Analyzing the Efficiency of BitTorrent and Related Peer-to-Peer Networks

David Arthur*, Rina Panigraphy†

March 30, 2005

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

1 Introduction

Over the last few years, BitTorrent [3] has emerged as a popular method for sharing data. 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]. Clients rarely have to wait long before finding a neighbor they can download from and the data distribution progresses quickly.

In this paper, we investigate this phenomenon from a theoretical perspective. We model a BitTorrent-like network as a graph of clients where, at each time step, each client can upload up to one data block to a neighboring client and each client can download up to one data block from a neighboring client. Unlike previous work, we simultaneously model 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. Within the context of this model, we first show bounds that help explain BitTorrent's success and we then discuss modified routing policies that perform even better.

We first consider a few simple deterministic routing policies and a natural randomized policy based on BitTorrent's protocols. We show that all of these algorithms will share n data blocks on a network with diameter d and maximum degree D in $O(D(n+d))$ time steps. This is optimal

*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.

within a factor of D. Although they are not always practical, this also leads to a class of routing algorithms that use an optimal $O(n + \ln m)$ time steps on random regular graphs with m vertices.

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

In Section 1.1, we review related work on the subject. In Section 2, we describe our model. In Section 3, we show the $O(D(n+d))$ 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 $O(n + (\ln m)^2)$ bound for BitTorrent-like graphs. Finally, in Section 6, we consider extensions to BitTorrent.

1.1 Related Work

There have been several empirical studies of BitTorrent, which demonstrate that its routing algorithm works well in practice. Izal et al. [9] and Pouwelse et al. [11] measure the performance of existing BitTorrent systems and show them to be efficient. Bharambe et al. [2] and Gkantsidis and Rodriguez [7] build simulations to measure the protocol'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 [12] apply flow analysis to BitTorrent-like networks and prove some bounds on download times. 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, and in particular, they do not consider flash crowds.

Yang and de Veciana [13] consider flash crowds under two different models. The first one allows very strong results but, like [12], it ignores the network topology and the data distribution, and it assumes an idealized upload policy that does not generalize to unstable graphs. The second model, although more realistic in many ways, assumes that the distribution of one data block will not slow down the distribution of other data blocks. As with [12], this is not

2 Modeling BitTorrent

We begin by reviewing BitTorrent's protocol and then modeling it and similar protocols. At the end of the section, we also analyze the networks that BitTorrent uses in practice.

2.1 BitTorrent Review

We begin by reviewing BitTorrent's protocol [4] and by constructing a model that captures its essential features. In general, BitTorrent distributes large files, called torrents, each on a separate network. A torrent is broken up into a large number of smaller blocks, each between around 32K and 256K, and these are then shared independently. Thus, a client which has not downloaded the entire file may still have completely downloaded several blocks, and it can then upload these 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 50 existing clients, which we call the client's neighbors. If the number of neighbors of a client ever falls below 20, the client 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. There are a couple exceptions to this last rule, however, primarily among clients downloading their first block. This will not affect our analysis, however.

2.2 The 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 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. This might be further complicated by clients joining and leaving the network in the middle of the routing. We do not model this process directly, but for each major result, we discuss the stability assumptions it requires.

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 upload, 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 polciy, 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. We consider only oblivious upload polcies 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 m denote the number of vertices in G.
- Let n denote the number of distinct data blocks.
- Let n_v denote the number of blocks that vertex v has a copy of. Also, let $n_{t,v}$ denote the value of n_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.

2.3 The BitTorrent Graph

The networks that BitTorrent constructs are, not surprisingly, particulary effective for routing. In this section, we prove several properties of these graphs.

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

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

Note that BitTorrent uses BitTorrent-50 graphs.

For any i, let $m(i)$ denote the median j such that there is an edge from v_j to v_i. Also, recursively define $m^k(i) = m(m^{k-1}(i))$ for $k > 0$ and $m^0(i) = i$ for. 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.

