresponds to the group of peers downloading one file. The existing studies on inter-swarm collaboration have been focusing on improving the content availability of unpopular files and reducing the file download times. We discuss schemes targeting at each of the objectives in the following.

1.1 Strategies to improve content availability

In a P2P file sharing network, a file is typically divided into many chunks for distribution, and peers in a swarm download chunks from each other until a complete copy of the file is obtained. In a popular swarm where many online peers exist, availability of different chunks is largely guaranteed. In an unpopular swarm, it is very likely that some chunks are missing from the online peers at one time, leading to prolonged time to complete the file download. Collaboration among different swarms is proposed to boost chunk availability, as illustrated in Fig. 1. Figure 1a shows a P2P file sharing system consisting of isolated swarms, which are joint to form a large collaborative swarm in Fig. 1b.

To improve the availability of unpopular files, Menasche et al.[6] propose bundling of contents for download in BitTorrent systems. Instead of disseminating



(a) Isolated swarms

(b) Collaboration among multiple swarms

Fig. 1 An illustration of inter-swarm collaboration in a P2P file sharing system

individual files in separate swarms, several files are bundled into a large file for distribution together. The idea behind such bundling is that when an unpopular file is bundled with some other popular or unpopular files, peers interested in any one of the files will download all the files in the bundled swarm; in this way, more peers cache the unpopular files, and more supplying neighbors can be found for downloading an unpopular file.

Content unavailability has been analyzed in Menasche et al.'s study[6], in both cases of bundled and non-bundled file distribution. Content unavailability is modeled as the probability that a peer finds no publisher available when it joins a swarm of a file. When no files are bundled in the BitTorrent system, the content unavailability probability is

$$P_k = \frac{1/r_k}{E[B_k] + 1/r_k} \tag{1}$$

with

$$E[B_k] = \frac{e^{r_k u_k} - 1}{r_k} \tag{2}$$

where r_k is the arrival rate of publishers of the content and u_k is the mean resident time of a publisher in the swarm. When K files are bundled for distribution, the probability becomes

$$P = \frac{1/R}{E[B] + 1/R} \tag{3}$$

with

$$E[B] = \frac{e^{RU} - 1}{R} \tag{4}$$

where R and U denote the arrival rate and mean residence time of publishers in this case, respectively. Consider the special case that the arrival rates and mean residence times of publishers are the same for all K files, i.e., $r_k = r$ and $u_k = u$. If $R = Kr$ and $U = Ku$, then

$$P = \frac{1/(Kr)}{(e^{K^2 ru} - 1)/(Kr) + 1/(Kr)} \tag{5}$$

Hence, the probability of content unavailability can be reduced by a factor of $e^{-\Theta(K^2)}$, when K files are bundled.

Given that bundling can improve the content availability in the system, the next question is how to choose the files to be bundled together. Han et al.[7] propose a systematic method for file bundling in BitTorrent systems, by clustering similar files into groups based on a few criteria, and bundling files according to the groups. For example, they infer types and features of the movies from the titles of the movie files, and bundle movies of similar types/features for distribution. The rational is that a peer interested in one file has a high probability of being interested in another similar file; by downloading them together in a bundle, the overhead of preloading files for collaborative distribution in the entire system is reduced. Their experimental observations confirm that such bundling can bring significant download performance gain.

1.2 Strategies to reduce download time

In Menasche et al.'s study[6], they define the download time T_D that a peer spends to get a file in a P2P file sharing system as the sum of two parts: (1) the wait time T_W that it spends in waiting for available publishers or neighbors holding the chunks of the file, and (2) the service time T_S that it actually spends in receiving the chunks, i.e., $T_D = T_W + T_S$. They conclude that when the service time dominates the overall download time (i.e., $T_S \gg T_W$), bundling K files together for distribution may increase the download time by up to a factor of K, as a peer now downloads K times as much content. Nevertheless, when the wait time dominates the download time (i.e., $T_W \gg T_S$), as is the case when the publisher arrival rate r in an individual swarm is low, bundling can significantly reduce the wait time by a factor of $\Theta\left(\frac{1}{Kr}\right)$, when K files are bundled for distribution.

