A Performance Study of BitTorrent-like Peer-to-Peer Systems

Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang

***Abstract*— This paper presents a performance study of BitTorrent-like P2P systems by modeling, based on extensive measurements and trace analysis. Existing studies on BitTorrent systems are single-torrent based and usually assume the process of request arrivals to a torrent is Poisson-like. However, in reality, most BitTorrent peers participate in multiple torrents and file popularity changes over time.**

Our study of representative BitTorrent traffic provides insights into the evolution of single-torrent systems and several new findings regarding the limitations of BitTorrent systems: (1) Due to the exponentially decreasing peer arrival rate in a torrent, the service availability of the corresponding file becomes poor quickly, and eventually it is hard to locate and download this file. (2) Client performance in the BitTorrent-like system is unstable, and fluctuates significantly with the changes of the number of online peers. (3) Existing systems could provide unfair services to peers, where a peer with a higher downloading speed tends to download more and upload less. Motivated by the analysis and modeling results, we have further proposed a graph based model to study interactions among multiple torrents. Our model quantitatively demonstrates that inter-torrent collaboration is much more effective than stimulating seeds to serve longer for addressing the service unavailability in BitTorrent systems. An architecture for inter-torrent collaboration under an exchange based instant incentive mechanism is also discussed and evaluated by simulations.

***Index Terms*— Peer-to-Peer, Overlay Network, File Sharing, BitTorrent**

I. Introduction

BITTORRENT [10] is a new generation of peer-to-peer (P2P) file sharing system that has become very popular recently. According to BigChampagne, there are nearly 7 million BitTorrent online users at the same time in August 2004, and nearly 10 million in August 2005 [1]. According to a recent measurement by CacheLogic, BitTorrent traffic represents 53% of all P2P traffic on the Internet in June 2004 [20]. Unlike traditional P2P systems such as Gnutella [2], KaZaa [3], and eDonkey/eMule/Overnet [4], in which peers sharing *different* files are organized together and exchange their desired files with each other, BitTorrent organizes peers sharing the *same* file into a P2P network and focuses on fast and efficient replication to distribute the file. In BitTorrent, a file is divided into small chunks, and a peer can download multiple chunks of the file in parallel. Peers with different file chunks are stimulated to exchange with each other through a "tit-for-tat" incentive mechanism, which enables peers with high uploading bandwidth to have corresponding high downloading bandwidth. In this way, BitTorrent prevents *free riding* effectively, which is very common in early P2P systems [7]. In contrast, P2P systems for exchanging different files such as KaZaa and eMule use participation levels or credit systems to track the contribution of each peer, and encourage peers to contribute by giving higher service priority to those peers with more contribution. Recently, reputation systems and game theoretic approaches for providing incentive in P2P networks have also been proposed [17], [18]. However, these systems are either too complex and unrealistic or easy to cheat and are misused [6], [8]. Compared to these systems, the direct "tit-for-tat" mechanism of BitTorrent is very simple, effective, and robust. In practice, BitTorrent systems scale fairly well during flash crowd period and have been widely used for various purposes, such as for distributing large software packages [9], [16].

Research has been conducted to study the effectiveness of BitTorrent systems [16], [21], [22], [27]. The most recent work shows the stability of BitTorrent systems through a fluid model, and verifies the effectiveness of its incentive mechanism [22]. However, this fluid model assumes a Poisson model for the downloading request arrival process, which has been shown to be unrealistic in an eight-month measurement study [21]. Consequently, the model can only characterize the performance of the BitTorrent system under stable conditions. In reality, as shown by our trace analysis, this stable period is very short. Furthermore, all existing studies on BitTorrent systems focus on the behaviors of single-torrent systems only, while our trace analysis shows that most peers ($> 85\%$) participate in multiple torrents.

In this paper, we present a performance study of BitTorrent-like P2P systems by modeling, based on extensive measurements and trace analysis. We first study the evolution of a single-torrent system. We found that although the existing system is effective for addressing the "flash crowd" problem upon the debut of a new file, it has the following limitations:

- Due to the exponentially decreasing peer arrival rate and the limited up time of seeds in a torrent, the service availability of the corresponding file becomes poor quickly, and eventually it is hard to locate and download this file.
- Client performance in the BitTorrent-like system is unstable, and fluctuates significantly with the changes of the number of online peers.

Manuscript received December 15, 2005; revised July 15, 2005. This paper was presented in part at the Internet Measurement Conference, Berkeley, California, USA, October 19-21, 2005.

Lei Guo, Enhua Tan, Xioaning Ding, and Xiaodong Zhang are with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA. (e-mail: {lguo, etan, dingxn, zhang}@cse.ohio-state.edu)

Sonsgqing Chen is with the Department of Computer Science, George Mason University, Fairfax, VA 22030, USA. (e-mail: sqchen@cs.gmu.edu)

Zhen Xiao is with IBM T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA. (e-mail: xiao@research.att.com)

Digital Object Identifier 10.1109/JSAC.2007.070110.

- Existing systems could provide unfair services to peers. In current BitTorrent systems, a peer with a higher downloading speed tends to download more and upload less.

Motivated by the results of the single-torrent system study, we further propose a graph-based model to quantitatively analyze the multi-torrent system. In detail, we (1) characterize the peer request pattern in multi-torrent environments; (2) study the service potentials a torrent can provide to and get from other torrents; (3) demonstrate that inter-torrent collaboration is much more effective than stimulating seeds to stay longer for addressing the service unavailability in BitTorrent systems. Guided by the modeling results, we discuss and evaluate a novel architecture to facilitate inter-torrent collaboration with an exchange based instant incentive mechanism, addressing the well-known problem of lacking incentives to seeds.

The remainder of this paper is organized as follows. Section II presents related work. In Section III, we demonstrate the limitations of existing BitTorrent-like systems through measurements and trace analysis, and propose an evolution model for single-torrent systems. We present our multi-torrent model in Section IV. Section V discusses an architecture for inter-torrent collaboration. Finally, we make concluding remarks in Section VI.

II. Other Related Work

The amount of P2P traffic and the population of P2P users on the Internet k increasing. A lot of studies have been performed on the measurements, modeling, and algorithms of different P2P systems. Saroiu and Gummadi et al. characterized the P2P file sharing traffic over the Internet, including Napster, Gnutella, and KaZaa systems in their measurement studies [23], [24]. Gummadi and Dunn et al. analyzed the popularity distribution of P2P files over the Internet and characterized the "download at most once" property of P2P clients [14]. Measurements and traffic analysis of BitTorrent systems have also been conducted recently. Izal and Urvoy-Keller et al. analyzed a five-month workload of a single BitTorrent system for software distribution that involved thousands of peers, and assessed the performance of BitTorrent at the flash crowd period [16] . In study [9], Bellissimo et al. analyzed the BitTorrent traffic of thousands of torrents over a two-month period, with respect to file characteristics and client access characteristics. In study [21], Pouwelse et al. presented the current infrastructure of BitTorrent file sharing systems, including the Web servers/mirrors for directory service, metadata distribution, and P2P file sharing. The authors also found that the arrival, abort, and departure processes of downloaders do not follow a Poisson distribution in the eight-month trace they collected, which was assumed in the previous modeling study [22].

A queuing model for P2P file sharing systems was proposed by Ge et al. in [13]. Yang and Veciana analyzed the service capacity of BitTorrent-like systems, and found that multi-part downloading helps P2P systems to improve performance during flash crowd period [27]. Based on their study, Qiu and Srikant further characterized the overall performance of BitTorrent-like systems using a simple fluid model, and analyzed the effectiveness of BitTorrent incentive mechanism using game theory [22]. Massoulie and Vojnovic introduced a probabilistic model of coupon replication systems, and analyzed the performance under an environment where neither altruistic user behaviors nor load balancing strategies (such as rarest first in BitTorrent) are supported [19].

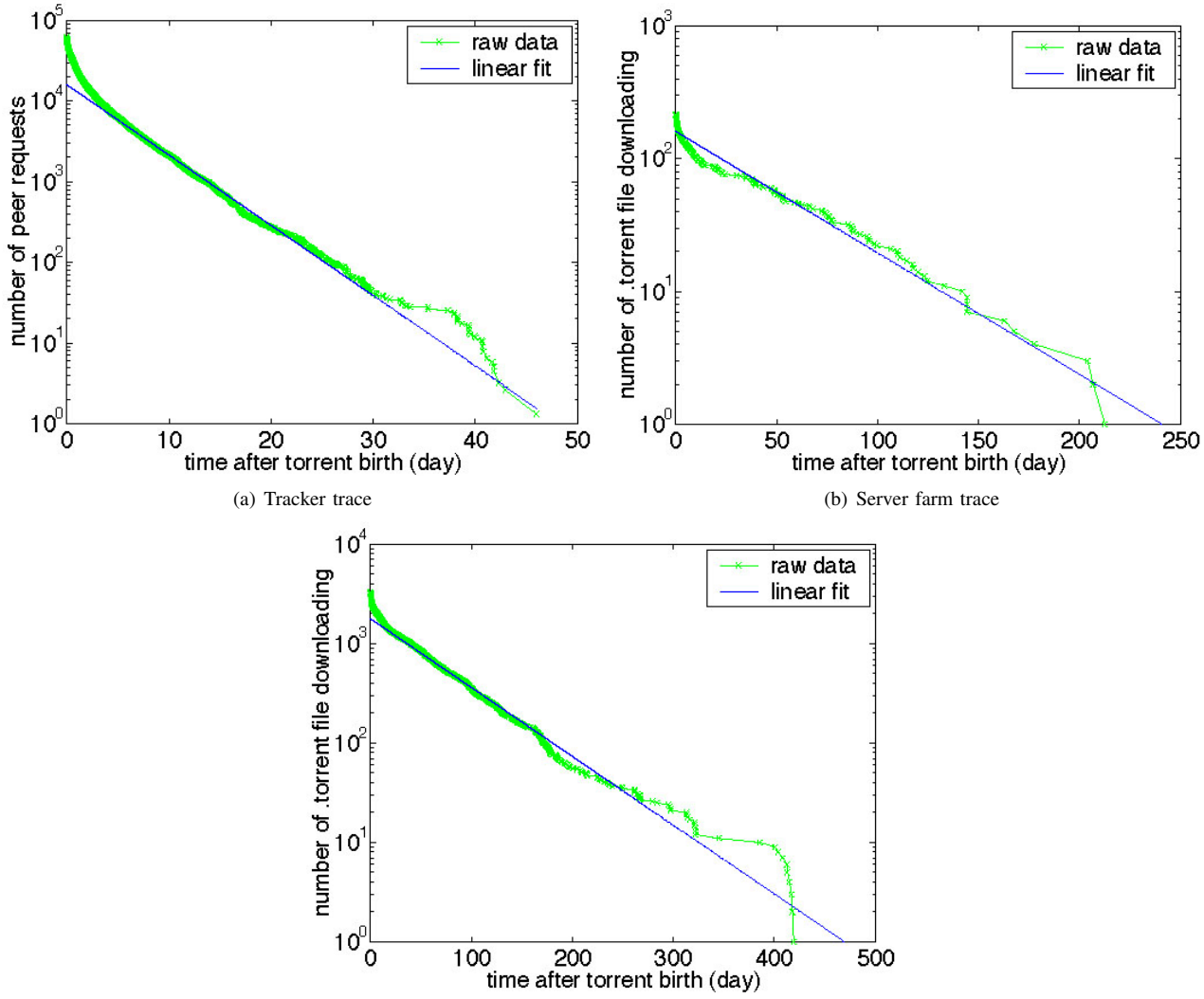
In study [26], Sripanidkulchai et al. proposed an interest-based content location approach for P2P systems. By self-organizing into small groups, peers with the same interest can collaborate more efficiently, which is similar to the BitTorrent networks, where all peers share the same file. Sherwood et al. proposed a P2P protocol for bulk data transfer, which aims to improve client performance and to reduce server load, by using enhanced algorithms over BitTorrent systems [25].

Different from all studies above, our modeling and trace analysis focus on the evolution of single-torrent systems and the inter-relation among multiple torrents over the Internet, revealing the limitations of current BitTorrent systems. Furthermore, we have proposed an innovative architecture to facilitate inter-torrent collaboration, which represents the first step towards making the BitTorrent-like system a reliable and efficient content delivery vehicle.

