PreeN: Improving Steady-State Performance of ISP-Friendly P2P Applications

S.M. Saif Shams*, Paal E. Engelstad, and Amund Kvalbein

Simula Research Laboratory, Norway
saif@simula.no, Paal.Engelstad@telenor.com,
amundk@simula.no
http://www.simula.no

Abstract. Much effort has been put into making P2P applications ISP-friendly, i.e. finding a solution that reduces inter-ISP traffic while maintaining the high file-sharing efficiency of existing P2P applications. Related works have analyzed different ISP-friendly P2P solutions by simulations. However, most of these simulation models have two common assumptions; 1) the peer arrival process follows a flash-crowd scenario, 2) the distribution of peers follows a uniform distribution. Most of the time, both these assumptions are not true for a P2P network. In this paper we show that the peer arrival process and the distribution of peers have a significant effect on the simulation results. We also show that solutions perform differently, e.g. in terms of the amount of inter-ISP traffic and download time, in a flash crowd scenario than in a steady state, limiting the applicability of the results and conclusions reported in previous works. Based on this new insight, we propose a Preemptive Neighbor Selection (PreeN) that makes an ISP-friendly P2P application efficient in steady state scenarios. *abstract* environment.

Keywords: Peer-to-peer, steady state, ISP-friendly P2P.

1 Introduction

P2P applications have gained popularity due to the efficiency in distributing large files among many users. P2P traffic constitutes a significant fraction of the total traffic in the internet [1]. However, there is a conflict of interests between P2P applications and ISPs [1][2]. While a P2P application solely prioritizes downloading performance without being concerned with network efficiency, an ISP prioritizes network efficiency to optimize the performance of all applications. This conflict has led to several proposals for ISP-friendly P2P applications that are less austere to ISPs but still maintain the efficiency of the existing BitTorrent (BT) protocol.

The basis for ISP-friendly P2P applications is to confine the majority of the P2P traffic inside the local ISP whenever possible. These proposals modify the peer selection strategy so that connections to other *local* peers (LP), peers in the

* Corresponding author.

L. Bononi et al. (Eds.): ICDCN 2012, LNCS 7129, pp. 458–472, 2012.

same ISP, are preferred over *remote* peers (RP), peers in other ISPs. To avoid the complexity of evaluating a real P2P network, most of the existing works [3][4] exploit two important assumptions: 1) a large number of peers join at the beginning of the simulation which is a Flash Crowd (FC) scenario, and 2) peers are uniformly distributed over the network. Both these assumptions significantly affect the efficiency of an ISP-friendly P2P application.

In a P2P network, there are frequent peer joining and leaving events [5], which is a significantly different scenario than the simulation models used in the previous works. After first staying in a FC scenario for typically a few hours, a P2P network enters in a Steady State (SS) scenario for a much longer period (for example several days). In the SS, peers join and leave the system spontaneously. A peer that appears later may not find available local peers, and thus, it might be forced to share content with remote peers. As a consequence, an ISP-friendly P2P application may generate more inter-ISP traffic than expected.

The second assumption contradicts the fact that, in a P2P network, the distribution of peers over the network is far from a uniform distribution [6]. When a large number of peers are uniformly distributed over the network, a peer most of the time finds a sufficient number of local peers to connect as neighbors. With this classical scenario, the reported ISP-friendly P2P applications perform nearly optimally in terms of inter-ISP traffic. However, in a realistic P2P network, majority of the ISPs contain only a few peers. Peers in these sparsely populated ISPs contribute with more inter-ISP traffic than the reported results. If an ISP-friendly P2P application is conservative to allow more remote peers, peers in a sparsely populated ISP may suffer by longer download time.

In this paper, we analyse the performance difference of existing P2P applications in different scenarios, and show how those P2P applications perform in SS with a realistic peer distribution model. Also, we propose a preemptive neighbor selection (PreeN) that continuously evaluates the ratio of LP and RP in peers' neighborhoods, which improves the performance of an ISP friendly P2P. Simulation results suggest that using PreeN, peers in a sparsely populated ISP experience a fast downloading performance, while peers in a densely populated ISP generate a less inter-ISP traffic than using other ISP-friendly P2P applications.

The rest of the paper is organized as follows. Section 2 provides an overview of traditional BitTorrent, existing ISP-friendly P2Ps, and the concepts of the FC and SS scenarios. Section 3 gives an illustration of how P2P applications perform in different scenarios. Section 4 describes the proposed ISP-friendly P2P application, PreeN. Section 5 presents our analysis of different ISP-friendly P2Ps in the FC and SS scenarios. Section 6 contains recommendations for future ISP-friendly P2Ps, and Section 7 concludes the paper.

