University of Würzburg
Institute of Computer Science
Research Report Series

Time-Discrete Analysis of the Crawling Strategy in an Optimized Mobile P2P Architecture

Tobias Hoßfeld, Andreas Mäder, Kurt Tutschku[1] and
Frank-Uwe Andersen[2]

Report No. 374 December 2005

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

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

Time-Discrete Analysis of the Crawling Strategy in an Optimized Mobile P2P Architecture

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

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

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. In [1], we have investigated the performance of a crawling peer by means of simulations. Now, we show a time-discrete analysis of the crawling peer's performance in order to investigate different scenarios and to enable parameter-sensitivity studies for further improvements of the crawling peer's strategy.
Keywords: crawling peer, mobile P2P architecture, file-sharing, queueing theory

1 Introduction

Currently, UMTS network operators are looking for applications which *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 accomplished for example by entities which on behalf of others store content, i.e. *caches*, 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 [2, 3]. 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. Furthermore, the P2P user does not want to pay for a large amount of mediation traffic on the air interface. The reduced mediation traffic, the discontinued signalling, and the short mediation times needed for mobile P2P file sharing networks ask for new architecture solutions for these kinds of services.

An efficient solution might state the use of new entities, in particular of the so-called *crawling peer*. Our architecture concept is presented in [4] and additionally comprises a cache peer and a modified index server. The crawling peer (CP) is placed in the wired part of the mobile network and locates files on behalf of 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.

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 [5], or *flooding concepts*, as used in Gnutella.

In [1], we have investigated the performance of a crawling peer by means of simulations. Now, we present an analytical performance evaluation based on time-discrete analysis in order to investigate different scenarios and to enable parameter-sensitivity studies for further improvements of the strategy of the crawling peer.

This paper is organized as follows. Section 2 describes the mobile P2P architecture. In Section 3 we discuss at which index servers mobile specific contents may be located. The considered network and the crawling peer are modeled in Section 4. The analytical approach is explained in Section 5. Some numerical results are given in Section 6 and Section 7 concludes this work.

2 Mobile P2P Architecture

The suggested mobile P2P architecture for third generation mobile networks first introduced in [4] is depicted in Figure 1. The suggested concept is based on the architecture of the popular eDonkey P2P file sharing application and was enhanced by three specific entities: the *cache peer*, the *mobile P2P index server*, and the *crawling peer*.

The *cache peer* is a modified eDonkey peer located in the wireline part of the mobile P2P architecture 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 [6]. The *mobile P2P index server* is a modified eDonkey index server. It tracks the frequently requested content, triggers the cache peer to fetch it, and forces the mobile peers to download the file from the cache peer, if available.

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

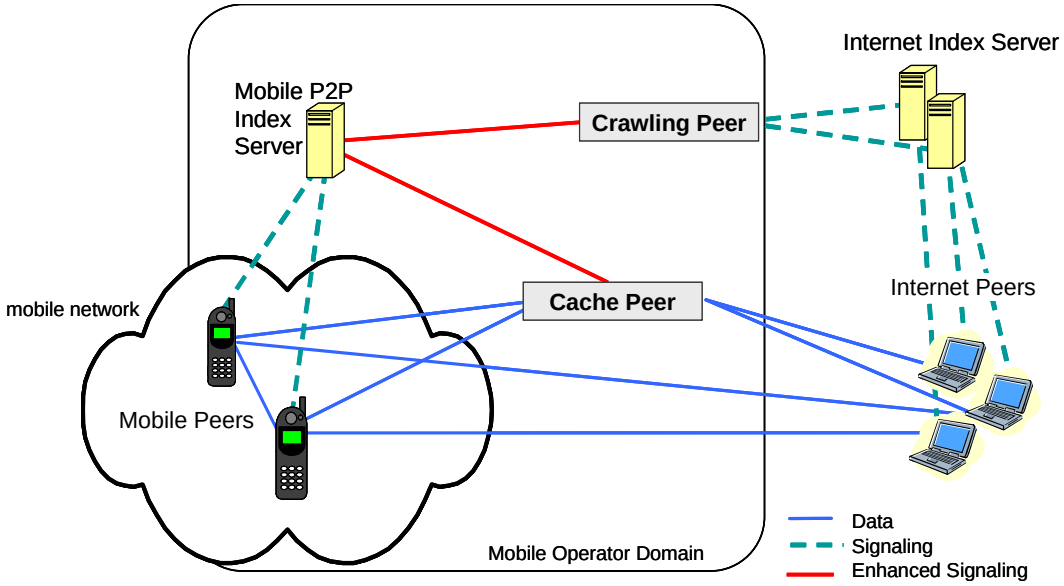


Figure 1: Architecture concept for a P2P file-sharing service optimized to mobile networks

mobile peer should not be allowed to contact external eDonkey servers. If a mobile peer would contact external index servers 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 sent 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 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 [7], which prevents a peer of issuing too many search queries to an index server. The crawling peer should query only index servers for which it has enough credit points.

