Determining the Peer Resource Contributions in a P2P Contract

Behrooz Khorshadi Xin Liu Dipak Ghosal
Department of Computer Science
University of California
Davis, CA 95616
{bkhorshadi, liu, ghosal} @cs.ucdavis.edu

Abstract

In this paper we study a scheme called P2P contract which explicitly specifies the resource contributions that are required from the peers. In particular, we consider a P2P file sharing system in which when a peer downloads the file it is required to serve the file to upto N other peers within a maximum period of time T. We study the behavior of this contribution scheme in both centralized and decentralized P2P networks. In a centralized architecture, new requests are forwarded to a central server which hands out the contract along with a list of peers from where the file can be downloaded. We show that a simple fixed contract (i.e., fixed values of N and T) is sufficient to create the required server capacity which adapts to the load. Furthermore, we show that T, the time part of the contract is a more important control parameter than N. In the case of a decentralized P2P architecture, each new request is broadcast to a certain neighborhood determined by the time-to-live (TTL) parameter. Each server receiving the request independently doles out a contract and the requesting peer chooses the one which is least constraining. If there are no servers in the neighborhood, the request fails. To achieve a good request success ratio, we propose an adaptive scheme to set the contracts without requiring global information. Through both analysis and simulation, we show that the proposed scheme adapts to the load and achieves low request failure rate with high server efficiency.

1 Introduction

Some recent measurements of the Internet traffic shows that P2P traffic, particularly generated by file sharing applications, is starting to dominate the traffic in the Internet [1]. Example of successful P2P file sharing applications include KaZaa [2], Gnuttella [4], and BitTorrent [5] among others.

The success and popularity of P2P is due to its inherent scalability and ad hoc and on-demand nature. The resources in a P2P system consists of contributions by the participating peers. Consequently, the stability and the scalability of the network depends on the peer resource contributions. Thus, in this paper, we propose and study a simple yet effective resource management scheme that we refer to as P2P contract. In this scheme, the resource contributions are explicitly requested from the peers in exchange of service from the P2P system. We consider a P2P file sharing system in which, when a peer downloads the file it is required to serve the file to upto N other peers within a maximum period of time T. We study the behavior of the proposed contract scheme in a centralized and a decentralized P2P network.

We note that the contract scheme is not an incentive or enforcement scheme by itself. Instead, assuming that an enforcement scheme is available, we propose and analyze the contract scheme for good performance and adaptability. Any enforcement/incentive schemes discussed in the literature can be used in combination with the proposed contract scheme. Furthermore, due to the simple structure of the contract scheme, a simple enforcement could be as follows: the server holds a small yet critical piece of the file until the contract is fulfilled, e.g., the the time period T has elapsed.

In the centralized case, all new requests are forwarded to a central controller which hands out the contract along with a list of peers from where the file can be downloaded. In the case of a decentralized P2P network, each new request is broadcast to a certain neighborhood determined by the time-to-live (TTL) parameter. Each server receiving the request independently hands out a contract and the requesting peer chooses the one which is least constraining and downloads the file from the corresponding server. If there are no servers in the neighborhood, the request fails. In the case of the decentralized system, to achieve a good request success ratio, we propose an adaptive scheme to set the contracts without requiring global information. In particular, each active peer node maintains a moving average estimate of the load by keeping track of the number of queries observed over time. The initial estimate is the average of the

Proceedings of the 2005 Second International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P'05)
0-7695-2417-6/05 $20.00 © 2005 IEEE
0-7695-2426-5/05 $20.00 © 2005 IEEE

load received from the servers from which node receives contracts. When the active server gives out a contract it sets T, the time part of the contract to be inversely proportional to the current estimate of the load.

The key contributions of this work are the following:

1. We analyze the performance of the (N, T) contract in the centralized case.

2. For the decentralized case, we propose and analyze an adaptive contract scheme that stabilizes the system (i.e., to keep the query failure rate low) without requiring global information.

3. In both cases, it is shown that T is a more important parameter than N in terms of controlling the service capacity of the system. On the other hand, the N part of the contract provides a contributing peer an upper-bound on the required bandwidth contributions.

