

Analyzing BitTorrent and Related Peer-to-Peer Networks

David Arthur*, Rina Panigrahy†
{darthur, rinap}@cs.stanford.edu

Abstract

We analyze protocols for disseminating a collection of data blocks over a network of peers with a view towards BitTorrent and related peer-to-peer networks. Unlike previous work, we accurately model the distribution of the individual data blocks, a process which is critical to the parallelism that makes BitTorrent successful in practice. We also consider multiple network topologies and routing algorithms.

We first demonstrate several routing algorithms that distribute b data blocks on a network with diameter d and maximum degree D in $O(D(b+d))$ phases of concurrent downloads with high probability. This is tight within a factor of D . We also specialize to the networks used by BitTorrent and we improve this bound to $O(b \ln n)$ phases where n is the number of clients. Finally, we discuss several practical extensions to BitTorrent, one of which improves the bound to a near-optimal $O(b + (\ln n)^2)$ phases.

1 Introduction

Over the last few years, BitTorrent [3] has emerged as a popular method for sharing data. Indeed, it accounts for up to thirty-five percent of all Internet traffic [4]. Each file is distributed on its own network as a number of independent data blocks. A client can share individual data blocks it has fully downloaded even if it has not finished downloading the entire file. This allows for a parallelism that is impossible if entire files are treated as atomic blocks.

The parallelism is most severely tested under “flash crowd” settings when a large number of clients join a network almost simultaneously. This does happen in practice and it creates a large demand on downloads without immediately increasing the system’s upload capacity. Empirical evidence suggests the relatively simple routing policy used by BitTorrent is quite effective even in these situations [2].

In this paper, we investigate this phenomenon from a theoretical perspective. We model BitTorrent and related networks as a graph of clients. In one time step,

each client can upload one data block to a neighboring client and each client can download one data block from a neighboring client. Unlike previous work (see Section 1.1), we simultaneously consider each block’s progress through the network, and we can thus accurately model when a client can download a new data block from its neighbors. This allows for a more realistic model than what has been used before.

We first consider a few natural routing policies based on BitTorrent’s protocols. We show that all of these algorithms share b data blocks on a network with diameter d and maximum degree D in $O(D(b+d))$ time steps, which is optimal within a factor of D . This analysis also leads to a class of somewhat practical routing algorithms that use an optimal $O(b + \ln n)$ time steps on random regular graphs with n vertices.

We also specialize our analysis to the graphs used by BitTorrent in practice. On such graphs, we show a natural randomized routing policy will share b data blocks among n clients in $O(b \ln n)$ time steps with high probability. Finally, we discuss several extensions to BitTorrent, one of which leads to a practical routing algorithm that uses a near-optimal $O(b + (\ln n)^2)$ time steps.

In Section 2, we describe our model. In Section 3, we show the $O(D(b+d))$ bound for a few deterministic routing algorithms. In Section 4, we extend this analysis to include the natural randomized routing algorithm. In Section 5, we prove the $O(b \ln n)$ bound for all BitTorrent-like graphs and we consider an extension that leads to the $O(b + (\ln n)^2)$ bound. Finally, we discuss several open problems in Section 6.

1.1 Related Work There have been several empirical studies of BitTorrent that demonstrate its routing policies work well in practice. Izal et al. [9] and Pouwelse et al. [13] measure the performance of existing BitTorrent systems and show them to be efficient. Bharambe et al. [2] and Gkantsidis and Rodriguez [8] build simulations to measure BitTorrent’s efficiency in a controlled setting. They are able to isolate some suboptimal behavior and they suggest several ways of avoiding it.

There has been some theoretical work as well. Qiu and Srikant [14] apply flow analysis to BitTorrent-like

*Supported in part by an NDSEG Fellowship, NSF grants EIA-0137761 and ITR-0331640, and an SNRC grant.

†Supported in part by an SGF Fellowship, NSF grants EIA-0137761 and ITR-0331640, and an SNRC grant.

networks and prove some bounds on time efficiency. However, they assume the data blocks each client has available for download at each time step is random and independent, which is certainly not true in practice. They also focus on the asymptotic behavior of a network with constant arrival and departure rates, which does not account for BitTorrent’s success under flash crowd settings.

Yang and de Veciana [15] consider flash crowds under two different models. The first one allows for strong results but, like the model in [14], it assumes an idealized network topology and an idealized upload policy, neither of which extend well to the unstable graphs that arise in practice. The second model, while more realistic in some ways, assumes that the distribution of one data block will not slow down the distribution of other data blocks. This ignores a complex interaction that is critical to understanding the routing.

We are unaware of any papers that model BitTorrent more closely than this, but there has also been some related work in the context of gossip and epidemic algorithms [1]. For example, Demers et al. [7] consider maintaining a replicated database by having databases repeatedly share all their data with another database chosen at random. Although the idea here is similar to that used by BitTorrent, the models are different in a few key ways. Demers et al. do not consider any network topology and they allow multiple updates to be simultaneously shared between two databases. We are interested in the interference caused when multiple blocks spread simultaneously, something which previous work in epidemic algorithms seems not to have covered.

