DogFish: Decentralized Optimistic Game-theoretic FIle SHaring

Seny Kamara[⋆1] and Alptekin Küpçü[⋆⋆2]

[1] Brown University
[2] Koç University

Abstract. Peer-to-peer (p2p) file sharing accounts for the most uplink bandwidth use in the Internet. Therefore, in the past few decades, many solutions tried to come up with better proposals to increase the social welfare of the participants. Social welfare in such systems are categorized generally as average download time or uplink bandwidth utilization. One of the most influential proposals was the BitTorrent. Yet, soonafter studies showed that BitTorrent has several problems that incentivize selfish users to game the system and hence decrease social welfare.
Previous work, unfortunately, did not develop a system that maximizes social welfare in a decentralized manner (without a trusted party getting involved in every exchange), while the proposed strategy and honest piece revelation being the only equilibrium for the rational players. This is what we achieve, by modeling a general class of p2p file sharing systems theoretically, then showing honest piece revelation will help achieve social welfare, and then introducing a new cryptographic primitive, called randomized fair exchange, to instantiate our solution.

Keywords: peer-to-peer file sharing, optimistic fair exchange, proof of storage, cryptographic protocol, game theory.

1 Introduction

The interaction between parties in peer-to-peer (p2p) file sharing networks is strategic and therefore the study of the incentives behind such networks have become an active area of research. The best known and most successful file sharing network BitTorrent, introduced by Cohen [18], accounts for the most uplink bandwidth use in the Internet [50]. File sharing in BitTorrent can be defined as a two-party game where the peers must decide whether or not to trade a block of the file they are trying to download. It was originally believed that the best strategy for the BitTorrent game was to play tit-for-tat, that is, if a peer provides another peer with a block then the second peer should reciprocate and provide the first peer with another block. However, the appearance of clients

⋆ Work done while at Microsoft Research.
⋆⋆ Corresponding author. Work partly done while at Microsoft Research.

like BitThief [40] and BitTyrant [46], which do not play tit-for-tat, has called this assumption into question. In particular, Levin et al. [37] argued that the BitTorrent game is more properly modeled as an auction.

During a BitTorrent round (roughly 10 seconds), to whom a peer connects is decided based on others' uploads to the peer at hand and the pieces they advertise to have. The BitTorrent protocol specifies that the peers should report the pieces they own honestly. Yet, they can over- or under-report the pieces they own, to gain strategical advantage, decreasing social welfare [37,49].

Consider an over-reporting peer. He may be attractive to many other peers since he seems to have something that they do not have; many peers would want to connect to him. But then, during their exchange, they may not be able to obtain anything useful from him, whereas he may obtain many useful blocks. As for an under-reporting peer, as observed before, the peer may gain some strategic advantage against BitTorrent's rarest piece first heuristic [37]. Moreover, if many peers are under-reporting, the system would face starvation [49].

In this paper we address this issue for the first time, as we present a BitTorrent-like p2p file sharing mechanism that incentivizes honest piece revelation, hence increasing social welfare, while working in a decentralized manner without any trusted party involvement per block exchange, and is an equilibrium for rational players. We achieve this goal in a simplified theoretical setting, through the use of novel cryptographic techniques, and a game-theoretic approach to p2p file sharing systems that encompasses different aspects of a protocol such as peer matching and block exchange. We do not claim that our protocol would replace BitTorrent in practice; rather we trust that our theoretical solution insight would help prominent researchers develop both theoretically-sound and practically-applicable protocols for this goal. Our contributions are:

- We propose the first theoretical *decentralized* file sharing protocol (with no trusted party involvement per exchange) that is an *equilibrium* with respect to realistic utility functions (i.e., the number of pieces downloaded).
- To achieve this we introduce and construct a new cryptographic functionality which we refer to as *randomized fair exchange* (RFE). We provide a security definition for RFE and a construction.
- While the use of RFE is enough to disincentivize under-reporting, it does not prevent over-reporting. We then combine RFE with *proofs of storage* to ensure over-reporting is also discouraged.
- Finally, we show that under a simple theoretical model, our solution achieves an equilibrium with the best possible *social welfare* (defined as upload bandwidth utilization) among a large class of protocols that first match peers based on their reported pieces, and then perform pair-wise exchanges.

2 Preliminaries

File sharing protocols. In the BitTorrent protocol files are divided into pieces, and pieces are divided into blocks (see [18] and [3] for details). Exchange takes

[3] http://bittorrent.com

place in terms of blocks. Only blocks of finished pieces are exchanged. A *torrent* file contains hashes of the pieces so that they can be verified to be the correct ones. There are two types of peers in the system: those who already have all the pieces of the torrent at hand (the ***seeders***), and those who do *not* have the complete set of pieces and are still downloading (the ***leechers***).

When a new peer joins the system, she obtains a *peerset* (a list of seeders and leechers) from the Tracker (which is a central entity, but does not get involved in exchanges) for the torrent she wants to download, and starts forming peer-to-peer connections with those peers. They announce her the pieces they have. Based on those announcements, she picks a subset of those peers to actually perform block exchanges with. In BitTorrent, usually a peer would pick 4 peers who have given her the most blocks in the previous round (e.g., past 10 seconds), and 1 more random peer (totaling 5 peers) to give a block to. Those exchanges are not necessarily fair [46,40]. Each time she obtains a new piece, she reports it to her peerset, which may not be done honestly either [37,49].

Throughout the paper, we simplify our discussion by using piece and block interchangeably, since we are describing a BitTorrent-like protocol and not necessarily BitTorrent itself. In our case, we only consider blocks in a torrent, where the hash of each one of them is known from the torrent, and the blocks can be exchanged as soon as they are downloaded.

Basic Game Theory. In an n-player game, we have players $P_1, \ldots, P_n$, where player P_i has m possible actions $A_i^1, \ldots, A_i^m$ and a real-valued utility function u_i. The utility u_i of player P_i is a function of an action profile A such that A specifies one action $A_j^{\ell_j}$ for every player P_j. The goal of the players is to obtain the highest possible utility.

Strategy s_i of player P_i is a probability distribution over the possible actions of player P_i. Let $s_i(a)$ denote the probability the strategy s_i assigns to the action a. Let $s = \{s_1, \ldots, s_n\}$ denote a strategy profile encompassing the strategies of every player. Further denote by $s_{-i} = s - \{s_i\}$ the strategy profile of all players *except* the player P_i. Given the strategies of each player, we can define expected utility U_i for player P_i as $U_i(s) = \Sigma_{A \in s} \left(u_i(A) * \Pi_{j=1}^{n} Pr[s_j(A_j)]\right)$.