3 Content Location in a Hybrid P2P File-Sharing Network

In a hybrid P2P network, index servers keep information on peers and respond to requests for that information, while the peers are responsible for hosting the information, since the index servers only operate as index database of the contents and do not store the files. In the proposed mobile P2P architecture, the crawling peer locates contents on behalf of the mobiles and sends search queries to the available index servers in the network. If an index server has not registered the file for which the crawling peer asked, the crawling peer sends the search query to the next index server.

The performance evaluation of the crawling strategy requires the file request success probability which models whether an index server has registered a file for which a query is sent or not. The success probability f_i on an individual index server i may be derived from the measurements in [1]. There, it is defined as

$$f_i = \frac{\mu(\tilde{F}_i)}{\sum_{i \in \mathcal{I}} \mu(\tilde{F}_i)}, \tag{1}$$

i.e. according to the distribution of the file registrations at the index servers. The measured number of registered files at index server i is denoted by $\tilde{F}_i$ and the mean number of registered files at server i by $\mu(\tilde{F}_i)$.

In this case the success probability f_i on an individual index server simply depends on the ratio of registered files at this server to the total number of available files in the network. However, in P2P file-sharing networks, like the eDonkey network, the creation of *user groups* can be seen at the different index servers. Users which have the same or similar interests are also connected to the same server. This allows short lookup times when searching for contents which can be classified to this area of interest. User groups may be communities which are interested for example in movies in French language or in the latest computer games for PSP.

The mobile P2P file sharing application is supported additionally by the mobile network operator. As a result a mobile subscriber using that service achieves the best performance if it connects to the operator's index server within the mobile P2P architecture. But this means that is very likely that the mobile P2P users will also create a user group at this index server, the *mobile P2P community*.

We assume that there are mobile specific content types like ring tones (midi files or mp3 files), digital images, small videos, or games, which are shared and of interest for the mobile P2P users. This means that the mobile users will search for and download mobile specific content, whereby most of the files will be registered at the operator's index server. Thus, the success probability to find a mobile specific content at another index server may be assumed to be equal for all other index servers. This results is the following file request success probability p_s at an arbitrary index server when the crawling peer searches on behalf of the mobiles at all index servers $\mathcal{I}$ in the network:

$$p_s = \sum_{i \in \mathcal{I}} \frac{f_i}{|\mathcal{I}|}. \tag{2}$$

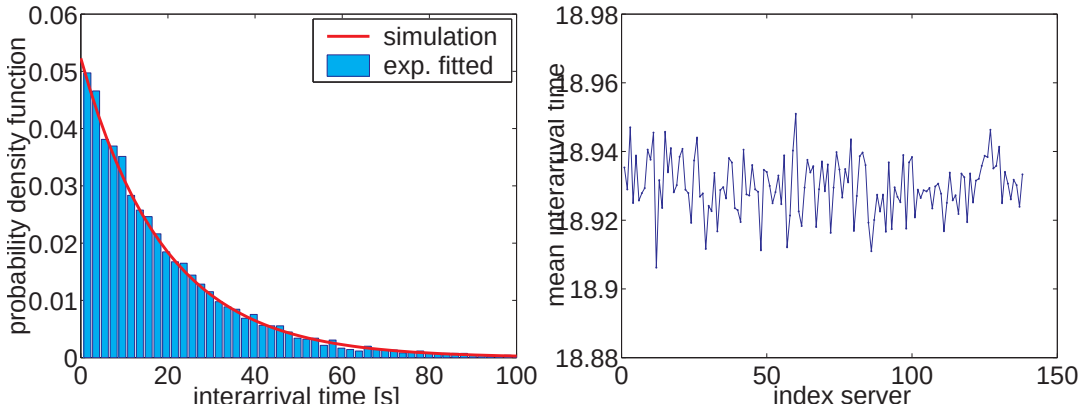
According to our measurements in [1] it holds $p_s = 0.7246\%$ and the number $|\mathcal{I}|$ of index servers was $|\mathcal{I}| = 138$. If the crawling peer asks every index server, the probability that the file is available in the network and registered at any index server, i.e. the probability to successfully

locate a file, is given by

$$1 - \prod_{i=1}^{|\mathcal{I}|} (1 - p_s)\,, \tag{3}$$

which is about 63.35% for the measured values p_s and $|\mathcal{I}|$.