2 Modeling BitTorrent

We begin by reviewing BitTorrent’s protocol and then presenting our model. At the end of the section, we prove a couple technical results about the networks that BitTorrent uses in practice.

2.1 BitTorrent Review We begin by briefly reviewing the essential features of BitTorrent’s protocol [5]. In general, BitTorrent distributes large files, called torrents, each on a separate network. A torrent is broken up into a large number of smaller data blocks, each usually between 32K and 256K, and these are then shared independently. A client that has not downloaded the entire file may still have completely downloaded several blocks, which it can upload to other clients. This allows clients to share the workload even as they are still downloading.

For each torrent, there is a corresponding central component called a tracker. When a client wishes to

download the torrent, it contacts the tracker. The tracker then returns a random subset of fifty existing clients, which we call the client’s neighbors. If a client ever has fewer than twenty neighbors due to disconnections, it may request a new set of neighbors from the tracker.

At any given time, a client attempts to upload to five of its neighbors. One of these neighbors is chosen at random, while the other four are chosen using a tit-for-tat system. Specifically, a client will attempt to upload to neighbors from which it is downloading fastest. When a client has a choice about what to download from a neighbor, it usually chooses the block that is least replicated among its other neighbors.

2.2 Our Model We are interested primarily in the routing itself, and thus we model only those aspects of BitTorrent that are directly relevant. In particular, we ignore the tit-for-tat scheme and we assume all clients have equal bandwidth.

We model the problem as that of routing data blocks on a directed graph over discrete time steps. Each vertex will begin with copies of certain data blocks. Then, during each time step, each vertex will choose one of its outgoing edges and then offer to upload along that edge. Each vertex can then accept up to one such offer and thus download up to one block each time step.

We assume that for each block B and each vertex v , there is a path from a vertex containing B to v . This guarantees that every vertex can eventually download every data block. We ask how many time steps this will take.

We do not formalize the process of clients joining and leaving the network during the routing, but with each major result, we include a note on the stability assumptions it requires. Also, surprisingly, the policy that vertices use to choose which block to download is largely inconsequential to our analysis. We will always assume that if a vertex is offered at least one block, it will download something. Beyond that, we will never make any assumptions on the download policy. Thus, we need not be concerned about BitTorrent’s policy of downloading the blocks that are least replicated among a client’s neighbors and the exceptions that go with this rule.

The upload policy, the underlying graph structure and the initial starting distribution of the data blocks all prove more important, and we will consider varying them in a number of ways. However, we will focus on oblivious upload policies that do not require a centralized component to coordinate actions.

Finally, we introduce some notation that will be used throughout the paper.

- Let G denote the directed graph upon which we are routing, and let n denote the number of vertices in G .
- Let b denote the number of distinct data blocks.
- Let b_v denote the number of blocks of which vertex v has a copy. Also, let $b_{v,t}$ denote the value of b_v after t time steps.
- Let the “distance” between a block B and a vertex v denote the length of the shortest path from a vertex containing B to v . Also, let d denote the maximum such distance over all pairs (B, v) .
- Let D denote the largest out-degree of any vertex in G .
- Let T denote the number of time steps before the routing completes.

As stated above, we assume d exists. Also, note that d is bounded from above by the diameter of the graph.

2.3 The BitTorrent Graph The networks that BitTorrent constructs are, not surprisingly, particularly effective for routing. In this section, we note several properties of these graphs, although we leave the proof to the Appendix.

We call a graph a BitTorrent- C graph ($C \geq 2$) if it is constructed by the following process.

1. Begin with C vertices, v_1, v_2, \dots, v_C , with an edge from v_j to v_i if and only if $j < i$.
2. While the total number of vertices is less than n , add a vertex and add directed edges from C existing vertices chosen at random to the new vertex.

Note that BitTorrent uses BitTorrent-50 graphs.

For any i , consider the set of integers j such that there is an edge from v_j to v_i . We let $m(i)$ denote a median element in this set. Also, recursively define $m^k(i) = m(m^{k-1}(i))$ for $k > 0$ and $m^0(i) = i$. We define the “median depth” of v_i to be the smallest k such that $m^k(i) = 1$. Clearly, the distance from v_1 to v_i is at most the median depth of v_i . Until Section 5, this is the only property of median depth we will use.

LEMMA 2.1. *With probability $1 - \frac{2}{n}$, the median depth of every vertex in a BitTorrent- C graph is at most $3 \lg n$, and the maximum out-degree in the graph is at most $3C(1 + \ln n)$.*

Proof. Consider a vertex v_i and let $X_j = \frac{m^j(i)}{m^{j-1}(i)}$. Note that, by symmetry, $E[X_j] = \frac{1}{2}$ and hence by independence, $E\left[\prod_{j=1}^{3 \lg n} X_j\right] = \frac{1}{n^3}$. The median depth of v_i is at most k if and only if $\prod_{j=1}^k X_j \leq \frac{1}{i}$. Thus,

it follows from Markov’s inequality that v_i has median depth at most $3 \lg n$ with probability at least $1 - \frac{1}{n^2}$. Therefore, with probability $1 - \frac{1}{n}$, every vertex has median depth at most $3 \lg n$.

