University of Würzburg
Institute of Computer Science
Research Report Series

Comparison of Crawling Strategies for an Optimized Mobile P2P Architecture

Tobias Hoßfeld, Andreas Mäder, Kurt Tutschku,
Phuoc Tran-Gia[1], Frank-Uwe Andersen[2], Hermann de Meer,
Ivan Dedinski[3]

Report No. 356 June 2005

[1] University of Würzburg
Department of Computer Science
Am Hubland, D-97074 Würzburg, Germany
{hossfeld,maeder,tutschku,trangia}@informatik.uni-wuerzburg.de

[2] SIEMENS AG
Siemensdamm 62, 13623 Berlin, Germany.
frank-uwe.andersen@siemens.com

[3] University of Passau, Chair of Computer Networks and Computer Communications
Innstaße 33, 94032 Passau, Germany.
{demeer,dedinski}@fmi.uni-passau.de

Comparison of Crawling Strategies for an Optimized Mobile P2P Architecture

Tobias Hoßfeld, Andreas Mäder, Kurt Tutschku, Phuoc Tran-Gia
University of Würzburg
Department of Computer Science
Am Hubland, D-97074
Würzburg, Germany
`{hossfeld,maeder, tutschku,trangia}@informatik.uni-wuerzburg.de`

Frank-Uwe Andersen
SIEMENS AG
Siemensdamm 62, 13623 Berlin, Germany.
`frank-uwe.andersen@siemens.com`

Hermann de Meer, Ivan Dedinski
University of Passau, Chair of Computer Networks and Computer Communications
Innstaße 33, 94032 Passau, Germany.
`{demeer,dedinski}@fmi.uni-passau.de`

Abstract

Abstract: Mobile networks differ from their wireline counterparts mainly by the high costs for air transmissions and by the mobility of the users. A new entity, denoted as the *crawling peer*, is suggested in order to optimize the resource mediation mechanism for a mobile P2P file sharing application. The crawling peer locates content on behalf of mobile peers. It is placed in the wireline part of the mobile network and thus, does not suffer from the above mentioned restrictions. The crawling peer is part of a comprehensive mobile P2P file sharing architecture [1] which is based on the popular eDonkey file sharing application. The performance of three querying strategies of the crawling peer is investigated with respect to banning at the index servers and the response time of requests, i.e. the time to find a file. The results show that the selection of an appropriate request strategy for the crawling peer maximizes the probability of locating a file while the probability to be banned by an eDonkey index server is minimized.

Keywords: P2P, mobile network architecture, resource mediation

1 Introduction

Despite the economical difficulties the last years have seen two success stories in networking. Cellular mobile networks have gained tremendous popularity, e.g. the number of GSM subscribers rose in Germany within ten years from 1.76 million (1993) to 63.5 million (2003) [2]. A similar extreme growth has only been matched by peer-to-peer (P2P) file sharing services like Napster, eDonkey/eMule or BitTorrent. Within the five years since the start of Napster, they have evolved to the most dominant application in the Internet in terms of transmission volume [3, 4]. A continuation of the GSM success story by UMTS was expected but, at least in Europe, is still evolving. This fact comes mainly from the absent of services and applications for this technology [5]. UMTS network operators are currently looking for applications which do both: *a)* exploit, qualitatively and quantitatively, the potential of the UMTS technology and *b)* motivate the user to adopt the new technology. In that way, *mobile P2P file-sharing* is an interesting candidate for such an application.

Mobile networks differ from wireline networks mainly by the limited capacity of radio channels and by the mobility of the users. The high costs of air transmission ask for a minimization of any signalling. The user mobility results in rapidly varying on-line states of users and leads to the discontinued relaying and buffering of signalling information. This can be achieved for example by entities which on behalf of others store content, i.e. *proxies*, or entities which locate information, i.e. *crawlers*.

P2P is a highly distributed application architecture where equal entities, denoted as *peers*, voluntarily share resources, e.g. files or CPU cycles, via direct exchange. The advantages of P2P services are the autonomous, load-adaptive, and resilient operation of these services. In order to share resources, the peers have to coordinate among each other which causes significant amount of signalling traffic [6, 7]. P2P applications support two fundamental coordination functions: *a) resource mediation* mechanisms, i.e. functions to search and locate resources or entities, and *b) resource access control* mechanisms, i.e. functions to permit, schedule, and transfer resources. In particular, mediation functions are responsible for the high amount of signalling traffic of P2P services. The *overall performance* of P2P applications is determined by the individual performance of the basic P2P control functions.

A P2P file swapping user is mainly interested in a short exchange time for files. Therefore the mediation time, i.e. the time to locate a file, and the time to exchange the file has to be minimized. A reduction of the mediation time becomes of even greater importance in mobile networks. The expected content of mobile P2P file sharing services is of small or size, e.g. ring tones or images. The user, however, is expecting an overall performance related to the size of the content.

The reduced mediation traffic on the air interface, the discontinued signalling, and the short mediation times needed for mobile P2P file sharing networks ask for new architecture solutions for these kinds of P2P services. An efficient solution might state the use of new entities, in particular of the so-called *crawling peer*. The crawling peer is placed in the wired part of the mobile network and locates files on behalf of mobile peers. Research on the mediation performance in P2P systems is fundamental. The crawling peer might be an alternative to highly distributed concepts such as *Distributed Hash Tables*, as used in Chord [8], or *flooding concepts*, as used in Gnutella [9].

In this paper we investigate the performance of a crawling peer as introduced in [1]. Section 2 describes the Mobile P2P architecture. In Section 3, we measured typical values of real eDonkey index servers which are used as input parameters in our investigation. The considered network and the crawling peer are modeled in Section 4. Numerical results with analytical approximations are given in Section 5 and Section 6 gives a conclusion and outlook on future work.

