Improving Traffic Locality in BitTorrent via Biased Neighbor Selection

Ruchir Bindal, Pei Cao and William Chan
Department of Computer Science, Stanford University
rbindal@stanford.edu
Jan Medval, George Suwala, Tony Bates, Amy Zhang
Cisco Systems, Inc.

***Abstract*— Peer-to-peer (P2P) applications such as BitTorrent ignore traffic costs at ISPs and generate a large amount of cross-ISP traffic. As a result, ISPs often throttle BitTorrent traffic as a way to control the cost. In this paper, we examine a new approach to enhance BitTorrent traffic locality, *biased neighbor selection*, in which a peer chooses the majority, but not all, of its neighbors from peers within the same ISP. Using simulations, we show that biased neighbor selection maintains the nearly optimal performance of BitTorrent in a variety of environments, and fundamentally reduces the cross-ISP traffic by stopping it from growing linearly with the number of peers. A key reason for its performance is the rarest first piece replication algorithm used by BitTorrent clients. Compared with existing locality-enhancing approaches such as bandwidth limiting, gateway peers, and caching, biased neighbor selection requires no dedicated servers and scales to a large number of BitTorrent networks. Furthermore, it can be combined with bandwidth limiting and caching to improve their performance further.**

Keywords: Peer-to-Peer, BitTorrent, performance, traffic locality, neighbor selection.

Technical area: Peer to Peer

I. Introduction

P2P content distribution applications such as BitTorrent [Coh03] have fundamental advantages over the traditional client-server model (i.e. web sites) and the fixed-infrastructure content distribution networks (i.e. Akamai). They are self-scaling; the supply of bandwidth grows linearly with the demand. They utilize the bi-sectional bandwidth among the peers, instead of channeling the traffic from a fixed set of points of presences on the Internet. Finally, they require little additional infrastructure, but merely better utilize the existing network bandwidth.

Thus, it should be no surprise that BitTorrent is wildly popular and a significant source of traffic on the Internet. According to a recent CNN report, BitTorrent traffic constitute at least 20% of the entire traffic volume on the Internet [ECo05]. Other vendors have observed that P2P applications, including BitTorrent, account for over 60% of the traffic seen by an ISP [Cac]. While the estimates wary, it's clear that BitTorrent is a force to be reckoned on the Internet.

Unfortunately, BitTorrent-like applications present significant traffic-engineering challenges for Internet service providers (ISPs). An ISP typically pays a tier-1 "core" ISPs for connectivity to the broad Internet, and traffic between the ISP and the outside world is costly for the ISP [Nor03]. Unfortunately, current implementations of BitTorrent ignore the underlying Internet topology or ISP link costs, and set up data transfers among randomly chosen sets of peers distributed around the Internet. As a result, they generate a significant amount of cross-ISP traffic and increase the operating cost of an ISP significantly.

The method most often deployed by ISPs to control BitTorrent traffic is "throttling", or bandwidth limiting. Since BitTorrent traffic typically runs over a fixed range of ports (6881 to 6889) [Coh05] and uses formats easy to decode, traffic shaping devices such as [Pac], [Inc], [PC], [San] are deployed to limit the amount of bandwidth consumed by the BitTorrent protocol. However, the main effect of throttling is to slow down the content transfer. It does not address the fundamental concern of the ISP, which is to improve the locality (i.e. reduce the cross-ISP traffic) of those transfers. As a result, throttling reduces the rate of cross-ISP traffic at the expense of increased download time and poorer user experience, a trade-off that the ISP would rather avoid.

At the root of the ISPs' dilemma is the following question: if there are N users within the ISP all wanting to download the same content, does the content travel into the ISP N times? In today's BitTorrent implementations, the answer is yes. To be precise, if the number of peers in the network is G, then on average the content traverses into the ISP $N*(1-N/G)$ times before all N users finish the download. Since N is typically much smaller than G, $N*(1-N/G)$ is close to N.

However, is the above answer an artifact of existing implementations, or is it a fundamental property of BitTorrent? The success of BitTorrent is in no small part due to its high performance in terms of user experienced download time, and Many analytical and simulation studies [MV05], [BHP05], [YdV04], [QS04] have shown that the existing BitTorrent algorithm is nearly optimal in that regard. Yet, all studies assume that when peers select neighbors, they select randomly among all neighbors, which is the root cause of high cross-ISP traffic. Yet, it is not known if BitTorrent still performs well if the random neighbor selection algorithm is changed. Put it another way, the studies show that random neighbor selection, which leads to high cross-ISP traffic, is a sufficient condition of performance optimality; but is it also a necessary condition?

This paper answers the above question with extensive simulations. We rely on simulations since it is difficult to capture all the relevant mechanisms in BitTorrent clients in an analytical model. Our simulator accommodates internal network bottlenecks, rather than assuming that bottlenecks are peers' upload links. The simulator faithfully captures the bit-vector exchanges among the peers, choking/unchoking, concurrent uploads and piece selection algorithms of the dominate BitTorrent client, since they are critical for determining cross-ISP traffic.

Our main conclusion is that biased neighbor selections, in which a peer chooses the majority, but not all, of its neighbors from peers within the same ISP, can reduce cross-ISP traffic significantly while keeping the download performance nearly optimal, in virtually all cases. Specifically,

- As long as the original seed has moderately high upload bandwidth (e.g. four times the prevailing upload bandwidth of the peers), biased neighbor selection result in no degradation in download times;
- The "rarest first replication" algorithm is key to the success of biased neighbor selection. Though other studies have shown that random piece selection algorithm can still lead to optimal performance in regular BitTorrent [BHP05], [MV05], we find that it does not work well with biased neighbor selection.
- For an individual ISP, if there are higher bandwidth peers external to the ISP, then the effectiveness of biased neighbor selections at reducing the traffic into the ISP is reduced. In this case, an additional mechanism, bandwidth throttling, can be combined with biased neighbor selection to reduce cross-ISP traffic with little impact on download performance.

We also compare biased neighbor selection with three existing traffic-shaping approaches: pure bandwidth throttling, a single externally-connected peer, and a caching solution. We show that:

- While bandwidth throttling does push peers toward intra-ISP nodes, its effect is limited by the initial neighbor selection of the peer, and combining bandwidth throttling with biased neighbor selection is much more effective.
- Using a single peer connect to the external nodes results in significant increase of download time. This is the case even if the designated peer has high upload and download bandwidth.
- For a cache to avoid increasing download time, a cache would have to have a download bandwidth that grows linearly with the number of peers inside an ISP, a requirement unlikely to be satisfied. However, if used in combination with a biased neighbor selection technique, a cache only needs to have a bandwidth that is a constant multiple factor over the peer's upload bandwidth for it to be achieve minimum cross-ISP traffic and minimum download time.

