Analyzing the Concept of Involving Low End Devices in a Cooperative Network

Péter Ekler
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
peter.ekler@aut.bme.hu

Hassan Charaf
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
hassan.charaf@aut.bme.hu

Abstract—**The development of hardware and software of low end mobile phones and even simple media players will increasingly allow different type of content to be consumed on them creating pressure for efficient ways to also access the content. Since efficient, scalable distribution of multimedia content is a key strength of peer-to-peer technologies, bringing that technology to the player devices seems attractive. This way low end devices would be able to join and cooperate in an existing peer-to-peer network. The paper introduces a complete J2ME based BitTorrent solution for mainstream phones, called MobTorrent which supports both downloading and uploading. We analyze the key elements of MobTorrent and introduce issues related to the limited capabilities of low end devices. The performance of the complete application is evaluated by speed measurements. In general handheld devices have small batteries, thus the energy consumption is a key property of applications on low end devices. The paper also discusses the possibility of implementing Bluetooth based local cooperation in MobTorrent and the key factors of how to bring peer-to-peer technology to simple media player devices.**

Key words—BitTorrent; Low end device;, Cooperative networks

I. INTRODUCTION

As the capabilities of handheld devices increase, the range of applications that they are able to handle grows. An interesting set of applications for handheld devices are peer-to-peer (P2P) applications. Listening to music has long been a favorite use of handheld devices with devices ranging from Walkman type C-cassette players to modern MP3 players. The current digital convergence devices such as mobile phones are also increasingly used as music players. The consumption of video and other multimedia content besides music on handheld devices is also increasing.

Since efficient, scalable distribution of multimedia content is a key strength of P2P technologies, bringing that technology to the player devices seems attractive. In that way the multimedia content would be found, accessed, and played with the same handheld device. Using a PC as an intermediary would not be needed.

The first experimental steps towards this direction have already been taken with the implementations of popular content sharing protocols, Gnutella and BitTorrent, for mobile phones [1]. The applications, Symella [2] and SymTorrent [3], are available in source code. Mobile phones are an easy target platform, since their essence is connectivity which is increasingly complemented with different kinds of applications, in our case, with multimedia players and programs.

However, the available Gnutella and BitTorrent applications are implemented on Symbian platform, which limits their use to a subset of mobile devices and especially to the high end models with a fair number of computation resources available.

In this research, we want to investigate how to bring peer-to-peer content sharing to low end handheld devices. Our theoretical target is not only mass market mobile phones but also other handheld devices. For instance, if a music player were equipped with wireless access, could it work as a peer in a content sharing network? What are the challenges of bringing peer-to-peer content sharing to low end devices? What are the requirements for the device capabilities?

To answer these questions, we have experimented with implementing a BitTorrent client called MobTorrent for J2ME platform (Java Platform, Micro Edition). MobTorrent is a complete BitTorrent implementation it supports both downloading and uploading.

Most of the mobile device manufacturers support J2ME applications on their phones. The first reason for this is that there are already several popular applications for this platform which people want to use on their devices. The second cause is that the platform is very reliable: the signing procedure of the applications protects the users against unsafe applications.

The rest of the paper is organized as follows. Section 2 describes related work in the area of BitTorrent performance and existing J2ME peer-to-peer implementations. Section 3 investigates the MobTorrent from the viewpoint of the applied technology. Section 4 introduces measurements results about the performance of MobTorrent. Section 5 discusses the possibility of porting BitTorrent to player devices. Section 6 concludes the paper and proposes issues for further research.

II. RELATED WORK

One of the most popular P2P protocol is the BitTorrent. Despite its popularity, the actual behavior of these systems over prolonged periods of time is still poorly understood. Pouwelse et al. [4] presented a detailed measurement study over a period of eight months of BitTorrent. They presented 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 uploaded content. The results were 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 have 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 %).

While it is well-known that BitTorrent is vulnerable to selfish behavior, Locher et al. [5] demonstrated that even entire files can be downloaded without reciprocating at all in BitTorrent. To this end, they presented 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 (peers who only share the content). They also illustrated how peers in a swarm react to various sophisticated attacks.

Wang et al. [6] investigate issues related to the development of wireless P2P games with J2ME. Their work is mainly focused on the Bluetooth communication protocol and APIs but they briefly touch some other issues as well.

