Addressing the Non-Cooperation Problem in Competitive P2P Systems

Sepandar Kamvar Beverly Yang Hector Garcia-Molina

Abstract

Large-scale competitive P2P systems are threatened by the *non-cooperation problem*, where peers do not forward queries to potential competitors. While non-cooperation is not a problem in current P2P free file-sharing systems, it is likely to be a problem in such P2P systems as pay-per-transaction file-sharing systems, P2P auctions, and P2P service discovery systems, where peers are in competition with each other to provide services. Here, we motivate why non-cooperation is likely to be a problem in these types of networks and present an economic protocol to address this problem. This protocol, called the RTR protocol, is based on the buying and selling of the right-to-respond (RTR) to each query in the system.

1 Introduction

While peer-to-peer networks have risen to prominence due to the success of free file-sharing networks like Napster and Kazaa, increasing emphasis is being placed on new applications of P2P, including pay-per-transaction networks, P2P auctions, and P2P service discovery systems. The problems in these networks are likely to be different than the problems encountered in free file-sharing networks. In this work, we argue that, while the *freeriding problem* is common in today's peer-to-peer networks, the *non-cooperation problem* is likely to be a bigger problem in future peer-to-peer networks.

In the context of free file-sharing, the *freeriding* problem has become a central issue: peers acting in their own best interest conserve their resources (i.e. bandwidth) by sharing no files, and hence, only a small fraction of altruistic users offer almost all the available content. For example, in the Gnutella file-sharing system [2], over 70% of the content was provided by just 5% of the users [1]. As a result, much existing work on incentives in P2P systems have been focused on solving the problem of freeriding (e.g., [3, 4]).

However, in applications where peers *gain* from answering queries, freeriding is unlikely to be a problem, since the necessary incentives are inherent. For example, in a pay-per-transaction file-sharing system where peers get paid for uploading files, peers will want to share files, because this generates income. In an auction system, where the auction advertisement is analogous to a query and bids analogous to query responses, peers will want to submit bids. Even in a free file-sharing system where users share their original music or artwork, users have an incentive to share the files in order to increase their publicity.

In each of the systems described above, not only are peers eager to provide services (e.g. share files), but they are in *competition* with other peers to provide their services. Competition is a problem in P2P frameworks that rely on peers to forward queries (e.g., Gnutella [2], DHTs like [7], etc.),[1] because a peer acting in its own best interests will not forward queries to potential competitors. For example, a peer providing a car rental service might not forward a query for car rental services. Instead, it could answer the query and then drop it, so as to improve its chances of gaining business. Therefore, P2P systems will no longer operate correctly due to non-cooperation, even though abundant services are available.

In this paper, we propose an economic protocol to ensure that peers cooperate in the *operation* of P2P systems in the face of competition. We present this protocol in Section 2, where we discuss how our proposed system will not only assure proper operation, but can also improve the efficiency and effectiveness of the search mechanism as well. In Section 3, we present a high-level attack analysis of the protocol, including a discussion of potential attacks and pitfalls, and how these issues are countered. Finally, in Section 5 we present a list of important areas requiring further investigation.

In this work, we illustrate our protocol on top of the Gnutella protocol for P2P search running a pay-

[1] The exceptions are systems that require no forwarding, such as Napster (http://www.napster.com).

per-transaction file-sharing application. Important items for future work include extending these ideas to architectures other than Gnutella, such as DHTs (see Section 5). The following discussion assumes the existence of an efficient micropayment scheme for P2P systems, such as that described in [8].

1.1 Preliminaries

The basic Gnutella search protocol works as follows: each user runs a client (or *peer*), which is connected to a small number of other peers (known as *neighbors*) in an overlay network. When a user submits a query, her peer will send the query message to all its neighbors, who will in turn forward the query to their neighbors, and so on. A peer that receives a query and finds that it can answer will send a response to the querying peer.[2] The querying peer will wait a period of time for responses to arrive, and then it will select one or more responding peers from which to buy services. In a pay-per-transaction application, the service offered is the download of a file, and the querying peer will pay the selected peer(s) for each download transaction. The price per download may vary depending on the file. Further details of the Gnutella protocol can be found in [2]. Clearly, if peers do not forward queries, the search mechanism will fail.

