Dynamic File-selection Policies for Bundling in BitTorrent-like Systems

Nissan Lev-tov, Niklas Carlsson, Zongpeng Li, Carey Williamson, Song Zhang
Department of Computer Science, University of Calgary
{nlevtov, ncarlsso, zongpeng, carey, sozhang}@cpsc.ucalgary.ca

***Abstract*—BitTorrent-like swarming technologies are very effective for popular content, but less so for the 'long tail' of files with disparate popularities, which do not have sufficiently many peers to enable efficient collaboration. Performance degradations are especially pronounced in swarms with reduced file availability. *Static bundling* groups files into a single data content. It requires no modification to the BitTorrent client, and has been shown to improve availability of unpopular files in BitTorrent swarms. However, as peers are forced to download undesired file pieces, download times increase, especially for peers downloading popular files.**

We propose to use Stochastic Games and Markov Decision Process (MDP) to model and analyze optimal peer strategies, in a selfish and a cooperative setting respectively, for a BitTorrent-like system with multiple files. Each peer wishes to download a subset of the files, and we allow peers to *dynamically* decide whether to collaborate with peers targeting a different set of files or not, given the current system state. The Stochastic Game and MPD models take into account both piece availability and average download times, and allow us to study if and when downloading unwanted content can be beneficial. We use dynamic programming to solve the two models, contrast the level of collaboration observed in the selfish and the cooperative settings, and propose an enhanced piece selection mechanism for BitTorrent-like systems with dynamic download decision making. We demonstrate the effectiveness of dynamic file piece selection through both simulations and experiments using a modified BitTorrent client.

I. Introduction

In swarming technologies, such as BitTorrent, files or content (a fixed set of files) are typically split into many smaller blocks/pieces. A user interested in a particular file can join a swarm of users downloading the same file, and exchange blocks with them. The peer-to-peer design of BitTorrent allows the system capacity to grow with the swarm size, yielding high scalability and tolerance to flash crowds. To encourage cooperation, BitTorrent uses a simple, yet robust, rate-based Tit-for-Tat incentive mechanism, in which peers prioritize helping others that provide them the best download rates. Such reciprocation ensures that cooperating peers typically achieve better download performance than selfish peers.

Despite all the advantages, BitTorrent-like swarming technologies are not beneficial for unpopular content, which lacks sufficiently many peers to enable efficient collaboration [1], [2]. With the original seeder (or publisher of the content) typically not staying in the system during the whole lifetime of the torrent, a peer attempting to download unpopular content may therefore find it unavailable [3].

One option available to a content provider is to bundle/group unpopular files into a single content. Higher aggregate popularity of the bundle leads to increased content availability. The idea of using bundling to promote unpopular content can be traced back to at least decades ago, when in the literature of economics, bundling had been proposed as a mechanism for increasing sales, extending monopoly power and smoothing demands across multiple goods. In swarming systems, the effectiveness of bundling for improving the availability of unpopular content was also verified [4].

Bundling can be done either statically or dynamically. With *pure static bundling* [1], a static set of files is grouped together by the publisher, and every peer participates by downloading the entire set. Although requiring no modifications to existing BitTorrent software, pure static bundling forces peers to download more than they want, resulting in increased download times. In *mixed static bundling*, peers select a subset of the bundled files to participate in their swarming. Such freedom of choice, however, is not always supported by the publishers. Both static methods do not adjust the bundling decision during the lifetime of the swarm. In *dynamic bundling* [5], peers may be assigned complimentary content (files or part of files) to download at the time they decide to download a particular file.

This work is orthogonal to the distinction of static versus dynamic bundling. Rather than determining which files should be bundled, the primary focus of this work is on determining which download rules the peers should use, in systems with a mix of desired and undesired files. A natural question here is if and when peers benefit from downloading content they do not want, but may be leveraged in cross-torrent cooperation. Of particular interest are policies with which peers dynamically select whether to collaborate with peers targeting different files. In the bundling context, the set of complimentary files that a peer voluntarily participates in distributing does not have to be assigned to the peer as part of a bundle.

We use Stochastic Games [6] and a Markov Decision Process (MDP) [7] to analyze desirable peer strategies in a BitTorrent-like system with multiple files, in which peers makes upload and download decisions based on the current system-state information. Stochastic Games are a classic tool for evaluating the dynamics of repeated games, in which the players select actions for maximizing their respective benefit and the system state transition is non-deterministic. Similarly, MDPs have been used to model decision making for social benefits, in systems where the outcomes are partly under the

control of a decision maker and partly random. We use the a Stochastic Game to analyze dynamic piece selection among selfish peers, and use an MDP for cooperating peers. The two approaches share a unified, probabilistic system model that accounts for piece availability, the Tit-for-Tat mechanism, and average download times.

We present detailed system modelling using the Stochastic Game and MDP models, including designing system states and computing state transition probabilities. Dynamic programming techniques are then employed to solve the Nash Equilibrium and the social optimal solutions. Our analysis reveal that collaboration are rather unlikely to happen in a selfish environment, although they might be beneficial. This suggests that, the Tit-for-Tat scheme, a practical incentive engineering solution for single content systems, are no longer effective in dynamic multi-file systems. In the social optimum, we observe a number of instances when the system benefits from peers downloading content they do not want themselves. These cases include, in particular: (i) when the uploader is at the end of its download, (ii) when the downloader is at the beginning of its download, and (iii) when the downloader is at the end of its download. We provide intuitive explanations on why collaboration can be beneficial in these scenarios.

Based on the analysis results, we conclude that future work on cross-torrent collaboration incentives is important to fully achieve the capacity of a P2P swarming system, by enabling necessary inter-file peer collaboration. We further propose a new, enhanced piece selection algorithm that dynamically selects which file content to download, from a peer's perspective. This enhancement allows peers in BitTorrent-like systems to decide, based on its own download interest and the current system state information, which file pieces to download and distribute. We conduct both simulations and experiments using a modified BitTorrent client to evaluate and verify the effectiveness of such dynamic piece selection policies.