2 Mobile P2P Architecture

The suggested mobile P2P architecture for third generation mobile networks first introduced in [1] and is depicted in Figure 1. The suggested concept is based on the architecture of the popular eDonkey P2P file sharing application [10, 11] and was enhanced by three specific entities: the *cache peer*, the *mobile P2P index server*, and the *crawling peer*.

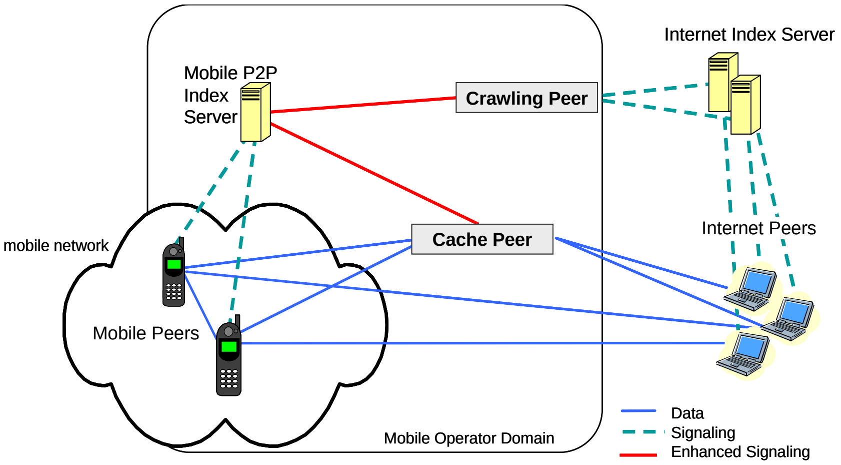


Figure 1: Architecture concept for a P2P file-sharing service optimized to mobile networks

The *cache peer* is a modified eDonkey peer that can be triggered to download often requested files and then offers these files to the community. It is located in the wireline and operator controlled part of the mobile network. The cache peer is assumed to have a high-speed Internet connection and sufficient large storage capacity. The application of the cache peer reduces the traffic caused by popular content on the radio interface since the file is served by a wireline peer and not by a mobile peer. In addition, mobile peers are served in average faster and more reliable [12]. Also the traffic to external networks is reduced.

The *mobile P2P index server* is a modified eDonkey index server. It tracks the frequently requested content and triggers the cache peer to fetch it. The mobile P2P index server advertises the cache peer to other mobile peers which search for the popular content. The mobile P2P index server hides all other mobile sources if the cache peer can provide the file. Thus, the mobile P2P index server forces the mobile peers to download the file from the cache peer.

The *crawling peer* is also located in the wireline part of the suggested mobile P2P architecture and searches content on behalf of other mobile peers. The crawling peer can locate files even when a mobile peer is not online. As a result, the search traffic is shifted to the wireline part of the network and the radio links are relieved from signalling traffic. It has to be noted that a mobile peer should not be allowed to contact external eDonkey server. If a mobile peer would contact external index directly then the mobile P2P index server can not track the files requested by mobile peers, that would result in less effective caching. Hence, the crawling peer is not queried directly by mobile peers. The mobile P2P index server triggers the crawling peer to search for content if it does not know the location of a file.

In general, an eDonkey peer, either a wireline peer or a mobile peer, can send search queries in a *local* or a *global* way. Local queries are restricted to the index server only to which the requesting peer is connected to. Global queries are send by the peer to multiple index server sequentially until sufficient sources of the requested content are found. If a peer starts a global query, it causes additional signalling traffic proportional to the number of index servers visited. The order of contacting index server is arbitrary and does not consider any properties of the

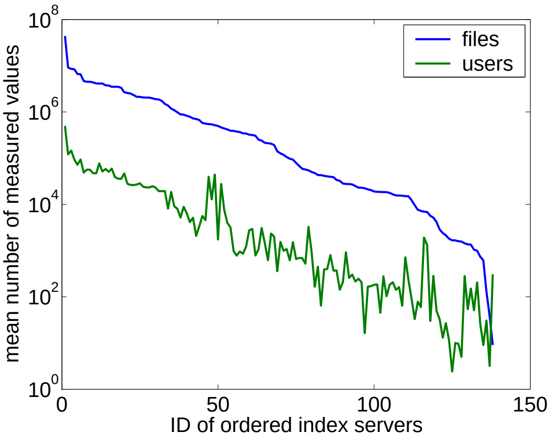


Figure 2: Mean number of files and of users per index server

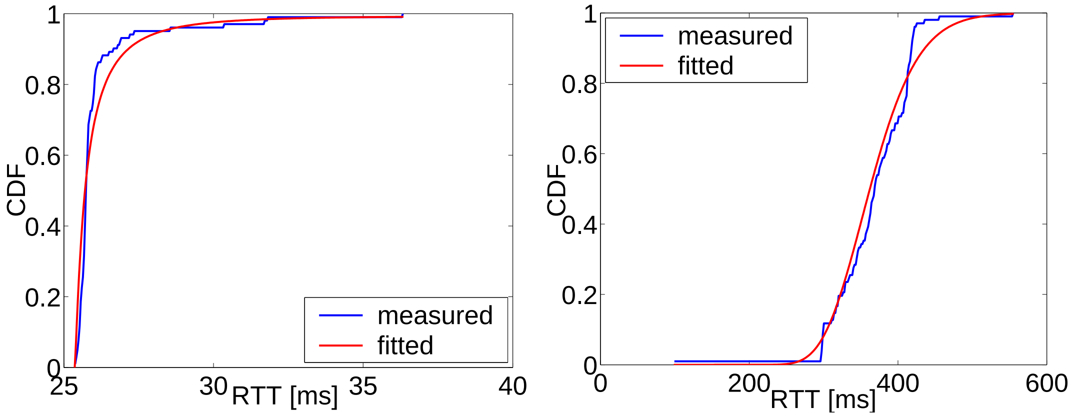
servers, e.g. number of files currently indexed. A more intelligent search strategy leads to significant improvements. The crawling peer might gather statistics about the index servers and preferably contact the servers that offer the most files first. This gives a better chance to find any results faster. In addition, a fast locating of files would also lead to reduced signalling traffic for global queries.