The remainder of this paper is organized as follows. In Section 2, we discuss P2P contracts and its implementation in a centralized and a distributed P2P network. We also briefly discuss the simulation model. In Section 3, we study the properties of P2P contracts in a centralized network. In Section 4, we present the performance of P2P contracts in a decentralized network under different methods for setting the contract. In Section 5 we briefly review the related work, followed by the conclusion in Section 6.

2 P2P Contract

A framework architecture for implementing P2P contracts for a file sharing system consists of three components: **a seed**, **active peers**, and **clients peers**. The seed is the primary node for the file and exists permanently. Under "low-demand" conditions, the seed functions as a traditional server and the client peers function as traditional clients. Under "high-demand" conditions, the seed draws a contract. In it, access is granted only to client peers that agree to serve files they retrieve to upto N other peers within a maximum time period T seconds. In other words, a peer is released from its obligation if it has held onto the file for T seconds but less than N other peers were directed to it and vice versa. When a client peer is ready to serve he file it becomes an active peer.

Note that a peer may begin to fulfill the aspect of the contract while it is downloading the file, as in BitTorrent [5]. This can actually help enforce the contract. In the analysis, however, we will consider the contract period and any service to begin only after after it has received the file.

2.1 Implementation Strategies

In this paper we focus on the resource management issues[1] in P2P contracts. A key issue is determing the terms of the contract, i.e., the values of N and T. When peer makes a request, the contributions required from the peer would be such that the total number of requests satisfied is maximized. This implies that the total serving capacity of the P2P system is equal to the demand. The key issue is that knowledge of the demand depends on the P2P system.

In a centralized architecture there is only one seed with the target file. All client peers know the location of the seed and they contact it directly to receive the file. Additionally, the seed and active peers can only serve a predefined number of files simultaneously defined by the server concurrency limit. When the seed receives a query, it has two possible responses: 1) give the file to the client peer for free or 2) offer a contract to the client peer. The seed monitors the the overall demand by measuring the number of requests from different client peers. If the demand is over a pre-defined threshold, then the seed issues a contracts. If the demand is less than the threshold, then the demand is low and the seed can afford to serve the file for free. In either case, the seed's decision to serve the file depends on the available capacity. If the seeds concurrency limit has been exceeded, the request is directed to an available active peer.

In the decentralized implementation there is also one seed. However, its address is not globally known. Peers begin flooding queries looking for seed/active peers. If the query reaches an active peer or the seed, then the active peer or the seed will hand out a contract to the requesting peer. Note that a requesting client peer may receive multiple contracts which may be different depending on the policy used by the active peers to determine the contract. The client peer will take the least constraining contract. If the request does not find any active peer (or the seed) within the neighborhood, the request fails. The manner in which an active peer decides on what contract to offer is discussed in Section 4.

2.2 Simulation Model

The simulation model we developed to study the P2P contract scheme was inspired in part by John Conway's Game of Life [14], in which a board is equally divided into a number of cells, where a cell represents a peer. For this model, we define a peer's neighbors as consisting of the cells in every direction including diagonals (a "Moore" neighborhood). Time is slotted and each slot is referred to a generation. A simulation consists of a certain number of generations.

[1] Issues related to security and enforcement of the contract are not addressed in this paper.

Throughout the simulation, peers behave according to a finite state machine which consists of the following states:

1. Non-participatory: in this state the peer does not want to obtain the target file and hence does not generate any query. It will, however, route queries.

2. Client Peer: in this state the peer is attempting to download the target file and is sending queries to search for it.

3. Active Peer: a client peer which accepted the contract becomes an active peer after it has received the file. In this state the peer serves the target file to other client peers to fulfill its contract.

4. Seed: A permanent server that permanently exists and has the file.

Snapshots of the simulation is shown in Figure 1. It shows two consecutive generations with the locations of the various entities; seed (S), active peer (AP), and client peer (CP).

3 Analysis of P2P Contract in a Centralized Architecture

As in the simulation model, we consider that time is slotted and refer to a slot as a generation. For the simplicity of the analysis, we make the following assumptions. First, all requests arrive at the beginning of a generation. Second, each active peer whose contract has not yet expired can serve only one user in each generation. Third, a user's request is fulfilled in one generation, i.e., the file download time is less than a generation.

