

Hybrid Peer-to-Peer Content Sharing in Mobile Networks

Péter Ekler

Department of Automation and Applied Informatics, Budapest, Hungary
Email: peter.ekler@aut.bme.hu

Imre Kelényi

Department of Automation and Applied Informatics, Budapest, Hungary
Email: imre.kelenyi@aut.bme.hu

István Dévai

Department of Automation and Applied Informatics, Budapest, Hungary
Email: istvan.devai@aut.bme.hu

Balázs Bakos

Nokia Siemens Networks, Budapest, Hungary
Email: balazs.bakos@nsn.com

Attila Kiss

Nokia Siemens Networks, Budapest, Hungary
Email: attila.kiss@nsn.com

Abstract— As the capabilities of mobile phones are increasing and they are able to consume different type of multimedia contents, it is important to provide an efficient content sharing solution for them. Since people usually store their content on their personal computer, we need architecture for content sharing which supports mobiles and personal computers as well. In this paper we propose an innovative content sharing solution, called Swarm, which offers the following benefits to service providers: cost efficient, mobility support, necessary control points to prevent misuse of the service and backend for the stored information. Swarm is a hybrid solution, it combines the advantages of peer-to-peer and client-server systems and it also considers the special abilities of mobile phones by enabling local cooperation. During the paper we introduce the proposed architecture with the main features, discuss our experience with the reference implementation, demonstrate calculations about the cost efficiency of Swarm and propose future plans for further improvements.

Index Terms— BitTorrent, Mobile phones, Local cooperation, Efficient content sharing, Hybrid solution

I. INTRODUCTION

Content distribution over computer networks has always been a big challenge. Before one is to create this type of service it is necessary to decide whether to design a client-server solution with central units or a peer-to-peer type of solution where the content is distributed with the help of network members.

The general architecture becomes even more complex if we plan to support mobile phones and other types of mobile devices as well. An interesting topic is how to create an efficient content sharing solution that does not require a large central server park but at the same time is able to support several clients as well as mobile devices.

This paper proposes a content sharing system for service providers that is efficient, supports mobility and provides a control point. Since mobility support is one of the key goals of the system, it is necessary to consider the special capabilities of these devices.

Mobile phones have several advantages like supporting several types of networking technologies as well as short range communication technologies, the latter of which can be utilized even in content sharing systems. Another advantage is that advanced software platforms of mobile phones make it possible to consume different types of multimedia content. In addition, the increased storage capacity of mobile phones makes it possible to store more and more of our favorite video or music on our device. Nowadays it is common to have 2, 4 or even 8 GB memory in a single mobile phone. With the evolution of mobile phones, the need for an efficient content sharing mechanism has emerged.

Despite these advantages mobile phones have limitations. The most significant from a content sharing point of view is the reduction in battery life-time if the radio is on.

The high-level architecture of a very simple content sharing system that supports mobile phones and enables

local cooperation (e.g. via Bluetooth) contains two types of the elements. One of them is the server, which provides the content and the others are the mobile clients that download the content. Besides, mobile devices can also recognize other nearby phones downloading the same content and can download parts of the content from one another via Bluetooth. Figure 1 illustrates this basic architecture.

In this approach we have to implement some kind of mechanism in mobile phones in order to search for nearby devices and find possibilities for cooperation. In what follows, we will refer to this as *local cooperation*.

The increasing capabilities of mobile devices allow the implementation of a new range of applications. An important set of applications are peer-to-peer (P2P) applications. The pure P2P solutions are very efficient from the service provider point of view (especially if the content is popular). Pure P2P solutions include Gnutella (LimeWire, BearShare), Napster, Kazaa. In these systems there is no need for large central servers, the participants (usually PCs connected through the internet) transfer the content between one another. In these systems the infrastructure and operational costs (internet access, energy consumption) are implicitly divided among the end-users. Content search is also conducted in a distributed manner. The problem is the lack of mobility support. The mobile devices have different characteristics than the PCs that these P2P systems have been designed for. They usually do not have a public IP address, cannot store large amounts of data, network bandwidth and cost are usually an issue, and the churn-rate (connecting to and disconnecting from the network) is high. In addition, there is no such as a control point in these systems responsible for removing content protected by third party copyrights and for creating a premium content service.

In this paper we propose an architecture that combines the P2P systems and client-server systems into a hybrid solution. This so-called Swarm architecture is based on the efficient BitTorrent protocol [1] (Section 3). This protocol is known as a P2P protocol, but it has certain central elements (tracker, torrent file catalogs) that make it a good building block for a hybrid system (Section 6). In Swarm home PCs are the potential members who can participate in content sharing. The term “home PCs” refers to those personal computers which are helping in the content distribution.

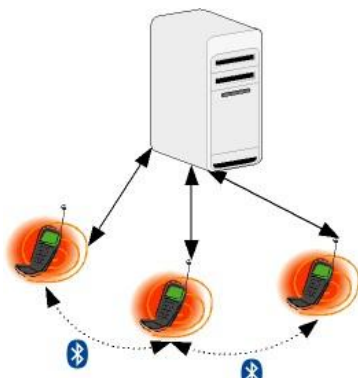


Figure 1. A download mechanism, which supports local cooperation

The rest of the paper is organized as follows. Section 2 describes related work in the area of cognitive and cooperative networks and P2P solutions on mobile phones. Section 3 describes the key aspects of the BitTorrent technology. Section 4 demonstrates how to bring the BitTorrent to mobile phones. Section 5 discusses how we can increase the efficiency of the mobile BitTorrent clients by enabling local cooperation. Section 6 describes the architecture of Swarm which uses mobile peer-to-peer clients. Section 7 describes the mobile clients for SWARM in more detail. Section 8 investigates the efficiency of SWARM, while Section 9 describes how we can further increase its efficiency by enabling local cooperation. Finally, Section 10 concludes the paper and proposes issues for further research.

II. RELATED WORK

Today wireless communication networks are increasingly becoming aware of the conditions of their component parts and surrounding environment. Cognition, a continuous process involving sensing, reasoning, understanding and reacting, can be applied to wireless networks in order to adapt the system to the highly dynamic wireless ecosystem. Fitzek et al [2] present a detailed overview of the rapidly evolving topic in modern communications: cognitive wireless networks. The ultimate goals are to enhance the efficiency in the use of radio resources as well as to improve both link and network performance. They introduce several modern communications and cooperative strategies among wireless devices which can be used in other applications as well.

One of the most popular P2P protocol is BitTorrent. Despite its popularity the actual behavior of this system over prolonged periods of time is still poorly understood. Pouwelse et al. [3] present a detailed measurement study over a period of eight months of BitTorrent. They show measurement results of the popularity and the availability of BitTorrent, of its download performance, of the content lifetime, and of the structure of the community responsible for verifying the uploaded content. The results are that the system is quite popular, but the number of active users in the system is strongly influenced by the availability of the central components. They also found that 90% of the peers experienced speeds were below 65 kB/sec. From the lifetime point of view they showed that only 9,219 out of 53,883 peers (17 %) have an uptime longer than one hour after they finished downloading. For 10 hours this number has decreased to only 1,649 peers (3.1 %), and for 100 hours to a mere 183 peers (0.34 %).

Guo et al [4] show that existing studies on BitTorrent systems are single-torrent based, while more than 85% of all peers participate in multiple torrents according to their trace analysis. They present measurements and analysis for multiple torrent environments. In a BitTorrent system, the service policy of seeds favors peers with high downloading speed in order to improve the seed production rate in the system. They demonstrate with measurements that the higher the downloading

performance peers have, the less uploading service they actually contribute. This indicates that peers with high speed finish downloading quickly and then quit the system soon, which defeats the design purpose of the seed service policy.