Also, the probability that there is an edge from v_j to v_i is at most $\frac{C}{i-1}$ for $i > C$. Thus, for each j , the out-degree of v_j is stochastically dominated by C plus the sum of independent random variables $Y_{C+1}, Y_{C+2}, \dots, Y_n$ where Y_i is 1 with probability $\frac{C}{i-1}$ and 0 otherwise. Since $C \geq 2$, this sum has expectation at most $C + C\left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1}\right) < C(1 + \ln n)$. The Chernoff bounds [11] now imply the out-degree for v_j is at most $3C(1 + \ln n)$ with probability $1 - \left(\frac{e^2}{3^3}\right)^{C(1 + \ln n)} > 1 - \frac{1}{n^2}$ since $C \geq 2$. The result now follows as above.

3 Delay- x Upload Policies

In this section, we consider a large class of upload policies with particularly nice properties. We then demonstrate a couple simple routing algorithms that run in $O(D(b + d))$ time steps on any graph. We also use these to construct near-optimal routing algorithms for random regular graphs.

DEFINITION 3.1. *Suppose a vertex u can upload a block to a vertex v at every time step over a span of x consecutive time steps. An upload policy is called “delay- x ” if it guarantees that u will offer an upload to v at least once during any such span.*

For example, consider the upload policy where each vertex repeatedly cycles through its out-edges in the same order and offers to upload along each one in turn. This is clearly delay- D . Although it is not strictly speaking part of our model, we may also consider the case where a vertex uploads to all of its neighbors simultaneously at a fraction of the speed. This is also delay- D . Finally, consider the upload policy where each vertex attempts to upload to a random neighbor. One can check that this is delay- $D(\ln b + 3 \ln n)$ with probability at least $1 - \frac{1}{n}$.

3.1 Sparse Graphs In this section, we show that if x is small, delay- x upload policies perform well on any graph with any initial block distribution. We have only demonstrated the existence of such upload policies on sparse graphs, but as shown in Section 2.3, this includes the graphs that BitTorrent uses in practice. In the next section, we show how to extend these results to some other classes of graphs.

LEMMA 3.1. *Suppose that $b_{v,0} \geq k$ for all v and that we route using a delay- x upload policy. Let S denote the set of blocks within distance $\delta \geq 1$ of some vertex v ,*

and let T_v denote the first time step after which v has downloaded every block in S . Then, for each $t \leq T_v$, we have $b_{v,t} \geq \lfloor \frac{t}{x} \rfloor + k + 1 - \delta$.

Proof. We define an “epoch” to be x time steps. Note that if, at the beginning of an epoch, there is an edge from u to v and if u has more blocks than v has, then v must download some block during the epoch. This follows from the fact that u must offer to upload to v at some time step during the epoch, and then v must download some block during that time step.

Using this, we prove the claim by induction on t . If $t < x$, the result is trivial.

Now, consider $t_0 \geq x$. Suppose the result holds for $t < t_0$, and we will show it holds for $(x, t) = (x_0, t_0)$. Towards that end, let N denote the set of vertices that have an edge to v . For $u \in N$, let T'_u be the first time step after which u has downloaded all the blocks beginning within a distance $\delta - 1$ of u . Finally, let $T'_N = \max\{T'_u | u \in N\}$.

If $t - x \geq T'_N$, then all of the blocks in S had a copy in N throughout the last x time steps. It follows that if $t \leq T_v$, then v must have downloaded a block during the last epoch. Thus, $b_{v,t} \geq b_{v,t-x} + 1 \geq \lfloor \frac{t}{x} \rfloor + k + 1 - \delta$ by our inductive hypothesis. This completes the proof of the inductive step in this case.

On the other hand, if $t - x < T'_N$, then there exists some u with $T'_u > t - x$. It then follows from our inductive hypothesis that $b_{u,t-x} \geq \lfloor \frac{t}{x} \rfloor + k + 1 - \delta$ and $b_{v,t-x} \geq \lfloor \frac{t}{x} \rfloor + k - \delta$. If equality holds in the latter case, then once again, it follows that v must have downloaded a block in the last epoch. Thus, $b_{v,t} \geq \lfloor \frac{t}{x} \rfloor + k + 1 - \delta$, and the inductive step follows.

THEOREM 3.1. *Let $k = \min b_{v,0}$ over all vertices v . If we route using a delay- x upload policy, then $\max(b - k, d) \leq T \leq x(b - k + d - 1)$.*

Proof. The lower bound on T follows immediately from the fact that b_v can increase by at most one each time step while the distance between a block and a vertex can decrease by at most one each time step. On the other hand, we know from Lemma 3.1 that $b_{v,x(b-k+d-1)} \geq \lfloor \frac{x(b-k+d-1)}{x} \rfloor + k + 1 - d = b$. The result follows.

In practice, we would expect $d = O(\ln n)$ and $b = \omega(\ln n)$ so the bound in Theorem 3.1 simplifies to approximately xb . Thus, we have constructed oblivious upload policies that are guaranteed to take no more than approximately Db time steps. Conversely, for any D , there exist graphs where any routing will require at least Db time steps. For example, consider the star graph with vertices v_1, v_2, \dots, v_{D+1} and with edges from v_i to

v_j if and only if $i = 1$ and $j > 1$. If v_1 begins with a copy of every block and none of the other vertices begin with any blocks, it is easy to check that a routing can only complete after v_1 participates in Db different uploads.