2 Background

2.1 Bittorrent

We follow the description of BT in [7]. In this nomenclature, there are two different types of peers, seeders and leechers, in a BT network. A seeder holds

the complete file whereas a leecher is in the process of downloading the file. An application tracker is a centralized component that stores information about all peers. The tracker provides a short list of participating peers to any peer that requests it. When the total number of participating peers grows larger, the tracker randomly selects a few of them to make a short list (default length is 50). Using that short list, the requesting peer then tries to connect with as many as 35 peers to share the content. A peer requests the tracker again for a list of peers if the number of neighbors falls below 20.

With BT, there are two important algorithms running in every peer, the choking/unchoking algorithm and the piece selection algorithm. The unchoking algorithm determines to which neighbor a peer shall upload content. The algorithm is also referred to as a tit-for-tat mechanism, because it will unchoke the neighbors from which the peer downloaded the highest amount of content in the last 20 seconds. To do that, a peer always keeps track of the contribution from each of its neighbors in the last 20 seconds. After a 10-second interval, a peer reevaluates the tit-for-tat mechanism. A peer simultaneously uploads to the four neighbors that are selected by the tit-for-tat mechanism.

To explore potential new contributors in the neighborhood, the BT protocol includes another unchoking mechanism called 'optimistic unchoke'. This unchoking mechanism randomly selects a fifth neighbor without considering its contribution to the peer. The BT protocol executes the optimistic unchoke every 30 seconds.

The piece selection algorithm is executed in a peer when it is unchoked by a neighboring peer. The algorithm is used to determine which chunk a peer will download from the neighbor. The algorithm will first prioritize incomplete (partially downloaded) chunks. In the absence of such chunks, it will select the rarest chunk in its neighborhood.

2.2 ISP-Friendly P2P Applications

B. Liu et al. [8] describe three different types of locality in an ISP-friendly P2P application, namely, *tracker locality*, *choker locality*, and *piece picker locality*. Of these, only *tracker locality* depends on modification of a tracker.

Tracker locality implies a mechanism of carefully choosing neighbors for a peer so that the peer gets more local peers than remote peers in its neighborhood. A direct consequence of this mechanism is a reduced inter-ISP traffic for all ISPs. *Biased neighbor selection* [3], and *P4P* [4] are examples of this type of ISP-friendly P2P applications.

Choker locality controls the unchoking algorithm of a BT application running in every BT client. With choker locality, a peer always prioritizes local peers when unchoking a peer. It modifies the tit-for-tat mechanism accordingly. OnO [9], and TopBT [6] are examples of this kind of ISP-friendly P2P applications.

Finally, *piece picker locality* focuses on micro-level localization where a peer is prohibited to copy a chunk from a remote peer if the chunk is available to any of its local peers. According to this mechanism, a peer will always collect

rare chunks from outside of the local ISP, enriching the aggregated collection of chunks by the peers inside an ISP. In [11], M. Lin proposed an ISP-friendly P2P application of this kind.

For ISPs, implementing *choker locality* or *piece picker locality* is easier than implementing *tracker locality*, because those P2Ps need modifications only at the client side. These modifications can be implemented by offering add-ons to the client software. To reduce the inter-ISP traffic, an ISP can invest to make an add-on for a particular P2P-client-application and can encourage its end-users to use that add-on. However, the literature [8][12] suggests that a localization mechanism without involvement of a tracker has a very limited influence on the inter-ISP traffic. Due to this reason we keep those mechanisms out of this study.

In this paper we consider ISP-friendly P2P applications that modify the neighbor selection process with the help of an application tracker: *tracker locality*. While *biased neighbor selection* proposes to have only one remote peer out of 35 neighbors, *P4P* proposes to have at least seven (20% of the max. number of neighbors) remote peers in the neighborhood. Authors in P4P believe that 7 remote peers instead of 1 may add robustness against peer failures. As we discuss in section 6, BT is robust against peer failures as long as the peer has a sufficient number of peers in its neighborhood. Thus, the additional number of remote peers, proposed in P4P, increases inter-ISP traffic without giving a significant improvement to the downloading performance.

2.3 Flash Crowd (FC) and Steady State (SS) Scenarios

In the beginning of a swarm, i.e. right after a file has been introduced to a P2P community, the P2P network is in a state where there is a large number of active peers joining within a short period of time, but only one or a few of them possess the whole file. This corresponds to a FC scenario, where initially only a few peers get access to the content. After they finish downloading a few chunks, they start uploading to some of their neighbors. Gradually the content spreads over the network. During this process, most of the peers wait to get the first few chunks before they start utilizing the tit-for-tat mechanism of BT. This waiting time increases the average download time of peers in FC.

