

# DESIGN OF APPLICATION-SPECIFIC INCENTIVES IN P2P NETWORKS

*Andrew Roczniak, Abdulmotaieb El Saddik*

*Ross Kouhi*

Multimedia Communications Research Lab  
University of Ottawa, Ottawa, Canada  
Email: {roczniak, abed}@mcrmlab.uottawa.ca

Bell Labs-Service Infrastructure Research  
Alcatel-Lucent, Ottawa, Canada  
ross.kouhi@alcatel-lucent.com

## ABSTRACT

A rational P2P node may decide not to provide a particular resource or to provide it with degraded quality. If nodes are very likely to behave this way, or if the failure of an P2P-based application is associated with serious consequences, then usage of P2P infrastructure to support this application becomes questionable.

Timely and extensive research works address this issue. These are presented and discussed in a broader survey of available strategies, tools and techniques to analyze and mitigate effects of rational behavior.

We are designing an incentive mechanism for a P2P-based sharing of multimedia files. Our contribution is to incorporate application-specific characteristics into the incentive mechanism in order to improve its performance. We illustrate this approach by analyzing a collaborative file sharing protocol when various incentive mechanisms are implemented. For each such mechanism, we consider the case when it uses the application-specific characteristic, and when it does not. In this publication, we report on our preliminary findings.

## 1. INTRODUCTION

P2P networks as network infrastructure offer many benefits (scalability, redundancy, choice) which permit the design of many compelling applications [1, 2, 3]. The distributed and autonomous nature of entities participating in P2P networks is the source of those benefits, but also of the difficulty with holding individual entities accountable for their actions. And if one is unable to enforce accountability, there is a great scope for a wide variety of actions, including self-interested behavior, which may be detrimental to other participants.

In general, the perception of usefulness of applications deployed over P2P networks will be highly sensitive to actions taken by P2P nodes, or of the robustness of the system in place to mitigate the impact of their unwanted actions or behavior. This perception is influenced by the consequence and likelihood of application failure and in practice this translates into corresponding amounts of resources, and hence money, dedicated to prevent that failure.

Many proposed systems attempt to enforce accountability in order to allow P2P applications to meet some performance or functional criteria [4, 5, 6]. Tackling the issue at the root, some propose to restrict the autonomy of participants or introduce a centralized audit infrastructure.

An active research area that does not impose explicit limits on the distributed and autonomous nature of P2P networks, implements incentive mechanisms which attempt to make certain actions more appealing than others to participants. Systems based on such approach

---

Parts of this work were performed by Andrew Roczniak during a research internship at Bell Labs - Service Infrastructure Research, Alcatel-Lucent, Ottawa, Canada

can be shown to yield good results in certain cases, but it seems unlikely that a one-size-fits-all solution will ever be proposed as it is a given application that tends to dictate what are the most appropriate incentives.

For example, distributed storage and file sharing have in common the need for storage space and bandwidth, but may differ on their relative need for transfer frequency, type of content (rare or popular) and tamper-proof guarantees, all of which will affect what the most effective incentives are. Moreover, many trade-offs can be made within an application class to ensure that entities derive the greatest value from their participation.

In the present paper, we assume that participants' behavior is driven by some degree of self-interest and our investigated solution is grounded in two facts. The first is that the designers of applications based on P2P networks are perfectly well placed to decide how important are the consequences of this behavior, and what are the best strategies to handle it. The second is that the designers can take advantage of application-specific characteristics to define or optimize an incentive mechanism.

The incentive mechanism we are designing is to be deployed for a large group of collaborators sharing large multimedia files. We consider a repetitive situation where a given peer is interested in sharing a file it created with all other participants, which are also interested in downloading it. A file will be made available for download by the source and the transfer process will be initiated roughly at the same time by all participants. All participants want to achieve some goal while preserving their own resources. We consider two possible goals for the source: shortest time interval between making the file available for download and availability of the complete file to all participants; and second, the complete file becomes available to all participants in approximately the same time. We consider one for the remaining peers: fastest download of the complete file.

The utility function of each participant takes into account both whether the goal was achieved and the amount of resources (bandwidth used for upload) that was necessary to achieve it.

A characteristic of this application is that a source may decide which peers is allowed to download its file and when. If the file is composed of individual pieces, then this characteristic holds true for them too. A case when this characteristic is not exploited is when peers make the decision which pieces to request from the source, or if the pieces are allocated by the source randomly. Exploiting this characteristic entails the source making the decision which piece can be downloaded by a peer.

We thus propose to leverage this application-specific characteristic by varying the way the source peer (peer with the complete file) uploads pieces of the file to various peers.

This is achieved by the source assigning equal and disjoint file partitions from which peers must request pieces (source peer directs

the initial phase of the file distribution). By assigning partitions to peers, the source would in fact encourage participants to collaborate in downloading a file. This approach will be compared to the case when peers randomly select which required pieces to request (source peer makes no decisions).

We have developed a lightweight and flexible protocol to download a file directly from the source or by additionally establishing connections with other peers in order to cooperatively complete the file download. A source, moreover, can decide which parts of the file each peer has access to. Depending on the behavioral model that dictates which strategy a peer will follow, the challenge is thus to design an application-specific incentive mechanism that results in participants achieving their respective goals at the lowest cost to themselves.

The rest of the paper is organized as follows. In the next section, we present strategies, formal tools and techniques available to P2P applications' designers wishing to take into account participants' rational behavior. Section 3 describes the protocol we are using to allow sharing of files between participants, as well as a description of our simulation setup. Section 4 presents our preliminary results as well as a discussion on the most appropriate strategy to handle rational behavior given application-specific characteristics. Section 5 concludes and discusses future work.

