No more crash or crunch: sustainable credit dynamics in a P2P community

R. Rahman, D. Hales, T. Vinkó, J. Pouwelse, H. Sips
Delft University of Technology, Delft, The Netherlands
email: rrameez@gmail.com,
Tel.: (+31) 15 2784475, Fax: (+31) 15 2786632

***Abstract*—Many peer-to-peer file sharing communities implement credit policies to incentivise users to contribute upload resources. Such policies implicitly assume a user model - how the user controlling each peer behaves. We show using an agent-based model that credit policies, based on bandwidth contribution, and a *selfish* user model, can lead to both "crunches" and "crashes" where the system seizes completely due to too little credit or too much credit. We explore the conditions that lead to these system pathologies and present a theoretical analysis that allows us to determine if a community is sustainable or will eventually crunch or crash. Finally we apply the analysis to produce a novel adaptive credit system that automatically adjusts credit policies to maintain sustainability.**

***Keywords*—P2P, economics, agent-based simulation, credits, incentives.**

I. Introduction

BitTorrent (BT) is a widely used peer-to-peer (P2P) protocol for distributing files over the Internet [2]. BT uses a swarm based approach in which peers interested in a particular file cooperate by trading small pieces of the file. By contributing upload bandwidth to others, peers collectively distribute the file without the need for high capacity central servers. In BT peers are incentivsed to contribute upload bandwidth via a direct reciprocity approach - a form of tit-for-tat (TFT).

The TFT approach, put simply, involves peers only contributing upload to other peers who reciprocate. Hence freeriding, where a peer only downloads without uploading, is punished.

However, the TFT approach does not incentivise a crucial BT activity called "seeding". A seeding peer stores the entire file and hence acts in a purely altruistic way by giving away pieces. To create a new BT swarm at least one seeder is required. When a peer has downloaded the entire file it automatically becomes a seeder unless the user of the peer decides to leave the swarm. A swarm containing many seeders provides high download rates for peers downloading from the swarm (termed "leechers"). Yet since seeding is not incentivised many swarms suffer from so-called "Hit and Run" (H&R) user behaviour where peers leave the swarm after downloading the file.

In order to address these, and other, issues private BT file-sharing communities have recently emerged. In many such communities the upload and download behaviour is recorded centrally, over time, across a population of swarms that are restricted to the community. Many of these communities apply policies to incentivise good overall upload / download behaviour - such as ratio enforcement. For example, a simple policy would be to expect all peers to have some minimum ratio of upload to download at all times. Other policy variants include requiring some minimum level of absolute credit (upload - download) over time, or detecting and punishing H&R behaviour.

In previous work [4] we showed, via an agent-based model, that, even with altruistic users, in which all peers seeded as much as possible, private community credit systems could, counter-inuitively, lead to poor performance due to a "credit crunch" or squeeze in which a few peers accumulated much of the credit in the system depriving others and hence decreasing overall system throughput. This meant that adding altruistic capacity to the system, in the form of high capacity peers who were willing to upload without reciprocation, could actually reduce overall performance (or throughput) - meaning the total amount of data exchanged in the system[1]. It should be noted that private BitTorrent communities regularly employ policies such as 'seeding bonus', 'free leech', rewarding seeding time instead of bandwidth, among other schemes. The fact that various private BitTorrent communities have to employ such policies is indicative of the fact that they are grappling with performances loss due to credit mobility issues.

In this paper we explore credit dynamics with two user model variants: *selfish*, where peers only contribute what is necessary to allow them to continue to download content, and *hoarders*, where peers desire to accumulate more credit than is necessary for their immediate downloading needs.

We find for populations of selfish peers, when too much credit is distributed too evenly, that this leads to a crash in which peers are not incentivised to contribute and hence the system seizes to zero throughput containing only leechers. *We define a crash as a situation in which due to credit abundance the system completely seizes up, providing no upload or download to any peers*. Conversely too little credit distributed over the peers leads to a crunch in which peers do not have enough credit to download leading to a seized system containing only seeders. *We define a crunch as a situation in which due to credit shortages the system completely seizes providing no upload or download to any peers*. We also

[1]This can be compared to Braess Paradox in which adding capacity to transport networks may reduce total flow under the assumption of rational actors – see: http://en.wikipedia.org/wiki/Braess's_paradox

observe that a population containing any number of hoarders will lead to a crunch eventually as credit is monopolised by them.

Specifically, we make the following contributions:

1) we demonstrate using simulations that credit crunches and crashes occur, and identify the conditions that lead to these extreme outcomes;
2) we present a theoretical analysis and produce a set of propositions that define if a system will crash, crunch, or be sustainable over a defined time horizon;
3) we evaluate a novel adaptive credit policy informed by our results and analysis. We demonstrate that the adaptive policy can avoid both crashes and crunches under extreme and changing conditions.

II. Model description

In order to explore the conditions under which credit crashes and crunches occur we designed an agent-based simulation model containing the essential properties of private tracker credit systems.

We abstracted away the particularites of real communities, so that the underlying credit dynamics become clear and can be analysed.

Our simulation model is based on *c*ycles. Each cycle represents a unit of time in which each peer is activated and may perform some activity - such as uploading or downloading data from other peers and initiating new seeding or leeching sessions. Peers accumulate and spend credit by participating in swarms which are supported by a Tracker. The Tracker keeps a record of all current swarm members and records the upload and download amounts against each peer. We describe the tracker, swarm and peer entities in more detail in the following subsections below.