2 RTR Protocol

At the core of our protocol is the concept of a *right to respond*, or RTR. An RTR is simply a token signifying that a peer has a right to respond to a query message. We choose this name ("right to respond"') in order to emphasize that a query is really a commodity. Peers should pay to receive the query, because that in turn brings in potential business. If a peer never receives any queries, then it can never provide its service to anyone. An analogous concept in real-life markets are companies that buy lists of emails or referrals from other companies, so that they have a new pool of potential customers.

Once a peer buys an RTR for a given query, it may do one or both of the following: (a) respond to the query and hope that it is chosen to upload its services, and (b) sell the RTR to other peers.[3] Peers can buy and sell RTRs with their neighbors only.

In this framework, selling an RTR is equivalent to forwarding a query. Hence, there is built-in incentive to forward queries, since peers get paid to do so. Of course, some peers may still choose to not forward any queries in order to increase the probability that they will be chosen to provide the service. However, their actions will be offset by those peers who hedge their risk by selling a few RTRs, and by those peers who speculate in RTRs (buying RTRs simply to resell them).

2.1 Basic Implementation

An RTR has the following format:

$$RTR = \{Q, ts, query\}_{SK_Q} \quad (1)$$

where Q is the identity of the querying peer, ts is the timestamp at which the query was first issued, and $query$ is the actual query string. These three values are signed by the querying peer's secret key SK_Q, so that RTRs cannot be forged. Hence, each query requires a single signature generation, and one verification per forward.

When a peer A forwards a query to a neighbor B, it will first send the offer containing partial RTR information and a price:

$$\textit{Offer} = \{rep(Q), ts, query, price\} \quad (2)$$

where $rep(Q)$ is the reputation of the querying peer (described below). The subject of pricing is discussed in the next section. The offer contains enough information for B to determine whether to purchase the RTR, and whether the RTR is a duplicate B has seen before. However, because the identity of Q is not revealed, B can not actually answer the query without purchasing the full RTR. If B decides not to purchase the RTR, he will simply drop the offer. Otherwise, B will send a purchase request to A, and peer A will forward the full RTR to B.

To prevent being spammed by useless offers, each time a peer connects to a new neighbor, it can specify the following "flow control" parameters:

- The desired *rate* (e.g., messages/day) of RTRs that will be delivered.

[2] In Gnutella, response messages are actually forwarded along the reverse path traveled by the query. In systems that do not require anonymity, sending responses directly to the querying peer is more efficient.

[3] If a peer sells an RTR, it may still respond to the query corresponding to the RTR. That is, a peer does not lose the right to respond to a query when it sells the RTR for that query.

- *Filters* that specify the "content" and "quality" of the desired RTRs.

Filters can be set on any of the three fields of an RTR: the query string, the reputation of the querying peer $rep(Q)$, or the timestamp. Filters on the query string specify the *content* of the RTR, and affect the probability that the purchaser can respond to the query. Content filters may have varying levels of restrictiveness. For example, a content filter may specify that the RTRs should only contain queries for a particular genre of music files. Or, a content filter may specify exactly what RTRs (i.e., for which exact files) will be purchased.[4] At the other extreme, the filter can be null, meaning all queries qualify. Such a setting is appropriate for peers who wish to make money by speculating on RTRs. Filters on the reputation of querying peer Q and the age of the query (indicated by the timestamp) specify the *quality* of the RTR, and affect the probability that a responding peer will be chosen and paid for its services.

If a peer receives too many RTRs that violate content or quality filters, then that peer can disconnect from its neighbor. It is therefore in each seller's interest to provide high-quality, relevant RTRs to its best ability – otherwise, it may lose potential customers (see Section 2.3).

At the same time, if a peer never buys any RTRs, even if the RTRs fit the filters and have high quality, the seller may disconnect from that peer as well. The seller can instead direct its time and resources on a different customer who is more willing to buy. In order to continue receiving RTRs, then, a peer must show that it is a good customer. Either the peer must set its filters properly (but this comes at a cost, described in the next section), or it must occasionally speculate on RTRs that it can not use directly, but that it might be able to resell.

In terms of overhead, 3 messages are exchanged for every query forward, as opposed to 1 in the Gnutella protocol. However, we believe that fewer messages will be exchanged in the RTR protocol overall; due to filters, queries can be routed more efficiently to the peers that want them (Section 2.3). In addition, in Section 4 we discuss the possibility of RTR *subscriptions*. Peers bulk-order RTRs through subscriptions, thereby reducing the overhead of negotiation.