## 2. STRATEGIES

As outlined in the introduction, the pursuit of self-interest by providers may impact the benefits derived from participating in a P2P application. If the ultimate goal is to be able to balance the application's desired robustness with an acceptable cost, then as a first step, an application specific analysis should be made of the impact of rationality on the benefits to participants. A second step would be to define the trade-off between a solution mitigating this impact and the cost of any implementation. The following strategies are based on the suggestions outlined in [7] and are summarized in Figure 1.

**Ignore it.** This do nothing alternative may be applicable where the cost of handling rationality outweighs potential benefits. Napster and Gnutella, the original file sharing applications, are practical examples of systems ignoring rational behavior while providing tangible benefits to users. Despite their documented shortcomings with respect to this behavior [8], they were instead laid low by an extraneous force and technical progress respectively.

**Eliminate it.** This, in essence, entails encapsulating the responsibility for negotiating and enforcing that services are reliable to an outside system, through an out-of-band mechanism or partial centralization of some aspects [9]. The former can apply to a group of friends or institutions entering into monitored service-level agreements, and is illustrated by the bounded-price mechanism proposed in [10]. This mechanism is deployed in environments with self-interested participants, and is based on contracts negotiated off-line with deviations assumed to be punished by monetary penalties. The second approach may rely on an omniscient and trusted third-party to identify participants and enforce contracts between them as proposed in [11]. This solution ties received services to a promise of participating in the cost of providing other services. The promise is codified in a contract and enforced by an entity that attempts to maximize the collective benefit received by all participants.

**Hide it.** The possibility of a service provider modifying its behavior is greatly reduced if trusted software and/or hardware is employed within P2P Application. In grid computing for example, the issue of behavior conformity, or the expectation that participants

should behave in line with the organization's rules, is addressed [12].

**Avoid it.** Identify and ignore participants deemed to pursue a goal markedly different from that of the rest of the community. As observed in [7], since failure handling techniques in traditional distributed systems differentiate between correct and faulty behavior, it could provide a basis for a mechanism to avoid rational behavior.

**Use it.** Specifically incorporate and leverage the behavior of service providers into the design of the application. The family of techniques that fall under this strategy, differentiate between the types of service providers' behavior. These can be classified [7] as *obedient* if they always follow the mandated protocol or specification irrespective of other considerations; *faulty* if they stop working or act arbitrarily; *rational* if their behavior is driven by attempts to optimize some function based on its knowledge and understanding of the context; and finally, *irrational* if they attempt to optimize some function that is not explicitly modeled in the description of the system. A brief overview of available techniques is presented in the next section.

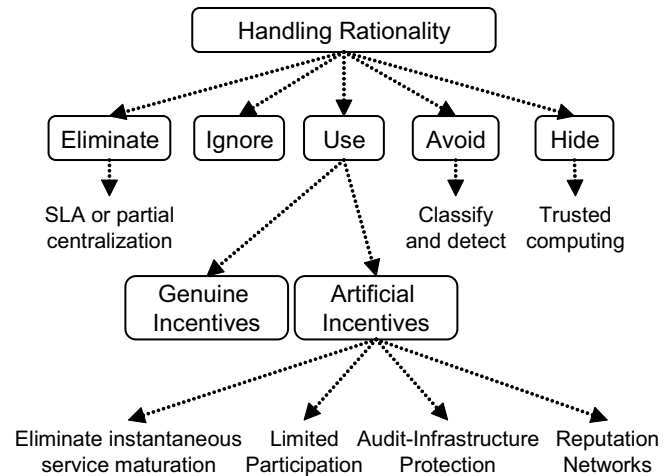


Fig. 1. Handling rationality: strategies and techniques

### 2.1. Leveraging Providers' Behavior

Rational and irrational service providers are said to follow some strategy. The goal, therefore, of leveraging providers' behavior is to "build a mechanism that enables strategizing nodes to act rationally, and incentivize rational nodes to behave well" [7]. Formal tools to analyze and synthesize incentive mechanisms are briefly presented in Section 2.2.

With respect to a behavior that seeks to maximize the usage of system's shared resources (bandwidth in file sharing, for example), [9] provides a taxonomy of "rational attacks", and provides a finer-grained analysis of potential solutions by distinguishing between genuine incentives which are "characterized by directly incentivizing cooperation" and artificial incentives which "incentivize evidence of cooperation" [9].

Reciprocal mechanisms, where the interaction is repetitive and participants do not exactly know when it will end, are an example of genuine incentives. Generally, situations where parties to a contract exchange their obligations in small increments, reduce the possibility of participants not keeping their side of the bargain. In the context of the download of a file, an example of a working reciprocal mechanism is the BitTorrent [13] protocol, even though available

implementations of this protocol may not necessarily be *faithful* to its desired specification [14]. Other examples include systems based on tokens or micro-payments [15]. Artificial incentives are arguably weaker than genuine incentives but correspondingly easier to implement and the following four methods are highlighted.

**Eliminate instantaneous service maturation.** If a service matures instantaneously, i.e., is consumed before any reciprocity is received, then the consumer of such a service can easily take advantage of the producer. Participants' behavior can be influenced by requiring them to leave a *security deposit* or to *pay their dues* [16] before they can become service consumers.

**Limit the number of entities.** If nodes are capable of remembering the outcome of the interaction with other participants, and the interaction is repetitive - a situation which can be modeled as Iterated Prisoner's Dilemma - then [17] argues that assuming all nodes behave rationally, the best available strategy is retaliatory and encourages cooperation.

