Efficient and Highly Available Peer Discovery: A Case for Independent Trackers and Gossipping

György Dán*, Niklas Carlsson†, Ilias Chatzidrossos*
School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden, {gyuri|iliasc}@ee.kth.se
Linköping University, Linköping, Sweden, niklas.carlsson@liu.se

***Abstract*—Tracker-based peer-discovery is used in most commercial peer-to-peer content distribution systems, as it provides performance benefits compared to distributed solutions, and facilitates the control and monitoring of the overlay. But a tracker is a central point of failure, and its deployment and maintenance incur costs; hence an important question is how *high tracker availability* can be achieved at *low cost*. We investigate highly available, low overhead peer discovery, using *independent trackers* and a simple *gossip protocol*. This work is a step towards understanding the trade-off between the overhead and the achievable peer connectivity in highly available distributed overlay-management systems for peer-to-peer content distribution.**

We propose two protocols that connect peers in different swarms efficiently with a constant, but tunable, overhead. The two protocols, Random Peer Migration (RPM) and Random Multi-Tracking (RMT), employ a small fraction of peers in a torrent to virtually increase the size of swarms. We develop analytical models of the protocols based on renewal theory, and validate the models using both extensive simulations and controlled experiments. We illustrate the potential value of the protocols using large-scale measurement data that contains hundreds of thousands of public torrents with several small swarms, with limited peer connectivity. We estimate the achievable gains to be up to 40% on average for small torrents.

I. Introduction

Efficient overlay-membership management and peer discovery are crucial components of large-scale peer-to-peer (P2P) content distribution systems. Peer discovery provides the set of potential neighbors with which a peer can exchange data and, in general, the efficiency of content distribution improves as the set of potential neighbors increases. Without peer discovery, a P2P system is not able to operate, and hence high system availability is a key requirement. Furthermore, as large-scale P2P systems can have up to tens of millions of peers simultaneously exchanging millions of distinct objects, efficiency and low-overhead are key requirements as well.

Fully distributed peer-discovery mechanisms, such as distributed hash tables (DHTs) used in BitTorrent [1], and unstructured overlays used in Gnutella, provide high availability. Maintaining consistency and avoiding stale routing tables under node churn at a low overhead is, however, challenging and affects the performance of DHT-based peer discovery in practice [1], [2]. In unstructured overlays, traffic overhead has to be traded for good peer discovery [3]. Furthermore, fully distributed overlay management and peer discovery render it more difficult for P2P content providers to monitor peer participation (e.g., for charging purposes), and to enforce content access control.

At the opposite end of the availability spectrum is centralized peer discovery using a single tracker. A tracker makes it easy to monitor peer participation, to enforce access control, and it is also efficient in terms of overhead. Many successful open-source and proprietary P2P content distribution systems rely on trackers for peer discovery (e.g., BitTorrent, PPLive [4] and Akamai Netsession [5], a hybrid P2P content delivery platform). Nevertheless, the tracker is a central point of failure and its traffic load is proportional to the number of peers.

While tracker availability can be improved by investing in reliable network connectivity, hardware and software (e.g., a reliable database), it is often cheaper to deploy multiple trackers based on commodity hardware and network access, and to ensure that failures would be independent (through geographic, topographic and vendor diversity). BitTorrent, for example, allows the use of multiple trackers through the Multi-tracker Metadata Extension (BEP-0012 [6]). However, using multiple trackers can have some adverse effects. On the one hand, if all peers communicate to all trackers, then the traffic costs increase proportionally to the number of trackers, leading to overhead. On the other hand, if every peer communicates with one tracker only, then the overlay is split into a number of disjoint sets of connected peers, even if gossip protocols are used within the disjoint sets of peers to reduce the tracker load; e.g., as in BitTorrent (the Peer Exchange Protocol [6], [7]) and in PPLive [4].

The focus of our work is to develop and demonstrate a hybrid approach, which has the advantages of the centralized and decentralized peer-discovery mechanisms, but avoids their respective disadvantages. In particular, the question we address in this paper is whether it is possible to achieve *highly available* and *efficient* peer-discovery, which avoids the formation of disjoint swarms, at *low overhead* by employing *independent trackers* and relying only on a *gossip protocol*. Our main contributions in this paper are the following:

1) We propose two novel distributed algorithms that rely on a gossip protocol only, and mix otherwise disjoint sets of peers into a single overlay. The overhead of both protocols is independent of the number of peers.
2) We develop an analytical model based on the theory of renewal-reward processes and show that, under certain conditions, the mixing performance of the algorithms is independent of the rate of peer churn and of the overlay size. We validate the analytical results using simulations and controlled experiments with BitTorrent clients.

3) Based on a large-scale measurement study of BitTorrent content popularity, we estimate the potential throughput performance improvement in today's BitTorrent.

The remainder of the paper is organized as follows. Section II describes our terminology and system model. Section III explains our proposed overlay management protocols. Section IV provides analytic models of the protocols. Section V shows simulation and experimental results to validate the models. Section VI shows performance results based on measurements data. Related work is discussed in Section VII. Section VIII concludes the paper.

II. System Model

Whereas our protocols and analysis are applicable to tracker-based P2P content distribution in general, throughout the paper we use BitTorrent terminology to refer to components of the P2P content distribution system and protocol. Since our focus is on the control plane, we do not discuss any aspects of data forwarding.

The P2P system consists of peers that are interested in a number of contents (e.g., they share a set of files or they distribute live or on-demand streaming content). The set of peers that share a particular content is called a *torrent*. We denote the set of torrents by $\mathscr{T}$. The number of peers in torrent $t \in \mathscr{T}$ is denoted by x_t. For the case of file sharing, we distinguish between peers that have the entire content and only upload content, called *seeds*, and peers that only have parts of the file and are downloading, called *leechers*.

Trackers are used to maintain state information about the peers currently having pieces of a particular file. We denote the set of trackers by $\mathscr{R}$, and by $\mathscr{R}(t)$ the set of trackers that track torrent $t \in \mathscr{T}$. $\mathscr{T}(r)$ denotes the set of torrents that are tracked by tracker $r \in \mathscr{R}$. The number of trackers $|\mathscr{R}(t)| \geq 1$ is a function of the target torrent availability, and is a system parameter. Our focus is on the case when $|\mathscr{R}(t)| > 1$.

A peer that wants to join a torrent has to obtain the *torrent meta-data*, which contains the set of available trackers for the torrent, $\mathscr{R}(t)$. The peer then chooses a subset of the available trackers, registers with those trackers, and periodically, once every *announce* period of time T_A, provides them with information about its state, e.g., when it completes download and when it leaves. Upon *announce request*, the tracker can also provide the peer with a subset of the registered peers. The term *swarm* is used to denote the set of peers that share the same file and are tracked by the same tracker. We denote the number of peers tracked by tracker r for torrent t, i.e., the swarm size, by $x_{t,r}$.

Apart from a tracker, a peer can also obtain peer information from other peers using a *gossip protocol*. Peers use the gossip protocol to periodically exchange addresses of known peers with their neighbors. Ideally, a peer would remember all addresses ever learned, and would gossip all known addresses to randomly chosen peers. In practice, however, gossiping is often limited to a slowly changing set of neighbors, and only the addresses of connected peers are distributed. To reflect this limitation we denote by p the peer list, i.e., the maximum number of peers that a peer is connected to in a swarm, $p \leq x_{t,r}$. Table I summarizes our notation.