In the rest of the paper, Sec. II reviews background information on Stochastic Games and MDP; Sec. III and Sec. IV establish and solve the system models, and analyze the results. Sec. V proposes a corresponding enhancement to BitTorrent; Sec. VI presents performance evaluation, Sec. VII discusses related research and Sec. VIII concludes the paper.

II. Modeling Preliminaries

We first present preliminaries on the modeling tools to be used, including Stochastic Games for modeling file swarming among selfish peers, and the Markov Decision Process for modeling coopeartive piece selection strategies.

A. Stochastic Games

Introduced by Lloyd Shapley in the early 1950s, a *Stochastic Game* is a dynamic game with probabilistic state transitions. The game is played in a sequence of stages, each with an initial state. Each player selects an action, and receives an instantaneous reward based on the current state and the set of actions chosen by all players. The game then moves to a random new state whose distribution also depends on the previous state and the chosen actions. Such a stage is repeated for a finite or infinite number of rounds. The total payoff to a player is taken as the discounted sum of the stage rewards, or the limit inferior of the averages of the stage rewards [6].

Formally, a Stochastic Game consists of a set of players N, a state space S, an action set A_i for each player $i \in N$, a transition probability P from $A \times S$ to S , where $A = \prod_{i \in N} A_i$, and $P(s \mid a, s')$ is the probability that the next state is s given the current state s' and the current action profile a. A reward function R is from $A \times S$ to $\Re^N$, and its i-th coordinate, R_i, is the payoff to player i as a function of s and a. The game starts at an initial state s_0. During each stage t, players first observe the state s^t, and simultaneously choose actions $a_i^t \in A_i$. The new state $s^{t=1}$ is then revealed according to the probability $P(\cdot \mid s^t, a)$.

A (pure) *policy* or *strategy* for a player is a choice of which action to take in each state in S. A *mixed policy* is a probability distribution over the policy profile. A *Nash equilibrium policy* contains a policy for each player, such that no player can benefit from unilaterally deviating from its current policy.

When all the dimensions of the game (number of players, set of actions, set of states, number of stages) are finite, a Stochastic Game always has a (mixed) Nash equilibrium. The same is true for a game with infinitely many stages if the total payoff is a discounted sum. Stochastic Games are generalizations of MDPs, which correspond to the case of a single player only.

B. Markov Decision Process

Markov Decision Processes (MDPs) provide a natural framework for modeling decision-making, when outcomes are partly under the control of a decision maker yet partly random. In BitTorrent, peer unchoking and file piece selection can exhibit such a probabilistic nature.

More precisely, an MDP is a discrete time stochastic control process, which is a special case of a Stochastic Game. In an MDP, there is only one centralized decision made during each stage. In the context of BitTorrent systems, this corresponds to centralized optimization of block exchange decisions among all peers. Given the current system state s and an action a, the process responds by randomly moving into a new state s', and giving the decision maker a reward $R_a(s, s')$. New state selection is governed by the state transition function $P_a(s, s') = P(s^{t+1} = s \mid s^t = s, a^t = a)$. Such a function possesses the Markov property, *i.e.*, it is conditionally independent of previous states and actions.

The core problem of MDPs is finding a policy for the decision maker, *i.e.*, a function $\pi(s)$ that specifies the action to take in state s, with a goal of maximizing a cumulative function of the rewards, *e.g.*, the average over a finite horizon or the expected discounted sum over an infinite horizon. In Sec. IV-A, we employ backward induction coupled with dynamic programming for solving the MDP model in BitTorrent-like systems.

III. P2P Swarm System Modeling

A. Motivation and Assumptions

We use Stochastic Games and MDPs to model the system behavior in multi-file systems with heterogenous file popularity, for selfish and cooperative peers, respectively. We describe the model for two files f_1, f_2, for ease of presentation; the model can be easily generalized to more than two files. Each file is divided into k blocks. There are three types of players u_1, u_2, u_3, interested in f_1, f_2, and both files, respectively. A type becomes *active* upon entering the system, and becomes *inactive* at the time it finishes downloading the desired file(s). A seed is initially present, ensuring piece availability, but leaves the system at a later time point.

The heterogenous file popularity is captured by n_u^t, which means that the system behaves as though there are n_u^t peers of type u at time t, where all these peers are in the same state as u and take the same actions. For simplicity, we first consider the case of time-invariant popularity, *i.e.*, $n_u^t = n_u$, and the total number of peers $n = n_{u_1} + n_{u_2} + n_{u_3}$. This assumption is more relevant for flash crowd scenarios, and can be extended to any fixed number of peer sets arriving at any point in time, where each set consists of peers of a certain file type. We also relax this time-invariant assumption in empirical studies in Sec. VI.

In our model, an action for each type is defined as a download file selection rule. We assume that peers have similar upload capacities and unbounded download capacity. The blocks of each file are assumed to be uniformly distributed. More specifically, if a type u has m blocks, then each subset of size m of $1,\ldots,k$, has the same likelihood to appear. Furthermore, since peers have a limited degree of patience, we assume the game is of a finite horizon, *i.e.*, the number of stages/steps is a fixed parameter T.

We next provide details on each element of the system model, including states, rewards, actions and state transition probabilities.

B. System States

A system state s in our model consists of three sub-states:

- **Type state:** A binary vector recording which types are active and which inactive.
- **Seed state:** A binary variable indicating whether the system is currently with or without a seed.
- **File state:** A 2D array storing the number of blocks of each file that each player has downloaded.