A strategy profile $s = \{s_1, \ldots, s_n\}$ is **a Nash equilibrium** if $\forall P_i \quad \forall s_i' \neq s_i \quad U_i(s_i \cup s_{-i}) \geq U_i(s_i' \cup s_{-i})$. A strategy profile s is **a strict Nash equilibrium** if the inequality is strict.

In a **computational setting**, s is considered a **computational Nash equilibrium** if there is a negligible function $neg(k)$ in the security parameter k such that $\forall P_i \quad \forall s_i' \neq s_i \quad U_i(s_i \cup s_{-i}) \geq U_i(s_i' \cup s_{-i}) - neg(k)$. This allows the strategy s to be negligibly worse than the other alternatives. Note that strictness is not important here, since there would be another negligible function that makes the inequality strict. Furthermore, we require that the actions of players can be implemented in polynomial time using probabilistic Turing machines (**PPT actions**, in short). ϵ-Nash equilibrium is a generalization where the negligible function is replaced with some other ϵ.

Proofs of Storage. Efficient proofs of storage (PoS) were introduced in 2007, independently by Ateniese et al. [4] and Juels and Kaliski [28]. Later on, Ateniese

et al. [5] and Dodis et al. [19] generalized these constructions, and Erway et al. [21] and Cash et al. [15] made them work with dynamic data for the first time.

In these constructions, the general idea is that a file is divided into blocks, then cryptographic tags are created for each block. In a regular PoS scenario, the client creates those tags, and outsources the file and the tags to the server for storage. Later, the client or some third party auditor can challenge the server to prove that her file is kept intact. The server sends back a short proof, which is verified using the associated PoS (public) key.

When applied to a p2p file sharing system, we consider PoS as follows: The creator of the torrent creates those tags, and hence each block is associated with a tag in the torrent file. Then, when peers advertise blocks, they can prove to each other that they indeed have the actual blocks they claim to have. Thus, the client role belongs to the torrent creator, and the auditor and the server roles are played by all the peers. Moreover, since the torrent contents are static once created, static proofs of storage with public verifiability (meaning anyone can verify integrity using public information) are enough (e.g., [4,51]).

Fair Exchange. In a fair exchange scenario, there are two parties Alice and Bob, where each has an item e_A and e_B, respectively, that they want to exchange. At the end of the protocol, it must be the case that either Alice obtains e_B *and* Bob obtains e_A, or neither of them obtains anything useful about the other party's item. It is known that fair exchange requires a trusted third party called the *Arbiter* [45], but optimistic protocols employ this Arbiter only when a problem occurs [3]. Previously, two-party fair exchange protocols were used in the BitTorrent setting [10,36], but assuming honest piece revelation.

Multi-party fair exchange protocols achieve similar fairness guarantees for not only two but possibly more parties [31]. When there are multiple parties, we talk about exchange topologies. For example, if we have players $P_1, \ldots, P_n$ in a *ring topology*, this means each P_i will send one item to $P_{(i+1) \mod n}$. We combine two-party fair exchange protocols with coin tossing to obtain a new primitive called *randomized fair exchange*.

Notation. Each player P_i has a set $\mathbf{S}_i$ of blocks she already holds, and a set $\mathbf{R}_i$ of blocks that she reports to have. Note that dishonest reporting means $\mathbf{S}_i \neq \mathbf{R}_i$. The notation $e \xleftarrow{\$} \mathbf{S}_i$ means that an element e is picked randomly from the set $\mathbf{S}_i$. Sets are bold fonted.

We denote by $(\text{OUT}_A; \text{OUT}_B) \leftarrow \Pi_{A,B}(X; Y)$ the execution of a two-party protocol Π between parties A and B, where X and Y are the inputs provided by and OUT_A and OUT_B are the outputs returned to A and B, respectively. This notation can be extended to multi-party protocols as well.

3 Related Work

Prisoners' Dilemma. Prisoners' Dilemma is considered by many as the underlying game of the BitTorrent protocol. Interestingly, even though tit-for-tat (TFT) is *not* a game-theoretic solution to the *finitely-repeated* Prisoners' Dilemma, Axelrod's experiments [6,7] show that it achieves the best results in overall utility

gained. Later on, Fader and Hauser [22] published experiments on multi-player version of the Generalized Prisoners' Dilemma, and argued that coalitions of mutually-cooperating players will emerge implicitly, and win the tournaments.

Radner [48] allows ϵ-departures from strict rationality, and shows that as the number of repetitions increase in the finitely-repeated Prisoners' Dilemma, the ϵ-Nash equilibria allow for longer periods of collaboration in the Prisoners' Dilemma. Later, Neyman [43] shows that if the players are limited to polynomial-sized finite state automata, then cooperation in the Prisoners' Dilemma will give a Nash equilibrium. Unfortunately, this result does not hold in the Turing machine model and hence we cannot conclude a computational equilibrium using probabilistic polynomial time players. Halpern [27] argues that if memory has a positive cost, then it acts as a discount factor and rationalizes TFT strategy even for finitely-repeated Prisoners' Dilemma.

If one thinks of BitTorrent as an unboundedly-repeated (or incomplete-information) Prisoners' Dilemma, then TFT can be seen as an equilibrium. Ellison [20] shows this is the case in *anonymous random-matching* scenarios. Feldman et al. [25] use Generalized Prisoners' Dilemma to model p2p networks.

Honest Piece Reporting. Levin et al. [37] were the first to describe BitTorrent as an auction rather than a TFT game. They show that under-reporting of pieces constitute a problem and it leads to a tragedy of commons scenario if all peers under-report. Unfortunately, BitTorrent does *not* enforce honest piece reporting. prTorrent [49] also presents, via simulations, that this under-reporting may cause starvation in BitTorrent. They also show that piece rarity, which may be manipulated via under-reporting, can indeed be modeled game-theoretically as a discount parameter in a repeated game such as BitTorrent.

Arteconi et al. [2] consider evolutionary p2p protocols where peers randomly compare their utilities and replicate better actions. They consider honest reporting as the main issue, and analyze effects of dishonest reporting.