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

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, $E\left[\Pi_{j=1}^{3\lg m} X_j\right] = \frac{1}{m^3}$. The median depth of v_i is at most k if and only if $\Pi_{j=1}^{k} X_j \leq \frac{1}{i}$. Thus, it follows from Markov's inequality that v_i has median depth at most $3\lg m$ with probability at least $1 - \frac{1}{m^2}$. Therefore, with probability $1 - \frac{1}{m}$, every vertex has median depth at most $3\lg m$.

Also, the probability that there is an edge from v_i to v_j is $\frac{C}{j}$. For each i, the out-degree of v_i is thus stochastically dominated by the sum of independent random variables $X_1, X_2, \ldots, X_m$ where X_j is 1 with probability $\frac{C}{j}$ and 0 otherwise. This sum has mean between $C\ln m$ and $C(\ln m + 1)$. Thus, by the Chernoff bounds [8], the out-degree for v_i is at most $3C(\ln m + 1)$ with probability $\left(\frac{e^2}{(3C)^3}\right)^{C(\ln m+1)} < \frac{1}{m^2}$ since $C \geq 2$. The result now follows as above. $\square$

The constants on the maximum out-degree can be tightened further, but in the interests of simplicity, we will use this bound.

3 Delay-x Upload Policies

In this section, we consider a large class of upload policies with particularly nice properties. Using this, we show a simple routing algorithm that always runs in $O(D(n+d))$ time steps on any graph. We also show how to use this to find near optimal routing algorithms for random regular graphs.

Definition 3.1. *Suppose there is an edge from a vertex u to a vertex v and u has a block that v does not. We say an upload policy is "delay-x" if it guarantees that u will offer to upload to v in at most x time steps.*

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-$D(\ln n + 3\ln m)$ with probability at least $1 - \frac{1}{m}$.

3.1 On 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 shown 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. Furthermore, in the next section, we show how to extend these results to some other classes of graphs.

Lemma 3.1. *Suppose that $n_{0,v} \geq k$ for all v and that we rout 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 $t \leq T_v$,*

$$n_{t,v} \geq \left\lfloor \frac{t}{x} \right\rfloor + k + 1 - \delta.$$

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 $\delta + t$. If $t < x$, the result is trivial. If $\delta = 1$, then as long as $t < T_v$, we know v must download a block during each epoch as noted above. Hence, $n_{t,v} \geq \left\lfloor \frac{t}{x} \right\rfloor + k$, as claimed.

Now, consider $x_0 \geq 2$ and $t_0 \geq x$. Suppose the result holds for $x + t < x_0 + t_0$, 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 during 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 $x \leq T_v$, then v must have downloaded a block during the last epoch. Thus, $n_{t,v} \geq n_{t-x,v} + 1 \geq \left\lfloor \frac{t}{x} \right\rfloor + k + 1 - \delta$ 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 $n_{t-x,u} \geq \left\lfloor \frac{t}{x} \right\rfloor + k + 1 - \delta$ and $n_{t-x,v} \geq \left\lfloor \frac{t}{x} \right\rfloor + k - \delta$. If equality holds in the latter case, then once again, it follows that v must have downloaded a block in the last epoch. Thus, $n_{t,v} \geq \left\lfloor \frac{t}{x} \right\rfloor + k + 1 - \delta$, and the inductive step follows. ∎

Theorem 3.2. *Let $k = \min n_{0,v}$ over all vertices v. If we rout using a delay-x upload policy, then*

$$\max(n-k, d) \leq T \leq x(n-k+d-1).$$

Proof. The lower bound on T follows immediately from the fact that n_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 for each vertex v,

$$\begin{aligned} n_{x(n-k+d-1),v} &\geq \left\lfloor \frac{x(n-k+d-1)}{x} \right\rfloor + k + 1 - d \\ &= n, \end{aligned}$$

and the result follows. ∎

In practice, we would expect $d = O(\ln m)$ and $n = \omega(\ln m)$ so the bound in Theorem 3.2 is approximately xn. Thus, we have constructed oblivous upload policies that are guaranteed to take no more than approximately Dn time steps. Conversely, for any D, there exist graphs where any routing will require at least Dn time steps. For example, consider the graph with vertices $v_1, v_2, \ldots, v_{D+1}$ 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 Dn 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 during 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 3.2 to BitTorrent-C graphs.