For the type state, let $isActive(u,s)$ be 1 if u has not finished downloading its file, and 0 otherwise. For example, $isActive(u_1,s) = 1$ iff $s_1(u_1) < k$. For the file state, let $s_f(u)$ denote the *local state* of u, *i.e.*, the number of blocks of file f that type u has. The state of the system can be represented by the position of the three players in a $(k+1)\times(k+1)$ grid, where the rows and columns correspond to the number of blocks of file f_1 and of file f_2, respectively. Figure 1 depicts an example state of the system.

The seed state helps decide whether reward can be issued to the players or not. In the case where the seed has left, and the system does not have sufficient aggregate piece availability to recover the files, stage rewards will be zero.

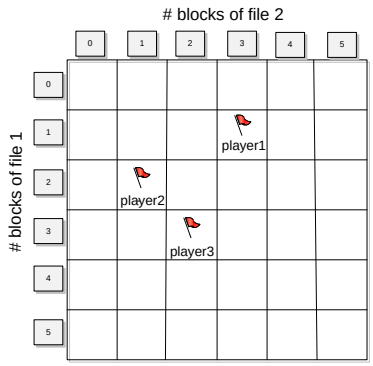

Fig. 1. Example file state.

C. The Reward Function

When a Nash Equilibrium of the Stochastic ame is seeked, each type optimizes its own reward; for the *social optimum* in MDP, the types act as a single agent optimizing the aggregated reward for all types. We consider three reward functions R for the players, modeling different performance measures.

- **Discounted average download rates**. The average rate a player receives at time t is multiplied by a discount factor γ^t. The players seek to maximize the discounted average number of received blocks of desired files during a specified time T.
- **Average bounded download time**. The players seek to minimize the average download time of desired files during the specified time T. If a type u has not finished downloading its desired file by time T, then its download time is taken as $T \times n_u$ in the average calculation.
- **Average availability**. The reward R for a type u is either $1 \times n_u$ if u has finished downloading by time T, or 0 otherwise.

D. Type Actions

At each time step, each type u chooses a download action $d(u,v)$ against each type v, among the following:

- $d(u,v) = d_1$**:** u downloads only blocks of f_1 from v (expected behavior of type u_1)
- $d(u,v) = d_2$**:** u downloads only blocks of file f_2 from v (expected behavior of type u_2).
- $d(u,v) = d_3$**:** u treats f_1 and f_2 as a single combined file, and downloads a block independent of its file origin (fixed bundling/expected behavior of type u_3).
- $d(u,v) = d_4$**:** u downloads unwanted blocks from v only if v does not have blocks u needs. In contrast to the first three rules, this rule is conditional and implies downloading blocks of the two files while giving priority to the file it wants.

E. State Transition Probabilities

We now describe the state transitions probabilities that define the *laws of motion* of the model. Essentially, we need to compute the probability distribution for the number of blocks of each file that a type u receives from a type v, during a given round. We wish to capture the Tit-for-Tat unchoking mechanism implemented in BitTorrent. The overall computation happens in five successive steps: **(i)** compute $p(u,v)$, probability that a type v has blocks that u wants, **(ii)** compute $b(u,v)$, probability of bidirectional interest between types u and v, **(iii)** compute the number of nodes with bidirectional interest with type v, **(iv)** compute the probability that v selects u to upload a block in f_i, and **(v)**, compute the number of blocks in f_i that u obtain. We next present details for each step.

(i) Compute $p(u,v)$.

Consider a system state s, and recall that the file state of a player type u is defined as $(s_1(u), s_2(u))$. We first present two lemmas.

Lemma 1: The probability that v has a block of file f_i that u does not have is

$$p_i(u,v) = \begin{cases} 1 & \text{if } s_i(u) < s_i(v) \text{ and} \\ 1 - \binom{s_i(u)}{s_i(v)} / \binom{k}{s_i(v)} & \text{otherwise} \end{cases}$$

Proof: Consider player u and the file f_i (containing k blocks). If $s_i(u) < s_i(v)$, v has a block to share with u with probability 1. If $s_i(u) \geq s_i(v)$, the probability that v will have a piece of the file f that u does not have is one minus the probability that one of the subsets of size $s_i(v)$ of the blocks that u has, is exactly the set of blocks that v has. Due to the uniformity assumption, this probability is $1 - \binom{s_i(u))}{s_i(v)} / \binom{k}{s_i(v)}$. □

Lemma 2: The probability that v has a block (of any file) that u does not have is

$$p(u,v) = p_1(u,v) + p_2(u,v) - p_1(u,v)p_2(u,v)$$

(ii) Compute $b(u,v)$.

Using probabilities $p_i(u,v)$ and $p(u,v)$, we next calculate the probabilities of bidirectional interest between types u and v, assuming current system state s and the download action profile d chosen by the peers. For instance, assume u and v both take download action d_1 (only download blocks of file f_1). If neither v or u possesses blocks of f_1, then $b(u,v) = 0$. If exactly one of u and v has zero blocks of f_1, then we take $b(u,v)$ to be $\frac{1}{5n}$, to model the *optimistic unchoke* mechanism that applies for choosing to upload to peers which are in the beginning of their download process. (See [8] for a description of the optimistic unchoke mechanism in BitTorrent). Otherwise, if $s_1(u) > s_1(v) > 0$, then with probability 1 u has a block of f_1 that v does not have, and thus the probability of bidirectional interest $b(u,v)$ is $p_1(u,v)$. Similarly, if $s_1(v) > s_1(u) > 0$, $b(u,v) = p_1(v,u)$. If $s_1(u) = s_1(v) > 0$, then either u and v have the same set of blocks, or each has a block that the other does not have. That is, if u has a block of f_1 that v does not have, then with probability 1 v also has a block that u does not have. Therefore in this equality case, the probability of bidirectional interest $b(u,v) = p_1(u,v)$. We summarize the calculation for several representative action profiles in Table I. Computing $b(u,v)$ for other profiles can be done in a similar fashion.

