Provably stable, distributed file sharing protocols

Barlas Oguz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-6
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-6.html

January 6, 2012

Provably stable, distributed file sharing protocols

by

Barlas Oğuz

Abstract

Provably stable, distributed file sharing protocols

by

Barlas Oğuz

Master of Science in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Venkat Anantharam, Chair

P2P systems provide a scalable solution for distributing large files in a network. The file is split into many chunks, and peers contact other peers to collect missing chunks to eventually complete the entire file. The so-called 'rare chunk' phenomenon, where a single chunk becomes rare and prevents peers from completing the file, is a threat to the stability of such systems. A protocol that forces peers to download the rarest chunk in the network first would solve this issue, however this solution requires global chunk availability information that is not readily available. We look for a distributed approximation to this solution. Although heuristics exist which perform well in practice, formal proofs of stability of such systems were lacking. In this document, we demonstrate a new system based on an approximate rare-chunk rule, allowing for completely distributed file sharing while retaining scalability and stability. We assume non-altruistic peers and the seed is required to make only a minimal contribution.

Professor Venkat Anantharam
Committee chair, advisor

Contents

List of Figures

Chapter 1

Introduction

The marvel of peer-to-peer (P2P) networks is their scalability and robustness. Both of these attributes are due in turn to the distributed nature of such systems. In a P2P file sharing system such as *BitTorrent* [2], a large file is split into many chunks. A peer who downloads a chunk can immediately start uploading that chunk to other peers, contributing to the resource pool of the sharing network.

To ensure the availability of all chunks of the file at all times, at least 1 seed (who has the complete file) is assumed to stay in the network at all times. However, in an open system, where peers are arriving according to a random arrival process, the presence of a single seed does not guarantee stability. It has been observed and demonstrated through various analytical models ([1], [3]–[5]) that if the peers contact each other and download chunks in a purely random fashion, a single chunk might be driven to near extinction, causing peers to stay in the system for a long time and driving the number of peers in the system to infinity. In this scenario, the rare chunk is difficult to obtain, because there is a high probability that the randomly contacted peer does not have it, and peers who have the rare chunk tend to leave the system quickly since they probably already have every other chunk. Thus the rare chunk becomes progressively rarer as new peers accumulate.

It is known that the rare chunk issue can be avoided through altruistic behavior of the peers [1], [6]. In this model, peers who complete the file stay in the system for an additional

random time to aid the remaining peers. It has recently been proven [7] that it is sufficient for peers to remain in the network for a time that is on average equal to the time it takes to download a single chunk. Unfortunately in real networks, the altruistic peer assumption may not hold.

A similar stabilizing effect is observed if the upload capacity of the seed is large enough to maintain a balanced chunk distribution in the system [7], [8]. However, in such scenarios, the demand on the seed scales with the number of peers in the system, reducing the scalability of the protocol.

In this paper, we present a new P2P file sharing protocol that is provably stable. Our protocol is completely distributed, and fully scalable, avoiding the pitfalls of a centralized tracker or a privileged seed. Moreover, we assume that peers who complete the download leave immediately. Stability depends on a probabilistic local rule that peers follow to approximate prioritizing the rare chunk.

We model an open system, where peers are arriving according to a Poisson process. In our model, the current chunk profiles of all the peers in the system defines the state for a Markov process. This is in line with the model described in [8]. In contrast with deterministic fluid models such as the one used in [6], we attempt to directly prove the stability of the dynamic system. This approach is more reassuring, as a well defined relationship between the stability of the dynamical system and that of the fluid models is yet to be formulated [4].

The protocol is presented in section 4.1. It is a modification of the 'majority' rule that was first proposed in [9], [10], where the authors used simulations and large system limits to argue that the rule leads to a stable system. The special 2-chunk case was studied in [4]. In section 3.4 we give a formal proof of stability for this special case within the framework introduced in that work. Stability of this protocol for the general case, however, remains a conjecture.

Our proposal implements a stricter rule to keep the rare chunks in the system longer, and allows us to prove stability in general for any number of chunks and any arrival rate.

The proof, presented in section 4.2, employs an unconventional Lyapunov function, which is the main contribution of this paper together with the new protocol. The form of the Lyapunov function is quite novel and might be useful in proving the stability of similar algorithms.

Chapter 2

BitTorrent protocol

The original BitTorrent protocol consists of a tracker and a swarm of peers. The purpose of the tracker is to keep a list of peers and serve a list of random connections upon request. Each incoming peer connects to the tracker to obtain a list of ip addresses of peers to download from. The peer maintains TCP connections for each of its neighbors (typically 25-55 connections). This results in a random graph structure which has proved to be robust and efficient. At any one time, only a fixed number (4 in the original specification) of these connections are said to be 'unchoked' (actively participating in downloading). The other connections may be used to opportunistically alter the set of 'unchoked' peers.

Information about the shared file is stored in a static file. The size of the shared file, the size of each piece, and a 20-byte hash of each piece is stored in this file. Most commonly, the file is divided into chunks of fixed size that is a power of two, most commonly $2^{18} = 256$ Kb (BitTorrent prior to version 3.2 uses $2^{20} = 1$ Mb as default).

2.1 Chunk availability

The original BitTorrent protocol specifies that a peer broadcasts the completion of each chunk download to each of its neighbors. This information is used to determine 'interest'. A download only happens if a peer is 'interested' in downloading from another peer and the

connection is 'unchoked'. The information about the availability of chunks may also be used to determine the order in which to download chunks. Random ordering might be sufficient most of the time, however, as discussed in the next chapter, this is insufficient to ensure stability. A better algorithm is to download the rarest chunk. Ideally, one would like to do this based on global information over the entire swarm. In BitTorrent, full information is available from (around 50) neighboring peers. This has proved to be sufficient in practice, although from the perspective of theory, the effectiveness of this protocol is open, as well as questions regarding how large a sample one needs to ensure good performance. These are the central questions that we will try to address in this document.