Also, while Theorem 3.1 can easily be shown in the special case where one vertex begins with every data block, this assumption is not required. Thus, in BitTorrent, even if the initial server were disconnected, the file sharing could continue efficiently as long as all the data existed somewhere in the network. More generally, it is easy to see the proof of Theorem 3.1 is fairly stable under network mutations. Specifically, as long as there exist paths of length at most d from each data block to each vertex that remain connected throughout the routing, the result continues to hold even as other vertices are removed.

Finally, we apply Theorem 3.1 to BitTorrent- C graphs.

COROLLARY 3.1. *If clients alternate uploads among their neighbors in a predetermined order on a BitTorrent- C graph, then $T \leq 3C(1 + \ln n)(b + 3 \lg n - 1)$ with probability $1 - \frac{2}{n}$.*

As commented on above, we would usually expect b to be much larger than $\lg n$, so this reduces to approximately $3Cb \ln n$.

3.2 Denser Graphs Clearly, Theorem 3.1 is only tight for relatively sparse graphs. Since networks are sparse in practice, we do not focus on finding a more general result. However, it is worth noting the theorem can still be applied indirectly to denser graphs. We give one example here.

PROPOSITION 3.1. *Suppose G is chosen randomly from the set of D -regular graphs ($D \geq 3$) on n vertices. Consider the upload policy where each vertex chooses 3 neighbors at random, and then cycles upload requests through just these neighbors. Routing with this upload policy will complete in $3b + O(\ln n)$ time steps with high probability, regardless of the initial block distribution.*

Proof. By having each vertex restrict its uploads to just 3 neighbors, we have essentially restricted G to a random graph with all out-degrees equal to 3. By [6], such a graph is connected and has diameter $O(\ln n)$ with high probability. The result now follows immediately from Theorem 3.1.

This bound is, in fact, essentially optimal for routing on any graph. If every block begins at one vertex, it will take $\lg n$ time steps before every vertex has downloaded at least one block. Similarly, it will take b time steps before a vertex beginning with 0 blocks can

download every block. Thus, routing will always take $\Omega(b + \lg n)$ time steps on any graph. Although it is not obvious, one can check that the standard randomized upload policy discussed earlier does not achieve this bound even on a complete graph.

4 A Randomized Upload Policy

In this section, we consider the upload policy where each vertex attempts to upload to a random neighbor at each time step. As discussed at the beginning of Section 3, this is delay- $D(\ln b + 3 \ln n)$ with high probability. Thus, Theorem 3.1 shows routing with this upload policy requires at most $D(\ln b + 3 \ln n)(b + d)$ time steps. Since the randomized upload policy is so natural, and since it is tied to what BitTorrent does in practice, we improve the bound to $O(D(b + d))$ in this section.

4.1 A Reduced Problem We will reduce the analysis of the randomized upload policy to the case of routing on a path. It is convenient, however, to analyze this simpler problem before presenting the reduction.

Towards that end, consider vertices, v_0, v_1, \dots, v_l . We distribute b indistinguishable coins among these vertices. Let $x_i = j$ if coin i is at vertex v_j , and without loss of generality, assume $x_i \leq x_{i+1}$ for all i . Now, we fix a probability p , and at each time step, we let x_i increase by 1 with probability p if $x_{i+1} > x_i$. We call this coin movement an output from v_i and an input to v_{i+1} . These random choices are made independently for each coin. Finally, let the random variable $T_{x_i, b, l, p}$ denote the number of time steps before every coin reaches v_l .

LEMMA 4.1. *Suppose $x'_i \geq x_i$ for all i . Then $T_{x_i, b, l, p}$ stochastically dominates $T_{x'_i, b, l, p}$.*

Proof. We prove this by induction on $k_{x_i} = \sum_{i=1}^b l - x_i$. When $k_{x_i} = 0$, the claim is trivial. Now suppose it holds for $k_{x_i} < k$ and consider a configuration with $k_{x_i} = k$.

Let X denote the configuration specified by x_i and let X' denote the configuration specified by x'_i . In both cases, we imagine choosing independently for each coin whether to try to increase x_i . Then, we actually increase x_i only if $x_{i+1} > x_i$. Clearly, this process is identical to the given one.

We consider the result of a particular set of choices simultaneously for X and X' . Fix some i . If $x'_i > x_i$ initially, then after this time step, we will still have $x'_i \geq x_i$. Conversely, suppose we start with $x'_i = x_i$. If we choose not to try to increase x_i , then clearly we will still have $x'_i = x_i$ at the end of the time step. Otherwise, since $x'_{i+1} \geq x_{i+1}$, we will increase x_i only if we increase x'_i . Thus, in any case, we will still have $x'_i \geq x_i$ at the end of the time step.

We may ignore the case where neither x_i or x'_i change for any i as that simply brings us back where we started. In the remaining cases, it follows from the inductive hypothesis that the time remaining for X after one time step stochastically dominates the time remaining for X' after one time step. The inductive step follows from combining these results for each set of random choices.