III. Modeling and Characterization of Single-Torrent Systems

In a BitTorrent system, the content provider creates a *meta file* (with the `.torrent` suffix name) for the *data file* it wants to share, and publishes the meta file on a Web site. Then the content provider starts a BitTorrent client with a full copy of the data file as the original *seed*. For each data file to be shared, there is a *tracker site*, whose URL is encoded in the meta file, to help peers find each other to exchange the file chunks. A user starts a BitTorrent client as a *downloader* at the beginning to download file chunks from other peers or seeds in parallel. As soon as a peer has downloaded a chunk, it is shared to the peer community so that other downloading peers have a new source of this chunk. A peer that has downloaded the file completely also becomes a seed that could in turn provide downloading service to other peers. All peers in the system, including both downloaders and seeds, self-organize into a P2P network, known as a *torrent* or a *swarm*. The initial seed can leave the torrent when there are other seeds available, and content availability and system performance in the future depend on the arrival and departure of downloaders and other seeds.

Although the effectiveness of BitTorrent systems during flash crowds, which normally happen soon upon the debut of a new file, has been widely studied through trace analysis and modeling [16], [21], [22], [27], the overall client performance in the lifetime of a torrent during which the file popularity changes has not been studied. However, the change of file popularity is particularly important for BitTorrent-like systems, where the service availability relies purely on the voluntary participation of peers. This is in contrast to a client-server model where a permanent site (i.e., a server) can provide persistent service. In this section, we propose an evolution model to study the effects of file popularity changes to the performance of a single-torrent system.



(a) Tracker trace

(b) Server farm trace

(c) Cable network trace

Fig. 1. The complementary CDF distribution of peer arrival time (time of a peer's first request to a torrent or time when a meta file was downloaded) after torrent birth for three BitTorrent traces (y-axis is in log scale)

A. Characterizing File Popularity Evolution

In this study, we analyze and model BitTorrent traffic based on two kinds of traces, one is data file downloading statistics of peers recorded by the tracker sites and the other is meta file downloading activities of BitTorrent users collected on the Internet. The BitTorrent data file downloading traces were collected from two popular dedicated tracker sites (although each torrent can have its own tracker site, there are many dedicated tracker sites on the Internet providing persistent service, each of which may host thousands of torrents), sampled every half an hour for 48 days from 2003-10-23 to 2003-12-10. This trace was collected by University of Massachusetts, Amherst [9] (abbreviated as the *tracker trace* in the remainder of this paper). We identify different peers and match multiple sessions of the same downloading with the similar methods used in study [16]. The firewalled peers, although cannot accept incoming connections and thus are not provided by the tracker to allow other peers to connect to, are still included in the tracker statistics. We extract the peer request time, downloading/uploading bytes, the downloading/uploading bandwidth of all peers of each torrent, and the information of each torrent such as torrent birth time and the size of data file. Due to page limit, we only present the analysis results of the larger tracker trace, which includes more than 1,500 torrents (about 550 torrents were fully traced during their lifecycles). The smaller trace has similar results.

The BitTorrent meta file downloading traces were collected from a large commercial server farm hosted by a major ISP and a large group of home users connected to the Internet via a well-known cable company, using the Gigascope appliance [12], from 2004-09-28 to 2004-10-07. The *server farm trace* includes about 50 tracker sites hosting hundreds of torrents, and the *cable network trace* includes about 3,000 BitTorrent users (by IP addresses) requesting thousands of torrents on the Internet. Both traces include the first IP packets of all HTTP downloading of the `.torrent` files, with the timestamp when the packet is captured (the downloading time of the `.torrent` file). This timestamp represents the peer arrival time to the torrent. We also extract the timestamp encoded in each `.torrent` file, which is the creation time of the meta file and represents the torrent birth time.

Figure 1(a) shows the complementary CDF (CCDF) distribution of the "relative" request arrival time for all fully-traced torrents in the tracker trace. We consider all requests to all torrents in the trace and normalize x and y coordinates as follows. The x coordinate is a "relative time" t, which is equal to the request arrival time to a torrent minus the birth time of this torrent, i.e., the age of the torrent when a request arrives at it. For a peer downloading the file in multiple sessions, only the first request is considered. So t denotes the arrival time of a peer to a torrent. The y coordinate at time t denotes the total number of requests to all torrents in the trace minus the cumulative number of requests to these torrents during time duration t since the requested torrent is born. The y-axis in the figure is not normalized to percentage (as normal CCDF plots) to keep the unit of y coordinates. Figures 1(b) and 1(c) show the CCDF distribution of the time when a `.torrent` file was downloaded after torrent birth in the server farm and in the cable network, respectively, similar to Figure 1(a). Note that y-axis is in log scale in the three figures.

All three curves can be fitted with straight lines. This consistent trend strongly suggests that after a torrent is born, the number of peer arrivals to the torrent decreases exponentially with time in general. The curves are not straight lines because each data set consists of many torrents, and the number of peer arrivals for different torrents may decrease exponentially with different attenuation parameters. To validate whether this claim holds for each individual torrent, we use the least square method to fit the logarithm of the complementary of the number of peer arrivals to each torrent along the time in the tracker trace. We define the *relative deviation* of this fitting at time t for a torrent as $\frac{|\log N_0(t) - \log N(t)|}{\log N_0(t)} \times 100\%$, where t is the age of the torrent when a peer arrives, $N_0(t)$ is the complementary value of the number of requests at t, and $N(t)$ is the fitting result. Figure 2 shows the distribution of average fitting deviation for each fully-traced torrent that has at least 20 peers during its lifetime. In this figure, each point in the x-axis denotes a torrent, sorted in non-ascending order of torrent population during the entire lifetime, and the corresponding value in y-axis denotes the average of relative fitting deviation of this torrent. We can see that the fitting is more accurate for torrents with larger populations, and the overall average relative deviation is only about 6%. We do not fit the curve for each individual torrent in the server farm and cable network trace, because the data collection duration is short so that they do not cover the whole lifespans of torrents. In the remainder of this paper, we only use the tracker trace for modeling and analysis.

We define the *popularity* of a BitTorrent data file at a time instant as the peer arrival rate of the corresponding torrent at that time, which is the derivative of the peer arrival time distribution of that torrent. Since the derivative of an exponential function is also an exponential function, we assume that the peer arrival rate of a torrent follows an exponential decreasing rule with time t

$$\lambda(t) = \lambda_0 e^{-\frac{t}{\tau}}, \tag{1}$$

where λ_0 is the initial arrival rate when the torrent starts, and τ is the attenuation parameter of peer arrival rate (file popularity). This equation characterizes the evolution of file popularity in a single-torrent system over time. In Section III-C, we will use a fluid model to evaluate the file popularity evolution again.

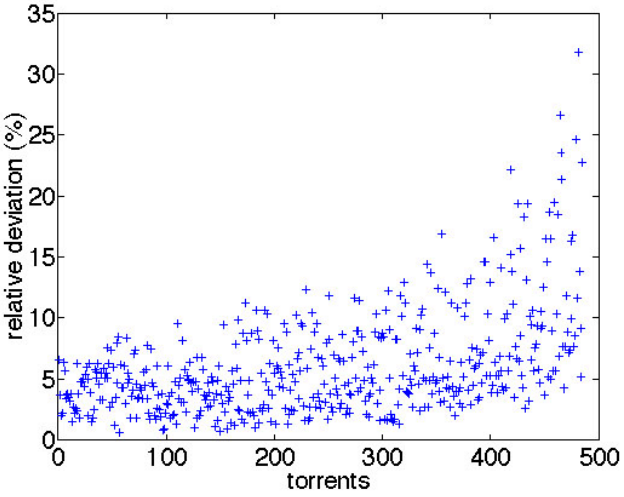


Fig. 2. Fitting deviations of fully-traced torrents in the tracker trace

B. Torrent Evolution and Service Availability

We define the *torrent lifespan* as the duration from the birth of the torrent to the time after which there is no complete copy of the file in the system, and thus new arriving peers cannot complete downloading. To simplify our model, we assume that the initial seed exits the system as soon as a downloader has downloaded the file completely. In practice, the initial seed may stay online in the system for a longer time, and some seeds may return to the system to serve the content.

The *inter-arrival time* between two successive arriving peers δt can be approximated as $\frac{1}{\lambda(t)}$. If we denote the rate at which seeds leave the system as γ, then the average service time of a seed can be approximated as $\frac{1}{\gamma}$. Since $\frac{1}{\gamma}$ is limited, according to the exponential decrease of peer arrival rate, the inter-arrival time of peers will grow exponentially, and finally there will be only one seed at a time. Thus, when $\delta t \approx \frac{1}{\lambda(t)} > \frac{1}{\gamma}$, a new peer arrives at time t cannot complete downloading before the last peer (seed) leaves, and the torrent is dead. Using Equation 1, we get the torrent lifespan

$$T_{life} = \tau \log(\frac{\lambda_0}{\gamma}). \tag{2}$$

Equation 2 shows the expectation of the real torrent lifespan. To verify Equation 2, we compute the initial peer arrival rate λ_0 and the torrent attenuation parameter τ for fully traced torrents in the tracker trace. From Equation 1, we have

$$\log \delta t = -\log \lambda_0 + \frac{t}{\tau}. \tag{3}$$

Both δt and t for each peer arrival can be extracted from the trace and we get $\log \lambda_0$ and $\frac{1}{\tau}$ using linear regression. We also compute the seed leaving rate γ as the the reciprocal of the average seed service time, which is extracted from the trace, too. Figure 3 shows the comparison of torrent lifespan computed from the tracker trace (indicated by *trace*) and that from the Equation 2 (indicated by *model*). In this figure,

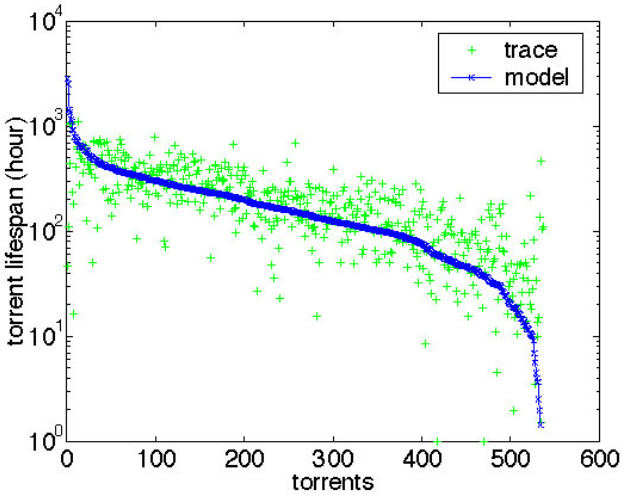


Fig. 3. The comparison of torrent lifespan: modeling and trace analysis (y-axis is in log scale)

each point in x-axis denotes a torrent, while each point in y-axis denotes the measurement result or the modeling result of torrent lifespan. The torrents in the x-axis are sorted in non-ascending order of the modeling results of torrent lifespans. As shown in the figure, our model fits the real torrent lifespan very well. The average lifespan of torrents is about 8.89 days based on the trace analysis and 8.34 days based on our model. The lifespans of most torrents are between 30 - 300 hours, and there are only a small number of torrents with extremely short or extremely long lifespans.

The *total population* of a torrent during its lifespan (in the number of peers) is

$$N_{all} = \int_0^\infty \lambda_0 e^{-\frac{t}{\tau}} dt = \lambda_0 \tau. \tag{4}$$

Among them, some peers may not be able to complete downloading due to lack of seeds, which we call *failed peers*, denoted as follows:

$$N_{fail} = \int_{T_{life}}^\infty \lambda_0 e^{-\frac{t}{\tau}} dt = \gamma \tau. \tag{5}$$

Thus, the *downloading failure ratio* of the torrent is

$$R_{fail} = \frac{N_{fail}}{N_{all}} = \frac{\gamma\tau}{\lambda_0\tau} = \frac{\gamma}{\lambda_0}. \tag{6}$$

Figure 4(a) shows the comparison of the torrent population computed from the tracker trace with that computed from our model for each fully-traced torrent. In this figure, each point in x-axis denotes a torrent, while each point in y-axis denotes the measurement result or the modeling result of the total population of this torrent during its entire lifespan. The torrents in the x-axis are sorted in non-ascending order of the modeling results of torrent populations. As evidenced by the figure, the modeling result and trace analysis are consistent. In addition, we can see that the distribution of the torrent population is heavily skewed: although there are several large torrents, most torrents are very small, and the average population of torrents is only about 102 peers.