2.2 Choking - unchoking and tit-for-tat incentive mechanisms

In BitTorrent, a peer 'unchokes' (picks) at most 4 other peers in its set of neighbors. This set is optimized based on the upload capacities of the neighboring peers. At specified intervals, the peer will opportunistically 'unchoke' a random peer in its neighbor set, and choke the worst performing active peer. This way, the protocol aims to efficiently utilize all upload capacity, as well as providing a tit-for-tat incentive mechanism for encouraging the supply of more upload capacity by the peers. Here, we will not concern ourselves with incentive mechanisms. We will model a uniform system where all peers have a fixed upload capacity and will assume that peers cannot 'cheat' the protocol. Details of the choking - unchoking mechanism of BitTorrent and its robustness properties can be found in [2]. The BitTorrent specification can be found at bittorrent.org.

Chapter 3

Analytic model and the rare chunk heuristic

3.1 Model

In our model, we attempt to capture the dynamics of a basic file sharing system with homogeneous peers. We model an open system, where peers arrive according to a Poisson process with rate λ. Peers leave immediately upon receiving all k chunks, in other words, peers are non-altruistic. A file is divided into k chunks, to be distributed in a P2P system. At a Poisson rate of 1, each peer can attempt to download a chunk which they desire, by sampling a number of other peers at random from the swarm. We assume downloads to be instantaneous, however the Poisson model ensures that the average download rate of each peer is capped at 1.

We seek a rule by which a peer can decide which chunk (if any) to download at each time slot from the current sample. This rule is assumed to be a function of the peer's current chunk profile, and the profiles of the sampled peers. (The rule could possibly depend on the past observations of the peer, as well as λ and k. However the rule we propose will not depend on these.) Having decided such a rule, the dynamics of the system become well

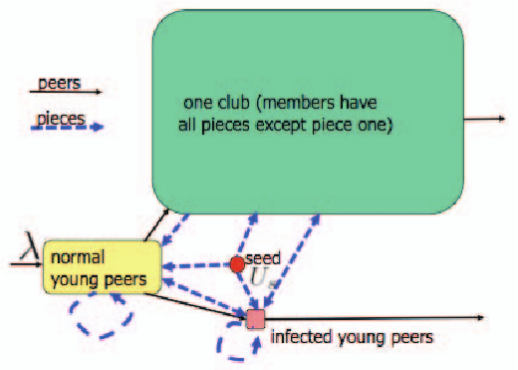

Figure 3.1. Instability of random chunk selection algorithm. (Reproduced from [1]).

defined and we will ask whether the resulting stochastic system is stable in the Lyapunov sense.

3.2 Missing chunk syndrome

Consider a purely randomized rule, where at each sampling instant, a peer contacts a single other peer and downloads a random chunk that he does not already posses. If the contacted peer has no such chunk, the download attempt is said to have failed. Otherwise, the peer who initiated contact adds a new chunk to his portfolio instantaneously. This random chunk selection algorithm has been demonstrated through various analytical models ([1], [3]–[5]) to be unstable. Figure 3.1 [1] illustrates how this instability comes about.

Note that with the random chunk selection rule, the stochastic system is Markov, with state summarized by the current chunk profiles of each peer in the network. Consider a state where the chunk distribution is severely unbalanced. The vast majority of peers already have all chunks except one (so called 'one-club' peers in the figure), a very small number of peers having the remaining rare-chunk (labeled as 'infected peers' in the figure), and a number of normal young peers are in neither of these sets. Since the Markov process is irreducible, there is positive probability of reaching such a state. If the peers contact each other and download chunks in a purely random fashion, the rare chunk is difficult to obtain, because there is a high probability that the randomly contacted peer does not have it, and

peers who have the rare chunk tend to leave the system quickly since they probably already have every other chunk. The normal young peers are likely to join the one-club peers before becoming infected, and thus do not contribute in serving the rare chunk to the rest of the network. Thus the rare chunk becomes progressively rarer as new peers accumulate.

The 'rare chunk' issue can be addressed by forcing peers to download the rarest chunk in the system first. This would ensure that new peers obtain the rare chunk before becoming 'one-club' peers, and thus the availability of rare chunks would increase with time. Balancing out the chunk distribution in this way stabilizes the system.

BitTorrent clients address the 'rare chunk' issue by forcing peers to download the rarest chunk in the set of neighboring peers first. As explained in the last chapter, this information is fully available. The set of neighboring peers (around 50 peers) constitutes a good chunk of the overall population in practice, and so this is found to be a good approximation to a global rarest-first algorithm. However, it is an open question as to whether this protocol is stabilizing. Moreover, the performance of BitTorrent is sensitive to the number of neighboring peers, as a larger number causes more overhead due to more TCP connections and the need to broadcast chunk availability data to all neighbors. We will show in the following sections that sampling only 3 neighbors at random suffices for the stability of the system.

3.3 Approximate rare-chunk heuristic

A simple approximation for the global rare-chunk first rule was described in [5].

The following is assumed:

- There is always exactly 1 seed in the system (denoted by s), who has all the chunks.
- At a Poisson rate of 1, each peer can sample randomly with replacement up to 3 peers from the current population $\cup\{s\}$. (A peer is allowed to sample itself.)
- At the time of sampling, the peer can choose to download at most 1 chunk which it does not already have, but shows up in the sample. The download is assumed to happen instantaneously.

- A newly arriving peer arrives with no chunks.