This states the intuitive result that moving a coin forward can only decrease the amount of time remaining, which we prove in the Appendix. Thus, suppose that instead of starting all the coins at v_0 , we start v_0 with 0 coins and then feed a coin to v_0 with probability $q < p$ independently at each time step. It follows from Lemma 4.1 that this only increases the amount of time before v_l obtains b coins. Now, the number of coins at v_0 can be modeled as a Markov chain that reaches a steady state since $q < p$. Furthermore, since the graph of states for this Markov chain is a tree, the process is reversible.

LEMMA 4.2. *In the steady state distribution, the probability that a coin is output from v_0 is q . Also, if v_0 is in the steady state distribution, then v_0 will still be in the steady state distribution in the next time step even conditioned on either possible output.*

Proof. This result is a discretized version of Burke's theorem [10]. The standard proof of Burke's theorem, based on reversibility, applies here as well.

More generally, we can consider the Markov chain with states giving the number of coins at each of v_0, v_1, \dots, v_{l-1} .

LEMMA 4.3. *Let b_i denote the number of coins at v_i . Then, the steady state of the given Markov chain occurs when each b_i is independently distributed as follows.*

$$b_i = \begin{cases} 0 & \text{with probability } \frac{p-q}{p}, \\ j \ (j > 0) & \text{with probability } \frac{(p-q)q^j(1-p)^{j-1}}{p^{j+1}(1-q)^j}. \end{cases}$$

Proof. We first consider the case where $l = 1$. Let $\pi_{i,j}$ denote the probability of transitioning from $b_0 = i$ to $b_0 = j$ in one time step, and let p_j denote the probability that $b_0 = j$ in the steady state. Then, by reversibility, $p_{j+1} = p_j \frac{\pi_{j,j+1}}{\pi_{j+1,j}}$. Now, $\pi_{0,1} = q$, $\pi_{j,j+1} = q(1-p)$ for $j > 0$, and $\pi_{j+1,j} = p(1-q)$ for all j . Combining this with the fact that $\sum_{j=0}^{\infty} p_j = 1$, we find that the steady state for b_0 is as claimed.

For the general case, suppose the b_i are distributed as described above. Then, after one time step, each b_i individually will still be distributed as above since

they each receive an input coin with probability q . Furthermore, since b_i does not depend on previous output from v_i , the b_i are still independent. The result follows.

We now have the tools to analyze $T_{x_i,b,l,p}$.

PROPOSITION 4.1. *If $l' \geq l$, then $T_{x_i,b,l,p} \leq \frac{16l'+4b}{p}$ with probability at least $1 - 2 \exp\left(-\frac{l'}{2}\right)$.*

Proof. Clearly, $T_{x_i,b,l',p}$ stochastically dominates $T_{x_i,b,l,p}$ so it suffices to prove the claim when $l' = l$.

Again, we assume the coins are fed to v_0 with probability q at each time step and we ask how long it takes for b of these coins to reach v_l . We now assume $q = \frac{p}{2-p}$, which, as required, is less than p as long as $p < 1$. Furthermore, we add a random number of dummy coins at each v_i so that the Markov chain described in Lemma 4.3 is already in the steady state. Lemma 4.1 implies that the time for b coins to go from v_0 to v_l in this scenario stochastically dominates $T_{x_i,b,l,p}$.

Let A denote the total number of coins on all of the vertices of a random instance of the Markov chain in its steady state. Then, $A = \sum_{i=0}^{l-1} b_i$ where b_i is specified as in Lemma 4.3. Note that for $j \geq 1$, $P[b_i \geq j] = \left(\frac{q(1-p)}{p(1-q)}\right)^{j-1} \cdot \frac{(p-q)q}{p^2(1-q)} \cdot \frac{1}{1-\frac{q(1-p)}{p(1-q)}}$, which simplifies to $\frac{q}{p \cdot 2^{j-1}} < \frac{1}{2^{j-1}}$. It follows that b_i is stochastically dominated by a geometric distribution with mean 2. Therefore, by the Chernoff bounds for the negative binomial distribution [12], A is at most $4l$ with probability $1 - \left(\frac{16}{27}\right)^l > 1 - \exp\left(-\frac{l}{2}\right)$.

Now, the number of coins that v_l receives in k time steps is precisely the sum of k independent random variables that are 1 with probability q and that are 0 otherwise. Thus, if $k = \frac{8l+2b}{q}$, the standard Chernoff bounds imply v_l receives $4l + b$ coins with probability at least $1 - \exp\left(-\frac{4l+b}{4}\right) > 1 - \exp\left(-\frac{l}{2}\right)$.

With probability $1 - 2 \exp\left(-\frac{l}{2}\right)$, it follows that after $\frac{8l+2b}{q}$ time steps, v_l will have received $4l + b$ coins, at most $4l$ of which did not start at v_0 . Thus, with probability $1 - 2 \exp\left(-\frac{l}{2}\right)$, we have $T_{x_i,l,b,p} \leq \frac{8l+2b}{q} < \frac{16l+4b}{p}$.

