Dealing with Cheaters in Anonymous Peer-to-Peer Networks

Paul Gauthier, Brian Bershad, and Steven D. Gribble
University of Washington
{gauthier,bershad,gribble}@cs.washington.edu

Technical Report 04-01-03
January 15, 2004

Abstract

As anonymous peer-to-peer file sharing networks transition from intellectual curiosity to societal reality, their long-term viability is seriously threatened by cheaters. A cheater either consumes resources without producing them (a freeloader), or advertises valuable content, but ultimately delivers that which is useless (a spoofer). In both cases, the cheater realizes some benefit from his actions without having to pay a commensurate cost. Because these networks are anonymous, the traditional accountability mechanisms developed for classic distributed systems do not apply.

In this paper we present a protocol that dramatically reduces and in many cases eliminates the benefit gained by cheaters in anonymous peer-to-peer file sharing networks. Our protocol is based on the notion of exchange: instead of allowing users to unidirectionally download content, in order to acquire a file, a user must simultaneously provide a file to somebody else. We organize users into "exchange groups," in which each user provides one file in order to acquire one file, with the aggregate exchange satisfying all participants.

Through exposition we show that our composite protocol works well in theory, eliminating the incentive to freeload and forcing spoofers to spend resources commensurate with the damage they cause. Through trace-driven simulation, we show that it works well in practice, resulting in a system in which users can acquire the content they want with reasonable delay.

1 Introduction

In the last few years, peer-to-peer file sharing networks have come into widespread use, attracting over 200 million users [22]. However, *cheaters* currently threaten the usability of these networks, and in the future may threaten their viability. One type of cheater is the *freeloader*, an individual who consumes more content than he contributes, in the limit always consuming and never contributing. The freeloader's intent is to acquire content without having to produce any, since producing content costs bandwidth with no direct gain. Another type of cheater is the *content spoofer*, someone who advertises one piece of content, but in the end delivers another. The spoofer's intent is to prevent the distribution of legitimate content by leveraging the network's "viral" properties, by causing unwitting users to further spread a spoofed file after downloading it. This would allow the spoofer to broadly spread bogus content without having to pay its actual distribution cost.

Today's peer-to-peer networks are largely anonymous, which exacerbates cheating. Participants can create as many identities as they wish, and there is no trusted authority that can vouch for, track, or authenticate identities. The ability to shed an old identity and create a new one without cost makes it impossible to hold users accountable for their actions over time.

This paper describes a practical file sharing protocol for anonymous peer-to-peer networks that at once deals with freeloaders and spoofers. The *Pretty Fair Exchange* (*PFE*) protocol presented in this paper ensures that a user must upload in order to download. Moreover, the protocol permits a downloader to preemptively prematurely abort an exchange if he is dissatisfied with the content (e.g., it is being spoofed), before having paid the bandwidth opportunity cost of the entire exchange. This early detection forces a spoofer to pay the transport cost of each spoofed bit, as it denies the spoofer the bandwidth-amplifying effects of viral distribution. The protocol requires no central authority, nor any notion of identity. Using trace-driven simulation, with traces drawn from a population of 25,000 peer-to-peer users over a six month period, we show that our protocol functions well in practice. In effect, *PFE* enables a sustainable cooperative system [6] in which only honest behavior is rewarded.

1.1 Our Motivation

Like the Web ten years ago, anonymous peer-to-peer networks (ap2p) have crossed the boundary from curiosity to reality in today's Internet fabric. There are dozens of unique ap2p networks in use today, the most active of which has tens of millions of users on-line offering to share tens of petabytes of content at any given time [22]. In such networks, one anonymous user offers to share content with others by making the content available for download. Content is shared when another anonymous user requests (by name) content that is offered for upload. Once downloaded, the receiving user in turn is expected to make that content available to other users, thereby increasing its availability.

These systems effectively make two critical assumptions about their users:

- *Users are altruistic,* voluntarily contributing upload bandwidth proportional to their consumed download bandwidth.
- *Users are honest,* truthfully advertising and delivering authentic content.

The first assumption intends to ensure that the aggregate bandwidth and storage capacity of the network scales with the number of users. The second intends to ensure that a user who “paid the price” (in time and bandwidth) to download content receives the benefit.

Unfortunately, as is often the case when greed and deceit have no immediate local consequences, these assumptions are not bearing out in practice. Studies have shown that most ap2p users are freeloaders, always downloading but never uploading [1]. With respect to honesty, in recent times some have begun to inject bogus content into the network with the intent of diverting users away from the true content [26].

Today’s peer-to-peer users are becoming increasingly aware of how cheaters impact the quality of the network. Gradually, the network “slows down” as more and more downloaders are served by relatively fewer uploaders. In addition, a user looking for content that has been spoofed may be forced to download, inspect, and discard bogus content many times in his search for the real content. Such a process frustrates users *and* places an additional load on the relatively decreasing set of uploaders. In the end, the network will consist only of spoofers serving up bogus content, as participants, including freeloaders, abandon it for another system. An ap2p network may ultimately be destroyed by the dual cancers of greed and deceit.

1.2 Models that Work

Fortunately, the real world offers many examples of sustainable, anonymous peer-based exchange systems. The local swap meet, a barter-based marketplace, functions as a pure ap2p network. An anonymous seller, offering an array of goods, for example cows, is approached by an anonymous potential buyer. The buyer is able to inspect the cows before delivering something of value, for example chickens, to the seller, and before slaughtering the cow only to discover that it is somehow diseased. Moreover, the seller is able to easily inspect the buyer’s offering for legitimacy (e.g., counting the chickens) before releasing his cows.

In this simple example, the buyer and seller directly satisfy one another’s requirements for value, yielding a two-way fair exchange. Freeloading cannot occur, since both the buyer and seller must produce something of value in order to receive something of value: the immediacy and symmetry of the exchange remove any need for altruism. Spoofing cannot occur, since both parties can inspect the goods and walk away if unsatisfied. Consequently, honesty is naturally encouraged because dishonest behavior is immediately observed and met with no reward. The exchange, and its legitimacy, are entirely centered around the goods transacted.

Barter introduces the challenge of matching up buyers and sellers. In the simplest case, where each of a pair of participants wants what the other is offering, the matching problem amounts to nothing more than shouting across the courtyard. More generally, though, it becomes necessary to find a group of buyers and sellers who, between them, completely satisfy one another’s needs. In so doing, a transaction can be conducted by a group in a single round.

1.3 Our Approach

The protocol we describe in this paper, *PFE*, follows the classic model of the anonymous marketplace described above. To download an object, a participant must also offer an object sought by another for uploading. Transactions occur in rounds, with each round resulting in the formation of a group of participants whose collective desires are mutually satisfiable. The objects are incrementally self-verifiable so that a receiver can determine early in the transaction that an object is bogus before the transaction completes. Because a participant must offer something to get something, freeloaders are eliminated. Because a participant can determine if he is receiving spoofed content and may thus abort the transaction, the incentive to spoof is eliminated, ultimately eliminating spoofers.