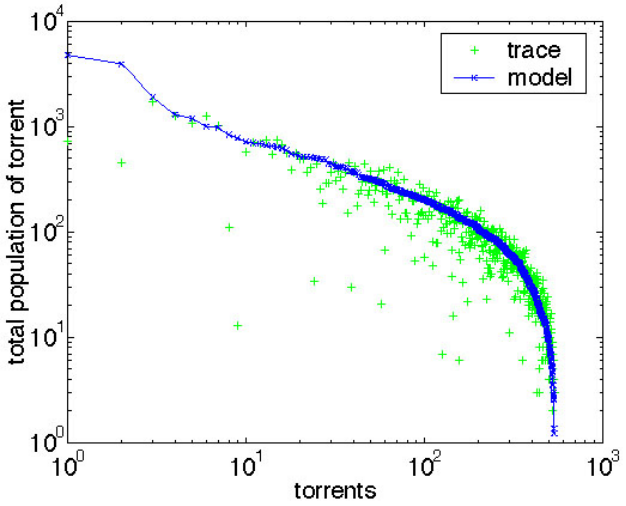
Figure 4(b) shows the downloading failure ratio based on trace analysis and on our model (plotted in a manner similar to that of Figure 4(a)). This fitting is not as good as that of Figure 4(a); the real failure ratio of torrents is lower than what our model predicts because there are some altruistic peers that serve the torrent voluntarily. That also explains why the torrent lifespan in the trace analysis (8.89 days) is slightly higher than that in our model (8.34 days). Furthermore, there are some torrents that have no failed peers in the trace because the seeds leave after the downloaders finish, but cannot be shown in the log scale plot. However, the average downloading failure ratio based on the trace analysis is still about 10%, which is non-trivial for a content distribution system.

Equation 5 implies that the number of failed peers in a torrent is independent of the initial peer arrival rate (the initial file popularity). Instead, the number of failed peers depends on the attenuation exponent of peer arrival rate (the attenuation speed of file popularity) and the seed departure rate. Figure 4(c) shows downloading failure ratios of torrents and their corresponding populations (plotted in the similar manner as that of Figure 4(a) and 4(b)). As reflected in the figure and indicated by Equation 5, the larger the torrent population, the lower the downloading failure ratio. It is interesting to note that the population of torrents, sorted in non-ascending order of their corresponding downloading failure ratios, forms several clear curves, each of which represents those torrents with similar evolution patterns (the attenuation parameter τ). On the right side of the figure, the failure ratio of the torrents is 0 due to the existence of some altruistic seeds, which always stay until the last downloader completes.

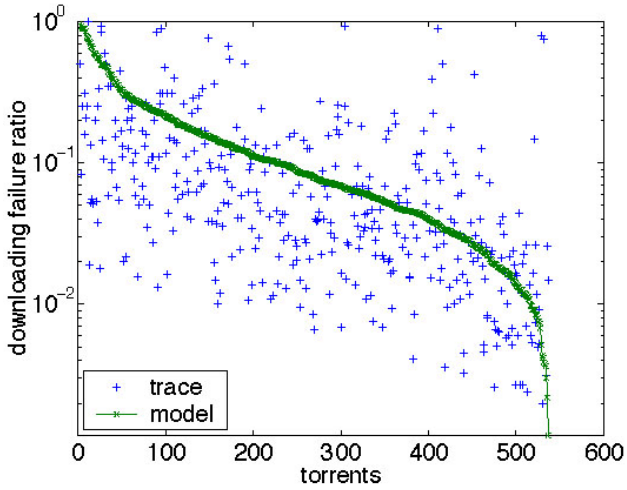
In the above analysis, we assume that peers always complete their downloading unless they cannot. We do not consider peers that abort downloading voluntarily when seeds are still available in the torrent. A peer may abort downloading due to (1) loss of interest to the data file; (2) slow downloading speed or small downloading progress. Figure 5(a) shows the distribution of the average downloading speed of peers that voluntarily abort and peers that download the data file completely. Figure 5(b) shows the distribution of downloading progress (the percentage of the entire data file that has been downloaded) when peers abort downloading voluntarily. The figures indicate that the probability for a peer to abort downloading voluntarily is almost independent of its downloading speed and the current downloading progress. This is consistent with the study [14], which found that P2P users are patient to wait days to weeks for the entire file downloading. Hence, the voluntary abort behavior of file downloadings is mainly due to the loss of user interest. Excluding peers that abort file downloading is equivalent to assuming that these peers are uninterested in the data file at the beginning, and thus does not affect our analysis.

C. Client Performance Variations

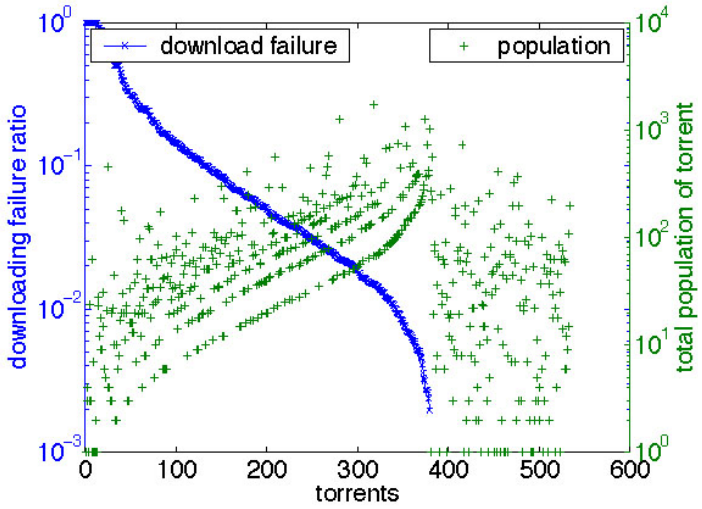
Study [22] proposed a fluid model for BitTorrent-like systems with constant peer arrival rate. We use the idea of the fluid model, but assume that peer arrival rate follows Equation 1. Assume the downloading bandwidth of a peer is greater than its uploading bandwidth, the basic ODE (ordinary



(a) Torrent population: modeling and trace analysis (in log-log scale)

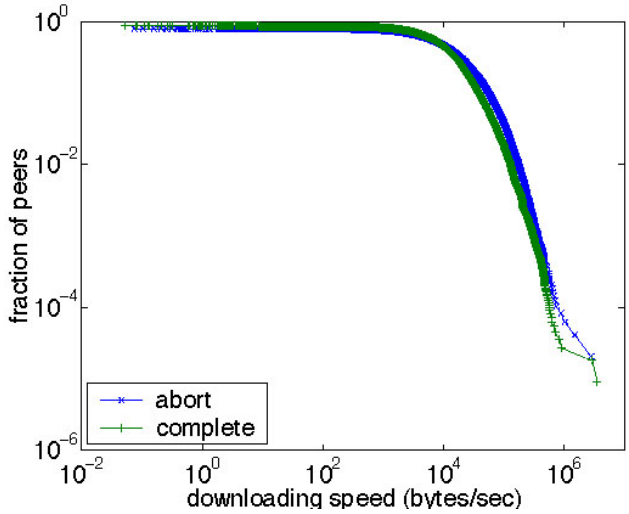


(b) Downloading failure ratio: modeling and trace analysis (y-axis is in log scale)

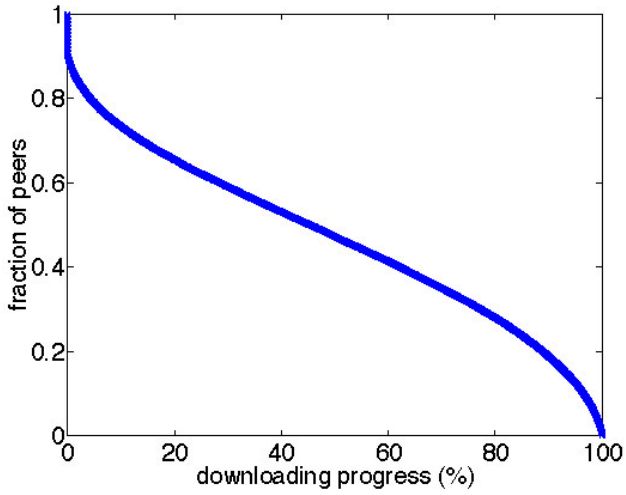


(c) The relation between torrent population and downloading failure ratio (y-axis is in log scale)

Fig. 4. Torrent population and downloading failure ratio for all fully-traced torrents

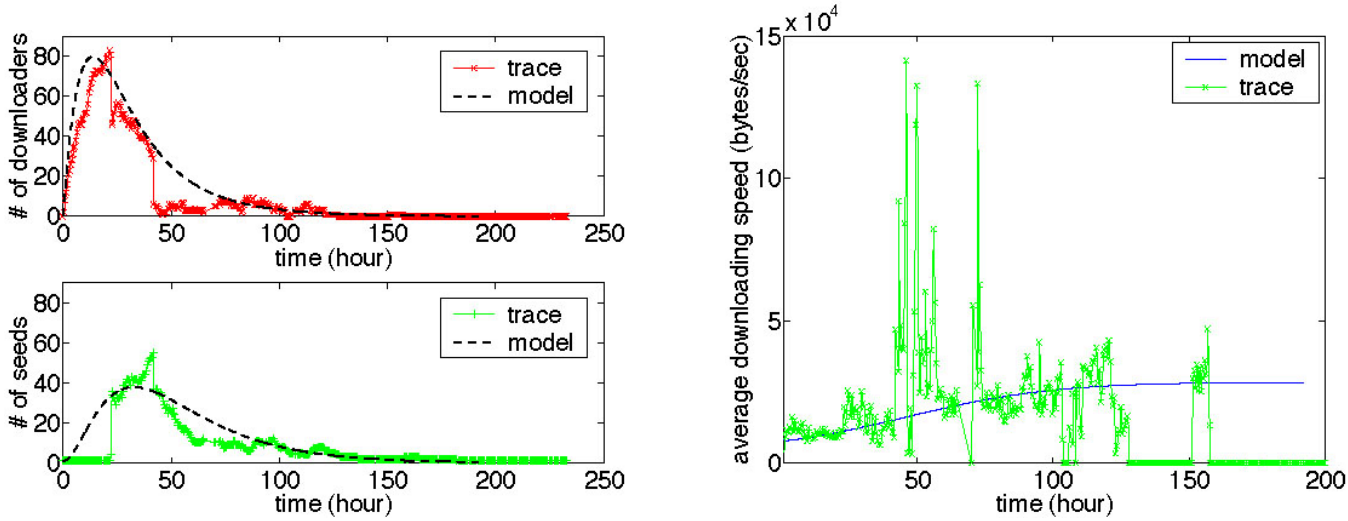


(a) The downloading speed distribution (complementary CDF, in log-log scale)



(b) The downloading progress distribution (complementary CDF)

Fig. 5. The peers abort downloading voluntarily



(a) Torrent evolution

(b) Downloading speed

Fig. 6. Torrent evolution under the fluid model

TABLE I
NOTATIONS AND ASSUMPTIONS FOR THE FLUID MODEL

$x(t)$	number of downloaders in the system at time t
$y(t)$	number of seeds in the system at time t
λ_0	the initial value of peer arrival rate
τ	the attenuation parameter of peer arrival rate
μ	the uploading bandwidth
γ	the rate at which seeds leave the system
θ	the rate at which downloaders relinquish downloading and exit the system
η	the file sharing efficiency, meaning the probability that a peer can exchange chunks with other peers

differential equation) set for the fluid model is

$$\begin{cases} \dfrac{dx(t)}{dt} = \lambda_0 e^{-\frac{t}{\tau}} - \theta x(t) - \mu(\eta x(t) + y(t)), \\ \dfrac{dy(t)}{dt} = \mu(\eta x(t) + y(t)) - \gamma y(t), \\ x(0) = 0, y(0) = 1, \end{cases} \tag{7}$$

where the meanings of the parameters in our fluid model are listed in Table I. These notations are adopted from work [22], [27].

When the ODE set has two different real eigenvalues $\psi_1 \neq \psi_2$, the resolution can be expressed as:

$$\begin{cases} x(t) = ae^{\psi_1 t} + be^{\psi_2 t} + d_1 e^{-\frac{t}{\tau}}, \\ y(t) = c_1 ae^{\psi_1 t} + c_2 be^{\psi_2 t} + d_2 e^{-\frac{t}{\tau}}, \end{cases} \tag{8}$$

where d_1, d_2, c_1, c_2, a, b are constant. The value of these constants and the detailed resolution of the fluid model can be found in Appendix A.

