

An ISP-friendly File Distribution Protocol: Analysis, Design and Implementation

Minghong Lin John C.S. Lui Dah-Ming Chiu

The Chinese University of Hong Kong

Email: {mhlin,cslui}@cse.cuhk.edu.hk, dmchiu@ie.cuhk.edu.hk

Abstract—In the past few years, P2P file distribution applications (e.g., BitTorrent) are becoming so popular that they are the dominating source of Internet traffic. This creates significant problems to Internet Service Providers (ISPs), not only because of the added complexity in traffic engineering, but the increase of traffic, in particular on the cross-ISP links, implies congestion and a higher operating cost. In this paper, we consider an ISP-friendly file distribution protocol which uses the “*exploiting-the-locality principle*” (ELP) to reduce the cross-ISP traffic. To show its benefit, we derive an upper and lower bound of cross-ISP traffic for the protocols which rely on ELP, and show that the cross-ISP traffic can be reduced significantly when the number of peers within an ISP increases. To carry out realistic study, we design and implement our ISP-friendly protocol (which is compatible with the current BitTorrent protocol) and carry out large scale experiments on PlanetLab to measure the reduction of the cross ISP-traffic and the file downloading time. More important, we also show how the proposed ISP-friendly protocol can handle the “*black-hole*” security attack. This paper sheds light on the merits and design direction of ISP-friendly content distribution protocols.

I. Introduction

P2P technology has been recognized as a mechanism to deploy scalable service on the Internet. This can be observed by the wildly popular services such as content distribution via BitTorrent [9], VoIP via Skype, and IPTV via PPLive. Unlike the traditional client-server or fixed-infrastructure content distribution (e.g., Akamai), P2P technology has the self-scaling property: the supply capacity grows linearly with the demand.

The popularity of P2P applications, in particular, the file distribution application like BitTorrent, introduces some challenging issues. Studies show that P2P applications account for over 60% of the traffic seen by an ISP [7]. 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 important, high operating cost for ISPs.

ISPs have several options to deal with the above problem. One approach is to control the file distribution traffic via packet throttling. However, this is not an effective solution since applications can always use dynamic port to bypass detection. Also, throttling discourages users within an ISP and these users may opt to switch to another ISP for service. Another approach for an ISP is to perform 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. Not only caching requires additional infrastructure and cost, but also introduces legal problem to the ISPs due to the copyright issue.

Researchers propose some techniques to reduce the cross-ISP traffic. One is to select a single peer, called the “gateway peer”, to connect to the external world [13]. This technique requires constant maintenance of the gateway architecture by the ISP and make sure that it would not be selfish and would be glad to upload to other internal peers within this ISP. Another technique is to modify the tracker to return more *internal* peers when a peer requests a neighbor list [6]. This technique weakens the connectivity of the P2P network in order to reduce the cross-ISP traffic. There are some proposals on enhancing the cooperations between ISPs and P2P users [25] [4] [5].

In this paper, we introduce an “*ISP-friendly file distribution protocol*”, which is based on the BitTorrent protocol and can reduce the cross-ISPs traffic. Our protocol differs from the above techniques in that it relies on pieces availability but not only file availability to decide which neighbor to chock (Choking is a temporary refusal to upload; It stops uploading, but downloading can still happen and the connection doesnt need to be renegotiated when choking stops). Thus it requires a little more information (bitmap in BitTorrent) to make better decision. Our proposal is end-hosts oriented, which can be deployed easily and at the same time, promote collaboration between ISPs and P2P users. The goal of our protocol is to reduce the cross-ISP traffic, maintain good file downloading performance and at the same time, do away with expensive infrastructure support. The protocol relies on the following idea: downloading pieces from internal peers (who belong to the same ISP) as many as possible, i.e., peers tend not to consider external peers (who belong to other ISPs) if the piece is held by some internal peers. We call it the “*exploiting-the-locality principle*” (ELP). The ELP is: *never download information from external peers if there exist at least one copy of the information among the internal peers*. It is possible to modify the BT clients’ interested behaviors to follow this principle without changing the topology of the BitTorrent network (Messages flows, such as *bitfield*, *have* etc., are the same as the official protocol, although the data flows are different), so that make the peer adapt to new situation much more quickly than the topology maintenance approach. The contributions of our works are:

- We *analytically quantify* the merits when file distribution protocols follow the ELP. In particular, we derive the

lower and upper bounds of incoming cross-ISP traffic under regular peer arrival (i.e., Poisson process) and bursty peer arrival (i.e., flash crowd).

- We propose and implement an ISP-friendly protocol on existing BitTorrent client software. It is compatible with the current BitTorrent protocol. We show that a client only needs to control the *incoming* cross-ISP traffic and the *outgoing* cross-ISP traffic will be reduced accordingly.
- We carry out experiments and measurements on Planet-Lab to demonstrate significant cross-ISP traffic reduction and good file downloading performance.
- We illustrate the *black-hole security attack* and show how the modified ISP-friendly protocol can overcome this problem.

The outline 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 distribution protocol follows the ELP. In Section III, an ISP-friendly file distribution protocol is presented. In Section IV, we discuss our experiments on PlanetLab and present our measurement results. Black-hole security attack is presented in Section V and we show how the modified ISP-friendly protocol can cope with the problem. Related work is given in Section VI and Section VII concludes.

II. Mathematical Models

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

Before we present the analysis of an ISP-friendly protocol, let us consider the current P2P file distribution system, such as BitTorrent, in which peers do not consider the boundary between ISPs in their data transfer. We call such kind of P2P file distribution 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} . Assume that the bandwidth distribution of peers is independent of which ISP the peers belong to. Considering a randomly chosen peer which resides in the ISP \mathcal{A} , the probability of choosing a peer outside ISP \mathcal{A} for the data transfer is

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

Thus the expected fraction of file content which is downloaded from (upload to) peers outside ISP \mathcal{A} is $(1 - n/N)$. The total amount of incoming (outgoing) 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 distribution system, for instance, N is much larger than 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). Obviously, only when the reduction of cross-ISP traffic is high, then one should consider designing and implementing an ISP-friendly file distribution protocol. In our analysis, we concentrate on two common scenarios in P2P file distribution: regular peer arrival and a big bursty peer arrival (flash crowd).