Let M denote the expected number of available active peers in a generation and let K denote the average number of requests that arrive in a generation. We want to derive the relationship between M, (N, T), and K in a stable system. A stable system refers to the case where the demand stabilizes after a period of burstness. In the following analysis, we consider the steady state behavior and in particular consider the case where $T \geq N \geq 2$; if $N = 1$, then the system will not be stable when the demand fluctuates. Note that if $N > T$, then the contract (N, T) is equivalent to the contract (T, T).

Let $x(i, j)$ denote the average number of active peers whose contract has i remaining services and will expire in j generations. We consider the end of a specific generation in which there are K new requests. If all the requests are served, then K client peers agree to the contract (N, T) and thus will become active peers. Thus $x(N, T) = K$ and $x(N - i, T) = 0$ for all $i > 0$. At each generation, each active peer is chosen to serve a request with probability p where $p = K/M$. Thus, we have

$$x(N-i, T-j) = K \binom{j}{i} p^i (1-p)^{j-i}, \quad 1 \leq i \leq j.$$

Then, the total number of active peers can be calculated as

$$\begin{aligned} M &= \sum_{j=0}^{T-1} \sum_{i=0}^{\min(j, N-1)} x(N-i, T-j) \\ &= TK - \sum_{j=N}^{T-1} x(0, T-j), \end{aligned}$$

where $\sum_{j=N}^{T-1} x(0, T-j)$ counts for the number of active peers who have fulfilled N requests and left the system before T generations. Thus, when $N \geq T$, we have $M = TK$. Further, when $N \geq 3$, the probability that an active peer serves N users within T generations is very small. Thus, the dominant reason for an active peer to fulfill its contract is that T generations have expired. Thus, for $N \geq 3$,

$$M \approx KT. \tag{1}$$

This is verified through both numerical and simulation results.

Next, consider the case $N = 2$ and $T \geq N$. We have

$$\begin{aligned} M &= \sum_{j=0}^{T-1} K(1-p)^j + \sum_{j=1}^{T-1} Kjp(1-p)^{j-1} \\ &\approx K \frac{2(1-e^{-pT}) - pTe^{-pT}}{p} \\ &\approx \frac{KT}{1.146}. \end{aligned}$$

Figure 2 shows the relationship between M and T for different values of N where $K = 10$. The x-axis is the time part of the contract, i.e., T, and the y-axis is the average number of active peers, obtained numerically. The case $N = \infty$ represents the contract (∞, T), which is equivalent to contract (T, T). In the figure, the four curves from bottom-up are $N = 2, 3, 4, \infty$, respectively. We can see that the case $N = 3$ is close to the case $N = \infty$ and the difference diminishes as N increases. This is confirmed by the simulation results, shown in Figure 3, where $K = 30$. Similarly, the four curves from bottom-up are $N = 2, 3, 4, \infty$, respectively.

In summary, the key observations are as follows:

1. M is a linear function of K. Thus, the system has a self-clocking property. For a fixed (N, T), the number of active peers increases linearly as the number of requests increases. The higher the demand, the larger the number of active peers.