[4]Indeed, such a filter results in a *super-peer* relationship in which a query is only forwarded from super-peer to client if the client can definitely answer the query.

2.2 Pricing

In this section, we discuss a simple pricing model for RTRs. We note that, like the pricing of any commodity in the real world, the pricing of RTRs will involve the estimation of many parameters. The purpose of the model is not to provide a straightforward price for the RTR, but to help us understand the factors that influence price, and pinpoint the parameters that need to be estimated.

Model. An RTR has value for two reasons:

- The peer holding the RTR may respond to the query and be selected to upload the file (for which she is paid).
- The peer holding the RTR may resell the RTR to some or all of its neighbors.

Let RTR_f denote an RTR corresponding to a query for file f. We assume any file f has a well-known price $price(f)$. Let N_A be the random variable denoting the income generated by the RTR for peer A, if A owns RTR_f. The income generated for A by holding RTR_f is given by:

$$N_A = S_A + R_A \tag{3}$$

where S_A is the random variable denoting the income gained for selling file f to the querying peer, and R_A is the random variable denoting the income gained from reselling the RTR. The value of RTR_f for peer A is simply $E(N_A)$, the expected value of N_A.

If A responds to the query and is selected to upload the file, $S_A = price(f)$. Otherwise, $S_A = 0$. Let I_A be an indicator variable denoting whether A owns file f ($I_A = 1$) or not ($I_A = 0$). We assume that if A owns the file, then A will respond to the query. Let p_A denote the probability that peer A is picked to upload the file, given that A responded to the query. The expected value of S_A is then:

$$E(S_A) = I_A \cdot p_A \cdot price(f) \tag{4}$$

Let I_N be the indicator variable denoting whether neighbor N buys the RTR from A ($I_N = 1$), or not ($I_N = 0$). Note that the price at which the peer sells the RTR may be different from the price at which it bought the RTR, and it may differ on a per-neighbor basis. Assuming a peer N pays the expected value $E(RTR_f, N)$ for RTR_f, the income R_A generated by reselling the RTR is:

$$E(R_A) = \sum_{N \in nb} E(I_N | E(RTR_f, N)) \cdot E(RTR_f, N) \quad (5)$$

where nb denotes the set of neighbors of A.

Combining equations 3, 4 and 5, we can get the following formula for the cost of an RTR:

$$\begin{aligned} E(N_A) &= E(S_A) + E(R_A) \\ &= I_A \cdot p_A \cdot price(f) + \\ &\quad \sum_{N \in nb} E(I_N | E(RTR_f, N)) \\ &\quad \cdot E(RTR_f, N) \end{aligned} \quad (6)$$

We will now discuss the matter of estimating terms in the above equation.

Parameter Estimation. Parameter I_A, indicating whether peer A owns file f, is known to A – no estimation is needed. The probability that A will be picked to upload the file, p_A, can be estimated in two simple ways. First, A can learn p_A over time by remembering how often it responded to a query for that file, and how often it was chosen to upload. Second, we note that p_A can be estimated as the inverse frequency of f across peers. Peer A can get an estimate of the frequency of its files through sampling, or perhaps statistics offered through a third party.

Estimating how many times an RTR can be resold, and at what price, can be aided by filters. If A knows, via content filter, that neighbor N owns file f, then A can guess with fair confidence that N will purchase the RTR. Furthermore, the lower bound for the RTR should be $E(S_A)$. If A knows that N is a speculator who buys many RTRs, then A may also assume that N will buy RTR_f, but at a wholesale price. If the RTR does not fit N's filters, A can assume N will not purchase the RTR.

Note that by setting content filters, a peer N is "giving away" information about its preferences that can make it a target for higher prices. When a peer decides to set filters, it is generally placing a premium on its load, over its cost. However, also recall that in order to be a "good customer", a peer can not specify very general filters and then rarely buy anything.

2.3 Impact of Buyer/Seller Expectations

Recall that peers can disconnect from their neighbors if they receive too many useless RTRs; hence, it is in a peer's interest to forward only high quality, relevant RTRs to its best ability – otherwise, it loses business. At the same time, peers can disconnect from their neighbors if they never buy any RTRs that they "should" be interested in. Here, we discuss how allowing peers to choose neighbors based on the expectation of "good behavior" can result in good system performance.