Alternatively, biased neighbor selection can also be viewed as a way for BitTorrent peers to bypass network bottlenecks. Our results essentially state that a peer can feel free to favor certain set of peers than others, whatever the reasons maybe, as long as the peer keeps a few randomly-chosen undesirable peers as neighbors. Thus, this technique has the potential to function as an Internet-wide traffic shaping mechanism for BitTorrent.

II. BitTorrent and Related Work

A. BitTorrent Protocols and Algorithms

BitTorrent [Coh03] is a P2P file-sharing application designed to distribute large files to a large user population efficiently. This is achieved by making use of the **upload** bandwidth of all nodes (called peers) **downloading** the file. In the following description, we use the term node and peer interchangeably.

To distribute a file via BitTorrent, the provider of the file divides the file into small blocks (usually 245KB in size), and generates another file containing meta-information about the data file (called the *torrent* file). The provider also runs a *tracker* for this file, either by running a tracker on its own or registering the file with a public tracker. A *tracker* is a central server that keeps track of all nodes downloading this file. The supplier then starts its BitTorrent client. The client automatically detects that it has the complete file and is thereby a *seed* node in the network. The torrent file is then published on the Internet using HTTP and interested downloaders can download it to run their BitTorrent clients with the torrent file as the input. The address of the tracker is embedded in the torrent, allowing peers to contact the tracker.

a) Forming the Random Graph: A peer who is interested in downloading the file must first contact the tracker hosting the file to join the BitTorrent network *of this file*. The network consists of the tracker, the original seed, and all nodes interested in downloading the file. Peers do not have full connectivity in this network. Rather, the network is a random graph.

The graph is formed as the follows. Each peer, p, upon first joining the network, contacts the tracker. The tracker then **randomly** selects C nodes (default C is 35), out of all the nodes in the network, and hands the list back to p. Peer p then initiates connections with those nodes. Later on, other peers joining the network may get p as one of nodes returned by the tracker and initiate connections with p. As a result, p' neighbors in the network include both nodes that p initiates connections to and those that initiates connections to p. Since peers may leave the network at any time, if p's neighbor count drops below a certain threshold (default is 20), p contacts the tracker again to obtain a new list of nodes.

Note that the above description is of the prevailing client implementation. There are a number of BitTorrent client implementations circulating. Some implementation is very aggressive at contacting the tracker; they obtain multiple lists of nodes, and then choose the best ones among them according to certain criteria. Since these clients place a higher burden on the tracker, their use is not recommended, and our paper studies the behavior of the default client implementation only.

b) Choking/Unchoking: The download proceeds mainly by peers exchanging blocks with their neighbors. Peers exchange bit vectors of the blocks in their possession with neighbors, at the beginning and whenever a peer obtains new content. Through the bit vector exchange, the peer p learns the up-to-date content at each neighbor. If a neighbor have blocks that p doesn't have, p sends an "interested" message to the neighbor. The neighbor, however, is not obligated to send blocks to p. Instead, when and if the neighbor sends blocks to p depends on the "choking/unchoking" algorithm in BitTorrent, also called the "tit-for-tat" mechanism.

The "choking/unchoking" algorithm determines, for each peer, among its neighbors who have expressed interests in its content, which of them it should give contents to. All connections are choked by default. If a peer decides to provide contents to another peer, it "unchokes" the connection with the other peer. A peer can upload to multiple peers at the same time; the current default limit on the number of concurrent uploads is 5. Hence, a peer has 5 un-choked connections at a time.

Four of these connections are chosen using a "tit-for-tat" criteria. A node keeps track of its download rate from all neighbors. Then, among the neighbors expressing interest, four with the highest download rate to this peer are un-choked. In other words, a peer rewards other peers who give data to it previously. For the original seed, which has nobody to download from, this decision is made based on the upload rate to the neighbors. So those peers which are downloading quickly from the seed will get a higher preference. This decision to choke/unchoke depends on the download rate as described above and is made periodically (every 10 seconds), after which some of the currently unchoked nodes may be choked and vice-versa.

The "tit-for-tat" mechanism naturally bias the traffic between peers toward the higher bandwidth routes. However, this bias is limited by the neighbor selection set up in the "network forming" step.

The last of the 5 connections is chosen according to a different mechanism, *optimistic unchoke*. In order to jump-start brand new peers and to find peers that may have a better upload rate to it, a node chooses a neighbor at random to unchoke, irrespective of the download rate from that peer. This random optimistically unchoked node is chosen once every 30 seconds and the chance of a new node being selected for an optimistic unchoke is three times that of a node that already holds some blocks.

c) Piece Selection: Once a peer p expresses interests in a neighbor's blocks, and the neighbor unchokes the connection with p, p can request a block from the neighbor. Exactly which block is read from the neighbor is determined by p using a "rarest first replication" algorithm. That is, among the blocks provided by a neighbor, the block that is least replicated among all neighbors of p is chosen.

Studies have shown that this rarest first selection algorithm is in fact not necessary for the optimal performance of BitTorrent. However, as we will show later in the paper, it is in fact quite important if the graph is not random.

B. Existing Studies on BitTorrent

At its heart, BitTorrent attempts to solve the "broadcasting problem", i.e. disseminating M messages in a population of N nodes in the shortest time. In the setting of the Internet where nodes can communicate in both directions simultaneously and have the same bandwidth, the lower bound on download time is $M + log_2(N)$ unit, where a unit is the time it takes for two nodes to exchange a message. Assuming that all nodes are completely connected, an optimal algorithm that relies on a centralized scheduler is described in [MW]. The algorithm goes in rounds, and establishes matching pairs among nodes in a deterministic fashion. Similar algorithms are also discussed in [YdV04], [GS05].

BitTorrent lacks a central scheduler, does not have a completely-connected graph and is not deterministic. However, BitTorrent appears to work exceedingly well. Simulation studies found that in typical settings of cable modem and DSL nodes the links are almost fully utilized all the time [BHP05], [GS05]. The results indicate that the random algorithms used by BitTorrent lead to nearly optimal performance.