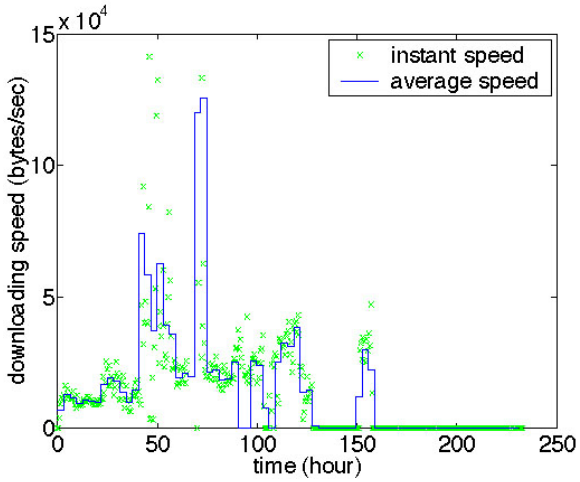
The average downloading speed of peers at time t is

$$u(t) = \mu \frac{\eta x(t) + y(t)}{x(t)} = \mu(\eta + \frac{y(t)}{x(t)}). \tag{9}$$

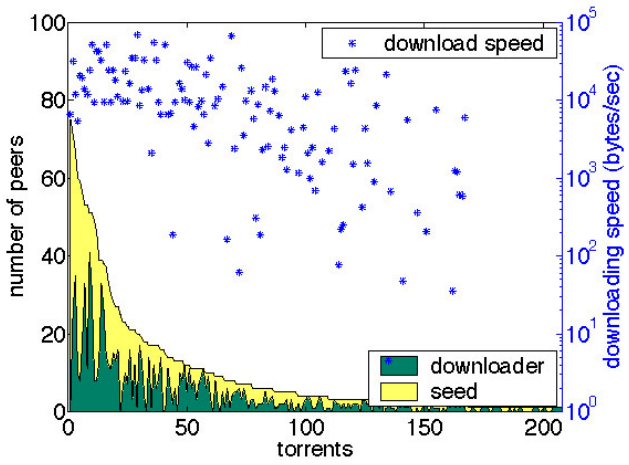
We use the tracker trace to validate the torrent evolution model. Similar to the peer arrival rate, the modeling results fit the trace better for torrents with larger populations. Figure 6(a) shows the torrent evolution by both our fluid model and the analysis results of a typical torrent in the trace. The figure shows that the number of downloaders increases exponentially in a short period of time after the torrent's birth (the flash crowd period), and then decreases exponentially, but at a slower rate. The number of seeds also increases exponentially at first, and then decreases exponentially at a slower rate. The peak time of the number of seeds lags behind that of the number of downloaders. As a result, $u(t)$ increases until the torrent is dead, and the resources of seeds cannot increase in proportion to service demand. Furthermore, due to the random arrival of downloaders and the random departure of seeds, average downloading performance fluctuates significantly when the number of peers in the torrent is small, as shown in Figure 6(b).

Figure 7(a) shows the performance variations of the torrent under two kinds of granularities. The *instant speed* represents the mean downloading speed of all peers in the torrent at that time instant, sampled every half an hour. The *average speed* represents the average value of the instant speed over the typical downloading time (the average downloading time of all peers). The figure shows that the client downloading speed at different time stages is highly diverse and can affect client downloading time significantly. The reason is that seeds play an important role in the client downloading performance. However, the generation of seeds is the same as the completeness of peer downloading, so the random fluctuation of downloading speed cannot be smoothed in the scale of typical downloading time when the number of peers is small.

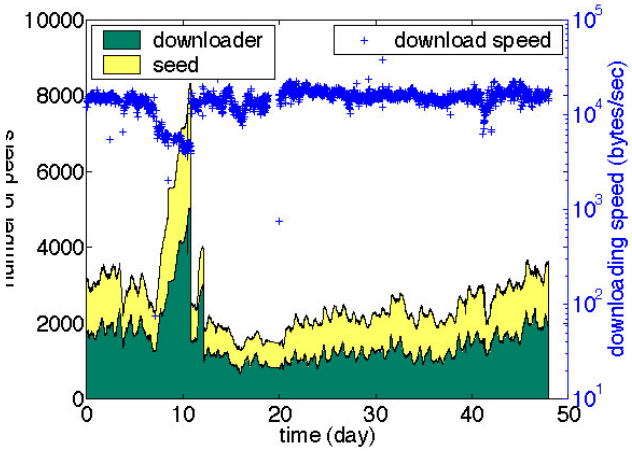
Figure 7(b) shows the number of peers and the average downloading speed for each torrent in the trace at 12:00:01 on 2003-11-15. In this figure, each point in x-axis denotes a torrent, while the left y-axis denotes the number of peers (the number of downloaders and seeds are represented with different colors and stacked together in the figure) in this torrent, and the right y-axis denotes average downloading speed of this torrent. The torrents in the x-axis are sorted in non-ascending order of the number of peers (downloaders and



(a) The downloading speed in the lifetime of a typical torrent



(b) The downloading speed (in log scale) and the number of downloaders/seeds for each torrent at a time instant



(c) The average downloading speed (in log scale) and the total number of downloaders/seeds for all torrents in the system

Fig. 7. Performance variations in BitTorrent systems

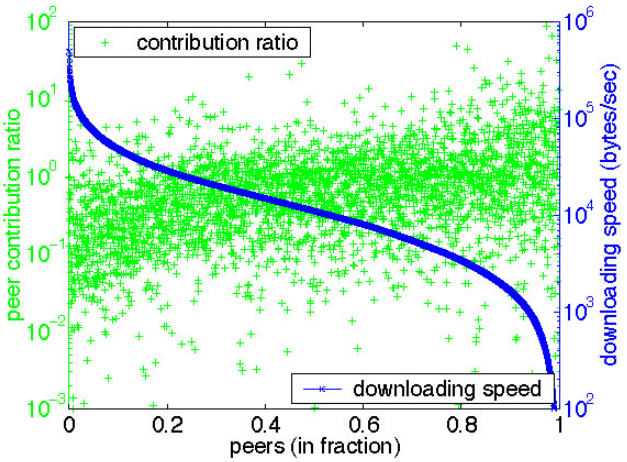
seeds) in each torrents. The results at other time instants are similar. In general, peers in torrents with larger populations have relatively higher and more stable downloading speed, while the downloading speed in torrents with small populations disperses significantly. When the number of peers in the torrent is small, the client downloading performance is easily affected by the individual behavior of seeds.

Figure 7(c) shows the total number of peers in all torrents (the number of downloaders and seeds are represented with different colors and stacked together in the figure) and the average downloading speed of all downloaders in the trace at different time stages. The average downloading speed of all torrents is shown to be much more stable than that of one torrent. The reason is that the downloader/seed ratio is much more stable due to the large population of the system. This motivates us to balance the service load among different torrents, so that each torrent can provide relatively stable downloading performance to clients in its lifespan.

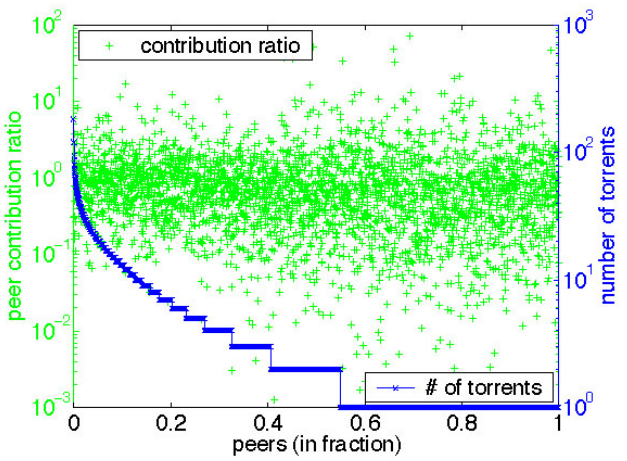
D. Service Fairness

In a BitTorrent system, the service policy of seeds favors peers with high downloading speed, in order to improve the seed production rate in the system, i.e., to have these high speed downloaders complete downloading as soon as possible and *wish* they will then serve other downloaders. In this subsection, we investigate whether this wish comes true in practice.

We define the *contribution ratio* of a peer as the total uploaded bytes over the total downloaded bytes of the peer. Figure 8(a) shows the peer downloading speed and the corresponding contribution ratio extracted from the trace. In this figure, each point in the x-axis denotes a peer, while the left y-axis denotes the contribution ratio of this peer, and the right y-axis denotes the average downloading speed of this peer. On the x-axis, peers are sorted in non-ascending order of their contribution ratios. If the service of BitTorrent system is fair, the peer contribution ratio and downloading speed should be highly positively correlated. However, the figure shows a rough trend that the peer contribution ratio increases when the downloading speed decreases. That is, the higher the downloading performance peers have, the less uploading service they actually contribute. This indicates that peers with high speed finish downloading quickly and then quit the system soon, which defeats the design purpose of the



(a) The peer downloading speed and contribution ratio



(b) The the number of torrents that each peer involves and the corresponding contribution ratio

Fig. 8. Fairness of seed service policy in BitTorrent systems (y-axis is in log scale)

seed service policy.

Figure 8(b) shows the number of torrents that each peer involves and its corresponding contribution ratio (plotted in the similar manner as that of Figure 8(a)). The figure shows no distinguishable correlation between the two, indicating that the main reason for seeds to leave old torrents is not to start new downloading tasks.

In summary, we observe that the BitTorrent's biased seed service policy in favor of high speed downloaders really affects the fairness to peers in downloading, and an incentive mechanism is needed to encourage seeds to contribute.

IV. Modeling Multiple Torrents in BitTorrent Systems

In the previous section, we have shown that client performance fluctuates significantly in single-torrent systems, but is very stable when aggregated over multiple torrents. Based on this observation, in this section, we study the correlation among multiple torrents through modeling and trace analysis, aiming to look for solutions to enable inter-torrent collaboration.

Although different torrents are independent from each other in the current BitTorrent systems, they are inherently related by peers that request multiple data files. A peer may download a data file, serve as a seed for that torrent for a while, and then go offline to *sleep* for a period of time. The peer may return sometime later and repeat the activities above. Thus, a peer's lifecyle consists of a sequence of *downloading*, *seeding*, and *sleeping* activities. If a peer stops using BitTorrent for a long time that is much longer than its typical sleeping time, we consider the peer as *dead*.

In the current BitTorrent systems, a peer is encouraged to exchange file chunks with other peers that are downloading the same file instead of serving old data files it has downloaded. Thus, in our model, we assume each peer joins (downloading and seeding) each torrent at most once, and joins one torrent at a time. Having these assumptions, we start to characterize peers in multiple torrents.

A. Characterizing Peer Request Pattern

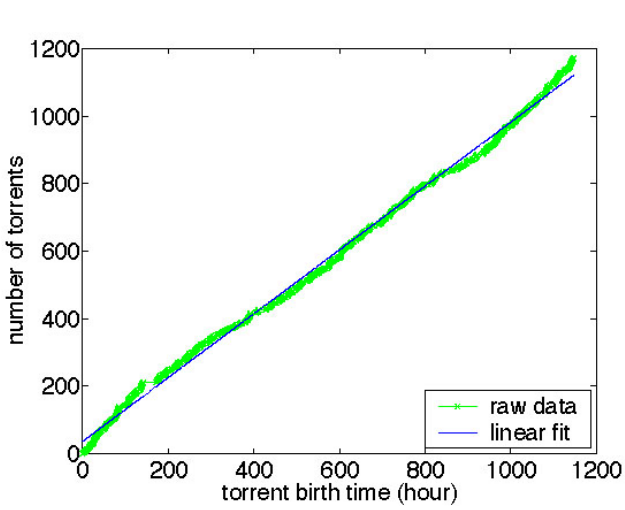
In the multi-torrent environment, both torrents and peers are born and die continuously. Figure 9(a) shows the CDF of torrent birth in the trace (indicated by *raw data*) and our linear fit. The average *torrent birth rate* (denoted as λ_t in the following context) is about 0.9454 torrent per hour. Figure 9(b) shows the CDF of torrent request arrivals (for all peers over all torrents) and our linear fit. We define the *torrent request rate* as the number of downloading requests for all torrents per unit time in the multi-torrent system, denoted as λ_q in the following context. Although the peer arrival rate of a single-torrent system decreases exponentially as shown in Figure 1, the torrent request rate in the multi-torrent system is almost a constant, about 133.39 requests per hour.