Good Sellers. First, let us focus on the effect of peers connecting to sellers based on the quality and relevance of the RTRs received from them. Note that even if a peer does not explicitly set filters, they can still selectively connect to sellers who happen to provide the RTRs that peer is interested in. Hence, peers always have incentive to forward RTRs of high quality, regardless of its neighbors' filters. Here we discuss the potential impact of each quality metric on overall effectiveness of the system.

(1) **Age.** Newer queries are preferred over older queries because they increase the chance that one can reply before the querying peer "closes" the query (e.g., closing the auction, selecting someone to download from, etc). Furthermore, a lower age decreases the chance that the RTR is a duplicate (i.e., received earlier from a different neighbor). Hence, peers should not forward an RTR with an age beyond a given threshold (otherwise, again, its neighbor will drop the connection due to the low quality RTRs). Like the *time-to-live* construct in Gnutella, the effect of dropping old RTRs will limit the size of the network that receives the query. This effect is important in order to prevent inefficient flooding of the network.

(2) **Reputation.** The goal of reputation systems is to allow users to avoid dealing with other users that are malicious (e.g., return corrupt files) or provide poor service (e.g., often die before a transaction is completed). In our scheme, by not forwarding RTRs belonging to low-reputation querying peers, we are ensuring that those who provide bad service are penalized with fewer results for their own queries. Global reputations in P2P systems can be managed, for example, by a system like EigenTrust [5]. Note that a reputation system is not necessary for the RTR protocol. In the absence of a reputation system, the quality of RTRs may be judged by age and relevance only.

(3) **Content.** Peers will connect to other peers who can provide them the RTRs that they want. Thus, content filters (when specified) serve as "routing hints" by which queries can be routed to those

peers containing relevant data, thereby resulting in more *efficient* search. Whether or not they specify filters, peers will connect to those peers who can provide them queries to which they are able to respond. Furthermore, perhaps a peer A is unhappy with the RTRs that its neighbor provides because, for example, the RTRs are too often for hard rock and too little for punk rock. In this case, A can disconnect and eventually find another neighbor who sells more RTRs for punk. Hence, we hope that communities, or "clusters" in the topology, will form around interests, which further lends itself to efficient, effective search.

In summary, by simply acting out of self-interest, peers in our scheme can form a forwarding policy that enforces reputation, limits flooding, and promotes intelligent routing/topology formation.

Good Buyers. Because servicing each customer incurs overhead (e.g., communication, filtering), a peer will not be interested in maintaining a customer who rarely buys anything. If the customer has set appropriate filters, then the fact that it rarely buys is offset by the fact that it will pay a higher price for the few RTRs it does buy. Hence, peers have an incentive to either set the appropriate filters on the incoming stream of RTRs, or else speculate on RTRs more often (that is, buy RTRs for the sole purpose of reselling them).

If a peer chooses to set its filters appropriately, then it is contributing to the overall efficiency of the system – only those queries it can directly use will be forwarded to it. On the other hand, if a peer chooses to speculate more frequently, then it improves the economy and increases the forwarding of RTRs in the system. A greater flow of RTRs will in general result in more query responses, which is one of the goals of the RTR protocol. Clearly, both types of peers – those who wish to speculate and those who wish to set filters – are both needed for the search mechanism to be both efficient and effective. We note that super-peer networks are an example of an efficient search mechanism in which both types of peers are present.

3 Attack Analysis

The main attack possible on the proposed incentive system is that of greedy peers generating bogus queries for services they do not really need. The motivation for doing so is to earn money by selling the initial RTRs to its neighbors.

The primary counter to this type of attack are the notions of reputation and past experience. That is, if a peer is unsatisfied with the RTRs that are being sold by one neighbor, he may disconnect from that peer and reconnect to a new peer. Furthermore, peers are unlikely to buy large quantities of RTRs from vendors with whom they have had no experience. Finally, if the peer gains a good reputation by consistently providing RTRs that lead to uploads, it will be able to charge a premium for its RTRs. Therefore, there is significant incentive not to sell RTRs for bogus queries to one's neighbors, especially in the presence of a global reputation system.

Another way to counter this attack is to make users pay to submit new queries. If there exists a lightweight centralized broker (which is probably required for any system using real money for payment), then a peer can buy a "right to query" (RTQ) token from the broker. An analogy to this idea is paying the owner of a website or billboard to post an advertisement. Each time that peer sends out a query, it must include a new RTQ along with its RTR messages. If the peer reuses a token, then anyone who receives two distinct RTRs (queries) with the same RTQ (right to query), can report the RTRs to the broker as evidence of foul play.