In [1] we have implemented a simulation to evaluate the performance of the crawling peer and its strategy. In the mobile network the users generate a Poisson arrival process with rate λ of requests for files which cannot be found in the mobile domain. Now, the crawling peer comes into play and starts to query the index servers in the file-sharing network. As parameter for the success probability of the crawling peer at an arbitrary index server we used p_s as defined in (2) and performed some simulation runs.



(a) PDF fitted exponentially

(b) Mean $1/\lambda_i^*$ for all i

Figure 2: Observed interarrival times by simulations

Figure 2 shows the results of the simulation runs. We take a look on the observed search queries of an arbitrary index server. In Figure 2(a) the probability density function (PDF) of the observed file request interarrival time at a single, arbitrary index server is plotted. The simulated curve is fitted with the PDF of an exponential distribution and a very good match is obtained. Thus, the file request arrival process still follows a Poisson process, however with a different rate $\lambda_i^* < \lambda$. Due to the same success probability for all index servers, all index servers behave equal. This can also be noticed when comparing the mean interarrival times $\frac{1}{\lambda_i^*}$ of all index servers $i \in \mathcal{I}$ which show only slightly differences. This observation, that the file request queries sent to individual index server still follow a Poisson process and that all index servers experience the same file request rate, was the starting point of the analytical approach.

4 Network and Crawling Peer Model

We consider a mobile P2P-network as proposed in [4] 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

$$\mathcal{I} = \{1, \cdots, N\} \tag{4}$$

according to a specific request strategy, the NoBan-strategy [1]. The search stops if either at least one request was successful, since we assume that additional sources – if available – can be found by eDonkey's source exchange mechanism, or, if no source has been found.

The banning of clients 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. A more detailed description of the banning mechanism can be found on the web [7].

The banning mechanism is modelled as following. An index server i has for each requesting peer, i.e. also for the crawling peer, a number of credit points c_i. 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 Δc points, while in turn in each second one point is added. So, a client is banned from a server if the request would cause a negative amount of credit points. Once the crawling peer is banned at an index server, it stays banned forever. This is a worst case assumption since we have no information about the ban time as it is implemented in the public eDonkey network.

Our *NoBan strategy* [1] avoids banning and achieves a small response time and a high probability to locate a file which is close to the maximal value (3). 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:

$$\mathcal{L}_x(y) = \begin{cases} 0, & \text{if server } y \text{ not yet requested,} \\ 1, & \text{if server } y \text{ already requested.} \end{cases} \tag{5}$$

The set S_x of not yet requested index servers for request x is therefore

$$S_x = \{i \in \mathcal{I} : L_x(i) = 0\}. \tag{6}$$

If the crawling peer has low credits c_i at an index server i, the search request is blocked at server i. This probability is denoted as $p_{b,i}$. 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 $i = \min\{j \in S_x : c_j \geq \Delta c\}$ which has sufficient credit points, $c_i \geq \Delta c$.

A request x is blocked completely if no more server $y \in S_x$ can be contacted due to available credits:

$$\forall y \in S_x : c_y < \Delta c. \tag{7}$$

We denote this blocking probability with p_b. 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 answered successfully or unsuccessfully.

5 Analytical Approach

In this section, we investigate the NoBan strategy under the assumption that the file request success probabilities f_i on each index server are equal, as motivated in Section 3,

$$\forall i, j \in \mathcal{I} : f_i = f_j =_{def} p_s. \tag{8}$$

As a consequence, all servers are equal and are therefore asked randomly for a file request x. In particular, the next index server is in this case randomly chosen from the set S_x of remaining, not yet asked servers. Since the file request arrivals follow a Poisson process with rate λ, the observed arrivals at each individual server i still follow a Poisson process, demonstrated in Figure 2. We denote the obtained rate at an individual index server with λ_i^*. Because of this notice, we can describe the analysis model as depicted in Figure 3.

