Enabling Mobile Peer-to-Peer Networking

Jens O. Oberender[1], Frank-Uwe Andersen[3], Hermann de Meer[1], Ivan Dedinski[1], Tobias Hoßfeld[2], Cornelia Kappler[3], Andreas Mäder[2], Kurt Tutschku[2]

[1] University of Passau, Chair of Computer Networks and Computer Communications. Innstraße 33, 94032 Passau, Germany.
[oberender|demeer|dedinski]@fmi.uni-passau.de
[2] University of Würzburg, Department of Distributed Systems. Am Hubland, 97074 Würzburg, Germany.
[maeder|hossfeld|tutschku]@informatik.uni-wuerzburg.de
[3] SIEMENS AG. Siemensdamm 62, 13623 Berlin, Germany.
[frank-uwe.andersen|cornelia.kappler]@siemens.com

Abstract. In this paper we present a P2P file-sharing architecture optimized for mobile networks. We discuss the applicability of current P2P techniques for resource access and mediation in the context of 2.5G/3G mobile networks. We investigate a mobile P2P architecture that is able to reconcile the decentralized operation of P2P file sharing with the interests of network operators, e. g. control and performance. The architecture is based on the popular eDonkey protocol and is enhanced by additional caching entities and a crawler.

1 Introduction

P2P file sharing systems account for a high percentage of the traffic volume in the fixed Internet, having exceeded http (WWW) or email traffic [1] [2]. The increasing availability of mobile data networks such as GPRS and UMTS in conjunction with attractive pricing schemes makes P2P file sharing an interesting application also in the mobile context. But the operation of P2P systems in mobile environments encounters several problems, such as a relatively narrow and expensive air interface, highly varying online states (presence) of the subscribers, a hierarchical network structure (GPRS), and limited device capabilities.

P2P is a distributed application architecture where equal entities, denoted as peers, voluntarily share resources, e.g. files or CPU cycles, via direct, end-to-end exchanges. In order to share resources, P2P applications need to support two fundamental coordination and control functions: *Resource mediation* mechanisms, i.e. functions to locate resources or entities, and *resource control* mechanisms, i.e. functions to permit, prioritize, and schedule the access to resources. *Pure P2P* architectures are implementing both mechanisms in a fully decentralized manner [3], while *Hybrid P2P* systems utilize central entities that collect mediation data. An example for a Hybrid P2P system is the eDonkey filesharing

protocol, where the index servers collect and distribute file location information about all peers.

The desire of mobile network operators is to add value to the P2P data flows and to turn P2P into a service they can charge for. When creating such services operators retain control on traffic and content. However the basic P2P user experience and connectivity should be preserved. In this paper, which is an extension of [4], we describe such a service and analyze its impact on the network usage by means of a simulation.

This paper is structured as follows. In Sec. 2, we analyze the requirements and objectives of mobile P2P systems. We also analyze the problems of mobile P2P file sharing systems, and map out possible solutions. In Sec. 3, we present our proposed mobile P2P architecture. We identify key concepts (Sec. 3.1), describe our extension of the eDonkey architecture (Sec. 3.2) and introduce the Cache Peer (Sec. 3.3), the enhanced Indexing Server (Sec. 3.4) and the Crawler (Sec. 3.5). Sec. 3.6 identifies caching parameters and introduces caching strategies for mobile networks. In order to evaluate the system performance, we define a simulation model. Sec. 4 outlines the restrictions imposed by mobile networks and concludes the cache strategy which fits best for P2P traffic. Sec. 5 presents the numerical evaluated cache strategies. Finally, Sec. 6 summarizes the efforts achieved so far and gives an outlook.

2 Requirements and Objectives of Mobile P2P Systems

Mobile wireless communication systems are in many aspects different from the fixed Internet. For the access to IP-based applications like WAP, Web or E-Mail, a great variety of access technologies such as GPRS, EDGE or the UMTS packet switched data services exists. Mobile access technologies differ in terms of the air interface, QoS-capabilities, available bit rates and underlying transport mechanisms in the core network. In the following, we consider some of these aspects and their implications for a P2P system.

The **Air Interface** is commonly seen as the bottleneck in mobile communication systems. Although 3G systems like UMTS provide bit rates up to 2 Mbps in TDD mode and up to 10 Mbps with the HSDPA technology (High Speed Downlink Packet Access), the cost of data transmissions over the air interface is generally higher than in fixed networks. This is even more true for 2G and 2.5G systems like GSM/GPRS with a theoretical maximum bit rate of 171 kbps and typically achievable bit rates between 28 and 50 kbps. Furthermore, the mean round-trip times are significantly higher than in wired systems due to the higher protocol overhead and complex error correction schemes, leading to a lower performance of especially TCP [5]. These results are also confirmed by our measurements of eDonkey via GPRS [6].