Later in a swarm's life, on the other hand, the content is spread throughout the P2P network, and a joining peer starts getting the content as soon as it joins the network. Thus, at this stage the average download time is much lower than the FC. This corresponds to a SS scenario. Formally, G. Veciana and X. Yang define the SS scenario in [13] as a state of the P2P network when the average downloading performance of peers becomes stable. Showing a trace analysis of a BT network, the authors describe that during a FC scenario (what they call a transient regime) the average throughput grows exponentially and becomes stable at one stage when the content is spread over the network. That stage is the starting point of the SS scenario. In a SS scenario, the average throughput remains fairly stable even with a fluctuion of peer arrival rate. When the peer arrival rate becomes steady, the number of peers in a P2P network also becomes steady and predictable [14].

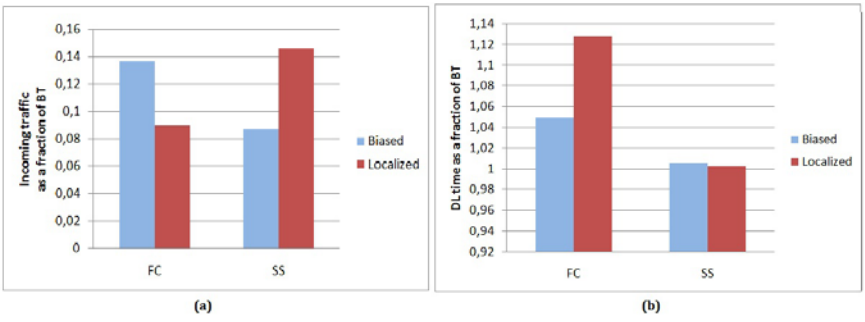

Fig. 1. (a) Inter-ISP traffic in FC and SS. (b)Download time (DL time) in FC and SS scenarios.

Due to the exponential growth of the average throughput, the FC (transient regime) ends quickly. In a swarm's lifetime, while a swarm may remain active for days, FC exists only in the first few hours [13]. Most of the traffic is then exchanged in SS and hence it is important to monitor the performance of ISP-friendly P2P in that scenario. Unlike in a FC, in this stage, there are a comparable amount of peers joining and leaving the network. In this dynamic situation, an application may perform differently than it does in a FC.

3 A Simple Illustration: Performance Difference in Different Scenarios

In this section, we show that P2P applications may perform differently in other scenarios than in a FC scenario with a uniform peer distribution. We present two ISP-friendly P2P applications, namely *Biased* [3] and *Localized*. Here, *Localized* P2P is an ISP-friendly P2P that tries to select all of the neighbors from a local ISP, if available. If there is no available local peers, a peer selects a necessary number of remote peers to have the maximum number of neighbors.

3.1 FC vs SS

In this subsection, we highlight the performance difference of a P2P application running in two different scenarios; FC and SS. We show that a conclusion based on a simulation analysis using a FC scenario may contradict a conclusion based on a SS scenario. In this experiment, a similar set of parameters has been used as is used in previous works [3][4].

Following those existing works, we use a uniform peer distribution over the network where each ISP contains 50 peers. Table 1 shows other parameters used in this experiment. In the FC scenario, 5000 peers join at the beginning of the simulation. Only one of those peers contains the whole file, and it stays until the simulation ends. Other peers leave the P2P network as soon as they finish the

download. Tuning the average number of joining and leaving peers, we establish a SS scenario with around 5000 active peers. In the SS on average 22 peers join and leave the network per second. Figure 1 shows the performance of the *Localized* and *Biased* P2P applications in both scenarios.

The figure shows that in FC the *Localized* application generates a much lower amount of inter-ISP traffic than the *Biased* application does (figure 1). However, in SS, the result is opposite, and *Localized* generates a higher amount of inter-ISP traffic than *Biased*. In FC, most of the peers in an ISP appear at the same time, and, with a *Localized or Biased* P2P, a peer most of the time finds a sufficient number of local peers to make neighbors. Since *Biased* allows less LPs (34 out of 35) than *Localized* (35 out of 35), the inter-ISP traffic generated by *Biased* is higher than the inter-ISP traffic generated by *Localized*.

In SS, the peers arrive at different points in time. As a result, when a peer appears in the system, it often finds that most of the existing local peers have the maximum number of neighbors, and thus, with *Localized* P2P, a peer is pushed to communicate with more RPs than expected. *Biased*, on the other hand, is allowing only one RP in each request. For this reason, in SS, *Localized* generates a higher amount of inter-ISP traffic than *Biased*.