A. Assumptions

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

- Peer 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 pieces of a file, the peer will leave the system immediately.
- All peers become aware of a piece as soon as this piece is downloaded into the ISP.
- The piece diversity of the P2P system is very good so that peers will be interested in each other with high probability.

Note that the last assumption is a common assumption for most fluid models of P2P systems [10], [11], [23]. One may argue that the *last piece problem* may destroy this assumption, but the measurement results in [1], [15] and the stochastic analysis [16], [17] show that peers show interest of each other most of the time and the last piece problem only affects the last few pieces. Its effect in the mathematical model can be safely ignored for a large file which contains thousands of pieces. This assumption means that the downloading rate for a given peer can be represented by a random variable which is independent of its downloading progress.

Note that based on the ELP, if there exists a seeder in an ISP, then all peers in that ISP will never download pieces 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 a upper and lower bound of this measure. Before we present the formal analysis, let us use an example as shown in Figure 1 to illustrate the idea. The file has 20 pieces and at this moment, there are 3 peers within the ISP. Let v_i be the fraction of progress in the file download for peer i . In this example, we have $v_1 = 0.3$ (6 pieces), $v_2 = 0.15$ (3 pieces) and $v_3 = 0.2$ (4 pieces). Since peers follow the ELP, only those missing pieces by all peers would be downloaded through the cross-ISP link. How many pieces would be downloaded through the cross-ISP link before the next peer departure? In the best case, when all peers possess *different* pieces from each others, then the external download will be $d = 1 - \sum_{i=1}^3 v_i = 0.35$ (7 pieces) and this is the

lower bound. In the worst case, the set of pieces possessed by any peer is a subset of the set of pieces possessed by the peer with the maximum progress. In this case, we need to download all missing pieces of peer 1 and it is equal to $d = 1 - \max_{i=1}^3 \{v_i\} = 0.7$ (14 pieces), which is the upper bound. The remaining question is how to uncondition the number of peers and v_i 's. We are now in the position to develop the mathematical model. We are going to apply fluid approximation in our analysis in the next two sections, which require that the file is divided into infinite (or so many) pieces that two peers will not request the same piece at the same time. Once this assumption is violated, the results would be violated. Thus this bounds would be a good approximation only when the file is divided into thousands of pieces or more. As mentioned before, we consider two scenarios: regular peer arrival case and flash crowd case. Let us first focus on the analysis of regular arrival case.

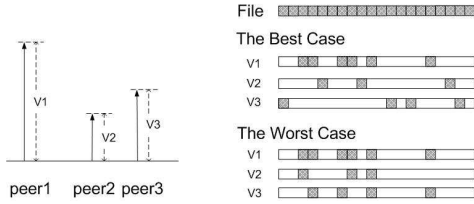


Fig. 1. Illustration of the lower and the upper bound of cross-ISP traffic

B. Homogeneous Case Analysis

Let us first consider the homogeneous case: 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 distribution 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 time 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

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

where $\bar{n} = \lambda T$ and $I_1(x)$ is the modified Bessel function.

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

Proof: Please refer to the Appendix for derivation. ■

Assume that each peer is aware of other peers' state in real time, then for a P2P system which follows the ELP, 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 distribution 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 time 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

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

where $\bar{n} = \lambda T$.

Proof: Please refer to the Appendix for derivation. ■

C. Heterogeneous Case Analysis

In here, we extend our model to consider the heterogeneous case where peers have different downloading time (or different bandwidth).

In a large P2P system, the total service capacity of the system scales up as the number of peers increases [26], and the downloading time is roughly independent of the number of peers in the system. We use T , which is now a random variable, to represent the file downloading time of a peer and extend the model to derive the bounds of the heterogeneous case.

Theorem 3: For a given ISP in which all peers use an ISP-friendly file distribution protocol, there is no seeder in that ISP and the peers arrival process is characterized by a Poisson process with an average rate λ . Let q_i be the probability that the downloading time for a peer will be τ_i , then the average amount of incoming cross-ISP traffic caused by each peer in the steady state, denoted by $E(d)$, is bounded by

$$e^{-\bar{n}} \bar{n}^{-1/2} I_1(2\sqrt{\bar{n}}) \leq E(d) \leq \frac{1}{\bar{n}} (1 - e^{-\bar{n}}),$$

where $\bar{n} = \lambda q_1 \tau_1 + \lambda q_2 \tau_2 + \dots$, and $I_1(x)$ is the modified Bessel function.

Proof: Please refer to the Appendix for derivation. ■

Remark: In summary, Theorem 3 gives the lower bound and upper bound of the average cross-ISP traffic caused by each peer when all peers adopt the ELP. To illustrate the spread of these bounds, we consider an ISP with different values of \bar{n} . Figure 2 illustrates the spread of these two bounds on the cross-ISP traffic, as well as the average cross traffic when one uses the random downloading strategy (e.g., the conventional P2P file distribution protocol) with $N = 200$ peers in the P2P system. One can observe that both bounds decrease quickly when n (the \bar{n} in the theorems), the average number of peers within that ISP, increases. Notice that the cross-ISP traffic for random downloading remains high. This justifies the design and implementation of an ISP-friendly file distribution protocol.

D. Flash Crowd Analysis

Let us now consider the flash crowd scenario when a large number of peers arrive to the ISP in a very short period of time. This occurs, for example, when a very popular movie or an OS kernel update is being first published to the Internet. Based on the same assumptions we made in the regular arrival analysis (except that peer arrival process is no longer Poisson), we can derive the upper and lower bound of the cross-ISP traffic.