A number of analytical studies support the above observation. In [MV05], BitTorrent is modeled as a coupon replication system, i.e. nodes aim to complete a collection of distinct coupons and exchange coupons with each other. The analysis in [MV05] proves that, as long as the neighbors are chosen either randomly among all peers, or randomly among peers with the same number of coupons, the performance of BitTorrent is asymptotically optimal. In particular, the probability that a peer can find new coupons among its neighbors is $> 1/2$. The optimal performance does not depend on nodes staying around after completing their collections or using the least-replicated-first replication strategy. Other studies using branching processes to model BitTorrent [YdV04] and using fluid dynamics to model BitTorrent [QS04] also came to similar conclusions.

All existing simulation and analytical studies, however, model the case of peers choosing neighbors randomly among all nodes in the network. Unfortunately, such neighbor selection policy is also the root cause of BitTorrent's high cross-ISP traffic. Thus, the goal of our study is to find neighbor selection policies that improve intra-ISP traffic locality while preserving the near-optimal performance of BitTorrent. Our solution, biased neighbor selection, achieves this goal.

In addition to bandwidth throttling, two other obvious methods for reducing cross-ISP traffic are caches [Cac], and "gateway peers" (a gateway peer is the only node inside an ISP that can connect to external peers) [KRP05]. A recent study uses trace-driven study to examine the cross-ISP traffic of the two approaches and found that they are comparable [KRP05]. However, the study did not look into peer download latency. We found that in order for these solutions not to increase download latency, the devices involved (caches or gateway peers) need to have much higher bandwidth than individual peers, and the bandwidth requirement grows as the number of peers inside the ISP grows. In contrast, changing the neighbor selection policy requires no extra infrastructure, and can be combined with these methods to improve them further.

There are numerous measurement studies of BitTorrent traffic on the Internet [Epe], [IUKB$^+$04], [GCX$^+$05]. The studies show that a BitTorrent network typically goes through three stages in its life: flash crowd, steady state and winding down [IUKB$^+$04], and the peer join rate decrease exponentially with time [GCX$^+$05]. Flash crowd occurs when

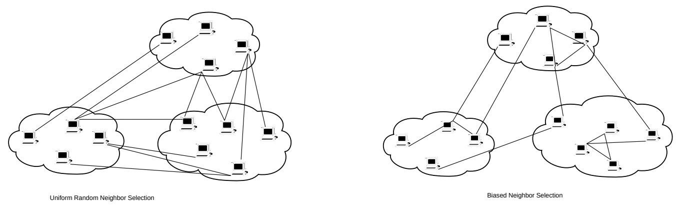

Fig. 1. Uniform random neighbor selection vs. biased neighbor selection. The graph on the left is a result of uniform random neighbor selection, while the graph on the right is a result of biased neighbor selection.

the content is first made available, and can last several days [IUKB+04]. Among the three stages, flash crowd generates the highest rate of traffic and is the most challenging for ISPs to handle. Thus, in this paper, we focus on the flash crowd scenario, though we also validate our results with a Poisson arrival pattern.

Using end systems for content distribution has been studied extensively in recent years. Most schemes build explicit overlay trees [Fra02], [CRZ00], [CDK+03], [PWCS02] or meshes [CDK+03], and are quite different from BitTorrent. Slurpie [SBB04] adopts the same randomized approaches as BitTorrent, and improves upon BitTorrent by combining a randomized back-off strategy and an effective group size estimator to avoid server overloads. It is shown that Slurpie performs better than BitTorrent in heterogeneous bandwidth environments. In our study, we focus on BitTorrent due to its popularity.

Finally, studies have investigated using network coding to improve BitTorrent [GR05]. The use of network coding solves the "last missing block" problem and significantly improves content availability in the network. The use of network coding is complementary to our technique, since we focus on neighbor selection algorithm only.

III. Biased Neighbor Selection: Concept and Implementation

The main technique we use is biased neighbor selection. In biased neighbor selection, a peer chooses its neighbors mostly from those within the same ISP, and only have a few that are outside the ISP. Specifically, we experiment with a parameter k, where for each peer, all but k neighbors are from the same ISP, and only k neighbors are chosen from outside the ISP. If there are fewer than $35 - k$ internal peers, more external neighbors are retained. However, in this case, the peer will contact the tracker again to find more internal neighbors.

In biased neighbor selection, peers within the same ISP form a cluster. They are highly connected. Each peer, however, does keep a few connection to the outside world, giving it a visibility into the available blocks from the outside world. The difference between the normal randomized neighbor selection and the biased neighbor selection is illustrated in Figure 1.

There are two ways to implement biased neighbor selection:

- *By modified tracker and client*: Biased neighbor selection can be implemented easily by changing the tracker and the client. The tracker selects $35 - k$ internal peers and k external peers to hand back to the client. If there are less than $35 - k$ internal peers, the tracker also notified the client to contact it again after certain duration.
  The challenge here lies in informing the tracker of the ISP locality. There are three possibilities to solve the problem. The tracker can use Internet topology maps or AS mappings to identify the ISP. ISPs wishing to preserve traffic locality can also publish their IP address ranges to trackers. Finally, since BitTorrent protocols run over HTTP, the ISP's HTTP proxy can append a new header "X-Topology-Locality" which contains a locality tag. All peers with the same locality tag are assumed to be from the same ISP.
- *by P2P traffic shaping devices*: In recent years, the need of ISPs to control P2P traffic has given rise to a new category of devices that we call P2P shaping devices. Situated along side the edge routers of the ISPs, these devices use deep packet inspection to identify P2P traffic and manipulate them. Representative vendors include CacheLogic [Cac], Sandvine [San] and Cisco's P-Cube appliances [PC]. For BitTorrent traffic which runs on HTTP [wik05], many HTTP proxy appliances can perform the role as well.
  These devices can easily keep tracks of peers inside the ISP for each BitTorrent network. When a peer joins the network, the devices can intercept and modify the responses from the tracker to the peer, and substitute outside peers with internal peers. Furthermore, when it is necessary to change a peer's neighbors (for example, when more internal peers join the network), the device can initiate TCP RESET on the connection between an internal peer and external neighbors, forcing the internal peer to contact the tracker to obtain new neighbors, at which point the device can manipulate the tracker's response.

Since biased neighbor selection can be implemented by P2P shaping devices without changing the clients or the tracker, we believe it is a practical method by ISPs to improve traffic locality.

IV. Evaluation Methodology and Criteria

To understand the interaction of neighbor selection with other BitTorrent algorithms, we built a discrete-event simulator and run the algorithms over a simplified network topology.

A. Representative Network Topology

The basic network consists of 14 ISPs each having 50 peers joining a BitTorrent network. All peers inside the ISP are modeled after cable modem and DSL nodes, and have asymmetric upload/download bandwidth. The upload bandwidth of these peers are 100kbps, and download bandwidth of 1Mbps. We call these peers "cable modem nodes".