TABLE I
BIDIRECTIONAL INTEREST PROBABILITY $b(u,v)$

Conditions (if-else in order applied)			
$d(u,v)$	$d(v,u)$	$s(u,v), s(v,u)$	$b(u,v)$
d_1	d_1	$s_1(u) = 0 \wedge s_1(v) = 0$	0
		$s_1(u) = 0 \vee s_1(v) = 0$	$\frac{1}{5n}$
		$s_1(u) \geq s_1(v)$	$p_1(u,v)$
		$s_1(u) < s_1(v)$	$p_1(v,u)$
d_1	d_2	$s_2(u) = 0 \wedge s_1(v) = 0$	0
		$s_2(u) = 0 \vee s_1(v) = 0$	$\frac{1}{5n}$
		$s_2(u) > s_2(v)$	$p_1(u,v)$
		$s_1(v) > s_1(u)$	$p_2(v,u)$
		$s_2(u) \leq s_2(v) \wedge s_1(v) \leq s_1(u)$	$p_1(u,v)p_2(v,u)$
d_1	d_3, d_4	$s_1(u) = 0 \wedge s_2(u) = 0 \wedge s_1(v) = 0$	0
		$s_1(u) = 0 \wedge s_2(u) = 0 \vee s_1(v) = 0$	$\frac{1}{5n}$
		$s_1(u) \geq s_1(v) \vee s_2(u) > s_2(v)$	$p_1(u,v)$
		$s_1(v) > s_1(u)$	$p(v,u)$
d_3, d_4	d_3, d_4	$s_1(u) = 0 \wedge s_2(u) = 0 \wedge s_1(v) = 0 \wedge s_2(v) = 0$	0
		$s_1(u) = 0 \wedge s_2(u) = 0 \vee s_1(v) = 0 \wedge s_2(v) = 0$	$\frac{1}{5n}$
		$s_1(v) > s_1(u) \vee s_2(v) > s_2(u)$	$p(v,u)$
		otherwise	$p(u,v)$

(iii) Compute the number of nodes with bidirectional interest with type v.

In order to calculate the probability that a peer of type v will choose a peer of type u (given S and d), in step (iv), we will compute how many nodes have bidirectional interest with v and therefore compete with u. To model the Tit-for-Tat mechanism, we assume that with j peers having bidirectional interests with v, the probability v will choose u is $5/j$. (The BitTorrent unchoking method usually assumes that about 5 peers are unchoked at the same time). The average number of peers in u having bidirectional interest with the peer in v is $\lambda B(u,v) = n_u b(v,u)$. The total average number of peers having bidirectional interest with the peer in v is

$$\lambda B(v) = \lambda B(u_1,v) + \lambda B(u_2,v) + \lambda B(u_3,v).$$

The probability that exactly j peers have bidirectional interest with the peer in v can be approximated using a Poisson distribution with average rate $\lambda B(v)$. Consequently, we have the following probability that v will choose to upload to u, assuming bidirectional interest between them:

$$\begin{aligned} Pr(Choose(v,u) \mid s,d) &= \sum_{i=0}^{n} \frac{5}{(i+1)!} \lambda B(v)^i e^{-\lambda B(v)} \\ &\approx \frac{5}{\lambda B(v)} (1 - e^{-\lambda B(v)}) b(u,v). \end{aligned}$$

TABLE II
PROBABILITY OF v UPLOADS f_i TO u, $Pr(Upload(v,u,i))$

Conditions		
$d(u,v)$	$type(u)$	$Pr(Upload(v,u,i))$
d_1	all	$Pr(Choose(v,u) \mid s,d)p_1(u,v)$
d_2	all	$Pr(Choose(v,u) \mid s,d)p_2(u,v)$
d_3	all	$Pr(Choose(v,u) \mid s,d)\frac{s_i(u)(k-s_i(u))}{s_1(v)(k-s_1(u))+s_2(v)(k-s_2(u))}$
d_4	u_i	$Pr(Choose(v,u) \mid s,d)p_i(u,v)$
	$u_j,\ j \neq i$	$Pr(Choose(v,u) \mid s,d)p_j(u,v)(1-p_i(u,v)$

(iv) Compute the probability that v selects u to upload a block in f_i.

We can now calculate the probability that v will upload a block of file f_i to u by multiplying the above probability with the probability p_i that the uploaded block will be of file f_i. In case u chooses download rule d_3, this probability can be calculated using Bayes Rule,

$$p_i = \frac{\frac{1}{2}\frac{s_i(v)}{k}\frac{k-s_i(u)}{k}}{\frac{1}{2}\frac{s_1(v)}{k}\frac{k-s_1(u)}{k}+\frac{1}{2}\frac{s_2(v)}{k}\frac{k-s_2(u)}{k}} = \frac{s_i(u)(k-s_i(u))}{s_1(v)(k-s_1(u))+s_2(v)(k-s_2(u))}$$

If u chooses download rule d_4 this probability is $p_i = p_i(u,v)$ in case $u = u_i$, and $p_i = p_j(u,v)(1-p_i(u,v))$ in case $u = u_j$ for $j \neq i$. Table II summarizes the probability that v uploads a block of file f_i to u , assuming the current state s and download action profile d.

(v) Compute the number of blocks in f_i that u obtains.

The number of blocks of file f_i that u receives from v in a round is approximated by the Poisson distribution with average rate $\lambda_v(u,i) = n_v Pr(Upload(v,u,i) \mid s,d)$, and the total number of blocks of file f_i that u receives in a round is approximated by the Poisson probability distribution with average rate

$$\lambda(u,i) = \lambda_{u_1}(u,i) + \lambda_{u_2}(u,i) + \lambda_{u_3}(u,i),$$

and the probability that type u will gain m blocks of file f_i in this round is

$$Pr(Gain(u,i,m) \mid s,d) \approx \frac{1}{(m)!}\lambda(u,i)^m e^{-\lambda(u,i)}.$$

Finally, the system state transition probabilities are obtained by multiplying the above probabilities for all types and files.

