Design and Analysis of ISP-friendly File Distribution Protocols

Minghong Lin* John C.S. Lui* Dah-Ming Chiu+
The Chinese University of Hong Kong
*Computer Science & Engineering Department
+Information Engineering Department
Email: {mhlin,cslui}@cse.cuhk.edu.hk, dmchiu@ie.cuhk.edu.hk

***Abstract*— In recent years, BitTorrent-like file-swarming applications are becoming so popular that they contribute to a large percentage of the current Internet traffic. Internet Service Providers (ISPs) not only need to cope with this traffic engineering problem, e.g., when and how to increase their network capacity, but more importantly, these P2P applications also increase their operating cost since large amount of the traffic has to go through the cross-ISP links. In this paper, we consider the design and analysis of an ISP-friendly file swarming protocol so as to reduce the cross-ISP traffic. We analytically show that the conventional P2P file-swarming protocols consume significant bandwidth on the cross-ISP links. We also derive an upper and lower bound for the cross-ISP traffic for ISP-friendly protocols which exploit the data locality property. We propose and implement an ISP-friendly protocol and carried out large scale experiments on the PlanetLab. Experimental results indicate that our protocol can significantly reduce the cross ISP-traffic and provide a reasonable file downloading time.**

I. Introduction

The P2P computing paradigm is a technology that can provide scalable services on the Internet. This can be observed by the wildly popular services such as content distribution via BitTorrent [6], VoIP via Skype, and IPTV services via PPLive [8]. Unlike the traditional client-server or fixed-infrastructure content distribution (e.g., Akamai), P2P technology has the self-scaling property: as the demand increases, so is the service capacity.

Due to the popularity of P2P applications, in particular, file-swarming applications which use the BitTorrent protocol, introduce some challenging issues. Studies show that P2P applications, including BitTorrent, account for over 60% of the traffic seen by an ISP [4]. Worse yet, pre-dominant of the traffic goes through the cross-ISP links since these applications do not distinguish between ISPs' boundaries. This not only presents significant traffic-engineering challenges to ISPs, but the large volume of cross-ISP traffic also implies an increase congestion level and more importantly, the operating cost of ISPs.

To address the above problem, ISPs have several options. For example, ISPs can rate limit the file swarming traffic, or perform packet inspection on traffic across the cross-ISPs links. However, these are not an effective solutions since applications can always use dynamic port, or encrypt their payload so as to bypass detection. Also, rate limiting an application will discourage users within an ISP and these users may choose to select another ISP for connectivity service. Another approach is for an ISP to perform content caching so as to limit the cross-ISP traffic. However, caching can be complicated since ISP needs to accurately determine which file to cache or replace. In short, caching requires additional investment on caching infrastructure, and also introduces legal and copyright problems.

The aim of this paper is to propose an *ISP-friendly file swarming protocol*. The objectives of the protocol are to reduce the cross-ISP traffic and to maintain reasonable file downloading performance. What is more, this is purely an end-system approach and we do not need to invest on any infrastructure to provide such kind of service. The protocol relies on a simple idea: from each peer's point of view, all other peers could be classified into two categories: *internal peers* and *external peers*. Internal peers are those peers which belong to the same ISP as itself while external peers are of different ISPs. To reduce cross-ISP traffic, each peer tries to use the "*exploiting-the-locality principle*" (ELP) as much as possible. The ELP follows a very simple rule: never download information from external peers if there exist at least one copy of the information among the internal peers.

The contributions of this paper are:

- We mathematically derive the average cross ISP-traffic for a conventional file-swarming protocol (e.g., BitTorrent).
- We *analytically quantify* the merits when file-swarming protocols follow the ELP. In particular, we derive the lower and upper bounds of incoming cross-ISP traffic when peers arrival is characterized by a Poisson process.
- We implement an ISP-friendly protocol on existing BitTorrent client software. We carry out experiments and measurements on PlanetLab to demonstrate significant cross-ISP traffic reduction and good file download performance of our ISP-friendly protocol.

The balance of our paper is as follows. In Section II, we present the mathematical models and derive the upper and lower bound of cross-ISP traffic when the file swarming protocol follows the ELP. In Section III, an ISP-friendly file-swarming protocol is presented. In Section IV, we present our measurement results on the Planetlab. Related work is given in Section V and finally, conclusion is given in Section VI.

II. MATHEMATICAL MODELS

We consider a P2P file-swarming system which distributes files to a large number of peers. The file to be distributed, say $\mathcal{F}$, is divided into many chunks. Formally, we have $\mathcal{F} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K\}$ in which the file $\mathcal{F}$ has $K \geq 1$ chunks, $\mathcal{C}_i$ is the i^{th} chunk of $\mathcal{F}$ and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for $i \neq j$. A peer that holds all chunks of the file is called a *seeder* while a peer that holds a subset of chunks is called a *leecher*. To download the file, a peer (or leecher) needs to download all K chunks.