All ISPs are assumed to be completely connected. We evaluate both the scenario when no cross-ISP bandwidth bottleneck exists, and the scenario when cross-ISP bandwidth bottleneck exists. In the case of cross-ISP bandwidth bottleneck, we assume that an ISP has a single limited bandwidth link to all of the other ISPs. This models the scenario where an ISP

Fig. 2. Simulated Network Topology

employed a bandwidth throttle on all BitTorrent traffic on its gateways.

In addition to cable modem ISPs, we consider the presence of high-bandwidth nodes that have symmetric links and have bandwidth typically higher than the cable modem nodes. We call these nodes “university nodes”. We assume that each university node has point-to-point links with each AS and also with each other. In our experiments we vary both the number and the bandwidth of these nodes to examine their impact on the BitTorrent network. In real life, a BitTorrent network is likely to have both the high bandwidth university nodes and the cable modem nodes present in the network.

Most of our simulations are conducted with all peers joining the network at once, i.e. the flash crowd scenario. We focus on flash crowd since it is the most challenging for ISPs to handle. Furthermore, we assume that all peers leave the network as soon as they finish download, but the original seed always stays online. Studies have shown that the majority of peers leave soon after they finish download [GCX$^+$05]. However, since the content provider is interested in distributing the content using BitTorrent, it is reasonable to expect that the original seed stays around to see the last of the flash crowd peers finish downloading.

B. Event-Driven Simulation

We built a discrete event simulator to calculate the download time of BitTorrent peers under various algorithms. The discrete event simulator models the following events in the BitTorrent network:

- Join: peers joining the network;
- Leave: peers joining the network;
- Block-Transfer: a block is sent from one peer to another;
- Peer-Report: peer’s periodic contact with the tracker;

Unlike simulators used in other studies, our simulator does not assume that the bottleneck is solely the upload links of the nodes, but rather accommodates bottlenecks in other network links. As a result, it needs to calculate the network delay of each block transfer based on the number of connections sharing bottleneck links and handle multiple bottlenecks.

The simulator calculates the network transfer delay in the following fashion. Every 100ms, the simulator recalibrate the transfer rate of each connection between peers, based on an idealized assumption of equal share and maximum capacity utilization. Each link has an associated upload and download bandwidth. For each link, we record the number of flows passing through the link. Every 100ms, an event is trigger to recalculate the transfer rate of all flow. Specifically, the simulator starts with the link that is most congested, i.e. with the lowest per-connection bandwidth. For each connection going over that link, it then goes through all links used by the connection, and subtracts the connection’s bandwidth from those links. It then finds the next most congested link, and repeat the above calculation.

The above approach assumes idealized performance of TCP. We do not simulate the overheads and dynamics of TCP. Each flow for a particular link is allocated an equal share of the remaining bandwidth. We do not model the propagation delay in the network and latency of control messages.

Each peer’s code is almost “fork-lifted” out of the original BitTorrent implementation. Data transfer events are translated into bytes received. Logic for calculating past download rates from the neighbors are implemented. The choking/unchoking algorithm is implemented faithfully. Each peer holds the bit-vector content of all its neighbors, and implements the “rarest first” replication algorithm.

C. Evaluation Criteria

The main evaluation criteria are two: download time and ISP traffic redundancy.

Measurement of download time include the cumulative distribution function (CDF), the 50th percentile value and the 95th percentile value.

The term “ISP traffic redundancy” means the average number of times the blocks of the content file travels into the ISP until all peers inside the ISP finish their download. The lower the redundancy, the lower the cross-ISP traffic. The lowest redundancy is 1. The highest redundancy is N, where N is the number of peers inside the ISP.

For each experiment, we run the simulation multiple times when different random seeds. The variance in the results from multiple runs is very low, $< 5\%$.

V. Biased Neighbor Selection: Benefits and Performance

Our experiments examine two network settings: a homogeneous network consisting of 700 cable modem nodes spread among 14 ISPs, and a heterogeneous network consisting of a number of university nodes and 700 cable modem nodes. In both cases, the original seed (the one provided by the content provider) is a separate node whose bandwidth varies. In the following discussion, we refer to the BitTorrent network using uniform random neighbor selection as “regular BT”, and the BitTorrent network using biased neighbor selection as “biased BT”.

To motivate biased neighbor selection, we first show that the bandwidth throttling is less than desirable.

A. Effects of Bandwidth Throttling

If ISPs apply bandwidth throttle to BitTorrent traffic, the effects are:

- Significantly increased download time, particularly when the seed has high bandwidth or that there are external high bandwidth nodes (e.g. university nodes);
- Moderate reduction in traffic redundancy, but a failure to reduce it to a very low level.

Without bandwidth throttling, the traffic redundancy is close to N, the number of nodes in the ISP. In uniform random neighbor selection, the number of external neighbors that a peer has on average is $35 * (1 - N/G)$ on average, where N is the number of peers in the ISP, and G is the total number of peers in the BitTorrent network. In homogeneous

ISP bottleneck	50th percentile	95th percentile	traffi c redundancy
no bottleneck	1.0	1.35	46.9
2.5Mbps	1.43	1.59	31.76
1.5Mbps	2.01	2.05	24.88
500kbps	3.33	3.53	21.65

TABLE I

NORMALIZED DOWNLOAD TIME AND TRAFFIC REDUNDANCY UNDER BANDWIDTH THROTTLE, IN A HOMOGENEOUS NETWORK. DOWNLOAD TIME OF 1.0 IS $5{,}312$ SECONDS.

Neighbor Selection	50th percentile	95th percentile	traffi c redundancy
Regular BT	5,312	7,152	46.72
Biased $k = 1$	5,168	6,206	3.44
Biased $k = 5$	5,172	6,281	9.74
Biased $k = 17$	5,220	5,872	21.38

TABLE II

DOWNLOAD TIME AND TRAFFIC REDUNDANCY OF REGULAR VS. BIASED NEIGHBOR SELECTION IN A HOMOGENEOUS NETWORK.

networks where there is no internal network bottleneck, all the neighbors have the same upload rate to the peer and hence they are indistinguishable from each other. As a result, the peer chooses uniformly randomly among its neighbors for uploads. The expected ISP traffic redundancy is thus $N * (35 * (1 - N/G))/35 = N * (1 - N/G)$. Since N is usually much smaller than G, the redundancy is nearly N.