If no centralized broker is available, we can still limit queries via a distributed quota enforcement system (e.g. [9]). Each peer is allotted a "quota" of RTQs (e.g., one query per day). RTQs can accumulate over time if not used. A peer can choose to use the RTQ (regardless of whether it really needs a service), or else sell it to peers who wish to buy it.

Note also that malicious users cannot send out a bogus RTR that lists a different peer as the querying peer, because each RTR is signed by the querying peer.

4 Extensions and Alternatives

Subscriptions. In order to reduce the overhead of the protocol, we can introduce the idea of *subscriptions*, in which peers "bulk order" RTRs over time from its neighbors rather than buying RTRs individually. As with the basic RTR protocol, peers can specify content and quality filters on subscriptions, as well as the volume of RTRs to be included. Sub-

scriptions can also be cancelled; therefore sellers still have an incentive provide the best RTRs it can.

While subscriptions reduce overhead in terms of system messages, they also involve considerably more complex decision-making, such as how to price subscriptions and how to decide which RTRs go in a subscription.

Quality of Service. Several approaches to combatting the free-riding problem have been presented (e.g., [3, 4]). Because the context of the problem is different – a system where peers free-ride versus a system in which peers compete to provide services – the techniques presented in these papers cannot be applied directly to solve our problem.

As an alternative to the RTR protocol, one can imagine applying techniques that have been used to combat the freerider problem to this problem. For example, using the technique presented in [4], we could assign reputations to peers based on the volume of queries they forward, and give them better quality of service (e.g., faster downloads) if their global reputation is high. As another example, peers who forward many queries can connect to other peers who also forward many queries. Therefore the topology forms in such a way that peers who provide many queries also receive many.

However, the first scheme requires that peers provide quality service, at a cost to themselves, to other peers from whom they may not have directly benefited. This goes against our principle of allowing peers to act purely out of self-interest. Both schemes also suffer from the drawback that they do not solve the non-cooperation problem. Competitive peers can continue to drop queries they want to answer, without other peers noticing.

Push Model. In our current model, the willingness of peers to pay for RTRs (because they represent potential business) "pull" queries through the network. A more conventional alternative is a "push" model in which the querying peer pays others to route its queries. An approach similar to the push model has been used, for example, in routing packets at the network level (e.g., [6]). However, these existing works assume parties can be trusted to charge a fee only if they actually do forward a message. In a system of untrusted peers, we would need a way to prove the path of a query message, which will likely incur very high overhead.

5 Future Work

We plan to focus our future work on the RTR Protocol in two main areas.

First, we plan to run simulations to evaluate the effectiveness of the protocol at promoting cooperation. A peer in our system must weigh a number of considerations, such as what types of RTRs it wants to buy and whether it wants to make money by uploading files or selling RTRs. Our current work involves defining a behavior model that is descriptive enough to capture different preferences, while remaining practical enough for efficient simulation.

We also plan to address the non-cooperation problem over different search mechanisms, such as DHTs. DHTs pose new challenges because peers can no longer choose their neighbors, nor do peers have a choice as to which queries they forward. Due to the rigid, deterministic topology and routing policy of DHTs, the "push" model of forwarding queries may be more appropriate in ensuring that messages are properly routed.

In summary, we believe that the *non-cooperation problem* will present a significant challenge to competitive P2P systems. We present one possible protocol that gives peers the incentive to cooperate in the operation of the system, even in the face of competition for providing services.

References

[1] E. Adar and B. Huberman. Free Riding on Gnutella. http://www.firstmonday.dk/issues/issue5_10/-adar/index.html, 2000.

[2] Gnutella website. http://gnutella.wego.com.

[3] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for sharing in peer-to-peer networks. In *Proc. ACM Conference on Electronic Commerce*, 2001.

[4] S Kamvar, M. Schlosser, and H. Garcia-Molina. Incentives for Combating Freeriding on P2P Networks. In *Proc. EURO-PAR*, 2003.

[5] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proc. WWW*, 2003.

[6] T. Roughgarden. Pricing network edges for heterogeneous selfish users. In *Proc. STOC*, 2003.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, 2001.

[8] B. Yang and H. Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. Technical report, Stanford University, 2003.

[9] B. Yang, S. Kamvar, and H. Garcia-Molina. Secure Score Management in P2P Systems. Technical report, Stanford University, 2003.