**Protect the audit infrastructure.** The evidence of cooperation must be protected if participants are to use it to make their decisions. This may be based for example on securing shared history or verification protocols [2].

**Deploy trust and reputation networks.** Whenever a consumer needs to make a selection between comparable services, the decision often hinges on the belief in the accuracy and truthfulness of the claimed quantitative (or qualitative) criteria. This belief can be strengthened by repetitive satisfactory interaction with the provider or based on a record of interaction evaluation made by other entities. In settings with large population of services or entities, it may be difficult or impossible to form an opinion about each member of the population, and thus using others' opinions to guide one's decision quickly becomes an appealing solution.

This approach however presents numerous challenges in practice, and for illustration purposes we highlight three of these [18]. First, the record must be found in a possible distributed environment, second, the precise evaluation methodology must be understood, and finally, a decision must be made as to how believable the evaluator is. Those challenges are usually addressed simultaneously, as illustrated by the following examples.

In the context of P2P file sharing networks, [19] presents an algorithm to aggregate local trust values (number of satisfactory transaction less number of unsatisfactory transactions) into a global reputation value. This is achieved through weighing other participants' reported local trust values by the local trust a node has in that participant. Authors report that this value is obtained relatively fast, which reduces the amount of required overhead traffic.

Noting that trust and reputation are context dependent, multifaceted and dynamic, [20] proposes a Bayesian model for representing the trust in services or recommendations of a specific participant. Authors note that this solution is applicable in cases where there is repetitive interaction between participants, either in small networks or large networks that exhibit small-world characteristics.

The above methods are furthermore sensitive to security issues, from protecting the reputation data and anonymity to identity [21].

## 2.2. Analysis and Synthesis Tools

Interaction between rational participants and impact of deployed incentive schemes are usually analyzed using game theory. On the other hand, synthesis of incentives is usually accomplished through *mechanism design* (MD). In the following, we briefly present salient points of these two tools using an illustrative example of children dividing chocolate bars amongst themselves.

**Game Theory** A "game" is played by "players" using "strategies"

that lead to "outcomes" which give each player some "payoff". Each player has a preference for some outcome (higher payoff) which may be given as a utility function  $u$  which maps outcomes to payoffs,  $u : \mathbb{O} \rightarrow \mathbb{P}$ . Players are rational in the sense that they will strive to maximize the utility function in every situation, i.e., obtain the highest payoff. Players will attempt to achieve that goal by playing a strategy, or taking actions, in response to every possible strategy other players might use. Outcomes are of course the result of this strategic interaction, and might not necessarily be what the players intended or desired.

Incentive mechanisms act by changing the payoff in certain outcomes, which hopefully changes which strategies a player will choose. In this light, some taxes or income deductions legislated by governments might be viewed as incentives to spend, save or invest. Given the "rules" of the game, which may be diverse, game theory allows to analyze or predict results of the interaction. For example specifying how many children and chocolate bars there are, if the interaction recurs regularly and if it is always the same children that interact, and assuming that the utility increases with increasing amount of the chocolate bar, one possible result is that rationality will drive a child to hoard all the chocolate. A set of strategies that lead to the highest payoff for each player may be viewed as a solution to that particular game, or an "equilibrium" upon which the game converges. A Nash equilibrium for example, is the set of strategies where no player can improve its payoff by changing its strategy given other players' strategies. A dominant-strategy equilibrium occurs when all participants have one strategy that is superior to all other available strategies, regardless of what other participants do.

Finite zero-sum games where perfect information - full knowledge of available strategies, payoffs and player preferences to outcomes - are relatively easy to analyze. However, in most non-zero-sum games there are multiple Nash equilibria and not all information may be known by all players which increases the complexity of the analysis.

**Mechanism Design** Mechanism design deals with the design of the rules of the game such that when rational participants with private information (which influences their choice of strategies) interact using those rules, their best choice of strategies lead to an outcome intended by the designer. For instance, if the goal is to have a chocolate bar divided equally between two children, then it can always be achieved when the rules specify that one child cuts the bar and the other subsequently chooses which piece to take<sup>1</sup>.

More formally, a mechanism is defined as a tuple of all strategies  $\Sigma$  available to all participants  $n$  and an outcome function  $f : \Sigma_1 \times \Sigma_2 \times \dots \times \Sigma_n \rightarrow \mathbb{O}$ . The mechanism will be designed for a particular outcome  $o \in \mathbb{O}$  if  $f(s^*) = o$  where  $s^* = (s_1, \dots, s_n)$  is an equilibrium solution to the game. Outcomes can of course be varied, from maximizing sale price in an auction to efficiently allocating load-balancing or routing resources for example. What can be said about this outcome? Of particular interest to designers wishing to optimize utility gained by a group is Pareto efficiency. This predicate indicates if a player can change its strategy for a higher payoff without another player receiving lower payoff, or, in other words, that an alternative outcome exists that is preferred by at least one player while others are indifferent. Mechanisms themselves will have certain (desirable) properties, and for example, mechanisms are Pareto optimal if they implement Pareto efficient outcomes and strategy-proof if players' dominant strategy leads them to the de-

<sup>1</sup>Those actions can be seen as an example of *commitment* in game theory

sired outcome. The situation where strategies followed by players do not naturally lead them to the desired outcome, are handled by a transfer function consisting of some kind of “subsidies” and “taxes” (incentive-compatible mechanism) which make it worthwhile for the player to choose strategies that lead to desired outcomes.

Two main issues underpin any successful implementation of a mechanism. The first, all players must at least obtain as much utility from participating as from not participating in a mechanism (all rational participants would abstain otherwise), and secondly, from a computational perspective, both the output and transfer function must be computed in polynomial time.