When executing an intelligent search strategy, the crawling peer has also to consider the *credit point system* in the eDonkey network [13], which prevents a peer of issuing too many search queries to a certain index server. The crawling peer should query only index servers for which it has enough credit points. If there are no credit points at all, the query should be blocked, or delayed. An additional option that can increase the search capacity is to have more than one crawling peer.

3 Measurements of Index Server Information

The performance evaluation of the crawling peer and its search strategies, cf. Section 5, needs basic performance values for the eDonkey index server behavior. Therefore, the average number of connected peers to an eDonkey index server and the number of registered files on the server have been measured. Additionally, the round trip times between a host at the University of Würzburg and the index servers were idnetified by sending ICMP packets with the standard Linux ping tool. A list of public eDonkey servers can be found in the Internet at [14], where information on the number of peers and files for each of the servers are updated every 15 minutes. The measurements were performed during April 2004.

In total, $N = 138$ different index server are investigated. The measured number of registered files at index server $i \in \mathcal{I} := \{1, \cdots, N\}$ is denoted by $\widetilde{F}_i$, the measured number of registered users by $\widetilde{U}_i$, and the measured round trip times by $\widetilde{R}_i$. The collected values revealed that the largest index servers host up to 500,000 peers with 44 millions of files, whereas about half of the servers have less than 1,000 connected peers with less than 150,000 files. In order to identify an index server, we use an ID. The index servers are decreasingly sorted by the mean



(a) Largest eDonkey server with ID 1

(b) Chinese server with ID 48

Figure 3: Cumulative distribution function of measured and fitted RTTs

number $\mu(\widetilde{F}_i)$ of registered files. The ID of an index server reflects its position in the sorted list, $\forall i, j \in \{1, \cdots, N\} : i < j \Leftrightarrow \mu(\widetilde{F}_i) > \mu(\widetilde{F}_j)$. Figure 2 shows the mean number $\mu(\widetilde{F}_i)$ of files and the mean number $\mu(\widetilde{U}_i)$ of users registered to an index server with the index server ID on the abscissa.

The time for answering a search request by an index server is modelled by using the round trip time. Figure 3 shows the cumulative distribution function (CDF) of the round trip times for two public index servers, the largest index server in the eDonkey network with ID 1 and a Chinese eDonkey server with ID 48. The latter has about 12,000 users with 540,000 shared files. The blue curves indicate the measured values which we fit by a lognormal distribution. In detail, the round trip R_i of an index server i is modelled by (1), whereas $\mu(x)$ and $\sigma(x)$ returns the mean value and the standard deviation of the values x, respectively. The resulting CDFs are plotted in Figure 3 as red curves and we can obtain a good match.

$$R_i = \text{DET}(d) + \text{LOGN}(m, s) = \min(\widetilde{R}_i) + \text{LOGN}\left(\mu(\widetilde{R}_i - \min(\widetilde{R}_i)), \sigma(\widetilde{R}_i)\right) \qquad (1)$$

Furthermore, we found out that two thirds of the public eDonkey server could be pinged, whereas one third did not answer the ICMP packets. The 46 not pingable servers are also modelled as the sum of a deterministic and a lognormal random variable according to (1). For the not pingeable servers we model the parameters d, s and m of the distributions in (1) as normal distributed random variables. The values are taken from the set of pingable servers $\mathcal{J} = \{j : \text{server } j \text{ is pingable}\}$.

$$\begin{aligned} & d \sim \text{N}\left(\mu(\min(\widetilde{R}_j)), \sigma(\min(\widetilde{R}_j))\right), \quad m \sim \text{N}\left(\mu(\mu(\widetilde{R}_j)), \sigma(\mu(\widetilde{R}_j))\right) - d, \\ & s \sim \text{N}\left(\mu(\chi(\widetilde{R}_j)), \sigma(\chi(\widetilde{R}_j))\right) \cdot m \quad \text{with} \quad \chi(\widetilde{R}_j) = \sigma(\widetilde{R}_j)/\mu(\widetilde{R}_j) \end{aligned} \qquad (2)$$

4 Model of the Network and the Crawling Peer

We consider a mobile P2P-network as proposed in [1] and as introduced in Section 2. In the mobile network, the users generate a Poisson arrival process of requests for files which cannot be found in the mobile domain. Therefore the requests are delegated to the crawling peer (CP). The request arrival rate is denoted with λ. The CP then asks for the file at the known index servers in $\mathcal{I}$ according to a specific request strategy. The asked index server has three answer options: The file is known and the location of the file is reported back to the CP, the file is not known, or the CP is banned. In order to increase the efficiency of the search, the CP may ask a number of k servers simultanously. The search stops if either at least one request was successful, since we assume that additional sources – if available – are found by eDonkey's source exchange mechanism, or, if no source has been found according to the request strategy. The file request success probability on an individual index server $i \in \mathcal{I}$ is modelled by the probability f_i, which is derived from the measurements we describe in Section 3. It is defined as

$$f_i = \frac{\mu(\tilde{F}_i)}{\sum_{i \in \mathcal{I}} \tilde{F}_i}, \tag{3}$$

i.e. according to the distribution of the file registrations at the index servers.