Luo et al. [41] define a multi-player game for sharing one piece, rather than a two-player game, to model BitTorrent-like file sharing systems, and propose a utility-based piece selection strategy. They define the concept of the marginal utility of a piece; a concept directly related to the under-reporting strategy described in the papers above. They also argue that as the number of peers increase, each peer is more likely to *not* report the piece that he owns.

Since under-reporting may change the attractiveness of a peer (and rareness of blocks), Guo et al. [26] and Okumuşoğlu et al. [44] argue that not all blocks should be equally valuable, and propose value-based variants of BitTorrent.

Game-Theoretic File Sharing. In light of the game-theoretic limitations of BitTorrent, alternative p2p file sharing protocols were proposed, including BAR-B [1], Equicast [30], FOX [38], and Ratfish [8]. Unfortunately, all these works have limitations. For example, BAR-B and FOX only handle static sets of users. The Equicast protocol is only an equilibrium under strong assumptions (e.g., the rate of leechers that join has to equal the rate of leechers that leave) and for restricted utility functions which do not appear to model the real utilities of users. The Ratfish protocol, while being an equilibrium with respect to realistic player

utilities, is a partly centralized solution where the trusted Tracker is involved in *every* exchange, which is undesirable in p2p settings. Similarly, Vilaça and Rodrigues [53] assume a trusted mediator. See [17] for a survey.

Another limitation of all these previous works is that none of these works show their solution's performance against a social choice function for p2p file sharing. As such, it is not clear how to evaluate these mechanisms or how to compare them. We also do not compare ourselves against solutions that employ monetary compensation, such as [54,52,10], or social network based reputation solutions [16], whose incentives for honest piece revelation are not clearly analyzed. See [29] for a survey of monetary incentives.

Theoretical Optimum. Fan et al. [24] define a performance metric (using the average download time) and a fairness metric (using upload vs. download bandwidth) and compare the original BitTorrent with some parameter-modified versions of it. They prove in their technical report [23] that when fairness is enforced, the average download time *increases*, slowing down the system.

Meng et al. [42] define a theoretical lower-bound for the distribution time of a file using BitTorrent-like systems. They use the *fluid model*, which allows each bit to be shared as soon as it is received, unlike BitTorrent that shares blocks once pieces are received. Yet, interestingly, Kumar and Ross [33] show experimentally that the error between the piece-based model's minimum download time and the fluid model's is less than 1%. Indeed, they claim that the difference between the two models can be safely ignored as long as the number of pieces in the file is much larger than the logarithm of the number of leechers, which is true for medium-sized or large files that we generally encounter in such systems.

4 Model

Our goal is to create a system maximizing social welfare in the equilibrium. We assume a homogeneous network, and analyze the protocol in rounds, where **at each round each peer can download at most one piece and upload at most one piece**. This makes our theoretical analysis easier. It also means that there is no bandwidth-based auction as in PropShare [37]. Instead, the matching between pairs will be done based on the mutual attractiveness in terms of the pieces a peer is missing. This allows us to focus on *piece revelation strategies* rather than bandwidth allocation issues.

Also note that **seeders are irrational** and altruistic entities, and there is no piece-revelation strategy for them. In our game-theoretic analysis, therefore, we keep them outside the discussion, since they are irrational. But, we allow seeders to help the system perform better by still distributing blocks.

A more realistic analysis would model a heterogeneous network, where there are several types of players with different upload/download capabilities (but the protocol may still proceed in rounds). Moreover, our analysis only partly covers malicious irrational entities: while the cryptographic protocols employed prevent malicious actions, our incentive mechanism only works against rational entities. Thus, malicious entities may still over- or under-report the pieces they own.

	1	2	3	...	m
P_1	1	1	1	1	1
P_2	1	1	1	1	1
P_3	1	1	1	1	1
...	1	1	1	1	1
P_n	1	1	1	1	1

	1	2	3	...	m	m+1	m+2	m+3	...	m+n-1
P_1	1	1	1	1	1	0	0	0	0	0
P_2	0	1	1	1	1	1	0	0	0	0
P_3	0	0	1	1	1	1	1	0	0	0
...	0	0	0	1	1	1	1	1	1	0
P_n	0	0	0	1	1	1	1	1	1	1

Table 1. Matrix representation of number of pieces received by players at each round. Rows are leechers. Columns are rounds.

5 Social Welfare for P2P File Sharing

In this section, we first define social welfare, then relate it to honest piece revelation for a general class of p2p file sharing protocols. This general class of protocols we consider incorporates two subprotocols: a peer matching protocol that pairs the peers, and a pair-wise block exchange protocol. This accurately models BitTorrent-like p2p file sharing systems restricted to a round-based exchange model. At each round, first peers are matched in pairs (remember our restriction that within one round only one block can be uploaded or downloaded), and then the exchange takes place. In the upcoming sections, we instantiate those subprotocols and show that they achieve the desired social welfare.

Social Welfare. When one considers the social welfare for a p2p file sharing system, there can be several metrics. One of the commonly used metrics in the literature is the *average download time*. Another good measure of an efficient system is indicated as the *utilization of the upload bandwidth* [11], since it ensures the system performs at its best in terms of distributing the file.

Let us denote a protocol as a matrix where *rows* denote the *parties* and the *columns* denote the *rounds*. Each cell i, t denotes the probability that peer P_i obtains some new block at round t (alternatively, it can denote the expected number of blocks downloaded). The socially optimal protocol would be the one where the cells are all 1 until the round m for every peer, where m is the number of blocks in the file. The average download time would then be $m * n/n = m$, which is optimal in our model. Moreover, the upload/download bandwidths will be fully utilized. This corresponds to the left side matrix in Table 1.

Realize that such a protocol may *not* always be achievable. Consider, for example, a **single seeder flash crowd** scenario. In round 1, only one peer can obtain some piece from the seeder, and all other peers will obtain 0 blocks. One socially optimal (and deterministic) protocol here would be the following: The seeder, at round t, sends piece number t to peer P_1. At the same round, each P_i sends piece $t - i$ to P_{i+1}, if she already has that piece, and stays idle otherwise. Thus, peer P_i finishes downloading at round $m + i - 1$.[4] This corresponds to

[4] For simplicity, representing common behavior, assume each P_i leaves the system the moment she finishes downloading (at the end of round $m + i - 1$). Afterward, at round $m + i$, the seeder sends the last block to peer P_{i+1} (thus every peer receives the last block from the seeder).