Bandwidth throttling can reduce traffic redundancy moderately, at the expense of increased download time. Table I shows the performance of bandwidth throttling in homogeneous networks. The seed is a separate node with 400Kbps uplink bandwidth. For clarity, the download time results are presented as ratio between the download time and a base download time, which is the 50th percentile download time for the no bottleneck case at 5,321 seconds. Figure 3 shows the CDF of the download times.

The simulation results for the no throttling case match the above analysis; in this case, for each ISP, 50 nodes of the 700 are internal and 650 are external, the expected redundancy according to the analysis is $50 * 650/700 = 46.4$. Note that though a 1.5Mbps leads to halving the redundancy, setting the bottleneck to 500kbps only reduces redundancy slightly further. It appears that beyond a certain level, bandwidth throttling cannot reduce traffic redundancy anymore, and its only effect is increased download time.

Our observations are the following. First, bandwidth throttling is not completely without merit. In the above results, the increase in download time is always less than linear to the reduction in the bottleneck bandwidth. The reason is the tit-for-tat mechanism. In tit-for-tat, when the bandwidth throttle is imposed, a peer naturally biases toward exchanging data with peers inside the same ISP, since internal peers have no bottleneck among them and have high upload rate to each other.

However, the effect of bandwidth throttle at moving traffic toward internal ones is constrained by the initial neighbor selection. If internal peers are simply not neighbors of each other, then they cannot exchange blocks. Thus, under uniform random neighbor selection, bandwidth throttle can only reduce traffic redundancy to a certain level, beyond which it is of no use.

B. *Biased Neighbor Selection in Homogeneous Networks*

The above observation motivates our technique: biased neighbor selection. Namely, for BitTorrent traffic to have ISP locality, the neighbors need to be chosen well.

Table II show the performance for biased neighbor selection, with the number of external neighbors k varying from 1 to 17 (half of all neighbors). Figure 4 shows the CDF of the download times.

The results show clearly that biased neighbor selections reduces the variation in download times among the peers, has median download times similar to regular BitTorrent, and reduces ISP traffic redundancy significantly. The reduction in download time variances has to do with the fact that pieces are replicated more evenly in biased BT, since the estimate of replication ratios of pieces is improved due to the clustering properties of the graph. The variation in k has little impact on download time, and lower k results in lower redundancy. Thus, $k = 1$ should be used. In particular, at $k = 1$, the redundancy is 3.44, meaning that on average, less than 4 copies of each data block enter the ISP in order to satisfy all 50 peers.

How does the redundancy of biased BT vary with number of peers inside an ISP? We increased the number of peers inside each ISP to 75 and 100. Surprisingly, the redundancy *decreases* as the number of peers inside an ISP increases. The redundancy is 2.64 for 75 peers per ISP, and to 1.83 for 100 peers per ISP.

We believe the counter-intuitive results have to do with the "local rarest first" replication policy employed by BitTorrent. When the number of peers inside an ISP is small, each peer has a good view of what blocks are present in the ISP. As a result, they tend to bring in blocks with low replication inside the ISP, and often this block is at the external peer. When the number of peers is large, each peer tends to choose blocks more randomly, and give rare blocks more time to replicate inside the ISP. Hence, the number of blocks replicated tend to be reduced.

We also change the seed bandwidth from 400KBps to 1Mbps. The traffic redundancy of biased BT is virtually unchanged. Thus, in homogeneous networks, the traffic redundancy is mainly determined by the number of peers inside an ISP, and appears to stay under 4 in all cases.

What if only one ISP uses biased neighbor selection while all others use the default algorithm? Figure 5 shows the CDF of download times when only one ISP uses biased neighbor selection with $k = 1$. The figure shows that there is little change in median download time when one ISP uses biased neighbor selection, and the spread between download times is reduced for all peers. The ISP traffic redundancy is increased to 25.2 in this case, since the ISP using biased neighbor selection allows other ISPs to connect to nodes inside the ISP, and the average number of external neighbors for the

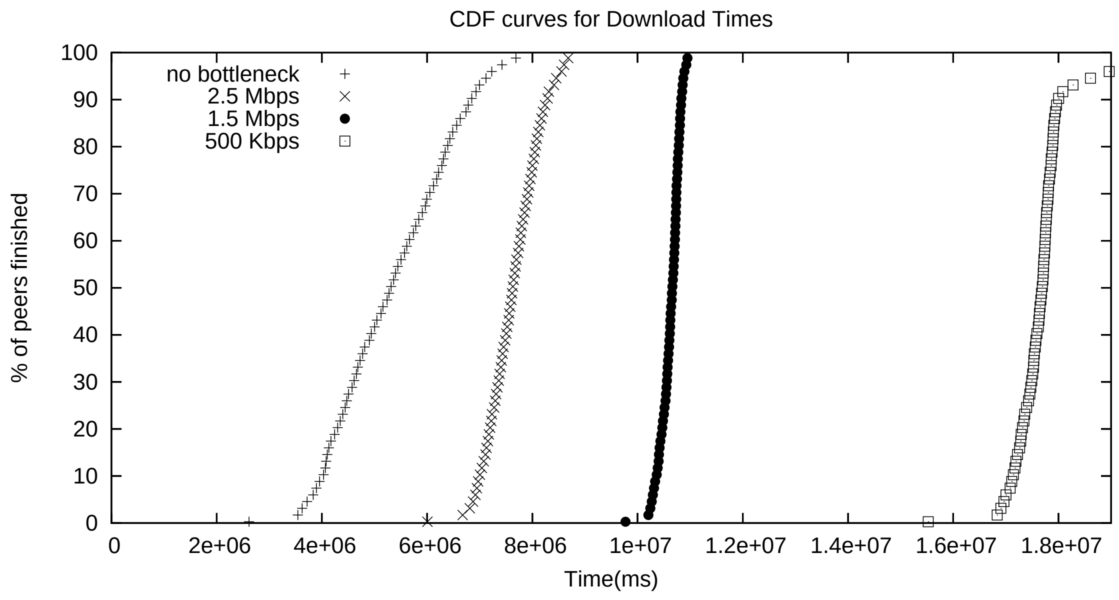

Fig. 3. CDF of download times under bandwidth throttling, in homogeneous networks.