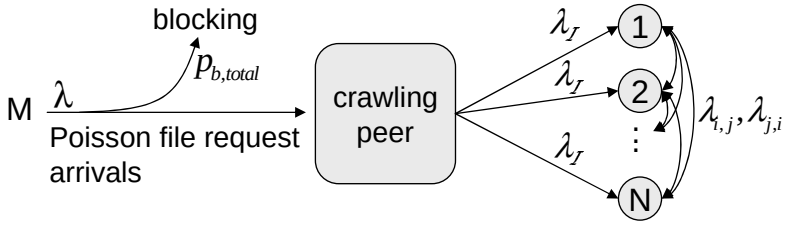


Figure 3: Illustration of the analysis model

The Poisson file requests arrivals are split equally among the N index servers:

$$\lambda_\mathcal{I} = \lambda N(1 - p_{p,i}). \tag{9}$$

If a search request is unsuccessfully answered at server i, the request is forwarded to a not yet requested server $j \in S$. The corresponding rate is

$$\lambda_{i,j} = \lambda_\mathcal{I}(1 - p_s) \tag{10}$$

which holds for all $i \neq j$. The observed rate at an index server i follows as

$$\lambda_i^* = \lambda_\mathcal{I} + \sum_{j \neq i} \lambda_{j,i} = \lambda_\mathcal{I}(N(1 - p_s) + p_s). \tag{11}$$

The probability $p_{b,total}$ that a search request is totally blocked, i.e. at all index servers, is

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

However, the derivation of the probabilities p_b and $p_{b,i}$ is more complex due to interaction with the search query rate λ, the number of index servers N, and the number of credit points c_i at index server i. In the following we use a numerical approach to retrieve the blocking probabilities $p_{b,i}$ and the observed search query rate λ_i^* at an index server i. The distribution of the number c_i of credit points at each server i can be calculated by using time-discrete analysis. In order to

get $p_{b,i}$ an equation system is then numerically solved using again iteration. We start with the description of the compuation of the steady state distribution of the credit points c_i for a given rate λ_i^*.

Let $X = c_i$ be a random variable which desribes the number of credit points of the CP at an arbitrary index server i and T describes a point in time. The time is discretized in intervals of length $\Delta T = 1$second. Then, $P(X = j|T = n)$ denotes the state probability that the CP has j credit points at time $n\Delta T = n$ seconds. The state probabilities form the components of the vector

$$\mathbb{X}_n = \begin{pmatrix} P(X = 0|T = n) \\ P(X = 1|T = n) \\ \vdots \\ P(X = c_{max}|T = n) \end{pmatrix}. \tag{13}$$

The expression $\mathbb{X}_n(j)$ returns the j-th element of the vector $\mathbb{X}_n$, i.e.

$$\mathbb{X}_n(j) = P(X = j|T = n). \tag{14}$$

In this time-discrete analysis, we use the power method to compute numerically the distribution of the number of credit points. Therefore, the state space has to be finite. This condition is fulfilled for eDonkey index servers and we consider a maximum number c_{max} of credit points.

The start vector $\mathbb{X}_0$ is defined as follows and initializes the iterative computation of the state probabilities $\mathbb{X}_n$:

$$\mathbb{X}_0(j) = \begin{cases} 0 & , 0 \leq j < c_{max}, \\ 1 & , j = c_{max}. \end{cases} \tag{15}$$

The probability $\mathbb{P}_n(j,k)$ denotes the conditional probability that the amount of credit points is j at time $n\Delta T$ under the condition that the CP issued $K_n = k$ search queries within the last time interval ΔT:

$$\mathbb{P}_n(j,k) = P(X = j|K_n = k). \tag{16}$$

The random variable K_n denotes the number of search queries from $(n-1)$ seconds until n seconds. Since the arrivals of search queries at an index server follow a Poisson process with rate λ_i^* (search requests per time unit ΔT), the number of search queries is Poisson distributed:

$$P(K_n = k) = \frac{(\lambda_i^* \Delta T)^k}{k!} e^{-\lambda_i^* \Delta t}, \quad k = 0, 1, 2, \ldots. \tag{17}$$

The power method requires again a finite number of states in order to describe the conditional probabilities $\mathbb{P}_n(j,k)$. Thus, the α-quantile of the distribution of K_n is used to assume the maximal number k_{max} of search queries:

$$P(K_n \leq k_{max}) = \alpha. \tag{18}$$

The conditional probability $\mathbb{P}_{n+1}$ can now be computed iteratively. First, we consider the case that $K_n = 0$ search queries were issued by the CP to the index server during the last second. After each second, the amount of credit points is increased by one. Since no search query was

issued, at least one credit point is availabe, i.e., $\mathbb{P}_{n+1}(0,0) = 0$. In order to truncate again the state space, $\mathbb{P}_{n+1}(c_{max}, 0)$ sums up the remaining probabilites.

$$\mathbb{P}_{n+1}(j,0) = \begin{cases} 0 & ,j = 0, \\ \mathbb{X}_n(j-1) & ,0 < j < c_{max}, \\ \mathbb{X}_n(c_{max}-1) + \mathbb{X}_n(c_{max}) & ,j = c_{max}. \end{cases} \tag{19}$$