Corollary 3.3. *If clients alternate uploads among their neighbors in a predetermined order on a BitTorrent-C graph, then*

$$T \leq 3C(\ln m + 1)(n + 3\lg m - 1)$$

with probability $1 - \frac{2}{m}$.

Proof. This follows immediately from Theorem 3.1 and Lemma 2.1. □

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

3.2 On Other Graphs

Although it is a strong result for sparse graphs, Theorem 3.2 weakens very quickly as the maximum degree increases. By changing the policy slightly, however, it can still be applied successfully to very dense graphs.

Proposition 3.4. *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. Then, routing will complete with this upload policy in $3n + O(\ln m)$ 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 reduced G to a random 3-regular graph. By [5], such a graph is connected and has diameter $O(\ln m)$ with high probability. The result now follows immediately from Theorem 3.2. □

This bound is, in fact, essentially optimal for routing on any graph. If every block begins at one vertex, it will take $\log m$ time steps before every vertex has downloaded at least one block. Similarly, it will take n time steps before a vertex beginning with 0 blocks can download every block. Thus, routing will always take at least $\max(n, \log m) \geq \frac{1}{2}(n + \log m)$ time steps on any graph.

On the other hand, the algorithm described in Proposition 3.4 may seem a little artificial. In particular, intuition suggests it should not be any better than the more natural algorithm where each vertex uploads to a random neighbor at each time step. In fact, it is better, as shown in the following proposition.

Proposition 3.5. *Suppose a routing uses the upload policy of uploading to a random neighbor at each time step, and suppose the download policy ensures a client is equally likely to download any of the data blocks offered by a single upload request.*

Then, the routing will require an expected $\omega(n + \ln m)$ time steps on a complete graph where one vertex u begins with every block and the other vertices begin with nothing.

Proof. Omitted. □

3.3 Unbounded Uploads and Gossip

It is worth noting that Theorem 3.2 almost completely solves a related problem. Suppose we modify our model so that a vertex can upload simultaneously at full speed to all of its neighbors. This is unrealistic for file sharing but it has some theoretical interest as a gossip problem. Then, Theorem 3.2 guarantees that regardless of what block each vertex downloads at each time step, the routing will complete in at most twice the optimal time. Furthermore, the upper bound is tight in the sense that if every vertex begins with a unique block, there always exists a routing scheme that completes in precisely $n - k + d - 1 = n + d - 2$ time steps.

4 Randomized Uploads

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-$D(\ln n + 3\ln m)$ with high probability. Thus, Theorem 3.2 shows routing with this upload policy requires at most $D(\ln n + 3\ln m)(n + d)$ time steps. Since the randomized upload policy is so natural, and since it is tied to what BitTorrent does in practice, we improve this bound to $O(D(n + d))$ in this section.

4.1 A Reduced Problem

We will reduce the analysis of random upload policies to the case of a path. It is convenient, however, to analyze this simpler problem before presenting the reduction.

Towards that end, consider vertices, $v_0, v_1, \ldots, v_l$. We distribute n 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 $T_{x_i,n,l,p}$ 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 $T_{x_i,n,l,p}$ stochastically dominates $T_{x'_i,n,l,p}$.*

Proof. We prove this by induction on $k_{x_i} = \sum_{i=1}^{n} l - x_i$. 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 proves the intuitive result that moving a coin forward can only decrease the amount of time remaining. 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 n coins. Now, the number of coins at v_0 can be modeled as a Markov chain, which reaches a steady state since $q < p$. Furthermore, since the graph of states for this Markov chain is a tree, the process is reversible. Thus, as with Burke's theorem in the analagous case of an M/M/1 queue, the following two facts are true.

1. In the steady state, the probability that a coin is moved out of v_0 is q.
2. If v_0 is in the steady state, then v_0 will be in the steady state in the next time step regardless of its output.

The first claim follows immediately from reversability. For the second claim, suppose v_0 just output a coin. The probability that v_0 has i coins in this case is, by reversability, equal to the probability that v_0 has i coins given that it is just about to be input a coin. The claim now follows from the fact that the input is independent of the number of coins already in v_0. The case whre v_0 did not output a coin follow similarly.