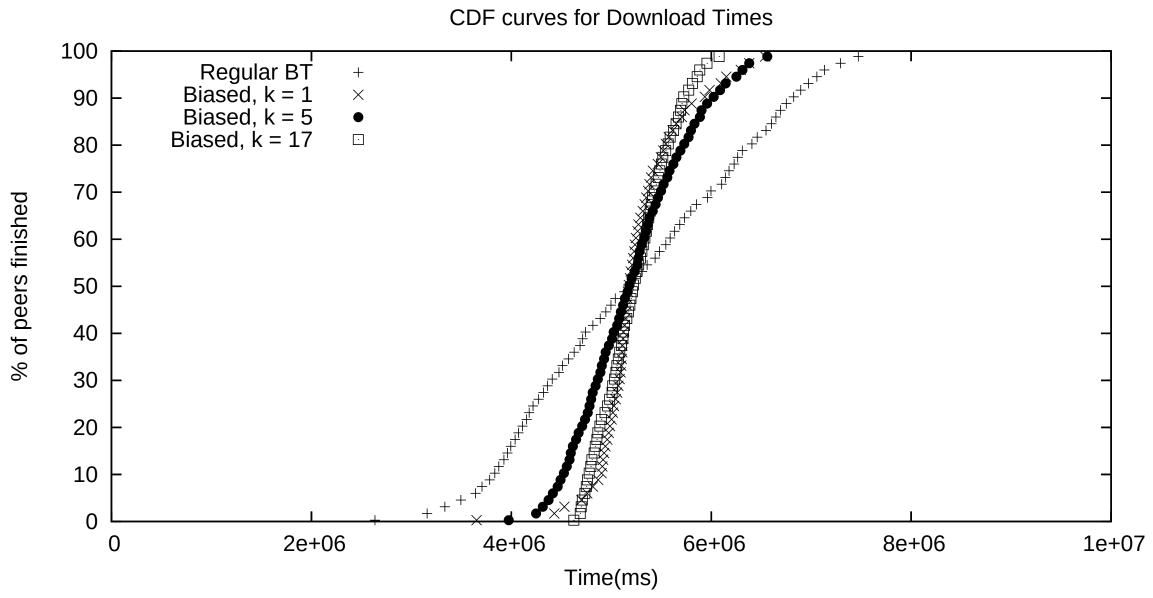

Fig. 4. CDF of download times under biased BT in homogeneous networks.

ISP using biased neighbor selection is high. If the ISP allows only k connections per node from other ISPs, then the traffic redundancy can be reduced.

Overall, biased BT is a practical and effective solution at reducing cross-ISP traffic caused by BitTorrent. Any ISP using it realizes an immediate benefit.

d) Interaction with Piece Selection Algorithm: The performance of biased neighbor selection, however, depends on the rarest first replication algorithm. Table III shows the 50th percentile, 95th percentile and traffic redundancy of least-replicated first versus random replication for biased neighbor selection of $k = 1$. Clearly, the reason that biased neighbor selection can perform nearly optimal is due to the “de-clustering” effect of least replicated first algorithm, which makes the peers more interested in blocks that are rare inside the ISP.

Piece Selection	50th percentile	95th percentile	traffic redundancy
Rarest first	1.84	2.51	14.4
Random	1.0	1.20	3.04

TABLE III
EFFECT OF PIECE SELECTION ALGORITHMS ON BIASED BT. THE DOWNLOAD TIME OF 1.0 MEANS 5,168 SECONDS.

e) Effect of the Original Seed: The single most important factoring affecting the download time of the BitTorrent network is the bandwidth of the original seed. We found that if the

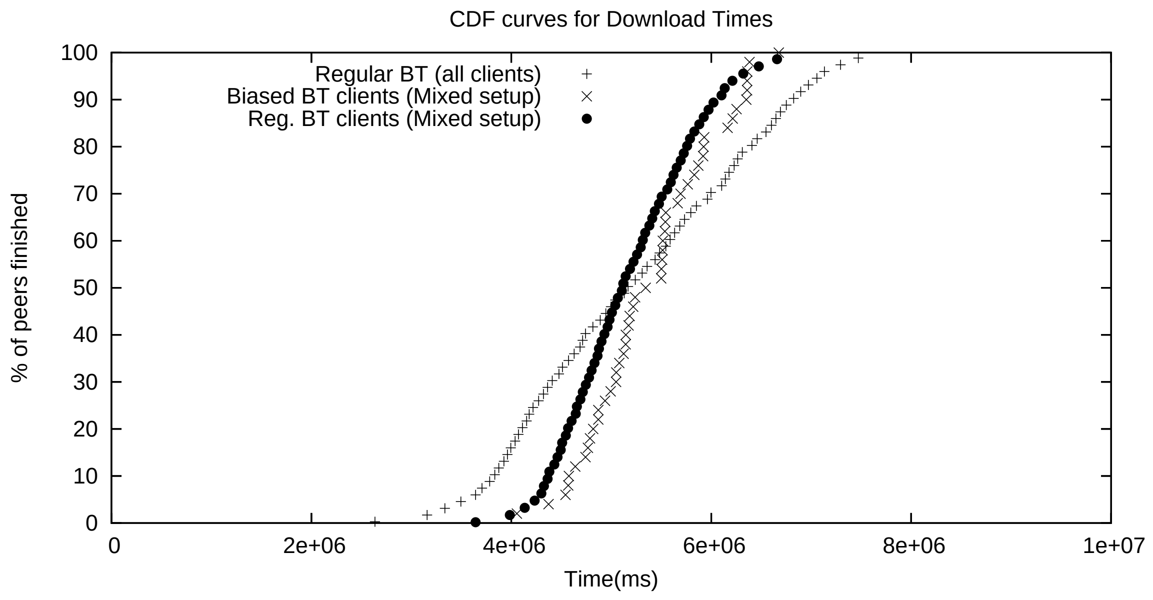

Fig. 5. CDF of download times when only one ISP uses biased neighbor selection with $k = 1$.

upload bandwidth of the original seed is 100Kbps or 200Kbps (i.e. 1x or 2x of cable modem peers' upload bandwidth), the average upload link utilization is around 50%. If the seed bandwidth is 400Kbps (4x cable modem peer's upload bandwidth) or higher, the average upload link utilization approach 100%. We speculate that the factor of 4 may have to do with the fact that BitTorrent use 5 parallel uploads each time. Other studies have reached similar conclusions [BHP05].

When the seed bandwidth is low, the effect of biased neighbor selection becomes unpredictable. Our experiments show that when the seed bandwidth is 100Kbps, biased neighbor selection at $k = 1$ reduces download times slightly (by about 5%), but when the seed bandwidth is 200Kbps, biased neighbor selection at $k = 1$ increases download time by as much as 40$. However, when the seed bandwidth is 400Kbps or higher, biased neighbor selection no longer changes the average download time.

For content providers who are eager to use BitTorrent to distribute their contents, establishing seeds with bandwidth higher than 4 times the average node upload bandwidth should not be a difficulty. Thus, we believe that the low seed bandwidth cases are rare in practice.