Free-riders in BitTorrent systems are those who download but do not upload any data. This may happen when the user specially configures or modifies his client software. In our case this behavior is not tolerated because this way, users are able to manually decrease the possibility of cooperation between each other. In [5] the authors present collected BitTorrent usage data across multiple file-sharing communities and analyze the factors that affect users' cooperative behavior. They found evidence that the design of the BitTorrent protocol results in increased cooperative behavior over other P2P protocols used to share similar content. They also showed that in torrents with a relatively low number of seeders, BitTorrent is successful in penalizing free-riding, in effect by increasing the download times of peers that free-ride. However, in torrents where seeders are plentiful, i.e., torrents with high seeding ratios, free-riders may download faster than collaborating peers.

In [6] the goal of the authors was to develop an analytical model of a free-rider in a BitTorrent network. They derived a continuous-time Markov model of a free-rider. Unlike previous analytical models which capture the behavior of the network as a whole, their proposed model is able to analyze the performance from the user's perspective.

Minglu Liy, Jiadi Yuy and Jie Wu [7] present a fluid model with two different classes of peers to capture the effect of free-riding on BitTorrent-like systems. With the model, they have found that BitTorrent's incentive mechanism is successful in preventing free-riding in a system without seeds, but may not succeed in producing a disincentive for free-riding in a system with a high number of seeds.

While it is well-known that BitTorrent is vulnerable to selfish behavior, Locher et al. [8] demonstrate that even entire files can be downloaded without reciprocating at all in BitTorrent. To this end, they present BitThief, a free-riding client that never contributes any real data. They showed that simple tricks suffice in order to achieve high download rates, even in the absence of seeders. They also illustrated how peers in a swarm react to various sophisticated attacks.

The goal of our research is to propose an efficient content sharing solution thus somehow we have to be prepared for free-rider behavior. Karonen and Nurminen [9] have introduced a mechanism for P2P network which can be used to avoid this negative behavior. They propose a P2P credit system especially targeted for the cases when mobile devices join P2P networks. Instead of limiting the incentive and reputation mechanisms to a single device their scheme encompasses all the connected devices of a user. They discuss the limitations of today's incentive schemes from the wireless devices point of view, present the P2P credit system concept and highlight its operation

with a number of use cases. They also illustrate the potential of their solution via mathematical analysis.

They inspect a number of scenarios and use them to estimate the effect of the P2P credit system. They analyze how the system behavior would improve if their proposed incentive credit mechanism encouraged a bigger number of the PC users to let their PCs share content all the time.

A key performance metric in a P2P sharing network is the ratio of the number of peers sharing content n_s and the number of peers downloading content n_D . This ratio between sharers and downloaders, which we denote by R , has a fundamental effect to the efficiency of file sharing.

The download completion time T depend on R ,

$$T = f(R). \tag{1}$$

The exact form of function f is difficult to know but experimental evidence allows to provide some estimates. According to the data in [10] T increases by 60% when R changes from 90% to 60%. When R changes from 90% to 40% the value of T becomes double. The value of R is thus a rather good indicator of the download time of the shared content. The ratio of sharers and downloaders can be expressed with the following formula

$$R = \frac{n_s}{n_D} = \frac{ma_m s_m + (1-m)a_p s_p}{ma_m d_m + (1-m)a_p d_p}. \tag{2}$$

where m is the percentage of mobile peers, a_m, a_p is the percentage of time the peer is active for mobile and PC peers respectively, s_m, s_p is the percentage of the active time that the peer is sharing content, and d_m, d_p is the percentage of the active time that the peer is downloading content. Karonen and Nurminen use this formula to analyze different scenarios.

A. All peers are equal

The basic assumption in most P2P content sharing studies has been that all peers are roughly equal. In this case $a_m = a_p = a, s_m = s_p = s, d_m = d_p = d$ and the formula reduces to

$$R = \frac{s}{d}. \tag{3}$$

The case $s < d$ presents the case where freeriders are consuming services without reciprocal contribution.

When the network consists of both mobile and PC peers the assumption is that both of them are used in the same fashion. However, energy-consumption [11], communication cost, and restricted network access (e.g. through NATs) limit the possibilities for the mobile peers to share their resources. This reduces s which results into smaller value of R and thus longer download times.

B. Mobile peers are only used for downloading content

In this scenario mobile peers are taking the freerider role to save battery and network traffic. Mobile peers never share any content while the PC peers share all the time while they are active. With parameter values $a_m = a_p = a, s_m = 0, s_p = 1, d_m = d_p = d$ the formula becomes

$$R = \frac{1 - m}{d}. \tag{4}$$

The percentage of mobile peers in the network thus controls the ratio. As long as the number of mobile peers is small they only have a minor performance degrading effect. However, if the consumption of multimedia in mobile devices increases and these devices increasingly access P2P networks to download content directly the operation of the network starts to suffer.

C. Mobile peers only for downloading, PC peers always active and sharing

This case corresponds to a possible situation which they hope the P2P credit system would guide the users. In this scenario mobile peers are only used for downloading. However, the performance degradation is compensated by the increased number of PC peer resources that are available in the network.

Especially in the industrialized countries most users of multimedia mobile phones also have their own PCs. With the P2P credit system as an incentive the assumption is that users would be able to increase the amount of time that their home PCs are sharing content. Instead of sharing only during the download operation (which is common today), the PCs would be sharing the content all the time. This would easily more than compensate for the fact that the mobile devices are not sharing at all.

If we assume that the same content is shared between mobile devices and PCs, and that the users spend the same amount of time to download content with both devices we come up with the formula ($a_m = t, a_p = 1, s_m = 0, s_p = 1, d_m = 1, d_p = k$)

$$R = \frac{1 - m}{t}. \tag{5}$$

where t is the percentage of time when the person is actively downloading content. E.g. downloading a movie a day using both devices with each download taking around 2.5h (according to [12] downloading a 1GB movie file would take 2.5 hours) would result into $t = 0.1$.

If we consider a case where $m = 0.5$, which corresponds to the case where every mobile users also has a home PC that is sharing content all the time, then $R = 5$. This is an impressive figure indicating that there are five times more peers sharing the content than downloading it.

The above analysis is based on a number of assumptions, however it shows that if we have a method which encourages home PCs to contribute in a P2P network, it can be used to decrease the download completion time T .

Later, when we introduce the architecture of Swarm (Section 6) it will become clear that home PCs have an important role in our solution because they can decrease the degree of overloading the central element. This way the previously introduced credit mechanism could be used to increase the number of home PCs in the network.

III. EFFICIENT CONTENT SHARING WITH BITTORRENT

In this section we briefly describe the BitTorrent protocol because it is important for understanding the rest of the paper and the calculations related to the efficiency of Swarm. Besides that later we will also use several terms in connection with BitTorrent technology which will be introduced in this section as well.

BitTorrent concentrates on efficient, distributed file transfer [1], it is designed to distribute large amounts of data without incurring the corresponding consumption in costly server and bandwidth resources; hence, it can be adequate for mobile file-sharing.

A. BitTorrent units

With BitTorrent, when several peers (clients) are downloading the same content simultaneously, they send different pieces (small parts of the whole content) of this content also to each other. This behavior belongs to one of the main advantages of the protocol.

Sharing files over BitTorrent requires at least one dedicated node in the network which is called as "Tracker". The tracker coordinates file distribution and can be asked for the shared resources which are under its supervision. If a peer requests for a specific content from the tracker, it returns several IP addresses which belongs to other peers who have the whole content (seeder) or parts of it. The network can contain also several trackers in order to distribute the traffic further.