Algorithm 1 Pairwise P2P File Sharing Protocol

while some peer is still downloading **do**
  $(P_{j_1}; P_{j_2}; ...; P_{j_n}) \leftarrow \mathcal{F}^{MATCH}_{P_1,P_2,...,P_n}(\mathbf{R}_1; \mathbf{R}_2; ...; \mathbf{R}_n)$
  // pair-wise such that if the output to P_i is P_{j_i} then the output to P_{j_i} is P_i
  for $i = 1$ to n **do**
    $(e_{k_i}; e_{k_{j_i}}) \leftarrow \mathcal{F}^{EXCH}_{P_i,P_{j_i}}(\mathbf{S}_i; \mathbf{S}_{j_i})$
    // actually, $n/2$ exchanges take place because of pair-wise matching and our round-based model
  end for
end while

the right side matrix in Table 1. Note that the upload bandwidths are again optimized to the best possible (except P_n).

Therefore, when we talk about socially optimal protocols, we cannot just talk about making every entry in the associated matrix 1. Instead, we need the following two properties to optimize the upload bandwidth in our model:

1) For every peer P_i, P_i needs to be matched with some interesting P_j (if such P_j exists). Here, interesting means P_j has some piece P_i does not have.
2) For every P_i, P_i needs to be able to download a new piece (assuming she is matched with some interesting P_j above).

Note that we want protocols that incentivize the behavior above. Observe that 1) needs to hold for *all* peers, and 2) requires P_j to have an incentive to send a new piece to P_i. One can create a global incentive mechanism for that, but it would not be very practical. Consider the single seeder flash crowd scenario above. In that deterministic protocol, no peer P_i has an incentive to send a piece to peer P_{i+1}. To enforce the protocol, one may employ a ring-topology fair exchange protocol that ensures either the whole ring completes, or no peer can receive a new piece from the previous one (including from the seeder). Unfortunately, this necessitates per round communication complexity that is *quadratic in the number of peers* [31].

Pairwise Protocols. Hence, we concentrate on local incentive mechanisms, and in particular, **pair-wise** ones. If we match *mutually interesting* peers and perform a fair exchange between them, then we incentivize the desired behavior using only simple, constant complexity two-party fair exchanges instead of a global multi-party fair exchange mechanism that is costly. Thus, we consider protocols of the following type, where $\mathcal{F}^{MATCH}$ denotes the functionality to match the peers, and $\mathcal{F}^{EXCH}$ denotes the exchange functionality: First peers are matched pair-wise according to their piece revelations, and then pair-wise exchanges take place within the same round. This is depicted in Algorithm 1.

Such protocols assume that the exchanges occur between pairs of peers. Due to our simplification that one piece can be exchanged per round, such a simplified version of BitTorrent would also fit the framework above. As observed before, it may be impossible to reach the social optimum with such a protocol, but on the other hand, global matching and exchange protocols would be impractically inefficient. Therefore, we choose to restrict ourselves to pairwise matching and

pairwise exchange protocols. As discussed, we know that to obtain the best possible social welfare (upload bandwidth utilization) in this restricted setting, we need to incentivize honest piece revelation and maximize the exchanges between the peers to obtain as many 1 values in the corresponding exchange matrix as possible. In the following sections, we first assume that every peer is matched with some interesting peer according to their piece revelations and show how to perform the $\mathcal{F}^{EXCH}$ phase by instantiating it via randomized fair exchange and proofs of storage. This is where our main contribution lies. Then, we finalize our DogFish protocol description by also instantiating the $\mathcal{F}^{MATCH}$ protocol via existing known solutions and finalizing our game-theoretic analysis.

6 $\mathcal{F}^{EXCH}$ Instantiation

We instantiate our $\mathcal{F}^{EXCH}$ functionality using randomized fair exchange (a primitive that we introduce and construct) together with proofs of storage. We first define these individual building blocks, and then provide our instantiation and its analysis. Throughout this section, we assume that $\mathcal{F}^{MATCH}$ is already completed matching mutually-interesting peers, and we concentrate on performing pair-wise exchanges during $\mathcal{F}^{EXCH}$. In practice, leechers may also get matched with seeders, but remember that seeder interactions are outside our game-theoretical scope, and hence in our protocols, we only deal with exchanges between two leechers.

6.1 Randomized Fair Exchange (RFE)

In regular fair exchange protocols, two parties exchange items such that at the end of the protocol, either both parties obtain each other's item, or neither party obtains anything useful [3]. RFE allows two parties to exchange elements from their sets of items in such a way that each party receives a *new* element *at random*. The RFE functionality is formally described in Fig. 1 and a construction is provided later.

RFE is a crucial building block of the DogFish protocol, and is used to instantiate the $\mathcal{F}^{EXCH}$ functionality. Intuitively, we want to prevent an adversarial user to strategically pick the pieces to download, hence breaking fairness and deviating the system away from social welfare for the sake of his selfish utility.

The $\mathcal{F}^{RFE}$ Functionality

– Upon receiving set $\mathbf{X}_1$ from P_1 and set $\mathbf{X}_2$ from P_2:
  1. if $\mathbf{X}_2 \setminus \mathbf{X}_1 = \emptyset$ or $\mathbf{X}_1 \setminus \mathbf{X}_2 = \emptyset$, sent $\perp$ to both parties.
  2. else, pick random $e_2 \xleftarrow{\$} \mathbf{X}_2 \setminus \mathbf{X}_1$ and $e_1 \xleftarrow{\$} \mathbf{X}_1 \setminus \mathbf{X}_2$ and send e_2 to P_1 and e_1 to P_2.

Fig. 1. The randomized fair exchange functionality.

The $\mathcal{F}^{PoS}$ Functionality

– Upon receiving the actual blocks $\mathbf{S}_1$ and their tags from P_1, and the claimed blocks $\mathbf{R}_1$ and the verification key from P_2:
  1. if $\mathbf{S}_1$ does not contain all the claimed blocks in $\mathbf{R}_1$, return REJECT to P_2;
  2. else, if all the blocks in $\mathbf{R}_1$ match their corresponding tags and the tags verify using the verification key, send ACCEPT to P_2.
  3. else, send REJECT to P_2.

Fig. 2. The proof of storage functionality in the p2p file sharing setting.