Finally, in all our experiments, the original seed does not use biased neighbor selection. Since the goal of the original seed is to distribute contents to as many nodes as possible, it should not use biased neighbor selection.

C. *Performance in Heterogeneous Networks*

What are the impacts of high-bandwidth peers? We added 7, 15 and 31 "university nodes" with upload/download bandwidth of 400Kbps to the above homogeneous network, and compare the performance of regular BitTorrent and biased BitTorrent. All university nodes leave the network as soon as they finish downloading the whole file. Furthermore, they do not use biased neighbor selection since they don't belong to any ISP.

ISP bottleneck	50th percentile	95th percentile	traffi c redundancy
no bottleneck	1.0	1.27	8.21
2.5Mbps	1.10	1.33	6.74
1.5Mbps	1.09	1.32	7.37
500kbps	1.12	1.34	4.40

TABLE V
COMBINATION OF BANDWIDTH THROTTLING AND BIASED BITTORRENT IN HETEROGENEOUS NETWORK. DOWNLOAD TIME OF 1.0 IS 4,446 SECONDS.

Table IV shows the results. Under regular BitTorrent, the addition of university nodes initially has no impact on overall download time, until the number of university nodes is high enough (over 4% of the total nodes in this case). Biased BitTorrent appears to take advantage of the presence of university nodes sooner, by reducing the 95th percentile download times. Furthermore, with 31 university nodes, biased BitTorrent outperforms regular BitTorrent slightly, mainly due to more uniform piece replications. Figure 6 shows the CDF of the download times under 31 university nodes.

However, the traffic redundancy of biased BitTorrent increases as the number of university nodes increases. This is understandable. The university nodes have high upload bandwidth and are favored by cable modem peers for block exchange. As a result, they also supply more blocks to cable modem peers, resulting in higher traffic redundancy.

Combining bandwidth throttle with biased BitTorrent in this case restores low redundancy and only increase download time slightly. Table V show the 50th and 95th percentile download times and traffic redundancy when each ISP throttles BitTorrent traffic and deploys biased neighbor selection. At 500Kbps bandwidth bottleneck, the traffic redundancy of biased BT is lowered to close to the value when external high bandwidth nodes is not present, and the download time is only increase

extra university nodes	Regular BitTorrent			Biased BT (k=1)		
	50th percentile	95th percentile	traffic redundancy	50th percentile	95th percentile	traffic redundancy
0	1.0	1.34	46.9	0.97	1.16	3.04
7	1.0	1.33	47.06	0.94	1.12	4.19
15	1.0	1.37	46.98	1.01	1.01	7.81
31	0.93	1.28	47.06	0.83	1.06	8.21

TABLE IV

NORMALIZED DOWNLOAD TIME AND TRAFFIC REDUNDANCY OF REGULAR VS. BIASED NEIGHBOR SELECTION AS THE NUMBER OF HIGH BANDWIDTH PEERS INCREASES. A DOWNLOAD TIME OF 1.0 IS 5,312 SECONDS.

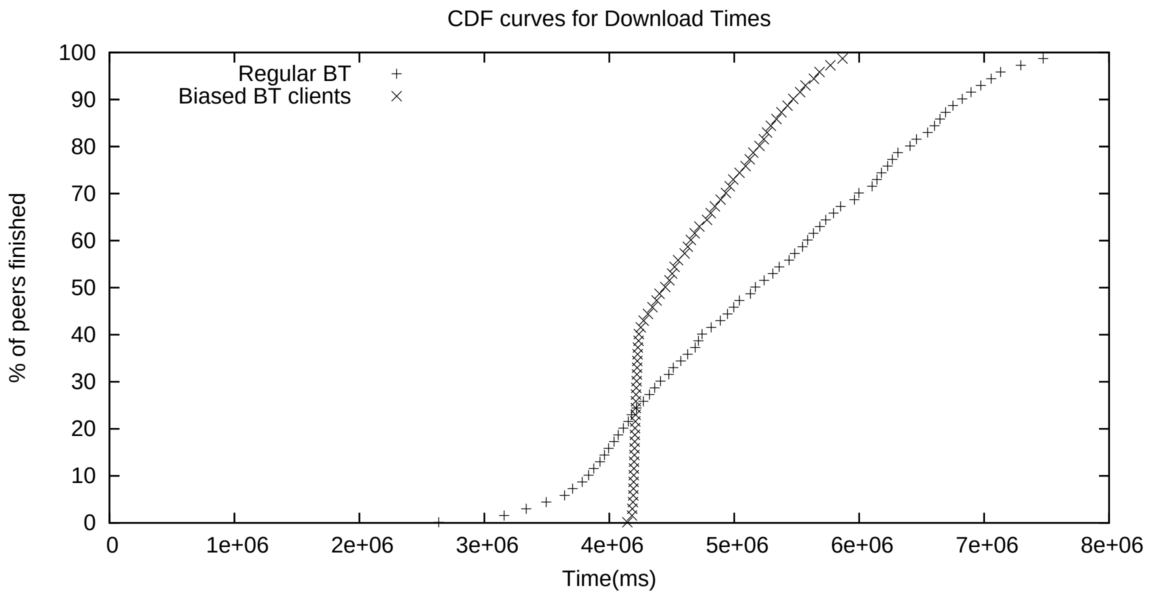

Fig. 6. CDF of download times of regular BT vs. biased BT, with 31 high bandwidth nodes in the network.

slightly ($< 12\%$). The bottleneck reduces the attraction of external high bandwidth nodes to internal peers, and thus reduces the cross-ISP traffic. Yet, since most neighbors of a peer are within the same ISP and are not crossing the bottleneck link, the bandwidth throttle increases the download time only slightly.

Overall, an ISP using both biased neighbor selection and bandwidth throttle can reduce its traffic from the external world significantly, regardless of whether the external peers are of high bandwidth. A proper value for the bandwidth throttle appears to be 5 or 6 times the average peer upload link bandwidth. The throttling should only be applied when biased neighbor selection is used; otherwise the peers' download times increase significantly.

Finally, another way to view our results is the following. If an ISP is deploying bandwidth throttling to BitTorrent traffic, then the peers inside the ISP can use biased neighbor selection to successfully avoid the bottleneck!

VI. COMPARISON WITH OTHER LOCALITY-ENHANCING APPROACHES

The above discussions show that biased neighbor selection performs much better than bandwidth throttling, and the two techniques can be combined for best results. This section examines two other techniques to reduce cross-ISP traffic: using a single peer, called "gateway peer", to connect to the external world [KRP05], and using a cache to store blocks sent to the ISP [Cac].