JXME [7], the JXTA [8] Java Micro Edition, provides a JXTA compatible platform on resource constrained devices. JXTA is a technology to create peer-to-peer applications based on Java. JXME has been an influential platform that has been used by a number of other researchers. Nützel and Kubek [9] discuss the development of a mobile extension to the HotPotato music distribution system using JXME. Bisignano et al. [10] have analyzed the use of JXME on handheld devices in the MANET context. Andersen and Torabi [11] propose a framework that is able to optimally choose an implementation matching the needs of an application. They demonstrate their system with a simple chat application running on JXME.

In the last two decades enormous efforts have been devoted to developing wireless communication technologies. Once affordable only to specific niche markets, these wireless communications are rapidly becoming everyones mainstream source of connectivity. In [12], the authors are introducing different concepts of cognitive and cooperative networks. Peer-to-peer networks also belong to this category. The authors are introducing a P2P based information retrieval system.

A key difference between previous research and our work is that our main goal is to allow low end mobile phones to join to a large, already existing, cooperative content distribution network. It allows us to investigate and measure the abilities of these devices and estimate what the requirements of this technology are.

III. J2ME Based BitTorrent Solution

J2ME is the platform of low end mobile devices. It is also available on mass market mobile phones from different manufacturers. Bringing the BitTorrent peer-to-peer technology to the mainstream phones is the next step towards the simpler player devices. In this section, we introduce the technologies that MobTorrent uses in order to determine the technological requirements of BitTorrent or other similar type of peer-to-peer protocols.

J2ME is ideal for this analysis because of its architecture. It is based on three main elements: configurations, profiles and optional packages.

(i) Configurations describe the capabilities of the virtual machine and provide the basic set of libraries for a broad range of devices. The configuration, targeting resource-constrained devices such as mobile phones is called Connected Limited Device Configuration (CLDC).

(ii) For defining a higher-level API, the J2ME platform specifies profiles on top of the configurations. The combination of Mobile Information Device Profile (MIDP) with CLDC is widely used to provide a complete Java application environment for handheld devices.

(iii) If we want to use other technology specific APIs in our application, we can import different kinds of optional packages which can be found in different JSRs [13] (Java Specification Request).

By introducing the main architectural elements of MobTorrent we show what JSRs and algorithms it uses.

A. Networking

The essence of BitTorrent is that content (one or more files) is downloaded in multiple pieces. The different pieces are downloaded from separate peers (if possible) in parallel which makes the aggregate download speed much higher than downloading from a single bandwidth limited peer only. Downloadable content is described by a torrent file, which contains the address of the tracker and the hash values of the content pieces. The tracker maintains a list of peers that are working on the download (or share) of the same content. MobTorrent can open and interpret the torrent files. After it connects and registers to the tracker, which sends back addresses of several peers that are able to serve the content. Then MobTorrent sends piece request messages to these peers which respond by starting to upload the content to us. While a peer is downloading content it can also upload the already received pieces to others.

The BitTorrent protocol [14] uses HTTP connections for communicating with the tracker and TCP connections for the download and upload procedure from/to the other peers. These communication protocols are supported in MIDP 2.0 [15] which supports sockets and also datagram connections.

A previous paper [16] has introduced some limitations from the network handling point of view of MobTorrent. The MobTorrent in that case was only able to download, thus, it was a limited implementation of BitTorrent. It demonstrated on the popular Nokia S40 platform for low end mobile phones that there are limitations about how many connections can be maintained at the same time. This number was 9 for TCP and HTTP connections together. Experiences in [16] show that downloading via 9 parallel connections is adequate in many cases. Section 4 introduces how the upload ability changes these numbers.

This shows that P2P applications have different platform requirements from other types of applications and that they raise problems that are not experienced by other ones.

B. File handling

MobTorrent needs to store the downloaded content on the file system of the device. Nowadays most of the mainstream mobile phones and even simple media players have enough memory capacity for music and even for large multimedia files. The J2ME has the ability to handle files with the File Connection Optional Package (FCOP). It is one of two optional packages defined by JSR-75 [13] which is supported by most phones.

File handling in J2ME differs from general file handling solutions. The main difference is that, on J2ME, we can access the file only via input and output streams which are slower than the other file handling implementations.

File handling is one of the reasons why the performance of MobTorrent is slower if we allow both download and upload. The reason for that is the following. If we allow both directions, then the application uses the file system more often, because it must also read the data from it before uploading a piece.