The downloading performance of P2P applications also varies based on in which scenario the applications are executing; FC or SS. In FC, both *Biased* and *Localized* have longer download time than BT, but, in SS, both P2Ps perform as efficiently as BT (figure 1).

In FC, only a few (in this case one) peers contains the file, and most of the peers wait for the first few chunks. With *Localized*, peers in one ISP have very limited communication with peers in other ISPs, Due to this reason, global spreading of content becomes very slow. *Biased*, however, allows one RP for each peer, and thus, it spreads the content faster than *Localized*. As a consequence, in FC, the downloading performance of *Localized* is less efficient than *Biased*. Because of the random peer selection, BT spreads the content quickly over the network reducing the waiting time for all peers. Thus, BT becomes the fastest P2P application in this scenario.

In SS, most of the peers contain parts of the whole file. A new peer starts getting the content as soon as it joins the system. The long waiting time in FC no longer exists in SS. Moreover, peers, who are using *Localized*, get a chance to connect with RPs, due to the sporadic peer arrival process. As a result, all those three P2P applications have the similar downloading performance.

In summary, the conclusion drawn in previous works, which assume the FC scenario, might not be applicable to the majority of a swarm's lifetime, which resembles a SS scenario.

3.2 Uniform vs Zipf Peer Distribution

In this subsection, we show that a more realistic peer distribution may change the conclusion of a P2P analysis that uses a uniform peer distribution. A recent work [6] shows that the distribution of peers follows closely a Zipf distribution. Distributing 5000 peers based on the Zipf distribution, we simulate a FC scenario,

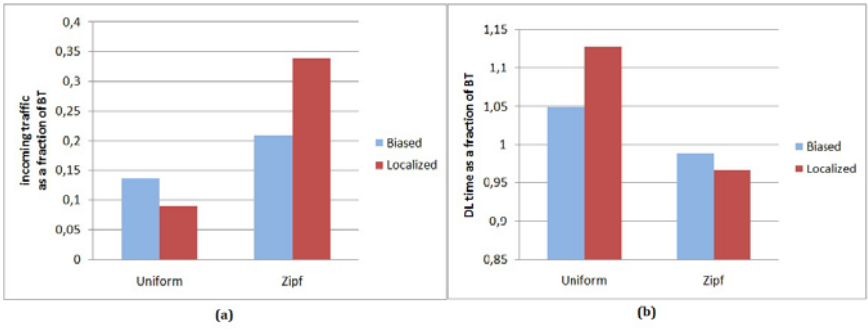

Fig. 2. (a)Average inter-ISP traffic increases with Zipf distribution (b) With Zipf distribution of peers, average download times of biased and Localized become comparable with BT

since previous works use the FC scenario, and compare the result with the result from the simulation using 5000 uniformly distributed peers.

Figure 2 shows that with a uniform peer distribution *Localized* generates less traffic than *Biased*, while the opposite is the case with the Zipf distribution. Likewise, *Localized* has a higher download time than *Biased* with the uniform distribution, while the opposite is true for the Zipf distribution.

In summary, the conclusions drawn in previous works, which assume a uniform peer distribution, might not be applicable to a realistic scenario with a Zipf-like peer distribution.

4 Proposed PreeN Application

In this section we propose a new ISP friendly P2P algorithm that is adapted to a steady state scenario and which performs well with peers localized according to the Zipf distribution.

This proposal includes a modification of the peer selection algorithm of the traditional Bittorrent application, requring modifications both to the application tracker and to the P2P client. This proposal implements a biased neighbor selection in two steps. First, the application tracker provides a biased peer list to a peer, and then a peer uses a pre-emptive mechanism when accepting neighboring request.

4.1 Modification to the Application Tracker

In the application tracker, we propose to modify only the neighbor selection method of the BT protocol. In our proposal, the tracker returns a sub set of 35 active peers with corresponding autonomous system number (ASN). Among these 35 peers, (35-k) peers are LPs and the rest of the k peers are RPs. We prefer that the value of k is 1, similar to *biased neighbor selection* [3]. However,

with the *biased neighbor selection*, a peer gets a fewer number of neighbors if there is insufficient number of local peers, because it is rigid to provide only one RP per response. We propose that when there is an insufficient number of local peers, the application tracker is free to choose peers from other ISPs to complete the list of 35 peers. This freedom will protect the downloading performance of peers with a few or no available LPs.