Next, we consider $K_n = k$ search queries for $k = 1, 2, \cdots, k_{max}$. A single search request costs Δc credit points. The probability $\mathbb{P}_{n+1}(j,k)$ that the transition to a number of credit points j that is larger than $k\Delta c$ is $\mathbb{X}_n(j + k\Delta c - 1)$. The transition to $j > c'_{max} = c_{max} - k\Delta c + 1$ is not possible as $k\Delta c$ credit points are consumed. To achieve less than $k\Delta c$ credit points either not enough credit points were available or the required credit points were assumed for the k queries. We obtain the following equation:

$$\mathbb{P}_{n+1}(j,k) = \begin{cases} \mathbb{X}_n(j-1) + \mathbb{X}_n(j + k\Delta c - 1) & ,0 \leq j < k\Delta c, \\ \mathbb{X}_n(j + k\Delta c - 1) & ,k\Delta c \leq j < c'_{max}, \\ \mathbb{X}_n(c_{max}-1) + \mathbb{X}_n(c_{max}) & ,j = c'_{max}, \\ 0 & ,j > c'_{max}. \end{cases} \tag{20}$$

In order to compute the state probability $\mathbb{X}_{n+1}(j)$, Bayes theorem is applied using (17), (19), and (20):

$$\mathbb{X}_{n+1}(j) = \sum_{k=0}^{k_{max}} \mathbb{P}_{n+1}(j,k) \cdot P(K_n = k). \tag{21}$$

The computation of $\mathbb{X}_{n+1}$ is now iterated until the steady state $\mathbb{X}$ is reached, i.e.

$$\mathbb{X}_n = \mathbb{X}_{n+1} = \mathbb{X}. \tag{22}$$

In practice, the condition whether the steady state is reached or not is realized by checking the absolute difference of the mean number of credit points for two consecutive iteration steps. If the difference is smaller than a given threshold ϵ, the terminating condition is fulfilled and the iteration is stopped:

$$|\mathrm{E}[\mathbb{X}_{n+1}] - \mathrm{E}[\mathbb{X}_n]| < \epsilon. \tag{23}$$

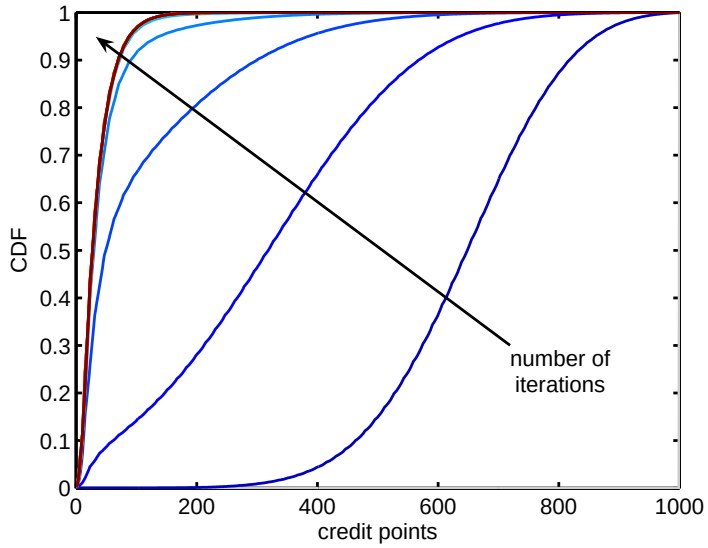


Figure 4: Number of iterations to get credit point distribution

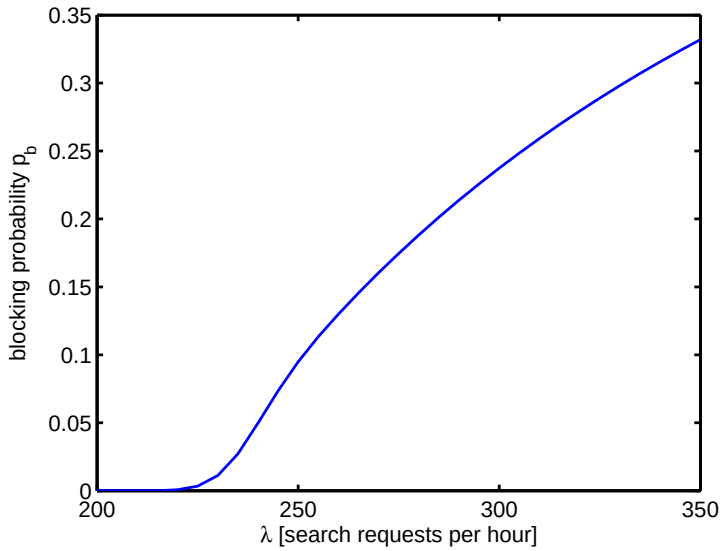


Figure 5: Blocking probability for a single server system, $N = 1$, for different rates λ