A. Tracker

The tracker supports a set of swarms that are available to the community and stores the upload and download amounts reported by each peer over time. The tracker implements a ratio enforcement policy in which peers with upload / download ratio less than one are stopped from downloading content until they increase their ratio, through seeding content in a swarm. In order to allow peers to begin downloading initially they are awarded initial credit equal to one file size. It should be noted that the tracker in our model, like in private BitTorrent communities, is a centralized component.

B. Peers

The community is represented by a set of peers. Each peer has the same fixed upload capacity, representing units of data per time unit. Download capacity is assumed to be infinite - the assumption being that upload is the bottleneck in most file sharing communities. We make a further simplifying assumption that peers have a maximum of one seeding or one leeching session active at any time.

We implement two user types: selfish peers and hoarders. User types are characterized by their leeching and seeding

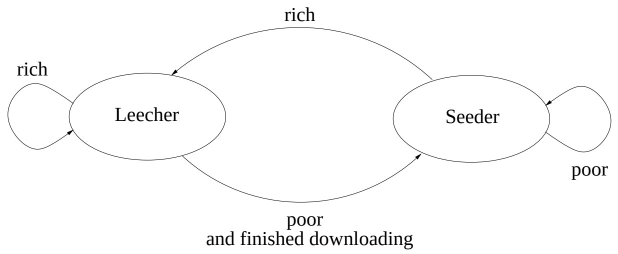


Fig. 1: A state transition diagram indicating how peers move between seeding and leeching sessions.

behaviour. Selfish peers only seed (to earn credit) if their current credit balance is less than one file size (C) otherwise they only leech. This captures the notion that a selfish peer only wants enough credit to download the next file. Hoarder peers behave in a similar way but have a higher threshold before they stop seeding.

More formally, we define two functions $u(t)$ and $d(t)$, which are the total number of units a peer has uploaded and the total number of units a peer has downloaded at time t. The credit of a peer at time t is $K(t) := u(t) - d(t)$, while the ratio is $R(t) = u(d)/d(t)$. Peers can always be considered in one of two states: “rich” or “poor”. Peers are considered to be “poor” except under the following conditions:

- A selfish peer is “rich” when $K(t) \geq C$.
- A hoarder peer is “rich” when $R(t) \geq 2$.

When a leeching peer finishes downloading a file it decides whether to seed that file or to select another file to leech (i.e. to download) from the set of swarms available. If a peer is rich it selects a new swarm with uniform probability and begins to leech without seeding the previously downloaded file. If a peer is poor then it seeds the downloaded file until it becomes rich and only then stops seeding and starts leeching in another swarm. Figure 1 shows a state transition diagram indicating how peers move between leeching and seeding states.

Our user model represents an abstracted form of behaviour compared to what is observed in real file sharing communities. Figure 2 shows the cumulative distribution of peer ratio over a four month period from a real community. It can be observed that the majority of peers are within one order of magnitude of ratio $R(t) = 1$ where t is the end of the period. Notice here that we also observe a small proportion of peers with very high ratios where $R(t) > 10$, indicating some extreme hoarding behaviour.

C. Swarms

The community comprises a set of swarms S. Each swarm may contain any number of seeders and leechers[2]. At any time each peer will be either a seeder or a leecher in a single swarm. In one time unit each swarm distributes upload units (pieces) between seeders and leechers and between leechers

[2]We ensured that all swarms have at least one seeder - if this is possible - by randomly redistributing a seeder to any swarm that becomes seederless.

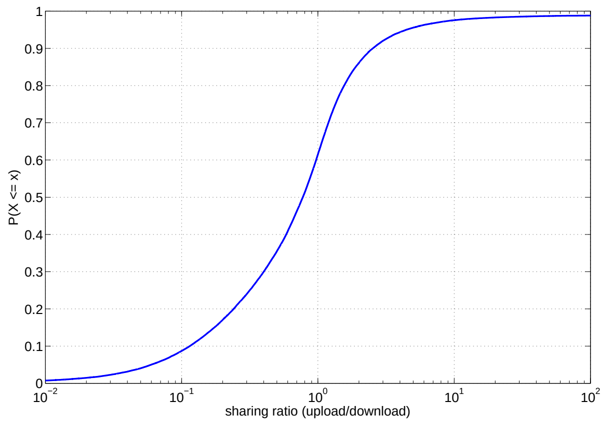


Fig. 2: A CDF of peer ratios collected from a private file sharing community.

and leechers. The latter distribution of upload captures the effect of the tit-for-tat (TFT) mechanism in BitTorrent. We do not assume any freeriding or differential upload capacity at the piece level. All peers utilise their full upload to contribute to others and all have equal upload capacities.

In more detail, we model the distribution of upload in a swarm in the following way. In our model a file is divided into units (or pieces in BitTorrent terminology). Similar to many BitTorrent implementations, each peer has four upload slots from which it can upload data to four different leeching peers per time cycle. At every time cycle, each peer in a swarm is fired in random order. When a peer is activated it chooses four leeching peers at random to upload data to. In line with the *rarest piece first* [6] selection algorithm in BitTorrent, peers upload the rarest pieces to other peers first. We define rarest pieces to be the least replicated pieces among all the peers in a swarm. Hence all pieces are ordered by rarity and each of the four receiving peers is given the rarest piece that it does not currently posses. In the case that a receiving peer already has all the pieces available in the sending peer then another peer is selected randomly to receive a piece. In this way each peer will send one piece to four other randomly selected peers in the swarm if this is possible.