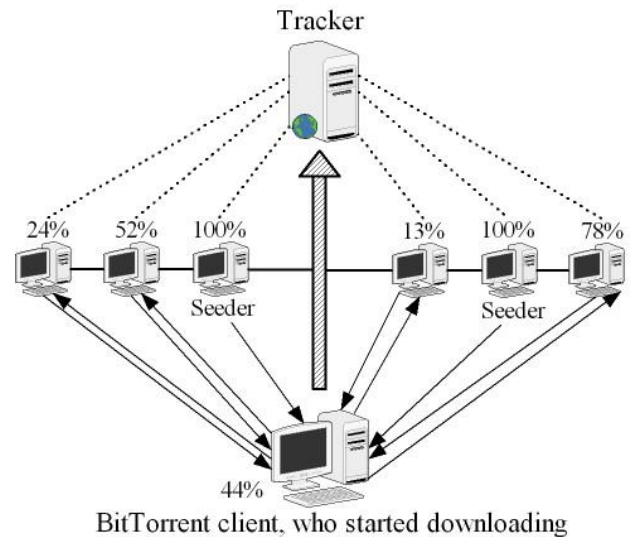


Figure 2. BitTorrent protocol

Figure 2 shows how download and upload can work at the same time in BitTorrent. The percentages illustrate how much of the content have already been downloaded by the relevant peer. Note, that nodes marked as *Seeder* have already downloaded the full content and their current task is only to share it with the rest of the network.

B. Torrent file

The process of content sharing via BitTorrent begins with the creation of a *torrent file*. This file contains metadata related to the tracker and the files to be shared. Figure 3 illustrates the structure of a torrent file.

We can estimate the size of a torrent file with the following formula:

$$S_{\text{torrentfile}} = S_{g_i} + S_{c_i} + S_{c_{trli}} \quad (6)$$

A torrent file has three main parts. The first one is where general information is stored. This part has constant size (S_{g_i}), it contains the HTTP address of the tracker, and information about the creator. The second part contains the names, the sizes and the directory hierarchy related to the shared files. The size of it is also small (S_{c_i}), but it depends in a small compass from the amount of shared files.

The shared content in BitTorrent is transferred in small pieces. The size of the pieces is pre-defined; it is usually around 64-256 KB. The third part of the torrent file contains the SHA1 hash values in 20 bytes of these pieces. During the protocol BitTorrent calculates another 20 byte SHA1 value from the last two part of the torrent file, which is used to identify the torrent in the network.

The piece hashes are used during the protocol to check the consistency of the downloaded pieces. This way the size of the third part ($S_{c_{trli}}$) depends from the size of content. E. g. if the size of the shared content is 2048 KB and it is divided into 64 KB pieces then,

$$S_{c_{trli}} = \frac{2048KB}{64KB} * 20B = 640B. \quad (7)$$

We can see from the previous calculations that the size of a torrent file is relatively small compared to the whole content. Usually it is around 4KB, thus transferring it over the network does not consume significant amount of bandwidth. A torrent file can be transferred using numerous ways; however it is usually hosted on a web server.

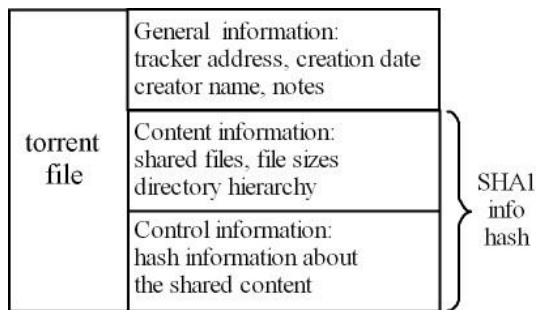


Figure 3. Torrent file structure

The size of the torrent file will be important when we describe calculations related to the efficiency of our Swarm solution (Section 8).

C. Content distribution via BitTorrent

In order to share some content, the torrent file related to the content needs to be registered to a tracker;

afterwards, any client which obtains the torrent file can connect to the swarm and download or upload the content. Peers are required to periodically check in with the tracker (this process is called “announcing”); thus, the tracker can maintain an up-to-date list of the participating clients.

Concerning legal issues, BitTorrent, similarly to any other file transfer protocol, can be used to distribute files without the permission of the copyright holder. However, a person who wishes to make a file available must run a tracker on a specific host and distribute the tracker’s address in the torrent file. This feature of the protocol also makes possible to locate the trackers who are responsible for illegal contents. It is far easier to request the service provider of the tracker to shut the server down than finding every user sharing a file on a fully decentralized peer-to-peer network.

IV. APPLYING BITTORRENT ON MOBILE DEVICES

The first experimental steps towards bringing P2P technology to mobile phones have already been taken with the implementations of popular content sharing protocols, Gnutella and BitTorrent, for high end mobile phones [13]. These applications, Symella [14] and SymTorrent [15], are available in source code at the Budapest University of Technology and Economics. However, these applications are implemented on the Symbian platform (for Nokia S60 devices) which limits their use to a subset of high end mobile devices.

The mainstream mobile phones are also able to consume most of the multimedia content like images and mp3 music. This way bringing efficient content sharing solution to those phones is also attractive. In a previous paper [16] we have discussed about bringing BitTorrent to Java ME platform.

Bringing BitTorrent technology to mobile devices is a challenging task due to the limited resources available on mobile phones. The situation is more difficult if the target devices are not only smart phones but also mainstream phones with even less resources.

When we speak about P2P solutions on mobile devices it is important to emphasize that in order to become a full member of a P2P community, the mobile device must fulfill the following specifications:

- a) Ability to connect to the network via the specific P2P protocol.
- b) Download and upload content.
- c) Publishing feature, which means that mobile users should also be able to create and share new contents to the P2P community.

If the device is only able to connect to the network and download content but cannot upload, then it is not a full member of the community. This behavior is not highly appreciated by the other members. In many cases (depending on the specific P2P implementation) the P2P network and the algorithms detect this selfish behavior and punish the specific peers with lower service rate or worse service quality.

A. SymTorrent

SymTorrent is the first and, at the point of writing, the only BitTorrent client for Symbian OS. Our main goal was to transfer the BitTorrent technology to a mobile platform and demonstrate the possible use cases of BitTorrent-based file sharing on a real device. In addition, we developed some new concepts during the development which resulted in an integrated client-tracker application. SymTorrent not only works as a standard BitTorrent client, it also has its own built-in tracker. Running a tracker on a mobile phone may seem a bit bizarre at first but it can have several interesting use cases. Sharing files instantly between a small group of users without depending on external servers is just one example.

Since SymTorrent was written in native C++, we did not have difficulties with accessing the more advanced services of the platform. Symbian OS is a multithreaded operating system that is capable of hosting applications using several sockets, file-access and complex user interface.

Since Symbian-based phones use different screen sizes and input methods, we implemented the UI-independent parts of the application in a separate DLL. This way of porting to different devices is much easier.

During the last year, SymTorrent has been downloaded more than twenty thousand times. Most users employ it as standard BitTorrent client for downloading files with their mobile phones through GPRS, 3G or WLAN.

B. MobTorrent

MobTorrent is a complete Java ME-based BitTorrent implementation supporting both downloading and uploading. Since mainstream phones usually support Java ME with MobTorrent we are able to involve them in a P2P network.

Although software development is easier in Java ME than in Symbian C++, we faced several difficulties during the implementation of MobTorrent. These issues are related to the Java ME implementation of the different mobile platforms.

The most significant problem is related to the socket handling implementation of Java ME [16]. A P2P application usually has to connect to several addresses before it finds one suitable. Several peers might be offline or not responding at all. The problem is that the timeout when the system realizes that the address is not responding is 244 seconds on Series 40 devices and 163 seconds on S60 devices. An attempt to connect to a peer that is not online causes thus a long delay. The problem is further complicated by the limitation of S40 platform: it can handle only one connection request at the same time (S60 is able to handle 8 connection requests in parallel).

This shows that P2P application has different platform requirements than other type of applications and that they bring up problems that are not experienced by other ones.

With the proposed Swarm architecture (Section 6) we are able to handle this issue efficiently, since the central unit is able to filter offline addresses for mobile clients.

C. Performance of SymTorrent and MobTorrent

In order to compare the two mobile BitTorrent clients we have tested them in a real environment [16]. Our test file was the torrent of the original BitTorrent client: bittorrent441.torrent.