In many real-world P2P file distribution systems, dedicated servers are utilized as backup content publishers. Therefore, content availability can be guaranteed, i.e., any chunk in a file can always be found at the dedicated servers. As a result, in these systems, the service time dominates the download time, and bundling files in a static fashion that peers have to completely download all files in the bundle, increases the total download time. To reduce the download time, dynamic bundling strategies have been proposed. In Lev-tov et al.'s design[5], a peer does not need to download a complete set B of chunks in the bundle; instead, it only downloads a subset $F \subset B$ of chunks in the bundle, which include the ones itself wants to download and some others for inter-swarm collaboration. The number of extra chunks to download may vary over time: when their own performance is guaranteed (chunks of desired files are dense) but chunks of the other files are rare, peers tend to be more "socially active", i.e., willing to download more chunks from other swarms; when their own file becomes rare or is likely to become rare in the near future, or when there is no rarity problem with the other files, peers turn to "selfish behavior", and ask only for chunks of the file they are interested in.

Carlsson et al.[8] design a *torrent inflation* strategy to utilize peer resource in some torrents to supplement insufficient upload capacity in other unpopular torrents. According to peers' download reports, the server detects some inflation files — the files whose download requests cannot be all served by peers themselves. Then the server assigns such inflation files to active peers which have extra upload capacities to share in other swarms. These active peers download some chunks in the inflation files in parallel with download of their requested files, and then upload chunks of the inflation files to the requesting peers. When many peers are requesting different chunks from the same peer, the tracker server in the BitTorrent system determines to which neighbors a peer should upload.

Finally, inter-swarm collaboration can not only help the P2P system by improving content availability and reducing download time, but also potentially benefit Internet Service Providers (ISPs). Wang and Liu[9] have investigated benefits of file bundling in BitTorrent on reducing inter-ISP traffic, by using a large collection of PlanetLab nodes[10] to interact with real-world BitTorrent trackers and peers in a three-month span. They observe that BitTorrent peers exhibit strong geographical locality, while the effectiveness of a locality mechanism, that a peer chooses the majority of its neighbors from peers within the same ISP, can be quite limited, when it is only employed in individual swarms of torrents. The reason is that in individual swarms, the number of peers in one ISP is typically small, and thus inter-ISP traffic is still unavoidable to download needed chunks. They also observe that it is prevalent in a BitTorrent system that after a peer has finished downloading and left the swarm of a torrent, it may participate in the swarm of another torrent. In this case, if upload capacity of this peer is utilized to serve peers in the previous torrent, the number of suppliers in a torrent is increased, and the possibility for peers to discover local neighbors is increased, leading to reduced inter-ISP traffic.

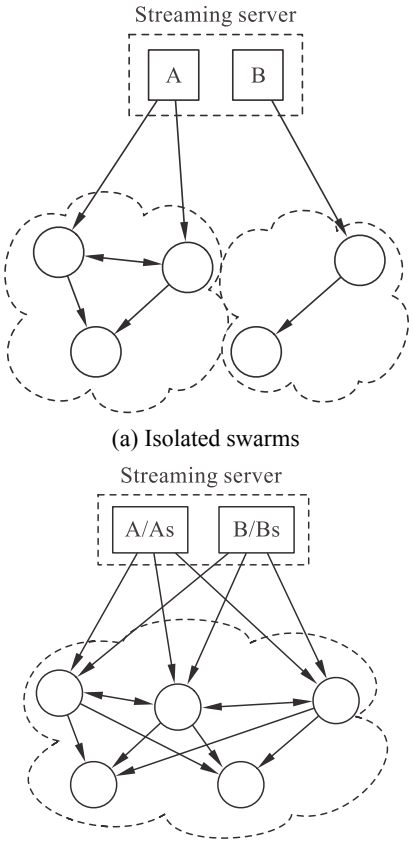
2 Inter-Swarm Collaboration in P2P Live Streaming

In a P2P live streaming system, each swarm corresponds to the group of peers streaming the same live video channel. In the traditional design of a multichannel P2P streaming system, peers in the same swarm allocate hard/soft caches to store chunks of the video stream, and retrieve available chunks from each other[2,11]. Timely chunk download is required to meet the playback deadline of chunks, according to the streaming playback rate in the channel. In a typical multi-channel P2P streaming system, popularity of different streaming channels may vary significantly. In an unpopular channel, due to the small number of concurrently online peers, it could be difficult for a peer to find enough neighboring peers to download the desired chunks from, and then the peer has to resort to the dedicated streaming server. Therefore, peers' contribution ratio (the fraction of media chunks served by peers in all the chunks retrieved) in unpopular channels could be much lower than that in popular channels[12].