Symbol	Definition
$\mathscr{R}$	Set of trackers
$\mathscr{T}$	Set of torrents
$\mathscr{R}(t)$	Set of trackers that track torrent t
$\mathscr{T}(r)$	Set of torrents that are tracked by tracker r
x_t	Number of peers associated with torrent t
$x_{t,r}$	Number of peers of torrent t that are tracked by tracker r
η	Fraction of peers implementing *RPM* or *RMT*
β	*RPM* or *RMT* protocol parameter (willingness)
p	Peer list length for the gossip protocol

TABLE I
Frequently used notation

III. Overlay Management Protocols

Efficient peer-discovery mechanisms are an important component in achieving high content-distribution efficiency in P2P systems. If every peer registers with one tracker chosen at random, then the tracker load is minimal both in terms of traffic and the amount of system state to be maintained. Nevertheless, the different swarms of the same torrent will be pairwise disjoint, and peers will not be aware of peers in other swarms. We refer to this scheme as *pick-one*. Alternatively, if every peer registers with all trackers, then the trackers' load increases proportional to the number of swarms per torrent weighted by the torrent sizes. As an advantage, all peers can potentially discover all other peers; however, this may require much higher traffic overhead. We refer to this scheme as *pick-all*. These two schemes are the two extremes in terms of overhead and overlay connectivity.

In the following we propose two protocols, Random Peer Migration (*RPM*) and Random Multi-Tracking (*RMT*), that provide good overlay connectivity at the price of low overhead (compared to *pick-one*), based on independent trackers and a gossip protocol. We analyze the protocols in Section IV.

A. Random Peer Migration (RPM)

Peers that follow the *RPM* protocol migrate between the swarms of the torrent at random, and whenever they arrive to a new swarm they distribute the addresses of the peers in the previous swarm using the gossip protocol. Letting peers migrate between swarms is a novel but simple way to increase the number of peers that know about each other (i.e., the virtual size of the swarms). What is particularly novel, and what makes *RPM* non-trivial, is that the migration rule is defined such as to ensure that the protocol has a constant, but tunable, tracker overhead independent of the torrent's size. At the same time it provides good performance for a wide range of swarm sizes, peer arrival rates and holding times.

The *RPM* protocol that we propose works as follows. Upon arrival a peer registers with a tracker $r \in \mathscr{R}(t)$ chosen uniform at random. Every time a peer downloading torrent t tracked by tracker r finishes uploading or downloading data worth $1/(\beta(|\mathscr{R}(t)|-1))$ portion of the shared content's size, it chooses to migrate to another swarm of the same torrent with probability $1/x_{t,r}$. We call the protocol parameter β the

willingness to migrate. If a peer chooses to migrate, the peer chooses uniformly at random a tracker r' other than tracker r (i.e., $r' \in \mathscr{R}(t) \setminus \{r\}$), and scrapes the chosen tracker. If $x_{t,r'} = 0$, the peer stays in swarm r; otherwise, it migrates to swarm r'. In order to migrate, the peer unregisters from tracker r, registers with tracker r', obtains a list of known peers from tracker r', and uses the gossip protocol to distribute the addresses of the peers it knows about from the previous swarm r to the peers it now knows in swarm r'. We refer to the peers whose addresses get to be known in swarm r' this way as external peers, as they are not tracked by tracker r'.

B. Random Multi-Tracking (RMT)

Peers that follow the *RMT* protocol associate with several trackers at random upon arrival, which is a rather natural way to achieve mixing between swarms, inspired by the *pick-all* scheme. The novelty lies in the choice of the number of multi-tracked peers. *RMT* provides good mixing between the swarms for a wide range of swarm sizes, peer arrival intensities and content sizes. At the same time it leads to constant, but tunable, tracker announce overhead.

The proposed *RMT* protocol works as follows. When a peer joins torrent t it scrapes all trackers $r \in \mathscr{R}(t)$ to obtain the number of peers x_t. If $x_t = 0$, then the peer registers with one tracker chosen uniform at random. If $x_t > 0$, then with probability $min(1, \frac{|\mathscr{R}(t)|\beta}{kx_t})$ it registers with k trackers chosen uniform at random, otherwise it registers with one tracker chosen uniform at random. We call the protocol parameter β the willingness to multi-track. Peers that are registered with k trackers maintain an equal number of connections with peers in the different swarms, on average p peers per swarm. To provide mixing they use the gossip protocol to distribute the addresses of the peers they are connected to.

IV. Protocol Performance

In the following we develop analytical models that provide insight into the effect of the protocol parameters on the protocol performance. Our focus is on a single torrent, and for simplicity we omit the subscript t in this section.

A. Performance Metrics

In order to evaluate the performance of the protocols we define the *virtual swarm size* as the average number of internal and external peers known in swarm r normalized with the total number of peers in the torrent

$$M_r = \frac{x_r + \sum_{r' \in \mathscr{R}(t) \setminus \{r\}} \bar{y}_{r,r'}}{x}, \tag{1}$$

where $\bar{y}_{r,r'}$ is the average number of external peers known in swarm r tracked in swarm r', and x is the number of peers in the torrent. Without *RPM* and *RMT* $\bar{y}_{r,r'} = 0$. In general, $\bar{y}_{r,r'} = \lim_{\tau\to\infty} \frac{1}{\tau} \int_0^\tau y_{r,r'}(h)dh$, where $y_{r,r'}(h)$ is the number of external peers in swarm r tracked in swarm r' at time h.

The *average virtual swarm size* for torrent t can be expressed as the weighted average $M = \frac{1}{x} \sum_{r \in \mathscr{R}(t)} x_r M_r$. This metric corresponds to the average effective swarm size observed by a peer and is upper bounded, $M \leq 1$. Without mixing $M = \sum_{r \in \mathscr{R}(t)} (x_r/x)^2$, which is minimal for uniform swarm sizes ($M = 1/|\mathscr{R}(t)|$). The gain of swarm management is then the increase of the virtual swarm size due to mixing

$$dM = M - \sum_{r \in \mathscr{R}(t)} (x_r/x)^2. \tag{2}$$

We quantify the overhead of the proposed protocols primarily in terms of the *tracker overhead* compared to the *pick-one* scheme. The overhead is due to the redundant state information maintained in the trackers and to the tracker scrapes and announce requests performed by the peers. We do not provide an analysis of the peers' overhead due to gossiping for two reasons. First, the overhead is no greater than if all peers were in a single swarm. Second, the amount of gossiping traffic is negligible compared to the amount of data traffic, as shown by our experiments with BitTorrent clients in Section V-B.

B. Virtual Swarm Size

In the following we describe our modeling assumptions and then derive closed form expressions for the average number of external peers $\bar{y}_{r,r'}$ and discuss its impact on the average mixing efficiency of each of the protocols.

Consider a torrent t and two of its swarms r and r'. Assume peers arrive to swarm r of the torrent according to a Poisson process with rate λ_r. A share η of the arriving peers follows the *RPM* or the *RMT* protocol. The holding time of the peers is exponentially distributed with mean $1/\mu_r$. The time it takes a peer to download the content in torrent t is exponentially distributed with mean $1/\nu_r$. In such a system peers depart with probability $\mu_r/(\mu_r + \nu_r)$ without finishing the download of the content. Using the above notation, the average number of peers tracked by tracker r is λ_r/μ_r, independent of the peers' holding time distribution. We do not incorporate the effect of β on ν_r in this model. This simplification is pessimistic for *RPM*: if peer migration increases the torrent throughput, then it increases ν_r, so that migration would become more frequent and swarm mixing would be more efficient.

In order to develop a lower bound on the average number of external peers from swarm r' known in swarm r, i.e., $\bar{y}_{r,r'}$, we model the protocols with a renewal reward process $\{(J_i, R_i) : i \geq 0\}$ [8]. The jump times J_i of the renewal process are the migration events in case of *RPM*, and the arrival epochs of a multi-tracked peer and the time instances of the announces made by the last arriving multi-tracked peer for *RMT*. The rewards R_i are the cumulative number of external peers during a renewal period. The reward over the i^{th} renewal period $[J_i, J_{i+1}]$ can be defined as $R_i = \int_{J_i}^{J_{i+1}} y_{r,r'}(\tau)d\tau$. The average number of external peers equals the average reward, $\bar{y}_{r,r'} = \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{i=1}^{N(\tau)} R_i$, where $N(\tau)$ is the counting process for the renewal process (i.e., the number of renewal epochs until time t). In the following we give lower bounds on $\bar{y}_{r,r'}$ using the above modeling assumptions for *RPM* and *RMT*.