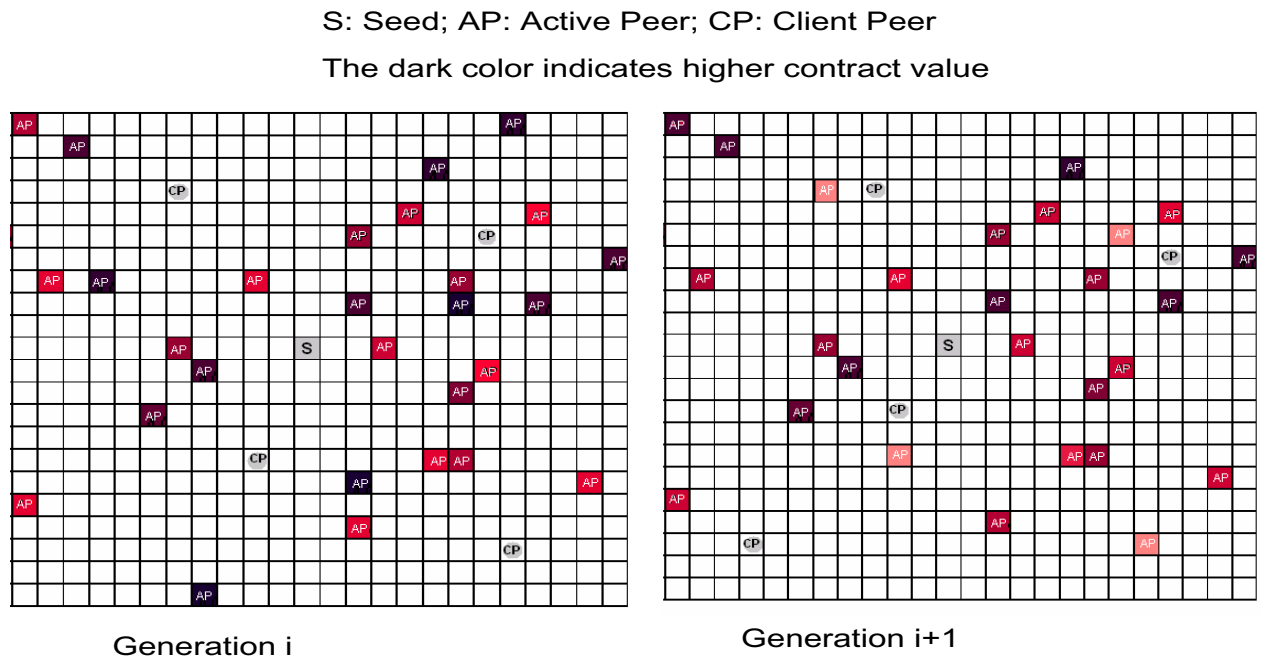

Figure 1. Snapshot form the simulation showing the seed, active peers, the client peers in two consecutive generations. The intensity of the cell correlates with the amount of remaining time part of the contract.

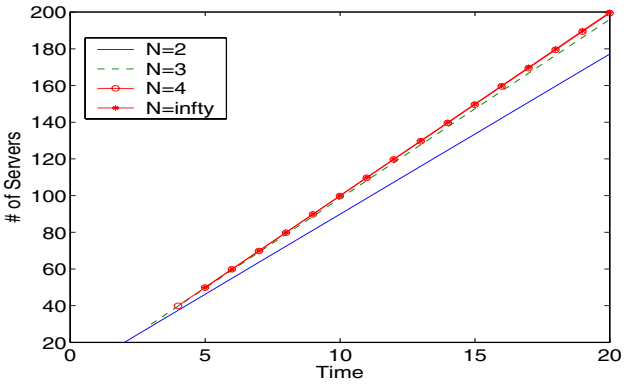

Figure 2. The effect of N and T on the expected number of active server M.

2. M is (almost) a linear function of T when $N \geq 4$. Thus, T is an important parameter. The larger the value of T, the larger the number of available active peers, the higher the probability of finding a close active peer, and thus the lower the communication cost of file sharing.

3. The impact of N is insignificant when $N \geq 3$. However, the N part of the contract provides an active peer an upper-bound on the required bandwidth contributions, which is at most $N \times$ file-size).

In the centralized scheme, when there is a sudden burst of requests, the system can adapt quickly to the burst. Consider the case where there is only one seed. The time needed to serve a burst of size L is roughly $\log_2(L+1)$. Again, for $N \geq 4$, the time difference between a finite N and an infinite N is negligible.

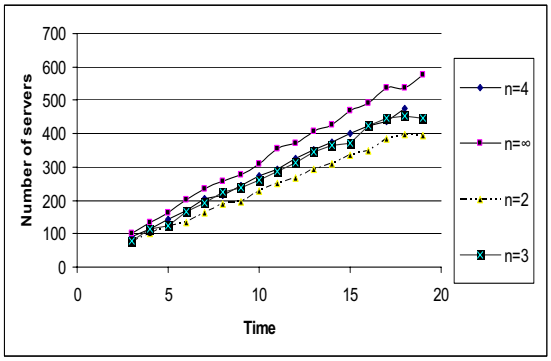

Figure 3. Simulation results of the number of active peers on the expected number of active peers M as a function of N and T.