To map an IP address to ASN, the tracker may use existing online services, for example CYMRU [15]. Along with the facility of a single request, CYMRU also has the facility of block requests that provides the ASNs of a set of IP addresses. During the initial communication between a client and application tracker, the tracker lets a peer know about the peer's ASN.

4.2 Modification in the Client Side

During the initial communication with the tracker, a peer learns about its own ASN. Later, the ASN is used during contacting other peers. Using the ASNs, a peer identifies local requests or remote requests. After getting a list of peers from the tracker, the peer contacts with those listed peers to make neighboring relations. The peer keeps the information of its neighbors' ASNs to implement the pre-emptive mechanism. When a peer receives a neighboring request, it accepts the request if the requested peer has space in its neighborhood. However, if it has the maximum number of neighbors, the peer uses pre-emptive mechanism.

Pre-emptive Mechanism. According to this mechanism, a local peer is given priority over an exiting remote neighbor. If a requested peer has the maximum number of neighbors with more than k RPs and the request is made from a local peer, the requested peer closes a connection with an existing RP and accepts the local peer as a neighbor. However, when a peer with the maximum number of neighbors receives a request from a RP, the request is silently discarded.

4.3 What Is the Value of k

In an ISP-friendly P2P, RPs are necessary to bring new content inside an ISP or spread the content globally to other ISPs. However, the necessity of bringing new content from other ISPs depends on the diversity of chunks inside the local ISP.

Let us try to measure the variety of chunks available inside an ISP during SS. Let p be the probability of having a particular chunk by a peer, n be the number of peers in an ISP, and f be the number of chunks in the file. Thus, the probability that none of the peers in the ISP has a particular chunk is, $P = (1-p)^n$ The expected number of unavailable chunks within the ISP is $f.(1-p)^n$.

Now, the question is what is the value of p in SS? In an SS, on average a specific number of peers join and leave the system. If we sort all peers by their age in ascending order and plot a bitmap of downloaded chunks, we will find a bitmap similar to Fig 3(a). This is because they have on average equal

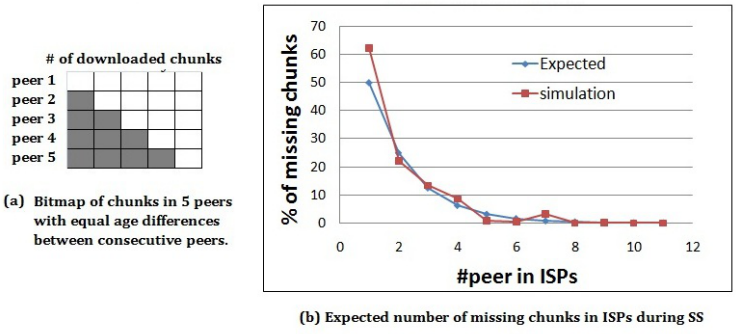

Fig. 3. (a) bitmap of chunks in a group of peers in SS. A dark block represents a downloaded chunk(b) Expected number of unavailable chunks in an ISP with different number of peers.

age differences between consecutive peers and the average downloading rate is steady (SS definition). Due to the rarest chunk mechanism in BT, a group of peers together will have equal number of different chunks in them, if they are well connected with each other. From Fig 3(a) it is clear that the probability of having a chunk by the group is 0.5. Using the number of chunks equals to 100 (f=100), we find the expected number of unavailable chunks in an ISP with different number of peers, Fig 3(b). From the figure we see that in an SS, an ISP contains at least one copy of each chunk, with a high probability, if it hosts more than 7 peers. Simulation results also support this analysis. This implies that in an SS scenario, content is well distributed over the network and we can agreesively reduce the required number of RPs.

However, with a biased neighbor selection, if we choose no RP (k=0) at all, then there is a chance that a group of 35 peers may create a clique disconnecting themselves from the rest of the network. Bindal et al. [3] show that a biased neighbor selection without any RP may end up in a very long donwloading time. Authors also suggest to use 1 RP in a neighborhood in a FC scenario. Considering both FC and SS, we recommend that proposed preemptive neighbor selection should use the ratio of local and remote peers as 34:1.

5 Evaluation of the Proposal

In this section, we describe our simulator, performance metrics, and then we analyse our simulation results.

5.1 Simulator Details

To simulate large scenarios with more than 5000 active peers, we develop a flow-based simulator. The simulator has a network model that includes configurable

upload bandwidth, download bandwidth, and the maximum number of concurrent outgoing flows for each peer. We assume that upload or download bandwidth of a peer is the bottleneck bandwidth for a flow.

Our discrete-event simulator also models the major parts of a trivial Bittorrent application including neighbor selection, tit-for-tat, optimistic unchoking, local rarest first policy, etc. In case of an ISP-friendly P2P, we replace the neighbor selection process used in BT.