This numerical method is a very robust and efficient approach. Figure 4 shows the cumulative distribution function of the credit points for the different iteration steps $\mathbb{X}_n$. The arrow indicates the number of iterations which were executed. Only a few iterations are required until the terminating condition (23) is fulfilled.

Figure 5 shows the computed blocking probabilities for different search request arrival rates λ.

In this case, we only consider a single server, i.e. $N = 1$, which is contacted for each file query. It can be seen that for file request rates smaller than 350 requests/hour the blocking probability p_b is vanishing. But the blocking probability significantly increases for larger rates resulting in inacceptable blocking probabilities, e.g. for $\lambda = 300$ requests/hour the blocking probability is already $p_b = 0.23$. However, in real eDonkey networks, there exist several index servers, thus a server does not see all of the requests. In our measurements, we investigated $N = 139$ servers.

With the knowledge of the distribution of the number of credit points $X = c_i$ at an index server i for a given rate λ_i^*, the occuring blocking probability $p_{b,i}$ can be computed that a search request is blocked at the server due to not enough credit points. However, the rate λ_i^* depends on the blocking probabilities $p_{b,j}$ of the other index servers $j \neq i$. Since we already know from Section 3 that we may consider all N index servers to be equal with respect to blocking probabilities $p_{b,i}$ and observed search query rates λ_j^*, it holds the following equation system for the steady state:

$$\begin{aligned} p_{b,i} &= \sum_{j=0}^{\Delta c-1} \mathbb{X}(j) = \sum_{j=0}^{\Delta c-1} P(X=j), \\ \lambda_i^* &= \frac{\lambda}{N} + \sum_{y=1}^{N-1} \lambda \cdot P(B=1|Y=y) \cdot P(Y=y). \end{aligned} \tag{24}$$

Hereby, Y is a random variable which describes the number of already contacted index servers. B is a random variable that the considered index server i is chosen. This means that B follows a Bernoulli distribution.

The conditional probability $P(B = 1|Y = y)$ denotes the probability that the index server i is contacted after y other index servers were contacted. $P(Y = y)$ is the probability that all y servers have not successfully answered a search query or that these servers were blocked. This means

$$P(Y=y) = \binom{N}{y} \left((1-p_{b,i}) \cdot (1-p_s) + p_{b,i}\right)^y. \tag{25}$$

According to the NoBan strategy a server is not contacted twice for the same search query. Thus, if already y servers were contacted, the probability that index server i is chosen follows as

$$P(B=1|Y=y) = \frac{1}{N-y}. \tag{26}$$

Inserting (25) and (26) in (24) leads to

$$\begin{aligned} p_{b,i} &= \sum_{j=0}^{\Delta c-1} \mathbb{X}(j) = \sum_{j=0}^{\Delta c-1} P(X=j), \\ \lambda_i^* &= \frac{\lambda}{N} + \sum_{y=1}^{N-1} \binom{N}{y} \cdot \lambda \cdot \frac{1}{N-y} \cdot \left((1-p_{b,i}) \cdot (1-p_s) + p_{b,i}\right)^y. \end{aligned} \tag{27}$$

The equation system (27) can now be solved numerically by iterating again until the steady state is reached, i.e., until the blocking probability $p_{b,i}$ and the observerd search query rate λ_i^* at server

i only changes slightly by a threshold ϵ in succeeding iteration steps. The success probability p_s that an index server has registered the searched file is given as input parameter. The iteration is initialized with $p_{b,i} = 1$.

6 Numerical Results

In this section, we present some numerical results for different parameters. We vary over a large range of realizations for the parameters. This is made possible by the time-discrete analysis and can be very efficiently numerically computed outperforming more time-consuming simulations.

First we take a look on the observed interarrival time of search queries which are forwarded to an individual index server i by the crawling peer. The total search queries in the system being issued to the crawling peer is described with the file request arrival rate λ. Figure 6 shows on the x-axis the file request arrival rate λ and on the y-axis the observed mean interarrival times $\frac{1}{\lambda_i^*}$ at an arbitrary index server i. The higher the load in the system, i.e. the higher the file request rate, the higher is also the load for individual index servers, which is expressed by smaller mean query interarrival times $\frac{1}{\lambda_i^*}$. From the convex shape of the curve, it can be seen that the crawling peer is a very efficient solution to realize resource mediation in P2P file-sharing networks and that the CP distributes the load in the network among the different index servers. Thus, it is possible to accomplish flash crowd arrivals of search requests without loosing the quality of the service.