In case of insufficient peer resource contribution, most existing P2P streaming systems resort to dedicated servers, which not only serve the original copy of each video stream, but also provide indispensable supplement of upload bandwidth in P2P streaming swarms[2-4,11]. Yin et al.[13] design a hybrid CDN

(Content Delivery Network)+P2P architecture for large-scale live streaming, where channels with excessive streaming requests are scheduled to be partially served by content servers close to the peers. Their design is implemented in the system Livesky, in which traditional and enhanced peers coexist — the former peers download only from neighboring peers while the latter can stream from both peers and edge CDN servers. Based on an extensive trace study of UUSee[4], another real-world P2P streaming system in China, Wu et al.[14] discovered that sometimes when no inter-swarm collaboration is in place, server upload bandwidth is essential to sustain a good streaming quality in both popular and less popular channels.

To further reduce the server cost, an effective approach is to exploit inter-swarm sharing of peer resources. Figure 2 gives an illustration. Figure 2a shows a streaming system without inter-swarm collaboration, where peers in channel A and channel B exchange media chunks in their own channels, respectively. In Fig. 2b, inter-swarm collaboration is introduced, and a peer watching channel A (or B) can help upload to peers in a different channel by preloading a sub-stream Bs (or As) for the other channel. In this way, the utilization of peer resources can be improved, as peers with extra upload capacities can be exploited by more neighbors.



(a) Isolated swarms

(b) Collaboration among multiple swarms

Fig. 2 An illustration of inter-swarm collaboration in a P2P streaming system

Targeting at a multi-channel P2P live streaming system, Wu and Li[15] advocate peer upload bandwidth contribution in multiple channels and design an optimal bandwidth allocation scheme to share peer resources across multiple swarms. Since peers in different streaming channels are competing for upload bandwidth in the system, strategies for resolving the conflicts are game theoretic in nature. The authors model dynamic bandwidth allocation in the multi-channel streaming system as dynamic auction games, and the outcome of peer strategies in the auction games provides an optimal streaming topology for all the channels, that minimizes the overall streaming cost.

Wang et al.[16] formulate linear programs to systematically model inter-swarm collaboration strategies in multi-channel P2P live streaming. The collaboration strategies are divided into three categories: (1) Naive Bandwidth allocation Approach (NBA), where a peer only contributes in its viewing channels and allocates its upload bandwidth to these channels proportional to their streaming rates; (2) Passive Channel-aware bandwidth allocation Approach (PCA), where a peer only contributes in its viewing channels and optimally allocates upload bandwidth to these channels, e.g., according to the relationship between demand and supply in these channels; and (3) Active Channel-aware bandwidth allocation Approach (ACA), where a peer contributes in not only its viewing channels, but also some other channels as a helper, and it optimally allocates upload bandwidth to these channels, according to similar criteria used in PCA. In each category, a linear program is modeled, which provides a numerical approach to explore the design space of collaboration in multi-channel streaming systems.

Wu et al.[17] propose a View-Upload Decoupling (VUD) mechanism in P2P live streaming, where the channels a peer is viewing and the channels it is uploading are decoupled. The servers are responsible to allocate peers to help in different channels. With such decoupled collaborations, stability of the supply of peer upload bandwidth in unpopular channels is guaranteed, since peer churns now have little impact on

bandwidth supply. On the other hand, what a peer downloads for viewing can be totally useless for upload. This may potentially lead to inefficient utilization of peer resources. To analyze the performance of the VUD mechanism, they develop infinite-server queueing network models[18,19], and apply them in two P2P streaming designs: the Isolated channel design (ISO) and VUD. For both of these designs, they calculate critical performance measures, and show that VUD can provide significantly better performance than ISO in heterogeneous P2P streaming systems.

3 Inter-Swarm Collaboration in P2P VoD Streaming

Inspired by its success in file distribution and live streaming applications, and in order to alleviate the server cost, the peer-to-peer paradigm has been applied to support large-scale VoD streaming in recent years[20]. Nevertheless, as compared to the P2P live streaming, the alleviation of server load is less significant in P2P VoD streaming, mainly due to the lower level of playback synchrony among VoD peers. In modern P2P VoD systems[2,4], peers' download bandwidths are commonly abundant (e.g., 1-3 Mbps ADSL connections) as compared to the representative streaming bit rates (500-800 Kbps); caches allocated at individual peers are typically as large as 2-3 GB[21]. It has been a common observation that a VoD peer's upload capacity idles as few neighbors request the chunks it currently caches, rendering a waste of peer resources[12].