Since both the torrent birth rate and torrent request rate are almost constant, it is natural to assume that the *peer birth rate* (denoted as λ_p in the following context) is also a constant. A peer is *born* when it appears in the system for the first time. However, as shown in Figure 9(c), the peer birth rate is high at the beginning of the trace collection duration, and then converges to a constant rate asymptotically. The reason is that peers appear in the trace for the first time may actually have been born before the trace collection, and the number of such peers decreases quickly after the trace collection starts. Thus, we take the asymptotic birth rate as the real birth rate of peers, which is about 19.37 peers per hour.

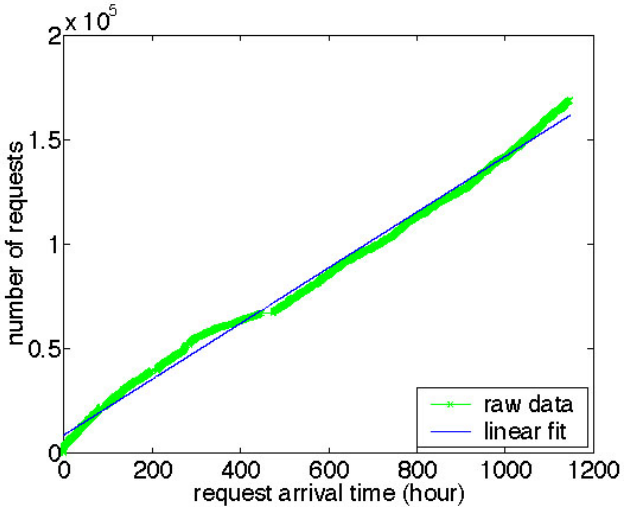
The constant peer birth rate and torrent request rate indicate that each peer only joins a limited number of torrents. However, the request rate of a peer might still change over time. We define the *peer request rate* as the number of requests a peer submits for different torrents per unit time. Assume the peer request rate can be expressed as

$$r(t) = r_0 e^{-\frac{t}{\tau_r}}, \tag{10}$$

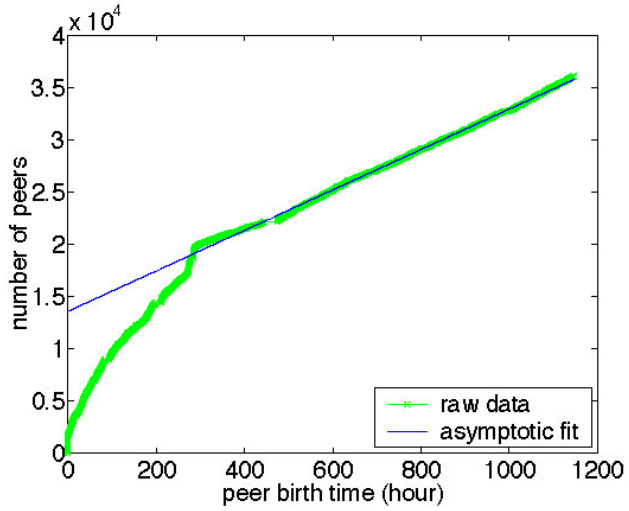
where t is the time duration after the peer is born, r_0 is the initial request rate, and τ_r is the attenuation parameter of the request rate. When $\tau_r \to \infty$, the peer has a constant request rate; when $\tau_r < 0$, the peer has an increasing request rate.



(a) Torrent birth



(b) Request arrivals of all peers over all torrents



(c) Peer birth

Fig. 9. The CDFs of torrent birth, peer request arrival, and peer birth over the trace collection time

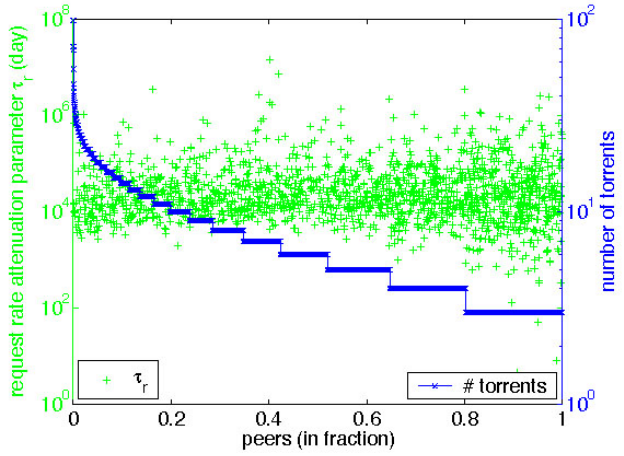
The inter-arrival time between two successive requests of a peer δt is $\frac{1}{r(t)}$ (note this inter-arrival time is different from the inter-arrival time of two successive arriving peers to a torrent described in Section III-B). Thus, we have

$$\log \delta t = -\log r_0 + \frac{t}{\tau_r}. \tag{11}$$

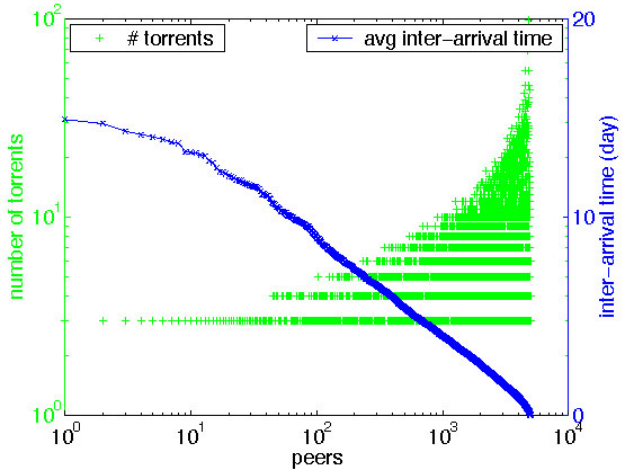
We extract δt and t from the trace for each peer requesting multiple torrents, and use linear regression to compute $\log r_0$ and $\frac{1}{\tau_r}$. Figure 10(a) shows the number of torrents that each peer requests and the corresponding τ_r, for peers requesting at least 3 torrents. In this figure, each point in the x-axis denotes a peer, while the left y-axis denotes the τ_r value of this peer, and the right y-axis denotes the number of torrents in which this peer participates. In x-axis, peers are sorted in non-ascending order of the number of torrents they join. As shown in the figure, the value of parameter τ_r in Equation 10 is extremely large compared to the typical duration of file downloading, with the mean value of about 77 years, which implies that the average request rates of peers do not change significantly over time. Further, τ_r is independent of the number of torrents that peers join. Thus, we can assume that the request processes of peers are Poisson-like with constant average request rates.

Figure 10(b) shows the average inter-arrival time of torrent requests for peers requesting multiple torrents (plotted in the similar manner as that of Figure 10(a)). As shown in the figure, it is intuitive to find that the upper bound of the number of torrents each peer requests increases with the decrease of inter-arrival time. However, for peers with similar request rates, the number of torrents they request are very diverse, since they stay in the system for different time durations. Figure 10(c) further plots the downloading speed versus the number of torrents that each peer joins (plotted in the similar manner as that of Figure 10(a)). There is no strong correlation between the two for peers with downloading speed > 1 KB per second. This implies that for peers whose downloading speed is large enough, the numbers of files they download is independent of their downloading speed.

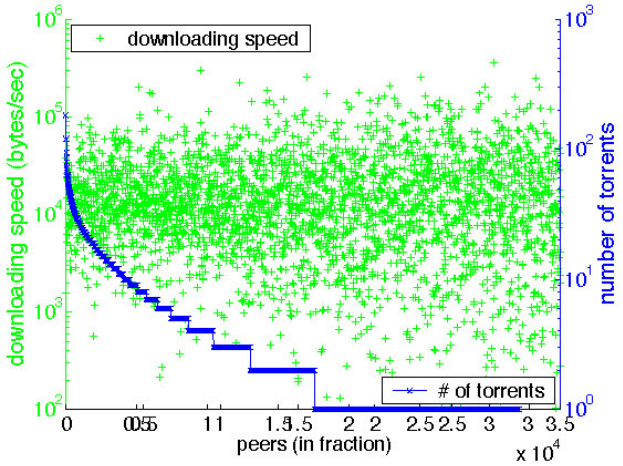
Thus, we assume that a peer joins a new torrent with probability p. For N peers in the system, during their whole lifecycles, there are Np^{m-1} peers that request at least m



(a) The attenuation of peers' requesting rates and number of torrents peers request (y-axis is in log scale)



(b) The inter-arrival time of peers' requests and number of torrents they join (in log-log scale)



(c) The downloading speed and number of torrents peers join (y-axis is in log scale)

Fig. 10. The request pattern of peers

torrents. Ranking peers in non-ascending order of the number of torrents they join, the number of torrents that a peer ranked i joins is

$$m = 1 + \frac{\log i - \log N}{\log p}. \tag{12}$$

In addition, a peer has the probability $1-p$ to download exactly 1 file, probability $p(1-p)$ to download exactly 2 files, and probability $p^{k-1}(1-p)$ to download exactly k files. So the mean number of torrents that a peer joins is:

$$\bar{m} = \sum_{k=1}^{\infty} kp^{k-1}(1-p) = \frac{1}{1-p}. \tag{13}$$

Figure 11(a) shows the distribution of the number of files that each peer downloads in the trace. The curve in the figure is a little convex, deviating from what Equation 12 predicts (a straight line when x-axis is in log scale). The reason is that the number of torrents joined by peers born before the trace collection is under-estimated, since some of these requests cannot be recorded in the trace. A similar situation exists for peers that are still active before the end of trace collection.

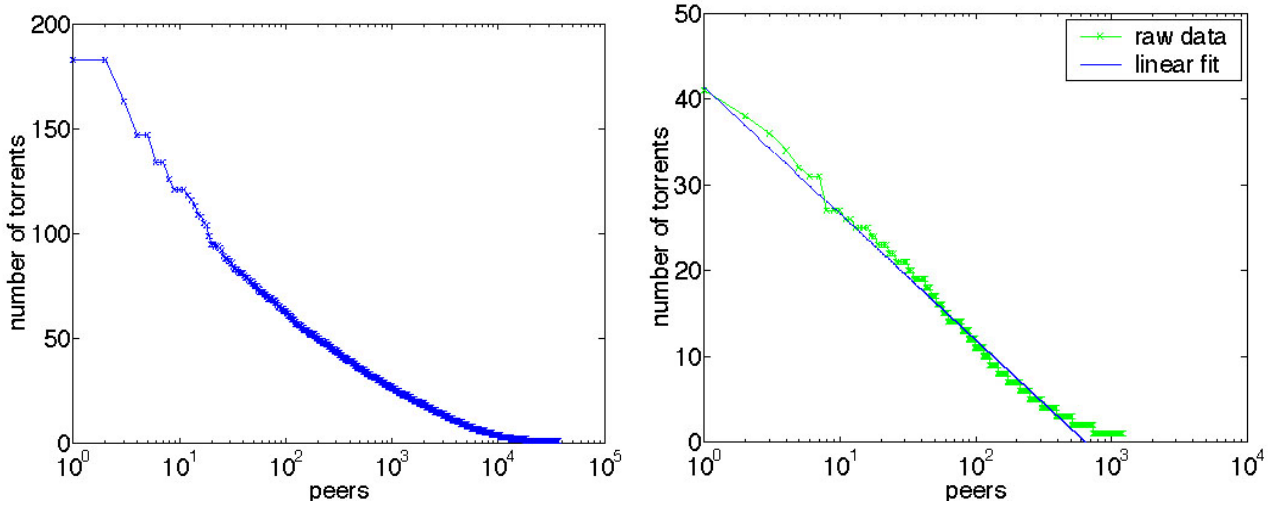
Figure 11(b) shows the distribution of number of torrents joined by each peer that was born in the middle of the trace collection duration (indicated by *raw data*) and our linear fit. The curve fits Equation 12 very well, and we estimate $p \approx 0.8551$ from the analysis, while the average number of torrents each peer joins is about 7.514.

To verify the probability model we use in the above analysis, we estimate p in another way as follows. Assuming that the peer birth rate is λ_p and the torrent request rate is λ_q, since each peer joins $\frac{1}{1-p}$ torrents during its lifetime in average, we have

$$\lambda_q = \frac{1}{1-p}\lambda_p. \tag{14}$$

Based on the peer request arrival rate and the peer birth rate we derived before (see Figure 9(b) and 9(c)), we have $p = 1 - \frac{\lambda_p}{\lambda_q} = 0.8548$. This is very close to the value we got from Equation 12, 0.8551, meaning that there are more than 85% peers joining multiple torrents.