Let us first consider a BT-like file swarming system which does not consider the boundary between ISPs in their data transfer. We call such kind of P2P file swarming as "*random downloading*". What we are interested in is the amount of cross-ISP traffic. Assume that the number of peers in the P2P system is N, n of which are within the ISP $\mathcal{A}$. For a randomly chosen peer which resides in the ISP $\mathcal{A}$, the average fraction of file content which are downloaded from peers outside ISP $\mathcal{A}$ is:

$$f = 1 - \frac{n}{N}. \quad (1)$$

This is also the probability of choosing a peer outside ISP a for the data transfer[1]. Thus, the total amount of incoming cross-ISP traffic is approximately ($n(1 - \frac{n}{N}) * \text{file size}$). This represents a large volume of cross-ISP traffic because usually there are many peers in a P2P file swarming system, for instance, $N >> n$ and n is relatively large.

In analyzing the performance of an ISP-friendly protocol, we seek to derive the amount of cross-ISP traffic if peers are willing to follow the *exploiting-the-locality principle* (ELP). Only when the reduction of cross-ISP traffic is high, then one should consider designing and implementing an ISP-friendly file swarming protocol. In this paper, we consider the case that peers arrival process is characterized by a Poisson process.

A. Assumptions

Unlike previous work which focused on the performance modeling of file downloading time, we model the the amount of cross-ISP traffic. For our mathematical model, we make the following assumptions:

- Peers arrival process is characterized by a Poisson process with an average rate λ.
- Peers are all *persistent* in the sense that they will not abort before they finish the file download.
- To ensure file availability, we assume there exists at least one *seeder* in the system: some peers are willing to publish the original file to the P2P network.
- Whenever a peer (or leecher) obtains all chunks of a file, the peer will leave the system immediately.
- The downloading rate of a peer is *relatively constant* during the progress of its file download. Nevertheless, we allow different peers to have different downloading rate. This assumption will be useful in our analysis.

[1]Since a download service is reciprocated by an upload service, the fraction of file content which are uploaded by a peer in ISP a is approximately equal to Eq. (1). We will verify this statement in Sec.IV.

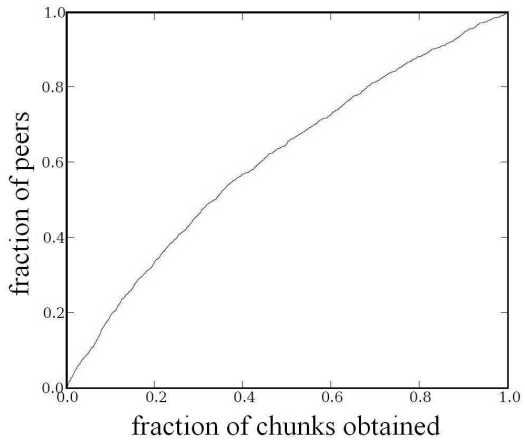


Fig. 1. CDF of the types of peers in a BitTorrent measurement

Note that the last assumption is usually made in many fluid models of P2P systems [7], [18]. To investigate the validity of the last assumption, we carried out measurement study on a file swarming system. The data were extracted during the steady downloading phase of the movie "Beyond Good and Evil" on December, 2003 through BitTorrent/Suprnova.org [1]. All peers (about 1500 peers) are classified based on the amount of chunks they obtain, e.g., a peer is of type i when it obtains i chunks. Figure 1 illustrates the cumulative distribution function (CDF) of types of peer (without seeders). The figure shows that the distribution of the types of peer is *roughly uniform*. In other words, let $E[N_i]$ be the average number of peers which is of type i, then

$$E[N_1] \approx E[N_2] \approx \cdots \approx E[N_K].$$

When peers are persistent or when the abort rate is low, a peer starts with type 1, then becomes type 2, and eventually becomes type K. Therefore, the average arrival rates of type i, for $i \in \{1, \ldots, K\}$, are relatively the same. Using the Little's result, we have:

$$E[T_1] \approx E[T_2] \approx \cdots \approx E[T_K],$$

where $E[T_i]$ is the sojourn time of being a type i peer. This expression implies that the downloading rate of a peer is relatively constant and independent on the progress of its file download and that our last assumption is reasonable.

When one uses the ELP, if there exists a seeder in an ISP, then all peers in that ISP will never download chunks from external peers and the incoming cross-ISP traffic is zero. This is a trivial case. We consider a more interesting case wherein the seeder does not reside in an ISP. The derivation of the cross-ISP traffic for an ISP-friendly protocol is complicated and it depends on the specific implementation of the protocol, but instead, one can derive close upper and lower bound of this measure. Before we present the formal analysis, let us use an example to illustrate the idea. Assume $\mathcal{F}$ is the file we want to distribute and $\mathcal{F}$ has 20 chunks. At the current

time, there are three peers within the same ISP. Let v_i be the fraction of progress in the file download for peer i. For this example, we have $v_1 = 0.3$ (six chunks), $v_2 = 0.15$ (three chunks) and $v_3 = 0.2$ (four chunks). When peers follow the ELP, only those missing chunks by all peers would be downloaded through the cross-ISP link. How many chunks would be downloaded through the cross-ISP link before the next peer departure? In the best case, when all peers possess *different* chunks from each others, then the total external download will be $d = 1 - \sum_{i=1}^{3} v_i = 0.35$ (seven chunks) and this is the lower bound of the cross-ISPs traffic. In the worst case, the set of chunks possessed by any peer is a subset of the set of chunks possessed by the peer with the maximum progress. For our example, let say that peer 1 has the maximum progress, therefore, we need to download all missing chunks of peer 1 and it is equal to $d = 1 - \max_{i=1}^{3}\{v_i\} = 0.7$ (or 14 chunks), which is the upper bound of the cross-ISP traffic. The remaining question is how to uncondition the number of peers and v_i's. We are now in the position to develop the upper and lower bound of the cross-ISPs traffic.