The two main restrictions of the air interface, a relatively low effective bandwidth and high latencies, make it essential to reduce the signalling overhead as much as possible to achieve an acceptable performance. Direct traffic between peers should be avoided as much as possible, since all mobile-to-mobile

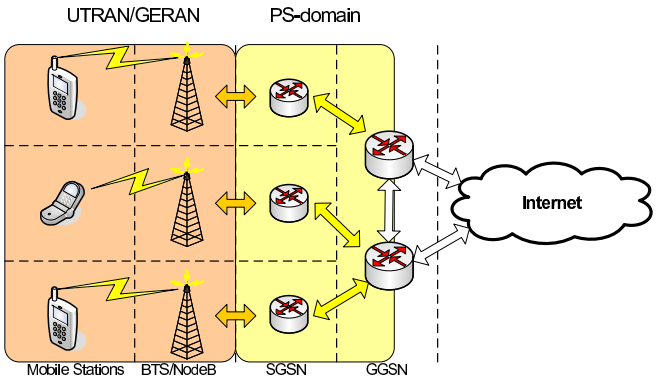

Fig. 1. Simplified scheme of core network for packet data transport

transmissions use twice the amount of air interface resources if compared to mobile-to-fixed-network transmissions.

Furthermore, the limitations of the transmission power and battery capacity cause the uplink bandwidth to be significantly more expensive in terms of network resource usage than the downlink. So the use of uploads from mobile devices must be more efficient when asked for. Battery consumption will be a long-lasting issue for mobile devices. Therefore, mobile user equipment will continue to have a lower online time if compared to non-mobile Internet devices, on which the majority of the P2P applications runs today. Reduced online time of the peers will greatly affect the download time and thus the user experience of P2P systems.

In general, the **Core Network** of a mobile communication system is designed hierarchically. For GPRS or UMTS, the data traffic stream of each mobile traverses along core network, from the UMTS Terrestrial Radio Access Network (UTRAN) through the packet-switched domain and back. At the GGSN, the mobile hosts get assigned an IP address. Therefore, the GGSN is both the interface to the Internet and to other mobiles in the mobile domain, making it also the point in the core network where all packet traffic is concentrated, see Fig. 1. Note that generally in the core network several GGSNs exist, each serving as a gateway for a large portion of the mobile network. This hierarchical, very centralized topology is in strong contrast to the flat, mesh-like overlay network topologies of most P2P systems.

One of the most important **Operational Requirements** of mobile network operators is to maintain control over the network and the ability to charge for provided services. Furthermore, operators would like to keep traffic in their own domain to avoid cost due to inter-domain traffic. This is true for both mobile and fixed-line operators. If mobile P2P is integrated into the service structure, it is therefore necessary to provide means for controlling and for charging. On the one hand, the control mechanisms for a mobile P2P system must be carefully chosen in order to avoid the total degeneration towards a centralized system. Control mechanisms should not tamper fundamental P2P concepts such as de-

centralization. The business model used for charging should also comply with P2P applications, e.g. reward users for sharing. On the other hand, a mobile P2P system can benefit from the existing infrastructure and services of a mobile communication system. The network providers know the location, the online status and the service agreement of the mobile user, which might be useful to avoid signalling overhead and to increase the quality of service.

3 An Architecture for Mobile P2P Filesharing-Systems

P2P filesharing systems extensively utilize network resources. As an optimization the architecture is adapted to that of the underlying network. The major challenge of mobile networks is their hierarchical infrastructure. This must be reflected in an architecture for mobile P2P systems. We designed a caching mechanism that efficiently maps filesharing onto mobile infrastructures.

3.1 EDonkey Features

To meet the requirements of operator-managed services with P2P-based content distribution, a hybrid P2P architecture has been selected. The chosen architecture is based on the eDonkey P2P file sharing protocol, because of its popularity and its proven robustness. It is classified as a hybrid P2P filesharing network, as data exchange is achieved decentrally between peers while mediation is provided by centralized index servers.

The eDonkey protocol introduced the **multi-source download protocol**, which is an integral part to the scalability of P2P file-sharing. It means, that the download process for one resource may utilize multiple sources. Coordination between several sources is somewhat hard, since files could be tampered or even renamed. The multi-source download protocol relies on the MD5 Hash-IDs. All copies of a resource carry the unique Hash-ID with them and so a requester can be sure that he downloads fragments of the very same file and version. Then the fragments can be afterwards compiled into the original resource.

The eDonkey protocol addresses the free-rider problem using **fragment sharing**. Any resource fragment is shared after completion check. Thus all peers provide fragments during their download. This increases the number of early available sources and the resource access load is distributed over the community.

The fragment sharing concept forces downloaders of a resource to share completed fragments with others. Multi-source downloading scenarios culminate into fully-connected graphs[4]. The original eDonkey architecture achieves source convergence centrally, i. e. by asking a super peer. Thus a bunch of peers, called a **horde**, exists that currently access the same resource. EMule introduced *source exchange*, a decentral approach to distribute sources. Because the peers communicate with each other, they can easily notify the list of sharing peers. This behaviour makes frequent request to the super peer unnecessary.