The banning of clients which request an index server to often has been introduced lately by the creators of the ”lugdunum index” server, which is the software platform of choice for the majority of the index servers in the public eDonkey network. The index server has for each requesting client a number of credit points. For each file request, the credit is decreased by normally 16 points, while in turn in each second one point is added. That means that a client which requests the server every 16 seconds or higher will not be banned. If a client is banned and asks again within a server-specific time, the ban time is prolonged. A more detailed description of the banning mechanism can be found on the web [13].

The banning mechanism is modelled as following. Each index server has c_i credit points. Initially, the credits are set to a value of c_{init}, which is around 1000 credits according to the references we found on the web. On each request at i, the credits are reduced by c_r points. Once the crawling peer is banned at an index server, it stays banned forever in our model. This is a worst case assumption since we have no information about the ban time as it is implemented in the public eDonkey network.

The return time from the begin of the request for an index server until the report of the results is modelled with the measured round trip times as introduced in Section 3. The access time to the file location database in the server is neglected.

The goal is now to identify request strategies which deliver good results in terms of the file request success probability p_s and the mean search time μ_s. We define the success probability $p_{s,i}$ as the probability that after i requested servers a file location has been reported back to the crawling peer successfully. A key factor for the performance of the request strategies is the the ban probability p_{ban}, since with a ban on an index server this server is ”lost” for the rest of the time. Next, we introduce three kind of request strategies and discuss the success probability of the strategies to locate content.

4.1 RaRe Strategy - Randomly Requesting Servers

The random request (RaRe) strategy constitutes the most straightforward approach. The crawling peer chooses a set of k index servers randomly from the list and sends a file request to each of them. The search stops as soon as at least one file location has been reported back to the crawling peer. The success probability $p_{s,i}$ for this approach corresponds to the one-shifted geometrical distribution:

$$p_{s,i} = \text{GEOM}_1(\mu(f_i), i), \quad \text{with} \quad \mu(f_i) = \frac{\sum_{i' \in \mathcal{I}} f_{i'}}{|\mathcal{I}|} \tag{4}$$

Note that equation (4) is valid only if we neglect banning.

4.2 Psi Strategy - Optimizing Success Probability $p_{s,i}$

The Psi strategy tries to optimize the success probability $p_{s,i}$ by ordering the list of index servers by their individual file request success probability f_i. If performing a file search, the crawling peer asks first the index server with the highest f_i, i.e. with the highest number of files, then the second highest and so on. Although it can be expected that with this strategy the success probability is high, it can also lead to problems due to a fast banning at the highest index servers in the list. This assumption is also verified in the results, cf. Section 5.1.

The pure success probability $p_{s,i}$ without banning is now given by

$$p_{s,i} = f_i \prod_{j=1}^{i-1} (1 - f_j)\,, \tag{5}$$

so it exceeds the RaRe strategy in this discipline, since the servers with the highest success probabilities are asked first. Figure 4 shows the pure success probability $p_{s,i}$ for the RaRe and the Psi strategy after contacting i index servers. The analytically derived equations for $p_{s,i}$ are also validated by comparing with simulations. The Psi strategy has a much more larger success probablity than the RaRe strategy after contacting the same number i of index servers. With respect to be banned at an index server, the influence of the request strategy on the performance of the crawling peer is depicted more detailed in Section 5.1.

4.3 NoBan Strategy - Smart Requesting without Banning

The NoBan strategy tries to combine the advantages of the first two strategies, which are a low banning probability of the RaRe strategy and a high success probability and a small response time for the Psi strategy. However, the price for the better performance is the higher complexity of the strategy in terms of computation and memory requirements. As the name suggests, the NoBan strategy tries to avoid banning at any costs. This is achieved by assuming that the crawling peer knows it's credit points at all known index servers. So, the crawling peer can avoid a ban if an index server i where it has low credits is put on a black list. In this case, the search request is blocked at server i. The probability is $p_{b,i}$. If a search request has not been successfully answered yet and all not requested servers are on the black list, the search request is completely blocked. The probability is p_b. Index servers on the black list are taboo for file requests until the

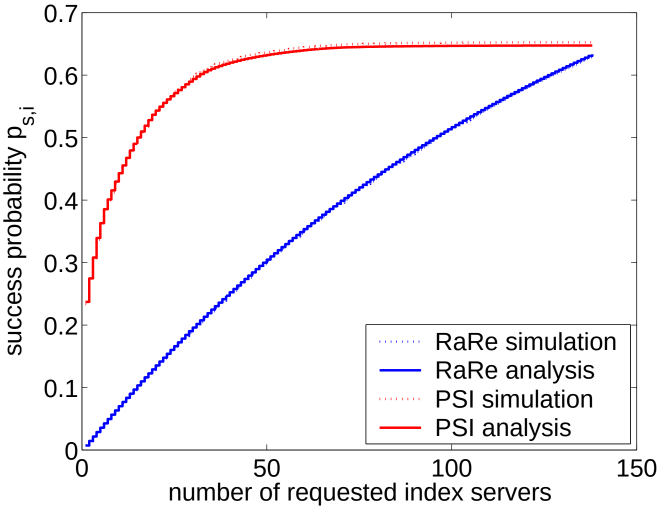


Figure 4: Success probability $p_{s,i}$ after contacting i servers

credit points are high enough to avoid banning. This strategy implies that some kind of signaling between the crawling peer and the index servers exists, such that the credit points are known.