It is important to note that our aim is to produce a simple model that captures the main characteristics of BitTorrent piece sharing under constant and ideal conditions in order to investigate the importance of credit dynamics. In the real world the actual interactions between peers would be highly influenced by differential bandwidths, different clients and the local nature of the piece rareness determination. Our aim here is not to produce simulations that align with measurements from target systems but to abstract the important mechanisms with respect to credit dynamics.

As stated previously, the tracker records all upload and download against each peer and hence keeps a running total of credit and ratio for each peer.

TABLE I: Results for selfish peers with constant credit

prop.of **rich** at start	avg. **throughput** (std.dev)	avg. prop. of **seeders** (std.dev)	final **state**
0.1	0.0003 (0.0000)	1.0000 (0.0000)	crunch
0.3	0.2183 (0.0014)	0.9525 (0.0014)	sustain
0.5	0.7769 (0.0023)	0.7685 (0.0023)	sustain
0.7	0.9684 (0.0036)	0.5064 (0.0036)	sustain
0.8	0.5867 (0.4780)	0.2485 (0.4780)	sustain/crash
0.9	0.0008 (0.0000)	0.0000 (0.0000)	crash

III. Simulation Results - Constant Credit

We performed a number of simulation experiments to explore the conditions under which a sustainable file sharing community is viable under the assumption of a fixed credit amount in the population. We used the following parameters: number of peers $N = 1000$, number of swarms $s = 100$, file size $C = 10$ units, peer upload capacity $U = 4$ units. The small file size means the simulation runs produce results at a large scale of granularity. We also performed runs with $C = 100$ and found no significant difference in results. In general our results are independent of the size of C. As stated previously we assume no limit on download capacity. For each experiment we performed 10 independent runs with different psuedo-random number seeds. Each run was executed to 2000 cycles.

A. Populations of selfish peers

In order to explore the results of different initial credit levels on the performance of populations containing all selfish peers we ran several simulation experiments varying a single parameter - the initial proportion of peers who are given enough credit to be "rich" (i.e. given initial credit of C).

Results can be seen in Table I. All values are averages of 10 runs. The columns have the following meanings: $rich$ shows the proportion of peers that are initially awarded C credit; $throughput$ gives the throughput of the system in cumulative units of data exchanged over the entire run (normalised); $seeders$ shows the proportion of peers in the population at the end of the run that are seeding; $state$ indicates the state of the system: $crunch$ means the system seized due to lack of credit and $crash$ indicates seizure due to too much credit. *Sustain* means the system finds a stable sustainable throughput avoiding both crashes and crunches. We ran extended runs up to 20,000 cycles and found the sustainable outcomes were maintained.

When the number of rich peers is initialised to 30%, 50% and 70% we see sustainable outcomes with increasing throughput and reduced number of seeders. This is intuitive since as the amount of credit increases in the system less peers are poor and hence more exchange of data can occur.

In the crunch state, where only 10% of peers are initialised as rich, notice that the system is composed of all seeders by the end of the run and hence no exchange of data can occur. Conversely, in the crash state, where 90% of peers are initialised as rich, all peers are leechers by the end of the run, again, meaning no exchange of data is possible. Inspection of individual runs evidences that crunches and crashes happen

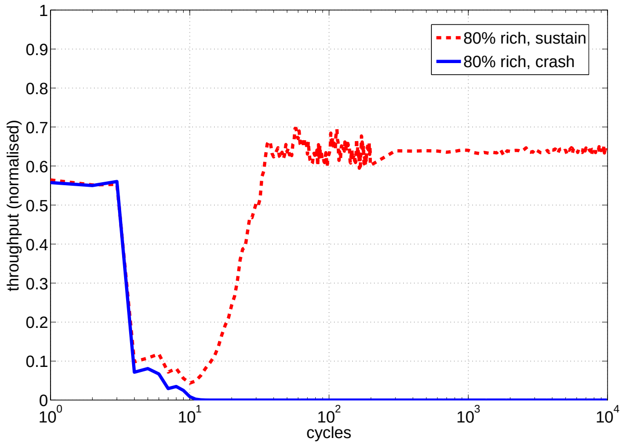


Fig. 3: Two alternative trajectories when initial number of rich peers is set to 80%. Notice both sustain and crash outcomes are possible.

quickly - within the first ten cycles or so. This is reflected in the low (almost zero) cumulative throughput values under crash and crunch states.

It is interesting to consider the results when the initial number of rich peers is set to 80%. As can be seen this produces both sustain and crash outcomes reflected in the high variance of throughput. We are not sure why exactly this happens. But what's obvious is that here we are very close to the threshold leading to a crash and we find path dependency based on initial random conditions leading to either a high sustainable throughput, in a third of runs, or a sudden crash otherwise. Figure 3 shows the two alternative trajectories. Such properties are common in complex systems where small differences in initial chance conditions can lead to radically different outcomes.

B. *Populations containing hoarder peers*

When we introduced *any* number of hoarder peers into our system it eventually led to a crunch - this is true even for a single hoarder in the population. The speed of the crunch depended on the number of hoarders. This is intuitive since hoarders seed (to earn credit) until they have a ratio $R(t) \geq 2$. This means that as the simulation progresses the hoarders eventually hold all the credit in the system and a crunch is inevitable.