When blocks to be exchanged are picked randomly by the functionality, as we will show later, the adversary loses any advantage gained by under-reporting. Consider a player who under-reported the pieces he owns. During RFE, it is possible that he will receive a piece that he already owns, hence gaining no utility from the exchange, and in general, decreasing his expected utility. On the contrary, honestly-reporting players, who were matched with mutually-interesting peers, are expected to gain positive utility in RFE (more details later). We instantiate $\mathcal{F}^{RFE}$ using fair exchange and coin tossing protocols in Section 7.

6.2 Proofs of Storage (PoS)

Another building block for our $\mathcal{F}^{EXCH}$ functionality is a proof of storage protocol. We present the protocol as adapted to the p2p file sharing setting. Remember that the creator of the torrent file constructed the tags and put them in the torrent, together with the public key that will be used for verification purposes (thus, we only consider PoS schemes without secret keys used during challenge verification [34]). Also, we just need static PoS solutions, since torrent contents never change. Thus, such a protocol may be instantiated via, for example, [51] or [4] with only *constant* communication cost.

Realize that if P_1 over-reported during the matching phase, meaning that he does not have all the blocks he claimed in $\mathbf{R}_1$, then P_2 will obtain a rejection signal. Similarly, if P_1 tries to use fake blocks, P_2 will reject. PoS, on the other hand, does not prevent under-reporting, since if one can prove storage of more blocks, he could also prove storage of fewer blocks. Remember though the RFE functionality discouraged under-reporting. PoS functionality discourages over-reporting. In the next section, we combine them to achieve our pair-wise exchange functionality. It is worth noting that $\mathcal{F}^{PoS}$ discourages over-reporting for P_1 only. But for our system to be an equilibrium, this must apply to all parties. Hence, we will employ two executions of $\mathcal{F}^{PoS}$ to discourage both parties.

6.3 $\mathcal{F}^{EXCH}$ and its Analysis

We first instantiate our pair-wise exchange protocol using the RFE and PoS functionalities described. The $\mathcal{F}^{EXCH}$ functionality instantiation is shown in

The $\mathcal{F}^{EXCH}$ Functionality Instantiation

- P_1 provides his blocks $\mathbf{S}_1$ and their PoS tags, the block identifiers $\mathbf{R}_2$ that P_2 claimed to know, and the PoS verification key.
- P_2 provides her blocks $\mathbf{S}_2$ and their PoS tags, the block identifiers $\mathbf{R}_1$ that P_1 claimed to know, and the PoS verification key.
  1. Run $\mathcal{F}^{PoS}$. If P_2 receives REJECT at the end, send $\perp$ to both parties and abort.
  2. Run $\mathcal{F}^{PoS}$ again, but with the roles of P_1 and P_2 reversed. If P_1 receives REJECT at the end, send $\perp$ to both parties and abort.
  3. If still not aborted, run $\mathcal{F}^{RFE}$.

Fig. 3. The pair-wise exchange functionality instantiation.

Peer i / j	Honest Reporting	Under Reporting	Over Reporting
Honest Reporting	1 , 1	1 , < 1	0 , 0
Under Reporting	< 1 , 1	< 1 , < 1	0 , 0
Over Reporting	0 , 0	0 , 0	0 , 0

Table 2. Two-player game during $\mathcal{F}^{EXCH}$ assuming peers are matched based on mutual interest. < 1 denotes some utility strictly less than one.

Figure 3. The idea is that, to be able to exchange blocks, peers must also prove to each other that they possess the blocks that they claim to own.

Game-theoretic Analysis. Since our simplified model only considers exchanging one block per round between peers, the best utility each pair of matched participants can obtain in one round in our analysis is $(1, 1)$ (both peers can obtain 1 block at the end of the exchange). Assuming the matching was done among mutually-interesting peers, the utilities of the matched peers using $\mathcal{F}^{EXCH}$ are described in Table 2. It is impossible assign a number for the "< 1" parts, since it depends on the runtime values of the sets, and the sheer number of possibilities make the analysis impractical; but it is also unnecessary. **In the exchange game, there is only one strict Nash equilibrium, and that is when both matched peers honestly reported the pieces they own.**[5]

The intuition is that the **PoS makes it irrational for any rational party to over-report**, because if she does, then it will be detected during the PoS stage and the exchange will be aborted, resulting in zero utility for the over-reporter as well.PoS, however, does not deter under-reporting because it cannot detect that a party has more blocks than it claims to have. But **under-reporting is**

[5] More precisely, the zeros in the game should be replaced with negligible utilities (of managing to break PoS security), < 1 values should be $< 1 - 1/m$ (one minus non-negligible, where the file has m blocks), and ones should be one minus negligible (due to the negligible probability of the fair exchange failing). Overall, we chose not to complicate the presentation with these details, but indeed our equilibrium is a *computational Nash equilibrium.*

The $\mathcal{F}^{COIN}$ Functionality

– Upon receiving integer v from P_1 and integer v' from P_2:
  1. if $v \neq v'$ or $v = v' = 0$, return $\perp$ to both parties.
  2. else, pick random integer $r \xleftarrow{\$} [1, v]$ and send it to both parties.

Fig. 4. The coin tossing functionality.

handled by the RFE protocol, because if a player under-reports, then she has a non-zero probability of receiving a block she already has, whereas if she reports honestly she will receive a block she does not have with certainty.

7 RFE Instantiation

To realize randomized fair exchange, we employ unfair coin tossing protocols together with regular two-party fair exchange protocols. [6]

7.1 Coin Tossing

Figure 4 shows the coin tossing (COIN) functionality, where the coin is picked from an agreed-upon range. For our purposes, even though we model the functionality fairly, it is enough to instantiate via an unfair protocol where only one party learns the result, and is supposed to send that to the other party [12,9]. Note that, coin tossing protocols generally employ commitments, and hence one cannot cheat at the resulting r, but can only prevent the other party from learning r (and hence outputting $\perp$).

7.2 (non-randomized) Fair Exchange

As explained before, fair exchange is a simple functionality which requires the existence of a trusted third party [45]. But, in *optimistic* fair exchange scenarios, this trusted Arbiter does not get involved in every exchange [3,35]. This is the main disadvantage of the RatFish protocol, where the trusted Tracker must get involved in every exchange [8]. Figure 5 shows the fair exchange (FEX) functionality.