All details of our BitTorrent-like system model are now obtained. The next step is to *solve the model*, including finding and analyzing Nash Equilibriums for the Stochastic Game, and computing and analyzing the socially-optimal piece selection rules for the MDP. We discuss such solutions and results in Sec. IV.

IV. OPTIMAL STRATEGIES FOR COOPERATIVE AND SELFISH PEERS

We first describe a dynamic programming method for solving the model established in Sec. III, then analyze the optimal cooperative behavior found for the MDP model, and the selfish behavior observed in the Stochastic Game model.

A. Solving The MDP Model with Dynamic Programming

Given the state transition function P and the reward function R of an MDP, standard algorithms [9] for calculating the optimal policy iteratively refine two arrays indexed by states. Each state has a value V for reward, and a policy π. When the algorithm converges, π contains the solution and $V(s)$ stores the reward for following that solution from state s.

Such an algorithm is based on the following two recursive equations, iteratively computed for all the states, until no further changes occur. Here $\gamma \in (0,1]$ is the discount factor.

$$\pi(s) = \arg\max_a \{R_a(s,s') + \gamma \sum_{s'} P_a(s,s')V(s')\}$$

$$V(s) = R_{\pi(s)}(s) + \gamma \sum_{s'} P_{\pi(s)}(s,s')V(s')$$

In *backward induction* [9], $\pi(s)$ is not stored, but calculated whenever needed, using the *Bellman Equation*:

$$V^t(s) = \max_a (R_a(s) + \gamma \sum_{s'} P_a(s,s')V^{t-1}(s')).$$

A dynamic programming approach can be employed to solve the above equation, where t decreases from T to 0. Below we provide details for each of the three objectives:

Reward $R_a(s)$ = discounted average download rates.

We have that

$$V^t(s) = \max_a \gamma \sum_{s'} P_a(s,s')(R(s,s') + V^{t-1}(s')),$$

where $V^T(s) = 0$, and $R(s,s')$ is the sum of the increments for the requested files, namely,

$$R(s,s') = n_{u_1}(s'_1(u_1) - s_1(u_1)) + n_{u_2}(s'_2(u_2) - s_2(u_2)) + n_{u_3}((s'_1(u_3) - s_1(u_3)) + (s'_2(u_3) - s_2(u_3))).$$

Reward $R_a(s)$ = bounded average download times.

We have that

$$V^t(s) = \min_a (R_a(s) + \sum_{s'} P_a(s,s')V^{t+1}(s')),$$

where $V^T(s) = 0$, and

$$R_a(s) = \sum_{i=1,2,3} isActive(u_i,s)n_{u_i}.$$

Reward $R_a(s)$ = bounded average availability.

We have that

$$V^t(s) = \max_a (\sum_{s'} P_a(s,s')V^{t+1}(s')),$$

where

$$V^T(s) = \sum_{i=1,2,3} (1 - isActive(u_i,s))n_{u_i}.$$

B. Social-Optimal Piece Selection

We have analyzed the results obtain for the Markov Decision Process in order to obtain insights regarding the social optimum, that will enable us to enhance the piece selection method in BitTorrent-like file swarming systems. In the majority of the cases we examined, about 90% of the download decisions are to download only the desired files, *e.g.*, d_1 for type u_1. However, we further identified a number of scenarios in which collaboration does occur.

(a) When the seed exits the system at an early time and the first type u_1 has very low popularity ($n_{u_1} < k$), then while the seed is active the more popular type u_2 frequently collaborates by choosing download rule d_1, d_3 or d_4 against u_1.

(b) When the total number of blocks of file f_i in the system is low (taking into account the popularity of the types as well as their number of blocks of f_i), then type u_j, $(j \neq i)$ sometimes collaborates with u_i. Especially (around 80% of the incidents of collaboration), collaboration occurs when:

b1. The other type (u_i) is at the end of its file download. (This can be explained by u_j "saving" blocks of u_i before u_i exits the system.)

b2. u_j is at the beginning of its file download, then it frequently uses download rule d_3 or d_2. (This can be explained by u_j increasing its probability of getting the first blocks.)

b3. u_j is at the end of its download but has blocks of f_i. (This can be explained by u_j delaying its finish so that u_i can download its rare blocks.)

b4. The number of blocks of file f_j in the system is not low.

The results suggest that the types tend to be "socially active" when their own performance is guaranteed (blocks of desired files are dense) but blocks of the other file are rare, and turn to "selfish behavior" (asking only for blocks of the file they are interested in), when their file becomes rare or is likely to become rare in the near future, or when there is no rarity problem with the other file. Furthermore, collaboration occurs mostly in the end states of either the download process of the downloader or the uploader.

C. The Stochastic Game Model and Selfish Behaviors

For the Stochastic Game model, we used a dynamic programming approach similar to that in Sec. IV-A to find an approximate Nash Equilibrium, for which the total gain is optimized. Note that if both types u_1 and u_2 follow the strategy of downloading only the files they want, then this strategy profile is in equilibrium since no type will benefit from deviating from it unilaterally. Although this is not the best equilibrium for social benefit, the best is not much different.

In the socially optimal approximate Nash Equilibrium observed, for up to 99.9% of the time each type downloads its desired file(s) only. The other rare cases occur in scenarios similar to the scenarios stated in Sec. IV-B.

In conclusion, we observe that collaboration among the peers, in the form of downloading unwanted data to help other peers, can indeed be beneficial. However, when peers are all selfish and focus on their own download performance only, such collaboration is much less likely to occur. This contrast confirms the necessity of incentive engineering in multi-file data swarming.