[4] unless the maximum number of connections is exceeded

To apply a file-sharing protocol on **mobile networks**, some minor issues had to be solved. Mobile Internet devices have an IP stack and thus can in principle be fully integrated into the Internet. However, security mechanisms like firewalling, NAT boxing and VPN interfere with basic P2P operations. P2P applications can handle firewalled connections as long as there is either one partner with open ports. The eDonkey/eMule terminus *HighID* denominates a peer in case foreign hosts can initiate communications with that peer. Firewalled peers need to establish the connection themselves, other peers may reply message but cannot initiate transmissions. These are marked with *LowID* indicating that these peers need actively connection establishment. P2P networks can handle firewalled connections as long as either one peer of a direct communication allows incoming connections. Rigid firewalling can be overcome using VPN, which we have also tested. While the mobile peer software can be updated, such a restriction would abandon connections to Internet peers that run current software releases.

3.2 Extended eDonkey architecture

Our enhancements to the eDonkey architecture comprise three parts: Modifications to the *index server*, a *cache peer* and a *crawler*. The index server monitors the file popularity and exports the collected data to the cache peer. The cache peer stores popular information at the network core. The crawling peers support the index server with resources that are unknown in the mobile operator domain. Extended signalling includes information from the mobile network domain, e.g. presence information. The architecture components shown in Fig. 2 will be described in detail in this Section.

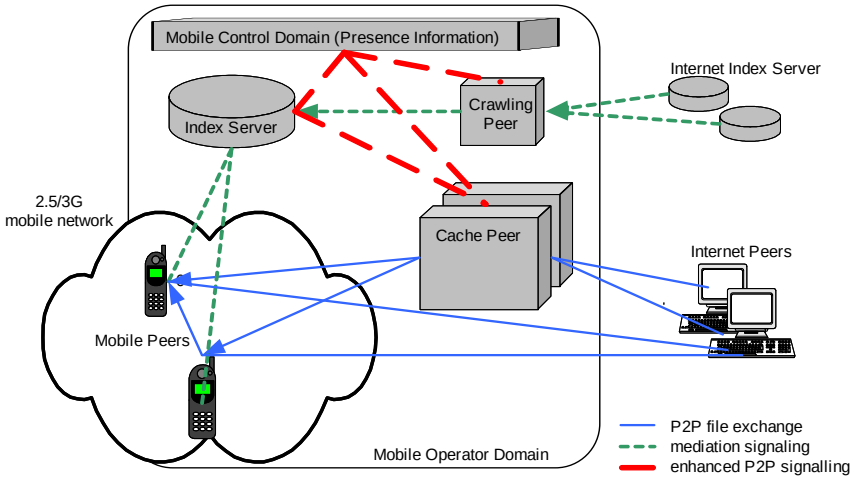

Fig. 2. Mobile P2P Architecture Overview

With centralized mediation we can gain information on popular files and cache these at the network core. This improves the overall performance, because network-edge-stored resources cause higher cost as they generate double traffic (from network edge to GGSN and then to network edge again) compared to a resource transfer from the network core. We assume here that networks of different providers are interconnected and exchange IP traffic, e. g. of a centrally operated cache peer.

Beneficial **caching for P2P** has two ingredients: identifying popular resources, and forcing peers to use the cached instance. First, to *identify popular resources* in a hybrid P2P network, information at the indexing server can be utilized. During download each peer frequently requests new sources from the indexing server; it signals the resource ID and receives a list of sharing peers. Because the indexing server can distinguish between requesting peers, it can derive statistics, which resources are heavily accessed. While the number of interested peers is incorporated, the current allocation and bandwidth remain unknown.

Second, to gain the full benefit of mobile services, the mobile peers must be encouraged to *access the cached instance* but no other. The standard peer software is designed to connect to any available source[5], thus gaining multiple waiting slots and improving chances of an early resource access. Modifications in the indexing server hide other sources as soon as a cache copy has been recorded. *Shrinking the source peer set* at the indexing server does not apply to peers that have requested sources earlier. Acquiring sources is done by merging any source candidates, there is no option for removing them. Outdated source IPs are deselected if they do not respond to ping messages.

A fixed-network cache peer can allow a significantly larger number of simultaneous uploads than any mobile peer. Because of the larger upload capacity, waiting queues are shorter and a faster response can be achieved. This is why we believe, that most often peers will receive fragments from the cached instances at the network-core rather than from network-edge peers. Our simulations show that this can reduce network traffic on the mobile backbone and shorten the download time for a file (cf. Sec. 5).

The proposed P2P components offer a value-added service. All components are optional, i. e. an operator could offer a mobile P2P service utilizing just the index server, crawler and proxy to enforce the use of local resources. Our current development of integrated features in this architecture is still ongoing.

3.3 Cache Peer

Peers access and share resources. In a way one could see them as a cache, because information that has been downloaded to the peer is also shared with the community. For caching an autonomous control is indispensable: a cache peer needs to decide, which resources should be cached. In our solution this decision will be determined using information provided from the indexing server. We did not investigate legal issues in our project.