The file request order is analog to the Psi strategy, i.e. we ask servers with the highest file count first. Consequently, the success probability corresponds to the success probability for the Psi strategy, too.

5 Results

In this section, we investigate the proposed request strategies with respect to success probability and response time. The strategies are compared for different load scenarios. The request arrival rate λ is defined as the number of search requests within one hour. Furthermore, we analyze the influence of the number of simultaneously requested servers on the performance of the crawling peer. Finally, we take a look into different criteria for blocking a search request according to the NoBan strategy.

5.1 Comparison of the Request Strategies

A search request for a file can be either blocked or not. If the request is not blocked and the crawling peer finds a source for the file, the search request is successfully answered. Otherwise, the search is unsuccessful. In this section, we consider $k = 1$ simultaneously requested servers. Figure 5 shows the cumulative distribution function of the response time of the crawling peer to search requests. We consider a scenario with very low search request arrival rate $\lambda = 25h^{-1}$. In this scenario, the obtained blocking probability p_b is zero when using the NoBan strategy. This means for unsuccessfully answered search requests that each of the N index server is contacted. The resulting response time T_{unsuc} is lognormally distributed, since the sum of lognormal ran-

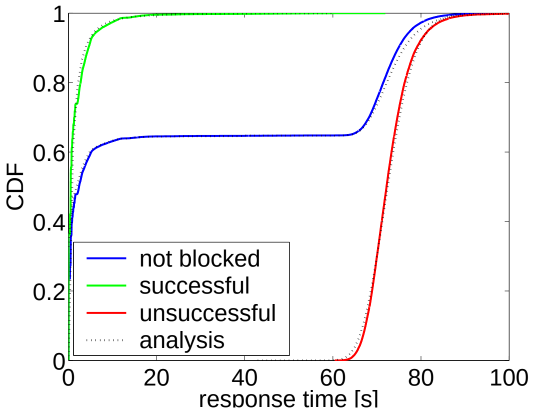


Figure 5: CDF of the response time of the crawling peer to search requests

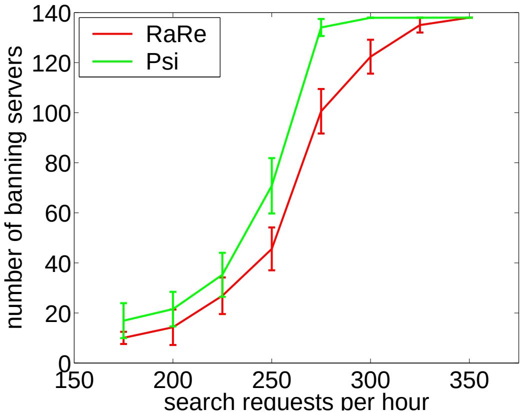


Figure 6: Number of servers from which the crawling peer is banned

dom variables can be effectively approximated as a lognormal random variable [15].

$$T_{unsuc} = \sum_{i=1}^{N} \min(R_i) + \text{LOGN}(\mu_{unsuc}, \sigma_{unsuc}) \text{ , with parameters} \tag{6}$$

$$\mu_{unsuc} = \sum_{i=1}^{N} (\mu(R_i) - \min(R_i)) \text{ , } \sigma_{unsuc} = \sqrt{\sum_{i=1}^{N} \sigma(R_i)^2} \tag{7}$$

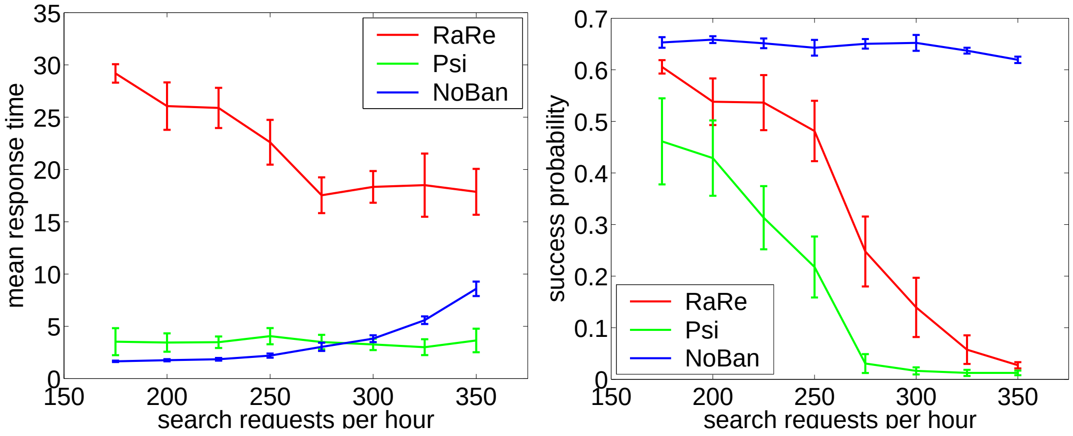
The density function $r_{suc}(t)$ of the response time T_{suc} for successful requests is computed by using the theorem of total probabilities. The index server i knows the searched file with probability f_i and the density function of its round trip time R_i is $r_i(t) = \frac{d}{dt} P\{R_i \leq t\}$

$$r_{suc}(t) = \left(\sum_{i=1}^{N-1} p_{s,i} \cdot r(t|i) \right) + r(t|N) \text{ with } p_{s,i} = f_i \prod_{j=1}^{i-1} (1 - f_i) \text{ and } r(t|i) = \bigcircledast_{j=1}^{i} r_j(t) \tag{8}$$

where $\circledast$ is the convolution of the density functions r_j. The probability is $p_{s,i}$ that index server i returns the first successful answer. For this case, we denote the resulting response time as $r(t|i)$. Analogously, the density function $r(t)$ of not blocked requests is computed by using the probability $p_{s,N} = f_N \prod_{i=1}^{N-1} (1 - f_i)$ of being successful after contacting all N servers.