C. Processing power

While we are developing for simple mobile devices, we also have to consider that they have limited processing power. BitTorrent protocol uses the Secure Hash Algorithm version 1.0 (SHA1) for checking the consistency of the downloaded content. This algorithm requires a lot of processing time on J2ME. When the application has downloaded a piece it has to calculate the hash value of the content and compare it to the value in the original torrent file. This mechanism allows us to find and avoid the faulty or badly downloaded pieces. Experiences shows that one piece out of 100 can be bad which means without hash checking the probability of error is quite high even for medium-sized content.

SHA1 algorithm is implemented in the Security and Trust Services API (SATSA), which can be found in the JSR-177 [13], but this JSR is not widely available on the simple devices. Thus, we implemented our own SHA1 algorithm to avoid using this JSR. According to our measurements on Nokia N91 devices the processing time of SHA1 calculation of one piece is the same in our implementation and the one in JSR-177.

Boland and Fisher [17] introduced and compared different hash algorithms, the result was that the SHA-1 is more complex and slower than the other algorithms, but presents a more robust solution than the other hashing algorithms considered. Changing the SHA-1 in our solution to other hash checking algorithm is not possible, because in that way we would lose the compatibility with the existing BitTorrent community.

In BitTorrent, we download the content in separate pieces. A general piece size is 64 KB and a piece is downloaded in separate blocks; in this case the block size is 16 KB. In SymTorrent, after the whole piece is downloaded it reads it out from the file system and does the hash calculation for it. This solution in MobTorrent was slow. Investigating this issue, we measured the speed of reading a 64KB size of piece from different position of a file and calculating its hash value.

TABLE I. PIECE READ AND HASH CALCULATION PERFORMANCE

	Begin		**Middle**		**End**	
millisecond	Read	Hash	Read	Hash	Read	Hash
file1 2 MB	1097	978	4398	978	7103	978
file2 7 MB	1109	978	10847	978	20210	978
file3 10 MB	1150	978	14478	978	27214	978

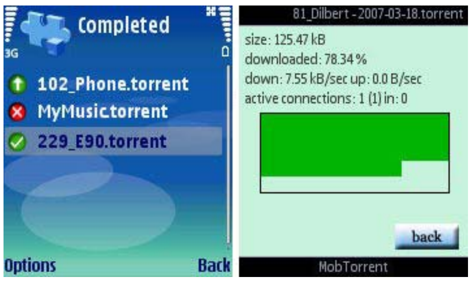
Table 1 illustrates how much time it takes to read the piece from a file and calculate the hash value, the values are in milliseconds. The measurements were made on a Nokia N93 device but the rates were the same on other devices, thus, they are not presented here. We chose file sizes which are typical on mobile devices like for mp3 and other multimedia files. We also measured the piece reading from different position of the file. In Table 1, we can see that not the hash calculation itself is the bottleneck but reading the piece from the file system. The reason for that is the file handling solution on J2ME. We can read from a file only via an InputStream and we are not allowed to seek in the file, we can only read it until we reach the right position.

In order to avoid this performance penalty, we calculate the hash value incrementally when a block of a piece arrives, thus, we do not have to read it from the file system after the whole piece arrived. To achieve this, we must query the blocks of a piece in the right order, which is feasible, because if a peer reports us that it has the whole piece, then we can query them in the right order.

D. User Interface

Using the limited screen space of the mobile device efficiently is important. Since MobTorrent performs most of its activities without user interaction the requirements for the user interface are not very severe. The screenshots in Figure 1 introduces the user interface of the application.

The screenshots were made on the phone. They illustrate the list of the torrents and the download state of a selected torrent.



size: 125.47 kB
downloaded: 78.34 %
down: 7.55 kB/sec up: 0.0 B/sec
active connections: 1 (1) in: 0

Figure 1. MobTorrent screenshots

IV. Mesurements and Results

The download performance of MobTorrent was already introduced in a previous paper from us [16] but in that version upload functionality was not implemented. The new version of MobTorrent is capable of both downloading and uploading.

The upload functionality exhibited unexpected performance deceleration. The reason for that is the previously introduced file handling problem of the J2ME, which we could avoid in the download but not in the upload. During the upload other peers request pieces from our client, which pieces we have to read even from the end of the file which is very slow, as it was shown before thus it decreases the performance of the whole application.