To simplify our analysis, we consider the homogeneous case: all peers have the *same* downloading rate. Thus, the file downloading time is the same for all peers. Without loss of generality, assume the file size is 1 and the file downloading time is T. We have the following result.

Theorem *1:* For a given ISP in which all peers use an ISP-friendly file swarming protocol, if there is no seeder in that ISP, peers arrival process is characterized by a Poisson process with an average rate λ and all peers in that ISP have the same downloading rate $1/T$, then the average amount of incoming cross-ISP traffic caused by each peer in the steady state, denoted by $E(d)$, is lower bounded by

$$E(d) \geq e^{-\rho}\rho^{-1/2}I_1(2\sqrt{\rho}),$$

where $\rho = \lambda T$ and $I_1(x)$ is the modified Bessel function.

Note that $\rho = \lambda T$ is the average number of peers in that ISP, and this lower bound is a *decreasing* function of ρ.

Proof: Due to the self scaling property of P2P systems, the service capacity of the system is proportional to the number of peers. Therefore, one can model the file-swarming P2P system within this ISP as an $M/D/\infty$ queueing system with arrival rate λ and service time T.

Let p_n denote the probability that there are n peers in the ISP. Since the service time is T, the probability that there are n peers in the ISP is equal to the probability that there are n arrivals between time $[t-T, t]$. Since the number of peers arriving in a time interval of length T is Poisson distributed with mean λT, we immediately obtain

$$p_n = \frac{(\lambda T)^n}{n!}e^{-\lambda T} = \frac{\rho^n}{n!}e^{-\rho} \qquad n = 0, 1, \ldots.$$

The above statement is valid for all $t > T$, and thus also for the limiting distribution.

Now consider when these peers have to download content from external peers, e.g., peers which belong to *other ISPs*. Assume that there are n peers within this ISP at a certain time. Let v_i denote the fraction of file content that peer i has obtained so far. Since the size of the file is 1, we have $v_i < 1$ for $i \in \{1, \ldots, n\}$. If $v = \sum_{i=1}^{n} v_i < 1$, then these n peers need to download *at least* $(1 - v)$ fraction of the file content from external peers before the next peer departure from this ISP.

We use the method of the *imbedded Markov Chain* [11] and select the departure points as our observation points. Since the arrival is a Poisson process, we have

$$p_n = \text{Prob}\left(\text{departure leaves } n \text{ peers in the systems}\right).$$

When a peer departs and observes that there are n peers within this ISP with $v = \sum_{i=1}^{n} v_i < 1$, then this ISP needs to consume at lease $(1 - v)$ of incoming cross-ISP traffic before the next peer departure.

When there are exactly n arrivals from a Poisson process in $[0, t]$, the unordered arrival times are uniformly, independently distributed over $[0, t]$. In our system, it means that all these n downloading progresses are uniformly, independently distributed over $[0, 1]$. Formally, let X_i be the random variable denoting v_i, we have $X_i \sim U[0, 1], i = 1, \ldots, n$. We are interested in $Y_n = \sum_{i=1}^{n} X_i$ and its corresponding density function $f(v|n)$. To derive Y_n and $f(v|n)$, one can use Laplace transformation method:

$$X_i(s) = \frac{1}{s}\left(1 - e^{-s}\right)$$

$$Y_n(s) = \prod_{i=1}^{n} X_i(s) = \frac{1}{s^n}(1 - e^{-s})^n = \frac{1}{s^n}\sum_{j=0}^{n} C_n^j(-1)^j e^{-js}.$$

Thus

$$\begin{aligned} f(v|n) &= \sum_{j=0}^{n} C_n^j(-1)^j \frac{(v-j)^{n-1}}{(n-1)!}u(v-j) \\ &= \sum_{j=0}^{\lfloor v \rfloor} C_n^j(-1)^j \frac{(v-j)^{n-1}}{(n-1)!} \end{aligned}$$

Focusing on the range $0 \leq v < 1$, we have

$$f(v|n) = \frac{v^{n-1}}{(n-1)!}, \qquad 0 \leq v < 1, n = 1, 2, \ldots.$$

Let d denote the incoming cross-ISP traffic between two consecutive peers departures. Since these n peers need to download at least $(1 - v)$ fraction of the file through the cross-ISP link before the next peer departure, we have

$$E(d|n) \geq \int_0^1 (1 - v)f(v|n)dv = \frac{1}{(n+1)!}, \quad n = 1, 2, \ldots.$$

Now consider the case $n = 0$. When a departing peer observes that there is no peer in the ISP, this means that new arriving peers need to download exactly one copy of the file via the cross-ISP link before the next peer departure. Thus

$$E(d|0) = 1 = \frac{1}{(0+1)!}.$$

Given $E(d|n)$ and p_n, one can derive $E(d)$, the average cross-ISP traffic caused by each departure.

$$\begin{aligned} E(d) &= E(E(d|n)) = \sum_{n=0}^{\infty} p_n E(d|n) \\ &\geq \sum_{n=0}^{\infty} \frac{\rho^n}{n!} e^{-\rho} \frac{1}{(n+1)!} = e^{-\rho} \rho^{-1/2} I_1(2\sqrt{\rho}) \end{aligned}$$

where $I_1(x)$ is the *modified Bessel function*. ∎

Remark: The above theorem provides a lower bound of the average fraction of cross-ISP traffic for each departing peer. Note that this lower bound is a decreasing function of ρ. In other words, when there are more peers in the ISP, there may be a higher reduction in the cross-ISP traffic.