It can be expected that a higher rate λ also leads to higher blocking probabilities $p_{b,i}$ which is investigated next. Again, we vary over the total file request rate λ in the network which is given on the x-axis in Figure 7. The resulting blocking probability $p_{b,i}$ that the crawling peer cannot

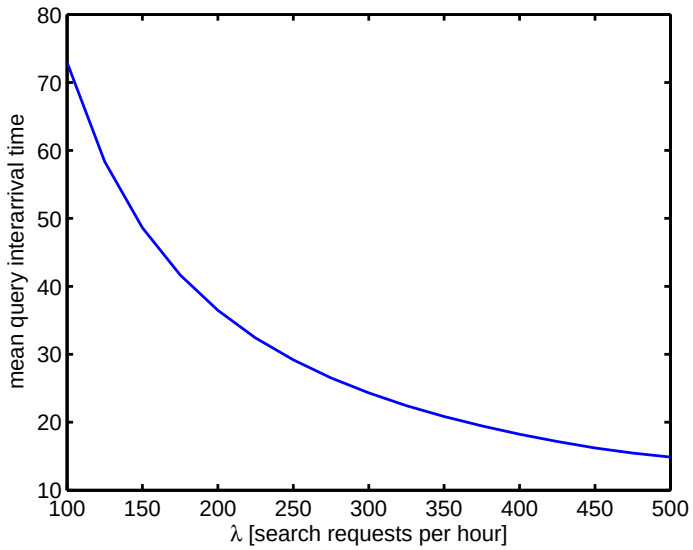


Figure 6: Observed mean interarrival times $\frac{1}{\lambda_i^*}$ at an arbitrary index server i

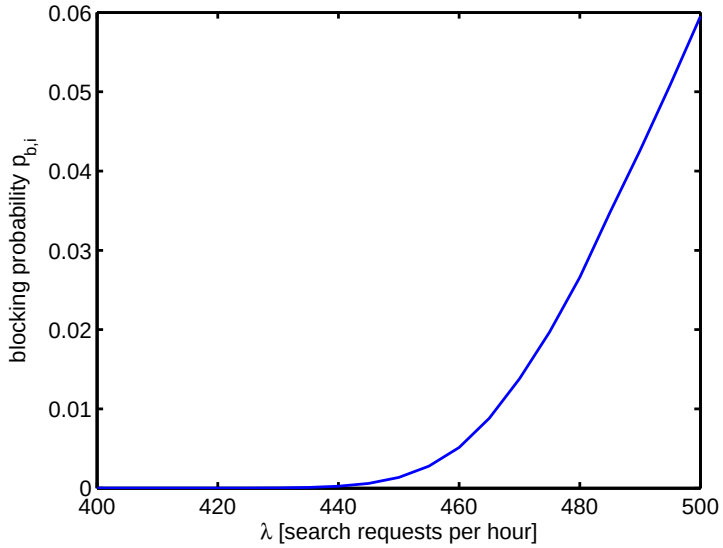


Figure 7: Blocking probability for different search arrival rates λ

forward a search request to an individual index server i is plotted versus λ. It can be seen that the crawling peer supports up to a mean number of 440 search requests per hour while achieving a blocking probability $p_{b,i}$ close to zero. If the total load in the system is permanently higher over time, e.g. about 500 requests per hour, the crawling peer has to block some search requests to the index server i due to not enough credit points and to avoid therefore to be banned at server i. Nevertheless, the total blocking probability $p_{b,total}$ is still for this rate close to zero, cf. Eq. (12).

The mobile network operator which supports the P2P file-sharing service can dimension the network in such a way that the experienced quality of the file-sharing service satisfies the user and the blocking probability falls below a given threshold. If the service provider operates k crawling peers in the mobile domain, the load can be distributed among the k CPs. This means that each of the k CPs only sees $\frac{1}{k}$ of the total load λ, i.e. each CP has then only to accomplish a file request rate $\frac{\lambda}{k}$. Now, the operator can choose k such that $p_{b,i}$ is vanishing. According to Figure 7 this means to find the minimal k such that $\frac{\lambda}{k} < 440$ requests per hour.