Theorem 4: For a given ISP in which all peers use an ISP-friendly file distribution protocol and there is no seeder in that ISP. At time $t = 0$, n peers arrive and there is no more peer arrival after $t > 0$. Let τ_{min} (τ_{max}) be the shortest (longest)

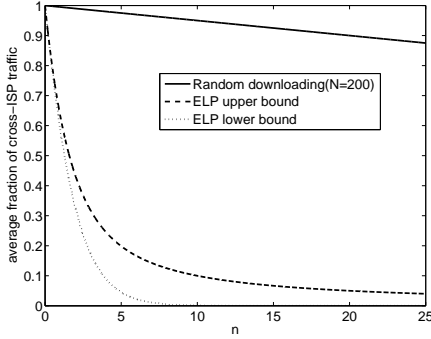


Fig. 2. Average fraction of cross-ISP traffic vs. the average number of peers in the ISP

downloading time of these peers. The average amount of incoming cross-ISP traffic caused by each peer, denoted by $E(d)$, is bounded by

$$1/n \leq E(d) \leq \left(1 + \log\left(\frac{\tau_{max}}{\tau_{min}}\right)\right) / n.$$

Proof: Please refer to the Appendix for derivation. ■

Remark: Notice that τ_{max} is the downloading time of peers with the lowest downloading rate, τ_{min} is the downloading time of peers with the highest downloading rate. It is interesting to observe that the upper bound of the cross-ISP traffic depends on τ_{max}/τ_{min} and n only.

III. An ISP-friendly BitTorrent Protocol

In this section, we present our *ISP-friendly file distribution protocol* which uses the ELP to reduce the cross-ISP traffic. To appreciate the proposed 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 software and our clients can communicate directly with existing BT peers. This feature is particularly important since this new service can then be incrementally deployed.

Under the BT protocol, a file is to be divided into many non-overlapping pieces (the default size is 256 KB) and there is at least one peer, which is called a seeder, who holds all these pieces and this seeder wants to publish the file. A peer can get the file either from the seeder, or from other peers holding those pieces it does not possess. Each peer offers upload service to other peers only to the extent that the service is reciprocated. By coupling the service each peer can receive to its upload contribution, the BT protocol successfully makes each peer 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 other peers and finds out what pieces these peers possess. Then this peer will send out an INTERESTED message to its connected peers, indicating that there exists some pieces 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 piece this peer wants. The piece 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 who provide it with the top four downloading rates. 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 pieces 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 will be replaced by this peer.

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

Let us now present our 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., small file downloading time). There are many details in our protocol, but the basic idea is: *a peer will not download a piece from external neighbors if he finds that this piece is held by some internal neighbors*.

To adopt ELP, it is necessary for a peer to distinguish peers that are within the ISP and 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 [3], or exploit the CDN information as suggested in [8]. In fact, an ISP can set up a “whois” server to provide this mapping service to all peers within its domain. It only needs to map all IP addresses belonging to itself and its customer ISPs as internal peers, and this can be easily constructed using the CIDR address format. An important point is that there is an *economic incentive* for an ISP to provide this type of mapping service. It can encourage peers to use the ISP-friendly protocol, therefore reduce the cross-ISPs traffic and its operating cost.

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

- 1) Divides its neighbors into two type, *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.

- 2) Creates a list C_I where $C_I[j]$ records the number of copies of the j^{th} piece that are within the *internal neighbors*. Similarly, creates a list C_E where $C_E[j]$ records the number of copies of the j^{th} piece that are within the *external neighbors*.
- 3) For a given peer, let \mathcal{F}_L denote the set of pieces held by this peer (or localhost). For a neighboring peer, let \mathcal{F}_R denote the set of pieces held by this neighbor. If it is an internal neighbor, sends an INTERESTED message to it if it has some pieces which are not possessed by the localhost, i.e., $\mathcal{F}_R \setminus \mathcal{F}_L \neq \emptyset$. If it is an external neighbor, sends an INTERESTED message to it if it has some pieces which are not possessed by *all* internal peers, i.e., $C_I[j] = 0$ for some $j \in \mathcal{F}_R \setminus \mathcal{F}_L$.
- 4) 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 piece k using the local rarest first policy over C_I :

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

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

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

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 piece k is a potential choice for downloading from a neighboring peer can be determined by the following decision function:

```
def want(k):
    return k ∈  $\mathcal{F}_R \setminus \mathcal{F}_L$  and
           ( $ISP_{neighbor} == ISP_{localhost}$  or  $C_I[k] == 0$ )
```

If $\text{want}(k)$ returns “False” for all piece index k , then the peer is *not interested* in this neighbor. If it returns “True” for some piece index k , then the peer will send an INTERESTED message to this neighbor and wait to be unchoked.

Upon unchoked by an internal (external) neighbor, the peer can use the function $\text{want}(k)$ to find out all potential pieces to request, and then look up the table C_I (C_E) to determine which piece 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 send the INTEREST message, and during the piece selection process, uses the ELP and the local rarest first policy. By doing so, a peer determine which peers to download from and also attempts to avoid downloading any duplicate piece which resides within the same ISP.

Before we leave this section, it is important for us to comment about 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 piece 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., piece 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 piece availability information cannot be updated instantaneously.

The above scenarios may lead to the situation that duplicated pieces 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., piece availability) more frequently with each other.

Notice that the ISP-friendly protocol only manage 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. This is verified by our experiments which are presented in the following section.

IV. Performance Evaluation 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, the inventor of the BitTorrent protocol [9]. Note that there are many other BitTorrent clients available, such as μ Torrent, BitComet, Azureus and so on. Since there is no de facto standard, Cohen’s BitTorrent client is considered as the reference for the BitTorrent protocol. Thus, this client is also called the “Official BitTorrent client”. It is an open source software, written in Python and can be executed on many different platforms. Most BitTorrent clients maintain compatibility with the official BitTorrent client. The main differences of these clients are the user interface, configuration options (e.g., caching option to reduce disk access) and certain extensions to the BitTorrent protocol (e.g., UDP transport to traverse NAT). 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 under two scenarios: regular peer arrival and flash crowd. For each scenario, we run two categories of experiments with the same settings, one with the official BitTorrent client, the other one with the ISP-friendly BitTorrent client, thus there will be four categories of experiments in total. In order to compare their cross-ISP traffic and the file downloading performance, each client logs the following information: starting time, ending time, bytes downloaded from internal/external neighbors, bytes uploaded to internal/external neighbors.