The situation is further exacerbated when we consider the large number of video channels a P2P VoD system provides: The popularity of the channels is largely skewed, typically following the *Zipf* distributions, where the majority are unpopular channels with a few tens of concurrent viewers or less[22]. Peers watching the unpopular videos often need to download video chunks from the streaming servers, as few concurrent peers are caching the chunks in need; on the other hand, the upload bandwidths at peers in those unpopular channels are largely idle and wasted due to the low chance of serving the chunks they cache. Existing measurements have shown that up to 70% of video chunks may still need to be supplied from the servers in modern P2P VoD systems[20].

To utilize peers' surplus upload and download bandwidths to assist in the streaming of the whole system, a few recent proposals advocate cross-channel help among the peers. The critical questions to answer in the design of an inter-swarm collaboration scheme include: how would a peer *actively* and *dynamically* decide when it has spare capacities to assist in the streaming of other chunks/channels (that it is not watching)? Which chunks or channels should it help? How should it best utilize its capacities to achieve most effective cross-channel and intra-channel chunk upload?

Zhang et al.[23] propose to utilize the idle capacities of helpers to assist in a P2P VoD channel, where helpers are idle Internet hosts with spare storage and upload resources (which may not be peers in the streaming system). Strategies are designed to maximize the net contribution of the helpers to other peers in the system, by choosing the best number of chunks to download onto each helper and the number of helpers to use in the system.

To exploit inter-swarm collaboration inside the P2P streaming system, in our recent study[24], we propose effective strategies to maximize the utilization of peers' resources, in order to maximize the streaming qualities in all the P2P VoD channels. In our design, each peer actively and strategically determines the supply-and-demand imbalance in different channels, as well as that among different chunks within each video; then it makes use of its surplus download capacity to fetch chunks with the most need, and serves those chunks using its idle upload bandwidth, all without impairing its own streaming quality. We next elaborate our proposed strategies of selecting the channel/chunk to help at each peer, as well as strategies for dynamic helping over time.

3.1 Strategies of helping channel/chunk selection

A peer first decides which channel it will assist in and then the chunks it will fetch to serve as a supplier.

3.1.1 Selection of the helping channel

We use a channel resource vector to indicate the upload resource shortage within each channel. Let channel resource index $u_c(t)$ denote the upload capacity needed from the dedicated server to support the streaming (i.e., downloading chunks for playback) in channel c in time slot t, which is defined as:

$$u_c(t) = b_c \mid X_c(t) \mid -r_c(t) \tag{6}$$

where b_c is the bitrate of channel c, $X_c(t)$ is the set of viewing peers in the channel at t, and $r_c(t)$ represents the overall amount of upload bandwidth provided by peers to support the viewing peers in $X_c(t)$. We note that $r_c(t)$ includes the upload bandwidth from viewing peers, as well as the net contribution (upload bandwidth to serve chunks minus bandwidth needed to download those chunks) from the helper peers.

We further normalize the vector $\{u_1(t), u_2(t), \cdots, u_M(t)\}$ with M, the number of channels in the system using $\overline{u}_c(t)=\dfrac{u_c(t)}{\sum_{i=1}^{M}u_i(t)}, c=1,\cdots,M$, and derive the channel resource vector $V(t)=\{\overline{u}_1(t),\overline{u}_2(t),\cdots,\overline{u}_M(t)\}$. We use each element in $V(t)$ as the probability in our helping channel selection: channels with larger $\overline{u}_c(t)$, i.e., relatively more bandwidth insufficiency from peer contributions, are more likely to be selected by a helper peer. The helping channel selected by a peer can be a different channel from its own viewing channel (the cross-channel assistance scenario) or can be the same as its viewing channel as well (the intra-channel help scenario).

3.1.2 Selection of chunks in the helping channel

After the helping channel is selected, a peer chooses the chunks to fetch in the channel. In our design, the peer will select a starting chunk, and then download F consecutive chunks in the stream from the starting chunk on, where F is an implementation parameter in our experiments.

We use a chunk resource vector to decide the starting position. Let $g_c^s(t)$ be the ratio of chunk s in channel c downloaded directly from the server, over the total number of chunk s downloaded in time t: $g_c^s(t)=\dfrac{K_c^s(t)}{Q_c^s(t)+K_c^s(t)}$, where $Q_c^s(t)$ is the number of copies of chunk s supplied by peers in time t and $K_c^s(t)$ is the number served by the dedicated server. We consider the average ratio of chunks served by the server over all the F consecutive chunks starting from chunk s, $\mu_c^s(t)$, defined as follows:

$$\mu_c^s(t)=\frac{\sum_{k=s}^{s+F'-1}g_c^k(t)}{F'};\ F'=\begin{cases}F, & s+F-1\leqslant L_c;\\ L_c-s+1, & \text{otherwise}\end{cases} \tag{7}$$