[5] Restricted number of connected to sources from 150 up

The cache peer is a specialised peer that stores popular files at the network core to reduce the amount of expensive air-interface usage. The cache peer owes its name to the fact that we recommend to implement it as an ordinary peer. It interfaces with both the mobile domain controller and the indexing server. These negotiate what resources should be stored at the network-core. In our implementation the cache peer receives the list of popular resources from the indexing server to adopt its caching strategy. Based on this, it decides whether to fetch or to drop a cached resource.

If the access characteristic measured at the index servers signals multiple downloads of a popular file, caching is initiated. For downloading of files, the cache peer uses the same mechanism as an arbitrary peer. At that moment, resource access control is partly shifted from the network-edge towards the network-core. This signalling of chunk completion is required as it switches the super peer behaviour; from then on, all other sources except the cache peer will be hidden from further source requests. New downloads from peers on the mobile infrastructure are prevented.

3.4 Enhanced index server

The eDonkey protocol belongs to the hybrid class of P2P systems using weakly centralized resource mediation, which is provided by several index servers. The index servers provide two essential services: name search and to answering source requests. In name search a peer asks for all resources that match a given string. Secondly, when peers start to download a certain resource, they ask for peers that currently share this resource.

In our solution we recommend using a single index server that administrates all resources known inside the mobile domain. Thus popular resources can be identified and then caching can be initiated. Two extensions can deliver sufficient information to bring the caching mechanism in place. First, we log source requests by resource ID. All peers that are connected to this index server frequently ask for any new source that has been discovered lately. In reverse, from this message we gain a list of resources that are actively downloading by now.

Second, we alter the response messages of resource requests. If the cache peer is contained in the result, all other sources are rejected. Note, that the cache peer publishes the first resource fragment and this will block all other sources with possibly other fragments. However the cache peer will download only complete resources and therefore should soon reach a state where the full resource is available.

3.5 Crawler

The eDonkey community offers a large variety of resources. If the primary index server does not return enough query hits, the software automatically connects to other available index servers. For the mobile context this weakly-decentralized mediation behavior is undesirable, since the mobile domain index server cannot

keep track of popular files. Besides, other index servers cannot distinguish cache peers and therefore cannot hide other sources.

To maximize the benefits of the modified eDonkey architecture, mobile peers must connect to one of the enhanced index servers. The crawler entity is used for coordination between index servers of the mobile domain with any external index servers (other operators or Internet). The index server requests unknown resources from the crawler, which fetches mediation data from the Internet index servers. Thus, any resource available inside the global eDonkey community can be located and accessed.

3.6 Caching Strategies for Mobile P2P File-sharing Systems

The cache peer is a central element in the current mobile P2P architecture. In order to realize the objectives of the cache (minimized external and air-to-air traffic, reachability of files despite the absence of their providing mobile peers, etc.), a specially optimized caching strategy is needed. It has to take into account the characteristics of the mobile network and of the file-sharing protocol in use.

Depending on the type of storage units to be cached, one can divide the caching mechanisms in file-based and chunk-based strategies. For file-based strategies, the granularity of the cache is a single file. However, the file-based strategies are not flexible enough if only small pieces of a big file are needed, as the not required parts of the file also occupy memory capacity of the cache. A chunk-based caching strategy works on chunks which are also the natural data exchange units in many P2P networks, like eDonkey. This granularity fits well to P2P traffic, as most of the files are not downloaded completely. The users usually search through many files until they find exactly the file they need. For each promising target file, only some chunks or chunk portions (typically at the beginning and the end) are downloaded. The user may decide, whether the resource is suited for his needs. Otherwise the user will cancel the download. A file based cache strategy cannot handle such download behavior effectively, while a chunk based strategy handles this automatically, since only requested chunks are cached.

In general, the chunk-based strategy leads to a better system performance than a file-based one due to the smaller granularity, the better adaptation to the user behavior, and the better utilization of the cache capacity. But in current GPRS networks we assume most of the exchanged files to be smaller than an eDonkey chunk of 9 MB, cf. Section 5. Thus, we do not differ between a file-based and a chunk-based strategy in the following. Moreover, a file based strategy would not have the additional overhead for fragmenting and defragmenting files.

Our cache strategy consists of two aspects: *Cache population strategy* and *cache replacement strategy*. The latter is used when the cache population conditions for a new file f are fulfilled and the file size s_f exceeds the available capacity of the cache. Here a certain ranking $X_f(i)$ to the stored files is applied. Depending on the strategy, the value $X_f(i)$ may include the number $Q_f(i)$ of file requests at the index server and the amount of uploaded traffic $V_f(i)$ of the cache for file f during the time interval $[(i-1)\Delta t; i\Delta t] =_{def} t_i$. If a file is inserted

into the cache during the time interval t_i, the file f with the minimal ranking value $X_f(i-1)$ of the prior interval t_{i-1} is replaced.