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

Lemma 4.2. *Let n_i denote the number of coins at v_i. Then, the steady state of the given Markov chain occurs when each n_i is independently distributed as follows.*

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

Proof. We first consider the case where $l = 1$. Let $\pi_{i,j}$ denote the probability of transitioning from $n_0 = i$ to $n_0 = j$ in one time step, and let p_j denote the probability that $n_0 = j$ in the steady state. Then, by reversability, $p_{j+1} = p_j \frac{\pi_{j,j+1}}{\pi_{j+1,j}}$. 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 n_0 is as claimed.

For the general case, suppose the n_i are distributed as described above. Then, after one time step, each n_i individually will still be distributed as above since they each receive an input coin with probability q. Furthermore, since n_i does not depend on previous output from v_i, the n_i are still independent. The result follows. □

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

Proposition 4.3. *If $l' \geq l$, then $T_{x_i,n,l,p} \leq \frac{8l'+4n}{p}$ with probability at least $1 - 2\exp\left(-\frac{l'}{2}\right)$.*

Proof. Clearly, $T_{x_i,n,l',p}$ stochastically dominates $T_{x_i,n,l,p}$ 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 n of these coins to reach v_l. However, 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 4.2 is already in the steady state. Lemma 4.1 implies that the time for n coins to go from v_0 to v_l in this scenario stochastically dominates $T_{x_i,n,l,p}$.

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, $A = \sum_{i=0}^{l-1} n_i$ where n_i is specified as in Lemma 4.2. Note that for $j \geq 1$, $P[n_i \geq j] = \left(\frac{q(1-p)}{p(1-q)}\right)^{j-1} \cdot \frac{(p-q)q}{p^2(1-q)} \cdot \frac{1}{1-\frac{q(1-p)}{p(1-q)}}$, which simplifies to $\frac{q}{p \cdot 2^{j-1}} < \frac{1}{2^{j-1}}$. It follows that n_i is stochastically dominated by a geometric distribution with mean 2. Therefore, by the Chernoff bounds for the negative binomial distribution [10], A is at most $2l$ with probability $1 - \left(\frac{2^4}{3^3}\right)^l > 1 - \exp\left(-\frac{l}{2}\right)$.

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 $k = \frac{4l+2n}{q}$, the standard Chernoff bounds imply v_l receives $2l+n$ coins with probability at least $1 - \exp\left(-\frac{2l+n}{4}\right) > 1 - \exp\left(-\frac{l}{2}\right)$.

The result now follows from the fact that $\frac{4l+2n}{q} \leq \frac{8l+4n}{p}$. □

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.

Lemma 4.4. *Suppose that some vertex u in G begins with a copy of every block. Then, fix a vertex v and let $u = v_0, v_1, v_2, \ldots, v_l = v$ be a path from u to v. Also, let n_j denote the number of blocks that v_j has a copy of, and let $x_i = \max\{j | \ n_k \geq n+1-i \ \forall k \leq j\}$. Finally, let T_v denote the number of time steps before v has downloaded a copy of every block.*

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

Proof. Once again, we prove this by induction on $k_{x_i} = \sum_{i=1}^{n} l - x_i$. 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 $n_{x_i} \geq n+1-i > n_{x_i+1}$. On the other hand, since $x_{i+1} > x_i$, we can further say that $n_{x_i+1} = n - i$. Now, v_{x_i} will request an upload to v_{x_i+1} with probability at least $\frac{1}{D}$. If this happens, n_{x_i+1} will increase to $n-i+1$ 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 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 that 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. ☐

We can now state our main result for the randomized upload policy.

Theorem 4.5. *Suppose a vertex u begins with a copy of every block. Let D denote the maximum outdegree in G, and suppose the distance from u to every other vertex is at most d.*

If we rout on this graph using the randomized upload policy, then $T \leq 4D(2d+n)$ with probability at least $1 - 2m \exp\left(-\frac{d}{2}\right)$.