PFE achieves two properties that today’s ap2p networks do not. Specifically:

- **Fairness.** PFE eliminates freeloaders by ensuring that a user may download no more than he uploads. In PFE, users acquire content by participating in group-wise exchanges instead of unidirectional transfers, providing content in order to acquire content.
- **Proportional damage.** PFE forces a spoofer to directly pay the transfer cost for every spoofed bit prior to its detection. This greatly reduces the realized value of large-scale spoofing. Proportional

damage is achieved as a consequence of early detection, since a user determines early that an object is bogus and will not act as an amplifier for spoofed content.

PFE achieves these properties while retaining the existing properties of ap2p networks. Namely, it maintains the anonymity of users and does not introduce any trusted third parties.

As illustrated by the example of the swap meet, the largest question that a system employing *PFE* faces will be: can the protocol succeed in grouping users so that, within the group, the offerings of one can be satisfied by the needs of another? The successful deployment of *PFE* therefore requires the system to have, in practice, a third property:

- **Liveness.** Changing the basic operation of an ap2p system from unidirectional transfer to groupwise exchange means that users' interests must align for the system to be sustainable: a user wanting an object must at the same time offer an object wanted by another.

Using trace-driven simulation, we show that the interests of today's file sharing users are well-aligned with the requirements of liveness. We demonstrate that enough exchanges to perpetuate the system can occur using relatively small groups. For example, over 93% of transfers can be satisfied by using groups of five or fewer members. Such small groups further make it difficult for a cheater to interfere with the progress of honest users. We also show that it is possible to find exchange groups even if the population size is small. This means that a large population can be partitioned into many subsets while forming groups, vastly simplifying the "matchmaking" process.

This paper makes three contributions. First, it presents a protocol, *PFE*, that defeats cheaters in ap2p networks by changing the fundamental primitive provided by an ap2p network from *download* to *download-while-uploading*. Second, it compares PFE to alternative approaches, including existing fair-exchange protocols, and it analyzes the shortcomings of these other approaches in light of the properties of ap2p networks. Third, and finally, using traces drawn from an actual ap2p network, it shows how well the protocol works in practice.

1.4 The Rest of This Paper

In the rest of this paper, we present *PFE* in more detail. In the next section we provide additional insight into the motivation of cheaters. In Section 3 we present alternative approaches and discuss their limitations. In Section 4 we present the PFE protocol. In Section 5 we present results of trace driven simulations which demonstrate the protocol's liveness in practice. Finally, in Section 6 we summarize and conclude.

2 The Motivation to Cheat

In this section, we consider in greater detail why a user might cheat in an ap2p network. Fundamentally, we have four types of cheaters:

- **Cheap freeloaders:** the cheap freeloader seeks to obtain content with the minimal possible cost, valuing his upload bandwidth more than his altruism. In current ap2p systems, the cheap freeloader is common [1].

- **Poor freeloaders:** the poor freeloader seeks to obtain content, is willing to exchange valid content for it, but has no valid content to exchange. Poor freeloaders do not exist in current ap2p systems, since there is no notion of exchange.

- **Protective spoofers:** a protective spoofer seeks to make it difficult for users to obtain a specific piece of content. To do so, the protective spoofer may advertise a spoofed copy of that content in the hope of attracting users away from the valid content. Protective spoofers may be willing to spend significant resources to accomplish their task. In today's ap2p systems, a protective spoofer can exploit the lack of integrity checking in systems to amplify his attack through viral propagation.

- **Malicious spoofers:** the malicious spoofer is an irrational user that seeks to damage as many transfers as possible, either to make it difficult for users to complete transfers, or to cause users to waste bandwidth. A malicious spoofer is likely to be constrained in the amount they are willing to invest in order to create trouble for others, and seeks to maximize disruption with minimal expended resources.

From the standpoint of the cheater, his actions have one of two effects. He causes "bandwidth damage' when he forces a victim to spend download bandwidth without receiving valid content. He gains a "content advantage" when he obtains valid content without contributing any upload bandwidth.

To be effective at dealing with cheaters in ap2p networks, a protocol must combat both content advantage and bandwidth damage at the same time. The freeloader loses his content advantage as soon as he is forced to spend upload bandwidth to receive content.

In general, it is impossible to eliminate all bandwidth damage from the Internet, where messages can be arbitrarily directed. Consequently, it is more reasonable to expect that damage should be proportional to the cost of creating it. An undesirable property would be to permit a single incident of bandwidth damage to be amplified through the unwitting participation of other parties (as could happen when content is not verified prior to acceptance).

3 Related Work

Malicious and greedy users have plagued shared computing systems for decades. Over time, several broad strategies have emerged to eliminate or contain their effects. We now discuss the strengths and weaknesses of these strategies as they relate to anonymous peer-to-peer file sharing systems.

Identify the offenders, and punish them. The simplest strategy for dealing with offenses such as spoofing or freeloading is to identify the perpetrators and punish them. To do this, the actions of a participant must be irrefutably tied to the identity of that participant so that misbehavior can be identified, and punishment meted out.

We often rely on centralized or hierarchical cryptographic authentication schemes, such as Kerberos [29] and public key infrastructures [9], to provide strong identity. Privacy concerns mean that participants may resist having a permanent identity associated with their actions, especially if that identity is tied to their real-life identity. Moreover, these schemes ultimately depend on a single, trusted root authority to generate new identities and attest to their authenticity. Accordingly, systems that employ them have a single point of failure. The systems may also suffer from scalability problems if the number or growth rate of active identities is large. Although decentralized authentication schemes exist (e.g., the PGP web of trust [33]), the lack of a single mutually trusted authority makes it difficult for strangers to trust in each others' purported identity.

Reputation systems [16] provide an alternative to directly identifying and punishing offenders. These systems indirectly reward a participant for good behavior and punish them for bad behavior by publishing a reputation metric that other participants can influence. For example, Ebay [19] allows users to add or subtract from the reputation of other users with whom they have engaged in transactions: users with poor reputations are presumably shunned. Similarly, Kazaa [22] rewards users that upload content or simply offer many high-quality files by increasing their "participation level": users with higher participation levels are given higher download priorities.

Unfortunately, a dedicated cheater can defeat a reputation system. If users can create new identities without cost (the Sybil attack [17]), they can invent many identities that artificially inflate each others' reputation. Alternatively, a user can simply abandon a tarnished identity and create a new one from scratch.

Make it expensive to misbehave. Rather than punishing offenders for past offenses, some systems make it monetarily expensive to misbehave. In these systems, a user must spend currency to receive service. In return for providing service, a user receives currency. The currency in these systems may be backed by real-world currency (such as in Netbill [12] or Chaumian ecash [10]), or it may be a fictitious, internal unit of currency that is useless outside the scope of the system (such as in Mojonation [25]). Unlike barter systems, currency systems don't suffer from the problem of matching users' wants and offered goods, since money is a good that everybody wants.