TABLE II. Download Speed (KB/sec) with Upload Functionality

	1 peer	2 peers	6 peers
MobTorrent	65	82	95
MobTorrent Up	18	20	21

Table 2 introduces how the download speed decreased when we allowed the upload functionality in MobTorrent. These measurements were made on a Nokia N93 device on WLAN network but the rates on other devices (Nokia 6280, Nokia 6230i) and on 3G network were the same.

MobTorrent Up in the table means that in that case one peer were downloading from our client, while we were downloading a 2 MB large file. The numbers in that row illustrate our download speed. The upload speed which the MobTorrent was able to provide was 17 KB/sec. When more peers were downloading from our device the upload speed did not increased even if they were served in separate threads, because the bottleneck was reading the data from the file system.

In general we can say that when other peers are downloading from us then our download speed decreases and the bottleneck is not the previously mentioned socket problem, that the system can maintain only 9 connections at the same time, but the file handling of the J2ME.

However, if there are no incoming connections, the previously introduced download performance is reasonably good, and, with the upload functionality, MobTorrent is able to share contents, thus it can become a full member of the community.

V. Deploying BitTorrent to Player Devices

By bringing the BitTorrent technology to the J2ME platform, we illustrated that it is possible to involve low end devices in a peer-to-peer cooperative network. If we analyze the architectural elements of MobTorrent, we can determine what technologies BitTorrent needs, which is important if we examine the possibility of bringing the technology to simple media player devices. The relevant technologies are the following:

(i) Network support: the target device has to be able to use a certain network connection in order to communicate with other members of the community. WLAN network support is becoming very popular in mobile devices but they are still not relevant in simple player devices. However there are for example digital picture frames which support WLAN connections.

(ii) File system: while BitTorrent is a content sharing system, it is important for the player device to handle the file system in an efficient way. We demonstrated with measurements that how slow file handling can decrease the performance of the application.

(iii) Multithreading: in a peer-to-peer protocol, the system must maintain several connections at the same time, thus it is necessary to support somehow multiple threads.

(iv) Processing power: BitTorrent uses SHA-1 algorithm and it also does other calculations, thus it is important for the player device to have a reasonably good processing power.

(v) User interface: if we want to bring the BitTorrent technology to player devices our goal is to hide from the user that a complex peer-to-peer protocol runs in the background but still the application needs a simple user interface for example for searching for the content to download, browsing downloadable contents, checking the status, etc.

While we discussed network connections, we did not mention the Bluetooth, however, it is very popular and widely available on handheld devices. There is already an experimental Bluetooth-based implementation of BitTorrent based on Symbian platform [3]. SymTorrent with local cooperation is an experimental project which aims at achieving smaller energy consumption by downloading over Bluetooth if the phones are in close proximity to each other.

The following equations illustrate a high level calculation, about how we can estimate the performance (speed of the content transfer) of a torrent client on low end devices (P), considering the previously mentioned factors and also the possibility of Bluetooth support (P_{blue}). The formulas are based on our experiences according to MobTorrent but it is worth to consider it in connection with other type of low end devices such as simple media players. It is hard to measure the concrete values in the equations, but our goal with it is more to illustrate each of the factors and their roles.

$$P = P_{poss} * \frac{1}{D_s + D_f + D_c + D_m} + P_{blue} \quad (1)$$

$$P_{blue} = P_{possBlue} * \frac{1}{D_f + D_c + D_m} \quad (2)$$

P_{poss} is the theoretical maximum performance of the application, which depends on the hardware of the specific device (bandwidth, processor, etc.). $P_{possBlue}$ is the theoretical maximum performance which could be achieved with the Bluetooth based local cooperation. D_s is the factor how the previously mentioned socket limitation decreases the performance, D_f is the decrease factor of the file handling on J2ME, D_c means how the processing power of the device slows down performance and D_m is the decrease factor which derives from the multi threading and multiple peer handling which is key property of peer-to-peer applications.

Bringing the Bluetooth based local cooperation on low end devices seems attractive, because those devices have even smaller battery, and the energy consumption is a key property of the applications. J2ME supports Bluetooth connections via the JSR-82 [13] thus implementing local cooperation is the next step for increasing the efficiency of MobTorrent.

VI. Conclusion and Future Work

In this paper, we have introduced MobTorrent, the first full-featured and complete BitTorrent client for J2ME platform. It supports downloading multiple torrent files at the same time, is capable of both downloading and uploading. This way low end mobile devices are able to become real members of the BitTorrent content distribution community.