Assume that each peer is aware of other peers' state in real time and information can be obtained instantaneously, then for a P2P system which uses the exploiting-the-locality principle, one can derive an upper bound of the average cross-ISP traffic as follows.

Theorem *2:* For a given ISP in which all peers use an ISP-friendly file swarming protocol, if there is no seeder in that ISP, peers arrival process is characterized by a Poisson process with an average rate λ and all peers in that ISP have the same downloading rate $1/T$, then the average amount of incoming cross-ISP traffic caused by each peer in the steady state, denoted by $E(d)$, is upper bounded by

$$E(d) \leq \frac{1}{\rho}(1 - e^{-\rho}),$$

where $\rho = \lambda T$.

Proof: Similar to the $M/D/\infty$ formulation in the proof of Theorem 1, one can use the method of the *Imbedded Markov Chain* and select the departure points as the observing points.

Consider the situation that a peer departs and observes that there are n peers within this ISP. The progress of these n peers are uniformly and independently distributed over $[0, 1]$, i.e., $X_i \sim U[0, 1], i = 1, \ldots, n$. Consider the peer whose progress is maximal. According to the ELP, those content held by this peer would not generate any cross-ISP traffic before the next peer departure. On the other hand, those content that are not being held by this peer may cause a data transfer over the cross-ISP link before the next peer departure (the content may be held by other internal peers). To derive the upper bound, we ignore the collision of two or more peers request the same chunk from external peers at the same time. We consider $Z_n = \max_{i=1}^{n} X_i$ and its corresponding density function as $g(v|n)$.

Since $\text{Prob}(Z_n \leq v) = \prod_{i=1}^{n} \text{Prob}(X_i \leq v)$, we have

$$g(v|n) = nv^{n-1}, \qquad 0 \leq v \leq 1, n = 1, 2, \ldots$$

This requires at most $(1 - v)$ fraction of the file via the cross-ISP link before the next peer departure. We have

$$E(d|n) \leq \int_0^1 (1 - v) g(v|n) dv = \frac{1}{n+1}, \qquad n = 1, 2, \ldots$$

Consider the case that $n = 0$. When a departing peer observes that there are no peer in the ISP, the new arriving peers need to download one copy of the file via the cross-ISP link before the next peer departure. Thus

$$E(d|0) = 1 = \frac{1}{0+1}$$

Given the upper bound of $E(d|n)$ and p_n, one can derive the upper bound of $E(d)$, the average cross-ISP traffic caused by each departure.

$$\begin{aligned} E(d) &= E(E(d|n)) = \sum_{n=0}^{\infty} p_n E(d|n) \\ &\leq \sum_{n=0}^{\infty} \frac{\rho^n}{n!} e^{-\rho} \frac{1}{n+1} = \frac{1}{\rho}(1 - e^{-\rho}) \end{aligned}$$ ∎

Remark: Note that this upper bound is a decreasing function of ρ, the average number of peers in the ISP. Since the lower bound is also a decreasing function of ρ, this implies that the cross-ISP traffic will reduce *significantly* when there are more peers in the ISP.

III. An ISP-friendly BitTorrent Protocol

Let us now present the *ISP-friendly file swarming protocol* which tries to use the ELP as much as possible to reduce the cross-ISP traffic. Before we present our protocol, we first provide a brief review of the BitTorrent (BT) protocol. Note that one design requirement of our protocol is that it has to be "*compatible*" with the current BitTorrent and our clients can communicate directly with existing BT peers. This feature is particularly important since this allows incremental deployment of new protocol and at the same time, providing the same level of service to users.

A. Brief review of the BitTorrent protocol

For the BT protocol, a file is to be divided into many non-overlapping chunks (the default size is 256 KB) and there is at least one peer, which is called a seeder, who holds all these chunks and that the seeder wants to publish the file. A peer can get the file either from the seeder, or other peers holding those chunks it does not possess. Each peer offers upload service to other peers only to the extend that the service is reciprocated. By coupling the service each peer can receive to its upload contribution, the BT protocol successfully makes each peer to play a role of a server and thereby improve the performance of the system. There is a special node called the *tracker*, which keeps track of all peers in the system. A peer needs to first contact the tracker to get a subset of peers who are downloading the file. This peer then establishes connections to these peers and finds out what chunks these peers possess. Then this peer will send out a message INTERESTED to its connected peers, indicating that there exists some chunks it does not possess and this peer wishes to receive some download service. One important point is that the INTERESTED message does *not* indicate which chunk this peer wants. The chunk selection is determined in later step.