TABLE I.
DOWNLOAD SPEED (KB/SEC) COMPARISON IN REAL ENVIRONMENT

(kB/sec)	3G	WLAN
SymTorrent N91	50	220
MobTorrent N91	48	79

In a real environment the download speeds depends on several factors that are beyond our control. We chose a popular torrent and ensured that all of the peers were alive during the download in order to avoid delays of the long socket timeout. In Table 1, we can see that with 3G network connection, the performance of the J2ME applications was comparable with the Symbian.

With WLAN connection, SymTorrent was much faster. It is due to the previously mentioned limitations of Java ME and the overhead of the Java Virtual Machine.

Fig. 4 illustrates SymTorrent and MobTorrent in screenshots.

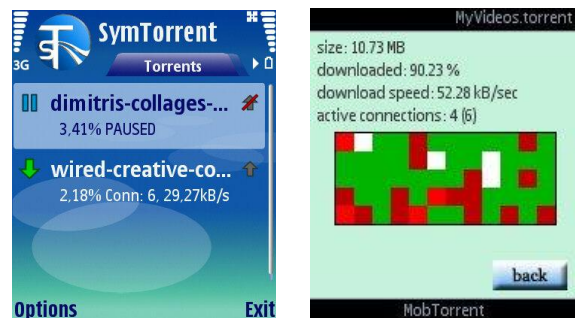


Figure 4. SymTorrent and MobTorrent

V. EXTENDING BITTORRENT WITH LOCAL COOPERATION

Improving the performance of BitTorrent in terms of transfer speed and energy consumption is a key goal of future protocol upgrades. One very promising concept is local cooperation, which can improve the speed of downloads and lower the energy consumption of the client.

Local cooperation [17] exploits the fact that short-range network connections can operate faster and more energy efficiently than long-range technologies. By enabling BitTorrent clients to connect to each other over short-range links, cooperative clusters can be formed. Peers in a cluster cooperate to obtain the content of a torrent. Since using the short-range links is more efficient, the goal is to minimize data traffic with peers connected over the long range links and obtain as much data as possible from the local cluster. However, to achieve this, the standard BitTorrent protocol must be complemented with additional protocol messages and algorithms. The locally connected peers must share extra information on their status with each other. The extended

protocol is referred to as GridTorrent. While GridTorrent remains compatible with standard BitTorrent peers, cooperative peers must support a couple of new messages to enable exchanging information in the local cluster.

The topology of a network with cooperative peers is illustrated in Figure 5. The cooperative peers, which are marked with the phone icon, are connected over short range links, typically over Bluetooth. They form the local cluster, which is marked with a cloud. Besides the locally connected peers, cooperative mobile peers also establish connections with peers on the Internet, over long range network interface, which can be HSDPA, GRPS or even WLAN.

Regarding the network interface used in the local cluster, there are several factors that affect the overall efficiency of the approach. The least overhead is generated if the local network interface supports broadcasting (e.g. UDP over WLAN). In this case each packet received by a local peer over the long-range link is broadcasted only once to the peers in the cluster. If broadcasting is not supported (e.g. TCP over WLAN), a packet must be sent to every peer in the cluster. Bluetooth is a special case. Theoretically, Bluetooth could support full point-to-point networks (scatternet), current mobile devices only implement the piconet scheme. Piconets have one master peer and up to seven slave peers. Connections can only be established by the master, which means that all data traffic between the slaves must be relayed through the master.

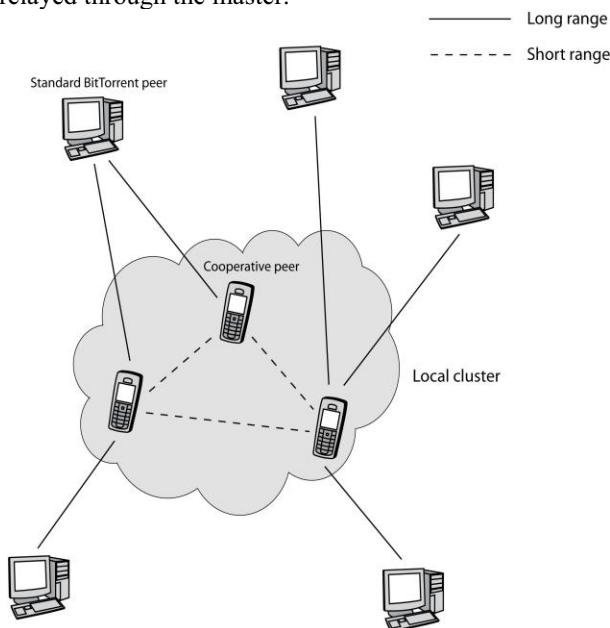


Figure 5. Topology of a cooperation-enabled network

Transferring data over two network channels instead of one obviously results in larger available bandwidth, thus better transfer speed. Let us assume that we can download content to our mobile via a BitTorrent client with T_v speed (kB/sec) and there is n mobile device nearby to which we can connect via some kind of short range network. P_b is the possibility that a nearby device downloads the same content as we and it allows

downloading the content from it (partner device). This way, the number of partner devices is:

$$n_b = P_b * n . \tag{8}$$

If we can download with T_b speed from another device via short range network and there is at least one partner device nearby ($n_b > 0$) then the download speed (T) can be calculated with the following formula:

$$T = T_v + n_b * T_b . \tag{9}$$

The energy consumption, however, is a more complicated matter. Operating two network interfaces simultaneously can also consume twice the amount of energy. The key to energy efficient cooperation are the energy per bit ratio and the energy consumption of the network interface in idle mode. Idle mode is when the peer is listening, but no data is transferred to either direction. Depending on the use-case and traffic pattern, the local network connection can be in idle mode for a significant percentage of the transfer period, which means that during this time the network interface constantly consumes some energy. If this idle energy is too high, the cooperation might not worth it at all in terms of energy consumption. Measurements performed on mobile devices using WLAN and 3G [11] shows that idle mode consumes less than 50% energy in case of 3G and less than 80% in case of WLAN which clearly indicates that there is a lot of potential in these approaches even if the local connection is not fully utilized.

VI. THE ARCHITECTURE OF SWARM

A. Introduction of the architecture

As it can be seen from the previous paragraphs, mobile clients differ from desktop machines in BitTorrent networks, due to resource constraints. In addition, the usage habits of the mobile IP network, the client software interface and the BitTorrent service are all different when put into a mobile context. To mitigate all these problems and provide a value added service to the standard BitTorrent protocol, the Swarm architecture provides the following:

a) A stable tracker and reliable seed service running on the infrastructure of the mobile operator. This has the ability to track all cell phone generated content on the BitTorrent network and if needed, provide a high bandwidth, high availability seeder to the network.

b) Provide an XML based interface so that mobile clients can browse and search the tracker for content efficiently.

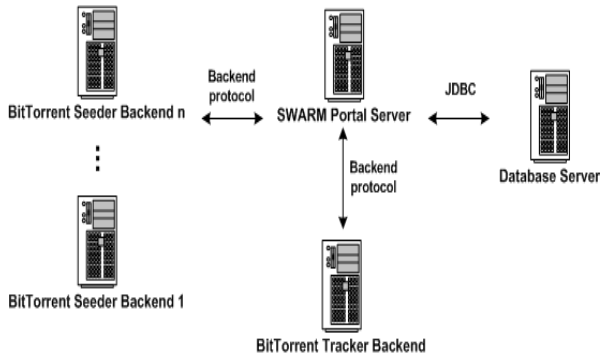


Figure 6. Central elements of Swarm

c) Provide statistics that allows optimization of traffic between mobile clients.

d) Enable the use of “home PCs”, that is, to aggregate the network traffic statistics of all the BitTorrent enabled devices of a user. This allows a cheaper seeding activity within the torrent network.