4 P2P Contract in a Decentralized Architecture

In a decentralized architecture there is no central controller where a request can obtain information regarding the locations of the seed and/or the active peer. Instead, each client peer sends a query with Time-to-live (TTL) parameter to its neighborhood requesting for the file. Each active peer, including the seed, that receives the request will generate a contract. As each active peer may have a different view of system, the client peer may receive different contracts. The client will accept the one that is least constraining and download the file from the corresponding active peer. If the query does not reach the seed or an active peer before the TTL expires, the query fails. One major concern is how to stabilize the system. We consider the system to be stable if the query success probability is high. In this section, we will focus on the stabilization of the system under our proposed contract scheme. First, we will show how to choose a fixed contract to stabilize the system assuming global information. Second, we propose a distributed scheme that tracks the system without requiring global information.

4.1 Stabilizing the System

For a query initiated at node (i_0, j_0) the number of cells reached by the query depends on the value of the TTL. Let $N(TTL)$ be the number of cells that receive the request. If Manhattan-distance is used to calculate the distance, then all nodes (i, j) where $|i - i_0| + |j - j_0| \leq TTL$ in the grid receive the query. On the other hand, if euclidean distance is used, then all nodes, where $\sqrt{(i - i_0)^2 + (j - j_0)^2} \leq TTL$, receive the query. Ignoring the boundary effect, a query is propagated to roughly $2TTL^2$ nodes in Manhattan distance and πTTL^2 in euclidean distance.

Let p be the probability that a node is an active peer. For this analysis, we assume that active peers are uniformly distributed; i.e., each node is an active peer with probability p independent of client peers. This assumption holds when client peers generate request independently and uniformly, and (almost) all requests are satisfied. Thus for a given value of TTL, the failure probability of the query generated by a client peer, denoted by P_f, is given by

$$P_f = (1 - p)^{N(TTL)}. \tag{2}$$

In other words, a query fails if none of the $N(TTL)$ neighbors are either active peers or seed.

Consider a fixed contract (∞, T); i.e., a time-only contract. This contract means that an active-peer will remain active for T generations and will serve all requests during the time interval. Given that the objective is to keep the query failure rate low, the expected number of active peer in a stable system is $M(T) = \lambda T(1 - P_f) \approx \lambda T$. Substituting in Eq. (2), we obtain

$$P_f \approx (1 - \lambda T)^{N(TTL)} \approx \exp(-N(TTL)\lambda T).$$

Let ϵ be the threshold probability of P_f. We have $-N(TTL)\lambda T \leq \ln(\epsilon)$. Given the values of λ and TTL, we have

$$T^* = \frac{-\ln(\epsilon)}{N(TTL)\lambda}. \tag{3}$$

Thus, a contract (∞, T) should stabilize the system if $T \geq T^*$. Furthermore, we note that $M(T^*) \approx (-\ln \epsilon)/N(TTL)$. Thus, the value of the contract T depends on both λ and TTL and is inversely proportional to the value of λ and TTL^2.

The above analysis is verified through our simulation. In the simulation, we consider one seed in a grid of 100×100 nodes of which 66% of them generate requests. We consider different load levels at which the potential nodes generate requests. For example a load of 2% means that on average there are $100 \times 100 \times 2/3 \times 2\%$ requests per generation. Each client peer with a successful request becomes an active peer for T generations thereby propagating the active peers in the network. The lower the load, the slower the propagation.

Parts (a) and (b) in Figure 4 show the number of active peers and the failure rate, respectively, as a function of the TTL. The number is averaged from the 50th generation to the 1000th generation. For a reasonable value of TTL, (i.e., TTL>8), the system stabilizes, i.e., the failure rate is low.

The average number of active peers is the same for different loads, as expected. We should note that for different loads, the average contract is different, which is not plotted here. Note that for smaller values of TTL, the system can also stabilize, but takes much longer. That is the reason that the number of active peers is small for small TTL when the load is low. It also explains why the failure rate for a lower load is higher (for larger TTL values). Note that we count the various performance metrics from the 50th generation at which point, for a lower loads, the system is still in the transient phase. As the TTL is increased, the number of active peers decreases, as expected. This implies that the system adapts to both load and TTL.

4.2 Adaptive Contracts Without Global Information

In the previous section, we discussed how a fixed contract can stabilize the system. In practice, an active peer has no global knowledge regarding either λ or TTL. In this section, we propose a distributed contract scheme without gathering global information.