1) Random Peer Migration (RPM): Consider a swarm r, to which peers migrate from swarm r'. The migrating peers know about p peers of swarm r', and spread the addresses of these peers upon their arrival to swarm r. Pessimistically, we assume

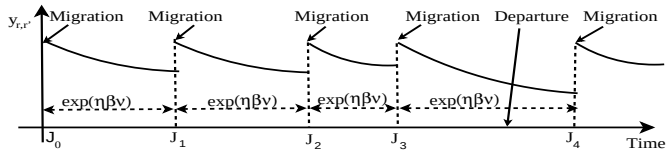


Fig. 1. Jump times and renewal periods in the renewal process for *RPM*. Jump times $J_0,\ldots,J_4$ correspond to the migration of peer $i=0,\ldots,4$ to the swarm. Peer 3 departs some time before J_4. The renewal period length is exponentially distributed.

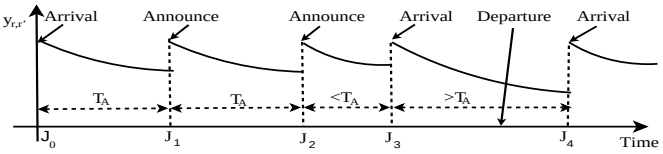


Fig. 2. Jump times and renewal periods in the renewal process for *RMT*. At J_0 peer 1 arrives, at J_1 peer 1 performs an announce, at J_2 peer 1 performs another announce, at J_3 peer 2 arrives and at J_4 peer 3 arrives. Peer 2 departs some time before J_4.

that the peers in swarm r forget about the external peers they learnt about from the previous migrating peer every time a peer migrates to swarm r from swarm r'. This assumption implies that all external peers from swarm r' known in swarm r are known due to the last migrating peer. Using this assumption we underestimate the number of external peers from swarm r' known in swarm r. Peers depart with intensity μ, so that time z after the last migration event the average number of external peers that remain from the original p peers is $y_{r,r'}(z)=pe^{-\mu z}$.

Figure 1 illustrates the renewal process; the jump times J_i and the instantaneous reward, which is equal to the average number of external peers $y_{r,r'}(\tau)$ at time τ. The time between migration events is exponentially distributed with intensity $\eta\beta\nu$, so the average renewal period length is

$$E[J_{i+1}-J_i]=(\eta\beta\nu)^{-1}, \tag{3}$$

and the average reward over a renewal period is

$$E[R_i]=\int_0^\infty\Big(\int_0^h pe^{-\mu z}dz\Big)\eta\beta\nu e^{-\eta\beta\nu h}dh=p(\eta\beta\nu+\mu)^{-1}. \tag{4}$$

We can calculate the average reward over time as

$$\bar{y}_{r,r'}=\lim_{\tau\to\infty}\frac{1}{\tau}\int_0^\tau y_{r,r'}(\tau)=\frac{E[R_i]}{E[J_{i+1}-J_i]}=p\frac{\eta\beta\nu}{\eta\beta\nu+\mu}, \tag{5}$$

where the second equality holds with probability one according to the renewal reward theorem [8], and the third equality is obtained by substituting (3) and (4). We can make three important observations based on (5).

First, the average number of external peers increases in the willingness to migrate β, but with a decreasing marginal gain. Since the scraping overhead increases linearly in β (see Section IV-C1), β should not be chosen too high.

Second, if one only considers torrents in which the average peer holding time is at least equal to the average time to download (i.e., $\nu\geq\mu$) then $\nu=\mu$ is a worst case scenario, and the number of external peers is lower bounded by $p\frac{\eta\beta}{1+\eta\beta}$.

Third, the bound is a function of the product $\eta\beta$, hence it is enough to focus on the effect of the parameter β for fixed η to understand the mixing efficiency. For simplicity, we can assume that $\eta=1$ and vary $\eta\beta$ by varying β. This result also shows that *RPM* can be highly beneficial even if only a small fraction η of peers implements the protocol.

2) Random Multi-Tracking (RMT): Consider the number of multi-tracked peers that are registered with tracker r and r'. These are the peers that contribute to the mixing between the two swarms. A multi-tracked peer that registers with k trackers transfers peer information between $k(k-1)/2$ pairs of swarms out of the $|\mathscr{R}(t)|(|\mathscr{R}(t)|-1)/2$ pairs of swarms, so that an arbitrary multi-tracked peer is registered with trackers r and r' with probability $\frac{k(k-1)}{|\mathscr{R}(t)|(|\mathscr{R}(t)|-1)}$. Hence, the arrival rate of multi-tracked peers registered with trackers r and r' is

$$\lambda_{r,r'}=\eta\lambda\frac{n\beta}{kx}\frac{k(k-1)}{|\mathscr{R}(t)|(|\mathscr{R}(t)|-1)}=\eta\beta\frac{k-1}{|\mathscr{R}(t)|-1}\mu, \tag{6}$$

where we used that $x=\lambda/\mu$ in steady state.

Every multi-tracked peer is connected to p peers per swarm. Whenever a multi-tracked peer arrives or announces to tracker r, it obtains a list of p peers from tracker r and disseminates the information among peers in swarm r'. But since peers depart with intensity μ, time z after the last announce the average number of external peers that remain from the original p peers is $y_{r,r'}(z)=pe^{-\mu z}$. Pessimistically, we assume that every time a new multi-tracked peer arrives to swarms r and r', the peers in swarm r forget about the external peers from swarm r' that they learnt about previously (and vice-versa). After the last multi-tracked peer departs, the number of known external peers keeps decaying until a new multi-tracked peer arrives. These assumptions are similar to the ones made for the analysis of *RPM*, and provide us with a lower bound on the number of external peers from swarm r' known in swarm r.

In the following we calculate the average renewal period length and the average reward during a renewal period. Figure 2 illustrates the three kinds of renewal periods; the jump times J_i and the instantaneous reward, which is equal to the average number of external peers $y_{r,r'}(\tau)$ at time τ.

The first kind of renewal period is between two announces performed by a multi-tracked peer. The multi-tracked peer performs the first announce upon arrival, and performs announces periodically every T_A time. The corresponding renewal periods in Figure 2 are $[J_0,J_1]$ and $[J_1,J_2]$. Such renewal periods happen if the last multi-tracked peer does not depart during an announce interval and no new multi-tracked peer arrives. This happens with probability $p_1=e^{-\mu T_A}e^{\lambda_{r,r'}T_A}$. The distribution of the length of this renewal period is deterministic, with probability density function $f_1(h)=\delta_{T_A}(h)$.

The second kind of renewal period is between an announce performed by a multi-tracked peer and the arrival of a new multi-tracked peer before the next announce period. The corresponding renewal period in Figure 2 is $[J_2,J_3]$. Such a renewal period happens if a new multi-tracked peer arrives before time T_A after the last announce. This happens with probability $p_2=1-e^{-\lambda_{r,r'}T_A}$. The length of this renewal period follows a truncated exponential distribution on the interval

$(0, T_A]$, with probability density function $f_2(h) = \frac{\lambda_{r,r'} e^{-\lambda_{r,r'} h}}{1-e^{-\lambda_{r,r'} T_A}}$.