C. *Discussion*

The results of our initial experiments indicate that a community will seize if all the swarms seize. A seized swarm is one that will not allow any peer within it to leave. A peer can leave either by downloading the entire file and then moving to another swarm (if the peer finishes the download and remains rich) or by seeding to earn enough credit to become rich so that it can move to another swarm for leeching. In the crunch condition there will not be enough leechers to satisfy seeders and in the crash condition there will not be enough seeders to satisfy leechers. In these cases the swarm becomes

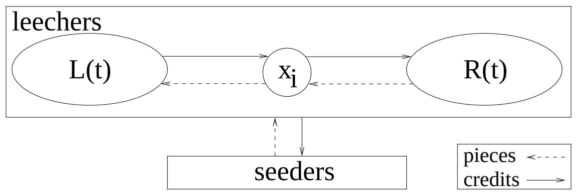


Fig. 4: Diagram showing the relationship between the L and R sets in relation to a given peer x.

a kind of trap or sink that, unless some new peers with certain characteristics enter the swarm in the future, stops the peers from further exchange in other swarms.

Given the state of a swarm, including all peer credit levels at some time step, it is possible to define if the swarm *will* seize in the future due to a crash or crunch condition - unless something changes. In the next section we present a theoretical analysis which specifies these conditions. Following this we validate the model by checking that these conditions apply to crashes and crunches. We then apply the conditions to test a novel mechanism for automatically adapting credit policies when the system is identified as heading for crash or crunch conditions in order to avoid them.

IV. Theoretical Results

We wish to determine for a given set of swarms if the system will crunch, crash or be sustainable. We assume a peer can either leech or seed, exclusively, and only exist in one swarm at one time. Firstly we define some terms and definitions and then present three propositions which give conditions for each of the three outcomes.

The following notation is used:

- S_ℓ is the swarm ℓ, where $\ell = 1, \ldots, s$, i.e. the number of swarms is s.
- the number of leechers and seeders in swarm ℓ is $x^\ell(t)$ and $y^\ell(t)$, respectively, at time t,
- x_i^ℓ is the leecher i and y_j^ℓ is the seeder j in swarm ℓ,
- $c_{x_i^\ell}(t)$ is the credit of leecher i and $c_{y_j^\ell}(t)$ is the credit of seeder j in swarm ℓ at time t.
- $p_{x_i^\ell}(t)$ is the proportion of the file that leecher x_i^ℓ has at time t,
- C is the amount of credit required to download a file (i.e. the file size).
- u is the upload bandwidth.

Moreover, define $L_i^\ell(t)$ as the set of leechers which have less pieces of the given file, represented by swarm ℓ, than leecher x_i^ℓ, and $R_i^\ell(t)$ is the set of leechers which have more pieces of the given file, represented by swarm ℓ, than peer x_i^ℓ at time t. Formally, $L_i^\ell(t) := \{j \ : \ p_{x_j^\ell}(t) < p_{x_i^\ell}(t)\}$ and $R_i^\ell(t) := \{j \ : \ p_{x_j^\ell}(t) > p_{x_i^\ell}(t)\}$. Figure 4 shows a diagram of these sets. It can be seen from the figure that we make the simplifying assumption that peers can only download pieces from those peers who have more pieces. Therefore, a peer can only give credit to those peers who have more pieces. It should be noted that in reality, this is not the case. If two peers have downloaded complementary parts of a file, they can exchange pieces between themselves no matter who has more pieces. However, the results in Section V demonstrate that this assumption doesn't impede upon the predictive powers of our subsequent analysis.

We also need to define the amount of credit that any given leecher can earn from other leechers:

$$q_{x_i^\ell}(t) := \sum_{j \in L_i^\ell(t)} (p_{x_i^\ell}(t) - p_{x_j^\ell}(t))$$

Given the above we can now define a set $X_\ell(t)$ containing those leechers with enough existing credit, plus credit earning potential, at time t, to download the entire file associated with swarm ℓ and still be in a rich state. As previously defined a peer is considered rich if it has credit of at least C (i.e. one file size). Leecher x_i^ℓ will be able to download a file and still remain rich if $c_{x_i^\ell}(t) - (1 - p_{x_i^\ell}(t))C \geq C$ holds. During the download x_i^ℓ is not only paying but also earning some credits from other leechers in the swarm ℓ, from those who have less pieces than x_i^ℓ. From these peers x_i^ℓ can earn $q_{x_i^\ell(t)} \cdot C$ amount. On the other hand, x_i^ℓ cannot earn credits from those who are seeding and who have more pieces. The number of those peers are $y^\ell(t) + |R_i^\ell(t)|$. Thus, from the other leechers, x_i can earn $q_{x_i^\ell}(t)/(y^\ell(t) + |R_i^\ell(t)|) \cdot C$ amount of credit. This leads to the definition

$$X_\ell(t) := \{x_i^\ell : c_{x_i^\ell}(t) + p_i^\ell(t)C + \frac{q_{x_j^\ell}(t)}{y^\ell(t) + |R_i^\ell(t)|} C \geq 2C\}.$$

Note that when a leecher finishes its download and becomes poor then it stays in the same swarm to seed. In this case X_ℓ remains the same, because $y + R_i$ does not change. According to our assumption no new seeders can join to this swarm from other swarms. X_ℓ can be changed when new leechers join the swarm as they can give the other leechers the chance to become rich after their download (X_ℓ could increase then), or when seeders are leaving the swarm (X_ℓ could decrease).