Electronic currency systems suffer from four problems: counterfeiting, high transaction costs, double spending, and inflation. Counterfeiting can be countered through the introduction of a centralized, trusted authority that mints and authenticates electronic coins. However, such systems create problems similar to those of centralized authentication schemes. High transaction costs may one day be eliminated through the use of micropayments [21], but at present a standardized protocol with widespread commercial and governmental support has not yet emerged, limiting adoption. Double spending can be combated by either requiring that coins be reconciled against centralized accounts as they are spent, or the use of identity schemes in which offenders' identities are revealed when they double-spend. Fundamentally, both solutions are plagued with the same problems found in central authentication schemes.

The phenomenon of inflation is relatively new in computer systems, and occurs whenever parties are able to assert value without having to prove it. For example, Mojonation [25] is an ap2p file sharing network that credits users for uploading. Unfortunately, Mojonation credits uploaders based on attestations from downloaders: after a successful transfer, the downloader attests that the uploader should be rewarded with credit. Because these attestations could not be validated in practice, attackers can simply create identities that would attest to transfers which never occurred, in effect creating money for nothing.

Explicitly verify important properties. Currency and fair exchange systems can prevent freeloaders, but they do not prevent spoofing. In fact, currency may increase the incentive to spoof if spoofers receive compensation for spoofed content. To defeat spoofers, users must be able to verify the integrity of content they download. Systems designers typically rely on cryptographic hashing to provide integrity. For example, many distributed file systems and file sharing systems ensure integrity by mandating that the name of a file (or a file block) should include a cryptographic hash of its content [2, 11, 23, 15, 18, 31].

Several researchers have proposed using peer-to-peer networks to provide a cooperative backup service [13, 14, 24]. Spoofing is much more insidious in backup systems than in file sharing systems, as users must continually re-verify the integrity and availability of their backed-up content arbitrarily far into the future. In file sharing, integrity only needs to be verified once, at the time a transfer takes place.

Align local and global interests. The essence of PFE is that it aligns the local interests of participants with the global interests of the system by requiring that participants contribute content in order to receive content. Other systems have considered the problem of aligning local and global interests. For example, SETI@Home users voluntarily donate computing resources because their local interests are naturally aligned with the global interests of the system – namely the discovery of extraterrestrial life forms. At another level, Akella et al. show that TCP congestion control in older Reno variants of TCP exhibit stable global properties in the face of greedy individuals, but that more recent variants can result in an inefficient global network given greedy local behavior [3].

Enforce fair-exchange. When we began this work, we felt that we would simply need to adapt one of the many existing fair-exchange protocols that have been proposed in the cryptography and security literature. As we delved further into these protocols, we began to realize that, irrespective of their implementation complexity and runtime overheads, these protocols were unsuited for use in ap2p networks. At once, they provided a level of transfer integrity greater than necessary for ap2p networks, *and* a level of bandwidth protection that was insufficient. In Section 4.5, after having described our protocol, we provide a detailed analysis of fair-exchange protocols, specifically pointing out how they are unsuitable for use in ap2p networks.

4 The Protocol

In this section of the paper, we describe our *Pretty Fair Exchange* (PFE) protocol. First, we describe the complete protocol to give a high-level, functional sense of how it operates. Next, we deconstruct the protocol to provide greater insight into why we chose particular technological elements for inclusion in the protocol, and why conventional fair-exchange protocols are unsuitable in our context.

```
PFE(wanted file wf, owned files {of})  { ①

  while (!done) { ②
    (dst d, src s, file to send f) =          } ③
          joincircle(wf, of);

    for (i = 1 to num blocks in file) {
      send(d, f[i]);                  } ④
      wf[i] = receive(s);

      if (!verify_block(wf[i]) {            } ⑤
        next while;
      }
    }

    if (verify_file(wf)) {
      of += wf;                       } ⑥
      done = true;
    }
  }
}
```

Figure 1: The Pretty Fair Exchange (PFE) protocol. This pseudocode illustrates how PFE functions, from the perspective of a participant. The inlined numbers label elements of the protocol that we discuss in the body of the paper.

As its name suggests, our protocol is only "pretty" fair, in that it cannot guarantee that freeloaders will see no content advantage or that spoofers will not be able to cause damage. Because of our self-imposed constraint of not introducing centralized or globally trusted components to the network, we believe it is impossible to make such guarantees. A freeloader will always be able to gain some advantage, since in an exchange, somebody has to "transmit first", exposing themselves to bandwidth damage and giving others a potential content advantage. Similarly, a spoofer will always be able to cause some damage, since he can always send bogus content, causing the other party to waste effort downloading it.

However, given our constraints, our protocol substantially reduces the potential impact of these attacks. Freeloaders gain at most a single block of content and bandwidth advantage, and spoofers must spend resources proportional to the damage they wish to cause. We now turn to the details of the protocol that make this possible.

4.1 Pretty Fair Exchange

Using pseudocode, Figure 1 presents the PFE protocol from the perspective of one of its participants. First, the user indicates to his file-sharing application that he is interested in acquiring a particular piece of content. The application invokes PFE, giving it a description of the desired file, and a pointer to the set of files the user is willing to barter for it **(1)**. Next, PFE invokes `join_circle`, a protocol component that finds and establishes a group of participants that mutually satisfy each others' interests **(3)**. The outcome of this group establishment phase will select a file that the user must provide to another member of the group and that destination's name, and the name of the member of the group that will act as a source for the file the user wants.

Once the group has been established, PFE enters into the exchange phase. During this phase, each member of the group alternates sending a block to its destination and receiving a block from its source **(4)**; we assume that all files are split into fixed-sized blocks, and that all participants agree on the block size during the group establishment phase. Note that blocks are sent in sequential order, always starting with the first block of the file. The exchange phase continues until all members of the group possess the files they want, or until the group falls apart because a member has cheated or has become unavailable. For simplicity of exposition, we temporarily assume that all files are of the same size. Spoofing is detected on a block-by-block basis: after a member receives the next block of his file, he *incrementally verifies* that the block is what he expected **(5)**, and only proceeds with the exchange if he continues to receive valid blocks. There are many ways a participant could incrementally verify a file; we discuss details below.

After PFE has downloaded all of the blocks of the desired file, the entire file is verified for correctness, again using whatever verification techniques are appropriate and available. If the file successfully verifies, that file is added to the set of files that can be exchanged in the future, and the protocol terminates **(6)**. By verifying the file before trading it in the future, we prevent the viral propagation of spoofed content. If the file does not verify successfully (or if a block failed to incrementally verify during the exchange phase), PFE re-attempts group establishment, making sure to compose the new group differently than the previous, failed group **(2,3)**.

4.2 Drilling Down

PFE relies on a small set of technical building blocks, each of which strengthen our desired goals of fairness, proportional damage, and liveness. We now describe these building blocks, and the properties they add.

Verification (possibly incremental). Verification (bullets **(5,6)** in Figure 1) allows a receiver to determine whether content is genuine. Verification prevents viral propagation, and therefore makes protective spoofers spend resources proportional to the number of transfers they seek to disrupt. Incremental verification is simply an optimization over verification which limits the bandwidth damage a participant incurs during a given exchange attempt.