### 3. PROTOCOL AND SETUP

In this section we present the salient points of our protocol that allows peers to discover themselves and the file to share, as well as the download process. Further details can be obtained in [22].

A rendez-vous (RV) peer is used to facilitate the startup phase of the download. All participants send a message to the RV peer advertising their presence and wait to discover 8 adverts (7 peers and 1 source) before proceeding with the actual file transfer. This ensures all peers start roughly at the same time. Only the source advertised that it has a file to share, and therefore all the peers attempt to connect to the source and request one piece of the file. As soon as a peer obtains a piece of the shared file, it will issue a content advertisement that contains the piece hash (for identification purposes) as well as the hash of the whole (parent) file. This ensures that peers interested in the file can discover other peers that are currently downloading that file as well. After joining such a *swarm*, peers will exchange state messages and attempt to download pieces from each other. The state message provides a way for peers to exchange information necessary to decide which pieces are available for upload. Once a peer has obtained all the pieces of the original file, it will create a content advertisement for the complete file and publish it at the RV peer. Other peers may however still make requests based on the previously discovered content advertisement for a piece. A message will be sent to the requesting peer notifying it to use the new advertisement instead. The above changes ensure that peers can share incomplete files with other peers.

Each peer will keep track of the pieces it currently owns (successfully downloaded), and of the pieces it has issued a request for. Pieces that are not part of either set and are available at another peer may be included in a request for pieces. The piece selection algorithm implementation uses two variants where pieces are selected randomly, or based on a preassigned partition. The format of the request is similar in both cases, and contains a sequence of numbers  $(i, \dots, k)$  identifying individual pieces. These in turn, are interpreted by the receiving peer as a series of requests for ranges  $(chunk_i, chunk_{i+1}), \dots, (chunk_k, chunk_{k+1})$  where  $i \neq k$ . A time-out is associated with every piece, triggered if a piece has not been received within 20 seconds. That specific piece then becomes a candidate for another request.

As the download progresses, there may be multiple peers offering the same piece, and each peer must keep track of the pieces it already owns,  $O_{own}\{\dots\}$ , and of the pieces currently requested but not downloaded yet,  $D_{downloading}\{\dots\}$ . Each request for pieces is made from selecting some pieces from the set of remaining pieces,  $R_{remain}\{\dots\}$  in order to create  $F_{full}\{f_1\dots f_{full}\}$ . In the random case, a peer  $i$  will send a request for pieces to peer  $j$  by picking randomly up to 15 pieces from  $R_i\{\dots\} \cap O_j\{\dots\}$ , where  $j$  can be another peer or the source. In the partition case, a peer  $i$  will send a request for pieces to the source by picking sequentially up to 15 pieces from

$R_i\{\dots\} \cap P_i\{f_n\dots f_m\}$ , where  $P_i\{f_n\dots f_m\}$  is the partition assigned to peer  $i$  by  $s$ . If  $R_i\{\dots\} \cap P_i\{f_n\dots f_m\} = \emptyset$  then the peer picks sequentially up to 15 pieces from  $R_i\{\dots\} \cap O_s\{\dots\}$ . If the other peer is not the source, then peer  $i$  sends a request for pieces to peer  $j$  by picking sequentially up to 15 pieces from  $R_i\{\dots\} \cap O_j\{\dots\}$ . Partitions of equal size are assigned by the source peer on a first-come, first-served basis, and the source has prior knowledge of the number of participating peers.

All the peers (except the source) attempt to discover new peers offering to share the file every 10 seconds. Once other sources of the file are discovered, peers start to behave as client and server with respect to each other. Peers behave as *client* when requesting pieces from other participants, and as *server* when fulfilling requests for pieces from other participants. Since the source has the complete file, it never behaves as a client and it never receives requests for its state information. Other participants implement exactly the same client behavior. They select up to 4 peers to connect to, chosen from a randomized list of discovered peers. A connected peer is dropped in favor of another peer if it fails to upload pieces in three attempts, either by not having any pieces to upload or if a requested piece is not received within 20 seconds. Upon receiving all the pieces to reconstruct the original file, a client updates its content advertisement and becomes a source.

We have developed our protocol using JXTA and a modified service based on Content Management Service (CMS). The setup consists of 3 bridged LANs, the first providing 100Mbps supports 4 PCs (a.k.a fast peers), the source (peer with the complete file) and the RV peer, the second providing 100Mbps supports 1 fast peer, while the last providing 10Mbps supports 2 PCs (a.k.a slow peers). We ran 5 tests for both the random and partition cases and we collected logs for each machine after each test. We use arithmetic averages over those tests in all cases, unless otherwise specified. The size of the transferred file is 152.5MB, each piece is 64kB, and thus we have 2383 pieces.

### 4. PRELIMINARY RESULTS & DISCUSSION

Our starting case is that all peers behave as they are mandated to do, without attempting to gain an advantage at the cost of other participants. By introducing self-interested behavior, we of course anticipate the performance to decrease. This decrease can be lessened by selecting an appropriate strategy and implementing an incentive mechanism based on the discussion in section 2. We hope to improve the results obtained by implementing this incentive system by using the application-specific characteristic of granting access according to a specified paron, and validating against the random access case.

In this section we describe our selected metrics and report on baseline simulation results, and discuss the most appropriate strategies given the goals of participants. Implementation of the incentive mechanism for both the random and partition cases are currently under development.