We focus on fair exchange of blocks, where the hash found in the torrent file enables checking that the block is authentic. In our case PoS tags serve the same purpose as hash values, and hence they can also be employed to check for correctness of the blocks exchanged, using existing instantiations [3,36].

[6] RFE is also related to oblivious transfer [47]. Indeed, at first, we were imagining a randomized oblivious fair exchange would be necessary, but it turns out we do not need obliviousness for the game theoretic analysis to go through.

The $\mathcal{F}^{FEX}$ Functionality

- Upon receiving item i_1 and hash h_2 from P_1 and item i_2 and hash h_1 from P_2:
  1. if $hash(i_1) \neq h_1$ or $hash(i_2) \neq h_2$, return $\perp$ to both parties.
  2. else, send i_2 to P_1 and i_1 to P_2.

Fig. 5. The fair exchange functionality.

The $\mathcal{F}^{RFE}$ Functionality Instantiation

- P_1 calculates the set $\mathbf{F}_1 = \mathbf{S}_1 - \mathbf{R}_2$ of blocks that he has but P_2 claims not to have. Note that if P_1 honestly reported, $\mathbf{S}_1 = \mathbf{R}_1$ and hence $\mathbf{F}_1 = \mathbf{R}_1 - \mathbf{R}_2$. Then, P_1 sets $v_1 = |\mathbf{F}_1|$.
- P_1 also calculates the set $\mathbf{F}_2 = \mathbf{R}_2 - \mathbf{S}_1$ of blocks that he does not have but P_2 claims to have. Again if P_1 honestly reported, we have $\mathbf{F}_2 = \mathbf{R}_2 - \mathbf{R}_1$. Then, P_1 sets $v_2 = |\mathbf{F}_2|$.
- P_2 computes the sets $\mathbf{F}'_1 = \mathbf{R}_1 - \mathbf{S}_2$ and $\mathbf{F}'_2 = \mathbf{S}_2 - \mathbf{R}_1$ (similarly, for honest P_2 we have $\mathbf{F}'_1 = \mathbf{R}_1 - \mathbf{R}_2$ and $\mathbf{F}'_2 = \mathbf{R}_2 - \mathbf{R}_1$). Then, P_2 sets $v'_1 = |\mathbf{F}'_1|$ and $v'_2 = |\mathbf{F}'_2|$.
  1. Run $\mathcal{F}^{COIN}$ where P_1 inputs v_1 and P_2 inputs v'_1, and they both obtain r_1. If any party obtains $\perp$ instead, abort.
  2. Run $\mathcal{F}^{COIN}$ again where P_1 inputs v_2 and P_2 inputs v'_2, and they both obtain r_2. If any party obtains $\perp$ instead, abort.
  3. Run $\mathcal{F}^{FEX}$ where the input of P_1 is the r_1^{th} element of $\mathbf{F}_1$ together with the hash/tag in the torrent file for the r_2^{th} element of $\mathbf{F}_2$. Similarly, the input of P_2 is the r_2^{th} element of $\mathbf{F}'_2$ together with the hash/tag in the torrent file for the r_1^{th} element of $\mathbf{F}'_1$.

Fig. 6. The randomized fair exchange functionality instantiation.

7.3 Randomized Fair Exchange Protocol

The idea is similar to our $\mathcal{F}^{EXCH}$ functionality in the sense that we repeat unidirectional protocols both ways. Hence, in $\mathcal{F}^{RFE}$ we execute coin tossing twice, and then perform fair exchange on those random items. Remember that the items' correctness are guaranteed via PoS tags during $\mathcal{F}^{FEX}$. The protocol is depicted in Figure 6.

Observe that while this functionality checks that the number of different blocks claimed by both parties are the same, this does not immediately prevent under- or over-reporting. For example, a party may under-report one piece and over-report another piece, such that the size of the difference remains the same. In general, we do not even need to ensure the sizes of set differences match, since PoS protocols within $\mathcal{F}^{EXCH}$ ensure that over-reporting is prevented, and randomization in $\mathcal{F}^{RFE}$ ensures that under-reporting is disincentivized. Finally, observe that $\mathcal{F}^{RFE}$ has *constant* communication and round complexity. (Further note that a parallel coin tossing protocol can be employed as well [39].)

8 The DogFish Protocol

Initialization. As DogFish is a variant of a BitTorrent-like p2p file-sharing protocol, we assume the existence of an external mechanism that enables the parties to find the swarm for a given file $\mathbf{f}$. In BitTorrent, this duty is handled by trackers and hence we assume the same. Moreover, to be able to use the PoS, we assume that the owner of the file encodes it with a PoS which yields a set of tags. We assume that the torrent file contains those tags, as well as any public information necessary to verify those tags (i.e., PoS verification key). Finally, we start our description assuming the users already downloaded the torrent file and contacted the tracker, thereby obtaining the list of other peers and the PoS tags and verification key.

Matching Phase. DogFish is a peer-to-peer file sharing protocol that is pairwise as in Algorithm 1. At the first phase of the protocol, for the $\mathcal{F}^{MATCH}$ peer matching phase, we need to employ some existing *mutual* matching protocol. The only requirement in the $\mathcal{F}^{MATCH}$ phase is that it matches peers if and only if they both reported some block that the other does not possess. Thus, if P_i and P_j are matched, both $\mathbf{R}_i$–$\mathbf{R}_j$ and $\mathbf{R}_j$–$\mathbf{R}_i$ should be non-empty. Remember that we treat irrational seeders separately, and they can get matched even though they are not interested in the other peer. Moreover, as many pairs as possible should be matched for getting closer to the social optimum.

For achieving this, we are faced with two alternatives: Existing BitTorrent papers assume that the $\mathcal{F}^{MATCH}$ protocol of BitTorrent matches mutually interesting peers (both parties have a piece that the other does not possess) through the rarest-first heuristic [11]. Therefore, we can simply employ the BitTorrent $\mathcal{F}^{MATCH}$ protocol. Alternatively, we can we use a distributed stable matching protocol [13,14,32] where parties share their reported blocks $\mathbf{R}_i$ and jointly compute a mutual matching in a distributed manner. As long as the same $\mathbf{R}_i$ values are used during $\mathcal{F}^{MATCH}$ and $\mathcal{F}^{EXCH}$ phases by rational peers, any cheating attempt during $\mathcal{F}^{MATCH}$ will be penalized during $\mathcal{F}^{EXCH}$. Thus, via one of these alternative methods, we assume DogFish obtains a matching of mutually interesting peers at the end of $\mathcal{F}^{MATCH}$. Observe furthermore that priotization (e.g., via rarest-first heuristic) does not affect our game theoretic analysis, as long as $\mathcal{F}^{MATCH}$ matches as many mutually interested peers as possible.