The most appropriate mechanism to perform verification likely depends on the nature of the file itself. For example, if the file is a media stream, the user could listen to the stream in real-time as it is being downloaded, canceling the exchange if the file isn't what he expected. Alternatively, the user could rely on a public, trusted database of incremental file hashes (e.g., Merkle trees), although this would introduce reliance on a trusted, centralized service into the ap2p network.[1] PFE doesn't take a specific stance on what verification mechanism should be used, but instead provides a hook into which verification mechanisms can be plugged.

Bandwidth barter. To prevent freeloading, we use a mechanism which ensures that somebody downloading content provides commensurate upload bandwidth. Bullet **(4)** of the protocol shows how we do this. Each party in the exchange makes sure that they bound their bandwidth damage to one block, by only sending their next block once they receive the previous block they are owed. A freeloader that wishes to receive all of the blocks of a file during a single exchange is forced to send nearly all blocks of the file they owe. Bandwidth bartering also bounds the content advantage that freeloaders can gain during a single exchange to a single block. Reducing the block size therefore reduces the content advantage that a freeloader can obtain during an exchange, but also reduces the bandwidth damage to which a participant is exposed.

Controlling block transmission order. Given that we split content into blocks, we need to decide on the order in which blocks are sent during an exchange. If a freeloader can request a specific order, that freeloader can exploit the one-block content advantage that bandwidth bartering permits to obtain the entire file, by downloading successive blocks in successive exchanges. By picking a globally enforced, fixed transmission order for blocks (in our case, a sequential order starting at block 1 and ending at the last block of the file), freeloaders have no sustainable content advantage, since the only block they can get without uploading a block is the first block of the file. They need to upload blocks to get latter blocks in the file.

A deterministic transmission order permits malicious spoofers to cause bandwidth damage across exchanges, however. The spoofer can upload all but one of the blocks it owes, forcing the recipient to redownload nearly the entire file during the next exchange to obtain that last block. Controlling block transmission order makes sense if the number of malicious spoofers is small, since with high probability, a victim will be able to join a non-malicious group on its next attempt. If the number of malicious spoofers in the system is high, there is nothing that any system can do to prevent them from causing substantial damage, as with all open systems. We return to this issue

[1] Such hash services are beginning to emerge in practice, for example `http://www.bitzi.com`.

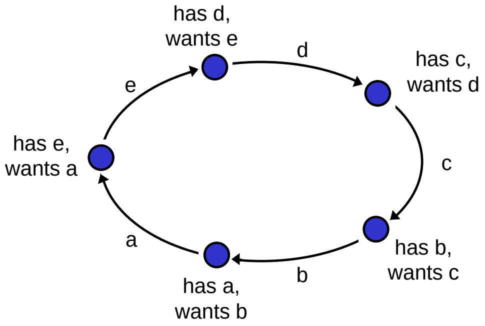


Figure 2: Exchange groups, or "circles". We generalize pairwise barters to exchange groups formed out of circles: each user in the circle provides content in one direction, and receives content from the other direction. Circles allow greater flexibility than pairs to satisfy exchange constraints.

in Section 4.4.

Exchange groups. PFE relies on barter: to obtain content, a user must provide content that somebody else wants. Pairwise exchange is a simple way of bartering, in which two peers directly satisfy each other's needs. However, as we will show in Section 5, pairwise exchange does not always provide adequate liveness. Fortunately, we can generalize pairwise exchange to group exchange, by introducing the notion of an "exchange circle" (Figure 2). In a circle, each participant provides content to the next person in the circle, and receives content from the previous person in the circle. Verification, bandwidth bartering, and deterministic block transmission ordering all generalize from pairs to circles. However, if a spoofer joins a circle, the damage caused by that spoofer is amplified by the number of participants in it, pressuring the system to prefer small circles during group establishment.

4.3 Forming Circles

PFE relies on the ability for peers to organize themselves into circles that mutually satisfy each others' interests during an exchange, as shown in Figure 2, but it does not specify a particular architecture or algorithm for doing this. We believe this is a separable part of the overall protocol, in that exchange group formation could be realized through any number of mechanisms. Some possibilities include:

Centralized matchmaking: The simplest architecture for forming exchange groups is for all peers to upload a list of files they possess and a list of files they are interested in to a centralized "matchmaker" service. Given such global information, finding circles is a matter of simple graph algorithms. Each peer in the system is a node in the graph, each file in the system is another node in the graph. The "owns-file" relationship is represented by a directed arc from a peer to a file, and the "wants-file" relationship is represented by a directed arc from a file to a peer. Forming exchange groups is a matter of finding circuits in the resulting bipartite graph. Centralized matchmaking has the advantage of complete information, but it has the obvious disadvantage of being a scalability bottleneck and a single point of failure in the system.

Partitioned matchmaking: Instead of having a single centralized matchmaker, an alternative is to have many dedicated matchmakers, and to partition the population of peers amongst these matchmakers. As we will show in Section 5, even with small population sizes, it is possible to form groups and to make the system live. This suggests that a partitioning strategy would work well, since each partition is effectively a separate, small population of users. Partitioned matchmaking trades optimality (global information) for robustness (no single points of failure).

Decentralized matchmaking: Instead of having dedicated, partitioned matchmakers, fully distributed equivalents could exist. One possibility is to have peers volunteer to be matchmakers, in a manner similar to how some peers in existing P2P file-sharing systems promote themselves to be "supernodes", indexing content to satisfy queries. Another possibility would have peers organize into an overlay, and to broadcast their "owns-file" and "wants-file" sets across the overlay; peers would listen to broadcasts as well as sending them, searching for possible circles and proposing them to each other as they form. A final possibility would be to use distributed hash tables (DHTs) [30, 27] to store the "owns-file" and "wants-file" sets of each user in a distributed, inverted index: given the name of a file, the DHT would return the set of users that want the file. Given the name of a user, the DHT would return the set of files that user owns.

We do not advocate one mechanism over another. In Section 5, we present trace-driven simulations that show that there is adequate opportunity to form circles using any of these mechanisms.

4.4 The Effectiveness of PFE

Returning to our two classes of cheaters (freeloaders and spoofers), we now consider the degree to which PFE defeats them, and the potential for an honest user to be harmed in a system that uses PFE.

4.4.1 Attacks by Freeloaders

The combination of bandwidth barter and deterministic block transfer order limits the potential gain of a freeloader to a single block. Freeloaders can easily obtain the first block of any file with no upload cost, but to acquire subsequent blocks, the freeloader must spend upload bandwidth proportional to downloaded content.

Freeloaders, as they exist in today's ap2p networks, can no longer exist.

A freeloader might choose not to provide the last block of its file during a transfer, in effect becoming a spoofer, and forcing the recipient of that file to re-fetch the entire file from another host. However, a freeloader has little incentive to do this, since they would save very little bandwidth by doing so, given that they have already uploaded virtually all of the file. A freeloader is greedy, not malicious, and bandwidth bartering has virtually eliminated the profitability of their greed.

4.4.2 Attacks by Spoofers

Verification prevents spoofers from being able to amplify the damage they cause by tricking unwitting peers from further propagating spoofed content. Because of this, a spoofer who wishes to inflict damage on a participant must spend resources proportional to the damage he wishes to cause.