$$r(t) = p_{s,N} \cdot r_{suc}(t) + (1 - p_{s,N}) \cdot r_{suc}(t) \circledast \frac{d}{dt} P\{T_{unsuc} \leq t\} \tag{9}$$

The derived response time in (9) is only valid for unblocked search requests and for not being banned from any index server. This assumption does not hold for the RaRe strategy and the Psi strategy. In this case, the crawling peer is already banned from index servers at a load where the blocking probability for the NoBan strategy is zero (i.e. $\lambda < 225\ h^{-1}$, cf. Figure 8). Figure 6



(a) Mean response time of successful requests

(b) Success probability of search requests

Figure 7: Influence of the request strategy on the performance of the crawling peer

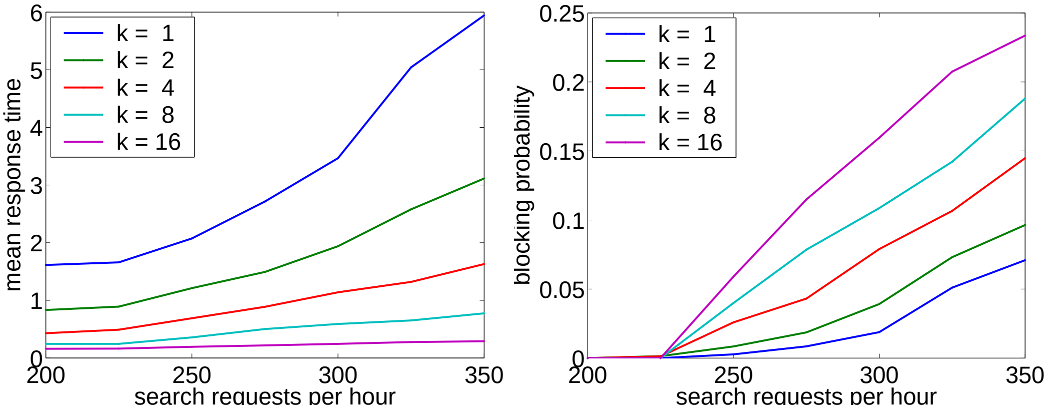
shows the mean number of banning servers and the corresponding confidence intervals for the RaRe and the Psi strategy when simulating 1,000 seach requests. For high search request arrival rates $\lambda > 300\ h^{-1}$, the crawling peer is almost banned from every index server. This results in a probability for a successfully answered request close to zero, cf. Figure 7(b).

We consider now the mean response time of the crawling peer for the different request strategies in dependence of the load. It seems astonishing that the response time of successfully answered requests increases for higher load when applying the NoBan strategy, while the response time remains on the same level or even decreases for the Psi and the RaRe strategy, respectively. This is illustrated in Figure 7(a). The reason is quite obvious, the number of blocked servers increases with the load for the NoBan strategy. On the other hand, the higher the number of banning servers is the less servers can be contacted which results in higher response time and lower success probabilites for the RaRe and Psi strategy, see Figure 7.

Considering the NoBan strategy, very small confidence intervals are obtained. For example, we simulated a scenario with the NoBan strategy for three different request rates $\lambda \in \{175, 225, 275\}h^{-1}$. For each scenario, 10,000 search requests were created and the number of repetitions is set to 20. The maximal observed relative error is only 0.49%. The relative error is defined as half-width of the confidence interval with a level of significance of $\gamma = 99\%$ normalized by the mean value. For this reason, the confidence intervals are no more plotted in the following sections where we only consider the NoBan strategy.

5.2 The Impact of Parallel Requests

In Section 5.1, we compared the request strategies if only one server at the time is asked for files. Now we examine the impact of the number of servers which are contacted in parallel, so $k > 1$. The motivation to increase the number of parallel requests is to reduce the mean file request response time. We consider the NoBan strategy only, since it turned out as the superior



(a) Mean response time of successful requests

(b) Blocking probability

Figure 8: Influence of k simultaneously requested servers according to NoBan strategy

request strategy in terms of response time and success probability, cf. Section 5.1.

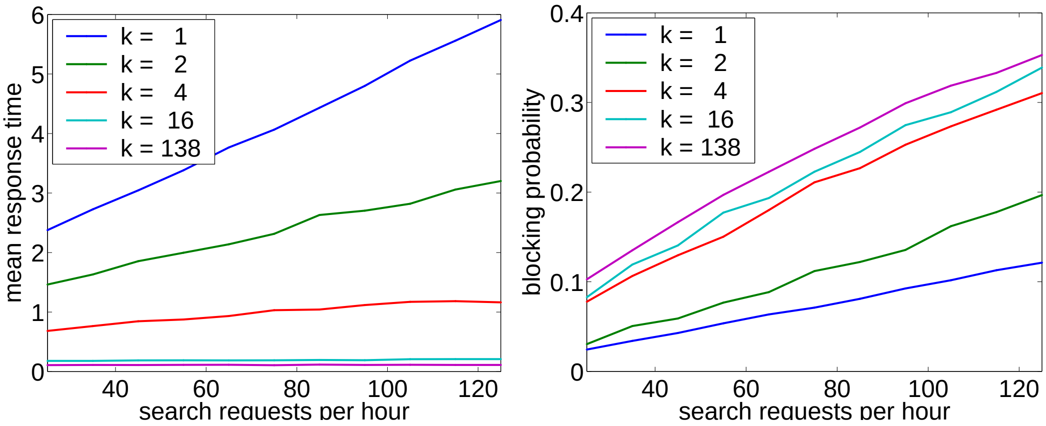
In Figure 8(a), the mean response time of the successsful searches over the mean search requests per hour for different numbers of k is shown. The mean response time is nearly halved if an additional parallel request process is added, such that with 8 parallel search processes the mean response time does not exceed the one-second mark, even for in high load situations.