Here L_c is the total number of chunks in channel c. We normalize the vector $\{\mu_c^1(t),\mu_c^2(t),\cdots,\mu_c^{L_c}(t)\}$ and derive the chunk resource vector $S_c(t)=\{\overline{\mu}_c^1(t), \overline{\mu}_c^2(t),\cdots,\overline{\mu}_c^{L_c}(t)\}$, where $\overline{\mu}_c^s(t)=\dfrac{\mu_c^s(t)}{\sum_{i=1}^{L_c}\mu_c^i(t)}, s=1,\cdots,L_c$. We use each element in $S_c(t)$ as the probability in our chunk selection within helping channel c: the chunks starting from s with larger $\overline{\mu}_c^s(t)$, i.e., with relatively more bandwidth demand from the server, are more likely to be selected. If the selected F chunks are already cached by the peer, it will run the channel/chunk selection again.

3.2 Strategies of dynamic helping

Peers dynamically decide when to assist in the helping channel and when to focus on the streaming of its own viewing channel, as well as how to schedule its upload of chunks to viewing and helper peers, respectively. We divide peers' requests for chunks into two types: a streaming request refers to the request for a chunk to be played by the requesting peer, and a helping request corresponds to the request for a chunk from a helper peer.

3.2.1 Switching among helping states

A peer's dynamic behavior is described using 3 helping *states*, which are decided by the amount of streaming and helping requests it has served in the previous time: (1) Non-active Helping (NH): the state when the utilization level of a peer's upload capacity is relatively low, given that the total number of chunks it has uploaded to others in the previous time slot is less than αU_p, where $\alpha\in(0,1)$ is a threshold parameter and U_p is the upload capacity of the peer. (2) Streaming Helping (SH): the state when a peer is serving more streaming requests than helping requests, such that the total number of chunks it has uploaded in the time slot is no smaller than αU_p, and the fraction of chunks uploaded for streaming requests is no lower than $\beta\in(0,1]$. (3) Fetching Helping (FH): the state when a peer serves more helping requests than streaming requests, such that the total number of chunks it has uploaded in the time slot is no smaller than αU_p, and the fraction of chunks uploaded for helping requests is no lower than $1-\beta$ (i.e., the fraction of upload to address streaming requests is lower than β).

Figure 3a illustrates the definition of the three states. Let $N(t)$ denote the total number of chunks the peer uploads in t, and $w(t)$ be the fraction of the streaming

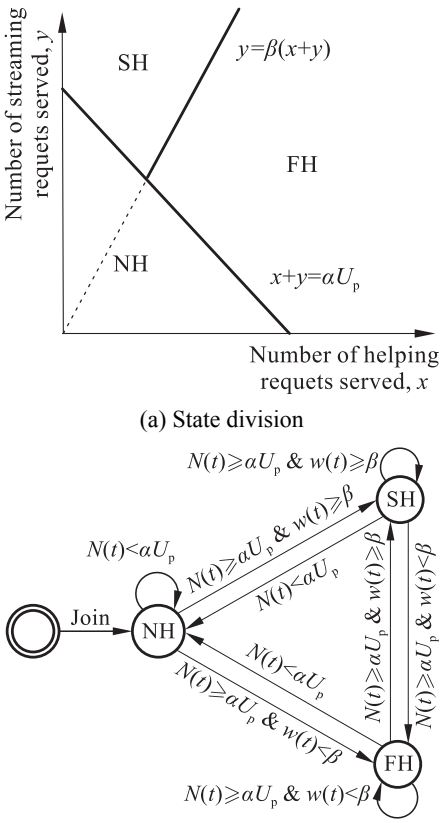


Fig. 3 Three helping states at each peer

chunks uploaded. Figure 3b further gives the transition among the three states, that occurs at the end of each time.

3.2.2 Chunk fetching from helping channels

At the beginning of a time slot, if a peer is in the SH state, it downloads only the chunks for viewing but not any chunks for helping. For a peer in the NH state, excepting downloading chunks for playback, it carries out a new helping channel and chunk selection procedure, and fetches chunks that are chosen, as long as the previous fetching process is done. For peers in the FH state, they carry out new channel selection and chunk fetching less frequently, i.e., every several time slots. In our experiments, a peer selects a new helping channel and the corresponding chunks if it has remained in the FH state for 5 time slots.