The central elements of Swarm (Fig. 6) are the following:

a) A Swarm Portal server that provides the XML based interfaces to the mobile clients, in addition to serving a traditional web application (Fig. 7) to world wide web users.

b) A database server that is storing torrent metadata needed for the directory and search functions. Currently this works by industry standard JDBC protocol.

c) Since the Swarm Portal Server does not implement the BitTorrent protocol itself, there is a tracker backend, that tracks torrents on request from the portal server. This is done by an easy to adopt interface using Remote Method Invocation (RMI) and allows plugging in various implementations of the backend.

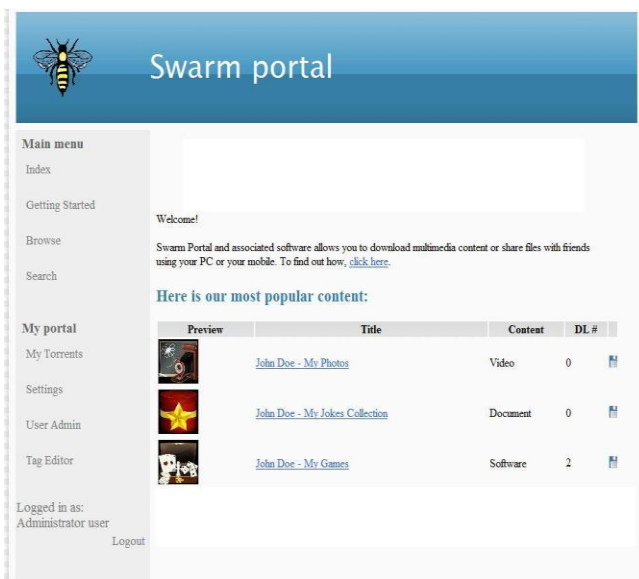


Figure 7. Swarm portal

d) A seed farm is connected to the Portal Server using the same backend protocol that allows seeding on request. The servers within the seed farm can be distributed across the globe for optimal content delivery.

Figure 8 displays the central elements of the architecture in context:

B. Tracker Functions

In a BitTorrent network, the role of the tracker is keeping track of peers within the network and supervising the upload and download process. First, we have to consider the upload case, which happens when the mobile user creates some content with his phone (video, audio, pictures, etc.) what he wants to share with the rest of the community. In the Swarm architecture, the following happens:

- 1) The user creates a torrent metadata file using our Swarm client application on the mobile device.
- 2) The user instructs the client to share this file using the Swarm architecture.
- 3) The mobile client logs into the Swarm portal server using HTTP, and uploads the torrent file.
- 4) The portal server commands the BitTorrent Tracker Backend to track the uploaded file.

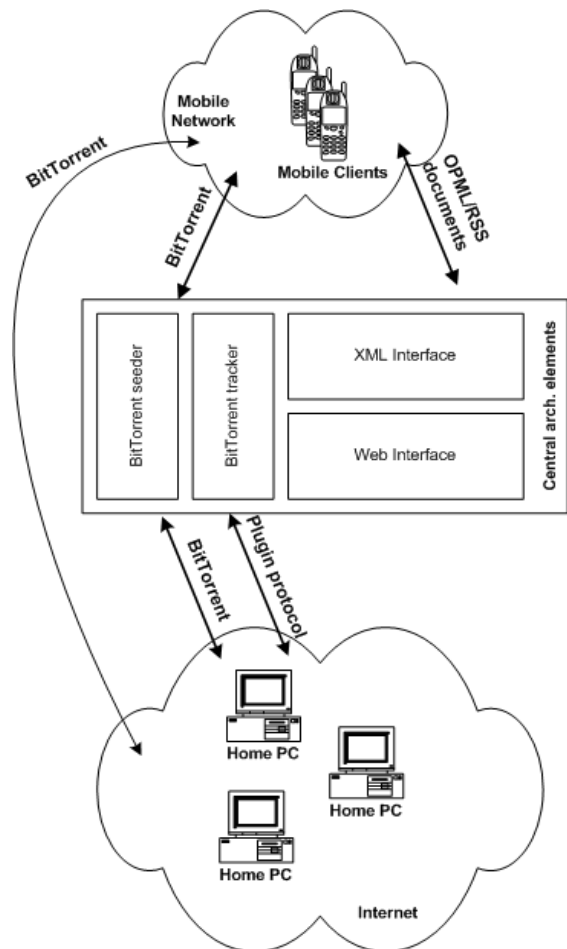


Figure 8. Central elements of Swarm in context

C. Seed Service

After the torrent file is uploaded to the portal and the backend inserts it into the tracker, other users will ask the tracker if there are any available seeds of a particular torrent. In the Swarm architecture however, content producers are also mobile devices, lacking a public IP address (they are usually behind a port restricted cone NAT). This presents the problem of initial seeding. The Swarm architecture solves this by introducing a reliable seed service in the architecture.

When a mobile phone uploads a torrent file to the portal server, the mobile client can request the backend of the portal to take over the responsibility of seeding. When this happens, one of the BitTorrent seeder backends is instructed by the portal server to enter "Forced downloading" mode. When the backend is in this mode, the mobile phone can connect to it, and upload the actual content via BitTorrent protocol; this solves the NAT traversal problem of 3G networks. When other users will contact the tracker to ask for seeds of the torrent, the tracker can return the IP address of the reliable seed service. This service has high availability and faster network connection than mobile clients have.

D. Directory and Search Service

Due to the limited display and input capabilities of a mobile phone, it is not feasible to expect the mobile user to search traditional websites for torrent files. The user needs software that works with the native GUI of the mobile operating system and allows sharing, browsing, searching and downloading of content with a click of a button. All the Swarm portal functions are available using a HTTP based interface, which returns with an OPML [18] or RSS [19] XML document that describes available content. The server stores various metadata about the content (title, author, type of content) and allows categorization and tagging.

For advanced users, the portal also offers a traditional web based interface, which can be used on a desktop PC with a web browser, thus it allows a quick and easy content management.

E. Seeding with PC seeds

Even for a mobile operator, offering huge amount of content can be costly both in terms of bandwidth and other resources (energy consumption, processing power, management costs, etc). The Swarm architecture allows the involvement of desktop PCs ("Home seeders"), which reduces the resources needed on the operator side. This works by giving the users a special software (which is basically a standard BitTorrent client, with special plugins for cooperation with the architecture), that allows mirroring shared content, thereby increasing the number of seeds in the network.

However, in the Swarm architecture one user can have multiple devices. For example a family can have one user, one PC at home and several mobile devices. They can use their mobile devices only for uploading pictures to a photo album, and the home PC will help to seed the content.

The implementation works simply by giving the IP address and port of the home PC seed to the mobile client during an initial seed. (Note that providing the port is important in order to allow multiple home PC seeds behind the same NAT via port forwarding). However, more sophisticated features can also be implemented like doing fail-over between the seeds at home and at the service provider side, maximizing performance and high availability.

F. Incentives and authentication

It is important to give incentives to users supporting the seeding activity within the BitTorrent network, thereby reducing costs at the operator side. This can be implemented by aggregating all BitTorrent traffic of a given user, so downloading activity in the 3G network is effectively compensated by Home PC seeding activity. It is also possible to incentivize other behavior, such as seeding of rare content, or seeding in peak hours, etc.

However, all these come down to authenticating all devices belonging to a particular user. Although there is no accepted standard of authentication within a BitTorrent network, it is possible to group network peers that represent the same user. This is done by customizing the torrent metadata file, when downloaded from the Portal Server. The Portal Server concatenates a unique character string to the announce URL (which is part of the metadata), thereby the torrent client is identifiable later.

G. Download via Swarm

The main advantage of Swarm from the usability point of view is that it hides the technological backgrounds from the user. In order to begin a download we do not have to know anything about P2P technologies or BitTorrent. We just have to find the proper content e. g. via the directory service and select the "start download" button.