Similarly, we define the set $Y_\ell(t)$ for seeders in the swarm ℓ containing those seeders which have the credit earning potential, at time t to become rich. A seeder earns credit from all the leechers. One leecher x_i^ℓ gives $(1 - p_{x_i^\ell}(t))C$ amount of credit to all the seeders and to those other leechers who have more pieces than x_i^ℓ. Thus we define

$$Y_\ell(t) := \{y_j^\ell : c_{y_j^\ell}(t) + \sum_{k=1}^{x^\ell(t)} \frac{1 - p_k^\ell(t)}{y^\ell(t) + |R_k^\ell(t)|} C \geq C\}.$$

Note that, similarly to X_ℓ, the set Y_ℓ can be changed in time only when new leechers are joining (Y_ℓ could increase) or when a seeder is leaving the swarm (Y_ℓ could decrease then).

We now introduce a temporal horizon by defining a function for download time for leechers and required seeding time for seeders.

The expected download time $T_{x_i^\ell}$ for the leecher x_i in the swarm ℓ depends on the remaining amount to be downloaded (which is $(1 - p_{x_i^\ell}(t))C$) and on the number of seeders and leechers in the swarm. The seeders can give pieces to all the leechers with the rate of $y^\ell(t)/x^\ell(t) \cdot u$. Moreover, every leecher x_k, which has more pieces than x_i, i.e. the peers in R_i can give pieces to those who have less pieces than x_k, which is the set L_k. The rate of this is $\sum_{k \in R_i} 1/|L_k| \cdot u$. Combining these observations we get:

$$T_{x_i^\ell}(t) := t + \frac{(1 - p_{x_i^\ell(t)}) \cdot C}{(\frac{y^\ell(t)}{x^\ell(t)} + \sum_{k \in R_i^\ell(t)} \frac{1}{|L_k^\ell(t)|}) \cdot u}.$$

Note that $T_{x_i^\ell}$ is only the expected time given current swarm composition. The actual time could be longer if new leechers joined or seeders left the swarm. We also need the expected time $T_{y_j^\ell}$ for a seeder y_j in swarm ℓ to become rich. A seeder, at time t, needs to earn $C - c_{y_j}^\ell(t)$ credit. The rate at which this can be obtained depends on the total upload capacity of all leechers less the leecher to leecher (TFT) interactions. This leads to the formula:

$$T_{y_j^\ell}(t) = t + \frac{C - c_{y_j^\ell}(t)}{\sum_{i=1}^{x^\ell(t)} \frac{1}{y^\ell(t) + |R_i^\ell(t)|} u}.$$

Note that $T_{y_j^\ell}$ is also only the expected time given the situation in the swarm at a time instance.

Proposition 1 - crunch. If $X_\ell(t')$ and $Y_\ell(t')$ are both empty for all $\ell = 1, \ldots, s$, then the system will crunch (i.e. the throughput becomes zero) at time

$$t' + \max_{\ell=1,\ldots,s} \max_{i=1,\ldots,x^\ell(t')} T_{x_i^\ell(t')}.$$

Proof. For the sake of simplicity we omit t' and ℓ from the formulas. If X and Y are both empty for all the swarms, then it means that there are no leechers and no seeders who become rich. Thus, there will be no exchange of credit in the whole system. This happens after the very last leecher finishes its download, which is the maximum of all the maximum of download times per swarms.

Proposition 2 - crash. If $|Y_\ell(t')| = y^\ell(t')$ and

$$\min_{k \in P_\ell(t')} T_{x_k^\ell}(t') > \max_{j=1,\ldots,y^\ell(t')} T_{y_j^\ell}(t')$$

for all the swarms, where $P_\ell(t) := \{i : x_i^\ell \notin X_\ell(t)\}$, then the system will crash (i.e. the throughput becomes zero) at time

$$\max_{\ell=1,\ldots,s} \max_{j=1,\ldots,y^\ell(t')} T_{y_j^\ell}(t').$$

Proof. The system crashes if there are no seeders. The condition $|Y| = y$ indicates that all the seeders in the whole system will be rich, thus they will become leechers. The set P contains those leechers which will not be rich after finishing their download. Note that this set can be empty for some swarms. Peers from P would stay to seed in their swarm if they finished their download. However, they cannot finish downloading the file if there are no seeders left in the swarm.

TABLE II: Results for selfish peers with adaptive credit

prop.of **rich** at start	avg. **throughput** (std.dev)	avg. prop. of **seeders** (std.dev)	final **state**
0.1	0.1922 (0.0221)	0.9556 (0.0221)	sustain
0.3	0.2220 (0.0100)	0.9544 (0.0100)	sustain
0.5	0.7770 (0.0023)	0.7761 (0.0023)	sustain
0.7	0.9674 (0.0024)	0.4970 (0.0024)	sustain
0.8	0.9742 (0.0163)	0.5300 (0.0163)	sustain
0.9	0.8797 (0.0689)	0.7266 (0.0689)	sustain