We note that from Eq. (3) that the value of T depends on $\lambda N(TTL)$. In other words, we need an estimate of $\lambda N(TTL)$, which is obtained through a moving average algorithm explained as follows. Consider a node that has become an active peer at time t_0. Along with its contract, its peer passes an estimate of $\lambda N(TTL)$, denoted as $\hat{\eta}(t_0)$. At the end of each generation, the active peer performs the following update:

$$\hat{\eta}(t+1) = \omega\hat{\eta}(t) + (1-\omega)y(t),$$

where $y(t)$ is the number of requests the active peer received during the current generation. If it receives a request, the active peer doles out a contract which is given by

$$\hat{T} = \frac{-\ln\epsilon}{\hat{\eta}(t)}.$$

Note that $E(y(t)) = \lambda N(TTL)$, and thus $\hat{\eta}(t_0)$ is an unbiased estimate of $\lambda N(TTL)$. Because of Jensen's inequality, we have $E(1/X) \geq 1/E(X)$ where X is a positive random variable. Thus,

$$E(\hat{T}) = E\left(\frac{-\ln\epsilon}{\hat{\eta}(t)}\right) \geq \frac{-\ln(\epsilon)}{N(TTL)\lambda} \geq T^*.$$

The above scheme is adaptive and lends itself to a totally distributed implementation in which each active peer gathers only local information. Furthermore, the above adaptive scheme generates a more conservative contract in the following sense: if the original scheme with fixed contracts based on known values of λ and TTL stabilizes the system, then the above adaptive system does as well.

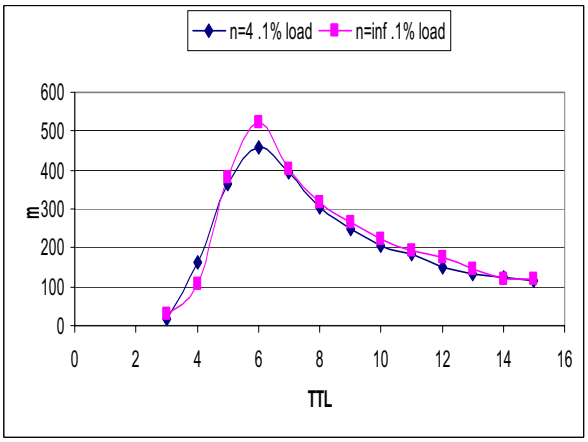

Figure 5. The average number of active peers m as a function of TTL for $N = 4$ and $N = \inf$.

The results are shown Figure 4 which also compares with the previous case where both the values of λ and TTL are given. In the figure, the legend "moving average" is the performance of the adaptive scheme without global information. We can see that it performs closely to the case where both the values of λ and TTL are given. This implies that the moving average gives a good estimate of $\lambda N(TTL)$.

In summary, the proposed adaptive contract scheme can stabilize the system without requiring global information or additional signaling overhead. The performance of the proposed scheme is close to that of the ideal case.

4.3 The impact of N

In the previous analysis and simulation, we only considered the time part of the contract; i.e., contract (∞, T). In this section, we study the impact of N in the contract. In the centralized system, we noticed that the value of N is not as important as that of T for relatively large N ($N \geq 4$). The same conjecture is verified via simulations. In Figure 5, it is plotted the average number of active peers when $N = 4$ and $N = \infty$, where we observe little difference. Further, the failure rates in these two cases are also very close. Thus, we conclude that the T part of the contract is more important than the N part of the contract, as in the centralized case.

4.4 Random TTL

In this case, we let each request randomly chooses a TTL value. In the simulation, TTL values are uniformly distributed between 4 and 12. Note that the active peers still estimate the load as we discussed earlier. The simulation