Figure 9 illustrates how a mobile client starts downloading the selected content via Swarm.

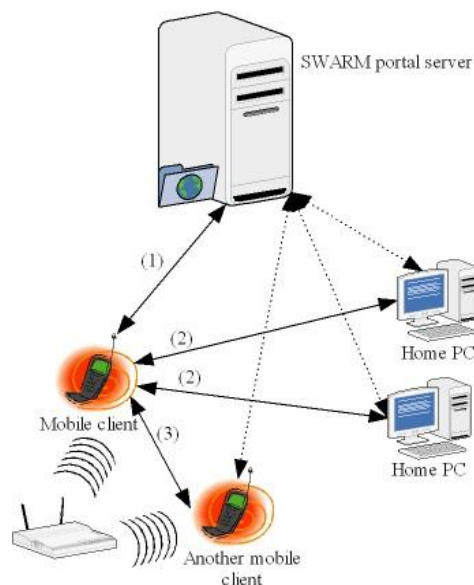


Figure 9. Download via Swarm

In the first step the clients sends a request to the Swarm server that he wants to download the selected content. Then the server checks which home PCs are seeding this content and it sends their IP addresses to the mobile client. After it the mobile client connects to these home PCs (via BitTorrent protocol) and starts the download from them (step 2). Meanwhile the server checks if there are other mobile phones who are seeding the same content and if its IP is accessible for the first mobile client (e.g. they are using the same router), then the server sends its IP address as well (step 3). In Section 9 we will extend this topology with local cooperation.

VII. SWARM MOBILE CLIENTS

By modifying mobile BitTorrent clients we managed to bring the previously introduced Swarm functions to mobile phones (Fig. 10.). This way we managed to create a P2P based content sharing solution for mobile phones where users do not have to know anything about P2P technologies or BitTorrent. The Symbian based Swarm application is a client for high end mobile devices and the Java ME based JSwarm application is a client for mainstream phones. Through these applications users are able to browse, search, download and even publish contents with the help of the Swarm server.

With the Symbian based solution users have richer user interface (UI) and more functions. For example if a user wants to publish something, it is possible to browse only between images or the videos on the phone, not the whole file system. In the case of Java ME we had to implement a whole file selection dialog. The reason is that on Symbian we are able to reach the low level APIs of the operating system.

The mobile applications have unique features that enhance the user experience and increase the usability:

a) Easy publish feature, which hides the complex operations. There is no need for the users to understand for example the BitTorrent technology.

b) The Symbian version tightly integrates with the S60 platform. For example, users are able to share content directly from the gallery or camera application by choosing the native "Send" menu option, which will offer Swarm as one of the delivery mechanism in addition to the default ones like SMS, MMS and Bluetooth.

c) Search and directory features, which enable users to find, browse and access the content on a very simple way.

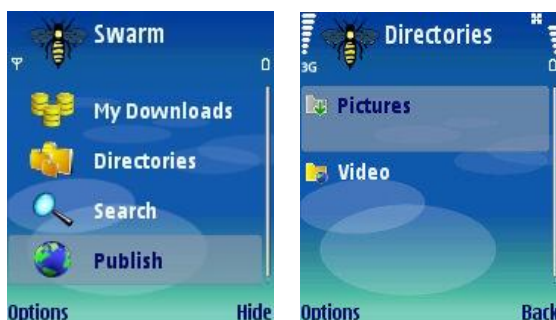


Figure 10. Swarm mobile client

VIII. THE EFFICIENCY OF SWARM SOLUTION

In this section we compare the efficiency of Swarm compared to a simple client-server solution. The main advantage of Swarm is that it contains a central element for managing the traffic and for backend functionalities, but the content distribution itself goes via BitTorrent.

The proposed Swarm architecture is basically a content sharing solution for service providers. If a service provider planes to implement a content sharing system for users, the first idea is to create a client-server like solution. However creating and maintaining that type of system is rather expensive if we consider bandwidth requirement, energy consumption, CPU usage and storage.

The main advantage of Swarm is that it distributes significant part of these costs to the user side; consequently it decreases the capital investments and operational costs for service provider. Following we describe the overhead of Swarm and we compare the operating cost of Swarm to a simple client-server solution. We will see that the cost of Swarm decreases as the network contains more and more home PCs. Besides that we will also investigate how Swarm efficiency can be increased by enabling local cooperation.

A. Storage overhead

Besides the original content the Swarm server has to store the torrents as well, which is an infinite overhead. However if we think about a bigger client-server solution where the provider has to maintain mirror servers to ensure the optimal availability then we realize that the Swarm does not need that kind of mirror server support.

TABLE II.
SIZE OF TORRENT FILES

	Torrent1	Torrent2	Torrent3	Torrent4
Content size	448 KB	1,16 MB	4,5 MB	7,78 MB
Torrent size	327 B	639 B	1,65 KB	2,66 KB
Overhead %	0.073	0.055	0.037	0.034

Table 2 represents typical torrent file sizes which are representing contents what people would expectedly download or share on their mobile devices. In Section 3 we have already discussed about the structure of a torrent file, now we can see real measurements about its size. The torrents were made with the torrent maker tool of the official BitTorrent client [20] with 64 KB of piece sizes. Torrent1 represents content from one image, Torrent2 from three images, Torrent3 from one mp3 and one image and Torrent4 from five mp3s. We can see that the sizes of the torrent files are very minimal compared to size of the represented content, thus transferring only the torrent file causes only a minimal overhead (less than one tenth of a percent) on the server.

B. Network overhead

From the networking point of view if we assume that all BitTorrent traffic goes through the servers and there

are no home PC support then the Swarm solution has overhead comparing to a client-server system.

The client server overhead for content downloads contains only the HTTP overhead. It was around 900 bytes for the GET request and 350 bytes for the response for downloading 500kB and 2MB files with an internet explorer (measured with WireShark network analyzer).

The overhead of the Swarm system for content downloads has the following components:

1) HTTP overhead: The first step in the Swarm architecture is to download the torrent file through HTTP protocol. It has the same overhead (approximately 1250 bytes for each file) as in the client-server system

2) Torrent file overhead: In the Swarm, the client has to download torrent file prior to download any content. This type of traffic does not exist in the client-server system because there the content is downloaded directly. The torrent file length depends on the content length (Table 2).

3) Peer handshake overhead: The handshake protocol must be performed once with every peer. Each handshake message has a fixed length of 69 bytes [1]. Assuming 10 peer handshakes per file [16] means 690 bytes of handshake overhead.

4) Piece exchange overhead: Each piece exchange has 9 bytes of overhead [1]. The overhead is the content length / piece size * 9 bytes. We can use 64kB piece size for an upper estimation of this overhead.

5) Announcement overhead: Peers periodically asks tracker for list of peers. The tracker response contains the list of peers. According to the BitTorrent specification [1] trackers return 50 peers by default. The size of a typical tracker response, which is: $26B + 50*(53B) = 2,61kB$. Typical announce interval is from 300 seconds to 600 seconds. Assuming a slow download speed [16] and frequent announcements for the upper limit of overhead the number of announcements are content length / speed (14kB/s) / announcement interval (300s). The total overhead is the number of announcements multiplied by 2.61kB that depends on content length.

TABLE III.
CONTENT DOWNLOAD OVERHEADS IN SWARM

[kB]	Content 1	Content 2	Content 3	Content 4
Content size	448	1160	4500	7780
HTTP overhead	1.250	1.250	1.250	1.250
Torrent file	0.327	0.639	1.650	2.660
Peer handshake	0.069	0.069	0.069	0.069
Piece exchange	0.063	0.163	0.633	1.094
Announce	0.278	0.721	2.796	4.835

Table 3 summarizes the overheads for different files with different content lengths. We used the same files as in the storage overhead calculation. The relative overhead for the client-server system OH_{cs} is the HTTP overhead divided by the content size. The relative overhead for the

Swarm system (OH_{swarm}) is the sum of all overheads in Table 3 divided by the content size. These overheads are summarized in the Table 4.