TABLE I
DEFAULT CONFIGURATIONS FOR THE OFFICIAL BITTORRENT

Configuration option	Default value
the maximum upload rate	20 KB/s
the maximum number of peers to upload to	4
the number of pieces downloaded before switching from random to rarest first	4
time interval to request more peers from the tracker	300 secs.
the minimum number of neighbors before requesting more peers from the tracker	20
the maximum number of neighbors	80

There are many configuration options for the official BitTorrent clients. Some default values are shown in Table I. 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 ($\geq 100KB/s$) much larger than the default setting to speed up their downloading. Since the PlanetLab resources are shared by many users simultaneously, very high volume traffic is not allowed. Thus we use the default maximum uploading rate ($20KB/s$), and reduce the file size and piece size by similar scale, so that the sojourn time would be similar to real torrents. The file size is $20MB$ (compared to $100MB - 200MB$ in real torrents). The piece size is $32KB$ (compared to $128KB - 256KB$ in real torrents). How to choose the piece size is studied in [19], and a conventional wisdom is stated in the unofficial BitTorrent specification [2]. 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. Although all nodes are in universities which seems to have good network condition (and seems different from real situation), we monitored the connections & transfer rates and found that the network condition is not very stable (similar to real situation) since CPU/Memory/Bandwidth is shared by many PlanetLab users at the same time for each PlanetLab node. Further more, we pick some PlanetLab nodes with low bandwidth (e.g. *-dsl.cs.cornell.edu) to represent more difference in network condition.

C. Regular Peer Arrival

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 (e.g. 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. There are two categories of experiments with the same settings, one with official BitTorrent client, one with ISP-friendly BitTorrent client. Each category is carried out 5 runs, each run lasts for 48 hours (about $48 * 3600/16 \sim 10000$ arrivals in each run). 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 3 illustrates the experimental results.

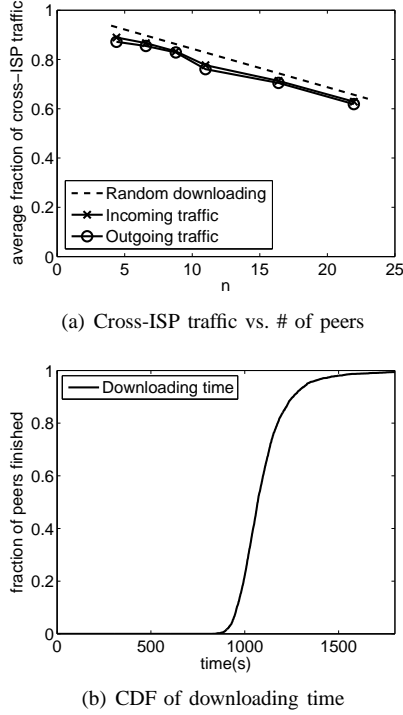


Fig. 3. Performance of the Official BitTorrent under Steady Peer Arrival

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. In Equation (1), we show the fraction of cross-ISP 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 3(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 4.

Figure 4(a) shows the average fraction of incoming and outgoing cross-ISP traffic generated by each peer in Berkeley with

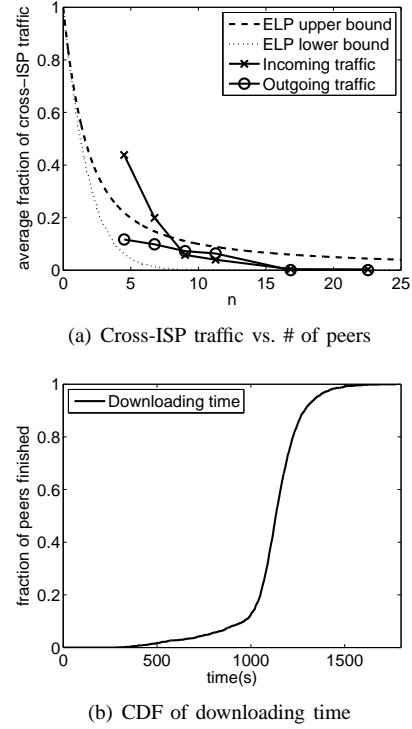


Fig. 4. Performance of the ISP-friendly BitTorrent under Steady Peer Arrival

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 4(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.

D. Flash Crowd

In here, we study the cross-ISP traffic and the file downloading time of the official BitTorrent and the ISP-friendly BitTorrent under the flash crowd scenario. There are five ISPs: a seeder and a tracker are located in Columbia. All peers arrive at $t = 0$ to the four ISPs. Number of peers in the ISPs are: 6 (Cornell), 12 (MIT), 18 (Princeton) and 24 (Berkeley). There are two categories of experiments with the same settings, one with official BitTorrent client, one with ISP-friendly BitTorrent client. Each category is carried out 20 runs ($60 * 20 = 1200$ arrivals in each category) to obtain a good confidence interval.

Experiment 3: Flash Crowd for Official BitTorrent

We use the official BitTorrent clients in this experiment. The results are shown in Figure 5.

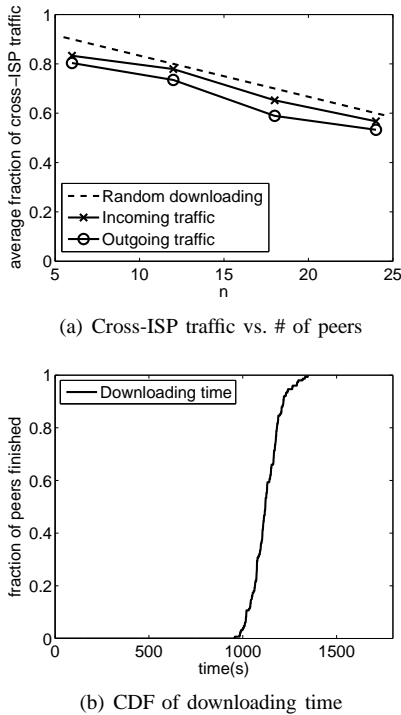


Fig. 5. Performance of the Official BitTorrent under Flash Crowd

Figure 5(a) shows the average cross-ISP traffic generated by each peer in different ISPs. The total number of peers in the system is 60, thus we plot the curve $f = 1 - n/60$ (using Eq. (1)) to represent the random downloading strategy. One can observe that the cross-ISP traffic generated by the official BitTorrent client is very close to Eq. (1). It means that the official BitTorrent client also generates significant amount of cross-ISP traffic in the flash crowd scenario. Another observation is that the outgoing cross-ISP traffic is slightly less than the incoming cross-ISP traffic. This is justified due

to the tit-for-tat policy in BitTorrent. The reason for the slight difference is that there is a seeder in the system who uploads to other peers but never perform any downloading.