In the next section, we describe the enhancement to the piece selection method based on the observations just made. The proposed enhancement applies to bundles containing any fixed number of files, not just two files as assumed in the simplifying model.

V. Piece Selection Method Modification

BitTorrent traditionally relies heavily on a *rarest first* policy for piece selection. With this policy, unchoked peers request with highest priority the piece that the least number of neighbors have, among blocks it desire. This section describes our modified piece selection algorithm. The algorithm is new in that it takes into account not only pieces that are desired, but also pieces the peer is not directly interested in. For simplicity, we assume that a peer is downloading a bundle with a set of pieces B, from which it wants a subset $F \subset B$ (and does not want the remaining subset $G = B \setminus F$). Furthermore, it is assumed that the piece IDs in the bundle are known and each subset can be identified.

Let $nRare(S)$ denote the number of pieces in a subset $S \subset B$ that is considered rare among the neighbors of the downloader. For the purpose of our simulations, we assume a piece is rare when there are no more than one copy of the piece in the set of neighbors. Let U and D denote the set of pieces that the uploader and downloader has, respectively.

Our modified piece selection algorithm is described in Algorithm 1. While the peer primarily download pieces from the desired subset F, there are cases in which the downloader asks for blocks of undesired files. Based on our modeling results using the MDP (Sec. IV-B), these cases corresponds to the following:

- **The uploader is at the end of its download.** Here, the condition of how far the uploader must be in its download for this condition to trigger is dynamically set according to the number of rare blocks observed by the downloader. In particular, if the downloader observes many rare blocks of the uploader's file, then it will start replicating them earlier, before the uploader exits the system. In this case it is also necessary that the downloader is not at the end of its own download.
- **The downloader is at the beginning of its download.** To reduce the time until a peer can start exchange pieces using the rate-based Tit-for-Tat technique, a new downloader may benefit from quickly getting rare pieces to share.
- **The downloader is at the end of its download.** Here, the threshold determining how far the downloader must be from finishing its download is dynamically set according to the number of rare blocks that the downloader has. With the current rule, a downloader that has many rare blocks will try to defer its departure from the system, helping replicate rare pieces.

Algorithm 1 *SelectPiece*(F,G,D,U)

$m_{F,\neg D} \leftarrow nRare(F \cap (B \setminus D))$
{# rare pieces in F that the downloader does not have}
$m_{G,\neg D} \leftarrow nRare(G \cap (B \setminus D))$
{# rare pieces in G that the downloader does not have}
$m_{F,D} \leftarrow nRare(F \cap D)$
{# rare pieces in F that the downloader has}
$m_{G,D} \leftarrow nRare(G \cap D)$
{# rare pieces in G that the downloader has}
if $m_{F,\neg D} = 0$ **and** $m_{G,\neg D} \geq 1$ **then**
  {downloader is missing rare pieces in G, but not in F}
  if $|U| \geq (|G| - m_{G,\neg D})$ **and** $|D| < (|F| - m_{G,\neg D}) - 1$ **then**
    {uploader is almost finished downloading G}
    return Rarest piece
  if $|D| \leq 1$ **then**
    {downloader is at the beginning of downloading F}
    return Rarest (or random) piece with priority to the pieces in F
  if $|D| > |F| - \max(m_{F,D}, m_{G,D}) + 1$ **then**
    {downloader is almost finished downloading F and has rare pieces}
    return Rarest piece in G.
return Rarest piece that belongs to F

Note that our algorithm is relatively easy to implement, does not require drastic changes of the current BitTorrent client, and does not require any changes to the communication between peers. For the purpose of our experiments in Sec. VI, we only changed the pieces selection used at the time of unchoke messages. Nonetheless, simulations results and experiments show that these slight modifications are indeed capable of improving performance of the entire system.

VI. Performance Evaluation

A. Simulation Results

To study the performance advantages of the modified piece selection policy, we used a discrete event simulator of a BitTorrent system. Our simulations assume that peers have uniform upload capacities, can upload one piece per second, and have unlimited download capacities. Identical scenarios were simulated both with and without our modification, to compare their performance . We present example results for three scenarios. In all scenarios there is a single seeder with upload capacity of one or two pieces per second, which remains in the system for the majority of the simulation time, but leaves before all peers have completed their download. Peers are interested in f_1, f_2, or both, and 30% of the peers interested in the less popular file are interested in both files.

In Scenario 1, there are a total of 200 peers arriving, 80% of which arrive (uniformly) to the system within the first 700 seconds, and the remaining 20% of peers arrive at an exponentially decreasing rate. The seed enters the system at time 0 and exits 1,600 seconds later. Both files have 400 pieces. Figure 2 depicts a comparison between the download

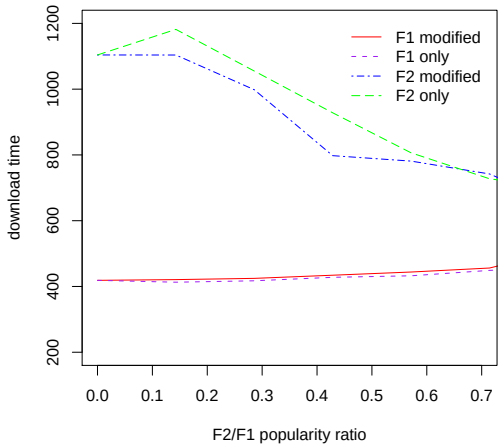

Fig. 2. Average download times for various popularity ratios in Scenario 1.