The design rationale lies in that we aim to guarantee sufficient upload bandwidth to serve the streaming requests, while maximally utilizing the surplus upload capacity to distribute chunks to helper peers. A peer in the SH state does not need to fetch more chunks from any helping channel, since the chunks it caches are already popular, as requested by many viewing peers; in this way, the upload consumption to serve this peer's helping requests can also be saved. A peer in the NH state or the FH state may still need to retrieve more chunks from the helping channels, in order to improve the utilization of its upload bandwidth. Peers in the FH state perform channel selection and chunk fetching less frequently than those in the NH state, as the upload capacities of the former are already better utilized as amplifiers for helping requests.

3.2.3 Upload schedule to serve different requests

A peer p may fetch and cache chunks from different channels, as a result of the dynamic channel and chunk selection strategies executed over time (an LRU (Least Recently Using) cache replacement strategy is applied when the peer cache becomes full). Therefore, it may be assigned by the tracker server to serve viewing and helper peers in multiple channels.

To schedule the upload of chunks at peer p, requests from the neighbors are added into a priority queue upon reception, where the priority of a request is decided as follows: (1) streaming requests have higher priorities over helping requests for chunks in all channels; (2) among the streaming requests, a request for a chunk in peer p's viewing channel is further prioritized; (3) the streaming requests in p's viewing channel and those in its helping channels, are prioritized based on deadlines of the chunk playback; (4) among the helping requests, one is prioritized if it corresponds to a chunk or channel with larger values of chunk or channel resource indices. The above priority rules are also applied to upload scheduling at the server, excepted (2).

In addition, a peer serves the requests within its upload capacity: if the number of requests received in one time is higher than its upload capacity, any unaddressed request will remain in the queue, until (1) it has stayed in the queue over 3 times (with respect to a helping request), or (2) the playback deadline of the requested chunk has been missed (for a streaming request).

We have conducted extensive evaluations of our design using real-world traces, collected from the Orbit Network[25] over a 3-month span in 2009, which is a commercial P2P VoD system in China with thousands of concurrent online users. We use the following statistics from the traces in our experiments: (1) the Zipf-like popularity distribution of 3000 channels in the system; (2) the Poisson-like arrival patterns during regular times and during a flash crowd scenario;

(3) peer session lengths, that a peer on average stays in a channel for 1/5 of the video length; (4) the number of videos a peer watches before leaving the system, which follows a Pareto distribution with range 1 to 70 and shape parameter $k=2$. The bitrates of videos in our experiments are 800 Kbps and the video durations are 900 s each. Each segment in a video has a fixed size of 16 KB. The average number of concurrent peers in our experiments is 600 during regular times, and 2500 during the flash crowd scenario. The average interval between two VCR (Video Cassette Recorder) commands issued by each peer is 5 min. Peers have heterogeneous upload (download) bandwidths which follow a Pareto distribution with range 512 Kbps to 10 Mbps (2 Mbps to 10 Mbps) and shape parameter $k=3$. We set the length of each time slot to 5 s. We compare in Fig. 4 the performance between our collaborative strategies and a native P2P VoD streaming scheme without cross- and intra- channel assistance. We observe a significant reduction of server load using our multi-channel collaborative strategies, as compared to the native one. On the other hand, our design scales with the increase of viewer numbers in the entire system, and it works well under unexpected flash crowds, as shown in Fig. 5.

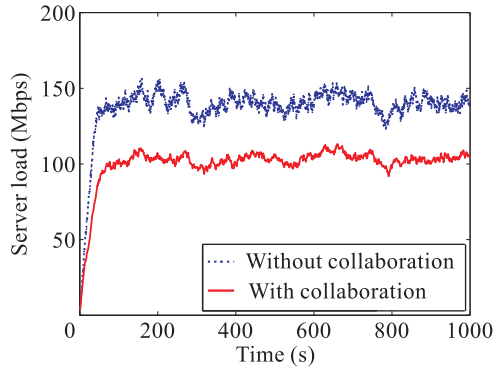


Fig. 4 Server load comparison under normal state

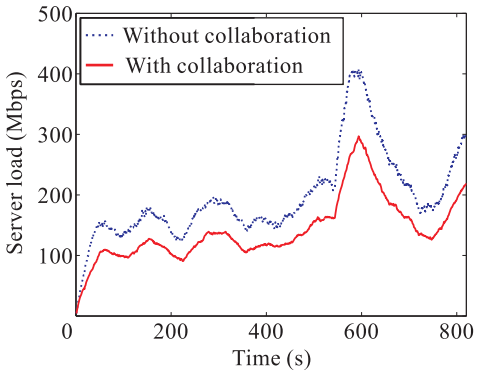


Fig. 5 Server load comparison under flash crowd

4 Implementation Discussions of Inter-Swarm Collaboration

Since inter-swarm collaborations require peers to join swarms that they are not originally supposed to, implementation of addition protocols is needed at the peer side and the server side. We survey and discuss the implementation challenges of inter-swarm collaboration strategies. Specifically, we divide the existing implementation into three categories and discuss each.