Figure 5(b) shows the CDF of the file downloading time. The results indicate that it is very “deterministic” in the sense that most peers finish the file download approximately at the same time.

Experiment 4: Flash Crowd for the ISP-friendly Protocol

The setting of this experiment is exactly the same as Experiment 3 except we use the ISP-friendly client. The results are shown in Figure 6.

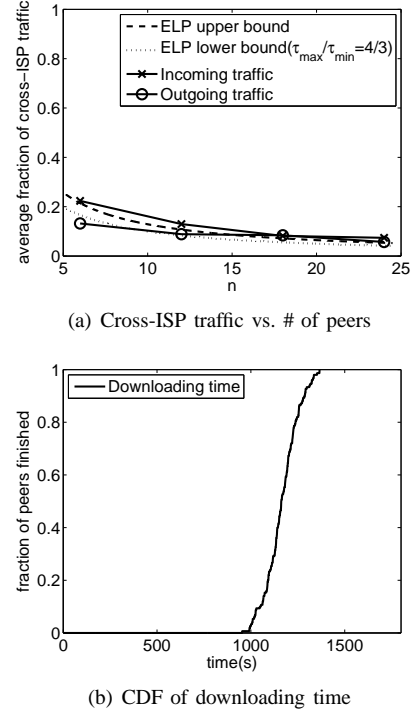


Fig. 6. Performance of the ISP-friendly BitTorrent under Flash Crowd

Figure 6(a) shows the average cross-ISP traffic generated by each peer in different ISPs. One can observe that the cross-ISP traffic is *significantly reduced* compared to the official BitTorrent client (Figure 5(a)). We can also observe that outgoing cross-ISP traffic is slightly less than incoming cross-ISP traffic due to the tit-for-tat policy.

Figure 6(b) shows the cumulative distribution function of the file downloading time. Again, it is very deterministic in that most peers can finish the file download around the same time. Compared with Figure 5(b), one can observe that the file downloading time of the ISP-friendly client is only *slightly* worse ($< 5\%$) than the official BT client.

V. Black Hole Security Attack

We have seen that the ISP-friendly protocol can greatly reduce the cross-ISP traffic while keeping good file downloading performance. In this section, we present the “*black hole attack*”, which may have a detrimental effect on the ISP-friendly file distribution protocol.

Consider a free-rider in a file distribution session. This free-rider will advertise to other peers that it has a lot of

pieces (or all pieces of the file) but it refuses to provide any upload service to other peers. This type of free-riders do exist in the current BitTorrent file distribution but they only receive minimal amount of service: free riders can only download pieces via the the “optimistic unchoked” connection. Therefore, the file downloading time of these free-riders is significantly larger than those normal peers who are willing to provide upload service. This type of free-riding, however, can be *detrimental* to the ISP-friendly protocol. In particular, when the free-rider announces that it has all pieces of the file (or it pretends to be a seeder), it prevents other internal peers obtaining information from external peers, and this may halt the whole file download process within the ISP. We call this the “*black hole attack*”.

To overcome the black hole attack, one needs to provide some mechanism to filter out the attackers or peers with very low uploading rate. One may first consider the black listing technique to do the peers filtering. But black listing needs addition collaboration among peers since one peer could not detect the attacker based on his own local view. This addition collaboration, will make the ISP-friendly protocol incompatible with the current BitTorrent protocol. Instead, we propose an *Enhanced ISP-friendly protocol* which can filter bad peers effectively while keeping the compatibility with the current BitTorrent protocol.

Similar to the ISP-friendly protocol proposed in Section III, each peer classifies its neighbors into two categories: *internal peers* and *external peers*. In the *Enhanced ISP-friendly protocol*, each peer will pick less than or equal to q internal peers as its *active co-agents* (we will show how to select *active co-agents* later). Denote the following set:

$$S = \{c | \text{piece } c \text{ is missed by all its } \textit{active co-agents}\}.$$

the only thing we need to modify compared to the previous ISP-friendly protocol is the decision function $want(k)$. Whether piece k is a potential choice for downloading from a neighbor can be determined by the following new decision function:

```
def want(k):
    return  $k \in \mathcal{F}_R \setminus \mathcal{F}_L$  and
           ( $ISP_{neighbor} == ISP_{localhost}$  or  $k \in S$ )
```

If $want(k)$ returns “False” for all piece index k , then the peer is *not interested* in this neighbor. If it returns “True” for some piece 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 new function $want(k)$ to find out all potential pieces to request, and then look up the table C_I (C_E) to determine which piece to request first based on the local rarest first policy.

Now let us discuss how to pick the *active co-agents*. The intuitive notion of *active coagents* is that, if peer A considers peer B as its *active co-agent*, it implies that peer A detects that peer B works well on uploading to internal peers, thus it may not be necessary for peer A to download the pieces which are held by B from external peers. As we emphasized earlier, we want to design a protocol which is compatible with the current BitTorrent protocol, thus we can pick the *active co-*

agent based on the local information only. A credible evidence that an internal peer works well on uploading to internal peers is that it uploads to you recently. Based on this notion, we develop the selection algorithm as follows.

- 1) Measuring the downloading rate r_i from each internal peer, where r_i is the average rate for the last T seconds.
- 2) Ranking the internal peers in a list according to r_i in decreasing order.
- 3) Truncating the list for peers with $r_i < R$.
- 4) Picking the top q peers as *active co-agents*.

There are three parameters in this algorithm, T , R and q . We do not want the *active co-agents* to change too rapidly. So we select T to be within 1 to 2 minutes. Threshold R is to prevent anti-snubbing attack in which a peer schedules to satisfy just one request per 60 seconds to avoid getting snubbed. A reasonable value for R is between $0.5KB/s - 2KB/s$. Lastly, q is crucial to reducing the cross-ISP traffic. Our measurement study shows that it is sufficient to set $q \geq 5$. To evaluate the enhanced ISP-friendly protocol, we carry out the regular peer arrival experiments as follows.