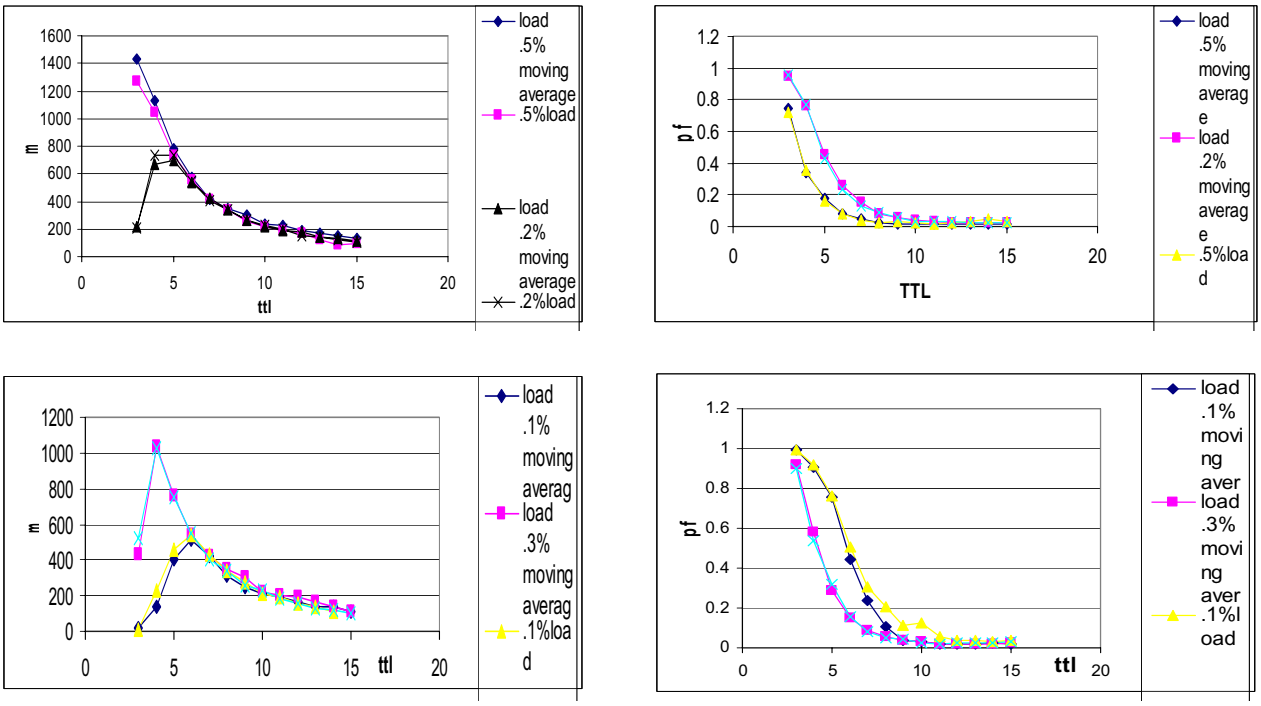

Figure 4. The comparison of the ideal case with the proposed moving average scheme with respect to number of active peers M and failure probability P_f for different load and different TTL values.

result is shown in Figure 6. Although the number of active peers generated in this case is close to the case where we have a fixed TTL=8, the failure rate is much higher. This is due to the fact when the value of TTL is small (small compare to the number of active peers in the system), the failure rate is very high. We also run a simulation where TTL is uniformly distributed between 8 and 15. In that case, the failure probability is kept below 1%.

5 Related Literature

Providing incentives for cooperation in P2P systems is an important area of research [13, 10, 11]. In altruistic approach users donate resources at their discretion, with no negative consequences for not doing so. Napster [3], Gnutella [4], among others, showed that such systems do work in practice. Unfortunately, free riders who may not derive much utility from the service are not discouraged from consuming resources, at the expense of the quality of service to users who value the service highly. Empirical evidence shows that as much as 70% of the peer are free riders [6].

The second approach uses micro payments to provide

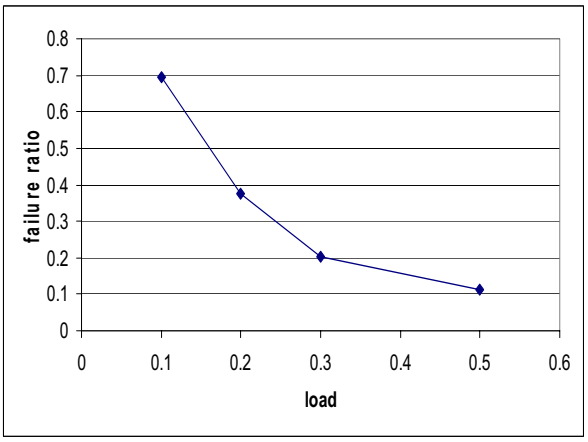

Figure 6. The average number of active peers m as a function of load with random TTL.