Let S be the total number of peers in the system (including the seed). Note that each peer (including the seed) can be sampled, on average, at most by three other peers per unit time. Therefore the average upload bandwidth per peer is bounded, and does not scale with λ or S. This is crucial for scalability.

Definition 3.3.1. *A chunk in a sample of* 3 *peers is* rare *if exactly* 1 *peer in the sample has that chunk. A chunk is called a* match *if it is contained in the sample but not in the sampling peer's profile.*

The rare-chunk protocol dictates that at the time of sampling, the peer is allowed to download a new chunk if and only if that chunk is a rare match. If there is more than one such chunk in the sample, the peers downloads one of these at random. If there are no such chunks, the attempt is skipped with no downloads taking place.

This protocol is shown to have good stability properties in simulation [9], however a formal stability proof is missing.

3.4 A stable protocol for the two chunk case

We now present a proof of stability for a slightly modified version of this protocol, albeit valid only for the 2-chunk case. This case was studied in [4], however a formal proof could not be found. Let x be the number of peers with chunk 1 (excluding the seed), and y be the number of peers with chunk 2 (excluding the seed). z is the number of peers with no chunks. The setup is slightly different. The sampling is done on a set that is restricted to only peers who have at least 1 chunk. Therefore, peers in z do not join the sampling pool until they download at least 1 chunk. This model actually violates the property that the average upload rate of peers should remain bounded, since if $z >> x, y$, the peers who have chunks will need to upload to a large number of peers simultaneously. The proof for this case is given for reference.

It is assumed that the peers in z interpret the seed as having either chunk 1 or 2 with equal probability, whenever it shows up in the sample.

The protocol dictates the following:

- Peers in z sample 3 peers at random and download only if the sample contains a rare chunk.
- Peers in x and y sample a single peer and download if they find a matching chunk.

Proof. Recall that the stochastic process of interest is a Markov process with state defined by the current peers chunk profiles. Let $q(\mathbf{x}, \mathbf{x}')$ denote the entries of the generator matrix of this Markov process. For any function on the state space,

Definition 3.4.1. *The* drift $\Delta f(\mathbf{x})$ *of a function* $f(\mathbf{x})$ *is defined as*

$$\Delta f(\mathbf{x}) = \sum_{\mathbf{x}' \neq \mathbf{x}} q(\mathbf{x}, \mathbf{x}')(f(\mathbf{x}') - f(\mathbf{x})).$$

We use the following well known tool for the proof:

Theorem 3.4.2. *[Foster-Lyapunov] Let L be a function on the state space with drift ΔL. Let $L \geq 0$ and let $\{L \leq l\}$ be a finite set for any finite constant $l > 0$. If for an $\epsilon > 0$, $\Delta L < -\epsilon$ on the set $S > c$, for a suitably chosen constant c, then the Markov process is positive recurrent.*

Such an L is called a Lyapunov function. For this stochastic system, we propose the following :

$$L(x, y, z) = 2z + 2(x + y + z) + (x - y)^2$$

We calculate the drift as follows:

$$E[\Delta z_n | \mathcal{F}_n] = \lambda - \frac{3(x + 1/2)(y + 1/2)z}{(x + y + 1)^2}$$

$$E[\Delta (x + y + z)_n | \mathcal{F}_n] = \lambda - \frac{x(y + 1) + y(x + 1)}{(x + y + 1)}$$

$$E[\Delta (x-y)_n^2 | \mathcal{F}_n] = (2(x-y)+1)dX^+ + (2(y-x)+1)dY^+ + (2(y-x)+1)dX^- + (2(x-y)+1)dY^-$$

Where

$$dX^+ = \frac{3(x+1/2)(y+1/2)^2 z}{(x+y+1)^3}, \quad dY^+ = \frac{3(x+1/2)^2(y+1/2)z}{(x+y+1)^3}$$

$$dX^- = \frac{x(y+1)}{(x+y+1)}, \quad dY^- = \frac{y(x+1)}{(x+y+1)}$$

After calculations we get

$$E[\Delta(x-y)_n^2|\mathcal{F}_n] = \frac{3(x+1/2)(y+1/2)z}{(x+y+1)^2} + \frac{x(y+1)+y(x+1)}{(x+y+1)} - \frac{6(x-y)^2(x+1/2)(y+1/2)z}{(x+y+1)^3} - \frac{2(x-y)^2}{(x+y+1)}$$

Summing up, we get

$$E[\Delta L_n|\mathcal{F}_n] = 4\lambda - \frac{3(x+1/2)(y+1/2)z}{(x+y+1)^2} - \frac{x(y+1)+y(x+1)}{(x+y+1)} - \frac{6(x-y)^2(x+1/2)(y+1/2)z}{(x+y+1)^3} - \frac{2(x-y)^2}{(x+y+1)}$$

We will argue that this is strictly negative on the set $max(x, y, z/(20\lambda+1)^2) > 10\lambda$. First note

$$\frac{x(y+1)+y(x+1)}{(x+y+1)} + \frac{2(x-y)^2}{(x+y+1)} \geq \frac{x^2+y^2+x+y}{(x+y+1)} \geq \frac{1}{2}max(x,y)$$

Omitting the third term which is negative, we have

$$E[\Delta g_n|\mathcal{F}_n] < 4\lambda - \frac{3(x+1/2)(y+1/2)z}{(x+y+1)^2} - \frac{1}{2}max(x,y)$$

One can easily verify that if $max(x, y) \geq 10\lambda$ then the last term is less than -5λ. Otherwise, $z \geq 10\lambda(20\lambda+1)^2$, in which case the middle term is less than -7λ. In both cases, we have a negative drift of at least λ. Theorem 3.4.2 completes the proof.