We configure our enhanced ISP-friendly client to the same settings as SectionIV (e.g., maximum uploading rate is $20KB/s$, maximum number of peers to upload to is 4, etc). But the size of the file for downloading is now doubled to $40MB$. The peer arrival for the whole P2P network is a Poisson process with average interarrival time being $31s$. We carry out the experiment multiple times with the peer’s average interarrival time as $500s$, $333s$, $250s$, $166s$ and $125s$ respectively for Berkeley. This implies that the ratio of peers in Berkeley and the peers in the whole P2P network is about $2/32$, $3/32$, $4/32$, $6/32$ and $8/32$ respectively. In the experiment, there is always one malicious free rider (or faked seeder) in Berkeley. The three parameters of the enhanced ISP-friendly protocol are set as $T = 2min$, $R = 0.5KB/s$ and $q = 10$. The results are shown in Figure 7.

A. Result and Discussion

In Figure 7(a), one can observe that the cross-ISP traffic is significantly reduced compared to the official BitTorrent (reduced to about $1/3$ in our experiments). In Figure 7(b), one can see that there is only a slight performance degradation ($< 10\%$) in the file downloading time compared to the official BitTorrent. Notice that the performance of the enhanced ISP-friendly protocol is not seriously affected by the malicious free rider. Actually the experiment results are similar to Figure 7 when there are several malicious peers. And the enhanced ISP-friendly protocol works well in the Flash Crowd experiments too. Due to the performance degradation, one may worry about the incentive for user to adopt such ISP-friendly protocol. Actually, the performance degradation is because we care about cross-ISP traffic now, and thus there are more constraints in our protocol compared to current BitTorrent (most approaches proposed until now slow downloading compared to current BitTorrent). Therefore, our aim is to keep a happy/tolerant relationship between P2P users and ISPs. The slight performance degradation in file downloading would be acceptable given that the large reduction of cross-ISP traffic.

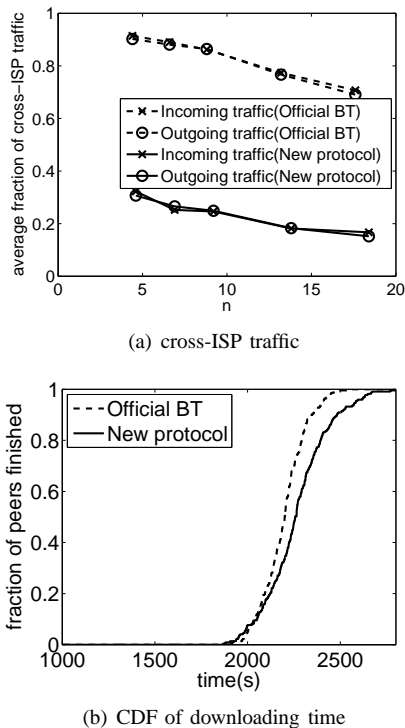


Fig. 7. Enhanced ISP-friendly BitTorrent under Regular Arrival

And the performance gap of file downloading time can be even reduced for the large ISP with many BitTorrent users in it. That is one reason why we propose a protocol compatible with current BitTorrent, so that it is possible to deploy it gradually. In the case that external peers upload much faster than internal peers, it is a tradeoff between saving cross-ISP traffic and maintaining good download performance. We may adjust the parameters R and q in the enhanced protocol to balance between this two objectives. Actually, such tradeoff is very common for locality-exploiting protocols (not only our protocol) and worth understanding more in the future.

VI. Related Work

In [18], authors provide a game-theoretic analysis on how to provide service differentiation on P2P networks. There are several analytical studies on file distribution systems. Yang et al. [26] study the service capacity of BT protocols and show that the service capacity scales well with the number of peers, thus providing fast downloading independent of the demand rate. Qiu et al. [23] extend the model and provide an analytical solution to a fluid model and show high scalability and stability of BT protocols. In [10], [11], authors examine the tradeoffs between performance and fairness of the BT protocol. Authors in [21] propose a deterministic scheduling algorithm to achieve the optimal makespan. In [20], authors provide a detailed stochastic model to investigate the stability and effectiveness of a P2P file distribution system and show that even by the “random piece selection” policy, the system throughout is still asymptotically optimal. In [16], [17], 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 [27], authors propose an analytical model for BT-like streaming applications. There are also numerous empirical studies on the BT protocol. Measurement in [12], [15], [22] study the availability, performance and the choking mechanism of the BT protocol.

There are a few studies addressing the issue of cross-ISP traffic. In [24], authors provide a mathematical framework to study the ISP peering and overlay traffic. In [13], authors propose to place some “gateway peers” to connect to external peers and other peers only download within the ISP. Authors in [6] examine a technique named “biased neighbor selection” to explore traffic reduction. In [25], authors propose a P4P architecture so as to allow ISPs to explicitly provide more information and guidelines to emerging applications such as P2P content distribution and P2P streaming services. For this architecture to be successful, it requires a number of ISPs and content providers to buy in to the architecture and that they have to “trust” each other in providing guidelines and information. Therefore, the P4P architecture needs to overcome a lot of business negotiations and the security concerns from ISPs in exposing their network information. In [5], authors propose a lightweight solution that relies on an oracle service. In [8], authors propose to utilize the redirection behavior of CDN so as to estimate the peers “locality”. This is an alternative approach to get the locality information instead of the oracle service [5], or our approach of using the “whois” service in the Internet. In general, any locality exploiting protocol needs some kind of locality information. To get the locality information, our system can use the “whois” service, or the oracle service or via CDN redirection so as to estimate the locality information.

VII. Conclusion

In this paper, we address how one can reduce the cross-ISP traffic for file distribution applications. We use a simple and effective idea: exploit the content locality to reduce the traffic. We analytically 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. Lastly, we illustrate the black hole security attack and how one can modify the proposed protocol to address this problem.

REFERENCES

- [1] <http://multiprobe.ewi.tudelft.nl/dataset.html>.
- [2] BitTorrent specification. In <http://wiki.theory.org/BitTorrentSpecification>.
- [3] CAIDA. CoralReef suite. <http://www.caida.org/tools/measurement/coralreef>.
- [4] V. Aggarwal, O. Akonjang, and A. Feldmann. Improving user and isp experience through isp-aided p2p locality. In *Proc. IEEE Global Internet Symposium*, 2008.
- [5] V. Aggarwal, A. Feldmann, and C. Scheideler. Can isps and p2p users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 2007.
- [6] 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.
- [7] CacheLogic. Advanced solutions for peer-to-peer networks. <http://www.cachelogic.com/>.