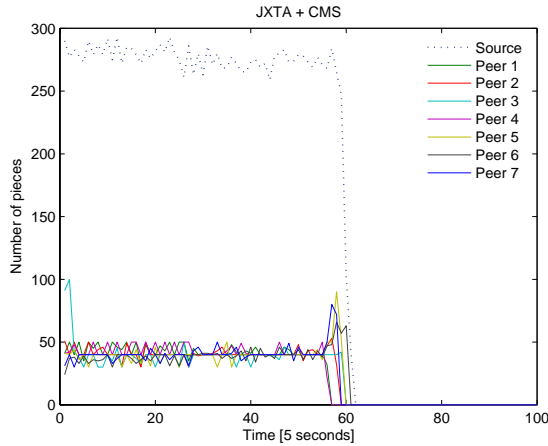
#### 4.1. Primary Metrics

The time necessary to complete a download and its cost, as measured in required upload bandwidth, are a primary metric. The download completes when all the pieces of the original file are received. The following sections present and discuss the obtained results.

##### 4.1.1. Time Required to Complete Download

Results of a simulation where 7 peers download a file only from a source peer is presented in Fig. 2. The graph shows the number of pieces received by each peer and the number of pieces transmitted

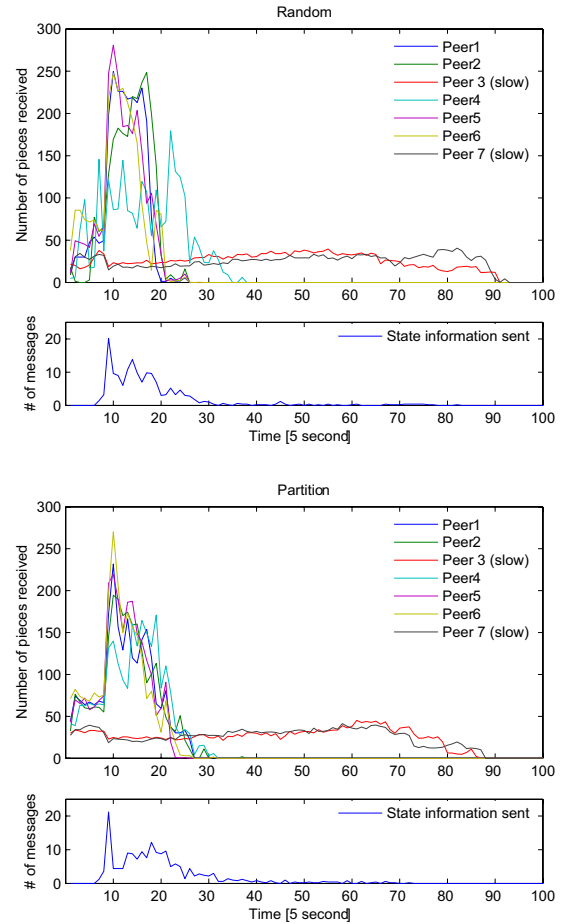
by the source as a function of time, in units of 5 seconds. The communication between peers being TCP-based, the source divides its bandwidth more or less equally among all the peers.



**Fig. 2.** Non-cooperating peers: number of pieces received by peers and sent by the source

Results of simulations where peers can download from each other are shown in Figure 3. Peers request pieces based on the information about which pieces a participant can provide. This information is obtained by requesting a state message. The number of state messages sent as a function of time is shown on the graphs as well.

Participants with fast connections complete downloading in less than 200 seconds (with most finishing around 130 seconds). Participants with slower connections (Peer 3 and 7), complete downloading in less than 450 seconds. The time necessary to complete downloading is roughly the same in both the random and partition cases, although most of the fast peers seem to finish around 20 seconds faster in the random case than in the partition case. Slow peers on the other hand seem to finish around 20 seconds faster in the partition case than in the random case. We have also ran test with only the source and one fast peer, yielding 97 and 81 seconds for random and partition cases respectively. Due to the limited nature of the experiments however, no definitive inference can be drawn, and further study is necessary. In both cases, peers start by downloading from the source, and around 30 seconds later obtain advertisements that allow them to start sharing pieces. This is an artifact of our implementation, where JXTA discovery happens periodically every 10 seconds. The period when peers download only from the source could be reduced by reducing this discovery period. The number of packets peers can obtain from the source will be a function of their own connection speed, and of how many peers the source is serving. After a peer discovers other sources of pieces, the number of pieces received increases for the fast peers and drops for the slow peers. This is expected since fast peers have more download bandwidth than is available to them from the source only, whereas slow peers must now share their limited bandwidth with other peers. In this case, slow peers cannot improve their performance by connecting to multiple peers, and should limit their download connections to less than 4 currently allowed in the simulation.



**Fig. 3.** Time to complete download

#### 4.1.2. Load on the Source and Users

We also look at the sustained load by the source and peers, expressed as the number of pieces transmitted as a function of time, in units of 5 seconds. The sum of the peers' contribution is shown instead of the load on individual peers. This gives us a comparison of the work performed by the source relative to the work performed by other participants and is shown in Figure 4.

We observe that the source provides 6793 and 9268 pieces while the peers provide 10363 and 7840 pieces in random and partition cases respectively. In both cases the number of the pieces transmitted is greater than required for 7 copies (17156 and 17108 against 16681) due to duplicate pieces. In both cases after around 100 seconds, the number of pieces transmitted by the source and peers decreases sharply. In the random case, the contribution by individual peers is roughly equal to the source's contribution reflecting the fact that pieces are by that time widely available. In the partition case however, the source provides relatively more pieces than the peers up until around 200 seconds. Additionally, participants rely relatively more on the source than other peers for pieces, implying that if the source is located on a slow peer, the partition case may cause slower download time than random case.