Uploading is called *unchoking* in BitTorrent. Each peer unchokes a fixed number of peers simultaneously (the default number is four). Which peers to unchoke is determined by the current downloading rate from these peers, i.e., each peer uploads to the four peers which provide it with the best downloading rate. This unchoking mechanism is called the *tit-for-tat* policy, and one implication of this policy is that it deters free-riding. Beside the tit-for-tat policy, there is another unchoking mechanism called the *optimistic unchoking*, which allows each peer to explore the downloading rates of other peers. Under the optimistic unchoking, each peer randomly selects another peer to upload without considering the service contribution of the selected peer. Optimistic unchoking serves two purposes: (1) it helps new peers to get some chunks so that they can contribute to the community and, (2) it is an attempt to discover another peer with a higher uploading rate. If this kind of peer is found, then the peer with the smallest downloading rate in the regular unchoking set (the four unchoke connections mentioned above) will be terminated and the peer with a higher uploading rate will be included in the regular unchoking set.

Downloading in BitTorrent is determined by the chunk selection policy called the *local rarest first*. When a peer is ready to download from another peer, usually there are several potential choices of chunks to download. Under the local rarest first strategy, a peer will choose the chunk which has the least number of copies among its connected neighbors to download. The local rarest first policy not only can balance the distribution of chunks in the system, but can also enhance the file availability.

B. ISP-friendly protocol

Let us now present an ISP-friendly protocol. In essence, it is a variant of the BitTorrent protocol which exploits ELP. The goal is to reduce the amount of cross-ISP traffic and at the same time, maintain good performance (e.g., file downloading time). There are many details in our protocol, but the basic idea is to try to use the ELP as much as possible.

To adopt ELP, it is necessary for a peer to distinguish between *internal peers* and *external peers*, in other words, peers that are within the ISP or peers that reside in other ISPs. For a BitTorrent peer, it obtains the IP addresses of its connected neighbors from the tracker. Therefore, a peer needs to find the relationship between an IP address and its associated ISP. This type of association can be easily constructed using tools like the ASFinder in the CoralReef suite [2] or exploit the CDN information [5]. In fact, an ISP can set up this "whois" server and provide this mapping service to all peers within its domain.

Being able to distinguish between internal peers and external peers, each peer can exploit the ELP via the following steps:

- Classifies its neighbors into two types: *internal neighbors* are the neighboring peers which belong to the same ISP as itself, and *external neighbors* are the neighboring peers which belong to other ISPs.
- Creates a list C_I where $C_I[j]$ records the number of copies of the j^{th} chunk that are within the *internal neighbors* only. Similarly, creates a list C_E where $C_E[j]$ records the number of copies of the j^{th} chunk that are within the *external neighbors* only.
- For a given peer, let $\mathcal{F}_L$ denote the set of chunks held by this peer (or localhost). For a neighboring peer, let $\mathcal{F}_R$ denote the set of chunks held by this neighbor. If it is an internal neighbor, sends an INTERESTED message to it if it has some chunks which are not possessed by the localhost, i.e., $\mathcal{F}_R \backslash \mathcal{F}_L \neq \emptyset$. If it is an external neighbor, sends an INTERESTED message to it if it has some chunks which are not possessed by *all* internal peers, i.e., $C_I[j] = 0$ for some $j \in \mathcal{F}_R \backslash \mathcal{F}_L$.
- Upon an unchoking event, the peer has to handle it differently depending on whether it was unchoked by an internal neighbor or external neighbor. If the peer was unchoked by an internal neighbor, the peer will request a chunk k using the local rarest first policy over C_I:

$$k = \underset{j}{\operatorname{argmin}} \{C_I[j]\}, \quad j \in \mathcal{F}_R \backslash \mathcal{F}_L. \tag{2}$$

  If the peer was unchoked by an external neighbor, the peer will request only those chunks which are not available in the internal neighbors and using the local rarest first policy over C_E:

$$k = \underset{j}{\operatorname{argmin}} \{C_E[j]\}, \quad j \in \mathcal{F}_R \backslash \mathcal{F}_L, C_I[j] = 0. \tag{3}$$

All other parts of the ISP-friendly protocol remain the same as the current BitTorrent protocol, e.g., tracking, tit-for-tat, optimistic unchoking and so forth.

According to the above mentioned modifications, whether chunk k is a potential choice for downloading from a neighbor can be determined by the following decision function:

decision function want(k):
return $k \in \mathcal{F}_R \backslash \mathcal{F}_L$ and
($ISP_{neighbor} == ISP_{localhost}$ or $C_I[k] == 0$

If want(k) returns "False" for all chunk index k, then the peer is *not interested* in this neighbor. If it returns "True" for some chunk index k, then the peer will send an INTERESTED to this neighbor and wait to be unchoked.

Upon unchoked by an internal (external) neighbor, the peer can use the function want(k) to find out all potential chunks to request, and then look up the table C_I (C_E) to determine which chunk to request first based on the local rarest first policy. Notice that when all neighbors are internal neighbors or all neighbors are external neighbors, this ISP-friendly protocol behaves exactly the same as the current BitTorrent protocol.

In summary, the ISP-friendly protocol proposed above uses the ELP to post the INTEREST message, and during the chunk selection process, uses the ELP and the local rarest first policy. By doing so, a peer differentiates which peers to download from and also try to avoid downloading any duplicate chunk which resides within the same ISP.