Obviously the set $max(x, y, z/(20\lambda+1)^2) > 10\lambda$ is much larger than needed. One can come up with a better set using the omitted term, and better bounds. However, this is not necessary to prove stability.

□

Unfortunately, the Lyapunov function used above does not work for the more general, many-chunk case. The reasoning is as follows. The quantity $z+(x+y+z)$ can be interpreted as the total number of missing chunks in the system (peers in z have 2 missing chunks, peers

in x and y have just 1). This quantity decreases by 1 every time a download happens, and increases by 2 every time a new peer arrives. Thus the drift of this quantity can be written as $\lambda - r$ where r is the total rate of downloads. When the chunk distribution in the system is balanced, the rate of downloads will be roughly proportional to the number of peers in the system. Thus the drift of this function will be negative with enough peers. When the chunk distribution is severely imbalanced (recall the missing piece syndrome), this quantity will have positive drift that is at most λ. The $(x-y)^2$ term however, always has negative drift and will cause the overall drift to be negative. This is due to the following fact:

Lemma 3.4.3.

$$P(\textit{chunk 1 is a rare match}) \geq P(\textit{chunk 2 is a rare match})$$

$$\iff$$

$$x < y.$$

This lemma is only true for the two chunk case, since when there are more chunks, a chunk might become so rare that the probability of observing it 0 times in a sample of 3 may become very large. Thus more common chunks might have a higher probability of being a 'rare match'. This difficulty makes the general case much more complicated and thus this protocol remains a heuristic.

Chapter 4

A provably stable protocol

In this chapter we introduce a modified p2p protocol which can be formally proven to be stable. The protocol is completely distributed and operates under random sampling and the non-altruistic peer assumption. It provides stability under any arrival rate, with only a single seed with unit upload capacity. This is the first such result under these rather restrictive assumptions.

4.1 Common chunk protocol

Let $S_i = \#$ peers who have chunk i including the seed, $i \in \{1, \ldots, k\}$.

$S_0 = \#$ peers who have no chunks.

$\bar{S}_i = S - S_i$.

$\bar{T}_i = \#$ peers who have only chunk i missing.

Definition 4.1.1. *A chunk in a sample of* 3 *peers is* rare *if exactly* 1 *peer in the sample has that chunk. A chunk is called a* match *if it is contained in the sample but not in the sampling peer's profile.*

We define the rule as follows:

- Peers with no chunks sample 3 peers at random and choose to download a chunk that is a rare match. If there is more than 1, they pick randomly among them. If there are no rare matches, the peer skips this time slot without downloading.

- Peers who have more than zero, but less than $k-1$ chunks sample only 1 peer at random, and download a chunk at random among those that match (no rare match required). Skip if there is no match.

- Peers who have $k-1$ chunks sample 3 peers at random. Download only if every chunk that the peer has appears at least twice in the sample, and there is a match. Otherwise skip without downloading.

Roughly, the first item is meant to stop arriving peers from acquiring a common chunk as their first chunk. The last item attempts to keep rare chunks from leaving the system. This should balance out the chunk distribution in the system and provide stability.

Note that by sampling only 3 peers, we are requiring the bare minimum that allows a majority rule. By sampling more peers, one could clearly do better, however our main purpose here is to demonstrate that stability is possible even in this restricted setting. In section 4.2.1, we introduce a parametric class of protocols which allow top level peers to sample up to m other peers. The parameter m can be adjusted to get a good trade-off between locality (availability of information) and performance. Performance comparisons are made through simulation in chapter 5.

In this paper, we model the proposed system by a Markov process with state space $\mathcal{X}$ described by the peers currently in the system and their chunk profiles. The description of the state space is essentially identical to that in [8]. Let $q(\mathbf{x}, \mathbf{x}')$ denote the entries of the generator matrix of this Markov process. For any function on the state space,

Definition 4.1.2. *The* drift $\Delta f(\mathbf{x})$ *of a function* $f(\mathbf{x})$ *is defined as*

$$\Delta f(\mathbf{x}) = \sum_{\mathbf{x}' \neq \mathbf{x}} q(\mathbf{x}, \mathbf{x}')(f(\mathbf{x}') - f(\mathbf{x})).$$

4.2 Proof of stability

We will split the proof into two cases according to whether $\lambda \leq \frac{1}{3k}$ or $\lambda > \frac{1}{3k}$. In each case, we will show the stability of this system by demonstrating a Lyapunov function for it.

Let r be the total rate of downloads. dS_i^+ is the virtual rate (stochastic intensity) at which a peer with no chunks downloads chunk i, and $d\bar{S}_i^-$ is the virtual rate at which a peer who is lacking only chunk i downloads i and leaves the system. We will need the following lemmas:

Lemma 4.2.1. *$r \geq \frac{Sr_0^2}{6k^2}$ where $r_0 = \sum_i dS_i^+$.*

Proof. Write $\bar{S}_i = S_0 + \bar{B}_i + \bar{T}_i$ to define $\bar{B}_i$. $\bar{B}_i$ is the number of peers who lack chunk i and have at least 1 and at most $k-2$ chunks. We can write

$$r \geq S_0 d(S_0) + \bar{T}_i d(\bar{T}_i) + \bar{B}_i d(\bar{B}_i)$$

where $d(.)$ denotes the virtual rate of downloads for an individual peer in each group. By definition, $d(S_0) = \sum_i dS_i^+ \geq dS_j^+$ for any j. Also

$$d(\bar{T}_i) \geq \frac{3\bar{T}_i^2 S_i}{S^3} = \frac{\bar{T}_i^2}{\bar{S}_i^2}\frac{3\bar{S}_i^2 S_i}{S^3} \geq \frac{\bar{T}_i^2}{\bar{S}_i^2} dS_i^+$$