Performance measurements show that using low end mobile devices for peer-to-peer content sharing with existing communities is feasible. Measurements also illustrate how the upload functionality decreases the performance of the application and how the file handling of J2ME is responsible for this deceleration.

In this paper, we have also analyzed the possibility of bringing BitTorrent to even simple media player devices. By analyzing the MobTorrent, we can say that the minimal requirement of a BitTorrent type of peer-to-peer application is that the platform of the device should support network handling, file handling methods, multithreading, reasonable processing power and minimal user interface. From the J2ME point of view, these requirements can be found in the MIDP 2.0 and in JSR-75.

One of the advantages of having BitTorrent on simple mobile phones is that in that way we do not need PC for downloading and we have the possibility of the on-the-move download. Another advantage is that it is very comfortable to download directly to the mobile phone if the user consumes the content on the device. An interesting detail is that downloading with a PC creates noise because of the cooling system and disk usage, while with a mobile phone the content distribution is completely silent.

In this paper, we have not analyzed mobile devices from the energy consumption point of view, however, in the future this will be important if the devices use the network more intensively. Future work will be to implement the Bluetooth-based local cooperation on J2ME platform. Thus, we could analyze the energy consumption of the devices deeply.

Acknowledgment

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

References

[1] 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

[2] B. Molnár, B. Forstner, I. Kelényi, “Symella 1.40”, Budapest University of Technology and Economics. Dec. 18, 2007. [Online]. Available: http://symella.aut.bme.hu

[3] I. Kelényi, P. Ekler, Zs. Pszota, “SymTorrent 1.30”, Budapest University of Technology and Economics. Dec. 18, 2007. [Online]. Available: http://symtorrent.aut.bme.hu

[4] 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

[5] T. Locher, P. Moor, S. Schmid, R. Wattenhofer, “Free Riding in BitTorrent is Cheap”, *In HotNets*, 2006

[6] A. Inge Wang, M. Sars Norum, C. Wolf Lund, "Issues related to Development of Wireless Peer-to-Peer Games in J2ME", *Telecommunications 2006*. AICT-ICIW '06. *Int. Conf. on Internet and Web Applications and Services/Advanced Int. Conf. on*, vol., no., 19-25 Feb. 2006, pp. 115-115

[7] JXTA Java Micro Edition overview, Dec. 18, 2007. [Online]. Available: https://jxta-jxme.dev.java.net/

[8] JXTA description, Dec. 27, 2007. [Online]. Available: https://jxta.dev.java.net/

[9] J. Nutzel, M. Kubek, "A Mobile Peer-to-Peer Application for Distributed Recommendation and Re-Sale of Music", AXMEDIS '06. *Automated Production of Cross Media Content for Multi-Channel Distribution, 2006. Second Int. Conf. on*, vol., no., Dec. 2006, pp.93-98

[10] M. Bisignano, G. Di Modica, O. Tomarchio, “A JXTA compliant framework for mobile handheld devices in ad-hoc networks”, *Computers and Communications, 2005*. ISCC 2005. *Proceedings. 10th IEEE Symposium on*, vol., no., June 2005, pp. 582-587, 27-30

[11] A. M. Andersen, T. Torabi, "A holistic framework for mobile environments", *Software Engineering Conf.*, 2006. Australian, vol., no., April 2006, pp. 7 pp.-, 18-21

[12] B. Forstner, G. Csúcs, I. Kelényi, H. Charaf, “Peer-to-Peer Information Retrieval Based on Fields of Interest”, *In Cognitive Wireless Networks*; F. H. P. Fitzek, M. D. Katz; ISBN: 978-1-4020-5978-0. Springer, 2007

[13] Java Specification Request overview, Dec. 18, 2007. [Online]. Available: http://www.jcp.org/en/jsr/overview

[14] BitTorrent specification, Dec. 18, 2007. [Online]. Available: http://wiki.theory.org/BitTorrentSpecification

[15] Mobile Information Device Profile 2 description, Dec. 18, 2007. [Online]. Available: http://developers.sun.com/mobility/midp/articles/midp2network

[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] Boland, T. and Fisher, G. (2000) "Selection Of Hashing Algorithms". [Online]. Available: http://www.nsrl.nist.gov/documents/hash-selection.doc