Proof. This follows immediately from Proposition 4.3 and Lemma 4.4. ☐

Since the proof of this theorem restricts to a single path, it holds as long as there exist paths from the original server to every other vertex that remain throughout the routing, even as other vertices are deleted. Also, as with Theorem 3.2, this result can be trivially improved when every vertex begins with a non-zero number of data blocks.

Finally, we apply Theorem 4.5 to BitTorrent-C graphs.

Corollary 4.6. *Let G be a BitTorrent-C graph where the initial vertex begins with a copy of every block. If we rout on G using a randomized upload policy, then $T \leq 12C(\ln m + 1)(n + 6 \lg m)$ with probability at least $1 - \frac{4}{m}$.*

Proof. By Lemma 2.1, G has maximum degree at most $3C(\ln m + 1)$ and depth at most $3 \lg m$ with probability $1 - \frac{2}{m}$. Therefore, by Theorem 4.5, $T \leq 12C(\ln m + 1)(n + 6 \lg m)$ with probability at least $1 - \frac{2}{m} - 2m \exp\left(-\frac{3 \lg m}{2}\right) \geq 1 - \frac{4}{m}$. ☐

In most applications, we would again expect n to be much larger than $\lg m$, so this reduces to approximately $12Cn \ln m$. A variant of Proposition 3.4 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. This dependence is not entirely necessary, and in this section, we show how to strengthen our results for BitTorrent-C graphs.

Suppose a graph has the following properties.

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

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 X_0 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 $X_0 = \{v_1, v_2, \ldots, v_C\}$ and for $i > 0$, letting X_i contain the vertices with median X_0 depth of i. It follows from Lemma 2.1 that G is an A-D-d tree with $A = \frac{C}{2}$, $D = 3C(\ln m + 1)$ and $d \leq 3 \lg m$ with probability $1 - \frac{2}{m}$.

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 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 $\frac{2\frac{D}{A}}{1-\frac{1}{e}} \cdot (d(4\ln n + 8\ln m) + n)$ time steps with probability at least $1 - \frac{1}{m}$.*

Proof. Let C be an arbitrary constant. We say epoch k begins when every vertex in X_i has at least $C(k-i)$ blocks for each i. Note that epoch 1 begins at the first time step and the block distribution ends at the end of epoch $d + \frac{n}{C}$.

Consider a vertex $v \in X_i$ during epoch k. Suppose v has less than $C(k-i+1)$ 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 $C(k-i+1)$ blocks, the number of blocks that v downloads in $k\frac{D}{A}$ time steps stochastically dominates the sum of k independent random variables that are 1 with probability $1 - \frac{1}{e}$ and 0 otherwise. Since v has at least $C(k-i)$ blocks at the beginning of epoch i, it follows from the Chernoff bounds that v has at least $C(k-i+1)$ blocks $\frac{2C\frac{D}{A}}{1-\frac{1}{e}}$ time steps after the beginning of epoch i with probability at least $1 - \exp(-\frac{C}{4})$.

There are m vertices, and each one must increase its number of blocks by C at most $\frac{n}{C} \leq n$ times. Thus, with probability at least $1 - mn\exp\left(-\frac{C}{4}\right)$, it holds for each i and k that each vertex in X_i will have at least $C(k-i+1)$ blocks within $\frac{2C\frac{D}{A}}{1-\frac{1}{e}}$ time steps after the start of epoch k. Therefore, with probability $1 - mn\exp\left(-\frac{C}{4}\right)$, each epoch will last at most $\frac{2C\frac{D}{A}}{1-\frac{1}{e}}$ time steps, and the process completes in at most

$$\frac{2C\frac{D}{A}}{1-\frac{1}{e}} \cdot \left(d + \frac{n}{C}\right) = \frac{2\frac{D}{A}}{1-\frac{1}{e}} \cdot (dC + n)$$

time steps. In particular, the desired result follows from taking $C = 4\ln n + 8\ln m$. □

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. It is also worth mentioning that, like the previous two theorems, Theorem 5.1 can be trivially improved if every vertex begins with more than zero data blocks.