Proposition 3 - sustainability. If the set

$$\mathcal{U}(t') := \{\ell \ : \ |X_\ell(t')| > 0 \text{ and } |Y_\ell(t')| < y^\ell(t')\}$$

is not empty then the system is sustainable until

$$t'' = t' + \min\{M_\mathcal{U}, M_\mathcal{V}\},$$

where

$$\begin{aligned} M_\mathcal{U} &:= \min_{\ell \in U}\{\min_{x \in X_\ell(t')} T_x, \min_{y \in Y_\ell(t')} T_y\}, \\ \mathcal{V} &:= \{\ell \ : \ |X_\ell(t')| > 0 \text{ or } |Y_\ell(t')| < y^\ell(t')\} \quad \text{and} \\ M_\mathcal{V} &:= \min_{\ell \in V}\{\min_{x \in X_\ell(t')} T_x, \min_{y \in Y_\ell(t')} T_y\}. \end{aligned}$$

Proof. The set $\mathcal{U}$ contains all swarms in which there are some leechers which will be rich after their download and seeders which will not be leaving (i.e. not be rich), based on the current situations at time t'. Those seeders who are characterised, as of time t', as not leaving the swarm could leave if new leechers joined the swarm. This would happen when a leecher or seeder from another swarm leaves and joins this swarm. Thus, giving the time horizon for sustainability we need to know the earliest time when the current situation could change. For this, we defined $M_\mathcal{U}$, which is the minimum time of the first leecher leaving any swarm in $\mathcal{U}$ and the first seeder becoming rich from any swarm in $\mathcal{U}$. Moreover, $M_\mathcal{V}$ gives the same for the swarms in $\mathcal{V}$. Finally, we need to take the minimum of $M_\mathcal{U}$ and $M_\mathcal{V}$ in order to get the prediction period for the system's sustainability.

V. Simulation Results - Adaptive Credit

Based on our experimental and theoretical results we designed a novel *proactive credit intervention mechanism* to avoid crashes and crunches. At each cycle we examine the system and compare it against the crash and crunch propositions derived from the theoretical analysis. Hence we can obtain an early warning for potential crunch or crash outcomes.

When the system is determined to be entering a crunch the system applies a new credit policy we term "freeleech" policy. This means leeching peers in the swarms do not pay any credit (no download is recorded) but seeders and other uploadeds are still credited with upload. Hence leechers can download for free and new credit is injected into the system.

When the system is determined to be entering a crash it applies a "freeseed" policy. This means seeding peers (and uploading leechers) in the swarm do not receive any credit (no upload is recorded) but leechers still pay credit to download. Hence seeders (and uploading leechers) upload for free and leechers pay credit that is removed from the system.

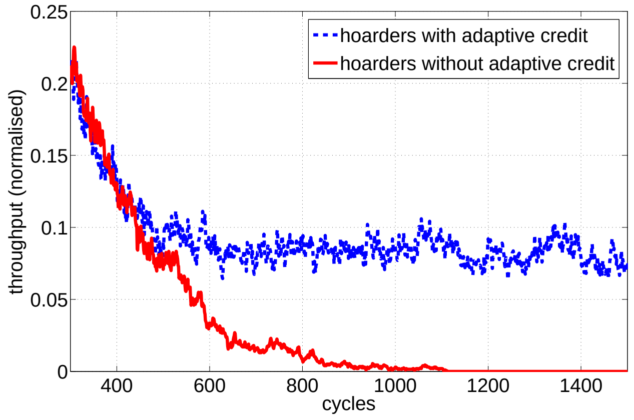


Fig. 6: Two typical runs from a system with and without the adaptive credit system containing initial 50% rich peers and 1% hoarders. Notice that without adaptive credit the system quickly crunches.

When the system is determined to be in a sustainable state then the regular credit policy is applied, that is, all upload and download is recorded as normal.

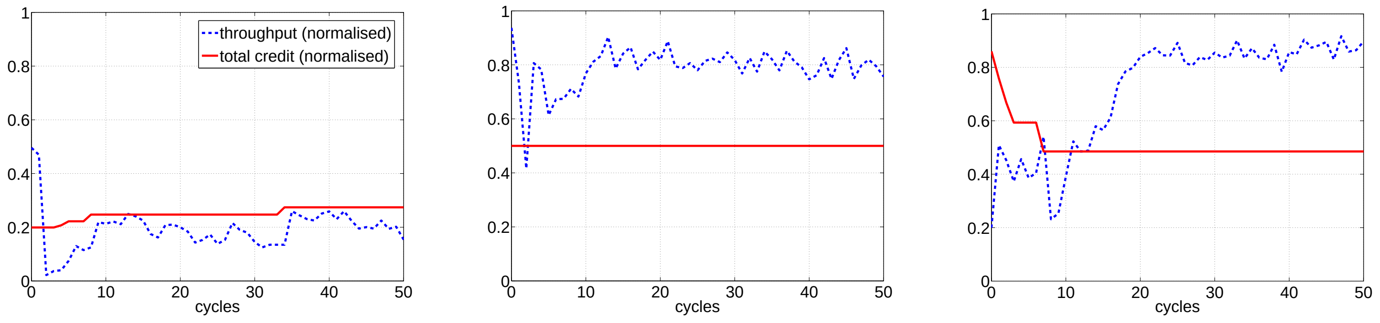
If one views uploaders as *producers* and downloaders as *consumers* then freeleech is rather like a 100% rebate for the consumer for any purchase and freeseed is like a 100% tax on the producer.