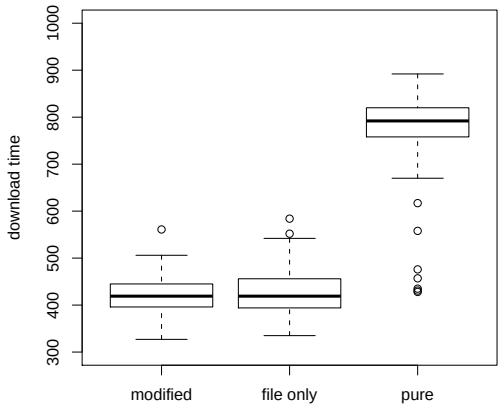

Fig. 3. Download time statistics for Scenario 2.

times using our proposed dynamic piece selection versus downloading desired files only (and then exit the system). The upper lines show the average download times for the less popular file and the lower lines show the average download times for the more popular file. The ratio between the low and high popularity varies from 0 to 1 (x-axis). We observe that the download times for the popular files (as well as the overall average) are similar, but the reduction in download times of the less popular file is evident.

In Scenarios 2 and 3, all peers arrive during the initial flash crowd. In Scenario 2, 200 peers arrive (at uniform rate) within the first 700 seconds, and the seed leaves the system after 800 seconds. Similar to Scenario 1, each file has 400 pieces. Here the ratio between the low and high popularity files is fixed to 0.1. In Scenario 2, on the other hand, the seed stays much longer, and does not leave until after 1,500 seconds. Here, we assume that there are 250 peers, all of which arrive (at a constant rate) within the first 800 seconds. Each file has 500 pieces, and the ratio between the low and high popularity files is set to 0.05.

Figure 3 depicts the median, lower and upper quartiles and extreme observations for the download times of peers

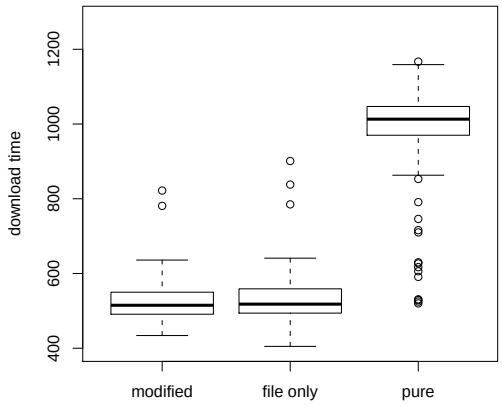

Fig. 4. Download time statistics for Scenario 3.

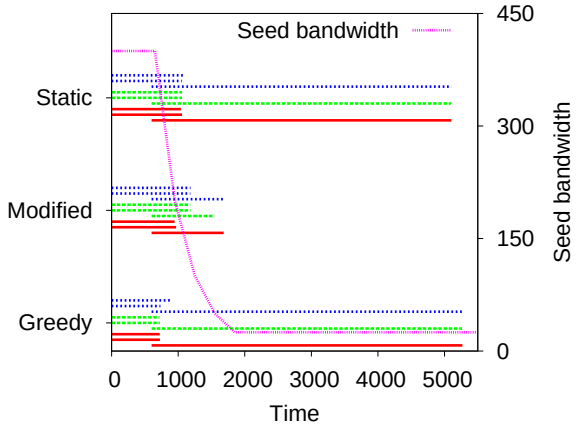

Fig. 5. Download times for experimental scenario.

in Scenario 2. Figure 4 depicts the corresponding statistics for Scenario 3. The leftmost box refers to the results using our proposed piece selection. The center box refers to the case where peers download only pieces from desired files, and the rightmost box refers to pure bundling, where peers exit the system after downloading the desired files. In these scenarios there is a slight gain in the median and quartiles, and a more considerable gain for the maximum values, when using our modification. In summary, these results show that the modification is effective in reducing the higher download times without giving up much in terms of average performance, especially when unpopular files present.

B. Experimental Results

Our implementation of the piece selection algorithm is based on the mainline BitTorrent version 3.3. We deploy one seed and a total of nine lechers (non-seed peers): eight distributed across the University of Calgary campus, and one remote peer in Toronto (two time zones away). Two files, each at approximately 130MB, are swarmed.

For the purpose of comparison, we have one implementation based on the piece selection algorithm in Sec. V, one implementation ("greedy") in which the peers only request pieces from desired files, and a third implementation in which the peers download all pieces in the bundle (static bundling). As previously noted, we only changed the piece selection module at the time of unchoke messages. Piece requests made due to an incoming *have* messages were initiated if the *have* message included a piece of F that the peer does not have. In all experiments, each peer requests the whole bundle (with metafile) at the beginning of the download; the policies only differ by the piece requests they make.

With the exception of the pure static bundling policy, the peers will leave the swarm as soon as they finish downloading their desired file(s). For static bundling, the peers leave when they have obtained both files, independent of their original interest. To emulate the under provisioning of the seed, we modify the source code and cap the seed's upload capacity dynamically from 400kB/s to 25kB/s: 400kB/s at the beginning, decreases to 200kB/s after 650 seconds, and decrease to half again in every other 300 seconds, until reaching 25kB/s. The leechers' upload throughput is fixed at 200kB/s all the time (small variations might occur due to external network traffic though). We do not put any additional caps to the download capacity of peers.

In our experiments, six lechers arrive at time 0 and another three 600 seconds later (just before the seed bandwidth starts to reduce). Here we are interested in a scenario of a fading torrent, where the original seeder has lost interest in seeding the content, yet we still wish to achieve *graceful termination*, with the "final" downloaders receiving reasonable service. Two peers from the first group and one from the second group want file 1 (red lines in Fig. 5), the same for file 2 (green) and for both files (blue). The left-most point of each line corresponds to the time of the peer's arrival and the right-most point corresponds to download completion. The purple line shows the seed bandwidth over time. We can observe that, the download times of the later arriving peers can be reduced by almost 80%, using our modified piece selection policy. There is a significant advantage using the modified dynamic piece selection policy as torrents are (inevitably) moving towards the end of their life span and fading.