Before we leave this section, it is important for us to comment the difference between the proposed ISP-friendly protocol and the idealized model as presented in Sec. II. In practice, the BitTorrent protocol (and the proposed ISP-friendly protocol) is quite involved. It contains many mechanisms to ensure good performance, such as *random first chunk selection, endgame mode, anti-snubbing* and so on. Furthermore, each peer only has a partial view of the whole P2P system and can only make decisions based on its local information. In addition, it takes time for information (e.g., chunk availability) to be propagated throughout the P2P network. Therefore, this ISP-friendly protocol may deviate from the ELP in the sense that

- Each peer may not be connected to all internal peers.
- The chunk availability information cannot be updated instantaneously.

The above scenarios may lead to the situation that duplicated chunks could be downloaded from external peers. The impact of the first scenario can be reduced if peers can contact the tracker more often to request for more neighbors. The impact of the second scenario can be reduced if peers can update their local information (e.g., chunk availability) more frequently with each other.

It is worthwhile for us to mention that the ISP-friendly protocol only aims at reducing the *incoming* cross-ISP traffic. By doing so, it also reduces the *outgoing* cross-ISP traffic because of the built-in tit-for-tat mechanism in BitTorrent. This mechanism enforces certain degree of fairness in data exchange and therefore the total amount of outgoing cross-ISP traffic is approximately equal to the incoming cross-ISP traffic. We will verify this claim in our experiments, which we will present in the following section.

IV. Experiments and Measurements

In order to evaluate the cross-ISP traffic reduction and the average file downloading time of the proposed ISP-friendly protocol, we modify a BitTorrent software to implement the ISP-friendly features mentioned in Sec. III and carry out experiments and measurements on the PlanetLab. To compare the proposed ISP-friendly protocol to the current BitTorrent protocol, we also instrument the same BitTorrent software to collect traffic information for comparison. In the following, we describe in detail on how we carry out the experiment.

A. Choice of the BitTorrent Client

The first BitTorrent client was developed by Bram Cohen [6]. In our experiments, we use Cohen's BitTorrent client, which is considered as the reference for the BitTorrent protocol. Thus, this client is also called the "Official BitTorrent client". Our goal is to evaluate the basic BitTorrent protocol and the proposed ISP-friendly BitTorrent protocol. Thus, we choose the official BitTorrent client and we instrument the official BitTorrent client version 4.4.0 which was released in 2006.

B. Experimental Setup

We carry out experiments when the peers arrival is characterized by a Poisson process. We carry out the experiment twice with the same settings, one with the official BitTorrent client, the other one with the ISP-friendly BitTorrent client. In order to compare their cross-ISP traffic and the file downloading performance, each client logs at least the following information: starting time, ending time, bytes downloaded from internal/external neighbors, bytes uploaded to internal/external neighbors.

There are many configuration options for the official BitTorrent clients. The main default parameters are: the maximum upload rate (default is 20 KB/s), the maximum number of peers to upload to (default is 4), the number of pieces downloaded before switching from random to rarest first piece selection (default is 4), time interval to request more peers from the tracker (default is 300 secs.), the minimum number of neighbors before requesting more peers from the tracker (default is 20), the maximum number of neighbors (default is 80) and so on. It is outside the scope of this study to evaluate the impact of each BitTorrent's parameter. In our experiments, we use the default parameters except that: the time interval to request more peers from the tracker is set to 60 seconds, the minimum number of neighbors before requesting more peers from the tracker is set to 80. We set these two parameters to help peers discover other peers and connect to them sooner.

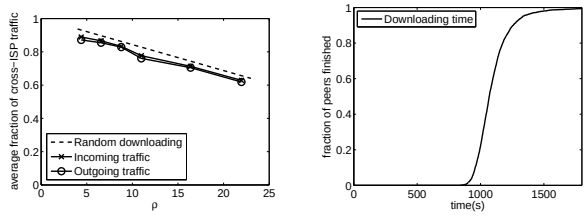
The typical file size of a BT file distribution ranges from tens to hundreds megabytes (files can be music albums, TV shows, movies and so on). Usually users will set the maximum uploading rate larger than the default setting 20KB/s to speed up their downloading. To avoid consuming too much bandwidth and other resource of the PlanetLab nodes, we use a relatively small file (20MB) for downloading, and the piece size is also scaled down to 32KB. There is a seeder in the system to ensure file availability in all our experiments. To avoid the seeder become the bottleneck, its maximum uploading rate is set to 50KB/s, larger than the maximum uploading rate of other peers.

Since most nodes in the PlanetLab are within universities, one can consider each university as an "ISP", and construct a database to map each PlanetLab node to "ISP"(There are some differences between "AS" and "ISP", but it does not matter to our experiments, or we may call it "AS-friendly protocol"). In our experiments, we consider six "ISPs": Berkeley (16 nodes), Columbia (3 nodes), Cornell (6 nodes), MIT (7 nodes), Princeton (11 nodes), and OTHER (32 nodes). Since there may be more than 60 peers for some experiments, we may assign several peers to the same node. But to avoid overloading the node, no more than three peers will be running on the same node at any time.