A. Populations of selfish peers

Table II shows the results of performing simulation runs with the same parameters as were used for the runs given in Table I but with the adaptive credit mechanism turned on. Notice that all runs produce a sustainable outcome including those runs (10% rich and 90% rich) which previously led to crunches and crashes. This indicates that the propositions have given early enough warning for the adaptive credit policy to avoid the crash and crunch conditions. If the system ever enters a crash or crunch condition then it would seize completely and the adaptive credit policies could not recover it. Hence these results are a form of experimental validation of the previously derived propositions. Figure 5 shows three typical runs for different initial amounts of credit. A crunch is avoided in (a) via the activation of freeleech at several cycles - note the increase in credit over time. A crash is avoided in (c) via the activation of freeseed within the initial cycles - note the decreasing credit over time.

B. Populations containing hoarder peers

In order to test if the adaptive credit system can deal with more extreme conditions we ran simulations in which a small subset (1%) of the population are set as hoarders. As stated previously any number of hoarders in a population will eventually lead to a crunch. This is because hoarders store-up increasing amounts of credit and eventually deprive all other peers of credit. Figure 6 shows two typical runs with and without the adaptive credit system. As can be seen, without



(a) Initial 10% rich peers (b) Initial 50% rich peers (c) Initial 90% rich peers

Fig. 5: Three typical runs for different initial numbers of rich peers with the adaptive credit mechanism turned on. Note how in the extreme cases for (a) and (c) the amount of credit in the system is dynamically changed preventing a crunch or crash respectively.

adaptive credit the system eventually crunches, whereas the adaptive credit system keeps the system sustainable.

However, notice that the throughput of the system is very low because the adaptive credit system does not attempt to optimise the system but rather only to avoid a crunch condition. Hence the system injects new credit each time a crunch is predicted. This additional credit is eventually collected by the hoarders and the process repeats.

We found that if we increase the number of hoarders to 5% of the population then the adaptive credit system could not prevent a crunch occurring. This is due to the user model assumptions we made in our analysis. That is, we assumed that peers will behave in the selfish way described in section II-B.

C. Discussion

The aim of the adaptive credit system was to avoid crunch and crash states. This was achieved but comes at a potential cost. The reason for using credit systems within private communities is to provide incentives for peers seeding content. The freeleech and freeseed policies temporarily suspend these incentives. It could be argued that this could lead to reduced performance if users learned to game the system by only downloading during freeleech periods and not seeding during freeseed periods. However, as we have seen, the credit interventions only occur for short periods in our runs. This potentially means that it would be impractical for a user to notice and take advantage of such periods. An additional refinement, that could help preserve incentives even during freeseed and freeleech periods, would be to parameterise the freeseed and freeleech "tax" amount. This would mean that rather than always taking 100% of any leecher or seeder credit, other values such as 50% could be used. Any value less than 100% would still provide incentives for good behaviour. Furthermore, the taxation amount could be variable, and could be applied in a continuous fashion, rather than getting triggered at the extreme conditions of crash and crunch.

As was observed in the situation where hoarders were introduced into the population, the adaptive policy depends on the assumptions of the selfish user model since this formed the basis of our theoretical analysis. It was interesting to note that the system *could* cope with a small proportion of hoarders (following a different behaviour from the selfish peers) but it is an open question as to how well it would cope with other small numbers of behavioural variants. We discuss this issue further in the conclusion section below.

VI. Related Work

To the best of our knowledge, there hasn't been much work done on studying crunches, crashes and sustainability in P2P systems with credit based incentive schemes. In previous work, we introduced the concept of a credit crunch with the aid of a simplified model of a private tracker. We considered a very simple user model of all altruists and only concentrated on "limited" crunches in which insufficient initial credit in the system led to decreased (and not zero) throughput [4]. The present paper builds upon that work and introduces a more nuanced user model, presents a theoretical analysis and studies crunches and crashes that seize the entire system.

Many P2P incentive schemes based on credits have been proposed in the literature such as [3], [10], [12]. These schemes usually build upon three components: 1) A virtual currency, 2) Micropayments and 3) An accounting structure. In a P2P setting there are issues in maintaining this structure. For the accounting structure, such schemes usually have to rely on trusted accounting centers or third parties. Although, the payments can occur in a decentralized way as proposed in [1]. Sirivianos et al present monetary exchanges facilitated by a centralized bank [11]. Great emphasis is laid on creating a non manipulable scheme of exchanges using cryptographic techniques. The presence of a centralized bank means that the scheme is not scalable but has greater security than a completely decentralized solution.

Vishnumurthy et al. present a system involving virtual currency where sets of bank nodes keep the transaction balance of peers [13]. Karma is defined as the value which captures the amount of resources that a peer has contributed and consumed. This represents the users standing in the global system. Importantly, the level of karma (or credit) in the system is maintained and measures are taken to avoid inflation and deflation that can occur when peers leave the system. In this way [13] is an important contribution because the work begins to realize the problems that are inherent in dealing with credit systems. In avoiding inflation and deflation, their only aim is to maintain the per-capita karma i.e. the total karma divided by the number of active users.

Kash et al. [5] show that in a scrip system, where agents can consume and produce services, both an overabundance of money supply and its shortage lead to inefficiency. Surplus credit can lead to a monetary crash where freeriding is encouraged. At the other end of the spectrum, a shortage in the money supply leads to agents not having enough money and not being able to afford services in the system. They also consider hoarders and how to optimise the credit supply. Our work is different in that we focus not on a generic service exchange scenario but a filesharing scenario inspired by BitTorrent private communities. Also we apply a selfish user model[3] rather than a utility optimising one. In addition we focus on detecting and avoiding extreme crashes and crunches, where the entire system seizes, rather than optimising the system.