The third kind of renewal period is between an announce performed by a multi-tracked peer and the arrival of a new multi-tracked peer, if the last multi-tracked peer departed before the arrival of the next one. The corresponding renewal period in Figure 2 is $[J_3, J_4]$. Such a renewal period happens if the last multi-tracked peer departs and the next multi-tracked peer arrives more than time T_A after the last announce. This happens with probability $p_3 = e^{\lambda_{r,r'} T_A}(1 - e^{\mu T_A})$. The length of the renewal period follows a shifted exponential distribution on the interval (T_A, ∞) with probability density function $f_3(h) = \lambda_{r,r'} e^{\lambda_{r,r'}(h-T_A)}$.

The average renewal period length is the weighted average

$$\begin{aligned} E[J_{i+1} - J_i] &= p_1 T_A + p_2 \int_0^{T_A} h f_2(h) dh + p_3 \int_{T_A}^{\infty} h f_3(h) dh \\ &= \frac{1}{\lambda_{r,r'}}(1 - e^{-(\mu+\lambda_{r,r'})T_A}), \end{aligned} \quad (7)$$

and the average reward over a renewal period is

$$\begin{aligned} E[R_i] = p_1 \int_0^{T_A} y_{r,r'}(\tau) d\tau + p_2 \int_0^{T_A} \int_0^{\tau} y_{r,r'}(h) f_2(h) dh d\tau \\ + p_3 \int_{T_A}^{\infty} \int_0^{\tau} y_{r,r'}(h) f_3(h) dh d\tau = p \frac{(1 - e^{-(2\mu+\lambda_{r,r'})T_A})}{(\mu + \lambda_{r,r'})} \end{aligned} \quad (8)$$

We can calculate the average reward over time as

$$\bar{y}_{r,r'} = \lim_{\tau\to\infty} \frac{1}{\tau} \int_0^{\tau} y_{r,r'}(h) dh = \frac{E[R_i]}{E[J_{i+1} - J_i]} \quad (9)$$

$$= p \frac{\lambda_{r,r'}(1 - e^{-(2\mu+\lambda_{r,r'})T_A})}{(\lambda_{r,r'} + \mu)(1 - e^{-(\mu+\lambda_{r,r'})T_A})} \quad (10)$$

$$= p \frac{\eta\beta \frac{k-1}{|\mathscr{R}(t)|-1}(1 - e^{-(2+\eta\beta \frac{k-1}{|\mathscr{R}(t)|-1})\mu T_A})}{(1 + \eta\beta \frac{k-1}{|\mathscr{R}(t)|-1})(1 - e^{-(1+\eta\beta \frac{k-1}{|\mathscr{R}(t)|-1})\mu T_A})}, \quad (11)$$

where (9) holds with probability one according to the renewal reward theorem [8], and (10) is obtained by substituting (7) and (8) into (9). We substitute (6) into (10) to obtain the relationship between the protocol parameters β and k and the average number of external peers. We can make three important observations based on (11).

First, (11) is a monotonically increasing concave function in β; i.e., the number of external peers increases in the willingness to multi-track with a decreasing marginal gain. Since the tracker overhead increases linearly in β (see Section IV-C2), β should not be chosen too high, like for *RPM*.

Second, similar to *RPM*, the bound is a function of the product $\eta\beta$, hence it is enough to focus on the effect of the parameter β for fixed η, as for *RPM*.

Third, (11) is a monotonically increasing concave function of k; hence choosing $k = |\mathscr{R}(t)|$ maximizes (11) for the same overhead. When $k = |\mathscr{R}(t)|$ and the peer departure rate μ increases, the average number of external peers converges to

$$\lim_{\mu\to\infty} \bar{y}_{r,r'} = p \frac{(k-1)\eta\beta}{(k-1)\eta\beta + (|\mathscr{R}(t)|-1)} = p \frac{\eta\beta}{\eta\beta + 1}, \quad (12)$$

which is equal to the lower bound obtained for *RPM* for $\nu = \mu$ in (5). Furthermore, $\bar{y}_{r,r'}$ is a monotonically decreasing, convex function of μ; hence, (12) is a lower bound for *RMT*. This seems counterintuitive at first, but can be explained by that a high peer arrival rate is needed to maintain a given torrent size under a high peer departure rate. A fraction of the arriving peers becomes multi-tracked, and hence provides good mixing.

C. Tracker Overhead

In the following we quantify the tracker overhead for the two protocols using the same notation as in Section IV-B. We show that the tracker overhead of *RPM* and *RMT* is constant independent of the number of peers.

1) Random Peer Migration (RPM): Let us consider the rate of migration in swarm r, with x_r peers and a download completion rate of ν_r. For this swarm, the instantaneous rate of peers migrating away from the swarm is

$$x_r \beta(|\mathscr{R}(t)| - 1)\frac{\nu_r}{x_r} = \beta(|\mathscr{R}(t)| - 1)\nu_r. \quad (13)$$

Since a migrating peer chooses the destination swarm uniform at random, the migration rate from swarm r to r' is $\beta\nu_r$. Similarly, the instantaneous migration rate to swarm r is

$$\frac{1}{|\mathscr{R}(t)| - 1} \sum_{r' \in \mathscr{R}(t)\setminus\{r\}} \beta(|\mathscr{R}(t)| - 1)\nu_{r'} = \sum_{r' \in \mathscr{R}(t)\setminus\{r\}} \beta\nu_{r'}. \quad (14)$$

We note that the two rates are equal if the per peer normalized throughput in the different swarms (ν_r) is equal.

With *RPM*, a peer only performs a tracker scrape if it chooses to migrate between swarms, and in this case it also performs a tracker announce. Therefore, the overhead of *RPM* is directly proportional to the willingness to migrate. The scrape rate and the announce rate per tracker and torrent is $\beta\nu(|\mathscr{R}(t)| - 1)$ due to migration. For example, if it takes on average one hour to download the content and there are two swarms, then the number of tracker announces and scrapes is β per hour. Per design, the announce and scrape rates per tracker and torrent are independent of the swarm sizes x_r.

2) Random Multi-Tracking (RMT): Let us consider the number of peers registered with k trackers. For an arrival rate of λ to the torrent when $x \geq \frac{|\mathscr{R}(t)|\beta}{k}$ the arrival rate of multi-tracked peers is

$$\lambda \frac{|\mathscr{R}(t)|\beta}{kx}. \quad (15)$$

The average peer holding time is $1/\mu$, and in steady state $x = \lambda/\mu$, so that the average number of peers registered with k trackers becomes

$$\lambda \frac{|\mathscr{R}(t)|\beta}{kx}\frac{1}{\mu} = \frac{|\mathscr{R}(t)|}{k}\beta \quad (16)$$

Every multi-tracked peer is registered with k trackers. Hence, the total tracker announce overhead is $|\mathscr{R}(t)|\beta$, and the announce overhead per tracker is β; i.e., directly proportional to the protocol parameter β. The announce overhead is independent of k by design as a consequence of the way the probability that a peer joins k trackers is selected. Unlike for *RPM*, for *RMT* the scrape overhead is λ, as all peers scrape all trackers *once*, upon arrival.

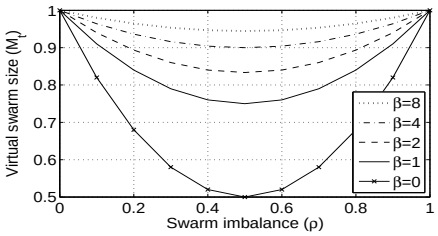


Fig. 3. Virtual swarm size vs. swarm imbalance for *RPM* for various values of $\eta\beta = \beta$, unlimited peer list, and $\mu = \nu$.

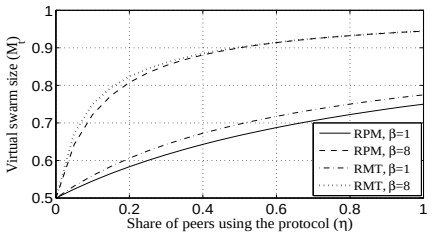


Fig. 4. Virtual swarm size vs. share of peers using the protocols. For *RPM* $\mu = \nu$, for *RMT* $k = n$ and $1/\mu = T_A$. Balanced swarms $\rho = 0.5$ and unlimited peer list.