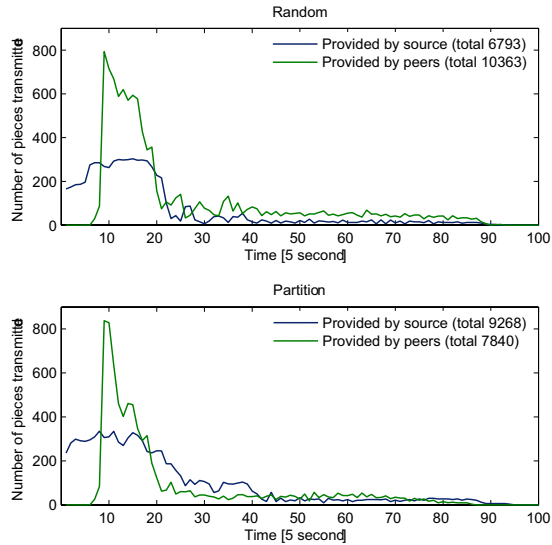


Fig. 4. Load on the source and users

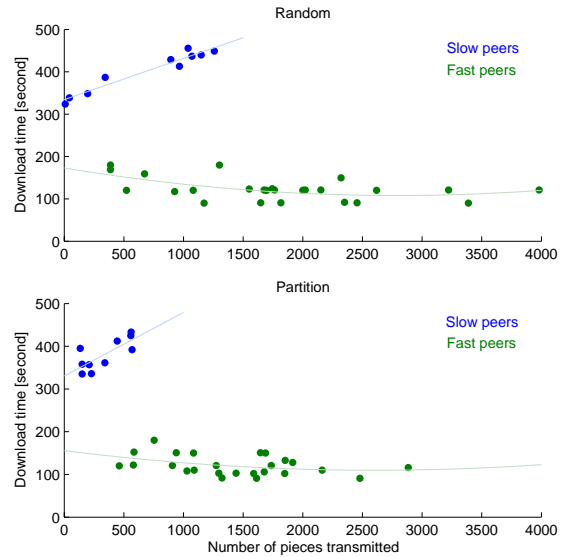


Fig. 5. Cost of sharing

## 4.2. Secondary Metrics

Secondary metrics investigate further the impact of the two ways of controlling which pieces a given peer can download from the source, namely random and partition. The metrics are the apparent cost of sharing and the impact of duplicate pieces on download time. The following sections present and discuss the obtained results.

### 4.2.1. Cost of Sharing

Next, we assess whether there is a relationship between providing pieces (behaving as a server with respect to other peers) and download time (time to receive all the required pieces). The scatter diagram in Figure 5 presents results for each trial (averages are not used), plotting download time as a function of number of pieces uploaded. To highlight the relationship, linear and quadratic fitting is used for slow and fast peers respectively.

Due to the wide bandwidth difference, slow peers fulfilling requests from fast peers will have a marginal effect on the fast peer's performance, but at a substantial cost to themselves. This is clearly shown in the diagram where increasing number of shared pieces increases the time necessary to complete the download. The trend for fast peers is more subtle, and shows a slight decrease of download time as the number of pieces transmitted increases, up to around 2500 pieces where the trend flattens (download time remains constant with increasing number of pieces). The initial decrease can be explained by the fact that by providing less pieces a peer removes upload bandwidth from the system, which increases the download time for all participants. Similarly, the flattening of the curve is consistent with the fact that once a peer finishes downloading the entire file, it starts to behave as a source to other peers. Finally, comparing the random and partition cases reveals broadly the same trends.

### 4.2.2. Impact of Duplicate Pieces

In order to assess the effect of duplicate pieces on the performance, the scatter diagram in Figure 6 presents results for each trial (averages are not used), plotting the number of duplicate pieces received as a function of download time.

Peers keep track of duplicate pieces only when they do not have

the entire file, and ignore (and do not log) any pieces that arrive after it has completed the download. Whenever a requested piece is not downloaded before its timeout occurs, another request will be issued for the same piece. The peer therefore can download multiple copies of the same piece, and only the first downloaded copy is not marked as duplicate. The subsequent request may be issued immediately or at a later time - depending whether the peer requests pieces sequentially or randomly, and if one peer was replaced by another in the list of connected peers. We note that 24 duplicate pieces constitute around 1% of the total file, and that on average peers do not exceed that number. Any impact due to duplicate pieces is therefore bound to be small on download time. Effectively, the fast peers' download time does not show any correlation with the number of duplicate pieces received, while slow peers show only a slight correlation. More interestingly, we note that in the random case, fast peers have more duplicate packets than slow peers (average of around 11 and 6 respectively) while in the partition case, this is reversed (average of 2 and 26 respectively). It is expected that this could have a more important impact on the performance if we were to change the time-out parameter for each piece from its current 20 seconds.

## 4.3. Selecting a Strategy

This section is concerned with discussing which strategy is best suited given the adopted model of peer behavior and participants' goals.

It is clear that if peers behave as they are mandated to do, then the best strategy is to do nothing - ignore rational behavior. Similarly, if the lack of possession of the file does not create a disadvantage with those peers that have it, or does not preclude further interaction, then doing nothing is again most likely the best strategy.