where the last inequality follows because $\frac{3\bar{S}_i^2 S_i}{S^3}$ is the probability of a rare match, but dS_i^+ might be smaller due to the possibility of multiple rare matches. It is left to note $d(\bar{B}_i) \geq \frac{dS_i^+}{3}$. To argue this, note that $d(\bar{B}_i)$ is the total virtual rate of downloads per peer for peers in $\bar{B}_i$, and a sample of chunk i is sufficient to result in a download (even if chunk i is not chosen to be downloaded). The probability of sampling chunk i in 1 go is at least $\frac{1}{3}$ the probability of sampling chunk i in 3 tries, which in turn at least as large as dS_i^+. We can now write

$$r \geq (S_0 + \frac{1}{3}\bar{B}_i) dS_i^+ + \frac{\bar{T}_i^3}{\bar{S}_i^2} dS_i^+ \geq \frac{\bar{S}_i}{6} dS_i^+.$$

Here we argue that $x + y + z = 1 \implies x + \frac{y}{3} + z^3 \geq \frac{1}{6}$.

$$\frac{\bar{S}_i}{S} \geq dS_i^+.$$

Therefore

$$r \geq \max_i \frac{S(dS_i^+)^2}{6} \geq \frac{r_0^2 S}{6k^2}.$$

□

Lemma 4.2.2.

$$r \geq \min_i \frac{S_i}{2k^3}, \quad \textit{if } S \geq 12.$$

Proof. Argue as in the previous lemma that for S_0, a rare match of i is sufficient for a download. For $\bar{B}_i$, a simple match is sufficient and $\frac{S_i}{S} \geq \frac{\bar{S}_i^2 S_i}{S^3}$. Therefore

$$r \geq (3S_0 + \bar{B}_i)\frac{\bar{S}_i^2 S_i}{S^3} + 3\frac{\bar{T}_i^3}{\bar{S}_i^2}\frac{\bar{S}_i^2 S_i}{S^3} \geq \frac{2\bar{S}_i^3 S_i}{3S^3}, \text{ for any } i.$$

Here we argue that $x + y + z = 1 \implies x + \frac{y}{3} + z^3 \geq \frac{2}{9}$. Since $\max_i \bar{S}_i \geq \frac{S-1}{k}$, we have the result when $S \geq 12$. □

First consider the case $\lambda \leq \frac{1}{3k}$.

We propose:

$$L_1 = \sum_i \bar{S}_i$$

The drift of L is

$$\Delta L_1 = k\lambda - r \leq \frac{1}{3} - r.$$

From lemma 4.2.2, we know

$$r > \frac{\bar{S}_i^3 S_i}{2S^3}, \text{ for any } i.$$

Picking i to be the rarest chunk, we observe that $r > \frac{1}{3} + \epsilon$ whenever $S > 3k^3$.

Now assume $\lambda \geq \frac{1}{3k}$.

We propose:

$$L = C\underbrace{\sum_i \bar{S}_i}_{L_1} + \underbrace{\sum_i \frac{S}{e^{S_i}} + \frac{S}{e^{S_0}}}_{L_2}$$

where C is a constant to be chosen later.

We will calculate the drift of L in two parts. A download of chunk i decreases $\bar{S}_i$ by one and leaves all other $\bar{S}_j$ unchanged. A new arrival increases each $\bar{S}_i$ by one. Therefore

$$\Delta L_1 = k\lambda - r.$$

Also

$$\begin{aligned}\Delta L_2 \leq & \sum_i \frac{\lambda}{e^{S_i}} - \lambda\left(\frac{S}{e^{S_0}} - \frac{S+1}{e^{S_0+1}}\right) \\ & - \sum_i S_0 S d S_i^+ \left[\frac{1}{e^{S_i}} - \frac{1}{e^{S_i+1}} + \frac{1}{e^{S_0}} - \frac{1}{e^{S_0-1}}\right] \\ & + \sum_i \bar{T}_i d\bar{S}_i^- \sum_{j\neq i}\left(\frac{S-1}{e^{S_j-1}} - \frac{S}{e^{S_j}}\right)\end{aligned}$$

The first two terms correspond to the the arrival of a new peer. The second term is the drift due to a peer with no chunks downloading chunk i. The last term corresponds to the event where a peer leaves the system after having downloaded chunk i.

The inequality is due to the fact that we omitted terms corresponding to transitions which keep S and S_0 constant, and in the last set of terms, for each individual i we ignored the terms corresponding to S_i and S_0. These transactions can only decrease L_2.

Since $d\bar{S}_i^- \leq \min_{j\neq i} \frac{3S_j^2}{S^2}$ and $\bar{T}_i \leq S_j, \quad \forall j \neq i$, the last term satisfies

$$\begin{aligned}& \sum_j \left(\frac{S-1}{e^{S_j-1}} - \frac{S}{e^{S_j}}\right) \sum_{i\neq j} \bar{T}_i d\bar{S}_i^- \\ & \qquad \leq \sum_j 3(e-1)(k-1)\frac{\bar{S}_j^3}{S^2}\frac{S}{e^{S_j}} < \frac{8k^2}{S}. \end{aligned} \tag{4.1}$$

We get

$$\begin{aligned}\Delta L_2 < & \sum_i \frac{\lambda}{e^{S_i}} + \frac{\lambda}{e^{S_0}} - \frac{\lambda(e-1)S}{e\cdot e^{S_0}} \\ & + \sum_i (e-1) S_0 S d S_i^+ \left[\frac{1}{e^{S_0}} - \frac{1}{e\cdot e^{S_i}}\right] + \frac{8k^2}{S}\end{aligned}$$