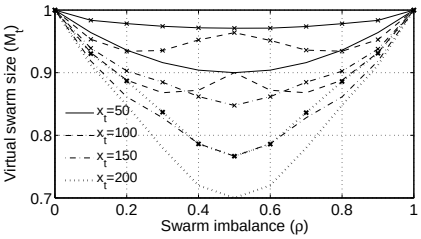


Fig. 5. Virtual swarm size vs. swarm imbalance for *RPM* with limited peer list ($p = 50$), $\eta\beta = \beta = 4$. Simulation results marked with 'x'; curves w/o markers show bound (5).

V. Numerical Results

In the following we show numerical results based on the models, and validate the analytical results via simulations and controlled experiments.

A. Analytical and Simulation Results

We start with an evaluation of the protocol parameters on the performance. To keep the number of variables low, we consider a torrent tracked by two trackers r_1 and r_2. This simplification does not affect the results significantly: for *RPM* the migration intensity between any pair of swarms of a torrent is independent of the number and size of the rest of the swarms, and for *RMT* for $k = |\mathscr{R}(t)|$ mixing is independent of $|\mathscr{R}(t)|$. Without loss of generality we set $\mu = 1$, and regulate the swarm sizes by choosing λ_r. We denote the imbalance of the swarm sizes by $\rho = \lambda_{r_1}/(\lambda_{r_1} + \lambda_{r_2})$, e.g., for $\rho = 0.5$ the two swarms have equal number of peers on average.

1) Unlimited Peer List: Ideally, peers perform gossiping with all peers in a swarm. In this case, for *RPM*, a peer migrating from swarm r' to swarm r can gossip about all the peers in swarm r', such that $p = x_{r'}$. Consequently, the average virtual swarm size is independent of the torrent size x, and can be expressed as a function of the swarm imbalance ρ by substituting (5) into (1) for both swarms and taking the weighted average. In this ideal case M is a quadratic function of ρ, and is convex because $d^2M/d\rho^2 \geq 0$ constant. Furthermore, M attains its minimum at $\rho = 0.5$ independent of the value of the other parameters. Figure 3 shows the average virtual swarm size as a function of the swarm imbalance for various values of β ($\eta\beta = \beta$ for $\eta = 1$). $\beta = 0$ corresponds to no *RPM*, and shows that the increase of the average virtual swarm size dM is highest for $\rho = 0.5$; i.e., when the average virtual swarm size is smallest.

Figure 4 shows the virtual swarm size as a function of the share η of the peers using *RPM* and *RMT*. We observe that *RMT* outperforms *RPM*, which is in accordance with the asymptotic result for $\mu \to \infty$ in (12), but the difference in terms of mixing performance is minor.

2) Limited Peer List: In practice, peers are connected to a subset of the swarm they belong to. For example, BitTorrent peers are typically not connected to more than 50 to 100 peers depending on the implementation [9]. Furthermore, peers only advertise connected peers, even if they might know the addresses of significantly more peers. In the following we show analytical and simulation results to evaluate the impact of the peer list on the virtual swarm size.

For the simulations we consider two distributions for the peer holding times. We use the exponential distribution to evaluate the tightness of the bounds in (5) and (11). Furthermore, we use the shifted Pareto distribution with distribution function $F(x) = 1 - (1 + x/b)^{-a}$, $b > 0$ and $a > 1$ [10] to evaluate the sensitivity of the results to the holding time distribution. For $a > 1$ the distribution is heavy-tailed; we use $a = 4$ and we choose the value of b such that the average holding time $1/\mu$ is matched. We are not aware of any measurement results that would provide an analytical distribution for the download times of BitTorrent peers. Hence, for *RPM* we use two distributions for the download time of the content, the exponential distribution and a normal distribution truncated at 0 with the same mean but a coefficient of variation of 0.7. The normal distribution of the download times was motivated by our experiments presented in Section V-B: using the Lilliefors test at 5-percent significance level we could not reject the hypothesis that the peers' download times came from a normal distribution. Inspection of the QQ-plots (not shown for brevity) of the download times also supported the hypothesis.

Figure 5 shows the average virtual swarm size as a function of the swarm imbalance for *RPM* for a peer list of at most 50 peers; i.e., $p = \min(50, x_{r'})$. Since in this case the virtual swarm size is not independent of the torrent size, we show results for four torrent sizes ($x = 50, 100, 150, 200$) and $\beta = 4$. With limited peer list perfect load balancing ($\rho = 0.5$) is not always the worst case scenario, but the lowest average virtual swarm size is only slightly lower than the one attained for $\rho = 0.5$. The simulation results with exponential holding time and download time distributions are better than the lower bound, as expected, but they show similar characteristics.

Though $\rho = 0.5$ is not always the worst case, the decrease of the virtual swarm size is biggest for $\rho = 0.5$ as the torrent size increases. Furthermore, if peers upon arrival choose a tracker uniform at random then $\rho = 1/|\mathscr{R}(t)|$. Figure 6(a) shows the average virtual swarm size for $\rho = 0.5$ as a function of the torrent population. The virtual swarm size starts to drop above $x_r = 50$ (i.e., $x = 100$), as the migrating peers cannot carry the addresses of all the peers of a swarm. The drop around $x_r = 50 = p$ is rather intuitive, and in general we can conclude that mixing is most efficient as long as $x_r \leq p$. Naturally, by increasing the peer list p of the clients, *RPM* can effectively

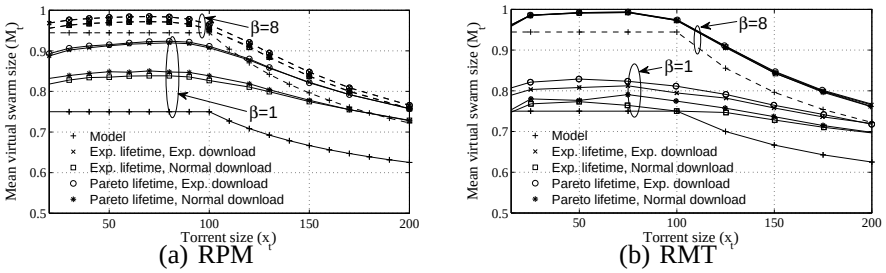


Fig. 6. Virtual swarm size vs. torrent population, when using limited peer list ($p = 50$), and various lifetime and download time distributions. Solid lines $\eta\beta = \beta = 1$; dashed lines $\eta\beta = \beta = 8$.

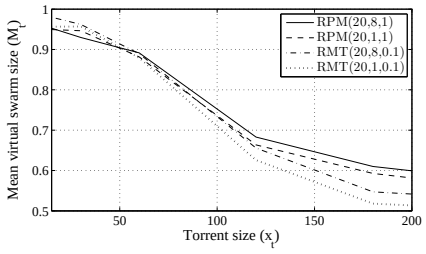


Fig. 7. Virtual swarm size vs. torrent population for *RPM*$(p, \eta\beta, \mu/\nu)$ and *RMT*$(p, \eta\beta, T_A\mu)$. Experimental results, 2 swarms, peer list ($p = 20$).

mix bigger swarms. The simulation results are insensitive to the holding time distribution, and for $\beta = 8$ the results are fairly insensitive to the download time distribution. However, the download time distribution seems to affect the results for $\beta = 1$; the smaller variance of the truncated normal distribution leads to a lower mixing efficiency for small torrents than the exponential distribution. Figure 6(b) shows the corresponding results for *RMT*, and allows us to draw similar conclusions.