RANDOM, FIFO, LRU (Least Recently Used), LFU (Least Frequently Used), LSB (Least Sent Bytes) are standard replacement strategies [7] that do not require any special application level knowledge about future caching events. E.g. the ranking value is $X_f(i) = Q_f(i)$ for LFU and $X_f(i) = V_f(i)$ for LSB, respectively. In the following sections we propose a cache strategy which is adapted onto P2P traffic in mobile telecommunication systems and incorporates the occurring restrictions of mobile P2P. This strategy is referred to as *Intelligent Memory Usage* (IMU).

IMU – Cache Population Strategy

The basic concept for inserting a file f into the cache is that the number of file requests exceeds a given threshold Θ at time $t \in t_i$, i.e. $Q_f(i-1) > \Theta$. If a file is inserted during the time interval t_i, it is not replaced during this period. The idea behind this is that we assume that the file is only inserted into the cache, because this increases the system performance during the current time interval t_i; otherwise Θ was badly chosen. The measurement values $X_f(i)$ are used in the following time interval t_{i+1} to decide which file is a replacement candidate.

Another condition for inserting a file f into the cache has to be fulfilled in order to ensure that f does not replace a more useful file with respect to the key aspects of the cache. This is done by checking if $Q_f(i) > Q_m(i)$ for the file m with the minimal replacement value $X_m(i-1)$.

IMU – Cache Replacement Strategy

One major problem of the standard cache strategies is that they do not take into account the lengths of the files they cache. For example, a FIFO strategy always removes the last file in its caching queue. If a new file of length 5 kB has to be cached, it is obviously not necessary to remove a file of length 1 GB when there are files of length 6 kB already in the cache. However, if the huge file is the last in the FIFO queue, it is removed.

One way to consider the file sizes in the replacement decision is to use a slightly modified LRU strategy. The kick-out criteria should not be directly related with the last access time for a file, like in the standard LRU. The amount $V_f(i)$ of traffic generated by this file during t_i describes in a more correct way the importance of the file, because minimizing of the traffic to external peers is a key aspect of the cache.

The size of a file on the other hand means additional costs for the cache peer, i.e. huge files use more storage resources. It would be reasonable to remove big files first, if they produce the same traffic as small files. This dependency is taken into account if the ranking value includes the factor $\left(\frac{V_f(i)}{s_f}\right)^{\alpha}$. α is the *weighting factor* and determines how strong it influences the ranking with respect to the number of file requests. s_f denotes the compressed file size because eDonkey compresses files before transmitting.

The number $Q_f(i)$ of file requests during time interval t_i is another measure indicating the popularity of a file. The more requests are seen at the index server, the more popular the file is. Hence, we define the initial ranking value of IMU

for a file f which is inserted during t_{i_0} by

$$X_f(i_0) = \Gamma_f(i_0) \text{ with } \Gamma_f(i) = \left(\frac{V_f(i)}{s_f}\right)^{\alpha} \cdot Q_f(i). \quad (1)$$

Since the file f is then not replaced during t_{i_0}, we are able to include historical values for the following time intervals. This avoids too fast reacting on very frequent changes of the file requests and smooths the variation over time. The parameter β is called the *aging factor*. We propose the following ranking value of IMU for $i > i_0$:

$$X_f(i) = \frac{\beta X_f(i-1) + \Gamma_f(i)}{2} \quad (2)$$

4 Simulation Model

In the literature, many papers about caches and their performance exist. Even for P2P networks, cache replacement policies are investigated, e.g. in [8–10]. However, in this work we propose the IMU strategy especially adapted on P2P traffic in mobile networks which is evaluated by means of the simulation.

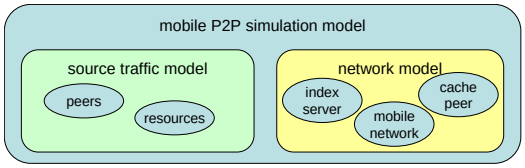

Fig. 3. Components of the mobile P2P simulation model

The mobile P2P simulation model consists of the source traffic model and the network model. The latter describes the restrictions of the P2P system because of the mobile network architecture. The source traffic model of a P2P specific system comprises the resources, i.e. the provided files, and the behavior of a peer, among other things the characteristics of a mobile subscriber. Figure 3 illustrates the components of the mobile P2P simulation model.

4.1 Peer and Resource Model

The resource model depicts the provided files and their popularity determining the file request arrival rate. In P2P networks, there is a large number N_{files} of files available. Typically, only a small number N_{pop} of very popular files generate a huge amount of traffic [11]. In our simulation, the request arrivals follow a Poisson process with rate λ_f for each file f.

We assume that there are mobile specific content types, like ring tones (MIDI or mp3) or digital images, which are shared in mobile P2P. The file sizes for different content types have been measured at the University of Würzburg. We

fitted the cumulative distribution function for the file size with a lognormal distribution which we applied in the simulation. Table 1 shows the measured parameters.