Exchange Phase. Once the matching is done, the round proceeds with the $\mathcal{F}^{EXCH}$ phase where we use the RFE and PoS protocols. Consider the two types of dishonest reporting: over-reporting and under-reporting.

1. If a peer P_i under-reports, her probability of getting a useful block (a block that she does not already possess) goes down (compared to honest reporting) because:
   - While the effect of under-reporting in the matching phase is unclear, she is potentially less likely to be matched. This is because while $\mathbf{R}_j$–$\mathbf{R}_i$ is now potentially larger due to under-reporting, $\mathbf{R}_i$–$\mathbf{R}_j$ is getting smaller, and hence P_i is potentially less interesting for other peers. But, this does not affect our analysis, as we show below.

- Even when P_i gets matched with some P_j, because random blocks are picked from $\mathbf{R}_i$–$\mathbf{R}_j$ and $\mathbf{R}_j$–$\mathbf{R}_i$ during RFE, it is possible that she gets some block that she under-reported (meaning that she already had the block but reported it as missing). Note that if she honestly reported instead, she was guaranteed to get a useful block after the exchange phase (assuming P_j was honest).

2. If a peer P_i over-reports, her probability of getting a useful block again goes down because:
   - Note that the effect of over-reporting during the matching phase is unclear. While over-reporting may create a larger $\mathbf{R}_i$–$\mathbf{R}_j$ and hence makes P_i more interesting for P_j, the set $\mathbf{R}_j$–$\mathbf{R}_i$ is potentially getting smaller, decreasing the chance of a mutual matching. A history-based peer selection may help here, by penalizing previous cheating attempts of a peer, but again with random matchings, the effect is unclear. Fortunately, as before, this does not affect our analysis.
   - Regardless of how matching is affected, when P_i gets matched with some P_j, over-reporting will be caught during the PoS part of the exchange phase (except with negligible probability), and hence she will not obtain any useful block. Note that if she honestly reported instead, she was guaranteed to get a useful block after $\mathcal{F}^{EXCH}$ (assuming P_j was honest).

The discussion above makes one thing clear: Even when $\mathcal{F}^{MATCH}$ can be gamed, our $\mathcal{F}^{EXCH}$ instantiation ensures honest piece revelation as its equilibrium, as long as $\mathcal{F}^{MATCH}$ always matches all *mutually-interesting* peers. When peers are matched based on mutual interest, our exchange protocol provides enough incentive to act honestly, even during the matching phase. Realize that, as long as all mutually interesting peers are matched, for example via distributed stable matching, we do not need a multi-round analysis. This is because under- or over-reporting in one round does not provide any particular advantage in the matching phase afterward: The main constraint in the matching phase is that each matched peer has at least one block the other peer does not have.

After each exchange, when the new round begins, the peers again advertise their blocks during matching. Since honest piece revelation is the rational thing to do, all rational peers will advertise honestly, including the new piece they obtained in the last round. The protocol proceeds this way for every round, as long as there are at least two leechers. Peers may join or leave the system freely.

In summary, by matching mutually-interesting pairs during $\mathcal{F}^{MATCH}$ (e.g., via the BitTorrent rarest-first mutual matching protocol [11] or via some distributed mutual matching protocol such as stable matching) and making sure both parties obtain the best possible utility during $\mathcal{F}^{EXCH}$ (via RFE and PoS), we maximize the utilization of the upload bandwidth, and hence obtain social welfare. Essentially, we maximized the entries in the matrix representation of the piece exchanges (among pairwise p2p file sharing protocols). Moreover, the DogFish mechanism, and hence honest piece revelation, is the equilibrium in a game where utilities of players are defined as the number of pieces they download in a round. Note that in contrast to RatFish [8], we only need trackers at the

beginning, and the rest of the protocol is purely peer-to-peer and decentralized (the Arbiter in our RFE instantiation gets involved *only* if there is a dispute during the fair exchange). This makes DogFish the only known decentralized equilibrium protocol for p2p file sharing achieving social welfare among a large class of protocols under realistic utility functions.

We constructed and analyzed DogFish as a theoretical proposal, hoping that prominent researchers will improve it to be practical. At the current stage, while the cryptographic protocols employed in DogFish are efficient computationally, the round complexity makes the proposal theoretical. For $\mathcal{F}^{MATCH}$, practical solution would be the existing BitTorrent rarest-first mutual matching protocol, since distributed stable matching is costly. For $\mathcal{F}^{EXCH}$, all sub-protocols (coin tossing, fair exchange, and proof of storage) are known to have $O(1)$ computational and communication costs for both parties (usually measured with milliseconds and kilobytes), with $O(1)$ rounds of interaction. But, while $O(1)$, each message passing round increases the total latency, which is another metric to optimize against in practice, and is not considered within our theoretical model. As future work, better $\mathcal{F}^{MATCH}$ protocols should be developed, number of rounds in $\mathcal{F}^{EXCH}$ should be optimized (potentially employing protocols that can be securely parallelized), and an analysis under a heterogeneous model should be conducted.

Acknowledgements

The authors acknowledge the support of TÜBİTAK, the Scientific and Technological Research Council of Turkey, under project number 111E019, as well as European Union COST Action IC1306.

References

1. A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. Bar fault tolerance for cooperative services. *ACM SIGOPS Operating Systems Review*, 39(5):45–58, 2005.
2. S. Arteconi, D. Hales, and O. Babaoglu. Greedy cheating liars and the fools who believe them. In *ESOA*, 2006.
3. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Selected Areas in Communications*, 18:591–610, 2000.
4. G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song. Remote data checking using provable data possession. *ACM Transactions on Information and System Security*, 14(1):12:1–12:34, 2011.
5. G. Ateniese, S. Kamara, and J. Katz. Proofs of storage from homomorphic identification protocols. In *ASIACRYPT*, 2009.
6. R. Axelrod. Effective choice in the prisoner's dilemma. *Journal of Conflict Resolution*, 24(1):3–25, 1980.
7. R. Axelrod. More effective choice in the prisoner's dilemma. *Journal of Conflict Resolution*, 24(3):379–403, 1980.