Eliminating rational behavior, or avoiding it, might be applicable in situations where the probability that a large number of peers will act at cross-purposes or that those actions have serious consequences in the context of the application. Especially in this last case however, one must be very sure that a P2P network is the best infrastructure for the given application. A rough sketch of the various strategies is

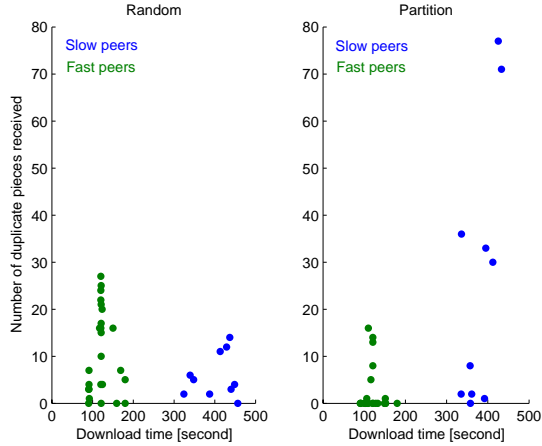


Fig. 6. Number of duplicate pieces

presented in Figure 7.

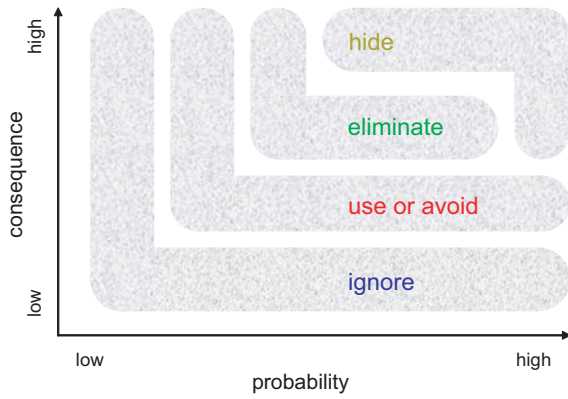


Fig. 7. Strategies given the consequences of application failure and probability of peers executing self-interested actions

In order to keep the simulation created by various strategies tractable, we assume that the strategies in question can only involve the peer selection (with whom a peer will collaborate) and the amount of collaboration (always, reciprocal). As an example of an alternate strategy, consider that a peer can wait until all have completed their download to start downloading itself (advertisements about the downloaded pieces of the file are available). This way, a peer would manage to download a file without uploading anything.

As our simulation results include peers with fast and slow connections, it is readily apparent that if we design for the shortest time required by the slowest download, this would imply that the fast peers will share at least around 2500 pieces (close to the size of the file to share), while the slow peers should not share at all. In the case of the slowest peers therefore, their goals and the goal of the source are perfectly aligned. The implication of this observation is that peers collaborating solely on a reciprocal basis will not yield the best result for the source (see discussion on BitTorrent in Section 4.3.1).

Designing for all peers obtaining the file in roughly same time, presents complications related to the upload bandwidth of the source and the number of peers. As can be seen from Figure 2, the source’s

goal can be achieved if peers do not collaborate. If on the other hand they do collaborate, then slower peers will always finish much later than the fast peers, unless a scheme more involved than random or partition is implemented by the source. In this case therefore, the best strategy seems to be to ignore rational behavior.

Based on the above discussion, the strategies of avoiding and using rational behavior are selected for further study, and include four distinct techniques. Within the strategy of avoiding rational behavior, we are concentrating on reinforcement learning where the environment consists of the source and other peers. Chosen techniques for using rational behavior include building a reputation network using the source as the authority, designing an auction as a means for allocating upload bandwidth between peers while the source’s bandwidth will be used as an extra “payment”, and finally, implementing a BitTorrent-like protocol for collaborative sharing.

Of those four techniques, the last is at the most advance stage. A few points need to be highlighted with respect to the applicability of an unadulterated version of the BitTorrent protocol.

#### 4.3.1. Bit Torrent

A BitTorrent file distribution usually consists of an ordinary web server, a *torrent* file, an active and centralized component *tracker*, an initial client with the full copy of the file to share and a set of downloading clients. A new client starts by downloading a torrent file containing the IP address of the tracker from a website indexing such files. The tracker keeps information about the peers that are currently active and acts as a rendez-vous point for all the clients of the torrent. Active clients periodically report their state to the tracker or when joining or leaving the torrent. Upon joining the torrent, a new client receives from the tracker a list of randomly selected active peers to connect to. Clients involved in a torrent cooperate to replicate the file by exchanging chunks of the file with one another. BitTorrent specifies two main algorithms, the *choke* and *rarest-first*, to make the process of chunk exchange efficient [23]. The choke algorithm encourages cooperation among peers, limits the number of peers a client concurrently exchanges chunks with and selects the best peer (optimistic unchoke), while the *rarest-first* algorithm controls which pieces a client will actually request in the set of pieces currently available for download.

BitTorrent is designed to efficiently replicate a file between a large number of peers that do not necessarily know each other, and in most likelihood will not cooperate on downloading another file again. In the case where the same peers collaborate on transferring multiple files, much of the information peers learn about each other is not used. Furthermore, strict tit-for-tat implementation might not be suitable for the peer wishing to upload the file in all circumstances. Considerable scope therefore for adaptation in special cases exists.

## 5. CONCLUSIONS

With the current trend of using services or resources that do not directly fall under the control of one authority, an entity may need to make adjustments to its requirements to account for rational behavior. Alternatively, an incentive mechanism can be designed and deployed in the P2P network to encourage participants to behave in a manner consistent with the application designers’ expectations. P2P applications designers are therefore the best authority to decide how important this issue is to their applications, and what is the best solution given their goals. To help in these decisions, we have presented a brief survey of state-of-the-art strategies and techniques to mitigate the impact of rational behavior. We have also presented preliminary metrics and results for a lightweight and flexible protocol

to be deployed for a large group of collaborators sharing large multimedia files. The file can be downloaded directly from the source or by additionally establishing connections with other peers in order to cooperatively download the file.