In the following experiments, we study the cross-ISP traffic and the file downloading time of the official BitTorrent and the proposed ISP-friendly BitTorrent in regular peer arrival scenario, i.e., peer arrival to the ISP is a Poisson

process. To carry out meaningful and realistic experiments, we instrument each ISP with a different peer arrival rate and peers from different ISPs participate in the same torrent file sharing. Note that we have six ISPs: Berkeley, Columbia, Cornell, MIT, Princeton, OTHER. In our experiments, we initiate the seeder and the tracker in Columbia and there is no other peer in Columbia. Peers are launched in the other five ISPs according to Poisson processes. We know that the sum of several independent Poisson arrival streams is still Poisson arrival, thus the peer arrival for the whole P2P network(containing five ISPs) is still Poisson. We carry out the experiment multiple times with the peer's average interarrival time as $250s$, $167s$, $125s$, $100s$, $67s$ and $50s$ respectively for a certain ISP(we choose Berkeley), and the peer arrival for other ISPs are adjusted accordingly to make sure that the peer arrival for the whole P2P network is a Poisson process with an average interarrival time being $16s$. This implies that the ratio of peers in Berkeley and the peers in the whole P2P networks will be about $4/64$, $6/64$, $8/64$, $10/64$, $15/64$ and $20/64$ respectively. The experiment lasts for 48 hours each time. With the log file, we can calculate the average downloading time T in Berkeley, and then derive the average number of peers by $\bar{n} = \lambda T$.

Experiment 1: Regular Peer Arrival for Official BitTorrent
We carry out the experiment using the official BitTorrent client with the settings mentioned above. Since the *maximum* uploading rate of a peer is 20KB/s, and there is only one seeder in the system whose upload rate is negligible comparing to the aggregate upload rates of all peers, therefore, the expected downloading rate of a peer in the system is upper bounded by 20 KB/s. For the experiment, the size of the published file is 20MB, thus the average file downloading time would be larger than $1000s$. This is confirmed by our experiment. Figure 2 illustrates the experimental results.



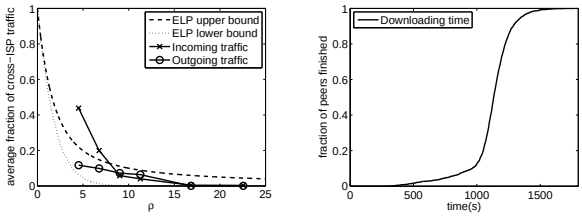
(a) Cross-ISP traffic vs. # of peers (b) CDF of downloading time

Fig. 2. Performance of the Official BitTorrent under Steady Peer Arrival

Figure 2(a) shows the average fraction of incoming and outgoing cross-ISP traffic generated by each peer in Berkeley with different average interarrival time. In Equation (1), we show the fraction of cross-ISPs traffic for the *random downloading* strategy and we also plot this curve in the figure. As stated in Equation (1), the expression is $f = 1 - n/N$ where n is the average number of peers in a certain ISP(It is Berkeley here.) and N is the average number of peers in the whole P2P system. Both n and N can be calculated by the average interarrival time and the average downloading time. From the figure, one can observe that the cross-ISP traffic generated by the official BitTorrent client is *very similar* to the random downloading strategy. It generates a lot of incoming and outgoing cross-ISP traffic. One can also observe that outgoing traffic is slightly less than the incoming traffic. The reason is that there is a seeder in the system and this seeder uploads to other peers but never perform any downloading. Therefore, other ISPs observe more incoming cross-ISP traffic.

Figure 2(b) shows the cumulative distribution function (CDF) of the file downloading time for Berkeley. It can be seen that the curve is sharp, which means that the downloading time for most peers are roughly the same.

Experiment 2: Regular Peer Arrival for the ISP-friendly Protocol
We use the same setting as Experiment 1 except the clients are replaced by our ISP-friendly clients discussed in Section III. The results are illustrated in Figure 3.



(a) Cross-ISP traffic vs. # of peers (b) CDF of downloading time

Fig. 3. Performance of the ISP-friendly BitTorrent under Steady Peer Arrival

Figure 3(a) shows the average fraction of incoming and outgoing cross-ISP traffic generated by each peer in Berkeley with different average interarrival time using ISP-friendly protocol. We also show the lower and upper bounds of the derived cross-ISP traffic model. One can observe that the cross-ISP traffic is *greatly reduced* compared to the official BitTorrent client. The experiment curve for the incoming traffic falls between the bounds when $\bar{n}$, the average number of peers in Berkeley is larger than 7. When $\bar{n}$ is small, the experiment curve exceeds the upper bound. The reason is that the peers in Berkeley are so rare compared to the whole P2P system, it is usually difficult for a newly arriving peer in Berkeley to discover and establish connection to other peers within Berkeley soon. Then this newly arriving peer may request pieces from external peers even these pieces are held by some internal peers, resulting an increase in the cross-ISP traffic. However, if $\bar{n}$ is small in Berkeley, the aggregate cross-ISP traffic will not be very significant. Notice that the ISP-friendly protocol differs from the official BitTorrent client only in the downloading strategy. However, the outgoing cross-ISP traffic is also *significantly reduced*. It is interesting to observe that the outgoing traffic is much less than the incoming traffic when $\bar{n}$ is small, and it can be interpreted like this: the newly arriving peer in Berkeley performs little uploading to external peers compared to downloading, since it has not many pieces to upload.