The most damage a spoofer can cause to a participant during an exchange happens when the spoofer causes the participant to receive all but one block of the file, forcing him to attempt to re-acquire the entire file during another exchange. If subsequent exchanges are also tainted by having a spoofer as a member, the damage to the participant accumulates. From the perspective of an honest participant, the amount of damage they are likely to experience is related to the probability that a spoofer exists within a group. If the probability that a spoofer exists within a group is p, then on average, a participant will successfully receive the file on exchange attempt number $\frac{1}{(1-p)}$, assuming the participant is unable to identify and blacklist the spoofer during a failed exchange.

The probability that a spoofer joins a particular group depends on a number of factors: the size of an exchange group, the number of files the spoofer offers (regardless of whether they actually possess the file), and the number of files the spoofer pretends to be interested in. The larger the group size, the more likely that spoofer is to join the group. Additionally, if a spoofer sabotages a large group, the spoofer effectively amplifies his damage by the number of participants in the group. For this reason, the system should prefer smaller groups.

4.5 Why Cryptographic Fair-Exchange Protocols Are Unsuitable

Fair-exchange protocols have been thoroughly studied in the literature over the past decade, with applications in contract signing [20], certified delivery of content [32], and electronic payment for electronic goods [12]. Fair-exchange protocols guarantee that during an exchange, no involved party can gain an advantage over other parties, even if the protocol halts for any reason at any time. To understand why existing fair-exchange protocols are poorly matched for the exchange of content in an ap2p network, we review the properties of these networks, and describe how fair-exchange is at odds with them.

Third parties are unwilling to participate in the exchange. Most existing fair exchange protocols involve the use of a trusted third party which acts as an escrow agent. An escrow agent may be required to download and store copies of the exchanged content, both to verify that the content is valid, and to reveal the content in the event that one of the parties refuses to do so.[2] Accordingly, escrow agents would have substantial bandwidth and storage requirements, creating a substantial barrier to deployment. Moreover, in an ap2p network, third parties would invariably assume some type of responsibility for the legitimacy of the content as it may relate to issues of copyright and ownership.

Anonymity must exist globally, not just transactionally. Variants of fair-exchange protocols seek to preserve the anonymity of a transaction: participants can exchange content without revealing their identity to each other, or without revealing the nature of transacted objects to non-participants. However, many of these systems (particularly those involving electronic commerce) assume that participants have long-lived identities, either so that misbehavers can be exposed and punished, or so that purchase orders can be drawn from participants' accounts. In an ap2p system, participants may not have any meaningful or persistent identity.

Exchanges may involve groups, not just pairs. Although liveness may require that the system facilitates exchange groups with more than two people, many fair-exchange protocols only provide pair-wise fair exchange.

Practical solutions are required. Ap2p systems are real, and they should only rely on practical technology. Many of the proposed fair-exchange protocols rely on exotic technologies (such zero-knowledge proofs [8] or homomorphic pre-images of signatures [5]), which may be impractical in real-world situations, and which do not have time-tested implementations. For an ap2p system to be successful, it must be deployable, and as a result, it must limit itself to practical technologies.

Fundamentally, existing fair exchange protocols are *only* concerned with eliminating any advantage that might be gained by a party. They have no notion of damage, and may even make it relatively easy to cause

[2] Optimistic verifiable fair-exchange protocols exist that only involve the escrow agent to resolve disputes, but these protocols are limited to the case in which the objects being exchanged are digital signatures on publicly known objects [5, 7], and as such, are not appropriate for the exchange of arbitrary files.

damage (e.g., through inaction) to another. Consequently, these protocols only serve to further the interests of the spoofer.

4.6 Summary

This section has presented Pretty Fair Exchange (PFE), a simple protocol built out of easily-understood components that prevents freeloaders and mitigates the damage caused by spoofers to the extent possible. In the next section of this paper, we consider the behavior of the protocol in light of its liveness constraint: does introducing the requirement that participants find an exchange group with matched interests lead to reasonable transaction progress in real networks?

5 Using Trace Driven Simulation to Demonstrate Liveness

In this section, we use trace-driven simulation to evaluate the *liveness* property of *PFE*. In considering liveness, we seek a system that emulates the properties of a *lively* exchange in the real world. Specifically, in a lively exchange, goods move smoothly in a timely fashion and with minimal complexity. Moreover, wealth is plentiful, thereby discouraging theft. Lastly, the marketplace functions well even with a modest number of participants, enabling it to scale down as participants exit, and scale up by means of partitioning the system as users enter.

As these qualities apply to anonymous peer-to-peer exchange networks, we answer the following questions in the context of *PFE*:

1. **Are users able to acquire the content they want with reasonable delay?** This question corresponds to *can a user join a group that will soon "close" in a transitive exchange?* Users may become extremely dissatisfied when infinite, or even very long, waiting times are the norm.

2. **Will poverty motivate users to cheat?** *PFE* rewards those with popular content and isolates those without. If this isolation is extreme, users will be encouraged to "act poor," advertising content that they do not have in order to attract others to trade with them. Although this will be detected during verification, it causes bandwidth damage to the entire group. On the other hand, if users are able to acquire the content they want with relatively high confidence, then they will have less motivation to cheat.

3. **Can complete groups be formed with relatively few members?** Small groups can be formed quickly, and can complete exchanges with greater simplicity. Moreover, in a small group, relatively fewer members are impacted by a single cheater (recall that the group-wise transaction is aborted if a single member cheats).

4. **Can the marketplace remain live even with a relatively modest number of participants?** Here, we are concerned with how many users must participate in the network in order for it to remain live. A network that makes progress with fewer users is more appealing than one that requires a massive membership, for two reasons. First, it requires a smaller critical mass and therefore has a larger operating range. Second, it enables a network of brokers who can work relatively independently of one another.

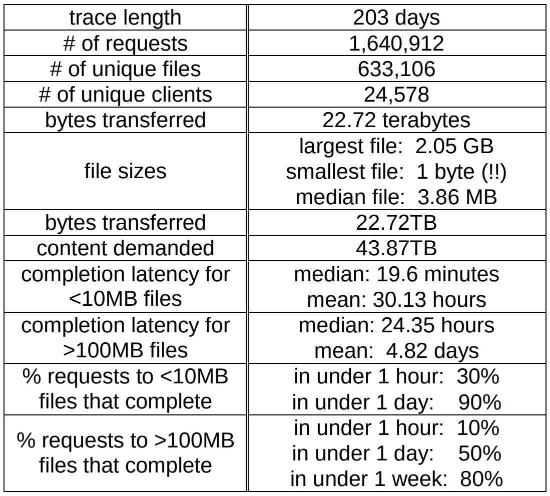
As we show in the remainder of this section, the answer to each of these questions is yes. Specifically, in a trace of over 1.6 million file requests, over 94% of them are eventually satisfied, with nearly 28% immediately, and over 50% in under one day. Of the nearly 22,000 users traced, over 86% are able to download *all* of the files they seek, and over 98% are able to download at least 90% of their desired files. Over 12% of all transfers occur in groups of size two, and all transfers can occur in groups of size five or less. Lastly, these same trends hold even when the population size is halved. At about 2000 users, the system begins to break down, and at 500 users, nearly 60% of requests go unfulfilled.