- [8] 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.
- [9] B. Cohen. Incentives build robustness in bittorrent. Technical report, bittorrent.org, 2003.
- [10] B. Fan, D. M. Chiu, and J. C. S. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *IEEE ICNP*, 2006.
- [11] B. Fan, J. C. S. Lui, and D.-M. Chiu. The design trade-offs of bittorrent-like file sharing protocols. *IEEE/ACM Trans. Netw.*, 17(2):365–376, 2009.
- [12] M. Izal, G. Urvoy-Keller, E. E. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice. Dissecting bittorrent: Five months in a torrent’s lifetime. In *PAM*, Apr 2004.
- [13] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *ACM IMC*, 2005.
- [14] L. Kleinrock. *Queueing Systems. Volume 1: Theory*. John Wiley & Sons, 1975.
- [15] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *ACM IMC*, 2006.
- [16] M. Lin, B. Fan, J. C. S. Lui, and D. M. Chiu. Stochastic analysis of file swarming systems. In *IFIP WG 7.3 Performance Conference*, 2007.
- [17] M. Lin, B. Fan, J. C. S. Lui, and D.-M. Chiu. Stochastic analysis of file-swarming systems. *Perform. Eval.*, 64(9-12):856–875, 2007.
- [18] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. Incentive and service differentiation in p2p networks: a game theoretic approach. *IEEE/ACM Trans. Netw.*, 14(5):978–991, 2006.
- [19] P. Marciniak, N. Liogkas, A. Legout, and E. Kohler. Small is not always beautiful. In *IPTPS*, 2008.
- [20] L. Massoulié and M. Vojnovic. Coupon replication systems. In *Proc. ACM SIGMETRICS*, 2005.
- [21] J. Mundinger, R. Weber, and G. Weiss. Optimal scheduling of peer-to-peer file dissemination. In *Journal of Scheduling*, 2007.
- [22] 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.
- [23] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proc. ACM SIGCOMM*, 2004.
- [24] J. H. Wang, D.-M. Chiu, and J. C. S. Lui. A game-theoretic analysis of the implications of overlay network traffic on isp peering. *Computer Networks*, 52(15):2961–2974, 2008.
- [25] H. Xie, R. Y. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4P: provider portal for applications. *SIGCOMM Comput. Commun. Rev.*, 38(4):351–362, 2008.
- [26] X. Yang and G. de Veciana. Service capacity of peer to peer networks. In *Proc. IEEE INFOCOM*, 2004.
- [27] Y. Zhou, D.-M. Chiu, and J. C. S. Lui. A simple model for analyzing P2P streaming protocols. In *ICNP*, pages 226–235, 2007.

APPENDIX

Proof for Theorem 1: 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 P2P file distribution 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

$$p_n = \frac{(\lambda T)^n}{n!} e^{-\lambda T} = \frac{\bar{n}^n}{n!} e^{-\bar{n}} \quad n = 0, 1, \dots$$

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 $i \in \{1, \dots, n\}$. 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* [14] and select the departure points as our observation points. Since the arrival is a Poisson process, we have

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

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 least $(1 - v)$ 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 $X_i \sim U[0, 1], i = 1, \dots, n$. 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:

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

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

Thus

$$f(v|n) = \sum_{j=0}^n C_n^j (-1)^j \frac{(v-j)^{n-1}}{(n-1)!} u(v-j)$$

$$= \sum_{j=0}^{\lfloor v \rfloor} C_n^j (-1)^j \frac{(v-j)^{n-1}}{(n-1)!}$$

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

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

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

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

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.

$$E(d) = E(E(d|n)) = \sum_{n=0}^{\infty} p_n E(d|n)$$

$$\geq \sum_{n=0}^{\infty} \frac{\bar{n}^n}{n!} e^{-\bar{n}} \frac{1}{(n+1)!} = e^{-\bar{n}} \bar{n}^{-1/2} I_1(2\sqrt{\bar{n}})$$

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

Proof for Theorem 2: 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., $X_i \sim U[0, 1], i = 1, \dots, n$. 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 or may not 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

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

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

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

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.

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

Proof for Theorem 3: Let T denote the random variable of the file downloading time of peers. Suppose T is a discrete random variable with possible outcomes of $\tau_1, \tau_2, \dots, \tau_m$ and

$$\text{Prob}(T = \tau_i) = q_i, \quad i = 1, 2, \dots, m, \quad \text{and} \quad \sum_{i=1}^m q_i = 1.$$

Let us first derive p_n , the probability that there are n peers in the ISP. One can split the Poisson arrival with rate λ into m independent Poisson arrival streams. The arrival rate of peers with downloading time τ_i is denoted by λ_i . Thus

$$\lambda_i = \lambda q_i, \quad i = 1, 2, \dots, m.$$

Using similar argument as in the previous section, the number of peers with downloading time τ_i in the ISP is Poisson distributed with mean $\lambda_i \tau_i$, therefore the probability that we have n peers of downloading time τ_i is

$$p_{n,i} = \frac{(\lambda_i \tau_i)^n}{n!} e^{-\lambda_i \tau_i} = \frac{\bar{n}_i^n}{n!} e^{-\bar{n}_i}, \quad i = 1, 2, \dots, m, \quad n = 0, 1, \dots$$

where $\bar{n}_i = \lambda_i \tau_i = \lambda q_i \tau_i$. The number of peers with downloading time τ_i is independent of the number of peers with other downloading time in the ISP. Since the sum of independent Poisson random variables is again Poisson, it follows that the total number of peers in the ISP p_n is Poisson distributed.

$$p_n = \frac{\bar{n}^n}{n!} e^{-\bar{n}}, \quad n = 0, 1, \dots$$

where $\bar{n} = \sum_{i=1}^m \bar{n}_i = \lambda \sum_{i=1}^m q_i \tau_i$.