4.2 The Reduction We now return to the case of routing with a randomized upload policy on a general graph G . As before, we let D denote the maximum out-degree in G . It is straightforward to prove the following reduction from routing time on a network to $T_{x_i,b,l,p}$ using Lemma 4.1. We leave the details to the Appendix.

LEMMA 4.4. *Suppose that some vertex u in G begins with a copy of every block. Then, fix a vertex v and let*

$u = v_0, v_1, v_2, \dots, v_l = v$ be a path from u to v . Also, let b_j denote the number of blocks that v_j has a copy of, and let $x_i = \max\{j \mid b_k \geq b + 1 - i \ \forall k \leq j\}$. Finally, let T_v denote the number of time steps before v has downloaded a copy of every block.

Then, $T_{x_i,b,l,\frac{1}{D}}$ stochastically dominates T_v .

Proof. We prove this by induction on $k_{x_i} = \sum_{i=1}^b l - x_i$. When $k_{x_i} = 0$, the claim is trivial. Now suppose it holds for $k_{x_i} < k$ and consider a configuration with $k_{x_i} = k$. Let X denote this routing problem, and let X' denote the corresponding stochastic process described in the previous section.

Consider some i for which $x_{i+1} > x_i$. By definition of x_i , we know $b_{x_i} \geq b + 1 - i > b_{x_{i+1}}$. On the other hand, since $x_{i+1} > x_i$, we can further say that $b_{x_{i+1}} = b - i$. Now, v_{x_i} will request an upload to $v_{x_{i+1}}$ with probability at least $\frac{1}{D}$. If this happens, $b_{x_{i+1}}$ will increase to $b - i + 1$ and hence, x_i will increase by at least 1.

Note this depended only on where v_{x_i} requested an upload, which is independent of v_j 's upload request for $j \neq x_i$. Thus, for each i satisfying $x_{i+1} > x_i$, we may define indicator variables A_i with the following properties.

1. A_i is 1 with probability $\frac{1}{D}$ and 0 otherwise.
2. If $A_i = 1$, then v_{x_i} requested an upload to $v_{x_{i+1}}$ and x_i increased by at least 1.
3. A_i is independent of A_j for $j \neq i$.

There is now a natural correspondence between whether $A_i = 1$ in X and whether x_i increases in X' . Note x_i can never increase by more than 1 in X' and when that happens, $A_i = 1$, and x_i increases by at least 1 in X . For each set of values of A , it follows from our inductive hypothesis and from Lemma 4.1 that the remaining time for X' stochastically dominates the remaining time for X . The inductive steps follows from combining these results for each set of values for A_i .

Our main result for the randomized upload policy now follows immediately from Proposition 4.1 and Lemma 4.4.

THEOREM 4.1. *Suppose a vertex u begins with a copy of every block. Let D denote the maximum out-degree in G , and suppose the distance from u to every other vertex is at most d . If we route on this graph using the randomized upload policy, then $T \leq 4D(4d + b)$ with probability at least $1 - 2n \exp\left(-\frac{d}{2}\right)$.*

Since the proof of this theorem considers only routing along a single path, it holds as long as there exist

paths of appropriate length from the source vertex to every other vertex that remain throughout the routing, even as other vertices are deleted.

Finally, we apply Theorem 4.1 to BitTorrent- C graphs.

COROLLARY 4.1. *Let G be a BitTorrent- C graph where the initial vertex begins with a copy of every block. If we route on G using a randomized upload policy, then $T \leq 12C(1 + \ln n)(b + 12 \lg n)$ with probability at least $1 - \frac{4}{n}$.*

In most applications, we would again expect b to be much larger than $\lg n$, so this reduces to approximately $12Cb \ln n$. Also, a variant of Proposition 3.1 follows immediately for a randomized upload policy that restricts to just 3 neighbors.

5 A Tighter Analysis for BitTorrent-like Graphs

All of our previous results, when applied to BitTorrent- C graphs, are heavily dependent on C . In this section, we strengthen our results on BitTorrent- C graphs to eliminate this dependency. The tighter analysis also suggests a simple modification to BitTorrent that achieves near-optimal efficiency.

Suppose a graph has the following properties.

1. Every vertex has out-degree at most D .
2. The vertices can be partitioned into sets X_1, X_2, \dots, X_d such that every vertex in X_i has at least A in-edges originating in $X_1 \cup X_2 \cup \dots \cup X_{i-1}$.

We will call such a graph an “ A - D - d tree”.

Now, consider a BitTorrent- C graph G and recall the definition of median depth from Section 2.3. We can similarly define the “median C -depth” of v_i to be the number of median edges that must be followed from v_i before reaching one of the initial C vertices. We partition the vertices of G into sets X_i by letting $X_0 = \{v_1, v_2, \dots, v_C\}$ and for $i > 0$, letting X_i contain the vertices with median C -depth of i . It follows from Lemma 2.1 that G is an A - D - d tree with $A = \frac{C}{2}$, $D = 3C(1 + \ln n)$ and $d \leq 3 \lg n$ with probability $1 - \frac{2}{n}$.

We now show the number of time steps for routing on an A - D - d tree with certain initial block distributions depends on $\frac{D}{A}$, rather than on just D .