5.1 Methodology

Our trace driven simulation is based on a measured workload [4] of the *Kazaa* file sharing network [22] . We monitored and recorded all Kazaa traffic flowing in and out of a large University over a 203 day period between May and December of 2002. The essential statistics about the trace are shown in Table 1.

In analyzing the traces, we make several additional assumptions which are not reflected directly in the trace, but which will hold in a system using *PFE*. First, we assume that any file downloaded by a user is permanently made available by that user for upload. Although this is not true in the system measured, two factors make it reasonable: (i) modern disks are sufficiently large to hold all downloaded content, and (ii) users recognize offered content as currency worth saving.

Our next two assumptions go to the issue of seeding content. In a real system, files are seeded into the system by some out-of-band method, such as obtaining them from an FTP server. Since this seeding is outside the scope of *PFE*, we follow a simpler seeding protocol. Firstly, when a new user comes on-line, we allow them to obtain the first ten files they request "for free," recognizing that a user must first have in order



trace length	203 days
# of requests	1,640,912
# of unique files	633,106
# of unique clients	24,578
bytes transferred	22.72 terabytes
file sizes	largest file: 2.05 GB smallest file: 1 byte (!!) median file: 3.86 MB
bytes transferred	22.72TB
content demanded	43.87TB
completion latency for <10MB files	median: 19.6 minutes mean: 30.13 hours
completion latency for >100MB files	median: 24.35 hours mean: 4.82 days
% requests to <10MB files that complete	in under 1 hour: 30% in under 1 day: 90%
% requests to >100MB files that complete	in under 1 hour: 10% in under 1 day: 50% in under 1 week: 80%

Table 1: Trace statistics. These statistics reflect the behavior of the Kazaa system as experienced by the traced users.

to get. Second, the first time a file is requested by any user, we assume that the file is already in existence and make it available to the user; in fact, we actually allow the first five transfers of any file to occur for free. Our choices of ten and five are arbitrary, and we have confirmed that our results are not sensitive to the degree of seeding, as long as some seeding occurs.

Our final assumption concerns the "cost" and "value" of content sought or offered by participants in terms of the bandwidth consumed. There is significant diversity in the files traded in file sharing networks [28] with file sizes spanning six orders of magnitude. Images and text files typically are a few kilobytes in size, most audio clips are approximately 3MB, and video files may be as large as a gigabyte. Given this, a user is likely to be unwilling to upload a gigabyte video file in order to receive a kilobyte text file. Consequently, our simulator splits large files into multiple one megabyte chunks. When a user desires a large file, he must engage in a separate exchange for each chunk. However, for purposes of the simulation, we do not consider a file request as "completed" until each and every chunk has been requested and returned to the user. Thus, a single request for, say, a 100MB file, will involve 100 separate exchanges (made in parallel), but will be counted as a single completion.

With these assumptions, we play back our trace into our simulation one record at a time, in time order. For each transfer that occurs in the trace, we add the referenced file to the set of files the user *desires*, and then we globally search the system to find an appropriate group whose *offerings* satisfy each others' desires. If we find such a group, we simulate an exchange, converting the exchanged files from desires to offerings for the appropriate users. We continue to search for groups until none are left, at which time we feed the next trace record into the simulator. If there is a choice of groups, we prioritize small groups over large groups.

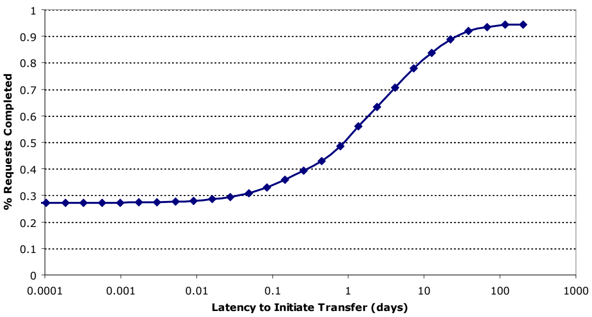


Figure 3: Completion rate and latency. Users are able to immediately get files they need 28% of the time, but some files take days or weeks to acquire. 94% of files are ultimately acquired. The average delay is 15.8 days and the median is about 95 minutes.

Our simulation is concerned only with the properties of the groups themselves, not with their ensuing transfer properties. Consequently, we do not model the bandwidth, latency, or reliability attributes of an exchange. Instead, exchanges occur instantaneously and reliably as soon as they become possible. Moreover, we do not directly simulate the impact of cheaters: we assume that users offering content are genuine. In practice, freeloaders would be detected early in a transfer, since they would not offer genuine content, and spoofers would force some fraction of transfers to fail, causing bandwidth damage and therefore overhead to the system, but not ultimately preventing progress.

5.2 Do users get the content they want and do they get it quickly?

In our simulation, we add a file to a user's desired set at the time the user requested that file in our trace. The user may be lucky, immediately finding a group having a member needing a file that the user has, or he may have to wait until such a group becomes available. If the user is very unlucky, a group will never form, and the user will never receive the file requested.

In Figure 3, we plot a CDF of the fraction of desired files that are successfully acquired as a function of the time it takes to form the groups that resulted in their acquisition. The graph shows that 94% of files are successfully acquired by the end of the trace, indicating that users' wants are indeed eventually satisfied. Moreover, we see that 28% of the time, a user who wants a file is able to acquire it immediately. However, if the user doesn't find his file right away, he may often have to wait a day to acquire it. In some cases, the wait may be as long as weeks.

Figure 4 illustrates the impact that file size has on

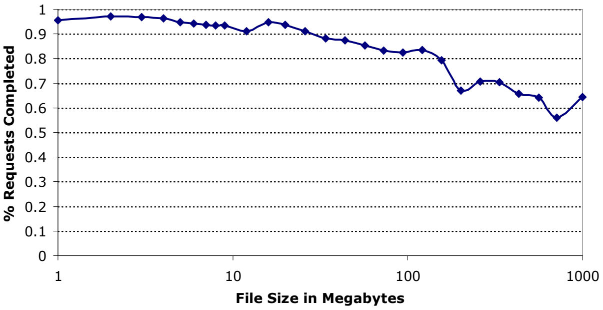


Figure 4: Acquisition rate vs. file size. Users' desires for small files are acquired over 95% of the time. Large files are less likely to be acquired, but fortunately, the vast majority of files requested in the trace are small.

completion. Smaller files, which for example represent audio, have a completion rate of over 95%. In contrast, the larger video files enjoy a completion rate of between 60% and 70%. In practice, a higher completion rate is indicative of a file's popularity. A request for a popular file is more likely to close a group than one for an unpopular file because it is more likely that there exists another user offering that more popular file. Similarly, an offering of a popular file is more likely to close a group than that of an unpopular file. This greater likelihood of closing a group manifests itself in terms of waiting time, since a user would be expected to wait less when requesting or offering a popular file.