In order to reflect the highly fluctuating connection status of a mobile peer, we describe a mobile peer by an ON/OFF-process. This means that a peer is either in the ON state, i.e., the peer is present in the mobile domain and is connected to the P2P network, or in the OFF state, i.e., the peer is not connected to the P2P network. In addition, the ON period and OFF period are determined by exponential distributions with means L_{ON} and L_{OFF}. Therefore, the transition rates between these two states are $\frac{1}{L_{ON}}$ and $\frac{1}{L_{ON}}$. During the ON period, the peer participates in the P2P network by providing its own files and requesting for other files. With probability p_{new}, a peer entering the ON state shares a new file.

Another mobile specific aspect is the small memory capacity of a mobile peer. If a newly requested file exceeds this capacity, the oldest files which are shared longest are deleted (FIFO) until sufficient memory is available for storing the new file. Additionally to the mobile peers, we also consider Internet peers. The main difference between both is the access type. In our simulations presented here, we use a ratio of 2:1 between GPRS and DSL users.

In the eDonkey application, we have an upload list reflecting the simultaneously served peers and a waiting list which contains all requesting peers. The upload bandwidth is equally split among the requesting peers in the upload list. Thus, the received bandwidth is very easy to calculate in a system where the bottleneck is the receiver upload bandwidth and not the possible download bandwidth. This is not valid in a mobile telecommunication system. Here, the problem is that not only the upload bandwidth of the peers but also the download bandwidth is a limiting factor due to the small bandwidth of mobile peers.

While the upload list is limited in order to guarantee a minimal download bandwidth, the waiting list is unlimited. A newly arriving file request joins the end of the waiting list; this also holds after downloading a download unit. It has to be noted that in eDonkey, a file is structured into chunks of 9.5 Mb and each chunk is downloaded in smaller pieces of fixed size, the so-called 'download units'. Immediately after downloading an entire chunk, it is provided as source of the file in the P2P network.

4.2 Mobile P2P Network Model

We consider GPRS, since we have performed measurements of eDonkey over GPRS (without our proposed mobile network elements) in parallel to the sim-

Table 1. Measurement of the file sizes for mobile P2P specific contents

	ring tone	game	image	mp3-audio
mean [kB]	8.5762	37.9288	420.2075	4829.3306
standard deviation [kB]	9.3479	26.5833	21.3963	2305.5083

ulation, cf. [6]. We assume that a peer always utilizes its full capacity in uplink and downlink direction. It is interesting to see that, starting with the GPRS data service, asymmetric data paths are introduced by 3GPP standards that may also change over time (due to real time cell effects). Another characteristic of the air interface which affects the performance of a mobile user is the significantly high round trip time (RTT), which is also depending on the number of subscribers in a cell. We consider the data transfer of eDonkey via TCP whose throughput is then slowed down.

In the eDonkey network, a user has to be connected to an index server for participating in the network. Thus, the index server immediately notices when a peer goes online. We additionally assume that the index server discovers instantaneously when a peer goes offline due to not replied hello-packets. Therefore, the user presence information is always known to the index server and all files in the network and their corresponding sources are also known.

Each peer searching for a file asks for sources at the index server, which sends 200 sources at maximum to the requesting peer. We have limited the number of sources according to the original eDonkey source code because the searching peer requests the file at every sharing peer. Hence, the requesting peer joins the waiting list of each sharing peer. The waiting time before entering the upload list is increased and the overall throughput and the effective download bandwidth of all peers in the upload queue of sharing peers is decreased. This problem is overcome by limiting the number of retrieved sources. The index server returns uniform randomly 200 sources in order to distribute the emerging load equally within the network.

If the cache peer shares the file requested by a peer, the cache peer is always returned as first element of the source list. It is also possible to select in the simulation that the cache peer is the only returned source and all other sources are hidden.

The cache peer is assumed to be attached to the network with a link with almost infinite capacity. In this case, we have selected a 4 Gbps link, so that we can rule out a bottleneck in the interconnection of the cache peer with the downloading peers. The number of parallel upload connections from the cache is limited to 400, i.e. 400 mobile subscribers may download from the cache with 21.44 Mbps.

4.3 Abstract Model

The goal of the abstract simulation is to answer which cache replacement strategy fits best for the mobile P2P network. In contrast to the detailed simulation, we only use a subset of the parameters for the abstract simulation resulting in a much smaller computing time. The distinctive feature of the P2P network which plays an important role for the investigation of the cache is the file request arrival process. The popular files are requested very often; there are also a lot of less popular files which also generate many requests in total. Thus, the file request arrival process has to be simulated in detail, while the used transport mechanism

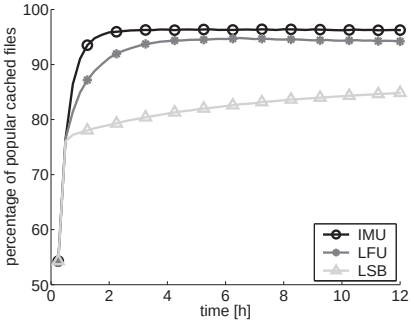

Fig. 4. Percentage of popular cached files

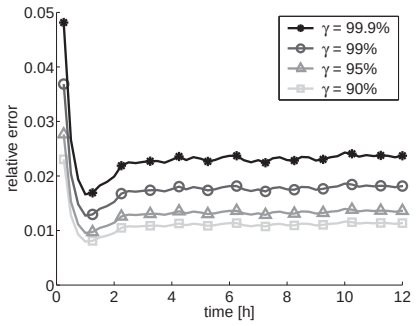

Fig. 5. Relative error of the results

or eDonkey's complex upload queue mechanism can be neglected for evaluating the performance of different cache strategies.