Figure 3(b) shows the cumulative distribution function

(CDF) of the file downloading time for Berkeley. The first observation is that the downloading time is slightly larger($< 10\%$) than the official BitTorrent. There are two reasons for the increase in file downloading time. First, since peers follow the ELP, the seeder, which resides in a different ISP, may remain idle since downloading from seeder is considered as cross-ISP traffic. Second, since some pieces can only be downloaded from internal peers according to the ELP, it will also degrade some downloading chance. However, the gap is not very large and it will be reduced if there are more peers within Berkeley. Another observation is that the variance of the file downloading time is a little larger than the official BitTorrent.

V. Related Work

There are several analytical studies on file swarming systems. Yang et al. [19] study the service capacity of BT protocols and show that the service capacity of BT scales well with the number of peers, thus providing fast downloading independent of demand rate. Qiu et al. [18] extend the above model and provide an analytical solution to a fluid model and show high scalability and stability of BT protocols. In [7], authors propose models to look at design tradeoffs between performance and fairness of the BT protocol. Mundinger et al. [16] propose a deterministic scheduling algorithm to achieve the optimal makespan. In [15], authors provide a detailed stochastic model to investigate the stability and effectiveness of a P2P file swarming system and show that even by the "random chunk selection" policy, the system throughout is still asymptotically optimal. In [13], the authors provide a detailed stochastic model to capture the peers' diversity (in terms of downloading progress) and show the change of downloading speed during the whole session under variety settings. In [20], authors propose an analytical model for BT-liked streaming applications. There are also numerous empirical studies on the BT protocol. Measurement in [9], [12], [17] study the availability, integrity, performance and the choking mechanism of the BT protocol.

There are only few studies addressing the issue of cross-ISP traffic. One approach is caching [4] but one has to address the copyright legal issue. In [10], authors propose to place some "gateway peers" to connect to external peers and other peers only download within the ISP. However, one has to address the issue of service availability due to sudden departure of gateway peers. Authors in [3] examine a technique named "biased neighbor selection" to explore traffic reduction, but the study was only carried out via simulation. In our work, we propose to exploit the *content locality* which requires no extra hardware investment from the ISP. We analytically evaluate the cross-ISP traffic reduction, and at the same time, propose and implement such mechanism to achieve the reduction while keeping good downloading performance.

VI. Conclusion and future work

In this paper, we address how one can reduce the cross-ISP traffic for file swarming applications. We use a simple and effective idea: exploit the content locality to reduce the traffic. We analytical show the significant cross-ISP traffic reduction when one uses the above principle. We then design and implement such mechanism on a BT software, carry out extensive experiments and measurements on the PlanetLab to demonstrate its effectiveness. Note that for designing an ISP-friendly protocol, one has to consider the possibility for black-hole attack. For detail discussion on this security problem and its solution, please refer to [14].

References

[1] http://multiprobe.ewi.tudelft.nl/dataset.html.

[2] CAIDA. CoralReef suite. http://www.caida.org/tools/measurement/coralreef.

[3] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *IEEE ICDCS*, 2006.

[4] CacheLogic. Advanced solutions for peer-to-peer networks. http://www.cachelogic.com/.

[5] D. R. Choffnes and F. E. Bustamante. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In *Proc. ACM SIGCOMM*, 2008.

[6] B. Cohen. Incentives build robustness in bittorrent, 2003.

[7] B. Fan, D. M. Chiu, and J. C. S. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *IEEE ICNP*, 2006.

[8] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang. Challenges, design and analysis of a large-scale P2P-VoD system. In *Proc. ACM SIGCOMM*, 2008.

[9] M. Izal, G. Urvoy-Keller, E. E. Biersack, P. Felber, A. A. Hamra, and L.Garces-Erice. Dissecting bittorrent: Five months in a torrent's lifetime. In *PAM*, Apr 2004.

[10] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *ACM IMC*, 2005.

[11] L. Kleinrock. *Queueing Systems. Volume 1: Theory*. John Wiley & Sons, 1975.

[12] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *ACM IMC*, 2006.

[13] M. Lin, B. Fan, J. C. S. Lui, and D. M. Chiu. Stochastic analysis of file swarming systems. In *Performance*, 2007.

[14] M. Lin, J. C. S. Lui, and D. M. Chiu. An isp-friedly file distribution protocol: Analysis, design and implementation. *CUHK-CSE Technical Report*, 2008.

[15] L. Massoulie and M. Vojnovic. Coupon replication systems. In *Proc. ACM SIGMETRICS*, 2005.

[16] J. Mundinger, R. Weber, and G. Weiss. Optimal scheduling of peer-to-peer file dissemination. In *Journal of Scheduling*, 2007.

[17] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The bittorrent P2P file-sharing system: Measurements and analysis. In *4th International Workshop on Peer-to-Peer Systems*, 2005.

[18] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proc. ACM SIGCOMM*, 2004.

[19] X. Yang and G. de Veciana. Service capacity of peer to peer networks. In *Proc. IEEE INFOCOM*, 2004.

[20] Y. Zhou, D. M. Chiu, and J. C. S. Lui. A simple model for analyzing P2P streaming protocols. In *IEEE ICNP*, 2007.