TABLE IV.
RELATIVE OVERHEADS OF CONTENT DOWNLOAD

	Content 1	Content 2	Content 3	Content 4
Content size [kB]	448	1160	4500	7780
Client-server overhead [%]	0.279	0.108	0.028	0.016
Swarm overhead [%]	0.444	0.245	0.142	0.127

The overhead of Swarm system is higher especially for larger files than the simple client-server system. But actually even the biggest overhead is less than half of a percent. For the rest of our analysis we use half percent as an upper estimation of Swarm content download overhead and use zero as a lower estimation of client-server content download overhead.

C. Benefits of Swarm

Following we describe how can we calculate the total cost of a client server solution and the Swarm solution when a user first browse the server for the right content then downloads it. After it we show how it is possible to calculate the benefit of Swarm.

Instead of the real cost of the system we use the traffic load assuming the cost of content download is proportional to the generated traffic, because higher traffic means higher CPU load, higher network load and higher energy consumption.

The download of large files (picture, music, video) is only one component of the total cost in a content management system. In a complex system there can be user management, catalogs with browse and search features, and other services and network traffic that cannot be handled (efficiently) with BitTorrent (like blogging, chat etc.). In our model we split the cost into two parts: first part is proportional to the traffic induced by content downloads (C_{cont}) and the second part contains all the rest. We call the later one as management cost (C_{mgmt}) because it is related to content management services.

The total cost of Swarm contains one parameter that is the seed ratio (S). It means that there are other seeds in the system than the central backend seed from where clients can download pieces of contents by using the BitTorrent protocol (e.g. home PCs). The value of this ratio is between zero and one: zero means everybody downloads from the central server and one means everybody downloads from elsewhere. The actual ratio depends on the system, users' behavior and incentive systems used to encourage upload [9]. Most of the mobile phones today and in the near future might not be able to seed data because of certain limitations like limited or expensive bandwidth or limited battery capacity. Because

of these we expect only PCs at the first time to seed content.

The PC seeds helps to remove a part of the traffic of content download, but they cannot help to eliminate the management cost and they cannot eliminate most of the BitTorrent overheads (like torrent file or announce overheads). For simplifying the formulas we use the following upper estimation: PC seeds are removing only the content download but nothing from the content download overhead.

With these assumptions the cost of client server system (C_{CS}) and cost of Swarm system (C_{Swarm}) are the following:

$$C_{CS} = C_{cont} + C_{cont} \cdot OH_{CS} + C_{mgmt} \quad (10)$$

$$C_{Swarm} = C_{cont} \cdot (1 - S) + C_{cont} \cdot OH_{Swarm} + C_{mgmt} \quad (11)$$

The relative cost of Swarm (C_{rel}) is the Swarm cost divided by the client-server cost. If this number is for example 0.7 it means the cost of Swarm system is 70% compared to classical client-server system so it is 30% cheaper. Of course the relative cost could be higher than one if there are no seeds at all (because of higher overhead of BitTorrent). The relative cost is:

$$C_{rel} = C_{Swarm} / C_{CS} = \frac{(1 - S + OH_{Swarm} + \alpha)}{(1 + OH_{CS} + \alpha)} \quad (12)$$

$$\alpha = \frac{C_{mgmt}}{C_{cont}} \quad (13)$$

This alpha value depends on the actual service, and higher value results in higher relative cost. If the service provider would like to run several social networking functions on its content sharing solution then the management cost can be high (so the alpha parameter is high), because the users usually browse, chat, write messages or blog entries, but download only few large files (like pictures) using the BitTorrent protocol. For example in a social network solution like MySpace, the management cost is higher than in the Google photo album where users just browse the images and after that they can download the selected ones. The relative cost may depend on other parameters like the maximum picture sizes allowed to upload or the user interest in downloading the pictures. For example if users are downloading twice as many pictures (but the other parameters of the system remain the same) then the alpha reduces to its half.

Currently in our Swarm mobile client solution the management cost is low, because the mobile application communicates with the server via OPML and there is not any social networking like services.

Finally we introduce benefit graph (Fig. 11). We assume the management cost per content download cost is 0.1 for a high download system (photo album) and 0.5 for a social network system (like MySpace).

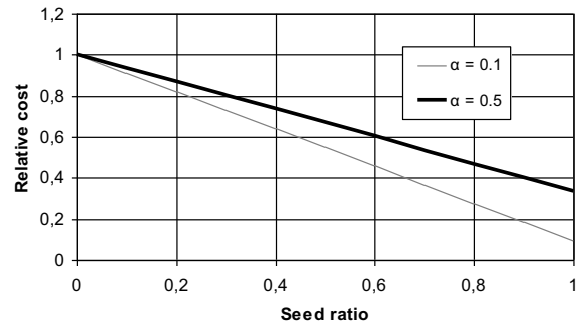


Figure 11. Relative cost of Swarm

We can see on Figure 11 that if the Seed ratio is higher - more and more home PCs are contributing in the network -, the relative cost of Swarm is lower, thus it is important to increase the number of home PCs in the network. In Section 2 we have discussed about a credit based solution [9] which can be easily applied in Swarm for this purpose.

D. Increase Swarm efficiency with local cooperation support

Besides encouraging home PCs to participate in the network we can increase the efficiency of Swarm by enabling local cooperation. In order to do so we do not have to modify Swarm protocol or any of the Swarm central elements, instead it is enough to implement a mechanism in the mobile Swarm clients. After implementing this mechanism the clients will use their short range radio (e.g. Bluetooth) to search for other nearby clients. If they find nearby clients they can check whether they have the same content by comparing the SHA1 identifiers of their torrents (Section 3).

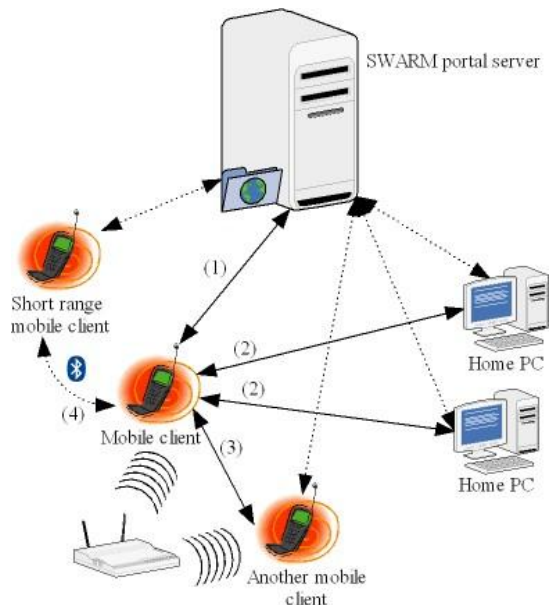


Figure 12. Applying local cooperation in Swarm

Figure 12 extends Figure 9 with local cooperation support. Steps 1-3 in Figure 12 are similar to Figure 9, but in Step 4 the phone who started downloading some content via Swarm uses its short range radio to search for other nearby phones and if it finds one suitable it initiates the download from it as well.

X. CONCLUSIONS AND FUTURE WORK

In this paper we proposed an efficient content sharing solution, called Swarm for mobile operators. Swarm supports PC and mobile phones as well and it considers also the special abilities of the mobile devices.

Swarm has hybrid architecture with a central unit, however the content distribution itself goes via BitTorrent, which is one the most advanced P2P protocol for content sharing nowadays. The members of the Swarm architecture are the Swarm server with a BitTorrent backend and portal functions, mobile clients and home PCs. One of the objectives of home PCs are to help in the content distribution. In Section 2 we have discussed about a credit based solution in order to encourage home PCs to participate in the network.