5 Numerical Results

In this section, we evaluate different cache strategies (LFU, LSB, IMU) by application of the abstract simulation. The target value which we use is the percentage Ψ of popular files which are stored in the cache. This value directly relates to the byte-hit-rate, the request-hit-rate, and the amount of traffic which is kept within the mobile domain. The latter aspect is very important for a provider. The more popular files are stored in the cache the more data is sent from the cache to the requesting peers with the maximal available download bandwidth of the peers and the more file requests are successfully served.

Afterwards, the influence of the proposed strategy on the mobile P2P network is demonstrated quantitatively by means of the detailed simulation. We consider the upload data volume and different interactions between cache peer and index server.

5.1 Comparison of the Cache Strategies

We simulate 100 popular mp3-files and start with 50% of them in the cache. The cache peer is dimensioned as large as the sum of the popular files' sizes. Thus, Ψ may reach 100%. The used parameters of IMU are $\alpha = 1$,$\beta = 0.5$,$\Delta t = 15$min, $\Theta = 4$. Figure 4 shows the percentage of popular cached files for each time period Δt. It seems to be astonishing that LFU is nearly as effective as the recommended IMU strategy, but the reason is the independent and identically distributed (iid) size for every file. Thus, the file request arrival rate is the main indicator for the file's popularity and LFU delivers good results. For a scenario with a more complex file size distribution, e.g. several large files, the LFU cannot return as good results as IMU, because a large, popular file may not be cached, although it produces more traffic. In this case, LSB may outstrip LFU.

In the regarded scenario, LSB comes off badly, although the transferred data volume is directly proportional to the number of file requests. Hence, LSB should

perform as well as LFU. But this is only valid if the time interval during which the measurements are taken is large enough for a download of a file to finish within this period. Figure 4 shows that IMU achieves the best results, because the transmitted data volume and the number of the file requests are considered. About 95% of the popular files are detected within a short time frame. Furthermore, IMU is a good base for estimating whether a file is really popular or not, because temporarily high or low measurement values during the last time interval are weighted by the historical values. Thus, the popularity of a file in the next time frame can be estimated by the actual measured value and the curve progression from past.

Figure 5 shows the relative error for different levels of significance γ. It is defined as the half-width of the confidence interval normalized by the mean value. We performed 1,000 simulation runs and obtained a relative error below 5% even for $\gamma = 99.99\%$. The peak at the beginning results from the random initialization of the cache.

5.2 Influence of the Proposed Strategy

The following numerical values focus on a single popular file and its influence on the transferred data volume on the upstream, while the file disperses in the P2P network. In a first step, we begin to pick out a single chunk file, i.e. a file whose file size is less than the chunk size of 9 MB, because the probability that a file is smaller than 9 MB is more than 90%.

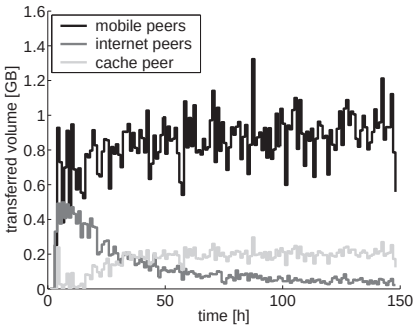

Fig. 6. Uploaded data volume of mobile and Internet peers and of the cache peer

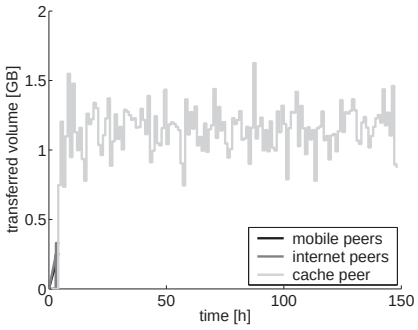

Fig. 7. Uploaded data volume with the cache peer as exclusively returned source

Figure 6 shows the data volume characteristic for a file of size 3MB with an initial diffusion of 0.1%, i.e. at the beginning of the simulation 0.1% share this file. On the y-axis, the totally transmitted data volume in uplink direction of mobile and Internet peers and the cache peer within a time interval of 1 hour is plotted. At this point, we have to refer back on Section 4.2. The index server returns randomly 200 sources of all sources to the searching peer. The latter requests download units of this file at every returned peer, independent of the access speed

of the providing peer. Obviously, 2/3 of the returned peers are mobile peers. With each mobile peer that successfully finishes its download, the probability for retrieving a mobile peer by the index server increases. Therefore, the data volume transmitted from the Internet peers decreases because the number of mobile, providing peers prevails the Internet peers.