This behavior becomes evident by viewing the waiting time for small files separately from that for larger files. From Figure 5, which shows separate completion rates and latencies for small files and large files, we see that the system is relatively responsive for the smaller ones. For small files (<10MB), the average completion time is 5 days and the median is 18 hours. For large files (>100MB), the average is 20.5 days and the median 21 days.

By way of comparison, the actual system we traced completed only 30% of its requests to small files in under and hour, and 10% took longer than a day. Overall, the measured system had an average small file completion latency of 30.13 hours and a median of 19.6 minutes. In contrast, only 10% of requests to larger files completed in less than an hour, 50% in less than a day, and 20% more than a week. The average completion latency for large files was 4.82 days and the median was 24.35 hours.

From this data, although we conclude that *PFE* delivers content slower than existing unfair protocols, the time delay for delivering files is within the range that users of today's systems tolerate. Furthermore, these delays will lessen as the system grows in population size, and in return for these delays, users are shielded from cheaters.

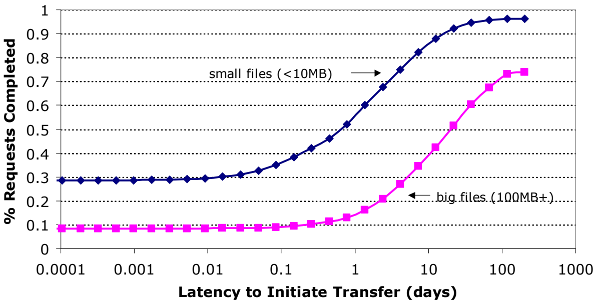


Figure 5: Completion rates and latencies broken out for small files and large files. Requests for the more popular small files complete much more quickly (average of 5 days, median of 18 hours) than for larger ones (average of 20.5 days, median of 21 days).

5.3 Will poverty motivate users to cheat?

As mentioned, if users with less content are locked out of groups because they can't satisfy others as well as richer users, they might be motivated to fabricate offers. Although verification will detect such activity, it is nevertheless undesirable as it incurs damage to the users who are spoofed. In terms of the traces, relative poverty would manifest itself as a non-uniform distribution of completions across the user population.

In Figure 6, we plot the distribution of completion rates across users. This graph shows that most users (86% of them) are able to successfully acquire all files that they are interested in. No user is completely stranded: the worst case user acquires 35% of the files he wants. Although there does exist a small subset of users who get fewer files than they want, the majority of users are completely satisfied, indicating that the system does not punish the poor, coercing them to cheat by lying about their content.

By way of contrast, approximately 66.2% of transactions in the traced Kazaa system failed. This poor success rate is partially due to the fact that peers in the system are overloaded because of freeloading, and partially because users do not make previously downloaded content available to others. Despite its rather poor success rate, the system we measured has managed to attract millions of users. From this, we conclude that users in a system with *PFE* could achieve substantially better service than they do today.

5.4 Can groups be small?

For practical reasons, it is desirable to bound the size of exchange groups. Large groups require more coordination, both to form them and to complete the exchange. Moreover, since a single cheater can cause the exchange to abort, the more participants in a group, the greater the impact of a single cheater. Consequently, it is important to understand whether groups

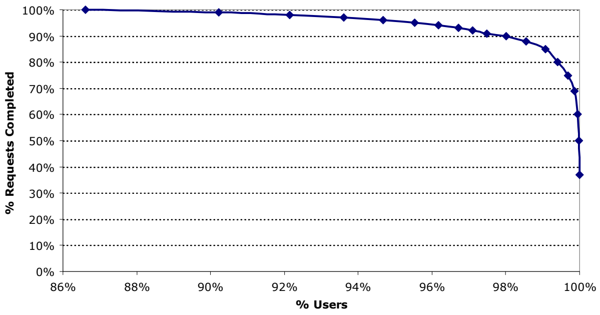


Figure 6: Completion rate distribution across users. The percentage of users able to achieve a given completion rate. More than 86% of users are able to download 100% of their desired files. 98% of users are able to download at least 90% of their desired files.

tend towards the larger or the smaller. We would like to understand the system's liveness properties when the group size is limited in order to determine if there is a cap value small enough to permit reasonable closures, yet large enough to sustain liveness.

In order to establish the effect of group size, we played back the traces several times with different maximum group sizes and observed the system's behavior. To implement a maximum group size in the simulator, we simply restricted the search algorithm so that it would not attempt to form a group larger than the maximum. For example, with a maximum group size of three, the simulator would seek cycles in a graph of "wants and offers" of length no greater than three for each new want or offer introduced.

Figure 7 shows the fraction of users' desired files that are successfully acquired as a function of the maximum permitted group size. Limiting the system to pairwise exchanges noticeably degrades system behavior. In contrast, there is no benefit in allowing groups of more than five members. From this, we conclude that a practical group construction algorithm can be limited to constructing relatively small groups (≤ 5 members), but that there is substantial benefit to supporting groups having more than two members.

Turning to the distribution of group sizes in a system having a maximum group size of five, we see from Figure 8 that the likelihood of participating in a given group size is nearly uniform. In contrast, since the closing of a larger group facilitates more transfers, their impact on the overall completion rate is largest. Even though we would prefer to restrict a system to only relying on pair-wise exchanges, we found that group exchanges with more than two participants are necessary for the liveness of the system.

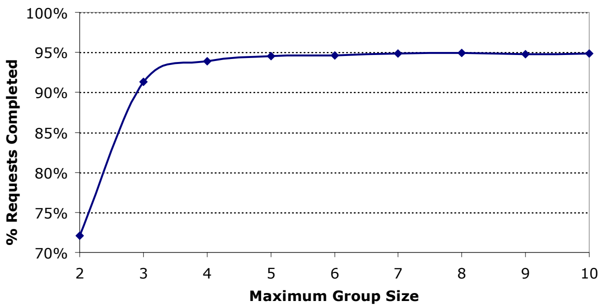


Figure 7: Completion rate vs. maximum group size. Even if we bound the maximum size of groups to five, users still acquire 95% of the files they want.

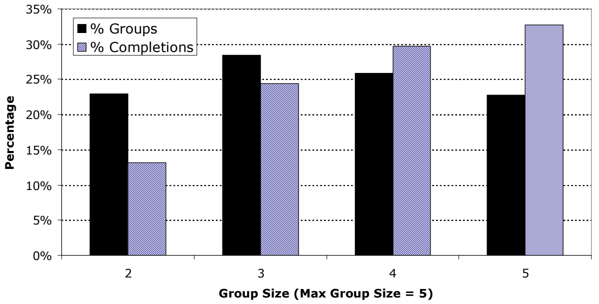


Figure 8: Distribution of groups and transfers for a system in which groups can be no larger than five. With a bounded group size, the distribution of actual sizes (% Groups) is relatively uniform. In contrast, the larger groups facilitate more transfers (% Completions), with the largest number of transfers occurring in groups of size five.

5.5 Is a small marketplace sufficient?

How many users must participate in a system before that system becomes viable? If this number is too large, the system will fail, as it will never attract the "critical mass" of users necessary to match interests. Conversely, if the number can be small, then it becomes feasible to partition users across brokers, enabling multiple brokers to serve the system in practice. To explore the issue of population size, we sub-sampled our trace to extract out smaller population sizes.