Since $\sum_i \frac{\lambda}{e^{S_i}} + \frac{\lambda}{e^{S_0}} < (k+1)\lambda$, we may write

$$\begin{aligned}\Delta L < & \frac{8k^2}{S} + \lambda + (C+1)k\lambda - Cr - \frac{\lambda(e-1)S}{e\cdot e^{S_0}} \\ & + \sum_i (e-1) S_0 S d S_i^+ \left[\frac{1}{e^{S_0}} - \frac{1}{e\cdot e^{S_i}}\right]\end{aligned}$$

Now we are ready to show

Theorem 4.2.3.

$$\Delta L < -\epsilon$$

with $\epsilon > 0$ *and* $C = 108ek^3$ *whenever* $S > 4Cke^{3C\lambda k^2 e^{6\lambda k^4}}$.

Proof. Since we assume that $\frac{8k^2}{S} < 1$, we are left with 5 terms which can be written as

$$\Delta L < [1 + \lambda + (C+1)k\lambda] - Cr - \frac{\lambda(e-1)S}{e \cdot e^{S_0}} + \frac{(e-1)S_0 r_0 S}{e^{S_0}} - \sum_i \frac{(e-1)S_0 S dS_i^+}{e \cdot e^{S_i}}$$

- If $r \geq 2k\lambda$:
  - If $\frac{(e-1)S_0 S r_0}{e^{S_0}} \geq \frac{Cr}{3}$: Then $r_0 \leq \frac{S_0}{6eke^{S_0}}$ by lemma 4.2.1 and $r_0 S_0 \leq \frac{S_0^2}{6eke^{S_0}} < \frac{1}{6ek}$. The third and fourth terms give at most $\frac{-(e-1)(\lambda - 1/6k)S}{e \cdot e^{S_0}}$, which is negative. Since $r \geq 2k\lambda$, we're done.
  - If $\frac{(e-1)S_0 S r_0}{e^{S_0}} < \frac{Cr}{3}$, we have
$$\Delta L < 1 + \lambda + (C+1)k\lambda - \frac{4}{3}Ck\lambda < -\epsilon.$$
- If $r < 2k\lambda$, then by lemma 4.2.2, $\exists S_{i^*} < 6k^4\lambda$. Since $dS_i^+ > \frac{3\bar{S}_i^2 S_i}{kS^3}$, the last term is at most $-\frac{S_0}{ke^{6k^4\lambda}}$. Here we used the bound $\frac{3(e-1)\bar{S}_{i*}^2}{eS^2} > 1$.
  - If $\frac{(e-1)S_0 S r_0}{e^{S_0}} \geq \frac{Cr}{3}$: Then $r_0 S_0 \leq \frac{S_0^2}{6eke^{S_0}} < \frac{1}{6ek}$. The third and fourth terms give at most $\frac{-(e-1)(\lambda - 1/6k)S}{e \cdot e^{S_0}} \leq \frac{-(e-1)\lambda S}{2e \cdot e^{S_0}}$, which is negative. If $S_0 \geq 3C\lambda k^2 e^{6k^4\lambda}$, the last term is less than $-\frac{3}{2}Ck\lambda$, and
$$\Delta L < 1 + \lambda + (C+1)k\lambda - \frac{3}{2}Ck\lambda < -\epsilon.$$
    Else $S_0 < 3C\lambda k^2 e^{6k^4\lambda}$, so we would have $-\frac{(e-1)\lambda S}{2e \cdot e^{S_0}} < -\frac{2(e-1)}{e}Ck\lambda$ and
$$\Delta L < 1 + \lambda + (C+1)k\lambda - \frac{2(e-1)}{e}Ck\lambda < -\epsilon$$
    since $\frac{2(e-1)}{e} > \frac{5}{4}$.

– If $\frac{(e-1)S_0Sr_0}{e^{S_0}} < \frac{Cr}{3}$, we can omit the second and fourth terms which add up to $\frac{(e-1)S_0Sr_0}{e^{S_0}} - Cr < 0$. Again by the same reasoning as above, if $S_0 \geq 3C\lambda^2 e^{6k^4\lambda}$, the last term is less than $-\frac{3}{2}Ck\lambda$, and

$$\Delta L < 1 + \lambda + (C+1)k\lambda - \frac{3}{2}Ck\lambda < -\epsilon.$$

Else $S_0 < 3C\lambda k^2 e^{6k^4\lambda}$, so we would have $-\frac{(e-1)\lambda S}{e \cdot e^{S_0}} < -\frac{2(e-1)}{e}Ck\lambda$ and

$$\Delta L < 1 + \lambda + (C+1)k\lambda - \frac{2(e-1)}{e}Ck\lambda < -\epsilon$$

since $\frac{2(e-1)}{e} > \frac{5}{4}$.

□

It is clear that both Lyapunov functions that are used satisfy the properties of theorem 3.4.2. We conclude that the proposed system is positive recurrent for any value of λ.

4.2.1 m-sampling

The rule for the peers that have all but one chunk can be eased as follows. A peer in $\bar{T}_i$ samples m peers at random with replacement instead of 3. Allow a download only if each chunk other than i is observed at least twice in the sample of m peers. This would ensure none of the chunks which leave are rare. Sampling more peers increases the complexity of the system (decreases locality), but allows for a more efficient search (peers could leave earlier). One could pick m to strike a good trade-off between complexity and performance. This parameter m can be regarded as modeling the number of neighboring peers in the BitTorrent protocol.