4.1 Publisher/server-based implementation

In this case, changes in the system implementation — as compared to the case without inter-swarm collaboration — mainly take place at the publishers/servers, while protocols at peer clients remain the same. In a P2P file sharing system with bundled distribution of multiple files[6,26], clients use the original protocols to discover neighbors and exchange file chunks with neighbors. The publishers of files decide which files are bundled and distributed together. To reduce the overhead of peers downloading unnecessary files, bundling is determined strategically[7]. In P2P streaming systems where inter-swarm collaboration requires global information, e.g., the ACA scheme in Ref. [16], the global information (such as the bandwidth availability in each of the channels) is maintained by a central server, in order to carry out the collaboration.

4.2 Client-based implementation

In a publisher/server-based implementation, it is difficult for publishers/servers to schedule inter-swarm collaboration at the chunk level, while typically entire file/streams are scheduled to be served by peers from other swarms. When collaboration mechanisms are implemented at the peers, more flexible utilization of peer resources can be achieved. Each peer can dynamically decide which chunks in which files it should assist in distribution at each given time. In Zhang et al.'s work[23], a helper locally decides how much portion of a file it should preload to assist in the multi-channel P2P VoD streaming, to maximize the ratio of the number of copies of chunks it can effectively upload over the number of chunks it has preloaded. In Lev-tov et al.'s design[5], a peer locally decides its policies of chunk download and upload in a BitTorrent-like system, and only contributes in other swarms

when its own download performance is guaranteed.

4.3 Hybrid implementation

Pure client-based implementation may lead to suboptimal resource utilization in the entire system, due to the lack of global information. To take advantage of both the global information at the publisher/server side and the flexibility of download/upload scheduling at the client side, hybrid implementation mechanisms have been proposed. In Wu et al.'s view-upload decoupling design of a multi-channel P2P live streaming system[27], a streaming server determines how groups of uploading peers are allocated to serve peers in each swarm in the system; meanwhile, peers optimize their upload bandwidth allocation to neighbors for streaming different sub-streams. In our design of a multi-channel P2P VoD streaming system[24], the channel resource vector, $V(t)$, is derived by the server and sent to peers for making helping channel selection decisions; each peer communicates with neighboring peers to locally calculate a chunk resource vector $S_c(t)$, and decides which chunks in this helping channel c to preload, based on the chunk resource vector.

5 Concluding Remarks

Collaboration among multiple swarms in a P2P content distribution system has been widely accepted as a promising approach to improve the utilization of peer resources. In this paper, we discuss important milestones in the literature towards bringing inter-swarm collaboration into real-world P2P systems. For P2P file sharing systems, we point out that not only the content availability but also the download time can benefit from simple collaboration strategies such as file/chunk bundling. In P2P live and on-demand streaming systems, inter-channel collaboration can also be effectively employed for reducing the load on dedicated streaming servers. In particular, we elaborate our recent design of collaboration strategies for a multi-channel P2P VoD system, and show that the server cost can be significantly reduced while high streaming qualities are guaranteed in the entire system.

Collaboration is however not free, due to the overhead of preloading/downloading chunks that a peer itself does not need, and more coordination efforts at the servers and peers. The existing research has widely employed optimization and game-theoretic models, and designed strategical algorithms to guarantee both the system-wise performance gain and individual peer's utility. As long as a good tradeoff between the performance gain and the protocol overhead can be achieved, we will continue to observe more and more real-world P2P systems to incorporate the collaborative designs in the near future.

References

[1] BitTorrent. http://www.bittorrent.com, 2011.

[2] PPLive. http://www.pplive.com, 2011

[3] Zatto. http://www.zattoo.com, 2011.

[4] UUSee. http://www.uusee.com, 2011.

[5] Lev-tov N, Carlsson N, Li Zongpeng, et al. Dynamic file-selection policies for bundling in BitTorrent-like systems. In: Proceedings of IEEE International Workshop on Quality of Service (IWQoS). Beijing, China, 2010: 1-9.