We follow the TCP implementation of R. Bindal [3] and H. Xie [4] in our model. In the simulator, we consider long-term average TCP throughput as a throughput on a path. Multiple flows on a link share the bandwidth of a link equally. On an event of a new flow arrival or departure, bandwidths of affected flows are recalculated. Since Bittorrent uses a request queue to keep the TCP flow continuous, the assumption of steady performance of TCP will not significantly influence the results [16].

5.2 Performance Metrics.

An ISP-friendly P2P application has two different objectives; first, it tries to improve the performance of the application and second, it tries to minimize the inter-ISP traffic. In this paper, we used two different metrics to measure these two criteria.

Download Time. By download time, we refer to the time a peer needs to download the whole file. The time starts when a peer joins the P2P system. The download time is the primary metric to measure the performance of a P2P application.

Number of Incoming Chunks to an ISP. The value is the ratio of the number of chunks received from RPs by the peers in an ISP to the total number of chunks received by the peers in that ISP. We do not consider the outgoing traffic from an ISP because the amount of incoming traffic is on average equal to the amount of outgoing traffic.

In this calculation, transit traffic is not considered. By transit traffic we mean the traffic that is generated from and destined to other ISPs.

5.3 Simulation Setup

Unless otherwise stated, we use the configuration parameters mentioned in Table 1 for our experiments.

Network Topology. For an ISP-level network topology, we adopt the topology model presented in [17], which captures several key properties of the Internet topology, including the hierarchical structure given by policy-based routing, the power-law degree distribution for inter-ISP relationship, and the specific ratio among number of ISPs in different tiers.

Table 1. Configuration parameter used in the simulation

Parameter	Value
Number of Peers	5000
Number of ISPs	100
Number of initial seeders	1
File size	25.6 MB
Number of Chunks	100
Upload bandwidth	1 Mbps
Download bandwidth	2 Mbps
Number of neighbors of each peer	35
Number of tit-for-tat flows	4
Number of optimistic unchokes	1
Avergae number of joining peers (in SS)	22/sec

Distribution of Peers over the Network. In reality, the number of peers that are part of a swarm will vary widely across ISPs. Experimental results suggest that the number of peers in different ISPs follow the Zipf distribution with alpha equals to 0.99 [6].

The Zipf distribution is a long tailed distribution. Most of the ISPs, in this distribution, are sparsely populated ISP from where only a few peers (less than 15) participate in the swarm. There are a few densely populated ISPs that contain more than 100 peers. Sparsely populated ISPs together contain a large portion of the whole P2P swarm.

Steady State. In this simulation model, we use poisson distribution for the peer arrival process. By tuning the average peer arrival rate, we create a SS scenario with around 5000 active peers. With the configuration parameters given in table 1, we find the arrival rate as 22 peers per second.

5.4 Simulation Results

In this section, we show that ISP-friendly P2P applications have different results in a SS scenario than in a FC scenario. Also, we show that proposed PreeN application can handle a SS scenario more efficiently than exisiting applications. To compare the efficiency of PreeN application, we compare PreeN with other two existing applications, *Biased* and *P4P*.

With an ISP-friendly P2P, in FC, peers in different ISPs have similar average download times, but, in SS, they have different average download times (figure 4 (a) and (b)). There are two main reasons behind it. The first one is that, in FC, neighborhoods remain unchanged for a long time, but, in SS, neighborhoods change frequently due to sporadic peer joining and leaving events. When a peer leaves, it reduces the neighborhood of another peer shrinking the possibility of getting optimistic unchok from its neighbors. Thus, with a small neighborhood,

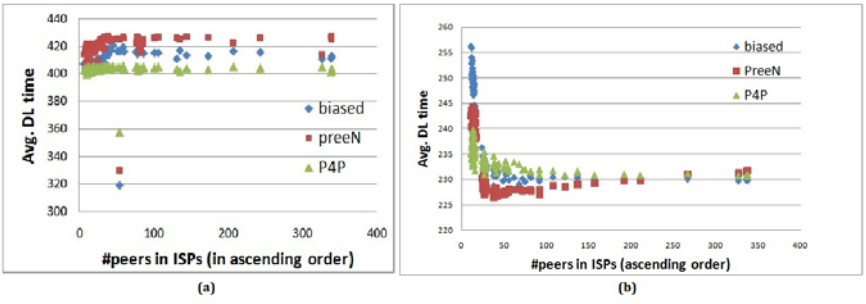

Fig. 4. (a) In FC, average DL time does not vary much across ISPs except the ISP with the initial seeder. (b) In SS, average DL time of peers in different ISPs varies.

a peer will have a less efficient downloading performance than with a large one. When a peer joins, in general, it requests more LPs than RPs. As a consequence, in SS, peers in sparsely populated ISPs have smaller neighborhoods resulting longer download time than others.