8. M. Backes, O. Ciobotaru, and A. Krohmer. Ratfish: a file sharing protocol provably secure against rational users. In *ESORICS*, 2010.
9. B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *IEEE FOCS*, 2002.
10. M. Belenkiy, M. Chase, C. Erway, J. Jannotti, A. Küpçü, A. Lysyanskaya, and E. Rachlin. Making p2p accountable without losing privacy. In *ACM WPES*, 2007.
11. R. M. Berciu. Designing incentives in p2p systems. Master's thesis, Baylor University, 2013.
12. M. Blum. Coin flipping by telephone: A protocol for solving impossible problems. In *CRYPTO*, 1981.
13. I. Brito and P. Meseguer. Distributed stable matching problems. In *CP*, 2005.
14. I. Brito and P. Meseguer. Distributed stable matching problems with ties and incomplete lists. In *CP*, 2006.
15. D. Cash, A. Küpçü, and D. Wichs. Dynamic proofs of retrievability via oblivious ram. *Journal of Cryptology*, 30(1):22–57, 2017.
16. K. Chen, H. Shen, K. Sapra, and G. Liu. A social network based reputation system for cooperative p2p file sharing. *IEEE Transactions on Parallel and Distributed Systems*, 26(8):2140–2153, Aug 2015.
17. G. Ciccarelli and R. L. Cigno. Collusion in peer-to-peer systems. *Computer Networks*, 55(15):3517–3532, 2011.
18. B. Cohen. Incentives build robustness in bittorrent. In *WEPS*, 2003.
19. Y. Dodis, S. Vadhan, and D. Wichs. Proofs of retrievability via hardness amplification. In *TCC*, 2009.
20. G. Ellison. Cooperation in the prisoner's dilemma with anonymous random matching. *The Review of Economic Studies*, 61(3):567–588, 1994.
21. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. *ACM Transactions on Information and System Security*, 17(4), 2015.
22. P. S. Fader and J. R. Hauser. Implicit coalitions in a generalized prisoner's dilemma. *Journal of Conflict Resolution*, 32(3):553–582, 1988.
23. B. Fan, D.-m. Chiu, and J. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. Technical report, The Chinese University of Hong Kong, 2006.
24. B. Fan, D.-m. Chiu, and J. C. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *IEEE ICNP*, 2006.
25. M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM EC*, 2004.
26. D. Guo, Y.-K. Kwok, and X. Jin. Valuation of information and the associated overpayment problem in peer-to-peer systems. *Computer Communications*, 80:59–71, 2016.
27. J. Y. Halpern. Beyond nash equilibrium: Solution concepts for the 21st century. In *ACM PODC*, 2008.
28. A. Juels and B. S. Kaliski. PORs: Proofs of retrievability for large files. In *ACM CCS*, 2007.
29. I. A. Kash, E. J. Friedman, and J. Y. Halpern. An equilibrium analysis of scrip systems. *ACM Trans. Econ. Comput.*, 3(3):13:1–13:32, June 2015.
30. I. Keidar, R. Melamed, and A. Orda. Equicast: scalable multicast with selfish users. *Computer Networks*, 53(13):2373–2386, 2009.
31. H. Kılınç and A. Küpçü. Optimally efficient multi-party fair exchange and fair secure multi-party computation. In *CT-RSA*, 2015.
32. A. Kipnis and B. Patt-Shamir. A note on distributed stable matching. In *IEEE ICDCS*, 2009.

33. R. Kumar and K. W. Ross. Peer-assisted file distribution: The minimum distribution time. In *IEEE HOTWEB*, 2006.
34. A. Küpçü. Official arbitration with secure cloud storage application. *The Computer Journal*, 58(4):831–852, 2015.
35. A. Küpçü and A. Lysyanskaya. Optimistic fair exchange with multiple arbiters. In *ESORICS*, 2010.
36. A. Küpçü and A. Lysyanskaya. Usable optimistic fair exchange. *Computer Networks*, 56:50–63, 2012.
37. D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. *ACM SIGCOMM Comput. Commun. Rev.*, 38(4):243–254, 2008.
38. D. Levin, R. Sherwood, and B. Bhattacharjee. Fair file swarming with fox. In *IPTPS*, 2006.
39. Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, 2003.
40. T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in bittorrent is cheap. In *HotNets*, 2006.
41. J. Luo, B. Xiao, K. Bu, and S. Zhou. Understanding and improving piece-related algorithms in the bittorrent protocol. *IEEE Transactions on Parallel and Distributed Systems*, 24(12):2526 – 2537, 2013.
42. X. Meng, P.-S. Tsang, and K.-S. Lui. Analysis of distribution time of multiple files in a p2p network. *Computer Networks*, 57(15):2900–2915, 2013.
43. A. Neyman. Bounded complexity justifies cooperation in the finitely repeated prisoners' dilemma. *Economics Letters*, 19(3):227–229, 1985.
44. O. Okumuşoğlu, M. F. Bayraktar, and A. Küpçü. Justtorrent: Value based-fairer and faster protocols for p2p file sharing. *International Journal of Engineering Science and Application*, 1(1):1–10, 2017.
45. H. Pagnia and F. C. Gartner. On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt University of Technology TUD-BS-1999-02, 1999.
46. M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent. In *NSDI*, 2007.
47. M. O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Harvard Aiken Computation Laboratory Technical Report TR-81, 1981.
48. R. Radner. Can bounded rationality resolve the prisoner's dilemma? *Essays in honor of Gerard Debreu*, pages 387–399, 1986.
49. S. D. Roy and W. Zeng. prtorrent: on establishment of piece rarity in the bittorrent unchoking algorithm. In *IEEE P2P*, 2009.
50. Sandvine. Global internet phenemona, December 2015.
51. H. Shacham and B. Waters. Compact proofs of retrievability. *Journal of Cryptology*, 26(3):442–483, 2013.
52. M. Sirivianos, X. Yang, and S. Jarecki. Robust and efficient incentives for cooperative content distribution. *IEEE/ACM Transactions on Networking*, 17(6):1766–1779, 2009.
53. X. Vilaça and L. Rodrigues. On the range of equilibria utilities of a repeated epidemic dissemination game with a mediator. In *ACM ICDCN*, 2015.
54. V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In *P2PECON*, 2003.