Technique	50th percentile	95th percentile
Regular BT	1.0	1.34
100Kbps gateway peers	2.59	2.81
400Kbps gateway peers	1.0	1.14

TABLE VI

NORMALIZED DOWNLOAD OF THE GATEWAY PEER APPROACH, COMPARED TO REGULAR BITTORRENT, IN HOMOGENEOUS NETWORK.

A. Biased Neighbor Selection vs. Gateway Peer

Gateway peer is a natural approach to eliminate redundant BitTorrent traffic into an ISP. The ISP designates a node as the gateway peer. All peers inside the ISP can only connect to each other and to the gateway peer, but only the gateway peer can connect to the external world.

The problem with this approach is that gateway peers cannot be regular peers. Table VI shows that if gateway peers have the same upload bandwidth as other peers, then the download time is increased significantly. The gateway peers need to have four

Technique	Peak bandwidth	Average bandwidth
Cache under Regular BT	3.61Mbps	1.73 Mbps
Cache with Biased BT	1.32Mbps	153Kbps

TABLE VII

PEAK AND AVERAGE UPLOAD BANDWIDTH NEEDS OF CACHES,.

times higher upload bandwidth to avoid increasing download times.

Furthermore, if only one ISP uses the gateway peer approach with a high bandwidth node, while all other ISPs use regular BitTorrent, then the nodes in other ISPs finish faster than the nodes in the ISP using gateway peer, by as much as 20% in one experiment. The reason for this behavior is that the gateway peer has nothing to gain from the internal peers, and via the tit-for-tat mechanism, would rather exchange blocks with external peers. This means that the extra equipments put up by the ISP benefits peers in other ISPs, certainly an undesirable result for the ISP.

Finally, using special high bandwidth nodes as gateway peers does not scale when peers participate in multiple BitTorrent networks, a common case as pointed out in [GCX$^+$05]. The gateway peer would need to allocate 400Kbps for each BitTorrent network to avoid increasing download times. In contrast, biased neighbor selection requires no extra equipments, and does not have such scalability problems.

B. Biased Neighbor Selection and Caches

Another approach to eliminate traffic redundancy is to use caches. Positioned at the ISP's gateway to the Internet, a cache stores blocks sent by external peers to internal peers, and when an internal peer wants to fetch a block from an external node, the cache intervenes transparently [Cac] and sends a locally-stored copy to the internal peer.

Caches also need high upload bandwidth. In order not to increase download times, the cache needs to deliver the data at the same bandwidth as the external peer would. To estimate the peak and average upload bandwidth needs of caches, we sum up the bandwidth of flows crossing the ISP boundary that are "intervened" by the cache (i.e. the block is delivered from the cache). Table VII shows the result. Under regular BitTorrent, both the peak and average upload bandwidth of the cache is high.

However, caches can be combined with biased neighbor selection. Table VII shows that with biased neighbor selection, the peak and the average bandwidth needs of the cache are significantly reduced. Furthermore, if caches are combined with both biased neighbor selection and bandwidth throttling, then the peak bandwidth requirement is limited by the bandwidth throttle. Thus, even for ISPs that deploy caches, biased neighbor selection should be used to improve the performance.

VII. SUMMARY AND FUTURE WORK

Biased neighbor selection works well. It is nearly optimal. Bandwidth throttling should be combined with biased neighbor selection. A caching approach should also be combined with biased neighbor selection.

Another way to look at biased neighbor selection is that it is a way for BitTorrent to bypass bottlenecks on the Internet. Our results essentially state that as long as k neighbors are chosen randomly from the "undesirable" pool of peers, the network would continue to function well.

Future work: implementation; clients auto-discovery of internal peers; using bandwidth as a way to "congestion" control P2P traffic.

REFERENCES

[BHP05] Ashwin Bharambe, Cormac Herley, and Venkata N. Padmanabhan. Understanding and deconstructing bittorrent performance. In *Proceedings of 2005 SIGMETRICS*, 2005.

[Cac] CacheLogic. Cachelogic - advanced solutions for peer-to-peer networks.

[CDK$^+$03] M. CASTRO, P. DRUSCHEL, A. KERMARREC, A. NANDI, A. ROWSTRON, and A. SINGH. Splitstream: High-bandwidth content distribution in a cooperative environment. In *Proceedings of IPTPS'03*, 2003.

[Coh03] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[Coh05] Bram Cohen. Bittorrent documentation: Protocol, 2005.

[CRZ00] Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, pages 1–12, 2000.

[ECo05] EContentMag.com. Chasing the user: The revenue streams of 2006, 2005.

[Epe] Pouwelse Garbacki Epema. The bittorrent p2p fi le-sharing system: Measurements and analysis.

[Fra02] Paul Francis. Your own internet distribution (yoid), 2002.

[GCX$^+$05] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measuremsnts, analysis and modeling of bittorrent-like systems. In *Proceedings of the Internet Measurement Conference 2005*, 2005.

[GR05] Christos Gkantsidis and Pablo Rodriguez Rodriguez. Network coding for large scale content distribution. In *Proceedings of IEEE Infocom 2005*, 2005.

[GS05] Prasanna Ganesan and Mukund Seshadri. On cooperative content distribution and the price of barter. In *Proceedings of 2005 ICDCS*, 2005.

[Inc] Cisco Systems Incorporated. Network based application recognition (nbar).

[IUKB$^+$04] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Hamra, and L. Garces-Erice. Dissecting bittorrent: Five months in a torrent's lifetime, 2004.

[KRP05] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki. Should internet service providers fear peer-assisted content distribution. In *Proceedings of the Internet Measurement Conference 2005*, 2005.

[MV05] Laurent Massouli&#233; and Milan Vojnovi&#262;. Coupon replication systems. *SIGMETRICS Perform. Eval. Rev.*, 33(1):2–13, 2005.

[MW] Jochen Mundinger and Richard Weber. Effi cient fi le dissemination using peer-to-peer technology.

[Nor03] William B. Norton. The evolution of the u.s. internet peering system, 2003.

[Pac] Packeteer. Packeteer packetshaper.

[PC] P-Cube. P-cube: Ip service control.

[PWCS02] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking, 2002.

[QS04] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks, 2004.

[San] Sandvine. Sandvine: Intelligent broadband network management.

[SBB04] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol, 2004.

[wik05] wiki.theory.org. Bittorrent protocol specifi cation v1.0, 2005.

[YdV04] X. Yang and G. de Veciana. Service capacity of peer to peer networks, 2004.