THEOREM 5.1. *Let G be an A - D - d tree and suppose all the vertices in X_0 begin with a copy of every block. With the standard random upload policy, routing will complete in at most $3.164 \frac{D}{A} \cdot (d(4 \ln b + 8 \ln n) + b)$ time steps with probability at least $1 - \frac{1}{n}$.*

Proof. Let K be a constant to be fixed later. We consider epochs where epoch k begins when every vertex in X_i has at least $K(k-i)$ blocks for each i . Note that epoch 1 begins at the first time step and the routing is complete at the end of epoch $d + \frac{b}{K}$.

Consider a vertex $v \in X_i$ during epoch k . Suppose v has less than $K(k-i+1)$ blocks. Then, every vertex in X_j for $j < i$ has more blocks than v . We know at least A of these vertices have edges to v , so it follows that v will download a block in the next $\frac{D}{A}$ time steps with probability at least $1 - (1 - \frac{1}{D})^{A \cdot \frac{D}{A}} \geq 1 - \frac{1}{e}$. Thus, as long as v has less than $K(k-i+1)$ blocks, the number of blocks that v downloads in $t \frac{D}{A}$ time steps stochastically dominates the sum of t independent random variables that are 1 with probability $1 - \frac{1}{e}$ and 0 otherwise.

Since v has at least $K(k-i)$ blocks at the beginning of epoch k , it follows from the Chernoff bounds that v has at least $K(k-i+1)$ blocks $\frac{2K \frac{D}{A}}{1 - \frac{1}{e}}$ time steps after the beginning of epoch i with probability at least $1 - \exp(-\frac{K}{4})$.

There are n vertices, and each one must increase its number of blocks by K at most $\frac{b}{K} \leq b$ times. Thus, with probability at least $1 - bn \exp(-\frac{K}{4})$, it holds for each i and k that each vertex in X_i will have at least $K(k-i+1)$ blocks within $\frac{2K \frac{D}{A}}{1 - \frac{1}{e}}$ time steps after the start of epoch k . Therefore, with probability $1 - bn \exp(-\frac{K}{4})$, each epoch will last at most $\frac{2K \frac{D}{A}}{1 - \frac{1}{e}}$ time steps, and the process completes in at most $\frac{2K \frac{D}{A}}{1 - \frac{1}{e}} \cdot (d + \frac{b}{K}) = \frac{2D}{1 - \frac{1}{e}} \cdot (dK + b)$ time steps. In particular, the desired result follows from taking $K = 4 \ln b + 8 \ln n$.

As vertices are added and deleted, this result will hold as long as the graph remains an A - D - d tree. On BitTorrent, a vertex’s neighbors are chosen randomly, so it is very unlikely that they would all disconnect very early. Furthermore, if a vertex ever has in-degree less than twenty, it can acquire new neighbors. Thus, with BitTorrent, the graph really does maintain its structure as an A - D - d tree even as clients leave the system.

Finally, we apply this result to BitTorrent- C graphs. Note that, in practice, the condition that multiple clients begin with all the packets is quite natural. The initial seed on a torrent is likely to have very good upload bandwidth so it can be approximated as multiple normal clients.

COROLLARY 5.1. *Consider a BitTorrent- C graph where the first C vertices begin with every data block. If we route using the standard randomized upload policy, the routing will complete in at most $19(1 + \ln n)(b + 12(\ln b)(\ln n) + 24(\ln n)^2)$ time steps with probability*

$1 - \frac{3}{n}$.

Proof. As noted above, a BitTorrent- C graph is an A - D - d tree with $A = \frac{C}{2}$, $D = 3C(1 + \ln n)$ and $d = 3 \lg n$ with probability $1 - \frac{2}{n}$. The result now follows from Theorem 5.1.

In most applications, we would again expect b to be larger than $(\lg n)^2$, so this reduces to approximately $19b \ln n$. In the next section, we show a related family of graphs for which this bound achieves a significantly tighter result.

5.1 Smoothed BitTorrent- C Graphs We consider a practical variant of BitTorrent- C graphs that performs even better under our analysis. The “practical” condition here is important. As shown in Section 3.2, there exist algorithms that perform very well on random regular graphs, but such algorithms restrict their routing to an extremely sparse subgraph that could easily be disrupted by clients joining and leaving the system. Similarly, we could modify a BitTorrent- C graph to always connect vertex i to the C vertices with index closest to $\frac{i}{2}$. This also performs well in theory, but it is again extremely sensitive to graph mutation.

We define a more practical variant here, which we call a “smoothed BitTorrent- C ” graph. As before, we begin with C vertices v_1, v_2, \dots, v_C with a directed edge from v_j to v_i if and only if $j < i$. Again, we add the remaining vertices in order and add directed edges to each from some C previous vertices. In this case, however, instead of choosing each previous vertex uniformly at random, we choose two previous vertices, and connect the new vertex to the one with higher index. Finally, we will be interested in the case where $C = C_0 \ln n$ for a constant $C_0 \geq 1$.