The upload rate of the cache peer oscillates about 0.2 Gigabit per hour. This is because of the bottleneck in the downlink of the mobile peers. The downlink like the uplink has fair shared bandwidth. So the bandwidth is shared in similar parts for each uploading peer, as long as the uploading peer can provide this amount of data. With too many active downlink connections at a specific peer, the cache peer is not able to bring its high bandwidth capacity to bear.

Figure 7 shows the same scenario, but the index server answers source queries for the file only with the cache peer, as soon as the cache peer provides the file. Neither the mobile peers nor the Internet peers upload data, after the upload from the cache peers has started. As we can see, this modification is required in order to utilize the cache peer completely.

Summarizing, the index server should answer a file request not only with the sources but also with the available upload bandwidth in order to minimize the download time. In this case, the cache peer is always preferred. The simulation results are given in Table 2. Another point is that the user behavior should be adapted in mobile P2P environment. E.g. in the case of downloading 8 files, one from the cache peer and the other from other mobile peers, the downlink bandwidth is equally split between all data connections. In this case, the upload bandwidth of the cache peer cannot be utilized. Thus, it would be more appropriate only to download 2 or 3 files in parallel. Although the throughput is the same in both cases, the time for a single file download is much lower in the latter case.

Table 2. Percentage of traffic kept in the mobile network and the byte-hit-rate

cache peer...	as single source	among other sources
traffic within mobile network	99.68%	89.06%
byte-hit-rate	99.18%	15.26%

6 Conclusions

Current access and mediation control mechanisms has been discussed for applicability in the mobile context and as a result the eDonkey file sharing system has been enhanced by caching peers and crawlers. The new architecture for a mobile P2P service has been implemented for GPRS (2.5G) networks and has also been tested with UMTS. With the eDonkey index server, the three components adapt the resulting overlay network to the core structure in 2.5G/3G networks and

especially to the needs of mobile operators and subscribers, thus improving the P2P performance.

Users remain in charge of access control, while network operators gain control on mediation of resources. This influence allows for keeping some P2P traffic inside the operator's network. Popular content will be cached at a central instance to reduce traffic. This also remedies bandwidth shortages caused from repetition of data transfers, which are observed in today's access networks, Unlike other mechanisms that oppress P2P traffic, this architecture offers a network-supported service that allows peers to cooperate with the global community.

The simulation model for mobile P2P architectures includes the proposed IMU cache strategy and the numerical results show good adaption outperforming standard cache strategies, like LFU or LSB. In future work, we will investigate more influence factors, like the mobile subscriber behavior or the mobile access type.

References

1. Azzouna, N.B., Clerot, F., Fricker, C., Guillemin, F.: Modeling ADSL traffic on an IP backbone link. In: Annals of Telecommunications, Traffic engineering and routing. (2004) 1260–1314
2. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. IEEE/ACM Trans. Netw. **12** (2004) 219–232
3. Barkai, D.: Peer-to-Peer Computing. Intel Press, Hillsborow, OR (2001)
4. Andersen, F.U., de Meer, H., Dedinski, I., Kappler, C., Mäder, A., Oberender, J.O., Tutschku, K.: An Architecture Concept for Mobile P2P File Sharing Services. In Dadam, P., Reichert, M., eds.: GI Jahrestagung (2). Volume 51 of LNI., GI (2004) 229–233
5. Meyer, M.: TCP performance over GPRS. In: First Wireless Communications and Networking Conference (IEEE WCNC), New Orleans, MS (1999) 1248–1252
6. Hoßfeld, T., Tutschku, K., Andersen, F.U.: Mapping of File-Sharing onto Mobile Environments: Feasibility and Performance of eDonkey with GPRS. Technical Report 338, University of Würzburg (2004)
7. Tanenbaum, A.: Modern Operating Systems. 2 edn. Prentice Hall (2004)
8. Wierzbicki, A., Leibowitz, N., Ripeanu, M., Wozniak, R.: Cache Replacement Policies Revisited The Case of P2P Traffic. European Transactions on Telecommunications, Special Issue on P2P Networking and P2P Services **15** (2004)
9. Leibowitz, N., Bergman, A., Ben-Shaul, R., Shavit, A.: Are file swapping networks cacheable? Characterizing P2P traffic. In: 7th Int. WWW Caching Workshop, Boulder, CO (2002)
10. Iyer, S., Rowstron, A., Druschel, P.: SQUIRREL: A Decentralized, Peer-to-Peer Web Cache. In: Twenty-First ACM Symposium on Principles of Distributed Computing, Monterey, CA (2002)
11. Wierzbicki, A., Leibowitz, N., Ripeanu, M., Wozniak, R.: Cache Replacement Policies For P2P File Sharing Protocols. European Transactions on Telecommunications **15** (2004) 559–569