B. Experimental Validation

As a proof of concept we implemented *RPM* and *RMT* in *rTorrent* [11], an open source C++ BitTorrent client. We changed the standard BitTorrent peer behavior [6], which is to pick a tracker uniform at random (*pick-one*) to our protocols. For the gossip protocol we relied on the Peer Exchange Protocol (PEX) [6], which is supported by the most recent versions of almost all popular BitTorrent clients [7]. Since the implementation of *RPM* and *RMT* is transparent to the trackers, we could use Opentracker [12] as tracker software.

We performed controlled experiments on a cluster of 5 hosts to validate the model. In our experiments peers arrived according to a Poisson process to a torrent, downloaded a content of 50*MB* (file size for a shorter TV episode), and departed upon download completion (i.e., $\nu/\mu = 1$). We limited the upload rates of the peers to $80KB/s$, and the peer list was $p = 20$. The announce period was set to $T_A = 60s$, so that every peer performed approximately 10 announces on average. Every experiment lasted for two hours, and the results shown are the averages of three to six experiments. We monitored the peers' neighbor lists to calculate the average virtual swarm size. Furthermore, we measured the amount of PEX traffic sent by every peer, and the amount of tracker traffic.

Figure 7 shows the average virtual swarm size as a function of the torrent size for the case of two swarms. The results for the two protocols closely resemble the analytical and simulation results, and show that the models capture the mixing of the swarms rather well. Figure 8 shows the number of announce requests sent to the tracker when using *RPM* or *RMT* divided by the number of requests sent when using *pick-one* (lines w/o marker). Surprisingly, for small torrents the number of announce requests is less than using *pick-one*. This is because peers finish downloading the content faster as an effect of mixing the small swarms. For larger torrents the announce overhead approaches 1, and is slightly higher for *RMT* than for *RPM*. The figure also shows the number of scrape requests sent per peer and tracker (lines w. marker). The scrape overhead for *RMT* is 1 (1 scrape upon peer arrival), while for *RPM* it converges to 0 as the torrent size increases. The higher overhead of *RMT* is the price of the better mixing performance compared to *RPM*.

Figure 9 shows the average virtual swarm size and the normalized PEX traffic as a function of the torrent size for the case of three swarms. Comparing Figures 7 and 9 we observe that the increase of the virtual swarm size (dM_t) due to mixing is nearly constant if we compare the results for equal swarm sizes $x_t/|\mathscr{R}(t)|$. The amount of PEX traffic increases with a decreasing marginal gain as a function of the torrent size, which is a sign that $p = 20$ limits the mixing of the swarms as the torrent size increases. Unlike the tracker overhead, the PEX traffic is not directly proportional to the protocol parameter β.

Figure 10 shows the average virtual swarm size and the normalized PEX traffic for a torrent of $x_t = 100$ peers tracked by 1 to 8 trackers. The virtual swarm size for both *RPM* and *RMT* decreases with a decreasing marginal rate, but is relatively high even for 8 swarms: an average peer knows more than 50% of all peers in the torrent (as opposed to 12.5% if all peers follow the *pick-one* scheme).

The figure also shows that the amount of PEX traffic decreases as the number of swarms increases. In general, the *amount of PEX traffic is negligible* compared to the peer upload capacities; this is the reason why we did not analyze the gossip protocol's overhead in Section IV. Figure 11 shows the frequency of announce and scrape requests sent to the trackers (in total) for the same experiment. The figure validates the analytical model: the request rate increases linearly in the number of swarms $|\mathscr{R}(t)|$ for given torrent size. However, the rate of increase is much smaller than for *pick-all*, which would correspond to a curve of unit slope.

VI. BitTorrent Case study

RPM and *RMT* could be deployed incrementally in BitTorrent without any changes to the existing tracker infrastructure. In the following we estimate the performance improvement that the proposed overlay management protocols could achieve, if deployed, based on BitTorrent measurements.

A. Empirical Data Set

We used two kinds of measurements to obtain our data set. First, we performed a screen-scrape of the torrent search engine *www.mininova.org*. In addition to claiming to be the

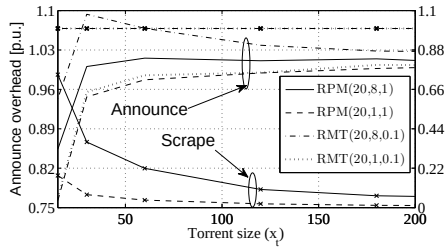


Fig. 8. Announce and scrape overhead vs. torrent population for *RPM*$(p,\eta\beta,\mu/\nu)$ and *RMT*$(p,\eta\beta,T_A\mu)$. Experimental results, 2 swarms, peer list ($p=20$).

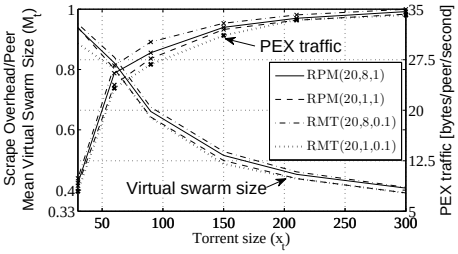


Fig. 9. Virtual swarm size and PEX traffic vs. torrent population for *RPM*$(p,\eta\beta,\mu/\nu)$ and *RMT*$(p,\eta\beta,T_A\mu)$. Experimental results, 3 swarms, peer list ($p=20$).

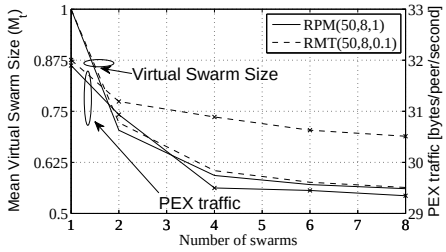


Fig. 10. Virtual swarm size and PEX traffic vs. number of swarms for *RPM*$(p,\eta\beta,\mu/\nu)$ and *RMT*$(p,\eta\beta,T_A\mu)$. Experimental results, $x_t=100$ peers, peer list ($p=20$).

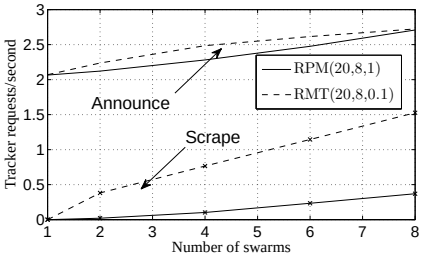


Fig. 11. Tracker overhead vs. number of swarms for *RPM*$(p,\eta\beta,\mu/\nu)$ and *RMT*$(p,\eta\beta,T_A\mu)$. Experimental results, $x_t=100$ peers, $p=20$.

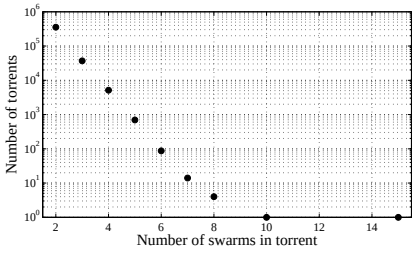


Fig. 12. Number of swarms per torrent for Oct. 10, 2008. More than 300 thousand torrents consist of two swarms.

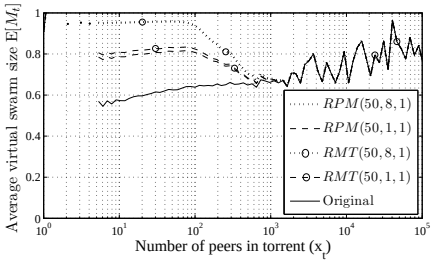


Fig. 13. Average virtual swarm size as a function of the number of peers for *RPM*$(p,\eta\beta,\mu/\nu)$ and *RMT*$(p,\eta\beta,T_A\mu)$.

largest torrent search engine, *mininova* was the most popular torrent search engine according to *www.alexa.com* during our measurement period (Alexa-rank of 75, August 1, 2008). From the screen-scrapes we obtained the sizes of about $330,000$ files shared using BitTorrent, and the addresses of $1,690$ trackers. Second, we scraped the $1,690$ trackers to obtain the number of leechers, seeds, and completed downloads for the torrents they track.