Having characterized the torrent request pattern of peers, finally we consider the distribution of the seeding time and the sleeping time of peers. According to our fluid model, $\frac{1}{\gamma}$ represents the average seeding time. Figure 12(a) and 12(b) show the probability distribution functions of the peer seeding time and the peer sleeping time in the system. Note



(a) For all peers in the trace

(b) For peers born in the middle of trace collection time

Fig. 11. Torrent involvement of peers (x-axis is in log scale)

that the y-axis is in log scale. Both the peer seeding time and sleeping time roughly follow the exponential distribution with probability density function $f_{sd}(t) = \frac{1}{\tau_{sd}} e^{-\frac{t}{\tau_{sd}}}$, and $f_{sl}(t) = \frac{1}{\tau_{sl}} e^{-\frac{t}{\tau_{sl}}}$, respectively. Based on the trace analysis, we estimate $\tau_{sd} = \frac{1}{\gamma} = 8.42$ hours, and $\tau_{sl} = 58.32$ hours.

B. Characterizing Inter-torrent Relations

In this subsection we study how different torrents are connected through peers that download multiple files, based on our previously verified assumptions.

For simplification, we consider a homogeneous multi-torrent environment where all torrents and peers have the same λ_0, τ, μ, η, γ, and average sleeping time. Consider all torrents that have been born in the system by the time instant t_0. We number the torrents and name their birth time as follows: the most recently born torrent by t_0 is torrent 1, with a birth time t_1; the torrent born just before torrent 1 is torrent 2, with a birth time t_2; ... ; and so on and so forth. Thus, for any two torrents i and j, if torrent i was born just before torrent j, we have $i = j + 1$ and $t_i < t_j$.

Assume the probability that a peer selects torrent i at time t as its k-th torrent is $P_i^k(t)$, $t \le t_0$ and $1 \le i < \infty$. We have $P_i^k(t) = 0$ if $t < t_i$. We denote $P_i^1(t)$ as $P_i(t)$ for simplicity, and assume that $P_i(t)$ satisfies

$$P_i(t) = \frac{e^{-\frac{t-t_i}{\tau}}}{\sum_{j=1}^{\infty} e^{-\frac{t-t_j}{\tau}}}, \tag{15}$$

where $t_j = t - \frac{j}{\lambda_t}$, $1 \le j < \infty$. Thus, we have

$$\begin{aligned} P_i(t) = \frac{e^{-\frac{i}{\lambda_t \tau}}}{\sum_{j=1}^{\infty} e^{-\frac{j}{\lambda_t \tau}}} &= (e^{\frac{1}{\lambda_t \tau}} - 1) e^{-\frac{i}{\lambda_t \tau}} \\ &= (e^{\frac{1}{\lambda_t \tau}} - 1) e^{-\frac{t-t_i}{\tau}}. \end{aligned} \tag{16}$$

When a peer requests its k-th data file, the data files that it has requested will not be selected. Assuming

$$P_i^k(t) = \alpha_k P_i(t), \tag{17}$$

the peer arrival rate of torrent i can be expressed as

$$\begin{aligned} \lambda_i(t) &= \alpha \lambda_q P_i(t) \\ &= \frac{\alpha}{1-p} \lambda_p (e^{\frac{1}{\lambda_t \tau}} - 1) e^{-\frac{t-t_i}{\tau}}, \end{aligned} \tag{18}$$

where $\alpha = \sum_{k=1}^{\infty} \alpha_k p^{k-1}(1-p)$. When $\lambda_t \gg r$, we have $\alpha_k \approx 1$ and $\alpha \approx 1$. Comparing Equation 1 with 18, we have $\lambda_0 = \frac{\alpha}{1-p} \lambda_p (e^{\frac{1}{\lambda_t \tau}} - 1)$.

Considering that a peer in a torrent may have downloaded files from other torrents, we can model the relationship among different torrents in the P2P system as a *directed graph*. Each node in the graph represents a torrent. A directed edge from torrent i to torrent j denotes that some peers in torrent i have downloaded the file from torrent j, and thus have the potential to provide service to peers in torrent j, even though they are not in torrent j currently. The weight of the directed edge $W_{i,j}$ represents the number of such peers. For simplicity, we define $W_{i,i} = 0$.

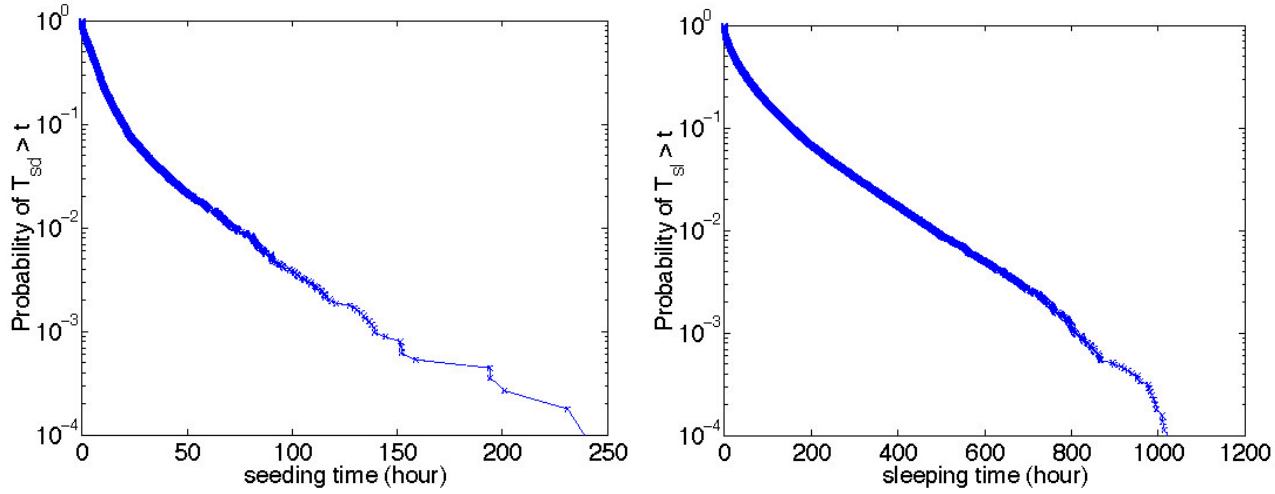
The graph changes dynamically over time. Now let us consider the graph at time t_0. During time $[t, t + dt]$, $t_j \le t < t_0$, there are $\lambda_j(t)dt$ peers who joined torrent j. Let $k(t) = \lfloor r(t_0 - t) \rfloor$. During time $[t, t_0]$, these peers can download up to $k(t) - 1$ torrents completely in addition to torrent j and may request (or be requesting) the next torrent at time t_0. Assuming $\alpha_k \approx 1$, for a peer who is active at time t, the probability that it is still active at time t_0, but does not request torrent i during $[t, t_0]$ is

$$Q_i(t) = p \times \prod_{l=1}^{k(t)-1} p \times (1 - P_i(t + \frac{l}{r})). \tag{19}$$

When $i \neq j$, we have

$$W_{i,j} = \int_{t_j}^{t_0} Q_i(t) \times P_i(t + \frac{k(t)}{r}) \times \lambda_j(t) dt. \tag{20}$$

Therefore, the weighted out-degree of torrent i represents the total potential capability its peers can provide to peers in



(a) The probability distribution of seeding time

(b) The probability distribution of sleeping time

Fig. 12. The seeding time and sleeping time of peers (y-axis is in log scale)

other torrents, denoted as SP_i, where

$$SP_i = \sum_{j=1}^{\infty} W_{i,j}. \tag{21}$$

Correspondingly, the weighted in-degree of torrent i represents the total potentials its peers can get from peers in other torrents, denoted as SG_j, where

$$SG_j = \sum_{i=1}^{\infty} W_{i,j}. \tag{22}$$

multorr-degree-allu multorr-degree-crap
ipid-spec-allu ipid-spec-crap

Figure 13(a) and 13(b) show the weighted out-degree and weighted in-degree at a time instant based on trace analysis and our probability model, respectively. In the figures, each point in the x-axis denotes a torrent, sorted in non-ascending order of weighted out-degree or weighted in-degree. The right y-axis in the figures denotes *torrent size*, the number of peers in the torrent at this time instant. In general, torrents with more peers tend to have a larger out-degree and in-degree, though the trend is very rough. The weighted out-degree and in-degree distribution according to our trace analysis follows power law rules roughly. It deviates from our model somewhat because of the heterogeneity of torrents in the real system.

C. Reducing Downloading Failure Ratio by Inter-torrent Collaboration

In the multi-torrent environment, old peers that had downloaded the file from a torrent may come back to download other data files, and the lifespan of this torrent can be extended if these old peers are willing to provide service. Assume the request arrival rate of this torrent is $\lambda(t)$ and $\lambda(t) = 0$ when $t < 0$. If we consider both new requesting peers and old returning peers, the peer arrival rate of the torrent is

$$\begin{aligned} \lambda'(t) &= \textstyle\sum_{l=0}^{k(t)} p^l \lambda(t - \frac{l}{r}) = \sum_{l=0}^{k(t)} p^l \lambda_0 e^{-\frac{t-\frac{l}{r}}{\tau}} \\ &= \lambda_0 e^{-\frac{t}{\tau}} \frac{q^{k(t)+1}-1}{q-1}, \end{aligned} \tag{23}$$

where $k(t) = \lfloor rt \rfloor$ and $q = pe^{\frac{1}{r\tau}}$ ($q > 1$ according to our trace analysis).

When $\lambda'(t) < \gamma$, the torrent is truly dead. The lifespan of a torrent without inter-torrent collaboration is $T_{life} = \tau \log(\frac{\lambda_0}{\gamma})$. Denoting the lifespan of the torrent with inter-torrent collaboration as T'_{life}, then $\lambda'(T'_{life}) = \gamma$, we have

$$\begin{aligned} \log\gamma &= \log\lambda_0 - \tfrac{T'_{life}}{\tau} + \log(q^{k(T'_{life})+1} - 1) - \log(q-1) \\ &\approx \log\lambda_0 - \tfrac{T'_{life}}{\tau} + (k(T'_{life}) + 1)\log q - \log(q-1) \\ &= \log\lambda_0 - \tfrac{T'_{life}}{\tau} + k(T'_{life})\log q + \log\tfrac{q}{q-1}. \end{aligned}$$

It leads to $\log(\frac{\lambda_0}{\gamma}\frac{q}{q-1}) \approx (\frac{1}{\tau} - r\log q)T'_{life}$. Thus

$$\begin{aligned} T'_{life} &\approx \tfrac{\tau\log(\frac{\lambda_0}{\gamma}\frac{q}{q-1})}{1-\tau r\log q} = \tfrac{\tau\log(\frac{\lambda_0}{\gamma}\frac{q}{q-1})}{\tau r\log\frac{1}{p}} \\ &> \tfrac{T_{life}}{\tau r\log\frac{1}{p}} = \beta T_{life}. \end{aligned} \tag{24}$$

According to the trace analysis and our modeling, $\beta = \frac{1}{\tau r\log\frac{1}{p}} \approx 6$. So we have

$$R'_{fail} = \frac{\int_{T'_{life}}^{\infty} \lambda_0 e^{-\frac{t}{\tau}} dt}{\int_0^{\infty} \lambda_0 e^{-\frac{t}{\tau}} dt} = e^{-\frac{T'_{life}}{\tau}} < R_{fail}^{\beta} \approx R_{fail}^{6}. \tag{25}$$