Assume that there are n peers in the ISP at a given time. Similar to the homogeneous case analysis, we know that the content that peer i holds, denoted as v_i , is uniformly and independently distributed over $[0, 1]$. Let X_i be the random variable denoting v_i , we have $X_i \sim U[0, 1], i = 1, \dots, n$. Given p_n and X_i , one can derive the lower bound and upper bound of the cross-ISP traffic similar to Theorem 1 and Theorem 2. The result is

$$e^{-\bar{n}} \bar{n}^{-1/2} I_1(2\sqrt{\bar{n}}) \leq E(d) \leq \frac{1}{\bar{n}} (1 - e^{-\bar{n}}),$$

where $\bar{n} = \lambda \sum_{i=1}^m q_i \tau_i$ and $I_1(x)$ is the modified Bessel function.

In fact, since each distribution function can be approximated arbitrary close by a discrete distribution function, one can conclude that the result holds for general downloading time distribution. ■

Proof for Theorem 4: Since there is no file content within the ISP at time $t = 0$, peers in this ISP should download at least one copy of the file through cross-ISP link. Thus the average cross-ISP traffic generated by each peer, denoted by $E(d)$, satisfies $E(d) \geq 1/n$.

To derive the upper bound of the average cross-ISP traffic, similar to the analysis in the regular arrival case, suppose the downloading time T is a discrete random variable. Its possible values are $\tau_1, \tau_2, \dots, \tau_m$. Without loss of generality, we assume that $\tau_1 \leq \tau_2 \leq \dots \leq \tau_m$. Peers arrive to the ISP at the same time $t = 0$, and peers may depart the system at time $t = \tau_i, i = 1, 2, \dots, m$.

Let d_i denote the incoming cross-ISP traffic during the time interval $[\tau_{i-1}, \tau_i]$ ($[0, \tau_1]$ for d_1). Let D denote the total incoming cross-ISP traffic during the whole flash crowd downloading, we have $D = \sum_{i=1}^m d_i$.

Consider those peers which depart at $t = \tau_1$. The incoming cross-ISP traffic generated during $[0, \tau_1]$ is d_1 , which is one copy of the file. After the departure of peers at τ_1 , the maximal progress of downloading in the ISP at time τ_1 are those peers who will finish at τ_2 , their progress at this time is τ_1/τ_2 . Thus during $[\tau_1, \tau_2]$, the internal peers will at most download $(1 - \tau_1/\tau_2)$ of the file context from external peers, i.e. $d_2 \leq 1 - \tau_1/\tau_2$. Similarly, one can consider the time interval $[\tau_{i-1}, \tau_i]$, the cross-ISP traffic $d_i \leq 1 - \tau_{i-1}/\tau_i$. Therefore, the total cross-ISP traffic during $[0, \tau_m]$ is

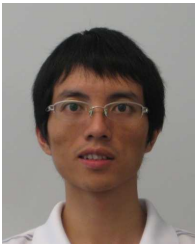
$$\begin{aligned} D &= \sum_{i=1}^m d_i \leq 1 + \sum_{i=2}^m (1 - \frac{\tau_{i-1}}{\tau_i}) = m - \sum_{i=2}^m \frac{\tau_{i-1}}{\tau_i} \\ &\leq m - (m-1) \left(\frac{\tau_1}{\tau_m} \right)^{1/(m-1)} \quad m \geq 2. \end{aligned}$$

The function $y = x - (x-1)a^{1/(x-1)}$ is an increasing function of x when $x \geq 2$, $0 < a \leq 1$. Thus

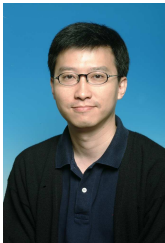
$$D \leq \lim_{m \rightarrow \infty} m - (m-1) \left(\frac{\tau_1}{\tau_m} \right)^{1/(m-1)} = 1 + \log \left(\frac{\tau_m}{\tau_1} \right).$$

Since there are n peers, the average cross-ISP traffic each peer generated, denoted by $E(d)$, satisfies

$$E(d) \leq \left(1 + \log \left(\frac{\tau_{max}}{\tau_{min}} \right) \right) / n. \quad \blacksquare$$



Minghong Lin received the B.S. degree in Computer Science from University of Science and Technology of China, and the M.Phil degree in Computer Science from the Chinese University of Hong Kong. Currently he is a Ph.D. student in Computer Science Department at California Institute of Technology.



John C.S. Lui received his Ph.D. in Computer Science from UCLA. He is currently the chairman of the Computer Science & Engineering Department at the Chinese University of Hong Kong. His research interests span both in systems as well as in theory/mathematics with the emphasis on the robustness, scalability, and security issues on the Internet. John received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award, as well as the co-recipient of the Best Student Paper Awards in the IFIP WG 7.3

Performance 2005 and the IEEE/IFIP Network Operations and Management (NOMS) Conference. He is an associate editor in the Performance Evaluation Journal, IEEE-TC, IEEE-TPDS and IEEE/ACM Transactions on Networking. John was the TPC co-chair of ACM Sigmetrics 2005, QTNA'09, and the General Co-chair for ICNP 2007.



Dah-Ming Chiu received the B.Sc degree in Electrical Engineering from Imperial Collage, University of London, and the Ph.D degree from Harvard University, in 1975 and 1980 respectively.

He was a Member of Technical Staff with Bell Labs from 1979 to 1980. From 1980 to 1996, he was a Principal Engineer, and later a Consulting Engineer at Digital Equipment Corporation. From 1996 to 2002, he was with Sun Microsystems Research Labs. Currently, he is a professor in the Department of Information Engineering in The Chinese University

of Hong Kong. He is known for his contribution in studying network congestion control as a resource allocation problem, the fairness index, and analyzing a distributed algorithm (AIMD) that became the basis for the congestion control algorithm in the Internet. His current research interests include economic issues in networking, P2P networks, network traffic monitoring and analysis, and resource allocation and congestion control for the Internet with expanding services. Two recent papers he co-authored with students have won best student paper awards from the IFIP Performance Conference and the IEEE NOMS Conference. Recently, Dr Chiu has served on the TPC of IEEE Infocom, IWQoS and various other conferences. He is a member of the editorial board of the IEEE/ACM Transactions on Networking, and the International Journal of Communication Systems (Wiley).