We performed the tracker-scrapes daily from October 10, 2008, to October 17, 2008 as part of an 11 months long measurement campaign [13]. All scrapes were performed at 8pm GMT. We performed correlation tests to identify and remove redundant tracker-scrapes (for example, from trackers with multiple hostnames) [14]. This way we removed redundant information about the same swarms of peers, and identified 721 independent trackers.

During the measurements we observed $40-60$ million active peers on a weekly basis. These peers downloaded more than 8 billion copies of over 10 million torrents in 48 weeks. Our measurement data shows that there is a substantial number of torrents with moderate popularity [13]. For example, about 2.84 million of the 2.86 million torrents observed on October 10, 2008 have less than 200 peers, and about 50% of the peers are in these torrents. Figure 12 shows the number of torrents with a given number of unique swarms (after removing duplicates). There are a substantial number of torrents that are served independently by multiple trackers. Out of these, a significant portion was shown to benefit from merging [14].

B. Mixing Efficiency

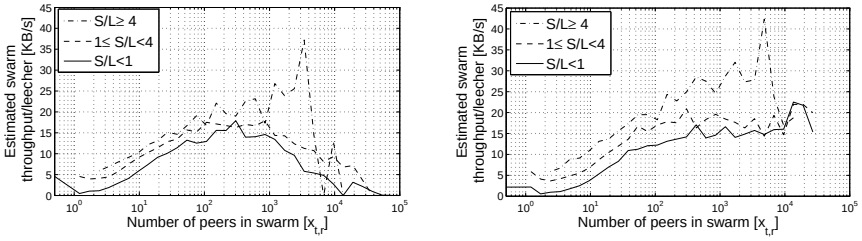
To evaluate the performance of *RPM* and *RMT*, we used the lower bound in (5) and (11), respectively, to estimate the average virtual swarm size M_t based on the number of peers in each swarm, for all multi-tracked torrents in the data set.

Figure 13 shows the average virtual swarm size as a function of the number of peers in the torrent for October 10, 2008 for *RPM* and *RMT*. The figure shows results for the original peer allocation and for various values of β (we set $\eta=1$, so $\eta\beta=\beta$ as discussed in Section III). Peer lists with $p=50$ peers are assumed. We note that both *RPM* and *RMT* increase the virtual swarm size significantly even for $\beta=1$, and the marginal gain of increasing β decreases. While for $\beta=1$ *RMT* slightly outperforms *RPM*, for $\beta=8$ the results are indistinguishable. As expected from Figure 6, the effect on the virtual swarm size is negligible above torrent sizes of a few hundred peers. However, for torrent sizes smaller than 100 peers, the proposed protocols consistently achieve at least 85% virtual swarm size when $\beta=1$ ($\geq 95\%$ when $\beta=8$).

C. Swarm Throughput

In the following we estimate the throughput of different sized swarms based on the large-scale data set. Then we use the throughput estimates to estimate the potential performance gains small torrents can obtain using our protocols.

1) Throughput Estimation: To estimate the throughput of any particular swarm we measured the number of seeds (S) and leechers (L) in the torrents, as well as the cumulative number of downloads (D) between two consecutive measurement times T_1 and T_2, separated by one day. Using these values we estimated the average throughput per leecher as the file size B divided by the estimated download (service) time $1/\nu$. Using Little's law $1/\nu=L/\lambda$, where L is the average number of leechers currently being served in the system and the arrival rate λ can be estimated as the number of download



(a) Starting 8pm GMT, Oct. 10, 2008. (b) Starting 8am GMT, Oct. 11, 2008.

Fig. 14. Throughput estimates for three classes of swarms based on 24 hours of measurement data. The throughput increases with the swarm size.

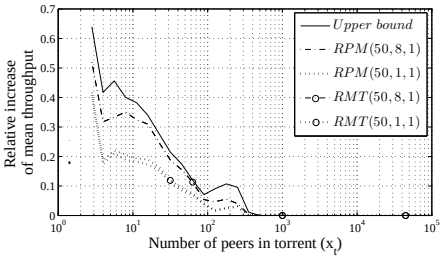


Fig. 15. Estimated speedup vs. torrent size after applying *RPM*$(p,\eta\beta,\mu/\nu)$ and *RMT*$(p,\eta\beta,T_A\mu)$.

completions D divided by the time (T_2-T_1) between the two consecutive measurements. To summarize, we have an estimated throughput of $\frac{BD}{L(T_2-T_1)}$. Finally, throughput estimates for any particular swarm type were obtained by taking the average over all swarms of that particular size.

Figures 14(a) and 14(b) show throughput estimation results based on a 24 hour interval starting at 8pm GMT on October 10, 2008 and at 8am GMT on October 11, 2008, respectively. Swarms are classified based on the total number of peers in the swarm $(S+L)$ and the seed-to-leecher ratio $\frac{S}{L}$; 60 bins are used for the swarm sizes and three curves are used for the different seed-to-leecher ratios. We note that the results for swarms up to just over $1,000$ peers are similar in both figures and are consistent with intuition and previous studies [15], [16].

For larger swarm sizes, however, the results in Figure 14(a) are surprising. We attribute the seemingly decreasing throughput to two factors. First, the estimation for large swarms is less accurate due to fewer samples. For example, while the number of swarms smaller than $1,000$ peers for which we could obtain content size information from *mininova.org* is $282,827$, there are only 861 swarms that are larger than $1,000$ peers. The lack of statistical significance for swarm sizes above $1,000$ peers is illustrated by the large fluctuation of the curves in the figures. Second, estimation errors may be due to inaccurate estimation of the average number of leechers L over the 24 hour measurement period. The popularity of these types of workloads typically follows a diurnal pattern. The measurement at 8pm GMT was done at peak hours, consequently the throughput estimate is pessimistic. The measurement at 8am GMT was done when swarms are smallest, and the estimate is hence optimistic. (This effect is captured by the almost consistently higher throughput estimates in Figure 14(b), relative to those in Figure 14(a).) The difference between the two throughput estimates for large swarms suggests that the popularity of large swarms may show heavier diurnal fluctuation than that of small swarms.

2) Estimated Throughput Improvements: For swarms below $1,000$ peers the two estimates coincide well, and we use the throughput estimates from 8pm GMT to estimate the speedup achieved by the proposed algorithms. The *upper bound* shown in the figures corresponds to perfect mixing (i.e., $M_t = 1$), which is the case if all peers follow the *pick-all* scheme.

Figure 15 shows the throughput improvement for leechers as a function of the torrent size for *RPM* and *RMT* for different

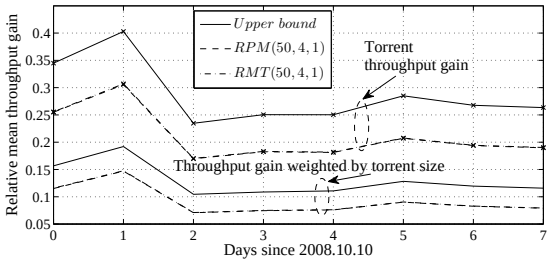


Fig. 16. Estimated speedup for torrents smaller than 300 peers vs. time.

values of the migration willingness β (again $\eta = 1$). The figure shows a pessimistic estimate of the throughput improvement for three reasons: (i) we used the analytical model to estimate the average virtual swarm size given the swarm sizes $x_{t,r}$, (ii) the peer list was limited to 50 peers, and (iii) the mixing model assumed that peers on average depart upon finishing the download of the content (i.e., $\mu = \nu$). In accordance with Figure 13, for $\beta = 1$ *RMT* slightly outperforms *RPM*, but for $\beta = 8$ they achieve the same gain. The gain for $\beta = 8$ is close to the upper bound, hence we conclude that a relatively low intensity of peer migration can lead to close to maximal gains. This observation coincides with the observations made in Section III-A about the effect of β. The results also indicate that the peer list limit of $p = 50$ does not significantly decrease the benefits of *RPM* for small torrents. With both *RPM* and *RMT* the throughput is typically increased by 40% (30%) or more on average when torrents have less than 10 peers, and by 10% (5%) or more on average when torrents have less than 200 peers. Torrents above approximately 200 peers are not affected by the protocols, but by increasing p one could increase the virtual swarm size for larger torrents as well.