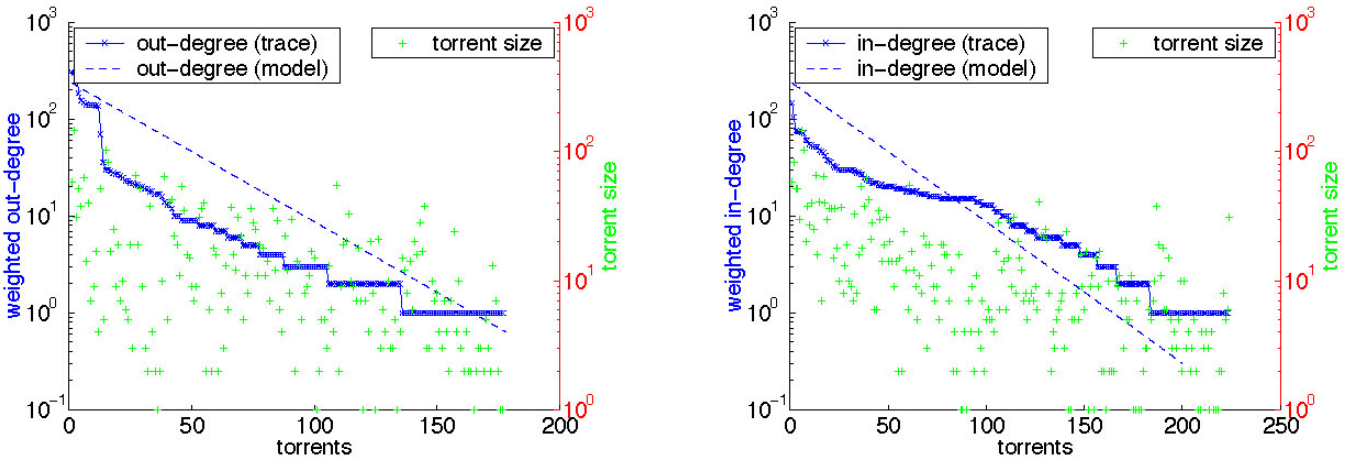
In the single-torrent model, since we cannot change the peer request pattern, the only way to decrease the downloading failure ratio is to decrease the seed leaving rate (i.e., to increase the seed service time). According to Equation 2 and 6, if the seed leaving rate is decreased from γ to γ^*, we have

$$T^*_{life} = \tau\log(\frac{\lambda_0}{\gamma^*}) = T_{life} + \tau\log\frac{\gamma}{\gamma^*}, \tag{26}$$

and

$$R^*_{fail} = \frac{\gamma^*}{\lambda_0} = R_{fail}\frac{\gamma^*}{\gamma}. \tag{27}$$

Comparing Equation 24 and 25 with Equation 26 and 27, we can see that inter-torrent collaboration is much more effective than stimulating seeds to serve longer in order to reduce the downloading failure ratio. Decreasing seed leaving rate can only extend torrent life span by a constant, while inter-torrent collaboration can increase torrent life span multiple



(a) Weighted out-degree

(b) Weighted in-degree

Fig. 13. The inter-torrent relation (y-axis is in log scale)

times. As a result, for reducing the downloading failure ratio, decreasing seed leaving rate has polynomial effect, while inter-torrent collaboration has exponential effect. For example, if the current downloading failure rate is 0.1, and seeds can be stimulated to stay 10 times longer (i.e., γ will decrease 10 times), then the downloading failure rate will decrease 10 times to 0.01. However, by inter-torrent collaboration, the downloading failure ratio can be as low as $0.1^6 = 10^{-6}$. The reason is that extending seed staying time only increases the service time for peers that arrive close to the seed generation time. With the passage of time, the peer arrival rate decreases exponentially, and finally the seed serving time will not be long enough for newly arriving peers. On the other hand, by exploiting inter-torrent collaboration, peers that have downloaded the file may return multiple times during a much longer period, and the downloading failure ratio can be significantly reduced to near zero.

V. A Discussion of Multi-Torrent Collaboration Systems

A. Tracker Site Overlay

We propose an architecture where tracker sites of different torrents self-organize into an overlay network to coordinate the collaboration among their peers. Each tracker site maintains a *Neighbor-Out Table* and a *Neighbor-In Table* to record the relationship with its neighboring torrents. The *Neighbor-Out Table* records the torrents that its peers can provide service to. The *Neighbor-In Table* records the torrents whose peers can provide service to this torrent. When a peer q joins a new torrent A, it uploads to its tracker site the information about from which torrents it had downloaded files previously. Then A's tracker site forwards this information to the tracker sites of those torrents where q had downloaded files from. By doing so, the torrents that are created independently by different content providers are connected together to form an overlay network, as shown in Figure 14(a). The tracker overlay is actually an implementation of the inter-torrent relation graph presented in Section IV-B. Figure 14(b) shows that the connectivity degree (unweighted) of the tracker overlay is heavily skewed and similar to P2P overlays like Gnutella networks. Thus, many existing search algorithms can be used in the tracker overlay.

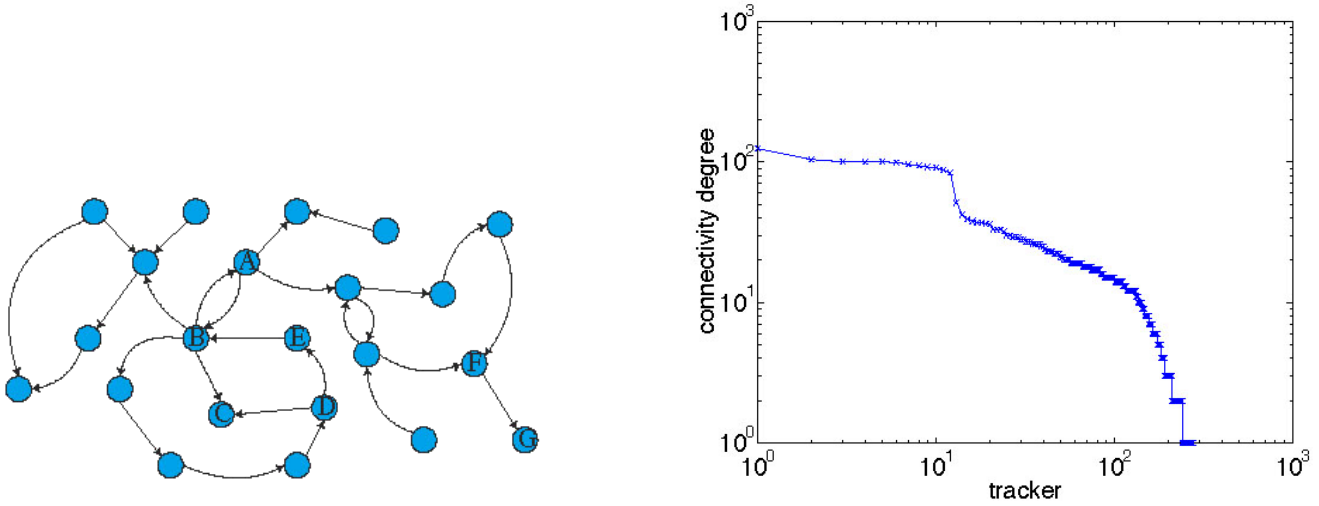
The tracker overlay provides a mechanism for inter-torrent collaboration. In the current architecture of BitTorrent systems, peers in different torrents cannot collaborate because they cannot find and communicate with each other. By using tracker overlay, peers in different torrents can exchange files that they have downloaded and balance their resource sharing. Our simulation shows that the tracker overlay can cover more than 99% torrents in the system.

In the tracker overlay architecture, the extra service load on the existing tracker sites is small. Assume a torrent has n peers at its peak time. Since the average number of torrents each peer involves is $\frac{1}{1-p}$, the neighbor table size is $O(\frac{n}{1-p})$. Furthermore, the tracker overlay is fully decentralized and has no single point of failure. Tracker overlay has better fault-tolerance and scalability than a central server solution.

Tracker overlay also provides a built-in mechanism to search content among multiple torrents. Currently, BitTorrent users rely on Web-based search engines to look for the content they want to download.

B. Multi-torrent Collaboration

BitTorrent assumes each peer is selfish, and a peer exchanges file chunks with those peers that provide it the best service. The incentive mechanism of BitTorrent systems is instant, because each peer must get corresponding benefit immediately for the service it provides. In contrast, KaZaa and eMule use a participation level or ID stored in the system to trace and identify the contribution of users. Peers with a higher level or higher ID will have a higher priority to be served. Thus, the contribution of a peer under this mechanism will be rewarded in a long term instead of instantly. Although KaZaa and eMule systems are multi-file based and thus the popularity changes of a file have little affect on the service availability of its downloading, their long-term incentive mechanisms are not as effective as the "tit-for-tat" mechanism in BitTorrent. The downloading speed of eDonkey/eMule/Overnet is much slower than that of BitTorrent because peers in the P2P network usually share and download a large number of files, making the



(a) An example of tracker site overlay

(b) Node connectivity of tracker site overlay (in log-log scale)

Fig. 14. Tracker site overlay

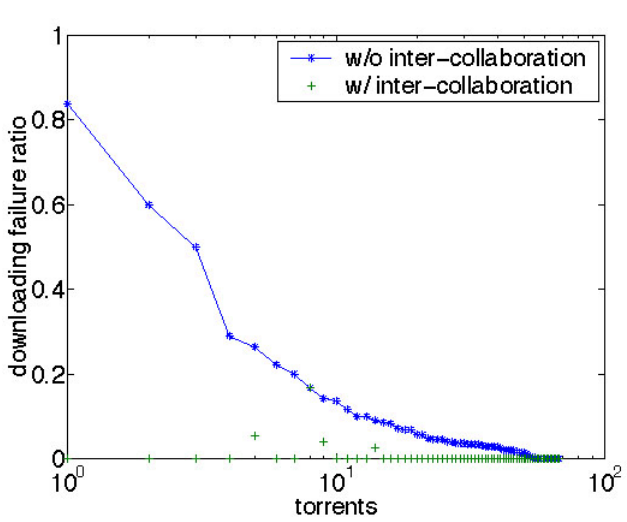
bandwidth available to each transfer much smaller than that in BitTorrent [5]. Furthermore, fraud prevention is also a big problem. For example, the participation level system in KaZaa has been cracked and thus one can set its participation level arbitrarily [6]. Compared with systems based on the long-term user reputation, the instant incentive mechanisms like "tit-for-tat" are simple, effective, and robust. Instead of going back to the long term incentive model of KaZaa and eDonkey systems, in this subsection, we propose an exchange based mechanism for *instant collaboration* among multiple torrents through the tracker site overlay, which still follows the "tit-for-tat" idea.

Our proposed inter-torrent collaboration strategy is as follows. First, if there exists a *directed cycle* among a number of torrents, such as torrents A, B and torrents B, C, D, E in Figure 14(a), then peers in these torrents can exchange file chunks through the coordination of the tracker site overlay. More specifically, a peer that needs service from peers in other torrents on the cycle does not have to serve these peers directly, because its contribution can be transfered to these peers along the cycle with the help of corresponding trackers. Since the contribution of each peer must be rewarded instantly, any fraudulent behavior will be identified and punished at once. Second, when no such cycles exist for a peer q who wants to get service from peers in other torrents, the peer can construct such a cycle as follows. Peer q may join these torrents temporarily and download some chunks of the files, even if it does not want these files itself. Through the coordination of corresponding tracker sites, the peer can provide uploading service for these chunks only, and attribute its service contribution to the peers it wants to get service from, so that these peers can get benefit from the peers that q serves and offer q the service it needs. Thus, a directed neighboring cycle is constructed. We call this technique *bandwidth trading*. The basic idea is that the bandwidth can only be shared through content downloading/uploading. Since a file chunk can be served to multiple peers in the system, bandwidth trading is very efficient and the overhead is trivial.

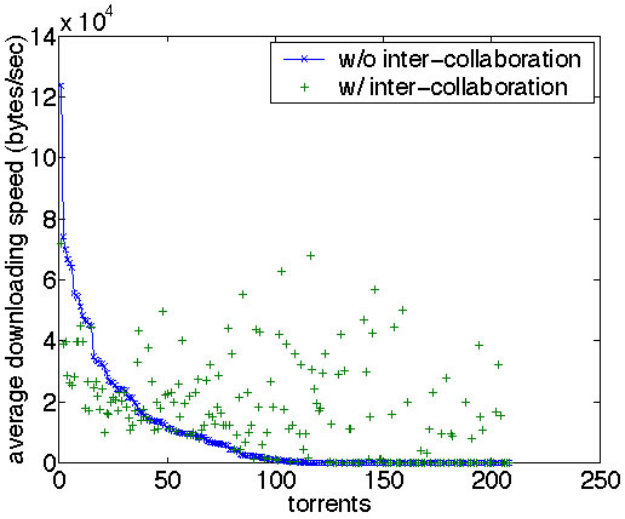
In such multi-torrent collaboration systems, a peer that has downloaded multiple files can get better service when downloading a new file (no matter it is popular or not) by serving the old files it has downloaded to the peer community, thus addressing the well-known problem of lacking incentives to seeds. Research [8] and [11] present a similar idea of using file exchange as an incentive for P2P content sharing. Different from these studies, our system aims to share bandwidth as well as content across multiple P2P networks.