Another parameter that is of interest is the maximal number c_{max} of credit points which a peer can gather at an index server. c_{max} influences how strong a peer is rewarded if it does not contact the index server for longer periods of time. The maximal number of credit points help to accomplish bursts in the arrival of search requests. If the file request arrival process shows a higher variance, a smaller number of maximal credit points will lead to higher blocking probabilities.

Figure 8 shows the blocking probability $p_{b,i}$ in dependence of the maximal number c_{max} of available credit points. The blue curve indicates the numerical solution of the time-discrete analysis which explains the small zigzag of the solution curve due to numerical inaccuracies. We only have fitted the numerical solution polynomial for visualization purposes to obtain a smoother curve without zigzag. From Figure 8, it can be seen that the blocking probabilities $p_{b,i}$

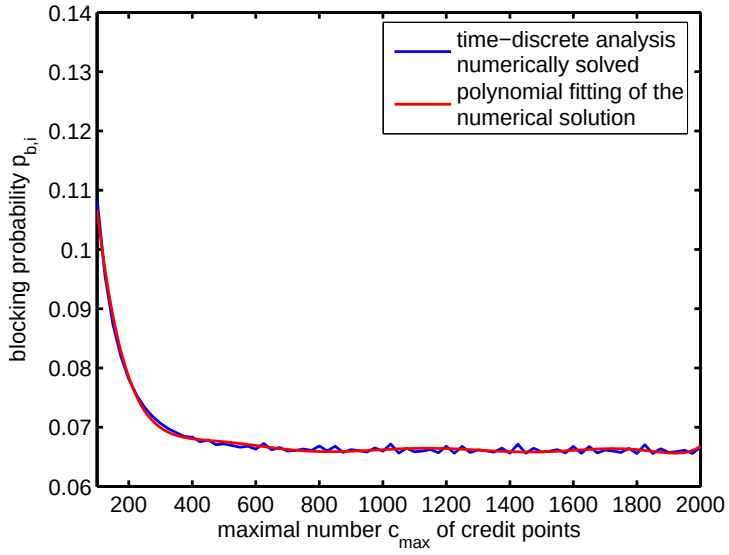


Figure 8: Blocking probability in dependence of the maximal number of available credit points

at individual index server stay constant if the maximal possible number of credit points exceeds a value of about 500 credit points. In that case, this results in a much more user-friendly (in terms of blocking probabilities), but still effective prevention of hammering the index server.

7 Conclusion and Outlook

Mobile networks differ from their wireline counterparts mainly by the high costs for air transmissions and by the mobility of the users. The crawling peer is suggested in order to optimize the resource mediation mechanism for a mobile P2P file-sharing application. The objective of this work was to investigate the crawling peer component, which optimizes the resource mediation mechanism in a mobile P2P architecture. We presented a time-discrete analysis for describing interactions between performance factors, like the observed arrival rate and the blocking probability. The computation of the performance factors was done using the power method and numerical iteration techniques. This approach enables parameter sensitivity studies and might lead to optimal value for tradeoff parameters. It helps to dimension the mobile P2P network in such a way that the experienced quality of the file-sharing service, e.g. in terms of successfully answered search requests, satisfies the user and that the blocking probability is below a given threshold.

In particular, we investigated the observed search queries at an arbitrary server and the resulting blocking probabilities for different arrival rates of search queries. As a result of the analysis, we found out that the crawling peer is a very efficient solution to realize resource mediation in P2P file-sharing networks and that the CP distributes the load in the network among the different index servers. Thus, it is possible to accomplish flash crowd arrivals of search requests without loosing the quality of the service. Furthermore, the analysis makes the dimensioning of the mo-

bile P2P file-sharing architecture possible. The mobile network operator which supports the P2P file-sharing service can dimension the network in such a way that the experienced quality of the file-sharing service satisfies the user and the blocking probability falls below a given threshold. Next we investigated the influence of the maximal number of credit points on the blocking probabilities. The time-discrete analysis shows that the blocking probabilities at an individual index server stay constant if the maximal possible number of credit points exceeds a certain value. In that case, a much more user-friendly (in terms of blocking probabilities), but still effective prevention of hammering the index servers is realized.

Acknowledgement

The authors would like to thank Dirk Staehle and Marie-Ange Remiche for the fruitful discussions during the course of this work.

References

[1] T. Hoßfeld, A. Mäder, K. Tutschku, P. Tran-Gia, F.-U. Andersen, H. de Meer, and I. Dedinski, "Comparison of crawling strategies for an optimized mobile p2p architecture," in *19th International Teletraffic Congress (ITC19)*, (Beijing, China), 9 2005.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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