We have discussed also about local cooperation function which we have already implemented in our Symbian based BitTorrent client, called SymTorrent. With this functionality SymTorrent is able to search for nearby phones via short range radio in order to find more appropriate peers for the current download process. By enabling local cooperation in Swarm we can increase its efficiency further.

Swarm requires less storage capacity, energy consumption and processing power on the service provider side comparing to a client-server solution, thus the service operator can provide Swarm services on lower prices.

The main advantage of this solution is the cost efficiency for service providers, because the produced traffic distributes in the network and significant part of the infrastructure and operation cost is handled by the Swarm client applications. Swarm implements several features to enhance the user experience on mobile devices when it comes to content sharing. Among those users can find features such as easy search, browsing directories and "one click" content publish directly from the built-in camera and gallery applications. Naturally, these user interface elements can be implemented in a traditional client-server based solution too. From user experience point of view, Swarm made a special effort in hiding all the complexity that comes with the hybrid architecture and BitTorrent technology. In practice, this means that user does not realize the complexity of the underlying system, what is more he or she gets the impression that the service is implemented with the traditional client-server approach.

Future work will be to implement P2P credit system into Swarm in order to encourage home PCs to participate in the content distribution. Furthermore, we plan to

prepare additional measurements on the performance of the reference implementation of the proposed Swarm architecture and compare the results with existing client-server systems.

In this paper we have not analyzed mobile clients from the energy consumption point of view, however in the future this will be important if the mobile phones use the network more intensively.

The architecture of Swarm is suitable for social networks, because central unit of Swarm can store the data of the social network and the relations between users. Furthermore mobile clients are ideal for handling presence functions. By extending Swarm with social networking capabilities we can increase the functionality of Swarm with additional features like:

- 1) Share contents only to selected persons.
- 2) Provide higher bandwidth to my friends.

This way we also plan to investigate how social network related functions can increase the usability of Swarm.

ACKNOWLEDGMENT

We thank Szabolcs Fodor from Nokia Siemens Networks for supporting the Swarm research project.

REFERENCES

- [1] BitTorrent specification, Oct. 13, 2008. [Online]. Available: <http://wiki.theory.org/BitTorrentSpecification>
- [2] Fitzek, Frank H.P.; Katz, Marcos D., "Cognitive Wireless Networks", ISBN: 978-1-4020-5978-0. Springer, 2007
- [3] J. Pouwelse, P. Garbacki, D. Epema, H. Sips, "The BitTorrent p2p file-sharing system: Measurements and analysis", IPTPS'05. *4th Int. Workshop on Peer-To-Peer Systems 2006*, Ithaca, New York, USA
- [4] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-like Systems", IMC'05. *ACM SIGCOMM Internet Measurement Conference 2005*, New Orleans, LA
- [5] N. Andrade, M. Mowbray, A. Lima, G. Wagner, M. Ripeanu, "Influences on Cooperation in BitTorrent Communities", *ACM SIGCOMM workshop on Economics of peer-to-peer systems 2005*, Philadelphia, Pennsylvania, USA
- [6] M. Barbera, A. Lombardo, G. Schembra, M. Tribastone, "A Markov Model of a Freerider in a BitTorrent P2P Network", *GLOBECOM'05. Global Telecommunications Conference 2005*, DIIT, Catania Univ., Italy
- [7] M. Liy, J. Yuy, J. Wu, "Free-Riding on BitTorrent-like Peer-to-Peer File Sharing Systems: Modeling Analysis and Improvement", *ICPPW 2007, International Conference on Parallel Processing Workshops 2007*, XiAn, China
- [8] T. Locher, P. Moor, S. Schmid, R. Wattenhofer, "Free Riding in BitTorrent is Cheap", *In HotNets, 2006*
- [9] O. Karonen, J.K. Nurminen, "Cooperation Incentives and Enablers for Wireless Peers in Heterogeneous Networks", *ICC Workshop 2008*
- [10] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, "Free-riding in BitTorrent Networks with the Large View Exploit," in *Proc. of IPTPS'07*, Bellevue, WA, February, 2007
- [11] J.K. Nurminen and J. Nöyranen, "Energy-Consumption in Mobile Peer-to-Peer – Quantitative Results from File

- Sharing," in 5th IEEE Consumer Communications & Networking Conference.
- [12] C. D. Carothers, R. LaFortune, W. D. Smith, and M. Gilder, "A Case Study In Modeling Large-Scale Peer-To-Peer File-Sharing Networks Using Discrete-Event Simulation," in Proceedings of the 2006 European of Modeling and Simulation Symposium which is part of the I3M Multiconference), Barcelona, Spain, October, 2006.
- [13] I. Kelényi, G. Csúcs, B. Forstner, H. Charaf, "Peer-to-Peer File Sharing for Mobile Devices", *In Mobile Phone Programming: Application to Wireless Networks*; F. Fitzek, F. Reichert Eds.; ISBN: 978-1-4020-5968-1. Springer, 2007
- [14] B. Molnár, B. Forstner, I. Kelényi, "Symella 1.40", Budapest University of Technology and Economics. Nov. 19, 2007. [Online]. Available: <http://symella.aut.bme.hu>
- [15] I. Kelényi, P. Ekler, Zs. Pszota, "SymTorrent 1.30", Budapest University of Technology and Economics. Nov. 19, 2007. [Online]. Available: <http://symtorrent.aut.bme.hu>
- [16] P. Ekler, J. K. Nurminen, A. J. Kiss "Experiences of implementing BitTorrent on Java ME platform", CCNC'08. *1st IEEE International Peer-to-Peer for Handheld Devices Workshop 2008*, Las Vegas, USA, to be published
- [17] G. P. Perrucci, F. H. P. Fitzek, M. V. Petersen, "Energy Saving Aspects Exploiting Heterogeneous Wireless Networks". In *Heterogeneous Wireless Access Networks*: Ekram Hossain Ed.; ISBN: 978-0-387-09776-3, Springer, 2009
- [18] OPML specification, Oct. 15, 2008. [Online]. Available: <http://www.opml.org/>
- [19] RSS specification, Oct. 15, 2008. [Online]. Available: <http://www.rss-specifications.com/>
- [20] Official BitTorrent client, Oct. 15, 2008. [Online]. Available: <http://www.bittorrent.com/>

Péter Ekler was born on May 29, 1984, in Zalaegerszeg, Hungary. In 2007, he received the Master's degree in technical informatics from Budapest University of Technology and Economics where he is currently a PhD student. His field of

research is social and peer-to-peer networks in mobile environments and modeling network protocols.

Imre Kelényi was born on June 25, 1983, in Budapest, Hungary. In 2007, he received the Master's degree in technical informatics from Budapest University of Technology and Economics. Currently he is a PhD student at Budapest University of Technology, preparing his thesis. His field of research is peer-to-peer networks and energy efficient mobile systems.

István Dévai was born on June 1, 1982, in Budapest, Hungary. In 2006, he received the Master's degree in computer engineering from the Budapest University of Technology, where he is currently a PhD student. His research areas include distributed, highly scalable applications based on the Java platform.

Balázs Bakos was born on November 26, 1973, in Keszthely, Hungary. In 1997 he received the Master's degree in Computer Science from the Budapest University of Technology and Economics. In 1997 he has joined Software and Applications Technologies laboratory at Nokia Research Center Finland as research engineer in the field of telecommunication workstation applications. Starting from 1999 he is based in Budapest and held various Research Manager positions at Nokia and Nokia Siemens Networks. His current research interests include peer-to-peer, social networking and rich Internet applications.

Attila Kiss was born on December 23, 1975, in Dombóvár, Hungary. In 1999 he received the Master's degree in engineering and computer science from the Technical University of Budapest, where he remained as a PhD correspondence student until 2002. In 1999 he has joined to Nokia Research Center in Nokia. Recently he is working at Nokia Siemens Networks as Senior Research Engineer.