The proof of stability generalizes to this case with little modification. In (4.1), note that $d\bar{S}_i^- \leq \min_{j \neq i} \binom{m}{2} \frac{S_j^2}{S^2}$. Therefore the last term would be bounded by $\binom{m}{2} \frac{8k^2}{3S}$, which can be bounded by 1, provided we modify the bound on S in theorem 4.2.3 with $\binom{m}{2}$. The rest of the proof goes through unaltered.

Chapter 5

Performance

From a performance point of view, some aspects of the protocol may strike the reader as inefficient. In particular, the rule for leaving the system is quite strict, and may cause substantial delay for the peers that have all but one chunk. Consider a state where most of the peers have a few or no chunks. A sample of 3 peers needs to contain at least $2k-1$ chunks (1 for the missing chunk, 2 each for the rest) for a peer to be able to leave the system. Therefore a peer with $k-1$ chunks will need to wait in the system, until the system becomes more saturated.

On the other hand, this ensures availability of all chunks to other peers, and reduces starvation in the network. This rule can be interpreted as forcing a degree of altruistic behavior and has a similar effect in terms of stability.

5.0.2 Simulations

We compare our proposed algorithm with the parameter m taking the values $\{3, 5, 10\}$, $m = 3$ being the original protocol proposed in section 4.1, with the rare chunk rule.

Figure 5.1 shows a system with 20 chunks and $\lambda = 10$. At time 0, only the seed is present. We can see all four systems reaching a stable state. The total number of peers for $m = 5$ and $m = 10$ behave roughly similar to the simple rare chunk algorithm, where

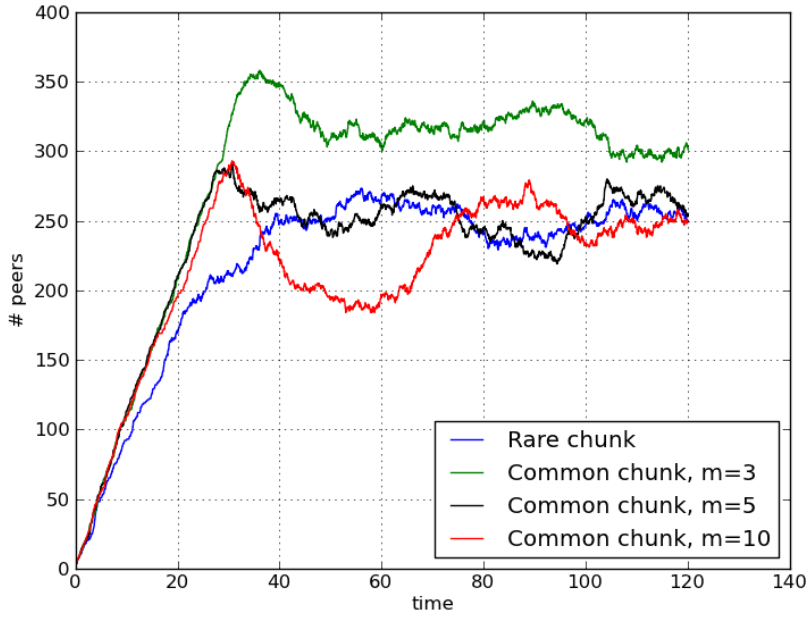

Figure 5.1. Reaching steady state from an empty system. $k = 20$. $\lambda = 10$.

$m = 3$ hovers slightly above the others due to the stricter rule keeping peers in the system for a longer time. The same behaviour is observed in figure 5.2, where all systems relax in a similar manner from an initial population of 1000 peers, all of which lack the same chunk.

As reported in [4], [9], the rare chunk rule seems to provide stability despite the lack of a conclusive proof in this direction. On the other hand, the newly proposed protocol performs competitively (and even more so with a suitably chosen parameter m) while having the advantage of a formal stability guarantee.

In figure 5.3, we compare the distribution of completion time for peers in each simulation. We run each system until it looks to have stabilized, then compare the time it takes for each new peer to leave the system. As expected, the performance of the system at stationarity improves with parameter m, with $m = 5$ having roughly equal performance with the rare chunk heuristic. Note that the common chunk protocol with $m = 3$ exhibits longer tails in the completion time distribution, due to the high variability of waiting time at the top level. Table 5.0.2 summarizes the mean and variance of the completion time of each algorithm.

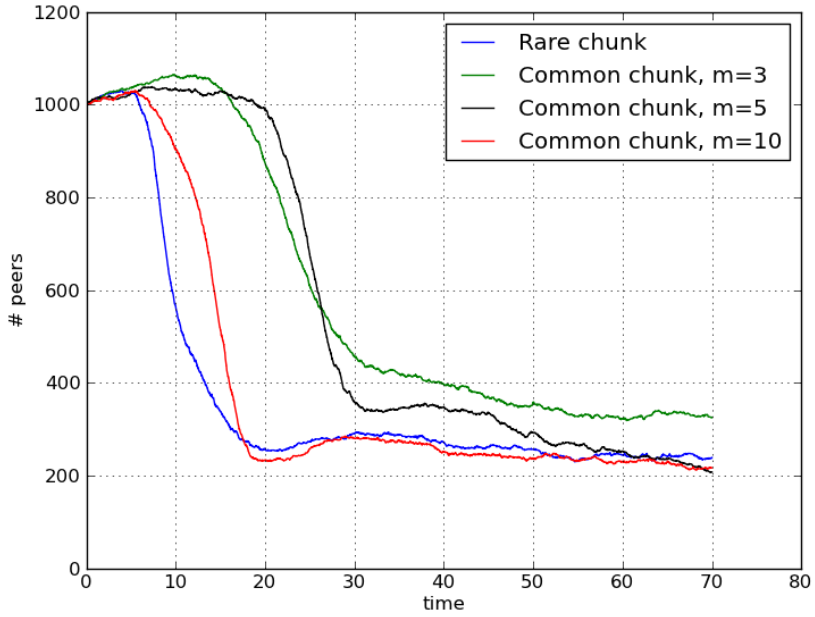

Figure 5.2. Relaxing from severe imbalance. 1000 peers all lack a single chunk at initialization. $k = 20$. $\lambda = 10$.