VII. Previous Research

While most research efforts towards understanding BitTorrent have focused on single-torrent systems, some recent studies consider multi-torrent environments [1], [3], [5], [10], [11]. For example, measurements have shown that more than 85% of torrent users simultaneously participate in multiple torrents [10], [12]. Techniques have been proposed that extend the "life-time" of torrents by having peers downloading one torrent also act as seeders in other torrents (of files that they may previously have downloaded) [12]. Yang *et al.* [11] propose a cross-torrent Tit-for-Tat scheme, based on peers aggregate download rates (across all torrents), which provide incentives for such behavior. Other related studies on incentives in inter-swarm exchanges include ones that propagate

peer reputation [13], as well as incentive-based token [14] and credit [15] schemes.

Perhaps most related to our work is that of Menasche *et al*. [1]. They use queuing models to show that "bundling" multiple files into a larger file may increase the file availability. While bundling is proposed for the purpose of improved file availability, in some cases, they also show that bundling can reduce the download times even for clients that only are interested in downloading part of a larger bundle. Dynamic bundling approaches have also been proposed in which peers are dynamically assigned secondary swarms, based on current state information, which they should help [5]. In contrast to such research on bundling, we consider a system in which peers can select which part of a bundle they wants to download, and focuses on the incentives (and games) that such peers are exposed to. In particular, we are interested in determining what the optimal strategies such peers may have. We consider strategies from both a selfish and a social perspective.

Other related work have considered swarm management [16], [2]. For example, Peterson *et al.* [16] proposed a swarm coordination system that directs bandwidth allocation at each peer, such that the peers allocate upload capacities between different torrents in order to optimize download performance. Dan and Carlsson [2] investigated the advantages offered by dynamically merging small swarms of the same torrent (rather than different torrents).

For single torrent systems, a stochastic differential equation model has been used to capture the peer behavior and file availability issues in BitTorrent [17]. In contrast with that work, we consider multi-file torrents and use controlled stochastic models, where the agents or players choose their actions to optimize their performance.

VIII. Conclusions

We consider peer strategies in the context of BitTorrent systems using file bundling, where a peer may only be interested in downloading a subset of the bundled files. Using a single unifying probabilistic model which takes into account factors including piece availability, Tit-for-Tat and average download times, we allow peers to dynamically select whether to collaborate with peers targeting different files, depending on the realtime state of the system. Using this framework, we apply both Stochastic Games and a Markov Decision Process (MDP) to analyze desirable peer strategies from a selfish and a social perspective, respectively. While selfish peers typically do not download any other content than what they truly want, our analysis identifies and characterizes cases where downloading unwanted content is socially beneficial, suggesting that future research on the topic of more advanced cross-torrent incentive schemes is not only important, but necessary if we want to achieve the social optimum. Based on these characterizations, we propose a new, enhanced piece selection method that dynamically selects which file content to download. Both simulations and experiments using a modified BitTorrent client are used to verify the effectiveness of the policy.

References

[1] D. S. Menasche, A. A. A. Rocha, B. Li, D. Towsley, and A. Venkataramani, " Content Availability and Bundling in Swarming Systems ," in *Proc. of CoNext*, Rome, Italy, Dec. 2009.

[2] G. Dán and N. Carlsson, "Dynamic Swarm Management for Improved BitTorrent Performance," in *Proc. of IPTPS*, Boston, MA, Apr. 2009.

[3] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems," in *Proc. of IEEE INFOCOM*, Anchorage, AK, May 2007.

[4] D. Menasche, G.Neglia, D.Towsly, and S.Zilberstein, "Strategic reasoning about bundling in swarming systems," in *Proc. of GameNets*, Istanbul,Turkey, May 2009.

[5] N. Carlsson, D. Eager, and A. Mahanti, "Using Torrent Inflation to Efficiently Serve the Long Tail in Peer-assisted Content Delivery Systems," in *Proc. of IFIP/TC6 Networking*, Chennai, India, May 2010.

[6] A.Neyman and S.Sorin, "Stochastic Games and Applications," in *Kluwer Academic Press*, 2003.

[7] R.Bellman, " A Markovian Decision Process ," *Journal of Mathematics and Mechanics*, vol. 6, 1957.

[8] A.Legout, G.Urvoy-Keller, and P.Michiardi, "Rarest first and choke algorithms are enough," in *Proc. of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 203–216.

[9] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957. Republished 2003 by Dover.

[10] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurement, Analysis, and Modeling of BitTorrent-like Systems," in *Proc. of ACM IMC*, Berkeley, CA, Oct. 2005.

[11] Y. Yang, A. L. H. Chow, and L. Golubchik, "Multi-Torrent: A Performance Study," in *Proc. of MASCOTS*, Baltimore, MD, Sept. 2008.

[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, " A Performance Study of BitTorrent-like Peer-to-Peer Systems ," *JSAC*, vol. 25, no. 1, pp. 155–169, 2007.

[13] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson, " One Hop Reputations for Peer to Peer File Sharing Workloads ," in *Proc. of NSDI*, San Francisco, CA, April 2008.

[14] A. Ramachandran, A. das Sarma, and N. Feamster, " Bitstore: An Incentive Compatible Solution for Blocked Downloads in BitTorrent ," in *Proc. of NetEcon*, San Diego, CA, June 2007.

[15] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, " Free-riding in BitTorrent Networks with the Large View Exploit," in *Proc. of IPTPS*, Bellevue, WA, Feb. 2007.

[16] R. S. Peterson and E. G. Sirer, "Antfarm: Efficient content distribution with managed swarms," in *Proc. of NSDI*, Boston, MA, May 2009.

[17] B.Fan, D. Chiu, and J. Lui, "Stochastic analysis and file availability enhancement of bt-like file sharing systems," in *Proc. of IEEE IWQoS*, New Haven, CT, June 2006.