economic incentives for resource contributions. Systems that use this approach belong to one of two categories depending on the type of currency used. A systems of the first type, such as Mojo Nation, uses an internal currency that cannot be redeemed for goods provided outside of the system. Participants of such a system earn credit every time they contribute resources to the system and they use earned credit to obtaining service from it. Because users cannot obtain service without credit, free riders do not exist. A system based on an external currency, such as Popular Power, differs in that the currency can be redeemed for goods outside of the system. Thus, participants in such a system have incentive to provide resources even if they have no need to obtain services. Free riders do not exist in such systems either [8].

Another approach is the tit-for-tat scheme [12]. In BitTorrent protocol [5], the set of peers attempting to download the file do so by connecting to several other peers simultaneously and download different pieces of the file from each other. To facilitate this process, BitTorrent uses a centralized software called the tracker which provides a new requesting peer a list of peers that have also requested the file. The downloader then establishes a connection to these other peers and download different pieces from each other following novel piece selection algorithms. Each peer is allowed to upload only to a fixed number (default is four) at a given time. Which peers to upload (process is called unchoke) is determined by the current downloading rate from these peers, i.e., each peer uploads to the four peers that provide it with the best downloading rate even though it may have received requests from more than four downloaders. Our paper builds upon the work in [13] which analyzes the BitTorrent protocol and suggests that the performance can be further improved if the downloaders remained in system for some time after the download was complete.

6 Conclusions

In this paper we propose and study an (N, T) P2P contract scheme. In particular, we consider a file sharing network in which when a peer downloads the file it is required to serve the file to upto N other peers within a maximum period of time T. We analyze the performance of the (N, T) contract scheme in a centralized system. Further, we propose an adaptive contract scheme in a decentralized system, which is shown to maintain a low request failure ratio with high active peer efficiency without global information. We show that in both cases, T is a more important parameter than N for $N \geq 4$. On the other hand, N is desirable because it provides an upper bound on the amount of bandwidth resource an active peer is required to contribute.

Contract enforcement is not discussed in the paper. Protocols developed in the literature, such as a credit system, can be used to enforce the contract. The contribution of the paper is to provide a specific resource contribution scheme and to analyze its performance. As future research, we will study be system when there are contract breaches.

References

[1] Top applications (bytes) for subinterface: SD-NAP traffic, 2002. www.caida.org/analysis/workload/byapplication/sdna

[2] Kazaa: URL http://www.kazaa.com.

[3] Napster. URL http://www.napster.com.

[4] Gnutella. URL http://www.gnutella.com.

[5] B. Cohen. Incentives build robustness in bittorrent, May 2003. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf.

[6] E. Adar and B. Huberman: Free Riding on Gnutella. First Monday, 2000.

[7] G. de Veciana and X. Yang. Fairness, incentives and performance in peer-to-peer networks. In the Forty-first Annual Allerton Conference on Communication, Control and Computing, Monticello, IL, Oct. 2003.

[8] P. Golle, K. Leyton-Brown, and I. Mironov: Incentives for Sharing in Peer-to-Peer Networks. ACM Conference on Electronic Commerce, 2001.

[9] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster: To Share or not to Share' An Analysis of Incentives to Contribute in Collaborative File Sharing Environments. Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA 94709, 2003.

[10] K. Lai, M. Feldman, I. Stoica, and J. Chuang: Incentives for Cooperation in Peer-to-Peer Networks. Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, 2003.

[11] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina: Incentives for Combatting Freeriding on P2P Networks (Research Note). Euro-Par 2003, Klagenfurt/Austria, August 26 - 29, 2003.

[12] K. Kong and D. Ghosal: Mitigating Server-Side Congestion Using Pseudoserving. IEEE/ACM Transactions on Networking, August 1999.

[13] D. Qiu and R. Srikant. Mdeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks, Proc. ACM SIGCOMM, Portland, OR, Sept. 2004.

[14] K. Sigmund: Games of Life: Explorations in Ecology, Evolution, and Behavior. Penguin, 1995.