Figure 16 shows the relative estimated throughput improvement for leechers in torrents smaller than 300 peers over a week. The curves marked with $\times$ show the non-weighted gain; the curves without markers show the gain weighted with the torrent sizes. The throughput gains are rather steady. For example, the average torrent with less than 300 peers sees an increase in throughput by 25% on average (and the average peer in such a torrent sees a throughput increase by 12%).

VII. Related Work

The efficiency of P2P content distribution typically improves with the number of peers in the swarm that can share data with others [14]–[17]. Sharing opportunities can

be improved by using efficient peer discovery techniques; e.g., by retrieving peer information from trackers, and by gossiping with other peers, as discussed in the paper. Distributed hash tables (DHTs) [1] and unstructured overlays [3] were proposed for the purpose of highly available peer-discovery. However, it has proven challenging for DHTs to maintain consistency and avoid stale routing tables under node churn at a low overhead, ultimately affecting the performance of DHT-based peer-discovery in practice [1], [2], [18]. With unstructured overlays, good peer discovery typically comes at the cost of high overhead [3], [18].

Peer-discovery in BitTorrent was the focus of [2], [7]. Neglia *et al.* [2] found that multiple trackers significantly improve the system availability, but at the same time can reduce the connectivity of the overlay. Wu *et al.* [7] found that using PEX provided good peer-discovery in BitTorrent and could lead to decreased download times. The authors in [14] proposed a tracker-based protocol to manage the swarms of a torrent dynamically, and showed its potential benefits based on measurements of BitTorrent content popularity.

A number of papers have considered improving content availability [16], [19]–[22] through cooperation across torrents (each with different content). Menasche *et al*. [16] showed that "bundling" multiple files into a larger file can improve the content availability. Other works assign a "helper" file to clients to assist [19]–[21]. Yang *et al.* [22] proposed rate-based incentives that motivate clients to act as seeds for other torrents than they currently are downloading. The goal of these works was to improve content availability through increased cooperation between peers sharing different contents.

Our work is complementary to the above works. Our focus is on exploring the potential of hybrid peer-discovery mechanisms, which provide highly available and efficient peer-discovery, based on multiple trackers and a gossip protocol. The distributed protocols we propose aim to mix otherwise disjoint swarms of the same torrent at a very low overhead. Their simplicity makes the proposed protocols easy to deploy both in systems with distributed trackers, such as BitTorrent and other managed P2P content distribution systems.

VIII. Conclusion

In this work we considered a hybrid approach to peer-discovery for P2P systems, which combines the advantages of centralized and distributed peer-discovery mechanisms. We proposed two novel protocols, *RPM* and *RMT*, that rely on independent trackers and a gossip protocol only and have a constant communication overhead independent of the torrent size. Both protocols employ a small portion of the peers to provide mixing between the swarms, and are very simple to implement in existing systems, such as BitTorrent.

We developed analytical models of the protocols based on the theory of renewal-reward processes and used them to give a fundamental understanding of the protocol performance. We validated the analytical model via extensive simulations and controlled experiments. Our results show that independent trackers and a gossip protocol are sufficient to provide highly available and efficient peer-discovery at low overhead. As an example of the potential value of these protocols, we performed a measurement-based protocol evaluation. Using large-scale measurement data from a large set of BitTorrent trackers, we showed that the proposed protocols could lead to a significant improvement in terms of peer throughput at a very low overhead. While our experimental protocol evaluation is done in the context of BitTorrent, the protocols are in general applicable to tracker-based overlay management systems.

IX. Acknowledgement

The authors thank Anil Can Akay for his help in the experimental evaluation. The work was partially funded by the ACCESS Linnaeus Centre and by CENIIT.

References

[1] J. Falkner, M. Piatek, P. John, A. Krishnamurthy, and T. Anderson, "Profiling a million user DHT," in *Proc. ACM IMC*, Oct. 2007.

[2] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems," in *Proc. IEEE INFOCOM*, May 2007.

[3] M. Castro, M. Costa, and A. Rowstron, "Debunking some myths about structured and unstructured overlays," in *Proc. of USENIX NSDI*, 2005.

[4] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurements study of a large-scale p2p iptv system," *IEEE Trans. on Multimedia*, vol. 9, no. 8, 2007.

[5] Akamai Netsession Interface, Nov. 2010. [Online]. Available: http://www.akamai.com/client

[6] BitTorrent Extension Protocols (BEP-0011 and BEP-0012), Nov. 2010. [Online]. Available: http://www.bittorrent.org

[7] D. Wu, P. Dhungel, X. Hei, C. Zhang, and K. Ross, "Understanding peer exchange in bittorrent systems," in *Proc. IEEE P2P*, 2010.

[8] S.M. Ross, *Stochastic Processes, 2nd ed.* John Wiley & Sons, 1996.

[9] A. Al-Hamra, N. Liogkas, A. Legout, and C. Barakat, "Swarming overlay construction strategies," in *Proc. IEEE ICCCN*, Aug. 2009.

[10] D. Leonard, Z. Yao, V. Rai, and D. Loguinov, "On Lifetime-based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks," *IEEE/ACM Trans. on Networking*, vol. 15, no. 3, 2007.

[11] rTorrent BitTorrent Client, Nov. 2010. [Online]. Available: http://libtorrent.rakshasa.no

[12] Opentracker Home Page, Nov. 2010. [Online]. Available: http://erdgeist.org/arts/software/opentracker/

[13] G. Dán and N. Carlsson, "Power-law revisited: A large scale measurement study of P2P content popularity," in *Proc. International Workshop on Peer-to-peer Systems (IPTPS)*, Apr. 2010.

[14] G. Dán and N. Carlsson, "Dynamic Swarm Management for Improved BitTorrent Performance," in *Proc. International Workshop on Peer-to-peer Systems (IPTPS)*, Apr. 2009.

[15] X. Yang and G. de Veciana, "Service Capacity of Peer-to-Peer Networks," in *Proc. IEEE INFOCOM*, Mar. 2004.

[16] D. Menasche, A. Rocha, B. Li, D. Towsley, and A. Venkataramani, "Content Availability in Swarming Systems: Models, Measurements and Bundling Implications," in *Proc. ACM CoNEXT*, Dec. 2009.

[17] D.Menasche, A. Rocha, E. Silva, R. Leao, D. Towsley, and A. Venkataramani, "Estimating self-sustainability in peer-to-peer swarming systems," in *Proc. IFIP Performance*, Nov. 2010.

[18] Y. Qiao and F. Bustamante, "Structured and unstructured overlays under the microscope," in *Proc. USENIX ATC*, 2006.

[19] R. S. Peterson and E. G. Sirer, "AntFarm: Efficient Content Distribution with Managed Swarms," in *Proc. NSDI*, May 2009.

[20] N. Carlsson, D. L. Eager, and A. Mahanti, "Using Torrent Inflation to Efficiently Serve the Long Tail in Peer-assisted Content Delivery Systems," in *Proc. IFIP/TC6 Networking*, May 2010.

[21] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurement, Analysis, and Modeling of BitTorrent-like Systems," in *Proc. ACM IMC*, Oct. 2005.

[22] Y. Yang, A. L. H. Chow, and L. Golubchik, "Multi-Torrent: A Performance Study," in *Proc. IEEE MASCOTS*, Sept. 2008.