The price for this gain is payed with increasing blocking probabilities. This can be seen in Figure 8(b). The blocking probabilities begin to grow as soon as the load exceeds the 225 request per hour mark. This corresponds to an mean interarrival time of $16s$, which is also the penalty in credits a request on an index server costs. The blocking probabilities also correlate to k, since with an higher number of parallel server requests the load increases too. So with $k = 8$, the users experience a blocking probability of ca. 20% in the case of 350 file requests per hour. This suggests that the number of index servers should be increased if possible.

5.3 Blocking Criteria for Search Requests

From Section 5.1 we have seen that it is most important to avoid being banned from any server. Otherwise, the success probability tends toward zero. In real eDonkey systems, index servers do not signal the amount of credit points to the requesting peer, cf. Section 4. In order to assure not being banned from any index server, we use a more stringent strategy for deciding when to block a search request. The required credit points c_i of the crawling peer for not being banned at index server i must exceed a value of $c_r = 16$. Since the credit points are increased for each second, we wait at least 16 seconds before contacting an index server again. This guarantees to be not banned from an index server. This modified NoBan strategy blocks requests by time and is denoted as NoBan-by-time strategy.

Figure 9 shows the blocking probability and the mean response times of successfully answered requests. In difference to the original NoBan-by-credits strategy (cf. Figure 8(b)), NoBan-by-time leads to much more blocked requests, even for pretty low loads. The reason is that the



(a) Mean response time of successful requests

(b) Blocking probability

Figure 9: Influence of the blocking criteria for requests according to the NoBan-by-time strategy

NoBan-by-time strategy does not cumulate periods of time during which no requests are issued. However, the credit points are increased during this period. This results in a kind of buffer which may accept search requests, even if they occur within 16 seconds. Figure 10 illustrates the influence of the blocking criterion on the number of blocked requests.

For the NoBan-by-time strategy, the probability $p_{b,i}$ that a search request to the index server i has to be blocked depends on the rate λ_i of request arrivals at server i. The corresponding interarrival time A_i is a random variable. It has to be noted that A_i depends on the former index servers $\{j \in \mathcal{I} : j < i\}$ because of blocked servers and successful responses. Regarding the first index server, $A_1 = A \sim \text{NEGEXP}(1/\lambda)$. For each index server i, we approximate the system by a $M/D/1$ queue. The offered load at server i is then $a_i = \frac{\lambda_i}{c_r^{-1}} = \lambda_i c_r$. Considering the blocking probability at server i it holds

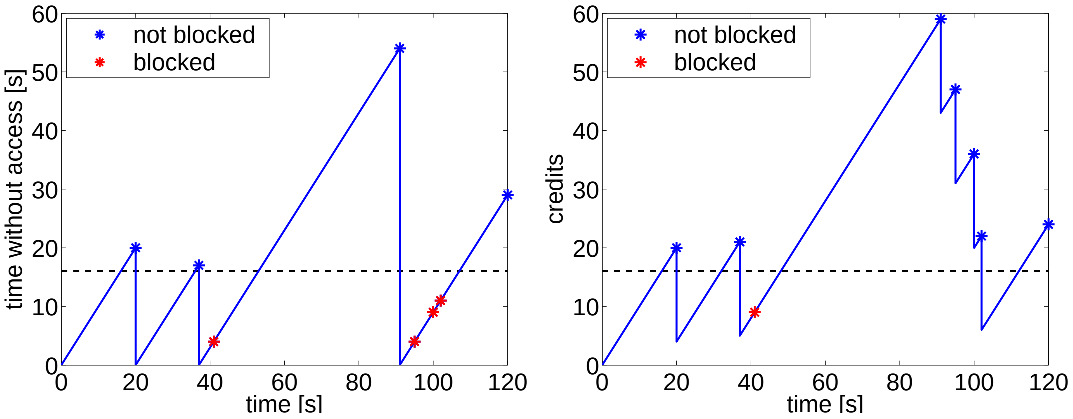
$$p_{b,i} = \frac{\lambda_i c_r}{1 + \lambda_i c_r} \text{ for the NoBan-by-time strategy.} \tag{10}$$

For the NoBan-by-credits strategy, the probability $p_{b,i}$ depends additionally on the credit points c_i at the index server i. The random variable c_i is time-discrete and reflects the number of credits at the end of each second, immediately after increasing by one.

$$p_{b,i} = P\{A_i < 1 | c_i \leq c_r\} \text{ for the NoBan-by-credits strategy.} \tag{11}$$

The distribution of the number of credit points can be calculated by using the power method. Then, the probability $p_{b,total}$ that a search request is totally blocked, i.e. at all index servers, can be formulated independent of the blocking criterion as follows:

$$p_{b,total} = \prod_{i=1}^{N} p_{b,i} \tag{12}$$



(a) blocking by time

(b) blocking by credits

Figure 10: Illustration of the blocking criteria for requests according to the NoBan strategy

However, the derivation of the blocking probability p_b as defined in Section 4.3 is more complex: If a search request has not been successfully answered yet and all not requested servers do not have enough credit points, the search request is blocked. The resulting blocking probability is denoted with p_b. For each file request x a list $\mathcal{L}_x$ of all index servers exists which denotes if server $y \in \mathcal{L}_x$ was already requested for request x:

$$\mathcal{L}_x(y) = \begin{cases} 0, & \text{if server } y \text{ not yet requested} \\ 1, & \text{if server } y \text{ already requested} \end{cases} .$$

A request x is blocked if no more server $y \in S = \{y : L(y) = 0\}$ can be contacted, i.e. credits $c_y < 16$. In the case of a blocked request it is $S \neq \emptyset$. Otherwise, $S = \emptyset$, each server was contacted, i.e. the search request was successfully or unsucessfully answered, but not blocked. For the NoBan strategy a file request x is always forwarded to the next available, not yet requested index server. This means that the next server i to be contacted for file request x is $i = \min\{j \in S : c_j \geq 16\}$ which has enough credit points, $c_i \geq 16$.