Currently deployed credit systems can be easily gamed since they rely on self-reporting of upload and download behaviour by peers – currently such behaviour is policed by human administrators. Additionally the BitTorrent protocol itself can be gamed through strategic clients that act selfishly [7], [8], [9].

VII. Conclusions

We have presented findings from an agent-based model of a private BitTorrent file sharing community which uses credits to incentivise uploading behaviour. We have examined the credit dynamics found in the model with a population of selfish peers that only upload in order to continue to download. We identified, in simulation, conditions that lead to both crunches and crashes where the system completely seizes - meaning no further sharing activity is possible. We applied a theoretical analysis to precisely characterise the conditions that lead to the system crunching or crashing. We validated the analysis against simulation runs and applied the derived conditions to implement a novel *credit intervention mechanism* that proactively stops the system seizing by temporarily changing the credit policies. A system that is predicted to crunch allows freeleeching (i.e. downloaders do not use any credit but uploaders still gain credit) conversely a system that is predicted to crash imposes freeseeding (i.e. downloaders use credit as normal but seeders do not gain credit). These interventions are only applied while the system is in such critical states. Freeleeching injects credit into the system and freeseeding removes credit from the system.

Our findings are based on the assumption of a selfish user model and an abstracted private community model. We have purposefully excluded differential upload and download rates and other plausible user model variants in order to understand the effect of credit dynamics in a system with known properties. Even given our assumptions we found that understanding and predicting credit dynamics and system behaviour was far from trivial.

Given these issues we can not claim that our adaptive credit policy could currently be deployed in a real private tracker community but would almost certainly need to be refined and informed by empirical work.

Possible future work could involve introducing a distribution of user model variants to our simulation model and analysis using a probabilistic rather than a deterministic approach. This may allow for some level of probabilistic prediction of crash or crunch states. More importantly, we may be able to induce probabilistic user models from empirical data collected from real private communities.

[3]The selfish user model could be interpreted as a so-called "satisficing" model since peers stop trying to gain credit when they reach a satisfaction threshold.

Acknowledgements

We wish to thank Boxun Zhang for supplying the data used to produce the CDF for sharing ratio. This work was partially supported by the Future and Emerging Technologies programme FP7-COSI-ICT of the European Commission through project QLectives (grant no.: 231200).

References

[1] Buttyan, L. and Hubaux, J.P. Enforcing service availability in mobile ad-hoc WANs. In *Proc. International symposium on Mobile ad hoc networking & computing*, pp.87–96, 2000.

[2] Cohen, B. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003

[3] Garbacki, P. and Epema, D.H.J. and van Steen, M. An Amortized Tit-For-Tat Protocol for Exchanging Bandwidth instead of Content in P2P Networks. In *Proc. First International Conference on Self-Adaptive and Self-Organizing Systems*, pp.119–128, 2007.

[4] Hales, D. and Rahman, R. and Zhang, B. and Meulpolder, M. and Pouwelse, J. BitTorrent or BitCrunch: Evidence of a credit squeeze in BitTorrent? In *Proc. IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pp.99–104, 2009.

[5] Kash, I.A. and Friedman, E.J. and Halpern, J.Y. Optimizing scrip systems: Efficiency, crashes, hoarders, and altruists. In *Proc. ACM Conference on Electronic Commerce*, pp.305–315, 2007.

[6] Legout, A. and Urvoy-Keller, G. and Michiardi, P. Rarest first and choke algorithms are enough. In *Proc. ACM SIGCOMM Conference on Internet Measurement*, pp.203–216, 2006.

[7] Levin, D. and LaCurts, K. and Spring, N. and Bhattacharjee, B. BitTorrent is an auction: analyzing and improving BitTorrent's incentives. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. SIGCOMM* , 2008.

[8] Locher, T. and Moor, P. and Schmid, S. and Wattenhofer, R. Free Riding in BitTorrent is Cheap. In *Proc. 5th Workshop on Hot Topics in Networks (HotNets)*, 2006

[9] Piatek, M. and Isdal, T. and Anderson, T. and Krishnamurthy, A. and Venkataramani, A. Do incentives build robustness in BitTorrent? In *Proc. of NSDI*, 2007

[10] Ramachandran, A. and Sarma, A.D. and Feamster, N. BitStore: An Incentive-Compatible Solution for Blocked Downloads in BitTorrent. In *Proc. 2nd Joint Workshop on Economics of Networked Systems and Incentive-Based Computing*, 2007.

[11] Sirivianos, M. and Park, J.H. and Yang, X. and Jarecki, S. Dandelion: Cooperative Content Distribution with Robust Incentives. In *Proceedings of the USENIX Annual Technical Conference*, pp.1–14, 2006.

[12] Turner, D.A. and Ross, K.W. A Lightweight Currency Paradigm for the P2P Resource Market. In *Proc. 7th International Conference on Electronic Commerce Research*, 2004.

[13] Vishnumurthy, V. and Chandrakumar, S. and Sirer, E.G. Karma: A secure economic framework for p2p resource sharing. In *Workshop on the Economics of Peer-to-Peer Systems*, 2003.