Protocol	E[T]	Var(T)
Rare chunk	25.6	36.4
Common chunk, m=3	30.4	114.1
Common chunk, m=5	24.9	35.2
Common chunk, m=10	22.7	25.2

Table 5.1. Mean and variance of completion time at stationarity.

Note that with 20 chunks, the theoretical minimum for the average is 20 time steps. The data is collected from around 3000 peers that completed the file during the simulation.

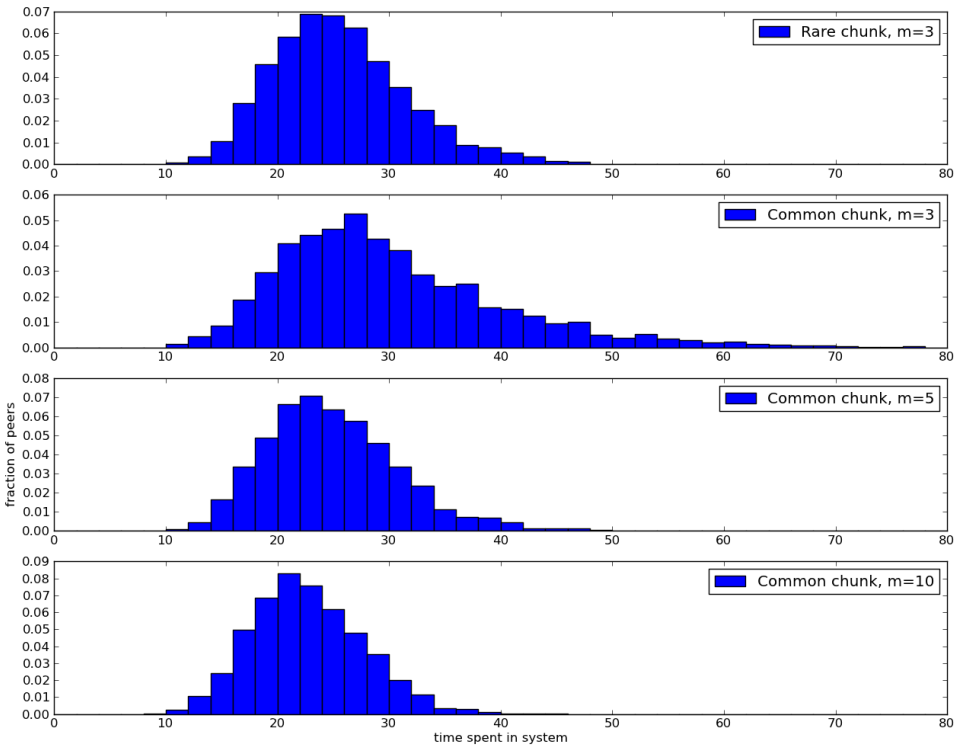

Figure 5.3. Empirical distribution of completion time at stationarity.

Chapter 6

Conclusion

Peer-to-peer schemes such as *BitTorrent* have been remarkably successful in revolutionizing the way files are spread in a network. Still, it is desirable to completely decentralize such protocols to avoid the pitfall of a single tracker. Naive attempts at such schemes have been plagued with the 'rare chunk' syndrome, which causes instability. While it has recently been shown that relatively minor altruistic behavior or a powerful seed can stabilize such systems, these properties are usually a luxury in real world networks.

In this paper we have demonstrated that a completely decentralized, stable peer-to-peer network is possible, even with completely non-altruistic peers and a single seed with minimal upload capacity. While earlier work has hinted at this result with heuristics and simulations, it had proved difficult to come up with a provably stable scheme. Although our original algorithm has drawbacks in terms of performance, we have suggested an improvement that allows trading locality for performance. Our proof was easily adapted to this case, which suggests that the methods presented here might allow for stability guarantees for other algorithms.

References

[1] F. Mathieu and J. Reynier, "Missing piece issue and upload strategies in flashcrowds and p2p-assisted filesharing," in *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on.* IEEE, 2006, pp. 112–112.

[2] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.

[3] I. Norros, B. Prabhu, and H. Reittu, "Flash crowd in a file sharing system based on random encounters," in *Proceedings from the 2006 workshop on Interdisciplinary systems approach in performance evaluation and design of computer & communications sytems.* ACM, 2006, p. 4.

[4] I. Norros, H. Reittu, and T. Eirola, "On the stability of two-chunk file-sharing systems," *Queueing Systems*, vol. 3, pp. 183–206, 2011.

[5] I. Norros, B. Prabhu, and H. Reittu, "On uncoordinated file distribution with non-altruistic downloaders," *Managing Traffic Performance in Converged Networks*, pp. 606–617, 2007.

[6] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4. ACM, 2004, pp. 367–378.

[7] J. Zhu and B. Hajek, "Stability of peer to peer systems," in *SIGACT-SIGOPS Symposium on Principles of Distributed Computing.* ACM, 2011, pp. 321–330.

[8] L. Massoulié and M. Vojnovićscho, "Coupon replication systems," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 2–13.

[9] H. Reittu, "A stable random-contact algorithm for peer-to-peer file sharing," *Self-Organizing Systems*, pp. 185–192, 2009.

[10] I. Norros and H. Reittu, "Urn models and peer-to-peer file sharing," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on.* IEEE, pp. 554–554.