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 5.2. *Consider a BitTorrent-C network where the first C vertices begin with every data block. If we rout using the standard randomized upload policy, the routing will complete in at most $9.5(\ln m + 1)(n + 12(\ln n)(\ln m) + 24(\ln m)^2)$ with probability $1 - \frac{3}{m}$.*

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

In most applications, we would again expect n to be much larger than $(\lg m)^2$, so this reduces to approximately $9.5n \ln m$. In the next section, we show a related family of graphs for which this bound achieves an even tighter result.

6 Extensions to BitTorrent

6.1 Smoothed BitTorrent-C Graphs

We first consider a practical variant of BitTorrent-C graphs, which performs even better in our analysis. The "practical" condition here is very important. As shown in Section 3.2, there exist algorithms that perform very well on random regular graphs, but such algorithms restrict 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 extremely well in theory, but it is again extremely sensitive to graph changes.

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

Lemma 6.1. *If $C = C_0 \ln m$, every vertex in a smoothed BitTorrent-C graph has median depth $O(\log m)$ and out-degree $O(C)$ with probability at least $1 - \frac{2}{m}$.*

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 $O(\log m)$ with probability $1 - \frac{1}{m^2}$.

We now consider the out-degree of vertices. 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 $\frac{2j-1}{i^2}$. Thus, the probability that there is an edge from v_j to v_i is at most $\frac{2C_0 j \ln m}{i^2}$. Again, these probabilities are independent for distinct i. Thus, the out-degree of v_j is stochastically dominated by the sum of independent random variables $X_j, X_{j+1}, \ldots, X_m$ where X_i is 1 with probability $\frac{2C_0 j \ln m}{i^2}$ and 0 otherwise.

Now, note that

$$\begin{aligned}
E\left[\sum_{i=j}^{m} X_i\right] &= 2C_0 j \ln m \sum_{i=j}^{m} \frac{1}{i^2} \\
&< 2C_0 j \ln m \left(\frac{1}{j^2} + \sum_{i=j+1}^{m}\left(\frac{1}{i-1} - \frac{1}{i}\right)\right) \\
&< 2C_0 j \ln m \left(\frac{1}{j^2} + \frac{1}{j}\right) \\
&\leq 4C_0 \ln m.
\end{aligned}$$

It follows from the Chernoff bounds that $\sum_{i=j}^{m} X_i = O(C_0 \ln m)$ with probability $1 - \frac{1}{m^2}$. □

Corollary 6.2. *Consider a smoothed BitTorrent-C network where the first C vertices begin with every data block. If we rout using the standard randomized upload policy, the routing will complete in at most $O(n + (\ln m)^2)$ time with probability $1 - \frac{3}{m}$.*

Proof. It follows from Lemma 6.1 that the graph is an A-D-d tree with $A = C$, $D = O(C)$ and $d = O(\ln m)$ with probability $1 - \frac{2}{m}$. The result now follows from Theorem 5.1. □

As noted before, the optimal running time is $\Omega(n + \ln m)$, and in practice, we would expect n to be much larger than $\ln m$, so this bound is near optimal.

6.2 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 that causes each client to usually choose 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, our results all hold for any download policy. Thus, BitTorrent could switch to the intuitively inferior policy of always downloading data blocks in order and it could still perform well. This would cause a few practical problems, most notably that clients could leave the network as soon as they downloaded the last block and thus downloading that block would become extremely slow. If this issue were resolved, however, our work suggests streaming is entirely feasible.

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 Protocol Specification v1.0. http://wiki.theory.org/BitTorrentSpecification.

[5] B. Bollobás and F. De La Vega. The diameter of random regular graphs. *Combinatorica*, 2(no. 2):125-134, 1982.

[6] 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*, 1987.

[7] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. Technical Report MSR-TR-2004-80, Microsoft Research, 2004.

[8] T. Hagerup and C. Rüb. A guided tour of chernoff bounds. Information Processing Letters, 33:305-308, 1990.

[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 Liftime. *PAM*, Apr. 2004.

[10] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice-Hall, Englewood Cliffs, NJ, 1994.

[11] 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.

[12] D. Qiu and R. Srikant, Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks. *SIGCOMM*, Sep. 2004.

[13] X. Yang and G. de Veciana, Service Capacity of Peer to Peer Networks. In *Proceedings of IEEE INFOCOM*, 2004.