LEMMA 5.1. *If $C = C_0 \ln n$, every vertex in a smoothed BitTorrent- C graph has median depth $O(\log n)$ and out-degree $O(C)$ with probability at least $1 - \frac{2}{n}$.*

Proof. Consider a vertex v_i being added to the graph. We choose $2C$ vertices and then connect half of them to C . In the worst case, the median edge to v_i connects to the vertex with rank $1.5C$ among the $2C$ vertices chosen. Thus, an argument similar to that used for Lemma 2.1 shows that v_i has median depth $O(\log n)$ with probability $1 - \frac{1}{n^2}$.

We now consider the out-degree of each vertex v_j . Again, consider v_i being added to the graph, and a vertex being chosen to connect to it. Then, v_j is chosen with probability $\frac{2j-1}{(i-1)^2}$. Thus, the probability that there is an edge from v_j to v_i is at most $\frac{2C_0 j \ln n}{(i-1)^2}$. These probabilities are independent for distinct i . Thus, the

out-degree of v_j is stochastically dominated by the sum of independent random variables $X_{j+1}, X_{j+2}, \dots, X_n$ where X_i is 1 with probability $\frac{2C_0 j \ln n}{(i-1)^2}$ and 0 otherwise.

Now, note that $E \left[\sum_{i=j}^n X_i \right] = 2C_0 j \ln n \sum_{i=j+1}^n \frac{1}{(i-1)^2} \leq 4C_0 \ln n$ since $\sum_{i=j+1}^n \frac{1}{(i-1)^2} < \frac{1}{j^2} + \sum_{i=j+2}^n \left(\frac{1}{i-2} - \frac{1}{i-1} \right) \leq \frac{2}{j}$. It follows from the Chernoff bounds that $\sum_{i=j}^n X_i = O(C_0 \ln n)$ with probability $1 - \frac{1}{n^2}$.

COROLLARY 5.2. *Consider a smoothed BitTorrent- C graph where the first C vertices begin with every data block. If we route using the standard randomized upload policy, the routing will complete in at most $O(b + (\ln n)^2)$ time steps with probability $1 - \frac{3}{n}$.*

Proof. It follows from Lemma 5.1 that the graph is an A - D - d tree with $A = C$, $D = O(C)$ and $d = O(\ln n)$ with probability $1 - \frac{2}{n}$. The result now follows from Theorem 5.1.

As noted earlier, the optimal running time is $\Omega(b + \ln n)$, and in practice, we would expect b to be much larger than $\ln n$, so the bound from Corollary 5.2 is near-optimal.

6 Extensions and Further Work

6.1 Streaming To share streamed data, it must be guaranteed that data blocks will be downloaded in approximately the correct order. Currently, BitTorrent does not support streaming in any way. It uses a download policy where each client chooses a data block that is least replicated among its neighbors, which in no way guarantees that data will arrive in any approximation of the correct order.

On the other hand, all of our results hold for any download policy. This implies that if BitTorrent switched to the intuitively inferior policy of always downloading data blocks in order, it could still perform well. A few practical problems would arise, however. Most notably, the last block would become very rare if clients left the network as soon as they download it. Attempting to resolve these issues might make for some interesting future work.

6.2 Further Work A few other open problems remain. First of all, we do not model either varying client bandwidths or BitTorrent’s tit-for-tat policy. It would be interesting to model these and to then determine what effect, if any, they would have on the results.

Secondly, it would be nice to improve the general $O(D(b + d))$ bound. Although this is relatively tight in the cases we are interested in, it can be very weak

sometimes, most notably with the complete graph. One can show this bound is the tightest possible in terms of a graph's maximum degree, minimum degree and diameter but perhaps a more universal bound could be found in terms of other graph parameters.

References

- [1] N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner Press, Second Edition, 1975.
- [2] A. Bharambe, C. Herley, and V. Padmanabhan. Understanding and Deconstructing BitTorrent Performance. Technical Report MSR-TR-2005-03, Microsoft Research, Jan. 2005.
- [3] BitTorrent. <http://bittorrent.com>.
- [4] BitTorrent Accounts for 35% of Traffic. <http://yro.slashdot.org/yro/04/11/04/1749257.shtml>.
- [5] BitTorrent Protocol Specification v1.0. <http://wiki.theory.org/BitTorrentSpecification>.
- [6] B. Bollobás and F. De La Vega. The diameter of random regular graphs. *Combinatorica*, 2(2), pages 125-134, 1982.
- [7] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart and D. Terry. Epidemic algorithms for replicated database maintenance. *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1-12, 1987.
- [8] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. Technical Report MSR-TR-2004-80, Microsoft Research, 2004.
- [9] M. Izal, G. Urvoy-Keller, E.W. Biersack, P. Felber, A. Al Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. *Proceedings of Passive and Active Measurements*, pages 1-11, 2004.
- [10] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, Second Edition, 1993.
- [11] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [12] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice-Hall, 1994.
- [13] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, and H.J. Sips. A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System. Technical Report PDS-2004-003. Delft University of Technology, The Netherlands, Apr. 2004.
- [14] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. *Proceedings of SIGCOMM*, pages 367-378, 2004.
- [15] X. Yang and G. de Veciana. Service Capacity of Peer to Peer Networks. *Proceedings of INFOCOM*, 2004.