By introducing self-interested behavior, we anticipate the performance of our protocol to decrease. This decrease can be lessened by selecting an appropriate strategy and implementing an incentive mechanism. Based on the goals of the users of the application, we have presented a discussion on most pertinent strategies, and have outlined four most promising techniques for an incentive mechanism implementation. The four techniques are reinforcement learning where the environment consists of the source and other peers, reputation network using the source as the authority, an auction as a means for allocating upload bandwidth between peers while the source's bandwidth will be used as an extra "payment" and a BitTorrent-like protocol. In the context of the selected application, a file to be shared is initially available on one participant only. This source thus has some degree of control over which participants are allowed to download from it, and what they are allowed to download. This application-specific characteristic can be implemented by segmenting the file into multiple partitions, and granting access to specific participants. On the other hand, if participants request and obtain file segments chosen randomly, then the application-specific characteristic is not being utilised. Future work includes modeling the self-interested behavior and implementing the four techniques (reinforcement learning, reputation network, auction and BitTorrent-like protocol) for both the random and partition cases. We hope to improve on the results obtained in the random case by optimising for the application-specific characteristic as implemented by the partition case.

## 6. REFERENCES

- [1] Stephanos Androutsellis-Theotokis and Diomidis Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.
- [2] G. Caronni and M. Waldvogel, "Establishing trust in distributed storage providers.," in *Peer-to-Peer Computing*, 2003, pp. 128–133.
- [3] Ahsan Habib and John Chuang, "Incentive mechanism for peer-to-peer media streaming.," in *IWQoS*. 2004, pp. 171–180, IEEE.
- [4] Panayotis Antoniadis, Costas Courcoubetis, and Robin Mason, "Comparing economic incentives in peer-to-peer networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 46, no. 1, pp. 133–146, 2004.
- [5] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang, "Robust incentive techniques for peer-to-peer networks.," in *ACM Conference on Electronic Commerce*, Jack S. Breese, Joan Feigenbaum, and Margo I. Seltzer, Eds. 2004, pp. 102–111, ACM.
- [6] Thomas Hummel, Øyvind Strømme, and Ryan M. La Salle, "Earning a living among peers - the quest for viable p2p revenue models.," in *36th Hawaii International Conference on System Sciences (HICSS-36 2003)*, 2003, p. 219.
- [7] J. Shneidman and D. C. Parkes, "Rationality and self-interest in peer to peer networks.," in *IPTPS*, M. Frans Kaashoek and Ion Stoica, Eds. 2003, vol. 2735 of *LNCS*, pp. 139–148, Springer.
- [8] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of napster and gnutella hosts," *Multimedia Systems*, vol. 9, no. 2, pp. 170–184, 2003.
- [9] S. J. Nielson, S. Crosby, and D. S. Wallach, "A taxonomy of rational attacks.," in *IPTPS*, Miguel Castro and Robbert van Renesse, Eds. 2005, vol. 3640 of *LNCS*, pp. 36–46, Springer.
- [10] M. Balazinska, H. Balakrishnan, and M. Stonebraker, "Contract-based load management in federated distributed systems.," in *NSDI*. 2004, pp. 197–210, USENIX.
- [11] D. Ghosal, B. K. Poon, and K. Kong, "P2P contracts: a framework for resource and service exchange," *Future Gener. Comput. Syst.*, vol. 21, no. 3, pp. 333–347, 2005.
- [12] W. Mao, F. Yan, and C. Chen, "Daonity: grid security with behaviour conformity from trusted computing," in *STC '06*, New York, NY, USA, 2006, pp. 43–46, ACM Press.
- [13] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, "Influences on cooperation in bittorrent communities," in *P2PECON '05*, New York, NY, USA, 2005, pp. 111–115, ACM Press.
- [14] J. Shneidman, D. Parkes, and L. Massoulié, "Faithfulness in internet algorithms," in *PINS'04*, New York, NY, USA, 2004, pp. 220–227, ACM Press.
- [15] P. Golle, K. Leyton-Brown, and I. Mironov, "Incentives for sharing in peer-to-peer networks.," in *ACM Conf. on Electronic Commerce*. 2001, pp. 264–267, ACM.
- [16] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster, "To share or not to share, an analysis of incentives to contribute in collaborative file sharing environments," in *P2PECON'03, Berkeley, USA*, 2003.
- [17] S. Jun and M. Ahamad, "Incentives in bittorrent induce free riding," in *P2PECON '05*, New York, NY, USA, 2005, pp. 116–121, ACM Press.
- [18] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision.," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.
- [19] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks.," in *Proceedings of WWW2003*, Budapest, Hungary, May 2003, pp. 640–651.
- [20] Y. Wang and J. Vassileva, "Bayesian network trust model in peer-to-peer networks.," in *AP2PC*, Gianluca Moro, Claudio Sartori, and Munindar P. Singh, Eds. 2003, vol. 2872 of *LNCS*, pp. 23–34, Springer.
- [21] B. N. Levine, C. Shields, and N. B. Margolin, "A survey of solutions to the sybil attack," Technical Report 2006-052, University of Massachusetts Amherst, October 2006.
- [22] Andrew Roczniak, Jamil Melhem, Pierre Levy, and Abdulmotaleb El-Saddik, "Design of distributed collaborative application through service aggregation.," in *DS-RT*. 2006, pp. 165–174, IEEE Computer Society.
- [23] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan, "Analyzing and improving a bittorrent network's performance mechanisms," in *IEEE INFOCOM*, 2006.