[6] Menasche D S, Rocha A A A, Li Bin, et al. Content availability and bundling in swarming systems. In: Proceedings of ACM International Conference on Emerging Networking Experiments and Technologies. Rome, Italy, 2009: 121-132.

[7] Han J, Chung T, Kim H, et al. Systematic support for content bundling in BitTorrent swarming. In: Proceedings of IEEE INFOCOM Workshops. San Diego, CA, USA, 2010: 1-2.

[8] Carlsson N, Eager D, Mahanti A. Using torrent inflation to efficiently serve the long tail in peer-assisted content delivery systems. In: Proceedings of IFIP NETWORKING. Chennai, India, 2010: 1-14.

[9] Wang Haiyang, Liu Jiangchuan. Exploring peer-to-peer locality in multiple torrent environment. *IEEE Transactions on Parallel and Distributed Systems*, 2011, **PP**(99): 1.

[10] PlanetLab. http://www.planet-lab.org, 2011.

[11] PPS. http://www.pps.com, 2011.

[12] Huang Cheng, Li Jin, Ross K W. Can internet video-on-demand be profitable? In: Proceedings of ACM SIGCOMM. Kyoto, Japan, 2007: 133-144.

[13] Yin Hao, Liu Xuening, Zhan Tongyu, et al. Design and deployment of a hybrid CDN-P2P system for live video streaming: Experiences with livesky. In: Proceedings of the 17th ACM International Conference on Multimedia. Beijing, China, 2009.

[14] Wu Chuan, Li Baochun, Zhao Shuqiao. Multi-channel live P2P streaming: Refocusing on servers. In: Proceedings of the 27th IEEE INFOCOM. Phoenix, AZ, USA, 2008:

1355-1363.

[15] Wu Chuan, Li Baochun. Strategies of conflict in coexisting streaming overlays. In: Proceedings of the 26th IEEE INFOCOM. Anchorage, USA, 2007: 481-489.

[16] Wang Miao, Xu Lisong, Ramamurthy B. Linear programming models for multi-channel P2P streaming systems. In: Proceedings of IEEE INFOCOM. San Diego, CA, USA, 2010: 1-5.

[17] Wu Di, Liang Chao, Liu Yong, et al. View-upload decoupling: A redesign of multi-channel P2P video systems. In: Proceedings of IEEE INFOCOM. Rio de Janeiro, Brazil, 2009: 2726-2730.

[18] Wu Di, Liu Yong, Ross K W. Modeling and analysis of multichannel P2P live video systems. *IEEE/ACM Transactions on Networking*, 2010, **18**(4): 1248-1260.

[19] Wu Di, Liu Yong, Ross K W. Queuing network models for multi-channel P2P live streaming systems. In: Proceedings of IEEE INFOCOM. Rio de Janeiro, Brazil, 2009: 73-81.

[20] Cheng Bin, Liu Xuezheng, Zhang Zhengyou, et al. A measurement study of a peer-to-peer video-on-demand system. In: Proceedings of International workshop on Peer-to-Peer Systems (IPTPS). Bellevue, USA, 2007.

[21] Huang Yan, Fu T Z J, Chiu D M, et al. Challenges, design and analysis of a large-scale P2P-VoD system. In: Proceedings of ACM SIGCOMM. Seattle, WA, USA, 2008: 375-388.

[22] Yu Hongliang, Zheng Dongdong, Zhao B Y, et al. Understanding user behavior in large-scale video-on-demand systems. In: Proceedings of EuroSys. Leuven, Belgium, 2006: 333-344.

[23] Zhang Hao, Wang Jiajun, Chen Minghua, et al. Scaling peer-to-peer video-on-demand systems using helpers. In: Proceedings of the 16th IEEE International Conference on Image Processing (ICIP). Cairo, Egypt, 2009: 3053-3056.

[24] Wang Zhi, Wu Chuan, Sun Lifeng, et al. Strategies of collaboration in multi-channel P2P VoD streaming. In: Proceedings of IEEE Global Telecommunications Conference (GLOBECOM). Miami, FL, USA, 2010: 1-5.

[25] Anyplex. http://www.anyplex.com, 2011.

[26] Menasche D S, Neglia G, Towsley D, et al. Strategic reasoning about bundling in swarming systems. In: Proceedings of International Conference on Game Theory for Networks (GameNets). Istanbul, Turkey, 2009: 611-620.

[27] Wu Di, Liang Chao, Liu Yong, et al. Redesigning multichannel P2P live video systems with view-upload decoupling. *Computer Networks*, 2010, **54**(12): 2007-2018.