In Figures 9 and 10, we show the completion rate and time of the simulation as a function of the population size of participating users. The first graph allows us to compare the percentage of requests completed for a given latency, and the second to see the fraction of requests that complete after about a day, and by the end of the simulation. From both figures, we see that system behavior changes relatively little until the population drops to below about 8000. At about 2000, the completion rate and latency degrades substantially, suggesting that the system reaches its critical mass with around 5000 users. In "Internet" terms, this is

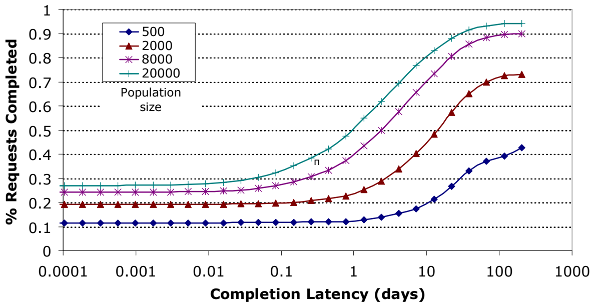


Figure 9: Latency vs. completion rate. Time required to satisfy a given fraction of the population's desires, for various population sizes.

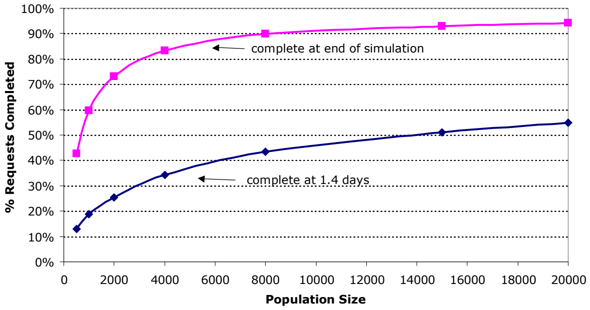


Figure 10: Population size vs. completion rate. Request completion rate as a function of population size, given a maximum transfer latency of 1.4 days, or an unbounded transfer latency.

a relatively small number, and we therefore conclude that *PFE* does not require a substantial user base to be effective, and that it can be supported with a network of brokers, each having relatively modest capacity.

5.6 Summary

From our trace-driven simulations, we conclude that it is feasible to use *PFE* in an ap2p file sharing network having a workload similar to today's systems. Our simulations confirm that these systems would have a high degree of liveness: most users would be able to acquire most (if not all) of their desired files. We have also shown that the system would satisfy requests fairly quickly: nearly a third of requests would be satisfied right away, and over 50% of requests would complete within a day. Finally, our data suggests that as the population grows, the quality of service that each user receives improves gradually, although critical mass is reached with a relatively modest number of users.

6 Conclusions

Today's anonymous peer-to-peer (ap2p) file sharing networks suffer damage caused to them by *cheaters.* We identify two kinds of cheaters: freeloaders, who consume resources without providing them, and spoofers, who attempt to cause users to waste bandwidth by downloading useless content. In this paper, we presented Pretty Fair Exchange (PFE), a protocol that mitigates the effects of such cheaters in ap2p networks.

The essence of PFE is that it changes the basic operation offered by an ap2p network from *download* to *download-while-uploading*. By forcing peers to provide content in order to obtain it, PFE prevents freeloaders from gaining any substantial advantage over other users. We accomplish this through *bandwidth bartering*: peers exchange content block-by-block, only uploading the next block once they have received their currently owed block. PFE also gives users the ability to verify content as they download it. Because of this, spoofers cannot trick other users into virally propagating spoofed content.

Because we explicitly chose not to introduce trusted or centralized components, PFE is only "pretty" fair. PFE eliminates most, but not all, advantage gained from freeloading: without a trusted third party to escrow content, a participant can leave an exchange without having transferred his last block of content. Similarly, without persistent, authenticatable identities, spoofers cannot be permanently blocked from the system. Nonetheless, PFE limits the advantage that freeloaders can gain to a small fraction of a file, and PFE forces spoofers to continually spend resources proportional to the damage they want to cause.

Using trace-driven simulation, we demonstrated that an ap2p network using PFE will be *live.* In practice, it is possible to organize users into exchange groups in which users mutually satisfy each other's wants. Despite needing to find compatible groups, we show that users acquire the content they want with reasonable delay, that groups can be formed even if the total population is small, and that in practice, small group sizes provide adequate system liveness.

References

[1] E. Adar and B. Huberman. Free riding on gnutella. In *First Monday, 5(10)*, October 2000.

[2] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, December 2002.

[3] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou. Selfish behavior and stability of the Internet: A game-theoretic analysis of TCP. In *Proceedings of the ACM SIGCOMM 2002 Conference*, Pittsburgh, PA, August 2002.

[4] Anonymous. Citation removed for purposes of anonymous review.

[5] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications.*

[6] R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, NY, 1984.

[7] F. Bao, R. Deng, , and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, Oakland,CA, May 1998.

[8] G. Brassard, D. Chaum, and C. Crepeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences (JCSS).*

[9] CCITT. Recommendation X.509: the directory – authentication framework, 1988.

[10] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings of Advances in Cryptology - CRYPTO 1988*, Santa Barbara, CA.

[11] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[12] B. Cox, J. Tygar, and M. Sirbu. Netbill security and transaction protocol. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, July 1995.

[13] L. B. Cox, C. D. Murray, and B. D. Noble. Pastiche: making backup cheap and easy. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, December 2002.

[14] L. P. Cox and B. D. Noble. Fairness in peer-to-peer storage systems. In *Submitted to the Ninth Workshop on Hot Topics in Operating Systems (HotOS IX)*, Lihue, Hawaii, May 1993.

[15] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October 2001.

[16] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar. A reputation system to increase MIX-net reliability. *Lecture Notes in Computer Science.*

[17] J. R. Douceur. The Sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, March 2002.

[18] P. Druschel and A.Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the Eighth IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.

[19] eBay Inc. `http://www.ebay.com`.

[20] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM (CACM).*

[21] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro. The Millicent protocol for inexpensive electronic commerce, December 1995.

[22] Kazaa Media Desktop. Usage statistics given at `http://www.kazaa.com`.

[23] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, Cambridge, MA, November 2000.

[24] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard. A cooperative Internet backup scheme. In *Proceedings of the 2003 USENIX Annual Technical Conference*, San Antonio, Texas, June 2003.

[25] MojoNation. `http://www.mojonation.net/MojoNation.html`.

[26] A. Orlowski. "I poisoned P2P networks for the RIAA". News article from The Register, `http://www.theregister.co.uk`.

[27] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[28] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of Internet content delivery systems. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, December 2002.

[29] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings USENIX Winter Conference 1988*, Dallas, Texas, USA.

[30] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM 2001 Technical Conf.*, August 2001.

[31] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium.*, Aug. 2000.

[32] J. Zhou and D. Gollman. A fair non-repudiation protocol. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1996.

[33] P. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, MA, 1995.