FTP servers, e.g. WinFTP, have an option called *AntiHammer* in order to ban users if they contact the server too often within a fixed time period. In this case, a crawling peer needs to use the NoBan-by-time strategy. On the other side, eDonkey index servers use a credit point system for avoiding hammering. This is much more comfortable for the users because of less blocked requests and shorter response time, cf. Figure 8 and Figure 9. For comparing both results, it has to be noted that the considered load for NoBan-by-credits is much higher than for NoBan-by-time. Nevertheless, the performance is even better in this case. For that reason, index servers or FTP servers should use a credit point system and signal the requesting users the amount of available credits. This results in a much more user-friendly, but equally effective prevention of hammering.

6 Conclusion and Outlook

The objective of this work was to investigate the crawling peer component, which optimizes the resource mediation mechanism in a mobile p2p architecure. The assumption was that too frequent requests to the index servers could lead to banning and to performance degradations.

To overcome this problem, three file request strategies have been proposed: Randomly Requesting Servers (”RaRe”), Optimizing Success Probability (”Psi”) and Smart Requesting without Banning (”NoBan”). The first two strategies do not explicitly avoid banning at the index servers, which leads to a trade-of between request success probability and response time. The performance of both strategies begins to decline with increasing load due to the destructive impact of banning at the index servers. Consequently, the NoBan strategy tries to avoid banning at all costs. We showed that this approach delivers a good performance even for high loads. This is traded for the cost of some additional intelligence in the crawling peer component.

Further investigation of the NoBan strategy showed, that a signalling of the credit points at the index servers would reduce the blocking probability at the crawling peer component (”NoBan-by-credits”). Without such a signalling, the crawling peer has to estimate the current number of credit points which leads in particular to increased blocking probabilities and therefore to decreased file request success probabilities (”NoBan-by-time”). So, an implementation of the NoBan-by-credits strategy meets the interests of mobile network operators which want to offer optimized, reliable and stable P2P services to their customers while maintaining the original P2P service experience.

Acknowledgement

The authors would like to thank Dirk Staehle for fruitful discussions during the course of this work.

References

[1] J. Oberender, F.-U. Andersen, H. de Meer, I. Dedinski, T. Hoßfeld, C. Kappler, A. Mäder, and K. Tutschku, “Enabling mobile peer-to-peer networking,” in *Mobile and Wireless Systems, LNCS 3427*, (Dagstuhl, Germany), 1 2005.

[2] Bundesministerium für Wirtschaft und Arbeit (Edt.), “Monitoring Informaitonswirtschaft - 7. Faktenbericht 2004.” available at `http://www.bmwi.de/`.

[3] N. Azzouna and F. Guillemin, “Experimental analysis of the impact of peer-to-peer application on traffic in commercial IP networks,” *European Transactions on Telecommunications*, vol. 15, no. 6, 2004.

[4] J. E. Gabeiras, “Panel Presentation on “Issues in Peer-to-Peer Networking” at COST279 Mid-Seminar, University of Roma “La Sapienza”, Jan. 21-22 2004, Italy..”

[5] F. Patalong, “UMTS Tagebuch: Was war, was soll’s?.” `http://www.spiegel.de/netzwelt/technologie/`.

[6] K. Tutschku and H. deMeer, “A measurement study on signaling on gnutella overlay networks,” in *Fachtagung - Kommunikation in Verteilten Systemen (KiVS) 2003*, (Leipzig, Germany), pp. 295–306, Feb. 2003.

[7] K. Tutschku, “A Measurement-based Traffic Profile of the eDonkey Filesharing Service,” in *5th Passive and Active Measurement Workshop (PAM2004)*, (Antibes Juan-les-Pins, France), Apr. 2004.

[8] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Looking up data in p2p systems,” *Communications of the ACM*, vol. 43, Feb. 2003.

[9] “The Gnutella Protocol Specification v0.4.” http://dss.clip2.com, 2001.

[10] “Meta Search Inc. eDonkey2000.” `http://www.eDonkey2000.com`, 2003.

[11] “eMule Project Team.” `http://www.emule-project.net`.

[12] T. Hoßfeld, K. Tutschku, F.-U. Andersen, H. de Meer, and J. Oberender, “Simulative performance evaluation of a mobile peer-to-peer file-sharing system,” in *NGI2005*, (Rome, Italy), 4 2005.

[13] Newsgroup: alt.pl.edonkey2000, “Explanation on blacklisting by servers.” `http://groups.google.de/groups?selm=79enjv06ablmsk5rmovd7nrckrhiiorj1s$%$404ax.com$\&$output=gplain`.

[14] “eDonkey Network Server List.” `http://ocbmaurice.dyndns.org/pl/slist.pl`.

[15] F. Fenton, “The sum of lognormal probability distributions in scatter transmission systems,” *IRE Trans. Commun. Syst.*, vol. CS-8, no. 3, pp. 57–67, 1960.