C. Performance Evaluation

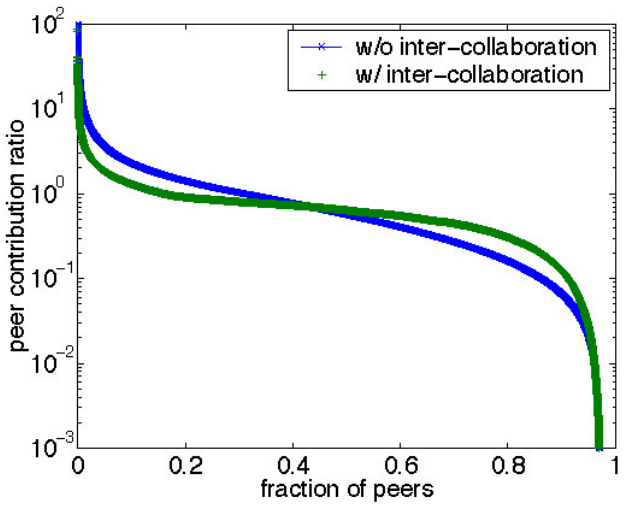
We evaluate our system design through simulations with the tracker trace used in previous sections. In the simulations, we assume seeds use a fair service policy that does not prefer any peer in any torrent as long as it can serve. Figure 15(a) shows the downloading failure ratio in the current BitTorrent systems (without inter-torrent collaboration) and that in our proposed system (with inter-torrent collaboration), respectively. We only consider torrents born in the initial period of the trace collection time in order to study the performance in these torrent's whole lifetime. In this figure, each point in the x-axis denotes a torrent, sorted in non-ascending order of the corresponding downloading failure ratio in the tracker trace (without inter-torrent collaboration). As shown in Figure 15(a), under inter-torrent collaboration, the downloading failure ratios in most torrents are actually zero or close to zero. Figure 15(b) shows the average downloading speeds of peers in different torrents at a certain time (12:00:01 on 2003-11-15). Each point in the x-axis denotes a torrent, sorted in non-ascending order of the average downloading speed of torrents at this time instant. The system with inter-torrent collaboration clearly provides a much better and more stable downloading service to clients. Figure 15(c) shows the peer contribution ratio (defined in Section III) in two systems. In our proposed system, the peer contribution ratio is better balanced. These preliminary results demonstrate that our proposed system design, though without



(a) The downloading failure ratio (x-axis is in log scale)



(b) The average download speed



(c) The ratio of uploaded bytes to downloaded bytes (y-axis is in log scale)

Fig. 15. The performance analysis of our system

complicated credit systems, can enhance the current BitTorrent system significantly.

VI. Conclusion

BitTorrent-like systems have become increasingly popular for content distribution and file sharing, and have contributed to a large amount of traffic on the Internet. In this paper, we have performed extensive trace analysis and modeling to study the behaviors of such systems. We found that the existing BitTorrent system provides poor service availability, fluctuating downloading performance, and unfair services to peers. Our model has revealed that these problems are due to the exponentially decreasing peer arrival rate and provides strong motivation for inter-torrent collaborations instead of simply giving seeds incentives to serve longer. We propose the design of a new architecture where the tracker sites of different torrents are organized into an overlay to facilitate inter-torrent collaboration. Out preliminary simulations have shown promising results.

Appendix

A. The Resolution of the Fluid Model

Equation 7 is a non-homogenous ODE equation system. A particular solution for 7 is

$$\begin{cases} x = d_1 e^{-t/\tau}, \\ y = d_2 e^{-t/\tau}, \end{cases} \tag{28}$$

where

$$\begin{cases} d_1 = \frac{-\lambda_0}{\frac{\mu^2\eta}{\mu-\gamma+1/\tau} - (\theta+\mu\eta-\frac{1}{\tau})}, \\ d_2 = \frac{\lambda_0}{\mu - \frac{(\theta+\mu\eta-1/\tau)(\mu-\gamma+1/\tau)}{\mu\eta}}. \end{cases} \tag{29}$$

The eigen equation is

$$\psi^2 + (\mu+\theta+\gamma-\mu)\psi + \mu\eta\gamma + \theta(\gamma-\mu) = 0. \tag{30}$$

When the corresponding homogenous equation system has two different real eigenvalues $\psi_1 \neq \psi_2$, the resolution can be expressed as

$$\begin{cases} x = ae^{\psi_1 t} + be^{\psi_2 t} + d_1 e^{-t/\tau}, \\ y = c_1 ae^{\psi_1 t} + c_2 be^{\psi_2 t} + d_2 e^{-t/\tau}, \end{cases} \tag{31}$$

where

$$\begin{cases} c_1 = -\frac{\psi_1+\theta+\mu\eta}{\mu}, \\ c_2 = -\frac{\psi_2+\theta+\mu\eta}{\mu}, \\ a = \frac{-c_2 d_1 + d_2 - 1}{c_2 - c_1}, \\ b = \frac{-c_1 d_1 + d_2 - 1}{c_1 - c_2}. \end{cases} \tag{32}$$

When the corresponding homogenous equation system has two equal real eigenvalues $\psi_1 = \psi_2$, the resolution can be expressed as

$$\begin{cases} x = (a+bt)e^{\psi_1 t} + d_1 e^{-t/\tau}, \\ y = (ac_1 + bc_2 + bc_1 t)e^{\psi_1 t} + d_2 e^{-t/\tau}, \end{cases} \tag{33}$$

where

$$\begin{cases} c_1 = -\frac{\psi_1+\theta+\mu\eta}{\mu}, \\ c_2 = -\frac{1}{\mu}, \\ a = -d_1, \\ b = \frac{1-d_2}{c_2}. \end{cases} \tag{34}$$

When the corresponding homogenous equation system has a pair of conjugate complex eigenvalues $\alpha \pm \beta i$, the resolution can be expressed as

$$\begin{cases} x = e^{\alpha t}(c_1 \cos \beta t + c_2 \sin \beta t) + d_1 e^{-t/\tau}, \\ y = -s e^{\alpha t}(c_1 \cos(\beta t + \phi) + c_2 \sin(\beta t + \phi)) + d_2 e^{-t/\tau}, \end{cases} \tag{35}$$

where

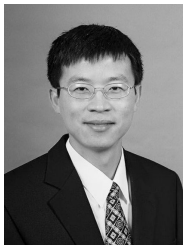
$$\begin{cases} s = \frac{1}{\mu\sqrt{(\alpha+\theta+\mu\eta)^2+\beta^2}}, \\ \phi = \tan^{-1}(\frac{\beta}{\alpha+\theta+\mu\eta}), \\ c_1 = -d_1, \\ c_2 = \frac{-1/s - d_2 + d_1 \cos\phi}{\sin\phi}. \end{cases} \tag{36}$$

This work is partially supported by the National Science Foundation under grants CNS-0098055, CNS-0405909, and CNS-0509054/0509061. Some preliminary results of this work have been presented in [15]. We would like to thank Mikel Lzal for providing BitTorrent traces to us. We appreciate Oliver Spatscheck, Keith W. Ross, and William L. Bynum for their constructive comments.

REFERENCES

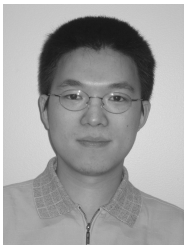
[1] http://www.bigchampagne.com/.
[2] http://www.gnutelliums.com/.
[3] http://www.kazaa.com/.
[4] http://www.edonkey2000.com/.
[5] http://en.wikipedia.org/wiki/BitTorrent.
[6] Hack kazaa participation level - the easy answer. http://www.davesplanet.net/kazaa/.
[7] E. Adar and B.Huberman. Free riding on Gnutella. Technical report, Xerox PARC, Aug. 2000.
[8] K. G. Anagnostakis and M. B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *Proc. of IEEE ICDCS*, Mar. 2004.
[9] A. Bellissimo, B. N. Levine, and P. Shenoy. Exploring the use of BitTorrent as the basis for a large trace repository. Technical Report 04-41, University of Massachusetts Amherst, June 2004.
[10] B. Cohen. Incentives build robustness in BitTorrent. In *Proc. of Workshop on Economics of Peer-to-Peer Systems*, May 2003.
[11] L. P. Cox and B. D. Noble. Samsara: Honor among thieves in P2P storage. In *Proc. of ACM SOSP*, Oct. 2003.
[12] C. Cranor, T. Johnson, and O. Spatscheck. Gigascope: A stream database for network applications. In *Proc. of ACM SIGMOD*, June 2003.
[13] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley. Modeling peer-peer file sharing systems. In *Proc. of IEEE INFOCOM*, Mar. 2003.
[14] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of ACM SOSP*, Oct. 2003.
[15] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *Proc. of Internet Measurement Conference*, Oct. 2005.
[16] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garc'es-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. In *Proc. of Passive & Active Measurement Workshop*, Apr. 2004.
[17] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proc. of ACM WWW*, May 2003.
[18] R. Ma, S. Lee, J. Lui, and D. Yau. A game theoretic approach to provide incentive and service differentiation in P2P networks. In *Proc. of ACM SIGMETRICS*, June 2004.
[19] L. Massoulie and M. Vojnovic. Coupon replication systems. In *Proc. of ACM SIGMETRICS*, June 2005.
[20] A. Parker. The true picture of peer-to-peer file sharing. http://www.cachelogic.com, 2004.
[21] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The BitTorrent P2P file-sharing system: Measurements and analysis. In *Proc. of International Workshop on Peer-to-Peer Systems*, Feb. 2005.
[22] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proc. of ACM SIGCOMM*, Aug. 2004.
[23] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An analysis of Internet content delivery systems. In *Proc. of USENIX OSDI*, Dec. 2002.
[24] S. Saroiu, K. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of ACM/SPIE MMCN*, Jan. 2002.
[25] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol. In *Proc. of IEEE INFOCOM*, Mar. 2004.
[26] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proc. of IEEE INFOCOM*, Mar. 2003.
[27] X. Yang and G. Veciana. Service capacity of peer to peer networks. In *Proc. of IEEE INFOCOM*, Mar. 2004.

Lei Guo received the B.S. degree in space physics and M.S. degree in computer science from the University of Science and Technology of China in 1996 and 2002, respectively. He received the S. Park Graduate Research Award at the College of William and Mary in 2005. He is currently a Ph.D. Candidate in the Department of Computer Science and Engineering at the Ohio State University. His research interests are in the areas of distributed systems, peer-to-peer systems, multimedia systems, and Internet measurement and modeling.

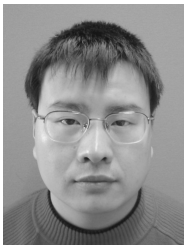


Songqing Chen is an assistant professor in the Department of Computer Science at George Mason University, Fairfax, VA. His research interests embrace various subjects in operating systems, distributed systems, and high performance computing. Chen received his Ph.D. in Computer Science from the College of William and Mary.

Zhen Xiao received his Ph.D. from Cornell University in January 2001. After that he worked as a senior technical staff member at AT&T Labs – Research, in Florham Park, NJ for five years where he received the "Research Excellence Award". He is now a Research Staff Member at IBM T.J. Watson Research Center. His research interests include SIP, multimedia, IPTV, Grid computing, Web technologies and content delivery, security and dependability, and reliable multicast.



Enhua Tan received his B.E. degree in Computer Science and Technology from University of Science and Technology of China in 2001, and received his M.E. degree in Computer Architecture from Institute of Computing Technology, Chinese Academy of Sciences in 2004. He is a currently a Ph.D. student of computer science and engineering at the Ohio State University.



Xiaoning Ding received the B.S. degree and M.S. degree in computer science from Northwestern Polytechnical University of China in 1996 and 1998, respectively. He is currently a Ph.D. student in Computer Science and Engineering Department of the Ohio State University.

Xiaodong Zhang is the Robert M. Critchfield Professor in Engineering, and Chair of Department of Computer Science and Engineering at the Ohio State University. He served as the Program Director of Advanced Computational Research at the National Science Foundation, 2001-2004. He is the associate Editor-in-Chief of IEEE Transactions on Parallel and Distributed Systems, and is serving on the Editorial Boards of the IEEE Transactions on Computers, IEEE Micro, and Journal of Parallel and Distributed Computing.