The second reason is that, in FC, only a few peers (in this simulation only one initial seeder) have the content, but, in SS, most of the peers have some content. As a result, in FC, a peer waits for the first few chunks, but, in SS, a peer gets the first few chunks very quickly. The waiting time is so high that the downloading performance differences among peers in different ISPs and performance difference among different ISP-friendly P2P applications become minor. In general, the average download time in a FC scenario is much higher than the average download time in SS. For BT, they are around 402 sec. and around 233 sec. respectively.

With *Biased*, the majority of the peers, those are from sparsely populated ISPs, suffer by a significantly longer (more than 10%) average download time than with BT (figure 4 (b)). Since, P4P and PreeN allow a peer to have the maximum number of neighbors, even there are an insufficient number of local peers, they perform faster than *Biased* in terms of download time.

In terms of the needs for the inter-ISP traffic, there is a difference between FC and SS. In FC, except for the ISP with the initial seeder, all other ISPs contain no chunks at all. However, in SS, most of the peers have some content in them. Thus, as we see in figure 5 (a) and (b), all of the P2P applications generate less inter-ISP traffic in SS than in FC. However, with the sporadic peer arrival and leaving events, in SS, ISP-friendly P2P applications end up with a higher number of average remote neighbors than the recommended number of RPs. For example, in a densely populated ISP, *Biased* has on average five RPs, while it recommends only one RP. PreeN adjusts this LP and RP ratio over the time, and thus, it generates 50% less inter-ISP traffic than what *Biased* does in a densely populated ISP.

Although, both *Biased* and PreeN have the provision of one RP, in SS, for densely populated ISPs, PreeN reduces more inter-ISP traffic than *Biased* because of its preemptive neighbor selection. With PreeN, a peer first starts with a

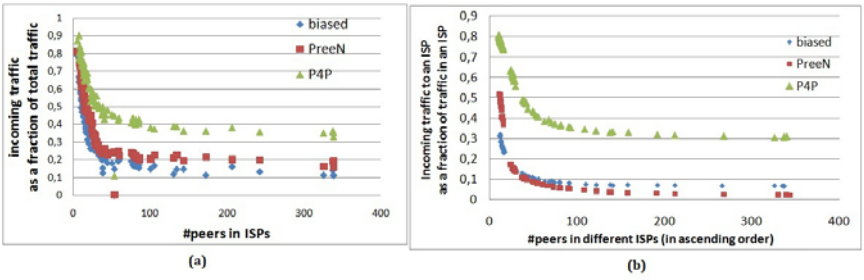

Fig. 5. (a)Incoming traffic to ISPs in FC. (b)Incoming traffic to ISPs in SS.

maximum number of neighbors and then, continuously updates its neighborhood with more local peers, if there are more than one RPs. However, with *Biased*, a peer starts with only one RP and other available LPs, then it requests the tracker for a new list of peers, if there is an insufficient number of available LPs. (The ISP may have many peers in it, but there may not be 34 available local peers.) In this case, the requesting peer gets a new list with another RP and the number of RPs in its neighborhood increases. With *Biased*, a peer can not discard an existing neighbors. Thus, in a densely populated ISP, with *Biased* the average number of RPs is larger than with PreeN.

PreeN allows a peer to have a maximum number of neighbors even with an insufficient number of local peers. With the increment of the number of joining peers, PreeN updates the neighborhoods in a distributed manner. Thus, PreeN adapts itself with different types of ISPs ranging from a very sparsely populated ISP to a densely populated ISP. For sparsely populated ISPs where content is scarce, PreeN helps the peers by providing additional RPs. For densely populated ISPs, PreeN reduces inter-ISP traffic by adjusting the ratio of local and remote peers.

6 Discussion

A BT protocol is robust against peer failures because it has a much larger number of neighbors than the number of concurrent uploaders. Whenever the number of neighbors goes below 20, a peer contacts with the application tracker to get a new list of active peers. Thus, unless most of the neighbors of a peer leave the system at once, which is very unlikely, the download performance will remain unaffected. Due to this reason, in a dynamic situation like SS, we see a consistent downloading performance among peers in ISPs with more than 20 peers (Fig 4 (b)). Based on this result, we say that for densely populated ISPs, adding more remote peers may increase inter-ISP traffic without improving downloading performance in SS.

A general challenge for ISP-friendly P2P methods is the lack of incentives for the end user. Native Bittorrent is very efficient in utilizing the available uploading

capacity [16]. Hence, ISP-friendly P2P methods that aim at localizing traffic will normally not be able to improve on Bittorrent's download times, as long as the local upload capacity is the limiting factor for the throughput of a connection between two peers. This is a reasonable assumption given today's generally well-provisioned core networks, and is used in our model. It may be argued [4] that reducing the network delay between peers will lead to a higher throughput, since a shorter roundtrip time might allow TCP to better utilize the available bandwidth. Measurement studies, however, indicate that this effect is very limited [6]. As mentioned by Choffnes and Bustamente [9], there exists ISPs that discriminate inter-ISP flows and give them a lower bandwidth than local flows. In such an environment, ISP-friendly P2P will also improve performance for end users. However, given the current focus on network neutrality, this service model has not seen widespread deployment.

7 Conclusion

Analysing the performance in FC, it is really hard to infer the performance of an ISP-friendly P2P application in SS. SS exists much longer time than FC and thus, it is important to analyse the performance of a P2P application also in this stage. Existing ISP-friendly P2P methods are based on preferring local peers over remote peers. More RPs increase the amount of inter-ISP traffic, but help a peer to get the content quickly. In general, if we allow a large number of RPs in a neighborhood, as P4P does, in SS, it will help peers in a sparsely populated ISP with lower download time, but in a densely populated ISP it will increase a lot of inter-ISP traffic. On the other hand, if we restrict the number of RPs, as *Biased* does, it will reduce the inter-ISP traffic a bit, but peers in sparsely populated ISPs will suffer significantly. Proposed PreeN can abridge the performance gap between *Biased* and P4P, and may provide the downloading performance as efficient as P4P, and traffic reduction lower than *Biased*.

References

1. Karagiannis, T., Rodriguez, P., Papagiannaki, D.: Should Internet Service providers Fear peer-Assisted Content Distribution, IMC, Berkeley (2005)
2. Aggarwal, V., Feldmann, A., Scheideler, C.: Can ISPs and P2P systems cooperate for improved performance? ACM CCR (July 2007)
3. Bindal, R., et al.: Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In: IEEE ICDCS (2006)
4. Xie, H., Yang, Y., Krishnamurthy, A., Liu, Y., Silberschatz, A.: P4P: provider Portal for Application. In: SIGCOMM, Washington, USA (2008)
5. Izal, M., Urvoy-Keller, G., Biersack, E.E., Felber, P., Hamra, A.A., Garces-Erice, L.: Dissecting BitTorrent: Five months in a torrents lifetime. Presented at the 5th Passive and Active MeasurementWorkshop (PAM), Antibes Juan-les-Pins, France (April 2004)
6. Ren, S., Luo, T., Chen, S., Guo, L., Zhang, X.: TopBT: A topology-aware and infrastructure-independent Bittorrent client. In: INFOCOM (2010)

7. Cohen, B.: Incentives build robustness in BitTorrent. In: Proc. 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley (June 2003)
8. Liu, B., Cui, Y., Lu, Y., Xue, Y.: Locality-Awareness in BitTorrent-Like P2P Applications. IEEE Transactions on Multimedia 11(3) (April 2009)
9. Choffnes, D.R., Bustamante, F.E.: Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In: SIGCOMM (2008)
10. Sheng, L., Wen, H.: Reducing cross-network traffic in P2P systems via localized neighbor selection. In: ChinaCom (2009)
11. Lin, M., Lui, J.C.S., Chiu, D.-M.: An ISP-Friendly File Distribution Protocol: Analysis, Design, and Implementation. IEEE Transactions on Parallel and Distributed Systems (2010)
12. Piatek, M., Madhyastha, H.V., John, J.P., Krishnamurthy, A., Anderson, T.: Pitfalls for ISP-friendly P2P design. In: The Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII), New York City, NY, USA (October 2009)
13. de Veciana, G., Yang, X.: Fairness, incentives and performance in peer-to-peer networks. In: The Forty-first Annual Allerton Conference on Communication, Control and Computing, Monticello, IL (October 2003)
14. Qiuand, D., Srikant, R.: Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In: SIGCOMM (2004)
15. IP to ASN mapping, `http://www.team-cymru.org/Services/ip-to-asn.html`
16. Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analyzing and Improving BitTorrent Performance. In: IEEE Infocom (2006)
17. Elmokashfi, A., Kvalbein, A., Dovrolis, C.: On the Scalability of BGP: the roles of topology growth and update rate-limiting. In: CoNext 2008, ed. by ACM (2008)